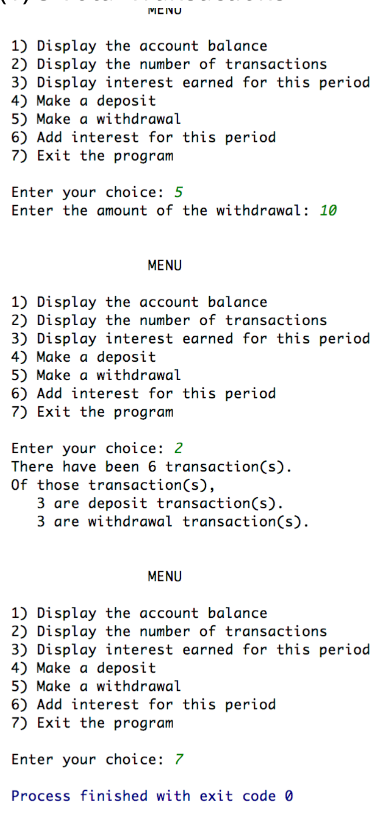
Ranoa, Julius
CSC 121 001 Computer Science I
28 September 2017 Thursday

Quiz 5
Modification of Home Software Company OOP Case Study (Program 7-17)
The source code below has been modified to report withdrawal and deposit transactions separately. Modified parts are highlighted in yellow.

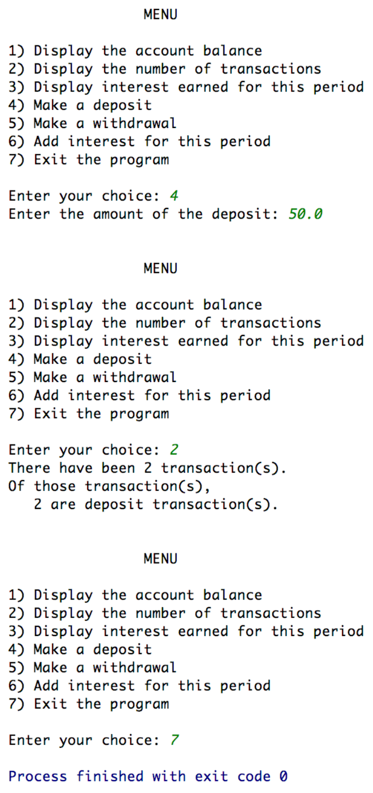
Screenshots of Runtime:
Note that the screenshots are truncated to only display the last parts of the output.

(1) 5 Total Transactions



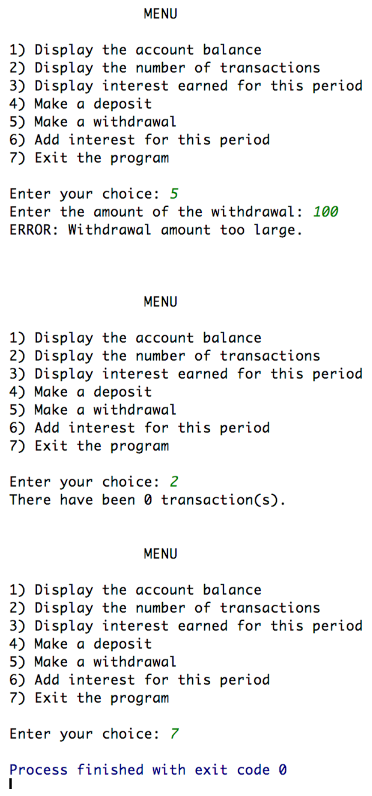
```
                MENU

1) Display the account balance
2) Display the number of transactions
3) Display interest earned for this period
4) Make a deposit
5) Make a withdrawal
6) Add interest for this period
7) Exit the program

Enter your choice: 5
Enter the amount of the withdrawal: 10


                MENU

1) Display the account balance
2) Display the number of transactions
3) Display interest earned for this period
4) Make a deposit
5) Make a withdrawal
6) Add interest for this period
7) Exit the program

Enter your choice: 2
There have been 6 transaction(s).
Of those transaction(s),
   3 are deposit transaction(s).
   3 are withdrawal transaction(s).


                MENU

1) Display the account balance
2) Display the number of transactions
3) Display interest earned for this period
4) Make a deposit
5) Make a withdrawal
6) Add interest for this period
7) Exit the program

Enter your choice: 7

Process finished with exit code 0
```

