Tema 2:
Estructuras de Control

1º Diseño de Aplicaciones Web

T2. Elementos de un programa informático

1. **Conceptos básicos del flujo de control**
2. **Estructuras de selección.**
3. **Estructuras de repetición.**
4. **Estructuras de salto.**
5. **Control de excepciones.**

1. Conceptos básicos del flujo de control

- Dentro del cuerpo de los métodos se ejecutan **sentencias** (instrucciones) que determinan el comportamiento del programa

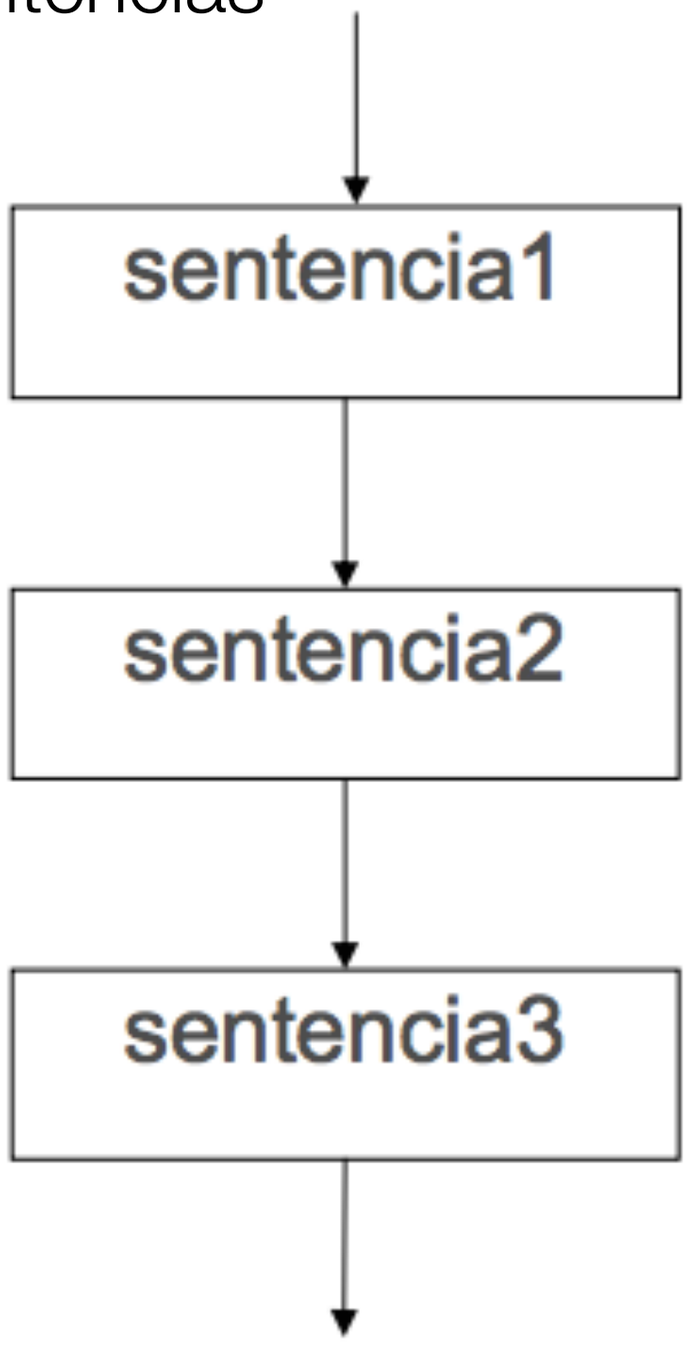
- Dichas sentencias se ejecutan en **secuencia**

- Cada **sentencia** termina con un **punto y coma**

- Las sentencias que hemos visto hasta ahora eran sencillas (**asignación**)

1. Conceptos básicos del flujo de control

Secuenciación de sentencias



1. Conceptos básicos del flujo de control

Un **bloque de sentencias** (o sentencia compuesta) es un número de sentencias Java simples rodeadas por llaves

- Los bloques definen un ámbito de variables
- Los bloques pueden anidarse

```
void metodo() {
   int n;
   ...
   { int k;
     ...
   }
...
}
```

1. Conceptos básicos del flujo de control

- Los lenguajes de programación proporcionan **sentencias de control** que permiten
  - Ejecución **opcional** de un bloque de sentencias
  - Ejecución **repetida** de un bloque de sentencias
- De esta forma, se pueden implementar algoritmos para calcular los valores resultantes

