CS2010 Assignment 2 Design Explanations

**3.1 Top K Transactions of All Users**

Given a line of transaction log from input file,

1. Void parser(String inputFileStr) parses the string.
2. Void parser(String inputFileStr) calls void addCustomer(String name, double amount, Date t), with the information it has parsed.
3. In Void addCustomer(String name, double amount, Date t), it creates a Transaction transaction.
4. It checks with the name of customer as key, whether this customer is present in the BST.
   * If present: update the customer object by adding the transaction object
   * Else: creates a new Customer object with relevant information
     1. maxHeap inside to maintain the top K transactions
5. After BufferedReader br finishes reading and parsing, iterator() from BST is used to sort the objects inside in chronological order and store them in Queue<Customer> q.
6. PrintWriter pw is used to print each Customer in q into the output file.

3.2 Average Amount

1. Customer class maintains three attributes, double sum, int numTransactions, double average.
2. Each time a new transactions is added into customer, these three attributes will be updated.
3. When required, accessors of its numTransactions and average are called to get these two values.

**3.3 Double Transactions**

1. In 3.1.4’s first bullet point, if present, besides updating the customer object, a method Boolean isDoubleTransaction(Transaction transaction) is called on the Customer object.
   1. Boolean isDoubleTransaction(Transaction transaction) calls the method contains(Transaction transaction) from PriorityQueue API, to check whether the maxHeap belonging to the customer contains the repeated transaction.
      1. To check the repeated transaction in maxHeap, Transaction.equals is overridden.
      2. In Transaction.equals, amount and date is compared.
      3. The transaction passed into the equals is stored into ArrayList<Transaction> doubleTransactions.
   2. Void processDoubleTransactoins() is called then, to get the ArrayList<Transaction> doubleTransactions from Transaction and convert the elements inside into string and put into ArrayList<String> doubleTransactions.
2. Pw then prints out strings from ArrayList<String> doubleTransactions into the destination file one by one.

3.4 & 3.5 Suspicious Transactions

1. During Step 6 in 3.1, a Queue<Customer> customerList is created to store Customer customer while pw is printing the element one by one.
2. processSuspiciousTransactions(customerList) is called to analyse customer in the customerList one by one.
3. Customer.hasSuspiciousTransaction() is called
   1. Top element is polled from maxHeap in the customer.
   2. K-1 elements is poled from maxHeap and their sum is added up.
   3. K-1 elements then are put back into maxHeap.
   4. If top element’s amount is greater than 5 times the average, true is returned.
   5. Else false is returned.
4. If true, transaction will be converted into a string and stored into ArrayList<String> suspiciousTransactions.
5. Pw then will print strings from ArrayList<String> suspiciousTransactions one by one into the file.

3.6 Big Spenders

1. PriorityQueue<Customer> is constructed at the start of the programme.
2. In Step 3 of 3.1, bigSpender adds every customer into its maxHeap.
3. bigSpenders maintains 20 top spenders in the heap.
4. Pw then prints out customer in bigSpenders one by one.