**Lab Exercises**

Class: 软件工程1802 Name: 杨靖仁 StuID:201816040212

Date: 2019-12-04 Score:

**Lab Exercise — Polymorphic Banking**

**I Lab Objectives**

In this lab, you will practice:

1. Creating an Account base class that contains virtual functions and derived classes SavingsAccount and CheckingAccount.
2. Defining virtual functions.
3. Calling virtual functions.
4. Downcasting with a pointer with the dynamic\_cast operator.

**II Description of the Problem (译文见教材P419 12.14)**

Develop a polymorphic banking program using the Account hierarchy created in Exercise 11.10. Create a vector of Account pointers to SavingsAccount and CheckingAccount objects. For each Account in the vector, allow the user to specify an amount of money to withdraw from the Account using member function debit and an amount of money to deposit into the Account using member function credit. As you process each Account, determine its type. If an Account is a SavingsAccount, calculate the amount of interest owed to the Account using member function calculateInterest, then add the interest to the account balance using member function credit. After processing an Account, print the updated account balance obtained by invoking base class member function getBalance.

**III Sample Output**



**IV Problem-Solving Tips**

1. To achieve polymorphism, declare the functions that should be called polymorphically as virtual. To indicate a virtual function within a class definition, add “virtual” before the function prototype. When the virtual functions are redefined in a derived class, those member function prototypes should also be preceded by the keyword virtual as a good programming practice.
2. To determine if a pointer to an Account object is actually pointing to a SavingsAccount object, downcast it to a SavingsAccount \* using the dynamic\_cast operator. If the pointer returned by this operation is not the null pointer (i.e., 0) then the object is a SavingsAccount object and that pointer can be used to access members unique to class SavingsAccount.
3. Remember that your compiler may require you to enable run-time type information (RTTI) for this particular project before this program will run correctly.

**V Your Solution**

// Lab 1: polymorphicBanking.cpp

// Processing Accounts polymorphically.

#include <iostream>

#include <iomanip>

#include <vector>

**using** **namespace** std;

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

#include "SavingsAccount.h" // SavingsAccount class definition

#include "CheckingAccount.h" // CheckingAccount class definition

**int** main()

{

// create vector accounts

/\* Write declarations for a vector of four pointers

to Account objects, called accounts \*/

vector<Account\*> accounts(4);

// initialize vector with Accounts

accounts[ 0 ] = **new** SavingsAccount( 25.0, .03 );

accounts[ 1 ] = **new** CheckingAccount( 80.0, 1.0 );

accounts[ 2 ] = **new** SavingsAccount( 200.0, .015 );

accounts[ 3 ] = **new** CheckingAccount( 400.0, .5 );

cout << fixed << setprecision( 2 );

// loop through vector, prompting user for debit and credit amounts

**for** ( size\_t i = 0; i < accounts.size(); i++ )

{

cout << "Account " << i + 1 << " balance: $"

<< accounts[i]->getBalance()<<endl/\* Call the getBalance function through Account pointer i \*/;

**double** withdrawalAmount = 0.0;

cout << "\nEnter an amount to withdraw from Account " << i + 1

<< ": ";

cin >> withdrawalAmount;

/\* Call the debit function through Account pointer i \*/

accounts[i]->debit(withdrawalAmount);

**double** depositAmount = 0.0;

cout << "Enter an amount to deposit into Account " << i + 1

<< ": ";

cin >> depositAmount;

/\* Call the credit function through Account pointer i \*/

accounts[i]->credit(depositAmount);

// downcast pointer

SavingsAccount \*savingsAccountPtr =**dynamic\_cast**<SavingsAccount\*>(accounts[i]);

/\* Write a dynamic\_cast operation to to attempt to downcast

Account pointer i to a SavingsAccount pointer \*/

// if Account is a SavingsAccount, calculate and add interest

**if** (**nullptr**!=savingsAccountPtr /\* Write a test to determine if savingsAccountPtr isn't 0 \*/ )

{

**double** interestEarned = savingsAccountPtr->calculateInterest()/\* Call member function calculateInterest

through savingsAccountPtr \*/;

cout << "Adding $" << interestEarned << " interest to Account "

