Junta de Andalucía
Consejería de Educación y Deporte

UD5
Programación Orientada a Objetos
–
Módulo de Programación
1º DAW

Autor: Fran Gómez
2025/2026

Objetivos

- **Comprender** los fundamentos de la POO
- **Identificar** los conceptos básicos de la POO en una aplicación.
- **Manejar** el lenguaje de programación Java
- **Reforzar** las estructuras de control y de datos estudiadas en los temas anteriores

RA y CE Centrales

RA2 Escribe y prueba programas sencillos, reconociendo y aplicando los fundamentos de la programación orientada a objetos
a) Se han identificado los fundamentos de la programación orientada a objetos.
b) Se han escrito programas simples.
c) Se han instanciado objetos a partir de clases predefinidas.
d) Se han utilizado métodos y propiedades de los objetos.
e) Se han escrito llamadas a métodos estáticos.
f) Se han utilizado parámetros en la llamada a métodos.
g) Se han incorporado y utilizado librerías de objetos.
h) Se han utilizado constructores.
i) Se ha utilizado el entorno integrado de desarrollo en la creación y compilación de programas simples

RA y CE secundarios

RA1 Reconoce la estructura de un programa informático, identificando y relacionando los elementos propios del lenguaje de programación utilizado.
a) Se han identificado los bloques que componen la estructura de un programa informático
b) Se han creado proyectos de desarrollo de aplicaciones
c) Se han utilizado entornos integrados de desarrollo.
d) Se han identificado los distintos tipos de variables y la utilidad específica de cada uno.
e) Se ha modificado el código de un programa para crear y utilizar variables.
f) Se han creado y utilizado constantes y literales.
g) Se han clasificado, reconocido y utilizado en expresiones los operadores del lenguaje.
h) Se ha comprobado el funcionamiento de las conversiones de tipo explícitas e implícitas.
i) Se han introducido comentarios en el código.

RA y CE secundarios

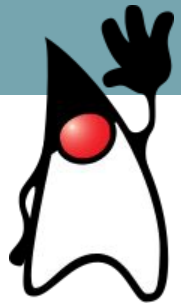
RA3 Escribe y depura código, analizando y utilizando las estructuras de control del lenguaje
a) Se ha escrito y probado código que haga uso de estructuras de selección.
b) Se han utilizado estructuras de repetición.
c) Se han reconocido las posibilidades de las sentencias de salto.
e) Se han creado programas ejecutables utilizando diferentes estructuras de control.
f) Se han probado y depurado los programas.
g) Se ha comentado y documentado el código.

¿Cómo lo vamos a evaluar?

Instrumento	Técnica
Portfolio: - Se evalúa el trabajo diario - Recurso para estudiar - Útil como CV	- Salir a corregir los ejercicios de clase - Defender los ejercicios necesarios frente al profesor el día antes del examen.
Proyecto: - Evaluación de todo lo aprendido y más - Aplicación en diferentes contextos - Colaborativo	- Exposición: explicar qué y cómo funciona a la clase, respondiendo a las preguntas.
Examen: - Evaluación individual y sumativa de lo fundamental	- Prueba escrita

Índice de contenidos

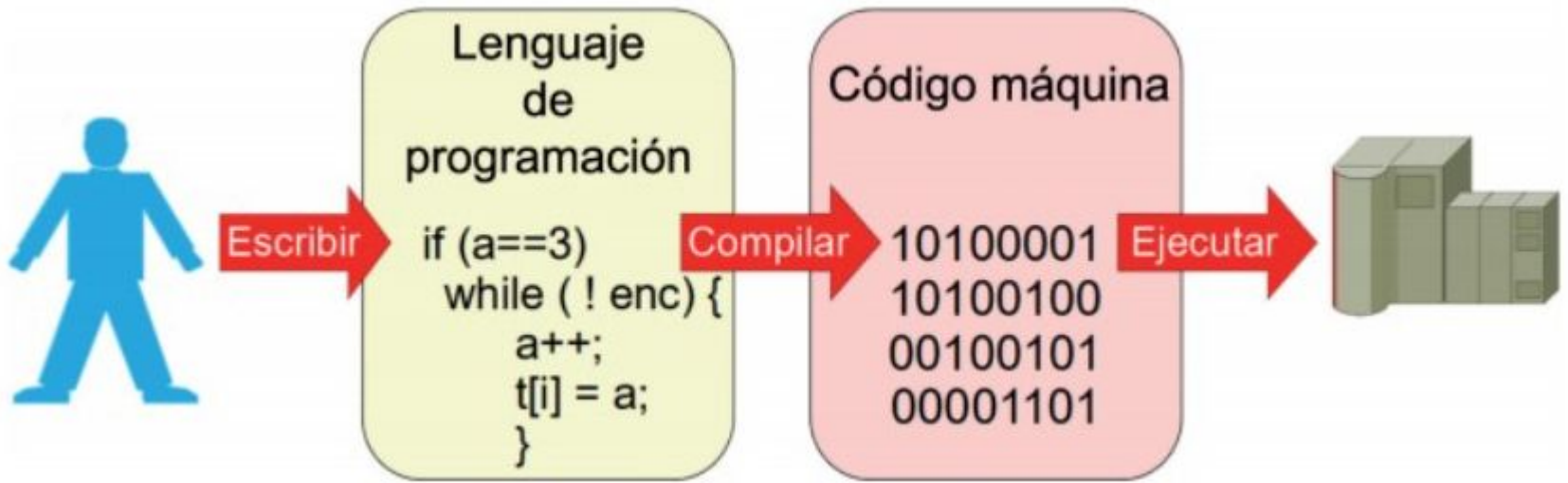
¿Por qué Java?



[Índice TIBOE]

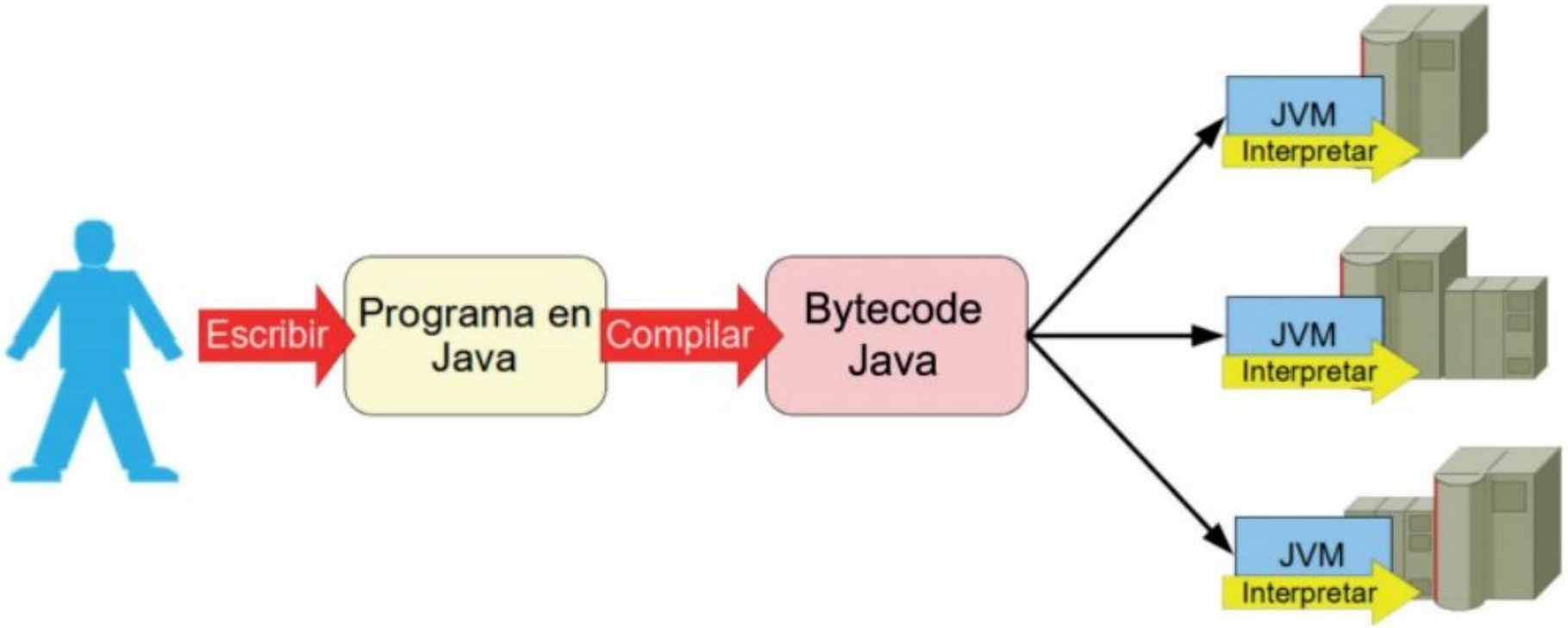
	Java	**C++**	**Python**
POO	Pura	Soporta	Soporta
Propósito	General: web, móvil, backend, empresa	Específico: videojuegos, rendimiento	Específico: web, IA, Machine learning
Aprendizaje	Intermedio	Difícil	Fácil
Multiplataforma	Sí	No	Interpretado
Popularidad	Alta	Alta	Alta

