COMPUTER SCIENCES AND SOFTWARE ENGINEERING
AUBURN UNVERSITY

COMP 2710
Software Construction

Fall 2014

Lab 4
Bank Transaction System
with Inheritance and Virtual Functions

Due: December 3, 2014 11:55pm

Points Possible: 100
Due: Written Portion: November 19th, 2014 by 11:55 pm. Submission via Canvas (25 points)
Program: December 3rd, 2014 by 11:55 pm via Canvas (75 points)

No late assignments will be accepted.
No collaboration between students. Students should NOT share any project code with each other. Collaborations in any form will be treated as a serious violation of the University's academic integrity code.

Goals:

- To develop a simple bank transaction application that take advantage of inheritance
- To use virtual functions to simplify implementation
- To understand the use of vectors that contain different type of objects
- To perform Object-Oriented Analysis, Design, and Testing
- To learn the use of vector class template

Process – 25 points:
Create a text, doc, or .pdf file named "<username>-4p" (for example, mine might read "lim-4p.txt") and provide each of the following. Please submit a text file, a .doc file or .pdf file (if you want to use diagrams or pictures, use .doc or .pdf). You are free to use tools like Visio to help you, but the final output needs to be .txt, .doc, or .pdf.

1. **Analysis:** Prepare use cases. Remember, these use cases describe how the user interacts with a bank transaction system (what they do, what the systems does in response, etc.). Your use cases should have enough basic details such that someone unfamiliar with the system can have an understanding of what is happening in the cash register system interface. They should not include internal technical details that the user is not (and should not) be aware of. *You must include a use case diagram that shows relationships between use cases. Your use case descriptions must include the following: Use case name, description, assumptions, actors, pre-conditions, post-conditions, steps (basic flow), variations (alternate flow), non-functional requirements, and issues.*
2. **Design**:
   a. Create a Class Diagram. Be sure to include:
      1) The name and purpose of the classes
      2) The member variables and the functions of the class

3) Show the interactions between classes (for example, ownership or dependency)
4) Any relevant notes that don't fit into the previous categories can be added

b. Create the data flow diagrams. Show all the entities and processes that comprise the overall system and show how information flows from and to each process.

3. **Testing**: Develop lists of ***specific*** test cases OR a driver will substitute for this phase:

1) For the system at large. In other words, describe inputs for "nominal" usage. You may need several scenarios. In addition, suggest scenarios for abnormal usage and show what your program should do (for example, entering a negative number for a menu might ask the user to try again).

2) For each object. (Later, these tests can be automated easily using a simple driver function in the object)

4. Implement the simple cash register application

5. **Test Results**: ***After developing the Bank Transaction System***, actually try all of your test cases (both system and unit testing). Show the results of your testing (a copy and paste from your program output is fine – don't stress too much about formatting as long as the results are clear). You should have test results for every test case you described. If your system doesn't behave as intended, you should note this. Note: Driver output will substitute for this phase.

Program Portion:

In this Lab assignment, you will implement a simple bank transaction application that allows the bank and the customers to perform the following functions:

1. Create different types of bank accounts, e.g. saving, checking, money market, and CD accounts
2. Perform different types of transactions, e.g. withdrawal, deposit, transfer, and inquiry
3. Print all the transactions recorded in the bank transaction system
4. Print all the deposits and withdrawals (with account name and types), and the final total Bank balance of all deposits minus the withdrawals and fees.

Banks have many different types of accounts, often with different rules for fees associated with transactions, such as withdrawals. Customers are allowed to transfer funds between accounts incurring the appropriate fees associated with withdrawal of funds from one account.

Write a program with a base class for a bank account and four derived classes (as described below) representing accounts with different rules for withdrawing funds. Write the following functions for each of these accounts:

1. Create a bank account of any type
2. Deposit funds into an account
3. Withdraw funds from an account, based on the rules for that account. The withdraw function should return an integer indicating the status (either ok or insufficient funds for the withdrawal to take place).
4. Transfer funds from one account (of any type) to another. A transfer is a withdrawal from one account and a deposit into the other. Since transfer can be done at any time with any type of account, the withdrawal function in the classes must be virtual.
5. Inquiry on the balance of an account of any type
6. Print all the transactions recorded in the bank transaction system
7. Print all the deposits and withdrawals (with account name and types), and the final total Bank balance of all deposits minus the withdrawal and fees.

Write a main program that can perform all the above functions, e.g. creates accounts of any type, transfer funds from one account to another, etc.

