**CPSC 24500: Reading and Writing To File**

**Summary:**

In this assignment we’ll build on from what we did last week by adding another two options to our terminal. Numerically they should be 5, and 6; while Exit should now be 7. The two options represent reading and writing to file. Your program will function the same, the user will enter a student, and once all the requirements are met, they will have the option of writing that student to a file. Option 5 will write the current student data to a file. Option 6 will read data. Whatever logical checks you implemented last week should now apply to number 5 instead.

The Administrator can then selection option 6 and display whatever student is currently written to the file. An important note: Right now we’re only writing 1 student to our file. Effectively each time we’re overwriting the file with a new student.

**Requirements**

Interface:

Same as week 1 with the following changes. The option to exit now becomes 7. Option 4 will remain, but rather than run the logical checks we wrote last week (is the data valid, is anything missing?) at this step, Option 4 will simply serve as a way for the administrator to check their work so far. You’ll want to run these checks on option 4 instead. The Menu would look something like this:

* 1. Enter the Students Name:
  2. Enter the Students’ Academic Year:
  3. Enter the Students GPA:
  4. Print Current Student.
  5. Write Data to File.
  6. Read Data from File.
  7. Exit.

Variables/Data Type/Methods for Program 1:

The assignment is a continuation from week 1. The data types and error checking for them should remain the same.

* 1. The student’s name is a string
  2. Their academic year a string (remember verifying this data against the acceptable list)
  3. Their GPA a double (remember it should be between 0.0 and 4.0).
  4. You will also need the Scanner and File classes added.

Error Handling:

Error handling from last week should be applied similarily. Ensure your user makes the correct selections: 1,2,3,4,5,6,7. Make sure they only enter in a double for GPA. Make sure the academic year they enter is one of these: Freshmen, Sophomore, Junior, Senior. Ensure GP is between 0.0 and 4.0.

Ensure you use try/catch when reading/writing to the file. File writing and reading is error prone and we want to make sure we gracefully exit if something happens.

All the same concepts of last week should still be applied. We’re just adding 2 new options to our program.

Last week there was an extra credit piece that required the user to answer all questions before they would be displayed to terminal. You must include this for this weeks assignment. The user cannot write to file unless they’ve answered all the questions (1,2,3).

If the user presses option 5, and the file is empty or doesn’t exist you must present them with a message informing them of this.

Methods:

Like last week 3 methods minimum. First our main method, a method to print the menu, and our error checking method. The error checking can be split into two if that’s easier for you.

2 new methods should be added. 1 method to write to file and the other to read from a file.

This means at a minimum we should have 5 methods. They are: main, printMenu, errorCheck,writeToFile,readFromFile. You can name them however you see fit, these are just examples.

**Extra Credit:**

* 1. Let the user specify a different file to read from. When the user presses number 6 to read from a file ask them for the full path. This is useful if, for example, a file needs to migrate to another directory, or is restored in some way. Ensure proper error handling if the user enters a bad path.
  2. Find a way to append data to the file instead of overwriting. **Hint:** how you open the file determines if you are appending or writing. Note writing will overwrite the file if it exists. This means every student the administrator creates and writes should show up in the file when I open or read it. **\*Note: You’re free to look this up. Oracle Documentation will have it, but others will as well.**

**Grading:**

It is assumed you’ve already implemented the bare minimum for assignment 1. If you did not, you will need to do so for this assignment.

* 1. Comments will always be required. Comment your code with your name to receive .5 points. Meaningful comments will get you the full 1 point.
  2. Your Program is named LastNameReadingWriting01.java. Replace LastName with your last name.
  3. The 5 methods specified (main, display menu, error check, write to file, read from file) are included in some form. If you implement more, I’ll still give you full credit as long as they’re implemented to do the same thing.
  4. The menu is displayed until the user selects the exit option.
  5. The user can answer question 1.
  6. The user can answer question 2.
  7. The user can answer question 3.
  8. The user can answer question 4.
  9. The user can answer question 5.
  10. The user can answer question 6.
  11. The user is able to exit by selecting 7.
  12. The user is not allowed to write to file unless all questions have been answered.

If your program crashes while executing you will lose 6 points. Ensure you have error handling for a graceful exit. Crashes happen, if you handle them you will still get some credit.

If your program cannot compile and execute will receive no credit. All code will be run from the Ubuntu VM. As always if you’d like me to run it on a different OS just let me know.

Last week a few of you were a little hesitant to use the code for clearing the terminal:

public static void clearTer(){  
System.out.print("\033[H\033[2J");  
System.out.flush();  
}

because it was on Stack. Some of you implemented some interesting solutions to clear the terminal without using this. I was impressed you thought outside the box like that, well done. However, when you’re using small pieces like this, or if you’re reading on how to do certain things with File Writing, I won’t take points off. The internet is a wonderful source of information, if you’re using something you learned that’s what we’re here for.

It only becomes a problem if you’ve been blatantly copying everything off Stack. For example, when we get into OOP an assignment will be to build a common board game. If you were to copy the entire program of say, someone else’s Monopoly game off Stack that would be a problem. But for specific problems like “How can I create a grid” or “How can I append to a text file instead of overwriting” it’s perfectly acceptable to look them up. After all, in days gone by we’d read documentation. Example is this week for extra credit you can append data to the file instead of overwriting. You’re welcome to “Google” this.

If you’re ever in doubt about something let me know, I’ll be glad to help.