Project6
CECS277 Fall 2021
Due October 13th 11:59 pm
Submit project6.zip before the deadline

Please make sure to follow the naming Please make sure to follow the naming convention for your project. If your project does not run because of the naming issues, you won't receive any credit.

`project6.zip should include` **`project6 Package`** and **`project6.jar`**

project6 package will include the class **BankAccount.java**, **SavingsAccount.java**, and **SavingsDemo.java**.

Plagiarism/Academic Integrity Policy

Cheating and plagiarism will not be tolerated in this course. Students are to do their own assignments. Cases of copying, cheating, and plagiarism of assignments and/or tests, and any other violations, will be pursued to the maximum extent permitted by the University, which can include expulsion from the University. This applies equally to students who intentionally assist other students in academic dishonesty.

Any form of plagiarism or cheating will result in a failing grade on the assignment (at a minimum), and could result in a failing grade in the course or even university disciplinary action.

To learn more about the University policy on Cheating and Plagiarism, visit:

Academic Information and Regulations-Cheating and Plagiarism

`SavingDemo is the main class.`

`BankAccount` and `SavingsAccount` Classes

Design an abstract class named `BankAccount` to hold the following data for a bank account:

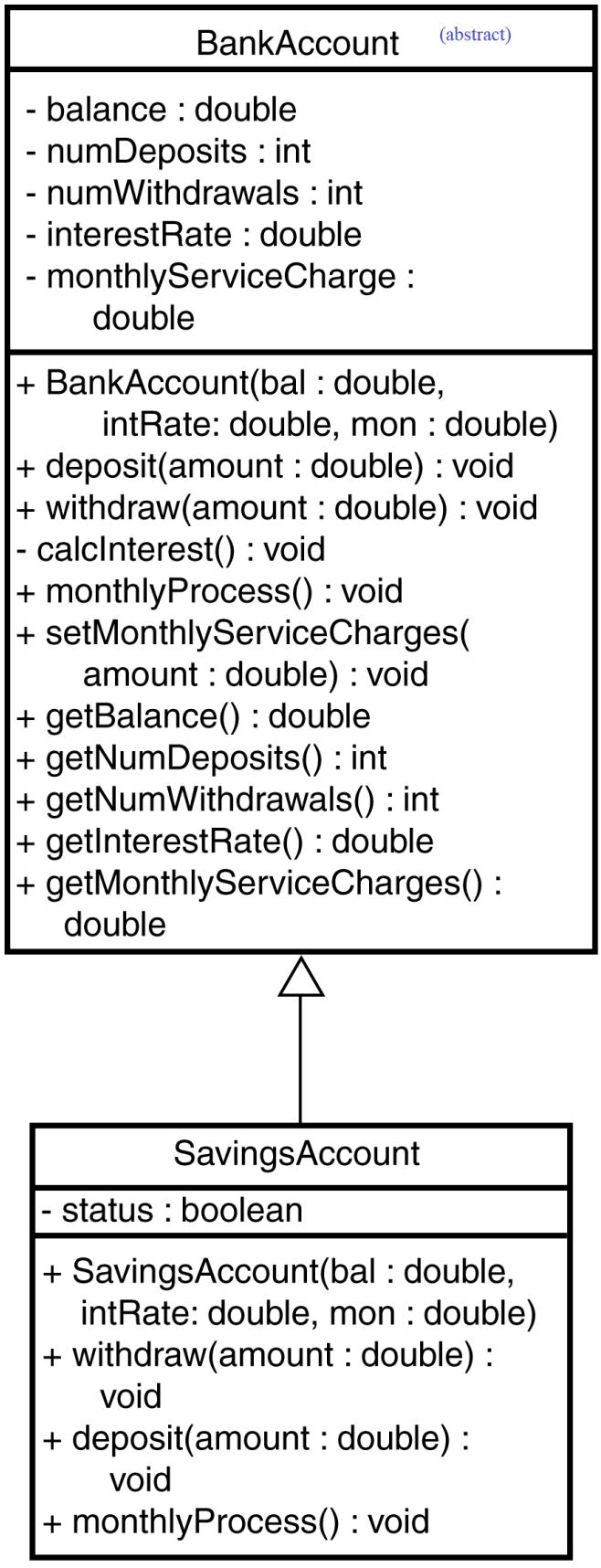
- Balance
- Number of deposits this month
- Number of withdrawals
- Annual interest rate
- Monthly service charges

