

# UD2.- Estructures Alternatives i Repetitives

(Conditionals i Bucles)

## Estructures Alternatives (Conditionals)

Les estructures alternatives són construccions que permeten alterar el flux seqüencial d'un programa de manera que, en **funció d'una condició o el valor d'una expressió**, el mateix pugui ser desviat en l'una o l'altra alternativa de codi.

Les estructures alternatives disponibles a Java són:

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

### 1.- Alternativa Simple (Sentència IF)

La sentència **if** permet l'execució d'una sèrie d'instruccions en funció del resultat d'una **expressió lògica**. El resultat d'avaluar una expressió lògica és sempre vertader (true) o fals (false).

És molt simple, en llenguatge natural seria: "si aquesta condició és **vertadera** llavors realitza les següents accions".

Codi	Ordinograma
<pre> if (condició) {     // Accions; } </pre> <p>El bloc <b>Accions</b> s'executa si la condició (expressió lògica) s'avalua a <b>true (és vertadera)</b>.</p> <pre> int cont=0 ; if (cont == 0){     System.out.println("cont és 0");     // més instruccions... } </pre> <p>Si dins del <b>if</b> només hi ha una instrucció, no és necessari posar les <b>claus</b>.</p> <pre> int cont=0 ; if (cont == 0)     System.out.println("cont és 0"); </pre>	<pre> graph TD     Entry(( )) --&gt; Condicion{Condicion}     Condicion -- Si --&gt; Acciones[Acciones]     Acciones --&gt; Connector(( ))     Condicion --&gt; Connector     Connector --&gt; Exit(( )) </pre>

## 2.- Alternativa Doble (Sentència IF-ELSE)

La sentència **if-else** permet l'execució d'una sèrie d'instruccions en funció del resultat d'una **expressió lògica**. El resultat d'avaluar una expressió lògica és sempre vertader (true) o fals (false).

És molt simple, en llenguatge natural seria: "si aquesta **condició** és **vertadera** llavors fes això, sinó fes allò d'altre".

Codi	Ordinograma
<pre> if (condició) {     // AccionsSI; }  else {     // AccionsNO; } </pre> <p>El bloc <b>AccionsSI</b> s'executa si la <b>condició</b> s'avalua a <b>true</b> (vertadera). En cas contrari, s'executa el bloc de <b>AccionsNO</b>.</p> <pre> int cont=0 ; if (cont == 0){     System.out.println("cont és 0");     // més instruccions... } else {     System.out.println("cont NO és 0");     // més instruccions... } </pre> <p>Si dins del <b>if</b> o del <b>else</b> només hi ha una instrucció, no és necessari posar les <b>claus</b>.</p> <pre> if (cont == 0)     System.out.println("cont és 0"); else     System.out.println("cont NO és 0"); </pre>	<pre> graph TD     Start(( )) --&gt; Condicion{Condicion}     Condicion -- NO --&gt; Acciones1[Acciones]     Condicion -- SI --&gt; Acciones2[Acciones]     Acciones1 --&gt; Merge(( ))     Acciones2 --&gt; Merge     Merge --&gt; End(( )) </pre>

**NOTA:** l'operador relacional per a comprovar si són iguals és **==**, no un sol **=** que correspon amb l'operador d'assignació. Aquest error no el detecta el compilador i és difícil d'esbrinar.

En moltes ocasions, es nien estructures alternatives **if-else**, (una dins de l'altra) de manera que es pregunte per una condició si anteriorment no s'ha complit una altra successivament.

**Per exemple** : suposem que realitzem un programa que mostra la nota d'un alumne en la forma (insuficient, suficient, bé, notable o excel·lent) en funció de la seua nota numèrica. Podria codificar-se de la següent forma:

```
import java.util.Scanner;

/* En este exemple suposem que l'usuari introdueix el número correctament.
 * No es realitza comprovació. */

public class NotaAlumne {

    public static void main(String args[]) {

        // Declarem variables i el constructor d'Scanner

        Scanner entrada = new Scanner(System.in);
        int nota;

        // Llegim la nota des de teclat

        System.out.println("Introdueix una nota entre 0 y 10");
        nota = entrada.nextInt();

        if (nota < 5 ) {
            System.out.println("Insuficient");
        } else if (nota < 6) {
            System.out.println("Suficient");
        } else if (nota < 7) {
            System.out.println("Bé");
        } else if (nota < 9) {
            System.out.println("Notable");
        } else {
            System.out.println("Excel·lent");
        }

    } // Del main()

} // De la classe
```

```
Console ✕
<terminated> NotaAlumne [Java Application] C
Introdueix una nota entre 0 y 10
6
Bé
```

OJ=. La clausula **else** (si nó)

```
if (nota < 5 )
    System.out.println("Insuficient");

if (nota < 6)
    System.out.println("Suficient");

if (nota < 7)
    System.out.println("Bé");

if (nota < 9)
    System.out.println("Notable");

if (nota <=10)
    System.out.println("Excel·lent");
```

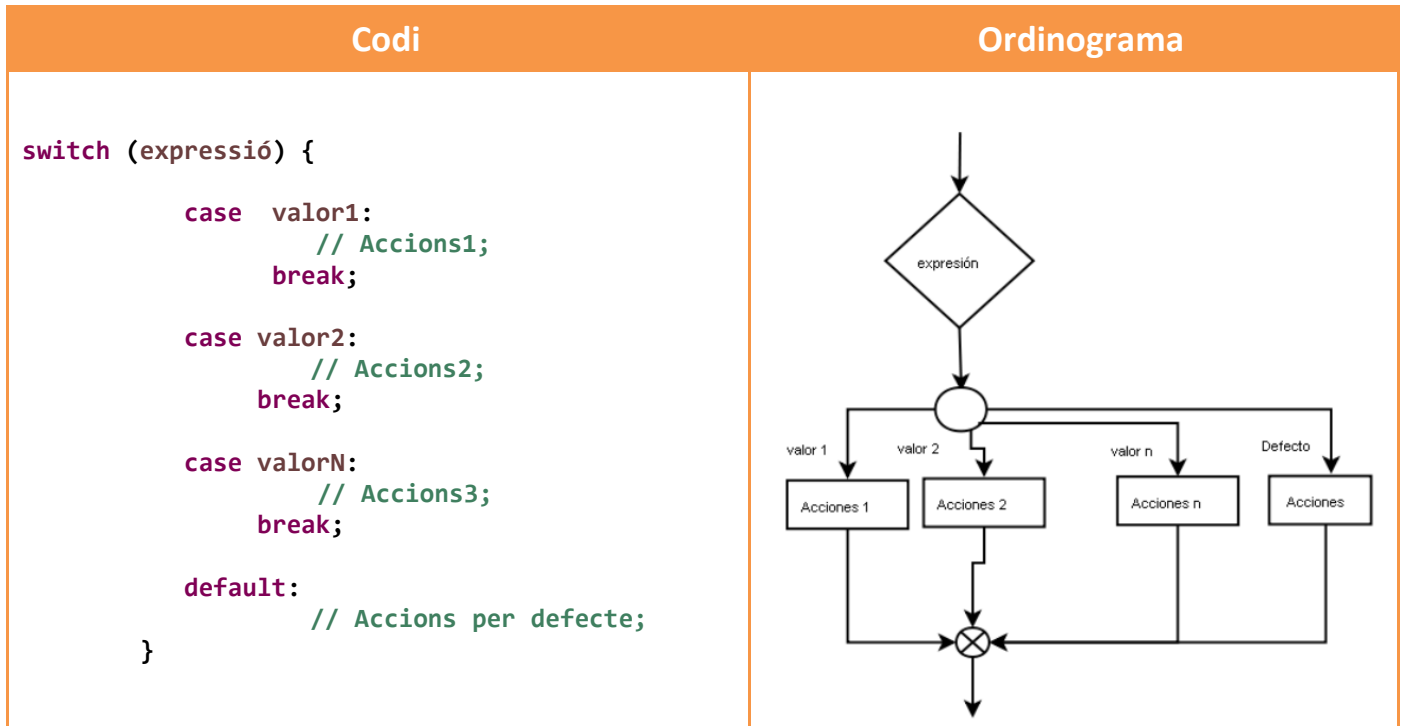
```
Console ✕
<terminated> NotaAlumne [Java Application] C:
Introdueix una nota entre 0 y 10
6
Bé
Notable
Excel·lent
```

## 2.- Alternativa Múltiple (Sentència SWITCH)

A vegades és necessari comparar el valor d'una **variable** amb una sèrie de valors concrets. La selecció múltiple és molt semblant (encara que no és exactament igual) a una seqüència de diverses sentències if com les del exemple anterior.

