EECS 2311 SecZ Winter 2024 Team 3 Refactoring Document

Hashir Jamil
Mher Eric Gyulumyan
Ali Sina
Ahan Bhargava
Oscar Ye

Refactoring Tasks Performed In Iteration 3:

There are dozens of cases of continuous refactoring occurring over the life cycle of the LongBox project from January, 2024 up and until March 31, 2024 (the final revision of this document). Before iteration 3 started, there were many small-scale cases where artifacts were renamed and directory structure was alternated. There were some larger refactoring activities as well where methods were cleaned up/reorganized and classes were separated/reworked. In general, short and informal refactoring like this have been done continuously throughout this project.

In this document the refactoring tasks discussed are those large scale changes that took place in iteration 3 of this project. These tasks are listed below and described in four parts. The goal is to discuss the code smells, the nature of the code before refactoring, the refactoring pattern/strategy used and the nature of the code after the refactoring was applied. The refactoring tasks below are large in nature and span across several user stories/features.

Task 1: Separation of GUI & Controller Classes

Developers Involved

1. Hashir Jamil
   - Built on HomeController, refactored action listener logic in HomeFrame to HomeController
2. Mher Eric Gyulumyan
3. Ali Sina
   - Built on FavoriteController, refactored action listers logic in FavoritePanel to FavoriteController
4. Ahan Bhargava
   - Refactored AuthenticationPage and ComicInfoPanel, by moving action listener logic from GUI class to controller classes AuthenticationController and ComicInfoController respectively.
5. Oscar Ye
   - Built on HomeController and Profile Controller
   - Refactored “about me” section in Profile Panel by moving action listener logic from Profile Panel to ProfileController

Detected Code Smells

- Long Method

- Large Class
- Long Parameter List
- Feature Envy
- Inappropriate Intimacy
- Divergent Change
- Shotgun Surgery

Nature of Code Before Refactoring Applied

GUI classes had too much logic inside that was inappropriate to be there. It made adjusting the code base tricky and it also felt very hacky and like unclean code. Swing was not being used according to the model view architectural style it is meant to implement. This caused the code to be less readable and maintainable. Furthermore, the GUI classes were getting to be very long and were involved with too many responsibilities leading to tightly coupled client code.

Refactoring Strategies/Patterns Applied

- Extract Method
- Extract Class
- Move Method

Nature of Code After Refactoring Applied

After this refactoring was completed, the dynamic actions of the GUI classes were abstracted away into the Business Layer of LongBox into the controller package which contains individual controller classes that facilitate events that occur at the user experience level.

Task 2: Abstraction of Database Access Methods to Service Layer

Developers Involved

1. Hashir Jamil
    - Created and configured all service layer classes. Incorporated service layer classes for comic book dao and comic book reading/favorites list dao.
    - Created tests for these new service classes
2. Mher Eric Gyulumyan
3. Ali Sina
4. Ahan Bhargava
5. Oscar Ye
    - Created service layer for star rating system to access database

Detected Code Smells

- Long Method
- Large Class
- Long Parameter List
- Feature Envy
- Inappropriate Intimacy
- Divergent Change
- Shotgun Surgery

Nature of Code Before Refactoring Applied

Before this refactoring was completed, the GUI classes were directly communicating with the data access layer. This caused tight coupling between the classes and difficulties maintaining the code.

Refactoring Strategies/Patterns Applied

- Extract Method
- Extract Class
- Move Method

Nature of Code After Refactoring Applied

After this refactoring was completed all actions involving data access were positioned around the Business Logic package called service. This change leads to the creation of four service classes that serve as the CRUD API for the system database.

Task 3: DTO-Entity Mapping Operations Moved From Associated Overloaded Constructors to New Mapping Classes

Developers Involved

1. Hashir Jamil
    - Created and tested all mapping classes.
    - Removed usages of Comic Book DTO/entity constructors in main and refactored several associated tests.
2. Ahan Bhargava
    - Removed usages of CommentDTO constructors
3. Oscar Ye
    - Removed usages of StarRatingDTO constructors

Detected Code Smells

- Long Method
- Large Class
- Long Parameter List
- Dead Code
- Feature Envy
- Inappropriate Intimacy

Nature of Code Before Refactoring Applied

Before refactoring the conversion from data transfer object to entity and vice-versa occurred using a complex series of constructors in both types of classes. This led to lots of unreadable code that made the classes too big. One particular issue was keeping track of

Refactoring Strategies/Patterns Applied

- Extract Method
- Extract Class
- Replace Parameter with Method Call
- Deletion
- Move Method
- Hide Delegate

Nature of Code After Refactoring Applied

After this refactoring was applied, the Dto classes have less responsibility and mapping is now done using custom mapper classes with static methods using the Java Streams API. This method is a well accepted implementation of the DTO and Repository pattern pairing that is present in LongBox. It will likely perform and scale much better. As well it will likely be more maintainable and readable for future development tasks.