

**MP0485**  
**Programación**  
**UF3. El lenguaje de programación Java**

## **3.2 Sentencias de control de flujo**

# Índice

---

☰	Objetivos	3
☰	Estructuras básicas	4
☰	Estructuras condicionales o alternativas	6
☰	Alternativa simple (if...)	7
☰	Alternativa doble (if... else...)	8
☰	Alternativa múltiple (switch)	11
☰	Estructura mientras (while)	16
☰	Estructura para (for)	19
☰	Resumen	22

## Objetivos

---

Con esta unidad perseguimos los siguientes objetivos:

1

Comprender los fundamentos del paradigma de programación estructurada y su aplicación en Java.

2

Realizar programas Java utilizando las tres estructuras básicas: secuencial, condicional y de repetición.

---

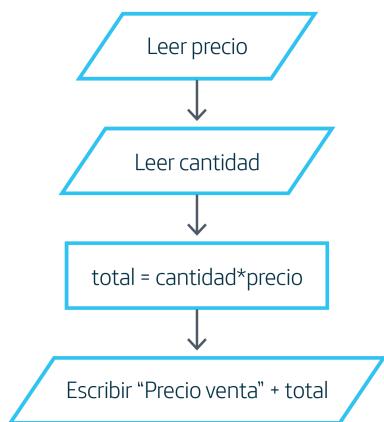
¡Ánimo y adelante!

# Estructuras básicas

Recuerda: el teorema del programa estructurado demuestra que "Todo algoritmo puede ser diseñado empleando solo tres estructuras básicas: secuencial, alternativa y repetitiva".

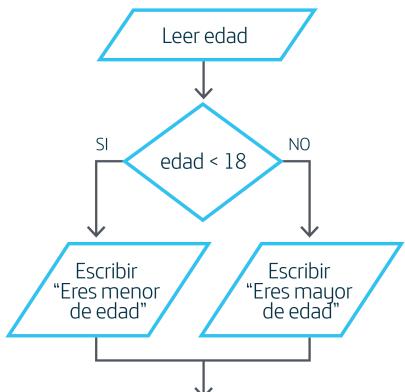
## SECUENCIAL

Sucesión lineal de varias instrucciones que se van ejecutando una tras otra.



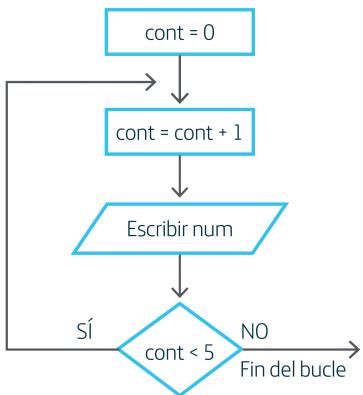
## ALTERNATIVA

Acción o grupo de acciones cuya ejecución depende del cumplimiento de una condición.



**REPETITIVA**

Grupo de acciones que se repiten (bucle o iteración) un número determinado de veces.



---

En esta lección repasarás todas las estructuras de control que ya aprendiste en una unidad anterior, pero esta vez verás cómo implementarlas en el lenguaje Java y tendrás oportunidad de ponerlo en práctica.

# Estructuras condicionales o alternativas

Un bloque de código compuesto por una o varias instrucciones se ejecutará en función del cumplimiento de una condición.

Existen tres variantes de sentencia condicional o alternativa:

1

Alternativa simple: un bloque de código compuesto por una o varias instrucciones se ejecutará en función del cumplimiento de una condición.

2

Alternativa doble: si se cumple una condición se ejecuta un bloque de código y si no se cumple, se ejecuta otro bloque de código diferente.

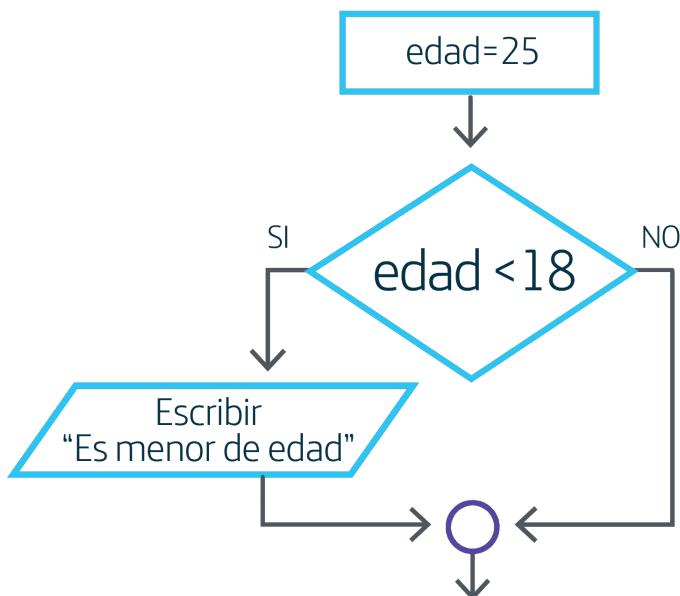
3

Alternativa múltiple (*switch*): dependiendo del contenido de una variable o del resultado de una expresión se ejecutará un bloque determinado de instrucciones dentro de un conjunto de ellos.

