链滴

JDK11-juc 包系列之 atomic 的 AtomicBoolean 类 (一)

作者：MaidongAndYida

原文链接：https://ld246.com/article/1553324556401

来源网站：链滴

<img src="https://ld246.com/images/img-loading.svg" alt="11.jpg" data-src="https://b3logfile.com/file/2019/03/11-cd6be773.jpg?imageView2/2/interlace/1/format/jpg">
~~第一次看源码，如果有理解不对的地方希望大家可以留下评论谢谢啦。~~
理解这个类之前需要先理解 java 内存模型
<hr>
<h3 id="51-63-line">51-63 line</h3>
<pre><code class="highlight-chroma">public class AtomicBoolean implements java.io.Serializable {

    private static final long serialVersionUID = 4654671469794556979L;
    private static final VarHandle VALUE;
    static {
        try {
            MethodHandles.Lookup l = MethodHandles.lookup();
            VALUE = l.findVarHandle(AtomicBoolean.class, "value", int.class);
        } catch (ReflectiveOperationException e) {
            throw new ExceptionInInitializerError(e);
        }
    }

    private volatile int value;

</code></pre>
上段代码涉及知识点：
<pre><code class="language-mysql highlight-chroma">1、Java9以后新增加并延到JDK11LTS版本的量句柄`</span><span class="highlight-n">VarHandle</span><span class="highlight-o">`
2、MethodHandles.Lookup
3、static{}代码块和部类
4、volatile关键字
</code></pre>
<h3 id="1-Java9以后新增加并延续到JDK11LTS版本的变量句柄">1、Java9 以后新增加并延续到 JDK11LTS 版本的变量句柄 <code>VarHandle</code></h3>

VarHandle 是对变量或参数定义的变量系列的动态强类型引用，包括静态字段，非静态字段，组元素或堆外数据结构的组件。在各种_访问模式_下都支持访问这些变量，包括普通读/写访问，易失读/写访问以及比较和交换。<code>这样理解：VarHandle利用访问模式可以用来改变AtomicBoolean的值，访问模式控制原子性和一致性属性</code>
访问模式分为以下几类：

读取访问模式，获取指定内存排序效果下的变量值。该组对应属于该组的访问模式的方法的组成方法中<code>get</code>，<code>getVolatile</code>，<code>getAcquire</code>，<code>getOpaque</ode>。
写入访问模式，在指定的内存排序效果下设置变量的值。该组对应属于该组的访问模式的方法的成的方法中<code>set</code>，<code>setVolatile</code>，<a href="https://ld246.com/forward?got=https%3A%2F%2Fdocs.oracle.com%2Fjavase%2F9%2Fdocs%2Fapi%2Fjava%2Flang%2Finvoe%2FVarHandle.html%23setRelease-java.lang.Object...-" target="_blank" rel="nofollow ugc"<code>setRelease</code>，<code>setOpaqe</code>。
原子更新访问模式，例如，在指定的内存排序效果下，原子地比较和设置变量的值。该组对应属该组的访问模式的方法的组成的方法中<code>cmpareAndSet</code>，<codeweakCompareAndSetPlain</code>，<a href="https://ld246.com/forward?goto=https%A%2F%2Fdocs.oracle.com%2Fjavase%2F9%2Fdocs%2Fapi%2Fjava%2Flang%2Finvoke%2FVarandle.html%23weakCompareAndSet-java.lang.Object...-" target="_blank" rel="nofollow ugc"<code>weakCompareAndSet</code>，<code>weakCompareAndSetAcquire</code>，<code>weakCompareAndSetRelease</code>，<code>compareAndExchangeAcquire</cod>，<a href="https://ld246.com/forward?goto=https%3A%2F%2Fdocs.oracle.com%2Fjavse%2F9%2Fdocs%2Fapi%2Fjava%2Flang%2Finvoke%2FVarHandle.html%23compareAndExch

