线程和并发

从并发性和安全性的基本概念出发,

使用类库提供放入基本并发构建块,用于避免并发危险,构造线程安全的类及验证线程安全的规则,将小的线程安全类组合成更大的线程安全类;

使用线程提高并发应用程序的吞吐量,识别可并行的任务,

提高单线程子系统的响应性;

确保并发程序执行预期任务;

提高并发代码的性能和可伸缩性,

显示锁.原子变量,非阻塞算法,开发自定义的同步工具类.

设计原则:单一 ,开闭,里氏替换,接口隔离,依赖反转

进程之间交换数据:套接字,信号处理器,共享内存,信号量以及文件

多个程序同时进行,资源利用率,公平性，便利性。

线程不安全,一种常见的并发安全问题,成为竞台条件.,在多线程情况下,getValue是否会返回唯一的值,取决于运行时对线程操作的交替执行方式.

安全性问题:永远不会发生糟糕的问题,

活跃性问题:某件正确的事情最终会发生,串行程序中,无意中造成的死循环,导致循环之后的代码无法执行.线程a等待b执行,线程b陷入死循环中,导致线程a无法执行.

性能问题:整体会正确执行,但是效果不好,服务时间过长、响应不够灵敏、吞吐率过低、资源消耗过高、可伸缩性较低。多线程能够提升程序的性能，但也会带来程序的运行时开销。在多线程程序切换时，频繁的出现上下文切换，会造成极大的开销：保存和恢复执行上下文，丢失局部性。并且CPU时间将更多地花费在线程调度而不是线程运行上。线程共享数据时，必须使用同步机制，而这些机制往往会抑制某些编译器的优化，使内存缓存区中的数据无效，以及增加共享内存总线的同步流量。

RMI中含有列集和散集。远程对象必须注意两个线程安全性问题：正确地协同在多个对象中共享的状态，以及对远程对象本身状态的访问(由于同一个对象可能会在多个线程中被同时访问。

java的主要同步机制是关键字Synchronized,他提供了一种独占的加锁方式。同步还包括volatile类型的变量，显式锁，以及原子变量。

在并发编程中，由于不恰当的执行时序而出现不正确的结果是一种非常重要的情况。叫做竞态条件（Race Condition）。

竞态条件和数据竞争：