You will use inheritance and virtual functions to simplify the definition of the classes that contain the above variations in functions for each of the different account types.

For the classes, create a base class called `BankAccount` that has the name of the owner of the account (a string), account type (a string) and the balance in the account (`double`) as the data member. Include member functions `deposit` and `withdraw` (each with a `double` for the amount as an argument) and accessor functions `getName` and `getBalance`. `Deposit` will add the amount to the balance (assuming the amount is non-negative) and `withdraw` will subtract the amount from the balance (assuming the amount is non-negative and is less than or equal to the balance) plus any fee or penalty. The `withdraw` function *must be implemented as a* ***virtual function*** so that the correct withdraw function of the derived class is called, even when the virtual function is called from a base class pointer. Include also the **virtual function** `printTransactions` to print all transactions in this account. The `BankAccount` class also contains a vector member variable to store all the transactions in this account. Since `printTransactions` *must be a* ***virtual function***, the overriding virtual function `printTransactions` of the derived class will be called to perform the correct print for each of the derived class. Include also the virtual function `computeBankBalance` to return either the amount of deposit, withdrawal, fees and penalties of each transaction in this account depending on what type of account this is. Again, `computeBankBalance` *must be implemented as a* ***virtual function***.

Create the following classes that are derived from the `BankAccount` class.

1. Create a class called `SavingAccount` that is derived from `BankAccount`. In a `SavingAccount` the user can withdraw any amount of fund without any restriction.
2. Create a class called `CheckingAccount` that is derived from `BankAccount`. In a `CheckingAccount` the user can withdraw any amount of fund without a fee only if the final balance after the withdrawal is $500 or more. If the balance after the withdrawal is less than $500, the withdrawal is charged a fee of $2.50 that will be subtracted from the balance. *The withdrawal function must override the one in the base class using virtual functions*.
3. Create a class called `MoneyMarketAccount` that is derived from `BankAccount`. In a `MoneyMarketAccount` the user gets two free withdrawals in a given period of time (don't worry about the time in this lab). After the free withdrawals have been used, a withdrawal fee of $1.50 is deducted from the balance per withdrawal. Hence the class must have data member to keep track of the number of withdrawals. *It must override the withdraw definition using virtual functions.*
4. Create a class called `CDAccount` class that is derived from `BankAccount`. In addition to having the name and balance, `CDAccount` also has an interest rate. CDs incur penalties for early withdrawal of funds. Assume that a withdrawal of funds (any amount) incurs a penalty of 25% of the annual interest earned on the account. Assume the amount withdrawn plus the penalty are deducted from account balance. Again, *the withdrawal function must override the one in the base class using virtual functions*.

For the purpose of this lab, do not worry about other functions and properties of these accounts (such as when and how interest is paid).

For the function to print all the transactions recorded in the bank transaction system, you must store all the transactions in a vector. For each of the different types of account information, the **virtual function** `print_transaction` function will first print the bank transaction banner and number and then prints the details of the transaction based on the derived class: `SavingAccount`, `CheckingAccount`, `CDAccount`, or `MoneyMarketAccount`.

For the function to print all the deposits and withdrawals (with account name and types), your program must keep ***a single list*** of different types of accounts in a vector of pointers to `BankAccount` objects, where all the accounts in the vector have current withdrawal or deposit activities. The program will repetitively do the following for each of the account: scan though all the transactions in each of the account to list amount of deposit, withdrawals and fees for each transaction in that account. When all the accounts have been processed, it will then print the final total Bank balance of all deposits minus the withdrawals and fees.

The user interface

Write a menu-based and text-based user interface for the simple Bank Transaction application where the user can process one of the seven options. Each transaction will be stored in a vector associated with the account. Each account that has active deposit or withdrawal transaction will be stored in a vector called `activeAccounts`. At any point the user can print out all the transactions stored in the Bank Transaction System.

The following is a sample of a program that uses the API provided by the `BankAccount` class. The sample program only shows some of the major features of the `BankAccount` class and may not show other minor features or details of the implementation.

