

TEMA5. BUCLES EN JAVA

Programación
CFGs DAW

Profesores:

Carlos Cacho

Raquel Torres

carlos.cacho@ceedcv.es

raquel.torres@ceedcv.es

Licencia



Reconocimiento - NoComercial - CompartirIgual (by-nc-sa): No se permite un uso comercial de la obra original ni de las posibles obras derivadas, la distribución de las cuales se debe hacer con una licencia igual a la que regula la obra original.

Nomenclatura

A lo largo de este tema se utilizarán distintos símbolos para distinguir elementos importantes dentro del contenido. Estos símbolos son:



Importante



Atención



Interesante

Revisiones

ÍNDICE DE CONTENIDO


- 1.Introducción.....4
- 2.Bucle for.....5
- 3.Bucle while.....7
- 4.Bucle do-while.....9
- 5.Ejemplos.....11
 - 5.1 Ejemplo 1.....11
 - 5.2 Ejemplo 2.....12
- 6.Agradecimientos.....14

UD05. BUCLES EN JAVA

1. INTRODUCCIÓN

Las estructuras de control son construcciones hechas a partir de palabras reservadas del lenguaje que permiten modificar el flujo de ejecución de un programa. De este modo, pueden crearse construcciones alternativas y bucles de repetición de bloques de instrucciones.


Hay que señalar que un bloque de instrucciones se encontrará encerrado mediante llaves {.....} si existe más de una instrucción.


 Los **bucles** son estructuras de repetición. Bloques de instrucciones que se repiten un número de veces, mientras se cumpla una condición o hasta que se cumpla una condición.


Existen tres construcciones para estas estructuras de repetición:

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

Como regla general puede decirse que:

 Se utilizará el bucle **for** cuando se conozca de antemano el número exacto de veces que ha de repetirse un determinado bloque de instrucciones.

 Se utilizará el bucle **do-while** cuando no se conoce exactamente el número de veces que se ejecutará el bucle, pero se sabe que por lo menos se ha de ejecutar una.

 Se utilizará el bucle **while** cuando es posible que no deba ejecutarse ninguna vez.

Estas reglas son generales y algunos programadores se sienten más cómodos

utilizando principalmente una de ellas. Con mayor o menor esfuerzo, puede utilizarse cualquiera de ellas indistintamente.

2. BUCLE FOR

El bucle *for* se codifica de la siguiente forma:

Código	Ordinograma
<pre>for (inicialización ; condición ; incremento) { bloque acciones; }</pre>	<pre>graph TD Start(()) --> Init[Iniciar contador] Init --> Cond{Condicion} Cond -- Falso --> Exit(()) Cond -- Verdadero --> Acc[Acciones] Acc --> Inc[Incrementar contador] Inc --> Cond</pre>

La cláusula *inicialización* es una instrucción que se ejecuta una sola vez al inicio del bucle, normalmente para inicializar un contador.

La cláusula *condición* es una expresión lógica, que se evalúa al inicio de cada

nueva iteración del bucle. En el momento en que dicha expresión se evalúe a false, se dejará de ejecutar el bucle y el control del programa pasará a la siguiente instrucción (a continuación del bucle *for*).

La cláusula *incremento* es una instrucción que se ejecuta en cada iteración del bucle como si fuera la última instrucción dentro del bloque de instrucciones. Generalmente se trata de una instrucción de incremento o decremento de alguna variable.

Por ejemplo: Programa que muestra los números naturales (1,2,3,4,5,6,...) hasta un número introducido por teclado.

```
20 public static void main(String[] args) throws IOException {
21     BufferedReader stdin=new BufferedReader(new InputStreamReader(System.in));
22     int num, cont;
23
24     System.out.print("Introduce el número máximo: ");
25
26     num =Integer.parseInt(stdin.readLine());
27
28     for (cont=1; cont<=num; cont++)
29     {
30         System.out.println("Número: " + cont);
31     }
32 }
```

Siendo la salida:

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

3. BUCLE WHILE

El bucle *while* se codifica de la siguiente forma:

Código	Ordinograma
<pre>while (Condición) { bloque acciones; }</pre>	<pre>graph TD; Entry(()) --> Condicion{Condicion}; Condicion -- Falso --> Exit(()); Condicion -- Verdadero --> Accion1[Accion 1]; Accion1 --> Accion2[Accion 2]; Accion2 --> Accion3[Accion 3]; Accion3 --> AccionN[Accion N]; AccionN --> Condicion;</pre>

El bloque de instrucciones se ejecuta mientras se cumple una condición (mientras Condición se evalúe a true), **la condición se comprueba ANTES de empezar** a ejecutar por primera vez el bucle, por lo que si se evalúa a false en la primera iteración, entonces el bloque de acciones no se ejecutará ninguna vez.

