LPS: Array, String y “Stream”

Federico Peinado
www.federicopeinado.es

Depto. de Ingeniería del Software e Inteligencia Artificial
disia.fdi.ucm.es

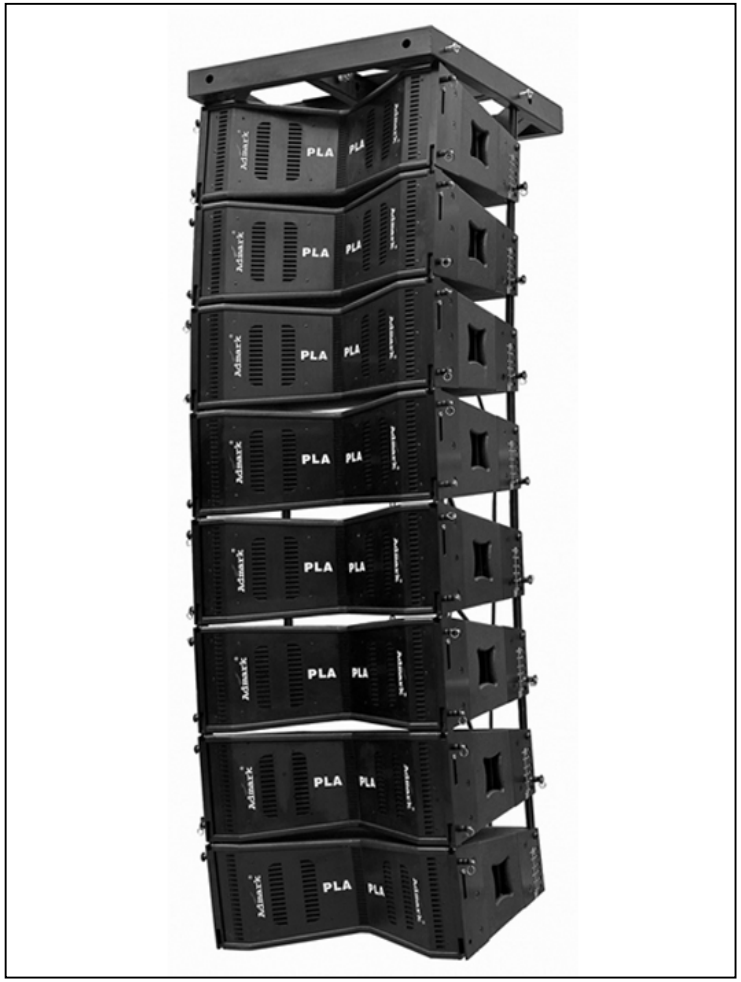
Facultad de Informática
www.fdi.ucm.es

Universidad Complutense de Madrid
www.ucm.es

Utilidades básicas de Java

- Los usos básicos de Java son similares a los de otros lenguajes de programación
  - Array (para representar una serie de elementos)
  - String (para representar cadenas de caracteres)
  - Varios tipos de “Streams”, para representar flujos de entrada/salida al sistema)

Array



Array

- Clase “especial” de Java que representa una serie de tamaño fijo de variables del mismo tipo
- Construcción en dos pasos
  1. Declaración

```
int[] v; // También se acepta int v[];
```

  2. Inicialización

```
v = new int[10];
```

  - Ambos pasos se pueden hacer al mismo tiempo

```
int[] v1 = new int[10];
int[] v2 = {3, 4, 5};
```

- Las variables contenidas pueden ser referencias a objetos

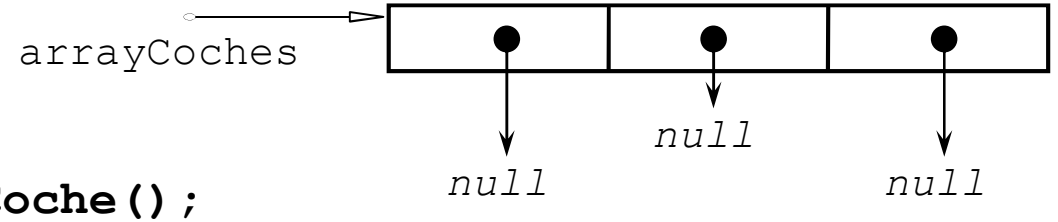
```
Rectangulo[] r = new Rectangulo[2];
r[0] = new Coordenada(0, 0);
r[1] = new Coordenada(1, 0);

Rectangulo[] r2 = r; // Recuerda, ¡esto no es clonar!
```

