ESTRUCTURA DE ALMACENAMIENTO - ARRAYS BIDIMENSIONALES

Programación

INDICE

ARRAYS BIDIMENSIONALES

Un array en Java puede tener más de una dimensión. El caso más general son los arrays bidimensionales también llamados **matrices** o **tablas**.

La dimensión de un array la determina el número de índices necesarios para acceder a sus elementos.

Los arrays unidimensionales solo utilizan un índice para acceder a cada elemento. Una matriz necesita dos índices para acceder a sus elementos.

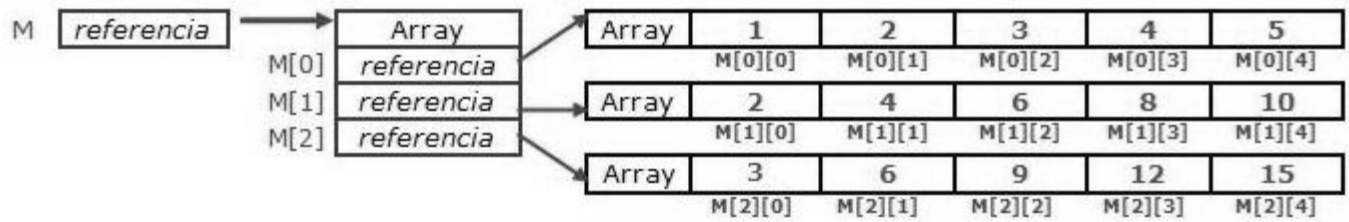
Gráficamente podemos representar una matriz como una tabla de n filas y m columnas cuyos elementos son todos del mismo tipo.

La siguiente figura representa un array (M) de 3 filas y 5 columnas:

	0	1	2	3	4
0	1	2	3	4	5
1	2	4	6	8	10
2	3	6	9	12	15

Pero en realidad **una matriz en Java es un array de arrays**.

Gráficamente podemos representar la disposición real en memoria del array anterior así:



La longitud del array M (M.length) es 3.

La longitud de cada fila del array (M[i].length) es 5.

Para acceder a cada elemento de la matriz se utilizan dos índices. El primero indica la fila y el segundo la columna.

CREAR MATRICES

Se crean de forma similar a los arrays unidimensionales, añadiendo un índice.

Por ejemplo:

matriz de datos de tipo int llamado ventas de 4 filas y 6 columnas:

```
int [][] ventas = new int[4][6];
```

matriz de datos double llamado temperaturas de 3 filas y 4 columnas:

```
double [][] temperaturas = new double[3][4];
```

MATRICES IRREGULARES

En Java se pueden crear **arrays *irregulares*** en los que el número de elementos de cada fila es variable. Solo es obligatorio indicar el número de filas.

Por ejemplo:

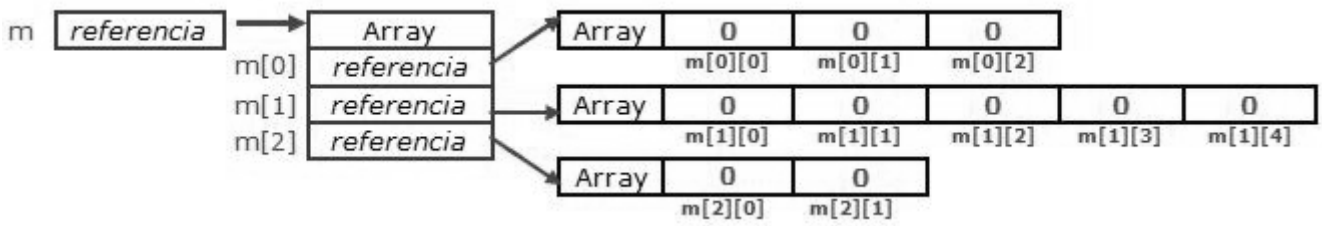
```
int [][] m = new int[3][];
```

crea una matriz m de 3 filas.

