CodInClub

Powered By BridgeLabz

Best Programming Practice

1. Use Variables including for Fixed, User Inputs, and Results
2. Use Methods instead of writing code in the main() function
3. Proper naming conventions for all variables and methods
4. Proper Program Name and Class Name
5. Handle Checked and Unchecked Exceptions wherever possible
6. Proper Method Name which indicates action taking inputs and providing result

Sample Program 1: Create a program to find all the occurrences of a character in a string using charAt() method

a. Take user input for the String and occurrences of the Character to find
b. Write a method to find all the occurrences of the characters.
    i. The logic used is to first find the number of occurrences of the character and
    ii. then create an array to store the indexes of the character
c. Call the method in the main and display the result

Java

```java
// Program to find all the occurrences of a character in a string
import java.util.Scanner;
class StringAnalyzer {
  // Method to find all the index of a character in a string using charAt()
  // method and return them in an array
  public static int[] findAllIndexes(String text, char ch) {
    // The count is used to find the number of occurrences of the character
    int count = 0;
    for (int i = 0; i < text.length(); i++) {
        if (text.charAt(i) == ch) {
            count++;
        }
    }

    // Create an array to store the indexes of the character
    int[] indexes = new int[count];
    int j = 0;
    for (int i = 0; i < text.length(); i++) {
        if (text.charAt(i) == ch) {
            indexes[j] = i;
            j++;
        }
    }
    return indexes;
```

```
    }
    public static void main(String[] args) {
      // Take user input for Text and Character to check Occurrences
      Scanner sc = new Scanner(System.in);
      System.out.print(Enter a text: ");
      String text = sc.nextLine();
      System.out.print("Enter a character to find the occurrences: ");
      char ch = sc.next().charAt(0);

      // Find the occurrences of the character
      int[] indexes = findAllIndexes(text, ch);

      // Display the occurrences of the character
      System.out.println("Indexes of the character '" + ch + "': ");
      for (int i = 0; i < indexes.length; i++) {
          System.out.print(indexes[i] + " ");
      }
    }
}
```

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

Sol:

```
import java.util.Scanner;

public class Main {

    public static int getStringLength(String s) {

        int count = 0;

        try {
```

```
            while (true) {
                s.charAt(count);
                count++;
            }
        } catch (Exception e) {
        }
        return count;
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String input = sc.next();
        int manualLength = getStringLength(input);
        int builtinLength = input.length();
        System.out.println("Length without using length(): " + manualLength);
        System.out.println("Length using built-in length(): " + 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

Sol:

```
import java.util.Scanner;
public class Main {
```

```
public static int getLength(String s) {
    int count = 0;
    try {
        while (true) {
            s.charAt(count);
            count++;
        }
    } catch (Exception e) {
    }
    return count;
}
public static String[] splitManual(String s) {
    int len = getLength(s);
    int spaces = 0;
    for (int i = 0; i < len; i++) {
        if (s.charAt(i) == ' ') spaces++;
    }
    int[] spaceIndexes = new int[spaces + 2];
    spaceIndexes[0] = -1;
    int index = 1;
    for (int i = 0; i < len; i++) {
        if (s.charAt(i) == ' ') spaceIndexes[index++] = i;
    }
    spaceIndexes[index] = len;
    String[] words = new String[spaces + 1];
    for (int i = 0; i < words.length; i++) {
        StringBuilder sb = new StringBuilder();
        for (int j = spaceIndexes[i] + 1; j < spaceIndexes[i + 1]; j++) {
            sb.append(s.charAt(j));
        }
```

```
            words[i] = sb.toString();
        }
        return words;
    }
    public static boolean compareArrays(String[] a, String[] b) {
        if (a.length != b.length) return false;
        for (int i = 0; i < a.length; i++) {
            if (!a[i].equals(b[i])) return false;
        }
        return true;
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String text = sc.nextLine();
        String[] manual = splitManual(text);
        String[] builtin = text.split(" ");
        boolean same = compareArrays(manual, builtin);
        System.out.println("Manual Split:");
        for (String word : manual) System.out.println(word);
        System.out.println("Built-in Split:");
        for (String word : builtin) System.out.println(word);
        System.out.println("Are both same? " + same);
    }
}
```

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

Sol:

