CS 485: Assignment 3 45 points: Execution: 18 pts, Coding Standards: 9 pts, Design 18 points

Bank Accounts DUE: UML Diagram, **Thursday** Mar 7, 3:00pm **(Hard Copy)**
DUE: Completed Solution, Wednesday, Mar 13, 3:30 pm

Goals for this assignment

1. Fix your 01_Bank
2. You are **not** required to match the design presented in class.
3. Add a money class that encapsulates long long
4. Add a container class that encapsulates a static array of 100 account pointers
5. Add more complicated interest calculations via the **Strategy Pattern**
6. Practice new C++ syntax and idioms
7. **Review your CS 250 C++ Notes**

► You need to add a class that encapsulates money (money is still stored as a long long which contains the number of pennies). You must provide all the normal mathematical and relational operators *that your bank needs* as members of the Money class.

► You need to add a container class to hide the fact that the backing store is an array. Your design should be flexible enough for you to easily change the backing store to another data structure (vector, list, etc) in the future. Note, that the container class must not contain any bank logic.

► Now the bank offers more complicated interest structures. *Flat interest* is exactly the method for interest calculation in the previous assignment. A single interest rate is provided, no matter the balance in the account. *Tiered interest* assigns different interest rates depending on how much money is in the account when interest is generated. For example, an account might earn 2% interest if the account balance is at least $1000, 1% interest if the account balance is at least $100 (and less than $1000), and 0% interest otherwise.

Both types of accounts accrue interest and both types of accounts can use either type of interest calculation. No interest is applied to an account that has a negative balance. *Interest deposits never cause a fee.*

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. This must be the design for the entire Bank system. **Highlight the new classes or edited classes by placing `lw=5` at the end of the Properties description for the class.** 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 it is due. 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: 02_Bank.
Copy all of your code from 01_Bank to 02_Bank as a starting point. You **MUST** do this by the following method
1) Create 02_Bank
2) Use the Windows File Explorer to open the 02_Bank directory
3) Use the Windows File Explorer to open the 01_Bank directory
4) Copy all .h and .cpp files from 01_Bank to 02_Bank
5) Inside of Visual Studio, add the files to the 02_Bank project using Add Existing item.

You must use Git or Subversion.

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 INTEREST MonthlyFee MinMonthlyBalance
C Acct# InitialBalance INTEREST MinBalance MinBalanceFee

INTEREST can either be:

F 0.01
For a flat interest rate of 0.01

T 2 10000 20000 0.01 0.02
For a tiered interest calculation of:
Balance >= 20000  Interest is 0.02
Balance >= 10000  Interest is 0.01

The 2 indicates that there are two tiers of interest.

T 3 0 10000 20000 0.001 0.01 0.02
For a tiered interest calculation of:
Balance >= 20000  Interest is 0.02
Balance >= 10000  Interest is 0.01
Balance >= 0 Interest is 0.001
```

The Commands remain the same.

Example Files -- NOTE THE CHANGE IN OUTPUT FORMAT!

Accounts.txt

```
S 1 10000 F 0.01 100 100
C 2 20000 T 2 10000 20000 0.01 0.02 0 100
```

Commands.txt

```
W 1 100
D 2 100
P
M
P
W 2 10000
W 1 9900
P
M
P
D 1 102
P
M
P
W 2 800
P
M
P
```

Expected Output

```
-------------
1, $100.00, F 1.00%, $1.00, $1.00
2, $200.00, T $100.00 $200.00 1.00% 2.00%, $0.00, $1.00
-------------
-------------
1, $99.00, F 1.00%, $1.00, $1.00
2, $201.00, T $100.00 $200.00 1.00% 2.00%, $0.00, $1.00
-------------
-------------
1, $99.99, F 1.00%, $1.00, $1.00
2, $205.02, T $100.00 $200.00 1.00% 2.00%, $0.00, $1.00
-------------
-------------
1, $0.99, F 1.00%, $1.00, $1.00
2, $105.02, T $100.00 $200.00 1.00% 2.00%, $0.00, $1.00
-------------
-------------
1, $-0.01, F 1.00%, $1.00, $1.00
2, $106.07, T $100.00 $200.00 1.00% 2.00%, $0.00, $1.00
-------------
-------------
1, $1.01, F 1.00%, $1.00, $1.00
2, $106.07, T $100.00 $200.00 1.00% 2.00%, $0.00, $1.00
-------------
-------------
1, $1.02, F 1.00%, $1.00, $1.00
2, $107.13, T $100.00 $200.00 1.00% 2.00%, $0.00, $1.00
-------------
-------------
1, $1.02, F 1.00%, $1.00, $1.00
2, $99.13, T $100.00 $200.00 1.00% 2.00%, $0.00, $1.00
-------------
-------------
1, $1.03, F 1.00%, $1.00, $1.00
2, $99.13, T $100.00 $200.00 1.00% 2.00%, $0.00, $1.00
-------------
```