

## Unitat 5.

### Bucles en Java

Autors: Carlos Cacho y Raquel Torres

Revisat per: Lionel Tarazon - [lionel.tarazon@ceedcv.es](mailto:lionel.tarazon@ceedcv.es)

Fco. Javier Valero – [franciscojavier.valero@ceedcv.es](mailto:franciscojavier.valero@ceedcv.es)

José Manuel Martí - [josemanuel.marti@ceedcv.es](mailto:josemanuel.marti@ceedcv.es)

Jose Cantó alonso – [j.cantoalonso@edu.gva.es](mailto:j.cantoalonso@edu.gva.es)

2023/2024

## Llicència



[CC BY-NC-SA 3.0 ES](https://creativecommons.org/licenses/by-nc-sa/3.0/es/) **Reconeixement – No Comercial – Compartir Igual (by- nc-sa)** No es permet un ús comercial de l'obra original ni de les possibles obres derivades, la distribució de les quals s'ha de fer amb una llicència igual a la que regula l'obra original. NOTA: Aquesta és una obra derivada de l'obra original realitzada per Carlos Cacho i Raquel Torres.

## Nomenclatura

Al llarg d'aquest tema s'utilitzaran diferents símbols per a distingir elements importants dins del contingut. Aquests símbols són:



Important



Atenció



Interessant

## Contingut

<b>1. Introducció .....</b>	<b>4</b>
<b>2. Bucle for.....</b>	<b>5</b>
<b>3. Bucle while .....</b>	<b>7</b>
<b>4. Bucle do-while .....</b>	<b>9</b>
<b>5. Exemples .....</b>	<b>11</b>
5.1. Exemple 1.....	11
5.1.1. <i>Ordinograma</i> .....	11
5.1.2. <i>Codi</i> .....	11
5.1.3. <i>Eixida</i> .....	11
5.2. Exemple 2.....	12
5.2.1. <i>Ordinograma</i> .....	12
5.2.2. <i>Codi</i> .....	13
5.2.3. <i>Eixida</i> .....	13
<b>6. Agraïments .....</b>	<b>14</b>

## 1. Introducció

Els bucles són estructures de repetició, blocs d'instruccions que es repeteixen un nombre de vegades mentre es complisca una condició o fins que es complisca una condició.

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

Existeixen tres construccions per a aquestes estructures de repetició:

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

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

Com a regla general és recomanable:



Utilitzar el bucle **for** quan es conega per endavant el nombre exacte de vegades que ha de repetir-se el bloc d'instruccions.



Utilitzar el 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 cap vegada.



Utilitzar el bucle **do-while** quan no sabem el nombre de vegades que ha de repetir-se el bloc i deurà executar-se almenys una vegada.

Aquestes regles són generals i alguns programadors se senten més còmodes utilitzant principalment una d'elles. Amb major o menor esforç, pot utilitzar-se qualsevol de les tres indistintament.

## 2. Bucle for

El bucle *for* es codifica de la següent forma:

Codi	Ordinograma
<pre>for (inicialització ; condició ; increment) {     bloc d'accions; }</pre>	<pre> graph TD     Start(( )) --&gt; Init[Iniciar contador]     Init --&gt; Cond{Condicion}     Cond -- Falso --&gt; Exit(( ))     Cond -- Verdadero --&gt; Acc[Acciones]     Acc --&gt; Inc[Incrementar contador]     Inc --&gt; Cond   </pre>

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 1:** Bucle que mostra per pantalla els nombres naturals de l'1 al 10:

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

- En la inicialització utilitzem **int i=1** per a crear la variable i amb un valor inicial de 1.

- La condició  **$i \leq 10$**  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 2:** Programa que mostra els nombres naturals (1,2,3,4,5,6,...) fins a un número introduït per teclat:

