CSCI 140 -- Project 4 (Vending Machine System II)

Must present and turn in Tuesday, 12/06/2016 (TTh Section)
Must present and turn in Monday, 12/05/2016 (MW Section)

Problem Statement

A company intends to build a vending machine system and it wants you to develop a customized object-oriented software for its system. This company currently has plan for four different models (100A, 100B, 100C, and 100D) and it is planning to add several more in the future. A customer can utilize one or more models by setting up a data file. Each machine shall be able to keep track of its inventory, money (if applicable), and transactions. Inventory for each system shall consist of a list of product items and a quantity for each. Money for each system shall consist of number of nickels, dimes, quarters, and dollar bills if applicable (no pennies). Each transaction shall consist of a product item, paid amount, and changes (if applicable). We will assume that the cost for each item is a multiple of 5 cents.

Model 100A accepts one-dollar bill only and it give back changes using coins as well as utilizing a lower denomination if needed (similar to project 1 and you can ignore special case like Q = 2, D = 4, N = 0, and you need to give back 55 cents). Model 100B accepts coins and one-dollar bills, but it returns changes in coins only and it utilizes lower denomination including special case like Q = 2, D = 4, N = 0, and you need to give back 55 cents if needed (similar to project 1 with extra credit 1 and 2). Model 100C accepts credit card payment only and it automatically rejects a purchase after two invalid credit card entries. See below for requirements in validation a credit card. Model 100D accepts coins and one-dollar bills (like 100B) and credit card (like 100C). In general, an invalid transaction will not be processed and money will be returned if applicable. **You are expected to provide an implementation for model 100A and model 100C at this time.**

All Credit Card numbers follow certain patterns. A credit card must have between 13 and 16 digits and it must start with a 4, 5, 37, or 6. In 1954, Hans Luhn of IBM proposed an algorithm for validating credit card numbers. This algorithm is useful to determine if a card number is entered correctly or if a credit card is scanned correctly by a scanner. Almost all credit card numbers are generated following this validity check, which can be described as follows:

1. Double every second digit from right to left. If doubling of a digit results in a two-digit number, add up the two digits to get a single-digit number. For example, if a particular second digit were 4, then computing that digit would be 4 x 2 = 8. If a particular second digit were 6, then computing that digit would be 6 x 2 = 12 → 1 + 2 = 3.

2. Now add all single-digit number from Step 1.

3. Add all digits in the odd places from right to left in the card number.

4. Sum the result from Step 2 and Step 3.

5. If the result from Step 4 is divisible by 10, the card number is valid. If the result from Step 4 is not divisible by 10, it is invalid.

You must use Luhn's algorithm to determine if the Credit Card number is valid or not.

- Example valid Credit Card number: **4388576018410707**
- Example invalid Credit Card number: **4388576018402625**

The vending machine system will load all machines with data from input data files (machines.txt and products.txt) upon a secret code is entered by an operator. Once an operator initiates the system with a special code, customers will be able to make a purchase by selecting a machine first, enters a purchase amount, and then selects an item. After the system is in use and after a correct secret code is entered, the system will generate a detailed report for each machine in an output file, **reports.txt**, and then shuts down all machines. See below for a sample input/output for models 100A and 100C and you should follow similar interface for models 100B and 100D. Do not include models that your system cannot support at this time.

```
Please enter a startup code --> some_secret_code<E>
Initialize machines. Please wait ...
Machines are ready.
Available machines: 100A1, 100A2, 100B1, 100B2, 100B3, 100C1, 100D1

Select a machine --> 100A1<E>
This machine accepts one-dollar bill only.
Available items:
     1A 50 candy bar
     1B 35 chocolate chips
     1C 75 cookies
     1D 60 brownie
     1E 65 donut
Insert your money --> 100 -1<E>
Select an item --> 1B<E>
You entered an amount of 100 cents.
The cost of this item is 35 cents.
Processing your purchase ...
Your change of 65 cents is given as:
    quarter(s): 2
    dime(s):    1
    nickel(s):  1
Thank you! Please take your item and
changes.

Select a machine --> 100A1<E>
This machine accepts one-dollar bill only.
Available items:
     1A 50 candy bar
     1B 35 chocolate chips
     1C 75 cookies
     1D 60 brownie
     1E 65 donut
Insert your money --> 100 -1<E>
Select an item --> 1B<E>
You entered an amount of 100 cents.
The cost of this item is 35 cents.
```

```
Processing your purchase ...
Insufficient changes!
Your transaction cannot be processed.
Please take back your dollar bill.

Select a machine --> 100A1<E>
This machine accepts one-dollar bill only.
Available items:
     1A 50 candy bar
     1B 35 chocolate chips
     1C 75 cookies
     1D 60 brownie
     1E 65 donut
Insert your money --> 100 -1<E>
Select an item --> 1C
You entered an amount of 100 cents.
The cost of this item is 75 cents.
Processing your purchase ...
Your change of 25 cents is given as:
    quarter(s): 0
    dime(s):    2
    nickel(s):  1
Thank you! Please take your item and
changes.

Select a machine --> 100C1<E>
This machine accepts credit card only.
Available items:
     1A 300 ham sandwich
     1B 275 egg sandwich
     1C 325 tuna sandwich
     1D 200 healthy salad
Enter your credit card number --> 4388576018402625<E>
Invalid credit card number was entered.
Enter your credit card number --> 4388576018410707<E>
Select an item --> 1B
The cost of this item is 275 cents.
Your credit card was successfully charged.
Thank you! Please take your item.

Select a machine --> some_secret_code<E>
Report is generating ...
System is shutting down.
```

