CS 535 OO Programming & Design
Fall Semester, 2013
Assignment 6

San Diego State University -- This page last updated 11/26/13

Assignment 6
Due Dec 12, 11:59 pm

1. (25 points - 15 working/meets requirements, 5 style/design, 5 unit tests) Banks need to be able to handle transactions in different currencies. We will start off supporting US dollars, Mexican Pesos and Canadian dollars. Since the currency symbol for al three currencies is "$" we will write the currencies with the three letter country code after the amount. For example we will write $10.00 USD, $20.00 CAD and $30.00 MXN for 30 US dollars, 20 Canadian dollars and 30 Mexican Pesos. There is a single space between the last digit in the amount and the country code. As the result "Currency fromString: '1.00'" is ambiguous. The form accepted by fromString: has to include the $ sign and the country code as in "Currency fromString: '$1.00 MXN'". There is a similar problem with fromNumber. So now the format of the method will be:

    Currency fromNumber: 3.12 as: 'CAD'.

   We also have a problem with currency arithmetic which is illustrated below.

```
| dollars pesos sum |
dollars := Currency fromString: '$1.00 USD'.
pesos := Currency fromString: '$2.00 MXN'.
sum := dollars + pesos.
```

   What should sum be? The rule we will use is that when adding or subtraction currency objects the second currency object is converted into the currency of the first object, then the arithmetic is done. So the sum above will be in USD. On November 21, 2013 the exchange rate for converting a MXN Peso to USD is 0.0770. (This is the bid rate according to OANDA. See http://www.oanda.com/currency/historical-rates/). So $2.00 MNX is equal to

    2 MXN * 0.0770 USD/MXN = 0.1540 USD

   Our bank will truncate currency exchanges so 4 MXN * 0.0770 USD/MXN = 0.30 USD. So the sum above should be $1.15 USD if the calculation was done on Nov 21, 2013. When depositing and withdrawing money from an account one uses the exchange rate at the day of the transaction, not the day the transaction is processed.

   Currency exchange rates change over time. We will assume that the exchange rates change daily. Files of previous exchange rates will be provided. When changing currency one needs to know the date the currency is being changed. This can affect currency arithmetic. Above the result of "dollars + pesos" depends on that date the addition is done on. When transactions are processes, deposits and withdrawals, and currency needs to be

converted use the date of the transaction timestamp for the date of the currency conversion. So the "+" operator may not be adequate for adding currency.

2. (15 points - 5 working/meets requirements, 5 style/design, 5 unit tests) All fees are also in USD. Our bank has decided to add two more types of customers: premium and gold. Premium customers are allowed to have there balance become -$10,000 USD before withdrawals are cancelled. The fee for exceeding that amount is $700 USD. Gold customers are allowed to have their balance become -$100,000 USD. The fee for exceeding that amount is $5,000 USD.

3. (45 points - 25 working/meets requirements, 15 style/design, 5 unit tests) Up to now we have been dealing with a single bank account. Now we will deal with multiple accounts. This requires some changes. First each account needs an account ID, which is a 12 digit string, for example '055012581'. Each bank has a routing number which is a 9 digit string, for example '000032451283'. So it takes 21 digits to identify an account as different banks can use the same account number. We will write the above example as ':055012581:000032451283'. The routing number for our bank will be '055012581'. All account information below is given in the format ':ROUTINGNUMBER:ACCOUNTNUMBER' without the quotes.

   As a result the format of NewAccount transaction is:

   transactionID transactionTimestamp NewAccount AccountID name customerType amount

   Where "name" can have multiple parts, each separated by spaces. AccountID is a 12 digit string. "customerType" is either "Normal", "Preferred", "Gold", "Premium". "amount" is an amount like: "$25.00 MXN". When an account is created the currency of the initial deposit will be the currency of the account. A sample NewAccount transaction is:

   551253 10.15.2013 17:30 NewAccount 000032451283 Roger Whitney Normal $250.13 CAD

   "Deposit Cash" transactions need account information. So the format becomes:

   transactionID transactionTimestamp Deposit Cash AccountInformation amount

   Where AccountInformation is of the form ':ROUTINGNUMBER:ACCOUNTNUMBER'. For example:

   557362 10.30.2013 19:31 Deposit Cash :055012581:000032451283 $13.15 USD

   "Deposit Check" transactions need account information for two accounts, the from account and the to account. These transactions also need a check number. So the format becomes:

   transactionID transactionTimestamp Deposit Check FromAccountInformation ToAccountInformation checkNumber hold amount

   The account information is in the same format as above. For example:

557364 10.30.2013 19:31 Deposit Check :049012581:000032451283 :055012581:000032451283 121 3 $13.15 USD

The Cancel transaction requires account information to be added. The AccountInformation is the account the check was deposited into. The format becomes:

transactionID transactionTimestamp Cancel AccountInformation transactionIDToCancel

For example:

182126 10.15.2013 17:30 Cancel :055012581:000032451283 17433

There are two types of withdrawals: Cash and Check. The Cash withdrawal format is:

transactionID transactionTimestamp Withdrawal Cash AccountInformation amount

Where AccountInformation is the account that cash is being depositing into. An example is:

4103 10.29.2013 19:29 Withdrawal Cash :055012581:000032451283 $13.15 MXN

A check withdrawal needs to have account information of both accounts involved. The format is:

transactionID transactionTimestamp Withdrawal Check FromAccountInformation ToAccountInformation checkNumber amount

An example is:

4103 10.29.2013 19:29 Withdrawal Check :049012581:000032451283 :055012581:000032451283 $13.15 MXN

Handling Cash withdrawals and deposits is straight forward. Dealing with checks is a bit more complex. The first transaction with a check is a deposit. The deposit of a check generates a withdrawal transaction in the bank account the check is write on. If the check bounces (that is the account does not have enough money to handle the check) then a cancel transaction is generated for the account the check was deposited into. In assignment 5 all these transactions were generated for you. Since we will be dealing with all accounts in a bank, your code has to generate some of the transactions. There are thee cases, depending on which accounts are involved in the transactions. Two accounts are involved in a Check. The account the check is deposited into and the account the check is drawn on. If both accounts are in our bank then our input file will contain the deposit transaction. We will need to generate the withdrawal transaction and send it to the correct account. If the account the check is drawn on can not cover the check a cancel transaction needs to be sent to the account the check was deposited into. If the check from outside our bank is deposited into an account in our bank then you need to generate the withdrawal transaction to send to the other bank (see below). If the check bounces the cancel transaction will be in the input file. If the check is drawn on an account in our bank and deposited in

account in another bank, then a withdrawal transaction will appear in the input file. If the account can not cover the check then you have to generate the Cancel transaction to send to the other bank.

We will send transactions to other banks by the outgoingTransactions method on the Bank class.

Bank Class

For grading purposes your project needs a Bank class. It needs the following methods:

Bank class>>routingNumber: aString
Returns a Bank object with the given routing number.

Bank>>transactionsFrom: aFilename
The bank processes the transactions in the file. The file can contain information about multiple accounts. It can contain multiple NewAccount transactions. A NewAccount transaction will occur before any other transactions on that account. This method can be called multiple times.

Bank>>outgoingTransactions
Returns a collection of transactions for other banks generated from this bank. If no such transactions exist then return an empty collection.

Bank>>account: aString
aString is an ACCOUNTNUMBER. Returns the bank account with the given account number.

How to turn in the assignment

In your image create Package called Assignment6. Make sure that all the code for this assignment is in your Assignment6 package. When you add a method to existing Smalltalk classes only include in your package the methods that you write. You will upload your Assignment6 package to your store account.