**Lab Exercises**

Class: Name: NaHCOx

Date: Score:

**Lab Exercise 1 — Account Hierarchy**

**I 实验目标**

通过本次实验掌握:

1. 创建account类的继承层次关系，包括Account类，SavingsAccount类和CheckingAccount类。
2. 使用private数据成员限制对数据的访问。
3. 在派生类中重新定义基类的成员函数。

**II 问题描述 (注：中文版12章 12.10)**

Create an inheritance hierarchy that a bank might use to represent customers’ bank accounts. All customers at this bank can deposit (i.e., credit) money into their accounts and withdraw (i.e., debit) money from their accounts. More specific types of accounts also exist. Savings accounts, for instance, earn interest on the money they hold. Checking accounts, on the other hand, charge a fee per transaction (i.e., credit or debit).

Create an inheritance hierarchy containing base class Account and derived classes SavingsAccount and CheckingAccount that inherit from class 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 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. 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. If it does, the balance should be left unchanged and the function should print the message "Debit amount exceeded

account 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.]

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., 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 to determine whether a fee should be charged.]

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.

**III 输出样例**



**IV 答案**

本次代码量较多，下面以此给出每个文件的**主要部分**，再说明进行了哪些修改。**Account.h, Account.cpp** 仅作为基类，无修改

**SavingsAccount.h**

#include "Account.h"

class SavingsAccount: public Account

{

public:

    SavingsAccount( double, double );

    double calculateInterest();

private:

    double interestRate;

};

#endif

进行了以下改动：

包含基类Account的头文件Account.h

声明构造函数SavingsAccount( double, double );

声明成员函数calculateInterest();

声明double类型成员变量interestRate

**SavingsAccount.cpp**

给出头文件中声明的函数定义

#include "SavingsAccount.h"

SavingsAccount::SavingsAccount(double initBalance, double initialRate)

    : Account (initBalance), interestRate(initialRate) { }

double SavingsAccount::calculateInterest()  {

    return interestRate \* getBalance();

}

**CheckingAccount.h**

#include "Account.h"

class CheckingAccount: public Account

{

public:

    CheckingAccount( double, double );

    void credit( double );

    bool debit( double );

private:

    double transactionFee;

    void chargeFee();

};

#endif

进行了以下改动：

包含基类Account的头文件Account.h

重新声明了继承自基类的credit(double)和debit(double)函数

声明了private成员函数chargeFee()

声明了double成员变量transactionFee

**CheckingAccount.cpp**

给出头文件中声明的函数定义

CheckingAccount::CheckingAccount(double initBalance, double initFee)

    : Account(initBalance), transactionFee(initFee) { }

void CheckingAccount::credit( double amount )

{

    Account::credit(amount);

    chargeFee();

}

bool CheckingAccount::debit( double amount )

{

    if ( Account::debit(amount) )   {

        chargeFee();

        return true;

    }

    return false;

}

void CheckingAccount::chargeFee()

{

    if ( getBalance() >= 1 )    {

        Account::debit( transactionFee );

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

    } else {

        cout << "$" << getBalance() << " transaction fee charged." << endl;

        setBalance( 0 );

    }

}

**bankAccounts.cpp**

第53行，调用account2的calculateInterest()方法并初始化interestEarned变量

double interestEarned = account2.calculateInterest();

第57行，给account2账户增加利息

account2.credit(interestEarned);

**输出结果：**

account1 balance: $50.00

account2 balance: $25.00

account3 balance: $80.00

Attempting to debit $25.00 from account1.

Attempting to debit $30.00 from account2.

Debit amount exceeded account balance.

Attempting to debit $40.00 from account3.

$1.00 transaction fee charged.

account1 balance: $25.00

account2 balance: $25.00

account3 balance: $39.00

Crediting $40.00 to account1.

Crediting $65.00 to account2.

Crediting $20.00 to account3.

$1.00 transaction fee charged.

account1 balance: $65.00

account2 balance: $90.00

account3 balance: $58.00

Adding $2.70 interest to account2.

