public class AtomicStampedReferencTest {  
 private static Double *version*=1.0D;  
 private static Integer *val*=1;  
 public static void main(String a[]){  
 AtomicStampedReference<Double> asr=new AtomicStampedReference<Double>(*version*,*val*);  
 for(int i=0;i<2;i++){  
 Thread t=new Thread(new Runnable() {  
 @Override  
 public void run() {  
 Double newVersion=Math.random();  
 Integer newVal=2;  
 //版本号变化导致无法成功  
 if(!asr.weakCompareAndSet(newVersion,newVersion,asr.getStamp(),newVal)){  
 version=newVersion;  
 System.out.println(Thread.currentThread().getName()+"--false-"+asr.getReference()+"---"+asr.getStamp());  
 }else{  
 System.out.println(Thread.currentThread().getName()+"--true-"+asr.getReference()+"---"+asr.getStamp());  
  
 }  
 }  
 });  
 t.start();  
  
 }  
 }  
}

通过版本号控制，

public boolean compareAndSet(V expectedReference,  
 V newReference,  
 int expectedStamp,  
 int newStamp) {  
 Pair<V> current = pair;  
 return  
 expectedReference == current.reference &&  
 expectedStamp == current.stamp &&  
 ((newReference == current.reference &&  
 newStamp == current.stamp) ||  
 casPair(current, Pair.*of*(newReference, newStamp)));  
}

如果版本号相同，值相同直接返回