Module 5

Introduction to Java/Swing

Java is commonly used for deploying applications across a network. Compiled Java code may be distributed to different machine architectures, and a native-code interpreter on each architecture interprets the Java code. The core functions found in the Java interpreter are called the JFC (Java Foundation Classes). JFC provides generally useful classes, including classes for GUIs, accessability and 2D drawing. The original GUI classes in Java are known as AWT - the Abstract Windowing Toolkit. AWT provides basic GUI functions such as buttons, frames and dialogs, and is implemented in native code in the Java interpreter.

By contrast, Swing is not implemented in native code - instead it is implemented in AWT. Swing and AWT can (and normally do) coexist - we may use the buttons from Swing, alongside AWT event handlers.

The advantages of Swing are:

1. Consistent look-and-feel - The look and feel is consistent across platforms.
2. Pluggable look-and-feel - The look and feel can be switched on-the-fly.
3. High-level widgets - the Swing components are useful and flexible.

In general, the Swing components are easier to use than similar AWT components.

5.1 How not to use Swing

The same concerns that applied to Tcl/Tk deployment apply to the use of Swing. If the target computers are slow, then the interpreter overhead may make the application frustratingly slow. With the rate of increase in speed in processors, this concern is minimized.

Processor intensive applications written in Java often seem to make the GUI appear slow and unresponsive. This is probably due to internal thread scheduling techniques in the interpreter.

5.2 Getting started

There are quite a few development environments for building Java applications and applets, and two of them are suggested for use in this course. However - if you have something better, or that you feel more comfortable with, please just use that. The systems are:

- **j2sdk1.3.1** - the Java development kit from Sun. It includes Java compilers, interpreters, debuggers and demo software, and local copies of it for WinXX and LINUX are found here:
  - http://www.comp.nus.edu.sg/~cs3283/ftp/Java/j2sdk-1_3_1_02-win.exe
  - http://www.comp.nus.edu.sg/~cs3283/ftp/Java/j2sdk-1_3_1_02-linux-i386.bin
- **Netbeans** - A GUI builder for Java applications and applets, again for WinXX and LINUX:
  - http://www.comp.nus.edu.sg/~cs3283/ftp/Java/NetBeansIDE-release331.exe
  - http://www.comp.nus.edu.sg/~cs3283/ftp/Java/NetBeansIDE-release331.tar.gz

Each of these systems is documented and described at public web sites - look at Sun's Java web site, and http://www.netbeans.org. In addition - there are local copies of some of the documentation here:

- The **JFC API** at http://www.comp.nus.edu.sg/~cs3283/ftp/Java/jfcapi/
- The Netbeans **API** at http://www.comp.nus.edu.sg/~cs3283/ftp/Java/OpenAPIs/
- The **Java tutorial** at http://www.comp.nus.edu.sg/~cs3283/ftp/Java/JavaTutorial/
- **Swing Connect** at http://www.comp.nus.edu.sg/~cs3283/ftp/Java/swingConnect/

Once you have installed the j2sdk, find the file called SwingSet2.jar, inside the demo heirarchy somewhere, and change to the directory. Then try:

```
java -jar SwingSet2.jar
```

5.3 Swing programming

In this course I hope to clarify the general style of Swing applications, and show sufficient examples to build *menu'd* GUI applications with interesting graphical interactions. The same strategy was used in the introduction to Tcl/Tk. A good book that covers this material in detail is

The JFC Swing Tutorial, by Kathy Walrath and Mary Campione.

The toplevel components provided by Swing are:

1. **JApplet** - for applets within web pages
2. **JDialog** - for dialog boxes
3. **JFrame** - for building applications