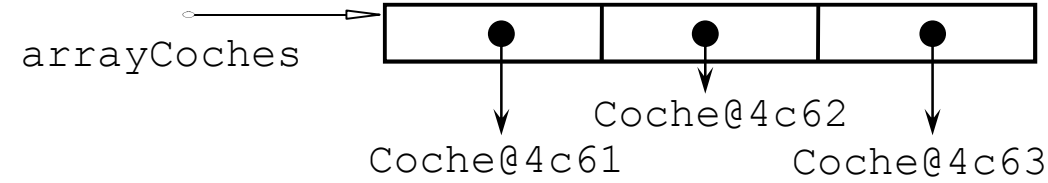
Funcionamiento en memoria

- Valores de tipos simples como contenido
  - Se reserva espacio en memoria para **`N * tamañodelTipo`**
  - Los datos se guardan en el propio objeto Array
- Referencias a objetos como contenido
  - Se reserva memoria para **`N`** referencias
  - En el objeto Array sólo hay guardadas referencias en cada posición

```
Coche[] arrayCoches = new Coche[3];
```




```
arrayCoches[0] = new Coche();
arrayCoches[1] = new Coche();
arrayCoches[2] = new Coche();
```



Manejo básico

- Las posiciones van de 0 a N-1 (como en C++)
  - Si accedemos fuera de estas posiciones se lanza una excepción específica para indicar el error
    `IndexOutOfBoundsException`
  - La longitud está disponible en un atributo length

```
int v[] = new v[10];
int num = v.length;  // num pasa a valer 10
```

- Acceso

```
v[0] = 0;
int x = v[0];
```

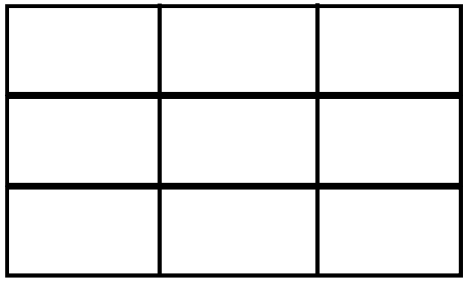
- Copia
  - Requiere usar un método estático de la clase **`System`**

```
System.arraycopy(Object src, int srcPos,
                 Object dest, int destPos, int length)
```

