



Cofinanciado por
la Unión Europea



MINISTERIO
DE TRABAJO
Y ECONOMÍA SOCIAL



Fondos Europeos



Junta de Andalucía
Instituto Andaluz de
Investigación y
Difusión del
Empleo



Ayuda
en Acción
Impulsa Empleo Joven

Sintaxis Java

Índice:

- 1.Comentarios
- 2.Tipos de datos primitivos
- 3.Variables
- 4.Caracteres especiales
- 5.Operadores
- 6.Entrada y salida de datos en consola
- 7.Control de flujo
- 8.Strings en Java
- 9.Arrays en Java
- 10.Funciones
11. Packages
- 12.Tipos enumerados
- 13.Casting
- 14.Excepciones



REGISTRO NACIONAL DE ASOCIACIONES N°611922
DECLARADA ENTIDAD DE UTILIDAD PÚBLICA ESTATAL
AGENCIA DE COLOCACIÓN: ID 0100000017

CENTRO EN MÁLAGA
C/DOS ACERAS 23, 29012
MÁLAGA | (+34) 952 300 500
ARRABAL@ARRABALEMPLEO.ORG

CENTRO EN CÁDIZ
TR.^a ALAMEDA DE SOLANO, 32, 11130
CHICLANA DE LA FRONTERA | (+34) 956 900 312
CHICLANA@ARRABALEMPLEO.ORG



1.Comentarios

- Comentario de una línea → Usa //
- Comentario de múltiples líneas → Usa /* ... */

2.Tipos de datos primitivos

Tipo	Tamaño	Rango de valores	Ejemplo
byte	8 bits	-128 a 127	byte b = 100;
short	16 bits	-32,768 a 32,767	short s = 30000;
int	32 bits	-2 ³¹ a 2 ³¹ -1	int i = 100000;
long	64 bits	-2 ⁶³ a 2 ⁶³ -1	long l = 10000000000L;
float	32 bits	Decimales con precisión simple	float f = 3.14f;
double	64 bits	Decimales con precisión doble	double d = 3.141592;
char	16 bits	Caracteres Unicode	char c = 'A';
boolean	1 bit	true o false	boolean b = true;

Estos tipos de datos **no son objetos** y se almacenan directamente en memoria, lo que los hace más eficientes en términos de rendimiento.

3.VARIABLES

Tipos de variables en Java:

- locales
- de instancia
- de clase

En Java, una **variable** es un espacio en memoria donde se almacena un valor. Para declarar una variable, se sigue esta estructura:

```
tipo nombre = valor;  
  
int edad = 25;
```





Variables locales

- Se declaran dentro de un método o bloque y solo existen en ese ámbito.

```
public void mostrarEdad() {
    int edad = 30; // Solo existe dentro de este método
    System.out.println(edad);
}
```

Variables de instancia (atributos)

- Se declaran dentro de una clase pero fuera de los métodos.
- No necesitan static y pertenecen a cada objeto.

```
class Persona {
    String nombre; // Variable de instancia
}
```

Variables estáticas (de clase)

- Se declaran con static y pertenecen a la clase, no a los objetos.

```
class Persona {
    static int contador = 0; // Variable compartida por todos los objetos
}
```





Cofinanciado por
la Unión Europea



Ejemplo Completo

```

class Persona {
    // Atributos de instancia
    String nombre;

    // Atributo estático
    static int contador = 0;

    // Constructor
    public Persona(String nombre) {
        this.nombre = nombre;
        contador++; // Incrementa el contador estático
    }

    // Método para mostrar la información
    public void mostrarInfo() {
        System.out.println("Nombre: " + nombre);
    }

    // Método estático para mostrar el contador
    public static void mostrarContador() {
        System.out.println("Número de personas creadas: " + contador);
    }
}

public class Main {
    public static void main(String[] args) {
        // Crear objetos
        Persona p1 = new Persona("Juan");
        Persona p2 = new Persona("Maria");

        // Mostrar la información de cada persona
        p1.mostrarInfo(); // Nombre: Juan
        p2.mostrarInfo(); // Nombre: Maria

        // Mostrar el contador estático
        Persona.mostrarContador(); // Número de personas creadas: 2
    }
}

```



REGISTRO NACIONAL DE ASOCIACIONES N°611922
DECLARADA ENTIDAD DE UTILIDAD PÚBLICA ESTATAL
AGENCIA DE COLOCACIÓN: ID 0100000017

CENTRO EN MÁLAGA
C/DOS ACERAS 23, 29012
MÁLAGA | (+34) 952 300 500
ARRABAL@ARRABALEMPLEO.ORG

CENTRO EN CÁDIZ
TR.^a ALAMEDA DE SOLANO, 32, 11130
CHICLANA DE LA FRONTERA | (+34) 956 900 312
CHICLANA@ARRABALEMPLEO.ORG



Explicación:

- **Variables de instancia:** String nombre es una variable que tiene un valor distinto para cada objeto (p1.nombre es "Juan" y p2.nombre es "Maria").
- **Variable estática:** static int contador es compartida entre todos los objetos. Cada vez que se crea un nuevo objeto, contador aumenta. En este caso, después de crear p1 y p2, el valor de contador es 2.

