UD3.- Elementos del lenguaje Java

Módulo: Programación
1.º DAM

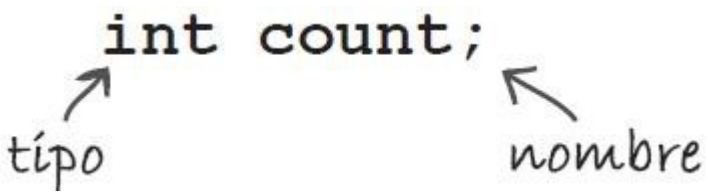
Unión Europea
Fondo Social Europeo
El FSE invierte en tu futuro

INDEX

- Variables
- Tipo de datos
- Constantes
- Literales
- Operadores

Variables

- Permiten almacenar datos, resultados y valores intermedios de un programa
- Tienen asociado un tipo de datos, que determina:
  - Conjunto de valores que la variable puede guardar
  - Operaciones que se pueden realizar con esa variable
- Declaración de variables en Java:

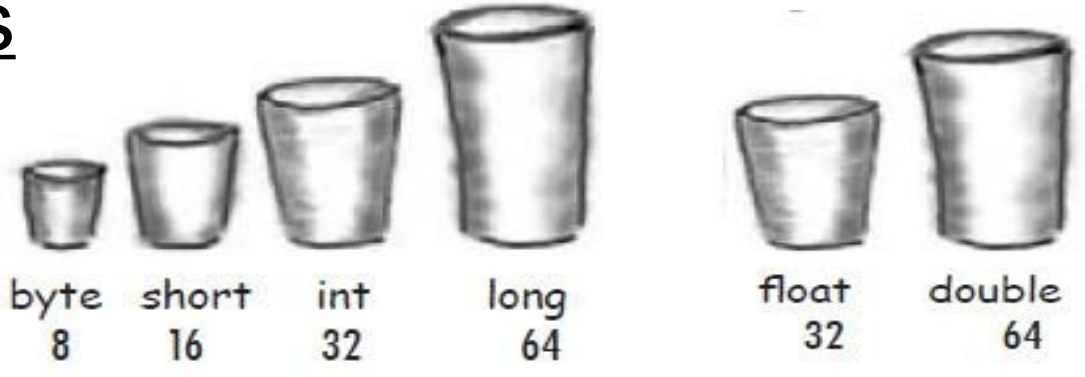


Tipo de datos en Java

- Tipo de datos primitivos
  - boolean, byte, char, double, float, int, long, short
- Tipo de datos referencia
  - String, Array, Class (objetos), Interface

Tipo de datos primitivos

Tipos numéricos



TIPO ENTERO TIPO DECIMAL

NOMBRE	TAMAÑO EN BITS	VALOR MÁXIMO
byte	8	127
short	16	32767
int	32	2147483647
long	64	9223372036854775807
float	32	3.4E+38
double	64	1.7E+308

Tipo de datos primitivos

- Tipo carácter

NOMBRE	TAMAÑO EN BITS	REPRESENTA
char	16	Representa carácteres (letras, números y símbolos especiales) Java utiliza la codificación UNICODE de 2 bytes

- Tipo booleano o lógico

NOMBRE	TAMAÑO EN BITS	REPRESENTA
boolean	8	Puede tomar los valores *true* o false.

Tipo de datos primitivos. Ejemplos

Declaración y asignación de primitivas

```
int x;
x = 234;
byte b = 89;
boolean isFun = true;
double d = 3456.98;
char c = 'f';
int z = x;
```

```
boolean isPunkRock;
isPunkRock = false;
boolean powerOn;
powerOn = isFun;
long big = 3456789;
float f = 32.5f;
```

Para indicar que se trata de un float, porque Java si encuentra un punto decimal, se piensa que es un doble, a no ser que encuentre la 'f'.

¿Qué pasa si el valor es mayor?

- No podemos poner mucha agua en un vaso pequeño.
- Sí podemos, pero se sale del vaso.
- Por ejemplo:
  - `int x = 24;`
  - `byte b = x;`
  - `// no funciona!!`
- El valor de x no será exacto
- Se perderán bits y precisión

Asignación de variables

- Asignando un valor literal después del símbolo =
  - `x=12;`
  - `isGood=true;`
- Asignando el valor de una variable a otra
  - `x=y;`
- Utilizando una expresión combinación de las dos anteriores
  - `x=y+43;`
- Asignando el valor en la misma sentencia de declaración
  - `char caro='A', puesto que2='\u0041';`
  - `float precioPatata=1.2F, precioChoco=2.3F;`

Ámbito de las variables

- En Java hay tres tipos de variables:
  - Variables de instancia: son aquellas que tendrán todos los objetos que se crearán de una clase. Tendrán, en términos generales, valores diferentes para cada objeto.
  - Variables de clase: serán iguales a todos los objetos de la clase.
  - Variables locales: son las que se utilizan en los métodos.

