

106 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("MainThread");
```

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

```
        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

PagingThread entered B.bar

PagingThread trying to call A.last()

MainThread trying to call B.last()

Inside A.last

Inside A.last

Back in other thread

Back in main thread

Sreyas Rao

13/08/24 BM22CS222