Salida esperada:

```
Nombre: Juan
Nombre: María
Número de personas creadas: 2
```

Reglas para nombrar variables en Java:

Permitido:

- Letras (nombre), números (edad1), _ y \$ (_id, \$precio).
- Deben **comenzar con una letra o _ o \$**, pero no con un número.

No permitido:

- Palabras reservadas (int, class, public, etc.).
- Espacios (mi nombre  → miNombre 

Convención:

- Usar **camelCase**: miVariable, nombrePersona.
- Constantes en **mayúsculas con guiones bajos**: final int MAX_EDAD = 100;.

Nota: En Java, no existen variables **globales** como tal (como en algunos otros lenguajes), pero se pueden simular utilizando **variables de clase (estáticas)**.

```
class Configuracion {
    static String appName = "MiAplicacion"; // Variable de clase (similar a "global")
}

public class Main {
    public static void main(String[] args) {
        System.out.println(Configuracion.appName); // Acceso sin crear una instancia
    }
}
```



REGISTRO NACIONAL DE ASOCIACIONES N°611922
DECLARADA ENTIDAD DE UTILIDAD PÚBLICA ESTATAL
AGENCIA DE COLOCACIÓN: ID 0100000017

CENTRO EN MÁLAGA
C/DOS ACERAS 23, 29012
MÁLAGA | (+34) 952 300 500
ARRABAL@ARRABALEMPLEO.ORG

CENTRO EN CÁDIZ
TR.ª ALAMEDA DE SOLANO, 32, 11130
CHICLANA DE LA FRONTERA | (+34) 956 900 312
CHICLANA@ARRABALEMPLEO.ORG



4. Caracteres especiales

Carácter	Descripción	Ejemplo de uso
\n	Nueva línea (salto de línea)	"Hola\nMundo" -> Hola (salto) Mundo
\t	Tabulador (espacio de tabulación)	"Nombre:\tJuan" -> Nombre: Juan
\\"	Barra invertida (escapa el carácter \)	"Ruta: C:\\Archivos" -> Ruta: C:\\Archivos
'	Comillas simples	"Es el símbolo: \' -> Es el símbolo: '
"	Comillas dobles	"Él dijo: \"Hola\" -> Él dijo: "Hola"
\r	Retorno de carro (carácter de fin de línea en algunas plataformas)	"Hola\rMundo"
\b	Retroceso (backspace)	"Hola\bMundo" -> HolMundo (elimina la "a")
\f	Form feed (salto de página, rara vez usado)	"Texto\fSiguiente"

Ejemplo:

```
System.out.println("Línea 1\nLínea 2");
```

Línea 1
Línea 2



5. Operadores

Operadores Aritméticos (Para operaciones matemáticas)

Operador	Descripción	Ejemplo (a = 10, b = 3)	Resultado
+	Suma	a + b	13
-	Resta	a - b	7
*	Multiplicación	a * b	30
/	División	a / b	3 (división entera)
%	Módulo (resto)	a % b	1

Operadores Relacionales (Comparaciones, devuelven true o false)

Operador	Descripción	Ejemplo (a = 10, b = 3)	Resultado
==	Igualdad	a == b	false
!=	Diferente	a != b	true
>	Mayor que	a > b	true
<	Menor que	a < b	false
>=	Mayor o igual	a >= b	true
<=	Menor o igual	a <= b	false

Operadores Lógicos (Para expresiones booleanas)

Operador	Descripción	Ejemplo (x = true, y = false)	Resultado
&&	AND lógico	x && y	false
`		`	OR lógico
!	NOT lógico	!x	false

Operadores de Asignación (Para asignar valores a variables)

Operador	Descripción	Ejemplo (a = 10)	Equivalente a	Resultado
=	Asignación	a = 5	—	a = 5
+=	Suma y asigna	a += 3	a = a + 3	a = 13
-=	Resta y asigna	a -= 2	a = a - 2	a = 8
*=	Multiplica y asigna	a *= 2	a = a * 2	a = 20
/=	Divide y asigna	a /= 2	a = a / 2	a = 5
%=	Módulo y asigna	a %= 3	a = a % 3	a = 1





Operadores de Incremento y Decremento

Operador	Descripción	Ejemplo (a = 5)	Resultado
++a	Pre-incremento (suma antes)	int b = ++a;	a = 6, b = 6
a++	Post-incremento (suma después)	int b = a++;	a = 6, b = 5
--a	Pre-decremento (resta antes)	int b = --a;	a = 4, b = 4
a--	Post-decremento (resta después)	int b = a--;	a = 4, b = 5

Operador Ternario (Para condiciones simples)

Operador	Descripción	Ejemplo (a = 10, b = 5)	Resultado
? :	Evaluación condicional	int min = (a < b) ? a : b;	min = 5

6. Entrada y salida de datos en consola

Entrada y Salida en Java

En Java, se utilizan diferentes métodos para **mostrar datos en pantalla (salida)** y **pedir datos al usuario (entrada)**.

