CS162 Assignment 2

Shoshana Abrass
abrasss@onid.oregonstate.edu
2015/02/01

Discovering requirements

As it happens we had a very similar assignment last quarter in CS161 (carLot.cpp), so I ended up using that same general approach here. The Items are stored as a vector in a List object, and the main program loop handles user input such as menu-action choices and capturing prices and quantities.

Design

main.cpp

```
List.myItems(); // initialize a vector of Items
read file into list; warn and proceed on error

do {
        menu of options: Add, Edit, Delete, Save, Quit
}
while{menu choice is not quit}
```

List.cpp

vector<Item> myItems;

List.addItem()

push_back Item onto vector

List.deleteItem()

List.printAll()

```
printColumnHeaders
for Item in List vector
        Item.printItem()
```

List.changeItem(index)

List.saveAll()

Item.cpp

string Name;

```
float Price;
int Qty;
string Unit;

Item.getName()

Item.getUnit();

Item.getQty();

Item.getPrice();
```

Reflection

The most interesting thought process this week was around what to put in the class and what to put in main.cpp. For example, all of the menu code is in main.cpp. Because of this, the changeItem() function is implemented in main.cpp instead of in the List class (addItem and deleteItem are both in List::). Maybe I should have named it something different. changeItem() collects information about how to modify the item, then calls the appropriate List:: function, changePrice() or changeQty() or deleteItem().

Early on I made a design decision not to mess with multiple filenames or cart names; the filename is hardcoded. Other people posted that they wrote support for multiple carts, and that's interesting, but I was already in the middle of coding something new to me: The logic that prompts the user to save the data before quitting **only if the data has changed** since the last save. This wasn't part of my initial design as you can see above, and it took me a few iterations to get it right. Finally I created a boolean as part of the List:: instance, because it makes sense to attach this value to a given list across multiple functions.

I briefly thought of making this value a 'static' member variable, since we read about those, but static variables apply to all created instances of the class. If I were to have multiple List:: instances, each instance should have it's own needToSaveData boolean.

Testing

NB: There are driver functions for fully testing the List:: class in the test.cpp file.

```
make test
```

to compile and run these tests.

Testing the Item:: class

By design, all input validation and error reporting is implemented in the List:: functions. We created test drivers for these cases however:

Spaces, numbers, weird chara product name
Backslash in product name
Negative quantity
default constructor
constructor with arguments

Price is zero
Qty is zero

Testing the List:: class

test case	input values	expected result	observed result
addItem()	n/a		*works as expected*
printItem()	n/a		*works as expected*
printAllItems()	n/a		*works as expected*

Testing main.cpp

test case	input values	expected result	observed result
Bad input filename	file does not exist	warn and continue	*Error is not implemented, silently continues*
Bad input file	file does not have read permission	error and quit	*Error is not implemented, silently continues*
Bad menu choice	Z	"unrecognized, try again"; back to main menu	*as expected*
Bad menu choice - edit an item when the list is empty	E	"List is empty"; back to main menu	*as expected*
Bad menu choice - Edit menu	bad line item: 101 bad letter option	error; back to main menu	*as expected*
Bad Qty - Edit function	-5	warning; leave quantity unchanged	*as expected*
Bad Price - Edit function	-10.99	warning; leave price unchanged	*as expected*
Menu choice: Add			*works as expected*
Menu choice: Change			*works as expected*
Menu choice: Delete			*works as expected*
Menu choice: Print			*works as expected*
Menu choice: Save			*works as expected*
Menu choice: Quit		Ask to save first, if data has changed	*works as expected*
EditMenu choice: Price			*works as expected*
EditMenu choice: Qty			*works as expected*