**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**

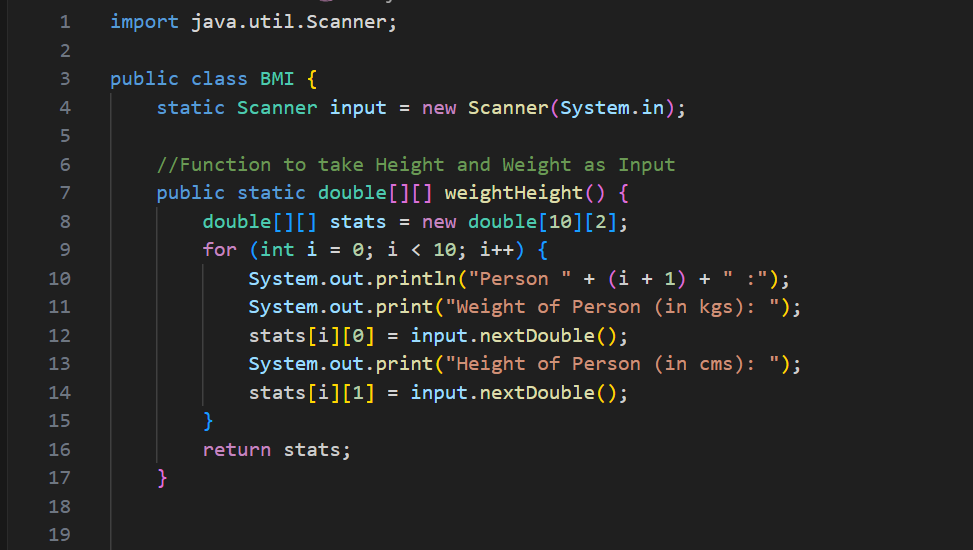
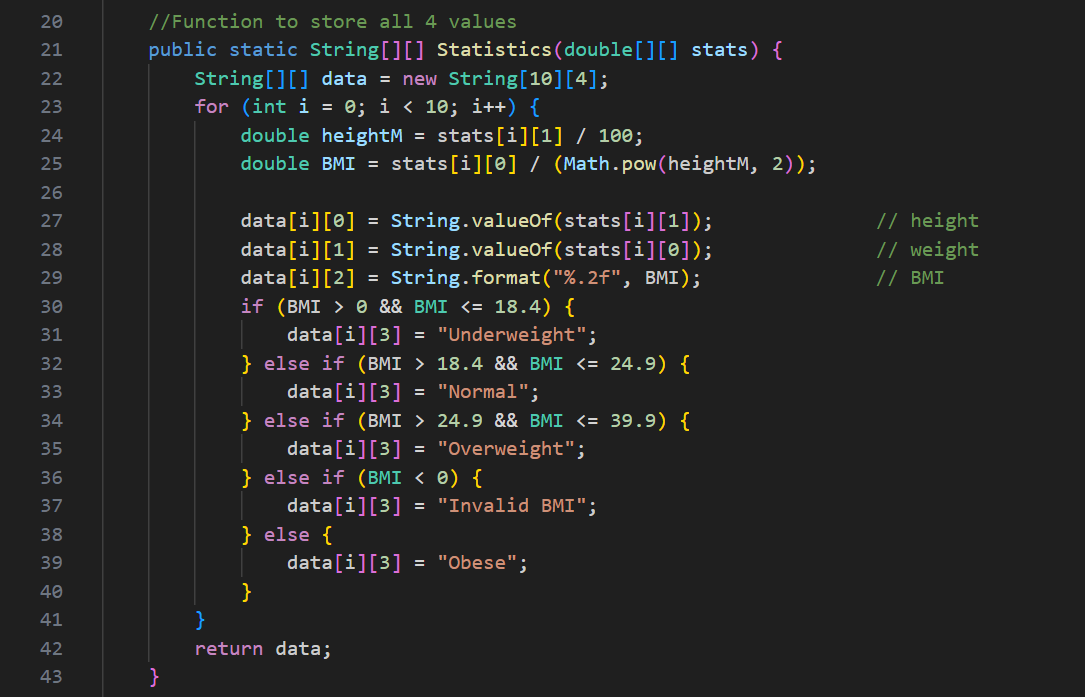
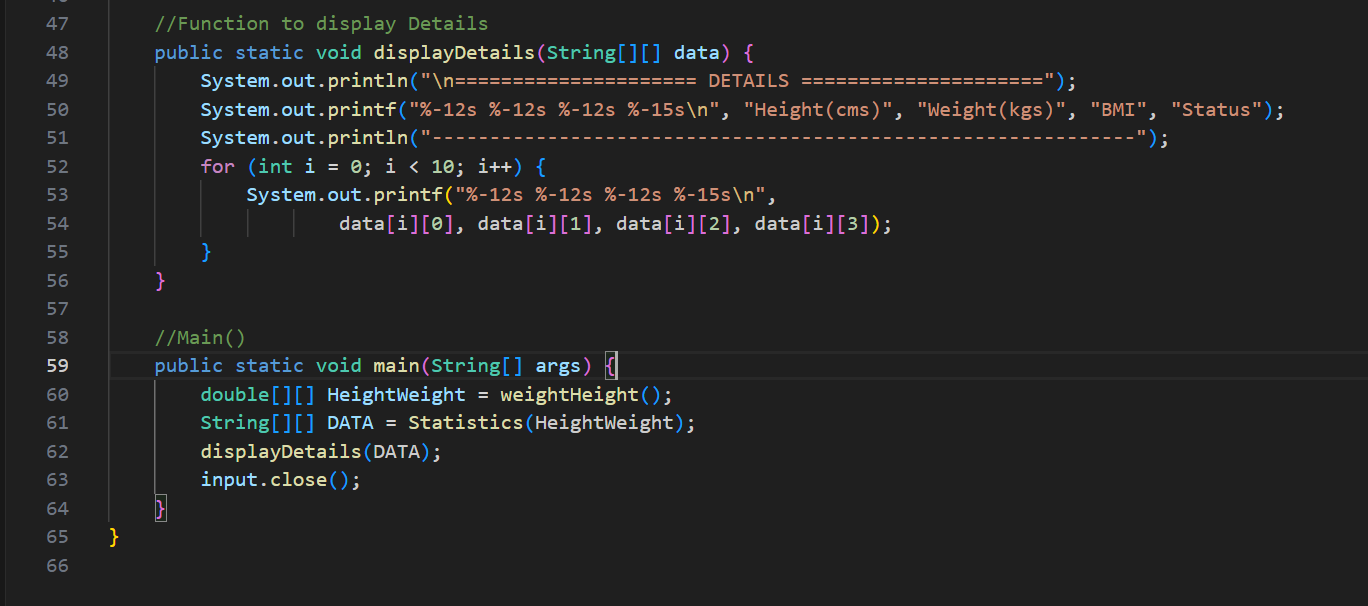
**array of height, weight, BMI, and status.**

