Submission Requirements

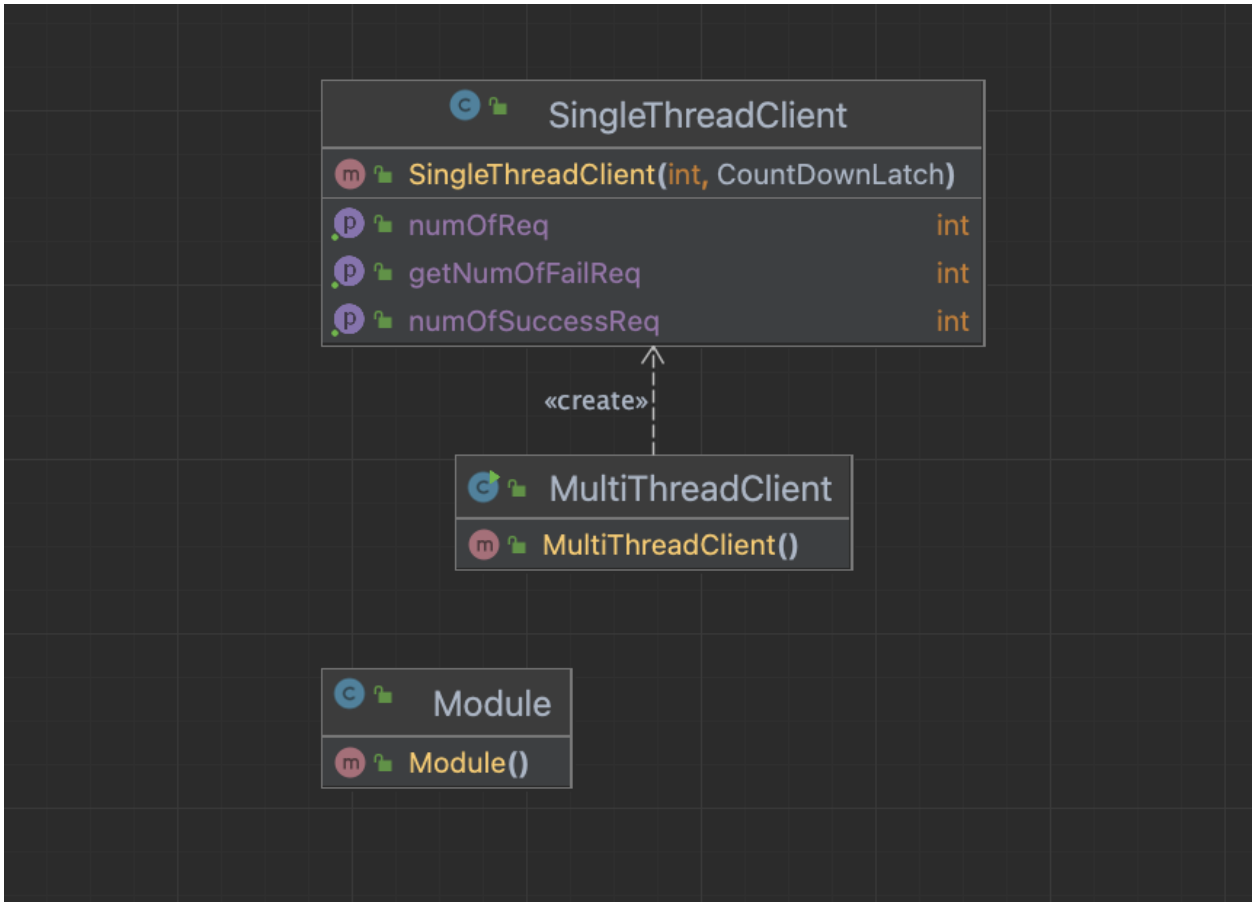
Submit your work to Canvas Assignment 1 as a pdf document. The document should contain:

1. the URL for your git repo. *Make sure that the code for the client part 1 and part 2 are in seperate folders in your repo*
2. a 1-2 page description of your client design. Include major classes, packages, relationships, whatever you need to convey concisely how your client works.
3. Client (Part 1) - This should be a screen shot of your output window with your wall time and throughput. Also make sure you include the client configuration in terms of number of threads used and Little's Law throughput predictions.
4. Client (Part 2) - run the client as per Part 1, showing the output window for each run with the specified performance statistics listed at the end.
5. The plot of your throughput over time.

