Lab 7 PDF

Name: Haichuan Wei

Lab: Lab 7

Date: 11/3/2021

Program Description:

This program was made to practice using C++ class. There is a main BankAccont.cpp file that the other files inherit from. The user will define data that will be plugged into each function and whichever account they want to change will be changed accordingly.

Program Source Code:

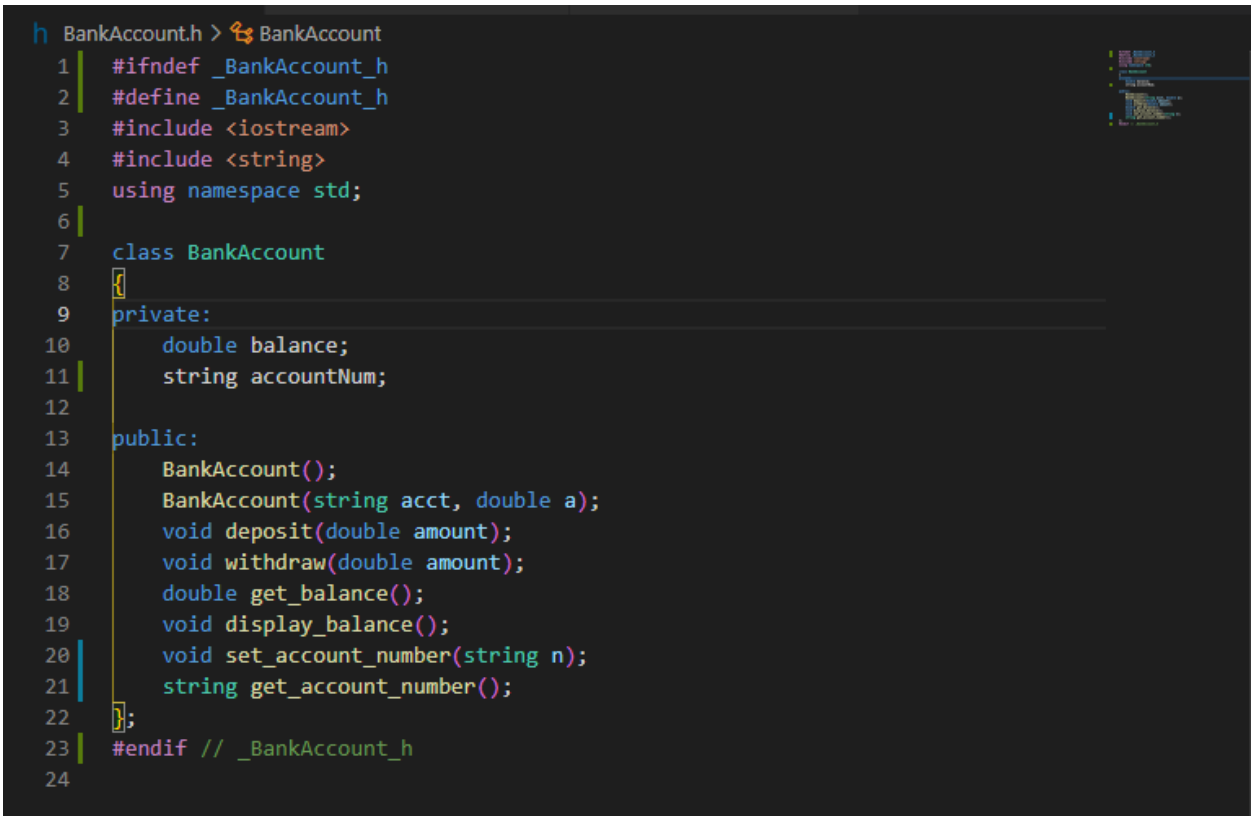
BankAccount.cpp

BankAccount.cpp > ...

```cpp
#include "BankAccount.h"
BankAccount::BankAccount()
{
    balance = 0;
    accountNum = "NA";
}

BankAccount::BankAccount(string acct, double a)
{
    accountNum = acct;
    balance = a;
}
void BankAccount::deposit(double amount)
{
    balance += amount;
}
void BankAccount::withdraw(double amount)
{
    if (amount > balance)
    {
        cout << "Insufficient funds! Trying to remove $" << amount << " But only $" << balance << "available! "
             << "Not removing anything!" << endl;
    }
    else
    {
        balance -= amount;
    }
}

double BankAccount::get_balance()
{
    return balance;
}
void BankAccount::display_balance()
{
    cout << accountNum << "'s balance is $" << balance << endl;
}
void BankAccount::set_account_number(string n)
{
    accountNum = n;
}
string BankAccount::get_account_number()
{
    return accountNum;
}

