

ESTRUCTURAS DE REPETICIÓN

Sumario

ESTRUCTURAS DE REPETICIÓN: BUCLES.....3

 BUCLE WHILE..... 4

 BUCLE DO WHILE..... 5

 BUCLE FOR..... 7

 BUCLE FOREACH..... 10

 BUCLES INFINITOS..... 10

ESTRUCTURAS DE REPETICIÓN: BUCLES

Los bucles son bloques de código Java que nos van a permitir realizar una misma acción un número N de veces sin necesidad de replicar una y otra vez el código necesario para sólo una acción. Nos van a ayudar a simplificar nuestro código y a evitar errores de escritura (mientras menos código tiene un algoritmo, menos probabilidad de errores tendrá).

Por lo general, tanto en Java como en otros lenguajes de programación tenemos dos tipos de bucles: los controlados por condición y los controlados por iterador.

- En los bucles **controlados por condición** vamos a tener una condición booleana que debe de evaluarse o antes, o después, de ejecutar el bloque de código asociado. Esta condición booleana será la que controle que el bucle se repita mientras esa condición sea verdadera. Por ejemplo, tenemos que pedirle a un usuario que introduzca un número par y, mientras no introduzca un par, el bucle se va a repetir indefinidamente. Dentro de este tipo de bucles tenemos:
 - o **While** → Es un bucle que se caracteriza porque se ejecutará desde 0 hasta N veces. Es decir, la condición se evaluará siempre antes de ejecutar el código asociado al bucle.
 - o **Do while** → Es un bucle que se caracteriza porque se ejecutará desde 1 hasta N veces. Es decir, la condición se evaluará una vez ejecutado el bloque de código del bucle. Este bucle se ejecutará, por lo tanto, SIEMPRE al menos 1 vez.
- En los bucles **controlados por iterador** vamos a tener un iterador (o contador) que va a controlar el número de veces que se debe repetir el bucle. Por ejemplo, tenemos que pedirle a un usuario que introduzca N números para sacar una media, mientras el contador sea menor a esos N números, el bucle se va a repetir y va a seguir pidiendo números al usuario. En este tipo de bucles tenemos:
 - o **For** → Es el bucle más usado de todos. Se caracteriza por tener una variable propia que funciona a modo de contador. Puede o no iniciarse, depende de cómo lo inicialicemos y de la condición que tenga en su definición.
 - o **ForEach** → En Java se escribirá realmente “for”, al igual que el anterior. En este tipo de bucles necesitamos un objeto que implemente el interface “iterator” sobre una colección de objetos finita a recorrer. Este bucle lo estudiaremos cuando veamos las colecciones en Java.

Realmente en la práctica un bucle controlado por condición se puede convertir en controlado por iterador (haciendo que la condición sea que el iterador llegue a N cantidad) y un bucle controlado por iterador se pueda convertir en controlado por condición (haciendo uso de lo que se denominan salidas tempranas del bucle o banderas *-flags* en inglés-, de lo que veremos ejemplos más adelante).

BUCLE WHILE

Es un bucle controlado por condición booleana. Este tipo de bucle va a funcionar “parecido” a como funciona el condicional “if”: si cuando lleguemos al bucle la condición para su activación se cumple, dará inicio el while y ejecutará las instrucciones del bloque asociado, pero si no se cumple, el código saltará el while hasta la siguiente instrucción a ejecutar. Cada vez que la condición se ha cumplido, tras ejecutar el código asociado, se vuelve a evaluar la condición (es cómo si volviéramos a donde teníamos nuestro “if” imaginario)

Por tanto, este bucle se puede ejecutar 0, 1 o n veces (hasta que la condición que marca en su inicio no se cumpla).