Código fuente vs código máquina

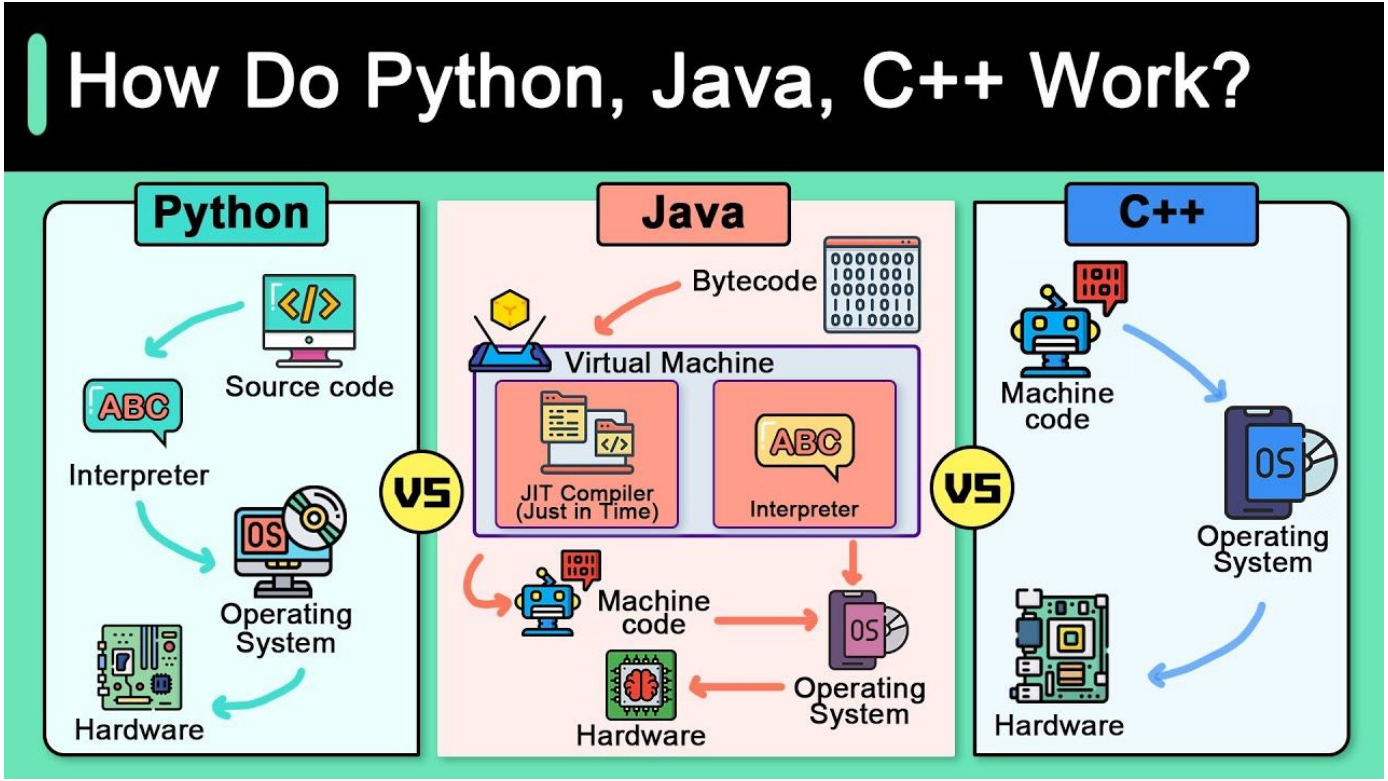


Cada plataforma (HW+SW) entiende sólo su conjunto de instrucciones

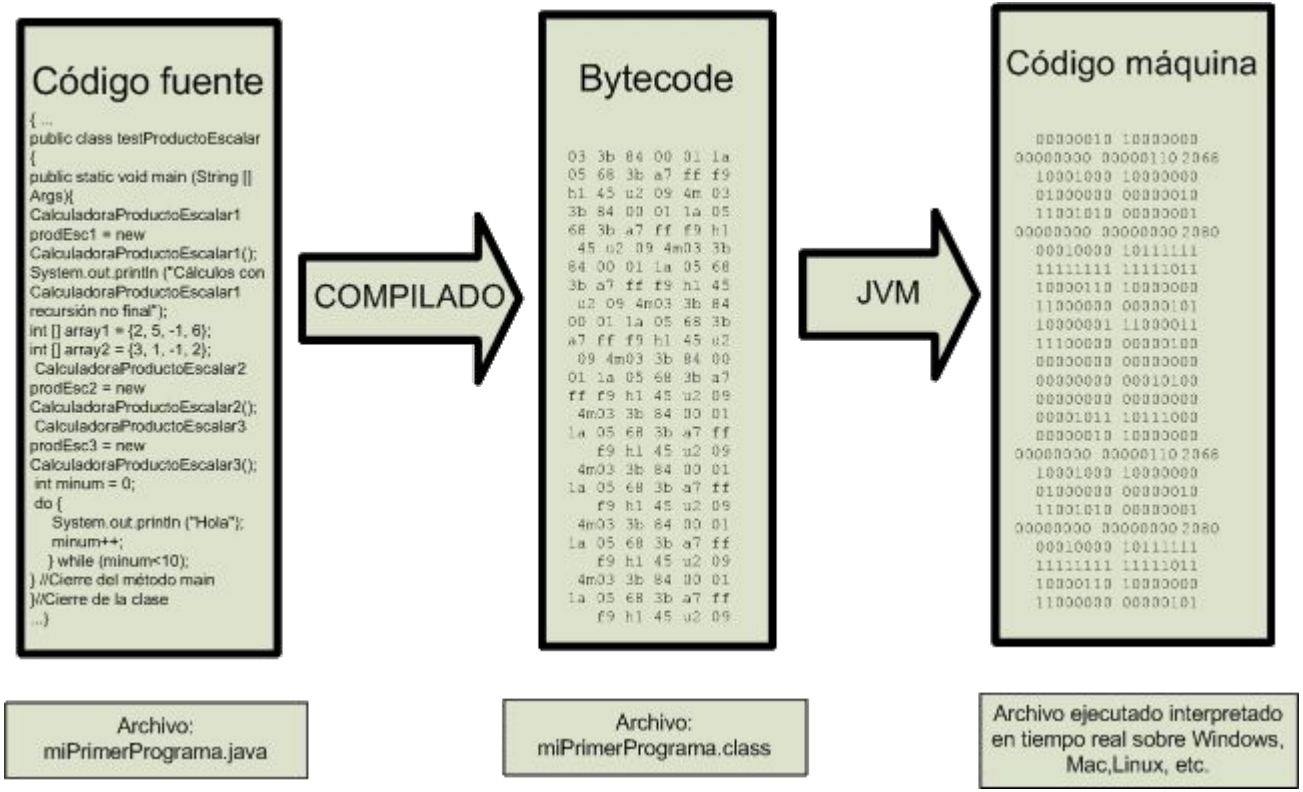
Multiplataforma



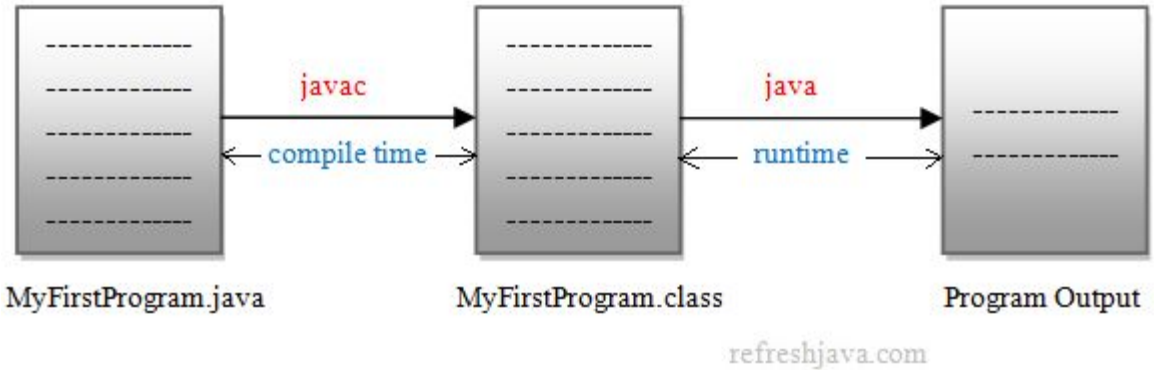
Maquina Virtual de Java



Código fuente vs código máquina



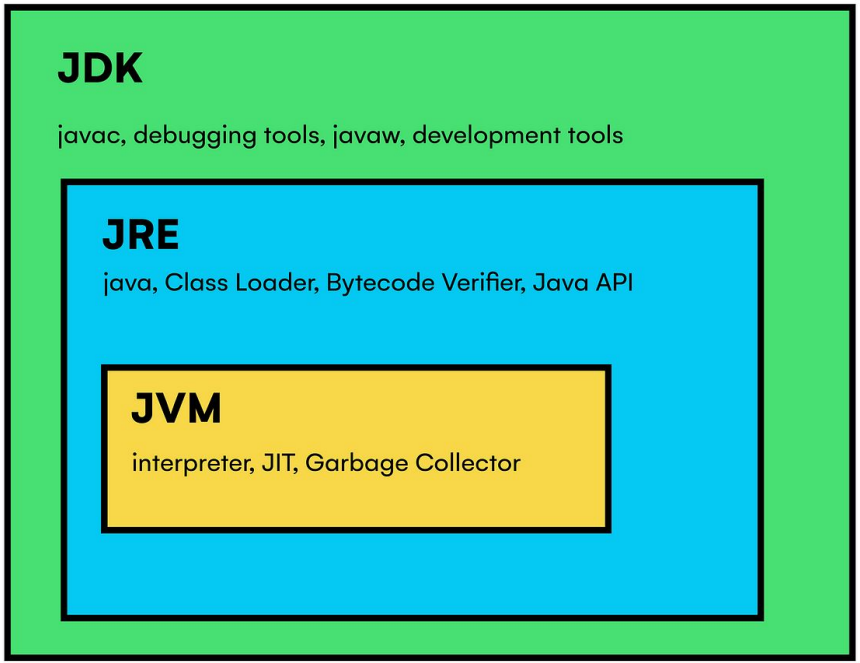
Tiempo de compilación vs tiempo de ejecución