Ámbito de las variables(II)

- El ámbito de una variable es la parte del programa en la cual es conocida y se puede utilizar.
- Una variable local se declara dentro del cuerpo de un método de una clase y es visible únicamente dentro de ese método.
- Las variables se pueden declarar en cualquier lugar del cuerpo del método, incluso después de instrucciones ejecutables, aunque es una buena costumbre declararlas al principio.
- También pueden declararse variables dentro de un bloque entre {…}
  - Únicamente serán "visibles" dentro de ese bloque.
  - Las variables definidas en un bloque tienen que tener

Identificadores de las variables

- Los identificadores son los nombres que se los dan a las variables, clases, interfaces, atributos y métodos de un programa.
- ¿Qué nombres pueden tener las variables?
  - Tiene que empezar con una letra, subrayado (_) o el carácter $. No puede empezar con un número.
  - Después del primer carácter, sí que se pueden utilizar números. No se pueden utilizar espacios en blanco ni símbolos de los operadores.
  - No pueden coincidir con ninguna palabra reservada.
  - El % no está permitido, sí el $ y la ç. También acentos ñ, pero no son recomendables.
  - Recuerda que Java es *case sensitive*.

Listado de palabras reservadas en Java

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

Y los valores *true* y false.

Convenciones para los identificadores

- Los nombres de las variables y los métodos deben empezar por minúscula y los de las clases por mayúscula.
- Si el identificador está formado por varias palabras, la primera se escribe en minúsculas (excepto para las clases) y el resto de palabras empiezan por mayúscula.
  - variable: anyoDeCreacion
  - clase: HolaMundo

Constantes

- Para declarar una constante utilizamos el modificador *final*
  - final double PI = 3.1415926536;

- El valor de una constante no puede ser modificado a lo largo de un programa, a diferencia de las variables.

- Tenemos que darle un valor cuando lo declaramos.

Asignación de variables. Ejemplos

`int size = 32;`	declara un entero llamado ***size*** asignándole el valor 32
`char initial = 'j';`	declara un carácter llamado ***initial*** asignándole el valor 'j'
`double d = 456.709;`	declara un doble llamado ***d*** asignándole el valor 456.709
`boolean isCrazy;`	declara un boolean llamado ***isCrazy*** sin asignarle valor
`isCrazy = true;`	asigna el valor true a la variable ***isCrazy***, antes definida
`int y = x + 456;`	declara un entero llamado ***y*** y le asigna el valor de la suma del valor de x más el valor 456

Literales

- Un literal es un valor que se expresa así mismo.

- Un literal entero puede expresarse:
  - en decimal (base 10) → Ejemplo:21
  - en octal (base 8) → Ejemplo: 025
  - en hexadecimal (base 16) → Ejemplo: x03A
  - puede añadirse al final la letra L o l para indicar que el entero es considerado como un long.

Literales

- Un literal real o decimal puede expresarse:
  - parte entera, (.) y parte fraccionaría
    - Ejemplos:

      345.678

      0.00056
  - notación exponencial o científica
    - Ejemplos: 3.45678e2
  - por defecto es de tipo double. Si queremos que se interprete como *float* tendremos que añadir el sufijo F o f.

Literales

- Literal carácter: se representan entre comillas simples (‘). Puede ser:
  - Un símbolo (letra)
    - Ejemplos: ‘a’ , ‘B’ , ‘{‘ , ‘ñ’ , ‘á’
  - El código Unicode del carácter en octal o hexadecimal.
    - ’\141’ → código Unicode en octal para ’a’
    - ’\u0061’ → código Unicode en hexadecimal para ’a’

- Una “secuencia de escapo”, para caracteres especiales

Secuencia	Significado
'\"'	Comilla simple
'\”'	Comilla doble
'\\'	Contrabarra
'\b'	Backspace
'\	Cambio de línea
'\r'	Retorno de carro
'\te	Tabulador

Literales

- Literal booleano
  - palabras reservadas *true* y false.
    - Ejemplo: boolean activado=false;
- Literal *string* o cadena de caracteres
  - No forman parte de los tipos de datos elementales de Java
  - Entre comillas dobles (“)
    - Ejemplos:
      - System.out.println(”Primera línea\nSegunda línea del string\n”);
      - System.out.println(”Hol\u0061”);

Conversión de tipo de datos

- Cuando se realiza una asignación:
  - identificador = expresión;
- tanto la variable como la expresión tienen que ser del mismo tipo o de tipos compatibles.