En llenguatge natural seria alguna cosa així com “Si **variable** val **valor1** llavors entra per **case valor1:**, si **variable** val **valor2** llavors entra per **case valor2:**,... si **variable** no val cap dels valors que hi ha en els diferents **case** llavors entra per **default:**

El format de **switch** és el que es mostra a continuació.



És molt important entendre que en el **switch** s'avalua una **expressió** (un valor concret com 0, 5, 1...) no una condició (**true** o **false**) com en el **if** i el **if-else**.

El programa comprova el valor de **expressió** i saltarà al **case** que corresponga amb aquest valor (**valor1** o **valor2** o ...) executant el codi de dita **case** (**Accions1** o **Accions2** o ...). Si no coincideix cap valor, saltarà al **default** i executarà les **Accions per defecte**.

És important afegir la sentència **break**; al final de cada **case**, ja que en cas **contrari el programa continuarà executant el codi de les altres accions** i normalment no voldrem que faci això (encara que Java permet fer-ho, és confús i per això està desaconsellat).

**Exemple:** Programa que demana un número (i) per teclat, si el número és 0 mostra el missatge "i és zero.", si el número és 1 mostra el missatge "i és u.", si el número és 2 mostra el missatge "i és dos.", en qualsevol altre cas mostra el missatge "el número és diferent a 0,1 o 2").

```
import java.util.Scanner;

public class ExempleSwitch {

    public static void main(String args[]) {

        // Declarem la variable entera i, i el constructor d'Scanner

        int i;
        Scanner entrada = new Scanner(System.in);

        // Llegim un número per teclat i l'assignem a la variable "i"

        System.out.println("Introdueix un número");
        i = entrada.nextInt();

        switch (i)
        {
            case 0:
                System.out.println("i és zero.");
                break;


            case 1:
                System.out.println("i és u.");
                break;


            case 2:
                System.out.println("i és dos.");
                break;


            default:
                System.out.println("el número és diferent a 0,1 o 2");


        } // Del Switch

    } // Del main()
} // De la classe
```

Console    
 <terminated> ExempleSwitch  
 Introdueix un número  
 0  
 i és zero.

Console    
 <terminated> ExempleSwitch  
 Introdueix un número  
 1  
 i és u.

Console    
 <terminated> ExempleSwitch  
 Introdueix un número  
 2  
 i és dos.

Console    
 <terminated> ExempleSwitch [Java Applica  
 Introdueix un número  
 3  
 el número és diferent a 0,1 o 2

