2º Administración de Sistemas Informáticos y Red
Módulo: *Implantación de Aplicaciones Web*

Ejercicios de JAVA.

Analiza los siguientes aspectos antes de codificar los ejercicios:

- ***Paradigma Orientado a Objetos - Java***
- ***Elementos del Lenguaje.***
- ***Sentencias de Control.***
- ***Jerarquía de Excepciones.***
- ***Excepciones try {} catch(){}.***
- ***Clase String y Clases envolventes (Double, Integer….)***
- ***Vectores y Matrices.***
- ***Paquetes en java: paquete lang y paquete útil;***
- ***¿Qué me proporciona la clase System?.***
- ***Analiza las siguientes clases que proporciona java: String, Integer, Double.***
- ***Webgrafías, referencias ….***

Enunciado:

1. Leer un número y mostrar por la salida estándar si dicho número es o no es par.
2. Leer 2 números y mostrar el producto de ellos.
3. Leer 2 números y determinar el mayor de ellos.
4. Leer 3 números y mostrar el mayor de ellos.
5. Leer un número y mostrar su tabla de multiplicar.
6. Leer una secuencia de 30 números y mostrar la suma y el producto de ellos.
7. Leer una secuencia de números, hasta que se introduce un número negativo y mostrar la suma de dichos números.
8. Leer dos números y realizar el producto median sumas.
9. Leer dos números y realizar la división mediante restas mostrando el cociente y el resto.
10. Leer una secuencia de números y mostrar su producto, el proceso finalizará cuando el usuario pulse a la tecla F.
11. Lee una secuencia de números y determina cual es el mayor de ellos.
12. Dado un número mostrar su valor en binario.
13. Generar enteros de 3 en 3 comenzando por 2 hasta el valor máximo menor que 30. Calculando la suma de los enteros generados que sean divisibles por 5.
14. Calcular la media de una secuencia de números.
15. Generar los N primeros términos de la serie de Fibonacci. El valor N(entero y positivo) deberá ser leído por el teclado. En esta serie los dos primeros números son 1, y el resto se obtiene sumando los dos anteriores: 1,1,2,3,5,8,13,21…
16. Leer una secuencia se números y mostrar cuales de ellos es el mayor y el menor, el proceso finalizará cuando se introduzca un número impar.
17. Leer una secuencia de números y sumar solo los pares mostrando el resultado del proceso.
18. Leer una secuencia de números y mostrar los 30 primeros pares leídos.
19. Leer una secuencia de números y mostrar la suma de los 30 números que ocupan posiciones de lectura par.
20. Leer un número y determinar su factorial.
21. Leer un número y determinar si es o no es primo.
22. Leer una secuencia de 30 números y mostrar la suma de los primos.
23. Leer una secuencia de 30 números y mostrar la suma de su factorial.
24. Calcular el valor del número $E=\sum(1/n!)$.

25. Implementar un programa que sea capaz de calcular el resultado de aplicar la fórmula siguiente (n i)=n! / (i! * (n-i)!).
26. Leer una secuencia de números y mostrar la suma de los pares y el producto de los que son múltiplo de 5.
27. Leer una secuencia de números y determinar el mayor de los pares leidos.
28. Leer una secuencia de números y mostrar el mayor de los múltiplos de 5 leídos y el menor de los múltiplos de 3 leídos.
29. Leer una secuencia de letras y mostrar la suma de sus códigos ASCII.
30. Dado un vector de 5 enteros actualizar cada pasición de dicho vector con un número leído.
31. Leer una secuencia de 20 números almacenarlos en un vector y mostrar la posición donde se encuentra el mayor valor leído.
32. Dado dos vectores A y B de 15 elementos cada uno, obtener un vector C donde la posición i se almacene la suma de A[i]+B[i].
33. Dado dos vectores A y B de 15 elementos cada uno, obtener un vector C donde la posición i se almacene la suma de A[i]+B[i] y mostrar el mayor de los C[i].
34. Dado una secuencia de número leídos y almacenados en un vector A mostrar dichos números en orden.
35. Dado una secuencia de número leídos y almacenados en un vector A y un número leído determinar si dicho número se encuentra o no en el vector.
36. Leer una secuencia de 20 números y almacenar en un vector sus factoriales.
37. Leer 20 números y almacenarlos de manera ordenada en un vector.
38. Dado dos matrices A y B obtener la suma.
39. Dado una matriz determinar la posición (i,j) del mayor.
40. Dado una matriz determinar la posición (i,j) del mayor y menor.
41. Leer un número y una letra si la letra es B mostrar el valor en binario, si es O en octal y si es H en hexadecimal.
42. Leer una secuencia de 20 números almacenarlos en un vector A[1..20] y mostar la suma de los elementos que ocupan posiciones pares y el mayor de los que ocupan posiciones impares.
43. Dada una matriz A[1..4][1..5] realiza la ordenación de la misma.
44. Dada una matriz A[1..4][1..5] realiza el proceso de ordenar solo por filas.
45. Dado un vector de números determina aquellos que sea primos.

