LAB-10

Bafna Gold
Date: Page:

c) demonstrate inter communication and deadlock.

```
class Q {
        int n;
        boolean valueset = False;
        synchronised int get() {
                while (!valueset) {
                try {
                        System.out.println("\n
                            Consumer Waiting \n");
                        wait();
                    }
                catch(InterruptException e)
                {
                    System.out.println("Interrupt
                        Exception caught");
                }
                System.out.println("Got :" + n);
                value set = False;
                System.out.println ("Intimate Producer\n");
                notify();
                return n;
        }
        synchronised void put(int n) {
              while (valueset)
                try {
                    System.out.println(
                        "Producer Waiting");
                wait();
                }
                catch(InterruptException e)
        System.out.println("
                InterruptedException Caught");
        }
        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, "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 PCFixed{
    public static void main(string args[])
    {
        Q q= new Q;
        new Producer(q);
        new Consumer(q);
    }
}
```

Output:

```
Put : 0
Intimate Consumer
Producer Waiting
Got : 0
Intimate Producer
Consumed : 0
Consumer Waiting
Put : 1
Intimate Consumer
Producer Waiting
Got : 1
Intimate Producer
Consumed : 1
Put : 2
Intimate Consumer
Producer Waiting
Got : 2
Intimate Consumer
Consumed : 2
~~Producer Waiting~~
Put : 3
Intimate consumer
Producer Waiting
Got : 3
Intimate Consumer
Consumed : 3
Put : 4
```

Deadlocking

```
class A {
      synchronised 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");
  }
```