RMIT University - Vietnam

School of Science and Technology

EEET2582 – Software Engineering Architecture and Design

Semester 3, 2019

Individual Assignment – 30%

Due: 4pm on Monday 02/12/2019

Objectives

The purpose of this assignment is to bring together the software architecture and software design patterns topics we have discussed, and assess your progress towards attainment of a selection of the learning objectives of the unit and your problem-solving skills.

On successful completion of this assignment you should be able to:

- ➢ encapsulate data/state information, and behaviours within an object
- ➢ write programs consisting of multiple objects which interact with each other to achieve the functionality of the system
- ➢ write a graphical, event-driven program to coordinate the interactions of objects
- ➢ write code to achieve call-backs using interfaces (mainly with respect to GUIs)
- ➢ utilise polymorphism and abstraction techniques (mainly with respect to GUIs)
- ➢ read objects from and write objects to binary format files; and read and write text to text files
- ➢ anticipate possible exceptions that could arise and write code to recover from them
- ➢ use collection classes from the Java API, including Map.
- ➢ write comparators to sort data according to criteria.
- ➢ invent and generate programmer-defined exceptions

Completing the assignment helps to demonstrate that you have attained the following course learning objectives

- Describe the essential elements of software architecture and software design;
- Critique existing architectures and designs;
- Discuss the issues related to architecturing and designing a large-scale software system;
- Describe and compare different software architecture styles and design patterns;

You should start working on the assignment even all topics that are being assessed were covered.

Overview of Assignment

You will develop a computer program, intended for use by the School of Science and Technology, RMIT University. The program allows course admin staffs to:

- ➢ entry of course and unit details via a graphical user interface

- display the details back on the screen, and to allow data persistence between different executions by using a file.

The assignment description has been broken down into fragments and sections to help convey the sense of what it is we wish you to achieve. You should read through the whole document to understand what the expectations are.

Context/Background information

The School of Science and Technology (SST) offers various courses to students such as Bachelor of Information Technology (BP162), Bachelor Of Robotics & Mechatronics Engineering (BH123), Bachelor Of Electrical & Electronic Engineering (BH073) and Bachelor Of Software Engineering (BH120). Each course's structure consists of a number of units students are allowed to enrol.

Under the BP162 course, there will a list of first year level units (COSC1xxx), a list of second year level units (COSC2xxx) and a list of third year level units (COSC3xxx). Every unit will be offered in first semester, second semester or third semester.

Under the BH120 and BH073 courses, there will a list of first year level units (EEET1xxx), a list of second year level units (EEET2xxx) and a list of third year level units (EEET3xxx). Every unit will be offered in first semester, second semester or third semester.

Under the BH123 course, there will a list of first year level units (OENG1xxx), a list of second year level units (OENG2xxx) and a list of third year level units (OENG3xxx). Every unit will be offered in first semester, second semester or third semester.

Functional requirements of the system

Overview

The key functional requirements to be satisfied by the program you produce for the assignment are as follows:

- The program should provide a graphical user interface to the functionality (Req-1)
- Ability to add new courses to the system (Req-2)
- Ability to add new units to the system (Req-3)
- Ability to assign units to courses (Req-4)
- Ability to assign staff members to courses and units (Req-5)
- Ability to delete courses (Req-6)
- Ability to delete units (Req-7)
- Ability to see a list of units available for the course (Req-8)
- Ability to display the details of a course (Req-9)
- Ability to display the details of a unit (Req-10)
- Ability to remember all information between consecutive runs of the program (Req-11)
- Ability to initialise a list of staff members with some pre-determined data (Req-12)
- Ability to exit the program (req-13)

In the sections which follow, the expectations for each of these will be elaborated.

Req-1: The program should provide a graphical user interface to the functionality

You are to develop a computer program that presents frames with buttons, text fields, etc. to the user, and which responds to events generated by the user interacting with the GUI. It must be possible to perform all main functions described in the other requirements. You may need to refer to the Java API Documentation for details about the methods pertaining to some of the GUI components.

Req-2: Ability to add new courses to the system

The computer program must allow you to create new courses. A course can either be an undergraduate or a postgraduate course. Each course has a course code, a name and a campus location (e.g. either SGS or HN).

One staff member is then assigned as the course director for each course on offer, while another staff member is the deputy course director for the course (See Req-5).

The course code must be 3 digits long and it is unique, i.e., no two courses can have the same code. You may wish to let the program allocate course codes to courses but this is not compulsory. However, if you let user input the code, it is your responsibility to ensure the provided code is unique.

Req-3: Ability to add new units to the system

The computer program must allow you to create new units. Each unit has a unit code and a unit name. The unit code must begin with a valid course shorthand (such as EEET or COSC or OENG) followed by 4 digits. The first digit of the four indicates the year level of the unit – e.g. EEET2001 indicates a second-year level unit. No two units can have the same unit code. Note that the SST only offers units from first year level to third year level.

Every unit is assigned a chief examiner and a lecturer, who are also staff members of the Faculty (See Req-5). It is offered in Semester 1, Semester 2 or Semester 3.

A unit must not be created if any of its information is not supplied, or is invalid. You should ensure that the data entered by the user is complete, before recording the details permanently in the system.

Req-4: Ability to assign units to courses

As mentioned, different units are allocated to a course. For an undergraduate course, only first year, second year and third year level units are permitted.

Req-5: Ability to assign staff members to courses and units

The program must allow the user to designate a staff member as the course director of a course and another staff member as the deputy course director of the same course. They must not be the same staff member taking up these roles.

The program must also allow the user to designate a staff member as a unit's chief examiner and a staff member as the unit's lecturer. The chief examiner and the lecturer can be the same staff member.

