Seminario de Java

Programación Orientada a Objetos
Curso 2006/2007

Contenido

1. Introducción
2. Primeros pasos con Java. El entorno Eclipse
3. La sintaxis del lenguaje Java
4. Clases y objetos
5. Cadenas y Entrada/Salida
6. Herencia, clases abstractas
7. Genericidad e interfaces
8. Colecciones e iteradores

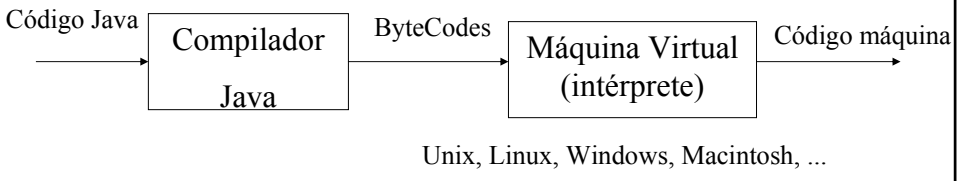
Contenido

1. **Introducción**
2. **Primeros pasos con Java. El entorno Eclipse**
3. La sintaxis del lenguaje Java
4. Clases y objetos
5. Cadenas y Entrada/Salida
6. Herencia, clases abstractas
7. Genericidad e interfaces
8. Colecciones e iteradores

1. Introducción.

- Sun Microsystems, Agosto 1995
- Navegadores con contenido interactivo
- Tecnología de implementación:
  - ByteCode + Máquina Virtual
  - Compilado e **interpretado**

⇒ - **portable**
- lento



Unix, Linux, Windows, Macintosh, ...

1.1 Breve historia de Java

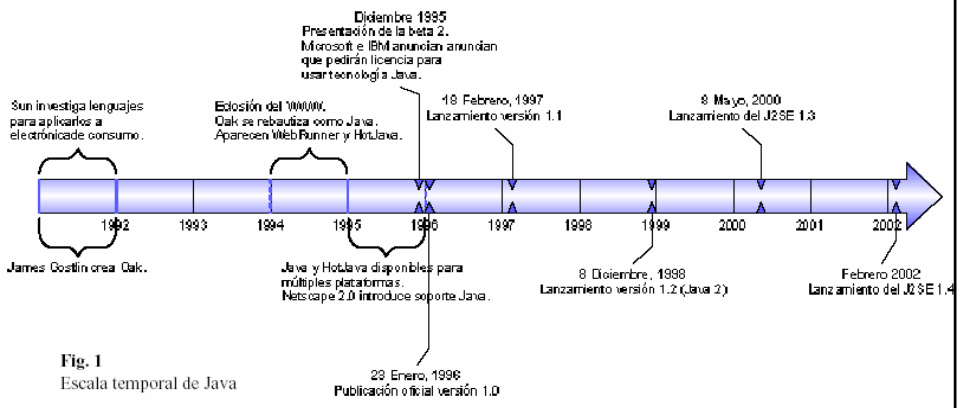


Fig. 1
Escala temporal de Java

- Verano de 2004 lanzamiento de la versión **1.5** renombrada **5.0**

1.2 Características de Java (1/2)

Artículo publicado en 1996 por James Gosling y Henry McGilton (**http://java.sun.com/docs/white/langenv/**)

- **Simple:**
  - En comparación con C++ no necesita ficheros cabecera, elimina punteros, muchas librerías disponibles, no hay que gestionar la memoria
- **Orientado a objetos:**
  - Herencia, polimorfismo
- **Distribuido:**
  - Orientado al desarrollo de aplicaciones relacionadas con Internet
- **Robusto:**
  - Lenguaje fuertemente tipado (detectar errores en tiempo de compilación)
  - Mecanismo de excepciones (control de errores en tiempo de ejecución)
  - *Garbage collection* (no hay que preocuparse de la gestión de memoria)

1.2 Características de Java (2/2)

- **Seguro:**
  - Evitar dañar la integridad del sistema cliente en aplicaciones en entornos de red o distribuidos
- **Arquitectura neutral:**
  - Bytecodes generados independientes de la plataforma
- **Portable**
- **Interpretado**:
  - Los bytecodes son interpretados por una máquina virtual dependiente de la plataforma
- **Alto rendimiento:**
  - Compiladores *Just In Time*
- **Multihilo** (*multithreading*)
- **Dinámico:**
  - Las librerías pueden extenderse sin afectar a los clientes

1.3 ¿Qué se mueve alrededor de Java?

