Department of Computer Science, New Campus
UNIVERSITY OF ENGINEERING AND TECHNOLOGY, LAHORE

Assignment

Submission date: 14/05/2025

Bank Accounts

This program should be designed and written by a group of 3 students. Here are some details:

- The requirements of the program should be analyzed.
- The parameters and return types of each function and class member function should be decided in advance.
- The program will be best implemented as a multi-file program.

Design a generic class to hold the following information about a bank account:

1. Balance
2. Number of deposits this month
3. Number of withdrawals
4. Annual interest rate
5. Monthly service charges

The class should have the following member functions:

A. **Constructor**: Accepts arguments for the balance and annual interest rate.
B. **deposit:** A virtual function that accepts an argument for the amount of the deposit. The function should add the argument to the account balance. It should also increment the variable holding the number of deposits.
C. **withdraw:** A virtual function that accepts an argument for the amount of the withdrawal. The function should subtract the argument from the balance. It should also increment the variable holding the number of withdrawals.
D. **calcInt:** A virtual function that updates the balance by calculating the monthly interest earned by the account, and adding this interest to the balance. This is performed by the following formulas:
    a. Monthly Interest Rate = (Annual Interest Rate / 12)
    b. Monthly Interest = Balance * Monthly Interest Rate
    c. Balance = Balance + Monthly Interest

E. **monthlyProc:** A virtual function that subtracts the monthly service charges from the balance, calls the calcInt function, and then sets the variables that hold the number of withdrawals, number of deposits, and monthly service charges to zero.

Next, design a savings account class, derived from the generic account class. The savings account class should have the following additional member:

- status (to represent an active or inactive account)

If the balance of a savings account falls below $25, it becomes inactive. (The status member could be a flag variable.) No more withdrawals may be made until the balance is raised above $25, at which time the account becomes active again. The savings account class should have the following member functions:

A. **withdraw:** A function that checks to see if the account is inactive before a withdrawal is made. (No withdrawal will be allowed if the account is not active.) A withdrawal is then made by calling the base class version of the function.
B. **deposit:** A function that checks to see if the account is inactive before a deposit is made. If the account is inactive and the deposit brings the balance above $25, the account becomes active again. The deposit is then made by calling the base class version of the function.
C. **monthlyProc:** Before the base class function is called, this function checks the number of withdrawals. If the number of withdrawals for the month is more than 4, a service charge of $1 for each withdrawal above 4 is added to the base class variable that holds the monthly service charges. (Don't forget to check the account balance after the service charge is taken. If the balance falls below $25, the account becomes inactive.)

Next, design a checking account class, also derived from the generic account class. It should have the following member functions:

A. **withdraw:** Before the base class function is called, this function will determine if a withdrawal (a check written) will cause the balance to go below $0. If the balance goes below $0, a service charge of $15 will be taken from the account. (The withdrawal will not be made.) If there isn't enough in the account to pay the service charge, the balance will become negative and the customer will owe the negative amount to the bank.
B. **monthlyProc:** Before the base class function is called, this function adds the monthly fee of $5 plus $0.10 per withdrawal (check written) to the base class variable that holds the monthly service charges.

Question 1 [CLO2, 2 marks]

Define and implement the classes according to the principles of object-oriented programming. Each class must include appropriate constructors, getters (accessors), and setters (mutators).

Question 2 [CLO3, 2 marks]

In the main function, dynamically create objects of each class. The main function should demonstrate interaction between these objects by prompting the user to enter deposit and withdrawal amounts for a savings account and a checking account. The program should then display the following monthly statistics: beginning balance, total deposits, total withdrawals, service charges, and ending balance.

Question 3 [CLO4, 2 marks]

Overload the following operators for each class:

A. Assignment operator (=)
B. Input stream operator (>>)
C. Output stream operator (<<)

Question 4 [CLO5, 2 marks]

Implement a proper inheritance hierarchy for the classes.

Question 5 [CLO6, 2 marks]

Demonstrate proper polymorphism in your implementation.