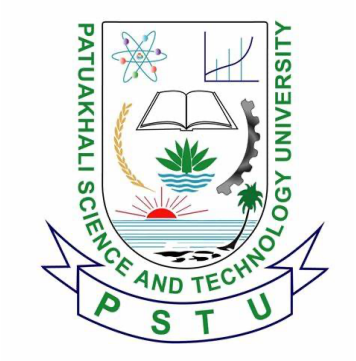
PATUAKHALI SCIENCE AND TECHNOLOGY UNIVERSITY



ASSIGNMENT ON- Assignment 08
(Chapter 07)

Course Code: CCE- 121

Submitted By: Md. Mehedi Hasan

ID: 2102016

Reg. No: 10143

Faculty of Computer Science & Engineering

7.1 Fill in the blanks in each of the following statements:

a) Lists and tables of values can be stored in **arrays** and **collection**.
b) An array is a group of **variables** (called elements or components) containing values that all have the same **type**.
c) The **enhanced for statement** allows you to iterate through an array's elements without using a counter.
d) The number used to refer to a particular array element is called the element's **index**.
e) An array that uses two indices is referred to as a(n) **two dimensional** array.
f) Use the enhanced for statement **for(double d: array)** to walk through double array numbers.
g) Command-line arguments are stored in **an array of String**.
h) Use the expression **args.length** to receive the total number of arguments in a command line. Assume that command-line arguments are stored in String[] args.
i) Given the command *java MyClass test*, the first command-line argument is **test**.
j) A(n) **ellipsis (...)** in the parameter list of a method indicates that the method can receive a variable number of arguments.

7.2 Determine whether each of the following is true or false. If false, explain why.

a) An array can store many different types of values.

Ans: False. An array can store only one type of value.

b) An array index should normally be of type float.

Ans: False. An array index can be of integer type.

c) An individual array element that's passed to a method and modified in that method will contain the modified value when the called method completes execution.

Ans: False. A called method receives and manipulates a copy of the value of such an element, so modifications do not affect the original value.

d) Command-line arguments are separated by commas.

Ans: False. They are seperated by whitespace.

7.3 Perform the following tasks for an array called fractions:

a) Declare a constant ARRAY_SIZE that's initialized to 10.

```
1  final int ARRAY_SIZE = 10;
```

b) Declare an array with ARRAY_SIZE elements of type double, and initialize the elements to 0.

```
1  double[] fractions = new double[ARRAY_SIZE];
```

c) Refer to array element 4.

```
1  fractions[4]
```

d) Assign the value 1.667 to array element 9.

```
1  fractions[9] = 1.667;
```

e) Assign the value 3.333 to array element 6.

```
1  fractions[6] = 3.333;
```

f)Sum all the elements of the array, using a for statement. Declare the integer variable x as a control variable for the loop.

```
1  double total = 0.0;
2        for (int x = 0; x < fractions.length; x++)
3                total += fractions[x];
```

7.4 Perform the following tasks for an array called table:

a) Declare and create the array as an integer array that has three rows and three columns.

```
1  int[][] table = new int[ARRAY_SIZE][ARRAY_SIZE];
```

b) How many elements does the array contain? **Nine**.

c) Use a for statement to initialize each element of the array to the sum of its indices. Assume that the integer variables x and y are declared as control variables.

```
1  for (int x = 0; x < table.length; x++)
2        for (int y = 0; y < table[x].length; y++)
3              table[x][y] = x + y;
```

7.5 Find and correct the error in each of the following program segments:

a) final int ARRAY_SIZE = 5;

ARRAY_SIZE = 10;

Ans: Assigning a value to a constant after it has been initialized.

Correction: Assign the correct value to the constant in a final int ARRAY_SIZE declaration or declare another variable.

b) Assume

```
1  int[] b = new int[10];
2        for (int i = 0; i <= b.length; i++)
3              b[i] = 1;
```

Ans: *Referencing an array element outside the bounds of the array (b[10]).*

Correction: Change the <= operator to <.

c) Assume int[][] a = {{1, 2}, {3, 4}};

a[1, 1] = 5;

Ans: *Array indexing is performed incorrectly.*

7.6 Fill in the blanks in each of the following statements:

a) A one-dimensional array p contains five elements. The names of the third and fourth elements are **p[2]** and **p[3]**.

b) A one-dimensional array k has three elements. The statement **k[1] = 2** sets the value of the second element to 2.

c) A statement to declare a two-dimensional int array r that has 3 rows and 4 columns is **int r[][] = new int[3][4]**.

d) A 5-by-6 array contains **5** rows, **6** columns and **30** elements.

e) The name of the element in column 5 and row 6 of an array d is **d[5][4]**.

