

## Level 2 Practice Programs

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

**Hint =>**

- a. Take user input using the `Scanner next()` method
- b. Create a method to find and return a string's length without using the built-in `length()` method. The logic for this is to use the infinite loop to count each character till the `charAt()` method throws a runtime exception, handles the exception, and then return the count
- c. The main function calls the user-defined method as well as the built-in `length()` method and displays the result

```
import java.util.Scanner;

public class StringLengthFinder {
    public static int findStringLength(String str) {
        if (str == null) {
            throw new NullPointerException("Input string cannot be null.");
        }
        int count = 0;
        try {
            while (true) {
                str.charAt(count); // Keep accessing characters until an
                // exception occurs
                count++;
            }
        } catch (IndexOutOfBoundsException e) {
            // This exception is expected when we go beyond the string's
            // end.
            return count;
        }
    }

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

```
String inputString = scanner.next(); // Use next() to read one
word. Use nextLine() to read a whole line.

scanner.close();

// Find the length using the custom method
int customLength = findStringLength(inputString);

System.out.println("Length of the string (using custom method): " +
customLength);

// Find the length using the built-in length() method
int builtInLength = inputString.length();

System.out.println("Length of the string (using built-in length()
method): " + builtInLength);
}
}
```

2. Write a program to split the text into words, compare the result with the split() method and display the result

**Hint =>**

- a. Take user input using the **Scanner nextLine()** method
- b. Create a Method to find the length of the String without using the built-in length() method.
- c. Create a Method to split the text into words using the charAt() method without using the String built-in **split()** method and return the words. Use the following logic
  - i. Firstly Count the number of words in the text and create an array to store the indexes of the spaces for each word in a 1D array
  - ii. Then Create an array to store the words and use the indexes to extract the words
- d. Create a method to compare the two String arrays and return a boolean
- e. The main function calls the user-defined method and the built-in **split()** method. Call the user defined method to compare the two string arrays and display the result

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

public class StringSplitter {

    public static int findStringLength(String str) {
        if (str == null) {
            throw new NullPointerException("Input string cannot be null.");
        }
        int count = 0;
```

```

try {
    while (true) {
        str.charAt(count);
        count++;
    }
} catch (IndexOutOfBoundsException e) {
    return count;
}
}

public static String[] splitText(String text) {
    if (text == null) {
        return new String[0]; // Return an empty array for null input
    }

    int textLength = findStringLength(text);
    int wordCount = 0;
    boolean inWord = false;

    // Count the number of words
    for (int i = 0; i < textLength; i++) {
        if (text.charAt(i) == ' ' || text.charAt(i) == '\t' ||
            text.charAt(i) == '\n' || text.charAt(i) == '\r') {
            inWord = false;
        } else if (!inWord) {
            inWord = true;
            wordCount++;
        }
    }

    // Store the indexes of spaces
    int[] spaceIndices = new int[wordCount]; // At most, there will be
wordCount -1 spaces. But allocate for wordCount for simplicity

    String[] words = new String[wordCount];
    int wordIndex = 0;
    int startIndex = 0;

```

```

        inWord = false; //reset
        for (int i = 0; i < textLength; i++) {
            if (text.charAt(i) == ' ' || text.charAt(i) == '\t' ||
text.charAt(i) == '\n' || text.charAt(i) == '\r') {
                if (inWord) {
                    words[wordIndex++] = text.substring(startIndex, i);
                    inWord = false;
                }
                startIndex = i + 1; //start of next word
            } else if (!inWord) {
                inWord = true;
                startIndex = i;
            }
        }
        //handle the last word.
        if(inWord && wordIndex < wordCount){
            words[wordIndex] = text.substring(startIndex, textLength);
        }
        return words;
    }

    public static boolean compareStringArrays(String[] array1, String[]
array2) {
        if (array1 == null && array2 == null) {
            return true;
        }
        if (array1 == null || array2 == null) {
            return false;
        }
        if (array1.length != array2.length) {
            return false;
        }
        for (int i = 0; i < array1.length; i++) {
            if (!array1[i].equals(array2[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();

        // Split the text using the custom method
        String[] customSplitResult = splitText(inputString);

        System.out.println("Words (using custom split method): " +
Arrays.toString(customSplitResult));

        // Split the text using the built-in split() method
        String[] builtInSplitResult = inputString.split("\\s+"); // Split
by any whitespace

        System.out.println("Words (using built-in split() method): " +
Arrays.toString(builtInSplitResult));

        // Compare the two String arrays
        boolean areEqual = compareStringArrays(customSplitResult,
builtInSplitResult);

        System.out.println("Are the two arrays equal? " + areEqual);
    }
}

