CS 485: Assignment 2 — 45 points: Execution: 18 pts, Coding Standards: 9 pts, Design 18 points

Assigned: Feb 18.

Bank Accounts

DUE: UML Diagram DRAFT, Wednesday, Feb 20, 3:30pm (Hard Copy)
DUE: UML Diagram, THURSDAY Feb 21, 3:00pm **(Hard Copy)**
DUE: Completed Solution, Mar 1, 3:30pm

Goals for this assignment

1. Practice your Object Oriented Design Skills
2. Practice your UML design skills
3. Practice new C++ syntax and idioms
4. **Review your CS 250 C++ Notes**
5. This is not an SDL application

We need a system that will support the backend operation of an online bank.

You need to represent both savings accounts and checking accounts. Both types of accounts have an account number and a balance. The account number for an account must never change. You do not need to represent a bank patron.

Each account type allows for deposits and withdraws to update the balance. A checking account has a minimum balance required to be kept at all times (this minimum balance is specific to a particular account and never changes for that account). If a transaction is made that moves the balance below the minimum balance or keeps the balance below the minimum balance, a fee is charged against the account. This minimum balance fee is specific to each checking account. The fee cannot incur a further fee. Account balances are allowed to be negative.

A savings account has a monthly fee. The fee is charged once a month (at the same time interest is generated), *only if* the savings account balance ever (since the previous round of interest generation) goes below (or stays below) a minimum balance as the result of a transaction or if the minimum monthly balance is not maintained at the time the interest is generated. The minimum monthly balance and monthly fee is specific to each savings account.

Both types of accounts accrue interest. An account has a single, set interest rate. No interest is applied to an account that has a negative balance. *Interest deposits never cause a fee.*

Store money as a long long and store money as pennies. All calculations with money must truncate (round down) to produce an integer.
long long amt = 100; // one dollar.

Your software will read the accounts from a file (Accounts.txt). Your software will also read the commands from a file (Commands.txt) and process the commands in order. These files are specified below.

You must produce a UML diagram of your design. The diagram must show inheritance, aggregation, and composition. You may choose to show other connections. **I strongly recommend** you bring your UML diagram to me for a review well before class on Feb 22. Use UMLet (http://umlet.com/) to produce your diagram. Include the XML file output by UMLet in your Visual Studio solution (just save the XML file into the **Visual Studio project directory**.)

Name your Visual Studio solution: CS485_Bank_*punetid*. Name your Visual Studio project: 01_Bank. You must use Git or Subversion and share your repository details with the instructor.

Follow the coding standards posted on the class website. Be sure to use VLD to check for memory leaks.

For this assignment, you will not receive any bad data in the files.

You should be able to handle up to 100 accounts. You should be able to process an unlimited number of commands.

Data File Format

Accounts.txt

```
S Acct# InitialBalance InterestRate MonthlyFee MinMonthlyBalance
C Acct# InitialBalance InterestRate MinBalance MinBalanceFee
```

Commands.txt

```
W Acct# Amt
D Acct# Amt
P
M
```

The possible Commands are:

- W - Withdraw
- D - Deposit
- P - Print all accounts (in the same order they appear in Accounts.txt)
- M - Charge monthly fees to savings accounts and Generate Interest in all accounts (in that order).

Example Files

Accounts.txt

```
S 1 1000 0.01 100 200
C 2 1000 0.01 100 50
```

Commands.txt

```
P
M
P
W 1 10
W 2 10
P
D 1 10
D 2 10
P
W 2 960
P
W 1 900
P
M
P
M
P
```

Expected Output

```
-------------
1, $10.00, 1.00%, 100, 200
2, $10.00, 1.00%, 100, 50
-------------
-------------
1, $10.10, 1.00%, 100, 200
2, $10.10, 1.00%, 100, 50
-------------
-------------
1, $10.00, 1.00%, 100, 200
2, $10.00, 1.00%, 100, 50
-------------
-------------
1, $10.10, 1.00%, 100, 200
2, $10.10, 1.00%, 100, 50
-------------
-------------
1, $10.10, 1.00%, 100, 200
2, $0.00, 1.00%, 100, 50
-------------
-------------
1, $1.10, 1.00%, 100, 200
2, $0.00, 1.00%, 100, 50
-------------
-------------
1, $0.10, 1.00%, 100, 200
2, $0.00, 1.00%, 100, 50
-------------
-------------
1, $-0.90, 1.00%, 100, 200
2, $0.00, 1.00%, 100, 50
-------------
```