MULTIDIMENSIONAL ARRAYS

Two-Dimensional Array Basics

- *Data in a table or a matrix can be represented using a two-dimensional array*
- *element in a two-dimensional array is accessed through a row and column index*
- syntax for declaring a two-dimensional array
  - elementType[][] arrayRefVar; or elementType arrayRefVar[][];
  - **int**[][] matrix; or **int** matrix[][];
- matrix = **new int[5][5];**

	[0]	[1]	[2]	[3]	[4]
[0]	0	0	0	0	0
[1]	0	0	0	0	0
[2]	0	0	0	0	0
[3]	0	0	0	0	0
[4]	0	0	0	0	0

```
matrix = new int[5][5];
```

	[0]	[1]	[2]	[3]	[4]
[0]	0	0	0	0	0
[1]	0	0	0	0	0
[2]	0	7	0	0	0
[3]	0	0	0	0	0
[4]	0	0	0	0	0

```
matrix[2][1] = 7;
```

	[0]	[1]	[2]
[0]	1	2	3
[1]	4	5	6
[2]	7	8	9
[3]	10	11	12

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

Initialize an array

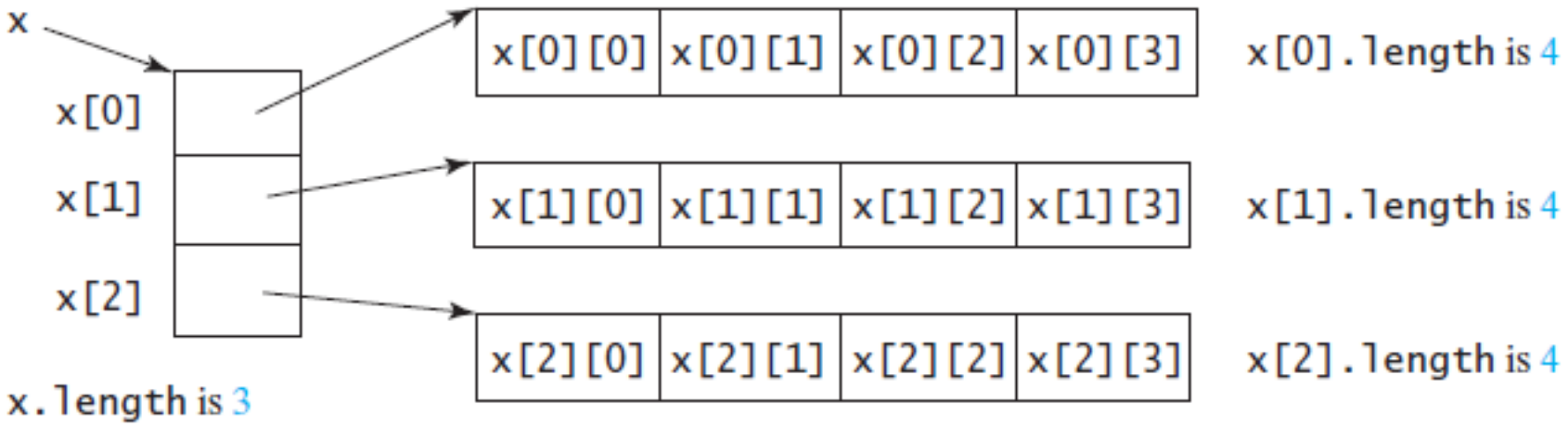
```
int[][] array = {
  {1, 2, 3},
  {4, 5, 6},
  {7, 8, 9},
  {10, 11, 12}
};
```

Equivalent

```
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;
```

Obtaining the Lengths of Two-Dimensional Arrays

- A two-dimensional array is actually an array in which each element is a one-dimensional array



