Level 3 Practice Programs

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.

BMI	Status
≤ 18.4	Underweight
18.5 - 24.9	Normal
25.0 - 39.9	Overweight
≥ 40.0	Obese

CODE-

```
import java.util.Scanner;

public class BMICalculator {

    // Method to calculate BMI and status
```

```
    public static String[][] calculateBMI(double[][] data) {
        String[][] result = new String[10][4]; // height, weight,
BMI, status

        for (int i = 0; i < 10; i++) {
            double weight = data[i][0];
            double heightCm = data[i][1];
            double heightM = heightCm / 100.0;

            double bmi = weight / (heightM * heightM);
            String status;

            if (bmi < 18.5) {
                status = "Underweight";
            } else if (bmi < 25) {
                status = "Normal weight";
            } else if (bmi < 30) {
                status = "Overweight";
            } else {
                status = "Obese";
            }

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

        return result;
    }

    // Method to display the result
```

```
    public static void displayResults(String[][] results) {
        System.out.printf("%-10s %-10s %-10s %-15s%n",
"Height(cm)", "Weight(kg)", "BMI", "Status");

System.out.println("----------------------------------------
---------");
        for (int i = 0; i < results.length; i++) {
            System.out.printf("%-10s %-10s %-10s %-15s%n",
                    results[i][0], results[i][1], results[i][2],
results[i][3]);
        }
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        double[][] personData = new double[10][2]; // 0 = weight,
1 = height

        System.out.println("Enter weight (kg) and height (cm) for
10 persons:");
        for (int i = 0; i < 10; i++) {
            System.out.printf("Person %d - Weight (kg): ", i +
1);
            personData[i][0] = scanner.nextDouble();
            System.out.printf("Person %d - Height (cm): ", i +
1);
            personData[i][1] = scanner.nextDouble();
        }

        String[][] results = calculateBMI(personData);
        displayResults(results);
    }
}
```

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.

CODE-

```
import java.util.Scanner;


public class UniqueCharacters {


    // Method to find the length of the string without using length()
    public static int findLength(String text) {
        int length = 0;
        try {
            while (true) {
                text.charAt(length); // will throw exception at end
                length++;
            }
        } catch (Exception e) {
            // End of string reached
        }
        return length;
    }
```

```
// Method to find unique characters using charAt()
public static char[] findUniqueCharacters(String text) {
    int len = findLength(text);
    char[] uniqueTemp = new char[len];
    int uniqueCount = 0;

    for (int i = 0; i < len; i++) {
        char currentChar = text.charAt(i);
        boolean isUnique = true;

        for (int j = 0; j < i; j++) {
            if (text.charAt(j) == currentChar) {
                isUnique = false;
                break;
            }
        }

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

    // Create final array with only unique characters
    char[] uniqueChars = new char[uniqueCount];
    for (int i = 0; i < uniqueCount; i++) {
        uniqueChars[i] = uniqueTemp[i];
    }

    return uniqueChars;
}
```

```
// Method to display characters
public static void displayUniqueCharacters(char[] uniqueChars) {
    System.out.print("Unique characters: ");
    for (char c : uniqueChars) {
        System.out.print(c + " ");
    }
    System.out.println();
}

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

        char[] uniqueChars = findUniqueCharacters(input);
        displayUniqueCharacters(uniqueChars);
    }
}
```

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.

```
CODE- import java.util.Scanner;

public class FirstNonRepeatingCharacter {

    // Method to find the first non-repeating character
    public static char findFirstNonRepeatingChar(String text) {
        int[] freq = new int[256]; // ASCII size

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

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

        return '\0'; // null character if no non-repeating character
found
    }

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

        char result = findFirstNonRepeatingChar(input);
```

```
        if (result == '\0') {
            System.out.println("No non-repeating character found.");
        } else {
            System.out.println("First non-repeating character: " +
result);
        }
    }
}
```

