

# Diseño de aplicaciones con JBuilder®

## JBuilder® X

Borland Software Corporation  
100 Enterprise Way  
Scotts Valley, California 95066-3249  
[www.borland.com](http://www.borland.com)

Consulte el archivo `deploy.html` ubicado en el directorio `redist` de su producto JBuilder para acceder a una lista completa de archivos que puede distribuir de acuerdo con su Declaración de licencia y Garantía limitada de JBuilder.

Borland Software Corporation puede tener patentes y/o solicitudes de patente pendientes en lo relativo a los asuntos tratados en este documento. Consulte el CD del producto o el cuadro de diálogo Acerca de para ver la lista de patentes aplicables. La entrega de este documento no supone la cesión de ninguna licencia sobre estas patentes.

Copyright © 1997–2004 Borland Software Corporation. Reservados todos los derechos. Todas las marcas y nombres de productos Borland son marcas comerciales o registradas de Borland Software Corporation en los Estados Unidos y en otros países. Las demás marcas pertenecen a sus respectivos propietarios.

Para conocer las condiciones y limitaciones de responsabilidad de terceros, consulte las Notas de la versión en el CD de JBuilder.

Impreso en los EEUU.

JBE001AWW21005designui 5E4R1103  
0304050607-9 8 7 6 5 4 3 2 1  
PDF

# Índice de materias

## Capítulo 1

### Diseño visual en JBuilder 1-1

Requisitos para poder diseñar visualmente	
una clase . . . . .	1-1
Comienzo por medio de asistentes . . . . .	1-2
Los JavaBeans . . . . .	1-3
Los contenedores . . . . .	1-3
Tipos de contenedores . . . . .	1-4
Las bibliotecas de componentes . . . . .	1-5
Convenciones de la documentación . . . . .	1-5
Asistencia y recursos para desarrolladores . . . . .	1-7
El servicio de asistencia técnica de	
Borland . . . . .	1-7
Recursos en línea . . . . .	1-8
World Wide Web . . . . .	1-8
Grupos de noticias de Borland . . . . .	1-8
Usenet, grupos de noticias . . . . .	1-8
Información sobre errores . . . . .	1-9

## Capítulo 2

### Introducción al diseñador 2-1

Utilización del diseñador . . . . .	2-1
La superficie de diseño . . . . .	2-3
La paleta de componentes . . . . .	2-4
Utilización del botón Selección de	
Beans . . . . .	2-4
El Inspector . . . . .	2-6
El árbol de componentes . . . . .	2-6
Categorías de diseñadores . . . . .	2-7
Diseñador de interfaces de usuario . . . . .	2-7
Diseñador de menús . . . . .	2-7
Diseñador de acceso a datos . . . . .	2-8
Diseñador por defecto . . . . .	2-8
Combinaciones de teclas del diseñador . . . . .	2-8

## Capítulo 3

### El árbol de componentes y el Inspector 3-1

Utilización del árbol de componentes . . . . .	3-1
Apertura de tipos concretos de	
diseñadores . . . . .	3-3
Adición de componentes . . . . .	3-3
Corte, copia y pegado de componentes . . . . .	3-4
Eliminación de componentes . . . . .	3-5
Utilización de Deshacer y Rehacer . . . . .	3-5

Cambio de nombre de los componentes . . . . .	3-5
Desplazamiento de componentes . . . . .	3-6
Presentación de los nombres de clase de	
los componentes . . . . .	3-6
Iconos del árbol de componentes . . . . .	3-6
Apertura del Inspector . . . . .	3-6
Resaltado de los valores de las	
propiedades . . . . .	3-7
Conversión de las propiedades en	
variables de clase . . . . .	3-8
Definición de la exposición de las	
propiedades . . . . .	3-8
Definición de los valores de las	
propiedades . . . . .	3-8
Definición de propiedades	
compartidas . . . . .	3-9
Definición de propiedades con la lista	
desplegable vacía . . . . .	3-10
El Inspector . . . . .	3-10

## Capítulo 4

### Tratamiento de sucesos 4-1

Vinculación de código de tratamiento de	
sucesos . . . . .	4-2
Creación del manejador de sucesos por	
defecto . . . . .	4-2
Eliminación de manejadores de sucesos . . . . .	4-3
Conexión de controles y sucesos . . . . .	4-3
Adaptadores de sucesos estándar . . . . .	4-4
Adaptadores anónimos de clase interna . . . . .	4-5
Elección del estilo de manejador de	
sucesos . . . . .	4-6
Ejemplos: conexión y tratamiento de sucesos . . . . .	4-7
Presentación de un texto cuando se pulsa	
un botón . . . . .	4-7
Apertura de un cuadro de diálogo desde	
un elemento de menú . . . . .	4-8

## Capítulo 5

### Creación de interfaces de usuario 5-1

Selección de componentes en la interfaz de	
usuario . . . . .	5-2
Adición a contenedores anidados . . . . .	5-3

Desplazamiento y redimensionamiento de componentes . . . . .	5-3
Gestión del diseño . . . . .	5-5
Agrupación de componentes. . . . .	5-5
Adición de componentes para la generación de aplicaciones . . . . .	5-6
Menús. . . . .	5-6
Cuadros de diálogo . . . . .	5-7
Componentes de base de datos . . . . .	5-9
Cambio del aspecto . . . . .	5-10
Aspecto durante la ejecución . . . . .	5-10
Aspecto en la fase de diseño . . . . .	5-11
Comprobación de la interfaz de usuario durante la ejecución . . . . .	5-12

## Capítulo 6

### Diseño de menús 6-1

Apertura del diseñador de menús. . . . .	6-1
Terminología de menús . . . . .	6-2
Herramientas de diseño de menús . . . . .	6-3
Creación de menús . . . . .	6-4
Adición de elementos de menú . . . . .	6-5
Inserción y eliminación de menús y elementos de menú . . . . .	6-5
Inserción de separadores . . . . .	6-6
Definición de teclas aceleradoras . . . . .	6-6
Desactivación (atenuación) de elementos de menú . . . . .	6-6
Creación de elementos de menú que utilicen marcas de selección . . . . .	6-7
Creación de elementos de menú con botones de radio Swing . . . . .	6-7
Traslado de elementos de menú . . . . .	6-8
Creación de submenús . . . . .	6-9
Traslado de un menú a un submenú. . . . .	6-10
Vinculación de sucesos de menú al código . . . . .	6-10
Creación de menús emergentes . . . . .	6-11

## Capítulo 7

### Temas avanzados 7-1

Gestión de la paleta de componentes. . . . .	7-1
Cómo añadir un componente a la paleta de componentes . . . . .	7-2
Selección de una imagen para un botón de la paleta de componentes. . . . .	7-4
Cómo añadir un componente a la paleta de componentes . . . . .	7-5
Eliminación de una ficha o un componente de la paleta de componentes. . . . .	7-6

Reorganización de la paleta de componentes. . . . .	7-6
Manejo de beans rojos . . . . .	7-7
La serialización . . . . .	7-7
Serialización de componentes en JBuilder . . . . .	7-8
Serialización de un objeto this. . . . .	7-8
Utilización de personalizadores en el diseñador . . . . .	7-10
Modificación de beans mediante personalizadores. . . . .	7-10
Gestión de cadenas de conjunto de recursos . . . . .	7-11

## Capítulo 8

### Gestores de diseño 8-1

Acerca de los gestores de diseño . . . . .	8-1
Diseños null y XYLayout. . . . .	8-2
Conceptos básicos acerca de las propiedades de los diseños . . . . .	8-3
Conceptos básicos acerca de las restricciones de los diseños . . . . .	8-4
Ejemplos de propiedades y restricciones de diseños . . . . .	8-4
Diseños de contenedores. . . . .	8-5
Modificación de las propiedades de los diseños . . . . .	8-5
Modificación de restricciones de diseños de componentes. . . . .	8-6
Conceptos básicos acerca de las propiedades de tamaño. . . . .	8-6
Tamaño y ubicación de la ventana de la interfaz de usuario durante la ejecución. . . . .	8-7
Dimensionamiento automático de una ventana con pack() . . . . .	8-8
Cálculo de preferredSize (tamaño recomendado) para los contenedores . . . . .	8-8
Diseños portables . . . . .	8-8
XYLayout . . . . .	8-8
Definición explícita del tamaño del contenedor de la interfaz de usuario utilizando setSize(). . . . .	8-9
Preparación del tamaño de la interfaz de usuario para que pueda portarse a varias plataformas . . . . .	8-9
Colocación de una ventana en la pantalla . . . . .	8-10
Inclusión en el código de las llamadas a métodos de tamaño y posicionamiento . . . . .	8-10
Adición de gestores de diseño personalizados . . . . .	8-11
Diseños suministrados con JBuilder . . . . .	8-12



Paso 4: Añadir un cuadro de diálogo	
Selector de fuentes . . . . .	10-15
Definición del marco del cuadro de diálogo y las propiedades del título. . .	10-16
Creación de un suceso para lanzar el Selector de fuentes. . . . .	10-17
Paso 5: Vinculación de sucesos de elemento de menú al cuadro de diálogo Selector de fuentes .	10-18
Paso 6: Vinculación de sucesos de elementos de menú a JColorChooser . . .	10-20
Paso 7: Adición de un manejador a un suceso de menú para borrar el área de texto . . . . .	10-22
Paso 8: Añadir un cuadro de diálogo selector de archivos . . . . .	10-23
Internacionalización de componentes Swing . . . . .	10-23
Paso 9: Añadir código para leer texto de un archivo . . . . .	10-24
Paso 10: Adición de código a los elementos de menú para guardar un archivo . . . . .	10-27
Paso 11: Adición de código para comprobar si se ha modificado un archivo. . . . .	10-29
Paso 12: Activación de los botones de la barra de herramientas . . . . .	10-31
Especificación del texto de la ayuda inmediata del botón. . . . .	10-32
Creación de los sucesos de botón . . . . .	10-32
Creación de un método fileOpen() . . . . .	10-33
Creación de un método helpAbout() . . . . .	10-34
Paso 13: Asociación del tratamiento de sucesos con el área de texto . . . . .	10-35
Paso 14: Adición de un menú contextual al área de texto. . . . .	10-36
Paso 15: Visualización del nombre y del estado del archivo en el título de la ventana . . . . .	10-38
Paso 16: Distribución de la aplicación	
Editor de texto a un archivo JAR . . . . .	10-42
Aspectos generales . . . . .	10-42
Ejecución del Creador de compiladores .	10-43
Ejecución desde la línea de comandos de la aplicación distribuida . . . . .	10-49
Modificación del archivo JAR y nueva prueba de la aplicación . . . . .	10-50

## Capítulo 11

### Tutorial: Creación de una interfaz de usuario con diseños anidados 11-1

Paso 1: Creación del proyecto de interfaz de usuario . . . . .	11-3
Utilización del Asistente para proyectos . .	11-3
Paso 2: Generación de los archivos fuente de la aplicación . . . . .	11-4
Utilización del Asistente para aplicaciones	11-4
Paso 3: Modificación del diseño del panel de contenido . . . . .	11-7
Paso 4: Cómo añadir los paneles principales . . . . .	11-9
Paso 5: Creación de barras de herramientas . . . . .	11-12
Paso 6: Cómo añadir botones a las barras de herramientas . . . . .	11-13
Paso 7: Cómo añadir componentes al panel central . . . . .	11-16
Paso 8: Creación de una barra de estado .	11-17
Paso 9: Conversión a diseños portables . .	11-18
Paso 10: Completar su diseño . . . . .	11-21

## Capítulo 12

### Tutorial: Creación de diseños GridBagLayout en JBuilder 12-1

Utilización de GridBagLayout . . . . .	12-3
Definición de GridBagLayout . . . . .	12-3
Área de visualización de los componentes. . . . .	12-4
Definición de las restricciones GridBagConstraints . . . . .	12-6
Dificultades de GridBagLayout . . . . .	12-7
Ventajas de GridBagLayout . . . . .	12-8
Simplificación de GridBagLayout . . . . .	12-8
Comienzo mediante un boceto en papel . . . . .	12-9
Utilización de paneles y diseños anidados . . . . .	12-12
Utilización del diseñador visual de JBuilder . . . . .	12-13
Creación de una interfaz de prototipo en XYLayout . . . . .	12-15
Creación de diseños GridBagLayout en JBuilder . . . . .	12-17
Acerca del diseño . . . . .	12-17
Paso 1: Diseñar la estructura . . . . .	12-18

Paso 2: Crear un proyecto para este tutorial . . . . .	12-23	Eliminación de los pesos y expansiones antes de efectuar los ajustes . . . . .	12-61
Paso 3: Añadir los componentes al contenedor . . . . .	12-23	Diseño visual del código GridBagLayout ya escrito . . . . .	12-62
Adición del panel principal al marco de interfaz . . . . .	12-24	Diferencias en el código . . . . .	12-62
Creación del panel izquierdo y adición de componentes. . . . .	12-25	Modificación del código para que funcione en el diseñador . . . . .	12-62
Creación del panel derecho y adición de componentes. . . . .	12-28	Código generado por JBuilder en la segunda parte. . . . .	12-63
Creación del panel inferior y adición de componentes. . . . .	12-28	Otros recursos sobre GridBagLayout .	12-65
Paso 4: Convertir el panel exterior en GridBagLayout . . . . .	12-29	GridBagConstraints . . . . .	12-65
Paso 5: Convertir los paneles exteriores en GridBagLayout . . . . .	12-30	Ancla. . . . .	12-65
Paso 6: Convertir el panel inferior en GridLayout . . . . .	12-30	Expansión . . . . .	12-66
Paso 7: Efectuar los ajustes finales . . .	12-31	Encuadres . . . . .	12-66
Panel GridLayout . . . . .	12-32	gridwidth, gridheight . . . . .	12-67
Paneles superiores . . . . .	12-36	ipadx, ipady . . . . .	12-68
Conclusión . . . . .	12-41	gridx, gridy . . . . .	12-68
Sugerencias y técnicas . . . . .	12-42	weightx, weighty . . . . .	12-69
Definición de restricciones individuales en el diseñador . . . . .	12-42	Ejemplos de restricciones de Peso . . . .	12-70
Ancla . . . . .	12-42		
Expansión. . . . .	12-43		
Encuadres . . . . .	12-43		
gridwidth, gridheight . . . . .	12-44		
ipadx, ipady . . . . .	12-45		
gridx, gridy . . . . .	12-46		
weightx, weighty . . . . .	12-47		
Comportamiento de las restricciones de peso . . . . .	12-48		
Modificación de restricciones mediante el ratón . . . . .	12-51		
Arrastrar componentes a celdas vacías . . . . .	12-52		
Arrastrar componentes a celdas ocupadas . . . . .	12-53		
Arrastrar componentes grandes a celdas pequeñas . . . . .	12-56		
Arrastrar los tiradores de redimensionamiento negros a una celda adyacente vacía . . . . .	12-57		
Cómo añadir componentes . . . . .	12-59		
Sugerencias diversas . . . . .	12-61		
Vuelta a XYLayout si se deben hacer ajustes significativos . . . . .	12-61		

## Apéndice A Migración de archivos desde otros IDE de Java

VisualAge . . . . .	A-1
Forte . . . . .	A-2
VisualCafé . . . . .	A-2

## Índice

I-1

# Tutoriales

Creación de una interfaz de usuario sencilla . . .	9-1	Creación de diseños GridBagLayout en	
Creación de un editor de texto en Java . . . .	10-1	JBuilder . . . . .	12-1
Creación de una interfaz de usuario con			
diseños anidados . . . . .	11-1		





## Diseño visual en JBuilder

El diseñador de JBuilder permite crear y modificar con rapidez y eficacia los archivos diseñables visualmente. En esta documentación se tratan los cuatro aspectos visibles del diseñador: el árbol de componentes, la superficie de diseño, el Inspector y la paleta de componentes. También se explican paso a paso el diseño de una interfaz de usuario y la creación de menús, el árbol de componentes, la superficie de diseño, el Inspector y la paleta de componentes.

### Consulte

- “El Visualizador de aplicaciones” en *Introducción a JBuilder* si desea repasar el diseño y la terminología del Visualizador de aplicaciones.

## Requisitos para poder diseñar visualmente una clase

---

Para que un archivo se pueda diseñar visualmente, es necesario que:

- Sea un archivo `.java`.
- No contenga errores de sintaxis.
- Utilice un constructor público por defecto.
- Defina una clase cuyo nombre coincida con el nombre del archivo.
  - La clase definida no puede ser interna ni anónima.

Los archivos que cumplen los requisitos anteriores pueden diseñarse utilizando el árbol de componentes y el Inspector. De esta forma, puede diseñar visualmente una clase que no aparezca en la interfaz de usuario.

**Nota** JavaBeans cumple con estos requisitos. Todos estos requisitos también se cumplen si se crean los archivos utilizando los asistentes para aplicaciones, applets, marcos, paneles y cuadros de diálogo de JBuilder.

La primera vez que se añade un componente al diseño, las herramientas de diseño visual de JBuilder comprueban que:

- La clase cuenta con un constructor público por defecto.
- La clase tiene un método `jbInit()` privado.
  - Se realiza una llamada correctamente a este método `jbInit()` desde el constructor por defecto.

Si JBuilder no encuentra el método `jbInit()` en la clase que se diseña, lo añade. También realiza todas las importaciones que necesite el componente.

**Nota** Si la clase no cuenta con un constructor público por defecto, el componente aparece como un cuadro rojo en la vista Diseño.

**Importante** Si va a migrar archivos de otros IDE Java a JBuilder, puede que necesite modificar su código para que los diseñadores de JBuilder puedan trabajar con los archivos. Las herramientas de diseño visual de JBuilder son capaces de reconocer archivos VisualAge si cumplen los requisitos de los archivos diseñables visualmente.

### Consulte

- El [Apéndice A, “Migración de archivos desde otros IDE de Java”](#)
- [“Manejo de beans rojos” en la página 7-7](#)

## Comienzo por medio de asistentes

---

El primer paso del diseño de una interfaz de usuario con JBuilder es crear o abrir una clase de contenedor diseñable, como un `Marco` o un `Panel`.

Abra la galería de objetos seleccionando `Archivo|Nuevo`. En la galería de objetos hay varias fichas que permiten acceder a asistentes, con los cuales se pueden generar archivos diseñables visualmente como applets, aplicaciones y cuadros de diálogo.

Estos asistentes importan todos los paquetes necesarios. Para empezar a utilizar el diseñador basta con abrir el archivo contenedor y pulsar la ficha `Diseño` del panel de contenido.

**Nota** Para añadir marcos, paneles y cuadros de diálogo al proyecto, seleccione `Archivo|Nuevo` y elija el asistente apropiado en la galería de objetos.

# Los JavaBeans

---

Los JavaBeans son clases autocontenidas que no necesitan más clases para estar completas. Están diseñados de forma que se pueden personalizar y se comunican eficazmente entre sí, de forma que pueden interactuar sin problemas. Expliquémoslo con un ejemplo: una rueda está diseñada para girar alrededor de un eje central, y no necesita más mecanismos para ello. Las ruedas se pueden personalizar de forma que encajen debajo de un coche o dentro de una polea; de esta forma interactúan con otros componentes del diseño para realizar una función más amplia.

Los JavaBeans deben aceptar estas funciones:

<i>Introspección</i>	Permite analizar los beans.
<i>Personalización</i>	Permite ajustar a las necesidades el aspecto y el comportamiento de los beans.
<i>Sucesos</i>	Permite que los beans se comuniquen.
<i>Persistencia</i>	Permite guardar el estado de ejecución de los beans.

Además, los JavaBeans deben tener una clase `BeanInfo`. `BeanInfo` describe su componente a las herramientas de diseño con claridad y eficacia. En primer lugar, `JBuilder` busca la `BeanInfo` y, si no la encuentra, utiliza la introspección para detectar las pautas de diseño del bean.

Los JavaBeans describen *componentes*. Los componentes son los elementos básicos que utilizan las herramientas de diseño visual de `JBuilder` para construir un programa. Para generar programas se seleccionan componentes, se personalizan y se conectan. `JBuilder` de Borland se suministra con un conjunto de componentes que puede utilizar directamente y que aparecen en la paleta de componentes. Si desea complementarlos, puede crear otros componentes o instalar componentes de terceros.

Examine los JavaBeans por medio de `BeanInsight`. Seleccione Herramientas\ `BeanInsight` y escriba el nombre de un bean compilado para examinar sus propiedades, sucesos, personalizadores, etc.

## Consulte

- La especificación JavaBeans en <http://www.javasoft.com/beans/docs/spec.html>.
- “Gestión de la paleta de componentes” en la página 7-1.

## Los contenedores

---

Los *contenedores* son componentes que contienen y gestionan otros componentes. Los contenedores amplían `java.awt.Container`. Normalmente, los contenedores aparecen como paneles, marcos y cuadros de diálogo durante la ejecución del programa. Por lo general, las operaciones de diseño de `JBuilder` se realizan en contenedores.

## Tipos de contenedores

- El componente *Window* es un contenedor independiente de alto nivel que no tiene bordes, barra de título ni barra de menús. A pesar de que puede utilizar un componente de ventana para implementar una ventana emergente como, por ejemplo, la pantalla de presentación, habitualmente se utiliza una subclase de `java.awt.Window` en la interfaz de usuario, como una de las enumeradas a continuación, en lugar de recurrir directamente a la clase `Window`:

*Marco*                      Una ventana de alto nivel, con borde y título. El componente marco cuenta con los controles estándar de las ventanas como, por ejemplo, un menú de control, botones para minimizar y maximizar la ventana y controles para redimensionar la ventana. También incluye una barra de menú.

Normalmente, el contenedor principal de la interfaz de usuario de una aplicación de Java, no así en las applets, es una subclase personalizada de `Frame`. En la mayoría de los casos, la personalización consiste en la creación de una instancia y la inclusión de otros componentes dentro del componente `Frame`, además de la definición de etiquetas, el enlace de los controles a los datos, etc.

*Cuadro de diálogo*                      Una ventana emergente, parecida a un marco, pero que no puede contener una barra de menús. Los componentes de cuadro de diálogo se utilizan para recoger datos del usuario y mostrar mensajes de advertencia, y suelen ser provisionales. Existen los siguientes tipos de cuadros de diálogo:

*Modal*: impide al usuario utilizar cualquier otra ventana de la aplicación hasta que se cierra este cuadro de diálogo.

*No modal*: permite introducir información, tanto en el cuadro de diálogo como en la aplicación.

*Cuadro de diálogo Archivo*                      Un cuadro de diálogo básico de Abrir/Guardar archivos, que proporciona acceso al sistema de archivos.

- Un *panel* es un contenedor sencillo de la interfaz de usuario, sin borde ni título, que se utiliza para agrupar otros componentes como botones, casillas de selección y campos de texto. Los componentes `Panel` están incrustados en otros elementos de la interfaz, como `Frame` y `Dialog`. También es posible anidarlos dentro de otros paneles.

*Applet*                      Una subclase de `Panel` empleada para construir un programa que se incrustará en una página HTML y se ejecutará en un visualizador de HTML o un visor de applets. Dado que `Applet` es una subclase de `Panel`, puede contener componentes, pero no dispone de borde ni de título.

## Las bibliotecas de componentes

---

Las bibliotecas de componentes son recopilaciones de componentes listos para su uso.

JBuilder proporciona varias bibliotecas de componentes JavaBean para el diseño de interfaces de usuario en la paleta de componentes. Estas bibliotecas incluyen:

- Java `AWT`
- Java `Swing`
- JBuilder `dbSwing`

Normalmente, los componentes de una biblioteca comparten características de alto nivel. Cuando dentro de una biblioteca, o en varias distintas, existen componentes de características similares, las diferencias de comportamiento, aspecto o requisitos pueden ayudar a decidir cuál es el más adecuado.

Por ejemplo, los componentes AWT tienen la ventaja de ser compatibles tanto con las versiones antiguas como con las nuevas de los navegadores de Internet. Tienen la desventaja de ser pesados, lo que reduce el rendimiento. Los componentes Swing tienen la ventaja de ser abreviados y proporcionan un funcionamiento más enérgico. La compatibilidad con los navegadores es cada vez un problema menos importante. La biblioteca `dbSwing` incluida en JBuilder Enterprise consta de subclases de componentes Swing que tienen la ventaja añadida de contar con las propiedades `dataSet` y `columnName`, que enlazan a datos los componentes `Swing`.

Para determinar qué componentes son los más adecuados para una tarea concreta, compare los componentes que realizan tareas similares. En muchos casos, hay varios componentes que realizan la operación deseada, pero se puede elegir uno en función de su aspecto, su funcionamiento, su facilidad de manejo o su parecido con las otras herramientas de las que se espera dispongan los usuarios finales.

## Convenciones de la documentación

---

En la documentación de Borland para JBuilder, el texto con significado especial se identifica mediante la tipografía y los símbolos descritos en la siguiente tabla.

**Tabla 1.1** Convenciones tipográficas y de símbolos

Tipografía	Significado
<b>Negrita</b>	La negrita se utiliza para las herramientas java, <code>bmj</code> (Borland Make for Java - Make de Borland para Java), <code>bcj</code> (Borland Compiler for Java - Compilador de Borland para Java) y opciones del compilador. Por ejemplo: <b><code>javac</code></b> , <b><code>bmj</code></b> , <b><code>-classpath</code></b> .
<i>Cursiva</i>	Las palabras en cursiva indican los términos nuevos que se definen y los títulos de libros; ocasionalmente se usan para indicar énfasis.
<i>Teclas</i>	Las teclas, como “Pulse <i>Esc</i> para salir de un menú”.

**Tabla 1.1** Convenciones tipográficas y de símbolos (continuación)

Tipografía	Significado
Letra monoespaciada	<p>El tipo monoespaciado representa lo siguiente:</p> <ul style="list-style-type: none"> <li>■ texto tal y como aparece en la pantalla</li> <li>■ lo que sea que tenga que escribir, como en “Escriba <code>Hola</code> a todos en el campo Título del Asistente para aplicaciones”</li> <li>■ nombres de archivos</li> <li>■ nombres de vías de acceso</li> <li>■ nombres de directorios y carpetas</li> <li>■ comandos, como <code>SET PATH</code></li> <li>■ código Java</li> <li>■ tipos de datos de Java, como <code>boolean</code>, <code>int</code> y <code>long</code></li> <li>■ identificadores de Java, como nombres de variables, clases, nombres de paquetes, interfaces, componentes, propiedades, métodos y sucesos</li> <li>■ nombres de argumentos</li> <li>■ nombres de campos</li> <li>■ palabras clave de Java, como <code>void</code> y <code>static</code></li> </ul>
[ ]	<p>Los corchetes, en las listas de texto o sintaxis, encierran elementos optativos. En estos casos no se deben escribir los corchetes.</p>
< >	<p>Los corchetes angulares se utilizan para indicar las variables en vías de acceso a directorios, opciones de comandos y ejemplos de código.</p> <p>Por ejemplo, <code>&lt;nombredearchivo&gt;</code> se puede utilizar para indicar que es necesario especificar un nombre de archivo (incluida su extensión); <code>&lt;nombredeusuario&gt;</code> indica que se debe escribir un nombre de usuario.</p> <p>Cuando reemplace las variables en vías de acceso a directorios, opciones de comandos y ejemplos de código, sustituya la variable entera, incluidos los corchetes angulares (<code>&lt; &gt;</code>). Por ejemplo, sustituya <code>&lt;nombredearchivo&gt;</code> por el nombre de un archivo, como por ejemplo <code>empleado.jds</code>, y quite los corchetes angulares.</p> <p><b>Nota:</b> los archivos HTML, XML, JSP y de otros formatos basados en etiquetas utilizan también corchetes angulares para delimitar elementos del documento, como por ejemplo: <code>&lt;font color=red&gt;</code> o <code>&lt;ejb-jar&gt;</code>. La siguiente convención describe la forma de especificar cadenas de variables dentro de los ejemplos de código en cuya sintaxis se utilizan corchetes angulares como delimitadores.</p>
<i>Cursiva, con remates</i>	<p>Este formato de texto se utiliza para indicar cadenas de variables dentro de los ejemplos de código que ya utilizan corchetes angulares como delimitadores. Por ejemplo, <code>&lt;url="jdbc:borland:jbuilder\\samples\\guestbook.jds"&gt;</code>.</p>
...	<p>En los ejemplos de código, los puntos suspensivos (...) indican código que se ha omitido para ahorrar espacio y mejorar la claridad. Si están en un botón, los puntos suspensivos indican que este conduce a un cuadro de diálogo de selección.</p>

JBuilder se puede utilizar con diversas plataformas. En la tabla siguiente se proporciona una descripción de las convenciones de plataformas utilizadas en la documentación.

**Tabla 1.2** Convenciones de plataformas

Elemento	Significado
Vías de acceso	En las vías de acceso de la documentación se utiliza la barra normal (/). En la plataforma Windows se utiliza la barra invertida (\).
Directorio inicial	La ubicación del directorio inicial varía según la plataforma y se indica con la variable <home>. <ul style="list-style-type: none"> <li>■ En UNIX y Linux, el directorio inicial puede variar. Por ejemplo, puede ser /user/&lt;nombre de usuario&gt; o /home/&lt;nombre de usuario&gt;.</li> <li>■ En Windows NT, el directorio inicial es C:\Winnt\Profiles\&lt;nombre de usuario&gt;.</li> <li>■ En Windows 2000 y XP, el directorio inicial es C:\Documents and Settings\&lt;nombre de usuario&gt;.</li> </ul>
Imágenes de pantalla	Las imágenes o capturas de pantalla utilizan el aspecto Metal en diversas plataformas.

## Asistencia y recursos para desarrolladores

Borland proporciona una serie de opciones de asistencia y recursos de información para ayudar a los desarrolladores a obtener el máximo rendimiento de los productos Borland. Entre estas opciones se incluye una gama de programas de asistencia técnica de Borland, así como servicios gratuitos en Internet, los cuales permiten consultar una amplia base de información y ponerse en contacto con otros usuarios de productos Borland.

### El servicio de asistencia técnica de Borland

Borland ofrece varios programas de asistencia para clientes actuales y potenciales. Se puede elegir entre varios tipos de asistencia, que van desde la ayuda en la instalación de los productos Borland hasta el asesoramiento de expertos y la asistencia pormenorizada.

Si desea más información sobre el servicio al desarrollador de borland.com, visite nuestra página web, en <http://www.borland.com/devsupport>.

Cuando se ponga en contacto con el servicio técnico tenga a mano la información completa sobre el entorno, la versión del producto utilizada y una descripción detallada del problema.

Si necesita más información sobre las herramientas o la documentación de otros proveedores, póngase en contacto con ellos.

## Recursos en línea

---

También puede obtener información de los siguientes recursos en línea:

<b>World Wide Web</b>	<a href="http://www.borland.com/">http://www.borland.com/</a> <a href="http://www.borland.com/techpubs/jbuilder/">http://www.borland.com/techpubs/jbuilder/</a>
<b>Newsletters</b>	Para suscribirse a las newsletters, rellene el formulario en línea que aparece en: <a href="http://www.borland.com/products/newsletters/index.html">http://www.borland.com/products/newsletters/index.html</a> .

## World Wide Web

---

Visite periódicamente [www.borland.com/jbuilder](http://www.borland.com/jbuilder). El equipo de desarrollo de productos Java publica en esta página documentación técnica, análisis de competitividad, respuestas a preguntas frecuentes, aplicaciones de ejemplo, software actualizado e información sobre productos nuevos y antiguos.

En particular, pueden resultar interesantes las siguientes direcciones:

- <http://www.borland.com/jbuilder/> (actualizaciones de software y otros archivos)
- <http://www.borland.com/techpubs/jbuilder/> (actualizaciones de documentación y otros archivos)
- <http://community.borland.com/> (contiene nuestra revista de noticias para desarrolladores en formato web)

## Grupos de noticias de Borland

---

Puede registrar JBuilder y participar en los grupos de debate sobre JBuilder, estructurados en hilos. Los grupos de noticias de Borland ofrecen un medio para que la comunidad internacional de clientes de Borland pueda intercambiar sugerencias y técnicas sobre los productos de Borland, así como las herramientas y tecnologías relacionadas.

Puede encontrar grupos de noticias, moderados por los usuarios, sobre JBuilder y otros productos de Borland, en <http://www.borland.com/newsgroups>.

## Usenet, grupos de noticias

---

En Usenet existen los siguientes grupos dedicados a Java y temas relacionados:

- [news:comp.lang.java.advocacy](#)
- [news:comp.lang.java.announce](#)
- [news:comp.lang.java.beans](#)
- [news:comp.lang.java.databases](#)
- [news:comp.lang.java.gui](#)



- `news:comp.lang.java.help`
- `news:comp.lang.java.machine`
- `news:comp.lang.java.programmer`
- `news:comp.lang.java.security`
- `news:comp.lang.java.softwaretools`

**Nota** Se trata de grupos moderados por usuarios; no son páginas oficiales de Borland.

## Información sobre errores

---

Si cree que ha encontrado un error en el software, por favor informe a Borland en alguno de los siguientes sitios:

- La página Support Programs en <http://www.borland.com/devsupport/namerica/>. Pulse el enlace “Reporting Defects” para llegar al formulario Entry.
- Quality Central en <http://qc.borland.com>. Siga las instrucciones de la sección “Bugs Reports” de la página Quality Central.

Cuando informe sobre un fallo, incluya todos los pasos necesarios para llegar a él, así como toda la información posible sobre la configuración, el entorno y las aplicaciones que se estaban utilizando junto con JBuilder. Intente explicar con la mayor claridad posible las diferencias entre el comportamiento esperado y el obtenido.

Si desea enviar felicitaciones, sugerencias o quejas al equipo de documentación de JBuilder, envíe un mensaje a [jpgpubs@borland.com](mailto:jpgpubs@borland.com). Envíe únicamente comentarios sobre la documentación. Tenga en cuenta que los asuntos relacionados con el servicio técnico se deben enviar al departamento de asistencia técnica para programadores.

JBuilder es una herramienta creada por desarrolladores y para desarrolladores. Valoramos intensamente sus aportaciones.



# Introducción al diseñador

El diseñador cuenta con funciones que permiten diseñar visualmente clases que contienen constructores públicos por defecto. “Diseñador” es un término colectivo que engloba varias herramientas adaptadas a distintos tipos de diseño. Estas herramientas incluyen el diseñador de interfaces de usuario, el diseñador de menús, el diseñador de acceso a datos y el diseñador por defecto. El diseñador por defecto se utiliza con los componentes que no encajan en ninguna de las otras tres categorías. Cuando se accede al diseñador, JBuilder abre el tipo de diseñador adecuado para el archivo activo.

El diseñador es una OpenTool. Si desea añadir un tipo de diseñador o personalizarlo, elija Ayuda | Temas de ayuda, seleccione la pestaña Contenido y abra la documentación de OpenTools. Lea el tema “JBuilder Designer/CMT OpenTools Concepts” de OpenTools en *Developing OpenTools*. Las API de Package com.borland.jbuilder.cmt y Package com.borland.jbuilder.designer se encuentran en Referencia de la API de OpenTools.

## Utilización del diseñador

---

Pulse la pestaña Diseño del panel de contenido para activar el diseñador. Cuando el diseñador está activo, las áreas de trabajo del Visualizador de aplicaciones se modifican e incluyen tareas de diseño:

- El panel de contenido muestra la superficie de diseño.

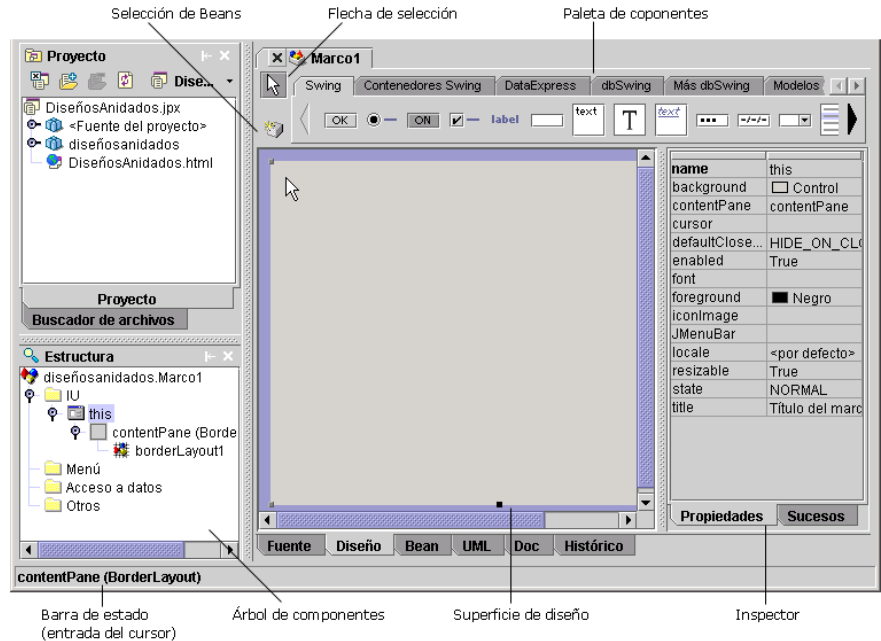
El panel de contenido también contiene:

- La paleta de componentes, que aparece en la parte superior.
- El Inspector, que aparece a la derecha.

- El panel de estructura contiene el árbol de componentes.

El componente seleccionado en el árbol de componentes o en la superficie de diseño aparece resaltado en el árbol y se muestra en el Inspector. La barra de estado indica el componente en el que se encuentra el cursor, en la superficie de diseño. Esto permite trabajar con un componente mientras ubica otros, sin cambiar el foco hasta que lo desee.

**Figura 2.1** El diseñador



En el ejemplo:

- El componente `jScrollPane` está seleccionado.

Esto se sabe porque aparece resaltado en el árbol de componentes, en la superficie de diseño se ve su tirador y es el componente activo del Inspector.

- El puntero se encuentra en la superficie de diseño, sobre el componente `jToolBar`.

Esto se sabe porque el nombre del componente aparece en la barra de estado.

- Los menús creados mediante el Asistente para aplicaciones están en su lugar.

Se puede afirmar esto ya que la carpeta del diseñador de menús tiene elementos. El trabajo de menú no aparece en el diseñador de interfaces de usuario, pero aún está ahí.

Si el usuario hace clic en la ubicación actual, `jToolBar` se selecciona, se resalta en el árbol de componentes, se refleja en el Inspector y se muestra en la barra de estado.

La tecnología Two-Way Tools (herramientas bidireccionales) de JBuilder mantiene sincronizadas las distintas partes del diseñador y el código fuente. Cambia inmediatamente el código según las modificaciones realizadas en el diseñador y viceversa.

## La superficie de diseño

---

La superficie de diseño es el cuaderno de dibujo virtual. Aquí se pueden añadir y eliminar componentes directamente, modificar el tamaño y ver el aspecto general del diseño mientras se realiza.

Si se selecciona un componente visible de la superficie de diseño, aparece en el Inspector, donde se pueden modificar sus propiedades.

Si desea anidar un componente dentro de un contenedor o activar sus tiradores, seleccione el contenedor. En la mayoría de los gestores de diseño es posible arrastrar los tiradores con el ratón para cambiar el tamaño o la posición de los componentes.

Para ajustar el tamaño de la superficie de diseño, arrastre el borde izquierdo del panel de contenido. Para ocultar los paneles de la izquierda del Visualizador de aplicaciones, elija `Ver/Ocultar todo`. Tenga en cuenta que esto incluye el árbol de componentes. Para volver a mostrar los paneles de la izquierda, elija `Ver/Mostrar todo`.

Todas las propiedades que se pueden modificar en la superficie de diseño se pueden modificar también en el Inspector.

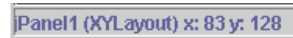
### Consulte

- [Capítulo 8, “Gestores de diseño”](#)
- [Capítulo 3, “El árbol de componentes y el Inspector”](#)
- [Capítulo 4, “Tratamiento de sucesos”](#)

### Localización de componentes

En los diseños complejos puede resultar difícil saber con certeza cuál de varios componentes posibles está seleccionado en la superficie de diseño. La barra de estado permite salir de dudas.

A medida que mueve el cursor del ratón, el nombre del componente sobre el que en cada momento se encuentra aparece en la barra de estado situada en la parte inferior del Visualizador de aplicaciones. Esto resulta especialmente útil si el componente que se intenta seleccionar está oculto o invisible en el diseñador, como puede ocurrir, por ejemplo, con un panel de una pila `CardLayout`. Si el panel que contiene el componente se encuentra en un diseño `XYLayout`, la barra de estado también muestra las coordenadas `x` e `y`.



jPanel1 (XYLayout) x: 83 y: 128

### Consulte

- [“Selección de componentes en la interfaz de usuario” en la página 5-2](#) si desea más información sobre la manipulación de componentes en la superficie de diseño.

## La paleta de componentes

---

La paleta de componentes proporciona un acceso rápido a todas las bibliotecas de componentes disponibles. Seleccione la pestaña que contenga el tipo de componentes deseado. Si desea ver una etiqueta de ayuda inmediata con el nombre de un componente, coloque el cursor sobre su icono. La paleta de componentes se puede personalizar.

Los componentes se añaden de las siguientes formas:

- Seleccione un componente de la paleta y haga clic en la superficie de diseño, en el lugar donde desee que aparezca la esquina superior izquierda del componente.

El componente se ubica en la superficie de contenido. Su ángulo superior izquierdo se coloca donde haya hecho clic con el ratón. El componente aparece también en el árbol de componentes.

- Seleccione un componente de la paleta y haga clic en la ubicación jerárquica adecuada, en el árbol de componentes.

El componente se coloca en el árbol. Si es visible, también aparece en el lugar adecuado de la superficie de diseño.

- Elija Edición | Añadir componente.

Se abre el cuadro de diálogo CVS Añadir: seleccione una biblioteca, elija un componente y pulse Aceptar.

El componente aparece en el árbol de componentes. Si es visible, también aparece en el lugar adecuado de la superficie de diseño.

### Consulte

- [“Adición de componentes” en la página 3-3](#)
- [“Gestión de la paleta de componentes” en la página 7-1](#)

## Utilización del botón Selección de Beans

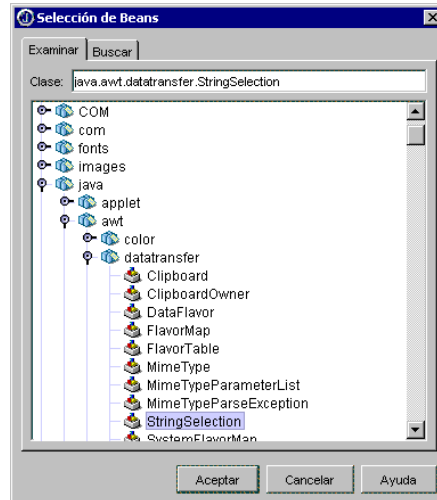


El botón Selección de Beans se encuentra en el borde izquierdo de la paleta de componentes bajo la flecha de selección. Muestra una lista de beans definida por el usuario. Cuando se elige un bean de la lista, Selección de Beans carga en el cursor una referencia al bean, como si se hubiera hecho clic en un componente de la paleta. Cuando se hace clic en la superficie de diseño, se añade el bean elegido.

Para añadir un bean a la lista desplegable Selección de Beans:

- 1 Compruebe que la biblioteca que contiene el bean se encuentra en la lista de bibliotecas necesarias para el proyecto, en el cuadro de diálogo Propiedades de proyecto. Si no es así, añada la biblioteca.
- 2 Pulse el botón Selección de Beans y elija Seleccionar en el menú desplegable.

Se abre el cuadro de diálogo Selección de Beans.



- 3 Utilice las fichas Buscar o Examinar:
  - En la ficha Buscar, empiece a escribir el nombre del bean en el campo Buscar.
  - En la ficha Examinar, empiece a escribir o amplíe el nodo correspondiente hasta localizar el bean que desee.

La ficha Buscar puede localizar el bean sólo con el nombre corto. En la ficha Examinar es necesario realizar una selección en el árbol o escribir el nombre de clase completo.

- 4 Seleccione el bean y pulse Aceptar.
- 5 Pulse nuevamente el botón Selección de Beans.

Observe que aparece un menú desplegable que contiene el nuevo Java Bean.

Si desea utilizar un bean mientras trabaja en el mismo proyecto, pulse el botón Selección de Beans y elija el bean en el menú.

### Consulte

- “Cómo añadir y configurar bibliotecas” en *Creación de aplicaciones con JBuilder*

## El Inspector

---

El Inspector muestra las propiedades y los sucesos del componente seleccionado y permite modificarlos por medio de menús contextuales, campos de texto y otros controles. Los editores personalizados se pueden utilizar en combinación con el Inspector. Seleccione un componente en el árbol de componentes o en la superficie de diseño. Sus propiedades y sucesos se muestran en el Inspector.

Haga clic en la pestaña Propiedades para mostrar y modificar las propiedades de un componente. Haga clic en la pestaña Sucesos para mostrar y modificar los sucesos de un componente.

Para ajustar el tamaño del Inspector, arrastre su borde izquierdo.

### Consulte

- [“Apertura del Inspector” en la página 3-6](#)
- [“Utilización de personalizadores en el diseñador” en la página 7-10](#)

## El árbol de componentes

---

El árbol de componentes permite acceder a los distintos tipos de diseñadores disponibles y proporciona una vista jerárquica de los componentes del archivo activo. También actúa como administrador de componentes y permite añadir, eliminar y reorganizar los componentes en la jerarquía de diseño.

Para seleccionar un tipo de diseñador, seleccione la carpeta correspondiente en el árbol de componentes: Menú, Interfaz de usuario, Acceso a datos o Por defecto. Cuando se pulsa la pestaña Diseño, el diseñador abre automáticamente el tipo de diseñador adecuado para el contenedor del archivo activo. Si lo desea, puede elegir otros diseñadores para trabajar con los componentes que se encuentran en el archivo activo o a los que se hace referencia en él.

Seleccione un componente en el árbol de componentes para ponerle el foco en la superficie de diseño. Sus propiedades y sucesos se muestran en el Inspector.

Para desplazar un componente dentro del árbol:

- 1 Selecciónelo.
- 2 Córtele por medio del atajo de teclado, el menú Edición o el menú contextual.
- 3 Seleccione el componente inmediatamente superior a la nueva posición.
- 4 Pegue el componente cortado.

### Consulte

- [“Utilización del árbol de componentes” en la página 3-1](#)



# Categorías de diseñadores

---

JBuilder proporciona distintos diseñadores para cuatro amplias categorías de JavaBeans:

- Diseñador de interfaces de usuario
- Diseñador de menús
- Diseñador de acceso a datos
- Diseñador por defecto

Estos diseñadores proporcionan conjuntos de funciones que facilitan el diseño de los componentes del tipo correspondiente. Puede cambiar de un tipo de diseñador al siguiente si activa los componentes en el tipo de diseñador que desee abrir. Puede cambiar entre tipos de diseñador de uno de estos tres modos:

- Haga clic dos veces en un componente del tipo de diseñador deseado en el árbol de componentes.
- Seleccione un componente del tipo de diseñador deseado en el árbol de componentes y pulse *Intro*.
- Haga clic con el botón derecho en un componente del tipo de diseñador deseado en el árbol de componentes y seleccione Activar el diseñador.

## Diseñador de interfaces de usuario

---

Los componentes de interfaz de usuario de un programa son los elementos que el usuario puede ver y utilizar durante la ejecución. Se derivan de `java.awt.Component`. En el diseñador aparecen los componentes de la interfaz de usuario que se ven normalmente durante la ejecución. Los componentes de la interfaz que normalmente no son visibles, como los menús emergentes, aparecen en la carpeta Por defecto del árbol de componentes.

Siempre que es posible, los componentes de interfaz de usuario de JBuilder están “activos” en la fase de diseño. Por ejemplo, las listas muestran los elementos que contienen y las rejillas de datos enlazadas a un conjunto activo de datos muestran los datos actuales.

### Consulte

- [Capítulo 5, “Creación de interfaces de usuario”](#)
- [Capítulo 9, “Tutorial: Creación de una interfaz de usuario sencilla”](#)

## Diseñador de menús

---

Los componentes de menú se derivan de `java.awt.MenuComponent`. Durante el diseño, JBuilder muestra los componentes de menú en la carpeta Menú del árbol de componentes.

El trabajo de menú aparece en la superficie de diseño sólo cuando está utilizando el Diseñador de menús. Por ejemplo, no aparece cuando utiliza el Diseñador de interfaz. Su trabajo de menú sigue siendo de la misma forma en que lo ha dejado, sólo que no se muestra en los otros diseñadores.

### Consulte

- [Capítulo 6, “Diseño de menús”](#)

## Diseñador de acceso a datos

---

Los componentes de acceso a datos representan componentes que no pertenecen a la interfaz de usuario y que sirven para conectar los controles enlazados a datos en una aplicación de base de datos. Los componentes de acceso a datos no aparecen en el contenedor de la interfaz de usuario en la fase de diseño, aunque sí aparecen en el árbol de componentes, en la carpeta *Acceso a datos*. Seleccione un componente de acceso a datos en el árbol de componentes para poder acceder a sus propiedades y sucesos en el Inspector.

### Consulte

- [“Componentes de base de datos” en la página 5-9](#)
- “Componentes DataExpress” en la *Guía del desarrollador de aplicaciones de base de datos*

## Diseñador por defecto

---

El diseñador por defecto proporciona herramientas de diseño visual para componentes ajenos a la superficie de la interfaz de usuario, los menús y el acceso a datos. Algunos ejemplos de estos son los elementos emergentes de la interfaz de usuario, como cuadros de diálogo u otros componentes JavaBeans que no forman parte de la interfaz de usuario, como `buttonGroup`. Para activar el diseñador por defecto, seleccione estos componentes en la carpeta *Por defecto* del árbol de componentes.

Cuando esté familiarizado con el árbol de componentes, la superficie de diseño, la paleta de componentes y el Inspector, podrá utilizar fácilmente el diseñador por defecto.

## Combinaciones de teclas del diseñador

---

Existen dos tipos de métodos abreviados de teclado: atajos de desplazamiento, que desplazan el foco de una zona a otra, y atajos de acción, en los que normalmente se utilizan el teclado y el ratón, que simplifican el trabajo con el diseñador.

## Atajos de desplazamiento

El ratón, la tecla *Tab* por sí sola, la combinación *Ctrl + Tab* y las teclas de flecha permiten desplazarse por el diseñador. Combine estas teclas abreviadas con la tecla *Mayús* para desplazar el foco al revés. Cuando se pulsa *Tab* o *Ctrl + Tab* el foco se desplaza por el siguiente orden:

- Panel de proyectos
- Árbol de componentes
- Paleta de componentes
- Superficie de diseño (utilice *Ctrl + Tab* para moverla)
- Inspector

## Atajos de acción

A continuación se proporciona una lista de atajos de ratón y teclado que facilitan el trabajo en el diseñador:

Tecla	Acción
<i>Ctrl+Hacer clic</i>	Selecciona/deselecciona individualmente componentes en el árbol de componentes o en la superficie de diseño.
<i>Mayús+Hacer clic</i>	Coloca varias instancias de un componente en la superficie de diseño. Utilice la flecha de selección de la paleta de componentes para deshacer la selección.
<i>Mayús+F10</i>	Muestra el útil menú contextual de la superficie de diseño, correspondiente al componente seleccionado en el árbol.
<i>Ctrl+X</i>	Corta una selección efectuada en la superficie de diseño o en el árbol de componentes y la lleva al Portapapeles.
<i>Ctrl+C</i>	Copia al Portapapeles una selección efectuada en la superficie de diseño o en el árbol de componentes.
<i>Ctrl+V</i>	Pega el contenido del Portapapeles en la superficie de diseño o en el árbol de componentes, en la posición del cursor.
<i>Ctrl+Flecha</i>	Desplaza el componente seleccionado un píxel en la dirección de la flecha.
<i>Ctrl+Mayús+Flecha</i>	Desplaza el componente seleccionado ocho píxeles en la dirección de la flecha.
<i>Ctrl+Z</i>	Deshace la operación más reciente. Puede utilizarse repetidamente para deshacer varias acciones sucesivas.
<i>Ctrl+Mayús+Z</i>	Rehace la última operación deshecha. Puede utilizarse repetidamente para rehacer varias anulaciones sucesivas.
<i>Ctrl+Supr</i> o <i>Supr</i>	Borra la selección.
<i>Mayús+Arrastrar</i>	Dibuja un rectángulo alrededor de un grupo de componentes y los selecciona en la superficie de diseño.
<i>Mayús+Arrastrar</i>	Limita los desplazamientos a las direcciones arriba, abajo, derecha, izquierda y ángulos de 45 grados.
<i>Alt+Arrastrar</i>	Arrastra un componente a un supercontenedor.
<i>Hacer clic+Arrastrar</i>	Arrastra el componente seleccionado a otro lugar.
<i>Ctrl+Arrastrar</i>	Arrastra una copia del objeto seleccionado a otro lugar.



# El árbol de componentes y el Inspector

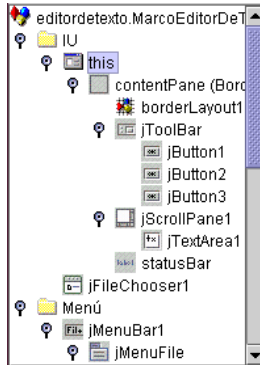
El árbol de componentes y el Inspector permiten acceder a todos los componentes del archivo activo y a todas sus propiedades y sucesos. El árbol de componentes muestra qué diseñador se está utilizando, qué componentes hay en el archivo activo y cuál de ellos se encuentra seleccionado. El Inspector muestra las propiedades y los sucesos del componente seleccionado. Seleccione un componente en la superficie de diseño o en el árbol de componentes. Sus propiedades se muestran en el Inspector. Cambie el nombre de un componente en el árbol de componentes o en el Inspector.

## Utilización del árbol de componentes

---

El árbol de componentes permite ver y gestionar los componentes en un archivo que se puede diseñar visualmente. Muestra todos los componentes en el archivo activo y sus relaciones, los gestores de diseño asociados a los contenedores de interfaz y el tipo de diseñador que utiliza cada componente. Permite el acceso a los comandos y controles para los diseñadores y los componentes. Los cambios efectuados en el árbol de componentes se reflejan inmediatamente en el Inspector, en la superficie de diseño y en el código fuente.

El árbol de componentes siempre muestra exactamente el componente seleccionado, por lo que permite realizar la selección con exactitud al margen del tipo de componente y de la complejidad del diseño.



El árbol de componentes permite ver y gestionar los tipos de diseñador y los componentes:

- Abrir el diseñador asociado de un componente, como el diseñador de menús.
- Añadir componentes de la paleta a la clase.
- Ver el nombre de un componente.
- Seleccionar un componente en el árbol de componentes para modificar sus propiedades y sucesos en el Inspector.
- Seleccionar un componente en el árbol de componentes para modificarlo en la superficie de diseño.
- Cambiar el nombre del componente.
- Desplazar un componente a otro contenedor o a otro lugar de la jerarquía.



Antes de seleccionar un componente, compruebe que la flecha de selección de la paleta de componentes está pulsada. Si no lo está, podría colocar accidentalmente un nuevo componente en el diseño.

El árbol de componentes admite la selección múltiple:

- Utilice la tecla *Ctrl* y el cursor para añadir elementos a la selección.
- Utilice la tecla *Mayús* y el cursor para seleccionar bloques.
- Mantenga pulsado el botón izquierdo del ratón y dibuje un rectángulo alrededor del grupo de componentes que desea cambiar.

El cursor se puede controlar con el ratón y con las teclas de flecha.



En `null` y en `XYLayout`, la esquina superior izquierda del componente se coloca en el lugar en el que se hace clic.

Los componentes ajustables aparecen con el tamaño por defecto.

- **Componentes visibles.** Haga clic en la superficie de diseño y, sin soltar el botón del ratón, arrastre hacia abajo o hacia la derecha hasta obtener el tamaño deseado.

En `null` y en `XYLayout`, la esquina superior izquierda del componente se coloca en el lugar en el que se hace clic.

**Nota**

En definitiva, el gestor de diseño de cada contenedor en la interfaz de usuario determinará el tamaño y la posición de los componentes.

Para añadir varias instancias de un componente:

- 1 Pulse la tecla *Mayús* al tiempo que selecciona el componente en la paleta de componentes.
- 2 Haga clic varias veces en la superficie de diseño o en el árbol de componentes con el objeto de añadir múltiples instancias del componente.
- 3 Cuando haya terminado, elimine la selección pulsando el botón Selección o eligiendo otro componente de la paleta.



**Nota**

Asegúrese de desactivar la selección. De otro modo, puede crear un componente adicional no deseado.

**Consulte**

- [Capítulo 8, “Gestores de diseño”](#) para obtener más información sobre `null`, `XYLayout` y otros gestores de diseño.

## Corte, copia y pegado de componentes

---

Para cortar, copiar o pegar componentes en el diseñador, seleccione el o los componentes en la superficie de diseño o en el árbol de componentes y, a continuación, realice una de estas operaciones:

- Seleccione el comando deseado en el menú Edición o utilice el método abreviado correspondiente a la función:

Cortar      *Ctrl+x*

Copiar      *Ctrl+c*

Pegar      *Ctrl+v*

- Haga clic con el botón derecho en los componentes seleccionados y seleccione Cortar, Copiar o Pegar en el menú contextual.



## Eliminación de componentes

---

Para borrar un componente, selecciónelo en la superficie de diseño o en el árbol de componentes. A continuación, seleccione Edición|Eliminar, utilice el método abreviado de teclado definido en su *configuración de teclado* o, bien, pulse la tecla *Supr*.

## Utilización de Deshacer y Rehacer

---

Para deshacer o volver a hacer una operación en el diseñador, puede optar por uno de estos métodos:

- Haga clic con el botón derecho en cualquier punto de la superficie de diseño o en el árbol de componentes y seleccione Deshacer o Volver a hacer en el menú contextual.
- Haga clic en cualquier parte de la superficie de diseño o el árbol de componentes y seleccione Edición|Deshacer (*Ctrl+Z*) o Edición|Volver a hacer (*Ctrl+Mayús+Z*).

Si se selecciona Deshacer varias veces, es posible deshacer varias acciones sucesivas. Esto anula los cambios realizados retrocediendo a través de las acciones y devolviendo el diseño a un estado anterior.

Volver a hacer invierte los efectos del último Deshacer. Si selecciona Volver a hacer varias veces, es posible recuperar varias acciones sucesivas. Volver a hacer se activa sólo tras un comando Deshacer.

## Cambio de nombre de los componentes

---

Puede cambiar el nombre de un componente mediante el árbol de componentes. Cuando lo haga, JBuilder realiza un perfeccionamiento de cambio de nombre en el nombre del componente.

Para hacerlo desde el árbol de componentes:

- 1 Seleccione el componente en el árbol de componentes.
- 2 Haga modificable el nombre del componente de una de las siguientes formas:
  - Pulse *F2*.
  - Haga clic con el botón derecho en el nombre del componente, en el árbol, y seleccione Cambiar nombre.
- 3 Escriba el texto del nuevo nombre.
- 4 Pulse *Intro*.

El nuevo nombre aparece en lugar del antiguo y todas las referencias a ese componente se actualizan correctamente.

## Desplazamiento de componentes

---

Para desplazar un componente con el ratón, arrástrelo a su nueva posición. Por medio del teclado:

- 1 Seleccione el componente que desea desplazar.
- 2 Córtele. Para ello, haga clic con el botón derecho del ratón y seleccione Cortar, o elija Edición|Cortar.
- 3 Seleccione el componente o carpeta que se encuentre inmediatamente debajo de donde desea pegar el componente.
- 4 Pegue el nuevo componente. Para ello, haga clic con el botón derecho del ratón y seleccione Pegar, o elija Edición|Pegar.

## Presentación de los nombres de clase de los componentes





---

Coloque el cursor sobre el nombre del componente. El nombre de la clase aparece en una etiqueta de ayuda inmediata.

## Iconos del árbol de componentes

---

Los iconos de los distintos componentes (paneles, botones, etc.) son versiones en miniatura de los que se utilizan en la paleta de componentes. Algunos de los iconos más comunes que aparecen en el árbol de componentes incluyen:

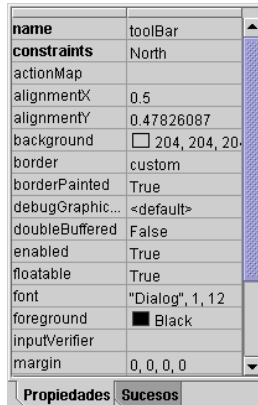
Icono	Explicación
	El archivo actual.
	El gestor de diseño del supercontenedor.
	El icono por defecto que se utiliza para un componente que no define su propio icono.
	Tipo de diseñador. Puede ser de interfaces, de menús, de acceso a datos, por defecto o personalizado.

## Apertura del Inspector

---

El Inspector aparece en el lado derecho del panel de contenido del diseñador. Le permite editar visualmente las propiedades de los componentes y adjuntar código a los sucesos de componente. Seleccione un componente en la superficie de diseño o en el árbol de componentes. Sus atributos se muestran en el Inspector. La pestaña Propiedades muestra todas las propiedades aceptadas y permite modificarlas. La pestaña Sucesos muestra todos los sucesos aceptados y permite modificarlos.

En la imagen que aparece a continuación, se selecciona un componente `toolBar` y se ve la pestaña Propiedades.



En la columna de la izquierda del Inspector se muestran las propiedades y los sucesos del componente. En la columna de la derecha se muestran los valores actuales.

Desde el Inspector, es posible:

- Cambiar el nivel de exposición de las propiedades.
- Cambiar los valores de las propiedades.
- Definir los valores iniciales de las propiedades de los componentes del contenedor, así como las del contenedor y su gestor de diseño.
- Crear el *código de tratamiento de sucesos*. Esto crea código para colocar sucesos en un contenedor que reciba sucesos de un componente dentro del contenedor.
- Traducir cadenas.

Los cambios que realice en el Inspector se reflejan inmediatamente en el código fuente y en el diseñador de interfaces de usuario.

### Consulte

- [Capítulo 4, “Tratamiento de sucesos”](#)
- [“Utilización de personalizadores en el diseñador” en la página 7-10](#) para averiguar cómo funcionan los personalizadores con el Inspector
- [Capítulo 10, “Tutorial: Creación de un editor de texto en Java”](#), donde puede adquirir práctica en el uso del Inspector

## Resaltado de los valores de las propiedades

Para que sea posible modificarlas en el Inspector, las variables deben:

- Ser variables de clase.
- Tener el nivel de exposición Oculto o Experto.

## Conversión de las propiedades en variables de clase

Las propiedades que son variables *estáticas* se pueden modificar en el Inspector. Para convertir una variable de instancia en una variable de clase por medio del Inspector:

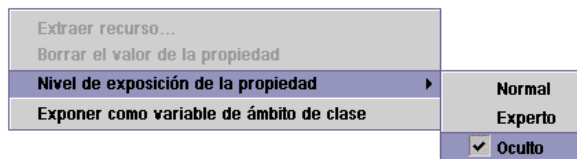
**1** En el Inspector, haga clic con el botón derecho del ratón en la propiedad.

**2** Elija Exponer como variable de ámbito de clase en el menú contextual.

En el Inspector aparece el valor correspondiente. JBuilder escribe una nueva declaración de variable y le aplica un valor. De esta manera, se puede manipular el valor fuera del contexto de la propiedad.

## Definición de la exposición de las propiedades

Es posible seleccionar el nivel de exposición de las propiedades del componente que aparecen en el Inspector, basándose en cómo están marcadas las propiedades en la clase BeanInfo del componente. Haga clic con el botón derecho en el Inspector y seleccione Nivel de exposición de la propiedad para acceder a un menú contextual con tres opciones:



**Normal** El Inspector muestra sólo las propiedades marcadas como Normal, no las marcadas como Oculto o Experto.

**Experto** El Inspector muestra las propiedades marcadas como Normal y Experto.

**Oculto** El Inspector muestra todas las propiedades: Normal, Experto y Oculto.

**Nota** Sólo se pueden ver las propiedades expuestas en la clase BeanInfo o definidas en el método `jbInit()` de la clase.

## Definición de los valores de las propiedades

Las propiedades son atributos que definen el aspecto de un componente y su respuesta durante la ejecución. En JBuilder, se definen las propiedades iniciales de un componente durante la fase de diseño. Por otro lado, el código puede cambiar estas propiedades durante la ejecución.

La ficha Propiedades del Inspector muestra las propiedades de los componentes seleccionados. Es aquí donde se definen durante la fase de diseño los valores de las propiedades de los componentes. La especificación de las propiedades durante la fase de diseño de hecho es una definición del estado inicial de un componente cuando se instancia a la interfaz de usuario durante la ejecución.

**Nota** Para modificar durante la ejecución los valores de las propiedades, se ha de añadir código en la sección principal de los métodos o en los manejadores de los sucesos, que pueden crearse en la ficha Sucesos del Inspector.

Para definir en la fase de diseño las propiedades los componentes:

**1** Seleccione un componente.

Puede elegir cualquier componente del árbol de componentes. Los componentes visibles también se pueden seleccionar en la superficie de diseño.

**2** Haga clic en la pestaña Propiedades del Inspector.

**3** Seleccione la propiedad que desea modificar, utilizando el ratón o las teclas de desplazamiento. Puede que necesite desplazarse hasta ver la propiedad buscada.

**4** Introduzca el valor en la columna derecha, de una de las maneras siguientes:

- Si hay un cuadro de texto, escriba el valor de la propiedad.
- Cuando el campo del valor tiene una lista desplegable, pulse la flecha junto a la propiedad y seleccione un valor.

Para desplazarse por la lista, utilice el ratón o las teclas de flecha *arriba* y *abajo*. Haga clic o pulse *Intro* en el valor deseado.

- Si el campo de valor muestra un botón de puntos suspensivos (...), haga clic en él y se mostrará un editor de la propiedad, como, por ejemplo, un selector de colores o de fuentes. Defina los valores en el editor de propiedades y después pulse *Aceptar* o *Intro*.

Cambie el nombre de un componente en el Inspector seleccionando su nombre en la columna de valor y escribiendo el nuevo nombre sobre el antiguo. Cuando cambia el nombre de un componente, JBuilder realiza una acción de perfeccionamiento de cambio de nombre, actualizando las referencias a ese componente correctamente.

## Definición de propiedades compartidas

Cuando se seleccionan varios componentes, el Inspector muestra sólo las propiedades que:

- Son comunes a los componentes.
- Se pueden modificar.

Cuando se cambia cualquiera de las propiedades compartidas en el Inspector, el valor de la propiedad cambia al nuevo valor en todos los componentes seleccionados.

**Nota** Si el valor de la propiedad compartida es diferente de un componente a otro, en el Inspector se muestra el valor por defecto o el valor del primer componente de la lista de selección.

Para definir propiedades para varios componentes:

- 1 Seleccione todos los componentes que van a compartir propiedades.
- 2 Seleccione y edite la propiedad en el Inspector.

### Definición de propiedades con la lista desplegable vacía

En ocasiones, el Inspector no puede proporcionar los valores de una propiedad. Para generar los valores:

- 1 En el Inspector, haga clic con el botón derecho del ratón en la propiedad.
- 2 Añada objetos de un tipo adecuado a la clase actual.

De esta forma se rellena la lista de valores de la propiedad. Use objetos inicializados para preferencia.

Normalmente, los objetos de componente adecuados se pueden añadir por medio del diseñador.

- 3 Ahora puede elegir estos objetos como valores, en la lista de valores de la propiedad.

### Consulte

- Ejemplo de “Los valores de las propiedades” en la [página 3-10](#).
- “Definición de la exposición de las propiedades” en la [página 3-8](#).

## El Inspector

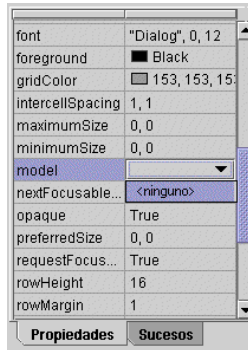
La información que presenta el Inspector de JBuilder se encuentra en la clase `BeanInfo` del bean o procede de la introspección del bean. Si no se ha especificado ningún editor de propiedades para una propiedad de la clase `BeanInfo` del bean (o si el bean no tiene una clase `BeanInfo`), el Inspector utiliza un editor por defecto basado en el tipo de datos del valor de la propiedad. Por ejemplo, si una propiedad toma una `cadena`, su editor permite escribir una cadena.

La lista de editores de propiedades, según el tipo, se guarda en `<propertyEditors.properties>`, en la carpeta `.jbuilder`. Si no hay registrado un editor por defecto para un tipo de dato determinado, JBuilder crea una lista desplegable que contiene todos los objetos con el tipo de datos adecuado y que están dentro del ámbito. Si no existieran objetos de ese tipo de dato en el ámbito, la lista desplegable estaría vacía. Es posible crear objetos del tipo de datos correcto en el ámbito. Estos objetos se muestran en la lista.

### Ejemplo

Por ejemplo, una `JTable` tiene una propiedad `model` que acepta objetos de tipo `TableModel`. Si añade componente `JTable` al diseño en el diseñador de

interfaces de usuario y, después, pulsa la flecha de la lista desplegable en la propiedad `model` en el Inspector, el valor de la lista desplegable es `<ninguno>`.



**Nota** Uno de los valores que busca el Inspector es el nivel de exposición de la propiedad. Para que la propiedad `model` que se utiliza sea visible en el Inspector, haga clic en ella con el botón derecho del ratón y seleccione Nivel de exposición de la propiedad!Oculto.

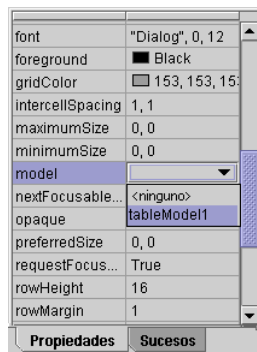
Para rellenar la lista desplegable de la propiedad `model` en el Inspector, añada a la clase objetos de tipo `TableModel`. Por ejemplo, se puede añadir una clase `TableModel` del paquete `javax.swing.table` desde la paleta de componentes:



- 1 Pulse el botón Selección de Beans de la paleta de componentes.
- 2 Pulse Seleccionar.
- 3 Amplíe el paquete `javax.swing.table`.
- 4 Seleccione `TableModel`.
- 5 Pulse Aceptar.
- 6 Haga clic en cualquier parte del árbol de componentes.

De esta forma se crea `tableModel1`.

Ahora la lista desplegable de la propiedad `model` para `JTable` en el Inspector contiene `tableModel1`.



**Nota** En algunos casos, puede que deba añadir a mano el objeto a la clase.





## Tratamiento de sucesos

Este capítulo explica cómo añadir, modificar y borrar manejadores de sucesos para componentes mediante el Inspector. También ofrece ejemplos específicos de cómo codificar manejadores de sucesos habitualmente utilizados para los componentes de cuadro de diálogo de JBuilder.

El *código de tratamiento de sucesos* se ejecuta cuando el usuario interactúa con la interfaz; por ejemplo, cuando pulsa un botón o selecciona un elemento de menú. El usuario puede interactuar con todos los componentes. Cuando esto ocurre, el componente correspondiente envía un mensaje. El programa espera estos mensajes y, cuando los recibe, reacciona de la forma adecuada. Para esperar los mensajes, el programa utiliza un *monitor de sucesos*; para reaccionar, utiliza un *manejador de sucesos*.

En la ficha Sucesos del Inspector se enumeran todos los sucesos que acepta el componente seleccionado. Cada suceso tiene una acción por defecto entre todas las posibles. Cuando se hace doble clic en un suceso en el Inspector, JBuilder escribe un monitor y un método *stub* (vacío) de tratamiento de sucesos, correspondiente a la acción por defecto, y muestra la vista Fuente con el cursor en este manejador. El código que describe lo que debe hacer el programa en respuesta al suceso, se rellena de forma manual.

Hay varios componentes visuales, como los cuadros de diálogo, que normalmente sólo aparecen cuando se ejecuta el código de tratamiento de sucesos. (Estos componentes aparecen en el diseñador por defecto). Por ejemplo, los cuadros de diálogo no forman parte de la superficie de la interfaz de usuario, sino que son elementos independientes que aparecen provisionalmente como resultado de una acción realizada por el usuario; por ejemplo, la selección de un menú o la pulsación de un botón. Por lo tanto, una parte del código relacionado con la utilización del cuadro de diálogo, como por

ejemplo una llamada a su método `show()`, tendrá que escribirse en el interior del método de tratamiento de sucesos. Este código es completamente personalizado.

### Consulte

- [“Apertura del Inspector” en la página 3-6](#)
- [“Requisitos para poder diseñar visualmente una clase” en la página 1-1](#)

## Vinculación de código de tratamiento de sucesos

---

Por medio de la ficha Sucesos del Inspector, puede vincular manejadores de sucesos a sucesos de componentes y borrar los manejadores existentes.

Para vincular código de tratamiento de sucesos a un suceso de un componente, realice los siguientes pasos:

- 1 Seleccione el componente en el árbol de componentes o en la superficie de diseño.
- 2 Seleccione la pestaña Sucesos, en el Inspector, para ver los sucesos del componente.
- 3 Seleccione un suceso. Utilice el botón del ratón o las teclas de flecha.
- 4 Haga doble clic en el nombre del suceso o pulse la tecla *Intro*.

JBuilder crea un manejador del suceso con un nombre por defecto que se puede modificar y traslada el foco del código fuente a dicho manejador.

JBuilder también inserta en la clase un fragmento de código denominado `adaptador`, que establece la conexión entre el suceso y el método de tratamiento.

- 5 Dentro de la sección principal del manejador del suceso, escriba el código que especifica cómo desea que responda el programa a este suceso del componente.

**Nota** Si desea averiguar qué métodos y sucesos admite un componente, consulte la documentación correspondiente a su clase. Para ello, haga doble clic en el componente, en el árbol de componentes, con el fin de cargar la clase en el Visualizador de aplicaciones. A continuación, seleccione la pestaña Doc.

### Consulte

- [“Conexión de controles y sucesos” en la página 4-3](#)

## Creación del manejador de sucesos por defecto

---

Para crear rápidamente un manejador del suceso por defecto:

- 1 Seleccione un componente en la paleta de componentes y añádala a su interfaz de usuario.

**2** Haga doble clic en el componente, en el diseñador.

Se crea un stub de suceso y el foco del código fuente pasa a dicho manejador.

**3** Añada el código necesario para completar el manejador del suceso.

**Nota** El suceso por defecto está definido por `BeanInfo` o bien como `actionPerformed`, si no se ha especificado uno.

## Eliminación de manejadores de sucesos

---

Para eliminar un manejador de sucesos:

- 1** Seleccione el componente en el árbol de componentes o en la superficie de diseño.
- 2** Seleccione la pestaña Sucesos en el Inspector.
- 3** Pulse el suceso deseado.
- 4** Resalte todo el nombre del manejador, en el campo Valor del suceso.
- 5** Pulse *Borrar*.
- 6** Pulse *Intro* para eliminar el nombre del manejador del suceso.

JBuilder elimina el gancho del método de manejo de sucesos asociado. Si el manejador estaba vacío, JBuilder también borra del código fuente la clase de adaptador. Elimine el método de forma manual.

## Conexión de controles y sucesos

---

Los adaptadores de sucesos conectan los sucesos a sus controles. Para esto se pueden utilizar adaptadores de sucesos estándar o adaptadores anónimos de clase interna.

Los adaptadores estándar crean una clase con nombre. La ventaja consiste en que el adaptador es reutilizable y más adelante se puede crear una referencia a él desde otra parte del código. Los adaptadores anónimos crean código incrustado. La ventaja consiste en que el código ocupa menos espacio. Sin embargo, sólo se puede utilizar una vez.

Cuando se utiliza un adaptador anónimo, JBuilder sólo crea el código del monitor y el stub de tratamiento de sucesos. Cuando se utiliza un adaptador estándar, JBuilder genera tres fragmentos de código:

- El stub de tratamiento de sucesos
- Un adaptador `EventAdapter`
- Un monitor `EventListener`

JBuilder crea una clase `EventAdapter` por cada conexión específica entre componente y suceso, utilizando como nombre de la clase el que

corresponde a dicho componente y dicho suceso. Este código se añade en una nueva declaración de clase, en la parte inferior del archivo.

Por ejemplo, en la aplicación `TextEdit`, `JBuilder` genera un tipo de adaptador de sucesos denominado `ActionAdapter` con el siguiente código:

```
// Crea el monitor de la conexión:
class TextEditFrame_jMenuFileExit_ActionAdapter implements ActionListener {
    TextEditFrame adaptee;

    // Conecta el adaptador a la clase:
    TextEditFrame_jMenuFileExit_ActionAdapter(TextEditFrame adaptee) {
        this.adaptee = adaptee;
    }

    // Proporciona ActionPerformed:
    public void actionPerformed(ActionEvent e) {
        adaptee.jMenuFileExit_actionPerformed(e);
    }
}
```

`JBuilder` también crea una línea de código en el método `jbInit()`. Esta línea conecta el código fuente del componente con el método de tratamiento de sucesos, por medio de `EventAdapter`. Para ello, añade un monitor. El método de monitor toma un parámetro del adaptador `EventAdapter` correspondiente.

En el ejemplo facilitado arriba, el `EventAdapter` se genera en el lugar. Su parámetro de constructor es la referencia `this` al `Marco`, que contiene el método para el tratamiento del suceso. Por ejemplo, la línea de código siguiente es la que realiza esta operación en la aplicación `Hola a todos`:

```
jButton1.addActionListener(new Marco1_jButton1_actionAdapter(this));
```

El nombre de la clase de adaptador es arbitrario. Lo que importa es que la referencia coincida.

