Second National Bank

You are to write a program that will keep track of 2 types of bank accounts - Checking and Saving - using an **array**. Your program will allow us to perform specific operations on the bank accounts stored in the array. Your program will have **4** classes defined as follows:

**BankAccount**

- 3 instance variables:
  - private String accountNumber
  - private double balance
  - private String accountHolderName

- 1 static variable:
  - private static int numberOfAccounts
  - keep track of the number of accounts opened so far

- 8 non-static methods:
  - public void deposit(double amount) - this method must only accept the amount if it is a non-negative number
  - public void withdraw(double amount) - it is possible for this method to allow a negative balance.
  - public double getBalance()
  - public void setAccountHolderName(String accountHolderName)
  - public String getAccountHolderName()
  - public void setAccountNumber(String accountNumber)
  - public String getAccountNumber()
  - public void printReport() - this method prints a report for the account including the account number, the owner of the account, and the balance.

- 1 static method:
  - public static int getNumberOfAccounts() - returns the number of accounts opened so far.

- 2 constructors:
  - BankAccount (String accountNumber, double balance, String accountHolderName) – This constructor must call the various setters of the class (including deposit()) in order to initialize the object's values. It should not directly initialize any variables (e.g. this.balance = balance is not allowed. Instead, use deposit(balance) to initialize the value).
  - BankAccount(String accountNumber, String accountHolderName) – This constructor must call the first constructor with a default balance of **0.0**.
SavingAccount extends BankAccount

- Override the withdraw() method of the super class to only allow 6 withdrawal operations per saving account. If a valid request to withdraw is made (that is, there have only been 6 or fewer withdrawals), this method must invoke the withdraw() method of its superclass in order to perform the operation.
- Ensure that withdraw() will not leave a negative balance
- For each constructor in the superclass, create a constructor that accepts the same parameters, and invoke the appropriate constructor from the superclass.

CheckingAccount extends BankAccount

- Override the withdraw() method of the super class to allow the following:
  - Allow a withdraw operation to exceed the the available balance by up to $100. If a withdrawal brings the balance below zero (0), subtract an additional $10 from the remaining balance as an overdraft protection fee.
- Add a method writeCheck(double amount) that also performs a withdrawal, however this method can exceed the available balance by any amount.
  - If a writeCheck() operation exceeds the balance by $100 or less, subtract an additional $10 from the remaining balance as an overdraft fee (you should invoke the withdraw() method to do this).
  - If the operation exceeds the available balance by more than $100, subtract an additional $35 from the remaining balance (do not invoke the withdraw() method for this. Instead, implement the logic within the writeCheck() method).
- For each constructor in the superclass, create a constructor that accepts the same parameters, and invoke the appropriate constructor from the superclass.

Bank.java

This class will contain your main method, along with a method to perform each of the following subtasks. Please read the following instructions fully and carefully:

1. Add a new account – This will ask the user to choose a type of account, and create either a new SavingAccount or CheckingAccount object and add it to the array if there is room. If the array is full, it should print an error message. It should ask the user to enter an account number, a name, and a balance. If all 3 values are entered, your program should use the appropriate constructor to create the object. If the balance is entered as 0 or less, your program should use the constructor that accepts two strings. If any other value is entered incorrectly (e.g. no name or account number) the program should print an error and no object should be created.
2. Deposit to an account – This subtask will ask the user to enter the account number that should receive the deposit, as well as the deposit amount. It will the cycle through the array containing the BankAccount objects until it finds one that matches the account number entered by the user. Once the correct object has been found, it should make the deposit by calling that object's deposit() method. If the account is not found, it should print an error message.
3. Withdraw from an account – Similar to the deposit subtask above.
4. Write a check – This task should ask the user to enter the account number for a Bank Account. If the account number that is entered is for a CheckingAccount object, then it should invoke the object’s writeCheck() method with an amount entered by the user. If the account number is for a SavingAccount object, the program should print an error message explaining that saving accounts do not have check writing privileges.
5. Sort all accounts entered so far and print a report for each – This method will sort each bank account based on their current balance in descending order. Once sorted, it will cycle through the array and call the printReport() method of each object. It must also print the total number of accounts, as well as the total and average balance of all accounts (Please note that you do not have a static variable in the BankAccount class to keep track of the total balance. You must therefore calculate it each time this subtask is executed.).
6. Search for a Bank Account – This subtask will ask the user to enter a full or partial name, and then search the array for a bank account that contains the entered string. It must call the printReport() method for each bank account that it finds that is a match.

When the program first runs, the user should be asked how many accounts in total can be entered, and an array initialized to hold the given number of BankAccount objects. Afterwards, use JOptionPane to ask the user which of the aforementioned tasks she would like to perform, along with an option to exit. When an option is selected, invoke the appropriate method in your main class to perform the given subtask. The user should be continuously prompted until they choose to exit.

Upload a zip file containing all classes.

DUE DATE: December 7th, at 11:59 PM