CS5010, Fall 2022

Assignment 2

Brian Cross
Assignment adapted from Abi Evans and Tamara Bonaci.

Submission details

- This assignment is due no later than 11:59pm on Monday, October 10th, 2022
- Create a topic branch from main called "assignment2" in your personal class repo.
- Make a folder called "assignment2" in your repo. You'll create your project here.
- When you are ready to submit, create a pull request to the main branch. Add the TAs as reviewers.
    - See Lab 2 for help
    - When you create a Pull Request, you may make updates if you see things you want to fix up, but do so before the deadline so that it isn't considered late.

Resources

You might find the following online resource useful while tackling this assignment:

- Java 17 Documentation
- JUnit 5 Getting Started Online Documentation
- Sample Java Project in Code_From_Lectures Repo
- Gradle Building Java Projects
- UML Diagram Online Documentation
- Java Tutorial: Interfaces and Inheritance (especially polymorphism)
- Java Tutorial: Generics

Gradle

In this assignment, you will continue to Gradle as we used in Assignment 1. You should be able to copy the build.gradle file from your previous assignment.

Submission Requirements

Your repository should contain a package **`assignment2`**, and within that package, a separate package for every problem.

Additional expectations:

- One `.java` file per Java class
- One `.java` file per Java test class
- One pdf or image file for each UML Class Diagram that you create
- All public methods must have tests
- Must have at least 70% code coverage (code and branching) from your tests.
- All non-test classes and non-test methods must have valid Javadoc

The latest .gitignore on Piazza should prevent these automatically for you:

- Any `.class` files
- Any `.html` files
- Any IntelliJ specific files

Lastly, a few more rules that you should follow in this assignment:

- Your classes should have a `public` modifier
- Instance fields should have a `private` modifier
- Use `this` to access instance methods and fields inside a class

Assignment

Problem 1

You are tasked with building part of a supermarket's inventory system, which will later support online ordering for in-store pick-up (Problem 2).

Part A – Representing Products

You will need to represent **`Products`** that are for sale in the store. At the moment, the inventory system only needs to support the following types of products:

- **`Grocery`**
- **`Household`**

It is likely that the system will be expanded in future to handle more types of products.

The system is in the very early stages of development, so you only need to keep track of a few specific products to enable testing of your implementation. For now, the system will only track the following types of **`Grocery`** products:

- **`Salmon`**
- **`Cheese`**
- **`Beer`**

For now, the system will only track the following types of **`Household`** products:

- **`Paper towels`**
- **`Shampoo`**

Each **`Product`** will need to track the following information

- **`Manufacturer`**, e.g. “Beechers”
- **`Product name,`** e.g. “Flagship”
- **`Price`**
- The **`minimum age`** a customer needs to be in order to buy the product. In most cases, this can be set to 0 but for age-restricted products, this field will need to be set when the product is created. For example, the minimum age for purchasing beer is 21.

In addition, all **`Grocery`** products should track the **`weight`** of the product in ounces and all

`Household` products should keep track of the number of individual **`units`** in a package. For example, a package of paper towels can contain a single roll, 2 rolls, or some other number. This information will be important in Problem 2.

Part B – Keeping Track of Stock

The system will represent the quantities of each **`Product`** as a **`StockItem`**, which contains the following information:

- A **`product`** as described above.
- The **`quantity`** of this product that the supermarket has in stock.

The system will need to support the following functionality for stock:

- Check if there are **enough items in stock** to complete a purchase.
- **Reduce the quantity of the item in the event of a purchase**. You should not allow a purchase to go ahead if there are not enough items in stock.

Part C – The Inventory

This part of the system pulls together all of the information about the supermarket’s stock and implements some of the core functionality for the system. For now, the **`Inventory`** will need to keep track of the following:

- **`Grocery stock`** – a list of all grocery StockItems
- **`Household stock`** – a list of all household StockItems

You may use Java's built-in collections to implement the lists.

The `inventory` should support the following functionality:

- When the supermarket decides to stock a new product, the inventory should allow **a new StockItem to be added** to the appropriate list. Note that this does not mean increasing the quantity of an existing item.
- The supermarket should also be able to **get the total retail value of all items in stock**.

Your tasks

1. Design Java classes to capture the above requirements and information.
2. Please write appropriate Javadoc documentation for all of your classes and methods.
3. Please write the corresponding test classes for all of your classes.
4. Please provide a final UML Class Diagram for your design.

Problem 2

Now that the basic inventory functionality is complete, you are tasked with extending the system to support online ordering for in-store pickup. As you will need to use the code written for Problem 1, you do not need to create a separate package for problem 2.

For this part of the system you will need to keep track of `Customers` and `Receipts`. You will also need to update the `inventory`.

Part A – Customers and Shopping Carts

Each `Customer` should have a **name**, an **age** and a **shopping cart**.

The **shopping cart** should contain a **list of products** the customer has added to their cart. It should enable the **total cost of all items** in the cart to be calculated.

Part B – Order Fulfillment and Processing

The supermarket wants the online ordering and fulfillment process to work as follows:

1. A customer views a product online and decides to **add the product to their shopping cart**.
    a. By default, the customer will not have to specify a quantity. In this case the system will add one item of the product to the cart. However, the customer can choose to add larger quantities to their cart.
    b. If the customer tries to add more items than are in stock, an error message will be thrown but the customer should be able to continue shopping.

   c. Adding an item to a customer's cart does not change the stock quantity for that item. Stock will be adjusted when the order is actually processed (#3 below).

   d. The customer continues shopping, adding more items to their cart. You do not need to worry about submitting the final order—another team will handle this part of the system.

2. Sometime after an order has been placed (you do not need to account for the passage of time), supermarket staff **fulfill the order** by gathering all the items in the cart ready for pickup. Because there is a delay between the customer adding items to their cart and the order being fulfilled, sometimes items that were in stock when the customer placed their order are out of stock by the time the order is fulfilled. In this case, the supermarket will **substitute** the original item with an equivalent item using the following rules:

   a. Products can only be substituted with the same specific type of product e.g. a Cheese product can only be substituted with another Cheese product.

   b. The substituted item must be in stock (i.e. have quantity greater than 0).

   c. Products can only be substituted with another product that is the same price as or cheaper than the original product.

   d. For Grocery products, the weight of the substitution must be the same as or greater than the original product.

   e. For Household products, the number of units in the package must be the same as or greater than the original product.

   f. If no suitable substitution is found, the item will be removed from the customer's cart.

3. Once all of the items in the shopping cart and any necessary substitutions have been gathered **the order is processed**, which involves the following steps.

   a. Any items the customer is not old enough to buy are removed from the cart.

   b. The quantities of all stock items purchased are updated in the system.

   c. The customer's shopping cart is emptied.

   d. A receipt is returned summarizing the order.

The **`Receipt`** should contain the following information:

- The **total price** paid.
- The **list of products the customer received** when the order was processed.
- A **list of any products that were out of stock** and could not be substituted.
- A **list of any products that were removed** from the order because the customer did not meet minimum age requirements.

Your tasks

1. Design Java classes to capture the above requirements and information.
2. Write appropriate Javadoc documentation for all of your classes and methods.
3. Write the corresponding test classes for all of your classes.
4. Provide a final UML Class Diagram for your design.