**1. Write a program to find and return the length of a string without using the length() method**

**Hint =>**

**a. Take user input using the Scanner next() method**

**b. Create a method to find and return a string's length without using the built-in length()**

**method. The logic for this is to use the infinite loop to count each character till the**

**charAt() method throws a runtime exception, handles the exception, and then return**

**the count**

**c. The main function calls the user-defined method as well as the built-in length() method**

**and displays the result**

import java.util.Scanner;

public class StringLengthFinder {

// Method to find length without using length()

public static int findLength(String str) {

int count = 0;

try {

while (true) { // infinite loop

str.charAt(count); // will throw exception when index out of range

count++;

}

} catch (StringIndexOutOfBoundsException e) {

// Exception occurs when we go beyond last character

}

return count;

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

// Take input

System.out.print("Enter a string: ");

String input = sc.next(); // takes single word input (as per question hint)

// Find length using custom method

int myLength = findLength(input);

// Find length using built-in method

int builtInLength = input.length();

// Display result

System.out.println("Length (user-defined method): " + myLength);

System.out.println("Length (built-in length() method): " + builtInLength);

sc.close();

}

}

**2. 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 CustomSplit {

// Method to find length without using length()

public static int findLength(String str) {

int count = 0;

try {

while (true) {

str.charAt(count); // throws exception when out of bounds

count++;

}

} catch (StringIndexOutOfBoundsException e) {

// exit loop

}

return count;

}

// Method to manually split text into words

public static String[] manualSplit(String text) {

int len = findLength(text);

// Step 1: Count words (spaces + 1)

int wordCount = 1;

for (int i = 0; i < len; i++) {

if (text.charAt(i) == ' ') {

wordCount++;

}

}

// Step 2: Store space indexes

int[] spaceIndexes = new int[wordCount - 1];

int idx = 0;

for (int i = 0; i < len; i++) {

if (text.charAt(i) == ' ') {

spaceIndexes[idx++] = i;

}

}

// Step 3: Extract words

String[] words = new String[wordCount];

int start = 0;

int w = 0;

for (int i = 0; i < spaceIndexes.length; i++) {

int end = spaceIndexes[i];

words[w++] = text.substring(start, end);

start = end + 1; // move after space

}

// last word

words[w] = text.substring(start, len);

return words;

}

// 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;

}

// MAIN

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

// Take input sentence

System.out.print("Enter a sentence: ");

String text = sc.nextLine();

// Custom split

String[] manualWords = manualSplit(text);

// Built-in split

String[] builtInWords = text.split(" ");

// Display results

System.out.println("\n--- Manual Split Result ---");

for (String w : manualWords) {

System.out.println(w);

}

System.out.println("\n--- Built-in Split Result ---");

for (String w : builtInWords) {

System.out.println(w);

}

// Compare arrays

boolean same = compareArrays(manualWords, builtInWords);

System.out.println("\nComparison Result: " + (same ? "Both methods match ✅" : "Mismatch ❌"));

sc.close();

}

}

**3. 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.Scanner;

public class WordsWithLengths {

// Method to find length without using length()

public static int findLength(String str) {

int count = 0;

try {

while (true) {

str.charAt(count); // throws exception when out of bounds

count++;

}

} catch (StringIndexOutOfBoundsException e) {

// stop when exception occurs

}

return count;

}

// Method to manually split string into words

public static String[] manualSplit(String text) {

int len = findLength(text);

// Count words (spaces + 1)

int wordCount = 1;

for (int i = 0; i < len; i++) {

if (text.charAt(i) == ' ') {

wordCount++;

}

}

// Store space indexes

int[] spaceIndexes = new int[wordCount - 1];

int idx = 0;

for (int i = 0; i < len; i++) {

if (text.charAt(i) == ' ') {

spaceIndexes[idx++] = i;

}

}

// Extract words

String[] words = new String[wordCount];

int start = 0, w = 0;

for (int i = 0; i < spaceIndexes.length; i++) {

int end = spaceIndexes[i];

words[w++] = text.substring(start, end);

start = end + 1;

}

// last word

words[w] = text.substring(start, len);

return words;

}

// Method to create 2D array [word, length]

public static String[][] wordsWithLengths(String[] words) {

String[][] result = new String[words.length][2];

for (int i = 0; i < words.length; i++) {

result[i][0] = words[i]; // word

int len = findLength(words[i]);

result[i][1] = String.valueOf(len); // store as String

}

return result;

}

// MAIN

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

// Input

System.out.print("Enter a sentence: ");

String text = sc.nextLine();

// Step 1: Split manually

String[] words = manualSplit(text);

// Step 2: Create 2D array

String[][] table = wordsWithLengths(words);

// Step 3: Display

System.out.println("\n--- Words and Their Lengths ---");

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 len = Integer.parseInt(table[i][1]); // convert back to int

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

}

sc.close();

}

}

**4. 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.Scanner;

public class ShortestLongestWord {

// Method to find length without using length()

public static int findLength(String str) {

int count = 0;

try {

while (true) {

str.charAt(count); // throws exception when out of bounds

count++;

