CS 150 Project 2 Report

Khalid Almotaery

11/8/2020

1 Introduction

Successful economic activities are one of the results of great economic modeling. In the age of “Big-data”, data-driven companies like Google and Netflix are shaping their models based on information from their massive databases. However, how can you make a great economic model without being a tech giant? An interesting yet cheap alternative to large databases is simulations. Simulations allow us to make models, expect results, test hypotheses, and understand systems without real-life data. This project will explore the power of simulations using the right data structures.

The aim is to build a restaurant simulation that allows us to optimize profit and minimize overflow and customer waiting time using data structures learned in CS150. Additionally, this work will compare LinkedList and ArrayDeque.

The scenario of the simulation goes as follows:

1. The restaurant opens at 6 AM
2. The restaurant has a given number of cashiers (dependent variable)
3. Customers arrive at a given time (dependent variable)
4. The customers get served by the cashiers for a ranged random amount of time.
5. The customers pay ranged random amount of money
6. When customers find an available cashier, they get served by that cashier
7. When do not find an available cashier, the wait in a queue
8. When the queue is eight times larger than the number of cashiers that customer becomes an “overflow.”
9. The restaurant closes at 9 PM and no more orders are accepted.

2 Approach

The project was approached by identifying needed data structures and implementing it following the Object-Oriented Programming principles. The data structures used to make the simulation possible are the following:

1. Queue
2. PriorityQueue
3. ArrayDeque
4. LinkedList
5. ArrayList

The data structures above were selected to mimic real-life structures. The Queue data structure acts just like an actual human line queue. This data structure only allows elements (customers) to come from the back and exit only if all the elements ahead are gone. PriorityQueue organizes data by a priority measure. With regard to this project, the priority variable is the time of each event(arrival or departure of a customer). The data structure ensures that events that happen sooner are happening before the events that will happen later. Thus, this data structure helps make the transitions in the restaurant to be more logical and synchronous. ArrayList and LinkedList were used as containers of customers. Moreover, ArrayList and ArrayDeque were used to implement the Queue interface. An ArrayList is simply a flexible array of elements. A LinkedList allows each node(customer) to have a link to the next node(customer). LinkedList and ArrayDeque operate in similar ways. Because of the structure of ArrayDeque it is more efficient and more flexible than LinkedList. LinkedList goes through the elements in a linear fashion. However, theoretically, the ArrayDeque goes in a circular manner, which allows it to save space and time.

Following OOP, classes named Event, Customer, Information, Simulation, and Controller were constructed. The Event class is responsible for representing the customers' Arrival and Departure events. The class holds information such as the time and type of the event. Instances of this class are gathered in the priority queue discussed earlier. The Customer class represents the customers by counting personalized information for each customer such as arrival time, service time, and money paid. Instances of this class will be placed in the ArrayList data structure. The Information class is responsible for reading the file inputs and generating attributes for each customer. In the Simulation class, the restaurant simulation is ran using the instances of the other classes and inner methods. Finally, the controller class was used to perform experiments. The program diagram is displayed in *figure 1* below:

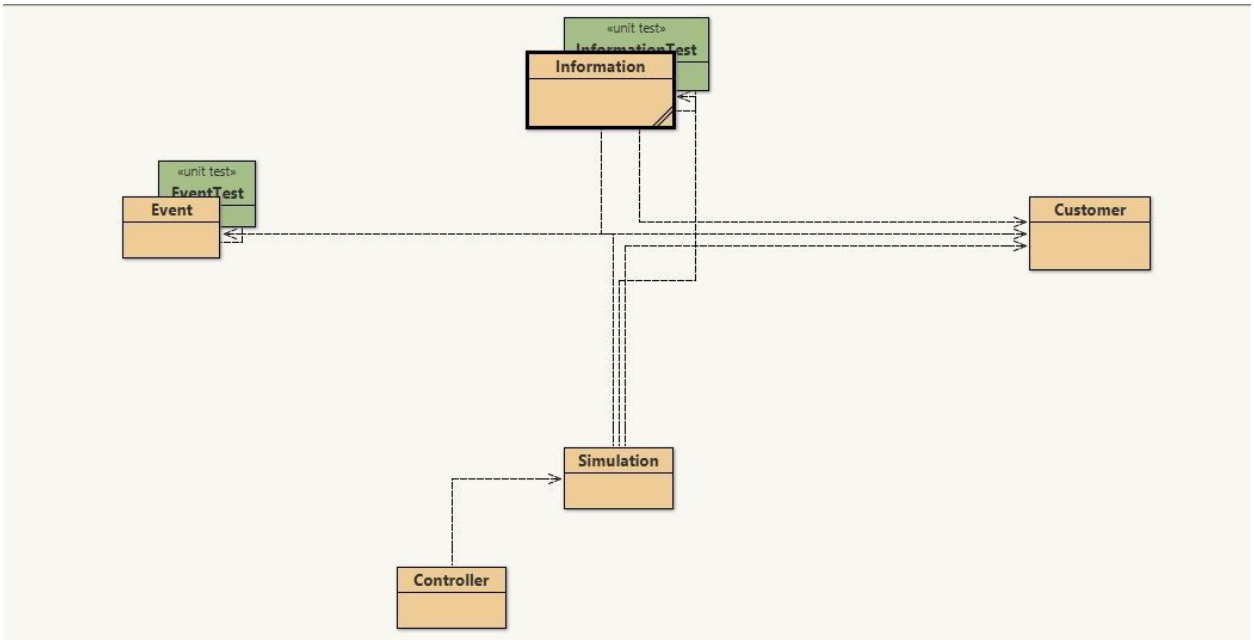


Figure 1

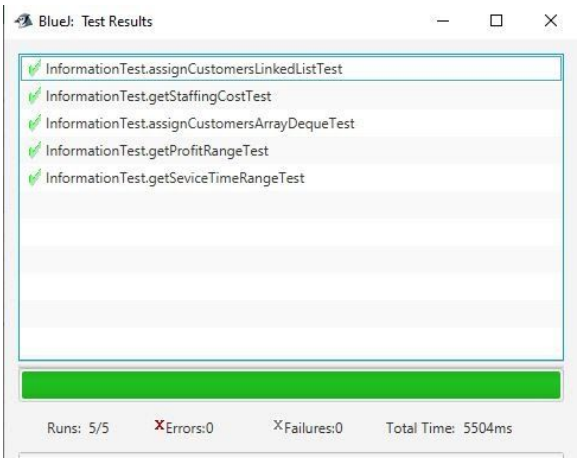


Figure 2

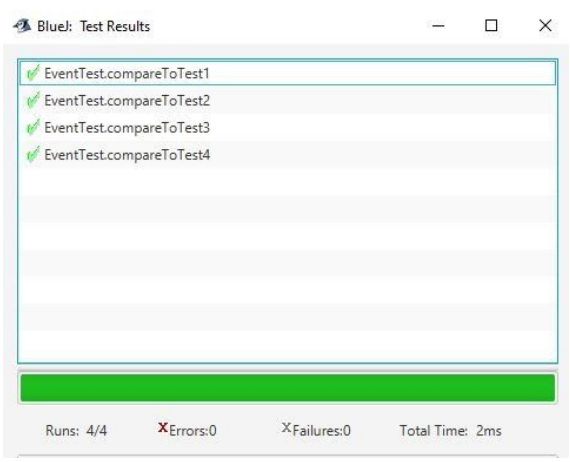


Figure 3

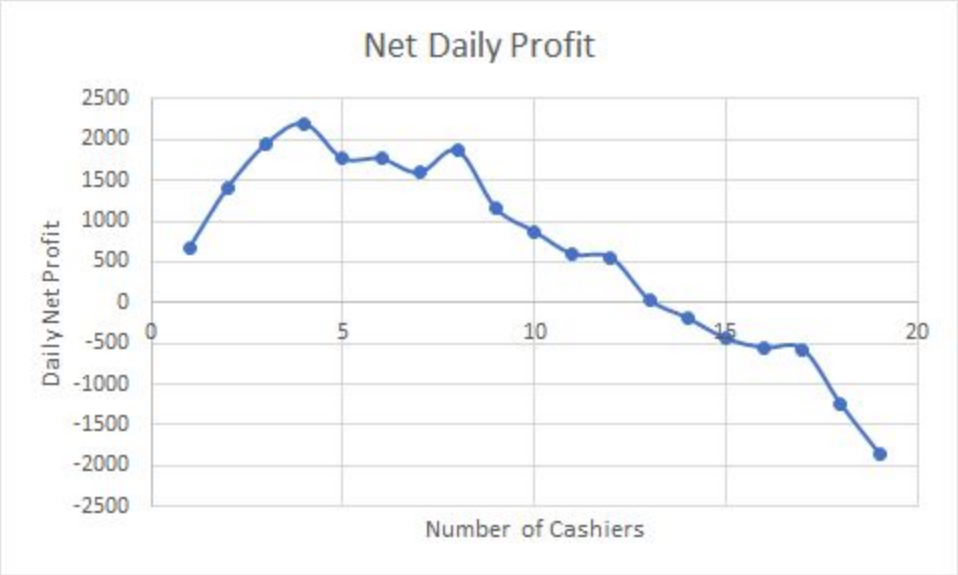
Figures 1 and 2 are the results of unit testing. It shows that all methods tested are working properly.