Sample Usage of the API:

```
int main()
{
/*
Create a vector or "basket of active account"
Note: The data type is a pointer to the BASE CLASS BankAccount
*/
vector<BankAcount *> activeAccounts;

//create some objects of the various derived classes

MoneyMarketAccount* mma = new MoneyMarketAccount;
/*
Omitted here are codes that follows that will enter the
information related to the account. The next step is to process a
deposit transaction.
*/
mma->deposit();

/*
The next virtual function will process the appropriate
withdrawal.
*/
```

```
mma->withdraw();

/*
Then, add the derived class objects to the vector.
This is legal, since activeAccounts stores pointers to
BankAccount objects.
*/
activeAccounts.push_back(mma);

CheckingAccount* ca = new CheckingAccount;

/*
Omitted here are codes that follows that will enter the
information related to the account. The next step is to process a
deposit transaction.
*/
ca->deposit();

/*
The next virtual function will process the appropriate
withdrawal.
*/
ca->withdraw();

/*
Then, add the pointer to the derived class objects to the vector.
*/
activeAccounts.push_back(ca);

/*
Call the function printTransactions() in the base class
BankAccount which will call the virtual function
printTransactions() as defined in the MoneyMarketAccount class to
output the complete transaction information on all the Money
Market transactions.
*/
activeAcounts[0]->printTransactions();

/*
Call the function printTransactions() in the base class
BankAccount which will call the virtual function
printTransactions() as defined in the CheckingAccount class to
output the complete transaction information on all the Checking
Account transactions.
*/
activeAcounts [1]->printTransactions();


}
```

The above sample code does not show the menu and how the appropriate account and transaction information are entered by the user into the account objects. In your program, you must integrate all these functions.

The user interface must check for correct input value from the users. If there is any error, e.g. selecting an invalid withdrawal amount, then the program must display the appropriate error message and continue to prompt for the correct input. Your program must not exit or terminate when there is an incorrect input value.

The name of your program must be called <username>_4.cpp (for example, mine would read "lim_4.cpp"

Use comments to provide a heading at the top of your code containing your name, Auburn Userid, and filename. You will lose points if you do not use the specific program file name, or do not have a comment block on **EVERY** program you hand in.

Your program's output need not exactly match the style of the sample output (see the end of this file for one example of sample output).

Important Notes:
You must use an object-oriented programming strategy in order to design and implement this cash register system (in other words, you will need write class definitions and use those classes, you can't just throw everything in main() or an independent function). A well-done implementation will produce a number of robust classes, many of which may be useful for future programs in this course and beyond. Some of the classes in your previous lab project may also be re-used here, possibly with some modifications. Remember good design practices discussed in class:

a) A class should do one thing, and do it well
b) Classes should NOT be highly coupled
c) Classes should be highly cohesive

- You should follow standard commenting guidelines.

- You DO NOT need any graphical user interface for this simple, text-based application. If you want to implement a visualization of some sort, then that is extra credit.

Error-Checking:

You should provide enough error-checking that a moderately informed user will not crash your program. This should be discovered through your unit-testing. Your prompts should still inform the user of what is expected of them, even if you have error-checking in place.

Submit your program through the Canvas online system. If for some disastrous reason Canvas goes down, instead e-mail your submission to TA – Steffi Mariya Gnanaprakasa P Arasu – at smg0033@tigermail.auburn.edu. Canvas going down is not an excuse for turning in your work late.

You should submit the two files in digital format. No hardcopy is required for the final submission:

<username>_4.cpp
<username>_4p.txt (script of sample normal execution and a script of the results of testing)