2. Estructuras de selección

- Las estructuras de selección permiten, en **función del valor lógico** de un selector, **ejecutar un bloque de sentencias u otro**

- Tipos de sentencias de selección:

    - Sentencia **if** / **if-else**

    - Sentencia **switch**

2. Estructuras de selección

La sentencia if es la sentencia básica de selección

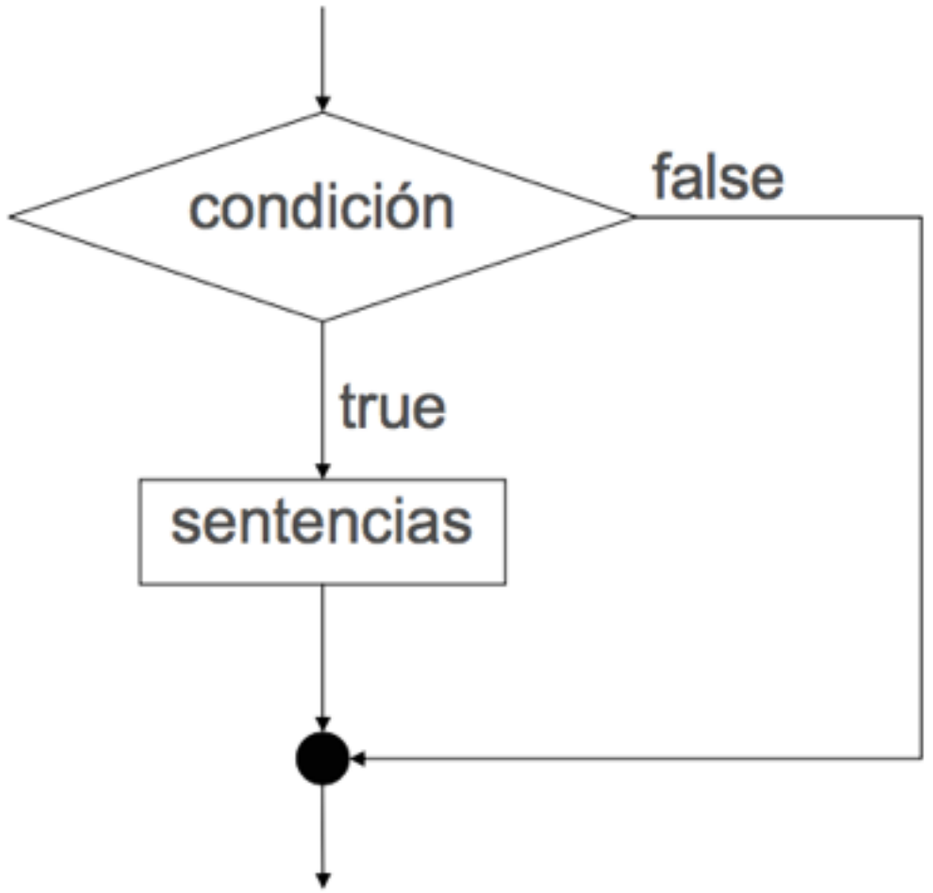
```
if (condición) {
    bloque de sentencias
}
```

donde

- **condición** es una expresión booleana
- **sentencias** representa un bloque de sentencias o una sentencia única
    - Si es una sentencia única, se pueden quitar las llaves

