

**Individual Assignment**

**TECHNOLOGY PARK MALAYSIA**

**CT074-3-2**

**CONCURRENT PROGRAMMING**

**APD2F2202CS**

**HAND OUT DATE: 22 August 2022**

**HAND IN DATE: 11 October 2022**

**NAME: LEE HONG YU**

**TP NO: TP063138**

# Introduction

In this assignment, we are given the task to simulate a café while applying java concurrency concepts and techniques. This implementation follows most of the requirements and additional requirements, but not all of them. Let us walk through the step by step with explanation and justification for the implementation of our Java Café.

Contents

[Introduction 2](#_Toc119093972)

[Table of Figures 2](#_Toc119093973)

[Implementation 3](#_Toc119093974)

[Sample Output 10](#_Toc119093975)

[Conclusion 18](#_Toc119093976)

# 

# Table of Figures

[Figure 1: Queue and Leave Probability Function 3](#_Toc119093951)

[Figure 2: Graph Function 3](#_Toc119093952)

[Figure 3: Math Probability Function 3](#_Toc119093953)

[Figure 4: Linked Blocking Deque 4](#_Toc119093954)

[Figure 5: queue() 4](#_Toc119093955)

[Figure 6: owner() 5](#_Toc119093956)

[Figure 7: server() 5](#_Toc119093957)

[Figure 8: Synchronized Queue Block 6](#_Toc119093958)

[Figure 9: Glass or Cup probability func, returns string 6](#_Toc119093959)

[Figure 10: Atomic Integers 7](#_Toc119093960)

[Figure 11: order() 8](#_Toc119093961)

[Figure 12: cleanTable() 9](#_Toc119093962)

[Figure 13: notifyClosed() 10](#_Toc119093963)

[Figure 14: Closing Output 10](#_Toc119093964)

[Figure 15: Owner Serving Instead of Waiter 17](#_Toc119093965)

# Implementation

This café has infinite number of potential customers instead of the finite 10 stated under the assignment’s requirements. Customers can queue up or decide to leave if the queue is too long. To decide which customer leaves or stays to queue, we can apply a probability function.

Text

Description automatically generated

Figure : Queue and Leave Probability Function

The figure below shows as the customer probability of leaving increases gradually, then rapidly approaches the limit of 100%, but never reaches 100%, as the queue length increases. One thing to note is that customer has a tiny chance of leaving even when the customer is first in queue. Again, this goes against the requirement of having only 10 customers, but I think this implementation is better at simulating realism.

Chart, line chart

Description automatically generated

Figure : Graph Function

A picture containing chart

Description automatically generated

Figure : Math Probability Function

When the clock starts running for 20 seconds, threads of customers and the two servers, owner and waiter, are created. When owner and waiter, who are both server class thread objects, will be created right away, and print to console a getting started message. Customers on the other hand are threads generated at a random interval using Math.Random functions. Upon creation, customer will call queue() method located at Serve class. Insite queue(), it first calculate the probability of customer leaving vs customer queuing up. If probability of leaving is greater than probability of queuing, the customer leaves. Otherwise, the customer calls on a synchronized queue\_block. Note that there is no integer given as parameter to new LinkedBlockingDeque as it allows for no limit to this queue\_block size.



Figure : Linked Blocking Deque

Instead of synchronizing the whole method, we can call synchronized block directly onto queue\_block as both customers and servers are using it. Even though Java blocking queue interface is thread safe, we synchronized the blocking queue to use wait and notify methods to use later. This also allow other threads to call other methods of the same class that would otherwise be locked if using synchronized keyword on any method, as any synchronized method will lock the entire class until the method operations are executed.

Customers who decided to stay in queue will be put() into the queue\_block until server check on whoever is in queue, take their order and serve them. If the customer is the first in queue, it will call notify() on queue\_block to notify server to order their drink.

Text

Description automatically generated

Figure : queue()

On the server side, after owner and waiter threads are created, they both rush to obtain an orderLock located at the Serve class. This is because only one server can take one order at a time. This generally leads to the outcome of whoever who first acquired the lock will mostly continue to take orders until the shop closes.

Text

Description automatically generated

Figure : owner()

The other server who didn’t acquire the lock will try again until closing time, but not after calling cleanTable() first to clean remaining glasses and cups left on the table. After cleaning, the server will try to acquire orderLock to order once again, but almost fail all the time because the other server will still be serving.

Text

Description automatically generated

Figure : server()

As a server acquire the orderLock, the server will now take a customer’s order. First the server check if the queue\_block is at size 0. If there is no customer in the block, server will wait until a new customer queues up and notify the server.

Text

Description automatically generated

Figure : Synchronized Queue Block

Once a customer is found, server will remove the customer from the queue with queue\_block.poll(). To order, server checks with the customer on what drinks they want by calling customer.getBeverage(). When customer threads are created, it calls its own method setBeverage(), which is a probability function of 50 50 chances in returning “glass” or “cup” string. As there is only two choices of beverages, coffee or tea, there is only two type of container item, glass or cup. This trick will be useful as we implement customer leaving behind their cups or glasses after finishing their drink and leave.

Text

Description automatically generated

Figure : Glass or Cup probability func, returns string

Server calls getBeverage. As it returns the string “glass” or “cup”, server will act accordingly. If “cup”, server calls on the ServingArea class, which we will name as workStation, cappuccino() method, and vice versa juice() for “juice”. Here, no synchronization is required. As only one server can order at a time, there is 0% chance of two threads calling any methods within workStation to cause synchronization errors. However, we have two atomic integers for glasses and cups, each with a value of 2. These two values are accessed by two potential threads, waiter and owner. When one of them is acquiring glass or cup, the other can be washing and adding them back to the inventory cupboard. Atomic Integer ensures that whichever thread does operation on this integer will respect atomicity, completing the execution of that line of code operation before any other lines of code runs in the program.

Text

Description automatically generated

Figure : Atomic Integers

After acquiring cups or glass respectively, server will call on multiple methods to prepare the drink. To make coffee, it will call on coffee(), milk(), and mix(). To make juice, server calls juiceTap() to pour juice. And now the drink is ready to be serve. This will return a true Boolean if juice or coffee succeed in making. The other possibility is that glass or cups ran out, and this will return false by juice() or cappuccino(). If false, the server will have to call on cleanTable() to acquire cups or glasses left on the table. As the total number of glasses and cups are fixed, we can ensure there will be glasses or cups at the table if the server fails to acquire them for some reason. After cleaning, server calls on cappuchino() or juice() once again to complete the order.

Text

Description automatically generated

Figure : order()

At the last few lines of code from the figure above, customer.setOrderTure() is called and is put to a Array Blocking Queue called table.



Figure : Array Blocking Queue

Purpose of setOrderTrue() is to simulate customer thread running while waiting for the order to be completed. This is done using while loop checking if order==True, if not, sleep for 50ms and loop again. Once order is set to true, this means the customer is put() to the table. Now customer goes to sleep for 1 second to simulate drinking time, and calls setLeaveTure() to set boolean leave==True before its thread is terminated. The purpose of leave boolean is to notify the servers on which customer sitting at the table has finished their drinks. While looping through the table block to check every customer’s leave Boolean, if false, the server cannot take away the customer’s glass or cup to wash and add back to the cupboard. If true, server calls the method table.remove(customer) to remove the customer from table after collecting his or her glass or cup.

Text

Description automatically generated

Figure : cleanTable()

If there are no customers in the table with Boolean leave==True, there will be no left over cups or glasses, hence the line printed: went to table but there is nothing to clean. Again, the only possible way there is no glasses or cups left on table is when the non ordering server calls cleanTable(), and never the server ordering.

That is all the possible events when the java café clock starts its 20 second operating time countdown. Once 20 second is up, it will execute notifyClosed() which prints out that it is closing time, and calls setClosingTime() method for customer generator, owner, and waiter.

Text

Description automatically generated

Figure : notifyClosed()

At this point, new customer thread will stop generating. As different servers have different responsibilities, waiter and owner will exit the café at different time. Waiter will leave once the shop closes and once the waiter finishes whatever order he or she might be doing. Owner will continue to order or stay to take over the job of ordering for the remaining customers queueing inside the queue\_block. Only when all remaining customers ordered, seated at the table, and leave after drinking, then owner will call a last tableClean() function to clear all remaining glasses and cups and add them back to the cupboard. Owner checks for table capacity, making sure it’s 0, and close the shop. The end.

**Text

Description automatically generated**

Figure : Closing Output

# Sample Output

Java Cafe Event Log

--------------------------------------------------

Owner started..

Waiter started..

Waiter is waiting for customer.

No customer currently in queue!

Customer 14 visits cafe. Queue: 1

There is a customer!

Waiter: What would you like to order?

Customer 14: Coffee please.

Owner WENT to table but there is nothing to clean.

Waiter: making Coffee now!

Waiter: Acquired a cup.

Waiter: Acquiring coffee..

Waiter: Acquiring milk..

Waiter: Mixing coffee and milk..

Waiter: Coffee is done making. Here's your coffee dear customer!

Customer 14 took a seat at the table. Table capacity is at 1/10

Waiter is waiting for customer.

No customer currently in queue!

Customer 15 visits cafe. Queue: 1

There is a customer!

Waiter: What would you like to order?

Customer 14 has finished drinking. Customer's cup is left on the table. Leaving now. Goodbye!

Customer 15: Coffee please.

Waiter: making Coffee now!

Owner is washing 1 cup(s) and 0 glass(es).

Owner: finish washing. Added 1 cup(s) and 0 glass(es).

Waiter: Acquired a cup.

Waiter: Acquiring coffee..

Waiter: Acquiring milk..

Waiter: Mixing coffee and milk..

Waiter: Coffee is done making. Here's your coffee dear customer!

Customer 15 took a seat at the table. Table capacity is at 1/10

Waiter is waiting for customer.

No customer currently in queue!

Customer 16 visits cafe. Queue: 1

There is a customer!

Waiter: What would you like to order?

Customer 16: Juice please.

Waiter: making Juice now!

Owner went to table but there is nothing to clean.

Waiter: Acquired a glass.

Customer 15 has finished drinking. Customer's cup is left on the table. Leaving now. Goodbye!

Waiter: Pouring juice from Juice Tap..

Waiter: Juice is done making! Here's your juice dear customer!

Customer 16 took a seat at the table. Table capacity is at 2/10

Customer 17 visits cafe. Queue: 1

There is a customer!

Waiter: What would you like to order?

Customer 17: Coffee please.

Waiter: making Coffee now!

Waiter: Acquired a cup.

Customer 16 has finished drinking. Customer's glass is left on the table. Leaving now. Goodbye!

Waiter: Acquiring coffee..

Waiter: Acquiring milk..

Waiter: Mixing coffee and milk..

Waiter: Coffee is done making. Here's your coffee dear customer!

Customer 17 took a seat at the table. Table capacity is at 3/10

Waiter is waiting for customer.

No customer currently in queue!

Customer 21 visits cafe. Queue: 1

Customer 20 visits cafe. Queue: 2

Customer 19 visits cafe. Queue: 3

Customer 18 visits cafe. Queue: 4

There is a customer!

Waiter: What would you like to order?

Customer 21: Juice please.

Waiter: making Juice now!

Owner is washing 1 cup(s) and 1 glass(es).

Owner: finish washing. Added 1 cup(s) and 1 glass(es).

Waiter: Acquired a glass.

Customer 17 has finished drinking. Customer's cup is left on the table. Leaving now. Goodbye!

Waiter: Pouring juice from Juice Tap..

Waiter: Juice is done making! Here's your juice dear customer!

Customer 21 took a seat at the table. Table capacity is at 2/10

There is a customer!

Waiter: What would you like to order?

Customer 20: Juice please.

Waiter: making Juice now!

Waiter: Acquired a glass.

Customer 21 has finished drinking. Customer's glass is left on the table. Leaving now. Goodbye!

Owner is washing 1 cup(s) and 1 glass(es).

Owner: finish washing. Added 1 cup(s) and 1 glass(es).

Waiter: Pouring juice from Juice Tap..

Waiter: Juice is done making! Here's your juice dear customer!

Customer 20 took a seat at the table. Table capacity is at 1/10

There is a customer!

Waiter: What would you like to order?

Customer 19: Coffee please.

Waiter: making Coffee now!

Waiter: Acquired a cup.

Customer 20 has finished drinking. Customer's glass is left on the table. Leaving now. Goodbye!

Waiter: Acquiring coffee..

Owner is washing 0 cup(s) and 1 glass(es).

Owner: finish washing. Added 0 cup(s) and 1 glass(es).

Waiter: Acquiring milk..

Waiter: Mixing coffee and milk..

Waiter: Coffee is done making. Here's your coffee dear customer!

Customer 19 took a seat at the table. Table capacity is at 1/10

There is a customer!

Waiter: What would you like to order?

Customer 18: Coffee please.

Waiter: making Coffee now!

Waiter: Acquired a cup.

Customer 19 has finished drinking. Customer's cup is left on the table. Leaving now. Goodbye!

Waiter: Acquiring coffee..

Waiter: Acquiring milk..

Waiter: Mixing coffee and milk..

Waiter: Coffee is done making. Here's your coffee dear customer!

Customer 18 took a seat at the table. Table capacity is at 2/10

Waiter is waiting for customer.

No customer currently in queue!

Customer 22 visits cafe. Queue: 1

There is a customer!

Waiter: What would you like to order?

Customer 22: Juice please.

Waiter: making Juice now!

Owner is washing 1 cup(s) and 0 glass(es).

Owner: finish washing. Added 1 cup(s) and 0 glass(es).

Waiter: Acquired a glass.

Customer 18 has finished drinking. Customer's cup is left on the table. Leaving now. Goodbye!

Waiter: Pouring juice from Juice Tap..

Waiter: Juice is done making! Here's your juice dear customer!

Customer 22 took a seat at the table. Table capacity is at 2/10

Waiter is waiting for customer.

No customer currently in queue!

Customer 27 visits cafe. Queue: 1

Customer 26 visits cafe. Queue: 2

Customer 25 visits cafe. Queue: 3

Customer 24 visits cafe. Queue: 4

Customer 23 visits cafe. Queue: 5

There is a customer!

Waiter: What would you like to order?

Customer 27: Coffee please.

Waiter: making Coffee now!

Clock: It's closing time. QUEUE UP NOW FOR LAST CALL!

Closing? Stop Generating Customers!

Owner : We're closing soon!

Waiter : We're closing soon!

Waiter: Acquired a cup.

Customer 22 has finished drinking. Customer's glass is left on the table. Leaving now. Goodbye!

Owner is washing 1 cup(s) and 1 glass(es).

Owner: finish washing. Added 1 cup(s) and 1 glass(es).

-------Closed, finishing remaining orders-------

Waiter: Acquiring coffee..

Waiter: Acquiring milk..

Waiter: Mixing coffee and milk..

Waiter: Coffee is done making. Here's your coffee dear customer!

Customer 27 took a seat at the table. Table capacity is at 1/10

Waiter: Cafe Closed. I'm taking my leave now. Goodbye.

There is a customer!

Owner: What would you like to order?

Customer 26: Coffee please.

Owner: making Coffee now!

Customer 27 has finished drinking. Customer's cup is left on the table. Leaving now. Goodbye!

Owner: Acquired a cup.

Owner: Acquiring coffee..

Owner: Acquiring milk..

Owner: Mixing coffee and milk..

Owner: Coffee is done making. Here's your coffee dear customer!

Customer 26 took a seat at the table. Table capacity is at 2/10

There is a customer!

Owner: What would you like to order?

Customer 25: Coffee please.

Owner: making Coffee now!

Owner: There is no more cups!

Owner is washing 1 cup(s) and 0 glass(es).

Owner: finish washing. Added 1 cup(s) and 0 glass(es).

Customer 26 has finished drinking. Customer's cup is left on the table. Leaving now. Goodbye!

Owner: Acquired a cup.

Owner: Acquiring coffee..

Owner: Acquiring milk..

Owner: Mixing coffee and milk..

Owner: Coffee is done making. Here's your coffee dear customer!

Customer 25 took a seat at the table. Table capacity is at 2/10

There is a customer!

Owner: What would you like to order?

Customer 24: Juice please.

Owner: making Juice now!

Owner: Acquired a glass.

Customer 25 has finished drinking. Customer's cup is left on the table. Leaving now. Goodbye!

Owner: Pouring juice from Juice Tap..

Owner: Juice is done making! Here's your juice dear customer!

Customer 24 took a seat at the table. Table capacity is at 3/10

There is a customer!

Owner: What would you like to order?

Customer 23: Juice please.

Owner: making Juice now!

Owner: Acquired a glass.

Customer 24 has finished drinking. Customer's glass is left on the table. Leaving now. Goodbye!

Owner: Pouring juice from Juice Tap..

Owner: Juice is done making! Here's your juice dear customer!

Customer 23 took a seat at the table. Table capacity is at 4/10

Customer 23 has finished drinking. Customer's glass is left on the table. Leaving now. Goodbye!

Owner is washing 2 cup(s) and 2 glass(es).

Owner: finish washing. Added 2 cup(s) and 2 glass(es).

Table capacity: 0

Owner cleaned and closed the shop. See you tomorrow!

Here is a sample screenshot as proof of work.

Text

Description automatically generated

Figure : Sample Output

In this sample, the waiter acquired the orderLock first and kept serving orders to customers until closing time. The owner who was continuously cleaning the table now takes over serving orders to the last customers remaining in queue.

We can add a 500 second sleep time to ensure that owner runs first before waiter started to show the possibility of owner ordering and waiter cleaning table.

Text

Description automatically generated

Figure : Owner Serving Instead of Waiter

# Conclusion

Some additional requirements that are not included in this implementation is the statistical analysis due to personal time rush and constraints. This has been a particularly fun assignment and I had a great time doing while learning about it. Thank you for reading.