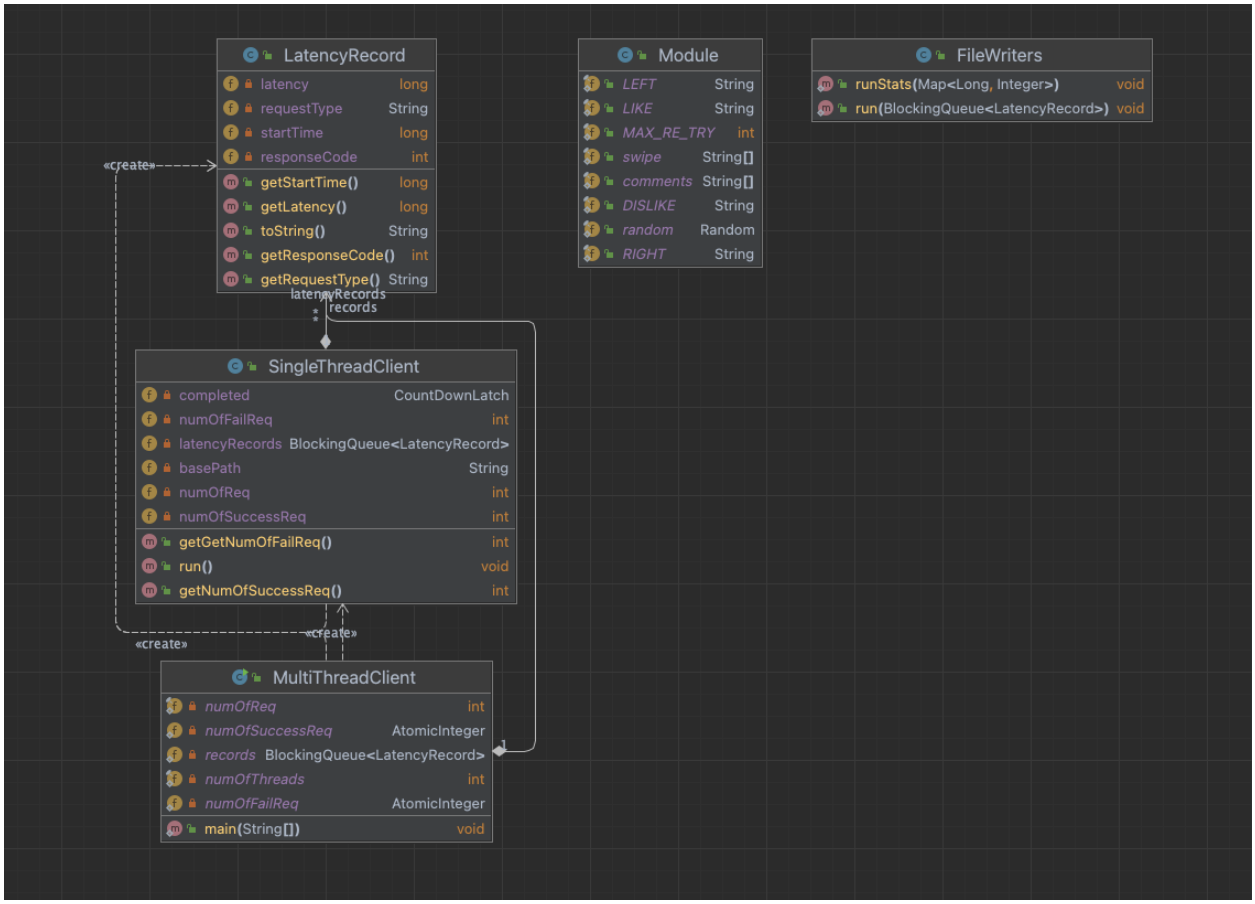
Submission details:

1. Git repo URL:
   https://github.com/Oliver1024/DistributedSystemDatingApp/tree/main/assignment1

2. Client design description:
   Part -1 uml:


SingleThreadClient
SingleThreadClient(int, CountDownLatch)
numOfReq
int
getNumOfFailReq
int
numOfSuccessReq
int
«create»
MultiThreadClient
MultiThreadClient()
Module
Module()

Part -2 uml:



The client part which I have is implemented multi-threading client-server architecture to simulate the client sending requests to the server. This client part2 contains 5 Java classes: FileWriters, LatencyRecord, Module, and MultiThreadClient and SingleThreadClient. I also have the same logic as in client part1.

SingleThreadClient class acts as a single client, which is executed by multiple threads. Each thread executes the run method of SingleThreadClient class and sends the number of requests specified in the numOfReq variable and the run method of the SingleThreadClient class sends the request to the server, records the latency of each request, and calculates the success and failure counts of the requests. This class creates an instance of a SwipeApi client using the Swagger API client library and sends a specified number of "swipe" requests to the API at the specified base URL. Each swipe request is represented by a "SwipeDetails" object, which includes the swiper ID, swipee ID, and a comment, all of which are randomly generated. If a swipe request fails, the code retries it a specified number of times before moving on to the next request.

FileWriters is a class that writes data to two separate CSV files. The first file records latency information, including "StartTime", "RequestType", "Latency", and "ResponseCode". The second file records performance statistics, including "Seconds" and "Throughput/second".

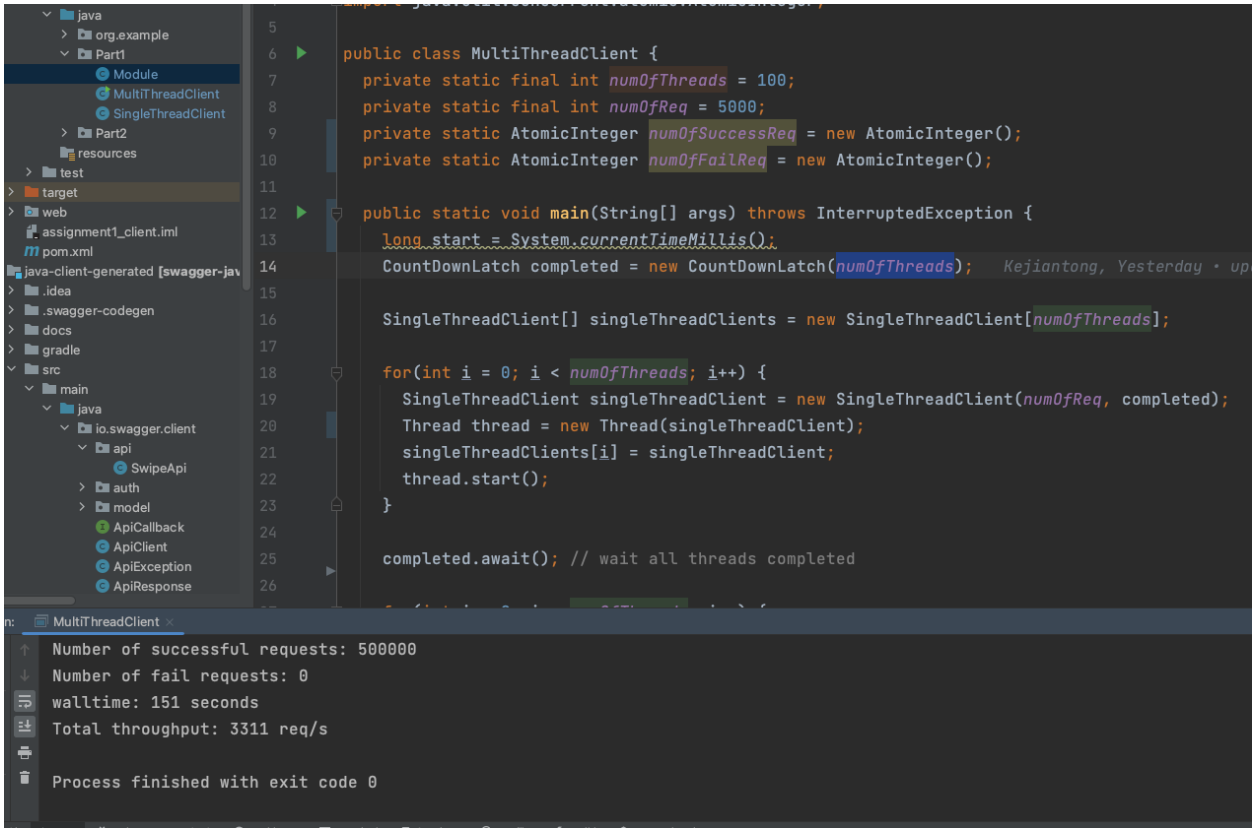
LatencyRecord is a simple data class that holds information about a latency record, including the start time, request type, latency, and response code.

