PROJECT 4 — BUS SIMULATION

Fatima Rahman, rahma176@umn.edu
Honors

To run my project, make sure you are using a machine that can run Java 8, unzip all the files in the Project 4 folder, open them in your preferred IDE, then compile and run BusSim.java's main method! Feel free to play around with the number of buses, where the RiderEvents are occurring, and the Stat class in general. To see what exactly is going on with each stop, I commented out two print statements to see how riders are approaching stops and how buses are boarding and deboarding riders at each stop — just uncomment out line 61 in RiderEvent and lines 79-81 in BusEvent. Also commented out is average bus capacity, which is on lines 87-88 in BusSim. My project hierarchy works much like anyone else's — everything is linked to BusSim. All the events, be it the RiderEvent or BusEvent, implements the Event class. I also make use of Q1Gen for my waitlines at each Stop, which implements the QGen interface. Bus and BusEvent make use of linear searches using for and while loops (time efficiency of O(n), which is optimal in terms of time taken to execute the code — $O(n^2)$ would have been far more costly). Priority Queues (taken from CSCI 1933 lecture) are also used when scheduling the agenda of events taking place. Math.random() was also used frequently to generate random destinations for Riders and arrival rates. This is about the max level of complexity any algorithms in this program reach, however many linear searches are implemented one after the other in BusEvent especially.

PROJECT HIERARCHY

BusSim
- Variables: agenda, riderStats, busStats, arrivalInterval
- Methods: main

Bus
- Variables: arrivalTime, lst (ArrayList of Riders), max, express, totalPassengersBoarded, stats, number
- Methods (besides constructor): addRider, removeRidersAtStop, isFull, isExpress, setArrivalTime, getNumber, getStats, getCurrentNumOfPassengers

BusEvent
- Variables: currentStop, bus, timeRemaining
- Methods (besides constructor): setStop, getStopNumber, getBus, setTimeRemaining, run

Rider
- Variables: arrivalTime, boarding, destination, stats
- Methods (besides constructor): getArrivalTime, getBoarding, getDestination, getStats, displayStats

RiderEvent
- Variables: currentStop, stopSelect, arrivalPercents
- Methods (besides constructor): decide, chooseArrivalRate, chooseStop, run

Stop
- Variables: waitline, name, number, stopStat
- Methods (besides constructor): enter, getQLength, getStopNumber, getName, remove, getStats, printStats, getWaitline, setWaitline

Stat

- Variables: index, lastUpdateTime, oldQLength, lastQUpdateTime, count, totalTime, maxWait, averageWait, maxQLength, averageQLength, averageServiceTime, capacity, oldCapacity, averageCapacity, lastCapacityUpdateTime
- Methods (other than constructor): getAverageQLength, updateQueueStats, updateBusyTimeStats, updateServiceTimeStats, updateWaitTimeStats, updateCapacityStats, displayStats

There seem to be no bugs with the code, however I was quite a bit sleep deprived when making this so some errant comments may be amiss. I also was unable to run this project until very recently due to some major technical difficulties with both laptops I used. Thanks!

ALL CREDIT FOR classes Q1Gen, PQ, Q, Q2, Event, Segment, NGen, and QGen lie with CSCI 1933 lecture. My Stat class is also heavily drawn from the Car Washer example from lecture. The rest of my classes are inspired by Lab 10 from CSCI 1933.