Resolución:

```
import java.io.*;
public class Ejercicio1{

   public static void main(String[] args) throws IOException
   {
      BufferedReader in=new BufferedReader(new InputStreamReader(System.in));
      System.out.println("Introduce un numero: ");

      try{
         int num = Integer.valueOf(in.readLine().trim()).intValue();
         if (num == 0)
            throw new Ejercicio1Exception("Error, numero nulo detectado");
          if ((num % 2) == 0)
            System.out.println("El numero introducido es PAR");
         else
            System.out.println("El numero introducido es IMPAR");
      }
      catch(Ejercicio1Exception e){
```

```
            System.out.println(e.getMessage());
        }
        catch(NumberFormatException e1) {
            System.out.println(" Error.."+ e1.getMessage());
        }
        catch(Exception e2) {
            System.out.println(e2.getMessage());
        }

        }
    }

public class Ejercicio1Exception extends Exception{

    /** Creates a new instance of Ejercicio1Exception */
    public Ejercicio1Exception(String mensaje) {
        super(mensaje);
    }

}

import java.io.*;
class Ejercicio2Exception extends Exception{}

public class Ejercicio2 {

    public static void main(String[] args) throws Ejercicio2Exception,IOException{
        BufferedReader in=new BufferedReader(new InputStreamReader(System.in));

        System.out.println("Introduce un numero: ");
        int num1 = Double.valueOf(in.readLine().trim()).intValue();
        System.out.println("Introduce otro numero: ");
        int num2 = Double.valueOf(in.readLine().trim()).intValue();

            if (num1 <= 0){
                try{
                    throw new Ejercicio2Exception();
                }catch(Ejercicio2Exception e){
                    System.out.println("Error, numero nulo detectado");
                }
            }
            else{
                int producto = num1*num2;
                System.out.println("El producto es: "+producto);
            }
    }
}

import java.io.*;
public class Ejercicio3 {

    public static void main(String[] args) throws IOException{
      BufferedReader in=new BufferedReader(new InputStreamReader(System.in));
      try{
            System.out.println("Introduce el primer numero: ");
            int num1 = Integer.valueOf(in.readLine().trim()).intValue();
            System.out.println("Introduce el segundo numero: ");
            int num2 = Integer.valueOf(in.readLine().trim()).intValue();
```

```
        if (num1 < 0)
          throw new Ejercicio1Exception("Error, el primer numero no puede ser negativo");
          else
             if (num2 < 0)
                throw new Ejercicio1Exception("Error, el segundo numero no puede ser
negativo");
        if (num1 > num2)
            System.out.println("El numero mayor es: "+num1);
        else
            if (num2 > num1)
                System.out.println("El numero mayor es: "+num2);
            else
                System.out.println("Los dos numeros son iguales");
        }
     catch(Ejercicio1Exception ex){
        System.out.println(ex.getMessage());
     }
     catch(Exception exc){
        System.out.println("Error, se llama a la clase Excepcion(general)");
     }
   }
}

import java.io.*;
public class Ejercicio4 {

   public static void main(String[] args) throws IOException{
      int mayor = 0;
      BufferedReader in=new BufferedReader(new InputStreamReader(System.in));
     try{
         System.out.println("Introduce el primer numero: ");
         int num1 = Integer.valueOf(in.readLine().trim()).intValue();
         System.out.println("Introduce el segundo numero: ");
         int num2 = Integer.valueOf(in.readLine().trim()).intValue();
         System.out.println("Introduce el tercer numero: ");
         int num3 = Integer.valueOf(in.readLine().trim()).intValue();

         if (num1 < 0)
            throw new Ejercicio1Exception("Error, el primer numero no puede ser negativo");
         else
            if (num2 < 0)
               throw new Ejercicio1Exception("Error, el segundo numero no puede ser negativo");
            else
               if (num3 < 0)
                  throw new Ejercicio1Exception("Error, el tercer numero no puede ser negativo");

         if (num1 > num2)
            mayor = num1;
         else
            if (num2 > num1)
               mayor = num2;
         if (num3 > mayor)
            mayor = num3;
         System.out.println("El numero mayor es: "+mayor);
     }
     catch(Ejercicio1Exception ex){
        System.out.println(ex.getMessage());
```

```
      }
      catch(Exception exc){
         System.out.println("Error, se llama a la clase excepcion(general)");
      }
   }
}


import java.io.*;
public class Ejercicio5 {

   public static void main(String[] args) throws IOException{
      BufferedReader in=new BufferedReader(new InputStreamReader(System.in));
      try{
         System.out.println("Introduce un numero: ");
         int num = Integer.valueOf(in.readLine().trim()).intValue();

         if (num < 0)
            throw new Ejercicio1Exception("Error, numero negativo detectado");
         else
            if (num > 10)
               throw new Ejercicio1Exception("Error, numero fuera de rango");

         for(int i=1 ; i<=10 ; i++)
            System.out.println(+num+"x"+i+"="+(num*i));
      }
      catch (Ejercicio1Exception ex){
         System.out.println(ex.getMessage());
      }
      catch (Exception exc){
         System.out.println("Error, se ha llamado a la clase excepcion(general)");
       }
   }
}

import java.io.*;
public class Ejercicio6 {

   public static void main(String[] args) throws IOException{

      int suma=0 , producto=1;
      int vector[];
      BufferedReader in=new BufferedReader(new InputStreamReader(System.in));

      System.out.println("Indica la longitud del vector:");
      try{
         int longitud=Integer.valueOf(in.readLine().trim()).intValue();

         if (longitud < 0)
            throw new Ejercicio1Exception("Error, numero negativo detectado");

         vector=new int[longitud];
      }
      catch (Ejercicio1Exception ex){
         System.out.println(ex.getMessage());
      }

         for(int i=0;i<vector.length;i++){
```

```
            System.out.println("Introduce un numero (pos["+i+"]: ");
            vector[i]=Integer.valueOf(in.readLine().trim()).intValue();
            suma = suma + vector[i];
            producto = producto*vector[i];
         }

      System.out.println("La suma es: "+suma);
      System.out.println("El producto es: "+producto);
   }
}

import java.io.*;
public class Ejercicio7 {

   public static void main(String[] args) throws IOException{

      int num , suma = 0;

      BufferedReader in=new BufferedReader(new InputStreamReader(System.in));

      do{
         System.out.println("Introduce un numero: ");
         num = Integer.valueOf(in.readLine().trim()).intValue();
         if (num < 0)
            break;
         suma = suma+num;
      }while (num >= 0);
      System.out.println("Suma = "+suma);
   }
}

import java.io.*;
public class Ejercicio8 {

   public static void main(String[] args) throws IOException{
      int suma=0 , cont=1;
      BufferedReader in=new BufferedReader(new InputStreamReader(System.in));
      System.out.println("Introduce un numero: ");
      int num1 = Integer.valueOf(in.readLine().trim()).intValue();
      System.out.println("Introduce otro numero: ");
      int num2 = Integer.valueOf(in.readLine().trim()).intValue();

      do {
        suma = suma + num1;
      cont++;
      } while (cont <= num2);

    System.out.println("Producto = "+suma);
  }
}

import java.io.*;
public class Ejercicio9 {

   public static void main(String[] args) throws IOException{
      int resta=0 , cont=1;
      BufferedReader in=new BufferedReader(new InputStreamReader(System.in));
      System.out.println("Introduce un numero: ");
```

```
        int num1 = Integer.valueOf(in.readLine().trim()).intValue();
        System.out.println("Introduce otro numero: ");
        int num2 = Integer.valueOf(in.readLine().trim()).intValue();

        do {
           resta = num1 - resta;
        cont++;
        } while (cont <= num2);

     System.out.println("Producto = "+resta);
    }

}

import java.io.*;
public class Ejercicio10 {

    public static void main(String[] args)  throws IOException{
        int num , producto = 1;

        BufferedReader in=new BufferedReader(new InputStreamReader(System.in));

        try{
           do{
              System.out.println("Introduce un numero: ");
              num = Integer.valueOf(in.readLine().trim()).intValue();

              if (num == 'F')
                 throw new Ejercicio10Exception("Se ha detectado el caracter 'F'");

              producto = producto * num;
           }while (num != 'F');
        }
        catch(Ejercicio10Exception ex){
           System.out.println(ex.getMessage());
        }
        catch(Exception exc){
           System.out.println(exc.getMessage());
        }
    }
}
public class Ejercicio10Exception extends java.lang.Exception {

    public Ejercicio10Exception(String msg) {
        super(msg);
    }
}

import java.io.*;
public class Ejercicio11 {

    public static void main(String[] args) throws IOException{
        int mayor=0;
        BufferedReader in=new BufferedReader(new InputStreamReader(System.in));

        System.out.println("¿Cuantos numeros quieres leer?");
        int n = Integer.valueOf(in.readLine().trim()).intValue();
```

```
        for (int cont=0 ; cont<n ; cont++){
            System.out.println("Introduce un numero: ");
            int num = Integer.valueOf(in.readLine().trim()).intValue();

            if (num > mayor)
                mayor = num;
        }
        System.out.println("El numero mayor es: "+mayor);
    }
}

import java.io.*;
public class Ejercicio12 {

    public static void Menu(){
        System.out.println("Menu de opciones:");
        System.out.println("1.Convertir a Binario");
        System.out.println("2.Convertir a Hexadecimal");
        System.out.println("3.Convertir a Octal");
        System.out.println("4.Salir");
        System.out.println("Introduce una opcion 1-4:");
    }

    public static void main(String[] args) throws IOException {

        BufferedReader in=new BufferedReader(new InputStreamReader(System.in));

        System.out.println("Introduce un numero en decimal:");
        int num = Integer.valueOf(in.readLine().trim()).intValue();

        //do {
            Menu();
            int opcion = Integer.valueOf(in.readLine().trim()).intValue();
            switch (opcion){
                case 1: System.out.println("Decimal");
                        System.out.println(Integer.toBinaryString(num));
                        break;
                case 2: System.out.println("Hexadecimal");
                        System.out.println(Integer.toHexString(num));
                        break;
                case 3: System.out.println("Octal");
                        System.out.println(Integer.toOctalString(num));
                        break;
                case 4: break;
                default: System.out.println("Introduce una opcion 1-4");
                        break;
            }
      //}while (opcion != 4);
    }
}

import java.io.*;
public class Ejercicio13 {

    /** método encargado de comprobar si el numero es divisible por 5 */
    public static boolean Divisibles(int cont) {
        boolean encontrado = false;
        if ((cont % 5)==0){
```

```
            System.out.println("nº: "+cont+" Es divisible");
            encontrado = true;
            return(encontrado);
        }
        else{
            System.out.println("nº: "+cont+" No divisible");
            return(encontrado);
        }
    }

    public static void main(String[] args) throws IOException{
        int suma=0;
        BufferedReader in=new BufferedReader(new InputStreamReader(System.in));

        for(int cont=2 ; cont<30 ; cont=cont+2){
            if (Divisibles(cont) == true)
                suma = suma + cont;
        }
        System.out.println("La suma de los divisibles es: "+suma);
    }
}

import java.io.*;
public class Ejercicio15 {

    public static void main(String[] args) throws IOException{
        BufferedReader in=new BufferedReader(new InputStreamReader(System.in));

        System.out.println("Introduce un numero: ");
        int num = Integer.valueOf(in.readLine().trim()).intValue();

        Matematica a = new Matematica();
        a.Calcula_Fibonacci(num);
        System.out.println(a.Calcula_Fibonacci(num));
    }

}

import java.io.*;
public class Ejercicio16 {

    public static void main(String[] args) throws IOException{
        int mayor = 0 , menor=0;

        BufferedReader in=new BufferedReader(new InputStreamReader(System.in));

        System.out.println("Introduce un numero: ");
        int num = Integer.valueOf(in.readLine().trim()).intValue();
        mayor = num;
        menor = num;
        do{
            System.out.println("Introduce un numero: ");
            num = Integer.valueOf(in.readLine().trim()).intValue();

            if (num > mayor)
                mayor = num;
            if (num < menor)
                menor = num;
```

```
        }while((num % 2) == 0);
        System.out.println("El numero mayor es: "+mayor);
        System.out.println("El numero menor es: "+menor);
    }
}

import java.io.*;
public class Ejercicio17 {

    public static void main(String[] args) throws IOException{
        int suma = 0;

        BufferedReader in=new BufferedReader(new InputStreamReader(System.in));

        System.out.println("¿cuantos numeros vas a introducir?");
        int n = Integer.valueOf(in.readLine().trim()).intValue();

        for(int cont=0 ; cont<n ; cont++){
            System.out.println("Introduce un numero: ");
            int num = Integer.valueOf(in.readLine().trim()).intValue();

            if((num%2)==0)
                suma = suma + num;
        }
        System.out.println("La suma de los numeros pares es: "+suma);
    }
}

import java.io.*;
public class Ejercicio18 {

    public static void main(String[] args) throws IOException{
        int v_pares[];
        int longitud = 30;
        BufferedReader in = new BufferedReader(new InputStreamReader(System.in));

        for(int i=0 ; i<30 ; i++){
            System.out.println("Introduce un numero: ");
            int num = Integer.valueOf(in.readLine().trim()).intValue();

            if ((num%2) == 0)
              v_pares[num]=Integer.valueOf(in.readLine().trim()).intValue();
        }
    }
}



import java.io.*;
public class Ejercicio19 {

    public static void main(String[] args) throws IOException{
        int pares = 0 , suma = 0;
        boolean encontrado = false;

        BufferedReader in=new BufferedReader(new InputStreamReader(System.in));
```

```
      do{
         System.out.println("Introduce un numero: ");
         int num = Integer.valueOf(in.readLine().trim()).intValue();

         Matematica a = new Matematica();
         a.Calcula_Par(num);

         if(a.Calcula_Par(num)!=encontrado){
            suma = suma + num;
            pares++;
         }
      }while(pares < 5);

      System.out.println("La suma de los pares es: "+suma);
   }
}

import java.io.*;
public class Ejercicio20 {

   /*método que calcula el factorial de un número*/
   public static int Factorial(int num){
      int f = 1;
      for (int i=2 ; i <= num ; i++)
         f = f * i;
      return(f);
   }

   public static void main(String[] args) throws IOException{
      BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
      System.out.println("Introduce un numero: ");
      int num = Integer.valueOf(in.readLine().trim()).intValue();
      System.out.println("Su factorial es: "+Factorial(num));
   }
}

import java.io.*;
public class Ejercicio21 {

   /** método que calcula si un número es primo */
   public static boolean Primo(int num) {
      boolean encontrado = false;
      int divisor=2;
      while(divisor < num){
         while( num % divisor != 0)
            divisor++;
            if(num == divisor)
               encontrado = true;
            else
               encontrado = false;
      }
      return(encontrado);
   }

   public static void main(String[] args) throws IOException{

      BufferedReader in=new BufferedReader(new InputStreamReader(System.in));
```

```
        System.out.println("Introduce un numero: ");
        int num = Integer.valueOf(in.readLine().trim()).intValue();

        if (Primo(num) == true)
           System.out.println("El numero "+num+" es primo");
        else
           System.out.println("EL numero "+num+" no es primo");
        }
}

import java.io.*;
public class Ejercicio22 {

   public static void main(String[] args) throws IOException{
        boolean encontrado = false;
        int suma = 0 , primos = 0;
        BufferedReader in=new BufferedReader(new InputStreamReader(System.in));

        do {
           System.out.println("Introduce un numero: ");
           int num = Integer.valueOf(in.readLine().trim()).intValue();

           Matematica a = new Matematica();
           a.Calcula_Primo(num);

           if(a.Calcula_Primo(num) != encontrado){
              suma = suma + num;
              primos++;
           }
        }while (primos < 5);
        System.out.println("La suma de los primos es: "+suma);
   }
}

import java.io.*;
public class Ejercicio23 {

   public static void main(String[] args) throws IOException{
        BufferedReader in=new BufferedReader(new InputStreamReader(System.in));

        System.out.println("Introduce un numero: ");
        int num = Integer.valueOf(in.readLine().trim()).intValue();

        Matematica a = new Matematica();
        a.Calcula_Factorial(num);

        System.out.println("Su factorial es: "+a.Calcula_Factorial(num));
   }
}


import java.io.*;
public class Ejercicio25 {

   public static void main(String[] args) throws IOException{
        BufferedReader in=new BufferedReader(new InputStreamReader(System.in));

        System.out.println("Introduce un numero: ");
```

```
        int num = Integer.valueOf(in.readLine().trim()).intValue();

        Matematica a = new Matematica();
        System.out.println(a.Calcula_numeroE(num));
    }

}

import java.io.*;
public class Ejercicio26 {

    public static void main(String[] args) throws IOException{
        BufferedReader in=new BufferedReader(new InputStreamReader(System.in));

        System.out.println("Introduce un numero:");
        int n = Integer.valueOf(in.readLine().trim()).intValue();

        System.out.println("Introduce un numero:");
        int i = Integer.valueOf(in.readLine().trim()).intValue();

        Matematica m = new Matematica();
        System.out.println("Resultado "+n+" sobre "+i+" =
"+m.Calcula_Factorial(n)/m.Calcula_Factorial(i)*m.Calcula_Factorial(n-i));
    }
}


import java.io.*;
public class Ejercicio27 {

    public static void main(String[] args) throws IOException{
        int vector[];
        int suma=0 , producto=1;

        BufferedReader in=new BufferedReader(new InputStreamReader(System.in));

        System.out.println("Indica la longitud del vector:");
        int longitud=Integer.valueOf(in.readLine().trim()).intValue();
        vector=new int[longitud];

        for(int i=0;i<vector.length;i++){
            System.out.println("Introduce un numero: ");
            vector[i]=Integer.valueOf(in.readLine().trim()).intValue();
            if ((vector[i] % 2)==0)
                suma = suma+vector[i];
            if ((vector[i] % 5)==0)
                producto = producto * vector[i];
        }
        System.out.println("La suma de los pares es: "+suma);
        System.out.println("El producto de los multiplos de 5 es: "+producto);
    }
}
import java.io.*;
public class Ejercicio28 {

    public static void main(String[] args) throws IOException{
        int vector[] , pares[];
        int mayor=0;
```

```
        BufferedReader in=new BufferedReader(new InputStreamReader(System.in));

        System.out.println("¿cuantos numeros quieres leer?");
        int longitud=Integer.valueOf(in.readLine().trim()).intValue();
        vector=new int[longitud];

        for (int i=0 ; i<vector.length ; i++){
            System.out.println("Introduce un numero: ");
            vector[i]=Integer.valueOf(in.readLine().trim()).intValue();
            if ((vector[i] % 2) == 0)
                System.out.println("El numero es PAR");
            if (vector[i] > mayor)
                mayor = vector[i];
        }
        System.out.println("EL mayor de los PARES leidos es: "+mayor);
    }
}

import java.io.*;
public class Ejercicio29 {
    public static void main(String[] args) throws IOException{
        int vector[];
        int mayor=0 , menor=0;
        BufferedReader in=new BufferedReader(new InputStreamReader(System.in));

        System.out.println("¿cuantos numeros quieres leer?");
        int longitud=Integer.valueOf(in.readLine().trim()).intValue();
        vector=new int[longitud];

        for(int i=0 ; i<vector.length ; i++){
            System.out.println("Introduce un numero:");
            vector[i] = Integer.valueOf(in.readLine().trim()).intValue();
            if ((vector[i] % 5) == 0)
                System.out.println("El numero es multiplo de 5");
                if (vector[i] > mayor)
                    mayor = vector[i];
            if ((vector[i] % 3) == 0)
                System.out.println("El numero es multiplo de 3");
                if (vector[i] < menor)
                    menor = vector[i];
        }

        System.out.println("El multiplo de 5 mas mayor es: "+mayor);
        System.out.println("El multiplo de 3 mas pequeño es: "+menor);
    }
}


import java.io.*;
public class Ejercicio30 {

    public static void main(String[] args)  throws IOException
    {

        int indice,suma;
        BufferedReader in=new BufferedReader(new InputStreamReader(System.in));
        String cadena=in.readLine().trim();
```

```
        //for(int indice=0,suma=0;indice<cadena.length();suma+=(int)cadena.charAt(indice++));
        for(indice=suma=0;indice<cadena.length();indice++)
            suma+=(int)cadena.charAt(indice);
        System.out.print(suma);
        System.in.read();

    }
}

import java.io.*;
public class Ejercicio31 {

    public static void main(String[] args) throws IOException{
        BufferedReader in=new BufferedReader(new InputStreamReader(System.in));

        int vector[] = new int[5];

        for(int i=0 ; i<5 ; i++){
            System.out.println("Introduce un numero: ");
            vector[i] = Integer.valueOf(in.readLine().trim()).intValue();
        }
        for(int i=0 ; i<5 ; i++)
            System.out.println("Posicion: "+i+" Numero: "+vector[i]);
    }
}

import java.io.*;
public class Ejercicio32 {

    public static void main(String[] args) throws IOException{
        BufferedReader in=new BufferedReader(new InputStreamReader(System.in));

        int i , mayor=0;
        int vector[] = new int[10];

        for (i=0 ; i<10 ; i++){
            System.out.println("Introduce un numero: ");
            vector[i] = Integer.valueOf(in.readLine().trim()).intValue();
        }

        for (i=0 ; i<10 ; i++){
            if (vector[i] > mayor)
                mayor = vector[i];
        }
        System.out.println("El numero mayor es: "+mayor);
    }
}




import java.io.*;
public class Ejercicio33 {

    public static void main(String[] args) throws IOException{
```

```
        BufferedReader in=new BufferedReader(new InputStreamReader(System.in));
        int vectorA[] = new int[5];
        int vectorB[] = new int[5];
        int vectorC[] = new int[5];
        int i,j;

        for (i=0 ; i<5 ; i++){
           System.out.println("Introduce un numero en el vector A: ");
           vectorA[i] = Integer.valueOf(in.readLine().trim()).intValue();
        }
        for (j=0 ; j<5 ; j++){
           System.out.println("Introduce un numero en el vector B: ");
           vectorB[j] = Integer.valueOf(in.readLine().trim()).intValue();
        }

        for (i=0 ; i<5 ; i++){
           for (j=0 ; j<5 ; j++)
              vectorC[i] = vectorA[i] + vectorB[j];

           }

        for (i=0 ; i<5 ; i++)
           System.out.println("Posicion: "+i+" Numero: "+vectorC[i]);
    }
}

import java.io.*;
public class Ejercicio34 {

    public static void main(String[] args) throws IOException{
        BufferedReader in=new BufferedReader(new InputStreamReader(System.in));

        int vectorA[] = new int[5];
        int vectorB[] = new int[5];
        int vectorC[] = new int[5];
        int i,j,mayor;

        for (i=0 ; i<5 ; i++){
           System.out.println("Introduce un numero en el vector A: ");
           vectorA[i] = Integer.valueOf(in.readLine().trim()).intValue();
        }
        for (j=0 ; j<5 ; j++){
           System.out.println("Introduce un numero en el vector B: ");
           vectorB[j] = Integer.valueOf(in.readLine().trim()).intValue();
        }

        for (i=0 ; i<5 ; i++){
           for (j=0 ; j<5 ; j++)
              vectorC[i] = vectorA[i] + vectorB[j];
        }
        mayor = vectorC[1];
        for (i=0 ; i<5 ; i++){
           if(vectorC[i] > mayor)
              mayor = vectorC[i];
        }
        System.out.println("El numero mayor del vector C es: "+mayor);
    }
```

```
}

import java.io.*;
public class Ejercicio35 {

   public static void main(String[] args) throws IOException{
      int i,j,aux;
      BufferedReader in=new BufferedReader(new InputStreamReader(System.in));

      System.out.println("¿Cuantos numeros vas a meter en el vector:?");
      int longitud = Integer.valueOf(in.readLine().trim()).intValue();

      int vector[] = new int[longitud];

      for (i=0 ; i<vector.length ; i++){
         System.out.println("Introduce un numero en el vector: ");
         vector[i] = Integer.valueOf(in.readLine().trim()).intValue();
      }

      for (i=0 ; i<vector.length ; i++){
         for (j=0 ; j<vector.length ; j++)
            if (vector[i] < vector[j]){
            aux = vector[i];
            vector[i] = vector[j];
            vector[j] = aux;
            }
      }

      System.out.println("El vector ordenado es: ");
      for (i=0 ; i<vector.length ; i++)
         System.out.println(vector[i]);
   }
}

import java.io.*;
public class Ejercicio36 {

   public static void main(String[] args) throws IOException{
      boolean encontrado = false;
      int i , pos=0;
      BufferedReader in=new BufferedReader(new InputStreamReader(System.in));

      System.out.println("¿Cuantos numeros vas a introducir en el vector:?");
      int longitud = Integer.valueOf(in.readLine().trim()).intValue();

      int vector[] = new int[longitud];

      for(i=0 ; i<vector.length ; i++){
         System.out.println("Introduce un numero en el vector: ");
         vector[i] = Integer.valueOf(in.readLine().trim()).intValue();
      }

      System.out.println("¿Que numero deseas buscar?:");
      int num = Integer.valueOf(in.readLine().trim()).intValue();

      for (i=0 ; i<vector.length ; i++){
         if (vector[i] == num){
            encontrado = true;
```

```
            pos = vector[i];
         }
      }
      if (encontrado == true)
         System.out.println("El numero existe y su posicion es:"+pos);
      else
         System.out.println("El numero no existe");
   }
}

import java.io.*;
public class Ejercicio37 {

   public static void main(String[] args) throws IOException{
      BufferedReader in=new BufferedReader(new InputStreamReader(System.in));
      int i;
      int vector[] = new int[10];

      for (i = 0 ; i<10 ; i++){
         System.out.println("Introduce un numero: ");
         int num = Integer.valueOf(in.readLine().trim()).intValue();
         Matematica f = new Matematica();
         vector[i] = f.Calcula_Factorial(num);
         System.out.println("Factorial de "+num+" : "+vector[i]);
      }
   }
}

import java.io.*;
public class Ejercicio38 {

   public static void main(String[] args) throws IOException{
      int i,j,aux;
      BufferedReader in=new BufferedReader(new InputStreamReader(System.in));

      int vector[] = new int[5];

      try{
         for(i=0 ; i<5 ; i++){
            System.out.println("Introduce un numero en el vector:");
            vector[i] = Integer.valueOf(in.readLine().trim()).intValue();
            if (vector[i] < 0)
               throw new Ejercicio38Exception("Error, numero negativo");
            for (j=0 ; j<vector.length ; j++){
               if (vector[i] < vector[j]){
               aux = vector[i];
               vector[i] = vector[j];
               vector[j] = aux;
               }
            }
         }
      System.out.println("El vector ordenado es: ");
      for (i=0 ; i<vector.length ; i++)
         System.out.println(vector[i]);
      }
      catch(Ejercicio38Exception e){
         System.out.println(e.getMessage());
      }
```