`JBuilder` crea la clase adaptador por medio de la implementación del método que contiene la interfaz `ActionListener`. El método que trata el suceso seleccionado realiza una llamada a otro método del componente adaptado (`Marco1`) para generar la respuesta deseada.

## Adaptadores de sucesos estándar

---

`JBuilder` genera una clase adaptador de sucesos que implementa la interfaz apropiada. Después instancia la clase en el archivo de la interfaz y la registra como monitor del componente. Por ejemplo, para un suceso `jButton1` llama a `jButton1.addActionListener()`. Todo este código puede verse en el código fuente. Todo lo que le queda por hacer es rellenar el método manejador del suceso al que llama el adaptador de acción cuando se produce el suceso.

Por ejemplo, se genera este código para un suceso `focusGained()`:

```
jButton1.addFocusListener(new Frame1_jButton1_focusAdapter(this));

void jButton1_focusGained(FocusEvent e) {
    // El código de respuesta al suceso va aquí
}

class Marco1_jButton1_focusAdapter extends java.awt.event.FocusAdapter {
    Marco1 adaptee;

    Frame1_jButton1_focusAdapter(Frame1 adaptee) {
        this.adaptee = adaptee;
    }

    public void focusGained(FocusEvent e) {
        adaptee.jButton1_focusGained(e);
    }
}
```

La ventaja de este adaptador es que se puede volver a utilizar, porque tiene nombre. La desventaja es que sólo cuenta con acceso público y de paquetes, lo cual puede ser un límite a su utilidad.

## Adaptadores anónimos de clase interna

---

JBuilder también puede generar adaptadores de sucesos de clase interna. Las clases internas tienen las siguientes ventajas:

- El código se genera incrustado, lo que simplifica su aspecto.
- La clase interna tiene acceso a todas las variables del ámbito donde se declara, a diferencia de los adaptadores de sucesos estándar que sólo tienen acceso de nivel público y de paquete.

Este tipo particular de adaptadores de sucesos de clase interna generados por JBuilder se denomina *adaptadores anónimos*. Este estilo de adaptador crea una clase de adaptador sin nombre. La ventaja es que el código resultante queda compacto y elegante. La desventaja es que sólo se puede utilizar para este suceso, porque no tiene nombre y, por lo tanto, no se le puede llamar desde otro lugar.

Por ejemplo, el código siguiente se genera para un suceso `focusGained()` utilizando un adaptador anónimo:

```
jButton1.addFocusListener(new java.awt.event.FocusAdapter() {
    public void focusGained(FocusEvent e) {
        jButton1_focusGained(e);
    }
})

void jButton1_focusGained(FocusEvent e) {
}
```

Compare este código con el ejemplo de código de adaptador estándar mostrado anteriormente. JBuilder generó ese código mediante una clase adaptador estándar. Las dos maneras de utilizar los adaptadores proporcionan el código que gestiona los sucesos `focusGained`, pero el planteamiento del adaptador anónimo es más compacto.

## Elección del estilo de manejador de sucesos

---

Cuando se crea un suceso en la pestaña Sucesos del Inspector, JBuilder genera un código de adaptador de sucesos en la clase contenedor para gestionar la monitorización de los sucesos. JBuilder permite elegir el estilo del código de tratamiento de sucesos que se genera de forma automática. Los dos estilos de adaptador de sucesos son:

- Adaptadores de sucesos estándar
- Adaptadores anónimos de clase interna

Para especificar el estilo de los adaptadores de suceso generados por JBuilder:

- 1 Seleccione ProyectoPropiedades de proyecto.
- 2 Abra la ficha Formato.
- 3 Seleccione la pestaña Generado de la ficha Formato.
- 4 En la opción Tratamiento de sucesos, elija Adaptador estándar o Adaptador anónimo.

Si desea que el código generado coincida con los manejadores de sucesos, active la opción Seguir código existente.

- 5 Pulse Aceptar.

Para elegir un estilo de adaptador de sucesos por defecto para todos los proyectos nuevos:

- 1 Elija ProyectoPropiedades de proyecto por defecto.
- 2 Abra la ficha Formato.
- 3 Seleccione la pestaña Generado de la ficha Formato.
- 4 En la opción Tratamiento de sucesos, elija Adaptador estándar o Adaptador anónimo.

Si desea que el código generado para el tratamiento de sucesos coincida con los manejadores de sucesos, active la opción Seguir código existente.

- 5 Pulse Aceptar.

# Ejemplos: conexión y tratamiento de sucesos

---

A continuación se facilitan ejemplos de manejadores de sucesos utilizados frecuentemente:

- Presentación de un texto cuando se pulsa un botón.
- Apertura de un cuadro de diálogo desde un elemento de menú.

## Presentación de un texto cuando se pulsa un botón

---

Este es un ejemplo sencillo de cómo asociar un código que muestra “Hola a todos” en respuesta a un suceso de botón.

- 1 Ejecute el Asistente para aplicaciones e inicie una nueva aplicación. Seleccione ArchivosNuevos, abra la pestaña General de la galería de objetos y haga doble clic en Aplicación.
- 2 Acepte todos los valores por defecto y pulse Finalizar.  
`Marco1.java` se abre en el editor.
- 3 Haga clic en la pestaña Diseño en la parte inferior de `Marco1.java` para mostrar el diseñador de interfaces de usuario.
- 4 Seleccione los componentes `JTextField` y `JButton` de la pestaña Swing de la paleta de componentes.
- 5 Colóquelos en el árbol de componentes o en la superficie de diseño.
- 6 Seleccione `jButton1` para que aparezca en el Inspector.
- 7 Seleccione la ficha Sucesos del Inspector.
- 8 Haga doble clic en el suceso `actionPerformed`.  
 En este momento aparecerá en el código fuente una nueva función vacía para el método manejador del suceso.
- 9 Introduzca el código siguiente dentro de las llaves del método de manejo de suceso:

```
jTextField1.setText("Hola a todos!");
```

- 10 Ejecute la aplicación para probarla.



Pulse el botón Ejecutar en la barra de herramientas o pulse *F9*.

Cuando aparezca la aplicación, haga clic en el botón para ver cómo aparece el texto “Hola a todos” en el campo de texto. En este ejemplo, el componente `JFrame` monitoriza el suceso `actionPerformed` del botón. Cuando se produce dicho suceso, `JFrame` establece el texto de `TextField`.

## Apertura de un cuadro de diálogo desde un elemento de menú

---

Al diseñar sus propios programas, será necesario que rellene los stubs de manejo de sucesos. En este ejemplo, se desea obtener del cuadro de diálogo `JFileChooser` el nombre de archivo introducido por el usuario y utilizar este valor para abrir o manipular un archivo.

El siguiente ejemplo muestra cómo llamar al cuadro de diálogo `JFileChooser` desde el elemento de menú `ArchivoAbrir`:

- 1 Ejecute el Asistente para aplicaciones en un proyecto.
- 2 Pulse Siguiente en el Paso 1 para aceptar los valores por defecto de este paso.
- 3 Seleccione Generar barra de menús en el paso 2 y, después, pulse Finalizar para aceptar el resto de los valores por defecto.
- 4 Seleccione el archivo `Marco` (`Marco1.java`) en el panel de proyectos y haga clic en la pestaña Diseño, en la parte inferior de la ventana del Visualizador de aplicaciones, para ver las herramientas de diseño visual.
- 5 En la ficha Contenedores Swing de la paleta de componentes, seleccione el componente `JFileChooser` y colóquelo en la carpeta IU del árbol de componentes. Se añade un componente denominado `jFileChooser1` a la sección IU del árbol.

### Nota

Si se coloca este componente en otra parte del árbol o del diseñador, se convierte en un subcomponente de `this` y no de la interfaz de usuario en su conjunto. Por tanto, se convierte en la interfaz de usuario del archivo `Marco1.java`.

- 6 Cree un elemento de menú Abrir en el menú Archivo como se explica a continuación: el código de ejemplo que aparece a continuación se generó con Abrir como `jMenuItem1`.
  - a En el árbol de componentes, seleccione `MenuBar1` y pulse *Intro* para abrir el Diseñador de menús.
  - b Coloque el cursor en el elemento de menú `ArchivoSalir` en el diseñador y pulse el botón Insertar elemento en la barra de herramientas del diseñador de menús. Se añade un nuevo elemento de menú vacío.
  - c Introduzca `Open` como nuevo elemento de menú.
- 7 Seleccione el elemento de menú Abrir en el diseñador de menús o en el árbol de componentes. A continuación, seleccione la pestaña Sucesos del Inspector.





- 8** Haga doble clic en el suceso `actionPerformed` en el Inspector para generar en el código fuente el siguiente stub para el método de tratamiento de suceso.

```
void jMenuItem1_actionPerformed(ActionEvent e) {  
}
```

JBUILDER le lleva directamente al método manejador del suceso del código fuente.

- 9** Dentro de las llaves del método de tratamiento de sucesos `actionPerformed`, escriba lo siguiente:

```
jFileChooser1.showOpenDialog(this);
```

- 10** Guarde los archivos.

Ahora, ejecute el programa y seleccione en él ArchivoAbrir.

### Consulte

- [Capítulo 10, “Tutorial: Creación de un editor de texto en Java”](#).



# Creación de interfaces de usuario

Para crear una buena interfaz de usuario no basta con saber programar. Existen muchos libros en los que se tratan importantes aspectos como la facilidad de manejo y los principios del diseño eficaz. En este capítulo se tratan las herramientas que proporciona JBuilder para facilitar el proceso de implementación del diseño de la interfaz de usuario.

Esto incluye determinadas tareas:

- Creación de un proyecto que contenga un contenedor principal de interfaz de usuario, como un marco, un panel o un cuadro de diálogo.
- Adición de componentes al contenedor de interfaz de usuario; pueden ser otros contenedores, controles visuales y componentes de base de datos.
- Definición de los valores de las propiedades.
- Asignación de código a los sucesos de componente, de modo que las IU respondan a las acciones de los usuarios.
- Configuración de los diseños de contenedores y de las restricciones de los componentes, de modo que las IU se vean y se comporten correctamente después de la distribución.

JBuilder facilita este proceso mediante asistentes para crear los archivos básicos del proyecto y herramientas de diseño visual para agilizar las tareas de diseño de la interfaz de usuario.

**Nota** En otros apartados se describen los casos en que las herramientas de diseño visual son las mismas en los distintos tipos de diseñadores. Este capítulo se centra en la funcionalidad específica del Diseñador de IU.

### Consulte

- “El Asistente para aplicaciones”, que crea un grupo elemental de archivos y elementos de IU.
- [“Adición de componentes” en la página 3-3](#) para aprender a añadir componentes a su diseño y ajustarlos.
- [“Definición de los valores de las propiedades” en la página 3-8](#) para cambiar los valores de las propiedades de los beans.
- [“Vinculación de código de tratamiento de sucesos” en la página 4-2](#) para utilizar el Inspector para adjuntar y modificar stubs de tratamiento de sucesos.
- [Capítulo 8, “Gestores de diseño”](#) para obtener información sobre el uso de gestores de diseño de Java para que su IU se vea como lo desee y se comporte correctamente cuando cambia su tamaño.

## Selección de componentes en la interfaz de usuario

---

Ya ha creado la estructura básica de su IU, añadiendo contenedores y otros elementos al diseño. Ahora ya puede trabajar con los componentes elementales de la IU en su diseño.

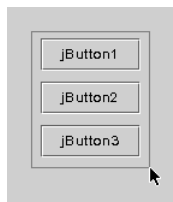
**Nota**



Antes de seleccionar un componente, compruebe que la flecha de selección de la paleta de componentes está pulsada. Si no lo está, podría colocar accidentalmente un nuevo componente en el diseño.

Para seleccionar varios componentes en la superficie de diseño, realice una de las operaciones siguientes:

- Mantenga pulsada la tecla *Ctrl* y, en la superficie de diseño, haga clic en los componentes, uno por uno.
- Mantenga pulsada la tecla *Mayús* y arrastre al ratón en la superficie de diseño para encerrar los componentes en un rectángulo.



Cuando todos los componentes que desea seleccionar estén dentro del rectángulo, suelte el botón del ratón. Si es necesario, puede utilizar *Ctrl+clic* para añadir o eliminar individualmente componentes del grupo seleccionado.

**Consulte**

- [“Adición de componentes” en la página 3-3](#) si desea más información sobre la forma de añadir componentes independientes o agrupados.
- [“Atajos de acción” en la página 2-9](#).

**Adición a contenedores anidados**

---

Para generar una interfaz medianamente compleja, es necesario anidar unos contenedores dentro de otros. Por ejemplo, puede necesitar añadir un panel desde otra parte a un contenedor `BorderLayout` que ya tiene otros dos paneles. Necesita disponer de un modo de indicar al diseñador qué contenedor debe recibir el componente seleccionado.

Para ello:

- 1 Seleccione el contenedor al que desea añadir el componente.  
 Selecciónelo en el árbol de componentes o en el diseñador. La barra de estado indica si se ha elegido el contenedor adecuado.
- 2 Seleccione en la paleta el componente que desea añadir.
- 3 Suelte el componente en el supercontenedor, en la superficie de diseño, mientras mantiene pulsado el botón del ratón.
- 4 Pulse la tecla *Alt* y manténgala pulsada mientras suelta el botón del ratón.

También puede añadir fácilmente un componente en un contenedor que se encuentre en un diseño anidado, soltándolo en el contenedor destino en el árbol de componentes del panel de estructura.

Cuando el componente se encuentre en el nuevo contenedor, podrá modificar sus restricciones para establecer su posición exacta.

**Desplazamiento y redimensionamiento de componentes**

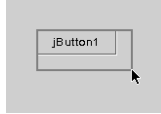
---

En muchos diseños, el gestor de diseño determina por completo el tamaño de los componentes mediante restricciones que impiden que se puedan dimensionar los componentes. No obstante, si el valor de la propiedad `layout` es `null` o `XYLayout` en el Inspector, es posible dimensionar los componentes cuando se colocan en la interfaz de usuario o redimensionarlos y desplazarlos más adelante.

Para dimensionar un componente cuando lo incorpora a la interfaz de usuario:

- 1 Seleccione el componente en la paleta de componentes.
- 2 Sitúe el cursor donde desea que aparezca el componente en el diseño.
- 3 Arrastre el puntero del ratón sin soltar el botón.

Cuando mueva el ratón, un perfil va indicándole el tamaño y la posición del control.

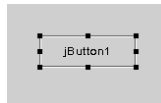


- 4 Suelte el botón del ratón cuando el perfil del control tenga el tamaño deseado.

Para redimensionar un componente, modifique sus restricciones en el Inspector o utilice el ratón:

- 1 Haga clic en el componente en la superficie de diseño o en el árbol de componentes para seleccionarlo.

Cuando un componente se encuentra seleccionado, en su perímetro aparecen unos pequeños cuadrados denominados *tiradores* o *cuadros de dimensionamiento*. En el centro de algunos contenedores aparece un tirador adicional, denominado *cuadro de desplazamiento*.



- 2 Haga clic en uno de estos cuadros y arrastre el ratón hasta conseguir la dimensión correcta.

Para desplazar un componente con el ratón:

- 1 Haga clic, en el componente en la superficie de diseño o en el árbol de componentes, para seleccionarlo.
- 2 Realice una de las siguientes acciones en la superficie de diseño:
  - Pulse en cualquier lugar dentro del componente y arrástrelo en cualquier dirección. Si el componente es un contenedor que está completamente cubierto con otros componentes, utilice el cuadro de traslado central para arrastrarlo.
  - Mantenga pulsada la tecla *Ctrl* y, mediante una de las teclas de desplazamiento, desplace el componente un píxel en la dirección elegida.
  - Mantenga pulsadas las teclas *Mayús+Ctrl* y, mediante una de las teclas de desplazamiento, desplace el componente ocho píxeles en la dirección elegida.

Para desplazar un componente sin usar el ratón:

- 1 Seleccione el componente en el árbol de componentes.
- 2 Corte el componente. Elija Edición|Cortar o utilice la combinación de teclas correspondiente de la emulación del editor.
- 3 Seleccione el supercontenedor en el árbol de componentes.

- 4 Pegue el componente. Elija Edición | Pegar o utilice la combinación de teclas correspondiente.

**Importante** Si no utiliza el ratón, asegúrese de añadir los componentes por el orden en que deben aparecer en el diseño final. Si un componente se añade en el lugar inadecuado, es necesario borrar los componentes siguientes, que se encuentran en su nivel de jerarquía o en niveles inferiores, y rehacer el diseño a partir del componente corregido.

### Consulte

- “[Corte, copia y pegado de componentes](#)” en la [página 3-4](#) si desea más información sobre la forma de desplazar componentes y cambiar su tamaño por medio del ratón.

## Gestión del diseño

---

Se parte de la base de que hay un proyecto lleno de componentes diseñables visualmente, que tienen propiedades y sucesos asociados. Antes de que la interfaz se haga demasiado complicada, es conveniente agrupar los componentes con el fin de obtener el aspecto más limpio posible. Es necesario comprobar que la interfaz tiene suficientes componentes para resultar útil. Cuando la interfaz de usuario esté completa y su aspecto y su comportamiento estén bajo control, se puede ajustar el aspecto y realizar una prueba destinada a comprobar que el efecto es el deseado.

En el resto de este capítulo se estudian estos temas avanzados.

### Agrupación de componentes

---

Algunos componentes en la paleta son contenedores que sirven para agrupar los componentes, de forma que se comporten como uno solo en la fase de diseño.

Por ejemplo, podría agrupar una fila de botones en un `Panel` para crear una barra de herramientas. También puede utilizar un contenedor para crear un grupo personalizado de botones, barras de estado o casillas de selección.

Cuando se añaden componentes a un contenedor, se crea una relación entre el contenedor y los componentes que contiene. Todas las operaciones que realice con los contenedores en fase de diseño, como mover, copiar o borrar, afectan también a todos los componentes agrupados dentro de ellos.

Para agrupar componentes situándolos dentro de un contenedor:

- 1 Añada un contenedor a la interfaz de usuario. Si utiliza `null` o `XYLayout`, puede arrastrarlo para dimensionarlo.
- 2 Añada todos los componentes al contenedor, asegurándose de que el puntero del ratón esté dentro de los límites del contenedor. (La barra de estado en la parte inferior de `AppBrowser` muestra el contenedor sobre el que se encuentra el puntero.) Puede soltar un componente nuevo de la paleta de componentes o arrastrar un componente hasta el interior del

nuevo contenedor. A medida que se añaden los componentes, estos aparecen dentro del contenedor seleccionado en la superficie de diseño y, dentro del contenedor, en el árbol de componentes.

**Sugerencia** Si desea que los componentes se mantengan donde los ha situado, cambie el diseño del contenedor a `null` o `XYLayout` antes de añadir componentes. En caso contrario, el tamaño y la posición de los componentes cambiará de acuerdo con el gestor de diseño utilizado por el contenedor. Puede cambiar el diseño final una vez añadidos todos los componentes.

## Adición de componentes para la generación de aplicaciones

Durante el diseño de clases, los componentes que no descenden de `java.awt.Component` (como los componentes de menús, cuadros de diálogo y base de datos), se tratan de forma diferente que los componentes de interfaz de usuario. Están representados en la paleta de componentes pero, cuando se añaden a la clase, solo pueden verse en el árbol de componentes. También utilizan diseñadores diferentes. Puede seleccionarlos en el árbol de componentes para cambiar sus propiedades en el Inspector, o hacer doble clic en ellos para abrir el diseñador asociado.

### Menús

Para añadir menús a la interfaz de usuario:

- 1 Haga clic en uno de los siguientes componentes de barra de menú o menú contextual de la paleta de componentes:

Pestaña Contenedores Swing	JMenuBar
	JPopupMenu
Pestaña AWT	MenuBar
	PopupMenu

- 2 Suéltelo en cualquier lugar dentro del árbol de componentes o de la superficie de diseño. Fíjese en que se sitúa en la carpeta `Menú` del árbol de componentes.
- 3 Haga doble clic sobre el menú componente en el árbol de componentes para abrir el diseñador de menús, o haga clic sobre él con el botón derecho del ratón y seleccione `Activar el diseñador`.
- 4 Añada elementos de menú al diseñador de menús.
- 5 Vincule sucesos a los elementos de menú por medio del Inspector o escribiendo manualmente el código.
- 6 Cierre el diseñador de menús haciendo doble clic sobre un componente de interfaz del árbol de componentes.

### Consulte

- [Capítulo 6, “Diseño de menús”](#)



## Cuadros de diálogo

Existen dos maneras de incorporar un cuadro de diálogo al proyecto:

- Utilice uno que ya existe de la paleta de componentes.
- Cree uno personalizado mediante el Asistente para cuadros de diálogo en la galería de objetos (Archivo|Nuevo).

Para añadir un cuadro de diálogo de la paleta:

- 1 Seleccione uno de los componentes de diálogo, como `JFileChooser`, de la paleta de componentes. Los puede encontrar en la pestaña Contenedores Swing y en la pestaña Más dbSwing.

- 2 Suéltelo en la carpeta `IU` del árbol de componentes.

**Nota**

En función de qué tipo de cuadro de diálogo sea, se coloca en la carpeta `IU` o en la carpeta Por defecto del árbol de componentes.

- 3 Vincule sucesos al elemento de menú que presentará el cuadro de diálogo durante la ejecución. Utilice la pestaña Sucesos del Inspector o escriba el código fuente.

**Sugerencia**

Seleccione cualquier componente dbSwing del árbol de componentes `frame` a `this` para que puedan verse durante la ejecución.

Para crear un cuadro de diálogo personalizado con el Asistente para cuadros de diálogo:

- 1 Seleccione Archivo|Nuevo y, a continuación, haga doble clic en el icono Asistente para cuadros de diálogo en la galería de objetos.
- 2 Asigne un nombre a la clase cuadro de diálogo y elija la clase básica de la que desea que herede.
- 3 Haga clic en Aceptar para cerrar el cuadro de diálogo.

En el proyecto se crea una clase shell de cuadro de diálogo, a la que se añade un componente Panel dispuesto para la configuración en el diseñador.

- 4 Complete los diseños de interfaz de usuario deseados y, después, vincule sucesos a los elementos de menú que presentarán el cuadro de diálogo durante la ejecución.

Para obtener más información sobre cómo asociar sucesos de menú con cuadros de diálogo, consulte el [Capítulo 4, “Tratamiento de sucesos”](#).

Una vez creado el cuadro de diálogo y diseñada su interfaz de usuario, deseará probar o utilizar el cuadro de diálogo desde alguna interfaz de usuario del programa.

Para utilizar cuadros de diálogo que no son beans:

- 1 Instancie su clase de cuadro de diálogo en algún lugar del código desde el que se tenga acceso a un `Frame`, el cual puede servir como superparámetro `Frame` del constructor de cuadros de diálogo. Un ejemplo típico de esto es un `Frame` cuyas interfaces de usuario se estén diseñando y que contenga un

botón o un elemento de menú cuya finalidad sea mostrar el cuadro de diálogo. En las applets, puede obtener el `Frame` mediante la llamada `getParent()`.

Para un cuadro de diálogo no modal (que se denomina `dialog1` en este ejemplo), puede utilizar el formulario del constructor que toma un único parámetro (el súper `Frame`) como sigue:

```
Dialog1 dialog1=new Dialog1(this);
```

Para un cuadro de diálogo modal, es necesario utilizar un formulario del constructor que tenga asignado al parámetro booleano `modal` el valor `true`, como en el siguiente ejemplo:

```
Dialog1 dialog1=new Dialog1(this, true);
```

Puede introducir esta línea como una variable de instancia en la parte superior de la clase (en cuyo caso el cuadro de diálogo se instancia durante la construcción del `Frame` y se podrá volver a utilizar), o bien puede situar esta línea de código en el manejador de sucesos `actionPerformed()` del botón que llama al cuadro de diálogo (en cuyo caso se instancia una nueva instancia del cuadro de diálogo cada vez que se pulse el botón). Con cualquiera de las opciones, esta línea crea una instancia del cuadro de diálogo pero todavía no lo hace visible.

(En caso de que el cuadro de diálogo sea un bean, debe asignar a la propiedad `frame` el valor del supermarco antes de la llamada `show()`, en vez de suministrar el marco al constructor.)

- 2 Antes de hacer visible el cuadro de diálogo instanciado, deben configurarse todos los valores por defecto que muestren los campos del cuadro de diálogo. Si piensa construir su cuadro de diálogo como un Bean (consultar más abajo), necesita que estos campos del cuadro de diálogo sean accesibles como propiedades. Esto se consigue definiendo métodos de obtención y asignación en la clase de su cuadro de diálogo.
- 3 A continuación, es necesario convertir el cuadro de diálogo en visible durante el suceso `actionPerformed()`, introduciendo una línea de código dentro del manejador de sucesos semejante a:

```
dialog1.show();
```

- 4 Cuando el usuario pulsa el botón Aceptar (o el botón Aplicar en un cuadro de diálogo no modal), el código que usa el cuadro de diálogo deberá llamar a los métodos de obtención de propiedades del cuadro de diálogo para leer la información introducida por el usuario desde fuera del cuadro de diálogo y, entonces, hacer algo con esa información.
  - Con un cuadro de diálogo modal, puede hacer esto justo después de llamar al método `show()`, puesto que `show()` no vuelve hasta que se abandona el cuadro de diálogo modal.
  - Con un cuadro de diálogo no modal, `show()` vuelve inmediatamente. Debido a esto, la clase de cuadro de diálogo necesita exponer sucesos cada vez que se pulse el botón. Al utilizar el cuadro de diálogo, es necesario registrar monitores para los sucesos del cuadro de diálogo y

añadir código en los métodos de tratamiento de sucesos, de forma que utilicen los métodos de obtención de propiedades para obtener la información introducida desde fuera del cuadro de diálogo.

### Consulte

- [Capítulo 10, “Tutorial: Creación de un editor de texto en Java”](#) si desea ver ejemplos de uso de componentes de cuadros de diálogo modales.

## Componentes de base de datos

Los componentes de base de datos son componentes JavaBean que controlan los datos y que se conectan a menudo a componentes de IU enlazados a datos. A menudo no se muestran por sí mismos en la interfaz de usuario. Se encuentran en la ficha DataExpress de la paleta de componentes.

Para añadir un componente de base de datos a la clase por medio del ratón:

- 1 Abra la pestaña DataExpress de la Paleta de componentes y haga clic en el componente que desee.
- 2 Suéltelo en el árbol de componentes o en la superficie de diseño.  
Aunque no se puede ver el componente en el Diseñador de interfaces, aparece en la carpeta Acceso a datos del árbol de componentes.
- 3 Modifique sus propiedades y añada manejadores de sucesos como lo haría con otros componentes.

Para añadir un componente de base de datos por medio de los menús:

- 1 Elija Edición\Añadir componente.  
Aparece el cuadro de diálogo Añadir componente.
- 2 Seleccione un componente de la biblioteca DataExpress.
- 3 Pulse Aceptar o *Intro*.

Para utilizar el diseñador de columnas:

- 1 Utilice cualquiera de los métodos explicados arriba para añadir un componente con columnas, como un `TableDataSet` o un `QueryDataSet`.
- 2 Pulse el icono de ampliación contiguo al componente para ampliarlo en el árbol de componentes.
- 3 Haga doble clic en el nodo `<newcolumn>` para abrir el diseñador de columnas.  
A continuación, ajuste el aspecto y el comportamiento de la columna.
- 4 Cierre el diseñador de columnas haciendo doble clic sobre cualquier componente del árbol de componentes que no sea de base de datos.

## Consulte

- “Aplicaciones de base de datos JBuilder” en la *Guía del desarrollador de aplicaciones de base de datos* si desea más información sobre el uso de los componentes de base de datos.

## Cambio del aspecto

---

En el desarrollo multiplataforma, los diseños deben tener un aspecto y un comportamiento predecibles en todas las plataformas con las que el programa vaya a ser compatible. La mayoría de los desarrolladores trabajan principalmente con una plataforma. Resulta difícil prever el aspecto de una interfaz de usuario en las plataformas con las que el desarrollador no está familiarizado.

JBuilder proporciona varios procedimientos directos para cambiar el aspecto durante la ejecución y durante el diseño.

## Aspecto durante la ejecución

JBuilder incluye los componentes Swing de interfaz gráfica de usuario (GUI) de Java Foundation Classes. La arquitectura Swing permite determinar el aspecto de la interfaz de usuario de los programas. Puede emplear esta característica de Java para crear aplicaciones que presenten el aspecto del escritorio habitual de un usuario. También puede conseguir que sus aplicaciones tengan el mismo aspecto en todas las plataformas mediante Java Metal.

JBuilder proporciona varias opciones de aspecto:

- Metal
- CDE/Motif
- Windows (sólo admitido en plataformas Windows)
- MacOS Adaptive (disponible únicamente en las plataformas Macintosh)

Cuando se crea una aplicación o un applet con los asistentes de JBuilder, se genera automáticamente el siguiente código en la clase que ejecuta el programa, `Aplicacion1.java` o `Applet1.java`, por ejemplo.

```
UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
```

Por ejemplo:

## Aplicación

```
//Método principal
public static void main(String[] args) {
    try {
        UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
    }
    catch(Exception e) {
    }
    new Application1();
}
```

## Applet (cuando la clase de base es javax.swing.JApplet)

```
//Inicializador estático para definir aspecto
static {
    try {
        //UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
        //UIManager.setLookAndFeel(UIManager.getCrossPlatformLookAndFeelClassName());
    }
    catch(Exception e) {
    }
}
```

JBUILDER utiliza el siguiente método para determinar el aspecto del programa:

```
UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
```

Este método detecta automáticamente la plataforma en la que se ejecuta el programa y adapta el aspecto en consecuencia.

La línea de código que establece la sentencia aspecto se encuentra dentro del bloque try/catch. El método setLookAndFeel() lanza un número de excepciones que necesitan ser capturadas y gestionadas explícitamente.

El aspecto del programa durante la ejecución se puede modificar cambiando el código del método

UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName()); por uno de los siguientes:

```
// Aspecto Metal:
UIManager.setLookAndFeel("javax.swing.plaf.metal.MetalLookAndFeel");

// Aspecto Motif:
UIManager.setLookAndFeel("com.sun.java.swing.plaf.motif.MotifLookAndFeel");
```

### Importante

Si desea que el programa se ejecute con el aspecto Motif, selecciónelo en el diseñador de modo que pueda ver el resultado final. Motif deja más espacio alrededor de algunos componentes, como los botones.

## Aspecto en la fase de diseño

El aspecto durante la ejecución depende de la configuración del código fuente. El aspecto seleccionado en el diseñador de JBUILDER es solo una vista previa y no tiene efecto alguno sobre el código fuente. El diseñador permite cambiar el aspecto en todo momento para ver cómo afectaría al diseño; más adelante se puede cambiar el código para establecer el aspecto elegido.

Hay dos formas de cambiar el aspecto durante la fase de diseño:

- Haga clic con el botón derecho en la superficie de diseño y seleccione Aspecto. Seleccione un aspecto alternativo en este submenú.

Esto solo cambia el aspecto en el diseñador y únicamente para el proyecto actual. Al utilizar este método, se puede diseñar en un aspecto y tener una vista previa del aspecto durante la ejecución, sin necesidad de salir del diseñador.

- Seleccione Herramientas|Opciones del IDE. Seleccione un aspecto alternativo en la lista desplegable Aspecto de la ficha Visualizador.

Esto cambia el aspecto del entorno de JBuilder, pero puede utilizar el primer método para cambiar el aspecto del diseñador.

La superficie de diseño se actualiza y presenta el aspecto seleccionado. Esto no modifica el código.

**Importante**

El cambio del aspecto en el diseñador no influye sobre el código. Se trata sólo de una vista previa que se muestra en el diseñador, pero no modifica el aspecto que presenta el programa mientras se ejecuta.

## Comprobación de la interfaz de usuario durante la ejecución

---

Cuando esté preparado para comprobar el programa, puede optar por limitarse a ejecutarlo o depurarlo y ejecutarlo simultáneamente.



Para ejecutar la aplicación, seleccione Ejecutar!Ejecutar proyecto, pulse *F9*, o haga clic en el botón Ejecutar en la barra de herramientas. JBuilder compila el programa y, si hay errores, se para la compilación para que se puedan corregir los errores y probar de nuevo.



Para depurar la aplicación, seleccione Ejecutar!Depurar, pulse *Mayús+F9*, o haga clic en el botón Depurar. Depure el programa y corrija los errores.

Si desea obtener más información sobre estos temas consulte los siguientes capítulos en *Creación de aplicaciones con JBuilder*.

### Consulte

En *Creación de aplicaciones con JBuilder*:

- “Ejecución de programas en Java”
- “Generación de programas en Java”
- “Compilación de programas en Java”
- “Depuración de programas en Java”
- “Distribución de programas en Java”

## Diseño de menús

Este capítulo muestra cómo diseñar menús visualmente. Mediante el diseñador de menús de JBuilder, se pueden diseñar visualmente tanto menús de barra como emergentes.

### Apertura del diseñador de menús

---

Para abrir el Diseñador de menús, seleccione primero la pestaña Diseño del AppBrowser para abrir el Diseñador de interfaces. Una vez que el diseñador esté abierto:

- 1 Amplíe la carpeta Menú en el árbol de componentes.
- 2 Haga doble clic en cualquier componente de menú del árbol de componentes.

La superficie de diseño cambia para mostrar las funciones del Diseñador de menús.

Si no existen componentes en la carpeta Menú, debe llevar primero un componente de menú al diseñador.

- 1 Utilice el cuadro de diálogo Añadir componentes o la paleta de componentes para añadir uno de los siguientes componentes de menú emergente o de barra de menú:
  - En la paleta Contenedores Swing:

JMenuBar  
JPopupMenu

- En la paleta AWT:

MenuBar

PopupMenu

- 2 Coloque el componente en el árbol de componentes o en la superficie de diseño.

Se sitúa automáticamente en la carpeta Menú del árbol de componentes.

Puede cambiar entre el Diseñador de menús y cualquier otro tipo de diseñador de uno de los tres modos siguientes:

- Haga clic dos veces en un componente del tipo de diseñador deseado en el árbol de componentes.
- Seleccione un componente del tipo de diseñador deseado en el árbol de componentes y pulse *Intro*.
- Haga clic con el botón derecho en un componente del tipo de diseñador deseado en el árbol de componentes y seleccione Activar el diseñador.

A medida que modifique elementos de menú en el diseñador de menús, los cambios se reflejan inmediatamente en el Inspector, en el árbol de componentes y en el código fuente. De forma análoga, cuando se realizan cambios en el código fuente, los cambios se reflejan en el IDE.

No es necesario guardar explícitamente el diseño de menú. Durante el proceso de trabajo, el diseñador de menús va generando el código que se guarda cuando se salve el archivo fuente `.java`. La próxima vez que abra el archivo `.java` y haga clic en un componente `MenuBar` del árbol de componentes, el diseñador de menús se abrirá y cargará de nuevo todos los elementos del componente.

## Terminología de menús

---

La terminología básica de las partes fundamentales de un menú es la siguiente:

- Un *menú* es una lista de opciones o elementos de menú que el usuario utiliza durante la ejecución.
- Los *elementos de menú* son las opciones que ofrecen los menús. Estos elementos pueden tener determinados atributos, como que estén desactivados (atenuados), cuando no están disponibles para el usuario, o que incluyan una marca de selección para activarlos/desactivarlos.
- La *barra de menús* aparece en la parte superior del marco y está formada por menús que contienen elementos individuales.
- Un *submenú* es un menú anidado que se abre haciendo clic en la flecha que aparece a la derecha de un determinado elemento de menú.
- Los *métodos abreviados de teclado* se visualizan a la derecha del elemento de menú y pueden ser específicos de una interfaz concreta. Por

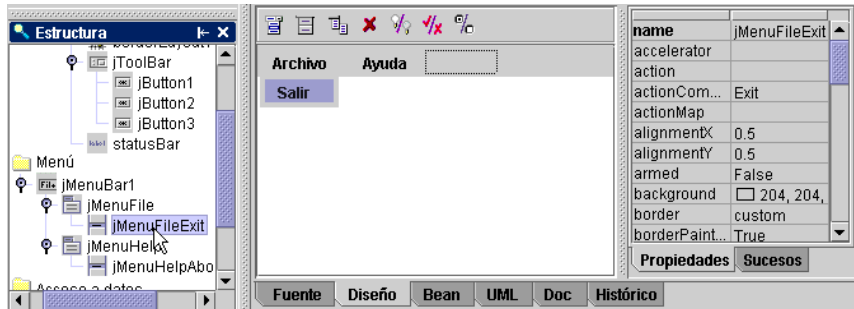


ejemplo, en muchos editores se utiliza *Ctrl+X* para indicar la operación Edición|Cortar.








- El *separador* es una línea en el menú que separa visualmente los diferentes grupos de elementos de menú.

## Herramientas de diseño de menús

Cuando se abre un elemento de menú en el diseñador de menús, el área de diseño cambia y presenta el siguiente aspecto:



Este diseñador de menús tiene su propia barra de herramientas y reconoce la pulsación de algunas teclas, como las teclas de cursor y las teclas *Insert* y *Supr.* La barra de herramientas del diseñador de menús contiene estas herramientas:

Herramienta	Acción
	Añade un espacio vacío para un nuevo menú a la izquierda del menú seleccionado o un nuevo elemento de menú por encima del elemento de menú seleccionado.
	Inserta una barra de separación inmediatamente encima del elemento de menú seleccionado.
	Crea un espacio vacío para un submenú anidado vacío y añade una flecha a la derecha del elemento de menú seleccionado.
	Borra el elemento de menú seleccionado (y todos sus submenús).
	Cambia los valores de la propiedad <code>enabled</code> del elemento seleccionado a <code>true</code> o <code>false</code> y lo atenúa mientras está desactivado ( <code>false</code> ).
	Permite utilizar una marca de verificación en el elemento de menú.
	Cambia el elemento de menú entre <code>JMenuItem</code> y <code>JRadioButtonMenuItem</code> .

**Nota** Haga clic con el botón derecho sobre un elemento de menú en el diseñador para ver un menú emergente que contiene muchos de estos comandos.

## Creación de menús

---

Los menús deben estar asociados a un contenedor `Marco` o `JMarco`. Para crear un menú en la aplicación, es necesario crear antes un archivo de componente frame. Para ello, realice una de las operaciones siguientes:

- Cree una aplicación con el Asistente para aplicaciones. En el Paso 2 del Asistente, marque Generar barra de menús.
- Abra el archivo de un componente frame (marco).
- Utilice el Asistente para marcos con el fin de añadir un archivo `Marco` al proyecto.

Para añadir un componente de menú a la interfaz de usuario:

- 1 Seleccione el archivo `Marco` o `JMarco` en el panel de proyecto.
- 2 Haga clic en la pestaña Diseño de la parte inferior del Visualizador de aplicaciones.
- 3 Seleccione el marco principal de la interfaz de usuario en la superficie de diseño o en el árbol de componentes.
- 4 Haga clic en el componente de menú deseado en la ficha AWT o en la ficha Contenedores Swing de la paleta de componentes.
- 5 Colóquelo en cualquier parte de la superficie de diseño o en el árbol de componentes.
  - El componente `MenuBar` o `JMenuBar` se adjunta al `Marco` principal de la interfaz de usuario y aparece en la parte superior de la aplicación.
  - El componente `PopupMenu` o `JPopupMenu` aparece cuando el usuario hace clic con el botón derecho del ratón en la interfaz de usuario. Los menús emergentes no tienen barra de menús.

El componente menú añadido muestra un nodo en el árbol de componentes y sus propiedades se muestran en el Inspector.

Por cada uno de los menús que desee incluir en la aplicación, debe añadir un componente de menú al contenedor de destino de la interfaz de usuario. El primer control `MenuBar` que se añade al contenedor de la interfaz de usuario se considera el control `MenuBar` actual de la interfaz de usuario. No obstante, en una aplicación se puede crear más de una barra de menús. Los nombres de las barras de menú se muestran en el Inspector en la propiedad `MenuBar` del marco. Para cambiar el `MenuBar` actual, seleccione uno de los menús de la lista desplegable de la propiedad `MenuBar`.

**Nota** Durante el diseño, los componentes de menú sólo están visibles en el diseñador de menús (no en el diseñador de interfaces de usuario). Sin embargo, puede verlos y seleccionarlos en todo momento en el árbol de componentes. Para ver qué aspecto tiene el menú en la interfaz de usuario, ejecute la aplicación.

## Adición de elementos de menú

---

Cuando se abre el diseñador de menús por primera vez, aparece un primer elemento de menú (vacío) marcado con un rectángulo punteado.

- 1 Escriba una etiqueta para el elemento de menú. (Para ver el cursor es necesario hacer doble clic dentro del rectángulo.)

Si la lista del menú está vacía, el campo tiene el ancho por defecto.

Cuando no está vacío, el campo adopta el ancho del elemento de menú más largo. Mientras escribe, el campo de texto se desplaza para adaptar las etiquetas que sobrepasan la longitud del campo de edición.

- 2 Pulse *Intro* o la tecla de *desplazamiento abajo* para añadir otro elemento de menú.

Se añade automáticamente un espacio vacío para el siguiente menú o elemento de menú.

JBuilder cambia automáticamente el nombre del elemento de menú en el árbol de componentes, para reflejar la nueva etiqueta.

- 3 Introduzca una etiqueta para cada elemento que desee añadir a la lista o pulse *Esc* para regresar a la barra de menús.

Puede utilizar las teclas de desplazamiento para moverse de la barra de menús a un menú y para moverse por los elementos de la lista.

## Inserción y eliminación de menús y elementos de menú

---

Para añadir un nuevo menú o elemento de menú en un menú existente, seleccione el rectángulo donde desea insertar el nuevo elemento de menú. Si se trata de un nuevo menú, se inserta en la barra de menús, a la izquierda del menú seleccionado; si es un elemento de menú, se incorpora justo encima del elemento seleccionado en la lista del menú. A continuación, realice una de las operaciones siguientes:



- Haga clic en el botón Desactivar de la barra de herramientas.
- Pulse la tecla *Insert*.
- Haga clic con el botón derecho y seleccione Insertar menú o Insertar elemento de menú.

Para borrar un elemento de menú, selecciónelo y realice una de las operaciones siguientes:



- Haga clic en el botón Borrar elemento de la barra de herramientas.
- Pulse la tecla *Supr*.

**Nota** A la derecha del último menú o debajo del último elemento de un menú, aparece un espacio vacío por defecto (que no puede borrar). Este no aparecerá en el menú durante la ejecución.

## Inserción de separadores

---

El separador introduce una línea horizontal entre elementos de menú. Puede utilizar los separadores para agrupar elementos de una lista de menú o sencillamente para marcar una división visual.

Para insertar un separador en un menú:

- 1 Seleccione el elemento de menú por encima del cual desee colocar el separador.



- 2 Haga clic en el botón Insertar separador de la barra de herramientas.

El separador se inserta encima del elemento de menú seleccionado.

## Definición de teclas aceleradoras

---

Las teclas aceleradoras permiten al usuario realizar una acción mediante una combinación de teclas. Por ejemplo, un método abreviado utilizado frecuentemente para Archivo|Guardar es *Alt + f, s*.

Para definir una tecla aceleradora de un elemento de menú:

- 1 Seleccione el elemento de menú en el diseñador de interfaces de usuario o en el árbol de componentes.
- 2 En el Inspector, seleccione la propiedad `accelerator` e introduzca un valor o seleccione una combinación de teclas en la lista desplegable. Esta lista es simplemente un subconjunto de las combinaciones válidas que puede utilizar.

Cuando se añade un método abreviado, aparece junto al elemento de menú sólo en tiempo de ejecución.

## Desactivación (atenuación) de elementos de menú

---

No es necesario eliminar un comando de menú para evitar que los usuarios accedan a determinadas funciones de un menú, cuando así lo exija el programa en una circunstancia específica. Por ejemplo, si no hay texto seleccionado en un documento, los elementos Cortar, Copiar y Borrar del menú Edición aparecen atenuados.

Para buscar una llamada a un método, siga uno de estos procedimientos:

- Haga clic en el botón Desactivar de la barra de herramientas.  
El botón Desactivar cambia el valor de la propiedad `enabled` de la opción de menú seleccionada de `true` (activada) a `false` (desactivada).
- En el Inspector, asigne el valor `false` a la propiedad `enabled` del elemento de menú. (A diferencia de la propiedad `visible`, `enabled` no oculta la opción de menú. Si tiene el valor `false`, se limita a atenuarla.)

## Creación de elementos de menú que utilicen marcas de selección

---

Para que un elemento de menú sea seleccionable debe, en primer lugar, cambiar el elemento de menú de un componente `JMenuItem` normal a un componente `JCheckboxMenuItem`. Los componentes `JCheckboxMenuItem` cuentan con una propiedad `state` (booleana) que determina si el suceso o el comportamiento asociados deben ejecutarse.

- La propiedad `state` del elemento seleccionado tiene el valor `true`.
- Y la de los elementos de menú no seleccionados tiene el valor `false`.

Si desea convertir un elemento de menú normal en un `JCheckboxMenuItem`, seleccione el elemento de menú y pulse el botón Comprobar en la barra de herramientas o haga clic con el botón derecho sobre el elemento de menú y seleccione Hacer seleccionable.



## Creación de elementos de menú con botones de radio Swing

---

El diseñador de menús permite crear elementos de menú Swing que formen parte de un `ButtonGroup` en el que solo puede seleccionarse un elemento del grupo. El elemento seleccionado muestra su estado de selección, haciendo que los demás elementos pasen al estado de no seleccionado.

Para crear un grupo de elementos de menú con botones de radio Swing:

- 1 En el diseñador de menús, cree un menú o menú anidado que contenga los elementos de menú que desea incluir en el grupo con botones de radio.
- 2 Haga clic con el botón derecho en todos estos elementos de menú en el diseñador y seleccione Conmutar elemento de radio. (También podría seleccionar el elemento de menú y pulsar el icono Botón de radio en la barra de herramientas del diseñador de menús.)
- 3 Pulse el botón Selección de Beans en la paleta de componentes para abrir el Visualizador de paquetes. (Si la lista Selección de Beans ya contiene paquetes, elija Seleccionar.)
- 4 Amplíe `javax.Swing` y haga doble clic en `ButtonGroup`. El cursor de selección incorpora `ButtonGroup`.
- 5 Haga clic en cualquier lugar del árbol o de la superficie de diseño. Esto añade un `buttonGroup1` a la carpeta Otros del árbol y añade la siguiente línea a las variables de clase:

```
ButtonGroup buttonGroup1 = new ButtonGroup();
```



- 6** Haga clic en la pestaña Fuente y, en el bloque `try` del constructor, a continuación de `jbInit()`, añada los `JRadioButtonMenuItem` a `buttonGroup1` de la siguiente manera:

```
//Construir el marco
public Marco1() {
    enableEvents(AWTEvent.WINDOW_EVENT_MASK);
    try {
        jbInit();
        buttonGroup1.add(jRadioButtonMenuItem1);
        buttonGroup1.add(jRadioButtonMenuItem2);
        buttonGroup1.add(jRadioButtonMenuItem3);
    }
    catch(Exception e) {
        e.printStackTrace();
    }
}
```

- 7** Pulse la pestaña Diseño y, en el Inspector, asigne la propiedad `selected` de uno de los elementos de menú con botones de radio a `true` (el que desee tener seleccionado por defecto). Se incorpora esta línea de código:

```
jRadioButtonMenuItem2.setSelected(true);
```

- 8** Si ahora se ejecuta el programa, aparecerá un botón de radio a la izquierda del elemento de menú cuya propiedad `selected` tiene el valor `true`. Si hace clic sobre alguno de los otros elementos de menú, el botón de radio se desplaza hacia el elemento recién seleccionado.

## Traslado de elementos de menú

---

En el diseñador de menús puede trasladar los menús y elementos de menú, sencillamente arrastrándolos y soltándolos con el ratón. Cuando se traslada un menú o un submenú, todos los elementos que contiene se trasladan con él.

Puede realizar los siguientes traslados:

- Menús dentro de una barra de menús
- Elementos de menú dentro de un menú
- Elementos de menús a otros menús
- Menús enteros para anidarlos dentro de otro elemento de menú. (Se convierten en submenús)
- Submenús a la barra de menús para crear otros menús
- Elementos de submenú a otros menús

La única limitación a estas operaciones tiene un fundamento jerárquico: no puede llevar un menú de la barra de menús hacia su propio contenido, ni trasladar un elemento de menú a su propio submenú. Sin embargo, puede llevar cualquier elemento a otro menú, independientemente de su posición original.

Para mover un menú o elemento de menú mediante el ratón:

- 1 Arrástrelo con el ratón hasta que la punta del cursor señale la nueva posición.

Para mover el menú o elemento de menú a otro menú, arrástrelo al menú de destino. Se abre el menú de destino.

- 2 Suelte el botón del ratón para colocar el elemento de menú en la nueva posición.

Para mover un menú o elemento de menú mediante el teclado, utilice el árbol de componentes.

### Consulte

- [“Desplazamiento de componentes” en la página 3-6](#) para obtener más información sobre el uso del árbol de componentes

## Creación de submenús

---

Muchas aplicaciones tienen menús que contienen listas desplegables situadas junto a un elemento de menú con el fin de proporcionar comandos adicionales relacionados. Este tipo de listas se indican con una flecha a la derecha del elemento de menú. JBuilder permite tantos niveles de menús anidados, o submenús, como desee incorporar al menú. Sin embargo, para realizar un diseño óptimo, es preferible que no utilice más de dos o tres niveles de menú en el diseño de la interfaz de usuario.

Cuando se traslada un menú de la barra de menús hacia otro menú, sus elementos se convierten en un submenú. Análogamente, si traslada un elemento de menú hacia otro submenú, sus elementos subordinados formarán otro menú anidado dentro del submenú.

Puede mover un elemento de menú hacia otro submenú o crear un sustituto en un nivel anidado junto a un elemento que ya exista y, a continuación, para anidar el menú, soltarlo sobre el sustituto.

Para crear un submenú:

- 1 Seleccione el elemento de menú dentro del cual desea crear un submenú y realice una de las operaciones siguientes:



- Pulse el botón Insertar submenú anidado en la barra de herramientas.
- Haga clic con el botón derecho en el elemento de menú y seleccione Insertar submenú anidado.

- 2 Escriba un nombre para el elemento de menú anidado o arrastre un elemento de menú existente sobre este sustituto.

- 3 Pulse *Intro* o Flecha abajo para crear el siguiente sustituto.

- 4 Repita los pasos 2 y 3 para cada elemento que desee crear en el menú anidado.

- 5 Pulse *Esc* para regresar al nivel anterior.

## Traslado de un menú a un submenú

---

También puede crear un submenú mediante la inserción de un menú de la barra de herramientas entre elementos de otro menú. Cuando se traslada un menú hacia una estructura de menús existente, todos sus elementos asociados se trasladan con él, creando un menú anidado intacto. Esto también es válido para desplazar submenús; si se traslada un submenú a otro submenú, se crea otro nivel de anidación.

## Vinculación de sucesos de menú al código

---

Los elementos de menú sólo disponen de un suceso: `actionPerformed`. El código que se añade al suceso `actionPerformed` de un elemento de menú se ejecuta cuando el usuario selecciona el elemento de menú o cuando utiliza su método abreviado de teclado.

Para añadir código al suceso `actionPerformed` de un elemento de menú:

- 1 Seleccione un elemento de menú en el diseñador de menús.
- 2 En el Inspector, seleccione la pestaña Sucesos.
- 3 Haga doble clic en la columna de `actionPerformed` para crear en el código fuente el esqueleto de un método de gestión de sucesos, que llevará un nombre por defecto.

**Nota**

Para redefinir el nombre por defecto del método de manejador de sucesos de `actionPerformed`, haga clic una vez en el campo de valor del suceso, escriba un nombre nuevo para el método de suceso y pulse *Intro*.

Si hace doble clic sobre el valor del suceso se muestra el código fuente. El cursor se sitúa en el nuevo método de gestión de sucesos de `actionPerformed`, donde puede escribirse el código.

- 4 Dentro de las llaves de apertura y cierre, introduzca o modifique el código que desea que se ejecute cuando el usuario haga clic en este comando de menú.

### Ejemplo: Apertura de un cuadro de diálogo desde un elemento de menú

Para presentar el cuadro de diálogo de apertura de archivos cuando el usuario seleccione ArchivoAbrir:

- 1 Cree un menú Archivo con un elemento de menú Abrir.
- 2 En la ficha Contenedores Swing de la paleta de componentes, seleccione el componente `JFileChooser` y colóquelo en la carpeta IU del árbol de componentes.
- 3 En el diseñador de menús o en el árbol de componentes, seleccione el elemento de menú Abrir.
- 4 En el Inspector, seleccione la ficha Sucesos.



- 5 Seleccione el suceso `actionPerformed` y haga doble clic para generar el siguiente esqueleto del método de gestión de sucesos en el código fuente, con el cursor en la ubicación adecuada, a la espera de que escriba:

```
void jMenuItem1_actionPerformed(ActionEvent e) {
    |
}
```

**Nota** JBuilder muestra el método de gestión de sucesos, si existiera.

- 6 Dentro del cuerpo del método `actionPerformed`, escriba lo siguiente:

```
jFileChooser1.showOpenDialog(this);
```

- 7 En un programa real, añadiría normalmente varias líneas de código personalizado en los métodos de gestión de sucesos. En este ejemplo, es posible que desee obtener el nombre de archivo introducido por el usuario y utilizar este valor para abrir o manipular ese archivo.

### Consulte

- [“Vinculación de código de tratamiento de sucesos” en la página 4-2](#)
- [“Conexión de controles y sucesos” en la página 4-3](#)

## Creación de menús emergentes

---

Para crear un menú emergente:

- 1 En el diseñador, seleccione un componente `PopupMenu` de la ficha AWT de la paleta de componentes o un componente `JPopupMenu` de la ficha Contenedores Swing y suéltelo en el árbol de componentes. Este es el elemento seleccionado del árbol.
- 2 Pulse *Intro* con `JPopupMenu` o `PopupMenu` seleccionado en el árbol de componentes, para abrir el diseñador de menús.
- 3 Añada uno o más elementos de menú.
- 4 Seleccione `this(BorderLayout)` en el árbol de componentes y pulse *Intro* para volver al diseñador.
- 5 Seleccione el panel u otro componente a cuyo suceso desee conectar el menú emergente, de modo que pueda ver ese componente en el inspector. En el ejemplo que aparece a continuación, se seleccionó `jPanel1`.
- 6 Haga clic en la pestaña Sucesos del Inspector.
- 7 Haga doble clic en el suceso con el cual desea que aparezca el menú emergente. En el siguiente ejemplo se seleccionó el suceso `MouseClicked`.
- 8 Edite el código del esqueleto del manejador de sucesos de forma que se parezca al siguiente:

## Creación de menús emergentes

```
void jPanel1_mouseClicked(MouseEvent e) {
    jPanel1.add(jPopupMenu1); // Debe añadir JPopupMenu al componente
                                // cuyo suceso está seleccionado.
    // En el caso de este suceso de ejemplo, se comprueba al hacer clic
    con el botón derecho del ratón.
    if (e.getModifiers() == Event.META_MASK) {
        // Hacer visible el jPopupMenu relativo a la posición actual del
        ratón
        // en el contenedor.
        jPopupMenu1.show(jPanel1, e.getX(), e.getY());
    }
}
```

- 9** Añada tantos manejadores de sucesos a los elementos del menú emergente como necesite su aplicación.

# Capítulo 7

## Temas avanzados

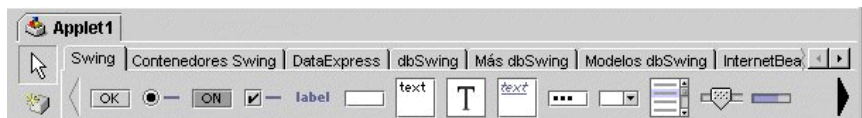
Es posible personalizar la paleta de componentes con el objeto de incluir más bibliotecas de componentes y JavaBeans individuales. Si un componente no cumple unos criterios específicos, puede aparecer como un bean rojo. Normalmente, las aplicaciones distribuidas requieren la serialización, el uso de personalizadores y la gestión de cadenas de conjunto de recursos. En la presente documentación se explica cómo se realizan esas tareas en JBuilder.

### Gestión de la paleta de componentes

---

La paleta de componentes proporciona un acceso rápido a todos los componentes de la vía de acceso a clases `CLASSPATH`. Por defecto, JBuilder muestra todos los componentes de la `CLASSPATH`, ordenados por bibliotecas. Por ejemplo, la pestaña Swing de la paleta muestra los componentes de la biblioteca Swing.

JBuilder incluye varias bibliotecas de componentes. Es posible elegir entre ellas por medio de las distintas pestañas de la paleta de componentes, el cuadro de diálogo Añadir componente y el botón Selección de Beans. Puede añadir componentes a las bibliotecas existentes o crear otras para ellos.



Es posible que desee instalar componentes adicionales incluidos con JBuilder, componentes que usted mismo haya creado o componentes de

terceras personas. Las siguientes secciones explican cómo instalar componentes y fichas adicionales en la paleta, eliminar los que no se utilicen y personalizar la paleta.

### Consulte

- “Las bibliotecas” en *Creación de aplicaciones con JBuilder*

## Cómo añadir un componente a la paleta de componentes

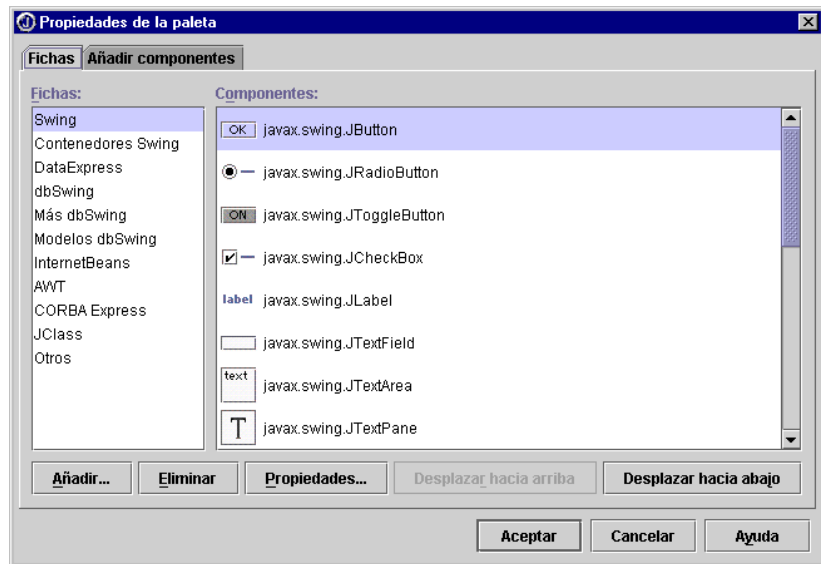
---

Si el componente es un JavaBean, puede añadirlo a la paleta de componentes.

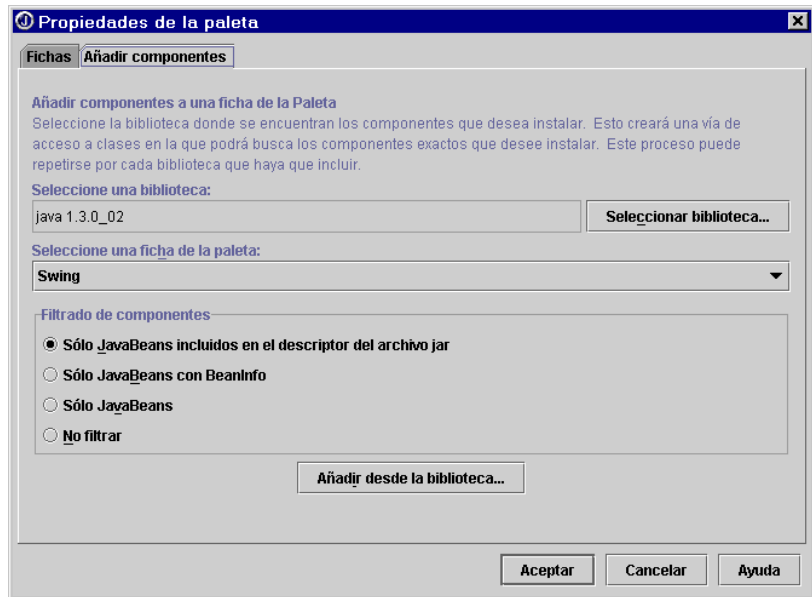
Si varios proyectos van a compartir el componente, se debe añadir a una biblioteca que se encuentre en la *vía de acceso a clases* en todos los proyectos en que se vaya a utilizar.

Para colocar un componente en la paleta de componentes:

- 1 Seleccione Herramientas|Configurar paleta o haga clic con el botón derecho del ratón en una pestaña de la paleta de componentes y elija Propiedades, para abrir el cuadro de diálogo Propiedades de la paleta.
- 2 Seleccione la pestaña Fichas. En la columna Fichas, seleccione dónde desea que aparezca el componente o haga clic en el botón Añadir para crear otra ficha.



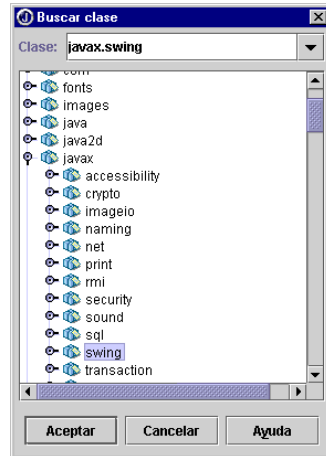
- 3 Haga clic en la pestaña Añadir componentes para seleccionar los que desee añadir.



- 4 Pulse el botón Seleccionar biblioteca para abrir el cuadro de diálogo Seleccionar otra biblioteca. Seleccione una biblioteca de la lista o cree una pulsando Nuevo y utilizando el Asistente para bibliotecas. Haga clic en Aceptar para cerrar el cuadro de diálogo.
- 5 Seleccione la ficha de la paleta donde desee que se añadan los componentes.
- 6 Seleccione una opción de filtrado:
- Sólo JavaBeans incluidos en el descriptor del archivo jar: añade automáticamente las clases JavaBeans a los archivos JAR de la biblioteca de la ficha seleccionada en la paleta de componentes.
  - Sólo JavaBeans con BeanInfo: muestra una lista de JavaBeans con BeanInfo en el cuadro de diálogo Buscar clase cuando está pulsado el botón Añadir desde la biblioteca.
  - Sólo JavaBeans: muestra una lista de JavaBeans sólo en el cuadro de diálogo Buscar clase cuando está pulsado el botón Añadir desde la biblioteca.
  - No filtrar: muestra una lista sin filtrar de todas las clases en el cuadro de diálogo Buscar clase cuando está pulsado el botón Añadir desde la biblioteca.
- 7 Pulse el botón Añadir desde la biblioteca.
- Si ha seleccionado la opción Sólo JavaBeans incluidos en el descriptor del archivo jar, se añaden los JavaBeans automáticamente en la ficha

seleccionada. Continúe hasta el último paso para llevar a cabo los cambios.

- Si ha seleccionado una de las demás opciones, aparece el cuadro de diálogo Buscar clase. Seleccione las clases deseadas en el cuadro de diálogo Buscar clase y pulse Aceptar. Aparece un cuadro de diálogo Resultado con las clases en una lista. Pulse Aceptar.



- 8 Pulse Aceptar para cerrar el cuadro de diálogo Propiedades de la paleta y guardar los cambios.

El gestor de la paleta de componentes de JBuilder es una OpenTool que permite añadir clases propias. No olvide que el material relativo a los procesos del programa (acceso a base de datos, conexiones por medio de sockets con otros ordenadores, interacciones de objetos distribuidos que utilizan RMI o CORBA, cálculos, etc.) se deben colocar en clases que no pertenezcan a la interfaz de usuario.

## Selección de una imagen para un botón de la paleta de componentes

---

La imagen de un botón de la paleta de componentes puede ser una de estas tres cosas:

- Una imagen definida en la clase BeanInfo del bean.
- Un archivo .gif seleccionado a través del cuadro de diálogo Propiedades del elemento.
- Una imagen por defecto suministrada por JBuilder, si no se utiliza ninguna de las anteriores.

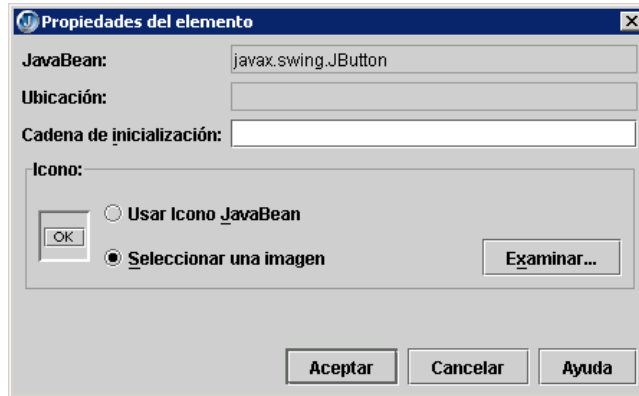
Si desea seleccionar una imagen para el botón de su paleta de componentes:

- 1 Seleccione Herramientas|Configurar paleta o haga clic con el botón derecho en la paleta de componentes y seleccione Propiedades.

Estas dos acciones abren el cuadro de diálogo Propiedades de la paleta.

- 2 Seleccione la pestaña Fichas.
- 3 Seleccione la biblioteca apropiada en la columna Fichas y el componente en la columna Componentes.
- 4 Haga doble clic en el componente o pulse Propiedades.

Aparece el cuadro de diálogo Propiedades del elemento.



**Nota** Aparece una imagen del componente seleccionado en el cuadro de diálogo Propiedades del elemento, a la izquierda de las opciones del Icono.

- 5 Realice una de las operaciones siguientes:
  - Seleccione Usar Icono JavaBean para utilizar en el botón la imagen con que cuenta el bean.
  - Elija Seleccionar una imagen y pulse el botón Examinar para seleccionar el archivo .gif que aparecerá en el botón.
  - Escriba en una cadena de inicialización diferente si desea utilizar un método de factoría o un constructor con más de un parámetro, en lugar del constructor de Bean sin parámetros por defecto.

Para obtener el mejor resultado, utilice un archivo .gif de 32x32.

- 6 Pulse Aceptar para cerrar el cuadro de diálogo Propiedades del elemento.
- 7 Pulse Aceptar en el cuadro de diálogo Propiedades de la paleta cuando haya finalizado.

## Cómo añadir un componente a la paleta de componentes

Puede añadir una ficha de componentes a la paleta de componentes si tiene la biblioteca de componentes en su vía de acceso. Para añadir una ficha a la paleta:

- 1 Seleccione Herramientas|Configurar paleta, o haga clic con el botón derecho en la paleta de componentes y seleccione Propiedades.

- 2 Elija la pestaña Fichas del cuadro de diálogo Propiedades de la paleta.
- 3 Pulse el botón añadir para abrir el cuadro de diálogo Añadir ficha.
- 4 Escriba el nombre de la ficha en el campo Nombre de ficha.
- 5 Pulse Aceptar.
- 6 Seleccione la nueva ficha en la columna Fichas y pulse el botón Desplazar hacia arriba para llevarla a la ubicación deseada en la paleta.

Las fichas que se añaden a la paleta de componentes también se muestran en el cuadro de diálogo Añadir componente.

## Eliminación de una ficha o un componente de la paleta de componentes

---

Para eliminar una ficha o un componente de la paleta:

- 1 Seleccione Herramientas|Configurar paleta o haga clic con el botón derecho en la paleta de componentes y seleccione Propiedades.
- 2 Elija la pestaña Fichas del cuadro de diálogo Propiedades de la paleta.
- 3 Seleccione la ficha en la columna Fichas y el componente en la columna Componentes.
- 4 Pulse Eliminar y después Aceptar.

Las fichas y componentes eliminados de la paleta de componentes ya no aparecen en el cuadro de diálogo Añadir componente.

**Nota** Cuando se elimina un componente de la paleta se borra únicamente el acceso directo. No se borra el componente de su biblioteca. Cuando se elimina un componente o una ficha de la paleta, sigue siendo posible acceder a él desde las fichas Examinar y Buscar del cuadro de diálogo Añadir componente.

## Reorganización de la paleta de componentes

---

Para cambiar el orden de las fichas o de los componentes de la paleta:

- 1 Seleccione Herramientas|Configurar paleta, o haga clic con el botón derecho en la paleta de componentes y seleccione Propiedades.
- 2 Seleccione la pestaña Fichas.
- 3 Seleccione una ficha en la columna Fichas o un componente en la columna Componentes.
- 4 Pulse Desplazar hacia arriba o Desplazar hacia abajo para desplazar el elemento seleccionado a una nueva ubicación.
- 5 Pulse Aceptar cuando termine.



## Manejo de beans rojos

---

Si JBuilder no puede crear la instancia de un componente para el diseñador, el componente aparece como un cuadro rojo con el nombre de clase en la parte superior. Esto se conoce como *bean rojo*. Los beans rojos aparecen siempre que:

- 1 La clase o su constructor no son `públicos`.
- 2 La clase ha lanzado excepciones sobre todos los constructores que JBuilder ha probado.
- 3 La clase es el objeto `this` y amplía una clase `abstracta`.

En los dos primeros casos se puede solucionar suministrando un constructor `public` por defecto.

El tercer caso es más complejo. Cuando JBuilder necesita instanciar un objeto para que sea el objeto `this`, no puede instanciar el objeto que se esté diseñando. Para evitar este problema, JBuilder instancia la superclase del objeto `this`. Pero si la superclase es `abstracta`, no se puede instanciar. Es necesario proporcionar una clase proxy concreta (no `abstracta`). Si es eso lo que necesita llevar a cabo, consulte las Notas de esta versión en el directorio que contiene su instalación de JBuilder.

## La serialización

---

*Serializar* un objeto es el proceso de convertirlo en una secuencia de bytes y guardarlo como archivo en un disco o enviarlo a través de la red. Cuando se efectúa una solicitud para recuperar un objeto de un archivo, o de otro punto de la red, la secuencia de bytes se *paraleliza* y recupera su estructura original.

En el caso de los JavaBeans, la serialización permite guardar fácilmente el estado inicial de todas las instancias de un tipo de clase o un bean. Si el bean se serializa en un archivo y después se paraleliza, la próxima vez que se utilice se recupera exactamente como lo dejó el usuario.

### Advertencia

Tenga mucho cuidado cuando serialice componentes. No lo intente si no conoce exactamente las implicaciones y ramificaciones del proceso.

### Consulte

- “Serialización” en *Procedimientos iniciales con Java*
- El artículo “Object Serialization” de Sun (en inglés) en <http://java.sun.com/j2se/1.3/docs/guide/serialization/>

## Serialización de componentes en JBuilder

---

JBuilder simplifica el proceso de serialización de JavaBeans en el diseñador. En primer lugar, modifique el bean con el Inspector para establecer las opciones que desee conservar. A continuación, siga estas instrucciones:

- 1 Seleccione el componente en el árbol y haga clic en él con el botón derecho del ratón.
- 2 Elija Serializar en el menú que aparece.

Aparece un cuadro de mensaje que muestra la vía de acceso y el nombre del archivo serializado. Tiene el mismo nombre que el componente original y una extensión `.ser`.
- 3 Haga clic en Aceptar para crear el archivo serializado. Ocurre lo siguiente:
  - Aparece un cuadro de diálogo de confirmación si el archivo ya existe.
  - El archivo de serialización `.ser` se crea tomando como raíz el directorio especificado en la vía de acceso a archivos fuente y crea el subdirectorio de paquete correspondiente (por ejemplo `misproyectos/java/awt`).
  - El archivo de serialización se añade al proyecto como nodo, de forma que pueda distribuirse junto con el proyecto.
  - Durante la compilación, JBuilder copia el archivo `.ser` de la Vía de acceso a archivos fuente a la Vía de salida.

La próxima vez que instancie ese bean mediante `Beans.instantiate()`, el archivo `.ser` estará guardado en memoria.

## Serialización de un objeto `this`

---

Para serializar, se necesita una instancia activa del objeto. Esto plantea un problema en el caso de los objetos `this`, porque no se instancian de la misma forma que los componentes añadidos al diseñador. Por tanto, es necesario otro método para obtener una instancia activa de `this`.

Es posible realizar la serialización escribiendo el código necesario, pero, generalmente, los objetos se serializan después de modificarlos mediante una herramienta RAD (por ejemplo, un inspector) o un personalizador.

### Ejemplo

El siguiente ejemplo utiliza un marco de IU `this`. Estas instrucciones pueden modificarse con el fin de que se puedan aplicar a cualquier tipo de objeto, como un panel, una clase, un cuadro de diálogo o un menú.

- 1 Abra el proyecto y cree la clase que desea serializar (`Marco1.java`, para este ejemplo).
- 2 Guárdela y compílela.

- 3 Abra otro archivo del proyecto que contenga una clase que haya sido o pueda ser diseñada visualmente con JBuilder (`OtherFile.java`, en este ejemplo).
- 4 Seleccione `OtherFile.java` en el panel de proyectos y cree la siguiente declaración de campo (variable de instancia) en la declaración de la clase, a continuación de las restantes declaraciones:

```
Frame1 f = new Frame1();
```

- 5 Haga clic en la pestaña Diseño para abrir el diseñador de interfaces de usuario en `OtherFile.java`.
- 6 Haga clic con el botón derecho en la variable de instancia "f" de la carpeta Otros del árbol de componentes y seleccione Serializar. Aparece un mensaje que indica la vía de acceso y el nombre del archivo serializado.
- 7 Pulse Aceptar.

Para emplear el archivo serializado cuando instancie `Marco1` en la aplicación, debe realizar la instanciación en el archivo `Application` mediante `Beans.instantiate()`, de la siguiente forma.

## 1 Cambie el constructor de `Aplicacion1.java` de

```
public Application1() {
    Marco1 frame = new Marco1();
    //Valida los marcos que tienen tamaños preestablecidos
    //Empaquetar marcos con información de tamaño preferente útil, p. ej. del
    diseño
    if (packFrame)
        frame.pack();
    else
        frame.validate();
    frame.setVisible(true);
}
```

a

```
public Application1() {
    try {
        Frame1 frame = (Frame1)Beans.instantiate(getClass().getClassLoader(),
            Frame2.class.getName());
        //Valida los marcos que tienen tamaños preestablecidos
        //Empaquetar marcos con información de tamaño preferente útil, p. ej. del
        diseño
        if (packFrame)
            frame.pack();
        else
            frame.validate();
        frame.setVisible(true);
    }
    catch(Exception x){
    }
}
```

## 2 Añada la siguiente sentencia import en la parte superior de

`Aplicacion1.java`:

```
import java.beans.*;
```

## Utilización de personalizadores en el diseñador

---

Los JavaBeans pueden designar a su propio *personalizador*, que es un editor diseñado especialmente para la clase del bean. En la fase de diseño en JBuilder, haga clic con el botón derecho del ratón en el componente, en el árbol de componentes, y seleccione Personalizador; los JavaBeans que tengan un personalizador muestran su cuadro de diálogo. Este cuadro de diálogo es un contenedor del personalizador y permite modificar el objeto como se haría en un editor de propiedades abierto desde el Inspector.

El personalizador no sustituye al Inspector de JBuilder. El JavaBean se limita a ocultar las propiedades que no debe mostrar el Inspector.

### Modificación de beans mediante personalizadores

---

Una forma de conservar los cambios de un bean personalizado es serializarlo. Por tanto, siempre que se cierre el cuadro de diálogo del personalizador, JBuilder le preguntará si desea guardar el objeto en otro archivo `.ser` (o sobrescribir el anterior).

- Si se ha cargado un bean en el diseñador de interfaces de usuario mediante `Beans.instantiate()` y se introducen más modificaciones mediante el personalizador, puede guardarse de nuevo, con el mismo nombre o en un archivo nuevo.
- Si se ha cargado un bean en el diseñador de interfaces de usuario mediante `Beans.instantiate()` y se introducen más modificaciones mediante el Inspector, JBuilder recurrirá a la técnica estándar de escribir el código fuente correspondiente a las modificaciones de las propiedades. En consecuencia, es posible introducir mediante el Inspector otros cambios en los beans personalizados (y serializados).

JBuilder intenta generar el código correspondiente a los cambios introducidos en las propiedades mediante el personalizador. Esto puede bastar o no, en función del nivel de cumplimiento de la especificación de JavaBeans que presente el personalizador. Ciertos personalizadores elaborados introducen cambios en los objetos que no pueden representarse en el código, con lo que la serialización representa el único medio de guardarlos.

#### Consulte

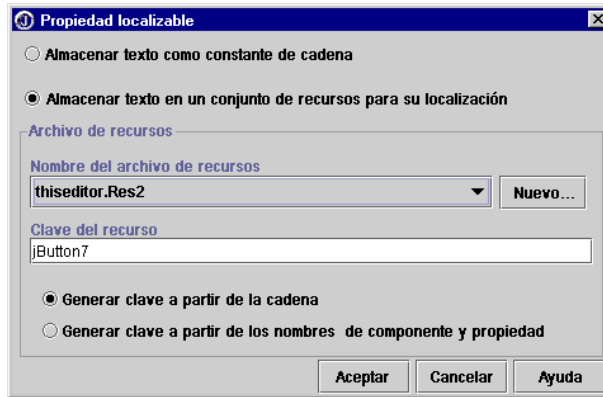
- “Creación de JavaBeans con BeansExpress” en *Creación de aplicaciones con JBuilder*.
- “Bean Customization” (en inglés) en <http://java.sun.com/docs/books/tutorial/javabeans/customization/index.html>.

