
ORACLE
Academy

Java Foundations

6-2

Bucles while y do-while

ORACLE
Academy

Objetivos

- En esta lección se abordan los siguientes objetivos:
  - Usar un bucle while en un programa Java (antes de la prueba)
  - Usar un bucle do-while en un programa Java (después de la prueba)
  - Comprender cuándo es más beneficioso un tipo de bucle frente a otro

¿Cuántas veces se tiene que repetir?

- En algunas situaciones, no se sabe cuántas veces se tiene que repetir algo
- Es decir, puede que tenga que repetir algún código hasta que se produzca una condición concreta

¿Cuántas veces se tiene que repetir?

- Observe un ejemplo:
  - Imagine que tiene que escribir un programa para introducir las calificaciones de los exámenes y averiguar su media, pero puede que no sepa cuántos exámenes se realizarán
  - En lugar de obligar a los usuarios a contarlos con antelación, puede permitirles que introduzcan las calificaciones de una en una y, a continuación, introducir -1 para indicar la finalización de las entradas

Bucle while

- En esos casos, deberá utilizar el bucle **while** más fácil
- Funciona de la siguiente manera:
  - El bucle **while** ejecuta continuamente un bloque de sentencias siempre que una condición concreta sea true

Sintaxis del bucle while

- La sentencia while evalúa la boolean expression
- Las sentencias en los corchetes angulares se ejecutan siempre que boolean expression sea true

```
while (<boolean expression>) {
    <statement(s)> ;
}//end while
```

Bucle previo a la prueba

- Un bucle previo a la prueba evalúa la condición antes de que el bucle se ejecute
- Si la condición es false, el bucle se para o puede que nunca se ejecute
- Los bucles for y while son previos a la prueba

Escenario de la cuenta atrás

- Vamos a escribir el escenario de la cuenta atrás tratado en la lección anterior mediante el bucle while:

Lo que conocemos	Nombre técnico	Código
Cuando se inicia el bucle...	Expresión de inicialización	`int i = 10;`
Continúe el bucle si...	Expresión de condición	`i >= 0;`
Después de cada bucle...	Expresión de actualización	`i--;`
Código para repetir	Sentencias de código	`System.out.println(i);`

Escenario de la cuenta atrás: Bucle while

```
public class CountDownWhile {

    public static void main(String[] args) {
        int i = 10;
        System.out.println("Countdown to Launch!");

        while (i >= 0) {
            System.out.println(i);
            i--;
        }//end while

        System.out.println("Blast Off!");
    }//end method main
}//end class CountDownWhile
```

Algunos bucles while nunca se ejecutan

- Es posible que el cuerpo del bucle nunca se ejecute Si las condiciones son tales que la expresión booleana ya era false, por ejemplo:

```
public class WhileLoopExample {
   public static void main(String args[]) {
      int num = 0;
      System.out.println("Let's count to 10!");
         while (num > 10) {
            num = num + 1;
            System.out.println("Number: " + num);
         }//end while
      System.out.println("We have counted to 10! Hurrah!");
   }//end method main
}//end class WhileLoopExample
```


En el ejemplo de la diapositiva, el valor inicial de `num` es `0` y la expresión booleana es `num > 10`, en lugar de `num < 10`. Ya es false desde el principio, ya que 0 nunca será mayor que 10. El bucle while evalúa la expresión booleana, `num > 10`, detecta que es false e imprime:

```
Let's count to 10!
We have counted to 10! Hurrah!
```

Quedar atascado en un bucle infinito

- Se quedará atascado en un bucle while si escribe una condición booleana que nunca se evalúe como false
- Esto se denomina un **bucle infinito** porque nunca deja de ejecutarse
- Si esto ocurre, el bucle se ejecutará para siempre o hasta que se envíe un comando de interrupción
- Debe evitar escribir bucles infinitos y comprobar siempre la expresión booleana para asegurarse de que los bucles terminan con normalidad

