SEG2105. Introduction to Software Engineering

Assignment 4

Fall 2015

- **Assignment due: November 30 2015 (by midnight).**
- You may work individually or in groups of two.

PART I (40 points)

A. [4x5 marks] **E128** – b,c,d,e (Hint. Delegation is NOT the *dominant* pattern in c and d).
B. [5 marks] **E135 (See Appendix A below)**
C. [5 marks] **E152 -** c
D. [10 marks] **E169**

A. [4x5 marks] **E128** – b,c,d,e (Hint. Delegation is NOT the *dominant* pattern in c and d).

E128 Find the design pattern that would be most appropriate for the following problems:

(b) You want to allow operations on instances of `RegularPolygon` that will distort them such that they are no longer regular polygons. How do you allow the operations without raising exceptions?

(c) Your program manipulates images that take a lot of space in memory. How can you design your program so that images are only in memory when needed, and otherwise can only be found in files?

(d) You have created a subsystem with 25 classes. You know that most other subsystems will only access about 5 methods in this subsystem; how can you simplify the view that the other subsystems have of your subsystem?

(e) You are developing a stock quote framework. Some applications using this framework will want stock quotes to be displayed on a screen when they become available; other applications will want new quotes to trigger certain financial calculations; yet other applications might want both of the above, plus having quotes transmitted wirelessly to a network of pagers. How can you design the framework so that various different pieces of application code can react in their own way to the arrival of new quotes?

B. [5 marks] **E135 (See Appendix A below)**

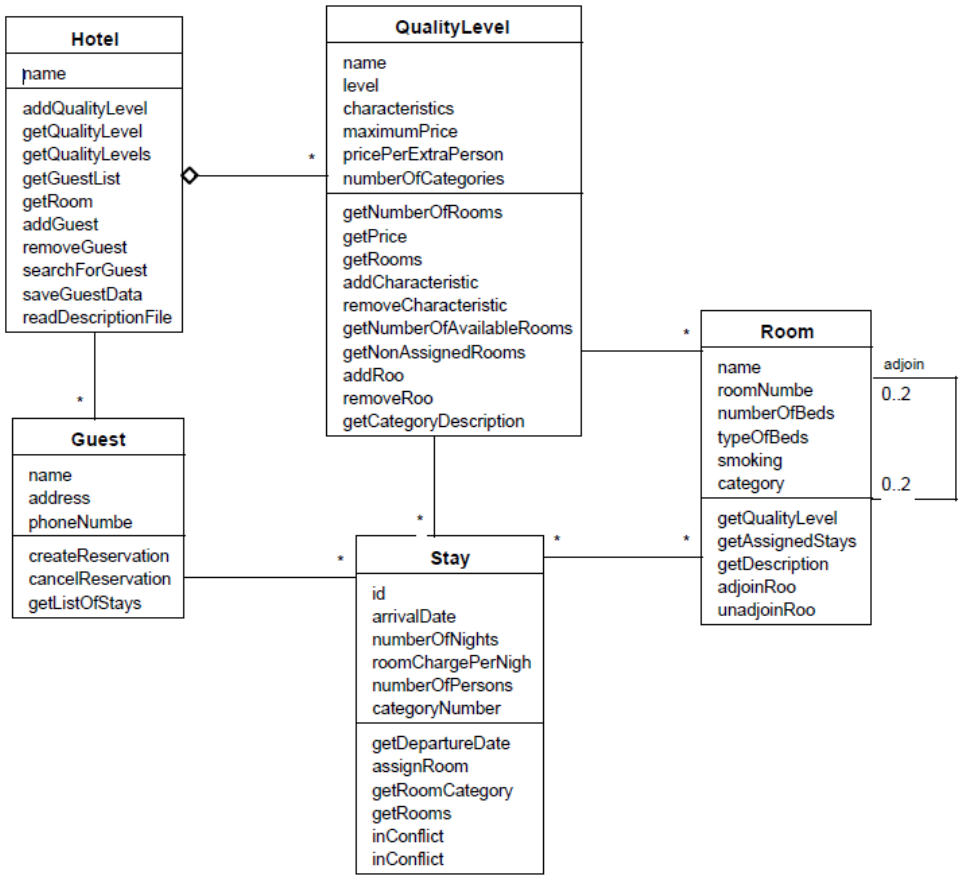
E135 Examine your class diagram for the Small Hotel Reservation System from Chapter 5.

