Capítulo 8
Matrices multidimensionales

Motivaciones

Hasta ahora, ha utilizado matrices unidimensionales para modelar colecciones lineales de elementos. Puede usar una matriz bidimensional para representar una matriz o una tabla. Por ejemplo, la siguiente tabla que describe las distancias entre las ciudades se puede representar utilizando una matriz bidimensional.

Distance Table (in miles)

	Chicago	Boston	New York	Atlanta	Miami	Dallas	Houston
Chicago	0	983	787	714	1375	967	1087
Boston	983	0	214	1102	1763	1723	1842
New York	787	214	0	888	1549	1548	1627
Atlanta	714	1102	888	0	661	781	810
Miami	1375	1763	1549	661	0	1426	1187
Dallas	967	1723	1548	781	1426	0	239
Houston	1087	1842	1627	810	1187	239	0

```
double[][] distances = {
  {0, 983, 787, 714, 1375, 967, 1087},
  {983, 0, 214, 1102, 1763, 1723, 1842},
  {787, 214, 0, 888, 1549, 1548, 1627},
  {714, 1102, 888, 0, 661, 781, 810},
  {1375, 1763, 1549, 661, 0, 1426, 1187},
  {967, 1723, 1548, 781, 1426, 0, 239},
  {1087, 1842, 1627, 810, 1187, 239, 0},
};
```

Objetivos

- Utilizar ejemplos de representación de datos utilizando matrices bidimensionales.
- Declarar variables para matrices bidimensionales, crear matrices y acceder a elementos de la matriz en una matriz bidimensional usando índices de filas y columnas.
- Programar operaciones comunes para matrices bidimensionales (visualización de matrices, suma de todos los elementos, búsqueda de los elementos mínimo y máximo, y barajado aleatorio).
- Pasar matrices bidimensionales a los métodos.
- Calificar preguntas de opción múltiple utilizando matrices bidimensionales.
- Resolver el problema del par más cercano utilizando matrices bidimensionales.
- Verificar una solución de Sudoku usando arreglos bidimensionales.
- Usar matrices multidimensionales.

Declarar/crear matrices bidimensionales

En Java, una matriz bidimensional se implementa mediante una tabla de tablas, es decir, una tabla cuyos elementos son a su vez, tablas.

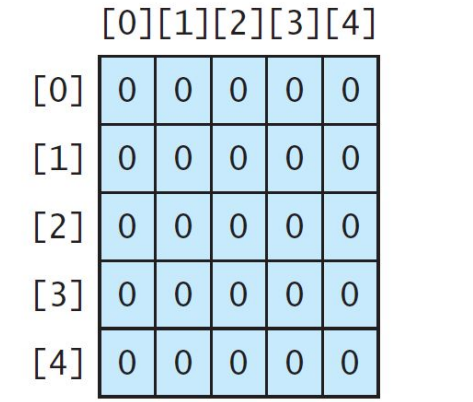
```
// Declara el array refvar
dataType[][] refVar;

// Crea el array y se lo assigna a refVar
refVar = new dataType[10][10];

// Combina declaración y creación en una sentencia
dataType[][] refVar = new dataType[10][10];

// Sintaxis alternativa
dataType refVar[][] = new dataType[10][10];
```

Declarar/crear matrices bidimensionales

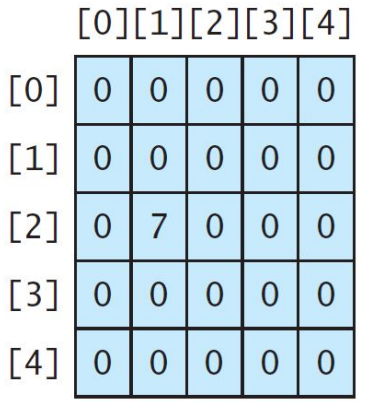
Para recorrer una tabla de dos dimensiones podemos usar bucles anidados, de tal forma que el de fuera recorra las filas con un índice y el de dentro, las columnas.

```
int[][] matrix = new int[10][10];
  o
int matrix[][] = new int[10][10];
matrix[0][0] = 3;

for (int i = 0; i < matrix.length; i++)
  for (int j = 0; j < matrix[i].length; j++)
    matrix[i][j] = (int)(Math.random() * 1000);

double[][] x;
```

