MÓDULO 0485 - PROGRAMACIÓN

UT 3-

ESTRUCTURAS DE CONTROL (PARTE 2) – BUCLES

Técnico Superior En Desarrollo De Aplicaciones Web

1º DAW (Grupo 1WV) (Vespertino)

Curso 2025-26

Profesor: Javier Rojo

ÍNDICE

ESTRUCTURAS DE CONTROL: BUCLES

ESTRUCTURAS DE CONTROL: BUCLES

ÍNDICE

INTRODUCCIÓN

INTRODUCCIÓN

- 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.
- Un bloque de instrucciones se encontrará encerrado mediante llaves {.....} si existe más de una instrucción al igual que suceden las estructuras alternativas (if… else… etc).
- Existen tres construcciones para estas estructuras de repetición:

1. Bucle ***for***
2. Bucle ***while***
3. Bucle ***do-while***

INTRODUCCIÓN

- Todo problema que requiera repetición puede hacerse con cualquiera de los tres, pero según el caso suele ser más sencillo o intuitivo utiliza uno u otro.
- Como regla general es recomendable:

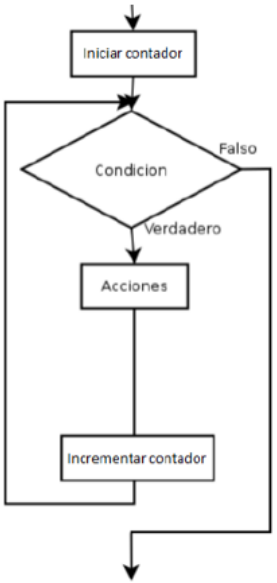
1. *Utilizar el bucle **for** cuando se conozca de antemano el número exacto de veces que ha de repetirse el bloque de instrucciones.*
2. *Utilizar el bucle **while** cuando no sabemos el número de veces que ha de repetirse el bloque y es posible que no deba ejecutarse ninguna vez.*
3. *Utilizar el bucle **do-while** cuando no sabemos el número de veces que ha de repetirse el bloque y deberá ejecutarse al menos una vez.*

OJO: 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 las tres indistintamente.

BUCLE FOR

BUCLE FOR

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

Código	Ordinograma
for (inicialización ; condición ; incremento) *{* *bloque acciones;* *}*	

BUCLE FOR

Código

```
for (inicialización ; condición ; incremento)
{
    bloque acciones;
}
```

- La cláusula ***inicialización*** es una instrucción que se ejecuta una sola vez al inicio del bucle, normalmente para inicializar un contador.
  - Por ejemplo **int i = 1;**
- La cláusula ***condición*** es una expresión lógica que se evalúa al inicio de cada 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*).
  - Se utiliza para indicar la condición en la que quieres que el bucle continúe.
  - Por ejemplo **i <= 10;**
- La cláusula ***incremento*** es una instrucción que se ejecuta al final de cada iteración del bucle (después del bloque de instrucciones).
  - Generalmente se utiliza para incrementar o decrementar el contador.
  - Por ejemplo **i++**; (incrementar i en 1).

BUCLE FOR

Código
for (inicialización ; condición ; incremento) *{* *bloque acciones;* *}*

Ejemplo 1: Bucle que muestra por pantalla los números naturales del 1 al 10:

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

- En la inicialización utilizamos **int i=1** para crear la variable i con un valor inicial de 1.
- La condición **i<=10** indica que el bucle debe repetirse mientras i sea menor o igual a 0.
- La actualización **i++** indica que, al final de cada iteración, i debe incrementarse en 1.