# Gestión de cadenas de conjunto de recursos

**Es una función de  
JBuilder Developer y  
Enterprise.**

Puede utilizar el Inspector para eliminar el valor de `cadena` de una propiedad de un archivo de `conjunto de recursos` (ResourceBundle) tras haber extraído los recursos del proyecto, o puede añadir un valor de `cadena` a un archivo de `conjunto de recursos` nuevo o existente.

Haga clic con el botón en una propiedad que admita un valor de `cadena` y seleccione `Conjunto de recursos`. Se abre el cuadro de diálogo `Propiedad localizable`.



Para eliminar una `cadena` extraída de un `conjunto de recursos`:

- 1 Seleccione el nombre de archivo del `conjunto de recursos` en la lista del cuadro de diálogo `Propiedad localizable`.
- 2 Seleccione `Almacenar texto como constante de cadena`. De esta forma, la `cadena` extraída se elimina del archivo de recursos.

Para añadir un valor de `cadena` a un archivo de `conjunto de recursos` nuevo o existente:

- 1 Seleccione la opción `Almacenar texto en un conjunto de recursos para su localización`.

Cuando el proyecto todavía no tiene un `conjunto de recursos`, al seleccionar esta opción aparece el cuadro de diálogo `Crear conjunto de recursos`.

Cuando ya existe un `conjunto de recursos`, se convierte en el preestablecido. Para cambiarlo, pulse `Nuevo`.

En cualquier caso, aparece el cuadro de diálogo `Crear conjunto de recursos`:

- a Seleccione o modifique el nombre del `conjunto de recursos`.
- b Seleccione el tipo de `conjunto de recursos`.
- c Pulse `Aceptar`.

Aparece el nuevo conjunto de recursos en el cuadro de diálogo Propiedad localizable.

**2** Indíquele a JBuilder cómo generar la clave. Elija una de las opciones siguientes:

- Generar clave a partir de la cadena: utilice el valor de la cadena desde la que accedió a este cuadro de diálogo.
- Generar clave a partir de los nombres de componente y propiedad: utilice los nombres del componente y de la propiedad desde los que accedió a este cuadro de diálogo, en lugar de la cadena de la propiedad.

La clave que genera la opción aparece en el campo Clave del recurso.

**3** Pulse Aceptar.

**Importante** Cuando un elemento extraído se elimina de la interfaz de usuario, no se borra automáticamente de su archivo de recursos. De esta forma se evita que se pierda la referencia en caso de que la clave se utilice en otra parte del código. Debe abrir el archivo y eliminar manualmente la entrada.

### Consulte

- “Internacionalización de programas con JBuilder” en *Creación de aplicaciones con JBuilder*
- Asistente para extracción de recursos en la ayuda en línea (Asistentes Recursos de cadenas)

## Gestores de diseño

Un programa escrito en Java puede distribuirse en varias plataformas. Si tiene que utilizar técnicas estándar para el diseño de interfaces de usuario y especificar las posiciones y tamaños absolutos de los componentes de la interfaz, es posible que no sea posible trasladarla. Lo que tiene un buen aspecto en su sistema de desarrollo puede no ser válido en otra plataforma. Para solucionar este problema, Java cuenta con un sistema de gestores de diseño portable. Estos gestores se utilizan para especificar las normas y restricciones del diseño de la interfaz de usuario, de una forma que permitirá portarla a otras plataformas.

Los gestores de diseño ofrecen las siguientes ventajas:

- Ubicación correcta de los componentes, con independencia de las fuentes, las resoluciones de pantalla y las plataformas.
- Ubicación inteligente de componentes, para los contenedores que se redimensionan dinámicamente durante la ejecución.
- Facilidad de conversión. Si una cadena crece en extensión, los componentes permanecen correctamente alineados.

### Acerca de los gestores de diseño

---

Los contenedores de interfaz de usuario de Java (`java.awt.Container`) utilizan un objeto especial, denominado *gestor de diseño*, para controlar la forma en que se posicionan y se dimensionan los componentes del contenedor cada vez que este se visualiza. El gestor de diseño organiza automáticamente los

componentes de un contenedor siguiendo un conjunto concreto de normas específicas del gestor.

El gestor de diseño define los tamaños y las posiciones de los componentes basándose en determinados factores, como por ejemplo:

- Las normas de diseño del gestor de diseño.
- Los valores de las propiedades del gestor de diseño, si las hay.
- Las restricciones de diseño asociadas a cada componente.
- Ciertas propiedades comunes a todos los componentes, como `preferredSize`, `minimumSize`, `maximumSize`, `alignmentX` y `alignmentY`.
- El tamaño del contenedor.

Cada gestor de diseño tiene sus ventajas e inconvenientes. Es posible elegir entre varios el más adecuado para los requisitos de cada contenedor.

Al crear un contenedor en un programa Java, puede aceptar el gestor de diseño por defecto de ese tipo de contenedor o redefinir el valor por defecto, especificando otro tipo de gestor de diseño.

Normalmente, si crea manualmente el código de la interfaz de usuario, redefinirá el gestor de diseño por defecto antes de añadir componentes al contenedor. Puede cambiar el diseño en cualquier momento mientras utiliza el diseñador. JBuilder adapta inmediatamente el código según haga falta. Si desea cambiar el diseño, puede añadir un gestor de diseño al código fuente del contenedor o seleccionar un diseño en la lista de la propiedad `layout` del contenedor en el Inspector.

**Importante** No es posible modificar las propiedades `layout` de un diseño <por defecto>. Si desea modificar las propiedades del gestor de un contenedor, ha de identificar explícitamente el gestor para acceder a sus propiedades en el Inspector.

## Diseños null y XYLayout

---

El gestor de diseño se utiliza en función del diseño global del contenedor que desea. Pero es posible que la utilización de algunos diseños con el diseñador resulte difícil, dado que toman inmediatamente el control de la posición y el tamaño de los componentes tan pronto como los añade al contenedor. Para mitigar este efecto mientras está diseñando una IU en el diseñador, puede utilizar un diseño `null` o el diseño personalizado de JBuilder de nombre `XYLayout`. Ambos dejan los componentes exactamente donde se colocan y permiten especificar el tamaño.



Existen ciertas diferencias entre estos dos tipos de diseño:

- `XYLayout` recibe el valor de la propiedad `preferredSize` de los componentes, de modo que si se ha definido `preferredSize` para el componente (valor -1), `XYLayout` adapta el tamaño del componente al aspecto propio del sistema. Esto no se puede hacer con el diseño `null`.
- `null` sobrecarga el código con llamadas a `setBounds()`. `XYLayout` no.

Si se empieza con `null` o `XYLayout` resulta más fácil crear el prototipo en el contenedor. Más adelante, después de añadir los componentes al contenedor, puede activar un diseño portable adecuado para su aplicación.

#### Importante

Recuerde convertir todos los contenedores de `null` o `XYLayout` antes de distribuir.

El diseño utiliza el posicionamiento absoluto, por lo que los componentes no se ajustan correctamente al cambiar el tamaño de los contenedores antecesores. Estos diseños no se ajustan correctamente a las distintas configuraciones de los usuarios y los sistemas y, por lo tanto, no son portables. Consulte [“XYLayout” en la página 8-8](#) para obtener más información.

Se pueden utilizar paneles anidados en algunos diseños, para agrupar componentes del marco principal, utilizando distintos diseños para el componente `Frame` y para los distintos paneles.

Pruebe los distintos diseños para ver qué efecto tienen sobre los componentes del contenedor. Si comprueba que el gestor de diseño seleccionado no proporciona los resultados que desea, pruebe con otro o anide varios paneles con diseños distintos para obtener el efecto deseado.

Para obtener una explicación más detallada de cada uno de los diseños, consulte los temas individuales de los diseños en [“Diseños suministrados con JBuilder” en la página 8-12](#).

#### Consulte

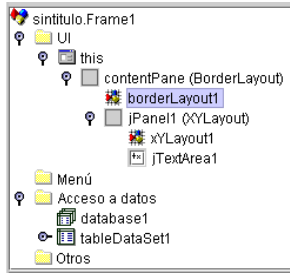
- [“XYLayout” en la página 8-8](#)
- [“Paneles y diseños anidados” en la página 8-53](#)
- [“Laying out components within a container” en Java Language Tutorial \(en inglés\) en <http://java.sun.com/docs/books/tutorial/uiswing/index.html>](#)

## Conceptos básicos acerca de las propiedades de los diseños

---

Normalmente, todos los contenedores cuentan con algún tipo de gestor de diseño asociado a su propiedad `layout`. El gestor de diseño tiene propiedades que pueden afectar al tamaño y la posición de todos los componentes que se añadan al contenedor. Estas propiedades pueden verse y editarse con ayuda del Inspector tras seleccionar el gestor de diseño en el árbol de componentes.

El gestor de diseño aparece como un elemento en el árbol, justo debajo del contenedor al que está vinculado.



## Conceptos básicos acerca de las restricciones de los diseños

---

Para cada componente que se añade a un contenedor, JBuilder puede crear una instancia del objeto `constraints` o generar un valor de restricción que proporciona información adicional acerca de cómo debe dimensionar y posicionar el gestor de diseño ese componente en particular. El tipo de objeto o valor de restricción creado depende del tipo de gestor de diseño utilizado. El Inspector muestra las restricciones de los componentes como si se trataran de propiedades del propio componente, lo que permite su edición.

## Ejemplos de propiedades y restricciones de diseños

---

A continuación, se ofrecen ejemplos de propiedades y restricciones de diseño:

- `BorderLayout` dispone de las propiedades `hgap` (separación horizontal) y `vgap` (separación vertical) que determinan la distancia entre los componentes; éstos cuentan con una restricción, denominada `constraints` en el Inspector, cuyos valores posibles son `NORTH`, `SOUTH`, `EAST`, `WEST` o `CENTER`.
- `FlowLayout` y `GridLayout` cuentan con propiedades que puede utilizar para modificar la alineación de los componentes o la separación horizontal o vertical existente entre ellos.
- `GridLayout` cuenta con propiedades que permiten especificar el número de filas y columnas.
- `GridBagLayout` carece de propiedades. No obstante, cada componente que se coloca en un contenedor `GridBagLayout` tiene asociado un objeto de restricción que dispone de numerosas propiedades relativas a la ubicación y el aspecto del componente, como:
  - La altura y anchura del componente
  - El lugar de la celda al que se ancla el componente
  - La forma en que el componente se expande en la celda
  - El tamaño adicional que rodea al componente en la celda

# Diseños de contenedores

JBuilder incluye la propiedad `layout` en el Inspector para contenedores. Puede seleccionar fácilmente un nuevo diseño para cualquier contenedor del diseñador.

Para seleccionar otro diseño:

- 1 Seleccione el contenedor en el árbol de componentes.
- 2 Haga clic en la pestaña Propiedades del Inspector y seleccione la propiedad `layout`.
- 3 Haga clic en la flecha Abajo que aparece al final del campo de valor de la propiedad `layout` y seleccione un diseño en la lista desplegable.

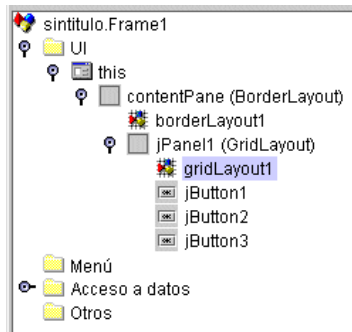
JBuilder realiza las operaciones siguientes:

- Sustituye el nuevo gestor de diseño en el árbol de componentes.
- Modifica el código fuente para añadir el nuevo gestor de diseño y actualiza la llamada del contenedor a `setLayout`.
- Cambia el diseño de los componentes en el diseñador.
- Actualiza las restricciones de diseño de los componentes del contenedor, tanto en el Inspector como en el código fuente.

## Modificación de las propiedades de los diseños

Para modificar las propiedades de un diseño en el Inspector:

- 1 En el árbol de componentes, seleccione el diseño que desea modificar. JBuilder muestra el diseño del contenedor justo debajo del contenedor en el árbol. Por ejemplo, en la siguiente figura aparece seleccionado `gridLayout1` de `jPanel1`.



- 2 En el Inspector, seleccione la ficha Propiedades y modifique los valores de las propiedades del diseño. Por ejemplo, en un diseño `GridLayout`, puede

cambiar el número de columnas o de filas de la rejilla, además del espacio horizontal y vertical que los separa.

name	gridLayout1
columns	0
hgap	0
rows	1
vgap	0
Propiedades    Sucesos	

El diseñador muestra los cambios inmediatamente y JBuilder modifica el método `jbInit()` del código fuente.

## Modificación de restricciones de diseños de componentes

Cuando se añade un componente a un contenedor, JBuilder crea un objeto o un valor de restricción adecuado para el gestor de diseño del contenedor. JBuilder inserta automáticamente este valor de restricción o este objeto en la propiedad `constraint` del componente desde el Inspector. También lo añade al código fuente como un parámetro de la llamada al método `add()` del método `jbInit()`.

Para editar las restricciones de diseño de un componente:

- 1 Selecciónelo en la superficie de diseño o en el árbol de componentes.
- 2 Seleccione la propiedad `constraints` en el Inspector.
- 3 Utilice la lista desplegable o el editor de propiedades para modificar las restricciones.

## Conceptos básicos acerca de las propiedades de tamaño

Los gestores de diseño utilizan varios elementos de información para determinar cómo deben posicionar y dimensionar los componentes en su contenedor. Los componentes de AWT cuentan con un conjunto de métodos que permiten a los gestores de diseño trabajar de una forma razonada para situar los componentes. Todos estos métodos existen para que el componente pueda comunicar el tamaño deseado al componente que se encarga de redimensionarlo (normalmente, un gestor de diseño).

Los métodos utilizados en este caso son parecidos a los “métodos de obtención” (“getters”) de propiedades y representan lo siguiente:

`getPreferredSize()`

El tamaño que elegiría el componente para sí, es decir, el tamaño ideal del componente para un aspecto óptimo. En función de las normas del gestor de diseño, la propiedad `preferredSize` puede tenerse en cuenta, o no, al diseñar el contenedor.

<code>getMinimumSize()</code>	El tamaño mínimo de utilización del componente. El valor <code>minimumSize</code> de un componente puede estar limitado, por ejemplo, por el tamaño de una etiqueta. En la mayoría de los controles AWT, <code>minimumSize</code> equivale a <code>preferredSize</code> . Por lo general, los gestores de diseño tienden a modificar <code>preferredSize</code> antes que <code>minimumSize</code> .
<code>getMaximumSize()</code>	El tamaño máximo de utilización del componente. Este valor existe para evitar que el gestor de diseño malgaste espacio en un componente que no puede utilizarlo, en lugar de utilizarlo con un componente que cuenta solamente con el tamaño indicado en su <code>minimumSize</code> . Por ejemplo, <code>BorderLayout</code> podría limitar el tamaño del componente central a su tamaño máximo y entregar el resto del espacio a los componentes de los bordes o limitar el espacio de la ventana exterior cuando ésta cambia de tamaño.
<code>getAlignmentX()</code>	El alineamiento óptimo del componente en el eje x en relación con otros componentes.
<code>getAlignmentY()</code>	El alineamiento óptimo del componente en el eje y en relación con otros componentes.

Para comprender la forma en que los distintos gestores de diseño utilizan estos datos, consulte la información de los diseños, en [“Diseños suministrados con JBuilder” en la página 8-12](#).

## Tamaño y ubicación de la ventana de la interfaz de usuario durante la ejecución

---

Si la clase de la interfaz de usuario descende de `java.awt.Window` (como por ejemplo `Frame` o `Dialog`), puede controlar su tamaño y su posición en tiempo de ejecución. El tamaño y la posición dependen de una combinación de dos factores: lo que realiza el código cuando se crea la ventana de la interfaz de usuario y lo que realiza el usuario para redimensionarla o cambiar su posición.

Cuando se crea la ventana de la interfaz de usuario y se añaden componentes a ella, cada uno de los componentes afecta al valor de `preferredSize` de toda la ventana, lo que normalmente hace que el valor `preferredSize` de la ventana contenedora aumente a medida que se añaden más componentes. El efecto exacto que esto tiene sobre el valor `preferredSize` depende del gestor de diseño del contenedor externo, así como de los de los diseños de los contenedores anidados. Si desea obtener más información acerca de cómo se calcula el valor `preferredLayoutSize` de los distintos diseños, consulte las secciones de este documento dedicadas a los distintos tipos de diseños.

El *tamaño* de la ventana de la interfaz de usuario, definido por el programa (antes de que el usuario redimensione ninguno de los elementos), depende de cuál de los métodos del contenedor se ejecute en último lugar en el código:

- `pack()`
- `setSize()`

La *posición* de la interfaz de usuario durante la ejecución será 0,0 en la esquina superior izquierda de la pantalla, a no ser que modifique estos valores cambiando el valor de la propiedad `location` del contenedor (por ejemplo, con una llamada a `setLocation()` antes de hacer que sea visible).

## Dimensionamiento automático de una ventana con `pack()`

---

Cuando se realiza una llamada al método `pack()` de una ventana, se solicita que esta calcule su `preferredSize`, en función de los componentes que contiene, cambiando sus dimensiones a este tamaño. Normalmente, esto tiene el efecto de utilizar un tamaño de ventana mínimo que respete el valor `preferredSize` de los componentes que contiene.

Puede realizar una llamada al método `pack()` para cambiar automáticamente una ventana al tamaño mínimo posible, manteniendo un aspecto adecuado en los controles y subcontenedores. Observe que el archivo `Application.java` creado por el Asistente para aplicaciones llama a `pack()` en el marco que crea. Esto hace que el tamaño del marco sea el de su valor `preferredSize` antes de hacer que sea visible.

## Cálculo de `preferredSize` (tamaño recomendado) para los contenedores

---

`preferredSize` se calcula de forma diferente en contenedores con diferentes diseños.

### Diseños portables

Los diseños portables, como `FlowLayout` y `BorderLayout`, calculan su valor `preferredSize` utilizando una combinación de las normas de diseño y el valor `preferredSize` de los componentes que se hayan añadido al contenedor. Si alguno de los componentes es un contenedor (por ejemplo un `Panel`), el valor `preferredSize` de dicho `Panel` se calcula en función de su diseño y sus componentes, repitiendo el cálculo en tantos niveles de contenedores anidados como sea necesario.

Para obtener más información acerca del cálculo de `preferredSize` de los distintos diseños, consulte las descripciones de los distintos diseños.

### XYLayout

En el caso de los contenedores `XYLayout`, el `preferredSize` (tamaño recomendado) del contenedor viene definido por los valores especificados en

las propiedades `width` y `height` del diseño `XYLayout`. Por ejemplo, si cuenta con las líneas de código siguientes en la inicialización del contenedor:

```
xYLayoutN.setWidth(400);
xYLayoutN.setHeight(300);
```

Y si utiliza `xYLayoutN` como gestor de diseño del contenedor, su valor `preferredSize` será de 400 por 300 píxeles.

Si uno de los paneles anidados en la interfaz de usuario tiene un diseño `XYLayout`, su tamaño `preferredSize` viene determinado por las llamadas a `setWidth()` y `setHeight()` del diseño, y ese es el valor que emplea el panel para calcular el `preferredSize` del siguiente contenedor externo.

Por ejemplo, en la aplicación por defecto Asistente para aplicaciones, el panel anidado que ocupa el centro del `BorderLayout` del marco inicialmente está en `XYLayout` configurado a un tamaño de 400 x 300. Esto tiene un efecto importante en el tamaño total del marco cuando se empaqueta, puesto que el panel anidado informa de que su tamaño `preferredSize` es de 400x300. El marco total será ese más los tamaños necesarios para satisfacer los otros componentes que lo rodean en el `BorderLayout` del marco.

## Definición explícita del tamaño del contenedor de la interfaz de usuario utilizando `setSize()`

---

Si realiza una llamada al método `setSize()` del contenedor (en lugar de llamar a `pack()` o tras llamar a `pack()`), el tamaño del contenedor cambiará a un tamaño específico en píxeles. Esto tiene básicamente el mismo efecto que si el usuario cambiara manualmente el tamaño del contenedor: redefine el efecto de `pack()` y `preferredSize` sobre el contenedor y lo cambia a un tamaño arbitrario.

### Importante

A pesar de que es posible cambiar el tamaño del contenedor a un ancho y un alto concretos, esto puede hacer que la interfaz sea menos portable, porque las distintas pantallas tienen diferentes resoluciones. Si ha fijado el tamaño explícitamente mediante `setSize()`, ha de llamar a `validate()` para que los objetos hijos se dispongan correctamente. (Observe que `pack()` llama a `validate()`.)

## Preparación del tamaño de la interfaz de usuario para que pueda portarse a varias plataformas

---

Normalmente, si desea que la interfaz de usuario sea portable, puede utilizar `pack()` sin utilizar explícitamente `setSize()` o, por el contrario, calcular cuidadosamente el tamaño en píxeles de las distintas pantallas, además de realizar un cálculo razonable del tamaño que debe utilizar.

Por ejemplo, puede decidir que, en lugar de llamar a `pack()`, prefiere que la interfaz de usuario aparezca siempre con un 75% del ancho y el alto de la pantalla. Para ello, puede añadir las líneas de código siguientes a la clase `Application`, en lugar de la llamada a `pack()`:

```
Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();  
frame.setSize(screenSize.width * 3 / 4, screenSize.height * 3 / 4);
```

**Nota** Además, para garantizar la portabilidad, cambie todos los contenedores `XYLayout` a un diseño portable después de la creación de prototipos.

## Colocación de una ventana en la pantalla

---

Si no sitúa explícitamente la interfaz de usuario en la pantalla, aparece en la esquina superior izquierda. Con frecuencia resulta más apropiado centrar la interfaz de usuario en la pantalla. Para ello, obtenga la altura y la anchura de la pantalla, résteles la anchura y la altura de la interfaz de usuario, divida la diferencia entre dos (para crear márgenes iguales en los lados opuestos de la interfaz de usuario) y utilice estas dos cifras de diferencia como posición de la esquina superior izquierda de la interfaz.

Un ejemplo de esto es el código generado con la opción `Centrar marco` en la pantalla del Asistente para aplicaciones. Esta opción genera código adicional en la clase `Aplicacion`, que, después de crear el marco, lo centra en la pantalla. Observe el código que genera esta opción, para ver un buen ejemplo de cómo centrar la interfaz de usuario.

```
//Centrar la ventana  
Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();  
Dimension frameSize = frame.getSize();  
if (frameSize.height > screenSize.height) {  
    frameSize.height = screenSize.height;  
}  
if (frameSize.width > screenSize.width) {  
    frameSize.width = screenSize.width;  
}  
frame.setLocation((screenSize.width - frameSize.width) / 2,  
    (screenSize.height - frameSize.height) / 2);
```

## Inclusión en el código de las llamadas a métodos de tamaño y posicionamiento

---

Las llamadas a `pack()`, `validate()`, `setSize()` y `setLocation()` pueden efectuarse desde la clase contenedora de la interfaz de usuario, por ejemplo `this.pack()`. También pueden efectuarse desde la clase que crea el contenedor, como por ejemplo `frame.pack()`, a continuación de la llamada al constructor, antes de `setVisible()`. Esta segunda opción es la que utiliza el código generado por el Asistente para aplicaciones: las llamadas a `pack()` o `validate()` y `setLocation()` se colocan en la clase `Aplicacion`, una vez se ha construido el marco y ha finalizado `jbInit()`.

¿Cómo decidir dónde ha de situar las llamadas para determinar los tamaños y las posiciones de los componentes de la interfaz de usuario?

- Si desea construir la interfaz desde varios puntos de la aplicación y prefiere que aparezca siempre con el mismo tamaño y en la misma posición,



considere situar las llamadas en el constructor de la clase contenedora de la interfaz de usuario (tras la llamada a `jbInit()`).

- Si la aplicación crea una instancia de la interfaz de usuario en un solo punto, como ocurre en la aplicación generada por el Asistente para aplicaciones, resulta perfectamente razonable situar el código de dimensionamiento y posicionamiento en el lugar donde se crea la interfaz, en este caso la clase `Application`.

## Adición de gestores de diseño personalizados

---

JBuilder acepta la integración otros gestores de diseño con su propio diseñador. Si desea que en la lista de la propiedad `layout` del Inspector aparezca un gestor de diseño personalizado, debe registrar para él un asistente para diseño. Puede ver una muestra del uso de este asistente en `samples/OpenToolsAPI/layoutassistant`.

El registro consiste en una llamada de una línea al método estático `initOpentool()`. Aquí se indica qué asistente debe gestionar el diseño.

Si el gestor de diseño personalizado utiliza una clase constraint, puede conseguirse un mayor nivel de integración mediante una clase que implemente `java.beans.PropertyEditor` para editar la restricción. Este editor de propiedades también debe encontrarse en la vía de acceso a clases de JBuilder.

A continuación, se necesitaría ampliar `BasicLayoutAssistant` y redefinir dos métodos:

```
public java.beans.PropertyEditor  getPropertyEditor() {
    //devolver una instancia del editor de la propiedad constraints
    return new com.mycompany.MyLayoutConstrainteditor();
}

public String getConstraintsType () {
    // devolver el nombre completo de la restricción clasificada como una
    cadena,
    // por ejemplo
    return "com.mycompany.myLayout";
}
```

`BasicLayoutAssistant` es una implementación estructural de la interfaz `com.borland.jbuilder.designer.ui.LayoutAssistant`.

Para que el diseñador pueda manipular el diseño, los gestores de diseño deben estar asociados a una clase que implemente esta interfaz. Es posible utilizar los diseños que no tengan un asistente para diseño asociado, pero no se les puede asignar `BasicLayoutAssistant`, por lo que no se pueden desplazar los componentes ni convertir los diseños de contenedor en diseños de este tipo.

### Consulte

- Documentación del concepto de los asistentes de diseño personalizado de OpenTools en *Developing OpenTools*
- El ejemplo de LayoutAssistant, `<jbuilder>/samples/OpenToolApi/LayoutAssistant/LayoutAssistant.jpx`

## Diseños suministrados con JBuilder

---

JBuilder de Borland cuenta con los gestores de diseño estándar siguientes de AWT de Java y Swing:

- BorderLayout
- FlowLayout
- GridLayout
- CardLayout
- GridBagLayout
- null
- XYLayout, que conserva el tamaño y la posición iniciales (coordenadas x,y) de los componentes colocados en un contenedor.
- PanelLayout, que controla el porcentaje del contenedor que ocupa cada uno de sus componentes.
- VerticalFlowLayout, que es muy similar a FlowLayout, con la excepción de que distribuye los componentes verticalmente, no horizontalmente.
- BoxLayout2, una clase englobadora de beans para BoxLayout de Swing que se puede seleccionar como un diseño en el Inspector.
- OverlayLayout2, una clase englobadora de beans para OverlayLayout de Swing que se puede seleccionar como un diseño en el Inspector.

Puede crear diseños personalizados propios o probar otros diseños, como los que contienen las clases `sun.awt` o los de los gestores de diseño de terceros. La mayoría son de dominio público.

**Nota** Si desea utilizar un diseño personalizado en el diseñador, es posible que necesite indicar un archivo de clase auxiliar de Java con el fin de que el diseñador pueda utilizar el diseño.

La mayoría de los diseños de interfaz de usuario utilizarán una combinación de diseños, por medio de la anidación de paneles con diseños diferentes, unos dentro de otros. Si desea ver cómo se hace esto, consulte las referencias siguientes.

### Consulte

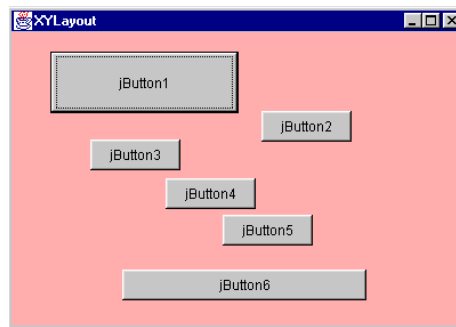
- [“Paneles y diseños anidados” en la página 8-53](#)
- [Capítulo 11, “Tutorial: Creación de una interfaz de usuario con diseños anidados”](#)

# XYLayout

`XYLayout` es un gestor de diseño personalizado de JBuilder. `XYLayout` sitúa los componentes en un contenedor en coordenadas x e y específicas, relativas a la esquina superior izquierda del contenedor. Independientemente del tipo de pantalla, el contenedor siempre mantendrá las posiciones x e y relativas de los componentes. Sin embargo, cuando redimensiona un contenedor con un `XYLayout`, los componentes **no** vuelven a posicionarse ni redimensionarse.

**Importante** Si cambia un diseño a `XYLayout` en el Inspector del diseñador, JBuilder añade esta sentencia de importación al código fuente: `com.borland.jbcl.layout.*`. Más adelante, cuando termine su interfaz de usuario y cambie `XYLayout` a un diseño más portable antes de la distribución, la sentencia de importación *no* se elimina. Es necesario que la elimine manualmente si no desea importar esa clase.

**Figura 8.1** Ejemplo de XYLayout



`XYLayout` resulta muy útil para crear prototipos de diseño. Cuando se diseñan interfaces de usuario más complicadas con varios paneles anidados, `XYLayout` puede utilizarse para el diseño inicial de paneles y componentes, tras lo cual puede seleccionar uno de los diseños estándar para el diseño final.

**Nota** Para asegurarse de que el diseño tenga un aspecto agradable en otras pantallas, no deje contenedores en `XYLayout` en el diseño final.

Puede utilizar las herramientas de diseño visual para especificar el tamaño del contenedor y las coordenadas x e y de sus componentes.

- Para establecer el tamaño del contenedor `XYLayout`, seleccione el objeto `XYLayout` en el árbol de componentes e introduzca en píxeles los valores de las propiedades `height` y `width` en el Inspector. De esta forma, definirá el tamaño del contenedor `XYLayout`.
- Para cambiar los valores x e y de un componente que se encuentra dentro de un contenedor `XYLayout`, realice una de las operaciones siguientes:
  - En la superficie de diseño, arrastre el componente hacia un nuevo tamaño. JBuilder actualiza automáticamente los valores de restricción del Inspector.

- Seleccione el componente en el árbol de componentes y haga clic en el campo de edición de la propiedad `constraints` para introducir las coordenadas de ese componente.

## Alineación de componentes en XYLayout

---

Es posible alinear un grupo de componentes seleccionados de un contenedor con diseño `XYLayout`. La alineación no funciona en otros diseños.

Con las operaciones de alineación, puede hacer que un conjunto de componentes tenga la misma anchura, altura, alineación a la izquierda, etcétera, de forma que aparezcan organizados con claridad.

Para alinear componentes:

- 1 Seleccione los componentes que desee alinear. El orden de la selección afecta a la alineación, como se describe en la tabla que aparece más abajo.
- 2 Haga clic con el botón derecho sobre el componente seleccionado en la superficie de diseño y seleccione la operación de alineación que desea realizar.

### Consulte

- [Capítulo 8, “Gestores de diseño”](#)

## Opciones de alineación para XYLayout

---

La tabla siguiente explica las opciones de alineación disponibles en el menú contextual:

Seleccione	Para
Trasladar al primero	Mover el componente seleccionado a la parte superior del orden Z.
Trasladar al último	Mover el componente seleccionado a la parte inferior del orden Z.
Alinear a la izquierda	Alinear los bordes izquierdos de los componentes con el borde izquierdo del primer componente seleccionado.
Centrar en horizontal	Alinear el centro de los componentes con el centro del primer componente seleccionado.
Alinear a la derecha	Alinear los bordes derechos de los componentes con el borde derecho del primer componente seleccionado.
Alinear arriba	Alinear los bordes superiores de los componentes con el borde superior del primer componente seleccionado.
Centrar en vertical	Alinear verticalmente el centro de los componentes con el centro del primer componente seleccionado.

Seleccione	Para
Alinear abajo	Alinear los bordes inferiores de los componentes con el borde inferior del primer componente seleccionado.
Distribución horizontal	Distribuir equidistantemente en horizontal los componentes seleccionados.
Distribución vertical	Distribuir equidistantemente en vertical todos los componentes seleccionados.
Tamaño horizontal igual	Aplicar a todos los componentes la anchura del primer componente seleccionado.
Tamaño vertical igual	Aplicar a todos los componentes la altura del primer componente seleccionado.

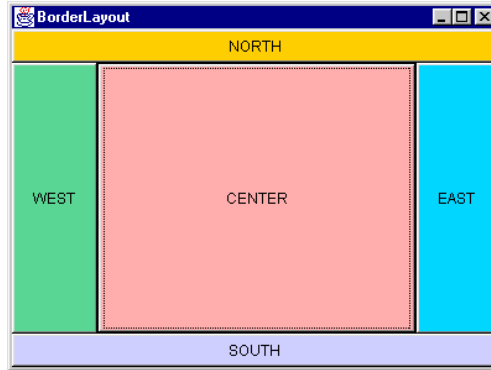
## Null

Diseño `Null` significa que no se ha asignado ningún gestor de diseño al contenedor. El diseño `Null` (de `Swing`) es muy similar al `XYLayout`, en el sentido de que permite situar los componentes en los contenedores en unas determinadas coordenadas `x` e `y` relativas a la esquina superior izquierda del contenedor. Las coordenadas `x` e `y` de cada componente se especifican en su propiedad `constraints`. Más adelante, puede cambiar a un diseño portable más apropiado a sus necesidades. Asegúrese de indicar un gestor de diseño para el contenedor antes de la distribución, porque, de otro modo, los componentes no se ajustarán al redimensionar el supercontenedor ni a las diferencias en usuarios y sistemas.

## BorderLayout

`BorderLayout` organiza los componentes de los contenedores en áreas denominadas `North` (Norte), `South` (Sur), `East` (Este), `West` (Oeste) y `Center` (Centro). Estas son las restricciones de colocación de `BorderLayout`.

- A los componentes que se encuentran en `North` y `South` se les aplica su altura recomendada y ocupan todo el ancho del contenedor.
- A los componentes que se encuentran en `East` y `West` se les aplica su anchura recomendada y ocupan todo el espacio vertical libre existente entre las áreas `North` y `South`.
- Los componentes que se encuentran en `Center` se amplían hasta ocupar todo el espacio restante.

**Figura 8.2** Restricciones de colocación de BorderLayout

Por defecto, JBuilder coloca los nuevos componentes en el centro del contenedor. A medida que se añaden componentes, los antiguos se desplazan hacia los lados: primero North (norte), después South (sur), después West (oeste) y por último East (este). Para cambiar la ubicación de un componente, selecciónelo, elija la propiedad Constraints en el Inspector y seleccione un lugar distinto en la lista desplegable.

En los contenedores `BorderLayout` solo hay cinco ubicaciones disponibles. Si necesita añadir más de cinco componentes a un contenedor, anídelos o utilice un gestor de diseño distinto. Para cambiar el gestor de diseño, seleccione el contenedor, elija la propiedad Layout en el Inspector y seleccione un gestor distinto en la lista desplegable.

En el AWT de Java, todas las ventanas (incluidos marcos y cuadros de diálogos) utilizan `BorderLayout` por defecto.

`BorderLayout` es adecuado para forzar a los componentes hacia los bordes del contenedor y para ocupar todo el centro del contenedor con un componente. Este es también el diseño que se debe utilizar si se desea que un componente llene por completo el contenedor.

`BorderLayout` es un gestor de diseño muy útil para los contenedores más grandes de la interfaz de usuario. Mediante la anidación de un panel dentro de cada una de las áreas de `BorderLayout` y ocupando cada uno de esos paneles con otros paneles de distintos diseños, puede obtener diseños de interfaz de usuario complejos.

## Definición de restricciones

---

Por ejemplo, para poner una barra de herramientas a lo largo de la parte superior de un contenedor de `BorderLayout`, podría crear un panel de botones `FlowLayout` y colocarlo en el área Norte del contenedor. Esto se hace mediante la selección del panel y cambiando a North su propiedad `constraints`, en el Inspector.

Para modificar la propiedad `constraints`:

- 1 Seleccione, en la superficie de diseño o en el árbol de componentes, el componente que desea colocar.
- 2 Seleccione la propiedad `constraints` en el Inspector.
- 3 Haga clic en la flecha Abajo de la lista desplegable de la propiedad `constraints` y seleccione el área que desee que ocupe el componente.
- 4 Pulse *Intro* o haga clic en cualquier otro lugar del Inspector para realizar los cambios. Este cambio se refleja inmediatamente en el código y en el diseño.

Si utiliza las herramientas de diseño visual de JBuilder para cambiar el diseño de un contenedor a `BorderLayout`, los componentes cercanos a los bordes se desplazan automáticamente para ocupar el borde más cercano. Un componente cercano al centro puede tener el valor `Center`. Si un componente se desplaza a una ubicación no deseada, puede corregir la propiedad `constraints` en el Inspector o arrastrar el componente en la superficie de diseño.

Mientras se arrastra un componente por un contenedor `BorderLayout`, la superficie de diseño presenta un rectángulo que muestra el área del contenedor en la que va a encajar el componente si se suelta en ese momento.

Cada una de las cinco áreas solo puede contener un componente (o un panel de componentes), de forma que debe ser cuidadoso cuando cambie un contenedor a `BorderLayout`.

- Asegúrese de que el contenedor no tenga más de cinco componentes.
- Utilice primero `XYLayout` para mover los componentes a sus posiciones aproximadas, con un solo componente cerca de cada borde.
- Si existen varios componentes en un área, agrúpelos dentro de un panel antes de la conversión.

**Nota** `BorderLayout` no tiene en cuenta el orden en el que se añaden los componentes al contenedor.

Por defecto, `BorderLayout` no añade espacios vacíos entre los componentes que administra. Sin embargo, puede utilizar el Inspector para especificar un espacio vacío horizontal o vertical, en píxeles, para un diseño asociado con un contenedor.

Para modificar el espacio que rodea a los componentes de `BorderLayout`, seleccione el objeto `BorderLayout` en el árbol de componentes, que aparece inmediatamente debajo del contenedor que controla. A continuación, utilice el Inspector para modificar el número de píxeles de las propiedades `hgap` y `vgap`.

## Consulte

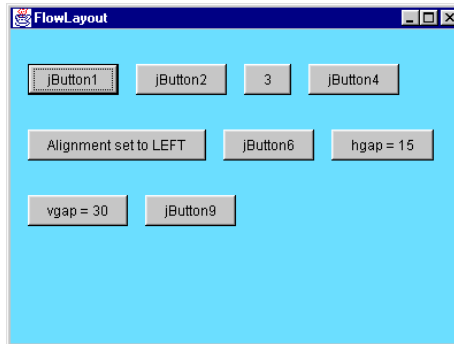
- [“Paneles y diseños anidados” en la página 8-53](#)

# FlowLayout

---

`FlowLayout` ordena los componentes en filas de izquierda a derecha y después de arriba a abajo, utilizando el `preferredSize` natural de cada componente. `FlowLayout` alinea tantos componentes en una fila como le es posible y se desplaza a continuación a la línea siguiente. Normalmente, `FlowLayout` se utiliza para ordenar botones en un panel. En el AWT de Java, todos los paneles (incluidas las applets) utilizan `FlowLayout` por defecto.

**Figura 8.3** Ejemplo de `FlowLayout`



**Nota** Si desea disponer de un panel en que los componentes se ordenen verticalmente en lugar de hacerlo horizontalmente, consulte [“VerticalFlowLayout” en la página 8-19](#).

Puede elegir la forma en que se ordenan los componentes en las filas de un contenedor `FlowLayout` mediante la especificación de una justificación de alineación de izquierda, derecha o centro. También puede especificar el espacio (horizontal o vertical) entre los componentes y las filas. En el diseñador puede emplear el Inspector para cambiar las propiedades `alignment` y `gap`.

## Alineación

---

LEFT - agrupa los componentes en el borde izquierdo del contenedor.  
CENTER - centra los componentes en el contenedor.  
RIGHT - agrupa los componentes en el borde derecho del contenedor.

La alineación por defecto en `FlowLayout` es CENTER.

Para cambiar la alineación, seleccione el objeto `FlowLayout` mostrado bajo el contenedor que controla en el árbol de componentes y asigne un valor en el Inspector para la propiedad `alignment`.



## Espacio

---

El espacio por defecto entre componentes de un `FlowLayout` es de 5 píxeles.

Para cambiar el espacio horizontal o vertical, seleccione el objeto `FlowLayout` en el árbol de componentes. A continuación, modifique el valor de píxeles de la propiedad `hgap` (espacio horizontal) o `vgap` (espacio vertical) en el Inspector.

## Orden de los componentes

---

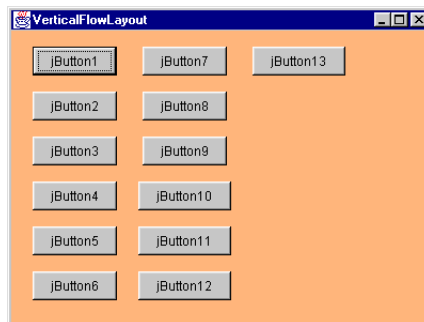
Para cambiar el orden de los componentes en un contenedor `FlowLayout`, arrastre el componente a la nueva ubicación o haga clic con el botón derecho en un componente y seleccione `Trasladar al primero` o `Trasladar al último`.

## VerticalFlowLayout

---

`VerticalFlowLayout` distribuye los componentes en columnas, de arriba abajo y de izquierda a derecha, en función del tamaño recomendado natural de cada contenedor. `VerticalFlowLayout` alinea tantos componentes en una columna como le es posible y se desplaza a la línea siguiente. Normalmente, `VerticalFlowLayout` se utiliza para ordenar botones en un panel.

**Figura 8.4** Ejemplo de `VerticalFlowLayout`



Puede elegir la forma en que se ordenan los componentes en las columnas de un contenedor `VerticalFlowLayout` mediante la especificación de una justificación de alineación de arriba, medio o abajo. También puede especificar la cantidad de espacio (horizontal o vertical) entre los componentes y las columnas. Asimismo dispone de propiedades para indicar que los componentes han de ocupar toda la anchura de la columna o que el último componente ha de ocupar la altura restante del contenedor. En el diseñador puede emplearse el Inspector para modificar estas propiedades.

## Alineación

---

TOP - agrupa los componentes en la parte superior del contenedor.

MIDDLE - centra verticalmente los componentes en el contenedor.

BOTTOM - agrupa los componentes de forma que el último se encuentre en la parte inferior del contenedor.

La alineación por defecto en un `VerticalFlowLayout` es TOP.

Con el fin de cambiar la alineación, seleccione el objeto `VerticalFlowLayout` que se muestra bajo el contenedor que controla en el árbol de componentes y, a continuación, en el Inspector, asigne un valor a la propiedad `alignment`.

## Espacio

---

El espacio por defecto entre componentes de un `VerticalFlowLayout` es de 5 píxeles.

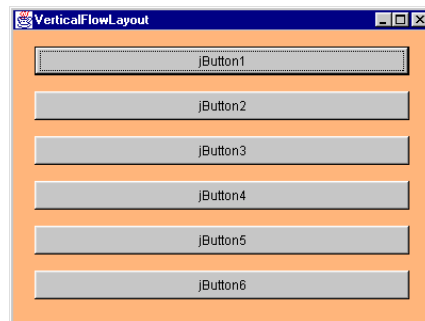
Para cambiar el espacio horizontal o vertical, seleccione el objeto `VerticalFlowLayout` en el árbol de componentes. A continuación, modifique el valor de píxeles de la propiedad `hgap` (espacio horizontal) o `vgap` (espacio vertical) en el Inspector.

## Relleno horizontal

---

`horizontalFill` establece un indicador de bordes para que los componentes se expandan horizontalmente hasta ocupar toda la anchura del contenedor.

**Figura 8.5** Ejemplo de `horizontalFill`



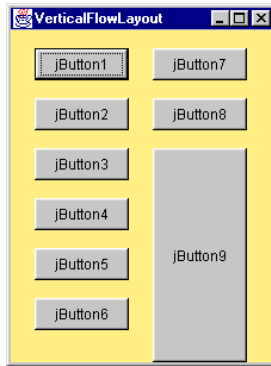
**Advertencia** Pueden producirse problemas si el panel principal dispone de menos espacio del necesario. También impide la presentación en varias columnas.

El valor por defecto de `horizontalFill` es `True`.

## Relleno vertical

---

`verticalFill` establece un indicador para que el último componente se expanda hasta ocupar la altura libre restante del contenedor.

**Figura 8.6** Ejemplo de verticalFill

El valor por defecto de `verticalFill` es `False`.

## Orden de los componentes

---

Para cambiar el orden de los componentes en un contenedor `VerticalFlowLayout`, arrastre el componente a la nueva ubicación o haga clic con el botón derecho en un componente y seleccione `Trasladar al primero` o `Trasladar al último`.

## BoxLayout2

---

`BoxLayout2` es el diseño `BoxLayout` de Swing integrado como bean para que pueda seleccionarse en el Inspector. Combina las funciones de `FlowLayout` y de `VerticalFlowLayout` en un solo gestor de diseño.

Cuando se crea un contenedor `BoxLayout2`, se especifica si su eje principal es el eje x (disposición de izquierda a derecha) o el eje y (disposición de arriba a abajo). Los componentes se disponen de izquierda a derecha (o de arriba a abajo) por el mismo orden en el que se han añadido al contenedor.

### Consulte

- `BoxLayout` en la documentación de Swing

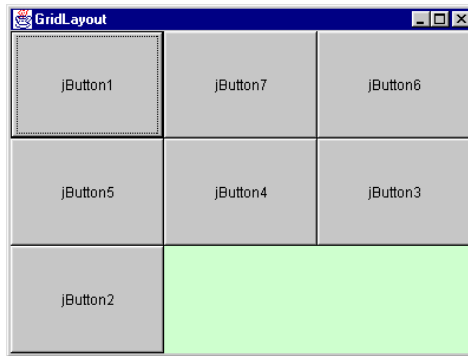
## GridLayout

---

`GridLayout` sitúa los componentes en una rejilla de celdas organizadas en filas y columnas. `GridLayout` expande los componentes de manera que llenen todo el espacio disponible dentro de su celda. Todas las celdas tienen exactamente el mismo tamaño y la rejilla es uniforme. Cuando se redimensiona un contenedor `GridLayout`, `GridLayout` cambia el tamaño de la

celda de forma que las celdas sean lo más grandes posible, teniendo en cuenta el espacio disponible para el contenedor.

**Figura 8.7** Ejemplo de GridLayout



Utilice `GridLayout` si desea diseñar un contenedor en el que los componentes han de tener el mismo tamaño, por ejemplo un teclado numérico o una barra de herramientas.

## Columnas y filas

---

Puede especificar el número de columnas y filas de una rejilla. La regla básica de `GridLayout` es que se puede asignar cero a las columnas o a las filas, pero no a ambas. Una de estas (filas o columnas) debe tener un valor distinto de cero, de forma que el administrador de `GridLayout` pueda calcular el otro valor.

Por ejemplo, si especifica cuatro columnas y cero filas para una rejilla que tiene 15 componentes, `GridLayout` crea cuatro columnas de 4 filas y la última fila contiene tres componentes. Por otro lado, si especifica tres filas y cero columnas, `GridLayout` crea tres filas con cinco columnas completas.

## Espacio

---

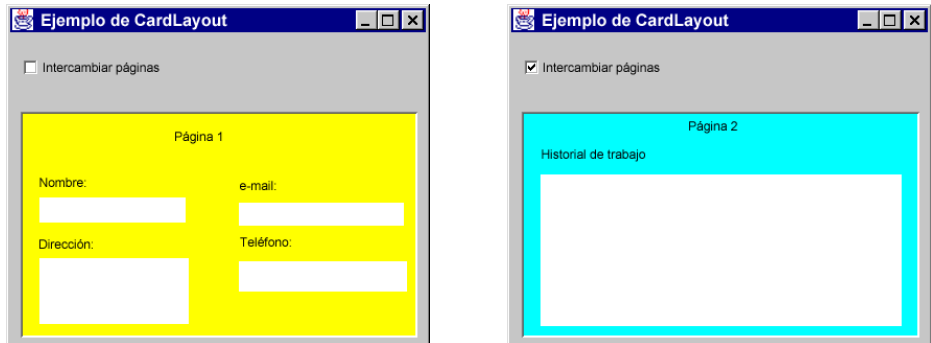
Además del número de filas y columnas, puede especificar el número de píxeles que debe separar las celdas, por medio del espacio horizontal (`hgap`) y el espacio vertical (`vgap`). El valor por defecto del espacio horizontal y el vertical es cero.

Para cambiar los valores de las propiedades de un contenedor `GridLayout` mediante las herramientas de diseño visuales, seleccione en el árbol de componentes el objeto `GridLayout` situado debajo del contenedor que controla, y, a continuación, en el Inspector, modifique los valores de `rows`, `cols`, `hgap` y `vgap`.

# CardLayout

`CardLayout` coloca los componentes (normalmente paneles) uno encima de otro en una pila parecida a una baraja de cartas. Sólo se ven de uno en uno, y se pueden inspeccionar utilizando otro control para seleccionar qué panel se coloca en la parte superior.

**Figura 8.8** Ejemplo de `CardLayout`



`CardLayout` es un diseño aconsejable cuando se tiene un área que contiene componentes distintos en momentos diferentes. Proporciona una manera de administrar dos o más paneles que necesitan compartir el mismo espacio de visualización.

`CardLayout` se asocia normalmente a un componente de control, como una casilla de selección o una lista. El estado del componente de control determina qué componente presenta `CardLayout`. El usuario elige el estado mediante la selección de un elemento de la interfaz.

## Creación de un contenedor `CardLayout`

El siguiente ejemplo de un contenedor `CardLayout` controlado por una casilla de selección muestra cómo crear el contenedor en el diseñador, y crear código con el fin de que la casilla de selección cambie los paneles. En este ejemplo se utilizan los componentes `JPanel` y `JCheckBox` de la ficha Swing de la paleta de componentes.

- 1 Cree un proyecto con el Asistente para aplicaciones.
- 2 Seleccione el archivo `marco1.java` en el panel de proyectos. A continuación, haga clic en la pestaña Diseño de la parte inferior del Visualizador de aplicaciones, para abrir el diseñador de interfaces de usuario.
- 3 Añada un panel (`jPanel1`) al componente `contentPane` en el diseñador de interfaces.
- 4 Asigne a su propiedad `layout` el valor `XYLayout`.
- 5 Añada un panel (`jPanel2`) a la mitad inferior de `JPanel1`.
- 6 Defina su diseño como `CardLayout`.

- 7 Coloque otro panel (`jPanel3`) sobre este `CardLayout`, haciendo clic en `jPanel2` en el árbol de componentes. Este nuevo panel ocupa completamente el panel `CardLayout`.

**Nota**

El primer componente que se añade a un panel `CardLayout` ocupa siempre el panel. Para añadirle más paneles, coloque el componente haciendo clic en el panel `CardLayout`, en el árbol de componentes, y no en la superficie de diseño.

- 8 Cambie su propiedad de color de fondo, `background`, o añádale componentes de interfaz de usuario para que se pueda distinguir.
- 9 Coloque otro panel (`jPanel4`) sobre `jPanel2` en el árbol de componentes. Ahora hay dos paneles por debajo de `jPanel2` en el árbol de componentes.
- 10 Cambie el color de fondo de `jPanel4` o añádale componentes.

## Creación de los controles

---

Ahora que dispone de una pila de dos paneles en un contenedor `CardLayout`, debe añadir un componente de control a la interfaz de usuario, como `JList` o `JCheckBox`, de forma que el usuario puede cambiar el foco entre cada uno de los paneles.

- 1 Añada a `JPanel1` un componente `JCheckBox` (`jCheckBox1`), que se utilizará para cambiar entre los dos paneles del contenedor `CardLayout`.
- 2 Abra en el Inspector la pestaña Sucesos de `jCheckBox1` y haga doble clic en el suceso `actionPerformed` para crearlo en el código fuente.
- 3 Añada el siguiente código al método `jCheckBox1_actionPerformed(ActionEvent)`:

```
if (jCheckBox1.isSelected())
    ((CardLayout)jPanel2.getLayout()).show(jPanel2, "jPanel4");
else
    ((CardLayout)jPanel2.getLayout()).show(jPanel2, "jPanel3");
```

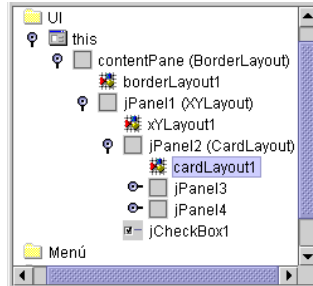
- 4 Compile y ejecute el programa. Active y desactive la casilla de selección para cambiar de panel en el contenedor `CardLayout`.

## Definición del espacio

---

Mediante el Inspector, puede especificar el espacio horizontal y vertical que rodea a una pila de componentes en un `CardLayout`.

- 1 Seleccione en el árbol de componentes el objeto `CardLayout` que aparece inmediatamente debajo del contenedor que controla.



- 2 Haga clic sobre la propiedad `hgap` (separación horizontal) o en `vgap` (separación vertical) del Inspector.
- 3 Introduzca un valor para determinar la separación en píxeles.
- 4 Pulse *Intro* o haga clic en cualquier otro lugar del Inspector para registrar los cambios.

## OverlayLayout2

---

`OverlayLayout2` es el diseño `OverlayLayout` de Swing integrado como bean para que pueda seleccionarse en el Inspector. Es muy similar a `CardLayout` en el sentido de que superpone los componentes.

A diferencia de `CardLayout` que solo permite ver los componentes de uno en uno, permite ver simultáneamente varios componentes; basta declararlos transparentes en el contenedor. Por ejemplo, pueden superponerse varias imágenes transparentes en el contenedor para obtener un gráfico compuesto.

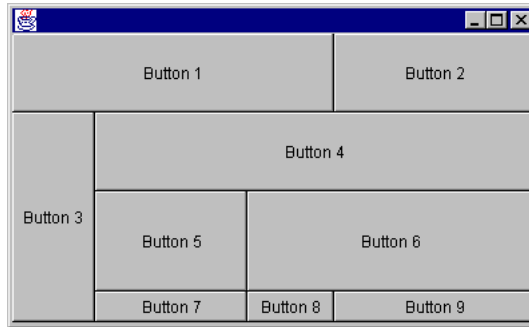
### Consulte

- `OverlayLayout` en la documentación de Swing

## GridBagLayout

---

`GridBagLayout` es un diseño extremadamente flexible y potente que ofrece un mayor nivel de control que `GridLayout` para situar componentes en un rejilla. `GridBagLayout` sitúa los componentes horizontal y verticalmente en una rejilla rectangular dinámica. No es necesario que todos los componentes tengan el mismo tamaño y también pueden ocupar varias celdas.

**Figura 8.9** Ejemplo de GridBagLayout

`GridBagLayout` determina la ubicación de sus componentes a partir de las restricciones y el tamaño mínimo de cada uno de ellos, y el tamaño recomendado del contenedor.

`GridBagLayout` puede adaptar una rejilla compleja o componentes contenidos en paneles menores, anidados en el contenedor `GridBagLayout`. Estos paneles anidados pueden emplear otros diseños y contener a su vez otros paneles de componentes. Este método de anidación presenta dos ventajas:

- Ofrece un control más preciso sobre la posición y el tamaño de cada componente, ya que se emplean diseños más adecuados para zonas concretas, como las barras de botones.
- Requiere menos celdas, lo que simplifica el diseño `GridBagLayout` y facilita su control.

### Consulte

- [“Utilización de GridBagLayout” en la página 12-3](#) para obtener una introducción de `GridBagLayout` y `GridBagConstraints`.
- [Capítulo 12, “Tutorial: Creación de diseños GridBagLayout en JBuilder”](#) para practicar el uso de `GridBagLayout`.
- [“Sugerencias y técnicas” en la página 12-42](#) para obtener información adicional sobre el uso de `GridBagLayout` y sobre las restricciones.

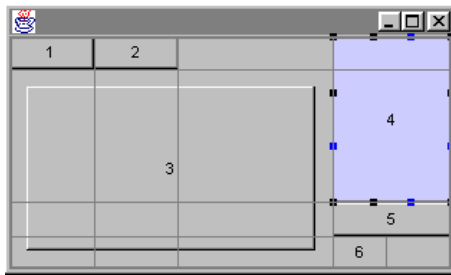
## Área de visualización

La definición de una celda de rejilla se hace del mismo modo en `GridBagLayout` que con `GridLayout`: una celda tiene una columna de ancho y una fila de alto. No obstante, a diferencia de `GridLayout`, en donde todas las celdas tienen el mismo tamaño, las celdas de `GridBagLayout` pueden combinar diferentes alturas y anchuras, así como un componente puede ocupar más de una celda en sentido horizontal y vertical.

El área ocupada por un componente se conoce como su *área de visualización* y se especifica con los `GridBagConstraints` del componente `gridwidth` y `gridheight` (número de celdas horizontales y verticales en el área de visualización).

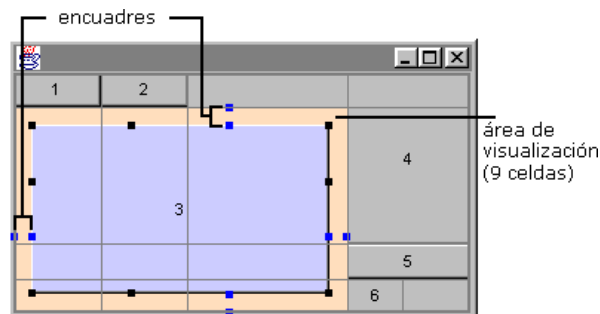


Por ejemplo, en el siguiente contenedor `GridBagLayout`, el componente “4” ocupa una celda (o columna) horizontalmente y dos celdas (filas) verticalmente. Por lo tanto, su área de visualización consta de dos celdas.



Un componente puede ocupar por completo su área de visualización (como ocurre con el componente “4” en el ejemplo anterior) o puede ser más pequeño que ella.

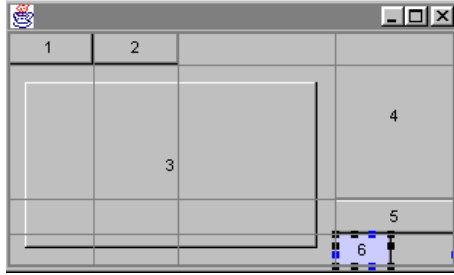
Por ejemplo, en el siguiente contenedor `GridBagLayout`, el área de visualización del componente “3” consta de nueve celdas, tres en sentido horizontal y tres en vertical. Sin embargo, el componente es más pequeño que el área de visualización porque tiene *encuadres* que crean una barrera entre los bordes del área de visualización y el componente.



Aun cuando este componente tiene restricciones de *expansión* horizontales y verticales, dado que también tiene *encuadres* en los cuatro lados del componente (por los tiradores azules dobles en cada lado del área de visualización), estas tienen prioridad sobre las restricciones de *expansión*. El resultado es que el componente solo ocupa el área de visualización hasta los *encuadres*.

Si intenta lograr que el componente sea más grande que su área de visualización actual, `GridBagLayout` aumenta el tamaño de las celdas en el área de visualización para adaptarse al nuevo tamaño del componente y deja espacio para los *encuadres*.

Un componente también puede ser más pequeño que su área de visualización cuando no hay **encuadres**, como el componente “6” del siguiente ejemplo.



Aun cuando el área de visualización tiene sólo una celda, no hay restricciones que extiendan el componente más allá de su tamaño mínimo. En este caso, la anchura del área de visualización queda determinada por los componentes más grandes que están por encima en la misma columna. El componente “6” se muestra en su tamaño mínimo y dado que es más pequeño que su área de visualización, está anclado en el borde izquierdo del área de visualización con una restricción de **ancla**.

Como se puede ver, `GridBagConstraints` desempeña un papel fundamental en `GridBagLayout`. Estas restricciones se tratan con detalle en el siguiente apartado dedicado a **“Acerca de GridBagConstraints”**.

### Consulte

- “How to use GridBagLayout” en el tutorial de Java de Sun en <http://java.sun.com/docs/books/tutorial/uiswing/layout/gridbag.html>
- “GridBagLayout” en la documentación del JDK

## Acerca de GridBagConstraints

`GridBagLayout` se sirve de un objeto `GridBagConstraints` para determinar la información relativa al diseño de los componentes incluidos en el contenedor `GridBagLayout`. Dado que no existe una relación unívoca entre los componentes del contenedor y el objeto `GridBagConstraints`, es necesario adaptar este a cada uno de aquéllos.

Los componentes `GridBagLayout` disponen de las siguientes restricciones:

- `ancla`
- `gridx`, `gridy`
- `ipadx`, `ipady`
- `gridwidth`, `gridheight`
- `expansión`
- `encuadres`
- `weightx`, `weighty`

`GridBagConstraints` permite controlar:

- La posición absoluta o relativa de los componentes.
- El tamaño absoluto o relativo de los componentes.
- El número de celdas que ocupa cada componente.
- Cómo se ocupa el área de visualización no utilizada en la celda
- La cantidad de peso que se asigna a los componentes con el objeto de controlar cómo utilizan la parte adicional del espacio disponible. Esto determina cómo se comporta el componente cuando se cambia el tamaño del contenedor.

Si desea una explicación detallada de las restricciones, con recomendaciones para utilizarlas y configurarlas en el diseñador, consulte los temas dedicados a cada una de ellas, que aparecen a continuación.

### Consulte

- [Capítulo 12, “Tutorial: Creación de diseños GridBagLayout en JBuilder”](#) si desea más información sobre el establecimiento de restricciones `GridBagConstraints` en el diseñador.

## Configuración manual de GridBagConstraints en el código fuente

---

Cuando se diseña un contenedor `GridBagLayout` mediante el diseñador, JBuilder crea un objeto `GridBagConstraints` por cada componente añadido a ese contenedor. `GridBagConstraints` tiene un constructor que lleva las once propiedades de `GridBagConstraints`.

Por ejemplo:

```
jPanel1.add(gridControl1,
    new GridBagConstraints(0, 0, 1, 1, 1.0, 1.0,
        GridBagConstraints.CENTER,
        GridBagConstraints.BOTH,
        new Insets(35, 73, 0, 0), 0, 0));
jPanel1.add(treeControl1,
    new GridBagConstraints(1, 0, 1, 2, 1.0, 1.0,
        GridBagConstraints.CENTER,
        GridBagConstraints.BOTH,
        new Insets(5, 0, 162, 73), 0, 0));
```

Los parámetros del constructor de `GridBagConstraints` pueden modificarse directamente en el código fuente o mediante el Editor de `GridBagConstraints` en el diseñador.

Si crea contenedores `GridBagLayout` escribiendo código manualmente, basta con que cree un objeto `GridBagConstraints` por cada contenedor `GridBagLayout`. `GridBagLayout` aplica los valores por defecto de `GridBagConstraints` a cada componente que se añade al contenedor o reutiliza los valores modificados más recientemente. Si desea que el componente que está añadiendo al

contenedor tenga un valor distinto para una restricción concreta, basta con especificar el valor deseado en ese componente. Este nuevo valor se aplica en los componentes que añada posteriormente, hasta que vuelva a cambiarlo.

**Nota** Si bien esta metodología de codificación de `GridBagLayout` produce el código más reducido (se reciclan valores de restricciones tomados de componentes previamente añadidos), también impide la edición visual del contenedor en el diseñador de JBuilder.

## Modificación del código de GridBagLayout para que funcione en el diseñador

---

Si ha escrito manualmente el código de un `GridBagLayout`, mediante un objeto `GridBagConstraints`, no puede editar ese contenedor en el diseñador a menos que introduzca las siguientes modificaciones en el código:

- Debe crear otro objeto `GridBagConstraints` con JDK 1.3 por cada componente que se añada al contenedor, que tiene el gran constructor con parámetros para los once valores de restricción, como se muestra más arriba.

## Diseño visual de GridBagLayout en el diseñador

---

`GridBagLayout` es un gestor de diseño complejo que exige cierta cantidad de estudio y experiencia para entenderlo, pero que, una vez se domina, resulta enormemente útil. JBuilder ha ampliado las herramientas de diseño visual con ciertas características que facilitan en gran medida el diseño y el control de `GridBagLayout`; por ejemplo: un Editor de `GridBagConstraints`, una rejilla, funciones de edición de arrastrar y soltar y un menú contextual para los componentes seleccionados.

Existen dos estrategias posibles para diseñar un `GridBagLayout` en el diseñador. Puede diseñarlo desde cero añadiendo componentes a un panel `GridBagLayout` o comenzar creando, en el diseñador, un prototipo del panel mediante otro diseño, como `XYLayout`, para convertirlo posteriormente en `GridBagLayout` cuando tenga todos los componentes organizados y dimensionados como desea.

Con independencia del método que emplee, es aconsejable que recurra a la anidación de paneles para agrupar los componentes, organizando el proceso de adentro afuera. Emplee estos paneles para definir las áreas importantes del contenedor `GridBagLayout`. De esta forma, simplifica notablemente el diseño de `GridBagLayout`, ya que tendrá menos celdas en la rejilla y menos componentes que necesiten `GridBagConstraints`.

## Conversión a GridBagLayout

---

Si opta por crear primero un prototipo a partir de otro diseño, como `XYLayout`, la conversión a `GridBagLayout` resulta mucho más limpia y sencilla si cuida la alineación de los paneles y los componentes cuando los coloque por primera vez, especialmente la alineación a la izquierda y a la parte superior. Tenga presente que está diseñando una rejilla. Intente colocar los componentes en una rejilla imaginaria y recurra a paneles anidados para reducir al mínimo el número de filas y columnas.

El empleo de `XYLayout` para crear prototipos cuenta con la ventaja de proporcionar funciones de alineación de componentes en su menú contextual.

Dado que el diseñador de interfaces de usuario convierte el contenedor de `XYLayout` en `GridBagLayout`, asigna valores de restricciones a los componentes en función del lugar que éstos ocupaban antes de realizar la conversión. Por lo general, solamente se requieren ajustes menores, si es que se requiere alguno.

Al efectuar la conversión a `GridBagLayout`, se asignan restricciones de peso a ciertos tipos de componentes (aquellos para los que normalmente sería previsible un aumento de tamaño cuando el contenedor se amplíe en tiempo de ejecución, como áreas de texto, campos, cuadros de grupo y listas). Si necesita introducir retoques en el diseño después de la conversión a `GridBagLayout`, se facilitará mucho la tarea si elimina previamente las restricciones de peso de los componentes (asignarles el valor cero).

Basta con que un componente tenga una restricción de peso mayor que cero para que sea difícil predecir cómo cambiará su tamaño en el diseñador, dadas las complejas interacciones entre los componentes del contenedor.

Es fácil detectar un `GridBagLayout` cuyos componentes tienen pesos no nulos, ya que los componentes no se agrupan en el centro del contenedor. En lugar de ello, ocupan todo el espacio disponible hasta los bordes del contenedor.

### Sugerencia

Cuando se anulan todos los pesos de los componentes de un `GridBagLayout`, existen dos posibilidades:

- Si el contenedor es lo bastante amplio para la rejilla, los componentes se agrupan en el centro del contenedor, con el espacio libre alrededor de los bordes de la rejilla.
- Si el contenedor es demasiado pequeño para los componentes, la rejilla se amplía hasta superar los bordes del contenedor y los componentes situados fuera de los bordes del contenedor quedan ocultos. Basta con aumentar el tamaño del contenedor hasta que dé cabida a todos los componentes. Si el contenedor `GridBagLayout` que está diseñando consta de un solo panel situado en el centro del marco principal de la interfaz de usuario, aumente el tamaño del marco. Podrá darle al contenedor el tamaño final cuando haya establecido las restricciones de todos los componentes.

**Consulte**

- “[weightx, weighty](#)” en la [página 8-41](#) para obtener más información sobre las restricciones `weight`.

## Incorporación de componentes a un contenedor GridBagLayout

---

Si desea partir de un contenedor `GridBagLayout` nuevo y añadirle todos los componentes desde cero, hay ciertos efectos que debe tener en cuenta.

- Dado que la restricción de peso por defecto es cero para todos los componentes, cuando añada el primer componente al contenedor, aparecerá en el centro con `minimumSize`. En este momento tiene una rejilla con una fila y una columna.
- El siguiente componente que añada se sitúa en una celda contigua, dependiendo del lugar en el que haga clic. Si hace clic por debajo del primer componente, se sitúa en la siguiente fila de esa columna. Si hace clic a la derecha del componente, se sitúa en la misma fila, pero en la siguiente columna. Los componentes añadidos posteriormente se ubicarán de la misma forma y por cada uno se añadirá una fila.
- Una vez que disponga de varios componentes, o de varias celdas que contengan componentes, puede arrastrarlos hasta otras celdas con el ratón o cambiar las restricciones `gridx` y `gridy` en el Editor de `GridBagConstraints`, disponible en el menú contextual de la superficie de diseño del componente.
- Con independencia de los componentes que añada, se agruparán todos en el centro del contenedor, a no ser que el tamaño de la rejilla supere al de este último. Si necesita un contenedor de mayor tamaño, basta con ampliarlo en el diseñador.
- Si ha ido creando filas y el diseño ha encajado correctamente en un determinado número de columnas y llega a una fila donde debe situar un número impar de componentes, considere la posibilidad de colocar en ella un panel que la ocupe por completo. Entonces podrá emplear en ese panel un diseño distinto que le permitirá conseguir el resultado final que desea.

Un buen ejemplo es el editor de la propiedad `Sort` que se muestra al final de este apartado. Todos los componentes del contenedor encajan en dos columnas, excepto los tres botones de la parte inferior. Si intenta añadir estos botones a la fila por separado, `GridBagLayout` no los gestionará correctamente. Se crearán otras columnas que influirán en la ubicación de los componentes situados por encima de los botones. Para simplificar la rejilla y cerciorarse de que los botones se comportarán de la forma esperada cuando el contenedor se redimensione durante la ejecución, puede emplear un panel `GridLayout` de dos columnas que los contenga.

## Configuración de GridBagConstraints en el Editor de GridBagConstraints

Todas las `GridBagConstraints` pueden establecerse en el diseñador sin necesidad de modificar el código fuente. Ello se consigue mediante los editores de `GridBagLayout` y de `GridBagConstraints`.

Una ventaja de utilizar el Editor de `GridBagConstraints` para establecer las restricciones es la posibilidad de cambiar simultáneamente las de varios componentes. Por ejemplo, si desea que los botones del contenedor `GridBagLayout` tengan el mismo tamaño adicional interno, puede mantener pulsada la tecla *Mayús* mientras los selecciona y, a continuación, abrir el Editor `GridBagConstraints` y cambiar el valor de esa restricción.

Para utilizar el Editor de `GridBagConstraints`:

- 1 Seleccione los componentes de `GridBagLayout` que desea modificar en el árbol de componentes o en la superficie de diseño.
- 2 Abra el Editor de `GridBagConstraints` de una de las siguientes formas:
  - Seleccione la propiedad `constraints` en el Inspector y, a continuación, pulse el botón de puntos suspensivos.
  - Haga clic con el botón derecho sobre el componente en la superficie de diseño y elija Restricciones.
  - Seleccione el componente en el árbol de componentes, pulse *Mayús+F10* y elija Restricciones.
- 3 Defina las restricciones en el editor de propiedades. A continuación pulse **Aceptar**.

**Nota** Si necesita ayuda para utilizar el Editor de `GridBagConstraints`, pulse el botón **Ayuda** o la tecla *F1*.

## Visualización de la rejilla

---

La superficie de diseño muestra una rejilla optativa que permite ver exactamente qué sucede con cada celda y componente del diseño.

- Para visualizar esta rejilla, haga clic sobre un componente del contenedor `GridBagLayout` y elija **Mostrar rejilla**. Junto a la opción de menú, aparece una marca de selección.
- Para ocultar la rejilla temporalmente, aunque **Mostrar rejilla** esté seleccionada, haga clic en un componente que no se encuentre en el contenedor `GridBagLayout` (también sirve el propio contenedor). La rejilla solamente está visible cuando un componente de `GridBagLayout` está seleccionado.
- Si desea ocultarla permanentemente, haga clic con el botón derecho en un componente y seleccione de nuevo **Mostrar rejilla** para que desaparezca la marca de selección.

## Utilización del ratón para cambiar restricciones

---

La superficie de diseño permite determinar algunas de las restricciones mediante el ratón: se puede arrastrar el componente deseado o tirar de sus tiradores de redimensionamiento. En los siguientes apartados dedicados a las restricciones se ofrecen instrucciones para establecer visualmente cada una de ellas.

## Utilización del menú contextual de GridBagLayout

---

Si hace clic con el botón derecho en un componente de `GridBagLayout` o lo selecciona y pulsa **Mayús+F10**, aparece un menú contextual que proporciona acceso inmediato al Editor de `GridBagConstraints`, con el fin de establecer y eliminar rápidamente determinadas restricciones.

Comando de menú	Acción
Mostrar rejilla	Muestra la rejilla de <code>GridBagLayout</code> en el diseñador de interfaces de usuario.
Restricciones	Muestra el Editor de <code>GridBagConstraints</code> del componente <code>GridBagLayout</code> seleccionado.
Eliminar tamaño adicional	Establece a cero los valores del tamaño adicional ( <code>ipadx</code> y <code>ipady</code> ) para el componente seleccionado.
Relleno horizontal	Cambia el valor de la restricción <code>fill</code> para el componente a <code>HORIZONTAL</code> . El componente se expande para ocupar el espacio horizontal disponible en la celda. Si <code>Expansión</code> tenía el valor <code>VERTICAL</code> , define la restricción como <code>AMBAS</code> .



Comando de menú	Acción
Relleno vertical	Cambia el valor de la restricción <code>fill</code> para el componente a <code>VERTICAL</code> . El componente se expande para ocupar el espacio vertical disponible en la celda. Si <code>Expansión</code> tenía el valor <code>HORIZONTAL</code> , define la restricción como <code>AMBAS</code> .
Eliminar relleno	Cambia el valor de la restricción <code>Expansión</code> para el componente a <code>ANULADA</code> .
Peso horizontal	Asigna 1.0 al componente la restricción de peso <code>weightx</code> .
Peso vertical	Asigna 1.0 al componente la restricción de peso <code>weighty</code> .
Eliminar pesos	Asigna 0.0 a las restricciones para el componente, <code>weightx</code> y <code>weighty</code> .

## GridBagConstraints

El siguiente apartado presenta las distintas `GridBagConstraints` por separado. Define cada una de ellas y especifica los valores válidos y por defecto; indica igualmente el modo de proceder para definir visualmente esa restricción en el diseñador.

### Consulte

- [Capítulo 12, “Tutorial: Creación de diseños GridBagLayout en JBuilder”](#) si desea más información sobre el establecimiento de restricciones `GridBagConstraint` en el diseñador.

### ancla

Cuando el componente sea más pequeño que el área en que se visualiza, utilice la restricción `ancla` para indicar al gestor de diseño dónde debe situar el componente dentro del área.

La restricción `ancla` sólo afecta al componente dentro de su propia área de visualización, en función de la restricción `expansión` del componente. Por ejemplo, si el valor de la restricción `expansión` de un componente es `GridBagConstraints.BOTH` (`AMBAS`, es decir que toda el área de visualización está ocupada, horizontal y verticalmente), la restricción `ancla` no tiene efecto alguno porque el componente ocupa la totalidad del área disponible. Para que la restricción `ancla` tenga efecto, asigne a la restricción `expansión` el valor `GridBagConstraints.NONE` (`ANULADA`), `GridBagConstraints.HORIZONTAL` (`HORIZONTAL`) o `GridBagConstraints.VERTICAL`.

### Definición de la restricción de ancla

Puede emplear el ratón para establecer la restricción `ancla` de los componentes de tamaño inferior a la celda que los contiene. Basta con hacer clic en el componente y arrastrarlo hacia la ubicación deseada, al borde de su área de visualización, de forma parecida a como se ancla una barra de herramientas móvil. Por ejemplo, para anclar un botón a la esquina superior

izquierda de la celda, haga clic en el centro del botón y arrástrelo hasta que la esquina superior izquierda toque la de la celda. De esta forma, el valor de `ancla` pasa a ser `NO`.

También puede modificarse la restricción `ancla` en el Editor de `GridBagConstraints`.

- 1 Active el menú contextual de diseño del componente de una de estas dos formas:
  - Haga clic con el botón derecho del ratón en el componente, en la superficie de diseño.
  - Seleccione el componente en el árbol de componentes y pulse *Mayús+F10*.
- 2 Elija Restricciones.
- 3 Seleccione el valor de `ancla` deseado en la zona Ancla y, a continuación, pulse Aceptar.
- 4 Pulse Aceptar.

## expansión

Si el área de visualización en la que se inserta un componente es superior a este último, la restricción `expansión` le permite indicar al gestor de diseño las zonas del área que debe asignar al componente.

En el caso de la restricción `ancla`, las restricciones de `expansión` solamente afectan al componente dentro de su área de visualización. Las restricciones de `expansión` indican al gestor de diseño que debe expandir el componente para que ocupe el área completa que se le da.

## Definición de la restricción de expansión

La manera más rápida de especificar las restricciones de `expansión` en un componente consiste en utilizar el menú contextual de ese componente en la superficie de diseño.

