OOPS

LAB-12

Question1:

Apply Interthread communication to solve the Producer-Consumer problem with a common or shared bounded buffer (Queue) holding up to5 elements.—The producer consumer problem is a synchronization problem. There is a fixed size buffer and the producer produces items and enters them into the buffer. The consumer removes the items from the buffer and consumes them.—Aproducer should not produce items into the buffer when the consumer is consuming an item from the buffer and vice versa. So,the buffer should only be accessed by the producer or consumer at a time.—Wheneverbuffer is filled up and no more space to add theelement into the queue(buffer) producer has to wait until the buffer is emptied by consumer. Whenever the buffer is empty and no more items are available for consumption the consumer should wait for producer to produce elements.Write a solution for N elements, where N is multiple of 5 or greater than 5,other than 0.

Code:

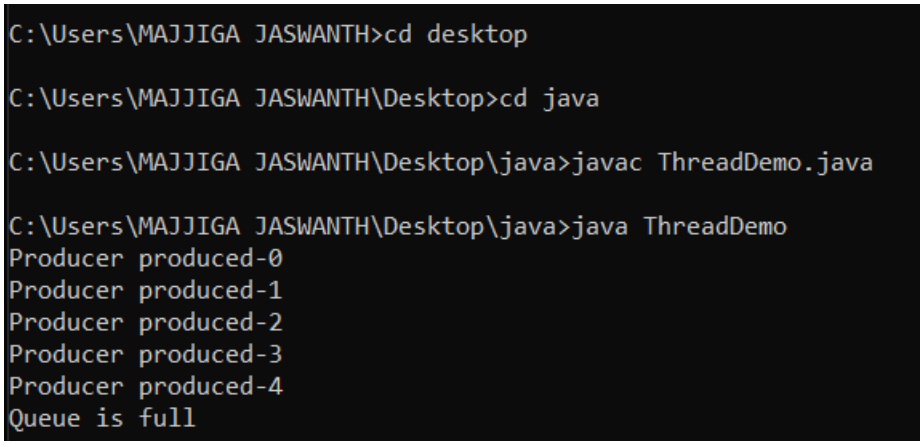
```
public class ThreadDemo {

public static void main(String args[])throws
InterruptedException {

final PC pc = new PC();

Thread t1 = new Thread(new Runnable() {

 public void run()

 {

 try {

 pc.produce();

 }

 catch (InterruptedException e) {

 e.printStackTrace();

 }

 }

 });
```

```
 Thread t2 = new Thread(new Runnable() {
 public void run()
 {
 try {
 pc.consume();
 }
 catch (InterruptedException e) {
 e.printStackTrace();
 }
 }
 });
 t1.start();
 t2.start();
 t1.join();
 t2.join();
}
}
class PC {
 int a[]=new int[5];
 int value = 0;
 public void produce() throws InterruptedException
 {
 while (true) {
 synchronized (this)
 {
 if(value==a.length)
 {
 System.out.println("Queue is full");
 }
 while (value==a.length)
```

```
wait();
System.out.println("Producer produced-"+ value);
a[value]=value+1;
value++;
notify();
Thread.sleep(1000);
}
}
}
public void consume() throws InterruptedException
{
while (true) {
synchronized (this)
{
if(value==0)
{
System.out.println("Queue is Empty");
}
while (value == 0)
System.out.println("Queue is empty");
wait();
int val=a[0];
for(int i=0;i<4;i++)
{
a[i]=a[i+1];
}
a[4]=0;
value--;
System.out.println("Consumer consumed-"+ val);
notify();
```

```
 Thread.sleep(1000);
 }
 }
 }
}
```

Output:



Question2:

