考试指导：java多线程设计模式详解之二 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/138/2021_2022__E8_80_83_E8_AF_95_E6_8C_87_E5_c97_138817.htm wait()/notify() 通常，多线程之间需要协调工作。例如，浏览器的一个显示图片的线程displayThread想要执行显示图片的任务，必须等待下载线程downloadThread将该图片下载完毕。如果图片还没有下载完，displayThread可以暂停，当downloadThread完成了任务后，再通知displayThread “ 图片准备完毕，可以显示了 ”，这时，displayThread继续执行。 以上逻辑简单的说就是：如果条件不满足，则等待。当条件满足时，等待该条件的线程将被唤醒。在Java中，这个机制的实现依赖于wait/notify。等待机制与锁机制是密切关联的。例如： synchronized(obj) { while(!condition) { obj.wait(). } obj.doSomething(). } 当线程A获得了obj锁后，发现条件condition不满足，无法继续下一处理，于是线程A就wait()。 在另一线程B中，如果B更改了某些条件，使得线程A的condition条件满足了，就可以唤醒线程A： synchronized(obj) { condition = true. obj.notify(). } 需要注意的概念是： # 调用obj的wait(), notify()方法前，必须获得obj锁，也就是必须写在synchronized(obj) {...} 代码段内。 # 调用obj.wait()后，线程A就释放了obj的锁，否则线程B无法获得obj锁，也就无法在synchronized(obj) {...} 代码段内唤醒A。 # 当obj.wait()方法返回后，线程A需要再次获得obj锁，才能继续执行。 # 如果A1,A2,A3都在obj.wait()，则B调用obj.notify()只能唤醒A1,A2,A3中的一个（具体哪一个由JVM决定）。 # obj.notifyAll()则能全部唤醒A1,A2,A3，但是要继续执

行obj.wait()的下一条语句，必须获得obj锁，因此，A1,A2,A3只有一个有机会获得锁继续执行，例如A1，其余的需要等待A1释放obj锁之后才能继续执行。 # 当B调用obj.notify/notifyAll的时候，B正持有obj锁，因此，A1,A2,A3虽被唤醒，但是仍无法获得obj锁。直到B退出synchronized块，释放obj锁后，A1,A2,A3中的一个才有机会获得锁继续执行。 wait()/sleep()的区别 前面讲了wait/notify机制，Thread还有一个sleep()静态方法，它也能使线程暂停一段时间。sleep与wait的不同点是：sleep并不释放锁，并且sleep的暂停和wait暂停是不一样的。obj.wait会使线程进入obj对象的等待集合中并等待唤醒。 但是wait()和sleep()都可以通过interrupt()方法打断线程的暂停状态，从而使线程立刻抛出InterruptedException。 如果线程A希望立即结束线程B，则可以对线程B对应的Thread实例调用interrupt方法。如果此刻线程B正在wait/sleep/join，则线程B会立刻抛出InterruptedException，在catch() {} 中直接return即可安全地结束线程。 需要注意的是，InterruptedException是线程自己从内部抛出的，并不是interrupt()方法抛出的。对某一线程调用interrupt()时，如果该线程正在执行普通的代码，那么该线程根本就不会抛出InterruptedException。但是，一旦该线程进入到wait()/sleep()/join()后，就会立刻抛出InterruptedException。 GuardedSuspention GuardedSuspention模式主要思想是： 当条件不满足时，线程等待，直到条件满足时，等待该条件的线程被唤醒。 我们设计一个客户端线程和一个服务器线程，客户端线程不断发送请求给服务器线程，服务器线程不断处理请求。当请求队列为空时，服务器线程就必须等待，直到客

户端发送了请求。 先定义一个请求队列：Queue package com.crackj2ee.thread. import java.util.*. public class Queue { private List queue = new LinkedList(). public synchronized Request getRequest() { while(queue.size()==0) { try { this.wait(). } catch(InterruptedException ie) { return null. } } return (Request)queue.remove(0). } public synchronized void putRequest(Request request) { queue.add(request). this.notifyAll(). } } 蓝色部分就是服务器线程的等待条件，而客户端线程在放入了一个request后，就使服务器线程等待条件满足，于是唤醒服务器线程。 客户端线程：ClientThread package com.crackj2ee.thread. public class ClientThread extends Thread { private Queue queue. private String clientName. public ClientThread(Queue queue, String clientName) { this.queue = queue. this.clientName = clientName. } public String toString() { return "[ClientThread-" clientName "]". } public void run() { for(int i=0. i Request request = new Request("" (long)(Math.random()*10000)). System.out.println(this " send request: " request). queue.putRequest(request). 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