# Sintaxis

public class MyClass {

public static void main(String[] args) {

System.out.println("Hello World");

}

}

Cada aplicación empieza con un nombre de clase que debe coincidir con el nombre del archivo: public class **NombreArchivo**

## Método principal:

### Main:

main() es un método obligatorio y está en todos los programas:

public static void **main**(String[] args)

### Static

Indica que el método es estático y permite acceder a las variables y métodos, aunque no tengamos una instancia del objeto que los contiene

### Public

Indica que el campo es accesible desde cualquier clase

### Clase(class)

### println:

Para imprimir una línea de texto en pantalla usamos el método println():

System.out.**println**("Hola mundo")

### Comentario:

Para poner un comentario en el código sin que afecte al mismo se usa // :

// Comentario

System.out.println("Hello World"); // Comentario

Para introducir comentarios de varias líneas se usa /\* y se cierra con \*/:

/\* HOLAAAAAAAAAAAAAAAAAAAAAAAAAAA

AAAAAAAAAAAAAAA \*/

System.out.println("Hello World");

## Variables de java:

Las variables con contenedores para almacenar valores

### Sintaxis:

tipo variable = valor;

También puedes declarar una variable sin asignar el valor y asignar el valor más tarde:

tipo variable;

variable = valor;

### String:

Almacena texto: **“Hola”**

String nombre = "Angel";

System.out.println(nombre);

### int:

Almacena números enteros: **9/10/34**

int Num = 15;

System.out.println(Num);

### float:

Almacena números de coma flotante: **5,88/-5,88**

float numf = 4,2;

System.out.println(numf);

### char:

Almacena caracteres individuales: **‘a’/’B’**

char Letra = 'A’;

System.out.println(Letra)

### boolean:

Almacena valores con dos estados: **Verdadero o falso**

Boolean bool = false;

System.out.println(bool);

# Introducir parámetros por línea de comandos

Se usa la clase Scanner para obtener información directamente del usuario introduciendo datos mediante el teclado. **Sintaxis:**

**import java.util.Scanner**; // Importa la clase Scanner

class MyClass {

public static void main(String[] args) {

**Scanner nombre = new Scanner(System.in)**; // Crea un objeto Scanner

System.out.println("Introduce el nombre de usuario");

String usuNombre **= nombre.nextLine()**; // Lee el texto introducido por el usuario

System.out.println("El nombre de usuario es: " + usuNombre); // Output user input

}

}

La entrada puede variar dependiendo del tipo de dato:

- nextBoolean(): Lee un valor booleano del usuario

- nextByte(): Lee un valor de byte del usuario

- nextDouble(): Lee un valor doble del usuario

- nextFloat(): Lee un valor flotante del usuario

- nextInt(): Lee un valor int del usuario

- nextLine(): Lee un valor de cadena del usuario

- nextLong(): Lee un valor largo del usuario

- nextShort(): Lee un valor corto del usuario

## Operadores aritméticos

Los operadores aritméticos sirven para realizar operaciones con variables y valores:

* +: suma dos valores
* -: Resta dos valores
* \*: Multiplica dos valores
* /: Divide dos valores
* %: Devuelve el resto de una división entre dos valores
* ++: Incrementa en 1 un valor
* --: Disminuye en 1 un valor

### Sintaxis:

int x = 100 + 50;

int y = x --;

## Operadores de asignación

* = (x=3): x=3
* += (x+=3): x = x + 3
* -= (x-=3): x = x - 3
* \*= (x\*=3): x = x \* 3
* /= (x/=3): x = x / 3
* %= (x%=3): x = x % 3
* &= (x&=3): x = x & 3
* |= (x|=3): x = x | 3
* ^= (x^=3): x = x ^ 3
* >>= (x>>=3): x = x >> 3
* <<= (x<<=3): x = x << 3

## Operadores de comparación

* == (x == y): Igual
* ¡= (x != y): No igual
* > (x > y): Mayor que
* < (x < y): Menor que
* >= (x >= y): Mayor o igual que
* <= (x <= y): Menor o igual que

## Operadores lógicos

* && (x<5 && 1<10): Devuelve el valor verdadero si las dos declaraciones son verdaderas
* || (x<5 || 1<10): Devuelve el valor verdadero si una declaración es verdadera
* ! [!(x < 5 && x < 10)]: Invierte el resultado, devuelve falso si el resultado es verdadero

# Strings de java

Los Strings son variables que almacenan texto. Contiene un conjunto de caracteres rodeados por comillas:

String saludo = "Hola";

Los String en java son objetos, que contienen métodos que realizan operaciones en los Strings

## Métodos:

Un método es un conjunto de instrucciones definidas dentro de una clase, que realizan tareas determinadas y que se pueden invocar con un nombre. La sintaxis para usar métodos es:

System.out.println(variable.**método()**)

### length

El método length indica el número de caracteres de un String:

String abc = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

System.out.println("El abecedario tiene: " + abc**.length()** +” letras”);

### toUpperCase() y toLowerCase()

* toUppercase: Cambia el texto a mayúscula
* toLowerCase: Cambia el texto a minúscula

String txt = "Hola mundo";

System.out.println(txt**.toUpperCase()**); // Devuelve "HOLA MUNDO"

System.out.println(txt**.toLowerCase()**); // Devuelve "hola mundo "

### IndexOf()

El método indexOF devuelve la primera posición de un texto específico en una cadena incluyendo los espacios en blanco

String txt = "Please locate where 'locate' occurs!";

