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.
Output

```
import java.util.Scanner;

public class BMICalculator {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int numMembers = 10;
        double[][] personData = new double[numMembers][2];

        // Take user input for weight and height
        for (int i = 0; i < numMembers; i++) {
            System.out.println("Enter details for person " + (i + 1) + ":");
            System.out.print("Enter weight (kg): ");
            personData[i][0] = scanner.nextDouble();
            System.out.print("Enter height (cm): ");
            personData[i][1] = scanner.nextDouble();
        }

        String[][] bmiData = calculateBMI(personData);
        displayBMI(bmiData);

        scanner.close();
    }

    // Method to calculate BMI and status
    public static String[][] calculateBMI(double[][] personData) {
        String[][] bmiResult = new String[personData.length][4];
        for (int i = 0; i < personData.length; i++) {
            double weight = personData[i][0];
            double heightCm = personData[i][1];
            double heightM = heightCm / 100.0;
            double bmi = weight / (heightM * heightM);

            String status;
            if (bmi < 18.5) {
                status = "Underweight";
            } else if (bmi < 24.9) {
                status = "Normal";
            } else if (bmi < 29.9) {
                status = "Overweight";
            } else {
                status = "Obese";
            }

            bmiResult[i][0] = String.valueOf(weight);
            bmiResult[i][1] = String.valueOf(heightCm);
            bmiResult[i][2] = String.format("%.2f", bmi);
            bmiResult[i][3] = status;
        }

        return bmiResult;
    }

    // Method to display BMI data in a table
    public static void displayBMI(String[][] data) {
        System.out.println("\n--- BMI Report ---");
        System.out.printf("%-10s %-10s %-10s %-15s\n", "Weight (kg)", "Height (cm)", "BMI",
"Status");
        System.out.println("---------------------------------------------");
        for (String[] row : data) {
            System.out.printf("%-10s %-10s %-10s %-15s\n", row[0], row[1], row[2], row[3]);
        }
    }
}
```

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

```
import java.util.Arrays;
import java.util.Scanner;

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

        char[] uniqueChars = findUniqueCharacters(text);
        System.out.println("Unique characters: " + Arrays.toString(uniqueChars));

        scanner.close();
    }

    // Method to find the length of a string without using .length()
    public static int getStringLength(String text) {
        int length = 0;
        try {
            while (true) {
                text.charAt(length);
                length++;
            }
        } catch (StringIndexOutOfBoundsException e) {
            return length;
        }
    }

    // Method to find and return unique characters
    public static char[] findUniqueCharacters(String text) {
        int length = getStringLength(text);
        char[] uniqueChars = new char[length];
        int uniqueCount = 0;
        for (int i = 0; i < length; i++) {
            char currentChar = text.charAt(i);
            boolean isUnique = true;

            // Check if the character has been seen before
            for (int j = 0; j < uniqueCount; j++) {
                if (uniqueChars[j] == currentChar) {
                    isUnique = false;
                    break;
                }
            }

            if (isUnique) {
                uniqueChars[uniqueCount] = currentChar;
                uniqueCount++;
            }
        }

        // Create a new array with the correct size
        char[] result = new char[uniqueCount];
        for (int i = 0; i < uniqueCount; i++) {
            result[i] = uniqueChars[i];
        }

        return result;
    }
}
```

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

```
import java.util.Arrays;
import java.util.Scanner;

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

        char nonRepeatingChar = findFirstNonRepeatingChar(text);
        if (nonRepeatingChar != 0) {
            System.out.println("The first non-repeating character is: "
+ nonRepeatingChar);
        } else {
            System.out.println("No non-repeating character found.");
        }

        scanner.close();
    }

    // Method to find the first non-repeating character
    public static char findFirstNonRepeatingChar(String text) {
        int[] freq = new int[256];
        Arrays.fill(freq, 0);

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

        // Find the first character with a frequency of 1
        for (int i = 0; i < text.length(); i++) {
            if (freq[text.charAt(i)] == 1) {
                return text.charAt(i);
            }
        }

        return 0; // Return a null character if none is found
    }
}
```

