EAST TENNESSEE STATE UNIVERSITY

CSCI 1260 – Introduction to Computer Science II

Project 4 – Persistent Class Roll Manager

Overview

This phase involves adding persistence to the **RollManager** that we developed in the previous phase of our project. We will preserve the data in a text file that we can restore the next time we use the Roll Manager by reading the data from the text file.

Specifications

Exceptions

The program should handle all **Exceptions** in a manner appropriate to the application. Remember that the user of the program may have made many additions or changes to his/her **class roll**, and the user would like a chance to save the updated **class roll** contents with those changes if possible before the program is terminated in case of an exception.

Text File

Using the test data you specified for the previous project in your design document, create a "|"-delimited text file named **Contacts.txt** using **Eclipse, Notepad++**, or another text editor. The line should have the following content in the order shown: **first name|last name|major|classification|gpa|hours|photo**. An example of a line in the file:

Chester|Drawers|CSCI|Sophomore|3.24|43|Chester.jpg

The first line in the file should be contain the **Course Information** such as **CourseNumber|CourseName|Instructor.** Subsequent lines represent individual **Students** found in the class roll. This file should have at least 10 **Students**, including at least one with invalid information. Have some fun making up the data if you are so inclined, but please don't use meaningless jumbles of random characters.

The file should be in a **folder** named **RollData** in the new project that is not in the **src** folder or other project subfolder. If you wish to keep old copies of the **RollBook** or to keep additional **RollBooks**, the **RollData** folder may contain as many text files as necessary.

RollManager Class Modifications

Add two **methods** and one **attribute** with appropriate **getter/setter** methods if needed to the **RollManager** class as follows:

- **saveNeeded** – **Boolean** attribute that indicates when changes have been made to the **class roll** since the last time it was saved so that the program can save a modified **class roll file** before exiting or before opening a different **RollManager** file.
- **fillFromFile** – method takes a string parameter representing the filename (and path) of an existing text file from which the method should populate the **RollManager (class roll)**
- **saveToFile** – method takes a string parameter representing the filename (and path) of a text file to which the method should save the contents of the **RollManager (class roll)**, replacing anything currently in that file

In addition, each method in the class that makes any changes to the **RollManager class roll** should set **saveNeeded** to **true** and any methods that have filled the **RollManager class roll** or saved the **RollManager** should set **saveNeeded** to **false**.

All interaction directly with the user should remain in the driver class.

Driver Modifications

The main menu for the driver will need additional choices for the following tasks.

- **Creating a new RollManager class roll**
- **Populating a RollManager class roll from a file**
- **Saving a RollManager class roll to a file**

In addition, the driver should be modified to check the **saveNeeded** status of the **existing RollManager** on which the program is **currently** working whenever the program is about to end and whenever it is about to create a new **RollManager class roll**. If a **save** operation is needed, the driver should save the **RollManager** class roll to a text file using the appropriate method of the **RollManager** class before continuing.

Whenever **file input** or **output** is about to be required, the driver should display the appropriate **OpenFileDialog** or **SaveFileDialog JFileChooser** and allow the user to select an appropriate text file to be processed. The **JFileChooser** dialogs MUST open into the **RollData folder** of project initially. The **path** of the **selected file** should be passed to the **fillFromFile** or **saveToFile** method as appropriate in each case.

Additional Requirements

Ethical Issues

Please read the Software Engineering Code of Ethics and Professional Practice found at http://www.acm.org/about/se-code/. Specifically consider principles 3.11 through 3.14, and how they relate to this specific project. Please state your answer as concisely as possible. Format, spelling, grammar, and so forth count. Submit a Microsoft Word document discussing how these principles relate to the project you are developing. Include this in the same zipped file as your code, named **FirstnameLastname_Project4_Ethics.docx**.

Project Documentation

Project documentation is NOT optional in this course. It is absolutely required in **every** programming assignment. See the **Documentation Standards** on the course website for details and for examples of proper documentation.

An otherwise perfect project may still receive a **failing grade** if the required documentation is missing, incomplete, or poorly done. **Eclipse** will help write much of the documentation for you if you set it up and use it as described in the first labs for this course.

Submission

Create a ZIP file named LastnameFirstname_Project4.zip (e.g., BennettBrian_Project4.zip). Include the following files.

Ethics Document

Place your completed ethics writeup (properly named) in the root of the ZIP file.

Code

Include a Source folder in the ZIP file. Place all source code into the Source folder in the ZIP file. If you navigate to your Project 4 folder in Windows Explorer (or Finder), you should see a src folder. Copy all contents (and the structure) of your src folder to the ZIP file's Source folder.

Due Date

The completed ZIP file must be submitted to the dropbox at or before 11:59pm on Thursday, March 31, 2016.