Lab 3, CSC/CPE 203 - collect data

Orientation

For this lab you will implement a program that mimics data analysis for an e-commerce site. You will implement a set of classes (as determined by you) to represent the data stored in a "log" file. You will then work with the data to compute various statistics on the collected data.

Objectives

- Understand the provided base code that reads data from a file into an array of strings
- Write code to process the data into appropriate data structures (including writing your own classes and creating hashmaps to store lists of those classes as appropriate)
- Write code that operates on the data in order to compute statistics about that data (including relating the various peices of data to one another and computing averages, differences, counting given specific instructions)

Given Files

Retrieve the files provided for this lab from Polylearn

These given files include base code to help in the task of analyzing log data. Your final program will need to read in e-commerce like data from a file and compute statistics based on the information in the log. The base code implements the file reading (but not the processing of the data read). You will need to complete the processing of the data for this lab, (including putting the data into appropriate data structures, i.e., not an array of strings). You are encouraged to read through the provided code.

Log File

Examine the provided log files. For example, look at the contents of `small.log`. You will find various entries representing customer use of an imaginary e-commerce site. Each entry will appear on a single line in the file and consist of an entry type tag followed by the corresponding entry attributes. The different types of entries and their attributes are as follows.

- **`START sessionId customerId`**
- **`VIEW sessionId productId price`**
- **`BUY sessionId productId price quantity`**
- **`END sessionId`**

Each `sessionId` is a unique identifier represented as a `String`, each `customerId` is a unique identifier represented as a `String`, each `productId` is a unique identifier represented as a `String`, each `price` is an integer number of cents, and each `quantity` is an integer.

Task

Your program will take the name of a log file as a command-line argument. It must then read the file and store the entries for processing. Your program will then output the results of computing the statistics discussed below.

You must determine how to represent the entries and which data structures to use for processing the data. Look at the provided code, there are several clues. Though you might be tempted to simplify the data (since not all of it is used in the required analysis), avoid that temptation for now. Define classes, as appropriate, to represent the entries (give consideration for the cohesion and coupling design principles).

You should note that within the provided source file there is an example of such processing and the use of a data structure (a map) to store some of the information. Based on the material discussed thus far, this data structure is used by first creating it and then passing it to another method to be populated. This then allows for the use of the populated data structure after the file processing. Some of you may not care for this mutation-based style, which is fine, but this is the most direct approach at this time.

Your program is only expected to properly analyze logically organized log files. In particular, for a given `sessionId`, all `VIEW` and `BUY` entries will come after a `START` entry and come before, if present, an `END` entry (it is conceivable that a session may not have ended when the snapshot of the log is considered).

Statistics

Your program must compute the following statistics.

- Print the average number of items viewed by visitors (as defined by session id) that do not make a purchase. In other words, over all `sessionId`s where a purchase *is not* made, compute the average number of views (VIEW entries). *Note that viewing the same item more than once should be counted - you are computing the average 'views' not the average number of items viewed.*
- For each `sessionId` associated with a purchase, print, for each productId, the purchase price (as cents) minus the average price of the items viewed during that session. (Keep in mind that there may be multiple purchases in a given session.) *Note that you do not need to account for the quantity of the purchased item, just compute the difference of the amount spent on a given product.*
- For each customer, for each product that customer purchased, print the number of sessions in which that customer viewed that product. *Note you are computing the number of sessions in which the purchased item was viewed, not the total number of views of that item.*

The output of your program when run on the provided `small.log` file should be as follows (the order within each breakdown is not significant).

```
Average Views without Purchase: 3.0

Price Difference for Purchased Product by Session
session4
   product1 -100.0
session2
   product1 -25.0
   product3 175.0
```

```
Number of Views for Purchased Product by Customer
customer2
   product1 1
customer1
   product1 2
   product3 2
```

How to tackle this lab...

This is a longer lab and requires you to solve some data processing challenges on your own. You might want to consider these steps.

- To start, make sure you understand the log file format.
- Consider class designs to store important entries from the log file.
- Look at main and how the data processing is proceeding (i.e. main calls populateDataStructures, which calls processFile, which calls processLine, which then processes the entries differently (you will write two of them)).
- Think about the collection data structure you will use to store all the data - you will allocate that data structure in main and then pass it to the relevant methods mentioned above until you get down to the point of actually creating objects and storing them. Get a grasp of the control flow and make sure the general idea of how you are going to store data makes sense - if it does not at this point, stop and talk to your instructor or neighbor.
- Fill in the important data processing steps (processViewEntry and processBuyEntry)
- Stop here and see if you can now uncomment the printOutExample method and have it print out your processed data. It should look as follows for `small.log`:

```
customer2
   in session4
           looked at product3
   in session5
           looked at product1
           looked at product3
           looked at product1
           looked at product4
customer1
   in session1
           looked at product1
           looked at product2
           looked at product3
   in session2
           looked at product1
           looked at product1
           looked at product1
           looked at product4
   in session3
           looked at product3
           looked at product2
```

- Now you are ready to write the code to compute statistics on your data; take each task one at a time. Write out pseudo-code first in your comments. What data do you need to iterate over for each step in the computation? Then replace your logic with actual Java syntax.
- Now test to make sure your output matches and you are done!

Submission

This lab is due by the end of lab in 1 week (Monday 10/15). Be sure to submit the code by the due date. Your instructor reserves the right to run further tests on your code. Demonstrate your working program to your instructor. Be prepared to show your source code. (Partial credit available up to instructor discretion).