**City University of Hong Kong**

2017-2018 Semester A

CS3343 Software Engineering Practice

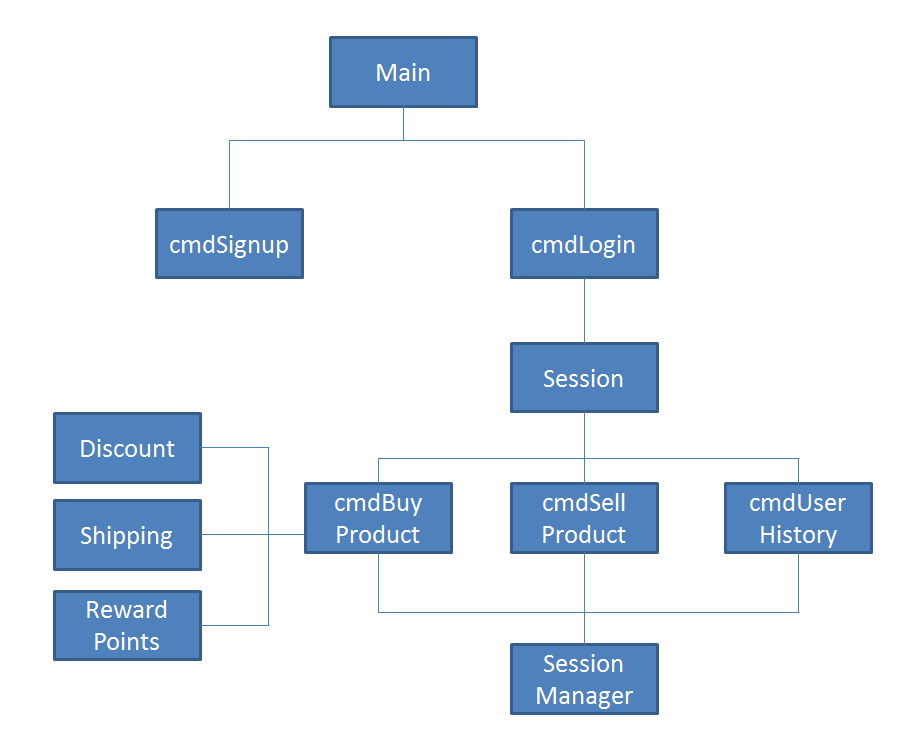
**Test Report**

Maverick:

The Smart Trading Platform

Group 14

|  |  |
| --- | --- |
| **Name** | **Student Number** |
| Dheeraj Khurana | 54227224 |
| Navwinder Singh | 54395023 |
| Ravinder Singh | 54390586 |
| Pang TSZ HO | 54387730 |
| Mathew Bilo | 54387858 |

1. **Hierarchy of Program Modules**

The hierarchy of program modules above shows the structure of our program. Main.java will call cmdSignup.java and cmdLogin.java according to the user’s choice. When the User login to our system, cmdLogin.java will invoke Session.java to which gain the user the ability to buy and sell items or view their transaction history. When the user wants to buy items, Session.java will invoke cmdBuyProduct.java. After the user select the item he wants, cmdBuyProduct.java will calls SessionManager.java to manage the user’s transaction details. During the end of the buy process, Discount.java, Shipping.java and RewardPoints.java will be invoke to compute the bill of the purchase.

1. **Testing Plan and Strategies**

We decided to use the Bottom-Up testing approach as our testing strategy. Most of the computational modules in our system are located at the bottom level of our program structure, using the Bottom-up approach allow us to start the testing process on those critical modules as soon as possible. In additional, the Bottom-up approach provide us a clear path as to which part of the program we will test next. This makes our life a lot easier as we can plan and control the whole testing process at ease.

We began our testing process performing unit testing on the following modules:

1. Discount.java
2. Shipping.java
3. RewardPoints.java
4. Sessionmanager.java
5. cmdSignup.java

After unit testing, we perform integration testing by testing the modules and all its underlying modules.

