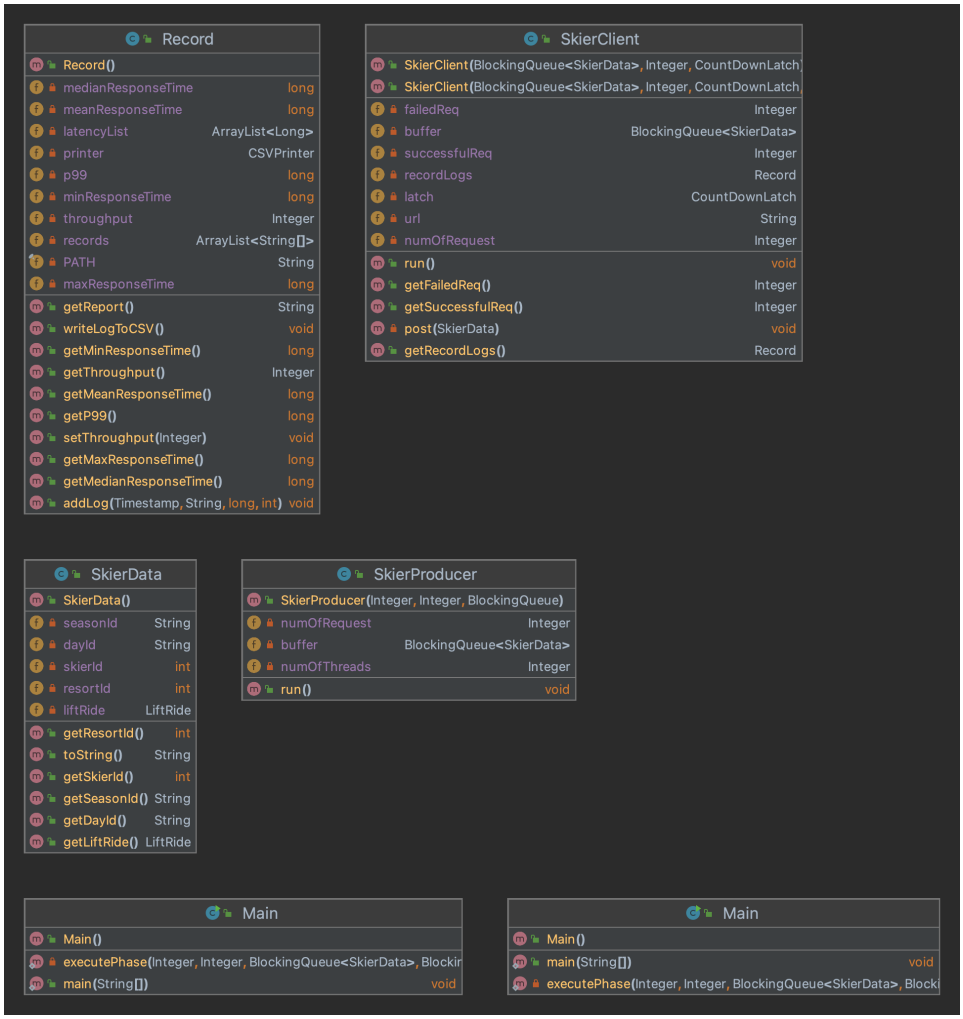
Github URL: https://github.com/jasongautama/cs6650-Distributed-Systems/tree/main/assignment1

Class Design



SkierClient

SkierClient is the consumer that implements Runnable that executes the POST request that is added by SkierProducer to the buffer.

SkierData

SkierData is a class that creates random data (SkierId, resortId, etc) and creates the JSON body for POST request by calling the *.toString()* method

SkierProducer

SkierProducer is a class that puts SkierData into the buffer given the amount of request to be consumed by SkierClient. In this case, SkierProducer produced 200,000 SkierData.

