用Swing制作欢迎屏幕 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__E7_94_A8Swing_E5_88_c104_144987.htm 几乎所有时髦的应用都有一个欢迎屏幕。欢迎屏幕既是宣传产品的方法之一，而且在长时间的应用启动过程中，欢迎屏幕还用来表示应用正在准备过程中。下面是一个最简单的欢迎屏幕实现：class SplashWindow1 extends JWindow{public SplashWindow1(String filename, Frame f){super(f).JLabel l = new JLabel(new ImageIcon(filename)).getContentPane().add(l, BorderLayout.CENTER).pack().Dimension screenSize =Toolkit.getDefaultToolkit().getScreenSize().Dimension labelSize = l.getPreferredSize().setLocation(screenSize.width/2 - (labelSize.width/2),screenSize.height/2 - (labelSize.height/2)).setVisible(true).screenSize = null.labelSize = null.}}SplashWindow1类从Swing的JWindow派生。JWindow是一个容器，它没有其他窗口所具有的各种窗口元素，如标题条、窗口管理按钮，甚至连突出显示的边框也没有。因此，JWindow对于制作欢迎屏幕来说是非常合适的。上面的代码假定图形文件在当前目录。图形通过ImageIcon装入内存，然后它就被放到了JWindow的中心。接着，窗口被pack()，这使得Swing把窗口调整到适当的大小，最后窗口被移到了屏幕的中心。如果我们运行上面的程序，可以发现虽然欢迎画面确实出现在屏幕中央，但遗憾的，它却不会关闭！要关闭欢迎画面，我们需要加入更多的代码：class SplashWindow2 extends JWindow{public SplashWindow2(String filename, Frame

f){super(f).JLabel l = new JLabel(new ImageIcon(filename)).getContentPane().add(l, BorderLayout.CENTER).pack().Dimension screenSize =Toolkit.getDefaultToolkit().getScreenSize().Dimension labelSize = l.getPreferredSize().setLocation(screenSize.width/2 - (labelSize.width/2),screenSize.height/2 - (labelSize.height/2)).addMouseListener(new MouseAdapter(){public void mousePressed(MouseEvent e){setVisible(false).dispose().}}).setVisible(true).}}和原先的SplashWindow1类相比，这个SplashWindow2类唯一的区别在于多出了一个安装到JWindow上的匿名MouseListener。经过这个改动之后，用户可以点击欢迎屏幕关闭它。现在我们有了一个很不错的欢迎屏幕，它可以通过点击的方法关闭，但它不会自己消失。接下来我们要加入代码，使得欢迎屏幕在显示一定的时间之后自动消失。这里我们要考虑到运用线程。class SplashWindow3 extends JWindow{public SplashWindow3(String filename, Frame f, int waitTime){super(f).JLabel l = new JLabel(new ImageIcon(filename)).getContentPane().add(l, BorderLayout.CENTER).pack().Dimension screenSize =Toolkit.getDefaultToolkit().getScreenSize().Dimension labelSize = l.getPreferredSize().setLocation(screenSize.width/2 - (labelSize.width/2),screenSize.height/2 - (labelSize.height/2)).addMouseListener(new MouseAdapter(){public void mousePressed(MouseEvent e){setVisible(false).dispose().}}).final int pause = waitTime.final

Runnable closerRunner = new Runnable(){public void run(){setVisible(false).dispose().}}.Runnable waitRunner = new Runnable(){public void run(){try{Thread.sleep(pause).SwingUtilities.invokeAndWait(closerRunner).}catch(Exception e){e.printStackTrace().// 能够捕获InvocationTargetException// 能够捕获InterruptedException}}}.setVisible(true).Thread splashThread = new Thread(waitRunner, "SplashThread").splashThread.start().}}这里的基本思路是利用一个在一定时间内暂停等待的Thread对象。在上面的代码中，线程的暂停时间是4秒。当这个线程唤醒时，它将关闭欢迎屏幕。由于Swing是非线程安全的，除非代码在事件分派线程上执行，否则它就不应该影响任何UI组件的状态。所谓事件分派线程，就是Swing中负责绘图和事件处理的线程。为了解决这个问题，Swing设计者赋予我们安全地把Runnable对象加入UI事件队列的能力。在本例中，我们用可运行对象closerRunner完成最关键的工作。我们把可运行对象传入SwingUtilities.invokeAndWait()静态方法，然后wingUtilities.invokeAndWait()进行所有未完成的UI操作，并执行传递给该方法的可运行对象closerRunner的run方法。通过运用一个独立的线程负责欢迎屏幕的关闭操作，应用担负起了显示和关闭欢迎屏幕之间的所有操作。如果要让欢迎屏幕总是显示且用户不能关闭它，你必须删除那些隐藏欢迎屏幕的代码。如果要让欢迎屏幕只能由用户手工关闭，你可以象使用任何其他JWindow对象一样调用SplashWindow3对象上的setVisible(false)和dispose()方法。总而言之，借助于SwingUtilities.invokeAndWait()方法，我们可以安全地创建出

多线程欢迎屏幕。具体实现时，欢迎屏幕可以由用户点击关闭，也可以在一定的时间之后自动关闭。Swing所支持的线程模型使得应用在显示欢迎屏幕之后仍能够响应其他事件和处理其他任务。 