- Una expresión puede asignarse a una variable siempre que sea de un tipo de menor tamaño que el tipo de la variable. Por lo tanto, podemos asignar en el siguiente orden:
  - byte → short → int → long → float → double

Conversión de tipo de datos

- Otras formas de conversión de tipo se pueden realizar explícitamente a través del *casting*.
  - (tipo) expresión
    - Ejemplo: num = (int) 34.56;
- También utilizando funciones adecuadas de algunos paquetes de Java.

Operadores. Operadores unarios

- Signo
  - Poner un signo + o un signo – antes de una expresión.
  - Ejemplos: +45, -32

Operadores. Operadores unarios

- Incremento (++) y decremento (--)
  - Aumentar y disminuir en 1 el valor de la variable
  - Pueden ir antes (pre) o detrás (post) de la variable.
  - Ejemplo:
    - `int valor, y=5;`
    - `i++;  //ahora i vale 6; es equivalente a i=i+1`
    - `i--;    //ahora i vale 5; es equivalente a i=i-1`
  - La diferencia entre pre y post aparece en una instrucción compleja:
    - `valor=i++; //ahora valor vale 5, i vale 6`
      `// equivale a { valor=i; i=i+1; }`
    - `valor=++i; // ahora valor vale 7, i vale 7`
      `// equivale a {i=i+1; valor=i; }`

Operadores. Operadores aritméticos

- Pueden realizar operaciones aritméticas que implican el cálculo de valores numéricos representados por literales, variables, otras expresiones, invocaciones a funciones y propiedades y constantes.

- Sintaxis

```
expresion1 operador_aritmético expresion2
```

- Ejemplo

```
int x;
x = 52 * 17;
x = 120 / 4;
x = 67 + 34;
x = 32 - 12;
X = 7 % 2;
```

Operadores. Operadores aritméticos

- + suma
- - resta
- * multiplicación
- / división
- % resto de la división entera

Operadores. Operadores de comparación

- Símbolos que evalúan expresiones condicionales y devuelven un *boolean*.

- Sintaxis

```
expresion1 operador_de_comparación expresion2
```

Operadores. Operadores de comparación

Operador	*true* si...	*false* si...
< (menor que)	expr1<expr2	expr1>=expr2
<= (menor o igual que)	expr1<=expr2	expr1>expr2
> (mayor que)	expr1>expr2	expr1<=expr2
>= (mayor o igual que)	expr1>=expr2	expr1<expr2
= = (igual)	expr1==expr2	expr1!=expr2
!= (distinto)	expr1!=expr2	expr1==expr2

Operadores. Operadores de comparación

- Ejemplo

```
int cantidad;
boolean pedidoGrande;
pedidoGrande = cantidad > 1000;
```

```
boolean testResult ;
testResult = ( 45 < 35 );
testResult = ( 45 == 45 );
testResult = (4 != 3 );
testResult = ('a' > 'b');
```

Método equals()

- El método equals() se utiliza para comparar los valores de los objetos.
- Uno de los usos más comunes se para comparar dos cadenas String (tipo no primitivo).

```
String s = new String(“Hola”);
if(s.equals(“Hola”)) {
   System.out.println(“Son iguales”);
```

Método equals() vs operador ==

- El operador == se utiliza para comparar valores de tipos primitivos de datos, es decir, devolverá true cuando los dos valores sean iguales.
- Si aplicamos el operador == a objetos (variables de referencia a objetos), lo que comparará es si las referencias apuntan al mismo objeto, se a decir, apuntan a la misma dirección de memoria.
- Por lo tanto no tenemos que utilizar el operador == para comparar objetos (ni tipos Strings o cualquier dato que no sea un tipo primitivo).

