LAB # 6

Deadlock in concurrency

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:- **(Deadlock Causes)**

DeadLockTask.java

```
public class DeadLockTask {
    private static final Object lockA = new Object();  2 usages
    private static final Object lockB = new Object();  2 usages
    private static final Object lockC = new Object();  2 usages
    public static void main(String[] args) {
        System.out.println("Name: Rayyan Wamiq | Roll no.: 2023F-BSE-095");
        Thread t1 = new Thread(() -> {
            synchronized (lockA) {
                System.out.println("T1 acquired Lock A");
                sleep( ms: 100);
                synchronized (lockB) {
                    System.out.println("T1 acquired Lock B");
                }
            }
        });
        Thread t2 = new Thread(() -> {
            synchronized (lockB) {
                System.out.println("T2 acquired Lock B");
                sleep( ms: 100);
                synchronized (lockC) {
                    System.out.println("T2 acquired Lock C");
                }
            }
        });
        Thread t3 = new Thread(() -> {
            synchronized (lockC) {
                System.out.println("T3 acquired Lock C");
                sleep( ms: 100);
                synchronized (lockA) {
                    System.out.println("T3 acquired Lock A");
                }
                    }
                }
            });
            t1.start();
            t2.start();
            t3.start();
    }
    private static void sleep(long ms) {  3 usages
        try { Thread.sleep(ms); }
        catch (InterruptedException ignored) {}
    }
}
```

Output:-

```
C:\Users\LAPCOM\.jdks\openjdk-25\bin\java.exe
Name: Rayyan Wamiq | Roll no.: 2023F-BSE-095
T1 acquired Lock A
T2 acquired Lock B
T3 acquired Lock C
```

Code:- (Deadlock Solution)

DeadlockSolved.java

```java
public class DeadlockSolved {
    private static final Object lockA = new Object();  1 usage
    private static final Object lockB = new Object();  1 usage
    private static final Object lockC = new Object();  1 usage
    public static void main(String[] args) {
        System.out.println("Name: Rayyan Wamiq | Roll no.: 2023F-BSE-095");
        Thread t1 = new Thread(() -> acquireLocksInOrder( threadName: "T1"));
        Thread t2 = new Thread(() -> acquireLocksInOrder( threadName: "T2"));
        Thread t3 = new Thread(() -> acquireLocksInOrder( threadName: "T3"));
        t1.start();
        t2.start();
        t3.start();
    }
    private static void acquireLocksInOrder(String threadName) {  3 usages
        synchronized (lockA) {
            System.out.println(threadName + " acquired Lock A");
            sleep( ms: 50);
            synchronized (lockB) {
                System.out.println(threadName + " acquired Lock B");
                sleep( ms: 50);

                synchronized (lockC) {
                    System.out.println(threadName + " acquired Lock C");
                }
            }
        }}
     private static void sleep(long ms) {  2 usages
        try { Thread.sleep(ms); }
        catch (InterruptedException ignored) {}
    }
}
```

Output:-

```
C:\Users\LAPCOM\.jdks\openjdk-25\bin\java.ex
Name: Rayyan Wamiq | Roll no.: 2023F-BSE-095
T1 acquired Lock A
T1 acquired Lock B
T1 acquired Lock C
T2 acquired Lock A
T2 acquired Lock B
T2 acquired Lock C
T3 acquired Lock A
T3 acquired Lock B
T3 acquired Lock C

Process finished with exit code 0
```