AWT vs Swing

By: Josh Fletcher

Abstract: *A brief comparison of Swing and the AWT when used for GUI developement.*

AWT vs Swing

When developing a Java program it is important to select the appropriate Java Graphical User Interface (GUI) components. There are two basic sets of components that you will most likely build your Java programs with. These two groups of components are called the Abstract Window Toolkit (AWT) and Swing. Both of these groups of components are part of the Java Foundation Classes (JFC).

An Overview of the AWT

AWT stands for Abstract Window ToolKit. The Abstract Window Toolkit supports GUI Java programming. It is a portable GUI library for stand-alone applications and/or applets. The Abstract Window Toolkit provides the connection between your application and the native GUI. The AWT provides a high level of abstraction for your Java program since it hides you from the underlying details of the GUI your program will be running on.

AWT features include:

- A rich set of user interface components.
- A robust event-handling model.
- Graphics and imaging tools, including shape, color, and font classes.
- Layout managers, for flexible window layouts that don't depend on a particular window size or screen resolution.
- Data transfer classes, for cut-and-paste through the native platform clipboard.

The AWT components depend on native code counterparts (called peers) to handle their functionality. Thus, these components are often called "heavyweight" components.

An Overview of Swing

Swing implements a set of GUI components that build on AWT technology and provide a pluggable look and feel. Swing is implemented entirely in the Java programming language, and is based on the JDK 1.1 Lightweight UI Framework.

Swing features include:

- All the features of AWT.
- 100% Pure Java certified versions of the existing AWT component set (Button, Scrollbar, Label, etc.).
- A rich set of higher-level components (such as tree view, list box, and tabbed panes).
- Pure Java design, no reliance on peers.
- Pluggable Look and Feel.

Swing components do not depend on peers to handle their functionality. Thus, these components are often called "lightweight" components.

AWT vs. Swing

There are, of course, both pros and cons to using either set of components from the JFC in your Java

applications. Here is a summary:

AWT:
Pros

- Speed: use of native peers speeds component performance.
- Applet Portability: most Web browsers support AWT classes so AWT applets can run without the Java plugin.
- Look and Feel: AWT components more closely reflect the look and feel of the OS they run on.

Cons

- Portability: use of native peers creates platform specific limitations. Some components may not function at all on some platforms.
- Third Party Development: the majority of component makers, including Borland and Sun, base new component development on Swing components. There is a much smaller set of AWT components available, thus placing the burden on the programmer to create his or her own AWT-based components.
- Features: AWT components do not support features like icons and tool-tips.

Swing:
Pros

- Portability: Pure Java design provides for fewer platform specific limitations.
- Behavior: Pure Java design allows for a greater range of behavior for Swing components since they are not limited by the native peers that AWT uses.
- Features: Swing supports a wider range of features like icons and pop-up tool-tips for components.
- Vendor Support: Swing development is more active. Sun puts much more energy into making Swing robust.
- Look and Feel: The pluggable look and feel lets you design a single set of GUI components that can automatically have the look and feel of any OS platform (Microsoft Windows, Solaris, Macintosh, etc.). It also makes it easier to make global changes to your Java programs that provide greater accessibility (like picking a hi-contrast color scheme or changing all the fonts in all dialogs, etc.).

Cons

- Applet Portability: Most Web browsers do not include the Swing classes, so the Java plugin must be used.
- Performance: Swing components are generally slower and buggier than AWT, due to both the fact that they are pure Java and to video issues on various platforms. Since Swing components handle their own painting (rather than using native API's like DirectX on Windows) you may run into graphical glitches.
- Look and Feel: Even when Swing components are set to use the look and feel of the OS they are run on, they may not look like their native counterparts.

In general, AWT components are appropriate for simple applet development or development that targets a specific platform (i.e. the Java program will run on only one platform).

For most any other Java GUI development you will want to use Swing components. Also note that the Borland value-added components included with JBuilder, like dbSwing and JBCL, are based on Swing components so if you wish to use these components you will want to base your development on Swing.

Tip: Whether you choose to use Swing or AWT for your Java program development, you should avoid mixing the two. There are many painting problems that can occur when you mix heavyweight AWT components with lightweight Swing.