Step Lab Work-Week-1

Q1) // Write a program to find vowels and consonants in a string and display the character type -

// Vowel, Consonant, or Not a Letter

// Hint =>

// a. Create a method to check if the character is a vowel or consonant and return the result.

// The logic used here is as follows:

// i. Convert the character to lowercase if it is an uppercase letter using the ASCII values

// of the characters

// ii. Check if the character is a vowel or consonant and return Vowel, Consonant, or Not

// a Letter

// b. Create a Method to find vowels and consonants in a string using charAt() method and

// return the character and vowel or consonant in a 2D array

// c. Create a Method to display the 2D Array of Strings in a Tabular Format

// d. Finally, the main function takes user inputs, calls the user-defined methods, and displays

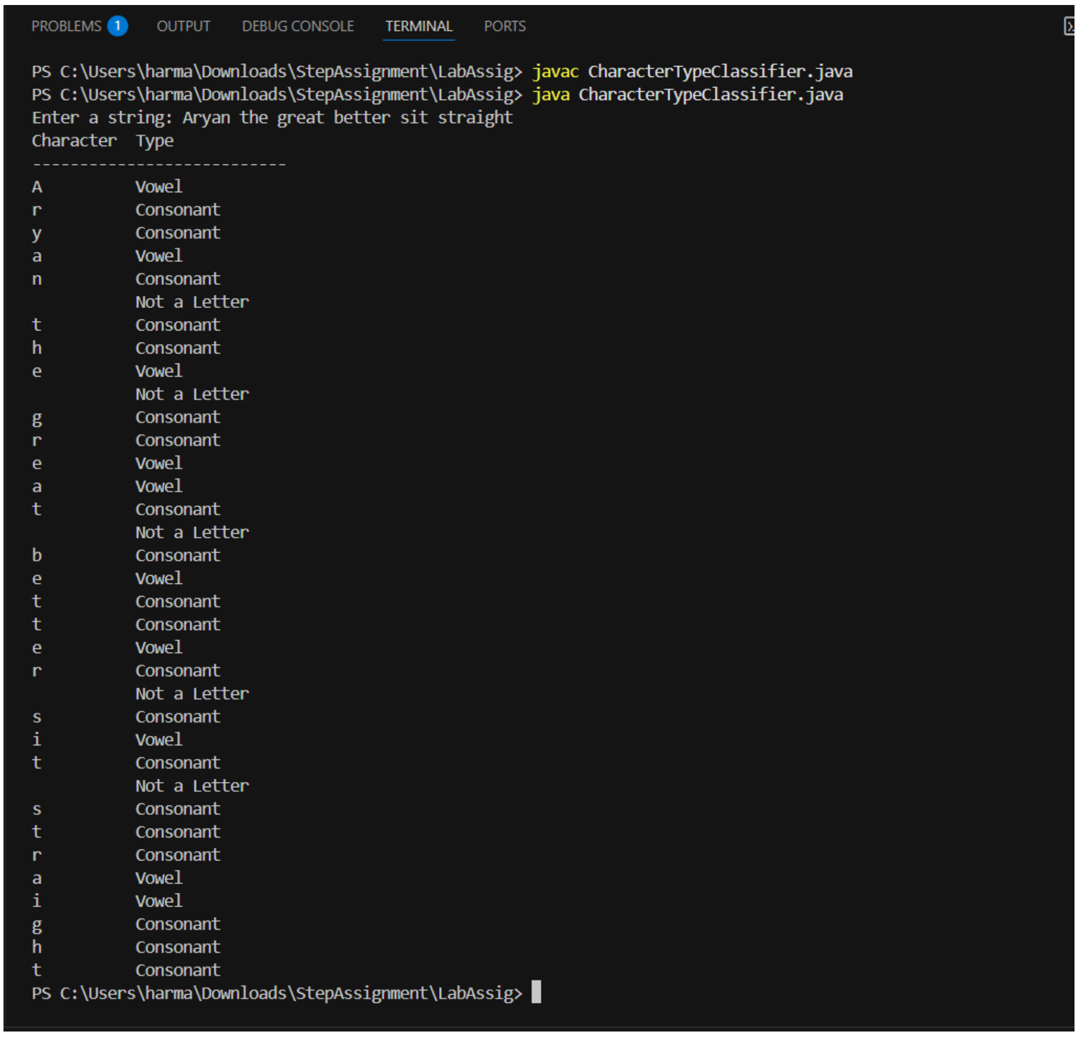
// the result.

