Demonstrate inter process Communication and deadlock

Inter Process Communication

class Q

{

int n;

boolean valueSet = false;

synchronized int get()

{ while (!valueSet)

try

{

System.out.println("Consumer waiting");

wait();

} catch (InterruptedException e)

{

System.out.println("InterruptedException caught");

}

System.out.println("Got: "+n);

valueSet = false;

System.out.println("Intimate Producer");

notify();

return n;

}

Synchronized void put(int n)

{

while (valueSet)

try

{

```
System.out.println("Producer waiting");
wait();
} catch(InterruptedException e)
{
 System.out.println("Interrupted Exception caught");
}

this.n = n;
valueSet = true;
System.out.println("Put: " + n);
System.out.println("Intimate Consumer");
notify();
}
}

// Producer class
class Producer implements Runnable
{
 Q q;
 Producer(Q q)
 {
  this.q = q;
  new Thread(this, "Producer").start();
 }
public void run()
 {
```

```
t i=0;
hile (i<=5)

{
 q.put(i++);

}
}
}
```

```
// Consumer class
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.");
        System.out.println("Sneha N Shastri - 1BM22CS283");
    }
}
```

Output:

```
Press Control-C to stop.
Sneha N Shastri - 1BM22CS283
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
Producer waiting
Got : 4
Intimate Producer
Consumed : 4
Put : 5
Intimate Consumer
Got : 5
Intimate Producer
Consumed : 5

b. 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");
 }
}

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");
}  <- System.out.println("Sneha N Shastri-1BM22CS283");

public static void main(String args[])
{
  new Deadlock();
}
}
```

Output:

```
RacingThread entered B.bar
MainThread entered A.foo
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
Sneha N Shastri
-1BM22CS283
```

13.02.24