```

3. Write a program to split the text into words and return the words along with their lengths in a 2D array

**Hint =>**

- a. Take user input using the **Scanner** **nextLine()** method
- b. Create a Method to split the text into words using the **charAt()** method without using the String built-in **split()** method and return the words.
- c. Create a method to find and return a string's length without using the **length()** method.
- d. Create a method to take the word array and return a 2D String array of the word and its corresponding length. Use String built-in function **String.valueOf()** to generate the String value for the number
- e. The main function calls the user-defined method and displays the result in a tabular format. During display make sure to convert the length value from String to Integer and then display

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

```
public class WordLengthArray {
    public static int findStringLength(String str) {
        if (str == null) {
            throw new NullPointerException("Input string cannot be null.");
        }
        int count = 0;
        try {
            while (true) {
                str.charAt(count);
                count++;
            }
        } catch (IndexOutOfBoundsException e) {
            return count;
        }
    }

    public static String[] splitText(String text) {
        if (text == null) {
            return new String[0]; // Return an empty array for null input
        }

        int textLength = findStringLength(text);
        int wordCount = 0;
        boolean inWord = false;

        // Count the number of words
        for (int i = 0; i < textLength; i++) {
            if (text.charAt(i) == ' ' || text.charAt(i) == '\t' ||
                text.charAt(i) == '\n' || text.charAt(i) == '\r') {
                inWord = false;
            } else if (!inWord) {
                inWord = true;
                wordCount++;
            }
        }
    }
}
```

```
// Store the indexes of spaces
String[] words = new String[wordCount];
int wordIndex = 0;
int startIndex = 0;
inWord = false;

for (int i = 0; i < textLength; i++) {
    if (text.charAt(i) == ' ' || text.charAt(i) == '\t' ||
text.charAt(i) == '\n' || text.charAt(i) == '\r') {
        if (inWord) {
            words[wordIndex++] = text.substring(startIndex, i);
            inWord = false;
        }
        startIndex = i + 1;
    } else if (!inWord) {
        inWord = true;
        startIndex = i;
    }
}

// Handle the last word
if (inWord && wordIndex < wordCount) {
    words[wordIndex] = text.substring(startIndex, textLength);
}

return words;
}

public static String[][] getWordLengthsArray(String[] words) {
    if (words == null) {
        return new String[0][0]; // Return an empty 2D array for null
input
    }

    int wordCount = words.length;
    String[][] wordLengths = new String[wordCount][2]; // 2 columns:
word, length

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

```

        wordLengths[i][0] = words[i];
        wordLengths[i][1] = String.valueOf(findStringLength(words[i]));
// Convert length to String
    }
    return wordLengths;
}

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

    // Split the text into words
    String[] words = splitText(inputString);

    // Get the 2D array of words and their lengths
    String[][] wordLengthsArray = getWordLengthsArray(words);