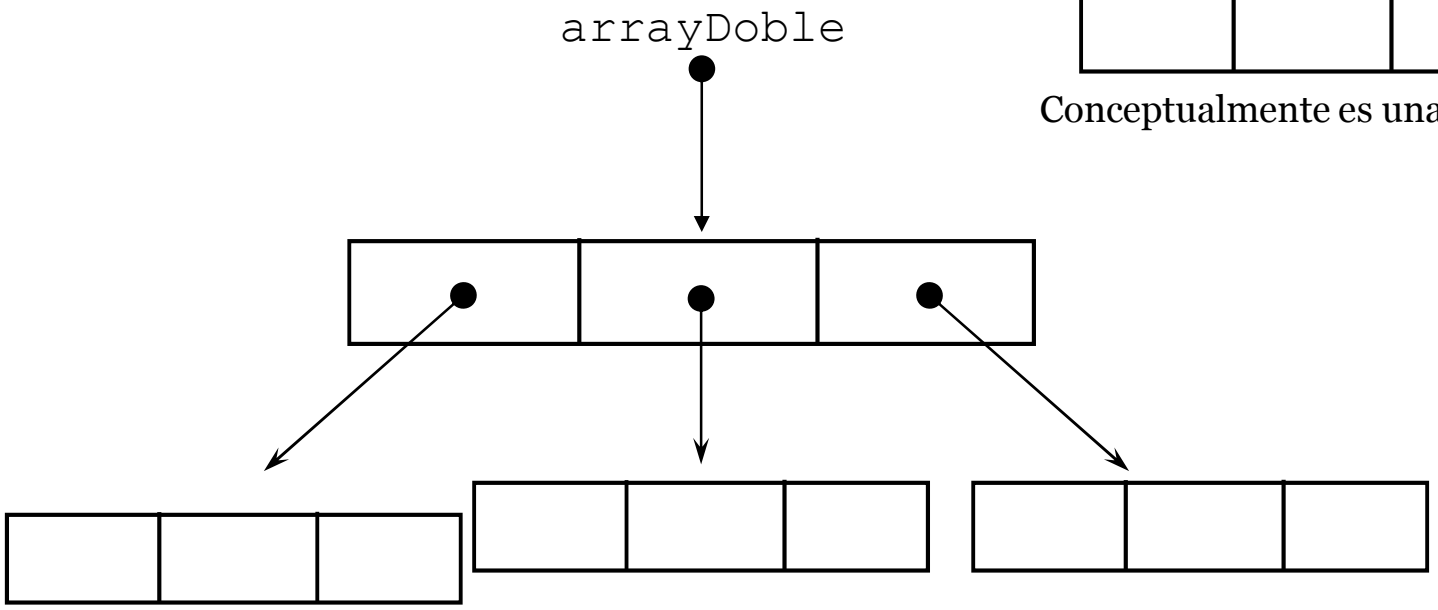
Arrays multidimensionales

- Arrays cuyo contenido son a su vez arrays

```
int[][] arrayDoble;
arrayDoble = new int[3][3];
```



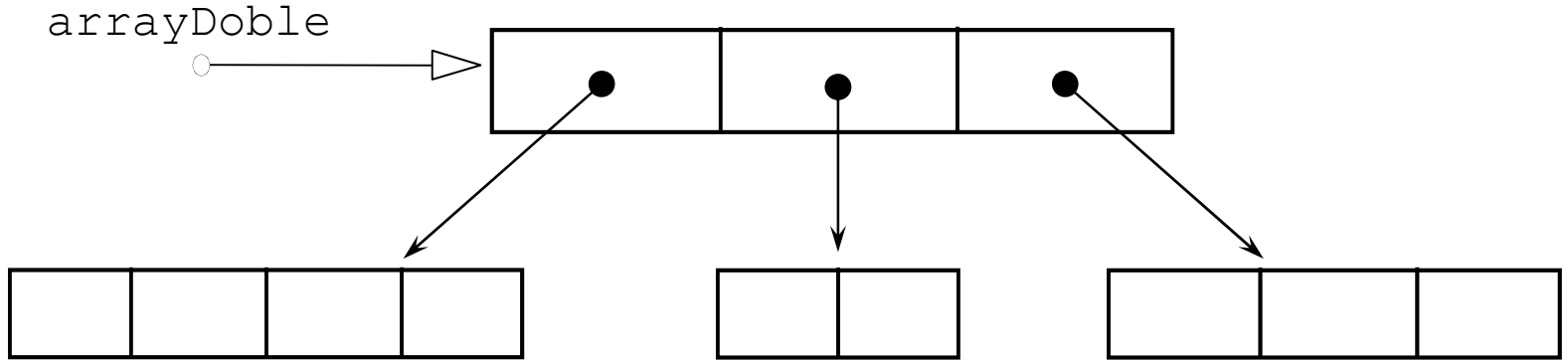
Conceptualmente es una matriz



Multidimensionales y asimétricos

- Al ser el array multidimensional una referencia, se permite que cada array de su contenido sea distinto

```
int[][] arrayDoble;
arrayDoble = new int[][3];
arrayDoble[0] = new int[4];
arrayDoble[1] = new int[2];
arrayDoble[2] = new int[3];
```



String

String

- Clase "especial" de Java que representa cadenas de caracteres
  `java.lang.String`
  - Ofrece varios métodos útiles para su manipulación
    - El contenido de la cadena se considera **inmutable**
    - Funcionan accediendo y sirviendo para hacer copias
- Es una clase **`final`**
  - Este modificador indica que ninguna otra clase puede heredar de ella (es debido a cuestiones *peliagudas* de su implementación interna)

