```java
-------------ReEntrantTask.java----------
package com.ameya.tasks;

import java.util.concurrent.locks.ReentrantLock;

public class ReEntrantTask implements Runnable {

    private String threadName;
    ReentrantLock r1;
    public ReEntrantTask(String threadName,ReentrantLock r1) {
        this.threadName=threadName;
        this.r1=r1;
    }
    @Override
    public void run() {
        System.out.println("Within run trying to acquire lock thread :: "+threadName);
        boolean locked=r1.tryLock();
        if(locked) {
            try {
                System.out.println("Thread :: "+threadName+" has aquired lock");
                lockMethod();
            }finally {
                r1.unlock();
                System.out.println("Count of locks held by thread :: "+threadName+" - "+r1.getHoldCount());
            }
        }else {
            System.out.println("Thread :: "+threadName+" not able to acquire lock");
        }

    }

    public void lockMethod() {
        System.out.println("Within lockMethod, thread :: "+threadName+" -- Waiting to acquire lock");
        r1.lock();
        try {
            System.out.println("Count of locks held by thread ::
```

```
            "+threadName+" - "+r1.getHoldCount());
        }finally {
            r1.unlock();
        }
    }
}
-------------------TestReentrantLock.java----------
package com.ameya.test;

import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.locks.ReentrantLock;

import com.ameya.tasks.ReEntrantTask;

public class TestReentrantLock {

    public static void main(String[] args) {
        ReentrantLock r1=new ReentrantLock();
        ExecutorService executor=Executors.newFixedThreadPool(2);
        for(int i=0;i<4;i++) {
            executor.execute(new ReEntrantTask("Thread -> "+i, r1));
        }
        executor.shutdown();
    }

}

```