COE318 – Lab 4: Bank Accounts

Objectives

- Implement an Account class.
- Implement a Bank class.
- Use loops.
- Use arrays.
- Write one of the classes from scratch (no template given as in previous labs)

Duration: one week.

Grading Scheme:

50% submitted source code

25% in-class demonstration and questions (during week 6 lab hours)

25% in-class quiz – Held during the first 5 mins of the lab class (during week 6 lab hours)

Overview

A classic application of object-oriented techniques is a bank account. In this lab, you will implement a simplified model of a bank account and of a bank itself.

A bank `Account` will track the account number, the owner's name and the current balance. The only operations allowed (apart from trivial getters) will be withdrawing and depositing money. A user will not be allowed to make a deposit or withdrawal of a negative or zero value. Furthermore, they will not be allowed to withdraw more money than they have in the account.

An account, of course, must belong a specific bank; hence you will also implement the `Bank` class. A `Bank` object will keep track of all of the accounts it has; it will also disallow two accounts with the same account number.

The Account Class

No template is given for the Account class, you will have to create it from scratch following the design instructions given here.

The `Account` class will consist of instance variables, a constructor and the following methods:

- `String getName()`
- `double getBalance()`
- `int getNumber()`

- `boolean deposit(double amount)`
- `boolean withdraw(double amount)`

In addition, you are provided with a `toString()` method below that you must not modify.

Account class toString() code

```
@Override
public String toString() {//DO NOT MODIFY
    return "(" + getName() + ", " + getNumber() + ", " +
            String.format("$%.2f", getBalance()) + ")";
}
```

The Constructor

The constructor's signature is:

```
public Account(String name, int number,
               double initialBalance)
```

In typical use, it could create an account for "Alice Jones" with an account number of 1234 and an initial balance of $100.00 with:

```
Account alice = new Account("Alice Jones", 1234, 100.0);
```

When you implement the constructor, you must also think about the instance variables you will need. (Hint: recall that a constructor often simply copies its arguments into instance variables. Also, if you have declared your instance variables, a Netbeans tool can be used to generate the code for the constructor automatically.)

The Getters

You should have little trouble implementing the getter methods (`getName()` returns the name of the account owner, `getNumber()` returns the account number and `getBalance()` returns the current balance.)

(Hint: The getters can be generated automatically in Netbeans as one of the Refactoring choices.)

The withdraw() and deposit() methods

Note that these methods return a boolean: `true` or `false`. If successful, they return `true`. However, an attempt to withdraw can fail if the balance is not big enough or if an attempt is made to withdraw zero or a negative value. A deposit can fail if the amount is negative or zero.

Of course, a successful deposit or withdrawal should update the account balance.

The Bank Class

You should not attempt to implement the Bank class until you have Account working

properly.

The Bank implementation will require that you use arrays and loops for the first time.

Unlike the `Account` class, you are given a template for this class (`Bank.java`) so you don't have to start it from scratch. And, to make your life a little easier and ensure that you start off the right way, the instance variables and the constructor have been given to you as well. (You should not modify the furnished constructor and you do not need any additional instance variables.)

Source Code

`Bank.java`, `MainAccount.java` and `MainBank.java` classes are provided with the lab. More details about these classes are in the instructions below.

Step 1: Create a Netbeans Project and *Account* class

1. Create a Netbeans project called `BankAccounts`.
2. Create a Java file (class library type) called `Account` and set the package to `coe318.lab4`. Note: you are not provided with a template this time, you have to write it from scratch. Do not modify the automatically generated statement:

```
package coe318.lab4;
```

3. Determine your instance variables and implement the constructor.
4. Implement the other methods.
5. You are provided with a `toString()` method for the Account class (above) which you should copy and paste into your class.
6. You may wish to write your own `public static void main(String[] args)` in the Account class itself. Write whatever you want to see if your methods are working.

Step 2: Test *Account* with *MainAccount*

1. Create a Java file (class library type) called `MainAccount` with package `coe318.lab4` and copy and paste the provided source code from `MainAccount.java`.
2. You can run this main method either by invoking Run file or changing the Netbeans configuration to specify the class containing the main method.
3. You should not continue until at least most of it works.

Correct output from MainAccount

```
(Bob, 789, $0.00)
(Alice, 123, $100.00)
(Alice, 123, $85.00)
```

```
(Alice, 123, $85.00)

(Alice, 123, $85.00)

(Alice, 123, $35.00)

(Bob, 789, $300.00)

(Bob, 789, $200.00)
```

Step 3: Create the *Bank* Class

1. Create a Java file called `Bank` and copy/paste the provided template code.
2. Create a Java file `MainBank` which will be used to test your code. Copy the provided `MainBank.java` code into this file. (Reset the main method in the Netbeans configuration to make this your main class)
3. Fix the methods until it works.

Correct output from MainBank

```
Toronto Dominion: 0 of 3 accounts
open Toronto Dominion: 1 of 3
accounts open

  (Charles, 234,
$200.00) td has
account # 456: true

Toronto Dominion: 2 of 3 accounts
  open (Charles, 234, $200.00)

  (Dora, 456, $300.00)

Bank of Montreal: 1 of 5 accounts
  open (Edward, 456, $400.00)
```

Step 4: Submit your lab

You must submit your lab electronically on D2L. Please make sure you hand over the quiz answer sheet to the TA at the end of the in-class quiz.

Please zip up your NetBeans project containing all source files and submit to the respective assignment folder on D2L.