WEEK 10

Demonstrate Inter process Communication and deadlock

Pegm 10

Demonstrate Inter Process Communication and deadlock.

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

    synchronized int get () {
        while(! valueSet )
        try {
            System.out.println("In Consumer waiting\n");

            wait ();
        }
        catch (InterruptedException e) {
        System.out.println( "Interrupted Excep. caught " );
        }
        System.out.println(" Got : " +n);
        valueSet = false;
        System.out.println(" In Intimate Producer \n");
        notify();
        return n;
    }
    synchronized void put(int n) {
        while(valueSet)
        try {
            System.out.println(" In Producer Wait
            wait();
        }
```

```
        catch (InterruptedException e) {
            System.out.println("Interrupted Exception
                                caught");
        }
        this.n = n;
        valueSet = true;
        System.out.println("Put: " + n);
        System.out.println("In Intimate Consumer");
        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<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++;
    }
}
}
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");
    }
}
```

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

synchronized int get() {
while(!valueSet)
try {
System.out.println("\nConsumer waiting\n");
wait();
} catch(InterruptedException e) {
System.out.println("InterruptedException caught");
}
System.out.println("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(InterruptedException e) {
System.out.println("InterruptedException 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<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++;
}
}
}

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.");
}
}
```

```
C:\Windows\System32\cmd.exe
C:\Users\Srinivas\OneDrive\Desktop\BMS\SEM 3\PROGRAMS>javac PCFixed.java

C:\Users\Srinivas\OneDrive\Desktop\BMS\SEM 3\PROGRAMS>java PCFixed
Press Control-C to stop.
Put: 0

Intimate Consumer


Producer waiting

Got: 0

Intimate Producer

Put: 1

Intimate Consumer

consumed:0
Got: 1
```

```
C:\Windows\System32\cmd.exe

Intimate Producer

Put: 2
consumed:1

Intimate Consumer


Producer waiting

Got: 2

Intimate Producer

Put: 3

Intimate Consumer

consumed:2

Producer waiting
```

```
C:\Windows\System32\cmd.exe
Got: 3

Intimate Producer

Put: 4

Intimate Consumer

consumed:3

Producer waiting

Got: 4

Intimate Producer

consumed:4
Put: 5

Intimate Consumer
```

```
C:\Windows\System32\cmd.exe
Producer waiting

Got: 5

Intimate Producer

Put: 6

Intimate Consumer

consumed:5

Producer waiting

Got: 6

Intimate Producer

Put: 7

Intimate Consumer
```

```
C:\Windows\System32\cmd.exe
Put: 13

Intimate Consumer


Producer waiting

Got: 13

Intimate Producer

consumed:13
Put: 14

Intimate Consumer

Got: 14

Intimate Producer

consumed:14
```

```
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(); }
        synchronized 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");
        }
        S.O.Pln(name + "trying to call A.last()");
        a.last(); }
        synchronized void last() {
            S.O.Pln(" Inside A.last"); } }
```

```
class Deadlock implements Runnable
{
    Aa = new A();
    Bb = new B();
    Deadlock () {
        Thread.currentThread().setName("MainTh
        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();
    }
}
```

```
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(); }
     synchronized 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(); }
synchronized 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); // 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(); }
}
```