Daniel Leone. CSCI 301-Section 54

StarID: mm8755tt

Due Date: Jul 3, 2023

Instructor: Dr. Jie Meichsner

1. **Design Document**

Introduction

This program implements a virtual ShoppingCart to allow a user to create a list of items for purchase. They are able to add, remove, and modify their order. The user will first add their selections into the cart. A list of their items will be displayed with the price total. They are able to adjust their cart after review of the current total by adding items, removing items, or modifying the quantities of items. The updated list and price total will display once modifications are complete.

Data Structures

Item is a class that contains data members that provide details about an individual item for purchase to include: name of type string, unitPrice of type double, and quantity of type int. String is an array of characters utilized to store name within the item class.

ShoppingCart is a class that inherits from class Bag<Item>. It contains one data member of type double that holds totalPrice.

Bag<ItemType> is a base class that contains an array of ItemType called items. In the context of this program, this array will be of type Item and will hold the contents of a ShoppingCart object. The array size is set by a constant integer of DEFAULT\_BAG\_SIZE. A vector of ItemType is utilized in a function that will copy the contents within array items and return the vector.

Functions

The main program utilizes nonmember functions for menu and display purposes.

* One module allows the user to initialize their ShoppingCart by continuously adding items until the user is done. This consists of one function.
* One module allows the user to add, remove, or modify the quantity of their ShoppingCart. This consists of one main function utilizing three sub functions. The sub functions provide the logic for “add”, “remove”, and “modify quantity.”
* One function displays the ShoppingCart contents and total price.
* One functions are provides a border for display purposes.

Item class provides functions to access and modify data members.

* Three getter functions retrieve values of the data members.
* Three setter functions modify values of the data members.
* Override of “operator ==” allows two Item objects to be compared. The comparison is of name and price.
* Override of “operator >>” and “operator<<” allows istream and ostream to utilize an Item object directly. These are friend functions to Item Class.

ShoppingCart class inherits multiple functions from base class Bag<Item>. It contains one class specific member function, and overrides two functions from class Bag<Item>.

* One getter function to retrieve the value of totalPrice.
* Overrides add function from Bag<Item>. It includes the addition of adjusting total price based on new Item price x quantity. Returns type bool to signal if the object was added or not.
* Overrides remove function from Bag<Item>. It includes the subtraction of adjusting total price based on new Item price x quantity. Returns type bool to signal if the object was removed or not.
* Inherits six additional functions from Bag<Item> that work on array items.

The Main Program

The program beings by initializing an empty ShoppingCart and displaying a welcome message to the user.

The program will provide the user with instruction on how to add items to the ShoppingCart. The add to cart menu will allow the user to continue to add items until he or she decides not to. When initial adding is complete, the contents and total price will be displayed for the user to review.

The program will provide the user the option to modify his or her cart at this time through the modify menu. The user can choose to add new items, remove current items, or modify the quantities of current items. The contents will be checked to see if the item is in the cart prior to removal or modification. The modification quantity will have to be a valid integer greater than 0 in order to modify. If these conditions are not met, then the user will receive a message informing them.

The program will display the updated contents and total price upon completion of modifying the cart. A “Thanks for shopping…” message will display before the program terminates.

**Structure Chart of the Main Program**

Main Program

menuAddToCart

displayTotal

menuModifyOrder

lineBorder

modifyAdd

modifyRemove

modifyChangeQuantity

**UML**

|  |
| --- |
| Item |
| -name : string  -unitPrice : double  -quantity : int |
| +Item() :  +Item(initName : string, initUnitPrice : double, initQuantity : int) :  +setName(updateName : string) : void  +setUnitPrice(updateUnitPrice : double) : void  +setQuantity(updateQuantity : int) : void  +getName() : string  +getUnitPrice() : double  +getQuantity() : int  +operator==(item2 : Item&) : bool  // FRIEND FUNCTION  +operator>>(ins : istream&, target : Item) : istream&  +operator<<(outs : ostream&, source : Item) : ostream& |

|  |
| --- |
| ShoppingCart |
| -totalPrice : double |
| +ShoppingCart() :  +getTotalPrice() : double  +add(newEntry Item&) : bool  +remove(anEntry Item&) : bool |

1. **Code List**

ads

1. **User Document**

The program project2 generates a virtual shopping cart for the user to add, remove, and modify items. Items are identified by their name and price, but not quantity. The user will begin by adding items to his or her cart. The contents and total price will be displayed. The user will be provided the option to modify his or her order. The updated contents and total price will be displayed when the user is done modifying his or her order.

