

Uso de construcciones de bucle

Bucles

Los bucles se utilizan frecuentemente en programas para repetir bloques de sentencias hasta que una expresión es false.

Hay tres tipos principales de bucles:

- Bucle `while`: se repite hasta que una expresión es true.
- Bucle `do/while`: se ejecuta una vez y, a continuación, se sigue repitiendo mientras es true.
- Bucle `for`: se repite un número definido de veces.

Comportamiento de repetición



```
while (!areWeThereYet) {  
  
    read book;  
    argue with sibling;  
    ask, "Are we there yet?";  
  
}  
  
Woohoo!;  
Get out of car;
```

Creación de bucles while

Sintaxis:

```
while (boolean_expression) {  
    code_block;  
} // end of while construct  
  
// program continues here
```

Si la expresión booleana es true, se ejecutará este bloque de código.

Si la expresión booleana es false, el programa sigue aquí.

Bucle while en Elevator

```
public void setFloor() {  
    // Normally you would pass the desiredFloor as an argument to the  
    // setFloor method. However, because you have not learned how to  
    // do this yet, desiredFloor is set to a specific number (5)  
    // below.
```

```
    int desiredFloor = 5;  
    while ( currentFloor != desiredFloor ){  
        if (currentFloor < desiredFloor) {  
            goUp();  
        }  
        else {  
            goDown();  
        }  
    }  
}
```

Si la expresión
booleana devuelve
true, se ejecutará el
bucle while.

Tipos de variables

```
public class Elevator {  
    public boolean doorOpen=false;  
    public int currentFloor = 1;  
    public final int TOP_FLOOR = 10;  
    public final int BOTTOM_FLOOR = 1;
```

Variables de
instancia (campos)

... < lines of code omitted > ...

```
public void setFloor() {  
    int desiredFloor = 5;  
    while ( currentFloor != desiredFloor ){  
        if (currentFloor < desiredFloor) {  
            goUp();  
        } else {  
            goDown();  
        }  
    } // end of while loop  
} // end of method  
} // end of class
```

Variable local

Ámbito de
desiredFloor

Bucle while: Ejemplo 1

Ejemplo:

```
float square = 4;    // number to find sq root of
float squareRoot = square;    // first guess
while (squareRoot * squareRoot - square > 0.001) { // How accurate?
    squareRoot = (squareRoot + square/squareRoot)/2;
    System.out.println("Next try will be " + squareRoot);
}
System.out.println("Square root of " + square + " is " + squareRoot);
```

Resultado:

```
Next try will be 2.5
Next try will be 2.05
Next try will be 2.0006099
Next try will be 2.0
The square root of 4.0 is 2.0
```

Bucle while: Ejemplo 2

Ejemplo:

```
int initialSum = 500;
int interest = 7;           // per cent
int years = 0;
int currentSum = initialSum * 100; // Convert to pennies
while ( currentSum <= 100000 ) {
    currentSum += currentSum * interest/100;
    years++;
    System.out.println("Year " + years + ": " + currentSum/100);
}
```

Comprueba si el dinero se ha duplicado ya.

Si no se ha duplicado, agrega el interés de otro año.

Resultado:

```
... < some results not shown > ...
Year 9: 919
Year 10: 983
Year 11: 1052
```

El bucle while se itera 11 veces antes de que la prueba booleana se evalúe en true.

Bucle while con contador

Ejemplo:

```
System.out.println("  /*");
```

```
int counter = 0;
```

```
while ( counter < 4 ) {
```

```
    System.out.println("    *");
```

```
    counter ++;
```

```
}
```

```
System.out.println("  */");
```

Declarar e inicializar una variable de contador.

Comprobar para ver si el contador ha excedido el valor 4.

Imprimir un asterisco e incrementar el contador.

Salida:

```
/*
```

```
*
```

```
*
```

```
*
```

```
*
```

```
*/
```

Bucle for

Bucle while:

La inicialización de la variable de contador se mueve aquí.

```
int counter = 0;
while ( counter < 4 ) {
    System.out.println("    *");
    counter ++;
}
```

El incremento del contador va aquí.

Bucle for:

```
for ( int counter = 0 ; counter < 4 ; counter++ ) {
    System.out.println("    *");
}
```

Las expresiones booleanas permanecen aquí.

