Amir Sotoodeh
CS256
5/15/18

Project #2: Bank Account

Github: https://github.com/excisionhd/CS256/blob/master/proj2.cpp

```
/****************************************************************
* file:      proj2.cpp
* author: Amir Sotoodeh
* class: CS 256 - C++
*
* assignment: Project 2
* date last modified: 5/15/18
*
* purpose: A project dealing with inheritance, virtual methods,
* and overriding methods.
*****************************************************************/

#include "stdafx.h"
#include <iostream>
#include <string>
#include <locale>

using namespace std;


class BankAccount {
protected:
      float balance;
      int numDeposits;
      int numWithdrawals;
      float annualIntRate;
      float serviceCharges;

public:
      BankAccount(float b, float i) { //constructor sets initial values of
fields when instantiated
            balance = b;
            annualIntRate = i;
            numDeposits = 0;
            numWithdrawals = 0;
            serviceCharges = 0;
      }

      virtual void depositMoney(float amount) { //deposits money into the
account
            balance = balance + amount;
            numDeposits++;
      }

      virtual void withdrawMoney(float amount) { //takes money out of the
account with an error if amount exceeds balance
            if (balance > amount) {
                  balance -= amount;
                  numWithdrawals++;
            }
            else {
                  cout << "Error: Not enough funds." << endl;
            }
      }

      virtual void calcInterest() { //calculates the interest based on the
interest rate
            float monthlyInterestRate = annualIntRate / 12;
            float monthlyInterest = balance * monthlyInterestRate;
            balance = balance + monthlyInterest;
      }

      virtual void monthlyProcess() { //processes the account after
serviceCharges and interest rates and sets fields back to 0.
            printStatistics();
            balance = balance - serviceCharges;
            calcInterest();
            numWithdrawals = 0;
            numDeposits = 0;
            serviceCharges = 0;
      }

      void printStatistics() {
            cout << "Balance: " << balance << endl;
            cout << "Number of Deposits: " << numDeposits << endl;
            cout << "Number of Withdrawals: " << numWithdrawals << endl;
            cout << "Annual Interest Rate: " << annualIntRate << endl;
            cout << "Service Charges: " << serviceCharges << endl;
      }

      float getBalance() {
            return balance;
      } //returns balance

      void setBalance(float b) {
            balance = b;
      }

      int getNumDeposits() {
            return numDeposits;
      }

      void setNumDeposits(int n) {
            numDeposits = n;
      }

      int getNumWithdrawals() {
            return numWithdrawals;
      } //returns number of withdrawals made

      void setNumWithdrawals(int n) {
            numWithdrawals = n;
      }

      float getAnnualIntRate() {
            return annualIntRate;
      }

      void setAnnualIntRate(float i) {
            annualIntRate = i;
      }

      float getServiceCharges() {
            return serviceCharges;
      } //returns the service charges

      void setServiceCharges(float i) {
            serviceCharges = i;
      } //sets the service charges
};

class SavingsAccount : public BankAccount {
private:
      bool status;

public:
      SavingsAccount(float b, float i) : BankAccount(b, i) {  }

      virtual void withdrawMoney(float amount) override { //calls superclass
method to withdraw
            if (balance < 25)
                  cout << "Error: Account is inactive" << endl;
            else {
                  BankAccount::withdrawMoney(amount);
            }
      }

      virtual void depositMoney(float amount) override { //calls superclass
```

```
method to deposit
            if (balance<25 && status == false) {
                  BankAccount::depositMoney(amount);
                  status = true;
            }
            else
                  BankAccount::depositMoney(amount);
      }

      string printActive() { //prints if the acount is active or not.
            if (balance<25)
                  return "Savings account is now inactive.";
            else
                  return "Savings account is now active.";
      }


      virtual void monthlyProcess() override { //unique method that charges an
additional dollar for every excess withdrawal after 4.
            int excessWithdraws = 0;
            if (getNumWithdrawals()>4) {
                  excessWithdraws = getNumWithdrawals() - 4;
                  setServiceCharges(getServiceCharges() + excessWithdraws);
                  BankAccount::monthlyProcess();
                  if (balance<25) {
                        status = false;
                        cout << "Account balance is " << balance << endl;
                        cout << "Savings account is now inactive.\n" << endl;
                  }
                  else {
                        status = true;
                        cout << "Account balance is " << balance << endl;
                        cout << "Savings account is now active.\n" << endl;
                  }
            }
            else {
                  BankAccount::monthlyProcess();
                  if (balance<25) {
                        status = false;
                        cout << "Account balance is " << balance << endl;
                        cout << "Savings account is now inactive.\n" << endl;
                  }
                  else {
                        status = true;
                        cout << "Account balance is " << balance << endl;
                        cout << "Savings account is now active.\n" << endl;
                  }
            }
```

```
	}
};

class CheckingAccount : public BankAccount {
public:
	CheckingAccount(float b, float i): BankAccount(b,i) {  }

	virtual void withdrawMoney(float b) override {
		if (balance - b < 0) {
			balance -= serviceCharges;
		}
		else {
			BankAccount::withdrawMoney(b);
		}
	}

	virtual void monthlyProcess() override  {
		serviceCharges = 5 + (numWithdrawals*0.1f);
		BankAccount::monthlyProcess();
		cout << "Account balance is now " << balance <<"."<< endl;
	}

};

int main(){
		bool hold = true;
		string accountChoice;
		string actionChoice;
		float amount = 0;
		SavingsAccount a1(0, 0); //creates instance of SavingsAccount
		CheckingAccount a2(0, 0); //creates instance of CheckingAccount



		do {
			cout << "Which account would you like to access, checking or
savings?: " << endl;
			cin >> accountChoice;

			if (accountChoice == "checking" || accountChoice ==
"Checking") {
				//a2.printStatistics();
				cout << "Which action do you wish to perform" << endl;
				cout<<"Withdraw, deposit, monthly processing?: " <<
endl;
				cin.ignore();
				getline(cin, actionChoice);
```

```
                        if (actionChoice == "deposit" || actionChoice ==
"Deposit") {
                              cout<<"Enter amount to deposit: " << endl;
                              cin >> amount;
                              a2.depositMoney(amount);
                              cout<<"Account balance is " << a2.getBalance()
<< "." << endl;
                        }
                        else if (actionChoice == "withdraw") {
                              cout << "Enter amount to withdraw: " << endl;
                              cin >> amount;
                              a2.withdrawMoney(amount);
                              cout<<"Account balance is " << a2.getBalance()
<< "." << endl;
                        }
                        else if (actionChoice=="monthly processing") {
                              a2.monthlyProcess();
                        }
                        else if (actionChoice == "q" || actionChoice =="Q") {
                              hold = false;
                        }

                  }
                  else if (accountChoice == "savings" || accountChoice
=="Savings") {
                        cout << "Which action do you wish to perform" << endl;
                        cout << "Withdraw, deposit, monthly processing?: "<<
endl;
                        cin.ignore();
                        getline(cin, actionChoice);
                        if (actionChoice == "deposit" || actionChoice ==
"Deposit") {
                              cout << "Enter amount to deposit: " << endl;
                              cin >> amount;
                              a1.depositMoney(amount);
                              cout << "Account balance is " << a1.getBalance()
<< "." << endl;
                              cout << a1.printActive() << endl;
                        }
                        else if (actionChoice == "withdraw" || actionChoice ==
"Withdraw") {
                              cout << "Enter amount to withdraw: " << endl;
                              cin >> amount;
                              a1.withdrawMoney(amount);
                              cout << "Account balance is " << a1.getBalance()
<< "." << endl;
                              cout << a1.printActive() << endl;
                        }
```

```
                              else if (actionChoice == "monthly processing") {
                                   a1.monthlyProcess();
                              }
                              else if (actionChoice == "q" || actionChoice == "Q") {
                                   hold = false;
                              }
                         }
                         else if (accountChoice == "q" || accountChoice == "Q") {
                              hold = false;
                         }

                    } while (hold == true);
          }
```