<< i + 1 << " (a SavingsAccount)" << endl;

/\* Use the credit function to credit interestEarned to

the SavingsAccount pointed to by savingsAccountPtr\*/

savingsAccountPtr->credit(interestEarned);

} // end if

cout << "Updated Account " << i + 1 << " balance: $"

<< /\* Call the getBalance function through Account pointer i \*/

accounts[i]->getBalance()<< "\n\n";

} // end for

} // end main

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* (C) Copyright 1992-2012 by Deitel & Associates, Inc. and \*

\* Pearson Education, Inc. All Rights Reserved. \*

\* \*

\* DISCLAIMER: The authors and publisher of this book have used their \*

\* best efforts in preparing the book. These efforts include the \*

\* development, research, and testing of the theories and programs \*

\* to determine their effectiveness. The authors and publisher make \*

\* no warranty of any kind, expressed or implied, with regard to these \*

\* programs or to the documentation contained in these books. The authors \*

\* and publisher shall not be liable in any event for incidental or \*

\* consequential damages in connection with, or arising out of, the \*

\* furnishing, performance, or use of these programs. \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Lab 1: Account.h

// Definition of Account class.

#ifndef ACCOUNT\_H

#define ACCOUNT\_H

**class** Account

{

**public**:

Account( **double** ); // constructor initializes balance

/\* Write a function prototype for virtual function credit \*/

**virtual** **void** credit( **double**);

/\* Write a function prototype for virtual function debit \*/

**virtual** **bool** debit( **double** );

**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

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* (C) Copyright 1992-2012 by Deitel & Associates, Inc. and \*

\* Pearson Education, Inc. All Rights Reserved. \*

\* \*

\* DISCLAIMER: The authors and publisher of this book have used their \*

\* best efforts in preparing the book. These efforts include the \*

\* development, research, and testing of the theories and programs \*

\* to determine their effectiveness. The authors and publisher make \*

\* no warranty of any kind, expressed or implied, with regard to these \*

\* programs or to the documentation contained in these books. The authors \*

\* and publisher shall not be liable in any event for incidental or \*

\* consequential damages in connection with, or arising out of, the \*

\* furnishing, performance, or use of these programs. \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Lab 1: Account.cpp

// Member-function definitions for class Account.

#include <iostream>

**using** **namespace** std;

#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

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* (C) Copyright 1992-2012 by Deitel & Associates, Inc. and \*

\* Pearson Education, Inc. All Rights Reserved. \*

\* \*

\* DISCLAIMER: The authors and publisher of this book have used their \*

\* best efforts in preparing the book. These efforts include the \*

\* development, research, and testing of the theories and programs \*

\* to determine their effectiveness. The authors and publisher make \*

\* no warranty of any kind, expressed or implied, with regard to these \*

\* programs or to the documentation contained in these books. The authors \*

\* and publisher shall not be liable in any event for incidental or \*

\* consequential damages in connection with, or arising out of, the \*

\* furnishing, performance, or use of these programs. \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Lab 1: CheckingAccount.h

// Definition of CheckingAccount class.

#ifndef CHECKING\_H

#define CHECKING\_H

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

**class** CheckingAccount : **public** Account

{

**public**:

// constructor initializes balance and transaction fee

CheckingAccount( **double**, **double** );

/\* Write a function prototype for virtual function credit,

which will redefine the inherited credit function \*/

**virtual** **bool** debit( **double** );

/\* Write a function prototype for virtual function debit,

which will redefine the inherited debit function \*/

**virtual** **void** credit( **double**);

**private**:

**double** transactionFee; // fee charged per transaction

// utility function to charge fee

**void** chargeFee();

}; // end class CheckingAccount

#endif

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* (C) Copyright 1992-2012 by Deitel & Associates, Inc. and \*

\* Pearson Education, Inc. All Rights Reserved. \*

\* \*

\* DISCLAIMER: The authors and publisher of this book have used their \*

\* best efforts in preparing the book. These efforts include the \*

\* development, research, and testing of the theories and programs \*

\* to determine their effectiveness. The authors and publisher make \*

\* no warranty of any kind, expressed or implied, with regard to these \*

\* programs or to the documentation contained in these books. The authors \*

\* and publisher shall not be liable in any event for incidental or \*

\* consequential damages in connection with, or arising out of, the \*

\* furnishing, performance, or use of these programs. \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Lab 1: CheckingAccount.cpp

// Member-function definitions for class CheckingAccount.

