CISC 3110 **Ari Mermelstein**

Homework #5: Due December 13

Overview

In this assignment, you will be asked to implement a Bank simulation program. In this simulation, you will have 3 main classes: `Currency`, `Account`, and `Customer`.

I will provide prototypes and descriptions for all of the member functions that you will need to implement.

The `Currency` class will have one member variable: the number of cents. A Currency object will be used to keep track of the money in a bank account.

member functions:

```
Currency();
```

This will initialize the number of cents to 0.

```
Currency(int cents);
```

This will initialize the number of cents to the specified value.

Note: You can implement both constructors as one using default arguments if you wish.

```
int getValue();
```

This will return the number of cents in the account.

operators:

```
Currency& operator+=(Currency& rhs);
Currency& operator-=(Currency& rhs);
```

These will increment/decrement the number of cents as specified by the Currency parameter.

```
friend ostream& operator<<(ostream& os, Currency& rhs);
```

This will print the amount of money in the account **in dollars**.

The `Account` class will have the following member variables: A Currency object that will store the balance of the account, and a name that describes the type of account (savings, checking, CD, etc.)

member functions:

```
Account();
```

This constructor doesn't have to do anything. By protocol, this constructor should not be called directly.

```
Account(const string& type);
```

This creates an account of type "type" with an initial balance of 0.

```
Account(const string& type, Currency& initAmount);
```

This creates an account of type "type" with an initial deposit of "initAmount"

```
void withdraw(Currency& money);
```

Withdraws "money" cents from the account. This function throws an exception if the balance would fall below 0.

```
void deposit(Currency& money);
```

Deposits "money" cents into the account.

```
Currency getBalance();
```

```
string getType();
```

Accessor functions for the Account class.

operators:

```
friend ostream& operator<<(ostream& os, Account& account);
```

This will print the type of the account with the balance.

The `Customer` class will have 5 member variables: the first name of the customer, the last name of the customer, the maximum number of accounts the customer can have, how many accounts the customer has actually opened, and a dynamically allocated array of Accounts.

member functions:

`Customer();`
This will not have to do anything. This will make it easier to create an array of Customers in the main program. By protocol, the client (i.e. you when you write the main() program) will not call this directly.

```
Customer(const string& first, const string& last, int numAccounts);
```

This constructor will dynamically allocate an array of Accounts of size "numAccounts." It will also set the other member variables accordingly.

```
void addAccount(string type, Currency& initialAmount);
```

This function will allow the Customer to open an account.

```
string getFirstName();

string getLastName();
```

Accessor functions for the Customer class.

```
void deposit(Currency& money, const string& type);

void withdraw(Currency& money, const string& type);

void balance(Currency& money, const string& type);
```

These functions will allow the Customer to withdraw or deposit money from an account. You can assume by protocol that each customer can only have at most one account of each type. Each function should throw an exception if the account doesn't exist. The withdraw function should throw an exception if the balance would be below 0.

```
Customer(Customer& another);

Customer& operator=(Customer& another);
```

```
~Customer();
```
Copy constructor, assignment operator, and destructor for the customer class. Why are these necessary?

You will also have to create a main program that does the following:

1. Create an array of Customers called *bank*.
2. Create a menu that gives the user the following options:
    - O. Become a new customer of the bank. If this chosen, ask the user for their name, and register them as a customer in the bank. If they are already a customer, print a message that they already have an account.
    - B. Allows the user to search for the account balance of a particular account. You should prompt the user for their name and account type.
    - W. Allows the user to withdraw money from a particular account. You should prompt the user for their name and account type. Complain to the user if their account would be overdraw.
    - D. Allows the user to deposit money from a particular account. You should prompt the user for their name and account type.
    - Q. Exit the program

Each of the above choices should be implemented in a function!

What to submit:

Currency.h Currency.cpp
Account.h Account.cpp
Customer.h Customer.cpp
bankProgram.cpp

Note: It is very hard to coordinate all of this on code blocks. It is much easier (I promise!) if you use linux, and manually compile each thing separately as you go. Be conscious of which files need to include which, and be careful which .o's you need to compile together.