Module: Core Java

Session 17: Ad. Concepts of Threading Practice

* This is a practice session; you will work on advanced threading assignments.
* You can discuss your doubts with the trainer

**Assignments:**

**Assignment 1 – spawns n threads assignment**

Write a Java program that spawns n threads, where n is a program argument. These

threads access a shared counter (initialized as 0) in a loop. In each iteration, they read the counter to a local (stack) variable, increment it, and store it back to the counter. When all threads complete 10000 iterations each, the program stops and prints the value of the shared counter.

Note that the final value may be smaller than the total number of iterations. Run the

program on 1, 4, 8 and 16 threads and report the results in a textual table in the “doc.txt

file”.

In your report, include the run time in miliseconds. You can measure the run time by

calling the System.currentTimeMillis() before and after the execution.

The Java concurrency package, java.util.concurrent, is a library that provides synchronization tools for Java programs. This library includes a built-in mutual exclusion primitive called ReentrantLock.

You are required to write a Java program doing the same task as in part A but protect

the shared counter using java.util.concurrent.ReentrantLock, so no two threads modify the counter at the same time.

Run the program on 1, 4, 8 and 16 threads and report the results in a textual table in the

doc.txt file. In your report, include the run time in miliseconds.

**Assignment 2 – Thread pool assignment**

Part 1:

Search - Your search mechanism will take as input a multi-word query and will return a list of documents where the given words appear.

1. Your score for this portion of the assignment will be based on the intelligence of your algorithm. The easiest solution is to simply return a list of documents where any of the words appear. Another approach is to give greater weight to (by placing earlier in the result set) documents where more than one of the query words appears. Yet another option is to give greater weight to the documents where the words specified appear closer to one another. For example, if you search for "computer science" and document 1 has computer at position 1 and science at position 2 while document 2 has computer at position 1 and science at position 100, document 1 would be given higher weight.

2. Your program will take as input the directory as before along with a file containing a set of queries. Each line of the file will contain a multi-word query.

The output of your program will be a text file results.txt that contains the result of running each query.

You are required to:

3. Use Generics when appropriate.

4. Use the java.util data structures where appropriate.

5. Keep in mind that Strings are immutable.

Part 2

Thread Pool - You will use a thread pool or work queue and process up to 10 text files in parallel. As your program traverses the directory specified by the user, for each txt file found insert a new job into the queue.

1. To accomplish this, you will need to implement a work queue and a locking mechanism for your inverted index data structure.
2. Your locking mechanism will ensure that only 1 thread may change the inverted index at a time. Multiple threads may read data in the inverted index simultaneously