Name: Amuldeep Dhillon
Class: CS116-02 Sha 2017 Spring
Assignment: Lab 4- Bank Accounts with Inheritance
Date: 4/7/2017

Description:

The program will create two bank accounts one Savings and one Checking The accounts checking and saving will be inherited from a class called account. Then it will deposit, withdraw, and transfer values from each account using Classes. If one tries to withdraw or transfer more than they possess they will be charged a 5 dollar penalty fee.

Inputs:

- Deposit 1000 in savings and 2000 in checkings
- Withdraw 1500 from checking
- Transfer 200 from savings to checking
- Attempt to withdraw 900 from savings, despite only having 800
- Withdraw 400 from checking

Outputs:

- The program will display my name, class, and time
- Show initial value of both accounts of 0
- Show deposit of 1000 in both accounts
- Show the 1000 in savings account and 1000 in checking account
- Show withdrawal of 1500 and transfer of 200
- Show the 800 in savings and the 700 in checking
- Show attempt of withdrawal of 900
- Show why that is not possible and why they will be fined
- Show the 795 in savings and the 700 in checkings
- Show the withdrawal of 400 from checkings account
- Show the 795 in savings and 300 in checkings

Source Code:

Lab.h:

```
/*
 *Contains all shared functions and classes along with other header
 * files
 * @author Amuldeep Dhillon
 * @version 1.0 4/6/2017
 *
*/
#ifndef LAB_H
#define LAB_H

#include <iostream>
#include <string>
#include <iomanip>
#include <ctime>
#include "account.h"
#include "checking.h"
#include "saving.h"
#include "bank.h"
void printMeFirst(std::string name, std::string courseInfo);
const double penalty = 5;
#endif
```

Account.h:

```
/*
 *Contains the class Account (the base class for Checking and Saving
 * @author Amuldeep Dhillon
 * @version 2.0 4/6/2017
 *
```

```
*/
#ifndef ACCOUNT_H
#define ACCOUNT_H
class Account
{
public:
  Account();
  Account(double bal);
  void setBalance();
  void setBalance(double bal);
  void deposit(double amount);
  void withdraw(double amount);
  double getBalance() const;
private:
  double balance;
};

#endif
```

Bank.h:

```
/*
 *Contains the class Bank
 * @author Amuldeep Dhillon
 * @version 1.0 4/6/2017
 *
*/
#ifndef BANK_H
#define BANK_H
class Bank
{
public:
  Bank();
  Bank(double checkingAmount, double savingsAmount);
  void printBalances() const;
```

```
   void deposit(double amount, std::string account);
   void withdraw(double amount, std::string account);
   void transfer(double amount, std::string account);

private:
   Checking checking;
   Saving savings;
};

#endif
```

Checking.h:

```
/*
 *Contains the class Checking which inherits Account
 * @author Amuldeep Dhillon
 * @version 1.0 4/6/2017
 *
*/
#ifndef CHECKING_H
#define CHECKING_H

class Checking:public Account
{
};


#endif
```

Saving.h:

```
/*
 *Contains the class Saving which inherits Account
 * @author Amuldeep Dhillon
 * @version 1.0 4/6/2017
 *
*/
#ifndef SAVING_H
#define SAVING_H
```

```
class Saving:public Account
{
};


#endif
```

PrintMeFirst.cpp:

```
/*
 *Purpose:
 * Print out programmer's information such as name, class information
 * and date/time the program is run
 *
 * @author Ron Sha
 * @version 1.0 1/27/2017
 *
 * @param name - Amuldeep Dhillon
 * @param courseInfo - Lab 4: CS 116-02 Thursdays
 * @return - none
*/


#include "lab.h"
void printMeFirst(std::string name, std::string courseInfo)

{

std::cout <<" Program written by: "<< name << std::endl; // put your name here

std::cout <<" Course info: "<< courseInfo << std::endl;

time_t now = time(0);  // current date/time based on current system

char* dt = ctime(&now); // convert now to string for

   std::cout << " Date: " << dt << std::endl;
```

```
}
```

