6/2/24

LAB-10

Demonstrate inter communication and deadlock

```
class Q {
    int n;
    boolean valueset = false;
    synchronize int get() {
    while (!valueset)
      try {
      System.out.println("\nConsumer waiting\n");
      wait();
    } catch (InterruptedException e) {
    System.out.println("InterruptException caught");
    }
    System.out.println("Got : " +n);
     valueset = false;
    System.out.println("\nIntimate producer\n");
     notify();
     return n;
     }
    synchronized void put(int n) {
     while (valueset)
      try {
      System.out.println("\nProducer waiting\n");
      wait();
      } catch (InterruptedException e) {
      System.out.println("InterruptException caught");
      }
     this.n = n;
      valueset = true;
      System.out.println("Put: " +n);
       System.out.println("\nIntimate Consume\n");
      notify();
}
}
class producer implements Runnable{
 Q q;
 producer (Q q){
  this.q = q;
  new Threads (this, "Producer").start();
 }
 public void run(){
  int i=0;
  while (i<15){
    q.put(i++);
  }
 }
}
class consume 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("Consume : "+r);
 i++;
 }
}
}
```

```
class PC fixed {
  public static void main (String args[]) {
  Q q = new Q();
    new producer (q);
    new consume (q);
    System.out.println ("Press control-C to stop");
```

output:-

```
put 0
intimate consume
Producer waiting
 got:0
Intimate Produces
consmed:0
consume waiting
 Put: 1
Intimate Consumer
 Producer waiting
 Got: 1
Intiamate produce
consumed: 1
Put: 2
Intimate Consume
Producer waiting
Got: 2
Put: 3
Intimed consumed
Producer waiting
Got: 3
Intimate consumed
consumed: 3
Put: 4
```

Shreyas.k
1BM22CS271