CS 485: Assignment 4 45 points: Execution: 18 pts, Coding Standards: 9 pts, Design 18 points

Bank Accounts DUE: UML Diagram, **Monday** Mar 18, 3:30pm **(Hard Copy)**
DUE: Completed Solution, Friday, Mar 22, 3:30 pm

Goals for this assignment

1. Fix your 02_Bank
2. Use enum class
3. Use exceptions
4. Add a currency class that can represent the currency of an amount of money.
5. Add exceptions to your Account Container and Money/Currency class
6. Practice new C++ syntax and idioms
7. **Review your CS 250 C++ Notes**

► You need to add a class that encapsulates the notion of currency. You must support the following currencies: US Dollars (USD, $), British Pounds (GBP, £), Euro (EUR, €), Japanese Yen (YEN, ¥). You must use an Enum class to store the type of currency.

► You must define an exception class: CurrencyMismatchException

► Each Money object must contain a Currency. When Money is read from a file a currency will be specified. Mathematical operations with Money must have operands with matching currencies. If unmatched currencies are used in a mathematical operation, throw an exception.

► All fees, minimum balances, and interest tiers for a particular account will use the same currency.

► For this project, any command that generates a currency mismatch exception should be ignored but must not crash the program.

► Your Account Container class must generate exceptions if an invalid index is accessed (-1 or size+1, std::range_error) or if you try to add an account to a full container (std::bad_array_new_length). These exceptions should lead to program termination *but display as useful an error message as you can.*

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: 03_Bank. Copy all of your code from 02_Bank to 03_Bank as a starting point. Follow the same procedure as last time to make this copy.
You must use Git and put your VS Solution into the private Bank_*githubname* repository created for you on the CS485s17 GitHub organization.

Follow the coding standards posted on the class website. Be sure to use mem_debug or 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 USD 10000 USD 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 USD 0 USD 10000 USD 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
```

All Money is now formatted as:

```
CUR 10000
Three letter currency abbreviation followed by a long long.
```

The Commands remain the same.

Example Files

Accounts.txt

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

Commands.txt

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

Expected Output

```
-------------
1, USD99.00, F 1.00%, USD1.00, USD1.00
2, GBP201.00, T GBP100.00 GBP200.00 1.00% 2.00%, GBP0.00, GBP1.00
-------------
-------------
1, USD99.99, F 1.00%, USD1.00, USD1.00
2, GBP205.02, T GBP100.00 GBP200.00 1.00% 2.00%, GBP0.00, GBP1.00
-------------
-------------
1, USD0.99, F 1.00%, USD1.00, USD1.00
2, GBP105.02, T GBP100.00 GBP200.00 1.00% 2.00%, GBP0.00, GBP1.00
-------------
-------------
1, USD-0.01, F 1.00%, USD1.00, USD1.00
2, GBP106.07, T GBP100.00 GBP200.00 1.00% 2.00%, GBP0.00, GBP1.00
-------------
-------------
1, USD1.01, F 1.00%, USD1.00, USD1.00
2, GBP106.07, T GBP100.00 GBP200.00 1.00% 2.00%, GBP0.00, GBP1.00
-------------
-------------
1, USD1.02, F 1.00%, USD1.00, USD1.00
2, GBP107.13, T GBP100.00 GBP200.00 1.00% 2.00%, GBP0.00, GBP1.00
-------------
-------------
1, USD1.02, F 1.00%, USD1.00, USD1.00
2, GBP99.13, T GBP100.00 GBP200.00 1.00% 2.00%, GBP0.00, GBP1.00
-------------
-------------
1, USD1.03, F 1.00%, USD1.00, USD1.00
2, GBP99.13, T GBP100.00 GBP200.00 1.00% 2.00%, GBP0.00, GBP1.00
-------------
```