Client design for cs6650 assignment1

* Packages
  + I have a client package and a csvfiles sub package.
* Major classes
  + Client class
    - Client class handles three different phases and has a main function to initialize the program.
      * Three phases
        + Key takeaways here are executor service and countdown latch. I use executor service to create a fix sized thread pool to execute multithreads. I use countdown latch to make sure when 10% of phase 1 threads finish, phase 2 can start running, and when 10% of phase 2 threads finish, phase 3 can start running. I use latch to ask threads to await until the latch counts down to 0.
      * Client class, as the main class, also calls stats class to generate statistics and populate relevant csv file.
  + Record class
    - I create the record object for each post request. It has before sending request timestamp, after receiving response timestamp, latency, and request type.
  + Stats class
    - Stats class calculates the statistics of the program and write each record to a csv file. It’s run after three phases are finished to make sure the csv parsing will not affect the wall time.
  + Thread class
    - Thread class represents a single thread. It implements runnable and has a run method. Each thread is executing the post request through swagger API and receive the response back from the server. Each thread also takes record of request amount, response amount, latency, etc. Countdown latch is used here to countdown when a thread finishes running.
* Choices of data structures
  + Blocking queue
    - I use blocking queue for an array of records that each thread generates. Since we will probably have around 360000 threads running, it’s important to use a thread-safe data structure to avoid unnecessary collision, waiting, or race conditions.
    - I also use blocking queue for collecting request and response count from each phase.
  + Atomic integer
    - I use atomic integer as the data structure for my request count, successful response count and failure count. Atomic integer is better than integer in multithread situations. It’s thread-safe and it’s more efficient for threads to access and update the amount.
* Overall workflow/relationships
  + The overall workflow is that when phase 1 starts running, thread numbers for phase 1 starts to run, and countdown latch starts to countdown. When first countdown latch reaches 0, phase 2 starts running. Then the second countdown starts to countdown. When the second one reaches 0, phase 3 starts running. After all phases finish their jobs and executor service will let go. Then the statistics will be generated and records will be saved to the csv file. After all the work done, the terminal will print out the stats eventually.







