**Code Refactoring Report for**

**Java Refactoring Test Project**

**Created By:**

Sajjad Ashraf

ssajjadashraf@yahoo.com

**Date:**

09 Oct, 2018

**Table of Contents**

[1. Introduction 3](#_Toc526845144)

[2. Steps for Improvements: 3](#_Toc526845145)

[ Java docs, Code formatting : 3](#_Toc526845146)

[ Handle Concurrent Requests 3](#_Toc526845147)

[ Removing element while Iterating through Array List 3](#_Toc526845148)

[ Update, Delete and Find user operation operate on the base of user’s email 3](#_Toc526845149)

[ Singleton Design pattern for ‘UserDao’ 4](#_Toc526845150)

[ Use proper HTTP method in ‘UserResource’ 4](#_Toc526845151)

[ New packages and Classes 4](#_Toc526845152)

[ Java Doc generation 4](#_Toc526845153)

[ Spring Boot Project Creation 4](#_Toc526845154)

[Git Repo : 4](#_Toc526845155)

[How to Run It: 4](#_Toc526845156)

# 

1. Introduction

The task is to refactor the ‘java-test’ project in a way that it would break any existing functionality, test cases and improve the code readability, extensibility and maintainability.

2. Steps for Improvements:

* Java docs, Code formatting :

I have added Java docs in every class it will help to understand the code. I have also formatted the code (code indentation) in every class of the project so that it looks consistent and easy to read. I also remove unused variable and imports (dead code).

* Handle Concurrent Requests

To facilitate concurrent requests, I have introduced ‘**synchronized**’ code block to the places where we are performing adding, updating and deleting a user in ‘UserDao’ class.

|  |
| --- |
|  |

We can also handle the concurrent request by using the following code Collections.synchronizedList(new ArrayList<>());

It is preferable to synchronize only part of a method which can cause a concurrency issue.

* Removing element while Iterating through Array List

‘deleteUser’ method of ‘UserDao’ class has been updated to avoid java.util.ConcurrentModificationException .

* Update, Delete and Find user operation operate on the base of user’s email

As given, email is a unique key, that's why these (above) operations must work on the basis of email, previously they were working on the basis of the name. An error will be thrown if the user does not provide the email address or at least one role while creating the user.

* Singleton Design pattern for ‘UserDao’

The way we are using ‘UserDao’ class, we have to apply a creational design pattern on it and according to the current situation Singleton Design pattern best suits for ‘UserDao’.

* Use proper HTTP method in ‘UserResource’

I have updated HTTP Method type to its correct corresponding CRUD operations type.

addUser to POST.

updateUser to PUT.

deleteUser to DELETE.

getUsers and find findUser to GET.

* New packages and Classes

I have created a few new packages and new classes,

like 'User' class, move under 'com.h2rd.refactoring.model' package, 'UserService' class created under 'com.h2rd.refactoringservice' package. The goal is to keep classes of the similar domain under the same package, so it makes easy to understand and update the code.

* Java Doc generation

Java doc of the project has been generated and can be view by browsing ‘java-test\doc\index.html’ file.

* Spring Boot Project Creation

I have some prior sound experience in Java Spring boot, So I have transformed the current project (whatever we have after refactoring ‘java-test’ project) into a new Spring Boot Application. I have kept the same test case that We had in ‘java-test’ project.

I know for Spring Boot Application, these test cases are not perfect enough to validate the application but due to lack of the time I could not write the Spring boot supported test cases, but still, these case works well.

Git Repo :

<https://github.com/sajjad037/java-test-sb.git>

How to Run It:

Go to ‘com.h2rd.refactoring’ package and simply run the ‘JavaTestSbApplication’ class. **Note**: make sure port 8080 is not used by any other service before you execute the project.