import java.util.Random;

import java.util.concurrent.locks.ReentrantReadWriteLock;

class ThreadLock extends Thread {

private static final ReentrantReadWriteLock rwlock = new ReentrantReadWriteLock();

@Override

public void run()

{

try {

readData();

sleep(2000);

writeData();

} catch (final Exception e)

{

System.out.println(e);

}

}

public void readData()

{

try {

rwlock.readLock().lock();

System.out.println(Thread.currentThread().getName() + " is reading and the value is 5");

} finally {

System.out.println(Thread.currentThread().getName() + " is exiting after reading.");

rwlock.readLock().unlock();

}

}

public void writeData()

{

try {

rwlock.writeLock().lock();

Random rand = new Random();

int n = rand.nextInt(50);

System.out.println(Thread.currentThread().getName() + " has the write lock and is writing.");

System.out.println("The new value is: " + (5 + n));

}

finally

{

System.out.println(Thread.currentThread().getName() + " is releasing the lock and exiting after writing.");

rwlock.writeLock().unlock();

}

}

}

public class ReentrantLockTest {

public static void main(final String[] args) throws Exception {

ThreadLock obj = new ThreadLock();

ThreadLock obj2 = new ThreadLock();

obj.start();

obj2.start();

}

}