

# DESARROLLO DE APLICACIONES WEB

## Unidad 3

### V a r i a b l e s   y   t i p o s   d e d a t o s

*1r DAW*

*IES La Mola de Novelda*

*Departament d'informàtica*

# ÍNDIX

1.- Definición.....	3
2.- Identificador de variable.....	3
3.- Palabras reservadas.....	3
4.- Declaración de variables.....	4
5.- Constantes.....	4
6.- Tipos de datos primitivos.....	5
6.1.- Números enteros.....	5
6.2.- Números en coma flotante.....	6
6.3.- Booleanos.....	7
6.4.- Caracteres.....	7
7.- Operadores.....	8
7.1.- Operadores aritméticos.....	8
7.2.- Operadores relacionales.....	8
7.3.- Operadores lógicos.....	8
7.4.- Operador de concatenación.....	9
7.5.- Precedencia de operadores.....	9
7.6.- Operador de asignación.....	10
8.- Cadenas de caracteres.....	10
9.- Clase Math.....	11

## Unidad 3: VARIABLES Y TIPOS DE DATOS

### 1.- DEFINICIÓN

Las variables son los contenedores de los datos que utiliza un programa. Cada variable ocupa un espacio en la memoria principal (**volátil**) del ordenador para almacenar un dato determinado. El contenido de las variables puede variar durante la ejecución del programa

Todas las variables tienen un nombre o **identificador**. Este nombre tiene que ser **significativo**. Por ejemplo, el siguiente código es difícil de conocer que hace:

$$x = vlp3 * Weeety - zxc$$

En cambio, el siguiente código es entendible:

$$\text{precioTotal} = \text{cantidad} * \text{precio} - \text{descuento}$$

### 2.- IDENTIFICADOR DE VARIABLE

Al definir un identificador de variables tienes que tener en cuenta los siguientes puntos:

- Los identificadores de las variables se escribirán en formato **lowerCamelCase**. Ejemplo:

**edadMinimaCarnet**

Adoptamos esta forma de crear identificadores de variables, pero no es la única

- **No** se permiten los **símbolos especiales** como \$, %, @, +, -, etc.
- Se pueden usar **números** en los identificadores pero **nunca al principio** de éste.
- No ponemos letra mayúscula al principio del identificador para no confundirlo con una clase.
- Java es una lenguaje fuertemente tipado, es decir, es necesario declarar todas las variables que utilizará el programa, indicando siempre el nombre o identificador y su tipo.

### 3.- PALABRAS RESERVADAS

En el lenguaje de programación Java se puede hacer uso de las palabras clave (*keywords*), también llamadas palabras reservadas, mostradas en la siguiente tabla. Dichas palabras, no pueden ser utilizadas como identificadores por los programadores para definir variables, constantes, etc

<b>abstract</b>	<b>default</b>	<b>if</b>	<b>private</b>	<b>this</b>
<b>boolean</b>	<b>do</b>	<b>implements</b>	<b>protected</b>	<b>throw</b>
<b>break</b>	<b>double</b>	<b>import</b>	<b>public</b>	<b>throws</b>
<b>byte</b>	<b>else</b>	<b>instanceof</b>	<b>return</b>	<b>transient</b>
<b>case</b>	<b>extends</b>	<b>int</b>	<b>short</b>	<b>try</b>
<b>catch</b>	<b>final</b>	<b>interface</b>	<b>static</b>	<b>void</b>
<b>char</b>	<b>finally</b>	<b>long</b>	<b>strictfp</b>	<b>volatile</b>
<b>class</b>	<b>float</b>	<b>native</b>	<b>super</b>	<b>while</b>
<b>const</b>	<b>for</b>	<b>new</b>	<b>switch</b>	<b>assert</b>
<b>continue</b>	<b>goto</b>	<b>package</b>	<b>synchronized</b>	<b>enum</b>

*true*, *false* y *null* no son considerados palabras clave de Java, sino literales. Ahora bien, tampoco se pueden utilizar como identificadores.

