Universidad Europea

UF3: Programación básica

3.1: Fundamentos del lenguaje Java

Ve más allá

Índice

Clases

Primera aproximación

Las clases son el **elemento base** para programar en java.

Las clases contendrán el código en java encargado de ejecutar algo. Este código estará formado por **variables**, **constantes** y **métodos**.

La **JVM** es la encargada de ejecutar una clase, pero no todas las clases son ejecutables. Una clase será ejecutable si contiene el **método main**.

De momento solo vamos a crear clases ejecutables.

Clases

Primera aproximación

```
package nombrepaquete;

public class NombreClase {

    public static void main(String[] args) {
        // TODO implementar código

    }

}
```

Palabras clave
Comentarios
TODO
Variables
Nombres (paquetes, clases, variables, métodos)

Clases

Ficheros

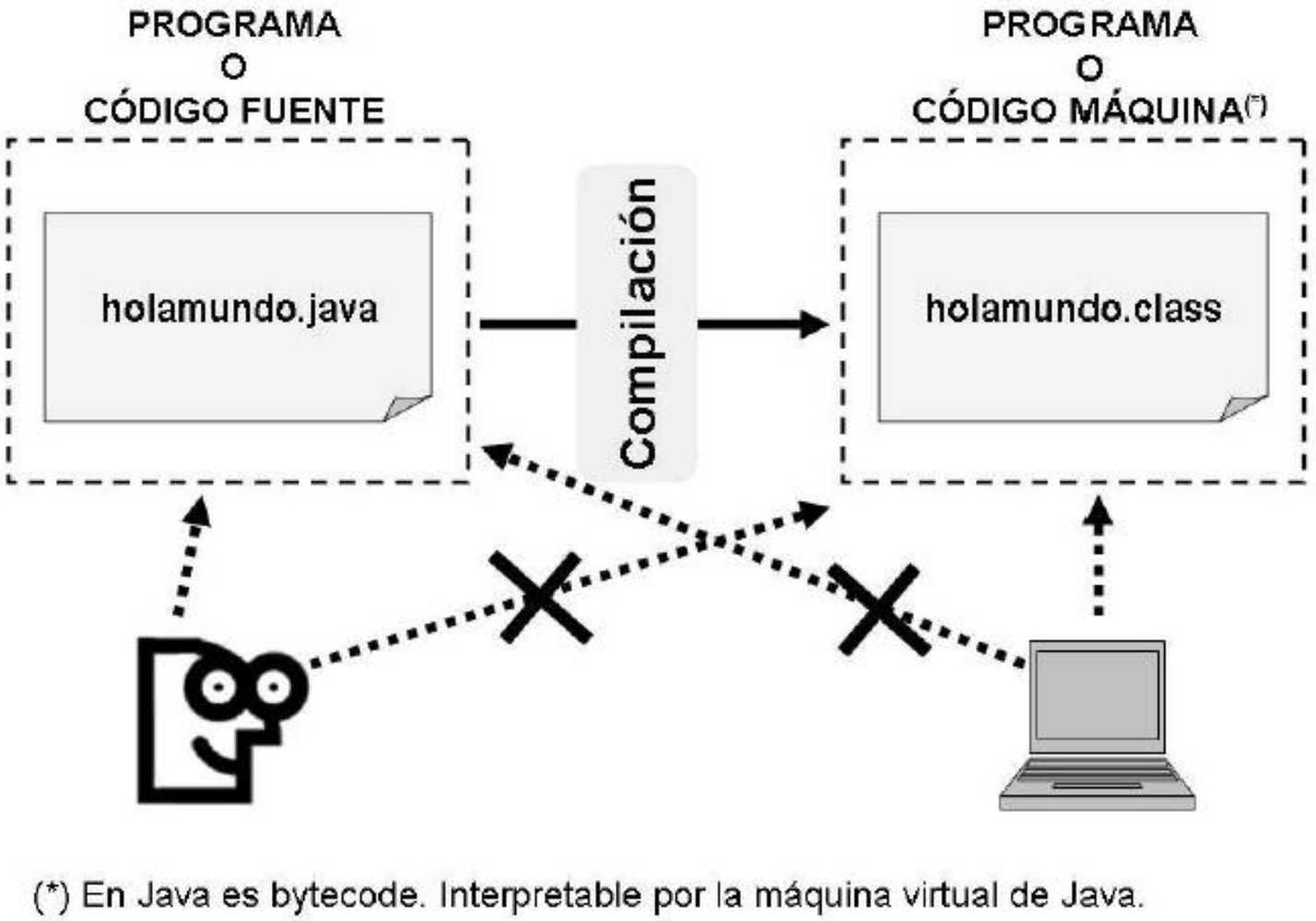
Al crear una clase con Eclipse, lo que se crea es un fichero con extensión **.java** (en la carpeta **src**).