Salida de datos (mostrar en pantalla)

Para imprimir texto o valores en la consola, usamos `System.out.print` o `System.out.println`:

- `System.out.print("Hola");` → Muestra el texto sin salto de línea.
- `System.out.println("Hola");` → Muestra el texto y luego hace un salto de línea.

Ejemplo:

```
System.out.print("Hola, ");
System.out.println("mundo!");
System.out.println("Bienvenido a Java.");
```

Hola, mundo!
Bienvenido a Java.





⚠ System.console() (solo funciona en algunas consolas)

- No necesita importar nada
- No siempre funciona en IDEs como Eclipse o VS Code

```
public class Main {
    public static void main(String[] args) {
        System.console().printf("Ingresa tu nombre: ");
        String nombre = System.console().readLine();
        System.out.println("Hola, " + nombre);
    }
}
```

⚠ Entrada de datos (pedir al usuario) con Scanner

Para recibir datos del usuario, usamos la clase Scanner:

1. Importar la clase Scanner:

```
import java.util.Scanner;
```

Crear un objeto Scanner

```
Scanner sc = new Scanner(System.in);
```

2. Usar los métodos de Scanner según el tipo de dato:

1. nextInt() → Lee un número entero.
2. nextDouble() → Lee un número decimal.
3. next() → Lee una palabra (sin espacios).
4. nextLine() → Lee una línea completa de texto.





Cofinanciado por
la Unión Europea



MINISTERIO
DE TRABAJO
Y ECONOMÍA SOCIAL



Fondos Europeos



JUNTA DE ANDALUCÍA
Ayuda en Acción
Impulsa Empleo Joven



Ayuda
en Acción
Impulsa Empleo Joven

💡 Ejemplo completo de entrada y salida

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        // Entrada de datos
        System.out.print("Ingresa tu nombre: ");
        String nombre = sc.nextLine();

        System.out.print("Ingresa tu edad: ");
        int edad = sc.nextInt();

        // Salida de datos
        System.out.println("Hola " + nombre + ", tienes " + edad + " años.");
    }
}
```

Ejemplo de ejecución:

```
Ingresa tu nombre: Ana
Ingresa tu edad: 25
Hola Ana, tienes 25 años.
```

💡 Resumen rápido:

- **Salida:** System.out.print() y System.out.println().
- **Entrada:** Scanner con nextInt(), nextDouble(), next(), nextLine().
- **Siempre cerrar Scanner (opcional, pero recomendable):**

```
sc.close();
```



REGISTRO NACIONAL DE ASOCIACIONES N°611922
DECLARADA ENTIDAD DE UTILIDAD PÚBLICA ESTATAL
AGENCIA DE COLOCACIÓN: ID 0100000017

CENTRO EN MÁLAGA
C/DOS ACERAS 23, 29012
MÁLAGA | (+34) 952 300 500
ARRABAL@ARRABALEMPLEO.ORG

CENTRO EN CÁDIZ
TR.ª ALAMEDA DE SOLANO, 32, 11130
CHICLANA DE LA FRONTERA | (+34) 956 900 312
CHICLANA@ARRABALEMPLEO.ORG



7. Control de flujo

Las **estructuras de control** en programación son mecanismos que permiten controlar el flujo de ejecución de un programa, ya sea mediante decisiones (**selección**), repeticiones de código (**iteración**) o saltos en la ejecución.

- **Estructuras de Selección (Condicionales)**
- **Estructuras de Iteración (Bucles)**

Estructuras de Selección (Condicionales)

Permiten tomar decisiones en el código.

if - else

```
int edad = 18;
if (edad >= 18) {
    System.out.println("Eres mayor de edad.");
} else {
    System.out.println("Eres menor de edad.");
}
```

switch

Evalúa una variable y ejecuta un bloque de código según su valor.

```
int dia = 3;
switch (dia) {
    case 1:
        System.out.println("Lunes");
        break;
    case 2:
        System.out.println("Martes");
        break;
    case 3:
        System.out.println("Miércoles");
        break;
    default:
        System.out.println("Otro día");
}
```



REGISTRO NACIONAL DE ASOCIACIONES N°611922
DECLARADA ENTIDAD DE UTILIDAD PÚBLICA ESTATAL
AGENCIA DE COLOCACIÓN: ID 0100000017

CENTRO EN MÁLAGA
C/DOS ACERAS 23, 29012
MÁLAGA | (+34) 952 300 500
ARRABAL@ARRABALEMPLEO.ORG

CENTRO EN CÁDIZ
TR.ª ALAMEDA DE SOLANO, 32, 11130
CHICLANA DE LA FRONTERA | (+34) 956 900 312
CHICLANA@ARRABALEMPLEO.ORG



Estructuras de Iteración (Bucles)

Permiten repetir un bloque de código varias veces.

while

Repite un bloque mientras la condición sea verdadera.

