Assignment 1
Yiming Ge

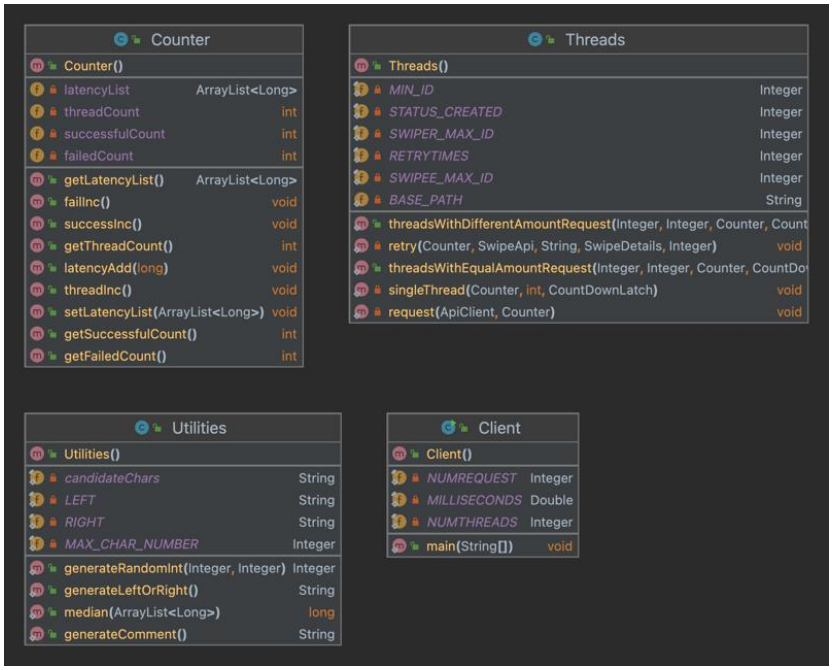
Client Design:
Client Part One:
In part one client, it has four classes: Client, Threads, Utilities and Counter.

Client Class is a main class which you can directly run to start the program with given requests and threads.

Threads Class contains two different thread function: When the number of requests can be dived by the number of threads evenly without any reminders, threadsWithEqualAmountRequests will be used. Otherwise, threadsWithDifferentAmountRequest will be used. Here is an example:
(threadsWithEqualAmountRequests):
8000 requests, 40 threads, each thread will handle 200 requests.
(threadsWithDifferentAmountRequests):
8000 request. 30 threads, each thread will handle 266 requests and the last thread will handle 20 more requests. 266 * 30 + 20 = 8000.

