Unit 4 Design Activity

"In the situation where I have to design new pieces of code for an existing project RIT has prepared me well. I am constantly creating classes under the guidelines of design patterns, low coupling, high cohesion, etc. I find that I write effective and maintainable code. However, in the situation where I have to maintain existing code, where no features are added, then I wasn't prepared well. I have been learning about refactoring on my own. It would be nice if RIT had a class about what to do in the situation where you work with old code."

The comment above was written by a student who graduated from RIT's Software Engineering program's first class in May 2001. It was in response to a survey question: "Generally, how well do you feel that RIT's Software Engineering program prepared you for your current job responsibilities?" His comment is directly applicable to the material in this course, Engineering of Software Subsystems.

The design activity for this unit is to analyze an existing program. Your team will reverse engineer the design and discover the intentional (and unintentional) uses of design patterns in the program. You will provide an analysis of the existing design describing what you perceive its strengths and weaknesses to be. You will also analyse the code base using an Eclipse metrics plug-in (metrics.sourceforge.net) available on our systems. After doing the analysis of the existing code base you will suggest a refactoring of the design to better adhere to the design principles that you worked with in this course. Principles such as coupling, cohesion, delegation vs. inheritance, assignment of responsibilities, elimination of *bad code smells*, metric values, etc. should be in your consideration when doing the analysis and refactoring. The refactored design should have some connection to the original design. The intention is not to throw away the design and start with a clean sheet of paper. As a minimum you will need to understand the current design so that you can present your analysis of it.

After doing the refactoring of the design the team will select some significant aspects of the refactoring to implement. The aspects of the design that will be implemented must be approved by your instructor by the date specified on the course schedule.

On the day scheduled for completion of this unit each team will do a presentation of their analysis of the existing design including its strengths and weaknesses in the dimensions of the design principles discussed in this course, its use of design patterns, analysis of the metrics for the original system and the team's suggested refactoring to improve the design. The design should be presented using UML diagrams. To highlight the improvements you made by refactoring the design, show sequence charts for, at least, two important operations executing in the original design and in your refactored design. Discuss the metrics report for the refactored design. In your presentation do not discuss every metric. Select the ones that are most important and relevant and that guided your refactoring.

The system that you are to work with is found in a zip file in this unit. It is a previous student team submission for SE361. The zip file includes a statement of the problem that this software system solves, all available documentation, and the code base.

Assessment

Component	Percent
Presentation	15
Reverse Engineering, Refactoring and Documentation	70
Implementation	15

Submission Instructions

1. A zip file named **`p362-unit4project-X.zip`** where ***X*** is your team number. The zip file contains:
    a. All the Java source files required to compile, link, and run your project.
    b. The final single design document should be contained in one Word file. It should contain the information specified below. ***Note it would not be an effective presentation of your design to take each item below, stack them one after the other and staple it together. You need to weave this information through your document in a manner that tells a cohesive story with prose guiding the reader and tying the sections together.***
        i. Title information, including the name of your team, the name of the project, the date, and a list of all the team members.
        ii. A short overview section describing the product and the features included.
        iii. One or more UML class diagrams showing the main classes and interfaces in your design, along with inheritance (generalization), association, aggregation, and composition relationships. Include cardinality and role indicators as you deem appropriate to make the diagram clear. Indicate the role a class plays in any patterns in which it participates. DO NOT include state or method information. You will need several class diagrams at different levels of abstraction and for different subsystems to completely document your design in a way that the reader can physically see and intelligently understand.
        iv. One or more other UML diagrams (e.g., sequence charts, collaboration diagrams, state diagrams) to provide insight into the key static and dynamic characteristics of the program both before and after refactoring.
        v. A table summarizing the responsibility(ies) of each major class.
        vi. A narrative analyzing the original design (note: the "original" design is what is in the code you reverse engineered), its weaknesses and strengths, fidelity to the design documentation, and use of design patterns.

vii. A narrative outlining how the refactored design reflects a balance among competing criteria such as low coupling, high cohesion, separation of concerns, information hiding, the Law of Demeter, extensibility, reusability, etc. This should include a discussion of the design patterns used to achieve this balance.

viii. A short reflection on the process of discovering the design in an existing software system.

ix. Discussion of your metrics analysis of up to four metric values including answers to the following questions:

A. What were the metrics for the original code base? What did these initial measurements tell you about the system.

B. How did you use these measurements to guide your refactoring?

C. What are the metrics for the refactored code base? This only has relevance for the areas of the refactored design that you implemented.

D. How did your refactoring effect the metrics? Did your refactoring improve the metrics? In all areas? In some areas? What contributed to these results?

c. A copy of the final PowerPoint presentation.

d. A file **README.txt** that includes any other information the team wishes the instructor to know. This must include instructions for building and running the program.

2. Upload the zip file to the *myCourses* site for this course and section, placing it in the **Unit 4 Refactoring Project** dropbox. The title of the submission should be "Unit 4 Project for Team *X*", where *X* your team number.

3. The zip file must be submitted by the end of the day on which the presentations for this unit are made. Late submissions will **not** be accepted; teams are advised to submit *something* early to ensure they receive at least minimal credit.

4. ***Presentation: hand in the printed slides before presentation.***