```

Lab 7 PDF

BankAccount.h



```h
BankAccount.h > BankAccount
#ifndef _BankAccount_h
#define _BankAccount_h
#include <iostream>
#include <string>
using namespace std;

class BankAccount
{
private:
    double balance;
    string accountNum;

public:
    BankAccount();
    BankAccount(string acct, double a);
    void deposit(double amount);
    void withdraw(double amount);
    double get_balance();
    void display_balance();
    void set_account_number(string n);
    string get_account_number();
};
#endif // _BankAccount_h

```

Lab 7 PDF

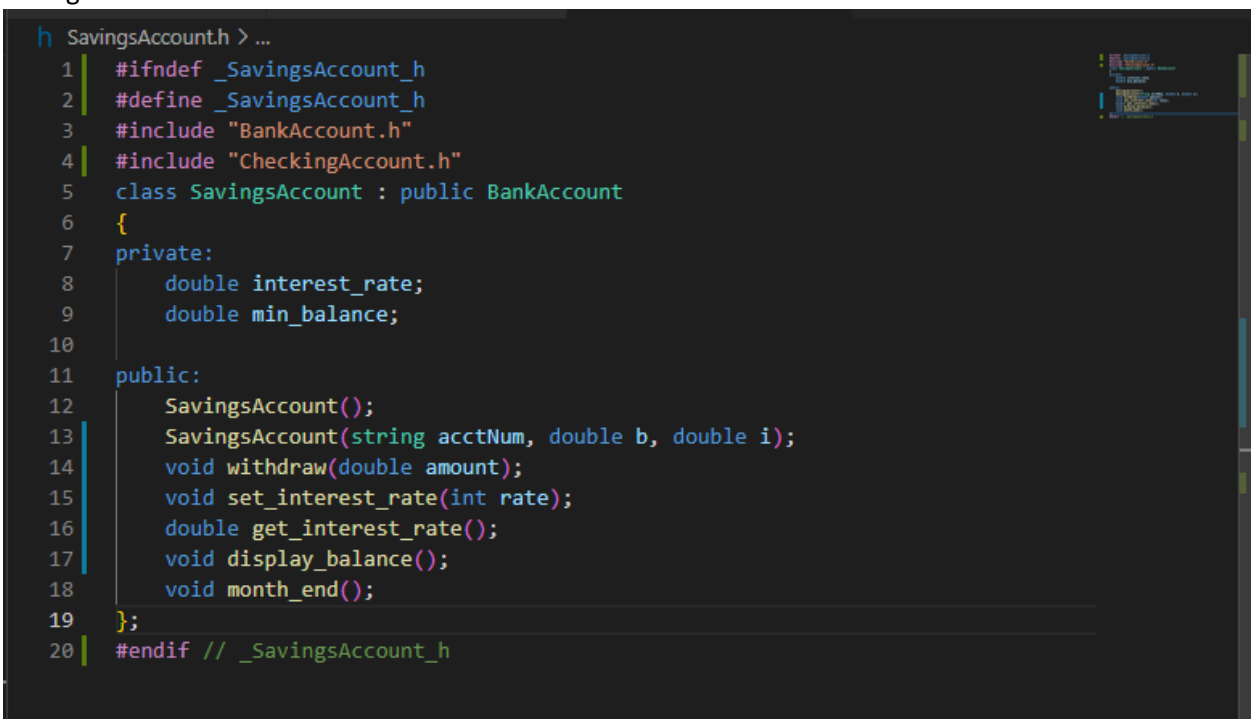
SavingsAccount.cpp

SavingsAccount.cpp > month_end()

```cpp
#include "SavingsAccount.h"
SavingsAccount::SavingsAccount()
{
    interest_rate = 0;
    min_balance = 0;
}
SavingsAccount::SavingsAccount(string acctNum, double b, double i)
{
    BankAccount(acctNum, b);
    interest_rate = i;
    min_balance = b;
}

void SavingsAccount::withdraw(double amounts)
{
    BankAccount::withdraw(amounts);
    if (BankAccount::get_balance() < min_balance)
    {
        min_balance = BankAccount::get_balance();
    }
}

void SavingsAccount::set_interest_rate(int rate)
