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

CODE-

```
import java.util.Scanner;


public class StringLengthWithoutLengthMethod {


    // Method to find the length of the string without using the
built-in length() method
    public static int getLengthWithoutMethod(String str) {
        int count = 0;
        try {
            while (true) {
                str.charAt(count);  // Try to access each character by
index
                count++;  // Increment count as long as charAt() does
not throw an exception
            }
        } catch (StringIndexOutOfBoundsException e) {
```

```
            // Exception will be thrown when count exceeds the string
length, we return the count
            return count;
        }
    }

    public static void main(String[] args) {
        // Create a scanner object to take input from the user
        Scanner scanner = new Scanner(System.in);

        // Prompt user for a string input
        System.out.println("Enter a string: ");
        String userInput = scanner.next();

        // Call the method to get the length without using the
built-in length() method
        int lengthWithoutMethod = getLengthWithoutMethod(userInput);

        // Call the built-in length() method
        int lengthWithMethod = userInput.length();

        // Display the results
        System.out.println("Length of the string using custom method:
" + lengthWithoutMethod);
        System.out.println("Length of the string using built-in
length() method: " + lengthWithMethod);

        scanner.close();  // Close the scanner
    }
}
```

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

CODE-

```
import java.util.Scanner;


public class TextSplitterWithoutSplitMethod {


    // Method to find the length of the String without using the
built-in length() method
    public static int getLengthWithoutMethod(String str) {
        int count = 0;
        try {
            while (true) {
                str.charAt(count);  // Try to access each character by
index
                count++;  // Increment count as long as charAt() does
not throw an exception
            }
        } catch (StringIndexOutOfBoundsException e) {
            // Exception will be thrown when count exceeds the string
length, we return the count
            return count;
        }
    }
```

```
    // Method to split the text into words without using the split()
method
    public static String[] splitTextWithoutSplit(String text) {
        int length = getLengthWithoutMethod(text);
        int wordCount = 0;

        // Count words by checking spaces
        for (int i = 0; i < length; i++) {
            if (text.charAt(i) == ' ') {
                wordCount++;
            }
        }
        wordCount++;  // The number of words is one more than the
number of spaces

        // Create an array to store the indexes of spaces
        int[] spaceIndexes = new int[wordCount - 1];
        int spaceIndex = 0;
        int currentWordIndex = 0;

        // Find the indexes of spaces
        for (int i = 0; i < length; i++) {
            if (text.charAt(i) == ' ') {
                spaceIndexes[spaceIndex++] = i;
            }
        }

        // Create an array to store the words
        String[] words = new String[wordCount];
        int start = 0;
```

```
        // Extract words from text based on space indexes
        for (int i = 0; i < wordCount; i++) {
            int end = (i == wordCount - 1) ? length : spaceIndexes[i];
            words[i] = text.substring(start, end).trim();
            start = end + 1;
        }

        return words;
    }

    // Method to compare two arrays of strings
    public static boolean compareArrays(String[] arr1, String[] arr2)
{
        if (arr1.length != arr2.length) {
            return false;
        }

        for (int i = 0; i < arr1.length; i++) {
            if (!arr1[i].equals(arr2[i])) {
                return false;
            }
        }
        return true;
    }

    public static void main(String[] args) {
        // Create a scanner object to take input from the user
        Scanner scanner = new Scanner(System.in);

        // Prompt user for a string input
        System.out.println("Enter a text: ");
        String userInput = scanner.nextLine();
```

```
        // Split the text using the custom method
        String[] wordsWithoutSplit = splitTextWithoutSplit(userInput);

        // Split the text using the built-in split() method
        String[] wordsWithSplit = userInput.split(" ");

        // Compare the two arrays and display the result
        boolean areArraysEqual = compareArrays(wordsWithoutSplit,
wordsWithSplit);

        System.out.println("Words using custom method: ");
        for (String word : wordsWithoutSplit) {
            System.out.println(word);
        }

        System.out.println("\nWords using built-in split() method: ");
        for (String word : wordsWithSplit) {
            System.out.println(word);
        }

        System.out.println("\nAre the results equal? " +
areArraysEqual);

        scanner.close();  // Close the scanner
    }
}
```

