Java

Empezar un programa

```
public class MyClass
{
   public static void main(String[] args)
   {
      System.out.println("Hello");
   }
}
```

```
public class MyClass
{
}
```

Este es el inicio del programa,es completamente necesario poner class al principio del programa para que este funcione,las llaves marcan el inicio y el fin del programa.

```
public static void main(String[] args)
    {
    }
```

Este es el cuerpo del programa,la parte principal que contendrá los comandos del programa.

En java,el bloque principal (**main**) siempre será público(**public**) estático (**static**) y vacío (**void**), y debe tener **string[]** como parámetros.

Normas

Todas las instrucciones deben acabar con punto y coma(;).

Variables

Declaración de variables

Siempre que queramos utilizar una variable tendremos que declararla. Se realiza siguiendo los siguientes pasos:

Se especifica el tipo de la variable.

Se especifica el nombre o identificador de la variable.

```
int myVariable;
```

Podemos asignar valores a las variables,tanto en el momento en el que las declaramos como más adelante en el código.

```
int myVariable = 3;
int myOtherVariable;
...
myOtherVariable = 5;
```

Podemos declarar varias variables a la vez.

```
int number1 = 0, number2, result = 1;
```

Los nombres de variables pueden contener letras(mayusculas o minúsuclas),dígitos y “_”,pero no pueden empezar por números ni contener espacios.

Ejemplos de variables válidas

```
int aNumber;
int another_number;
int number1;
int _one_more_number;
```

Ejemplo de variables no válidas

```
int 1number;
int another number;
```

Declaración de constantes

Las constantes son valores que nunca cambian,en java se declaran como **static final**.

```
class MyClass
{
   static final int MAX_USERS = 10;
   ...
}
```

Tipos de datos

Tipos numéricos

Números enteros

Tipo de dato	Memoria	Rango
Byte	1	-128 to 127
Short	2	-65536 to 65535
Int	4	2.147.483.648 to 2.147.483.647
Long	8	Hasta 18/19 dígitos

Números reales

Los números reales permiten decimales.

Tipo de dato	Memoria	Rango
Float	4	Hasta 6/7 cifras significativas
Double	8	Hasta 15 cifras significativas

Si queremos darle un valor directamente a un float,tendremos que poner “f” al final del valor,si no dará un error de compilación.

```
float pi = 3.14159f;
```

```
double pi = 3.14159265359;
```

Overflow

El “overflow” es cuando excedemos el valor máximo del tipo de variable,tendremos que tener en cuenta que clase de operaciones vamos a hacer al elegir el tipo de variable,para evitar esto.

Tipo texto

Hay 2 tipos de variables de texto:

Char

Se emplea cuando queremos utilizar un único carácter o símbolo.

```
char symbol = 'a';
```

String

Se emplea cuando queremos emplear textos.

```
String text = "Hello world";
```

Secuencias de escape

Hay algunos caracteres que son difíciles de representar con el teclado en un código fuente,por lo que existen las secuencias de escape para representarlas.

Secuencia	Significado
\n	Salto de línea
\t	Tabulación
\"	Comillas dobles
\'	Comillas simples
\\	Barra (/)

Se pueden poner tanto en char como en string.

```
char newLine = '\n';
String message = "Hello world.\n\"Quoted text\"";
```

Operaciones con caracteres

Podemos realizar varias operaciones básicas con caracteres,hay que tener en cuenta que java toma los caracteres internamente como valores numéricos,asignando cada carácter a un código numérico,las letras del alfabeto están representadas por códigos numéricos consecutivos de la “a” a la “z”.

```
char symbol = 'a';
symbol += 3;
```

El valor de symbol en este caso sería ‘d’.

Podemos utilizar el operador “+” para concatenar.

```
String text = "Hello" + 3;
```

El texto resultante sería “Hello3”.

Si quieres realizar una operación y luego concatenarla,debes priorizar la operación empleando paréntesis.

```
String text = "Hello" + (3 + 2);
```

El texto resultante sería “Hello5”.