<u>Source Code</u>:

```
import java.util.*;
```

```
public class CharacterTypeClassifier {
    // Method to check if a character is vowel, consonant, or not a letter
    public static String checkCharType(char ch) {
        if (ch >= 'A' && ch <= 'Z') {
            ch = (char)(ch + 32);
        }
        if (ch >= 'a' && ch <= 'z') {
            if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u') {
                return "Vowel";
            } else {
                return "Consonant";
            }
        } else {
            return "Not a Letter";
        }
    }

    // Method to build 2D array of character and its type
    public static String[][] classifyCharacters(String text) {
        int length = 0;
        try {
            while (true) {
                text.charAt(length);
                length++;
            }
        } catch (IndexOutOfBoundsException e) {
            // End of string
        }
        String[][] result = new String[length][2];
        for (int i = 0; i < length; i++) {
            char ch = text.charAt(i);
            result[i][0] = String.valueOf(ch);
            result[i][1] = checkCharType(ch);
        }
        return result;
    }

    // Method to display 2D array in tabular format
    public static void displayTable(String[][] data) {
        System.out.printf("%-10s %-15s\n", "Character", "Type");
        System.out.println("--------------------------");
        for (String[] row : data) {
            System.out.printf("%-10s %-15s\n", row[0], row[1]);
        }
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a string: ");
        String input = sc.nextLine();

        String[][] characterData = classifyCharacters(input);
        displayTable(characterData);
        sc.close();
    }
}
```

Output:



PROBLEMS 1   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

```
PS C:\Users\harma\Downloads\StepAssignment\LabAssig> javac CharacterTypeClassifier.java
PS C:\Users\harma\Downloads\StepAssignment\LabAssig> java CharacterTypeClassifier.java
Enter a string: Aryan the great better sit straight
Character  Type
---------------------------
A          Vowel
r          Consonant
y          Consonant
a          Vowel
n          Consonant
           Not a Letter
t          Consonant
h          Consonant
e          Vowel
           Not a Letter
g          Consonant
r          Consonant
e          Vowel
a          Vowel
t          Consonant
           Not a Letter
b          Consonant
e          Vowel
t          Consonant
t          Consonant
e          Vowel
r          Consonant
           Not a Letter
s          Consonant
i          Vowel
t          Consonant
           Not a Letter
s          Consonant
t          Consonant
r          Consonant
a          Vowel
i          Vowel
g          Consonant
h          Consonant
t          Consonant
PS C:\Users\harma\Downloads\StepAssignment\LabAssig> 
```

Q2) // Write a program to split the text into words and find the shortest and longest strings in a

// given text

// Hint =>

// a. Take user input using the Scanner nextLine() method

// b. Create a Method to split the text into words using the charAt() method without using the

// String built-in split() method and return the words.

// c. Create a method to find and return a string's length without using the length() method.

