

Open ended Exercise.

- * Demonstrate Inter process communication and deadlock.

i) IPC program

Class of

Int n;

boolean valueSet = false;

Synchronized int get() {

while (!valueSet)

{

System.out.println("In consumer waiting\n");

wait(1);

}

catch (InterruptedException e) {

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

}

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

valueSet = true;

System.out.println("Intimate producer\n");

notify();

```
return n;  
}  
synchronized void put(int n) {  
    while (valueSet)  
        try {  
            System.out.println("/n producer waiting  
wait 1);  
        } catch (InterruptedException e) {  
            System.out.println("InterruptedException  
caught");  
        }  
        this.n = n;  
        valueSet = true;  
        System.out.println("put: " + n);  
        System.out.println("/n 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 k = q.get();
            System.out.println("consumer : " + k);
            i++;
        }
    }
}

```

```

class ipc {
    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:

Intimate consumer
producer waiting

press control-c to stop
Got: 0

Intimate producer
consumed: 0
put: 1

~~Intimate consumer
producer waiting
Got: 1~~

Intimate producer
consumed: 1
put: 2

Intimate producer

~~consumes: 2~~

producer waiting

Got: 2

Intimate producer

consumes: 2

put: 3

Intimate consumer

producer waiting

Got: 3

Intimate ~~consumer~~ producer

~~Got~~ consumes: 3

put: 4

~~Intimate consumer~~

~~producer waiting~~

Got: 4

Intimate consumer

consumes: 4

put: 5

2.9i) Deadlock program

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(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 + " thing 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();
```

```
Dead lock() {
```

```
Thread.currentThread().setName("main  
Thread");
```

```
Thread t = new Thread(this, "Racing  
Thread");
```


```
t.start();
```

```
a.foo(b); // get lock on a in this  
thread.
```

```
public void run() {
    b.bar(a);
}
```

```
System.out.println("Back in other  
thread");
```

```
}  
public static void main(String args[]) {  
    new Deadlock1();  
}
```

Output 

Main Thread entered A.foo

Racing Thread entered B.bar

Main Thread trying to call B.last()

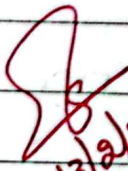
Inside A.last.

Back in main thread.

~~Racing Thread trying to call A.last()~~

~~Inside A.last.~~

~~Back in other thread.~~


13/2/2024