1. Write a java program to illustrate inter thread communication using Producer Consumer Problem.

Code :

```
import java.util.LinkedList;

class SharedBuffer {
   private LinkedList<Integer> buffer = new LinkedList<>();
   private int capacity;

   public SharedBuffer(int capacity) {
      this.capacity = capacity;
   }

   public void produce(int item) throws InterruptedException {
      synchronized (this) {
         while (buffer.size() == capacity) {
            wait();
         }
         buffer.add(item);
         System.out.println("Produced: " + item);
         notify();
      }
   }

   public int consume() throws InterruptedException {
      synchronized (this) {
         while (buffer.isEmpty()) {
            wait();
         }
         int item = buffer.removeFirst();
         System.out.println("Consumed: " + item);
         notify();
         return item;
      }
   }
}

class Producer extends Thread {
   private SharedBuffer buffer;

   public Producer(SharedBuffer buffer) {
      this.buffer = buffer;
   }
   @Override
   public void run() {
```

```
        try {
            for (int i = 1; i <= 5; i++) {
                buffer.produce(i);
                Thread.sleep(1000);
            }
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}

class Consumer extends Thread {
    private SharedBuffer buffer;

    public Consumer(SharedBuffer buffer) {
        this.buffer = buffer;
    }

    @Override
    public void run() {
        try {
            for (int i = 1; i <= 5; i++) {
                int item = buffer.consume();
                Thread.sleep(1000);
            }
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}

public class procom {
    public static void main(String[] args) {
        SharedBuffer buffer = new SharedBuffer(3);
        Producer producer = new Producer(buffer);
        Consumer consumer = new Consumer(buffer);

        producer.start();
        consumer.start();
    }
}
```

Output -

```
Produced: 1
Consumed: 1
Produced: 2
Consumed: 2
Produced: 3
Consumed: 3
Produced: 4
Consumed: 4
Produced: 5
Consumed: 5
```