Week 1 - S1 – Assignment HW

Name: Ramesh Harisabapathi Chettiar

Roll Number: RA2411030010263

Course: Networking and Communications

Semester: 3

Date of Submission: 19/08/2025

Lab Practice Programs (Any Six)

1. An organization took up the exercise to find the Body Mass Index (BMI) of all the persons in a team of 10 members. For this create a program to find the BMI and display the height, weight, BMI, and status of each individual

Hint =>

a. Take user input for the person's weight (kg) and height (cm) and store it in the corresponding 2D array of 10 rows. The First Column stores the weight and the second column stores the height in cm

b. Create a Method to find the BMI and status of every person given the person's height and weight and return the 2D String array. Use the formula BMI = weight / (height * height). Note unit is kg/m^2. For this convert cm to meter

c. Create a Method that takes the 2D array of height and weight as parameters. Calls the user-defined method to compute the BMI and the BMI Status and stores in a 2D String array of height, weight, BMI, and status.

d. Create a method to display the 2D string array in a tabular format of Person's Height, Weight, BMI, and the Status

e. Finally, the main function takes user inputs, calls the user-defined methods, and displays the result.

```java
import java.util.Scanner;

public class BMI {
    static Scanner input = new Scanner(System.in);

    //Function to take Height and Weight as Input
    public static double[][] weightHeight() {
        double[][] stats = new double[10][2];
        for (int i = 0; i < 10; i++) {
            System.out.println("Person " + (i + 1) + " :");
            System.out.print("Weight of Person (in kgs): ");
            stats[i][0] = input.nextDouble();
            System.out.print("Height of Person (in cms): ");
            stats[i][1] = input.nextDouble();
        }
        return stats;
    }

    //Function to store all 4 values
    public static String[][] Statistics(double[][] stats) {
        String[][] data = new String[10][4];
        for (int i = 0; i < 10; i++) {
            double heightM = stats[i][1] / 100;
            double BMI = stats[i][0] / (Math.pow(heightM, 2));

            data[i][0] = String.valueOf(stats[i][1]);              // height
            data[i][1] = String.valueOf(stats[i][0]);              // weight
            data[i][2] = String.format("%.2f", BMI);               // BMI
            if (BMI > 0 && BMI <= 18.4) {
                data[i][3] = "Underweight";
            } else if (BMI > 18.4 && BMI <= 24.9) {
                data[i][3] = "Normal";
            } else if (BMI > 24.9 && BMI <= 39.9) {
                data[i][3] = "Overweight";
            } else if (BMI < 0) {
                data[i][3] = "Invalid BMI";
            } else {
                data[i][3] = "Obese";
            }
        }
        return data;
    }
```

```
47      //Function to display Details
48      public static void displayDetails(String[][] data) {
49          System.out.println("\n==================== DETAILS ====================");
50          System.out.printf("%-12s %-12s %-12s %-15s\n", "Height(cms)", "Weight(kgs)", "BMI", "Status");
51          System.out.println("-----------------------------------------------------------");
52          for (int i = 0; i < 10; i++) {
53              System.out.printf("%-12s %-12s %-12s %-15s\n",
54                      data[i][0], data[i][1], data[i][2], data[i][3]);
55          }
56      }
57
58      //Main()
59      public static void main(String[] args) {
60          double[][] HeightWeight = weightHeight();
61          String[][] DATA = Statistics(HeightWeight);
62          displayDetails(DATA);
63          input.close();
64      }
65  }
66
```

2. Find unique characters in a string using the charAt() method and display the result

Hint =>

a. Create a Method to find the length of the text without using the String method length()

b. Create a method to Find unique characters in a string using the charAt() method and return them as a 1D array. The logic used here is as follows:

i. Create an array to store the unique characters in the text. The size is the length of the text

ii. Loops to Find the unique characters in the text. Find the unique characters in the text using a nested loop. An outer loop iterates through each character and an inner loop checks if the character is unique by comparing it with the previous characters. If the character is unique, it is stored in the result array

