6. Sentencias repetitivas o bucles

Objetivos:

a) Describir el funcionamiento de las sentencias iterativas o bucles (for, while y do-while)
b) Interpretar el resultado de una secuencia simple o combinada de sentencias de control
c) Codificar una tarea sencilla convenientemente especificada, utilizando la secuencia y combinación de sentencias iterativas y condicionales

Los bucles, iteraciones o sentencias repetitivas modifican el flujo secuencial de un programa permitiendo la ejecución reiterada de una sentencia o sentencias. En Java hay tres tipos diferentes de bucles: `for`, `while` y `do-while`.

6.1. Sentencia for

Es un bucle o sentencia repetitiva que

i) ejecuta la sentencia de inicio
ii) verifica la expresión booleana de término.
   a. si es cierta, ejecuta la sentencia entre llaves y la sentencia de iteración para volver a verificar la expresión booleana de término
   b. si es falsa, sale del bucle.

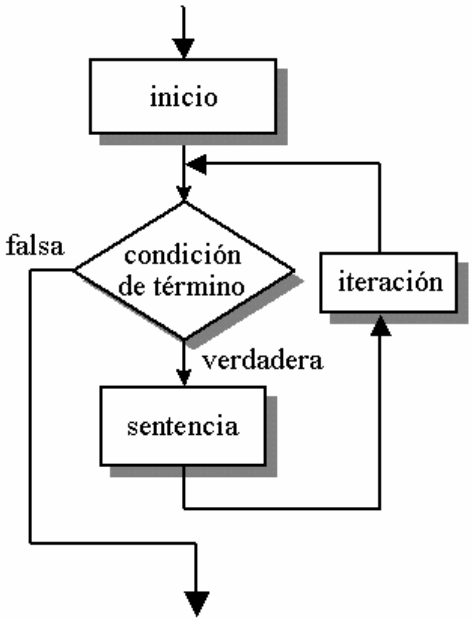

Figura 6.1. Flujograma de la sentencia for

Sintaxis:

```
for (inicio; termino; iteracion)
    sentencia;
```

o si se desean repetir varias sentencias:

```
for (inicio; termino; iteracion) {
    sentencia_1;
    sentencia_2;
    sentencia_n;
    }
```

Las llaves sólo son necesarias si se quieren repetir varias sentencias, aunque se recomienda su uso porque facilita la lectura del código fuente y ayuda a evitar errores al modificarlo. Habitualmente, en la expresión lógica de término se verifica que la variable de control alcance un determinado valor. Por ejemplo:

```
for (i = valor_inicial; i <= valor_final; i++) {
    sentencia;
}
```

Es completamente legal en Java declarar una variable dentro de la cabecera de un bucle `for`. De esta forma la variable (local) sólo tiene ámbito dentro del bucle. Ejemplo sencillo:

```
   System.out.println("Tabla de multiplicar del 5");
   for (int i =0 ; i <= 10; i++) {
       System.out.println(5 + " * " + i + "     = " + 5*i );
   }
```

Salida por pantalla al ejecutar el código anterior:

```
5 * 0 = 0
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50
```

A continuación se muestra un ejemplo completo de un programa que visualiza la tabla de multiplicar del valor numérico entero introducido como parámetro de la línea de ejecución:

```
/**
 *  TablaProducto: Ejemplo de sentencia for
 *  Visualiza la tabla de multiplicar del valor introducido
 *  como parametro en la linea de comandos
 *  A. Garcia-Beltran, 16 de marzo de 2004
 */
public class TablaProducto {
   public static void main (String [] args) {
      int valor;
      valor = Integer.parseInt(args[0]);
      System.out.println("Tabla de multiplicar del numero " + valor);
      for (int i=1; i<=10; i++) {
         System.out.println(valor + " * " + i + " = " + valor*i );
      }
   }
```

```
}
```