El mismo ejemplo anterior sería:

```
22 public static void main(String[] args) throws IOException {
23     BufferedReader stdin=new BufferedReader(new InputStreamReader(System.in));
24     int num, cont;
25
26     System.out.print("Introduce el número máximo: ");
27
28     num =Integer.parseInt(stdin.readLine());
29
30     cont = 1;
31
32     while (cont <= num)
33     {
34         System.out.println("Número: " + cont);
35         cont++;
36     }
37 }
```

Y la salida:

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


4. BUCLE DO-WHILE

El bucle *while* se codifica de la siguiente forma:

Código	Ordinograma
<pre>do { bloque acciones; } while (Condición);</pre>	<pre>graph TD Entry(()) --> A1[Accion 1] A1 --> A2[Accion 2] A2 --> A3[Accion 3] A3 --> AN[Accion N] AN --> Cond{Condicion} Cond -- Verdadero --> Entry Cond -- Falso --> Exit(())</pre>

En este tipo de bucle, **el bloque de instrucciones se ejecuta siempre una vez por lo menos**, y ese bloque de instrucciones se ejecutará mientras *Condición* se evalúe a *true*.

⚡ Por lo tanto, entre las instrucciones que se repiten deberá existir alguna que, en algún momento, haga que *Condición* se evalúe a *false*, de lo contrario el bucle sería infinito!!!

Continuando con el mismo ejemplo:

```
22 public static void main(String[] args) throws IOException {
23     BufferedReader stdin=new BufferedReader(new InputStreamReader(System.in));
24     int num, cont;
25
26     System.out.print("Introduce el número máximo: ");
27
28     num =Integer.parseInt(stdin.readLine());
29
30     cont = 1;
31
32     do
33     {
34         System.out.println("Número: " + cont);
35         cont++;
36     }while (cont <= num);
37 }
```

Y la salida:

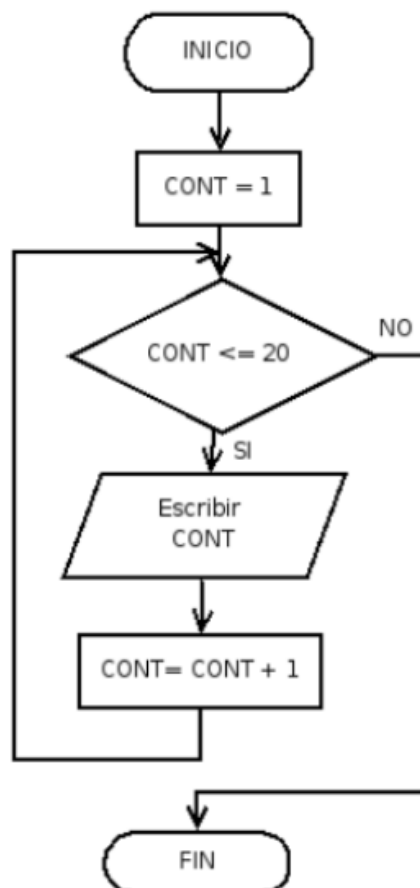
```
run:
Introduce el número máximo: 6
Número: 1
Número: 2
Número: 3
Número: 4
Número: 5
Número: 6
BUILD SUCCESSFUL (total time: 2 seconds)
```

5. EJEMPLOS

5.1 Ejemplo 1

Programa que muestre por pantalla los 20 primeros números naturales (1, 2, 3, ..., 20). (Ejercicio UD3_1)

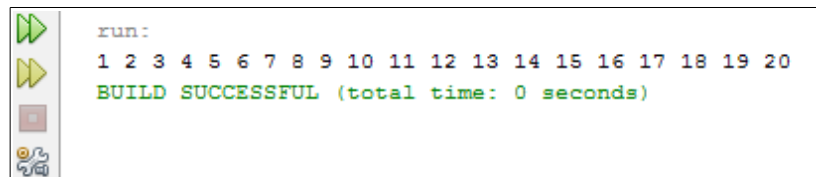
Ordinograma:



Código:

```
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     }
```

Salida:

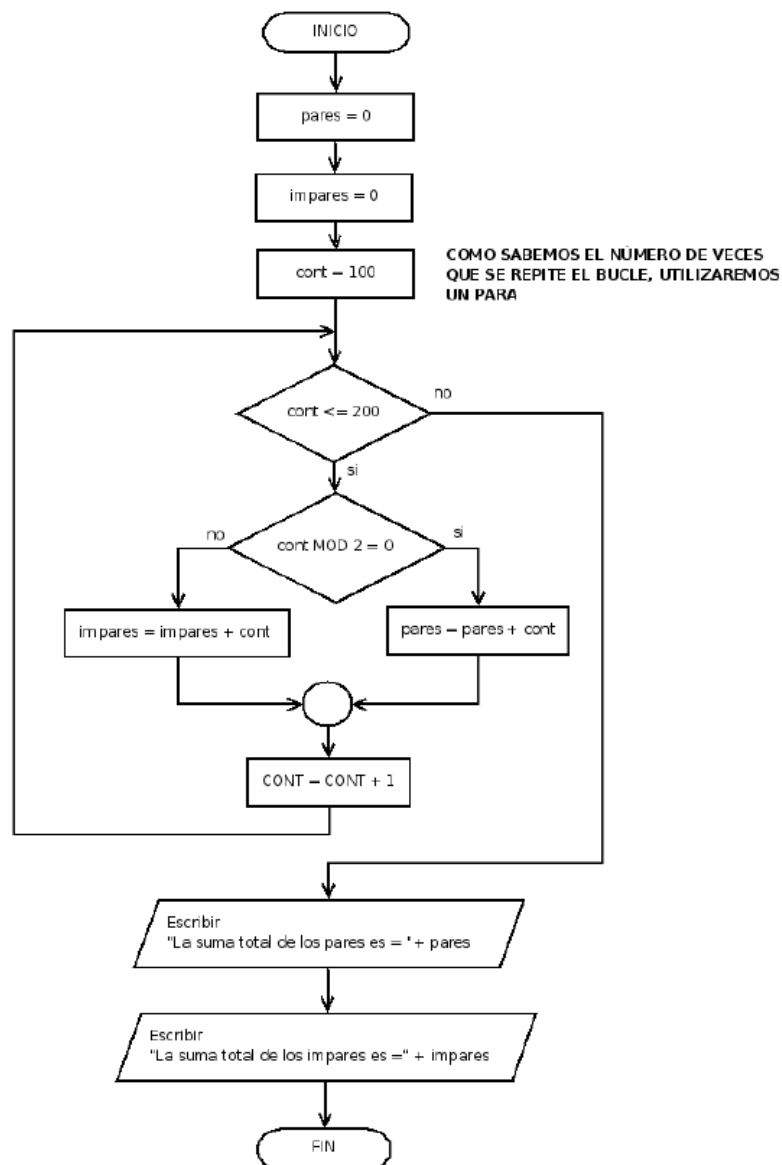


```
run:  
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20  
BUILD SUCCESSFUL (total time: 0 seconds)
```

5.2 Ejemplo 2

Programa que suma independientemente los pares y los impares de los números comprendidos entre 100 y 200. (Ejercicio UD3_11)

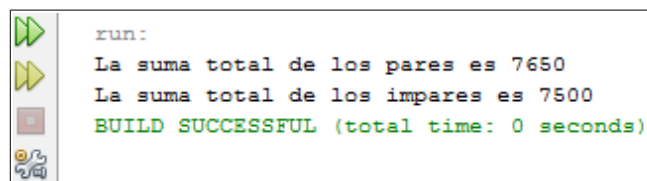
Ordinograma:



Código:

```
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 }
```

Salida:



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

6. AGRADECIMIENTOS

Apuntes actualizados y adaptados al CEEDCV a partir de la siguiente documentación:

[1] Apuntes Programación de José Antonio Díaz-Alejo. IES Camp de Morvedre.