CSE 215L: Programming Language II Lab
Faculty: Silvia Ahmed, Sec – 4, 5
Lab instructor: Marufa Ferdausi
Lab 05 – Fall 2019

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

- 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

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];
```

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

(*Pattern recognition: consecutive four equal numbers*) Write the following method that tests whether the array has four consecutive numbers with the same value.

public static boolean isConsecutiveFour(**int**[] values)

Write a test program that prompts the user to enter a series of integers and displays if the series contains four consecutive numbers with the same value. Your program should first prompt the user to enter the input size—i.e., the number of values in the series. Here are sample runs:

```
Enter the number of values: 8
Enter the values: 3 4 5 5 5 5 4 5
The list has consecutive fours
Enter the number of values: 9
Enter the values: 3 4 5 5 6 5 5 4 5
The list has no consecutive fours
```