7.7 Determine whether each of the following is true or false. If false, explain why.

a) To refer to a particular location or element within an array, we specify the name of the array and the order of the element in the array, assuming ordering starts at position 1.

Ans: False. Ordering starts at position 0.

b) An array declaration initializes the elements in the array to the integer 0 by default.

Ans: True.

c) To indicate that 200 locations should be reserved for integer array p, you write the declaration int p[] = new int[200];

Ans: True.

d) For an application that initializes the elements of a twenty-element integer array to zero, it is preferable to use some kind of loop.

Ans: False. Java will by default initialize them to 0.

e) To access all the elements in a two-dimensional array using a loop, the traversal across rows must be done in the outer loop and the traversal across columns in the inner loop.

Ans: True.

7.8 Write Java statements to accomplish each of the following tasks:

a) Display the value of the tenth element of array r.

```
1       System.out.println(array[9]);
```

b) Initialize each of the six elements of one-dimensional integer array g to -1.

```
1  public class Test {
2     public static void main(String[] args) {
3        int[] array = new int[6];
4        for (int i = 0; i < array.length; i++) {
5           array[i] = -1;
6        }
7     }
8  }
```

c) Find the maximum of the first one-hundred elements of floating-point array c.

```
1  float maximum = 0;
2        for (int i = 0; i < array.length; i++) {
3           if (array[i] > maximum) {
4              maximum = array[i];
5           }
6        }
```

d) Copy a hundred-element array a into a hundred-element array b, but in reverse order.

```
1  int[] b = new int[100];
2        for (int i = a.length - 1; i >= 0; i--) {
3           b[i] = a[i];
4        }
```

e) Compute the product of the third to the tenth elements, both inclusive, in a hundred element integer array w.

```
1  int product = 1;
2        for (int i = 2; i < 9; i++) {
3           product *= b[i];
4        }
```

7.9 Consider a two-by-three integer array t.

a) Write a statement that declares and creates t.

```
1  public class Test {
2     public static void main(String[] args) {
3        int[][] t = new int[2][3];
4     }
5  }
```

b) How many rows does t have?
Ans: 2

c) How many columns does t have?
Ans: 3

d) How many elements does t have?
Ans: 6

e) Write access expressions for all the elements in row 1 of t.

```
t[1][0] = 1;
t[1][1] = 2;
t[1][2] = 3;
```

f) Write access expressions for all the elements in column 2 of t.

```
t[0][2] = 1;
t[1][2] = 2;
```

g) Write a single statement that sets the element of t in row 0 and column 1 to zero.

```
t[0][1] = 1;
```

h) Write individual statements to initialize each element of t to zero.

```
1   t[0][0] = 0;
2   t[0][1] = 0;
3   t[0][2] = 0;
4   t[1][0] = 0;
5   t[1][1] = 0;
6   t[1][2] = 0;
```

i) Write a nested for statement that initializes each element of t to zero.

```
 1  public class Test {
 2    public static void main(String[] args) {
 3      int[][] t = new int[2][3];
 4      for (int row = 0; row < t.length; row++) {
 5        for (int column = 0; column < t[row].length; column++) {
 6          t[row][column] = 0;
 7        }
 8      }
 9    }
10  }
```

j) Write a nested for statement that inputs the values for the elements of t from the user.

```
 1  import java.util.Scanner;
 2  public class Test {
```

```
3     public static void main(String[] args) {
4        int[][] t = new int[2][3];
5        Scanner input = new Scanner(System.in);
6        int value = input.nextInt();
7        input.close();
8        for (int row = 0; row < t.length; row++) {
9           for (int column = 0; column < t[row].length; column++) {
10             t[row][column] = value;
11          }
12       }
13    }
14 }
```

k) Write a series of statements that determines and displays the smallest value in t.

```
1  int smallest = t[0][0];
2        for (int row = 0; row < t.length; row++) {
3           for (int column = 0; column < t[row].length; column++) {
4              if (t[row][column] < smallest) {
5                 smallest = t[row][column];
6              }
7           }
8        }
9        System.out.printf("Smallest value in t is %d%n", smallest);
```