**EXEMPLE SWITCH:** Programa mostra un menú d'opcions per calcular l'àrea d'un quadrat, o un rectangle o un triangle segons tria l'usuari. En cada cas el programa demanarà les dades necessaries.

```
import java.util.Scanner;

public class MenuOpcionsAmbSwitch {
    public static void main(String[] args) {

        Scanner entrada = new Scanner(System.in);
        int opcio;
        double costat;
        double base;
        double altura;
        double area;

        System.out.println(" CÀLCUL DE ÀREES");
        System.out.println(" -----");
        System.out.println(" 1. Quadrat");
        System.out.println(" 2. Rectàngle");
        System.out.println(" 3. Triàngle");
        System.out.print("\n Tria una opció (1-3): ");

        opcio= entrada.nextInt();

        switch (opcio) {
            case 1:
                System.out.print("\nHas triat calcular l'àrea d'un QUADRAT: ");
                System.out.print("\nIntrodueix el costat del quadrat en cm: ");
                costat = entrada.nextDouble();
                area = costat*costat;
                System.out.println("\nL'àrea del quadrat és " + area + " cm2");
                break;

            case 2:
                System.out.print("\nHas triat calcular l'àrea d'un RECTANGLE: ");
                System.out.print("\nIntrodueix la base del rectangle en cm: ");
                base = entrada.nextDouble();
                System.out.print("Introdueix l'altura del rectangle en cm: ");
                altura = entrada.nextDouble();
                area = base*altura;
                System.out.println("El área del rectangle es " + area + " cm2");
                break;

            case 3:
                System.out.print("\nHas triat calcular l'àrea d'un TRIANGLE: ");
                System.out.print("\nIntrodueix la base del triangle en cm: ");
                base = entrada.nextDouble();
                System.out.print("Introdueix l'altura del triangle en cm: ");
                altura = entrada.nextDouble();
                area = (base*altura)/2;
                System.out.println("El área del triangle es " + area + " cm2");
                break;

            default:
                System.out.print("\nOPCIÓ INCORRECTA !!!!.");

        } //Del switch
    } //Del main()
} //De la classe
```

```
<terminated> MenuOpcionsAmbSwitch [Java Application]
CÀLCUL DE ÀREES
-----
1. Quadrat
2. Rectàngle
3. Triàngle

Tria una opció (1-3): 1

Has triat calcular l'àrea d'un QUADRAT:
Introdueix el costat del quadrat en cm: 3

L'àrea del quadrat és 9.0 cm2
```

```
<terminated> MenuOpcionsAmbSwitch [Java Appli]
CÀLCUL DE ÀREES
-----
1. Quadrat
2. Rectàngle
3. Triàngle

Tria una opció (1-3): 4

OPCIÓ INCORRECTA !!!!.
```

```
<terminated> MenuOpcionsAmbSwitch [Java Appli]
CÀLCUL DE ÀREES
-----
1. Quadrat
2. Rectàngle
3. Triàngle

Tria una opció (1-3): 3

Has triat calcular l'àrea d'un TRIANGLE:
Introdueix la base del triangle en cm: 3

Introdueix l'altura del triangle en cm: 5

L'àrea del triangle es 7.5 cm2
```

```
<terminated> MenuOpcionsAmbSwitch [Java Application] C
CÀLCUL DE ÀREES
-----
1. Quadrat
2. Rectàngle
3. Triàngle

Tria una opció (1-3): 2

Has triat calcular l'àrea d'un RECTANGLE:
Introdueix la base del rectangle en cm: 3
Introdueix l'altura del rectangle en cm: 5

L'àrea del rectangle es 15.0 cm2
```

## Estructures Repetitives (Bucles)

Els **bucles** són **estructures de repetició**, s'utilitzen per repetir un conjunt de sentències o instruccions un número determinat de voltes mentre es complisca una condició (o fins que es complisca una condició) .

Per exemple, imagina que és necessari introduir la notes de 40 alumnes amb la finalitat de calcular la mitjana, la nota màxima i la nota mínima. Podríem declarar 40 variables i escriure 40 voltes les intruccions per demanar una nota per teclat i l'assignar-la a una variable : ( **System.out.println("Introdueix una nota: "); i nota1 = entrada.nextInt();** ), però no sembla una cosa molt eficient. **És molt més pràctic ficar dins d'un bucle aquelles sentències que volem que es repetisquen.**

Normalment **existeix una condició d'eixida**, que fa que el flux del programa abandone el bucle i continue just en la següent sentència. **Si no existeix** condició d'eixida o si aquesta condició **no es compleix mai**, es produiria el que es diu un **bucle infinit i el programa no acabaria** mai.

Un bloc d'instruccions es trobarà tancat mitjançant claus {...} si existeix més d'una instrucció igual que succeeixen les estructures alternatives (if - else).

Les estructures de repetició les podem construir de 3 maneres:

- Bucle **for**
- Bucle **while**
- Bucle **do - while**

Tot problema que requereix repetició pot fer-se amb qualsevol dels tres tipus de bucles, però segons el cas sol ser més senzill o intuïtiu utilitza l'un o l'altre.

En general, és recomanable utilitzar

- Bucle **for** quan es coneix per endavant **el número exacte de vegades** que ha de repetir-se el bloc d'instruccions.
- Bucle **while** quan **no sabem el nombre de vegades** que ha de repetir-se el bloc i és possible que **no haja d'executar-se** mai.
- Bucle **do - while** quan **no sabem el nombre de vegades** que ha de repetir-se el bloc i ha d'executar-se OBLIGATÒRIAMENT, **almenys una vegada**.



# 1.- Bucle for

Se sol utilitzar quan es coneix prèviament el nombre exacte d'iteracions (repeticions) que es realitzaran.