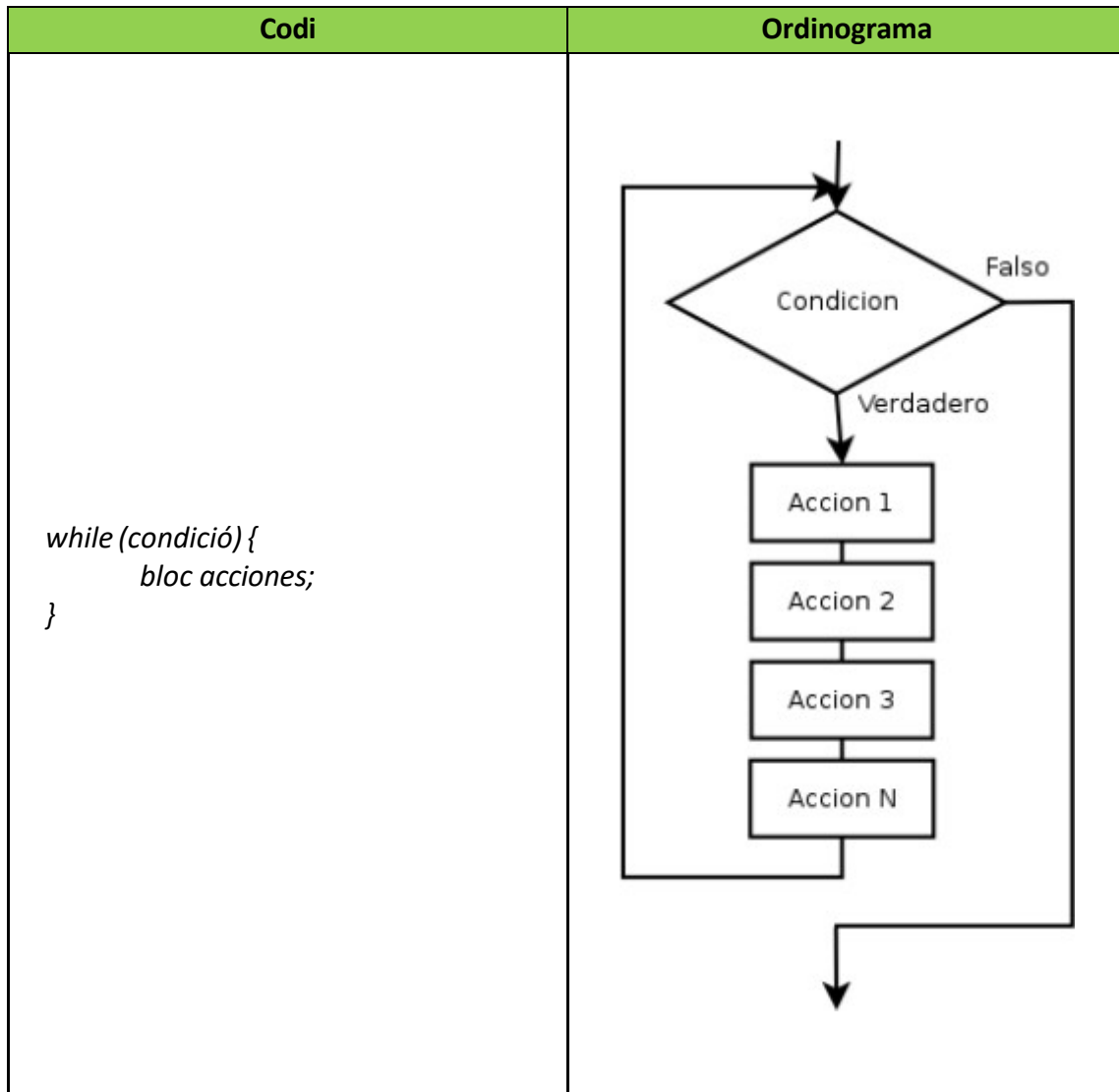
```
6 public static void main(String[] args) {  
7     Scanner sc = new Scanner(System.in);  
8     int max;  
9     System.out.print("Introduce el número máximo: ");  
10    max = sc.nextInt();  
11    for (int i = 1; i <= max; i++) {  
12        System.out.println("Número: " + i);  
13    }  
14 }  
15 }  
16 }
```

Sent l'eixida:

```
run:  
Introduce el número máximo: 5  
Número: 1  
Número: 2  
Número: 3  
Número: 4  
Número: 5  
BUILD SUCCESSFUL (total time: 6 seconds)
```

### 3. Bucle while

El bucle *while* es codifica de la següent forma:



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.

