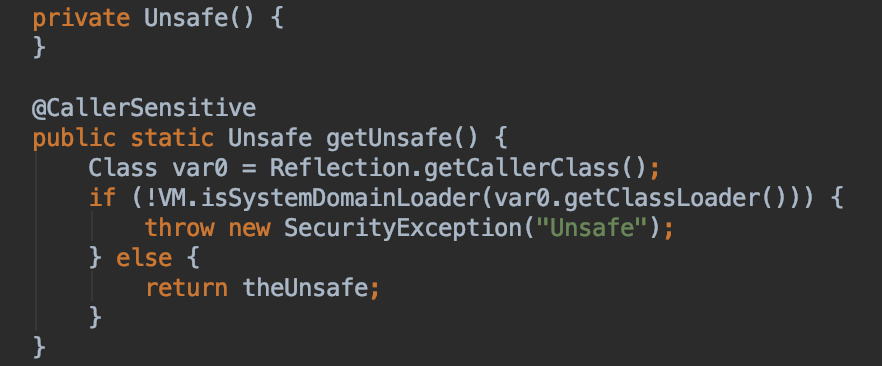
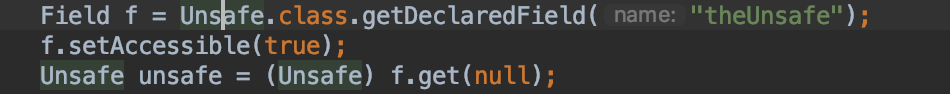
# UnSafe源码类解读

Sun.misc.UnSafe 可以在任意的内存处读写数据，CAS原子操作，线程操作；这个类由于不太安全，所以JDK也不推荐在开发者代码中使用，所以，在用户代码中是不能直接构造得到这个类的对象的，从源码可以看到不能通过构造方法的方式，提供了一个单例的对象。



而且只能从从引导类加载器加载的类中进行加载，其他途径调用这个方法会抛出SecurityException的异常。

但是可以通过反射的方法得到这个类型的单例静态属性theUnsafe。这样可以直接得到Unsafe类的单例对象，而绕过了安全检查。



public native Object allocateInstance(Class cls)

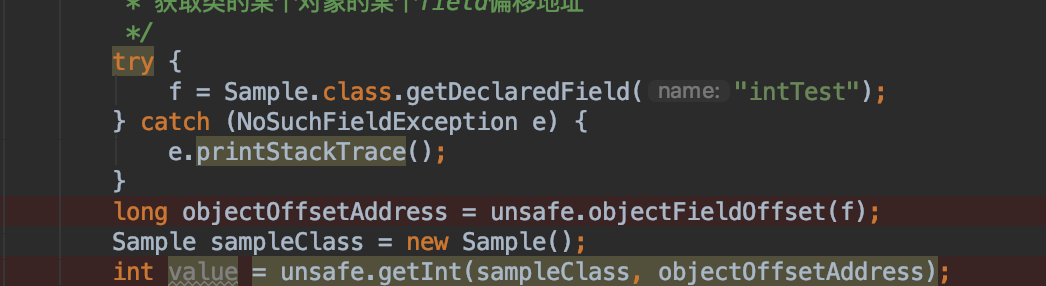
throws InstantiationException;

不调用构造方法就可以生成一个Class类的对象，不会经过域初始化与构造初始化，基本类型都初始化为0，引用类型都是null。

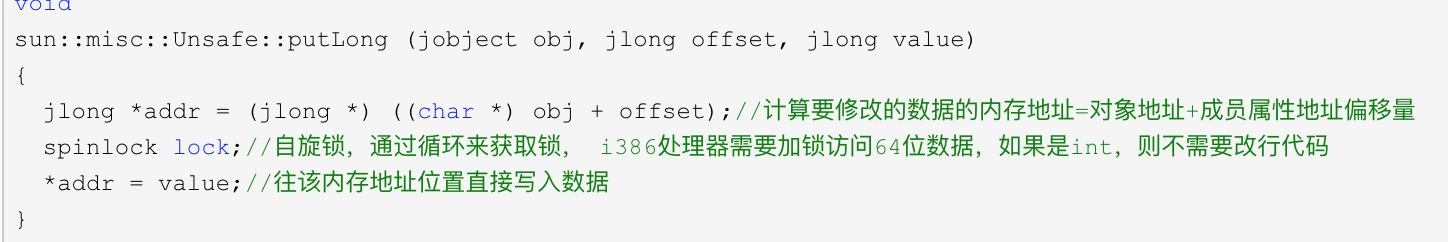
public native long objectFieldOffset(Field var1);