iii. Create a new array to store the unique characters

```java
/*2. Find unique characters in a string using the charAt() method and display the result
Hint =>
a. Create a Method to find the length of the text without using the String method length()
b. Create a method to Find unique characters in a string using the charAt() method and
return them as a 1D array. The logic used here is as follows:
i. Create an array to store the unique characters in the text. The size is the length of
the text
ii. Loops to Find the unique characters in the text. Find the unique characters in the text
using a nested loop. An outer loop iterates through each character and an inner loop
checks if the character is unique by comparing it with the previous characters. If the
character is unique, it is stored in the result array
iii. Create a new array to store the unique characters */

import java.util.Scanner;

public class UniqueCharacters{

    //a. Create a Method to find the length of the text without using the String method length()
    public static int Length(String text) {
        int count = 0;
        try {
            while (true) {
                text.charAt(count); // Throws exception when index is out of bounds
                count++;
            }
        } catch (IndexOutOfBoundsException e) {
            // End of string reached
        }
        return count;
    }

    //Function to count the number of unique characters in the String
    public static char[] uniqueChar(String text) {
        char[] unique = new char[Length(text)]; // temporary array
        int size = 0; // count of unique chars

        for (int i = 0; i < Length(text); i++) {
            char ch = text.charAt(i);
            boolean found = false;

            // checking if character is already in unique[]
            for (int j = 0; j < size; j++) {
                if (ch == unique[j]) {
                    found = true;
                    break;
                }
            }

            // if the character is not found in the array, add to unique[] array
            if (!found) {
                unique[size] = ch;
                size++;
            }
        }

        // creating final array(result) with only unique chars
        char[] result = new char[size];
        for (int i = 0; i < size; i++) {
            result[i] = unique[i];
        }
        return result;
    }
```

```java
    //Print Function
    public static void displayDetails(String text,char[] result){
        System.out.println("Entered text--->" + text);
        System.out.println("Unique Characters in the above text-->");
        for (int i = 0 ; i < Length(new String(result)); i++){
            System.out.println("Character "+(i+1)+" -->"+result[i]);
        }
        System.out.println("Out of " + Length(text) + " ,there is/are only " +Length(new String(result)) + " Unique Characters.");
    }
    //Main
    public static void main(String args[]){
        Scanner input = new Scanner(System.in);

        System.out.println("Enter text-->");
        String text = input.nextLine();

        char uniqueCharacters[] = uniqueChar(text.trim());

        displayDetails(text,uniqueCharacters);

    }
}
```

OUTPUT-

```
Enter text-->
              Ramesh Harisabapathi Chettiar
Entered text--->              Ramesh Harisabapathi Chettiar
Unique Characters in the above text-->
Character 1 --> R
Character 2 --> a
Character 3 --> m
Character 4 --> e
Character 5 --> s
Character 6 --> h
Character 7 -->
Character 8 --> H
Character 9 --> r
Character 10 --> i
Character 11 --> b
Character 12 --> p
Character 13 --> t
Character 14 --> C
Out of 60 ,there is/are only 14 Unique Characters.
```

3. Write a program to find the first non-repeating character in a string and show the result

Hint =>

a. Non-repeating character is a character that occurs only once in the string

b. Create a Method to find the first non-repeating character in a string using the charAt() method and return the character. The logic used here is as follows:

i. Create an array to store the frequency of characters in the text. ASCII values of characters are used as indexes in the array to store the frequency of each character. There are 256 ASCII characters

ii. Loop through the text to find the frequency of characters in the text

iii. Loop through the text to find the first non-repeating character in the text by checking the frequency of each character

c. In the main function take user inputs, call user-defined methods, and displays result.

```java
import java.util.Scanner;

public class NonRepeating {
    // Custom method to calculate string length without using .length()
    public static int Length(String text) {
        int count = 0;
        try {
            while (true) {
                text.charAt(count); // Throws exception when index is out of bounds
                count++;
            }
        } catch (IndexOutOfBoundsException e) {
            // End of string reached
        }
        return count;
    }

    // Method to find the first non-repeating character
    public static char firstNonRepeatingChar(String text) {
        int[] freq = new int[256]; // Frequency array for ASCII characters

        // First pass: count frequency of each character
        for (int i = 0; i < text.length(); i++) {
            freq[text.charAt(i)]++;
        }

        // Second pass: find the first character with frequency 1
        for (int i = 0; i < text.length(); i++) {
            if (freq[text.charAt(i)] == 1) {
                return text.charAt(i);
            }
        }
        return '\0'; // Return null character if no non-repeating character found
    }

    // Main method
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        System.out.print("Enter Text --> ");
        String text = input.nextLine().trim();

        char result = firstNonRepeatingChar(text);

        if (result != '\0') {
            System.out.println("\nFirst Non-Repeating Character of the String --> " + result);
        } else {
            System.out.println("There are no Non-Repeating Characters in the String.");
        }

        input.close();
    }
}

