**Cash Register Application**

Robert Skelton

COMP-2710

Lab 4

11.22.13

**Part 1: Analysis**

The following are the use cases for the Cash Register Application:

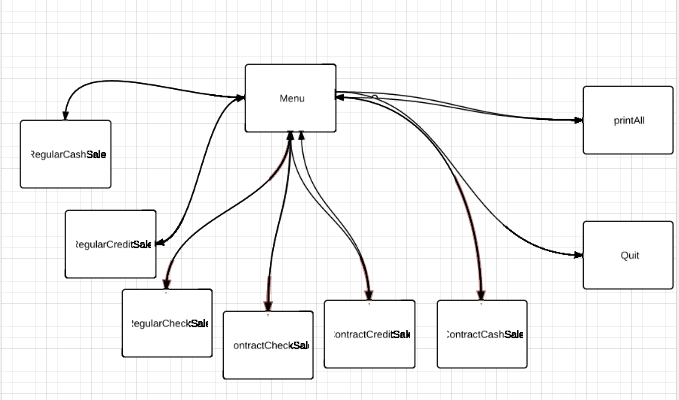
The use cases could include all of the types of inputs given when prompted for certain information, such as the customer name, credit card number and expiration date, the name of the item you are ringing up, the price of the item you are ringing up, the drivers license number of the customer, the amount of cash given to pay for your purchase, and your contractor ID number. All of these inputs are not type checked, so if you enter in a string when it asks for an integer, such as when prompted for the contract ID number, the program WILL crash. For instance, I typed in “hello” for the contract ID number, and the program looped asking me for the item name and price. Do not do that. To end entry of inputting items and prices, type a single asterisk, “\*” when prompted for the item name. If you enter a “\*” for the price, you will kill the program. Don’t kill the program.

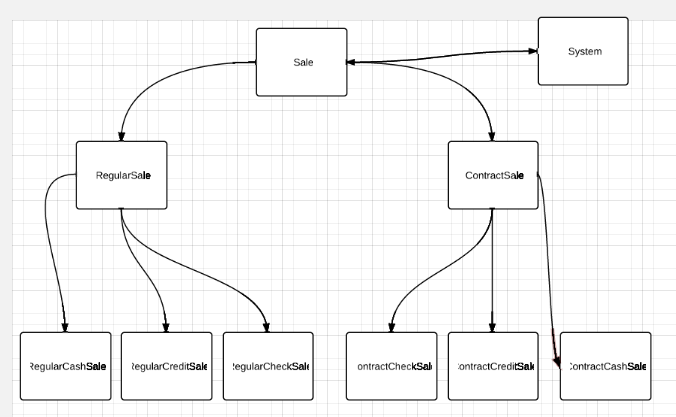
* **Part 2: Design**

The following will be the main classes that could be implemented into the program. Likely data values and functions will be listed within each class.