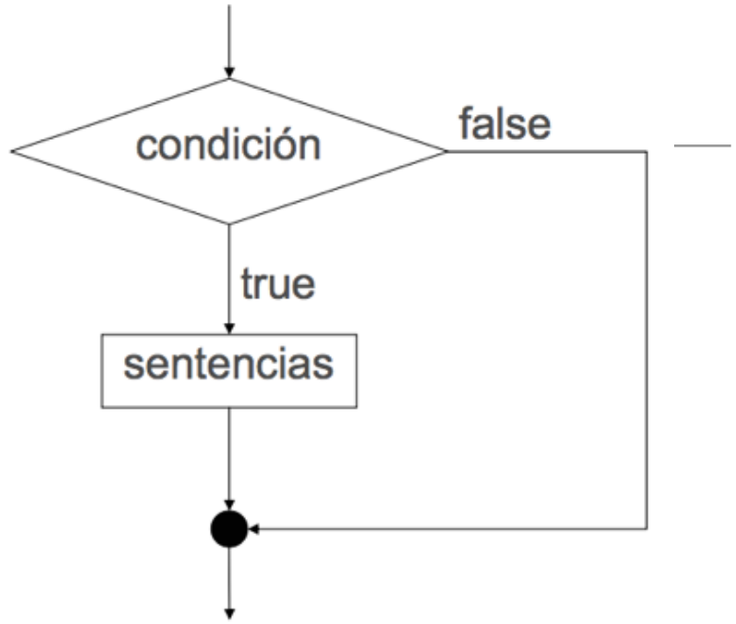
2. Estructuras de selección

Diagrama de la sentencia **if**



- Si la **condición** es **cierta** (si el valor de la expresión es true) se ejecutará el bloque de sentencias asociado
- Si la **condición** es **falsa**, dicho bloque de sentencias no se ejecutará

2. Estructuras de selección



En la **condición** se suelen utilizar:

operadores de **comparación**:

==, >=, <=, >, <, !=

operadores **lógicos**:

cond1 **&&** cond2 es verdad si ambas condiciones son ciertas

cond1 **||** cond2 es verdad si alguna de las condiciones es cierta.

!condition es verdad si la condición es falsa.

2. Estructuras de selección

```
public void main (String args[]){
    if (nombre == null)
        System.out.println(“Nombre vacío”);
    if (año < 0)
        System.out.println(“Año incorrecto”);
}
```

¿Cuál sería el diagrama de este código?

2. Estructuras de selección

```
public void main (String args[]){
    int number = 55;

    if (number != 0) {
        System.out.println("The number was not equal to 0");
    }

    if (number >= 1000) {
        System.out.println("The number was greater than or
        equal to 1000");
    }
}
```

¿Cuál sería el diagrama de este código?

2. Estructuras de selección

```
public void main (String args[]){
    int first = 1;
    int second = 3;

    boolean isLesser = first < second;

    if (isLesser) {
            System.out.println(first + " is less than " + second
            + "!");
    }
}
```

¿Cuál sería el diagrama de este código?

2. Estructuras de selección

La sentencia if-else

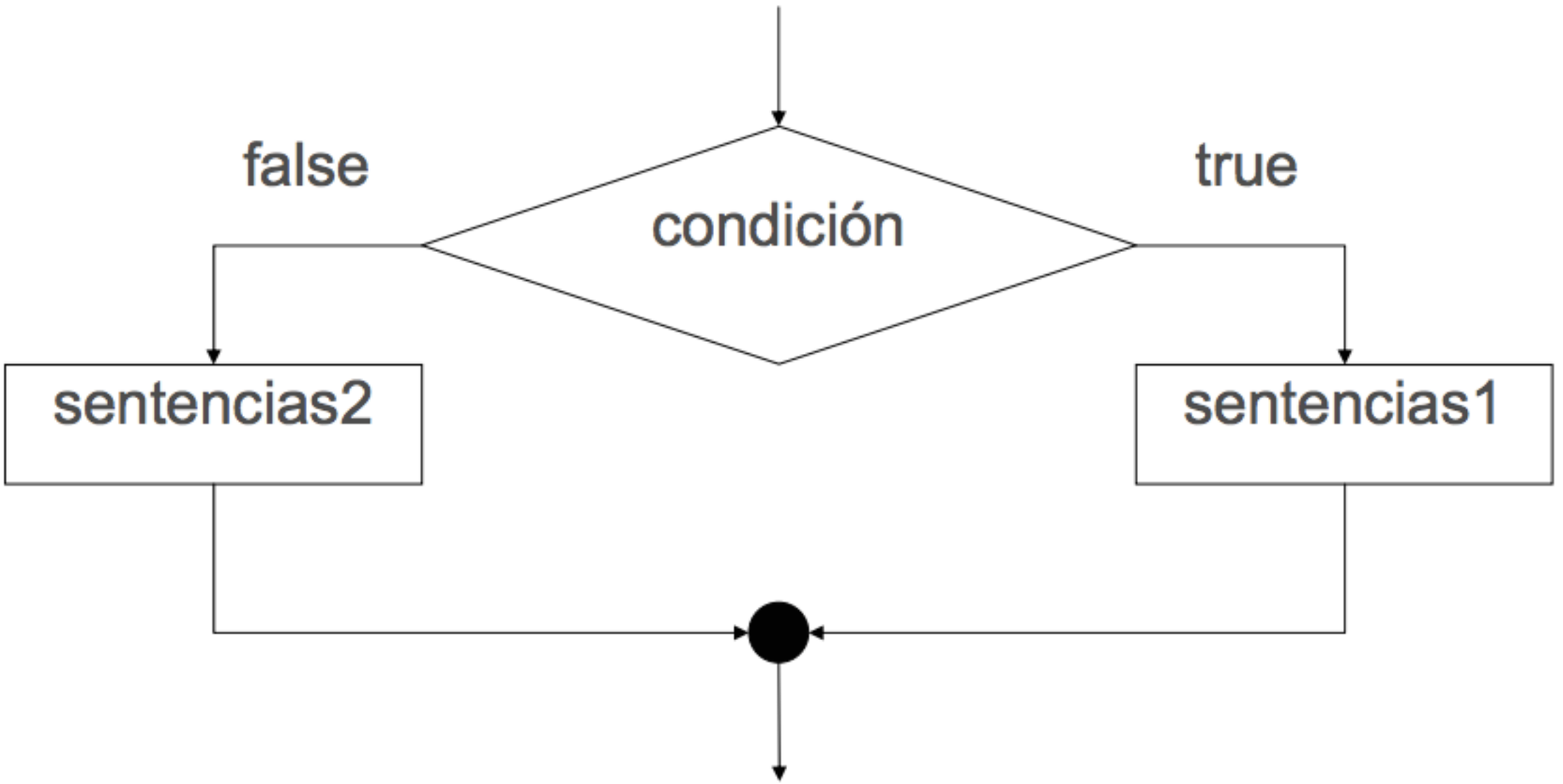
- Opcionalmente, se puede añadir una parte **else** a la sentencia **if**

