#ifndef ACCOUNT\_H

#define ACCOUNT\_H

class Account

{

public:

Account( double );

virtual void credit( double );

virtual bool debit( double );

void setBalance( double );

double getBalance();

private:

double balance;

};

#endif

#include <iostream>

#include "Account.h"

using namespace std;

Account::Account( double initialBalance )

{

if ( initialBalance >= 0.0 )

balance = initialBalance;

else

{

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

balance = 0.0;

}

}

void Account::credit( double amount )

{

balance = balance + amount;

}

bool Account::debit( double amount )

{

if ( amount > balance )

{

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

return false;

}

else

{

balance = balance - amount;

return true;

}

}

void Account::setBalance( double newBalance )

{

balance = newBalance;

}

double Account::getBalance()

{

return balance;

}

#ifndef CHECKING\_H

#define CHECKING\_H

#include "Account.h"

class CheckingAccount : public Account

{

public:

CheckingAccount( double, double );

virtual void credit( double );

virtual bool debit( double );

private:

double transactionFee;

void chargeFee();

};

#endif

#include <iostream>

#include "CheckingAccount.h"

using namespace std;

CheckingAccount::CheckingAccount( double initialBalance, double fee )

: Account( initialBalance ) // initialize base class

{

transactionFee = ( fee < 0.0 ) ? 0.0 : fee;

}

void CheckingAccount::credit( double amount )

{

Account::credit( amount );

chargeFee();

}

bool CheckingAccount::debit( double amount )

{

bool success = Account::debit( amount ); // attempt to debi

if ( success )

{

chargeFee();

return true;

}

else

return false;

}

void CheckingAccount::chargeFee()

{

Account::setBalance( getBalance() - transactionFee );

cout << "$" << transactionFee << " transaction fee charged." << endl;

}

#ifndef SAVINGS\_H

#define SAVINGS\_H

#include "Account.h"

class SavingsAccount : public Account

{

public:

SavingsAccount( double, double );

double calculateInterest();

private:

double interestRate;

};

#endif

#include "SavingsAccount.h"

SavingsAccount::SavingsAccount( double initialBalance, double rate )

: Account( initialBalance )

{

interestRate = ( rate < 0.0 ) ? 0.0 : rate;

}

double SavingsAccount::calculateInterest()

{

return getBalance() \* interestRate;

}