Richard G Baldwin (512) 223-4758, baldwin@austin.cc.tx.us,
http://www2.austin.cc.tx.us/baldwin/

The AWT and Swing, A Preview

Java Programming, Lecture Notes # 73, Revised 05/11/98.

Preface

Students in Prof. Baldwin's **Intermediate Java Programming** classes at ACC are responsible for knowing and understanding all of the material in this lesson.

Discussion

This lesson provides a very brief preview of some of what you can expect to find in subsequent lessons regarding the Abstract Windows Toolkit (**AWT**) and the **Swing** component set.

The user interface of a modern computer program often involves techniques to activate many of the human senses. We use *icons, text boxes, images, sound, boardroom graphics, etc.*

Up to this point, we haven't been too concerned with these aspects of programming because there was a lot that you needed to learn to prepare yourself for understanding material of this sort. That is about to change.

Much of the actual programming that you will do with Java will involve those aspects of the interface that we commonly refer to as the **Graphical User Interface (GUI)**.

As of 5/10/98, there are two primary packages that are used for **GUI** programming under JDK 1.1.6:

1. java.**awt**.*
2. com.sun.java.**swing**.*

There are, of course, numerous other packages that are used in support of these two.

The **AWT** material was made available to Java programmers early in the life of Java. This was the original material that was used to create graphical user interfaces. Major improvements to the **AWT** were introduced with the release of JDK 1.1.

The **Swing** components became available in released form for use with JDK 1.1 around the beginning of 1998. These components added significantly to the ability of the programmer to create GUIs, both in terms of functionality and cosmetics.

The capability and cosmetics of the AWT were very limited but **Swing** made GUI programming in Java competitive in the real world. A Java programmer no longer need apologize for the quality of the GUIs that she can create.

We expect that these two packages may become more integrated (causing changes in your **import** statements) with the release of JDK 1.2, (probably sometime in 1998) but hopefully the concepts involved won't be greatly different.

As of 3/5/97, there were more than fifty classes defined in package **java.awt**. We will discuss some of the more important **AWT** classes in subsequent lessons.

As of 5/10/98, the *com.sun.java.swing* package contains more than 75 classes and about 20 interfaces. You might expect, therefore, that learning to use this material effectively won't be a trivial task.

It is very important to understand that **Swing** is an extension of, and not a replacement for the AWT. While it is true that there is some overlap (for example a **Swing JButton** component might be viewed as an improved functional replacement for an **AWT Button** component, and once you begin using **Swing** buttons you may choose to never again use an **AWT** button), the basic functionality of **Swing** is built upon the functionality of the **AWT**.

Therefore, as students, we cannot simply skip over an understanding of the **AWT** and move on to **Swing**. The **AWT** is the foundation for **Swing**.

We must first understand the **AWT** and then understand how **Swing** extends and improves on the **AWT**. I will attempt to integrate an understanding of both the **AWT** and **Swing** in the remaining lessons in these Java tutorials.

We will begin by introducing you to a few simple components of each type and use these components to teach you about such topics as event-driven programming, layout, graphics, etc. Then, time permitting, we will dig a little deeper into the more complex aspects of both the **AWT** and **Swing** components and other features.

What I won't do is show you a lot of pictures of various **AWT** and **Swing** components as is the case with many books and other tutorials (although such pictures can be important for an appreciation of GUI programming.). (Have you noticed how many Java books use copies of the JavaSoft documentation as filler material to make the book appear to contain more information than it actually contains? At least half of many of the books currently in print is nothing more than a reproduction of the documentation that you can download for free from JavaSoft. Oh well, enough of that!)

If you want to see some pictures of **AWT** and **Swing** components (which would be only natural), you can create them yourself on your own computer screen.

For examples of the **AWT** components, simply look in the folders in the software that you downloaded from JavaSoft. When you install JDK 1.1.6, a folder named "demo" will be created that contains about two-dozen sample programs. Many of these sample programs have graphical user interfaces that make use of the **AWT**. Just run the programs to see examples of the use of the **AWT**.

When you download and install **Swing** 1.0.1, a folder named "examples" will be created. This folder contains about nine folders, each of which contains a demonstration application or applet that makes use of **Swing**. You can run these programs to see the examples on your computer screen.

A particularly interesting demonstration application is the one named **SwingSet**. One of the new components in **Swing** is a tabbed pane that looks much like a common cardboard file folder with a labeled tab on the top, bottom, left, or right. The **AWT** doesn't contain such a component.

This demonstration starts with about twenty such tabbed panes on the screen, each one of which demonstrates one aspect of the use of **Swing**. By clicking on each of the labeled tabs, you can select and exercise one aspect of **Swing**. In addition, there are five menus that contain selections, some of which impact the behavior of some aspect of the demonstration.

While you are there, pay attention to the fact that virtually all of the **Swing** components are also containers, so it is possible to cause other items (such as images) to be contained in components such as buttons and menus.

Take a look at the pane labeled *RadioButtons* and see how two different images of JavaSoft's little creature named Duke can be made to function as a radio button. In this case, the selected Duke is waving while the unselected Dukes aren't waving.

Duke shows up again under *ToggleButtons* where the button which has been toggled has Duke animated in a child's **swing**.

The *Checkboxes* pane uses light bulbs that either are or are not illuminated to illustrate selection of **Checkbox** items.

The examples on the *Slider* pane are truly impressive (the **AWT** doesn't have a slider component, although it is possible to use a **ScrollBar** as a crude slider).

Take a look at the *ListBox* pane to see another example of using images inside of a component.

The *DebugGraphics* pane demonstrates how to run your program in slow motion so that you can see how the components are assembled for debugging purposes. Note that a **Slider** is used to control the speed of assembly of the components.

And of course, every where you turn in this demo, you will see *tool tips* that are not a part of the **AWT**. For a little comic relief, take a look at the *ToolTips* pane.

Don't forget to pull down the *Options* menu and select the "look and feel" of the different panes as you view them.

Actually, words are inadequate to describe what you are going to find when you install and run the **SwingSet** demonstration. To use a corny phrase made famous by an old TV commercial (which many of you are probably too young to remember), "Try it, you'll like it."

-end-