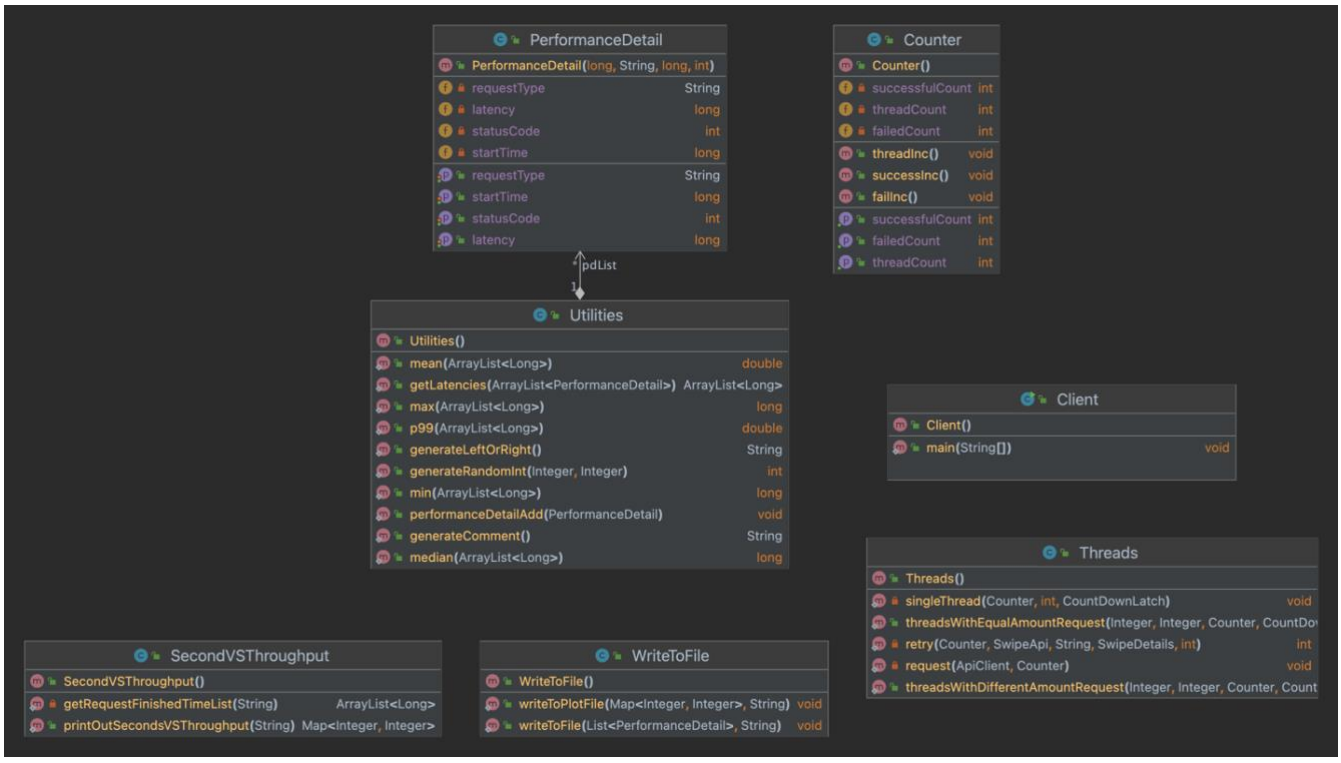
Counter class is the class to record fail and success request counts

Utilities class contain all the functions to generate random stuffs like swipe direction, comments, IDs.



Client Part Two:

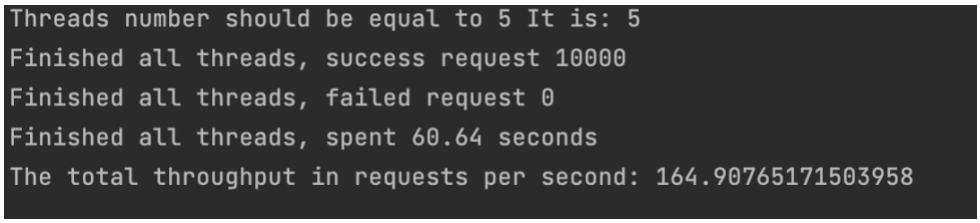
In part two client, it has 7 classes:
Client, Threads, Utilities and Counter are similar with part1.
PerformanceDetail SecondVSThroughput and WriteToFile are three additional classes.
PerformanceDetail class is to record each request information including request type, latency, status code and start time.
WriteToFile class contains all the functions to write array list stored info and map stored info into CSV file.
SecondVSThroughput class is a class to calculate the requests per second during the whole running time. Store time vs requests for task4 diagram.
x-axis values: unit is seconds, from 0 to test wall time, with intervals of one second
y-axis values: unit is throughput/second, showing the number of requests completed in each second of the test.



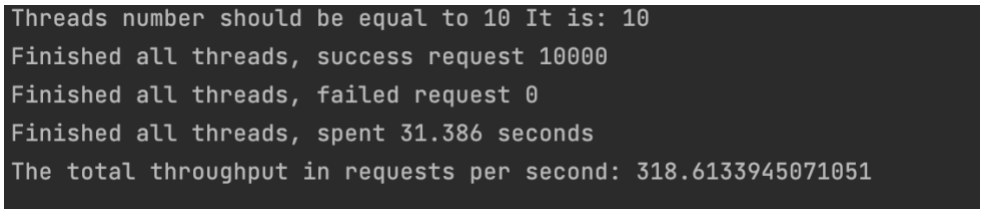
Client Part1 results:

10000 requests
1 thread:

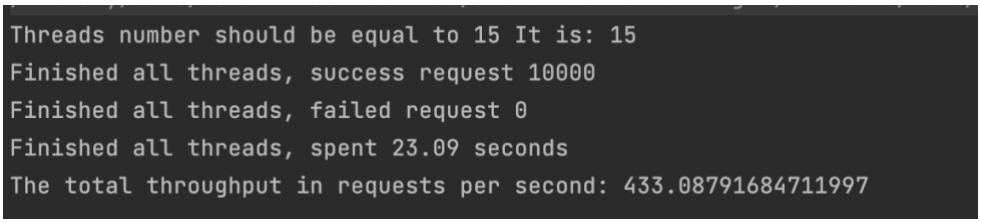
5 threads:



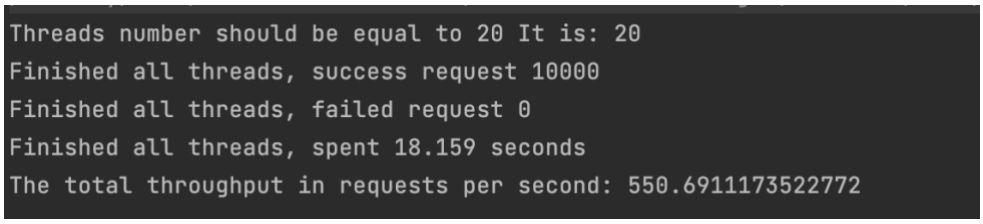
10 threads:



15 threads:

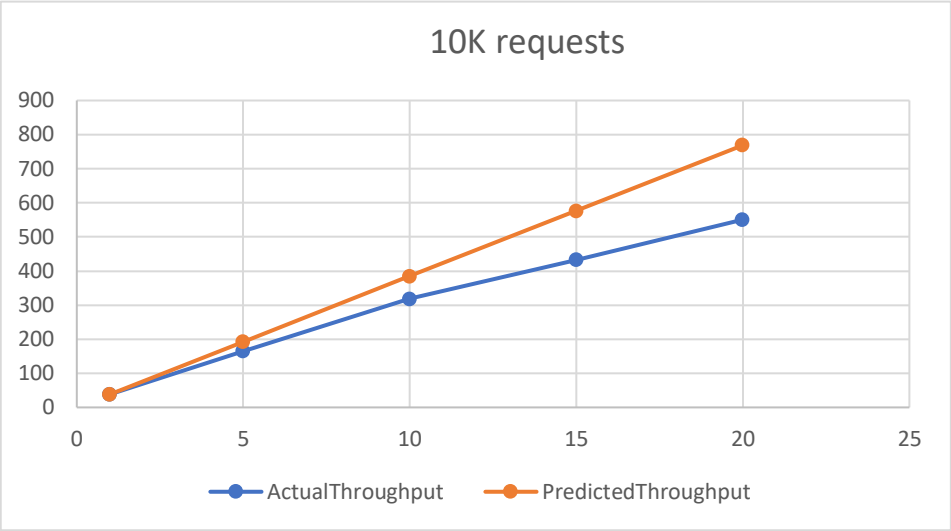


20 threads:



Threads	ActualThroughput	PredictedThroughput

1	38.44	38.44
5	164.91	192.2
10	318.61	384.4
15	433.08	576.6
20	550.69	768.8



500K request

75 threads:

```
Threads number should be equal to 75 It is: 75
Finished all threads, success request 500000
Finished all threads, failed request 0
Finished all threads, spent 231.839 seconds
The total throughput in requests per second: 2156.6690677582287
```

100 threads:

```
Threads number should be equal to 100 It is: 100
Finished all threads, success request 500000
Finished all threads, failed request 0
Finished all threads, spent 182.514 seconds
The total throughput in requests per second: 2739.5158727549665
```

125 threads:

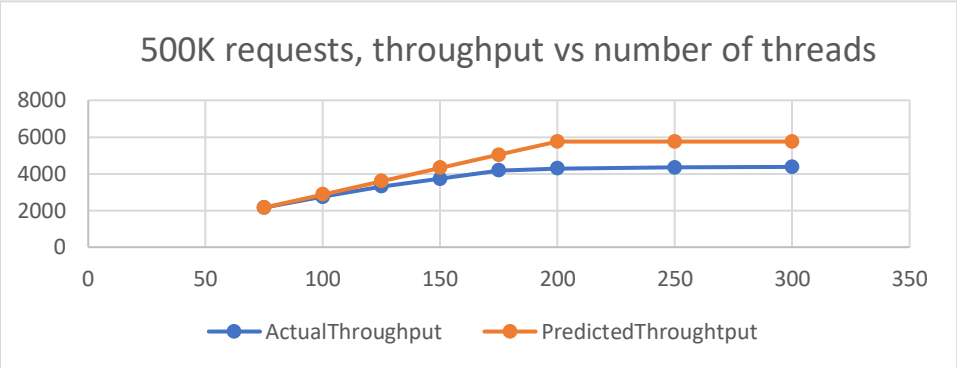
```
Threads number should be equal to 125 It is: 125
Finished all threads, success request 500000
Finished all threads, failed request 0
Finished all threads, spent 151.604 seconds
The total throughput in requests per second: 3298.066014089338
```

150 threads:

```
Threads number should be equal to 150 It is: 150
Finished all threads, success request 500000
Finished all threads, failed request 0
Finished all threads, spent 134.072 seconds
The total throughput in requests per second: 3729.339459394952

Process finished with exit code 0
```

Threads	ActualThroughput	PredictedThroughtput
75	2156.67	2156.67
100	2739.51	2875.56
125	3298.06	3594.45
150	3729.33	4313.34
175	4174.04	5032.23
200	4286.06	5751.12
250	4332.53	5751.12
300	4372.51	5751.12



As we can see, 500K diagram's trend looks very similar with 10k test diagram's trend at the level below 200 threads. Above 200 threads, the 200 maximum Tomcat thread is approached, thus the throughput almost does not change at all.
Although there's a small difference between actual throughput and predicted throughput, in the actual situation it still almost follows the little's law.

Client Part2 results:

The Performance records are saved in Performances.csv, The task4 plot data are saved in Plot.csv. These csv files are in Twinder/Clients/Client_Part2/ClientSwagger.