l) Write a single printf statement that displays the elements of the first row of t.

```
1  for (int column = 0; column < t[0].length; column++) {
2           System.out.printf("%d ", t[0][column]);
3        }
```

m) Write a statement that totals the elements of the third column of t. Do not use repetition.

```
int total = t[0][2] + t[1][2];
```

n) Write a series of statements that displays the contents of t in tabular format. List the column indices as headings across the top, and list the row indices at the left of each row.

```
1        System.out.printf("%s%8s%8s%8s%n", " ", "0", "1", "2");
2        for (int row = 0; row < t.length; row++) {
3           System.out.printf("%d", row);
4           for (int column = 0; column < t[row].length; column++) {
5              System.out.printf("%8d", t[row][column]);
6           }
7           System.out.println();
8        }
```

7.10 (Pixel Quantization)

```
1  public class PixelQuantization {
```

```
    public static void main(String[] args) {
        int[] pixelValues = {15, 35, 50, 75, 90, 105, 130, 155, 180, 200};
        quantizePixels(pixelValues);

        for (int value : pixelValues) {
            System.out.print(value + " ");
        }
    }

    private static void quantizePixels(int[] pixels) {
        for (int i = 0; i < pixels.length; i++) {
            int value = pixels[i];

            if (value >= 0 && value <= 20) {
                pixels[i] = 10;
            } else if (value <= 40) {
                pixels[i] = 30;
            } else if (value <= 60) {
                pixels[i] = 50;
            } else if (value <= 80) {
                pixels[i] = 70;
            } else if (value <= 100) {
                pixels[i] = 90;
            } else if (value <= 120) {
                pixels[i] = 110;
            } else if (value <= 140) {
                pixels[i] = 130;
            } else if (value <= 160) {
                pixels[i] = 150;
            } else if (value <= 180) {
                pixels[i] = 170;
            } else {
                pixels[i] = 190;
            }
        }
    }
}
```

7.11 Write statements that perform the following one-dimensional-array operations:

a) Set elements of index 10–20, both inclusive, of integer array counts to zero.

```
1       for (int i = 10; i <= 20; i++) {
2           ar[i] = 0;
3       }
```

b) Multiply each of the twenty elements of integer array bonus by 2.

```
1       for (int i = 0; i < 20; i++) {
2           ar[i] *= 2;
3       }
```

c) Display the ten values of integer array bestScores, each on a new line.

```
1        for (int i = 0; i < 10; i++) {
2           System.out.println(ar[i]);
3       }
```

7.12 (Duplicate Elimination)

```
 1  import java.util.Scanner;
 2
 3  public class RemoveDuplicates {
 4     public static void main(String[] args) {
 5        int[] numbers = new int[10];
 6        initializeArray(numbers, -1);
 7        inputNumbers(numbers);
 8        removeDuplicates(numbers);
 9        displayArray(numbers);
10     }
11
12     private static void initializeArray(int[] array, int value) {
13        for (int i = 0; i < array.length; i++) {
14           array[i] = value;
15        }
16     }
17
18     private static void inputNumbers(int[] numbers) {
19        Scanner scanner = new Scanner(System.in);
20        System.out.println("Enter ten numbers: ");
21        for (int i = 0; i < numbers.length; i++) {
22           numbers[i] = scanner.nextInt();
```

```
23        }
24        scanner.close();
25    }
26
27    private static void removeDuplicates(int[] numbers) {
28        for (int i = 0; i < numbers.length; i++) {
29            for (int j = i + 1; j < numbers.length; j++) {
30                if (numbers[i] == numbers[j]) {
31                    numbers[j] = -1; // Set duplicate value to -1
32                }
33            }
34        }
35    }
36
37    private static void displayArray(int[] array) {
38        System.out.println("Array without duplicate values:");
39
40        for (int value : array) {
41            if (value != -1) {
42                System.out.print(value + " ");
43            }
44        }
45    }
46 }
```

7.13 Label the elements of a five-by-six two-dimensional array table to indicate the order in which they're set to zero by the following program segment:

```
1  for (int col = 0; col < 6; col++)
2  {
3        for (int row = 0; row < 5; row++)
4        {
5                table[row][col] = 0;
6        }
7  }
```

1	6	11	16	21	26
2	7	12	17	22	27
3	8	13	18	23	28
4	9	14	19	24	29

5	10	15	20	25	30