{
    rate = interest_rate;
}
double SavingsAccount::get_interest_rate()
{
    return interest_rate;
}
void SavingsAccount::display_balance()
{
    cout << "Account:" << BankAccount::get_account_number() << "'s interest rate is " << interest_rate << endl;
    cout << BankAccount::get_account_number() << "'s Balance is $" << min_balance << endl;
}
void SavingsAccount::month_end()
{
    BankAccount::deposit(min_balance);
    double interest = min_balance * interest_rate / 100;
    BankAccount::deposit(interest);
    min_balance = BankAccount::get_balance();
}

```

Lab 7 PDF

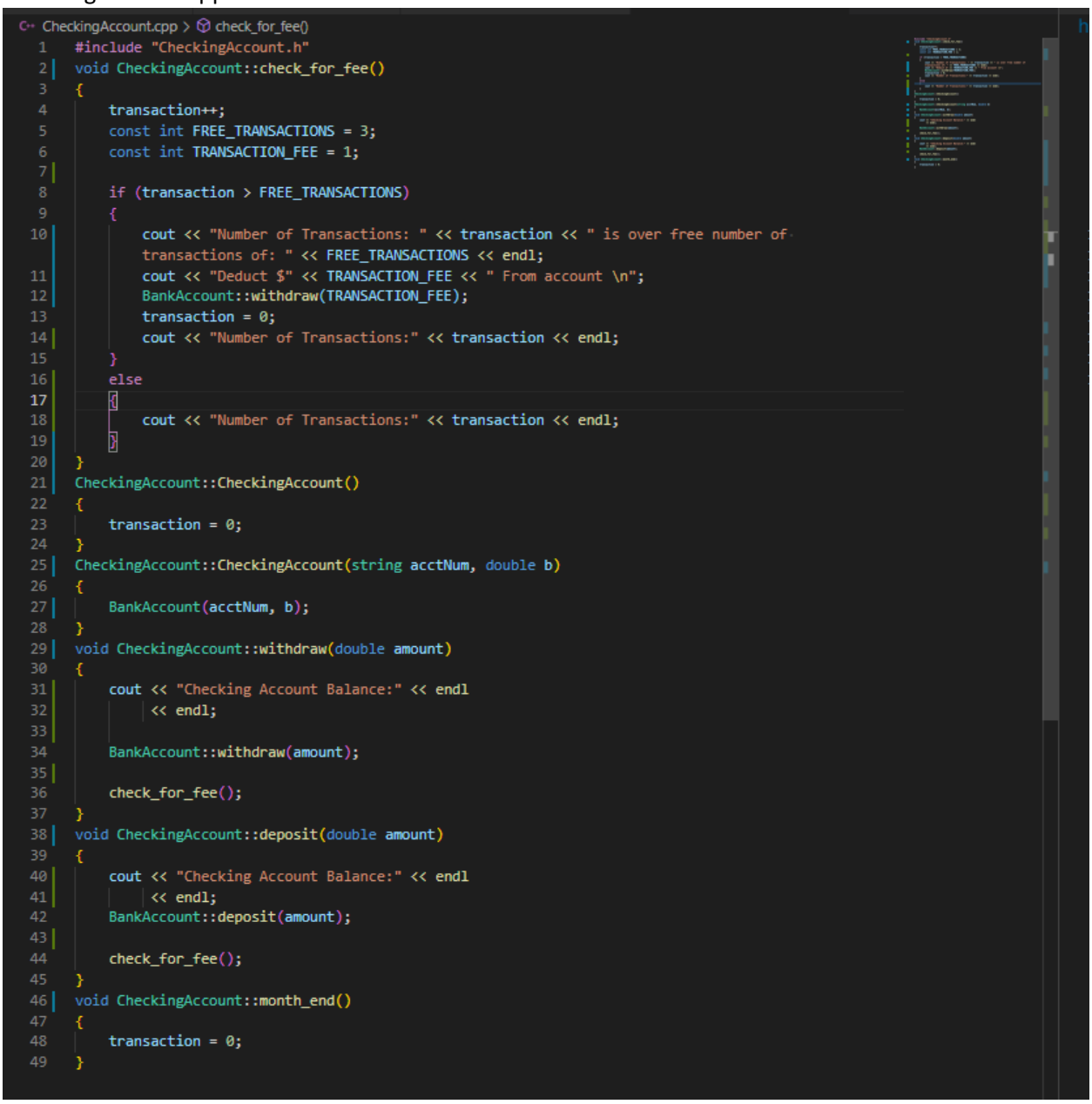
SavingsAccount.h



```h
#ifndef _SavingsAccount_h
#define _SavingsAccount_h
#include "BankAccount.h"
#include "CheckingAccount.h"
class SavingsAccount : public BankAccount
{
private:
    double interest_rate;
    double min_balance;

public:
    SavingsAccount();
    SavingsAccount(string acctNum, double b, double i);
    void withdraw(double amount);
    void set_interest_rate(int rate);
    double get_interest_rate();
    void display_balance();
    void month_end();
};
#endif // _SavingsAccount_h
```