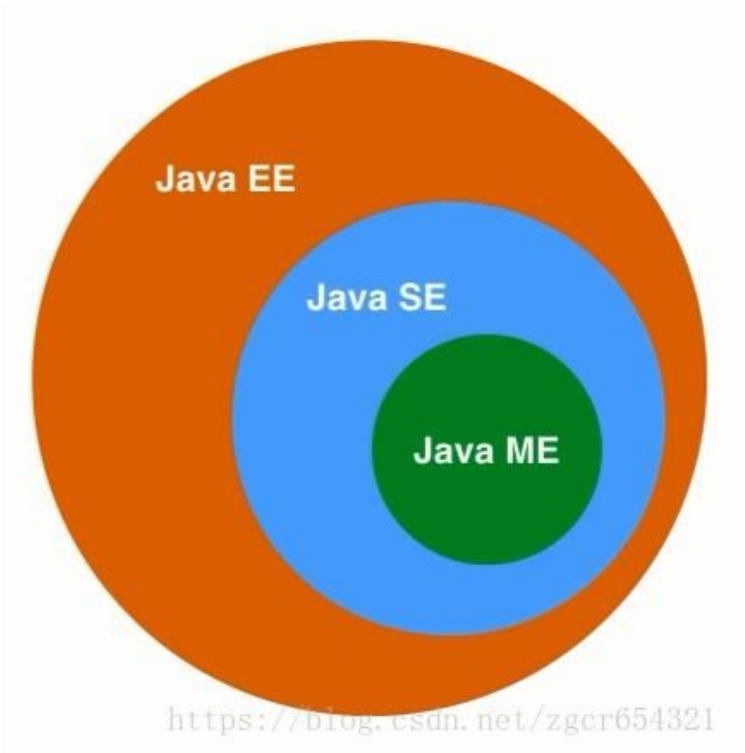


Tiempo de compilación vs tiempo de ejecución

	Compile time	**runtime**
Definición	Código fuente se traduce a código máquina	Instrucciones compiladas o interpretadas se ejecutan
Tareas	- Comprobación de sintaxis - Optimización del código - Generación de código binario	- Ejecución de las instrucciones - Asignación y liberación de memoria - Interacción con el usuario o sistema operativo
Errores	Sintaxis, declaraciones de variables incorrectas, problemas de tipos de datos, etc.	Excepciones (división por cero, acceso a memoria no válida, etc.)

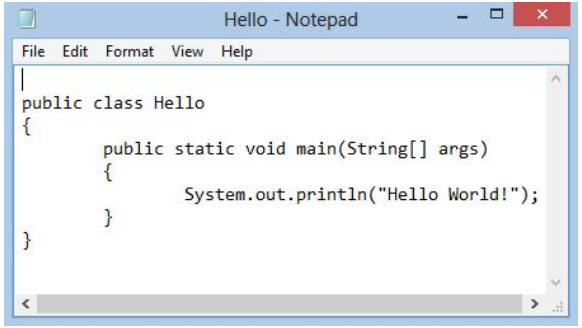
Instalación de Java





Consola

1. Escribir el código fuente → Fichero de texto con extensión .java



2. Compilar código fuente (.java) → Instalar compilador (JDK)

```
>javac HelloWorld.java
```

3. Ejecutar código ejecutable o bytecode (.class) → Instalar JVM (JRE)

```
>java HelloWorld
```

Palabras reservadas

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

literales reservados: true, false, null

deprecadas: const, goto

Variable

nombreVariable = <valor de la variable>

a = b + 1

Identificador

Nombran variables, funciones, clases y objetos; cualquier cosa que el programador necesite identificar o usar.

En Java, un identificador comienza con una letra, un subrayado (_) o un símbolo de dólar ($). Los siguientes caracteres pueden ser letras o dígitos.

Java es sensible a mayúsculas

Java Code Conventions

- Nombre variables y métodos: lowerCamelCase
- Nombre clases: UpperCamelCase

Tipos de datos: variables y memoria

00	00111000
01	11111101
02	01011110
03	10000111
…	

Paso de decimal a binario:

00111000 11111101 01011110 10000111

Dar un tipo de dato a una variable es la forma de decir cuánto va a ocupar esa variable en memoria.

Ejemplo:

int telefono = 956128903;

El tipo entero ocupa 4 bytes

Tipos primitivos

Nombre	Tipo	Tamaño	Defecto	Ejemplo	Rango
boolean	Lógico	1 bit	false	*boolean salir = true;*	true - false
char	Carácter	2 bytes	null	char c = ‘a’;	Unicode
byte	Entero	1 byte	0	byte edad = 30;	[-128, 127]
short	Entero	2 bytes	0	short distancia = 1500;	[-32.768, 32.767]
int	Entero	4 bytes	0	int poblacionMundial = 8_000_000_000;	$[-2^{63}, 2^{63}-1]$
long	Entero	8 bytes	0	long cuentaBancaria = 987654321012345678L;	
float	Real	4 bytes	0	float piAproximado = 3.14159f;	
double	Real	8 bytes	0	double velocidadDeLuz = 299792.458;	

Declaración de variables


```
tipo nombreDeVariable;
```

double precio; ⇒ Declaración

precio = 2.50; ⇒ Inicialización

double precio = 2.50; ⇒ Todo en una línea

¡Hay que inicializar una variables antes de usarla!

Rango


```
byte a = 127;
a = a + 1;
```

¿Cuánto vale a?

byte → [-128 - 127] → a = -128

Ejercicio

Declara una variable de cada tipo de dato.

A continuación inicializa cada una a un valor de tu elección.

Después imprime su valor por consola.

Finalmente intenta desbordar el tamaño de alguna de las variables.

Constantes

final tipo nombreDeVariable;

final double PI = 3.141592;

No se puede cambiar el valor una vez inicializada

Comentarios

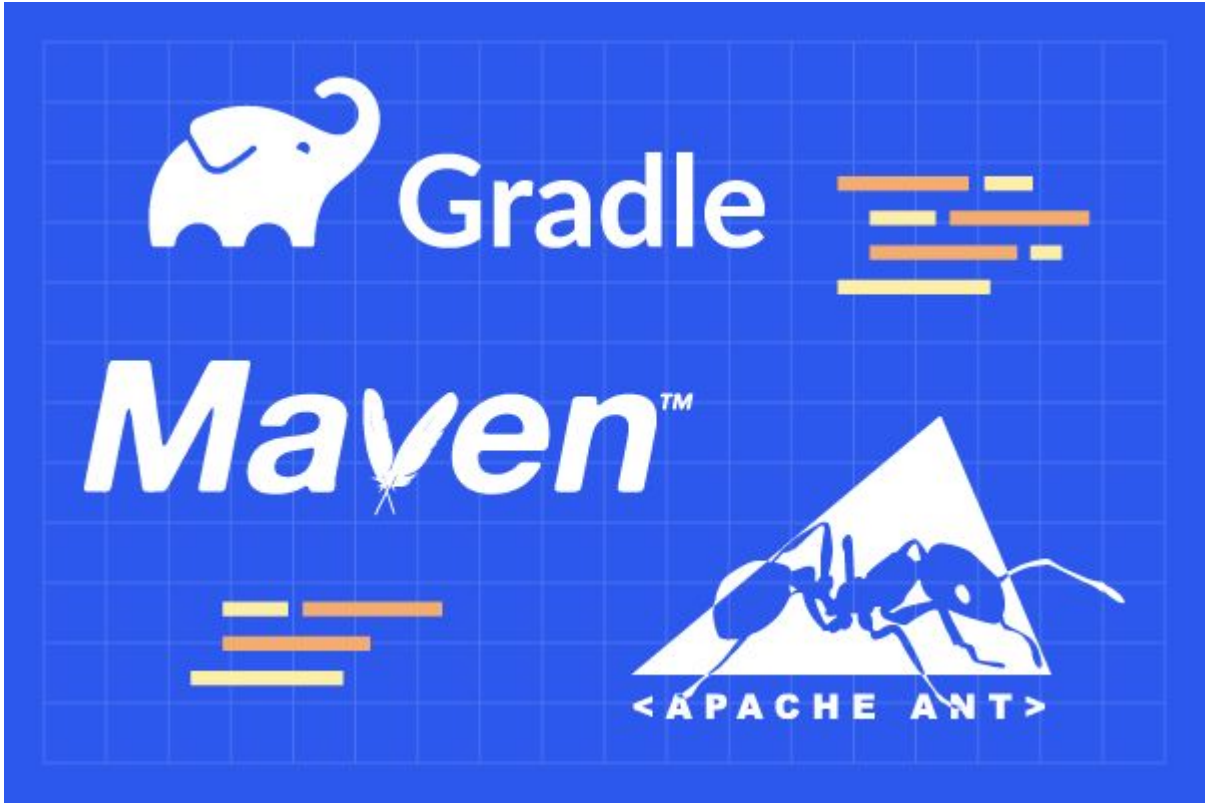
Comentario multilínea: cualquier texto incluido entre los caracteres `/*` (apertura de comentario) y `*/` (cierre de comentario) será interpretado como un comentario, y puede extenderse a través de varias líneas.