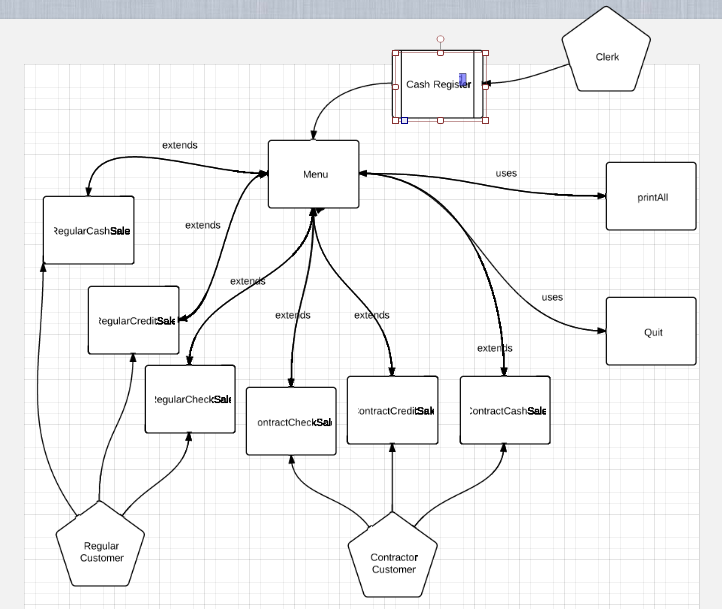
* System class: The System class contains functions that will help with the organization of the program.
* Constructor System: Creates a new System object.
* Function quit: Ends the program gracefully.
* Function menu: The main menu of the program, where other functions are called. This is driven by a switch case with an integer.
* Function mask: Masks a credit card, turns all digits except the last 4 into X’s.
* Sale class:
* Constructor Sale: Creates a Sale object and defines the tax rate.
* Function IntToString: There are two versions of this function. One takes a double, and the other takes an int, and they are morphed into a string, which makes writing to the vector easier the way that I implemented it.
* double taxRate: Set at 8 percent.
* double discount: Set at 15 percent, but only for Contractors. Nice deal!
* vector SalesList: holds a list of all the Sales, which can be printed out whenever requested with option 7.
* function printAll: Prints all the sales, used with option 7.
* RegularSale Class:
* RegularSale constructor: Inherited variables from Sale, and helps define three other RegularSale offshoot classes.
* ContractorSale Class: Inherited variables from Sale, and helps define three other ContractorSale offshoot classes.
* ContractorSale constructor: Inherited variables from Sale, and helps define three other ContractorSale offshoot classes.
* ContractorSaleCash Class:
* ContractorCash function: Runs everything needed when a contractor pays with cash.
* ContstuctorSaleCash constructor: empty constructor, just used to create the object.
* ContractorSaleCredit Class:
* ContractorCredit function: Runs everything needed when a contractor pays with a credit card.
* ContractorSaleCredit constructor: empty constructor, just used to create the object.
* ContractorSaleCheck Class:
* ContractorCheck function: Runs everything needed when a contractor pays with a check.
* ContractorSaleCheck constructor: empty constructor, just used to create the object.
* RegularSaleCash Class:
* RegularCash function: Runs everything needed when a non-contractor pays with cash.
* RegularSaleCash constructor: empty constructor, just used to create the object.
* RegularSaleCredit Class:
* RegularCredit function: Runs everything needed when a non-contractor pays with a credit card.
* RegularSaleCredit constructor: empty constructor, just used to create the object.
* RegularSaleCheck Class:
* RegularCheck function: Runs everything needed when a non-contractor pays with a check.
* RegularSaleCheck constructor: empty constructor, just used to create the object.

The Class Diagram for my implementation of Automatic Maze Path Finder is shown below.



A basic Data Flow Diagram can be found below.

A Basic Use Case Diagram can be found below



**Part 3: Testing**

The following are test cases are for the system at large. These test cases will be tested after I implement the Cash Register program in the C++ programming language. The screenshots from all of the tests is shown below all of the specific test cases, and includes all of the test cases. I had to do more screen shots after I implemented the credit card mask, so in the first section of screenshots, ignore the credit card purchases. They are corrected in the second set of screen shots.

Some possible test cases include:

1) Regular Check, Item: Porsche, Price $88000, Customer: Robert Skelton, Drivers License No: 123988. Yeah, I bought a Porsche with a check. Then print all.

2) Regular Cash, Item: Ferrari, Price $900. So Cheap. Paid $1000 for it. Then print all.

3) Regular Credit Card, Item: Saw, Price $15, Item: Drill, Price $55, Customer Name: John Smith, Credit Card number: 12398210

Expiration Date: 01/15, then print all.

