Chapter 8 Multidimensional Arrays

CS1: Java Programming
Colorado State University

Original slides by Daniel Liang
Modified slides by Chris Wilcox
Announcements

● Lab Wednesday
● Review in Lecture Wednesday – Bring Questions
● Exam Friday
● Spring Break next week – get some rest!
Motivations

Thus far, you have used one-dimensional arrays to model linear collections of elements. You can use a two-dimensional array to represent a matrix or a table. For example, the following table that describes the distances between the cities can be represented using a two-dimensional array.

<table>
<thead>
<tr>
<th></th>
<th>Chicago</th>
<th>Boston</th>
<th>New York</th>
<th>Atlanta</th>
<th>Miami</th>
<th>Dallas</th>
<th>Houston</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chicago</td>
<td>0</td>
<td>983</td>
<td>787</td>
<td>714</td>
<td>1375</td>
<td>967</td>
<td>1087</td>
</tr>
<tr>
<td>Boston</td>
<td>983</td>
<td>0</td>
<td>214</td>
<td>1102</td>
<td>1763</td>
<td>1723</td>
<td>1842</td>
</tr>
<tr>
<td>New York</td>
<td>787</td>
<td>214</td>
<td>0</td>
<td>888</td>
<td>1549</td>
<td>1548</td>
<td>1627</td>
</tr>
<tr>
<td>Atlanta</td>
<td>714</td>
<td>1102</td>
<td>888</td>
<td>0</td>
<td>661</td>
<td>781</td>
<td>810</td>
</tr>
<tr>
<td>Miami</td>
<td>1375</td>
<td>1763</td>
<td>1549</td>
<td>661</td>
<td>0</td>
<td>1426</td>
<td>1187</td>
</tr>
<tr>
<td>Dallas</td>
<td>967</td>
<td>1723</td>
<td>1548</td>
<td>781</td>
<td>1426</td>
<td>0</td>
<td>239</td>
</tr>
<tr>
<td>Houston</td>
<td>1087</td>
<td>1842</td>
<td>1627</td>
<td>810</td>
<td>1187</td>
<td>239</td>
<td>0</td>
</tr>
</tbody>
</table>
Other Representations?

What are some other representations of multi-dimensional arrays?
Declare/Create Two-dimensional Arrays

// Declare array ref var
dataType[][] refVar;

// Create array and assign its reference to variable
refVar = new dataType[10][10];

// CombineD
dataType[][] refVar = new dataType[10][10];
Declarating Variables of Two-dimensional Arrays and Creating Two-dimensional Arrays

```java
int[][] matrix = new int[10][10];

matrix[0][0] = 3;
```
Two-dimensional Array Illustration

matrix = new int[5][5];
matrix[2][1] = 7;

int[][] array = {
    {1, 2, 3},
    {4, 5, 6},
    {7, 8, 9},
    {10, 11, 12}
};
Declaring, Creating, and Initializing Using Shorthand Notations

You can also use an array initializer to declare, create and initialize a two-dimensional array. For example,

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

Same as

```java
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;
```
Lengths of Two-dimensional Arrays

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

![Diagram showing the lengths of the nested arrays](image_url)
Lengths of Two-dimensional Arrays, cont.

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

array.length
array[0].length
array[1].length
array[2].length
array[3].length

array[4].length // ArrayIndexOutOfBoundsException
```
Ragged Arrays

Each row in a two-dimensional array is itself an array. So, the rows can have different lengths. Such an array is known as a ragged array. For example,

```java
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
Ragged Arrays, cont.

```java
int[][] triangleArray = {
    {1, 2, 3, 4, 5},
    {2, 3, 4, 5},
    {3, 4, 5},
    {4, 5},
    {5}
};
```
Initializing arrays with random values

```java
for (int row = 0; row < mat.length; row++)
{
    for (int col = 0; col < mat[row].length; col++)
    {
        mat[row][col] = (int)(Math.random() * 100);
    }
}
```
Printing arrays

```java
for (int row = 0; row < mat.length; row++) {
    for (int col = 0; col < mat[row].length; col++) {
        System.out.print(mat[row][col] + " ");
    }
    System.out.println();
}
```
Summing elements by column

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

    System.out.println(column + " is " + total);
}
```