CPS235 Object Oriented Programming in C++

LAB4 BESE-15A *22nd March 2010*

Inheritance in C++

Objectives

By the end of this lab, you should be able to

- Understand Inheritance in C++
- Use composition and inheritance

Instructions

- If you have any problems, you are encouraged to consult with me.
- The lab should be submitted to the following folder.
  \\csdept\data\Assignments\Lec Aisha Khalid\OOP\15A\Lab4
- Make sure that you show your programs to me and answer any questions that I may have in order to get full credit for the lab.

Question 1: Account Inheritance Hierarchy *(To be submitted)*

Create an inheritance hierarchy containing base class **`Account`** and derived classes **`SavingsAccount`** and **`CheckingAccount`** that inherit from class **`Account`**. Make sure that all arguments and member functions are declared const where appropriate.

a. 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 is 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:

   i. Member function Credit should add an amount to the current balance

   ii. 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."

   iii. Member function getBalance should return the current balance.

b. Derived class SavingsAccount should inherit the functionality of an Account, but also include a data member of type double indicating the yearly 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. (Make sure to call the base class constructor appropriately.)

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.]

c. 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., if 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 in the* CheckingAccount *version of the function to determine whether a fee should be charged.]*

d. 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.

Question 2: Inheritance and Composition

Given the following **declaration** for a TestScore class, write the implementation of the member functions.

```
class TestScore
{
      private:
       string stuName;
       int stuScore;
   public:
       TestScore();
       TestScore(string name, int score);
       string GetName()const;    //returns stuName
       int GetScore() const;     //returns stuScore
       void display();           //displays stuName and stuScore
};
```

a. Write a derived class **declaration** called IDScore that adds an integer stuID as a private member, and that supplies
   1. A default constructor
   2. A constructor whose parameters correspond to the three members (stuName, stuScore and stuID) and
   3. A GetID method which returns stuID
   4. A display function which overrides the display function of the TestScore class, calls TestScore's display function in the function body and also displays the stuID

b. Write the implementation for the IDScore class.

c. Write the specification and implementation for a class called Exam that uses composition to create an array of 10 objects of class IDScore. The class can use the default constructor provided by the compiler. The class has a function that assigns an IDSCore object to a location in the array, given the object and the location as

parameters. The class should also have a function that returns the IDScore object at the position specified by its parameter.

d. You can write code of your choice in main to test these classes.

Question 3: Test Inheritance Types

Look at the given program (testInheritanceTypes.cpp) and modify the code to test all three types of inheritance – public, private and protected. This program contains three classes (Mother, Daughter and GrandDaughter) and the main function. Change the values of the inheritance types and see what values can be accessed at the various levels of derived classes. When you change inheritance types, the program could cause errors while compiling. Understand how the inheritance types affect those errors and comment out/change the appropriate statements to compile correctly.