```
import java.util.Scanner;
public class Main {
    public static int getLength(String s) {
        int count = 0;
        try {
            while (true) {
                s.charAt(count);
                count++;
            }
        } catch (Exception e) {
        }
        return count;
    }
    public static String[] splitManual(String s) {
        int len = getLength(s);
        int spaces = 0;
        for (int i = 0; i < len; i++) {
            if (s.charAt(i) == ' ') spaces++;
        }
        int[] spaceIndexes = new int[spaces + 2];
        spaceIndexes[0] = -1;
        int index = 1;
        for (int i = 0; i < len; i++) {
            if (s.charAt(i) == ' ') spaceIndexes[index++] = i;
        }
```

```
        spaceIndexes[index] = len;
        String[] words = new String[spaces + 1];
        for (int i = 0; i < words.length; i++) {
            StringBuilder sb = new StringBuilder();
            for (int j = spaceIndexes[i] + 1; j < spaceIndexes[i + 1]; j++) {
                sb.append(s.charAt(j));
            }
            words[i] = sb.toString();
        }
        return words;
    }
    public static String[][] wordWithLengths(String[] words) {
        String[][] result = new String[words.length][2];
        for (int i = 0; i < words.length; i++) {
            result[i][0] = words[i];
            result[i][1] = String.valueOf(getLength(words[i]));
        }
        return result;
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String text = sc.nextLine();
        String[] words = splitManual(text);
        String[][] data = wordWithLengths(words);
        System.out.println("Word\tLength");
        for (int i = 0; i < data.length; i++) {
            System.out.println(data[i][0] + "\t" + Integer.parseInt(data[i][1]));
        }
    }
}
```

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.

Sol:

```
import java.util.Scanner;

public class Main {
    public static int getLength(String s) {
        int count = 0;
        try {
            while (true) {
                s.charAt(count);
                count++;
            }
        } catch (Exception e) {
        }
        return count;
    }
    public static String[] splitManual(String s) {
        int len = getLength(s);
        int spaces = 0;
        for (int i = 0; i < len; i++) {
            if (s.charAt(i) == ' ') spaces++;
        }
```

```
    int[] spaceIndexes = new int[spaces + 2];
    spaceIndexes[0] = -1;
    int index = 1;
    for (int i = 0; i < len; i++) {
        if (s.charAt(i) == ' ') spaceIndexes[index++] = i;
    }
    spaceIndexes[index] = len;
    String[] words = new String[spaces + 1];
    for (int i = 0; i < words.length; i++) {
        StringBuilder sb = new StringBuilder();
        for (int j = spaceIndexes[i] + 1; j < spaceIndexes[i + 1]; j++) {
            sb.append(s.charAt(j));
        }
        words[i] = sb.toString();
    }
    return words;
}
public static String[][] getWordLengthPairs(String[] words) {
    String[][] result = new String[words.length][2];
    for (int i = 0; i < words.length; i++) {
        result[i][0] = words[i];
        result[i][1] = String.valueOf(getLength(words[i]));
    }
    return result;
}
public static int[] findMinMaxIndexes(String[][] data) {
    int minIndex = 0;
    int maxIndex = 0;
    for (int i = 1; i < data.length; i++) {
        int length = Integer.parseInt(data[i][1]);
```

```
            int minLen = Integer.parseInt(data[minIndex][1]);
            int maxLen = Integer.parseInt(data[maxIndex][1]);
            if (length < minLen) minIndex = i;
            if (length > maxLen) maxIndex = i;
        }
        return new int[]{minIndex, maxIndex};
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String text = sc.nextLine();
        String[] words = splitManual(text);
        String[][] wordLengthPairs = getWordLengthPairs(words);
        int[] indexes = findMinMaxIndexes(wordLengthPairs);
        System.out.println("Shortest word: " + wordLengthPairs[indexes[0]][0]);
        System.out.println("Longest word: " + wordLengthPairs[indexes[1]][0]);
    }
}
```

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.

Sol:

```
import java.util.Scanner;
public class Main {
    public static String checkCharType(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";
        }
    }
    return "Not a Letter";
}
public static int[] countVowelsAndConsonants(String s) {
    int vowels = 0;
    int consonants = 0;
    int i = 0;
    try {
        while (true) {
            char ch = s.charAt(i);
            String type = checkCharType(ch);
            if (type.equals("Vowel")) vowels++;
            else if (type.equals("Consonant")) consonants++;
            i++;
        }
    } catch (Exception e) {
    }
    return new int[]{vowels, consonants};
}
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
```

