**1.原子更新基本类型**

·AtomicBoolean：原子更新布尔类型。

·AtomicInteger：原子更新整型。

·AtomicLong：原子更新长整型

·int addAndGet（int delta）：以原子方式将**输入的数值**与**实例中的值**（AtomicInteger里的value）相加，并**返回结果**。

·boolean compareAndSet（int expect，int update）：如果**输入的数值**等于**预期值**，则以**原子方式**将该值设置为**输入**的值。

·int getAndIncrement()：以**原子方式**将当前值加1，注意，这里返回的是**自增前的值**。

·void lazySet（int newValue）：**最终会设置成newValue**，使用lazySet设置值后，可能导致其他线程在之后的**一小段时间**内还是可以**读到旧的值**。

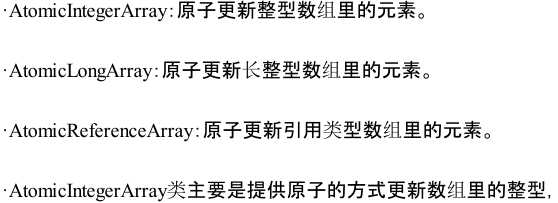
·int getAndSet（int newValue）：以**原子方式**设置为**newValue的值**，并**返回旧值**。

**getAndIncrement是如何实现原子操作的呢？**

|  |
| --- |
| **public** **final** **int** getAndIncrement() {  **for** (;;) {  **int** current = get();  **int** next = current + 1;  **if** (compareAndSet(current, next))  **return** current;  }  }  **public** **final** **boolean** compareAndSet(**int** expect, **int** update) {  **return** unsafe.compareAndSwapInt(**this**, valueOffset, expect, update);  } |

无限循环第一步先取得atomicInteger存储的数值，，第二步对atomicInteger的当前数值进行加1操作，，关键的第三步调用compareAndSet方法来进行原子更新操作，该方法先通过this和valueoffset计算出当前的值的大小，然后except是传入的current的值，比较计算出的值和expect的值是否相等，，相等意味着atomicInteger的值没有被其他线程修改过，则将atomicInteger的当前数值更新成next的值，如果不等compareAndSet方法会返回false，程序进行for循环重新进行compareAndSet操作。

**2.原子更新数组**



数组value通过构造方法传递进去，然后AtomicIntegerArray会将当前数组复制一份，所以当AtomicIntegerArray对内部的数组元素进行修改时，不会影响传入的数组