AWT 是Abstract Window ToolKit (抽象窗口工具包)的缩写，这个工具包提供了一套与本地图形界面进行交互的接口。AWT 中的图形函数与操作系统所提供的图形函数之间有着一一对应的关系，我们把它称为peers。 也就是说，当我们利用 AWT 来构件图形用户界面的时候，我们实际上是在利用操作系统所提供的图形库。由于不同操作系统的图形库所提供的功能是不一样的，在一个平台上存在的功能在另外一个平台上则可能不存在。为了实现Java语言所宣称的"一次编译，到处运行"的概念，AWT 不得不通过牺牲功能来实现其平台无关性，也就是说，AWT 所提供的图形功能是各种通用型操作系统所提供的图形功能的交集。由于AWT 是依靠本地方法来实现其功能的，我们通常把AWT控件称为重量级控件。

Swing 是在AWT的基础上构建的一套新的图形界面系统，它提供了AWT 所能够提供的所有功能，并且用纯粹的Java代码对AWT 的功能进行了大幅度的扩充。例如说并不是所有的操作系统都提供了对树形控件的支持， Swing 利用了AWT 中所提供的基本作图方法对树形控件进行模拟。由于 Swing 控件是用100%的Java代码来实现的，因此在一个平台上设计的树形控件可以在其他平台上使用。由于在Swing 中没有使用本地方法来实现图形功能，我们通常把Swing控件称为轻量级控件。

AWT和Swing之间的基本区别：AWT 是基于本地方法的C/C++程序，其运行速度比较快；Swing是基于AWT 的Java程序，其运行速度比较慢。对于一个嵌入式应用来说，目标平台的硬件资源往往非常有限，而应用程序的运行速度又是项目中至关重要的因素。在这种矛盾的情况下，简单而高效的AWT 当然成了嵌入式Java的第一选择。而在普通的基于PC或者是工作站的标准Java应用中，硬件资源对应用程序所造成的限制往往不是项目中的关键因素，所以在标准版的Java中则提倡使用Swing， 也就是通过牺牲速度来实现应用程序的功能。

通俗的话:

AWT 是抽象窗口组件工具包，是 java 最早的用于编写图形节目应用程序的开发包。

Swing 是为了解决 AWT 存在的问题而新开发的包，它以 AWT 为基础的。

==============================================================================================================

[Java GUI编程中AWT/swing/SWT的优缺点](http://www.cnblogs.com/dugang/archive/2010/10/22/1858478.html)

在[**Java**](http://lib.csdn.net/base/java) GUI编程的过程中，选择何种工具包一直的众多程序员的“十字路口”，而目前市场上最为普遍的就是AWT、swing和SWT。

AWT

　　AWT是Abstract Window Toolkit（抽象窗口工具包）的缩写，是SUN专门针对[**java**](http://lib.csdn.net/base/java) GUI编程提供的最早的也是最初的开发工具包，用来建立和设置[**Java**](http://lib.csdn.net/base/java)图形用户界面，这个工具包提供了一套与本地图形界面交互的接口。AWT 中的图形函数与[**操作系统**](http://lib.csdn.net/base/operatingsystem)所提供的图形函数之间有着一一对应的关系（peers）。也就是说，当我们利用 AWT 来构件图形用户界面的时候，实际上是在利用操作系统所提供的图形库。不过由于不同操作系统的图形库所提供的功能是不完全一样，所以在一个平台上存在的功能在另外一个平台上则可能不存在。这就导致一些应用程序在[**测试**](http://lib.csdn.net/base/softwaretest)时界面非常美观，而一旦移植到其他的操作系统平台上后就可能变得“惨不忍睹”。为了实现Java语言"一次编译，到处运行"，AWT 不得不通过牺牲功能来实现其平台无关性，其所提供的图形功能被定格为各种通用型操作系统所提供的图形功能的交集。由于AWT 是依靠本地方法来实现其功能的，所以通常把AWT组件称为重量级组件。

　　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**](http://lib.csdn.net/base/embeddeddevelopment)设备中有着自己的优势：