You are being provided with a plain-text file (“staffs.txt”), which contains details about the staff members of the School. Upon completion of the assignment, this file should be used to populate the system with staff member objects (See Req-13).

Req-6: Ability to delete courses

There will be times when a course is under review and it may no longer be offered to students. The program must allow for the ability to delete a course from the system.

Req-7: Ability to delete units

Sometimes individual units will not be offered due to staff shortage, so you need to delete their details from the system. Therefore, the program must provide for this as well.

Req-8: Ability to see a list of units available for the course

It must be possible for the program to display a list of all units currently assigned to a selected course of the user’s choosing. The details to display include the unit codes and the corresponding unit names of the unit. You should use a JTextArea, (possibly with a JScrollBar and JScrollPanel), in order to display this information to the user.

Req-9: Ability to display the details of a course

The program should allow the user to choose a particular course and to display the details of the selected course. The details to display include the course code, the course name, the course director and the deputy course director.

Req-10: Ability to display the details of a unit

The program should allow the user to choose a particular unit and to display the details of the selected unit. The details to display include the unit code, the unit name, the chief examiner and the lecturer.

Req-11: Ability to remember all information between consecutive runs of the program

The data which is entered into the program or altered while it is running, that has not been explicitly deleted within the program, needs to be retained for future times that the program is run. To achieve this, you are expected to use object serialization to ensure all objects representing information about courses and units are stored to a file before the program exits. The next time the program is started, you should bring the objects from that file back into memory to re-establish the system’s state as it was prior to exiting.

Req-12: Ability to initialise a list of staff members with some pre-determined data

You are provided with a plain-text file (“staffs.txt”), which contains details about all the staff members of the School. You should use this file to populate the system with staff member objects.

The organisation of the text file is described below. The first line of the file is an integer stating how many ‘staff’ records will immediately follow. This is followed by staff records of the form:

- A line giving the staff id and the staff name, separate by a comma.
- A line giving the address

Your program should be designed to load this file only if it is the 'first' run, i.e. it should not load it after having serialized data from a previous run. The intention is that you would throw-away the file after using it, however you will find that sometimes after re-compiling your source code, it won't like reading the serialized data.

NOTE: The marker may make use of a *different* version of this file to test your program. However, it will conform to the structure explained above.

Req-13: Exiting the program

Needless to say, the program must allow the user to choose when to exit from it. Don't forget to save the data so it can be loaded the next time the program is run.

Additional Assessment Requirements

The following are not necessarily 'functional' requirements, but they are still required to be completed by you as you work on the assignment.

Req-14: NullPointerExceptions and custom exceptions

You should take care to ensure that problem-domain classes do not accept null-values for parameters either in constructors, or normal methods, when these are expecting objects. In such cases, the most appropriate exception is NullPointerEception (to be consistent with the behaviour of classes in the Java Class library).

Additionally, there may be cases where you think it appropriate for problem domain classes to generate some exceptions of a custom-type.

(Do not worry about making exceptions for application domain classes, i.e. your user interface classes.)

Req-15: Must use Map collection class to remember courses and units

As every course is uniquely identified by a course code and every unit is uniquely identified by a unit code, you should use map data structures to store courses and units – the codes (course or unit) will be the keys and the respective objects are the values. Then, in places where you are required to choose a course or a unit, you should allow the user to enter the code in a text field and retrieve the respective value.

Req-16: Use of specific GUI components

You must make use of each of the following somewhere in your program:

- JTextField
- JPanel
- JButton
- JRadioButton
- JComboBox
- JFrame

You are allowed to use other components too, but must ensure the above are used in at least one place.

Req-17: Separation of Business Logic from Graphics UI Logic.

You should design classes such that they can be classified as sitting in one of four domains (foundation, architecture, business or application domain). What this means for your assignment is that the code which represents information should be separate from the code which runs the user interface. Please take care to make separate classes for business-logic and for graphics/UI-logic.

Req-18: Sketch the Screens for your program before implementing them

Draw a sketch of the screens you plan to implement in Java. That way, if you don't complete the coding for it, the marker can know what you were planning to produce. You may be asked to show this to the lecturer before the due date, in class time. This requirement won't affect your grade.

Req-19: Coding requirements

The code must compile and execute successfully in the Java 8 (or later) SDK. Your code files must compile without any error messages if they are to be considered by the marker when deciding on the grade to be awarded. The marker does not have time to debug your code, and the marker might not give feedback comments regarding code which does not compile.

Code-Presentation requirements

Please keep in mind the following things regarding the presentation of code assignments:

- All code should be indented properly to show that you understand the structure of the code.
- Commenting should be present, and relevant. Any style of commenting is permitted, but it should be neither excessive nor too minimal.
- Identifiers will be named appropriately
- Preferably, you will provide an explanation of the purpose of each class, attribute, method, and method parameter, in a manner that is obvious to the reader of your code. These descriptions are allowed to be done using Javadoc style, but this is not mandatory and is not 'worth' more than another style. The preferred style for methods is to use the headings of Name, Purpose, Passed and Returns, maybe also Effect, to describe the method's place and use within the system. *Passed* refers to parameters passed in; *Returns* is the value sent back to the caller.

For example:

```
//Name:    setUpperLimit
//Purpose: to set the highest valid number of greyhounds allowed to run in the race
//Passed:  upperLimit - the maximum number of greyhounds allowed.
// Don't make it too excessive or else the dogs will trample each other
// Returns: true - if the number is acceptable, false otherwise.
// Input:   None is performed
// Output:  None is performed
// Effect:  This 'Race' object is modified so that the maximum number of dogs in
//          the race will be set to a new value.
//          If there are already an excess of dogs added to the race, then an
//          exception will be thrown (DogsAlreadyInRaceException).
public boolean setUpperLimit(int upperLimit)
throws DogsAlreadyInRaceException
{ ... }
```