CS3360: Design and Implementation of Programming Languages

Programming project #1
Due: September 20th, 2013 at 11:59pm via email to your TA

Email project to: afgarciacontreras@miners.utep.edu

Notes:

1) This project is to be done ***individually***.
2) The program must fulfill the requirements, regardless of the interface. This means that the quality of the interface will not be graded, as long as it clearly shows that the requirements are fulfilled.
3) The grading of this project will be as follows: ***40% for the program*** you have to implement, ***55% for the report*** (*see below for details*), and ***5% for your progress report*** at mid-point, in person to your TA.

Goals of the project: Demonstrate understanding and fluent practice of the features of object-oriented programming languages (in this case, Java). Understand the features of aspect-oriented programming languages (in this case, AspectJ) and be able to put them to practice and reflect on their use.

To be done:

1) In the first part of this project, you have to implement a program that does the following:

- It implements a CS degree plan management tool.
- It helps CS students manage their degree plan. It also helps advisors see the degree plans of their advisees (based on students-users' ID).
- It allows users (students) to keep track of their own degree plan by logging in and entering their courses and grades each semester (or, for the purpose of this project, whenever they want), and it keeps track of their GPA.

In addition, it has at least the following functionalities for students:

- it allows users to add courses to their plan (electives and the choice of core curriculum courses differ from user to user for instance);
- it allows users to remove courses from their plan (if they change their minds and drop a course for instance);
- it automatically removes from the plan courses that have been failed (unless taken again), but keeps them in the computation of the GPA;

    - each course has general information, such as a title, a course number, and user-specific information, such as a semester attached to it, and a grade.

- It allows users (advisors) to validate degree plans. Degree plans should be validated, otherwise the corresponding student account is frozen (there is a hold on it) and the student notified.
- It allows both students and advisors to print the degree plans they have access to.

Your program should exhibit the following OOP features:

    - Inheritance
    - Polymorphism
    - Overridden methods
    - Overloading

and have:

    - At least one abstract class

2) In the second part of your project, you have to add the following features, using AspectJ for at least three of the features, to your code implemented for part I. These features are as follow:

- It keeps track (in two different lists) of the names of the students whose GPA is falling under 2.5 and 2 respectively. Advisors will be able to access only the part of the lists that is about their student advisees (not students advised by someone else).
- It keeps track of the log-ins. Advisors should be able to see when their advisees have logged in over a give period of time.
- It does not allow a student to enter a course more than 3 times (if 3 times have been failed).
- It does not allow a student to enter a course if it has been taken already and passed.
- It does not allow a student to remove a course that already has a grade assigned to it and is not a failed course (failed courses are removed automatically anyway).

3) In a supporting document:

1st part:

- Describe what you program does and how to use it;
- List the class variables, instance variables, class methods, instance methods;
- List and justify the use of your global variables.
- List and justify the use of 4 local variables.
- Justify the inheritance hierarchy you designed, the need for overriding methods, the choice of your abstract class(es).
- Describe what you program does and how to use it;

2nd part:

- List and describe the changes you made from your original OOP (Java) program;
- List all aspects you implemented and describe (justify) them;
- List all tasks (if any) that you did not implement as an aspect and justify why;
- List the possible variables and methods affected or concerned by the aspects you implemented;

Finally:

- Reflect on your original implementation of the degree plan management tool. In particular: could you have implemented your original tool differently so that implementing the aspects on top of it would have been easier or not necessary? Describe how. If you believe your original implementation was just fine, justify why.

Note: *a template of the report will be provided on piazza.*

<u>Deliverables:</u>

- Code of the first part of the project (Java only)
- Code of the enhanced project (see second part: Java and AspectJ)
- A readme file for the two programs
- A report

Note: Make sure to follow best programming practices (including proper indentation, commenting, documenting, …) as well as the report template (provided on piazza by your TA).

<u>Upcoming deadlines:</u>

- *PHP programming assignment: October 11*
- *Functional programming assignment (Haskell): November 8*
- *Logic programming assignment (Prolog): December 6*