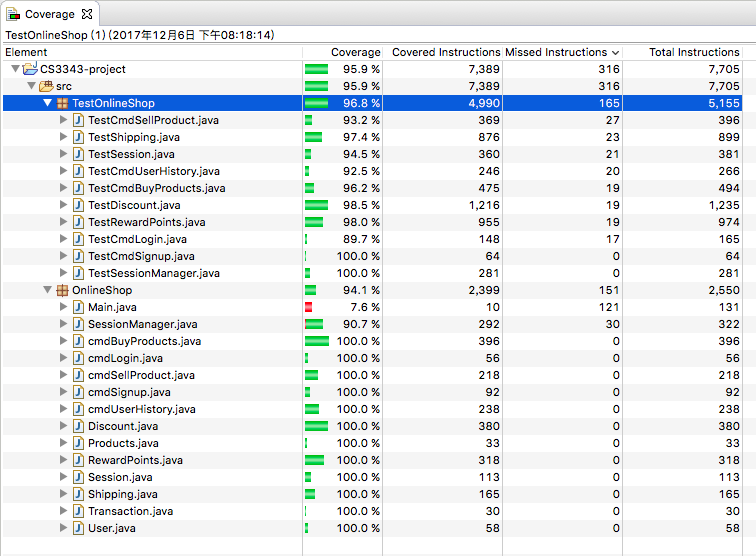
1. cmdBuyProduct.java + Discount.java + Shipping.java + RewardPoints.java + Sessionmanager.java
2. cmdSellProduct.java + Sessionmanager.java
3. cmdUserHistory.java + Sessionmanager.java
4. Session.java + cmdBuyProduct.java + Discount.java + Shipping.java + RewardPoints.java + cmdSellProduct.java + cmdUserHistory.java + Sessionmanager.java
5. cmdLogin.java + Session.java + cmdBuyProduct.java + Discount.java + Shipping.java + RewardPoints.java + cmdSellProduct.java + cmdUserHistory.java + Sessionmanager.java

Lastly, we conduct system testing on the whole system.

1. **Code Refactoring**

During the testing process, we have identified a lot of code smells in our system. To deal with the code smells we discovered, we conducted some code refactoring to improve the quality of our system bit by bit while ensuring the functionality of the code remain the same.

Coverage before refactoring:

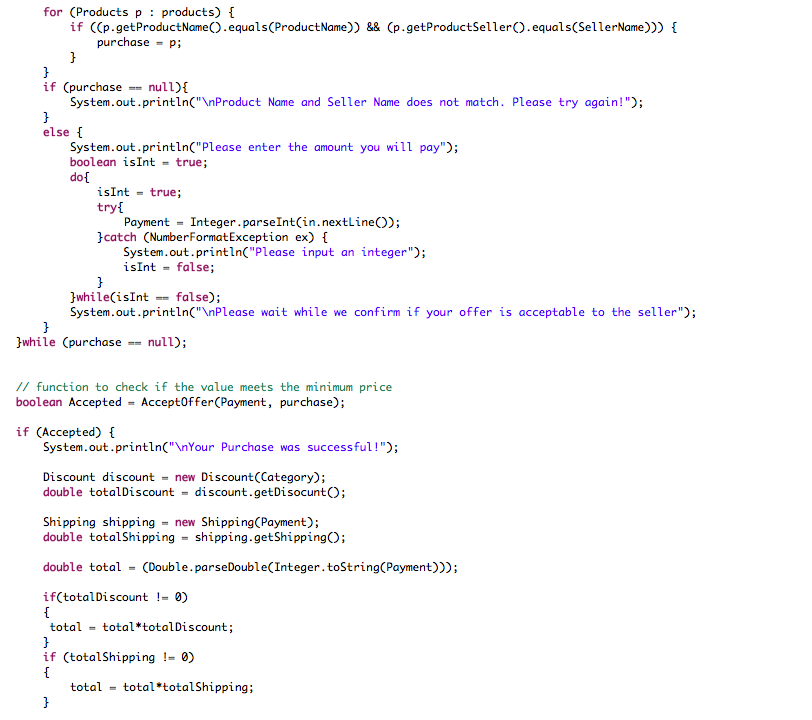


Here are some example of code refactoring we have done to our system:

1. Extract Method:

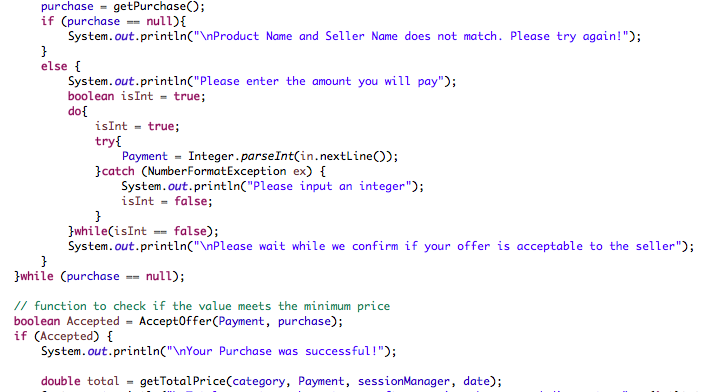
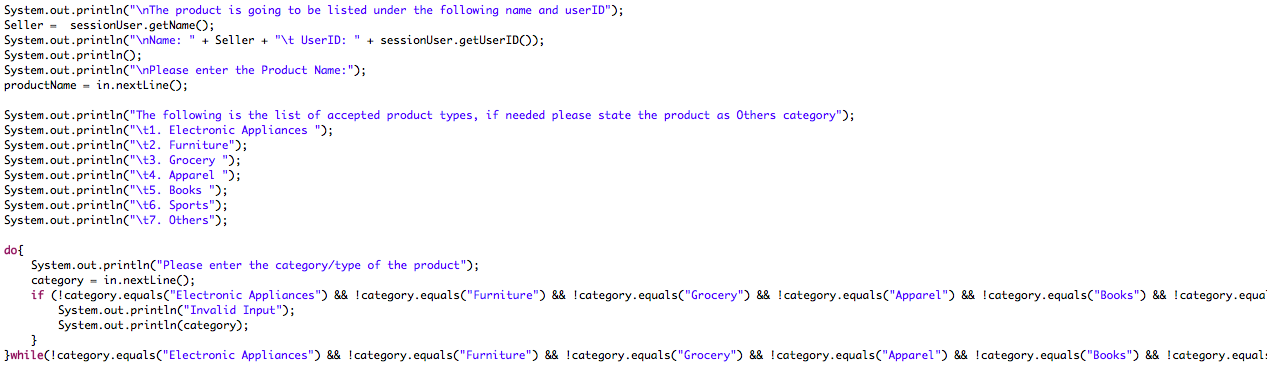
Before:

The execute() method in cmdBuyProduct.java has a lot of if else loop handling different part of the shopping process which makes the code look messy.



After:

The code for getting user’s purchase and total price is extracted to make the code looks cleaner and nicer.