Automáticamente se crea también, si no hay errores en el código que impidan la compilación, el fichero con extensión **.class (bin)**, que contiene los **bytecodes**, o código compilado, correspondientes al código de la clase que hayamos creado.

En un fichero .java puede haber más de una clase pero, por el momento, nosotros vamos a crear ficheros con una sola clase.

Clases

¿Recuerdas?



(*) En Java es bytecode. Interpretable por la máquina virtual de Java.

Comentarios

Tipos de comentarios

Java permite tres tipos de comentarios, utilizando los símbolos:

- // **(doble barra)**. Permite comentar una sola línea.
- **/* y */**. Se usan para comentarios de más de una línea.
- **/** y */**. Utilizados para la documentación de javadoc.

Este sistema permite documentar a partir del código fuente.

```
//comentario 1 línea

/* Este comentario ocupa
un par de líneas */

/**
 * Comentario JavaDoc
 * @author Sara V.
 *
 */
```

Sentencias e instrucciones

Las sentencias nos permiten realizar acciones en Java.

Ejemplos:

- `int entero = 1;`
- `String nombre = "Jaime";`
- `System.out.println("Hola " + nombre + ", ¿cómo estás?");`
- `suma = entero + 5;`

En Java todas las sentencias deben finalizar con **punto y coma** (;).

Variables

Es una zona de memoria con un nombre que la identifica donde se puede **almacenar información** del tipo que defina el programador.

Declaración de variable:

<tipoDato> <nombreVariable>;

```
int numero;
double temperatura;
```

Se pueden definir variables sin asignarles un valor (sin inicializar). Dependiendo del tipo de dato la variable tendrá un **valor por defecto.**

Las **variables locales** son las variables que se declaran dentro de un bloque de código se crean cuando el bloque se declara y se destruyen cuando finaliza la ejecución de dicho bloque.

Constantes

Se utilizan para almacenar datos que nunca varían (ej.- IVA, π...). Con esto nos aseguramos de que su valor no sea modificado nunca, y también localizamos el valor del dato en una sola línea de código.

Declaración de constante:

```
[static] final <tipo> <NOMBRE> = <valor>;
```

- **`static:`** si se incluye, implica que sólo hay una copia de esta constante en el programa.
- **`final:`** indica que es una constante
- Luego tenemos el tipo de datos, el nombre, y por último el valor que toma.

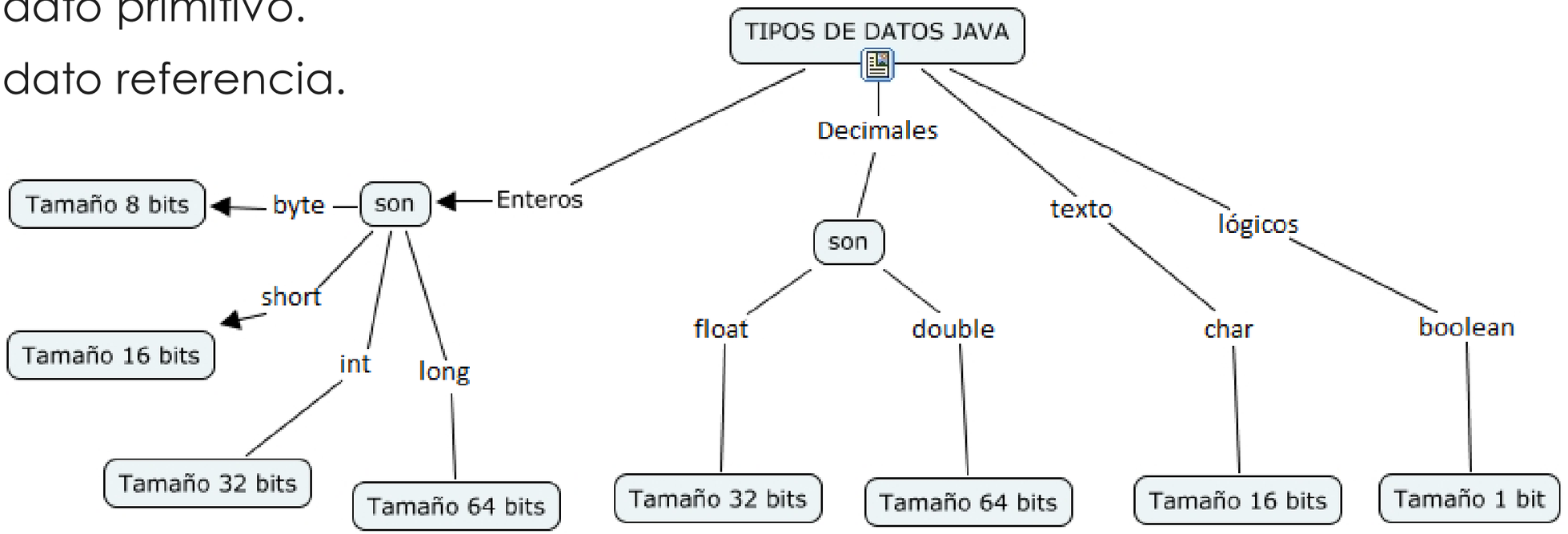
```
static final double PI = 3.141592;
```