(2) No withdrawals



```
                MENU

1) Display the account balance
2) Display the number of transactions
3) Display interest earned for this period
4) Make a deposit
5) Make a withdrawal
6) Add interest for this period
7) Exit the program

Enter your choice: 4
Enter the amount of the deposit: 50.0


                MENU

1) Display the account balance
2) Display the number of transactions
3) Display interest earned for this period
4) Make a deposit
5) Make a withdrawal
6) Add interest for this period
7) Exit the program

Enter your choice: 2
There have been 2 transaction(s).
Of those transaction(s),
   2 are deposit transaction(s).


                MENU

1) Display the account balance
2) Display the number of transactions
3) Display interest earned for this period
4) Make a deposit
5) Make a withdrawal
6) Add interest for this period
7) Exit the program

Enter your choice: 7

Process finished with exit code 0
```

(3) One failed withdrawal



```
                MENU

1) Display the account balance
2) Display the number of transactions
3) Display interest earned for this period
4) Make a deposit
5) Make a withdrawal
6) Add interest for this period
7) Exit the program

Enter your choice: 5
Enter the amount of the withdrawal: 100
ERROR: Withdrawal amount too large.


                MENU

1) Display the account balance
2) Display the number of transactions
3) Display interest earned for this period
4) Make a deposit
5) Make a withdrawal
6) Add interest for this period
7) Exit the program

Enter your choice: 2
There have been 0 transaction(s).


                MENU

1) Display the account balance
2) Display the number of transactions
3) Display interest earned for this period
4) Make a deposit
5) Make a withdrawal
6) Add interest for this period
7) Exit the program

Enter your choice: 7

Process finished with exit code 0
```

Source Code:
The following files are included in the project.

- Account.h
- Account.cpp
- main.cpp

Project files are also posted in GitHub.
https://github.com/TheLoneWoof1102/FA17_CSC121001/tree/master/Source%20Code/Sandbox-Wk6.Ch7-8

Account.h

```
// Account.h is the Account class specification file.
class Account
{
private:
    double balance;
    double intRate;
    double interest;
    int depositTransactions;
    int withdrawalTransactions;

public:

    // Constructor
    Account(double rate = 0.045, double bal = 0.0) {
        balance = bal;  intRate = rate;
        interest = 0.0; depositTransactions = 0;
        withdrawalTransactions = 0;
    }

    void makeDeposit(double amount) {
        balance += amount;
        depositTransactions++;
    }

    bool withdraw(double amount);  // Defined in account.cpp

    void calcInterest() {
        interest = balance * intRate;
        balance += interest;
    }

    double getBalance() { return balance; }
    double getInterest() { return interest; }

    int getDepositTransactions() { return depositTransactions; }
    int getWithdrawalTransactions() { return withdrawalTransactions; }
    int getTransactions() {
        return depositTransactions + withdrawalTransactions;
    }
};
```

End-of-File: Account.h

Account.cpp

```
// Account.cpp is the Account class function implementation file.
#include "Account.h"

bool Account::withdraw(double amount) {
    if (balance < amount)
        return false;   // Not enough in the account
    else {
        balance -= amount;
        withdrawalTransactions++;
        return true;
    }
}
```

