Milestone 1
Questions

Bardia Parmoun 101143006

Guy Morgenshtern 101151430

1. What is a constructor? When does it get used in Java?
When an instance of a class is first defined, its main attributes and parameters are initialized in a method called the constructor. This is a special method since it has no return type. Constructors can sometimes have input parameters which are used to initialize the attributes of the object but that is not always required.

2. Did you need to specify a constructor for the StoreManager class? Why or why not?
Since StoreManager only creates an inventory upon initialization, the default constructor with no parameters that Java provides can be used. This means no constructors need to be explicitly specified. However, we included one that takes in an existing Inventory in the case it's needed in the future.

3. Explain what a default constructor is.
When the programmer does not provide a constructor the compiler will provide a default constructor which initializes the attributes of the objects. It is not recommended to use a default constructor since defining initial values are often critical to how the object behaves.

4. What are object references and where are they used in this milestone?
An object reference points to a location in memory where the objects methods and attributes are held. They are used in the milestone every time a new object is created in the program. For example, in StoreManager the Inventory variable named "inv" is a pointer/reference to the data within it in memory.

5. Summarize the most important differences between an ArrayList, LinkedList, and Array. Which one did you use in this milestone and why? If you used something else, you must explain why as part of your Change Log.

Arraylist, Arrays, and Linkedlists are three examples of data collection tools. Each one of these tools has its features and limitations.

Arrays can store multiple variables of the same type in a single variable. Each item has an index and the value of the item can be accessed using that index. Arrays can have multiple dimensions to represent different things such as matrices. Some disadvantages with arrays that their size is fixed and must be set upon creation. It is also very inefficient to insert and remove specific elements in and from the array.

Arraylists are collections as well but they can provide more features compared to arrays. The main difference is the ability to shrink or grow. In other words, the size of the arraylist is not fixed and it can easily change. Secondly they also provide builtin methods to insert and remove elements at specific locations in the arraylist. The main disadvantage of using arraylists is that the processing of resizing arraylists or accessing elements in the middle of the arraylist is often timconsuimg. In addition, arraylists cannot hold primitive data types, but this can be bypassed by using wrapper classes such as Integer for int and Byte for byte.

LinkedList is a collection that contains other objects. It has the option of adding or removing elements builtin and its size can easily change upon demand. However, the main disadvantage when it comes to using linkedlists is accessing random elements from them. Since the elements are not indexed, in order to find a specific element, the programmer needs to loop through the entire linkedlist and find the specific index. Linkedlists also require extra memory in order to save the pointers and the linkage between the objects.

In this project, this team decided to use two arraylists to represent the type of the products and the number of each product available. These arraylists work in parallel and for index i, in the products arraylist the type of the product and in the stocks arraylist the number of that product which is available can be found. This makes it easier to efficiently access the required information for a specific product and later change that information.

6. What is encapsulation and how is it relevant to this milestone?

Encapsulation is a way to wrap attributes and methods in a class so that they can only be accessed by specific classes. The different attribute access modifiers are “private” which hides the attributes from any other class, “public” which allows access to any class, and “protected” which only allows inherited classes access.

In this milestone encapsulation is used to hide data within different classes. Every class variable is given the “private” modifier and is assigned a getter and sometimes a setter method to access the attribute. Those methods are assigned as “public” to give other classes access to them.

Change Log

Product Class:

- Setting all the main attributes to final since they cannot be changed after the product is created.
- Included three getters to get information about the main attributes of this class such as name, id, and price
- No setters were included since the values of these attributes cannot be changed after they are created
- Name is defined as String since it deals with the name of the product.
- Id is defined as integer since in this store every product is given a unique integer id to help distinguish them.
- Price is given a double type since prices are often floating numbers and double was chosen as the main type to avoid limitations on the range of the values.

Inventory:

- Two parallel ArrayLists were created to keep track of the products and their stock. Each index in the products list corresponds to its respective stock in the stock list.
- ArrayLists were chosen because they provide both fast iteration (used to check if a product is in the inventory) and fast random item access (used to access a product's information or stock when an index is found/given). ArrayLists also dynamically grow and shrink automatically allowing for Products to be added or removed without any overhead.
- A private method called "findProductIndex" was added that takes in a product and returns an int that is the product's index in the inventory arraylist.
    - If the product is not in the inventory list then it returns -1.