La **sintaxi** és la següent:

```
for (inicialització ; condició ; increment)
{
    // bloc d'accions (sentències);
}
```

- La clàusula inicialització és una instrucció que **s'executa una sola vegada** a l'inici del bucle, normalment per a inicialitzar un comptador. Per exemple:

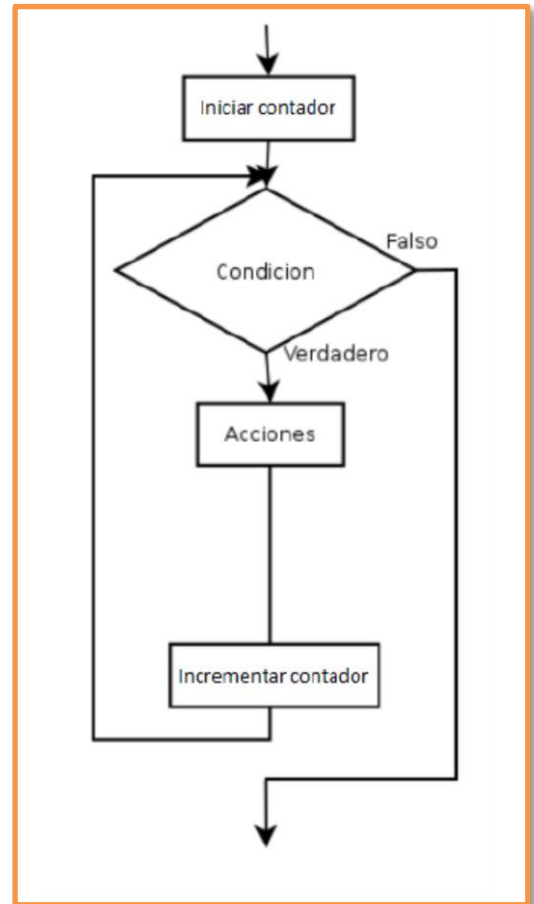
```
int i = 1;
```

- La clàusula condició és una **expressió lògica que s'avalua a l'inici de cada iteració del bucle**. En el moment en què aquesta expressió s'avalua a **false** es deixarà d'executar el bucle i el control del programa passarà a la següent instrucció (a continuació del bucle \*for).  
**S'utilitza per a indicar la condició en la qual vols que el bucle continue.** Per exemple:

```
i <= 10;
```

- La clàusula increment és una instrucció que **s'executa al final de cada iteració** del bucle (després del bloc d'instruccions). Generalment s'utilitza per a incrementar o decrementar el comptador. Per exemple:

```
i++ (incrementar i en 1).
```



**EXEMPLE: Bucle (for) que mostra per pantalla els números naturals de l'1 al 10:**

```
public static void main(String[] args) {
    for(int i = 1; i <= 10 ; i++) {
        System.out.print(i);
    }
}
```

- En la inicialització utilitzem **int i = 1** per a crear la variable **i** amb un valor inicial de 1.
- La condició **i <= 10** indica que el bucle ha de repetir-se mentre **i** siga menor o igual a 10.
- L'actualització **i++** indica que, al final de cada iteració, **i** ha d'incrementar-se en 1.

**EXEMPLE: Programa que mostra els números naturals (1,2,3,4,5,6,...) fins a un número introduït per teclat.**

```
public static void main(String[] args) {
    Scanner entrada = new Scanner (System.in);
    int max;
    System.out.print("Introdueix el número màxim: ");
    max = entrada.nextInt();

    for(int i = 1; i <= max; i++)
        System.out.print("\nNúmero: " + i);
}
```

```

Console
<terminated> calculos [Java Application]
Introdueix el número màxim: 7

Número: 1
Número: 2
Número: 3
Número: 4
Número: 5
Número: 6
Número: 7
  
