Holly Buteau

Assignment 2

Design:

Class Item

Private:

String itemName

String unit

Int numToBuy

Double UnitPrice

Double total

Int array Size

Int counter

Item item Array

Public:

List()

~List()

Void getItems()

Double calculateTotal ()

Void displayArray()

Void removeItem()

getItems:

*for loop* to ask for user input to get all items to fill array and fill the array

calculateTotal

*for loop* to cycle through array and get the extended price, then add up the total.

displayArray

*for loop* to display contents of array

voidRemoveItem

prompt user to name the item to be removed. Remove from array with *for loop* to move every element of array over by 1.

Class Item  
Private:

String itemName

String unit

Int numToBuy

Double unitPrice

Public:

Item()

Item (string, string, double, int)

Void setItemName(string n)

Void setUnit (string u)

Void setUnitPrice (double p)

Void setNumToBuy (int q)

String getItemName ()

String getUnit()

Double getUnitPrice ()

Int getNumToBuy ()

== operator? Not sure yet

setItemName

itemName = n;

setUnit

unit = u;

setUnitPrice

unitPrice = p;

setNumToBuy

numtoBuy = q;

getItemName

return itemName

getUnit

return unit

getUnitPrice

return unitPrice

get numToBuy

return numToBuy

==operator

Not sure yet, need to read more

Reflection:  
Quite a bit changed from my initial design. Most prominently was that I wasn’t sure how to implement the overload operator. I had to read quite a it and check the discussion forum to get a decent understanding of how to do this. Another change I made was that I didn’t understand the proper syntax to create the itemArray. I had initially created a regular static array with a singular pointer. I kept getting errors in my code and I wasn’t sure how to make it work. I sat in on a Q&A session with a TA and he shed some light on my problem. I needed to initially create a dynamic array with a double pointer. Then I could easily transfer objects from one to the other. Another change I made was that I created a menu. I was initially asking the user to fill the array and to press a button to stop filling the array. That became a huge headache of syntax errors and core dumps. At that point, I knew it would be easier to have a menu, let the user place one item in the array, and bump them back to the menu to choose again. That also made debugging easier for me because I could see where the breakdown was in my code. I also added a boolean to my getItems method to easily check to see if items being entered already existed. Another big change that I made was that I wasn’t prepared for the removeItem method. I thought I could just take in the name of the item the user wanted to enter. That proved to be incredibly difficult. I tried over and over but I had trouble accepting and comparing strings. At that point, I asked the user for the index of the item to be removed. Those were by far the biggest changes I made to the program. Most of the Item class remained the same with the exception of the overload operator. I did change quite a few things in the List class, but a lot of that was variables that I realized would be helpful to the program, such as the array size and counter to keep track of how many variables were in the array. I learned quite a bit from this program. It helped solidify and expand my knowledge of pointers and I have a better grasp of the overload operator.

Test:

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Input Values | Expected Outcome | Observed Outcome |
| User can input an item | Item (“apples, bunches, 3, 5) | Information is displayed as:  Item Name: Apples  Unit: Bunches  Number to Buy: 3  Unit Price: 5 | Information is displayed as:  Item Name: Apples  Unit: Bunches  Number to Buy: 3  Unit Price: 5 |
| User can input strings with spaces | “Potato Chips” | Strings as displayed with all letters present, nothing is cut off or omitted. | Strings as displayed with all letters present, nothing is cut off or omitted. |
| User can remove items from array | Prompt:  “Give the index of the item to be removed”  Choice:  1 | Item is successfully removed and list does not display it or calculate its price in the total | Item is successfully removed and list does not display it or calculate its price in the total |
| Array is displayed correctly | Item (“apples, bunches, 3, 5)  Item (“bananas”, bunch, 2, .50) | All information is displayed with labels. | All information is displayed with labels. |
| Extended price is displayed correctly | Item (“apples, bunches, 3, 5) | Information is displayed as:  Item Name: Apples  Unit: Bunches  Number to Buy: 3  Unit Price: 5  Extended Price: 15 |  |
| Total price is displayed correctly | Item (“apples, bunches, 3, 5)  Item (“bananas”, bunch, 2, .50) | Information is displayed as:  Item Name: Apples  Unit: Bunches  Number to Buy: 3  Unit Price: 5  Extended Price: 15  Item Name: bananas  Unit: bunch  Number to Buy: 2  Unit Price: .50  Extended Price: 1  Total Price: 16 | Information is displayed as:  Item Name: Apples  Unit: Bunches  Number to Buy: 3  Unit Price: 5  Extended Price: 15  Item Name: bananas  Unit: bunch  Number to Buy: 2  Unit Price: .50  Extended Price: 1  Total Price: 16 |