**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.**

**** **** 

**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**

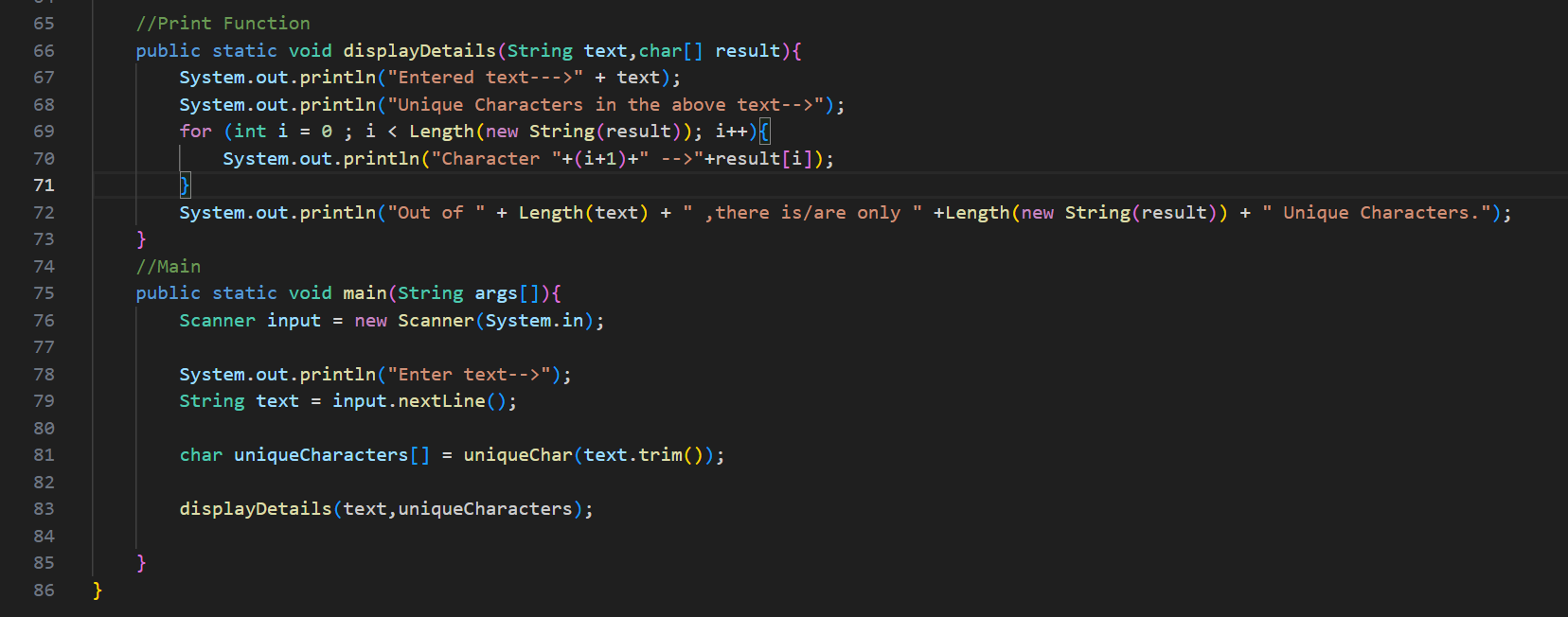
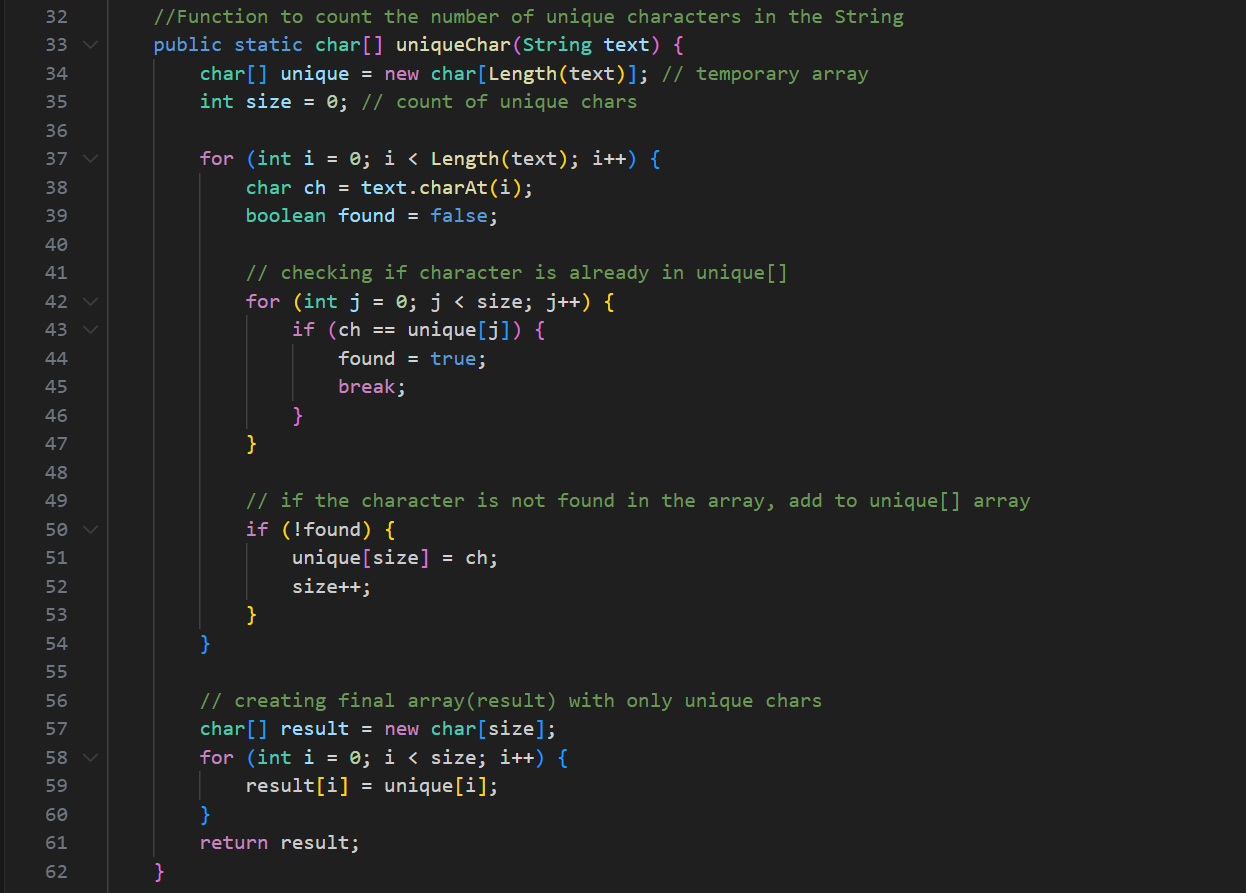
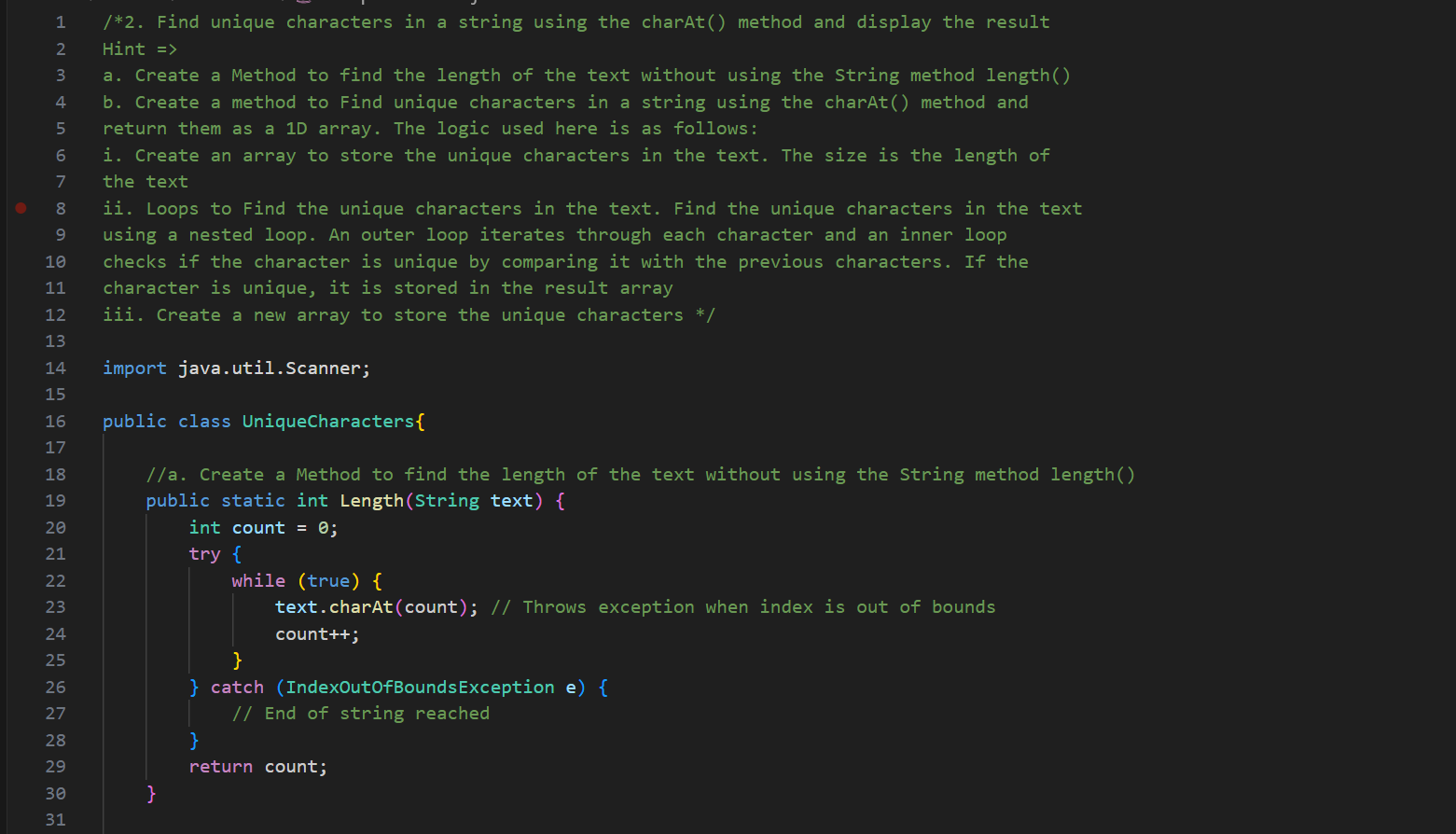
**ii. Loops to Find the unique characters in the text. Find the unique characters in the text**