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

CODE-

```
import java.util.Scanner;

public class WordAndLengthIn2DArray {


    // Method to find the length of the String without using the
built-in length() method
    public static int getLengthWithoutMethod(String str) {
        int count = 0;
        try {
            while (true) {
                str.charAt(count);  // Try to access each character by
index
                count++;  // Increment count as long as charAt() does
not throw an exception
            }
        } catch (StringIndexOutOfBoundsException e) {
            // Exception will be thrown when count exceeds the string
length, we return the count
            return count;
        }
    }


    // Method to split the text into words without using the split()
method
```

```
public static String[] splitTextWithoutSplit(String text) {
    int length = getLengthWithoutMethod(text);
    int wordCount = 0;

    // Count words by checking spaces
    for (int i = 0; i < length; i++) {
        if (text.charAt(i) == ' ') {
            wordCount++;
        }
    }
    wordCount++;  // The number of words is one more than the
number of spaces

    // Create an array to store the indexes of spaces
    int[] spaceIndexes = new int[wordCount - 1];
    int spaceIndex = 0;
    int currentWordIndex = 0;

    // Find the indexes of spaces
    for (int i = 0; i < length; i++) {
        if (text.charAt(i) == ' ') {
            spaceIndexes[spaceIndex++] = i;
        }
    }

    // Create an array to store the words
    String[] words = new String[wordCount];
    int start = 0;

    // Extract words from text based on space indexes
    for (int i = 0; i < wordCount; i++) {
        int end = (i == wordCount - 1) ? length : spaceIndexes[i];
```

```
            words[i] = text.substring(start, end).trim();
            start = end + 1;
        }

        return words;
    }

    // Method to create a 2D array with word and length
    public static String[][] getWordsAndLengths(String[] words) {
        String[][] wordLengthArray = new String[words.length][2];

        for (int i = 0; i < words.length; i++) {
            wordLengthArray[i][0] = words[i];  // Store the word
            wordLengthArray[i][1] =
String.valueOf(getLengthWithoutMethod(words[i]));  // Store the length
of the word as a string
        }

        return wordLengthArray;
    }

    public static void main(String[] args) {
        // Create a scanner object to take input from the user
        Scanner scanner = new Scanner(System.in);

        // Prompt user for a string input
        System.out.println("Enter a text: ");
        String userInput = scanner.nextLine();

        // Split the text using the custom method
        String[] words = splitTextWithoutSplit(userInput);
```

```
        // Get the words and their corresponding lengths in a 2D array
        String[][] wordsAndLengths = getWordsAndLengths(words);

        // Display the result in a tabular format
        System.out.println("\nWord\tLength");
        for (int i = 0; i < wordsAndLengths.length; i++) {
            // Convert length from String to Integer and display
            System.out.println(wordsAndLengths[i][0] + "\t" +
Integer.valueOf(wordsAndLengths[i][1]));
        }

        scanner.close();  // Close the scanner
    }
}
```

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

   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. 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.