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

- Indica otro bloque de sentencia a ejecutar si la condición es falsa

2. Estructuras de selección

Diagrama de la sentencia **if-else**



2. Estructuras de selección

```
public void main (String args[]){
    Scanner reader = new Scanner(System.in);
    System.out.println(“Introduce un valor: ”);
    int a = reader.nextInt();
    System.out.println(“Introduce otro valor: ”);
    int b = reader.nextInt();
    if (a>b) {
        System.out.println(“El número mayor es “ + a);
    } else {
        System.out.println(“El número mayor es “ + b);
    }
}
```

2. Estructuras de selección

Escribir un programa en java que pida introducir una nota por teclado y devuelva un mensaje diciendo si el examen está aprobado o suspendido.

Dibuja primero el diagrama de flujo y después codifica el programa.

2. Estructuras de selección

La sentencia if anidada

- Las sentencias if se pueden anidar. También las otras sentencias de control que veremos a continuación

- Dentro del bloque de sentencias del if (o de los bloques de if-else) puede encontrarse otra sentencia if

- **Importante**

  - A qué **if** corresponde cada **else**

  - Cuál es el estado del programa en cada punto

  - ¡Una buena indentación!

2. Estructuras de selección

```
if (i == 10) {

    if (j < 20) a = b;

    if (k > 100) c = d;

    else a = c;

}

else a = d;
```

A que ifs pertenecen?

