Tutorial Sheet – The Observer Pattern/Java Event Model

The purpose of this tutorial is to review your understanding of the **Observer Pattern** and **the Java Event Model** by modifying the *Prof-Student-TA example* from earlier.

Right now, the Prof maintains a list of Students and a reference to the TeachingAssistant. When a midterm is set or postponed, the Prof must call the appropriate method on each type of object. For example, when the midterm is set the study(...) method is called on the Students and the proctor(...) method is called on the TeachingAssistant. If we add more people that are interested in the midterms (CourseAuditors, DepartmentStaff, etc.) the Prof will need to maintain more lists and keep track of what methods need to be called on each object type

Observer Pattern

Use the Java Observer and Observable APIs to help you with the following.

1. To start off, one class will need to extend Observable and the other two classes will need to implement Observer. Modify your code with these changes.
2. Modify the main method in Prof. Instead of having separate method to add Students and TeachingAssistants, modify the code to use the built-in Observable function for adding an Observer.
3. There are two areas where the Prof can be changed: setMidterm(...) and postponeMidterm(...). First the Prof needs to indicate that it has changed in some way, and then it needs to inform the Observers that it has changed. Modify your code to do this using the appropriate Observable methods.
4. Finally, the Student and TeachingAssistant classes need to implement the update(...) method. Implement this method in each class. Keep in mind that this method will be called when the midterm is set and when it is postponed, so you will need to know which happened so you can perform the appropriate behaviour (study/proctor or party/postpone).
5. Verify your code outputs similar text to the console as the original Prof-Student-TA example.

Java Event Model

1. Start again from the original Prof-Student-TA example.
2. Modify the code to use the Java Event Model. Unlike with the Observer Pattern, no detailed instructions will be given. Instead, you can use the lecture notes to walk yourself through the process of changing the code to the Java Event Model.
3. Again, verify your output is similar to the original example and the Observer example above.

Capture Your Design

Let's now capture the design decisions that you have made in **UML Class and Sequence Diagrams**. For this, you can use the Violet tool that is available at:

http://www.horstmann.com/violet/

Build a class and sequence diagram for each of the three implementations:

- Original Prof-Student-TA
- Observer pattern Prof-Student-TA
- Java Event Model Prof-Student-TA