ReadWriteLock

共享锁 与 独占锁

java.util.concurrent.locks.ReentrantReadWriteLock

- 读锁 —— 共享锁
- 写锁 —— 独占锁

读	读	不互斥
读	写	互斥
写	写	互斥

ReentrantReadWriteLock.ReadLock 和 ReentrantReadWriteLock.WriteLock

- 实现了Lock接口
- 都是可重入锁，但不是ReentrantLock的子类

锁升级和锁降级

- 已经持有WriteLock，再加ReadLock，可以，会成功
- 已经持有ReadLock，再加WriteLock，不会成功，一直等待

```
ReentrantReadWriteLock rwLock = new ReentrantReadWriteLock();
```

```
rwLock.readLock().lock();
rwLock.writeLock().lock(); //死锁，这句会一直等待
```

```
rwLock.writeLock().lock();
rwLock.readLock().lock();      //在持有写锁的情况下，同一线程是可以获取读锁的。
rwLock.writeLock().unlock(); //可以先释放写锁；（先释放那个没有要求）
rwLock.readLock().unlock();
```

```java
public class CachedData {
    private Data data;
    private long validUntil;
    private final ReentrantReadWriteLock rwLock = new ReentrantReadWriteLock();

    public void processCachedData() {
        rwLock.readLock().lock();
        if (System.currentTimeMillis() > validUntil) {   // 缓存失效，需要更新缓存
            rwLock.readLock().unlock();
            rwLock.writeLock().lock();
            try {                // 重新检查是否缓存已经到期，因为可能在等待写锁期间，其他线程已经更新了缓存
                if (System.currentTimeMillis() > validUntil) {    // 更新缓存
                    data = new Data();
                    validUntil = System.currentTimeMillis();
                }
               rwLock.readLock().lock();        // 先写锁；后读锁是允许的，准备锁降级
            } finally {
                rwLock.writeLock().unlock();   // 已经释放写锁，依旧持有读锁
            }
        }

        try {
            use(data);                         // 使用缓存的data进行业务处理
        } finally {
           rwLock.readLock().unlock();
        }
    }
}
```