Lab 7 PDF

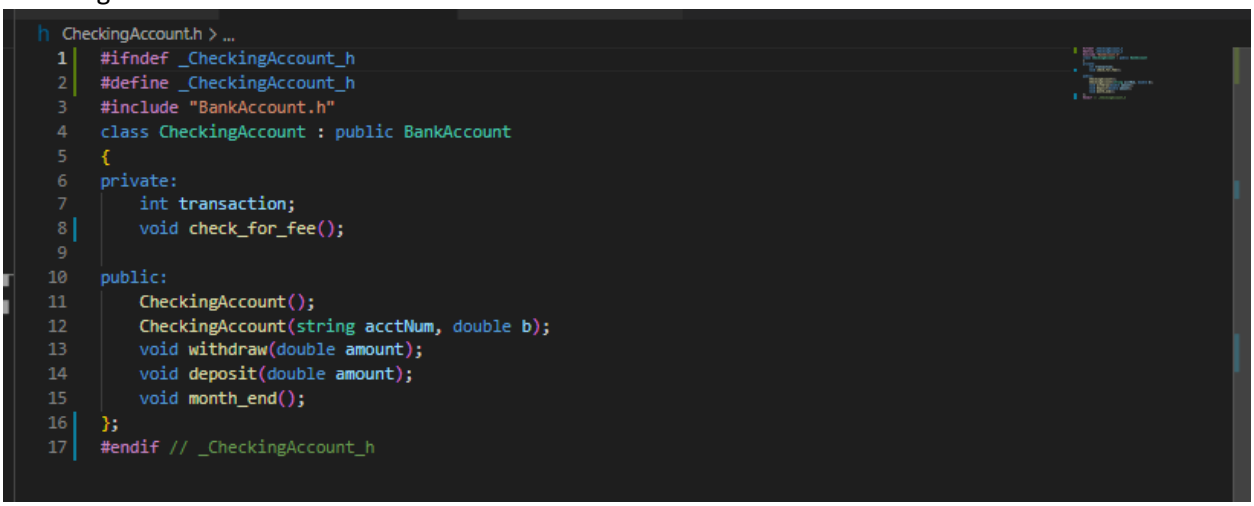
CheckingAccount.cpp



```cpp
#include "CheckingAccount.h"
void CheckingAccount::check_for_fee()
{
    transaction++;
    const int FREE_TRANSACTIONS = 3;
    const int TRANSACTION_FEE = 1;

    if (transaction > FREE_TRANSACTIONS)
    {
        cout << "Number of Transactions: " << transaction << " is over free number of
        transactions of: " << FREE_TRANSACTIONS << endl;
        cout << "Deduct $" << TRANSACTION_FEE << " From account \n";
        BankAccount::withdraw(TRANSACTION_FEE);
        transaction = 0;
        cout << "Number of Transactions:" << transaction << endl;
    }
    else
    {
        cout << "Number of Transactions:" << transaction << endl;
    }
}
CheckingAccount::CheckingAccount()
{
    transaction = 0;
}
CheckingAccount::CheckingAccount(string acctNum, double b)
{
    BankAccount(acctNum, b);
}
void CheckingAccount::withdraw(double amount)
{
    cout << "Checking Account Balance:" << endl
         << endl;

    BankAccount::withdraw(amount);

    check_for_fee();
}
void CheckingAccount::deposit(double amount)
{
    cout << "Checking Account Balance:" << endl
         << endl;
    BankAccount::deposit(amount);

    check_for_fee();
}
void CheckingAccount::month_end()
{
    transaction = 0;
}
```