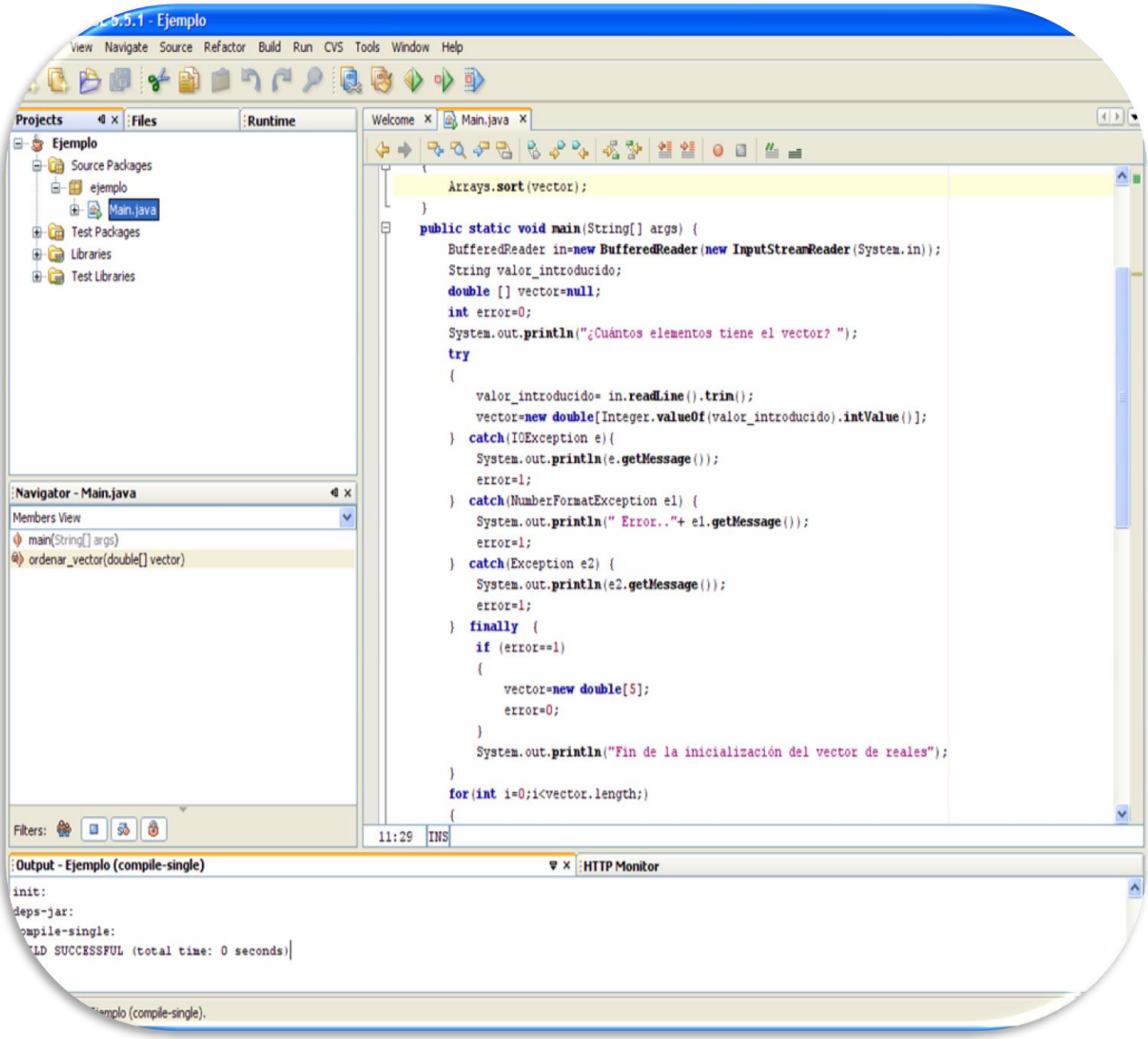
```
        catch(Exception ex){
            System.out.println("Error, se llama a la clase general Exception");
        }
    }
}

public class Ejercicio38Exception extends Exception{

    /** Creates a new instance of Ejercicio38Exception */
    public Ejercicio38Exception(String mensaje) {
        super(mensaje);
    }
}

import java.io.*;
public class Ejercicio39 {

    public static void main(String[] args) throws IOException{
        BufferedReader in=new BufferedReader(new InputStreamReader(System.in));

        int vector1[][] = new int[5][5];
        int vector2[][] = new int[5][5];
        int vectorS[][] = new int[5][5];
        int fil,col;

        for(fil=0 ; fil<5 ; fil++)
            for(col=0 ; col<5 ; col++){
                System.out.println("Vector 1. Fila:"+fil+" Columna: "+col);
                vector1[fil][col] = Integer.valueOf(in.readLine().trim()).intValue();
            }
        for(fil=0 ; fil<5 ; fil++)
            for(col=0 ; col<5 ; col++){
                System.out.println("Vector 2. Fila:"+fil+" Columna: "+col);
                vector2[fil][col] = Integer.valueOf(in.readLine().trim()).intValue();
            }
        for(fil=0 ; fil<5 ; fil++)
            for(col=0 ; col<5 ; col++){
                vectorS[fil][col] = vector1[fil][col] + vector2[fil][col];
                System.out.println("Suma: "+vectorS[fil][col]);
            }
    }
}

import java.io.*;
public class Ejercicio40 {

    public static void main(String[] args) throws IOException{
        BufferedReader in=new BufferedReader(new InputStreamReader(System.in));
        int vector[][] = new int[3][3];
        int mayor[][] = new int[1][1];
        int fil , col;

        for(fil=0 ; fil<3 ; fil++)
            for(col=0 ; col<3; col++){
                System.out.println("Vector1. Fila: "+fil+" Columna: "+col);
                vector[fil][col] = Integer.valueOf(in.readLine().trim()).intValue();
            }
        for(fil=0 ; fil<3 ; fil++)
```

```
            for(col=0 ; col<3; col++){
                if (vector[fil][col] > mayor[fil][col])
                    vector[fil][col] = mayor[fil][col];
                    System.out.println(mayor[fil][col]);
                }
    }

 }


import java.io.*;
public class Ejercicio41 {

    public static void main(String[] args) throws IOException{

        BufferedReader in=new BufferedReader(new InputStreamReader(System.in));

        int vector[][] = new int[3][3];
        int mayor[][] = new int[1][1];
        int menor[][] = new int[1][1];
        int fil , col;

        for(fil=0 ; fil<3 ; fil++)
            for(col=0 ; col<3; col++){
                System.out.println("Vector1. Fila: "+fil+" Columna: "+col);
                vector[fil][col] = Integer.valueOf(in.readLine().trim()).intValue();
            }
        for(fil=0 ; fil<3 ; fil++)
            for(col=0 ; col<3; col++){
                if (vector[fil][col] > mayor[fil][col])
                    vector[fil][col] = mayor[fil][col];
                    System.out.println(mayor[fil][col]);
                }
    }

 }

import java.io.*;
import java.text.*;
public class Ejercicio42 {

    public static void main(String[] args) throws IOException {

        BufferedReader in=new BufferedReader(new InputStreamReader(System.in));

        System.out.println("Introduce un numero en decimal:");
        int num = Integer.valueOf(in.readLine().trim()).intValue();

        char opcion;
        System.out.println("Introduce (B)inario/(H)exadecimal/(O)ctal:");
        opcion = (char)System.in.read();
        System.in.read();

        if (opcion == 'B')
            System.out.println(Integer.toBinaryString(num));
        else
             if (opcion == 'H')
                 System.out.println(Integer.toHexString(num));
```

```
            else
              if (opcion == 'O')
                  System.out.println(Integer.toOctalString(num));
    }
}


import java.io.*;
public class Ejercicio43 {

    public static void main(String[] args) throws IOException{
        BufferedReader in=new BufferedReader(new InputStreamReader(System.in));

        int vector[] = new int[5];
        int i , suma=0 , mayor=0;

        for(i=0 ; i<5 ; i++){
            System.out.println("Introduce un numero en el vector:" );
            vector[i] = Integer.valueOf(in.readLine().trim()).intValue();
        }

        Matematica p = new Matematica();

        for(i=0 ; i<5 ; i++){
            if (((vector[i]) % 2) == 0);
                suma = suma + vector[i];
        }
        for(i=0 ; i<5 ; i++){
            if (vector[i] > mayor)
                mayor = vector[i];
        }
        System.out.println("Los numeros pares del vector suman: "+suma);
        System.out.println("El numero mayor de los impares es el: "+mayor);
    }
}

import java.io.*;
public class Ejercicio44 {

    public Ejercicio44() {}

    public static void main(String[] args) throws IOException{
        BufferedReader in=new BufferedReader(new InputStreamReader(System.in));

        int vector[][] = new int[5][5];
        int fil=4,col=5,cont=0,aux,i,j;

        for(i=0 ; i<fil ; i++)
            for(j=0 ; j<col ; j++){
                System.out.println("Fila:"+i+" Columna: "+j);
                vector[i][j] = Integer.valueOf(in.readLine().trim()).intValue();
            }
        int a=0,b=0;
        int aux_a,aux_b;

        aux_a = a;
        aux_b = b;
```

```
        while (cont < (fil*col)){
            for(i=aux_a ; i<fil ; i++){
                for(j=aux_b ; j<col ; j++){
                    if (vector[i][j] < vector[a][b]){ //comparamos y si es menor intercambiamos valores
                        aux = vector[i][j];
                        vector[i][j] = vector[a][b];
                        vector[a][b] = aux;
                    }
                }
                aux_b = 0;
            }
            b++;
            if (b >= fil){
                b = 0;
                a++;
                aux_a++;
            }
            cont++;
            aux_b = b;
        }

        for(i=0 ; i<fil ; i++) //se muestra la matriz ordenada
            for(j=0 ; j<col ; j++)
                System.out.println(vector[i][j]);
    }
}

import java.io.*;
public class Ejercicio46 {

    public static void main(String[] args) throws IOException{
        int i;
        BufferedReader in=new BufferedReader(new InputStreamReader(System.in));

        System.out.println("¿Cuantos numeros vas a introducir?:");
        int longitud = Integer.valueOf(in.readLine().trim()).intValue();

        int vector[] = new int[longitud];
        Matematica p = new Matematica();

        for(i=0 ; i<vector.length ; i++){
            System.out.println("Introduce un numero en el vector: ");
            vector[i] = Integer.valueOf(in.readLine().trim()).intValue();
            p.Calcula_Primo(vector[i]);
        }
    }
}
```

