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:

```
   private static class ThreadDemo1 extends Thread {
      public void run() {
         synchronized (Lock1) {
            System.out.println("Thread 1: Holding lock 1...");

            try { Thread.sleep(10); }
            catch (InterruptedException e) {}
            System.out.println("Thread 1: Waiting for lock 2...");

            synchronized (Lock2) {
               System.out.println("Thread 1: Holding lock 1 & 2...");

               try { Thread.sleep(10); }
                catch (InterruptedException e) {}
                System.out.println("Thread 1: Waiting for lock 3...");

                synchronized (Lock3) {
                   System.out.println("Thread 1: Holding lock 1 , 2 & 3...");
                }
            }
         }
      }
   }

private static class ThreadDemo2 extends Thread {
   public void run() {
       synchronized (Lock1) {
            System.out.println("Thread 2: Holding lock 1...");

            try { Thread.sleep(10); }
            catch (InterruptedException e) {}
            System.out.println("Thread 2: Waiting for lock 2...");

            synchronized (Lock2) {
               System.out.println("Thread 2: Holding lock 1 & 2...");

               try { Thread.sleep(10); }
                catch (InterruptedException e) {}
                System.out.println("Thread 2: Waiting for lock 3...");

                synchronized (Lock3) {
                   System.out.println("Thread 2: Holding lock 1 , 2 & 3...");
                }
         }
      }
   }
}
```

```
private static class ThreadDemo3 extends Thread {
       public void run() {
           synchronized (Lock1) {
                 System.out.println("Thread 3: Holding lock 1...");

                 try { Thread.sleep(10); }
                 catch (InterruptedException e) {}
                 System.out.println("Thread 3: Waiting for lock 2...");

                 synchronized (Lock2) {
                    System.out.println("Thread 3: Holding lock 1 & 2...");

                    try { Thread.sleep(10); }
                     catch (InterruptedException e) {}
                     System.out.println("Thread 3: Waiting for lock 3...");

                     synchronized (Lock3) {
                        System.out.println("Thread 3: Holding lock 1 , 2 & 3...");
                 }
              }
           }
       }
    }
```

```
public class deadlock {
    public static Object Lock1 = new Object();
       public static Object Lock2 = new Object();
       public static Object Lock3 = new Object();

       public static void main(String args[]) {
          ThreadDemo1 T1 = new ThreadDemo1();
          ThreadDemo2 T2 = new ThreadDemo2();
          ThreadDemo3 T3 = new ThreadDemo3();
          T1.start();
          T3.start();
          T2.start();
       }
```

```
Thread 1: Holding lock 1...
Thread 1: Waiting for lock 2...
Thread 1: Holding lock 1 & 2...
Thread 1: Waiting for lock 3...
Thread 1: Holding lock 1 , 2 & 3...
Thread 2: Holding lock 1...
Thread 2: Waiting for lock 2...
Thread 2: Holding lock 1 & 2...
Thread 2: Waiting for lock 3...
Thread 2: Holding lock 1 , 2 & 3...
Thread 3: Holding lock 1...
Thread 3: Waiting for lock 2...
Thread 3: Holding lock 1 & 2...
Thread 3: Waiting for lock 3...
Thread 3: Holding lock 1 , 2 & 3...
```