**Java 1: Test #4**

Due: Jan 17, 11:30pm

Please remember to read over the assignment rubric before answering these questions. Each question should be written in its own file for upload. When all questions are answered place the source code for each question in a folder, zip it up, then upload it to the test 4 upload area on the LMS. If there are any concerns, let me know right away. Feel free to work with others to solve the questions but please code them on your own.

**Question 1: [100 marks]**

**This question alone accounts for all the marks of Test 4.**

* **Please read the question carefully.**
* **I have provided expert instructions to get this program working with a software engineering design pattern I use at work, which organizes code in a maintainable format and includes best practices. This design pattern is known as MVC.**
* **Ensure you are recognizing how modular the code becomes throughout the process.**
* **You will notice separation of concerns throughout the classes ensuring dependencies of objects are kept separate from one another.**
* **Your grade for this question will be solely based on how well you can follow the instructions for the implementation provided, the output of the program, and how well the code was written in terms of the DRY (DON’T REPEAT YOURSELF) principle.**
* **Good luck!**

Write a program utilizing Java’s GUI library to create a user interface (UI) shown in the screenshot below. The UI will contain a form with four labels and four matching text fields showing a student’s data. To store a student’s data a model of type student should be created containing these four fields. The defaults should be set in the default constructor. The student’s data is as follows:

* Student id. (Default value 0)
* Student first name. (Default value N/A)
* Student last name. (Default value N/A)
* Student average. (Default value 0)

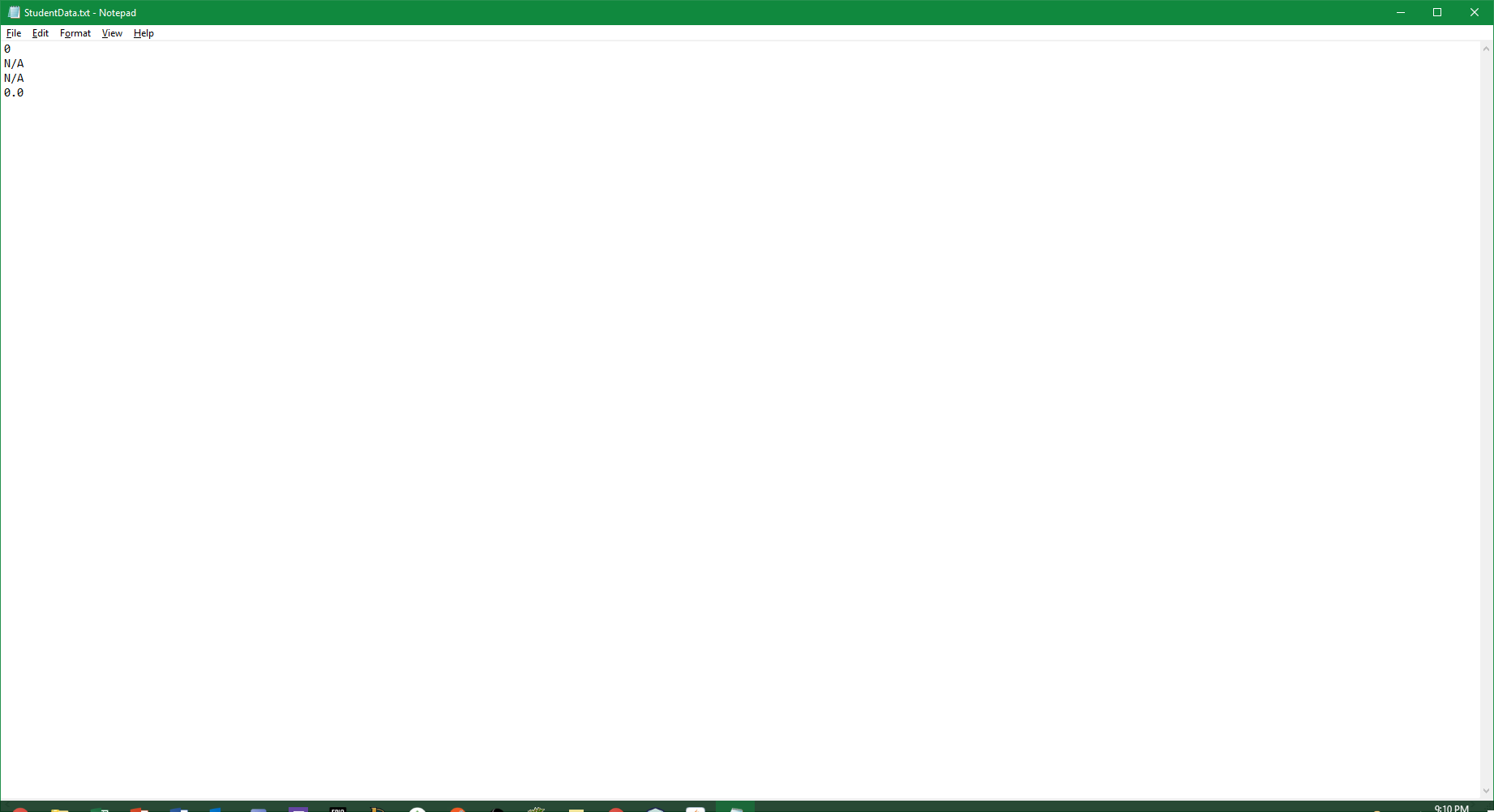
Below the form will be three buttons, which serve different purposes. **To make things clear we are just working with one student in question. The text file will hold data for only one student and any modifications to the data will effect that one student.** Using event handlers the buttons implementations should be as follows:

