CS231: Project 5

Checkout Lines

Parth Parth | CS231L-A | Dr. Alan Harper

Abstract

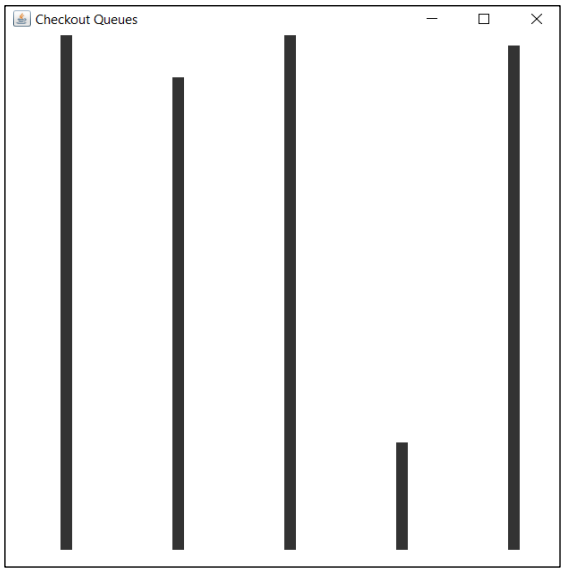
In the project, a simulation of checkout lines has been made. Different strategies have been employed by the simulated customers to compare them and not only get an optimal strategy, but also get an optimal number of items for checking out using each strategy.

The data structure used to implement this is a queue. This is not different from a daily life queue in that the first person to enter the queue (first in) is the first person to leave the queue (first out). This is called a FIFO data structure. In this project, the queue is a simple linear queue that is one ended (i.e. one entry point and one exit point only).

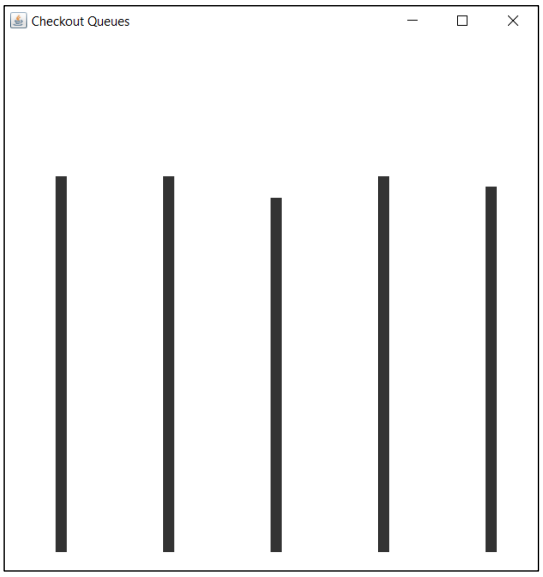
Experimentation and Results

In the following pictures, the PickyCustomer, RandomCustomer, and Pick2Customer classes have been shown. The purpose is to exhibit how there is an optimal number of items for each strategy, exceeding which, the queues grow exceptionally long and keep growing.

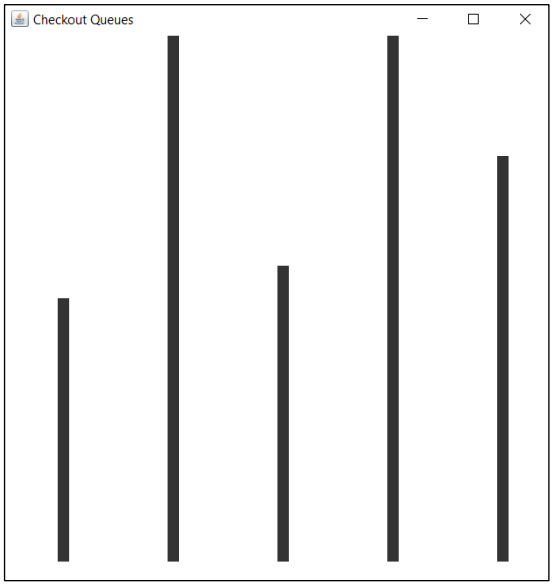
Queues growing too long:



RandomCustomer with 5 checkout queues and between 1-10 items after 2000 time steps

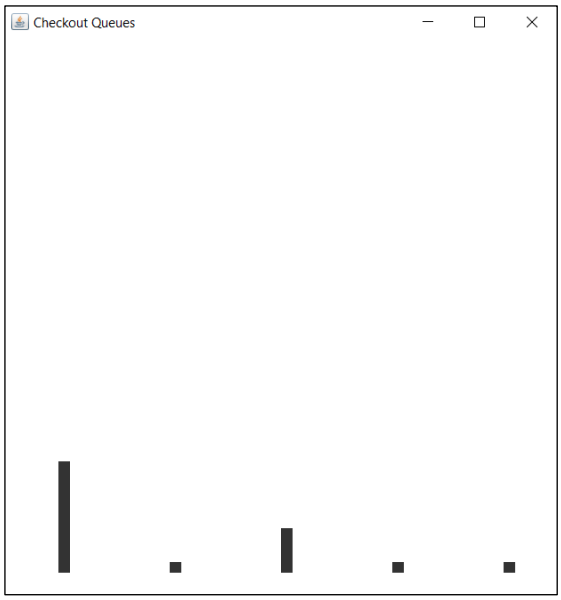


PickyCustomer with 5 checkout queues and between 1-10 items after 2000 time steps