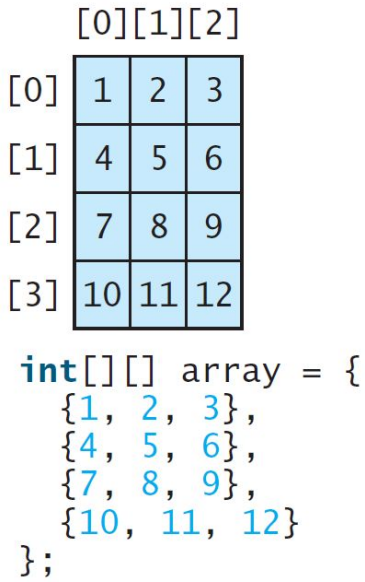
Ilustración de matriz bidimensional



(a)



(b)



(c)

matrix.length? 5

matrix[0].length? 5

array.length? 4

array[0].length? 3

Declarar, crear e inicializar

También podemos usar un inicializador de matriz para declarar, crear e inicializar una matriz bidimensional. Por ejemplo,

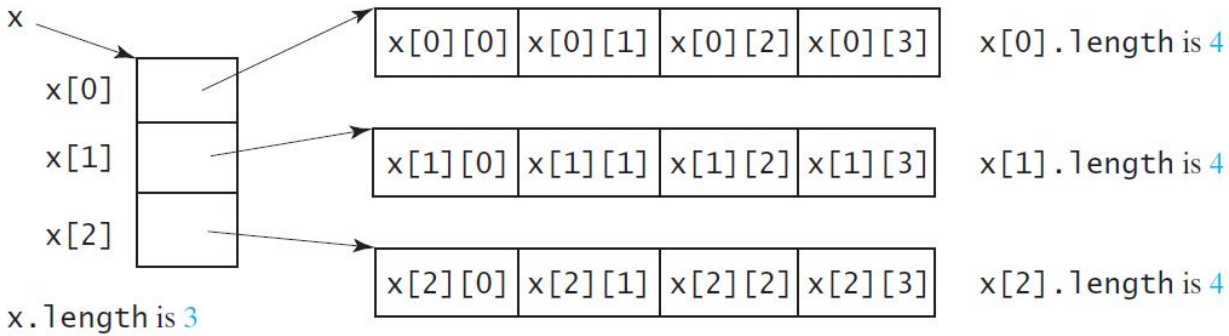
```
int[][] array = {
  {1, 2, 3},
  {4, 5, 6},
  {7, 8, 9},
  {10, 11, 12}
};
```

Igual a

```
int[][] array = new int[4][3];
array[0][0] = 1; array[0][1] = 2; array[0][2] = 3;
array[1][0] = 4; array[1][1] = 5; array[1][2] = 6;
array[2][0] = 7; array[2][1] = 8; array[2][2] = 9;
array[3][0] = 10; array[3][1] = 11; array[3][2] = 12;
```

Longitudes de matrices bidimensionales

int[][] x = new int[3][4];




```
int[][] array = {
  {1, 2, 3},
  {4, 5, 6},
  {7, 8, 9},
  {10, 11, 12}
};
```

array[4].length ArrayIndexOutOfBoundsException

array.length

array[0].length

array[1].length

array[2].length

array[3].length

Arrays Irregulares

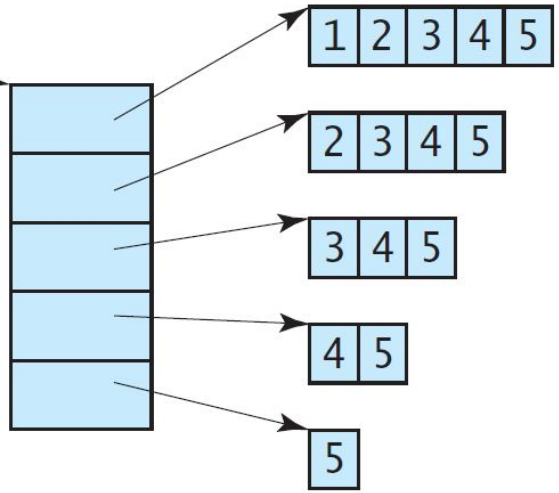
Cada fila en una matriz bidimensional es en sí misma una matriz. Entonces, las filas pueden tener diferentes longitudes. Tal matriz se conoce como una matriz desigual. Por ejemplo,

```
int[][] matrix = {
        {1, 2, 3, 4, 5},
        {2, 3, 4, 5},
        {3, 4, 5},
        {4, 5},
        {5}
        };
```

```
matrix.length is 5
matrix[0].length is 5
matrix[1].length is 4
matrix[2].length is 3
matrix[3].length is 2
matrix[4].length is 1
```

Arrays Irregulares, cont.

```
int[][] triangleArray = {
  {1, 2, 3, 4, 5},
  {2, 3, 4, 5},
  {3, 4, 5},
  {4, 5},
  {5}
};
```



