Assignment #4 (CS210 – S24 – 10146)

# Coding #1: **Student Score Analyzer (15 Points)**

## **Overview**

Develop a Java program that reads student score data from a text file, processes this data using arrays, and outputs a report to another file. This assignment will help you practice working with arrays, file I/O, and basic data analysis techniques in Java.

## **Objective**:

* File Reading: Read data from a file containing student scores.
* Data Storage: Use arrays to store and manipulate the scores.
* Data Analysis: Calculate various statistics from the scores.
* File Writing: Write an analysis report to an output file.

### **Part A: Score Analysis Class**

1. Method Specification:

* calculateAverage method:
  + parameter: int[] scores
  + returns: double
* findMaxScore method:
  + parameter: int[] scores
  + returns: int
* findMinScore method:
  + parameter int[] scores
  + returns: int

1. Implementation Details:

* Sum elements and calculate average:
  + Sum all elements in the array and divide by the count to get the average.
  + Return 0.0 if the array is empty or has zero length.
* Find maximum and minimum scores:
  + Traverse the array to find the maximum and minimum values.
  + Handle edge cases where array might be empty.
* Implement the methods without using external libraries or predefined array utility functions.

1. Unit Tests:

* Test each method.
* Test with typical score arrays.
* Handle boundary conditions like empty array or array with a single number.

### **Part B: Application Entry Point**

1. A separate class with main method:

* Create a separate class ScoreAnalysisApp that includes the main method.

1. Program Behavior:

* args parameter for main method:
  + args[0]: string for the input file name
  + args[1]: string for the output file name
* Read and process input:
  + Read all line in the input file where each line starts with a student’s name followed by student scores that are separated by ‘,’.
  + Parse the student’s name and scores from each line.
* Call analysis methods:
  + Use the above methods to analyze the scores.
* Output analytical results:
  + Write the results to the output file.
  + For each student, print the name, average score formatted with one decimal point, maximum and minimum score.
  + For example: John Doe: Avg 85.0, Max 95, Min 75

### **Part C: Sample Input and Output**

Input:

John Doe, 80, 70, 90, 66

Jane Smith, 100, 85, 95

Output:

John Doe: Avg 76.5, Max 90, Min 66

Jane Smith: Avg 93.3, Max 100, Min 85

Explanation:

* John Doe has scores 80, 70, 90 and 66. The average is , maximum 90 and minimum 66.
* Jane Smith has scores 100, 85 and 95. The average is , maximum 100 and minimum 85.

# Coding #2: **Text Formatter** (15 points)

## **Objective**:

## Develop a Java program that reads text from a file, formats it according to specific rules, and writes the formatted text to an output file. This exercise will enhance your skills in string manipulation, file I/O, and applying formatting rules.

### **Part A: Text formatting class**

1. Method Specification:

* formatText method:
  + parameter: String text
  + returns: String

1. Implementation Details:

* Ensure each line contains at most 80 characters.
* Consolidate multiple consecutive white spaces into a single white space.
* Remove spaces before punctuation marks.
* Start each paragraph with two white spaces.

1. Unit Tests:

* Test consecutive spaces: verify that multiple spaces are reduced to a single space.
* Test line length: ensure no lines exceed 80 characters.
* Test paragraph indentation: check each paragraph starts with two spaces.
* Test punctuation spacing: ensure no spaces exit before punctuation.

### **Part B: Application Entry Point**

1. A separate class with main method:

* Create a separate class TextFormatterApp that includes the main method.

1. Program Behavior:

* args parameter for main method:
  + args[0]: string for the input file name
  + args[1]: string for the output file name
* Read the input text from a file, treating each line of text as a paragraph.
* Format the paragraph using formatText method.
* Write the formatted text to the output file.

### **Part C: Sample Input and Output**

Input: (single line as a paragraph)

This is a sample text that will be formatted. It contains multiple spaces and punctuation. This is a single paragraph ! !

Output: (multiple lines after formatting)

This is a sample text that will be formatted. It contains multiple spaces and

punctuation. This is a single paragraph!!

Explanation:

* The input contains text with irregular spaces and improperly formatted paragraphs. The output demonstrates correct spacing, punctuation handling, and paragraph indentation under the 80-character limit.

### **Submission Details**

* **Code comments**: Ensure the code is well-commented.
* **Compilation and execution**: Verify that the code compiles and runs as expected.
* **Packaging:** package all project files in a zip file with your name as part of the file name.