**using a nested loop. An outer loop iterates through each character and an inner loop**

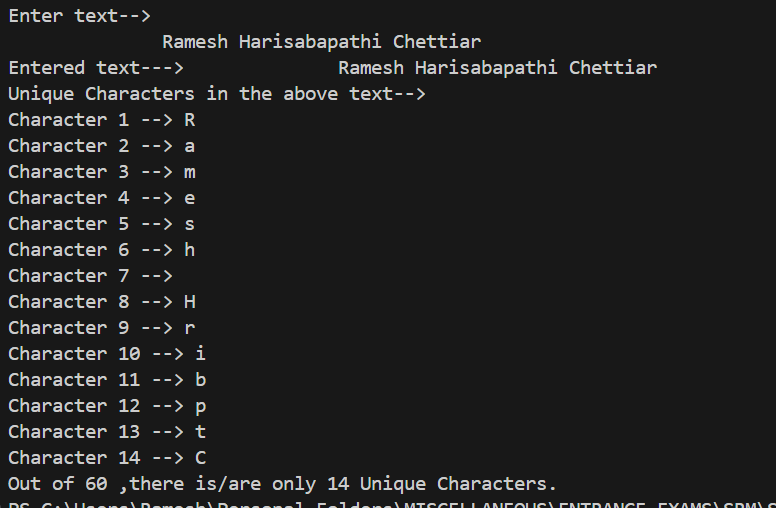
**checks if the character is unique by comparing it with the previous characters. If the**

**character is unique, it is stored in the result array**

**iii. Create a new array to store the unique characters**



**OUTPUT-**

****

**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.**

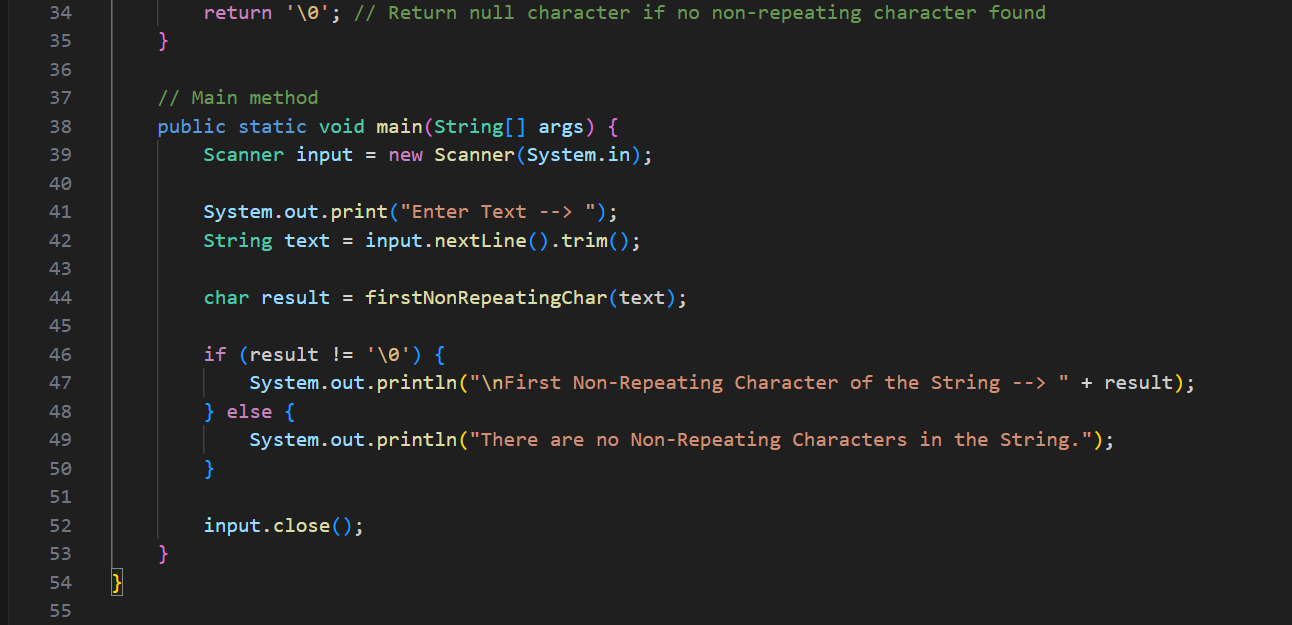
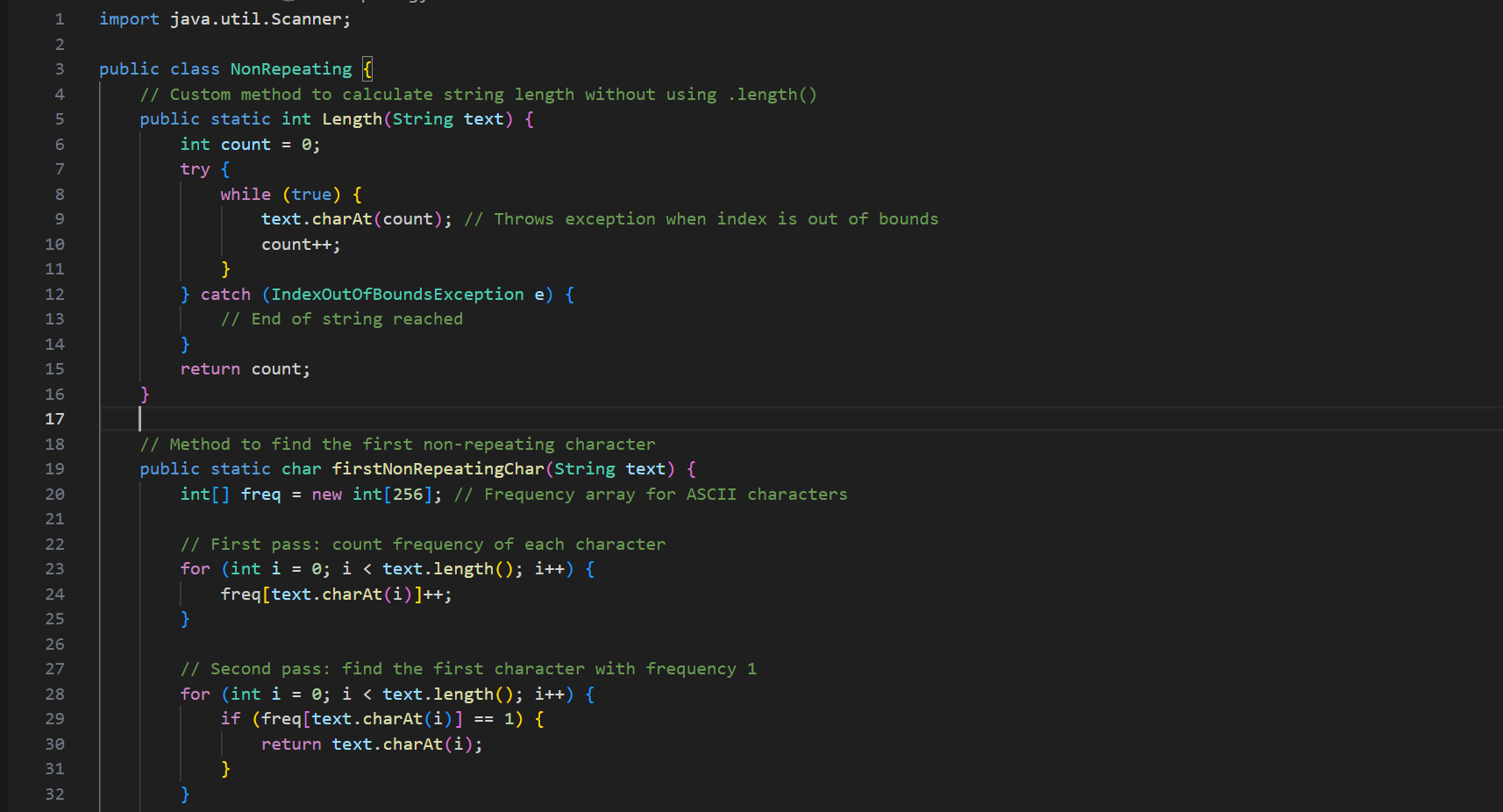
**There are 256 ASCII characters**