    // Display the result in a tabular format
    System.out.println("\nWord\tLength");
    System.out.println("-----");
    for (String[] row : wordLengthsArray) {
        // Convert the length back to an Integer for display
        int length = Integer.parseInt(row[1]);
        System.out.println(row[0] + "\t" + length);
    }
}
}

```

4. Write a program to split the text into words and find the shortest and longest strings in a given text

**Hint =>**

- Take user input using the **Scanner** *nextLine()* method
- Create a Method to split the text into words using the *charAt()* method without using the String built-in **split()** method and return the words.
- Create a method to find and return a string's length without using the *length()* method.



- d. Create a method to take the word array and return a 2D String array of the word and its corresponding length. Use String built-in function String.valueOf() to generate the String value for the number
- e. Create a Method that takes the 2D array of word and corresponding length as parameters, find the shortest and longest string and return them in an 1D int array.
- f. The main function calls the user-defined methods and displays the result.

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

public class ShortestLongestString {

    public static int findStringLength(String str) {
        if (str == null) {
            throw new NullPointerException("Input string cannot be null.");
        }
        int count = 0;
        try {
            while (true) {
                str.charAt(count);
                count++;
            }
        } catch (IndexOutOfBoundsException e) {
            return count;
        }
    }

    public static String[] splitText(String text) {
        if (text == null) {
            return new String[0]; // Return an empty array for null input
        }

        int textLength = findStringLength(text);
        int wordCount = 0;
        boolean inWord = false;

        // Count the number of words
```

```

    for (int i = 0; i < textLength; i++) {
        if (text.charAt(i) == ' ' || text.charAt(i) == '\t' ||
text.charAt(i) == '\n' || text.charAt(i) == '\r') {
            inWord = false;
        } else if (!inWord) {
            inWord = true;
            wordCount++;
        }
    }

    // Store the indexes of spaces
    String[] words = new String[wordCount];
    int wordIndex = 0;
    int startIndex = 0;
    inWord = false;

    for (int i = 0; i < textLength; i++) {
        if (text.charAt(i) == ' ' || text.charAt(i) == '\t' ||
text.charAt(i) == '\n' || text.charAt(i) == '\r') {
            if (inWord) {
                words[wordIndex++] = text.substring(startIndex, i);
                inWord = false;
            }
            startIndex = i + 1;
        } else if (!inWord) {
            inWord = true;
            startIndex = i;
        }
    }

    // Handle the last word.
    if (inWord && wordIndex < wordCount) {
        words[wordIndex] = text.substring(startIndex, textLength);
    }

    return words;
}

```

```

public static String[][] getWordLengthsArray(String[] words) {
    if (words == null) {
        return new String[0][0]; // Return an empty 2D array for null
input
    }

    int wordCount = words.length;

    String[][] wordLengths = new String[wordCount][2]; // 2 columns:
word, length

    for (int i = 0; i < wordCount; i++) {
        wordLengths[i][0] = words[i];
        wordLengths[i][1] = String.valueOf(findStringLength(words[i]));
// Convert length to String
    }
    return wordLengths;
}

public static String[] findShortestAndLongest(String[][] wordLengths) {
    if (wordLengths == null || wordLengths.length == 0) {
        return null; // Handle null or empty input
    }

    String shortest = wordLengths[0][0]; // Initialize with the first
word
    String longest = wordLengths[0][0]; // Initialize with the first
word

    for (int i = 1; i < wordLengths.length; i++) {
        String currentWord = wordLengths[i][0];
        int currentLength = findStringLength(currentWord);

        if (currentLength < findStringLength(shortest)) {
            shortest = currentWord;
        }

        if (currentLength > findStringLength(longest)) {
            longest = currentWord;
        }
    }
}

```

```

    }

    }

    return new String[] { shortest, longest };
}

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

    // Split the text into words
    String[] words = splitText(inputString);

    // Get the 2D array of words and their lengths
    String[][] wordLengthsArray = getWordLengthsArray(words);

    // Find the shortest and longest strings
    String[] result = findShortestAndLongest(wordLengthsArray);

    // Display the result
    if (result != null) {
        System.out.println("Shortest string: " + result[0]);
        System.out.println("Longest string: " + result[1]);
    } else {
        System.out.println("No words found in the input.");
    }
}
}

```

5. Write a program to find vowels and consonants in a string and display the count of Vowels and Consonants in the string

**Hint =>**

- a. Create a method to check if the character is a vowel or consonant and return the result. The logic used here is as follows:
  - i. Convert the character to lowercase if it is an uppercase letter using the ASCII values of the characters

