UF3.1
Fundamentos del lenguaje Java

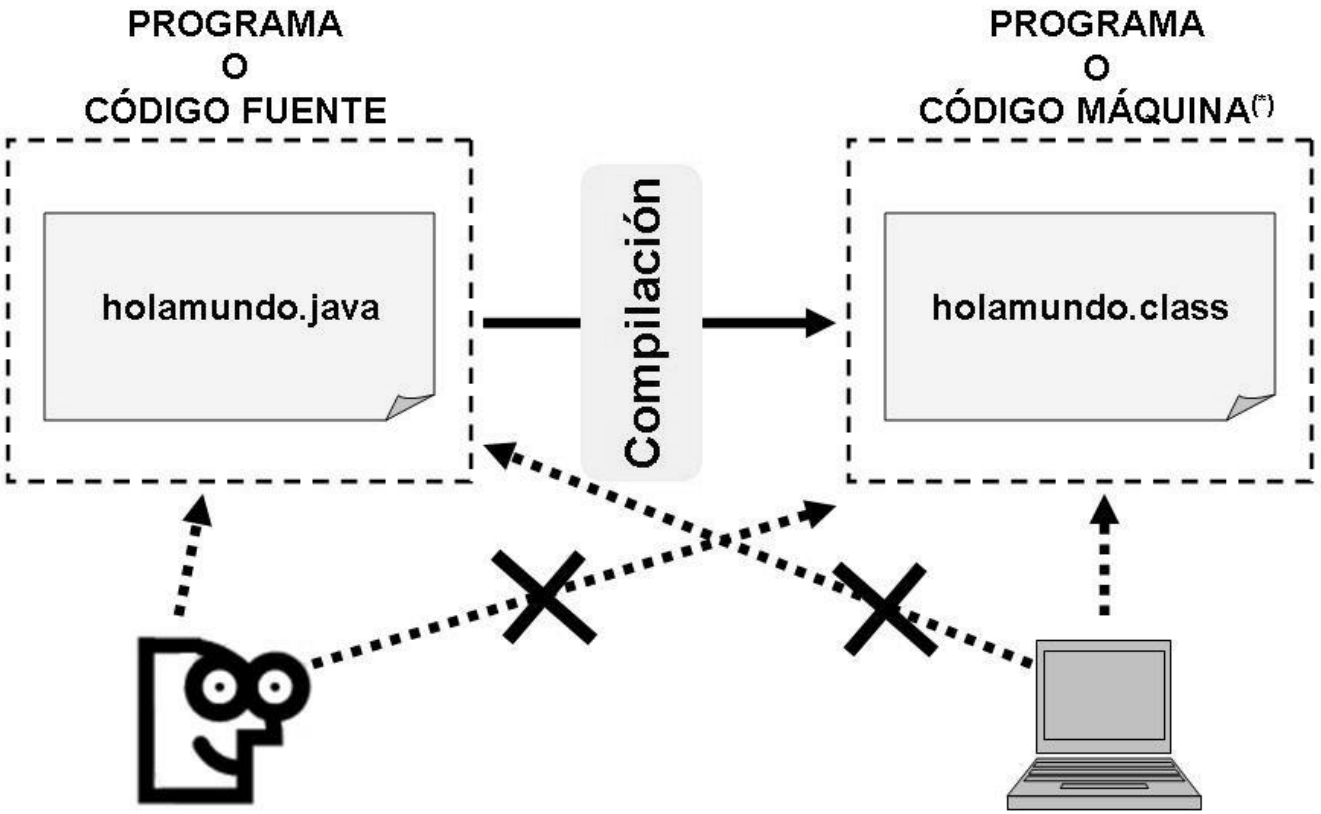
Centro Profesional
Universidad Europea Madrid
LAUREATE INTERNATIONAL UNIVERSITIES

CONTENIDOS

1. Recordando y JDK
2. Comentarios
3. Sentencias
4. Variables y constantes
5. Tipos de datos y conversiones
6. Operadores y expresiones
7. Convenciones