- 1 Active el menú contextual de diseño del componente de una de estas dos formas:
  - Haga clic con el botón derecho del ratón en el componente, en la superficie de diseño.
  - Seleccione el componente en el árbol de componentes y pulse *Mayús+F10*.
- 2 Realice una de las operaciones siguientes:
  - Seleccione Relleno horizontal para establecer el valor `HORIZONTAL`.
  - Seleccione Relleno vertical para establecer el valor `VERTICAL`.
  - Seleccione Relleno horizontal y Expandir en vertical para establecer el valor `AMBAS`.
  - Seleccione Eliminar relleno para establecer el valor `ANULADA`.

También puede definir la restricción `expansión` en el Editor de GridBagConstraints.

- 1 Active el menú contextual de diseño del componente de una de estas dos formas:
  - Haga clic con el botón derecho del ratón en el componente, en la superficie de diseño.
  - Seleccione el componente en el árbol de componentes y pulse *Mayús+F10*.
- 2 Elija Restricciones.
- 3 Seleccione el valor de `expansión` en la zona Expansión y pulse Aceptar.

## gridwidth, gridheight

Utilice estas restricciones para especificar el número de celdas de las que se compone una fila (`gridwidth`) o una columna (`gridheight`) del componente. El valor de esta restricción se introduce en número de celdas y no en píxeles.

### Definición de las restricciones gridwidth y gridheight

También puede definir los valores de las restricciones `gridwidth` y `gridheight` en el Editor de GridBagConstraints.

- 1 Active el menú contextual de diseño del componente de una de estas dos formas:
  - Haga clic con el botón derecho del ratón en el componente, en la superficie de diseño.
  - Seleccione el componente en el árbol de componentes y pulse *Mayús+F10*.
- 2 Elija Restricciones.
- 3 En el área Rejilla, introduzca el valor de `gridwidth`, en el campo Anchura, o el de `gridheight`, en el campo Altura. Indique el número de celdas que ha de ocupar el componente en la celda o la columna.
  - Si desea que el valor sea de tipo `RELATIVO`, introduzca -1.
  - Si desea que el valor sea de tipo `RESTO`, introduzca 0.

El valor de `gridwidth` y `gridheight` también puede cambiarse con el ratón: basta variar el tamaño del componente hasta que ocupe las celdas contiguas.

## gridx, gridy

Utilice estas restricciones para especificar la posición de la celda donde se encuentra la esquina superior izquierda del componente. Por ejemplo, `gridx=0` es la primera columna de la izquierda, mientras que `gridy=0` es la primera fila de la parte superior. Por lo tanto, un componente que tenga las restricciones `gridx=0` y `gridy=0` se situará en la primera celda de la rejilla (arriba a la izquierda).

`GridBagConstraints.RELATIVE` especifica que el componente debe situarse de forma relativa al componente anterior, como se explica a continuación:

- Cuando se utiliza con `gridx`, indica que el componente debe situarse inmediatamente a la derecha del último componente añadido.
- Cuando se utiliza con `gridy`, indica que el componente debe situarse inmediatamente debajo del último componente añadido.

### Definición de la ubicación de la celda de rejilla

Puede emplear el ratón para seleccionar la celda cuya esquina superior derecha ha de ocupar el componente. Basta hacer clic cerca de la esquina superior izquierda del componente y arrastrarlo hasta la celda deseada. Si desplaza componentes que ocupen varias celdas, cerciórese de hacer clic en la esquina superior izquierda del componente; si no lo hace así, obtendrá resultados no deseados. En ocasiones, debido a los valores de las restricciones del componente, cuando este se desplaza a otra celda mediante el ratón, se alteran otras restricciones (por ejemplo, puede cambiar el número de celdas ocupado por el componente).

El Editor de `GridBagConstraints` permite modificar con más precisión `gridx` y `gridy` sin alterar accidentalmente otras restricciones.

- 1 Active el menú contextual de diseño del componente de una de estas dos formas:
  - Haga clic con el botón derecho del ratón en el componente, en la superficie de diseño.
  - Seleccione el componente en el árbol de componentes y pulse *Mayús+F10*.
- 2 Elija Restricciones.
- 3 En el área de la rejilla, ingrese el número de columna para el valor `gridx` en el campo X o el número de fila para el valor `gridy` en el campo Y. Si desea que el valor sea de tipo `RELATIVO`, introduzca -1.

#### Importante

Si emplea el ratón para desplazar el componente a una celda ocupada, el diseñador de interfaces de usuario impide la superposición de componentes insertando una fila y una columna. Si reubica el componente mediante el Editor de `GridBagConstraints`, el diseñador **no** comprueba si los componentes se superponen.

### encuadres

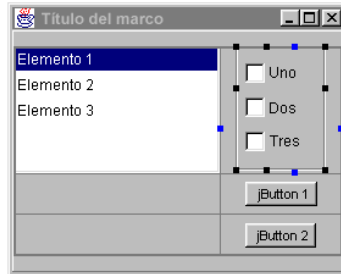
Utilice `encuadres` para especificar el espacio externo (márgenes) en píxeles, que media entre el componente y los bordes de su área de visualización. La propiedad `encuadre` establece un espacio constante entre el borde del componente y el borde de la celda que lo contiene. Por lo tanto, `encuadre` actúa como “frontera” del componente, con el objeto de mantenerlo alejado de los bordes de la celda. Por ejemplo, si altera la anchura de un componente ampliando el `encuadre` a la izquierda y a la derecha y supera los límites de la celda, esta se expande para acomodar tanto al componente como a sus

encuadres. Debido a ello, las restricciones expansión y tamaño adicional nunca restan espacio a los encuadres.

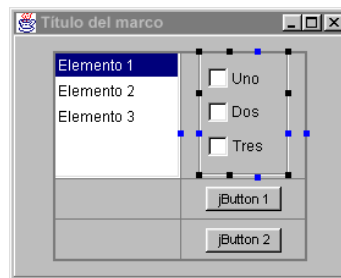
### Definición de valores de encuadres

La superficie de diseño muestra tiradores azules de redimensionamiento en los componentes seleccionados de `GridBagLayout` para señalar la ubicación y el tamaño de sus encuadres. El tamaño de los encuadres se varía arrastrando los tiradores azules con el ratón.

- Si el valor de `encuadre` es cero, solo se ve un tirador azul en el lado correspondiente de la celda, como se muestra a continuación:



- Si el valor de `encuadre` es mayor que cero, la superficie de diseño muestra dos tiradores azules en el lado correspondiente: uno en el borde de la celda y otro en el borde del área de visualización. El tamaño del `encuadre` es la distancia en píxeles entre los dos tiradores. Para redimensionar el `encuadre`, basta arrastrar uno de los tiradores.



Si desea un control más preciso de los valores de `encuadre`, puede especificar el número de píxeles exacto mediante el Editor de `GridBagConstraints`.

- Haga clic con el botón derecho en el componente (en el diseñador de interfaces de usuario) y seleccione Restricciones; esto mostrará el Editor de `GridBagConstraints`.
- En la zona Encuadre, indique el número de píxeles para cada uno de los encuadres: superior, izquierdo, inferior o derecho.

**Nota** Pese a que `encuadre` admite valores negativos, no es recomendable emplearlos, ya que el componente se superpondría a los componentes contiguos.

## ipadx, ipady

Estas restricciones indican el tamaño adicional de un componente:

- `ipadx` especifica el número de píxeles que debe añadirse a la anchura mínima del componente.
- `ipady` especifica el número de píxeles que debe añadirse a la altura mínima del componente.

Utilice `ipadx` e `ipady` para especificar el número de píxeles con los que se quiere incrementar el tamaño mínimo del componente, es decir, el tamaño adicional que se le quiere dar. Por ejemplo, la anchura del componente será, al menos, la anchura mínima más `ipadx` en píxeles. El código sólo lo añade una vez y lo divide por igual entre ambos lados del componente. De forma parecida, la altura del componente será de al menos la altura mínima más `ipady` en píxeles.

## Ejemplo

Cuando se añaden a un componente que tiene un tamaño recomendado de 30 píxeles de ancho y 20 píxeles de alto:

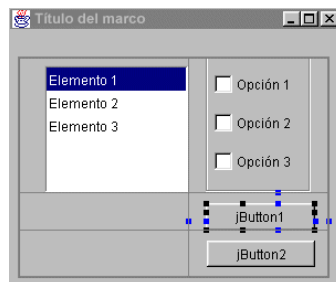
- Si `ipadx= 4`, el componente tiene una anchura de 34 píxeles.
- Si `ipady= 2`, el componente tiene una altura de 22 píxeles.

## Definición del tamaño de las restricciones de relleno

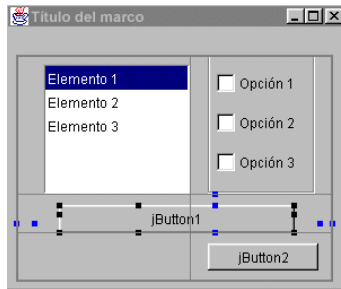
Puede especificar el tamaño adicional de un componente haciendo clic en los tiradores negros que aparecen en los bordes del componente y arrastrándolos con el ratón.

Si arrastra el tirador hasta que supere el borde de la celda e invada la celda contigua, el componente ocupa ambas celdas (el valor de `gridwidth` o `gridheight` se incrementa en una celda).

Antes:



Después:



Si desea controlar con más precisión los valores de tamaño adicional, el Editor de GridBagConstraints le permitirá especificar el número exacto de píxeles.

- 1 Active el menú contextual de diseño del componente de una de estas dos formas:
  - Haga clic con el botón derecho del ratón en el componente, en la superficie de diseño.
  - Seleccione el componente en el árbol de componentes y pulse *Mayús+F10*.
- 2 Elija Restricciones.
- 3 En el área Tamaño adicional, indique el número de píxeles deseado en los campos Anchura y Altura.

Para eliminar rápidamente el tamaño adicional (asignarle el valor cero), haga clic con el botón derecho en el diseñador de interfaces de usuario y elija Eliminar tamaño adicional. También puede seleccionar varios componentes y por el mismo procedimiento eliminar de una sola vez el tamaño adicional de todos ellos.

**Nota** Los valores negativos hacen que el componente sea menor que su tamaño recomendado y son perfectamente válidos.

## weightx, weighty

Las restricciones de peso permiten establecer cómo se distribuye el espacio adicional de los contenedores `GridBagLayout`, horizontal (`weightx`) y verticalmente (`weighty`), cuando se redimensionan. Los pesos determinan qué parte del espacio adicional deben obtener las celdas y los componentes cuando se aumenta el tamaño del contenedor más allá de su tamaño por defecto.

Los valores de peso son del tipo `double` y se expresan como una proporción. Solo se admiten los valores positivos. Se admiten todos los formatos de proporción. Asigne mentalmente un peso total a los componentes de la misma fila o columna y reparta ese peso entre los componentes. Cuando sume todos los pesos de los componentes deberá obtener el peso total que había calculado.

- El peso vertical de una fila determina la altura de esa fila en relación con las demás filas. Este peso iguala al valor `weighty` más grande de los

componentes de la fila. El valor determinante es la altura, que se mide en el eje y.

- El peso horizontal de una columna determina el ancho de esa columna en relación con las demás columnas. Este peso iguala al valor `weightx` más grande de los componentes de la columna. El factor determinante es el ancho, que se mide en el eje x.

En teoría, sólo los componentes más grandes de una fila o columna determinan el diseño, con lo que solo se necesita un componente por fila o columna para determinar el peso.

### Definición de las restricciones `weightx` y `weighty`

Para definir en el diseñador las restricciones de `peso` de un componente, abra su menú contextual de diseño. Seleccione el componente en el árbol y pulse `Mayús+F10` o haga clic con el botón derecho del ratón en el componente y elija `Peso horizontal (x)` (`weightx`) o `Peso vertical (y)` (`weighty`). Con ello, se elige el valor 1.0. Para anular los pesos, haga clic con el botón derecho sobre el componente y elija `Eliminar pesos`. Esta operación puede aplicarse simultáneamente a varios componentes: mantenga pulsada la tecla `Mayús` y seleccione los componentes deseados; a continuación, haga clic con el botón derecho y elija la opción de menú correspondiente.

Si desea que las restricciones de `peso` tengan un valor distinto de 0,0 y 1,0, puede introducir el que desee mediante el Editor de `GridBagConstraints`.

- 1 Active el menú contextual de diseño del componente de una de estas dos formas:
  - Haga clic con el botón derecho del ratón en el componente, en la superficie de diseño.
  - Seleccione el componente en el árbol de componentes y pulse `Mayús+F10`.
- 2 Elija `Restricciones`.
- 3 Introduzca un valor comprendido entre 0.0 y 1.0 en los campos X (`weightx`) e Y (`weighty`) de la zona `Peso` y, a continuación, pulse `Aceptar`.

#### Importante

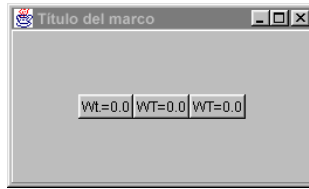
Dado que las restricciones de `peso` añaden complejidad al redimensionamiento en el diseñador de interfaces y los resultados obtenidos son difíciles de predecir. Establecer estas restricciones ha de ser el último paso en el diseño de un `GridBagLayout`.

### Ejemplos de cómo las restricciones de peso afectan al comportamiento de los componentes

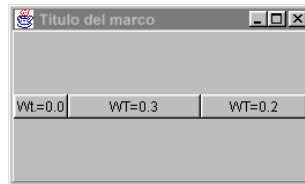
- Si todos los componentes tienen una restricción de `peso` cero en una sola dirección, los componentes se agrupan en el centro del contenedor en dicha dimensión y no sobrepasarán sus tamaños recomendados.



`GridBagLayout` añade un espacio adicional entre la rejilla de celdas y los bordes del contenedor.



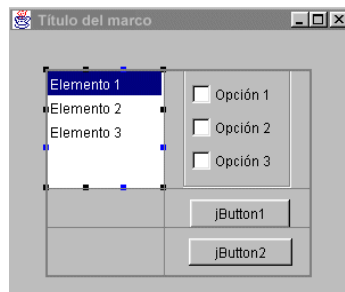
- Por ejemplo, si tiene tres componentes con la restricción `weightx` de 0,0, 0,6 y 0,4 respectivamente y aumenta el tamaño del contenedor, ninguna parte del espacio sobrante se asigna al primer componente, pero las 6/10 partes sí se aplican al segundo componente, y otras 4/10 partes al tercero.



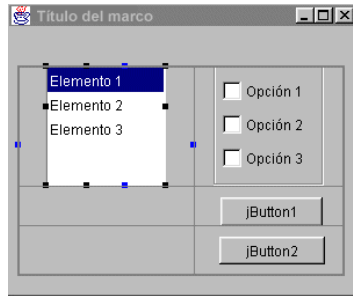
- Si desea que el tamaño de los componentes pueda aumentar, ha de establecer las restricciones `peso` y `expansión`. Por ejemplo, si un componente tiene la restricción `peso`, pero no la restricción `expansión` horizontal, el espacio adicional se asigna al tamaño adicional entre los bordes izquierdo y derecho del componente y los bordes de la celda. Aumenta la anchura de la celda sin variar el tamaño del componente. Si un componente tiene las restricciones `peso` y `expansión`, el espacio adicional se añade a la celda y el componente se amplía para ocupar ese incremento de tamaño en la dirección de la restricción `expansión` (en este caso, horizontal).

Este efecto se muestra en las tres figuras siguientes.

En el primer ejemplo, todos los componentes del panel `GridBagLayout` tienen una restricción de `peso` nula. Debido a ello, los componentes se agrupan en el centro del panel `GridBagLayout` y el espacio sobrante de este se distribuye en el margen entre los bordes exteriores de la rejilla y el propio panel. El tamaño de la rejilla está determinado por el tamaño recomendado de los componentes, más los `encuadres` y el tamaño adicional (`ipadx` o `ipady`).



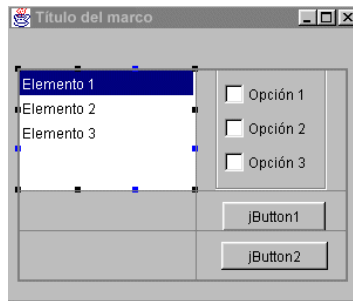
Seguidamente, se establece una restricción de `peso` horizontal con valor 1,0 para `ListControl`. Observe que, inmediatamente después de asignar un `peso` a un control, el diseño de la interfaz de usuario deja de estar centrado en el panel. Dado que se ha establecido una restricción de `peso` horizontal, el gestor de `GridBagLayout` toma el espacio sobrante del panel, previamente repartido a ambos lados de la rejilla, y lo coloca en la celda que contiene el componente `ListControl`. Observe asimismo que `ListControl` no ha cambiado de tamaño.



### Sugerencia

Si, una vez establecidas las restricciones de `peso` de los componentes, existe más espacio del deseado en las celdas, reduzca el tamaño del `marco` de la interfaz de usuario hasta conseguir la cantidad de espacio adicional que desee. Para ello, seleccione el `marco` `this(BorderLayout)` en la superficie de diseño o en el árbol de componentes; seguidamente, haga clic en los tiradores negros y arrástrelos hasta que el `marco` adquiera el tamaño deseado.

Por último, si se establece una restricción de `expansión` horizontal a `ListControl`, el componente se expande hasta ocupar el nuevo valor horizontal de la celda.



- Si un componente de una columna tiene la restricción `weightx`, `GridBagLayout` asigna la totalidad de la columna a ese valor. Recíprocamente, si un componente de una fila tiene la restricción `weighty`, la fila entera se asigna a ese `peso`.

### Consulte

- “GridBagConstraints” en la documentación del JDK

## Código fuente de ejemplo de GridBagLayout



Observe que a excepción de los tres botones de la parte inferior, el resto de los componentes encaja perfectamente en una rejilla de dos columnas. Si intenta que cada botón ocupe una celda, deberá añadir una tercera columna al diseño, con lo cual deberá distribuir los restantes componentes entre las tres columnas. Probablemente con seis columnas llegaría a una solución razonable, pero los botones no conservarían su tamaño cuando el contenedor se redimensione en tiempo de ejecución.

Existen otras dos formas de resolver este problema:

- Coloque un panel `GridLayout` con dos columnas en la parte inferior de la rejilla y añádale los tres botones.
- Coloque un panel `GridLayout`, con los tres botones dentro, en el marco `BorderLayout` exterior. Establezca la restricción `SOUTH` para el primer panel y la restricción `CENTER` para `BorderLayout`.

De cualquier forma, cuando se redimensione el contenedor, los botones se comportarán correctamente.

Este es el código fuente del ejemplo de `GridBagLayout`:

```
package sort;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import com.borland.jbcl.layout.*;

public class Frame1 extends JFrame {
    JPanel contentPane;
    BorderLayout borderLayout1 = new BorderLayout();
    JPanel jPanel1 = new JPanel();
    JPanel jPanel2 = new JPanel();
    JLabel jLabel1 = new JLabel();
    JList jList1 = new JList();
    JButton jButton1 = new JButton();
    JCheckBox jCheckBox1 = new JCheckBox();
    JButton jButton2 = new JButton();
    JCheckBox jCheckBox2 = new JCheckBox();
    JPanel jPanel3 = new JPanel();
    JList jList2 = new JList();
```

```
JLabel jLabel2 = new JLabel();
JPanel jPanel4 = new JPanel();
JButton jButton3 = new JButton();
JButton jButton4 = new JButton();
JButton jButton5 = new JButton();
GridBagLayout gridBagLayout1 = new GridBagLayout();
GridBagLayout gridBagLayout2 = new GridBagLayout();
GridBagLayout gridBagLayout3 = new GridBagLayout();
GridLayout gridLayout1 = new GridLayout();

//Construir el marco
public Marco1() {
    enableEvents(AWTEvent.WINDOW_EVENT_MASK);
    try {
        jbInit();
    }
    catch(Exception e) {
        e.printStackTrace();
    }
}

//Inicialización del componente
private void jbInit() throws Exception {
    contentPane = (JPanel) this.getContentPane();
    contentPane.setLayout(borderLayout1);
    this.setSize(new Dimension(400, 300));
    this.setTitle("Sort");
    jPanel1.setLayout(gridBagLayout3);
    jPanel2.setBorder(BorderFactory.createRaisedBevelBorder());
    jPanel2.setLayout(gridBagLayout2);
    jLabel1.setFont(new java.awt.Font("SansSerif", 0, 12));
    jLabel1.setForeground(Color.black);
    jLabel1.setText("Available columns");
    jList1.setBorder(BorderFactory.createLoweredBevelBorder());
    jButton1.setFont(new java.awt.Font("SansSerif", 0, 12));
    jButton1.setBorder(BorderFactory.createRaisedBevelBorder());
    jButton1.setText("Add to Sort");
    jCheckBox1.setText("Case insensitive");
    jCheckBox1.setFont(new java.awt.Font("Dialog", 0, 12));
    jButton2.setText("Remove from Sort");
    jButton2.setBorder(BorderFactory.createRaisedBevelBorder());
    jButton2.setFont(new java.awt.Font("SansSerif", 0, 12));
    jCheckBox2.setFont(new java.awt.Font("Dialog", 0, 12));
    jCheckBox2.setText("Descending");
    jPanel3.setLayout(gridBagLayout1);
    jPanel3.setBorder(BorderFactory.createRaisedBevelBorder());
    jList2.setBorder(BorderFactory.createLoweredBevelBorder());
    jLabel2.setFont(new java.awt.Font("SansSerif", 0, 12));
    jLabel2.setForeground(Color.black);
    jLabel2.setText("Sorted columns");
    jButton3.setText("Help");
```

```

jButton4.setText("OK");
jButton5.setText("Cancel");
jPanel4.setLayout(gridLayout1);
gridLayout1.setHgap(10);
gridLayout1.setVgap(10);
contentPane.add(jPanel1, BorderLayout.CENTER);
jPanel1.add(jPanel2, new GridBagConstraints(1, 0, 1, 1, 1.0, 1.0
    ,GridBagConstraints.CENTER, GridBagConstraints.BOTH,
        new Insets(6, 10, 0, 19), 0, 2));
jPanel2.add(jList1, new GridBagConstraints(0, 1, 1, 1, 1.0, 1.0
    ,GridBagConstraints.CENTER, GridBagConstraints.BOTH,
        new Insets(0, 7, 0, 9), 160, 106));
jPanel2.add(jButton1, new GridBagConstraints(0, 2, 1, 1, 0.0, 0.0
    ,GridBagConstraints.CENTER, GridBagConstraints.NONE,
        new Insets(8, 7, 0, 9), 90, 2));
jPanel2.add(jCheckBox1, new GridBagConstraints(0, 3, 1, 1, 0.0, 0.0
    ,GridBagConstraints.CENTER, GridBagConstraints.NONE,
        new Insets(11, 13, 15, 15), 31, 0));
jPanel2.add(jLabel1, new GridBagConstraints(0, 0, 1, 1, 0.0, 0.0
    ,GridBagConstraints.WEST, GridBagConstraints.NONE,
        new Insets(4, 7, 0, 15), 54, 8));
jPanel1.add(jPanel3, new GridBagConstraints(0, 0, 1, 1, 1.0, 1.0
    ,GridBagConstraints.CENTER, GridBagConstraints.BOTH,
        new Insets(6, 9, 0, 0), 0, 2));
jPanel3.add(jList2, new GridBagConstraints(0, 1, 1, 1, 1.0, 1.0
    ,GridBagConstraints.CENTER, GridBagConstraints.BOTH,
        new Insets(0, 7, 0, 9), 160, 106));
jPanel3.add(jButton2, new GridBagConstraints(0, 2, 1, 1, 0.0, 0.0
    ,GridBagConstraints.CENTER, GridBagConstraints.NONE,
        new Insets(8, 7, 0, 9), 50, 2));
jPanel3.add(jCheckBox2, new GridBagConstraints(0, 3, 1, 1, 0.0, 0.0
    ,GridBagConstraints.CENTER, GridBagConstraints.NONE,
        new Insets(11, 13, 15, 15), 56, 0));
jPanel3.add(jLabel2, new GridBagConstraints(0, 0, 1, 1, 0.0, 0.0
    ,GridBagConstraints.WEST, GridBagConstraints.NONE,
        new Insets(4, 7, 0, 15), 67, 8));
jPanel1.add(jPanel4, new GridBagConstraints(0, 1, 2, 1, 1.0, 1.0
    ,GridBagConstraints.CENTER, GridBagConstraints.HORIZONTAL,
        new Insets(15, 71, 13, 75), 106, 0));
jPanel4.add(jButton4, null);
jPanel4.add(jButton5, null);
jPanel4.add(jButton3, null);
}

//Sobreescrito para salir de System Close
protected void processWindowEvent(WindowEvent e) {
    super.processWindowEvent(e);
    if (e.getID() == WindowEvent.WINDOW_CLOSING) {
        System.exit(0);
    }
}
}

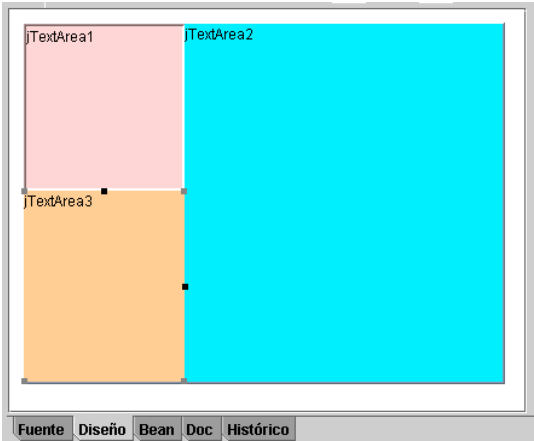
```

# PaneLayout

---

`PaneLayout` permite especificar el tamaño de un componente en relación con componentes del mismo nivel. `PaneLayout`, aplicado a un panel o a un marco, permite controlar el porcentaje del contenedor que tendrán los componentes en relación a los otros, pero no crea barras de división desplazables entre los paneles.

**Figura 8.10** Ejemplo de `PaneLayout`



En un diseño `PaneLayout`, la colocación y el tamaño de cada componente se especifica en relación a los componentes que se han añadido anteriormente al contenedor. Cada componente especifica un objeto `PaneConstraints` que indica al gestor de diseño de qué componente debe tomar espacio y qué parte del espacio existente debe tomar. Todos los objetos `PaneConstraints` del componente se aplican al contenedor tal y como existían cuando se añadieron al contenedor. Es muy importante el orden en el que se añaden los componentes al contenedor.

## Variables `PaneConstraint`

---

Los componentes `PaneConstraints` tienen una restricción que consta de cuatro variables: cadena `name`, cadena `splitComponentName`, cadena `position` y float `proportion`.

Cadena <code>name</code>	El nombre de este componente (debe ser único para todos los componentes del contenedor, como en <code>CardLayout</code> ).
Cadena <code>splitComponentName</code>	El nombre del componente del que tomará espacio para conseguir espacio para este componente.

Cadena <code>position</code>	<p>El borde de <code>splitComponentName</code> al que se ancla este componente.</p> <p>Estos valores colocan el componente en:</p> <table> <tr> <td><code>PaneConstraints.TOP</code></td><td>Cadena <code>splitComponentName</code>.</td></tr> <tr> <td><code>PaneConstraints.BOTTOM</code></td><td>Cadena <code>splitComponentName</code>.</td></tr> <tr> <td><code>PaneConstraints.RIGHT</code></td><td>A la derecha de <code>splitComponentName</code>.</td></tr> <tr> <td><code>PaneConstraints.LEFT</code></td><td>A la izquierda de <code>splitComponentName</code>.</td></tr> <tr> <td><code>PaneConstraints.ROOT</code></td><td>Este componente es el primer componente añadido.</td></tr> </table>	<code>PaneConstraints.TOP</code>	Cadena <code>splitComponentName</code> .	<code>PaneConstraints.BOTTOM</code>	Cadena <code>splitComponentName</code> .	<code>PaneConstraints.RIGHT</code>	A la derecha de <code>splitComponentName</code> .	<code>PaneConstraints.LEFT</code>	A la izquierda de <code>splitComponentName</code> .	<code>PaneConstraints.ROOT</code>	Este componente es el primer componente añadido.
<code>PaneConstraints.TOP</code>	Cadena <code>splitComponentName</code> .										
<code>PaneConstraints.BOTTOM</code>	Cadena <code>splitComponentName</code> .										
<code>PaneConstraints.RIGHT</code>	A la derecha de <code>splitComponentName</code> .										
<code>PaneConstraints.LEFT</code>	A la izquierda de <code>splitComponentName</code> .										
<code>PaneConstraints.ROOT</code>	Este componente es el primer componente añadido.										
Coma flotante <code>proportion</code>	La proporción de <code>splitComponentName</code> que se asignará a este componente. Un número entre 0 y 1.										

## Cómo se añaden los componentes a PaneLayout

`PaneLayout` añade los componentes a los contenedores en la siguiente forma:

- El primer componente siempre ocupa toda el área del contenedor. La única variable importante del diseño `PaneConstraint` es `name` (el nombre), de forma que otros componentes tengan un valor que pueden especificar como su `splitComponentName`.
- El segundo componente no puede elegir su `splitComponentName`. Debe utilizar el `name` del primer componente.
- El `splitComponentName` de los demás componentes puede ser el valor de `name` de cualquier componente añadido anteriormente al contenedor.

## Creación de un contenedor PaneLayout en el diseñador

Para crear y rellenar un contenedor `PaneLayout`:

- 1 Añada un contenedor a la interfaz de usuario en el diseñador.
- 2 Cambie la propiedad `Layout` del contenedor a `PaneLayout`. Esto le permite acceder a las propiedades de `PaneLayout` en el Inspector y cambiar la anchura de las barras de división.
- 3 Suelte un componente en el contenedor `PaneLayout`. Este componente llenará el contenedor mientras no añada otro componente para dividirlo.
- 4 Seleccione el componente.

Para cambiar el tamaño mediante el ratón:

- a Coloque el cursor en la esquina que anclará parte del componente.
- b Mantenga pulsado el botón del ratón y arrastre el cursor a través de la esquina del componente original, para anclarlo en dos dimensiones.

Una de las anclas define dónde desea dividir el área entre los componentes.

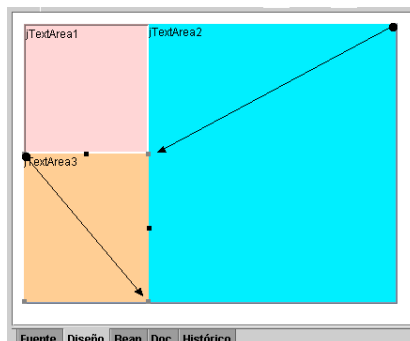
Para cambiar el tamaño del componente mediante el teclado:

- a Seleccione la propiedad `constraints` en el Inspector.
  - b Pulse la *barra espaciadora* para activar la celda del Inspector para la propiedad `constraints`.
  - c Pulse *Ctrl+Tab* para mover el foco al botón de puntos suspensivos (...) de la celda del Inspector.
  - d Pulse la *barra espaciadora* para activar el editor de propiedades de `constraints`.
  - e En el editor de propiedades de `PaneConstraints`, utilice el teclado para definir la posición del componente dentro del panel y la proporción de espacio que ocupa el componente.
- 5 Para añadir un tercer componente a `PaneLayout`, dibújelo de la misma forma para definir su posición con respecto a los otros componentes.
  - 6 Utilice el mismo método para añadir otros componentes.

Por ejemplo:

- Si desea que los paneles separen verticalmente dos mitades, izquierda y derecha, del panel, arrastre el ratón desde la esquina superior izquierda hasta la mitad del borde inferior.
- Si desea que los paneles separen horizontalmente dos mitades, superior e inferior, del panel, arrastre el ratón desde la esquina superior izquierda hasta la mitad del borde derecho.
- Por ejemplo, para dividir la parte izquierda del contenedor, comience a dibujar la posición del tercer componente desde la mitad del borde izquierdo del panel hacia la esquina inferior izquierda del segundo componente.

En la imagen que se muestra a continuación, las flechas indican las vías del cursor que se han utilizado para obtener el tamaño de los componentes que se muestran. Para que los bordes de los componentes sean más visibles en esta imagen, se ha añadido un borde biselado a cada componente.





**Importante** Si el primer componente añadido al contenedor `PaneLayout` es también un contenedor, el diseñador de interfaces de usuario supone que desea añadir el segundo componente al contenedor exterior, en lugar de al contenedor `PaneLayout`. Para indicar al diseñador de interfaces de usuario que desea añadir componentes a un contenedor que no se encuentra en la parte superior del eje Z, seleccione el contenedor de destino y mantenga pulsada la tecla `Ctrl` mientras hace clic o arrastra el componente en la superficie de diseño.

## Modificación de la ubicación y del tamaño del componente en el Inspector

---

Puede utilizar el Inspector para modificar cualquier borde `splitComponent` al que deba anclarse un componente y la proporción de `splitComponent` que dicho componente debe ocupar.

Para ello:

- 1 Seleccione el componente.
- 2 En el Inspector, seleccione la propiedad `constraints`. A continuación pulse el botón de puntos suspensivos para abrir el editor de la propiedad `Constraints`.
- 3 Seleccione una de las posiciones siguientes: Top (Arriba), Bottom (Abajo), Left (Izquierda), Right (Derecha) o Root (Raíz). Estos valores son relativos al componente que se indica en el campo `Divide a` del editor de la propiedad.
- 4 Especifique la proporción que debe ocupar este componente dentro del componente que divide.
- 5 Pulse Aceptar.

**Nota** También puede cambiar el tamaño del componente, seleccionándolo y arrastrando los tiradores de redimensionamiento. También se permite trasladar un componente; no obstante, con ello cambiará el orden de incorporación de los componentes.

## Creación de un prototipo de la interfaz de usuario

---

Antes de comenzar la creación de la interfaz de usuario, es recomendable que haga un boceto en papel del diseño, con el objeto de obtener una idea de la estrategia general que utilizará para colocar los distintos paneles y componentes y asignar los diseños. También puede crear directamente un prototipo de la interfaz de usuario en el diseñador. `JBuilder` proporciona un asistente de diseño `null` y `XYLayout`, que facilitan las tareas iniciales de diseño.

## Los diseños `XYLayout` y `null` en la creación de prototipos

---

Cuando añada un panel de cualquier tipo al diseñador, notará que la propiedad `layout` que se muestra en el Inspector indica `<diseño por defecto>`. Esto significa que el diseñador utilizará automáticamente el diseño por defecto de este contenedor. Sin embargo, debería cambiar inmediatamente la propiedad `layout` al gestor de diseño que desea utilizar, de forma que este aparezca en el árbol de componentes y pueda modificar sus restricciones en el Inspector. No es posible cambiar las propiedades del `<diseño por defecto>`.

Para controlar el diseño de los componentes durante la creación de prototipos, cambie los contenedores a `XYLayout` o `null` inmediatamente después de colocarlos en el diseño. Estos diseños utilizan coordenadas de píxeles para colocar los componentes. Esto significa que los componentes que se añaden a un contenedor se quedan en el lugar en el que se han situado y conservan el tamaño establecido con el ratón.

### Consulte

- [“Diseños suministrados con JBuilder” en la página 8-12](#) para obtener más información sobre las restricciones de diseño.

## Diseño de las zonas de mayor tamaño en primer lugar

---

Es recomendable empezar por el diseño de las zonas grandes de la interfaz de usuario y configurar sobre la marcha los detalles de estas zonas, utilizando únicamente los diseños `XYLayout` o `null`. Una vez que el diseño sea de su agrado, trabaje sistemáticamente desde las áreas interiores hacia el exterior, eligiendo diseños más portables para los paneles, por ejemplo, `FlowLayout`, `BorderLayout` o `GridLayout`, realizando pequeños ajustes si es necesario.

Normalmente comenzará por incorporar un contenedor al diseño y luego le añadirá componentes. No obstante, puede dibujar un nuevo contenedor alrededor de componentes existentes, aunque estos componentes no se anidarán automáticamente en el nuevo panel. Después de dibujar el contenedor, debe trasladar explícitamente cada componente al contenedor. Incluso quizá deba desplazarlo fuera del contenedor y volverlo al interior. Controle el árbol de componentes con el objeto de verificar que se anida apropiadamente. Todos los componentes de un contenedor aparecen sangrados bajo él en el árbol de componentes. Si el componente está en el mismo nivel de sangrado que un panel, quiere decir que no se encuentra dentro de este.

## Almacenamiento del trabajo antes de realizar experimentos

---

Las primeras veces que realice diseños en JBuilder trabajará inevitablemente por ensayo y error, especialmente cuando comience a utilizar diseños distintos de `XYLayout` o `null`. Asegúrese de guardar su archivo antes de experimentar con un cambio de diseño. De esta forma, si no funciona, puede

volver atrás. Si tiene control de versiones, utilícelo correctamente, especialmente con los gestores de diseño desconocidos.

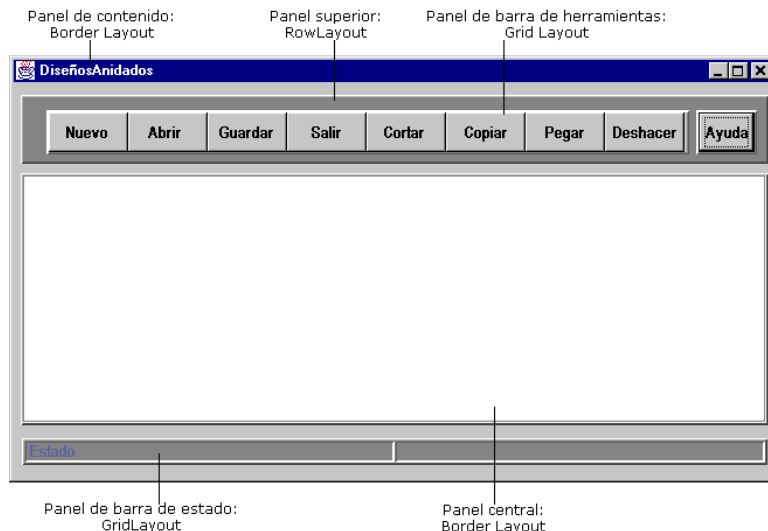
Aunque planifique previamente la interfaz de usuario, es posible que vea que el diseño no funciona como esperaba. Esto puede significar volver a trabajar con el diseño y utilizar una configuración diferente de contenedores, componentes y diseños. Por ello, es aconsejable que copie el archivo contenedor (por ejemplo `Marco1.java`) con un nombre y una ubicación distintos, durante los momentos críticos del proceso de diseño, para no tener que empezar de nuevo.

Algo que acelerará el trabajo de diseño de la interfaz de usuario en el futuro es crear componentes JavaBeans independientes, como por ejemplo barras de herramientas, barras de estado, grupos de casillas de selección o cuadros de diálogo que pueda añadir a la paleta de componentes y reutilizar casi sin modificaciones.

## Paneles y diseños anidados

La mayoría de los diseños de interfaz de usuario en Java utilizan varios tipos de diseños para obtener los resultados deseados, anidando varios paneles con diseños diferentes en el contenedor principal. También es posible anidar unos paneles dentro de otros y obtener más control sobre la ubicación de los componentes. Creando un diseño compuesto y utilizando el gestor de diseño más adecuado para cada panel, es posible agrupar y ordenar los componentes de forma funcional y portable.

Por ejemplo, el siguiente ejemplo de interfaz de usuario demuestra la utilización de paneles anidados con diseños diferentes. Utiliza `BorderLayout`, `FlowLayout` y `GridLayout`. Toda la interfaz está en `contentPane` utilizando `BorderLayout`:



### Consulte

- [Capítulo 11, “Tutorial: Creación de una interfaz de usuario con diseños anidados”](#) si desea seguir el tutorial que desarrolla esta interfaz de usuario
- [Capítulo 5, “Creación de interfaces de usuario”](#) si desea más información sobre el diseñador

## Tutorial: Creación de una interfaz de usuario sencilla

Este es un tutorial muy sencillo diseñado para introducirle rápidamente al diseñador. Le llevará diez o quince minutos completarlo.

Con un nuevo proyecto (Archivo|Nuevo proyecto):

- 1 Cree una aplicación: pulse Archivo|Nuevo, seleccione la ficha General y haga doble clic sobre Aplicación.
- 2 Pulse Siguiente en el Asistente para aplicaciones y asegúrese de que las primeras cuatro opciones *no están activadas*. (No necesita estos elementos de la interfaz para esta aplicación.)
- 3 Pulse Finalizar en el Asistente para aplicaciones.

Ahora tiene una aplicación con dos archivos JAVA:

- `Application.java`, con el método `main()` que funciona como el punto de entrada del programa.
- `Frame1.java`, donde se encuentra el código de la interfaz de usuario.

Ya está listo para comenzar.

## Creación de la interfaz de usuario

---

- 1 Haga doble clic sobre `Framel.java` en el panel de proyecto para abrirlo en el editor.
- 2 Haga clic sobre la pestaña Diseño de la parte inferior del panel de contenido. Se abre el diseñador.

Observe que solo un componente está en `this`: el `contentPane`. El diseño por defecto de `contentPane` es `BorderLayout`, lo que significa que los componentes que se colocan en este contenedor se adherirán a los márgenes norte, sur, este u oeste o se colocarán en el centro. Es recomendable diseñar la interfaz IU antes de determinar qué diseño se va a utilizar, por lo que se debe cambiar la propiedad de diseño a una que permita ser manipulada libremente.

- 1 Seleccione `contentPane` en el árbol de contenido (que aparece en el panel de estructura).
- 2 Observe el Inspector, a la derecha de la superficie de diseño. Observe la propiedad `layout`.

### Sugerencia

Las propiedades y los sucesos se enumeran alfabéticamente en sus tablas dentro del Inspector.

- 3 Haga clic sobre la columna derecha, llamada `BorderLayout` para la propiedad de diseño.

Aparece una lista de gestores de diseño.

- 4 Seleccione `null`.

“Null” no es un gestor de diseño, es un sustituto del gestor de diseño que no impone ningún comportamiento de diseño propio. Cuando coloca un componente en un contenedor con gestión de diseño null, el componente se ubica exactamente donde lo ha colocado. Puede cambiar su tamaño y moverlo en la superficie de diseño al centro de su contenido. Cuando el diseño tenga la apariencia correcta, cambie el diseño por un gestor de diseño que ofrezca respuesta. De esta forma, puede cambiarse el tamaño de la interfaz dinámicamente durante la ejecución y trasladarse a cualquier plataforma.

A veces, es necesario hacer algunos arreglos en el diseño después de aplicar un gestor de diseño. Eso es de esperar.

Mientras tanto, añada un componente al diseño null `contentPane`:

- 1 Observe la paleta de componentes en la parte superior de la superficie de diseño. Haga clic sobre la pestaña Swing para ver los componentes Swing.
- 2 Pulse el icono de `JButton`. Esto lo carga en su cursor.  
Pulse `contentPane` en la superficie de diseño donde desee que esté el ángulo superior izquierdo del botón. El botón se coloca en el `contentPane`.
- 3 Como está utilizando un diseño null, puede mover el botón donde desee. Colóquelo en el ángulo inferior derecho, no muy cerca de los bordes.

`jButton1` es un nombre sin significado. Cámbielo a uno más representativo:

- 1 Haga clic con el botón derecho del ratón en `jButton1` en el árbol de componentes y seleccione Cambiar nombre.
- 2 Cambie su nombre por `pushButton` y pulse *Intro*.

Este comando cambia el nombre `jButton1` por `pushButton` en todo lugar en que se hace referencia a este botón JButton en particular.

## Los diseños

---

Ahora se aplicará un diseño portable:

- 1 Seleccione `contentPane` en el árbol de componentes.

De esta forma el Inspector muestra todas las propiedades del supercontenedor del botón.

- 2 Observe el Inspector. Pulse sobre el valor `null` junto a `diseño`.

Los otros diseños estarán disponibles.

Observe dónde está el botón. La mayoría de los diseños tendrán problemas con esto puesto que es el único componente en una posición no central del contenedor. No obstante, `GridBagLayout` puede manejarlo. (Con interfaces más complejas, es más difícil controlar este gestor de diseño, pero es lo suficientemente poderoso y flexible para que valga la pena el esfuerzo.)

- 1 Seleccione `GridBagLayout` de la lista de diseños.
- 2 Ejecute la aplicación.

Si lo desea, puede jugar con la nueva aplicación. Observe que, cuando reduce su tamaño, el botón no permanece dentro de la parte visible del contenedor. Esto se debe a que el Editor de restricciones de JBuilder para `GridBagLayout` ha seleccionado las restricciones que se acercan más a su diseño base. No obstante, estas restricciones son demasiado estrictas. Las restricciones deben utilizarse sólo para ajustar un diseño.

## Ajuste de restricciones

---

- 1 En el Inspector, elija la propiedad `ORBConnect`. Haga clic sobre la celda que enumera las restricciones. Aparece un botón pequeño de puntos suspensivos (...) en el borde derecho de la celda.
- 2 Pulse el botón de puntos suspensivos (...).  
Aparece el editor de `GridBagConstraints`.
- 3 Observe los valores que están completos y sus valores altos. El Editor de restricciones estaba intentando traducir el diseño `null` de la forma más precisa. Es preferible que sea más flexible.

- 4 Comience con el área de ancla. Actualmente está seleccionado el centro. Seleccione SE.
- 5 Luego, observe los encuadres. Estos determinan el espacio que se encuentra alrededor de un componente.  
Establezca los valores de Superior e Izquierda en 0 (cero). Esto se debe a que, como el componente está anclado en Sudeste, no es necesario que se suba o baje más.
- 6 Establezca los valores de Inferior y Derecha en 30. Esto ofrece una cantidad de relleno adecuada, manteniendo al botón a una distancia correcta del ángulo inferior derecho.
- 7 Pulse Aplicar en el editor. Esto muestra cómo se verá en el diseñador.
- 8 Cuando haya finalizado con el diseño, pulse Aceptar en el Editor de restricciones.
- 9 Ejecute su aplicación y cambie su tamaño. El botón permanece a la misma distancia del ángulo inferior derecho y se mueve cuando se cambia el tamaño del contenedor.

## Vincular sucesos

---

Luego, se añadirá una acción al botón.

- 1 En el árbol de componentes, seleccione `jButton1`.

Normalmente se seleccionan componentes en el árbol de componentes para evitar toda confusión a medida que la interfaz se hace más compleja. También se pueden seleccionar pulsando sobre ellos en la superficie de diseño.

### Sugerencia

Cuando se encuentre en la superficie de diseño, observe la barra de estado en la parte inferior izquierda para determinar exactamente sobre qué componente se encuentra el cursor.

- 2 En el Inspector, elija la propiedad `ORBConnect`. Está escrito `jButton1`. Cámbielo a `Pinche aquí`.

- 3 En el Inspector, seleccione la pestaña Sucesos.

Esto da acceso a todos los sucesos disponibles para ese componente.

- 4 Busque el suceso `actionPerformed()` en la parte superior o cerca de esta. Haga doble clic sobre la celda de la derecha próxima al suceso.

Se realizan dos acciones: se crea un stub del método en el código de `Frame1.java` y se cambia al editor con el cursor dentro del stub, donde se inserta el código que definirá la acción.

- 5 Escriba lo siguiente dentro del stub:

```
jButton.setText(";Ha pinchado aquí!");
```



El código presenta el siguiente aspecto:

```
void jButton1_actionPerformed(ActionEvent e) {  
    jButton.setText(";Ha pinchado aquí!");  
}
```

**6** Ejecute el programa y pulse el botón. Compruebe que funciona.

Ahora ya sabe cómo funcionan las principales partes del diseñador. Además, ya conoce las estrategias elementales útiles para el diseñador y el IDE: usar la galería de objetos para acceder a los asistentes, usar los atajos de teclado y los menús contextuales para acceder rápidamente a los comandos, usar el ratón en el Inspector y usar los distintos paneles en el IDE de una forma coordinada.

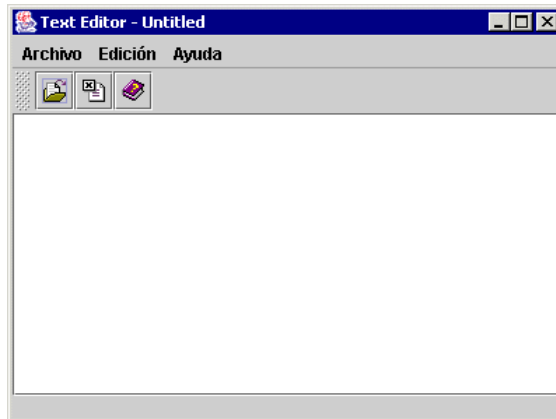
Puede consultar otros tutoriales sobre cómo asignar sucesos ([Capítulo 10](#), “[Tutorial: Creación de un editor de texto en Java](#)”), utilizar diseños ([Capítulo 11](#), “[Tutorial: Creación de una interfaz de usuario con diseños anidados](#)”) y utilizar GridBagLayout ([Capítulo 12](#), “[Tutorial: Creación de diseños GridBagLayout en JBuilder](#)”).



# Capítulo 10

## Tutorial: Creación de un editor de texto en Java

En este detallado tutorial se utiliza JBuilder para crear, comprobar y ejecutar una aplicación en Java llamada “Editor de texto”. Esta aplicación es un simple editor de textos capaz de leer, escribir y modificar archivos de texto.



Podrá definir el color del texto, así como el color de fondo del área de edición de texto.

Apenas invertirá dos horas en completar este tutorial.

## Aspectos tratados en este tutorial

---

Algunos de los pasos de este tutorial son propios de las ediciones JBuilder Developer y Enterprise. Esto se indica al principio de dichos pasos.

El tutorial del editor de texto crea el proyecto y el conjunto de archivos diseñables visualmente mediante el Asistente para proyectos y el Asistente para aplicaciones. A continuación, el tutorial muestra cómo utilizar las herramientas de diseño visual, modificar el diseño de la interfaz de usuario, enlazar sucesos y modificar el código fuente. El tutorial le informa paso a paso sobre la forma de tratar los sucesos de los componentes y tareas más comunes, como los elementos de menú, la barra de botones, el área de texto y los sucesos del sistema. Contiene ejemplos que muestran cómo realizar las siguientes tareas:

- Utilización del cuadro de diálogo `JFileChooser` para poder seleccionar un archivo.
- Lectura y escritura de un archivo de texto y aprovechamiento de su contenido en un `JTextArea`.
- Configuración de los colores de texto y de fondo.
- Definición de la fuente mediante el cuadro de diálogo `dbSwing` Selector de fuente.
- Presentación de información en una barra de estado y en el título de una ventana.
- Incorporación manual de código para tratar sucesos de interfaz de usuario.
- Inserción de un elemento de menú y un botón que ejecuten el mismo código, colocando este último en un nuevo método *auxiliar* al que se llama desde ambos manejadores de sucesos.
- Adición de un menú contextual para el componente `JTextArea`.
- Seguimiento del nombre del archivo actual y de las modificaciones experimentadas desde la última vez que se guardó. Aprendizaje de cómo llevar a cabo esta tarea mediante `ArchivoNuevo`, `ArchivoAbrir`, `ArchivoGuardar`, `ArchivoGuardar como` y de cómo modificar los archivos y salir de ellos.
- Distribución de la aplicación “Editor de texto” a un archivo JAR. Es una función de JBuilder Developer y Enterprise.

Este tutorial contiene el código y el texto que se van a añadir. Si sigue el tutorial en pantalla, puede copiar y pegar el código y los bloques de texto en los campos correspondientes.

### Importante

Si utiliza un sistema basado en UNIX y ha instalado JBuilder como usuario “root”, pero lo está ejecutando como usuario normal, copie el árbol `Samples` a un directorio para el que posea permisos completos de lectura y escritura.

Este tutorial supone que usted está familiarizado con Java y con el IDE (Entorno integrado de desarrollo) de JBuilder. Si desea obtener más información acerca de Java, consulte *Procedimientos iniciales con Java*. Para obtener más información sobre el IDE de JBuilder IDE, consulte “El entorno de JBuilder” en *Creación de aplicaciones con JBuilder*.

Si desea obtener información útil para ver e imprimir los tutoriales, consulte el apartado Tutoriales en Sugerencias de JBuilder. El apartado de opciones de accesibilidad en las Sugerencias de JBuilder contiene sugerencias sobre la utilización de las funciones de JBuilder para mejorar la facilidad de uso de JBuilder por parte de personas con discapacidades.

Para obtener información sobre las convenciones utilizadas en este tutorial y en otra documentación de JBuilder, consulte [“Convenciones de la documentación” en la página 1-5](#).

## Código de ejemplo de este tutorial

---

Para ver el código fuente completo del ejemplo del editor de texto, abra el proyecto de ejemplo:

### En Foundation

```
<jbuilder>/samples/swing/SimpleTextEditor/  
SimpleTextEditor.jpx
```

### En Developer y Enterprise

o en el código FontChooser

```
<jbuilder>/samples/TextEditor/TextEditor.jpx
```

El proyecto `SimpleTextEditor` no incluye el código de distribución ni el de definición del color de fondo. El proyecto `TextEditor` sí los contiene.

### Consulte

- [Capítulo 1, “Diseño visual en JBuilder”](#)
- “El Visualizador de aplicaciones” en *Introducción a JBuilder*
- “Creación de programas en Java” en *Creación de aplicaciones con JBuilder*
- “Depuración de programas en Java” en *Creación de aplicaciones con JBuilder*

## Paso 1: Configuración

---

En este tutorial se va a crear un editor de texto que permite crear, modificar y guardar archivos.

La funcionalidad para crear archivos se añade después de crear otras funciones. En primer lugar vamos a encargarnos de los métodos de apertura y modificación de archivos. Para ello se necesita un archivo con el que trabajar.

- 1 Con el administrador de archivos, cree un archivo de texto llamado `tester.txt`.

Asegúrese de que tiene acceso completo de lectura y escritura y de que este archivo se puede modificar a voluntad sin que afecte a ningún trabajo.

- 2 Escriba texto en él. Puede escribir en el texto que aparece abajo:

Texto para utilizar.

El texto que se extienda más de una línea para comprobar esa línea, engloba correctamente los trabajos y los muestra como debería.

### 3 Guarde el archivo.

A continuación, cree un proyecto y los archivos necesarios para la creación de la interfaz de usuario del editor de texto. Se va a utilizar el Asistente para proyectos con el objeto de crear el proyecto y configurar algunos de sus parámetros y, a continuación, se utilizará el Asistente para aplicaciones con el fin de crear los archivos.

## Creación de un proyecto

---

El Asistente para proyectos crea un proyecto de JBuilder en el que trabajar.

1 Elija Archivo|Nuevo proyecto para iniciar el Asistente para proyectos.

2 Realice los siguientes cambios en el Paso 1:

- **Nombre:** Editor de texto.

**Nota**

Por defecto, JBuilder utiliza este nombre de proyecto como el nombre del directorio del proyecto y el nombre del paquete de las clases contenidas.

- Seleccione la opción Generar archivo de notas del proyecto.

Cuando se activa esta opción, el Asistente para proyectos crea un archivo HTML para las notas del proyecto y lo añade.

- Si tiene otros proyectos abiertos, desactive la opción Añadir proyecto al grupo activo. Este proyecto es independiente.

3 Acepte todas las demás opciones por defecto del Paso 1.

4 Pulse Siguiente para ir al Paso 2 del Asistente para proyectos.

5 Acepte las vías de acceso por defecto del Paso 2.

6 Pulse Siguiente para ir al Paso 3 del Asistente para proyectos.

7 Rellene los campos optativos de la clase Javadoc.

**a** En el campo Título, escriba Tutorial del editor de texto.

**b** En el campo Descripción, escriba Tutorial que muestra las funciones de diseño visual de JBuilder.

**c** Escriba su nombre en el campo @author.

Si lo desea, deje en blanco los otros campos.

Esta información se guarda en el archivo HTML del proyecto. Se utiliza en los comentarios de Javadoc cuando selecciona la opción Crear comentarios de cabecera, presente en muchos de los asistentes de JBuilder, como los asistentes para aplicaciones y clases.

8 Acepte los restantes valores por defecto de esta ficha.

**9** Pulse Finalizar para crear el proyecto.

Un archivo de proyecto y un archivo de proyecto HTML se añaden al proyecto y sus nodos aparecen en el panel de proyecto.

**Consulte**

- “Gestión de las vías de acceso” en *Creación de aplicaciones con JBuilder*
- “Creación y gestión de proyectos” en *Creación de aplicaciones con JBuilder*

## Selección de las opciones de estilo de código del proyecto

---

Ahora se van a configurar las opciones de estilo de código. Estas opciones controlan la forma en que JBuilder escribe los stubs de tratamiento de sucesos y el código de generación de instancias. El tratamiento de sucesos y la instanciación se tratan más a fondo en una etapa posterior de este tutorial.

Para cambiar las opciones de estilo de escritura de código:

- 1** Haga clic con el botón derecho sobre `TextEditor.jpx` en el panel de proyecto (superior izquierda).
  - 2** Elija Propiedades en el menú que aparece.  
Aparece el cuadro de diálogo Propiedades de proyecto.
  - 3** Seleccione Formateo/Generado en el cuadro de diálogo Propiedades de proyecto.  
Aquí se elige el estilo de los manejadores de sucesos que deben generarse. JBuilder puede emplear clases internas anónimas o clases adaptador independientes. En este tutorial, se utiliza el segundo tipo.
  - 4** Consulte Tratamiento de sucesos.
  - 5** Seleccione la opción Adaptador estándar.
- Nota** Con independencia del estilo de manejadores de sucesos empleado, el código que escriba en el método será el mismo.
- 6** Desactive la opción Seguir código existente. De esta forma, el código del tutorial resulta algo más previsible.
  - 7** Acepte el valor de Paquete por defecto para Accesibilidad de las variables de instancia.  
JBuilder le ofrece la opción de instanciar objetos mediante `Beans.instantiate()` en lugar de la palabra clave `new`. En este tutorial se emplea `new`.
  - 8** Compruebe que la opción Usar Beans.instantiate(...) *no* se encuentra activada.
  - 9** Pulse Aceptar para cerrar el cuadro de diálogo Propiedades de proyecto.

### Consulte

- “Elección del estilo de manejador de sucesos” en la página 4-6

## Utilización del Asistente para aplicaciones

---

Ahora que el proyecto está creado es necesario llenarlo con archivos diseñables visualmente. Añádanse entonces los archivos de la aplicación al proyecto.

- 1 Abra la galería de objetos seleccionando `Archivo|Nuevo`.
  - 2 Seleccione la pestaña General.
  - 3 Haga doble clic en el icono Aplicación para abrir el Asistente para aplicaciones.
  - 4 Cambie el nombre de clase de la aplicación en el Paso 1:
    - Nombre de clase: `TextEditClass`Acepte el nombre del paquete por defecto.
  - 5 Pulse Siguiente para ir al Paso 2 del Asistente para proyectos.
  - 6 Cambie el nombre y el título de la clase de marco:
    - Clase: `TextEditFrame`
    - Título: `Text Editor`
  - 7 Seleccione todas las opciones del Paso 2. El asistente genera automáticamente el código correspondiente a las opciones seleccionadas.

Fíjese en el significado de cada opción según las va desactivando, para saber qué esperar del código generado.
  - 8 Pulse Siguiente para avanzar al Paso 3, donde JBuilder crea la configuración de ejecución por defecto.

Compruebe que esta opción se encuentra activada y acepte el nombre por defecto de la configuración.

El proyecto es nuevo, por tanto, no tiene configuraciones base disponibles.
  - 9 Pulse el botón Finalizar.

El Asistente para aplicaciones añade al proyecto el archivo `.java` y archivos de imagen.
- Nota** Aparece también un nodo de paquetes de código fuente en el panel de proyecto si está seleccionada la opción Recopilación automática de paquetes fuente en la ficha General del cuadro de diálogo Propiedades de Proyecto (`Proyecto|Propiedades de proyecto`).
- 10 Guarde el proyecto utilizando `Archivo|Guardar proyecto` “`TextEditor.jpx`”.

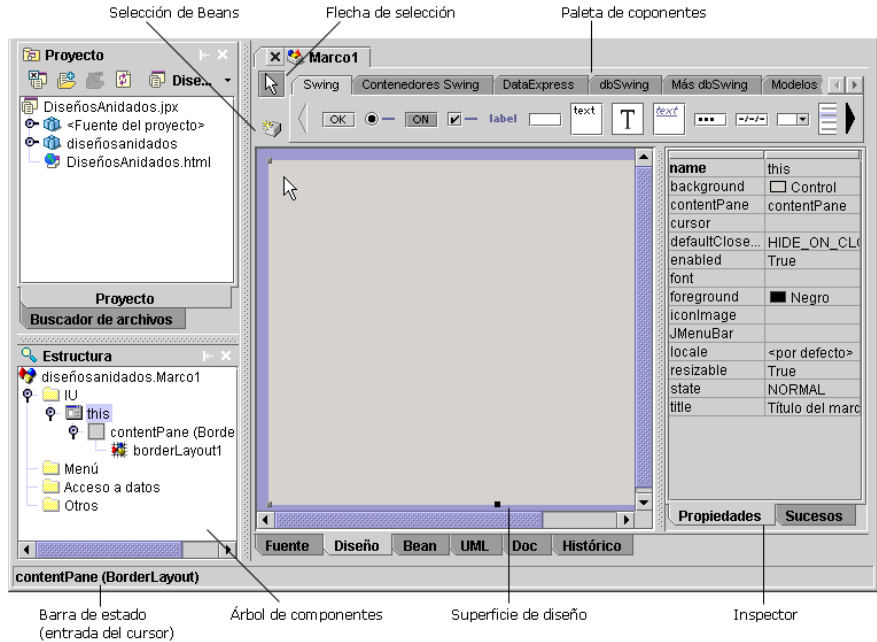
Pulse la pestaña Diseño del archivo abierto, `TextEditFrame.java`. La pestaña Diseño, ubicada en la parte inferior de la ventana del Visualizador de



aplicaciones, abre el diseñador de interfaces de usuario. Observe los cambios en el IDE de JBuilder:

- En el panel de contenido aparece el diseñador de interfaces.
- En el panel de estructura aparece el árbol de componentes, con `this` seleccionado como el componente activo.
- En el panel de contenido aparece la superficie de diseño.
- El Inspector aparece a la derecha de la superficie de diseño.

**Figura 10.1** JBuilder en vista diseño



**Sugerencia** Para ver en qué componente de la superficie de diseño se encuentra el puntero, mire la barra de estado. Esto resulta útil con los diseños de interfaz complejos.

**Sugerencia** Si el área de diseño es demasiado pequeña para mostrar toda la interfaz de usuario en el Visualizador de aplicaciones, cámbielo de tamaño arrastrando los bordes con el ratón o seleccione la barra de división deseada por medio de Ventana|Seleccionar divisor del Visualizador y utilice las teclas de flecha.

## Supresión de la ocultación automática de JFrame

Por defecto, un `JFrame` se oculta cuando se pulsa su cuadro de cierre. Este no es el comportamiento deseado para este tutorial, porque el Asistente para aplicaciones añadió un manejador para llamar a `System.exit(0)` cuando se pulsa el botón de cierre. Más adelante se añadirá código a este manejador

para que pregunte al usuario si guarda el archivo al salir y no es conveniente que la ventana se oculte automáticamente si el usuario dice no.

Para cambiar el comportamiento por defecto:

- 1 Seleccione `this` en el árbol de componentes.
- 2 Haga clic en la pestaña Propiedades del Inspector.
- 3 Seleccione el valor de la propiedad `defaultCloseOperation`, `HIDE_ON_CLOSE`.
- 4 Seleccione `DO_NOTHING_ON_CLOSE` de la lista desplegable de la propiedad.

## Configuración del aspecto

---

Si ha cambiado el aspecto por defecto de JBuilder, entonces configure JBuilder de modo que el diseñador utilice el aspecto Metal. En este tutorial se va a utilizar el aspecto Metal porque es muy parecido en todas las plataformas aceptadas.

Puede configurar el aspecto en el menú contextual del diseñador o en el cuadro de diálogo Opciones del IDE de JBuilder, pero esta elección no influirá en el aspecto de la interfaz de usuario durante la ejecución. Para pasar a un aspecto concreto durante la ejecución, debe configurarse explícitamente en el método `main()` de la clase que ejecuta la aplicación. En este caso, el método `main()` se encuentra en `TextEditClass.java`.

El Asistente para aplicaciones genera por defecto la siguiente línea de código en el método `main()` de la clase ejecutable:

```
UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
```

Esto supone que el aspecto durante la ejecución será cualquiera que esté utilizando el sistema anfitrión.

Para especificar Metal, siga este procedimiento:

- 1 Haga doble clic en `TextEditClass.java` en el panel de proyectos y abra el archivo en el panel fuente.
- 2 Haga clic en `main(String[] args)` en el panel de estructuras, abajo a la izquierda, o desplácese por el panel de contenido hasta encontrar `public static void main(String[] args){`.
- 3 Seleccione la línea de código `setLookAndFeel()` y cópiela en la línea que está inmediatamente debajo.
- 4 Quite el comentario de la primera versión de esta línea de código con dos barras inclinadas:

```
// UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());  
  
UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
```

**5** Cambie el argumento de la nueva versión para asignar el aspecto Metal:

```
UIManager.setLookAndFeel("javax.swing.plaf.metal.MetalLookAndFeel");
```

**6** Seleccione **ArchivarGuardar todo** para guardar el proyecto y sus archivos y prosiga con el siguiente paso.

Resulta conveniente guardar los archivos frecuentemente durante este tutorial, por ejemplo al final de cada paso.

**Consulte**

- [“Cambio del aspecto” en la página 5-10](#)

## Paso 2: Añadir un área de texto

---

En este paso se crea un área de texto que rellena por completo el marco de la interfaz de usuario entre la barra de menús por encima y la barra de estado por debajo. Para lograrlo, el gestor de diseños del contenedor principal de la interfaz de usuario debe utilizar `BorderLayout`.

Un contenedor `BorderLayout` se divide en cinco áreas: North (Norte), South (Sur), East (Este), West (Oeste) y Center (Centro). Cada una de ellas puede albergar un solo componente, por lo que el máximo es cinco en el contenedor. Con este propósito, un panel que contiene múltiples componentes se considera como un único componente. El componente Norte se coloca en la parte superior del contenedor, el componente Este se coloca a la izquierda, etc. Un componente colocado en el área Center ocupa el espacio del contenedor no ocupado por otras áreas que contengan componentes.

**Consulte**

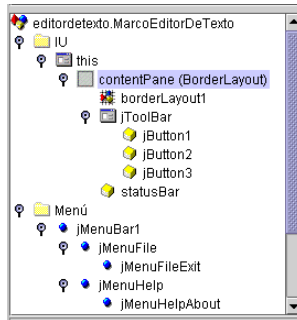
- [“BorderLayout” en la página 8-15.](#)

El Asistente para aplicaciones crea un componente `JMarco` que es el contenedor principal de la interfaz. Este componente `JMarco` es el componente `this`, que contiene un objeto `JPanel` llamado `contentPane`, que ya utiliza `BorderLayout`. Lo único que hay que hacer ahora es añadir los componentes del área de texto a `contentPane`.

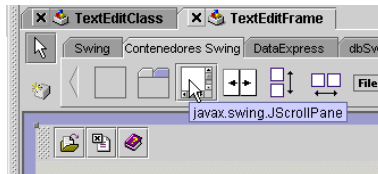
Para ello, se añadirá un panel de desplazamiento y se colocará un componente de área de texto dentro de él. El panel de desplazamiento proporciona barras de desplazamiento al área de texto.

- 1** Seleccione la pestaña `TextEditFrame`, en la parte superior del editor.
- 2** Haga clic en la pestaña **Diseño** si aún no está seleccionada.

- Haga clic en el componente `contentPane` del árbol de componentes para seleccionarlo, como se muestra a continuación.



- Pulse la pestaña Contenedores Swing en la paleta de componentes y seleccione el componente `JScrollPane`.



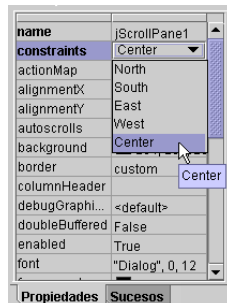
- Haga clic en el centro de `contentPane` del diseñador de interfaces.

De esta forma, el componente `JScrollPane` se coloca en el panel `contentPane` con la restricción `Center`.

En este caso, la barra de herramientas ocupa el área North (superior) y la barra de estado ocupa el South (inferior). Ya que no hay componentes asignados a East y West, el componente panel de desplazamiento (`JScrollPane`) ocupa el área Center y se expande hacia los bordes izquierdo (West) y derecho (East) del contenedor.

Si no lo consigue, seleccione Edición|Deshacer e inténtelo de nuevo.

- Seleccione el nuevo componente `jScrollPane1` en el árbol de componentes.
- Fíjese en el valor de la propiedad `constraints` del Inspector y compruebe que se le está asignando el valor `Center`. Si no es así, seleccione `Center` en la lista desplegable.

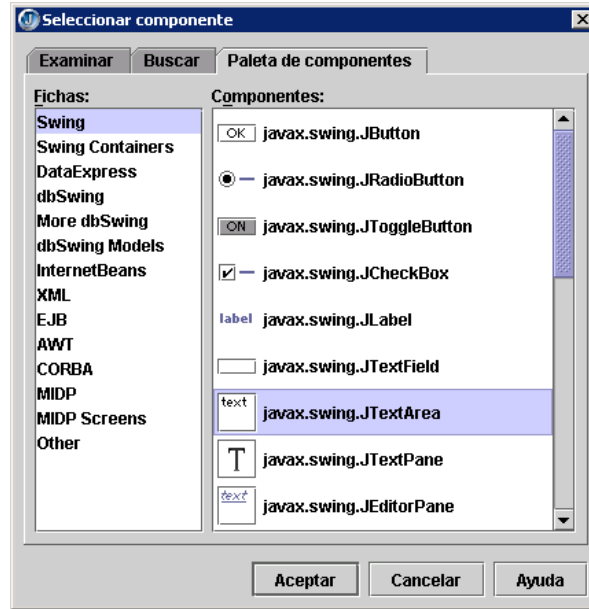


Añada el área de texto, con un modo nuevo de añadir componentes al diseño. Esta vez se va a utilizar el cuadro de diálogo Seleccionar componente.

- 1 Elija Edición|Seleccionar componente.

De esta forma se muestra el cuadro de diálogo Seleccionar componente.

- 2 Elija Swing en el área de fichas de la paleta de componentes.
- 3 Seleccione `javax.swing.JTextArea` en el área de componentes:



- 4 Haga clic en Aceptar para cerrar el cuadro de diálogo.

El nuevo componente se añade a `this`.

- 5 Elija Edición|Cortar con el nuevo componente seleccionado.
- 6 Seleccione `jScrollPane1`.

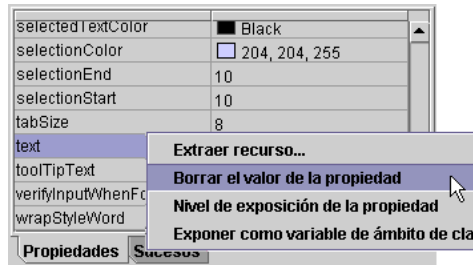
Cuando se vuelve a pegar el componente área de texto, dependerá de este componente.

- 7 Elija Edición|Pegar.

El componente `jTextArea1` aparece debajo de `jScrollPane1` en el árbol de componentes. En la superficie de diseño se muestra anidado dentro de él.

Dentro del panel de texto hay un texto innecesario. Vamos a eliminarlo.

- Haga clic con el botón derecho en la propiedad `text`, en el Inspector, y seleccione **Borrar el valor de la propiedad**.



Por último, debe definir algunas propiedades en `jTextArea1` de manera que ajuste automáticamente las líneas de texto y lo haga en los espacios entre palabras.

**Sugerencia** El Inspector muestra las propiedades por orden alfabético.

En el Inspector, asigne estos valores a las siguientes propiedades:

- `background = white`
- `lineWrap = true`
- `wrapStyleWord = true`

A continuación, compile el programa y ejecútelo para ver qué aspecto ofrece.

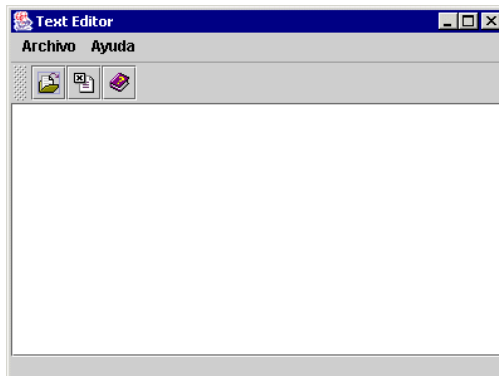
- Seleccione **Proyecto|Ejecutar Make** del proyecto en el menú.

Este comando compila todos los archivos del proyecto. Genera los archivos `TextEditClass.class` y `TextEditFrame.class` en el subdirectorio `classes` del directorio del proyecto. Debería compilarse sin errores.



- Pulse el botón **Ejecutar** en la barra de herramientas de JBuilder, pulse **F9** o seleccione **Ejecutar|Ejecutar proyecto** en la barra de menús.

Ahora la interfaz de usuario de ejecución ofrecerá un aspecto similar al siguiente:



Observe que no hay barras de desplazamiento. Esto se debe a que las propiedades `horizontalScrollBarPolicy` y `verticalScrollBarPolicy` de

`jScrollPane` tienen el valor `AS_NEEDED` por defecto. Si desea que las barras de desplazamiento sean visibles en todo momento, debe cambiar estos valores de propiedad por `ALWAYS`. Se dejarán estas propiedades como están.

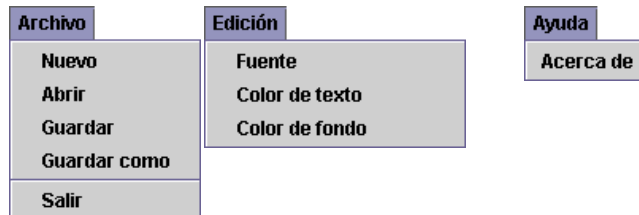
- 1 En la aplicación “Editor de texto”, seleccione Archivo/Salir para cerrar la ventana de ejecución.
- 2 Para cerrar el panel de mensajes, pulse el botón de cierre en la esquina de la pestaña de mensajes:



A continuación se van a crear y rellenar menús utilizables.

## Paso 3: Crear menús

En este paso se van a crear estos menús:



El diseñador de menús se puede utilizar para crear y modificar menús. Se van a crear elementos de menú, añadir un menú e insertar una barra de separación.

Existen varias formas de acceder a estos comandos. En este tutorial se muestran muchas de ellas. Cuando decida cuál es el acceso que prefiere, podrá elegir el modo para utilizarlo en lo sucesivo con los comandos similares.

- 1 Haga clic en la pestaña Diseño de `TextEditFrame.java` si aún no está seleccionada.
- 2 Abra el diseñador de menús. Haga doble clic en `jMenuFileExit` en la carpeta Menú, en el árbol de componentes, o selecciónelo y pulse *Intro*.

De esta forma, la superficie de diseño da paso al diseñador de menús, con `jMenuFileExit` seleccionado.

- 3 Introduzca un elemento de menú por medio de la barra de herramientas del diseñador de menús:



- a Pulse el botón Insertar menú de la barra de herramientas del diseñador de menús.
- b Escriba `Nuevo` directamente en la posición del nuevo elemento de menú.
- c Pulse *Intro* para aceptar el texto escrito.

- 4 Introduzca un elemento de menú por medio del menú contextual del diseñador de menús:
  - a Seleccione el elemento de menú Archivo en la superficie de diseño.  
El menú Archivo se amplía.
  - b Haga clic con el botón derecho del ratón en el elemento de menú Salir.  
De esta forma se muestra un menú con todos los comandos del diseñador.
  - c Elija Insertar elemento de menú en el menú contextual del diseñador de menús.
  - d Seleccione el campo de texto en la ficha Propiedades del Inspector.
  - e Escriba *Abrir*.
  - f Pulse *Intro* para aceptar el texto escrito y bajar a la línea siguiente.
- 5 Inserte otros dos elementos de menú. Cree los siguientes elementos de menú con una de las técnicas explicadas arriba:
  - a Guardar
  - b Guardar como
- 6 Ahora inserte una barra entre los elementos Salir y Guardar como:
  - a Seleccione el elemento de menú Salir.
  - b Pulse el botón Insertar separador.



El menú Archivo ya está terminado. Vamos a crear el menú Edición.

- 1 Haga clic con el botón derecho en la barra principal de menús y seleccione Insertar menú.  
Así se crea un menú entre los menús Archivo y Ayuda.
- 2 Escriba *Edición* como nombre del menú.
- 3 Pulse *Intro* para descender hacia la siguiente entrada vacía. No es necesario pulsar *Insert* aquí porque este menú no contiene ningún elemento después de la entrada actual.

**Nota**

En el diseñador de menús siempre aparece una línea en blanco en la parte inferior de los menús. No es un elemento de menú; solo es un marcador utilizado por JBuilder. Para añadir encima el nuevo elemento se debe utilizar Insertar elemento de menú.

**Sugerencia**

Para borrar una entrada, selecciónela y haga clic en el botón Borrar de la barra de herramientas, o pulse la tecla *Supr* dos veces. La primera vez que se pulsa la tecla *Supr* se borra el texto de la entrada. La segunda vez elimina la entrada del menú.