- **JDBC API** *Java Database Connectivity*
- **Java RMI** *Remote Method Invocation* aplicaciones distribuidas
- **Java IDL** *Interfaz Definition Language.* Puente de compatibilidad con el modelo estándar de objetos CORBA
- **JavaBeans** Especificación de componentes basado en Java

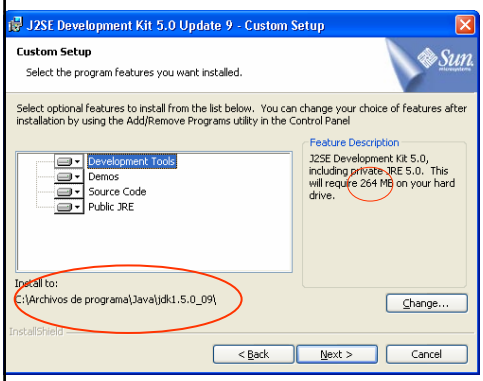
2.Primeros pasos con Java

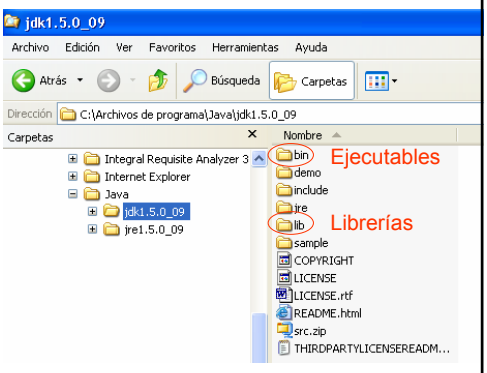
http://java.sun.com

- **SDK** (*Software Development Kit*)
  - Necesario para crear (desarrollar y compilar) programas
- **JRE** (*Java Runtime Environment*)
  - Necesario para ejecutar un programa Java
  - SDK = JRE + herramientas de desarrollo
- La tecnología Java se organiza en subáreas:
  - **J2SE** (*Java 2 Platform, Standard Edition*)
  - **J2EE** (*Java 2 Platform, Enterprise Edition*)
    - SDK + soporte para servicios web y componentes
  - **J2ME** (*Java 2 Platform, Micro Edition*): entorno flexible y robusto para aplicaciones que funcionan en dispositivos tales como teléfonos móviles, PDA, …
  - **Java Web Services**: aplicaciones basadas en la web que utiliza estándar XML y protocolos de transporte para intercambiar datos con los clientes que le invocan.

Instalando J2SE 5.0

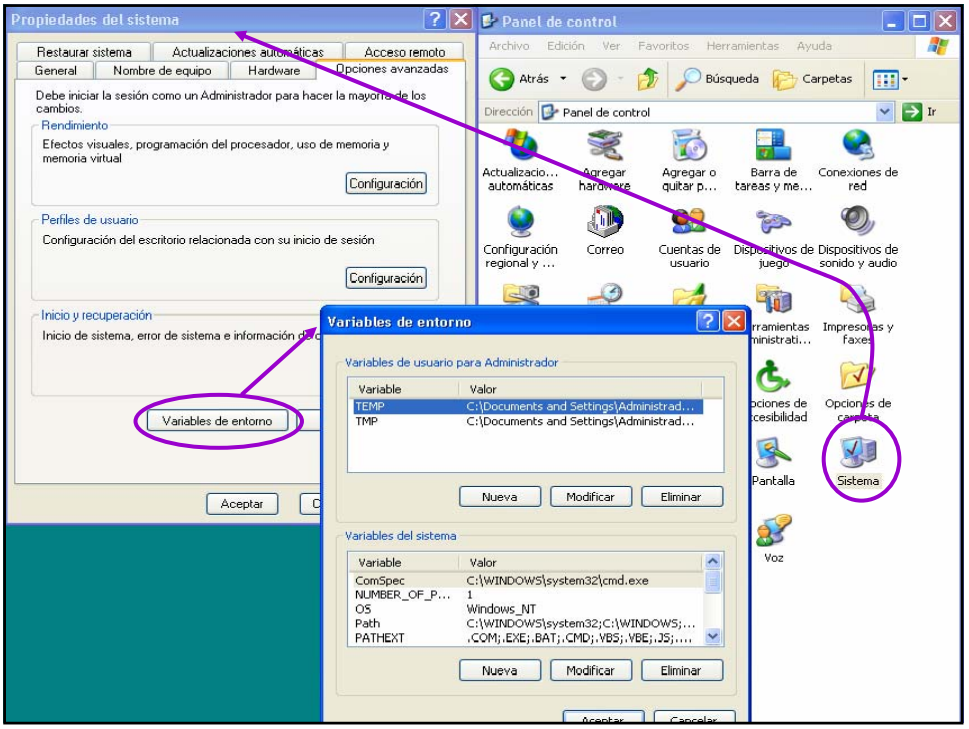
- Descargar de http://java.sun.com/j2se/1.5.0/download.jsp





Configurar variables de entorno

- **JAVA_HOME**
  - Directorio en el que está instalado el J2SE SDK
  - Por ejemplo, `c:\set JAVA_HOME = c:\Archivos de programa\Java\jdk1.5.0_09`
- **CLASSPATH**
  - Ruta de acceso a las API's de Java
  - `c:\set CLASSPATH=.;%JAVA_HOME%\lib\tools.jar;`
    `%JAVA_HOME%\lib\dt.jar;`
  - `.;` Carpeta o directorio actual de trabajo
- **PATH** del sistema
  - Directorio donde se encuentran los ejecutables de java
  - `c:\set PATH=%JAVA_HOME%\bin;%PATH%`
- Cambiar las variables de entorno en Windows
  - Panel de Control >> Sistema >> Opciones avanzadas >> Variables de entorno



Compilación y ejecución en la línea de comandos

- Compilación:

```
c:\javac Fichero.java  ENTER  ⇨ Fichero.class
```

- **Ejecución:**

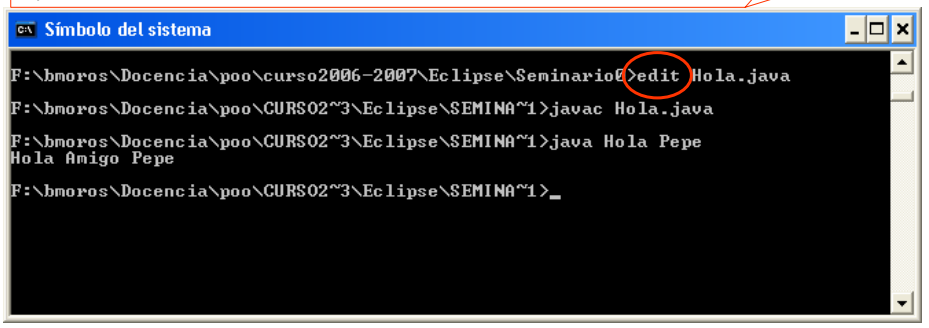
```
c:\java Fichero [argumentos]
```

- `Fichero.java` es un fichero de texto que contiene la clase principal (clase que contiene el main)
  - `public static void main (String [] args)`

Ejercicio: Compila y ejecuta desde la línea de comando

```
//Fichero de texto Hola.java

public class Hola {
  public static void main(String[] args) {
      System.out.println("Hola amigo "+args[0]);
  }
}
```



Ejecutables en Java

- Existen algunos compiladores pero se pierde la portabilidad
- Soluciones:
  - Crear un **.bat** ejecutable que contenga la llamada:

    ```
    java nombre_clase_prinpal
    ```

  - Crear un **.jar** ejecutable
    - Comprimir ficheros de la aplicación en un JAR
    - Modificar la metainformación indicando la clase principal:
      - META-INF/MANIFEST.MF
      - añadir la entrada: `Main-Class: clasePrincipal`
    - Sintaxis creación: **jar** `cvf destino.jar *.class`

      `c = nuevo; v = verbose; f = indicamos el nombre destino`
    - Ejecutar: `java -jar destino.jar [argumentos]`