Comentario hasta final de línea: todo lo que sigue a los caracteres `//` hasta el final de la línea se considera un comentario.

```
/* esto es un comentario
que se extiende
durante varias líneas */
int numeroPaginas; //declaramos la variable numeroPaginas como un entero
/** este comentario será utilizado en caso de utilizar una herramienta
de generación automática de documentación */
```

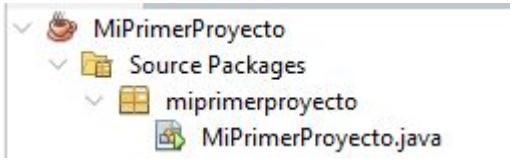
Entornos de desarrollo Integrado (IDE)

Project Management



Paquetes

Los ficheros en Java se organizan en paquetes (como si fueran carpetas)




```
package miprimerproyecto;

public class MiPrimerProyecto {
    public static void main(String[] ar
```

Si no se indica ningún paquete se crea uno por defecto llamado “default”

Paquetes

Para usar el contenido de un paquete en nuestro código, hay que usar el nombre cualificado:

```
java.time.LocalTime queHoraEs = java.time.LocalTime.now();
```

O bien importar el paquete y ahorrarnos tener que poner el nombre cualificado delante:

```
import java.time.LocalTime;
LocalTime queHoraEs = LocalTime.now();
```

```
import java.time.*; //importa todas las clases del paquete java.time
```

Paquetes

El compilador siempre importa por defecto el paquete **java.lang**

```
int x = -3;
int valorAbsoluto = Math.abs(x);
System.out.println("Valor absoluto de " + x + " = " + valorAbsoluto);
```

¿En qué paquete estará System?

API de Java

Application Programming Interface

Librería con todas la funcionalidades que incorpora Java

Se organiza en módulos que son conjuntos de paquetes

Podemos verlos en la documentación de Oracle o inspeccionando el JDK en nuestro IDE

¿En qué paquete estará System?

Vamos a buscarlo en la documentación…

Salida y Entrada de datos

Salida por consola:

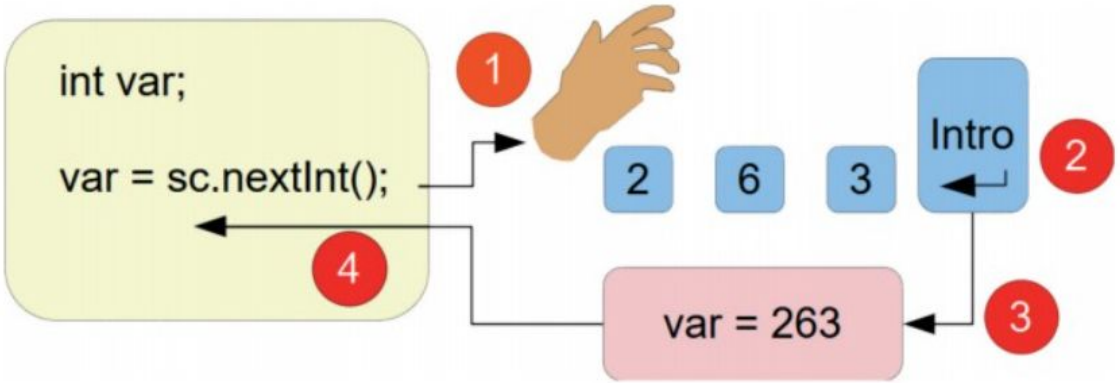
```
System.out.println("Hola."); //inserta un retorno de carro
System.out.print("La hora\ndel sistema es:\n"); //dos retornos de carro
System.out.print(queHoraEs);
```

```
Hola.
La hora
del sistema es:
10:20:32.294
```

```
java.time.LocalTime queHoraEs = java.time.LocalTime.now();
```

Salida y Entrada de datos

Clase Scanner




```
Scanner sc = new Scanner(System.in); //crea el nuevo escáner
double numero; //declaramos la variable numero
numero = sc.nextDouble(); //se detiene la ejecución del programa hasta que
//escribimos un número en el área de Output y pulsamos Intro.
//ahora disponemos del valor introducido, a través de la variable numero
System.out.println("Ha escrito: " + numero);
```

Ejercicio 1

Diseñar un programa que pida un número al usuario por teclado y a continuación lo muestre por consola.

```
Introduzca un número:
5
El número es: 5
BUILD SUCCESSFUL (total time: 5 seconds)
```

Operadores aritméticos

Símbolo	Descripción
+	Suma
+	Más unario: positivo
–	Resta
–	Menor unario: negativo
*	Multiplicación
/	División
%	Módulo
++	Incremento en 1
––	Decremento en 1

```
a ++;
b --;
```

es equivalente a:

```
a = a + 1;
b = b - 1;
```

Ejercicio 2

Escribe un programa que pida la edad al usuario y muestre la edad que tendrá el año que viene

Operadores


```
int a, b, c; //declaramos las variables de tipo entero
a = 1; //a la variable "a" le asignamos 1
b = a ++; //primero asignamos el valor de "a" a "b", y después incrementamos "a"
c = ++ a; //primero incrementamos "a", y después asignamos su valor a "c"
```

```
int a, b, c;
a = 1;
b = a++;
c = ++a;
```

a → 3 b → 1 c → 3

¿Cuánto valen a, b y c?

```
double a = 1.0/10.0 + 2.0/10.0; //el resultado debería ser 3/10
a = a * 10.0; //el resultado debería ser 3
System.out.println(a); //muestra 3.000000000000004
```

Errores de precisión

Ejercicio 3

Escribir una aplicación que pida el año actual y el de nacimiento del usuario. Debe calcular su edad, suponiendo que en el año en curso el usuario ya ha cumplido años.

Ejercicio 4

El tipo short permite almacenar valores comprendidos entre -32768 y 32767. Escribir un programa que compruebe que el rango de valores de un tipo se comporta de forma cíclica, es decir, el valor siguiente al máximo es el valor mínimo.

Ejercicio 5

Crear una aplicación que calcule la media aritmética de dos notas enteras. Hay que tener en cuenta que la media puede contener decimales.

Ejercicio 6

Diseñar una aplicación que calcule la longitud y el área de una circunferencia. Para ello, el usuario debe introducir el radio (que puede contener decimales).

Recordamos:

$$\text{Longitud} = 2 * \pi * \text{radio}$$
$$\text{Área} = \pi * \text{radio2}$$

Declara PI como una constante.

Alternativamente usa la clase Math de Java.

Operadores relacionales

Símbolo	Des
==	Igual que
!=	Distinto que
<	Menor que
<=	Menor o igual que
>	Mayor que
>=	Mayor o igual que

Sólo devuelven true o false

Ejercicio 7

Realizar una aplicación que solicite al usuario su edad y le indique si es mayor de edad (mediante un literal booleano: true o false).

Operadores lógicos

Símbolo	Descripción
&&	Operador and: Y
||	Operador or: O
!	Operador not: negación

También devuelven true o false, pero además sus operandos también son booleanos.

Operadores lógicos

a	b	a && b
falso	falso	falso
cierto	falso	falso
falso	cierto	falso
cierto	cierto	cierto

a	b	a || b
falso	falso	falso
cierto	falso	cierto
falso	cierto	cierto
cierto	cierto	cierto

a	!a
falso	cierto
cierto	falso

Ejercicio 8

¿Cuánto vale cada una de estas expresiones?

`3 <= 5 && 2 == 2` → true

`3 <= 5 && 2 >10` → false

`1 != 2 || 5 < 3` → true

`!(1 < 2)` → false

true || (cualquier condición) → true
false && (cualquier condición) → false

Evaluación perezosa

Ejercicio 9

Diseñar un algoritmo que nos indique si podemos salir a la calle.

Existen aspectos que influyen en esta decisión: si está lloviendo y si hemos terminado nuestras tareas.

Solo podremos salir a la calle si no está lloviendo y hemos finalizado nuestras tareas. Existe una opción en la que, indistintamente de lo anterior, podremos salir a la calle: el hecho de que tengamos que ir a la biblioteca (para realizar algún trabajo, entregar un libro, etc.).

Solicitar al usuario (mediante un booleano) si llueve, si ha finalizado las tareas y si necesita ir a la biblioteca. El algoritmo debe mostrar mediante un booleano (true o false) si es posible que se le otorgue permiso para ir a la calle.

Operadores opera y asigna

Símbolo	Descripción
+=	Suma y asigna
-=	Resta y asigna
*=	Multi
/=	Divi
%=	Mód

```
x *= 2;
```

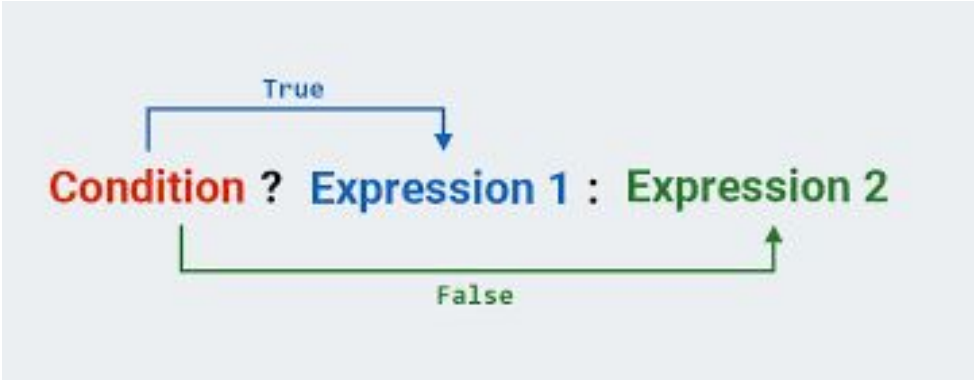
es equivalente a:

```
x = x * 2;
```

Ejercicio 10

Un frutero necesita calcular los beneficios anuales que obtiene de la venta de manzanas y peras. Por este motivo, es necesario diseñar una aplicación que solicite las ventas (en kilos) de cada semestre para cada fruta. La aplicación mostrará el importe total sabiendo que el precio del kilo de manzanas está fijado en 2,35 € y el kilo de peras en 1,95 €.