- 4 Siga creando el menú Edición. Utilice la técnica que prefiera para añadir los elementos. Añada tres elementos de menú:
  - a Fuente



**b** Color del texto**c** Color de fondo

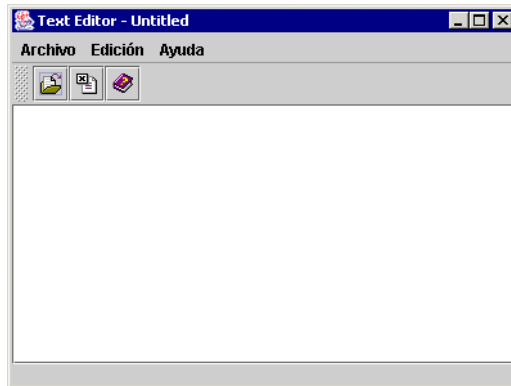
Si alguna entrada tiene una longitud superior al área de edición, el texto se desplazará automáticamente a medida que escriba. Cuando pulse *Intro*, el diseñador de menús ajustará el ancho del menú con el objeto de adaptarse al elemento más largo de la lista.

- 5 Cierre el diseñador de menús con un doble clic en cualquier componente de la carpeta de la interfaz de usuario del árbol de componentes.

Esto hará que el panel de contenido cambie al diseñador de interfaces.

- 6 Guarde el archivo y ejecute la aplicación.

Ahora la interfaz de usuario ofrecerá un aspecto similar al siguiente:



Debería estar en condiciones de experimentar con la interfaz de usuario y escribir texto en el área de texto, pero los botones no funcionarán todavía aunque sí lo harán los menús Archivo|Salir y Ayuda|Acerca de.

Ya se han creado y rellenado los menús requeridos por esta interfaz de usuario. Ahora se les va a añadir la funcionalidad.

**Consulte**

- [“Creación de menús” en la página 6-4](#)

## Paso 4: Añadir un cuadro de diálogo Selector de fuentes

---

Comencemos por enlazar los sucesos de menú, empezando con el elemento de menú Edición|Fuente. Cuando esté hecho, este elemento de menú abrirá el cuadro de diálogo Selector de fuentes.

Ahora se va a añadir un cuadro de diálogo Selector de fuentes a `TextEditFrame.java` para que lo utilice el elemento de menú Fuente:

- 1 Abra `TextEditFrame.java` en el diseñador.
- 2 Elija Edición|Añadir componente.

Aparece el cuadro de diálogo Seleccionar un componente.

- 3 En el campo Fichas, seleccione Más dbSwing. Haga clic en la pestaña de la paleta de componentes si aún no está abierta.



- 4 Seleccione el componente `com.borland.dbswing.FontChooser` del campo Componentes.

- 5 Pulse Aceptar para añadir el selector de fuentes al diseño. Aparece como `fontChooser1` en la carpeta Por defecto del árbol de componentes.

Sólo verá el componente del cuadro de diálogo selector de fuentes en el árbol de componentes, no en el diseñador de interfaces.

## Definición del marco del cuadro de diálogo y las propiedades del título

---

Para que un cuadro de diálogo funcione correctamente durante la ejecución, debe establecerse su propiedad `frame`. La propiedad `frame` debe hacer referencia a un `java.awt.Frame` o descendiente para poder mostrarse. En este caso, el marco que se necesita referenciar es `this` (`MarcoEditorDeTexto`). Si no logra hacer esto, no se verá el cuadro de diálogo y aparecerá un mensaje de error durante la ejecución. También debe asignarse un valor a la propiedad `title` para que el cuadro de diálogo tenga un título.

Para asignar valores a las propiedades `frame` y `title`:

- 1 Seleccione `fontChooser1` en la carpeta Por defecto del árbol de componentes.
- 2 Haga clic en el valor de la propiedad `frame` en el Inspector.
- 3 Seleccione `this` de la lista desplegable de valores.
- 4 Haga clic en el valor de la propiedad `title`.
- 5 Escriba la palabra `Fuente` como valor.
- 6 Pulse *Intro*.

Como resultado de ello, se añaden las líneas siguientes al código fuente en el método `jbInit()`:

```
fontChooser1.setFrame(this);  
fontChooser1.setTitle("Fuente");
```

Al situar el componente `FontChooser` en el árbol de componentes y configurar estas propiedades, en la clase se ha creado código que instancia un cuadro de dialogo `FontChooser`, y asigna el valor "Fuente" a la propiedad `title` y el valor `this` a la propiedad `frame`. Sin embargo, este código no muestra el diálogo, ni lo utiliza de ninguna manera. Antes se debe enlazar el cuadro de diálogo con el elemento de menú. Esto tiene que hacerse por medio del *manejador de sucesos* del elemento de menú `Edición\Fuente`. A continuación crearemos el código necesario.

## Creación de un suceso para lanzar el Selector de fuentes

Creación de un suceso para el elemento de menú Edición|Fuente, que lanzará el Selector de fuentes:

### Sugerencia

- 1 Seleccione el elemento de menú Edición|Fuente en el árbol de componentes. Para ello se escoge Edición|Fuente en la superficie de diseño. En el árbol de componentes, este componente se debe llamar `jMenuItem5` y se debe encontrar en el segundo nodo de menú, `jMenu1`.

Da igual que el componente elemento de menú Fuente tenga un nombre distinto. Pero asegúrese de seleccionar el correspondiente al elemento de menú Fuente. La propiedad `text` de este elemento de menú, en el Inspector, se llama "Fuente".

- 2 Haga clic en la pestaña Sucesos del Inspector.

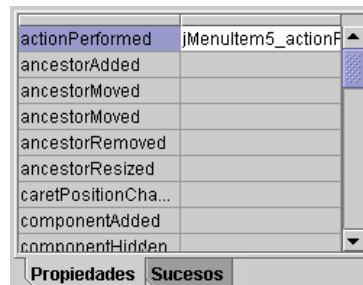
Se enumeran por orden alfabético todos los sucesos que acepta el componente seleccionado.

- 3 Haga clic en el campo de valor (la segunda columna) del suceso `actionPerformed`.

En los menús, botones y otros muchos componentes de la interfaz de usuario de Java, `actionPerformed` es el suceso principal de usuario, que debería *capturar* para responder a la interacción del usuario cuando utiliza el menú o el botón.

El nombre del método de tratamiento de sucesos aparece en el campo de valor. Si el método no existe todavía, esta operación muestra el nombre propuesto por defecto para el nuevo método de gestión del suceso. Para este nuevo manejador de sucesos, el nombre propuesto es

`jMenuItem5_actionPerformed`.



- 4 Haga doble clic en el valor de este suceso o pulse *Intro* para crear el suceso.

Si el método de tratamiento del suceso es nuevo, esta operación generará un stub vacío para el método en el código fuente.

Independientemente de si el método es nuevo o ya existe, el foco de ventana cambiará a código fuente en el editor y colocará el cursor dentro del método de tratamiento de sucesos.

En el caso de un método nuevo de tratamiento de sucesos, como es el caso, verá que la sección principal del método no contiene todavía código alguno.

- 5 Escriba esta línea de código en el cuerpo de este nuevo método vacío (entre las llaves de apertura y cierre):

```
fontChooser1.showDialog();
```

Ahora el método debería parecerse a este:

```
void jMenuItem5_actionPerformed(ActionEvent e) {  
    fontChooser1.showDialog();  
}
```

**Sugerencia**

Para aumentar el área de visión en el panel de contenido, desplace con el ratón los divisores de los bordes o elija Ventana|Seleccionar divisor|Proyecto/Contenido, y desplace el divisor por medio de las teclas de flecha.

- 6 Guarde y ejecute la aplicación. El elemento de menú Edición|Fuente debería abrir el cuadro de diálogo Selector de fuentes. Si no, compruebe que la propiedad `frame` tiene el valor `this`.
- 7 Salga de la aplicación “Editor de texto” y cierre el panel de mensajes de JBuilder.

Aunque intente cambiar la fuente, aún no sucederá nada. Esto se debe a que la aplicación no está utilizando el resultado del `FontChooser` para cambiar el texto del área de edición. Esto será lo siguiente que hagamos.

## Paso 5: Vinculación de sucesos de elemento de menú al cuadro de diálogo Selector de fuentes

---

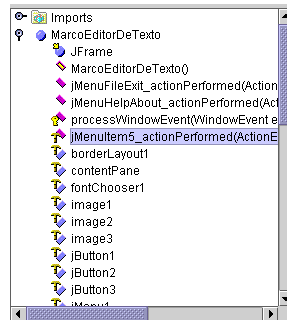
En este paso se asocia el cuadro de diálogo Selector de fuentes al componente de área de texto, de modo que pueda utilizarlo.

- 1 Si aún no lo ha hecho, haga clic en la pestaña Fuente y seleccione el método de tratamiento de sucesos del elemento de menú Fuente (`jMenuItem5_actionPerformed(ActionEvent e)`) recién creado.

**Sugerencia**

Para localizar rápidamente este método en el código fuente, haga clic en el nodo siguiente del panel de estructura, situado en la parte inferior izquierda del Visualizador de aplicaciones. Observe que el orden de los elementos en su panel de estructura quizá no sea exactamente igual que aquí; el orden depende de la configuración de las opciones Orden de estructura de

la ficha Estructura Java del cuadro de diálogo Propiedades de visualización de la estructura.



- 2 Introduzca este código en el método de tratamiento de sucesos para el elemento de menú Fuente (`jMenuItem5`), entre las llaves de apertura y cierre, asegurándose de reemplazar el código antiguo `fontChooser1.showDialog();`:

```
// Gestiona el elemento de menú "Edición Fuente"

// Obtiene la fuente existente en el área de texto y la coloca en
// el Selector de fuentes
// antes de mostrarlo, para que se modifique la
// fuente actual.
fontChooser1.setSelectedFont(jTextArea1.getFont());

// Comprueba el valor devuelto por showDialog() para verificar si el
// usuario ha pulsado Aceptar.
// Obtiene la nueva fuente del Selector de fuentes.
if (fontChooser1.showDialog()) {

    // Asigna a la fuente de jTextArea1 el valor
    // seleccionado por el usuario antes de pulsar Aceptar.
    jTextArea1.setFont(fontChooser1.getSelectedFont());
}
```

Todo el método debería tener ahora el siguiente aspecto:

```
void jMenuItem5_actionPerformed(ActionEvent e) {
    // Gestiona el elemento de menú "Edición Fuente"

    // Obtiene la fuente existente en el área de texto y la coloca
    // en el selector de fuentes antes de mostrarlo,
    // para que se modifique la fuente actual.
    fontChooser1.setSelectedFont(jTextArea1.getFont());

    // Comprueba el valor devuelto por showDialog() para verificar si el
    // usuario
    // ha pulsado Aceptar. Obtiene la nueva fuente del Selector de
    // fuentes.
    if (fontChooser1.showDialog()) {
```

```
// Asigna a la fuente de JTextArea1 el valor
// seleccionado por el usuario antes de pulsar Aceptar.
jTextArea1.setFont(fontChooser1.getSelectedFont());
}
}
```

**Sugerencia**

Para ahorrar tiempo al escribir el código, puede copiar y pegar los ejemplos de arriba, desde el visualizador de ayuda a su código, de la forma siguiente:

- a** Seleccione el código en el Visualizador de la ayuda y cópielo. En este ejemplo, resalte todo el método de tratamiento de sucesos. No olvide comprobar que las llaves de apertura y cierre son correctas en número.
- b** Elija Edición|Copiar en el menú del Visualizador de la ayuda o utilice el atajo de teclas correspondiente a la asignación de teclado.
- c** Haga clic en la pestaña Fuente para abrir el editor en el Visualizador de aplicaciones.
- d** Resalte el código que desea sustituir. En este ejemplo, resalte todo el método en el código fuente.

**Advertencia**

Tenga cuidado al pegar. No elimine ninguna llave importante, como la de cierre en la definición de clases.

- e** Seleccione Edición|Pegar en el menú principal de JBuilder o utilice el correspondiente método abreviado de teclado.
  - f** Compruebe el nivel de sangrado del código insertado y ajústelo para que coincida con su código. Coloque un sangrado en un bloque. Para ello, seleccione el texto y pulse la tecla `Tab`.
- 3** Guarde y ejecute la aplicación y escriba algo en el área de texto.
  - 4** Seleccione el texto y utilice el elemento de menú Edición|Fuente para cambiar la fuente.
  - 5** Salga de la aplicación “Editor de texto” y cierre el panel de mensajes de JBuilder.

En esta aplicación se cambia la fuente de la totalidad del área de texto, no sólo la del texto seleccionado. La configuración de fuente no es permanente, por lo que cuando se cierra la aplicación y se vuelve a abrir se muestra de nuevo el texto por defecto. En este tutorial no se incluye el código que activa estas funciones, pero puede hacerlo después como un ejercicio independiente.

## Paso 6: Vinculación de sucesos de elementos de menú a JColorChooser

---

A continuación se crean los sucesos de menú Edición|Color de texto y Edición|Color de fondo y se los vincula con el cuadro de diálogo JColorChooser de Swing.

Al no necesitar asignar valores a ninguna de las propiedades de `JColorChooser` en esta aplicación, no es preciso añadir el componente a la base de código de la interfaz de usuario. Puede llamarlo directamente desde el manejador del suceso `actionPerformed()` de un elemento de menú del siguiente modo:

- 1 Vuelva al diseñador de menú de `TextEditFrame.java`.
- 2 Seleccione el segundo elemento de menú del árbol de componentes en Edición (`jMenuItem6`) que tiene escrito “Color de texto” en la propiedad `actionCommand`, en la ficha Propiedades del Inspector.
- 3 Seleccione la pestaña Sucesos en el Inspector y haga clic tres veces en el suceso `actionPerformed()` para crear el manejador del suceso:

```
void jMenuItem6_actionPerformed(ActionEvent e) {
}
```

- 4 Añada el código siguiente en el stub del manejador del suceso (incluyendo los comentarios si lo desea):

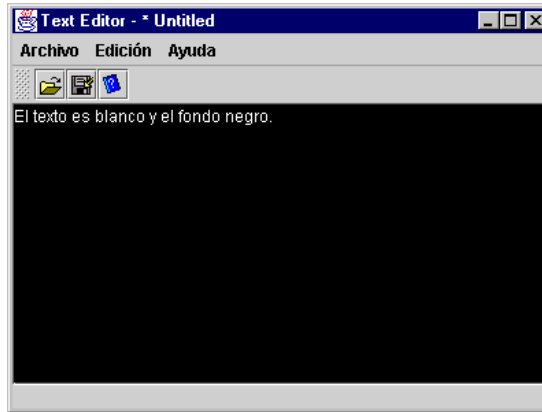
```
//Gestiona el elemento de menú "Color de texto"
Color color = JColorChooser.showDialog(this,"Color de texto",
    jTextArea1.getForeground());
if (color != null) {
    jTextArea1.setForeground(color);
}
```

- 5 Vuelva al diseñador de menús.
- 6 Seleccione el tercer elemento de menú en el árbol de componentes, en Edición (`jMenuItem7`), que debe tener la etiqueta “Color de fondo” en la propiedad `actionCommand`. Cree un suceso `actionPerformed()` para él, tal como hizo con `jMenuItem6`.
- 7 Inserte el siguiente código en el suceso `actionPerformed()` de `jMenuItem7`:

```
// Gestiona el elemento de menú "Color de fondo"
Color color = JColorChooser.showDialog(this,"Color de fondo",
    jTextArea1.getBackground());
if (color != null) {
    jTextArea1.setBackground(color);
}
```

- 8 Guarde el archivo, compile y ejecute la aplicación.

Escriba texto y haga pruebas con los colores de primer plano y de fondo. La aplicación ofrecerá el siguiente aspecto, si elige texto blanco con fondo negro:



- 9 Salga de la aplicación “Editor de texto” y cierre el panel de mensajes de JBuilder.

A continuación se va a asignar una función a `Archivo|Nuevo`.

## Paso 7: Adición de un manejador a un suceso de menú para borrar el área de texto

---

Vamos a enlazar el elemento de menú `Archivo|Nuevo` con un manejador de sucesos que borra el texto antiguo del área de texto cuando se abre un archivo.

- 1 Vuelva al diseñador de menús.
- 2 Seleccione el elemento de menú `Archivo|Nuevo` del árbol de componentes (probablemente `JMenuItem`).
- 3 Cree un suceso `actionPerformed()` de la forma explicada antes.
- 4 Inserte en él el siguiente código:

```
// Gestiona el elemento de menú Archivo|Nuevo.  
// Borra el texto del área del texto.  
jTextArea1.setText("");
```

- 5 Guarde y ejecute la aplicación, escriba algo en el área de texto y vea qué sucede al seleccionar `Archivo|Nuevo`. Debería borrar el contenido.

Observe que no pregunta si desea guardar el archivo antes. Para poder tratar este aspecto, tendrá que configurar la infraestructura para la lectura y escritura de archivos de texto, con el objeto de controlar si el archivo ha cambiado y necesita guardarse, etc. Comenzaremos la utilización de archivos en el paso siguiente.



- 6 Salga de la aplicación “Editor de texto” y cierre el panel de mensajes de JBuilder.

## Paso 8: Añadir un cuadro de diálogo selector de archivos

---

Vamos a enlazar el elemento de menú `Archivo|Abrir` con un manejador de un suceso que presenta al usuario un `JFileChooser` (cuadro de diálogo para abrir archivos) para archivos de texto. Cuando el usuario selecciona un archivo y hace clic en `Aceptar`, el manejador del suceso abre el archivo de texto y coloca su contenido dentro de `JTextArea`.



- 1 Vuelva al diseñador y seleccione el componente `JFileChooser` de la ficha Contenedores Swing de la paleta.
- 2 Haga clic en la carpeta IU del árbol de componentes para colocar el componente en el diseñador de interfaces. (Si hace clic en la superficie de diseño, el componente se colocará en una sección equivocada del árbol.)
- 3 Seleccione el elemento de menú `Archivo|Abrir` en el árbol de componentes (probablemente `jMenuItem2`).
- 4 Cree un suceso `actionPerformed()` e introduzca este código:

```
// Gestiona el elemento de menú Archivo|Abrir.
// Utilizar la versión OPEN del cuadro de diálogo, comprobar el valor
// devuelto de Aceptar/Cancelar
if (JFileChooser.APPROVE_OPTION == jFileChooser1.showOpenDialog(this)) {

    // Muestra el nombre del directorio y archivos abiertos en la barra de
    // estado.
    statusBar.setText("Opened " + jFileChooser1.getSelectedFile().getPath());

    // El código debe ir aquí para cargar realmente el texto
    // en el JTextArea.
}
```

- 5 Guarde la aplicación y ejecútela.
- 6 En el menú `Archivo|Abrir`, seleccione un archivo y pulse `Aceptar`.  
Debe aparecer el nombre del archivo y el directorio completo en la línea de estado en la parte inferior de la ventana. Sin embargo, el área de texto seguirá vacía. Nos ocuparemos de ello en el siguiente paso.
- 7 Cierre la aplicación “Editor de texto” antes de continuar.

## Internacionalización de componentes Swing

---

**Los usuarios de JBuilder Foundation deberán omitir este paso e ir al Paso 9.**

Imagine que vamos a internacionalizar esta aplicación con el fin de que se ejecute en varios idiomas. Esto significa que se debe añadir una línea de código para que los componentes Swing `JFileChooser` y `JColorChooser` aparezcan en el idioma en que se está ejecutando la aplicación.

- 1 **Añada la siguiente línea de código en la clase `MarcoEditorDeTexto` de `MarcoEditorDeTexto.java`:**

```
IntlSwingSupport intlSwingSupport1 = new IntlSwingSupport();
```

**El código presenta el siguiente aspecto:**

```
public class TextEditFrame extends JFrame {  
    IntlSwingSupport intlSwingSupport1 = new IntlSwingSupport();  
    JPanel contentPane;  
    JMenuBar menuBar1 = new JMenuBar();  
    JMenu menuFile = new JMenu();  
    ...  
}
```

**Nota**

En este tutorial, la sentencia de importación `import com.borland.dbswing.*;` se ha añadido automáticamente junto con el componente `dbSwingFontChooser`. En otros casos se puede compilar el archivo y utilizar a continuación *Optimización de importaciones* con el objeto de añadir automáticamente las sentencias de importación necesarias.

Ahora, cuando ejecute la aplicación en otro idioma, el `JFileChooser` y `JColorChooser` aparecerán en el idioma correspondiente.

- 2 Guarde la aplicación.

**Consulte**

Consulte estos temas en *Creación de aplicaciones con JBuilder*

- “Internacionalización de programas con JBuilder”
- “Adición y configuración de bibliotecas”
- “Optimizar importaciones”

## Paso 9: Añadir código para leer texto de un archivo

---

En este paso, vamos a añadir código para leer el texto del archivo seleccionado por el usuario y ponerlo en el `JTextArea`. Para esto es necesario añadir un método a `TextEditFrame.java` y ajustar el manejador de sucesos que lo llama.

En primer lugar, habrá que añadir un método a la clase que se encargará de realizar la operación de apertura del archivo. Este método se llamará `openFile()`.

- 1 Cambie al editor en `TextEditFrame.java`.
- 2 Añada la importación siguiente a la lista de importaciones de la parte superior del archivo:

```
import java.io.*;
```

- 3 Inserte el siguiente método `openFile()`.

**Puede colocar este método en cualquier lugar de la clase. Un buen lugar para ubicarlo es justo después del código del método `jbInit()` y justo antes del suceso `jMenuFileExit_actionPerformed()`.**

```
// Abrir el archivo con nombre; lee el texto del archivo al JTextArea;
informar a la barra de estado.
void openFile(String fileName) {
    try {
        // Abrir un archivo con nombre.
        File file = new File(fileName);

        // Obtener el tamaño del archivo abierto.
        int size = (int)file.length();

        // Asignar cero a un contador para realizar un recuento de
        // los caracteres que se han leído del archivo.
        int chars_read = 0;

        // Crear un lector de entrada basado en el archivo, para leer
        // los datos.
        // FileReader gestiona las conversiones de código de caracteres
        // internacionales.
        FileReader in = new FileReader(file);

        // Crea una matriz de caracteres del tamaño del archivo,
        // para utilizarla como búfer de datos, en el que leer
        // los datos del texto.
        char[] data = new char[size];

        // Leer todos los caracteres disponibles en el búfer.
        while(in.ready()) {
            // Incrementar el recuento de cada carácter leído,
            // y acumularlos en el búfer de datos.
            chars_read += in.read(data, chars_read, size - chars_read);
        }

        in.close();

        // Crear una cadena temporal que contenga los datos,
        // y asignar la cadena a JTextArea.
        JTextArea.setText(new String(data, 0, chars_read));

        // Muestra el nombre del directorio y archivos abiertos en la barra de
        // estado.
        statusBar.setText("Abierto "+fileName);
    }

    catch(IOException e) {
        statusBar.setText("Error al abrir "+fileName);
    }
}
```

- 4 Haga clic en el manejador del suceso de ArchivoAbrir del panel de estructura para localizarlo en el código fuente. Se llamará `jMenuItem2_actionPerformed(ActionEvent)` si el componente elemento de menú ArchivoAbrir se llama `jMenuItem2`.
- 5 Reemplace el código fuente en el manejador del suceso de ArchivoAbrir `if()` que contenía previamente:

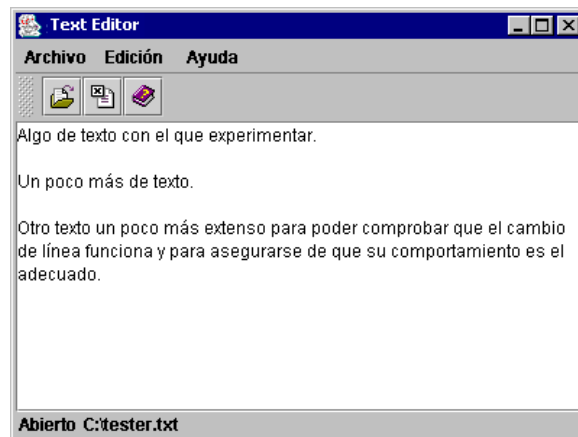
```
// Muestra el nombre del directorio y archivos abiertos en la barra de estado.  
statusBar.setText("Opened "+jFileChooser1.getSelectedFile().getPath());  
  
// El código debe ir aquí para cargar realmente el texto  
// desde el archivo al JTextArea.
```

con este nuevo método `openFile()`, empleando el nombre de directorio y archivo concatenados.

```
// Llamar a openFile para intentar cargar el texto desde el archivo al JTextArea  
openFile(jFileChooser1.getSelectedFile().getPath());  
//pinta el menú de nuevo una vez que el elemento se ha seleccionado  
this.repaint();
```

- 6 Guarde y ejecute el programa y abra `tester.txt` en el editor.

En el editor de texto debe aparecer el archivo de texto correcto:



- 7 Salga de la aplicación "Editor de texto" y cierre el panel de mensajes de JBuilder.

Vamos a seguir asociando elementos de menú a sucesos de menú. A continuación vamos a hacer que la aplicación pueda guardar los cambios realizados en un archivo.

## Paso 10: Adición de código a los elementos de menú para guardar un archivo

---

Se precisa un código que vuelva a grabar el archivo en el disco cuando se seleccione `ArchivoGuardar` y `ArchivoGuardar` como. El programa necesita saber si el archivo que se guarda es nuevo o si se ha modificado un archivo anterior. Cuando un archivo se ha modificado desde la última grabación, se dice que está *sucio*.

Se va a añadir una variable de instancia `String` que almacene el nombre del archivo abierto, además de añadir métodos para comprobar si se trata de un archivo sucio y escribir de nuevo el texto.

- 1 Haga clic en `jFileChooser1` en el panel de estructura. Esto le llevará a la última entrada de la lista de declaraciones de variables de instancia (dado que `jFileChooser1` fue la última declaración realizada).

- 2 Añada las siguientes declaraciones al final de la lista después de `jFileChooser1`:

```
String currFileName = null; //Vía completa y nombre de archivo. null
                             //significa
                             // nuevo/sin título.
boolean dirty = false;      // false significa que el archivo no
                             // se ha modificado.
```

- 3 Haga clic en el método `openFile(String fileName)` del panel de estructura para buscarlo rápidamente en el código fuente. Sitúe el cursor en el método, a continuación de la línea siguiente que lee el archivo en `JTextArea`:

```
jTextArea1.setText(new String(data, 0, chars_read));
```

- 4 Inserte el siguiente código en esta posición:

```
// Almacenar en caché el nombre de archivo abierto actualmente para
// utilizarlo al guardar...
this.currFileName = fileName;
// ...y marcar la sesión de modificación como borrada
this.dirty = false;
```

- 5 Cree un método `saveFile()` al que pueda llamar desde el manejador del suceso de `ArchivoGuardar`. Puede colocarlo justo después del bloque del método `openFile()`. Este método escribe el nombre de archivo en la barra de estado al guardar:

```
// Guardar archivo actual; gestionar los que no tienen nombre de archivo;
// informar a la barra de estado.
boolean saveFile() {

    // Gestionar donde aún no exista nombre de archivo.
    if (currFileName == null) {
        return saveAsFile();
    }

    try {
        // Abrir el archivo del nombre actual.
        File file = new File (currFileName);
```

## Paso 10: Adición de código a los elementos de menú para guardar un archivo

```
// Crear un escritor de salida que escribirá ese archivo.
// FileWriter gestiona las conversiones de códigos de caracteres
internacionales.
FileWriter out = new FileWriter(file);
String text = jTextArea1.getText();
out.write(text);
out.close();
this.dirty = false;

// Muestra el nombre del directorio y archivos abiertos en la barra de
estado.
statusBar.setText("Error al guardar "+currFileName);
return true;
}
catch(IOException e) {
    statusBar.setText("Error al guardar "+currFileName);
}
return false;
}
```

Después de crear el código que aparece arriba, el panel de estructura presenta un mensaje de error: “No se encontró el método `saveAsFile()` en la clase `texteditor.TextEditFrame`”. Ahora nos ocuparemos de esto.

- 6 Cree el siguiente método `saveAsFile()`. Recibe una llamada de `saveFile()` cuando se guarda un archivo. Lo utilizará también el elemento de menú `Archivo|Guardar` como, del que nos ocuparemos más adelante. Añada el código siguiente justo a continuación del bloque del método `saveFile()`:**

```
// Guardar el archivo actual, preguntando al usuario el nuevo nombre de
destino.
// Informar a la barra de estado.
boolean saveAsFile() {
    // Utilizar la versión SAVE del cuadro de diálogo, comprobar el valor
    devuelto de Aceptar/Cancelar
    if (JFileChooser.APPROVE_OPTION == jFileChooser1.showSaveDialog(this)) {
        // Asignar la selección del usuario al nombre de archivo actual,
        // a continuación realizar un saveFile normal
        currFileName = jFileChooser1.getSelectedFile().getPath();
        //pinta el menú de nuevo una vez que el elemento se ha seleccionado
        this.repaint();
        return saveFile();
    }
    else {
        this.repaint();
        return false;
    }
}
```

- 7 Vuelva al diseñador de menús y cree un manejador del suceso `actionPerformed()` para el elemento de menú `Archivo|Guardar` (probablemente `jMenuItem3`). Inserte el siguiente código:**

```
//Gestionar el elemento de menú Archivo|Guardar.
saveFile();
```

- 8 Cree un manejador de sucesos `actionPerformed()` para el elemento de menú `Archivo|Guardar` como (`jMenuItem4`) e introduzca este código:**

```
//Gestionar el elemento de menú Archivo|Guardar como.  
saveAsFile();
```

- 9 Guarde, compile y ejecute el programa.
- 10 Utilice la aplicación para abrir `texter.txt`, modifique el archivo y guarde los cambios. Realice más cambios y guarde `tester.txt` como `tester_1.txt`.
- 11 Salga de la aplicación “Editor de texto” y cierre el panel de mensajes de JBuilder.

## Paso 11: Adición de código para comprobar si se ha modificado un archivo

---

El programa debe controlar si un archivo se ha modificado desde que se creó, abrió o guardó, de forma que pueda preguntar al usuario si debe guardarse antes cerrar el archivo o salir del programa. Para ello, se añadirá una variable booleana denominada `dirty`, a la que ya se hacía referencia en el código anterior

- 1 Haga clic en el siguiente método del manejador del suceso de Archivo|Nuevo en el panel de estructura: `jMenuItem1_actionPerformed(ActionEvent e)`.
- 2 Añada el código siguiente al final de este método para borrar el contenido de las variables `dirty` y `currFileName`. Colóquelo inmediatamente detrás de la línea `jTextArea1.setText("");`, y antes de la llave de cierre.

```
// Borra el nombre de archivo actual y define el archivo como limpio.  
currFileName = null;  
dirty = false;
```

Se utiliza el componente cuadro de diálogo `JOptionPane` para presentar un cuadro de mensaje de confirmación para constatar si el usuario desea guardar un archivo modificado antes de cerrarlo cuando selecciona Archivo|Abrir, Archivo|Nuevo o Archivo|Salir. Este cuadro de diálogo se abre con una llamada a un método de clase en `JOptionPane`, por lo que no es necesario añadir un componente `JOptionPane` al programa.

- 3 Añada el método `okToAbandon()` siguiente al código fuente. Puede situar este método justo a continuación del bloque del método `saveAsFile()`:

```
// Comprobar si el archivo se ha modificado.  
// Si es así, pedir al usuario una decisión guardar/no guardar/cancelar.  
boolean okToAbandon() {  
    int value = JOptionPane.showConfirmDialog(this, "¿Guarda los  
    cambios?",  
                                              "Text Edit", JOptionPane.YES_NO_CANCEL_OPTION) ;  
  
    switch (value) {  
        case JOptionPane.YES_OPTION:  
            // sí, guarde los cambios  
            return saveFile();  
        case JOptionPane.NO_OPTION:
```

## Paso 11: Adición de código para comprobar si se ha modificado un archivo

```
// No; desechar las modificaciones y devolver true sin guardar
return true;
case JOptionPane.CANCEL_OPTION:
default:
    // Cancelar el cuadro de diálogo sin guardar ni cerrar
    return false;
}
}
```

El método no está completo aún, pero se terminará más adelante.

Se llamará a este método cuando el usuario seleccione `ArchivolNuevo`, `ArchivolAbrir` o `ArchivolSalir`. La finalidad del método es comprobar si es necesario guardar el texto. Si el texto fue modificado, este método utiliza un cuadro de diálogo de mensaje *Sí/No/Cancelar* para preguntar al usuario si desea guardar el archivo.

Este método realiza también una llamada a `saveFile()` si el usuario hace clic en el botón *Sí*. Cuando el método finaliza, el valor `booleano` devuelto indica, si es `true`, que puede abandonarse el archivo porque no se ha modificado o porque el usuario ha hecho clic en el botón *Sí* o *No*. Si el valor devuelto es `false`, querrá decir que el usuario ha hecho clic en *Cancelar*. En un paso posterior, se añadirá el código que verifica si el archivo fue modificado.

De momento, este método trata el archivo como modificado en todos los casos, aunque no se hayan realizado cambios en el texto. Más tarde añadirá un método para cambiar la variable `dirty` a `true` cuando el usuario escribe en el área de texto. También añadirá código al principio de `okToAbandon()` para comprobar el valor de la variable `dirty`.

- 4 Sitúe las llamadas a este método `okToAbandon()` en la parte superior de los manejadores de los sucesos de `ArchivolNuevo` y `ArchivolAbrir`, así como en el manejador del suceso de `ArchivolSalir` generado por el asistente. En cada caso, compruebe el valor que devuelve `okToAbandon()` y realice solamente la operación si el valor devuelto es `true`.

### Sugerencia

Para encontrar rápidamente estos manejadores, haga clic en ellos en el panel de estructura. También puede buscar en el panel de estructura cambiando el foco a este panel y escribiendo.

A continuación aparecen los manejadores de los sucesos modificados:

- En el caso de **ArchivolNuevo** coloque una sentencia `if` en el cuerpo del método, de manera que ese código se ejecute solamente si `okToAbandon()` devuelve `true`. El método modificado debe tener el siguiente aspecto:

```
void jMenuItem1_actionPerformed(ActionEvent e) {
    // Gestiona el elemento de menú ArchivolNuevo.
    if (okToAbandon()) {
        // borrar el texto del TextArea
        jTextArea1.setText("");
        // borra el nombre de archivo actual y define el archivo como limpio:
        currFileName = null;
        dirty = false;
    }
}
```



```
}
}
```

- En **Archivo|Abrir**, coloque una sentencia `if` en el método para los casos en que `okToAbandon()` devuelva `true`, y añada código para volver inmediatamente del método si `okToAbandon()` devuelve `false`.

El método modificado debe tener el siguiente aspecto:

```
void jMenuItem2_actionPerformed(ActionEvent e) {
    // Gestiona el elemento de menú Archivo|Abrir.
    if (!okToAbandon()) {
        return;
    }
    // Utilizar la versión OPEN del cuadro de diálogo, comprobar el valor
    devuelto de Aceptar/Cancelar
    if (JFileChooser.APPROVE_OPTION == jFileChooser1.showOpenDialog(this)) {
        // Llamar a openFile para intentar cargar el texto desde el archivo hasta
        TextArea
        openFile(jFileChooser1.getSelectedFile().getPath());
    }
    this.repaint();
}
```

- En el caso de **Archivo|Salir**, escriba una comprobación de `okToAbandon()` antes de la línea de código que sale de la aplicación. El método modificado debe tener el siguiente aspecto:

```
//Realizar Archivo | Salir
public void jMenuItemFileExit_actionPerformed(ActionEvent e) {
    if (okToAbandon()) {
        System.exit(0);
    }
}
```

Ahora, estos métodos de tratamiento de los sucesos de menú realizan su función si `okToAbandon()` devuelve `true`.

- 5 Guarde y ejecute el programa e intente abrir, editar y guardar `tester.txt` y `tester_1.txt`.

Recuerde que `okToAbandon()` aún no está completo. En este momento, actúa como si el archivo estuviese modificado. El resultado es que, de momento, el cuadro de mensaje de confirmación aparece siempre que se utilicen **Archivo|Nuevo**, **Archivo|Abrir** o **Archivo|Salir**, aunque no se hayan producido cambios en el texto. Si el archivo no se ha modificado, pulse **Cancelar** para cerrar el cuadro de diálogo y seguir ejecutando el comando.

- 6 Salga de la aplicación “Editor de texto” y cierre el panel de mensajes de `JBuilder`.

## Paso 12: Activación de los botones de la barra de herramientas

Cuando se generaron los archivos con el Asistente para aplicaciones se activó la opción **Generar barra de herramientas**. Por ello, `JBuilder` generó código para una `JToolBar` y lo rellenó con tres componentes `JButton` que ya presentan iconos. Todo lo que hay por hacer es especificar el texto en la etiqueta de los botones y la ayuda inmediata, y crear un suceso

`actionPerformed()` por botón, desde donde se llamará al método de tratamiento del suceso apropiado.

## Especificación del texto de la ayuda inmediata del botón

---

Para ello:

- 1 Vuelva al diseñador de interfaces de usuario.
- 2 Seleccione `jButton1` en `jToolBar`, en el árbol de componentes.
- 3 Haga clic en la pestaña Propiedades del Inspector.
- 4 Haga clic en la propiedad `toolTipText` para resaltar su entrada.
- 5 Escriba `Abrir archivo` si aún no lo dice y pulse *Intro*.
- 6 Repita este proceso para `jButton2` y `jButton3`, utilizando este texto:
  - Escriba `Guardar archivo` en `jButton2`.
  - Escriba `Acerca de` en `jButton3`.

## Creación de los sucesos de botón

---

Hasta ahora se han creado manejadores de sucesos mediante el Inspector. Vamos a utilizar un método abreviado para crear los sucesos de botón.

Muchos componentes definen un suceso por defecto en sus clases `BeanInfo`. Por ejemplo, un botón define `actionPerformed()` como su suceso por defecto. Para generar rápidamente un manejador de sucesos para el suceso por defecto, haga doble clic sobre el control en la superficie de diseño.

Mediante este método abreviado, cree sucesos para los botones, de este modo:

- 1 Haga doble clic en `jButton1` en la superficie de diseño. Esta acción lo lleva al editor y el cursor se sitúa en el nuevo suceso `jButton1_actionPerformed(ActionEvent e)` para el botón `Abrir`.
- 2 Introduzca este código para llamar al método `fileOpen()`:

```
//Gestionar el botón Abrir de la barra de herramientas
fileOpen();
```
- 3 Cree un suceso `jButton2_actionPerformed(ActionEvent e)` para el `jButton2` y llame a `saveFile()` desde él:

```
//Gestionar el botón Guardar de la barra de herramientas
saveFile();
```
- 4 Cree un suceso `jButton3_actionPerformed(ActionEvent e)` para el `jButton3`, y llame a `helpAbout()` desde él:

```
//Gestionar el botón Acerca de de la barra de herramientas
helpAbout();
```

Observe que el código en los manejadores de los sucesos `jButton1` y `jButton3` realizan llamadas a métodos que todavía no existen: `fileOpen()` y `helpAbout()`. Es necesario crearlos.

## Creación de un método `fileOpen()`

---

El método `fileOpen()` realiza las operaciones que contiene actualmente el método de tratamiento del elemento de menú `ArchivolAbrir`. No obstante, dado que es necesario realizar las mismas operaciones cuando se pulsa el botón `Abrir`, crearemos un método denominado `fileOpen()` de forma que podamos tener sólo una copia de este código y llamarlo desde el menú `ArchivolAbrir` y desde el botón `Abrir`.

- 1 Cree el stub del método `fileOpen`. Puede situar este método encima del método `openFile(String fileName)`. El stub debe parecerse al siguiente:

```
// Gestionar el menú o botón Archivo|Abrir, llamando a
// okToAbandon y openFile en caso necesario.
void fileOpen() {
}
```

- 2 Diríjase al manejador del suceso `ArchivolAbrir`, `jMenuItem2_actionPerformed()`. Seleccione el código comprendido entre el primer comentario y la última llave de cierre, en `jMenuItem2_actionPerformed()`. El código seleccionado debe ser:

```
if (okToAbandon()) {
    return;
}
// Utilizar la versión OPEN del cuadro de diálogo, comprobar el valor
devuelto de Aceptar/Cancelar
if (JFileChooser.APPROVE_OPTION == jFileChooser1.showOpenDialog(this))
{
    // Llamar a openFile para intentar cargar el texto desde el archivo
    hasta TextArea
    openFile(jFileChooser1.getSelectedFile().getPath());
}
this.repaint();
```

- 3 Corte este código para sacarlo del bloque `jMenuItem2_actionPerformed()` y péguelo en el nuevo stub del método `fileOpen()`.

Éste es el aspecto del método `fileOpen()` una vez finalizado:

```
// Gestionar el menú o botón Archivo|Abrir, llamando a okToAbandon y
openFile
// cuando se necesitan
void fileOpen() {
    if (!okToAbandon()) {
        return;
    }

    // Utilizar la versión OPEN del cuadro de diálogo, comprobar el valor
    devuelto de Aceptar/Cancelar
    if (JFileChooser.APPROVE_OPTION == jFileChooser1.showOpenDialog(this)) {
        // Llamar a openFile para intentar cargar el texto desde el archivo
        hasta TextArea
        openFile(jFileChooser1.getSelectedFile().getPath());
    }
}
```

```
    }  
    this.repaint();  
}
```

- 4 Ahora, llame a `fileOpen()` desde el manejador del suceso de elemento de menú `Archivo|Abrir`. Cuando se haya añadido la llamada al stub, el manejador de sucesos presentará el siguiente aspecto:

```
void jMenuItem2_actionPerformed(ActionEvent e) {  
    // Gestiona el elemento de menú Archivo|Abrir.  
    fileOpen();  
}
```

## Creación de un método `helpAbout()`

---

Realice lo mismo con el elemento de menú `Ayuda|Acerca de` y el botón `Acerca de`. Traslade el código que contiene actualmente el manejador del suceso de `Ayuda|Acerca de` a un nuevo método `helpAbout()` y llámelo desde los manejadores de los sucesos del menú y del botón.

- 1 Coloque este stub de método `helpAbout()` en el código, inmediatamente antes del método `fileOpen()`:

```
// Mostrar el cuadro de diálogo Acerca de.  
void helpAbout() {  
}
```

- 2 Corte el siguiente código de `jMenuHelpAbout_actionPerformed()` y péguelo en el nuevo stub del método `helpAbout()`:

```
TextEditFrame_AboutBox dlg = new TextEditFrame_AboutBox(this);  
Dimension dlgSize = dlg.getPreferredSize();  
Dimension frmSize = getSize();  
Point loc = getLocation();  
dlg.setLocation((frmSize.width - dlgSize.width) / 2 + loc.x,  
    (frmSize.height - dlgSize.height) / 2 + loc.y);  
dlg.setModal(true);  
dlg.show();
```

- 3 Introduzca la llamada a `helpAbout()`; en `jMenuHelpAbout_actionPerformed()` de manera que el método quede como sigue:

```
//Realizar Ayuda | Acerca de  
public void jMenuItem2_actionPerformed(ActionEvent e) {  
    helpAbout();  
}
```

- 4 Ahora, guarde y ejecute la aplicación. Pruebe los botones `Abrir`, `Guardar` y `Acerca de`. Compárelos con los elementos de menú `Archivo|Abrir`, `Archivo|Guardar` y `Ayuda|Acerca de`.
- 5 Salga de la aplicación “Editor de texto” y cierre el panel de mensajes de `JBUILDER`.

## Paso 13: Asociación del tratamiento de sucesos con el área de texto

---

Ahora, asocie el tratamiento de sucesos con `JTextArea` de forma que el programa pueda marcar el archivo con el indicador `dirty` cuando el usuario lo modifique.

Para entender lo que se va a hacer, recuerde que Swing está estructurado de forma que la interfaz de usuario sea completamente independiente de los datos que se representan en ella. Los componentes de la interfaz tienen un conjunto de clases propio, y la información que se representa tiene un conjunto de clases que contienen y manejan los datos. Las clases que contienen los datos se denominan *modelos*. Las listas utilizan `ListModel`, las tablas utilizan `TableModel` y los documentos utilizan `Document`.

Vamos a añadir un `DocumentListener` de Swing al modelo `DocumentModel` de `jTextArea`, y se van a buscar sucesos que realicen inserciones, eliminaciones o cambios en el archivo.

- 1 Cambie a modo diseño y seleccione `jTextArea1`.
- 2 Haga clic con el botón derecho del ratón en la propiedad `document`, en la columna de la izquierda del Inspector.
- 3 Elija Exponer como variable de ámbito de clase en el menú contextual.  
Se ha colocado un objeto `document1` en la carpeta Por defecto del árbol de componentes, donde ahora se podrá configurar sus propiedades y sucesos.
- 4 Seleccione `document1` en el árbol y cambie a la pestaña Sucesos del Inspector.
- 5 Cree un suceso `changedUpdate()`. Para ello, haga doble clic en su campo de valor.

Busque en el método `jbInit()` este nuevo `DocumentListener`:

```
document1.addDocumentListener(new TextEditFrame_document1_documentAdapter(this));
```

- 6 Inserte el siguiente código en el stub del suceso `document1_changedUpdate(DocumentEvent e)` que ha creado:  

```
dirty = true;
```
- 7 Vuelva al diseñador, seleccione `document1`, y cree dos sucesos más desde el Inspector para `document1: insertUpdate()` y `removeUpdate()`. Introduzca en estos sucesos la misma línea de código que utilizó para el suceso `changedUpdate()`.

De esta forma, se asegurará de que la escritura de cualquier carácter en el área de texto asignará al indicador `dirty` el valor `true`.

- 8 Añada las tres líneas siguientes en la parte superior del método `okToAbandon()` de forma que compruebe efectivamente el valor de `dirty`.

```
if (!dirty) {  
    return true;  
}
```

El método `okToAbandon()` debe presentar el siguiente aspecto:

```
// Comprobar si el archivo se ha modificado.  
// Si es así, pedir al usuario una decisión guardar/no guardar/cancelar.  
boolean okToAbandon() {  
    if (!dirty) {  
        return true;  
    }  
    int value = JOptionPane.showConfirmDialog(this, "¿Guardar los  
cambios?",  
                                              "Text Edit", JOptionPane.YES_NO_CANCEL_OPTION) ;  
    switch (value) {  
        case JOptionPane.YES_OPTION:  
            // sí, guarde los cambios  
            return saveFile();  
        case JOptionPane.NO_OPTION:  
            // No; desechar las modificaciones y devolver true sin guardar  
            return true;  
        case JOptionPane.CANCEL_OPTION:  
        default:  
            // Cancelar el cuadro de diálogo sin guardar ni cerrar  
            return false;  
    }  
}
```

- 9 Guarde el trabajo, ejecute el programa y compruebe si los estados modificado y no modificado del archivo funcionan correctamente: el cuadro de mensaje Guardar cambios no debe aparecer si se utiliza `ArchivoNuevo`, `ArchivoAbrir` o `ArchivoSalir` en un archivo limpio (no modificado), pero sí cuando se utilicen estos comandos en un archivo sucio (modificado).
- 10 Salga de la aplicación “Editor de texto” y cierre el panel de mensajes de `JBuilder`.

## Paso 14: Adición de un menú contextual al área de texto

---

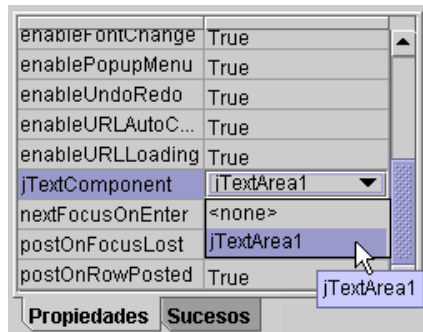
El componente `DBTextDataBinder` añade un *menú contextual* a los componentes de texto Swing para facilitar la realización de tareas sencillas como cortar, copiar o pegar datos del portapapeles. Los menús contextuales son menús a los cuales se accede cuando se hace clic con el botón derecho del ratón en un objeto de la interfaz de usuario, y contienen únicamente comandos relacionados con ese objeto.

`DBTextDataBinder` posee también acciones incorporadas para cargar y guardar archivos en un `JTextArea`, pero que no permite recuperar el nombre del archivo

cargado o guardado. Esta información es necesaria porque esta aplicación muestra el nombre de archivo en la barra de estado. Por ello se va a añadir un `DBTextDataBinder`, que se va a asociar a `jTextArea1`, y se van a suprimir sus acciones de Abrir y Guardar.



- 1 Haga clic en la pestaña Diseño y seleccione el componente `DBTextDataBinder` en la pestaña Modelos dbSwing de la paleta de componentes.
- 2 Colóquelo en cualquier parte del diseñador o en el árbol de componentes. Se situará en la carpeta Acceso a datos del árbol con el nombre `dbTextDataBinder1`.
- 3 Seleccione `dbTextDataBinder1` en el árbol de componentes y abra la lista de valores de su propiedad `jTextComponent` en el Inspector.
- 4 Elija `jTextArea1` en la lista desplegable.



Con este proceso asociará `dbTextDataBinder1` a `jTextArea1` escribiendo la siguiente línea de código en el método `jbInit()`.

```
dbTextDataBinder1.setJTextComponent(jTextArea1);
```

- 5 Seleccione la propiedad `enableFileLoading` de `dbTextDataBinder1` y asígnele el valor `false` mediante la flecha de la lista desplegable.
- 6 Realice la misma operación con la propiedad `enableFileSaving`.
- 7 Guarde su trabajo y ejecute la aplicación.

Observe que ahora tiene un menú contextual cuando hace clic con el botón derecho del ratón en el área de texto. Observe también que no tiene elementos de menú para Abrir y Guardar.

**Nota** Puede añadir los elementos del menú contextual a la barra de menús y de herramientas mediante las clases `Public Static Action` de `DBTextDataBinder`, pero tendría que proporcionar los iconos y escribir el código manualmente. Si necesita un ejemplo de cómo poder hacerlo, consulte el ejemplo `TextPane` en la carpeta de ejemplos de JBuilder: `<jbuilder>/samples/dbswing/TextPane`.

### Consulte

- La documentación de la API relacionada con el componente `DBTextDataBinder`. Para leerlo:
    - a Pulse la pestaña Fuente de `TextEditFrame.java`.
    - b Seleccione `dbTextDataBinder1` en el panel de estructura. Se encuentra dentro del componente `TextEditFrame`.  
La declaración `dbTextDataBinder1` aparece resaltada en el editor.
    - c Coloque el cursor dentro del nombre de la clase `DBTextDataBinder`.
    - d Haga clic con el botón derecho del ratón y seleccione Buscar definición.  
En el editor se abre el archivo de código fuente `DBTextDataBinder`.
    - e Haga clic en la pestaña Doc para ver la documentación de la API.
- Cierre la aplicación “Editor de texto” antes de proseguir con el siguiente paso.

## Paso 15: Visualización del nombre y del estado del archivo en el título de la ventana

---

En este último paso, añada unas líneas de código que utilizan la barra de título de la aplicación para mostrar el nombre de archivo actual y un asterisco si el archivo ha sido modificado.

Para ello, debe crear un método para actualizar la barra de título, que habrá de llamarse desde los lugares en los que el código cambia el nombre de archivo actual o el indicador `dirty`. Este nuevo método se llamará `updateCaption()`.

- 1 Dentro del panel de estructura, haga clic en el método `jMenuFileExit_actionPerformed(ActionEvent e)`. De esta forma traslada el cursor a dicho método de tratamiento de sucesos y lo resalta en el editor.
- 2 Coloque el cursor justo *encima* de este método e introduzca el siguiente bloque de método `updateCaption`:

```
// Actualiza la barra de título de la aplicación para mostrar el nombre
// del archivo y
// su estado de modificación.
void updateCaption() {
    String caption;

    if (currFileName == null) {
        // Sintetizar el nombre "Sin título" si todavía no tiene nombre.
        caption = "Sin título";
    }
    else {
        caption = currFileName;
    }
}
```



```
// Añadir un "*" en el título si el archivo ha sido modificado.
if (dirty) {
    caption = "*" + caption;
}
caption = "Editor de texto - " + caption;
this.setTitle(caption);
}
```

Ahora, coloque la llamada al método `updateCaption()` en cada punto donde cambie el indicador `dirty` y donde se cambie el `currFileName` (nombre actual del archivo). La ubicación de la nueva llamada al método se indica en una línea de comentario, en el bloque de código inferior.

- 1 Coloque la llamada `updateCaption()`; dentro del bloque `try` del constructor `TextEditFrame()`, en la línea inmediatamente siguiente a la llamada a `jbInit()`. El bloque `try` tendrá un aspecto parecido a éste:

```
//Construir el marco
public TextEditFrame() {
    enableEvents(AWTEvent.WINDOW_EVENT_MASK);
    try {
        jbInit();
        updateCaption();    // <---- HERE
    }
    catch(Exception e) {
        e.printStackTrace();
    }
}
```

- 2 Coloque la última línea en el bloque `try` del método `openFile()`, que presentará el siguiente aspecto:

```
try {
    // Abrir un archivo con nombre.
    File file = new File(fileName);

    // Obtener el tamaño del archivo abierto.
    int size = (int)file.length();

    // Asignar cero a un contador para realizar un recuento de
    // los caracteres que se han leído del archivo.
    int chars_read = 0;

    // Crear un lector de entrada basado en el archivo, para leer los
    // datos.
    // FileReader gestiona las conversiones de código de caracteres
    // internacionales.
    FileReader in = new FileReader(file);

    // Crea una matriz de caracteres del tamaño del archivo,
    // para utilizarla como búfer de datos, en el que leer
    // los datos del texto.
    char[] data = new char[size];
```

## Paso 15: Visualización del nombre y del estado del archivo en el título de la ventana

```
// Leer todos los caracteres disponibles en el búfer.
while(in.ready()) {
    // Incrementar el recuento de cada carácter leído,
    // y acumularlos en el búfer de datos.
    chars_read += in.read(data, chars_read, size - chars_read);
}
in.close();

// Crear una cadena temporal que contenga los datos,
// y asignar la cadena a JTextArea.
jTextArea1.setText(new String(data, 0, chars_read));

// Almacenar en caché el nombre de archivo abierto actualmente para
utilizarlo al guardar...
this.currFileName = fileName;
// ...y marcar la sesión de modificación como borrada
this.dirty = false;

// Muestra el nombre del directorio y archivos abiertos en la barra de
estado.
statusBar.setText("Abierto "+fileName);
updateCaption();    // <---- HERE
}
catch (IOException e)
{
    statusBar.setText("Error al abrir "+fileName);
}
```

### 3 Colóquelo justo antes de return true; en un bloque try del método

```
saveFile().

try
{
    // Abrir el archivo del nombre actual.
    File file = new File (currFileName);

    // Crear un escritor de salida que escribirá ese archivo.
    // FileWriter gestiona las conversiones de códigos de caracteres
internacionales.
    FileWriter out = new FileWriter(file);
    String text = jTextArea1.getText();
    out.write(text);
    out.close();
    this.dirty = false;

    // Muestra el nombre del directorio y archivos abiertos en la barra
de estado.
    statusBar.setText("Error al guardar "+currFileName);
    updateCaption();    // <---- HERE
    return true;
}
catch(IOException e) {
```

```
        statusBar.setText("Error al guardar "+currFileName);
    }
    return false;
}
```

**4 Conviértala en la última línea de código en el bloque if del manejador del menú Archivo|Nuevo, jMenuItem1\_actionPerformed():**

```
void jMenuItem1_actionPerformed(ActionEvent e) {
    // Gestiona el elemento de menú Archivo|Nuevo.
    if (okToAbandon()) {
        // borrar el texto del TextArea
        jTextArea1.setText("");
        // borra el nombre de archivo actual y define el archivo como
        limpio:
        currFileName = null;
        dirty = false;
        updateCaption();    // <---- HERE
    }
}
```

**5 Cuando se activa por primera vez el indicador dirty en un archivo, en respuesta a la introducción de texto por parte del usuario. Esto se hace en todos los manejadores del suceso document1. Los manejadores de sucesos se deben cambiar a lectura:**

```
void document1_changedUpdate(DocumentEvent e) {
    if (!dirty) {
        dirty = true;
        updateCaption();    // <---- HERE
    }
}

void document1_insertUpdate(DocumentEvent e) {
    if (!dirty) {
        dirty = true;
        updateCaption();    // <---- HERE
    }
}

void document1_removeUpdate(DocumentEvent e) {
    if (!dirty) {
        dirty = true;
        updateCaption();    // <---- HERE
    }
}
```

**6 Ejecute la aplicación y observe cómo cambia la barra de título cuando se realizan las operaciones siguientes:**

- Cambie el nombre del archivo, utilizando Archivo|Guardar como.
- Escriba en el área de texto, para hacer que el archivo cambie a modificado. Observe que en la barra de título aparecerá el asterisco \* tan pronto como se escriba.

- Guarde el archivo, para anular el valor de modificación.
- Intente llevar a cabo estas acciones mediante el menú contextual.

¡Enhorabuena! Ha utilizado las herramientas de diseño visual de JBuilder para crear un editor de texto que funciona, escrito enteramente en Java.

Ha finalizado el tutorial. Compare su código con el del ejemplo, `<jbuilder>/samples/SimpleTextEditor`.

Diríjense al Paso 16 para distribuir esta aplicación y ejecutarla desde la línea de comandos.

**Usuarios de JBuilder Foundation:**

**Usuarios de las ediciones Developer y Enterprise de JBuilder.**

## Paso 16: Distribución de la aplicación Editor de texto a un archivo JAR

---

**Este paso es aplicable únicamente a JBuilder Developer y Enterprise.**

Ahora que ha creado la aplicación “Editor de texto”, puede distribuir todos los archivos en un Archivo recopilatorio Java (JAR, Java Archive File) utilizando el Creador de recopilatorios de JBuilder.

### **Nota**

Aunque no haya realizado los Pasos 1 al 15 de este tutorial, puede efectuar este paso utilizando el proyecto de ejemplo Editor de texto de la carpeta `samples/TextEditor/` de la instalación de JBuilder. Para ello, debe modificar las vías de acceso especificadas en el tutorial de forma que apunten a `/samples/TextEditor/` y sus subdirectorios.

## Aspectos generales

---

La distribución es un tema avanzado que requiere algo de estudio y experiencia para su comprensión. El Creador de recopilatorios de JBuilder reduce esta complejidad y ayuda a crear un archivo de revisiones con todos los requisitos necesarios para la distribución.

Este paso del tutorial proporciona instrucciones para distribuir la aplicación “Editor de texto” de forma explícita. No pretende ser un ejemplo que cubra todas las situaciones que se pueden presentar cuando se distribuyen programas Java. Cada aplicación o applet que se distribuye presenta sus propios y singulares aspectos de distribución, por lo que resulta difícil generalizar. A lo largo de este paso, se proporcionan enlaces para obtener más información sobre distribución, incluido el correspondiente al tutorial de Java™ de Sun en <http://java.sun.com/docs/books/tutorial/>.

El primer paso para la distribución de cualquier programa es identificar qué archivos, del proyecto y de biblioteca, se van a incluir en el archivo recopilatorio. Esto ayuda a determinar las clases, recursos y dependencias que se deben incluir. Si se incluyen en el archivo de revisiones todas las clases, recursos y dependencias, el archivo resulta excesivamente grande. Sin embargo, la ventaja es que no necesita proporcionar al usuario final ningún otro archivo, ya que el archivo recopilatorio contiene todo lo necesario

para ejecutar el programa. Si se excluyen clases, recursos o dependencias parcial o totalmente, será necesario proporcionarlos por separado al usuario final.

El Creador de recopilatorios no incluirá el JDK en el archivo recopilatorio. Presupone que las clases JDK ya existen en el ordenador de destino en forma de JDK instalado, entorno de ejecución Java o Plug-in Java, o que se proporcionarán en la instalación.

El Creador de recopilatorios de JBuilder crea un nodo de recopilatorios en el panel de proyecto, que facilita el acceso al recopilatorio. El archivo recopilatorio puede crearse o modificarse en cualquier fase del desarrollo. También se puede ver el contenido del archivo recopilatorio y del archivo descriptor.

## Ejecución del Creador de recopilatorios

Para ejecutar el asistente Creador de recopilatorios y crear el nodo de archivo recopilatorio y el archivo recopilatorio del tutorial Editor de texto:

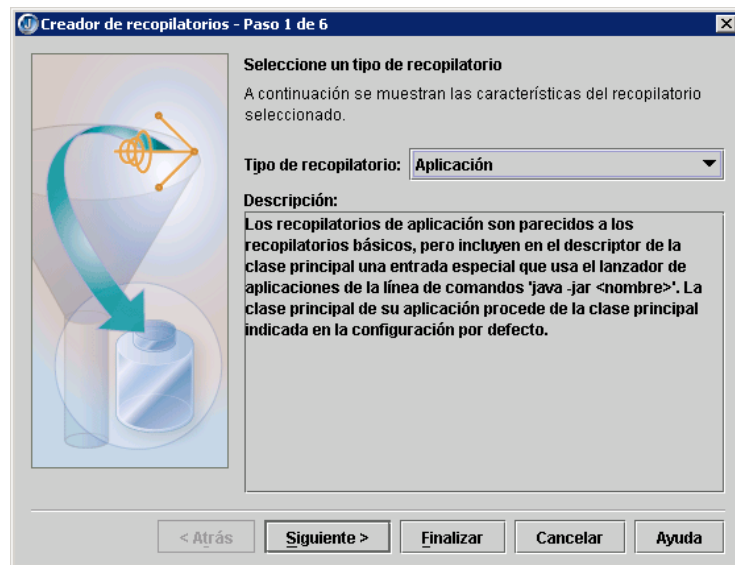
1 Guarde todos los archivos del proyecto y compílelo.

2 Seleccione Asistentes\Creador de recopilatorios.

Se muestra el Paso 1 del Creador de recopilatorios.

3 Seleccione Aplicación como Tipo de archivo recopilatorio.

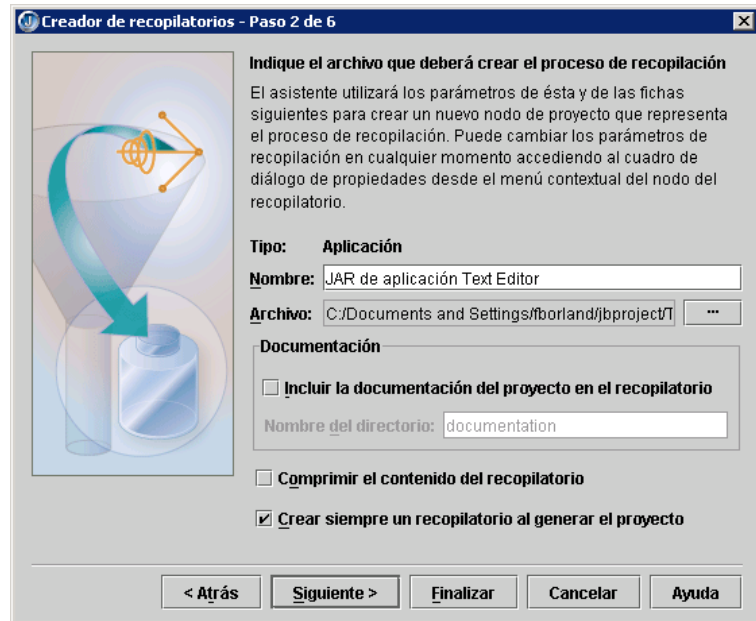
El paso 1 del Asistente tiene el siguiente aspecto:



4 Haga clic en Siguiente para avanzar al Paso 2 del asistente.

- 5 Cambie el nombre del archivo recopilatorio por `TextEditorApplication` en el campo Nombre. Éste es el nombre del nodo de archivo recopilatorio que aparecerá en el panel de proyecto.
- 6 Acepte el nombre y la vía de acceso por defecto del archivo JAR: `<project path>/TextEditor.jar`.
- 7 Acepte el resto de los valores por defecto de esta ficha.

Cuando haya terminado, el Paso 2 del asistente debería tener este aspecto:



- 8 Haga clic en Siguiete para ir al Paso 3 del asistente, en el que se indicarán las clases y recursos del proyecto que se distribuirán.

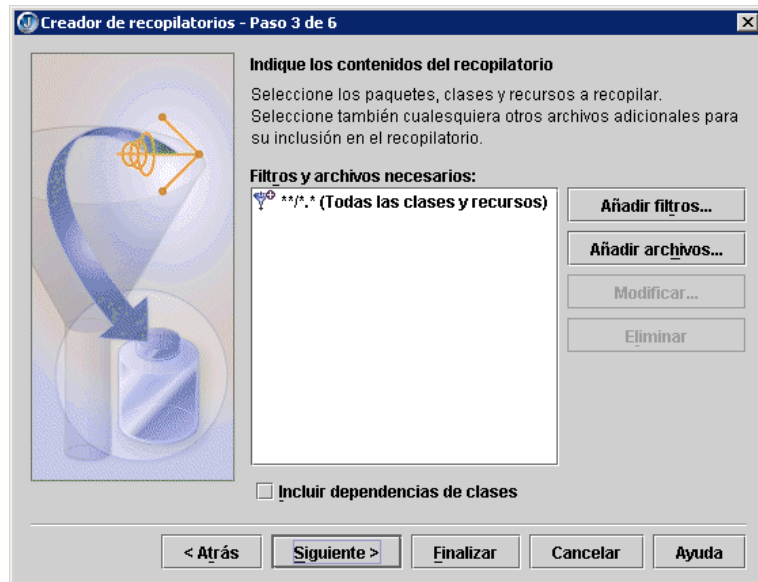
Las clases y recursos del proyecto son los contenidos en la vía de archivos generados, definida en la ficha Vías de acceso del cuadro de diálogo Propiedades de proyecto. Normalmente, suele ser el directorio `classes` del proyecto. En este tutorial, acepte los valores por defecto para que el asistente incluya todas las clases y recursos de la vía de archivos generados.

#### Importante

Aunque esta opción es la más segura y simplifica la distribución, hace que el archivo recopilatorio sea mayor. Si por alguna razón su proyecto incluye archivos innecesarios en la vía de salida, éstos son incluidos también, haciendo muy grande su archivo de distribución. En este caso puede que le interese aplicar filtros, con lo que se excluirían archivos y clases innecesarias, o se añadirían clases y archivos. Use los botones Añadir filtros y Añadir archivos para formar el archivo recopilatorio. Use los botones Modificar y Eliminar para cambiar o eliminar los filtros existentes y

los archivos incluidos. A continuación pruebe la aplicación distribuida para asegurarse de que ha incluido todos los archivos necesarios.

El Paso 3 del asistente tendrá este aspecto:



### Consulte

- “Añadir filtros” en *Creación de aplicaciones con JBuilder* para aprender a utilizar los filtros de esta ficha.

### 9 Haga clic en Siguiente para avanzar al Paso 4 del asistente.

En este paso, se selecciona cómo se incluye el contenido de las bibliotecas en el archivo recopilatorio. Habitualmente, las bibliotecas no se incluyen en el archivo recopilatorio, sino que se suministran como archivos JAR separados y se incluyen en la `CLASSPATH` en la ejecución. Ésta es la manera más fácil de efectuar una distribución y la que crea el archivo JAR más pequeño. No obstante, en este ejemplo, incluirá las bibliotecas en el recopilatorio.

#### Nota

Los contenidos del recopilatorio se establecen de forma separada para cada elemento de la lista.

- Seleccione dbSwing en la lista.
- Active la opción Incluir clases necesarias y todos los recursos, del área Configuración de biblioteca.
- Seleccione DataExpress en la lista.
- Seleccione Incluir clases necesarias y todos los recursos.

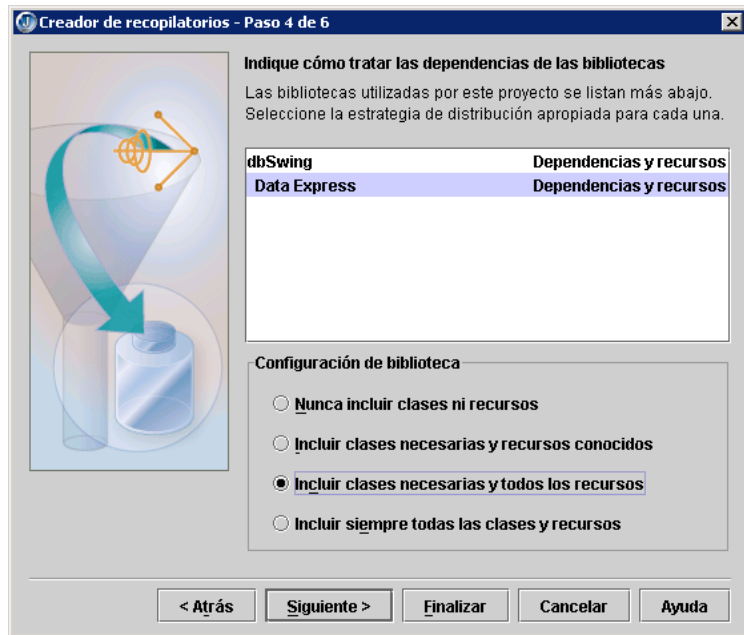
Aunque no se ha utilizado la biblioteca DataExpress en este tutorial, algunas clases dbSwing dependen de clases DataExpress. Por ello, es necesario incluirlas en el recopilatorio.

Ambas bibliotecas se distribuyen con Dependencias y recursos.

**Precaución**

El Creador de recopilatorios no siempre puede encontrar todos los archivos. Es recomendable que pruebe la aplicación distribuida, añada cualquier archivo ausente y vuelva a distribuir.

El paso 4 del Asistente tiene el siguiente aspecto:



**10** Haga clic en Siguiente para ir al Paso 5, en el que se crea el archivo descriptor.

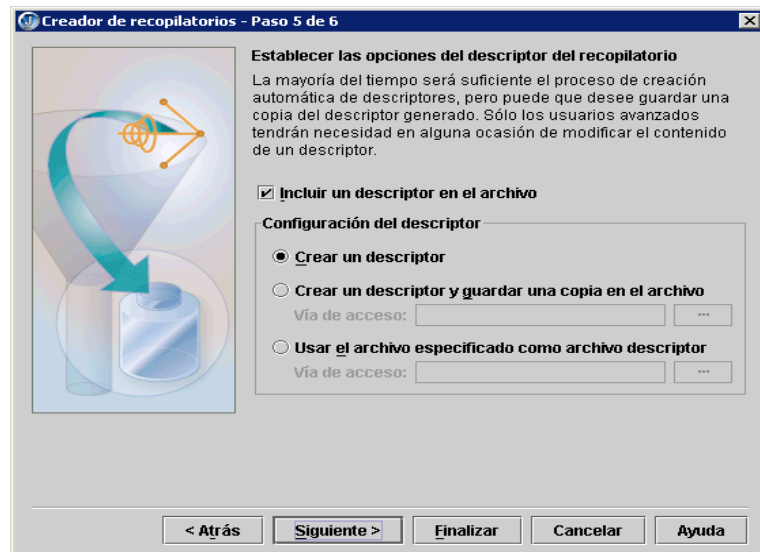
En un archivo recopilatorio sólo puede haber un archivo descriptor, y tiene siempre la vía de acceso META-INF/MANIFEST.MF.

**11** Acepte los valores por defecto en el Paso 5 del asistente. El resultado es el siguiente:

- Incluir automáticamente el archivo descriptor en el archivo recopilatorio.
- Crear automáticamente el archivo descriptor.



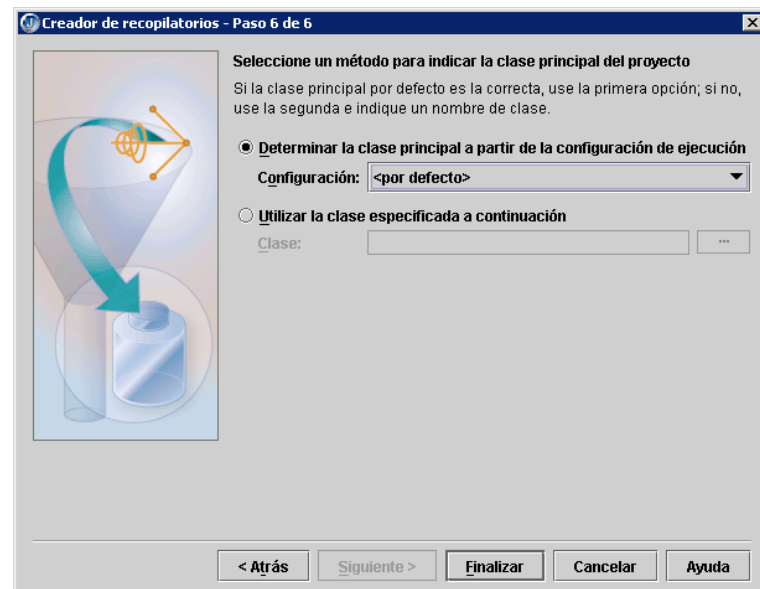
El Paso 5 del asistente tendrá este aspecto:



**12** Haga clic en Siguiendo para ir al paso 6, dónde decidirá cómo el Creador de recompilatorios halla la clase principal.

En este tutorial, deje el parámetro para la clase principal por defecto. Esta opción utiliza la clase principal de la configuración de ejecución por defecto, especificada en la ficha Ejecutar del cuadro de diálogo Propiedades de proyecto.

El Paso 6 del asistente tendrá este aspecto:



**13** Haga clic en Finalizar para crear el nodo de archivo recopilatorio.

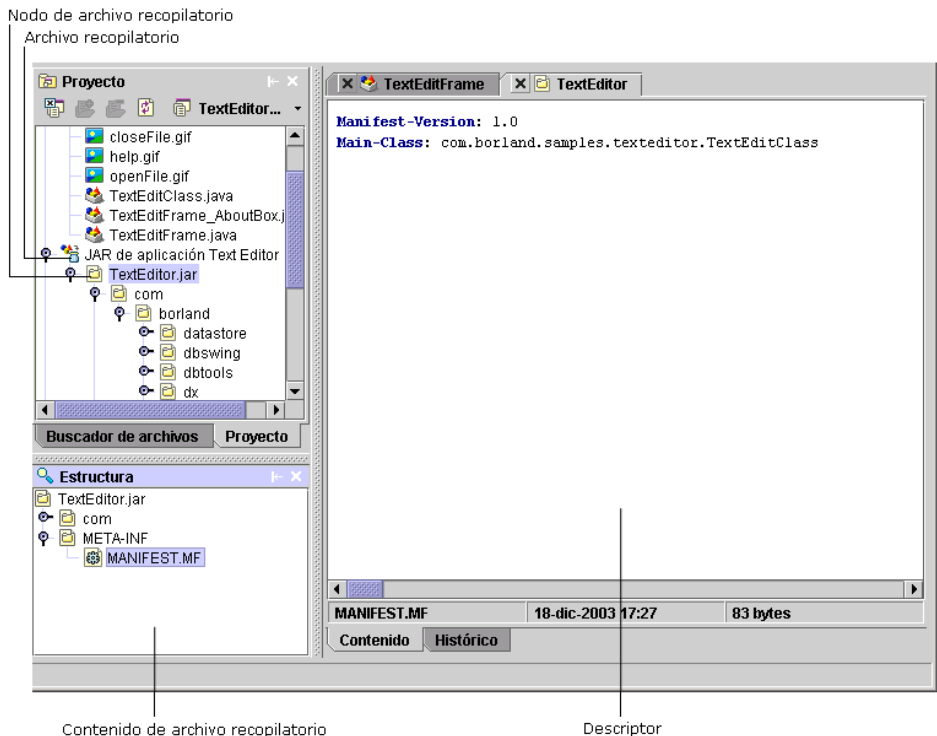
En el panel de proyecto se puede ver el nodo de archivo recopilatorio `TextEditorApplication`. Puede hacer clic con el botón derecho sobre el nodo de archivo recopilatorio y generarlo o cambiar sus propiedades.

**14** Seleccione Proyecto|Crear proyecto o haga clic con el botón derecho en el nodo del recopilatorio y elija Crear para crear el proyecto y generar el archivo JAR.

**15** Para ver el archivo recopilatorio, deberá ampliar el nodo de archivo recopilatorio en el panel de proyecto.

**16** Haga doble clic sobre el archivo recopilatorio `TextEditor.jar`.

En el panel de estructura aparece su contenido y en el panel de contenido aparece el contenido del archivo descriptor. En estos momentos, JBuilder debería tener este aspecto:



Observe las dos cabeceras del archivo descriptor:

Manifest-Version: 1.0	Informa de que las entradas del descriptor toman la forma de pares “cabecera:valor” y que se trata de la versión 1.0 de la especificación del descriptor.
Main-Class: texteditor.TextEditClass	Indica que <code>TextEditClass.class</code> es el punto de entrada para la aplicación (la clase que contiene el método <code>public static void main(String[] args)</code> , el cual ejecuta la aplicación).

### Consulte

- “El Creador de compiladores” en *Creación de aplicaciones con JBuilder*.
- “Understanding the Manifest” (en inglés) en <http://java.sun.com/docs/books/tutorial/jar/basics/manifest.html>.

## Ejecución desde la línea de comandos de la aplicación distribuida

---

Antes de ejecutar la aplicación desde la línea de comandos, debe cerciorarse de que la variable de entorno `PATH` de su sistema operativo coincide con el directorio `jre/bin/` del JDK, el entorno de ejecución. El proceso de instalación de JBuilder garantiza que éste sepa dónde encontrar los archivos de clases JDK. No obstante, una vez que abandone el entorno JBuilder, su sistema necesita saber dónde están instalados los archivos de clase para el ejecutor Java. La forma de definir la variable de entorno `PATH` depende del sistema operativo utilizado.

