LAB - 10

Sagar . I . Banjan 4C
1BM22CS233

Inter process communication

```
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("Interrupted Exception
                    caught");
            }
        System.out.println("Got: " + n);
        valueSet = false;
        System.out.println("\n Intimate 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 Intimate 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<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++;
        }
    }
}

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:-

```
put:0
Press control-c to stop
Intimate Consumer
Producer waiting
Got:0
Intimate Producer
consumed:0
put:1
Intimate consumer
Producer waiting
Got:1
Intimate Producer
consumed:1
put:2
  :
  :
```

Dead lock

```
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");
    }
}
class Deadlock implements Runnable {
        A a = new A();
        B b = new B();
        Deadlock(){
            Thread.currentThread().setName("Main Thread");
            Thread t = new Thread(this, "Racing Thread");
            t.start();
            a.foo(b);
            System.out.println("Back in main thread");
        }
        public void run(){
            b.bar(a);
            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

DOI
13-2-24