Ficheros jar y Manifest

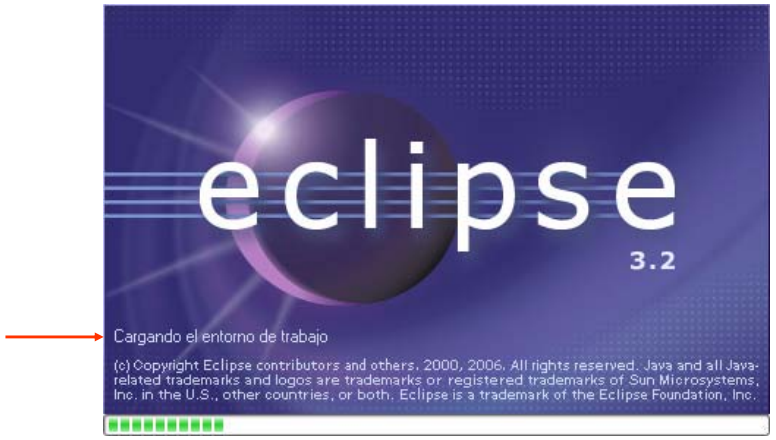
- Definir un fichero con la nueva entrada de Manifest (llámese `myManifest.mf`)

  ```
  Main-Class: clasePrinpal
  ```

- Crear el fichero jar utilizando un fichero Manifest existente y comprimiendo todos los ficheros del directorio `classes`:

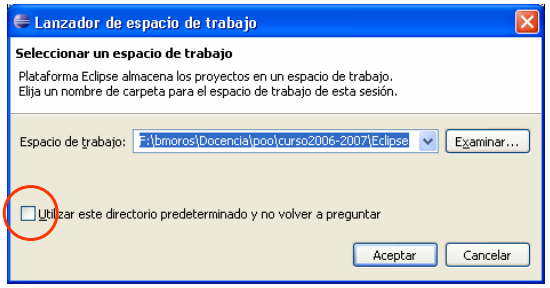
  ```
  jar cvfm destino.jar myManifest.mf -C classes/ .
  ```

- Los ficheros se deben colocar en el mismo orden en el que se ponen los modificadores "f" y "m"
- Ejecutar: **`java -jar destino.jar [argumentos]`**



Espacio de trabajo (1/2)

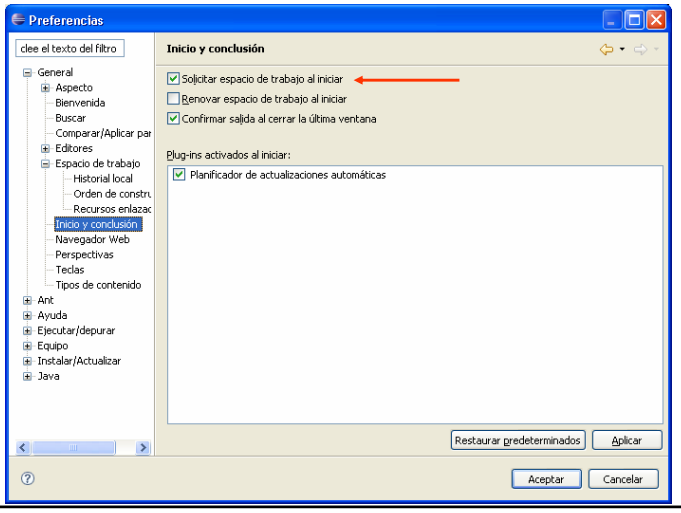
- Los proyectos se almacenan por defecto en el directorio especificado como el "***espacio de trabajo***".
- ¡¡Cualquier elemento que se importe al espacio de trabajo **se duplica**!!
- Recomendación: Establecer como espacio de trabajo un directorio propio para las prácticas.



