Java技巧：用匿名类来实现简化程序调试 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/234/2021_2022_Java_E6_8A_80_E5_B7_A7_c104_234247.htm 在Java中，匿名类（Anonymous inner classes）多用来处理事件（event handle）。但其实，它们对于debug也很有帮助。本文将介绍如何利用匿名类来简化你的debug。 我们该如何调试那些非自己源码的方法调用呢？比方说，对Jbutton.setEnable()的调用。Java提供的匿名类，可以很好的解决这个问题。 通常，当我们继承一个类时，我们可以通过提供新的方法来覆盖（override）该类中现有的方法： 英文代码public class MyButton extends JButton { public void setVisible(boolean visible) { // Rolling our own visibility }}在实例化（instantiate）MyButton类之后，任何对方法setVisible()的调用，都会调用上面代码中的setVisible()方法。可问题是，我们不想仅仅为了覆盖一个方法而继承整个类，尤其是所需的实例（instantiation）很有限的时候。匿名类使得我们能在实例化的同时覆盖方法。 如果我们只想在某个JButton对象中加入我们自己的可视逻辑，那么我们可以在申明这个button对象的同时重写这个方法： JButton myButton = new JButton() { public void setVisible(boolean visible) { // Rolling our own visibility }}.这段代码都做了什么？花括号（ { } ）中间的代码申明了setVisible()方法，并覆盖了JButton类中的那个，但这仅限于myButton对象。我们没有改变JButton类，也没有申明一个新类，我们仅给了一个特殊的JButton对象它自己的可视逻辑。 在面向对象术语中，myButton是一个从JButton类继承而来的无名，也就是匿名，类的对象。 这种

创建匿名类并同时覆盖方法的技术用在什么时候？假设你在编写一段Swing程序，在你向一个GUI物件（element）中添加一个event listener（假设叫作ActionListener）之前，你已经编写了一段这种机制的代码。现在，我们假设我们有个庞大的类，里面有很多按钮，但是有一个按钮时隐时现。你想知道为什么会这样。利用上面的代码并在setVisible()方法上设置断点。然后，当你运行你的程序时，你设置的断点就会在恰当的地方暂停程序。检查栈轨迹（stack trace），我们会发现没有按所预期的那样来调用setVisible()方法的原因并修复这个它。 匿名类在debug类似这种源码不可得的类的时候很有用。即便在源码可得的情况下，在大量使用的方法（如setVisible）上设置断点，也是件很麻烦的事情，因为我们在每个实现了setVisible()方法的类的对象上都要转入断点。而匿名类可针对某个特定的对象进行“ 外科手术 ”式的debug。