CIS2571 – Intro to Java

Chapter7 → Multidimensional Arrays

Information consolidated from a variety of textbook and online resources

Topic Objectives

- Understand how two dimensional arrays can be used to represent matrices or tables
- Know how to use two dimensional arrays
  - Declare
  - Create
  - Access Elements
- Recognize the differences in a ragged two dimensional array
- Understand how to represent multidimensional arrays in Java

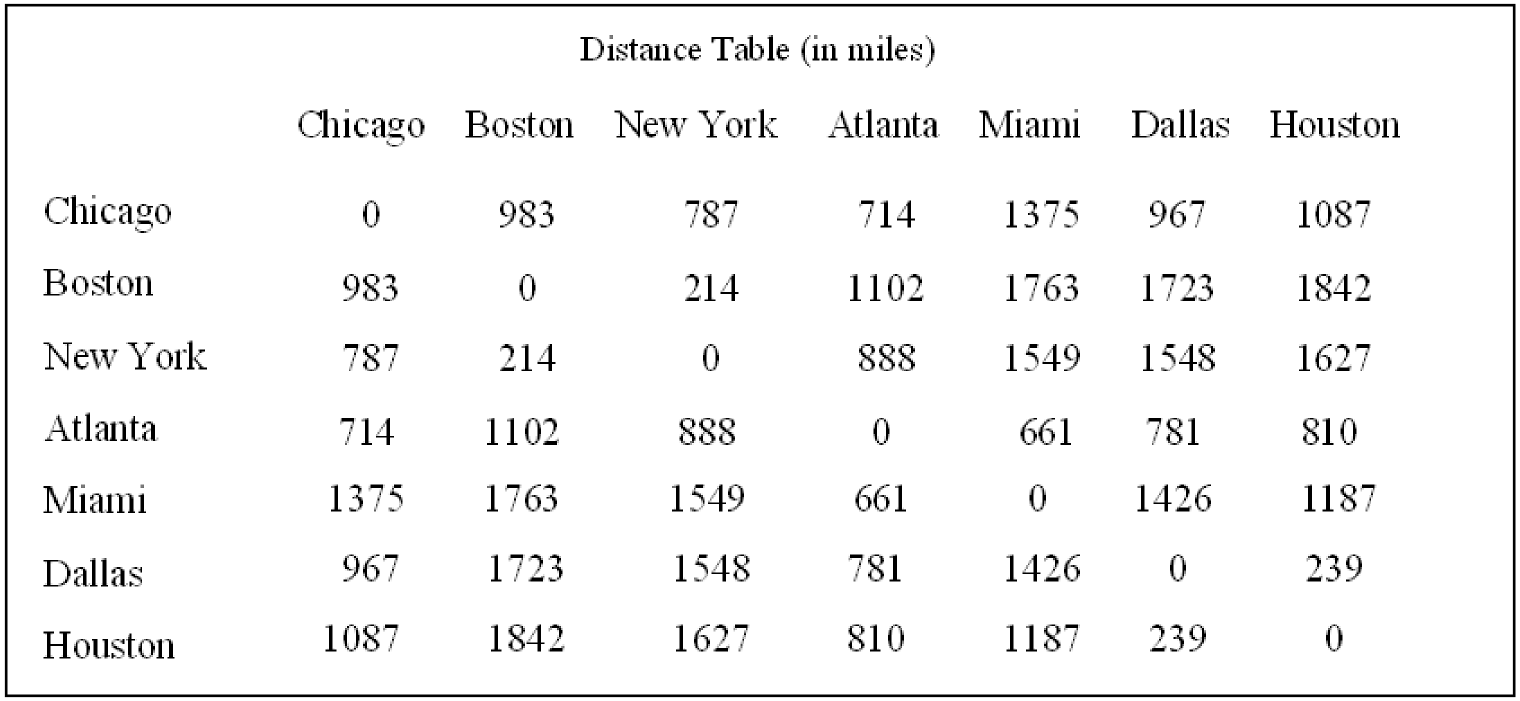
One Dimensional Arrays

- One dimensional arrays are used to model **linear collections of data**
- For example, the following list displays the number of students in each programming course

	students
CIS1400 – Programming Logic and Technique	*110*
CIS1510 – Graphical User Interface Programming	*15*
CIS2510 – Advanced Graphical User Interface Programming	*0*
CIS2541 – C++ Language Programming	*45*
CIS2542 – Advanced C++ Language Programming	*25*
CIS2571 – Introduction to Java	*45*
CIS2572 – Collections in Java	*15*

Two Dimensional Arrays

- Two dimensional array are used to model a collection of data such as a **matrix**, or **table**
- For example, the following table lists distances between cities



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

Two Dimensional Arrays

- And the two dimensional array of distances can be represented as follows:

```
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}
};
```

