## Refactor 1: Commit 43eb08c53171b041dd9805e59aee71948da737b2

The first refactoring is renaming two duplicate classes from two separate subclasses. The classes are called “setID(String userEmail)”, from classes StudentWindow and FacultyWindow. I use the simple **Rename Method** as the name of the method does not actually reveal its purpose. While the method is USED to set the user’s ID, the actual method itself only **gets** the ID from the text database.

Using IntelliJ, I use the IDE to find all instances of the method and where it is used. Luckily, it is only used in the constructor of the class to set the user’s email, and as such I make sure to change the called method name as well for both student and faculty window. I made sure to compile and test the two classes separately before and after the changes with Junit. For the change, for testing, since I call it several times, when I changed the actual method name, I used IntelliJ’s refactor Rename function to change all instances as well to help save time.

The method was tested by feeding it different user emails, both that were valid and invalid to ensure the exact same outputs. Hence, I used two valid emails from the database, then 3 invalid emails: “invalidemail”, empty string, and “123”. All of the tests were completed successfully. The code is better structured afterwards due to the fact that someone reading the method name would easily be able to tell what the method does, rather than having to read through the whole method to find out that the method actually **gets** the user’s ID, not **sets** it.

Changed files: StudentWindow.java, FacultyWindow.java

Added files: StudentWindowTest.java, FacultyWindowTest.java, PatronWindowTest.java (Used as a test suite)

## Refactor 2:

The second refactoring was to remove methods from both the **StudentWindow** and the **FacultyWindow** classes and store the four methods into the **Utilities** class. The four methods were: **getBooks(String), getBookName(String), getDate(String)** and **listBooks(String)**. There was a lot of duplicate methods in both classes, 6 in total. Of them, these four were the most logical to move to a different class. I used the refactoring method **Move Method**, as these methods are technically dealing with the database and therefore should be in the utilities class. I left a reference in the original classes to call the new ones with the parameters, plus the student/faculty’s email if needed. As such, I was able to remove around 100 lines of code in total for duplicated code.

The process of refactoring was first identifying where the methods were being used. As they were only being used in its class, it was pretty simple. For the sake of being able to test, I left the same named methods inside the Students and Faculty Window classes. I then ensured that all of the references were updated; for example, in some methods, I would call Utilities.xxx. This then just became xxx, as it was moved into the Utilities class. Information about the user that the methods needed were just passed as parameters.

As a result, the Utilities class has four new methods, while Students and Faculty Window have lost four, while leaving some referencing middle man methods. I tested the code by testing initially getBookName(), which was the first method I was planning on removing. However, after seeing where it was used, ListBooks(), I decided to just test the two larger methods, getDate and listBooks to ensure functionality. I created instances of the class and called the methods with the necessary parameters.

The code is better structured now since I removed a ton of duplicated code, that could’ve been placed elsewhere. The result of the refactoring enables further refactorings if I want to clean up the two window classes even further.

Changed files: StudentWindow.java, FacultyWindow.java, Utilities.java

Changed Junit files: StudentWindowTest.java, FacultyWindowTest.java