Salida por pantalla al ejecutar: `$>java TablaProducto 4`↵

```
Tabla de multiplicar del numero 4
4 * 1 = 4
4 * 2 = 8
4 * 3 = 12
4 * 4 = 16
4 * 5 = 20
4 * 6 = 24
4 * 7 = 28
4 * 8 = 32
4 * 9 = 36
4 * 10 = 40
```

Puede incluirse un bucle `for` dentro de otro (bucles `for` anidados), por ejemplo:

```
   System.out.println("Tablas de multiplicar del 1, 2 y 3");
   for (int i=1; i<=3; i++) {
      System.out.println("Tabla de multiplicar del " + i);
      for (int j=0; j<=10; j++) {
          System.out.println(j + " * " + i + "  = " + j*i );
      }
   }
```

O como en el siguiente programa completo:

```
/**
 *  Doblefor: Ejemplo de sentencias for anidadas
 *  Visualiza por pantalla una cadena que almacena
 *  valores numericos enteros
 *  A. Garcia-Beltran, 16 de marzo de 2004
 */

public class Doblefor {
   public static void main (String [] args ) {
   String s="";
   for (int i=0 ; i<10; i++) {
       s=s+i+" ";
       for (int j=0 ; j<10; j++) {
           s=s+" "+i+j;
           }
       s=s+"\n";
       }
   System.out.println(s);
   }
}
```

Salida por pantalla al ejecutar: `$>java DobleFor`↵

```
0  00 01 02 03 04 05 06 07 08 09
1  10 11 12 13 14 15 16 17 18 19
2  20 21 22 23 24 25 26 27 28 29
3  30 31 32 33 34 35 36 37 38 39
4  40 41 42 43 44 45 46 47 48 49
5  50 51 52 53 54 55 56 57 58 59
6  60 61 62 63 64 65 66 67 68 69
7  70 71 72 73 74 75 76 77 78 79
8  80 81 82 83 84 85 86 87 88 89
9  90 91 92 93 94 95 96 97 98 99
```

Por otro lado, tanto la sentencia de inicio como la de iteración pueden componerse de varias sentencias separadas por comas. Por ejemplo:

```
   for (int i = 0, j = 10 ; i <= 10; i++, j--) {
       // . . .
       }
```

la anterior sentencia es equivalente a:

```
   int j = 10;
   for (int i = 0; i <= 10; i++) {
       // . . .
       j--;
       }
```

6.2. Sentencia while

Es un bucle o sentencia repetitiva con una condicion al principio. Se ejecuta una sentencia mientras sea cierta una condición. La sentencia puede que no se ejecute ni una sola vez.

Sintaxis:

```
[inicializacion;]
while (expresionLogica) {
     sentencias;
     [iteracion;]
}
```

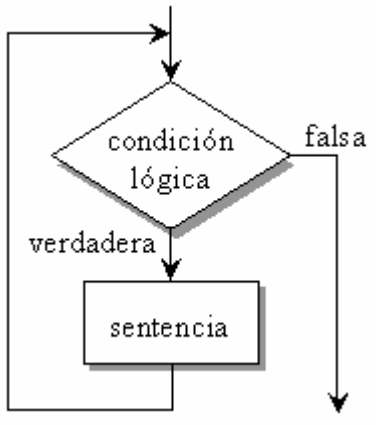

Figura 6.2. Flujograma de la sentencia while

Ejemplo de programa:

```
/**
 *  Ejemplo de sentencia while
 *  Calcula cuantos años deben pasar para duplicar una cantidad
 *  invertida a un determinado interes anual constante
 *  A. Garcia-Beltran - marzo, 2004
 */
public class Duplica {
   public static void main (String [] args) {
     double cantidadInicial=1;
```

```
     double cantidad=cantidadInicial;
     double interes=4;
     int anhos=0;
     while (cantidad < 2*cantidadInicial) {
       anhos++;
       cantidad += cantidad*interes/100;
     }
     System.out.println("La cantidad inicial es = " + cantidadInicial);
     System.out.println("El interes es = " + interes);
     System.out.println("La cantidad final es = " + cantidad);
     System.out.println("El numero de años es = " + anhos);
   }
}
```

Ejemplo de ejecución y salida correspondiente por pantalla:

```
$>java Duplica↵

La cantidad inicial es = 1.0
El interes es = 4.0
La cantidad final es = 2.025816515378531
El numero de años es = 18.0
```

Por convención: El carácter de llave de apertura `{` se coloca al final de la misma línea de la sentencia `while`. El carácter de llave de cierre `}` empieza una nueva línea y se alinea con la palabra `while`.