El mateix **exemple 2** d'abans, fet amb un bucle **while** seria:

```
7 public static void main(String[] args) {  
8     Scanner sc = new Scanner(System.in);  
9     int max, cont;  
10    System.out.print("Introduce el número máximo: ");  
11    max = sc.nextInt();  
12    cont = 1;  
13    while (cont <= max) {  
14        System.out.println("Número: " + cont);  
15        cont++;  
16    }  
17 }
```

l'execució:

```
run:  
Introduce el número máximo: 5  
Número: 1  
Número: 2  
Número: 3  
Número: 4  
Número: 5  
BUILD SUCCESSFUL (total time: 6 seconds)
```



## 4. Bucle do-while

El bucle *do-while* es codifica de la següent forma:

Codi	Ordinograma
<pre>do {     bloc d'accions; } while (condició);</pre>	<pre> graph TD     Entry(( )) --&gt; A1[Accion 1]     A1 --&gt; A2[Accion 2]     A2 --&gt; A3[Accion 3]     A3 --&gt; AN[Accion N]     AN --&gt; C{Condicion}     C -- Verdadero --&gt; Entry     C -- Falso --&gt; Exit(( ))   </pre>

En aquesta mena 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 iteració que, en algun moment, faci que '*condición*' s'avalua a '*false*'. Si no el bucle no acabaria mai!

El mateix **exemple 2** anterior, fet amb un bucle **do-while** seria:

```
public static void main(String[] args) {  
    Scanner sc = new Scanner(System.in);  
    int max, cont;  
    System.out.print("Introduce el número máximo: ");  
    max = sc.nextInt();  
    cont = 1;  
  
    do {  
        System.out.println("Número: " + cont);  
        cont++;  
    } while (cont <= max);  
}
```

I l'eixida:

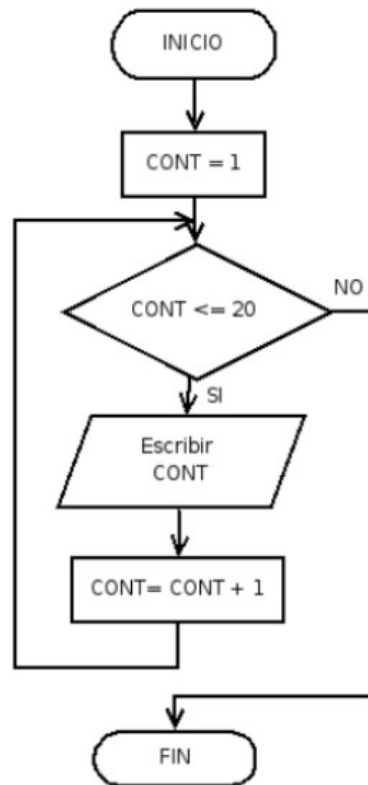
```
run:  
Introduce el número máximo: 5  
Número: 1  
Número: 2  
Número: 3  
Número: 4  
Número: 5  
BUILD SUCCESSFUL (total time: 6 seconds)
```

## 5. Exemples

### 5.1. Exemple 1

Programa que mostre per pantalla els 20 primers nombres naturals (1, 2, 3... 20).

#### 5.1.1. Ordinograma



#### 5.1.2. Codi

```

12 public class Ejercicio1 {
13
14     public static void main(String[] args) {
15         int cont;
16
17         for(cont=1;cont<=20;cont++)
18             System.out.print(cont + " ");
19
20         System.out.print("\n");
21     }
22 }
  
