Page No.
Date. / /

13/2/24

13 (i) (IPC)

(ii)
```
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 Ultimate Producer \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("Interrupted Exception caught");
  }
  this.n=n;
  valueset=true;
  System.out.println("Put: "+n);
  System.out.println("\n Ultimate Consumer\n");
  notify();
}
```

Date. / /

```
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++;
    }
  }
}
```

Page No.
Date. / /

```
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:

Press control c to stop
Put: 0
Intimate 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
got: 2
Intimate producer
Consumed: 2
put: 3
Intimate Consumer
Producer waiting
got: 3
Intimate Producer
Consumed: 3
put: 4
Intimate Consumer
got: 4
Intimate Producer
Consumed: 4

Page No.
Date. / /

(ii) (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");
  }
}
```

Page No.
Date. / /

```
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); // get lock on a in this thread.
        System.out.println("Back in main thread");
    }
    public void run() {
        b.bar(a); // get lock on b in other thread.
        System.out.println("Back in other thread");
    }
    public static void main(String args[]) {
        new Deadlock();
    }
}
```

Output:

```
MainThread entered A.foo
RacingThread entered B.bar
MainThread 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
```