Volvamos al escenario de la cuenta atrás

- ¿Qué sucede si accidentalmente habíamos escrito **`i++`** en lugar de **`i--`** en el bucle while?

```
int i = 10;
System.out.println("Countdown to Launch!");
while (i >= 0) {
   System.out.println(i);
   i++;
}//end while
System.out.println("Blast Off!");
```

- Continúa agregando 1 a **`i`**, manteniendo su valor a más de 10 para siempre
- Se trata de un bucle infinito porque la condición booleana es siempre true y este programa continúa la ejecución

Uso del bucle while y la clase scanner

- Los bucles while a menudo se utilizan con entrada mediante la clase Scanner

```
public static void main(String[] args) {
   Scanner console = new Scanner(System.in);
   int sum = 0;

   System.out.println("Enter a number (-1 to quit): ");
   int num = console.nextInt();
   while (num != -1) {
       sum = sum + num;
       System.out.println("Enter a number (-1 to quit): ");
       num = console.nextInt();
   }//end while

   System.out.println("The sum is " + sum);
}//end method main
```

ORACLE Academy

JFo 6-2
Bucles while y do-while

El ejemplo de la diapositiva genera la siguiente salida:

```
Enter a number (-1 to quit):
20
Enter a number (-1 to quit):
40
Enter a number (-1 to quit):
-1
The sum is 60
```

Uso del bucle while y la clase Scanner

- Ejemplo:
  - Un programa que solicita a los usuarios números hasta que escriben -1 y, a continuación, genera la suma

```
public static void main(String[] args) {
   Scanner console = new Scanner(System.in);
   int sum = 0;

   System.out.println("Enter a number (-1 to quit): ");
   int num = console.nextInt();
   while (num != -1) {
       sum = sum + num;
       System.out.println("Enter a number (-1 to quit): ");
       num = console.nextInt();
   }//end while

   System.out.println("The sum is " + sum);
}//end method main
```


El ejemplo de la diapositiva genera la siguiente salida:

```
Enter a number (-1 to quit):
20
Enter a number (-1 to quit):
40
Enter a number (-1 to quit):
-1
The sum is 60
```

Ejercicio 1

- Cree un nuevo proyecto y agréguele el archivo `SquareRootWhile.java`
- Modifique `SquareRootWhile.java` para utilizar un bucle while para solicitar repetidamente a los usuarios que escriban un número hasta que escriban un número no negativo y, a continuación, se calcula la raíz cuadrada
- Salida esperada:

```
Type a non-negative integer: -5
Invalid number, try again: -1
Invalid number, try again: 11
The square root of 11 is 3.166
```

Bucle posterior a la prueba

- Un bucle posterior a la prueba evalúa su condición al final del bucle, no al principio
- El bucle do-while es posterior a la prueba

Bucle do-while

- El bucle do-while es un bucle while modificado que le permite ejecutar el bucle una vez, antes de comprobar la condición booleana
- Sintaxis:

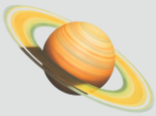
```
do{
 <statement(s)>
}while(<condition>);
```

El bucle do-while necesita un punto y coma después de la condición al final del bucle

ORACLE Academy

Si la condición es false, el bucle todavía se ejecuta al menos una vez, pero se para al final del bucle. Por lo tanto, las sentencias en el bloque do se ejecutan siempre, al menos una vez.

