Introduction to Homework 11 : Student Management System

For HW11, you may work as a group (no more than 2 students). Note: As an alternative to this assigned homework, you may have already defined and been approved for an optional final project of your own design.

This is just an introduction to Homework 11. More details and instructions will be provided once the assignment is officially "assigned".

This homework deals with the following topics:

- Object-Oriented Programming Design
- Inheritance
- File I/O
- Regex

Introduction

In this homework, you will implement a console-based student management system. The objective of this assignment is to design a system for students to manage their courses. There will be three main user roles in the application: Admin, Student, and Professor.

In the student management system, a) A student can log in to his/her account, view/add/drop courses, check their course schedule, and view grades. b) A professor can view course information he/she has, and view the student lists for these courses. c) An admin can view course/student/professor lists, and add/delete courses/students/professors. The course information will be in the *courseInfo.txt* file. There will also be *three files* containing student/professor/admin information. The student management system will read and parse all of the files. Once all information has been loaded into the system, you'll be able to log in as a(n) student/professor/administrator to test the system.

Your Task

1. Read and parse the provided files in Java, cleaning them up if/when needed. You may assume the information in the files is valid.
    a. Courses information file – *courseInfo.txt*. This file contains information for a number of courses. The information for each course is on a separate line. The pieces of information for each course are separated by semicolons (";"). We want you to read the file in and load the information.
    b. Admin information file – *adminInfo.txt*. This file contains basic information for administrators including username, password, name, and ID. You need to read the file and load the information.

c. Student information file – *studentInfo.txt*. This file contains basic information for students.
d. Professor information file – *profInfo.txt*. This file contains basic information for professors.
e. You may assume all of the information in the files is valid. For example, all information for professors in the *courseInfo.txt* file is given in the *proInfo.txt* file. We suggest that you load the list of professors before loading the list of courses.

2. Design the students management system
We are not going to provide you with a specific design for this homework. Feel free to design your own student management system. Here are some suggestions:
   a. You need a **FileInfoReader** class that reads and parses the files.
   b. You need a class that defines a **Course**. The **Course** class is expected to have instance variables which store all of the information for the course. It also needs to provide functionality, for example, checking if one course has a time conflict with another course, adding/removing students from the course, and other helper methods you may need.
   c. You need classes that define a **Student**, **Professor**, and **Administrator**. And they are expected to share some common attributes and to share as much code as possible. We want you to think hard about how you can accomplish this. This answer lies within the scope of the object-oriented concepts we have covered in this course. For example, you might have a **User** class with a subclass **Student** for the functionality of student operations like adding/dropping/viewing courses, a subclass to represent **Professor**, and a subclass **Admin** to add/delete/view information for courses, students and professors.
   d. There should be a **Controller** class that launches and runs the management system, displays the menu, helps with user login, accomplishes the different user operations and continuously takes in user input.

Last, but not least, if you are not given a design to stick to, it might be in your best interest to start with a piece of paper and lay out what your classes and methods will be. In other words, do not dive into Eclipse and expect things to work out without some forethought.

Testing

Regardless of your design, we expect you to write unit tests for every public method. The only public methods that you can leave untested are those that

perform file I/O, and simple getters and setters. You should keep the file I/O in as few methods as possible.

Evaluation

You will be graded out of 40 points:

- Code writing, functionality, and design (20 pts)
  - Did you set up your program with the basic required package structure?
  - Does the system work as expected?
  - Did you make good design decisions about code reuse?
    - How much code is being shared between Student, Professor and Admin?
    - How is this code sharing being achieved in your design?
  - Does the code take too long to run? (10 seconds would be too long!)
- Loading, parsing, and navigating files (10 pts)
  - Did you keep file I/O in as few methods as possible?
  - Did you correctly load valid data from every file?
  - What data structure(s) did you use?
- Unit testing (5 pts)
  - Did you write unit tests for every public method (excluding methods that perform file I/O and simple getters and setters)?
- Style & Javadocs (5 pts)
  - Adding Javadocs to all methods and variables, and comments to all non-trivial code
  - Generating HTML files from the Javadocs in your code using Eclipse