Java Puzzle Ball MOOC
Lab 3: Finish the Inheritance Structure

Overview

The `SavingsAccount` and `CheckingAccount` classes of Lab 2 duplicated a lot of the same fields and methods. When this occurs between classes, it's an indication that it may be beneficial to create some sort of inheritance structure. This reduces duplication and simplifies code editing. With an inheritance structure in place, sub classes inherit the fields and methods of their super class. Sub classes can contain their own unique methods not found in the super class, or override inherited methods. This structure is already setup between the abstract `Account` class and `CheckingAccount`. Study these classes and use your observations to complete the rest of the lab.

Tasks

Download the Lab 3 Start State. This is a zip (JPB_MOOC_Lab3.zip) containing a NetBeans project. This project is slightly different from the Lab 2 solution. Unzip the file and open the project in NetBeans. The program contains an abstract `Account` class, `SavingsAccount`, `CheckingAccount`, and `TestClass` classes. An inheritance structure is partially setup between these classes. Your goal is to finish this work. You'll also need to clean up the output.

Abstract Account

The role of the abstract `Account` class is to contain the code which all types of bank accounts have in common.

Checking Account

The `CheckingAccount` class is already written to inherit from the `Account` class. Study this code to guide your edits of the `SavingsAccount` class. Remember, checking accounts are incapable of earning interest.

Savings Account

Allow this class to inherit from the `Account` class. Remove fields and methods that are better inherited rather than explicitly written in the class. Leave the fields and methods that are unique to the class. Provide any modifications to inherited methods whose implementation should be overridden.

Test Class and Main Method

The `main` method currently creates and tests the same instances as it did in the Lab 2 solution file. Play with the `main` method to create instances and test changes you've made. You'll also notice a few things about the output that could be improved. You'll need to make these changes in the other classes in your program.

- The output can get long and complicated. It's difficult to distinguish where a new account is created. When a new account is created, clearly state that it's a new account and automatically print the account details. You shouldn't need to call `printDetails()` from the `main` method
- When an error occurs (like depositing a negative number), it's difficult to distinguish which account is affected. Automatically print account details following stats following an error message.
- Ensure there are no unnecessary blank lines when printing an account's details.

Sample Output

```
New Account:
Savings Account #0
Account Owner: Duke
Balance: $100.0
Interest Rate: 0.02

New Account:
Savings Account #1
Account Owner: Damien
Balance: $200.0
Interest Rate: 0.02

New Account:
Savings Account #2
Account Owner: Jessica
Balance: $500.0
Interest Rate: 0.02

New Account:
Savings Account #3
Account Owner: Herbert
Balance: $500.0
Interest Rate: 0.02

Deposit: $5000.0
Savings Account #0
Account Owner: Duke
Balance: $5100.0
Interest Rate: 0.02

Widthdraw: $100.0
Savings Account #0
Account Owner: Duke
Balance: $5000.0
Interest Rate: 0.02

Interest: $8.347222222222223
Savings Account #0
Account Owner: Duke
Balance: $5008.333333333334
Interest Rate: 0.02

Interest: $8.361134259259261
Savings Account #0
Account Owner: Duke
Balance: $5016.680555555557
```

```
Interest Rate: 0.02

New Account:
Checking Account #4
Account Owner: Duke
Balance: $0.0

New Account:
Checking Account #5
Account Owner: Duke
Balance: $500000.0

Cannot deposit negative amount: -5000.0
Checking Account #4
Account Owner: Duke
Balance: $0.0

Deposit: $5000.0
Checking Account #4
Account Owner: Duke
Balance: $5000.0

Cannot withdraw negative amount: -10000.0
Checking Account #4
Account Owner: Duke
Balance: $5000.0

Cannot withdraw $10000.0 from $5000.0 account
Balance cannot go negative.
Checking Account #4
Account Owner: Duke
Balance: $5000.0

Withdraw: $10000.0
Checking Account #5
Account Owner: Duke
Balance: $490000.0
```