Module is a class with several constants and a random number generator. The constants define swipe directions (LEFT, RIGHT), comments (LIKE, DISLIKE), and the maximum number of retries (MAX_RE_TRY).

MultiThreadClient is the main class that creates and runs multiple SingleThreadClient threads. Each SingleThreadClient makes multiple requests and records latency information and performance statistics. The MultiThreadClient class waits for all SingleThreadClient threads to complete and calculates the overall success rate of the requests and the wall-clock time. The final results are stored in two CSV files by the FileWriters class.

3. Client - Part 1:

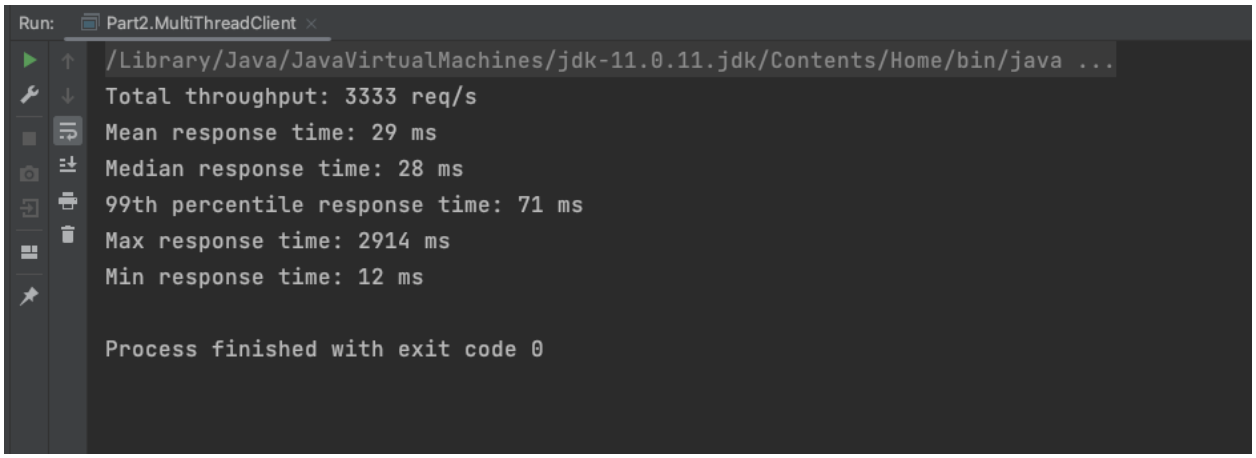
Configure for client:
Number of threads: 100
Number of request per thread: 5000
Total requests: 500k
Thorughput: 3311 req/s