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.

```
CODE- import java.util.Scanner;

public class CharacterFrequency {


    // Method to find frequency of characters using charAt()
    public static String[][] findCharacterFrequencies(String
text) {
        int[] freq = new int[256]; // ASCII size
```

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

// Count how many unique characters are there
int uniqueCount = 0;
boolean[] counted = new boolean[256];
for (int i = 0; i < text.length(); i++) {
    char ch = text.charAt(i);
    if (!counted[(int) ch]) {
        uniqueCount++;
        counted[(int) ch] = true;
    }
}

// Prepare the result array
String[][] result = new String[uniqueCount][2];
int index = 0;
boolean[] added = new boolean[256];

for (int i = 0; i < text.length(); i++) {
char ch = text.charAt(i);
int ascii = (int) ch;
if (!added[ascii]) {
    result[index][0] = String.valueOf(ch);
    result[index][1] = String.valueOf(freq[ascii]);
    added[ascii] = true;
    index++;
}
```

```
        }

        return result;
    }

    // Display the frequency result
    public static void displayFrequencies(String[][] result) {
        System.out.printf("%-10s %-10s%n", "Character",
"Frequency");
        System.out.println("----------------------");
        for (String[] row : result) {
            System.out.printf("%-10s %-10s%n", row[0], row[1]);
        }
    }

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

        String[][] frequencies = findCharacterFrequencies(input);
        displayFrequencies(frequencies);
    }
}
```

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.

CODE-

```
import java.util.Scanner;

public class CharFrequencyWithUniques {


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

        char[] temp = new char[length];
        int count = 0;

        for (int i = 0; i < length; i++) {
            char current = text.charAt(i);
            boolean isUnique = true;

            for (int j = 0; j < i; j++) {
```

```
            if (text.charAt(j) == current) {
                isUnique = false;
                break;
            }
        }

        if (isUnique) {
            temp[count++] = current;
        }
    }

    char[] uniqueChars = new char[count];
    for (int i = 0; i < count; i++) {
        uniqueChars[i] = temp[i];
    }

    return uniqueChars;
}

// Method to find frequency using unique characters
public static String[][] findFrequency(String text) {
    int[] freq = new int[256]; // ASCII characters

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

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

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

        return result;
    }

    // Display method
    public static void displayResult(String[][] result) {
        System.out.printf("%-10s %-10s%n", "Character", "Frequency");
        System.out.println("----------------------");
        for (String[] row : result) {
            System.out.printf("%-10s %-10s%n", row[0], row[1]);
        }
    }

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

        String[][] frequencyResult = findFrequency(input);
        displayResult(frequencyResult);
    }
}
```

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.

CODE-

```
import java.util.Scanner;

public class CharFrequencyNestedLoops {

    // Method to find frequencies using nested loops
    public static String[] findFrequencies(String text) {
        char[] chars = text.toCharArray();
        int[] freq = new int[chars.length];

        for (int i = 0; i < chars.length; i++) {
            freq[i] = 1;
            if (chars[i] == '0') continue;

            for (int j = i + 1; j < chars.length; j++) {
                if (chars[i] == chars[j]) {
                    freq[i]++;
                    chars[j] = '0'; // mark duplicate
                }
            }
        }
    }
```

```
// Count non-zero (valid) characters
int validCount = 0;
for (int i = 0; i < chars.length; i++) {
    if (chars[i] != '0') {
        validCount++;
    }
}

// Build result string array
    String[] result = new String[validCount];
    int index = 0;
    for (int i = 0; i < chars.length; i++) {
        if (chars[i] != '0') {
            result[index] = chars[i] + " - " + freq[i];
            index++;
        }
    }

    return result;
}

// Display the frequencies
public static void displayResult(String[] result) {
    System.out.println("Character Frequencies:");
    for (String entry : result) {
        System.out.println(entry);
    }
}

public static void main(String[] args) {
```

```
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a string: ");
        String input = scanner.nextLine();

