ENSF 409 – Principles of Software Development
Summer 2021

Lab Assignment #6: Software Engineering Best Practices

Due Dates
Submit electronically on D2L **before 1:00 PM on Friday July 30th**

The objectives of this lab are to gain experience with and understand the following concepts:

1. Code Review
2. SOLID Principles


The following rules apply to this lab and all other lab assignments in future:

1. Before submitting your lab reports, take a moment to make sure that you are handing in all the material that is required. If you forget to hand something in, that is your fault; you can't use `I forgot' as an excuse to hand in parts of the assignment late.
2. **20% marks** will be deducted from the assignments handed in up to **24 hours** after each due date. It means if your mark is X out of Y, you will only gain 0.8 times X. There will be no credit for assignments turned in later than 24 hours after the due dates; they will be returned unmarked. Exceptions can be made but the students must inform that Instructor beforehand to accommodate if acceptable.


Exercise - 1: Code Review (20 Marks)

Code review is an important practice adopted by developers to ensure the quality of the code, as well as the growth of the developers involved. It can be an expensive process due the amount of time it requires, but the number of companies that employ this technique clearly indicates that the benefits of conducting code reviews clearly outweigh the cost.

Therefore, it is important for software engineering students to become familiar with this practice. Please see some good resource:

https://smartbear.com/learn/code-review/best-practices-for-peer-code-review/

https://courses.cs.washington.edu/courses/cse403/13sp/lectures/10-codereviews.pdf

You are strongly encouraged to go beyond the provided resources and do additional reading on code reviews and software engineering best practices.

What to do:

Each student must:

- Conduct a code review for another student
- Subject his/her code to a review by another student

Students can choose to have **ONE** of the following 2 programs reviewed:

- The code for the tool shop developed in assignment 3
- The code developed for post-lab 4, exercise 3 (tic-tac-toe)

Please consider the following in your code review:

- Each review should take about 30 to 45 minutes.
- The review must review between 200 to 400 lines of code (but no more!).
- The reviewer must be careful to point out different **code smells** as discussed in class including:
  - Issues related to naming and naming conventions
  - Spelling mistakes
  - Violation of SOLID principles
  - Issues with “overly complicated” code
  - Etc.

What to Hand in:

- **Reviewers** are to download the code review template from D2L and document their review. OnlyReviewers submit the template (convert to PDF before submitting).

- **Developers** submit a reflection: list the most important lessons learned from your code review. Elaborate on each item briefly (Convert to PDF before submitting).


Exercise - 2: SOLID Principles (35 Marks)

Task 1) (10 marks) Consider the following classes in answering parts a and b.

Part a (3 marks) Which SOLID principle is this program violating? Briefly explain.

Part b (7 marks) If needed, re-write each class **completely** to remove the code smell in question. If not, simply write **no change needed** for a class.

Write any classes or interfaces you are adding to this program in this space:

```
abstract public class Vehicle {
        abstract public String getVehicleType ();
        abstract public String getEngineType ();
}
```

```
//If this class needs to change, rewrite it
//completely. If not, write: "No change needed".
```



public class Car **extends** Vehicle{ **private** String vehicleType; **private** String engineType; Car(String vType, String eType) {vehicleType = vType; engineType = eType; } @Override **public** String getVehicleType() { **return** vehicleType; } @Override **public** String getEngineType() { **return** engineType; } }	**//If this class needs to change, rewrite it** **//completely. If not, write: "*No change needed*".**
public class Bicycle **extends** Vehicle{ **private** String vehicleType; Bicycle(String vType) { vehicleType = vType; } @Override **public** String getVehicleType() { **return** vehicleType; } @Override **public** String getEngineType() { **return null**; } }	**//If this class needs to change, rewrite it** **//completely. If not, write: "*No change needed*".**

Task 2) (15 marks) Consider the following classes in answering parts a, b, and c.

public class GraphicCreator { **public void** drawShape(Shape s) { **if** (s.getShapeType() == 1) drawSquare((Square) s); **else if** (s.getShapeType() == 2) drawCircle((Circle) s); } **public void** drawCircle(Circle c) {//Some code. } **public void** drawSquare(Square s) {//Some code. } }	**public class** Shape { **private int** shapeType; Shape (**int** type){shapeType = type;} **public int** getShapeType() { **return** shapeType; } }
public class Circle **extends** Shape{Circle (){**super**(2);} }	**public class** Square **extends** Shape{Square (){**super**(1);} }


Part a (5 marks) Draw the class diagram for the classes above. Make sure to include all fields and methods in yourclass diagram.

Part b (4 marks) Which SOLID principle is this program violating? Briefly explain.


Part c (6 marks) Draw a class diagram for a proper design that removes the code smell in the above program.Clearly include all fields and methods in your class diagram.

What to hand in: Please submit your solutions in a pdf file for tasks 1 and 2.