BUCLE FOR

Código
for (inicialización ; condición ; incremento) *{* *bloque acciones;* *}*

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

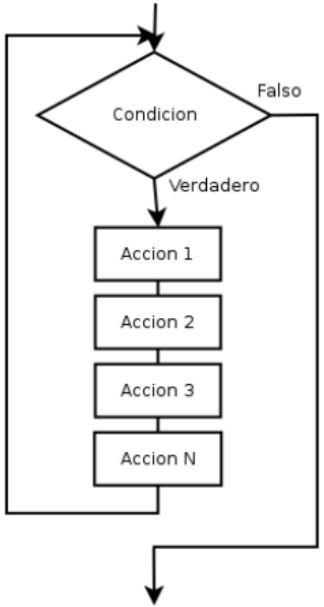
```
6       public static void main(String[] args) {
7           Scanner sc = new Scanner(System.in);
8           int max;
9           System.out.print("Introduce el número máximo: ");
10          max = sc.nextInt();
11          for (int i = 1; i <= max; i++) {
12              System.out.println("Número: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

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

BUCLE WHILE

BUCLE WHILE

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

Código	Ordinograma
while (condición) { *bloque acciones;* *}*	

BUCLE WHILE

Código
while (condición) { *bloque acciones;* *}*

- 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 tanto, si se evalúa a false en la primera iteración, entonces el bloque de acciones no se ejecutará ninguna vez.

BUCLE WHILE

Código
while (condición) { *bloque acciones;* *}*

El mismo **ejemplo 2** anterior hecho con un bucle **while** sería:

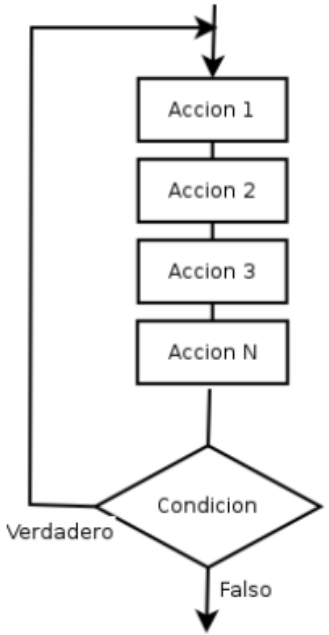
```
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int max, cont;
        System.out.print("Introduce el número máximo: ");
        max = sc.nextInt();
        cont = 1;
        while (cont <= max) {
            System.out.println("Número: " + cont);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

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

BUCLE DO-WHILE

BUCLE DO-WHILE

1. El bucle do-*while* se codifica de la siguiente forma:

Código	Ordinograma
do *{* *bloque acciones;* *}* *while (condición);*	

BUCLE DO-WHILE

Código

```
do
{
    bloque acciones;
}
while (condición);
```

- En este tipo de bucle, **el bloque de instrucciones se ejecuta siempre al menos una vez**
  - Ese bloque de instrucciones se ejecutará mientras **condición** se evalúe a *true*.
  - Por ello en el bloque de instrucciones deberá existir alguna que, en algún momento, haga que *condición* se evalúe a *false*. → ¡Si no el bucle no acabaría nunca!

BUCLE DO-WHILE

Código
do *{* *bloque acciones;* *}* *while (condición);*

El mismo **ejemplo 2** anterior hecho con un bucle do-**while** sería:

```
    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);

    }
}
```

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

VARIABLES ESPECÍFICAS

VARIABLES ESPECÍFICAS

- En los bucles se suelen utilizar tres tipos de variables específicas:

1. **Contadores**: variable numérica que permite contar las ocurrencias de un suceso
2. **Acumuladores**: variable numérica que permite acumular operaciones
3. **Banderas**: variable numérica que permite determinar si ha ocurrido o no un suceso

VARIABLES ESPECÍFICAS

- Ejemplo de uso de un contador:

```
public static void main(String[] args) {

    Scanner sc = new Scanner( source: System.in);
    int contador = 0, numLeido;

    for (int i= 0; i < 5; i++) {
        System.out.println( x: "Introduce un número:");
        numLeido = sc.nextInt();
        if (numLeido % 2 == 0) {
            contador ++;
        }
    }

    System.out.println("Se han introducido " + contador + " numeros pares");
}
```

VARIABLES ESPECÍFICAS

- Ejemplo de uso de un acumulador:

```
public static void main(String[] args) {

    Scanner sc = new Scanner( source: System.in);
    int numIntroducidos = 5, numLeido;
    float acum = 0, media;

    for (int i = 1; i <= numIntroducidos; i++) {
        System.out.println( x: "Introduce un número:");
        numLeido = sc.nextInt();
        acum = acum + numLeido;
    }

    media = acum / numIntroducidos;
    System.out.println("La media de los números es " + media);
}
```

VARIABLES ESPECÍFICAS

- Ejemplo de uso de una bandera:

```
public static void main(String[] args) {

    Scanner sc = new Scanner( source: System.in);
    int numIntroducidos = 5, numLeido;
    boolean flag = false;

    for (int i = 1; i <= numIntroducidos; i++) {
        System.out.println( x: "Introduce un número:");
        numLeido = sc.nextInt();
        if (numLeido % 2 == 0) {
            flag = true;
        }
    }

    if (flag == true) {
        System.out.println( x: "Se ha introducido, al menos, un número par");
    } else {
        System.out.println( x: "No se ha introducido ningún número par");
    }
}
```

