North South University
Department of Electrical and Computer Engineering
CSE 215L: Programming Language II Lab
Lab Manual - 6
Lab Instructor: Taif Al Musabe

Objective:

- To declare array reference variables and create arrays
- To declare, create, and initialize an array using an array initializer
- To copy contents from one array to another
- To develop and invoke methods with array arguments and return values
- 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

Declaring Array Variables and Creating Arrays

Here is the syntax for declaring an array variable:

```
elementType[] arrayRefVar;
```

After an array variable is declared, you can create an array by using the **new** operator and assign its reference to the variable with the following syntax:

```
arrayRefVar = new elementType[arraySize];
```

Declaring an array variable, creating an array, and assigning the reference of the array to the variable can be combined in one statement as:

```
elementType[] arrayRefVar = new elementType[arraySize];
```

Accessing Array Elements

The array elements are accessed through the index. Array indices are **0** based; that is, they range from **0** to **arrayRefVar.length-1**. Each element in the array is represented using the following syntax, known as an *indexed variable*:

```
arrayRefVar[index];
```

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;
      }
}
```

Task – 1

(*Count occurrence of numbers*) Write a program that reads the integers between 1 and 100 and counts the occurrences of each. Assume the input ends with 0. Here is a sample run of the program:

```
Enter the integers between 1 and 100: 2 5 6 5 4 3 23 43 2 0
2 occurs 2 times
3 occurs 1 time
4 occurs 1 time
5 occurs 2 times
6 occurs 1 time
23 occurs 1 time
43 occurs 1 time
```

Note that if a number occurs more than one time, the plural word ―times‖ is used in the output.

Task – 2

(*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
```

Homework – 1

(*Print distinct numbers*) Write a program that reads in ten numbers and displays the number of distinct numbers and the distinct numbers separated by exactly one space (i.e., if a number appears multiple times, it is displayed only once). (Hint: Read a number and store it to an array if it is new. If the number is already in the array, ignore it.) After the input, the array contains the distinct numbers. Here is the sample run of the program:

```
Enter ten numbers: 1 2 3 2 1 6 3 4 5 2
The number of distinct number is 6
The distinct numbers are: 1 2 3 6 4 5
```

Homework – 2

(*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
```