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

```
import java.util.Arrays;
import java.util.Scanner;

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

        String[][] frequencies = findFrequencies(text);
        System.out.println("\n--- Character Frequencies ---");
        for (String[] entry : frequencies) {
            System.out.println("Character: " + entry[0] + ", Frequency: " + entry[1]);
        }
        scanner.close();
    }

    // Method to find the frequency of characters
    public static String[][] findFrequencies(String text) {
        int[] freq = new int[256];
        Arrays.fill(freq, 0);

        // Count character frequencies
        for (int i = 0; i < text.length(); i++) {
            freq[text.charAt(i)]++;
        }

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

        String[][] result = new String[uniqueCount][2];
        int resultIndex = 0;

        // Store characters and their frequencies
        for (int i = 0; i < 256; i++) {
            if (freq[i] > 0) {
                result[resultIndex][0] = String.valueOf((char) i);
                result[resultIndex][1] = String.valueOf(freq[i]);
                resultIndex++;
            }
        }

        return result;
    }
}
```

5. Write a program to find the frequency of characters in a string using unique characters and display the result
Hint =>
a. Create a method to Find unique characters in a string using the charAt() method and return them as a 1D array. Use Nested Loops to find the unique characters in the text
b. Create a method to find the frequency of characters in a string 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. Call the uniqueCharacters() method to find the unique characters in the text
iv. Create a 2D String array to store the unique characters and their frequencies.
v. Loop through the unique characters and store the characters and their frequencies
c. In the main function take user inputs, call user-defined methods, and displays result.
Output

```
import java.util.Arrays;
import java.util.Scanner;

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

        String[][] frequencies = findFrequencies(text);
        System.out.println("\n--- Character Frequencies ---");
        for (String[] entry : frequencies) {
            System.out.println("Character: " + entry[0] + ", Frequency: " + entry[1]);
        }

        scanner.close();
    }

    // Method to find the frequency of characters
    public static String[][] findFrequencies(String text) {
        int[] freq = new int[256];
        Arrays.fill(freq, 0);

        // Count character frequencies
        for (int i = 0; i < text.length(); i++) {
            freq[text.charAt(i)]++;
        }

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

        String[][] result = new String[uniqueCount][2];
        int resultIndex = 0;

        // Store characters and their frequencies
        for (int i = 0; i < 256; i++) {
            if (freq[i] > 0) {
                result[resultIndex][0] = String.valueOf((char) i);
                result[resultIndex][1] = String.valueOf(freq[i]);
                resultIndex++;
            }
        }

        return result;
    }
}
```

6. Write a program to find the frequency of characters in a string using nested loops and display the result
Hint =>
a. Create a method to find the frequency of characters in a string and return the characters and their frequencies in a 1D array. The logic used here is as follows:
i. Create an array to store the frequency of each character in the text and an array to store the characters in the text using the toCharArray() method
ii. Loops to Find the frequency of each character in the text and store the result in a frequency array. For this use a Nested Loop with an Outer loop to iterate through each character in the text and initialize the frequency of each character to 1. And an Inner loop to check for duplicate characters. In case of duplicate increment the frequency value and set the duplicate characters to '0' to avoid counting them again.
iii. Create a 1D String array to store the characters and their frequencies. For this Iterate through the characters in the text and store the characters and their frequencies
b. Finally, the main function takes user inputs, calls the user-defined methods, and displays the result.
Output

```
import java.util.Arrays;
import java.util.Scanner;

public class CharacterFrequencyUnique {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a string: ");
        String text = scanner.nextLine();
        String[][] frequencies = findFrequencies(text);
        System.out.println("\n--- Character Frequencies ---");
        for (String[] entry : frequencies) {
            System.out.println("Character: " + entry[0] + ", Frequency: " + entry[1]);
        }

        scanner.close();
    }

    // Method to find unique characters
    public static char[] findUniqueCharacters(String text) {
        int length = 0;
        try {
            while (true) {
                text.charAt(length);
                length++;
            }
        } catch (StringIndexOutOfBoundsException e) {}

        char[] uniqueChars = new char[length];
        int uniqueCount = 0;

        for (int i = 0; i < length; i++) {
            char currentChar = text.charAt(i);
            boolean isUnique = true;
            for (int j = 0; j < uniqueCount; j++) {
                if (uniqueChars[j] == currentChar) {
                    isUnique = false;
                    break;
                }
            }
            if (isUnique) {
                uniqueChars[uniqueCount] = currentChar;
                uniqueCount++;
            }
        }

        char[] result = new char[uniqueCount];
        for (int i = 0; i < uniqueCount; i++) {
            result[i] = uniqueChars[i];
        }
        return result;
    }
    // Method to find the frequency of characters using unique characters
    public static String[][] findFrequencies(String text) {
        int[] freq = new int[256];
        Arrays.fill(freq, 0);

        for (int i = 0; i < text.length(); i++) {
            freq[text.charAt(i)]++;
        }

        char[] uniqueChars = findUniqueCharacters(text);
        String[][] result = new String[uniqueChars.length][2];

        for (int i = 0; i < uniqueChars.length; i++) {
            char uniqueChar = uniqueChars[i];
            result[i][0] = String.valueOf(uniqueChar);
            result[i][1] = String.valueOf(freq[uniqueChar]);
        }

        return result;
    }
}
```