Account.cpp:

```
#include "lab.h"
#include "account.h"
using namespace std;

/*
 *Purpose:
 * set account balance to zero
 *
 * @author Amuldeep Dhillon
 * @version 1.0 2/17/2017
 *
 * @param none
 * @Inputs - none
 * @Outputs - none
 * @return - none
*/
Account::Account(){
   balance = 0;
}

/*
 *Purpose:
 * set account balance to specified amount
 *
 * @author Amuldeep Dhillon
 * @version 1.0 2/17/2017
 *
 * @param the amount that balance is set to
 * @Inputs - none
 * @Outputs - none
 * @return - none
*/
Account::Account(double bal){
   balance = bal;
}
```

```
/*
 *Purpose:
 * set balance to zero
 *
 * @author Amuldeep Dhillon
 * @version 1.0 2/17/2017
 *
 * @param none
 * @Inputs - none
 * @Outputs - none
 * @return - none
*/
void Account::setBalance(){
   balance = 0;
}

/*
 *Purpose:set balance to specified amount
 *
 *
 * @author Amuldeep Dhillon
 * @version 1.0 2/17/2017
 *
 * @param the amount the balance should be
 * @Inputs - none
 * @Outputs - none
 * @return - none
*/
void Account::setBalance(double bal){
   balance = bal;
}

/*
 *Purpose:
 * add a specified value to balance
 *
 * @author Amuldeep Dhillon
 * @version 1.0 2/17/2017
```

```
 *
 * @param the amount added to balance
 * @Inputs - none
 * @Outputs - none
 * @return - none
*/
void Account::deposit(double amount){
   balance = balance + amount;
}

/*
 *Purpose:
 * remove specified amount from balance
 *
 * @author Amuldeep Dhillon
 * @version 1.0 2/17/2017
 *
 * @param the amount to be removed from balance
 * @Inputs - none
 * @Outputs - none
 * @return - none
*/
void Account::withdraw(double amount){
   balance = balance - amount;
}

/*
 *Purpose:
 * return the value of balance
 *
 * @author Amuldeep Dhillon
 * @version 1.0 2/17/2017
 *
 * @param none
 * @Inputs - none
 * @Outputs - none
 * @return - the double balance
*/
double Account::getBalance() const{
```

```
    return balance;
}
```

Bank.cpp:

```
#include "lab.h"
#include "bank.h"
using namespace std;

/*
 *Purpose:
 * set Bank Account balances to zero
 *
 * @author Amuldeep Dhillon
 * @version 2.0 4/6/2017
 *
 * @param - none
 * @Inputs - none
 * @Outputs - none
 * @return - none
*/
Bank::Bank(){
    checking.setBalance();
    savings.setBalance();
}


/*
 *Purpose:
 * set Bank Account balances to a specified amounts
 *
 * @author Amuldeep Dhillon
 * @version 2.0 4/6/2017
 *
 * @param - checkingAmount, savingAmount
 * @Inputs - none
 * @Outputs - none
 * @return - none
*/
```

```
Bank::Bank (double checkingAmount,double savingAmount){
   checking.setBalance(checkingAmount);
   savings.setBalance(savingAmount);
}


/*
 *Purpose:
 * shows the bank account balances
 *
 * @author Amuldeep Dhillon
 * @version 2.0 4/6/2017
 *
 * @param - none
 * @Inputs - none
 * @Outputs - The amount in both accounts
 * @return - none
*/
void Bank::printBalances() const{
   cout << "Savings Account Balance: $" << fixed << setprecision(2) << savings.getBalance()
<< endl;
   cout << "Checking Account Balance: $" << fixed << setprecision(2) << checking.getBalance()
<< endl;
}

/*
 *Purpose:
 * Adds money to either checking or saving accounts
 *
 * @author Amuldeep Dhillon
 * @version 2.0 4/6/2017
 *
 * @param - the amount to deposit, which account
 * @Inputs - none
 * @Outputs - none
 * @return - none
*/
void Bank::deposit(double amount,string account){
   if(account == "C"){
```

```
        checking.deposit(amount);}
    else{
        savings.deposit(amount);
    }
}

/*
 *Purpose:
 * removes the specified amount from a specified account
 * checks to see if user actaully has the money to withdraw
 * charges 5 dollars if they don't
 *
 * @author Amuldeep Dhillon
 * @version 2.0 4/6/2017
 *
 * @param - amount to withdraw and from which account
 * @Inputs - none
 * @Outputs - Message telling user they don't own that amount
 * and that they have been charged a penalty.
 * @return - none
*/
void Bank::withdraw(double amount, string account){
    if(account == "C"){
        if(amount > checking.getBalance()){
            cout << "Only $" << checking.getBalance() << " are available. "
            << "But tring to withdraw $" << amount << ". Deduct $5 from account";
            checking.withdraw(penalty);}
        else{
            checking.withdraw(amount);}
        }
    else{
        if(amount > savings.getBalance()){
            cout << "Only $" << checking.getBalance() << " are available, "
            << "but tring to withdraw $" << amount << ". Deduct $5 from account\n";
            savings.withdraw(penalty);}
        else{
            savings.withdraw(amount);}
    }
}
```

```
/*
 *Purpose:
 * sends money from one account to another
 *
 * @author Amuldeep Dhillon
 * @version 2.0 4/6/2017
 *
 * @param - amount to transfer and from which account to remove
 * @Inputs - none
 * @Outputs - message telling user they do not own that amount
 * and that they have been charged a penalty
 * @return - none
*/
void Bank::transfer(double amount, string account){
   if(account == "C"){
         if(checking.getBalance() >= amount){
               checking.withdraw(amount);
               savings.deposit(amount);
         }
         else{
               checking.withdraw(penalty);}
         }
         if(savings.getBalance() >= amount){
               savings.withdraw(amount);
               checking.deposit(amount);
         }
         else{
               savings.withdraw(penalty);}
}
```