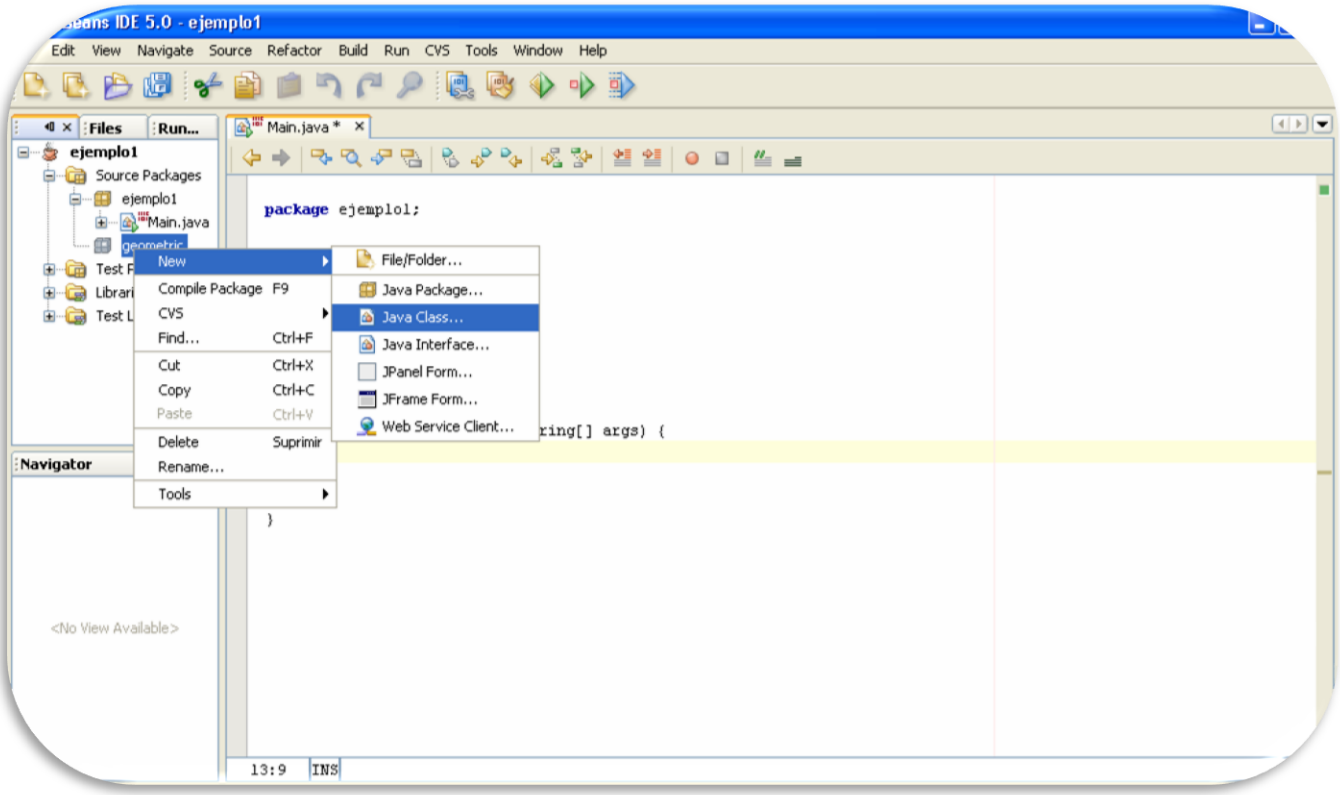
PASO 1.-

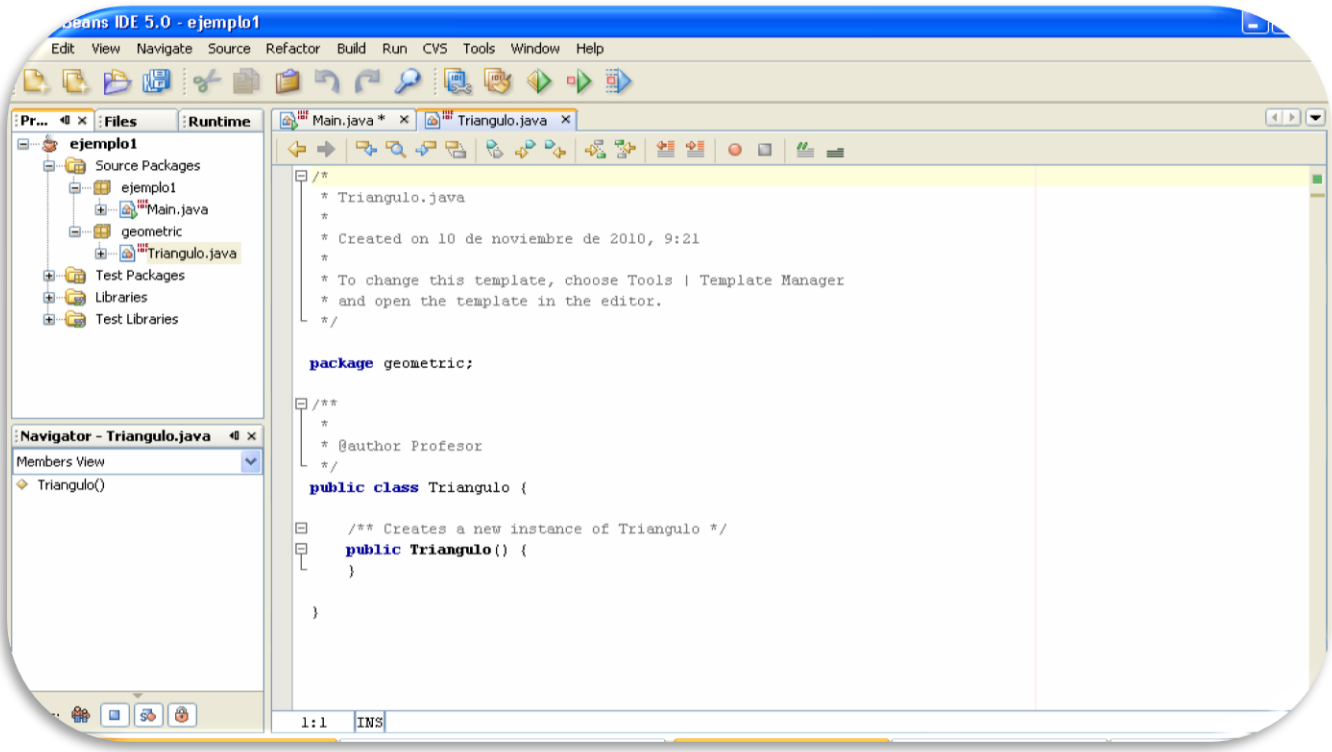

Ejemplo
View Navigate Source Refactor Build Run CVS Tools Window Help
Projects
Files
Runtime
Ejemplo
Source Packages
ejemplo
Main.java
Test Packages
Libraries
Test Libraries
Welcome
Main.java
Arrays.sort(vector);
}
public static void main(String[] args) {
BufferedReader in=new BufferedReader(new InputStreamReader(System.in));
String valor_introducido;
double [] vector=null;
int error=0;
System.out.println("¿Cuántos elementos tiene el vector? ");
try
{
valor_introducido= in.readLine().trim();
vector=new double[Integer.valueOf(valor_introducido).intValue()];
} catch(IOException e){
System.out.println(e.getMessage());
error=1;
} catch(NumberFormatException e1) {
System.out.println(" Error.."+ e1.getMessage());
error=1;
} catch(Exception e2) {
System.out.println(e2.getMessage());
error=1;
} finally {
if (error==1)
{
vector=new double[5];
error=0;
}
System.out.println("Fin de la inicialización del vector de reales");
}
for(int i=0;i<vector.length;)
{
Navigator - Main.java
Members View
main(String[] args)
ordenar_vector(double[] vector)
Filters:
11:29
INS
Output - Ejemplo (compile-single)
HTTP Monitor
init:
deps-jar:
compile-single:
LD SUCCESSFUL (total time: 0 seconds)


Definición de una clase
Campos
Constructor(es)
Método(s)
Instrucciones

```
package ejemplo;

import java.io.*;
import java.lang.*;
import java.util.*;

public class Main {
   private static void ordenar_vector(double [] vector)
   {
      Arrays.sort(vector);
   }
   public static void main(String[] args) {
      BufferedReader in=new BufferedReader(new InputStreamReader(System.in));
      String valor_introducido;
      double [] vector=null;
      int error=0;
      System.out.println("¿Cuántos elementos tiene el vector? ");
      try
      {
         valor_introducido= in.readLine().trim();
         vector=new double[Integer.valueOf(valor_introducido).intValue()];
      }  catch(IOException e){
         System.out.println(e.getMessage());
         error=1;
      }  catch(NumberFormatException e1) {
         System.out.println(" Error.."+ e1.getMessage());
         error=1;
      }  catch(Exception e2) {
         System.out.println(e2.getMessage());
         error=1;
      }  finally  {
         if (error==1)
         {
            vector=new double[5];
            error=0;
         }
         System.out.println("Fin de la inicialización del vector de reales");
      }
      for(int i=0;i<vector.length;)
      {
            System.out.println("Introduce vector ["+i+"]: ");
            try
            {
               valor_introducido= in.readLine().trim();
               vector[i]=Double.valueOf(valor_introducido).doubleValue();
            }  catch(IOException e){
               System.out.println(e.getMessage());
               error=1;
            }  catch(NumberFormatException e1) {
               System.out.println(" Error.."+ e1.getMessage());
               error=1;
            }  catch(Exception e2) {
               System.out.println(e2.getMessage());
               error=1;
            }  finally  {
               if (error==1)
               {
                  vector[i]=0.0;
                  error=0;
               }
               i++;
            }
      }
      ordenar_vector(vector);
      System.out.println("El resultado de la ordenación es: ");
      for(int i=0;i<vector.length;i++)
      {
            System.out.println("Introduce vector ["+i+"]: "+vector[i]);
      }
      System.exit(0);
   }
}
```

PASO 2.-


beans IDE 5.0 - ejemplo1
Edit View Navigate Source Refactor Build Run CVS Tools Window Help
Files
Run...
ejemplo1
Source Packages
ejemplo1
Main.java
geometric
Test P
Librari
Test L
Main.java *
package ejemplo1;
New
Compile Package F9
CVS
Find... Ctrl+F
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Delete Suprimir
Rename...
Tools
File/Folder...
Java Package...
Java Class...
Java Interface...
JPanel Form...
JFrame Form...
Web Service Client...
ring[] args) {
Navigator
<No View Available>
13:9
INS


beans IDE 5.0 - ejemplo1
Edit View Navigate Source Refactor Build Run CVS Tools Window Help
Pr...
Files
Runtime
ejemplo1
Source Packages
ejemplo1
Main.java
geometric
Triangulo.java
Test Packages
Libraries
Test Libraries
Main.java *
Triangulo.java
/*
* Triangulo.java
*
* Created on 10 de noviembre de 2010, 9:21
*
* To change this template, choose Tools | Template Manager
* and open the template in the editor.
*/
package geometric;
/**
*
* @author Profesor
*/
public class Triangulo {
/** Creates a new instance of Triangulo */
public Triangulo() {
}
}
Navigator - Triangulo.java
Members View
Triangulo()
1:1
INS

```
package geometric;

import java.lang.*;
import java.util.*;

// http://moisesrbb.tripod.com/java3.htm

public class Triangulo {
/*********************************************************************
 [public] [final | abstract] class Clase [extends ClaseMadre] [implements Interfase1 [, Interfase2 ]…]
 o bien, para interfaces:
 [public] interface Interfase [extends InterfaseMadre1 [, InterfaseMadre2 ]…]
*********************************************************************/

   // Definición de los atributos (private, public, protected)

   /*****************************************************************
    [private|protected|public] [static] [final] [transient] [volatile] Tipo NombreVariable [= Valor];
   *****************************************************************/
   public double altura;
   private double base;

   public Triangulo() // Constructor por defecto
   {
      this.altura=0.0;
      this.base=0.0;
   }

   public Triangulo(double a) // Constructor sobrecargado
   {
      this.altura=a;
      this.base=0.0;
   }

   public Triangulo(double a, double b) // Constructor sobrecargado
   {
      this.altura=a;
      this.base=b;
   }

   /*****************************************************************************************
    [private|protected|public] [static] [abstract] [final] [native] [synchronized] TipoDevuelto NombreMétodo ( [tipo1
nombre1[, tipo2 nombre2 ]…] ) [throws excepción1 [,excepción2]… ]
    *****************************************************************************************/
   public double get_base()
   {
      return this.base;
   }

   public double area()
   {
      return (this.base*this.altura)/2;
   }

   public boolean comparaTO (Triangulo t)
   {
      if ((this.altura==t.altura) && (this.base==t.get_base()))
         return true;
      return false;
    }
}
```

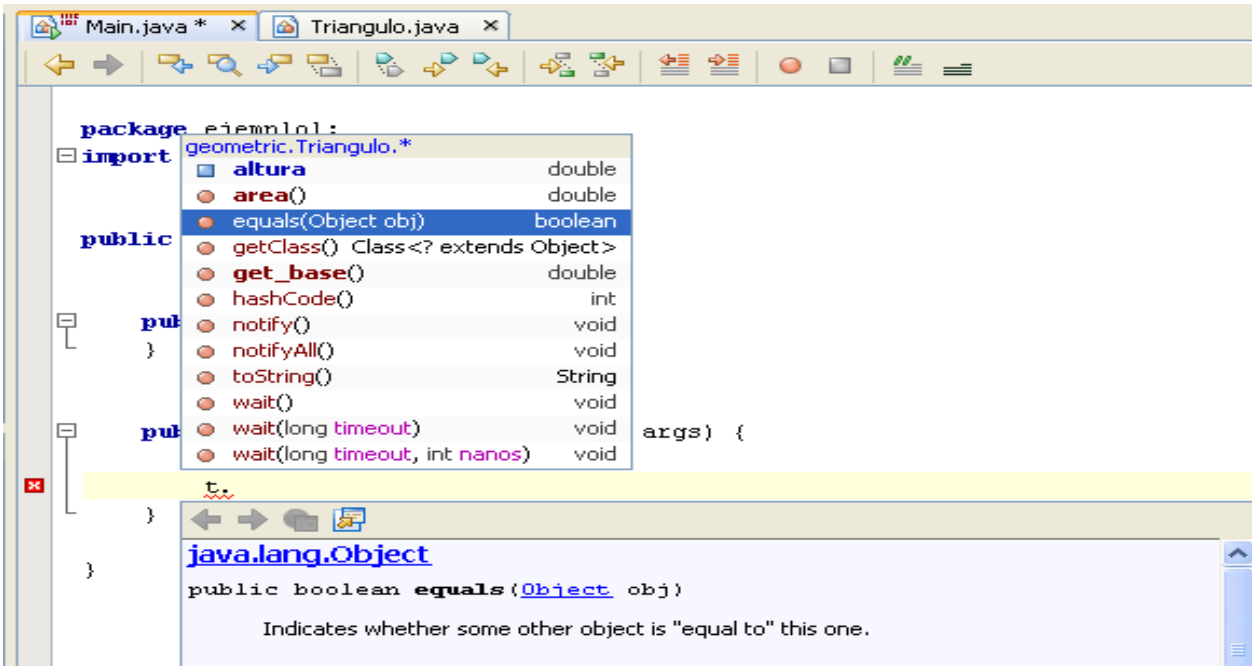
PASO 3.-

La clase Object

La clase *Object* es la superclase de todas las clases da Java. Todas las clases derivan, directa o indirectamente de ella. Si al definir una nueva clase, no aparece la cláusula *extends*, Java considera que dicha clase desciende directamente de *Object*.

La clase *Object* aporta una serie de funciones básicas comunes a todas las clases:

- *public boolean equals(Object obj)*: Se utiliza para comparar, en valor, dos objetos. Devuelve *true* si el objeto que recibe por parámetro es igual, en valor, que el objeto desde el que se llama al método. Si se desean comparar dos referencias a objeto se pueden utilizar los operadores de comparación == y !=.
- *public int hashCode()*: Devuelve un código hash para ese objeto, para poder almacenarlo en una *Hashtable*.
- *protected Object clone() throws CloneNotSupportedException:* Devuelve una copia de ese objeto.
- *public final Class getClass()*: Devuelve el objeto concreto, de tipo *Class*, que representa la clase de ese objeto.
- *protected void finalize() throws Trowable*: Realiza acciones durante la recogida de basura.
- …. etc.



