**Code Review of the Team Project**

In this exercise, each team will apply code review practice to their team project, aiming to find as many problems as possible. There is no need to test the program or make changes during code review. The findings should be used to improve your project after the review. In addition to looking for defects and bugs, you should check: (1) whether the entire project has followed the coding standard in a consistent manner, (2) whether the project has followed the design principles introduced in class, and (3) whether there are code smells that indicate the need for refactoring. The following checklists provide basic guidelines. You may add new items to each of the checklists.

Checklist #1: Coding Standard

* Are there any violations of naming conventions?
  + Packages, classes, methods, variables, constants
  + Production code vs test code?
* Is the ordering convention of method arguments followed in each method?
* Are all comments meaningful and valid?
  + Are the precondition and post-condition of each method documented?
* Is the same style (e.g., on the same line vs a separate line) used for all curly braces of code blocks?
* …

Checklist #2: Design Principles

* Does each class have a good abstraction and good class interface?
* Is the visibility of each variable, method, and class (private, protected, public, default) appropriate?
* Is the command-query separation principle violated?
* Operand principle for method arguments: any method arguments are options?
* Design by contract: whenever it is used, is the precondition is specified, reasonable, and available?
* Defensive programming: whenever it is used, is it necessary and reasonable?
* Is any of the SOLID principles violated?
* …

Checklist #3: Code Smells

* Is there any magic number or unnamed constant?
* Is there any unnecessary global variable?
* Is there duplicate code?
* Are there long methods?
* Does any method have a long parameter list?
* Is there any over-complex expression?
* Is there any class that is essentially a data record?
* Is there any switch or if-then-else statement that needs to be replaced with polymorphism?
* Is there any variable or method name whose intent is unclear?
* Are there similar methods in multiple classes?
* …

For each team, one member should lead the code review, one member should take notes and record findings from the review, and one member should be responsible for the checklists. The leader of the presentation should first give a brief introduction to the program structure (classes and their relationships) and introduce the team’s coding standard, then present the source code of each class under review. Probably you won’t have enough time to review all classes, thus you need to determine which classes will be reviewed in advance.

After the exercise, each team should email the findings of code review to the instructor. The findings for each class should be summarized as follows:

***Class name***

|  |  |  |  |
| --- | --- | --- | --- |
| ***Checklist*** | ***Checklist item*** | ***Finding*** | ***Notes*** |
| Coding standard | Naming conventions |  |  |
| … |  |  |
| Design principles | Good class interface |  |  |
| … |  |  |
| Code smells | Magic numbers |  |  |
| … |  |  |
| Defects and bugs | Buggy code snippet | What is the bug | Why it is a bug |
|  |  |  |