PROG 1400 - ASSIGNMENT 1

JAVA BASICS AND OOP CONCEPTS

Assignment Value: *15%* of overall course mark.

Due Date: **Oct. 13,2023.**

Late submissions will receive the standard late submission penalty as stated in the course outline.

Assignment Instructions:

Use IntelliJ to create a Java console application.

Submissions:

Submit your complete folder project or for all classes from your project in zipped folder on Bright space (Assignment 2 Dropbox). Make the project's name (folder name) is "Assignment2_Yourname".

Evaluation:

To ensure the greatest chance of success on this assignment, be sure to check the marking rubric contained at the end of this document or in Brightspace. The rubric contains the criteria your instructor will be assessing when marking your assignment.

Program – IT Courses

You have been asked to write a Java console application to track and gather statistics for IT courses. The application requirements are as follows:

- The IT courses are 2 courses with 6 students per course.
- The user will be presented with a set of data entry options for 2 Courses of 6 Students (see console output below)
    - For courses, gather the courses name
    - For students, gather the student's name, assignment1 mark, and asssignment2 mark.
    - Validate the student mark that should be greater than or equal 0.0 and less than or equal 100.0
- After gathering all the data, you will generate 2 reports (See example below):
    - A Course stats report that displays the following:
        - Average of Assignment1, and average of Assignment2
        - The average for both assignments.
        - Course rating
    - The course rating is calculated using the following formula:
        - A – If the average of assignment1 and assignment 2 is greater than or equal 90.
        - B – If the average of assignment1 and assignment 2 is greater than or equal to 80 AND less than 90
        - C – If the average of assignment1 and assignment 2 is greater than or equal 70 AND less than 80
        - D – If the average of assignment1 and assignment 2 is greater than or equal 60 AND less than 70
        - F – If the average of assignment1 and assignment 2 is less than 60
    - The Stats per student report will report the course name, their name, the student marks in Asssignment1 and Assignment2 (see below).
- Program Organization: Make 3 Java classes: A Course class, A student class and a Main class that launches the application, gathers input and displays output. Course and Student should have appropriate properties for their types. These classes should also have appropriate methods like outputStudentDetails() etc.

Examples and Testing

In the section below, you will be presented with console output of a successful execution of a sample solution to the program, which should help demonstrate how your input/output on the program should work. In addition to the sample values used in the screenshot(s), there should be validation on all marks' inputs. You can expect your instructor to grade your assignment by trying different values than below. **In other words, you should thoroughly test your code before submitting!**

Sample Output - Make sure your program can output data *exactly* as shown below. Bold Text is for user entry.

```
Programming IT Courses

COURSE ENTRY

=================================
Enter name for course # 1:

Java Programming

Enter name for course # 2:

Python Programming

Student Entry

================================

Enter students for Java Programming:

Enter name for student # 1:

Randy McDonald

Enter Assignment1 mark for Randy David:

70

Enter Assignment2 mark for Randy David:

80

Enter name for student # 2:

…

…


REPORT: Stats per Course

================================

Java Programming : Assignment1 – 70   Assignment2 – 80      Average - 75
Course Rating: C

Python Programming : Assignment1 - 90 Assignment2 – 96      Average - 93
Course Rating: A


REPORT: Stats per student

================================

Java Programming

Ron David:      Assignment1 – 70      Assignment2 - 78      Total - 148
```

```
Jones Eduard:   Assignment1 – 60      Assignment2 - 80      Total – 140
Jacob Sears:    Assignment1 – 85      Assignment2 - 85      Total – 170
Mike Jackson:   Assignment1 – 60      Assignment2 - 60      Total – 120
Sally Roberts:  Assignment1 – 65      Assignment2 - 65      Total – 130
Vahid Tom:      Assignment1 – 90      Assignment2 – 90      Total – 180
Python Programming
Sally John:     Assignment1 – 88      Assignment2 – 62      Total – 150
Smith Joe:      Assignment1 – 77      Assignment2 – 63      Total – 140
Joe Kirby:      Assignment1 – 98      Assignment2 – 82      Total – 180
Riley Scott:    Assignment1 – 80      Assignment2 – 80      Total – 160
Robert Don:     Assignment1 – 65      Assignment2 – 66      Total – 131
Sheila McDonald:Assignment1 – 75      Assignment2 – 80      Total – 155
```

IT Programming

Criteria	Insufficient (0 pts)	Needs Development (1-2 pts)	Sufficient (3-4 pts)	Excellent (5 pts)	Marks	X
Input / Output	Little to no effort was made or contains too many errors / omissions.	A reasonable effort was made, but there are multiple areas for improvement.	A good effort was made, but at least one error or omission exists.	• All course and student inputs can be successfully entered, and use descriptive prompts • The report output lines are well-formatted and contain all expected information • All output values are formatted, where appropriate, using proper numerical formatting (e.g. preceded by two decimal places)		
OOP	Little to no effort was made or contains too many errors / omissions.	A reasonable effort was made, but there are multiple areas for improvement.	A good effort was made, but at least one error or omission exists.	Solution displays strong understanding of OOP fundamentals. Classes are created, with all required properties and methods as expected. Constructors are included for each class (except Main) and used appropriately. Object instantiation done correctly and in the appropriate class (Main).		2
Data Validation	Little to no effort was made or contains too many errors / omissions.	A reasonable effort was made, but there are multiple areas for improvement.	A good effort was made, but at least one error or omission exists.	Data input is validated properly: **marks must be greater than or equal to zero and less than or equal to 100.00**. **Strings must be at least 3 characters.**		
Course Rating	Little to no effort was made or contains too many errors / omissions.	A reasonable effort was made, but there are multiple areas for improvement.	A good effort was made, but at least one error or omission exists.	Each Course is assigned the correct rating, according to program requirements. Output is given with the right class rating.		
Course Report	Little to no effort was made or contains too many errors / omissions.	A reasonable effort was made, but there are multiple areas for improvement.	A good effort was made, but at least one error or omission exists.	The course report is present and displays expected report data. Output is well-formatted, clearly labeled and has an easily readable layout.		
Student Report	Little to no effort was made or contains too many errors / omissions.	A reasonable effort was made, but there are multiple areas for improvement.	A good effort was made, but at least one error or omission exists.	The student report is present and displays expected report data. Output is well-formatted, clearly labeled and has an easily readable layout.		
Array & Object Usage	Little to no effort was made or contains too many errors / omissions.	A reasonable effort was made, but there are multiple areas for improvement.	A good effort was made, but at least one error or omission exists.	Arrays are used to store objects as expected. Proper interaction with object arrays is demonstrated.		
Comments & Best Coding Practices (At least 60% of the functional requirements must be complete)	Little to no effort was made or contains too many errors / omissions.	A reasonable effort was made, but there are multiple areas for improvement.	A good effort was made, but at least one error or omission exists.	Organizational or explanatory comments are used extensively, most are meaningful and easily understood. A consistent naming convention was used for most of the program and deviated very little. Source code was clean, consistently well-formatted and easy to read.		
				Total:		**/45**