500K requests with 150 threads.

```
Threads number should be equal to 150 It is: 150
Finished all threads, success request 500000
Finished all threads, failed request 0
Finished all threads, spent 133.239 seconds
######Part2 Stats#######
The mean response time in milliseconds: 39.465918
The median response time in milliseconds: 35
The total throughput in requests per second: 3752.655003414916
The p99 response time in milliseconds: 114.0
The min response time in milliseconds: 14
The max response time in milliseconds: 2834
2 second: 150 requests
3 second: 304 requests
4 second: 1497 requests
5 second: 1886 requests
6 second: 1939 requests
7 second: 1740 requests
8 second: 2527 requests
9 second: 2882 requests
10 second: 3650 requests
11 second: 3483 requests
12 second: 3885 requests
13 second: 4099 requests
14 second: 3878 requests
15 second: 3959 requests
16 second: 4115 requests
17 second: 4045 requests
```

```
18 second: 4250 requests
19 second: 3662 requests
20 second: 3879 requests
21 second: 4108 requests
22 second: 4157 requests
23 second: 4227 requests
24 second: 3689 requests
25 second: 3812 requests
26 second: 4421 requests
27 second: 4011 requests
28 second: 3841 requests
29 second: 3233 requests
30 second: 3621 requests
31 second: 3908 requests
32 second: 4017 requests
33 second: 3771 requests
34 second: 4093 requests
35 second: 4047 requests
36 second: 4058 requests
37 second: 4142 requests
38 second: 4215 requests
39 second: 4124 requests
40 second: 3876 requests
41 second: 4154 requests
42 second: 3912 requests
43 second: 4095 requests
44 second: 3784 requests
45 second: 3841 requests
```

```
46 second: 3989 requests
47 second: 3487 requests
48 second: 2690 requests
49 second: 2486 requests
50 second: 1935 requests
51 second: 2650 requests
52 second: 3411 requests
53 second: 3690 requests
54 second: 3401 requests
55 second: 4020 requests
56 second: 4054 requests
57 second: 4082 requests
58 second: 3850 requests
59 second: 3822 requests
60 second: 4052 requests
61 second: 3724 requests
62 second: 3736 requests
63 second: 3776 requests
64 second: 4125 requests
65 second: 4028 requests
66 second: 3621 requests
67 second: 4232 requests
68 second: 3983 requests
69 second: 4199 requests
70 second: 3888 requests
71 second: 4075 requests
72 second: 4244 requests
73 second: 4173 requests
```

```
74 second: 4095 requests
75 second: 4382 requests
76 second: 4039 requests
77 second: 4212 requests
78 second: 3823 requests
79 second: 3970 requests
80 second: 4157 requests
81 second: 3795 requests
82 second: 4328 requests
83 second: 3778 requests
84 second: 4113 requests
85 second: 4421 requests
86 second: 3919 requests
87 second: 4274 requests
88 second: 4463 requests
89 second: 3939 requests
90 second: 4238 requests
91 second: 4273 requests
92 second: 4374 requests
93 second: 4246 requests
94 second: 4291 requests
95 second: 4110 requests
96 second: 4248 requests
97 second: 4074 requests
98 second: 4338 requests
99 second: 4252 requests
100 second: 4284 requests
101 second: 4350 requests
```

```
102 second: 3963 requests
103 second: 3933 requests
104 second: 4220 requests
105 second: 4301 requests
106 second: 4267 requests
107 second: 3948 requests
108 second: 3891 requests
109 second: 4317 requests
110 second: 4292 requests
111 second: 4291 requests
112 second: 4180 requests
113 second: 4167 requests
114 second: 4248 requests
115 second: 4201 requests
116 second: 4125 requests
117 second: 4388 requests
118 second: 4403 requests
119 second: 4120 requests
120 second: 4450 requests
121 second: 4010 requests
122 second: 4328 requests
123 second: 4299 requests
124 second: 4193 requests
125 second: 4267 requests
126 second: 4078 requests
127 second: 4229 requests
128 second: 4225 requests
129 second: 4178 requests
```

```
129 second: 4178 requests
130 second: 3728 requests
131 second: 3054 requests
132 second: 678 requests

Process finished with exit code 0
```

500k requests with 125 threads:

```
Threads number should be equal to 125 It is: 125
Finished all threads, success request 500000
Finished all threads, failed request 0
Finished all threads, spent 155.066 seconds
######Part2 Stats#######
The mean response time in milliseconds: 38.269114
The median response time in milliseconds: 34
The total throughput in requests per second: 3224.433467039841
The p99 response time in milliseconds: 113.0
The min response time in milliseconds: 14
The max response time in milliseconds: 2699
```

500k requests with 100 threads:

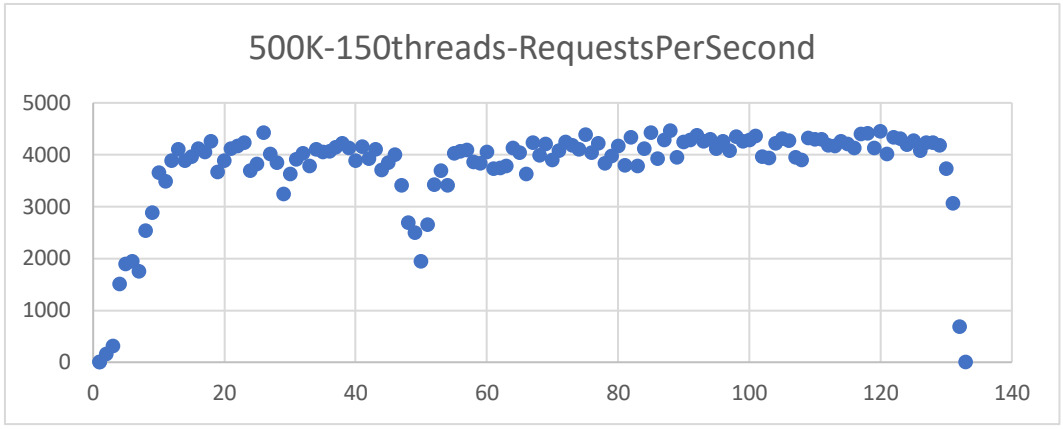
```
Threads number should be equal to 100 It is: 100
Finished all threads, success request 500000
Finished all threads, failed request 0
Finished all threads, spent 184.712 seconds
######Part2 Stats#######
The mean response time in milliseconds: 36.636978
The median response time in milliseconds: 34
The total throughput in requests per second: 2706.9167135865564
The p99 response time in milliseconds: 103.0
The min response time in milliseconds: 14
The max response time in milliseconds: 2021
```

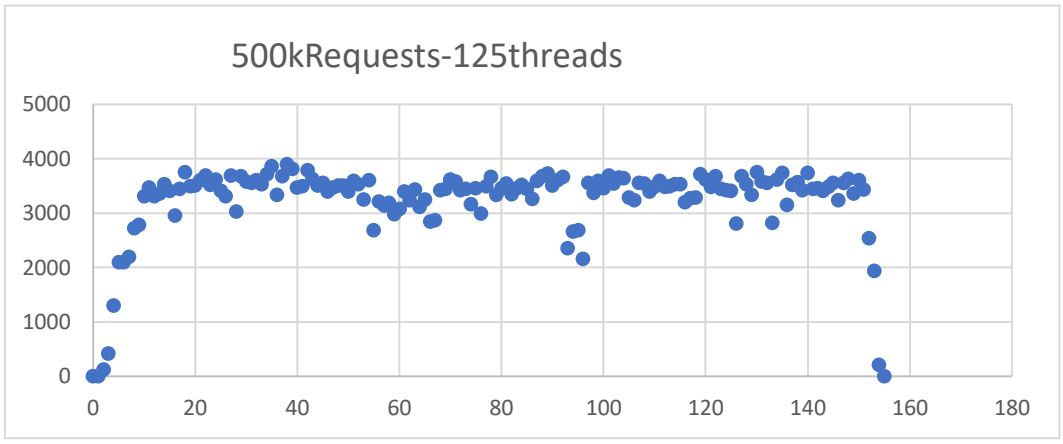
500k requests with 75 threads:

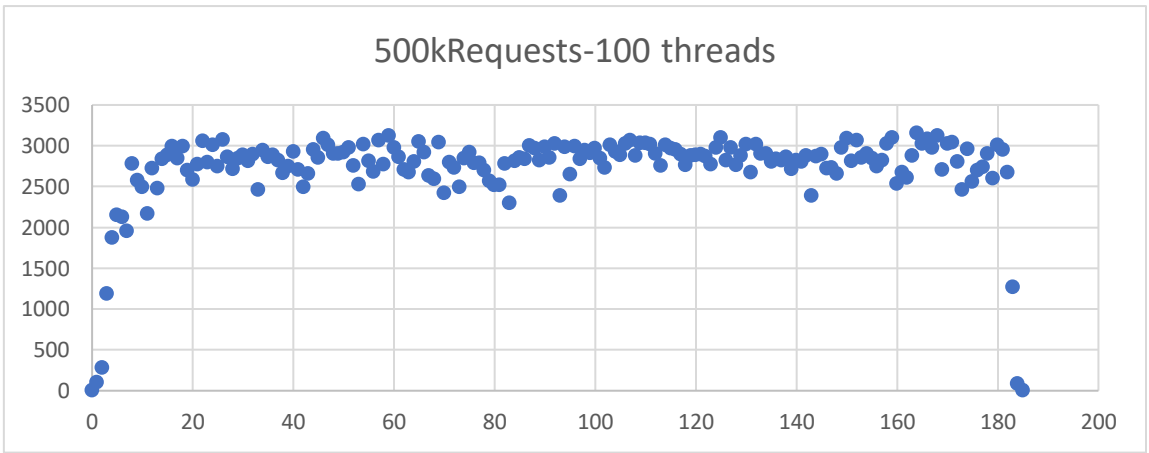
```
Threads number should be equal to 75 It is: 75
Finished all threads, success request 500000
Finished all threads, failed request 0
Finished all threads, spent 241.676 seconds
######Part2 Stats#######
The mean response time in milliseconds: 35.836648
The median response time in milliseconds: 34
The total throughput in requests per second: 2068.885615452093
The p99 response time in milliseconds: 89.0
The min response time in milliseconds: 14
The max response time in milliseconds: 1493
```