PASO 4.-

Herencia

Pero además de esta técnica de composición/creación de clases es posible pensar en casos en los que una clase es una extensión de otra. Es decir una clase es como otra y además tiene algún tipo de característica propia que la distingue. Por ejemplo podríamos pensar en la clase Empleado y definirla como:

```
package departamento;

public class Empleado {
    public String nombre;
    public int numEmpleado, sueldo;

    static private int contador = 0;

    public Empleado() {}

    public Empleado(String nombre, int sueldo)
    {
        this.nombre = nombre;
        this.sueldo = sueldo;
        this.numEmpleado = ++this.contador;
    }

    public void aumentarSueldo(int porcentaje) {
        this.sueldo += (this.sueldo * porcentaje / 100);
    }

    public String toString() {
        return "Num. empleado " + this.numEmpleado + " Nombre: " +
this.nombre +  " Sueldo: " + this.sueldo;
    }
```

En el ejemplo el Empleado se caracteriza por un nombre (String) y por un número de empleado y sueldo (enteros). La clase define un constructor que asigna los valores de nombre y sueldo y calcula el número de empleado a partir de un contador (variable estática que siempre irá aumentando), y dos métodos, uno para calcular el nuevo sueldo cuando se produce un aumento de sueldo (método aumentarSueldo) y un segundo que devuelve una representación de los datos del empleado en un String.(método toString).

Con esta representación podemos pensar en otra clase que reuna todas las características de Empleado y añada alguna propia. Por ejemplo, la clase Ejecutivo. A los objetos de esta clase se les podría aplicar todos los datos y métodos de la clase Empleado y añadir algunos, como por ejemplo el hecho de que un Ejecutivo tiene un presupuesto.

Así diriamos que la clase Ejecutivo extiende o hereda la clase Empleado. Esto en Java se hace con la clausula **extends** que se incorpora en la definición de la clase, de la siguiente forma:

```
package departamento;

public class Ejecutivo extends Empleado
{
    public int presupuesto;

    public Ejecutivo()
    {
    super();
    }

    public Ejecutivo(String nombre, int sueldo,int p)
    {
       super(nombre,sueldo);
       this.presupuesto=p;
    }

    public String toString() {
        String s = super.toString();
        s = s + " Presupuesto: " + this.presupuesto;
        return s;
    }

    public void asignarPresupuesto(int p) {
        presupuesto = p;
    }
}
```

Con esta definición un Ejecutivo es un Empleado que además tiene algún rasgo distintivo propio. El cuerpo de la clase Ejecutivo incorpora sólo los miembros que son específicos de esta clase, pero implícitamente tiene todo lo que tiene la clase Empleado.

A Empleado se le llama clase base o superclase y a Ejecutivo clase derivada o subclase.

Redefinición de métodos. El uso de `super`.

Además se podría pensar en redefinir algunos métodos de la clase base pero haciendo que métodos con el mismo nombre y características se comporten de forma distinta. Por ejemplo podríamos pensar en rediseñar el método toString de la clase Empleado añadiendo las características propias de la clase Ejecutivo.

```
public String toString() {
    String s = super.toString();
    s = s + " Presupuesto: " + this.presupuesto;
    return s;
}
```

De esta forma cuando se invoque `jefe.toString()` se usará el método toString de la clase Ejecutivo en lugar del existente en la clase Empleado.

Observese en el ejemplo el uso de **`super`**, que representa referencia interna implícita a la clase base (superclase). Mediante **`super.`**`toString()` se invoca el método toString de la clase Empleado

Inicialización de clases derivadas

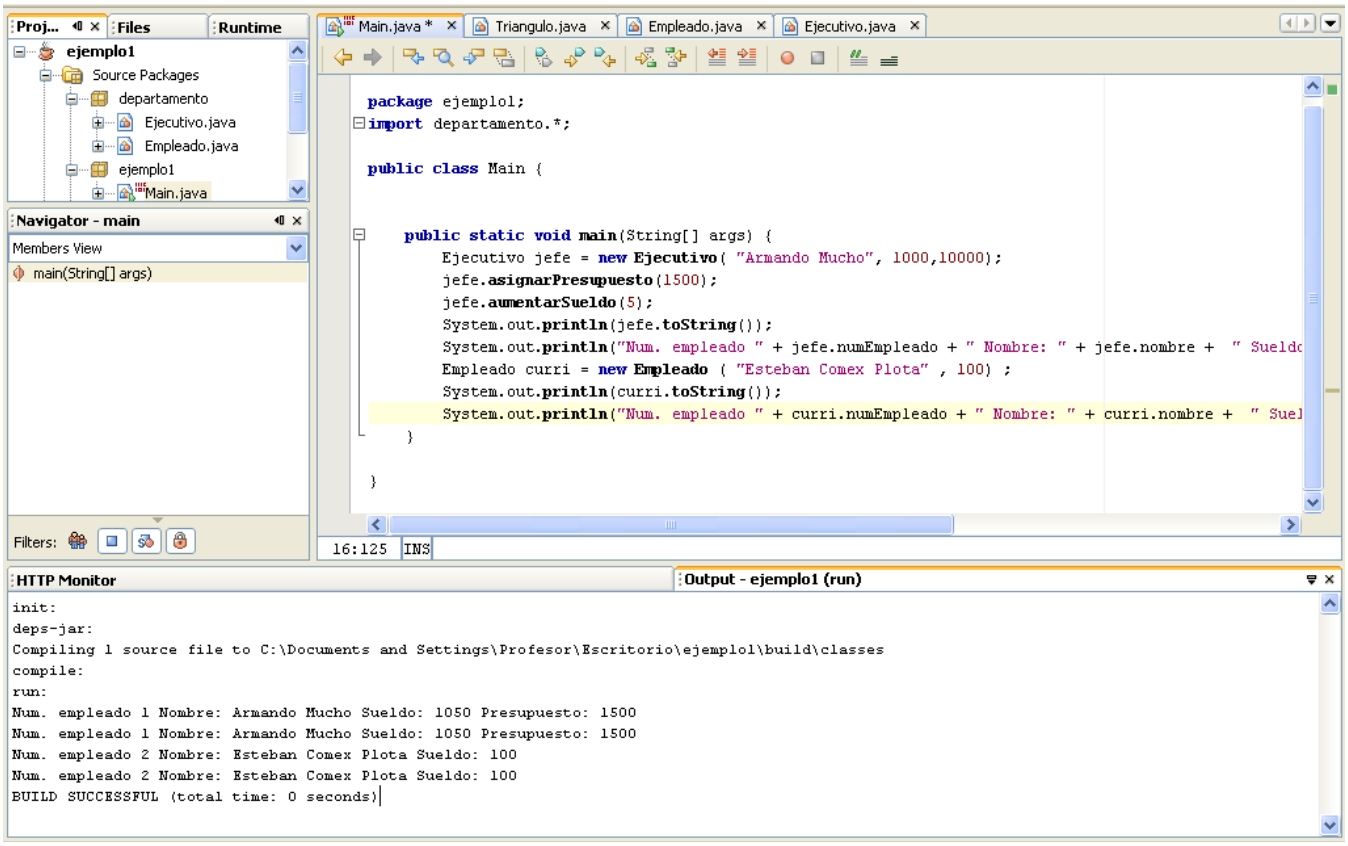
Cuando se crea un objeto de una clase derivada se crea implícitamente un objeto de la clase base que se inicializa con su constructor correspondiente. Si en la creación del objeto se usa el constructor no-args, entonces se produce una llamada implícita al constructor no-args para la clase base. Pero si se usan otros constructores es necesario invocarlos explícitamente.

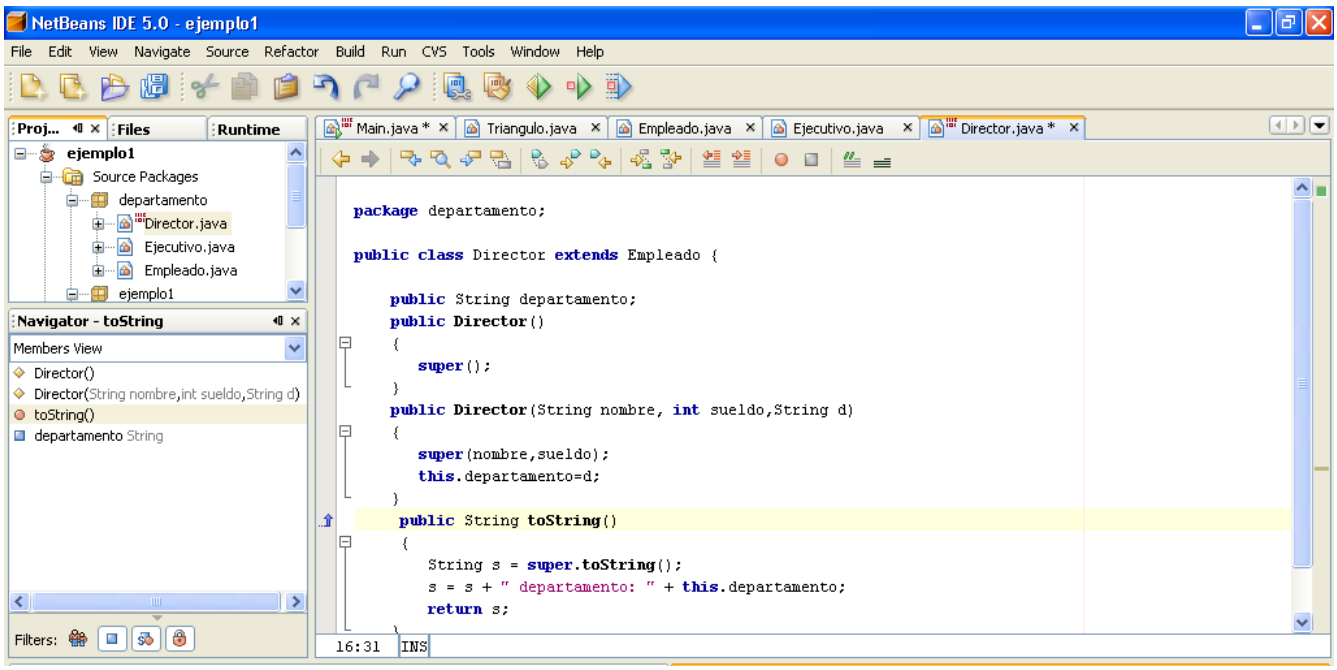
En nuestro ejemplo dado que la clase método define un constructor, necesitaremos también un constructor para la clase Ejecutivo, que podemos completar así:

```
public Ejecutivo(String nombre, int sueldo,int p)
{
   super(nombre,sueldo);
   this.presupuesto=p;
}
```

Observese que el constructor de Ejecutivo invoca directamente al constructor de Empleado mediante `super(`*`argumentos`*`)`. En caso de resultar necesaria la invocación al constructor de la superclase debe ser la primera sentencia del constructor de la subclase.

```
package ejemplo1;
import departamento.*;

public class Main {


    public static void main(String[] args) {
        Ejecutivo jefe = new Ejecutivo( "Armando Mucho", 1000,10000);
        jefe.asignarPresupuesto(1500);
        jefe.aumentarSueldo(5);
        System.out.println(jefe.toString());
        System.out.println("Num. empleado " + jefe.numEmpleado + " Nombre: " + jefe.nombre +  "
Sueldo: " + jefe.sueldo + " Presupuesto: " + jefe.presupuesto);
        Empleado curri = new Empleado ( "Esteban Comex Plota" , 100) ;
        System.out.println(curri.toString());
        System.out.println("Num. empleado " + curri.numEmpleado + " Nombre: " + curri.nombre +  "
Sueldo: " + curri.sueldo);

    }
}
```






```
package departamento;

public class Director extends Empleado {

    public String departamento;
    public Director()
    {
      super();
    }
    public Director(String nombre, int sueldo,String d)
    {
      super(nombre,sueldo);
      this.departamento=d;
    }
     public String toString()
     {
       String s = super.toString();
       s = s + " departamento: " + this.departamento;
       return s;
    }
}
```

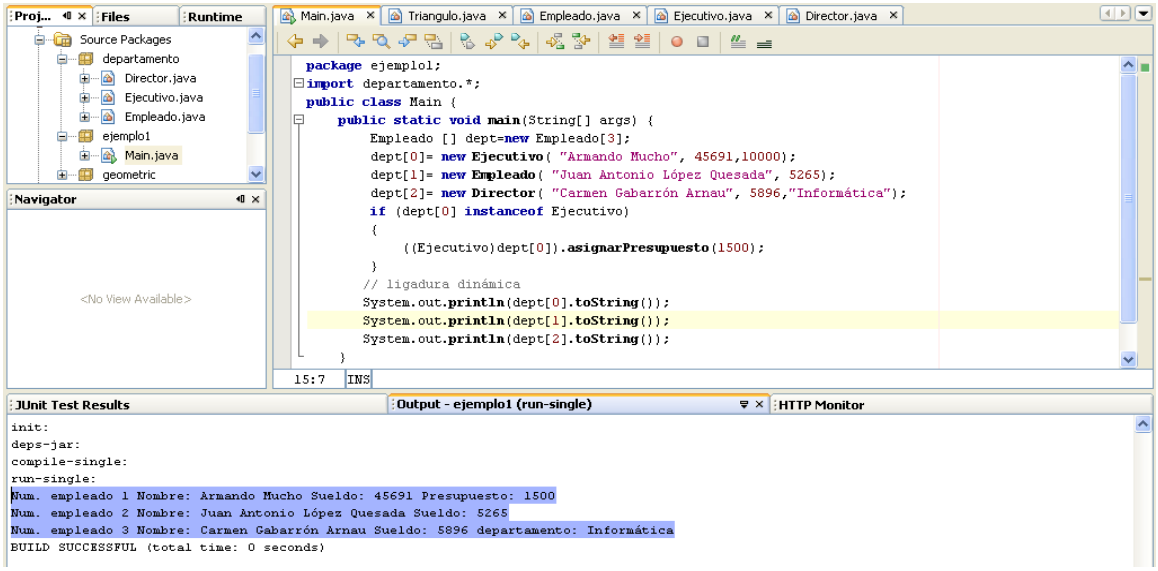
PASO 5.-

Enlace o ligadura dinámica

```
package ejemplo1;
import departamento.*;
public class Main {
   public static void main(String[] args) {
      Empleado [] dept=new Empleado[3];
      dept[0]= new Ejecutivo( "Armando Mucho", 45691,10000);
      dept[1]= new Empleado( "Juan Antonio López Quesada", 5265);
      dept[2]= new Director( "Carmen Gabarrón Arnau", 5896,"Informática");
      if (dept[0] instanceof Ejecutivo)
      {
         ((Ejecutivo)dept[0]).asignarPresupuesto(1500);
      }
     // ligadura dinámica
     System.out.println(dept[0].toString());
     System.out.println(dept[1].toString());
     System.out.println(dept[2].toString());
   }
}
```

La llamada al tiempo de ejecución, cuando se conoce realmente los objetos conectados a r, y cuál es la versión de f() apropiada. Este enfoque de resolución de llamadas se denomina ligadura dinámica y es mucho más lenta y compleja que la estática

- *Hay tres enfoques posibles a la hora de escoger entre ligadura estática o dinámica:*
- *Establecer la ligadura estática por defecto. El programador puede activar la ligadura dinámica para una función concreta cuando lo ve necesario (C++)*
- *Utilizar un compilador inteligente que decide la ligadura estática o dinámica en función del empleo que se hace de cada función (Eiffel)*
- *Establecer la ligadura dinámica para todas las funciones y evitar problemas a costa de eficiencia en la ejecución (Smalltalk, Java)*



PASO 6.-

Clases: abstract y extends

Concepto

Hay ocasiones, cuando se desarrolla una jerarquía de clases en que algún comportamiento está presente en todas ellas pero se materializa de forma distinta para cada una. Por ejemplo, pensemos en una estructura de clases para manipular figuras geométricas. Podríamos pensar en tener una clase genérica, que podría llamarse FiguraGeometrica y una serie de clases que extienden a la anterior que podrían ser Circulo, Poligono, etc. Podría haber un método dibujar dado que sobre todas las figuras puede llevarse a cabo esta acción, pero las operaciones concretas para llevarla a cabo dependen del tipo de figura en concreto (de su clase). Por otra parte la acción dibujar no tiene sentido para la clase genérica FiguraGeometrica, porque esta clase representa una abstracción del conjunto de figuras posibles.

