Supplement: Pluggable Look and Feel
For Introduction to Java Programming
By Y. Daniel Liang

Lightweight components consume fewer resources and can be transparent, but they lack the AWT's platform-specific look-and-feel advantage. To address this problem, a new pluggable look-and-feel feature was introduced in Java.
The pluggable look-and-feel feature lets you design a single set of GUI components that automatically has the look-and-feel of any OS platform. The implementation of this feature is independent of the underlying native GUI, yet it can imitate the native behavior of the native GUI.
Currently, Java supports the following three look-and-feel styles:

- Metal
- Motif
- Windows

To see an example that demonstrates these three styles, change the directory to c:\book, and type the following command at the DOS prompt: **java -jar SimpleExample.jar**

Figure 1 shows a sample run of the program.

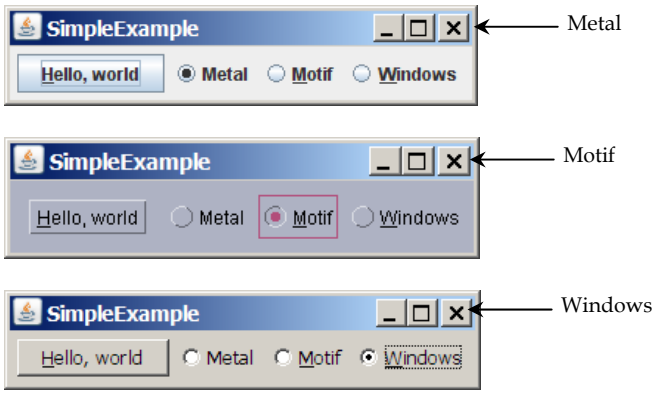

Figure 1

The SimpleExample demonstrates three look-and-feel styles.

The Metal style, also known as the *Java style*, gives you a consistent look regardless of operating system. The Windows style is currently only available on Windows due to Windows copyright restrictions. The Motif style is used on Unix operating systems.
The javax.swing.UIManager class manages the look-and-feel of the user interface. You can use one of the following three methods to set the look-and-feel for Metal, Motif, or

Windows:

```
UIManager.setLookAndFeel
  (UIManager.getCrossPlatformLookAndFeelClassName());
UIManager.setLookAndFeel
  (new com.sun.java.swing.plaf.motif.MotifLookAndFeel());
UIManager.setLookAndFeel
  (new com.sun.java.swing.plaf.windows.WindowsLookAndFeel());
```

The setLookAndFeel method throws UnsupportedLookAndFeelException, so you have to put the method inside a try-catch block for it to compile. To ensure that the setting takes effect, the setLookAndFeel method should be executed before any of the components are instantiated. Thus, you can put the code in a static block, as shown below:

```
static {
  try {
    // Set a look-and-feel, e.g.,

    // UIManager.setLookAndFeel

    //   (UIManager.getCrossPlatformLookAndFeelClassName());

  }
  catch (UnsupportedLookAndFeelException ex) {}
}
```

Static initialization blocks are executed when the class is loaded. For more information on static initialization blocks, please refer to Supplement III.J, "Initialization Blocks."