Using Two Dimensional Arrays

- Declaring Array Reference Variables
  - Declare a variable to reference the array and specify the array's element type

```
elementType[][] matrixRefVar;
elementType matrixRefVar[][]; // allowed,
                                // but !preferred
```

- Creating Array and Assigning to Reference Variable

```
matrixRefVar = new elementType[rowSize][colSize];
```

Using Two Dimensional Arrays

- Declaring, Creating, and Assigning Arrays

```
elementType[][] matrixRefVar = new
  elementType[rowSize][colSize];
```

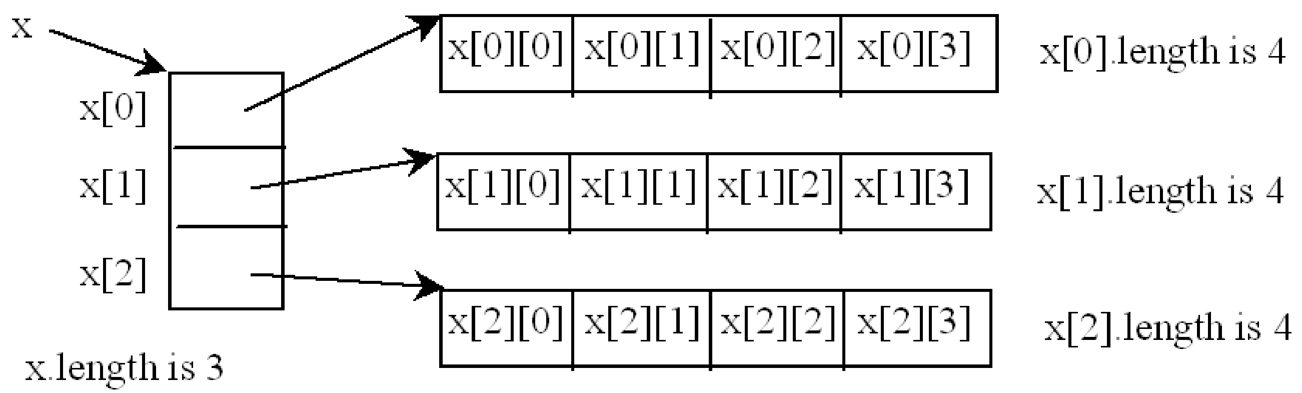
- Assigning values to array elements

```
matrixRefVar[row][col] = value;
matrixRefVar[row, col] = value; // error!
```

- Like one dimensional arrays, elements initialized to:
  - 0 for numerical
  - '\u0000' for char *(nul)*
  - false for boolean

Using Two Dimensional Arrays

- Two dimensional array is an array of one dimensional arrays
- The length of the two dimensional array:
  - Number of rows → **`matrixRefVar.length`**
  - Number of cols → **`matrixRefVar[rowIndex].length`**


x
x[0]
x[1]
x[2]
x.length is 3
x[0][0] x[0][1] x[0][2] x[0][3]
x[1][0] x[1][1] x[1][2] x[1][3]
x[2][0] x[2][1] x[2][2] x[2][3]
x[0].length is 4
x[1].length is 4
x[2].length is 4

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

Two Dimensional Array Example

	[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]	[3]
[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}
};
```

matrix.length → 5
matrix[0].length → 5

array.length → 4
array[0].length → 3

Two Dimensional Array Example

- Array initializer used to declare, create, and initialize two dimensional array.

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

```
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;
```

equivalent

Processing Two Dimensional Arrays

- Processing all elements of a two dimensional array
  - Loop for each row
    - Loop for each column in row

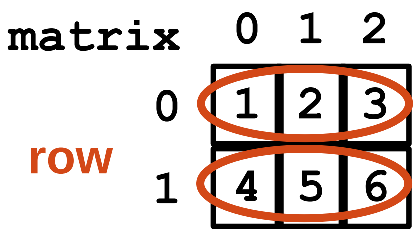
```
for (int row = 0; row < matrix.length; row++)
  for (int col = 0; col < matrix[row].length; col++)
    System.out.println(matrix[row][col]);
```

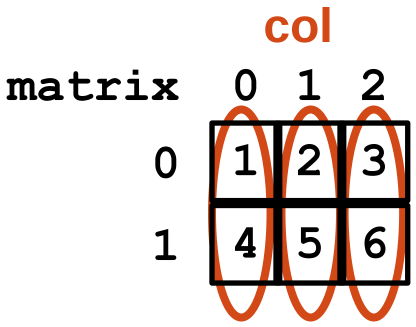
- Examples for processing arrays:
  - Initializing elements with values
  - Displaying elements
  - Summing elements

$$\begin{array}{c|ccc} \texttt{matrix} & 0 & 1 & 2 \\ \hline 0 & 1 & 2 & 3 \\ 1 & 4 & 5 & 6 \end{array}$$

Processing Two Dimensional Arrays

- Sub-array processing
  - For each row
    - hold row index constant, loop over column indices
  - For each column
    - hold column index constant, loop over row indices





Passing Two Dimensional Arrays to Methods

Define the method