#include <iostream>

**using** **namespace** std;

#include "CheckingAccount.h" // CheckingAccount class definition

// constructor initializes balance and transaction fee

CheckingAccount::CheckingAccount( **double** initialBalance, **double** fee )

: Account( initialBalance ) // initialize base class

{

transactionFee = ( fee < 0.0 ) ? 0.0 : fee; // set transaction fee

} // end CheckingAccount constructor

// credit (add) an amount to the account balance and charge fee

**void** CheckingAccount::credit( **double** amount )

{

Account::credit( amount ); // always succeeds

chargeFee();

} // end function credit

// debit (subtract) an amount from the account balance and charge fee

**bool** CheckingAccount::debit( **double** amount )

{

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

**if** ( success ) // if money was debited, charge fee and return true

{

chargeFee();

**return** **true**;

} // end if

**else** // otherwise, do not charge fee and return false

**return** **false**;

} // end function debit

// subtract transaction fee

**void** CheckingAccount::chargeFee()

{

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

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

} // end function chargeFee

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* (C) Copyright 1992-2012 by Deitel & Associates, Inc. and \*

\* Pearson Education, Inc. All Rights Reserved. \*

\* \*

\* DISCLAIMER: The authors and publisher of this book have used their \*

\* best efforts in preparing the book. These efforts include the \*

\* development, research, and testing of the theories and programs \*

\* to determine their effectiveness. The authors and publisher make \*

\* no warranty of any kind, expressed or implied, with regard to these \*

\* programs or to the documentation contained in these books. The authors \*

\* and publisher shall not be liable in any event for incidental or \*

\* consequential damages in connection with, or arising out of, the \*

\* furnishing, performance, or use of these programs. \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Lab 1: SavingsAccount.h

// Definition of SavingsAccount class.

#ifndef SAVINGS\_H

#define SAVINGS\_H

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

**class** SavingsAccount : **public** Account

{

**public**:

// constructor initializes balance and interest rate

SavingsAccount( **double**, **double** );

**double** calculateInterest(); // determine interest owed

**private**:

**double** interestRate; // interest rate (percentage) earned by account

}; // end class SavingsAccount

#endif

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* (C) Copyright 1992-2012 by Deitel & Associates, Inc. and \*

\* Pearson Education, Inc. All Rights Reserved. \*

\* \*

\* DISCLAIMER: The authors and publisher of this book have used their \*

\* best efforts in preparing the book. These efforts include the \*

\* development, research, and testing of the theories and programs \*

\* to determine their effectiveness. The authors and publisher make \*

\* no warranty of any kind, expressed or implied, with regard to these \*

\* programs or to the documentation contained in these books. The authors \*

\* and publisher shall not be liable in any event for incidental or \*

\* consequential damages in connection with, or arising out of, the \*

\* furnishing, performance, or use of these programs. \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Lab 1: SavingsAccount.cpp

// Member-function definitions for class SavingsAccount.

#include "SavingsAccount.h" // SavingsAccount class definition

// constructor initializes balance and interest rate

SavingsAccount::SavingsAccount( **double** initialBalance, **double** rate )

: Account( initialBalance ) // initialize base class

{

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

} // end SavingsAccount constructor

// return the amount of interest earned

**double** SavingsAccount::calculateInterest()

{

**return** getBalance() \* interestRate;

} // end function calculateInterest

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* (C) Copyright 1992-2012 by Deitel & Associates, Inc. and \*

