Mangal
Date
Page

LAB-10

Demonstrate IPC & Deadlock

→ ~~Process cont~~

```
class Q {
  int n;
  boolean valueset = false;
  synchronized int get() {
    while (!valueset)
      try {
          System.out.println("\n consumer waiting\n");
          wait();
      }
    catch (InterruptedException e) {
        System.out.println(" InterruptedException
                caught");
    }
    System.out.println("Got: " + n);
    valueSet = false;
    System.out.println("\n Intimate procedure \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 Produces implements Runnable {
  Q q;
  produces(Q q) {
    this.q=q;
    new Thread(this, "Produces").start();
  }
  public void run() {
    int i=0;
    while(i<15) {
      q.put(i++);
    }
  }
}
class Consumes 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 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");
    }
}
```

→ code for 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("MainThread");
    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();
}
}
```

o/p

Press Control-C to stop
Put: 0
Intimate consumer
Producer waiting
Got: 0
Intimate Producer
Put: 1
Intimate consumer
Producer waiting
Consumed: 0
Got: 0
Consumed: 1
Put: 2
Intimate Producer
Consumed: 2
Put: 3
Intimate consumer
Producer waiting
Got: 3

Deadlock:
Racing Thread entered B.bar
Main thread entered A.foo
Racing thread trying to call A.last
Main thread trying to call B.last

4/12/24