En los siguientes apartados aprenderás a utilizar estas tres variantes.

## Alternativa simple (if...)

Un bloque de código compuesto por una o varias instrucciones se ejecutará en función del cumplimiento de una condición.



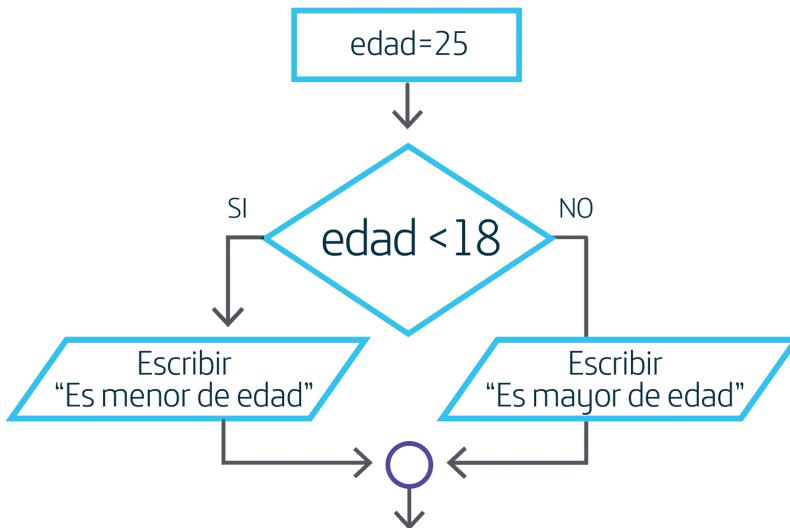
Estructura alternativa simple.

```
int edad = 12;
if (edad<18) {
    System.out.println("Es menor de edad");
}
```

Estructura alternativa simple en Java.

## Alternativa doble (if... else...)

Si se cumple una condición se ejecuta un bloque de código y si no se cumple se ejecuta otro bloque de código diferente.



Alternativa doble.

```
int edad = 25;  
if (edad<18) {  
    System.out.println("Es menor de edad");  
}  
else {  
    System.out.println("Es mayor de edad");  
}
```

Alternativa doble en Java.

## Varios *if* en escalera

Dentro del bloque *if* o dentro del bloque *else*, puede ir cualquier grupo de instrucciones, incluso otro *if*. Esto puede formar lo que se llama *if* anidados o en escalera.

```
int edad = 12;
if (edad<13) {
    System.out.println("Infantil");
}
else {
    if (edad<18) {
        System.out.println("Adolescente");
    }
    else {
        if (edad<40) {
            System.out.println("Joven mayor de edad");
        }
        else {
            if (edad<65) {
                System.out.println("Adulto mayor de edad");
            }
            else {
                System.out.println("Jubilado");
            }
        }
    }
}
```

Sentencias *if* en escalera.

Esto funciona pero resulta engorroso; cuantos más peldaños tiene la escalera, más enrevesada parece la estructura. La mayoría de los lenguajes de programación permiten simplificar los *if* anidados de la siguiente forma:

