ISYS 1117/1118 Software Engineering Fundamentals

Assignment 2 – Student Management System Implementation (Draft) 2015 Semester 1

Objectives

This assignment is designed to guide the students to move from the problem domain to the solution domain and to expose students to some of the tools common in industry. Specifically you will reverse engineer the code given to you to arrive at some initial design, before extending them. Subsequently, you will implement and test your design using Java, version control and testing tools.

Mark Allocation

Students are expected to work individually (20 hours) and as a team (30 hours) for this major assignment. Group marks (15) will be based on 4 separate diagrams and your working code. Individual marks (10) will be based on your individual design and coding in one part of the system.

Group Part (Total 15 marks)

1. Study the Java code given and draw a class diagram. **(2 mark)**
2. Draw a state diagram and an activity diagram for the following scenario: **(2x2 marks)**
   A lecturer can choose to shortlist or reject an applicant. An applicant must be shortlisted before the administrator can make an offer. An applicant can choose to accept or reject an offer at this stage. If the offer is accepted within 2 days an applicant will be appointed otherwise it will be automatically rejected. An applicant can also choose to reject an offer (explicitly). Please note the client has updated the requirements, since assignment 1. Also, you do not need to show the Tutor in this diagram.
3. Draw an extended class diagram that shows all the classes (including abstract), attributes, methods, associations and multiplicity where: **(4 marks)**

a) Administrators can add new course offerings. Only one per course in any one semester. (*****)
b) Administrator can add lectures specifying day, time and venue (*****)
c) Administrator can assign lecturers to the lectures(*****)
d) Administrator can add tutorials specifying date, time and venue. Tutorials for each course offering should be labelled T1, T2, …
e) Administrator can appoint suitable applicants as tutors (Note: shortlisting/deleting not handled in this iteration.
f) Administrator can assign tutor to specific tutorials
g) Administrator can admit students and grant exemptions for those with advanced standing
h) Students can enrol into courses for which they meet the necessary prerequisites
i) Students can withdraw from a course until census date (after deregistering from any tute)
j) Students can register (as long as tute is not full) and deregister from tutorials
k) Users can view timetables for any venue, lecturer, tutor or student
(***** indicates already completed)

4. You are required to implement all of the use cases above. You may work as a team for this section but for the individual part you must select two use cases each (which are not marked with * as they are partially implemented in the code given). **(5 marks)**

Individual Part **(Total 10 marks)**

1. Each student must present two sequence diagrams based on use cases in page 1. **(5 marks)**
2. Write 5 test cases per student. Each student must do at least three significant test cases from the list below. Some sample JUnit code is given in AppTest.java. **(5 marks)**

Significant Test Cases

- System should prevent students from enrolling in more than 4 course offerings
- System should prevent not allow a student (identified uniquely by first-name, surname and phone) to be admitted twice.
- System should not allow students to enrol into a course offering without the necessary prerequisites
- System should prevent students from enrolling or withdrawing after the census date (users must be allowed to vary the census date for testing purpose)
- System should not allow students to register or deregister from tutorials after the census date (users must be allowed to vary it for testing purpose)
- System should prevent more than one offering per course in any semester (*)
- System should prevent any tutor from being assigned two tutorials at the same time
- System should prevent overlapping tutorials in any venue
- System should prevent students from enrolling into an offering twice
- System should prevent students from registering into a tutorial twice
- System should prevent students from registering into a tutorial that is already full
- System should ensure each timetable is complete and consistent (matches with others)

Bonus Marks (up to 5)

No additional enhancement should be attempted until after completing the basic part

1. Separate menus for separate actors (with distinct passwords) (2 marks)
2. Ability to save and retrieve the objects (consider serialization) (2 marks)
3. Use of GUI (Buttons, tables, menus etc.) (2 marks)

Submission Details: 29th May 2015 4.30 pm. No late submission allowed.

All code and diagrams must be printed and bound together. This document will become handy in your capstone projects and your future employment. Please note your work must be submitted to the general office by 4.30 pm on the last day of the semester (29th May 2015). Your presentations will commence on 1st June.

FAQ

1. Can I redesign the system for the enhanced class diagram? Yes, as long as you cover all the use cases and testing requirements. You need not use/extend the initial code given.
2. Do I need to write the test code in JUnit? Yes, this is a standard API and commonly used. It comes as part of Eclipse. JUnit Information available on: http://junit.org/faq.html
3. Must I use GitHub? Yes, unless you have a very valid reason (consult with head tutor).
4. What additional help can I get? We will organize more feedback sessions in week 9-11.
5. Do I need to show exceptions in class diagram? No they will clutter the diagrams.
6. Must I use JCF? No, but it makes the programming much easier once you get the hang of it. There will be some notes on JCF in the assignment folder.

Assumptions and Scope

- Lectures are taken by a single lecturer but a lecturer can take any number of lectures.
- Tutorials are taken by a single tutor but a tutor can take any number of tutorials for one or more courses.
- Venues are not used for any other purpose and can be used from 8.00 to 21.00 Monday (day-of week 1) to Friday (day-of-week 5).
- Only one offering allowed for a course in any particular semester.
- Name, address and phone must be kept in the system for tutors and students.
- Tutor payments and bank details need not be maintained by the system.
- The system need not allow for change of lecture, tutorial, lecturer or tutor in this iteration.
- A student must be admitted to the program (college) before being allowed to enrol.

How do I get started with the given code?

In Eclipse choose File - Import – Existing Projects into Workspace – Select Archive File Option – choose the file "AMS.zip" and you are ready to run the program. In the menu choose the View Menu option to view the existing list of lecturers, venues and courses. The code will be briefly explained in the lecture in week 8.

Sample Code

Sample code is given only to help you get started as soon as possible. You are free to change/enhance the design and code.

Classes

Class	Purpose
AppTest	JUnit tests
App	Main System level class
Course	Definition class
CourseOffering	Changes from semester to semester
Lecture	Captures all details about lectures (the lecturer need not be known when it is created)
Lecture	Captures all details related to the lecturer
Lesson	Superclass for Lecture (and possible Tutorial)
Venue	stores all the venue related details
Menu	Helper class
ClashException	Exception class thrown whenever clash detected
PreExistException	Exception class thrown whenever an object already exist

How do I get help with JUnit?

http://junit.org/faq.html

Possible Process (Activity Diagram) for Assignment 2