}

} catch (StringIndexOutOfBoundsException e) {

// stop when exception occurs

}

return count;

}

// Method to manually split string into words

public static String[] manualSplit(String text) {

int len = findLength(text);

// Count words (spaces + 1)

int wordCount = 1;

for (int i = 0; i < len; i++) {

if (text.charAt(i) == ' ') {

wordCount++;

}

}

// Store space indexes

int[] spaceIndexes = new int[wordCount - 1];

int idx = 0;

for (int i = 0; i < len; i++) {

if (text.charAt(i) == ' ') {

spaceIndexes[idx++] = i;

}

}

// Extract words

String[] words = new String[wordCount];

int start = 0, w = 0;

for (int i = 0; i < spaceIndexes.length; i++) {

int end = spaceIndexes[i];

words[w++] = text.substring(start, end);

start = end + 1;

}

// last word

words[w] = text.substring(start, len);

return words;

}

// Method to create 2D array [word, length]

public static String[][] wordsWithLengths(String[] words) {

String[][] result = new String[words.length][2];

for (int i = 0; i < words.length; i++) {

result[i][0] = words[i]; // word

int len = findLength(words[i]);

result[i][1] = String.valueOf(len); // store as String

}

return result;

}

// Method to find shortest and longest words

// Returns a 1D array: [shortestLength, longestLength]

public static int[] findShortestAndLongest(String[][] table) {

int shortest = Integer.MAX\_VALUE;

int longest = Integer.MIN\_VALUE;

for (int i = 0; i < table.length; i++) {

int length = Integer.parseInt(table[i][1]);

if (length < shortest) {

shortest = length;

}

if (length > longest) {

longest = length;

}

}

return new int[]{shortest, longest};

}

// MAIN

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

// Input

System.out.print("Enter a sentence: ");

String text = sc.nextLine();

// Step 1: Split manually

String[] words = manualSplit(text);

// Step 2: Create 2D array with word + length

String[][] table = wordsWithLengths(words);

// Step 3: Find shortest and longest

int[] result = findShortestAndLongest(table);

// Step 4: Display all words + lengths

System.out.println("\n--- Words and Their Lengths ---");

System.out.printf("%-15s %-10s\n", "Word", "Length");

System.out.println("------------------------------");

for (int i = 0; i < table.length; i++) {

System.out.printf("%-15s %-10d\n", table[i][0], Integer.parseInt(table[i][1]));

}

// Step 5: Display shortest and longest

System.out.println("\nShortest word length: " + result[0]);

System.out.println("Longest word length: " + result[1]);

sc.close();

}

}

**5. 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:**

**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.Scanner;

public class VowelConsonantCounter {

// Method to check if a character is vowel/consonant

public static String checkChar(char ch) {

// Convert uppercase to lowercase using ASCII (A=65, Z=90 → a=97, z=122)

if (ch >= 'A' && ch <= 'Z') {

ch = (char)(ch + 32); // convert to lowercase

}

// Check if letter

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 count vowels and consonants

public static int[] countVowelsConsonants(String str) {

int vowels = 0, consonants = 0;

try {

int index = 0;

while (true) {

char ch = str.charAt(index); // throws exception when out of bounds

String type = checkChar(ch);

if (type.equals("Vowel")) {

vowels++;

} else if (type.equals("Consonant")) {

consonants++;

}

index++;

}

} catch (StringIndexOutOfBoundsException e) {

// loop ends

}

return new int[]{vowels, consonants};

}

// MAIN

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter a string: ");

String input = sc.nextLine();

// Count vowels & consonants

int[] result = countVowelsConsonants(input);

System.out.println("\n--- Result ---");

System.out.println("Vowels: " + result[0]);

System.out.println("Consonants: " + result[1]);

sc.close();

}

}

**6. 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.**

import java.util.Scanner;

public class VowelConsonantType {

// Method to check if character is vowel, consonant, or not a letter

public static String checkChar(char ch) {

// Convert uppercase → lowercase using ASCII

if (ch >= 'A' && ch <= 'Z') {

ch = (char)(ch + 32); // to lowercase

}

// Check if alphabet

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 analyze string and return 2D array

public static String[][] analyzeString(String str) {

int len = 0;

// find length without length()

try {

while (true) {

str.charAt(len);

len++;

}

} catch (StringIndexOutOfBoundsException e) {

// stop

}

// Create 2D array

String[][] result = new String[len][2];

for (int i = 0; i < len; i++) {

char ch = str.charAt(i);

result[i][0] = String.valueOf(ch);

result[i][1] = checkChar(ch);

}

return result;

}

// Method to display 2D array in tabular format

public static void displayTable(String[][] table) {

System.out.printf("%-10s %-15s\n", "Character", "Type");

System.out.println("----------------------------");

for (int i = 0; i < table.length; i++) {

System.out.printf("%-10s %-15s\n", table[i][0], table[i][1]);

}

}

// MAIN

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter a string: ");

String input = sc.nextLine();

// Analyze string

String[][] result = analyzeString(input);

// Display table

System.out.println("\n--- Character Analysis ---");

displayTable(result);

sc.close();

}

}