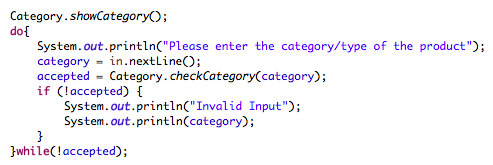
1. Extract Class

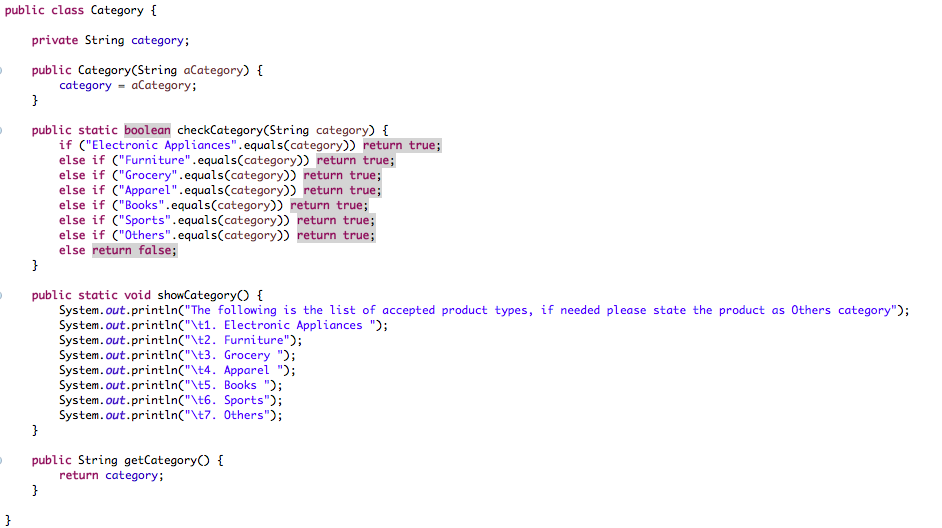
Before:

A lot of system.out.print statement is needed to print out the category list and a long if conditions is needed to check if the input is correct or not.

After:

Checking related to category is extracted to form the Category class. Now, we only need to use one line to call the method in Category.java to show the category list and check if input is correct or not.





1. Extract Interface

Since the singleton class SessionManager contains all the data of users, products and transactions, we want to setup a test stub when testing classes that are associated with SessionManager to make sure the data are not interfere by the test cases.

Before:

Cannot extends singleton class.

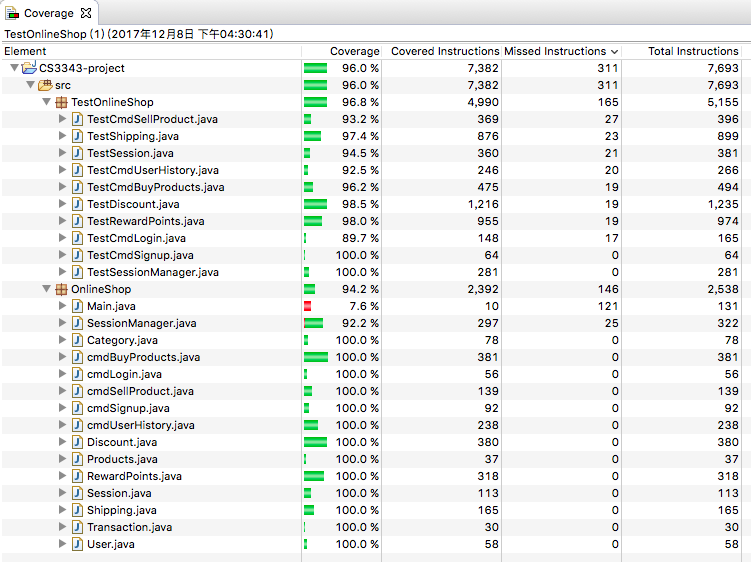


After:

Our test stub can extends Manager.



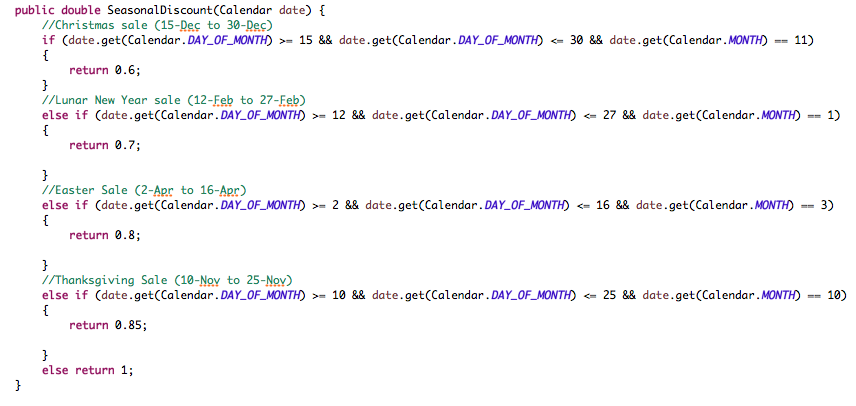
Coverage after refactoring:



1. **Coverage Strategies**

Both control flow testing and predicate testing were adopted during the testing process of our system. For control flow testing, we make sure that our test cases cover as much statement, branches, loops and path as possible.

