Java线程知识的深入分析Java认证考试 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/629/2021_2022_Java_E7_BA_BF_E7_A8_8B_c104_629775.htm 一般来说,我们把正在计算机中执行的程序叫做"进程"(Process) ,而不将其称为程序(Program) 。所谓"线程"(Thread),是"进程"中某个单一顺序的控制流。新兴的操作系统,如Mac,Windows NT,Windows 95等,大多采用多线程的概念,把线 程视为基本执行单位。线程也是Java中的相当重要的组成部分之一。 甚至最简单的Applet也是由多个线程来完成的。在Java中, 任何一个Applet的paint()和0update()方法都是由AWT(Abstract Window Toolkit)绘图与事件处理线程调用的,而Applet 主要的里程碑方法init(),start(),stop() 和destory() 是由执行该Applet的应用调用的。 单线程的概念没有什么新的地方,真正有趣的是在一个程序中同时使用多个线程来完成不同的任务。某些地方用轻量进程(Lightweig ht Process)来代替线程,线程与真正进程的相似性在于它们都是单一顺序控制流。然而线程被认为轻量是由于它运行于整个程序的上下文内,能使用整个程序共有的资源和程序环境。 作为单一顺序控制流,在运行的程序内线程必须拥有一些资源作为必要的开销。例如,必须有执行堆栈和程序计数器在线程内执行的代码只在它的上下文中起作用,因此某些地方用"执行上下文"来代替"线程"。 线程属性 为了正确有效地使用线程,必须理解线程的各个方面并了解Java 实时系统。必须知道如何提供线程体、线程的生命周期、实时系统如何调度线程、线程组、什么是幽灵线程(Demo nThread)。 (1)线程体 所有的操作都发生在线程体中,在Java中线程体是从Thread类继承的run()

方法,或实现Runnable接口的类中的run()方法。当线程产生并初始化后,实时系统调用它的run()方法。run()方法内的代码实现所产生线程的行为,它是线程的主要部分。 (2)线程状态 附图表示了线程在它的生命周期内的任何时刻所能处的状态以及引起状态改变的方法。这图并不是完整的有限状态图,但基本概括了线程中比较感兴趣和普遍的方面。以下讨论有关线程生命周期以此为据。 ●新线程态(New Thread) 产生一个Thread对象就生成一个新线程。当线程处于"新线程"状态时,仅仅是一个空线程对象,它还没有分配到系统资源。因此只能启动或终止它。任何其他操作都会引发异常。 ●可运行态(Runnable) start()方法产生运行线程所必须的资源,调度线程执行,并且调用线程的run ()方法。在这时线程处于可运行态。该状态不称为运行态是因为这时的线程并不总是一直占用处理机。特别是对于只有一个处理机的PC而言,任何时刻只能有一个处于可运行态的线程占用处理 机。Java通过调度来实现多线程对处理机的共享。 ●非运行态(Not Runnable) 当以下事件发生时,线程进入非运行态。 ①suspend()方法被调用. ②sleep()方法被调用. ③线程使用wait()来等待条件变量. ④线程处于I/O等待 ●死亡态(Dead) 当run()方法返回,或别的线程调用stop()方法,线程进入死亡态 。通常Applet使用它的stop()方法来终止它产生的所有线程。 (3)线程优先级 虽然我们说线程是并发运行的。然而事实常常并非如此。正如前面谈到的,当系统中只有一个CPU时,以某种顺序在单CPU情况下执行多线程被称为调度(scheduling)。Java采用的是一种简单、固定的调度法,即固定优先级调度。这种算法是根据处于可运行态线程的相对优先级来实行调度。当线程产生时,它继承原线程的

优先级。在需要时可对优先级进行修改。在任何时刻,如果有多条线程等待运行, 系统选择优先级最高的可运行线程运行。只有当它停止、自动放弃、或由于某种原因成为非运行态低优先级的线程才能运行。如果两个线程具有相同的优先级,它们将被交替地运行。 Java实时系统的线程调度算法还是强制性的,在任何时刻,如果一个比其他线程优先级都高的线程的状态变为可运行态,实时系统将选择该线程来运行。 (4)幽灵线程 任何一个Java线程都能成为幽灵线程。它是作为运行于同一个进程内的对象和线程的服务提供者。例如,HotJava浏览器有一个称为" 后台图片阅读器"的幽灵线程,它为需要图片的对象和线程从文件系统或网络读入图片。幽灵线程是应用中典型的独立线程。它为同一应用中的其他对象和线程提供服务。幽灵线程的run()方法一般都是无限循环,等待服务请求。 (5)线程组 每个Java线程都是某个线程组的成员。线程组提供一种机制,使得多个线程集于一个对象内,能对它们实行整体操作。譬如,你能用一个方法调用来启动或挂起组内的所有线程。Java线程组由ThreadGroup类实现。当线程产生时,可以指定线程组或由实时系统将其放入某个缺省的线程组内。线程只能属于一个线程组,并且当线程产生后不能改变它所属的线程组。 多线程程序 对于多线程的好处这就不多说了。但是,它同样也带来了某些新的麻烦。只要在设计程序时特别小心留意,克服这些麻烦并不算太困难。 (1)同步线程 许多线程在执行中必须考虑与其他线程之间共享数据或协调执行状态。这就需要同步机制。在Java中每个对象都有一把锁与之对应。但Java不提供单独的lock和unlock操作。它由高层的结构隐式实现, 来保证操作的对应。(然而,我们注意到Java虚拟机提供

单独的monito renter和monitorexit指令来实现lock和unlock操作。) synchronized语句计算一个对象引用,试图对该对象完成锁操作, 并且在完成锁操作前停止处理。当锁操作完成synchronized语句体得到执行。当语句体执行完毕(无论正常或异常),解锁操作自动完成。作为面向对象的语言,synchronized经常与方法连用。一种比较好的办法是,如果某个变量由一个线程赋值并由别的线程引用或赋值,那么所有对该变量的访问都必须在某个synchromized语句或synchronized方法内。 现在假设一种情况：线程1与线程2都要访问某个数据区,并且要求线程1的访问先于线程2, 则这时仅用synchronized是不能解决问题的。这在Unix或Windows NT中可用Simaphore来实现。而Java并不提供。在Java中提供的是wait()和notify()机制。使用如下: synchronized method-1(…){ call by thread 1. // access data area. available=true. notify() } synchronized method-2(…){ // call by thread 2. while(!available) try{ wait(). // wait for notify(). }catch (Interrupted Exception e){ } // access data area } 其中available是类成员变量,置初值为false。如果在method-2中检查available为假,则调用wait()。 wait()的作用是使线程2进入非运行态,并且解锁。在这种情况下,method-1可以被线程1调用。当执行 notify()后。线程2由非运行态转变为可运行态。当method-1调用返回后。线程2 可重新对该对象加锁,加锁成功后执行wait()返回后的指令。这种机制也能适用于 其他更复杂的情况。 (2)死锁 如果程序中有几个竞争资源的并发线程，那么保证均衡是很重要的。系统均衡是指每个线程在执行过程中都能充分访问有限的资源。系统中没有饿死和死锁的线程。Java并不提供对死锁的检测机制

。对大多数的Java程序员来说防止死锁是一种较好的选择。最简单的防止死锁的方法是对竞争的资源引入序号，如果一个线程需要几个资源，那么它必须先得到小序号的资源，再申请大序号的资源。 小结 线程是Java中的重要内容,多线程是Java的一个特点。虽然Java的同步互斥不如某些系统那么丰富,但适当地使用它们也能收到满意的效果。 