CODE-

```
import java.util.Scanner;

public class ShortestAndLongestString {
```

```
    // Method to find the length of the String without using the
built-in length() method
    public static int getLengthWithoutMethod(String str) {
        int count = 0;
        try {
            while (true) {
                str.charAt(count);  // Try to access each character by
index
                count++;  // Increment count as long as charAt() does
not throw an exception
            }
        } catch (StringIndexOutOfBoundsException e) {
            // Exception will be thrown when count exceeds the string
length, we return the count
            return count;
        }
    }

    // Method to split the text into words without using the split()
method
    public static String[] splitTextWithoutSplit(String text) {
        int length = getLengthWithoutMethod(text);
        int wordCount = 0;

        // Count words by checking spaces
        for (int i = 0; i < length; i++) {
            if (text.charAt(i) == ' ') {
                wordCount++;
            }
        }
        wordCount++;  // The number of words is one more than the
number of spaces
```

```
// Create an array to store the indexes of spaces
int[] spaceIndexes = new int[wordCount - 1];
int spaceIndex = 0;
int currentWordIndex = 0;

// Find the indexes of spaces
for (int i = 0; i < length; i++) {
    if (text.charAt(i) == ' ') {
        spaceIndexes[spaceIndex++] = i;
    }
}

// Create an array to store the words
String[] words = new String[wordCount];
int start = 0;

// Extract words from text based on space indexes
for (int i = 0; i < wordCount; i++) {
    int end = (i == wordCount - 1) ? length : spaceIndexes[i];
    words[i] = text.substring(start, end).trim();
    start = end + 1;
}

    return words;
}

// Method to create a 2D array with word and length
public static String[][] getWordsAndLengths(String[] words) {
    String[][] wordLengthArray = new String[words.length][2];

    for (int i = 0; i < words.length; i++) {
```

```
            wordLengthArray[i][0] = words[i];  // Store the word
            wordLengthArray[i][1] =
String.valueOf(getLengthWithoutMethod(words[i]));  // Store the length
of the word as a string
        }

        return wordLengthArray;
    }

    // Method to find the shortest and longest word using a 2D array
of words and their lengths
    public static int[] findShortestAndLongest(String[][]
wordLengthArray) {
        int shortestLength = Integer.MAX_VALUE;
        int longestLength = Integer.MIN_VALUE;
        String shortestWord = "";
        String longestWord = "";

        for (int i = 0; i < wordLengthArray.length; i++) {
            int wordLength = Integer.valueOf(wordLengthArray[i][1]);

            // Check for the shortest word
            if (wordLength < shortestLength) {
                shortestLength = wordLength;
                shortestWord = wordLengthArray[i][0];
            }

            // Check for the longest word
            if (wordLength > longestLength) {
                longestLength = wordLength;
                longestWord = wordLengthArray[i][0];
            }
        }
```

```
        return new int[] { shortestLength, longestLength };
    }

    public static void main(String[] args) {
        // Create a scanner object to take input from the user
        Scanner scanner = new Scanner(System.in);

        // Prompt user for a string input
        System.out.println("Enter a text: ");
        String userInput = scanner.nextLine();

        // Split the text using the custom method
        String[] words = splitTextWithoutSplit(userInput);

        // Get the words and their corresponding lengths in a 2D array
        String[][] wordsAndLengths = getWordsAndLengths(words);

        // Find the shortest and longest word lengths
        int[] shortestAndLongest =
findShortestAndLongest(wordsAndLengths);

        // Display the results
        System.out.println("\nShortest word length: " +
shortestAndLongest[0]);
        System.out.println("Longest word length: " +
shortestAndLongest[1]);


        scanner.close();  // Close the scanner
    }
}
```

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.

CODE-

```
import java.util.Scanner;


public class VowelAndConsonantCounter {


    // Method to check if the character is a vowel, consonant, or not
a letter
    public static String checkCharacterType(char ch) {
        // Convert to lowercase if character is uppercase
        char lowerChar = Character.toLowerCase(ch);


        // Check if the character is a vowel
        if (lowerChar == 'a' || lowerChar == 'e' || lowerChar == 'i'
|| lowerChar == 'o' || lowerChar == 'u') {
            return "Vowel";
        }


        // Check if the character is a consonant
        if (lowerChar >= 'a' && lowerChar <= 'z') {
            return "Consonant";
        }
```

```
        // If the character is not a letter
        return "Not a Letter";
    }

    // Method to find the count of vowels and consonants in the string
    public static int[] countVowelsAndConsonants(String str) {
        int vowelCount = 0;
        int consonantCount = 0;

        // Iterate through the string and check each character
        for (int i = 0; i < str.length(); i++) {
            char currentChar = str.charAt(i);
            String type = checkCharacterType(currentChar);

            // Increment counters based on character type
            if (type.equals("Vowel")) {
                vowelCount++;
            } else if (type.equals("Consonant")) {
                consonantCount++;
            }
        }

        // Return the counts in an array: index 0 for vowels, index 1
for consonants
        return new int[]{vowelCount, consonantCount};
    }

    public static void main(String[] args) {
        // Create a scanner object to take input from the user
        Scanner scanner = new Scanner(System.in);
```

```
        // Prompt user for a string input
        System.out.println("Enter a string: ");
        String userInput = scanner.nextLine();

        // Call the method to count vowels and consonants
        int[] counts = countVowelsAndConsonants(userInput);

        // Display the result
        System.out.println("Number of Vowels: " + counts[0]);
        System.out.println("Number of Consonants: " + counts[1]);

        // Close the scanner
        scanner.close();
    }
}
```