Otro ejemplo de programa que emplea la sentencia while

```
/**
 *  Ejemplo de sentencia while
 *  Visualiza los argumentos de la linea de comandos
 *  A. Garcia-Beltran - marzo, 2004
 */
public class Eco {
  public static void main (String [] args ) {
    int i=0;                            // Inicializa la variable de control
    while (i < args.length) {           // Mientras quede algun argumento
      System.out.print(args[i] + " ");  // Visualiza el argumento i-esimo
      i++;                              // Incrementa la variable de control
    }
  System.out.println();                 // Salto de linea
  }
}
```

Ejemplo de ejecución y salida correspondiente por pantalla:

```
$>java Eco Esto es una prueba↵

Esto es una prueba
```

Se recuerda que el vector `args` contiene todos los parámetros o argumentos indicados en la línea de comandos. El primer elemento de este vector es `args[0]`. El tamaño del vector puede determinarse añadiendo `.length` a su identificador. Como el índice del primer elemento del vector es `0`, si el tamaño del vector es *n*, entonces el último elemento del vector tiene índice *n-1*. En el ejemplo anterior de ejecución del programa `eco`, `args[0]` toma como valor la cadena `"Esto"`, `args[1]` vale `"es"`, `args[2]` vale `"una"` y `args[3]` vale `"prueba"`.

6.3. Sentencia do-while

Es un bucle o sentencia repetitiva con una condicion al final. Se ejecuta una sentencia mientras sea cierta una condición. En este caso, la sentencia se ejecuta al menos una vez.

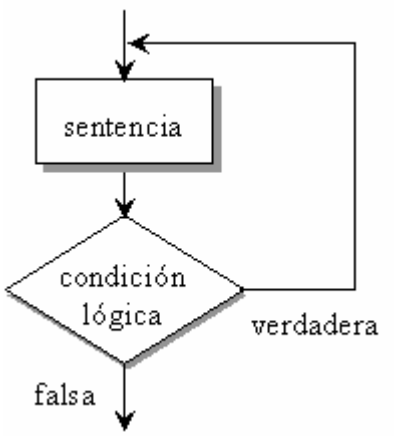

Figura 6.3. Flujograma de la sentencia do/while

Sintaxis:

```
do {
   sentencias;
   [iteracion;]
} while (expresionLogica);
```

Ejemplo de programa:

```
/**
 *  Ejemplo de sentencia do-while
 *  Calcula cuantos años deben pasar para duplicar una cantidad
 *  invertida a un determinado interes anual constante
 *  A. Garcia-Beltran - marzo, 2004
 */
public class Duplica2 {
   public static void main (String [] args) {
     double cantidadInicial=1;
     double cantidad=cantidadInicial;
     double interes=5;
     int anhos=0;
     do
       {
       anhos++;
       cantidad += cantidad*interes/100;
     } while (cantidad < 2*cantidadInicial);
     System.out.println("La cantidad inicial es = " + cantidadInicial);
     System.out.println("El interes es = " + interes);
     System.out.println("La cantidad final es = " + cantidad);
     System.out.println("El numero de años es = " + anhos);
   }
}
```

Ejemplo de ejecución y salida correspondiente por pantalla:

```
$>java Duplica2↵

La cantidad inicial es = 1.0
El interes es = 5.0
La cantidad final es = 2.0789281794113683
El numero de años es = 15.0
```