3 Methods

In an aim to answer the questions of this report, different testing methods were implemented. To understand the fit number of cashiers to have maximum profit, the simulation ran 19 times varying the number of cashiers from 1 to 19. To find out how many cashiers are needed to have a lower customer waiting time and percent overflow, the program ran 9 times

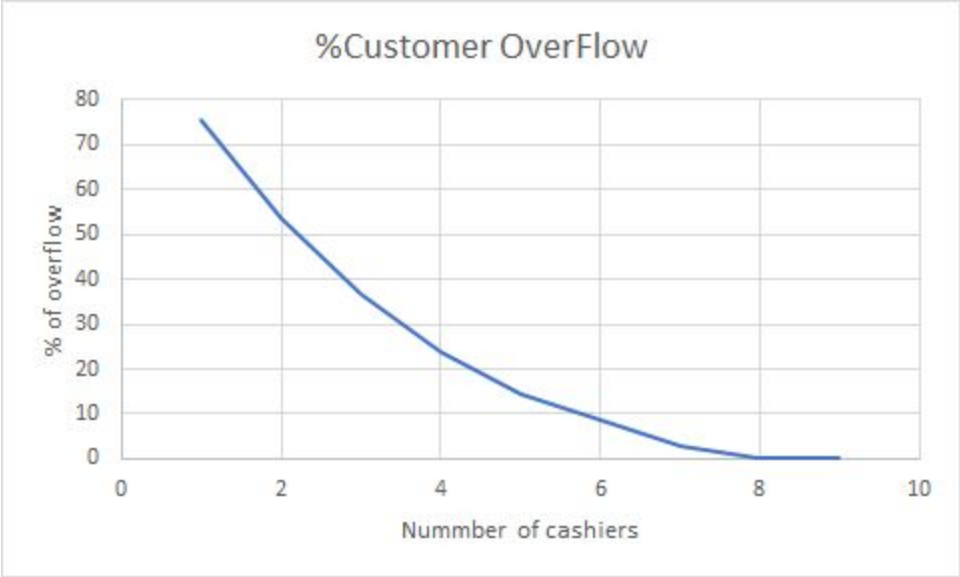
varying the number of cashiers from 1 to 9. Finally, to compare ArrayDeque and LinkedList the program ran for 9 times for each data structure.

4 Data and Analysis



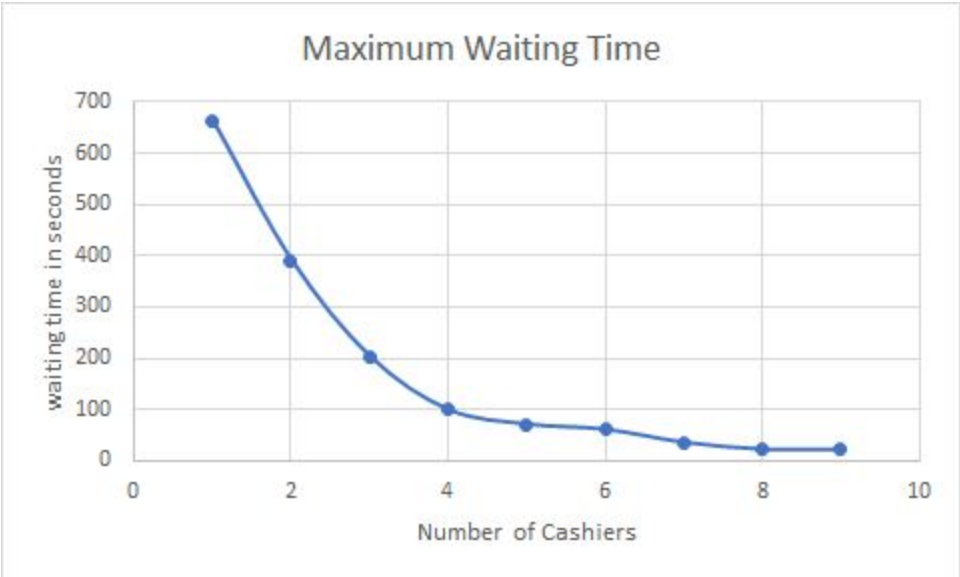
Graph 1

Graph 1 shows the relationship between the number of cashiers and the net daily profit. The net daily profit is calculated by subtracting the profit gained by the cost of operation, which is the number of cashiers multiplied by their salary. It can be noted from the graph above that it is trending in a parabolic way. Having this trend means that there will be a point where the net profit is maximized. In this experiment, the number of cashiers needed to reach the highest amount of daily net profit was **4** cashiers, with a profit of **$2193**. The data shows that having more or less than 4 cashiers will have a negative effect on the daily net profit. In fact, the graph shows that having more than 15 cashiers results in a negative net profit margin.