The program’s name is **project2.cpp**. It is located at the following directory on **centOS**:

**/home/STCLOUDSTATE/mm8755tt/PROJECTS/PROJECT\_2**

To compile and link it, simply enter:

**g++ -o s\_cart project2.cpp**

To run the program, enter **s\_cart**, then follow the prompts to add items to the shopping cart. The items will be added in the format: *name unitPrice quantity*. The program assumes that the user will only try to add a maximum of six items. If the user goes over this limit, the items after item six will not be added.

The beginning of the program will look like this with example user input.

**prompt> s\_cart**

**Enter the item you selected as the following order:**

**name unitPrice quantity**

**(Name can not contain any space. Otherwise errors happen!)**

**-->T-shirt 19.99 2**

After the first selection, the user will be asked whether he or she would like to add another item simply with **y** or **n** as such. If **y**, he or she will provide another item. If **n**, the user will receive a display of his or her cart along with the total price:

**Want to continue y/n-->y**

**-->Sweater 39.99 2**

**Want to continue y/n-->n**

**Here is your order:**

**------------------------------------------------------------------------**

**Name Unit\_Price Quantity**

**T-shirt $19.99 2**

**Sweater $39.99 2**

**The total charge is $119.96**

**------------------------------------------------------------------------**

The user will be provided the option to modify his or her cart. When prompted, a **y** or **n** will allow the user to choose. If the user chooses **y**, the options to add, remove, or modify quantity will be presented with selection being numerical values **1** for add, **2** for remove, and **3** for modify quantity.

**Want to modify your order? y/n-->y**

**What do you want? Enter 1: add 2: remove 3: change quantity\n-->1**

If the user chooses to add an item to the cart, they will be prompted in a similar way as the start of the program.

**Enter the item you selected as the following order:**

**name unitPrice quantity**

**-->iphone\_case 25.50 3**

**The item has been added.**

The user will be prompted again asking whether he or she wants to modify the order.

**Want to modify your order? y/n-->y**

**What do you want? Enter 1: add 2: remove 3: change quantity\n-->2**

If the user chooses to remove an item, they will receive the following prompt. Note, that if the item does not exist or he user does not input the name and price properly, the user will be prompted with a message stating that the item is not in the cart. The user will once again be prompted to choose whether he or she wants to modify the cart.

**Enter the item to remove as the following order:**

**name unitPrice quantity**

**-->Sweater 29.99 2**

**No such item in your shopping cart!**

**Want to modify your order? y/n-->y**

**What do you want? Enter 1: add 2: remove 3: change quantity\n-->2**

**Enter the item to remove as the following order:**

**name unitPrice quantity**

**-->Sweater 39.99 2**

**The item has been remove.**

The user will be prompted again asking whether he or she wants to modify the order.

**Want to modify your order? y/n-->y**

**What do you want? Enter 1: add 2: remove 3: change quantity\n-->3**

If the user chooses to modify the quantity of an item, they will receive the following prompt. If the item is not in the cart, the user cannot modify the quantity of the item and will be returned to modify order selection. If the user tries to modify an item by a value of less than zero, they will be prompted to input a new value until a valid value is made.

**Enter the item to change as the following order:**

**name unitPrice quantity**

**-->T-shirt 19.99 2**

**Enter a new quantity --> -1**

**-1 is not a valid input.**

**Enter a new quantity --> 1**

**The quantity has been modified.**

If the user is done and chooses **n**,the updated contents and total price will be displayed and the program will terminate.

**Want to modify your order? y/n-->n**

**Here is your updated order:**

**------------------------------------------------------------------------**

**You have ordered the following items:**

**Name Unit\_Price Quantity**

**iphone\_case $25.50 3**

**T-shirt $19.99 1**

**The total charge is $96.49**

**Thanks for shopping in XXX SHOPPING CENTER.**

**------------------------------------------------------------------------**

1. **Test Data Plan**

ddad

1. **Summary**

In this project we implemented a virtual shopping cart that uses various prompts to allow a user to add, remove, and modify items through user input. It displays the contents of the cart and displays the total price.

The implementation focused on the concept of inheritance. ShoppingCart inherited from base class Bag<Item>. Many aspects of inheritance were required to properly execute the program. One example is utilizing base class functions through the subclass. Another example is overriding virtual functions to fit the more specific subclass. Having an understanding of the base class is very important to fully utilize the base class and maximize efficiency.

A secondary concept that was worked on was working with a template class. We did not create a template class, but we were required to work with one. Creating the Item class and having it function with Bag<ItemType> functions was not difficult, but it was important that criteria were met for them to work properly.

Including the implementation file of a class at the end of the header file was new for me. It allowed the main program (or any program that includes the header file) to include implementations by simply “including” the header file. This worked well for compiling and cohesiveness.

The current implementation can be improved. It makes some assumptions of the user. An improvement would be implementing better controls for user input, to keep the user bounded by the parameters of the program.