La descripción de la funcionalidad de todas las palabras reservadas se puede encontrar en:

<https://www.abrirllave.com/java/palabras-clave.php>

## 4.- DECLARACIÓN DE VARIABLES

Como hemos dicho anteriormente, antes de poder utilizar una variable, ésta se debe declarar. La declaración de una variable sigue la siguiente regla:

**tipo**      *nombreVariable*;

Ejemplos:

**int**      *día*;

**bool**      *decision*;

También podemos darle un valor inicial cuando declaramos una variable. Por ejemplo:

**int**      *día* = 365;

## 5.- CONSTANTES

Variables cuyo valor no cambia a lo largo de todo el programa.

```
public class Prova {
    public static void main(String args[]){
        final double PI = 3.1415926536;
        PI = 2.3;
    }
}
```

```
lliurex@lliurex-ttlportatil:~/Música$ javac Prova.java
Prova.java:6: error: cannot assign a value to final variable PI
    PI = 2.3;
    ^
1 error
lliurex@lliurex-ttlportatil:~/Música$
```

## 6.- TIPOS DE DATOS PRIMITIVOS

Tipo	Descripción	Bytes	Rango	Valor por defecto
<b>byte</b>	Entero muy corto	1	-128 a 127	0
<b>short</b>	Entero corto	2	-32.768 a 32.767	0
<b>int</b>	Entero	4	-2.147.486.648 a 2.147.486.647 ( $-2^{31}$ a $2^{31}-1$ )	0
<b>long</b>	Entero largo	8	-9.223.372.036.854.775.808 a 9.223.372.036.854.775.807	0L
<b>float</b>	Número con punto flotante de precisión individual con hasta 7 dígitos significativos	4	+/-1.4E-45 (+/-1.4 times $10^{-45}$ ) a +/-3.4E38 (+/-3.4 times $10^{38}$ )	0.0f
<b>double</b>	Número con punto flotante de precisión doble con hasta 16 dígitos significativos	8	+/-4.9E-324 (+/-4.9 times $10^{-324}$ ) a +/-1.7E308 (+/-1.7 times $10^{308}$ )	0.0d
<b>char</b>	Carácter Unicode	\u0000 a \uFFFF	'\u0000'	
<b>boolean</b>	Valor verdadero o false	1	true o false	false

### 6.1.-NÚMEROS ENTEROS

Los tipos *byte*, *short*, *int* y *long* sirven para almacenar datos enteros. Para almacenar enteros en formato **octal** pondremos un **0** delante i para enteros hexadecimal anteponeamos **0x**.

```

public class Prova {
    public static void main(String args[]){
        int x=3;
        int y=020;
        int z=0x14;
        System.out.println("Enter: "+x);
        System.out.println("Octal: "+y);
        System.out.println("Hexadecimal: "+z);
    }
}

```

```

lliurex@lliurex-ttlportatil:~/Música$ ls
Prova.java
lliurex@lliurex-ttlportatil:~/Música$ javac Prova.java
lliurex@lliurex-ttlportatil:~/Música$ java Prova
Enter: 3
Octal: 16
Hexadecimal: 20
lliurex@lliurex-ttlportatil:~/Música$

```

No se acepta en general asignar variables de distinto tipo. Sí se pueden asignar valores de variables enteras a variables enteras de un tipo superior (por ejemplo asignar un valor *int* a una variable *long*). Pero al revés no se puede:

```

public class Prova {
    public static void main(String args[]){
        int x=3;
        long y=20;
        y=x;
        x=y;
    }
}

```

```

lliurex@lliurex-ttlportatil:~/Música$ javac Prova.java
Prova.java:7: error: incompatible types: possible lossy conversion
    from long to int
        x=y;
        ^
1 error
lliurex@lliurex-ttlportatil:~/Música$

```

La solución es hacer un **cast** (casting – conversión entre tipos). Esta operación permite

convertir valores de un tipo a otro. Se usa así:

```
public class Prova {  
    public static void main(String args[]){  
        int x=3;  
        long y=20;  
        y=x;  
        x=(int)y;  
    }  
}
```

```
lliurex@lliurex-ttlportatil:~/Música$ javac Prova.java  
lliurex@lliurex-ttlportatil:~/Música$
```