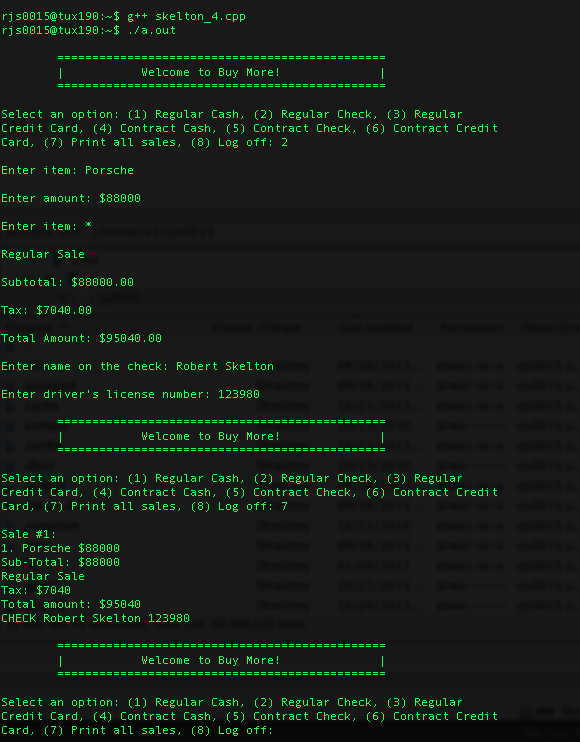
4) Contract Check, Contractor ID: 1239780, Item: Wood, Price $2000. Lots of wood. Name on check: Home Depot, Drivers Licenser number: 123980. Then print all.

5) Contract Credit, Contract ID: 1239807, Item: Truck, Price $20000, Item: Grill, Price, Item: Charcoal Price $1500, Price: $10.50, Name on Credit Card: Chicken Man, Credit Card Number: 1320801, Expiration Date: 01/20, then print all.

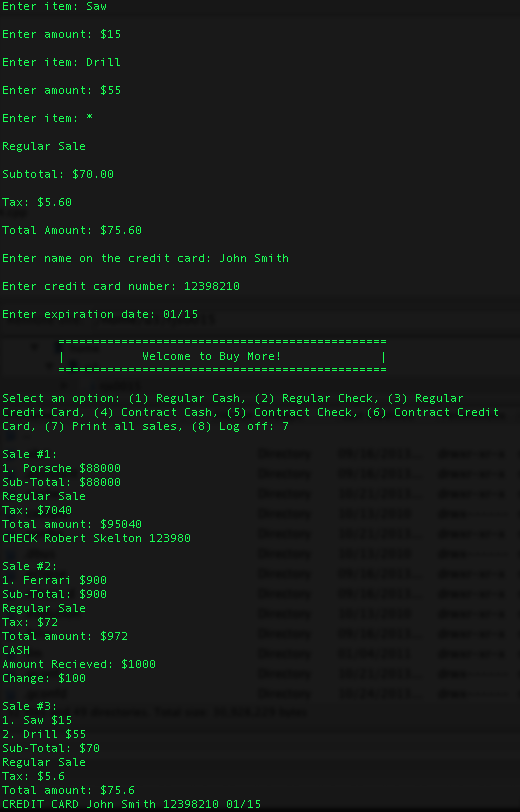
6) Print all again, and it should show all the previous entries in a correctly formatted output.

