CodInClub

Powered By BridgeLabz

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

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.

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


public class UniqueCharactersFinder {
```

```java
public static int findStringLength(String str) {
    if (str == null) {
        return 0;
    }
    int count = 0;
    try {
        while (true) {
            str.charAt(count);
            count++;
        }
    } catch (IndexOutOfBoundsException e) {
        return count;
    }
}

public static char[] findUniqueCharacters(String text) {
    if (text == null || text.isEmpty()) {
        return new char[0];
    }

    int textLength = findStringLength(text);
    char[] uniqueChars = new char[textLength];
    int uniqueCount = 0;

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

        for (int j = 0; j < uniqueCount; j++) {
            if (currentChar == uniqueChars[j]) {
                isUnique = false;
                break;
            }
        }
```

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

        char[] result = new char[uniqueCount];
        for (int i = 0; i < uniqueCount; i++) {
            result[i] = uniqueChars[i];
        }
        return result;
    }

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

        char[] uniqueCharsArray = findUniqueCharacters(inputString);
        System.out.println("Unique characters: " +
Arrays.toString(uniqueCharsArray));
    }
}
```

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.

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

```java
public class FirstNonRepeatingCharacter {

    public static char findFirstNonRepeatingCharacter(String str) {
        if (str == null || str.isEmpty()) {
            return '\0';
        }

        int[] charFrequencies = new int[256];
        int strLength = str.length();

        for (int i = 0; i < strLength; i++) {
            char c = str.charAt(i);
            charFrequencies[c]++;
        }

        for (int i = 0; i < strLength; i++) {
            char c = str.charAt(i);
            if (charFrequencies[c] == 1) {
                return c;
            }
        }

        return '\0';
    }

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

        char firstNonRepeating =
findFirstNonRepeatingCharacter(inputString);
        if (firstNonRepeating != '\0') {
            System.out.println("The first non-repeating character is: " +
firstNonRepeating);
```

```
        } else {
            System.out.println("No non-repeating character 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.

```
import java.util.Scanner;

public class StringUtils {

    public static char findFirstNonRepeatingCharacter(String str) {
        if (str == null || str.isEmpty()) {
            return '\0';
        }

        int[] charFrequencies = new int[256];
        int strLength = str.length();

        for (int i = 0; i < strLength; i++) {
            char c = str.charAt(i);
            charFrequencies[c]++;
        }

        for (int i = 0; i < strLength; i++) {
            char c = str.charAt(i);
            if (charFrequencies[c] == 1) {
```

```
                return c;
            }
        }

        return '\0';
    }

    public static String[][] getCharacterFrequencies(String str) {
        if (str == null || str.isEmpty()) {
            return new String[0][0];
        }

        int[] charFrequencies = new int[256];
        int strLength = str.length();

        for (int i = 0; i < strLength; i++) {
            char c = str.charAt(i);
            charFrequencies[c]++;
        }

        int uniqueCharCount = 0;
        for (int frequency : charFrequencies) {
            if (frequency > 0) {
                uniqueCharCount++;
            }
        }

        String[][] charFrequencyArray = new String[uniqueCharCount][2];
        int arrayIndex = 0;
        for (int i = 0; i < 256; i++) {
            if (charFrequencies[i] > 0) {
                charFrequencyArray[arrayIndex][0] = String.valueOf((char)
i);
                charFrequencyArray[arrayIndex][1] =
String.valueOf(charFrequencies[i]);
                arrayIndex++;
```

```
            }
        }
        return charFrequencyArray;
    }

    public static void displayCharacterFrequencies(String[][]
charFrequencies) {
        if (charFrequencies == null || charFrequencies.length == 0) {
            System.out.println("No characters to display.");
            return;
        }

        System.out.println("Character\tFrequency");
        System.out.println("-------------------------");
        for (String[] row : charFrequencies) {
            System.out.println(row[0] + "\t\t" + row[1]);
        }
    }

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

        char firstNonRepeating =
StringUtils.findFirstNonRepeatingCharacter(inputString);
        if (firstNonRepeating != '\0') {
            System.out.println("The first non-repeating character is: " +
firstNonRepeating);
        } else {
            System.out.println("No non-repeating character found.");
        }

        String[][] frequencyArray =
StringUtils.getCharacterFrequencies(inputString);
```

```
        StringUtils.displayCharacterFrequencies(frequencyArray);
    }
}
```

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.