- ii. Check if the character is a vowel or consonant and return Vowel, Consonant, or Not a Letter
- b. Create a Method to Method to find vowels and consonants in a string using charAt() method and finally return the count of vowels and consonants in an array
- c. Finally, the main function takes user inputs, calls the user-defined methods, and displays the result.

```
import java.util.Scanner;

public class VowelConsonantCounter {

    public static String checkCharacterType(char ch) {
        // Convert to lowercase
        if (ch >= 'A' && ch <= 'Z') {
            ch = (char) (ch + 32); // Add 32 to get lowercase ASCII value
        }

        if (ch >= 'a' && ch <= 'z') {
            if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch ==
'u') {
                return "Vowel";
            } else {
                return "Consonant";
            }
        } else {
            return "Not a Letter";
        }
    }

    public static int[] countVowelsAndConsonants(String str) {
        if (str == null || str.isEmpty()) {
            return new int[] {0, 0}; // Handle null or empty string
        }

        int vowelCount = 0;
        int consonantCount = 0;
```

```

        int strLength = str.length(); //avoid calculating length in every
loop iteration.

        for (int i = 0; i < strLength; i++) {
            char ch = str.charAt(i);
            String charType = checkCharacterType(ch);
            if (charType.equals("Vowel")) {
                vowelCount++;
            } else if (charType.equals("Consonant")) {
                consonantCount++;
            }
            // else, do nothing, it is not a letter
        }
        return new int[] { vowelCount, consonantCount };
    }

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

        int[] counts = countVowelsAndConsonants(inputString);

        if (counts != null) {
            System.out.println("Vowel Count: " + counts[0]);
            System.out.println("Consonant Count: " + counts[1]);
        } else {
            System.out.println("Invalid input string."); // Handle the null
case, though it should not happen
        }
    }
}

```

6. Write a program to find vowels and consonants in a string and display the character type - Vowel, Consonant, or Not a Letter

**Hint =>**

- a. Create a method to check if the character is a vowel or consonant and return the result. The logic used here is as follows:
  - i. Convert the character to lowercase if it is an uppercase letter using the ASCII values of the characters
  - ii. Check if the character is a vowel or consonant and return Vowel, Consonant, or Not a Letter
- b. Create a Method to find vowels and consonants in a string using charAt() method and return the character and vowel or consonant in a 2D array
- c. Create a Method to display the 2D Array of Strings in a Tabular Format
- d. Finally, the main function takes user inputs, calls the user-defined methods, and displays the result.

```
import java.util.Scanner;

public class VowelConsonantDisplay {

    public static String checkCharacterType(char ch) {
        if (ch >= 'A' && ch <= 'Z') {
            ch = (char) (ch + 32);
        }

        if (ch >= 'a' && ch <= 'z') {
            if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u') {
                return "Vowel";
            } else {
                return "Consonant";
            }
        } else {
            return "Not a Letter";
        }
    }

    public static String[][] getCharacterTypes(String str) {
        if (str == null) {
            return new String[0][0];
        }

        int strLength = str.length();
```

```
String[][] charTypes = new String[strLength][2];

for (int i = 0; i < strLength; i++) {
    char ch = str.charAt(i);
    charTypes[i][0] = String.valueOf(ch);
    charTypes[i][1] = checkCharacterType(ch);
}

return charTypes;
}

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

    System.out.println("Character\tType");
    System.out.println("-----");
    for (String[] row : charTypes) {
        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();

    String[][] charTypesArray = getCharacterTypes(inputString);
    displayCharacterTypes(charTypesArray);
}
}
```