```
int contador = 1;
while (contador <= 5) {
    System.out.println("Número: " + contador);
    contador++;
}
```

do-while

Ejecuta al menos una vez el bloque de código y luego repite mientras la condición sea verdadera.

```
int numero = 1;
do {
    System.out.println("Número: " + numero);
    numero++;
} while (numero <= 5);
```

for

Se usa cuando se conoce el número de iteraciones.

```
for (int i = 1; i <= 5; i++) {
    System.out.println("Iteración: " + i);
}
```

for-each

Sirve para recorrer colecciones como arrays o listas.

```
int[] numeros = {1, 2, 3, 4, 5};
for (int num : numeros) {
    System.out.println("Número: " + num);
}
```





8.Strings en java

En Java, un String es una secuencia de caracteres **inmutable**, lo que significa que su valor no puede cambiar una vez creado.

Creación de Strings

Se pueden crear de dos maneras:

- ◊ **Usando literales ("")** (recomendado, usa el *pool* de strings para optimizar memoria):

```
String saludo = "Hola";
```

- ◊ **Usando new** (crea un nuevo objeto en memoria):

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

Operaciones Comunes con Strings

Concatenación

```
String nombre = "Juan";
String saludo = "Hola, " + nombre;
System.out.println(saludo); // Hola, Juan
```

También con `.concat()`:

```
String mensaje = "Hola".concat(" Mundo");
```

Obtener Longitud

```
int longitud = saludo.length();
```

Acceder a un Carácter Específico

```
char letra = saludo.charAt(0); // 'H'
```



REGISTRO NACIONAL DE ASOCIACIONES N°611922
DECLARADA ENTIDAD DE UTILIDAD PÚBLICA ESTATAL
AGENCIA DE COLOCACIÓN: ID 0100000017

CENTRO EN MÁLAGA
C/DOS ACERAS 23, 29012
MÁLAGA | (+34) 952 300 500
ARRABAL@ARRABALEMPLEO.ORG

CENTRO EN CÁDIZ
TR.^a ALAMEDA DE SOLANO, 32, 11130
CHICLANA DE LA FRONTERA | (+34) 956 900 312
CHICLANA@ARRABALEMPLEO.ORG



Recorrer un String

```
for (int i = 0; i < saludo.length(); i++) {
    System.out.println(saludo.charAt(i));
}
```

Comparación de Strings

- ◊ Comparación con equals() (recomendada)

```
String a = "Hola";
String b = "Hola";
boolean sonIguales = a.equals(b); // true
```

- ◊ Comparación con == (NO recomendado para contenido, compara referencias)

```
boolean mismoObjeto = (a == b);
```

- ◊ Comparación ignorando mayúsculas/minúsculas

```
boolean iguales = a.equalsIgnoreCase("hola"); // true
```

Buscar Substrings

- ◊ Ver si contiene una subcadena:

```
boolean contiene = saludo.contains("la"); // true
```

- ◊ Obtener una subcadena:

```
String parte = saludo.substring(0, 4); // "Hola"
```





Modificar Strings

◊ Convertir a minúsculas o mayúsculas:

```
String minusculas = saludo.toLowerCase();
String mayusculas = saludo.toUpperCase();
```

◊ Reemplazar caracteres o palabras:

```
String nuevoSaludo = saludo.replace("Hola", "Adiós"); // "Adiós, Juan"
```

◊ Eliminar espacios al inicio y fin:

```
String texto = " Hola ";
String limpio = texto.trim(); // "Hola"
```

◊ Dividir un String (split)

```
String frase = "Java,Python,C++";
String[] lenguajes = frase.split(",");
```

◊ Convertir Otros Tipos a String

```
int numero = 100;
String numeroStr = String.valueOf(numero);
```

◊ Convertir un String a Número

```
int num = Integer.parseInt("123");
double decimal = Double.parseDouble("3.14");
```





9.Arrays en java

Los **arrays** en Java son estructuras de datos que almacenan elementos **del mismo tipo** en posiciones contiguas de memoria. Su tamaño es **fijo** una vez definido.

Declaración y Creación de un Array

Para declarar un array, se usa la siguiente sintaxis:

```
tipo[] nombreArray;
```

Para crearlo, usamos new e indicamos su tamaño:

```
int[] numeros = new int[5]; // Array de 5 enteros
```

también podemos inicializarlo directamente con valores:

```
int[] numeros = {1, 2, 3, 4, 5};
```

Acceso a Elementos de un Array

Se accede a los elementos por su **índice**, que comienza en 0:

```
int primerElemento = numeros[0]; // Obtiene el primer elemento
numeros[2] = 10; // Modifica el tercer elemento
```

Recorrer un Array

Podemos recorrerlo con diferentes estructuras de control:

Usando un for tradicional

```
for (int i = 0; i < numeros.length; i++) {
    System.out.println(numeros[i]);
}
```

Usando un for-each

