Dan Imbimbo

COMS 143- Final Project Outline

BankAccount.h

The BankAccount class is used as the base class for Savings & Checking Accounts. It contains private members for monthly interest and monthly interest rate, and a private member function for calculating interest based on those two private members. Another private member function it contains is for returning the monthly service charge dollar amount. The class also has protected members for the starting balance amount, the current balance amount, the annual interest rate, the number of deposits, the number of withdrawals(which I misspelled throughout the entire program...) and the monthly service charge amount. BankAccount has a no-arg constructor that sets all its members to 0 and sets the decimal precision to 2 and fixed, and it has a destructor. There are public member functions for setting the balance, which sets the current and starting balance amounts, for setting the annual interest rate, for returning the starting balance, for returning the number of deposits, for returning the number of withdrawals, and for returning the current balance. It also has several virtual public member functions, which are to be overridden in the classes that derive from this one. Among these virtual member functions are one for withdrawing from the balance and incrementing the withdrawal number, one for depositing to the balance and incrementing the deposit number, and one for running the monthly procedure which calls to the function for returning monthly charges, subtracts the monthly charges from the balance and then calls the function for calculating interest to determine the final monthly balance.

SavingsAccount.h

The SavingsAccount class derives from the BankAccount base class. It contains a private member for a boolean flag used to determine the current status of activity for the account. There is also contains a no-arg constructor that sets the flag to true(active) and calls the base class constructor, and it has a destructor. SavingsAccount also contains a withdraw member function overriding the one from the base class, checking if the balance is under $25 and if so sets the flag to false, making the account inactive (if the class is inactive then no further withdrawals can be made until the balance is over $25). However, if the balance is over $25 then the withdraw function calls to the base class's withdraw function, to subtract from the balance and increment the withdrawal number. This class also includes a deposit member functions overriding the one from the base class, checking if the account is currently inactive and if the deposit will raise the balance back over $25, and if so sets the account back to active. It also calls to the base class's deposit function, to add to the balance and increment the deposit number. There is also a monthly procedure member function overriding the one from the base class, checking the withdrawal numbers and adding $1 to the monthly charges for every withdrawal over 4. Then, the base class's monthly procedure member function is called, which subtracts the monthly service charges from the balance and calls to calculate the monthly interest to add to the balance. Afterwards, the derived member function checks if the balance went under 25 after the monthly charges were subtracted from it, and if so sets the account status to inactive.

<u>CheckingAccount.h</u>

The CheckingAccount class derives from the BankAccount base class. It contains a private member with a constant value of 15, for the service charge deducted if a withdrawal is made that would make the balance go under $0. There is also contains a no-arg constructor that calls the base class constructor, and it has a destructor. CheckingAccount also contains a withdraw member function overriding the one from the base class, checking if the balance is currently under $0, which prevents further withdrawals from being made. It also checks if the withdrawal would make the current balance go under $0 and if so deducts a $15 service charge from the balance. If the deducted service charge causes the balance to go under $0, then the negative amount is owed. However, if the balance is not currently under $0 and the withdrawal would not make the current balance go under $0, then the withdraw function calls to the base class's withdraw function, to subtract from the balance and increment the withdrawal number. There is also a monthly procedure member function overriding the one from the base class, adding a $5 monthly service charge and further charges of $0.10 per withdrawal made. Then, the base class's monthly procedure member function is called, which subtracts the monthly service charges from the balance and calls to calculate the monthly interest to add to the balance.

<u>BankDriver</u>

The driver for this program has a Person structure with a name member and a SavingsAccount object and CheckingAccount object. The main method has everything inside a while that is controlled by a restart flag that determines at the end whether to restart the program or not based on user input. The program first calls to a function that asks the user for the number of accounts to be made, this is stored in the variable count, which will be used as the size for a dynamically allocated array of Person structures. It then calls to another function that asks the user for the annual interest rate, this is stored in the variable aRate. Next, the dynamically allocated array is created, and a for loop is entered that will end once the index is one less than the count(looping through each account holder's Savings & Checking accounts). Within the for loop, the user is asked for the name of the first account holder in the array, which affects the name member of the structure. A do while loop is then entered asking the user for the balance of the SavingsAccount object for the first account holder in the array. Then, the annual interest rate is set in that object with aRate. Another do while loop is then entered, which functions as a menu that allows deposits and withdrawals to be made to the SavingsAccount object, and allows for the monthly procedure to be run to leave the savings account menu and display its information. After the menu is left, then a do while loop is entered asking for the balance of the CheckingAccount object for the first account holder in the array. Then, the annual interest rate is set in that object with aRate. Another do while loop is then entered, which functions as a menu that allows deposits and withdrawals to be made to the CheckingAccount object, and allows for the monthly procedure to be run to leave the checking account menu and display its information. After the for loop does this for all the account holders in the array, a function is called, passing the restart flag by reference, and asks the user whether they wish to restart the program from the start or end it. Regardless of the answer, once that function is left, the dynamically allocated memory is freed, and then the program either restarts or terminates.