StoreManager:

- An attribute of type Inventory was created to represent the backing inventory of this class.
- Two constructors were created one with no input and one which takes an instance of the Inventory as its input parameter. This was included in case there is already an inventory in place and that should be used with the store (mainly for testing purposes)
- The checkStock method was added to check the number of products available in the inventory given the product.
- The makeTransaction class takes an array of orders as its input parameters and processes the total cost of the transaction. The return type of this class was set

to double as it deals with prices and those were defined as double in the product class.

- In the case that the transaction cannot be completed since there might not be enough of a specific program the function will return -1 for the total price to indicate that there was a problem with the transaction

Main (for testing):

- A main class was provided to test the functionality of the different classes.
- First an instance of the inventory is created as an example of a sample inventory which will be then attached to the StoreManager classes.
- Four sample products are created and are added to the inventory using the addStock function
- Multiple orders are created and tested with the makeTransaction methods. These orders test different things such as what happens if a customer orders a product that is not in the inventory or what happens if the order goes over the number of available products.
- At the end the number of products available for each product is displayed.

UML Diagram of Classes

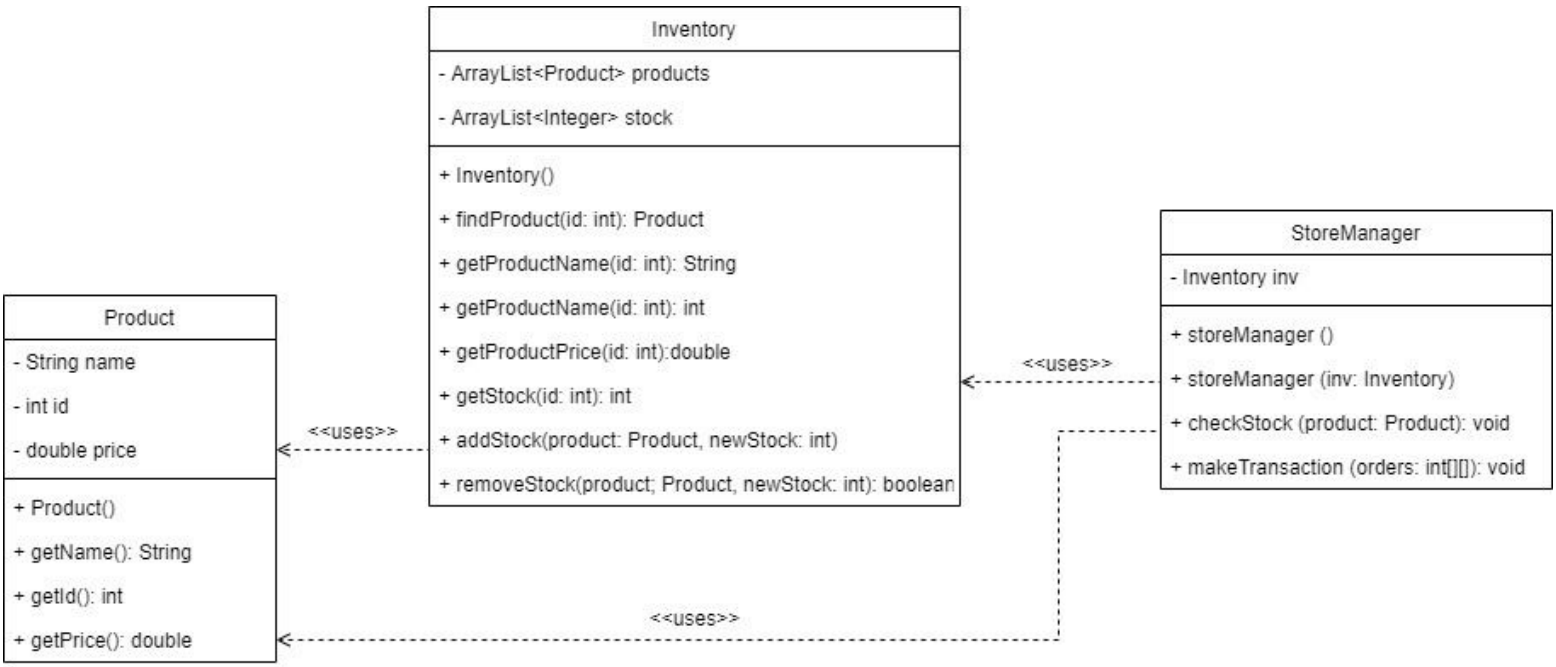

Figure 1: UML Diagrams of the main classes of this milestone