\* Pearson Education, Inc. All Rights Reserved. \*

\* \*

\* DISCLAIMER: The authors and publisher of this book have used their \*

\* best efforts in preparing the book. These efforts include the \*

\* development, research, and testing of the theories and programs \*

\* to determine their effectiveness. The authors and publisher make \*

\* no warranty of any kind, expressed or implied, with regard to these \*

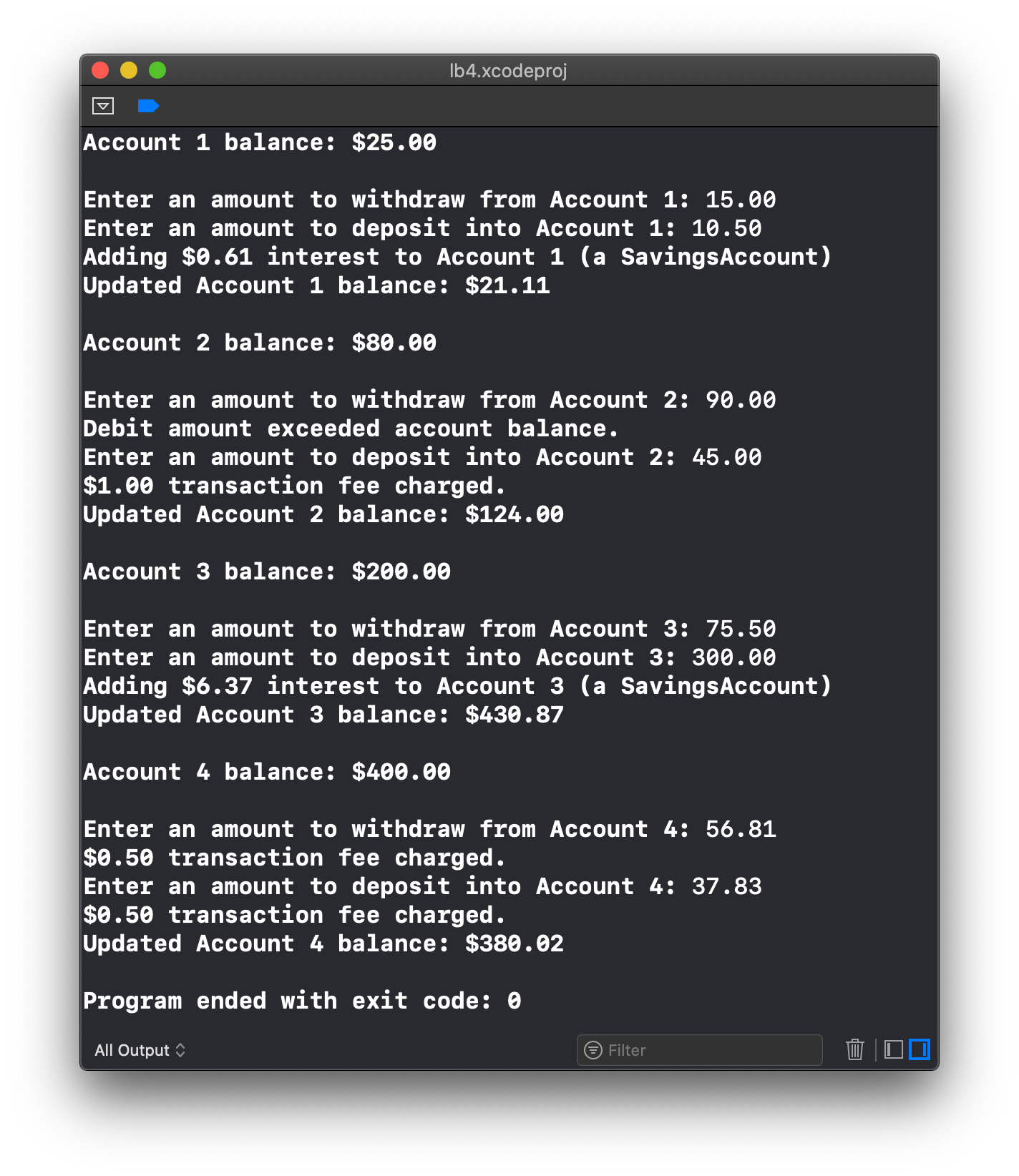
\* programs or to the documentation contained in these books. The authors \*

\* and publisher shall not be liable in any event for incidental or \*

\* consequential damages in connection with, or arising out of, the \*

\* furnishing, performance, or use of these programs. \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

****