Ivan Capistran

01282924

Lab 07

UnbufferedSimulation

First test

Results for simulation: 1 clerks, 2 customers with service times of 10

Total time: 10

Number of customers serviced: 1

bank.Customer:[arrival=0, service=10, departure=10]

Total service time: 10

Total system time: 10

/\*In this test

“one customer leaves because there is no available clerk”.

In the handleEvent method when the second customer object is sent through for process, the first if statement is checked and the findAvailableClerk() method returns a null clerk. Since the simpleSimulation method in the UnbufferedSimulationTester class creates the customer objects with 0 arival time there will never be an available clerk for customer two. When the handleEvent method call is finished for the second customer object, there was no “service”, so that is why the handleEvent method did not add a new event type SERVICE\_FINISHEDto the eventQueue. Therefor the customer “leaves” since it was not serviced. (a SERVICE\_FINISHED event is proof that customers have been serviced by a clerk, since customers are used to manipulate clerk variables so they can generate new Events. These new events are added to the eventQueue in the unbufferedSimulation class when the findAvailableClerk method returns true in the handleEvent method call, which will call itself again in the run() method.)

\*/

Second test

Results for simulation: 2 clerks, 2 customers with service times of 10

Total time: 10

Number of customers serviced: 2

bank.Customer:[arrival=0, service=10, departure=10]

bank.Customer:[arrival=0, service=10, departure=10]

Total service time: 20

Total system time: 20

//Here 2 customers arrived and there were 2 available clerks. Both customers were servied by //clerks and there were only 2 new additions of serviced\_finished events to the eventQueue

Third test

Results for simulation: Notes example with 3 clerks and customer service time 30

Total time: 87

Number of customers serviced: 5

bank.Customer:[arrival=0, service=30, departure=30]

bank.Customer:[arrival=10, service=30, departure=40]

bank.Customer:[arrival=15, service=30, departure=45]

bank.Customer:[arrival=37, service=30, departure=67]

bank.Customer:[arrival=57, service=30, departure=87]

Total service time: 150

Total system time: 150

//Here customers were generated at specific times in the //notesExampleSimulation method.

//When all three clerks are busy the customers generated at time 20, and 25 are not //“serviced”.

//so when customer 6 and 7 arrive with time arrivals of 37 and 57 there are clerks available for both of them.

BufferedSimulation

BufferedSimulationTester class written by Ivan Capistran

Custom Test 1: 5 clerks, 20 customers, departure time 10

Results for simulation: 5 clerks, 20 customers with service times of 10

Total time: 40

Number of customers serviced: 20

bank.Customer:[arrival=0, service=10, departure=10]

bank.Customer:[arrival=0, service=10, departure=10]

bank.Customer:[arrival=0, service=10, departure=10]

bank.Customer:[arrival=0, service=10, departure=10]

bank.Customer:[arrival=0, service=10, departure=10]

bank.Customer:[arrival=0, service=10, departure=20]

bank.Customer:[arrival=0, service=10, departure=20]

bank.Customer:[arrival=0, service=10, departure=20]

bank.Customer:[arrival=0, service=10, departure=20]

bank.Customer:[arrival=0, service=10, departure=20]

bank.Customer:[arrival=0, service=10, departure=30]

bank.Customer:[arrival=0, service=10, departure=30]

bank.Customer:[arrival=0, service=10, departure=30]

bank.Customer:[arrival=0, service=10, departure=30]

bank.Customer:[arrival=0, service=10, departure=30]

bank.Customer:[arrival=0, service=10, departure=40]

bank.Customer:[arrival=0, service=10, departure=40]

bank.Customer:[arrival=0, service=10, departure=40]

bank.Customer:[arrival=0, service=10, departure=40]

bank.Customer:[arrival=0, service=10, departure=40]

Total service time: 200

Total system time: 500

Custom Test 2 : 10 clerks, 12 customers, 30 departure time

Results for simulation: 10 clerks, 12 customers with service times of 30

Total time: 60

Number of customers serviced: 12

bank.Customer:[arrival=0, service=30, departure=30]

bank.Customer:[arrival=0, service=30, departure=30]

bank.Customer:[arrival=0, service=30, departure=30]

bank.Customer:[arrival=0, service=30, departure=30]

bank.Customer:[arrival=0, service=30, departure=30]

bank.Customer:[arrival=0, service=30, departure=30]

bank.Customer:[arrival=0, service=30, departure=30]

bank.Customer:[arrival=0, service=30, departure=30]

bank.Customer:[arrival=0, service=30, departure=30]

bank.Customer:[arrival=0, service=30, departure=30]

bank.Customer:[arrival=0, service=30, departure=60]

bank.Customer:[arrival=0, service=30, departure=60]