Main.cpp:

```
/*
 *Purpose:
 * Execute the entire program
 *
 * @author Amuldeep Dhillon
 * @version 1.0 2/17/2017
```

```
 *
 * @param none
 * @Inputs - Classes Bank and Account
 * @Outputs- the outputs of each respective program
 * @return - 0
*/
#include "lab.h"
using namespace std;

int main(){

  printMeFirst("Amuldeep Dhillon", "Lab 4 CS-116-02 - 2017 Spring");  // you must call this function 1st

  Bank myBank;
  cout << "\nInital bank balances: \n";
  myBank.printBalances();   /* set up empty accounts */

  cout << "\nAdding some money to accounts: \n";

  cout << "\nAdding $1000 to saving \n";
  cout << "Adding $2000 to checking \n";
  myBank.deposit(1000, "S");  /* deposit $1000 to savings */
  myBank.deposit(2000, "C");  /* deposit $2000 to checking */
  myBank.printBalances();

  cout << "\nTaking out $1500 from checking,and moving $200 from";
  cout << " savings to checking.\n";

  myBank.withdraw(1500, "C"); /* withdraw $1500 from checking */
  myBank.transfer(200, "S");  /* transfer $200 from savings */
  myBank.printBalances();


  cout << "\nTrying to withdraw $900 from Savings.\n";
  myBank.withdraw(900,"S");
  myBank.printBalances();

  cout << "\nTrying to withdraw $400 from Checking.\n";
```

```
  myBank.withdraw(400,"C");
  myBank.printBalances();


  return 0;
}
```

Makefile:

```
# begin of Makefile
#
CC=g++
#
CFLAGS = -c -Wall -I/usr/include/mysql
#LFLAGS = -L/usr/lib/mysql -lmysqlclient
LFLAGS =

all: main

main: main.o bank.o account.o printMeFirst.o
   $(CC) main.o bank.o account.o  printMeFirst.o -o lab $(LFLAGS)

bank.o: bank.cpp bank.h
   $(CC) $(CFLAGS) bank.cpp

account.o: account.cpp account.h
   $(CC) $(CFLAGS) account.cpp

printMeFirst.o: printMeFirst.cpp
   $(CC) $(CFLAGS) printMeFirst.cpp

clean:
   rm *.o lab

run:
   ./lab

#end of Makefile
```

Screen Shots:

```
cs:lab4$ ./lab
 Program written by: Amuldeep Dhillon
 Course info: Lab 4 CS-116-02 - 2017 Spring
 Date: Thu Apr  6 14:32:48 2017


Inital bank balances:
Savings Account Balance: $0.00
Checking Account Balance: $0.00

Adding some money to accounts:

Adding $1000 to saving
Adding $2000 to checking
Savings Account Balance: $1000.00
Checking Account Balance: $2000.00

Taking out $1500 from checking,and moving $200 from savings to checking.
Savings Account Balance: $800.00
Checking Account Balance: $700.00

Trying to withdraw $900 from Savings.
Only $700.00 are available, but tring to withdraw $900.00. Deduct $5 from account
Savings Account Balance: $795.00
Checking Account Balance: $700.00

Trying to withdraw $400 from Checking.
Savings Account Balance: $795.00
Checking Account Balance: $300.00
```