Graph 2

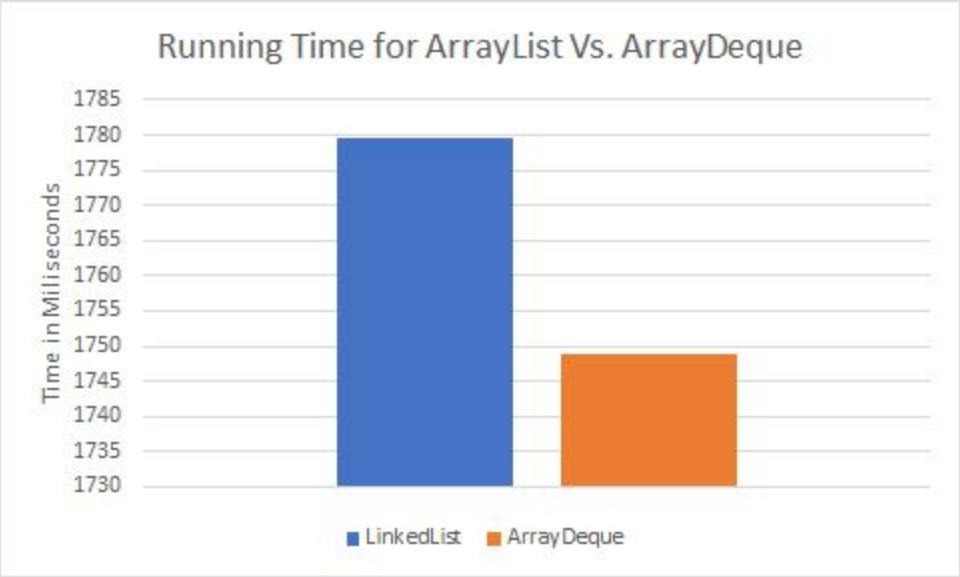
Graph 2 shows the relationship between the number of cashiers and the percentage of customers being an overflow. The percentage of overflow was calculated by dividing the number of customers who were overflow and dividing it by the total number of customers who entered the restaurant and multiplying the division by one hundred. The above graph is following the inverse square trend. This means that the maximum percentage of overflow will accrue with the lowest number of cashiers and the minimum percentage of the overflow will accrue with the highest number of cashiers. In this experiment, having 1 cashier will result in **75.654%** of customer overflow. The percentage of the overflow will be **0%** when there are more than 8 cashiers. This data shows the pros of having more cashiers as it increases customer satisfaction.



Graph 3

Graph 3 displays the relationship between the number of cashiers and the maximum waiting time of customers in the line queue. Similarly to *graph 2,* the above graph is drawn following the

inverse square trend. With one cashier, the waiting time is maximized at **663.625** seconds. Unlike the previous graph, *graph 3* does not show the number of cashiers needed to make the waiting time 0 seconds. Although increasing the number of cashiers will eventually eliminate the waiting time, a high number of cashiers are needed to be present which means the simulation must run many more times. The PC used to run the simulation will also take much more time, which makes this report lack this information.



Graph 4

Graph 4 shows the average time between using ArrayDeque and LinkedList when implementing the Queue structure. It can be noted from the graph above that LikedList take longer the ArrayDeque to implement the simulation. The reason behind the difference in performance comes from how the two structures differ. Theoretically, the ArrayDeque uses a “circular array” which “provides better performance” (Ge, X. (2020). Lecture Notes 5: LinkedList, Stack, Queue, PriorityQueue [PowerPoint slides]).

5 Conclusion

In conclusion, the simulation implemented in this project gave clear answers needed to maximize profits, minimize waiting time, and customer overflow. Furthermore, it was understood that there is an opportunity cost associated with maximizing profits, which is customer service. The number of cashiers needed to optimize customer satisfaction is not the same when optimizing profits. Choosing 4 cashiers will optimize profit, but it will make the percentage of overflow to be **23.86%** and the maximum waiting time to be **99.25** seconds. However, if the restaurant manager places customer service as top priority, 8 cashiers will result in **0%** customer overflow, **21.839** seconds of maximum waiting time and a profit of **$1862.** In other words, the manager will lose **$331** to maintain customer satisfaction.

Works Cited

Ge, Xial. "Lecture Notes 5: LinkedList, Stack, Queue, PriorityQueue" Computer Science 150, 11 Nov. 2020, Lafayette College. Microsoft PowerPoint presentation.

Util. (July 14, 2020) August 29, 2020 from

https://docs.oracle.com/javase/8/docs/api/java/util/package-sumIllmary.html

Junit Assert. (July 14, 2020)) August 30, 2020 from

https://junit.org/junit4/javadoc/latest/org/junit/Assert.html

Junit Test. (July 14, 2020)) August 30, 2020 from

https://docs.oracle.com/javame/test-tools/javatest-441/html/junit.htm