// d. Create a method to take the word array and return a 2D String array of the word and its

// corresponding length. Use String built-in function String.valueOf() to generate the String

// value for the number

// e. Create a Method that takes the 2D array of word and corresponding length as

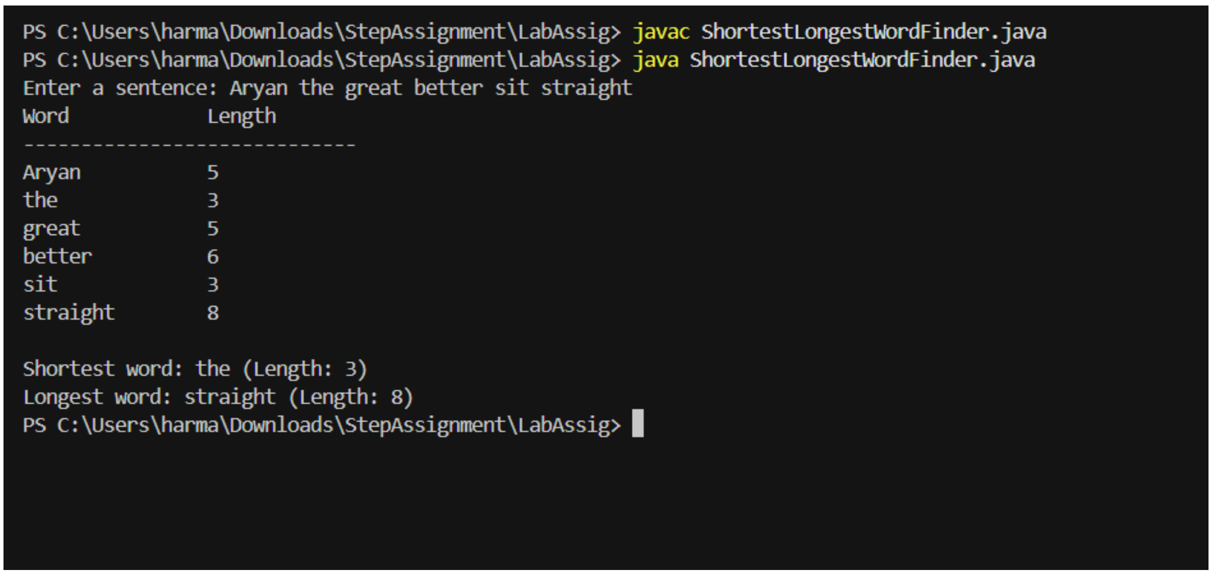
// parameters, find the shortest and longest string and return them in an 1D int array.

// f. The main function calls the user-defined methods and displays the result.

```
import java.util.*;


public class ShortestLongestWordFinder {
    // Method to find length of a string without using length()
```

```
public static int getLength(String str) {
    int count = 0;
    try {
        while (true) {
            str.charAt(count);
            count++;
        }
    } catch (IndexOutOfBoundsException e) {
        // End of string
    }
    return count;
}

// Method to split text into words without using split()
public static String[] customSplit(String str) {
    List<String> words = new ArrayList<>();
    int len = getLength(str);
    int start = 0;
    for (int i = 0; i <= len; i++) {
        if (i == len || str.charAt(i) == ' ') {
            if (start < i) {
                words.add(str.substring(start, i));
            }
            start = i + 1;
        }
    }
    return words.toArray(new String[0]);
}

// Method to build 2D String array of word and its length
public static String[][] buildWordLengthTable(String[] words) {
    String[][] table = new String[words.length][2];
    for (int i = 0; i < words.length; i++) {
        table[i][0] = words[i];
        table[i][1] = String.valueOf(getLength(words[i]));
    }
    return table;
}

// Method to find shortest and longest word indices
public static int[] findShortestLongest(String[][] table) {
    int minIndex = 0, maxIndex = 0;
    int minLength = Integer.parseInt(table[0][1]);
    int maxLength = Integer.parseInt(table[0][1]);
    for (int i = 1; i < table.length; i++) {
        int length = Integer.parseInt(table[i][1]);
        if (length < minLength) {
            minLength = length;
            minIndex = i;
        }
        if (length > maxLength) {
            maxLength = length;
            maxIndex = i;
        }
    }
    return new int[]{minIndex, maxIndex};
}

public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    System.out.print("Enter a sentence: ");
    String input = sc.nextLine();

    String[] words = customSplit(input);
    String[][] table = buildWordLengthTable(words);
    int[] result = findShortestLongest(table);

    System.out.printf("%-15s %-10s\n", "Word", "Length");
    System.out.println("-----------------------------");
    for (int i = 0; i < table.length; i++) {
        String word = table[i][0];
        int length = Integer.parseInt(table[i][1]);
```

```
            System.out.printf("%-15s %-10d\n", word, length);
        }

        System.out.println("\nShortest word: " + table[result[0]][0] + " (Length: " + table[result[0]][1] + ")");
        System.out.println("Longest word: " + table[result[1]][0] + " (Length: " + table[result[1]][1] + ")");
        sc.close();
    }
}
```