```
        String input = sc.nextLine();
        int[] result = countVowelsAndConsonants(input);
        System.out.println("Vowels: " + result[0]);
        System.out.println("Consonants: " + result[1]);
    }
}
```

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.

Sol:

```
import java.util.Scanner;
public class Main {
    public static String checkCharType(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";
            }
        }
```

```
    return "Not a Letter";
}
public static String[][] getCharDetails(String s) {
    int i = 0;
    int count = 0;
    try {
        while (true) {
            s.charAt(count);
            count++;
        }
    } catch (Exception e) {
    }
    String[][] result = new String[count][2];
    try {
        while (i < count) {
            char ch = s.charAt(i);
            result[i][0] = String.valueOf(ch);
            result[i][1] = checkCharType(ch);
            i++;
        }
    } catch (Exception e) {
    }
    return result;
}
public static void displayCharTable(String[][] arr) {
    System.out.println("Character\tType");
    for (int i = 0; i < arr.length; i++) {
        System.out.println(arr[i][0] + "\t\t" + arr[i][1]);
    }
}
```

```
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String input = sc.nextLine();
        String[][] result = getCharDetails(input);
        displayCharTable(result);
    }
}
```

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

Sol:

```
import java.util.Scanner;
public class Main {
    public static int[] trimIndices(String s) {
        int start = 0;
        int end = 0;
        try {
            while (s.charAt(end) != '\0') {
                end++;
            }
        } catch (Exception e) {
        }
        end--;
```

```
    while (start <= end && s.charAt(start) == ' ') {
        start++;
    }
    while (end >= start && s.charAt(end) == ' ') {
        end--;
    }
    return new int[]{start, end};
}
public static String customSubstring(String s, int start, int end) {
    String result = "";
    for (int i = start; i <= end; i++) {
        result += s.charAt(i);
    }
    return result;
}
public static boolean compareStrings(String s1, String s2) {
    try {
        int i = 0;
        while (true) {
            if (s1.charAt(i) != s2.charAt(i)) {
                return false;
            }
            i++;
        }
    } catch (Exception e) {
        try {
            s2.charAt(i);
            return false;
        } catch (Exception ex) {
            return true;
```

```
        }
      }
    }
    public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      String input = sc.nextLine();
      int[] indices = trimIndices(input);
      String trimmedCustom = customSubstring(input, indices[0], indices[1]);
      String trimmedBuiltIn = input.trim();
      boolean isSame = compareStrings(trimmedCustom, trimmedBuiltIn);
      System.out.println("Custom Trimmed: '" + trimmedCustom + "'");
      System.out.println("Built-in Trimmed: '" + trimmedBuiltIn + "'");
      System.out.println("Both are same: " + isSame);
    }
}
```

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.

Sol:

```
import java.util.Scanner;
import java.util.Random;
public class Main {
    public static int[] generateRandomAges(int n) {
      int[] ages = new int[n];
```

```
    Random rand = new Random();
    for (int i = 0; i < n; i++) {
        ages[i] = rand.nextInt(90) + 10;
    }
    return ages;
}
public static String[][] checkVotingEligibility(int[] ages) {
    String[][] result = new String[ages.length][2];
    for (int i = 0; i < ages.length; i++) {
        result[i][0] = String.valueOf(ages[i]);
        if (ages[i] < 0) {
            result[i][1] = "false";
        } else if (ages[i] >= 18) {
            result[i][1] = "true";
        } else {
            result[i][1] = "false";
        }
    }
    return result;
}
public static void displayTable(String[][] data) {
    System.out.println("Age\tCan Vote");
    for (int i = 0; i < data.length; i++) {
        System.out.println(data[i][0] + "\t" + data[i][1]);
    }
}
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int[] ages = generateRandomAges(10);
    String[][] result = checkVotingEligibility(ages);
```

```
        displayTable(result);
    }
}
```

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

Sol:

```
import java.util.Scanner;

public class Main {
    public static String getComputerChoice() {
        int choice = (int) (Math.random() * 3);
        if (choice == 0) return "rock";
        if (choice == 1) return "paper";
        return "scissors";
    }
    public static String findWinner(String user, String computer) {
        if (user.equals(computer)) return "Draw";
        if (user.equals("rock") && computer.equals("scissors")) return "User";
        if (user.equals("scissors") && computer.equals("paper")) return "User";
        if (user.equals("paper") && computer.equals("rock")) return "User";
        return "Computer";
    }
```

