

06/2/24

WEEK-10

+ Demonstrate Inter process communication & deadlock

```
public class DeadlockExample {
    final sharedResource sharedResource = new
    sharedResource();
```

```
    thread process 1 = new Thread() -> {
        try {
            sharedResource.method 1();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
```

```
};
```

```
    thread process 2 = new Thread() -> {
        try {
            sharedResource.method 2();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
```

```
};
```

```
process 1.start();
process 2.start();
```

```
?
```



```
class sharedresources {
```

```
    private final object lock1 = new object();
    private final object lock2 = new object();
```

```
    public void method1() throws InterruptedException {
        synchronized (lock1) {
            System.out.println("Method 1 acquired lock 1");
        }
    }
```

}

}

```
    public void method2() throws InterruptedException {
        synchronized (lock2) {
            System.out.println("Method 2 acquired lock 2");
            Thread.sleep(1000);
        }
    }
```

```
    synchronized (lock1) {
```

```
        System.out.println("Method 2 acquired lock 1");
    }
```

}

}

}

}

Output:-

Method 1 acquired lock 1

Method 2 acquired lock 2



Output:- Main thread entered A.foo  
Racing thread entered B.foo  
Main thread trying to call B.last()  
Inside A.last  
Back in Main thread  
Racing thread trying to call A.last()  
Inside B.last  
Back in other thread.

SFF  
123  
66.02.14



## Program - 10

+ Inter process communication

class Q {

int m;

boolean valueSet = false;

synchronized int get()

{

while (!valueSet)

try {

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

wait();

} catch (InterruptedException e)

{

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

}

System.out.println("got: " + m);

valueSet = false;

System.out.println("\n Enterate Produce!");

notify();

return m;

}



```
synchronized void put(int m) {
    while (value set)
```

```
    try {
```

```
        System.out.println("1m Producer waiting\n");
        wait();
```

```
    }
```

```
    catch (InterruptedException e)
```

```
    {
```

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

```
    }
```

```
    this.m = m;
```

```
    value set = true;
```

```
    System.out.println("1m Enterate Consumer\n");
    notify();
```

```
    }
```

```
}
```

```
class Producer implements Runnable
{
```

```
    a, q;
```

```
    produces(a, 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 {  
    {
```

```
        a, q;
```

```
    consumer(a, q) {
```

```
        this.q = q;
```

```
        new Thread (this, "Consumer").start();
```

```
    }
```

```
    public void run()
```

```
    {  
        int i = 0;
```

```
        while (i < 15) {
```

```
            int s = q.get();
```

```
            System.out.println("Consumed!" + s);
```

```
            i++;
```

```
        }
```

```
    }
```



class pfixed {

public static void main (String args[])

{

new producer(q);

new consumer(q);

System.out.println ("Press control-c to stop.");

}

Output :-

Put : 1

Get : 1

Put : 2

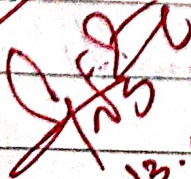
Get : 2

Put : 3

Get : 3

Put : 4

Get : 4

  
13.02.2024