编写高效的线程安全类 PDF转换可能丢失图片或格式，建议阅读原文
https://www.100test.com/kao_ti2020/267/2021_2022__E7_BC_96_E5_86_99_E9_AB_98_E6_c104_267891.htm Java 编程语言为编写多线程应用程序提供强大的语言支持。但是，编写有用的、没有错误的多线程程序仍然比较困难。本文试图概述几种方法，程序员可用这几种方法来创建高效的线程安全类。并发性只有当要解决的问题需要一定程度的并发性时，程序员才会从多线程应用程序中受益。例如，如果打印队列应用程序仅支持一台打印机和一台客户机，则不应该将它编写为多线程的。一般说来，包含并发性的编码问题通常都包含一些可以并发执行的操作，同时也包含一些不可并发执行的操作。例如，为多个客户机和一个打印机提供服务的打印队列可以支持对打印的并发请求，但向打印机的输出必须是串行形式的。多线程实现还可以改善交互式应用程序的响应时间。Synchronized 关键字虽然多线程应用程序中的大多数操作都可以并行进行，但也有某些操作（如更新全局标志或处理共享文件）不能并行进行。在这些情况下，必须获得一个锁来防止其他线程在执行此操作的线程完成之前访问同一个方法。在 Java 程序中，这个锁是通过 synchronized 关键字提供的。清单 1 说明了它的用法。清单 1. 使用 synchronized 关键字来获取锁public class MaxScore { int max. public MaxScore() { max = 0. } public synchronized void currentScore(int s) { if(s> max) { max = s. } } public int max() { return max. }}这里，两个线程不能同时调用 currentScore() 方法；当一个线程工作时，另一个线程必须阻塞。但是，可以有任意数量的线程同时通过 max() 方法

访问最大值，因为 max() 不是同步方法，因此它与锁定无关。试考虑在 MaxScore 类中添加另一个方法的影响，该方法的实现如清单 2 所示。清单 2. 添加另一个方法public synchronized void reset() { max = 0. }这个方法（当被访问时）不仅将阻塞 reset() 方法的其他调用，而且也将阻塞 MaxScore 类的同一个实例中的 currentScore() 方法，因为这两个方法都访问同一个锁。如果两个方法必须不彼此阻塞，则程序员必须在更低的级别使用同步。清单 3 是另一种情况，其中两个同步的方法可能需要彼此独立。清单 3. 两个独立的同步方法import java.util.*.public class Jury { Vector members. Vector alternates. public Jury() { members = new Vector(12, 1). alternates = new Vector(12, 1). } public synchronized void addMember(String name) { members.add(name). } public synchronized void addAlt(String name) { alternates.add(name). } public synchronized Vector all() { Vector retval = new Vector(members). retval.addAll(alternates). return retval. }}此处，两个不同的线程可以将 members 和 alternates 添加到 Jury 对象中。请记住，synchronized 关键字既可用于方法，更一般地，也可用于任何代码块。清单 4 中的两段代码是等效的。 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com