PAGE NO:
DATE:

Demonstrate Inter process communication and deadlock.

class Q{ (IPC)
int n;
boolean valueSet = false;
synchronized int get(){
while(!valueSet)
try{
System.out.println("consume waiting\n");
wait();
}catch(InterruptedException e){
System.out.println("Interrupted Exception caught");
}
System.out.println("Interrupted Exception 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(InterruptException e){
System.out.println("Interrupted Exception caught");
}
this.n=n;
valueSet = true;
System.out.println("Put:"+n);
System.out.println("\nIntimate 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;
    Consumer(Q q)
    {
        this.q = q;
        new Thread(this, "Consumer").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("Press Control-C to stop.");
    }
}
```

Output:

Press Control-C to stop
Put : 0
Intimate Consumer
Producer waiting
Got : 0
Intimate Producer
consumed : 0
Put : 1
Intimate Producer
consumed : 0
Put : 1
Intimate cosumer
Producer waiting
Got : 1
Intimate Producer
consumed : 2
Put : 2
Intimate Consumer
Producer waiting
Got : 2
Intimate Producer
consumed : 2

Put : 3
Intimate Producer
consumed : 2
Put : 3
Intimate Consumer
Producer waiting
Got : 3
Intimate Producer
consumed : 3
Put : 4
Intimate consumer
Producer waiting
Got : 4
Intimate Producer
consumed : 4
Put : 5
Intimate consumer
Producer waiting
Got : 5
Intimate Producer
consumed : 5
Put : 6
Intimate consumer
Producer waiting
Got : 6

PAGE NO:
DATE:

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("Insid 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("MainThread");
        Thread t = new Thread(this, "RacingThread");
        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
Main Thread trying to call B.last()
Inside A.last
Back in main thread
RacingThread trying to call A.last()
Inside A.last
```

PAGE N
DATE :

Back in other thread

13/9/2024