        String[] frequencies = findFrequencies(input);
        displayResult(frequencies);
    }
}
```

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

CODE- `import java.util.Scanner;`

```
public class PalindromeCheck {

    // Logic 1: Iterative check from start and end
    public static boolean isPalindromeIterative(String text) {
        int start = 0;
        int end = text.length() - 1;

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

    // Logic 2: Recursive check
    public static boolean isPalindromeRecursive(String text, int
start, int end) {
        if (start >= end) return true;
        if (text.charAt(start) != text.charAt(end)) return false;
        return isPalindromeRecursive(text, start + 1, end - 1);
}

// Logic 3: Using character arrays
public static boolean isPalindromeUsingArrays(String text) {
    char[] original = text.toCharArray();
    char[] reversed = reverseString(text);

    for (int i = 0; i < original.length; i++) {
        if (original[i] != reversed[i]) {
```

```
            return false;
        }
    }
    return true;
}

// Method to reverse a string using charAt()
public static char[] reverseString(String text) {
    int len = text.length();
    char[] reversed = new char[len];

    for (int i = 0; i < len; i++) {
        reversed[i] = text.charAt(len - 1 - i);
    }

    return reversed;
}

// Main method
public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter text to check if it is a palindrome:
");
    String input = scanner.nextLine();

    boolean result1 = isPalindromeIterative(input);
    boolean result2 = isPalindromeRecursive(input, 0,
input.length() - 1);
    boolean result3 = isPalindromeUsingArrays(input);

    System.out.println("\nPalindrome Check Results:");
```

```
        System.out.println("Using Logic 1 (Iterative): " + (result1 ?
"Palindrome" : "Not Palindrome"));

        System.out.println("Using Logic 2 (Recursive): " + (result2 ?
"Palindrome" : "Not Palindrome"));

        System.out.println("Using Logic 3 (Char Array): " + (result3 ?
"Palindrome" : "Not 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.

CODE-

```
import java.util.Scanner;


public class AnagramChecker {


    // Method to check if two strings are anagrams
    public static boolean areAnagrams(String text1, String text2) {
        // Step 1: Check lengths
        if (text1.length() != text2.length()) {
            return false;
        }

        // Step 2: Create frequency arrays
        int[] freq1 = new int[256]; // for text1
```

```
    int[] freq2 = new int[256]; // for text2

    // Step 3: Count character frequencies
    for (int i = 0; i < text1.length(); i++) {
        freq1[text1.charAt(i)]++;
        freq2[text2.charAt(i)]++;
    }

    // Step 4: Compare frequencies
    for (int i = 0; i < 256; i++) {
        if (freq1[i] != freq2[i]) {
            return false;
        }
    }

    return true;
}

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

    System.out.print("Enter first text: ");
    String text1 = scanner.nextLine();

    System.out.print("Enter second text: ");
    String text2 = scanner.nextLine();

    boolean result = areAnagrams(text1, text2);

    System.out.println("\nResult:");
```

```
        if (result) {
            System.out.println("\"" + text1 + "\" and \"" + text2 +
"\" are Anagrams.");
        } else {
            System.out.println("\"" + text1 + "\" and \"" + text2 +
"\" are NOT Anagrams.");
        }
    }
}
```

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

```
July 2005
Sun Mon Tue Wed Thu Fri Sat
                        1   2
  3   4   5   6   7   8   9
 10  11  12  13  14  15  16
 17  18  19  20  21  22  23
 24  25  26  27  28  29  30
 31
```