Para resolver esta problemática Java proporciona las clases y métodos abstractos. Un método abstracto es un método declarado en una clase para el cual esa clase no proporciona la implementación (el código). Una clase abstracta es una clase que tiene al menos un método abstracto. Una clase que extiende a una clase abstracta debe implementar los métodos abstractos (escribir el código) o bien volverlos a declarar como abstractos, con lo que ella misma se convierte también en clase abstracta.

Declaración e implementación de métodos abstractos

Siguiendo con el ejemplo del apartado anterior, se puede escribir:

```
abstract class FiguraGeometrica {
    . . .
    abstract void dibujar();
    . . .
}

class Circulo extends FiguraGeometrica {
    . . .
    void dibujar() {
        // codigo para dibujar Circulo
        . . .
    }
}
```

La clase abstracta se declara simplemente con el modificador **`abstract`** en su declaración. Los métodos abstractos se declaran también con el mismo modificador, declarando el método pero sin implementarlo (sin el bloque de código encerrado entre {}). La clase derivada se declara e implementa de forma normal, como cualquier otra. Sin embargo si no declara e implementa los métodos abstractos de la clase base (en el ejemplo el método `dibujar`) el compilador genera un error indicando que no se

han implementado todos los métodos abstractos y que, o bien, se implementan, o bien se declara la clase abstracta.

Por ejemplo, la clase *Number* es una clase abstracta que representa cualquier tipo de números (y sus métodos no están implementados: son abstractos); las clases descendientes de ésta, como *Integer* o *Float*, sí implementan los métodos de la madre *Number*, y se pueden instanciar.

Ejemplos:

Escribir un programa que genere un array que pueda almacenar objetos de las clases `Integer`, `Float`, `Double` *y* `Byte`. *Pista:* `Number[]x = new Number[];`

Escribir un programa que genere un array que pueda almacenar objetos de las clases `Integer`, `Float`, `Double` *y* `Byte`. *Pista:* `Object[]x = new Object[];`

PASO 7.-

Concepto de Interface

El concepto de Interface lleva un paso más adelante la idea de las clases abstractas. En Java una interface es una clase abstracta pura, es dcir una clase donde todos los métodos son abstractos (no se implementa ninguno). Permite al diseñador de clases establecer la forma de una clase (nombres de métodos, listas de argumentos y tipos de retorno, pero no bloques de código). Una interface puede también contener datos miembro, pero estos son siempre static y final. Una interface sirve para establecer un 'protocolo' entre clases.

Para crear una interface, se utiliza la palabra clave interface en lugar de class. La interface puede definirse public o sin modificador de acceso, y tiene el mismo significado que para las clases. Todos los métodos que declara una interface son siempre public.

Para indicar que una clase implementa los métodos de una interface se utiliza la palabra clave implements. El compilador se encargará de verificar que la clase efectivamente declare e implemente todos los métodos de la interface. Una clase puede implementar más de una interface.

Declaración y uso

Una interface se declara:

```
interface nombre_interface {
    tipo_retorno nombre_metodo ( lista_argumentos ) ;
    . . .
}
```

Por ejemplo:

```
interface InstrumentoMusical {
    void tocar();
    void afinar();
    String tipoInstrumento();
}
```

Y una clase que implementa la interface:

```
class InstrumentoViento extends Object implements InstrumentoMusical
{
    void tocar() { . . . };
    void afinar() { . . .};
    String tipoInstrumento() {}
}

class Guitarra extends InstrumentoViento {
    String tipoInstrumento() {
        return "Guitarra";
    }
}
```

La clase `InstrumentoViento` implementa la interface, declarando los métodos y escribiendo el código correspondiente. Una clase derivada puede también redefinir si es necesario alguno de los métodos de la interface.

Referencias a Interfaces

Es posible crear referencias a interfaces, pero las interfaces no pueden ser instanciadas. Una referencia a una interface puede ser asignada a cualquier objeto que implemente la interface. Por ejemplo:

```
InstrumentoMusical instrumento = new Guitarra();
instrumento.play();
System.out.prinln(instrumento.tipoInstrumento());

InstrumentoMusical i2 = new InstrumentoMusical(); //error.No se puede
instanciar
```

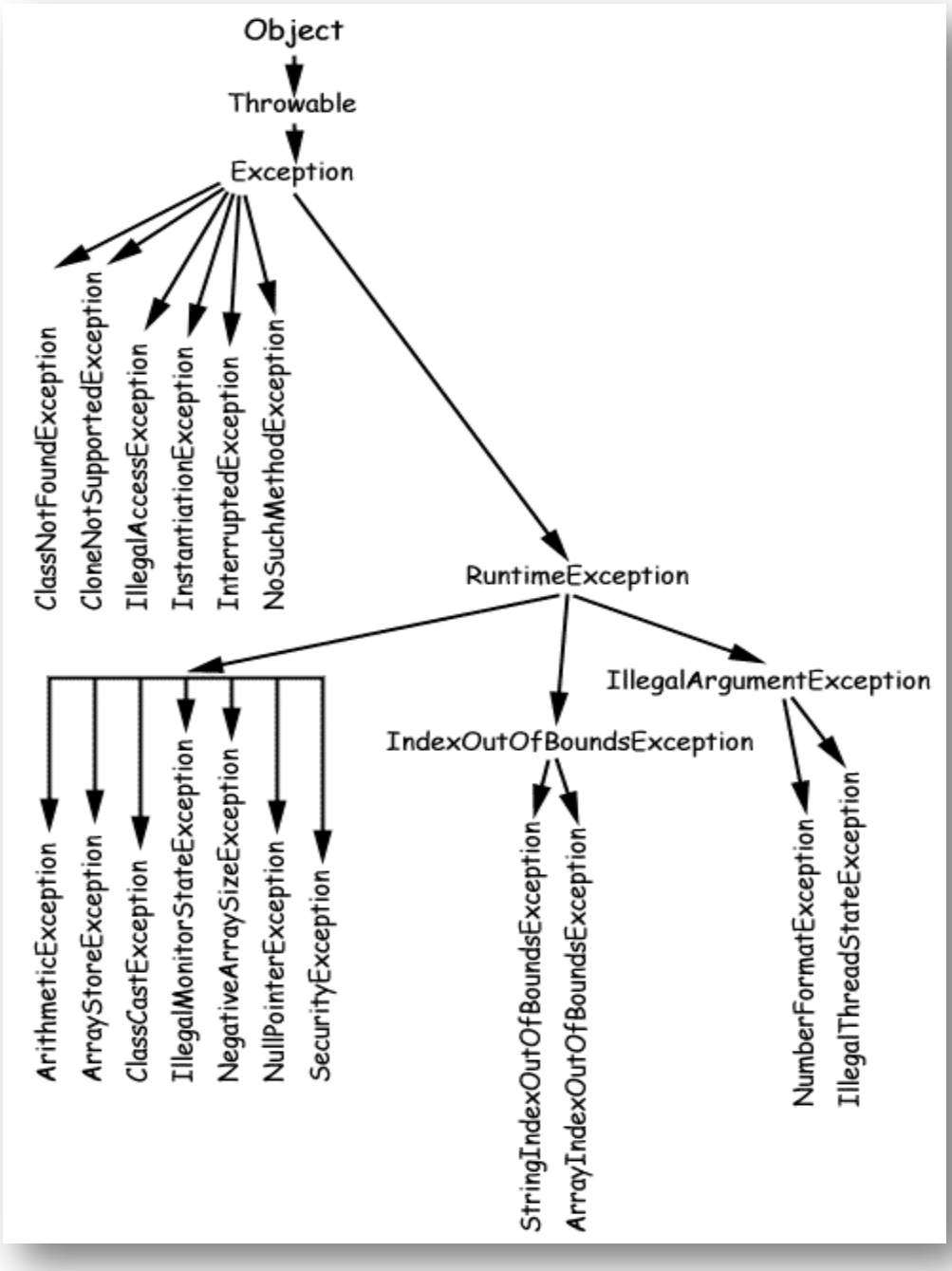
Extensión de interfaces

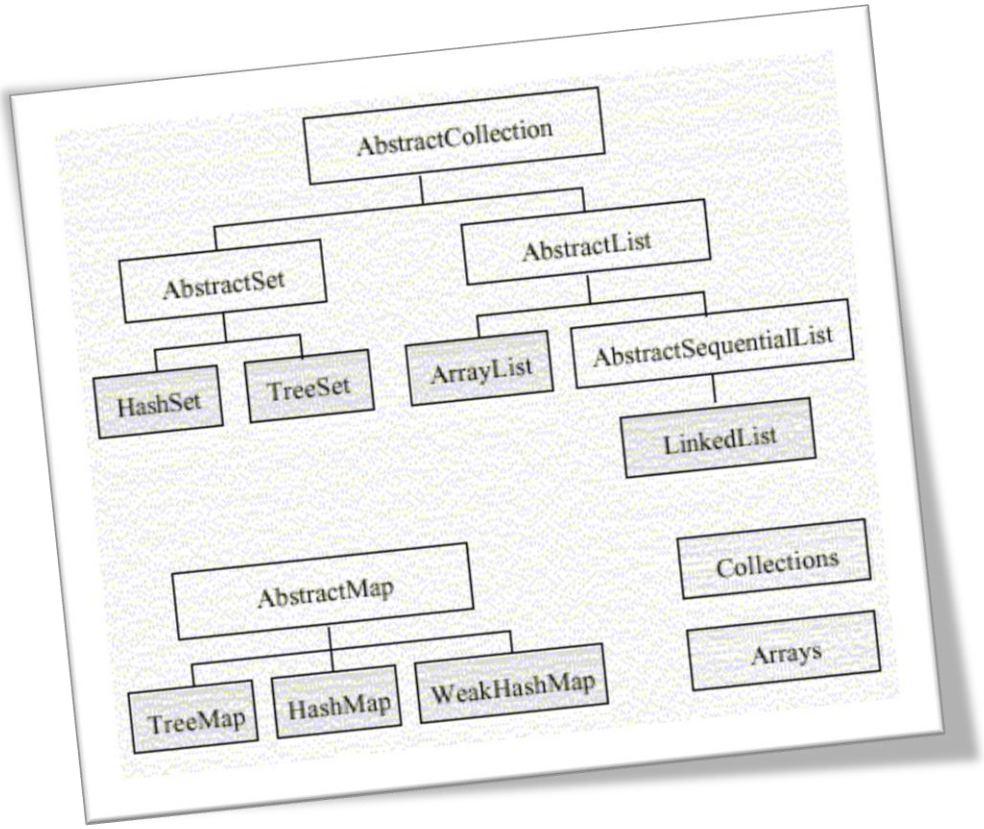
Las interfaces pueden extender otras interfaces y, a diferencia de las clases, una interface puede extender más de una interface. La sintaxis es:

```
interface nombre_interface  extends nombre_interface  , . . . {
    tipo_retorno nombre_metodo ( lista_argumentos ) ;
    . . .
}
```

PASO 8.-

Ejemplos de Jerarquías en JAVA




AbstractCollection
AbstractSet
AbstractList
HashSet
TreeSet
ArrayList
AbstractSequentialList
LinkedList
AbstractMap
Collections
Arrays
TreeMap
HashMap
WeakHashMap

Ejercicio.- 1

Crear una clase pública de nombre EjercicioString1 que contenga sólo al método main y partiendo de la String "En mi próxima vida, creeré en la reencarnación" declarada e inicializada muestre por consola lo siguiente:

1. Su longitud
2. El carácter asociado al índice 7
3. La subcadena "creeré"
4. El índice que ocupa el carácter 'x'
5. La String transformada en mayúsculas
6. Por último, comprobar si el primer carácter de la String es 'E' y mostrar por consola un mensaje que lo indique.

Por consola:

Longitud de la cadena: 46

Índice 7 asociado al carácter r

La subcadena generada desde 20 incluido hasta 26 excluido es creere

El índice del carácter x es 9

La cadena en mayúsculas es: EN MI PROXIMA VIDA, CREERE EN LA REENCARNACION

La String comienza por E

Ejercicio.- 2

Crear una clase pública de nombre EjercicioString2 que contenga sólo al método main y que muestre por consola el número de veces que aparece la letra "a" en la siguiente String "Mañana es sábado sabadete y voy a irme a tomar unas copillas por los barrios bajos de Logroño"

Por consola:

La letra a aparece 14 veces

Ejercicio.- 3

Verificar si una cadena de texto almacenada en la String nif, es un NIF correcto o no. Si lo es, se mostrará por consola su parte numérica; si no lo es se mostrará el mensaje "NIF no valido". Se tendrá en cuenta lo siguiente:

Suponer que los NIFs tienen 8 dígitos y, a continuación, una letra (no importa que sea mayúscula o minúscula).

PISTAS: dos condiciones que debe cumplir el NIF: tener 9 caracteres y que el último sea una letra. Comprobado esto, verificar que el resto de caracteres son dígitos.

RECOMENDACIONES:

Usar el método length() de java.lang.String para conocer el número de caracteres de una cadena de texto.

Usar el método estático isLetter(char c) de java.lang.Character para comprobar que un carácter es una letra.

Usar el método estático isDigit(char c) de java.lang.Character para comprobar que un carácter es un dígito.

Usar el método substring(int inicio, int fin) de java.lang.String para obtener la parte numérica del nif.

