URBAN EDGE 13/02/24

# Lab week-10A

Demonstrate inter process communication and deadlock

```
class Q
{
  int n;
  boolean ValueSet = false;
  Synchronized int get() {
    while (!ValueSet)
      try {
      System.out.println("\nConsumer waiting\n");
      wait();
      } catch (InterruptedException e) {
          System.out.println("InterruptedException caught");
      }
      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("InterruptedException caught");
  }
  this.n = n;
  ValueSet = true;
  System.out.println("Put: " + n);
  System.out.println("\nIntimate Consumer\n");
  notify();
  }
}
```

URBAN EDGE 13/02/24

```
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<5) {
  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<5) {
  int r = q.get();
  System.out.println("consumed: " + r);
  i++;
  }
  }
}
```

URBAN EDGE 13/02/24

```
class PCFixed {
    public static void main (String args[]) {
    Q q = new Q();
    new Produce (q);
    new Consumer (q);
    System.out.println ("Press Control-C to stop.");
}
```

O/p:

Press Control-C to stop.
Put : 0

Intimate Consumer
Produce waiting
Got : 0
Intimate Produce
Put : 1

Intimate Consumer
Produce waiting
Consumed : 0
Got : 1

Intimate produce
Consumed: 1
Put : 2
Intimate Consumer
Produce waiting
Got : 2
Intimate Produce
Consumed : 2
Put : 3