(a) Determine which patterns, if any, you already applied, without knowing it.

(b) See if you can improve your class diagram by applying one or more of the patterns discussed in this chapter.

APPENDIX A - Description of the Hotel System

The Hotel system manages the front-desk activities of the 'Interface Rapids Hotel'. The system will be used to enter reservations as well as to check guests in and out of the hotel. The hotel contains rooms in which guests can stay. Some hotel rooms adjoin others; that is, there are internal doors between them. Each hotel room is assigned a quality level (e.g. a larger room or a room with a view would be better than a smaller room without a view). Each room also has a certain number and type of beds, a room number, and a smoking/non-smoking status. Each quality level has a maximum daily rate, although the rate that a guest pays may be less. When a hotel guest wishes to make a reservation, the hotel clerk asks him or her which nights he or she wants to stay and the type of room he or she wants. The system must verify if room(s) are available on those nights before allowing a reservation to be made. The hotel needs to record basic information about each guest, such as his or her name, address, telephone number, credit card etc. A reservation can be canceled at any time but some fees (a percentage of the room price) may be charged if the cancelation is done too late. When a guest checks in, a room is allocated to him or her until he or she checks out. When the customer requests a specific room, this can be allocated in advance at the discretion of the manager. The system must keep track of the guest's account, and print his or her bill.



C. [5 marks] **E152 - c**

E152 Draw sequence diagrams representing the following interactions :

(c) A client wishes to open a new account at a bank branch. To do so, his instance of class `Client` must first be retrieved from the central bank server. For a new client, an instance of `Client` must be created. An instance of `BankAccount` is then created using the `Client` object. A deposit must then immediately follow, to complete the account creation process.

D. [10 marks] **E169**

E169 Draw detailed state diagrams for

(a) `ConnectionToClient` in the OCSF framework.

(b) A client in Phase 5 of SimpleChat, which can be in various states that have to do with connecting to servers, logging in, forwarding of messages, etc. Hint: you may want to draw separate state diagrams for different aspects of the client's operation.

PART 2 (60 points)

A. [20 marks] **Create a UML class diagram for system described below.** Make sure you include correct multiplicity. Show all attributes and associations plus at least five important operations. If generalizations are necessary, show them too. Marks will be given for effort, even if you don't have a perfect solution. However, marks will be lost for the common types of mistakes we talked about in class (e.g. poor generalizations, wrong multiplicity, etc).

B. [30 marks] Design a **state diagram** describing the states a *Survey* instance can be in. You can define composed Boolean expressions to make your diagram more readable.

C. [10 marks] Improve your class diagram by applying two of the **patterns** discussed in Chapter 6. Describe the changes to be made in your initial class diagram (**only**).

An Online Survey System

You are creating a system that will allow people to create online surveys (questionnaires). There are two types of users: Survey makers and survey takers.

Each survey maker has an account and each account can have any number of surveys associated with it. Surveys go through a series of steps as they are being set up. The survey is first 'being designed'. When the initial design is complete, the survey maker can tell the system that the survey is now 'being tested'. During this step users can be asked to use the survey, but the data is thrown away; changes can be made to the survey at any time. Previous versions of the survey are always kept in case the survey maker makes a mistake and wants to go back to a previous one. After testing the survey changes so that it is 'gathering data'. Finally, after the date gathering is done, they survey is 'closed'. A survey can be reopened at a later date, and a survey (and all its data) can be deleted entirely.

A survey has a series of pages, and each page has an ordered set of items. Pages and items can be used on several different surveys. Some items are just text that explains the survey. The remaining items are questions. Questions all have some text asking the question and then accept answers of various kinds:

- Multiple choice (with the option to allow the user to select just one or many answers).
- Text (with maximum number of characters and lines that can be specified by the survey maker)
- An integer
- Boolean

If a user gives a certain answer to some multiple choice questions, then the survey software can be made to automatically skip to a certain page. So for example, a survey can ask someone if they own a car; if the survey taker says they don't own a car then the survey can skip all questions about the car.

The answers of every survey taker must be saved. On some surveys each survey taker is given a number. They can then come back and modify their answers until a certain date.

The company expects to have many thousands of users, so has many servers around the world. Data for each survey is kept on at least three servers in different places. One place is the master, the others are backups. However, when a user accesses a survey, data is transferred from wherever the master is stored (or from a backup if the master is down).