Aima Khan
2023F-BSE-04

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:

(A) Deadlock Version (Un-sequenced locks)

```
public class DeadlockExample {

    public static Object Lock1 = new Object();
    public static Object Lock2 = new Object();
    public static Object Lock3 = new Object();

    public static void main(String[] args) {
        ThreadA t1 = new ThreadA();
        ThreadB t2 = new ThreadB();
        ThreadC t3 = new ThreadC();

        t1.start();
        t2.start();
        t3.start();
    }
```

```
    static class ThreadA extends Thread {
        public void run() {
            synchronized (Lock1) {
                System.out.println("Thread A: Holding Lock1...");
                try { Thread.sleep(20); } catch(Exception e){}

                System.out.println("Thread A: Waiting for Lock2...");
                synchronized (Lock2) {
                    System.out.println("Thread A: Holding Lock1 & Lock2");
                }
            }
        }
    }
```

```
    static class ThreadB extends Thread {
        public void run() {
            synchronized (Lock2) {
                System.out.println("Thread B: Holding Lock2...");
                try { Thread.sleep(20); } catch(Exception e){}

                System.out.println("Thread B: Waiting for Lock3...");
                synchronized (Lock3) {
                    System.out.println("Thread B: Holding Lock2 & Lock3");
                }
            }
        }
    }
```

```
    static class ThreadC extends Thread {
        public void run() {
            synchronized (Lock3) {
                System.out.println("Thread C: Holding Lock3...");
                try { Thread.sleep(20); } catch(Exception e){}

                System.out.println("Thread C: Waiting for Lock1...");
                synchronized (Lock1) {
                    System.out.println("Thread C: Holding Lock3 & Lock1");
                }
            }
        }
    }
}
```

OUTPUT

```
Thread A: Holding Lock1...
Thread B: Holding Lock2...
Thread C: Holding Lock3...
Thread A: Waiting for Lock2...
Thread B: Waiting for Lock3...
Thread C: Waiting for Lock1...
```

B) Deadlock Solution (Sequenced Locks: Lock1 → Lock2 → Lock3)

```
public class DeadlockSolved {

    public static Object Lock1 = new Object();
    public static Object Lock2 = new Object();
    public static Object Lock3 = new Object();

    public static void main(String[] args) {
        ThreadA t1 = new ThreadA();
        ThreadB t2 = new ThreadB();
        ThreadC t3 = new ThreadC();

        t1.start();
        t2.start();
        t3.start();
    }
```

```
  static class ThreadA extends Thread {
      public void run() {
          synchronized (Lock1) {
              System.out.println("Thread A: Holding Lock1...");
              try { Thread.sleep(20); } catch(Exception e){}

              synchronized (Lock2) {
                  System.out.println("Thread A: Holding Lock1 & Lock2");

                  synchronized (Lock3) {
                      System.out.println("Thread A: Holding all locks");
                  }
              }
          }
      }
  }
```

```
   static class ThreadB extends Thread {
       public void run() {
           synchronized (Lock1) {
               System.out.println("Thread B: Holding Lock1...");
               try { Thread.sleep(20); } catch(Exception e){}

               synchronized (Lock2) {
                   System.out.println("Thread B: Holding Lock1 & Lock2");

                   synchronized (Lock3) {
                       System.out.println("Thread B: Holding all locks");
                   }
               }
           }
       }
   }
```

```
    static class ThreadC extends Thread {
        public void run() {
            synchronized (Lock1) {
                System.out.println("Thread C: Holding Lock1...");
                try { Thread.sleep(20); } catch(Exception e){}

                synchronized (Lock2) {
                    System.out.println("Thread C: Holding Lock1 & Lock2");

                    synchronized (Lock3) {
                        System.out.println("Thread C: Holding all locks");
                    }
                }
            }
        }
    }
}
```

OUTPUT

```
Thread A: Holding Lock1...
Thread A: Holding Lock1 & Lock2
Thread A: Holding all locks
Thread B: Holding Lock1...
Thread B: Holding Lock1 & Lock2
Thread B: Holding all locks
Thread C: Holding Lock1...
Thread C: Holding Lock1 & Lock2
Thread C: Holding all locks
```