# UF.1
# Sentencias de Control Java


JAVA™

Universidad Europea de Madrid

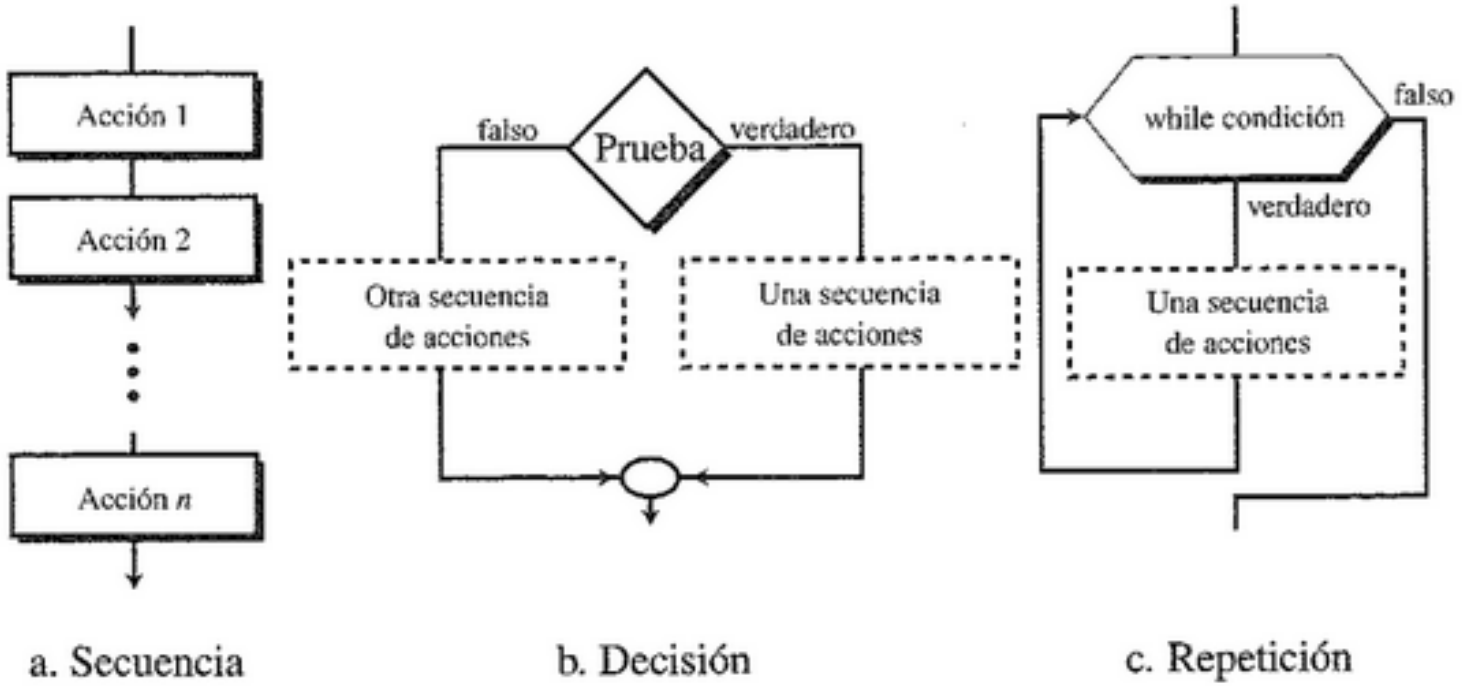
LAUREATE INTERNATIONAL UNIVERSITIES

# CONTENIDOS

1. Introducción
2. Tipos
    a. Selección
        1. If
        2. switch
    b. Iteración
        1. while
        2. do while
        3. For
3. Control de flujo y bucles

## Sentencias de Control

Nos permite cambiar el orden de las instrucciones ejecutadas en nuestros programas.

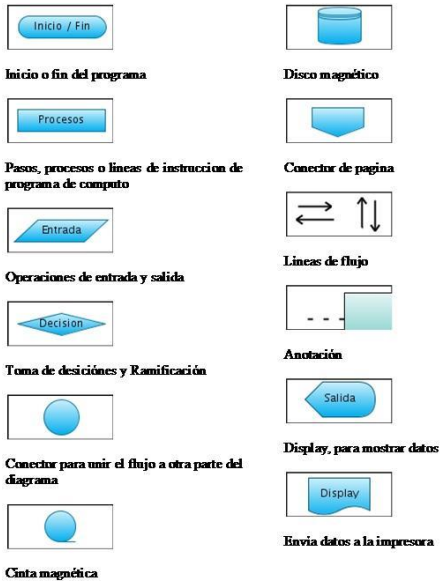


# TIPOS

## Sentencias de Control

Hay dos tipos de sentencias o estructuras de control:

1. Estructuras de selección / decision control structures
   Nos permite seleccionar secciones específicas del código para ser ejecutado, a partir de una condición.
2. Estructuras de iteración / repetition control structures
   Nos permite ejecutar secciones específicas del código una cantidad determinada de veces.