Output

```
Which account would you like to access, checking or savings?:
checking
Which action do you wish to perform
Withdraw, deposit, monthly processing?:
deposit
Enter amount to deposit:
1000
Account balance is 1000.
Which account would you like to access, checking or savings?:
savings
Which action do you wish to perform
Withdraw, deposit, monthly processing?:
deposit
Enter amount to deposit:
1000
Account balance is 1000.
Savings account is now active.
Which account would you like to access, checking or savings?:
checking
Which action do you wish to perform
Withdraw, deposit, monthly processing?:
withdraw
Enter amount to withdraw:
10
Account balance is 990.
Which account would you like to access, checking or savings?:
savings
Which action do you wish to perform
Withdraw, deposit, monthly processing?:
withdraw
Enter amount to withdraw:
```

```
10
Account balance is 990.
Savings account is now active.
Which account would you like to access, checking or savings?:
savings
Which action do you wish to perform
Withdraw, deposit, monthly processing?:
withdraw
Enter amount to withdraw:
10
Account balance is 980.
Savings account is now active.
Which account would you like to access, checking or savings?:
savings
Which action do you wish to perform
Withdraw, deposit, monthly processing?:
withdraw
Enter amount to withdraw:
10
Account balance is 970.
Savings account is now active.
Which account would you like to access, checking or savings?:
savings
Which action do you wish to perform
Withdraw, deposit, monthly processing?:
withdraw
Enter amount to withdraw:
10
Account balance is 960.
Savings account is now active.
Which account would you like to access, checking or savings?:
savings
Which action do you wish to perform
Withdraw, deposit, monthly processing?:
withdraw
Enter amount to withdraw:
940
Account balance is 20.
Savings account is now inactive.
Which account would you like to access, checking or savings?:
checking
Which action do you wish to perform
Withdraw, deposit, monthly processing?:
monthly processing
Balance: 990
```

```
Number of Deposits: 1
Number of Withdrawals: 1
Annual Interest Rate: 0
Service Charges: 5.1
Account balance is now 984.9.
Which account would you like to access, checking or savings?:
savings
Which action do you wish to perform
Withdraw, deposit, monthly processing?:
monthly processing
Balance: 20
Number of Deposits: 1
Number of Withdrawals: 5
Annual Interest Rate: 0
Service Charges: 1
Account balance is 19
Savings account is now inactive.

Which account would you like to access, checking or savings?:
q
Press any key to continue . . .
```