Programming Exercise: Counting Website Visits

Assignment: Website Visits

In this assignment you will continue to build on the **LogEntry** and **LogAnalyzer** classes that you worked on in the last lesson. You will continue to use the method **parseEntry** from the **WebLogParser** class, and you should not modify this class. You will write several methods to solve problems about web logs. There are four small files you can use to test the methods: **short-test_log**, **weblog-short_log**, **weblog2-short_log**, and **weblog3-short_log**. You should write code to test the methods in a **Tester** class that creates a LogAnalyzer object.

Specifically, you should do the following:

- In the **LogAnalyzer** class, write the method **countVisitsPerIP**, which has no parameters. This method returns a HashMap<String, Integer> that maps an IP address to the number of times that IP address appears in **records**, meaning the number of times this IP address visited the website. Recall that **records** stores LogEntrys from a file of web logs. For help, refer to the video in this lesson on translating to code. Be sure to test this method on sample files.
- In the **LogAnalyzer** class, write the method **mostNumberVisitsByIP**, which has one parameter, a HashMap<String, Integer> that maps an IP address to the number of times that IP address appears in the web log file. This method returns the maximum number of visits to this website by a single IP address. For example, the call **mostNumberVisitsByIP** on a HashMap formed using the file **weblog3-short_log** returns 3.
- In the **LogAnalyzer** class, write the method **iPsMostVisits**, which has one parameter, a HashMap<String, Integer> that maps an IP address to the number of times that IP address appears in the web log file. This method returns an ArrayList of Strings of IP addresses that all have the maximum number of visits to this website. For example, the

call **iPsMostVisits** on a HashMap formed using the file **weblog3-short_log** returns the ArrayList with these two IP addresses, 61.15.121.171 and 84.133.195.161. Both of them visited the site three times, which is the maximum number of times any IP address visited the site.

- In the **LogAnalyzer** class, write the method **iPsForDays**, which has no parameters. This method returns a HashMap<String, ArrayList<String>> that uses **records** and maps days from web logs to an ArrayList of IP addresses that occurred on that day. A day is in the format "MMM DD" where MMM is the first three characters of the month name with the first letter capital and the others in lowercase, and DD is the day in two digits (examples are "Dec 05" and "Apr 22"). For example, for the file **weblog3-short_log**, after building this HashMap, if you print it out, you will see that Sep 14 maps to one IP address, Sep 21 maps to four IP addresses, and Sep 30 maps to five IP addresses.
- In the **LogAnalyzer** class, write the method **dayWithMostIPVisits**, which has one parameter that is a HashMap<String, ArrayList<String>> that uses **records** and maps days from web logs to an ArrayList of IP addresses that occurred on that day. This method returns the day that has the most IP address visits. If there is a tie, then return any such day. For example, if you use the file **weblog3-short_log**, then this method should return the day most visited as Sep 30.
- In the **LogAnalyzer** class, write the method **iPsWithMostVisitsOnDay**, which has two parameters—the first one is a HashMap<String, ArrayList<String>> that uses **records** and maps days from web logs to an ArrayList of IP addresses that occurred on that day, and the second parameter is a String representing a day in the format "MMM DD" described above. This method returns an ArrayList<String> of IP addresses that had the most accesses on the given day. For example, if you use the file **weblog3-short_log**, and the parameter for the day is "Sep 30", then there are two IP addresses in the ArrayList returned: 61.15.121.171 and 177.4.40.87. Hint: This method should call another method you have written.