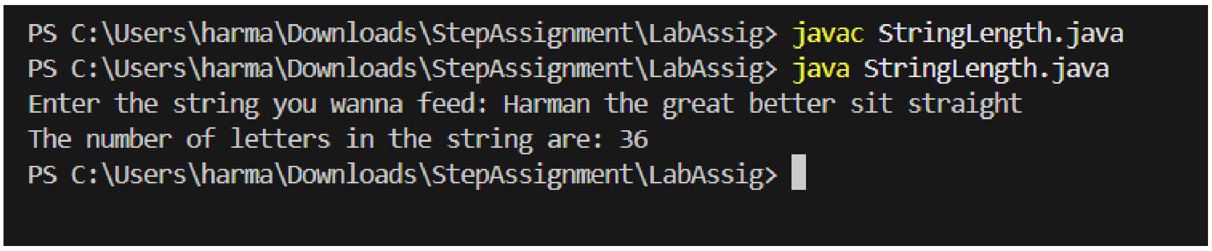


```
PS C:\Users\harma\Downloads\StepAssignment\LabAssig> javac ShortestLongestWordFinder.java
PS C:\Users\harma\Downloads\StepAssignment\LabAssig> java ShortestLongestWordFinder.java
Enter a sentence: Aryan the great better sit straight
Word            Length
----------------------------
Aryan           5
the             3
great           5
better          6
sit             3
straight        8

Shortest word: the (Length: 3)
Longest word: straight (Length: 8)
PS C:\Users\harma\Downloads\StepAssignment\LabAssig>
```

Q3)

```
import java.util.*;

public class StringLength {
    public static int countLength(String str) {
        int i = 0;
        try {
            while (true) {
                str.charAt(i);
                i++;
            }
        } catch (StringIndexOutOfBoundsException e) {
            // This exception is expected and used to know we've reached
the end.
        }
        return i;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the string you wanna feed: ");
        String str = sc.nextLine();

        int length = countLength(str);
        System.out.println("The number of letters in the string are: " +
length);
        sc.close();
    }
```

}



```
PS C:\Users\harma\Downloads\StepAssignment\LabAssig> javac StringLength.java
PS C:\Users\harma\Downloads\StepAssignment\LabAssig> java StringLength.java
Enter the string you wanna feed: Harman the great better sit straight
The number of letters in the string are: 36
PS C:\Users\harma\Downloads\StepAssignment\LabAssig>
```

Q4) // Write a program to split the text into words, compare the result with the split() method and

// display the result

// Hint =>

// a. Take user input using the Scanner nextLine() method

// b. Create a Method to find the length of the String without using the built-in length() method.

