Section Thirty – Look and Feel (LAF)

In this section the Developer will learn…

- How to develop an application with a different look and feel …

LAF – Using setLookandFeel()

Java has been developed to be used on a variety of different platforms, however it might be that the Developer wishes to impose a different style across all of the processes in the system, this is particularly good if Users are a mixture of Mac and Window users etc, both would expect their system to look like a typical system on their Operating System.

Also some Java Swing Components default to a Mac look and feel even if being run in a Windows environment so the Developer may have to set the LAF for Windows to match all of the Components.

LAFs are also commercially available to give totally unique appearance to the system.

LAF – Using setLookandFeel()

The Developer will need to use the UIManager and will need to import the following …

import javax.swing.UIManager;

The standard way of setting look and feel is as follows, note the use of the exception handling …

try { UIManager.setLookAndFeel(<style>); }
catch (Exception e) { }

Note that the Developer has chosen to ignore an exception with this process …if the requested look and feel isnt available, then the default will display instead.

LAF – Standard LAF Styles

Before looking at using third party LAFs, here we examine the built in LAFs, bear in mind this course is built on a Windows environment therefore only LAFs for this platform will be displayed …

The Developer can run the following to display the available LAFs in a list …

```
import javax.swing.UIManager;

public class laf {
        public static void main(String[] args) {
            UIManager.LookAndFeelInfo[] lafs = UIManager.getInstalledLookAndFeels();
            for (UIManager.LookAndFeelInfo laf: lafs){
                    System.out.println(laf);
                }
        }
}
```

LAF – Standard LAF Styles

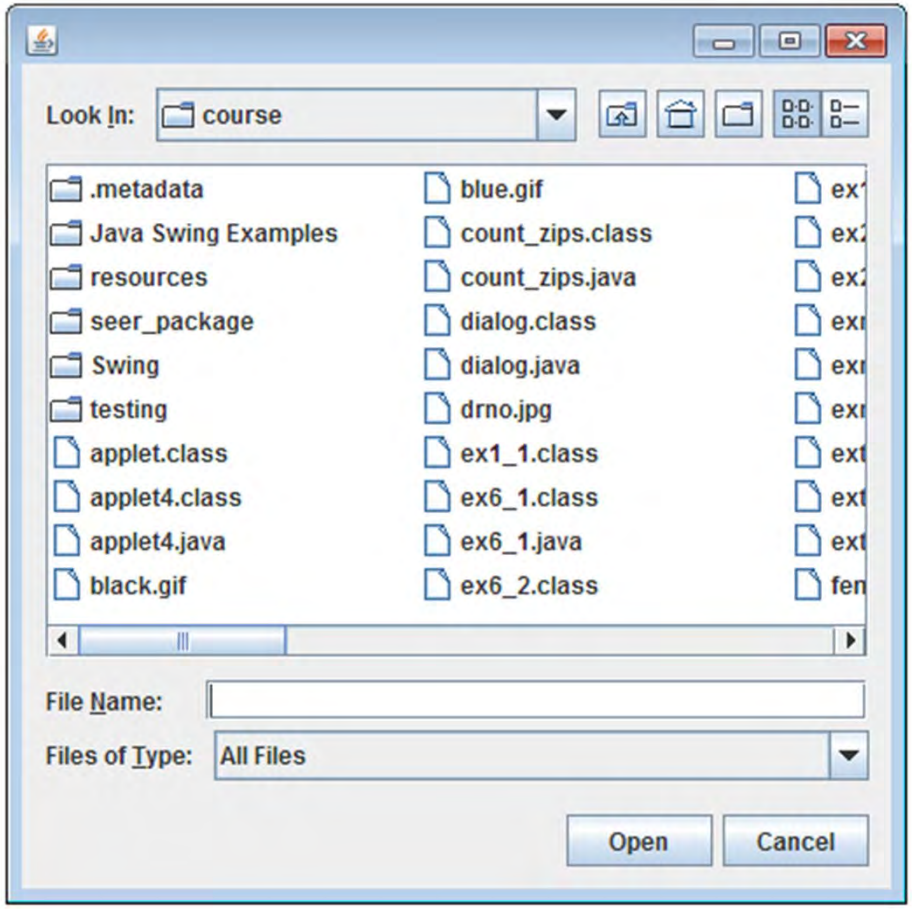
The output is as follows …

javax.swing.UIManager$LookAndFeelInfo[Metal javax.swing.plaf.metal.MetalLookAndFeel]
javax.swing.UIManager$LookAndFeelInfo[Nimbus javax.swing.plaf.nimbus.NimbusLookAndFeel]
javax.swing.UIManager$LookAndFeelInfo[CDE/Motif com.sun.java.swing.plaf.motif.MotifLookAndFeel]
javax.swing.UIManager$LookAndFeelInfo[Windows com.sun.java.swing.plaf.windows.WindowsLookAndFeel]
javax.swing.UIManager$LookAndFeelInfo[Windows Classic com.sun.java.swing.plaf.windows.WindowsClassicLookAndFeel]

