java类Timer和TimerTask的使用 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/144/2021_2022_java_E7_B1_BBTim_c104_144357.htm 这两个类使用起来非常方便，可以完成我们对定时器的绝大多数需求 Timer类是用来执行任务的类，它接受一个TimerTask做参数 Timer有两种执行任务的模式,最常用的是schedule,它可以以两种方式执行任务:1:在某个时间(Data)，2:在某个固定的时间之后(int delay).这两种方式都可以指定任务执行的频率.看个简单的例子:import java.io.IOException.import java.util.Timer.public class TimerTest ...{ public static void main(String[] args)...{ Timer timer = new Timer(). timer.schedule(new MyTask(), 1000, 2000).//在1秒后执行此任务,每次间隔2秒,如果传递一个Data参数,就可以在某个固定的时间执行这个任务. while(true)...{//这个是用来停止此任务的,否则就一直循环执行此任务了 try ...{ int ch = System.in.read(). if(ch-c==0)...{ timer.cancel().//使用这个方法退出任务 }} catch (IOException e) ...{ // TODO Auto-generated catch block e.printStackTrace(). }}} static class MyTask extends java.util.TimerTask...{ @Override public void run() ...{ // TODO Auto-generated method stub System.out.println("_____"). }}} 如果你使用的是JDK 5 ,还有一个scheduleAtFixedRate模式可以用,在这个模式下,Timer会尽量的让任务在一个固定的频率下运行,举例说明:在上面的例子中,我们想让MyTask在1秒钟后,每两秒钟执行一次,但是因为java不是实时的(其实java实时性很差.....),所以,我们在上个程序中表达的原义并不能够严格执行.如果我们调用的是scheduleAtFixedRate,那么,Timer会尽量让你

的Task执行的频率保持在2秒一次.运行上面的程序,假设使用的是scheduleAtFixedRate,那么下面的场景就是可能的:1秒钟后,MyTask 执行一次,因为系统繁忙,之后的2.5秒后MyTask 才得以执行第二次,然后,Timer记下了这个延迟,并尝试在下一个任务的时候弥补这个延迟,那么,1.5秒后,MyTask 将执行的三次."以固定的频率而不是固定的延迟时间去执行一个任务"果然很方便吧^_^下面给出一个复杂点的例子,其中告诉大家怎么退出单个TimerTask,怎么退出所有Taskpackage MyTimerTest.import java.io.IOException.import java.util.Timer./**//* * 本类给出了使用Timer和TimerTaske的主要方法,其中包括定制任务,添加任务 * 退出任务,退出定时器. * 因为TimerTask的status域是包级可访问的,所以没有办法在java.util.包外 * 得到其状态,这对编程造成一些不便 .我们不能判断某个Task的状态了. * */public class TimerTest ...{ public static void main(String[] args) ...{ Timer timer = new Timer(). MyTask myTask1 = new MyTask(). MyTask myTask2 = new MyTask(). myTask2.setInfo("myTask-2"). timer.schedule(myTask1, 1000, 2000). timer.scheduleAtFixedRate(myTask2, 2000, 3000). while (true) ...{ try ...{ byte[] info = new byte[1024]. int len = System.in.read(info). String strInfo = new String(info, 0, len, "GBK").//从控制台读出信息 if (strInfo.charAt(strInfo.length() - 1) ==) ...{ strInfo = strInfo.substring(0, strInfo.length() - 2). } if (strInfo.startsWith("Cancel-1")) ...{ myTask1.cancel().//退出单个任务 // 其实应该在这里判断myTask2是否也退出了,是的话就应该break.但是因为无法在包外得到 // myTask2的状态,所以,这里不能做出是否退出循环的判断. } else if

(strInfo.startsWith("Cancel-2")) ...{ myTask2.cancel(). } else if (strInfo.startsWith("Cancel-All")) ...{ timer.cancel().//退出Timer break. } else ...{ // 只对myTask1作出判断,偷个懒^_^ myTask1.setInfo(strInfo). } } catch (IOException e) ...{ // TODO Auto-generated catch block e.printStackTrace(). } } } static class MyTask extends java.util.TimerTask ...{ String info = "^_^". @Override public void run() ...{ // TODO Auto-generated method stub System.out.println(info). } public String getInfo() ...{ return info. } public void setInfo(String info) ...{ this.info = info. } }}