Programming Assignment 8: Objects and Classes

Total Points (40 pts) - Due Wednesday, March 8 at 11:59 PM

This assignment will give you practice with creating classes and defining new types of objects.

Part 1: Counting Transactions (25 points)

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. The constructor for this class creates a random account number.
Save this class to your directory and study it to see how it works.

Write code to do the following:

Constructors

1. Overload the constructor as follows:
   a. public **Account (double initBal, String owner, int number)** - initializes the balance, owner, and account number as specified
   b. public **Account (double initBal, String owner)** - initializes the balance and owner as specified; randomly generates the account number.
   c. public **Account (String owner)** - initializes the owner as specified; sets the initial balance to 0 and randomly generates the account number.

Methods

2. Overload the **withdraw** method with one that also takes a **fee** and deducts that fee from the account.
3. Suppose the bank wants to keep track of how many accounts exist.
   a. 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.
   b. Add code to the **constructor** to increment this variable every time an account is created.
   c. 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.
4. 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.
5. Add public methods to return the values of each of the variables you just added, e.g., **public static int getNumDeposits().**
6. Modify the **withdraw** and **deposit** methods to update the appropriate static variables at each withdrawal and deposit.

Part 2: Process Transactions (15 points)

Write a class **ProcessTransactions** with a main method that creates two bank account objects and enters a loop that allows the user to enter transactions for either account until asking to quit. 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?

Sample Run 1:

```
There are 2 accounts
The following accounts are available:

Account number: 1234
Account owner: Sue
Account balance: 1000.0

Account number: 5678
Account owner: Joe
Account balance: 1000.0

Enter the number of the account you would like to access: 1234
Would you like to make a deposit (D) or withdrawal (W)? D
Enter the amount: 700

More transactions? (y/n)y

Enter the number of the account you would like to access: 1234
Would you like to make a deposit (D) or withdrawal (W)? W
Enter the amount: 600

More transactions? (y/n)y

Enter the number of the account you would like to access: 5678
Would you like to make a deposit (D) or withdrawal (W)? W
Enter the amount: 3000
Insufficient funds

More transactions? (y/n)y

Enter the number of the account you would like to access: 5678
Would you like to make a deposit (D) or withdrawal (W)? D
Enter the amount: 5000

More transactions? (y/n)n
Total number of deposits: 2
Total number of withdrawals: 0
Total amount of deposits: 5700
```

```
Total amount of withdrawals: 0
Would you like to enter transactions for another day? (y/n)y
The following accounts are available:

Account number: 1234
Account owner: Sue
Account balance: 1100.0

Account number: 5678
Account owner: Joe
Account balance: 6000.0

Enter the number of the account you would like to access: 1234
Would you like to make a deposit (D) or withdrawal (W)? W
Enter the amount: 600

More transactions? (y/n)y

Enter the number of the account you would like to access: 1234
Would you like to make a deposit (D) or withdrawal (W)? D
Enter the amount: 800

More transactions? (y/n)y

Enter the number of the account you would like to access: 5678
Would you like to make a deposit (D) or withdrawal (W)? D
Enter the amount: 769

More transactions? (y/n)n
Total number of deposits: 2
Total number of withdrawals: 0
Total amount of deposits: 1569
Total amount of withdrawals: 0
Would you like to enter transactions for another day? (y/n)n
```

Sample Run 2:

```
There are 2 accounts
The following accounts are available:

Account number: 1234
Account owner: Sue
Account balance: 1000.0

Account number: 5678
Account owner: Joe
Account balance: 1000.0

Enter the number of the account you would like to access: 1234
Would you like to make a deposit (D) or withdrawal (W)? W
Enter the amount: 700

More transactions? (y/n)y

Enter the number of the account you would like to access: 5678
Would you like to make a deposit (D) or withdrawal (W)? D
Enter the amount: 7000
```

```
More transactions? (y/n)y

Enter the number of the account you would like to access: 1234
Would you like to make a deposit (D) or withdrawal (W)? W
Enter the amount: 8000
Insufficient funds

More transactions? (y/n)n
Total number of deposits: 1
Total number of withdrawals: 0
Total amount of deposits: 7000
Total amount of withdrawals: 0
Would you like to enter transactions for another day? (y/n)y
The following accounts are available:

Account number: 1234
Account owner: Sue
Account balance: 300.0

Account number: 5678
Account owner: Joe
Account balance: 8000.0

Enter the number of the account you would like to access: 7897
Would you like to make a deposit (D) or withdrawal (W)? W
Enter the amount: 78
Sorry, invalid account number.

More transactions? (y/n)y

Enter the number of the account you would like to access: 5678
Would you like to make a deposit (D) or withdrawal (W)? W
Enter the amount: 89

More transactions? (y/n)n
Total number of deposits: 0
Total number of withdrawals: 0
Total amount of deposits: 0
Total amount of withdrawals: 0
Would you like to enter transactions for another day? (y/n)n
```

Prevent Point Loss:

- Always include both a source and (for console apps) a run.
- Do not add or change any characters on the console output lines.
- Indent correctly and convert all tabs to 3 spaces each.
- Filter input. Any mutator, constructor, or other member methods (function) that uses a passed parameter as a basis for setting private data, should test for the validity of that passed parameter - that means range checking, string length testing, or any other test that makes sense for the class and private member.
- Use symbolic names, not literals. Never use a numeric literal like 1000, 3 or 52 for the size of an array, or the maximum value of some data member in your code. Instead, create a symbolic constant and use the constant. In other words, use ARRAY_SIZE or MAX_CARDS, not 1000 or 52.
- You should comment your code with a heading at the top of your class with your name, section, and a description of the overall program. All method headers should be commented as well as all complex sections of code. Comments should explain each method's behavior, parameters, return values, and assumptions made by your code, as appropriate.
- Properly encapsulate your objects by making data your fields private.
- Properly use indentation, good variable names, and types. Do not have any lines of code longer than 80 characters.

Submission Instructions

- Execute the program and copy/paste the output that is produced by your program into the bottom of the source code file, making it into a comment. I will run the programs myself to see the output.
- Make sure the run "matches" your source. If the run you submit could not have come from the source you submit, it will be graded as if you did not hand in a run.
- Use the Assignment Submission link to submit the source code file.
- Submit the following file:
    - Account.java
    - ProcessTransactions.java
- Do not submit **.class** files.