AWT，SWT和Swing都有相似的事件监听器Java认证考试 PDF转换可能丢失图片或格式，建议阅读原文https://www.100test.com/kao_ti2020/644/2021_2022_AWT_EF_BC_8CSWT_E5_c104_644753.htm 如多线程模型中所做的那样，然而这同样会引入在多线程模型中出现的同步问题。许多GUI工具集都有自己先天性的机制来解决这一问题，例如，在组件树上的组合锁。开发者不需要为如何安全编程而操心。这样的工具集被成为线程安全的。AWT就是其中之一。然而，由于不必要的同步使得建立这样的GUI系统过于负责并造成了额外的开销。因此，Swing和SWT被设计为非线程安全的，这意味着开发者必须谨慎地实现他们的多线程任务。SWT和Swing在运行时线程安全行为上有一个小小的区别。SWT总是检查改变组件的操作是否在事件分发线程上执行。这样，开发者就能够发现同步问题。而Swing不这样，这是Swing的一个不知之初，这其实并不难实现。 Event Dispatching Thread事件分发线程 AWT读取操作系统中的基本事件并在java代码中处理它们。AWT事件分发线程被命名为?AWT-{OS}-Thread。这个线程由AWT 系统隐式启动。当AWT应用程序启动时，这个线程在背后启动，在操作系统上抽取和移除事件。当诸如按钮动作这样的逻辑事件被触发时，它传递给注册在按钮上的操作监听器。开发者也能为AWT组件编写鼠标，键盘和绘制事件的事件监听器。这通常通过扩展AWT Canvas组件来完成。这一组件支持了所有已提供的事件，而且你可以通过重写事件处理方法，实现一个自定义的组件。 然而，在Swing中，这是一个不同的线程。Swing把AWT事件作为自身事件系统的一个输入。它获

取AWT事件并做一些初始化处理，产生一个对应的Swing事件并把它放到自己的事件队列上。Swing也有自己的事件分发线程，通常命名为EventQueue-0。这个线程从事件队列中抽取Swing事件，就如同AWT从操作系统中抽取那样。然后它把事件分发给目标组件。通常事件首先被分发给组件的顶层容器，然后由顶层容器的 dispatch方法处理，它可能被再分发或重定向到一个Swing组件，在那里继续由自己的监听器进行处理。 例如，当一个鼠标移过一个JButton或JFrame时，一个指向JFrame的AWT事件在AWT线程上触发。AWT线程检查它是否需要作更多的处理，然后把它包装成一个Swing的MouseEvent对象并把它添加到EventQueue队列中。当获得MouseEvent事件后，EventQueue-0抽取这个事件并判断出目标Swing组件。这里，这个组件是JButton。然后它产生了一个包含相对坐标位置和事件源的新的MouseEvent重定向到这个JButton上，然后调用这个 JButton的dispatch以继续分发。JButton的dispatch过滤事件给特定的方法最终实现由鼠标监听器在该点上的分发的点击。 SWT更类似于AWT。唯一的区别是它要求开发者显式地书写事件循环代码。然而底层的实现细节是不同于AWT的。看看SWT的读取和分发事件代码，它会让你想起MFC的代码风格。 Event Listener事件监听器 AWT，SWT和Swing都有相似的事件监听器模型。它们都使用观察者模式，组件和监听器的连接方式是把监听器添加到组件上,这组成了一个对象网络的模型。当事件被触发并分发给组件，组件调用它的监听器以处理事件。一个监听器也可以继续分发事件给后续的处理，或产生一个新的事件并把它广播到网络中的其他节点上。基本上有两种不同的广播事件方

式。一种是同步调用监听器。另一种是异步地将事件发送回队列，它将在新一轮的事件分发中被分发出去。 除了直接发送给队列的方式，Swing还有一些其他的分发异步事件的方法。其中之一是调用SwingUtilities 或EventQueue 的invokeLater，这一方法包装一个Runnable对象到事件中并把它发送给事件队列。这确保了Runnable的run方法能在事件分发线程上执行，保证了线程安全。实际上，Swing的Timer好SwingWorker基于这一机制实现。SWT也有相应的发送异步事件的方式。它的 invokeLater的对应方法是display.asyncExec，它以一个Runnable对象作为参数。 AWT AWT是Sun不推荐使用的工具集。然而它在许多非桌面环境如移动或嵌入式设备中有着自己的优势。 更少的内存。它对运行在有限环境中的GUI程序的开发，是合适的。 1.更少的启动事件。由于AWT组件是本地由操作系统实现的。绝大多数的二进制代码已经在如系统启动的时候被预装载了，这降低了它的启动事件。 2.更好的响应。由于本地组件由操作系统渲染。 3.从java 1.x时代就为JRE支持的标准GUI工具集，你不用单独安装它，你不用担心平台差异的问题。 4.成熟稳定的。它能够正常工作并很少使你的程序崩溃。 然而，事物都有它们不好的一面。让我们来例数它吧。 1. 更少的组件类型。表和树这些重要的组件缺失了。它们是桌面应用程序中普遍使用的。 2.缺乏丰富的组件特征。按钮不支持图片附着。这很明显是它遵循的设计原则造成的。 3.不支持Look And Feel。AWT被设计为使用本地组件。因此，它依赖系统来提供Look And Feel支持。如果目标系统并不支持这一特性，那么AWT将无法改变它的Look And Feel。 4.无扩展性。AWT的组件是本地组件。JVM中

的AWT类实例实际只是包含本地组件的引用。唯一的扩展点是AWT的Canvas组件，你可以从零开始创建自定义组件。然而无法继承和重用一个已有的AWT组件。 SWT SWT有如下优势： 1.丰富的组件类型。SWT提供了种类繁多的组件，从基础组件如按钮和标签到高级的表格和树。 2.相对的丰富组件特性。尽管SWT也遵循最大公倍数原则，它采用模拟的方式重新设计了对更多组件特性的支持。所以同AWT相比，它有着相对丰富的组件特性。 3.更快的响应时间。基于和AWT同样的原因，SWT组件包装了本地组件，由操作系统实现渲染。操作系统通常对渲染处理做了优化，保存GUI二进制代码为标准库，减少了内存的使用，提高了响应性能。 4.更少的内存消耗。既然操作系统为本地组件提供了优化，这一点就容易理解了。 以上就是AWT，SWT和Swing都有相似的事件监听器的内容。 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