Felipe Lima - 109290055

Task 1- Diagram

TextUI	**calls**
TextUI	
MainMenu	
RouteChoice	DisplayInventory() from Store Items() from Store ItemMenu() from TextUI AddItemToCart() from Store DisplayCart() from Store CartItems() from Store RemoveItemFromCart() from Store Checkout() from Store ClearCart() from Store
ItemMenu	

ShoppingCart	**calls**
ShoppingCart	
AddItem	get_id() from ShoppingCart get_id() from Item IncreaseQuantity() from ShoppingCart
RemoveItem	get_quantity() from Item DecreaseQuantity() from Item
DisplayCart	
ClearCart	
get_items()	

Item	**calls**
Item	
get_id()	
get_quantity()	
get_cost()	
get_type()	
IncreaseQuantity	
DecreaseQuantity	
ToString	
Clone	
operator<<	

Store	**Calls**
Store	

DisplayInventory	
Items	get_id() from Item ToString() from Item get_quantity() from Item
CartItems	get_items() from Store ToString() from Item get_quantity() from Item
AddItemToCart	get_id() from Item DecreaseQuantity() from Item Clone() from Item AddItem() from Store
RemoveItemFromCart	get_items() from Store get_id() from Item RemoveItem() from Store IncreaseQuantity() from Item
DisplayCart	DisplayCart() from Store
Checkout	get_items() from Store get_cost() from Item get_quantity() from Item get_type() from Item ClearCart() from Store
ClearCart	ClearCart() from Store

Task 2- Clone() -

Clone is creating a new “Item” with all the same values for the fields as the “Item” used to call it. It is specifically used to create a new “Item” that will be added to cart_ called by the “Item” selected by the user. It is included in this class definition so when called it has access to all of the private variables in that class and is able to correctly create the clone of the “Item” object used to call it.

Trait	TextUI	ShoppingCart	Item	Store
cohesive (one single abstraction)	TextUI fulfills the cohesive trait. It focus only on displaying text user interface and the user's interaction with it.	ShoppingCart fulfills the cohesive trait. It has all the methods that related only to actions performed in the shopping cart.	Item fulfills the cohesive trait. It stores and modifies all the information necessary for each item of the store. Item has only this function.	Store fulfills the cohesive trait. It deals specifically with actions from the Store and although it has calls to other classes, it handles only the actions in the Store.
complete (provides a complete interface) - complete implementation	TextUI fulfills the completeness trait. All the text displayed for the user and the user interactions are handled by this class.	ShoppingCart fulfills the completeness trait. It covers all the functions of a shopping cart and all the actions a user can perform with the cart such as add or remove an item.	Item fulfills the completeness trait. It successfully edits and sores the information of each item and doesn't depend on any other classes to do so.	Store fulfills the completeness trait. It provides a complete interface integrating all classes and making the program work. It makes the actions from the user impact the store.
clear (the interface makes sense) – easy to understand	TextUI fulfills the clarity trait. Interface is clear and well organized. Each method has a specific function within the class.	ShoppingCart fulfills the clarity trait. Clear and well designed methods that clearly perform a specific function within the program.	Item fulfills the clarity trait. Interface and the function of each method is clear to the programmer.	Store fulfills (but could do better in) the clarity trait. Although the interface makes sense if you make an effort, the naming of functions can be confusing, for example the function DisplayCart() only calls DisplayCart()

				again but from a different class.
convenient (makes things simpler in the long run)	TextUI fulfills (but could do better) in the convenience trait. Although it makes the program simple to understand and easy to modify in the long run, it's not the most convenient for the user as the flow and way the information is displayed can be confusing.	ShoppingCart fulfills the convenience trait. Really easy to understand and to modify if necessary. Great on the long run.	Item fulfills the convenience trait. It makes it simple on the long run as each method has a very specific actinon and for every possible action dealing with Items, there is a method to do it.	Store fulfills the convenience trait. Since it makes things simple on the long run as methods have specific and unique jobs.
consistent (names, parameters, ordering, behavior should be consistent)	TextUI fulfills the consistency trait. Follows naming conventions and operations are consistent with each other. Although the order of the functions isn't the same on the .h and .cpp files.	ShoppingCart fulfills the consistency trait. Follows naming conventions and operations are consistent with each other. Although the order of the functions isn't the same on the .h and .cpp files	Item fulfills the consistency trait. Follows all naming conventions and each method is consistent with the other.	Store fulfills the consistency trait. Naming are consistent with each other, as well as parameters. Although the order of the functions isn't the same on the .h and .cpp files