Tipos de datos

Un tipo de dato indica qué valores puede contener esa variable. Las variables deben ser de un tipo de dato en concreto.

En Java, los tipos de datos pueden clasificarse en dos grupos:

- Tipo de dato primitivo.
- Tipo de dato referencia.



Tipos de datos

Datos primitivos

Tipo de datos	Información representada	Rango	Descripción
byte	Datos enteros	-128 ⟷ +127	Se utilizan 8 bits (1 byte) para almacenar el dato.
short	Datos enteros	-32768 ⟷ +32767	Dato de 16 bits de longitud (independientemente de la plataforma).
int	Datos enteros	-2147483648 ⟷ +2147483647	Dato de 32 bits de longitud (independientemente de la plataforma).
long	Datos enteros	-9223372036854775808 ⟷ +9223372036854775807	Dato de 64 bits de longitud (independientemente de la plataforma).
char	Datos enteros y caracteres	0 ⟷ 65535	Este rango es para representar números en unicode, los ASCII se representan con los valores del 0 al 127. ASCII es un subconjunto del juego de caracteres Unicode.
float	Datos en coma flotante de 32 bits	Precisión aproximada de 7 dígitos	Dato en coma flotante de 32 bits en formato IEEE 754 (1 bit de signo, 8 para el exponente y 24 para la mantisa).
double	Datos en coma flotante de 64 bits	Precisión aproximada de 16 dígitos	Dato en coma flotante de 64 bits en formato IEEE 754 (1 bit de signo, 11 para el exponente y 52 para la mantisa).
boolean	Valores booleanos	true/false	Utilizado para evaluar si el resultado de una expresión booleanas es verdadero (true) o falso(false).

Tipos de datos

Datos primitivos

Tipo de Dato	Tipo de Información	Espacio que ocupa	Valor por defecto
byte	Entero	1 byte (8bits)	0
short	Entero	2 bytes	0
int	Entero	4 bytes	0
long	Entero	8 bytes	0
char	Texto	2 bytes	'\u0000'
float	Decimal	4 bytes	0
double	Decimal	8 bytes	0
boolean	Lógico	1 bit	false

Tipos de datos

Datos de referencia

Además de los ocho tipos de datos primitivos, existen los tipos de datos referencia, que son tres:

- Arrays.
- Clases
- Interfaces.

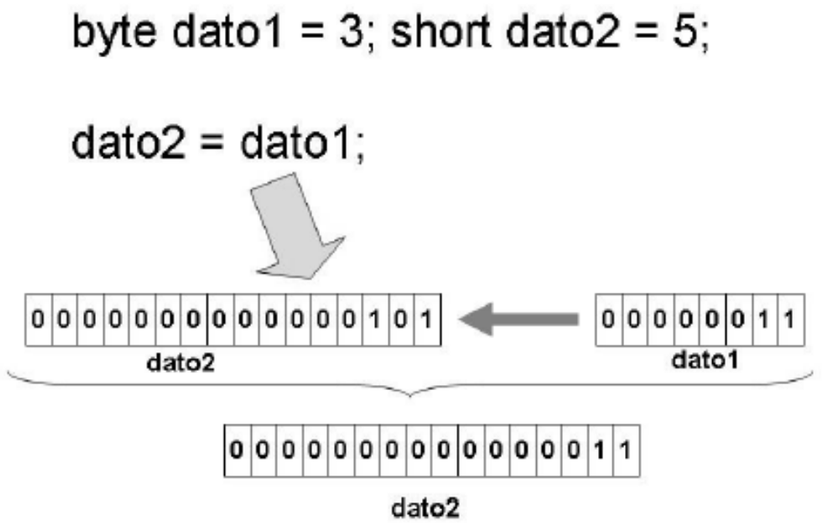
Los iremos viendo poco a poco.

