CSE 215L: Programming Language II Lab
Faculty: Silvia Ahmed, Sec – 12, 13
Lab 05 – Summer 2020

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

- 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

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;
        }
}
```

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

(*Reverse the numbers entered*) Write a program that reads ten integers and displays them in the reverse of the order in which they were read.

Task – 2

(*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 – 3

(*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
```

Task – 4

A number **n** is prime by checking whether **2**, **3**, **4**, **5**, **6**, . . . , **n/2** is a divisor. If a divisor is found, **n** is not prime. A more efficient approach is to check whether any of the prime numbers less than or equal to $\sqrt{n}$ can

divide **n** evenly. If not, **n** is prime. Display the first 50 prime numbers using this approach. You need to use an array to store the prime numbers and later use them to check whether they are possible divisors for **n**.

Task – 5

(*Count single digits*) Write a program that generates 100 random integers between 0 and 9 and displays the count for each number. (*Hint*: Use an array of ten integers, say **counts**, to store the counts for the number of 0s, 1s, . . . , 9s.).

Task – 6

(*Average an array*) Write two overloaded methods that return the average of an array with the following headers:

public static int average(**int**[] array)
public static double average(**double**[] array)

Write a test program that prompts the user to enter ten double values, invokes this method, and displays the average value.

Task – 7

(*Sorted?*) Write the following method that returns true if the list is already sorted in increasing order.

public static boolean isSorted(**int**[] list)

Write a test program that prompts the user to enter a list and displays whether the list is sorted or not. Here is a sample run. Note that the first number in the input indicates the number of the elements in the list. This number is not part of the list.

```
Enter list: 8 10 1 5 16 61 9 11 1
The list is not sorted
Enter list: 10 1 1 3 4 4 5 7 9 11 21
The list is already sorted
```

Task – 8

(*Identical arrays*) The arrays **list1** and **list2** are *identical* if they have the same contents. Write a method that returns **true** if **list1** and **list2** are identical, using the following header:

public static boolean equals(**int**[] list1, **int**[] list2)

Write a test program that prompts the user to enter two lists of integers and displays whether the two are identical. Here are the sample runs. Note that the first number in the input indicates the number of the elements in the list. This number is not part of the list.

```
Enter list1: 5 2 5 6 6 1
Enter list2: 5 5 2 6 1 6
Two lists are identical
Enter list1: 5 5 5 6 6 1
Enter list2: 5 2 5 6 1 6
Two lists are not identical
```

Task – 9

(*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 – 10

(*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 – 11

(*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 – 12

(*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 – 13

(*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 – 14

(*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