Para ejecutar el tutorial de Editor de texto desde la línea de comandos:

- 1 Cambie a su ventana de línea de comandos y cambie al directorio `TextEditor` donde se halla el archivo JAR.
- 2 Compruebe que Java está en su `PATH` tecleando `java` en la línea de comandos. Si es así, se mostrarán el uso y las opciones de Java. Si no se encuentra en su `PATH`, configure la variable de entorno `PATH` con el directorio `bin` del JDK.
- 3 Escriba en la línea de comandos:

```
java -jar TextEditor.jar
```

donde

- `java` es la herramienta de Java que ejecuta el archivo JAR.
- `jar` es la opción que indica a la MV de Java que se trata de un archivo recopilatorio.
- `TextEditor.jar` es el nombre del archivo recopilatorio.

El archivo descriptor ya indica en la cabecera de la clase principal la clase que se ejecutará, no es necesario por lo tanto especificar el nombre de la

clase al final de este comando. Además, todas las clases, recursos y dependencias están incluidos en el archivo JAR recopilatorio, por lo que no será necesario especificar una `classpath` (vía de acceso a clases) ni copiar bibliotecas de JBuilder en este directorio.

**Nota**

Cuando utiliza la opción `-jar` para ejecutar un archivo JAR, la ejecución Java pasa por alto los parámetros explícitos de la vía de acceso a las clases. Si intenta ejecutar este archivo JAR cuando no está en el directorio `TextEditor`, necesita utilizar el siguiente comando Java:

```
java -jar -classpath <vía_completa><nombre_clase_princ>
```

El módulo de ejecución (runtime) de Java busca en el archivo JAR la clase de inicio y las otras clases utilizadas por la aplicación. La MV de Java utiliza tres vías para buscar los archivos: la vía de la clase de inicio, la de las extensiones instaladas y la vía de las clases del usuario.

Si la aplicación no funciona, examine los errores generados en la ventana de línea de comando. Compruebe que la carpeta `jbuilder.lib` se encuentra en su `CLASSPATH`. Asegúrese de que está en el directorio correcto y de que no hay errores de escritura en el comando.

- 4 Pruebe la aplicación donde funciona, para asegurarse de que lo hace correctamente. Cree, guarde y abra un archivo. Haga clic con el botón derecho en el editor de texto para ver si funciona el menú contextual. Además, la aplicación podría ejecutarse y aún así tener errores. Compruebe la ventana de línea de comandos para ver si hay algún mensaje de error. Lea los mensajes de error, si los hay, para buscar las clases o paquetes que falten.

**Consulte**

- “How Classes Are Found” en <http://java.sun.com/j2se/1.3/docs/tooldocs/findingclasses.html> si desea más información sobre la forma en que Java busca las vías de acceso.

## Modificación del archivo JAR y nueva prueba de la aplicación

---

Si tiene errores de ejecución, necesita añadir cualquier clase que falte al archivo JAR, utilizando el Creador de recopilatorios. Si no ha tenido errores, se puede saltar estos pasos:

- 1 Vuelva al proyecto Editor de texto en JBuilder.
- 2 Haga clic con el botón derecho del ratón en el grupo EJB, en el panel del proyecto, y elija Propiedades.
- 3 Elija la pestaña adecuada y haga los cambios necesarios.
- 4 Pulse Aceptar para cerrar el cuadro de diálogo de propiedades.
- 5 Haga clic con el botón derecho en el nodo del recopilatorio y elija Crear para reconstruir el archivo JAR.

- 6** Repita el procedimiento de prueba con el archivo JAR modificado tal como se describe en “Ejecución desde la línea de comandos de la aplicación distribuida” en la página 10-49, pruebe la aplicación donde funciona.

¡Eso es todo!

Como puede ver, el proceso de distribución exige relacionar mucha información. La distribución es mucho más que crear simplemente un archivo JAR. No sólo tiene que asegurarse de que proporciona todas las clases, recursos y bibliotecas necesarios en su juego de distribución; sino que además tiene que preocuparse de otros asuntos, como conocer las herramientas `java` y `jar`. Además, existen diferencias a la hora de ejecutar aplicaciones basadas en JDK 1.1 y Java 2.

Tómese el tiempo necesario para estudiar la abundante información disponible en los enlaces de la sede web de Sun suministrados aquí, así como en otras fuentes de Internet y en los innumerables y excelentes libros escritos sobre el tema.

### Consulte

- “Distribución de programas en Java” en *Creación de aplicaciones con JBuilder*.
- “El Creador de recopilatorios” en *Creación de aplicaciones con JBuilder*.
- Sede web de Sun sobre las herramientas básicas, en <http://java.sun.com/j2se/1.3/docs/tooldocs/tools.html#basic>.
- El tutorial de Sun sobre archivos Jar en <http://java.sun.com/docs/books/tutorial/jar/index.html>.



# Tutorial: Creación de una interfaz de usuario con diseños anidados

Este tutorial describe paso a paso la creación de una interfaz de usuario de una aplicación hipotética como, por ejemplo, un procesador de textos sencillo. Muestra cómo crear la interfaz de usuario con las herramientas de diseño visual de JBuilder, mediante paneles anidados y los gestores de diseño más sencillos. Utiliza `BorderLayout`, `FlowLayout`, y `GridLayout`. A causa de su complejidad, `GridBagLayout` y `CardLayout` se tratan con detalle por separado. Para obtener más información sobre los gestores de diseño, consulte el [Capítulo 8, “Gestores de diseño”](#) y el tutorial de Sun “Creating a GUI with JFC/Swing”, que se encuentra en <http://java.sun.com/docs/books/tutorial/uiswing/index.html>.

En este tutorial, utilizará `JPanel` de Swing en todos los componentes de panel, ya que al contrario que el `Panel` AWT, tiene una propiedad `border`. La propiedad `border` se utiliza para añadir bordes a los paneles, de forma que sea posible ver sus límites cuando se les añaden componentes en el diseñador. Una vez finalizado el diseño de la interfaz de usuario, es posible eliminar los bordes donde se desee. El componente `JPanel` está situado en la pestaña de Contenedores Swing de la paleta de componentes, situada en la parte superior del diseñador de interfaz de usuario de JBuilder. A lo largo de este tutorial, todas las referencias a paneles conllevan el uso de `JPanel`.

En este tutorial también se cambiarán algunos de los gestores de diseño de paneles durante la fase de diseño a `XYLayout` (JBuilder Profesional o Enterprise) y diseño `null` (JBuilder Personal). `XYLayout`, un gestor de diseño de JBuilder, sitúa los componentes en un contenedor en coordenadas `x,y` específicas, relativas a la esquina superior izquierda del contenedor.

Independientemente del tipo de pantalla, el contenedor siempre retendrá las posiciones relativas x,y de los componentes. Diseño `null` significa que no hay ningún administrador de diseño asignado al contenedor. El diseño `null` (de Swing) es muy parecido al `XYLayout` en cuanto a que puede poner componentes en un contenedor con coordenadas específicas x,y relativas a la esquina superior izquierda del contenedor.

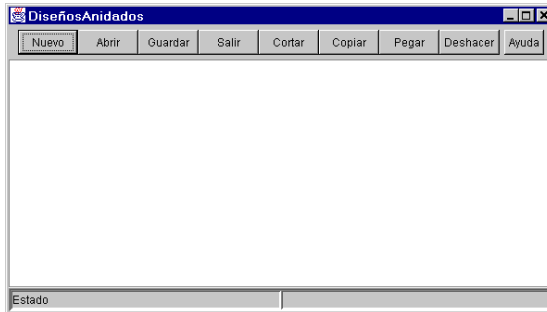
El uso del posicionamiento absoluto del `XYLayout` o del `null` simplifica el diseño de su interfaz de usuario, ya que puede controlar el tamaño y la colocación de los componentes. Pero la desventaja es que no se ajustan bien a las diferencias de los sistemas, y por tanto, no son diseños portables. Debido a que `XYLayout` y `null` usan posicionamiento absoluto, los contenedores y componentes no se restauran correctamente cuando el usuario restaura la ventana de la aplicación. Por tanto, es mejor no dejar un contenedor en `XYLayout` o `null` para la distribución debido a su falta de portabilidad. Los administradores de diseño AWT, que no usan posicionamiento absoluto, hacen más fácil el ajuste de su aplicación a la apariencia de un sistema diferente, a los diferentes tamaños de fuente del sistema y al tamaño cambiante de un contenedor. Por ello, son más portables que `XYLayout` y `null`. Por lo tanto, tras haber completado el diseño de la interfaz de usuario de este tutorial, se cambiarán los gestores de diseño de todos los contenedores a los diseños AWT más portables.

Para obtener el mayor control posible en el diseño de las interfaces, además de una estructura más sencilla y menos anidada, es mejor utilizar una combinación de `GridBagLayout` y las técnicas de anidado que se explican en este tutorial. Si desea más detalles acerca de `GridBagLayout`, consulte el [Capítulo 12, “Tutorial: Creación de diseños GridBagLayout en JBuilder”](#). Si quiere tomarse en serio el desarrollo de interfaz de usuario con Java, es importante que dedique tiempo a aprender cómo se usa `GridBagLayout`. Una vez que lo entienda, lo considerará indispensable.

La aplicación de interfaz de usuario que está a punto de diseñar contiene varios paneles que soportan componentes, tales como botones, etiquetas, y un área de texto. Ya que este tutorial está centrado en el diseño de interfaz de usuario, la aplicación diseñada no será completamente funcional. Por ejemplo, si pincha en el botón Guardar de la barra de herramientas, no pasará nada. Así mismo, este tutorial no es la única manera de diseñar esta interfaz de usuario. Por ejemplo, normalmente usaría el componente `JToolBar` al crear barras de herramientas y `GridBagLayout`. En este tutorial, se usan paneles y botones de la barra de herramientas para mostrar el uso de paneles anidados y diseños. Los botones pueden ser cualquier tipo de componente en el diseño.



La interfaz de usuario que va a diseñar es la siguiente:



**Nota** Las imágenes de pantalla en este tutorial utilizan el aspecto Metal del entorno de desarrollo integrado de JBuilder y del entorno de ejecución de la aplicación.

Este tutorial supone que usted está familiarizado con Java y con el IDE (Entorno integrado de desarrollo) de JBuilder. Si desea obtener más información acerca de Java, consulte *Procedimientos iniciales con Java*. Para obtener más información sobre el IDE de JBuilder IDE, consulte “El entorno de JBuilder” en Creación de aplicaciones con JBuilder.

Si desea obtener información útil para ver e imprimir los tutoriales, consulte el apartado Tutoriales, en Sugerencias de JBuilder. El apartado de opciones de accesibilidad en las Sugerencias de JBuilder contiene sugerencias sobre la utilización de las funciones de JBuilder para mejorar la facilidad de uso de JBuilder por parte de personas con discapacidades.

Para obtener información sobre las convenciones utilizadas en este tutorial y en otra documentación de JBuilder, consulte [“Convenciones de la documentación” en la página 1-5](#).

## Paso 1: Creación del proyecto de interfaz de usuario

---

Antes de crear la aplicación, es necesario crear un proyecto para todos los archivos de las aplicaciones. Una vez creado el proyecto, cree los archivos fuente con la ayuda del Asistente para aplicaciones. Cree el proyecto por medio del Asistente para proyectos.

### Utilización del Asistente para proyectos

---

Para abrir el Asistente para applets:

- 1 Elija Archivo|Nuevo proyecto para iniciar el Asistente para proyectos.
- 2 Realice los siguientes cambios en el Paso 1:
  - Nombre: `DiseñosAnidados`.

**Nota**

Por defecto, JBuilder utiliza este nombre de proyecto para crear los nombres del directorio del proyecto y el paquete de las clases.

- Seleccione la opción Generar archivo de notas del proyecto. Cuando se activa esta opción, el Asistente para proyectos crea un archivo HTML para las notas del proyecto y lo añade.
- Compruebe que la opción Añadir proyecto al grupo activo se encuentra desactivada. De lo contrario, este proyecto se añadiría al trabajo real.

- 3** Acepte las restantes opciones por defecto en el Paso 1.
- 4** Pulse Siguiente para ir al Paso 2 del Asistente para proyectos.
- 5** Acepte las vías de acceso por defecto del Paso 2.
- 6** Pulse Siguiente para ir al Paso 3 del Asistente para proyectos.
- 7** Rellene los campos de la clase Javadoc. Esta información se guarda en el archivo HTML del proyecto. Se utiliza en los comentarios de Javadoc si selecciona la opción Crear comentarios de cabecera de muchos de los asistentes de JBuilder, como los asistentes para aplicaciones y clases.
- 8** Pulse Finalizar para crear el proyecto. Un archivo de proyecto y un archivo de proyecto HTML se añaden al proyecto y aparecen en el panel de proyecto.

**Consulte**

- “Creación y gestión de proyectos” en *Creación de aplicaciones con JBuilder*.
- “Gestión de las vías de acceso” en *Creación de aplicaciones con JBuilder*.
- “Cómo genera JBuilder las vías de acceso” en el capítulo “Gestión de las vías de acceso” de *Creación de aplicaciones con JBuilder* si desea más información sobre el segundo paso del Asistente para proyectos.
- “Localización de los archivos” en el capítulo “Gestión de las vías de acceso” de *Creación de aplicaciones con JBuilder*.

A continuación, se crearán los archivos de la aplicación con la ayuda del Asistente para aplicaciones.

## Paso 2: Generación de los archivos fuente de la aplicación

---

Una vez que tenga un proyecto, puede utilizar el Asistente para aplicaciones para generar los archivos de la aplicación de forma automática.

### Utilización del Asistente para aplicaciones

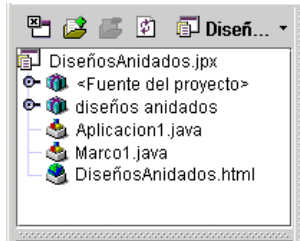
---

Para abrir el Asistente para aplicaciones:

- 1** Abra la galería de objetos seleccionando Archivo|Nuevo.
- 2** Seleccione la pestaña General si no es la pestaña activa.

- 3 Haga doble clic en el icono Aplicación para abrir el Asistente para aplicaciones. Observe que el nombre del paquete por defecto se extrae del nombre del proyecto.
- 4 Pulse el botón Finalizar. No se utilizarán los archivos adicionales generados en el Paso 2 del asistente para aplicaciones; en cambio, se utilizarán los valores por defecto del Paso 1 del asistente.

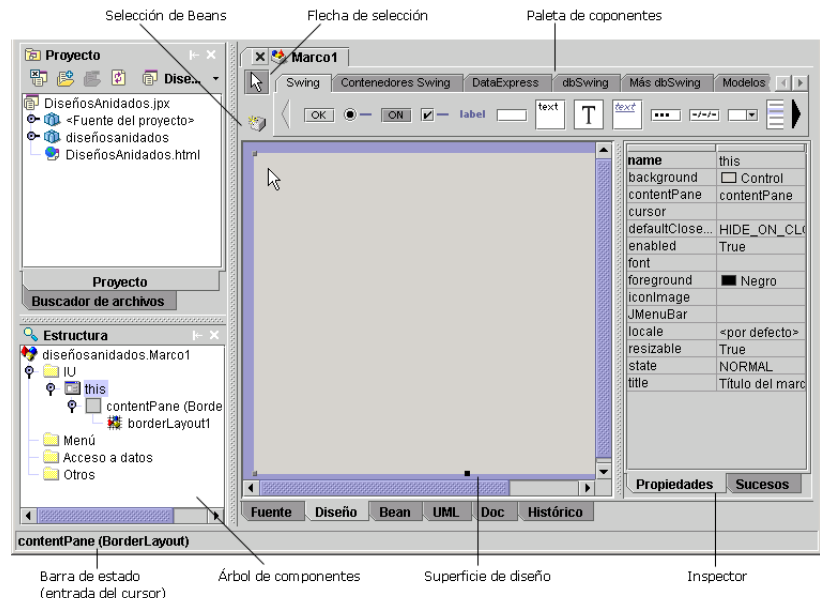
El proyecto aparece en el panel de proyecto del Visualizador de aplicaciones con `Marco1.java` abierto en el panel de contenido. Hay dos archivos más en el panel de proyecto: `Aplicacion1.java`, que contiene el método `main()` y `Marco1.java`, el contenedor de la interfaz del usuario.



**Nota** Aparece también un nodo de paquetes de código fuente en el panel de proyecto si está seleccionada la opción Recopilación automática de paquetes fuente en la ficha General del cuadro de diálogo Propiedades de Proyecto (Project/Propiedades de proyecto).

- 1 Haga clic en la ficha Diseño de la parte inferior de `Marco1.java` en el panel de contenido para abrir el diseñador visual en modo de diseño de interfaz de usuario.

**Figura 11.1** Diseñador de interfaces de usuario



Ahora, el panel de estructura contiene un árbol de componentes y el panel de contenido contiene el diseñador de interfaces, con el Inspector a la derecha y la paleta de componentes sobre la superficie de diseño.

Observe la estructura del diseño de la interfaz de usuario en el árbol de componentes, tal y como la ha creado el Asistente para aplicaciones. Por el momento hay un marco llamado `this`, en el que se encuentra el elemento `contentPane (BorderLayout)` en la carpeta llamada `IU`. Bajo esta carpeta y este marco se colocarán todos los componentes visuales de la interfaz de usuario que se añadan al diseño. Así diseñará el `contentPane` en el diseñador de interfaces de usuario. Observe también que el marco `this` está resaltado en el árbol de componentes y que los tiradores de redimensionamiento se encuentran en cada esquina del marco del diseñador. El marco está anclado en su esquina superior izquierda. Las propiedades para `this` se muestran en el Inspector a la derecha del diseñador.

**Sugerencia**

La barra de estado bajo el panel de estructura muestra información acerca de cualquier componente sobre el que se sitúe el ratón en el diseñador.



- 2 En la ficha Propiedades del Inspector, el valor de la propiedad `title` es "Título del marco".
- 3 Haga clic tres veces en `Título del marco` para resaltarlo, escriba `Diseños anidados` en su lugar y pulse *Intro*. El nuevo título aparece en la barra de título del marco durante la ejecución.

A continuación, aumente el área sobre la que está trabajando en el diseñador de interfaces, de modo que tenga más espacio.

- 4 Haga clic en el tirador (nib) de la esquina inferior derecha y arrástrelo en diagonal hacia fuera del marco, para ampliarlo en el diseñador. Así se redimensiona el marco contenedor principal de su interfaz de usuario, la instancia del componente `Marco1` de la clase `JFrame`.

**Sugerencia**

Si no puede ver el tirador en la esquina inferior derecha, maximice el Visualizador de aplicaciones antes de agrandar el marco. Y si desea aún más espacio, seleccione `VerIPanel` de proyecto para esconder el proyecto y los paneles de estructura. Sin embargo, tenga en cuenta que hacer esto oculta el árbol de componentes que está en el panel de estructura, por lo

que deberá cambiar siempre que desee ver el árbol. También puede agrandar el diseñador arrastrando sus bordes.



El tamaño real que presenta en la pantalla un contenedor de la interfaz de usuario durante la ejecución no depende necesariamente del tamaño con que se define en el diseñador de interfaces de usuario. El gestor de diseño del contenedor es el que determina este tamaño.

Puede cambiar manualmente el tamaño del marco de la interfaz de usuario, redimensionándolo durante la ejecución.

En el siguiente paso se verá cómo cambiar el gestor de diseño de `contentPane`.

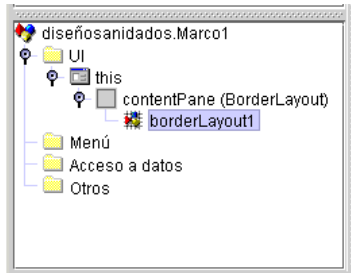
## Paso 3: Modificación del diseño del panel de contenido

---

Los contenedores de interfaz de usuario de Java utilizan un objeto especial, denominado gestor de diseño, para controlar la forma en que se posicionan y se dimensionan los componentes del contenedor cada vez que éste se visualiza. Los gestores de diseño le ofrecen un diseño de interfaces de usuario lleno de ventajas, entre las que se encuentran la portabilidad entre distintas plataformas, el redimensionamiento dinámico de los componentes durante la ejecución y la facilidad para la traducción con cadenas de diferente tamaño.

Un gestor de diseño coloca los componentes automáticamente en un contenedor basado en las normas de diseño y la configuración de propiedades, las `restricciones de diseño` asociadas con cada componente, las `propiedades comunes de los componentes` (`preferredSize`, `minimumSize`, `maximumSize`, `alignmentX`, y `alignmentY`) y el tamaño del contenedor. Por defecto, el gestor de diseño de `contentPane` es `BorderLayout`. Puede ver el gestor de

diseño si hace clic en el icono a la izquierda de `contentPane` en el árbol de componentes y, a continuación, expande el árbol.



`BorderLayout` se utiliza mejor en el diseño de interfaces si se colocan cinco o menos componentes y uno de ellos ocupa en el centro la mayor parte del espacio. `BorderLayout` organiza los componentes en cinco ubicaciones: Center, North, South, East, y West, donde Center es la más grande.

Si deja `contentPane` en `BorderLayout`, puede arrastrar por error un panel con el ratón a otra área del marco de `BorderLayout`, con lo que disminuiría su altura o anchura y se desplazaría hacia uno de los extremos del marco. Si esto ocurre tras haber llenado el diseño de la interfaz de usuario con numerosos componentes, no le resultaría fácil detectar cuál es el problema.

Para que esto no ocurra, cambie `contentPane` a `XYLayout` o a `null`, que admiten un mayor control en la colocación de los componentes.

- 1 Seleccione `contentPane` en el árbol de componentes.
- 2 Seleccione su propiedad `layout` en la ficha Propiedades del Inspector.
- 3 Haga clic en la flecha de lista desplegable y seleccione `XYLayout` o `null`.

Si se empieza con `null` o `XYLayout` resulta más fácil crear el prototipo en el contenedor. Más adelante, después de añadir los componentes al contenedor, puede activar un diseño portable adecuado para su aplicación. Es mejor no dejar el contenedor en `null` o en `XYLayout`, porque utilizan la colocación absoluta y, como consecuencia, los componentes no se ajustan bien al redimensionar los supercontenedores. Estos diseños no se ajustan correctamente a las distintas configuraciones de los usuarios y los sistemas, y por lo tanto no son portables. Es mejor utilizar estos diseños en la fase de diseño y, a continuación, al final de esta fase, convertir todo en diseños portables. Al final del tutorial, cambie el gestor de diseño de nuevo a `BorderLayout`, una vez que todos los demás paneles ya tengan sus diseños definitivos.

En el siguiente paso se verá cómo añadir los paneles principales a `contentPane`.

## Paso 4: Cómo añadir los paneles principales

Ahora, comience a añadir paneles al diseño de su interfaz de usuario. Cuando acabe, el diseño debe tener un aspecto parecido a este:



1 Seleccione `contentPane` en el árbol de componentes.



2 Pulse la pestaña Contenedores Swing de la paleta de componentes, en la parte superior del diseñador, y pulse el icono `JPanel`.

### Sugerencia

Desplace el ratón sobre un componente de la paleta de componentes para ver su nombre en la ayuda inmediata.

3 Dibuje en el primer panel haciendo clic y arrastrando en sentido diagonal desde la esquina superior izquierda del diseñador hacia el lado derecho del `contentPane`, con lo que se crea un panel que ocupa el cuarto superior del diseño. Este panel va a ser el contenedor de las dos barras de herramientas de la parte superior de la aplicación.

Observe que en el árbol de componentes se ha añadido uno denominado `jPanel1` en la carpeta `IU`, debajo de `contentPane`. Los tiradores de redimensionamiento que aparecen en los bordes del panel indican su tamaño y su posición. Haga clic en el icono de ampliación si desea ver el gestor de diseño de `jPanel`, `FlowLayout`. `FlowLayout` se utiliza al colocar varios componentes en una fila. Como va a colocar dos paneles de barras de herramientas en una fila de este panel superior, deje el diseño como `FlowLayout`.

4 Haga clic mientras pulsa la tecla *Mayús* en el icono de `JPanel` de la paleta de componentes y dibuje dos paneles más en el diseñador. No se preocupe si el diseño no es perfecto. Podrá mejorarlo más adelante. Observe que el segundo y tercer panel también utilizan `FlowLayout`. Se cambiarán más adelante, antes de añadir componentes. Compruebe en el árbol de componentes que los tres paneles están anidados dentro del `contentPane`.

### Sugerencia

Si hace clic mientras pulsa la tecla *Mayús* en el componente `JPanel` de la paleta de componentes, puede añadir varios paneles sin hacer clic en el



icono de `JPanel` cada vez. Esto resulta muy útil si se van a añadir varios componentes idénticos a un diseño. Una vez añadidos los paneles, haga clic en la flecha de selección a la izquierda de la paleta, para eliminar la selección del componente `JPanel`.

- 5 Haga clic en la flecha de selección de la paleta de componentes para deshacer la selección de la característica de selección múltiple. Cada uno de los tres paneles que ha añadido va a contener otros componentes. El panel superior tendrá dos paneles, cada uno de ellos con una barra de herramientas con botones. El panel central tendrá un panel de desplazamiento y una área de texto. El panel inferior tendrá una barra de estado con dos etiquetas. Para distinguir mejor cada panel en el diseñador, cambie el valor de la propiedad `border` a `RaisedBevel`. Lo hará más rápido seleccionando todos los paneles y cambiando la propiedad de borde de todos los paneles al mismo tiempo.
- 6 Seleccione `jPanel1`, `jPanel2` y `jPanel3` haciendo clic mientras pulsa la tecla *Mayús*. Los puede seleccionar en el diseñador o en el árbol de componentes.
- 7 Seleccione `RaisedBevel` en la lista desplegable de la propiedad `border` en la ficha Propiedades del Inspector.

Si bien no es necesario utilizar un borde en los paneles durante el diseño, el trabajo resultará más sencillo cuando se aniden varios paneles y componentes. Esto se debe a que en cuanto seleccione un componente en la paleta y haga clic en el panel, los tiradores de redimensionamiento del panel desaparecen y no podrá ver si aún está dentro de él cuando esté redimensionando el nuevo componente.

Sólo con fines demostrativos, las imágenes de este tutorial también muestran los paneles en diferentes tonos de gris para facilitar su diferenciación. Cambiar el color de fondo de los paneles es otra manera de hacerlos visibles en el diseñador si no se desea utilizar los bordes.

Ajuste el diseño con la ayuda de algunas de las opciones del menú del diseñador. Por ejemplo, puede hacer que estos tres paneles tengan la misma anchura en sentido horizontal.

- 1 Cancele la selección múltiple en el árbol de componentes seleccionando otro componente.
- 2 Seleccione el panel superior en el diseñador y utilice los tiradores de redimensionamiento para ajustar la anchura.
- 3 Mientras mantiene pulsada la tecla *Mayús*, haga clic en los paneles central e inferior. Ya están seleccionados los tres paneles.
- 4 Haga clic con el botón derecho del ratón sobre uno de los paneles seleccionados y elija la opción Tamaño horizontal igual. Los paneles central e inferior se ajustarán a la misma anchura que el panel superior. A continuación seleccione Alinear a la izquierda y Distribución vertical en el menú contextual del diseñador.



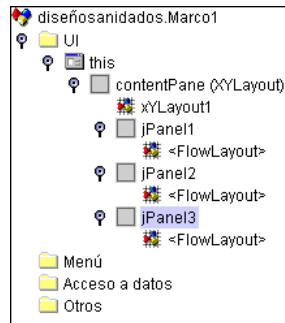
- 5 Si desea cancelar la selección múltiple, haga clic en cualquier componente que no esté seleccionado o en el árbol de componentes.

#### Sugerencia

El primer componente seleccionado está emparejado, tenga cuidado con cuál selecciona en primer lugar.

Si desea arrastrar un panel entero, selecciónelo y coloque el ratón en el tirador central. Aparece una flecha cuádruple. Haga clic y arrastre el componente a la nueva posición. Tenga cuidado de no arrastrarlo a un contenedor distinto, como contentPane. Si se equivoca, elija Edición|Deshacer y el diseño volverá a su estado anterior.

Observe ahora la estructura del diseño de la interfaz de usuario en el árbol de componentes:



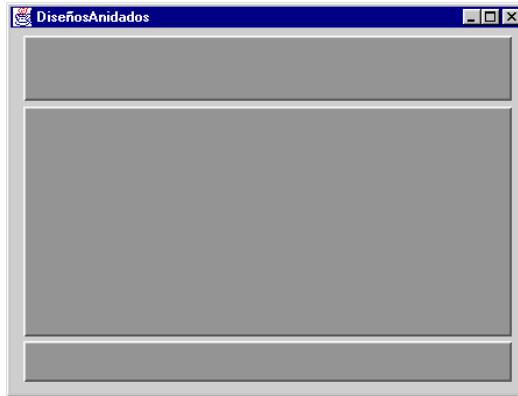
A continuación, cambie el nombre de los paneles que ha añadido, con el fin de que tengan más significado.

- 1 Seleccione el contenedor en el árbol de componentes. Haga clic con el botón derecho del ratón en el panel y seleccione en el menú Cambiar nombre. Asigne a cada panel un nombre adecuado:
  - Panel superior: VolverAlInicio
  - Panel central: Central
  - Panel inferior: BarraDeEstado

Llegados a este punto es conveniente guardar el proyecto.

- 2 Elija Archivo|Guardar todo en el menú principal.

- 3 Elija Ejecutar|Depurar proyecto o pulse el botón Depurar de la barra de herramientas. Las barras de herramientas se asemejarán a las siguientes:



Puede que tenga que modificar el tamaño del marco `this` o la posición de los paneles después de ejecutar la aplicación. Antes de agrandar ningún panel, seleccione `this` y aumentelo. A continuación, al ejecutar la aplicación, redimensione la ventana de la aplicación. Observe que los paneles no se redimensionan al mismo tiempo que la ventana. Esta es la razón por la que no debe dejar el diseño en `XYLayout` o en `null`. Al final del tutorial, al cambiar el diseño de `contentPane` a `BorderLayout`, los paneles se redimensionarán de forma correcta al mismo tiempo que la ventana.

- 4 Salga de la aplicación.
- 5 Haga clic con el botón derecho del ratón en la pestaña Aplicacion1 del panel de mensajes y, a continuación, seleccione la opción Eliminar todas las pestañas para cerrar este panel.

En el siguiente paso se verá cómo añadir paneles adicionales al panel superior.

## Paso 5: Creación de barras de herramientas

---

Antes de añadir paneles al panel superior, que va a contener dos barras de herramientas, debe cambiar el gestor de diseño del panel superior a `XYLayout` o a `null`, con el objeto de tener un control total en la colocación de los paneles que va a añadir. A continuación, debe añadir dos paneles y cambiarles el nombre por otros más significativos. El primero albergará la barra de herramientas izquierda, y el segundo la derecha.

- 1 Seleccione el panel superior.
- 2 Cambie el gestor de diseño de `<default layout>` (`FlowLayout`) a `XYLayout` o a `null`.
- 3 Dibuje dos paneles en el panel superior con el mismo método utilizado en el paso 1 y que tengan la misma apariencia de la imagen que se muestra a

continuación. No se preocupe todavía del tamaño y la posición. Podrá mejorarlos más adelante.



- 4 Seleccione cada panel en el árbol de componentes y cambie el nombre de los paneles en el panel superior de la siguiente manera:
  - Panel superior izquierdo: `barradeherramientas_izq`
  - Panel superior derecho: `barradeherramientas_dcha`
- 5 Cambie los bordes de los dos paneles a `RaisedBevel`.  
A continuación, ajuste la longitud horizontal de los paneles de las barras de herramientas y alinéelos en la parte superior.
- 6 Mantenga pulsada la tecla `Ctrl` mientras selecciona `barradeherramientas_izq` y `barradeherramientas_dcha` en el árbol de componentes.
- 7 En el diseñador de interfaces de usuario, haga clic con el botón derecho sobre uno de estos paneles seleccionados y elija `Alinear arriba`.
- 8 Haga clic con el botón derecho mientras los dos paneles están seleccionados y elija la opción `Tamaño vertical igual`.
- 9 Guarde el trabajo.

En el siguiente paso se verá cómo añadir botones a las barras de herramientas.

## Paso 6: Cómo añadir botones a las barras de herramientas

Antes de añadir botones a las barras de herramientas, debe cambiar los gestores de diseño del panel de las dos barras a `GridLayout`, para que los botones que se añadan tengan el mismo tamaño. `GridLayout` se utiliza para colocar componentes de igual tamaño en una rejilla de filas y columnas.

- 1 Seleccione `barradeherramientas_izq` y `barradeherramientas_dcha` y cambie sus propiedades `layout` a `GridLayout` en el Inspector. Ahora, se añadirán los

botones a los paneles de la barra de herramientas y se definirán sus etiquetas.

2 Seleccione la pestaña Swing de la paleta de componentes.



3 Haga clic mientras pulsa la tecla *Mayús* en el componente `JButton` y, a continuación, haga clic ocho veces en el panel `barradeherramientas_izq` del árbol de componentes. El primer botón rellena el panel por completo y, al añadir botones, `GridLayout` les da el mismo tamaño.

4 Haga clic en la flecha de selección de la paleta de componentes para deshacer la selección del componente `JButton`.

5 Seleccione cada vez un botón, comenzando por el botón izquierdo más alejado, y cambie su propiedad `text` en el Inspector de la siguiente manera:

- Nuevo
- Abrir
- Guardar
- Salir
- Cortar
- Copiar
- Pegar
- Deshacer

**Importante**

Es posible que aún no pueda ver el texto en los botones porque todavía deben ajustarse los márgenes. Haga lo mismo con todos los botones cuando los haya añadido.

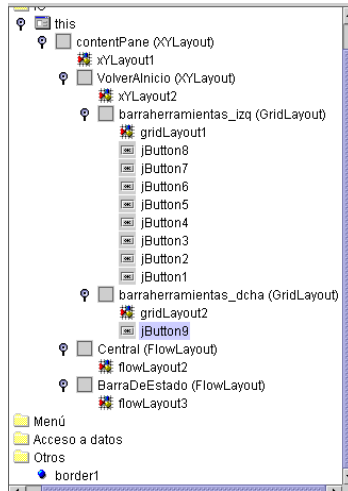
**Nota**

Si los botones no están en el orden correcto, haga clic con el botón derecho del ratón sobre uno y elija en el menú contextual del diseñador Trasladar al primero o Trasladar al último.

6 Coloque `jButton9` en el panel `barradeherramientas_dcha` y cambie su texto a *Ayuda*.

Lo más importante en este momento es que los botones estén completamente anidados dentro de sus respectivos paneles. Asegúrese de que todos estén incrustados correctamente, comprobando el árbol de componente para ver si aparecen sangrados bajo el panel correcto. Si alguno de los botones no se anidó en su panel, se verá en el árbol de componentes al mismo nivel que los paneles.

El árbol de componentes tendrá una apariencia parecida a la siguiente, aunque sus botones pueden estar en diferente orden:



A continuación, reduzca los márgenes de modo que se pueda leer el texto de los botones.

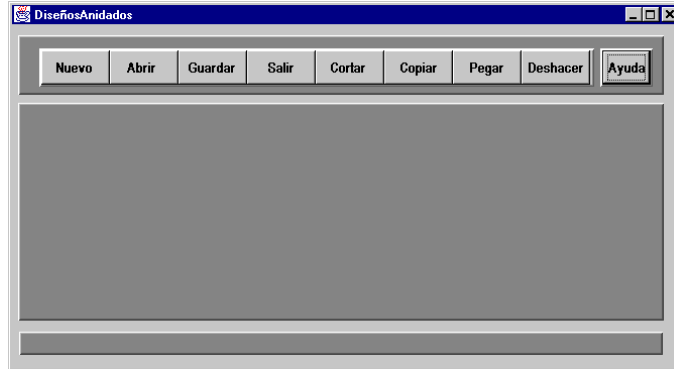
- 1 Seleccione todos los botones en el diseñador o en el árbol de componentes. Puede utilizar la tecla *Mayús* si desea seleccionar componentes de forma consecutiva, y la tecla *Ctrl* para seleccionar el botón Ayuda en la barra de herramientas derecha del árbol de componentes.
- 2 Asigne a la propiedad `margins` el valor `2,2,2,2`.

**Nota** Si no se puede ver el texto completo de los botones, ensánchelos primero seleccionando `this` en el árbol de componentes y, a continuación, `VolverAlInicio`, arrastrando hacia el centro para ensanchar el resto de los paneles.

Por último, deje un pequeño espacio entre los botones en el panel `left_toolbar`. Esto se logra cambiando la separación horizontal en el gestor de diseño. Recuerde que el primer elemento de los contenedores en el árbol de componentes es su gestor de diseño.

- 1 Seleccione el objeto `GridLayout` para `barradeherramientas_izq`.
- 2 En el Inspector, cambie el valor de la propiedad `hgap` a 2 y, a continuación, pulse *Intro*.
- 3 Guarde su trabajo y ejecute la aplicación. Observe el espacio añadido entre los botones.

Ahora su diseño debe ofrecer el siguiente aspecto:



**Optativo:** si los botones no son iguales durante la ejecución y durante el diseño, es posible que esto se deba a que el aspecto elegido en JBuilder no coincide con el aspecto predeterminado del sistema. Si se cambia el aspecto de JBuilder (Herramientas|Preferencias), se cambia para el IDE de JBuilder, pero no para el aspecto durante la ejecución. Las capturas de pantalla de este tutorial utilizan Metal tanto para JBuilder como para el proceso de ejecución. Para obtener un aspecto distinto al del sistema durante la ejecución, abra `Aplicacion1.java`, y cambie la siguiente línea de

```
UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
```

a una de las siguientes:

```
UIManager.setLookAndFeel("javax.swing.plaf.metal.MetalLookAndFeel");
UIManager.setLookAndFeel("com.sun.java.swing.plaf.motif.MotifLookAndFeel");
UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLookAndFeel");
```

**4** Guarde su trabajo y cierre la aplicación y el panel de mensajes.

A continuación, añada componentes en el panel del medio.

## Paso 7: Cómo añadir componentes al panel central

El panel central va a contener un panel de desplazamiento y un área de edición de texto. En primer lugar, debe cambiar el gestor de diseño a `BorderLayout`. Es recomendable elegir este diseño si se colocan cinco o menos componentes, y si se desea que un componente ocupe por completo el diseño. `BorderLayout` cuenta con cinco áreas: Center, North, South, East, y West, donde Center ocupa el mayor espacio. En este ejemplo, desea que el panel de desplazamiento y el área de texto ocupan por completo el panel Central con una restricción de Center.

**1** Seleccione el panel `Central` en el árbol de componentes y cambie su gestor de layout a `BorderLayout`.



**2** En la paleta de componentes, seleccione la pestaña Contenedores Swing.

**3** Seleccione el componente `JScrollPane` y arrástrelo al diseñador. `JScrollPane` se utiliza para mostrar un componente, como por ejemplo el área de texto,

que es demasiado grande para mostrarse o que cambia de forma dinámica. `JScrollPane` debe ocupar por completo el panel `Central`. Si no cambia, asigne a la propiedad `constraints` de `JScrollPane` el valor `Center` en el Inspector.

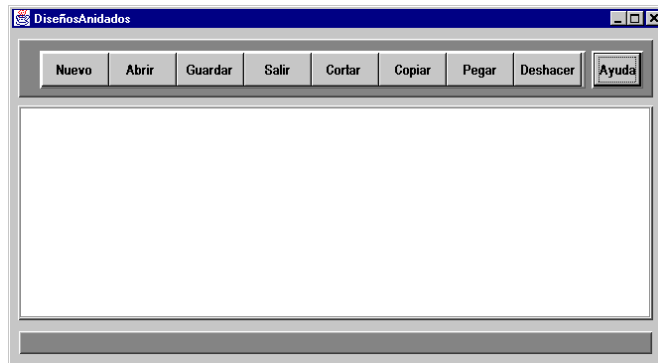


#### Nota

Los componentes Swing con una propiedad `text` tienen un valor de texto por defecto introducido en la propiedad `text`. Puede eliminarlo si resalta el texto mostrado como valor de la propiedad `text` y, a continuación, pulsa *Borrar* y después *Intro*.

- 4 Pulse la pestaña `Swing` de la paleta de componentes y seleccione el componente `JTextArea`. Suéltelo en `JScrollPane`, en el diseñador o en el árbol de componentes. Debe ocupar completamente `JScrollPane`.
- 5 Guarde otra vez el trabajo y ejecute la aplicación para ver la apariencia de la interfaz de usuario.

Éste es el aspecto que debe presentar ahora la interfaz de usuario:



- 6 Salga de la aplicación y cierre el panel de mensajes.
- En el siguiente paso se verá cómo crear la barra de estado.

## Paso 8: Creación de una barra de estado

Ahora se va a trabajar en el último panel del diseño de la interfaz de usuario, la barra de estado. Aunque este área, como las demás, no va a funcionar de forma perfecta, puede hacer que tenga la apariencia de una barra de estado de una interfaz de usuario. Una vez que haya realizado esto, la barra de estado debe tener un aspecto semejante a éste:



Cree el panel `BarraDeEstado` del siguiente modo:

- 1 Cambie el diseño del panel `BarraDeEstado` a `GridLayout`. Al añadir las etiquetas, éstas tienen el mismo tamaño, como los botones de la barra de herramientas.

**Label**

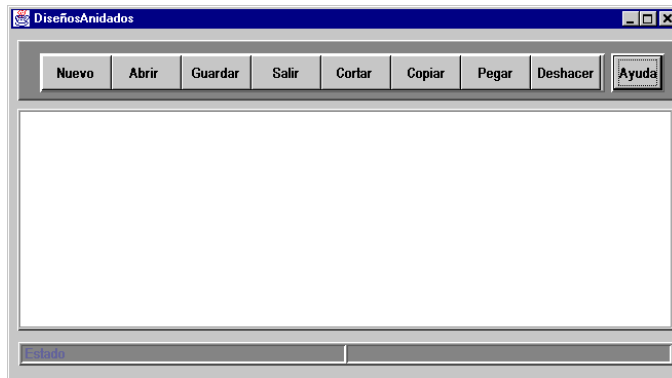
- 2 Añada dos componentes Swing `JLabel` al panel `statusbar`, tal y como se muestra. Asegúrese de que están contenidos en el panel `BarraDeEstado`.
- 3 Seleccione las dos etiquetas y cambie sus propiedades `border` a `LoweredBevel`, para conseguir una apariencia tridimensional como en otras muchas barras de estado.
- 4 Seleccione el objeto `GridLayout` del panel `statusbar`, en el árbol de componentes, y asigne a la propiedad `hgap` el valor 2, para ensanchar el área elevada entre las barras de estado. Esta operación es solamente estética y no es fundamental.

**Importante**

En JBuilder no es posible editar las propiedades `layout` de un <Diseño por defecto>. Si desea modificar las propiedades del gestor de diseño del contenedor, debe especificar un gestor de diseño explícito. Así, se puede tener acceso a sus propiedades en el Inspector.

- 5 Seleccione la etiqueta izquierda y cambie el valor por defecto de su propiedad `text` a `Status`.
- 6 Seleccione la etiqueta derecha y elimine el valor por defecto de su propiedad `text` y, a continuación, pulse *Intro*.
- 7 Guarde su trabajo y ejecute la aplicación.

Las barras de herramientas se asemejarán a las siguientes:



- 8 Salga de la aplicación y cierre el panel de mensajes.

En este momento, ha completado la primera fase del trabajo de diseño de la interfaz de usuario, tras añadir todos los componentes a los contenedores, comenzando con los exteriores, de mayor tamaño, y continuando con los más pequeños, definidos dentro de sus contenedores.

El siguiente paso consiste en convertir los paneles en otros diseños.

## Paso 9: Conversión a diseños portables

---

En este paso, empezará a trabajar en sentido inverso al habitual, convirtiendo los paneles en diseños más portables. Recuerde que no debe dejar nada en

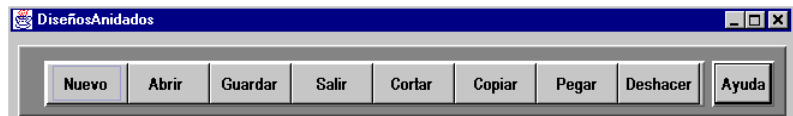


`XYLayout` o `null` debido al posicionamiento absoluto de sus componentes y a su falta de portabilidad.

Ahora, cambie el diseño por el del panel superior que contiene las barras de herramientas y alinee las barras a la izquierda:

- 1 Seleccione el panel superior y cambie la propiedad `layout` en el Inspector a `FlowLayout`. Los dos paneles en el panel superior, `barradeherramientas_izq` y `barradeherramientas_dcha`, se desplazan ahora de izquierda a derecha.
- 2 Seleccione el objeto `FlowLayout` del panel superior en el árbol de componentes y cambie la propiedad `alignement` a la izquierda. Normalmente las barras de herramientas están alineadas a la izquierda en el diseño de interfaces de usuario.
- 3 Guarde y ejecute su aplicación y fíjese cómo las barras de herramientas están alineadas a la izquierda en la interfaz de usuario.

En este momento, los paneles superiores deben parecerse a los siguientes:

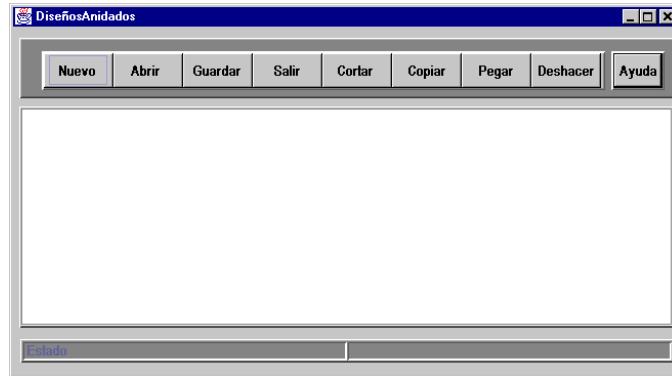


- 4 Salga de la aplicación y vuelva al diseñador.
- 5 Seleccione el panel superior en el diseñador y hágalo más estrecho que los botones para ver qué hace `FlowLayout` cuando el panel es más estrecho. Fíjese cómo el botón de Ayuda que está en el panel `barradeherramientas_dcha` se desplaza a la segunda fila. Este es el comportamiento del `FlowLayout`. Ajuste el panel hasta que ambas barras de herramientas estén en la fila superior.

Finalmente, cambie el diseño de `contentPane` a `BorderLayout`.

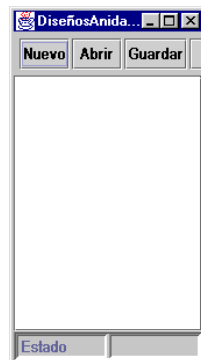
- 6 Seleccione `contentPane` en el árbol de componentes y cambie de `XYLayout` a `BorderLayout`. Debe asignarse el panel `VolverAInicio` a North (Norte), el panel Central a Center (Centro), y la `BarraDeEstado` a South (Sur). Si no es así, seleccione cada uno de los paneles y cambie sus propiedades `constraints` en el Inspector.
- 7 Guarde su trabajo y ejecute la aplicación.

Ahora, todos los componentes de la interfaz deben encontrarse en los lugares adecuados. Si su diseño es demasiado grande o pequeño, vuelva al diseñador de interfaz de usuario, seleccione `this` en el árbol de componentes y redimensione el marco.



Intente restaurar la ventana de aplicación ahora que `XYLayout` ha sido reemplazado por otros diseños más portables. Fijese en el comportamiento de `BorderLayout`: cuando hace la ventana más grande, el área central ocupa todo el espacio posible mientras que las demás áreas, en este caso North (la parte de arriba) y South (la parte de abajo), se expanden lo suficiente como para llenar las áreas restantes. North y South (la parte de arriba y la de abajo) sólo pueden expandirse horizontalmente para llenar el espacio que queda y no pueden agrandarse verticalmente. Esto se convierte en un problema cuando la ventana se hace más estrecha.

Ahora, haga la ventana muy estrecha y observe el panel superior que contiene las barras de herramientas. El panel superior no puede redimensionarse debido al número de botones que lo llenan, así que los botones se esconden.



Esto pasa porque los botones están en su tamaño mínimo para mostrar el texto y la parte superior de `BorderLayout` sólo puede estirarse horizontalmente y no verticalmente. Por lo tanto, los botones no se pueden ajustar. A través de este ejemplo, puede ver que el `BorderLayout` no es la mejor opción para diseños más complicados que contengan barras de herramientas. Es mejor

usarlo sólo cuando tenga varios componentes que necesiten rellenar Center, North, South, East y West (Centro, Norte, Sur, Este y Oeste). Por otro lado, `GridbagLayout`, en conjunción con otros administradores de diseño, es ideal para diseños de interfaz de usuario más complicados y merece la pena aprender a usarlo.

A continuación, diríjase al último paso, donde dará los últimos toques a su interfaz de usuario.

## Paso 10: Completar su diseño

Ya está hecho todo el trabajo. Sólo falta un poco de orden y limpieza. Si cambió el color de los paneles, ahora puede devolverlos a su color gris original.

- 1 Seleccione todos los paneles de su diseño usando la opción de selección múltiple del árbol de componentes.
- 2 En el Inspector, haga doble clic en la propiedad `background` y seleccione gris claro (valores RGB 192, 192, 192). No se preocupe de momento por el aspecto que presenta en el diseñador. Los bisels dejan mucho espacio en blanco. También le convendría seleccionar todos los botones y asignarles el mismo fondo que a los paneles.

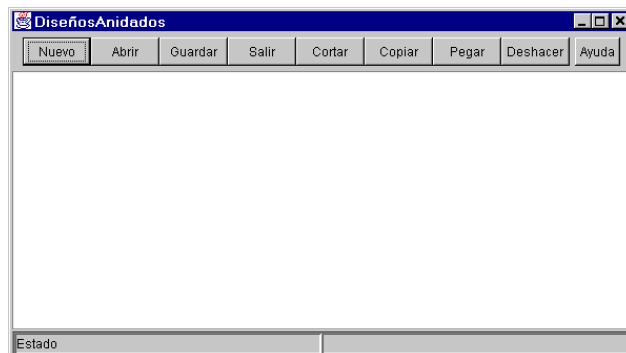
A continuación, elimine los bordes no deseados.

- 3 Seleccione los paneles del diseño cuyos bordes desea eliminar y asigne a la propiedad `border` el valor `<ninguno>`.

Elija los bordes que desee, basándose si lo prefiere en el aspecto general del diseño. Los bordes se pueden personalizar haciendo clic en el botón de puntos suspensivos que se encuentra a la derecha de la propiedad `border` para abrir el editor de propiedades de los bordes.

En este tutorial se han quitado todos los bordes excepto el borde `LoweredBevel` (biselado) de las etiquetas de la barra de estado.

Ahora, la interfaz de usuario está completa excepto unos toques de acabado que quizá deba añadir. Éste es el aspecto aproximado que debe tener su diseño final:



Esto es todo. A pesar de que la fase de aprendizaje parece lenta, una vez que se familiarice con los distintos diseños podrá planificarlos e implementarlos con mayor facilidad.

Es fácil comprender que el uso de varios niveles de paneles puede ser bastante tedioso y complicado, incluso con los diseños más fáciles. Tiene más sentido aprender el manejo de `GridBagLayout`. De este modo, los diseños de interfaz serán mucho más sencillos y fáciles de controlar. Se utilizarán diseños anidados, pero sólo de uno o dos niveles de profundidad.

`GridBagLayout` controla el resto del comportamiento del diseño.

Si desea más detalles acerca de `GridBagLayout`, consulte el [Capítulo 12](#), “[Tutorial: Creación de diseños GridBagLayout en JBuilder](#)”.

Para aprender a escribir código que responda a sucesos de usuario en sus aplicaciones, consulte el [Capítulo 10](#), “[Tutorial: Creación de un editor de texto en Java](#)”.

Con esto concluye el tutorial.

# Capítulo 12

## Tutorial: Creación de diseños GridBagLayout en JBuilder

Muestra cómo crear un contenedor de interfaz de usuario GridBagLayout UI utilizando las herramientas de diseño visual de JBuilder. El objetivo de este tutorial consiste en proporcionar al usuario un amplio conocimiento de la forma en que GridBagLayout funciona en JBuilder y demostrarle la forma de simplificar el diseño de GridBagLayout. Aunque esta información está dirigida al trabajo con JBuilder, gran parte se refiere al uso de GridBagLayout en general.

El tutorial es válido para todas las plataformas admitidas, ya que la funcionalidad de JBuilder y GridBagLayout es la misma. Las imágenes de este tutorial se han elaborado en Windows.

Este tutorial utiliza XYLayout para el diseño de la interfaz de usuario. Si dispone de JBuilder Personal, sustituya XYLayout por diseño null.

Este tutorial supone que usted está familiarizado con Java y con el IDE (Entorno integrado de desarrollo) de JBuilder. Para obtener más información sobre los constructores, consulte *Procedimientos iniciales con Java*. Para obtener más información sobre el IDE de JBuilder IDE, consulte “El entorno de JBuilder” en *Introducción a JBuilder*.

En este tutorial también se supone que conoce al menos GridBagLayout, aunque el tutorial se puede llevar a cabo sin comprender necesariamente cómo funciona.

## Consulte

- [“GridBagLayout” en la página 8-25](#) para obtener una introducción a los conceptos de GridBagLayout.
- [“Utilización de GridBagLayout” en la página 12-3](#) para obtener una introducción de GridBagLayout y GridBagConstraints.
- “Opciones de accesibilidad” en Sugerencias de JBuilder, que contiene sugerencias sobre el uso de JBuilder a cargo de personas con discapacidades.
- [“Convenciones de la documentación” en la página 1-5](#) para obtener información sobre las convenciones utilizadas en este tutorial y en otra documentación de JBuilder.

El tutorial de GridBagLayout está dividido en tres apartados:

- [“Utilización de GridBagLayout” en la página 12-3](#)

En la primera parte del tutorial se explica en qué consisten el gestor GridBagLayout y qué son los objetos GridBagConstraints. Da una descripción detallada de cada una de las restricciones y explica cómo configurarlas en el diseñador visual de JBuilder. Aquí se explica también la complejidad de GridBagLayout y se enseñan las formas de simplificar su diseño por medio del diseñador.

- [“Creación de diseños GridBagLayout en JBuilder” en la página 12-17](#)

En la segunda parte se recorren los pasos necesarios para crear un cuadro de diálogo con GridBagLayout. Muestra cómo planificar la interfaz de usuario antes de empezar y da ejemplos de las diferencias de comportamiento del contenedor con diferentes diseños posibles.

- [“Sugerencias y técnicas” en la página 12-42](#)

La tercera parte es una recopilación de sugerencias y técnicas para el trabajo con GridBagLayout en JBuilder. En este apartado se examina el comportamiento de cada restricción, con ejemplos que muestran el resultado de su modificación en el diseñador. Se incluye un ejemplo de código generado por JBuilder como resultado de la creación del ejemplo de interfaz de usuario en la segunda parte. En esta parte también se explica cómo cambiar el código GridBagLayout para que se pueda diseñar visualmente en JBuilder.

La tercera parte incluye también:

- [“GridBagConstraints” en la página 12-65](#)

Una descripción general de las restricciones GridBagConstraints y sus valores.

- [“Ejemplos de restricciones de Peso” en la página 12-70](#)

Ejemplos ilustrados de restricciones de Peso aplicadas de distintas formas.

# Utilización de GridBagLayout

---

`GridBagLayout` es un gestor de diseño complejo que exige cierta cantidad de estudio y experiencia para entenderlo, pero que, una vez se domina, resulta enormemente útil. JBuilder ha ampliado las herramientas de diseño visual con ciertas características que facilitan en gran medida el diseño y el control de `GridBagLayout`; por ejemplo: un Editor de `GridBagConstraints`, una rejilla visual, funciones de edición mediante ratón y un menú contextual especial para los componentes del contenedor `GridBagLayout`.

Existen dos estrategias posibles para diseñar un `GridBagLayout` en el diseñador visual. Puede diseñarlo desde cero añadiendo componentes a un panel `GridBagLayout` o comenzar creando, en el diseñador, un prototipo del panel mediante otro diseño, como `XYLayout` o `null`, para convertirlo posteriormente en `GridBagLayout` cuando tenga todos los componentes organizados y dimensionados como desea. Este método puede agilizar considerablemente el proceso de diseño y en él se centra este tutorial.

Con independencia del método que emplee, es aconsejable que aproveche la anidación de paneles para agrupar los componentes. Emplee los paneles para definir las áreas importantes del contenedor `GridBagLayout`. De esta forma, simplifica notablemente el diseño de `GridBagLayout`, ya que tendrá menos celdas en la rejilla y menos componentes que necesiten `GridBagConstraints`.

Para obtener más información sobre los paneles anidados, consulte [“Paneles y diseños anidados” en la página 8-53](#).

Este tutorial utiliza `XYLayout`. Si lo prefiere sustituya el diseño `null` siempre que se mencione.

## Definición de GridBagLayout

---

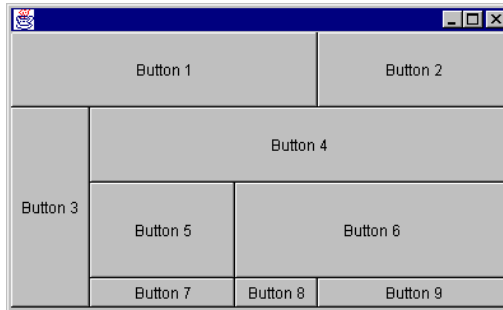
En Java, para crear una interfaz de usuario se añaden componentes a un objeto contenedor, como `Marco` o `Panel`, y se utiliza el gestor de diseño para controlar el tamaño y la posición de los objetos dentro del contenedor. Por defecto, todos los objetos contenedores tienen un objeto de gestor de diseño que controla su diseño.

`GridBagLayout` es un gestor de diseño muy flexible y avanzado que implementa la interfaz `LayoutManager2` y determina el lugar y la forma en que debe dar diseño a los objetos a partir de las restricciones `GridBagConstraints`. Coloca los componentes en horizontal y en vertical en una rejilla rectangular dinámica, pero proporciona más control en el tamaño y la posición de los componentes que `GridLayout` (donde todas las celdas de la rejilla tienen el mismo tamaño y cada una contiene un componente).

A diferencia de `GridLayout` los componentes de `GridBagLayout`, no tienen por qué tener el mismo tamaño y pueden abarcar varias celdas. Además, no es necesario que las columnas y las filas de la rejilla tengan la misma anchura y altura.

`GridBagLayout` controla la colocación de sus componentes a partir de los valores de sus objetos `GridBagConstraints`, los tamaños mínimos y el tamaño recomendado del contenedor.

**Figura 12.1** Ejemplo de `GridBagLayout`



La ventaja principal de `GridBagLayout`, tal y como se muestra en este ejemplo, es la capacidad de los componentes para aumentar o encoger de forma segura. Esta característica proporciona una mayor portabilidad de aplicación entre distintas plataformas, resoluciones y localización de productos en los que cambia la longitud de las cadenas.

En el ejemplo anterior, algunos botones ocupan sólo una celda (una fila y una columna) de la rejilla, pero otros abarcan varias celdas (se extienden por varias filas y columnas). Se puede ver el número exacto de celdas que ocupa cada componente en el diseñador, cuando se muestra la rejilla de un contenedor `GridBagLayout`. La diferencia entre el área de visualización de los componentes y el área que ocupan se explica en el siguiente tema de ayuda, “Área de visualización de los componentes”.

## Área de visualización de los componentes

---

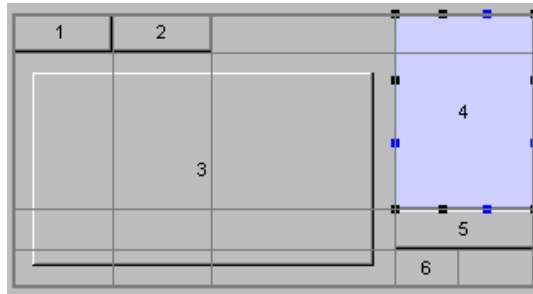
La definición de una celda de rejilla se hace del mismo modo en `GridBagLayout` que con `GridLayout`: una celda tiene una columna de ancho y una fila de alto. No obstante, a diferencia de `GridLayout` en donde todas las celdas tienen el mismo tamaño, las celdas de `GridBagLayout` pueden combinar diferentes alturas y anchuras, así como un componente puede ocupar más de una celda en sentido horizontal y vertical.

El área ocupada por un componente se conoce como su *área de visualización* y se especifica con los `GridBagConstraints` del componente `gridwidth` y `gridheight` (número de celdas horizontales y verticales en el área de visualización).

Por ejemplo, en el siguiente contenedor `GridBagLayout`, el componente 4 ocupa una celda (o columna) horizontalmente y dos celdas (filas) verticalmente. Por lo tanto, su área de visualización consta de dos celdas.



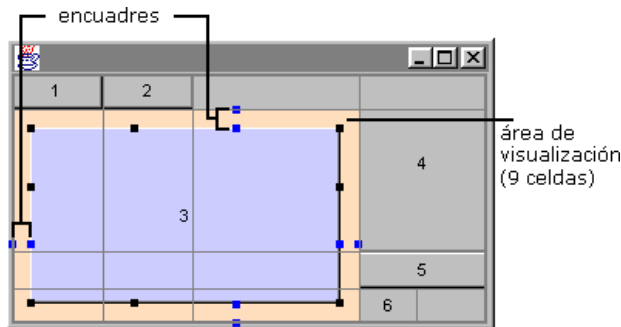
**Figura 12.2** Área de visualización: una celda en sentido horizontal, dos celdas en sentido vertical



Un componente puede ocupar por completo su área de visualización (como ocurre con el componente 4 en el ejemplo anterior) o puede ser más pequeño que ella.

Por ejemplo, en el siguiente contenedor `GridBagLayout`, el área de visualización del componente 3 consta de nueve celdas, tres en sentido horizontal y tres en sentido vertical. Sin embargo, el componente es más pequeño que el área de visualización porque tiene `encuadres` que crean una barrera entre los bordes del área de visualización y el componente.

**Figura 12.3** Área de visualización: tres celdas en sentido horizontal, tres celdas en sentido vertical

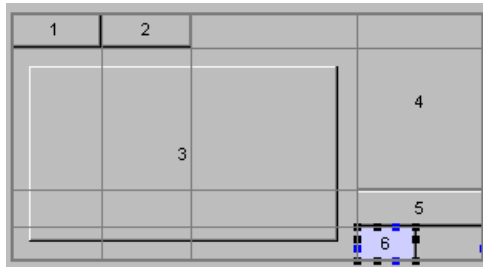


Aun cuando este componente tiene restricciones de `expansión` horizontales y verticales, dado que también tiene `encuadres` en los cuatro lados del componente (por los tiradores azules dobles en cada lado del área de visualización), éstas tienen prioridad sobre las restricciones de `expansión`. El resultado es que el componente sólo ocupa el área de visualización hasta los `encuadres`.

Si intenta lograr que el componente sea más grande que su área de visualización actual, `GridBagLayout` aumenta el tamaño de las celdas en el área de visualización para adaptarse al nuevo tamaño del componente y deja espacio para los `encuadres`.

Un componente también puede ser más pequeño que su área de visualización cuando no hay encuadres, como el componente “6” del siguiente ejemplo.

**Figura 12.4** Componente más pequeño que su área de visualización



Aun cuando el área de visualización tiene sólo una celda, no hay restricciones que extiendan el componente más allá de su tamaño mínimo. En este caso, la anchura del área de visualización queda determinada por los componentes más grandes que están por encima en la misma columna. El componente 6 se muestra en su tamaño mínimo y dado que es más pequeño que su área de visualización, está anclado en el borde izquierdo del área de visualización con una restricción de `ancla`.

Como se puede ver, `GridBagConstraints` desempeña un papel fundamental en `GridBagLayout`. Estas restricciones se tratan con detalle en el siguiente apartado dedicado a [“Definición de las restricciones `GridBagConstraints`”](#) y en [“Sugerencias y técnicas” en la página 12-42](#).

## Definición de las restricciones `GridBagConstraints`

`GridBagLayout` se sirve de un objeto `GridBagConstraints` para determinar la información relativa al diseño de los componentes incluidos en el contenedor `GridBagLayout`. Dado que no existe una relación unívoca entre los componentes del contenedor y el objeto `GridBagConstraints`, es necesario adaptar éste a cada uno de aquéllos.

`GridBagConstraints` controla:

- La posición absoluta o relativa de los componentes.
- El tamaño absoluto o relativo de los componentes.
- El número de celdas que ocupa cada componente.
- La forma en que se ocupa el espacio sin ocupar en el área de visualización de un componente.
- La cantidad de relleno interno y externo de cada componente.
- Qué cantidad de `Peso` se asigna a los componentes para controlar qué componentes utilizan una parte mayor del espacio disponible. Esto controla el comportamiento del componente cuando se cambia el tamaño del contenedor o se muestra la interfaz en distintas plataformas.

Los componentes `GridBagLayout` disponen de las siguientes restricciones:

- `ancla`
- `expansión`
- `gridx, gridy`
- `gridwidth, gridheight`
- `encuadres`
- `ipadx, ipady`
- `weightx, weighty`

### Consulte

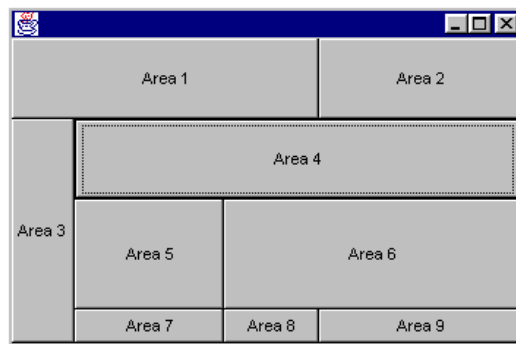
- `java.awt.GridBagConstraints.html` en <http://java.sun.com/j2se/1.3/docs/api/java/awt/GridBagConstraints.html>.
- `java.awt.GridBagLayout.html` en <http://java.sun.com/j2se/1.3/docs/api/java/awt/GridBagLayout.html>.

## Dificultades de GridBagLayout

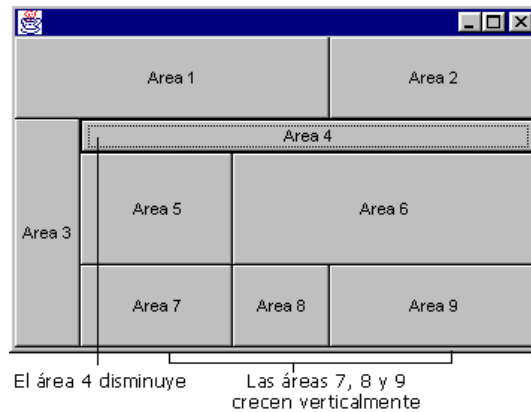
Cuando se empiezan a modificar las restricciones de un diseño `GridBagLayout` es frecuente que se obtengan resultados inesperados que pueden parecer incomprensibles. La mayor dificultad de la asignación de restricciones a componentes de contenedores `GridBagLayout` es la comprensión del efecto que tendrán las modificaciones efectuadas en un componente sobre los demás componentes de la rejilla. El comportamiento de las restricciones de un componente depende de los demás componentes del contenedor y sus restricciones. Por ejemplo, si se eliminan los valores de `peso` de un componente, la posición de los otros puede variar.

Los dos ejemplos siguientes muestran el efecto de cambiar el valor de la restricción `weighty` del Área 4 de 1.0 a 0.0. Note cómo la fila se contrae y la fila inferior se expande.

**Figura 12.5** El valor de la restricción `weighty` de Área 4 es 1,0



**Figura 12.6** El valor de la restricción weighty de Área 4 cambiado a 0,0



JBuilder reduce el tiempo de aprendizaje de `GridBagLayout` al mostrarle el efecto de los cambios inmediatamente en una superficie de diseño visual.

## Ventajas de GridBagLayout

---

`GridBagLayout` proporciona un control absoluto del comportamiento de los componentes y la forma en que se muestran cuando se cambia el tamaño del contenedor o se presenta en plataformas de otro tipo. De esta forma se garantiza que la aplicación distribuida tendrá el aspecto y el comportamiento apropiados en todas las plataformas aceptadas.

La mayoría de los manuales y tutoriales evitan tratar en profundidad `GridBagLayout` y en muchos se desaconseja su uso. Es posible realizar gran parte del trabajo de diseño de la interfaz de usuario con una combinación de otros diseños.

Si ha intentado utilizar `GridBagLayout` alguna vez, habrá descubierto que es muy complejo y de difícil utilización al principio. Si pretende que el diseño funcione correctamente tal y como se desea, hay que aplicarse a la aburrida tarea de comprobar y modificar errores de las restricciones en el código y, a continuación, compilar y ejecutar para ver si funciona. Hasta que se entiende por completo el comportamiento de las distintas restricciones y el efecto que ejercen sus modificaciones en el diseño, `GridBagLayout` resulta muy difícil de utilizar.

Como ocurre con todas las situaciones complejas, la forma más sencilla de utilizar esta herramienta consiste en simplificarla.

## Simplificación de GridBagLayout

---

JBuilder proporciona una forma mucho más sencilla de efectuar el trabajo de diseño con `GridBagLayout`, por lo que incluso los principiantes en la programación con Java pueden utilizarlo. Mediante una combinación del

diseñador visual, `XYLayout` o `null` de JBuilder, y las extraordinarias funciones de conversión de diseño de JBuilder, todo el código de diseño inicial se genera automáticamente, con lo que se libera al programador de la mayor parte del tedioso trabajo de tanteo y le deja solamente los últimos ajustes.

Las siguientes sugerencias sirven para facilitar la comprensión de `GridBagLayout` tanto a los principiantes en Java como a programadores expertos, con el fin de acelerar su trabajo de diseño de interfaces:

- Comience por hacer un boceto en papel.
- Utilice paneles y diseños anidados.
- Utilice el diseñador de JBuilder.
- Cree una interfaz de prototipo en `XYLayout`.

## Comienzo mediante un boceto en papel

Empiece siempre el diseño de `GridBagLayout` en un papel. Tómese su tiempo para realizar un boceto del diseño final y decidir en qué parte es mejor incluir paneles anidados junto con otros diseños. Los paneles anidados son fundamentales, porque simplifican el diseño y dan control absoluto en la colocación de los componentes.

Por ejemplo, si desea colocar una barra de herramientas en el diseño de `GridBagLayout` utilice un panel `GridLayout` anidado para colocar los botones, en lugar de ponerlos directamente en el contenedor `GridBagLayout`. Intente configurar el diseño de modo que el contenedor `GridBagLayout` deba controlar un número mínimo de paneles y componentes.

Al principio, puede no ser evidente la importancia de planificar de antemano en papel. Pero, si comienza realizando un prototipo sin un plan, rápidamente descubre cuánto tiempo hubiera ahorrado si lo hubiera pensado lógicamente antes de comenzar. Finalmente, su conocimiento y habilidad con los distintos gestores de diseño serán lo suficientemente avanzados para que le permitan omitir este paso y comenzar con su prototipo directamente en el diseñador. Pero, al principio, valen la pena el tiempo y el esfuerzo adicionales que implica la planificación. Consulte [“Paso 1: Diseñar la estructura” en la página 12-18](#).

## Dificultades al añadir componentes tras la conversión a GridBagLayout

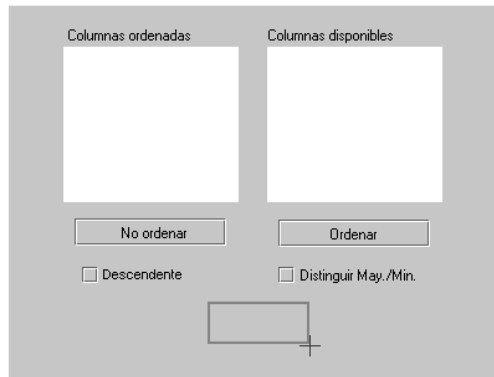
Si planear el diseño antes de empezar es siempre recomendable cada vez que se diseña una interfaz de usuario, es incluso más importante en el caso de `GridBagLayout`. Cuando se añaden o se mueven componentes de un contenedor `GridBagLayout` pueden ocurrir resultados inesperados. Si ya tiene pensados los requisitos del diseño antes de empezar, puede reducir al mínimo la cantidad de ajustes necesarios después de convertir el diseño de `XYLayout` en `GridBagLayout`. Consulte [“Creación de una interfaz de prototipo en XYLayout” en la página 12-15](#).

En la siguiente animación se demuestra lo que puede ocurrir cuando se añaden componentes a un panel después de cambiarlo de `XYLayout` a `GridBagLayout`. En este ejemplo se utiliza el mismo diseño que se va a crear en la segunda parte de este tutorial, [“Creación de diseños GridBagLayout en JBuilder” en la página 12-17](#), y en él se muestra cómo se ha añadido el

primero de los tres botones en la parte inferior del diseño. No obstante, en este caso, ninguno de los componentes del panel se ha agrupado en paneles anidados y la conversión a `GridBagLayout` se llevó a cabo antes de intentar añadir los botones en la parte inferior. Puede comprobar lo difícil que resulta controlar la colocación de un solo botón.

Puede observar también en las dos primeras imágenes que aparecen a continuación la dificultad a la que se enfrenta `GridBagLayout` para decidir dónde colocar el botón nuevo. A pesar de que se ha trazado en centro de la fila de los botones, `GridBagLayout` lo limita a la primera columna y la primera fila.

**Figura 12.7** Trazado del botón nuevo en el centro de la fila

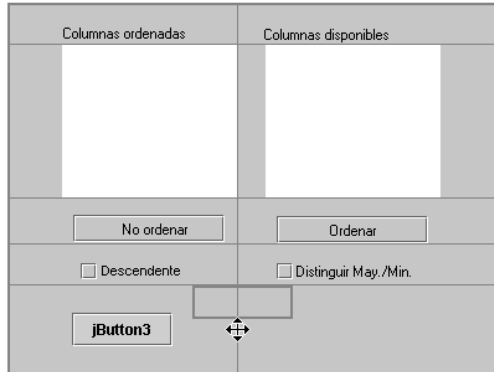
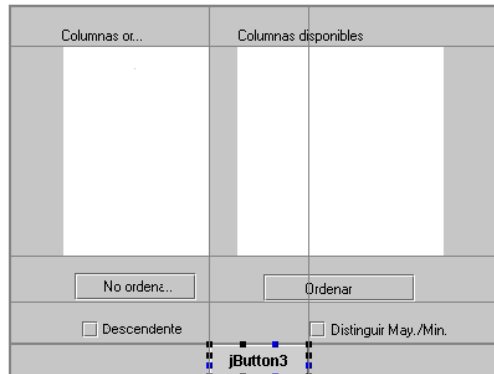


**Figura 12.8** `GridBagLayout` suelta el botón y lo coloca en la columna 1



En las dos imágenes siguientes, el botón se vuelve a arrastrar hasta el centro de la fila. `GridBagLayout` cambia el número de columnas del diseño mientras intenta colocar el botón en la nueva posición.

Observe que cuando se coloca el cursor directamente sobre la línea central entre las dos columnas, `GridBagLayout` coloca el botón en la parte superior. Si coloca el cursor en cualquier lugar del centro, el botón se colocará en una columna ya existente, no en una nueva.

**Figura 12.9** Trazado del botón de nuevo en el centro de la fila**Figura 12.10** GridBagLayout crea una columna central para el botón

Por desgracia, como los componentes que se han mostrado no están colocados en dos paneles diferentes, se alteran las restricciones y el tamaño de los demás componentes, ya que los de la derecha ocupan ahora dos columnas.

Por todo ello resulta evidente que si se hubiera añadido el botón mientras la interfaz de usuario estaba todavía en `XYLayout`, se podría haber colocado exactamente en el lugar y el modo en que se deseaba, sin afectar al lugar y tamaño recomendados de los demás componentes.

**Nota** Como se podrá comprobar más adelante, si se añade un panel en la parte inferior que contenga los botones, se consigue un mayor control al colocarlos en `GridBagLayout`.

### Consulte

- [“Creación de una interfaz de prototipo en XYLayout” en la página 12-15.](#)

## Utilización de paneles y diseños anidados

La mayoría de los diseños de interfaz de usuario en Java utilizan varios tipos de diseños para obtener los resultados deseados. A menudo se obtiene el mejor resultado anidando varios paneles con diseños diferentes en el contenedor principal. También es posible anidar unos paneles dentro de otros y obtener más control sobre la ubicación de los componentes. Creando un diseño compuesto y utilizando el gestor de diseño más adecuado para cada panel, es posible agrupar y ordenar los componentes de forma funcional y portable.