Main - Part 1

executePhase(Integer numOfThreads, Integer requestPerThread, BlockingQueue<SkierData> buffer, BlockingQueue<SkierClient> resultBuffer)

- Create new Thread for Producer to produce 200k SkierData → BlockingQueue<SkierData> buffer
- Created ExecutorService for ThreadPool given the amount of N threads
- Add SkierClient to ThreadPool and run the thread
    - SkierClient consume SkierData from buffer (*buffer.take()*)
- Once completed, add SkierClient to BlockingQueue (resultBuffer) for the statistics at the end of Main
- Display statistics

Main - Part 2

executePhase(Integer numOfThreads, Integer requestPerThread, BlockingQueue<SkierData> buffer, BlockingQueue<SkierClient> resultBuffer, Record logs)

- The flow is the same as #Main - Part 1
- In addition, takes in Record logs that is use to calculate the needed statistics for Part 2 (refer to Record class below)

Record

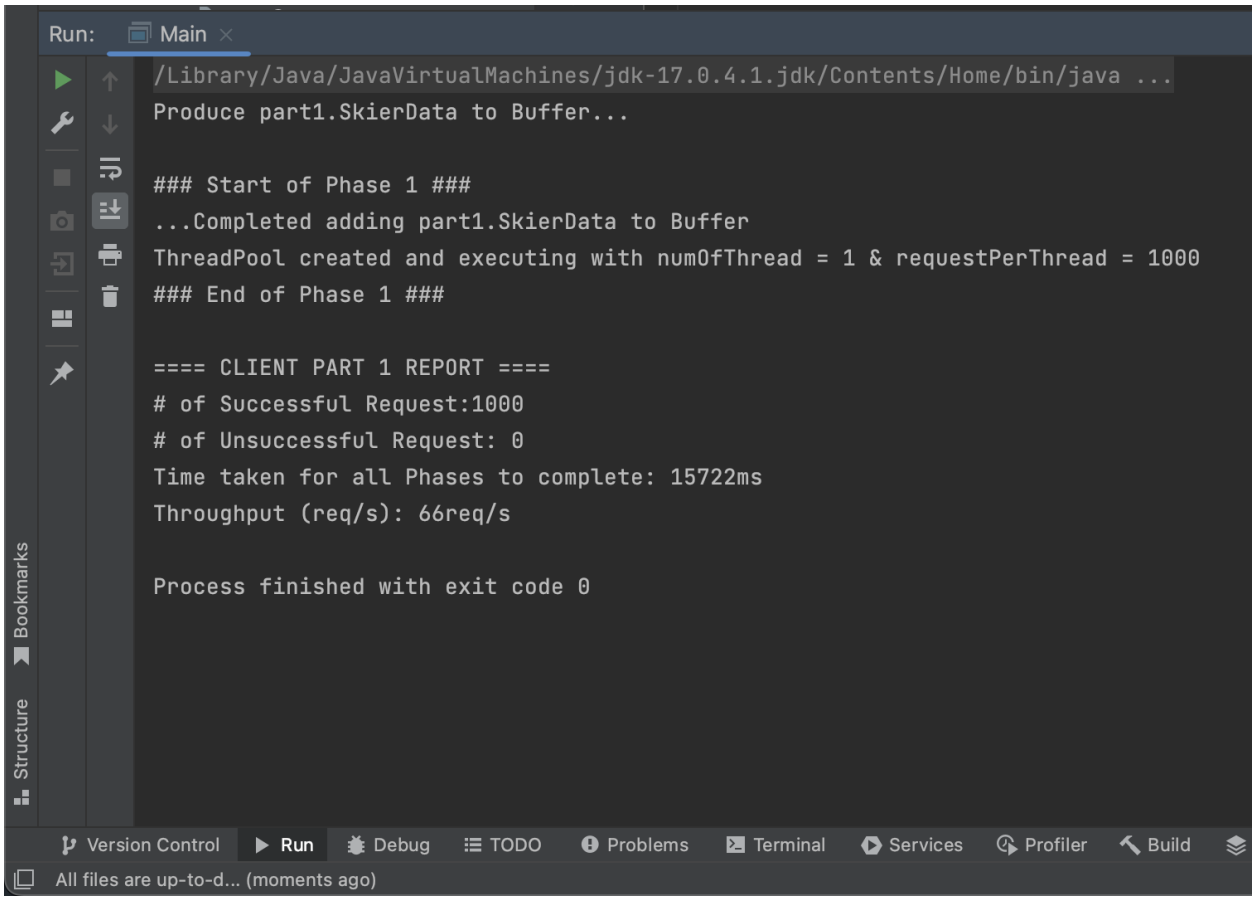
The purpose of Record class is to store log data that consist of start time, request type, latency, and status code for each POST request by calling the *.addLog(...)*. Once all the requests are completed, call the *writeLogToCSV( )* to generate the log file in .csv format.

Finally, part2/Main.java program will call the *getReport()* to print the statistics (mean, p99, throughput, etc) to the console.