Operador ternario




```
int a, b;
a = 3 < 5 ? 1 : -1; // 3 < 5 es cierto: a toma el valor 1
b = a == 7 ? 10 : 20; // a (que vale 1) == 7 es falso: b toma el valor 20
```

Ejercicio 11

Escribir un programa que pida dos números y muestre el mayor de ellos.

Hazlo usando el operador ternario

Ejercicio 12

Escribir un programa que pida un valor al usuario y muestre su valor absoluto.

Primero hazlo con la función de Math y luego usando el operador ternario

Precedencia de operadores

Descripción	Operador
Posfijos	`expr++ expr--`
Unarios prefijos	`++expr --expr +expr -expr !expr`
Aritméticos	`* / %`
Aritméticos	`+-`
Relacionales	`< <= > >=`
Comparación	`== !=`
AND lógico	`&&`
OR lógico	`||`
Ternario	`? :`
Asignación	`= += -= *= /= %= &= ^=`

Ejercicio 13

1. Escribe un programa en Java que evalúe las siguientes expresiones.
2. Muestra el resultado de cada expresión y explica qué operadores tienen mayor precedencia.
3. Usa operadores aritméticos, relacionales, lógicos, y operadores de asignación.

Expresiones:

1. `10 + 5 * 2 > 20 && 4 == 4`
2. `!(7 + 3 > 10) || 3 * 2 <= 6`
3. `10 / 2 + 3 * 5 == 19 && true`
4. `int x = 5; x += 3 * 2;`
5. `boolean b = false; b = !b || 7 % 2 == 1;`

Conversión de tipos

Toda variable tiene un tipo y sólo admite valores de ese tipo

```
int a = 1;
double x = 2.3;
```

Sin embargo, podemos realizar conversiones entre tipos de dos formas:

- Implícita: de un tipo de menor capacidad a uno de mayor
- Explícita: de un tipo de mayor capacidad a uno de menor ⇒ hay pérdida de información

`int a = 2.3` ⇒ Error de compilación
Asigna un real a una variable entera

Conversión de tipos

Conversión implícita

```
int a = 1;
double x = a;
```

La conversión de tipos también se conoce como 'Casting'

Conversión explícita

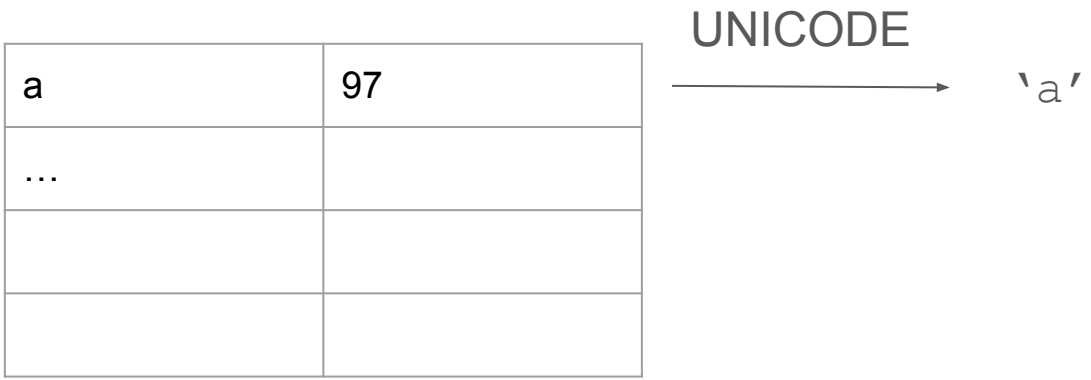
```
double x = 2.3;
int a = (int)x;
```

⇒ hay pérdida de información pero no da error

Conversión de tipos

¿Qué tipo de conversión está ocurriendo aquí?

```
int a = 'a';
```



Ejercicio 14

Escribir un programa que solicite las notas del primer, segundo y tercer trimestre (notas enteras que se solicitarán al usuario).

El programa debe mostrar la nota media del curso como se utiliza en el boletín de calificaciones (solo la parte entera) y como se usa en el expediente académico (con decimales).

Ejercicio 15

Realizar un programa que pida como entrada un número decimal y lo muestre redondeado al entero más próximo.

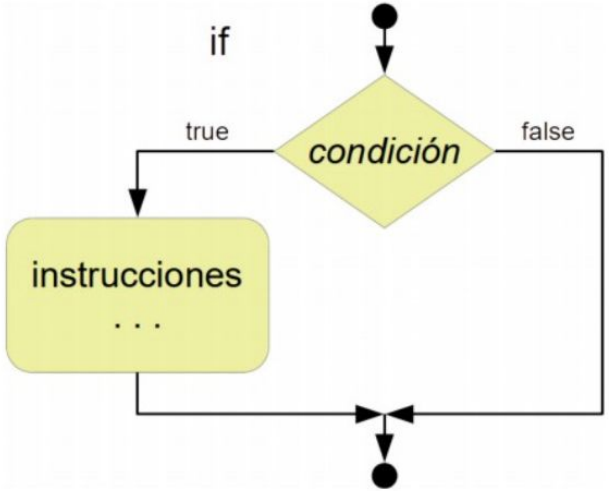
Condicionales

Condicional simple (if)

Condicional doble (if-else)

Condicional simple (switch)

Condicional simple




```
if (condición) {
    bloque de instrucciones
    ...
}
```

Condicional simple

Ejercicio 16

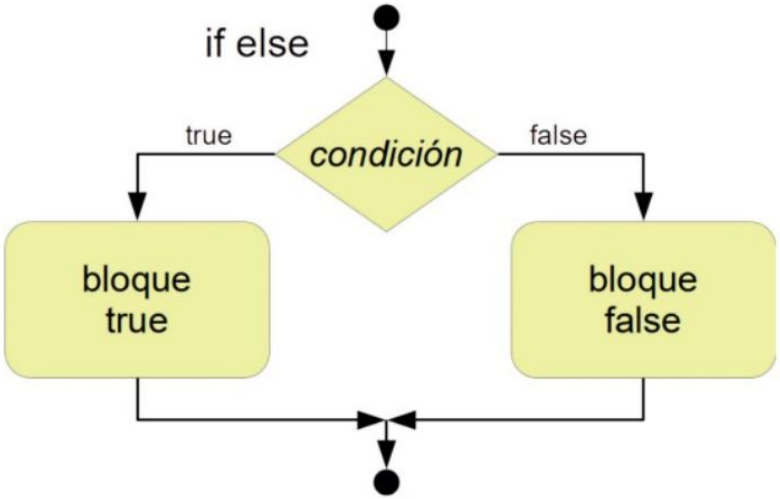
Escribe un programa que pida un número entero al usuario y compruebe si el número está comprendido en el rango: [a, b], donde a y b son números aleatorios.

Si el número está fuera de rango no se realizará ninguna acción.

Si el número está dentro del rango, se mostrarán cuáles eran los valores de a y b.

En todos los casos, tanto si está en rango como fuera de él, mostrará al final del programa cuál fue el número introducido.

Condicional doble




```
if (condición) {
   bloque true //se ejecuta cuando la condición es cierta
} else {
   bloque false //se ejecuta cuando la condición es falsa
}
```

Condicional doble

Ejercicio 17

Pedir dos números enteros y decir si son iguales o no

Modificar el programa para que también diga cual es el mayor

Condicional doble

Ejercicio 18

Implementar un programa que pida por teclado un número decimal e indique si es un número casi-cero, que son aquellos, positivos o negativos, que se acercan a 0 por menos de 1 unidad, aunque curiosamente el 0 no se considera un número casi-cero. Ejemplos de números casi-cero son el 0,3, el -0,99 o el 0,123; algunos números que no se consideran casi-ceros son: el 12,3, el 0 o el -1.