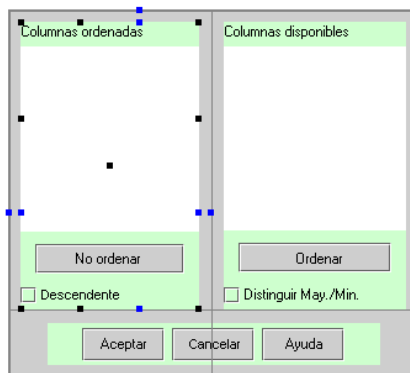
Pese a que `GridBagLayout` puede albergar una rejilla compleja, se comporta más eficazmente (y de forma más fiable) cuando se organizan los componentes en paneles de menor tamaño, anidados dentro del contenedor `GridBagLayout`. Estos paneles anidados pueden emplear otros diseños, incluido `GridBagLayout`, y contener a su vez otros paneles de componentes. Este método presenta varias ventajas:

- Ofrece un control más preciso sobre la posición y el tamaño de cada componente, ya que se emplean diseños más adecuados a zonas concretas, como las barras de herramientas.
- Reduce el número real de componentes controlados por `GridBagLayout`, lo que simplifica el diseño en gran medida.
- También reduce las posibilidades de obtener un comportamiento inesperado al modificar restricciones.
- Así como la necesidad de efectuar más modificaciones después de la conversión a `GridBagLayout`.

**Nota** Es mejor agrupar los componentes en paneles anidados si la anidación puede contribuir a que la rejilla quede dividida en menos celdas distribuidas uniformemente. Cuanto menor sea el número de componentes del contenedor `GridBagLayout` más fácil resultará controlar su colocación.

Por ejemplo, en la interfaz utilizada para este tutorial se pueden colocar en dos columnas todos los componentes excepto los tres botones de la parte inferior.

**Figura 12.11** Diseño de interfaces de usuario para el tutorial de `GridBagLayout`





La colocación de tres botones en la parte inferior del panel aumenta el número total de columnas, lo que dificulta la colocación de los otros componentes. Además, resulta más difícil conseguir que estos tres botones conserven el mismo tamaño en mitad del cuadro de diálogo cuando se cambia el tamaño del contenedor si son componentes separados del panel `GridBagLayout`.

Si se dejan los botones en el panel `GridBagLayout` cuando el contenedor se estira en horizontal, los botones de la parte inferior se alejan cada vez más, en vez de quedarse juntos en el centro.

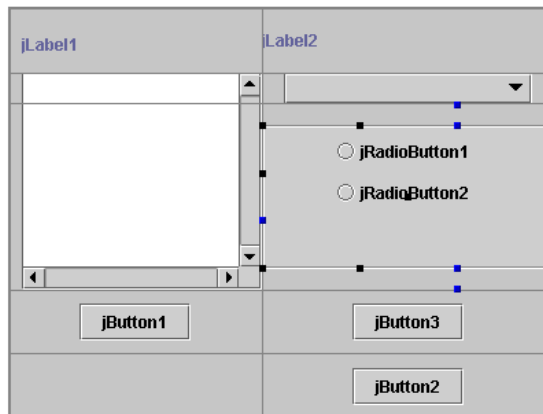
Si, por el contrario, agrupa los tres botones en un solo panel `GridLayout`, ese panel puede ocupar dos columnas del `GridBagLayout`. De este modo es posible tener dos columnas en total. Esto resulta mucho más fácil de gestionar para `GridBagLayout`. Además, si se colocaron los botones en el cuadro de diálogo tal y como estaba previsto al cambiar el tamaño del contenedor, se quedan juntas en el centro.

## Utilización del diseñador visual de JBuilder

Las herramientas de diseño visual de JBuilder logran que la utilización de `GridBagLayout` resulte mucho más fácil, rápida y segura:

- Se puede realizar el trabajo inicial en `XYLayout`, lo que permite controlar la posición y el tamaño exactos de los componentes. Cuando se termina el trabajo de diseño, se cambia a `GridBagLayout`. JBuilder efectúa todo el trabajo de cálculo de valores de restricciones de los componentes del diseño y genera el código automáticamente, con lo que simplifica y agiliza en gran medida todo el proceso.
- El diseñador muestra una rejilla de guía para realizar el trabajo de diseño en `GridBagLayout`. Esta rejilla aparece siempre que se hace clic en un componente de `GridBagLayout`, y muestra las celdas y las relaciones entre éstas y los componentes. Esta rejilla se puede activar y desactivar, y se oculta cuando se hace clic en un componente de un tipo de diseño distinto.

**Figura 12.12** Diseñador con la rejilla activada:



- Se puede utilizar el método de arrastrar y soltar para modificar las restricciones de los componentes en la superficie de diseño. Los componentes del contenedor `GridBagLayout` tienen tiradores para ajustar el tamaño, la posición, los Encuadres y el Tamaño adicional.

Cuando la rejilla está a la vista, esto permite ver con exactitud qué está ocurriendo en ella y en todos sus componentes cuando se arrastra o se estira uno de ellos. Los valores se actualizan inmediatamente en el código fuente y en el editor de la propiedad constraints.

Para activar la presentación de la rejilla en el diseñador, haga clic con el botón derecho del ratón en un componente del contenedor `GridBagLayout` y elija Mostrar rejilla.

### Nota

Estos tiradores se tratan en el apartado “Sugerencias y técnicas”, junto a las restricciones que los utilizan.

- Si lo prefiere, puede asignar y modificar todos los valores de constantes en el editor de la propiedad constraints, disponible en el menú contextual de la superficie de diseño, o desde el Inspector. El editor de la propiedad `GridBagConstraints` permanece abierto mientras se aplican restricciones a varios componentes `GridBagLayout` siempre que no se haga clic en componentes de otro tipo. Esto permite experimentar con los ajustes y ver el resultado sobre la marcha.

Por supuesto, si lo desea puede modificar los valores de constantes directamente en el código fuente, porque los cambios entre el diseñador y el editor siempre están sincronizados. De nuevo, el resultado tiene efecto inmediatamente en el diseñador.

- JBuilder proporciona varios niveles de deshacer, con lo que los experimentos resultan más fáciles y menos arriesgados. Si ocurre algo inesperado o no le gusta el cambio, puede volver a un estado anterior. Es inevitable que esto ocurra cuando se empieza a diseñar con `GridBagLayout`.
- JBuilder crea un objeto `GridBagConstraints` con un constructor que toma las once propiedades por cada objeto que se añade a un contenedor `GridBagLayout` mediante el diseñador:

```
public GridBagConstraints(int gridx,
                           int gridy,
                           int gridwidth,
                           int gridheight,
                           double weightx,
                           double weighty,
                           int anchor,
                           int fill,
                           Insets insets,
                           int ipadx,
                           int ipady)
```

Por ejemplo, si se añade un botón a un panel `GridBagLayout` llamado `jPanel1`, JBuilder genera el siguiente código:

```
jPanel1.add(jButton1, new GridBagConstraints(0, 0, 1, 1, 0.0, 0.0,
GridBagConstraints.CENTER, GridBagConstraints.NONE,
new Insets(0, 0, 0, 0), 0, 0));
```

## Creación de una interfaz de prototipo en XYLayout

La principal ventaja de crear las interfaces de prototipo en `XYLayout` de JBuilder es que los componentes se mantienen exactamente en la misma posición y con el mismo tamaño con que se han creado. En el menú contextual del componente también hay numerosas opciones disponibles para alinear varios componentes: izquierda, derecha, centrar, superior, inferior, centro, mismo tamaño en horizontal o en vertical e igual espacio en horizontal o en vertical.

`XYLayout` permite disponer los componentes exactamente de la forma que se desea que tengan y convertir el contenedor a `GridBagLayout`, para que JBuilder calcule automáticamente las celdas de la rejilla y los valores de las restricciones. JBuilder efectúa esta conversión correctamente, pero en muchos casos es conveniente ajustar algunas restricciones para obtener exactamente el comportamiento deseado. Normalmente, esto se debe a dos motivos:

- El proceso de conversión puede aplicar restricciones de Peso y Expansión no deseadas a los componentes, por lo que no se obtiene el resultado esperado.
- El número de celdas que calcula JBuilder suele ser mayor que el previsto. Por ejemplo, si intenta centrar algo que abarca varias celdas, como una barra de herramientas, puede que necesite ajustar el número de columnas o filas que abarca el componente (`gridwidth` o `gridheight`).

Aun así, la mayor parte de la tarea de escritura de código `GridBagLayout` se efectúa automáticamente, lo que agiliza en gran medida todo el proceso. Cuando vaya conociendo más cómo afectan las restricciones al comportamiento de los componentes y cómo JBuilder realiza la conversión de `XYLayout` a `GridBagLayout`, podrá anticipar qué paneles anidados son necesarios para que el diseño funcione bien.

A continuación se presentan los pasos básicos para esta estrategia de diseño:

- 1 Cree el contenedor que más tarde se convertirá a `GridBagLayout`. Puede ser el contenedor principal de la interfaz del usuario o un panel que se encuentra dentro de éste.
- 2 En caso necesario, cambie su diseño a `XYLayout`. La forma más fácil de hacerlo consiste en cambiar la propiedad `layout` en el Inspector.
- 3 Añada al contenedor todos los componentes mientras se encuentra en `XYLayout`. Utilice paneles anidados para reducir el número real de componentes controlados por `GridBagLayout`.
- 4 Acérquese todo lo posible al diseño final, para que la conversión a `GridBagLayout` sea más exacta. Aproveche las opciones de alineación del menú contextual de `XYLayout` para ajustar la colocación, el tamaño y la alineación de los componentes.

- Importante**
- 5** Cuando la interfaz esté prácticamente terminada, convierta el contenedor principal en `GridBagLayout`.
- Normalmente, cuando se utilizan paneles anidados en un diseño, los paneles interiores se convierten en primer lugar al formato deseado, y las conversiones sucesivas se van haciendo hacia fuera, hasta llegar al contenedor principal, que es el último en convertirse. Sin embargo, la estrategia es diferente con `GridBagLayout`.
- Cuando se convierte un contenedor de `XYLayout` a `GridBagLayout`, se deben dejar los paneles interiores de `XYLayout` hasta haber convertido el contenedor exterior en `GridBagLayout`. Esto se debe a que `JBuilder` determina durante el proceso de conversión el número de filas y columnas que se deben crear en la rejilla a partir de la anchura y la altura recomendadas del componente en el momento de la conversión. El proceso de conversión mantiene la anchura y altura recomendadas del panel `XYLayout`. Este proceso determina el número de columnas que hay que crear y los Encuadres necesarios, basándose en esas dimensiones.
- Por ejemplo, si intenta centrar un panel de barra de herramientas `GridLayout` en la parte inferior del contenedor `GridBagLayout`, como en la interfaz de ejemplo, si se convierte en primer lugar este panel en `GridLayout` se reducirá para ajustarse a los botones. En función de la anchura de los componentes del resto del contenedor `GridBagLayout`, es posible que la conversión no expanda el panel `GridLayout` para abarcar todas las columnas del contenedor.
- Si se deja el panel en `XYLayout` durante la conversión a `GridBagLayout` (extendido a lo ancho del contenedor), `JBuilder` detecta que es necesario centrarlo en el panel y ensancharlo para que abarque todas las columnas del contenedor. El panel de barra de herramientas se comporta correctamente dentro del contenedor `GridBagLayout` después de la conversión en `GridLayout`.
- 6** Convierta los paneles interiores a los diseños deseados.
- 7** Efectúe las modificaciones deseadas en las restricciones, para perfeccionar el diseño. Esto consiste sobre todo en modificar los Encuadres (por ejemplo, haciendo coincidir los Encuadres izquierdo y derecho de los componentes que se desea centrar en la celda) y asegurarse de que las restricciones de Expansión y Peso se han aplicado de la forma deseada.
- Si desea sugerencias sobre la forma de ajustar el diseño después de convertirlo a `GridBagLayout`, consulte la [“Sugerencias y técnicas” en la página 12-42](#).
- 8** Guarde y ejecute el programa. Cambie de tamaño el marco de formas distintas con el fin de comprobar si hay comportamientos imprevistos. En caso necesario, efectúe ajustes adicionales hasta que se sienta satisfecho con el resultado.

# Creación de diseños GridBagLayout en JBuilder

Este tutorial utiliza `XYLayout`. Si lo prefiere sustituya el diseño `null` siempre que se mencione.

## Acerca del diseño

En esta parte del tutorial de `GridBagLayout` se recorre paso a paso el diseño de un contenedor `GridBagLayout` en JBuilder. Se va a crear el siguiente cuadro de diálogo típico, que contiene varios controles y un grupo de tres botones centrados en la parte inferior.

**Figura 12.13** Interfaz de usuario del Tutorial `GridBagLayout`



Se ha elegido este diseño en concreto a causa de la complejidad que añade la colocación de un número impar de botones en la parte inferior. Además, es muy común encontrarse con esta situación.

Mientras se sigue esta parte del tutorial se debe tener en cuenta que, a causa de las diferencias en el trazado y la colocación de los componentes, es posible que el diseño no tenga el mismo aspecto que el del ejemplo ni se comporte de la misma forma. Sin embargo, será suficientemente parecido para obtener el mismo resultado.

- Los pasos 1 a 3 de este tutorial consisten en crear el diseño en el diseñador.
- En los pasos 4 a 6 se convertirá en `GridBagLayout`.
- El paso 7 explica la forma de ajustar las restricciones para retocar el diseño, así como los motivos de estas modificaciones.

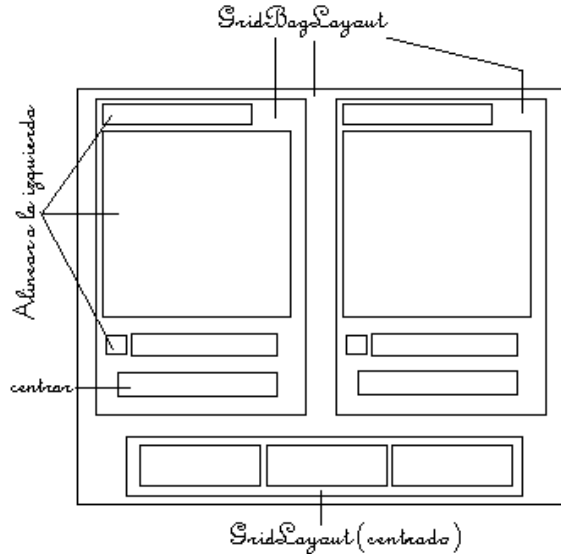
Tómese su tiempo y no tenga miedo de experimentar. El diseñador facilita la realización de pruebas con distintas opciones, cuyo efecto se ve en el acto. Guarde el archivo `Marco1.java` antes de introducir modificaciones, para poder deshacer los cambios.

## Paso 1: Diseñar la estructura

El primer paso en el diseño de la interfaz consiste en crear un esbozo de su estructura y contenido, en papel. Consulte [“Comienzo mediante un boceto en papel” en la página 12-9](#).

Esto ya se ha hecho anteriormente.

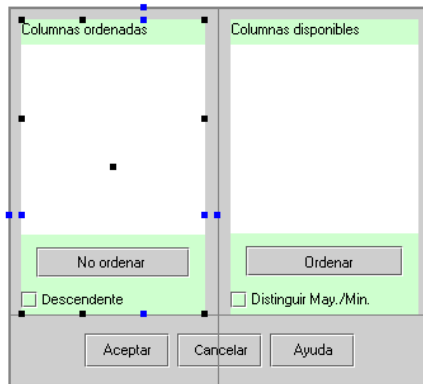
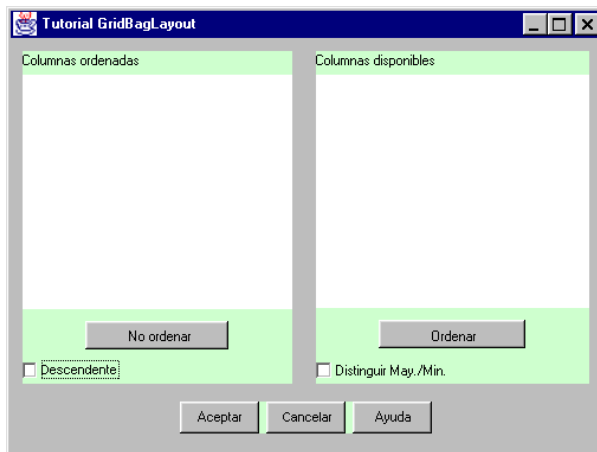
**Figura 12.14** Esbozo del diseño planeado



En el boceto se han agrupado los componentes en tres paneles que controlará el contenedor `GridBagLayout`: dos paneles del mismo tamaño que recogen los componentes de la parte principal de la interfaz y un panel inferior para los botones de aceptación, cancelación y ayuda. Esto se ha hecho por dos motivos:

- Los componentes de la parte superior se ajustan de forma conveniente en dos columnas. Si se agrupan en dos paneles, la conversión a `GridBagLayout` crea solamente dos columnas y el panel inferior ocupa estas dos columnas y se coloca en el centro del panel `GridBagLayout`.
- Este diseño reduce a tres el número total de componentes de `GridBagLayout`, lo que facilita el control del diseño.

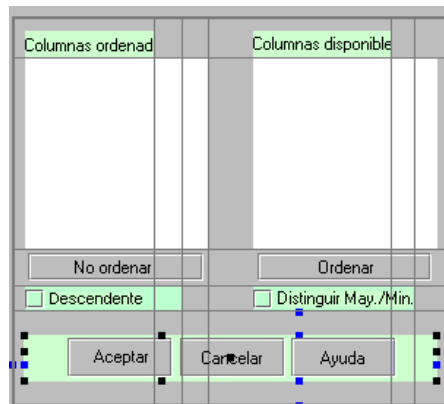
A continuación se demuestra la forma en que JBuilder gestionaría la conversión. Observe que la rejilla sólo muestra dos columnas y dos filas en el diseñador. (El color de fondo de los paneles se ha cambiado en este ejemplo para demostrar con más claridad la forma en que JBuilder determina la posición de las divisiones.)

**Figura 12.15** Conversión a GridBagLayout mediante el diseñador**Figura 12.16** Ejecución, antes de redimensionar**Figura 12.17** Ejecución, después de redimensionar

También se puede utilizar un solo panel para los tres botones de la parte inferior, de forma que `GridBagLayout` controle los demás componentes por separado, en lugar de anidarlos en paneles. Esto puede funcionar si se pone especial atención en asignar a todos los componentes de la parte superior la misma anchura en los dos lados. Sin embargo, `GridBagLayout` crea más columnas según el lugar donde termine cada componente, lo que complica mucho más el diseño y lo hace más difícil de controlar.

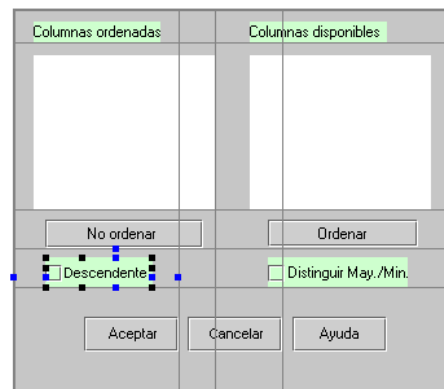
Observe que la conversión ha creado seis columnas en esta ocasión. (De nuevo, se ha cambiado el color de fondo de los componentes transparentes para que se aprecie fácilmente dónde terminan y demostrar cómo se calculan las columnas y filas.)

**Figura 12.18** Resultados de la conversión sin paneles anidados en las columnas superiores



Por último, si no se utilizan paneles interiores para agrupar los componentes, el trabajo de `GridBagLayout` es mucho más difícil. Además de crear más celdas aún, determina cuántas utiliza cada componente para un área de visualización y qué componentes tienen restricciones de Peso. Cuantos más componentes haya en el diseño, más posibilidades habrá de interpretar de forma incorrecta la idea original.

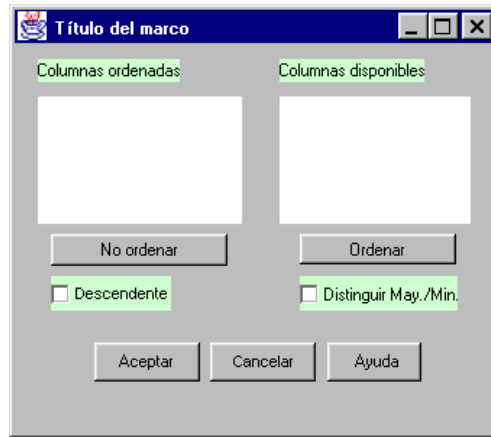
**Figura 12.19** Resultados de la conversión sin paneles anidados internos





A continuación se muestra cómo se comporta un panel `GridBagLayout` sin paneles internos al redimensionarlo en tiempo de ejecución:

**Figura 12.20** Panel `GridBagLayout` sin paneles internos antes de redimensionar



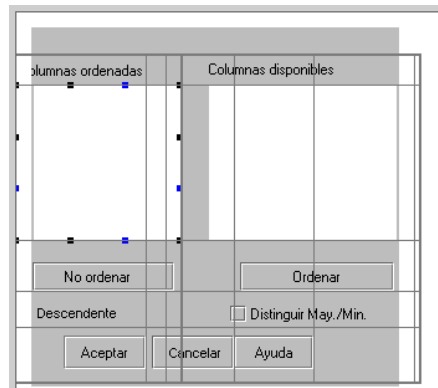
**Figura 12.21** Panel `GridBagLayout` sin paneles internos después de redimensionar



Tal y como se puede comprobar, los componentes no hacen lo que se desea. Si no se agrupan en paneles, es prácticamente imposible controlar su colocación y tamaño al redimensionarlos.

Además, si no se crean paneles hay que alinear los componentes de forma homogénea, con mucha precisión, para evitar que la rejilla tenga el siguiente aspecto:

**Figura 12.22** Posibles resultados sin paneles internos y sin la alineación homogénea de componentes



Cuando la rejilla es mayor que el marco, esto puede indicar que GridBagLayout debe controlar demasiados objetos dispares, por lo que las decisiones que toma respecto a las restricciones de Peso (`gridwidth`, `gridheight` y `ancla`) hacen que el diseño requiera una rejilla que supera la capacidad del contenedor. Si esto ocurre, el área de presentación de los objetos sobrepasa el borde del Marco.

Si este problema se da durante la conversión en GridBagLayout es probable que no merezca la pena perder el tiempo intentando corregirlo. Sería necesario invertir muchas horas en la modificación de las limitaciones y ni siquiera estaría garantizado el resultado satisfactorio.

Es mejor deshacer la última acción (`Ctrl+Z`) para volver a XYLayout. Ajuste el diseño en XYLayout y vuelva a convertirlo.

Para obtener una mejor conversión a GridBagLayout, intente realizar estos cambios mientras se encuentra en XYLayout:

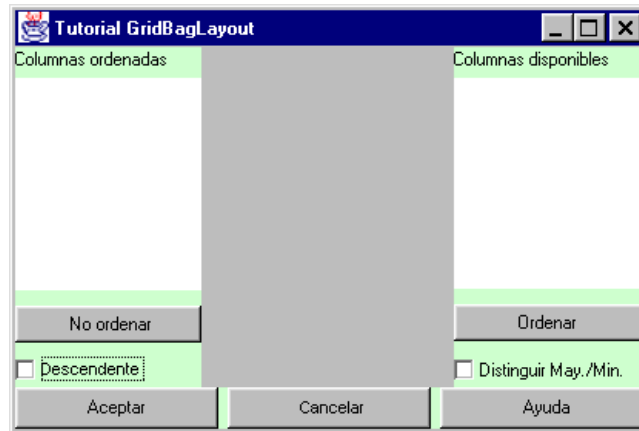
- Agrupe más componentes en paneles anidados, si es posible.
- Haga el diseño más simétrico. Iguale en la medida de lo posible los componentes iniciales y finales para reducir el número de celdas de la cuadrícula.

Si decide seguir adelante desde aquí, pruebe a ensanchar el marco lo suficiente para mostrar todas las celdas (sobre todo las columnas), para ver cuántas hay realmente en el diseño. De este modo puede modificar el número de celdas que abarca cada componente y reducir así el número total de celdas de la cuadrícula. Después puede corregir la posición de `gridx` y `gridy` de las áreas de visualización, así como el lugar que ocupan en ellas los componentes. En este caso también puede ser aconsejable eliminar todas las restricciones de Peso hasta fijar todas las demás.

Para evitar estas dificultades, utilice paneles anidados en su diseño.

Se podría pensar que si se utilizan cuatro paneles (tres dentro de un panel GridBagLayout) también se podría usar BorderLayout en el panel principal en lugar de GridBagLayout. A continuación, la imagen muestra la diferencia entre la forma de gestionar los componentes de BorderLayout y la de GridBagLayout.

**Figura 12.23** Resultados al usar BorderLayout en lugar de GridBagLayout



## Paso 2: Crear un proyecto para este tutorial

---

JBuilder utiliza proyectos para organizar los archivos relacionados en carpetas.

Para iniciar un nuevo proyecto:

- 1 Elija Archivo|Nuevo proyecto para iniciar el Asistente para proyectos.
- 2 Si lo desea, modifique la ruta de acceso y el nombre del proyecto antes de pulsar Finalizar.
- 3 Elija Archivo|Nuevo para abrir la galería de objetos y pulse la pestaña General. Haga clic en el icono Aplicación en la pestaña General, en la galería de objetos, para abrir el Asistente para aplicaciones.
- 4 Seleccione el icono Aplicación y haga doble clic en él o pulse *Intro*.
- 5 Acepte todos los valores por defecto y haga clic en Finalizar.
- 6 Guarde el proyecto: elija Archivo|Guardar proyecto.

## Paso 3: Añadir los componentes al contenedor

---

A continuación se va a crear el diseño de interfaz que utiliza tres paneles anidados dentro de un panel GridBagLayout principal para agrupar los componentes.

**Sugerencia** Dado que en ocasiones se debe trabajar con varios componentes, a continuación se repasará la forma de gestionar más de un componente en el diseñador.

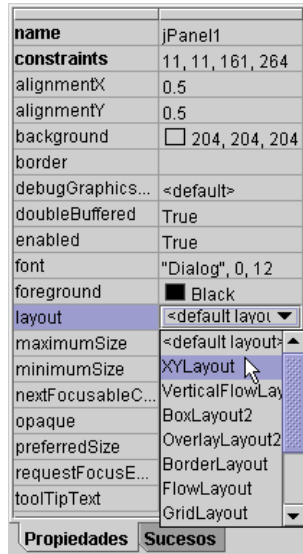
- Si tiene problemas a la hora de determinar dónde se encuentra en la superficie de diseño, lea el nombre del componente en la barra de estado. La barra de estado indica el componente sobre el que se encuentra la flecha, en la superficie de diseño.
- Si le resulta difícil seleccionar un componente en la superficie de diseño, selecciónelo en el árbol de componentes.
- Si desea seleccionar varios componentes a la vez en la superficie de diseño o en el árbol de componentes, haga clic sobre ellos con la tecla *Ctrl* pulsada.
- Cuando se modifica la alineación de varios componentes, el *primero* que se selecciona es el que utilizan los demás como referencia.
- Para retirar la selección múltiple, haga clic en cualquier componente que no esté seleccionado o en el árbol de componentes.
- Para abrir el menú contextual de un panel que contenga varios componentes, seleccione el panel, coloque el cursor en el tirador central hasta que se convierta en una flecha cuádruple y haga clic con el botón derecho del ratón. Si tiene dificultades para seleccionar el panel en el diseñador, selecciónelo en el árbol de componentes y coloque el cursor sobre el tirador central, en el diseñador.

**Figura 12.24** Selección del tirador central



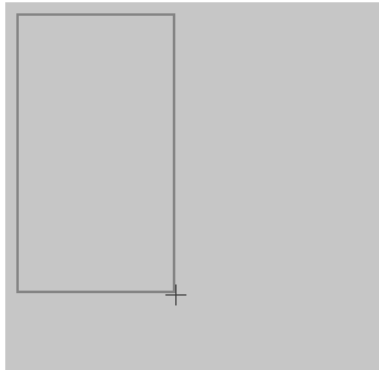
## Adición del panel principal al marco de interfaz

- 1 Seleccione el archivo de `marco` del proyecto (`Marco1.java`) y abra la pestaña Diseño para activar el diseñador de interfaces. El componente `this` es el supercontenedor de la interfaz de usuario. Su diseño por defecto, que no debe modificar aquí, es `BorderLayout`.
- 2 Pulse la pestaña Contenedores Swing en la paleta de componentes y seleccione el componente `JPanel`. Haga clic en el centro del marco para añadir este panel. De esta forma, el panel (`jPanel1`) se coloca en el centro y llena todo el marco.
- 3 Seleccione `jPanel1` en el árbol de componentes o en la superficie de diseño. Seleccione la propiedad Layout en el Inspector y cambie el diseño a `XYLayout`.

**Figura 12.25** Cambie el diseño en el Inspector

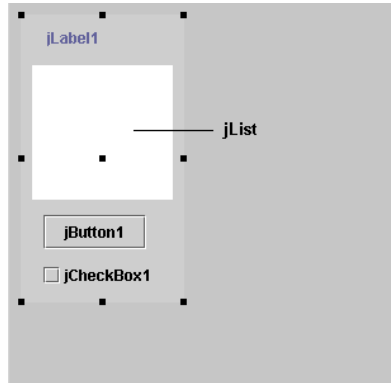
## Creación del panel izquierdo y adición de componentes

- 1 Añada un componente `jPanel` de la pestaña Contenedores Swing a la parte superior izquierda de `jPanel1`. Estírelo casi hasta la mitad en vertical y aproximadamente dos tercios hacia abajo, dejando algo de margen respecto al borde izquierdo de `jPanel1`, como se muestra a continuación. Este es `jPanel2`.

**Figura 12.26** GridBagLayout tutorial, `jPanel2`

- 2 Cambie el valor de `layout` de `jPanel2` a `XYLayout`. Puede cambiar el color de fondo de forma temporal para que le resulte más fácil verlo.
- 3 Añada los siguientes componentes en la pestaña Swing, empezando en la esquina superior izquierda de `jPanel2`: `jLabel`, `jList`, `Button` y `Checkbox`. Puede que necesite estirar el componente `jList` para hacerlo más grande después de añadirlo al panel.

**Figura 12.27** Tutorial GridBagLayout, componentes de JPanel2



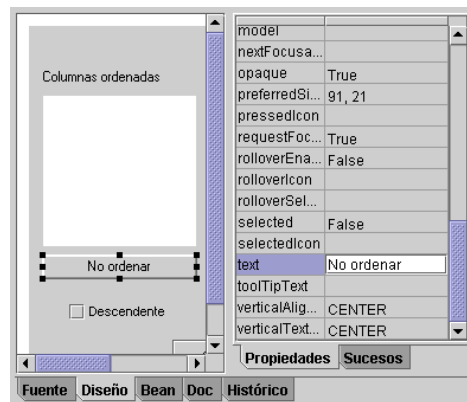
- 4 En el Inspector, asigne a la propiedad `text` de estos componentes los siguientes valores:

`jLabel1` = "Columnas ordenadas"

`jButton1` = "No ordenar"

`jCheckbox1` = "Descendente"

**Figura 12.28** Propiedad `text`



- 5 Iguale la fuente para `jLabel1`, `jButton1` y `jCheckbox1`:

- Pulse la tecla *Ctrl*, seleccione `jLabel1`, `jButton1` y `jCheckbox1` en el árbol de componentes.
- Haga clic en la propiedad `font` en el Inspector.
- Haga clic en el botón puntos suspensivos para mostrar el cuadro de diálogo Fuente.
- Si aún no lo ha hecho, cambie el valor de la propiedad `font` de 12 pts a 11 pts y pulse Aceptar.

**Figura 12.29** Cuadro de diálogo Fuente

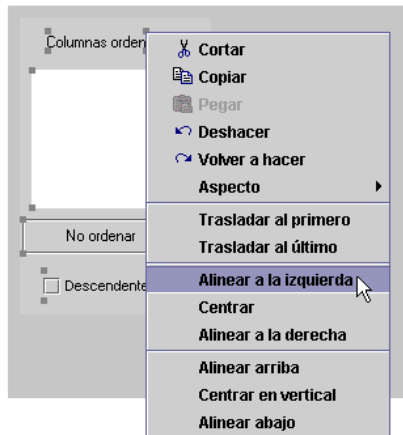
### Igualar el tamaño y la alineación de los componentes

- 1 Alinee los componentes por medio del menú contextual `XYLayout`. Mantenga pulsada la tecla *Ctrl* mientras hace clic en `jLabel1`, `jList1`, `jButton1` y `jCheckbox1` en la superficie de diseño.

#### Sugerencia

Cuando el cursor se encuentra sobre un componente en la superficie de diseño, su nombre aparece en la barra de estado.

- 2 Manteniendo el cursor sobre uno de los componentes seleccionados, haga clic con el botón derecho, en la superficie de diseño, para que aparezca el menú contextual y seleccione Alinear a la izquierda.

**Figura 12.30** Alinear a la izquierda

- 3 Ahora asigne a `jLabel1`, `jButton1` y `jCheckbox1` la misma anchura que a `jList1`:
  - a Mantenga pulsada la tecla *Ctrl* para seleccionar los cuatro componentes, empezando por `jList1`.
  - b Haga clic con el botón derecho sobre uno de los componentes seleccionados y escoja Mismo tamaño horizontal.

- Dado que `jList1` se ha seleccionado en primer lugar, los otros componentes del grupo adoptan su anchura.
- Nota** Aunque en realidad no es necesario que todos los componentes tengan la misma anchura, en este caso es preferible, porque simplifica la rejilla creada durante la conversión en `GridBagLayout`. De esta forma se reduce el número de columnas generadas para `GridBagLayout`, como se demostró en el [“Paso 1: Diseñar la estructura” en la página 12-18](#).
- Sugerencia** para obtener un resultado mejor es conveniente intentar igualar los componentes inicial y final cuando se colocan en `XYLayout` para convertirlos en `GridBagLayout`. Siempre que pueda, intente que los componentes tengan una estructura de cuadrícula homogénea y no escalonada.
- Cuando el final de un componente de una fila se solapa con el principio de otro componente de una fila distinta, `GridBagLayout` tiene dificultades para calcular el número de celdas que debe crear, cuáles deben tener Peso, cuántas celdas debe abarcar el componente y cómo se deben aplicar los Encuadres y el Tamaño adicional. Con frecuencia se obtiene un resultado incorrecto, muy difícil de arreglar.

## Creación del panel derecho y adición de componentes

A continuación se explica una forma fácil de crear el panel derecho, ya que es prácticamente idéntico al izquierdo.

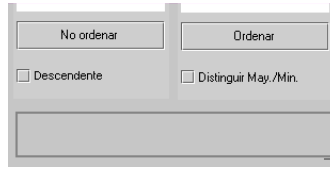
- 1 Haga clic con el botón derecho en `jPanel2` en el diseñador y elija Copiar en el menú contextual.
- 2 Coloque el cursor a la derecha de `jPanel2` en el diseñador, aproximadamente en el lugar en que desea que aparezca la esquina superior izquierda de `jPanel3`.
- 3 Haga clic con el botón derecho y seleccione Pegar. Se crea un panel llamado `jPanel3` que contiene `jLabel2`, `jList2`, `jButton2` y `jCheckbox2`.
- 4 Cambie como sigue los valores de la propiedad text de `jLabel2`, `jButton2` y `jCheckbox2`:  

```
jLabel2 = "Columnas disponibles"  
jButton2 = "Ordenar"  
jCheckbox2 = "Distinguir May./Min."
```
- 5 Ahora se deben alinear los dos paneles en vertical. Mantenga pulsada la tecla *Ctrl* y seleccione primero `jPanel2` y después `jPanel3`. Haga clic con el botón derecho sobre uno de los paneles seleccionados y pulse Alinear arriba en el menú contextual.

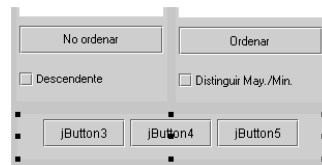
## Creación del panel inferior y adición de componentes

- 1 Añada el último panel, `jPanel4`. Arrastre un nuevo componente `jPanel` en la parte inferior, abarcando toda la anchura de `jPanel1`, e intente alinear su borde izquierdo con el borde izquierdo de `jPanel2`, y el borde derecho con el borde derecho de `jPanel3`.



**Figura 12.31** Trazado del panel inferior

- 2 Haga clic con el botón derecho en `jPanel4` y elija **Centrar**.
- 3 Arrastre tres componentes `jButton` a `jPanel4`.
- 4 Cambie el tipo de letra de estos botones para que coincida con los demás componentes.

**Figura 12.32** Tutorial GridBagLayout, añadir el panel de botones**Nota**

Dado que el diseño por defecto de `jPanel` es `FlowLayout`, esto se puede aprovechar provisionalmente para añadir los botones al panel. Cuando se sueltan botones en un panel `FlowLayout`, el gestor de diseño los centra a una distancia horizontal regular. Después puede pasar directamente a `GridLayout` sin necesidad de utilizar `XYLayout`.

- 5 Cambie la propiedad `text` de los tres botones para que muestren **Aceptar**, **Cancelar** y **Ayuda**, por este orden. El gestor de diseño ajusta la anchura de los botones para alojar el texto. No se preocupe por cambiarlos de tamaño o posición, porque cuando convierta este panel en `GridLayout` todos los botones tendrán el mismo tamaño, altura y anchura.

¡Enhorabuena! Ya ha terminado con el diseño inicial. Guarde el archivo antes de continuar.

## Paso 4: Convertir el panel exterior en GridBagLayout

---

Ahora puede convertir `jPanel1` en `GridBagLayout`.

- 1 En primer lugar, compruebe que los tres paneles están alineados correctamente y que hay suficiente espacio entre los paneles interiores y los bordes de `jPanel1`.

**Sugerencia**

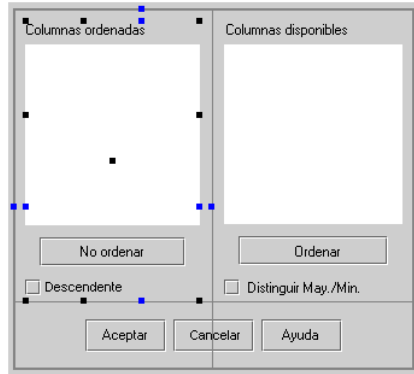
Para obtener un resultado mejor, no amontone los componentes en los contenedores, ya que esto podría impedir que coincidieran el tamaño mínimo colectivo o recomendable de los componentes y el tamaño mínimo o recomendable de los contenedores, incluido el `Marco`.

- 2 Seleccione `jPanel1` y cambie su propiedad `layout` a `GridBagLayout`. Esto tiene como resultado un cambio inapreciable en el diseño, sobre todo porque los

componentes se han agrupado en tres paneles que puede manejar fácilmente el gestor de GridBagLayout.

Si selecciona uno de los paneles de dentro del contenedor GridBagLayout, como `jPanel2`, debería ver una rejilla de sólo dos columnas y dos filas, como se muestra a continuación:

**Figura 12.33** Columnas después de la conversión



## Paso 5: Convertir los paneles exteriores en GridBagLayout

---

En este diseño da igual convertir en GridBagLayout el contenedor principal y después convertir estos dos paneles o seguir el orden inverso. Dado que GridBagLayout es el diseño de destino de `jPanel2` y `jPanel3`, el tamaño de los paneles no cambia con la conversión (a diferencia de lo que ocurre cuando se cambia a GridLayout o FlowLayout, que cambia el tamaño para alojar los componentes).

Para cambiar estos dos paneles a GridBagLayout:

- 1 Compruebe la limpieza de la alineación de los componentes, con el fin de reducir el número de columnas necesarias para la rejilla.
- 2 Seleccione `jPanel2` y `jPanel3` y cambie su diseño a GridBagLayout.

No se aprecia una gran diferencia en estos paneles después de convertirlos en GridBagLayout, aunque se puede perder parte del margen que rodea a los dos paneles.

No se preocupe si no todo es perfecto todavía. En el Paso 7 se harán los ajustes finales.

## Paso 6: Convertir el panel inferior en GridLayout

---

El último panel que se convierte es el inferior:

- 1 Seleccione `jPanel4` y asigne a su propiedad layout el valor GridLayout.

**Nota** Dado que `jPanel4` se ha dejado en `FlowLayout` durante la conversión de `jPanel1` en `GridBagLayout`, y dado que se ha extendido a lo ancho de `jPanel1`, debe haber permanecido centrado en dos columnas del contenedor después de la conversión a `GridBagLayout`. Con vistas a la conversión, JBuilder adopta para `jPanel1` el tamaño que tenía en `XYLayout` y asigna restricciones basadas en este tamaño en píxeles (por ejemplo, asignando a `gridwidth` un valor igual al número de columnas generadas durante la conversión, o el valor de `ipadx`).

Ahora, cuando se cambia el diseño a `GridLayout`, el panel conserva la anchura que tenía en `XYLayout`, pero los botones se expanden para llenar el panel, sin espacio entre ellos. Además, el panel es mayor de lo necesario. Estos detalles se arreglarán en el paso 7, cuando se efectúen los últimos ajustes en las restricciones de los componentes.

2 Guarde el archivo antes de empezar a modificarlo.

## Paso 7: Efectuar los ajustes finales

---

En lugar de mencionar las restricciones necesarias en este diseño, se examinarán los componentes por separado para mostrar el motivo de cada acción. De esta forma se proporciona una comprensión mejor de la forma en que las restricciones afectan a los componentes, a las celdas y a los otros componentes.

Se debe tener en cuenta que las restricciones sólo afectan a los componentes cuando cambian de tamaño si uno de ellos, como mínimo, tiene restricciones `weightx` o `weighty`. Sin embargo, no es muy normal crear contenedores `GridBagLayout` sin utilizar restricciones de Peso. Si ninguno de los componentes tiene restricciones de Peso, el cambio de tamaño no tiene ningún efecto sobre su colocación. Todos los componentes se acumulan, con el tamaño mínimo o recomendado, en el centro del panel `GridBagLayout`, y el espacio adicional que adquiere el contenedor con la Expansión sólo aumenta la distancia entre sus bordes y el de los componentes.

Otro aspecto importante es que existen varias combinaciones de restricciones que proporcionan el mismo resultado. Por ejemplo, para mantener el panel `GridLayout` centrado y con un tamaño homogéneo en la parte inferior del contenedor `GridBagLayout` se pueden utilizar encuadres, ancla, tamaño adicional o una combinación.

La mayoría de las modificaciones del resto de esta sección del tutorial se han realizado en el Editor de `GridBagConstraints`.

**Figura 12.34** Editor de GridBagConstraints

Para abrir el Editor de GridBagConstraints:

- 1 Seleccione un componente de un contenedor `GridBagLayout`. Si el componente es un panel que contiene otros componentes, selecciónelo y coloque el cursor en el tirador central, hasta que se convierta en una flecha cuádruple.

**Nota**

No puede abrir el editor de GridBagConstraints para `JPanel1` porque se trata de un componente incluido en un contenedor `BorderLayout`. Sólo puede abrir el editor de GridBagConstraints para componentes dentro de un contenedor `GridBagLayout`, tales como `JPanel2` o `JPanel3`.

- 2 Haga clic con el botón derecho y elija Restricciones en el menú contextual.

Después de terminar con el tutorial, experimente con esta interfaz. Abra el editor de GridBagConstraints y observe el resultado de distintos valores de restricciones.

A continuación, ajuste primero el panel `GridLayout` (`JPanel4`).

## Panel GridLayout

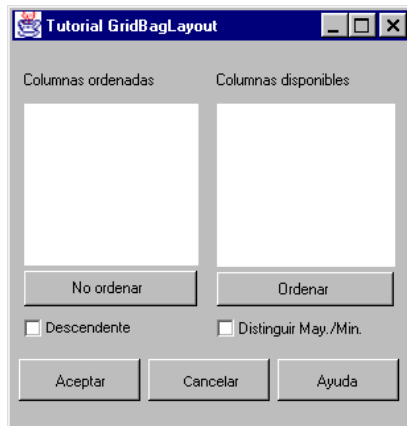
Cuando se convierte `JPanel4` en `GridLayout`, los botones que contiene se expanden completamente, de forma que no queda ningún espacio vacío. Introduzca un pequeño hueco entre los botones. Consiste simplemente en asignar un valor para la propiedad `hgap` del propio `GridLayout`.

Para cambiar el valor de espacio horizontal:

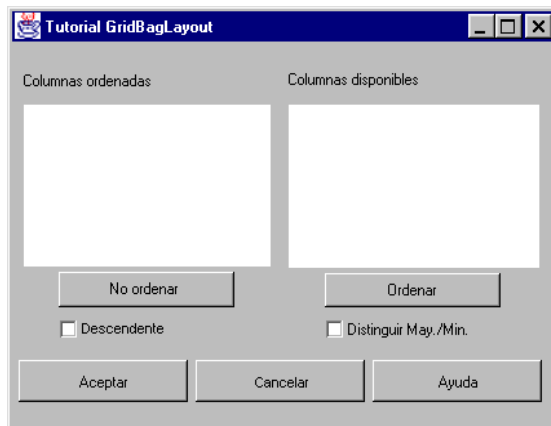
- 1 Haga clic en el nodo `gridLayout1` del árbol de componentes (justo debajo del nodo `jPanel4`).
- 2 Haga clic en la propiedad `hgap` de `gridLayout1`, en el Inspector, e introduzca un valor 6 para los píxeles, a continuación pulse *Intro*.

A continuación se debe reducir el tamaño de los botones. Dado que se trata de un componente `GridLayout`, los botones llenan la rejilla, y si se amplía el marco también se expanden los botones, como se muestra a continuación.

**Figura 12.35** GridLayout relleno antes de redimensionar



**Figura 12.36** GridLayout relleno después de redimensionar



Éste es el comportamiento habitual de `GridLayout`: los componentes que contiene llenan el panel, independientemente de su tamaño, aunque por supuesto se respetan los valores de espacio horizontal y vertical alrededor de los botones.

Por tanto, para controlar el tamaño de los botones en la rejilla es necesario restringir el tamaño del panel `GridLayout` mediante sus restricciones `GridBagConstraints`.

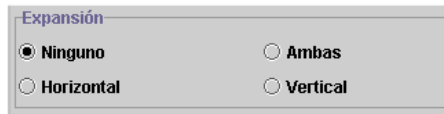
## Expansión

Seleccione `jPanel4` en el árbol de componentes y pulse el botón de puntos suspensivos para que la propiedad `constraints` del Inspector abra el editor de `GridBagConstraints`. Si se examinan los valores de restricción asignados a `jPanel4`, se puede observar que sus restricciones de expansión horizontales y verticales están activadas. Cuando ocurre esto, `GridBagLayout` expande el panel para llenar por completo el área de visualización, hasta llegar al borde de los `encuadres` definidos. Si no hay `Encuadres`, el panel llena el área de visualización hasta el borde de las celdas.

Éste no es el comportamiento deseado para este panel `GridLayout`, ya que lo que se desea es que los botones del panel tengan el tamaño preferente. Para conseguirlo, es necesario eliminar las restricciones de `Expansión` del panel.

Para eliminar las restricciones de `Expansión` vertical y horizontal a la vez, marque Ninguna como valor de restricción en el editor de `GridBagConstraints` y pulse Aceptar.

**Figura 12.37** Restricciones de expansión



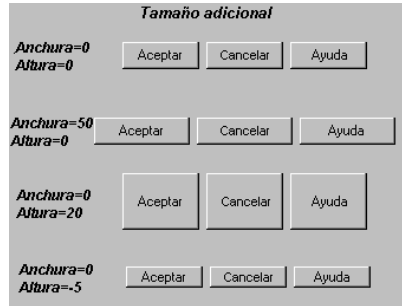
Alternativamente, para eliminar las restricciones de `Expansión` de un componente, puede hacer clic con el botón derecho sobre él, donde el cursor se convierta en una flecha de dos puntas, y seleccionar Eliminar relleno en el menú contextual.

Observe que esta acción no hace que los botones se encojan hasta su tamaño preferente. Necesita también ajustar los valores de tamaño adicional para conseguirlo.

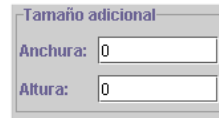
## Tamaño adicional

Tamaño adicional (`ipadx`, `ipady`) cambia el tamaño real del componente de panel añadiendo un número específico de píxeles a su anchura o altura mínima. El tamaño mínimo del panel `GridLayout` tiene el tamaño suficiente para mostrar los botones en su tamaño mínimo, además de la anchura establecida en la propiedad `hgap` de `GridLayout`. En el caso de los botones existe la propiedad `margin`, que también se incluye en el cálculo de su tamaño mínimo.

Si desea asignar a los botones y al panel el tamaño mínimo, no es necesario que haga nada más con la restricción Tamaño adicional. Si desea que los botones del panel `GridLayout` tengan un tamaño superior al mínimo, puede indicar cuántos píxeles se deben añadir. También se pueden reducir los botones por medio de valores negativos.

**Figura 12.38** Ejemplo de distintos valores de Tamaño adicional

En este tutorial, dado que el tamaño mínimo de los botones de `jPanel4` es aceptable, asigne cero a las dos restricciones de Tamaño adicional.

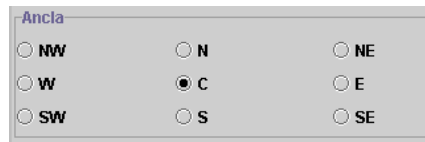
**Figura 12.39** Restricciones de relleno

**Nota** Puede eliminar los valores de relleno para varios componentes si selecciona varios componentes, hace clic con el botón derecho y elige Eliminar tamaño adicional.

Observe cómo los botones se encogen hasta su tamaño preferente.

## Ancla

Para asegurarse de que `jPanel4` permanece centrado en su área de visualización se debe asignar el valor Center a la restricción Ancla. Dado que el panel se ha centrado antes de la conversión en `GridBagLayout` es probable que la restricción Ancla ya tenga el valor Center. De todas formas, se debe abrir el editor de `GridBagConstraints` para comprobar que la configuración es la siguiente:

**Figura 12.40** Restricciones de ancla

Ahora que el valor de Expansión es Ninguno, no hay valores de relleno y el de Ancla es Center, cuando se cambia de tamaño el contenedor los botones permanecen centrados, con el tamaño reducido, cuando se cambia de tamaño el marco.

## Encuadres

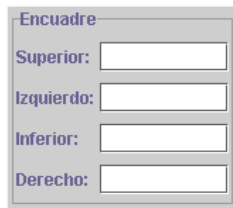
Los **Encuadres** definen la zona que no puede ocupar un componente, entre sus bordes y los de su área de visualización. Se pueden comparar con los márgenes de una página. El número de píxeles del **Encuadre** no varía cuando se cambia el tamaño del contenedor.

Si la **Expansión** del componente se encuentra activada, llena el área de visualización hasta los **Encuadres**. Cuando se cambia el tamaño del contenedor, el componente se expande hasta alcanzar los **Encuadres**.

En el caso de este panel **GridLayout**, dado que se ha desactivado la **Expansión** y se ha anclado en el centro del área de visualización, no sirve de nada añadir **Encuadres** a los bordes izquierdo y derecho del área de visualización. Lo único que se debe hacer aquí es comprobar que los **Encuadres** izquierdo y derecho tienen el mismo valor (cero).

Sin embargo, es necesario definir los **Encuadres** superior e inferior para que quede algo de espacio por encima y debajo de **jPanel4**. En el editor de restricciones **GridBagConstraints**, asígneles un valor de 15 píxeles.

**Figura 12.41** Restricciones de los Encuadres



Ya se ha arreglado el panel **GridLayout**. Ahora nos ocuparemos de los paneles superiores.

## Paneles superiores

Es necesario efectuar algunas tareas de limpieza y ajuste de restricciones en **jPanel2**, **jPanel3** y sus componentes.

### gridwidth y gridheight

Primero, abra el editor de **GridBagConstraints** para cada uno de los componentes de **jPanel1**, y en el área **Rejilla**, compruebe que cada componente sólo especifica un valor de 1 celda para los valores de anchura y altura (**gridwidth** y **gridheight**). Corríjalo si no es así.

**Nota** No ajuste los valores X o Y en el área **Rejilla**.



**Figura 12.42** gridwidth y gridheight

Rejilla

X: 1

Y: 1

Anchura: 1

Altura: 1

**Sugerencia**

Puede modificar a la vez los valores de restricciones para todos los componentes de un contenedor `GridBagLayout`. Mantenga pulsada la tecla *Ctrl* y seleccione todos los componentes, a continuación, haga clic con el botón derecho sobre uno de ellos y seleccione Restricciones. Cambie las que desee y haga clic en Aplicar o Aceptar.

**Expansión**

A continuación hay que hacer que todos los componentes (con excepción de los botones) y sus contenedores llenen el área de visualización. Igual que ocurría con el panel `GridLayout`, no es conveniente que los botones se expandan cuando se cambia el tamaño del marco.

De nuevo, y trabajando con un panel a la vez, seleccione cada componente de `Jpanel12` y `Jpanel13`, salvo el botón, y elija Ambas en la restricción `Expansión`. Haga clic con el botón derecho en el centro de los componentes y utilice el menú contextual.

Por último se debe hacer que los paneles llenen el área de visualización del contenedor `GridBagLayout` principal. Compruebe que `Jpanel12` y `Jpanel13` también tienen el valor Ambas en la restricción `Expansión`.

**Ancla**

Los dos paneles y sus componentes de etiqueta, lista y casilla de selección tienen el valor Ambas en la restricción `Expansión`. Como todos estos componentes rellenan su área de presentación tanto horizontal como verticalmente, las restricciones de `ancla` no tienen efecto. Sencillamente, no hay espacio dentro del área de presentación para que se muevan los componentes.

Si desea comprobarlo, pruebe a cambiar las restricciones `Ancla` de estos componentes, ejecutar el programa y cambiar de tamaño el contenedor. Comprobará que no ocurre ningún cambio.

Los únicos componentes de estos paneles sobre los que el `Ancla` puede tener efecto son los botones, que carecen de `Expansión`. Dado que no llenan su área de visualización, se pueden desplazar por ella. Para asegurarse de que los botones permanecen centrados en su área de visualización, asigne el valor Center a la restricción `Ancla`.

## Encuadres

Dado que los componentes de estos dos paneles tienen los mismos `Encuadres`, asegúrese de que presentan el mismo aspecto. Ninguno de los componentes necesita `Encuadres` izquierdo y derecho, ya que los paneles que los contienen son invisibles y controlan el espaciado del contenedor principal. Sin embargo, los `Encuadres` superior e inferior añaden algo de espacio entre los componentes de los paneles.

Configure los valores de `Encuadre` de los componentes de `jPanel2` y `jPanel3` en el área `Encuadre` del editor de `GridBagConstraints` de la siguiente forma:

Etiquetas	Superior = 0, Izquierdo = 0, Inferior = 4, Derecho =10
Listas	Superior = 0, Izquierdo = 0, Inferior = 0, Derecho =10
Botones	Superior = 10, Izquierdo = 10, Inferior = 0, Derecho =10
Casillas de selección	Superior = 6, Izquierdo = 0, Inferior = 0, Derecho = 0

Por último, para poner algo de espacio entre la parte superior y los lados de estos dos paneles y el contenedor exterior (`jPanel1`), asigne los siguientes valores a la restricción `Encuadres` de `jPanel2` y `jPanel3`:

Superior = 10, Izquierdo= 10, Inferior = 0, Derecho =10

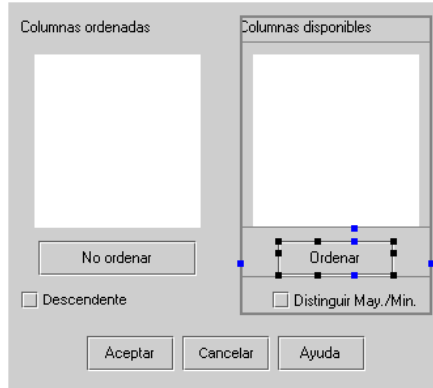
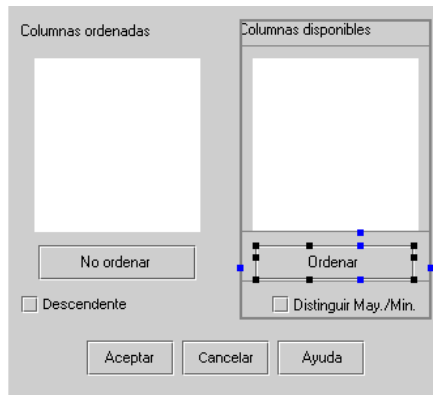
**Nota** No es necesario definir `Encuadres` para la parte inferior, ya que los `Encuadres` de la parte superior del panel `GridLayout` se ocupan de este espacio.

## ipadx, ipady

Un lugar de este diseño donde la restricción `ipadx` (tamaño adicional horizontal) resulta adecuada para controlar el tamaño del botón Añadir a la clasificación es `jPanel3`. Si se deja `ipadx` en cero para los dos botones, el botón Añadir a la clasificación se muestra en el tamaño mínimo, que no coincide con el tamaño del botón No ordenar.

Se puede utilizar el tamaño adicional horizontal (`ipadx`) para aumentar la anchura del botón y asignarle la misma que al otro:

- 1 Seleccione el botón No ordenar y abra el editor de `GridBagConstraints`. Asegúrese de que los valores de la altura y anchura de Tamaño adicional son cero y que las restricciones `Expansión` tienen todavía el valor Ninguno.
- 2 A continuación seleccione el botón Ordenar e introduzca un valor de 33 píxeles para la anchura del Tamaño adicional. Esta cifra es la utilizada en los botones de la interfaz de ejemplo. Si en su interfaz tiene un resultado distinto, experimente con distintos valores hasta dar con uno que funcione.

**Figura 12.43** Botón Ordenar sin ipadx**Figura 12.44** Botón Añadir a la clasificación con ipadx

**Sugerencia** Si el cambio de `ipadx` a cero no hace al botón No ordenar suficientemente grande como para mostrar todo el texto, puede seleccionar 'this' en el árbol de componentes para mostrar los tiradores del contenedor principal y arrastrar el tirador derecho para ensancharlo un poco.

Si se desea se puede reducir ligeramente la altura de estos botones respecto a su tamaño preferente, introduciendo un valor negativo en la altura del Tamaño adicional (en el ejemplo se ha usado -3). Por supuesto, esto no es necesario.

Ni los demás componentes de estos paneles ni los paneles mismos necesitan Tamaño adicional. Dado que `jPanel2` y `jPanel3` tienen restricciones de *Expansión*, redefinen cualquier Tamaño adicional asignado.

También se puede observar que los componentes de la lista utilizan `ipadx` e `ipady`, que en realidad no son necesarias. Dado que las listas de este ejemplo no contienen elementos, si se eliminan las restricciones `ipadx`, `ipady` y *Expansión*, las listas desaparecen. Su tamaño mínimo depende del número de elementos de la lista. Las restricciones `ipadx` e `ipady` se han añadido para establecer un tamaño determinado, con fines didácticos.

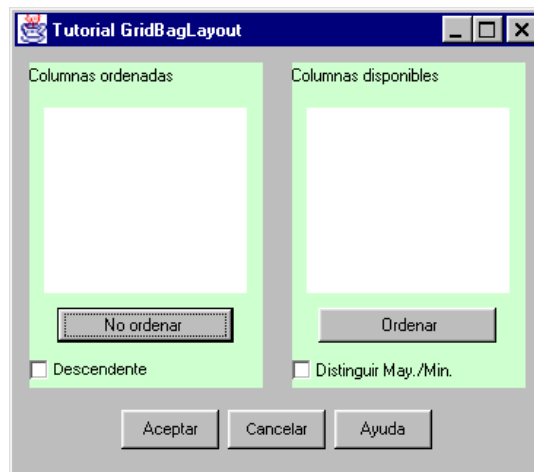
### weightx y weighty

Como se dijo anteriormente, si desea que los componentes de un contenedor cambien de tamaño según se cambia el del contenedor, tiene que asignar valores de restricción `weightx` y/o `weighty` al menos a un componente horizontal y verticalmente. Las restricciones de peso especifican cómo distribuir el espacio extra creado en el contenedor cuando éste es redimensionado.

Si desea que el tamaño de los componentes pueda aumentar, ha de establecer las restricciones de `Peso` y/o `Altura`, además de las restricciones de `Expansión` para una dirección (horizontal o vertical). Por ejemplo, si un componente tiene la restricción de peso (`weightx`), pero no la restricción `expansión` horizontal, el espacio adicional se asigna al tamaño adicional entre los bordes izquierdo y derecho del componente y los bordes de la celda. Aumenta la anchura de la celda sin variar el tamaño del componente. Si un componente tiene las restricciones de peso (`weightx` o `weighty`) y las restricciones de `expansión`, el espacio adicional se añade a la celda y el componente se amplía para ocupar ese incremento de tamaño en la dirección de la restricción `expansión` (en este caso, horizontal).

En primer lugar se extrajeron todos los valores de las restricciones de `Peso` definidos en la conversión. Éste fue el resultado:

**Figura 12.45** Sin restricciones de `Peso`, antes de redimensionar



**Figura 12.46** Sin restricciones de Peso, después de redimensionar

Observe que los componentes se acumulan en el centro.

Se ha determinado que los componentes que se quiere que crezcan son `jPanel2` y `jPanel3`, y los componentes de lista que se encuentran dentro de ambos. Se han practicado distintas combinaciones de restricciones de peso en estos componentes para ver los resultados. La siguiente lista le guiará a los resultados:

- “Restricciones de Peso en paneles y listas” en la página 12-70
- “Restricciones de Peso en paneles, pero no en listas” en la página 12-70
- “Restricciones de Peso en listas, pero no en paneles” en la página 12-70
- “Sólo restricciones de Peso (`weightx`) horizontales en los cuatro componentes” en la página 12-71
- “Sólo restricciones de Peso (`weighty`) verticales en los cuatro componentes” en la página 12-71
- “Sólo restricciones de Peso en un componente de panel y lista de la fila” en la página 12-71

Se buscan restricciones de peso en paneles y listas. Asigne a las restricciones de Peso de los cuatro componentes el valor 1.0 en el editor de `GridBagConstraints`: `jPanel2`, `jPanel3`, `jList1` y `jList2`.

## Conclusión

¡Enhorabuena! Ya ha completado este tutorial y entiende mejor el funcionamiento de `GridBagLayout` y las distintas restricciones `GridBagConstraints`.

En este ejercicio se puede apreciar claramente que para crear un diseño `GridBagLayout` con el aspecto y el comportamiento deseados es necesario hacer varias pruebas. JBuilder simplifica este proceso, ya que crea

rápidamente el código de `GridBagLayout` inicial y el programador sólo tiene que ajustarlo.

`GridBagLayout` es una herramienta muy útil, pero no resulta fácil de utilizar. Tenga en cuenta que, como en todo, cuanto más practique, cada vez le resultará más sencillo.

## Sugerencias y técnicas

---

Este tutorial utiliza `XYLayout`. Si lo prefiere sustituya el diseño `null` siempre que se mencione.

### Definición de restricciones individuales en el diseñador

---

#### Ancla

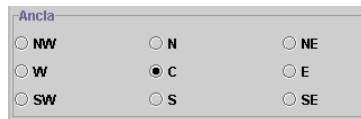
Existen dos formas de definir la restricción `Ancla` de un componente en el diseñador:

- Basta con hacer clic en el componente y arrastrarlo hacia la ubicación deseada, al borde de su área de visualización, de forma parecida a como se ancla una barra de herramientas móvil.

Por ejemplo, para anclar un botón a la esquina superior izquierda del área de visualización, haga clic en el centro del botón y arrástrelo hasta que la esquina superior izquierda toque la del área de visualización. Esto asigna a la restricción de `ancla` el valor `NO`, tanto en el editor de `GridBagConstraints` como en el código.

- Seleccione el valor de una restricción de `ancla` en el editor de `GridBagConstraints`.

**Figura 12.47** Restricciones de ancla en el Editor de `GridBagConstraints`



Para ello:

- a Seleccione el componente en la superficie de diseño.
- b Haga clic con el botón derecho sobre el componente o componentes y seleccione Restricciones, para ver el Editor de `GridBagConstraints`.
- c Haga clic en el valor deseado de la zona `Ancla` y pulse Aceptar.

De este modo cambiará el valor de la restricción en el código y el componente se colocará en su nuevo punto de anclaje, en la superficie de diseño.

**Nota** Si mueve el componente con el ratón se actualiza el valor de la restricción `ancla` en el editor `GridBagConstraints`. De igual forma, cuando se cambia el valor de la restricción en el editor de `GridBagConstraints`, el componente se coloca en su nueva posición en la superficie de diseño.

## Expansión

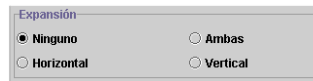
La manera más rápida de especificar las restricciones de `Expansión` en un componente consiste en utilizar el menú contextual de ese componente en la superficie de diseño.

- 1 Haga clic con el botón derecho sobre el componente en la superficie de diseño para abrir el menú contextual.
- 2 Realice una de las operaciones siguientes:
  - Seleccione `Expansión horizontal` para establecer el valor `HORIZONTAL`.
  - Seleccione `Expansión vertical` para establecer el valor `VERTICAL`.
  - Seleccione `Expansión horizontal` y `Expansión vertical` para asignarles el valor `AMBAS` (para ello es necesario abrir dos veces el menú contextual).
  - Seleccione `Eliminar relleno` para establecer el valor `ANULADA`.

También puede definir la restricción `expansión` en el Editor de `GridBagConstraints`.

- 1 Haga clic con el botón derecho sobre el componente, en la superficie de diseño, y seleccione `Restricciones`; esto mostrará el Editor de `GridBagConstraints`.
- 2 Seleccione el valor de restricción que desea en la zona `Expansión` y pulse `Aceptar`.

**Figura 12.48** Restricciones de expansión en el Editor de `GridBagConstraints`

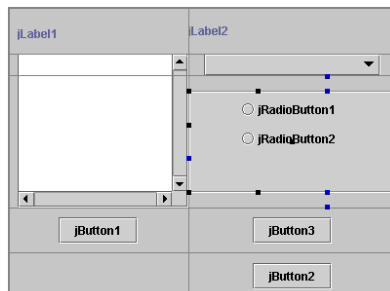


## Encuadres

La superficie de diseño muestra tiradores azules de redimensionamiento en los componentes seleccionados de `GridBagLayout` para señalar la ubicación y el tamaño de sus `encuadres`. El tamaño de los `Encuadres` se varía arrastrando los tiradores azules con el ratón.

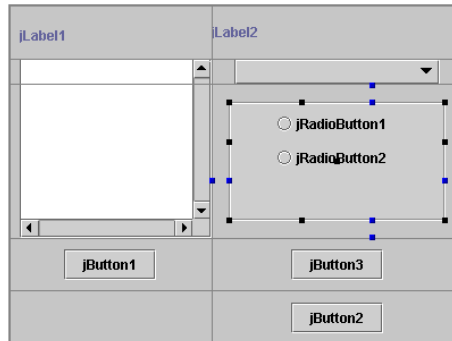
Si el valor de `Encuadre` es cero, sólo se ve un tirador azul en el lado correspondiente de la celda, como se muestra a continuación.

**Figura 12.49** Encuadres con valor cero en parte superior e inferior del componente



Si el valor de Encuadre es mayor que cero, la superficie de diseño muestra dos tiradores azules en el lado correspondiente: uno en el borde de la celda y otro en el borde del área de visualización. El tamaño del Encuadre es la distancia en píxeles entre los dos tiradores. Para redimensionar el Encuadre, basta con arrastrar uno de los tiradores.

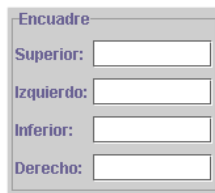
**Figura 12.50** Encuadres mayores que cero en el lado izquierdo y derecho del componente



Si desea un control más preciso de los valores de Encuadre, puede especificar el número de píxeles exacto mediante el Editor de GridBagConstraints.

- 1 Haga clic con el botón derecho sobre el componente, en la superficie de diseño, y seleccione Restricciones; esto mostrará el Editor de GridBagConstraints.
- 2 En la zona Encuadre, indique el número de píxeles para cada uno: superior, izquierdo, inferior o derecho.

**Figura 12.51** Encuadres en el Editor de GridBagConstraints



**Nota** Pese a que Encuadre admite valores negativos, no es recomendable emplearlos, ya que el componente se superpondría a los componentes contiguos.

## gridwidth, gridheight

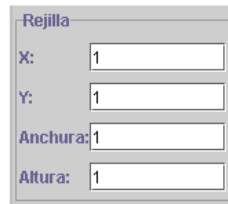
También puede definir los valores de las restricciones `gridwidth` y `gridheight` en el Editor de GridBagConstraints.

- 1 Haga clic con el botón derecho sobre el componente, en la superficie de diseño, y seleccione Restricciones; esto mostrará el Editor de GridBagConstraints.



- 2 En el área Rejilla, introduzca el valor de `gridwidth`, en el campo Anchura, o el de `gridheight`, en el campo Altura. Indique el número de celdas que ha de ocupar el componente en la celda o la columna.

**Figura 12.52** Restricciones `gridwidth` y `gridheight` en el Editor de `GridBagConstraints`



- Si desea que el valor sea de tipo `RELATIVO`, introduzca -1.
- Si desea que el valor sea de tipo `RESTO`, introduzca 0.

**Nota** JBuilder no asigna nunca el valor `RESTO` a `gridwidth` o `gridheight` durante la conversión a `GridBagLayout`.

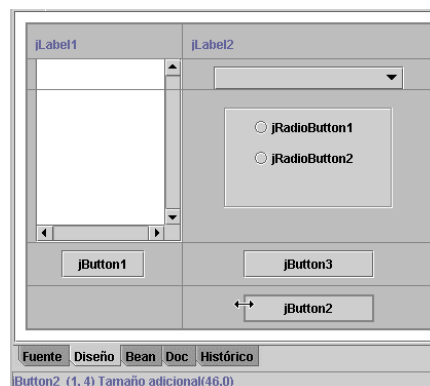
El valor de `gridwidth` y `gridheight` también puede cambiarse modificando el tamaño del componente hasta que ocupe las celdas contiguas (arrastrando un tirador de redimensionamiento negro hasta el otro lado del borde de la celda).

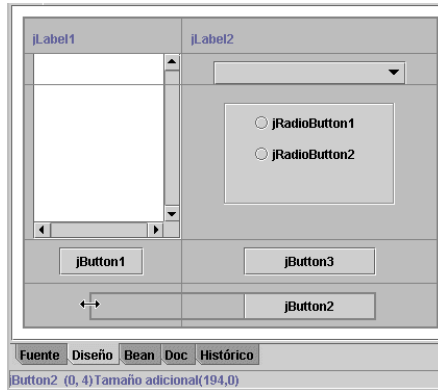
## ipadx, ipady

Es posible especificar el Tamaño adicional (`ipadx` o `ipady`) de un componente haciendo clic en los tiradores negros que aparecen en los bordes del componente y arrastrándolos con el ratón, para ampliarlo o reducirlo. Si se asigna al componente un tamaño mayor que el preferente aparece un valor en píxeles positivo. Si se asigna al componente un tamaño menor que el preferente aparece un valor en píxeles negativo.

Si arrastra el tirador hasta que supere el borde de la celda e invada la celda contigua, el componente ocupa ambas celdas (el valor de `gridwidth` o `gridheight` se incrementa en una celda).

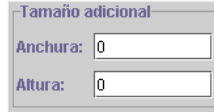
**Figura 12.53** Antes:  `jButton2 gridwidth = 1 celda, ipadx = 46`



**Figura 12.54** Después: jButton2 gridwidth = 2 celdas, ipadx aumentada a 194

Si desea controlar con más precisión los valores de `ipadx` e `ipady`, el Editor de GridBagConstraints le permitirá especificar el número exacto de píxeles.

- 1 Haga clic con el botón derecho sobre el componente, en la superficie de diseño, y seleccione Restricciones; esto mostrará el Editor de GridBagConstraints.
- 2 En el área Tamaño adicional, indique el número de píxeles deseado en los campos Anchura y Altura.

**Figura 12.55** Tamaño adicional en el Editor de GridBagConstraints

**Nota** Los valores negativos hacen que el componente sea menor que su tamaño recomendado y son perfectamente válidos.

Para eliminar rápidamente las restricciones `ipadx` e `ipady` (asignarle el valor cero), haga clic con el botón derecho sobre el componente, en la superficie de diseño, y elija Eliminar tamaño adicional. También puede seleccionar varios componentes y por el mismo procedimiento eliminar de una sola vez el tamaño adicional de todos ellos.

## gridx, gridy

Puede emplear el ratón para seleccionar la celda cuya esquina superior derecha ha de ocupar el componente. Basta hacer clic cerca de la esquina superior izquierda del componente y arrastrarlo hasta la celda deseada. Si desplaza componentes que ocupen varias celdas, cerciórese de hacer clic en la esquina superior izquierda del componente; si no lo hace así, obtendrá resultados no deseados. En ocasiones, debido a los valores de las restricciones del componente, cuando éste se desplaza a otra celda mediante el ratón se alteran otras restricciones (por ejemplo, puede cambiar el número de celdas ocupado por el componente).

El Editor de GridBagConstraints permite modificar con más precisión `gridx` y `gridy` sin alterar accidentalmente otras restricciones.

- 1 Haga clic con el botón derecho sobre el componente, en la superficie de diseño, y seleccione Restricciones; esto mostrará el Editor de GridBagConstraints.
- 2 En el área de rejilla, ingrese el número de columnas para el valor X o el número de filas para el valor Y. Si desea que el valor sea de tipo `RELATIVO`, introduzca -1.

**Figura 12.56** Rejilla en el Editor de GridBagConstraints

**Sugerencia** Según mueve el componente en la superficie de diseño, las posiciones `gridx` (columna) y `gridy` (fila) se muestran y actualizan en la barra de estado en la parte inferior derecha. “col:” es `gridx` y “fila:” es `gridy`. Los valores en el editor GridBagConstraints también se actualizan.

**Nota** Si emplea el ratón para desplazar el componente a una celda ocupada, el diseñador de interfaces de usuario impide la superposición de componentes insertando una fila y una columna. Si reubica el componente mediante el Editor de GridBagConstraints, el diseñador **no** comprueba si los componentes se superponen.

## weightx, weighty

Si desea aumentar el tamaño de las celdas, `weightx` y `weighty` deben tener un valor distinto de cero.

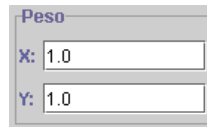
Para definir en la superficie de diseño las restricciones de peso de un componente, haga clic con el botón derecho sobre él y elija Peso horizontal o Peso vertical. Con ello, se elige el valor 1.0.

Para eliminar las restricciones de peso (asignarles el valor cero), haga clic con el botón derecho sobre el componente y elija Eliminar Pesos. Esto se puede hacer en varios componentes de un contenedor: mantenga pulsada la tecla *Mayús* mientras selecciona los componentes deseados, haga clic con el botón derecho del ratón y elija Eliminar Pesos.

Si desea que las restricciones de peso tengan un valor distinto de 0.0 y 1.0, puede introducir el que desee mediante el Editor de GridBagConstraints.

- 1 Haga clic con el botón derecho sobre el componente o componentes y seleccione Restricciones, para ver el Editor de GridBagConstraints.
- 2 Introduzca un valor comprendido entre 0.0 y 1.0 en los campos X e Y de la zona Peso y, a continuación, pulse Aceptar.

**Figura 12.57** Restricciones de Peso en el Editor de GridBagConstraints



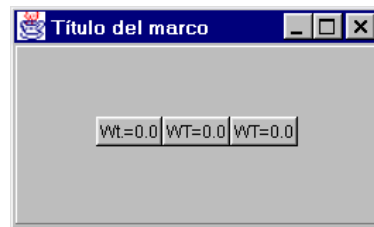
**Importante** Las restricciones de peso pueden hacer que sea difícil de predecir el comportamiento en cuanto al tamaño en el diseñador. Por ello, la definición de estas restricciones debe ser el último paso en el diseño de `GridBagLayout`.

## Comportamiento de las restricciones de peso

A continuación se muestran ejemplos de la forma en que las restricciones de peso afectan al comportamiento de los componentes:

- Si todos los componentes tienen una restricción de Peso cero en una sola dirección, los componentes se agrupan en el centro del contenedor en dicha dimensión y no sobrepasarán sus tamaños recomendados. `GridBagLayout` añade un espacio adicional entre la rejilla de celdas y los bordes del contenedor.

**Figura 12.58** Todos los componentes tienen restricciones de peso cero en la misma dirección



- Por ejemplo, si tiene tres componentes con la restricción `weightx` de 0.0, 0.3 y 0.2 respectivamente y aumenta el tamaño del contenedor, ninguna parte del espacio sobrante se asigna al primer componente, pero las 3/5 partes sí se aplicarán al segundo componente, y otras 2/5 partes al tercero.

**Figura 12.59** Tres componentes con restricciones `weightx` de valor 0.0, 0.3 y 0.2 respectivamente



- Si desea que el tamaño de los componentes pueda aumentar, ha de establecer las restricciones de peso y `expansión` para una dirección (horizontal o vertical).

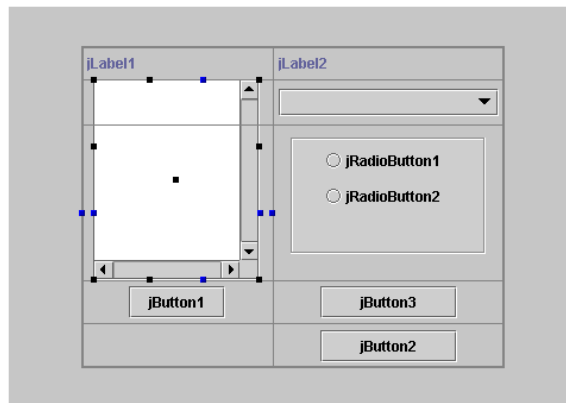
Por ejemplo:

- Si un componente tiene la restricción `weightx` con un valor distinto de cero, pero no la restricción `expansión` horizontal, el espacio adicional se asigna al Tamaño adicional entre los bordes izquierdo y derecho del componente y los bordes del área de visualización. Aumenta la anchura del área de visualización sin variar el tamaño del componente.
- Si un componente tiene las restricciones de peso y `expansión`, el espacio adicional se añade al área de visualización y el componente se amplía para ocupar ese incremento de tamaño en la dirección de la restricción `expansión` (en este caso, horizontal).