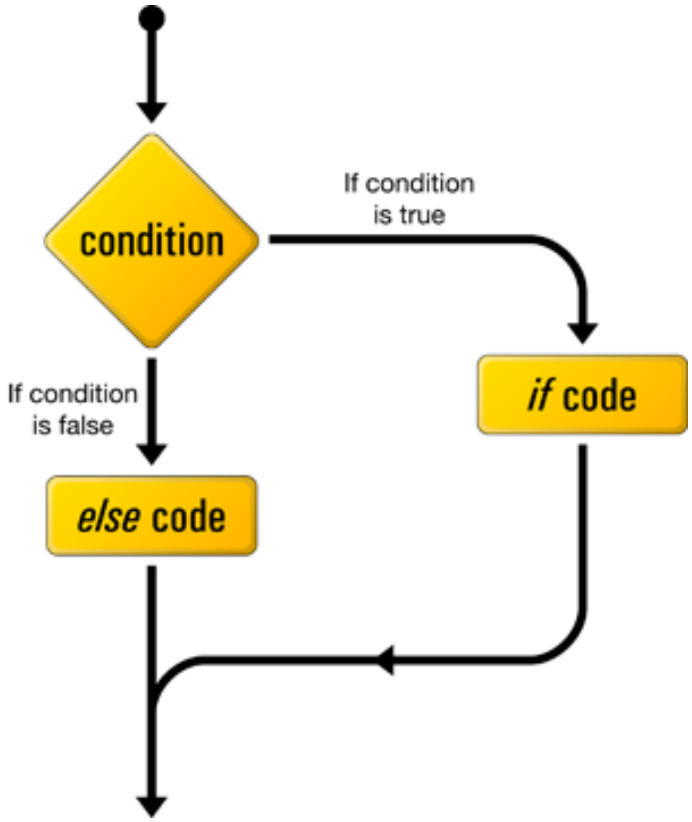
# SELECCIÓN - IF

Instrucciones condicionales

Declaraciones que nos permiten seleccionar y ejecutar bloques específicos del código mientras otras partes son ignoradas.

Tipos:

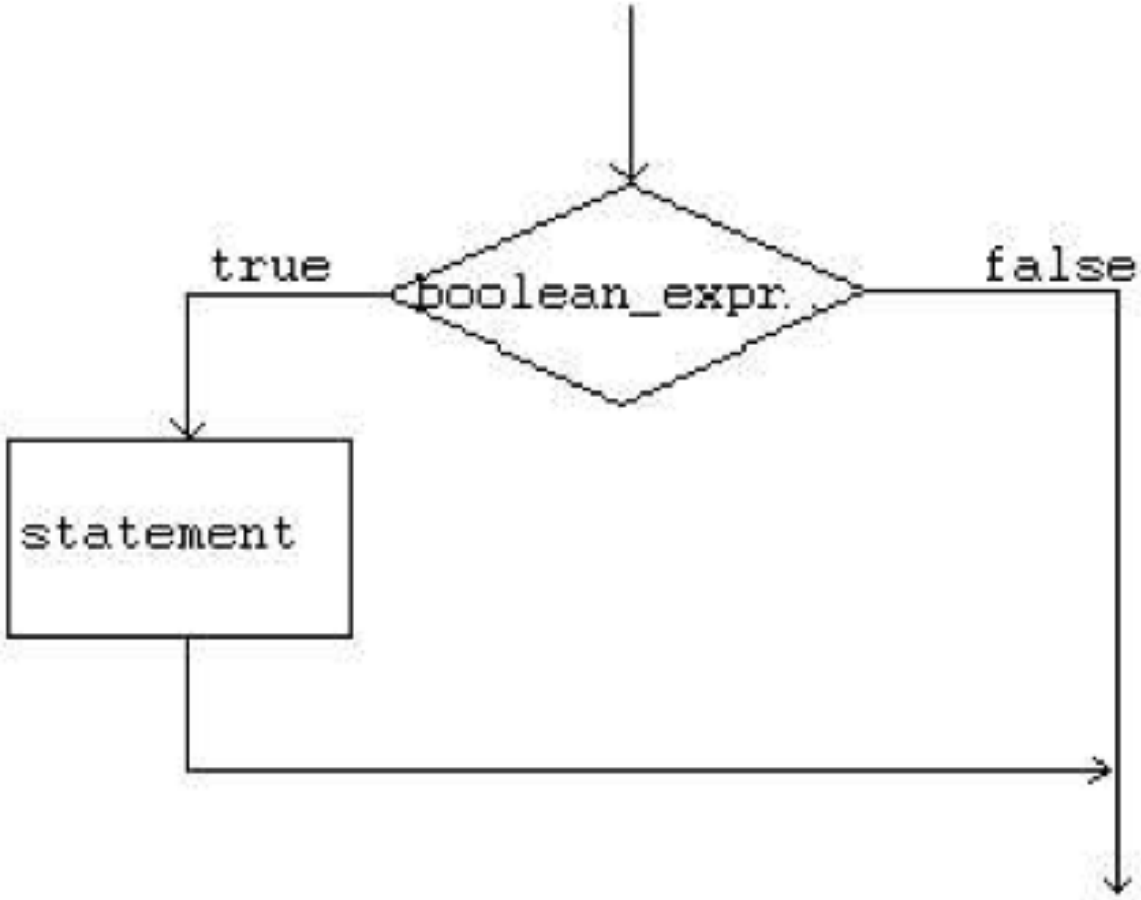
```
if (boolean_expression)
        statement;

if (boolean_expression){
        statement1;
   }
   else{
        statement2;
   }

if (boolean_expression1)
        statement1;
   else if (boolean_expression2)
        statement2;
   else
        statement3
```



