

9. Bucles

9.1 Introducción

Los bucles se utilizan para repetir un conjunto de sentencias. Por ejemplo, imagina que es necesario introducir la notas de 40 alumnos con el fin de calcular la media, la nota máxima y la nota mínima. Podríamos escribir 40 veces la instrucción que pide un dato por teclado y convertir cada uno de esos datos a un número con decimales con 40 instrucciones `Double.parseDouble()`, no parece algo muy eficiente. Es mucho más práctico meter dentro de un bucle aquellas sentencias que queremos que se repitan.

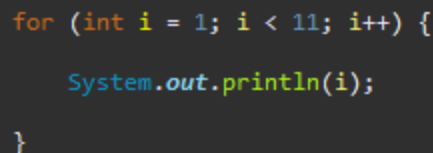
Normalmente existe una condición de salida, que hace que el flujo del programa abandone el bucle y continúe justo en la siguiente sentencia. Si no existe condición de salida o si esta condición no se cumple nunca, se produciría lo que se llama un bucle infinito y el programa no terminaría nunca.

9.2 El bucle for

Se suele utilizar cuando se conoce previamente el número exacto de iteraciones (repeticiones) que se van a realizar. La sintaxis es la siguiente:

```
for (expresion1 ; expresion2 ; expresion3) {  
  
    sentencias  
  
}
```

Justo al principio se ejecuta *expresion1* y normalmente se usa para inicializar una variable. El bucle se repite mientras se cumple *expresion2* y en cada iteración del bucle se ejecuta *expresion3*, que suele ser el incremento o decremento de una variable. Con un ejemplo se verá mucho más claro.



```
for (int i = 1; i < 11; i++) {  
    System.out.println(i);  
}
```

En este ejemplo, `int i = 1` se ejecuta solo una vez, antes que cualquier otra cosa; como ves, esta expresión se utiliza para inicializar la variable *i* a 1. Mientras se cumpla la condición `i < 11` el contenido del bucle, o sea, `System.out.println(i);` se va a ejecutar. En cada iteración del bucle, `i++` hace que la variable *i* se incremente en 1. El resultado del ejemplo es la impresión en pantalla de los números del 1 al 10.

Intenta seguir mentalmente el flujo del programa. Experimenta inicializando la variable *i* con otros valores, cambia la condición con `>` o `<=` y observa lo que sucede. Prueba también a cambiar el incremento de la variable *i*, por ejemplo con `i = i + 2`.

9.3 El bucle while

El bucle while se utiliza para repetir un conjunto de sentencias siempre que se cumpla una determinada condición. Es importante reseñar que la condición se comprueba al comienzo del bucle, por lo que se podría dar el caso de que dicho bucle no se ejecutase nunca. La sintaxis es la siguiente:

```
while (expresion) {  
  
    sentencias  
  
}
```

Las sentencias se ejecutan una y otra vez mientras expresión sea verdadera. El siguiente ejemplo produce la misma salida que el ejemplo anterior, muestra cómo cambian los valores de i del 1 al 10.

```
int i = 1;  
while (i < 11) {  
    System.out.println(i);  
    i++;  
}
```

En el siguiente ejemplo se cuentan y se suman los números que se van introduciendo por teclado. Para indicarle al programa que debe dejar de pedir números, el usuario debe introducir un número negativo; esa será la condición de salida del bucle. Observa que el bucle se repite mientras el número introducido sea mayor o igual que cero.

```

Scanner s = new Scanner(System.in);

System.out.println("Por favor, vaya introduciendo números y pulsando INTRO.");
System.out.println("Para terminar, introduzca un número negativo.");

int numeroIntroducido = 0;
int cuentaNumeros = 0;
int suma = 0;

while (numeroIntroducido >= 0) {
    numeroIntroducido = Integer.parseInt(s.nextLine());
    cuentaNumeros++; // Incrementa en una la variable
    suma += numeroIntroducido; // Equivale a suma = suma + numeroIntroducido
}

System.out.println("Has introducido " + (cuentaNumeros - 1) + " números positivos.");
System.out.println("La suma total de ellos es " + (suma - numeroIntroducido));

```

9.4 El bucle do-while

El bucle *do-while* funciona de la misma manera que el bucle *while*, con la salvedad de que expresión se evalúa al final de la iteración. Las sentencias que encierran el bucle *do-while*, por tanto, se ejecutan como mínimo una vez. **while es más legible, usaremos preferentemente while a do-while.** La sintaxis es la siguiente:

```

do {

    sentencias

} while (expresion)

```

Veamos un ejemplo. En este caso se van a ir leyendo números de teclado mientras el número introducido sea par; el programa parará, por tanto, cuando se introduzca un número impar.

```
Scanner s = new Scanner(System.in)          ;
int numero;

do {
    System.out.print("Dime un número: ");
    numero = Integer.parseInt(s.nextLine());

    if (numero % 2 == 0) {// comprueba si el número introducido es par
        System.out.println("Numero par: " + numero);
    } else {
        System.out.println("Cerrando ejecución por impar.");
    }
} while (numero % 2 == 0);

s.close();
```

Ejercicio 9.1

Muestra los números múltiplos de 5 de 0 a 100 utilizando para ello un bucle for.