Ejercicio.- 4

Calcular el volumen de un cilindro y el de una esfera previa introducción de la altura y radio del cilindro, así como del radio de la esfera. Se definirá un método para el cálculo del volumen del cilindro y otro para el de la esfera. Se emplearán métodos estáticos de la clase Math y la variable de campo estática que almacena el valor de pi.

Volumen esfera=(4/3)*PI*R^3 Volumen cilindro=PI*R^2*H

NOTA: cuidado con las fórmulas que contienen fracciones. Java considera (4/3) como 1 ya que, por defecto, los números enteros se almacenan en una variable int y el cociente de dos enteros para el programa es otro entero. Habría que hacer que el numerador fuera un double para que el cociente también lo fuera. Cómo?

Por ejemplo sustituyendo (4/3) por (4.0/3). De este modo se tiene un cociente entre un double y un entero. Es decir, un double.

Ejercicio.- 5

Realizar un programa Java compuesto de una clase pública de nombre AdivinarNumero que contenga sólo al método main. Su objetivo será permitir que el usuario adivine un número entero generado aleatoriamente y comprendido entre [0,100] que se almacenará, dentro del código del programa, en una variable int a la que se llamará numero.

El programa pedirá un número por teclado e informará de si el número que introduce el usuario es mayor o menor que el que se trata de averiguar. Si no se acierta a la primera, no importa porque tiene que dejar introducir números de forma ininterrumpida. Cuando el usuario acierte, se mostrará un mensaje de felicitación y el número de intentos empleados. A tener en cuenta:

- Si el usuario introduce un numero no comprendido entre [0,100], el programa mostrará un mensaje informativo
- Si el usuario teclea asterisco, el programa deberá finalizar
- La generación aleatoria del número a adivinar se realizará con el método estático "void random()" de java.lang.Math.

```
int seleccionado = (int) Math.round((Math.random()*10))

private static int aleatorio(int max,int min) {
int num = (int)(Math.round(Math.random() * (max-min))) + min;
return num;
}
```

Ejercicio.- 6

Identifique los datos que decidiría utilizar para almacenar el estado de los siguientes objetos en función del contexto en el que se vayan a utilizar:

a. Un punto en el espacio.
b. Un segmento de recta.
c. Un polígono.
d. Una manzana (de las que se venden en un mercado).
e. Una carta (en Correos)
f. Un libro (en una biblioteca)
g. Una canción (en una aplicación para un reproductor MP3).
h. Un ordenador (en una red de ordenadores)

Declare las correspondientes clases en Java, defina los constructores que considere adecuados e implemente los correspondientes métodos para el acceso y la modificación del estado de los objetos (esto es, los métodos get y set).

Ejercicio.- 7

- Dado una secuencia de número leídos y almacenados en un vector A mostrar dichos números en orden.
- Dado una secuencia de número leídos y almacenados en un vector A y un número leído determinar si dicho número se encuentra o no en el vector.
- Leer una secuencia de n números y almacenar en un vector sus factoriales.
- Leer n números y almacenarlos de manera ordenada en un vector.
- Dado dos matrices A y B obtener la suma.
- Dado una matriz determinar la posición (i,j) del mayor.

Ejercicio.- 8

Rellenar los 3 sitios donde aparecen puntos suspensivos en el siguiente código JAVA para conseguir que en pantalla aparezca la palabra *"MAS"* al ejecutar el programa de la clase *PruebaTest*:

```
class Test {
private int val=50;
public char devolver(char ent, int cond) {
        char res = 'M';
        if (cond!=val) {
                val = cond;
                res = ent;
                }
        return res;
        }
} /* Fin clase Test */



class PruebaTest {
public static void main(String args[]) {
        Test ref = new Test();
        int num = 100;
        char c = 'X';
        System.out.print( "" +  ref.devolver( c, num - ..... ) );
        c = 'S' ;
        System.out.print( "" +  ref.devolver(....., num ) );
```

```
        c = 'S' ;
        num = num / 2;
        System.out.println( "" + ref.devolver( c, num * ..... ) );
        c = 'S' ;
        }
} /* Fin clase PruebaTest */
```

Ejercicio.- 9

Describir y corregir los 4 errores del siguiente código JAVA:

```
class Local {
public double comprobar Primero (datos[], long busca) {
   if (busca!=primero) {
      primero = busca;
      }
   return (datos[0]==busca);
   }
}
```

Ejercicio.- 10

Definir los campos (no los métodos) de las clases de objetos necesarias para almacenar la información relativa a unas muestras de alcoholemia recogidas en un análisis preventivo de la Dirección General de Tráfico, de forma que:

1. Para cada conductor interesa tener su nombre; D.N.I. y una colección de referencias a las muestras que se le han tomado durante el período de estudio.

2. Cada muestra tendrá los siguientes datos: Día y hora de la muestra; código del puesto de control preventivo (dos letras y dos número); matricula del vehículo; una referencia al conductor que conducía el vehículo en el momento de la toma de la muestra; y la tasa de alcohol espirado en aire (entre 0 y 2.5 mg/l).

Ejercicio.- 11

Dadas las clases **Cuadrado** y **Superficie** siguientes, escribir un método denominado **cuadrar** que reciba un objeto de tipo *Superficie* y devuelva otro de tipo *Cuadrado*, tal que el cuadrado devuelto tenga el mismo área que la superficie dada. Utilizar Math.sqrt() para calcular la raíz cuadrada.

```
class Cuadrado {
   public float lado;
   }

class Superficie {
   public float area;
   }
```

Ejercicio.- 12

Explicar cuáles son y por qué se producen las salidas en pantalla que aparecen al ejecutar el programa contenido en la clase **PruebaClaseA**.

```
class ClaseA {
   public short campo;
   }


class PruebaClaseA {
public static void main(String args[]) {
   ClaseA a1 = new ClaseA();
   ClaseA a2 = new ClaseA();
   ClaseA a3 = new ClaseA();
   a1.campo=150;
   a2.campo=150;
   a3 = a2;
   if (a1 == a2) { System.out.println("Check1: True"); }
   if (a1 == a3) { System.out.println("Check2: True"); }
   if (a2 == a3) { System.out.println("Check3: True"); }
   }
}
```

Nota: Las comprobaciones de igualdad entre referencias comprueban si el objeto referenciado por ambas es el mismo, no si los contenidos de los campos son los mismos.

Ejercicio.- 13

Escribir el código JAVA de un programa que admita como parámetros en la línea de comando dos datos:

- Un número de tipo *int* al que llamaremos tamMuestra
- Un número de tipo *double* al que llamaremos limite

Si el número de parámetros no es exactamente 2 debe escribir un mensaje de error y terminar. Si no hay error, el programa debe pedir al usuario que introduzca un número de datos de tipo *double* igual a *tamMuestra*. Al terminar el programa informará al usuario del número de datos introducidos que cumplían la condición de estar en el intervalo: [-limite,+limite].

Ejercicio.- 14

Necesitamos programar una clase JAVA para almacenar los datos sobre actividad sísmica en un punto geográfico a lo largo de los 365 días del año. Los valores a almacenar estarán expresados en la escala de Ritcher, tomados por un sismógrafo con una precisión de hasta 10^{-3} (deducir el tipo de dato).

La clase necesitará los siguientes métodos:

- **introducirSiguienteDato** : que colocará el dato recibido por parámetro en el siguiente día que no esté utilizado. Cada vez que se llama a este método se pasa al día siguiente (el objeto debe *recordar* el número de días que se han introducido hasta el momento).

- **diasTranscurridos** : que devolverá el número del última día para el que se ha introducido un dato.

- **recuperarDato**: que devolverá el valor del dato relativo al número de día que se recibe como parámetro.

- **mediaActividad** : que devolverá el valor medio de la actividad producida durante los días que ya han sido introducidos.

Ejercicio.- 15

Describir y corregir los 4 errores del siguiente código JAVA:

```
public void metodoA(int[], int pos) {
   int sum=0;
   while( i < pos ) {
      sum = sum + vec[i];
      }
   return sum;
   }
```

Ejercicio.- 16

Dada la siguiente clase Triangulo:

```
class Triangulo {
   public float base, altura;
   public float area() {
      return (base * altura) / 2;
      }
   }
```

Escribir un método denominado **achatar** que reciba como argumento un objeto de tipo *Triangulo* y devuelva otro objeto de tipo *Triangulo* con un 10% menos de altura que el original (h'=0.9h), pero ajustando la base para que tengan el mismo área (b'=(b*h)/h').

Ejercicio.- 17

¿Cuáles son las salidas en pantalla que produce el programa contenido en esta clase?

```
class ClaseB {
private int rec=1;


public int test(int a, int b) {
   if (a==rec) {
```

```
            System.out.println("Ok: "+a);
        }
        rec=rec*b;
        return rec;
    }


    public static void main(String args[]) {
        ClaseB refb = new ClaseB();
        int x=2;
        int y=1;
        y = refb.test(x,y);
        System.out.println("Paso 1, res="+y);
        y = refb.test(y,x);
        System.out.println("Paso 2, res="+y);
        x = refb.test(x,0);
        System.out.println("Paso 3, res="+x);
        }
    }
```

Ejercicio.- 18

Supongamos una máquina que dispone de un conjunto de N pulsadores numerados de 0 a N-1, cada uno de los cuales puede estar activo o no, en un instante de tiempo. Necesitamos programar una clase en JAVA, que permita almacenar la información de cuáles de esos pulsadores están o no activos en un instante dado. La clase necesitará los siguientes métodos:

activarPulsador : que recibirá un número de pulsador y hará que figure como activo.

desactivarPulsador : que recibirá un número de pulsador y hará que figure como NO activo.

activado : que recibirá un número de pulsador y devolverá si este está activo o no.

numActivos : que devolverá el número de pulsadores que están activados.

Ejercicio.- 19

Dada la clase **Trayecto** siguiente escribir un método denominado **uneTrayectos** que reciba como parámetros dos objetos de tipo **Trayecto** y devuelva un nuevo objeto de esa misma clase con: el origen del primero; el destino del segundo; y como distancia la suma de las distancias de los dos trayectos originales.

```
class Trayecto {
    public String origen;
    public String destino;
    public double distancia;
    }
```

Ejercicio.- 20

Hacer un seguimiento del programa y describir razonadamente qué salidas produce en pantalla.

```
class Interruptor {
private boolean abierto = true;
public int impulso( int tension ) {
   int result = 0;
   if ( abierto ) {
      abierto = false;
      result = tension;
      }
   else {
      abierto = true;
      }
   return result;
   }
}

public static void main( String args[] ) {
   Interruptor in1, in2;

   in1 = new Interruptor();
   in2 = new Interruptor();

   int control = 100;
```

```
control = in1.impulso( 50 ) + in2.impulso( control );
System.out.println( "Salida1: " + control );
System.out.println( "Salida2: " + in1.impulso( control ) );
System.out.println( "Salida3: " + in1.impulso( control + 100 ) );
System.out.println( "Salida4: " + in2.impulso( 250 ) );
}
```

Ejercicio.- 21

Un polinomio p(x) = Sumatorio(ai xi) puede representarse con un array de números de tipo double de la siguiente forma: En la posición i del array estará el valor del coeficiente ai. El array tendrá tantos elementos como el grado del polinomio más 1 (para poder almacenar también el término de grado (0).

Por tanto 12x5 -4x3 + 7x -3 se representará con un array de longitud seis: [-3 , 7 , 0 , -4 , 0 , 12]

Escribir un método denominado sumaPolinomios. El método recibe 2 polinomios representados por dos arrays de tipo double. En caso de que los dos polinomios no sean del mismo grado, devolvera null. Si son del mismo grado, devolverá un nuevo array de tipo double con la suma de los dos polinomios de entrada (el valor de cada coeficiente en el array de salida es la suma de los coeficientes del mismo grado de los arrays de entrada).

Ejercicio.- 22

Escribir el código JAVA de una clase denominada **Hucha**, con las siguientes características. Cada objeto de esta clase representará una hucha electrónica, que sólo puede contener monedas de 1Eur., de 2Eur., o billetes de 5Eur. La hucha deberá almacenar información acerca de cuantos billetes y monedas de cada clase tiene en su interior. Además, cada hucha podrá estar *cerrada* o no.

De una hucha cerrada no se puede sacar dinero.

Exisitirán los siguientes métodos:

1. **Constructor** : El constructor recibirá un valor que servirá para fijar si la hucha está cerrada o no.
2. **abrirHucha** : Hará que la hucha quede abierta. No importa si antes estaba abierta o cerrada.
3. **ingresar** : Recibirá un número de monedas de 1Eur. otro de monedas de 2Eur., y otro de billetes de 5Eur., que se sumarán al contenido de la hucha.

4. **valorTotal** : Devolverá el valor total en Eur. del contenido de la hucha.
5. **hayBilletes** : Devolverá *false* si no hay billetes en la hucha, *true* en caso contrario.
6. **sacarBilletes** : Recibirá un número de billetes. Si la hucha está cerrada debe escribir un mensaje en pantalla y no hacer nada. Si el número de billetes que se desea sacar es mayor que el número de billetes que queda en la hucha también se mostrará un mensaje y no se hará nada. En cualquier otro caso, se restará el número de billetes.

Ejercicio.- 23

Dadas las clases Lugar y Conexion siguientes, escribir un método denominado **conectar** que reciba como parámetros dos objetos de tipo *Lugar* y devuelva un nuevo objeto *Conexion* con los identificadores de cada uno de los objetos de entrada, y la distancia entre los mismos. Para calcular la distancia utilizar: double Math.sqrt(double).

```
class Lugar {
    public long identificador;
    public double x;
    public double y;
    }

class Conexion {
    public long idA;
    public long idB;
    public double distancia;
    }
```

Ejercicio.- 24

Dado el esqueleto de la clase Hormiga siguiente, escribir el código de un método denominado **deficit**, que reciba un array de *Hormiga* y que devuelva verdadero si el total de comida consumida por toda la población representada en el array supera el total de comida recogida. Devolverá falso en caso contrario.

```
class Hormiga {
   ...
   public long comidaRecogida( ) {
      ... }
```

```
public long comidaConsumida( ) {
   ... }
...
}
```

Ejercicio.- 25

Escribir el código JAVA de una clase denominada **ObraEdificio**, con las siguientes características. Cada objeto de esta clase deberá tener información acerca de la superficie total a cimentar de una obra, y de la superficie que ya ha sido cimentada (en m^2, sin decimales). Existirán los siguientes métodos:

1. **Constructor** : En la creación se fijará el area total a cimentar para este edificio.
2. **cimentando** : Recibirá un número indicando los metros de superficie que se están cimentando en ese momento. Si la suma de la superficie previamente cimentada más la que se supone que se está cimentando ahora, es mayor que la superficie total a cimentar, se escribirá un mensaje de error, y no se modificará nada. En caso contrario, se acumulará la cantidad de metros que se están cimentando con los que ya se habían cimentado antes.
3. **restaPorCimentar** : Devolverá los metros que faltan por cimentar para alcanzar la superficie total.
4. **terminado** : Devolverá verdadero si la superficie cimentada ya es igual a la superficie total a cimentar.

