School of Computing
National University of Singapore
CS2010: Data Structures and Algorithms 2
Semester 2, AY 2015/16

Assignment 2 – Transaction Monitoring (via Trees and Heaps)
Assigned – Feb 2nd, 2016
Due – Feb 19 (Friday by 11.59pm), 2016 (in-lab assessment Feb 29, 2016)
Sorry, CNY has complicated the day we can do the assessment, so we will need to wait until after the break.

1) Objective

- Practice using Trees and Priority Queue ADT
- Applying ADT to realistic problems
- More fun with Java

2) Preparation

Download A2.zip. The zip contains the following: java source code (`TransMonitor.java`) and three (3) data files: `Visa.log, Mastercard.log, AmEx.log`.

3) The input

You are working in a data analytics company that provides services to FinTech (financial technology) companies who deal with payments. You are given data logs of payment transactions of thousands of customers over a period of time. The data is in the following format:

```
Name Amount Date Time
STRING $##.## DD/MM/YYYY HH:MM:SS
```

Example:

```
--
YU $1.21 28/01/2016 15:10:46
LIN $54.59 28/01/2016 15:12:56
AUGUSTA $131.46 28/01/2016 15:16:03
ADENA $65.22 28/01/2016 15:18:07
CAMILA $86.99 28/01/2016 15:20:11
TAMMARA $90.02 28/01/2016 15:22:12
LYNDIA $89.00 28/01/2016 15:24:20
--
```

In this log, we will assume that the names are unique, so each time a name appears it is the same person making another transaction on a different date for a different amount. For each data file, you don't know the number of names that will appear, or how many transactions each person may make. You may assume that the dates/time are always *increasing*.

3) Assignment requirements

To get an "average" grade, the minimum you are required to do is to accomplish task 3.1. This is the "basic requirement". To get an "above average" grade, you need to implement additional tasks as listed.

3.1 – Top K Transactions of All Users [Minimum Require for the Assignment (65/100 ~ B grade)]

a) From the console the user should be able to specify the `input_filename` and a value `K`.

b) Create a text file called `Report_input_filename_K.txt`. In this file, print out the names in the transaction log sorted with the top K transactions amount they made (you can omit the $ for the amounts). Note that some people may have less than K transactions.

A sample input would look like this:

```
Enter log file to open? Visa.log
Number of top transactions to find (K)? 5
```

Sample output: `Report_Visa.txt`

```
……
MANUELA : 136.95 136.50 136.22 134.98 134.77
MANY : 126.10 125.04 124.70 122.79 122.73
MAO : 113.88 113.78 113.41 113.09 112.82
MAPLE : 60.63 60.06 59.94 59.81 59.52
……
```

Additional tasks beyond the minimum requirement

3.2 [Average Amount] (+5) In addition to the top K transactions each person has made, also print the total number of transactions for that person and the average dollar amount of all transactions.

Example:

```
……
MANUELA : 136.95 136.50 136.22 134.98 134.77  # of transactions = 133; avg = $68.41
MANY : 126.10 125.04 124.70 122.79 122.73  # of transactions = 99; avg = $69.41
MAO : 113.88 113.78 113.41 113.09 112.82  # of transactions = 190; avg = $59.11
MAPLE : 60.63 60.06 59.94 59.81 59.52  # of transactions = 101; avg = $30.43
……
```

3.3 [Double Transactions] (+15) As you parse the transaction log, e*fficiently* detect "double transactions". This is when two transactions of the same amount by the same individual appears within 5 minutes of each other. Your implementation should not greatly slow down your program. You should write these double transactions at the top of your file, *before* you print the results for task 3.1.

Example:

```
……
Double Transactions
MARIANA : $525.60; 24/6/2016 21:16:39; $525.60; 24/6/2016 21:16:42
(others)
……
```

3.4 [Suspicious Transactions #1] (+5) For this task, you need to find people who have an unusual transactions. An unusual transaction is defined as the top transaction among the top-K transactions that is more than 5x the average of the remaining (K-1) transactions. This list of suspicious transactions should be reported after the list generated by part 3.1. List the names and transaction amounts that are considered suspicious.

3.5 [Suspicious Transactions #2] (+5) Include the date and time of the transaction as part of #5.

Example of 3.4 & 3.5

```
……
Suspicious Transactions
LAVERN : $870.00; 30/1/2016 2:43:28
(others)
……
```

3.6 [Big Spenders] (+15) Print a list of the 20 people who spent the most total money in the log. In this list, give their first and last transaction dates (no need for the time) and the total amount spent in that time frame. The "Big Spender List" should be reported at the end of the report.

Example:

```
……
Top 20 Spenders
ABIGAIL : Total Amount = $145669.92; First Transaction on 30/1/2016; Last Transaction on 2/6/2018
ADELLA : Total Amount = $135759.00; First Transaction on 30/1/2016; Last Transaction on 2/6/2018
ABBY : Total Amount = $72885.80; First Transaction on 31/1/2016; Last Transaction on 2/6/2018
……
```

4) Required Data Structures

One of the beauties of coding and software development is that you can solve a problem in many ways. It is often a creative task and you are in full control. However, since this is CS2010, you will be required for use the data structures we have learned in class.

Specifically, for the basic requirement 3.1:

1) You must use a Binary Search Tree (standard or balanced Tree) to store the names of the people in the log.

2) You must use a priority queue (heap) to compute the top K transactions.

You are free to use the code described in lecture (BST, Left-Leaning Red-Black Tree (BST), and PQ [heaps]) implementations, or write your own. You may also modify the class code, e.g. to make max-Heaps allow variable array sizes. You are also free (and encouraged) to use available Java classes that provide these ADT. If you do use built-in Java classes, make sure they implement the required data-structures (e.g. a Java LinkedList is not a BST).

For the additional requirements (3.2-3.6) you have no restrictions, but you need to justify your design – see submission instructions. You may not get full marks if your design is not efficient.

5) Submission Instruction

What to submit?

(1) Your source code (*.java files), (2) a short write-up describing your design for each requirement and (3) three sample reports: (Visa/MasterCard/AmEx). Do not include the data log files.

Submit the following:

1. Submit your code on IVLE before 11.59pm Feb 19 (late assignments will be deducted -15 pnts per day).
2. Please put your files in a folder and zip the folder. Use the following convention to name your zipped folder: MatriculationNumber_yourName_Assignment2. For example, if your matriculation number is A1234567B, and your name is Chow Yuen Fatt, for this assignment, your file name should be A1234567B_ChowYuenFatt_Assignment2.zip.
3. It is up to you to test to make sure that your Java code in the zipped file is complete. It is recommended that you unzip your Java code in a different location and test it to make sure it runs.

Assessment (Feb 29, 2016 – In Lab)

- You will be assessed individually in the lab session (you must attend your assigned lab!)
- You'll have approximately 4-5 minute Q&A session with Dr. Brown or one of the TAs (Hakki, Sixing or Abdel).
- We will run the code directly from your submitted IVLE submission, you won't be able to bring an updated version to the lab.
- We will run your code and then ask you anything about your code we want. Be prepared to answer questions.