4. Arrays multidimensionales

Un array multidimensional es, en la práctica, un array en el que cada posición está ocupada por otro array. El número de veces que repetimos dicho proceso marca las dimensiones de la estructura.

0	1	2	3	4
Dato	Dato	Dato	Dato	Dato

Array unidimensional

0			1			2			3		
0,0	0,1	0,2	1,0	1,1	1,2	2,0	2,1	2,2	3,0	3,1	3,2
Dato	Dato	Dato	Dato	Dato	Dato	Dato	Dato	Dato	Dato	Dato	Dato

Array multidimensional (en este caso bidimensional)

Figura 4.1. Array unidimensional y array multidimensional

Por comodidad, los arrays bidimensionales, llamados generalmente matrices, se representan gráficamente como si estuviesen formados por filas y columnas. Cada fila y cada columna tienen asociado un número (empezando por 0, como en cualquier array), y la referencia de una posición concreta viene dada por la fila y por la columna en la que se encuentra.

	Columna 0	**Columna 1**	**Columna 2**	**Columna 3**
Fila 0	Posición 0,0	Posición 0,1	Posición 0,2	Posición 0,3
Fila 1	Posición 1,0	Posición 1,1	Posición 1,2	Posición 1,3
Fila 2	Posición 2,0	Posición 2,1	Posición 2,2	Posición 2,3

Tabla 4.1. Array de dos dimensiones con 3 filas y 4 columnas

IMPORTANTE

En lo sucesivo, para evitar confusiones, hablaremos de array para referirnos a un array unidimensional, y de matriz para referirnos a un array de más de una dimensión —en general de dos—. En las matrices de dos dimensiones, la primera siempre hace referencia a las filas y la segunda a las columnas. En el caso de la figura, estamos pues ante una matriz 3 x 4.

4.1. Matrices en Java

Al igual que ocurre con los arrays unidimensionales, para declarar una matriz en Java necesitamos especificar cuántas dimensiones tiene y la longitud de cada una de ellas, así como darle un nombre a la estructura y concretar de qué tipo es la información que contiene.

Suponiendo que la matriz de la figura anterior (recordemos, de tamaño 3 x 4) contuviese elementos de tipo entero, se declararía de la siguiente forma:

```
int[][] matriz = new int[3][4];
```

Si queremos realizar la declaración y la asignación de valores al mismo tiempo, lo haremos de una manera muy similar a como vimos para los arrays unidimensionales, aunque debemos separar con llaves y con una coma cada nueva fila que añadamos a la matriz. Aquí tenéis nuestra matriz de tres filas y cuatro columnas, declarada y rellenada con una única instrucción.

```
int [][] matriz = {{8, 3, 10, 6}, {10, 4, 11, 12}, {13, 17, 4, 0}};
```

La primera y la última llave envuelven toda la matriz, mientras que las llaves interiores envuelven cada fila. Entre fila y fila, ponemos una coma.

Al igual que ocurría con los arrays unidimensionales, si nos encontramos con una matriz de la que no sabemos su tamaño, lo podemos averiguar con la propiedad **length**. En este caso, sin embargo, al haber dos dimensiones necesitamos averiguar, por un lado, las filas y, por otro, las columnas.

```
int filas = matriz.length;
// nos devuelve la cantidad de filas que tiene la matriz
int columnas = matriz[0].length;
// devuelve la cantidad de columnas de la primera fila
//que será la misma que en el resto de filas
```

IMPORTANTE

Recuerda que las filas y las columnas de una matriz también empiezan por el número 0. Así pues, en una matriz de 3 filas y 4 columnas, la primera fila es la 0 y la última es la 2 (¡la 3 no existe!), mientras que la primera columna también es la 0 y la última sería la 3.

Lógicamente, para recorrer una matriz no es suficiente con un bucle. Puesto que cada posición viene definida por su fila y su columna, necesitamos un índice para las filas y otro para las columnas. El orden de recorrido puede ser cualquiera, pero en general se recorren primero las filas empezando por la 0, y dentro de una misma fila se hace lo propio con las columnas, también empezando desde 0.

El siguiente código rellenará el contenido de cualquier matriz de dos dimensiones, sea cual sea su tamaño, pidiendo los datos desde la consola:

```
Scanner teclado = new Scanner(System.in);
int[][] matriz = new int[3][3];
for(int fila = 0; fila < matriz.length; fila++) {
    for(int columna = 0; columna < matriz[fila].length; columna++) {
        System.out.print("Introduzca el contenido de la posición [" + fila +
"," + columna + "]: ");
        matriz[fila][columna] = teclado.nextInt();
    }
}
```

Por su parte, el siguiente código mostraría su contenido:

```
System.out.println("==== CONTENIDO DE LA MATRIZ ====");
for(int fila = 0; fila < matriz.length; fila++) {
   for(int columna = 0; columna < matriz[fila].length; columna++) {
      System.out.print("Contenido de la posición [" + fila + "," + columna +
"]: ");
      System.out.println(matriz[fila][columna]);
   }
}
```

```
==== CONTENIDO DE LA MATRIZ ====
Contenido de la posición[0,0]:1
Contenido de la posición[0,1]:2
Contenido de la posición[0,2]:3
Contenido de la posición[1,0]:4
Contenido de la posición[1,1]:5
Contenido de la posición[1,2]:6
Contenido de la posición[2,0]:7
Contenido de la posición[2,1]:8
Contenido de la posición[2,2]:9
```

Figura 4.2. Mostrando el contenido de una matriz,

Si queremos mostrar los datos en forma de matriz, con sus filas y columnas, tan solo tenemos que pedir al sistema que muestre por pantalla los datos de una fila y añadir un salto de línea cuando vayamos a pasar a la fila siguiente. Fíjate en que usaremos **\n** para representar un salto de línea y **\t** para el tabulador.

```
System.out.println("==== CONTENIDO DE LA MATRIZ ====");
for(int fila = 0; fila < matriz.length; fila++) {
   for(int columna = 0; columna < matriz[fila].length; columna++) {
      System.out.print("[" + matriz[fila][columna] + "]");
   }
   System.out.print("\n"); // salto de línea entre fila y fila
}
```

El resultado por pantalla sería ahora sería el siguiente:

```
==== CONTENIDO DE LA MATRIZ ====
[1] [2] [3]
[4] [5] [6]
[7] [8] [9]
```

Figura 4.3. Mostrando el contenido de una matriz en el formato de filas y columnas.

IMPORTANTE

En el caso de los arrays multidimensionales de más de 2 dimensiones, aunque estos parezcan complicados de entender, lo único que habría que hacer es añadir un bucle más por dimensión. Disponemos de algún ejemplo en el material adicional.

Existe una clase específica de Java para trabajar con arrays de cualquier tipo, lo que facilita diversas tareas, como ordenarlos, copiar un array en otro, etcétera. Podemos ver el funcionamiento de la **clase Arrays** en el material complementario.