

## LAB # 6

### Deadlock in concurrency:

#### OBJECTIVE:

Implementing multiple thread blocked resources with help of lock and deadlock conditions.

#### Lab Task:

Create three threads by implementing thread synchronization block through 3 locks. (Hint: Apply un-sequenced lock to analyze deadlock and solve it through provided solution:

#### Code:

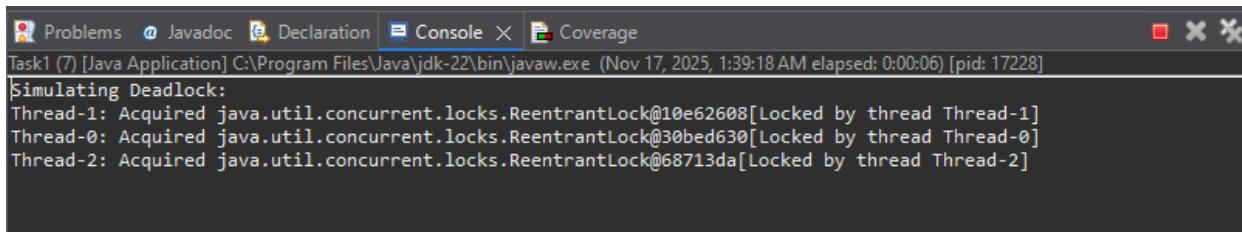
```

1 package Lab6;
2 import java.util.concurrent.locks.Lock;
3 import java.util.concurrent.locks.ReentrantLock;
4 public class Task1 {
5     static final Lock lock1 = new ReentrantLock();
6     static final Lock lock2 = new ReentrantLock();
7     static final Lock lock3 = new ReentrantLock();
8     static class DeadlockThread extends Thread {
9         private final Lock firstLock, secondLock;
10        DeadlockThread(Lock firstLock, Lock secondLock) {
11            this.firstLock = firstLock;
12            this.secondLock = secondLock;}
13        public void run() {try {
14            firstLock.lock();
15            System.out.println(Thread.currentThread().getName() + ": Acquired " + firstLock);
16            Thread.sleep(100);
17            secondLock.lock();
18            System.out.println(Thread.currentThread().getName() + ": Acquired " + secondLock);
19            Thread.sleep(100);} catch (InterruptedException e) {
20                e.printStackTrace();} finally {
21                unlock(firstLock);
22                unlock(secondLock);}
23        private void unlock(Lock lock) {
24            if (lock instanceof ReentrantLock && ((ReentrantLock) lock).isHeldByCurrentThread()) {
25                lock.unlock();}
26        static class ResolvedThread extends Thread {
27            private final Lock[] locks;
28            ResolvedThread(Lock... locks) {
29                this.locks = locks;}
30        public void run() {try {
31            for (Lock lock : locks) {
32                lock.lock();
33                System.out.println(Thread.currentThread().getName() + ": Acquired " + lock);
34                Thread.sleep(50);}
35                Thread.sleep(100);} catch (InterruptedException e) {
36                    e.printStackTrace();} finally {
37                    for (int i = locks.length - 1; i >= 0; i--) {
38                        unlock(locks[i]);}}
39        private void unlock(Lock lock) {
40            if (lock instanceof ReentrantLock && ((ReentrantLock) lock).isHeldByCurrentThread()) {
41                lock.unlock();}
42        static class TryLockThread extends Thread {
43            private final Lock[] locks;
44            TryLockThread(Lock... locks) {
45                this.locks = locks;}
46        public void run() {try {
47            if (tryLockAll()) {
48                System.out.println(Thread.currentThread().getName() + ": Acquired all locks");
49                Thread.sleep(100); } else {
50                    System.out.println(Thread.currentThread().getName() + ": Could not acquire all locks,
51                } catch (InterruptedException e) {
52                    e.printStackTrace();} finally {
53                        unlockAll();}
54        private boolean tryLockAll() {
55            for (Lock lock : locks) {
56                if (!lock.tryLock()) return false;}
57                return true;}
58        private void unlockAll() {
59            for (Lock lock : locks) {
60                unlock(lock);}
61        private void unlock(Lock lock) {
62            if (lock instanceof ReentrantLock && ((ReentrantLock) lock).isHeldByCurrentThread()) {
63                lock.unlock();}
64        public static void main(String[] args) throws InterruptedException {
65            System.out.println("Simulating Deadlock:");
66            Thread t1 = new DeadlockThread(lock1, lock2);
67            Thread t2 = new DeadlockThread(lock2, lock3);
68            Thread t3 = new DeadlockThread(lock3, lock1);
}

```

```
54     private boolean tryLockAll() {
55         for (Lock lock : locks) {
56             if (!lock.tryLock()) return false;}
57         return true;}
58     private void unlockAll() {
59         for (Lock lock : locks) {
60             unlock(lock);}}
61     private void unlock(Lock lock) {
62         if (lock instanceof ReentrantLock && ((ReentrantLock) lock).isHeldByCurrentThread()) {
63             lock.unlock();}}
64     public static void main(String[] args) throws InterruptedException {
65         System.out.println("Simulating Deadlock:");
66         Thread t1 = new DeadlockThread(lock1, lock2);
67         Thread t2 = new DeadlockThread(lock2, lock3);
68         Thread t3 = new DeadlockThread(lock3, lock1);
69         t1.start(); t2.start(); t3.start();
70         t1.join(); t2.join(); t3.join();
71         System.out.println("\nSolving Deadlock with Consistent Lock Order:");
72         Thread t1Resolved = new ResolvedThread(lock1, lock2, lock3);
73         Thread t2Resolved = new ResolvedThread(lock1, lock2, lock3);
74         Thread t3Resolved = new ResolvedThread(lock1, lock2, lock3);
75         t1Resolved.start(); t2Resolved.start(); t3Resolved.start();
76         t1Resolved.join(); t2Resolved.join(); t3Resolved.join();
77         System.out.println("\nSolving Deadlock with tryLock():");
78         Thread t1TryLock = new TryLockThread(lock1, lock2, lock3);
79         Thread t2TryLock = new TryLockThread(lock1, lock2, lock3);
80         Thread t3TryLock = new TryLockThread(lock1, lock2, lock3);
81         t1TryLock.start(); t2TryLock.start(); t3TryLock.start();
82         t1TryLock.join(); t2TryLock.join(); t3TryLock.join();}
```

### Output:



The screenshot shows an IDE's console tab with the following output:

```
Problems Javadoc Declaration Console Coverage
Task1 (7) [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe (Nov 17, 2025, 1:39:18 AM elapsed: 0:00:06) [pid: 17228]
Simulating Deadlock:
Thread-1: Acquired java.util.concurrent.locks.ReentrantLock@10e62608[Locked by thread Thread-1]
Thread-0: Acquired java.util.concurrent.locks.ReentrantLock@30bed630[Locked by thread Thread-0]
Thread-2: Acquired java.util.concurrent.locks.ReentrantLock@68713da[Locked by thread Thread-2]
```