```

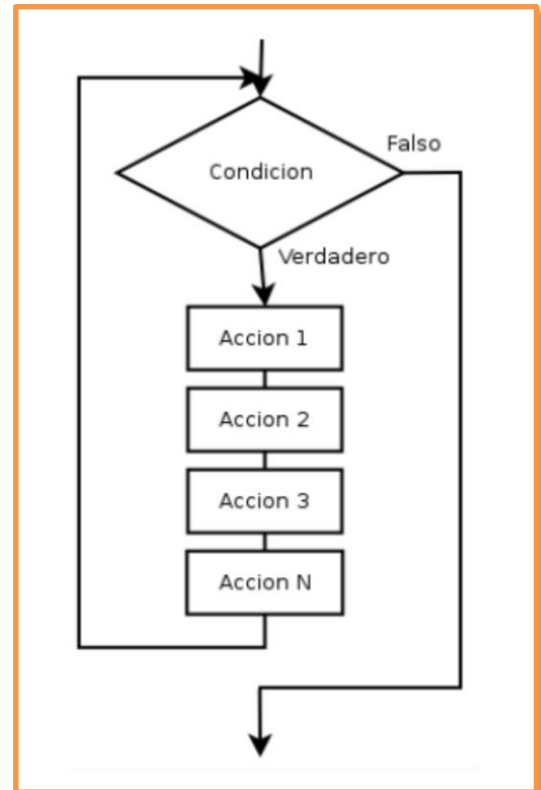
## 2.- Bucle while

El bucle **while** s'utilitza per a repetir un conjunt de sentències **sempre que es compleixi una determinada condició**. És important ressenyar que **la condició es comprova al començament del bucle**, per la qual cosa es **podria** donar el cas que aquest bucle **no s'executara mai**.

La sintaxi és la següent:

```
while (condició) {
    // bloc d'accions (Instruccions);
}
```

- El bloc d'instruccions s'executa mentre es compleix una condició (mentre **condició** s'avalua a **true**).
- **La condició es comprova ABANS de començar** a executar per primera vegada el bucle, per la qual cosa si s'avalua a **false** en la **primera iteració**, llavors el bloc d'accions no s'executarà cap vegada.



**EXEMPLE:** Bucle (while) que mostra per pantalla els nùemros naturals de l'1 al 10:

```
public static void main(String[] args) {
    int i = 1;
    while (i < 11) {
        System.out.println(i);
        i++;
    }
}
```

- Inicialitzem la variable que ens servirà de **guarda** del bucle **int i=1** a un valor inicial de 1.
- La condició **i < 11** indica que el bucle ha de repetir-se mentre i siga menor **ESTRICTE** que 11.
- Al final de cada iteració, i ha d'incrementar-se en 1. (**i++**)

**IMPORTANT:** Estos dos programas, una amb un bucle **while** i l'altre amb el bucle **for** SÓN exactament iguals i progueixen la mateixa eixida per pantalla. **Notar que les condicions (i < 11) i (i <= 10) també són iguals.**

```
public static void main(String[] args) {
    for(int i = 1; i <= 10 ; i++) {
        System.out.print(i);
    }
}
```

**EXEMPLE:** En el següent exemple es **compten** i es sumen els **números** que es van introduint pel teclat.

- Per a indicar-li al programa que ha de deixar de demanar números, l'usuari ha d'introduir un número negatiu; esta serà la del bucle.
- Observa que el bucle es repeteix mentre el número introduït siga major o igual que zero.

```
import java.util.Scanner;

public class ExempleWhile {

    public static void main(String[] args) {

        Scanner teclat = new Scanner(System.in);

        int numeroIntroduït = 0; // Per a que entre en el bucle
        int comptaNumeros = 0;
        int suma = 0;

        System.out.println("Per favor, introduceix números i ves polzant INTRO.");
        System.out.println("Per finalitzar, introduceix un número negatiu.");

        while (numeroIntroduït >= 0) {
            numeroIntroduït = teclat.nextInt();
            comptaNumeros++; // Incrementa en un la variable
            suma += numeroIntroduït; // Equival a suma = suma + numeroIntroduït
        }

        System.out.println("Has introduït " + (comptaNumeros - 1) + " números positivos.");
        System.out.println("La suma total de ellos es " + (suma - numeroIntroduït));

    } //Del main()
} // de la classe
```

```
<terminated> ExempleWhile [Java Application] C:\Program Files\Java
Per favor, introduceix números i ves polzant INTRO.
Per finalitzar, introduceix un número negatiu.
6
2
1
5
9
-1
Has introduït 5 números positivos.
La suma total de ellos es 23
```

**Analitzem les instruccions:**

```
System.out.println("Has introduït " + (comptaNumeros - 1) + " números positivos.");
```

- Hem de restar 1 a `comptaNumeros` perquè en este cas quan llegim de teclat el número negatiu per a eixir del bucle **while**, ja estem dins d'ell, i s'incrementa el total de números, per **tant es suma un de més que hi haurà que restar**.

```
System.out.println("La suma total de ellos es " + (suma - numeroIntroduït));
```