On the other hand, we make sure that every conditions and decision in our program has been evaluated to be true and false. C/DC coverage is applied such that both condition coverage and decision coverageare satisfied by the same test set. Here is a example:

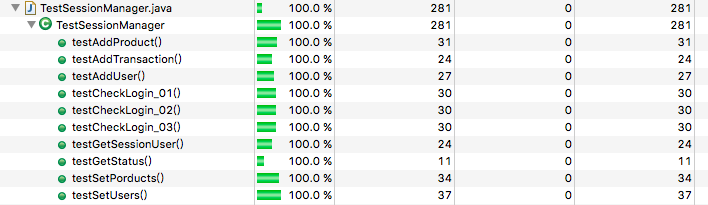


Truth Table:

A = date.get(Calendar.DAY\_OF\_MONTH) >= 15 && date.get(Calendar.DAY\_OF\_MONTH) <= 30 && date.get(Calendar.MONTH) == 11

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Case | DAY\_OF\_MONTH | MONTH | DAY\_OF\_MONTH >= 15 | DAY\_OF\_MONTH < = 30 | MONTH = 11 | A |
| testSeasonalDiscount\_01 | 3 | 5 | false | true | false | false |
| testSeasonalDiscount\_02 | 20 | 11 | true | true | true | true |
| testSeasonalDiscount\_03 | 14 | 11 | false | true | true | false |
| testSeasonalDiscount\_04 | 31 | 11 | true | false | true | false |
| testSeasonalDiscount\_05 | 15 | 1 | true | true | false | false |

1. **Test Cases**
2. SessionManager.java



1. testAddProduct and testSetProduct

Purpose: to test the ability of SessionManager in handling products.

1. testAddUser and testSetUser

Purpose: to test the ability of SessionManager in handling user data.

1. testAddTransactions

Purpose: to test the ability of SessionManager in handling transactions.

1. testCheckLogin\_01 – 03

Purpose: to test the ability of SessionManager to determine which user has login.

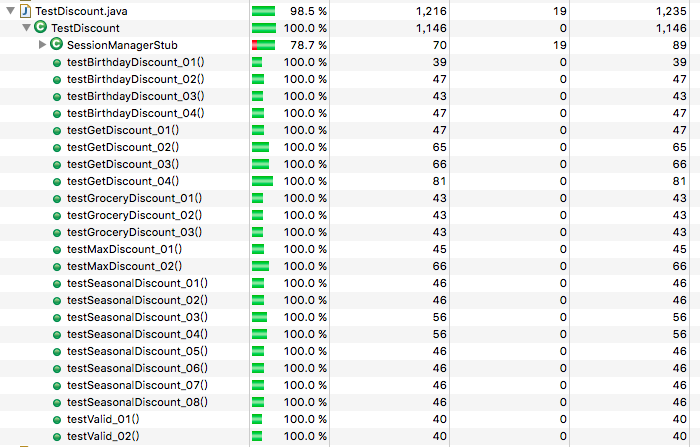
1. testGetSessionUser

Purpose: to test if SessionManager can return the correct session user.

1. testGetStatus

Purpose: to test if SessionManager can return the correct status.

1. Discount.java



1. SessionManagerStub

Purpose: to inject fake data for testing so that we don’t have to mess with actual data.

1. testBirthdayDiscount\_01-04

Purpose: Check whether the method BirthdayDiscount() can determine which discount to return according to the user’s birthday.

1. testSeasonalDiscount\_01-08

Purpose: Check whether the method SeasonalDiscount() can determine which discount to return according the day the user is shopping.

