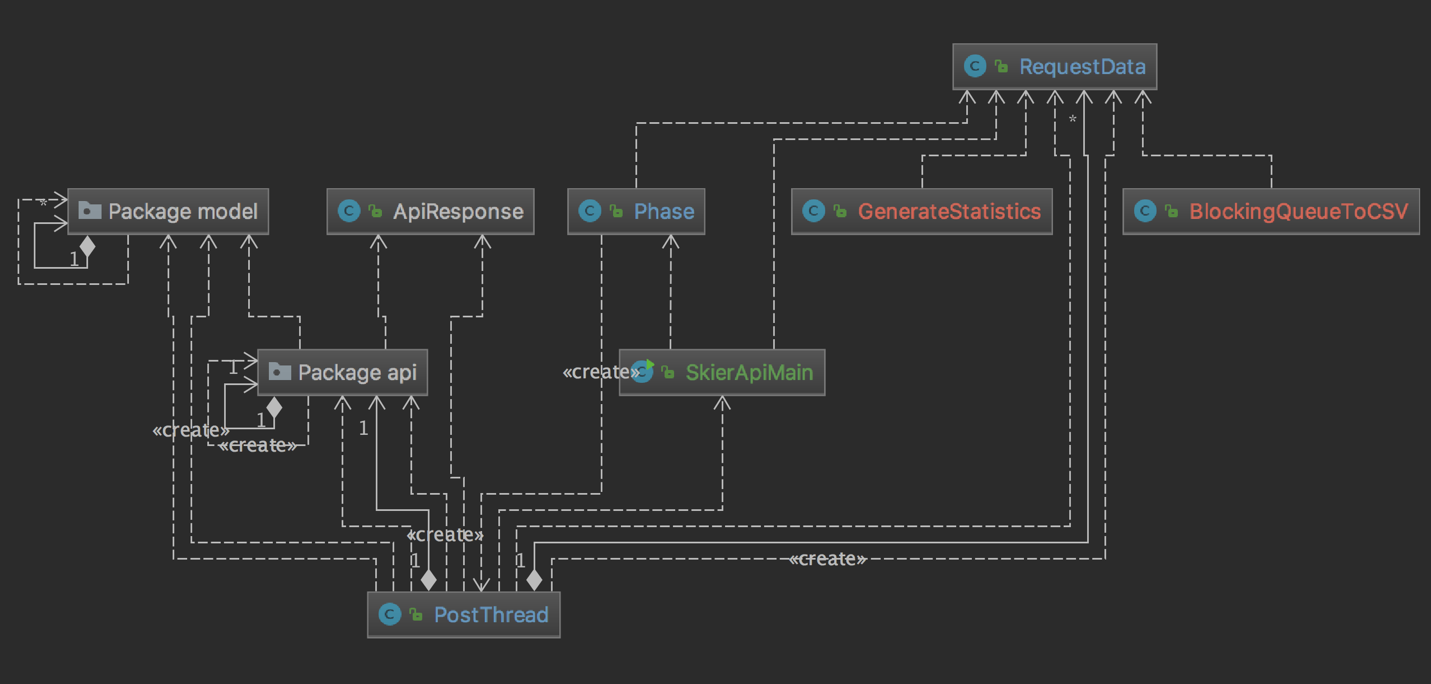
**CS6650 Assignment 1**

Yixing Shao

10/09/2019

**1. URL for Git Repo:**

**2. Client Design (Include major classes, packages, relationships, whatever you need to convey concisely how your client works)**



My client is based of the swagger server APIs with some add-ons. The main method is invoked within *SkierApiMain* class. In this class, all static variables are defined and all phases are called within this class. Countdown latches are also initiated with the given number of threads to wait for and get passed into each phase. Another overall countdown latch is applied to make sure all threads has ended before calculating for wall time and other statistics. All these countdown latches are passed into each phase and threads for countdown operation.

*Phase* class is used to initiated each phase, including warmup, peak and cooldown. Within each phase, a certain number of threads gets initialized where post operations will be done within each thread.

Each thread is initialized within *PostThread* class. In this class, given number of post operations is done by having the class implements Runnable and @Override run(). Global variables that were initiated in the main class were used to count the number of posts that was successfully sent, as well as the fail posts. To make sure that the process is thread-safe, synchronized methods *incrementSuccessful* and *incrementFail* were adopted to ensure the correctness of calculation.

To store data of timestamp, latency and other information that was related to each post. A class named *RequestData* was used. The object was stored in a blockingqueue which was updated within each thread when a request is sent and a response was received.

In order to generating .csv files and display the data and statistics. I have two seperated classes named *BlockingQueueToCSV* and *GenerateStatistics*. Apache commons math package was used to help calculating statistical results.

**3. Results**

**Part1**: Client (Part 1) - run your client with 32, 64, 128 and 256 threads, with numSkiers=20000, numLifts=40 and numRuns=20. Include the outputs of each run in your submission (showing the wall time) and plot a simple chart showing the wall time by the number of threads.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Threads | 256 | 128 | 64 | 32 |
| Wall time (millisec) | 305439 | 322780 | 402166 | 590734 |
| numRequestSuccessful | 400000 | 400000 | 400000 | 400000 |
| numRequestFail | 0 | 0 | 0 | 0 |

**Part2**: Client (Part 2) - run the client as per Part 1, showing the output window for each run. Also generate a plot of throughput and mean response time against number of threads.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Threads | 256 | 128 | 64 | 32 |
| Wall time (millisec) | 307627 | 337858 | 421697 | 600321 |
| Difference than part1  (%) | 0.71 | 0.04 | 0.05 | 0.02 |
|  | | | | |
| numRequestSuccessful | 400000 | 400000 | 400000 | 400000 |
| numRequestFail | 0 | 0 | 0 | 0 |
|  | | | | |
| Mean Latency (ms) | 167.97 | 82.60 | 47.46 | 30.79 |
| Median Latency (ms) | 111 | 84 | 43 | 26 |
| 99th percentile Latency (ms) | 1080 | 290 | 156 | 103 |
| Max Latency (ms) | 10782 | 1564 | 1779 | 1336 |
| Latency Standard Deviation | 217.10 | 48.41 | 31.19 | 26.61 |
| Throughput | 1300.28 | 1239.23 | 994.61 | 677.12 |