Multidimensional Arrays

Motivations (1 of 2)

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.

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

Motivations (2 of 2)

```
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},
};
```


Objectives (1 of 2)

8.1 To give examples of representing data using two-dimensional arrays (§8.1).

8.2 To declare variables for two-dimensional arrays, create arrays, and access array elements in a two-dimensional array using row and column indexes (§8.2).

8.3 To program common operations for two-dimensional arrays (displaying arrays, summing all elements, finding the minimum and maximum elements, and random shuffling) (§8.3).

8.4 To pass two-dimensional arrays to methods (§8.4).

Declare/Create Two-dimensional Arrays

```
// Declare array ref var
dataType[][] refVar;
// Create array and assign its reference to
variable
refVar = new dataType[10][10];
// Combine declaration and creation in one
statement
dataType[][] refVar = new dataType[10][10];
// Alternative syntax
dataType refVar[][] = new dataType[10][10];
```

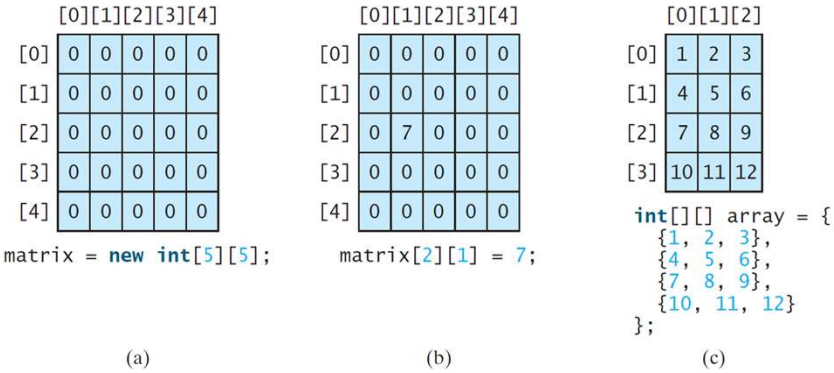

Declaring Variables of Two-dimensional Arrays and Creating Two-dimensional Arrays

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

or

```
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;
```

Two-dimensional Array Illustration



matrix.length? 5

matrix[0].length? 5

array.length? 4

array[0].length? 3

Pearson

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,

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

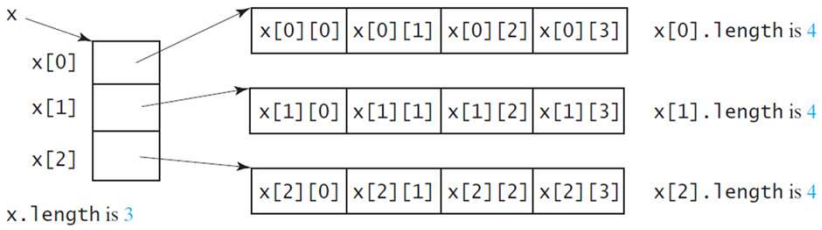
Same as

```
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;
```

Pearson

Lengths of Two-dimensional Arrays (1 of 2)

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



Lengths of Two-dimensional Arrays (2 of 2)

```
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 (1 of 2)

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,

```
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

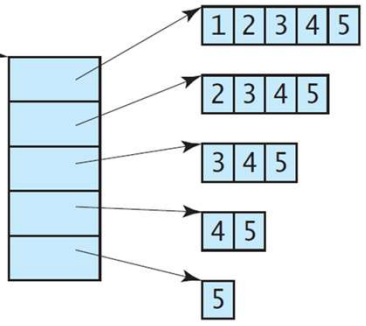
matrix[2].length is 3

matrix[3].length is 2

matrix[4].length is 1

Ragged Arrays (2 of 2)

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



Processing Two-Dimensional Arrays

See the examples in the text.

1. (Initializing arrays with input values)
2. (Printing arrays)
3. (Summing all elements)
4. (Summing all elements by column)
5. (Summing all elements by column)
6. (Which row has the largest sum)
7. (Finding the smallest index of the largest element)
8. (**Random shuffling**)

Passing Tow-Dimensional Arrays to Methods

PassTwoDimensionalArray