Assignment 6

**Due:** Friday, April 2 by 11:59 PM

**Objective**

This assignment should give you practice with various aspects of serializing objects in Java and working with files of serialized objects. You will also get some more practice handling exceptions.

**Task**

Write a program to handle Student objects, including saving and loading a variable number of these objects and creating new Student objects. Your final jar file will contain the following classes:

* Student.java
* StudentHandler.java

StudentHandler.java will be built entirely by you to match the sample output as well as possible. Student.java will be edited by you. You will begin with the Student class found in the Student.java file on the Assignment 6 Canvas page.

**Each file should have a comment including your name at the top of the file. Each file should also have appropriate comments throughout. Make sure you indicate anywhere you change Student.java with comments**.

**Requirements**

1. **Student Class:** Most of the student class is provided for you. You will need to make changes to meet the following requirements:

* You should be able to serialize the Student class after making changes to it.
* All member data of the Student class should be serialized **EXCEPT** the double grade and any static variables (Java automatically does not attempt to serialize static member data).
* You should not NEED to change anything else about the Student class, but if you do, make sure you mark it especially clearly with comments.

1. **StudentHandler Data**: StudentHandler should contain one piece of data called students. This should be some collection of Students. You may store this in whatever way you feel appropriate, with the exception that it must allow at least 20 Student Records (in case you use something like an array). I recommend that rather than using an array of Students, you use something from the Java Collections Framework (an ArrayList should work wonderfully), but it is up to you.

1. **StudentHandler Class Methods:** This class will contain your main method (discussed below) as well as the following non-static methods.

* StudentHandler() – A single parameterless constructor, should just instantiate the member data
* void saveStudents(Scanner s) – This method will ask the user for a filename to save under and save the entire collection of students held in this StudentHandler as serialized objects
* void loadStudents(Scanner s) – This method will first clear all students out of the StudentHandler, then ask the user for a filename to load and load an entire collection of students into the StudentHandler
* void addStudent(Scanner s) – This method will prompt the user for the various pieces of a Student object (name, grades, etc.) and create a new object and add it to the StudentHandler
* void printAllStudents() – This method will print all of the students, sorted by name (last name first, if those are the same, then sort by first name) as well as the count of the number of student records (you are **REQUIRED** to use the static totalStudents member data for this count)
* void clearAllStudents() – This method will clear all of the students from the collection/array

1. **The Main method**: First, you should create a StudentHandler object. From there, your main method should print the menu (as seen in sample output) and ask the user for input. The user should input an integer and then you call the appropriate method (on your StudentHandler object) based on the user input. When the user inputs '6', then the program should quit.

1. **Error handling**: You will need to handle various errors and exceptions in this project. Each of the following errors should be handled appropriately (as indicated):

* Invalid menu input: If the user inputs invalid menu input (either a non-integer or an integer not between 1 and 6), then print an appropriate error message and go back to the beginning of your loop
* Invalid input in addStudent() method: If the user inputs an invalid item for any of the inputs in this method, print an appropriate error message and begin the method again
* File exception: If there is an error reading a file for reading or writing, indicate this to the user and go back to the main menu
* Invalid Student File: If the user attempts to read from a file that does not consist of only Student objects, this will trigger a ClassNotFoundException which you should handle by printing an error message and going back to the main menu

1. Hints:

* You only need the classes/methods I mention above, but if you want to add more, feel free (make them private, please)
* Remember to use ObjectOutputStream and ObjectInputStream for your file I/O
* As mentioned above, it is easier to use a collection such as ArrayList to hold your students, so you can use the built-in functionality, but it is not required
* Pay attention to the non-serialized data when loading data from the serialized file
* Exceptions don't "use" the input from a Scanner, so if you are using a Scanner, make sure to include a line like s.NextLine() in your exceptions to process the bad input
* Again, remember that you are **REQUIRED** to use the static totalStudents member data when printing the number of students, do not simply count the elements in your collection

**Sample Output**

User input is underlined (there has been input before this, so there will be 3 total Students)

1: Print out all loaded students

2: Add student

3: Clear students

4: Save students to file

5: Load students from file

6: Quit

Please input the number of your choice: cd

Invalid choice, try again.

1: Print out all loaded students

2: Add student

3: Clear students

4: Save students to file

5: Load students from file

6: Quit

Please input the number of your choice: 2

Please input a first name: Joe

Please input a last name: Blow

Please input student homework grades one at a time (negative value to finish): c98

Invalid input, please try inputting the student again

Please input a first name: Joe

Please input a last name: Blow

Please input student homework grades one at a time (negative value to finish): 98

Please add another homework grade (negative value to finish): 77

Please add another homework grade (negative value to finish): -4

Please input student test grades one at a time (negative value to finish): 75

Please add another test grade (negative value to finish): 88

Please add another test grade (negative value to finish): 99

Please add another test grade (negative value to finish): -2

1: Print out all loaded students

2: Add student

3: Clear students

4: Save students to file

5: Load students from file

6: Quit

Please input the number of your choice: 4

Please input the filename to save as: TheFile.txt

1: Print out all loaded students

2: Add student

3: Clear students

4: Save students to file

5: Load students from file

6: Quit

Please input the number of your choice: 1

First name: Joe

Last name: Blow

Final Grade: 87.41666666666666

First name: Another

Last name: Student

Final Grade: 89.26666768391928

First name: Test

Last name: Student

Final Grade: 89.5

Printed 3 Student Records

1: Print out all loaded students

2: Add student

3: Clear students

4: Save students to file

5: Load students from file

6: Quit

Please input the number of your choice: 3

1: Print out all loaded students

2: Add student

3: Clear students

4: Save students to file

5: Load students from file

6: Quit

Please input the number of your choice: 1

Printed 0 Student Records

1: Print out all loaded students

2: Add student

3: Clear students

4: Save students to file

5: Load students from file

6: Quit

Please input the number of your choice: 5

Please input the filename to load from: TheFile.txt

1: Print out all loaded students

2: Add student

3: Clear students

4: Save students to file

5: Load students from file

6: Quit

Please input the number of your choice: 1

First name: Joe

Last name: Blow

Final Grade: 87.41666666666666

First name: Another

Last name: Student

Final Grade: 89.26666768391928

First name: Test

Last name: Student

Final Grade: 89.5

Printed 3 Student Records

1: Print out all loaded students

2: Add student

3: Clear students

4: Save students to file

5: Load students from file

6: Quit

Please input the number of your choice: 6

Goodbye!

**Submitting**

Pack all of your files (class files **and** source code) into a **fully runnable** JAR file called hw6.jar. The main program that the jar file should execute should be in class StudentHandler. I should be able to run the main() method from your file with the command:

java -jar hw6.jar

Submit your jar file via the Canvas submission link for Assignment 6.