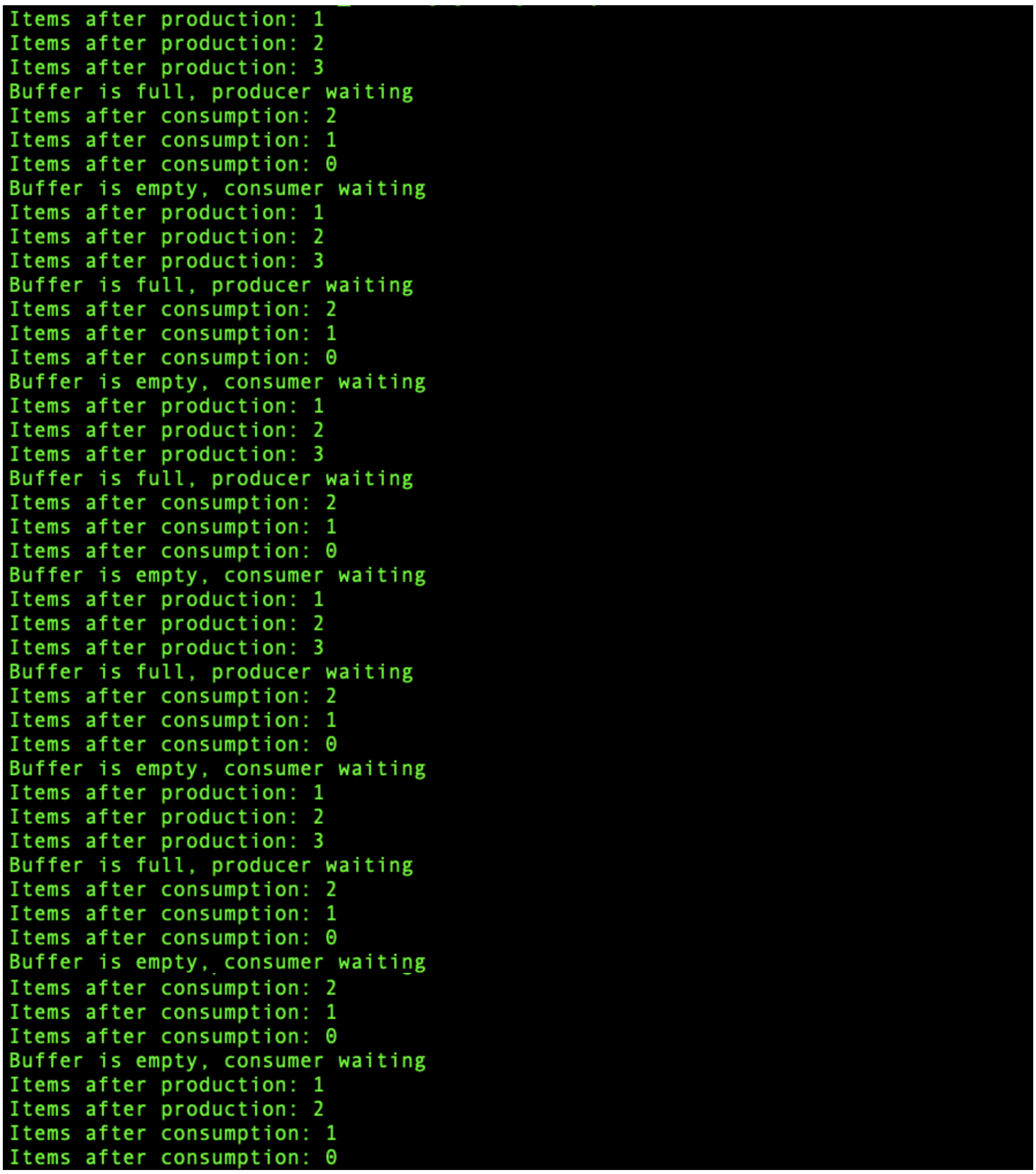
OOP Assignment 3

NAME: RITABROTO GANGULY
ROLL: 001910501090

Implement a multithreaded program in Java to solve the producer-consumer problem.
Producer and Consumer are the two entities here who share the same buffer.
The producer can either go to sleep or discard data if the buffer is full.
The next time the consumer removes an item from the buffer, it notifies the producer, who starts to fill the buffer again.
In the same way, the consumer can go to sleep if it finds the buffer to be empty.
The next time the producer puts data into the buffer, it wakes up the sleeping consumer.
An inadequate solution could result in a deadlock where both processes are waiting to be awakened.

```
import java.io.*;
import java.util.*;

class Product{
	private int items = 0;
	public synchronized void produce(){
		while(items>=MyThread.BUFF_SIZE){
			System.out.println("Buffer is full, producer waiting");
			try{
				wait();
			}catch(InterruptedException e){System.out.println(e);}
		}
		items++;
		notifyAll();
		System.out.println("Items after production: "+items);
	}
	public synchronized void consume(){
		while(items<=0){
			System.out.println("Buffer is empty, consumer waiting");
			try{
				wait();
			}catch(InterruptedException e){System.out.println(e);}
		}
		items--;
		notifyAll();
		System.out.println("Items after consumption: "+items);
	}
}

class ConsumerRun implements Runnable {
	Product p;
	ConsumerRun(Product p){this.p = p;}

	public void run(){
		for(int i=1;i<=MyThread.RUNS;i++)
			p.consume();
	}
}
class ProducerRun implements Runnable {
	Product p;
	ProducerRun(Product p){this.p = p;}

	public void run(){
		for(int i=1;i<=MyThread.RUNS;i++)
			p.produce();
	}
}

class MyThread{
	public static final int BUFF_SIZE = 3;
	public static final int RUNS = 20;

	public static void main(String[] args) throws Exception{
		Product p = new Product();
		Thread pt = new Thread(new ProducerRun(p));
		Thread ct = new Thread(new ConsumerRun(p));
		pt.start();
		ct.start();
	}
}
```



```
Items after production: 1
Items after production: 2
Items after production: 3
Buffer is full, producer waiting
Items after consumption: 2
Items after consumption: 1
Items after consumption: 0
Buffer is empty, consumer waiting
Items after production: 1
Items after production: 2
Items after production: 3
Buffer is full, producer waiting
Items after consumption: 2
Items after consumption: 1
Items after consumption: 0
Buffer is empty, consumer waiting
Items after production: 1
Items after production: 2
Items after production: 3
Buffer is full, producer waiting
Items after consumption: 2
Items after consumption: 1
Items after consumption: 0
Buffer is empty, consumer waiting
Items after production: 1
Items after production: 2
Items after production: 3
Buffer is full, producer waiting
Items after consumption: 2
Items after consumption: 1
Items after consumption: 0
Buffer is empty, consumer waiting
Items after production: 1
Items after production: 2
Items after production: 3
Buffer is full, producer waiting
Items after consumption: 2
Items after consumption: 1
Items after consumption: 0
Buffer is empty, consumer waiting
Items after consumption: 2
Items after consumption: 1
Items after consumption: 0
Buffer is empty, consumer waiting
Items after production: 1
Items after production: 2
Items after consumption: 1
Items after consumption: 0
```