7. Write a program to trim the leading and trailing spaces from a string using the **`charAt()`** method



**Hint =>**

- Create a method to trim the leading and trailing spaces from a string using the `charAt()` method. Inside the method run a couple of loops to trim leading and trailing spaces and determine the starting and ending points with no spaces. Return the start point and end point in an array
- Write a method to create a substring from a string using the `charAt()` method with the string, start, and end index as the parameters
- Write a method to compare two strings using the `charAt()` method and return a boolean result
- The main function calls the user-defined trim and substring methods to get the text after trimming the leading and trailing spaces. Post that use the String built-in method `trim()` to trim spaces and compare the two strings. And finally display the result

```
import java.util.Scanner;

public class StringTrimmer {

    public static int[] trimSpaces(String str) {
        if (str == null || str.isEmpty()) {
            return new int[] { 0, -1 };
        }

        int start = 0;
        int end = str.length() - 1;

        while (start <= end && Character.isWhitespace(str.charAt(start))) {
            start++;
        }

        while (end >= start && Character.isWhitespace(str.charAt(end))) {
            end--;
        }

        return new int[] { start, end };
    }

    public static String substring(String str, int start, int end) {
        if (str == null) {
            return null;
        }
    }
```

```
}

if (start > end) {
    return "";
}

if (start < 0) {
    start = 0;
}

if (end >= str.length()) {
    end = str.length() - 1;
}

StringBuilder sb = new StringBuilder();
for (int i = start; i <= end; i++) {
    sb.append(str.charAt(i));
}
return sb.toString();
}

public static boolean compareStrings(String str1, String str2) {
    if (str1 == null && str2 == null) {
        return true;
    }
    if (str1 == null || str2 == null) {
        return false;
    }

    int len1 = str1.length();
    int len2 = str2.length();

    if (len1 != len2) {
        return false;
    }

    for (int i = 0; i < len1; i++) {
        if (str1.charAt(i) != str2.charAt(i)) {
```

```

        return false;
    }
}

return true;
}

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

    int[] trimIndices = trimSpaces(inputString);
    String trimmedStringCustom = substring(inputString, trimIndices[0],
trimIndices[1]);

    String trimmedStringBuiltIn = inputString.trim();

    boolean areEqual = compareStrings(trimmedStringCustom,
trimmedStringBuiltIn);

    System.out.println("Trimmed String (Custom): \"" +
trimmedStringCustom + "\"");
    System.out.println("Trimmed String (Built-in): \"" +
trimmedStringBuiltIn + "\"");
    System.out.println("Strings are equal: " + areEqual);
}

```

8. Write a program to take user input for the age of all 10 students in a class and check whether the student can vote depending on his/her age is greater or equal to 18.

**Hint =>**

- Create a method to define the random 2-digit age of several students provided as method parameters and return a 1D array of ages of n students
- Create a method that takes an array of age as a parameter and returns a 2D String array of age and a boolean true or false to indicate can and cannot vote. Inside the method firstly validate the age for a negative number, if a negative cannot vote. For valid age check for age is 18 or above to set true to indicate can vote.
- Create a method to display the 2D array in a tabular format.
- Finally, the main function takes user inputs, calls the user-defined methods, and displays the result.

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

public class VotingEligibility {

    public static int[] generateRandomAges(int numStudents) {
        if (numStudents <= 0) {
            return null;
        }
        int[] ages = new int[numStudents];
        for (int i = 0; i < numStudents; i++) {
            ages[i] = (int) (Math.random() * 90 + 10); // Generate age
between 10 and 99
        }
        return ages;
    }

    public static String[][] checkVotingEligibility(int[] ages) {
        if (ages == null) {
            return null;
        }
        String[][] eligibility = new String[ages.length][2];
        for (int i = 0; i < ages.length; i++) {
            if (ages[i] < 0) {
                eligibility[i][0] = String.valueOf(ages[i]);
                eligibility[i][1] = "false"; // Cannot vote if age is
negative
            } else {
                eligibility[i][0] = String.valueOf(ages[i]);
                eligibility[i][1] = (ages[i] >= 18) ? "true" : "false";
            }
        }
        return eligibility;
    }

    /**
     * Displays the voting eligibility information in a tabular format.

```

```

*
* @param eligibility A 2D String array containing age and eligibility.
*/
public static void displayEligibility(String[][] eligibility) {
    if (eligibility == null) {
        System.out.println("No eligibility data to display.");
        return;
    }
    System.out.println("Age\tCan Vote");
    System.out.println("-----");
    for (String[] row : eligibility) {
        System.out.println(row[0] + "\t" + row[1]);
    }
}

public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    int numStudents = 10;
    int[] studentAges = new int[numStudents];

    for (int i = 0; i < numStudents; i++) {
        System.out.print("Enter age for student " + (i + 1) + ": ");
        try {
            studentAges[i] = scanner.nextInt();
        } catch (InputMismatchException e) {
            System.out.println("Invalid input. Please enter a valid
integer for age.");
            scanner.next(); // Consume the invalid input
            i--; // Repeat the current iteration to get the correct
input
        }
    }
    scanner.close();

    String[][] eligibilityList = checkVotingEligibility(studentAges);
    displayEligibility(eligibilityList);
}

```