* **Get Data:** Should retrieve then display a student’s data fields stored in a text file. Your program should read from the student file then construct the student object based on the data read from the file. Once the student object is created the object’s getter methods should be used to populate the text fields. A screenshot of the text file is below. Please note that the text file should have the defaults of the student object written to it already. (Meaning you have to put this data in the text file manually)
* **Set Data:** In the text fields for the student, a user should be able to enter the data they wish and this button should rewrite to the student text file with the new data. If done correctly, when you re-run the program and press the get data button the updated student data will be displayed. **Do not worry about validation; assume the user enters the correct data.**
* **Reset Data:** This button once pressed should simply reset the student’s form fields to their default values as mentioned above. The concept of this button event is similar to set data except you are writing the default values of student to the file. Ensure the operation of this event updates the fields with the defaults by using the implementation for the get data button. This button should create a default student object then use the getters to retrieve the data when writing to the file.

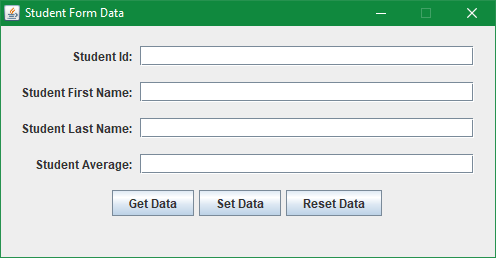
**Program Setup:**

1. Make a package called **Models**.
   1. This package should only contain a single file being the student class.
   2. The class should have the student fields as mentioned above.
   3. All data members should have basic getters and setters.
   4. The default constructor should make a default student object with the default data values as mentioned above.
2. Make a package called **DataService**.
   1. This package should only contain a single file, containing a class, which is named **PrimaryDataService**.
   2. This class should not have any constructors, the implicit one Java defines is acceptable.
   3. This class should have only three methods.
      1. **readStudentData():** Reads student data from the file and sets the properties of the student object while being read. This method should have a return type of student. Hence, it returns the student object read.
      2. **writeStudentData(Student student):** Has one parameter of type student where the student object from the method’s parameter has its data members read from its getters in order to be written to the student file. This method should not return anything.
      3. **resetStudentData():** Has no parameters and no return type. This method simply creates a default student object and writes its data to the student file with the use of its getter methods. Please note the implementation of this method is identical to the **writeStudentData(Student student)** the only difference is a default student object is created to have its values written.
3. Make a package called **Controllers.**
   1. This package should only contain a single file, containing a class, which is named **PrimaryController**.
   2. There should only be one constructor for this class.
      1. The constructor does not take any parameters.
      2. The constructor simply creates an instance of the **PrimaryDataService** class and sets the created instance to a data member of type **PrimaryDataService.**