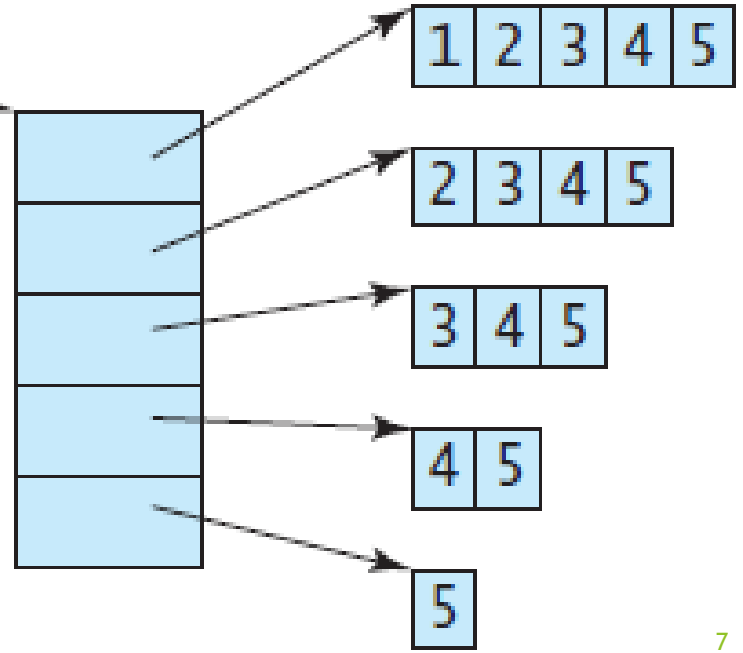
Obtaining the Lengths of Two-Dimensional Arrays (Contd…)

- The length of an array **x** is the number of elements in the array, which can be obtained using **x.length**
  - **x[0]**, **x[1]**, . . . , and **x[x.length-1]**
  - **x[0].length**, **x[1].length**, . . . , and **x[x.length-1].length**

Ragged Arrays

- Each row in a two-dimensional array is itself an array. Thus, the rows can have different lengths. An array of this kind is known as a *ragged array*

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



create a ragged array

- **int**[][] triangleArray = **new int**[5][];

  triangleArray[**0**] = **new int**[5];

  triangleArray[**1**] = **new int**[4];

  triangleArray[**2**] = **new int**[3];

  triangleArray[**3**] = **new int**[2];

  triangleArray[**4**] = **new int**[1];

Processing Two-Dimensional Arrays

- *Nested* **for** *loops are used to process a two-dimensional array*
- *Initializing arrays with input values*
- 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();
}
}
```

Processing Two-Dimensional Arrays (Contd…)

- Other Processes
  - *Initializing arrays with random values*
  - *Printing arrays*
  - *Summing all elements*
  - *Summing elements by column*
  - *Which row has the largest sum?*

Passing Two-Dimensional Arrays to Methods

- *When passing a two-dimensional array to a method, the reference of the array is passed to the method*
- You can pass a two-dimensional array to a method just as you pass a one-dimensional array

```
public class PassTwoDimensionalArray {
  public static void main(String[] args) {
  int[][] m = getArray(); // Get an array
  System.out.println("\nSum of all elements is " + sum(m));
   }
public static int[][] getArray() {
  Scanner input = new Scanner(System.in);
  int[][] m = new int[3][4];
  System.out.println("Enter " + m.length + " rows and “ + m[0].length + " columns: ");
  for (int i = 0; i < m.length; i++)
  for (int j = 0; j < m[i].length; j++)
       m[i][j] = input.nextInt();
  return m;
  }
```

Multidimensional Arrays

- *two-dimensional array consists of an array of one-dimensional arrays*
- *Three dimensional array consists of an array of two-dimensional arrays*
- **double**[][][] scores = **new double**[**6**][**5**][**2**];
- **Suppose x = new int[2][2][5]**
  - **x[0]** and **x[1]** are two-dimensional arrays
  - **x[0][0]**, **x[0][1]**, **x[1][0]**, and **x[1][1]** are one-dimensional arrays and each contains five elements
  - **x.length** is **2**
  - **x[0].length** and **x[1].length** are **2**
  - **x[0][0].length**, **x[0][1].length**, **x[1][0].length**, and **x[1][1].length** are **5**