```
}
}
```

9. Rock-Paper-Scissors is a game played between a minimum of two players. Each player can choose either rock, paper, or scissors. Here the game is played between a user and a computer. Based on the rules, either a player or a computer will win. Show the stats of player and computer win in a tabular format across multiple games. Also, show the winning percentage between the player and the computer.

**Hint =>**

- The rule is:** rock-scissors: rock will win (rock crushes scissors); rock-paper: paper wins (paper covers rock); scissors-paper: scissors win (scissors cuts paper)
- Create a Method to find the Computer Choice using the Math.random
- Create a Method to find the winner between the user and the computer
- Create a Method to find the average and percentage of wins for the user and the computer and return a String 2D array
- Create a Method to display the results of every game and also display the average and percentage wins
- In the main take user input for the number of games and call methods to display results

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

public class RockPaperScissors {

    public static String getComputerChoice() {
        Random random = new Random();
        int choice = random.nextInt(3);
        if (choice == 0) {
            return "rock";
        } else if (choice == 1) {
            return "paper";
        } else {
            return "scissors";
        }
    }

    public static String determineWinner(String userChoice, String
computerChoice) {
        if (userChoice.equals(computerChoice)) {
            return "tie";
        } else if (userChoice.equals("rock")) {
```

```

        if (computerChoice.equals("scissors")) {
            return "user";
        } else {
            return "computer";
        }
    } else if (userChoice.equals("paper")) {
        if (computerChoice.equals("rock")) {
            return "user";
        } else {
            return "computer";
        }
    } else { // userChoice is scissors
        if (computerChoice.equals("paper")) {
            return "user";
        } else {
            return "computer";
        }
    }
}

public static String[][] calculateStats(int[] userWins, int[]
computerWins, int numberOfGames) {
    String[][] stats = new String[3][2];
    int totalUserWins = 0;
    int totalComputerWins = 0;

    for (int i = 0; i < numberOfGames; i++) {
        totalUserWins += userWins[i];
        totalComputerWins += computerWins[i];
    }

    double userWinPercentage = (double) totalUserWins / numberOfGames *
100;
    double computerWinPercentage = (double) totalComputerWins /
numberOfGames * 100;

```

```

        stats[0][0] = "User Wins";
        stats[0][1] = String.valueOf(totalUserWins);
        stats[1][0] = "Computer Wins";
        stats[1][1] = String.valueOf(totalComputerWins);
        stats[2][0] = "User Win %";
        stats[2][1] = String.format("%.2f", userWinPercentage) + "%";

        return stats;
    }

    public static void displayResults(String[] userChoices, String[]
computerChoices, String[] results, String[][] stats, int numberOfGames) {
        System.out.println("\nGame Results:");
        System.out.println("-----");
        System.out.println("Game\tUser Choice\tComputer Choice\tResult");
        System.out.println("-----");

        for (int i = 0; i < numberOfGames; i++) {
            System.out.println((i + 1) + "\t" + userChoices[i] + "\t\t" +
computerChoices[i] + "\t\t" + results[i]);
        }
        System.out.println("\nGame Stats:");
        System.out.println("-----");
        for (String[] row : stats) {
            System.out.println(row[0] + ": " + row[1]);
        }
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the number of games to play: ");
        int numberOfGames = scanner.nextInt();
        scanner.nextLine();

        String[] userChoices = new String[numberOfGames];
        String[] computerChoices = new String[numberOfGames];
    }

```



```
String[] results = new String[numberOfGames];
int[] userWins = new int[numberOfGames];
int[] computerWins = new int[numberOfGames];

for (int i = 0; i < numberOfGames; i++) {
    System.out.print("Enter your choice (rock, paper, or scissors):");
    String userChoice = scanner.nextLine().toLowerCase();
    while (!userChoice.equals("rock") && !userChoice.equals("paper") && !userChoice.equals("scissors")) {
        System.out.println("Invalid choice. Please enter rock, paper, or scissors:");
        userChoice = scanner.nextLine().toLowerCase();
    }
    userChoices[i] = userChoice;
    String computerChoice = getComputerChoice();
    computerChoices[i] = computerChoice;
    String result = determineWinner(userChoice, computerChoice);
    results[i] = result;

    if (result.equals("user")) {
        userWins[i] = 1;
    } else if (result.equals("computer")) {
        computerWins[i] = 1;
    } else {
        userWins[i] = 0;
        computerWins[i] = 0;
    }
}

scanner.close();

String[][] stats = calculateStats(userWins, computerWins, numberOfGames);

displayResults(userChoices, computerChoices, results, stats, numberOfGames);
}
```

