**Deadlock in java**

Deadlock in java is a part of multithreading. Deadlock can occur in a situation when a thread is waiting for an object lock, that is acquired by another thread and second thread is waiting for an object lock that is acquired by first thread. Since, both threads are waiting for each other to release the lock, the condition is called deadlock.



**Example of Deadlock in java**

public class TestDeadlockExample1 {

  public static void main(String[] args) {

    final String resource1 = "ratan jaiswal";

    final String resource2 = "vimal jaiswal";

    // t1 tries to lock resource1 then resource2

    Thread t1 = new Thread() {

      public void run() {

          synchronized (resource1) {

           System.out.println("Thread 1: locked resource 1");

           try { Thread.sleep(100);} catch (Exception e) {}

           synchronized (resource2) {

            System.out.println("Thread 1: locked resource 2");

           }

         }

      }

    };

    // t2 tries to lock resource2 then resource1

    Thread t2 = new Thread() {

      public void run() {

        synchronized (resource2) {

          System.out.println("Thread 2: locked resource 2");

          try { Thread.sleep(100);} catch (Exception e) {}

          synchronized (resource1) {

            System.out.println("Thread 2: locked resource 1");

          }

        }

      }

    };

    t1.start();

    t2.start();

  }

}

Output: Thread 1: locked resource 1

Thread 2: locked resource 2

Inorder to resolve the deadlock problem , we have to change the order of locks in thread t2

    // t2 tries to lock resource1 then resource2

    Thread t2 = new Thread() {

      public void run() {

        synchronized (resource1) {

          System.out.println("Thread 2: locked resource 2");

          try { Thread.sleep(100);} catch (Exception e) {}

          synchronized (resource2) {

            System.out.println("Thread 2: locked resource 1");

          }

        }

      }

}