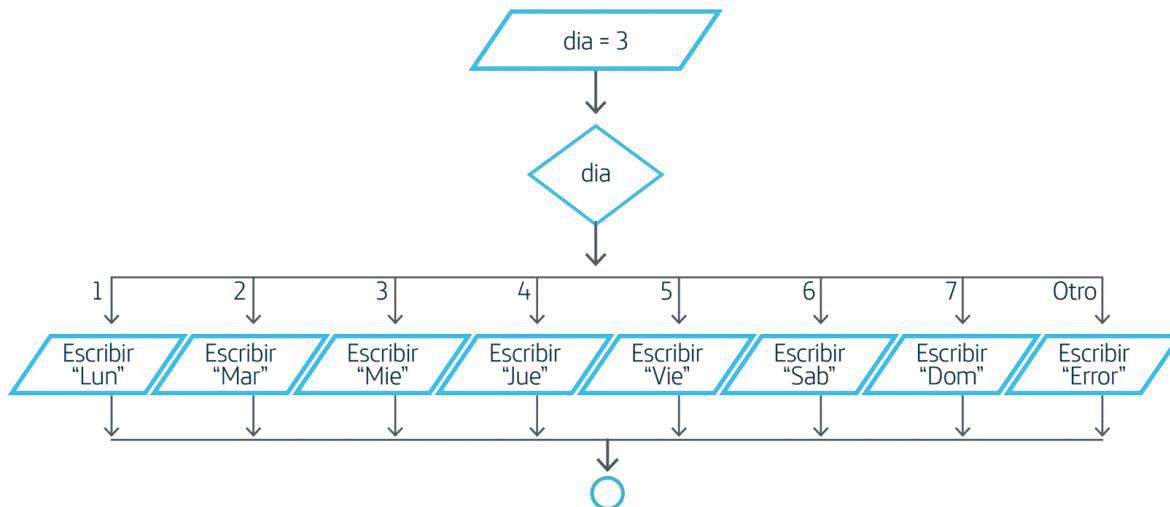
```
int edad = 12;
if (edad<13) {
    System.out.println("Infantil");
}
else if (edad<18){
    System.out.println("Adolescente");
}
else if (edad<40) {
    System.out.println("Joven mayor de edad");
}
else if (edad<65) {
    System.out.println("Adulto mayor de edad");
}
else {
    System.out.println("Jubilado");
}
```

Simplificación de los *if* anidados.

---

## Alternativa múltiple (switch)

Dependiendo del contenido de una variable o del resultado de una expresión se ejecutará un bloque determinado de instrucciones dentro de un conjunto de ellos.



Alternativa múltiple.

Según el valor de la variable *dia*, escribirá una cadena u otra en pantalla.

```
int dia=3;
switch (dia) {
    case 1:
        System.out.println("Hoy es lunes");
        break;
    case 2:
        System.out.println("Hoy es martes");
        break;
    case 3:
        System.out.println("Hoy es miércoles");
        break;
    case 4:
        System.out.println("Hoy es Jueves");
        break;
    case 5:
        System.out.println("Hoy es Viernes");
        break;
    case 6:
        System.out.println("Hoy es Sábado");
        break;
    case 7:
        System.out.println("Hoy es Domingo");
        break;
    default:
        System.out.println("El número " + dia +
                           " no es un dia de la semana");
}
```

Alternativa múltiple en Java.

La sentencia *switch* de Java va evaluando los distintos casos hasta encontrar uno que se cumpla. Cuando uno de los casos se cumple ejecuta todos hasta terminar el *switch*. Por este motivo hay que poner una sentencia *break* (salir de la estructura de paréntesis) para evitar que se ejecuten el resto de los casos.

Si no se ha cumplido ninguno de los casos se ejecutará el bloque *default*.

Prueba a quitar todos los *break* y verás que el programa mostrará lo siguiente:

 Hoy es Miércoles

Hoy es Jueves

Hoy es Viernes

Hoy es Sábado

Hoy es Domingo

El número 3 no es un día de la semana

---

## ¿Y es posible hacer que varios *case* ejecuten el mismo bloque de instrucciones sin repetir el código?

Sí, el siguiente ejemplo agrupa varios casos en un solo bloque de instrucciones:

```
public class Principal {  
    public static void main(String[] args) {  
        int valor=7;  
        switch(valor) {  
            case 1:  
            case 2:  
            case 3:  
                System.out.println("El valor es 1, 2 o 3");  
                break;  
            case 4:  
            case 5:  
            case 6:  
                System.out.println("El valor es 4, 5 o 6");  
                break;  
  
            case 7:  
            case 8:  
            case 9:  
                System.out.println("El valor es 7, 8 o 9");  
                break;  
            default:  
                System.out.println("El valor de mayor que 9")  
        }  
    }  
}
```

## ¿Y la variable o expresión a evaluar tienen que ser siempre un número entero?

No, la expresión a evaluar puede ser de muy diversos tipos:

- **Un número entero.**
- **Un número real**, aunque es menos típico, ya que los *case* no permiten especificar rangos de valores. Por ejemplo *case >=1 && <=10* no está permitido. Es imposible contemplar todos los posibles valores entre 1 y 10 con todas las variaciones en sus decimales.
- **Un char.**
- **Una cadena de caracteres.** Esto es una mejora a partir de la versión 7 de Java, en versiones anteriores no era posible.

