**CS160 Project 3**

**Due: November 12, 11:59 PM Total Points: 100 pts**

## Problem Statement

You have been hired by a travel firm. Design and implement a program to quote currency exchange rates for different currencies from different banks. The advantage is the firm’s clients can get the best rates on foreign currency when traveling.

## Problem Description

Our project requires us to define Banks. Each Bank can exchange three currencies to and from US dollars. For each currency, there is a rate at which the bank will buy and sell the currency in exchange for US dollars. In addition, the bank charges a commission as a fee for performing the exchange. All of this information is specified in a text file for each bank, which will be in the following format:  
  
bank name

commission\_rate

currency\_code exchange\_rate

currency\_code exchange\_rate

currency\_code exchange\_rate

As an example, consider these two banks:

Bank1.txt:

Bank Of 'Merica

0.11

EUR 0.89

AUD 1.41

RUB 64.85

Bank2.txt:

This Is a Trustworthy Bank

0.30

EUR 0.95

JPY 110.5

RUB 5.85

The first bank above is called Bank of ‘Merica and takes a 20% commission. It will trade .89 Euros for 1 US dollar, 1.41 Australian dollars for 1 US dollar, and 64.85 Russian Rubles for 1 US dollar. The project assumes two banks, in two files – bank1.txt and bank2.txt.

Your code will first read in the bank files to establish the banks we are using. After reading the bank information, your program should prompt the user to enter

* the type of transaction ("buy" or "sell")
* the currency code (e.g., "EUR" or "eur" or "Eur")
* the amount of the foreign currency to buy or sell

Then the program will display quotes from the two banks for the proposed currency exchange. All of this input and output happens in the console. Here is an example which defines the messages you should print (output is in normal font, **input is bold font**) and how you should handle incorrect input:

Do you want to buy or sell? **what**

Please try again.

Do you want to buy or sell? **buy**

Which currency? **SCHRUTEBUCKS**

I'm sorry, There is no bank that can exchange SCHRUTEBUCKS. Please start again.

Do you want to buy or sell? **sell**

Which currency? **jpy**

There is one bank that can exchange JPY.

How many JPY? **4858.28**

This is a Trustworthy Bank will give you 30.78 USD for 4858.28 JPY, after collecting a commission of 13.19 USD (30%)

Do you want to perform another transaction? **yes**

Do you want to buy or sell? **buy**

Which currency? **Eur**

There are two banks that can exchange EUR.

How many EUR? **1000**

Bank of ‘Merica will give you 1000.00 EUR for 1247.19 USD after collecting a commission of 123.60 USD (11%)

This is a Trustworthy Bank will give you 1000.00 EUR for 1368.42 USD after collecting a commission of 315.79 USD (30%)

Do you wish to perform another transaction? **no**

Have a nice day.

All currency amounts must be rounded to two digits. The commission percentage is rounded to the nearest whole percent. The currency code is always printed in UPPER CASE as it appears in the bank file. The user’s input should be accepted with any capitalization they wish, though.

# Project Requirements

The requirements for this project are provided as a list of classes that must be created in your implementation of the project. This is your first project writing your own classes and making objects of them. You may add other methods not included here if you desire, but your code must include the following classes with the specified methods:

### **Currency Class (10 points)**

Each currency line in the bank data files will be translated into an object instance of the Currency class that should contain the following:

* [2 pts] Two private data fields, one to hold the currency code and one to hold the exchange rate. Remember that the exchange rate for the same currency can be different for different banks.
* [4 pts] A constructor that takes two arguments: the currency code and exchange rate. The constructor should assign the value of the private members from the values of the arguments.
* [4 pts] An accessor method for each data field.

### **Bank Class (34 points)**

Each bank will be represented by an object instance of a Bank class, which should contain the following:

* [5 pts] Five private data fields
  + Name of the bank
  + Commission rate
  + Three fields for the three currencies exchanged by the bank. Each of these should have data type Currency.
* A constructor that takes a file name (a String) as an argument. The constructor should...
  + [2 pts] Open the file.
  + [8 pts] Read the bank information from the file and use it to initialize all of the bank's data fields.
* [3 pts] A method called supportCurrency that takes a currency code (a String) as its only argument. It should return true if the bank can exchange the given currency and false if it cannot.
* [4 pts] A method called getRate that takes a currency code (a String) as its only argument. It should return the rate of exchange for that currency, or -1.0 if the bank does not exchange that currency.
* Two methods: quoteBuy and quoteSell. Both methods takes two arguments: a String for the currency code and a double for the amount of foreign currency. Each method will return a Quote object (defined below) if the bank supports that currency and null if the bank does not. Call the getRate method you wrote within your quote methods.  
  Use the following formulas to calculate the number of US dollars and the bank's commission:
  + For selling *amt* units of currency with exchange rate exch*Rate*, the number of US dollars before commission is *amt* / exch*Rate*; call that result *base*. Then the commission is the commission rate *comRate* (between 0.0 and 1.0) times *base* and the amount of US dollars the customer will be paid is *base* minus the *commission*.
    - base = amt / exchRate
    - commission = comRate \* base
    - totalPay = base - commission
  + For buying *amt* units of currency with exchange rate exch*Rate* and commission rate *comRate* (between 0.0 and 1.0), the number of US dollars the customer will pay is amt \* exchRate; call that result dollarsOwed. The commission is comRate \* dollarsOwed and the total cost is dollarsOwed \* commission.
    - dollarsOwed = amt \* exchRate
    - commission = dollarsOwed \* comRate
    - totalCost = dollarsOwed + commission

[For quoteBuy and quoteSell, the method prototype is worth 1 point, the correct calculation is worth 3 points, and creating and returning a Quote object is worth 2 points.]

### **Quote Class (16 points)**

You will define a Quote class to capture the information about a potential currency exchange. It should contain the following:

* [3 pts] Seven private data fields:
  + Name of the bank issuing the quote
  + Currency code of the currency given to the bank. Call this codeToBank.
  + Currency amount given to the bank. Call this amtToBank.
  + Currency code of the currency received from the bank. Call this codeFromBank.
  + Currency amount received from the bank. Call this amtFromBank.
  + Commission rate charged by the bank.
  + Dollar amount of the commission.
  + A DecimalFormat object for formatting currencies (i.e., two digits after the decimal point).
* [3 pts] A constructor that takes 7 arguments, corresponding to the first 7 data fields listed above. The constructor should set those fields from the values of the arguments and create the DecimalFormat object.
* [10 pts] A method called toString that takes no arguments and returns a String. It should return the message that will be printed out for the quote, as listed above (i.e., "Shady Airport Bank will give you 1000.00 CAD for 843.91 USD, after collecting a commission of 168.78 USD (20%)"). The commission amount should always be reported in US dollars.

### **CurrencyExchange Class (25 points)**

Your *main* method should be in a CurrencyExchange class. It should follow this control flow:

* [2 pts] Create two Bank objects, one for each file "bank1.txt" and "bank2.txt"
* do{
  + [1 pt] Prompt the user if they want to buy or sell and get the user input
  + [2 pts] If the input is not valid, print error message and use continue to restart the loop
  + [1 pt] Prompt the user for the foreign currency and get the user input
  + [3 pts] Check how many banks support that currency
    - [1 pt] If none, print the error message and use continue to restart the loop
    - [1 pt] Otherwise, print how many banks can exchange that currency
  + [1 pt] Prompt the user for the amount of foreign currency and get the user input
  + [3 pts] Use the quoteBuy or quoteSell methods of the Bank objects to get a Quote from every bank supporting the foreign currency.
  + [4 pts] Print the quote(s) from the bank(s), using the toString method of the Quote object(s).
  + [4 pts] Use the console to prompt the user for another transaction
* [2 pts] while the user wants to continue performing transactions
* Reference the sample output above when developing this loop.

**Documentation Requirements**

* Each class and each method should have a JavaDoc style comment, using /\*\* \*/ commenting style above the class name and method name
* Code should be legibly indented
* Include a header comment at the top of the main file (in this case, CurrencyExchange), like:

**/\***

Thomas Bolinger

CS160 Project 3

\*/

## Testing

To test your code, try making different bank files and ensuring that your project works and calculates correctly. You should feel free to do the calculation by hand to ensure that the results are correct in your code.

## Submission

Submit all your code in a zip file (or your equivalent, like a rar or a tar) on Blackboard.

## Evaluation

**Correctness (85 points).** These points will be allocated as follows:

10 points for correct *Currency* class

34 points for correct *Bank* class

16 points for correct *Quote* class

25 points for correct *CurrencyExchange* main class

**Documentation and Design (15 points):** Your code will receive up to 15 points for following the documentation requirements specified above and following the design specified in the project document.