// c. Create a Method to split the text into words using the charAt() method without using the

// String built-in split() method and return the words. Use the following logic

// i. Firstly Count the number of words in the text and create an array to store the

// indexes of the spaces for each word in a 1D array

// ii. Then Create an array to store the words and use the indexes to extract the words

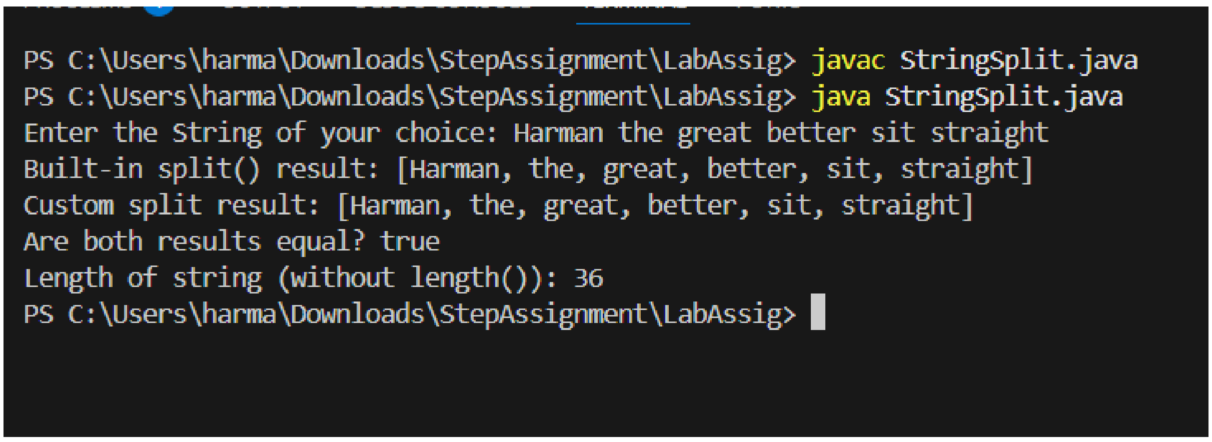
// d. Create a method to compare the two String arrays and return a boolean

// e. The main function calls the user-defined method and the built-in split() method. Call the

// user defined method to compare the two string arrays and display the result

```
import java.util.*;

public class StringSplit {
    // Method to find the length of the String without using length()
    public static int getLength(String str) {
        int count = 0;
        try {
            while (true) {
                str.charAt(count);
                count++;
            }
        } catch (IndexOutOfBoundsException e) {
            // End of string reached
        }
        return count;
    }

    // Method to split the text into words without using split()
    public static String[] customSplit(String str) {
        List<String> words = new ArrayList<>();
        int len = getLength(str);
        int start = 0;
        for (int i = 0; i <= len; i++) {
            if (i == len || str.charAt(i) == ' ') {
                if (start < i) {
                    words.add(str.substring(start, i));
                }
                start = i + 1;
            }
        }
        return words.toArray(new String[0]);
    }

    // Method to compare two String arrays
    public static boolean compareArrays(String[] arr1, String[] arr2) {
        if (arr1.length != arr2.length) return false;
        for (int i = 0; i < arr1.length; i++) {
            if (!arr1[i].equals(arr2[i])) return false;
        }
        return true;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
```

```
        System.out.print("Enter the String of your choice: ");
        String str = sc.nextLine();

        // Using built-in split()
        String[] builtInSplit = str.split(" ");

        // Using custom split
        String[] customSplitResult = customSplit(str);

        // Compare the two arrays
        boolean areEqual = compareArrays(builtInSplit,
customSplitResult);

        // Display results
        System.out.println("Built-in split() result: " +
Arrays.toString(builtInSplit));
        System.out.println("Custom split result: " +
Arrays.toString(customSplitResult));
        System.out.println("Are both results equal? " + areEqual);
        System.out.println("Length of string (without length()): " +
getLength(str));
    }
}
```



```
PS C:\Users\harma\Downloads\StepAssignment\LabAssig> javac StringSplit.java
PS C:\Users\harma\Downloads\StepAssignment\LabAssig> java StringSplit.java
Enter the String of your choice: Harman the great better sit straight
Built-in split() result: [Harman, the, great, better, sit, straight]
Custom split result: [Harman, the, great, better, sit, straight]
Are both results equal? true
Length of string (without length()): 36
PS C:\Users\harma\Downloads\StepAssignment\LabAssig>
```

Q5) // Write a program to find vowels and consonants in a string and display the count of Vowels

// and Consonants in the string

// Hint =>

// a. Create a method to check if the character is a vowel or consonant and return the result.

