Assignment Set 1 (Concurrent programming)

- **Assignment will be evaluated by the TAs.**
- **You should submit report (for answering non-code related questions), complete source codes and executable files.**
- **All codes must be properly documented and good code writing practice should be followed (carry marks).**
- **Copying is strictly prohibited. Any case of copying will automatically result in F for the whole course, irrespective of your performance in the other parts of the lab.**

- **Submission deadline: 10th September, 2017**

- **Total weight = 20%**
- **Marks distribution: 40, 60**

Problem 1: Multiple File Search Utility

Suppose we have N text files. We want to have a utility to search for a phrase from the N different files. All the searched phrases, with the name of the file(s) where it has been found, should be stored in another file (let's call it "OUTPUT.txt"). If the searched phrase is found in multiple files, the output file should be written with the following principle.

Suppose you have searched for "Programming Languages" and it is found in two files. In both the files, "Programming" has been found as the 1st word. However, in the first file the word "Languages" has been found as the 17th word whereas in the 2nd file the same has been found as the 9th word. In the output file, file 2 will be written first, then file 1 (sorting will be done based on the *relative positions* of the matched phrase, closest match will be written first). The OUTPUT.txt fill will then look like,

> "Programming Languages" is found in
>
> file2.txt: "Programming" is found as 1st word, "Languages" is found as 9th word
>
> file1.txt: "Programming" is found as 1st word, "Languages" is found as 17th word

Note:

- All the words in a searched phrases must be found; otherwise the file name should not be included in the OUTPUT.txt.
- Relative sequence of the words in the searched phrase is important. (e.g., if "Programming" is found at location x but "Languages" is found in location y < x, the file will not be considered. "Languages" should be found at location > x).

- **Relative position** is to be calculated as follows: If "Programming" is found at location 'm' and "Languages" is found at location "n" in a particular file, the relative position is "n-m". For instance, relative position for the searched phrase "Programming Languages" in file1.txt is (17-1=16) whereas the same for file2.txt is (9-1=8). Sorting (in ascending order) of the name of the file to be written in OUTPUT.txt is to be done based on these relative positions.
- If a user search "Computer Science" after searching the "Programming Languages" and it is found that the word "Computer" is present as the 2nd word in file1.txt, 11th word in file5.txt, 12th word in file6.txt, 1st word in file9.txt and 20th word in file 10; whereas the word "Science" is present as the 5th word in file1.txt, 5th word in file file6.txt, 13th word in file9.txt and 21st word in file10.txt. Then the contents of the OUTPUT.txt will be as follows.

"Programming Languages" is found in

file2.txt: "Programming" is found as 1st word, "Languages" is found as 9th word

file1.txt: "Programming" is found as 1st word, "Languages" is found as 17th word

"Computer Science" is found in

file10.txt: "Computer" is found as 20th word, "Science" is found as 21st word

file1.txt: "Computer" is found as 2nd word, "Science" is found as 5th word

file9.txt: "Computer" is found as 1st word, "Science" is found as 13th word

- If the number of words present in the searched word is more than two then the relative position should be calculated by adding up pairwise relative positions. For example, if a user searched for the "Programming Languages Lab" and the word "Programming" is found as the 4th word, "Languages" as the 8th word and "Lab" as the 10th word in fileX.txt, the relative position of the searched phrase is (8-4)+(10-8)=4+2=6.
- If the searched phrase is found at multiple places within a single file, the least relative position will be considered. In case of a tie (multiple occurrence with the same relative position), first one should be considered.

Implement the proposed solution using concurrency constructs in JAVA. You should show two implementations. In the first, you use the older Java constructs (Thread/Runnable & Synchronization keyword). In the second, you should use constructs from the Java.util.concurrent package. For sorting, you may use either of the "quick sort" and "merge sort" algorithm. **You must have to use the 'fork-join' principle in creating the sorted list.**

Problem 2: Classroom visualization

In this problem, we wish to develop a system to visualize how the students are seating in a classroom. In order to do that, we are assuming the following.