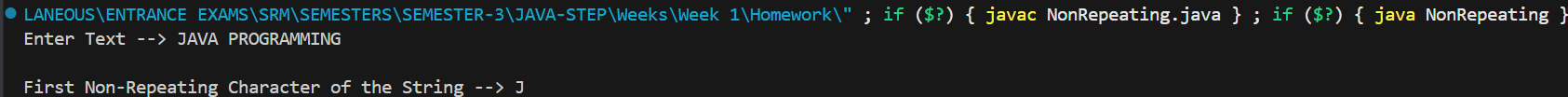
**ii. Loop through the text to find the frequency of characters in the text**

**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.**

****

**OUTPUT🡪**

**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**

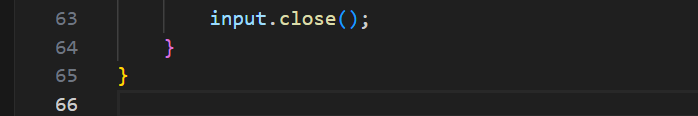
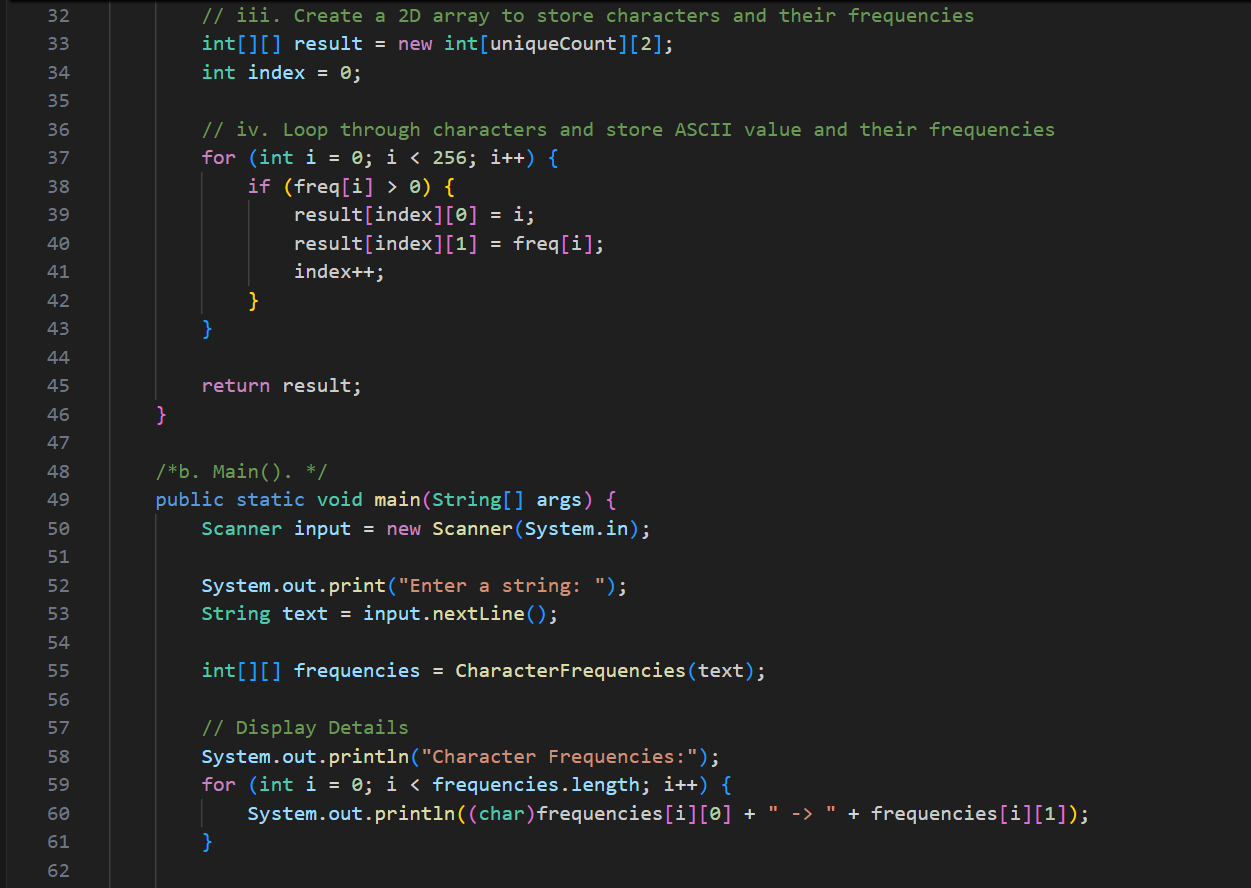
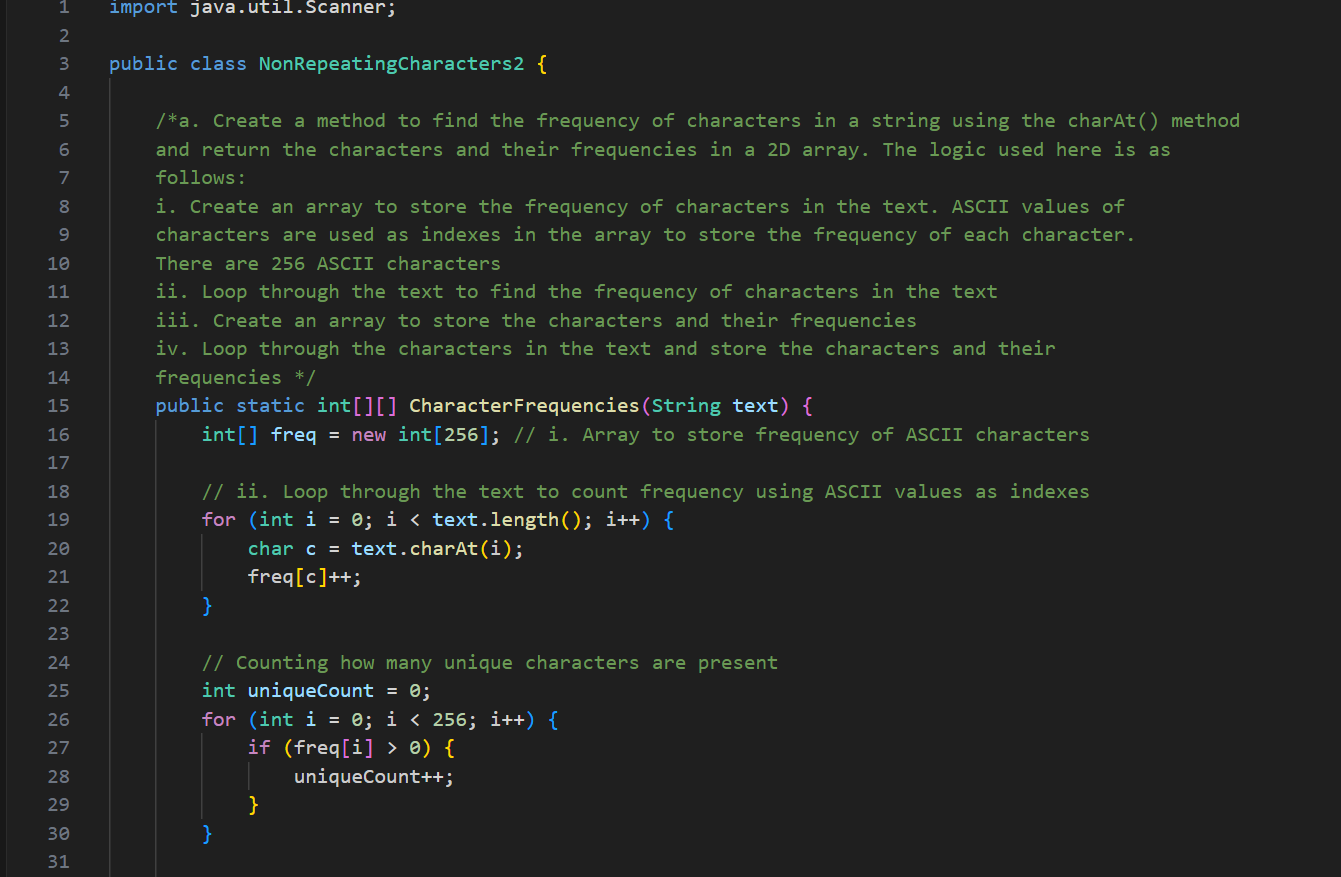
**ii. Loop through the text to find the frequency of characters in the text**