A cada fila se le puede asignar un número distinto de columnas:

```
m[0] = new int[3];
m[1] = new int[5];
m[2] = new int[2];
```

Gráficamente podemos representar la disposición real en memoria del array anterior así:



INICIALIZAR MATRICES

Un array es un objeto, por lo tanto, cuando se crea, a sus elementos se les asigna automáticamente un valor inicial:

- 0 para arrays numéricos
- '\u0000' (carácter nulo) para arrays de caracteres
- false para arrays booleanos
- null para arrays de String y de referencias a objetos.

También podemos dar otros valores iniciales al array cuando se crea.

Los valores iniciales se escriben entre llaves separados por comas.

Los valores que se le asignen a cada fila aparecerán a su vez entre llaves separados por comas.

El número de valores determina el tamaño de la matriz.

Por ejemplo:

```
int [][] numeros = {{6,7,5}, {3, 8, 4}, {1,0,2}, {9,5,2}};
```

se crea la matriz numeros de tipo int, de 4 filas y 3 columnas, y se le asignan esos valores iniciales.

Este array tiene 4 elementos. Cada uno de estos 4 elementos es otro array de 3 elementos.

Para acceder a los elementos de los arrays:

```
int a = numeros[0][0]; // a es 6

int b = numeros[1][2]; // b es 4

int c = numeros[2][2]; // c es 2
```

Asignando valores iniciales se pueden crear también matrices irregulares.

```
int [][] a = {{6,7,5,0,4}, {3, 8, 4}, {1,0,2,7}, {9,5}};
```

Crea una matriz irregular de 4 filas. La primera de 5 columnas, la segunda de 3, la tercera de 4 y la cuarta de 2.

RECORRER MATRICES

Para recorrer una matriz se anidan dos bucles for. En general para recorrer un array multidimensional se anidan tantas instrucciones for como dimensiones tenga el array.

Ejemplo de recorrido de una matriz en Java:

Programa que lee por teclado números enteros y los guarda en una matriz de 5 filas y 4 columnas. A continuación muestra los valores leídos, el mayor y el menor y las posiciones que ocupan.

```
public class Bidimensional2 {

    public static void main(String[] args) {

        final int FILAS = 5, COLUMNAS = 4;

        Scanner sc = new Scanner(System.in);

        int i, j, mayor, menor;

        int filaMayor, filaMenor, colMayor, colMenor;

        int[][] A = new int[FILAS][COLUMNAS];

      System.out.println("Lectura de elementos de la matriz:");

        for (i = 0; i < FILAS; i++) {

            for (j = 0; j < COLUMNAS; j++) {

                System.out.print("A[" + i + "][" + j + "]= ");

                A[i][j] = sc.nextInt();

            }

        }

        System.out.println("valores introducidos:");

        for (i = 0; i < A.length; i++) {

            for (j = 0; j < A[i].length; j++) {

                System.out.print(A[i][j] + " ");

            }

            System.out.println();

        }

        mayor = menor = A[0][0];

      //se toma el primero como mayor y menor

        filaMayor = filaMenor = colMayor = colMenor = 0;
```

```
        for (i = 0; i < A.length; i++) {  //
            for (j = 0; j < A[i].length; j++) {
                if (A[i][j] > mayor) {
                    mayor = A[i][j];
                    filaMayor = i;
                    colMayor = j;
                } else if (A[i][j] < menor) {
                    menor = A[i][j];
                    filaMenor = i;
                    colMenor = j;
                }
            }
        }
        System.out.print("Elemento mayor: " + mayor);
        System.out.println(" Fila: "+ filaMayor + " Columna: "
+ colMayor);
        System.out.print("Elemento menor: " + menor);
        System.out.println(" Fila: "+ filaMenor + " Columna: "
+ colMenor);
    }
}
```

En este ejemplo se han utilizado dos formas distintas para recorrer la matriz:

- utilizando en el for el número de filas y columnas
- utilizando en el for el atributo length

Para recorrer arrays irregulares como el siguiente:

int [][] a = {{6,7,5,0,4}, {3, 8, 4}, {1,0,2,7}, {9,5}};

Usaremos siempre length para obtener el número de columnas que tiene cada fila:

```
for (i = 0; i < a.length; i++) {  //número de filas
    for (j = 0; j < a[i].length; j++) { //número de columnas de cada fila
        System.out.print(a[i][j] + " ");
    }
    System.out.println();
}
```