1. There is a pre-stored database of students' images with the roll number serving as the image file name (like 166101010.jpg). The database is created before the start of a course.

At the beginning of a lecture, students are required to provide their details (roll no, name and seat number). The students provide their details using their own devices (laptop/smartphone). You have to do the following.

1) Develop a login system in java swing to collect students' name, roll number and seat number. The system collects the students' data and sends it to the server connected through Wi-Fi.
2) When more than one student open their devices and send requests to server for storing students' data at the same time, the server might be overloaded. As a result, there might be significant delay in serving the requests. To solve this problem, use the concepts of concurrency, thread and synchronization in java.

Once the students' data (name, roll number, seat number) are available, you are required to display a layout of the classroom seating arrangement on the screen (let's call it "teacher's display"). In the layout, the pre-stored image of a student should be displayed at the seat where s/he is seating.

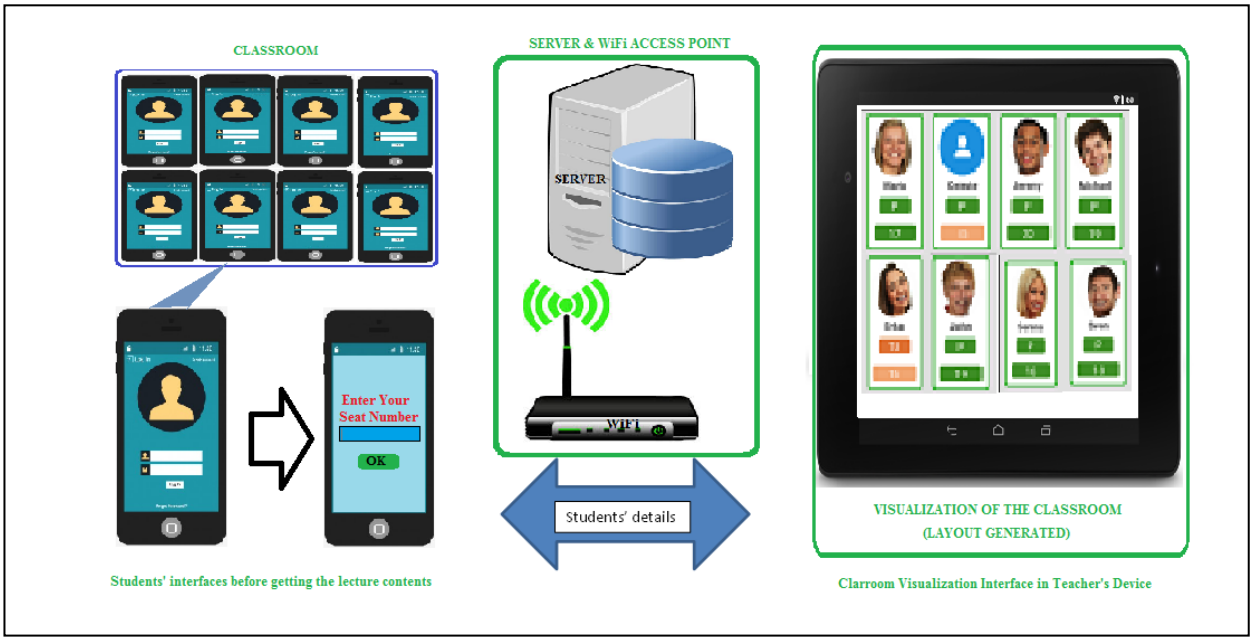

Fig: Illustration of the system

Try the above problem with various strategies:

I. Layout generation without using thread(s).
II. Layout generation with thread(s) but without using the fork-join framework.
III. Layout generation with thread(s) and using the fork-join framework.

Justify the number of threads creation in your program. Also report your observational differences among the above three strategies (if any).