# 原子性

原子是世界上的最小单位，具有不可分割性。比如 a=0；（a非long和double类型） 这个操作是不可分割的，那么我们说这个操作时原子操作。再比如：a++； 这个操作实际是a = a + 1；是可分割的，所以他不是一个原子操作。非原子操作都会存在线程安全问题，需要我们使用同步技术（sychronized）来让它变成一个原子操作。一个操作是原子操作，那么我们称它具有原子性。

**[Java](http://lib.csdn.net/base/javase" \o "Java SE知识库" \t "http://blog.csdn.net/u010359884/article/details/_blank)**的concurrent包下提供了一些原子类，我们可以通过阅读API来了解这些原子类的用法。比如：AtomicInteger、AtomicLong、AtomicReference等。

# 可见性

可见性，是指线程之间的可见性，一个线程修改的状态对另一个线程是可见的。也就是一个线程修改的结果。另一个线程马上就能看到。比如：用volatile修饰的变量，就会具有可见性。volatile修饰的变量不允许线程内部缓存和重排序，即直接修改内存。所以对其他线程是可见的。但是这里需要注意一个问题，**volatile只能让被他修饰内容具有可见性，但不能保证它具有原子性。**比如 volatile int a = 0；之后有一个操作 a++；这个变量a具有可见性，但是a++ 依然是一个非原子操作，也就这这个操作同样存在线程安全问题。

# 他们之间关系

原子性是说一个操作是否可分割。可见性是说操作结果其他线程是否可见。这么看来他们其实没有什么关系。

# 实例

1. **package** com.chu.test.thread;
2. /\*\*
3. \* 可见性分析
4. \* @author Administrator
5. \*
6. \*volatile 会拒绝编译器对其修饰的变量进行优化。也就不会存在重排序的问题。volatile只会影响可见性，不会影响原子性。
7. \*下面程序如果不加
8. \*/
9. **public** **class** Test {
11. **volatile** **int** a = 1;
12. **volatile** **boolean** ready;
14. **public** **class** PrintA **extends** Thread{
15. @Override
16. **public** **void** run() {
17. **while**(!ready){
18. Thread.yield();
19. }
20. System.out.println(a);
21. }
22. }
23. **public** **static** **void** main(String[] args) **throws** InterruptedException {
24. Test t = **new** Test();
25. t.**new** PrintA().start();
26. //下面两行如果不加volatile的话，执行的先后顺序是不可预测的。并且下面两行都是原子操作，但是这两行作为一个整体的话就不是一个原子操作。
27. t.a = 48; //这是一个原子操作，但是其结果不一定具有可见性。加上volatile后就具备了可见性。
28. t.ready = **true**;//同理
29. }
31. }    
    上面程序如果变量a不用volatile修饰那么输出结果很可能就是0.。