**Java Assignment 2**

*This* ***assignment*** *is submitted in partial fulfilment of*

*the requirements for the course*

*of*

**Java & PHP Lab**

**CSE 230**

By

Vivek Kumar Ahirwar

(Scholar Number: 191112419)



**MAULANA AZAD**

**NATIONAL INSTITUTE OF TECHNOLOGY**

**BHOPAL – 462 003 (INDIA)**

**March 2021**

**Assignment Questions**

[1.](#_gjdgxs) Write a JAVA program to illustrate threading using the producer consumer problem. The buffer size is fixed. The producer produces items but not when buffer is full and consumer consumes items but not when buffer is empty. 3

[1.1 Solution: 3](#_30j0zll)

[1.2 Source Code: 3](#_1fob9te)

[1.3 Output: 5](#_3znysh7)

## Write a JAVA program to illustrate threading using the producer consumer problem. The buffer size is fixed. The producer produces items but not when buffer is full and consumer consumes items but not when buffer is empty.

### 1.1 Solution:

The ‘Thread’ class in *java.lang* package provides various methods to control the behaviour of threads in an application.

To implement the problem we are using the *wait()* and *notify()* methods for the inter-thread communication that are part of the *java.lang.Object* class.

### 1.2 Source Code:

package Java;

import java.util.\*;

//import java.lang.Object;

public class Threading extends Thread{

*boolean* prod, cons;

    static Buffer b = new Buffer();

    Threading(*boolean* *p*){

        prod = *p*;

        cons = !*p*;

    }

    public *void* run() {

        try {

            if(prod)

                b.producer();

            else

                b.consumer();

        } catch(InterruptedException e) {

            e.printStackTrace();

        }

    }

    public static *void* main(String[] *args*) {

        Threading producer = new Threading(true);

        Threading consumer = new Threading(false);

        producer.start();

        consumer.start();

    }

}

class Buffer {

    ArrayList<Integer> buff = new ArrayList<>();

*int* capacity = 7;

    public *void* producer() throws InterruptedException {

*int* product = 1;

        while(true)

            synchronized(this) {

                while(buff.size()==capacity)

                    wait();

                System.out.println("Producing : "+product);

                buff.add(product++);

                notify();

                Thread.sleep(500);

            }

    }

    public *void* consumer() throws InterruptedException {

        while(true)

            synchronized(this) {

                while(buff.size() == 0)wait();

                System.out.println("Consuming :"+buff.remove(0));

                notify();

                Thread.sleep(500);

            }

    }

}

### 1.3 Output:

