**Difference between ReentrantLock vs StampedLock**

1. StampedLock is an alternative to Reentrant Read Write Lock.
2. Another key is that unlike RWLocks, StampedLocks **are not reentrant**.
3. **StampedLocks are faster when there is more read and less write.** This is done by optimistic locking for read operations

**How StampedLock works**

1. They employ a concept of stamps that are long values that serve as tickets used by any lock / unlock operation.
2. This means that to unlock a R/W operation you need to pass it its correlating lock stamp.
3. Pass the wrong stamp, and you’re risking an exception, or worse – unexpected behaviour.
4. They have the downside that threads can now deadlock against themselves.

StampedLock lock = new StampedLock();

long stamp = lock.tryOptimisticRead();

try {

System.out.println("Optimistic Lock Valid: " + lock.validate(stamp));

Thread.sleep(100);

System.out.println("Optimistic Lock Valid: " + lock.validate(stamp));

Thread.sleep(1000);

System.out.println("Optimistic Lock Valid: " + lock.validate(stamp));

} catch (Exception e) {

} finally {

lock.unlock(stamp);

}

An optimistic read lock is acquired by calling tryOptimisticRead() which always returns a stamp **without blocking the current thread**, no matter if the lock is actually available. If there's already a write lock active the returned stamp equals zero. You can always check if a stamp is valid by calling lock.validate(stamp).