

**Name: -Shubham B.
Pasale**

1. Create a Deadlock class to demonstrate deadlock in multithreading environment

```
package shubham;

public class Assignment_8 {

    public static void main(String[] args) {

        final String resource1 = "Shubham Pasale";

        final String resource2 = "Rahul Mhetre";

        Thread t1 = new Thread() {

            public void run() {

                synchronized (resource1) {

                    System.out.println("Thread 1: locked resource 1");

                    try { Thread.sleep(100); } catch (Exception e) {}

                    synchronized (resource2) {

                        System.out.println("Thread 1: locked resource 2");

                    }

                }

            }

        };

        Thread t2 = new Thread() {

            public void run() {

                synchronized (resource2) {

                    System.out.println("Thread 2: locked resource 2");

                    try { Thread.sleep(100); } catch (Exception e) {}

                    synchronized (resource1) {

                        System.out.println("Thread 2: locked resource 1");
```

```

        }
    }
}
};
t1.start();
t2.start();
}
}

```

Output:-

Thread 1: locked resource 1

Thread 2: locked resource 2

2. Create multiple threads using anonymous inner classes

```
package shubham;
```

```
import java.util.concurrent.ExecutorService;
```

```
import java.util.concurrent.Executors;
```

```
public class Assignment_8 {
```

```
    public static void main(String[] args)
```

```
{
```

```
    new Assignment_6().startThreads();
```

```
}
```

```
    private void startThreads()
```

```
{
```

```
ExecutorService taskList
= Executors.newFixedThreadPool(2);
taskList.execute(new InnerClass(1));
taskList.execute(new InnerClass(2));
taskList.execute(new InnerClass(3));
taskList.execute(new InnerClass(4));
taskList.execute(new InnerClass(5));
taskList.shutdown();
}

private void pause(double seconds)
{
    try {
        Thread.sleep(Math.round(1000.0 * seconds));
    }
    catch (InterruptedException e) {
        e.printStackTrace();
    }
}

// Inner Class

public class InnerClass implements Runnable {

    private int loopLimit;

    InnerClass(int loopLimit)
    {
        this.loopLimit = loopLimit;
    }
}
```

```

public void run()
{
    for (int i = 0; i < loopLimit; i++) {
        System.out.println(
            Thread.currentThread().getName()
            + " Counter: " + i);
        pause(Math.random());
    } }
}

```

Output:-

```

pool-1-thread-1 Counter: 0
pool-1-thread-2 Counter: 0
pool-1-thread-2 Counter: 1
pool-1-thread-1 Counter: 0
pool-1-thread-2 Counter: 0
pool-1-thread-1 Counter: 1
pool-1-thread-1 Counter: 2
pool-1-thread-2 Counter: 1
pool-1-thread-2 Counter: 2
pool-1-thread-2 Counter: 3
pool-1-thread-1 Counter: 0
pool-1-thread-1 Counter: 1
pool-1-thread-1 Counter: 2
pool-1-thread-1 Counter: 3
pool-1-thread-1 Counter: 4

```