CSE 174 – Fall 2021

lab6: 20 points – Due Saturday, Oct 2, 2021 by 11:59pm

Grade Calculator

Alan has a bunch of text files containing student first names and grades for his various classes. He wants the grades automatically transformed into letter grades, along with the number of students in his class counted and their average grade totaled. These letter grades and averages he then wants saved into another file, but this one with no identifying information (i.e., no names).

He could buy software to do it for himself, but that costs money. Therefore, he's asked you to write the program for him.

He has an example class with fake students and grades inside a text file, and called the file **class1.txt** (download the file here) which looked like the following:

```
 1 Jerry      86
 2 Eric       97
 3 Karen      90
 4 Valerie    72
 5 Patricia   99
 6 Jim        62
 7 Han        80
 8 Scott      94
 9 Derek      0
10 Marilyn    88
11 Elyse      26
12 Arthur     89
13 Winni      88
14 Jessie     41
15 Addison    93
16 Homer      65
17 Max        17
18 Carmela    72
19 Briget     61
20 Bob        11
21 Blanca     63
22 Roseline   95
23 Kelly      100
```

You can assume all grades are always integer values.

Continue on the Next Page!

He's created a mockup of what he wants the application to look like. This application should take in the name of the file to be read in and the name of the file where the data can be output to. Then, **after his program processes the file**, it will output the count of the number of students and the class average to the screen.

```
 ----jGRASP exec: java Lab6
Enter a file name: class1.txt
Enter an output file name: class1_out.txt
Number of students: 23
Class Average: 69.09
 ----jGRASP: operation complete.
```

In addition, he wants to output the results of each row to the file for error checking. Check out the results from his class1_out.txt file

```
1  B
2  A
3  A
4  C
5  A
6  D
7  B
8  A
9  F
10 B
11 F
12 B
13 B
14 F
15 A
16 D
17 F
18 C
19 D
20 F
21 D
22 A
23 A
24 Number of students: 23
25 Class Average: 69.09
```

Continue on the Next Page!

Alan doesn't use any + or - modifiers for grades, so you can just worry about printing letter grades relating to the below table:

Student Score	Letter Grade
Greater than or equal to 90	A
Greater than or equal to 80	B
Greater than or equal to 70	C
Greater than or equal to 60	D
Less than 60	F

How to start writing this program:

The following steps is a thought process for writing this program. You'll need to plan for what you need to do and write that down, and start implementing it step by step.

1. Getting an input filename from the user and creating a Scanner object with the given filename.
2. Getting an output filename from the user and creating a PrintWriter object.
3. Start reading from the input file as long as **there is a line** inside the file.
    - Skip over the name, we don't care about identifying features.
    - Determine the letter grade from the score (**read them as an int value**) following the student's name.
    - Write the result inside the output file.
    - Add up the original score so at the end (after the loop is done) the summation can be used to calculate a class average score displayed to the console, and be written inside the output file as well.
    - Count how many students are read so far from the input file so at the end (after the loop is done) can be displayed.
4. When the loop is done, calculate the average and display the student count and average on the console.
5. Print the student count and average inside the output file.
6. Close the input and output files (Don't forget to do this or else nothing will show up in your files!!).

Follow a Plan:

These steps are a simple and an effective way of starting a program:

- Coming up with an idea
- Coming up with a strategy for doing it
- Writing down everything step by step
- Implementing the code step by step

As a programmer, you need to get used to doing the same thing. As you practice more you get faster and faster in a way that you can plan the strategy in your head without writing it down (just like your instructors!). But, all programmers started from writing out all the steps.

So, open a new java file on jGrasp, call it **Lab6**, and start implementing all the steps.

Test your code comprehensively:

As a programmer, you always need to test your program using different inputs to make sure everything in your code works as expected. Create new text files and test your code with them. Check the result to make sure that your code is generating the exact result.

- You can always assume the user gives the correct filenames and there are always acceptable numbers (positive numbers) inside the given input file.

Additional tests:

There are three additional tests on the CODE plugin **that you are not able to see, but your code needs to pass them** in order to get full points. These additional tests simulate a real world situation when other people use your program and you never know how many values might be used inside the input file.

- One of the questions that you might ask yourself or your instructor is:
    - I can't see what values the Code Plugin is using to test my code with, so I can't fix my code when I can't see them!!!
- However, in the real world you may not actually be able to get this information, hence why we are giving you practice now. It is your responsibility to make sure your code can handle anything the CODE can throw at it!
- Think about all possible cases that your code can't handle and **ask your instructor about them** to see if your code needs to solve them, **instead of asking your instructor what the inputs used in hidden tests are.**
    - For example you can ask:
        - Should I be worried about dealing with negative numbers inside the input file?
        - Should I be worried if the input file is empty and there is nothing inside the file?
        - By the way, the answer to those two questions is no. You don't need to be worried about these two situations.

Continue on the Next Page!

Submission

After testing your code thoroughly, when you believe that you have correctly solved the problem, and that your code follows the style guidelines, submit it on canvas. If the autograder rejects your solution, fix it and resubmit it again.

Scoring Rubric (Full Rubric is on Canvas):

- **Successful Submission via CODE (3 points):** You will receive full credit for this if your code passes all the tests. Otherwise you will get zero or partial credit.
- **Additional Test Cases (9 points):** You will receive full credit for this if your code passes all the additional tests. Otherwise you will get zero or partial credit.
- **Correct Style (3 points):** You will receive full credit if your code has 0 style errors. Also your code has comments at the top at least with your name and description, and some comments inside the code to explain the code.
- **Proper Programming Practices (5 Points):** If your program is written by applying all of the programming practices **covered in class** (e.g. variable naming, minimal solutions, proper logic, avoiding hard coded values, using loops properly, etc.) you will receive full credit.