

10) Demonstrate Inter Process Communication (IPC) & Deadlock

② IPC

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

```

System.out.println("Put:" + n);
System.out.println("Intimate consumer");
notify();
}

```

```

}
class Producer implements Runnable {

```

```

    Q q;

```

```

    Producer(Q q) {

```

```

        this.q = q;

```

```

        new Thread(this, "Procedure").start();

```

```

    }

```

```

    public void run() {

```

```

        int i = 0;

```

```

        while (i < 3) {

```

```

            q.put(i++);

```

```

        }

```

```

    }

```

```

}

```

```

class Consumer implements Runnable {

```

```

    Q q;

```

```

    Q q;

```

```

    Consumer(Q q) {

```

```

        this.q = q;

```

```

        new Thread(this, "Consumer").start();

```

```

    }

```

```

    public void run() {

```

```

        int i = 0;

```

```

        while (i < 3) {

```

```

            int x = q.get();

```

```

            System.out.println("Consumed: " + x);

```

```

            i++;

```

```

        }

```

```

    }

```



```

class PCFixed {
    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

```

put: 0
Intimate custom Consumer
producer waiting
got: 0
Intimate Producer
Put = 1
Intimate consumer
Producer waiting
consumed: 0
got: 1
Intimate Producer
consumed: 1
Put = 2
Intimate consumer
Producer waiting
consumed: 1
got: 2
Intimate Producer
put: 2
Intimate consumer
Producer waiting
consumed: 2

```

Date     /    /      
Page     

(b) 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");

}



```
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);
```

```
        thread.
```

```
        System.out.println("Back in main thread");
```

```
    }
```

```
public void run () {
```

```
    b.bar(a);
```

```
    thread.
```

```
    System.out.println("Back in other thread");
```

```
}
```

```
public static void main (String args[]) {
```

```
    new Deadlock();
```

```
}
```

```
}
```

O/P

Main Thread entered A.foo

RacingThread entered B.bar

Main Thread trying to call B.last()


Inside A.last

Back in main thread

RacingThread trying to call A.last()

Inside A.last

~~Back in other thread~~

  
13.02.24