Peter Moldenhauer
moldenhp@oregonstate.edu
CS 162 – 400
Assignment 2

Assignment 2 Reflection Doc

Design description:

The goal of this assignment is to build a shopping list program that allows the user to add and remove items to a list as well as print out the list contents to the computer screen. Each item is required to have a name, a unit type, a quantity desired and a unit price. The display of the list needs to contain all of the user added information for each item (the four properties just mentioned) as well as the extended price of each item (quantity * unit price). In addition, the list must also contain and be displayed with the total price of all of the items on the list.

There are also a few other specific requirements to be aware of when building this shopping list program. First, the user should not be allowed to add an item that is currently already on the list. Also, the size of the list should begin with a total of only 4 possible items to be added to it. If the user chooses to add more than 4 items, the list should dynamically increase in size to compensate for the additional item(s) the user enters. Last, if the use attempts to remove an item that is not on the list, the user should be notified that this item is not on the list.

The below bulleted list gives a quick visualization of what the problem description and requirements entail:

- Implement a grocery list program
    - Needs the following attributes
        - Item name
        - Unit of measure
        - Quantity
        - Price per unit
        - Total price
    - Needs the following actions
        - Add item
        - Delete item
        - Print out list

In order to succeed in solving and completing this problem for the assignment, the problem description must be first broken down to identify the potential classes to be used. After picking apart the problem description into the nouns and refining the list of nouns, I decided to use two classes for this program: Item and List.

The item class will contain attributes for name, unit, quantity, price per unit and total. It's default constructor will initialize all of these data fields to either being blank or a value of 0. There will also be get and set methods for each of these data fields which will be called when the user wants to set/get each specific field in the program. This class will also have a calculate total function which calculates the extended price of each item (quantity * unit price) based on user input.

The list class will be responsible for all of the actions that go into creating and editing the list. To start, the list class will have a private array of Items (item objects) which starts at the capacity of 4 items. The list will also have functions to add items to the list, remove items off the list and to print out the contents of the list to the screen. There will also be a function to increase the size of the array of items if the user inputs over 4 items. Since this new array of larger size will have to be dynamically created, the list class will also have to contain a destructor that deallocates the memory for the array before the program ends.

The below list gives a quick visual representation of the classes, data members and functions:

- Item
    - Data Members
        - Name
        - Unit
        - Quantity
        - Price per unit
        - Total
    - Functions
        - Constructor
        - Set/get name
        - Set/get unit
        - Set/get quantity
        - Set/get price per unit
        - Calculate total

- List
    - Data Members
        - Item Array
        - Array size
    - Functions
        - Constructor
        - Destructor
        - Add item
        - Remove item
        - Print list

The primary responsibilities for my main function is to create a List object and to display a menu of options to the user. In this menu, the user will input what actions he/she would like to take. Based on whatever the user inputs will determine what list function will be called to modify the list accordingly.

Below is a bulleted list to visualize what the main function will be responsible for:

- Main function
    - Create list object
    - Display menu to user
    - Call appropriate list function based on user input.

Design changes and analysis:

Overall, there were not too many parts of the original design that had to be changed. One was how the output was handled. Initially, I planned to print out each one of the item data members individually for each one of the items in the array in the List class. Later I realized that it would make more sense to create a print function within the item class. Then, I would just need to call that print function for each item in the array.

There were a few other functions that I decided on adding throughout the process. Some of these functions were data validation functions which were called after every user input. These functions would test to see if the input was appropriate and if it was not then it would loop to re-prompt the user to enter the data again. In the list class, I also added a total cost function as well as a find item function. The total cost function looks at each item in the array and then adds up the extended cost among all of the items in the array. The find item function is used to loop through the array of items to locate a specific item by its name.

Once I successfully built the main function, item class and list class and the program was running properly, I then added an operator== overload function in the item class. This function was then used to determine if a name that the user enters is the same as an item that is already in the item array. If the names are similar, a new item of the same name would not be added to the item list.

Test plan and test results:

For being a fairly small program, there are a lot of ways this program could crash or not run properly between user input and the different function calls. In general I wanted to be sure to have functions that would validate all of the user input in the program. If I know that the program does not let the user input data that would be problematic for the program and there are still errors in the program, then the problems are due to how I am calling the functions. In the process of coding the program it is also important to use an incremental development approach. I will build a portion of the code and test it right

away instead of waiting to test after the entire program is built. When the program is entirely built, I need to test each of the activities that take place in the program. These activities include: the menu in the main function, item objects added to array, item objects are built (instantiated) properly, item objects removed from array, items are not added to the array if there is already an item with the same name on the array, all items are printed out, the total price of all items in the array is accurate.

