Milestone 3 Report

Tawseef Patel (101145333)

Gilles Myny (101145477)

Questions:

1. What were the testing methods/strategies that you used for this milestone? Be detailed. Use the testing terminology presented in lectures.

The methods and strategies used to test this milestone was Junit 5.4 as this is the latest public version available. JUnit 5 is different from JUnit 4 because of the tags that are used such as *@BeforeAll*, *@BeforeEach*, *@Test*, *@AfterAll*. The tags that were used to test the Inventory and StoreManager classes were *@BeforeAll* to initialize the variables that are declared in the global scope of the test class. Next the *@Test* tags were used to test out most of the methods from the StoreManager and Inventory to make sure they were operating correctly and/or giving the right output. In cases where bugs were found such as in the Inventory class, those bugs were quickly fixed to make sure the program works correctly as expected.

2. Were there some things you were unable to test with JUnit? What were they, and why were you not able to? (Think about the levels of testing.)

Most of the things were able to be tested in the test classes except for those methods which had object type returns which cannot be directly compared because of the simple operation of the method. For example, the getInv() method in StoreManager is not necessarily required to be tested as it is a getter which only has one line of code which allows access to the inventory that the store manager controls. Also when testing the getInv() method which has a Object return type, it is difficult and very tedious to figure out as using the Assertions.arrayEquals method would return the address of the object rather than the contents of it.

3. With respect to Question 2, should these parts of the code be tested? Should every inch of code be tested in general?

All parts of codes do not need to be tested in cases where methods are very simple and it is obvious what it is doing, for example a getter method which has one line of code returning a variable which has been initialized via a setter is not necessary to test because it is obvious what is being done.

Changelog
Version 3.0 (2020-03-21)
Release Highlights
- StoreView.java
- StoreManager.java
- ShoppingCart.java
- Inventory.java
- Product.java
- storeTest.StoreManagerTest
- storeTest.InventoryTest
- StoreUML_M3.pdf

Features
- StoreView.java:
New Additions: Added exception handling for InputMismatchException. This exception occurs when the user input is not an integer. Fixed a bug where a user was allowed to remove products that did not exist from cart.

> The StoreView is designed to be the front end of the project where it creates a static StoreManager who has control of the entire front end and allows the user to select the StoreView they wish to browse on. This class contains the main method which takes user input for the StoreView they wish to view followed by giving the option to Browse the store, go to Cart, get Help, or Quit the current StoreView. The user can add an item to cart by clicking B for browse after selecting their desired StoreView and can then enter the product id followed by the quantity they want, and the product is then added to cart. The user then has the option to click B once again for browse or C for cart which then prints out the items in cart and gives the option to Remove from cart, Checkout, or Continue back. If the user selects R for remove then the user has to enter the product id and quantity they wish to remove. If the user selects CH for checkout then the transaction is processed and the option to select a new StoreView is brought up once again. If the user selects CO to continue back to the store they can then choose B for browse, C for Cart, H for help or Q for quit. Once a user selects Q, the StoreView is deactivated and cannot be used again and if all the active StoreViews are deactivated then the program finishes with exit code 0.

- StoreManager.java
New Additions: Added the ability to retrieve a shopping cart object that is stored and controlled by the manager using a given cartID.

> Milestone 2 Description: Completely revamped the StoreManager class such that upon creation of the manager, there is an inventory that is created along with a HashMap of carts and the associated cartID. From there, multiple methods were implemented such as getters to call on from the StoreViews manager to access the inventory and cart that are both controlled by the StoreManager. Further, the StoreManager now has the ability to add/remove items from the cart. Milestone 1 Description: creates new Inventory and a variety of products given their name,

ID, and price. Tests adding the created products to the inventory with stock amount; restocking products given their ID’s with amount to restock; removing stock of products given their ID’s with amount to remove. Returns a Boolean value if all tests are successful.

- ShoppingCart.java

New Additions: Added the ability to retrieve the quantity of a product that is in the cart using a Product object. Also added a method which creates and returns an arraylist of all the products in the shoppingcart.

The ShoppingCart class is designed to add and remove products from the users shopping cart in the store after they have browsed via StoreView.java. This is done by using a HashMap which has a Key as a Product and value which is an Integer representing the quantity of the product. The addToCart method is designed to add a certain quantity of a certain product which the user selects into the cart and then prints out all the items in cart along with the total. The removeFromCart method is designed to do the exact opposite. It removes a certain quantity of a product if the product exists in cart along with a quantity greater than what the user wants to remove and then prints the updated cart. The viewCart method was created so the user can access the ShoppingCart and view the items in the cart, price, and total.

- Inventory.java

New Additions: bug fixes

Milestone 2 Description: Javadoc Format, added the ability to use Hashmap instead of two ArrayLists. Renamed methods and created new ones to better accommodate the move to Hashmap.
Milestone 1 Description: Inventory class allows StoreManager to create a new inventory. Contains two ArrayList’s for products and stock information. Setters allow StoreManager to create (if the product does not exist) or add and remove product stock given the product name or the product ID. Getters allow StoreManager to get product info with the product ID, returns the price of a quantity of items given the product ID, or get stock amount with the product ID.

- Product.java

Milestone 2 Description: Javadoc formatting, added and implemented much more efficient equals methods which now have the ability to check given a product object attributes, if it equals to the attributes of a product object.
Milestone 1 Description: Allows StoreManager to create and set a name, ID, and price for the product. Includes getters and setters for required public information. Creates a print override for proper printing of each product to console. Checks for duplicate products.