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.

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


public class CharacterTypeFinder {
```

```
    // Method to check if the character is a vowel, consonant, or not
a letter
    public static String checkCharacterType(char ch) {
        // Convert to lowercase if character is uppercase
        char lowerChar = Character.toLowerCase(ch);

        // Check if the character is a vowel
        if (lowerChar == 'a' || lowerChar == 'e' || lowerChar == 'i'
|| lowerChar == 'o' || lowerChar == 'u') {
            return "Vowel";
        }

        // Check if the character is a consonant
        if (lowerChar >= 'a' && lowerChar <= 'z') {
            return "Consonant";
        }

        // If the character is not a letter
        return "Not a Letter";
    }

    // Method to find vowels and consonants in the string and return
them in a 2D array
    public static String[][] findVowelsAndConsonants(String str) {
        int length = str.length();
        String[][] result = new String[length][2];

        // Iterate through the string and check each character
        for (int i = 0; i < length; i++) {
            char currentChar = str.charAt(i);
            String characterType = checkCharacterType(currentChar);
```

```
            result[i][0] = String.valueOf(currentChar);  // Store the
character
            result[i][1] = characterType;  // Store the type (Vowel,
Consonant, Not a Letter)
        }

        return result;
    }

    // Method to display the 2D array of characters and their types in
a tabular format
    public static void displayTable(String[][] result) {
        System.out.printf("%-15s%-15s\n", "Character", "Type");
        System.out.println("-----------------------------");

        for (int i = 0; i < result.length; i++) {
            System.out.printf("%-15s%-15s\n", result[i][0],
result[i][1]);
        }
    }

    public static void main(String[] args) {
        // Create a scanner object to take input from the user
        Scanner scanner = new Scanner(System.in);

        // Prompt user for a string input
        System.out.println("Enter a string: ");
        String userInput = scanner.nextLine();

        // Call the method to find vowels and consonants
        String[][] result = findVowelsAndConsonants(userInput);
```

```
        // Display the result in a tabular format
        displayTable(result);

        // Close the scanner
        scanner.close();
    }
}
```

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

   Hint =>

   a. 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
   b. Write a method to create a substring from a string using the charAt() method with the string, start, and end index as the parameters
   c. Write a method to compare two strings using the charAt() method and return a boolean result
   d. 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

CODE-

```
import java.util.Scanner;

public class StringTrimAndCompare {

    // Method to trim leading and trailing spaces from a string using
charAt() method
    public static int[] findTrimIndexes(String str) {
        int start = 0;
        int end = str.length() - 1;

        // Find the start index by skipping leading spaces
```

```
while (start <= end && str.charAt(start) == ' ') {
    start++;
}

// Find the end index by skipping trailing spaces
while (end >= start && str.charAt(end) == ' ') {
    end--;
}

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

    // Method to create a substring using charAt() method from start
index to end index
    public static String createSubstring(String str, int start, int
end) {
        StringBuilder substring = new StringBuilder();

