Assignment 02 - Bank Accounts

CSCI 242 - Computer Science II :: Version 2.0.2

Due date is posted in the LMS.

Please read the entire assignment description before beginning the project.

Objective

The objectives of this assignment are:

1. To reinforce the concepts of inheritance and polymorphism in Java.
2. To understand how to implement a class given its UML class diagram.

To accomplish this, you will implement a superclass from which other classes will inherit. You will also write a driver class that instantiates the objects and delivers the program output.

Introduction

Do you have a bank account? It is pretty hard to get around in this world without at least one bank account! In fact, you probably have several bank accounts: savings, checking/debit and maybe a college education savings account.

Bank accounts are examples of the concepts of *inheritance* and *polymorphism*. All back accounts share certain properties and behaviors. Yet, each account is slightly different from the other.

The Assignment

Your assignment is to implement a superclass called `Account`. From there, you will implement two subclasses, `Savings` and `Checking`, that will inherit from `Account` and another class, `CollegeSavings`, that inherits from `Savings`.

To accomplish all of this, you will be given a UML class diagram that describes the classes and class inheritance structure. From this class diagram (and a description), you will implement your classes.

All the project requirements are listed below. This is an *individual* assignment so you are not to show your code to any other student. You are allowed to discuss the assignment with classmates but you are not to show your code in any way to any other student.

In general:

- Your project must compile error-free to be considered for grading. If your program does not compile without errors, you will receive a grade of 0 (zero).
- Your project must comply with the course guidelines found in "Coding Guidelines" in the LMS, **Home > General Course Information > Coding and Documentation Guidelines**. This includes documenting using Javadoc. Deductions will be made for programs that to not comply with the document.
- The format of your output must match exactly that of your instructors. See the section titled "Sample Run" below. Note that you should use your own data so the data will be different but the ***format*** should be the same. Deductions will be made for programs that to not comply with output requirements.

- Make sure all constant values are coded as Java constants (i.e., final), not "hard-coded" into the code itself.
- You are not allowed to make changes to any visibility modifiers for instance variables or methods. Visibility modifiers are as shown in the class diagram.

Package:	`edu.uwp.cs.csci242.assignments.bankaccount`
Project Name:	`A02_BankAccount`
Zip filename:	`bankaccount.zip`

The Classes

The class diagram shown in ***Figure 1*** describes the classes you are to use to implement your bank accounts. Note that the class diagram uses a "modified version" of UML that describes the classes in a slightly more Java-like format. For example, method parameter names are given to help you understand the parameters usage.

The Account Class

The `Account` class is the "top-level" superclass. The class represents a general bank account object. The following requirements pertain to the `Account` class:

- The member variables usage is as shown in the class diagram.
- `withdraw()` and `deposit()` have a lower limit for the amount to deposit or withdraw. It does not make sense to withdraw or deposit less that $1.00. If a user violates this lower limit, the methods should throw an `IllegalArgumentException`. Both return the new account balance.
- The private method `makeId()` randomly generates a new account identifier (id). Accounts are 10 "digits" long and only contain the "digits" zero through nine. ***HINT***: Use Java's `Random` object along with a `StringBuilder` to randomly select characters from a set of known characters.

The Checking Class

The `Checking` class *is an* `Account`. The following requirements pertain to the `Checking` class:

- The three-arg constructor class calls the `Accounts` constructor.
- `getInterest()` returns the interest earned but does not modify the account.
- `applyInterest()` returns the interest earned and modifies the account.
- The interest earned is 2% of the balance over $700 and is calculated using simple interest.

The Savings Class

The `Savings` class *is an* `Account` and is very similar to the Checking class, as the class diagram shows. The main difference is how interest is calculated. Interest earned for savings is 1.25% of the balance and has no limit. Interest for Savings is calculated using compound interest.

The CollegeSavings Class

The `CollegeSavings` class *is a* `Savings`. The following requirements pertain to the `CollegeSavings` class:

- The `withdraw()` method overrides `Savingss'` `withdraw()`. There is a penalty of 10% for withdrawing from the college savings account if the withdraw is not for college.
- `withdrawForCollege()` is the method to use when withdrawing to pay tuition.