Tiene la siguiente sintaxis:

```
while (condicion){ //es parecido a un if, pero si la condicion es cierta,
                  //no solo se ejecutan las instrucciones
                  // también se repite una y otra vez
    instrucciones;
}
```

Por ejemplo, vamos a realizar un programa que solicite al usuario introducir un numero mayor de 10 y que mientras no lo haga, el bucle se repita de forma indefinida:

```
public class BucleWhile {
    public static void main(String[] args) {
        Scanner scEntrada = new Scanner(System.in);
        int intNumUsuario;

        System.out.print("Introduzca un numero mayor de 10: ");
        intNumUsuario = scEntrada.nextInt();

        while(intNumUsuario <= 10){
            System.out.println("ERROR! Ha introducido el numero "
                               + intNumUsuario + " se requiere un numero mayor de 10");
            System.out.print("Introduzca un numero mayor de 10: ");
            intNumUsuario = scEntrada.nextInt();
        }
        scEntrada.close();

        System.out.println("Su numero es: " + intNumUsuario);
    }
}
```

Lo primero que vamos a hacer es pedirle al usuario un número para inicializar la variable que nos va a servir como condición en el while (intNumUsuario). Una vez que lleguemos al bucle while el programa hará lo siguiente:

1. Comparará si el número que ha introducido es menor o igual a 10.
2. Si lo es (por ejemplo, un 5) la condición dará como respuesta un TRUE y, por tanto, se iniciarán las instrucciones dentro del bucle, es decir, se informará al usuario del error y se le volverá a solicitar un número. Este nuevo número se sobrescribirá sobre el anterior y se volverá a valorar la condición.
3. Una vez que la condición sea FALSE (bien porque de primeras haya introducido un número mayor a 10 o bien porque lo haya introducido en una de las iteraciones) el bucle no se ejecutará y el programa pasará a la siguiente instrucción.

Este sería el resultado por consola en el caso de que introduzca un número menor y luego uno mayor:

```
D:\JDK\bin\java.exe "-javaagent:D:\IntelliJ\IntelliJ IDEA Community
Introduzca un numero mayor de 10: 4
ERROR! Ha introducido el numero 4 se requiere un numero mayor de 10
Introduzca un numero mayor de 10: 11
Su numero es: 11
```

Este sería el resultado por consola en el caso de que introduzca un número mayor ya desde el inicio:

```
D:\JDK\bin\java.exe "-javaagent:D:\Int
Introduzca un numero mayor de 10: 11
Su numero es: 11
```

Es decir, es un bucle que, como se dice más arriba puede llegar a no ejecutarse nunca ya que la inicialización de la condición se produce fuera del bucle (lectura inicial de la variable intNumUsuario).

BUCLE DO WHILE

Es otro tipo de bucle controlado por condición. Se diferencia del anterior tipo, el bucle while, tanto por la posición en la que escribimos la condición (detrás del bloque de código del bucle) como por el tiempo en el que se evalúa la condición: justo después de la ejecución del bloque de código del bucle. Es decir, al menos este bloque se ejecutará una vez. Si una vez ejecutado el bloque de código del bucle la condición se evalúa a TRUE, entonces se repite de nuevo el bloque de código, una y otra vez hasta que se evalúe la condición a FALSE.

Dicho de otra forma, la condición, que se valorará al final, también funciona como un if (ya que en sí misma es un while) de forma que, si da como resultado

un TRUE, el bucle se repetirá y si da como resultado un FALSE, el bucle se romperá y la ejecución del programa seguirá hacia delante.

