Kristine Trinh

nlt895

11190412

**Report**

Tests were run 1000 times for every simulation

First simulation results:

Total number of customers served: 357

Maximum number of customers served in 1 window per hour: 32

Maximum number of customers served in 2 window per hour: 32

Average number of customers served in 1 window per hour: 29

Average number of customers served in 2 window per hour: 29

Average time in the queue: 101min 28sec

Maximum time in the queue: 204min 37sec

Average 1 queue length: 119

Average 2 queue length: 119

Maximum 1 queue length: 238

Maximum 2 queue length: 237

Second simulation results:

Total number of customers served: 358

Maximum number of customers served in 1 window per hour: 32

Maximum number of customers served in 2 window per hour: 32

Average number of customers served in 1 window per hour: 29

Average number of customers served in 2 window per hour: 29

Average time in the queue: 101min 3sec

Maximum time in the queue: 202min 50sec

Average queue length: 238

Maximum queue length: 476

What I can say from the results, statistic is very similar. Customers wait the same time. The only difference I see is that in second variant queue grows faster as all customers are in this one queue, and decreases also faster as two windows are serving.

So the essence of this fable is as follows: you won’t get served faster anyway

**About design:**

To Customer class I’ve added field timeSpentAtTheWindow. Also static field customerCount to for counting created customers. Also getters and setter for them.

To CustomerQ I added “Queue<Customer> customers” field and length() method to get current queue length.

To Event class I added windowNumberField which can store number of window to which Customer approaches or number of window to by which queue is standing.

It has inner Enum for event types.

Also added method compareTo to sort events in PriorityQueue by eventClockTime.

Abstract LList class is totally different from example diagram. It has inner class Element with fields value, prev, next. Also it has fields: head and tail of type Element and size of type int. It implements method add(), wich adds elements to the tail and increases size, isEmpty() and size(). It has abstract method removeNext();

LList4Q extends LList and implements removeNext() method by decreasing size, removing and returning the head;

EventQ extends LList and implements removeNext() method by decreasing size, searching for closest event (the one with less time) removing and returning it;