```

OUTPUT→

```
● LANEOUS\ENTRANCE EXAMS\SRM\SEMESTERS\SEMESTER-3\JAVA-STEP\Weeks\Week 1\Homework\" ; if ($?) { javac NonRepeating.java } ; if ($?) { java NonRepeating }
Enter Text --> JAVA PROGRAMMING

First Non-Repeating Character of the String --> J
```

4. Write a program to find the frequency of characters in a string using the charAt() method and display the result

Hint =>

a. Create a method to find the frequency of characters in a string using the charAt() method and return the characters and their frequencies in a 2D array. The logic used here is as follows:

i. Create an array to store the frequency of characters in the text. ASCII values of characters are used as indexes in the array to store the frequency of each character. There are 256 ASCII characters

ii. Loop through the text to find the frequency of characters in the text

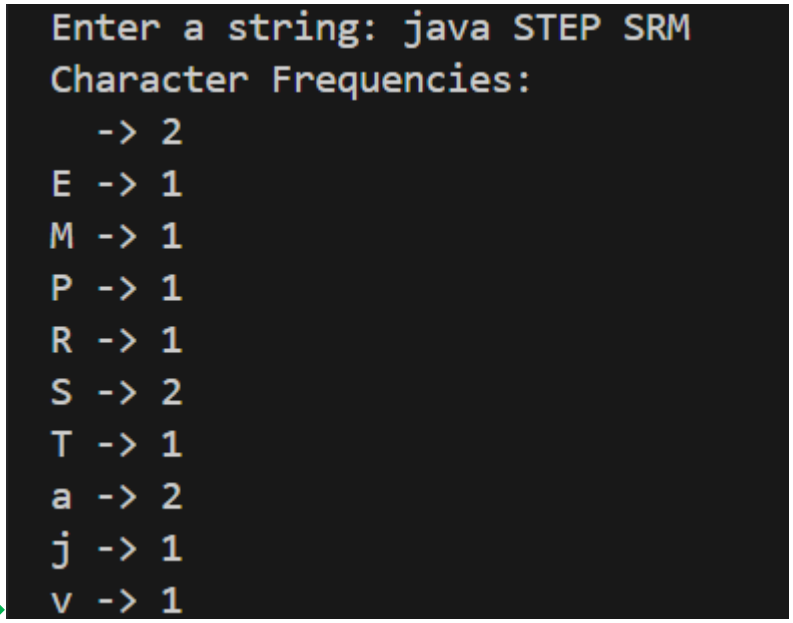
iii. Create an array to store the characters and their frequencies

iv. Loop through the characters in the text and store the characters and their frequencies

b. In the main function take user inputs, call user-defined methods, and displays result.

```java
import java.util.Scanner;

public class NonRepeatingCharacters2 {

    /*a. Create a method to find the frequency of characters in a string using the charAt() method
    and return the characters and their frequencies in a 2D array. The logic used here is as
    follows:
    i. Create an array to store the frequency of characters in the text. ASCII values of
    characters are used as indexes in the array to store the frequency of each character.
    There are 256 ASCII characters
    ii. Loop through the text to find the frequency of characters in the text
    iii. Create an array to store the characters and their frequencies
    iv. Loop through the characters in the text and store the characters and their
    frequencies */
    public static int[][] CharacterFrequencies(String text) {
        int[] freq = new int[256]; // i. Array to store frequency of ASCII characters

        // ii. Loop through the text to count frequency using ASCII values as indexes
        for (int i = 0; i < text.length(); i++) {
            char c = text.charAt(i);
            freq[c]++;
        }

        // Counting how many unique characters are present
        int uniqueCount = 0;
        for (int i = 0; i < 256; i++) {
            if (freq[i] > 0) {
                uniqueCount++;
            }
        }

        // iii. Create a 2D array to store characters and their frequencies
        int[][] result = new int[uniqueCount][2];
        int index = 0;

        // iv. Loop through characters and store ASCII value and their frequencies
        for (int i = 0; i < 256; i++) {
            if (freq[i] > 0) {
                result[index][0] = i;
                result[index][1] = freq[i];
                index++;
            }
        }

        return result;
    }

    /*b. Main(). */
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        System.out.print("Enter a string: ");
        String text = input.nextLine();

        int[][] frequencies = CharacterFrequencies(text);

        // Display Details
        System.out.println("Character Frequencies:");
        for (int i = 0; i < frequencies.length; i++) {
            System.out.println((char)frequencies[i][0] + " -> " + frequencies[i][1]);
        }

        input.close();
    }
}
```