```
for (int num : numeros) {
    System.out.println(num);
}
```





Insertar y Eliminar Elementos en un Array

Los arrays en Java tienen **tamaño fijo**, por lo que **no se pueden insertar ni eliminar elementos directamente**. Para hacerlo, usamos estructuras dinámicas como ArrayList, o creamos un nuevo array con el tamaño modificado.

Insertar un Elemento (creando un nuevo array)

```
int[] original = {1, 2, 3, 4};
int nuevoElemento = 5;
int[] nuevoArray = new int[original.length + 1];

for (int i = 0; i < original.length; i++) {
    nuevoArray[i] = original[i];
}

nuevoArray[nuevoArray.length - 1] = nuevoElemento;
```

Eliminar un Elemento (creando un nuevo array sin el elemento)

```
int indiceAEliminar = 2; // Posición del elemento a eliminar
int[] nuevoArray = new int[original.length - 1];

for (int i = 0, j = 0; i < original.length; i++) {
    if (i != indiceAEliminar) {
        nuevoArray[j++] = original[i];
    }
}
```

Otras Operaciones Comunes con Arrays

Obtener la longitud del array

```
int tamano = numeros.length;
```

Copiar un array

```
int[] copia = Arrays.copyOf(numeros, numeros.length);
```

Ordenar un array

```
Arrays.sort(numeros);
```



REGISTRO NACIONAL DE ASOCIACIONES N°611922
DECLARADA ENTIDAD DE UTILIDAD PÚBLICA ESTATAL
AGENCIA DE COLOCACIÓN: ID 0100000017

CENTRO EN MÁLAGA
C/DOS ACERAS 23, 29012
MÁLAGA | (+34) 952 300 500
ARRABAL@ARRABALEMPLEO.ORG

CENTRO EN CÁDIZ
TR.ª ALAMEDA DE SOLANO, 32, 11130
CHICLANA DE LA FRONTERA | (+34) 956 900 312
CHICLANA@ARRABALEMPLEO.ORG



10. Funciones en Java

En Java, una **función** (o método) es un bloque de código reutilizable que realiza una tarea específica. Puede recibir parámetros y devolver un resultado.

Las funciones en Java son clave para estructurar el código de manera modular y reutilizable. Se pueden definir con distintos tipos de retorno, sobrecargarlas y usarlas de forma estática o mediante instancias de una clase.

Declaración de una Función

Una función en Java tiene la siguiente estructura:

```
tipoDeRetorno nombreFuncion(parámetros) {
    // Código de la función
    return valor; // Opcional si el tipo de retorno no es void
}
```

Ejemplo de una función que suma dos números y devuelve el resultado:

```
public int sumar(int a, int b) {
    return a + b;
}
```

Llamar a una Función

Para usar una función, se debe invocar desde otro método (como main):

```
public class Ejemplo {
    public static int sumar(int a, int b) {
        return a + b;
    }

    public static void main(String[] args) {
        int resultado = sumar(5, 3);
        System.out.println("Resultado: " + resultado);
    }
}
```





Cofinanciado por
la Unión Europea



MINISTERIO
DE TRABAJO
Y ECONOMÍA SOCIAL



Fondos Europeos



Funciones con y sin Parámetros

◊ Función sin parámetros

```
public void saludar() {  
    System.out.println("Hola, mundo!");  
}
```

◊ Función con parámetros

```
public void mostrarMensaje(String mensaje) {  
    System.out.println(mensaje);  
}
```

Funciones con Diferentes Tipos de Retorno

◊ Función que devuelve un valor (int)

```
public int cuadrado(int num) {  
    return num * num;  
}
```

◊ Función sin retorno (void)

```
public void imprimirMensaje() {  
    System.out.println("Este método no devuelve nada.");  
}
```

◊ Función que devuelve un boolean

```
public boolean esPar(int numero) {  
    return numero % 2 == 0;  
}
```



REGISTRO NACIONAL DE ASOCIACIONES N°611922
DECLARADA ENTIDAD DE UTILIDAD PÚBLICA ESTATAL
AGENCIA DE COLOCACIÓN: ID 0100000017

CENTRO EN MÁLAGA
C/DOS ACERAS 23, 29012
MÁLAGA | (+34) 952 300 500
ARRABAL@ARRABALEMPLEO.ORG

CENTRO EN CÁDIZ
TR.^a ALAMEDA DE SOLANO, 32, 11130
CHICLANA DE LA FRONTERA | (+34) 956 900 312
CHICLANA@ARRABALEMPLEO.ORG



Cofinanciado por
la Unión Europea



MINISTERIO
DE TRABAJO
Y ECONOMÍA SOCIAL



Fondos Europeos



Funciones static vs No Estáticas

- ◊ **Método static (puede llamarse sin crear un objeto de la clase)**

```
public static void metodoEstatico() {
    System.out.println("Método estático");
}
```

- ◊ **Método no estático (requiere una instancia de la clase)**

```
public void metodoNoEstatico() {
    System.out.println("Método no estático");
}
```

Ejemplo de llamada a un método no estático:

```
Ejemplo obj = new Ejemplo();
obj.metodoNoEstatico();
```

Paso de Parámetros por Valor o referencia

En Java, los **tipos primitivos** se pasan **por valor** (se copia el dato):

