Travis Dennis

CS 365

Lab 4

Part I, Producer and Consumer

module-info.java | Factory.java | Producer.java | Channel.java | MessageQueue.java | Consumer.jav

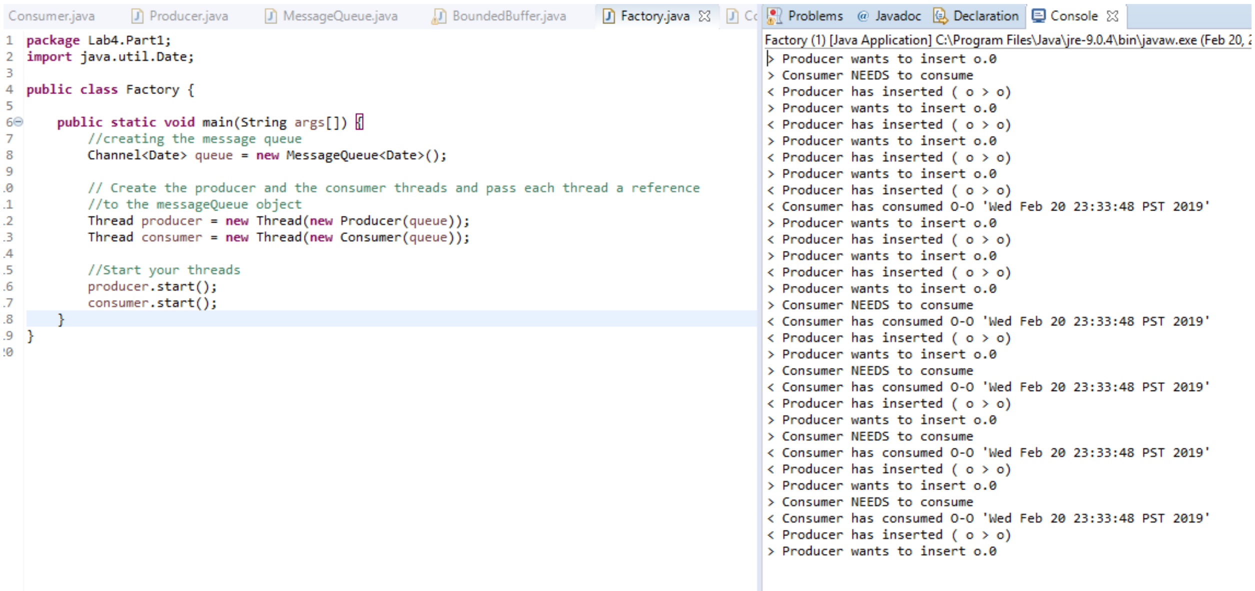
```java
package Lab4;
import java.util.Date;

public class Factory {

    public static void main(String args[]) {
        //creating the message queue
        Channel<Date> queue = new MessageQueue<Date>();

        // Create the producer and the consumer threads and pass each thread a reference
        //to the messageQueue object
        Thread producer = new Thread(new Producer(queue));
        Thread consumer = new Thread(new Consumer(queue));

        //Start your threads
        producer.start();
        consumer.start();
    }
}
```

Problems | Javadoc | Declaration | Console

Factory [Java Application] C:\Program Files\Java\jre-9.0.4\bin\javaw.exe (Feb 20, 2019, 11:07:43 PM)

```
Producer produced Wed Feb 20 23:07:43 PST 2019
Consumer wants to consume.
Consumer consumed Wed Feb 20 23:07:43 PST 2019
Consumer wants to consume.
Producer produced Wed Feb 20 23:07:46 PST 2019
Producer produced Wed Feb 20 23:07:48 PST 2019
Consumer wants to consume.
Consumer consumed Wed Feb 20 23:07:46 PST 2019
Producer produced Wed Feb 20 23:07:50 PST 2019
Consumer wants to consume.
Consumer consumed Wed Feb 20 23:07:48 PST 2019
Consumer wants to consume.
Consumer consumed Wed Feb 20 23:07:50 PST 2019
Producer produced Wed Feb 20 23:07:54 PST 2019
Producer produced Wed Feb 20 23:07:56 PST 2019
Consumer wants to consume.
Consumer consumed Wed Feb 20 23:07:54 PST 2019
Producer produced Wed Feb 20 23:07:59 PST 2019
Consumer wants to consume.
Consumer consumed Wed Feb 20 23:07:56 PST 2019
Producer produced Wed Feb 20 23:08:01 PST 2019
Consumer wants to consume.
Consumer consumed Wed Feb 20 23:07:59 PST 2019
Producer produced Wed Feb 20 23:08:04 PST 2019
```

Part II Bounded Buffer



```java
package Lab4.Part1;
import java.util.Date;

public class Factory {

    public static void main(String args[]) {
        //creating the message queue
        Channel<Date> queue = new MessageQueue<Date>();

        // Create the producer and the consumer threads and pass each thread a reference
        //to the messageQueue object
        Thread producer = new Thread(new Producer(queue));
        Thread consumer = new Thread(new Consumer(queue));

        //Start your threads
        producer.start();
        consumer.start();
    }
}
```

```
Factory (1) [Java Application] C:\Program Files\Java\jre-9.0.4\bin\javaw.exe (Feb 20,
> Producer wants to insert o.0
> Consumer NEEDS to consume
< Producer has inserted ( o > o)
> Producer wants to insert o.0
< Producer has inserted ( o > o)
> Producer wants to insert o.0
< Producer has inserted ( o > o)
> Producer wants to insert o.0
< Producer has inserted ( o > o)
< Consumer has consumed O-O 'Wed Feb 20 23:33:48 PST 2019'
> Producer wants to insert o.0
< Producer has inserted ( o > o)
> Producer wants to insert o.0
< Producer has inserted ( o > o)
> Producer wants to insert o.0
> Consumer NEEDS to consume
< Consumer has consumed O-O 'Wed Feb 20 23:33:48 PST 2019'
< Producer has inserted ( o > o)
> Producer wants to insert o.0
> Consumer NEEDS to consume
< Consumer has consumed O-O 'Wed Feb 20 23:33:48 PST 2019'
< Producer has inserted ( o > o)
> Producer wants to insert o.0
> Consumer NEEDS to consume
< Consumer has consumed O-O 'Wed Feb 20 23:33:48 PST 2019'
< Producer has inserted ( o > o)
> Producer wants to insert o.0
> Consumer NEEDS to consume
< Consumer has consumed O-O 'Wed Feb 20 23:33:48 PST 2019'
< Producer has inserted ( o > o)
> Producer wants to insert o.0
```

Part IV Revision

We need to make the bridge crossing not only free of deadlock, but also free from starvation. That could be caused by always giving north bound farmers absolute priority over southbound and if a large amount of northbound traffic appears. Thus we need to have a system to split the traffic flow fairly so that starvation never occurs while deadlock is still avoided. Since we have communication between each side, we can determine a ticket system and the side that accumulates more than three tickets ahead of the other side will start pulling every other ticket until both sides have evened out to closer. When the bridge is open, the next ticket in line shall cross. Since we are allowing one ticket to cross at a time there should not be any deadlock.

This means we can keep track of the number of crosses on each side, determine if one side is unbalanced, and implement a process to handle or share the load of the unbalance until it evens out.