The following chars show the various test cases, input, expected outcomes and observed outcomes of the testing process.

Testing menu selection			
Test case	**Input values**	**Expected outcome**	**Observed outcome**
Character entered	'a'	Print error and prompt again	As expected
String entered	"Hello World"	Print error and prompt again	As expected
Input negative or 0	0, -17	Error and prompt again	As expected
Input number too high	8	Error and prompt again	As expected
Select add item	1	Prompt user to enter item name	As expected
Select remove item	2	Prompt user to enter item name	As expected
Select print list	3	Print list, or print error if no items are in the list	As expected
Select exit	4	Program exits	As expected

Testing add item function			
Test case	**Input values**	**Expected outcome**	**Observed outcome**
Name contains spaces	"coffee beans"	Program continues no error message	Input only saved first word, need to fix
Name contains no spaces	"soup"	Program continues no error message	As expected
Name is blank	" "	Program continues no error message	As expected
Unit contains spaces	"half pound"	Program continues no error message	As expected
Unit is blank	" "	Program continues no error message	As expected
Quantity is a string	"test"	Error message	As expected
Quantity is a letter or character	'a' '&'	Error message	As expected

Quantity is 0 or negative number	0, -19	Error message	As expected
Price is a string	“test”	Error message	As expected
Price is a letter or character	‘g’ ‘%’	Error message	As expected
Price is negative number	-3	Error message	As expected
Add first item to list	“beer”,“can”, 4, 1.29	Successful add message displayed	As expected
Add second item to list	“eggs”, “carton”, 2, 3.45	Successful add message displayed	As expected
Add item with same name an item already on list	“eggs”	Error message, item not added to list	Item was still added, need to fix

Testing remove item function			
Test case	**Input values**	**Expected outcome**	**Observed outcome**
Enter name of item not on list	“Computer”	Error message	As expected
Enter name of item when list is empty	“Door”	Error message	As expected
Enter name of item on the list	“eggs”	Confirmation message	As expected
Enter bad data, numbers or characters	7, -89, 1.3, ‘&’	Error message	As expected

Testing print list function			
Test case	**Input values**	**Expected outcome**	**Observed outcome**
Print list after item is added	Enter data for add item, select print list option	List displayed with appropriate item(s)	Items displayed but format is bad, need to fix
Print list when list is empty	Select print list option without entering in item data	Error message	As expected
Print list after item is removed from list	Remove item and select print list option	List displayed with the appropriate item removed	As expected
Print list and check that total list price is accurate	Enter data for items and select print list option	List displayed with all appropriate items along with correct total price of items	As expected

Problems encountered and how they were solved:

There were a few problems that I encountered throughout the course of building and testing my program. The first problem that I had an issue with is getting the program to handle spaces in names or units properly. Initially I just used cin to get all of the data but over the course of testing I realized that only the first words of names with multiple words were being stored in the variables. I was able to fix this issue through the use of the getline() function.

I hand another issue with figuring out how to properly code the operator overload function so that an item would not be added to the list with the same name as any items on the list already. Initially I tried to get around this by simply adding the item to the list and deleting it right away if there are duplicate names in the array. However, after rereading the assignment description again and realizing that I needed to overload the == operator and do this BEFORE an item is added to the array, I had to change up my program. I had to read section 11.13 in the book a few times before fully understanding how to accomplish this task with operator overloading. Soon, I was able to code my program by creating a separate object outside of the array when the user enters a name. The operator overload function then compared the separate object with each of the current objects in the array list to determine whether or not the item should be added to the array.

The last issue that I had was to format the output properly so that it was neat. From the start I wanted to display all of the items in the list in rows and columns neatly but this was a lot easier said than done with the text in the command window. I initially just used spaces between each category (name, unit, price, etc.) but the formatting would get messy as soon as I entered a name that was too large or had spaces. I soon figured out that if I used the tab character \t in my print statements that this would work a lot better than spaces. With the use of tab, I am able to allow a large space for a given item name to be entered without messing up the formatting of the rest of the categories to be displayed.