Bucles anidados

Un bucle anidado es una estructura en la que un bucle está contenido dentro del cuerpo de otro bucle.

Por ejemplo, imagina que quieres imprimir algo como la siguiente tabla de números, donde en la fila y columnas superiores aparecen las posiciones y dentro de las filas tenemos fila x columna.

1	2	3	4	5	6	7	8	9
2	4	6	8	10	12	14	16	18
3	6	9	12	15	18	21	24	27
4	8	12	16	20	24	28	32	36

Para producir esta tabla de multiplicar, podríamos usar los siguientes bucles *for* anidados:

```
1   1    for (int fila = 1; fila <= 4; fila++) { //para cada una de las 4
filas
2   2        for (int col = 1; col <= 9; col ++) //para cada una de las 9
columnas
3   3            System.out.print(col * fila + "\t"); //muestra la
multiplicación
4   4        System.out.println(); //Empieza una nueva fila
5   5    }
```

Indentamos el código para hacer que sea más legible. En este ejemplo, el bucle externo controla el número de filas en la tabla, es decir, nuestra elección de fila como su contador de bucle.

La instrucción println() (línea 4) se ejecuta después de que el bucle interno haya terminado de iterar, lo que nos permite imprimir una nueva fila en cada iteración del bucle externo.

El bucle interno imprime los nueve valores en cada fila imprimiendo la expresión *col * fila*. Obviamente, el valor de esta expresión depende de ambas variables de bucle.

Analicemos un poco el ejemplo anterior:

1. ¿Cuántas veces se ejecuta la instrucción for en la línea 2? El bucle interno se ejecuta una vez por cada iteración del bucle externo. Por lo tanto, se ejecuta cuatro veces, que es el mismo número de veces que se ejecuta la línea 4.
2. ¿Cuántas veces se ejecuta la declaración de la línea 3? El cuerpo del bucle interno se ejecuta 36 veces, 9 veces por cada ejecución de la línea 2.

Patrones de FOR anidado

A veces es útil usar la variable del bucle externo como límite para el bucle interno. Por ejemplo, veamos el siguiente patrón:

```
#####
####
###
##
#
```

El número de símbolos # en cada fila varía inversamente con el número de fila. En la fila 1, tenemos cinco símbolos; en la fila 2 tenemos cuatro; y así sucesivamente hasta la fila 5, donde tenemos un #.

Para producir este tipo de patrón bidimensional, necesitamos **dos contadores**:

- uno para contar el número de fila y
- otro para contar el número de símbolos # en cada fila.

> Debido a que tenemos que imprimir los símbolos de cada fila antes de pasar a la siguiente fila, el ciclo externo contará los números de fila y el ciclo interno contará los símbolos en cada fila.

La siguiente tabla muestra la relación que queremos:

Fila	Límite (6-i)	Núm. símbolos
1	6-1	5
2	6-2	4
3	6-3	3
4	6-4	2
5	6-5	1

Si dejamos que j sea el contador del bucle interno, entonces j estará limitado por la expresión *6 - i*. Esto conduce a la siguiente estructura de bucle anidado:

```
1       for (int i = 1; i <= 5 ; i++) {
```

```
        for (int j = 1; j <= (6 - i); j++) {
            System.out.print('#');
        }
        System.out.println();
    }
```

Otra solución si no queremos usar un literal en la condición del bucle interno sería:

```
    for (int i = 1; i <= 5 ; i++) {
        for (int j = 5; j >= i; j--) {
            System.out.print('#');
        }
        System.out.println();
    }
```

A menudo los literales que aparecen como límites en los bucles for se denominan números mágicos y pueden crear problemas de redundancia en el código o código no legible. Para solucionar esto utilizamos constantes:

```
    final int MAX_WIDTH = 5;

    for (int i = 1; i <= MAX_WIDTH ; i++) {
        for (int j = MAX_WIDTH; j >= i; j--) {
            System.out.print('#');
        }
        System.out.println();
    }
```