CSCI 2120
Homework 3: Serialization

Introduction

For this assignment, starting with implemented classes `Student` and `StudentDatabase`, you will write classes that will serialize and deserialize an entire `StudentDatabase` (with very little code, and with very little concern for formatting).

Procedure

1) You are provided an implementation of a class that represents a Student, and a class that represents a `StudentDatabase`. We have also provided you with a class, `MakeRandomStudents`, that can be run to produce random Student data, in a specified format, and print it out to the screen. For example, running the following from the command line:

```
java MakeRandomStudents 100 > students.csv
```

will cause the program to generate 100 random students, and the ">" means "redirect standard output to the following file". You can use ">" to redirect the printed output of any program to a file from within the shell. A single ">" will overwrite the file if it exists, or create it if it does not. A double ">>" will APPEND to the file if it exists, or create it if it does not. Use this command to produce some test data to work with and have a look at the file produced. The ".csv" file extension stands for "comma-separated values". Have a look inside the file produced with a text editor and you'll get the idea.

2) You are also provided with a few other, fully implemented classes, `StudentDatabaseCSVFileReader` (which can read in the data you just generated from a file and build a `StudentDatabase` object from it), and a class that holds a main method `ReadSortAndWriteStudentDataFromCSVFile` which takes two arguments: the formatted text file to read in, and the formatted text file to write out. After reading, the Student objects in the `StudentDatabase` are sorted based on their GPA, and the data is written out to the file listed as the second argument. For example:

```
java ReadSortAndWriteStudentDataFromCSVFile students.csv sortedStudents.csv
```

will read in the formatted text file students.txt and write out the sorted data to a file called sortedStudents.csv.

The writer part, the class `StudentDatabaseCSVFileWriter`, is where the "writing out" happens. You'll notice that the write() method is not fully implemented (search for the string `YOUR CODE HERE` in the comments). You'll finish writing this method, so that the output is **in exactly the same format as the input file was**. Use both the Reader, and

the `MakeRandomStudents` classes as a guide to what the format is. **(30 points)**

3) Write two new classes, `StudentDatabaseSerializedFileWriter`, and `StudentDatabaseSerializedFileReader`, that, in a similar way, will serialize and deserialize the entire `StudentDatabase` in one write or read operation, respectively. You'll need to make both `Student` and `StudentDatabase` subtypes of `Serializable` to accomplish this. You can use the code of `ReadSortAndWriteStudentDataFromCSVFile` to get an idea of how these new classes would be utilized. **(70 points)**

A JUnit tester `StudentDatabaseConversionTester` will be provided for you. Run this to verify your classes.

BONUS (15 points – required for honors students): write a general program SortStudentDatabase, that takes the same arguments as the program above, but based on the suffix of the input and output files (.csv or .ser) chooses which mechanism to use to read and write the data.

Submission

You will add, commit, and push your .java files to Gitlab in a subfolder of your repository called HW3. You need not bother with JUnit testers for this project.

Grading

Your code should provide fully functional object serialization and should be able to easily serialize and deserialize the entire database at once.