Test Scenario

Attempt #1 - Testing throughput

Sending 1,000 requests using 1 thread to EC2 instance located in us-west-2 (Oregon)



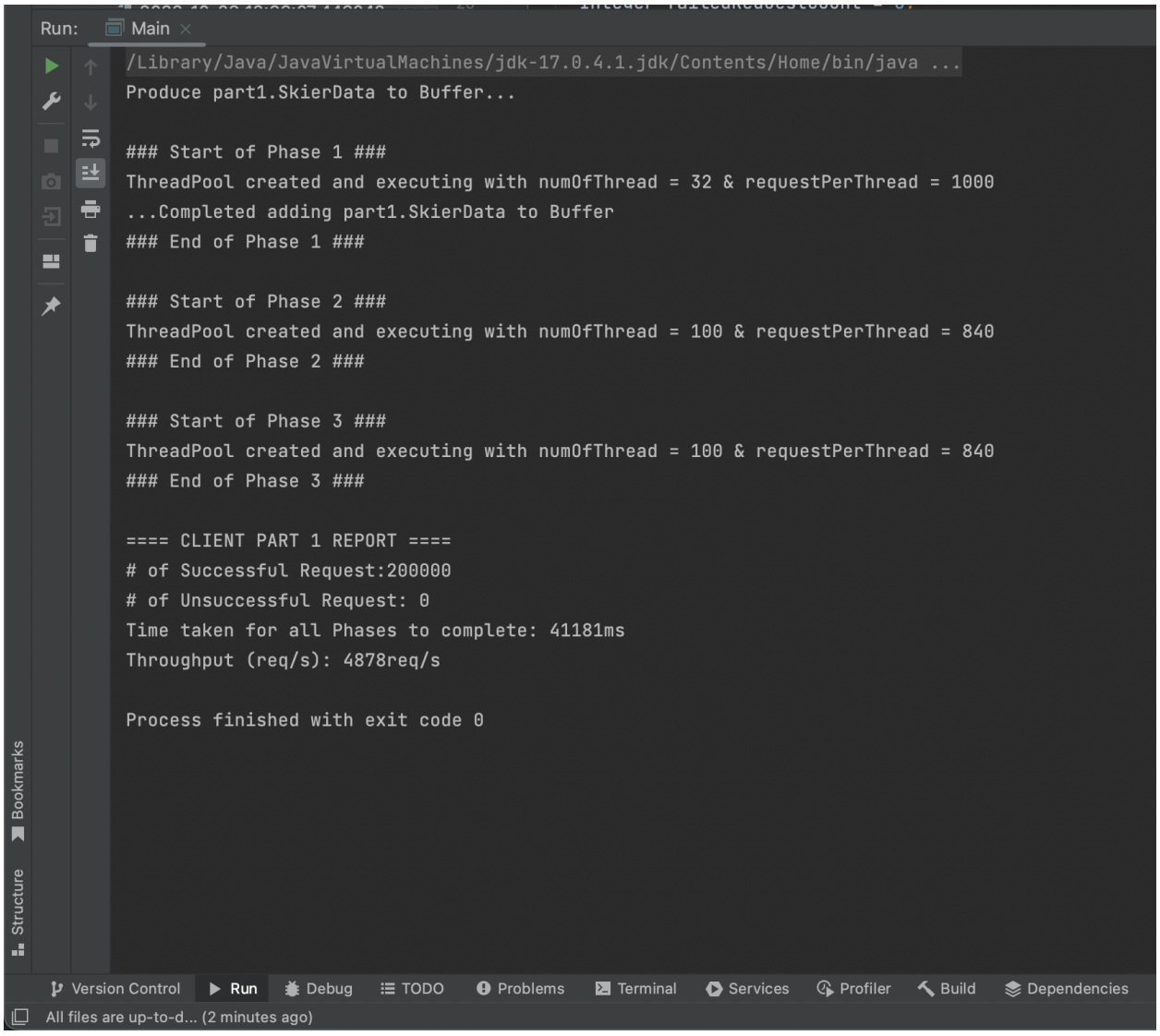
Little's Law Prediction

L = λ * W
L = Level of WIP (Min. Qty of items in the system at any moment of time)
λ = Throughput Rate (Qty of items going through the system per period of time)
W = Throughput Time (Avg time an item spends inside the system per period of time)