Escenario de la cuenta atrás: Bucle do-while



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

    int i = 10;
    System.out.println("Countdown to Launch!");

    do {
        System.out.println(i);
        i--;
    }while (i >= 0);

    System.out.println("Blast Off!");
}//end method main
```

Ejecutado una vez antes de evaluar la condición

ORACLE Academy

JFo 6-2
Bucles while y do-while

Resultado:

```
Countdown to Launch!
10
9
8
7
6
5
4
3
2
1
0
Blast Off!
```

Ejercicio 2

- Agregue el archivo `SumofNums.java` al proyecto creado para el ejercicio 1
- Examine `SumofNums.java`, que suma una secuencia de 10 números enteros que introduce el usuario
- ¿Se puede implantar el mismo mediante un bucle do-while?

Bucle for estándar en comparación con el bucle while

- Diferencias entre estos dos bucles:
- En un bucle for:
  - Las sentencias de inicialización, condición e incremento se unen en una sola línea, lo que hace que el bucle sea más fácil de entender e implantar

Bucle for estándar en comparación con el bucle while

- Diferencias entre estos dos bucles:
- En un bucle while:
  - La inicialización se realiza antes de iniciar el bucle
  - La sentencia condicional siempre se pone en el inicio del bucle
  - Las sentencias de incremento se pueden combinar con una condición o embeber en el cuerpo del bucle

En las siguientes tres diapositivas, verá un ejemplo de bucle `while` en la parte superior de la diapositiva. En la parte inferior, verá la misma lógica implantada mediante un bucle `for` estándar.
Los tres elementos esenciales de un bucle `while` también se encuentran presentes en el bucle `for`, pero en distintos lugares.
1. El contador (i) se ha declarado y se inicializa fuera del bucle `while` en la línea 1.
2. El contador se ha incrementado en el bucle `while` en la línea 4.
3. La expresión booleana que determina el número de iteraciones de bucles aparece entre paréntesis para el bucle `while` en la línea 2.
En el bucle `for`, los tres elementos aparecen entre paréntesis, como se indica en la diapositiva.
La salida de cada sentencia es la misma.

Comparación del contador de inicialización

Bucle `while`

```
int i = 10;
while (i >= 0) {
    System.out.println(i);
    i--;
} //end while
System.out.println("Blast Off!");
```

Inicializar contador

Bucle `for`

```
for (int i = 10; i >= 0; i--) {
    System.out.println(i);
}//end for
System.out.println("Blast Off!");
```

Comparación de la expresión booleana

Bucle `while`

```
int i = 10;
while (i >= 0) {
    System.out.println(i);
    i--;
} //end while
System.out.println("Blast Off!");
```

Bucle `for`

```
for (int i = 10; i >= 0; i--) {
    System.out.println(i);
}//end for
System.out.println("Blast Off!");
```

Comparación del contador de incremento

Bucle `while`

```
int i = 10;
while (i >= 0) {
    System.out.println(i);
    i--;
} //end while
System.out.println("Blast Off!");
```

Incrementar contador

Bucle `for`

```
for (int i = 10; i >= 0; i--) {
    System.out.println(i);
}//end for
System.out.println("Blast Off!");
```

¿Qué bucle puedo utilizar?

Tipo de bucle	Definición	¿Cuándo utilizarlo?
while	Bucle previo a la prueba que se repite hasta que una condición especificada sea false	Se debe utilizar cuando no está seguro del número de veces que el bucle se debe ejecutar o incluso si no debe en absoluto
do-while	Bucle posterior a la prueba que ejecuta el bucle antes de comprobar la condición y, a continuación, se repite hasta que la condición sea false	Se debe utilizar cuando sepa que el código se debe ejecutar al menos una vez y, posiblemente, más veces en función de la condición
for	Bucle que contiene un contador inicializado y aumenta el contador con cada ejecución a través del bucle Se repite hasta que la condición sea false	Se utiliza cuando tenga que ejecutar un bucle un número determinado de veces, o cuando lo tiene que incrementar a través de un juego de datos El contador también se puede utilizar como un índice para acceder a los datos de uno en uno

Resumen

- En esta lección, debe haber aprendido lo siguiente:
  - Usar un bucle while en un programa Java (antes de la prueba)
  - Usar un bucle do-while en un programa Java (después de la prueba)
  - Comprender cuándo es más beneficioso un tipo de bucle frente a otro


ORACLE
Academy