Boolean

El tipo booleano es otro tipo básico de dato,permite representar dos valores opuestos:

true(verdadero) y **false**(falso).

```
boolean b = true;
```

Uso de operadores relacionales.

```
int n = 10;
boolean check1 = n > 5;     // true
boolean check2 = n != 10;   // false
```

```
String s1 = "Hello";

boolean check1 = s1 == "Hello";      // Does not work as expected
boolean check2 = s1.equals("Hello"); // OK
```

Conversión entre tipos de datos

A veces necesitaremos convertir un valor de un tipo a otro.

Typecast

La conversión entre valores numéricos es bastante sencilla,solo tendremos que hacer un "typecast",especificando entre paréntesis el tipo al que queremos convertir el valor.

```
float pi = 3.1416f;
int piInteger = (int)pi;
```

El valor resultante de piInteger sería 3.

Se puede hacer al contrario también.

```
int number = 5;
double realNumber = (double)number;
```

El valor resultante de realNumber sería 5,0.

No es necesaria la conversión si el tipo original es más pequeño que el tipo al que lo queremos convertir.

```
byte value = 3;
int number = value;
```

Se puede emplear para convertir divisiones enteras en reales.

```
float result = (float) 3 / 2;
```

En este caso el resultado de esa división sería float,aún siendo ambos valores int.

En general,cada operación aritmética intenta producir un resultado del mismo tipo que los de sus operandos,no obstante en las multiplicaciones y divisiones java intentar convertir el tipo del resultado a un tipo mayor,no obstante,podemos especificarle que siga empleando el tipo original.

```
byte a = 3, b = 2;
byte result = (byte)(a + b);
```

Java intentaría convertir el resultado a int,pero seguirá siendo byte ya que se lo hemos especificado.

Si combinamos 2 tipos en una operación aritmética el resultado sería del tipo mayor.

```
float a = 3.5f;
int b = 4;
float result = a * b;
```

Convertir de string/a string

De string

Conversión a int

```
int value = Integer.parseInt("23");
```

Conversión a float

```
float value = Float.parseFloat("3.1416");
```

Conversión a double

```
double value = Double.parseDouble("3.1416");
```

Conversión a byte

```
byte value = Byte.parseByte("23");
```

Conversión a short

```
short value = Short.parseShort("23");
```

Conversión a long

```
short value = Long.parseLong("23");
```

A string

Hay 2 formas de convertir un número a string:

Concatenar el número a un string vacío(“”)

```
int number = 23;
String text = "" + number;
```

Empleando **String.valueOf** para convertir el valor específico en string.

```
int number = 23;
String text = String.valueOf(number);
```

Operadores

Operadores aritméticos

Operador	Significado
+	Suma
-	Resta
*	Multiplicación
/	División
%	Resto(módulo)

Orden de operaciones

1º multiplicación/división/módulo

2º suma/resta

Si son varias del mismo rango,se evaluan de izquierda a derecha.

Operadores de asignación

Operador	Significado
=	Asignación simple
+=	Auto suma
-=	Auto resta
*=	Auto multiplicación
/=	Auto división
%=	Auto módulo

operadores relacionales

Operador	Significado
>	Mayor a
>=	Mayor o igual a
<	Menor a
<=	Menor o igual a
==	Igual a
!=	Distinto de

Para saber si dos valores son iguales se emplea == y no =,este se emplea para asignar valores a variables.

El operador == no funciona con los strings,si queremos comparar si dos string son iguales,tendremos que usar equals en vez de ==.

