**SSE 659 Project 2**

Refactoring the Deli App

Mike Dumas, Kei’Shawn Tention, Jewl Johnson

**Abstract**

Demonstrate knowledge of topics covered in Chapters 3-12 of the text Refactoring – Improving the Design of Existing Code. Each team member will submit his or her own report.

Kei’Shawn Tention’s Submission

March 23, 2015

Table of Contents

1. Introduction 3

2. Development Tools 3

2.1 Eclipse 3

2.2 GitHub 3

3. What is Refactoring? 3

3.1 What are we Refactoring? 3

4. TDD in Conjunction with Refactoring 4

4.1 Deli App Tests 4

5. Chapter 7 – Moving Features Between Objects 5

5.1 Overview 5

5.2 Implementation 5

6. Chapter 8 – Organizing Data 5

6.1 Overview 5

6.2 Code Clean Up 5

6.3 Implementation 7

7. Conclusion 8

8. Appendix 8

8.1 Results 8

8.2 Source Code 8

# 1. Introduction

The purpose of Project 2 is to demonstrate one’s knowledge of the topics covered in Chapters 3 – 12 of Refactoring: Improving the Design of Existing Code, written by Martin Fowler.

# 2. Development Tools

## 2.1 Eclipse

## 2.2 GitHub

# 3. What is Refactoring?

## 3.1 What are we Refactoring?

The Deli Training application is a rather complex application. The classes that make up the application are as followed:

* *main.java*
* *Products.java*
* *Results.java*
* *testing.java*
* *GameEngine.java*

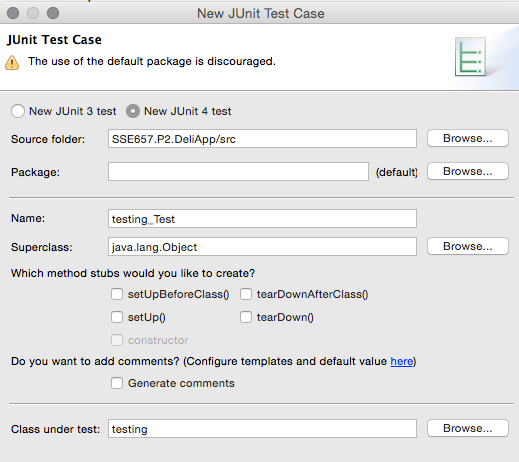
For the most part, the classes drive the GUI aspect of the Deli Application. The source code for each class be viewed in [Appendix 8.2](#_8.2_Source_Code). The two main classes that control the functionality of the Deli Application are *testing.java* and *GameEngine.java*.

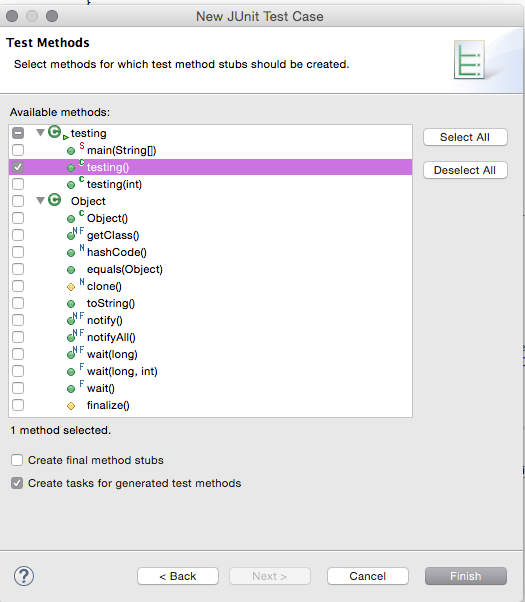
After thoroughly examining each class file, we noticed that the *testing.java* and *GameEngine.java* are the classes that refactoring could be applied to. In *testing.java*, we noticed that the method *loadProducts()*  was rather lengthy, which was an indication that refactoring was necessary. Also within that method, we recognized duplicate code. In *GameEngine.java*, we recognized the excessive use of conditional statements and concluded that refactoring can be performed to simplify the code.

Each member of the team had their own particular area of concern to refactor, hence the separate reports. One are I want to address and refactor pertains to the Organizing Data, which is covered in Chapter 8 of Martin Fowler’s text, Refactoring: Improving the Design of Existing Code. The second area of concern is Moving Features Between Objects, which is covered in Chapter 7 of the same text. [Section 5](#_5._Chapter_7) and [Section 6](#_6._Chapter_8) will discuss my knowledge and implementation of Chapters 7 and 8, respectively.

# 4. TDD in Conjunction with Refactoring

## 4.1 Deli App Tests





# 5. Chapter 7 – Moving Features Between Objects

## 5.1 Overview

## 5.2 Implementation

# 6. Chapter 8 – Organizing Data

## 6.1 Overview

## 6.2 Code Clean Up

Before implementing the concepts behind data organization, I notice that the *testing.java* class contained unnecessary lines of code. It seems that the lines of code were initially used to do validation within the class file. The side-by-side view of the code refactoring is show below.

|  |  |
| --- | --- |
| Original *testing.java* | Modified *testing.java* |
| **…..**  **…..**  **…..**  **public** **class** testing {  **protected** JFrame frmDeliTrainingApplication;  Random generator = **new** Random(System.*currentTimeMillis*());  **int** max = 0, rand\_product = 0, rand\_product2 = 0, rand\_option = 0, score = 0, questions = 0, difficulty = 0;  String name = "", name2 = "", type = "", type2 = "";  GameEngine Engine = **new** GameEngine();  Results resultsWindow;    /\*\*  \* Launch the application.  \*/  **public** **static** **void** main(String[] args) {  EventQueue.*invokeLater*(**new** Runnable() {  **public** **void** run() {  **try** {  testing window = **new** testing();  window.frmDeliTrainingApplication.setExtendedState(JFrame.***MAXIMIZED\_BOTH***);  window.frmDeliTrainingApplication.setVisible(**true**);  } **catch** (Exception e) {  e.printStackTrace();  }  }  });  }  /\*\*  \* Create the application.  \*/  **public** testing() {  loadProducts();  pickProducts();  initializeTesting();    }    **public** testing(**int** difficulty){ //Constructor passing in level of difficulty  **this**.difficulty = difficulty;  loadProducts();  pickProducts();  initializeTesting();  }    **private** **void** loadProducts() {  …..  …..  ….. | **…..**  **…..**  **…..**  **public** **class** testing {  **protected** JFrame frmDeliTrainingApplication;  Random generator = **new** Random(System.*currentTimeMillis*());  **int** max = 0, rand\_product = 0, rand\_product2 = 0, rand\_option = 0, score = 0, questions = 0, difficulty = 0;  String name = "", name2 = "", type = "", type2 = "";  GameEngine Engine = **new** GameEngine();  Results resultsWindow;      **public** testing(**int** difficulty){ //Constructor passing in level of difficulty  **this**.difficulty = difficulty;  loadProducts();  pickProducts();  initializeTesting();  }    **private** **void** loadProducts() { …..  ….. |

## 6.3 Implementation

# 7. Conclusion

# 8. Appendix

## 8.1 Results

## 8.2 Source Code