Hay que tener en cuenta en estos casos que si el valor asignado sobrepasa el rango del elemento, el valor convertido no tendrá ningún sentido:

```
public class Prova {  
    public static void main(String args[]){  
        int x=1200;  
        byte y=20;  
        y=(byte)x;  
        System.out.println("Valor d'x: "+y);  
    }  
}
```

```
lliurex@lliurex-ttlportatil:~/Música$ javac Prova.java  
lliurex@lliurex-ttlportatil:~/Música$ java Prova  
Valor d'x: -80  
lliurex@lliurex-ttlportatil:~/Música$
```

## 6.2.-NÚMEROS EN COMA FLOTANTE

Los decimales se almacenan en los tipos *float* y *double*. Se les llama de coma flotante por como son almacenados por el ordenador. Los decimales no son almacenados de forma exacta por eso siempre hay un posible error. En los decimales de coma flotante se habla, por tanto de precisión. Es mucho más preciso el tipo *double* que el tipo *float*.

Con una **f** al final del número indicamos que es un **float**, y con un **d** al final del número indicamos que es un **double**.

```
public class Prova {  
    public static void main(String args[]){  
        double d1=12.01,d2=0.23d;  
        float f1=20, f2=2.3f;  
        System.out.println("Valors: "+d1+" "+d2+" "+f1+" "+f2);  
    }  
}
```

```
lliurex@lliurex-ttlportatil:~/Música$ javac Prova.java  
lliurex@lliurex-ttlportatil:~/Música$ java Prova  
Valors: 12.01 0.23 20.0 2.3  
lliurex@lliurex-ttlportatil:~/Música$
```

## 6.3.-BOOLEANOS

Los valores booleanos (o lógicos) sirven para indicar si algo es verdadero (*true*) o falso (*false*)

The first screenshot shows a Java class named `Prova` with a `main` method. It declares two boolean variables, `b` and `c`, with values `true` and `false` respectively. The `main` method prints the values of `b` and `c` using `System.out.println`. The terminal output shows the command `javac Prova.java` and `java Prova`, resulting in the output `Valors: true false`.

The second screenshot shows the same Java class, but with an additional variable `a` declared as `boolean a` without an initial value. The terminal output shows an error: `Prova.java:5: error: variable a might not have been initialized`.

## 6.4.-CARACTERES

Los valores de tipo carácter sirven para almacenar símbolos de escritura (en Java se puede almacenar cualquier código *Unicode*). Los valores *Unicode* son los que Java utiliza para los caracteres. Los caracteres siempre van entre comillas simples.

The first screenshot shows a Java class named `Prova` with a `main` method. It declares two character variables, `a` and `b`, with values `'a'` and `67` respectively. The `main` method prints the values of `a` and `b` using `System.out.println`. The terminal output shows the command `javac Prova.java` and `java Prova`, resulting in the output `Lletres: a C`.

También hay una serie de caracteres especiales que van precedidos por el símbolo `\`, son estos:

Secuencia de escape	Significado	Secuencia de escape	Significado
<code>\b</code>	Retroceso	<code>\r</code>	Retorno de carro
<code>\t</code>	Tabulador	<code>\"</code>	Carácter comillas dobles
<code>\n</code>	Salto de línea	<code>'</code>	Carácter comillas simples
<code>\f</code>	Salto de página	<code>\\</code>	Barra diagonal

The first screenshot shows a Java class named `Prova` with a `main` method. It prints the string `"\HOLA\","caracola"\n"cotxe."` using `System.out.println`. The terminal output shows the command `javac Prova.java` and `java Prova`, resulting in the output `"HOLA",caracola` followed by a new line and `cotxe.`

## 7.- OPERADORES

### 7.1.-OPERADORES ARITMÉTICOS

Hay que tener en cuenta que el resultado de estos operadores varían notablemente si usamos enteros o si usamos números de coma flotante.