Métodos disponibles

`toUpperCase()` / `toLowerCase()`

- Devuelven la cadena copiada toda en mayúsculas/minúsculas

`length()`

- Devuelve la longitud de la cadena
  - En este caso sí disponemos de un método, no como en los arrays

`charAt(i)`

- Devuelve el carácter en la posición i de la cadena

`compareTo(s)`

- Comparación lexicográfica entre cadenas (según el orden "alfabético" en que se colocarían devuelve <0, 0 ó >0)
- Devuelve *cero* en el caso de que **`equals(s)`** devuelve *cierto*

`compareToIgnoreCase(s)`

- Igual que el anterior pero ignorando mayúsculas/minúsculas

`startsWith(s)` / `endsWith(s)`

- Comprueba si la cadena empieza/termina con un determinado prefijo/sufijo

…

Concatenación de cadenas

- Se usa el operador **+** (ó **+=**)
  - El compilador usará de forma implícita **`StringBuilder`**
  - Realiza automáticamente la conversión de tipos distintos a String

```
String s = "hola ";
s = s + "mundo " + 47;
```

- **`java.lang.StringBuilder`**
  - Conjunto mutable de caracteres
- **`java.lang.StringBuffer`**
  - Conjunto mutable de caracteres y *seguro ante hebras*
  - Se añaden nuevos caracteres mediante **`append`**
  - Se puede convertir en una cadena mediante **`toString()`**

```
StringBuffer sb = new StringBuffer();
sb.append("Hola ");
sb.append(nombre);
String s = sb.toString();
System.out.println(s);
```

Conversiones explícitas

```
Integer(String cadena); // String a Integer

Integer.valueOf("4"); // String a Integer
Integer.valueOf(cadena); // String a Integer

int k = Integer.parseInt("4"); // String a int
int k = Integer.parseInt(cadena); // String a int


Integer objInt = new Integer(450);
String s = objInt.toString(); // Integer a String
```

Separación de cadenas

`substring(beginIndex, endIndex)` **`substring(beginIndex)`**

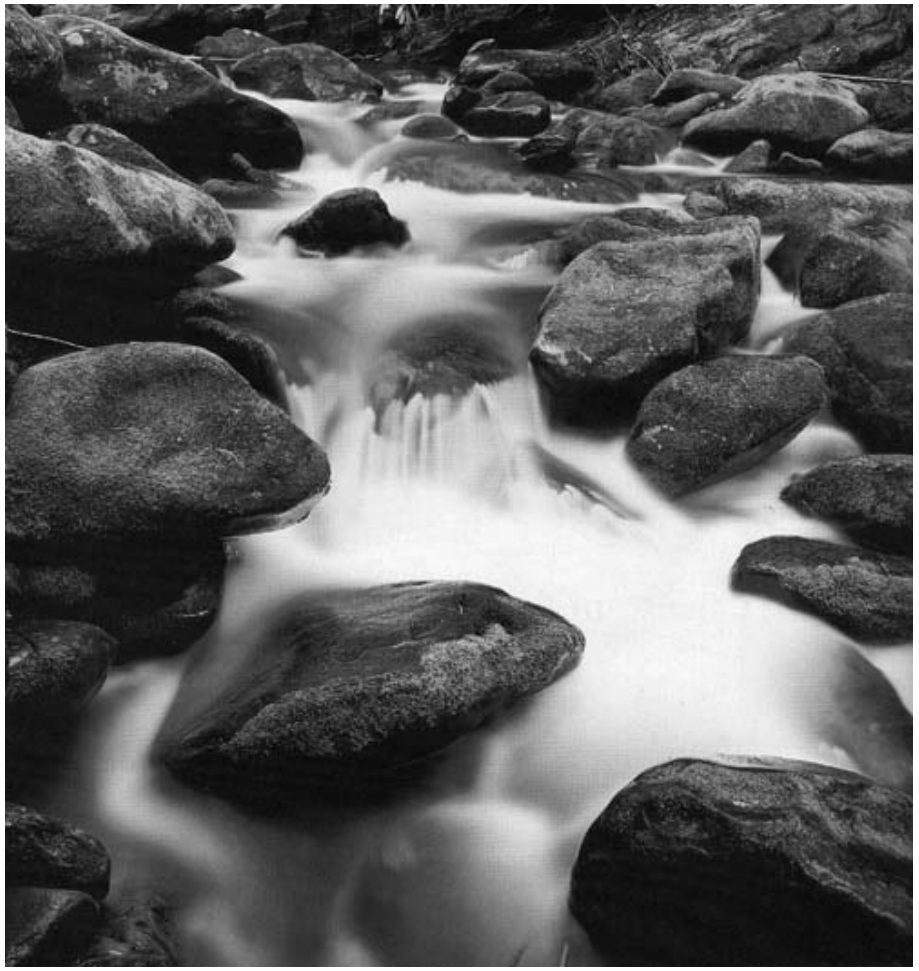
- Devuelve una subcadena que empieza en **`beginIndex`** (y termina en **`endIndex`**, si éste se especifica)