```

#### 5.1.3. Eixida

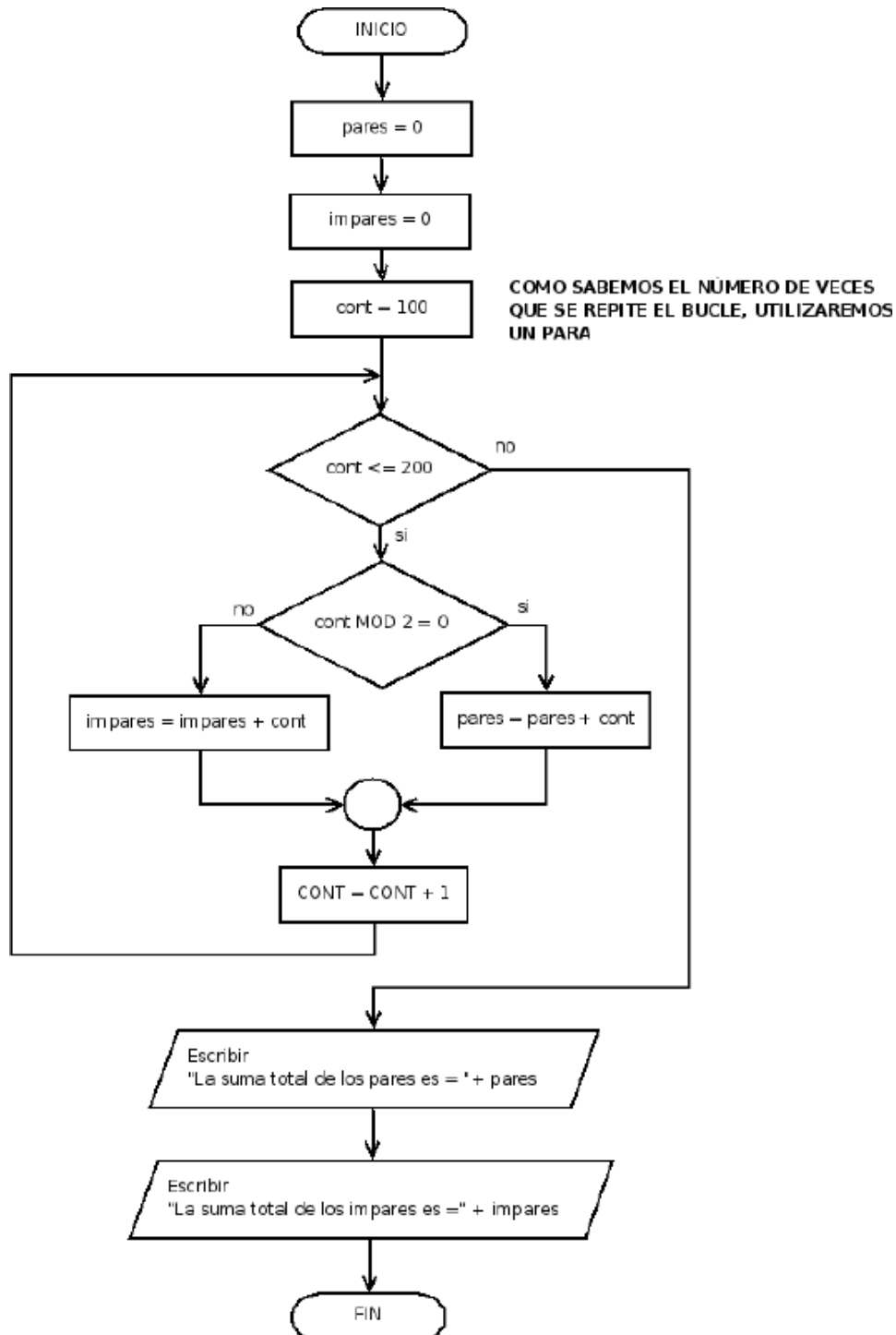
```

run:
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
BUILD SUCCESSFUL (total time: 0 seconds)
  
```

## 5.2. Exemple 2

Programa que suma independentment els parells i els imparells dels números compresos entre 100 i 200.





### 5.2.1. Ordinograma



## 5.2.2. Codi

```
12 public class Ejercicio11 {  
13  
14     public static void main(String[] args) {  
15         int pares, impares, cont;  
16  
17         pares = 0;  
18         impares = 0;  
19  
20         for(cont=100; cont <= 200; cont++)  
21         {  
22             if(cont % 2 == 0)  
23                 pares = pares + cont;  
24             else  
25                 impares = impares + cont;  
26         }  
27  
28         System.out.println("La suma total de los pares es " + pares);  
29         System.out.println("La suma total de los impares es " + impares);  
30     }  
31 }  
32 }
```

## 5.2.3. Eixida

```
run:  
La suma total de los pares es 7650  
La suma total de los impares es 7500  
BUILD SUCCESSFUL (total time: 0 seconds)
```

## 6. Agraïments

Anotacions actualitzades i adaptats al CEEDCV a partir de la següent documentació:

- [1] Anotacions Programació de José Antonio Díaz-Alejo. IES Camp de Morvedre.