



## TEMA 03

**PROGRAMACIÓN  
CFGS DAM**

**2023/2024**

**Versión: 231001.2241**

## ESTRUCTURAS ALTERNATIVAS

# ÍNDICE DE CONTENIDO

<b>1. Introducción.....</b>	<b>3</b>
<b>2. Estructuras alternativas.....</b>	<b>3</b>
2.1 Estructura Alternativa Simple (if).....	3
2.2 Estructura Alternativa Doble (if-else).....	4
2.3 Estructura Alternativa Múltiple (switch).....	6
<b>3. ejemplos.....</b>	<b>8</b>
3.1 Ejemplo 1.....	8
3.2 Ejemplo 2.....	10
<b>4. Licencia.....</b>	<b>12</b>
4.1 Agradecimientos.....	12

## UD3. ESTRUCTURAS ALTERNATIVAS

### 1. INTRODUCCIÓN

Como ya vimos, las estructuras alternativas son construcciones que permiten alterar el flujo secuencial de un programa de forma que en función de una condición o el valor de una expresión, el mismo pueda ser desviado en una u otra alternativa de código.

Las estructuras alternativas disponibles en Java son:

- Alternativa Simple (if)
- Alternativa Doble (if-else)
- Alternativa Múltiple (switch)

### 2. ESTRUCTURAS ALTERNATIVAS

#### 2.1 Estructura Alternativa Simple (if)

La alternativa simple se codifica de la siguiente forma:

<i>if</i>	
<pre>if (condición) {     // Acciones; }</pre> <p>El bloque de Acciones se ejecuta si la condición se evalúa a true (es verdadera).</p> <pre>if (cont == 0) {     System.out.println("cont es 0");     // más instrucciones... }</pre> <p>Si dentro del if solo hay una instrucción, no es necesario poner las llaves.</p> <pre>if (cont == 0)     System.out.println("cont es 0");</pre>	<pre>if (condicion) {     sentencias }</pre> <pre>graph TD     Entry(( )) --&gt; Cond{condición}     Cond -- verdadero --&gt; Sent[sentencias]     Cond -- falso --&gt; Exit(( ))     Sent --&gt; Exit     Exit --&gt; Exit2(( ))</pre>

## 2.2 Estructura Alternativa Doble (if-else)

La alternativa doble permite indicar qué código ejecutar si la condición es falsa.

### *If else*

```
if (condición)
{
    // AccionesSI;
}
else
{
    // AccionesNO;
}
```

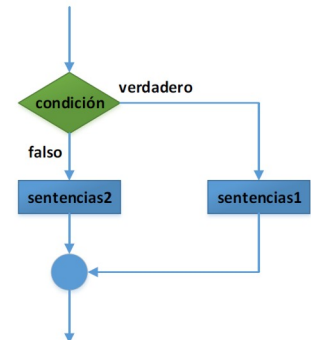
El bloque AccionesSI se ejecuta si la condición se evalúa a true (verdadera). En caso contrario, se ejecuta el bloque de AccionesNO.

```
if (cont == 0)
{
    System.out.println("cont es 0");
    // más instrucciones...
}
else
{
    System.out.println("cont no es 0");
    // más instrucciones...
}
```

Si dentro del if o el else solo hay una instrucción, no es necesario poner las llaves.

```
if (cont == 0)
    System.out.println("cont es 0");
else
    System.out.println("cont no es 0");
```

```
if (condicion)
{
    sentencias1
}
else
{
    sentencias2
}
```



⚡ Recordad que el operador relacional para comprobar si son iguales es `==`, no un solo `=` que corresponde con el operador de asignación. Este error no lo detecta el compilador y es difícil de averiguar.

En muchas ocasiones, se anidan estructuras alternativas `if-else`, de forma que se pregunte por una condición si anteriormente no se ha cumplido otra sucesivamente.