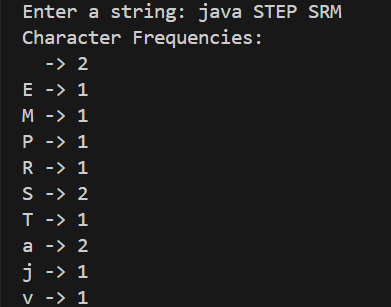
**iii. Create an array to store the characters and their frequencies**

**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.**

****

**OUTPUT🡪**

**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**

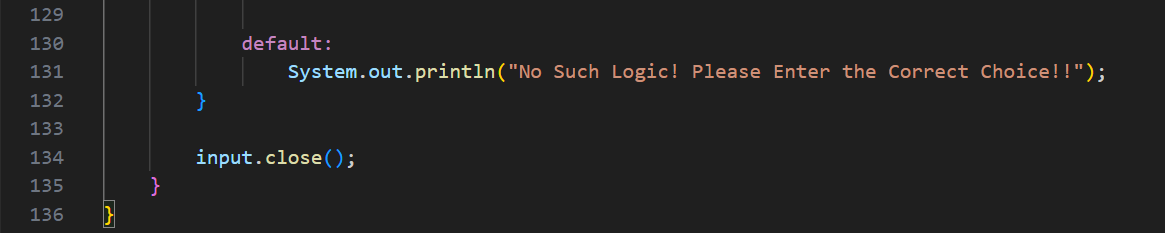
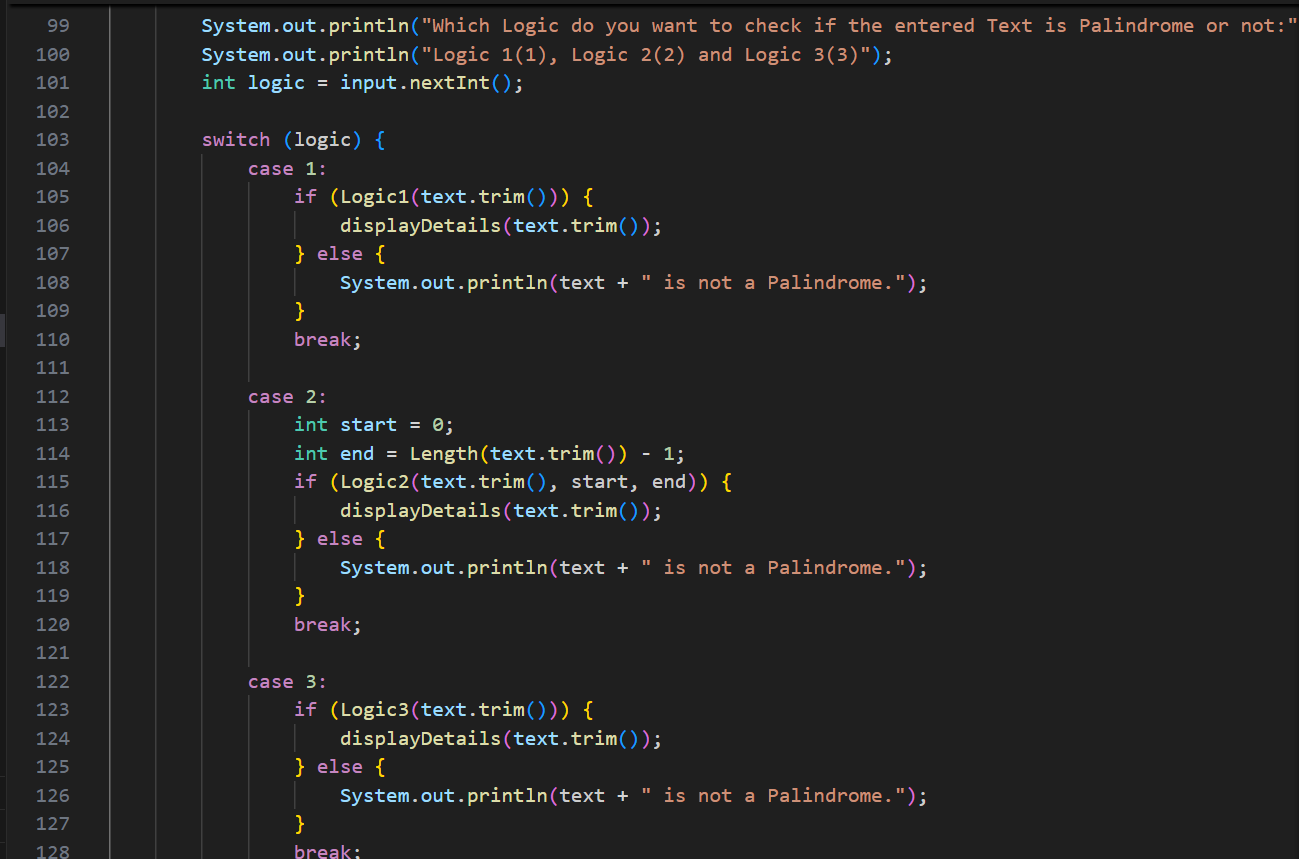
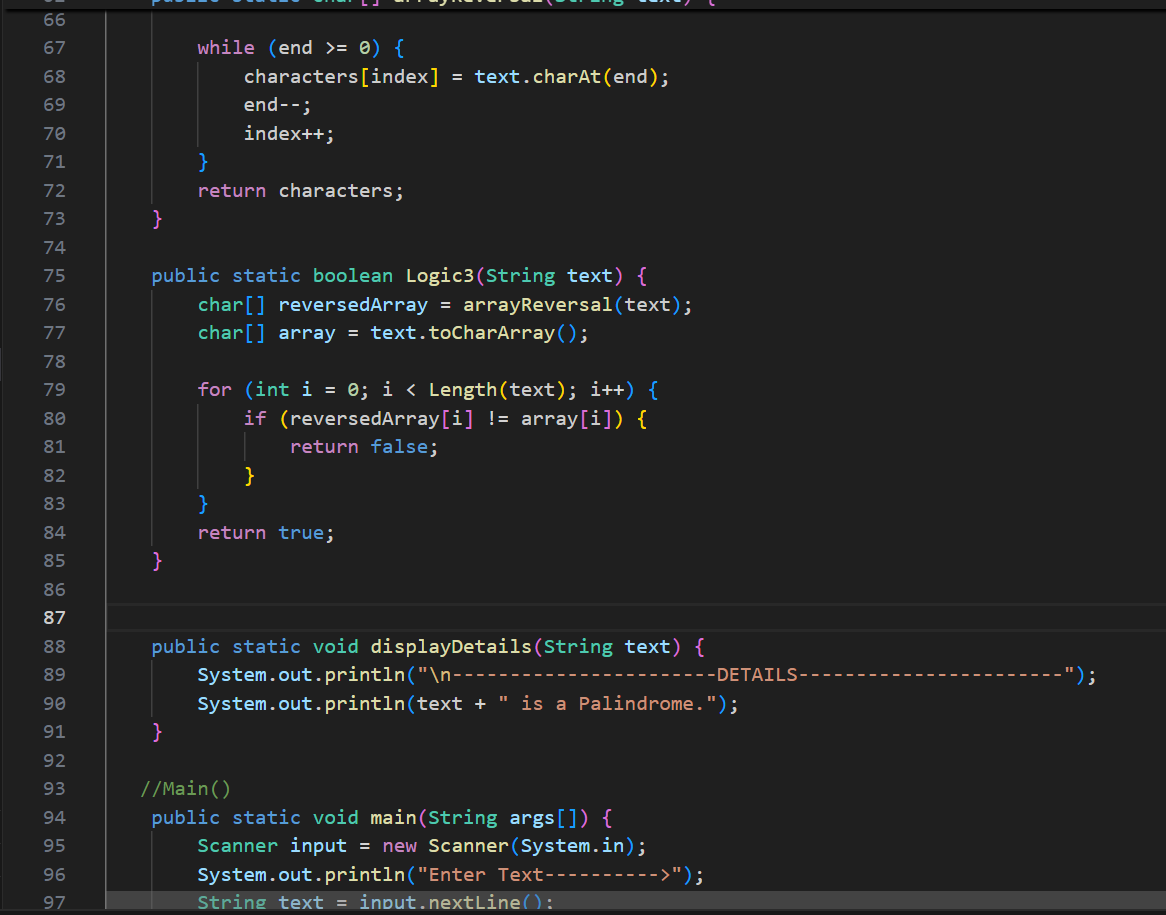
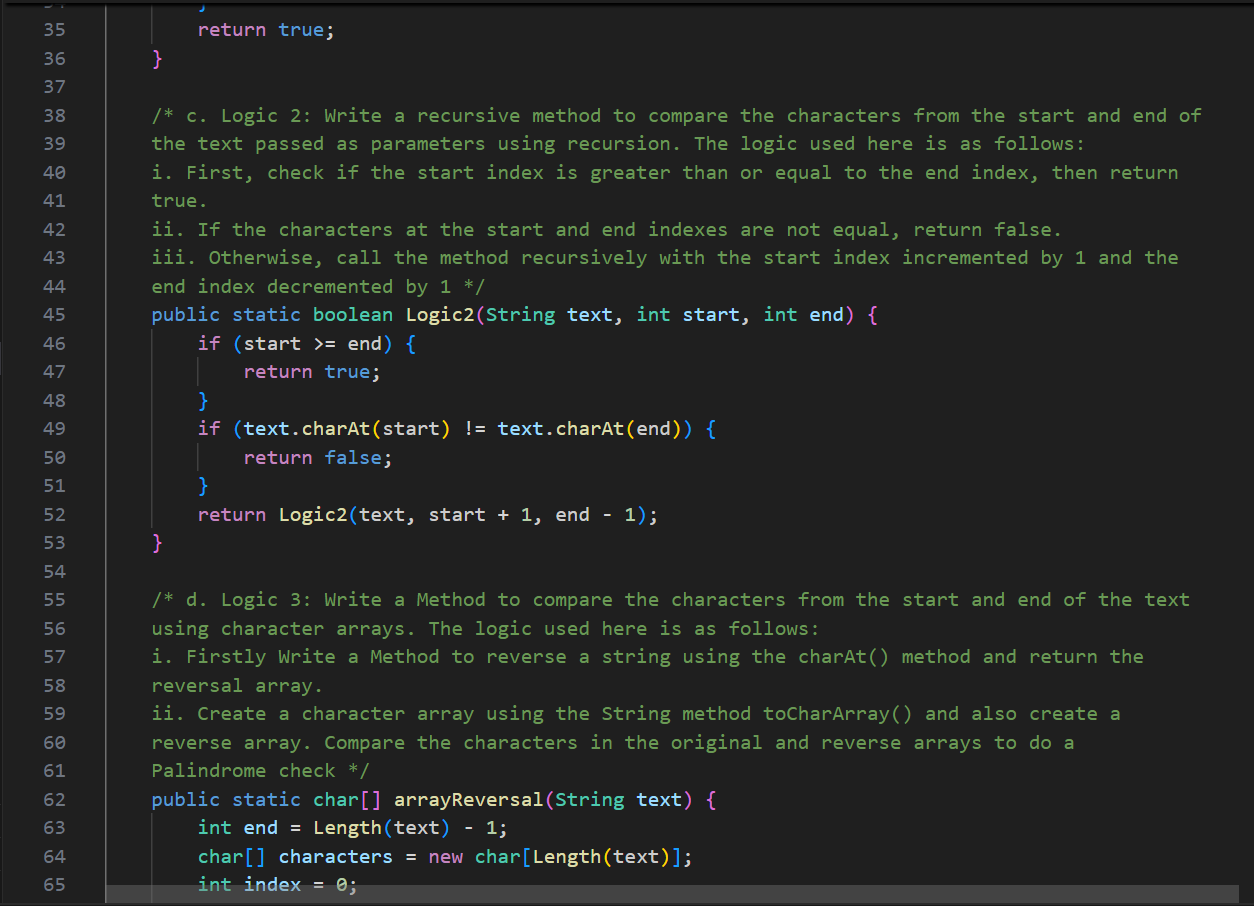
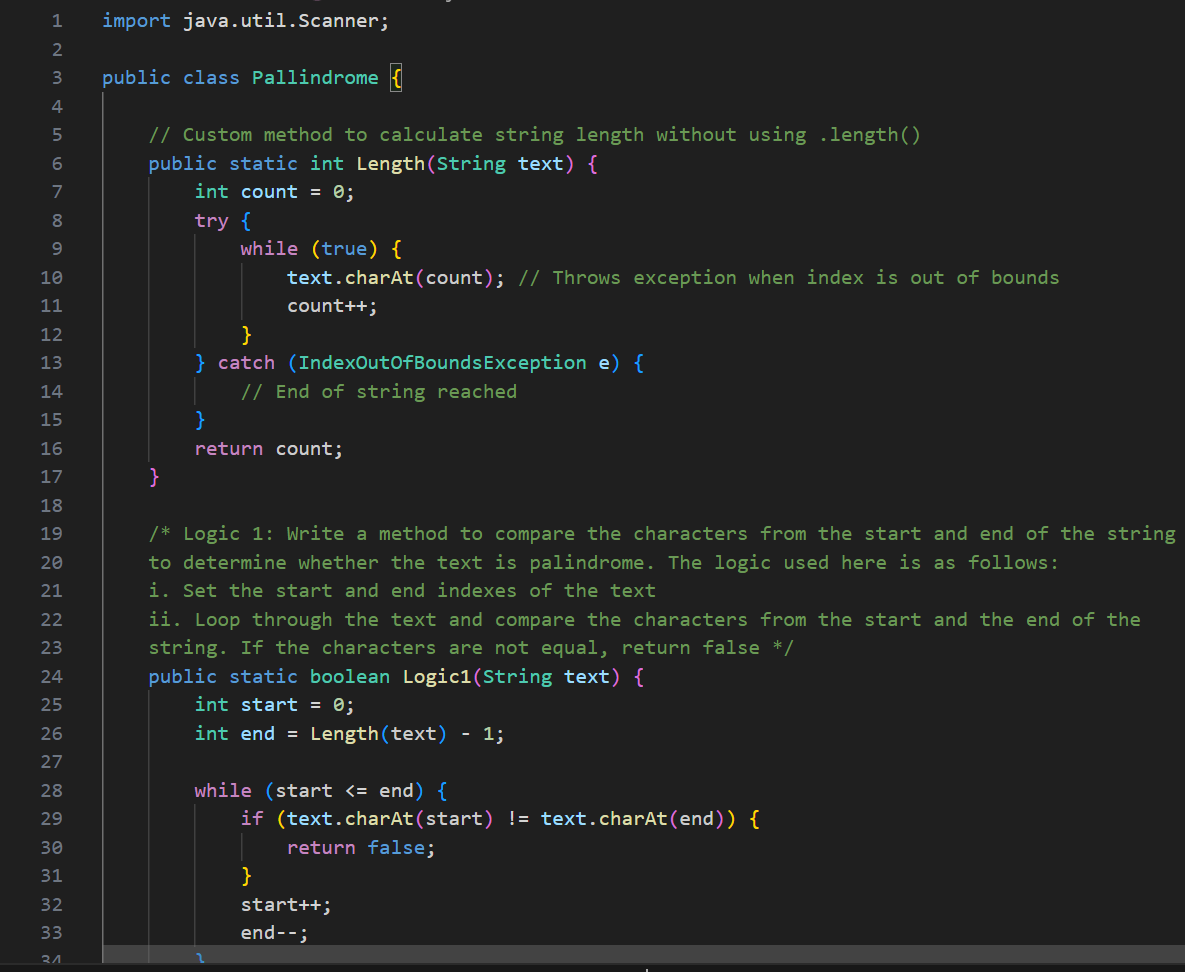
**reversal array.**