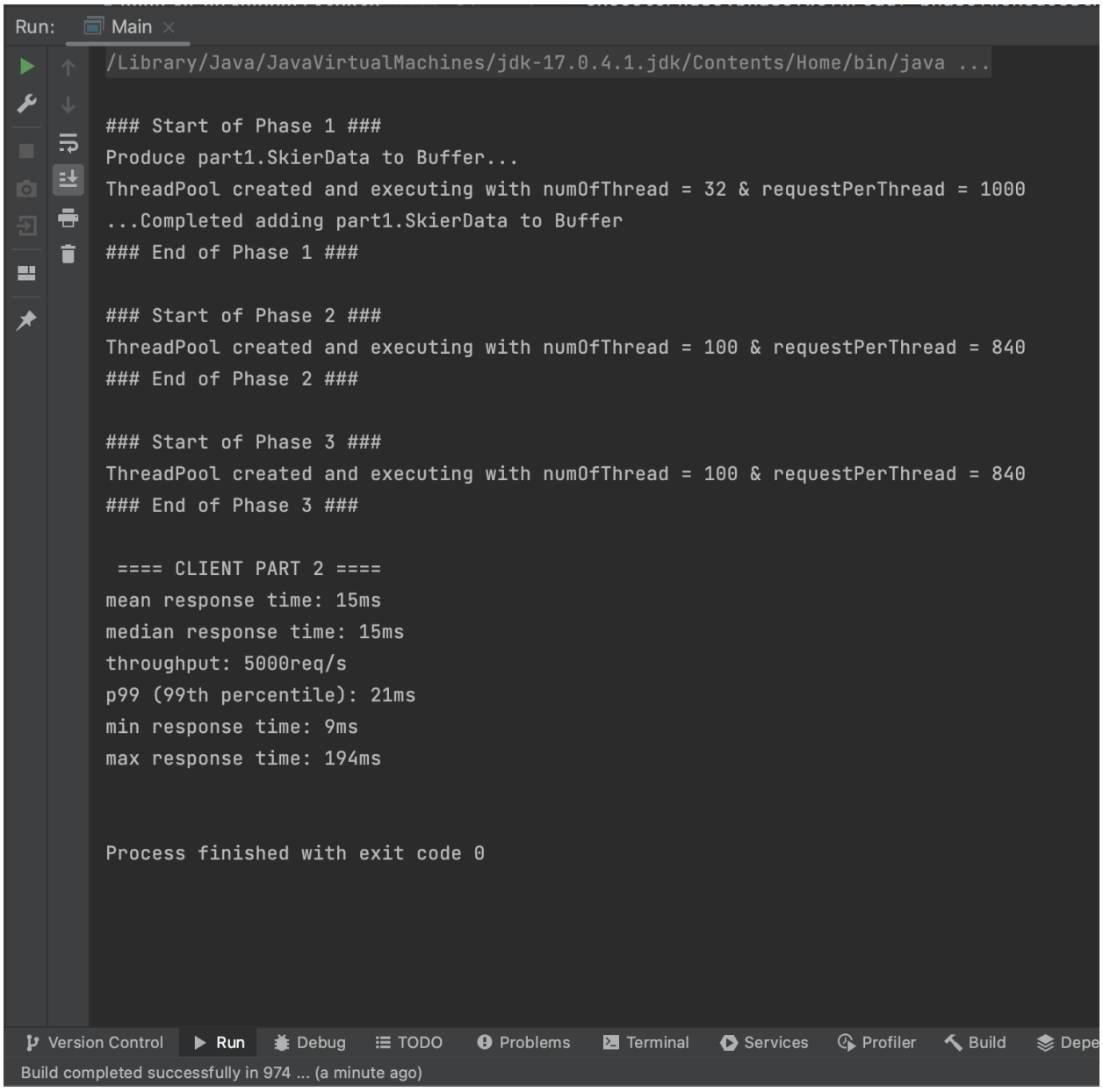
L = 1000req
W = 15.722s
λ = L / W
= 1000 req / 15.722s
= 63.605 req/s (Throughput time)

Using the Little's Law formula, we can see that the prediction (*63.605req/s*) is aligned with the actual throughput (*66 req/s*) ~ or actually higher throughput than the prediction.

Client (Part 1) output



Client (Part 2) output



Throughput comparison (2.44% difference)
Client 1: 4,878 req/s
Client 2: 5,000 req/s