Part I

Here is the pickcode to start. There is a new class - TestBankAccounts which I created for you to test some new methods out. You will need to uncomment lines in Main and TestBankAccounts for testing purposes.

1. Together we are going to add an acct number that every account has, it will increment one after the other. Create a static method getNumOfAccounts

2. Suppose the bank wants to keep track of how much money is in all BankAccounts. In BankAccount:
    - Declare a private static variable to represent this sum.
    - Add code to the constructor and necessary methods(deposit...) that will manage this variable.
    - Add a static method public static double getTotalMoney that returns the total amount of money in all BankAccounts. Think about why this method should be static – its information is not related to any particular BankAccount.
    - Test this by uncommenting a line in printInfo of TestBankAccounts

3. Add a method public void close() to your BankAccount class. This method should close the current BankAccount by appending (means adding to end) “CLOSED” to the BankAccount name and setting the balance to 0. (The BankAccount number should remain unchanged.) This is going to affect how much money the bank has. Deal with this.
    - Test this by uncommenting line in Main and also testCloseAccount of TestBankAccounts

4. Add a method: public static BankAccount consolidate(BankAccount acct1, BankAccount acct2) to your BankAccount class that creates a new BankAccount object whose

balance is the sum of the balances in acct1 and acct2 and closes acct1 and acct2. The new BankAccount should be returned. Two important rules of consolidation:

- Create getter methods in BankAccount - getName, getAcctNum, getBalance
- header would look like: `public static BankAccount consolidate(BankAccount acct1, BankAccount acct2)`
- Only BankAccounts with the same name can be consolidated. The new BankAccount gets the name on the old BankAccounts but a new BankAccount number. The new interest rate is 0.
- Two BankAccounts with the same number cannot be consolidated. Otherwise this would be an easy way to double your money! Return null if they cant do it.
- So if they can be consolidated:
  i. Find total in both accounts **double total =...**
  ii. Find name of either account; **String name=...**
  iii. Close old accounts
  iv. Create new account with name and total (and interest of 0) -> ie: **BankAccount combined=new BankAccount(___, ___)**
  v. Return that **BankAccount**
- To test, uncomment in main and bank account

Check these conditions before creating the new BankAccount. If either condition fails, do not create the new BankAccount or close the old ones; print a useful message and return null.

Part II Transfering funds

5. Add a method public boolean transfer(`BankAccount` receivingAcct, double amount) to the BankAccount class that allows the user to transfer funds from one bank BankAccount to another. If you call `acct1.transfer(receivingAcct,957.80)` should transfer $957.80 from acct1 to receivingAcct. Be sure to clearly document which way the transfer goes!
   - Test this out
6. Add a static method to the BankAccount class that lets the user `transfer` money between two BankAccounts without going through either BankAccount. You can (and should) call the method `transfer` just like the other one – you are overloading this method. Your new method should take two BankAccount objects and an

amount and transfer the amount from the first BankAccount to the second BankAccount. The signature will look like this:

public static boolean transfer(BankAccount sendingAcct, BankAccount receivingAcct, double amount)Test this

These are some headers below for BankAccount:

```
public static BankAccount consolidate(BankAccount acctOne, BankAccount
acctTwo) {
		// TODO Auto-generated method stub
		return null;
	}

	public boolean transfer(BankAccount receivingAcct, double amount)
	{
		// TODO Auto-generated method stub

	}

	public static boolean transfer(BankAccount sendingAcct, BankAccount
receivingAcct, double amount)
	{
		// TODO Auto-generated method stub

	}
```

Exceeds

Start with this Bank class. This class will be an interface for bank workers. When the constructor is called print out a menu of choices for the user. Those choices

- a. create new account
- b. select individual account
- c. consolidate accounts (this will only work if there are atleast 2 accounts and you have one free)
- d. transfer funds
- e. list open accounts
- f. list bank info

- g. exit

If b is chosen

- a. make deposit
- b. make withdrawl
- c. close
- d. exit to main menu

Have the ability to have at least 3 accounts.