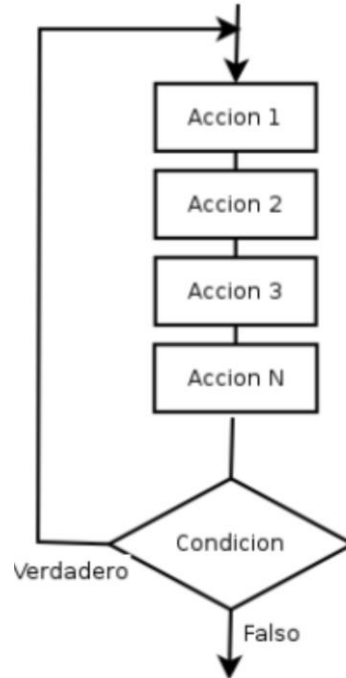
- Pel mateix motiu, l'últim valor introduït (el negatiu per a eixir del bucle, també es sumarà a la última iteració, i per tant **tindrem que eliminar eixa quantitat**.

### 3.- Bucle do-while

El bucle **do-while** funciona de la mateixa manera que el bucle **while**, amb l'excepció que la **condició** s'avalua al final de la iteració.

```
do {
    // bloc d'accions (Instruccions);
} while (condició) {
```

- En este tipus de bucle, **el bloc d'instruccions s'executa sempre almenys una vegada**, i aqueix bloc d'instruccions s'executarà mentre **condició** s'avalua a **true**.
- Per això en el bloc d'instruccions haurà d'existir alguna que, en algun moment, faça que **condició** s'avalua a **false**. Si no el bucle no acabaria mai!



**EXEMPLE .-** Bucle (do-while) que mostra per pantalla els núemros naturals de l'1 al 10: (Equivalent als dos anteriors)

```
public static void main(String[] args) {
    int i = 1;
    do {
        System.out.println(i);
        i++;
    } while (i < 11);
}
```

Console  
<terminated>

1  
2  
3  
4  
5  
6  
7  
8  
9  
10

- Inicialitzem la variable que ens servirà de **guarda** del bucle **int i=1** a un valor inicial de 1.
- En cada iteració, **i** ha d'incrementar-se en 1. (**i++**)
- Al final de cada iteració, s'avalua la **condició** **i < 11** si esta és igual a **true** tornem a començar una nova iteració del bucle des de la clàusula **do**

EXEMPLE: En aquest cas s'aniran llegint números de teclat mentre el número introduït siga parell; el programa parará, per tant, quan s'introduïska un nombre imparell.

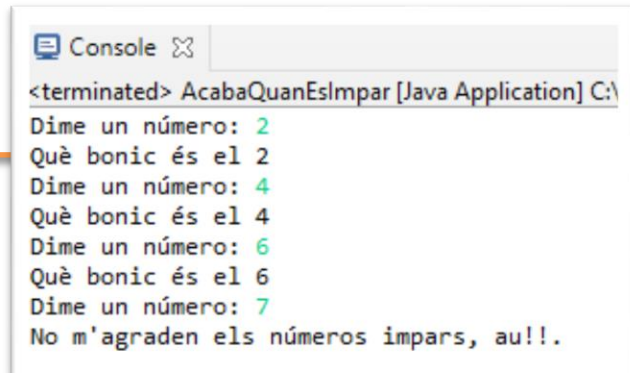
```
import java.util.Scanner;

public class AcabaQuanEsImpar {

    public static void main(String[] args) {
        Scanner entrada = new Scanner(System.in);
        int numero;

        do {
            System.out.print("Dime un número: ");
            numero = entrada.nextInt();

            if (numero % 2 == 0) { // comprova si el número introduït és parell
                System.out.println("Què bonic és el " + numero);
            } else {
                System.out.println("No m'agraden els números impars, au!!.");
            }
        } while (numero % 2 == 0);
    } // del main()
} // de la classe
```



```
<terminated> AcabaQuanEsImpar [Java Application] C:\
Dime un número: 2
Què bonic és el 2
Dime un número: 4
Què bonic és el 4
Dime un número: 6
Què bonic és el 6
Dime un número: 7
No m'agraden els números impars, au!!.
```