CS F213 : OOP Lab 3

Constructor and Method Overloading

March 9, 2021

The objective of this lab is to test your understanding of constructor and function overloading in Java. In this lab you will simulate the working of a Bank. For this task, you are required to create 3 classes - Account, Customer and Bank.

General Instructions

1. Make sure that the source code is written in the default package.
2. Ensure that the name of each file is the same as the corresponding class name.
3. Ensure that the name of methods is the same as those given in Javadoc. Failure to comply may cause deduction in credit even if the implementation is correct.
4. You won't be provided with a template file (helper code) this time . You have to implement all the classes by yourself (including their respective constructors).
5. Do not copy directly the names of methods from Javadoc. Write them out yourselves.
6. You are required to upload all three classes at once. It is not possible to test individual classes on Codepost.
7. Be sure to check your code on the test cases provided locally (as jar file), before submitting on Codepost. Remember that Codepost takes only the latest submission and not the best submission into consideration.
8. The advisable order of writing the code:

   a) Account b) Customer c) Bank
9. Only the one account, with **index 0** in the Accounts array, can be a savings account.

NOTE : The below description of classes is extremely brief. Only the names of fields and methods have been written (along with a 1-liner description wherever required). Students MUST refer to the javadoc for the complete details (use, return types, arguments, etc.) of all methods and fields .

Class Descriptions

1. **Account**
   - **Fields**
     1. accountNumber
     2. accountType
     3. interestRate
     4. accountCurrentBalance
     5. minBalance
     6. maturityPeriod
   - **Constructors**
     1. Account(int account_number)
     2. Account(int account_number, double min_balance)
     3. Account(int account_number, double balance, int maturity_period)
   - **Methods**
     1. getAccountNumber()
     2. getAccountType()
     3. getInterestRate()
     4. getBalance()
     5. getMaturityPeriod()
     6. deposit(double depositAmount)
     7. withdraw(double withdrawAmount)
     8. matureFD()
     9. yearEnd()

2. Customer

- **Fields**
  1. customerName
  2. accounts
  3. maxAccounts - To be initialised to 5
  4. numAccounts
- **Constructors**
  1. Customer(customerName)
  2. Customer(customerName, minBalance)
- **Methods**
  1. getCustomerName()
  2. getNumberOfAccounts()
  3. addFDAccount(amount, numYears)
  4. depositAmount(amount)
  5. withdrawAmount(amount)
  6. fixedDepositsMatured()

3. **Bank**
   - **Fields**
     i. customers
     ii. maxCustomers - To be initialised to 10
     iii. numCustomers
   - **Constructors**
     i. Bank()
   - **Methods**
     i. addCustomer(Name)
     ii. addCustomer(Name, minBalance)
     iii. depositMoney(Name, amount)
     iv. endYear()
     v. getNumCustomers()
     vi. matureFixedDeposit(Name)
     vii. transferMoney(sender, receiver, amount)
     viii. withdrawMoney(Name, amount)

Test Cases

Method	Marks Allotted
addCustomer	0.5
withdrawMoney	0.5
openFD	1
transferMoney	1
endYear	1
depositMoney	1
fixedDepositsMatured	2