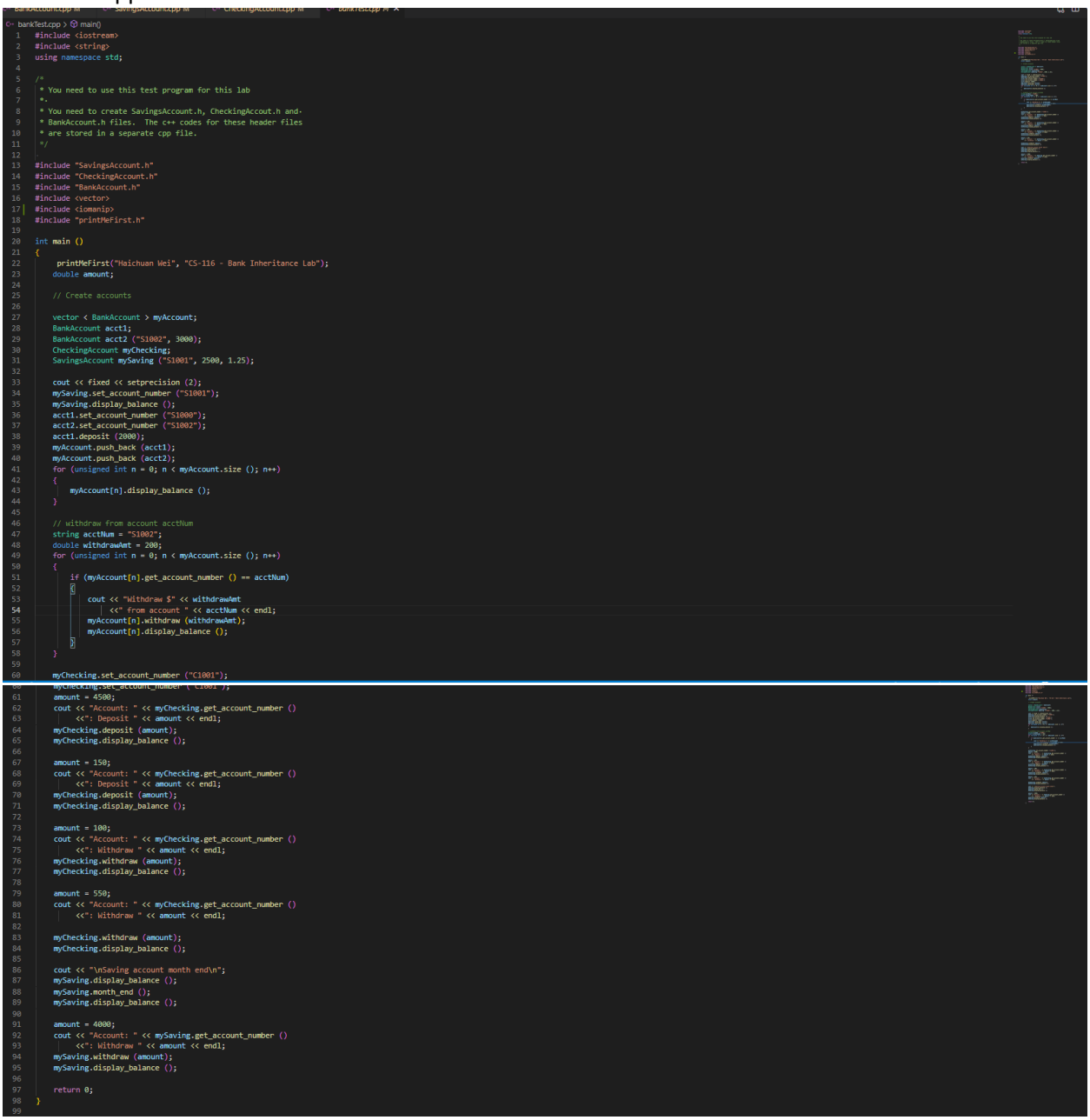
Lab 7 PDF

CheckingAccounts.h



Lab 7 PDF

bankTest.cpp



```cpp
#include <iostream>
#include <string>
using namespace std;

/*
 * You need to use this test program for this lab
 *.
 * You need to create SavingsAccount.h, CheckingAccout.h and
 * BankAccount.h files.  The c++ codes for these header files
 * are stored in a separate cpp file.
 */

#include "SavingsAccount.h"
#include "CheckingAccount.h"
#include "BankAccount.h"
#include <vector>
#include <iomanip>
#include "printMeFirst.h"

int main ()
{
     printMeFirst("Haichuan Wei", "CS-116 - Bank Inheritance Lab");
    double amount;

    // Create accounts

    vector < BankAccount > myAccount;
    BankAccount acct1;
    BankAccount acct2 ("S1002", 3000);
    CheckingAccount myChecking;
    SavingsAccount mySaving ("S1001", 2500, 1.25);

    cout << fixed << setprecision (2);
    mySaving.set_account_number ("S1001");
    mySaving.display_balance ();
    acct1.set_account_number ("S1000");
    acct2.set_account_number ("S1002");
    acct1.deposit (2000);
    myAccount.push_back (acct1);
    myAccount.push_back (acct2);
    for (unsigned int n = 0; n < myAccount.size (); n++)
    {
        myAccount[n].display_balance ();
    }

    // Withdraw from account acctNum
    string acctNum = "S1002";
    double withdrawAmt = 200;
    for (unsigned int n = 0; n < myAccount.size (); n++)
    {
        if (myAccount[n].get_account_number () == acctNum)