RECORDANDO…

…tu primer programa



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

JDK

Java Development Kit

- El JDK (Java Development Kit), aunque no contiene ninguna herramienta gráfica para el desarrollo de programas, sí que contiene aplicaciones de consola y herramientas de compilación, documentación y depuración. El JDK incluye el JRE (Java Runtime Environment) que consta de los mínimos componentes necesarios para ejecutar una aplicación Java, como son la máquina virtual y las librerías de clases.
- El JDK contiene, entre otras, las siguientes herramientas de consola:
  - javann. Es la máquina virtual de Java.
  - javacnn. Es el compilador de Java. Con él es posible compilar las clases que desarrollemos.
  - javapnn. Es un desensamblador de clases.
  - jdbnn. El depurador de consola de Java
  - javadocnn. Es el generador de documentación.
  - appletviewernn. Visor de Applets.
- JDK también se denomina SDK (Standard Development Kit) o incluso J2SE (Java 2 platform Standard Edition).

- **Pruébalo:** Para conocer la versión de java con la que estamos trabajando basta con ejecutar lo siguiente en una shell o intérprete de comandos: java –version

COMENTARIOS

Tres formas

Java permite tres tipos de comentarios, utilizando los símbolos:

// (doble barra). Permite comentar una sola línea.

Ejemplo: // este es un comentario de una línea

/* y */. Se usan para comentarios de más de una línea.

Ejemplo: /* Este comentario ocupa
un par de líneas */

/** y */. Utilizados para la documentación de javadoc.
Este sistema permite documentar a partir del código fuente.

```
/**
 * Simple HelloButton() method.
 * @version 1.0
 * @author john doe <doe.j@example.com>
 */
HelloButton()
{
  JButton hello = new JButton( "Hello, wor
  hello.addActionListener( new HelloBtnList

  // use the JFrame type until support for t
  // new component is finished
  JFrame frame = new JFrame( "Hello Button"
  Container pane = frame.getContentPane();
  pane.add( hello );
  frame.pack();
  frame.show();          // display the fra
}
```

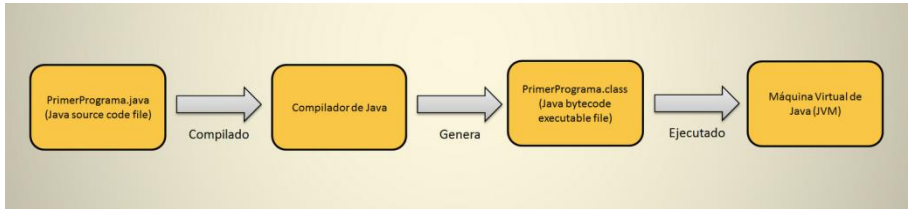
SENTENCIAS

Instrucciones

Las sentencias nos permiten realizar acciones en Java. Ejemplos de sentencias en Java son:

```
int entero = 1;
System.out.println(“Hola ” + nombre + “, ¿cómo estás? “);
import java.lang;
suma = entero + 5;
```

Fijaos que todas las sentencias deben finalizar con el símbolo punto y coma (;).
También es posible convertir un conjunto de sentencias como si se tratase de una sola, encerrándolas entre llaves ({ y }). Es útil, por ejemplo, para el uso de sentencias condicionales o bucles.



VARIABLES

Definición

- Una variable no es más ni menos que una zona de memoria donde se puede almacenar información del tipo que desee el programador.
- Para la declaración de variables se debe introducir el tipo y a continuación el nombre de la variable, como por ejemplo:

```
class suma {
          static int n1=50; // variable miembro de la clase
          public static void main(String [] args) {
                    int n2=30, suma=0; // variables locales
                    suma=n1+n2;
                    System.out.println("LA SUMA ES: " + suma);
          }
}
```

- Como puede verse en el ejemplo anterior, las variables se declaran dentro de un bloque (por bloque se entiende el contenido entre las llaves { }) y son accesibles solo dentro de ese bloque.
- Las variables declaradas en el bloque de la clase como n1 se consideran miembros de la clase, mientras que las variables n2 y suma pertenecen al método main y solo pueden ser utilizados en el mismo. Las variables declaradas en el bloque de código de un método son variables que se crean cuando el bloque se declara, y se destruyen cuando finaliza la ejecución de dicho bloque.