Su sintaxis sería la siguiente:

```
do {  
    instrucciones;  
} while (condicion);
```

Por ejemplo, vamos a crear un programa que cree una suma aleatoria de dos números (comprendidos entre 1 y 10) y vamos a preguntarle al usuario el resultado:

```
public class BucleDoWhile {  
    public static void main(String[] args) {  
        Scanner scEntrada = new Scanner(System.in);  
        int intNum1;  
        int intNum2;  
        int intResultadoUsuario;  
        int intResultadoReal;  
  
        intNum1 = (int) (Math.random() * 10 + 1);  
        intNum2 = (int) (Math.random() * 10 + 1);  
        intResultadoReal = intNum1 + intNum2;  
  
        System.out.println("Calcule la siguiente suma:");  
        do{  
            System.out.print(intNum1 + " + " + intNum2 + " = ");  
            intResultadoUsuario = scEntrada.nextInt();  
        }while (intResultadoUsuario != intResultadoReal);  
        scEntrada.close();  
  
        System.out.println("Has acertado.");  
    }  
}
```

En este ejemplo lo primero que va a hacer el programa será crear 2 números aleatorios entre 1 y 10 los cuales se guardarán en 2 variables: `intNum1` e `intNum2` y, a continuación, realizará la suma de estos números la cual se guardará en la variable `intResultadoReal`.

Una vez completado este primer bloque de instrucciones, el programa seguirá adelante y llegará al bucle `do while`:

1. Solicitará al usuario que calcule la suma de los 2 números.
2. El usuario introducirá un resultado.
3. El programa comparará ese resultado (`intResultadoUsuario`) con el resultado real de la suma, si este resultado es distinto dará como respuesta un TRUE a la condición y deberá repetirse el bucle solicitando

al usuario que vuelva a hacer la operación. Si, por el contrario, el usuario acierta el resultado, la condición devuelve un FALSE (ya que `intResultadoUsuario` será igual a `intResultadoReal`) y se romperá un bucle continuando con la ejecución del programa.

Este sería el caso de un bucle donde el usuario no acierta a la primera:

```
D:\JDK\bin\java.exe "-javaa
Calcule la siguiente suma:
1 + 6 = 8
1 + 6 = 9
1 + 6 = 7
Has acertado.
```

Este sería el caso de un bucle donde el usuario acierta a la primera:

```
D:\JDK\bin\java.exe "-javaag
Calcule la siguiente suma:
1 + 9 = 10
Has acertado.
```

En este caso podemos ver que el bloque de instrucciones (el solicitar la suma) se ejecuta siempre al menos una vez, por lo que este bucle se va ejecutar al menos una vez y durante un número N de veces (hasta que la condición de la parte `while` no se cumpla, en este caso, hasta que el usuario acierte el resultado, lo cual puede que sea NUNCA. Hablaremos de los bucles infinitos más adelante).

BUCLE FOR

Es un tipo de bucle controlado por un iterador (contador) que decide el número de veces que se debe ejecutar el bucle.

Tiene la siguiente sintaxis:

```
for (inicializacion; condicion; incremento) {
    Instrucciones;
}
```

Es decir:

- Inicialización. Es la variable (o variables) que va a funcionar como contador. Esta variable puede iniciarse en el propio `for` o fuera. Por ejemplo, vamos a iniciar un contador llamado `intCont` que comience en 0: `int intCont = 0`. Se ejecuta al inicio, antes de empezar el propio bucle, solo una vez.

- Condición. Al igual que en los anteriores va a funcionar como una condición if, mientras la condición se cumpla va a devolver un TRUE y, por tanto, se va a ejecutar el bloque de código asociado y después se ejecutará el incremento (explicado en el siguiente párrafo). Un instante después volverá a evaluar la condición. Cuando la condición en alguna de las iteraciones no se cumpla esta devolverá un FALSE y el bucle se romperá de forma que el programa seguirá hacia delante justo detrás del bloque del bucle for.
- Incremento. Será la modificación de la variable que controla la condición. Se ejecuta al final de cada iteración.

Por lo tanto, la inicialización se ejecuta una vez y solo una vez, el bucle y el incremento entre 0 y N veces.