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

CODE-
```
import java.util.Scanner;

public class CalendarDisplay {

    // Method to get the name of the month
    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 if a year is a leap year
    public static boolean isLeapYear(int year) {
        return ((year % 4 == 0) && (year % 100 != 0)) || (year % 400
== 0);
    }

    // Method to get the number of days in a month
    public static int getNumberOfDaysInMonth(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 day of the week the month starts on (0 =
Sunday, 6 = Saturday)
    public static int getStartDay(int month, int year) {
        int d = 1;
        int y0 = year - (14 - month) / 12;
        int x = y0 + y0/4 - y0/100 + y0/400;
        int m0 = month + 12 * ((14 - month) / 12) - 2;
```

```
        int d0 = (d + x + (31 * m0) / 12) % 7;
        return d0;
    }

    // Method to print the calendar
    public static void printCalendar(int month, int year) {
        int startDay = getStartDay(month, year);
        int numOfDays = getNumberOfDaysInMonth(month, year);

        System.out.println("\n   " + getMonthName(month) + " " +
year);
        System.out.println("Sun Mon Tue Wed Thu Fri Sat");

        // Print spaces for the first day
        for (int i = 0; i < startDay; i++) {
            System.out.print("    ");
        }

        // Print the days of the month
        for (int day = 1; day <= numOfDays; day++) {
            System.out.printf("%3d ", day);
            if ((day + startDay) % 7 == 0) {
                System.out.println();
            }
        }
        System.out.println(); // final newline
    }

    // Main method
    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();

        printCalendar(month, year);
    }
}
```

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

```
CODE- import java.util.Arrays;
import java.util.Random;
import java.util.Scanner;

public class CardDeck {

    // Method to initialize the deck of cards
    public static String[] initializeDeck() {
        String[] suits = {"Hearts", "Diamonds", "Clubs", "Spades"};
        String[] ranks = {"2", "3", "4", "5", "6", "7", "8", "9",
"10", "Jack", "Queen", "King", "Ace"};
        String[] deck = new String[suits.length * ranks.length];

        int index = 0;
        for (String suit : suits) {
            for (String rank : ranks) {
```

```
                deck[index++] = rank + " of " + suit;
            }
        }
        return deck;
    }

    // Method to shuffle the deck of cards
    public static void shuffleDeck(String[] deck) {
        Random random = new Random();
        for (int i = 0; i < deck.length; i++) {
            int randomCardNumber = i + random.nextInt(deck.length - i);
            // Swap the current card with the random card
            String temp = deck[i];
            deck[i] = deck[randomCardNumber];
            deck[randomCardNumber] = temp;
        }
    }

    // Method to distribute the cards to players
    public static String[][] distributeCards(String[] deck, int numOfCards, int numOfPlayers) {
        if (numOfCards % numOfPlayers != 0) {
            System.out.println("Cards cannot be evenly distributed among the players.");
            return null;
        }

        int cardsPerPlayer = numOfCards / numOfPlayers;
        String[][] players = new String[numOfPlayers][cardsPerPlayer];

        int cardIndex = 0;
        for (int i = 0; i < numOfPlayers; i++) {
            for (int j = 0; j < cardsPerPlayer; j++) {
                players[i][j] = deck[cardIndex++];
            }
        }
```

```
        return players;
    }

    // Method to print the players and their cards
    public static void printPlayersCards(String[][] players) {
        for (int i = 0; i < players.length; i++) {
            System.out.println("Player " + (i + 1) + ": " +
Arrays.toString(players[i]));
        }
    }

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

        // Initialize the deck
        String[] deck = initializeDeck();
        int numOfCards = deck.length;

        // Shuffle the deck
        shuffleDeck(deck);

        // Get input for number of players and cards per player
        System.out.print("Enter number of players: ");
        int numOfPlayers = scanner.nextInt();
        System.out.print("Enter number of cards to distribute (must be
divisible by number of players): ");
        int numOfCardsToDistribute = scanner.nextInt();

        // Distribute cards to players
        String[][] players = distributeCards(deck,
numOfCardsToDistribute, numOfPlayers);

        if (players != null) {
            // Print the players and their cards
            printPlayersCards(players);
        }
```

```
            scanner.close();
        }
    }
```