## SELECCIÓN - IF

Instrucciones condicionales

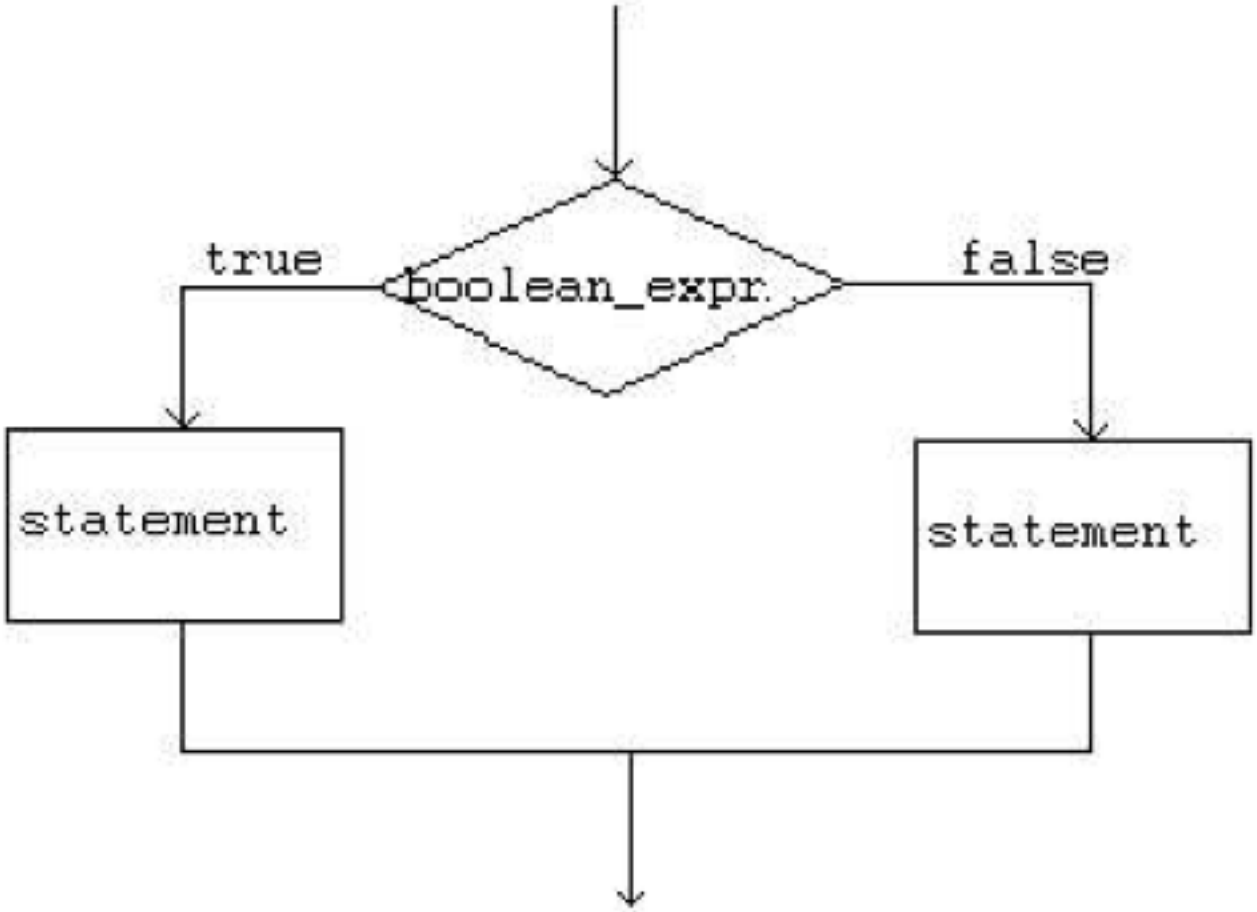
Diagramas **if**:



## SELECCIÓN - IF

Instrucciones condicionales

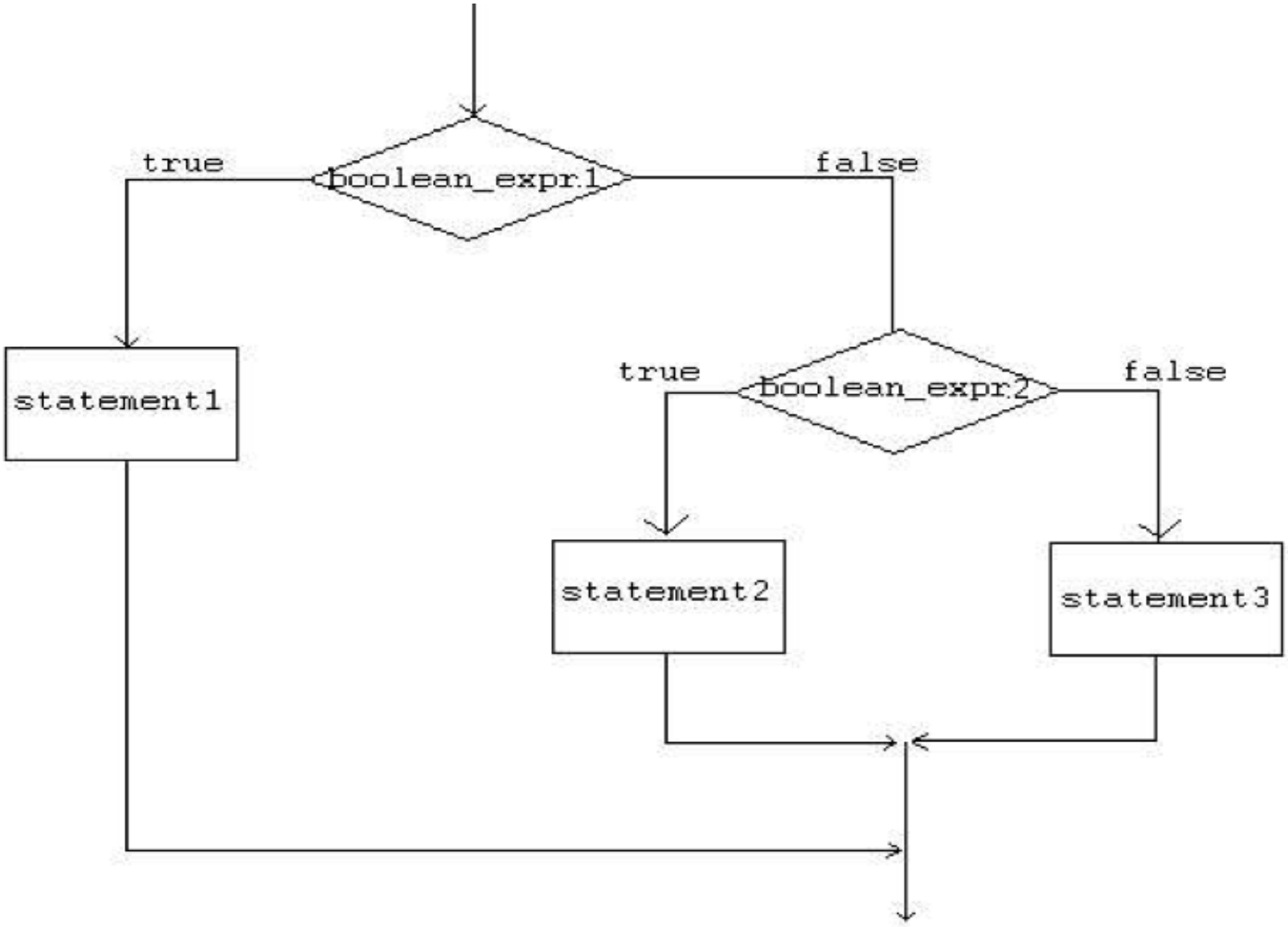
Diagramas **if else**:



# SELECCIÓN - IF

Instrucciones condicionales

Diagramas **if else if**:



Instrucciones condicionales

**Errores comunes**

1. La condición dentro de la declaración If no evalúa un valor boleano. Por ejemplo:

```
//WRONG
int number = 0;
if (number) {
                    //some statements here
}
```

La variable number no es un tipo booleano

2. Escribir elseif en vez de else if.

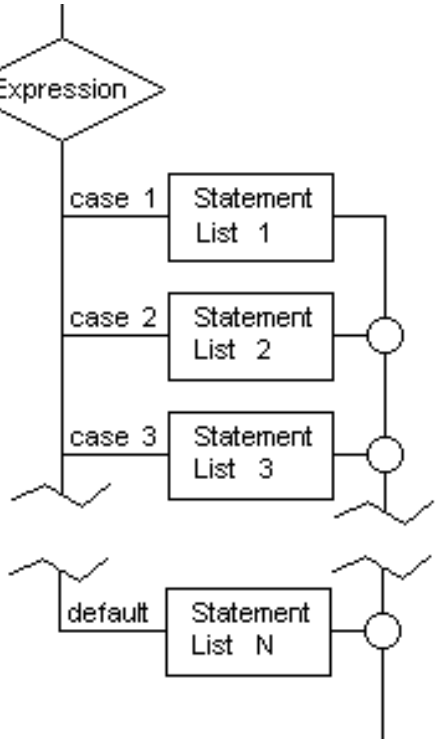
# SELECCIÓN - SWITCH

Instrucciones selección

La sentencia switch se encarga de estructurar una selección múltiple. Al contrario del enunciado if-else que sólo podemos indicar dos alternativas, maneja un número finito de posibilidades

La estructura general del enunciado switch es la siguiente:

```
switch  (expresión) {
        case constante1:
                sentencia1;
                ...
                break;
        ...
        case constanteN:
                sentenciaN;
                ...
                break;
        default:
                sentencia;
                ...
                break
}
```



# SELECCIÓN - SWITCH

Instrucciones selección

Declaración del **switch**

- El valor de la expresión y de las constantes tiene que ser de tipo char, byte, short o int .
- Al evaluar la expresión de switch, el intérprete busca una constante con el mismo valor.
- Si la encuentra, ejecuta las sentencias asociadas a esta constante hasta que tropiece con un break.
- La sentencia break finaliza la ejecución de esta estructura.
- Si no encuentra ninguna constante que coincida con la expresión, busca la línea default.
- Si existe, ejecuta las sentencias que le siguen. La sentencia default es opcional.