Procesamiento de matrices bidimensionales

1. Inicializando matrices con valores de entrada
2. Impresión de matrices
3. Sumando todos los elementos
4. Sumando todos los elementos por columna
5. Encontrar el índice más pequeño del elemento más grande
6. Mezcla aleatoria

Inicializando matrices con valores de entrada

```
java.util.Scanner input = new Scanner(System.in);
System.out.println("Enter " + matrix.length + " rows and " +    matrix[0].length + " columns: ");
for (int row = 0; row < matrix.length; row++) {
    for (int column = 0; column < matrix[row].length; column++) {
        matrix[row][column] = input.nextInt();
    }
}
```

Inicializando matrices con valores aleatorios

```
for (int row = 0; row < matrix.length; row++) {
    for (int column = 0; column < matrix[row].length; column++) {
        matrix[row][column] = (int)(Math.random() * 100);
    }
}
```

Imprimiendo matrices

```
 for (int row = 0; row < matrix.length; row++) {
     for (int column = 0; column < matrix[row].length; column++) {
         System.out.print(matrix[row][column] + " ");
     }
     System.out.println();
}
```

Sumando todos los elementos

```
int total = 0;
for (int row = 0; row < matrix.length; row++) {
    for (int column = 0; column < matrix[row].length; column++) {
        total += matrix[row][column];
    }
}
```

Sumando todos los elementos por columna

```
 for (int column = 0; column < matrix[0].length; column++) {
     int total = 0;
     for (int row = 0; row < matrix.length; row++)
         total += matrix[row][column];
     System.out.println("Sum for column " + column + " is "  + total);
}
```

Mezcla aleatoria

```
for (int i = 0; i < matrix.length; i++) {
    for (int j = 0; j < matrix[i].length; j++) {
        int i1 = (int)(Math.random() * matrix.length);
        int j1 = (int)(Math.random() * matrix[i].length);
        // Swap matrix[i][j] with matrix[i1][j1]
        int temp = matrix[i][j];
        matrix[i][j] = matrix[i1][j1];
        matrix[i1][j1] = temp;
    }
}
```

Pasar matrices bidimensionales a los Métodos

Podemos pasar una matriz bidimensional a un método tal y como se pasa una matriz unidimensional.

El siguiente programa da un ejemplo con dos métodos.

El primer método, getArray(), devuelve una matriz bidimensional,
y el segundo método, sum(int[][] m), devuelve la suma de todos los elementos en una matriz.

Problemas elementales guiados

1. Crear y mostrar matrices

Crea una matriz bidimensional de números enteros y muestra sus elementos en forma de tabla.

Tareas:

Declara una matriz de 3x3 e inicialízala con valores de tu elección.

Usa bucles for anidados para recorrer la matriz y mostrar los elementos en la consola.

2. Sumar los elementos de una matriz

Calcula la suma de todos los elementos de una matriz bidimensional.

Tareas:

Declara e inicializa una matriz de 4x4.

Usa bucles para recorrer la matriz y sumar sus elementos.

Muestra el resultado en la consola.

3. Transponer una matriz

Transponer una matriz significa intercambiar filas por columnas.

Tareas:

Crea una matriz de 3x2.

Crea una nueva matriz de 2x3 que sea la transpuesta de la original.

Imprime ambas matrices.

Problemas elementales guiados

4. Buscar el elemento máximo y mínimo

Encuentra el elemento más grande y más pequeño en una matriz bidimensional.

Tareas:

Declara una matriz de números enteros (puede ser 5x5).

Usa bucles para recorrer la matriz y encontrar los valores máximo y mínimo.

Muestra ambos valores junto con sus posiciones (índices).

5. Suma de filas y columnas

Calcula la suma de los elementos de cada fila y de cada columna en una matriz.

Tareas:

Declara una matriz de 3x3.

Usa bucles para sumar los elementos de cada fila y cada columna.

Muestra los resultados en la consola.

6. Matriz identidad

Crea una matriz identidad de tamaño n×n, donde todos los elementos de la diagonal principal son 1 y el resto son 0.

Tareas:

Pide al usuario que introduzca el tamaño n de la matriz.

Crea y muestra la matriz identidad resultante.

Problemas elementales guiados

7. Multiplicación de matrices

Descripción: Implementa la multiplicación de dos matrices.

Tareas:

Declara e inicializa dos matrices compatibles para multiplicación (por ejemplo, de 2x3 y 3x2).

Realiza la multiplicación y almacena el resultado en una nueva matriz.

Imprime la matriz resultante.

8. Rotar una matriz

Rota una matriz cuadrada n×n 90° en sentido horario.

Tareas:

Crea una matriz cuadrada de 4x4.

Implementa un algoritmo para rotarla 90°.