These next few slides show how they 'look and feel' !!, note how the JFileChooser changes not only its colour and font but its structure too …

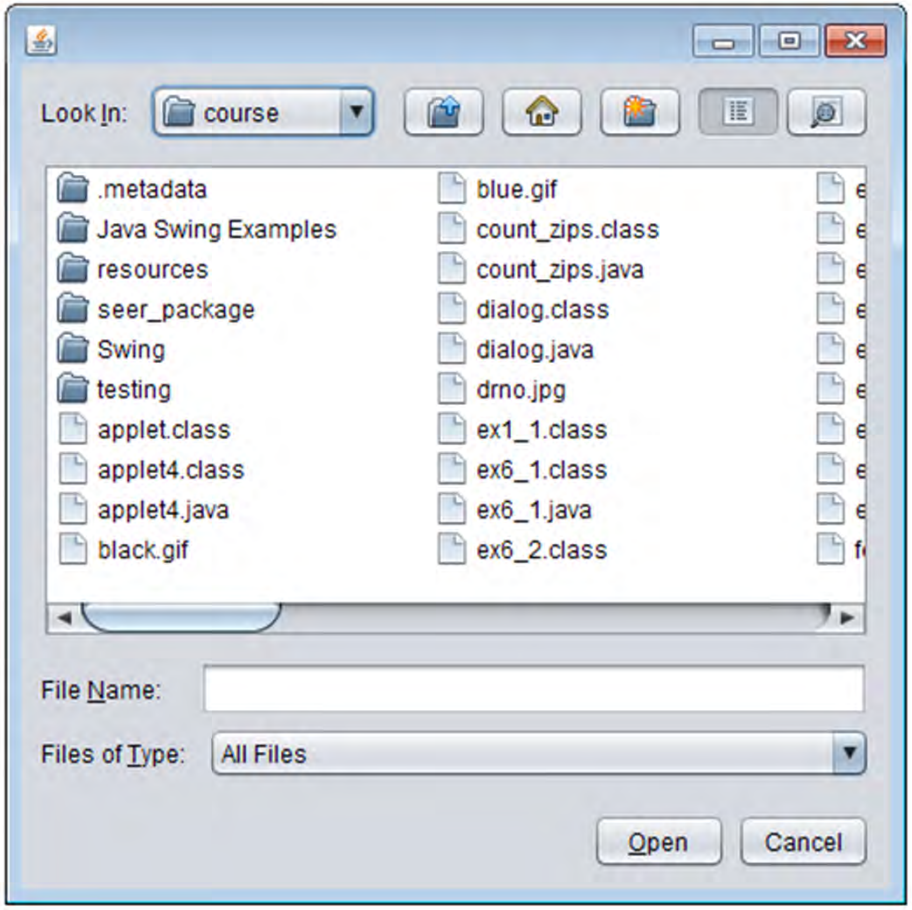
LAF – javax.swing.plaf.metal.MetalLookAndFeel

UIManager.setLookAndFeel("javax.swing.plaf.metal.MetalLookAndFeel")



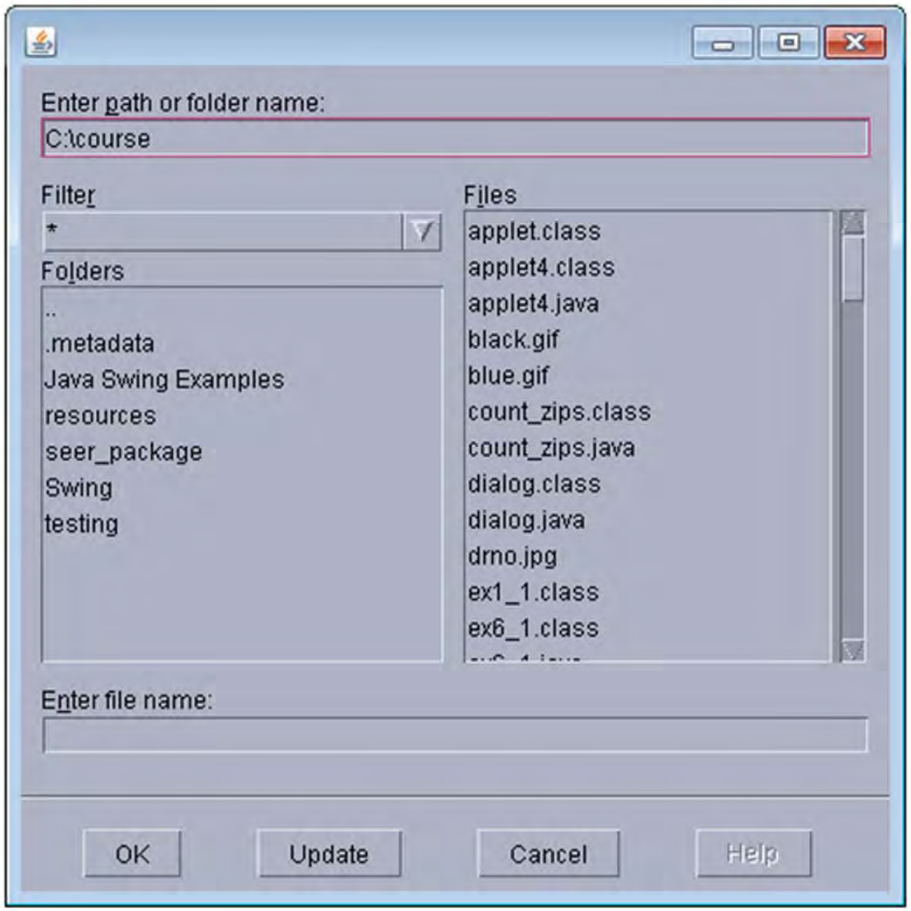
LAF – javax.swing.plaf.nimbus.NimbusLookAndFeel