# SELECCIÓN - SWITCH

Instrucciones selección

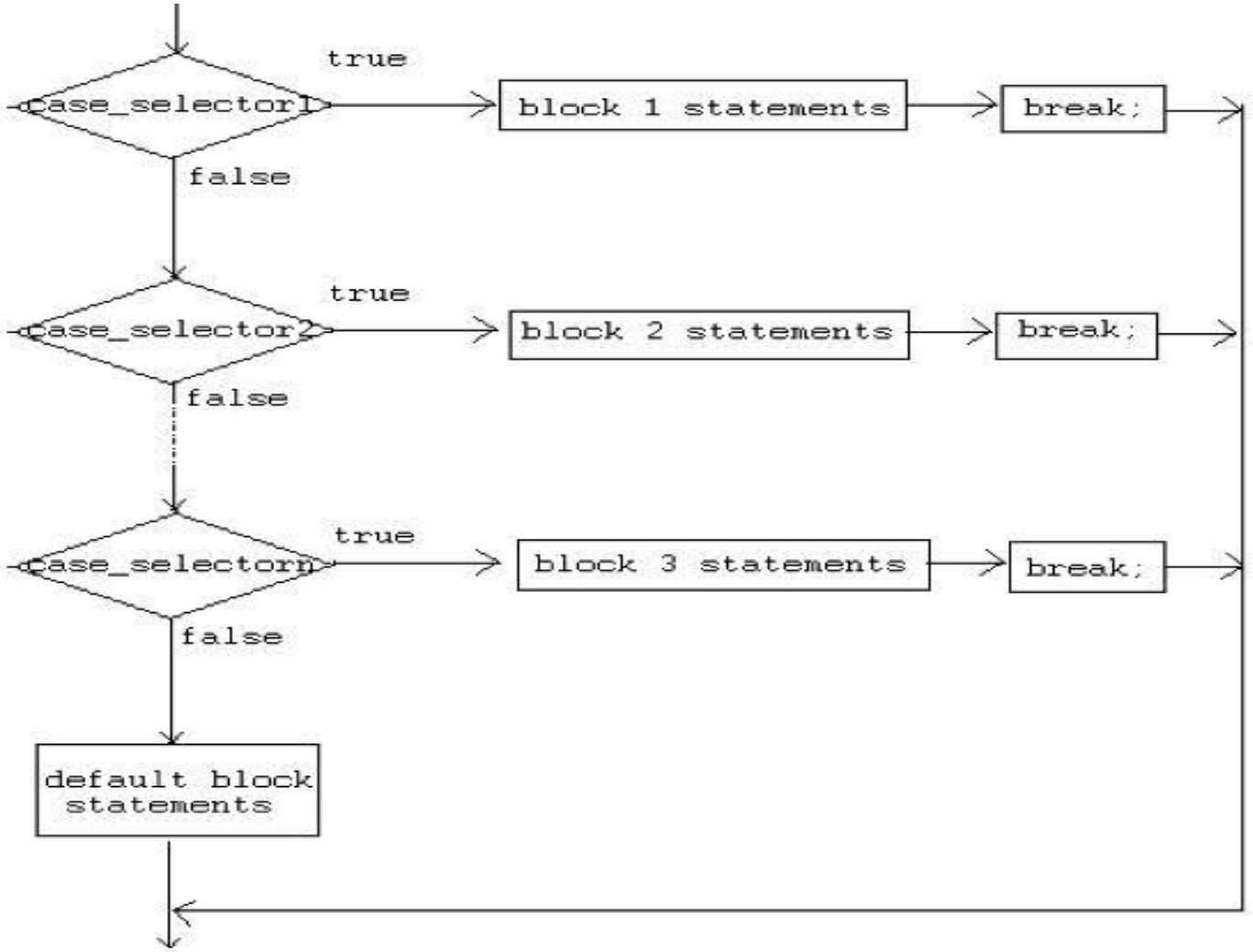
Ejemplo de **switch**

```
public class Lineas{
   public static void main(String args[]){
     int j = 0;
     switch (j) {
     case 5:
       System.out.println("5********");
     case 4:
       System.out.println("4********");
       break;
     case 3:
       System.out.println("3********");
     case 2:
       System.out.println("2********");
     case 1:
       System.out.println("1********");
     default:
       System.out.println(“Por defecto");
     }
   }
}
```

# SELECCIÓN - SWITCH

Instrucciones selección

Diagrama de **switch**



Instrucciones selección

**Errores comunes**

•Cuando un case en un switch se ha encontrado la misma condición, todas las declaraciones relacionadas con este case se ejecutan. No sólo eso, las declaraciones relacionadas con los cases siguientes son también ejecutados.

•Para evitar que el programa siga ejecutando los case posteriores, usamos la declaración break como última declaración.

# ITERACIÓN

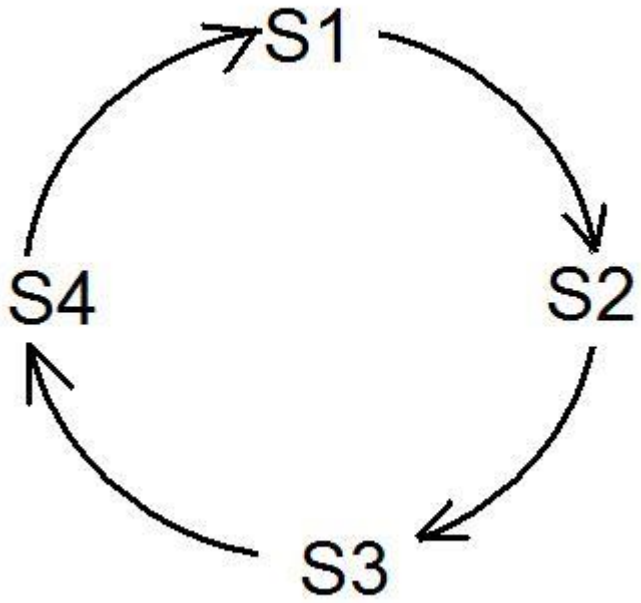
Sentencias repetitivas o bucles o ciclos

Estructuras o sentencias de iteración

•Nos permite ejecutar secciones específicas del código una cantidad determinada de veces.

