8.18 Main Lab 8 - Online shopping cart

Students:

This content is controlled by your instructor, and is not zyBooks content. Direct questions or concerns about this content to your instructor. If you have any technical issues with the zyLab submission system, use the "Trouble with lab?" button at the bottom of the lab.

Background

This was a Prep Lab and has been promoted, with some adjustments, to be the new main lab for chapter 8. This main lab extends the earlier prep lab "Online shopping cart". (You should save this as a separate project from the earlier prep lab). You will create an on-line shopping cart like you might for your on-line purchases. The goal is to become comfortable with setting up classes and using objects.

This is your first lab with objects so please review the Objects and Classes section of the style guide.

Requirements

Note: Both shopping cart labs do some of their testing using unit tests. A unit test is meant to test if a certain function works correctly. Thus, in a unit test we can call one of your functions directly with our own set of parameters without going through "main," and check to see if the tested function returns the correct result or accomplishes the desired task. In order for you to pass a unit test you MUST spell the functions exactly as in the specification and have your parameters in the same order as given in the specification. For testing class member functions, your other members must also be spelled, etc. as specified, and working correctly, because sometimes the unit test must call other member functions as part of testing the targeted function. If you have any of these types of errors the unit test will give no feedback (other than the fact the you didn't pass it). Hopefully this explanation will help you consider the things which you should debug in that case.

The points in parenthesis roughly match the order of the test cases you will submit against.

(1) Extend the ItemToPurchase class per the following specifications. Keep the same spellings for previous members as in the prep lab.

- Parameterized constructor to assign item name, item description, item price, and item quantity (default values of "none" for name , and 0 for price and quantity, as in the prep lab). Price is int for simplicity). (3 pt)

- Public member functions
  - SetDescription() mutator & GetDescription() accessor (5 pts)
  - PrintItemCost() - Outputs the item name followed by the quantity, price, and subtotal
  - PrintItemDescription() - Outputs the item name and description
- Private data members
  - string itemDescription - Initialized in default constructor to "none"

Ex. of PrintItemCost() output:

```
Bottled Water 10 @ $1 = $10
```

Ex. of PrintItemDescription() output:

```
Bottled Water: Deer Park, 12 oz.
```

(2) Create three new files:

- ShoppingCart.h - Class declaration
- ShoppingCart.cpp - Class definition
- main.cpp - main() function (Note: main()'s functionality differs significantly from the warm up)

Build the ShoppingCart class with the following specifications. Note: Some can be function stubs (empty functions) initially, to be completed in later steps.

- Default constructor (3 pt)
- Parameterized constructor which takes the customer name and date as parameters (5 pt)
- Private data members
  - string customerName - Initialized in default constructor to "none"
  - string currentDate - Initialized in default constructor to "January 1, 2016"
  - vector < ItemToPurchase > cartItems
- Public member functions
  - GetCustomerName() accessor

- GetDate() accessor
- AddItem()
  - Adds an item to cartItems vector. Has parameter ItemToPurchase, which it will insert into the vector cartItems. Does not return anything.
- RemoveItem()
  - Removes item from cartItems vector. Has a string (an item's name) parameter. Does not return anything.
  - If item name cannot be found, output this message: `Item not found in cart. Nothing removed.`
- UpdateQuantity()
  - Updates the quantity of item from cartItems vector. Has a string (an item's name) parameter and an parameter for the new quantity. Does not return anything.
  - If item name cannot be found, output this message: `Item not found in cart. Nothing modified.` If the quantity is set to 0, still leave the item in the cart, but treat it as a 0 in all counts, etc.
- GetNumItemsInCart() (4 pts)
  - Returns quantity of all items in cart. Has no parameters.
- GetCostOfCart() (4 pts)
  - Determines and returns the total cost of items in cart. Has no parameters.
- PrintTotal()
  - Outputs total number and cost of objects in cart.
  - If cart is empty, output this message: `SHOPPING CART IS EMPTY`
- PrintDescriptions()
  - Outputs each item's description.
  - no empty message required.