Sample report for one machine:

```
Machine: 100A1
Initial Balance: $1.10 (0 $, 2 Q, 4 D, 4 N)

Trans Item     Cost    Paid ($, Q, D, N) Changes(Q, D, N)
  1     1B      35      100 (1  0  0  0)     65 (2  1  1)
  2     1C      75      100 (1  0  0  0)     25 (0  2  1)
Total Cost:   110

Current Balance: $2.20 (2 $, 0 Q, 1 D, 2 N)

Code     Id     Description        Initial   Current
  1A   1034    candy bar             5         4
  1B   1000    chocolate chips      10         9
```

```
  1C    1100    cookies              10        10
  1D    1123    brownie              20        20
  1E    1222    donut                 5         5
```

Sample report for one machine:

```
Machine: 100C1
Initial Balance: $0.00 (0 $, 0 Q, 0 D, 0 N)

Trans Item     Cost    Paid ($, Q, D, N)   Changes(Q, D, N)
  1     1B      275     275 (0  0  0  0)      0   (0  0  0)
Total Cost:     275

Current Balance: $2.75 (0 $, 0 Q, 0 D, 0 N)

Code      Id     Description        Initial   Current
  1A    6774    ham sandwich           5         5
  1B    6869    egg sandwich           5         4
  1C    6879    tuna sandwich          2         2
  1D    7555    healthy salad         10        10
```

There will be a data file, **machines.txt**, contains information about the number of machines, model, money, and available items. Another data file, **products.txt**, contains all product items to be loaded in the machines. We will also make these assumptions as well:

- Each product item consists of an id, description, and price.
- A purchase amount will be entered as a list of denominations and terminates by a sentinel value of -1 (e.g., `25 10 25 -1 or 100 100 -1`).
- An invalid denomination being entered will be ignored.
- The input data files contain valid data.
- Start up all machines at the same time.
- Shut down all machines at the same time.

Analysis and Design

Make sure you understand all the requirements. Ask for clarifications and missing information before you move on to the design phase. Select an additional model that you are going to implement. It must be an object-oriented system and make sure to take advantage of composition and inheritance in your design.

Draw a UML class diagram showing both has-a and is-a relationships as applicable. Include as much attributes and behaviors as you can for each class. Feel free to use pseudocode to document some complex behaviors. You may want to design the layout of the report as well.

Implementation

Write a vending machine application by putting those classes together. The program first loads data from products.txt and machines.txt (should work with other data files as long as they follow the same format – more products and/or more machines). It then allows customers to make purchases. Print helpful messages to the screen so that we know what is going on. You should be able to utilize polymorphism so that you will not need to do unnecessary work (*hint: an array/vector of generic machine objects*). Incorporate as many useful OOP feature as you can and points will be deducted if you do not utilizing at least some useful OOP features (e.g., const member functions, composition and aggregation, inheritance, operator overloading, etc.).

Team project: It is not required but it is highly recommended that you work in a team of two or three people since there will be too much work for one person. You must sign up by a specified deadline if it is going to be a team project (MW section -- Wednesday, 11/16/16; TTh section -- Thursday, 11/17/16). Make sure to split the responsibilities equally and work together (each person on the team should be responsible for one or more classes). You do need to provide a short description on who did what and your experience as a team project. If there are features that you like to add to your application, let me know ahead of time. Each team must do a short presentation about your project (including a 1-person team) or points will be deducted.

Extra credit: You can earn up to 6 additional points for two of the following (each feature is worth 3 points):

- Fully utilizing OOP approach – minimal ***main()*** with well-defined related classes (3 points)
- Implementation of each additional model (3 points)
- Refill machine -- add more coins and product items to a machine (3 points)
- Purchase multiple items in one transaction (3 points)

Bonus Points: The best overall project will earn an extra 6 points, the second best overall project will earn an extra 4 points, and the third best overall project will earn an extra 2 points. It will be based on the following:

- Your program is robust and easy to use (clear instructions and good data validation)
- Good presentation/demonstration (handout, PowerPoint, planned test cases/scenarios, etc.)
- Work well as a team
- Fully implementation of required features

Please provide documentation and applying good coding style because it is part of the grade. Use the provided template as the starting point. You must come up with a sufficient number of test cases since the test cases are also part of the grade. Please submit the following items **in a folder** if flash drive or CD is included for each team (can also submit source code ahead of time via Moodlerooms (MR) and a folder is not

needed). You can also submit just a hardcopy of the title page and remaining items electronically via MR (a PDF file of all printouts and actual source files in a .zip file). Each team only needs to submit one set of project submission.

1. Title page with name, class, project number, and relevant information about your program (compiler and system used, file names).
2. Status and team information (if applicable). Clearly document extra features and missing features if applicable.
3. Design documentation -- class diagram and pseudocode if applicable.
4. Printouts of any input/output. Include the test cases on project sheet at the minimum.
5. A printout of the source code.
6. A copy of source code on a flash drive or CD or MR -- source code (.h and .cpp files in a .zip file).

Your program will be graded as follow:

- Correctness/Efficiency: 40 points
- Test Cases: 5 points
- Documentation/Coding Style: 10 points
- Presentation: 5 points