If-else de una línea


```
if (condición)
        Instrucción;
else
        Instrucción;
```

No es necesario poner los bloques { } cuando sólo hay una instrucción dentro pero sí recomendable por legibilidad

Anidamiento


```
if (edad >= 0) { // Verificamos que la edad sea válida
        if (edad < 13) {
            System.out.println("Es un niño.");
        } else if (edad < 18) {
            System.out.println("Es un adolescente.");
        } else if (edad < 65) {
            System.out.println("Es un adulto.");
        } else {
            System.out.println("Es una persona mayor.");
        }
} else {
        System.out.println("Edad no válida.");
}
```

```
if (edad >= 0) { // Verificamos que la edad sea válida
        if (edad < 13) {
            System.out.println("Es un niño.");
        } else {
            if (edad < 18) {
                System.out.println("Es un adolescente
            } else {
                if (edad < 65) {
                    System.out.println("Es un adulto.")
                } else {
                    System.out.println("Es una person
        }
} else {
        System.out.println("Edad no válida.");
}
```

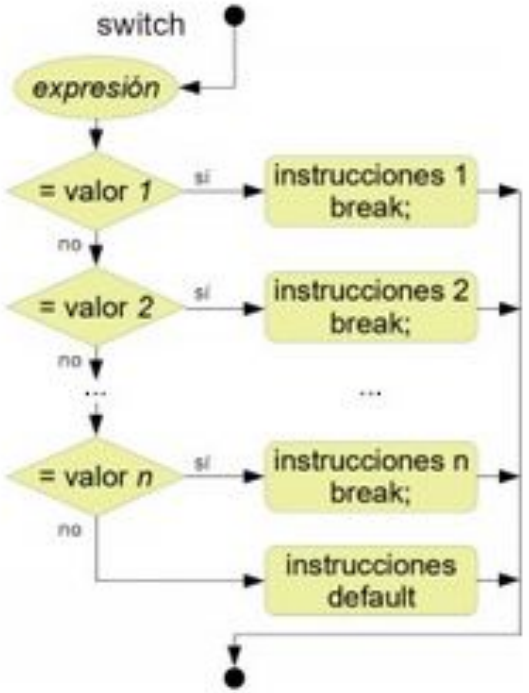
Condicional doble

Ejercicio 19

Pedir tres números y mostrarlos ordenados de mayor a menor

Condicional múltiple


```
switch (expresión) {
case valor1:
    conjunto instrucción 1;
    break;
case valor2:
    conjunto instrucción 2;
    break;
...
case valorN:
    conjunto instrucción N;
    break;
default:
    conjunto instrucción en otro caso;
}
```



Condicional múltiple

Ejemplo

```
int a = 1;
switch (a) {
    case 1:
        System.out.println("primero");
        break;
    case 2:
        System.out.println("segundo");
        break;
    case 3:
        System.out.println("tercero");
        break;
    default:
        System.out.println("otros");
}
```

¿Qué imprimirá? Vamos a probarlo…

¿Qué imprimir si quito el break del caso 2?

¿Cómo puedo agrupar varios casos en uno?

Condicional múltiple

Switch como expresión (en lugar de como sentencia)

```
int dia = 5;
 String diaDeLaSemana = switch (dia) {
     case 1 -> "Lunes";
     case 2 -> "Martes";
     case 3 -> "Miércoles";
     case 4 -> "Jueves";
     case 5 -> "Viernes";
     case 6 -> "Sábado";
     case 7 -> "Domingo";
     default -> "Número inválido";
 };
 System.out.println("El día de la semana es: " + diaDeLaSemana);
```

Ejercicio 20

Pedir una nota entera de 0 a 10 y mostrarla de la siguiente forma: insuficiente (de 0 a 4), suficiente (5), bien (6), notable (7 y 8) y sobresaliente (9 y 10).

Hacerlo primero usando la versión tradicional (sentencia)

Ampliación: después repetirlo usando la versión de expresión

Condicional múltiple (switch)

Ejercicio 21

Pedir el número del mes en el año y decir cuántos días tiene ese mes

Condicional múltiple

Switch como expresión (en lugar de como sentencia) y además en lugar de devolver una expresión sencilla puede hacerse un cálculo en un bloque → yield

```
int mes = 3;
int dias = switch (mes) {
    case 4, 6, 9, 11 -> 30;
    case 2 -> {
        int anio = 2023;
        boolean esBisiesto = (anio % 4 == 0 && anio % 100 != 0) || anio % 400 == 0;
        yield esBisiesto ? 29 : 28;
    }
    default -> 31;
};
System.out.println("El mes tiene " + dias + " días");
```

Bucles

- ¿Qué es una Iteración?
- ¿Para qué se usan los bucles?

Bucles

1. while
2. do while

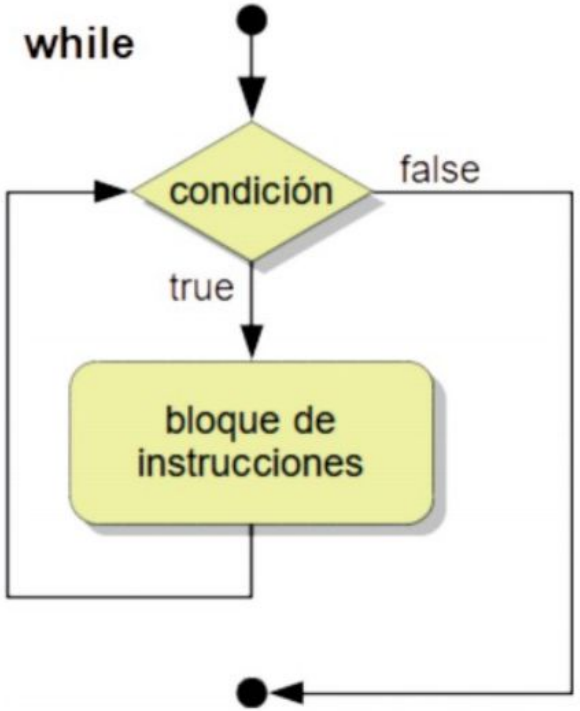
Controlados por condición

3. for

Controlado por contador

while


```
while (condición) {
    bloque de instrucciones
    ...
}
```



While

Ejercicio 22

Diseñar un programa que muestre, para cada número introducido por teclado, si es par, si es positivo y su cuadrado. El proceso se repetirá hasta que el número introducido sea 0.

```
while (condición) {
    bloque de instrucciones
    ...
}
```

While

Ejercicio 23

Modificar el ejercicio 16 para que el programa no finalice hasta que el usuario acierte con un número dentro del rango.

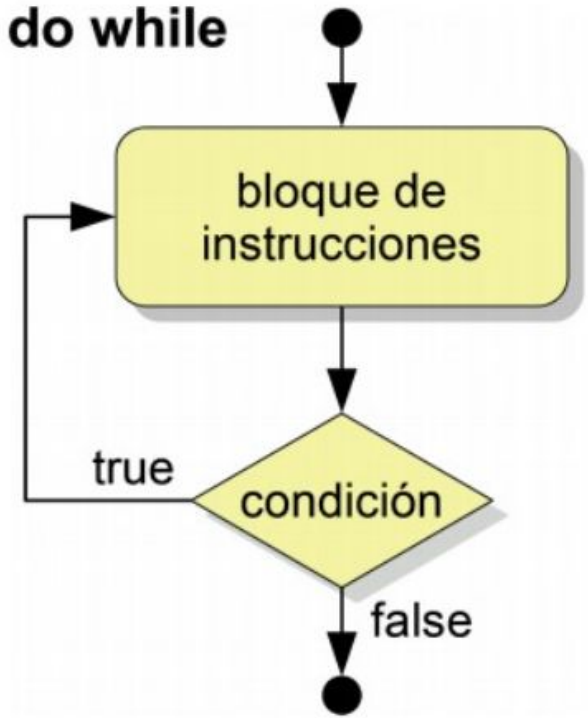
While

Ejercicio 24

Diseña una aplicación que muestre la edad máxima y mínima de un grupo de alumnos. El usuario introducirá las edades y terminará escribiendo un -1.

do while


```
do {
    bloque de instrucciones
    ...
} while (condición);
```



do-while

Ejercicio 25

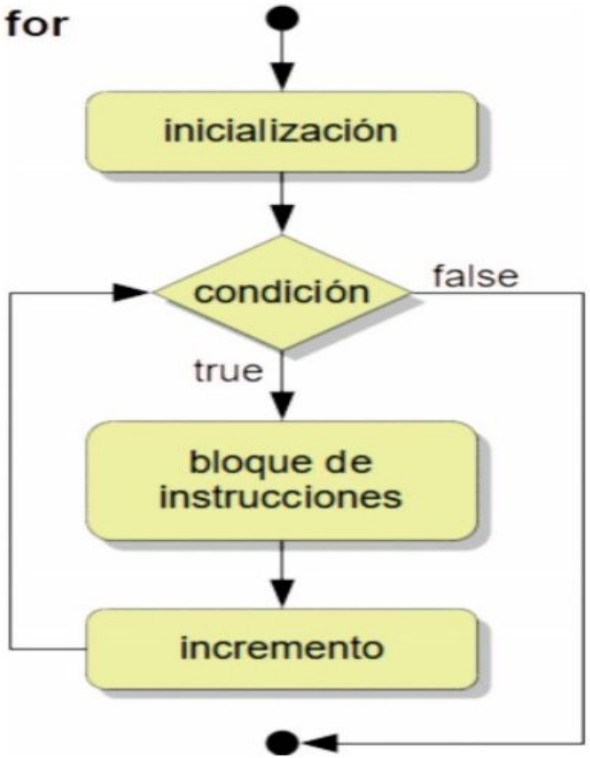
Desarrollar un juego que ayude a mejorar el cálculo mental de la suma.

El jugador tendrá que introducir la solución de la suma de dos números aleatorios comprendidos entre 1 y 100.

Mientras la solución introducida sea correcta, el juego continuará. En caso contrario, el programa terminará y mostrará el número de operaciones realizadas correctamente.

for


```
for (inicialización; condición; incremento) {
    bloque de instrucciones
    ...
}
```



Ejercicio 26

Implementa la aplicación Eco, que pide al usuario un número y muestra en pantalla la salida:

Eco...
Eco...
Eco…

Se muestra «Eco...» tantas veces como indique el número introducido. La salida anterior sería para el número 3.

for

Ejercicio 27

Escribir todos los múltiplos de 7 menores que 100

Ejercicio 28

Pedir un número y calcular su factorial.

Por ejemplo, el factorial de 5 se denota 5! y es igual a 5x4×3×2×1= 120.