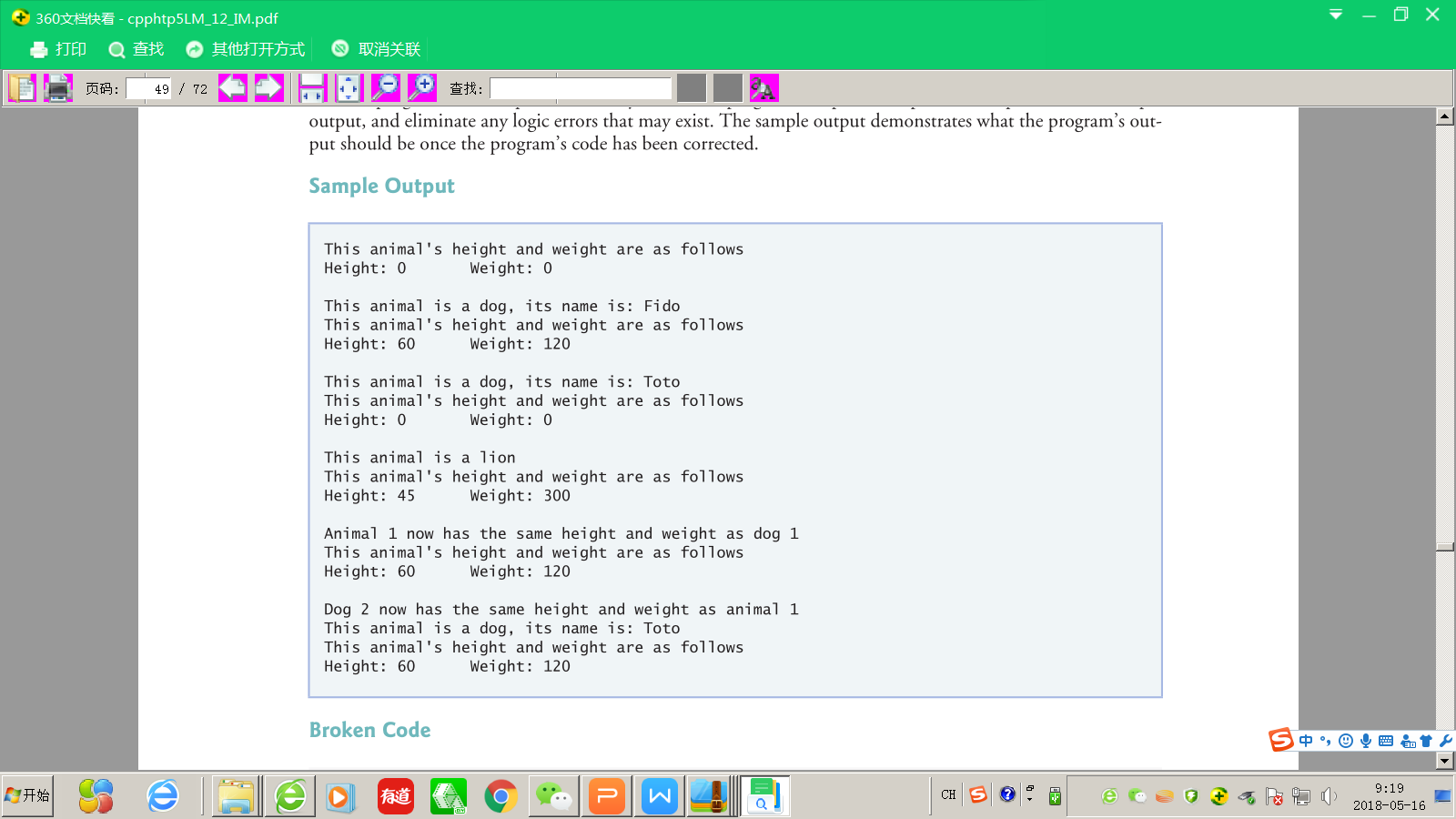
New account2 balance: $92.70

**Debugging**

**问题描述**

本部分程序不能正常运行，修改所有的语法错误使得程序能编译成功。运行程序，输出运行结果，并与下面的输出样例比较，修改所有的逻辑错误。

**输出样例**



**答案**

**Animal.h**与**Animal.cpp**

删除getName() const函数

**Dog.h**

class Dog : public Animal

{

public:

    Dog( const int = 0, const int = 0, string = "Toto" );

    void print() const;

    void setName( string );

    string getName() const;

    Dog& operator=( const Animal& o );

private:

    string name;

};

更正print()函数的大小写错误，使得可以覆盖基类的同名函数

声明getName()函数（将Animal类的同名函数移至这里），以返回字符串Name的值

重载了赋值符号，使得可以把基类对象赋值给Dog类对象

**Dog.cpp**

以下为改动部分：

string Dog::getName() const

{

    return name;

}

void Dog::setName( string n )

{

    name = n;

}

void Dog::print() const

{

    cout << "This animal is a dog, its name is: " << name << endl;

    Animal::print();

}

Dog& Dog::operator=( const Animal& o )

{

    setHeight(o.getHeight());

    setWeight(o.getWeight());

    return \*this;

}

给出getName()函数定义

setName()函数应该把n的值赋值给name而不是反之

print()函数除匹配头文件的改动外，调用的print()函数应该是Animal::print()而不是（缺省的）this->print()；原先的调用会导致无限递归，最终循环输出无法停止

给出赋值运算符重载的函数定义

**Lion.h**

没有继承自基类，应该改为

class Lion : public Animal

**Line.cpp**

类似Dog.cpp中print()的错误，这里也有同样的错误，应该改为

void Lion::print() const

{

    cout << "This animal is a lion\n";

    Animal::print();

}

**Debugging.cpp**

没有包含Dog.h头文件

#include "Dog.h"

原27行，缺少分号

cout << "Dog 2 now has the same height and weight as animal 1\n";

**输出结果：**

This animal's height and weight are as follows

Height: 0       Weight: 0

This animal is a dog, its name is: Fido

This animal's height and weight are as follows

Height: 60      Weight: 120

This animal is a dog, its name is: Toto

This animal's height and weight are as follows

Height: 0       Weight: 0

This animal is a lion

This animal's height and weight are as follows

Height: 45      Weight: 300

Animal 1 now has the same height and weight as dog 1

This animal's height and weight are as follows

Height: 60      Weight: 120

Dog 2 now has the same height and weight as animal 1

This animal is a dog, its name is: Toto

This animal's height and weight are as follows

Height: 60      Weight: 120