

# UD2.- Estructures Repetitives

(Bucles)

## Estructures Repetitives (Bucles)

Els **bucles** són **estructures de repetició**, s'utilitzen per repetir un conjunt de sentències o instruccions un número determinat de voltes mentres es compleix una condició (o fins que es compleixi 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 instruccions 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 requereixi repetició pot fer-se amb qualsevol dels tres tipus de bucles, però segons el cas sol ser més senzillo 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;
```

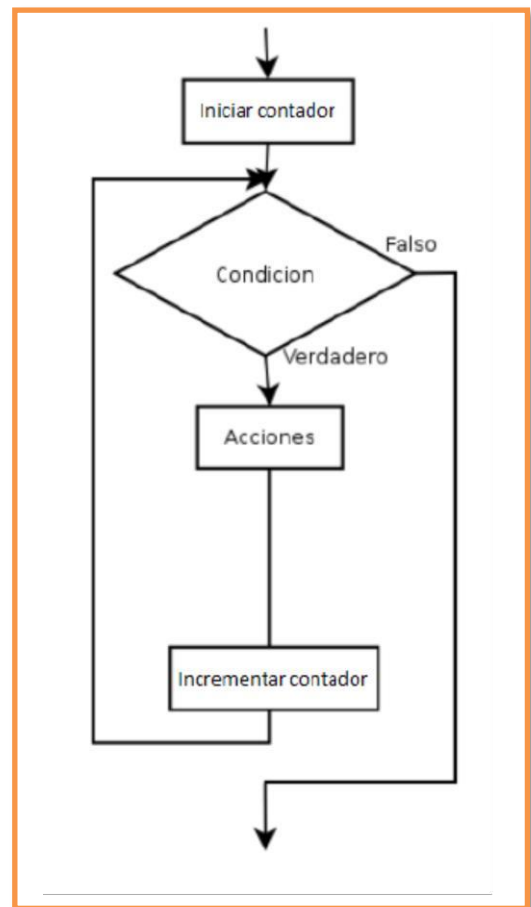
- La clàusula condició és una **expressió lògica que s'avalua a l'inici de cada iteració del bucle**. En el moment en què aquesta expressió s'avalua a **false** es deixarà d'executar el bucle i el control del programa passarà a la següent instrucció (a continuació del bucle \*for).

**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).
```



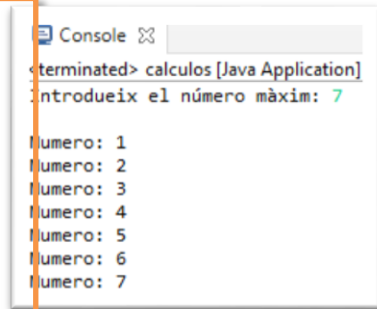
**EXEMPLE: Bucle (for) que mostra per pantalla els números naturals de l'1 al 10:**

```
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
- L'actualització **i++** indica que, al final de cada iteració, i ha d'incrementar-se

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("\nNumero: " + i);  
}
```



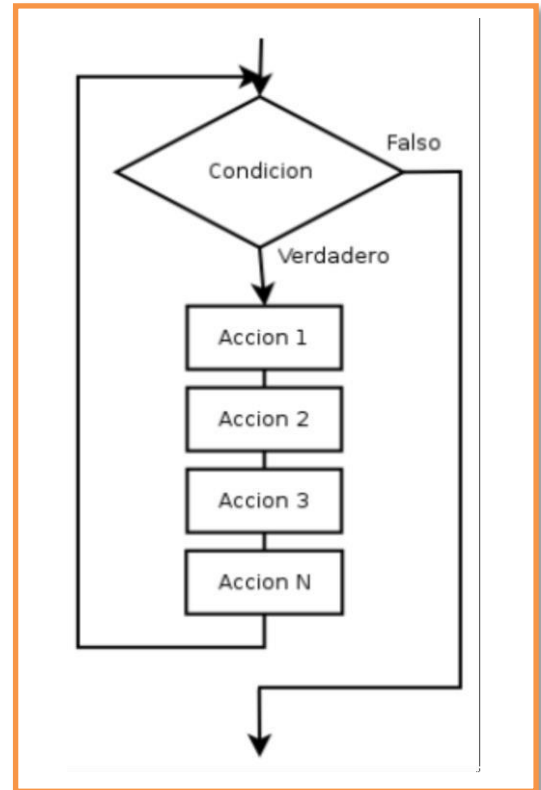
## 2.- Bucle while

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);
}
```

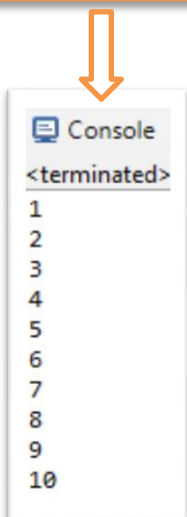
- El bloc d'instruccions s'executa mentre es compleix una condició (mentre **condició** s'avalua a **true**).
- La **condició es comprova ABANS de començar** a executar per primera vegada el bucle, per la qual cosa si s'avalua a **false** en la **primera iteració**, llavors el bloc d'accions no s'executarà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



**EXEMPLE:** Bucle (while) que mostra per pantalla els números 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 produeixen la mateixa eixida per pantalla. **Notar que les condicions (i < 11) i (i <= 10) també són**

```
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));
```

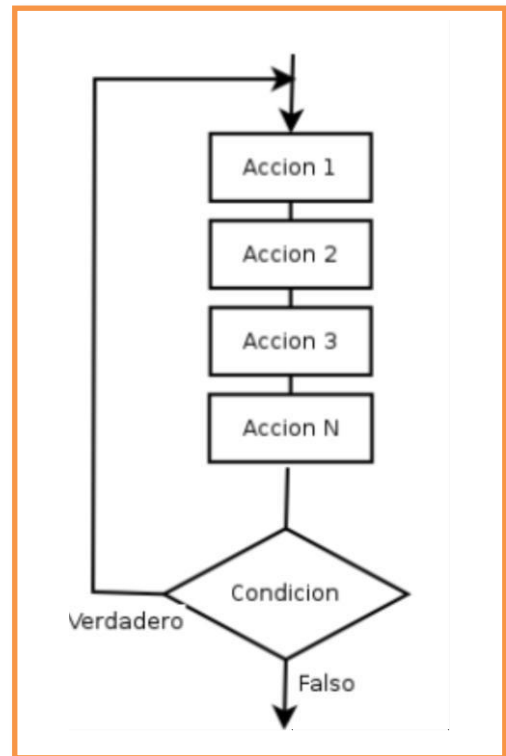
- Pel mateix motiu, l'últim valor introduït (el negatiu per a eixir del bucle, també es sumarà a la última iteració, i per tant **findrem que eliminar eixa quantitat**.

### 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, faci que **condició** s'avalua a **false**. Si no el bucle no acabaria mai!



**EXEMPLE .- Bucle (do-while) que mostra per pantalla els números 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'introduísca un nombre imparell.

```
import java.util.Scanner;

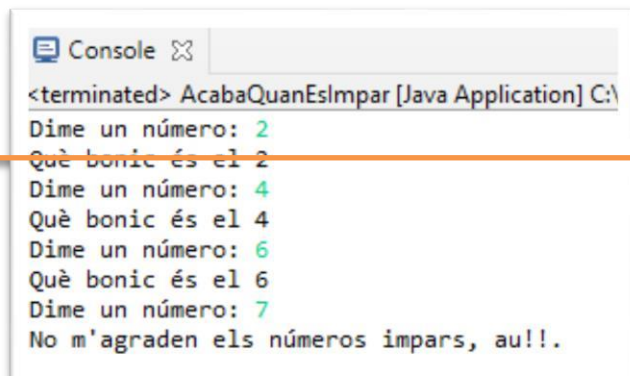
public class AcabaQuanEsImpar {

    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!!.
```