**Objective**

We’re going to utilize what we have learned about inheritance to create a banking app. The architecture of the main menu system is mostly up to you as long as each of the account types can be withdrawn from or deposited to.

Concepts we will be using / reinforcing in this lab:

* Inheritance
* Text file input/output

**BaseAccount.h/cpp – 20 points**

Create a BaseAccount class. This will be our base class that will serve as our blueprint for our derived classes. Give the class a protected float data member for the account balance and a protected int for the number of withdrawals (Initialize these both to 0 in either the class member declaration area or in a default constructor). Also add the following public methods to the class:

void Withdraw(float amount); // decreases balance by amount. Does NOT “move” the money anywhere, simply reduces the balance and adds 1 to number of withdrawals

void Deposit(float amount); // increases balance by amount

float GetBalance() const; // returns the balance

Withdraw in BaseAccount will only decrease the balance. All other logic will exist in the child classes.

**CheckingAccount.h/cpp – 20 points**

Create this class that publicly derives from BaseAccount. It has no additional data members and must override the Withdraw function to enforce these rules:

* Checking account charges a $5 fee if more than 10 withdrawals are made

Remember that when overriding a function the original base function is no longer called, so be sure to either call the base method (BaseAccount::Withdraw()) or manually decrease the balance again.

**SavingsAccount.h/cpp – 20 points**

Create this class that publicly derives from BaseAccount. It has no additional data members and must override the Withdraw function to enforce these rules:

* Savings account will not allow more than 3 withdrawals

**CreditAccount.h/cpp – 20 points**

Create this class that publicly derives from BaseAccount. Add a private int data member for the amount spent so far to be able to compare to the constant spending limit of 40. Override the Withdraw function to enforce these rules:

* Credit accounts have a spending $40 limit and charge a $5000 fee if you go over that spending limit

**Main.cpp – 20 points**

Main will need three variables of types CheckingAccount, SavingsAccount, and CreditAccount. Initialize their balance amounts to whatever you want. Then add a menu system that will allow for deposits and withdrawals to be made to the various accounts. Show on-screen feedback for all transactions such as “$150 deposited to CheckingAccount!”.

Finally, implement use of the ofstream class at the end of your program to write out to a text file the three balance amounts for each of the account variables. Remember the ordering so that when you write the input code the balances will go to the correct account variable.

Example file contents:  
1000.5  
960  
12.75

Confirm your output code is working by opening the file in a text editor to see if its contents are what you expect it to be. The file will be in the same directory as your .vcxproj file (the project, not solution).

If all is well, implement use of the ifstream class at the beginning of your main so that, if the text file exists (.is\_open()), the contents will be deposited into your three account variables. If the file does not exist, keep the original initialization deposit code you had before.

HINT: Text file streams work exactly the same as cin (ifstream) and cout (ofstream).