OPERADOR	NOMBRE	EJEMPLO	DESCRIPCIÓN
+	suma	20 + x	suma dos números
-	resta	a - b	resta dos números
*	multiplicación	10 * 7	multiplica dos números
/	división	altura / 2	divide dos números
%	resto (módulo)	5 % 2	resto de la división entera
++	incremento	a++	incrementa en 1 el valor de la variable
--	decremento	a--	decrementa en 1 el valor de la variable

### 7.2.-OPERADORES RELACIONALES

'>': Mayor que  
 '<': Menor que  
 '==': Iguales  
 '!=': Distintos  
 '>=': Mayor o igual que  
 '<=': Menor o igual que

Permiten comparar variables según relación de igualdad/desigualdad o relación mayor/menor. Devuelven siempre un valor *boolean*.

### 7.3.-OPERADORES LÓGICOS

Nos permiten construir expresiones lógicas

'&&': devuelve true si ambos operandos son true.

'||': devuelve true si alguno de los operandos son true.

'!': Niega el operando que se le pasa.

A	B	A && B	A    B	! A
true	true	true	true	false
true	false	false	true	false
false	true	false	true	true
false	false	false	false	true



```
public class Prova {

    public static void main(String args[]){

        boolean carnetConducir=true;
        int edad=20;
        boolean puedeConducir= (edad>=18) && carnetConducir;
        //Si la edad es de al menos 18 años y carnetConducir es
        //true, puedeConducir es true

        System.out.println("Puede conducir: "+puedeConducir);

    }

}
```

```
lliurex@lliurex-ttlportatil:~/Música$ javac Prova.java
lliurex@lliurex-ttlportatil:~/Música$ java Prova
Puede conducir: true
lliurex@lliurex-ttlportatil:~/Música$
```

```
public class Prova {

    public static void main(String args[]){

        boolean mayorDeEdad, menorDeEdad;
        int edad = 21;

        mayorDeEdad = edad >= 18; //mayorDeEdad será true
        menorDeEdad = !mayorDeEdad; //menorDeEdad será false

        System.out.println("Mayor de edad: "+mayorDeEdad+"\n"+"Menor de edad: "+menorDeEdad);

    }

}
```

```
lliurex@lliurex-ttlportatil:~/Música$ javac Prova.java
lliurex@lliurex-ttlportatil:~/Música$ java Prova
Mayor de edad: true
Menor de edad: false
lliurex@lliurex-ttlportatil:~/Música$
```

## 7.4.-OPERADOR DE CONCATENACIÓN

Operador de concatenación con cadena de caracteres '+': Lo hemos utilizado anteriormente.

## 7.5.-PRECEDENCIA DE OPERADORES

A veces hay expresiones con operadores que resultan confusas. Por ejemplo:

**resultado = 8 + 4 / 2;**

**resultado = (8 + 4) / 2;**

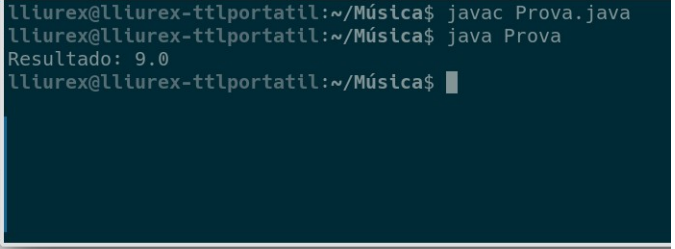
El orden de preferencia de los operadores en Java es:

operador			
<b>O</b>	<b>[]</b>	<b>.</b>	
<b>++</b>	<b>--</b>	<b>~</b>	<b>!</b>
<b>*</b>	<b>/</b>	<b>%</b>	
<b>+</b>	<b>-</b>		
<b>&gt;&gt;</b>	<b>&gt;&gt;&gt;</b>	<b>&lt;&lt;</b>	<b>&lt;&lt;&lt;</b>
<b>&gt;</b>	<b>&gt;=</b>	<b>&lt;</b>	<b>&lt;=</b>
<b>==</b>	<b>!=</b>		
<b>&amp;</b>			
<b>^</b>			
<b> </b>			
<b>&amp;&amp;</b>			
<b>  </b>			
<b>?:</b>			
<b>=</b>	<b>+=, -=, *=, ...</b>		