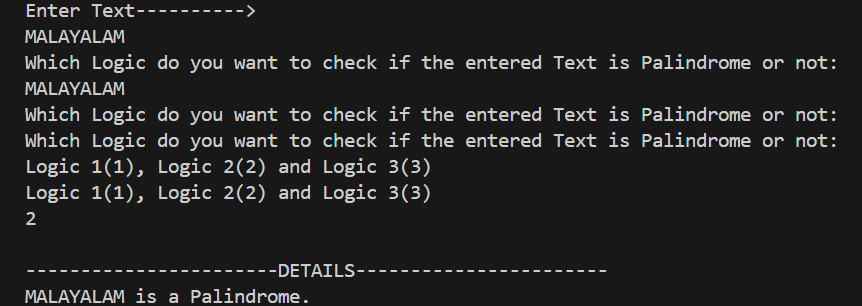
**ii. Create a character array using the String method toCharArray() and also create a**

**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**

****

**OUTPUT🡪**

**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**

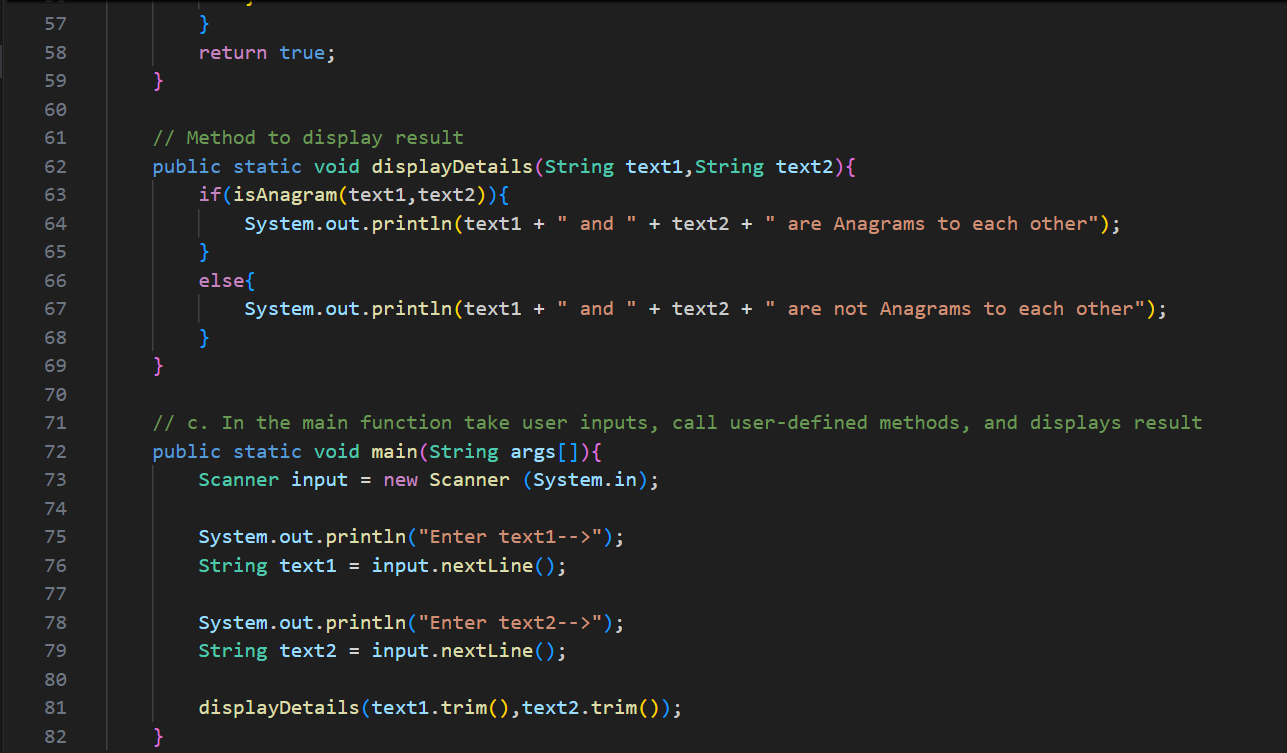
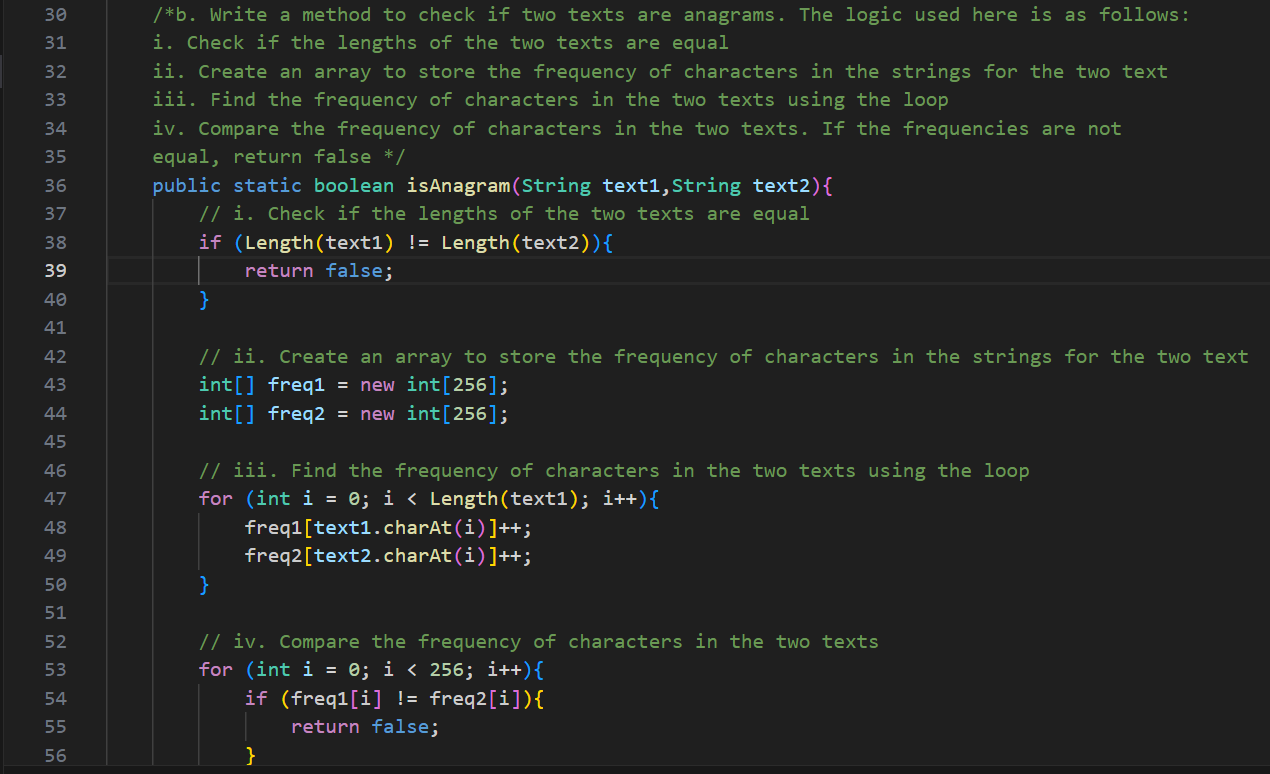