Tipos de datos

Conversiones. Implícitas

Se realiza de forma automática entre dos tipos de datos diferentes.

Requiere que la variable destino (a la izquierda) tenga **más precisión** (pueda almacenar más bytes) que la variable origen (situada a la derecha).



Tipos de datos

Conversiones. Explícitas (cast)

En este caso es el programador el que fuerza la conversión mediante una operación llamada **cast** con el formato:

```
(tipo) expresión
```

Ejemplo:

```
int idato = 5;
byte bdato;
bdato = (byte) idato;
System.out.println(bdato);
// sacará 5 por pantalla
```

Operadores y expresiones

Aritméticos

Son los que se utilizan para operaciones matemáticas.

Operador	Uso	Operación
+	A + B	Suma
-	A - B	Resta
*	A * B	Multiplicación
/	A / B	División
%	A % B	Módulo o resto de una división entera

Ejemplo:

```
int n1=2, n2;
n2 = n1 * n1; // n2=4
n2 = n2-n1; // n2=2
n2 = n2+n1+15; // n2=19
n2 = n2/n1; // n2=9
n2 = n2%n1; // n2=1
```

Operadores y expresiones

Relacionales

Sirven para evaluar la igualdad y la magnitud:

Operador	Uso	Operación
<	A< B	A menor que B
>	A > B	A mayor que B
<=	A<= B	A menor o igual que B
>=	A>= B	A mayor o igual que B
!=	A!= B	A distinto que B
= =	A= = B	A igual que B

Ejemplo:

```
int m=2, n=5;
boolean res;
res = m > n;//res=false
res = m < n;//res=true
res = m >= n;//res=false
res = m <= n;//res=true
res = m == n;//res=false
res = m != n;//res=true
```

Operadores y expresiones

Lógicos

Sirven para realizar operaciones lógicas:

Operador	Uso	Operación
&& o &	A&& B o A&B	A AND B. El resultado será true si ambos operandos son true y false en caso contrario.
|| o |	A || B o A | B	A OR B. El resultado será false si ambos operandos son false y true en caso contrario.
!	!A	Not A. Si el operando es true el resultado es false y si el operando es false el resultado es true.
^	A ^ B	A XOR B. El resultado será true si un operando es true y el otro false, y false en caso contrario.

```
int m=2, n=5;
boolean res;
res =m > n && m >= n; //res=false
res =!(m < n || m != n); //res=false
```

Operadores y expresiones

Unarios o Unitarios

Son los que sólo requieren un operador para funcionar:

Operador	Uso	Operación
~	~A	Complemento a 1 de A
-	-A	Cambio de signo del operando
--	A--	Decremento de A
++	A++	Incremento de A
!	! A	Not A (ya visto)

Ejemplo:

```
int m=2, n=5;
m++; // m=3
n--; // n=4
```

Operadores y expresiones

Operadores de asignación

Sirven para asignar valores a las variables:

Operador	Uso	Operación
=	A = B	Asignación. Operador ya visto.
*=	A *= B	Multiplicación y asignación. La operación A*=B equivale a A=A*B.
/=	A /= B	División y asignación. La operación A/=B equivale a A=A/B.
%=	A %= B	Módulo y asignación. La operación A%=B equivale a A=A%B.
+=	A += B	Suma y asignación. La operación A+=B equivale a A=A+B.
-=	A -= B	Resta y asignación. La operación A-=B equivale a A=A-B.

Ejemplo:

```
int num=5;
num += 3; // num = 8
//equivale a num = num + 3
```

Operadores y expresiones

Operador condicional ternario

La sintaxis y significado del operador condicional ternario es la siguiente:

```
variable = expresiónCondicional? valor1 : valor2;
```

Interpretación: si la expresiónCondicional es cierta, la variable toma el valor1, y en caso contrario toma el valor2.