```
public static void printArray(int[][] matrix) {
  for (int row = 0; row < matrix.length; row++) {
    for (int col = 0; col < matrix[row].length; col++)
      System.out.print(matrix[row][col] + “ “);
    System.out.print("\n");
  }
}
```

Invoke the method

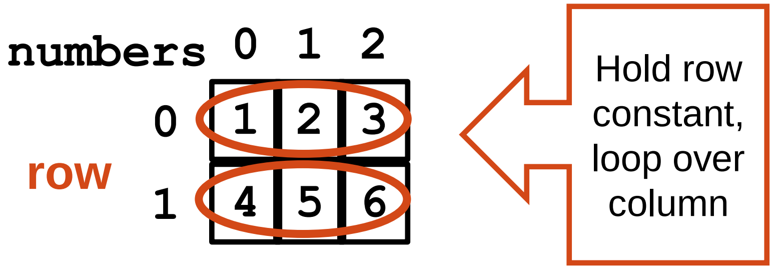
```
int[][] numbers = {{1, 2, 3}, {4, 5, 6}};
printArray(numbers);
```

See Test2DArray.java

Processing Two Dimensional Arrays

- Summing Elements By Row

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

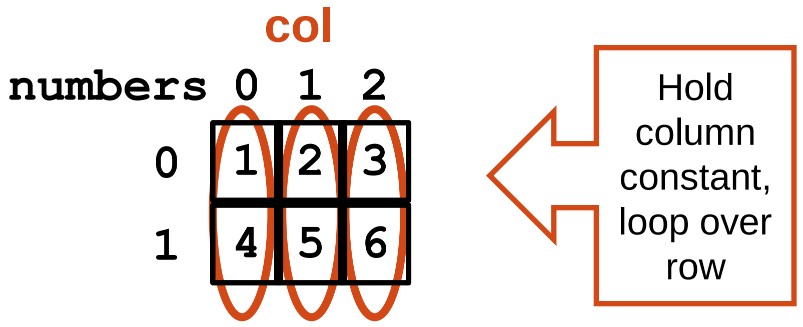


See Test2DArray.java

Processing Two Dimensional Arrays

- Summing Elements By Column

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

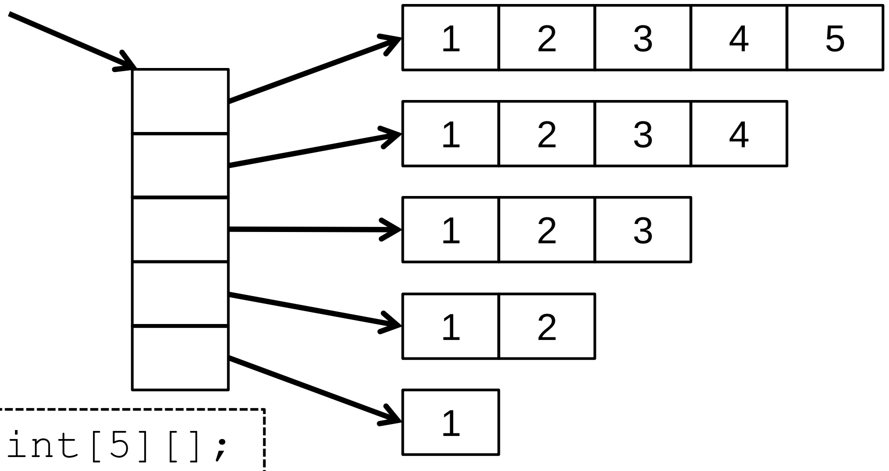


See Test2DArray.java

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**.

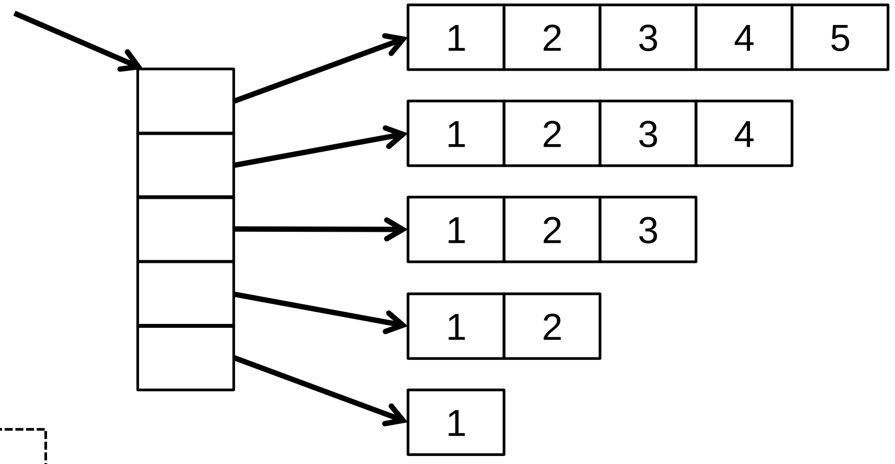
```
int [][] triangleArray = {
  {1, 2, 3, 4, 5},
  {1, 2, 3, 4},
  {1, 2, 3},
  {1, 2},
  {1}
};
```