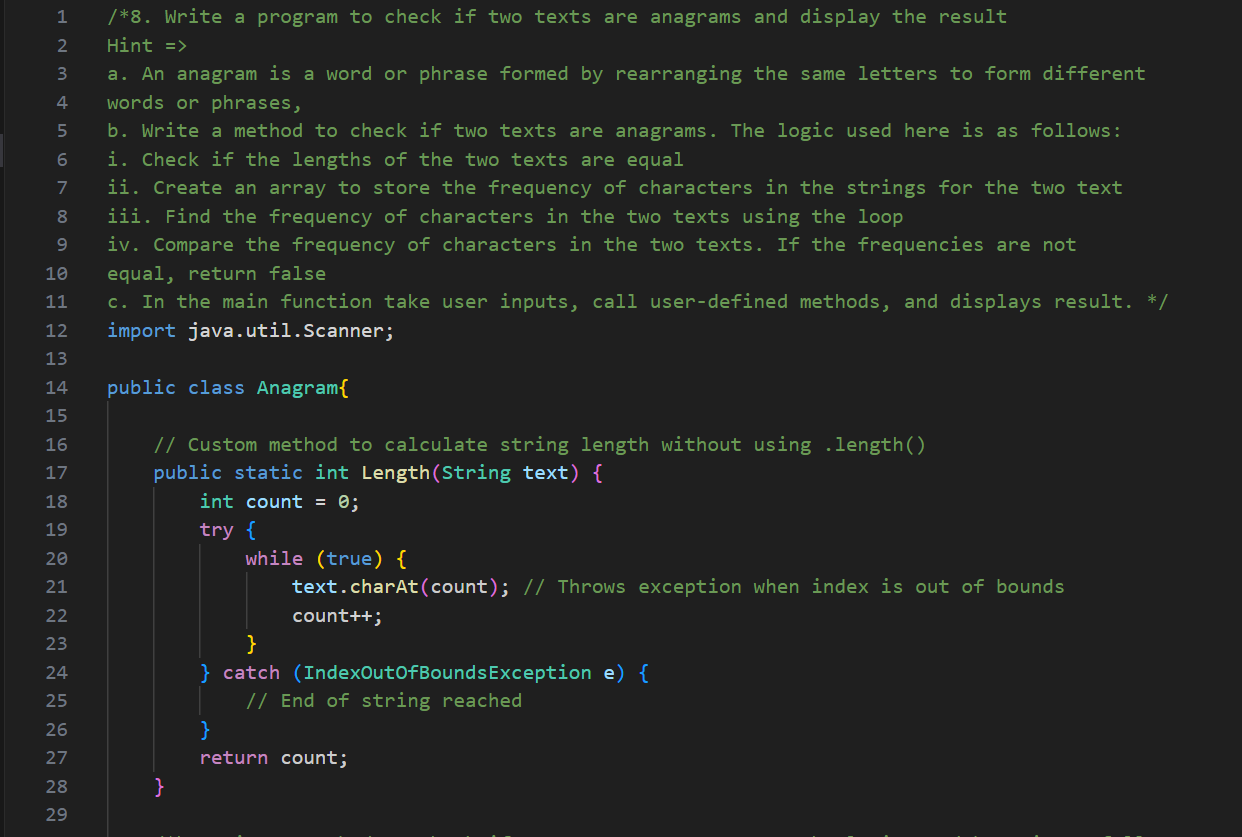
**ii. Create an array to store the frequency of characters in the strings for the two text**

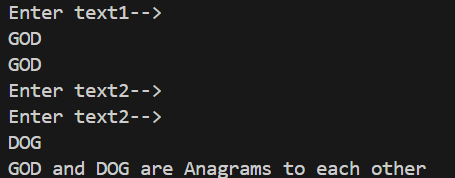
**iii. Find the frequency of characters in the two texts using the loop**

**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.**

****

**OUTPUT🡪**