Lab 10

13/03/2023

9 Demonstrate Inter Process Communication and Deadlock

```
class Q {
    int n;
    boolean valueSet = false;

    synchronized int get() {
        while (!valueSet) {
            try {
                System.out.println("\n Consumer Waiting");
                wait();
            } catch (InterruptedException e) {
                System.out.println("Interrupted
                Exception
                caught");
            }
        }
        System.out.println("Got: " + n);
        valueSet = false;
        System.out.println("\n Notifying 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("InterruptedException
                caught");
            }
        }
```

```
this.n = n;
valueSet = true;
System.out.println("Put: " + n);
System.out.println("\n Notifying consumer\n");
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 < 6) {
            q.put(i++);
        }
    }
}

class Consumer implements Runnable {
    Q q;
    Producer(Q q) {
        this.q = q;
        new Thread(this, "Producer").start();
    }
    public void run() {
        int i = 0
        while(i < 6) {
            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("Consumed: " + i);
        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.");
    }
}
```

→ Output

```
Press Control-C to stop.
Put 0
Notifying Consumer
Producer waiting
Got: 0
Notifying Producer
Put: 1
Notifying Consumer
Producer waiting
```

consumed: 0

Got: 1

Notifying Consumer

~~Producer waiting~~

~~Got: 1~~

consumed: 1

Put: 2

Notifying Consumer

Producer waiting

Got: 2

Notifying Producer

consumer: 2

~~Produ~~ Put: 3

Notifying Consumer

Producer Waiting

Got: 3

Notifying Producer

consumed: 3

Put: 4

Notifying Consumer

Producer Waiting

Got: 4

Notifying Producer

consumed: 4

Put: 5

Notifying Consumer

Got: 5

Notified Producer

consumed: 5