CSCI 2134 Lab 9: Class-Level Refactoring

Winter 2020

Due Date: Thursday, April 2, 2020 at 11:59pm, submitted via Git

Objective

In this lab, you will work on your own or with another student to identify possible class-level refactoring opportunities in some existing code.

Preparation

1. Ensure that you have your Integrated Development Editor (IDE) installed.
2. Ensure that you are able to write and run a JUnit test case in the IDE. You will not have the time to both debug your IDE environment and complete the lab task within the lab time. The lab time should be able to concentrate on the lab task.
3. Clone the Lab 9 repo: https://git.cs.dal.ca/courses/2020-winter/csci-2134/lab9/????.git where ???? is your CSID.
4. Review the documentation and provided code. These files are located in the **`src`** and **`docs`** directories of the cloned project. Note: This is the same code that you are extending in Assignment 4.

Resources

- Code specifications in the **`docs`** directory of the Lab 9 repository.
- Code base to be refactored is in the **`src`** directory of the Lab 9 repository.
- Unit tests to be used for regression testing in **`test`** directory of Lab 9 repository.
- Lecture notes on class-level refactoring from class (lectures 17, 18, 19).

Procedure

Set-up

1. Open the project you created in preparation for this lab
2. Open the **`refactor.txt`** file in **`docs`** and fill in your name and your partner's name (if there is one).

Part 1: Identify Refactoring Opportunities

1. Ensure that the JUnit tests are passing.
2. Based on your knowledge of the source code, which you have now seen for the past few labs and assignments, identify code smells that indicate opportunities for class-level refactoring. **Note: You should identify at least 3 different class-level refactoring opportunities in total (there are many).** Most opportunities will span multiple files. For example, there is duplicate code, which has been extracted into the superclass as an example of what you are supposed to do.
3. Document the refactoring opportunity in the file **`docs/refactor.txt`** you will need to include: (see example in the file)

- Name of affected files/classes
- Locations (methods and/or line numbers) of the affected issues
- Description of the identified issue
- Description of the proposed refactoring

4. **Commit and push to the repository. Be sure that the refactori.txt file is committed and pushed.**
5. Repeat steps 2-4 until the requisite number of refactoring opportunities have been documented.

Part 2: Perform a Class-Level Refactoring

6. Ensure that the JUnit tests are passing.
7. Select a refactoring opportunity from the list you created in **`docs/refactor.txt`**.
8. Document (in **`docs/refactor.txt`**) which refactoring you will perform.
9. Perform the refactoring.
10. Rerun the tests to ensure you did not break the code. (Fix it if you do.)
11. If the tests are not passing and you do not have time to fix them, note this in **`docs/refactor.txt.`**
12. **Commit and push to the repository. Be sure that both the code and the refactori.txt file is being committed and pushed.**

Part Bonus: Perform a Second Class-Level Refactoring

13. For a bonus mark, complete another class refactoring selected from your list and document it in **`docs/refactor.txt`**.
14. **Commit and push to the repository. Be sure that both the code and the refactori.txt file is being committed and pushed.**

Note: Only one of the partners needs to submit.

Grading

The lab will be marked out of 4 points:

Task	2 Points	1 Point	0 Points
Identify refactoring Opportunities	At least 3 refactoring opportunities were identified	At least 2 refactoring opportunities were identified	Less than 2 refactoring opportunities identified
Perform a refactoring	A refactoring was performed successfully and all tests pass	A refactoring was attempted, but some tests are broken	No significant attempt to perform a refactoring
Bonus Refactoring	N/A	A second refactoring was performed successfully and all tests pass	A second refactoring was not performed successfully