摘自infoq，原作者 方腾飞。

**volatile**关键字：并不能保证操作的原子性，只能保证数据的可见性。

1.6以后的锁有四种状态：无锁，偏向锁，轻量锁，重量锁。锁可以升级，但不能降级，只是为了提高获得和释放锁的效率。

偏向锁：同一个线程多次请求同一个锁，为了提高效率，只是记录一个偏向线程ID，每次请求的时候进行CAS(比较并且设置)操作。

轻量锁：线程在执行同步块前，JVM会现在当前线程的帧栈中创建一块控件用于保存锁记录，并将对象头中的Mark Word复制过来，然后使用CAS尝试将Mark Word替换为锁指针，如果替换成功则表示无竞争，否则，就会使用自旋来尝试获取锁。

重量锁：阻塞。自旋消耗CPU，因此锁不降级。

**线程池**：

new ThreadPoolExecutor(corePoolSize,maximumPoolSize,keepAliveTime,timeUnit,

runnableTaskQueue, handler);

corePoolSize：线程池基本大小。当向线程池提交任务时，如果线程池未满，即时此时有空闲线程，也会创建新的线程。如果使用了线程池的prestartAllCoreThreads方法，则会提前创建并启动所有基本线程。

maximumPoolSize：线程池最大线程数。如果使用了无界的任务队列，则该参数无效。

keepAliveTime：线程池的工作线程空闲后，保持的存活时间。

timeUnit：keepAliveTime的单位。

runnableTaskQueue：任务队列。可选ArrayBlockingQueue、LinkedBlockingQueue、SynchronousQueue等。

ThreadFactory：线程工厂，可以用它为创建出来的线程设置更有意义的名称。

RejectedExecutionHandler：饱和策略。当队列和线程池都满了，说明线程池处于饱和状态。可以丢弃、抛出异常等，也可自定义。

一般使用：

ExecutorService executor = Executors.newXXXThreadPool();

shutdown和shutdownNow的区别：便利线程池中的工作线程，逐个调用线程的interrupt方法来中断线程，所以无法响应中断的任务可能永远无法关闭。shutdownNow将线程池的状态设置为STOP，然后尝试停止所有暂停运行和正在执行的线程，并返回结果列表；shutdown只是将线程池状态改为SHUTDOWN，并尝试终止所有没有在运行任务的线程。

一般情况下调用shutdown，如果不在乎任务有没有执行完，则可以调用shutdownNow。

**ConcurrentHashMap**

哈希算法：将任意输入转为相同长度的输出的加密方式，输出值成为哈希值。

哈希表：

HashMap不是线程安全的，HashTable效率低下(因为竞争激烈，阻塞严重)。

锁分段：HashTable之所以效率低下，是因为所有操作都竞争同一把锁。假如容器里有多把锁，每把锁只用来锁定容器中的部分数据，那么当多个任务请求不同部分数据的时候就不会存在锁竞争。ConcurrentHashMap使用的就是这种技术。

**原子操作（atomic operation）**

不可被中断的一个或一系列操作。

**线程安全队列的实现**：

实现方法：阻塞(出队、入队使用相同或不同的锁)，非阻塞(循环CAS)

ConcurrentLinkedQueue：基于链接节点的无界线程安全队列。