10. Create a program to take input marks of students in 3 subjects physics, chemistry, and maths. Compute the percentage and then calculate the grade as shown in figure below

Grade	Remarks	Marks
A	(Level 4, above agency-normalized standards)	80% and above
B	(Level 3, at agency-normalized standards)	70-79%
C	(Level 2, below, but approaching agency-normalized standards)	60-69%
D	(Level 1, well below agency-normalized standards)	50-59%
E	(Level 1- , too below agency-normalized standards)	40-49%
R	(Remedial standards)	39% and below

Hint =>

- Write a method to generate random 2-digit scores for Physics, Chemistry and Math (PCM) for the students and return the scores. This method returns a 2D array with PCM scores for all students
- Write a Method to calculate the total, average, and percentages for each student and return a 2D array with the corresponding values. Please ensure to round off the values to 2 Digits using **Math.round()** method
- Write a Method to calculate the grade based on the percentage as shown in the ref table and return a 2D array of students' grade
- Finally write a Method to display the scorecard of all students with their scores, total, average, percentage, and grade in a tabular format.

```
import java.util.Scanner;

public class StudentGradeCalculator {

    public static int[][] generateRandomScores(int numStudents) {
        if (numStudents <= 0) {
            return null;
        }
        int[][] scores = new int[numStudents][3];
        for (int i = 0; i < numStudents; i++) {
            scores[i][0] = (int) (Math.random() * 90 + 10);
            scores[i][1] = (int) (Math.random() * 90 + 10);
            scores[i][2] = (int) (Math.random() * 90 + 10);
        }
        return scores;
    }

    public static double[][] calculateStats(int[][] scores) {
```

```

    if (scores == null) {
        return null;
    }
    int numStudents = scores.length;
    double[][] stats = new double[numStudents][3];
    for (int i = 0; i < numStudents; i++) {
        int total = scores[i][0] + scores[i][1] + scores[i][2];
        double average = (double) total / 3;
        double percentage = (average / 100) * 100;

        stats[i][0] = total;
        stats[i][1] = Math.round(average * 100.0) / 100.0;
        stats[i][2] = Math.round(percentage * 100.0) / 100.0;
    }
    return stats;
}

public static String[] calculateGrades(double[][] stats) {
    if (stats == null) {
        return null;
    }
    int numStudents = stats.length;
    String[] grades = new String[numStudents];
    for (int i = 0; i < numStudents; i++) {
        double percentage = stats[i][2];
        if (percentage >= 90) {
            grades[i] = "A+";
        } else if (percentage >= 80) {
            grades[i] = "A";
        } else if (percentage >= 70) {
            grades[i] = "B+";
        } else if (percentage >= 60) {
            grades[i] = "B";
        } else if (percentage >= 50) {
            grades[i] = "C";
        } else {
            grades[i] = "Fail";
        }
    }
    return grades;
}

```

```

    public static void displayScorecard(int[][] scores, double[][] stats,
String[] grades) {
        if (scores == null || stats == null || grades == null) {
            System.out.println("No data to display.");
            return;
        }

        System.out.println("-----");
        System.out.println("Student\tPhysics\tChemistry\tMaths\tTotal\tAverage\tPe
rcentage\tGrade");

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

        for (int i = 0; i < scores.length; i++) {
            System.out.println((i + 1) + "\t" + scores[i][0] + "\t\t" +
scores[i][1] + "\t\t" + scores[i][2] + "\t\t"
                + (int) stats[i][0] + "\t\t" + stats[i][1] + "\t\t" +
stats[i][2] + "\t\t" + grades[i]);
        }

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

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the number of students: ");
        int numStudents = scanner.nextInt();
        scanner.close();

        int[][] studentScores = generateRandomScores(numStudents);
        double[][] studentStats = calculateStats(studentScores);
        String[] studentGrades = calculateGrades(studentStats);

        displayScorecard(studentScores, studentStats, studentGrades);
    }
}

```