        // Construct the substring using charAt() method
        for (int i = start; i <= end; i++) {
            substring.append(str.charAt(i));
        }

        return substring.toString();
    }

    // Method to compare two strings using charAt() method and return
a boolean result
    public static boolean compareStrings(String str1, String str2) {
        if (str1.length() != str2.length()) {
            return false;
        }
```

```
        // Compare characters using charAt() method
        for (int i = 0; i < str1.length(); i++) {
            if (str1.charAt(i) != str2.charAt(i)) {
                return false;
            }
        }

        return true;
    }

    public static void main(String[] args) {
        // Create a scanner object to take input from the user
        Scanner scanner = new Scanner(System.in);

        // Prompt user for a string input
        System.out.println("Enter a string: ");
        String userInput = scanner.nextLine();

        // Call the method to find trim indexes (leading and trailing
spaces)
        int[] trimIndexes = findTrimIndexes(userInput);
        int start = trimIndexes[0];
        int end = trimIndexes[1];

        // Create the substring with leading and trailing spaces
removed using charAt()
        String trimmedStringUsingCharAt = createSubstring(userInput,
start, end);

        // Trim the string using the built-in trim() method
        String trimmedStringUsingBuiltIn = userInput.trim();
```

```
        // Compare the two trimmed strings
        boolean areStringsEqual =
compareStrings(trimmedStringUsingCharAt, trimmedStringUsingBuiltIn);


        // Display the results
        System.out.println("Trimmed String using charAt(): " + "\"" +
trimmedStringUsingCharAt + "\"");
        System.out.println("Trimmed String using built-in trim(): " +
"\"" + trimmedStringUsingBuiltIn + "\"");
        System.out.println("Are both trimmed strings equal? " +
areStringsEqual);


        // Close the scanner
        scanner.close();
    }
}
```

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 =>

   a. 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
   b. 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.
   c. Create a method to display the 2D array in a tabular format.
   d. Finally, the main function takes user inputs, calls the user-defined methods, and displays the result.

CODE-

```
import java.util.Scanner;


public class StudentVoting {
```

```
    // Method to define the random 2-digit age of students and return
a 1D array of ages
    public static int[] getStudentAges(int numberOfStudents) {
        int[] ages = new int[numberOfStudents];
        Scanner scanner = new Scanner(System.in);

        // Taking age input from the user for each student
        for (int i = 0; i < numberOfStudents; i++) {
            System.out.print("Enter age of student " + (i + 1) + ":
");
            ages[i] = scanner.nextInt();
        }
        return ages;
    }

    // Method to check if a student can vote based on age and return a
2D array of age and voting status
    public static String[][] checkVotingEligibility(int[] ages) {
        String[][] result = new String[ages.length][2];

        for (int i = 0; i < ages.length; i++) {
            int age = ages[i];
            String votingStatus = "Cannot Vote";  // Default status

            // Validate if age is negative
            if (age < 0) {
                votingStatus = "Invalid Age";
            } else if (age >= 18) {
                votingStatus = "Can Vote";
            }

            result[i][0] = String.valueOf(age);  // Store the age as
string
```

```
            result[i][1] = votingStatus;          // Store voting
eligibility
        }
        return result;
    }

    // Method to display the 2D array in a tabular format
    public static void displayVotingStatus(String[][] result) {
        System.out.printf("%-10s%-15s\n", "Age", "Voting Status");
        System.out.println("-----------------------------");

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

    public static void main(String[] args) {
        // Define the number of students
        int numberOfStudents = 10;

        // Get the ages of the students
        int[] studentAges = getStudentAges(numberOfStudents);

        // Check the voting eligibility for each student
        String[][] votingStatus = checkVotingEligibility(studentAges);

        // Display the result in a tabular format
        displayVotingStatus(votingStatus);
    }
}
```

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 =>

   a. **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)
   b. Create a Method to find the Computer Choice using the Math.random
   c. Create a Method to find the winner between the user and the computer
   d. Create a Method to find the average and percentage of wins for the user and the computer and return a String 2D array
   e. Create a Method to display the results of every game and also display the average and percentage wins
   f. In the main take user input for the number of games and call methods to display results

CODE-

