Unit 1 (Refactoring) project – Bowling Alley Simulation

(Due 20th February, 5:00 pm)

In this project you will analyze and refactor an existing software system. Your team will reverse engineer the design from existing code and documentation (if any); and propose refactoring to improve the program's structure for future maintenance and evolution. Second year undergraduate students at RIT implemented the software system.

The code can be analyzed using a metrics plug-in (metrics.sourceforge.net). Metrics plug-in is a very old piece of software and is listed here only as a reference. Feel free to use any metrics analysis system that you are aware of and/or is more recent. Principles such as coupling, cohesion, delegation vs. inheritance, assignment of responsibilities, elimination of bad code smells, metric values, etc. should be in your consideration while choosing the metrics plug-in for doing the analysis and refactoring. Note that you are not expected to throw away the original design and start with a clean sheet of paper. The new implementation should not have any behavioral change.

Each team should document their analysis of the existing design including its strengths and weaknesses in the dimensions of the design principles mentioned in the previous paragraph, and the suggested refactoring to improve the design. The refactorings must be documented in a tabular format. The refactored design should be represented using UML diagrams (class and sequence diagrams). To highlight the improvements you made by refactoring the design, show sequence diagrams for two important operations as is in the original design and in your refactored design.

It's important to analyze the system pre- and post- refactoring to provide the appropriate metric values.

NOTE: The system was developed in an earlier version of Java that did not support templates; your refactoring should *not* focus on this aspect of the original system.

All your work must be regularly updated in your version control repository!!!. In addition, the following submission instructions must be adhered to for the final submission.

Submission Instructions

1. Create a zip file named **TeamMembers_Unit1.zip** (where ***TeamMembers*** are your team members first names) containing three directories name src (for source code), doc (for the major document), misc (for all miscellaneous documents)
    - **a.** The src directory should contain all the source code (appropriately packaged – if necessary)
    - **b.** The doc directory should contain a single design document in either MS Word or PDF format. It should contain the information specified below.

Note: It would not be an effective presentation of your design to take each item below as Q & A, 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 the project, the date of submission, a list of all the team members with names and roll numbers, effort (number of hours) put in by each team member, role played by each team member.

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. You may decide on the appropriate level of abstraction with respect to state or method information. You may 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 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. Discussion of your metrics analysis of up to eight metric values including answers to the following questions:

1. What were the metrics for the code base? What did these initial measurements tell you about the system.
2. How did you use these measurements to guide your refactoring?
3. How did your refactoring affect the metrics? Did your refactoring improve the metrics? In all areas? In some areas? What contributed to these results?

c. In the misc. document, you may put in all the miscellaneous management related documents. For example, task trackers, meeting minutes, etc.

2. Upload the zip file to U1 Project submission folder in Moodle. One submission per group will suffice.