        {
            cout << "Withdraw $" << withdrawAmt
                 <<" from account " << acctNum << endl;
            myAccount[n].withdraw (withdrawAmt);
            myAccount[n].display_balance ();
        }
    }

    myChecking.set_account_number ("C1001");
    amount = 4500;
    cout << "Account: " << myChecking.get_account_number ()
         <<": Deposit " << amount << endl;
    myChecking.deposit (amount);
    myChecking.display_balance ();

    amount = 150;
    cout << "Account: " << myChecking.get_account_number ()
         <<": Deposit " << amount << endl;
    myChecking.deposit (amount);
    myChecking.display_balance ();

    amount = 100;
    cout << "Account: " << myChecking.get_account_number ()
         <<": Withdraw " << amount << endl;
    myChecking.withdraw (amount);
    myChecking.display_balance ();

    amount = 550;
    cout << "Account: " << myChecking.get_account_number ()
         <<": Withdraw " << amount << endl;

    myChecking.withdraw (amount);
    myChecking.display_balance ();

    cout << "\nSaving account month end\n";
    mySaving.display_balance ();
    mySaving.month_end ();
    mySaving.display_balance ();

    amount = 4000;
    cout << "Account: " << mySaving.get_account_number ()
         <<": Withdraw " << amount << endl;
    mySaving.withdraw (amount);
    mySaving.display_balance ();

    return 0;
}
```

Code Output:

All data displayed matched the example