1. testValid\_01-02

Purpose: Check whether the method valid() can determine if the user is valid for getting a discount.

1. testGroceryDiscount\_01-03

Purpose: Check whether the method GroceryDiscount() can determine which discount to return according the day the user is shopping and the category of product they brought.

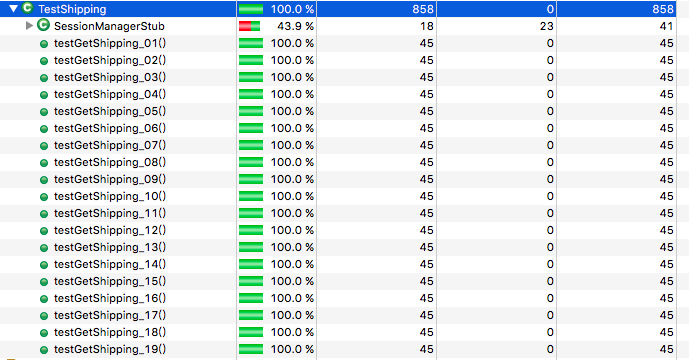
1. testMaxDiscount\_01-02

Purpose: Since a user can only get two discount at max, these test case check whether the method MaxDiscount() can compute the max discount possible for the user to get.

1. testGetDiscount\_01-04

Purpose: Check whether the method GetDiscount() can determine which discount to return and covers the rest of the discount computation leftover.

1. Shipping.java



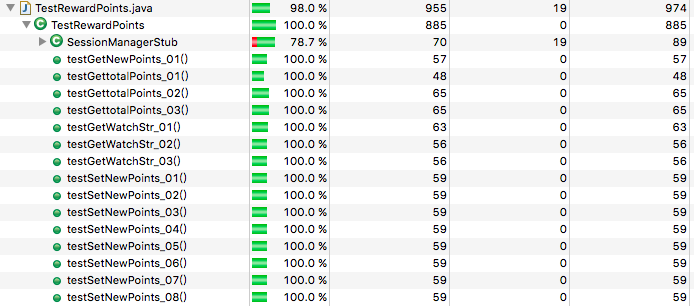
1. SessionManagerStub

Purpose: to inject fake data for testing so that we don’t have to mess with actual data.

1. testGetShipping\_01-19

Purpose: Check whether the method GetShipping() can compute which discount to return according to the region the user is from, when the user is making the purchase and the amount they spent.

1. RewardPoints.java



1. SessionManagerStub

Purpose: to inject fake data for testing so that we don’t have to mess with actual data.

1. testGetNewPoints

Purpose: Check whether the method GetNewPoints() can compute the correct reward points to return according to the day the user made a purchase.

1. testGettotalPoints\_01-03

Purpose: Check whether the method GetTotalPoints() can compute the correct total reward points to return according to the day the user made a purchase and the user’s transaction history.

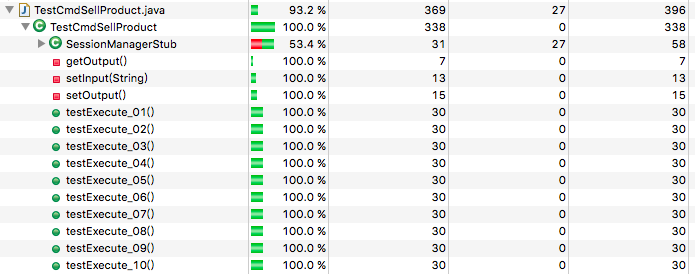
1. testGetWatchStr\_01-03

Purpose: Check whether the method GetWatchStr() can correctly compute if the user’s reward points are enough to claim their reward.

1. testSetNewPoints\_01-08

Purpose: Check whether the method SetNewPoints() can compute the correct reward points to return according to the day the user made a purchase.