•Tipos:

- while
- do-while
- for



# ITERACIÓN - WHILE

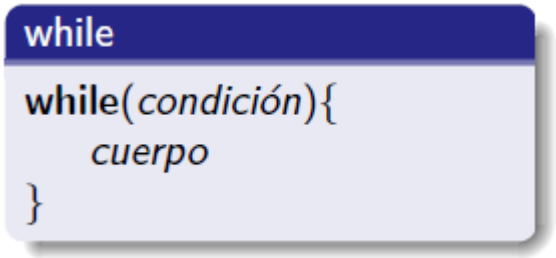
Sentencias repetitivas

Sentencia **while**

La iteración continuará hasta que su condición sea falsa.

while tiene la siguiente forma o sintaxis:

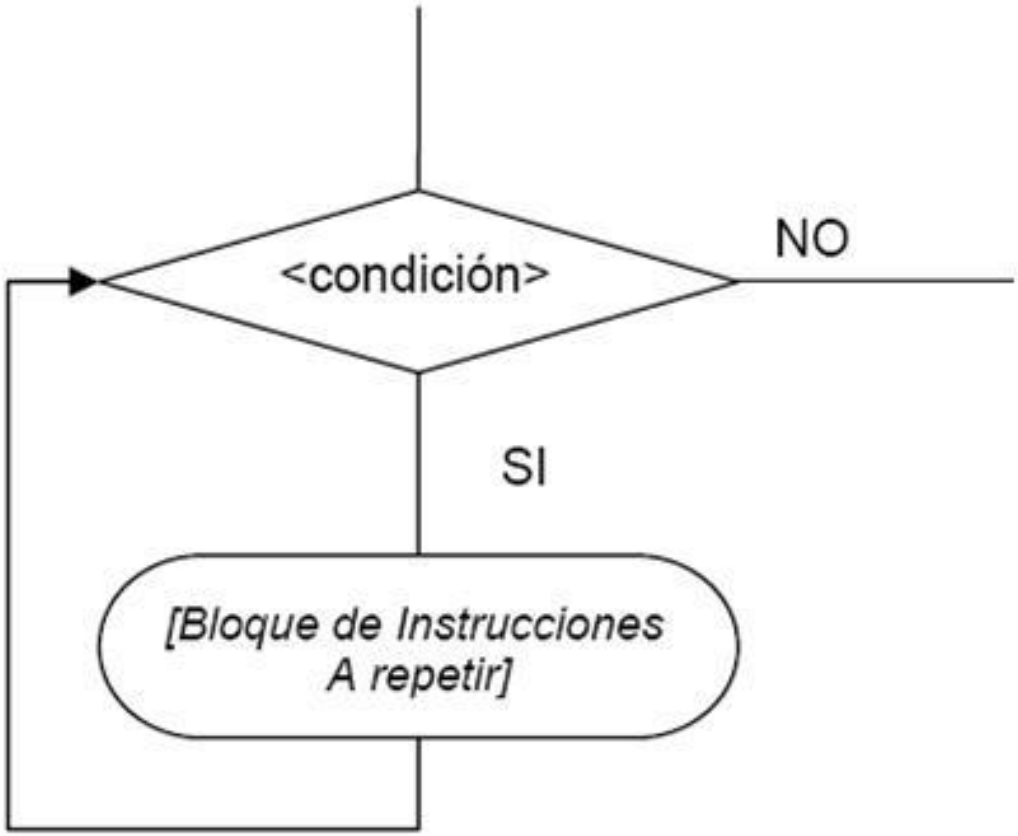
```
while (boolean_expression) {
        statement1;
        statement2;
        . . .
}
```



# ITERACIÓN - WHILE

Sentencias repetitivas

Diagrama **while**



# ITERACIÓN - WHILE

Sentencias repetitivas

Ejemplo 1

```
//Loop
int x = 0;
while (x<10)  {
        System.out.println(x);
        x++;
}
```

¿Qué hace?

# ITERACIÓN - WHILE

Sentencias repetitivas

Ejemplo 2

```
//infinite loop
int x = 0;
while (x<10)  {
        System.out.println(x);
}
```

¿Por qué?

# ITERACIÓN - WHILE

Sentencias repetitivas

Ejemplo 2

```
//No loops
int x = 10;
while (x<10)  {
        System.out.println(x);
        x++;
}
```

¿Por qué?