```
import java.util.Scanner;


public class RockPaperScissorsGame {


    // Method to generate the computer's choice
    public static String getComputerChoice() {
        int randomChoice = (int) (Math.random() * 3); // Generates a
random number between 0 and 2

        switch (randomChoice) {
            case 0: return "Rock";
            case 1: return "Paper";
            case 2: return "Scissors";
            default: return "";
        }
    }

    // Method to find the winner between the user and the computer
```

```
    public static String getWinner(String userChoice, String
computerChoice) {
        if (userChoice.equals(computerChoice)) {
            return "Tie";
        }

        switch (userChoice) {
            case "Rock":
                return (computerChoice.equals("Scissors")) ? "Player"
: "Computer";
            case "Paper":
                return (computerChoice.equals("Rock")) ? "Player" :
"Computer";
            case "Scissors":
                return (computerChoice.equals("Paper")) ? "Player" :
"Computer";
            default:
                return "Invalid";
        }
    }

    // Method to calculate the average and percentage of wins for the
user and the computer
    public static String[][] calculateStats(int playerWins, int
computerWins, int totalGames) {
        String[][] stats = new String[3][2];
        stats[0][0] = "Player Wins";
        stats[1][0] = "Computer Wins";
        stats[2][0] = "Total Games";

        stats[0][1] = String.valueOf(playerWins);
        stats[1][1] = String.valueOf(computerWins);
        stats[2][1] = String.valueOf(totalGames);
```

```
        double playerWinPercentage = (totalGames == 0) ? 0 : ((double)
playerWins / totalGames) * 100;
        double computerWinPercentage = (totalGames == 0) ? 0 :
((double) computerWins / totalGames) * 100;


        stats[0][1] += " (" + String.format("%.2f",
playerWinPercentage) + "%)";
        stats[1][1] += " (" + String.format("%.2f",
computerWinPercentage) + "%)";

        return stats;
    }


    // Method to display the results of each game and the overall
stats
    public static void displayResults(int playerWins, int
computerWins, int totalGames) {
        // Display individual game results
        System.out.println("\nGame Results:");
        System.out.printf("%-20s%-20s%-20s\n", "Game Number", "Player
Choice", "Computer Choice");

        for (int i = 1; i <= totalGames; i++) {
            // Player's choice and Computer's choice can be added if
you track each game
            String playerChoice = "Player's Choice"; // Replace this
with actual input
            String computerChoice = getComputerChoice();
            String winner = getWinner(playerChoice, computerChoice);
            System.out.printf("%-20d%-20s%-20s%-20s\n", i,
playerChoice, computerChoice, winner);
        }
```

```
        // Display overall stats
        String[][] stats = calculateStats(playerWins, computerWins,
totalGames);
        System.out.println("\nOverall Stats:");
        System.out.printf("%-20s%-20s\n", "Category", "Count /
Percentage");
        for (int i = 0; i < stats.length; i++) {
            System.out.printf("%-20s%-20s\n", stats[i][0],
stats[i][1]);
        }
    }

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

    // Get the number of games to be played
    System.out.print("Enter the number of games to be played: ");
    int numberOfGames = scanner.nextInt();

    // Variables to track the results
    int playerWins = 0;
    int computerWins = 0;
    int ties = 0;

        // Loop through the games
        for (int i = 1; i <= numberOfGames; i++) {
            System.out.print("Enter your choice (Rock, Paper, or
Scissors): ");
            String userChoice = scanner.next();


            String computerChoice = getComputerChoice();
            String winner = getWinner(userChoice, computerChoice);
```

```
if (winner.equals("Player")) {
    playerWins++;
} else if (winner.equals("Computer")) {
    computerWins++;
} else {
    ties++;
}