Total service time: 360

Total system time: 420

Custom Test 3: Random Simulation 15 clerks

Results for simulation: 15 Clerks

Total time: 165

Number of customers serviced: 15

bank.Customer:[arrival=7, service=9, departure=16]

bank.Customer:[arrival=2, service=23, departure=25]

bank.Customer:[arrival=37, service=22, departure=59]

bank.Customer:[arrival=33, service=35, departure=68]

bank.Customer:[arrival=66, service=6, departure=72]

bank.Customer:[arrival=30, service=44, departure=74]

bank.Customer:[arrival=41, service=45, departure=86]

bank.Customer:[arrival=55, service=38, departure=93]

bank.Customer:[arrival=93, service=7, departure=100]

bank.Customer:[arrival=57, service=44, departure=101]

bank.Customer:[arrival=91, service=12, departure=103]

bank.Customer:[arrival=78, service=35, departure=113]

bank.Customer:[arrival=108, service=7, departure=115]

bank.Customer:[arrival=86, service=30, departure=116]

bank.Customer:[arrival=147, service=18, departure=165]

Total service time: 375

Total system time: 375

Custom Test 4: Notes example Simulation

Results for simulation: Notes example with 7 clerks and customer service time 20

Total time: 77

Number of customers serviced: 7

bank.Customer:[arrival=0, service=20, departure=20]

bank.Customer:[arrival=10, service=20, departure=30]

bank.Customer:[arrival=15, service=20, departure=35]

bank.Customer:[arrival=20, service=20, departure=40]

bank.Customer:[arrival=25, service=20, departure=45]

bank.Customer:[arrival=37, service=20, departure=57]

bank.Customer:[arrival=57, service=20, departure=77]

Total service time: 140

Total system time: 140

First test

Results for simulation: 1 clerks, 2 customers with service times of 10

Total time: 20

Number of customers serviced: 2

bank.Customer:[arrival=0, service=10, departure=10]

bank.Customer:[arrival=0, service=10, departure=20]

Total service time: 20

Total system time: 30

Second test

Results for simulation: 2 clerks, 2 customers with service times of 10

Total time: 10

Number of customers serviced: 2

bank.Customer:[arrival=0, service=10, departure=10]

bank.Customer:[arrival=0, service=10, departure=10]

Total service time: 20

Total system time: 20

Third test

Results for simulation: Notes example with 3 clerks and customer service time 30

Total time: 90

Number of customers serviced: 7

bank.Customer:[arrival=0, service=30, departure=30]

bank.Customer:[arrival=10, service=30, departure=40]

bank.Customer:[arrival=15, service=30, departure=45]

bank.Customer:[arrival=20, service=30, departure=60]

bank.Customer:[arrival=25, service=30, departure=70]

bank.Customer:[arrival=37, service=30, departure=75]

bank.Customer:[arrival=57, service=30, departure=90]

Total service time: 210

Total system time: 246

//Explanation  
// Similarities – Both classes service the customer in the handleEvent method

// Differences – In the handleEvent method for UnbufferedSimulation class, whenever the findAvaiableClerk() method call returns a null clerk when a customer event is pass through the method, the customer will not be used to generate a service\_finished event from the clerk object. In other terms the customer is “lost” and will not be “serviced” when the first if statement doesn’t process.

The BufferedSimulation class has a waiting line so to say that will hold the customer object to be ready to be serviced when the next clerk is available. The waiting line is a LinkedQueue that is initialized in the constructer. In the handleEvent class whenever a CUSTOMER\_ARIVAL event is passed, the first if statement catches it and then it will check if there is an available clerk or not. If there is an available clerk then the customer is serviced, if there is not an available clerk then the customer is added to the LinkedQueue. When a SERVICED\_FINISHED event passes through the handleEvent method it will be caught by the else if statement block and when the clerk is free it then will check the LinkedQueue and select the first customers that are “waiting in line”.