DASS Assignment - 3
(Team assignment)
Bowling Game

Deadline: 2nd April 2020 (11:55 pm)

Assignment Setters: Mohit, Gunjan, Vijayraj, Eesha

Bowling is a fun sport, especially when played in a large group. But, with the current situation of COVID-19, it will be safe if we can enjoy the game right on our computers! Don't worry we are not asking you to build a game rather in this assignment you will refine an already built game :p

For this assignment, you will analyze and refactor the design and implementation of an existing software system and implement some new cool features. Your team will reverse engineer the design from existing code and propose refactoring to improve the program's structure for future maintenance and evolution.

For the first part of the assignment, we will look into code analysis and refactoring, we will use the principles such as coupling, cohesion, delegation vs. inheritance, assignment of responsibilities, elimination of bad code smells. 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. 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 presented using UML diagrams (class and sequence diagrams). To highlight the improvements you made by refactoring the design, show sequence diagrams for two important operations executing in the original design and in your refactored design.

The second part of this assignment includes the implementation of some new cool features which will make this game even more fun to play :D. So, here are the set of features we want the teams to implement:

1. Make the code extensible and working for multi-player, let the maximum number of players be 6. Provide an option to add and store players names
2. Add a database layer to implement the persistence of the scores and players. Provide a searchable view to make ad-hoc queries on the stored data. Some possible queries - highest/lowest scores, Top Player, etc.
3. Implement pause and resume features in the game. The players should be able to pause the game and continue the game from the point where they left even after closing the game. You will have to provide an additional option of continuing an existing game to implement this feature.

Submission Instructions

1. Create a zip file named TeamNumber..zip (where team members are your team members first names) containing three directories name src (for source code), doc (for the report), misc (for all miscellaneous documents)
    a. The src directory should contain all the source code after refactoring and new implementations (appropriately packaged – if necessary)
    b. **The doc directory should contain a design document in 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, 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, 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. 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.
vii. A table containing all code smells found along with their short description (eg: which file contains it, what is the exact problem etc).
viii. 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 what was done to achieve this balance.
ix. Discussion of your metrics analysis of up to eight metric values including answers to the following questions:
    1. What were the metrics for the codebase? 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. There should be an additional document describing the newly implemented features (**name the doc as New_Features in PDF format**) which will contain the following:
    i. The UML class diagrams and UML sequence diagrams for the newly implemented features only.
    ii. Descriptions of the new classes and functions implemented.

Note: The teams will be awarded 0 marks for indulging in any sort of plagiarism

Helpful Information

1. You can analyze the codebase using a metrics plug-in (metrics.sourceforge.net). Feel free to use a metrics analysis system that might be more recent.
2. The codebase is in Java, so we highly recommend to use an IDE such as Eclipse.
3. The class diagrams should be created from scratch and not with the help of any tool. Draw.io is an excellent tool for making UML diagrams from scratch.