Lab Program (10): Demonstrate Inter process communication and deadlock

i) Implementation of a producer and consumer

```
class Q {
int n;  boolean valueset = false;
synchronized int get() {
```

```
while(!valueSet)
try {
System.out.println("\n Consumer waiting");
wait();
}
catch(InterruptedException e) {
System.out.println("InterruptedException
caught");
}
System.out.println("Got:"+n);
valueSet = false;
System.out.println("\n Intimate Producer\n");
notify();
return n;
}
synchronized void put(int n)
{
while(valueSet)
try {
System.out.println("\n Producer waiting\n");
wait();
}
catch(InterruptedException e) {
System.out.println("InterruptedException
caught");
}
this.n=n;
valueSet=true;
System.out.println("Put:"+n);
System.out.println("\n Intimate Consumer
\n");
notify();
}
}
```

```
class Producer implements Runnable {
Q q;
Producer (Q q) {
this.q = q;
new Thread (this, "Producer").start();
}
public void run() {
int i=0;
while (i<15) {
q.put(i++);
}
}
}
class Consumer implements Runnable {
Q q;
Consumer(Q q) {
this.q=q;
new Thread (this, "Consumer").start();
}
public void run() {
int i=0;
while (i<15) {
int r = q.get();
System.out.println("consumed :"+r);
i++;
}
}
}
class PC Fixed {
public static void main (String args[]) {
Q q = new Q();
new Producer(q);
```

new Consumer(q);
System.out.println("Press Control-C to stop");
}
}

Output:

```
Press Control-C to stop
Put : 0

Intimate Consumer
Producer waiting
Got = 0

Intimate Producer
Put : 1

Intimate Consumer
Producer waiting
Consumed : 0
Got : 1

Intimate Producer
Consumed : 1
Put : 2
```

ii) Deadlock:

```
class A {
synchronized void foo(B b) {
String name = Thread.currentThread().
getName();
System.out.println(name + "entered A.foo");
```

```
try {
Thread.sleep(1000);
} catch (Exception e)
{
System.out.println("A Interrupted");
}
System.out.println(name + "trying to
call B.last()");}
b.last();
}
void last(){
System.out.println("Inside A.last");
}
}

class B{
synchronized void bar(A a)
{
String name = Thread.currentThread().
getName();
System.out.println(name + "entered B.bar");
try {
Thread.sleep(1000);
}
catch (Exception e){
System.out.println("B Interrupted");
}
System.out.println(name + "trying to
call A.last()");
a.last();
}
```

```
void last() {
System.out.println("Inside A.last");
}
}
class Deadlock implements Runnable
{
A a = new A();
B b = new B();
Deadlock() {
Thread.currentThread().setName("Main Thread");
Thread t = new thread(this, "RacingThread");
t.start();
a.foo(b);
System.out.println("Back in main thread");
}
public void run() {
b.bar(a);
System.out.println("Back in other thread");}
public static void main(String args[]) {
new Deadlock();
}}
```

Output:

```
RacingThread entered B.bar
Main Thread entered A.foo
RacingThread trying to call A.last()
Inside A.last
Back in otherthread
MainThread trying to call B.last()
Inside A.last
Back in mainthread
```

13.02.24