```
import java.util.Scanner;

public class StringUtilClass {

    public static char findFirstNonRepeatingCharacter(String str) {
        if (str == null || str.isEmpty()) {
            return '\0';
        }

        int[] charFrequencies = new int[256];
        int strLength = str.length();

        for (int i = 0; i < strLength; i++) {
            char c = str.charAt(i);
            charFrequencies[c]++;
        }

        for (int i = 0; i < strLength; i++) {
            char c = str.charAt(i);
            if (charFrequencies[c] == 1) {
                return c;
```

```
            }
        }

        return '\0';
    }

    public static String[][] getCharacterFrequencies(String str) {
        if (str == null || str.isEmpty()) {
            return new String[0][0];
        }

        int[] charFrequencies = new int[256];
        int strLength = str.length();

        for (int i = 0; i < strLength; i++) {
            char c = str.charAt(i);
            charFrequencies[c]++;
        }

        int uniqueCharCount = 0;
        for (int frequency : charFrequencies) {
            if (frequency > 0) {
                uniqueCharCount++;
            }
        }

        String[][] charFrequencyArray = new String[uniqueCharCount][2];
        int arrayIndex = 0;
        for (int i = 0; i < 256; i++) {
            if (charFrequencies[i] > 0) {
                charFrequencyArray[arrayIndex][0] = String.valueOf((char)
i);
                charFrequencyArray[arrayIndex][1] =
String.valueOf(charFrequencies[i]);
                arrayIndex++;
            }
```

```
        }
        return charFrequencyArray;
    }

    public static void displayCharacterFrequencies(String[][]
charFrequencies) {
        if (charFrequencies == null || charFrequencies.length == 0) {
            System.out.println("No characters to display.");
            return;
        }

        System.out.println("Character\tFrequency");
        System.out.println("-------------------------");
        for (String[] row : charFrequencies) {
            System.out.println(row[0] + "\t\t" + row[1]);
        }
    }

    public static char[] uniqueCharacters(String text) {
        if (text == null || text.isEmpty()) {
            return new char[0];
        }

        int textLength = text.length();
        char[] uniqueChars = new char[textLength];
        int uniqueCount = 0;

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

            for (int j = 0; j < uniqueCount; j++) {
                if (currentChar == uniqueChars[j]) {
                    isUnique = false;
                    break;
```

```
                }
            }

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

        char[] result = new char[uniqueCount];
        for (int i = 0; i < uniqueCount; i++) {
            result[i] = uniqueChars[i];
        }
        return result;
    }

    public static String[][] getCharacterFrequenciesUsingUnique(String str)
{
           if (str == null || str.isEmpty()) {
            return new String[0][0];
        }
        char[] uniqueCharsArr = uniqueCharacters(str);
        String[][] charFrequencyArray = new
String[uniqueCharsArr.length][2];
        for (int i = 0; i < uniqueCharsArr.length; i++) {
            int count = 0;
            for (int j = 0; j < str.length(); j++) {
                if (str.charAt(j) == uniqueCharsArr[i]) {
                    count++;
                }
            }
            charFrequencyArray[i][0] = String.valueOf(uniqueCharsArr[i]);
            charFrequencyArray[i][1] = String.valueOf(count);
        }
        return charFrequencyArray;
    }
```

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

        char firstNonRepeating =
StringUtilClass.findFirstNonRepeatingCharacter(inputString);
        if (firstNonRepeating != '\0') {
            System.out.println("The first non-repeating character is: " +
firstNonRepeating);
        } else {
            System.out.println("No non-repeating character found.");
        }

        String[][] frequencyArray =
StringUtilClass.getCharacterFrequencies(inputString);
        StringUtils.displayCharacterFrequencies(frequencyArray);

        String[][] frequencyArrayUnique =
StringUtilClass.getCharacterFrequenciesUsingUnique(inputString);
        System.out.println("\nCharacter Frequencies using Unique
Characters:");
        StringUtils.displayCharacterFrequencies(frequencyArrayUnique);
    }
```