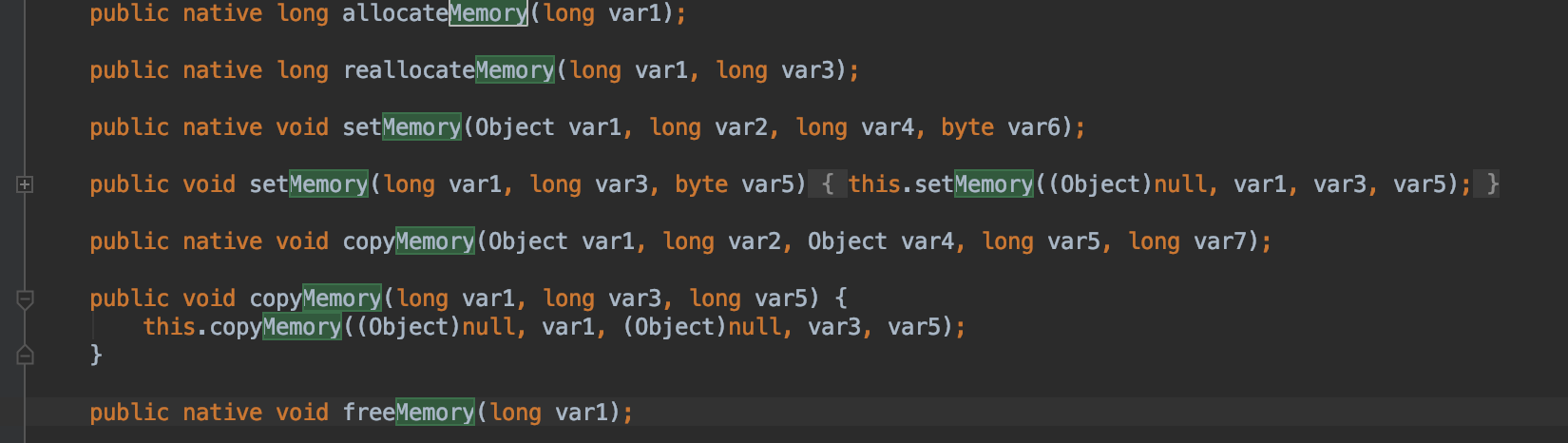
返回属性在对象存储中的相对的偏移量；

一堆putXXXX(Object object,long offset，XXXX value),getXXXX(Object object,long offset)的方法可以直接读取写入对象偏移量处的值，offset就是属性的相对的偏移量；

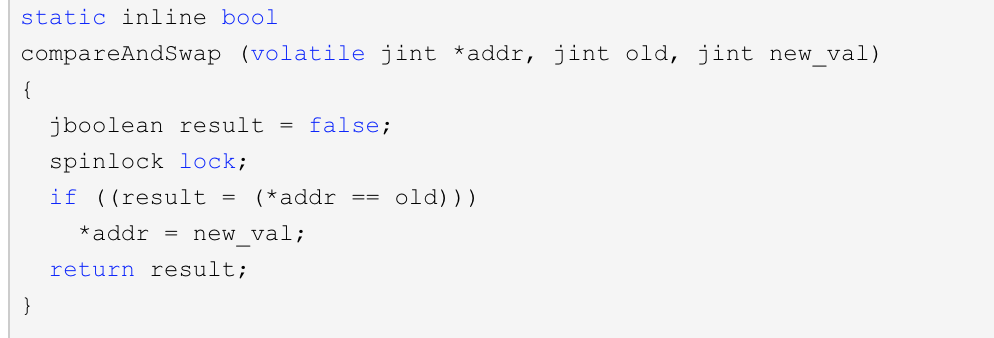


比如获取int类型的值。





CAS操作的源码：



Unsafe API含有的接口：

Info：addressSize，pageSize，返回低级的内存信息；

Objects：allocateInstance，objectFieldOffset，用于操作对象与对象字段的方法；

Classes:staticFieldOffset,defineClass,defineAnonymousClass,ensureClassInitialized,用于操作类及其静态字段的方法。

Arrays:arrayBaseOffset,arrayIndexScala 操作数组的方法；

Synchronization: monitorEnter,tryMonitorEnter,monitorExit,compareAndSwapInt,putOrderedInt,低级的同步原语；

* Memmory:直接内存访问方法，allocateMemory
* copyMemory
* freeMemory
* getAddress
* getInt
* putInt
* 阻塞和唤醒

pack()  
unpack()