Week 6 Spring 2013 Labs

A Flexible Account Class

File *Account.java* contains a definition for a simple bank account class with methods to withdraw, deposit, get the balance and account number, and return a String representation. Note that the constructor for this class creates a random account number. Save this class to your directory and study it to see how it works. Then modify it as follows:

1. Overload the constructor as follows:

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

2. 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.

Counting Transactions

File *Account.java* (see **A Flexible Account Class** exercise) contains a definition for a simple bank account class with methods to withdraw, deposit, get the balance and account number, and return a String representation. Note that the constructor for this class creates a random account number. Save this class to your directory and study it to see how it works. Now modify it to keep track of the total number of deposits and withdrawals (separately) for each day, and the total amount deposited and withdrawn. Write code to do this as follows:

1. Add four private static variables to the Account class, one to keep track of each value above (number and total amount of deposits, number and total of withdrawals). Note that since these variables are static, all of the Account objects share them. This is in contrast to the instance variables that hold the balance, name, and account number; each Account has its own copy of these. Recall that numeric static and instance variables are initialized to 0 by default.

2. Add public methods to return the values of each of the variables you just added, e.g., *public static int getNumDeposits()*.

3. Modify the *withdraw* and *deposit* methods to update the appropriate static variables at each withdrawal and deposit

4. File *ProcessTransactions.java* contains a program that creates and initializes two Account objects and enters a loop that allows the user to enter transactions for either account until asking to quit. Modify this program as follows:

- After the loop, print the total number of deposits and withdrawals and the total amount of each. You will need to use the Account methods that you wrote above. Test your program.
- Imagine that this loop contains the transactions for a single day. Embed it in a loop that allows the transactions to be recorded and counted for many days. At the beginning of each day print the summary for each account, then have the user enter the transactions for the day. When all of the transactions have been entered, print the total numbers and amounts (as above), then reset these values to 0 and repeat for the next day. Note that you will need to add methods to reset the variables holding the numbers and amounts of withdrawals and deposits to the Account class. Think: should these be static or instance methods?

Transfering Funds

File *Account.java* (see **A Flexible Account Class** exercise) contains a definition for a simple bank account class with methods to withdraw, deposit, get the balance and account number, and print a summary. Save it to your directory and study it to see how it works. Then write the following additional code:

1. Add a method *public void transfer(Account acct, double amount)* to the Account class that allows the user to transfer funds from one bank account to another. If *acct1* and *acct2* are Account objects, then the call *acct1.transfer(acct2,957.80)* should transfer $957.80 from acct1 to acct2. Be sure to clearly document which way the transfer goes!

2. Write a class TransferTest with a main method that creates two bank account objects and enters a loop that does the following:
   - Asks if the user would like to transfer from account1 to account2, transfer from account2 to account1, or quit.
   - If a transfer is chosen, asks the amount of the transfer, carries out the operation, and prints the new balance for each account.
   - Repeats until the user asks to quit, then prints a summary for each account.

3. Add a static method to the Account class that lets the user transfer money between two accounts without going through either account. You can (and should) call the method transfer just like the other one – you are overloading this method. Your new method should take two Account objects and an amount and transfer the amount from the first account to the second account. The signature will look like this:

   ```
   public static void transfer(Account acct1, Account acct2, double amount)
   ```

   Modify your TransferTest class to use the static transfer instead of the instance version.