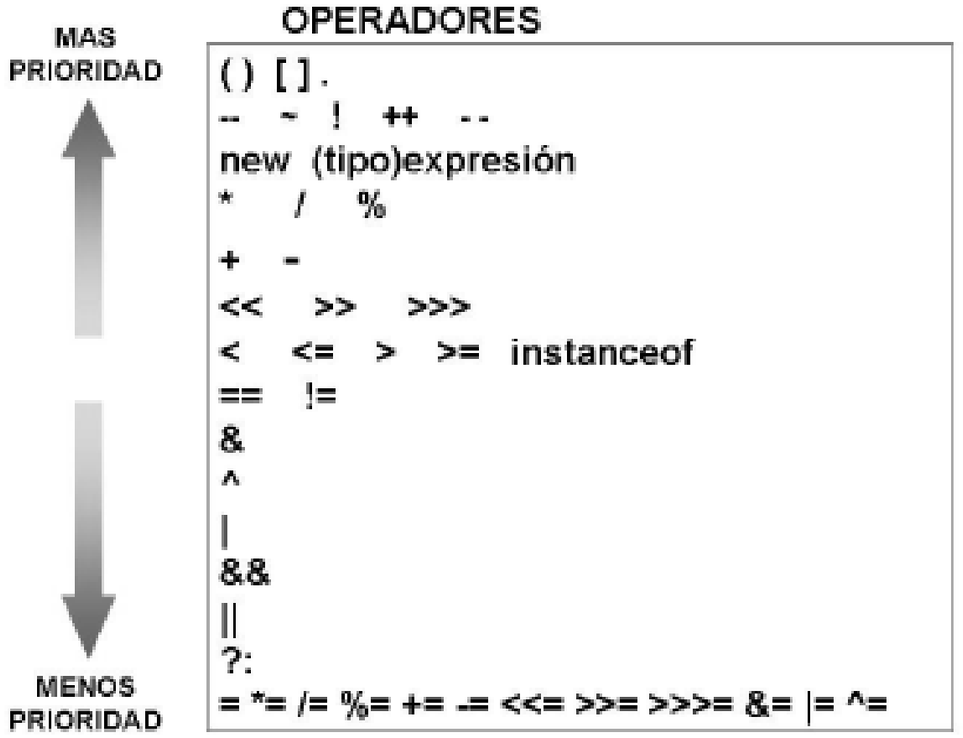
Ejemplo:

```
int x=3;
int y=5;
mayor = (x>y)? x : y; //mayor = 5;
```

Operadores y expresiones

Prioridad de los operadores

La precedencia o prioridad de operadores es la siguiente:



Consejo:

Utiliza **paréntesis**. Los programas quedan más legibles y el flujo de ejecución se controla más fácilmente que si dependemos de la prioridad.

Convenciones Java

¿Por qué las necesitamos?

- El 80% del coste del tiempo de vida de un software se va en **mantenimiento**.
- Es muy raro que un software lo mantenga durante toda su vida su autor original.
- Las convenciones de nombrado mejoran la **lectura del código**, permitiendo a los desarrolladores entender el nuevo código más rápidamente y mejor.
- Si lanzamos nuestro código fuente como un producto, necesitamos asegurarnos de que está tan **bien empaquetado y limpio** como cualquier otro producto que creemos.
- Para que las convenciones funcionen, todos aquellos que escriban software debe adherirse a ellas.

Convenciones Java

Aquí tenéis una recopilación de todas las convenciones de Java. Vamos a ver sólo unas cuantas.

Indentación:

- Como norma general se usarán **cuatro espacios**.
- Las líneas de código no deben superar **80 caracteres.**
- Las líneas de comentarios no deben superar **70 caracteres**.
- Cuando una expresión no cabe en una sola línea: romper después de una coma, romper antes de un operador, alinear la nueva línea al principio de la anterior.

Convenciones Java

Comentarios

Los comentarios deben contener solo la información que es relevante para la lectura y la comprensión del programa. Existen dos tipos de comentarios: de documentación (JavaDoc) y de implementación.

Declaraciones

- Se recomienda declarar **una variable por línea**.
- **Inicializar** las variables locales donde están declaradas y colocarlas **al comienzo del bloque**.

Convenciones Java

Nombres

Los nombres de las variables, métodos, clases, etc, hacen que los programas sean más fáciles de leer ya que pueden darnos información acerca de su función.

- **Variables**: Deben ser cortas (pueden ser de una letra) y *significativas*. Si son varias palabras la primera debe estar en minúscula. Ejemplos: `i, j, sumaTotal.`
- **Constantes**: El nombre debe ser *descriptivo*. Se escriben en **mayúsculas** y si son varias palabras van unidas por un carácter de subrayado. Ejemplo: `MAX_VALOR`.