# Sentencias de Decisión en Java

Las sentencias de decisión permiten controlar el flujo de ejecución de un programa, ejecutando diferentes bloques de código según las condiciones evaluadas.

## Estructuras de Decisión en Java

### Sentencia if

La sentencia if ejecuta un bloque de código si la condición evaluada es verdadera. Una condición es una expresión que devuelve un valor booleano (true o false).

#### Sintaxis:

if (condicion) {

// Bloque de código a ejecutar si la condición es verdadera

}

#### Ejemplo:

int edad = 30;

if (edad >= 18) {

System.out.println("Eres mayor de edad");

}

### Sentencia if-else

La sentencia else se usa para ejecutar un bloque de código cuando la condición del if es falsa.

#### Sintaxis:

if (condicion) {

// Bloque de código a ejecutar si la condición es verdadera

} else {

// Bloque de código a ejecutar si la condición es falsa

}

#### Ejemplo:

int edad = 16;

if (edad >= 18) {

System.out.println("Eres mayor de edad");

} else {

System.out.println("Eres menor de edad");

}

### Sentencia if-else if-else

Se utiliza para evaluar múltiples condiciones en secuencia. Cada condición es evaluada en orden, y el primer bloque cuyo if o else if sea verdadero se ejecutará. Si ninguna condición es verdadera, se ejecutará el bloque else.

#### Sintaxis:

if (condicion1) {

// Bloque de código si la condicion1 es verdadera

} else if (condicion2) {

// Bloque de código si la condicion2 es verdadera

} else {

// Bloque de código si ninguna condición anterior es verdadera

}

#### Ejemplo:

int calificacion = 85;

if (calificacion >= 90) {

System.out.println("Excelente");

} else if (calificacion >= 70) {

System.out.println("Aprobado");

} else {

System.out.println("Reprobado");

}

## Depuración paso a paso en Java

Para identificar errores en tu programa, puedes ejecutar el código paso a paso utilizando herramientas de depuración (debug). Sigue estos pasos:

1. **Establecer un punto de ruptura**: Haz clic a la izquierda de una línea de código en tu editor (por ejemplo, en IntelliJ IDEA o Eclipse) para colocar un punto rojo. Esto detendrá la ejecución en esa línea.
2. **Ejecutar en modo depuración**: En lugar de ejecutar el programa con "Run", elige "Debug".
3. **Inspeccionar variables y flujo**: Usa la consola de depuración para avanzar paso a paso y observar el comportamiento del código y los valores de las variables.

Esto te ayudará a identificar errores y comprender el flujo de tu programa.

## Sentencia switch

La sentencia switch permite evaluar un solo valor contra múltiples casos posibles. Es útil para reemplazar estructuras con múltiples else if.

#### Sintaxis:

switch (valorAEvaluar) {

case valor1:

// Bloque de código para valor1

break;

case valor2:

// Bloque de código para valor2

break;

default:

// Bloque de código si ningún caso coincide

}

#### Ejemplo:

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("Día no válido");

}

#### Notas:

* El uso de break es importante para evitar que el programa continúe ejecutando los casos siguientes.
* Si se omite el break, se produce el "fall-through", lo que significa que el programa ejecutará todos los casos siguientes hasta encontrar un break o llegar al final.

## Precedencia de Operadores en Java

Cuando se combinan operadores en una expresión, Java sigue un orden de precedencia para evaluarlos:

1. **Paréntesis y corchetes**
2. **Operadores unarios** (-, ++, --)
3. **Operadores aritméticos** (\*, /, %)
4. **Operadores aritméticos** (+, -)
5. **Operadores relacionales** (<, <=, >, >=)
6. **Operadores de equivalencia** (==, !=)
7. **Operadores lógicos** (&&, ||)
8. **Asignación** (=, +=, -=, etc.)

#### Ejemplo de evaluación:

Dada la expresión:

int resultado = 12 / 3 + 2 \* 3 - 1;

Paso a paso:

1. División: 12 / 3 = 4
2. Multiplicación: 2 \* 3 = 6
3. Suma: 4 + 6 = 10
4. Resta: 10 - 1 = 9

El resultado final es 9.