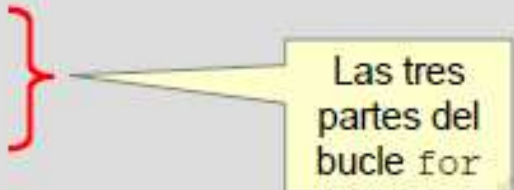
Desarrollo de un bucle for

Sintaxis:

```
for (initialize[,initialize]; boolean_expression; update[,update]) {  
  
    code_block;  
  
}
```

Ejemplo:

```
for (String i = "|", t = "-----";  
     i.length() < 7 ;  
     i += "|", t = t.substring(1) ) {  
  
    System.out.println(i + t);  
  
}
```



Las tres
partes del
bucle for

Bucle for anidado

Código:

```
int height = 4;
int width = 10;

for (int rowCount = 0; rowCount < height; rowCount++ ) {

    for (int colCount = 0; colCount < width; colCount++ ) {
        System.out.print("@");
    }
    System.out.println();
}
```

Bucle while anidado

Código:

```
String name = "Lenny";
String guess = "";
int numTries = 0;

while (!guess.equals(name.toLowerCase())) {
    guess = "";
    while (guess.length() < name.length()) {
        char asciiChar = (char)(Math.random() * 26 + 97);
        guess = guess + asciiChar;
    }
    numTries++;
}
System.out.println(name + " found after " + numTries + " tries!");
```

Bucles y matrices

Uno de los usos más comunes de los bucles es el de trabajar con juegos de datos.

Todos los tipos de bucles resultan útiles:

- Bucles `while` (para comprobar un valor concreto)
- Bucles `for` (para pasar por toda la matriz)
- Bucles `for` mejorados

Bucle for con matrices

ages (matriz de tipos int)



El índice empieza en 0.

El último índice de la matriz es `ages.length - 1`.

`ages[i]` accede a valores de matriz conforme `i` pasa de 0 a `ages.length - 1`.

```
for (int i = 0; i < ages.length; i++ ) {  
    System.out.println("Age is " + ages[i] );  
}
```

Definición de valores en una matriz

ages (matriz de tipos int)



El bucle accede a cada elemento de la matriz por turnos.

```
for (int i = 0; i < ages.length; i++ ) {  
    ages[i] = 10;  
}
```

Cada elemento de la matriz está definido en 10.

Bucle for mejorado con matrices

ages (matriz de tipos int)



El bucle accede a cada elemento de la matriz por turnos.

Cada iteración devuelve el siguiente elemento de la matriz en age.

```
for (int age : ages ) {  
    System.out.println("Age is " + age );  
}
```

Bucle for mejorado con ArrayLists

names (ArrayList de tipos String)

George Jill Xinyi ... Ravi

El bucle accede a cada elemento de la ArrayList por turnos.

Cada iteración devuelve el siguiente elemento de la ArrayList en `name`.

```
for (String name : names ) {  
    System.out.println("Name is " + name);  
}
```

Uso de break con bucles

Ejemplo de break:

```
int passmark = 12;
boolean passed = false;
int[] score = { 4, 6, 2, 8, 12, 34, 9 };
for (int unitScore : score ) {
    if ( unitScore > passmark ) {
        passed = true;
        break;
    }
}
System.out.println("One or more units passed? " + passed);
```

No es necesario volver a pasar por el bucle, por lo tanto utilizar break.

Salida:

```
One or more units passed? true
```

Uso de continue con bucles

Ejemplo de continue:

```
int passMark = 15;
int passesReqd = 3;
int[] score = { 4, 6, 2, 8, 12, 34, 9 };
for (int unitScore : score ) {
    if (score[i] < passMark) {
        continue;
    }
    passesReqd--;
    // Other processing
}
System.out.println("Units still reqd " + Math.max(0,passesReqd));
```

Si la unidad falla, seguir comprobando la siguiente unidad.

Codificación de un bucle `do/while`

Sintaxis:

```
do {  
  
    code_block;  
  
}  
while (boolean_expression); // Semicolon is mandatory.
```

Codificación de un bucle do/while

```
setFloor() {  
    // Normally you would pass the desiredFloor as an argument to the  
    // setFloor method. However, because you have not learned how to  
    // do this yet, desiredFloor is set to a specific number (5)  
    // below.  
    int desiredFloor = 5;  
  
    do {  
        if (currentFloor < desiredFloor) {  
            goUp();  
        }  
        else if (currentFloor > desiredFloor) {  
            goDown();  
        }  
    }  
    while (currentFloor != desiredFloor);  
}
```

Comparación de construcciones de bucle

- Utilice el bucle `while` para iterar indefinidamente con las sentencias y para ejecutar las sentencias cero o más veces.
- Utilice el bucle `do/while` para iterar indefinidamente con las sentencias y para ejecutar las sentencias *una* o más veces.
- Utilice el bucle `for` para pasar por las sentencias un número predefinido de veces.