Milestone 2
Questions

Bardia Parmoun 101143006

Guy Morgenshtern 101151430

1. What kind of relationship is the StoreView and StoreManager relationship? Explain.

The relationship between the two classes is aggregation. StoreManager is a part of the StoreView class, but StoreManager can also exist independently of the StoreView class.

2. Due to their behavioral similarity, would it make sense to have ShoppingCart extend Inventory, or the other way around? Why or why not?

Having ShoppingCart extend Inventory would make the most sense. The Inventory class is the more general of the two classes. A ShoppingCart can be viewed as a personal Inventory for a customer when they enter the store. Since the hierarchy of inheritance goes from the most general parent class, to the most specific child class, having ShoppingCart inherit from Inventory makes the most sense.

3. What are some reasons you can think of for why composition might be preferable over inheritance? These do not have to be Java-specific

Inheritance creates tightly coupled classes. This can cause errors in the code if the parent class is changed without making the proper adjustments in the child class. Composition results in more loose coupling between the classes where changes can be made in the classes without as much risk breaking the code.

Since composition is a “has a” relationship, it is beneficial to use when two classes are not related to each other, but one is made of the other. For example, a house class is made up of multiple room classes. In this example, a room is just a part of the house, not a type of house.

4. What are some reasons you can think of for why inheritance might be preferable over composition? These do not have to be Java-specific.

One of the main reasons as to why inheritance might be more preferable could be due to the fact that some classes might have similar methods and instead of rewriting those methods, inheritance could be used. Another reason why inheritance could be more preferable might be to give some ordering to the classes and this way it will be easier to keep track of them.

Changelog

Product

- The attributes for product were changed to final to reflect that they do not change and as a result they were converted to all capital letters.

Inventory

- HashMap replaced the two parallel ArrayLists for the products and their stock
    - Each key is a product and its value is the product's stock
    - Chosen because accessing items in a HashMap is much faster than in an ArrayList
- A private method initializeInventory was added to make the process of initializing the inventory easier
- A new constructor was added to this class to initialize an inventory with a HashMap of products already in place. This constructor is especially used in the ShoppingCart method to create a cart with an empty hashmap.

StoreManager

- An HashMap attribute of carts was added to this class to keep track of all the carts and their id
    - Each key is an id that identifies the cart with and the value is the ShoppingCart object
- An attribute of id_counter was added to keep track of the id of carts which can later assign new IDs to each cart
- The method assignNewCartID assigns a new id to each cart and adds that cart to the HashMap of all the carts
- The methods getCarts, getInventory, and getAvailable products were added so the StoreView class can easily communicate with the StoreManager class without using any other class.
- The transaction method was replace with checkout method in this class which now gets a cart as its parameter and goes through all of its products and adds up the total price. It later displays the price for the user and removes the cart.

ShoppingCart

- This class inherits from the Inventory due to some of their similarities
- This class has an id attribute which keeps track of the cartId. There are getter and setter methods provided for this attribute.
- There is a getTotalPrice method provided that calculates the total price of a cart based on the price of Product and the number of each
- The constructor for this class takes an id attribute as its parameter and assigns it to the cartId attribute. It later calls the parent class with an empty HashMap.

StoreView

- Private StoreManager object to keep track of the main store
- Private cardID integer that's unique for every new cart
    - Obtained through StoreManager class
- Store Navigation:
    - Upon launch users are prompted to input which StoreView they would like to join
        - There are a set number of StoreView's, if the input is outside the range, user cannot join (default range set to 3 right now)
    - Users are then prompted to type in a command
        - 'help' lists out all commands and their uses
        - 'browse' lists all available items and their stock
        - 'addtocart' prompts the user to enter the product they want to add to their cart, as well as how much
            - The users are first asked to type out the name of the product and later the number of that product that they want.
            - Only after the proper name and the proper amount (less than total amount and >0) was entered their order is recorded and is added to their cart/
        - 'removefromcart' prompts the user to enter the product they want to remove from their cart, as well as how much of it
        - 'quit' will exit the store
        - Any other input is invalid and results in a 'Invalid command was entered' message, the user can then input a new command
    - Commands are not case sensitive
    - Users can always view their cart total at the top of the GUI
    - The emptyCart command is used to empty the cart of the user after the user enters quit. This is to return those products back to the inventory.
    - The findProduct method is used to find the product object using its name.
    - The displayGUI method is used to handle different instructions that are related to the commands. The program starts in Main and after a storeview is selected it repeatedly calls displayGUI so the user is prompted for a new command and the appropriate action is taken.
    - The displayItems method is used to display all the items in the inventory with their price and amount so they users can use them. This method is called with the commands 'browse' and 'addtocart'.