**Logo

Description automatically generated San Francisco Bay University**

**CS360L - Programming in C and C++ Lab**

**Lab Assignment #8**

**Due day: 07/19/2022**

**Instruction:**

1. **Push the answer sheets/source code to Github**
2. **Please follow the code style rule like programs on handout.**
3. **Overdue lab assignment submission can’t be accepted.**

**4. Take academic honesty and integrity seriously (Zero Tolerance of Cheating & Plagiarism)**

1. Create an inheritance hierarchy that a bank might use to represent customers’ bank accounts. All customers at this bank can deposit (i.e., credit) money into their accounts and withdraw (i.e., debit) money from their accounts. More specific types of accounts also exist. Savings accounts, for instance, earn interest on the money they hold. Checking accounts, on the other hand, charge a fee per transaction (i.e., credit or debit).

Create an inheritance hierarchy containing base class *Account* and derived classes *SavingsAccount* and *CheckingAccount* that inherit from class *Account*. Base class *Account* should include one data member of type *double* to represent the account balance. The class should provide a constructor that receives an initial balance and uses it to initialize the data member. The constructor should validate the initial balance to ensure that it’s greater than or equal to *0.0*. If not, the balance should be set to *0.0* and the constructor should display an error message, indicating that the initial balance was invalid. The class should provide three member functions. Member function *credit* should add an amount to the current balance. Member function *debit* should withdraw money from the *Account* and ensure that the debit amount does not exceed the *Account’s* balance. If it does, the balance should be left unchanged and the function should print the message *"Debit amount exceeded account balance."* Member function *getBalance* should return the current balance.

Derived class *SavingsAccount* should inherit the functionality of an *Account*, but also include a data member of type *double* indicating the interest rate (percentage) assigned to the *Account*. *SavingsAccount’s* constructor should receive the initial balance, as well as an initial value for the *SavingsAccount’s* interest rate. *SavingsAccount* should provide a *public* member function *calculateInterest* that returns a *double* indicating the amount of interest earned by an account. Member function *calculateInterest* should determine this amount by multiplying the interest rate by the account balance. [*Note: SavingsAccount* should inherit member functions *credit* and *debit* as is without redefining them.]

Derived class *CheckingAccount* should inherit from base class *Account* and include an additional data member of type *double* that represents the fee charged per transaction. *CheckingAccount’s* constructor should receive the initial balance, as well as a parameter indicating a fee amount. Class *CheckingAccount* should redefine member functions *credit* and *debit* so that they subtract the fee from the account balance whenever either transaction is performed successfully. *CheckingAccount’s* versions of these functions should invoke the base-class *Account* version to perform the updates to an account balance. *CheckingAccount’s debit* function should charge a fee only if money is actually withdrawn (i.e., the debit amount does not exceed the account balance). [*Hint:* Define *Account’s debit* function so that it returns a *bool* indicating whether money was withdrawn. Then use the return value to determine whether a fee should be charged.]

After defining the classes in this hierarchy, write a program that creates objects of each class and tests their member functions. Add interest to the *SavingsAccount* object by first invoking its *calculateInterest* function, then passing the returned interest amount to the object’s *credit* function.