Lab 10

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 {
    System synchronized void bar(A a) {
    String name = Thread.currentThread().
                        getName()
    System.out.println("B Interrupted");
}
    System.out.println(name + "trying to
                        call A.last()");
    a.last();
}
```

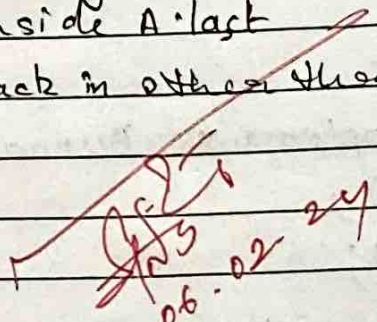
```
void last(){
    System.out.println("Inside A.last");
}
class Deadlock implements Runnable{
    A a = new A();
    B b = new B();

Deadlock(){
Thread.currentThread().setName("Main Thread");
Thread t = new Thread(this, "Racing Thread");
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 :-

```
MainThread entered A.foo
RacingThread entered B.bar
MainThread trying to call B.last()
In side A.last
```

Back in main thread
Racing Thread trying to call A.last()
Inside A.last
Back in other thread.



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 (value set) {
        try {
            system.out.println ("Producer waiting");
            wait ();
        }
        catch (Interrupted Exception e) {
            system.out.println (e);
        }
        this.n = n;
        value set = true;
        system.out.println ("Put : '+n');
        System.out.println ("Tell consumer");
        notify ();
    }
}
class Producer implements Runnable {
    Qq;
    Producer (Qq) {
        this.q = q;
        new Thread (this, "Producer").
                                    start();
    }
    public void run () {
        int i = 0
        while (i<3) {
            q.put (i++);
        }
    }
class consumer implements Runnable {
    Qq;
    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);
```

Output

```
Put : 1
Got : 1
Put : 2
Got : 2
Put : 3
Got : 3
```