Salidas anticipadas

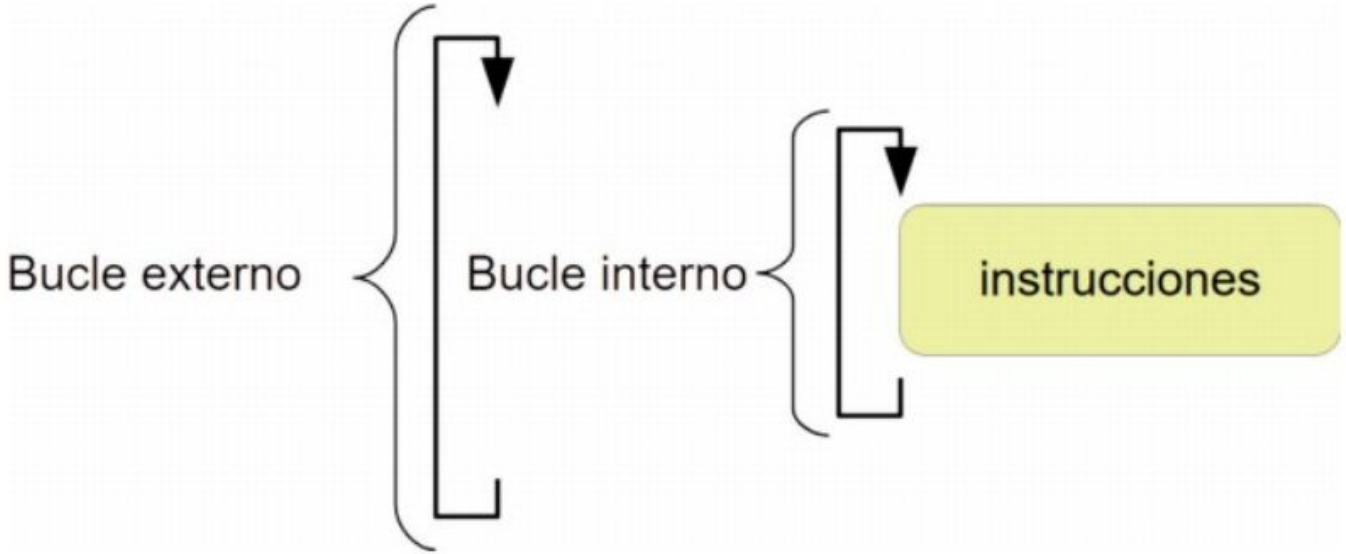
No se recomienda su uso.

- `break`: interrumpe completamente la ejecución del bucle.
- `continue`: detiene la iteración actual y continúa con la siguiente.

```
i = 1;
while (i <= 10) {
   System.out.println("La i vale" + i);
   if (i == 2) {
      break;
   }
   i++;
}
```

```
i = 0;
while (i < 10) {
   i++;
   if (i % 2 == 0) { //si i es par
      continue;
   }
   System.out.println("La i vale " + i);
}
```

Bucles anidados



- Independientes
- Dependientes

Bucles anidados independientes

La variable del bucle exterior **no** interviene en el bucle interior

```
for (i = 1; i <= 4; i++) {
    for (j = 1; j <= 3; j++) {
        System.out.println("Ejecutando...");
    }
}
```

¿Cuántas veces se ejecuta la instrucción?

Bucles anidados independientes

Ejercicio 29

Diseñar una aplicación que muestre las tablas de multiplicar del 1 al 10

Tabla del 1	Tabla del 2	Tabla del 3	Tabla del 4	Tabla del 5
1 × 0 = 0	2 × 0 = 0	3 × 0 = 0	4 × 0 = 0	5 × 0 = 0
1 × 1 = 1	2 × 1 = 2	3 × 1 = 3	4 × 1 = 4	5 × 1 = 5
1 × 2 = 2	2 × 2 = 4	3 × 2 = 6	4 × 2 = 8	5 × 2 = 10
1 × 3 = 3	2 × 3 = 6	3 × 3 = 9	4 × 3 = 12	5 × 3 = 15
1 × 4 = 4	2 × 4 = 8	3 × 4 = 12	4 × 4 = 16	5 × 4 = 20
1 × 5 = 5	2 × 5 = 10	3 × 5 = 15	4 × 5 = 20	5 × 5 = 25
1 × 6 = 6	2 × 6 = 12	3 × 6 = 18	4 × 6 = 24	5 × 6 = 30
1 × 7 = 7	2 × 7 = 14	3 × 7 = 21	4 × 7 = 28	5 × 7 = 35
1 × 8 = 8	2 × 8 = 16	3 × 8 = 24	4 × 8 = 32	5 × 8 = 40
1 × 9 = 9	2 × 9 = 18	3 × 9 = 27	4 × 9 = 36	5 × 9 = 45
1 × 10 = 10	2 × 10 = 20	3 × 10 = 30	4 × 10 = 40	5 × 10 = 50

Tabla del 6	Tabla del 7	Tabla del 8	Tabla del 9	Tabla del 10
6 × 0 = 0	7 × 0 = 0	8 × 0 = 0	9 × 0 = 0	10 × 0 = 0
6 × 1 = 6	7 × 1 = 7	8 × 1 = 8	9 × 1 = 9	10 × 1 = 10
6 × 2 = 12	7 × 2 = 14	8 × 2 = 16	9 × 2 = 18	10 × 2 = 20
6 × 3 = 18	7 × 3 = 21	8 × 3 = 24	9 × 3 = 27	10 × 3 = 30
6 × 4 = 24	7 × 4 = 28	8 × 4 = 32	9 × 4 = 36	10 × 4 = 40
6 × 5 = 30	7 × 5 = 35	8 × 5 = 40	9 × 5 = 45	10 × 5 = 50
6 × 6 = 36	7 × 6 = 42	8 × 6 = 48	9 × 6 = 54	10 × 6 = 60
6 × 7 = 42	7 × 7 = 49	8 × 7 = 56	9 × 7 = 63	10 × 7 = 70
6 × 8 = 48	7 × 8 = 56	8 × 8 = 64	9 × 8 = 72	10 × 8 = 80
6 × 9 = 54	7 × 9 = 63	8 × 9 = 72	9 × 9 = 81	10 × 9 = 90
6 × 10 = 60	7 × 10 = 70	8 × 10 = 80	9 × 10 = 90	10 × 10 = 100

Bucles anidados dependientes

El bucle interno depende de alguna variable del bucle externo

```
for (i = 1; i <= 3; i++) {
   System.out.println("Bucle externo, i=" +i);
   j = 1;
   while (j <= i) {
      System.out.println("...Bucle interno, j=" + j);
      j++;
   }
}
```

Realiza la traza de ejecución ¿qué se mostrará por consola?

Ejercicio 30

Pedir por consola un número n y dibujar un triángulo rectángulo de n elementos de lado, utilizando para ello asteriscos (*). Por ejemplo para n = 4:

```
* * * *
* * *
* *
*
```

Funciones: introducción

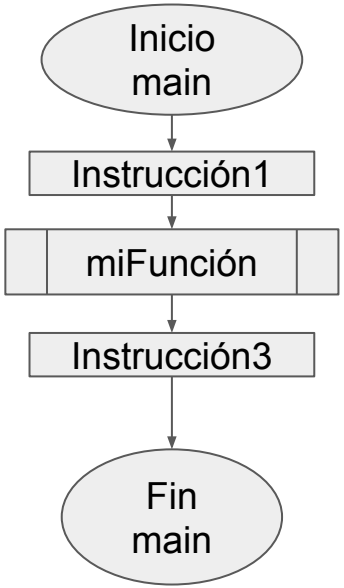

```java
public class Tienda {
    public static void main(String[] args) {
        // Datos para tres productos
        double precioProducto1 = 50.0;
        double precioProducto2 = 75.0;
        double precioProducto3 = 100.0;

        boolean esMiembro = true; // Descuento solo para miembros

        // Producto 1
        double descuentoProducto1 = esMiembro ? precioProducto1 * 0.1 : 0; // 10% de descuento
        double precioFinalProducto1 = precioProducto1 - descuentoProducto1;
        System.out.println("Precio final del Producto 1: " + precioFinalProducto1);

        // Producto 2
        double descuentoProducto2 = esMiembro ? precioProducto2 * 0.1 : 0;
        double precioFinalProducto2 = precioProducto2 - descuentoProducto2;
        System.out.println("Precio final del Producto 2: " + precioFinalProducto2);

        // Producto 3
        double descuentoProducto3 = esMiembro ? precioProducto3 * 0.1 : 0;
        double precioFinalProducto3 = precioProducto3 - descuentoProducto3;
        System.out.println("Precio final del Producto 3: " + precioFinalProducto3);
    }
}
```

calcularPrecioConDescuento

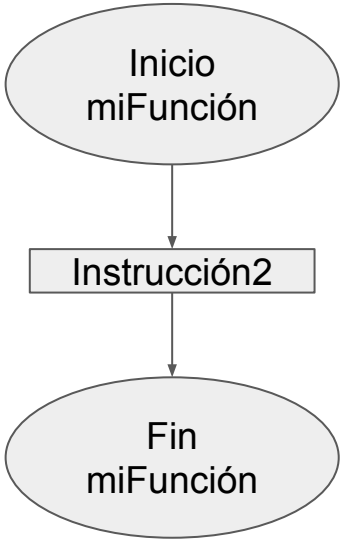
Funciones: introducción


```java
public class Tienda {

    // Función para calcular el precio final con descuento
    public static double calcularPrecioConDescuento(double precio, boolean esMiembro) {
        double descuento = esMiembro ? precio * 0.1 : 0;
        return precio - descuento;
    }

    public static void main(String[] args) {
        // Datos para tres productos
        double precioProducto1 = 50.0;
        double precioProducto2 = 75.0;
        double precioProducto3 = 100.0;

        boolean esMiembro = true;

        // Llamadas a la función para calcular el precio final de cada producto
        double precioFinalProducto1 = calcularPrecioConDescuento(precioProducto1, esMiembro);
        System.out.println("Precio final del Producto 1: " + precioFinalProducto1);

        double precioFinalProducto2 = calcularPrecioConDescuento(precioProducto2, esMiembro);
        System.out.println("Precio final del Producto 2: " + precioFinalProducto2);

        double precioFinalProducto3 = calcularPrecioConDescuento(precioProducto3, esMiembro);
        System.out.println("Precio final del Producto 3: " + precioFinalProducto3);
    }
}
```

Funciones: introducción



Programa principal



Definición de la función

Funciones: ¿qué es una función?

