**Write-up on the Methodology**

The application is a Spring-Boot Application.

**Requirement:** Java 1.8+ (As Java streams, which is included from 1.8 version is used)

Jar file to be executed in the following format:

***Java –jar LogExtractor.jar <from\_time> <to\_time> <path\_of\_logs>***

1. Date strictly expected in the format of *"yyyy-MM-dd'T'HH:mm:ss.SSSS'Z'"* as specified in the problem statement
2. Logs file present in the local system. (Not cloud services like AWS S3)

Ex*: java –jar LogExtractor.jar 2020-01-31T20:12:40.1234Z 2020-02-02T12:12:38.1234Z C:\Users\{path}*

**Assumptions**:

1. Log files are present in local system.
2. Logs in each file are in sequential order i.e. ***logs in ascending order based on time***

**Aspects covered in the code / Justification:**

1. Given a path, only those files are considered with file names in the format of ***LogFile\*.log***
2. Leveraged the concept of parallel stream so as to enable the use of CPU cores for fast processing – Number of threads are configured to be as the number of available processors (avoids the overhead in switching threads)
3. Filter the various log files in parallel – eliminating the files which doesn’t contain the logs of the required period.
4. Log lines will not be stored in memory.
5. Only the first line and last line of each file are considered for filtering. (Filter based on the log file starting and ending timestamps and comparing with the required period) – Using RandomAccessFile, without traversing through the file, we directly read the last line
6. This greatly reduces the number of files to be completely read in the next step – a huge performance benefit
7. Only those files which doesn’t get filtered out in the previous step are considered and read sequentially.
8. It is assured that required time period logs will be present in the file.
9. Reads the log file sequentially and prints all the logs that fall into required period by comparing timestamps.
10. Once the timestamp of the current log line goes beyond the “*to\_time*” i.e. beyond our needed time, we do not read the next lines anymore and close the Reader. (As it is assumed logs will be in ascending order based on time) - avoid reading of complete file unless necessary
11. Time taken for the extraction is printed in the last line.

**Output:**

1. Logs will not be printed in time-sorted way if the logs of given period are present in multiple files, since, we read the files in parallel.
2. If we want to avoid this, we may convert List of files into *“stream”* instead of doing it into “*parallelStream*”.