CPSC 501: Advanced Programming Techniques
Assignment 1: Refactoring

The focus of this assignment is the practical application of refactoring to some **existing object-oriented code**. **Find or create a running program** that is poorly structured and can be improved by refactoring. ***Ideally, this will be code that you yourself have written in the past.*** This code must have a minimal level of complexity. **The project must consist of at least 5 classes** that are coupled together to make the program function, and there **must contain an inheritance structure** consisting of multiple classes.

Within this project find locations that would benefit from refactoring to perform at least **five refactorings**, where each of the refactorings is **different from the others**, and where at **least two of the refactorings involve substantial changes to the internal design of the system**. Since the course has used Java examples and the JUnit testing framework, it would be best if the program to refactor was written in the Java language, but a program written in another object-oriented language is possible, provided that you can complete all requirements of the assignment (in particular, that you use a framework to do unit testing).

Following the principles outlined in the course, apply at least five refactorings to your code.

You must use version control (most likely GIT, but other version control systems are possible like SVN) **to keep track of all the refactorings you do**. Make sure you document and commit each refactoring using a meaningful message in the version control system, and make sure you can extract any version of your code from the repository.

You must also do unit testing as you do the refactoring. It is likely you will add or modify tests as you refactor your code into smaller methods. **The testing code must also be kept under version control.**

Type a brief digital formal report that describes how you did your refactoring. Due to the requirements of this report it is expected that the report will take 2.5 pages at least (at least a half page per refactoring and possibly up to a full page or more). This should explain what you did for **EACH** refactoring, answering the following questions:

1. What code in which files was altered. (don't include the full class files, only the parts relevant to the refactoring).
2. What needed to be improved? That is, what "bad code smell" was detected? Use the terminology found in Chapter 3 of the Fowler text available from the course website/D2L.
3. What refactoring was applied? What steps did you follow? Use the terminology and mechanics outlined in the Fowler text and illustrate the process with well-chosen code fragments taken from particular versions of your system.
4. What code in what files was the result of the refactoring.
5. How was the code tested?
6. Why is the code better structured after the refactoring? Does the result of the refactoring suggest or enable further refactorings?

- Use SHA (GIT) or version numbers (SVN) to cross-reference your code as you describe each refactoring.

1. Your submission should include complete listings of the first and last versions of your program, as well as a complete listing of the final version of your test code.
2. In your submission, to show that you have used version control in a comprehensive manner throughout the process, also print out the log (history) for each file (i.e. use the "git log" command) altered.

Bonus: In-class Presentations

Prepare a 5-minute presentation that explains the refactorings you applied to your code and include it in your submission. Since this is not much time, you might want to highlight the more interesting changes you made. Effective presentations will most likely use presentation materials like handouts, transparencies, or PowerPoint slides.

Presentations will be done in the tutorials. Because time is limited, it is possible not everyone will get a chance to present his or her work. You are strongly encouraged to volunteer to do a presentation (it's like the code reviews that are often done in industry). If, on the presentation day, there are not enough volunteers, the TA will pick students at random to do a presentation, so be prepared for this. Anyone who submits a satisfactory presentation will get bonus marks. Unsatisfactory presentation submission will receive some or no marks at all.

Submit the following using the Assignment 1 Dropbox in D2L:

1. An electronic copy of your report.
2. An electronic copy of the first and last versions of your refactored code.
3. An electronic copy of the final version of your unit test code.
4. An electronic copy of your version control logs. (alternatively making your repository available to your TA so that they can view your version control history in UofC GitLab, GitHub, etc. through web view is also allowed and even recommended)
5. (BONUS) 5-minute presentation of the refactorings you made.

Advanced Programming Techniques
Assignment 1 Grading

Student:_____

Version control (including log files)	10	_____		
Unit Tests	10	_____		
Refactorings (minimum of 5)	10	_____		
Report	10	_____		
Total	**40**	_____		**%**
Bonus: In-class presentation			10%	**%**
Total				**%**
Letter				_____

Letter	**A+**	**A**	**A-**	**B+**	**B**	**B-**	**C+**	**C**	**C-**	**D+**	**D**	**F**
Min. Perc.	**95%**	**90%**	**85%**	**80%**	**75%**	**70%**	**65%**	**60%**	**55%**	**50%**	**45%**	**Below 45%**