**A Bank Account Class**

**Part I Method (40 points)**

1. File *Account.java* contains a partial definition for a class representing a bank account. Save it to your directory and study it to see what methods it contains. Then complete the Account class as described below. Note that you won't be able to test your methods until you write ManageAccounts in question #2.

a. Fill in the code for method *toString*, which should return a string containing the name, account number, and balance for the account.

b. Fill in the code for method *chargeFee*, which should deduct a service fee from the account.

c. Modify *chargeFee* so that instead of returning void, it returns the new balance. Note that you will have to make changes in two places.

d. Fill in the code for method *changeName* which takes a string as a parameter and changes the name on the account to be that string.

2. File *ManageAccounts.java* contains a shell program that uses the Account class above. Save it to your directory, and complete it as indicated by the comments.

3. Modify ManageAccounts so that it prints the balance after the calls to chargeFees. Instead of using the getBalance method like you did after the deposit and withdrawal, use the balance that is returned from the chargeFees method. You can either store it in a variable and then print the value of the variable, or embed the method call in a println statement.

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Account.java

//

// A bank account class with methods to deposit to, withdraw from,

// change the name on, charge a fee to, and print a summary of the account.

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

public class Account

{

private double balance;

private String name;

private long acctNum;

//----------------------------------------------

//Constructor -- initializes balance, owner, and account number

//----------------------------------------------

public Account(double initBal, String owner, long number)

{

balance = initBal;

name = owner;

acctNum = number;

}

//----------------------------------------------

// Checks to see if balance is sufficient for withdrawal.

// If so, decrements balance by amount; if not, prints message.

//----------------------------------------------

public void withdraw(double amount)

{

if (balance >= amount)

balance -= amount;

else

System.out.println("Insufficient funds");

}

//----------------------------------------------

// Adds deposit amount to balance.

//----------------------------------------------

public void deposit(double amount)

{

balance += amount;

}

//----------------------------------------------

// Returns balance.

//----------------------------------------------

public double getBalance()

{

return balance;

}

//----------------------------------------------

// Returns a string containing the name, account number, and balance.

//----------------------------------------------

public String toString()

{

}

//----------------------------------------------

// Deducts $10 service fee

//----------------------------------------------

public void chargeFee()

{

}

//----------------------------------------------

// Changes the name on the account

//----------------------------------------------

public void changeName(String newName)

{

}

}

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// ManageAccounts.java

//

// Use Account class to create and manage Sally and Joe's

// bank accounts

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

public class ManageAccounts

{

public static void main(String[] args)

{

Account acct1, acct2;

//create account1 for Sally with $1000

acct1 = new Account(1000, "Sally", 1111);

//create account2 for Joe with $500

//deposit $100 to Joe's account

//print Joe's new balance (use getBalance())

//withdraw $50 from Sally's account

//print Sally's new balance (use getBalance())

//charge fees to both accounts

//change the name on Joe's account to Joseph

//print summary for both accounts (use toString() implicitly)

}

}

**Part II Static Variable and Static Method (30 points)**

Suppose the bank wants to keep track of how many accounts exist.

* 1. In *Account.java,*
     + - 1. Declare a private static integer variable numAccounts to hold this value. Like all instance and static variables, it will be initialized (to 0, since it’s an int) automatically.
         2. Add code to the constructor to increment this variable every time an account is created.
         3. Add a static method *getNumAccounts* that returns the total number of accounts. Think about why this method should be static – its information is not related to any particular account.
  2. Modify *ManageAccounts.java* so at the end of the program it uses the *getNumAccounts* method to find how many accounts were created. Save it to your directory, then use it to test your modified Account class.

**Part III Method Overloading (30 points)**

1. Overload the constructor as follows:

* public Account (double balance, String name) – initializes the balance and owner as specified; randomly generates the account number.
* public Account (String name) – initializes the owner as specified; sets the initial balance to 0 and randomly generates the account number.

1. Overload the *withdraw* method with one that also takes a fee and deducts that fee from the account.

File *TestAccount.java* contains a simple program that exercises these methods. Save it to your directory, study it to see what it does, and use it to test your modified Account class.

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// TestAccount.java

//

// A simple driver to test the overloaded methods of

// the Account class.

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

import java.util.Scanner;

public class TestAccount

{

public static void main(String[] args)

{

String name;

double balance;

long acctNum;

Account acct;

Scanner scan = new Scanner(System.in);

System.out.println("Enter account holder's first name");

name = scan.next();

acct = new Account(name);

System.out.println("Account for " + name + ":");

System.out.println(acct);

System.out.println("\nEnter initial balance");

balance = scan.nextDouble();

acct = new Account(balance,name);

System.out.println("Account for " + name + ":");

System.out.println(acct);

System.out.println("\nEnter account number");

acctNum = scan.nextLong();

acct = new Account(balance,name,acctNum);

System.out.println("Account for " + name + ":");

System.out.println(acct);

System.out.print("\nDepositing 100 into account, balance is now ");

acct.deposit(100);

System.out.println(acct.getBalance());

System.out.print("\nWithdrawing $25, balance is now ");

acct.withdraw(25);

System.out.println(acct.getBalance());

System.out.print("\nWithdrawing $25 with $2 fee, balance is now ");

acct.withdraw(25,2);

System.out.println(acct.getBalance());

System.out.println("\nBye!");

}

}

**Part IV (Extra Credit: 30 points)**

Suppose the bank wants to offer the service of consolidating two accounts (meaning two accounts belong to the same person)

* 1. In *Account.java,* add a method void close() to your Account class. This method should close the current account by appending “CLOSED” to the account name and setting the balance to 0. (The account number should remain unchanged.) Also decrement the total number of accounts.

1. In *Account.java,* add a static method *Account consolidate(Account acct1, Account acct2)* to your Account class that creates a new account whose balance is the sum of the balances in acct1 and acct2 and closes acct1 and acct2. The new account should be returned. Two important rules of consolidation:

* Only accounts with the same name can be consolidated. The new account gets the name on the old accounts but a new account number (you can pick any number).
* Two accounts with the same number cannot be consolidated. Otherwise this would be an easy way to double your money!

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

1. Write another test program *TestConsolidation.java* (Do not use *ManageAccounts.java*) that prompts for and reads in three names and creates an account with an initial balance of $100 for each. Print the three accounts, then close the first account and try to consolidate the second and third into a new account. Now print the accounts again, including the consolidated one if it was created.

**Submission**

Total Points (Part I, II, and III): 100 points

Submit *Account.java, ManageAccounts.java, and TestAccount.java*

Extra Credit for Part IV: 30 points

Submit *Account.java* and *TestConsolidation.java*