```
public void cambiarValor(int x) {
    x = 10; // No afecta la variable original
}
```

Los **objetos** se pasan **por referencia**, pero el objeto en sí no cambia, solo sus atributos:

```
public void modificarObjeto(Persona p) {
    p.nombre = "Nuevo Nombre"; // Modifica el atributo del objeto original
}
```



REGISTRO NACIONAL DE ASOCIACIONES N°611922
DECLARADA ENTIDAD DE UTILIDAD PÚBLICA ESTATAL
AGENCIA DE COLOCACIÓN: ID 0100000017

CENTRO EN MÁLAGA
C/DOS ACERAS 23, 29012
MÁLAGA | (+34) 952 300 500
ARRABAL@ARRABALEMPLEO.ORG

CENTRO EN CÁDIZ
TR.^a ALAMEDA DE SOLANO, 32, 11130
CHICLANA DE LA FRONTERA | (+34) 956 900 312
CHICLANA@ARRABALEMPLEO.ORG



Cofinanciado por
la Unión Europea



Métodos con Parámetros Variables (var args)

Podemos definir un método que acepte un número variable de argumentos:

```
public void imprimirNumeros(int... numeros) {
    for (int num : numeros) {
        System.out.println(num);
    }
}
```

Llamada:

```
imprimirNumeros(1, 2, 3, 4, 5);
```

Métodos Recursivos

Un método puede llamarse a sí mismo para resolver problemas como el factorial:

```
public int factorial(int n) {
    if (n == 0) return 1;
    return n * factorial(n - 1);
}
```



REGISTRO NACIONAL DE ASOCIACIONES N°611922
DECLARADA ENTIDAD DE UTILIDAD PÚBLICA ESTATAL
AGENCIA DE COLOCACIÓN: ID 0100000017

CENTRO EN MÁLAGA
C/DOS ACERAS 23, 29012
MÁLAGA | (+34) 952 300 500
ARRABAL@ARRABALEMPLEO.ORG

CENTRO EN CÁDIZ
TR.^a ALAMEDA DE SOLANO, 32, 11130
CHICLANA DE LA FRONTERA | (+34) 956 900 312
CHICLANA@ARRABALEMPLEO.ORG



11. Packages en Java

Los paquetes (packages) son una forma de organizar y agrupar clases relacionadas en diferentes archivos para mantener el código estructurado y evitar conflictos de nombres.

¿Qué es un paquete en Java?

Un **paquete** es simplemente un directorio que contiene clases y otras subcarpetas con más clases.

Ejemplo de una estructura de paquetes:

```

📁 proyecto
  |- 📁 src
    |- 📁 com.miempresa.utilidades
      |- Utilidades.java
      |- Constantes.java
    |- 📁 com.miempresa.modelos
      |- Persona.java
  |- Main.java
  
```

Aquí, com.miempresa.utilidades y com.miempresa.modelos son paquetes que agrupan clases con propósitos específicos.

Cómo Crear un Paquete en Java

Para declarar que una clase pertenece a un paquete, usamos la palabra clave **package** al inicio del archivo:

◊ **Ejemplo de una clase en un paquete**

```

package com.miempresa.utilidades; // Define el paquete

public class Utilidades {
    public static void saludar() {
        System.out.println("Hola desde Utilidades!");
    }
}
  
```



REGISTRO NACIONAL DE ASOCIACIONES N°611922
DECLARADA ENTIDAD DE UTILIDAD PÚBLICA ESTATAL
AGENCIA DE COLOCACIÓN: ID 0100000017

CENTRO EN MÁLAGA
C/DOS ACERAS 23, 29012
MÁLAGA | (+34) 952 300 500
ARRABAL@ARRABALEMPLEO.ORG

CENTRO EN CÁDIZ
TR.^a ALAMEDA DE SOLANO, 32, 11130
CHICLANA DE LA FRONTERA | (+34) 956 900 312
CHICLANA@ARRABALEMPLEO.ORG



Otra clase en otro paquete

```
package com.miempresa.modelos;

public class Persona {
    public String nombre;

    public Persona(String nombre) {
        this.nombre = nombre;
    }
}
```

Cómo Importar Clases de Otros Paquetes

Para usar clases de otro paquete, debes **importarlas** con **import**:

```
import com.miempresa.utilidades.Utilidades;
import com.miempresa.modelos.Persona;

public class Main {
    public static void main(String[] args) {
        Utilidades.saludar();
        Persona p = new Persona("Juan");
        System.out.println("Nombre: " + p.nombre);
    }
}
```

También puedes importar **todas las clases de un paquete** con *:

```
import com.miempresa.utilidades.*;
```



REGISTRO NACIONAL DE ASOCIACIONES N°611922
DECLARADA ENTIDAD DE UTILIDAD PÚBLICA ESTATAL
AGENCIA DE COLOCACIÓN: ID 0100000017

CENTRO EN MÁLAGA
C/DOS ACERAS 23, 29012
MÁLAGA | (+34) 952 300 500
ARRABAL@ARRABALEMPLEO.ORG

CENTRO EN CÁDIZ
TR.^a ALAMEDA DE SOLANO, 32, 11130
CHICLANA DE LA FRONTERA | (+34) 956 900 312
CHICLANA@ARRABALEMPLEO.ORG



Paquetes y Modificadores de Acceso

Si una clase o método está marcada como private o sin modificador (*default*), no será accesible desde otro paquete.

- ◊ Para permitir el acceso desde otros paquetes, usa public:

```
package com.miempresa.utilidades;

public class Utilidades {
    public static void saludar() { // Debe ser público
        System.out.println("Hola!");
    }
}
```

- ◊ Si el método fuera private, no podríamos llamarlo desde Main.

Uso de import static para Métodos Estáticos

Si tienes métodos estáticos en una clase, puedes importarlos directamente para usarlos sin el nombre de la clase:

```
import static com.miempresa.utilidades.Utilidades.saludar;

public class Main {
    public static void main(String[] args) {
        saludar(); // Sin necesidad de escribir Utilidades.saludar();
    }
}
```



12.Tipos Enumerados (enum) en Java

En Java, un **enumerado** (enum) es un tipo de dato especial que representa un **conjunto fijo de valores constantes**. Se usa cuando una variable solo puede tomar ciertos valores predefinidos, como días de la semana, colores, estados, etc.

Los enum en Java son útiles para representar conjuntos de valores constantes de manera clara y segura.

Declaración de un enum

Un enum se define con la palabra clave enum y los valores se separan por comas:

```
enum Dia {
    LUNES, MARTES, MIERCOLES, JUEVES, VIERNES, SABADO, DOMINGO
}
```

- ◊ Por convención, los valores de un enum se escriben en **mayúsculas**.

Uso Básico de un enum

Puedes usar un enum como cualquier otro tipo de dato:

```
public class EjemploEnum {
    public static void main(String[] args) {
        Dia hoy = Dia.LUNES;
        System.out.println("Hoy es: " + hoy); // Hoy es: LUNES
    }
}
```

Recorrer un enum

Puedes recorrer todos los valores con values():

```
for (Dia d : Dia.values()) {
    System.out.println(d);
}
```

Salida:

LUNES
MARTES
MIERCOLES
JUEVES
VIERNES
SABADO
DOMINGO





Métodos Útiles en los enum

◊ Obtener el nombre del valor como String

```
System.out.println(Dia.LUNES.name()); // "LUNES"
```

◊ Obtener el índice de un valor (ordinal())

```
System.out.println(Dia.LUNES.ordinal()); // 0
System.out.println(Dia.VIERNES.ordinal()); // 4
```

◊ Convertir un String a enum (valueOf())

```
Dia dia = Dia.valueOf("MARTES");
System.out.println(dia); // MARTES
```

Uso de switch con enum

Puedes usar enum en una estructura switch:

```
public class EjemploSwitch {
    public static void main(String[] args) {
        Dia hoy = Dia.LUNES;

        switch (hoy) {
            case LUNES:
                System.out.println("Es lunes, ánimo!");
                break;
            case VIERNES:
                System.out.println("Es viernes, casi fin de semana!");
                break;
            default:
                System.out.println("Es otro día...");
        }
    }
}
```





13. Casting en Java

El **casting** en Java es el proceso de convertir un valor de un tipo de dato a otro. Se usa para hacer compatibles datos de distintos tipos, pero debe hacerse con cuidado para evitar pérdida de información o errores en tiempo de ejecución.

Tipos de Casting en Java

◊ Casting Implícito (Widening)

Se da cuando se convierte un tipo de dato más pequeño a uno más grande. Java lo hace automáticamente porque no hay riesgo de pérdida de datos.

Ejemplo:

```
int numEntero = 10;
double numDecimal = numEntero; // Conversión automática de int a double
System.out.println(numDecimal); // 10.0
```

 Se permite porque **double** tiene mayor capacidad que **int**.

◊ Casting Explícito (Narrowing)

Se da cuando se convierte un tipo más grande a uno más pequeño. Se debe hacer **manualmente** porque puede haber pérdida de datos.

Ejemplo:

```
double numDecimal = 10.99;
int numEntero = (int) numDecimal; // Conversión explícita
System.out.println(numEntero); // 10 (se pierde la parte decimal)
```

 Java no lo hace automáticamente porque **int** no puede almacenar decimales.





Casting entre Tipos Primitivos

De → A	byte	short	char	int	long	float	double
byte	✗	✓	✗	✓	✓	✓	✓
short	✗	✗	✗	✓	✓	✓	✓
char	✓	✗	✗	✓	✓	✓	✓
int	✗	✗	✗	✗	✓	✓	✓
long	✗	✗	✗	✗	✗	✓	✓
float	✗	✗	✗	✗	✗	✗	✓
double	✗	✗	✗	✗	✗	✗	✗

- ✓ (✓) Se puede hacer conversión directa
- ✗ (✗) Requiere casting explícito

Conversión entre Números y Strings en Java

En Java, convertir entre **números** y **cadenas de texto (String)** es una operación común. Aquí te muestro cómo hacerlo de forma sencilla y eficiente.

- ◊ Convertir un Número a un String

Usando String.valueOf() (Método más recomendado)

Convierte cualquier tipo numérico a String:

