Project Design

Servlet Design

The server was implemented as a Java Servlet in the `SwipeServlet` class. This class extends the `HttpServlet` class and therefore overrides the `doGet` and `doPost` methods. However, only the `doPost` method is relevant to this assignment.

When the `doPost` method is called, it first validates the given `HttpServletRequest` object's path. It then reads the request's json body into a buffer utilizes the `Gson` api to parse it into a `PostRequestJson` object which is defined as a static nested class within the `SwipeServlet`. `PostRequestJson` has three `String` fields which are associated with the json payload's `swiper`, `swipee`, and `comment` fields.

After json parsing and validation is conducted, the `SwipeServlet` sets the response status to `HttpServletResponse.SC_OK` (i.e. HTTP 200), and informs the client that the POST request was sucessful.

If the request's url path was null or empty, the response code is set to `HttpServletResponse.SC_NOT_FOUND` (i.e. HTTP 404), and the client is informed of the missing url parameters.

If the url path exists but is not valid (i.e. it does not match the pattern `"/swipe/{leftorright}/`, then the response code is set to `HttpServletResponse.SC_NOT_FOUND`, and the client is informed that the path is formatted incorrectly.

Client Design

The client was implemented in the `SwipeClient` class. It has variables to track the number of successful requests and the number of unsuccessful requests. It also has a static `CountDownLatch` which is used for thread synchronization. It has a static nested class named `Requester` which handles the HTTP request logic using the Apache `HttpClient` api. The `Requester` class represents a single thread (i.e. it implements the `Runnable` interface). Its `run` method handles executes N HTTP requests, where $N=$ `numRequestsToSend` which is assigned at instantiation of the `Requestor`.

In part 2, when a thread recieves a response (successful or unsuccessful), it calls the static synchronized `updateRecord()` method on `SwipeClient`. By passing a newly instantiated `RequestStats` object to `updateRecord()` the `Record` class with is updated with info about that completed request (start time, request type, latency, and response code). The `Record` class represents the `.csv` file holding information about all the requests made by the client which are later used to calculate and plot the statistics. The `RequestStats` class represents a single row in the `Record` class.

If the response code was not `HttpServletResponse.SC_NOT_FOUND`, then the `SwiperClient.unsuccessfulRequests` is incremented and the request is reattempted four more times (five total attempts). If the response is successful `SwiperClient.successfulRequests` is incremented and the next request is generated.

Once a thread finishes its N requests, it calls `countDown()` on the `CountDownLatch` to signal to the main thread that it is finished. When all threads are complete, the main thread, which was blocked by having called `await()` on the `CountDownLatch`, then proceeds to calculate the overall statistics and plots the throughput

over time by calling the `calculateRecordStatistics()` and `plotRequestsCompletedOverTime()` on its `Record` instance.