// The logic used here is as follows:

// 3

// i. Convert the character to lowercase if it is an uppercase letter using the ASCII values

// of the characters

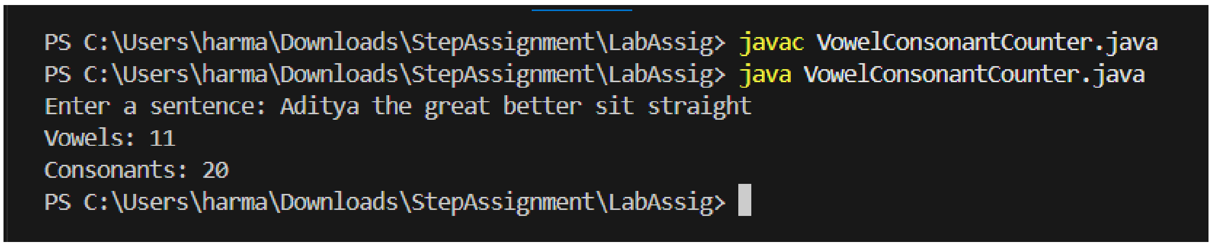
// ii. Check if the character is a vowel or consonant and return Vowel, Consonant, or Not

// a Letter

// b. Create a Method to Method to find vowels and consonants in a string using charAt()

```
// method and finally return the count of vowels and consonants in an
array
// c. Finally, the main function takes user inputs, calls the user-defined
methods, and displays
// the result.

import java.util.*;

public class VowelConsonantCounter {
    // Method to check if a character is vowel, consonant, or not a letter
    public static String checkCharType(char ch) {
        // Convert to lowercase if uppercase
        if (ch >= 'A' && ch <= 'Z') {
            ch = (char)(ch + 32);
        }
        if (ch >= 'a' && ch <= 'z') {
            if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u') {
                return "Vowel";
            } else {
                return "Consonant";
            }
        } else {
            return "Not a Letter";
        }
```

```java
}

// Method to count vowels and consonants in a string
public static int[] countVowelsConsonants(String text) {
    int vowels = 0, consonants = 0;
    int i = 0;
    try {
        while (true) {
            char ch = text.charAt(i);
            String type = checkCharType(ch);
            if (type.equals("Vowel")) {
                vowels++;
            } else if (type.equals("Consonant")) {
                consonants++;
            }
            i++;
        }
    } catch (IndexOutOfBoundsException e) {
        // End of string
    }
    return new int[]{vowels, consonants};
}

public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a sentence: ");
        String input = sc.nextLine();

        int[] counts = countVowelsConsonants(input);
        System.out.println("Vowels: " + counts[0]);
        System.out.println("Consonants: " + counts[1]);
        sc.close();
    }
}
```



```
PS C:\Users\harma\Downloads\StepAssignment\LabAssig> javac VowelConsonantCounter.java
PS C:\Users\harma\Downloads\StepAssignment\LabAssig> java VowelConsonantCounter.java
Enter a sentence: Aditya the great better sit straight
Vowels: 11
Consonants: 20
PS C:\Users\harma\Downloads\StepAssignment\LabAssig>
```

Q6) // Write a program to split the text into words and return the words along with their lengths in a