// Display the result of the current game
System.out.println("Your choice: " + userChoice);
            System.out.println("Computer's choice: " +
computerChoice);
            System.out.println("Winner: " + winner);
            System.out.println("------------------------------");
        }

        // Display the overall results and stats
        displayResults(playerWins, computerWins, numberOfGames);

        // Close the scanner
        scanner.close();
    }
}
```

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 =>

a. 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
b. 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
c. 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
d. Finally write a Method to display the scorecard of all students with their scores, total, average, percentage, and grade in a tabular format.

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

public class StudentScorecard {

    // Method to generate random 2-digit scores for Physics,
Chemistry, and Math
    public static int[][] generateRandomScores(int numOfStudents) {
        Random rand = new Random();
        int[][] scores = new int[numOfStudents][3]; // 2D array to
hold scores for each student

        // Generate random scores for each student
        for (int i = 0; i < numOfStudents; i++) {
            scores[i][0] = rand.nextInt(100) + 1; // Physics score (1
to 100)
            scores[i][1] = rand.nextInt(100) + 1; // Chemistry score
(1 to 100)
```

```
            scores[i][2] = rand.nextInt(100) + 1; // Math score (1 to
100)
        }

        return scores;
    }

    // Method to calculate total, average, and percentage for each
student
    public static double[][] calculateTotalAveragePercentage(int[][]
scores) {
        int numOfStudents = scores.length;
        double[][] result = new double[numOfStudents][4]; // 2D array
to hold total, average, percentage for each student

        for (int i = 0; i < numOfStudents; i++) {
            double total = scores[i][0] + scores[i][1] + scores[i][2];
// Total marks
            double average = total / 3; // Average marks
            double percentage = (total / 300) * 100; // Percentage
(out of 300)

            result[i][0] = total;        // Total marks
            result[i][1] = average;      // Average marks
            result[i][2] = percentage;   // Percentage
            result[i][3] = Math.round(percentage * 100.0) / 100.0; //
Rounded percentage to 2 decimal places
        }

        return result;
    }

    // Method to calculate the grade based on percentage
    public static String[][] calculateGrades(double[][] percentages) {
        int numOfStudents = percentages.length;
        String[][] grades = new String[numOfStudents][1];

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

```
double percentage = percentages[i][3];

// Determine grade based on percentage
if (percentage >= 90) {
    grades[i][0] = "A+";
} else if (percentage >= 80) {
    grades[i][0] = "A";
} else if (percentage >= 70) {
    grades[i][0] = "B+";
} else if (percentage >= 60) {
    grades[i][0] = "B";
            } else if (percentage >= 50) {
                grades[i][0] = "C+";
            } else if (percentage >= 40) {
                grades[i][0] = "C";
            } else {
                grades[i][0] = "F";
            }
        }

        return grades;
    }

    // Method to display the scorecard of all students
    public static void displayScorecard(int[][] scores, double[][]
result, String[][] grades) {
        System.out.printf("%-10s%-10s%-10s%-10s%-10s%-10s%-10s\n",
"Student", "Physics", "Chemistry", "Math", "Total", "Percentage",
"Grade");

        // Display scores, total, percentage, and grade for each
student
        for (int i = 0; i < scores.length; i++) {

System.out.printf("%-10d%-10d%-10d%-10d%-10.2f%-10.2f%-10s\n",
                                i + 1,
                                scores[i][0],
                                scores[i][1],
```

```
                                scores[i][2],
                                result[i][0],
                                result[i][3],
                                grades[i][0]);
        }
    }

    public static void main(String[] args) {
        int numOfStudents = 5; // You can change the number of
students here

        // Generate random scores for students
        int[][] scores = generateRandomScores(numOfStudents);

        // Calculate total, average, and percentage for each student
        double[][] result = calculateTotalAveragePercentage(scores);

        // Calculate grades based on the percentage
        String[][] grades = calculateGrades(result);

        // Display the scorecard
        displayScorecard(scores, result, grades);
    }
}
```