CSE 215L: Programming Language II Lab
Faculty: Silvia Ahmed, Sec – 4, 5
Lab instructor: Marufa Ferdausi
Lab 06 – Fall 2019

Objective:
After today's lab, the students should be able:

- To declare variables for two-dimensional arrays, create arrays, and access array elements in a two-dimensional array using row and column Indexes
- To pass two-dimensional arrays to methods
- To use multidimensional arrays

Two-dimensional arrays declaration	Two-dimensional arrays creation
elementType[][] arrayVar	new elementType [ROW_SIZE][COLUMN_SIZE]
Two-dimensional array elements	**Two-dimensional array using an array initializer**
arrayVar[rowIndex][columnIndex]	elementType[][] arrayVar = {{row values}, . . . , {row values}}
Three-dimensional arrays	
elementType[][][] arrayVar new elementType[size1][size2][size3]	

Following code gives an example with two methods. The first method, **getArray()**, returns a two-dimensional array, and the second method, **sum(int[][] m)**, returns the sum of all the elements in a matrix.

```
import java.util.Scanner;
public class PassTwoDimensionalArray {
	public static void main(String[] args) {
		int[][] m = getArray(); // Get an array
		// Display sum of elements
		System.out.println("\nSum of all elements is " + sum(m));
	}
	public static int[][] getArray() {
		Scanner input = new Scanner(System.in);
		 // Enter array values
		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;
	}
	public static int sum(int[][] m) {
		 int total = 0;
		 for (int row = 0; row < m.length; row++) {
		  for (int column = 0; column < m[row].length; column++) {
		   total += m[row][column];
		  }
		 }
	return total;
	}
}
```

Here is a sample run:

```
Enter 3 rows and 4 columns:
1 2 3 4
5 6 7 8
9 10 11 12
Sum of all elements is 78
```

Task – 1

(*Sum elements column by column*) Write a method that returns the sum of all the elements in a specified column in a matrix using the following header:

public static double sumColumn(**double**[][] m, **int** columnIndex)

Write a test program that reads a 3-by-4 matrix and displays the sum of each column. Here is a sample run:

```
Enter a 3-by-4 matrix row by row:
1.5 2 3 4
5.5 6 7 8
9.5 1 3 1
Sum of the elements at column 0 is 16.5
Sum of the elements at column 1 is 9.0
Sum of the elements at column 2 is 13.0
Sum of the elements at column 3 is 13.0
```

Task – 2

(*Sum the major diagonal in a matrix*) Write a method that sums all the numbers in the major diagonal in an *n* * *n* matrix of **double** values using the following header:

public static double sumMajorDiagonal(**double**[][] m)

Write a test program that reads a 4-by-4 matrix and displays the sum of all its elements on the major diagonal. Here is a sample run:

```
Enter a 4-by-4 matrix row by row:
1 2 3 4.0
5 6.5 7 8
9 10 11 12
13 14 15 16
Sum of the elements in the major diagonal is 34.5
```

Task – 3

(*Largest row and column*) Write a program that randomly fills in 0s and 1s into a 4-by-4 matrix, prints the matrix, and finds the first row and column with the most 1s. Here is a sample run of the program:

```
0011
0011
1101
1010
The largest row index: 2
The largest column index: 2
```

Task – 4

(*Locate the largest element*) Write the following method that returns the location of the largest element in a two-dimensional array.

public static int[] locateLargest(**double**[][] a)

The return value is a one-dimensional array that contains two elements. These two elements indicate the row and column indices of the largest element in the two-dimensional array. Write a test program that prompts the user to enter a twodimensional array and displays the location of the largest element in the array. Here is a sample run:

```
Enter the number of rows and columns of the array: 3 4
Enter the array:
23.5 35 2 10
4.5 3 45 3.5
35 44 5.5 9.6
The location of the largest element is at (1, 2)
```

Task – 5

(*Sort two-dimensional array*) Write a method to sort a two-dimensional array using the following header:

public static void sort(**int** m[][])

The method performs a primary sort on rows and a secondary sort on columns. For example, the following array

{{**4**, **2**},{**1**, **7**},{**4**, **5**},{**1**, **2**},{**1**, **1**},{**4**, **1**}} will be sorted to {{**1**, **1**},{**1**, **2**},{**1**, **7**},{**4**, **1**},{**4**, **2**},{**4**, **5**}}.

Task – 6

(*Shuffle rows*) Write a method that shuffles the rows in a two-dimensional **int** array using the following header:

public static void shuffle(**int**[][] m)

Write a test program that shuffles the following matrix:

int[][] m = {{**1**, **2**}, {**3**, **4**}, {**5**, **6**}, {**7**, **8**}, {**9**, **10**}};

Task – 7

(*Markov matrix*) An *n* * *n* matrix is called a *positive Markov matrix* if each element is positive and the sum of the elements in each column is 1. Write the following method to check whether a matrix is a Markov matrix.

public static boolean isMarkovMatrix(**double**[][] m)

Write a test program that prompts the user to enter a 3 * 3 matrix of double values and tests whether it is a Markov matrix. Here are sample runs:

```
Enter a 3-by-3 matrix row by row:
0.15 0.875 0.375
0.55 0.005 0.225
0.30 0.12 0.4
It is a Markov matrix
```

```
Enter a 3-by-3 matrix row by row:
0.95 -0.875 0.375
0.65 0.005 0.225
0.30 0.22 -0.4
It is not a Markov matrix
```

Task – 8

(*Strictly identical arrays*) The two-dimensional arrays **m1** and **m2** are *strictly identical* if their corresponding elements are equal. Write a method that returns **true** if **m1** and **m2** are strictly identical, using the following header:

public static boolean equals(**int**[][] m1, **int**[][] m2)

Write a test program that prompts the user to enter two 3 * 3 arrays of integers and displays whether the two are strictly identical. Here are the sample runs.

```
Enter list1: 51 22 25 6 1 4 24 54 6
Enter list2: 51 22 25 6 1 4 24 54 6
The two arrays are strictly identical
```

```
Enter list1: 51 25 22 6 1 4 24 54 6
Enter list2: 51 22 25 6 1 4 24 54 6
The two arrays are not strictly identical
```