nge-java.lang.Object...-" target="_blank" rel="nofollow ugc"><code>compareAndExchange</code>，<code>compareAnExchangeRelease</code>，<code>getAndSe</code>，<code>getAndSetAcquire</code>，<code>getAndSetRelease</coe>。
数字原子更新访问模式，例如，通过在指定的内存排序效果下添加变量的值，以原子方式获取和置。该组的属于该组的相应的访问模式的方法包括的方法中<code>getAndAdd</code>，<codegetAndAddAcquire</code>，<code>etAndAddRelease</code>，
按位原子更新访问模式，例如，在指定的内存排序效果下，以原子方式获取和按位 OR 变量的值该组对应属于该组的访问模式的方法的组成的方法中<a href="https://ld246.com/forward?goto=https%3A%2F%2Fdocs.oracle.com%2Fjavase%2F9%2Fdocs%2Fapi%2Fjava%2Flang%2Finvoke%2VarHandle.html%23getAndBitwiseOr-java.lang.Object...-" target="_blank" rel="nofollow ugc"<code>getAndBitwiseOr</code>，<code>getAndBitwiseOrAcquire</code>，<code>getAndBitwiseOrRelease</code>，<code>getAndBitwiseAnd</code>，<code>getAndBitwiseAndAcquire</code>，<code>getAndBitwiseAndRelease</code>，<code>getAndBitwiseXor</code>，<code>getAndBitwiseXorAcquire</code>，<aref="https://ld246.com/forward?goto=https%3A%2F%2Fdocs.oracle.com%2Fjavase%2F9%2Focs%2Fapi%2Fjava%2Flang%2Finvoke%2FVarHandle.html%23getAndBitwiseXorRelease-java.lng.Object...-" target="_blank" rel="nofollow ugc"><code>getAndBitwiseXorRelease</code>

/a>。

<h3 id="MethodHandles-Lookup">MethodHandles.Lookup</h3>
从上面代码可以看出，MethodHandles.Lookup 调用其内部的方法产出了 <code>VarHandle<code> 对象。
MethodHandles.Lookup:一个查找对象是用于创建方法处理，当创建需要访问检查的工厂。方句柄在调用它们时不执行访问检查，而是在创建它们时执行。因此，在创建方法句柄时必须强制执行法句柄访问限制。强制执行这些限制的调用者类称为查找类</>。
需要创建方法句柄的查找类将调用<code>MethodHandles.looku</code>为自己创建工厂。当 <code>Lookup</code> 被创建工厂对象，查找类的身份被确，并安全地存储在 <code>Lookup</code> 对象。然后，查找类（或其委托）可以在 <code>Look</code> 对象上使用工厂方法来为访问检查的成员创建方法句柄。这包括允许查找类的所有方法，造函数和字段，甚至是私有的。<code>简单理解：通过MethodHandles.Lookup反射出某方法/某象/某成员</code>
查找可能会失败，因为查找类无法访问包含类，或者因为缺少所需的类成员，或者因为查找类无访问所需的类成员，或者因为查找对象不够信任访问该成员。在任何这些情况下，<code>Reflective perationException</code> 都将从尝试查找中抛出。确切的类将是以下之一：

NoSuchMethodException - 如果请求方法但不存在
NoSuchFieldException - 如果请求了一个字段但不存在
IllegalAccessException - 如果成员存在但访问检查失败

<h3 id="static--代码块和内部类">static{}代码块和内部类</h3>
当类被载入时，静态代码块被执行，且只被执行一次，静态块常用来执行类属性的初始化。java 中态代码块的用法 static 用法详解
内部类分为静态内部类，成员内部类，局部内部类，匿名内部类四种。Java 内部类详解
<h3 id="volatile关键字">volatile 关键字</h3>
整个类的核心就是 <code>volatile</code> 关键字,它保证了变量在线程只见是可见的。volatile 和 synchronized 的区