CONSTANTES

Definición

- Las constantes se declaran siguiendo el siguiente formato:

```
final [static] <tipo de datos> <nombre de la constante> = <valor>;
```

- Donde el calificador final identificará que es una constante, la palabra static si se declara implicará que solo existirá una copia de dicha constante en el programa aunque se declare varias veces, el tipo de datos de la constante seguido del nombre y por último el valor que toma. Ejemplo:

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

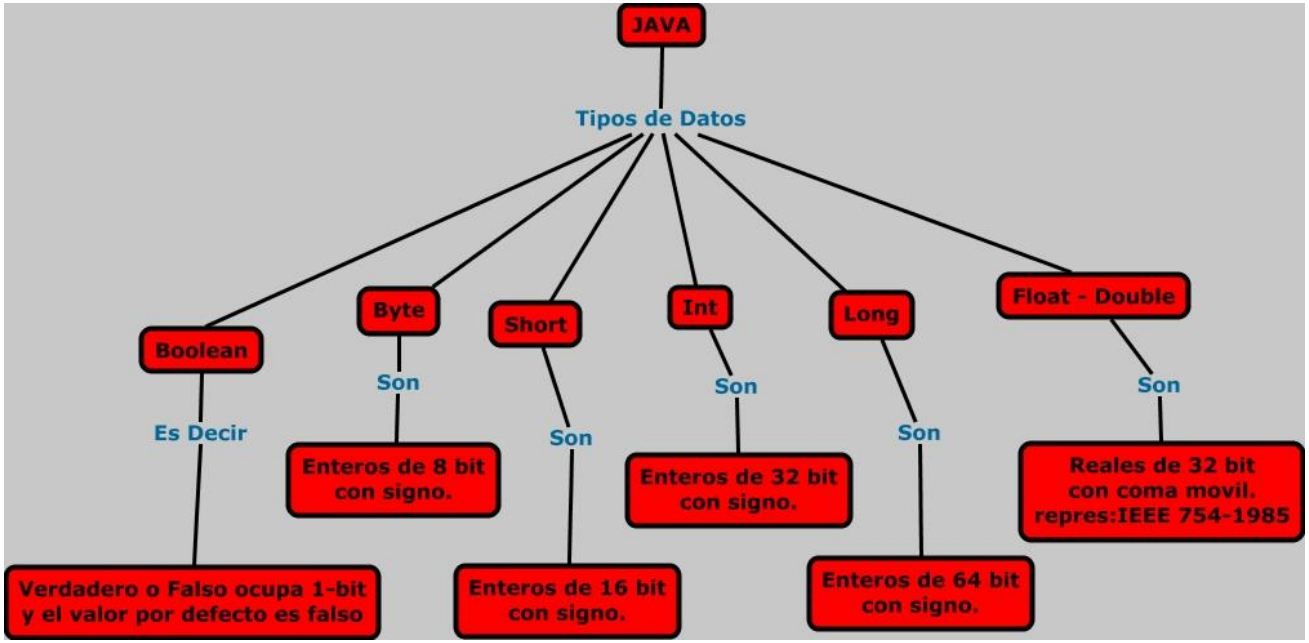
- **Importante**: Las constantes se utilizan en datos que nunca varían (IVA, PI, etc.). Utilizando constantes y no variables nos aseguramos que su valor no va a poder ser modificado nunca. También utilizar constantes permite centralizar el valor de un dato en una sola línea de código (si se quiere cambiar el valor del IVA se hará solamente en una línea en vez de si se utilizase el literal 18 en muchas partes del programa).

TIPOS DE DATOS

Datos Primitivos

Las variables deben ser de un tipo de dato en concreto. Un tipo de dato indica qué valores puede contener esa variable. En Java, los tipos de datos pueden clasificarse en dos grupos:

Tipo de dato primitivo.
Tipo de dato referencia.



