**BUCLES**

Los bucles son elementos Java que nos van a permitir realizar una misma acción un número N de veces sin necesidad de replicar el código necesario, por tanto, nos van a ayudar a simplificar código y a evitar errores de escritura.

Principalmente tenemos 2 tipos de bucles: los controlados por condición y los controlados por iterador.

* En los bucles **controlados por condición** vamos a tener una condición que va a controlar que el bucle se repita hasta que esa condición no se cumpla. 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:
  + ***While*** 🡪 Es un bucle que se caracteriza porque se puede iniciar desde 0 hasta N veces. Es decir, la condición que debe tener en cuenta debe estar inicializada de antes.
  + ***Do while*** 🡪 Es un bucle que se caracteriza porque se puede iniciar desde 1 hasta N veces. Es decir, la condición que debe tener en cuenta se inicializa dentro del bucle. Este bucle se hara SIEMPRE al menos 1 vez.
* En los bucles **controlados por iterador** vamos a tener un iterador (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 contrador 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:
  + ***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.

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-).

**BUCLE WHILE**

Es un bucle controlado por condición que puede iniciarse o no en función de una variable que se haya iniciado antes de la llegada al bucle. Es decir, este bucle va a funcionar como un if, si cuando lleguemos al bucle la condición para su activación se cumple, dará inicio el while, si no se cumple, el código saltará el while hasta la siguiente instrucción a ejecutar.

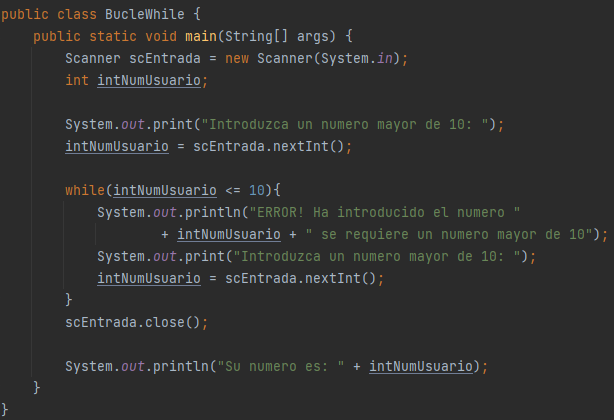
Por tanto, este bucle se puede iniciar o no y se va a repetir un número N de veces (hasta que la condición que marca en su inicio no se cumpla).

Tiene la siguiente sintaxis:

while (condicion){

instrucciones;

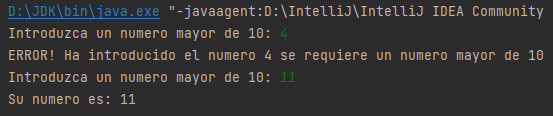
}

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:

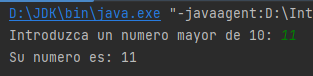
Lo primero que vamos a hacer es pedirle al usuario un número para inicializar la variable que nos va a servir como condición en el while (intNumUsuario). Una vez que lleguemos al bucle while el programa hará lo siguiente:

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 sobreescribirá 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 numero 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:



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



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.

**BUCLE DO WHILE**

Es otro tipo de bucle controlado por condición, se diferencia del while en que en este caso siempre se va a iniciar al menos una vez ya que la condición a valorar no se encuentra fuera del bucle sino dentro.

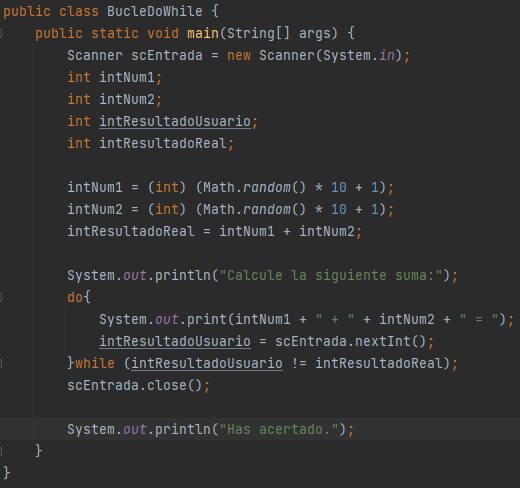
La condición, que se valorará al final, también funciona como un if (ya que en sí misma es un while) de forma que, si da como resultado un TRUE, el bucle se repetirá y si da como resultado un FALSE, el bucle se romperá y la ejecución del programa seguirá hacia delante.

