CS601-01/CS601-02: Lab3 Part 2

Efficient HotelDataBuilder

Due Date: Oct 6, 11:59pm

For this part of the lab, you will make several changes to your **HotelDataBuilder** class:

1. JSON files with reviews should still be processed concurrently, but you are not allowed to use Executors, ExecutorService or any other classes from `java.util.concurrent` package.
2. You are required to use IBM's **WorkQueue** class that has been provided to you in the concurrent package of your project (note: we are **not** referring to Java's built in package *concurrent*). If you look at the code of class WorkQueue, you will see that it has an array of PoolWorker-s and a queue of Runnable tasks. You can use the **execute** method of class WorkQueue to add each Runnable task to the queue.
3. **You are not allowed to invoke shutdown() and awaitTermination() methods from class WorkQueue**. (Pretend that we might want to reuse the threads later for other tasks, and do not want to destroy them).

   For this part of the lab, you are required to implement a different mechanism for determining whether all the tasks have completed. **Maintain a variable numTasks in class HotelDataBuilder**, and increment it when you create a new FileWorker, and decrement it when a FileWorker is done with the task.

   Write a method `waitUntilFinished()` that *waits* if the number of tasks is greater than 0 (it means not all the tasks have been completed). Think of where you need to call notifyAll().

   Before printing hotel info to a file, call `waitUntilFinished()` to make sure you call `printToFile` only after all the tasks have been completed.

   Make sure you **synchronize access to numTasks** - it is **not** enough to make it volatile!
4. Make your code more efficient by having each `FileWorker` add reviews to their own **local ThreadSafeHotelData** (or HotelData), defined inside the run() method of the FileWorker class. Merge it to the "big" ThreadSafeHotelData only when the FileWorker is done parsing all the reviews. This should reduce the amount of blocking between the threads.
5. After you process all reviews for a particular hotel, compute and set the average rating for this hotel.
6. Use log4j2 `Logger` to write to a log file that contains debug messages about when each Runnable task has started, when it got completed, and when printing has occurred.

The starter code for this lab is your code for part 1 of lab 3. Create a different branch in your lab3 repo, called **part2.**

Your code should still pass all tests in:

- **HotelBuilderTest**
- **ReentrantReadWriteLockTest**

for this part of the lab.

Extra Credit (<=2 pt depending on difficulty - discuss with the instructor):
Concurrent search.
Create a file with multiple *search queries*: for instance, it might contain queries to search for all the users who rated a given hotel above or below a certain threshold, and queries that search for reviews that contain a certain keyword. For instance, the following sample file contains three such queries (find users who rated hotel #10323 with a rating above 4; find users who rated the hotel #1047 below 2, find all reviews of hotel #1146383 that contain the word "bedbugs"):

```
10323 Users r>4
1047 Users r<2
1146383 Reviews Contains bedbugs
```

Feel free to propose your own format of the input file with queries. Have multiple such queries in an input search file, process them **concurrently**, and output the results to the file (think of a reasonable format). You may **not** use java.util.concurrent package for extra credit. I recommend that you discuss your approach with the instructor before you start implementing it. You are responsible for writing your own tests for the extra credit part of the assignment. The professor will likely grade extra credit interactively.

Submission: Your lab3 part 2 should be submitted to the **part2** branch of your private lab3-username repo. Please keep pushing your code to github as you work on the lab. You need to have at least 3 meaningful commits to this branch on github, otherwise we will assign a 0 for the lab.
Several students from each section of the class will be asked to come in for a code review for this lab. Failure to answer questions about the code may result in a 0 for the assignment.