OUTPUT→


Enter a string: java STEP SRM
Character Frequencies:
-> 2
E -> 1
M -> 1
P -> 1
R -> 1
S -> 2
T -> 1
a -> 2
j -> 1
v -> 1

7. Write a program to to check if a text is palindrome and display the result

Hint =>

a. A palindrome is a word, phrase, number, or other sequence of characters that reads the same forward and backward

b. Logic 1: Write a method to compare the characters from the start and end of the string to determine whether the text is palindrome. The logic used here is as follows:

i. Set the start and end indexes of the text

ii. Loop through the text and compare the characters from the start and the end of the string. If the characters are not equal, return false

c. Logic 2: Write a recursive method to compare the characters from the start and end of the text passed as parameters using recursion. The logic used here is as follows:

i. First, check if the start index is greater than or equal to the end index, then return true.

ii. If the characters at the start and end indexes are not equal, return false.

iii. Otherwise, call the method recursively with the start index incremented by 1 and the end index

d. Logic 3: Write a Method to compare the characters from the start and end of the text using character arrays. The logic used here is as follows:

i. Firstly Write a Method to reverse a string using the charAt() method and return the reversal array.

ii. Create a character array using the String method toCharArray() and also create a reverse array. Compare the characters in the original and reverse arrays to do a Palindrome check

e. Finally, in the main method do palindrome check using the three logic and display result

```java
import java.util.Scanner;

public class Pallindrome {

    // Custom method to calculate string length without using .length()
    public static int Length(String text) {
        int count = 0;
        try {
            while (true) {
                text.charAt(count); // Throws exception when index is out of bounds
                count++;
            }
        } catch (IndexOutOfBoundsException e) {
            // End of string reached
        }
        return count;
    }

    /* Logic 1: Write a method to compare the characters from the start and end of the string
    to determine whether the text is palindrome. The logic used here is as follows:
    i. Set the start and end indexes of the text
    ii. Loop through the text and compare the characters from the start and the end of the
    string. If the characters are not equal, return false */
    public static boolean Logic1(String text) {
        int start = 0;
        int end = Length(text) - 1;

        while (start <= end) {
            if (text.charAt(start) != text.charAt(end)) {
                return false;
            }
            start++;
            end--;
        }
```

```
        return true;
    }

    /* c. Logic 2: Write a recursive method to compare the characters from the start and end of
    the text passed as parameters using recursion. The logic used here is as follows:
    i. First, check if the start index is greater than or equal to the end index, then return
    true.
    ii. If the characters at the start and end indexes are not equal, return false.
    iii. Otherwise, call the method recursively with the start index incremented by 1 and the
    end index decremented by 1 */
    public static boolean Logic2(String text, int start, int end) {
        if (start >= end) {
            return true;
        }
        if (text.charAt(start) != text.charAt(end)) {
            return false;
        }
        return Logic2(text, start + 1, end - 1);
    }

    /* d. Logic 3: Write a Method to compare the characters from the start and end of the text
    using character arrays. The logic used here is as follows:
    i. Firstly Write a Method to reverse a string using the charAt() method and return the
    reversal array.
    ii. Create a character array using the String method toCharArray() and also create a
    reverse array. Compare the characters in the original and reverse arrays to do a
    Palindrome check */
    public static char[] arrayReversal(String text) {
        int end = Length(text) - 1;
        char[] characters = new char[Length(text)];
        int index = 0;

        while (end >= 0) {
            characters[index] = text.charAt(end);
            end--;
            index++;
        }
        return characters;
    }

    public static boolean Logic3(String text) {
        char[] reversedArray = arrayReversal(text);
        char[] array = text.toCharArray();

        for (int i = 0; i < Length(text); i++) {
            if (reversedArray[i] != array[i]) {
                return false;
            }
        }
        return true;
    }


    public static void displayDetails(String text) {
        System.out.println("\n----------------------DETAILS----------------------");
        System.out.println(text + " is a Palindrome.");
    }

    //Main()
    public static void main(String args[]) {
        Scanner input = new Scanner(System.in);
        System.out.println("Enter Text---------->");
        String text = input.nextLine();
```

```
        System.out.println("Which Logic do you want to check if the entered Text is Palindrome or not:"
        System.out.println("Logic 1(1), Logic 2(2) and Logic 3(3)");
        int logic = input.nextInt();

        switch (logic) {
            case 1:
                if (Logic1(text.trim())) {
                    displayDetails(text.trim());
                } else {
                    System.out.println(text + " is not a Palindrome.");
                }
                break;

            case 2:
                int start = 0;
                int end = Length(text.trim()) - 1;
                if (Logic2(text.trim(), start, end)) {
                    displayDetails(text.trim());
                } else {
                    System.out.println(text + " is not a Palindrome.");
                }
                break;

            case 3:
                if (Logic3(text.trim())) {
                    displayDetails(text.trim());
                } else {
                    System.out.println(text + " is not a Palindrome.");
                }
                break;

              default:
                 System.out.println("No Such Logic! Please Enter the Correct Choice!!");
          }

          input.close();
      }
  }
```

