

## 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:

```

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

```

```

54 private boolean tryLockAll() {
55     for (Lock lock : locks) {
56         if (!lock.tryLock()) return false;
57     }
58     return true;
59 }
60 private void unlockAll() {
61     for (Lock lock : locks) {
62         unlock(lock);
63     }
64 }
65 private void unlock(Lock lock) {
66     if (lock instanceof ReentrantLock && ((ReentrantLock) lock).isHeldByCurrentThread()) {
67         lock.unlock();
68     }
69 }
70 public static void main(String[] args) throws InterruptedException {
71     System.out.println("Simulating Deadlock:");
72     Thread t1 = new DeadlockThread(lock1, lock2);
73     Thread t2 = new DeadlockThread(lock2, lock3);
74     Thread t3 = new DeadlockThread(lock3, lock1);
75     t1.start(); t2.start(); t3.start();
76     t1.join(); t2.join(); t3.join();
77     System.out.println("\nSolving Deadlock with Consistent Lock Order:");
78     Thread t1Resolved = new ResolvedThread(lock1, lock2, lock3);
79     Thread t2Resolved = new ResolvedThread(lock1, lock2, lock3);
80     Thread t3Resolved = new ResolvedThread(lock1, lock2, lock3);
81     t1Resolved.start(); t2Resolved.start(); t3Resolved.start();
82     t1Resolved.join(); t2Resolved.join(); t3Resolved.join();
83     System.out.println("\nSolving Deadlock with tryLock():");
84     Thread t1TryLock = new TryLockThread(lock1, lock2, lock3);
85     Thread t2TryLock = new TryLockThread(lock1, lock2, lock3);
86     Thread t3TryLock = new TryLockThread(lock1, lock2, lock3);
87     t1TryLock.start(); t2TryLock.start(); t3TryLock.start();
88     t1TryLock.join(); t2TryLock.join(); t3TryLock.join();
89 }

```

### Output:

```

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]

```