```
import java.util.concurrent.Semaphore;

class _4_mutex {

    static Semaphore readLock = new Semaphore(1);

    static Semaphore writeLock = new Semaphore(1);

    static int readCount = 0;

    static class Read implements Runnable {

        @Override

        public void run() {

            try {

//Acquire Section

                readLock.acquire();

                readCount++;

                if (readCount == 1) {

                    writeLock.acquire();

                }

                readLock.release();

//Reading section

                System.out.println("Thread "+Thread.currentThread().getName() + " is READING");

                Thread.sleep(1500);

                System.out.println("Thread "+Thread.currentThread().getName() + " has FINISHED
READING");

//Releasing section

                readLock.acquire();

                readCount--;

                if(readCount == 0) {

                    writeLock.release();

                }

                readLock.release();

            } catch (InterruptedException e) {

                System.out.println(e.getMessage());

            }
        }
    }
    static class Write implements Runnable {
        @Override
        public void run() {
            try {
                writeLock.acquire();
                System.out.println("Thread "+Thread.currentThread().getName() + " is WRITING");
                Thread.sleep(2500);
                System.out.println("Thread "+Thread.currentThread().getName() + " has finished
WRITING");
                writeLock.release();
            } catch (InterruptedException e) {
                System.out.println(e.getMessage());
            }
        }
    }
    public static void main(String[] args) throws Exception {
        Read read = new Read();
        Write write = new Write();
        Thread t1 = new Thread(read);
        t1.setName("thread1");
        Thread t2 = new Thread(read);
        t2.setName("thread2");
        Thread t3 = new Thread(write);
        t3.setName("thread3");
        Thread t4 = new Thread(read);
        t4.setName("thread4");
        t1.start();
        t3.start();
        t2.start();
        t4.start();
    }
}
```