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

```
import java.util.Arrays;
import java.util.Scanner;

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

        String[][] frequencies = findFrequencies(text);
        System.out.println("\n--- Character Frequencies ---");
        for (String[] entry : frequencies) {
            if (entry[0] != null) {
                System.out.println("Character: " + entry[0] + ", Frequency: " + entry[1]);
            }
        }
        scanner.close();
    }

    // Method to find character frequencies using nested loops
    public static String[][] findFrequencies(String text) {
        char[] charArray = text.toCharArray();
        int n = charArray.length;
        String[][] result = new String[n][2];

        int resultIndex = 0;
        for (int i = 0; i < n; i++) {
            if (charArray[i] == '0') {
                continue; // Skip characters already counted
            }

            int count = 1;
            for (int j = i + 1; j < n; j++) {
                if (charArray[i] == charArray[j]) {
                    count++;
                    charArray[j] = '0'; // Mark as counted
                }
            }
            result[resultIndex][0] = String.valueOf(charArray[i]);
            result[resultIndex][1] = String.valueOf(count);
            resultIndex++;
        }

        String[][] finalResult = new String[resultIndex][2];
        for(int i = 0; i < resultIndex; i++){
            finalResult[i] = result[i];
        }

        return finalResult;
    }
}
```

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
Output

```
import java.util.Arrays;
import java.util.Scanner;

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

        String[][] frequencies = findFrequencies(text);
        System.out.println("\n--- Character Frequencies ---");
        for (String[] entry : frequencies) {
            if (entry[0] != null) {
                System.out.println("Character: " + entry[0] + ", Frequency: " + entry[1]);
            }
        }
        scanner.close();
    }

    // Method to find character frequencies using nested loops
    public static String[][] findFrequencies(String text) {
        char[] charArray = text.toCharArray();
        int n = charArray.length;
        String[][] result = new String[n][2];

        int resultIndex = 0;
        for (int i = 0; i < n; i++) {
            if (charArray[i] == '0') {
                continue; // Skip characters already counted
            }

            int count = 1;
            for (int j = i + 1; j < n; j++) {
                if (charArray[i] == charArray[j]) {
                    count++;
                    charArray[j] = '0'; // Mark as counted
                }
            }
            result[resultIndex][0] = String.valueOf(charArray[i]);
            result[resultIndex][1] = String.valueOf(count);
            resultIndex++;
        }

        String[][] finalResult = new String[resultIndex][2];
        for(int i = 0; i < resultIndex; i++){
            finalResult[i] = result[i];
        }

        return finalResult;
    }
}
```

9. Create a program to display a calendar for a given month and year. The program should take the month and year as input from the user and display the calendar for that month. E.g. for 07 2005 user input, the program should display the calendar as shown below

Hint =>
a. Write a Method to get the name of the month. For this define a month Array to store the names of the months
b. Write a Method to get the number of days in the month. For this define a days Array to store the number of days in each month. For Feb month, check for Leap Year to get the number of days. Also, define a Leap Year Method.
c. Write a method to get the first day of the month using the Gregorian calendar algorithm

$y0 = y - (14 - m) / 12$
$x = y0 + y0/4 - y0/100 + y0/400$
$m0 = m + 12 \times ((14 - m) / 12) - 2$
$d0 = (d + x + 31m0 / 12) \bmod 7$
d. Displaying the Calendar requires 2 for loops.
i. The first for loop up to the first day to get the proper indentation. As in the example above 3 spaces from Sun to Thu as to be set as July 1st starts on Fri
ii. The Second for loop Displays the days of the month starting from 1 to the number of days. Add proper indentation for single-digit days using %3d to display the integer right-justified in a field of width 3. Please note to move to the next line after Sat
Output

```java
import java.util.Arrays;
import java.util.Scanner;

public class AnagramChecker {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the first string: ");
        String text1 = scanner.nextLine();
        System.out.print("Enter the second string: ");
        String text2 = scanner.nextLine();

        if (areAnagrams(text1, text2)) {
            System.out.println("'" + text1 + "' and '" + text2 + "' are anagrams.");
        } else {
            System.out.println("'" + text1 + "' and '" + text2 + "' are not anagrams.");
        }

        scanner.close();
    }

    // Method to check if two strings are anagrams
    public static boolean areAnagrams(String text1, String text2) {
        if (text1.length() != text2.length()) {
            return false;
        }

        int[] freq1 = new int[256];
        int[] freq2 = new int[256];
        Arrays.fill(freq1, 0);
        Arrays.fill(freq2, 0);

        // Find frequency of characters in both strings
        for (int i = 0; i < text1.length(); i++) {
            freq1[text1.charAt(i)]++;
            freq2[text2.charAt(i)]++;
        }

        // Compare the frequency arrays
        for (int i = 0; i < 256; i++) {
            if (freq1[i] != freq2[i]) {
                return false;
            }
        }

        return true;
    }
}
```

10. Write a program to create a deck of cards, initialize the deck, shuffle the deck, and distribute the deck of n cards to x number of players. Finally, print the cards the players have.
Hint =>
a. Create a deck of cards with suits "Hearts", "Diamonds", "Clubs", "Spades" and ranks from "2", "3", "4", "5", "6", "7", "8", "9", "10", "Jack", "Queen", "King", and "Ace"
b. Calculate the number of cards in the deck and initialize the deck
int numOfCards = suits.length * ranks.length;
c. Write a Method to Initialize the deck of cards with suits and ranks and return the deck. The deck is an array of strings where each string represents a card in the deck represented as "rank of suit" e.g., "2 of Hearts"
d. Write a Method to Shuffle the deck of cards and return the shuffled deck. To shuffle the card iterate over the deck and swap each card with a random card from the remaining deck to shuffle the deck. Please find the steps below
Step1: Use for Loop Iterate over the deck and swap each card with a random card from the remaining deck
Step 2: Inside the Loop Generate a random card number between i and n using the following code
int randomCardNumber = i + (int) (Math.random() * (n - i));
Step 3: Swap the current card with the random card
e. Write a Method to distribute the deck of n cards to x number of players and return the players. For this Check the n cards can be distributed to x players. If possible then Create a 2D array to store the players and their cards
f. Write a Method to Print the players and their cards
Output

```
import java.util.Scanner;

public class CalendarGenerator {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter month (1-12): ");
        int month = scanner.nextInt();
        System.out.print("Enter year: ");
        int year = scanner.nextInt();

        displayCalendar(month, year);

        scanner.close();
    }

    // Method to get month name
    public static String getMonthName(int month) {
        String[] months = {"", "January", "February", "March", "April", "May", "June", "July",
"August", "September", "October", "November", "December"};
        return months[month];
    }

    // Method to check for leap year
    public static boolean isLeapYear(int year) {
        return (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0);
    }

    // Method to get number of days in a month
    public static int getNumDaysInMonth(int month, int year) {
        int[] days = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
        if (month == 2 && isLeapYear(year)) {
            return 29;
        }
        return days[month];
    }

    // Method to get the first day of the month using Gregorian calendar algorithm
    public static int getFirstDayOfMonth(int month, int year) {
        int y0 = year - (14 - month) / 12;
        int x = y0 + y0 / 4 - y0 / 100 + y0 / 400;
        int m0 = month + 12 * ((14 - month) / 12) - 2;
        return (1 + x + (31 * m0) / 12) % 7;
    }

    // Method to display the calendar
    public static void displayCalendar(int month, int year) {
        System.out.println("\n--- " + getMonthName(month) + " " + year + " ---");
        System.out.println("  Sun  Mon  Tue  Wed  Thu  Fri  Sat");
        int firstDay = getFirstDayOfMonth(month, year);
        int numDays = getNumDaysInMonth(month, year);

        // Print indentation for the first week
        for (int i = 0; i < firstDay; i++) {
            System.out.print("     ");
        }

        // Print the days
        for (int day = 1; day <= numDays; day++) {
            System.out.printf("%5d", day);
            if ((day + firstDay) % 7 == 0 || day == numDays) {
                System.out.println();
            }
        }
    }
}
```