End-of-File: Account.cpp

main.cpp

1 out of 3 pages.

```
// This client program uses the Account class to perform simple
// banking operations. This file should be combined into a
// project along with the Account.h and Account.cpp files.
#include <iostream>
#include <iomanip>
#include "Account.h"
using namespace std;

// Function prototypes
void displayMenu();
char getChoice(char);
void makeDeposit(Account &);
void withdraw(Account &);
void printTransactionInfo(Account);

int main() {
    const char MAX_CHOICE = '7';
    Account savings;             // Account object to model savings account
    char choice;

    cout << fixed << showpoint << setprecision(2);
    do {
        displayMenu();
        choice = getChoice(MAX_CHOICE);  // This returns only '1' - '7'
        switch(choice) {
            case '1':
                cout << "The current balance is $";
                cout << savings.getBalance() << endl;
                break;
            case '2':
                printTransactionInfo(savings);
                break;
            case '3':
                cout << "Interest earned for this period: $";
                cout << savings.getInterest() << endl;
                break;
            case '4': makeDeposit(savings);
                break;
            case '5': withdraw(savings);
                break;
            case '6':
                savings.calcInterest();
                cout << "Interest added.\n";
        }
    } while(choice != '7');
    return 0;
}
```

main.cpp is continued next page.

cont'd – main.cpp

```
/****************************************************************
 *                        displayMenu                           *
 * This function displays the user's menu on the screen.        *
 ****************************************************************/
void displayMenu() {
    cout << "\n\n                MENU\n\n";
    cout << "1) Display the account balance\n";
    cout << "2) Display the number of transactions\n";
    cout << "3) Display interest earned for this period\n";
    cout << "4) Make a deposit\n";
    cout << "5) Make a withdrawal\n";
    cout << "6) Add interest for this period\n";
    cout << "7) Exit the program\n\n";
    cout << "Enter your choice: ";
}

/*****************************************************************
 *                            getChoice                          *
 * This function gets, validates, and returns the user's choice. *
 *****************************************************************/
char getChoice(char max) {
    char choice = cin.get();
    cin.ignore();           // Bypass the '\n' in the input buffer

    while (choice < '1' || choice > max)
    {
        cout << "Choice must be between 1 and " << max << ". "
             << "Please re-enter choice: ";
        choice = cin.get();
        cin.ignore();       // Bypass the '\n' in the input buffer
    }
    return choice;
}

/****************************************************************
 *                        makeDeposit                           *
 * This function accepts a reference to an Account object.      *
 * The user is prompted for the dollar amount of the deposit,   *
 * and the makeDeposit member of the Account object is          *
 * then called.                                                 *
 ****************************************************************/
void makeDeposit(Account &account) {
    double dollars;

    cout << "Enter the amount of the deposit: ";
    cin  >> dollars;
    cin.ignore();
    account.makeDeposit(dollars);
}
```

main.cpp is continued next page.

cont'd – main.cpp

```
/*****************************************************************
 *                        withdraw                               *
 * This function accepts a reference to an Account object.       *
 * The user is prompted for the dollar amount of the withdrawal,*
 * and the withdraw member of the Account object is then called.*
 *****************************************************************/
void withdraw(Account &account) {
    double dollars;

    cout << "Enter the amount of the withdrawal: ";
    cin  >> dollars;
    cin.ignore();
    if (!account.withdraw(dollars))
        cout << "ERROR: Withdrawal amount too large.\n\n";
}

void printTransactionInfo(Account acct) {
    int total = acct.getTransactions(),
        dTotal = acct.getDepositTransactions(),
        wTotal = acct.getWithdrawalTransactions();

    cout << "There have been " << total << " transaction(s).\n";
    if (total > 0) {
        cout << "Of those transaction(s),\n";
        if (dTotal > 0) {
            cout << "    " << dTotal
                 << " are deposit transaction(s).\n";
        }
        if (wTotal > 0) {
            cout << "    " << wTotal
                 << " are withdrawal transaction(s).\n";
        }
    }
    cout << "";
}
```

End-of-File: Account.cpp