Vamos a crear un programa que, a través de un for, nos calcule la suma de N numeros:

```
public class BucleFor {
    public static void main(String[] args) {
        Scanner scEntrada = new Scanner(System.in);
        int intCantidadNumeros;
        int intSumaTotal = 0;
        int intNumUsuario;

        System.out.print("Introduzca cantidad de numeros que quiere sumar: ");
        intCantidadNumeros = scEntrada.nextInt();

        for (int intCont = 0; intCont < intCantidadNumeros; intCont++){
            System.out.println("ITERACION " + (intCont + 1));
            System.out.print("Introduzca numero " + (intCont + 1) + ": ");
            intNumUsuario = scEntrada.nextInt();
            intSumaTotal += intNumUsuario;
        }

        System.out.println("La suma es: " + intSumaTotal);
    }
}
```

En este caso vamos a pedirle al usuario una cantidad de números que guardaremos en la variable intCantidadNumeros.

Y declaramos un bucle for que tiene los siguientes parámetros:

- Inicialización: int intCont = 0. Vamos a ir desde un contador 0 hasta N veces donde N será intCantidadNumeros - 1. Eso hace, que si por ejemplo, introducimos en intCantidadNumeros el número 4, que intCont tome los valores 0, 1, 2, 3 y cuando llegue al 4, al no cumplir la condición de ser menor, se abandona el bucle, ejecutándose este 4 veces.
- Condición: intCont < intCantidadNumeros. El bucle se va a repetir mientras que esta condición de un TRUE, es decir, mientras intCont sea menor a la cantidad de números que quiere introducir el usuario. Es

menor porque iniciamos el contador a 0, si iniciáramos el contador a 1, la condición debería ser esta: `intCont <= intCantidadNumeros`.

- Incremento. Al final de cada iteración sumamos uno al iterador.

Bien, vamos a suponer que el usuario quiere introducir 3 números esto es lo que haría internamente el for:

- ITERACIÓN 1

`for (int intCont = 0; intCont < 3; intCont++)`

1. Inicializa `intCont` a 0.
2. Comprueba que `intCont` es menor que 3 (la condición) → Como lo es da como resultado un `TRUE` y, por tanto, ejecuta las instrucciones que se encuentran dentro del bucle.
3. Una vez completadas las instrucciones manda la orden de que `intCont` debe aumentar en 1 su valor.

```
D:\JDK\bin\java.exe "-javaagent:D:\IntelliJ\IntelliJ
Introduzca cantidad de numeros que quiere sumar: 3
ITERACION 1
Introduzca numero 1: 10
```

- ITERACIÓN 2

`//intCont == 1` pues venimos de `intCont++`

`for (; intCont < 3; intCont++)`

1. Comprueba que `intCont` es menor que 3 → Como lo es da como resultado un `TRUE` y ejecuta las instrucciones del bucle.
2. Se vuelve a aumentar `intCont` en 1. Ahora `intCont` es 2

```
ITERACION 2
Introduzca numero 2: 12
```

- ITERACIÓN 3

`for (; intCont < 3; intCont++)`

1. Sabemos que `intCont` es 2 pues venimos de `intCont++` sobre `intCont==1`.
2. Comprueba que `intCont` es menor que 3 → Como lo es informa de un `TRUE` y ejecuta las instrucciones del bucle.
3. Se vuelve a aumentar el `intCont` en 1.

```
ITERACION 3
Introduzca numero 3: 15
```


