Java AtomicInteger

AtomicInteger

AtomicInteger，一个提供原子操作的 Integer 的类。

在 Java 语言中，++i 和 i++操作并不是线程安全的，在使用的时候，不可避免的会用到 synchronized 关键字。而 AtomicInteger 则通过一种线程安全的加减操作接口。

主要接口包括:

```
public final int get() //获取当前的值
public final int getAndSet(int newValue)//获取当前的值，并设置新的值
public final int getAndIncrement()//获取当前的值，并自增
public final int getAndDecrement() //获取当前的值，并自减
public final int getAndAdd(int delta) //获取当前的值，并加上预期的值
```

使用 AtomicInteger:

```
class Test2 {
        private AtomicInteger count = new AtomicInteger();

        public void increment() {
                        count.incrementAndGet();
        }
        //使用 AtomicInteger 之后，不需要加锁，也可以实现线程安全。
        public int getCount() {
                return count.get();
        }
}
```

AtomicInteger 由硬件提供原子操作指令实现的。在非激烈竞争的情况下，开销更小，速度更快。

AtomicInteger 的关键域只有一下 3 个：

```
// setup to use Unsafe.compareAndSwapInt for updatesprivate static final Unsafe
unsafe = Unsafe.getUnsafe();
private static final long valueOffset;
static {
         try {
                valueOffset =
unsafe.objectFieldOffset(AtomicInteger.class.getDeclaredField("value"));
        } catch (Exception ex) {
               throw new Error(ex);
        }
    }
private volatile int value;
```

unsafe 是 java 提供的获得对对象内存地址访问的类。它的作用就是在更新操作时提供“比较并替换”的作用。实际上就是 AtomicInteger 中的一个工具。

valueOffset 是用来记录 value 本身在内存的偏移地址的，这个记录，也主要是为了在更新操作在内存中找到 value 的位置，方便比较。**注意这个偏移量是 value 在 AtomicInteger 对象内部的偏移量，而不是在整个内存中的偏移量**。

注意：value 是用来存储整数的时间变量，这里被声明为 volatile，就是为了保证在更新操作时，当前线程可以拿到 value 最新的值（并发环境下，value 可能已经被其他线程更新了）。

这里，我们以自增的代码为例，可以看到这个并发控制的核心算法：

```
/**
*Atomicallyincrementsbyonethecurrentvalue.
*
*@returntheupdatedvalue
*/
publicfinalintincrementAndGet(){for(;;){//这里可以拿到 value 的最新值
intcurrent=get();
intnext=current+1;if(compareAndSet(current,next))
returnnext;
}
}

publicfinalbooleancompareAndSet(intexpect,intupdate){//使用 unsafe 的 native 方法，实现高效的硬件级别 CAS
        returnunsafe.compareAndSwapInt(this,valueOffset,expect,update);
}
```