Veamos un ejemplo donde el valor a evaluar es un *char*:

```
public class Principal {  
    public static void main(String[] args) {  
        char valor='b';  
        switch(valor) {  
            case 'a':  
                System.out.println("El valor es a");  
                break;  
            case 'b':  
                System.out.println("El valor es b");  
                break;  
            case 'c':  
                System.out.println("El valor es c");  
                break;  
            case 'd':  
                System.out.println("El valor es d");  
                break;  
            default:  
                System.out.println("El valor es distinto de  
                }  
            }  
        }
```

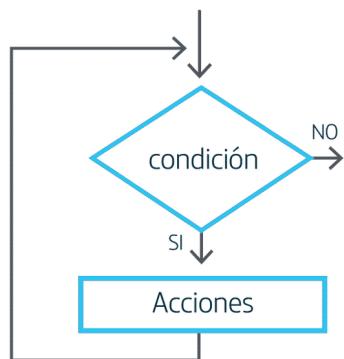
Y ahora vamos a probar con una cadena de caracteres:

```
public class Principal {  
    public static void main(String[] args) {  
        String user = "Juan";  
  
        switch (user) {  
            case "Pepe":  
                System.out.println("Hola Pepe, eres bienven  
                break;  
            case "Juan":  
                System.out.println("¿Qué pasa Juan?, adelan  
                break;  
            case "Luis":  
                System.out.println("Hola Luis, quieres juga  
                break;  
            default:  
                System.out.println("Lo siento "+user+" no p  
        }  
    }  
}
```

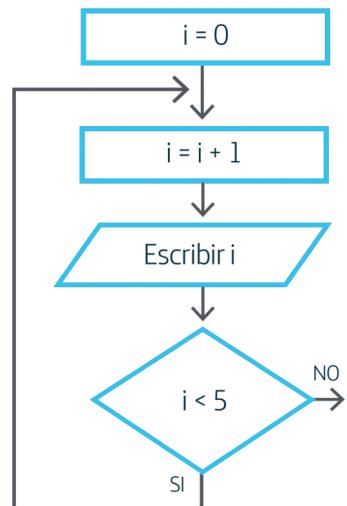
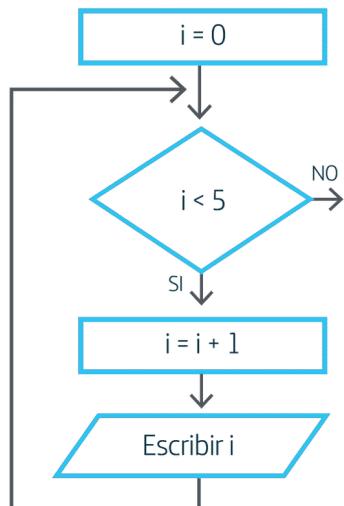
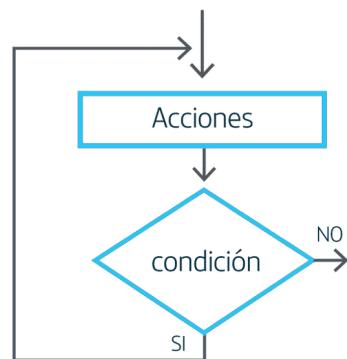
## Estructura mientras (while)

Una estructura *mientras* permite ejecutar un bucle mientras se cumpla una condición.

Mientras (condición / acción)



Mientras (acción / condición)



Estructuras *mientras*.

En la anterior imagen puedes comprobar que existen dos variantes para la estructura *mientras*.

1

**Condición / Acción:** primero evalúa la condición y luego ejecuta el bloque de acciones. En este caso el bucle puede ejecutarse de 0 a infinitas veces.

2

**Acción / Condición:** primero ejecuta el bloque de acciones y luego evalúa la condición. En este caso el bucle puede ejecutarse de 1 a infinitas veces.

La imagen muestra la estructura y un ejemplo para contar del 1 al 5 con las dos variantes.

Veamos ambos ejemplos implementados en Java:

## Condición / Acción: contar del 1 al 5

```
int i=0;
while (i<5) {
    i=i+1;
    System.out.println(i);
}
```

Estructura *mientras* condición/acción en Java.

En este ejemplo el bucle se ejecuta 5 veces con los valores del contador 1 a 5. Observa que con este tipo de estructura el bucle podría ejecutarse 0 veces; por ejemplo, si inicializas previamente la variable *i* con el valor 6 se provoca que nunca se cumpla la condición.

---

## Acción / Condición: contar del 1 al 5

```
int i=0;  
do {  
    i=i+1;  
    System.out.println(i);  
} while (i<5);
```

Estructura *mientras* acción/condición en Java.

---

Con esta variante de la instrucción *while* puedes comprobar que, aunque inicialices la variable *i* con el valor 6, el bucle al menos se ejecuta una vez, ya que primero ejecutará las acciones y luego evaluará la condición.