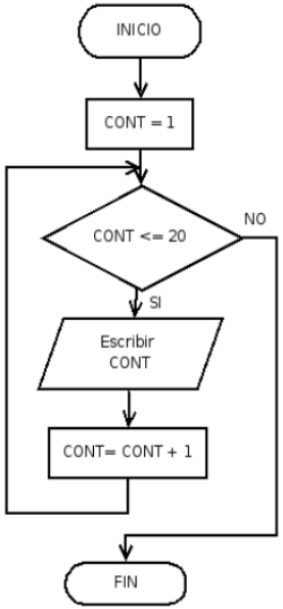
EJEMPLOS

EJEMPLO 1

- **Enunciado:** Programa que muestre por pantalla los 20 primeros números naturales (1, 2, 3… 20).

EJEMPLO 1

- **Enunciado:** Programa que muestre por pantalla los 20 primeros números naturales (1, 2, 3… 20).
- **Diagrama:**



EJEMPLO 1

- **Solución:**

```
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  }
```

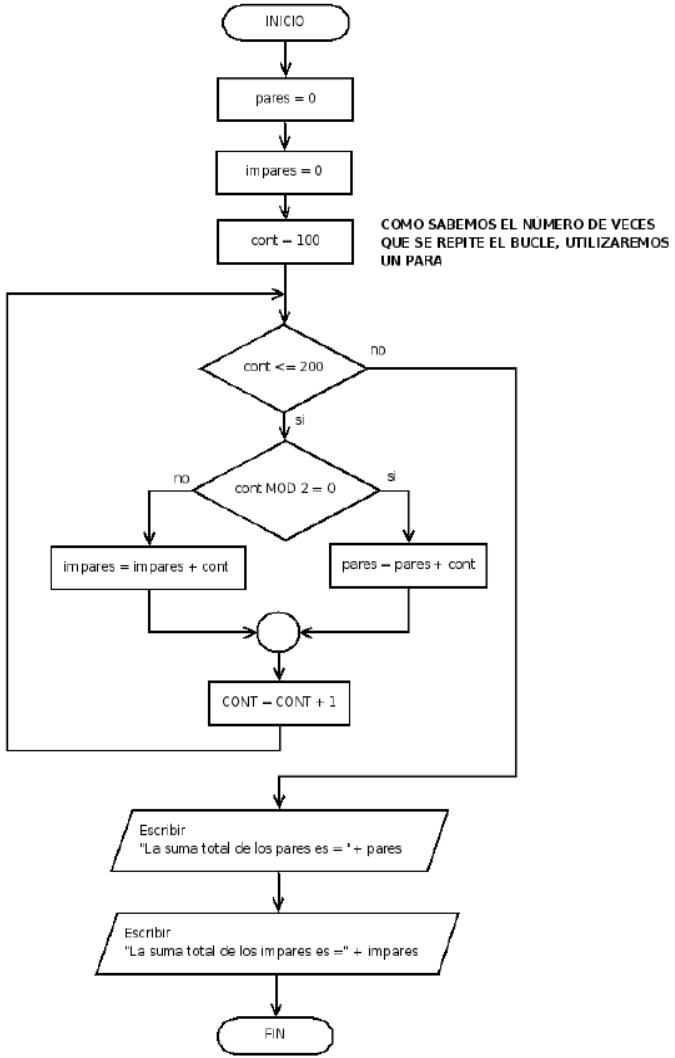
```
run:
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
BUILD SUCCESSFUL (total time: 0 seconds)
```

EJEMPLO 2

- **Enunciado:** Programa que suma independientemente los pares y los impares de los números comprendidos entre 100 y 200.

EJEMPLO 2

- **Enunciado:** Programa que muestre por pantalla los 20 primeros números naturales (1, 2, 3… 20).
- **Diagrama:**



EJEMPLO 2

- **Solución:**

```
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  }
```

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

BIBLIOGRAFÍA

BIBLIOGRAFÍA

Apuntes actualizados y adaptados a partir de la siguiente documentación:

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

2. [2] Apuntes Programación de Javier Valero Lionel Tarazón. Ceedcv.

¿DUDAS?

FIN

Javier Rojo
fjrojom001@educarex.es