UIManager.setLookAndFeel("javax.swing.plaf.nimbus.NimbusLookAndFeel")



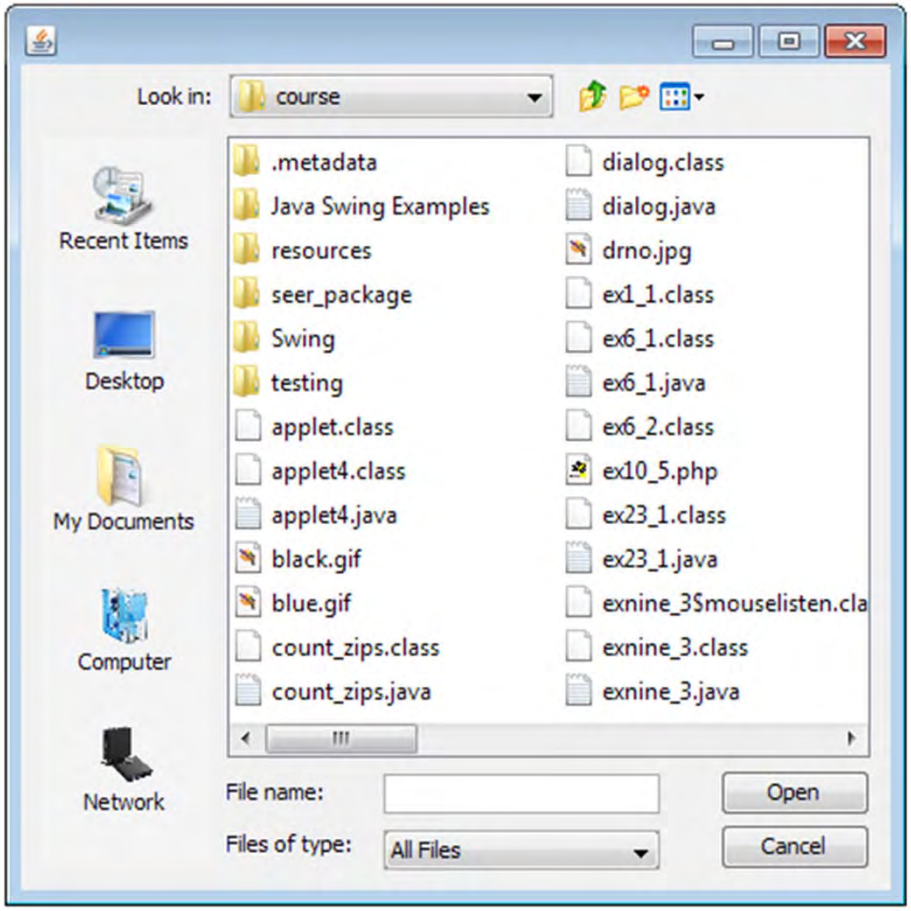
LAF – com.sun.java.swing.plaf.motif.MotifLookAndFeel

UIManager.setLookAndFeel("com.sun.java.swing.plaf.motif.MotifLookAndFeel")



LAF – com.sun.java.swing.plaf.windows.WindowsLookAndFeel

UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLookAndFeel")



LAF – com.sun.java.swing.plaf.windows.WindowsClassicLookAndFeel

UIManager.setLookAndFeel(
"com.sun.java.swing.plaf.windows.WindowsClassicLookAndFeel ")