Programming Assignment: Final Project

Determine Letter Grades

5185 Java Programming for Beginners

Instructor: Bineet Sharma

Summary: Write a program that will determine the letter grade of students of a class (university class, not java class) using your understanding of *class* (java class), objects, exception handling, and collection in Java.

Description: Your program will need to accept two command line arguments. The first argument will be the name of a disk file that contains the names of students, and their test scores, separated by commas followed by one or more spaces. Each line in the file will contain scores for one student. The second argument will be the name of an output disk file. Your program will create a new output disk file using that name. The output file will have grade information of all students whose information was read from input file, in a sorted order. You will be writing one line in the output file for one student. You will write name of one student and the letter grade he/she got in the class in each line (you should format the texts in the output file). The format of the data in the input file is fixed, however the number of students in the input file is unknown during compile time. The name of input and output files could be anything and only known during run time. Besides writing to the output file, you will also need to display the averages of scores along with minimum and maximum scores for each test in the screen/console.

Calculation: The test scores are weighed. There are four quizzes, 40% total, midterm I is 20%, midterm II is 15% and the final is 25%. All scores in the input file are recorded out of 100. You need to apply the weight for each score in the program to calculate the final score. The final score is tabulated as follows:

Final Score = quiz1 * .10 + quiz2 * .10 + quiz3 * .10 + quiz4 * .10 + midi * .20 + midii * .15 + final * .25
Determination of letter grade is according to the following logic:

Final Score >= 90% then letter grade is A, 80%-89% B, 70%-79% C, 60-69% D, <= 59% F

Sample input data file: input_data.txt (the input data format is: name, quiz1, quiz2, quiz3, quiz4, midi, midii, final,e.g.:

```
Thui Bhu,           100, 90, 80, 100, 89, 99, 88
Ariana B. Smith,    90, 90, 100, 100, 99, 100, 95
Emily Gonzales,     100, 90, 100, 70, 78, 78, 80
Jennifer L,         80, 90, 90, 100, 89, 99, 85
Maria Jones,        65, 72, 77, 68, 62, 70, 65
Bill Gates,         60, 54, 38, 62, 65, 60, 50
Escobar Morris,     83, 77, 88, 76, 79, 72, 76
Anne Latner,        80, 80, 85, 95, 90, 95, 90
..
<<more if there are more students>>
```

Note about input file: You can use the above data as template to create your input file. You can use any text editor of your choice including IDE. You can assume the data will be correctly formatted as described above when I test your program, however, I will have my own input file, with different name and different number of students. You should keep the input file in your project's default folder when testing using IDE. Refer to the separate document provided which shows you how to set the command line arguments to pass the input and output file names to your program.

Sample output file: output_data.txt (the output format is: Name: letter grade, sorted by name), e.g.

```
Letter grade for 8 students given in input_data.txt file is:

Anne Latner:                B
Ariana B. Smith:            A
Bill Gates:                 F
Emily Gonzales:             B
Escobar Morris:             C
Jennifer L:                 B
Maria Jones:                D
Thui Bhu:                   A
…
<<more if there are more students>>
```

Sample Run of the program (name of the application is **TestLetterGrader):** Remember that there are two sets of outputs. Letter grade is written in the output disk file (which is not shown in the screen), and score averages are displayed on the console (and are not written in the disk file). Here is an example of run in the command line you could also run this from your IDE (e.g. eclipse). You just need to provide these arguments in the class properties.

Example Run 1:

C:>java TestLetterGrader input_data.txt output_data.txt

Letter grade has been calculated for students listed in input file input_data.txt and written to output file output_data.txt

Here is the class averages:

	Q1	Q2	Q3	Q4	MidI	MidII	Final
Average:	*82.25*	*80.38*	*82.25*	*83.88*	*81.38*	*84.13*	*78.63*
Minimum:	*60*	*54*	*38*	*62*	*62*	*60*	*50*
Maximum:	*100*	*90*	*100*	*100*	*99*	*100*	*95*

Press ENTER to continue . . .

C:>_

Sample Run 2:

C:>java TestLetterGrader data1.txt data2.txt

Letter grade has been calculated for students listed in input file data1.txt and written to output file data2.txt

Here is the class averages:

	Q1	*Q2*	*Q3*	*Q4*	*Mid1*	*Mid2*	*Final*
Average:	*82.25*	*80.38*	*82.25*	*83.88*	*81.38*	*84.13*	*78.63*
Minimum:	*60*	*54*	*38*	*62*	*62*	*60*	*50*
Maximum:	*100*	*90*	*100*	*100*	*99*	*100*	*95*

Press ENTER to continue . . .

C:>_

Score: Maximum score you can receive for this final project is 105 and it is divided into the following categories:

1) Program compiles, runs, and provides the correct answers in the format as shown on screen and on disk. (70%)
2) The application design is modular and has good choices of classes and methods with proper data encapsulation. (20%)
3) Proper error checking, exception handling and descriptive comments are used (10%)

Design Suggestion: You need to model this problem into a software application – **TestLetterGrader** (driver class). Your final program should be implemented as Object Oriented and are instantiated using new. The driver class, **TestLetterGrader**, uses your main class, **LetterGrader** , which really determines the grade – meaning LetterGrader has all the code to determine the grader, TestLetterGrader simply uses the LetterGrader.

You need to identify classes, their hierarchies (inheritance and/or interface) to be used. The driver class will simply parse the command line arguments and use the main class to read the student's scores, do the calculation, output the result and close the files. Only **TestLetterGrader** and **LetterGrader** class are needed to be public classes. You will need only one main method in the application and that should be in the **TestLetterGrader** class only. You can have a main in **LetterGrader** for testing purposes, but, it will not be used when your application, the driver class, **TestLettergrader** is run.

An simplified pseudo code for the driver class (TestLettergrader.java), your application, may look like this:

```
public class TestLetterGrader {
      public static void main (String args[]) {
            //test if there are two valid arguments then, create the object
            //if not give right message and exit
           LetterGrader letterGrader = new LetterGrader(args[0], args[1]);
                                                          //LetterGrader is your main class,
                                                          //args[0] has input file name, and
                                                          //args[1] has output file name
           letterGrader.readScore();              //reads score and stores the data in member variables
           letterGrader.calcLetterGrade();      //determines letter grade and stores information
           letterGrader.printGrade();             //writes the grade in output file
           letterGrader.displayAverages();     //displays the averages in console
           letterGrader.doCleanup();            //use it to close files and other resources

           //remember you need to take care of any errors or exceptions
          // in any of these activities must be taken care of
   }
}
```

Submission requirement: Submit your source code, all java files (*.java), through UCSC web-portal (as attachments) before the deadline. You DO NOT need to submit anything else.

Naming requirements, especially for **public** items:

Important Class: The public class, which does all the work, reading input data, calculating the grade averages and printing data in disk file and console – **LetterGrader.java.**
Interface (is optional): **IGrader.java**
Package (is optional)**: utility**
Private Items: You can choose any names you want for private classes which does not affect file names. You can add all private classes in your main class, LetterGrader.java (yes you can have multiple classes in one .java file as long as there is only one public class in that file). If you are planning to inherit from a class of your own to your public class, in that case, leave the access modifier of parent class to nothing (that makes it a package access as you can't inherit from a private class).
Driver Class (your application): The public class which acts like a driver program is really the application name – **TestLetterGrader.java**.