Date: / /

Lab Program - 10

ProCon.java:

```
class Q {
    int n;
    boolean valueSet = false;
    synchronized int get() {
        while (!valueSet) {
            try {
                System.out.println("Consumer Waiting");
                wait();
            } catch (InterruptedException e) {
                System.out.println(e);
            }
            System.out.println("Got: " + n);
            valueSet = false;
            System.out.println("Tell producer");
            notify();
        }
        return n;
    }
    synchronized void put(int n) {
        while (valueSet) {
            try {
                System.out.println("Producer waiting");
                wait();
            } catch (InterruptedException e) {
                System.out.println(e);
            }
            this.n = n;
            valueSet = true;
            System.out.println("Put: " + n);
            System.out.println("Tell Consumer");
            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 < 3) {
            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 < 3) {
            int r = q.get();
            System.out.println ("Consumed: " + r
            i++;
        }
    }
}

class ProCon {
    public static void main (String args[]) {
        Q q = new Q();
        new Producer (q); new Consumer (q);
    }
}
```

o/p
Put: 1
Got: 1
Put: 2
Got: 2
Put: 3
Got: 3

Deadlock.java:

```
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("MainThread");
        Thread t = new Thread(this, "RacingThread");
        t.start();
        a.foo(b);
        System.out.println("Back in main");
    }
    public void run(){
        b.bar(a);
        System.out.println("Back in other");
    }
    public static void main(String args[]){
        new Deadlock();
    }
}
```

Output:

```
MainThread entering A.foo
RacingThread entering B.bar
MainThread trying to call B.last()
Inside A.last
Back in main thread
RacingThread trying to call A.last()
```