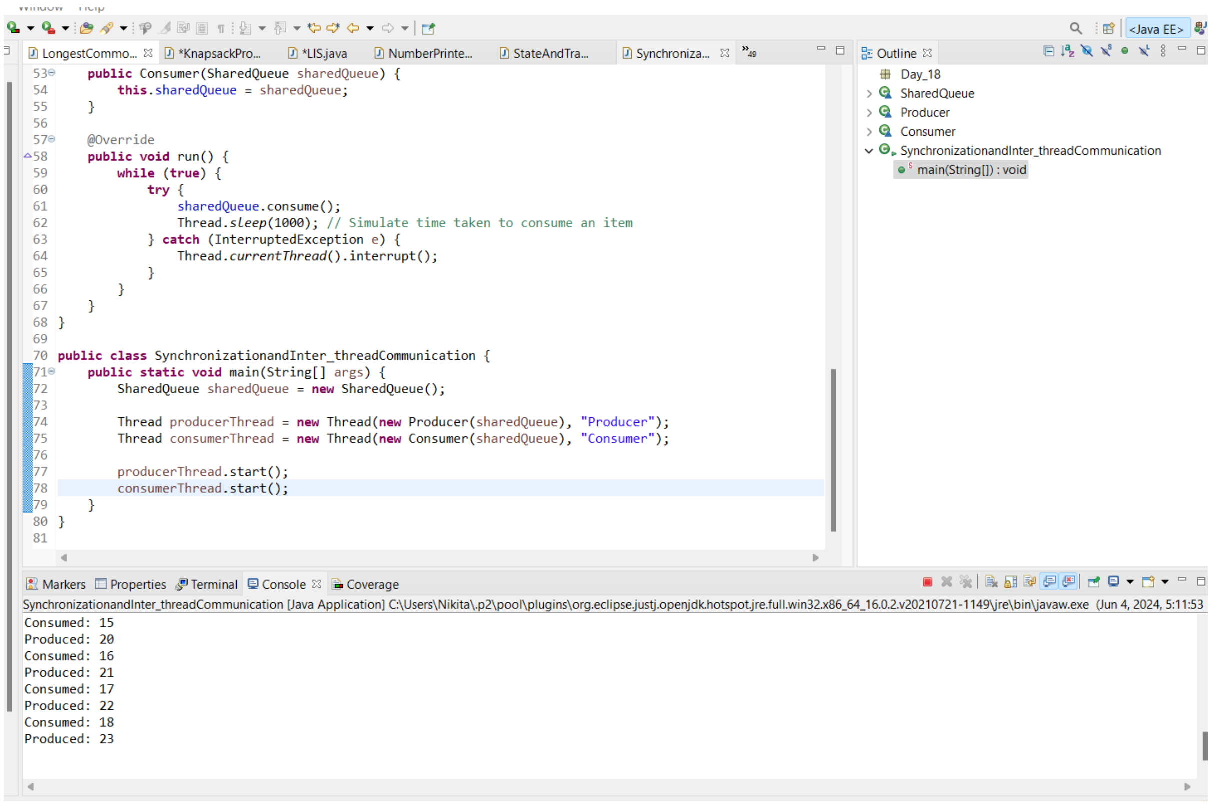
Synchronization and Inter-thread Communication

Implement a producer-consumer problem using wait() and notify() methods to handle the correct processing sequence between threads.

```
import java.util.LinkedList;
import java.util.Queue;

class SharedQueue {
   private final Queue<Integer> queue = new LinkedList<>();
   private final int MAX_SIZE = 5;

   public synchronized void produce(int value) throws InterruptedException {
      while (queue.size() == MAX_SIZE) {
         wait(); // Wait if the queue is full
      }
      queue.add(value);
      System.out.println("Produced: " + value);
      notifyAll(); // Notify consumers that an item is available
   }

   public synchronized int consume() throws InterruptedException {
      while (queue.isEmpty()) {
         wait(); // Wait if the queue is empty
      }
      int value = queue.poll();
      System.out.println("Consumed: " + value);
      notifyAll(); // Notify producers that space is available
      return value;
   }
}

class Producer implements Runnable {
   private final SharedQueue sharedQueue;

   public Producer(SharedQueue sharedQueue) {
        this.sharedQueue = sharedQueue;
    }

    @Override
    public void run() {
        int value = 0;
        while (true) {
            try {
                sharedQueue.produce(value++);
                Thread.sleep(500); // Simulate time taken to produce an item
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
            }
        }
    }
}

class Consumer implements Runnable {
    private final SharedQueue sharedQueue;

    public Consumer(SharedQueue sharedQueue) {
        this.sharedQueue = sharedQueue;
    }

    @Override
    public void run() {
        while (true) {
            try {
                sharedQueue.consume();
                Thread.sleep(1000); // Simulate time taken to consume an item
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
            }
        }
```

```java
    }
}


public class SynchronizationandInter_threadCommunication {
    public static void main(String[] args) {
        SharedQueue sharedQueue = new SharedQueue();

        Thread producerThread = new Thread(new Producer(sharedQueue), "Producer");
        Thread consumerThread = new Thread(new Consumer(sharedQueue), "Consumer");

        producerThread.start();
        consumerThread.start();
    }
}
```



```java
    public Consumer(SharedQueue sharedQueue) {
        this.sharedQueue = sharedQueue;
    }

    @Override
    public void run() {
        while (true) {
            try {
                sharedQueue.consume();
                Thread.sleep(1000); // Simulate time taken to consume an item
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
            }
        }
    }
}

public class SynchronizationandInter_threadCommunication {
    public static void main(String[] args) {
        SharedQueue sharedQueue = new SharedQueue();

        Thread producerThread = new Thread(new Producer(sharedQueue), "Producer");
        Thread consumerThread = new Thread(new Consumer(sharedQueue), "Consumer");

        producerThread.start();
        consumerThread.start();
    }
}
```

```
SynchronizationandInter_threadCommunication [Java Application] C:\Users\Nikita\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_16.0.2.v20210721-1149\jre\bin\javaw.exe  (Jun 4, 2024, 5:11:53
Consumed: 15
Produced: 20
Consumed: 16
Produced: 21
Consumed: 17
Produced: 22
Consumed: 18
Produced: 23
```