Ejercicio 9.2

Muestra los números múltiplos de 5 de 0 a 100 utilizando para ello un bucle while.

Ejercicio 9.3

Muestra los números del 420 al 160, contando de 20 en 20 hacia atrás utilizando un bucle for.

Ejercicio 9.4

Muestra los números del 420 al 160, contando de 20 en 20 hacia atrás utilizando un bucle while.

Ejercicio 9.5

Realiza el control de acceso a una caja fuerte. La combinación será un número de 4 cifras. El programa nos pedirá la combinación para abrirla. Si nos equivocamos, se nos mostrará el mensaje “Lo siento, combinación incorrecta” y si acertamos se nos dirá “La caja fuerte se ha abierto”. Tendremos cuatro oportunidades para abrir la caja fuerte.

Ejercicio 9.6

Muestra la tabla de multiplicar de un número introducido por teclado.

Ejercicio 9.7

Realiza un programa que nos diga cuántos dígitos tiene un número introducido por teclado.

Ejercicio 9.8

Escribe un programa que calcule la media de un conjunto de números positivos introducidos por teclado. El programa no sabe cuántos números se introducirán. El usuario indicará que ha terminado de introducir los datos cuando inserte un número negativo.

Ejercicio 9.9

Escribe un programa que muestre en tres columnas, el cuadrado y el cubo de los 5 primeros números enteros a partir de un número introducido por teclado.

Ejercicio 9.10

Escribe un programa que muestre los n primeros términos de la serie de Fibonacci. El primer término de la serie de Fibonacci es 0, el segundo es 1 y el resto se calcula sumando los dos anteriores, por lo que tendríamos que los términos son 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144... El número n se debe introducir por teclado.

Ejercicio 9.11

Escribe un programa que lea una lista de diez números y determine cuántos son positivos, y cuántos son negativos.

Ejercicio 9.12

Escribe un programa que diga si un número introducido por teclado es o no primo. Un número primo es aquel que sólo es divisible entre él mismo y la unidad.

Ejercicio 9.13

Realiza un programa que pinte una pirámide por pantalla. La altura se debe pedir por teclado. El carácter con el que se pinta la pirámide también se debe pedir por teclado.

Escribe un programa que lea un número n e imprima una pirámide de números con n filas como en la siguiente figura:

```
  1
 121
12321
1234321
```

Ejercicio 9.14

Realiza un programa que pida un número por teclado y que luego muestre ese número al revés.

Ejercicio 9.15

Realiza un programa que pida primero un número y a continuación un dígito. El programa nos debe dar la posición (o posiciones) contando de izquierda a derecha que ocupa ese dígito en el número introducido.

Ejercicio 9.16

Escribe un programa que calcule el factorial de un número entero leído por teclado.

Ejercicio 9.17

Realiza un programa que pinte la letra L por pantalla hecha con asteriscos. El programa pedirá la altura. El palo horizontal de la L tendrá una longitud de la mitad (división entera entre 2) de la altura más uno.

```
Introduzca la altura de la L: 5
*
*
*
*
* * *
```

Ejercicio 9.18

Realiza un programa que pinte la letra U por pantalla hecha con asteriscos. El programa pedirá la altura. Fíjate que el programa inserta un espacio y pinta dos asteriscos menos en la base para simular la curvatura de las esquinas inferiores.

```
Introduzca la altura de la U: 5
*   *
*   *
*   *
*   *
*   *
***
```

Ejercicio 9.19

Realiza un programa que dibuje una X hecha de asteriscos. El programa debe pedir la altura. Se debe comprobar que la altura sea un número impar mayor o igual a 3, en caso contrario se mostrará un mensaje de error.

Por favor, introduzca la altura de la X: 5

```
*   *
* *
*
* *
*   *
```

Ejercicio 9.20

Los números capicúa son aquellos que se leen igual hacia delante y hacia atrás. Implementa un programa que diga si un número introducido por teclado es o no capicúa. Se recomienda usar *long* en lugar de *int* ya que el primero admite números más largos.

Ejercicio 9.21

Escribe un programa que sea capaz de insertar un dígito dentro de un número indicando la posición. El nuevo dígito se colocará en la posición indicada y el resto de dígitos se desplazará hacia la derecha. Las posiciones se cuentan de izquierda a derecha empezando por el 1. Suponemos que el usuario introduce correctamente los datos. Se recomienda usar *long* en lugar de *int* ya que el primero admite números más largos.

```
Por favor, introduzca un número entero positivo: 406783
Introduzca la posición donde quiere insertar: 3
Introduzca el dígito que quiere insertar: 5
El número resultante es 4056783
```