Develop a simple chat application between two users using "Send-Wait-Receive" Protocol: Once a user sends a message, he waits till a message is received from other user. The users are "User1" and "User2". At initial stage of application, User1 is in sending mode and User2 is in receiving mode. These two users are sending and receiving the messages alternatively.—Create a Chat class with two methods: sendMessage and recvMessage—Create two threads to represent two users,User1 and User2.—Use Interthread communication to exchange messages.—No need to maintain any chat history.Example:User1: HiUser2: HelloUser1: R u preparing for exam?User2: YesUser1: Ok. ByeUser2: Bye.User1(User2):HIUser2(User1):HIUser2(User1):HelloUser1(User2) :Hello

Code

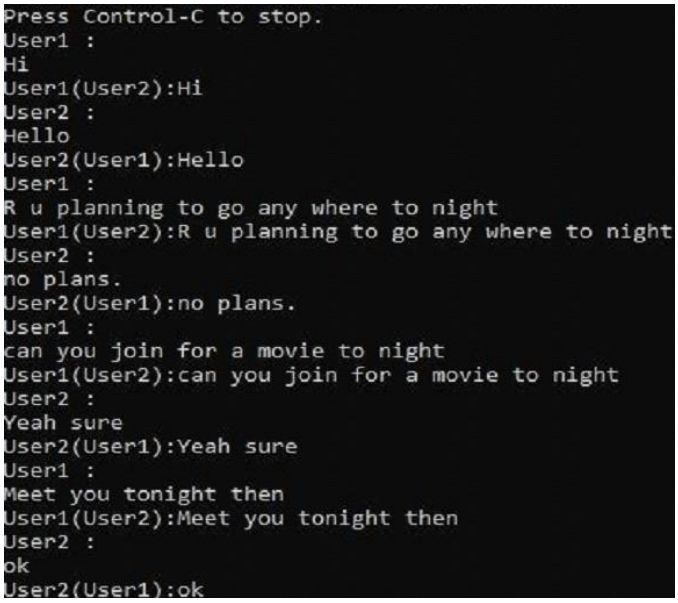
```
import java.util.*;

class Q

{

Scanner insendMessage=new

Scanner(System.in);

int n;
```

```
String Message;
boolean valueSet = false;
synchronized int recvMessage()
{
while(!valueSet)
try
{
wait();
}
catch(InterruptedException e)
{
System.out.println("InterruptedException caught");
}
System.out.println("User2 : ");
Message=insendMessage.nextLine();
valueSet = true;
System.out.println("User2(User1):" + Message);
valueSet=false;
notify();
return n;
}
synchronized void sendMessage(int n)
{
while(valueSet) try
{
wait();
}
catch(InterruptedException e)
{
System.out.println("InterruptedException caught");
```

```
}          this.n = n;
System.out.println("User1 : ");
Message=insendMessage.nextLine();
valueSet = true;
System.out.println("User1(User2):" + Message);
notify();
}
}
class Producer implements Runnable
{
Q q;
Thread t;
Producer(Q q)
{
this.q=q;
t=new Thread(this, "Producer");
}
public void run()
{
int i = 0;
while (true)
{
if(i>5)
{
System.exit(0);
}
q.sendMessage(i++);
}
}
}
```

```
class Consumer implements Runnable
{
Q q;
Thread t;
Consumer (Q q)
{
this.q = q;
t=new Thread (this,"Consumer");
}
public void run ()
{
while(true)
{
q.recvMessage();
}
}
}
class Chat
{
public static void main(String args[])
{
Q q =new Q();
Producer p = new Producer (q);
Consumer c = new Consumer (q);
p.t.start();
c.t.start();
System.out.println("Press Control-C to stop. ");
}
}
```

Output:


Press Control-C to stop.
User1 :
Hi
User1(User2):Hi
User2 :
Hello
User2(User1):Hello
User1 :
R u planning to go any where to night
User1(User2):R u planning to go any where to night
User2 :
no plans.
User2(User1):no plans.
User1 :
can you join for a movie to night
User1(User2):can you join for a movie to night
User2 :
Yeah sure
User2(User1):Yeah sure
User1 :
Meet you tonight then
User1(User2):Meet you tonight then
User2 :
ok
User2(User1):ok