Definición:
Una función es un bloque de código independiente y reutilizable que realiza una tarea específica.

Características de una Función:

- **Nombre:** Identifica la función dentro del programa.
- **Parámetros:** Valores que se pasan a la función para que opere.
- **Valor de Retorno:** El resultado que devuelve la función después de ejecutarse.
- **Reutilización:** Se puede llamar a la función múltiples veces.

Ventajas de Usar Funciones:

- Organización del código en bloques lógicos.
- Reducción de código repetitivo.
- Mejora en la legibilidad y mantenimiento del programa.

Programación estructurada → funciones y procedimientos

POO → métodos

Funciones: sintaxis

Declaración o definición de una función:

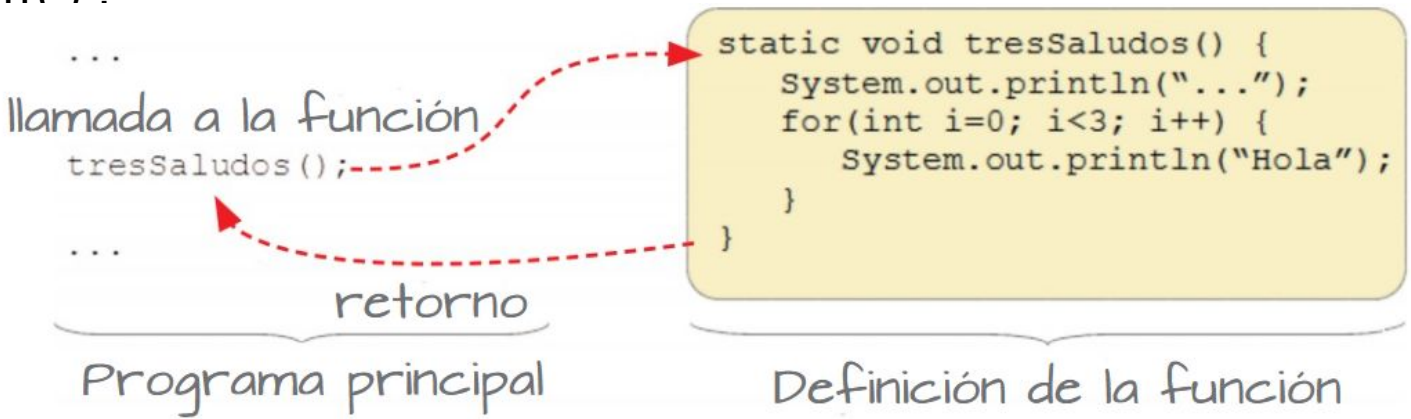
Prototipo o cabecera

```
private static tipo nombreFunción(tipoParam param,...) {
    instrucciones
}
```

Cuerpo

Invocación de una función:

```
nombreFunción();
```



Funciones: ejemplo


```java
public class EjemploFuncion {
    // Función que calcula el doble de un número
    public static int calcularDoble(int numero) {
        return numero * 2;
    }

    public static void main(String[] args) {
        int resultado = calcularDoble(5);  // Llama a la función con el valor 5
        System.out.println("El doble de 5 es: " + resultado);
    }
}
```

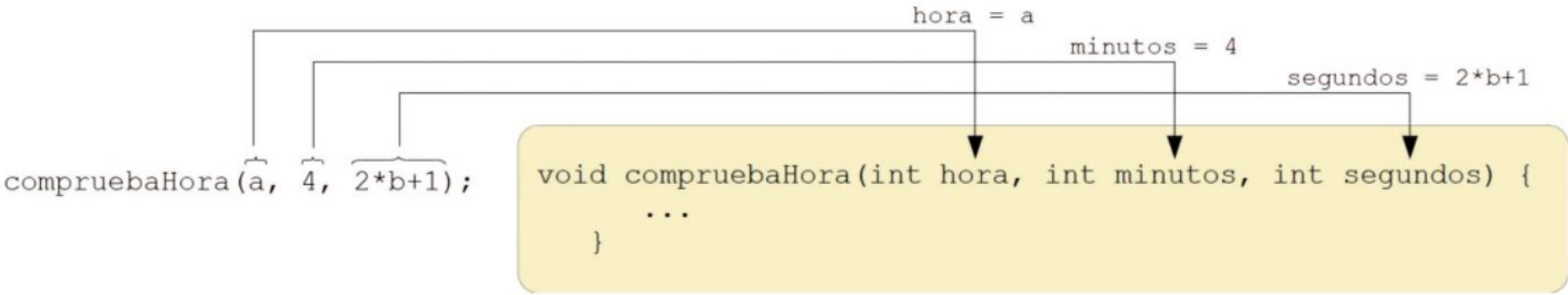
La definición de una función puede hacerse **antes o después** del main

Funciones: parámetros y argumentos


```java
public class Calculadora {
    public static int sumar(int a, int b) { // 'a' y 'b' son parámetros
        return a + b;
    }
}

public class Ejemplo {
    public static void main(String[] args) {
        int resultado = Calculadora.sumar(5, 10); // 5 y 10 son argumentos
        System.out.println("Resultado: " + resultado);
    }
}
```

Los parámetros pueden ser: literales, variables o expresiones.
Ejemplo: sumar(5, 10); sumar(n, m); sumar(5, 2*n);

Funciones: paso de parámetros



En Java los parámetros toman su valor como una copia del valor de la expresión o variable utilizada en la llamada; este mecanismo de paso de parámetros se denomina **paso de parámetros por valor** o **por copia.**

Además, las variables de la función son **locales** y tienen el ámbito del bloque de la función, por tanto los cambios en esas variables dentro de la función no se reflejan fuera.

Funciones: paso de parámetros


```java
public class PasoPorValor {

    // Método que intenta cambiar el valor de un número
    public static void cambiarValor(int numero) {
        numero = 20; // Cambiamos el valor de 'numero' dentro del método
        System.out.println("Dentro de cambiarValor: " + numero); // Imprime 20
    }

    public static void main(String[] args) {
        int numeroOriginal = 10; // Valor inicial de la variable
         // Imprime 10
        System.out.println("Antes de llamar a cambiarValor: " + numeroOriginal);
        // Llamada al método cambiarValor
        cambiarValor(numeroOriginal);
        // Verificamos si el valor de numeroOriginal cambió
        System.out.println("Después de llamar a cambiarValor: " + numeroOriginal);
         // Imprime 10
    }
}
```

Funciones: ejercicios

Ejercicio 31

Crea una función de nombre variosSaludos que reciba como parámetro un entero n y no devuelva nada, imprimiendo la palabra “Hola” tantas veces como indique n.

Invoca la función con el valor literal 3

Invocala ahora mediante una variable leída por teclado

Finalmente, invocala con el valor doble de n

Funciones: devuelve un valor

- Hasta ahora: procedimientos → devolviamos ***void***
- Ahora: funciones → devolvemos un valor

```
tipo nombreFunción(parámetros) {
    ...
    return (valor);
}
```

Funciones: devolver valor: ejemplo


```
static int suma(int x, int y) { //cada llamada devuelve un int
   int resultado;
   resultado = x + y;
   return(resultado); //sustituye la llamada por el valor de resultado
}

  int a = suma(2, 3);
  int b = suma(7, 1) * 5;
```

Ejercicio 32

Diseñar una función que recibe como parámetros dos números enteros y devuelve el máximo de ambos.

Imprimir el valor devuelto por la función dentro del código de la función main.

Funciones: sobrecarga

Funciones con el mismo nombre pero distinto número o tipo de parámetros

```
//función sobrecargada
static int suma(int a, int b) {
   int suma;
   suma = a + b;
   return(suma);
}
//función sobrecargada
static double suma(int a, double pesoA, int b, double pesoB) {
   double suma;
   suma = a * pesoA / (pesoA + pesoB) + b * pesoB / (pesoA + pesoB);
   return(suma);
}
```

Ejercicio 33

Repetir el ejercicio anterior pero ahora para calcular el máximo de tres números.

Se debe sobrecargar la función, por lo que ambas funciones deben estar en el mismo archivo y tener el mismo identificador.

Opcional: dentro de esta función utiliza la función del ejercicio 32 para realizar los cálculos.

Ejercicio 34

Un centro de investigación de la flora urbana necesita una aplicación que muestre cuál es el árbol más alto. Para ello se introducirá por teclado la altura (en centímetros) de cada árbol (terminando la introducción de datos cuando se utilice -1 como altura). Los árboles se identifican mediante etiquetas con números únicos correlativos, comenzando en 0. Diseñar una aplicación que resuelva el problema planteado.