`StringTokenizer`

- Separación de una cadena en *tokens* (o lexemas)
- Recibe la cadena en el constructor y se pueden indicar cuales han de ser los delimitadores de *tokens*
  - Por defecto son los espacios en blanco, saltos de línea, etc.
- **`hasMoreTokens()`**
  - Verifica si hay más *tokens* en la cadena
- **`nextToken()`**
  - Devuelve el siguiente *token* encontrado

```
StringTokenizer st = new StringTokenizer("this is a
test");
while (st.hasMoreTokens()) {
    System.out.println(st.nextToken());
}
```

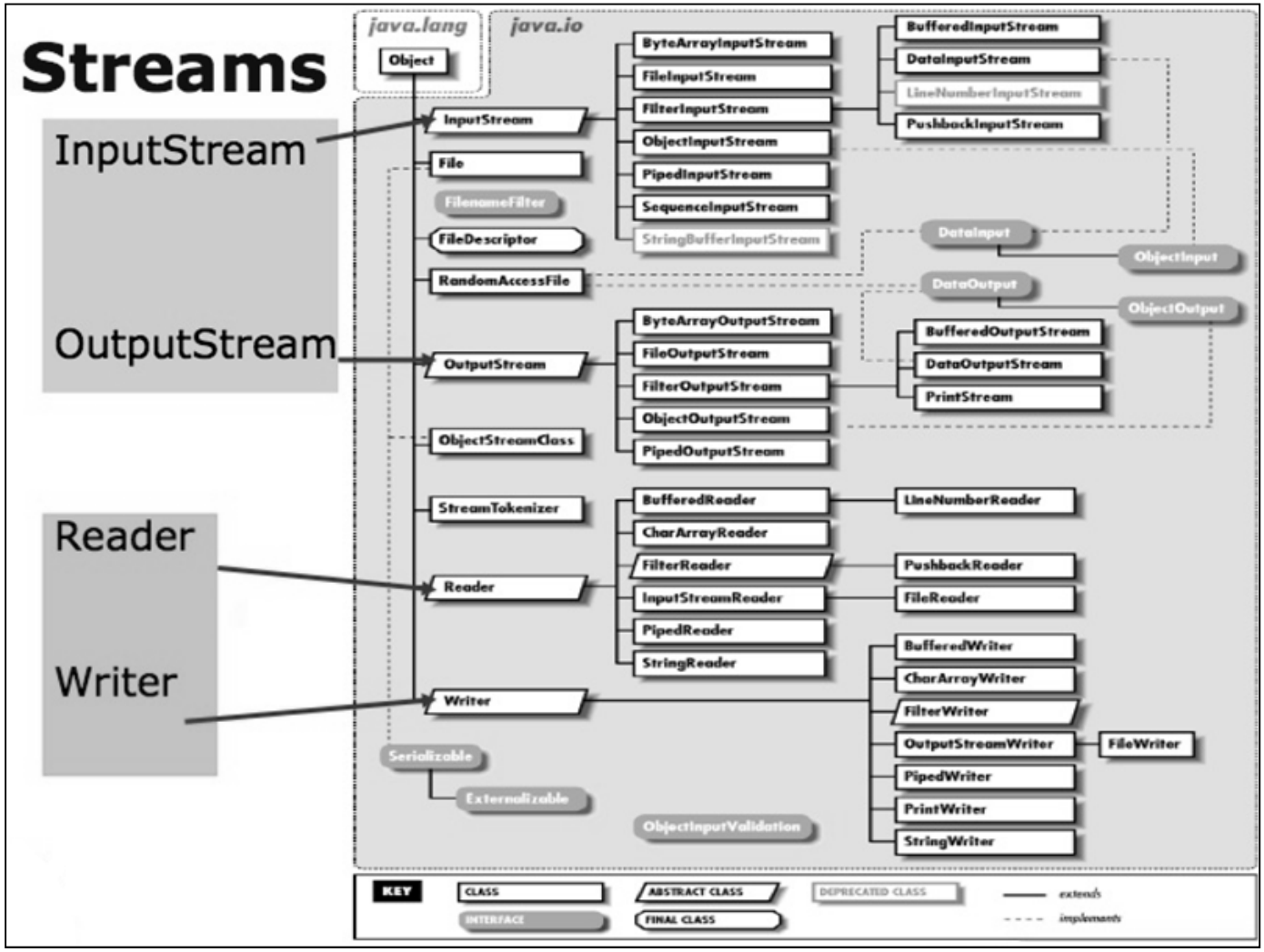
“Stream”