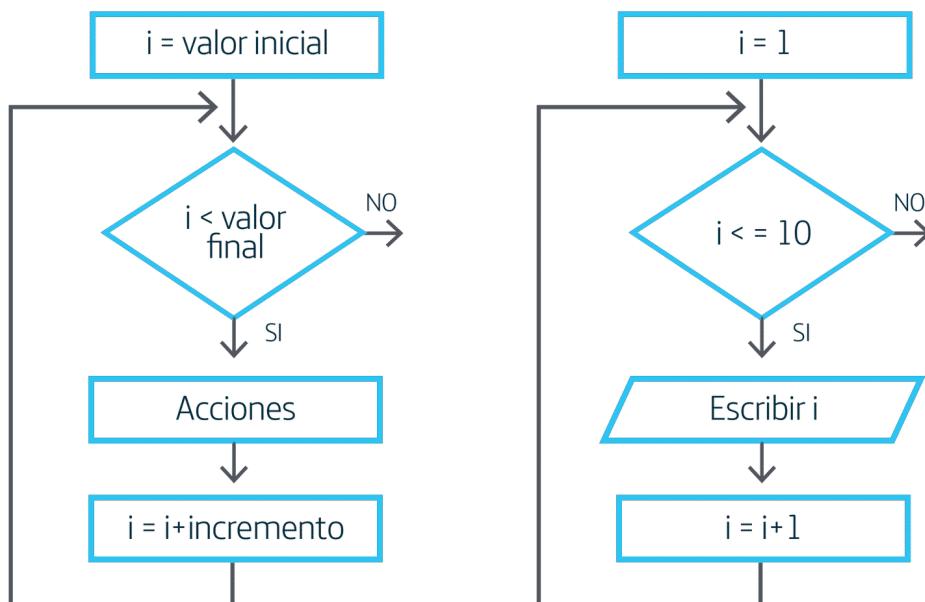
## Estructura para (for)

La estructura *para* es un tipo de **iteración donde siempre interviene un contador.**

En la imagen de más abajo, el ordinograma de la izquierda te muestra el funcionamiento general de la estructura *para (for)*.

El ordinograma de la derecha refleja un programa que escribe en pantalla los números del 1 al 10.

Estructura para (for), siendo i el índice o contador del bucle



---

## El ciclo de una estructura *for* consta de estos pasos:

- 1 Se inicializa un índice o contador con un valor inicial.
- 2 Se evalúa la condición.
- 3 Si la condición ha sido falsa termina la iteración y si la condición ha sido verdadera se ejecutan las acciones del bucle.
- 4 Después de ejecutar las acciones del bucle se incrementa el contador en un valor positivo o negativo.
- 5 Regreso al punto 3 (se evalúa la condición).

El formato Java de la instrucción *for* es así:

```
for (expresión de inicialización; condición; expresión de incremento) {  
    Acciones del bucle  
}
```

Estructura *para (for)* en Java.

---

---

Vamos a ponerlo en práctica contando del 1 al 10 en Java.

```
int i;
for (i=1; i<=10; i++) {
    System.out.println(i);
}
System.out.println(i);
```

Contar del 1 al 10 con una estructura *para (for)* de Java.

El anterior ejemplo inicializa el contador con el valor 1 y luego se irá incrementando hasta alcanzar un valor mayor de 10.

Observa que he colocado una línea “*System.out.println(i);*” después del bucle. Es para que puedas comprobar el valor del índice o contador una vez terminado el ciclo *for*. La variable *i* termina con el valor 11 porque siempre hay un incremento después de las acciones a realizar.

Ahora realiza pruebas cambiando las expresiones de inicialización, condición e incremento. Por ejemplo:

```
int i;
for (i=10; i>=1; i--) {
    System.out.println(i);
}
```

Contar en orden inverso.

## Resumen

---

Has terminado la lección, vamos a ver los puntos más importantes que hemos tratado.

- El teorema del programa estructurado demuestra que “**Todo algoritmo puede ser diseñado empleando solo 3 estructuras básicas**: secuencial, alternativa y repetitiva”.
- La **estructura secuencial** consiste en la ejecución lineal de una instrucción tras otra.
- La **estructura alternativa** permite ejecutar un bloque de instrucciones dependiendo del cumplimiento de una condición. Hay alternativa simple, doble y múltiple.
- La **estructura repetitiva** permite crear iteraciones o bucles (porciones de programa que se repiten un número determinado de veces). Las estructuras repetitivas son: *mientras (while)* y *para (for)*.



**PROEDUCA**