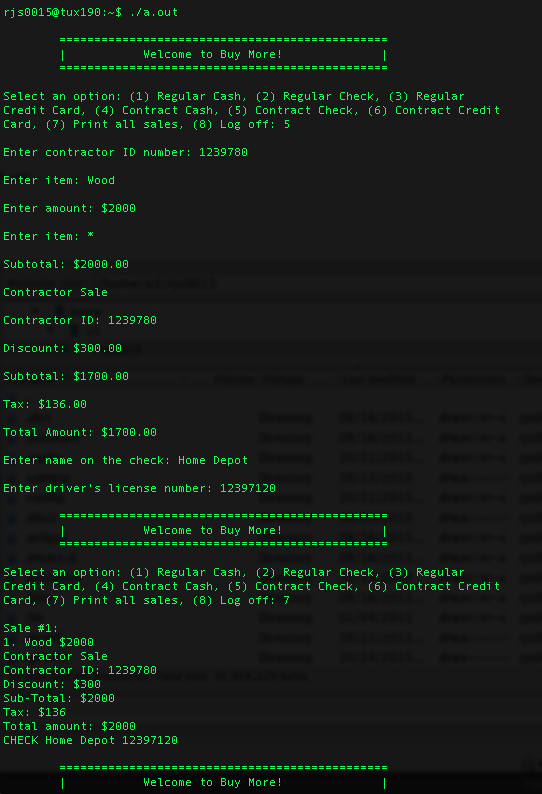
**Specific Test Cases**

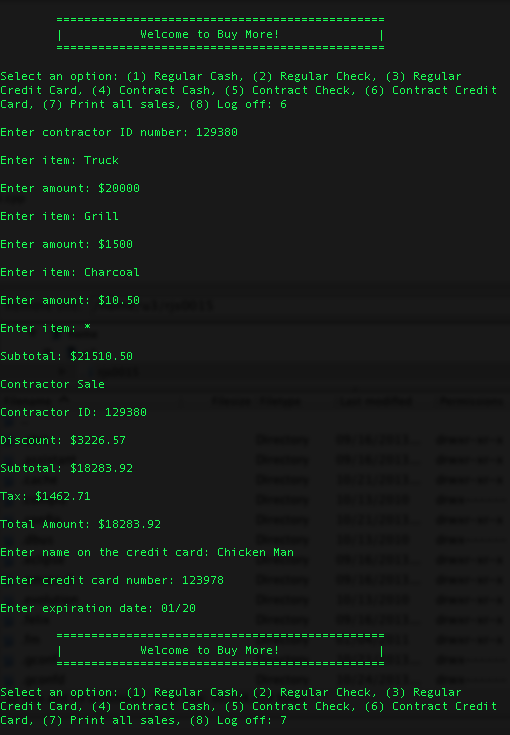
Before Mask implemented (Lim took a while to email me back)