Método equals() vs operador ==

```
String cadena = new String("Hola");

String mensaje = new String("Hola");

//Serán diferentes porque son objetos distintos
if(cadena == mensaje) {
  System.out.println("Son iguales");
} else {
  System.out.println("Son diferentes");
}
//Serán iguales porque se compara su valor
if(cadena.equals(mensaje)) {
  System.out.println("Son iguales");
} else {
  System.out.println("Son diferentes");
}
```

Operadores. Operadores lógicos

- Realizan una evaluación lógica de expresiones y devuelven un valor *boolean*.
- Sintaxis

```
expresion1 operador_lógico expresion2
```

- Ejemplo

```
edad>18 && sexo=='H'

edad>18 || sexo=='H'
```

Operadores. Operadores lógicos

Operador	Función
&&	Combina dos expresiones. Cada expresión tiene que ser *true* para que toda la expresión sea *true*.
||	Combina dos expresiones. Si una expresión es *true,* toda la expresión es *true*.
!	Devuelve el negativo lógico de la entrada.

Operadores lógicos. Tablas de verdad

Operador AND

expr1	expr2	resultado
false	false	false
false	true	false
true	false	false
true	true	true

Operador NOT

expr1	resultado
false	true
true	false

Operador OR

expr1	expr2	resultado
false	false	false
false	true	true
true	false	true
true	true	true

Combinar operadores lógicos y de comparación

- Podemos combinar operadores de comparación y operadores lógicos con instrucciones condicionales.

```
edad <= 25 && edad >=14
```

Operadores. Operadores de asignación

Operador	Ejemplo
=	edad=34;
+=	edad+=1; edad=edad+1;
-=	edad-=3; edad=edad-3;
*=	edad*=2; edad=edad*2;
/=	edad/=2; edad=edad/2;
%=	edad%=2; edad=edad%2;

Operadores. Operadores de concatenación

- Permite generar una cadena de caracteres a partir de otras dos

  expr1 + expr2

- Ejemplo: “Hola” + “,” + “buen día”

Operadores. Prioridad

- 1.Paréntesis, de dentro hacia fuera.
- 2. Unarios
- 3. Aritméticos
  - A. Multiplicativos: * / %
  - B. Sumativos: + -
- 4. Comparativos o relacionales
- 5. Lógicos o booleanos
  - A. NOT
  - B. AND
  - C. OR
- 6. Asignación
- Cuando aparecen operadores de la misma prioridad juntos en una expresión, el compilador evalúa cada operación de izquierda a derecha.

Operadores. Prioridad

- Ejemplo:
  - a = -3 + 5 + 2 * 4 – 4 / 2 * 3 - 5% 2;
  - ¿Cuál es el valor final de esta expresión?
  - ¿Y de la siguiente expresión?
  - b = -3 + 5 + 2 * 4 – 6 / 4 * 3 - 5% 2;

Visualización de información por pantalla

- Métodos *print()* y println().
  - *println()* incluye un salto de línea al final de la salida

```
System.out.print("Se imprime este mensaje sin
salto de línea");

System.out.println("Se imprime este mensaje con
un salto de línea");
```

Entrada de datos por teclado

- El método *read( )* lee un único carácter

```
char c = (char) System.in.read();
```

- Esta forma de leer la información del teclado es poco práctica (imagina que tuviéramos que leer cada línea de texto carácter a carácter...)
  - Solución: declarar un objeto de la clase *Scanner* y emplear sus métodos.

Entrada de datos por teclado

```
//Importem el fitxer on està la classe
import java.util.Scanner;

public class Exemple {
	public static void main (String[] args) {
		int primerNum;
		//Es declara l'objecte lector de la classe Scanner
		Scanner lector = new Scanner(System.in);
			...
		System.out.print("Escriu un número i polsa retorn: ");

		//Es llig un valor enter per teclat i s'espera al retorn.
		primerNum = lector.nextInt();
		lector.nextLine();
	...
	}
}
```

Entrada de datos por teclado. Métodos

Método	Tipo de datos leído
lector.nextByte()	byte
lector.nextShort()	short
lector.nextInt()	int
lector.nextLong()	long
lector.nextFloat()	float
lector.nextDouble()	double
lector.nextBoolean()	boolean
lector.next()	String
lector.nextLine()	String

Tipos enumerados

- Son conjuntos de valores constantes para los cuales no hay un tipo predefinido.
  - Por ejemplo: para representar los días de la semana, estaciones del año, meses del año, etc...

```
public class Semana {

    public enum DiaSemana{LUNES, MARTES, MIERCOLES,
                          JUEVES, VIERNES, SABADO, DOMINGO}

    public static void main (String[] args){

        DiaSemana hoy = DiaSemana.JUEVES;

        DiaSemana ultimo=DiaSemana.DOMINGO;

        System.out.println( "Hoy es " +hoy +"\n Y el ultimo dia es "
                            +ultimo );
    }
}
```