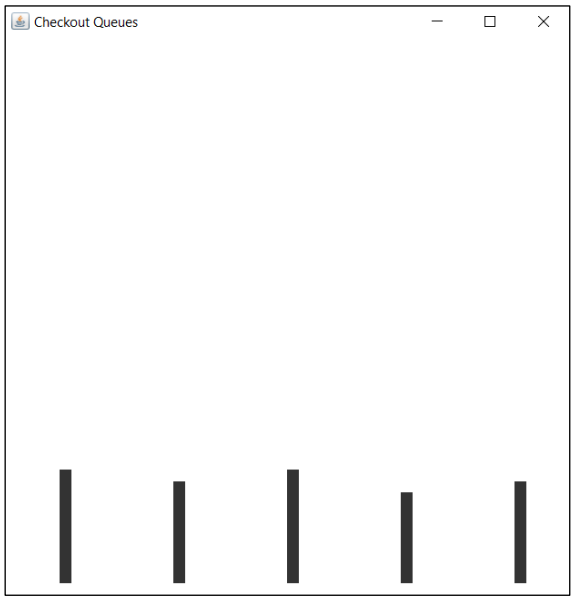


Pick2Customer with 5 checkout queues and between 1-10 items after 2000 time steps

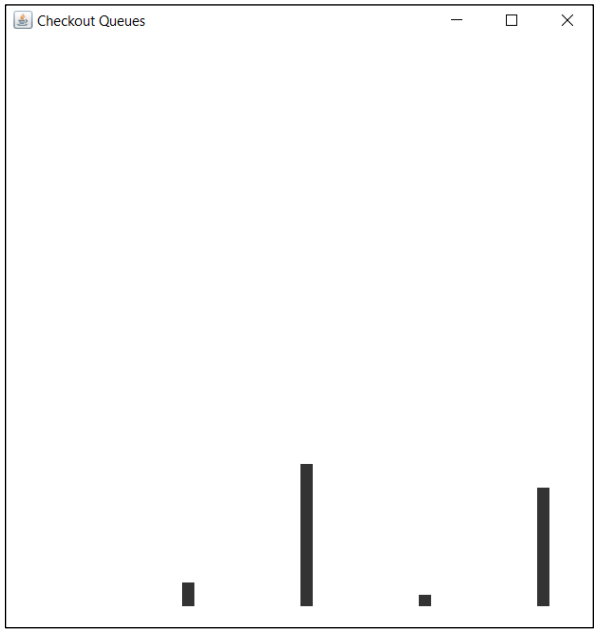
More manageable queues:



RandomCustomer with 5 checkout queues and between 1-8 items after 2000 time steps



PickyCustomer with 5 checkout queues and between 1-8 items after 2000 time steps



Pick2Customer with 5 checkout queues and between 1-8 items after 2000 time steps

Therefore, the optimal number for *RandomCustomer* is 8, *PickyCustomer* is 8, and *Pick2Customer* is 8. *PickyCustomer* can also use 9, but the queues oscillate between becoming so big to go off the canvas and medium, and the time jumps by 12x, so the optimal number is 8.

All three kinds of customers were simulated for 1000 time steps with average time and standard deviation per customer being calculated after every 100 time steps. The results were as follows:

Time steps	Random Customer		Pick2 Customer		Picky Customer	
	Average	S.D.	Average	S.D.	Average	S.D.
100	15.2	8.2	10.8	5.0	11.0	6.6
200	17.0	10.0	13.2	9.7	15.3	9.8
300	16.9	9.5	15.9	13.1	17.7	11.6
400	17.1	9.3	19.1	15.5	19.7	12.1
500	16.1	8.6	20.2	14.9	19.0	11.6
600	15.9	9.4	20.8	15.6	19.6	11.7
700	16.9	10.5	21.2	15.4	20.0	12.4
800	17.4	11.1	21.0	14.9	20.2	14.5
900	18.0	12.1	21.2	14.8	21.0	17.0
1000	18.9	13.4	21.7	15.1	21.9	19.7
Average	16.94	10.21	18.51	13.4	18.54	12.7

All times are in milliseconds (ms)

S.D.=Standard Deviation

Thus, it is somewhat clear that the strategy that the picky customer employs (i.e. to select the shortest line possible) is the best since it has the least average time and also the least standard deviation.

A deeper exploration has been done in extension 2.

Extension 1

In this extension, I have implemented the queue using an array instead of using a Node based queue implementation.

An interesting exploration tangent this extension taught me was that Java does not allow creating arrays of generic types, so I had to create an array of type *Object*. This works as all classes defined by the user would automatically be subclasses of the *Object* class.

Extension 2

In my opinion, the analysis did above does not paint the full picture as it takes some time for the queues to get to dynamic equilibrium.

To remove the bias generated from that, I have run all three classes for 1,000,000 simulations and selected 1,000 simulations at random in between and the results are tabulated below:

Time steps	Random Customer		Pick2 Customer		Picky Customer	
	Average	S.D.	Average	S.D.	Average	S.D.
100	25.3	20.6	26.7	20.7	10.5	3.0
200	25.3	20.6	26.7	20.7	10.5	3.0
300	25.3	20.6	26.6	20.7	10.5	3.0
400	25.3	20.6	26.6	20.7	10.5	3.0
500	25.2	20.5	26.6	20.6	10.5	3.0
600	25.2	20.5	26.6	20.6	10.5	3.0
700	25.2	20.5	26.5	20.6	10.5	3.0
800	25.2	20.5	26.5	20.5	10.5	3.0
900	25.2	20.5	26.5	20.5	10.5	3.0
1000	25.2	20.5	26.5	20.5	10.5	3.0
Average	25.24	20.54	26.58	20.61	10.5	3.0

All times are in milliseconds (ms)

S.D.=Standard Deviation

This makes it very evident that the Picky Customer strategy is much more efficient by at least 2.5x. 

References/Acknowledgements

The *LandscapeDisplay* class was retrieved from https://cs.colby.edu/aharper/courses/cs231/f21/labs/lab05/LandscapeDisplay.java .