Name: Amuldeep Dhillon
Class: CS116-02 Sha 2017 Spring
Assignment: Lab 2- Bank Accounts
Date: 2/19/2017

Description:
The program will create two bank accounts one Savings and one Checking. 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:
```
#ifndef LAB_H
#define LAB_H

#include <iostream>
#include <string>
#include <iomanip>
```

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

account.h

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

#endif
```

bank.h

```
#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:
  Account checking;
```

```
   Account savings;
};

#endif
```

printMeFirst:

```
/*
 *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 1: 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

```
/*
```

```
 *Purpose:
 * Set Account balances to zero or specified amount
 * Subtract withdrawals
 * Add deposits
 *
 * @author Amuldeep Dhillon
 * @version 1.0 2/17/2017
 *
 * @param none
 * @Inputs - balances, withdrawls, deposits
 * @Outputs - balances
 * @return - Balances
*/
#include "lab.h"
#include "account.h"
using namespace std;

Account::Account(){
   balance = 0;
}

Account::Account(double bal){
   balance = bal;
}

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

void Account::withdraw(double amount){
   balance = balance - amount;
}

double Account::getBalance() const{
   return balance;
}
```

bank.cpp

```
/*
```

```
 *Purpose:
 * set Bank Account balances to zero or specified amounts
 * Print the balances
 * Add money into accounts
 * Remove money from the accounts
 * Transfer money from one account to another
 *
 * @author Amuldeep Dhillon
 * @version 1.0 2/17/2017
 *
 * @param - checkingAmount, savingAmount, amount, account
 * @Inputs - Money amount, account types
 * @Outputs - Balances
 * @return - Balances
*/
#include "lab.h"
#include "bank.h"
using namespace std;

Bank::Bank(){
   checking = Account();
   savings = Account();
}

Bank::Bank (double checkingAmount,double savingAmount){
   checking = Account(checkingAmount);
   savings = Account(savingAmount);
}

void Bank::printBalances() const{
   cout << "Savings Account Balance: $" << fixed << setprecision(2) << savings.getBalance()
<< endl;
   cout << "Checking Account Balance: $" << fixed << setprecision(2) << checking.getBalance()
<< endl;
}

void Bank::deposit(double amount,string account){
   if(account == "C"){
         checking.deposit(amount);}
```

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

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);}
    }
}

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
 * @return - 0
*/
#include "lab.h"
using namespace std;

int main(){

  printMeFirst("Amuldeep Dhillon", "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
```

Screenshot: