CSCI-201: Principles of Software Development

Fall 2020

Lab07: Doordash Driver

Due: Oct. 13/14

Introduction:

This is a famous concurrency problem that deals with multiple independent threads and a single shared resource. For delivery companies such as Doordash, there can be multiple customers (independent threads) with a single driver (shared resource). If there are no orders, the driver will go to sleep, requiring the next customer who makes an order to wake him up. If the driver is currently making a delivery, a new customer will have to wait for the driver to finish. If there are too many customers already waiting, a new customer cannot place an order. In this lab you will practice:

- Locks and Conditions
- Producer/Consumer Pattern
- Multi-Threaded Programming Design

Implementation:

Download the provided Java files (*Customer.java, DoordashDriver.java, Lab7.java* and *Util.java)* and drag them into your newly created Eclipse project.

Customer.java and Util.java are already implemented. You do not need to modify those files.

DoordashDriver.java implements run for one driver. The driver serves customers from the same pool, and the number of available spots in the app's queue is set to three. You are to implement the methods in that class. The logic for this lab is extremely important. You should spend time understanding the existing code.

Lab7.java is the main driver. In the main() method you instantiate a DoordashDriver thread and use ExecutorService to manage Customer threads. Shut down the service once the driver finishes serving the customers.

Testing:

Please note, your program should not have any exceptions thrown. Below is one possible output (just a part) of your program

```
2020-10-1 11:24:52.285 - Customer 1 is waiting
2020-10-1 11:24:52.285 - No customers, so time to sleep...
2020-10-1 11:24:52.322 - Customers currently waiting: 1
2020-10-1 11:24:52.323 - Someone woke me up!
2020-10-1 11:24:52.323 - Customer 1 is getting delivery.
2020-10-1 11:24:53.215 - Customer 2 is waiting
2020-10-1 11:24:53.215 - Customers currently waiting: 2
2020-10-1 11:24:53.325 - Customer 1 is done getting delivery.
2020-10-1 11:24:53.325 - Checking for more customers...
2020-10-1 11:24:53.326 - Customer 1 is leaving.
2020-10-1 11:24:53.326 - Customer 2 is getting delivery.
2020-10-1 11:24:53.683 - Customer 3 is waiting
2020-10-1 11:24:53.683 - Customers currently waiting: 3
2020-10-1 11:24:54.297 - Customer 4 is waiting
2020-10-1 11:24:54.298 - Customers currently waiting: 3,4
2020-10-1 11:24:54.327 - Customer 2 is done getting delivery.
2020-10-1 11:24:54.327 - Checking for more customers...
2020-10-1 11:24:54.327 - Customer 2 is leaving.
2020-10-1 11:24:54.327 - Customer 3 is getting delivery.
2020-10-1 11:24:54.343 - Customer 5 is waiting
2020-10-1 11:24:54.343 - Customers currently waiting: 4,5
2020-10-1 11:24:54.705 - Customer 6 is waiting
2020-10-1 11:24:54.705 - Customers currently waiting: 4,5,6
```

Grading Criteria:

Labs are graded based on your understanding of the course material. To receive full credit, you will need to:

1) complete the lab following the instructions above
2) show your understanding of the lab material by answering questions upon check-off

If there is a discrepancy between your understanding of the material and your implementation (i.e. if your code is someone else's work), you will receive a grade of 0 for the lab.

Implementation: **(8 Points)**

1) *Lab7.java*

a) 2 - the main() method is complete and working
b) 1 - the student is on the right track, but the implementation is incomplete
c) 0 - the student implements less than 50%

2) *DoordashDriver. Java.* In this class you grade three methods: addCustomerToWaiting(), wakeUpDriver(), and run().

a) 2 – for each method if it is complete and working
b) 1 - the student is on the right track, but the implementation is incomplete
c) 0 - the student implements less than 50%

Check-off Questions: **(2 Points)**

Please randomly select one question.

Question 1
What will happen if two threads try to modify same resource without synchronization in java?

Question 2
What is deadlock in java?

Question 3
What are the similarities and differences between a lock and a semaphore?

Question 4
Can await(), notify(), and notifyAll() be invoked from any object? What is the purpose of these methods?

Question 5
Explain a lock fairness policy.