5. Bucles

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 `System.console().readLine()` 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.

5.1 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.

```java
/**
 * Bucle for
 *
 * @author Luis José Sánchez
 */
public class EjemploFor {
  public static void main(String[] args) {
    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. Mientra 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`.

5.2 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 `expresion` sea verdadera. El siguiente ejemplo produce la misma salida que el ejemplo anterior, muestra cómo cambian los valores de `i` del 1 al 10.

```java
/**
 * Bucle while
 *
 * @author Luis José Sánchez
 */
public class EjemploWhile {
  public static void main(String[] args) {
    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.

```java
/**
 * Bucle while que termina cuando se introduce por teclado un
 * número negativo.
 *
 * @author Luis José Sánchez
 */
public class CuentaPositivos {
  public static void main(String[] args) {
    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(System.console().readLine());
      cuentaNumeros++; // Incrementa en uno 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));
  }
}
```

5.3 Bucle `do-while`

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

```
do {

  sentencias

} while (expresion)
```

El siguiente ejemplo es el equivalente `do-while` a los dos ejemplos anteriores que cuentan del 1 al 10.

```
/**
 * Bucle do-while
 *
 * @author Luis José Sánchez
 */
public class EjemploDoWhile {
  public static void main(String[] args) {
    int i = 1;

    do {
      System.out.println(i);
      i++;
    } while (i < 11);
  }
}
```

Veamos otro 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.

```java
/**
 * Bucle do-while que termina cuando se introduce por teclado un
 * número impar.
 *
 * @author Luis José Sánchez
 */
public class TerminaCuandoEsImpar {
  public static void main(String[] args) {
    int numero;

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

      if (numero % 2 == 0) {// comprueba si el número introducido es par
        System.out.println("Qué bonito es el " + numero);
      } else {
        System.out.println("No me gustan los números impares, adiós.");
      }
    } while (numero % 2 == 0);
  }
}
```

Te invito a que realices una modificación sobre este último ejemplo. Después de pedir un número, haz que el programa diga `¿Quiere continuar? (s/n)`. Si el usuario introduce una `s` o una `S`, el programa deberá continuar pidiendo números.

Cualquier bucle que un programador quiera realizar en Java lo puede implementar con `for`, `while` o `do-while`; por tanto, con una sola de estas tres opciones tendría suficiente. No obstante, según el programa en cuestión, una u otra posibilidad se adapta mejor que las otras y resulta más elegante. Con la experiencia, te irás dando cuenta cuándo es mejor utilizar cada una.