“Stream”

- Dos clases “especiales” principales que se usan para representar flujos de datos
  - InputStream
    - Es cualquier *fuente* desde el que se reciben datos (ficheros, socket de red, otro programa…)
  - OutputStream
    - Es cualquier *destino* hacia el que se trasmiten datos (ficheros, socket de red, otro programa…)
- Todos trabajan con secuencias de datos
  - Algunos solo retrasmiten datos
  - Otros permiten manipular y transformar datos
  - Y otros incluso pueden interconectarse formando un *pipeline* de flujos de datos

Jerarquía de Streams en Java



Trasmitir por salida estándar

- **`System.out`**
  - Atributo público de la clase **`System`**, de tipo **`java.io.PrintStream`**
  - Tiene métodos para escribir los distintos tipos de datos
  - De cada uno tiene dos versiones
    - **`print`**
    - **`println // añade salto de línea al final`**

```
System.out.println("Cadena en la primera línea");
int dia = 26;
String mes = "octubre"
System.out.print("Hoy es ");
System.out.print(dia);
System.out.println(" de " + mes);

Cadena en la primera línea
Hoy es 26 de octubre
```

Formatear por salida estándar

- Similar a **`printf`** de C++ (desde Java 1.5)
- Clase **`java.util.Formatter`**
  - Esta clase se usa en los métodos **`printf`** y **`format`** de la clase **`PrintStream`**
  - Similar (pero no igual) a la usada en C
    - Después de **`%`** aparece el número de argumento, seguido de **`$`** con el tipo del mismo

```
Punto p1, p2;
…
System.out.printf("Segmento: (%1$d ,%2$d) - (%3$d
   ,%4$d)\n", p1.getX(), p1.getY(), p2.getX(), p2.getY());
```

Recibir por entrada estándar

- **`System.in`**
  - Atributo público de la clase **`System`**, de tipo **`java.io.InputStream`**
  - Lee bytes usando el método **`read`**
  - Puede generar excepciones **`java.io.IOException`**
  - Es algo incómodo para leer órdenes del usuario

```
int b = 0 ;
try {
 b = System.in.read() ;
} catch (java.io.IOException ex ) {
  System.out.println("Excepción capturada");
}
  System.out.println("Leído "+ (char)b);
```

Recibir por entrada estándar

- **`java.util.Scanner`** (desde Java 1.5)
  - Tiene métodos **`nextInt`**, **`nextLong`**, **`nextLine`**, etc.
  - También se puede preguntar si existen datos con **`hasNextInt`**, **`hasNextLong`**, **`hasNextLine`**, etc.

```
java.util.Scanner sc  = new java.util.Scanner(System.in);
int dia; int mes; long anyo;
System.out.println("Introduzca la fecha:");
System.out.print("Dia: ");
dia = sc.nextInt();
System.out.print("Mes: ");
mes= sc.nextInt();
System.out.print("Año: ");
anyo = sc.nextLong();
```

Críticas, dudas, sugerencias…

Federico Peinado
www.federicopeinado.es