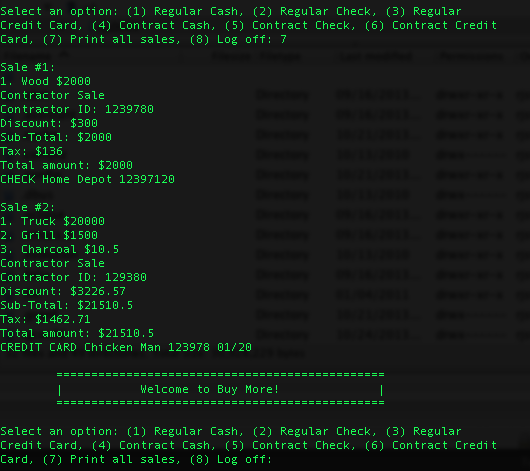
****

****

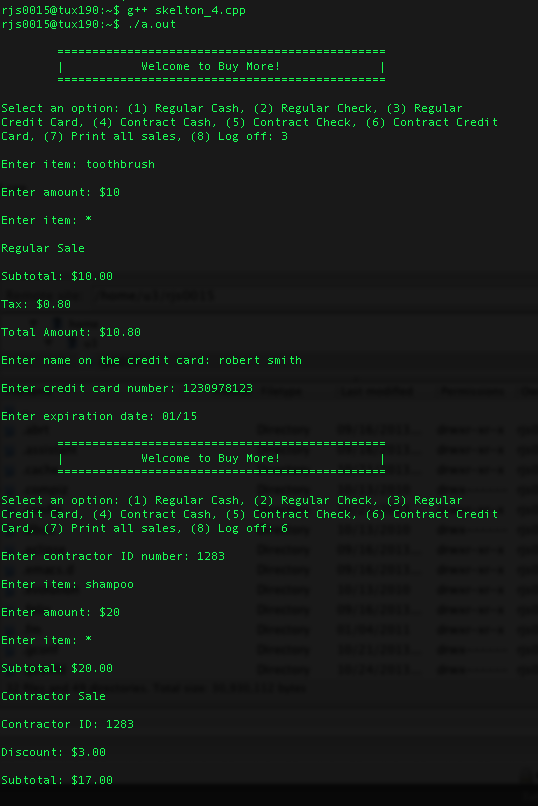


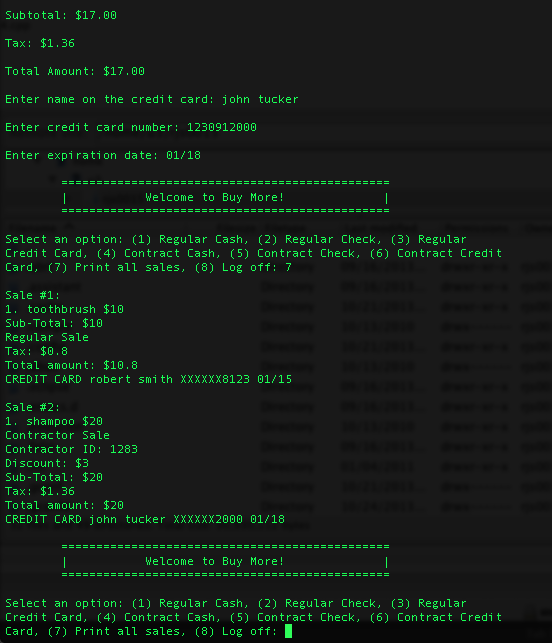


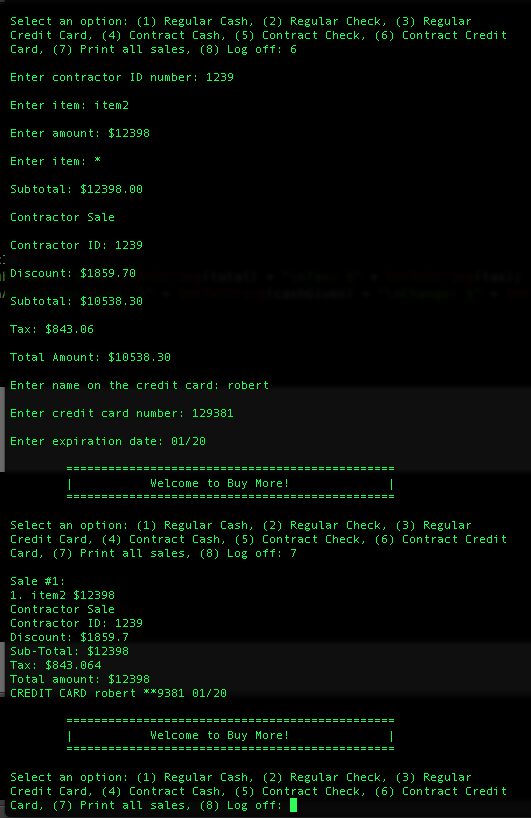


****

After mask implemented:





****

**Conclusion**

The most difficult part of this project was formatting, and it was just tedious. The virtual functions were helpful, and made using different attributes in new objects much easier. This is the most useful program I have written to date. I was having trouble getting customer’s names when using a credit card or a check, because I wasn’t accounting for spaces. I fixed this by using two getline() in a row, and that solved my issue. Also, I was unsure of how to format everything, and unsure about many of the directions since the example program given was so much different than what the write up implied to do, so I did not have a way to get Contractor ID, or mask the credit card number initially. This was not difficult to fix, however. I also thought that you changed the credit card mask to an X, and it was supposed to be a \*. I changed that in my final screen shot, and it works in the final version of my program.

-Robert Skelton