Lab 9

L1 slot

Valiveti manikanta bhuvanesh

19BCD7088

1. Apply Interthread communication to solve the Producer-Consumer problem with a common or shared bounded buffer(Queue) holding upto 5 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. A producer 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. When ever buffer is filled up and no more space to add the element into the queue(buffer) producerhas to wait until the buffer is emptied by consumer. When ever the buffer is empty and no more items are available for consumption the consumer should wait for producer to produceelements. Write a solution for N elements, where N is multiple of 5 or greater than 5 other than0

```
import java.util.*;

class Queue

{

int n;

boolean run = false;

synchronized int get()

{

while(!run)

try

{

wait();

}

catch(InterruptedException e)

{

System.out.println(e);

}

System.out.println("Produced " + n);

run=false;

notify();
```

```
return n;
}
synchronized void set(int n)
{
while(run)
try
{
wait();
}
catch(InterruptedException e)
{
System.out.println(e);
}
this.n = n;
run = true;
System.out.println("Consumed " + n);
notify();
}
}
class Producer implements Runnable
{
Queue q;
Thread t;
Producer(Queue q)
{
this.q=q;
t=new Thread(this, "Producer");
}
public void run()
{
int i = 0;
```

```
while (true)
{
if(i>5)
{
System.exit(0);
}
q.set(i++);
}
}
}
class Consumer implements Runnable
{
Queue q;
Thread t;
Consumer (Queue q)
{
this.q = q;
t=new Thread (this,"Consumer");
}
public void run ()
{
while(true)
{
q.get();
}
}
}
class Sell
{
public static void main(String args[])
{
```

Queue q =new Queue();

Producer p = new Producer (q);

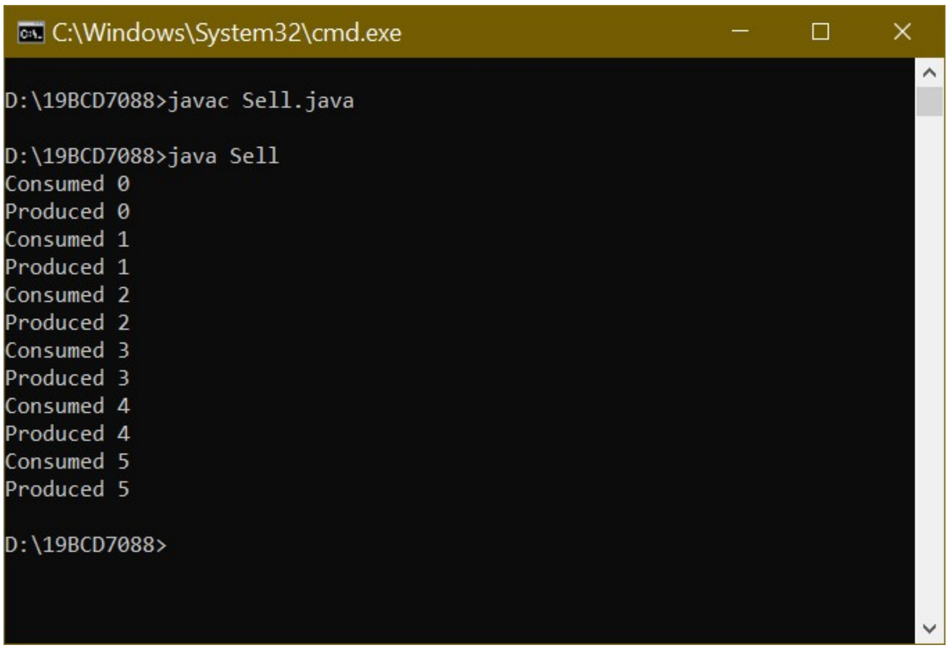
Consumer c = new Consumer (q);

p.t.start();

c.t.start();

}

}



```
C:\Windows\System32\cmd.exe

D:\19BCD7088>javac Sell.java

D:\19BCD7088>java Sell
Consumed 0
Produced 0
Consumed 1
Produced 1
Consumed 2
Produced 2
Consumed 3
Produced 3
Consumed 4
Produced 4
Consumed 5
Produced 5

D:\19BCD7088>
```

2. 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: Hi User2: Hello User1: R u preparing for exam? User2: Yes User1: Ok. Bye User2: Bye.

import java.util.*;

class Queue1

{

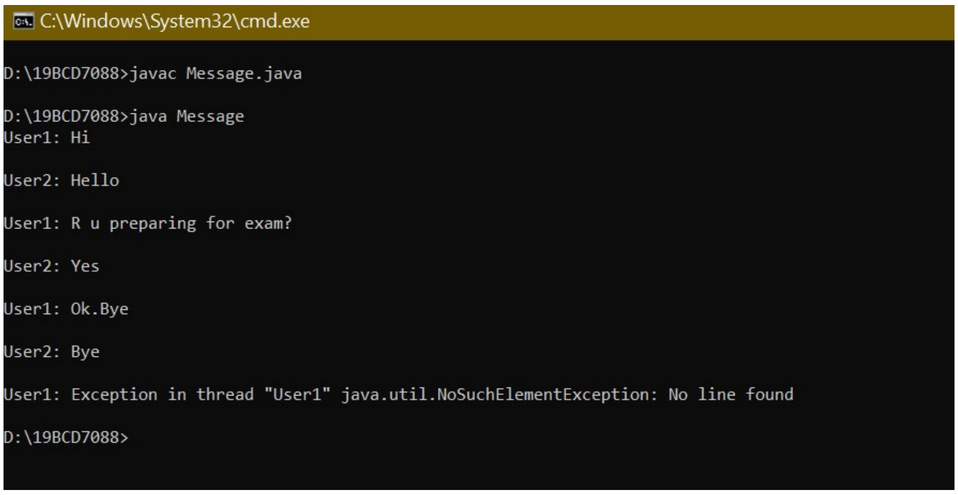
Scanner sc=new Scanner(System.in);

int n;

String msg;

```
boolean run = false;
synchronized int recvMessage()
{
while(!run)
 try
{
wait();
}
catch(InterruptedException e)
{
System.out.println(e);
}
System.out.print("User2: ");
msg=sc.nextLine();
run = true;
System.out.println();
run=false;
notify();
return n;
}
synchronized void sendMessage(int n)
{
while(run)
              try
{
wait();
}
catch(InterruptedException e)
{
System.out.println(e);
}
this.n = n;
System.out.print("User1: ");
msg=sc.nextLine();
run = true;
System.out.println();
notify();
}
}
class User1 implements Runnable
{
Queue1 que;
Thread t;
User1(Queue1 que)
{
this.que=que;
t=new Thread(this, "User1");
}
public void run()
{
int i = 0;
while (true)
{
if(i>5)
{
System.exit(0);
}
que.sendMessage(i++);
}
}
}
class User2 implements Runnable
```

```
{
Queue1 que;
Thread t;
User2 (Queue1 que)
{
this.que = que;
t=new Thread (this,"User2");
}
public void run ()
{
while(true)
{
que.recvMessage();
}
}
}
public class Message{
public static void main(String args[])
{
Queue1 que =new Queue1();
User1 u1 = new User1 (que);
User2 u2 = new User2 (que);
u1.t.start();
u2.t.start();
}
}
```


C:\Windows\System32\cmd.exe

```
D:\19BCD7088>javac Message.java

D:\19BCD7088>java Message
User1: Hi

User2: Hello

User1: R u preparing for exam?

User2: Yes

User1: Ok.Bye

User2: Bye

User1: Exception in thread "User1" java.util.NoSuchElementException: No line found

D:\19BCD7088>
```