GPA Calculator

1 Introduction

This assignment will give you practice in console and file IO, arithmetic operations, values and variables, structured programming and methods. Read these instructions thoroughly, including the tips and help section.

2 Problem Description

As a (probably) freshman at Georgia Tech, you're quite worried about maintaining that perfect GPA. Being the industrious type, you decide to use your newfound Java knowledge to develop a program that calculates it for you for as many semesters as you want.

3 Solution Description

Write a Java program in a file named `GpaCalc.java` that:

1. Contains a helper method, `static void processInput()` that:
   (a) Prompts the user for the name of the semester they are computing GPA for, e.g, "Spring 2015"
   (b) Prompts the user to input the name of a course, the number of credit hours it is worth, and their grade in that course.
   (c) Accepts input and continues to prompt until the user is done.
   (d) Computes and displays the user's overall GPA rounded to two decimal places.
   (e) Writes to a file the original input from the user, as well as the calculated GPA rounded to 2 decimal places. Each course title, credits, and grade should be on a separate line. The file's base name should be the semester and extension should be `.txt`. The base file name should not contain any spaces and should be lowercase. For example, if the user entered "SprING 2015" as the semester, the output file would be `spring2015.txt`
2. Calls this method and then asks the user if they would like to run the program again.

4 Sample Output

Your program may look something like:

```
$ Enter the semester: Spring 2015
$ Enter the course title: CS 1331
$ Enter the number of credits: 3
$ Enter the grade (A=4, B=3, etc.) 4
$ Would you like to enter another course? (y/n) y
$ Enter the course title: ENGL 1102
$ Enter the number of credits: 3
$ Enter the grade (A=4, B=3, etc.) 3
$ Would you like to enter another course? (y/n) y
$ Enter the course title: PHYS 2212
$ Enter the number of credits: 4
$ Enter the grade (A=4, B=3, etc.) 2
$ Would you like to enter another course? (y/n) n
$ Overall GPA: 2.90
$ Would you like to calculate for another semester? (y/n) y
$ Enter the semester: Fall 2014
$ Enter the course title: CS 1301
$ Enter the number of credits: 3
$ Enter the grade (A=4, B=3, etc.) 4
$ Would you like to enter another course? (y/n) y
$ Enter the course title: ENGL 1102
$ Enter the number of credits: 3
$ Enter the grade (A=4, B=3, etc.) 3
$ Would you like to enter another course? (y/n) n
$ Overall GPA: 3.50
$ Would you like to calculate for another semester? (y/n) n
```

The corresponding files would be "spring2015.txt" and "fall2014.txt", and would contain:

```
CS 1331 - 3 credits. Grade: 4
ENGL 1101 - 3 credits. Grade: 3
PHYS 2211 - 4 credits: Grade: 2
GPA: 2.90
```

and

```
CS 1301 - 3 credits. Grade: 4
ENGL 1102 - 3 credits. Grade: 3
GPA: 3.50
```

5 Tips and Help

- Make sure you read this document thoroghly and understand its instructions.
- Make sure that when we ask you to name something a certain way, you name it *exactly* that.
- You may find that when trying to write to a file, you encounter a compile-error about about an "exception." As noted in the slides on IO, you can remove this error for now by adding `throws Exception` to the end of the method header inside which the error occurs.
- **MAJOR PROTIP PLEASE READ**: you'll almost certainly find that when reading input from the console you run into a very strange problem when calling `nextLine()` directly after alling `nextInt()` on a `Scanner`. This is due to a lingering newline character being left in the input stream that gets read by `nextLine()` before giving the user a chance to type anything. You can solve this problem by including an additionall call to `readLine()`, e.g.,

  ```
  int someNum = myScanner.nextInt();
  myScanner.nextLine();
  String someVal = myScanner.nextLine();
  ```

6 Checkstyle

You must run checkstyle on your submission. The checkstyle cap for this assignment is **10** points. Review the Style Guide and download the Checkstyle jar. Run Checkstyle on your code like so:

```
$ java -jar checkstyle-6.2.1.jar *.java
Audit done. Errors (potential points off):
0
```

The message above means there were no Checkstyle errors. If you had any errors, they would show up above this message, and the number at the end would be the points we would take off. The Java source files we provide contain no Checkstyle errors. For this assignment, there will be a maximum of **10** points lost due to Checkstyle errors (1 point per error). In future homeworks we will be increasing this cap, so get into the habit of fixing these style errors early!

7 Turn-in Procedure

Submit all of the Java source files you modified and resources your program requires to run to T-Square. Do not submit any compiled bytecode (`.class` files), the Checkstyle jar file, or the `cs1331-checkstyle.xml` file. When you're ready, double-check that you have submitted and not just saved a draft.

Please remember to run your code through Checkstyle!

Verify the Success of Your Submission to T-Square

Practice safe submission! Verify that your HW files were truly submitted correctly, the upload was successful, and that the files compile and run. It is solely your responsibility to turn in your homework and practice this safe submission safeguard.

1. After uploading the files to T-Square you should receive an email from T-Square listing the names of the files that were uploaded and received. If you do not get the confirmation email almost immediately, something is wrong with your HW submission and/or your email. Even receiving the email does not guarantee that you turned in exactly what you intended.
2. After submitting the files to T-Square, return to the Assignment menu option and this homework. It should show the submitted files.
3. Download copies of your submitted files from the T-Square Assignment page placing them in a new folder.
4. Recompile and test those exact files.
5. This helps guard against a few things.
   (a) It helps insure that you turn in the correct files.
   (b) It helps you realize if you omit a file or files. [1] (If you do discover that you omitted a file, submit all of your files again, not just the missing one.)
   (c) Helps find last minute causes of files not compiling and/or running.

[1]Missing files will not be given any credit, and non-compiling homework solutions will receive few to zero points. Also recall that late homework will not be accepted regardless of excuse. Treat the due date with respect. The real due date is midnight. Do not wait until the last minute!