Project 04 - Extended Banking

Sunday, July 8 at 11:59pm

Introduction

Extend the inheritance hierarchy from the previous project by changing the classes to template classes. Do not worry about rounding in classes that are instantiated as integer classes, you may just use the default rounding. You will add an additional data member, method, and bank account type that inherits SavingsAccount ("CD", or certificate of deposit).

Deliverables

- A driver program (driver.cpp)
- An implementation of Account class (account.h)
- An implementation of SavingsAccount (savingsaccount.h)
- An implementation of CheckingAccount (checkingaccount.h)
- An implementation of CDAccount (cdaccount.h)

1 Account class

Change to a template class and add/modify the following items.

1.1 Data Members

accountNumber (string)

1.2 Member Functions

Constructors

Add accountNumber to the constructor.

Accessors

string getAccountNumber() Returns the account number.

void displayAccount() Prints the account type, fee or interest rate, and balance. Determine if this should be virtual, pure virtual, or regular.

2 SavingsAccount class

Change to a template class. Add accountNumber to the constructor. You may make interestRate ‘protected’ so you can directly access it in CDAccount. No other modifications.

3 CheckingAccount class

Change to a template class. Add accountNumber to the constructor. No other modifications.

4 CDAccount class

Implement a template class that is derived from SavingsAccount class.

4.1 Data Members

No additional data members (inherits its data members).

4.2 Member Functions

Constructors

Defines a parameterized constructor which calls the SavingsAccount constructor.

Destructor

Defines destructor. If you have allocated memory, clean it up.

Mutators

void debit(amount) Redefines the member function to debit nothing from the balance and return false, since you are not allowed to debit from a CD during its term.

void credit(amount) Invokes the base class credit member function to add to the account.

5 The Driver Program

In your driver, prompt the user for input as shown below. Create two vectors (savingsAccounts and checkingAccounts). The first has Account pointers with type double that point to a CD and a Savings account, the second has Account pointers with integer type and point to two Checking accounts. Initialize the objects with the user’s input and appropriate constructors.

Next, create a loop that, for each savingsAccount, allows the user to deposit and withdraw from the account using the base class member functions. After withdrawing and depositing, downcast to a SavingsAccount pointer and add interest. Print the updated information by invoking the base class member function displayAccount().

Create another loop for each checkingAccount that allows the user to deposit and withdraw from the account using the base class member functions. After withdrawing and depositing, print the updated information by invoking the base class member function displayAccount(). Verify that, even if doubles are passed to the constructor, their values are coerced to integers.

5.1 Example input:

```
CD Account 1
Please enter the account number: 98765432
Please enter the balance: 70
Please enter the interest rate: 0.1
(repeat for the savings account)

Checking Account 1
Please enter the account number: 01010101
Please enter the balance: 70.55
Please enter the fee: 1.99
(repeat for next checking account)
```

5.2 Example output:

```
CD Account 1 balance: $70.00
Enter an amount to withdraw from Account 1: 10.00
Cannot debit from a CD account.
Enter an amount to deposit into Account 1: 35.00
Adding $10.50 interest to Account 1
CD account 01234567 has an interest rate of 0.10 and a balance of $115.50

Checking Account 1 balance: $70
Enter an amount to withdraw from Account 1: 9.99 0
$1 transaction fee charged
Enter an amount to deposit into Account 1: .99
$1 transaction fee charged.
Checking account 01010101 has a $1 transaction fee and a balance of $59
```

6 Submission

The TA will provide a makefile to compile your program. Do not modify the makefile, as the TA will use it to grade your work. The TA may use a different driver program with different test cases than yours, so make sure to thoroughly test the program, and that you are invoking member functions properly. If there are questions of clarification, please post to Piazza.

Submit the following files to Canvas (named **EXACTLY** as shown below) **without zipping** for grading purposes:

1. driver.cpp
2. account.h
3. savingsaccount.h
4. checkingaccount.h
5. cdaccount.h

The rubric is as follows:

1. A program that does not compile will result in a zero
2. Copying or sharing code results in a zero on the project and a FF in the course
3. Runtime error and compilation warning (or improperly named/submitted) 10%
4. Commenting and style 10%

5. Functionality 80% (classes were implemented and used as specified)