# DASS ASSIGNMENT 2

## Code Review and Refactoring

Deadline: 12th March 2023
Concerned TAs: Swetha Vipparla, Siddharth Mavani

**Total Marks: 50**
**Bonus Marks: 20**

In the previous assignments, you created software systems from scratch, utilizing a variety of technologies and frameworks as well as adhering to sound coding practices to produce expandable code. We'll now examine code reading and analysis, which is a useful talent to have in addition to writing code.

You will examine and refactor the design of an existing software system for this assignment. In order to improve the code structure for upcoming maintenance and evolution, your team will examine and analyze the provided codebase, reverse engineer the design, and make refactoring suggestions.

Please find the assigned zip file for your team here.
The codebase you will have to work on will be $< your\ team\ number \ \%\ 33\ +\ 1 >$.zip.

## Part 1: Code Review

For this part of the assignment, you will analyze the code to understand the existing design including its strengths and weaknesses. The design can be presented using UML diagrams. You will also find and report all bugs and code smells in the code.

References:
1. https://blog.codinghorror.com/code-smells/
2. https://en.wikipedia.org/wiki/Code_smell

## Part 2: Refactoring

Now that you have analyzed the existing codebase and its design, you will suggest how you would improve the design of the code by using the various good practices taught in class. Use OOPs concepts, Design Patterns, elimination of code smells etc for this purpose.

**Note: You do not need to refactor the code; only suggest improvements.**

## Submission format

Every team has to submit a document <TeamNumber>.pdf containing the following sections in the mentioned order:

### 1. Basic Information

Title information, including team number, team members, roll numbers etc.
Mention clearly the contribution made by each member of the team.

### 2. Overview

A short overview section describing the software system you will study and its features.
*(5 Marks)*

### 3. UML Class Diagrams and Summary of Classes

a. Show the main classes and interfaces in your UML class design, along with inheritance (generalization), association, aggregation, and composition relationships for the original design. 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.
*(10 Marks)*

b. A table summarizing the responsibilities of the major classes.
*(5 Marks)*

### 4. Code Smells

A table containing all code smells found along with their short description and suggested refactoring method (eg: which file contains it, what is the exact problem etc).
*(15 Marks)*

| Code smell | Description of Code Smell | Suggested refactoring |
|---|---|---|
| | | |

Include other columns as needed.

## 5. Bugs

A table containing all bugs found along with their short description and suggested refactoring method.
*(15 Marks)*

| Bug | Description of Bug | Suggested Refactoring |
|---|---|---|
| | | |

Include other columns as needed.

## BONUS

1. Implement the suggested refactoring for 3 of the code smells. Include code snippets in your document and explain your modifications/additions. We will be running your refactored code to ensure the functionality is maintained. Attach the codebase along with the assignment pdf and submit it as a zip.
   *(15 Marks; 5 * 3 refactorings)*

   Refactoring trivial code smells will not fetch you any bonus marks. Following are examples of this:

   - Simply changing function names to remove inconsistent naming conventions.
     Eg: Changing MoVePaddLe() to movePaddle()
   - Adding comments to eliminate lack of documentation.
   - Removing unused code.

2. Automatic refactoring is a topic of great interest in software engineering research. As the name suggests, it is the process of automatically refactoring code according to certain constraints and requirements. Why not try our hand at this?

   Propose an approach for automating the process of refactoring code according to the requirements of this project (i.e addressing design smells with consideration for code metrics). While you don't need to implement this, you need to describe your solution in detail. You will be graded on the clarity and feasibility of your solution. Make reasonable assumptions, but feel free to think outside the box. And note that simple is fine but frivolous is not.
   *(5 Marks)*