Ex. of PrintTotal() output:

```
John Doe's Shopping Cart - February 1, 2016

Number of Items: 8
```

```
Nike Romaleos 2 @ $189 = $378

Chocolate Chips 5 @ $3 = $15

Powerbeats 2 Headphones 1 @ $128 = $128



Total: $521
```

Ex. of PrintDescriptions() output:

```
John Doe's Shopping Cart - February 1, 2016



Item Descriptions

Nike Romaleos: Volt color, Weightlifting shoes

Chocolate Chips: Semi-sweet

Powerbeats 2 Headphones: Bluetooth headphones
```

(3) In main(), prompt the user for a customer's name and today's date. Output the name and date. Create an object of type ShoppingCart. Note that you have to declare (and thus create) your cart after you have input the name and date, since you have to set those with the parameterized constructor since you do not have setter functions for name and date. (3 pt)

Ex.

```
Enter Customer's Name: John Doe
```

```
Enter Today's Date: February 1, 2016

Customer Name: John Doe

Today's Date: February 1, 2016
```

(4) Implement the PrintMenu() function. PrintMenu() has a ShoppingCart parameter, and outputs a menu of options to manipulate the shopping cart. Each option is represented by a single character. Build and output the menu within the function. You will execute the different options from the PrintMenu function, rather than return the option.

If an invalid character is entered, continue to prompt for a valid choice. Hint: Implement Quit before implementing other options. Call PrintMenu() in the main() function. Continue to execute the menu until the user enters q to Quit. (7 pts)

Ex:

```
MENU

a - Add item to cart

d - Remove item from cart

c - Change item quantity

i - Output items' descriptions

o - Output shopping cart

q - Quit



Choose an option:
```

(5) Implement Output shopping cart menu option. (7 pts)

Ex:

```
OUTPUT SHOPPING CART

John Doe's Shopping Cart - February 1, 2016

Number of Items: 8



Nike Romaleos 2 @ $189 = $378

Chocolate Chips 5 @ $3 = $15

Powerbeats 2 Headphones 1 @ $128 = $128



Total: $521
```

(6) Implement Output item's description menu option.

Ex.

```
OUTPUT ITEMS' DESCRIPTIONS

John Doe's Shopping Cart - February 1, 2016



Item Descriptions (5 pts)

Nike Romaleos: Volt color, Weightlifting shoes
```

```
Chocolate Chips: Semi-sweet

Powerbeats 2 Headphones: Bluetooth headphones
```

(7) Implement Add item to cart menu option. (7 pts)

Ex:

```
ADD ITEM TO CART

Enter the item name: Nike Romaleos

Enter the item description: Volt color, Weightlifting shoes

Enter the item price: 189

Enter the item quantity: 2
```

(8) Implement remove item menu option. (10 pts)

Ex:

```
REMOVE ITEM FROM CART

Enter name of item to remove: Chocolate Chips
```

(9) Implement Change item quantity menu option. (12 pts) - More points since the tests includes print descriptions.

Ex:

```
CHANGE ITEM QUANTITY

Enter the item name: Nike Romaleos
```

```
Enter the new quantity: 3
```

Items Evaluated by the TA's Inspection (25 points)

- Compliance with the style guide. in particular consult the section on Objects and Classes.
- You must create each of the classes and members as specified. Where given, use the exact spelling of function names, as that is necessary in order for unit tests to function properly.

Instructions

Deliverables

`ShoppingCart.cpp`
,
`ShoppingCart.h`
,
`ItemToPurchase.h`
,
`main.cpp`
and
`ItemToPurchase.cpp`

We will expect the above file(s) to be submitted

Compile command

```
g++ ShoppingCart.cpp main.cpp ItemToPurchase.cpp -Wall -o a.out
```

We will use this command to compile your code

Submit your files below by dragging and dropping into the area or choosing a file on your hard drive.

ShoppingCart.cpp

ShoppingCart.h

ItemToPurchase.h

main.cpp

ItemToPurchase.cpp