Java开发中的线程安全选择与Swing PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/145/2021_2022_Java_E5_BC_80_E5_8F_91_c104_145387.htm Swing API的设计目标是强大、灵活和易用。特别地，我们希望能让程序员们方便地建立新的Swing组件，不论是从头开始还是通过扩展我们所提供的一些组件。 出于这个目的，我们不要求Swing组件支持多线程访问。相反，我们向组件发送请求并在单一线程中执行请求。本文讨论线程和Swing组件。目的不仅是为了帮助你以线程安全的方式使用Swing API，而且解释了我们为什么会选择现在这样的线程方案。 本文包括以下内容： 单线程规则：Swing线程在同一时刻仅能被一个线程所访问。一般来说，这个线程是事件派发线程（event-dispatching thread）。 规则的例外：有些操作保证是线程安全的。 事件分发：如果你需要从事件处理（event-handling）或绘制代码以外的地方访问UI，那么你可以使用SwingUtilities类的invokeLater()或invokeAndWait()方法。 创建线程：如果你需要创建一个线程比如用来处理一些耗费大量计算能力或受I/O能力限制的工作你可以使用一个线程工具类如SwingWorker或Timer。 为什么我们这样实现Swing：我们将用一些关于Swing的线程安全的背景资料来结束这篇文章。 Swing的规则是： 一旦Swing组件被具现化（realized），所有可能影响或依赖于组件状态的代码都应该在事件派发线程中执行。 这个规则可能听起来有点吓人，但对许多简单的程序来说，你用不着为线程问题操心。在我们深入如何撰写Swing代码之前，让我们先来定义两个术语：具现化（realized）和事件派发线程

（event-dispatching thread）。 具现化的意思是组建的paint()方法已经或可能会被调用。一个作为顶级窗口的Swing组件当调用以下方法时将被具现化：setVisible(true)、show()或（可能令你惊奇）pack()。当一个窗口被具现化，它包含的所有组件都被具现化。另一个具现化一个组件的方法是将它放入到一个已经具现化的容器中。稍后你会看到一些对组件具现化的例子。 事件派发线程是执行绘制和事件处理的线程。例如，paint()和actionPerformed()方法会自动在事件派发线程中执行。另一个将代码放到事件派发线程中执行的方法是使用SwingUtilities类的invokeLater()方法。 