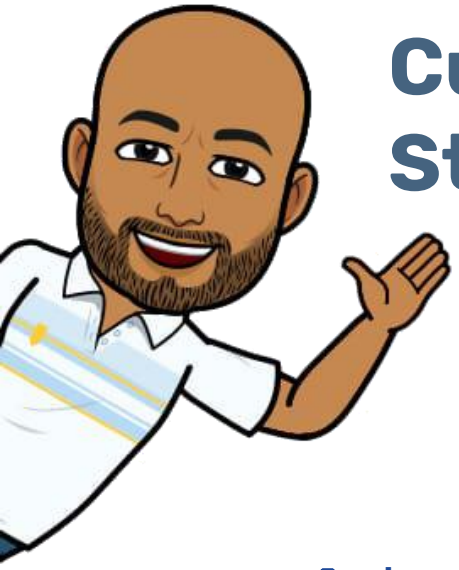


Curso de Java Standard

Ing. Octavio Robleto

octavio.robleto@gmail.com

https://octaviorobleto.com

Introducción

También conocidas como estructuras repetitivas, nos permiten ejecutar varias veces los mismos bloques o sentencias mientras se cumpla una o varias condiciones.

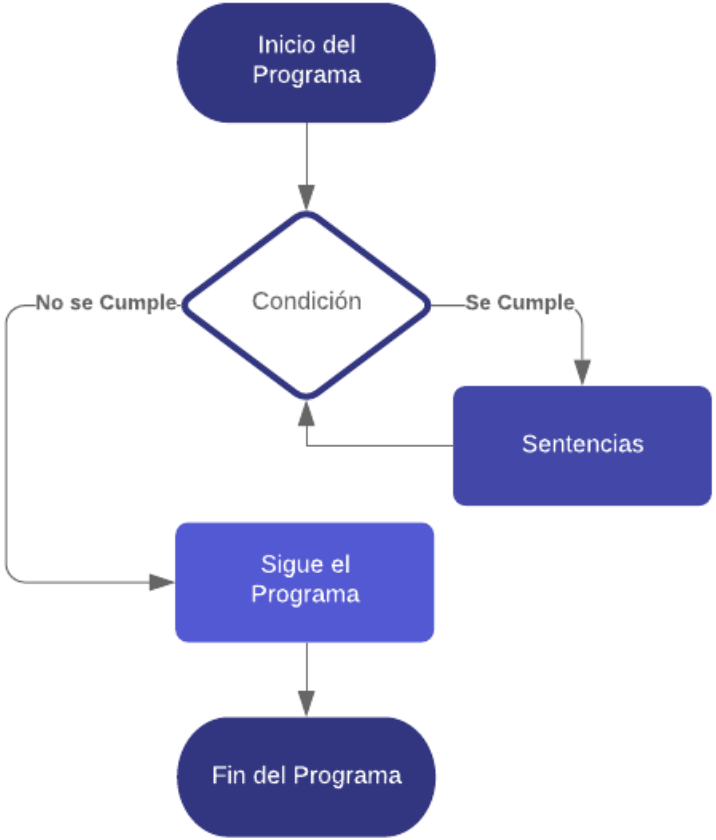
A cada entrada en el ciclo (bucle) se le conoce como iteración.

Existen dos clasificaciones:

1. **Indeterminados**: no sabemos a ciencia cierta la cantidad de iteraciones que realizará, estos ciclos son el **while** y do **while**.

2. **Determinados**: son aquellos ciclos que sabemos de antemano la cantidad de iteraciones máximas que va a tener por ejemplo el ciclo **for**.

Algo muy importante de los ciclos es que debemos garantizar que en el algún momento deje de cumplirse la condición para así poder salir de el; de lo contrario quedaría en un bucle infinito.



while

Esta estructura es muy similar a la que vimos en el condicional if, la diferencia radica en que si se cumple la condición ejecutara las sentencias y/o bloques que se encuentren dentro, volverá al inicio del mismo y evaluará la condición nuevamente, esto se realizara hasta que la condición deje de cumplirse.

```
// con llaves
while (condicion) {
    sentencia1;
    sentencia2;
}

// sin llaves
while (condicion)
    sentenciaUnica;
```

```
int num1, num2;
num1 = 0;
num2 = 5;

while (num1 <= num2) {
    System.out.println("Veces dentro del ciclo:" + (num1 + 1));
    num1++;
}

int num = 0;
boolean seguir = true;

while(seguir){
    System.out.println("Veces dentro del ciclo:" + (num + 1));
    if (num == 3) {
        seguir = false;
    }
}
```

do while

Trabaja de forma muy similar al while, la única diferencia es que en este ciclo la condición se evalúa al final.

Esto permite que las sentencias se ejecuten por lo menos una vez, a diferencia del “while” que evalúa la condición al principio y puede que nunca se ejecuten las sentencias que contiene.

```
// con llaves
do {
    sentencia1;
    sentencia2;
} while (condicion);

// sin llaves
do
    sentenciaUnica;
while (condicion);
```

```
int num1, num2;
num1 = 0;
num2 = 5;

do {
    System.out.println("Veces dentro del ciclo:" + (num1 + 1));
    num1++;
} while (num1 <= num2);

int num = 0;
boolean seguir = true;

do {
    System.out.println("Veces dentro del ciclo:" + (num + 1));
    if (num == 3) {
        seguir = false;
    }
} while (seguir);
```

for

Si detallamos el siguiente código podremos notar que, aunque sabemos que el while esta entre los ciclos indeterminados, vemos que tenemos la cantidad máxima de iteraciones en 10, además, podemos observar que para que el bucle no se vuelva infinito se incrementa al final del bloque la variable a comparar en el condicional.

```
int num1 = 0;

while (num1 < 10) {
    System.out.println("Veces dentro del ciclo:" + (num1 + 1));
    num1++;
}
```

Para este tipo de situaciones tenemos el ciclo for, que posee una forma mas cómoda de hacer lo mismo.

```
// con llaves
for (inicio; condicion; depuesSentencias) {
    sentencia1;
    sentencia2;
}

// sin llaves
for (inicio; condicion; depuesSentencias)
    sentenciaUnica;
```

```
for (int num2 = 0; num2 < 10; num2++) {
    System.out.println("Veces dentro del ciclo:" + (num2 + 1));
}
```

for

Esta estructura tiene algo en particular y es que podemos omitir lo que contiene entre los paréntesis, pero sin dejar de colocar los punto y coma.

```
// con llaves
for ( ; ; ) {
    sentencia1;
    sentencia2;
}
```

Lo que logramos con esto, es que lo que se encuentre dentro del bloque se ejecute infinitamente, y por que nos permite esto? Algunas veces dependiendo de la solución necesitamos que el bloque de incremento o bloque “despuesSentencias” se ejecute antes, otras veces, que la variable inicial sea usada después de haber termina el ciclo.

```
int num2 = 0
for ( ; num2 < 10; ) {
    num2++;
    System.out.println("Veces dentro del ciclo:" + (num2 + 1));
}
System.out.println("Valor final de la variable" + num2);
```

break y continue

La sentencia “**break”** la vimos en el condicional switch que hacia que saliera de la estructura, en este caso hace exactamente lo mismo indicándole al programa que salga del ciclo inmediato al que pertenece.

La otra sentencia “**continue**” le indica al programa que omita todas las instrucciones que se encuentren después de esta palabra dentro del bucle y salte a la siguiente iteración.