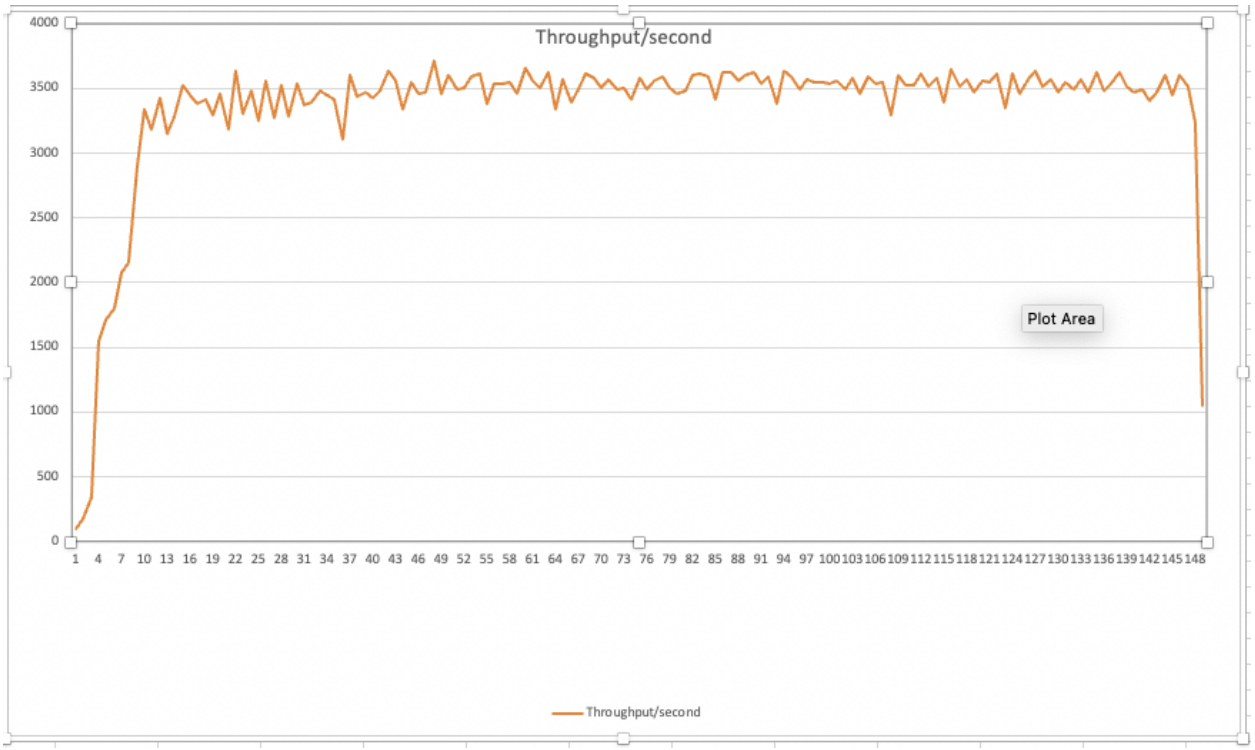
First, I send 10000 requests using 1 thread, and get the response time(w) = 18.6ms
Then, apply to little's law,
Expected Little's Law throughput = 100/18.6 = 5376 req/s

I also have tried many times using other configuration to send 500k requests. Like I have tested 150 threads and 200 threads, but they don't give me a very larger throughput.

4. Client - Part2:



Task 4:


Throughput/second
4000
3500
3000
2500
2000
1500
1000
500
0
1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49 52 55 58 61 64 67 70 73 76 79 82 85 88 91 94 97 100 103 106 109 112 115 118 121 124 127 130 133 136 139 142 145 148
Plot Area
Throughput/second