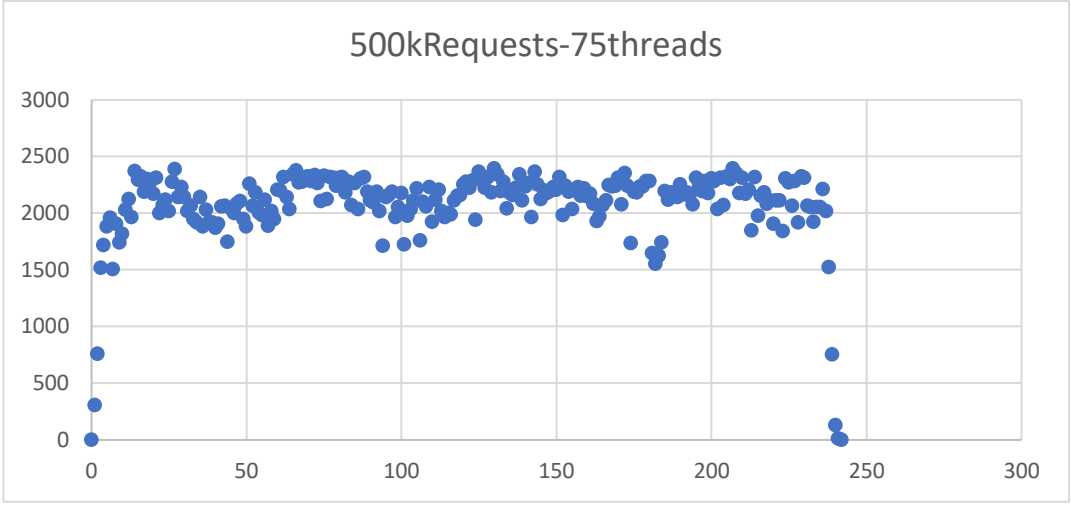
All of part2 clients using time are within 5% of Client Part1.

Draw the requests vs seconds plot.









Bonus:
Theoretically, the Spring Boot can potentially be faster and more efficient than a traditional servlet-based approach for a multithreaded client. This is because Spring Boot provides a number of performance optimizations, such as asynchronous processing and connection pooling, that can help to improve the speed and efficiency of your application when handling multiple concurrent requests.

Spring Boot server
500K request 75 threads:

```
Threads number should be equal to 75 It is: 75
Finished all threads, success request 500000
Finished all threads, failed request 0
Finished all threads, spent 235.202 seconds
The total throughput in requests per second: 2125.832263331094

Process finished with exit code 0
```

500K request 150 threads:

```
Threads number should be equal to 150 It is: 150
Finished all threads, success request 500000
Finished all threads, failed request 0
Finished all threads, spent 135.091 seconds
The total throughput in requests per second: 3701.208814798913

Process finished with exit code 0
```

However, the actual throughput and spent seconds for spring boot server are similar with http servlet. The actual difference in speed and efficiency is very small.
I think it is because we implemented a simple application (server) and it only needs to handle a few number of concurrent requests.