Assignment 3: Mocking, Code Smell, and Refactoring

Assignment Instructions

Prerequisites: To complete this assignment, you will need experience with **mocking, code smells**, **refactoring** and **Test-Driven Development (TDD)**. Mocking was discussed and demonstrated in **Lecture 6**. Code smells and refactoring were discussed and demonstrated in **Lectures 8** and **9**. If you are still facing difficulties with TDD, BrightSpace contains extensive materials on TDD. The following tools and libraries should be installed on your machine and they should be working properly to complete this assignment:

- Android Studio 4.1.1+
- Git
- Mockito 3.4.0
- Espresso framework.

Assignment Repository: https://git.cs.dal.ca/masud/csci3130-winter2021-a3

Note: To access the assignment repository, you must log in to your GitLab account.

Assignment Deadline: Feb 26 2021 5:00 PM.

Expectations: A single **PDF document** with a naming structure of **BannerID-CSCI3130-A3.pdf** should be submitted to **Brightspace** by the deadline. **Fork** the assignment repository to your **Dalhousie GitLab account** and work with your forked repository. Your solution code should be **pushed** to your **forked repository**.

- Correct *document format* and *naming structure* (**1 mark**).
- GitLab link of **your forked repository** from the assignment 3 repository is provided in the PDF document (**1 mark**).
- Make your assignment repository **private**, and add user **Prof3130** as a *maintainer* to your repository (**1 mark**)
- **Q1:** Screenshot of the passing unit test + Commits submitted to GitLab with appropriate messages (**07 marks**)
- **Q2:** Screenshot of the passing unit tests + Commit submitted to GitLab with an appropriate message (**09 marks**)
- **Q3:** Screenshot of the passing unit tests + Commit submitted to GitLab with an appropriate message (**05 marks**)
- **Q4:** Screenshot of the passing unit tests + Commit submitted to GitLab with an appropriate message (**05 marks**)
- **Q5:** Screenshot of the passing Espresso tests + Commit submitted to GitLab with an appropriate message (**05 marks**)
- Push all commits to the *master* branch of your GitLab repository (**1 mark**).

  Total marks: **35**

Questions

Q1 (07 marks): Let us consider that you have been developing an e-commerce application for an ABC company. The company wants to send a promotion offer to all the customers through your application. You have a production class called `PromoOfferSender` that takes `IPromoOffer` as a parameter in its `sendOffer()` method. Unfortunately, `sendOffer()` method is buggy and `IPromoOffer` interface is not implemented yet. However, due to a client's deadline, you still need to test `PromoOfferSender` class. As you might remember, when the implementation of a class or an external dependency is missing, we can mock them using a mocking framework called *Mockito*. Lecture 6 discussed and demonstrated this framework. Please do the following.

- Execute `PromoOfferSenderTest` and see the test failing.
- Complete the implementation of `convertToDate()` method to fix the bug in `sendOffer()` method. Make a separate commit with an appropriate message to GitLab (**03 marks**).
- Use *Mockito* framework to mock `IPromoOffer` interface and then pass the failed test from `PromoOfferSenderTest`. Once done, make a separate commit with an appropriate message to GitLab and attach a screenshot of the passing test to the answer PDF. (**04 marks**).

Q2 (09 marks): Let us consider that your e-commerce application has three types of users - *sellers*, *buyers* and *administrator*. A seller can place online ads for his/her products through your application. A buyer can respond to these ads and place online orders. On the other hand, an administrator can manage both the buyers and sellers, and can place both online ads or orders. Once a user logs in, the application determines the user's type and then provides appropriate access levels. As a part of TDD, you and your partner already implemented an ad-hoc version of access control mechanism in `AppUser` class. However, your current implementation of `AppUser` contains a `switch-case` statement, which is considered to be a code smell. Interestingly, `switch-case` statement can be replaced with *polymorphism* concept from OOP and the current code can be improved through refactoring. This problem has been extensively discussed in Lectures 8 and 9. Please do the following.

- Execute the tests from `AppUserTest` class and see the tests passing.
- Refactor `AppUser` class by (1) converting it into an interface and (2) creating three new classes – `Seller`, `Buyer` and `Admin` – that contain their desired behaviours. For example, a buyer should be allowed to place online orders but not the ads. (**05 marks**).
- Modify the *constructor* from each test method in `AppUserTest` using appropriate classes (e.g., `Seller`, `Buyer`, `Admin`) and make sure that these tests are still passing with the refactored code. Once done, make a separate commit with an appropriate message to GitLab and attach a screenshot of the passing tests to the answer PDF (**04 marks**).

Q3 (05 marks): Let us consider that your e-commerce application has an in-memory database called `ProductManager`. It stores and delivers three types of items – `Product`, `Vegetable` and `Clothes` where both `Vegetable` and `Clothes` inherit the behaviours from `Product` class. However, the current implementation contains *refused bequest* code smells, which might have led to bugs. (For example, ideally, clothes are supposed to be produced in a factory and vegetables are supposed to be grown in a farm). As a result, none of the tests from `ProductManagerTest` is currently passing. This code smell has been discussed and demonstrated in Lecture 8. Please do the following.

- Execute `ProductManagerTest` class and see the tests to fail.
- Modify `Product`, `Vegetable` and `Clothes` classes using *push down* refactoring technique and *method overriding* concept from object-oriented programming, and make the failed tests pass. Once done, make a separate commit with an appropriate message to GitLab and attach a screenshot of the passing tests to the answer PDF (**05 marks**)

Q4 (05 marks): Let us consider, in your application, you are representing customers and their phone numbers using two different classes – `Customer` and `PhoneNumber`, as provided in the assignment repository. However, the current implementation is smelly since it contains a *feature envy* code smell. This code smell has been discussed and demonstrated in Lecture 8. You will refactor the code to get rid of this smell. Please do the following.

- Execute `CustomerTest` class and see the tests to pass.
- Refactor both `Customer` and `PhoneNumber` classes using *extract method* and *move method* refactoring techniques and make sure that the tests from `CustomerTest` are still passing with the refactored code. Once done, make a separate commit with an appropriate message to GitLab and attach a screenshot of the passing tests to the answer PDF (**05 marks**)

Q5 (05 marks): Let us consider that you are working on a fun math project. Once the assignment repository is built and deployed on an emulator, you will see one text box and two press buttons – **FACTORIAL** and **SQUARE**. The text box accepts a number, and the buttons calculate the factorial and square of the number. The design can be found in `layout/activity_main.xml` file and the business code can be found in the `MainActivity` class. However, the current business code contains copy/pasted code (a.k.a., *duplicate code*) in several places, which can be refactored using *extract method* refactoring technique. At least two new methods can be extracted from the current version of the duplicate code. Please do the following.

- Execute `ExampleInstrumentedTest` and see the Espresso tests to pass.
- Refactor the business code in `MainActivity`, remove the duplicate code using *extract method* technique and then make sure that the Espresso tests from `ExampleInstrumentedTest` are still passing with the refactored code. Once done, make a separate commit with an appropriate message to GitLab and attach a screenshot of the passing tests to the answer PDF (**05 marks**)

Version: Feb 08, 2021