System.out.println(txt**. IndexOf("locate")**); // Outputs 7

### Concatenación de cadenas

El operador + Se puede usar para concatenar textos

String nombre = "Ángel";

String Apellido = "Marco";

System.out.println(nombre **+** " " **+** Apellido);

También se puede usar el método concat() para concatenar Strings:

String nombre = "Ángel";

String Apellido = "Marco";

System.out.println(nombre.**concat(Apellido)**);

## Caracteres especiales

Los Strings se escriben en declaraciones que están delimitadas por comillas (“), por tanto, si un texto incluye comillas puede dar lugar a errores:

* \’: ‘
* \”: “
* \\: \

## Math

Es una class que contiene un conjunto de métodos para realizar cálculos matemáticos

### Math.max(x,y)

Es un método para encontrar el número más grande en una lista

**Math.max(**5, 10**)**; \\Devuelve 10

### Math.min(x,y)

Es un método para encontrar el número más pequeño en una lista

**Math.min(**5, 10**)**; \\Devuelve 5

### Math.random()

Devuelve un número aleatorio entre 0 y 1

Math.random

### Math.round()

Redondea un número con decimales

Math.round(número \* 100.0) / 100.0

El número de 0 es igual al número decimales que quieras poner

### Math.pow()

Eleva una base a un exponente

Math.pow(base, exponente)

## Booleans

Son valores que solo pueden ser o verdaderos o falsos

### Expresiones booleanas

Es una expresión de Java que devuelve un valor booleano. Se pueden usar operadores de comparación

int x = 10;

int y = 9;

System.out.println(x > y); //Devuelve true

# IF … Else

Establece una condición y una acción en caso de que se cumpla o no esa condición:

* If: Establece el código que se ejecutara en caso de que se cumpla una condición
* Else: Establece el código que se ejecutara en caso de que no se cumpla la condición establecida en el if
* Else if: Establece otro if en caso de que no se cumpla la condición establecida en el if

**If (x<y) {**

System.out.print(“y es mayor que X”)

} **else if (x==y) {**

System.out.print(“y es igual a X”)

} **else {**

System.out.print(“x es mayor que y”)

}

# Switch

Sirve para elegir un bloque de código concreto en un caso determinado

**switch(expression) {**

**case** x:

// bloque de código

**break**;

**case** y:

// bloque de código

**break**;

**default**:

// bloque de código

}

# Bucles

Ejecutan un bloque de código repetidamente hasta o mientras se cumpla una condición.

## While

Recorre un bloque de código mientras se cumpla una condición:

int i = 0;

**while (i < 5) {**

System.out.println(i);

i++;

}

### Do…while

Es una variante del bucle while, que ejecuta el bloque una vez antes de comprobar si la condición es verdadera, y luego se repetirá el bucle mientras la condición se esté cumpliendo

int i = 0;  
**do {**

System.out.println(i);

i++;

}

**while (i < 5);**

## For

Se usa cuando sabes cuantas veces quieres repetir un bloque de código:

for (declaracion1; declaracion2; declaracion3) {

// bloque de código a ejecutar

}

**La declaración 1** se ejecuta (una vez) antes de la ejecución del bloque de código.

**La declaración 2** define la condición para ejecutar el bloque de código.

**La declaración 3** se ejecuta (cada vez) después de que se haya ejecutado el bloque de código.

El siguiente ejemplo imprimirá los números del 0 al 4

**for (int i = 0; i < 5; i++) {**

System.out.println(i);

}

# CASting

Operación que permite realizar una conversión entre diferentes tipos de variables

## Casting implícito

Cuando el tipo de dato de destino es mayor en memoria que el del tipo de origen

byte -> short -> char -> int -> long -> float -> double

int datoInt = 9;

double datoDouble = datoInt;

## Casting explícito

Cuando el tipo de dato de destino es menor en memoria que el del tipo de origen

double -> float -> long -> int -> char -> short -> byte

double datoDouble = 9.78d;

int datoInt = **(int)** datoDouble

# Parse

Es un método que consiste en cambiar el tipo de dato a una cadena de texto

## Sintaxis

String texto = “Texto”;

Tipo **cambio** = **Nombretipo**.parse**tipo**(texto);

System.out.println(**cambio**);

## ParseInt:

String fecha = “2005”;

Int numero = **Integer.parseInt(**fecha**)**;

System.out.println(“El año de nacimiento es: ”+numero);

# Manejo de excepciones

Para manejar y prevenir errores se usa **try** y **catch** que sirven para detectar excepciones y responder de una forma determinada

## Try y catch

Try sirve para “intentar” ejecutar una línea de código, y catch se encarga de realizar una acción concreta en caso de que la línea de código produzca un error

**try {**

línea de código;

**} catch (Excepción)** **{**

Respuesta al error;

}

# Arrays

Los arrays se usan para almacenar valores múltiples en una sola variable, sin tener que declarar variables con valores separados. Para declarar un array, se define el tipo de variable con corchetes:

String [] juegos;

Para insertar valores en el array:

String [] juegos = {“Nintendo”, “Xbox”, } ;//Array de cadenas

Int[] numeros = {10, 20, 30}; //Array de enteros

## Acceder a los datos

Para acceder a los datos de un array se hace referencia a su número índice

String**[]** coches = **{"Volvo", "BMW", "Ford", "Mazda"};**

System.out.println(coches**[0]**);

// Devuelve Volvo

## Cambiar datos de un array

Para cambiar un dato específico de un array, tienes que llamar al array y al número de índice del dato que quieras cambiar:

coches [3] = “Opel” // Cambiaría el cuarto dato de la lista (comienza a partir de 0) a Opel