En la tabla anterior los operadores con mayor precedencia está en la parte superior, los de

menor precedencia en la parte inferior. De izquierda a derecha la precedencia es la misma. Es decir, tiene la misma precedencia el operador de suma que el de resta.

```
public class Prova {  
    public static void main(String args[]){  
        double resultado = 9 / 3 * 3;  
        System.out.println("Resultado: "+resultado);  
    }  
}
```



```
lliurex@lliurex-ttlportatil:~/Música$ javac Prova.java  
lliurex@lliurex-ttlportatil:~/Música$ java Prova  
Resultado: 9.0  
lliurex@lliurex-ttlportatil:~/Música$
```

El resultado podría ser uno o nueve. En este caso el resultado es nueve, porque la división y el producto tienen la misma precedencia; por ello el compilador de Java realiza primero la operación que este más a la izquierda, que en este caso es la división. Una vez más los paréntesis podrían evitar estos conflictos.

Cuando se tenga dudas, **la mejor manera de crear expresiones es poner paréntesis.**

## 7.6.-OPERADOR DE ASIGNACIÓN

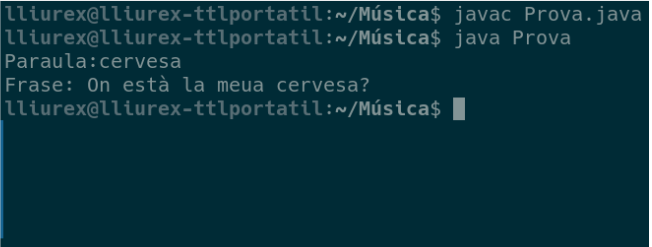
El operador de asignación es el carácter igual (=). Se utiliza para dar un valor a una variable. Hay que aclarar que una asignación no es una ecuación. Por ejemplo, **x = 7 + 1** es una asignación en la cual se evalúa la parte derecha (**7 + 1**), y el resultado de esa evaluación se almacena en la variable que hay a la izquierda del igual, es decir en la **x**. Por lo tanto la variable x contendrá el valor 8.

En cambio, la sentencia **x + 1 = 23 \* 2** no es una asignación válida, ya que en el lado izquierdo del igual sólo puede haber un nombre de variable.

## 8.- CADENAS DE CARACTERES

Las cadenas de caracteres se utilizan para almacenar palabras y frases. Todas las cadenas de caracteres van entre doble comillas.

```
public class Prova {  
    public static void main(String args[]){  
        String meuaParaula = "cervesa";  
        String meuaFrase = "On està la meua cervesa?";  
        System.out.println("Paraula:"+meuaParaula);  
        System.out.println("Frase: "+meuaFrase);  
    }  
}
```



```
lliurex@lliurex-ttlportatil:~/Música$ javac Prova.java  
lliurex@lliurex-ttlportatil:~/Música$ java Prova  
Paraula:cervesa  
Frase: On està la meua cervesa?  
lliurex@lliurex-ttlportatil:~/Música$
```

## 9.- CLASE MATH

Se echan de menos operadores matemáticos más potentes en Java. Por ello se ha incluido una clase especial llamada **Math** dentro del paquete **java.lang**. Esta clase posee métodos muy interesantes para realizar cálculos matemáticos complejos. Para poder utilizar esta clase, se debe incluir esta instrucción:

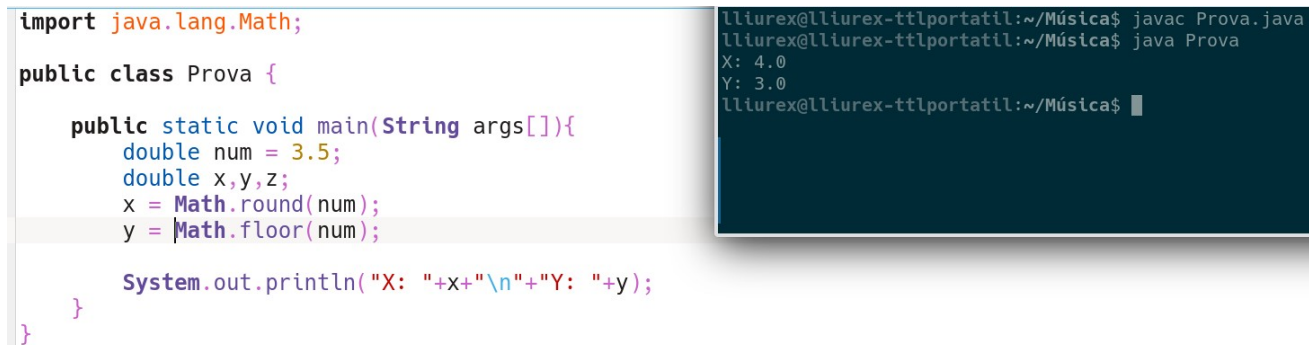
```
import java.lang.Math;
```

Con la palabra reservada **import** estamos indicando al compilador que incorpore un conjunto de librerías, tanto propias como externas o librerías propias del lenguaje Java. Éstas se indican en la ruta (**java.lang.Math**). Con **import** podemos agregar a nuestro proyecto una o varias clases (paquetes) según lo que necesitemos. De ellas solo podemos utilizar las funciones que sean públicas. Podemos ver los paquetes o librerías de Java en la siguiente web:

<https://docs.oracle.com/javase/7/docs/api/>

Ejemplo: El ejercicio realiza una conversión de decimal a entero por medio de las funciones que nos proporciona la clase **Math**.

- **Math.round(num)**: Redondeo al siguiente número entero.
- **Math.floor(num)**: Entero mayor, que sea inferior o igual a num.



```
import java.lang.Math;

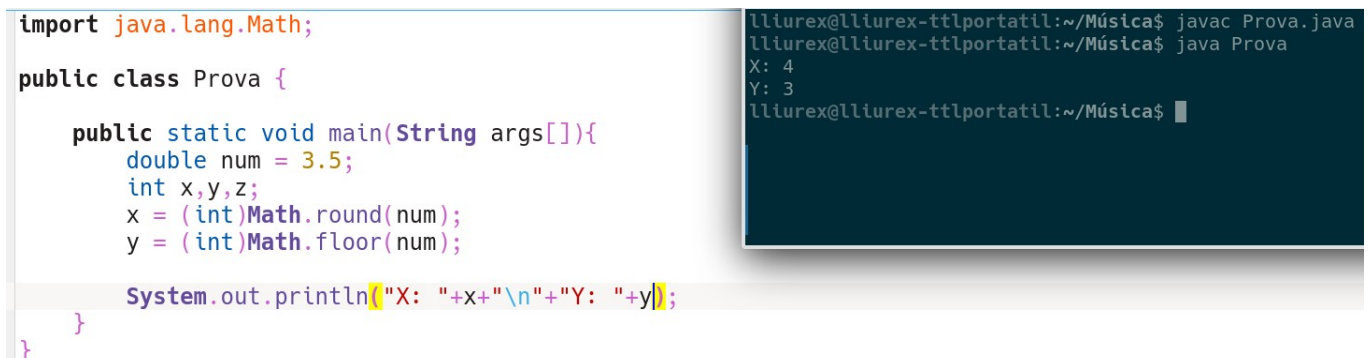
public class Prova {

    public static void main(String args[]){
        double num = 3.5;
        double x,y,z;
        x = Math.round(num);
        y = Math.floor(num);

        System.out.println("X: "+x+"\n"+"Y: "+y);
    }
}
```

```
lliurex@lliurex-ttlportatil:~/Música$ javac Prova.java
lliurex@lliurex-ttlportatil:~/Música$ java Prova
X: 4.0
Y: 3.0
lliurex@lliurex-ttlportatil:~/Música$
```

Y si queremos obtener el resultado en una variable del tipo entera, haremos un **casting**:



```
import java.lang.Math;

public class Prova {

    public static void main(String args[]){
        double num = 3.5;
        int x,y,z;
        x = (int)Math.round(num);
        y = (int)Math.floor(num);

        System.out.println("X: "+x+"\n"+"Y: "+y);
    }
}
```

```
lliurex@lliurex-ttlportatil:~/Música$ javac Prova.java
lliurex@lliurex-ttlportatil:~/Música$ java Prova
X: 4
Y: 3
lliurex@lliurex-ttlportatil:~/Música$
```