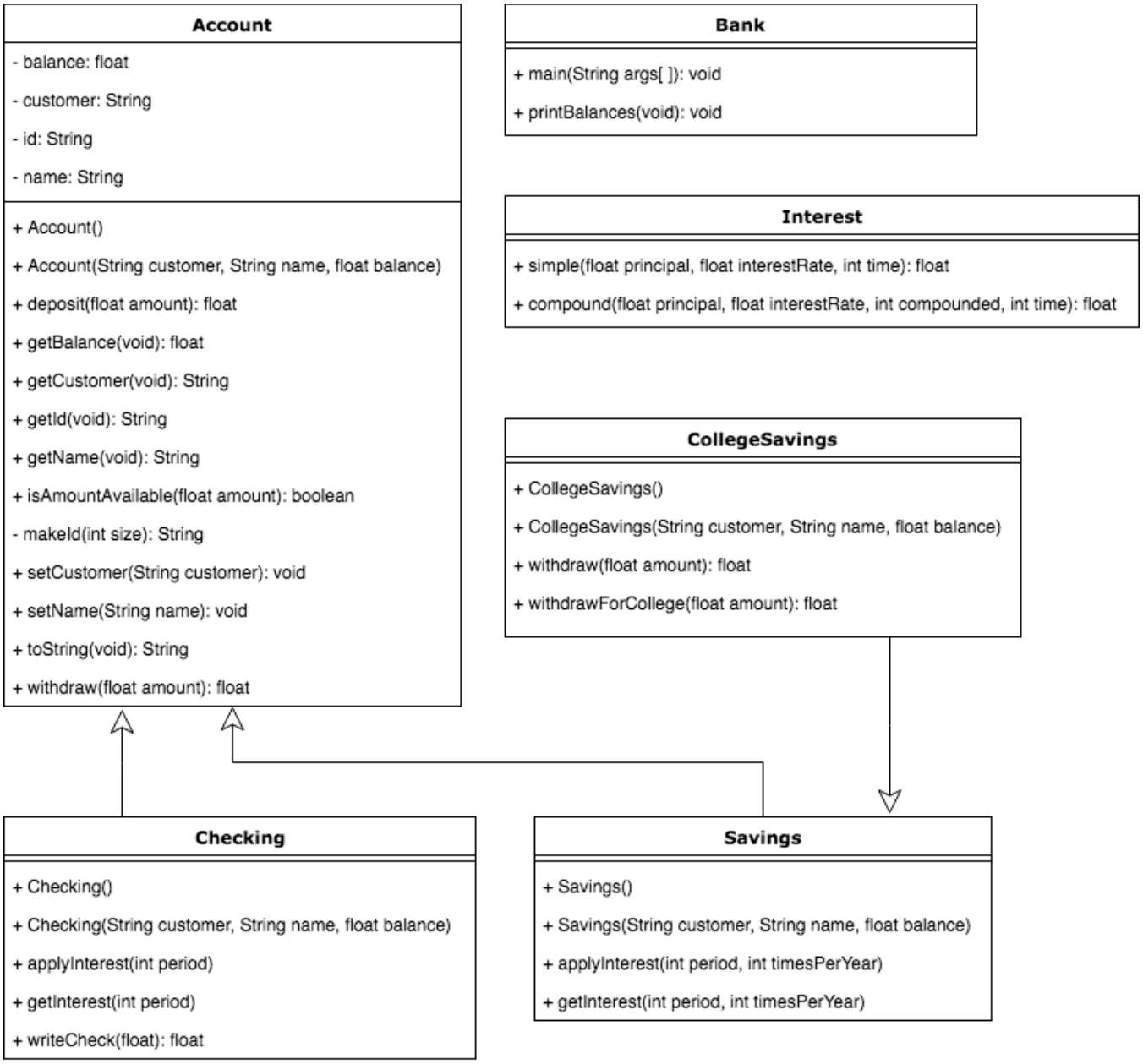

Figure 1 - The BankAccount class diagram

The Bank Class

The `Bank` class is the main driver. This class should use the other classes to build accounts that a fictional person will use. The `Bank` class will create three accounts: a savings account, a checking account and a college savings account. In addition, the Bank class will perform transactions against these accounts. A sample run is shown below in "Sample Run".

The Interest Class

The interest class encapsulates calculating interest for accounts. The two methods implement the calculation for simple interest and compound interest. For compound interest, the interest is compounded monthly.

Sample Run

The following is a sample run of the `Bank` main program:

```
+-- Customer: Jared Burgess ---
| Current balance of checking [2741549269], 'Checking account' is: $2,000.00
| Current balance of college savings [7760777070], 'College savings account' is: $10,000.00
| Current balance of savings [4474263176], 'Savings account' is: $1,000.00
+----------------------------
Got a paycheck: $5000.0.  1000 in college savings, then half in savings & half in checking.
+-- Customer: Jared Burgess ---
| Current balance of checking [2741549269], 'Checking account' is: $4,000.00
| Current balance of college savings [7760777070], 'College savings account' is: $11,000.00
| Current balance of savings [4474263176], 'Savings account' is: $3,000.00
+----------------------------
Time for interest!.
+-- Customer: Jared Burgess ---
| Current balance of checking [2741549269], 'Checking account' is: $4,066.00
| Current balance of college savings [7760777070], 'College savings account' is: $11,138.29
| Current balance of savings [4474263176], 'Savings account' is: $3,037.72
+----------------------------
Write some bills...
> House mortgage, $150.00...
> House mortgage, $1975.45...
> Gas and electric, $145.68...
> Water, 60.34...
> TV/Internet/phone, $277.45...
+-- Customer: Jared Burgess ---
| Current balance of checking [2741549269], 'Checking account' is: $1,457.08
| Current balance of college savings [7760777070], 'College savings account' is: $11,138.29
| Current balance of savings [4474263176], 'Savings account' is: $3,037.72
+----------------------------
Live life...
> Groceries, $226.90...
> Night out, $177.80...
> Fix car, $733.66...
+-- Customer: Jared Burgess ---
| Current balance of checking [2741549269], 'Checking account' is: $318.72
| Current balance of college savings [7760777070], 'College savings account' is: $11,138.29
| Current balance of savings [4474263176], 'Savings account' is: $3,037.72
+----------------------------
Vacation, $4000.00...
> Funds not available...  Must transfer...
> Not enough in checking+savings... Take from college fund at a penalty...
> Funds available... going on vacation!
+-- Customer: Jared Burgess ---
| Current balance of checking [2741549269], 'Checking account' is: $318.72
| Current balance of college savings [7760777070], 'College savings account' is: $9,024.46
| Current balance of savings [4474263176], 'Savings account' is: $37.72
+----------------------------
Tuition, $8000.00...
+-- Customer: Jared Burgess ---
| Current balance of checking [2741549269], 'Checking account' is: $318.72
| Current balance of college savings [7760777070], 'College savings account' is: $1,024.46
| Current balance of savings [4474263176], 'Savings account' is: $37.72
+----------------------------
```

Submission

Zip up your entire project folder into a single zip file named `A02.zip` and submit that one file. Submit the Zip file to the LMS dropbox labeled `Assignment 02 – Bank Account`. I should be able to unzip the entire project and load the entire project without incident.

Grading

See the document titled, “Programming Rubric” found at **Home > General Course Information** the for details on how the program will be graded.

Ω