- ITERACIÓN 4

`for (; intCont < 3; intCont++)`

1. Ahora `intCont` es 3.

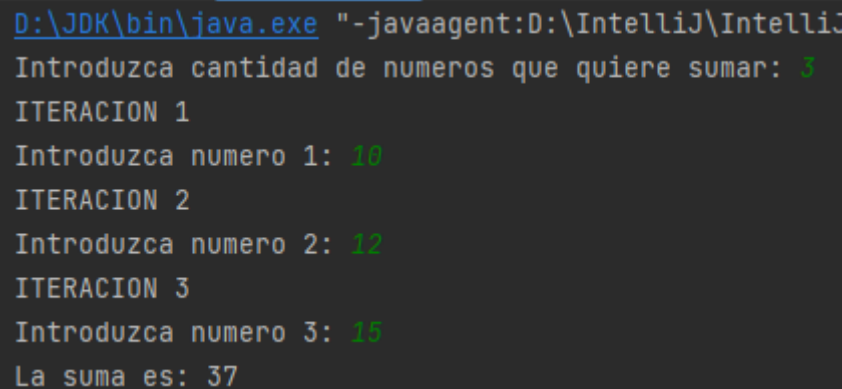
2. Comprueba que intCont es menor que 3 → Como NO lo es da como resultado un FALSE y no ejecuta las instrucciones (se rompe el bucle)

El programa completa su función informando al usuario de la suma de los 3 números.



```
La suma es: 37
```

La consola de resultados completa sería esta:



```
D:\JDK\bin\java.exe "-javaagent:D:\IntelliJ\IntelliJ
Introduzca cantidad de numeros que quiere sumar: 3
ITERACION 1
Introduzca numero 1: 10
ITERACION 2
Introduzca numero 2: 12
ITERACION 3
Introduzca numero 3: 15
La suma es: 37
```

Es decir, el bucle se ha repetido un total de 3 veces (la 4 iteración no ha llegado a ejecutarse ya que el contador era igual a la cantidad de números que quería introducir el usuario).

BUCLE FOREACH

Aunque en algunos lenguajes se escribe foreach para este tipo especial de for, en Java mantenemos la palabra reservada for tanto para el for “clásico” como para el for-each, estando por lo tanto “sobrecargada”. Para usar este tipo de for necesitamos una colección iterable, y un objeto que hace de iterador. Como estos conceptos los estudiaremos más adelante en el curso, dejaremos la explicación del bucle for de tipo foreach para ese momento.

BUCLES INFINITOS

Cuando se diseña un algoritmo iterativo, es decir, que busca solucionar un problema que necesita ejecutar un proceso un número repetido de veces, tenemos que tener en cuenta en su diseño no cometer el error (muy común cuando se empieza a programar) de crear un bucle de infinitas ejecuciones. Estos bucles infinitos son los responsables, en la gran mayoría de casos, de los “cuelgues” de vuestros dispositivos (PCs, portátiles, móviles, etc), de esos pantallazos “fijos” en los que los programas dejan de “responder” a vuestras interacciones.

Pero, ¿qué es un bucle infinito? Por definición un bucle infinito es cualquier bucle cuya condición booleana evaluada NUNCA se evalúa a False. Por lo

tanto, el sistema de ejecución no parará una y otra vez de ejecutar el bloque de código asociado al bucle no llegando a terminar, nunca. De ahí su nombre: bucle infinito.

Por ejemplo, veamos el siguiente bucle que podemos poner en cualquier lugar de alguno de nuestros ejercicios:

```
for (int i=0; i>=0; i++) {  
    System.out.println("Vamos por la iteración: " + i);  
}
```

Si analizamos el bucle, vemos que *i* empezará en 0 y después ira aumentando el valor sumando un 1 en cada iteración (*i++*). El problema como describíamos en la definición está en la condición: SIEMPRE será verdadera, pues el programador ha puesto que para repetir una y otra vez el número *i* debe ser 0 o mayor que 0, y esto siempre se cumple en cada una de las iteraciones. Si ejecutáis este código, tendréis que parar la ejecución a mano en vuestro IDE.

Hay más ejemplos de bucles infinitos. Por ejemplo:

```
boolean bSeguir = true;  
while (bSeguir) {  
    System.out.println("El valor de la condición es: " + bSeguir);  
}  
//Otro más:  
while(true) {  
    //otro más  
}
```

Por lo tanto, cuando programéis los bucles, comprobad que contando con la condición inicial de llegada al bucle, su condicional y los cambios de estado de las variables, en algún momento la condicional será FALSE.