Su sintaxis sería la siguiente:

do {

instrucciones;

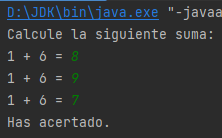
} while (condicion);

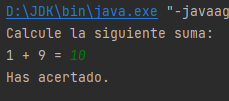
Por ejemplo, vamos a crear un programa que cree una suma aleatoria de dos números (comprendidos entre 1 y 10) y vamos a preguntarle al usuario el resultado:

En este ejemplo lo primero que va a hacer el programa será crear 2 números aleatorios entre 1 y 10 los cuales guardará 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 y 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:

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

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 numero N de veces (hasta que la condición de la parte while no se cumpla, en este caso, hasta que el usuario acierte el resultado).

**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:

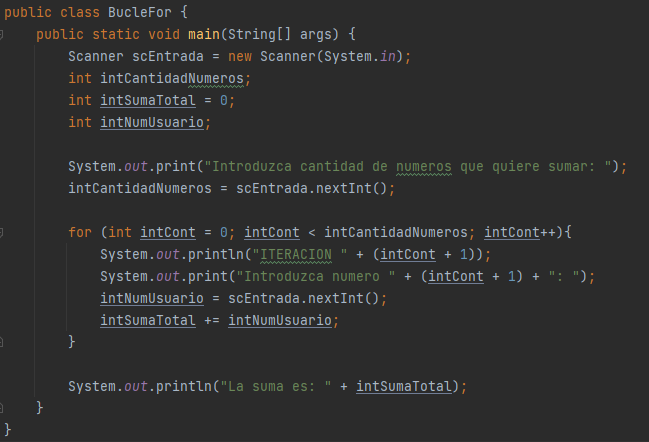
for (inicializacion; condicion; incremento) {

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 de cada iteración.
* 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 repetir el bucle, cuando la condición no se cumpla devolverá un FALSE y el bucle se romperá de forma que el programa seguirá hacia delante.
* Incremento. Será la modificación de la variable que controla la condición. Se ejecuta al final de cada iteración.

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

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

Y declaramos un bucle for que tiene los siguientes parámetros:

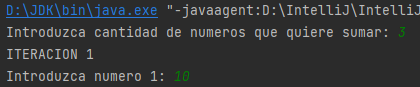
* Inicialización: int intCont = 0. Vamos a ir desde un contador 0 hasta N veces.
* Condición: intCont < intCantidadNumeros. El bucle se va a repetir mientras que esta condición de un TRUE, es decir, mientras intCont sea menor a la cantidad de números que quiere introducir el usuario. Es menor porque iniciamos el contador a 0, si iniciaramos 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.



* *ITERACIÓN 2*

for (intCont = 1; intCont < 3; intCont++)

1. Cambia el valor de intCont a 1.
2. Comprueba que intCont es menor que 3 🡪 Como lo es da como resultado un TRUE y ejecuta las instrucciones del bucle.
3. Se vuelve a aumentar intCont en 1.



* *ITERACIÓN 3*

for (intCont = 2; intCont < 3; intCont++)

1. Cambia el valor de intCont a 2.
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.



* *ITERACIÓN 4*

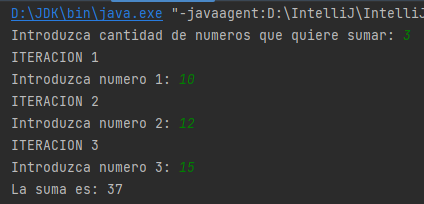
for (intCont = 3; intCont < 3; intCont++)

1. Cambia el valor de intCont a 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 consola de resultados completa sería esta:



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 al número de números que quería introducir el usuario).