OUTPUT→

```
Enter Text---------->
MALAYALAM
Which Logic do you want to check if the entered Text is Palindrome or not:
MALAYALAM
Which Logic do you want to check if the entered Text is Palindrome or not:
Which Logic do you want to check if the entered Text is Palindrome or not:
Logic 1(1), Logic 2(2) and Logic 3(3)
Logic 1(1), Logic 2(2) and Logic 3(3)
2

----------------------DETAILS----------------------
MALAYALAM is a Palindrome.
```

8. Write a program to check if two texts are anagrams and display the result

Hint =>

a. An anagram is a word or phrase formed by rearranging the same letters to form different words or phrases,

b. Write a method to check if two texts are anagrams. The logic used here is as follows:

i. Check if the lengths of the two texts are equal

ii. Create an array to store the frequency of characters in the strings for the two text

iii. Find the frequency of characters in the two texts using the loop

iv. Compare the frequency of characters in the two texts. If the frequencies are not equal, return false

c. In the main function take user inputs, call user-defined methods, and displays result.

```java
/*8. Write a program to check if two texts are anagrams and display the result
Hint =>
a. An anagram is a word or phrase formed by rearranging the same letters to form different
words or phrases,
b. Write a method to check if two texts are anagrams. The logic used here is as follows:
i. Check if the lengths of the two texts are equal
ii. Create an array to store the frequency of characters in the strings for the two text
iii. Find the frequency of characters in the two texts using the loop
iv. Compare the frequency of characters in the two texts. If the frequencies are not
equal, return false
c. In the main function take user inputs, call user-defined methods, and displays result. */
import java.util.Scanner;

public class Anagram{

    // Custom method to calculate string length without using .length()
    public static int Length(String text) {
        int count = 0;
        try {
            while (true) {
                text.charAt(count); // Throws exception when index is out of bounds
                count++;
            }
        } catch (IndexOutOfBoundsException e) {
            // End of string reached
        }
        return count;
    }

    /*b. Write a method to check if two texts are anagrams. The logic used here is as follows:
    i. Check if the lengths of the two texts are equal
    ii. Create an array to store the frequency of characters in the strings for the two text
    iii. Find the frequency of characters in the two texts using the loop
    iv. Compare the frequency of characters in the two texts. If the frequencies are not
    equal, return false */
    public static boolean isAnagram(String text1,String text2){
        // i. Check if the lengths of the two texts are equal
        if (Length(text1) != Length(text2)){
            return false;
        }

        // ii. Create an array to store the frequency of characters in the strings for the two text
        int[] freq1 = new int[256];
        int[] freq2 = new int[256];

        // iii. Find the frequency of characters in the two texts using the loop
        for (int i = 0; i < Length(text1); i++){
            freq1[text1.charAt(i)]++;
            freq2[text2.charAt(i)]++;
        }

        // iv. Compare the frequency of characters in the two texts
        for (int i = 0; i < 256; i++){
            if (freq1[i] != freq2[i]){
                return false;
            }
```

```java
        }
        return true;
    }

    // Method to display result
    public static void displayDetails(String text1,String text2){
        if(isAnagram(text1,text2)){
            System.out.println(text1 + " and " + text2 + " are Anagrams to each other");
        }
        else{
            System.out.println(text1 + " and " + text2 + " are not Anagrams to each other");
        }
    }

    // c. In the main function take user inputs, call user-defined methods, and displays result
    public static void main(String args[]){
        Scanner input = new Scanner (System.in);

        System.out.println("Enter text1-->");
        String text1 = input.nextLine();

        System.out.println("Enter text2-->");
        String text2 = input.nextLine();

        displayDetails(text1.trim(),text2.trim());
    }
```

OUTPUT→

```
Enter text1-->
GOD
GOD
Enter text2-->
Enter text2-->
DOG
GOD and DOG are Anagrams to each other
```