Por ejemplo: supongamos que realizamos un programa que muestra la nota de un alumno en la forma (insuficiente, suficiente, bien, notable o sobresaliente) en función de su nota numérica. Podría codificarse de la siguiente forma:

```
1
2 import java.util.Scanner;
3
4 public class Nota {
5
6     public static void main(String[] args) {
7         Scanner entrada = new Scanner(System.in);
8         int nota;
9         //Suponemos que el usuario introduce el número correctamente.
10        //No hacemos comprobación
11        System.out.println("Dame un número entre 0 y 10");
12
13        nota = entrada.nextInt();
14
15        if (nota < 5) {
16            System.out.println("Insuficiente");
17        } else if (nota < 6) {
18            System.out.println("Suficiente");
19        } else if (nota < 7) {
20            System.out.println("Bien");
21        } else if (nota < 9) {
22            System.out.println("Notable");
23        } else {
24            System.out.println("Sobresaliente");
25        }
26    }
27
28 }
```

Siendo la salida:

```
run:
Dame un número entre 0 y 10
8
Notable
BUILD SUCCESSFUL (total time: 11 seconds)
```

🔊 Es muy recomendable usar la tecla tabulador en las instrucciones de cada bloque. Como se puede ver en el ejemplo, cada **else** está alineado con su **if** asociado, de esta forma es más fácil leer el código.

### 2.3 Estructura Alternativa Múltiple (switch)

La alternativa múltiple se codifica de la siguiente forma:

Código	Ordinograma
<pre>switch (expresión) {     case valor1:         // Acciones1;         break;     case valor2:         // Acciones2;         break;     case valorN:         // AccionesN;         break;     default:         // Acciones por defecto; }</pre>	

Es muy importante entender que en el **switch** se evalúa una **expresión** (un valor concreto como 0, 5, 1...) **no una condición** (verdadera o falsa) como en el **if** y el **ifelse**.

El programa comprueba el valor de la expresión y saltará al 'case' que corresponda con dicho valor (valor1 o valor2 o ...) ejecutando el código de dicho 'case' (Acciones1 o Acciones2 o ...). Si no coincide ningún valor, saltará al 'default' y ejecutará las acciones por defecto.

Es importante añadir la sentencia **break;** al final de cada 'case', ya que de lo contrario el programa seguirá ejecutando el código de las demás acciones y normalmente no queremos que haga eso (aunque Java permite hacerlo, es confuso y por ello está desaconsejado).

Un ejemplo sería el siguiente:

```
1
2 import java.util.Scanner;
3
4 public class Alternativa_Multiple {
5
6     public static void main(String[] args) {
7         Scanner entrada = new Scanner(System.in);
8         int dia;
9
10        System.out.println("Dame un número entre 1 y 7:");
11
12        dia = entrada.nextInt();
13
14        switch (dia) {
15            case 1:
16                System.out.println("Lunes");
17                break;
18            case 2:
19                System.out.println("Martes");
20                break;
21            case 3:
22                System.out.println("Miércoles");
23                break;
24            case 4:
25                System.out.println("Jueves");
26                break;
27            case 5:
28                System.out.println("Viernes");
29                break;
30            case 6:
31                System.out.println("Sábado");
32                break;
33            case 7:
34                System.out.println("Domingo");
35                break;
36            default:
37                System.out.println("Error el número debe estar entre 0 y 7");
38        }
39    }
40 }
41
42 }
```

Y la salida:

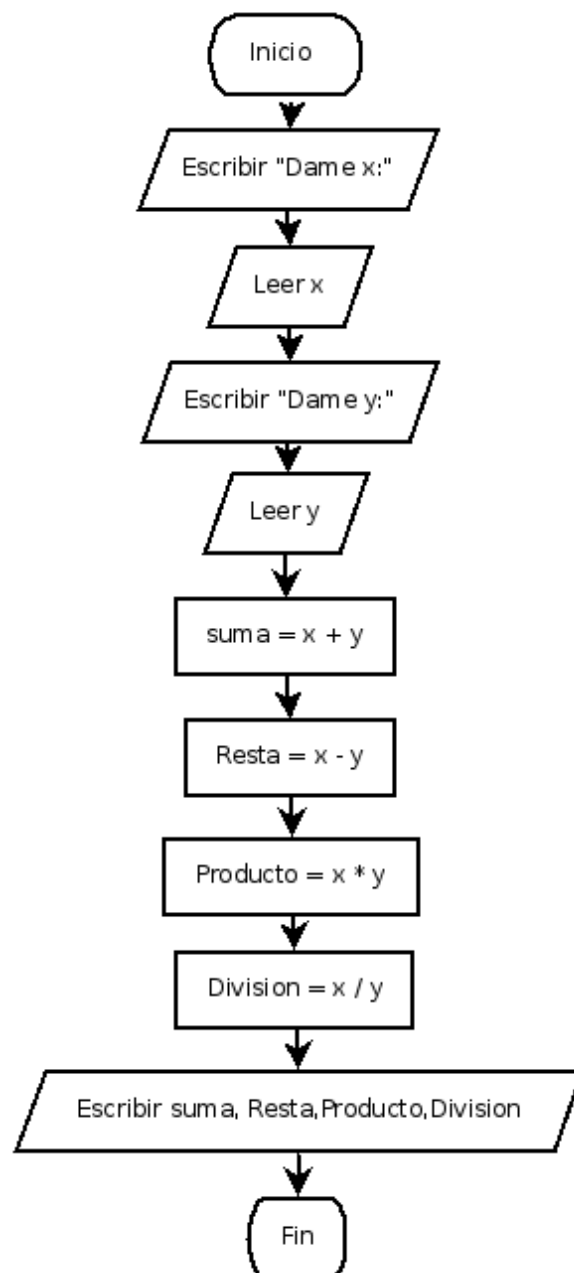
```
run:
Dame un número entre 1 y 7:
4
Jueves
BUILD SUCCESSFUL (total time: 2 seconds)
```

### 3. EJEMPLOS

#### 3.1 Ejemplo 1

Programa que lea dos números, calcule y muestre el valor de sus suma, resta, producto y división.

Ordinograma:

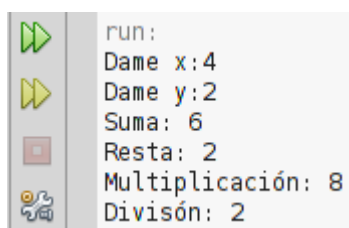




Código:

```
1  package ejemplo1;
2
3  import java.util.Scanner; // Importamos la clase Scanner
4
5  public class Ejemplo1 {
6
7      public static void main(String[] args) {
8
9          // Declaramos las variables que vamos a necesitar
10         int x, y, suma, resta, mult, div;
11
12         // Creamos el objeto Scanner para leer por teclado
13         Scanner reader = new Scanner(System.in);
14
15         // Pedimos y leemos x
16         System.out.print("Dame x:");
17         x = reader.nextInt();
18
19         // Pedimos y leemos y
20         System.out.print("Dame y:");
21         y = reader.nextInt();
22
23         // Realizamos los cálculos necesarios
24         suma = x + y;
25         resta = x - y;
26         mult = x * y;
27         div = x / y;
28
29         // Mostramos los cálculos por pantalla
30         System.out.println("Suma: " + suma);
31         System.out.println("Resta: " + resta);
32         System.out.println("Multiplicación: " + mult);
33         System.out.println("División: " + div);
34     }
35 }
```

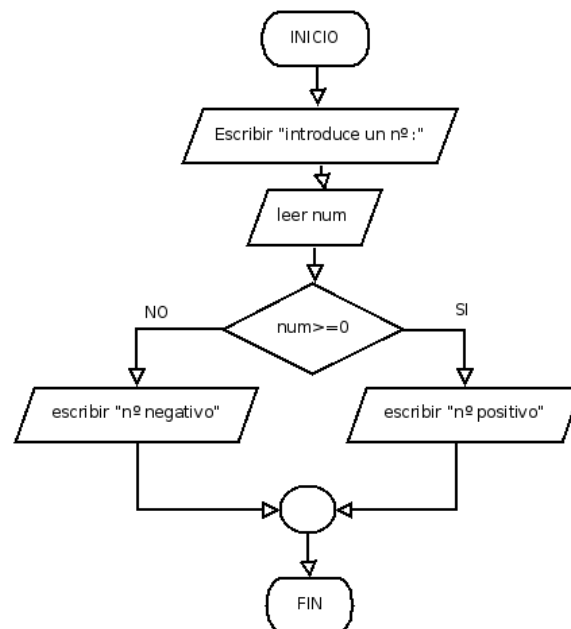
Salida:



```
run:
Dame x:4
Dame y:2
Suma: 6
Resta: 2
Multiplicación: 8
División: 2
```

