作业内容：

CAS底层原理？谈谈对UnSafe的理解？为什么不用synchronized也能实现++操作？

- 底层原理

  > CAS 的底层原理就是每次更新前都比较当前值是否被其他线程修改过，如果没有则修改如果修改过则失败。

- UnSafe

  > UnSafe 它是CAS 的核心类，它里面的方法多数都被标记为native，它是由C++ 实现的。它可以直接操作内存，每次去内存中取出数据与期望的数据作对比，然后根据对比的结果作更新操作。

- ++操作

  > 因为每次更新都会比较原来的值，如果不等则重新操作，这样保证了结果的正确性，也实现了原子操作，同时还不需要额外的锁的消耗。

ABA问题？原子更新引用知道吗？

- ABA 问题

  > CAS 最严重的一个缺点就是ABA 问题。所谓ABA 问题其实就是在线程1中拿到了数据A，准备做CAS 操作的时候，CPU 资源被收回。
  >
  > 这个时候，线程2对这个数据进行了两次操作，第一次先将其变更为了B，然后再变更回了A。之后当线程1得到CPU 资源 回来通过CAS 操作数据时，发现数据A 是可以正常变更的，然后按正常的操作变更。
  >
  > 这个效果就是所谓的ABA 问题。

- 原子更新引用

  > 原子更新引用，我们要使用带时间戳的原子类对象替代普通原子引用类对象，以提高线程安全。从而避免ABA 问题的产生。

多线程下集合不安全问题该如何解决

> 多线程下集合不安全的问题就是使用JUC包下面提供的线程安全的集合类对象。