Este efecto se muestra en las tres figuras siguientes.

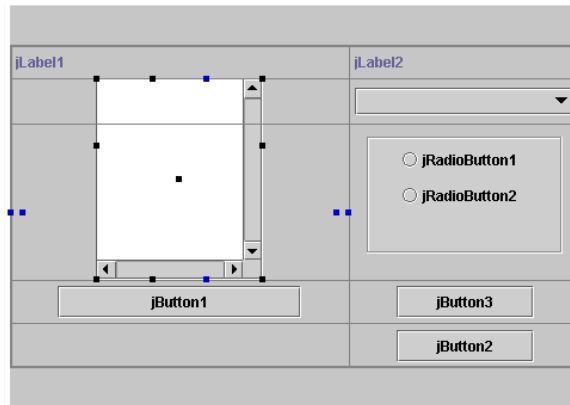
En el primer ejemplo, todos los componentes del panel `GridBagLayout` tienen una restricción de peso nula. Debido a ello, los componentes se agrupan en el centro del panel `GridBagLayout`, y el espacio sobrante de éste se distribuye en el margen entre los bordes exteriores de la rejilla y el propio panel. El tamaño de la rejilla está determinado por el tamaño recomendado de los componentes, más las restricciones de `encuadres`, `ipadx` e `ipady`).

**Figura 12.60** `GridBagLayout` con restricciones de peso cero en todos los componentes



Cuando se cambia el tamaño de este marco, los componentes conservan el tamaño preferente y se acumulan en el centro, mientras crece el espacio a su alrededor para llenar el contenedor ampliado.

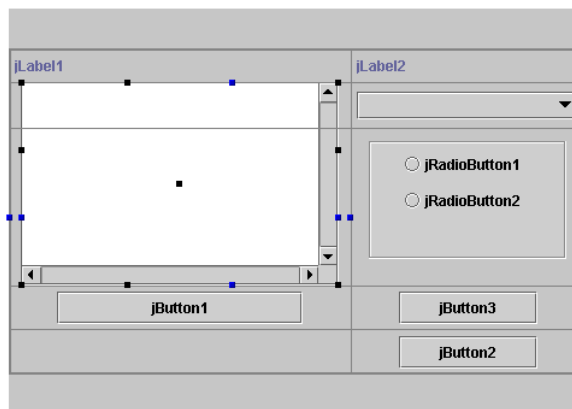
En el siguiente ejemplo, `jScrollPane1` tiene una restricción de peso horizontal (`weightx`) de 1.0. Observe que, inmediatamente después de asignar un peso a un control, el diseño de la interfaz de usuario deja de estar centrado en el panel. Dado que se ha establecido una restricción de Peso horizontal, el gestor de `GridBagLayout` toma el espacio sobrante del panel, previamente repartido a ambos lados de la rejilla, y lo coloca en la celda que contiene el control de lista `jScrollPane1`. Observe también en el mismo ejemplo que `jScrollPane1` no cambia de tamaño.

**Figura 12.61** JScrollPane1 con weightx=1.0, sin restricciones de expansión**Nota**

this

Si, una vez establecidas las restricciones de peso de los componentes, existe más espacio del deseado en las celdas, reduzca el tamaño del marco de la interfaz de usuario hasta conseguir la cantidad de espacio adicional que desee. Para ello, seleccione el marco `this` en el árbol de componentes (`this`); seguidamente, haga clic en los tiradores negros y arrástrelos hasta que el marco adquiriera el tamaño deseado.

En el último ejemplo, `JScrollPane1` tiene una restricción `weightx` de valor 1.0 y una restricción horizontal de expansión. Observe que `JScrollPane1` se expande para rellenar el ancho del área de visualización.

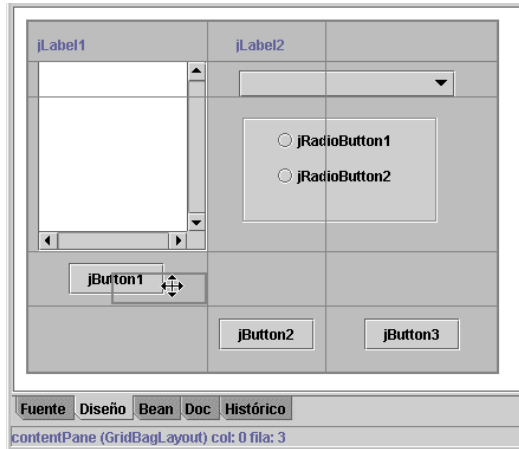
**Figura 12.62** JScrollPane1 con weightx=1.0, Expansión=Horizontal**Importante**

Si un componente de una columna tiene la restricción `weightx`, `GridBagLayout` asigna la totalidad de la columna a ese valor. De la misma manera, si un componente de una fila tiene la restricción `weighty`, la fila entera se asigna a ese valor.

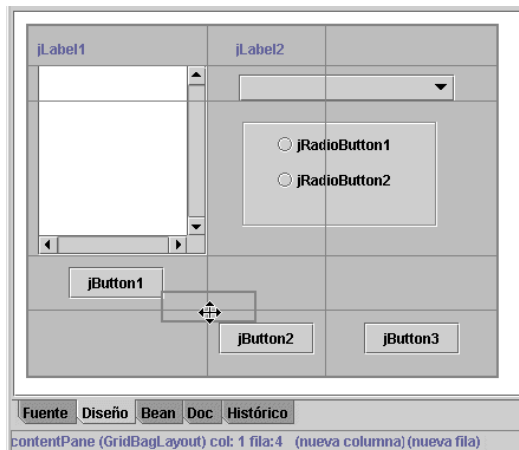
## Modificación de restricciones mediante el ratón

Al arrastrar un componente en el diseñador, su esquema indica hacia dónde se desplazará. La barra de estado indica en qué fila y columna se encuentra el componente y, si la celda de destino está ya ocupada, indica qué columna y/o fila se ha de crear para colocar el componente en la nueva posición.

**Figura 12.63** Componente al arrastrarlo en la celda actual del diseñador



**Figura 12.64** Componente al arrastrarlo a la celda nueva ocupada en el diseñador



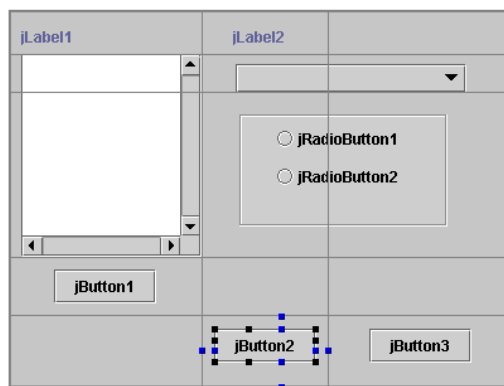
Cuando se desplaza un componente a otra parte del área de visualización, JBuilder calcula el Ancla más cercana y aplica los Encuadres necesarios para que se mantenga en su lugar.

## Arrastrar componentes a celdas vacías

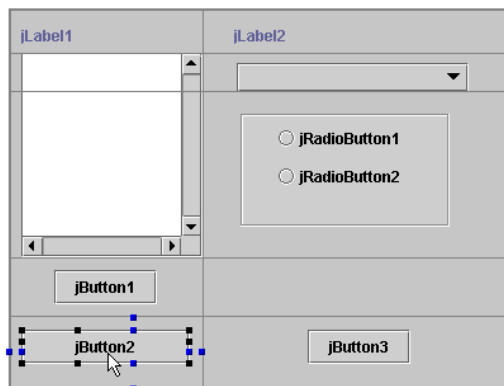
Al arrastrar un componente a una celda vacía, JBuilder mantiene la configuración de las restricciones de **Encuadre** y de **Expansión**.

Por ejemplo, si la restricción de **expansión** está configurada en **Horizontal**, al desplazar el componente a una celda más ancha, **GridBagLayout** expande el componente para rellenar la nueva celda.

**Figura 12.65** Antes de desplazar el `jButton2`, con la restricción de expansión configurada en **HORIZONTAL**



**Figura 12.66** Desplazamiento del `jButton2` al mismo tamaño o a una celda mayor



Observe que, en este caso, el desplazamiento del `jButton2` a una celda vacía bajo `jButton1` también provoca que se elimine una columna de la rejilla, ya que ningún componente la necesita. Los valores de las restricciones para los demás componentes se modifican de forma automática para adaptarse a este cambio (por ejemplo, `gridwidth` cambia de 2 a 1 en el caso de los componentes que ocupaban antes las columnas 1 y 2).

**Nota** La primera columna en la rejilla es la número 0.

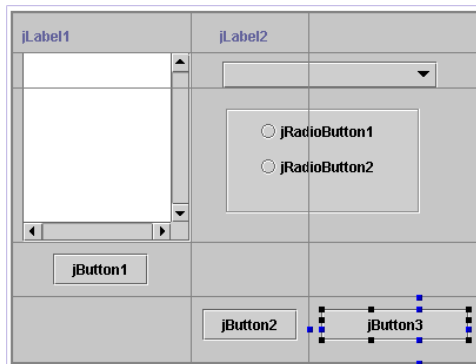


Este es un buen ejemplo, no obstante, de que modificar un diseño después de que el contenedor se ha convertido a `GridBagLayout` puede ser engañoso, ya que pueden producirse resultados inesperados.

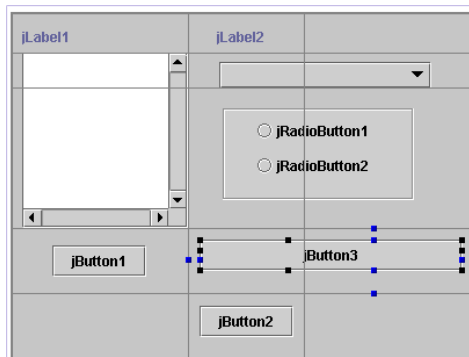
Si el componente que está desplazando es mayor que la celda a la que desea trasladarlo, `GridBagLayout` coloca el componente en dos celdas y, a continuación, aplica las restricciones de expansión y de Encuadre.

Por ejemplo, si la restricción de expansión para `jButton3` está configurada en `HORIZONTAL` y la arrastra a la celda sobre `jButton2`, que es más pequeña que el tamaño del botón, `GridBagLayout` hace que el botón ocupe las columnas 1 y 2.

**Figura 12.67** Antes de desplazar el `jButton3`, con la restricción de expansión configurada en `HORIZONTAL`



**Figura 12.68** Desplazamiento del `jButton3` a una celda más pequeña sobre el `jButton2`



Consulte [“Arrastrar componentes grandes a celdas pequeñas” en la página 12-56](#).

## Arrastrar componentes a celdas ocupadas

Esta acción tiene el efecto más notable en la modificación por el método de arrastrar y soltar, porque una celda no puede contener más de un componente. Por tanto, si intenta mover un componente a una celda ocupada, `GridBagLayout` realiza otros arreglos para el traslado. Crea una columna o fila para el componente.

Esto tiene como consecuencia el cambio de las restricciones de posición en la rejilla de muchos de los demás componentes. Sus restricciones `gridx` o `gridy` cambian si su posición es posterior o inferior a las nuevas columnas y filas de la rejilla. Sus restricciones `gridwidth` o `gridheight` pueden cambiar si a las columnas o filas que ya están ocupadas les afectan las nuevas.

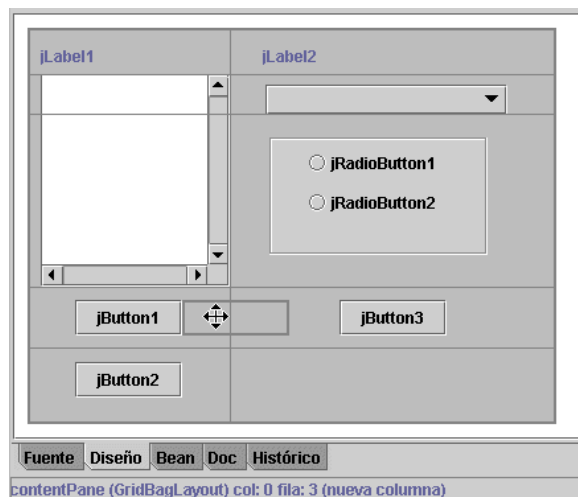
Es muy importante entender esta acción y el comportamiento resultante. Si está utilizando el diseñador para crear una interfaz de usuario en `GridBagLayout`, en lugar de hacerlo en diseño `XYLayout` o en `null`, la mayor parte del tiempo la rejilla estará llena cada vez que desee arrastrar un componente nuevo, con lo que habrá de crear nuevas columnas o filas y cambiar las restricciones para los componentes.

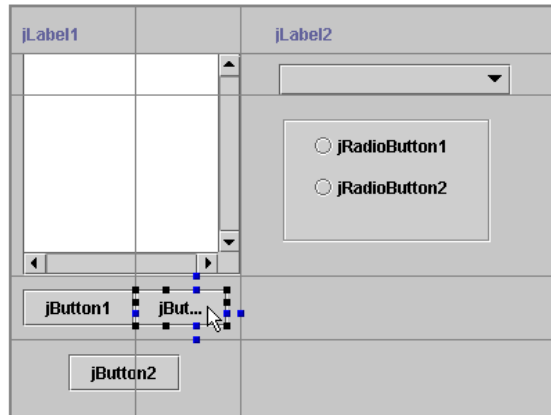
Los ejemplos que aparecen a continuación muestran lo que ocurre al arrastrar el `jButton3` a la celda adyacente, ocupada por el `jButton1`. Observe que al arrastrar el `jButton3` a la celda de `jButton1`, la barra de estado indica en qué columna o fila está el cursor y si se va a crear una fila o columna cuando se suelte el botón.

`GridBagLayout` lleva a cabo los siguientes cambios al arrastrar `jButton3` a la misma celda que `jButton1`:

- Se inserta una columna entre las dos que se han utilizado.
- Se aumenta la restricción `gridwidth` para `jLabel1`, el `jScrollPane1` y el `jButton2` a dos columnas en lugar de una.
- Se elimina el Encuadre derecho de `jButton1` y el Encuadre izquierdo de `jButton3`.

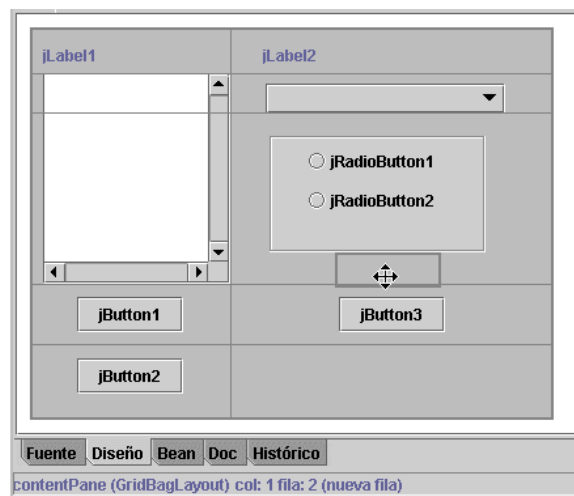
**Figura 12.69** Arrastre de `jButton3` a la celda de `jButton1`

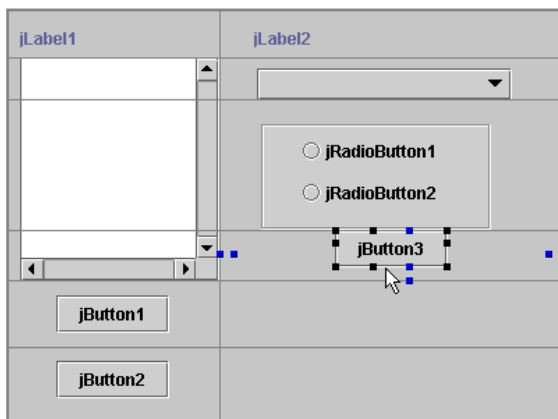


**Figura 12.70** Después de desplazar el botón en horizontal

Si se arrastra el `jButton3` a la celda con el panel de casillas de verificación (`jPanel1`):

- Se inserta una fila nueva antes de `jButton1`, dividiendo la fila 2 por la mitad (ocupada por el `jPanel1` y la parte inferior de `jScrollPane1`).
- Ha aumentado el valor de `gridheight` de `jScrollPane1`, de dos a tres filas.
- Se quita espacio de la fila grande ocupada por el `jScrollPane1` y el `jPanel1`.
- Se elimina el Encuadre inferior para `jPanel1` y el superior para `jButton3`.

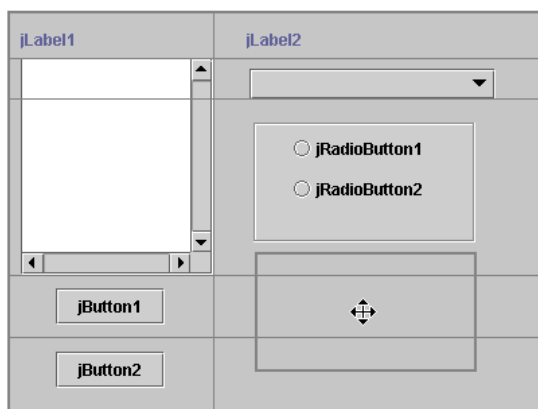
**Figura 12.71** Arrastre del `jButton3` a la celda con `jPanel1`

**Figura 12.72** Después de desplazar el botón en vertical

**Nota** Se obtiene el mismo resultado si se arrastra un botón nuevo a la celda ocupada por el `jButton1` o por `jPanel1`, excepto que los componentes nuevos no tienen restricciones de Expansión o Encuadre, mientras que un componente que se ha desplazado mantiene la configuración de sus restricciones de expansión y encuadres.

### Arrastrar componentes grandes a celdas pequeñas

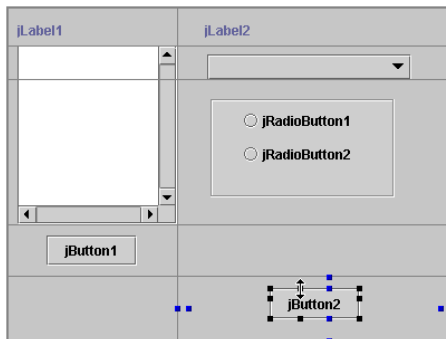
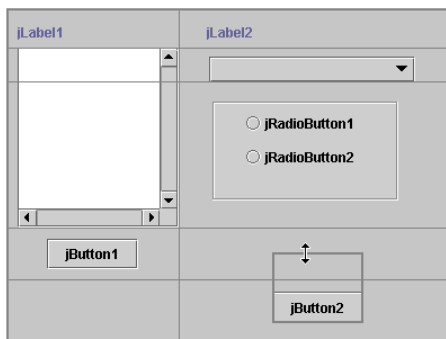
Si arrastra un componente a una celda más pequeña, el componente ocupa cuantas celdas vacías necesite, mientras mantiene sus restricciones de expansión y encuadres. Si el componente necesita más celdas de las disponibles (esto es, si llega a una celda ocupada o al borde del contenedor) las últimas celdas ocupadas aumentan de tamaño para alojar el resto del componente, incluidos sus Encuadres.

**Figura 12.73** Arrastre del `jPanel1` a una celda pequeña vacía

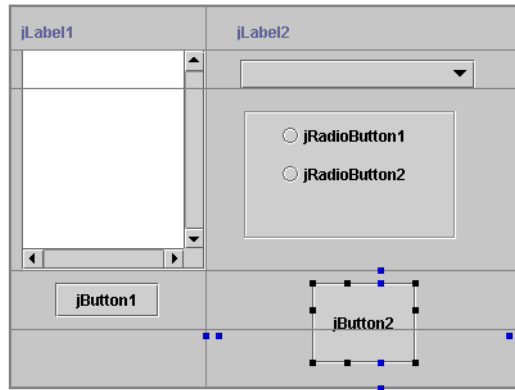
**Figura 12.74** Después de arrastrar jPanel1

### Arrastrar los tiradores de redimensionamiento negros a una celda adyacente vacía

Cuando se arrastra el tirador de redimensionamiento negro de un componente a una celda vacía contigua aumenta su área de visualización (anchura o altura en celdas) en una celda, en la dirección del movimiento.

**Figura 12.75** Antes de arrastrar el tirador de redimensionamiento superior de jButton2 a una celda superior vacía**Figura 12.76** Durante el arrastre

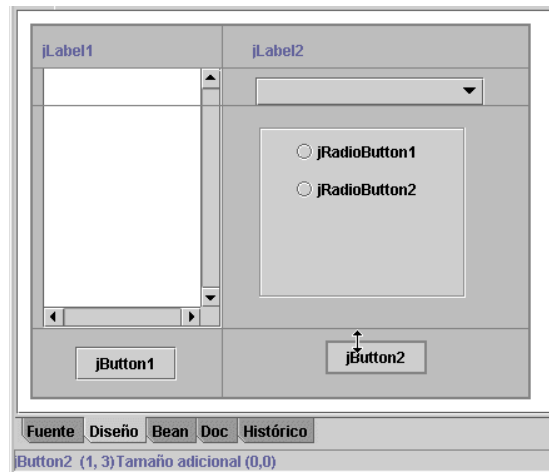
**Figura 12.77** Después del arrastre, jButton2 tiene gridheight de valor 2 con Encuadres sin cambios



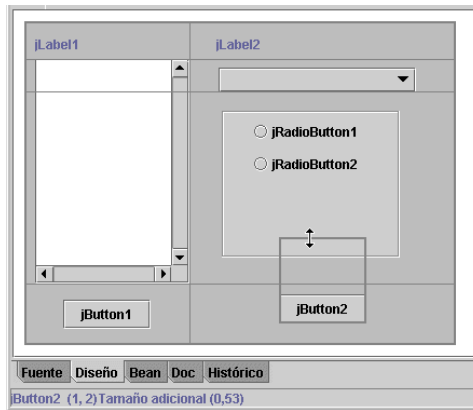
### Arrastrar los tiradores de redimensionamiento negros a una celda adyacente ocupada

Cuando se arrastra el tirador de redimensionamiento negro de un componente a una celda ocupada contigua aumentan los valores de sus restricciones `ipadx` e `ipady`. Observe cómo muestra esto la barra de estado en el siguiente ejemplo.

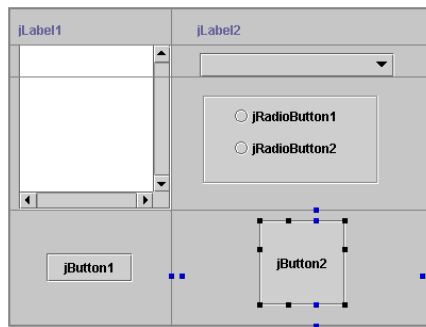
**Figura 12.78** Antes de arrastrar, jButton2 tiene Tamaño adicional cero



**Figura 12.79** Arrastrar tirador de redimensionamiento a la celda ocupada aumenta el Tamaño adicional



**Figura 12.80** Después de arrastrar, las celdas son mayores, el Tamaño adicional ha aumentado



## Cómo añadir componentes

Al añadir un componente nuevo al contenedor GridBagLayout, el lugar donde se pulsa para soltar el componente determina las columnas o filas que se crean para adaptarse a él.

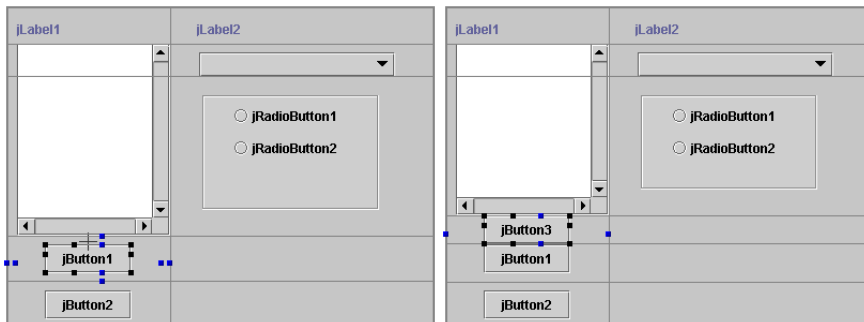
**Nota** Las restricciones de expansión y de encuadres para un componente nuevo que se añade a GridBagLayout tienen un valor `None`.

- Para crear una fila sobre un componente, haga clic en la parte superior de éste.
- Para crear una fila bajo un componente, haga clic en la parte inferior de éste.
- Para crear una columna a la izquierda de un componente, haga clic a la izquierda de éste.
- Para crear una columna a la derecha de un componente, haga clic a la derecha de éste.

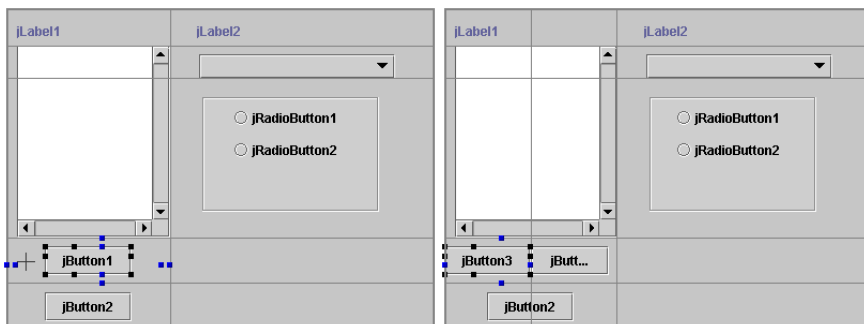
Después de seleccionar un componente en la paleta de componentes, vigile la barra de estado mientras mueve el ratón sobre la rejilla para ver qué sucede. La barra de estado indica la columna y la fila que ocupará el componente (posiciones `gridx` y `gridy`), y si se va a crear una fila o una columna para alojarlo.

Los ejemplos siguientes muestran la adición de un nuevo componente en cada lado de `jButton1`:

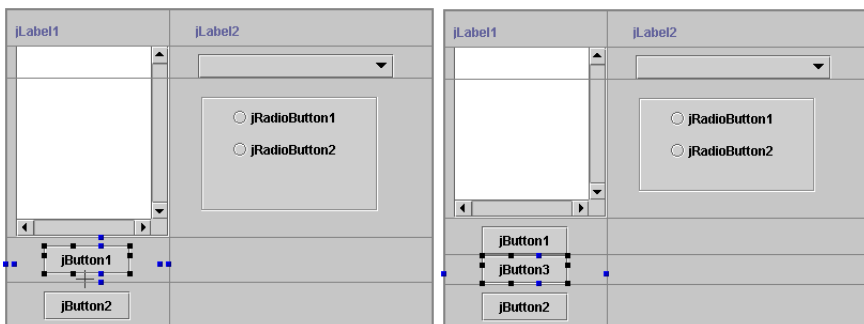
**Figura 12.81** Si hace clic sobre `jButton1`, se crea una fila para `jButton3` sobre `jButton1`



**Figura 12.82** Si hace clic a la izquierda de `jButton1`, se crea una columna para `jButton3` a la izquierda de `jButton1`

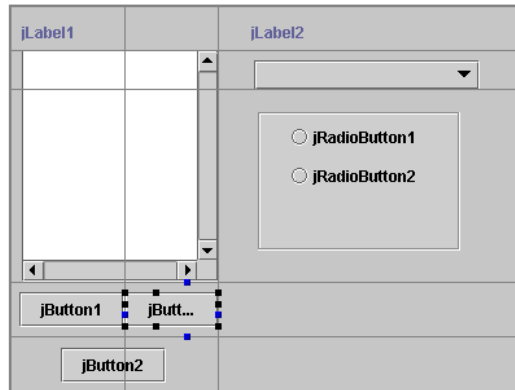


**Figura 12.83** Si hace clic debajo de `jButton1`, se crea una fila para `jButton3` debajo de `jButton1`





**Figura 12.84** Si hace clic a la derecha de jButton1, se crea una columna para jButton3 a la derecha de jButton1



## Sugerencias diversas

### Vuelta a XYLayout si se deben hacer ajustes significativos

Si no ha obtenido el resultado deseado en la conversión o necesita añadir componentes, vuelva a `XYLayout`, efectúe los cambios y vuelva a convertir el diseño en `GridBagLayout`. Esto puede ser más rápido y fácil que intentar añadir componentes en `GridBagLayout`. Es mejor dejar que JBuilder realice la tarea de calcular y asignar las restricciones.

### Eliminación de los pesos y expansiones antes de efectuar los ajustes

Probablemente tendrá que realizar algunos ajustes después de la conversión a `GridBagLayout`. Durante la conversión de `XYLayout` en `GridBagLayout`, JBuilder asigna automáticamente los valores de la restricción de peso a algunos componentes.

Si se encuentra con dificultades al empezar a mover componentes o controladores de tamaño en la superficie de diseño, haga clic en **Deshacer** y, a continuación, elimine los valores de restricción de peso de todos los componentes del contenedor `GridBagLayout`. Las restricciones de peso son la principal causa de comportamiento inesperado al mover y cambiar el tamaño de componentes de forma gráfica en un diseño `GridBagLayout`. Si se eliminan antes todas las restricciones de peso resulta más fácil efectuar los ajustes correctos en las otras restricciones.

**Nota** Esto también se puede aplicar a las restricciones de *expansión*. Si se eliminan resulta más fácil realizar los ajustes.

Ajuste todas las demás restricciones que haya que modificar. Cuando todas las demás restricciones tengan los valores deseados, añada las restricciones de peso únicamente a los componentes que las necesiten.

## Diseño visual del código GridBagLayout ya escrito

---

### Diferencias en el código

Si se crean contenedores `GridBagLayout` escribiendo código manualmente, basta con crear un objeto `GridBagConstraints` para el contenedor `GridBagLayout` y volver a utilizarlo cada vez que se le añadan componentes. Si desea que el componente que está añadiendo al contenedor tenga valores distintos de los del añadido previamente para restricciones concretas, basta con cambiar estas restricciones para utilizarlas con el nuevo componente. Estos nuevos valores se aplicarán a los componentes que añada posteriormente, hasta que vuelva a cambiarlos.

#### Importante

Si bien esta metodología de codificación de `GridBagLayout` produce el código más reducido (se reciclan los objetos `GridBagConstraints` tomados de componentes previamente añadidos), también impide la edición visual del contenedor en el diseñador de JBuilder.

Cuando se diseña un contenedor `GridBagLayout` mediante el diseñador, JBuilder crea un objeto `GridBagConstraints` por cada componente añadido a ese contenedor. El objeto `GridBagConstraints` tiene un constructor que toma las once propiedades de `GridBagConstraints`, de forma que el código generado por el diseñador siempre puede seguir el mismo patrón.

```
public GridBagConstraints(int gridx,
                           int gridy,
                           int gridwidth,
                           int gridheight,
                           double weightx,
                           double weighty,
                           int anchor,
                           int fill,
                           Insets insets,
                           int ipadx,
                           int ipady)
```

Por ejemplo:

```
jPanell.add(jButton1, new GridBagConstraints(0, 0, 1, 1, 0.0, 0.0,
GridBagConstraints.CENTER, GridBagConstraints.NONE,
new Insets(0, 0, 0, 0), 0, 0));
```

### Modificación del código para que funcione en el diseñador

Si ha escrito manualmente el código de un `GridBagLayout`, mediante un objeto `GridBagConstraints`, no puede editar ese contenedor en el diseñador de interfaces a menos que introduzca las siguientes modificaciones en el código:

Se debe crear un objeto `GridBagConstraints` por cada objeto con un gran constructor que tenga parámetros para los once valores de restricción, como se muestra más arriba.

## Código generado por JBuilder en la segunda parte

A continuación se muestra el código real generado por JBuilder al crear la interfaz GridBagLayout en la segunda parte del tutorial, [“Creación de diseños GridBagLayout en JBuilder” en la página 12-17.](#)

```
package gbl;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
//import com.borland.jbcl.layout.*;

public class Frame1 extends JFrame {

    //Construir el marco
    BorderLayout borderLayout1 = new BorderLayout();
    JPanel jPanel1 = new JPanel();
    JPanel jPanel2 = new JPanel();
    JLabel jLabel1 = new JLabel();
    JList jList1 = new JList();
    JButton jButton1 = new JButton();
    JCheckBox jCheckBox1 = new JCheckBox();
    JLabel jLabel2 = new JLabel();
    JButton jButton2 = new JButton();
    JPanel jPanel3 = new JPanel();
    JCheckBox jCheckBox2 = new JCheckBox();
    JList jList2 = new JList();
    JPanel jPanel4 = new JPanel();
    JButton jButton3 = new JButton();
    JButton jButton4 = new JButton();
    JButton jButton5 = new JButton();
    GridBagLayout gridBagLayout1 = new GridBagLayout();
    GridBagLayout gridBagLayout2 = new GridBagLayout();
    GridBagLayout gridBagLayout3 = new GridBagLayout();
    GridLayout gridLayout1 = new GridLayout();

    public Marco1() {
        enableEvents(AWTEvent.WINDOW_EVENT_MASK);
        try {
            jbInit();
        }
        catch(Exception e) {
            e.printStackTrace();
        }
    }

    //Iniciación del componente

    private void jbInit() throws Exception {
        this.getContentPane().setLayout(borderLayout1);
        this.setSize(new Dimension(332, 304));
        jPanel2.setBackground(new Color(192, 192, 255));
    }
}
```

```

jLabel1.setText("Sorted Columns");
jLabel1.setFont(new Font("Dialog", 0, 11));
jButton1.setText("Remove from Sort");
jCheckBox1.setText("Descending");
jLabel2.setFont(new Font("Dialog", 0, 11));
jButton2.setText("Add to Sort");
jPanel2.setBackground(new Color(192, 192, 255));
jPanel3.setLayout(gridBagLayout3);
jCheckBox2.setText("Case Sensitive");
jButton3.setText("Cancel");
jButton4.setText("Help");
jButton5.setText("OK");
gridLayout1.setHgap(6);
jPanel4.setLayout(gridLayout1);
jLabel2.setText("Available Columns");
jPanel2.setLayout(gridBagLayout2);
jPanel1.setLayout(gridBagLayout1);
this.setTitle("Título del marco");
this.getContentPane().add(jPanel1, BorderLayout.CENTER);
jPanel1.add(jPanel2, new GridBagConstraints(0, 0, 1, 1, 1.0, 1.0,
    GridBagConstraints.CENTER, GridBagConstraints.BOTH,
    new Insets(10, 10, 0, 10), 0, 0));
jPanel2.add(jLabel1, new GridBagConstraints(0, 0, 1, 1, 0.0, 0.0,
    GridBagConstraints.WEST, GridBagConstraints.BOTH,
    new Insets(0, 0, 2, 0), 0, 4));
jPanel2.add(jList1, new GridBagConstraints(0, 1, 1, 1, 1.0, 1.0,
    GridBagConstraints.WEST, GridBagConstraints.BOTH,
    new Insets(0, 0, 0, 0), 128, 128));
jPanel2.add(jButton1, new GridBagConstraints(0, 2, 1, 1, 0.0, 0.0,
    GridBagConstraints.CENTER, GridBagConstraints.NONE,
    new Insets(7, 0, 0, 0), 0, 0));
jPanel2.add(jCheckBox1, new GridBagConstraints(0, 3, 1, 1, 0.0, 0.0,
    GridBagConstraints.WEST, GridBagConstraints.BOTH,
    new Insets(6, 0, 0, 0), 0, 0));
jPanel1.add(jPanel3, new GridBagConstraints(1, 0, 1, 1, 1.0, 1.0,
    GridBagConstraints.CENTER, GridBagConstraints.BOTH,
    new Insets(10, 10, 0, 10), 0, 0));
jPanel3.add(jLabel2, new GridBagConstraints(0, 0, 1, 1, 0.0, 0.0,
    GridBagConstraints.WEST, GridBagConstraints.BOTH,
    new Insets(0, 0, 2, 0), 0, 4));
jPanel3.add(jList2, new GridBagConstraints(0, 1, 1, 1, 1.0, 1.0,
    GridBagConstraints.CENTER, GridBagConstraints.BOTH,
    new Insets(0, 0, 0, 0), 128, 128));
jPanel3.add(jButton2, new GridBagConstraints(0, 2, 1, 1, 0.0, 0.0,
    GridBagConstraints.CENTER, GridBagConstraints.NONE,
    new Insets(7, 0, 0, 0), 32, 0));
jPanel3.add(jCheckBox2, new GridBagConstraints(0, 3, 1, 1, 0.0, 0.0,
    GridBagConstraints.WEST, GridBagConstraints.BOTH,
    new Insets(6, 0, 0, 0), 0, 0));
jPanel1.add(jPanel4, new GridBagConstraints(0, 1, 2, 1, 0.0, 0.0,
    GridBagConstraints.CENTER, GridBagConstraints.NONE,
    new Insets(12, 59, 12, 59), 0, 0));

```

```

jPanel4.add(jButton5, null);
jPanel4.add(jButton3, null);

jPanel4.add(jButton4, null);
}

//Sobreescrito para salir de System Close

protected void processWindowEvent(WindowEvent e) {
    super.processWindowEvent(e);
    if (e.getID() == WindowEvent.WINDOW_CLOSING) {
        System.exit(0);
    }
}
}

```

## Otros recursos sobre GridBagConstraints

- **java.awt.GridBagConstraints** en <http://java.sun.com/j2se/1.3/docs/api/java/awt/GridBagConstraints.html>.
- **java.awt.GridBagLayout** en <http://java.sun.com/j2se/1.3/docs/api/java/awt/GridBagLayout.html>.

# GridBagConstraints

---

## Ancla

---

### Descripción

Cuando el componente sea más pequeño que el área en que se visualiza, utilice la restricción `ancla` para indicar al gestor de diseño dónde debe situar el componente dentro del área.

La restricción `ancla` sólo afecta al componente dentro de su propia área de visualización, en función de la restricción `expansión` del componente. Por ejemplo, si el valor de la restricción `expansión` de un componente es `GridBagConstraints.BOTH` (AMBAS, es decir que toda el área de visualización está ocupada, horizontal y verticalmente), la restricción `ancla` no tiene efecto alguno porque el componente ocupa la totalidad del área disponible. Para que la restricción `ancla` tenga efecto, asigne a la restricción `expansión` el valor `GridBagConstraints.NONE` (ANULADA), `GridBagConstraints.HORIZONTAL` (HORIZONTAL) o `GridBagConstraints.VERTICAL`.

### Valores válidos

```

GridBagConstraints.CENTER
GridBagConstraints.NORTH
GridBagConstraints.NORTHEAST
GridBagConstraints.EAST

```

```
GridBagConstraints.SOUTHEAST  
GridBagConstraints.SOUTH  
GridBagConstraints.SOUTHWEST  
GridBagConstraints.WEST  
GridBagConstraints.NORTHWEST
```

### Valor por defecto

```
GridBagConstraints.CENTER
```

Consulte las sugerencias para la configuración de restricciones de `ancla` en la [página 12-42](#).

## Expansión

---

### Descripción

Si el área de visualización en la que se inserta un componente es superior a este último, la restricción `expansión` le permite indicar al gestor de diseño las zonas del área que debe asignar al componente.

En el caso de la restricción `ancla`, las restricciones de `expansión` solamente afectan al componente dentro de su área de visualización. Las restricciones de `expansión` indican al gestor de diseño que debe expandir el componente para que ocupe el área completa que se le da.

### Valores válidos

<code>GridBagConstraints.NONE</code>	El componente no se expande.
<code>GridBagConstraints.BOTH</code>	El componente se expande horizontal y verticalmente para ocupar la totalidad del área.
<code>GridBagConstraints.HORIZONTAL</code>	El componente se expande para ocupar toda el área sólo horizontalmente.
<code>GridBagConstraints.VERTICAL</code>	El componente se expande para ocupar toda el área sólo verticalmente.

### Valor por defecto

```
GridBagConstraints.NONE
```

Consulte las sugerencias para la configuración de restricciones de `ancla` en la [página 12-43](#).

## Encuadres

---

### Descripción

Utilice `encuadres` para especificar el espacio externo (márgenes) en píxeles, que media entre el componente y los bordes de su área de visualización. La propiedad `encuadre` establece un espacio constante entre el borde del

componente y el borde de la celda que lo contiene. Por lo tanto, `encuadre` actúa como “frontera” del componente, con el objeto de mantenerlo alejado de los bordes de la celda. Por ejemplo, si altera la anchura de un componente ampliando el `encuadre` a la izquierda y a la derecha y supera los límites de la celda, ésta se expande para acomodar tanto al componente como a sus `encuadres`. Debido a ello, las restricciones `expansión` y `tamaño adicional` nunca restan espacio a los `encuadres`.

### Valores válidos

```
insets = new Insets(n,n,n,n)
```

Superior, Izquierdo, Inferior y Derecho (donde cada uno de los parámetros representa el número de píxeles existente entre el área de visualización y el correspondiente borde de la celda).

### Valores por defecto

```
insets = new Insets(0,0,0,0)
```

Consulte las sugerencias para la configuración de restricciones de `ancla` en la [página 12-43](#).

## gridwidth, gridheight

---

### Descripción

Utilice las restricciones `gridwidth` y `gridheight` para establecer el número de celdas de una fila (`gridwidth`) o columna (`gridheight`) que utiliza el componente. El valor de esta restricción se introduce en número de celdas y no en píxeles.

### Valores válidos

```
gridwidth=nn, gridheight=nn
```

Donde `nn` es un número entero que representa el número de celdas por columna y por fila.

`GridBagConstraints.RELATIVE (-1)` Especifica que este componente es el penúltimo de la fila (`gridwidth`) o la columna (`gridheight`). Los componentes que utilizan el valor `GridBagConstraints.RELATIVE` (relativo) ocupan todas las celdas restantes, excepto la última. Por ejemplo, en una fila con seis columnas, si el componente empieza en la columna número 3, una propiedad `gridwidth` del componente `RELATIVE` se extiende a las columnas 3, 4 y 5. Observe que las columnas y filas empiezan la numeración en 0 en la rejilla.

`GridBagConstraints.REMAINDER (0)` (resto) Especifica que este componente es el último de la fila (`gridwidth`) o la columna (`gridheight`).

**Valor por defecto**

`gridwidth=1, gridheight=1`

Consulte las sugerencias para la configuración de restricciones de `gridwidth`, `gridheight` en la [página 12-44](#).

**ipadx, ipady**

---

**Descripción**

Utilice `ipadx` e `ipady` para especificar el número de píxeles con los que se quiere incrementar el tamaño mínimo del componente, es decir, el tamaño adicional que se le quiere dar. Por ejemplo, la anchura del componente será, al menos, la anchura mínima más `ipadx` en píxeles. El código sólo lo añade una vez y lo divide por igual entre ambos lados del componente. De forma parecida, la altura del componente será de al menos la altura mínima más `ipady` en píxeles.

Estas restricciones indican el tamaño adicional de un componente:

- `ipadx` especifica el número de píxeles que debe añadirse a la anchura mínima del componente.
- `ipady` especifica el número de píxeles que debe añadirse a la altura mínima del componente.

**Ejemplo**

Cuando se añaden a un componente que tiene un tamaño mínimo recomendado de 30 píxeles de ancho y 20 píxeles de alto:

- Si `ipadx= 4`, el componente tiene una anchura de 34 píxeles.
- Si `ipady= 2`, el componente tiene una altura de 22 píxeles.

**Valores válidos**

`ipadx=nn, ipady=nn`

**Valor por defecto**

`ipadx=0, ipady=0`

Consulte las sugerencias para la configuración de restricciones de `ipadx`, `ipady` en la [página 12-45](#).

**gridx, gridy**

---

**Descripción**

Utilice estas restricciones para especificar la posición de la celda donde se encuentra la esquina superior izquierda del componente. Por ejemplo, `gridx=0` es la primera columna de la izquierda, mientras que `gridy=0` es la primera fila de la parte superior. Por lo tanto, un componente que tenga las restricciones `gridx=0` y `gridy=0` se situará en la primera celda de la rejilla (arriba a la izquierda).



`GridBagConstraints.RELATIVE` especifica que el componente debe situarse de forma relativa al componente anterior, como se explica a continuación:

- Cuando se utiliza con `gridx`, indica que el componente debe situarse inmediatamente a la derecha del último componente añadido.
- Cuando se utiliza con `gridy`, indica que el componente debe situarse inmediatamente debajo del último componente añadido.

### Valores válidos

`gridx=nn, gridy=nn`

`GridBagConstraints.RELATIVE (-1)`

### Valor por defecto

`gridx=1, gridy=1`

Consulte las sugerencias para la configuración de restricciones de `gridx`, `gridy` en la [página 12-46](#).

## weightx, weighty

---

### Descripción

Las restricciones de peso permiten establecer cómo se distribuye el espacio adicional de los contenedores `GridBagLayout`, horizontal (`weightx`) y verticalmente (`weighty`), cuando se redimensionan. Los pesos determinan qué parte del espacio adicional deben obtener las celdas y los componentes cuando se aumenta el tamaño del contenedor más allá de su tamaño por defecto.

Los valores de los pesos son de tipo `double` y se especifican con un número de 0.0 a 1.0, ambos inclusive. Un cero significa que el componente no debe obtener espacio adicional, mientras que 1.0 significa que el componente debe obtener la totalidad del espacio disponible.

- El peso de una fila es el resultado de sumar los valores máximos de `weightx` de todos los componentes de la fila.
- El peso de una columna es el resultado de sumar los valores máximos de `weighty` de todos los componentes de la columna.

**Importante** Si desea aumentar el tamaño de las celdas, `weightx` y `weighty` deben tener un valor distinto de cero.

### Valores válidos

`weightx=n.n, weighty=n.n`

### Valor por defecto

`weightx=0.0, weighty=0.0`

Consulte las sugerencias para la configuración de restricciones de `weightx`, `weighty` en la [página 12-47](#).

## Ejemplos de restricciones de Peso

Figura 12.85 Restricciones de Peso en paneles y listas

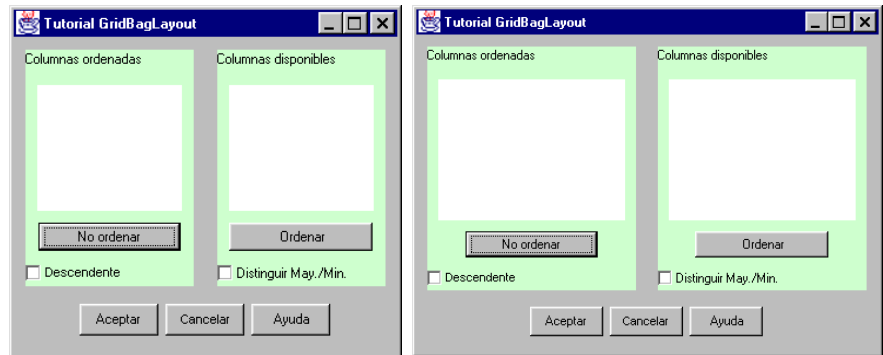


Figura 12.86 Restricciones de Peso en paneles, pero no en listas

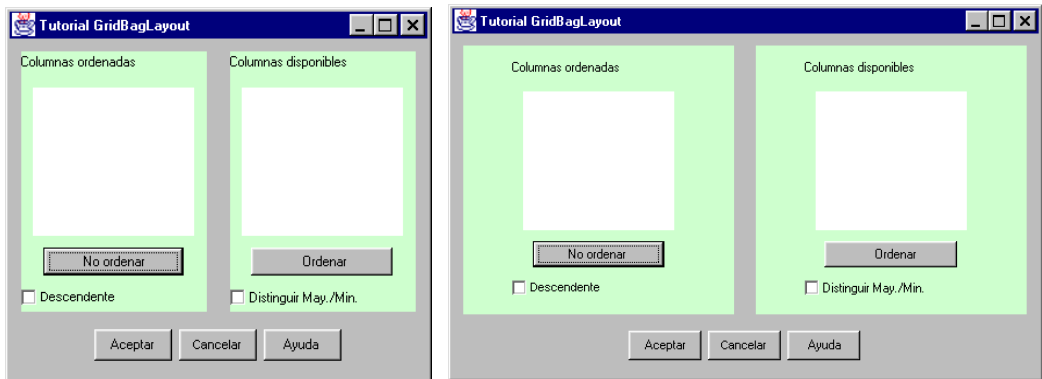
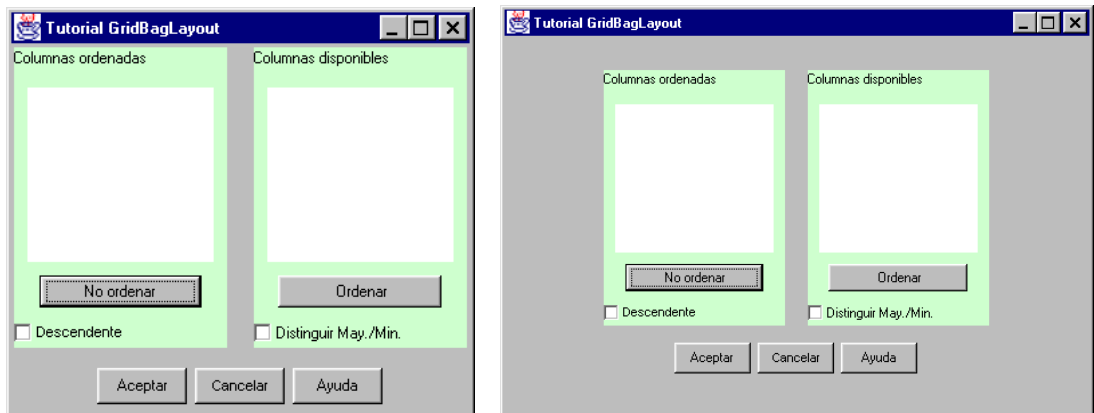
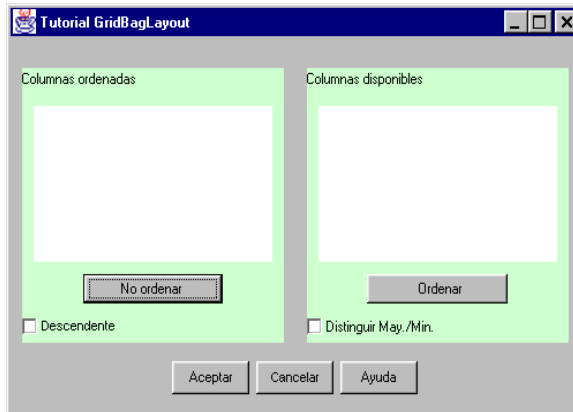
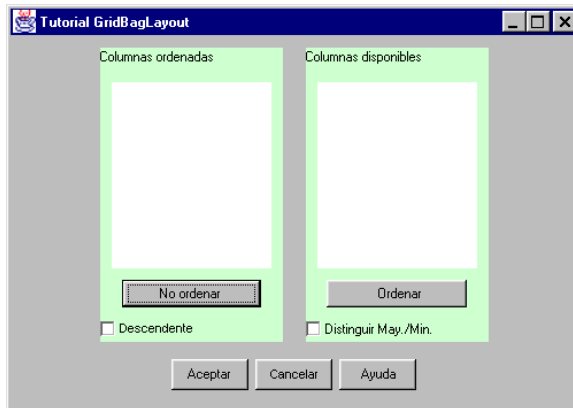
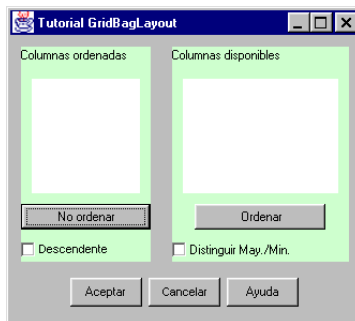


Figura 12.87 Restricciones de Peso en listas, pero no en paneles



**Figura 12.88** Sólo restricciones de Peso (weightx) horizontales en los cuatro componentes**Figura 12.89** Sólo restricciones de Peso (weighty) verticales en los cuatro componentes**Figura 12.90** Sólo restricciones de Peso en un componente de panel y lista de la fila





# Migración de archivos desde otros IDE de Java

JBuilder permite migrar archivos y aplicaciones desarrolladas en otros IDE Java. En algunos casos, es necesario modificar el código para que el archivo se pueda diseñar visualmente con la ayuda de las herramientas de diseño visual de JBuilder. Los archivos Java deben cumplir determinados requisitos para que sean diseñables visualmente.

Para ver un ejemplo de un archivo diseñable visualmente, cree un proyecto de JBuilder (ArchivoNuevo proyecto) y utilice el Asistente para aplicaciones (ArchivoNuevo) para crear una nueva aplicación.

## Consulte

- [“Requisitos para poder diseñar visualmente una clase” en la página 1-1.](#)

## VisualAge

---

JBuilder se prueba en VisualAge versión 4.5.1. No es necesario realizar modificaciones en estos archivos. Las herramientas de diseño visual de JBuilder son capaces de reconocer estos archivos si cumplen los requisitos de un archivo diseñable visualmente. Utilice el Asistente para Proyecto para código existente con el fin de crear un proyecto que importe el árbol de código fuente. El asistente Proyecto para código existente es una característica de JBuilder SE y Enterprise.

### Consulte

- “Creación de un proyecto a partir de archivos existentes” en *Creación de aplicaciones con JBuilder*.

## Forte

---

Los archivos Java creados en Forte se han de modificar de la siguiente manera:

- 1 Cree un método `jbInit()`.
- 2 Ponga el código de inicialización de la interfaz de usuario en el método `jbInit()`. Esto incluye código que añade componentes al contenedor y que configura el tratamiento de sucesos, pero no el código que ejecuta ese tratamiento.
- 3 Ponga todas las declaraciones de componentes fuera del método `jbInit()` al nivel de clase.

## VisualCafé

---

El asistente para la importación de proyectos de VisualCafé automatiza el proceso de conversión a JBuilder de proyectos creados en VisualCafé. Este asistente se encuentra en la ficha Proyecto de la galería de objetos.

Para importar un proyecto de VisualCafé:

- 1 Seleccione Archivo|Nuevo.  
Se abre la galería de objetos.
- 2 Seleccione la ficha Proyecto.
- 3 Seleccione el icono Importar proyecto de VisualCafé.
- 4 Para abrir el asistente, haga doble clic en el icono, pulse Aceptar o pulse la tecla *Intro*.

### Consulte

- El tema “Asistente para la importación de proyectos de VisualCafé” en la ayuda en línea. Pulse el botón Ayuda del asistente o elija Ayuda|Temas de ayuda, abra la ficha Buscar y escriba `visualcafe`.

# Índice

## A

- accesibilidad
  - añadir componentes de interfaz 3-3
  - diseñador
    - comandos de método abreviado de teclado 2-9
    - desplazamiento 2-9
  - editores de propiedades en el Inspector 8-50
  - modificar componentes en el Inspector 8-51
  - teclas de método abreviado del diseñador 2-8
- adaptadores
  - anónimos de clase interna 4-3, 4-5
    - seleccionar 4-6
  - de clases internas 4-6
  - de suceso
    - clase interna anónima 4-5
    - estándar 4-4
  - de sucesos estándar 4-3, 4-4
    - seleccionar 4-6
- agrupar componentes 5-5, 12-12
- ajuste de las dimensiones del marco en tiempo de ejecución
  - interfaz de usuario durante la ejecución 8-7
- añadir
  - componentes 3-3
    - a contenedores anidados 5-3
    - a GridBagLayout 8-32, 12-59
    - a la paleta de componentes 7-2
    - añadir a la interfaz 5-6
    - base de datos 5-9
  - menús 6-4, 6-5
- árbol de componentes 2-6, 3-1
  - abrir diseñadores 3-3
  - accesibilidad 3-3
  - añadir componentes 3-3
  - cambiar el nombre del componente 3-5
  - iconos 3-6
  - mover componentes 3-6
  - ver nombres de clase 3-6
- archivos
  - de imagen
    - paleta de componentes 7-4
  - migrar archivos desde las herramientas de desarrollo de Java A-1
- arrastrar y soltar
  - diseño visual 2-3
- asignar
  - valores a propiedades 3-6
  - valores a sucesos 3-6
- Asistencia a desarrolladores 1-7

- asistentes
  - en diseño visual 1-2
  - para aplicaciones
    - archivos de interfaz de usuario generados 1-2
- aspecto
  - "fase de diseño" 5-11
  - cambiar 5-10
  - durante la ejecución 5-10
  - en la fase de diseño 5-11
  - Java Metal 5-10
  - MacOS Adaptive 5-10
  - Metal 5-10
  - tiempo de ejecución 5-10
  - tiempo de ejecución y fase de diseño 5-10
  - Windows 5-10

## B

- barras
  - de estado
  - diseñador 2-3
  - de herramientas
    - diseñador de menús 6-3
- beans rojos 7-7
- bibliotecas de componentes 1-5
  - paleta de componentes 2-4
- BorderLayout 8-15
  - restricciones (configuración) 8-16
- Borland
  - asistencia
    - a desarrolladores 1-7
    - técnica 1-7
  - contacto 1-7
  - correo electrónico 1-9
  - grupos de noticias 1-8
  - informar sobre errores 1-9
  - recursos en línea 1-8
  - World Wide Web 1-8
- borrar
  - componentes en el diseñador 3-5
  - manejadores de sucesos 4-3

## C

- cambiar
  - propiedades de los diseños 8-3, 8-5
  - restricciones
    - de diseños 8-4
    - de diseños de componentes 8-6
- CardLayout 8-23

- crear controles 8-24
- Espacio 8-25
- CDE/Motif - Aspecto Solaris 5-10
- celda de rejilla
  - definición 8-26
- clases
  - de adaptadores de sucesos 4-6
  - descripción general 4-3
  - requisitos de diseño visual 1-1
- código
  - generado por los sucesos 4-3, 4-6
- componentes
  - adjuntar manejadores de sucesos 4-2
  - agrupar 5-5, 12-12
  - ajenos a la interfaz, relacionados con datos 2-8
  - alinear 8-14
  - añadir
    - a contenedores anidados 5-3
    - a paleta de componentes 7-2
    - a PaneLayout 8-49
    - al botón Selección de Beans 2-4
    - al diseño 3-3
    - componentes no visuales a la interfaz 5-6
  - asignar valores a propiedades 3-10
- AWT
  - comparación 1-5
  - Swing, dbSwing 1-5
- base de datos 5-6, 5-9
- beans rojos 7-7
- buscar en la superficie de diseño 2-3
- cambiar el nombre en el árbol 3-5
- contenedores 1-3
  - para agrupación 5-5
- convertir a GridBagLayout 12-59
- cortar, copiar y pegar componentes 3-4
- DataExpress 2-8
- dbSwing
  - comparación 1-5
- de base de datos
  - añadir a la interfaz 5-6, 5-9
- de menú 6-1
  - menú emergente 2-8
- de terceros 7-2
- definición de propiedades compartidas 3-9
- del cuadro de diálogo
  - cuadros de diálogo emergentes 2-8
- del interfaz de usuario
  - agrupar 5-5
  - añadir a paleta de componentes 7-1
  - seleccionar 5-2
- descripción general 1-3
- Dialog 1-4
- disponer en rejillas 8-21, 8-25

- eliminar en el diseño de interfaces de usuario 3-5
- en columnas 8-19
- en filas 8-18
- Frame 1-4
- gestionar en el árbol de componentes 3-1
- Inspector 2-6
- manipular en diseños de interfaces de usuario 3-4, 5-3
- maximumSize 8-6
- menú *Consulte* diseñador de menús
- menús 6-1
- minimumSize 8-6
- modificar restricciones de diseño 8-6
- modificar y desplazar en el árbol de componentes 2-6
- no visuales
  - añadir a la interfaz 5-6, 5-9
- paleta de componentes 2-4
- Panel 1-4
- preferredSize 8-6
- preinstalados 7-1
- propiedad
  - alignmentX 8-6
  - alignmentY 8-6
- que no son de la interfaz de usuario 2-8
- redimensionar 5-3
- serialización 7-7, 7-8
- Swing
  - comparación 1-5
  - trasladar 5-3
  - ver nombres de clase 3-6
- VerticalFlowLayout 8-20
- visuales 2-7
- Window 1-4
- Window, Frame, Dialog, Panel 1-4
- XYLayout 8-14
- comprobar
  - interfaces de usuario 5-12
- configuración de teclado
  - diseñador 2-9
- conjunto de recursos
  - almacenar valores de propiedad String 7-11
- contenedores 8-7
  - colocar la interfaz de usuario en la pantalla 8-10
- componentes 5-5
- descripción general 1-3
- dimensionar automáticamente 8-8
- dimensionar explícitamente 8-9
- GridBagLayout
  - añadir componentes 8-32
  - área de visualización 8-26
  - diseñar visualmente 8-30



- ejemplo 8-45
- modificar el código fuente para su diseño 8-30
- visualizar la rejilla 8-34
- GridLayout 8-21
- propiedad preferredSize 8-8
- seleccionar diseño 8-1
- Window, Frame, Dialog, Panel 1-4
- controles 2-7
- visuales 2-7
  - Consulte* diseñador de interfaces de usuario
  - Consulte* diseñador de menús
- convenciones de la documentación 1-5
- convenciones de plataformas 1-7
- cortar, copiar y pegar
  - componentes JavaBean 3-4
- creación de interfaces de usuario 8-51
- crear menús 6-1, 6-4
  - añadir elementos 6-5
  - configuración de teclado 6-6
  - desactivar elementos 6-6
  - elementos botones de radio 6-7
  - elementos seleccionables 6-7
  - Inserción de separadores 6-6
  - menú emergente 6-11
  - submenús 6-9
  - sucesos de menú 6-10
  - trasladar a submenús 6-10
  - trasladar elementos 6-8
- cuadros de diálogo
  - añadir
    - a la interfaz 5-6, 5-9
    - al proyecto 5-7
  - crear desde un fragmento de código 5-7
  - JFileChooser 4-8
  - llamada desde un elemento de menú 4-8
  - utilización de uno que no es un bean 5-7

## D

---

- definir
  - clase Applet 1-4
  - el tamaño del contenedor
    - automáticamente 8-8
    - explícitamente 8-9
- desactivar elementos de menú 6-6
- deshacer/volver a hacer
  - en el diseñadores 3-5
- desplazamiento
  - en el diseñador 2-9
- dimensionar contenedores
  - para portabilidad 8-9
  - por medio de pack() 8-8
  - usar setSize() 8-9
- diseñadores

- Consulte* diseñador de interfaces de usuario 2-1
- accesibilidad 2-8
- añadir componentes 3-3
- árbol de componentes 2-6, 3-1
- asignar valores a propiedades 3-8
- barra de estado 2-3
- borrar componentes 3-5
- componentes rojos 7-7
- configuración de teclado 2-9
- cortar, copiar y pegar componentes 3-4
- de acceso a datos 2-8
- de interfaces de usuario 2-7
  - agrupar componentes 5-5
  - añadir
    - componentes 5-3
    - componentes de base de datos 5-9
    - componentes no visuales 5-6
    - cuadros de diálogo 5-7
    - menús 5-6
  - desplazamiento y redimensionamiento de componentes 5-3
  - seleccionar componentes 5-2
  - serializar componentes 7-7, 7-8
  - utilizar personalizadores 7-10
- de menús 2-7, 6-1
  - barra de herramientas 6-3
  - configuración de teclado 6-6
  - desactivar elementos de menú 6-6
  - elementos botones de radio 6-7
  - herramientas 6-3
  - insertar o eliminar elementos 6-5
  - menú emergente 6-11
  - separadores 6-6
  - submenús, crear 6-9
  - trasladar a submenús 6-10
  - trasladar elementos 6-8
  - vincular sucesos 6-10
- deshacer/volver a hacer 3-5
- diseñador
  - de acceso a datos *Consulte* DataExpress
  - de interfaces de usuario 5-1
  - de menús 6-1
  - Consulte* diseñador de menús
- GridBagLayout 12-13
- mover componentes 3-6
- orden de tabulación 2-9
- partes 2-1
- por defecto 2-8
- Presentación de los nombres de clase de los componentes 3-6
- requisitos de diseño visual 1-1
- superficie de diseño 2-3
- teclas de método abreviado 2-8

- tipos de diseñador 3-3
- tipos de diseñadores visuales 2-7
- visualizar la rejilla 12-13
- diseños
  - añadir personalizados 8-11
  - anidados 8-53
  - arrastrar y soltar 2-3
  - BorderLayout 8-15
  - BoxLayout2 8-21
  - cambiar 8-3, 8-4
  - CardLayout 8-23
  - columnares 8-19
  - combinar columnas y filas 8-21
  - creación de prototipos de interfaces de usuario 8-51
  - de interfaz usuario 1-1
  - de la interfaz de usuario
    - sugerencias 8-52
  - de la interfaz de usuario.
    - sugerencias 8-52
  - diseño por defecto 8-1
  - ejemplos
    - de las propiedades 8-4
    - de restricciones 8-4
  - en filas 8-18
  - FlowLayout 8-18
  - GridBagLayout 8-25
    - con diseñador de interfaces de usuario 8-30
  - GridLayout 8-21
  - interfaz usuario 1-1
  - IU prototipo 8-51, 8-52
  - null 8-15
    - diferencias respecto a los diseños
      - XYLayout 8-2
  - OverlayLayout 8-25
  - OverlayLayout2 8-25
  - PaneLayout 8-48
    - Consultar* PaneLayout
  - por defecto 8-1
  - portables 8-8
  - rejillas de datos 8-21
  - seleccionar en el Inspector 8-1
  - suministrados con JBuilder 8-12
  - tutorial 11-1
  - VerticalFlowLayout 8-19
  - visual 12-13
    - agrupar componentes 5-5
    - contenedores 1-3
    - en JBuilder 1-1
    - JavaBeans 1-3
    - paleta de componentes 2-4
    - requisitos 1-1
    - usar el diseñador 2-1
    - uso de la superficie de diseño 2-3

- utilización de los asistentes 1-2
- XYLayout 8-8, 8-13

## E

- editores
  - de GridBagConstraints 8-33
  - de JavaBeans
    - Inspector 2-6
  - de propiedades
    - GridBagConstraints 8-33
  - personalizadores 7-10
- ejemplos
  - código fuente de GridBagLayout 8-29
  - llamada desde un elemento de menú 6-10
  - llamar a un cuadro de diálogo desde un
    - menú 4-8
  - Serialización de un objeto this 7-8
- espacio
  - FlowLayout 8-19
  - VerticalFlowLayout 8-20
- exponer propiedades en el Inspector 3-8

## F

- FlowLayout 8-18
  - espacio 8-19
  - orden de componentes 8-19
- Forte
  - migrar archivos a JBuilder A-2
- fuentes
  - convenciones de la documentación 1-5

## G

- gestores de diseño 8-1
  - Consulte* diseños
    - añadir personalizados 8-11
    - descripción general 8-1
    - diseño por defecto 8-1
    - no asignados 8-15
    - seleccionar en el Inspector 8-5
- getAlignmentX() 8-6
- getAlignmentY() 8-6
- getMaximumSize() 8-6
- getMinimumSize() 8-6
- getPreferredSize() 8-6
- GridBagConstraints 8-28, 12-65
  - ancla 8-35
  - definición de las restricciones 8-35
  - encuadres 8-38
  - escribir el código fuente a mano 8-29
  - expansión 8-36
  - gridheight 8-37
  - gridwidth 8-37
  - gridx 8-37

- gridy 8-37
- ipadx 8-40
- ipady 8-40
- modificar 8-34
- weightx 8-41
- weighty 8-41
- GridBagLayout 8-25
  - agrupar componentes 12-12
  - añadir componentes 12-59
  - conversión 8-31
  - definición 12-3
  - descripción general 12-3
  - diseñar visualmente 12-13
  - ejemplo 12-3
  - menú contextual de los componentes 8-34
  - simplificado 12-8
  - Sugerencias y técnicas 12-42
  - tutorial 12-1
  - ventajas 12-8
- GridLayout 8-21
  - Columnas y filas 8-22
  - Espacio 8-22
- grupos de noticias 1-8
  - Borland y JBuilder 1-8
  - public 1-8
  - Usenet 1-8

## H

---

- herramientas
  - Diseñador de menús 6-3

## I

---

- iconos
  - árbol de componentes 3-6
  - paleta de componentes 7-4
- importar
  - desde otros IDE A-1
- informar sobre errores 1-9
- Inspector 2-6, 3-6
  - almacenar cadenas 7-11
  - asignar valores a propiedades 3-8
  - definición de propiedades compartidas 3-9
  - editores de propiedades 3-10
  - en privado 3-10
  - exponer diferentes niveles de propiedades 3-8
  - Resaltado de los valores de las propiedades 3-7
- instalar
  - componentes en la paleta de componentes 7-2
- interfaces de usuario 5-1
  - agrupar componentes 5-5
  - añadir
    - componentes 3-3, 5-3

- componentes de base de datos 5-6, 5-9
- cuadros de diálogo 5-6
  - menús 5-6, 6-4
- anidación 8-53
- aspecto 5-10
- colocar en la pantalla 8-10
- comprobar durante la ejecución 5-12
- cortar, copiar y pegar componentes 3-4
- creación de un prototipo en el diseñador 8-51
- crear con asistentes 1-2
- desplazamiento y redimensionamiento de componentes 5-3
- diseñador visual 2-7
- diseño visual 2-7
  - Consulte* diseñadores de interfaces de usuario
- insertar o eliminar elementos de menú 6-5
- seleccionar componentes 5-2
- tutorial 11-1
- IU prototipo
  - utilizar XYLayout 8-52

## J

---

- JavaBeans 1-3
  - Consulte* componentes
  - codificar visualmente *Consulte* diseñador
  - contenedores 1-3
  - paleta de componentes 2-4
  - Selección de Beans 2-4
- JBuilder
  - bibliotecas de componentes 1-5
  - grupos de noticias 1-8
  - informar sobre errores 1-9

## L

---

- LayoutManager2 12-3
- líneas de rejilla
  - visualizar en GridBagLayout 8-34
- lista desplegable
  - sin valores de propiedades 3-10
- localizar valores de la propiedad String 7-11

## M

---

- manejadores de sucesos 4-1
  - borrar 4-3
  - crear 4-6
  - crear para suceso por defecto 4-2
  - ejemplo de botón 4-7
  - ejemplo de diálogo 4-8
  - ejemplos 4-7
  - vincular a componentes 4-2
- maximumSize 8-6
- menús

- añadir a la interfaz 5-6, 5-9
- configuración de teclado 6-6
- anidados
  - crear
    - Consulte* submenús
- crear 6-9
- crear *Consultar* crear menús
- desplegables
  - crear 6-9
- diseño 6-1
- emergente 6-11
- insertar o eliminar elementos 6-5
- terminología 6-2

migrar archivos

- de otras herramientas de desarrollo en Java A-1

minimumSize 8-6

mover componentes en el diseñador 3-6

## N

Nivel de exposición de la propiedad 3-8

## O

objeto 'this'

- serialización 7-8

orden de tabulación

- diseñador 2-9

OverlayLayout 8-25

## P

pack() 8-7

- dimensionar contenedores automáticamente 8-8
- utilizar en código fuente 8-10

paleta de componentes 2-4

- añadir
  - componentes 7-2
  - páginas 7-5
- eliminar componentes 7-6
- eliminar fichas 7-6
- gestionar 7-1
- imágenes de botón 7-4
- reorganizar 7-6
- Selección de Beans (botón) 2-4

PaneLayout 8-48

- añadir componentes 8-49
- crear en el diseñador de interfaces de usuario 8-49
- ubicación y tamaño del panel 8-51
- variables
  - PaneConstraint 8-48

paneles

- añadir al contenedor CardLayout 8-23

- anidación 8-53, 12-12
- cambiar en CardLayout 8-23
- de estructura
  - árbol de componentes 3-1
- y diseños anidados 12-12
  - tutorial 11-1

personalizadores 7-10

personalizar

- añadir a paleta de componentes 7-5

portabilidad

- dimensionar contenedores 8-9

preferredSize 8-6, 8-8

propiedades

- alignmentX 8-6
- alignmentY 8-6
- de los diseños
  - cambiar 8-5
  - ejemplos 8-4
- definición
  - en el Inspector 3-8
  - en varios componentes 3-9
- definir 3-6, 3-10
- diseños 8-5
- exponer como variable de clase 3-8
- modificar 3-6
- modificar diseños 8-3
- valores de las propiedades en el Inspector 3-7

pulsaciones de teclas

- teclas de método abreviado del diseñador 2-8

## R

redimensionar componentes 5-3

rejilla

- presentar en el diseñador 12-13

Resaltado de los valores de las propiedades 3-7

restricciones

- ancla 12-65
- BorderLayout 8-15
- CardLayout 8-23
- configuración en el diseñador UI 12-42
- de ancla 8-35, 12-65
  - configuración en el diseñador 12-42
- definir 8-35
- de diseños 8-4
  - cambiar en GridBagLayout 8-34
  - configurar con el Editor de GridBagConstraints 8-33
  - ejemplos 8-4
  - rejillas 8-28
- de expansión 8-36, 12-66
  - configuración en el diseñador 12-43
- especificar 8-36
- de GridBagLayout
  - modificar mediante ratón 12-51

- de gridx 8-37
  - especificar 8-38
- de gridy 8-37
  - especificar 8-38
- de ipadx 8-40
  - definir 8-40
- de ipady 8-40
  - definir 8-40
- de los Encuadres 8-38, 12-66
  - configuración en el diseñador 12-43
  - definir 8-39
- de peso 12-69
  - configuración en el diseñador 12-47
  - ejemplos 12-70
  - weightx 8-41
  - weightx, definir 8-42
  - weighty 8-41
  - weighty, definir 8-42
- de rejilla
  - gridwidth, gridheight 12-67
  - gridx, gridy 12-68
- de tamaño adicional
  - configuración en el diseñador 12-45
  - ipadx, ipady 12-45, 12-68
- definir BorderLayout 8-16
- encuadres 12-66
- expansión 12-66
- FlowLayout 8-18
- GridBagConstraints 12-65
- GridBagLayout 8-25, 8-28, 8-33, 8-34, 8-35
- gridheight 8-26, 8-37
  - especificar 8-37
- GridLayout 8-21
- gridwidth 8-26, 8-37
  - especificar 8-37
- gridwidth, gridheight 12-67
  - configuración en el diseñador 12-44
- gridx, gridy 12-68
  - configuración en el diseñador 12-46
- ipadx, ipady 12-68
  - configuración en el diseñador 12-45
- modificar
  - mediante ratón 12-51
  - un diseño 8-4, 8-6
- OverlayLayout 8-25
- PaneLayout 8-48
- pesos 12-69
- relleno 12-68

## S

- scope
  - tipos de datos de propiedades 3-10
- Selección de Beans
  - añadir componentes 2-4

- separadores
  - insertar en menús 6-6
- serializar
  - componentes 7-7, 7-8
  - objetos
    - alternativas 7-8
- setSize() 8-7
  - dimensionar contenedores explícitamente 8-9
  - utilizar en código fuente 8-10
- SplitPanel 8-49
  - ubicación y tamaño del panel 8-51
- submenús
  - crear 6-9
- sucesos 4-1
  - adaptadores *Consulte* adaptadores
    - de sucesos
      - añadir sucesos de botón 4-7
      - código generado 4-3, 4-6
      - creación y modificación 3-6
    - de botón 4-7
    - de menú
      - crear 6-10
      - ejemplo 6-10
      - ejemplo de vínculo a código 4-8
    - de menú actionPerformed() 1 6-10
    - ejemplo de diálogo 6-10
    - sucesos de elementos de menú 6-10
    - vincular a elementos de menú 4-8
  - de menú
    - crear
      - Consulte* crear menús
- superficie de diseño 2-3

## T

- tamaño
  - de la ventana de la interfaz en ejecución 8-7
  - de pantalla
    - interfaz de usuario durante la ejecución 8-7
- tareas de diseño 5-1
- teclas
  - aceleradoras
    - añadir a menús 6-6
  - de método abreviado
    - añadir a menús 6-6
    - diseñador 2-8
- terminología
  - diseñar menús 6-2
- tipos
  - de adaptador 4-6
  - de datos de objetos
    - añadir tipos de datos de objetos al Inspector 3-10
  - de diseñador
    - acceder 3-3

- tratamiento de sucesos *Consulte* manejadores de sucesos
- tutoriales
  - creación de una interfaz de usuario sencilla 9-1
  - crear un editor de texto 10-1
  - diseños anidados 11-1

## U

- Usenet, grupos de noticias 1-8

## V

- valores de la propiedad String
  - almacenar en ResourceBundle 7-11
- variables
  - exponer como clase 3-8
- VerticalFlowLayout 8-19, 8-20
  - espacio 8-20
  - Orden de los componentes 8-21
  - Relleno horizontal 8-20
  - Relleno vertical 8-20
- VisualAge
  - migrar archivos a JBuilder A-1
- VisualCafe
  - migrar archivos a JBuilder A-2
- volver a hacer/deshacer
  - en el Diseñador 3-5

## W

- windows
  - colocar en la pantalla 8-10

## X

- XYLayout 8-8, 8-13
  - componentes 8-14
  - creación de prototipos 8-52
  - diferencias respecto a los diseños null 8-2
- XYLayout
  - opciones de alineación 8-14