   3. This class should have only three methods.
      1. **getStudentData():** This method simply returns a student object. This student object is acquired by calling the <PrimaryDataService Ref Variable>.readStudentData() method from the class data member as set in the constructor.
      2. **setStudentData(Student student):** This method accepts a student object as a parameter and utilizes the <PrimaryDataService Ref Variable>.writeStudentData(**Student student**) method to pass the student object to.
      3. **resetStudentData():** This method does not return any value. It simply calls the <PrimaryDataService Ref Variable>.resetStudentData() method.
4. Make a package called **View**.
   1. This package should contain two files.
      1. The first file is named **Driver**, which simply starts the program by creating a new object of type **PrimaryView**. The showApplication() method should be called to show the GUI.
      2. The second file should be called **PrimaryView**.
   2. The specifications of the **PrimaryView** class is as follows:
      1. Extends JFrame.
      2. A constructor, which creates a default instance of **PrimaryController** and sets this object to a data member in the class.
      3. A method, which will create the GUI as shown in the screen shots. You can call this method **constructorFramePanel()**. Should be private. Please note you may need to make some components data members of the class. This is up to you to figure out.
      4. A method called **refreshStudentData()**. This method corresponds to the implementation of the get data button action listener. Should be private.
      5. A method called **updateStudentData ().** This method corresponds to the implementation of the set data action listener. Should be private.
      6. A method called **resetStudentData ().** This method corresponds to the implementation of the reset data action. Should be private.
      7. A method called **showApplication().** This method should be public and sets the properties of the JFrame and should call the **constructorFramePanel()** to get the panel to add to the frame.
   3. Note: The action listeners for the buttons should be anonymous or lamda expressions. The code written in the action listeners should just simply call the respective methods as mentioned above.

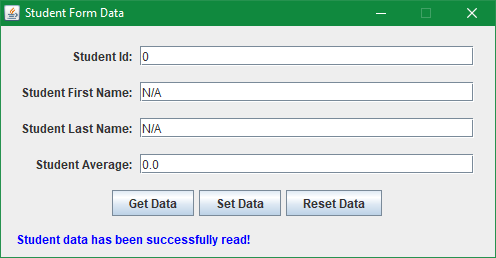
**Output Screenshot 1:** The following is a screenshot of the text file for the student object. Notice how it has the defaults set. You need to write these into the file before it is run.



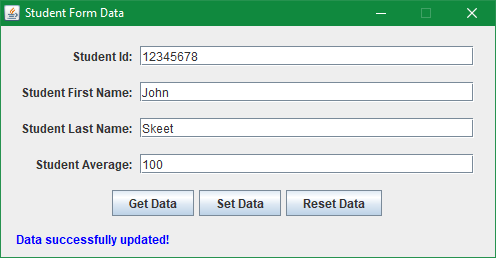
**Output Screenshot 2:** The following is a screenshot when the program is first run. No data is to be displayed in the text fields.



**Output Screenshot 3:** The following is a screenshot when the get data button is pressed for the first time. The text file with the default values are read and displayed in the text fields.



**Output Screenshot 4:** The following is a screenshot when the set data button is pressed after entering the data in the text fields.



**Output Screenshot 5:** The following is a screenshot when the reset data button is pressed. After the data is written to the file the implementation of the get data button is used to display the reset data to the screen.