# ITERACIÓN – DO WHILE

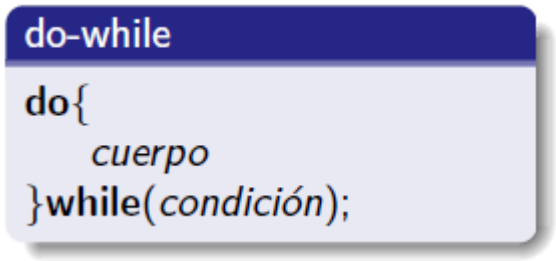
Sentencias repetitivas

Sentencia **do - while**

La sentencia de iteración do-while es de tipo posprueba.
Primero realiza las acciones y luego pregunta.

Do - while tiene la siguiente forma o sintaxis:

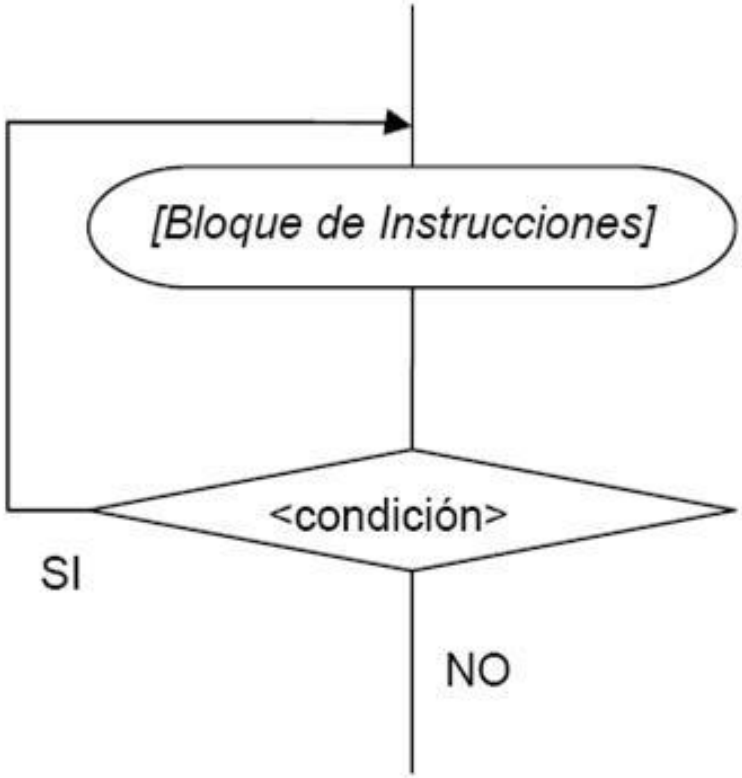
```
do{
        statement1;
        statement2;
        . . .
} while (boolean_expression);
```



# ITERACIÓN - WHILE

Sentencias repetitivas

Diagrama **do while**



# ITERACIÓN – DO WHILE

Sentencias repetitivas

Ejemplo 1

```
int x = 0;
do  {
        System.out.println(x);
        x++;
} while (x<10);
```

¿Qué hace?

# ITERACIÓN – DO WHILE

Sentencias repetitivas

Ejemplo 2

```
//infinite loop
int x = 0;
do  {
        System.out.println(x);
        x++;
} while (x>0);
```

¿Por qué?

# ITERACIÓN – DO WHILE

Sentencias repetitivas

Ejemplo 2

```
//one loop
int x = 0;
do  {
        System.out.println(x);
        x++;
} while (x<0);
```

¿Por qué?

# ITERACIÓN – DO WHILE

Sentencias repetitivas

**Errores comunes**

•Un error de programación común se da cuando se utiliza el bucle do-while y se olvidar de escribir el punto y coma después de la expresión while:

```
do{
    ...
 }while(boolean_expression)//WRONG->forgot semicolon;
```

•Al igual que en el while, controlar entrar en u “bucle infinito”

# ITERACIÓN – DO WHILE

Sentencias repetitivas

Sentencia **for**

Permite la ejecución del mismo código un número de veces.

For tiene la siguiente forma o sintaxis:

**for** (<inicialización>;<condición>;<actualización>)
<bloque-o-instrucción>

**for**

```
for(inicialización; condición; iteración){
    cuerpo
}
```

Siempre equivalente a un bucle while
Como en el caso de do-while, muchas veces un bucle for es más compacto que un while

# ITERACIÓN - FOR

Sentencias repetitivas

Diagrama **for**