Source Code

Account.hpp

```
#ifndef ACCOUNT_H
#define ACCOUNT_H

#include <iostream>
#include <string>

int const DAYS_PER_MONTH = 30;
double const SAVING_RATE = 0.06;

using namespace std;
//You need to implement all the member functions for the class Account
//.............................................................
class Account
{
public:
Account(); // doing nothing
Account(double b); // set balance to b
void deposit(double amount); // balance is increased by amount
void withdraw(double amount); // balance is reduced by amount
// make sure you can't withdraw if
// amount is larger than balance
double getBalance() const; // return the account balance
void setBalance(double b);
private:
double balance;
};
Account::Account()
{

}

Account::Account(double b)
{
   balance = b;
}
void Account::deposit(double amount)
{
   balance += amount;
}

void Account::withdraw(double amount)
{
   if(amount>balance)//only withdraws if balance is greater than the amount to be withdrawn
   {
       cout<<"insufficient funds. please enter a lower amount."<<endl;
   }
   else
   {
       balance-=amount;
   }
}
double Account::getBalance() const {//for accessing the balance attribute. is const because we
dont wanna change it
   return balance;
}

void Account::setBalance(double b){//for setting the value of balance in the checking/savings
constructor
   balance = b;
}
#endif
```

Checking.hpp

```
#ifndef CHECKING_H
#define CHECKING_H

#include "Account.hpp"
#include <iostream>
#include <string>

using namespace std;
/**
Determine the daily interest and deposit it into the account.
Checking accounts yield interest of 3 percent monthly on balances over $1,000.

Calling dailyInterest() will calculate daily interest and add the daily interest to
the balance
*/
// define class Checking which will inheritance class
// object Account - modify ????? below with correct code
class Checking: public Account // modify ?????? with correct code to inherent Account
{
public:
Checking(); // doing nothing
Checking(double b); // set balance to b
void dailyInterest();
};
//You need to implement all the member functions for the class Checking
void Checking::dailyInterest()
{
   double balance= getBalance();
   const double RATE = 0.03;
   const double MIN_BALANCE = 1000;

   if (balance>MIN_BALANCE)
   {   // balance - min balance because interest is calculated on the amount of money above
the minimum balance
       double amount =((balance-MIN_BALANCE)*RATE)/DAYS_PER_MONTH;
       deposit(amount);
   }
}
Checking::Checking(double b)
{
   setBalance(b);
}

#endif
```

Savings.hpp

```
#ifndef SAVINGS_H
#define SAVINGS_H

#include "Account.hpp"
#include <iostream>
#include <string>

using namespace std;

// define class Savings which will inherit class
// object Account - modify ????? below with correct code
class Savings : public Account // modify ??? with correct code to inherent Account
{
public:
Savings(); // doing nothing
```

```
Savings(double b); // initialize balance to b
void dailyInterest();
};
//You need to implement all the member functions for the class Savings
/** */

Savings::Savings(double b){
   setBalance(b);
}

/*
Determine the daily interest and deposit it into the account.
*/
void Savings::dailyInterest()
{
//
// calculate the daily interest rate which is
// balance * SAVING_RATE / DAYS_PER_MONTH
   double balance = getBalance();
   double amount =(balance*SAVING_RATE)/DAYS_PER_MONTH;

// Then call member function to deposit the interest to the
// balance
   deposit(amount);
}
//.............................................................


#endif



#endif
```

Lab5.cpp

```
#include <iostream>
#include <string>
#include "Account.hpp"
#include "Savings.hpp"
#include "Checking.hpp"
#include "printmefirst.hpp"

using namespace std;

/*Use the code below and modified the code to implement the requirement
of the lab*/

/*
Use the following test template to test your program
*/
int main()
{
printMeFirst("Sheharyar","CS-116");
Checking c = Checking(1000.0);
Savings s = Savings(1000.0);
for (int i = 1; i <= DAYS_PER_MONTH; i++)
{
c.deposit(i * 5);
c.withdraw(i * 2);
s.deposit(i * 5);
s.withdraw(i * 2);
c.dailyInterest();
s.dailyInterest();
if (i % 10 == 0) // use % to only print out days 10, 20, 30
{
cout << "day " << i << "\n";
```

```
cout << "Checking balance: " << c.getBalance() << "\n";
cout << "Savings balance: " << s.getBalance() << "\n";
}
}
return 0;
}
```

Purpose

The purpose of this lab is to practice and implement inheritance. We use inheritance to save time and not repeat code.

Logic

In this lab we create a parent class Account which has two daughter classes Savings and Checking. Both savings and checking need the functions withdraw, deposit, getBalance and setBalance so we put them in the the parent class Account. The dailyInterest functions are different for each account type so we keep them in each of the daughter classes Checking and Savings.

```
class Savings : public Account
class Checking: public Account
```

We are using “public” inheritance because we want access the balance variables and functions in the derived class. The table below shows the effects of different kinds of inheritance.

Base class member access specifier	Type of Inheritence		
	Public	Protected	Private
Public	Public	Protected	Private
Protected	Protected	Protected	Private
Private	Not accessible (Hidden)	Not accessible (Hidden)	Not accessible (Hidden)

Test Case:

```
 Program written by: Sheharyar
 Course info: CS-116
 Date: Thu Nov 01 16:10:35 2018

day 10
Checking balance: 1165.66
Savings balance: 1186.51
day 20
Checking balance: 1634.64
Savings balance: 1680.1
day 30
Checking balance: 2409.99
Savings balance: 2486.97
PS C:\Projects\CS116\Lab5-Inheritance>
```