Milestone 5
Questions

Bardia Parmoun 101143006
Guy Morgenshtern 101151430

1. What is the difference between an interface and abstract class?

Interfaces serve as templates for classes and describe the behaviour that the classes that implement it must have. Abstract classes are used to describe common attributes and behaviours that subclasses will have. Abstract classes can contain instance variables that are inherited by derived classes. Interfaces cannot include any variables. A class can implement multiple interfaces but can only extend from one class (abstract or not). In addition, abstract classes have the ability to provide implementation but interfaces are restricted to only providing signatures.

2. Why we should “code against”/”program to” an interface (in relation to OOP)?

We should program to an interface because related classes who implement the interface will have templates for their methods. This allows for loose-coupling between classes and also provides a framework for communication between the classes. Since method names, parameters and return types are defined ahead of time, it is less likely that changing behaviour in classes will cause any damage. In addition, by using interfaces we can take advantage of multiple inheritance and we can we have a class implement multiple interfaces.

The only disadvantage of using interfaces could be the fact that there will be code repetition between multiple classes that use the exact same methods. In that case one class would implement the interface and the other classes extend that class.

Changelog

Inventory

- The methods addStock, removeStock, getProducts, getProductInfo, getStock, and findProduct were removed. The implementations of these methods were transferred to the abstract class productStockContainer.

- The attribute products was removed from Inventory as this is now a protected attribute in productStockContainer shared among Inventory and ShoppingCart.
- Inventory now extends ProductStockContainer

ShoppingCart

- ShoppingCart no longer extends Inventory and instead it extends ProductStockContainer.
- The constructor of shopping can now longer call super and instead it has to initialize the products attribute itself.

StoreManager

- All the function calls to the old Inventory methods had to be changed to their new names that are stored in ProductStockContainer:
    - addStock = addProductQuantity
    - removeStock = removeProductQuantity
    - getStock = getProductQuantity
- Two new methods were added to StoreManager to add some functionality to StoreView. These methods are specifically used to display the number of items in the cart and the number of different products.
- getNumOfItemsInCart returns the total number of items in the cart and getNumOfProductsInCart is used to return the total number of products that are in the cart.

StoreView

- The methods displayOrder and displayCart were modified to display the total number of items that are in the cart along with the total number of products. These methods take advantage of the methods getNumOfItemsIncart and getNumOfProductsInCart in the StoreManager.

ProductStockContainer

- Although this is an abstract class, it does not have any abstract methods and all of the methods in this class provide full implementation since the methods between Inventory and ShoppingCart are the exact same.
- This class is an abstract class that both Inventory and ShoppingCart inherit from.
- The reason we chose an abstract class over an interface is that both Inventory and ShoppingCart have some shared functionalities and for those functionalities they use the exact same functions with the exact same implementations. As a result, using an abstract class could help avoid repetition.
- The only attribute of this class is products which keeps track of the items in the inventory/cart along with their stocks.

- It has some methods that were previously in Inventory such as:
    - addStock = addProductQuantity
    - removeStock = removeProductQuantity
    - getStock = getProductQuantity
- The method getProductQuantity is used to return the total number of products in Inventory/StoreManager which is used in the getNumOfProductsInCart method in StoreManager.
- In addition to these methods we added the getProducts method which returns the hashmap that contains the information about the products and their stock in both Inventory and ShoppingCart.

Although ProductStockContainer is an abstract class, in the UML it is not marked with italics since it does not have any abstract methods. This was recommended by Professor Martin.