2. Estructuras de selección

La sentencia if-else-if

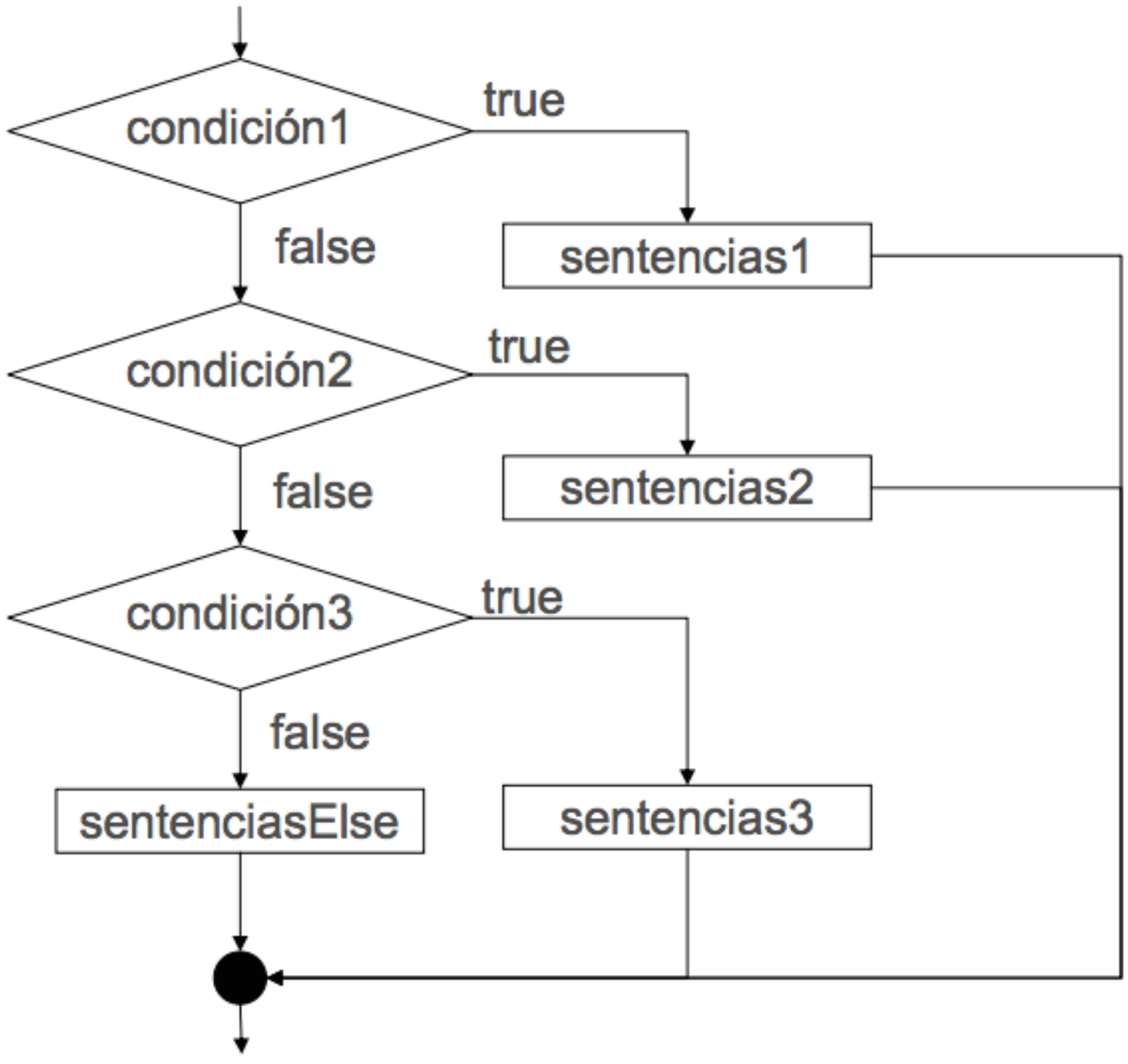
- Habitual cuando hay más de una condición.

- Basada en los if anidados, se anidan en la parte del **else**

```
if (condición1) {
   sentencias1
} else if (condición2) {
   sentencias2
} else if (condición3) {
   sentencias3
}
...
else {
   sentenciasElse
}
```

2. Estructuras de selección

Diagrama de la sentencia **if-else-if**



2. Estructuras de selección

```
public void main (String args[]){
    Scanner reader = new Scanner(System.in);
    System.out.println("Introduce un mes: ");
    int mes = reader.nextInt();
    String estacion;

     if (mes == 12 || mes == 1 || mes == 2)
        estacion = "Invierno";
    else if (mes == 3 || mes == 4 || mes == 5)
        estacion = "Primavera";
    else if (mes == 6 || mes == 7 || mes == 8)
        estacion = "Verano";
    else if (mes == 9 || mes == 10 || mes == 11)
        estacion = "Otoño";

    System.out.println("La estación es: "+estacion);
}
```

2. Estructuras de selección

La sentencia switch

- Selección de bloques de sentencias entre múltiples casos dependiendo del valor de una expresión entera.

```
switch (expresion) {
   case valor1:
     sentencias;
     break;
   case valor2:
   case valor3:
     sentencias;
     break;
   …
   default:
     sentencias;
     break;
}
```

2. Estructuras de selección

La sentencia switch

- Puede haber default o no

- Si un bloque contiene un break, se continua con la sentencia siguiente al switch

- Si no, continúa comprobando los siguientes valores, o va al default

- Los valores tienen que ser una expresión constante conocida en tiempo de ejecución

2. Estructuras de selección

```
public static void main (String[] args) {

        int dias;
        int mes; = 8;
        switch (mes) {
            case 4:
            case 6:
            case 8:
            case 10:
                dias = 30;
                break;
            case 2:
                dias = 28;
                break;
            default:
                días = 31;
                break;
        }
        System.out.println(“El mes “ + mes + “ tiene “ + dias + “ días.”);

}
```