 比较AWT和Swing 1.AWT和Swing组件体系 a）Swing：通过在父组件上动态调用add（）和remove（）方法，来对组件树进行动态维护，因此，可以存在没有父组件的子组件。 b）SWT：通过在构造函数的参数中，指定父组件，因此，不可能存在没有父组件的子组件。 2.AWT和Swing组件绘制 a）Swing：通过调用组件的paint（）方法来进行组件的绘制，组件进一步对他的子组件调用paint（），从而完成整棵组件树的绘制。通过子类化，重写paint（）方法，可以完全定制一个组件的绘制。 b）SWT：SWT组件只是本地平台上实际组件的一个代理，实际组件采用win32编程方式进行绘制，绘制的过程不在JVM中完成。因此，子类化也不能对绘制进行完全定制。当SWT组件进行绘制时，他在完成了本地的绘制以后，会返回JVM，然后对所有已注册的PaintListener发出通知，因此，通过注册PaintListener可以对组件绘制进行一定程度上的修改。 3.AWT和Swing事件模型 a）Swing中的事件会跟据当前的焦点进行发送。父组件不能对事件进行过滤。Swing中的事件监听器都是有类型的，因此每个组件所能支持的事件类型是通过方法名来限定的。不能扩展。 b）SWT中的事件也是根据当前的焦点进行发送。父组件不能对事件进行过滤。SWT中支持有类型的监听器，也支持无类型的监听器，通过addListener（type，listener），我们可以为组件添加任何类型的监听器。 4.AWT和Swing包设计 a）Swing是一个纯粹的组件库，他没有

图形的操作，他的绘制最终转发给Java 2D来完成。 b）SWT包含了组件库和图形库，他的结构与AWT比较接近。 5. 其它AWT和SWT实现原理不同：AWT控件相当于是一笔一画绣出来的，参加 java.awt.Component.paint（Graphics g），性能差；而SWT调用了操作系统后台原生库，org.eclipse.swt.widgets.Control类型中就没有类似的paint方法了，但是有一个接口org.eclipse.swt.graphics.Drawable.实现原理的不同带来了性能和用户视觉的差异。 SWING 是基于AWT提供的MVC不完全实现，JFACE是基于SWT提供的MVC不完全实现。两者的核心作用都是提供了viewer和模型封装的概念，从设计实现层面看，SWING更加唯美一些，JFACE则更加侧重于实用。SWING和 JFACE两者都偏重于行为控制上下文，而对UI数据（例如，一个文本框控件中的内容）的管理都没有做太多的设计。JFACE中略有改进，提供了 setData的概念。JFACE是为了Eclipse而生的，虽然后来 RCP出来了，又做了其他的包装宣传，典型的体现就是提供了JFace Text Framework和其他一些用户构建Eclipse元素的UI支持，也提供了一些系统资源管理（ImageRegsitry、 ResourceManager等）等附加功能。 SWING框架与代码很优美，但效率低，特别是高级控件如各种选择框超慢，与Windows本身不协调，使用体验差；而且再怎么使用LookAndFeel，还是不美观，总是感觉画的不清晰。 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