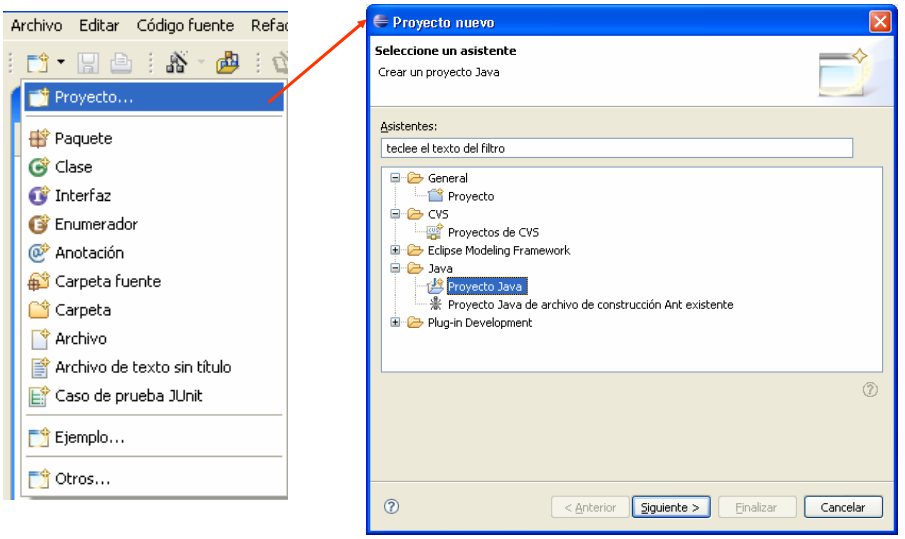
No marcar!!

Espacio de trabajo (2/2)

- Restaurar la solicitud del espacio de trabajo al inicio.
- Menú Ventana >>Preferencias



Nuevo proyecto Java (1/2)

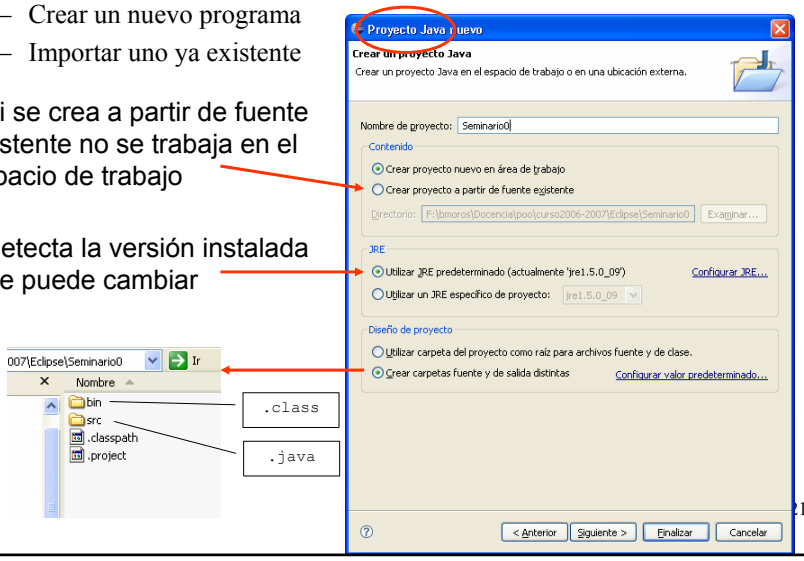


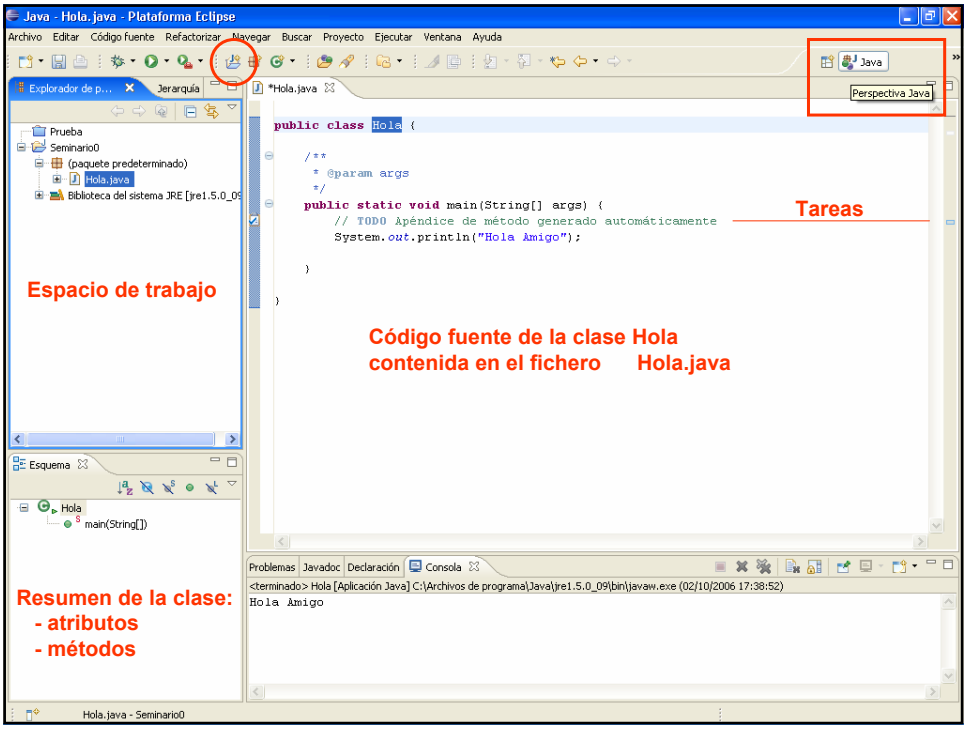
Nuevo proyecto Java (2/2)