Ejercicio.- 26

Dada la clase **Viaje** siguiente escribir un método denominado **uneViaje** que reciba como parámetros dos tipos **Viaje** y devuelva un nuevo objeto de esa misma clase con: el origen del primero; el destino del segundo; y como distancia la suma de las distancias de los dos viajes originales si el destino del primero coincide con el origen del segundo, sino se cumple dicha condición se devolverá NULL.

```
class Trayecto {
    public String origen;
    public String destino;
    public double distancia;
    }
```

Ejercicio.- 27

Se desea llevar un control del estado de una cuenta corriente; la cuenta corriente está caracterizada por su saldo y sobre ella se pueden realizar tres tipos de operaciones:

- saldo: devuelve el saldo de la cuenta (puede ser negativo).
- imposición (cantidad): ingresa en la cuenta una cantidad de dinero.
- reintegro (cantidad): saca de la cuenta una determinada cantidad de dinero.

Suponga que la cuenta inicialmente tiene un saldo de cero. Escriba una clase CuentaCorriente que implemente la funcionalidad descrita; escriba un pequeño programa principal para probar su funcionamiento.

Ejercicio.- 28

Se quiere definir una clase que permita controlar un sintonizador digital de emisoras FM; concretamente, lo que se desea es dotar al controlador de una interfaz que permita subir (up) o bajar (down) la frecuencia (en saltos de 0.5 MHz) y mostrar la frecuencia sintonizada en un momento dado (display). Supondremos que el rango de frecuencias a manejar oscila entre los 80 Mhz y los 108 MHz y que al inicio, el controlador sintoniza a 80 MHz. Si durante una operación de subida o bajada se sobrepasa uno de los dos límites, la frecuencia sintonizada debe pasar a ser la del extremo contrario.

Ejercicio.- 29

Queremos modelar una casa con muchas bombillas, de forma que cada bombilla se puede encender o apagar individualmente. Para ello haremos una clase Bombilla con una variable privada que indique si está encendida o apagada, así como un método que nos diga si una bombilla concreta está. Cada objeto Bombilla se enciende y se apaga individualmente; pero sólo responde que está encendida si su interruptor particular está activado y además hay luz general. El código Java tiene esta estructura (Escriba un programa "main" que lo pruebe):

```
class Bombilla
{
private ...; // interruptor
public void enciender () { ... }
public void apagar () { ... }
public boolean encendida () { ... }
 public Bombilla (){ ... }
}

class Iluminacion {
private ...;  // Bombillas de la casa
private ...;  // Número de bombillas de la casa
public void activaGeneral () { ... }
public void desactivaGeneral () { ... }
public apagar_bombilla() { ... }
public encender_bombilla() { ... }
public numero_bombillas_encendidas (){ ... }
public numero_bombillas_apagadas (){ ... }
public Iluminación { ... }
}
```

Ejercicio.- 30

Escriba una clase Java que represente a los Gpolígono (2,5 puntos):

Atributos:

Números de lados (*int*). int nlados;

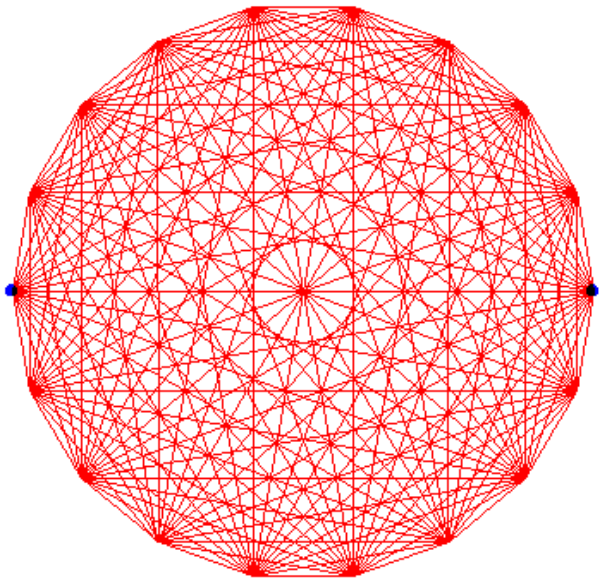
Vector de (*class* ***Punto*****)**. Punto [] posicionamiento;

Color de relleno (*class* ***Rgb***). Rgb color;

Clase Punto	Clase Rgb
class Punto {	**class Rgb {**
public int x;	**public int r;**
public int y;	**public int g;**
……..	**public int b;**
}	**……. }**

Métodos:

- public gpoligono(){// Constructor por defecto}
- public gpoligono(int num, Rgb c){/* Mediante este constructor sobrecargado se proporciona el número de lados y su color. Dentro del cuerpo de este método te preguntará por cada uno de los puntos (X,Y) que formarán su posicionamiento en el espacio.*/}
- public int getlados(){// Método que devuelve los lados del Gpolígono.}
- public Rgb getcolor(){// Método que devuelve el color del Gpolígono.}
- public Punto getposicion_n(int p){/* Método que devuelve un punto del Gpolígono.*/}
- public boolean compareTopoligonos(Gpoligono a){/* Método que dado un polígono determina si es igual a él. Son iguales cuando coinciden el color, número de lados y su posicionamiento. */}



Ejercicio.- 31

Escriba una clase Java que represente un Viaje (2,5 puntos):

Atributos:

- Números de trayectos (*int*). int ntra;
- Vector de (*class Trayecto*). Trayecto [] sectrayectos;

Clase Trayectoria

```
public class Trayecto {
    public String origen;
    public String destino;
    public double
distancia;
    .......
    }
```

Métodos:

- public viaje(){// Constructor por defecto}
- public viaje(int num){/* Mediante este constructor sobrecargado se proporciona el número de trayectos. Dentro del cuerpo de este método te preguntará cada uno de los trayectos que forman el viaje.*/}
- public Trayecto mayortrayecto()){/* Devuelve el mayor trayecto en longitug.*/}
- public double distanciaviaje(){// Resultado de la suma de los trayectos}
- public boolean combinables(Viaje v1) {/*Si coincide el destino de último trayecto de v1 con el origen del primer trayecto de this, los trayectos son combinables true*/}.

Ejercicio.- 32

Construir una clase Factura que descienda de la clase Precio y que incluya dos atributos específicos llamados emisor y cliente y, al menos, un método llamado imprimirFactura.

Ejercicio.- 33

Construir una clase final Math2 que amplíe las declaraciones de métodos estáticos de la clase Math y que incluya funciones que devuelvan, respectivamente, el máximo, el mínimo, el sumatorio, la media aritmética y la media geométrica de un array de números reales dado como parámetro.

Ejercicio.- 34

Diseñe jerarquías de clases para representar los siguientes conjuntos de objetos:

1. Una colección de CDs, entre los cuales hay discos de música (CDs de audio), discos de música en MP3 (CD-ROMs con música), discos de aplicaciones (CD-ROMs con software) y discos de datos (CD-ROMs con datos y documentos).
2. Los diferentes productos que se pueden encontrar en una tienda de electrónica, que tienen un conjunto de características comunes (precio, código de barras…) y una serie de características específicas de cada producto.
3. Los objetos de una colección de monedas/billetes/sellos.

Implemente en Java las jerarquías de clases que haya diseñado (incluyendo sus variables de instancia, sus constructores y sus métodos get/set). A continuación, escriba sendos programas que realicen las siguientes tareas:

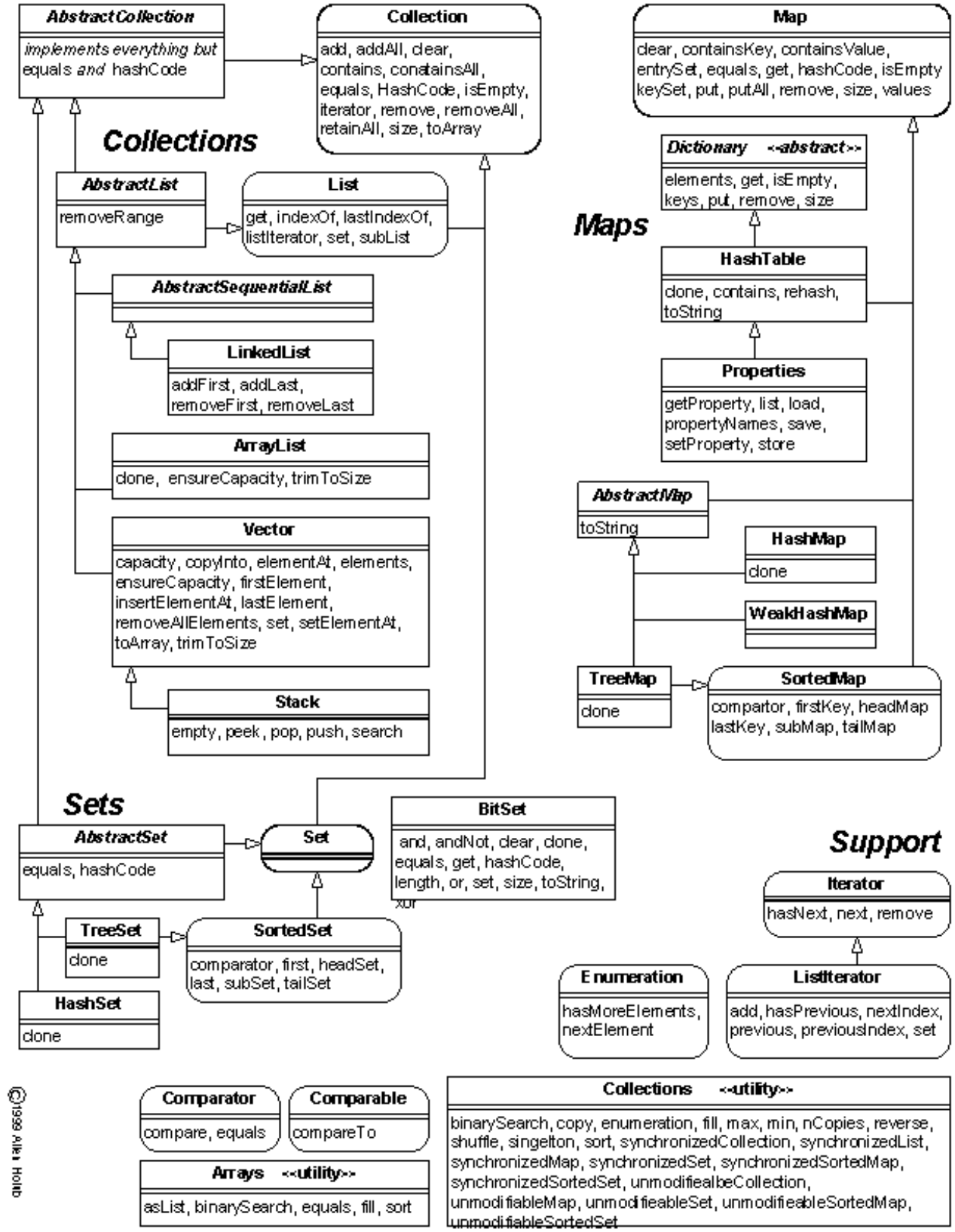
a. Buscar y mostrar todos los datos de un CD concreto (se recomienda definir el método toString en cada una de las subclases de CD).

b. Crear un carrito de la compra en el que se pueden incluir productos y emitir un ticket en el que figuren los datos de cada producto del carrito, incluyendo su precio y el importe total de la compra.

c. Un listado de todos los objetos coleccionables cuya descripción incluya una cadena de caracteres que el programa reciba como parámetro.

Ejercicio.- 35

Analiza las clases más representativas de la Jerarquía Collection, más concretamente la Clase ArraList y la Clase Vector. Pon algún ejemplo.



The "operations" compartment shows only those methods that are introduced in a given class; inherited and overriden methods are not shown

Ejercicio.- 36

Analiza la clase File y pon un ejemplo.

La clase *File* tiene tres constructores

- *File(String path)*
- *File(String path, String name)*
- *File(File dir, String name)*

El parámetro *path* indica el camino hacia el directorio donde se encuentra el archivo, y *name* indica el nombre del archivo. Los métodos más importantes que describe esta clase son los siguientes:

- *String getName()*
- *String getPath()*
- *String getAbsolutePath()*
- *boolean exists()*
- *boolean canWrite()*
- *boolean canRead*
- *boolean isFile()*
- *boolean isDirectory()*
- *boolean isAbsolute()*
- *long lastModified()*
- *long length()*
- *boolean mkdir()*
- *boolean mkdirs()*
- *boolean renameTo(File dest);*
- *boolean delete()*
- *String[] list()*
- *String[] list(FilenameFilter filter)*

Ejercicio.- 37

Analiza las siguientes clases y pon un ejemplo.

- Arrays.
- ArrayList
- Vector
- Comparator y Comparable.

https://sites.google.com/site/apuntesdejava/Home/comparator-y-comparable

http://es.debugmodeon.com/articulo/interfaces-comparator-y-comparable-ii

```
import java.util.Comparator;
public class NombreComparator implements Comparator {
/*
 negativo si o1 < o2
 cero     si o1 = o2
 positivo si o1 > o2
*/
  public int compare(Object o1, Object o2) {
    Persona u1 = (Persona) o1;
    Persona u2 = (Persona) o2;
    return u1.nombre.compareTo(u2.nombre);
  }
}
```

```
import java.util.Comparator;
public class EdadComparator implements Comparator {
/*
 negativo si o1 < o2
 cero     si o1 = o2
 positivo si o1 > o2
*/
 public int compare(Object o1, Object o2) {
  Persona u1 = (Persona) o1;
    Persona u2 = (Persona) o2;
    return u1.edad - u2.edad;
  }
}
```

```
public static void main(String[] args) {
        ArrayList lista = new ArrayList();
        lista.add(new Persona(32, "Juan Antonio"));
        lista.add(new Persona(21, "Zarauz"));
        lista.add(new Persona(15, "Antonio"));
        lista.add(new Persona(2, "García"));
        System.out.println("\n Ahora ordenados por nombre...");
        Collections.sort(lista, new NombreComparator());
        for(int i=0;i<lista.size();i++)
        {
           System.out.println(((Persona)lista.get(i)).nombre+ " - "+((Persona)lista.get(i)).edad);
        }
        System.out.println("\n y ahora ordenados por edad...");
        Collections.sort(lista, new EdadComparator());
        for(int i=0;i<lista.size();i++)
        {
           System.out.println(((Persona)lista.get(i)).nombre+ " - "+((Persona)lista.get(i)).edad);
        }
}
```