```
public static String[][] calculateStats(int userWins, int computerWins, int totalGames) {
    String[][] stats = new String[2][3];
    stats[0][0] = "User";
    stats[1][0] = "Computer";
    stats[0][1] = String.valueOf(userWins);
    stats[1][1] = String.valueOf(computerWins);
    double userPercent = ((double) userWins / totalGames) * 100;
    double computerPercent = ((double) computerWins / totalGames) * 100;
    stats[0][2] = String.format("%.2f", userPercent) + "%";
    stats[1][2] = String.format("%.2f", computerPercent) + "%";
    return stats;
}
public static void displayResults(String[][] results, String[][] stats) {
    System.out.println("Game\tUser\tComputer\tWinner");
    for (int i = 0; i < results.length; i++) {
        System.out.println((i + 1) + "\t" + results[i][0] + "\t" + results[i][1] + "\t\t" + results[i][2]);
    }
    System.out.println("\nPlayer\tWins\tWin %");
    for (int i = 0; i < stats.length; i++) {
        System.out.println(stats[i][0] + "\t" + stats[i][1] + "\t" + stats[i][2]);
    }
}
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    System.out.print("Enter number of games: ");
    int n = sc.nextInt();
    sc.nextLine();
    String[][] results = new String[n][3];
    int userWins = 0, computerWins = 0;
    for (int i = 0; i < n; i++) {
```

```
        System.out.print("Enter your choice (rock/paper/scissors): ");
        String user = sc.nextLine().toLowerCase();
        String computer = getComputerChoice();
        String winner = findWinner(user, computer);
        if (winner.equals("User")) userWins++;
        else if (winner.equals("Computer")) computerWins++;
        results[i][0] = user;
        results[i][1] = computer;
        results[i][2] = winner;
      }
      String[][] stats = calculateStats(userWins, computerWins, n);
      displayResults(results, stats);
    }
}
```

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.

Sol:

```
import java.util.Scanner;

public class Main {
    public static int[][] generateScores(int students) {
        int[][] scores = new int[students][3];
        for (int i = 0; i < students; i++) {
            for (int j = 0; j < 3; j++) {
                scores[i][j] = (int) (Math.random() * 51) + 50; // Generating random marks between 50 to 100
            }
        }
        return scores;
    }
    public static double[][] calculateResults(int[][] scores) {
        int students = scores.length;
        double[][] results = new double[students][3];
        for (int i = 0; i < students; i++) {
            int total = scores[i][0] + scores[i][1] + scores[i][2];
            double avg = total / 3.0;
            double perc = (total / 300.0) * 100;
            results[i][0] = total;
            results[i][1] = Math.round(avg * 100.0) / 100.0;
            results[i][2] = Math.round(perc * 100.0) / 100.0;
        }
        return results;
    }
    public static String[] calculateGrades(double[][] results) {
        String[] grades = new String[results.length];
```

```
        for (int i = 0; i < results.length; i++) {
            double perc = results[i][2];
            if (perc >= 80) grades[i] = "A (Level 4, above agency-normalized standards)";
            else if (perc >= 70) grades[i] = "B (Level 3, at agency-normalized standards)";
            else if (perc >= 60) grades[i] = "C (Level 2, below, but approaching agency-normalized
standards)";
            else if (perc >= 50) grades[i] = "D (Level 1, well below agency-normalized standards)";
            else if (perc >= 40) grades[i] = "E (Level 1-, too below agency-normalized standards)";
            else grades[i] = "R (Remedial standards)";
        }
        return grades;
    }
    public static void displayScoreCard(int[][] scores, double[][] results, String[] grades) {
        System.out.println("ID\tPhysics\tChemistry\tMaths\tTotal\tAverage\tPercentage\tGrade");
        for (int i = 0; i < scores.length; i++) {
            System.out.println((i + 1) + "\t" + scores[i][0] + "\t" + scores[i][1] + "\t\t" + scores[i][2] + "\t"
+
                    (int) results[i][0] + "\t" + results[i][1] + "\t" + results[i][2] + "%\t\t" + grades[i]);
        }
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter number of students: ");
        int n = sc.nextInt();
        int[][] scores = generateScores(n);
        double[][] results = calculateResults(scores);
        String[] grades = calculateGrades(results);
        displayScoreCard(scores, results, grades);
    }
}
```