- Todo tiene que estar dentro de un ***proyecto***:
  - Crear un nuevo programa
  - Importar uno ya existente

• Si se crea a partir de fuente existente no se trabaja en el espacio de trabajo

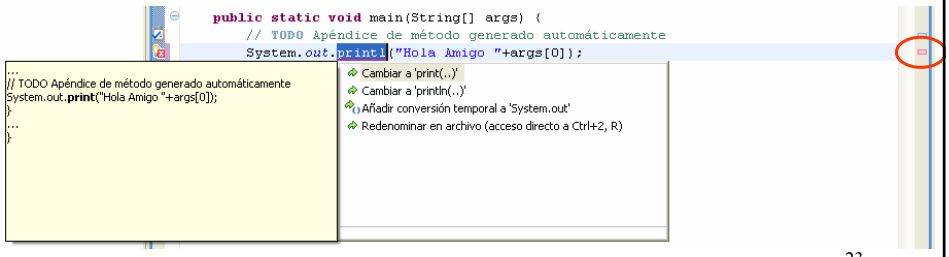
• Detecta la versión instalada
• Se puede cambiar





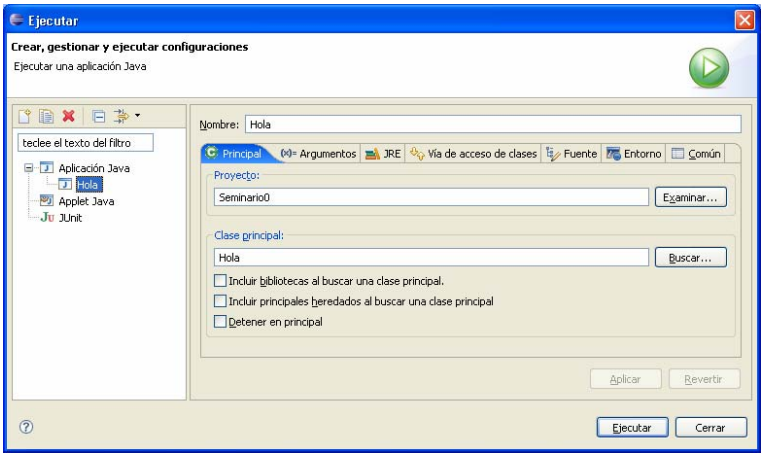
Errores antes de la compilación

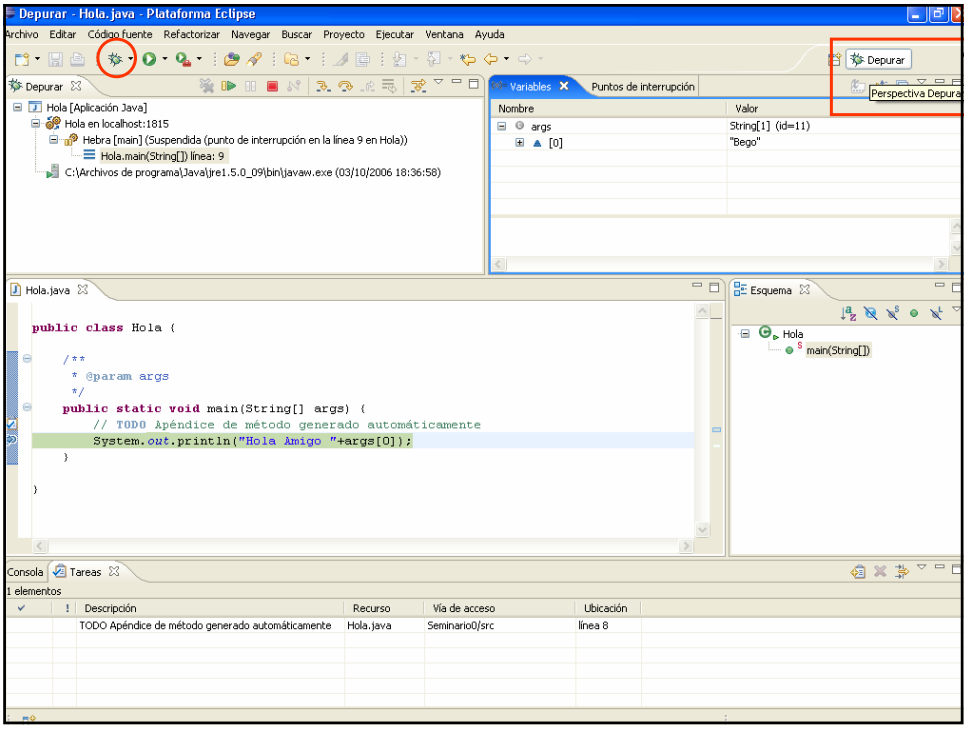
- Errores en tiempo real subrayando el código incorrecto.
- Autocorregir:
  - Ctrl + 1
  - Icono de bombilla >> clic en el icono



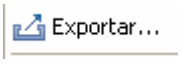
Ejecutar un proyecto

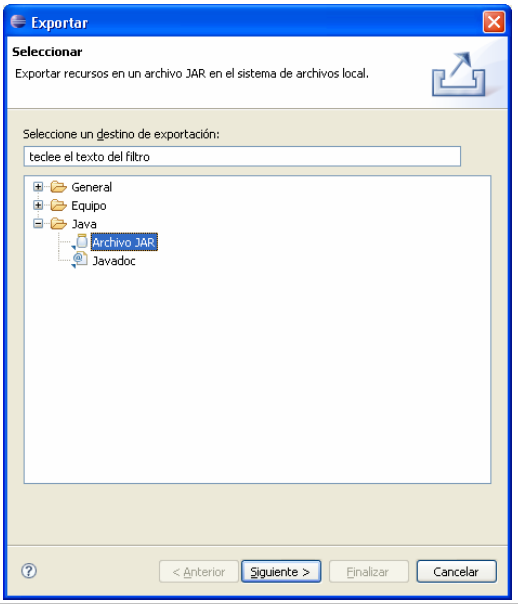
- Indicar la clase principal, aquella que contiene el main que debe ejecutarse.
- Ejecutar >> Ejecutar … >> Aplicación Java




Depurar - Hola.java - Plataforma Eclipse
Archivo Editar Código fuente Refactorizar Navegar Buscar Proyecto Ejecutar Ventana Ayuda
Depurar
Perspectiva Depura
Depurar
Hola [Aplicación Java]
Hola en localhost:1815
Hebra [main] (Suspendida (punto de interrupción en la línea 9 en Hola))
Hola.main(String[]) línea: 9
C:\Archivos de programa\Java\jre1.5.0_09\bin\javaw.exe (03/10/2006 18:36:58)
Variables
Puntos de interrupción
Nombre
Valor
args
String[1] (id=11)
[0]
"Bego"
Hola.java
public class Hola {
/**
* @param args
*/
public static void main(String[] args) {
// TODO Apéndice de método generado automáticamente
System.out.println("Hola Amigo "+args[0]);
}
}
Esquema
Hola
main(String[])
Consola
Tareas
1 elementos
Descripción
Recurso
Vía de acceso
Ubicación
TODO Apéndice de método generado automáticamente
Hola.java
Seminario0/src
línea 8

Crear .jar


Exportar...


Exportar
Seleccionar
Exportar recursos en un archivo JAR en el sistema de archivos local.
Seleccione un destino de exportación:
teclee el texto del filtro
General
Equipo
Java
Archivo JAR
Javadoc
< Anterior
Siguiente >
Finalizar
Cancelar