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

The AWT Package, An Overview

Java Programming, Lecture Notes # 110, Revised 02/21/98.

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.

JDK 1.1 was formally released on February 18, 1997. This lesson was originally written on March 5, 1997 using the software and documentation in the JDK 1.1 download package. Additional updates have been made since then.

Introduction

This lesson is primarily a preview of what you can expect to find in several lessons immediately following this one. This and the next several lessons concentrate on the package **java.awt** where most of the functionality exists for providing the user interface to your application or applet.

- Note that it was the AWT portion of Java that experienced the most significant changes in the release of the Java Development Kit, version 1.1. Much of what you may have learned earlier about version 1.0 was

```
replaced using new concepts. These lessons will concentrate on the use
of JDK 1.1 rather than JDK 1.0.
```

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

This and the next several lessons will concentrate on those aspects of the interface that we commonly refer to as the **Graphical User Interface (GUI)**. We will leave other aspects of the interface, such as sound, to be covered in subsequent lessons.

Many of the sample programs in the earlier lessons on event handling made use of simple GUI forms without providing much of an explanation as to what was going on. The next few lessons will attempt to rectify that situation by explaining many aspects of creating a Graphical User Interface.

Package java.awt Classes

As mentioned above, much of the material in this and the next several lessons will be based on the classes of the package **java.awt**.

As of 3/5/97, there were more than fifty classes defined in package **java.awt**. Of that total, we will be mainly concerned with the ones discussed in the following sections. The classes are grouped in the general order that we will be studying them and each grouping will usually be presented as a separate lesson. However, some groups are so large that it may be necessary to break the group up into more than one lesson.

We will also be making heavy use of the classes in the **java.awt.event** package, but you are already familiar with most of those classes from your study of the previous lessons on event handling in JDK 1.1.

Arranging Components in Containers

We can place our components in their containers using absolute position coordinates, or we can use any of several different layout managers to do the job. Using layout managers is considered by many to be the safer approach because this approach tends to automatically compensate for differences in screen resolution between platforms.

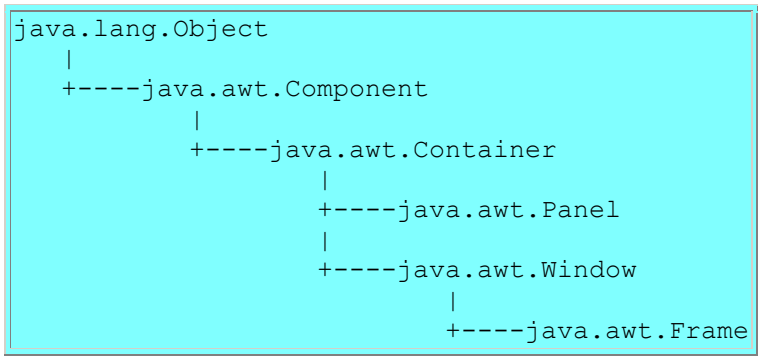
The different layout managers are defined in classes having the following names:

- **BorderLayout**
- **CardLayout**
- **FlowLayout**
- **GridBagLayout**
- **GridBagConstraints**
- **GridLayout**

In addition to the layout manager classes, there is a class named **GridBagConstraints** that is used to work with the **GridBagLayout** class as well as a class named **Insets** that is used to work with other classes.

The Container Classes

The following hierarchy diagram shows where the container classes **Panel, Window,** and **Frame** fit into the overall inheritance hierarchy. We will discuss these three classes as a group, and also discuss their relationship with their superclasses: **Container, Component,** and **Object.**



Generally, these classes are used to contain components or instances of other classes.

The Non-Text Input Components

The components in the next group are generally used to allow the user to input (non-text) information to the program, although in some circumstances they could also be used to output information to the user as well. **Non-text** in this context means that the user can input information without the requirement to use the keys on the keyboard.

All of these classes extend the class **Component**.

- **Button**
- **Checkbox**
- **Choice**
- **List**

The class named **CheckboxGroup** does not extend **Component**, but extends **Object** instead. As the name suggests, objects of this class can be used to group **Checkbox** components so that they behave in a mutually exclusive fashion.

The lesson on non-text input will discuss the classes listed above.

Text Input and Output Components

The lesson on text input and output will deal with the classes **TextField** and **TextArea** along with the class **Label**.

The **TextField** class can be used to produce a component that will accept one line of user text as input. It can also be used to display one line of text in a "non-editable" fashion..

The **TextArea** class can be used to produce a component that will accept multiple lines of user text as input. It can also be used to display multiple lines of text in a "non-editable" fashion.

TextField and **TextArea** extend **TextComponent** which extends **Component**.

The **Label** class can be used to display one line of text. This component is inherently non-editable and that attribute cannot be modified by the program.

Using a Scrollbar Component for Data Input

A previous lesson in the series on event handling discussed the use of the **Scrollbar** component in considerable depth. We will let that discussion suffice for out investigation of the **Scrollbar** component.

Dialogs

In the lesson on Dialogs, we will investigate the use of the **Dialog** class. This is a class that produces a dialog, or a window that takes input from the user.

This lesson will also investigate the **FileDialog** class. This class can be used to produce an object that displays a file selection dialog.

The Canvas Component

A **Canvas** component is a generic component which needs to be subclassed in order to add functionality. We will investigate various uses of this class in the lesson on the **Canvas** component.

Menus

The inheritance hierarchy for menus is as shown below.

```
java.lang.Object
        |
        +----MenuShortcut
        |
        +----java.awt.MenuComponent
                |
                +----java.awt.MenuBar
```

```
                    |
                    +----java.awt.MenuItem
                              |
                              +----java.awt.Menu
                              |
                              +----java.awt.CheckboxMenuItem
                              |
                              +----java.awt.PopupMenu
```

As you might suspect from looking at this diagram, there are a number of interesting issues associated with menus, and we will investigate those issues in the lesson on menus.

Graphics - Working with Shapes

The **Graphics** class, which extends the **Object** class, is the abstract base class for all graphics contexts allowing an application to draw onto components or onto off-screen images.

A **Graphics** object encapsulates the state information needed for the various rendering operations that Java supports. This state information includes:

- The Component to draw on
- A translation origin for rendering and clipping coordinates
- The current clip
- The current color
- The current font
- The current logical pixel operation function (XOR or Paint)
- The current XOR alternation color (see setXORMode)

Graphics is a large topic which will encompass several different lessons.

The first lesson in the **Graphics** series will investigate the use of the following classes for the realization of objects of the type indicated by the name of the class.

- **Rectangle**
- **Point**
- **Polygon**
- **Dimension**

Graphics - Working With Fonts

The lesson on graphics and fonts will concentrate on the classes **Font** and **FontMetrics**. As you can probably surmise from the names, these classes let you control the font of output text, and also let you determine information about the size of text produced with a particular font.

Graphics - Working With Images

This lesson on images will concentrate on the classes in the following list:

- **Image**
- **MediaTracker**
- **Color**
- **SystemColor**
- **ScrollPane**

Working with the PrintJob Class

As the name implies, this lesson will be concerned with the new printing capability of JDK 1.1.

Working with the Toolkit Class

The **ToolKit** class "is used to bind the abstract AWT classes to a particular native toolkit implementation." There are some interesting methods in this class and we will take a look at them.

Review

Since this lesson is primarily a preview of subsequent lessons, there is no reason to provide review material for this lesson.

-end-