The class should have the following methods:

Constructor	**The constructor should accept arguments for the balance and annual interest rate.**
deposit	A method that accepts an argument for the amount of the deposit. The method should add the argument to the account balance. It should also increment the variable holding the number of deposits.
withdraw	A method that accepts an argument for the amount of the withdrawal. The method should subtract the argument from the balance. It should also increment the variable holding the number of withdrawals.
calcInterest	A method that updates the balance by calculating the monthly interest earned by the account, and adding this interest to the balance. This is performed by the following formulas: *Monthly Interest Rate = (Annual Interest Rate/12)* *Monthly Interest = Balance * Monthly Interest Rate* *Balance = Balance + Monthly Interest*
monthlyProcess	A method that subtracts the monthly service charges from the balance, calls the calcInterest method, then sets the variables that hold the number of withdrawals, number of deposits, and monthly service charges to zero.

Next, design a `SavingsAccount` class that extends the `BankAccount` class. The `SavingsAccount` class should have a status field to represent an active or inactive account. If the balance of a savings account falls below $25, it becomes inactive. (The `status` field could be a `boolean` variable.) No more withdrawals can be made until the balance is raised above $25, at which time the account becomes active again. The savings account class should have the following methods:

`withdraw`	A method that determines whether the account is inactive before a withdrawal is made. (No withdrawal will be allowed if the account is not active.) A withdrawal is then made by calling the superclass version of the method.
`deposit`	A method that determines whether the account is inactive before a deposit is made. If the account is inactive and the deposit brings the balance above $25, the account becomes active again. The deposit is then made by calling the superclass version of the method.
`monthlyProcess`	Before the superclass method is called, this method checks the number of withdrawals. If the number of withdrawals for the month is more than 4, a service charge of $1 for each withdrawal above 4 is added to the superclass field that holds the monthly service charges. (Don't forget to check the account balance after the service charge is taken. If the balance falls below $25, the account becomes inactive.)


BankAccount
(abstract)
- balance : double
- numDeposits : int
- numWithdrawals : int
- interestRate : double
- monthlyServiceCharge : double
+ BankAccount(bal : double, intRate: double, mon : double)
+ deposit(amount : double) : void
+ withdraw(amount : double) : void
- calcInterest() : void
+ monthlyProcess() : void
+ setMonthlyServiceCharges(amount : double) : void
+ getBalance() : double
+ getNumDeposits() : int
+ getNumWithdrawals() : int
+ getInterestRate() : double
+ getMonthlyServiceCharges() : double
SavingsAccount
- status : boolean
+ SavingsAccount(bal : double, intRate: double, mon : double)
+ withdraw(amount : double) : void
+ deposit(amount : double) : void
+ monthlyProcess() : void

s

Sample output in one run:

```
 Create a SavingsAccount object with a $100 balance,  3% interest rate, and a monthly service charge of $2.50
Balance: $100.00
Number of deposits: 0
Number of withdrawals: 0

Please enter the amount you want to deposit: $10
Do you want to make another deposit?yes
Please enter the amount you want to deposit: $20
Do you want to make another deposit?yes
Please enter the amount you want to deposit: $35
Do you want to make another deposit?no
 Display what we've done so far.

Balance: $165.00
Number of deposits: 3
Number of withdrawals: 0
```

```
How much do you want to Withdrawal? $100
Do you want to make another Withdrawal?yes
How much do you want to Withdrawal? $50
Do you want to make another Withdrawal?yes
How much do you want to Withdrawal? $10
Do you want to make another Withdrawal?yes
How much do you want to Withdrawal? $1
Do you want to make another Withdrawal?yes
How much do you want to Withdrawal? $1
Do you want to make another Withdrawal?no
 Display what we've done so far.

Balance: $15.00
Number of deposits: 3
Number of withdrawals: 2

Doing the monthly processing.
Balance: $12.53
Number of deposits: 0
Number of withdrawals: 0
```