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.

```
import java.util.Scanner;

public class CharacterFrequencyFinder {

   public static String[] getCharacterFrequenciesNestedLoop(String str) {
       if (str == null || str.isEmpty()) {
           return new String[0];
       }

       char[] textChars = str.toCharArray();
       int strLength = textChars.length;
       int[] frequencies = new int[strLength];
       int uniqueCharCount = 0;

       for (int i = 0; i < strLength; i++) {
           if (textChars[i] != '0') {
               int count = 1;
               for (int j = i + 1; j < strLength; j++) {
                   if (textChars[i] == textChars[j]) {
                       count++;
                       textChars[j] = '0';
                   }
               }
               frequencies[uniqueCharCount] = count;
               uniqueCharCount++;
           }
       }
       String[] result = new String[uniqueCharCount * 2];
       int resultIndex = 0;
       for (int i = 0; i < strLength; i++) {
           if (textChars[i] != '0') {
```

```
                result[resultIndex++] = String.valueOf(textChars[i]);
                result[resultIndex++] = String.valueOf(frequencies[i]);
            }
        }
        return result;
    }

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

        String[] frequencyArrayNestedLoop =
CharacterFrequencyFinder.getCharacterFrequenciesNestedLoop(inputString);
        System.out.println("\nCharacter Frequencies using Nested Loop:");
        if (frequencyArrayNestedLoop.length == 0)
             System.out.println("No characters to display");
        else {
            System.out.println("Character\tFrequency");
            System.out.println("-------------------------");
            for (int i = 0; i < frequencyArrayNestedLoop.length; i += 2) {
                System.out.println(frequencyArrayNestedLoop[i] + "\t\t" +
frequencyArrayNestedLoop[i + 1]);
            }
        }
    }
}
```

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

```
import java.util.Scanner;

public class PalindromeChecker {

   public static boolean isPalindromeIterative(String text) {
       if (text == null) {
           return false;
       }
       String cleanText = text.replaceAll("[^a-zA-Z0-9]",
"").toLowerCase();
       int start = 0;
       int end = cleanText.length() - 1;

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

   public static boolean isPalindromeRecursive(String text) {
```

```
        if (text == null) {
            return false;
        }
        String cleanText = text.replaceAll("[^a-zA-Z0-9]",
"").toLowerCase();
        return isPalindromeRecursiveHelper(cleanText, 0, cleanText.length()
- 1);
    }

    private static boolean isPalindromeRecursiveHelper(String text, int
start, int end) {
        if (start >= end) {
            return true;
        }
        if (text.charAt(start) != text.charAt(end)) {
            return false;
        }
        return isPalindromeRecursiveHelper(text, start + 1, end - 1);
    }

    public static char[] reverseString(String str) {
        if (str == null) {
            return null;
        }
        int length = str.length();
        char[] reversed = new char[length];
        for (int i = 0; i < length; i++) {
            reversed[i] = str.charAt(length - 1 - i);
        }
        return reversed;
    }

    public static boolean isPalindromeCharArray(String text) {
        if (text == null) {
            return false;
        }
```

```
        String cleanText = text.replaceAll("[^a-zA-Z0-9]",
"").toLowerCase();
        char[] originalChars = cleanText.toCharArray();
        char[] reversedChars = reverseString(cleanText);

        if (reversedChars == null)
            return true;

        if (originalChars.length != reversedChars.length) {
            return false;
        }

        for (int i = 0; i < originalChars.length; i++) {
            if (originalChars[i] != reversedChars[i]) {
                return false;
            }
        }
        return true;
    }

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

        boolean isPalindrome1 = isPalindromeIterative(inputString);
        System.out.println("Is palindrome (Iterative): " + isPalindrome1);

        boolean isPalindrome2 = isPalindromeRecursive(inputString);
        System.out.println("Is palindrome (Recursive): " + isPalindrome2);

        boolean isPalindrome3 = isPalindromeCharArray(inputString);
        System.out.println("Is palindrome (Character Array): " +
isPalindrome3);
    }
```

```
}
```

