SWEN 383-01 Exam 2 Topics

Tuesday NOV 21, 2017

Format

- 50 Minutes (entire class)
- Closed book, reference sheet provided with the class diagram of each pattern covered
- Optionally, you may bring one 8 1/2 x 11 notes sheet, both sides OK
- 4 questions, short answer, small design problems

Topics

(Based on class room discussion, project work and resources provided via the course website and myCourses.)

- Design Patterns (only those we have covered in class)
    - Model-View-Controller (MVC)
    - Mediator
    - Façade
    - Proxy
    - Command

Sample Questions

Below are two 10-point questions representative of those you will see on the first exam. It is guaranteed that *at least* one of these questions will be among the ones that comprise the exam. Feel free to work with anyone in the class or to consult any material on the Internet in developing your answers. You may also ask questions of your instructor, but he or she will only answer those which serve simply to clarify what is being requested.

Remember: You are allowed to bring one 8-1/2 inch by 11 inch sheet of paper to the exam; you may put whatever you want on both sides of the paper. Your instructor may ask you to turn in your sheet at the end of the exam. You will also be given a handout with pattern descriptions (including some that have not been covered and thus are not applicable to this exam).

If you need any special accommodations for the exam, please make me aware of this via email and provide me with all the materials I need by class on Thursday, Nov 16, 2017.

Q1 Proxy Pattern (10 pts)

1. The **Proxy** pattern and **Adapter** pattern both have a similar structure (class diagram). Briefly describe how the patterns differ.

2. Consider a class **Customer** that invokes a method **booking()** in another class **Reservation**. These classes are not located in the same physical machine, so the **Customer** invocation of the **booking()** method must be done remotely (i.e. across the network). We do not want our design to require the **Customer** class to implement the network messaging required to access the **Reservation** class in the remote location where it resides. Provide a class diagram showing how we could incorporate a ***remote proxy*** to achieve the goal of the **Customer** class not having to implement network messaging when invoking the **booking()** method.

Q2 Mediator Pattern (10 pts)

Consider the design of a multi-player, interactive, real-time, video network game. In this game, each player can (a) view his or her immediate surroundings in the game's virtual world, (b) see summary information on the opposing players in the game, and (c) make moves in the game based on this knowledge. If an opposing player enters the field of view, this will be shown visually on the player's screen. Players can join or depart at any time during the game.

As players join, depart, and make moves in the game, the resulting game information must be updated for all current players. Your job is to design the coordination portion of the game.

How might the **Mediator** pattern be applied to this problem? Provide a class diagram showing the structure and possible flow of information among the various classes. You may select your own class names.