A sample execution is shown below, where the bold fonts indicate input by the user.

```
>  lim_4

        ============================================================
        |                  Welcome to Aubie Bank!                  |
        ============================================================

Select an option: (1) Create account, (2) Deposit fund, (3) Withdraw fund, (4)
Transfer fund, (5) Inquiry, (6) Print all transactions, (7) Print Bank balance,
(8) Quit:  1

Enter name:  Kevin
Enter account type:  CD Account
Enter interest rate (in percentage):  5

        ============================================================
        |                  Welcome to Aubie Bank!                  |
        ============================================================

Select an option: (1) Create account, (2) Deposit fund, (3) Withdraw fund, (4)
Transfer fund, (5) Inquiry, (6) Print all transactions, (7) Print Bank balance,
(8) Quit:  2

Enter name:  Kevin
Enter account type:  CD account
Enter deposit amount:  $ 4500.00
Balance:  $ 4500.00

        ============================================================
        |                  Welcome to Aubie Bank!                  |
        ============================================================

Select an option: (1) Create account, (2) Deposit fund, (3) Withdraw fund, (4)
Transfer fund, (5) Inquiry, (6) Print all transactions, (7) Print Bank balance,
(8) Quit:  1

Enter name:  Kathy
Enter account type:  Money market account

        ============================================================
        |                  Welcome to Aubie Bank!                  |
        ============================================================

Select an option: (1) Create account, (2) Deposit fund, (3) Withdraw fund, (4)
Transfer fund, (5) Inquiry, (6) Print all transactions, (7) Print Bank balance,
(8) Quit:  2

Enter name:  Kathy
Enter account type:  Money market account
Enter deposit amount:  $ 3000.00
Balance:  $ 3000.00

        ============================================================
        |                  Welcome to Aubie Bank!                  |
        ============================================================
```

```
Select an option: (1) Create account, (2) Deposit fund, (3) Withdraw fund, (4)
Transfer fund, (5) Inquiry, (6) Print all transactions, (7) Print Bank balance,
(8) Quit:  1

Enter name:  Joan
Enter account type:  Checking account


        ============================================================
        |                  Welcome to Aubie Bank!                  |
        ============================================================

Select an option: (1) Create account, (2) Deposit fund, (3) Withdraw fund, (4)
Transfer fund, (5) Inquiry, (6) Print all transactions, (7) Print Bank balance,
(8) Quit:  2

Enter name:  Joan
Enter account type:  Checking account
Enter deposit amount:  $ 1500.00
Balance:  $ 1500.00

        ============================================================
        |                  Welcome to Aubie Bank!                  |
        ============================================================

Select an option: (1) Create account, (2) Deposit fund, (3) Withdraw fund, (4)
Transfer fund, (5) Inquiry, (6) Print all transactions, (7) Print Bank balance,
(8) Quit:  1

Enter name:  Trent
Enter account type:  Saving account

        ============================================================
        |                  Welcome to Aubie Bank!                  |
        ============================================================

Select an option: (1) Create account, (2) Deposit fund, (3) Withdraw fund, (4)
Transfer fund, (5) Inquiry, (6) Print all transactions, (7) Print Bank balance,
(8) Quit:  2

Enter name:  Trent
Enter account type:  Saving account
Enter deposit amount:  $ 2000.00
Balance:  $ 2000.00

        ============================================================
        |                  Welcome to Aubie Bank!                  |
        ============================================================

Select an option: (1) Create account, (2) Deposit fund, (3) Withdraw fund, (4)
Transfer fund, (5) Inquiry, (6) Print all transactions, (7) Print Bank balance,
(8) Quit:  3

Enter name:  Kathy
Enter account type:  Money market account
Enter withdrawal amount:  $ 500.00
Balance:  $ 2500.00

        ============================================================
        |                  Welcome to Aubie Bank!                  |
        ============================================================
```

```
Select an option: (1) Create account, (2) Deposit fund, (3) Withdraw fund, (4)
Transfer fund, (5) Inquiry, (6) Print all transactions, (7) Print Bank balance,
(8) Quit:  3

Enter name:  Kevin
Enter account type:  CD account
Enter withdrawal amount:  $ 1500.00
Withdrawal Penalty:  $ 56.25
Balance:  $ 2943.7

        ============================================================
        |                  Welcome to Aubie Bank!                  |
        ============================================================

Select an option: (1) Create account, (2) Deposit fund, (3) Withdraw fund, (4)
Transfer fund, (5) Inquiry, (6) Print all transactions, (7) Print Bank balance,