TIPOS DE DATOS

Datos Primitivos

Los tipos de datos primitivos son ocho y pueden almacenar los valores que especificamos en la siguiente tabla:

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

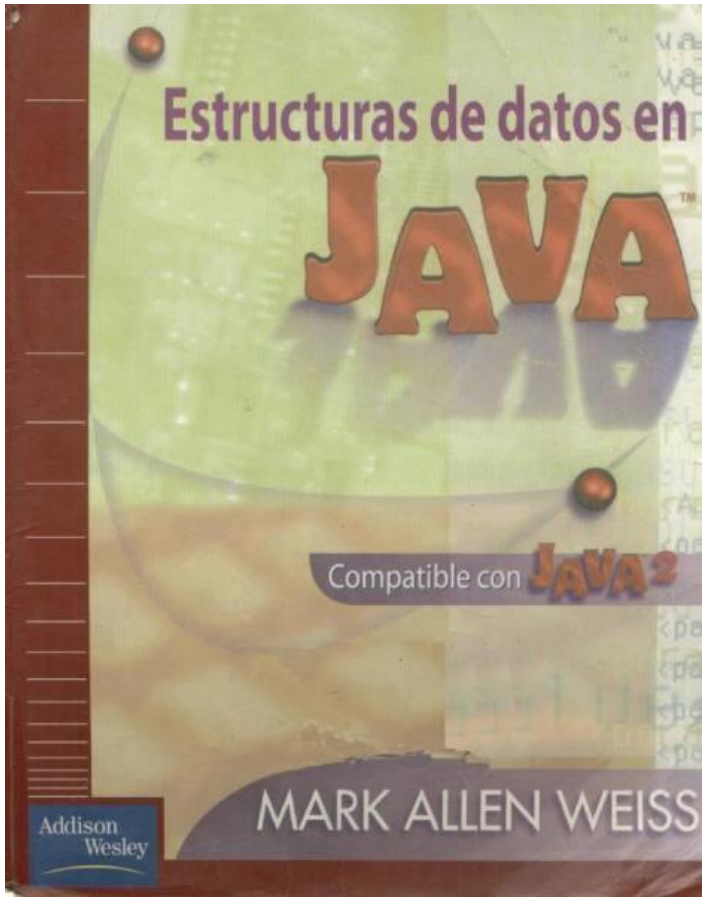
Utilización de los tipos

A continuación, se muestran ejemplos de utilización de tipos de datos en la declaración de variables.

Tipo de dato	Código
byte	byte a;
short	short b, c=3;
int	int d = -30; int e = 0xC125;
long	long b=434123 ; long b=5L ; /* la L en este caso indica Long*/
char	char car1='c'; char car2=99; /*car1 y car2 son lo mismo porque el 99 en decimal es la 'c' */
float	float pi=3.1416; float pi=3.1416F; /* la F en este caso indica Float*/ float medio=1/2F; /*0.5*/
double	double millón=1e6; /* 1x106 */ double medio1/2D; /*0.5 la D en este caso indica Double*/

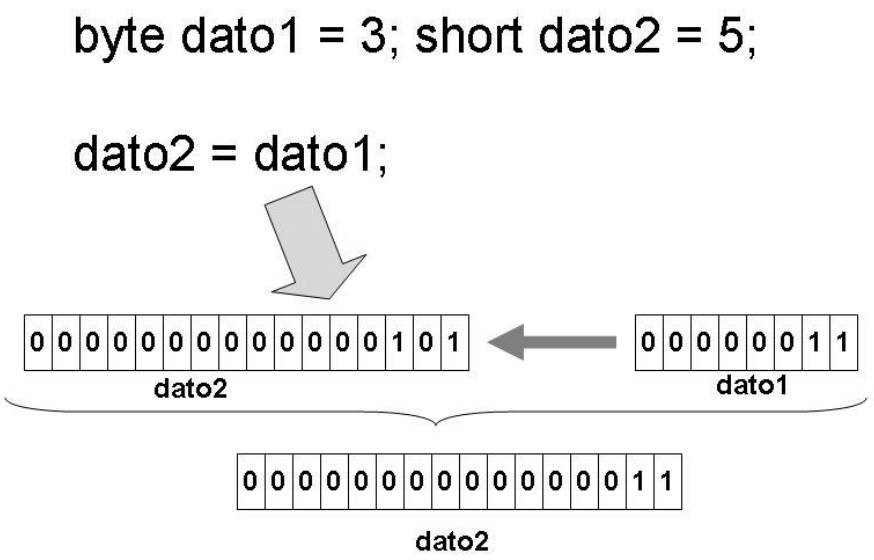
Datos de referencia

Además de los ocho tipos de datos primitivos, existen los tipos de datos referencia, que son tres: vectores, clases e interfaces.