### 3.2 Ejemplo 2

Programa que lee un número y me dice si es positivo o negativo. Consideraremos el cero como positivo.



```
1  package ejemplo2;
2
3  import java.util.Scanner; // Importamos la clase Scanner
4
5  public class Ejemplo2 {
6
7      public static void main(String[] args) {
8
9          // Declaramos la variable num
10         int num;
11
12         // Creamos el objeto Scanner para leer por teclado
13         Scanner reader = new Scanner(System.in);
14
15         // Pedimos y leemos x
16         System.out.print("Introduce un nº: ");
17         num = reader.nextInt();
18
19         // A estructura alternativa doble
20         if (num >= 0)
21             System.out.println("Número positivo");
22         else
23             System.out.println("Número negativo");
24     }
25 }
```

```
run:
Introduce un nº: 10
Número positivo
```



### Práctica 1: Condicionales

Realiza los siguientes ejercicios:

1. 01-Parimpar → Dado un número entero positivo determine si es par o impar.
2. 04-TresCuadrados → Escriba un programa que cada vez que se ejecute calcule tres cuadrados y nos muestre el valor de su lado, su área y nos diga qué cuadrado es el mayor.
3. 05-Diasemana → Dado un número del 1 al 7 indique a que día de la semana corresponde
4. 06-Seguros → Dada la siguiente tabla de precios de una compañía aseguradora:  
  
100 € - Jóvenes Sanos  
120 € - Jóvenes Enfermos – Edad Media Sanos  
140 € - Edad Media Enfermos – Edad Avanzada Sanos  
No se admiten – Edad Avanzada Enfermos  
Donde se considera:  
Gente Joven a menores de 28 años  
Gente de Edad Media entre 28 y 50 años  
Gente de Edad Avanzada a mayores de 50 años  
  
Escribir un programa que pida **edad** y estado de **Salud** (S/E) y nos devuelva el **importe** de la póliza.
5. 07-Radar → En la nueva normativa de tráfico una velocidad superior al 50% a la permitida, siempre que el exceso sea mayor a 30 km/h, supone la retirada del carnet de conducir. Escribir un programa que dada la velocidad máxima permitida y la velocidad de circulación muestre si se debe retirar el carnet.
6. 08-Billetes → Escribe un programa que determine el menor número de billetes y monedas de curso legal (euros) equivalente a cierta cantidad de dinero (1, 2, 5, 10, 20, 50, 100, 200, 500 euros)
7. 09-Bisiesto → Calcular si un año es bisiesto. La regla completa para saber si un año es bisiesto, es que sea divisible por 4, excepto que si es divisible por 100 lo sea también por 400. Realizar la comprobación mediante una sola operación lógica y muestre en la pantalla un mensaje con el resultado.  
  
Para saber si un año es bisiesto podemos utilizar:  
  
$$\text{EsABisiesto} = \text{anyo} \% 4 == 0 \text{ and } \text{anyo} \% 100 != 0 \text{ or } \text{anyo} \% 400 == 0$$
8. 10-Fechacorrecta → Dado tres número decir si forman una fecha exacta (no tenemos en cuenta los años bisiestos)

## 4. LICENCIA



[CC BY-NC-SA 3.0 ES](https://creativecommons.org/licenses/by-nc-sa/3.0/es/) Reconocimiento – No Comercial – Compartir Igual (by-nc-sa)

No se permite un uso comercial de la obra original ni de las posibles obras derivadas, la distribución de las cuales se debe hacer con una licencia igual a la que regula la obra original. Esta es una obra derivada de la obra original de Carlos Cacho y Raquel Torres. Revisado por Lionel Tarazon - [lionel.tarazon@ceedcv.es](mailto:lionel.tarazon@ceedcv.es) y Fco. Javier Valero – [franciscojavier.valero@ceedcv.es](mailto:franciscojavier.valero@ceedcv.es)

### 4.1 Agradecimientos

Apuntes actualizados y adaptados al CEEDCV a partir de la siguiente documentación:

[1] Apuntes Programación de José Antonio Díaz-Alejo. IES Camp de Morvedre.

## Nomenclatura

A lo largo de este tema se utilizarán distintos símbolos para distinguir elementos importantes dentro del contenido. Estos símbolos son:



Importante



Atención



Interesante