(8) Quit:  3

Enter name:  Kathy
Enter account type:  Money market account
Enter withdrawal amount:  $ 800.00
Balance:  $ 1700.00

        ============================================================
        |                  Welcome to Aubie Bank!                  |
        ============================================================

Select an option: (1) Create account, (2) Deposit fund, (3) Withdraw fund, (4)
Transfer fund, (5) Inquiry, (6) Print all transactions, (7) Print Bank balance,
(8) Quit:  3

Enter name:  Joan
Enter account type:  Checking account
Enter withdrawal amount:  $ 1100.00
Withdrawal Fee:  $ 2.50
Balance:  $ 397.50

        ============================================================
        |                  Welcome to Aubie Bank!                  |
        ============================================================

Select an option: (1) Create account, (2) Deposit fund, (3) Withdraw fund, (4)
Transfer fund, (5) Inquiry, (6) Print all transactions, (7) Print Bank balance,
(8) Quit:  3

Enter name:  Kathy
Enter account type:  Money market account
Enter withdrawal amount:  $ 200.00
Withdrawal fee:  $ 1.50
Balance:  $ 1498.50

        ============================================================
        |                  Welcome to Aubie Bank!                  |
        ============================================================

Select an option: (1) Create account, (2) Deposit fund, (3) Withdraw fund, (4)
Transfer fund, (5) Inquiry, (6) Print all transactions, (7) Print Bank balance,
(8) Quit:  6

Transaction # 1:
Account created
Name:  Kevin
Account type:  CD Account
```

```
Interest rate (in percentage):  5

Transaction # 2:
Deposit
Name:  Kevin
Account type:  CD account
Deposit amount:  $ 4500.00
Balance:  $ 4500.00

Transaction # 3:
Account created
Name:  Kathy
Account type:  Money market account

Transaction # 4:
Deposit
Name:  Kathy
Account type:  Money market account
Deposit amount:  $ 3000.00
Balance:  $ 3000.00

Transaction # 5:
Account created
Name:  Joan
Account type:  Checking account

Transaction # 6:
Deposit
Name:  Joan
Account type:  Checking account
Deposit amount:  $ 1500.00
Balance:  $ 1500.00

Transaction # 7:
Account created
Name:  Trent
Account type:  Saving account

Transaction # 8:
Deposit
Enter name:  Trent
Account type:  Saving account
Deposit amount:  $ 2000.00
Balance:  $ 2000.00

Transaction # 9:
Withdrawal
Name:  Kathy
Account type:  Money market account
Withdrawal amount:  $ 500.00
Balance:  $ 2500.00

Transaction # 10:
Withdrawal
Name:  Kevin
Account type:  CD account
Withdrawal amount:  $ 1500.00
Withdrawal Penalty:  $ 56.25
Balance:  $ 2943.7

Transaction # 11:
Withdrawal
Name:  Kathy
```

```
Account type:  Money market account
Withdrawal amount:  $ 800.00
Balance:  $ 1700.00

Transaction # 12:
Withdrawal
Name:  Joan
Account type:  Checking account
Withdrawal amount:  $ 1100.00
Withdrawal Fee:  $ 2.50
Balance:  $ 397.50

Transaction # 13:
Withdrawal
Name:  Kathy
Account type:  Money market account
Withdrawal amount:  $ 200.00
Withdrawal fee:  $ 1.50
Balance:  $ 1498.50



        ============================================================
        |                  Welcome to Aubie Bank!                  |
        ============================================================

Select an option: (1) Create account, (2) Deposit fund, (3) Withdraw fund, (4)
Transfer fund, (5) Inquiry, (6) Print all transactions, (7) Print Bank balance,
(8) Quit:  7



Transaction # 2:
Deposit
Name:  Kevin
Account type:  CD account
Deposit amount:  $ 4500.00

Transaction # 4:
Deposit
Name:  Kathy
Account type:  Money market account
Deposit amount:  $ 3000.00
Balance:  $ 3000.00

Transaction # 6:
Deposit
Name:  Joan
Account type:  Checking account
Deposit amount:  $ 1500.00
Balance:  $ 1500.00

Transaction # 8:
Deposit
Enter name:  Trent
Account type:  Saving account
Deposit amount:  $ 2000.00
Balance:  $ 2000.00

Transaction # 9:
Withdrawal
Name:  Kathy
```

```
Account type:  Money market account
Withdrawal amount:  $ 500.00
Balance:  $ 2500.00

Transaction # 10:
Withdrawal
Name:  Kevin
Account type:  CD account
Withdrawal amount:  $ 1500.00
Withdrawal Penalty:  $ 56.25
Balance:  $ 2943.7

Transaction # 11:
Withdrawal
Name:  Kathy
Account type:  Money market account
Withdrawal amount:  $ 800.00
Balance:  $ 1700.00

Transaction # 12:
Withdrawal
Name:  Joan
Account type:  Checking account
Withdrawal amount:  $ 1100.00
Withdrawal Fee:  $ 2.50
Balance:  $ 397.50

Transaction # 13:
Withdrawal
Name:  Kathy
Account type:  Money market account
Withdrawal amount:  $ 200.00
Withdrawal fee:  $ 1.50
Balance:  $ 1498.50


Total Bank Balance = $6,960.25



        ============================================================
        |                  Welcome to Aubie Bank!                  |
        ============================================================

Select an option: (1) Create account, (2) Deposit fund, (3) Withdraw fund, (4)
Transfer fund, (5) Inquiry, (6) Print all transactions, (7) Print Bank balance,
(8) Quit:  8


        ============================================================
        |            Thank you for banking with Aubie Bank!        |
        ============================================================
```