```
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];
```

Ragged Arrays

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

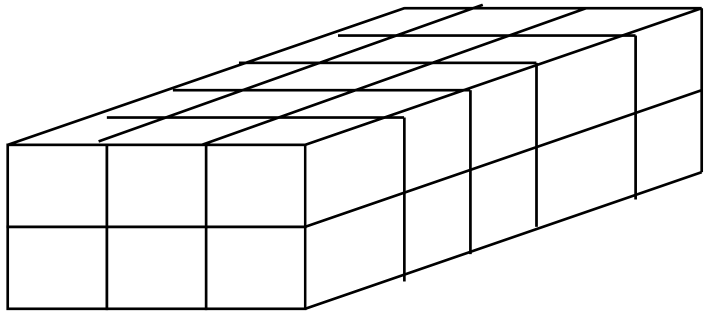


```
  triangleArray.length is 5
triangleArray[0].length is 5
triangleArray[1].length is 4
triangleArray[2].length is 3
triangleArray[3].length is 2
triangleArray[4].length is 1
```

See Test2DRaggedArray.java

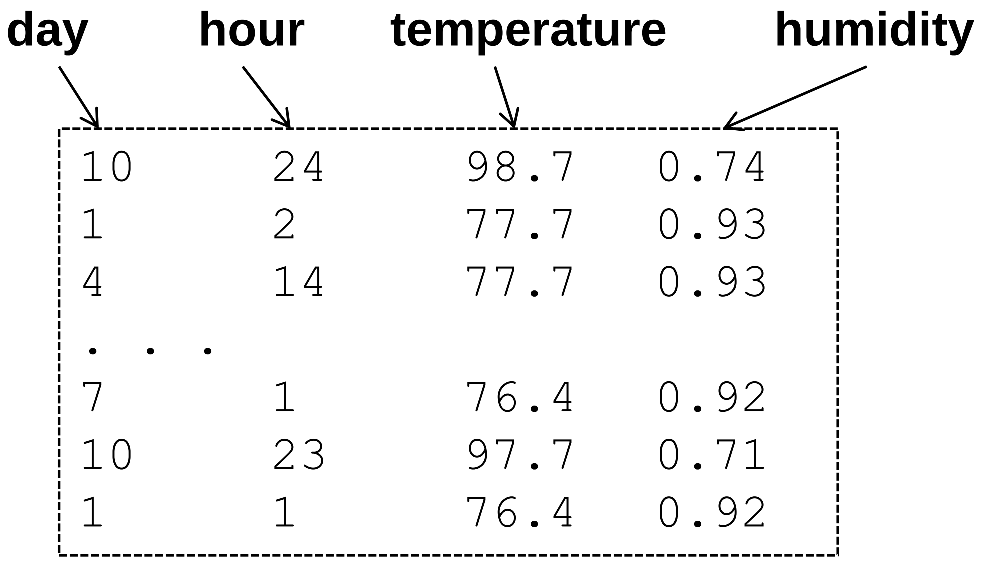
Multidimensional Arrays

- In Java, n-dimensional arrays can be created for any integer n.
- An n-dimensional array can be generalized to an array of n-1 dimensional arrays
    - Two dimensional array is an array of one dimensional arrays
    - Three dimensional array is an array of two dimensional arrays
    - Etc.



Multidimensional Arrays Example

- A meteorology station records the temperature and humidity every hour of every day and store data for past ten days in file
  - Format of data file: Weather.txt



```
10        24        98.7    0.74
1         2         77.7    0.93
4         14        77.7    0.93
. . .
7         1         76.4    0.92
10        23        97.7    0.71
1         1         76.4    0.92
```

Multidimensional Arrays Example

- Problem → calculate the average daily temperature and humidity for 10 days
- Create a three-dimensional array variable to store the temperature and humidity [2] at each hour of every day [24] for the past ten days [10]:

```
double[][][] data = new double[10][24][2];
```

Which day
Which hour
Temperature or Humidity

See 7.5 Weather.java