Operadores lógicos

Operador	Significado
&&	Y
||	O
!	Negación

El operador Y(&&) permite unir dos comparaciones,el resultado solo será true cuando ambas sean ciertas.

```
int n = 10, m = 5;

boolean c1 = n > 5 && m < 10;   // true && true = true
boolean c2 = n > 5 && m > 10;   // true && false = false
```

El operador O(||) permite unir dos comparaciones,el resultado será true cuando una de ellas sea cierta.

```
int n = 10, m = 5;

boolean c1 = n > 5 || m > 10;   // true || false = true
boolean c2 = n < 5 || m > 10;   // false || false = false
```

El operador ! Sirve para negar o invertir una expresión.

```
int n = 10;

bool c1 = n > 5;    // true
bool c2 = !c1;      // false
```

El orden de los operadores es el siguiente:

Primero se resuelven los paréntesis

Luego se evalúa es operador **!**

Luego se evalúa el operador **&&**

Finalmente se evalúa el operador ||

Auto incremento y autodecremento

Si ponemos el operador antes,primero se realizará el aumento/decremento y luego la expresión.

```
int result = 3;
result++;
```

Si ponemos el operador después,primero se realizará la expresión y luego el aumento/decremento.

```
int result = 3;
result++;
```

Auto incremento

Automáticamente aumenta el valor en 1.

```
int result = 3;
result++;
```

Auto decremento

Automáticamente disminuye el valor en 1.

```
int result = 3;
result++;
```

Comentarios

Los comentarios sirven para añadir texto que el compilador ignora,pero que puede ayudar a entender el código o hacer aclaraciones.

En java hay 2 tipos de comentarios:

Comentarios de una línea

```
// We declare an integer variable
int variable = 3;
```

Comentarios de varias líneas

```
/* This is a comment of
   multiple lines before
   declaring a variable */
int variable = 3;
```

Input y Output

y le permiten al usuario introducir datos.

Output

Permiten mostrar cosas por pantalla.

Se pueden emplear las instrucciones:

System.out.print

Muestra el texto por pantalla.

System.out.println

Muestra el texto por pantalla y hace un salto de línea.

```
int result = 12;
System.out.println("The result is " + result);
System.out.print("Have a nice day!");
```

Output formateado

Además de las formas anteriores,si queremos darles un formato al output podemos emplear la instrucción **System.out.printf**.

Empleamos unos caracteres especiales para determinar el tipo del dato.

```
System.out.printf("The number is %d", number);
```

Símbolos que representan los tipos:

%d Para números enteros(long, int).

%f Para números reales (float and double).

%s Para strings.

%c Para characters (char).

%n Para realizar un salto de línea.

Podemos añadir todos los símbolos que queramos,pero tendremos que especificar al final ese mismo número de parámetros.

```
System.out.printf("The average of %d and %d is %f",
   number1, number2, average);
```

En los símbolos %d y %f podemos añadir cierta información entre el “%” y la letra,para especificar información del formato.

Podemos especificar el número de cifras.

```
System.out.printf("The number is %05d", number);
```

Si el número introducido es 33 el número sería 00033

Si no ponemos 0 al principio aparecerán espacios vacíos en su lugar

```
System.out.printf("The number is %10d", number);
```

Si el número introducido es 33 el número sería “ 33”

Podemos especificar el número de cifras en números reales.

```
System.out.printf("The number is %3f", number);
```

Podemos especificar a su vez el número de cifras decimales.

```
System.out.printf("The number is %3.3f", number);
```

Input

Para obtener input del usuario, utilizamos **Scanner**.

Tenemos que importar **java.util.Scanner** para poder usarlo.

Tenemos que crear un elemento Scanner y usar sus métodos para obtener datos del usuario.

Algunos métodos son:

nextLine:Para leer texto.

nextInt:Para leer enteros.

nextFloat:Para leer números reales.

nextBoolean:Para leer booleanos.

Etc…

```
import java.util.Scanner;
...
public class ClassName
{
   public static void main(String[] args)
   {
      Scanner sc = new Scanner(System.in);
      int number = sc.nextInt();
      String text = sc.nextLine();
      sc.close();
   }
```

Esto nos ayuda a leer tipos de datos específicos.

```
int number1, number2;
number1 = sc.nextInt();
number2 = sc.nextInt();
```

System.console().readLine()

Similar al método newline,este leerá toda la línea hasta que presiones enter,**SIEMPRE** obtendremos un string,por lo que tendremos que convertirlo al tipo correspondiente más tarde.

```
System.out.println("Write a number:");
String text = System.console().readLine();
int number = Integer.parseInt(text);
```