1.更少的内存：对运行在有限环境中的GUI程序的开发，是合适的。

2.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。

3.更好的响应：由于本地组件由操作系统渲染。

4.成熟稳定的：能够正常工作并很少使你的程序崩溃。

然而事物具有两面性，AWT也有着许多缺点：

1.更少组件类型：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。

3.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，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

Swing

　　Swing 是在AWT的基础上构建的一套新的图形界面系统，是JFC（Java Foundation Class）的一部分，是试图解决AWT缺点的一个尝试。它提供了AWT 所能够提供的所有功能，并且用纯粹的Java代码对AWT 的功能进行了大幅度的扩充。所有的swing组件实际上也是AWT的一部分。Swing 对基于对等体的组件使用的术语是重量级，对于模拟的组件使用的术语是轻量级。实际上，Swing 可以支持在一个 GUI 中混合使用重量级组件和轻量级组件，不过一般将其称之为轻量级组件。

Swing是三者中最强大的GUI工具集，同时它也是SUN推荐使用的GUI工具集，通过几个本版的修改和扩展后，现在的swing在某些领域有着明显优势：

1.丰富的组件类型:Swing提供了非常广泛的标准组件。这些组件和SWT一样丰富。基于它良好的可扩展性，除了标准组件，Swing还提供了大量的第三方组件。许多商业或开源的Swing组件库在开发多年后都已经可以方便地获取了。

2.丰富的组件特性:Swing不仅包含了所有平台上的特性，它还支持根据程序所运行的平台来添加额外特性。Swing组件特性遵循特定原则，易于扩展，因此能够提供较SWT和AWT更多的功能。

3.好的组件API模型支持:Swing遵循MVC模式，这是一种非常成功的设计模式。它的API成熟并设计良好。经过多年的演化，Swing组件APIs变得越来越强大，灵活和可扩展。它的API设计被认为是最成功的GUI API之一。较之SWT和AWT更面向对象，也更灵活而可扩展。

4.标准的GUI库：Swing和AWT一样是JRE中的标准库。因此，你不用单独地将它们随你的应用程序一起分发。它们是平台无关的，不用担心平台兼容性。

5.成熟稳定：由于它是纯Java实现的，不会有SWT的兼容性问题。Swing在每个平台上都有相同的性能，不会有明显的性能差异。

6.可扩展和灵活性。Swing完全由Java代码实现。Swing基于MVC的结构使得它可以发挥Java作为一门面向对象语言的优势。它提供了许总体上良好的性能。

　　当然，swing也有着许多不足之处：比如swing比AWT和SWT更多的内存消耗。Swing自己实现了所有组件。因此，它在运行时装载了大量的类。而在运行时Java在堆上创建小的对象导致了额外的堆空间消耗。而许多小的对象难以有效地被垃圾回收机制回收。因此，Swing应用程序通常会因无法及时回收冗余的对象而导致性能下降。

SWT

　　SWT是Standard WidgetToolkit的缩写，是由IBM构建的一个新的GUI库，其目的在于尝试彻底解决AWT和swing带来的诸多问题，提供比AWT更为丰富的组件集。SWT和swing一样痛Java代码模拟了一些平台缺失的组件，不过与 AWT 的概念相比，SWT 是一个低级的 GUI 工具包，在构建SWT的过程中，构建者从 AWT 和 Swing 实现中学习了很多经验，他们试图构建一个集二者优点于一体而没有二者的缺点的系统。因此SWT可以说是AWT与swing的融合体。

　　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。

不足之处：

1.不在JRE的标准库中。因此必须将它和程序捆绑在一起，并为所要支持的每个操作系统创建单独的安装程序。

2.不够成熟和稳定。SWT因其设计上的一些缺陷，如资源管理，Windows友好等，被认为是不稳定的。它可以在Windows上表现得很好，但在其他操作系统上，它经常是不稳定且容易崩溃的。这很大程度上是因为它把资源管理交给开发者来处理，而并不是所有的开发人员能够正确地处理这些。

3.在非Windows平台下的性能不高

4.无Look AndFeel 支持。和AWT同样的原因。

5.不可扩展。

　　综上所述，在对GUI编程时的工具包选择得根据具体项目决定：若是需要用低内存来运行GUI程序，使用AWT会是一个不错的选择，而如果只考虑平台的移植性可优先考虑SWT/swing