Imprime la matriz antes y después de la rotación.

9. Diagonal principal y secundaria

Calcula la suma de los elementos de la diagonal principal y secundaria de una matriz cuadrada.

Tareas:

Declara una matriz de 4x4.

Calcula las sumas de las diagonales principal y secundaria.

Muestra los resultados en la consola.

Caso práctico: Calificación de la prueba de opción múltiple

Respuesta de los estudiantes

	0	1	2	3	4	5	6	7	8	9
Student 0	A	B	A	C	C	D	E	E	A	D
Student 1	D	B	A	B	C	A	E	E	A	D
Student 2	E	D	D	A	C	B	E	E	A	D
Student 3	C	B	A	E	D	C	E	E	A	D
Student 4	A	B	D	C	C	D	E	E	A	D
Student 5	B	B	E	C	C	D	E	E	A	D
Student 6	B	B	A	C	C	D	E	E	A	D
Student 7	E	B	E	C	C	D	E	E	A	D

Key to the Questions:

	0	1	2	3	4	5	6	7	8	9
Key	D	B	D	C	C	D	A	E	A	D

Objetivo: escribir un programa que califica la prueba de opción múltiple. Supongamos que hay ocho estudiantes y diez preguntas, y las respuestas se almacenan en una matriz bidimensional. Cada fila registra las respuestas de un alumno a las preguntas, como se muestra en la siguiente matriz

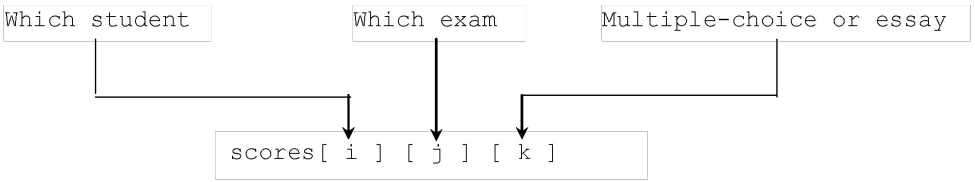
El programa debe calificar la prueba y muestra el resultado. Comparar las respuestas de cada alumno con la clave, contar el número de respuestas correctas y mostralas.

Matrices multidimensionales

Ocasionalmente, deberá representar estructuras de datos n-dimensionales. En Java, puede crear matrices n-dimensionales para cualquier entero n.

La forma de declarar variables de matriz bidimensional y crear matrices bidimensionales se puede generalizar para declarar variables de matriz n-dimensional y crear matrices n-dimensional para n>= 3.

```
double[][][] scores = {
  {{7.5, 20.5}, {9.0, 22.5}, {15, 33.5}, {13, 21.5}, {15, 2.5}},
  {{4.5, 21.5}, {9.0, 22.5}, {15, 34.5}, {12, 20.5}, {14, 9.5}},
  {{6.5, 30.5}, {9.4, 10.5}, {11, 33.5}, {11, 23.5}, {10, 2.5}},
  {{6.5, 23.5}, {9.4, 32.5}, {13, 34.5}, {11, 20.5}, {16, 7.5}},
  {{8.5, 26.5}, {9.4, 52.5}, {13, 36.5}, {13, 24.5}, {16, 2.5}},
  {{9.5, 20.5}, {9.4, 42.5}, {13, 31.5}, {12, 20.5}, {16, 6.5}}
};
```



Problema: Calculando el total de puntos

Objetivo: escribir un programa que calcule la puntuación total de los estudiantes en una clase. Supongamos que las puntuaciones se almacenan en una matriz tridimensional llamada scores. El primer índice en puntajes se refiere a un estudiante, el segundo se refiere a un examen y el tercero se refiere a la parte del examen.

Supongamos que hay 7 estudiantes, 5 exámenes y cada examen tiene dos partes: la parte de opción múltiple y la parte de programación. Así pues, scores[i][j][0] representa el puntaje en la parte de opción múltiple para el alumno de i en el examen de j.

El programa mostrará el puntaje total de cada estudiante.

Problema: información meteorológica

Supongamos que una estación meteorológica registra la temperatura y la humedad en cada hora de cada día y almacena los datos de los últimos diez días en un archivo de texto llamado weather.txt. Cada línea del archivo consta de cuatro números que indican el día, la hora, la temperatura y la humedad.

Escribir un programa que calcule la temperatura y humedad diaria media durante los 10 días.

```
1 1 76.4 0.92
1 2 77.7 0.93
...
10 23 97.7 0.71
10 24 98.7 0.74
```

(a)

```
10 24 98.7 0.74
1 2 77.7 0.93
...
10 23 97.7 0.71
1 1 76.4 0.92
```

(b)