**作业五（习题12.10，Account类）**

**1．问题描述**

创建一个与银行账户相关的类继承层次。银行的所有账户都可以存款和取款。存款能够产生一定的利息，查询和取款交易要缴纳一定的手续费。

要求：基类：Account(参考实验提示)；

派生类：SavingAccount和CheckingAccount；

SavingAccount：继承Account的成员函数；构造函数接收两个参数：存款初始值（initialBalance）和利率（rate）；增加一个数据成员：利率(interestRate)，增加public类型的成员函数用于计算利率(calculateInterest())。

CheckingAccount：构造函数应接收到两个参数，一个是存款初始值（initialBalance），一个是手续费（fee）；增加一个数据成员：手续费（transactionFee）；重新定义成员函数credit()和debit()，以便能够从存款余额中减去手续费，要求成员函数通过调用基类的成员函数来更新存款数目，debit()函数应该在确定取款之后才能扣除手续费。

**2. 实验提示**

**1）类定义示例：**

#ifndef ACCOUNT\_H

#define ACCOUNT\_H

class Account

{

public:

 Account( double ); // constructor initializes balance

 void credit( double ); // add an amount to the account balance

 bool debit( double ); // subtract an amount from the account balance

 void setBalance( double ); // sets the account balance

 double getBalance(); // return the account balance

private:

 double balance; // data member that stores the balance

}; // end class Account

#endif

**// Account.cpp**

#include <iostream>

using std::cout;

using std::endl;

#include "Account.h" // include definition of class Account

// Account constructor initializes data member balance

Account::Account( double initialBalance )

{

 // if initialBalance is greater than or equal to 0.0, set this value

 // as the balance of the Account

 if ( initialBalance >= 0.0 )

 balance = initialBalance;

 else // otherwise, output message and set balance to 0.0

 {

 cout << "Error: Initial balance cannot be negative." << endl;

 balance = 0.0;

 } // end if...else

} // end Account constructor

// credit (add) an amount to the account balance

void Account::credit( double amount )

{

 balance = balance + amount; // add amount to balance

} // end function credit

// debit (subtract) an amount from the account balance

// return bool indicating whether money was debited

bool Account::debit( double amount )

{

 if ( amount > balance ) // debit amount exceeds balance

 {

 cout << "Debit amount exceeded account balance." << endl;

 return false;

 } // end if

 else // debit amount does not exceed balance

 {

 balance = balance - amount;

 return true;

 } // end else

} // end function debit

// set the account balance

void Account::setBalance( double newBalance )

{

 balance = newBalance;

} // end function setBalance

// return the account balance

double Account::getBalance()

{

 return balance;

} // end function getBalance

**2）测试函数示例**

int main()

{

 Account account1( 50.0 ); // create Account object

 SavingsAccount account2( 25.0, .03 ); // create SavingsAccount object

 CheckingAccount account3( 80.0, 1.0 ); // create CheckingAccount object

 cout << fixed << setprecision( 2 );

 // display initial balance of each object

 cout << "account1 balance: $" << account1.getBalance() << endl;

 cout << "account2 balance: $" << account2.getBalance() << endl;

 cout << "account3 balance: $" << account3.getBalance() << endl;

 cout << "\nAttempting to debit $25.00 from account1." << endl;

 account1.debit( 25.0 ); // try to debit $25.00 from account1

 cout << "\nAttempting to debit $30.00 from account2." << endl;

 account2.debit( 30.0 ); // try to debit $30.00 from account2

 cout << "\nAttempting to debit $40.00 from account3." << endl;

 account3.debit( 40.0 ); // try to debit $40.00 from account3

 // display balances

 cout << "\naccount1 balance: $" << account1.getBalance() << endl;

 cout << "account2 balance: $" << account2.getBalance() << endl;

 cout << "account3 balance: $" << account3.getBalance() << endl;

 cout << "\nCrediting $40.00 to account1." << endl;

 account1.credit( 40.0 ); // credit $40.00 to account1

 cout << "\nCrediting $65.00 to account2." << endl;

 account2.credit( 65.0 ); // credit $65.00 to account2

 cout << "\nCrediting $20.00 to account3." << endl;

 account3.credit( 20.0 ); // credit $20.00 to account3

 // display balances

 cout << "\naccount1 balance: $" << account1.getBalance() << endl;

 cout << "account2 balance: $" << account2.getBalance() << endl;

 cout << "account3 balance: $" << account3.getBalance() << endl;

 // add interest to SavingsAccount object account2

 double interestEarned = account2.calculateInterest();

 cout << "\nAdding $" << interestEarned << " interest to account2."

 << endl;

 account2.credit( interestEarned );

 cout << "\nNew account2 balance: $" << account2.getBalance() << endl;

 return 0;

} // end main

**3．结果示例**

****