```
int num = 100;
String strNum = String.valueOf(num);
System.out.println(strNum); // "100"
```

Concatenación con "" (Forma rápida y simple)

```
double pi = 3.1416;
String strPi = pi + "";
System.out.println(strPi); // "3.1416"
```

- ✓ Fácil y rápida, pero menos clara en código grande.





Usando Integer.toString() y similares

Métodos específicos para cada tipo numérico:

```
int num = 50;
String str1 = Integer.toString(num);
String str2 = Double.toString(3.14);
String str3 = Float.toString(9.8f);
System.out.println(str1); // "50"
System.out.println(str2); // "3.14"
System.out.println(str3); // "9.8"
```

◊ Convertir un String a un Número

Usando Integer.parseInt() y similares

Métodos para convertir String a su tipo numérico correspondiente:

```
String strNum = "123";
int num = Integer.parseInt(strNum);
double numD = Double.parseDouble("3.14");
float numF = Float.parseFloat("5.5");
long numL = Long.parseLong("1234567890");
System.out.println(num); // 123
System.out.println(numD); // 3.14
System.out.println(numF); // 5.5
System.out.println(numL); // 1234567890
```

 Si la cadena contiene caracteres no numéricos, se lanzará una **NumberFormatException**:

```
String strInvalido = "12A";
int numero = Integer.parseInt(strInvalido); // ERROR en tiempo de ejecución
```

 Para evitar errores, podemos manejar la excepción con try-catch:

```
try {
    int numero = Integer.parseInt("12A");
    System.out.println(numero);
} catch (NumberFormatException e) {
    System.out.println("Error: La cadena no es un número válido.");
}
```





14. Excepciones en Java

Las **excepciones** en Java son errores que ocurren durante la ejecución del programa. Java tiene un sistema para **detectar, manejar y evitar que el programa se detenga bruscamente** cuando ocurre un error.

¿Cómo funcionan?

Cuando ocurre un error, Java genera un **objeto de excepción** que contiene información sobre el problema. Si no se maneja, el programa se detiene.

Ejemplo de error sin manejar:

```
int resultado = 10 / 0; // Error: División por cero (ArithmeticException)
```

El programa se detendrá y mostrará un mensaje de error.

Manejo de Excepciones (try-catch)

Para evitar que el programa se detenga, usamos try-catch:

```
try {
    int resultado = 10 / 0; // Código que puede generar un error
} catch (ArithmeticException e) {
    System.out.println("Error: No se puede dividir por cero.");
}
```

Si ocurre una excepción dentro de try, Java ejecuta el catch y el programa sigue funcionando.

finally (Código que siempre se ejecuta)

El bloque finally se usa para **ejecutar código sin importar si hubo una excepción o no**.

```
try {
    int[] numeros = {1, 2, 3};
    System.out.println(numeros[5]); // Error: Índice fuera de rango
} catch (ArrayIndexOutOfBoundsException e) {
    System.out.println("Error: Índice inválido.");
} finally {
    System.out.println("Este mensaje siempre se muestra.");
}
```





Cofinanciado por
la Unión Europea



Lanzar Excepciones (throw)

Podemos generar excepciones manualmente usando throw:

```
public void verificarEdad(int edad) {
    if (edad < 18) {
        throw new IllegalArgumentException("Debe ser mayor de edad.");
    }
    System.out.println("Acceso permitido.");
}
```

Crear Excepciones Personalizadas

Podemos definir nuestras propias excepciones extendiendo Exception:

```
class MiExpcion extends Exception {
    public MiExpcion(String mensaje) {
        super(mensaje);
    }
}
```

Y luego lanzarla en el código:

```
public void validar(int valor) throws MiExpcion {
    if (valor < 0) {
        throw new MiExpcion("El número no puede ser negativo.");
    }
}
```

Resumen

- ✓ Las excepciones evitan que el programa se detenga por errores inesperados.
- ✓ Se manejan con try-catch y opcionalmente finally.
- ✓ Podemos lanzar errores con throw y crear excepciones personalizadas.



REGISTRO NACIONAL DE ASOCIACIONES N°611922
DECLARADA ENTIDAD DE UTILIDAD PÚBLICA ESTATAL
AGENCIA DE COLOCACIÓN: ID 0100000017

CENTRO EN MÁLAGA
C/DOS ACERAS 23, 29012
MÁLAGA | (+34) 952 300 500
ARRABAL@ARRABALEMPLEO.ORG

CENTRO EN CÁDIZ
TR.^a ALAMEDA DE SOLANO, 32, 11130
CHICLANA DE LA FRONTERA | (+34) 956 900 312
CHICLANA@ARRABALEMPLEO.ORG