Laboratory 11: Chapter 9 Classes: A deeper look, Part 1
Programming Methodologies Lab (14:332:254)
April 16, 2018

Problem Definition

Create an inheritance hierarchy containing base class Account and derived class Savings-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 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. 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.]*

Part 1: Class Definition

Implement the class definition for class Account and *SavingsAccount* in the files called Account.h and SavingsAccount.h. The class definition should contain only the prototypes of the member functions and double members.

Part 2.1: Member-function definition

Implement the member-function definition of classes in the files called Account.cpp and SavingsAccount.cpp. This .cpp file contains the actual implementation of the member functions of the two classes.

Part 2.2: Test program

Write a program that tests your class Account and SavingAcounts. Call your file testAccount.cpp. This program should:

1. Instantiate an object of class SavingAccounts and Accounts.
2. Add some credit to both objects.
3. Add some credit to both objects.
4. Withdraw some amount for both objects.
5. Print the interest for the saving account object.
6. Print the current balance of each account.

Grading

Part 1: Class Definition	30%	Required
Part 2: Member-function definition and test program	70%	Required

Remember to include following information at the beginning of testAccount.cpp file as a comment.

```
// Course Number and section: 14:332:254:XX
// Lab Instructor: Kazem Cheshmi
// Date Performed: 04/XX/2018
// Date Submitted: 04/XX/2018
// Submitted by: YOUR NAME, RUID
```

NOTE – You need to **Upload** your files and then **Submit** them. The TAs will have no access to your files if you forget to submit them.

If you do not submit your files to sakai, you will not receive a grade.

Software Copying Policy

If your lab assignment is found to be a copy of another student's lab, you will not be given credit for this assignment. If this happens more than once, you may be in jeopardy of failing the course/lab.

The Software Copying Policy can be found on sakai under “Course Content and Related Materials /Calendar & Course Information”.