1. cmdSellProduct.java



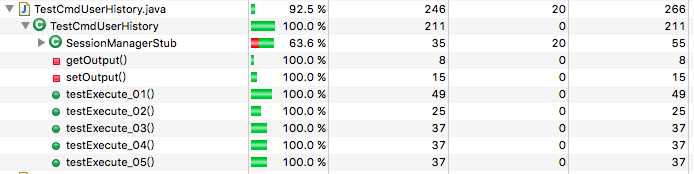
1. SessionManagerStub

Purpose: to inject fake data for testing so that we don’t have to mess with actual data.

1. testExecute\_01-10

Purpose: to test every case that could happen during the sell process.

1. cmdUserHistory.java



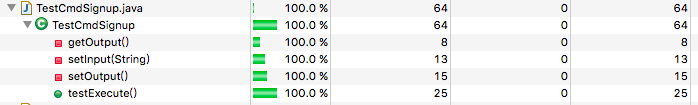
1. SessionManagerStub

Purpose: to inject fake data for testing so that we don’t have to mess with actual data.

1. testExecute\_01-05

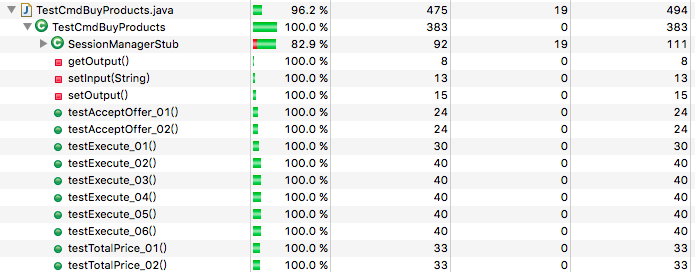
Purpose: to test every case that could happen during the check user history process.

1. cmdSignup.java



1. testExecute

Purpose: to test the sign up process.

1. cmdBuyProduct.java
2. SessionManagerStub

Purpose: to inject fake data for testing so that we don’t have to mess with actual data.

1. testAcceptOffer\_01-02

Purpose: to test whether the method AcceptOffer() can determine if the offer of the buyer is accepted by the seller.

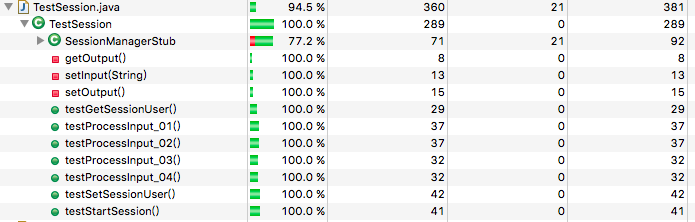
1. testTotalPrice\_01-02

Purpose: to test whether the method getTotalPrice() can compute the correct total price after discount and shipping.

1. testExecute\_01-06

Purpose: to test every case that could happen during the shopping process.

1. Session.java



1. SessionManagerStub

Purpose: to inject fake data for testing so that we don’t have to mess with actual data.

1. testGetSessionUser and testSetSessionUser

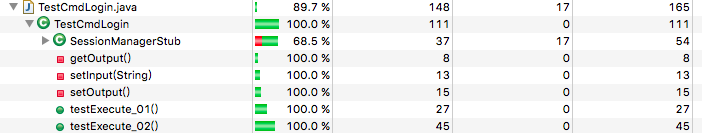
Purpose: check the ability of Session to determine the correct session user.

1. testProcessInput\_01-04

Purpose: to test every case that could happen during the shopping process.

1. testStartSession

Purpose: integration testing on Session.java + cmdBuyProduct.java + Discount.java + Shipping.java + RewardPoints.java + cmdSellProduct.java + cmdUserHistory.java + Sessionmanager.java

1.  cmdLogin.java
2. testExecute\_01-02

Purpose: to test every case that could happen during the login process and integration testing on cmdLogin.java + Session.java + cmdBuyProduct.java + Discount.java + Shipping.java + RewardPoints.java + cmdSellProduct.java + cmdUserHistory.java + Sessionmanager.java.