// 2D array

// Hint =>

// a. Take user input using the Scanner nextLine() method

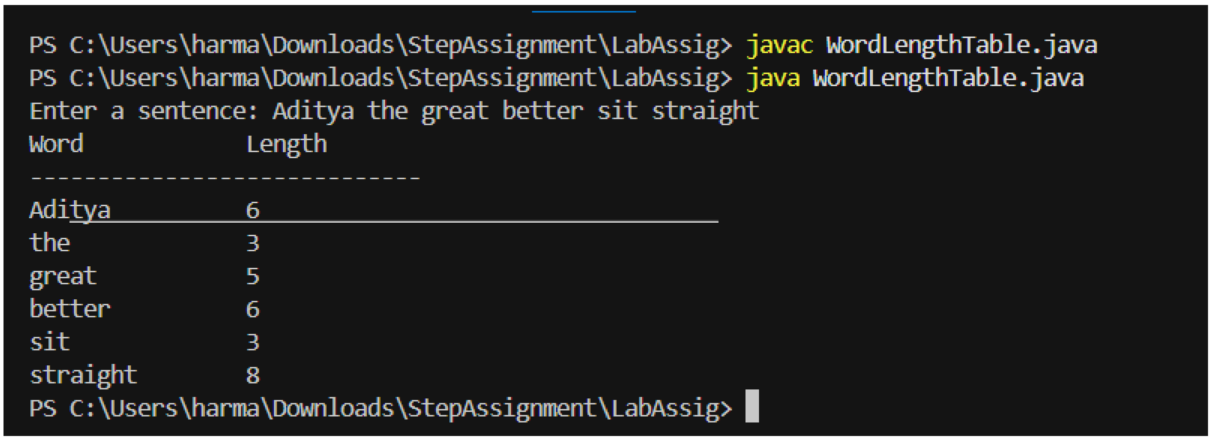
// b. Create a Method to split the text into words using the charAt() method without using the

// String built-in split() method and return the words.

// c. Create a method to find and return a string's length without using the length() method.

```
// d. Create a method to take the word array and return a 2D String
array of the word and its
// corresponding length. Use String built-in function String.valueOf()
to generate the String
// value for the number
// e. The main function calls the user-defined method and displays the
result in a tabular
// format. During display make sure to convert the length value from
String to Integer and
// then display

import java.util.*;

public class WordLengthTable {
    // Method to find length of a string without using length()
    public static int getLength(String str) {
        int count = 0;
        try {
            while (true) {
                str.charAt(count);
                count++;
            }
        } catch (IndexOutOfBoundsException e) {
            // End of string
        }
        return count;
    }

    // Method to split text into words without using split()
    public static String[] customSplit(String str) {
        List<String> words = new ArrayList<>();
        int len = getLength(str);
        int start = 0;
        for (int i = 0; i <= len; i++) {
            if (i == len || str.charAt(i) == ' ') {
                if (start < i) {
                    words.add(str.substring(start, i));
                }
                start = i + 1;
            }
        }
        return words.toArray(new String[0]);
    }

    // Method to build 2D String array of word and its length
    public static String[][] buildWordLengthTable(String[] words) {
        String[][] table = new String[words.length][2];
        for (int i = 0; i < words.length; i++) {
            table[i][0] = words[i];
```

```
            table[i][1] = String.valueOf(getLength(words[i]));
        }
        return table;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a sentence: ");
        String input = sc.nextLine();

        String[] words = customSplit(input);
        String[][] table = buildWordLengthTable(words);

        System.out.printf("%-15s %-10s\n", "Word", "Length");
        System.out.println("----------------------------");
        for (int i = 0; i < table.length; i++) {
            String word = table[i][0];
            int length = Integer.parseInt(table[i][1]);
            System.out.printf("%-15s %-10d\n", word, length);
        }
        sc.close();
    }
}
```



```
PS C:\Users\harma\Downloads\StepAssignment\LabAssig> javac WordLengthTable.java
PS C:\Users\harma\Downloads\StepAssignment\LabAssig> java WordLengthTable.java
Enter a sentence: Aditya the great better sit straight
Word            Length
------------------------------
Aditya          6
the             3
great           5
better          6
sit             3
straight        8
PS C:\Users\harma\Downloads\StepAssignment\LabAssig>
```