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.

```
import java.util.Scanner;

public class AnagramChecker {

   public static boolean areAnagrams(String text1, String text2) {
       if (text1 == null || text2 == null) {
           return false;
       }

       String cleanText1 = text1.replaceAll("[^a-zA-Z0-9]",
"").toLowerCase();
       String cleanText2 = text2.replaceAll("[^a-zA-Z0-9]",
"").toLowerCase();

       if (cleanText1.length() != cleanText2.length()) {
           return false;
       }

       int[] charFrequencies1 = new int[256];
       int[] charFrequencies2 = new int[256];

       for (char c : cleanText1.toCharArray()) {
           charFrequencies1[c]++;
       }
       for (char c : cleanText2.toCharArray()) {
           charFrequencies2[c]++;
```

```
        }

        for (int i = 0; i < 256; i++) {
            if (charFrequencies1[i] != charFrequencies2[i]) {
                return false;
            }
        }
        return true;
    }

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

        boolean areAnagrams = areAnagrams(text1, text2);
        System.out.println("Are the two texts anagrams? " + areAnagrams);
    }
}
```

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 × ((14 − m) / 12) − 2

d0 = (d + x + 31m0 / 12) mod 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

```
import java.util.Scanner;

public class CalendarGenerator {

    public static String getMonthName(int month) {
        String[] months = {
            "", // Index 0 is not used
            "January", "February", "March", "April",
            "May", "June", "July", "August",
            "September", "October", "November", "December"
        };
        return (month >= 1 && month <= 12) ? months[month] : "Invalid
Month";
    }

    public static int getDaysInMonth(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 (month >= 1 && month <= 12) ? days[month] : -1; // -1 for
invalid month
    }

    public static boolean isLeapYear(int year) {
        return (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0);
    }

    public static int getFirstDayOfMonth(int year, int month) {
        int day = 1;
        int y = year - (14 - month) / 12;
        int x = y + y / 4 - y / 100 + y / 400;
        int m = month + 12 * ((14 - month) / 12) - 2;
        int d = (day + x + (31 * m) / 12) % 7;
        return d;
    }

    public static void displayCalendar(int year, int month) {
```

```
        String monthName = getMonthName(month);
        int daysInMonth = getDaysInMonth(month, year);
        int firstDay = getFirstDayOfMonth(year, month);

        if (daysInMonth == -1) {
            System.out.println("Invalid month or year.");
            return;
        }

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

        for (int i = 0; i < firstDay; i++) {
            System.out.print("    ");
        }

        for (int day = 1; day <= daysInMonth; day++) {
            System.out.printf("%3d ", day);
            if ((firstDay + day) % 7 == 0) {
                System.out.println();
            }
        }
        System.out.println();
    }

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

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

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

```
import java.util.Random;
import java.util.Scanner;

public class DeckOfCards {

   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" };
       int numOfCards = suits.length * ranks.length;
       String[] deck = new String[numOfCards];
       int index = 0;

       for (String suit : suits) {
           for (String rank : ranks) {
               deck[index++] = rank + " of " + suit;
           }
       }
       return deck;
   }

   public static String[] shuffleDeck(String[] deck) {
       int n = deck.length;
       Random random = new Random();

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

```
            int randomCardNumber = i + random.nextInt(n - i);
            String temp = deck[i];
            deck[i] = deck[randomCardNumber];
            deck[randomCardNumber] = temp;
        }
        return deck;
    }

    public static String[][] distributeCards(String[] deck, int numPlayers,
int cardsPerPlayer) {
        int totalCards = deck.length;
        if (numPlayers <= 0 || cardsPerPlayer <= 0 || numPlayers *
cardsPerPlayer > totalCards) {
            System.out.println("Cannot distribute cards with these
parameters.");
            return null;
        }

        String[][] players = new String[numPlayers][cardsPerPlayer];
        int cardIndex = 0;
        for (int i = 0; i < numPlayers; i++) {
            for (int j = 0; j < cardsPerPlayer; j++) {
                players[i][j] = deck[cardIndex++];
            }
        }
        return players;
    }

    public static void printDistributedCards(String[][] players) {
        if (players == null) {
            return;
        }
        for (int i = 0; i < players.length; i++) {
            System.out.println("Player " + (i + 1) + ":");
            for (int j = 0; j < players[i].length; j++) {
                System.out.println(players[i][j]);
```

```
            }
            System.out.println();
        }
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the number of players: ");
        int numPlayers = scanner.nextInt();
        System.out.print("Enter the number of cards per player: ");
        int cardsPerPlayer = scanner.nextInt();
        scanner.close();

        String[] deck = initializeDeck();
        String[] shuffledDeck = shuffleDeck(deck);
        String[][] distributedCards = distributeCards(shuffledDeck,
numPlayers, cardsPerPlayer);

        printDistributedCards(distributedCards);
    }
}
```