Conversiones implícitas

Se realiza de forma automática entre dos tipos de datos diferentes. Requiere que la variable destino (la colocada a la izquierda) tenga más precisión que la variable origen (situada a la derecha).



CONVERSIONES DE TIPOS

Conversiones explícitas (cast)

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

```
(tipo) expresión
```

- Como puede ser comprensible no se pueden realizar conversiones entre enteros y booleanos o reales y booleanos.
- Intenta evitar las conversiones de tipos en la medida de lo posible. En algunas conversiones explícitas como ya supondrás pueden perder información en algunos casos.
- Un ejemplo de conversión explícita sería el siguiente:

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

OPERADORES Y EXPRESIONES

Operadores Aritméticos

Los operadores aritméticos son utilizados para realizar 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

En el siguiente ejemplo se puede observar la utilización de operadores aritméticos:

```
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 Relacionales

Con los operadores relacionales se puede 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

En el siguiente ejemplo se puede observar la utilización de operadores relacionales:

```
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 Lógicos

Con los operadores lógicos se pueden 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.

En el siguiente ejemplo se puede observar la utilización de operadores lógicos:

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

Operadores Unarios o Unitarios

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)

En el siguiente ejemplo se puede observar la utilización de operadores unitarios:

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

OPERADORES Y EXPRESIONES

Operadores de Asignación

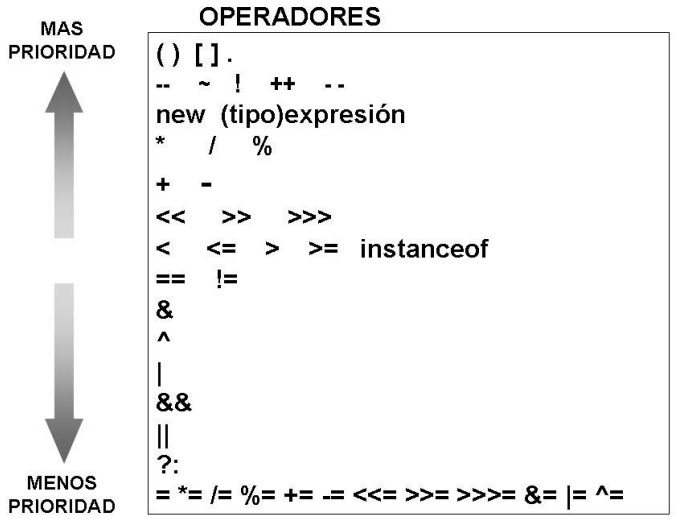
Operador	Uso	Operación
=	A = B	Asignación. Operador ya visto.
*=	A *= B	Multiplicación y asignación. La operación A*=B equivale a A=A*B.
/=	A /= B	División y asignación. La operación A/=B equivale a A=A/B.
%=	A %= B	Módulo y asignación. La operación A%=B equivale a A=A%B.
+=	A += B	Suma y asignación.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

En el siguiente ejemplo se puede observar la utilización de operadores de asignación:

```
int num=5;
num += 5;            // num = 10, equivale a num = num + 5
```

Precedencia de Operadores

La precedencia de operadores se resume en la siguiente tabla:



Consejo: Utiliza paréntesis y de esa forma puedes dejar los programas más legibles y controlar las operaciones sin tener que depender de la precedencia.

Ejemplo:

```
int a = 4;
a = 5 * a + 3;
```

Se desea conocer el valor que tomará a. Para ello se mira en la tabla y se puede observar que el operador * tiene más precedencia que el operador +, con lo cual primero se ejecutará 5 * a, y al resultado de esta operación se le sumará 3. El resultado de la expresión será 23 y por lo tanto el valor de a será 23 al ejecutar este código.

CONVENCIONES

Java

Las convenciones de codificación son importantes para los programadores por varias razones:

- El 80% del coste del tiempo de vida de una pieza de software se va en mantenimiento.
- Casi nunca ningún software es mantenido durante toda su vida por su autor original.
- Las convenciones de nombrado mejoran la lectura del software, permitiendo a los ingenieros 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 ella. Todos.

CONVENCIONES

Un documento