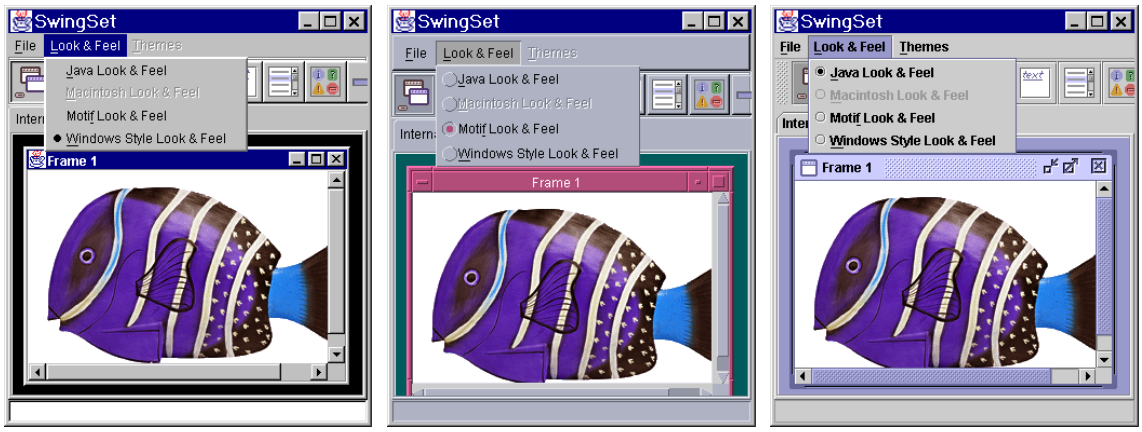
All other Swing components derive from the **JComponent** class. **JComponent** provides

- **Tool tips** - little windows with explanations
- **Pluggable look and feel** - as described
- **Layour management** - items within the component
- **Keyboard action management** - Hot keys and so on.
- And other facilities

Swing implements an MVC architecture.

5.3.1 Pluggable look and feel

It is relatively easy to change the look and feel of an application - here are three:



If you wished to use the WinXX look-and-feel, in the main of your application, you can make the following call:

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

5.4 Example application

It is traditional to begin with a “Hello World” example, but I will start with “Hello Singapore”, and you will have to move up to “Hello World” as you progress.

CODE LISTING **t2.java**

```
public class t2 extends javax.swing.JFrame {
    public t2() {
        initComponents();
    }
    private void initComponents() {
        jLabel2 = new javax.swing.JLabel();
        addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent evt) {
                exitForm(evt);
            }
        });
        jLabel2.setText("Hello Singapore!");
        jLabel2.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        getContentPane().add(jLabel2, java.awt.BorderLayout.CENTER);
        pack();
    }
    private void exitForm(java.awt.event.WindowEvent evt) {
        System.exit(0);
    }
    public static void main(String args[]) {
        new t2().show();
    }
    private javax.swing.JLabel jLabel2;
}
```

This code should not require much explanation - it just instantiates a **JLabel**, and sets the text field. Perhaps the only explanation needed is why it is so large! The code is generated from a GUI builder, and follows a particular software architecture. In this presentation of Swing, I will use the same, despite the possibility of smaller code-size applications.

When we compile and run this application we get:

The call to **getContentPane** returns the **contentPane** object for the frame - this is a generic AWT container for components associated with each **JFrame**. The **addWindowListener** call is from **java.awt.Window**, and adds the specified window listener to receive window events from this window.

5.5 Example applet

An equivalent Hello-world applet:

CODE LISTING **HelloWorldApp.java**

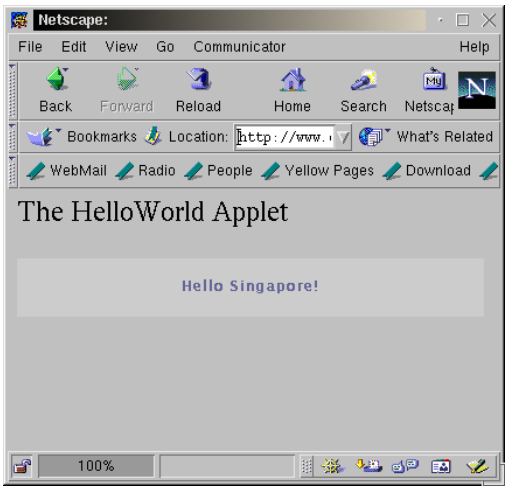
```
public class HelloWorldApp extends javax.swing.JApplet {
    public HelloWorldApp() {
        initComponents();
    }
    private void initComponents() {
        jLabel1 = new javax.swing.JLabel();
        jLabel1.setText("Hello Singapore!");
        jLabel1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        getContentPane().add(jLabel1, java.awt.BorderLayout.CENTER);
    }
    private javax.swing.JLabel jLabel1;
}
```

This code follows the same structure - it just instantiates a **JLabel**, and sets the text field, although in this code, the class extends a **JApplet** instead of a **JFrame**. When we compile and run this application we get a **HelloWorldApp.class** file, which has to be referenced in a web page:

CODE LISTING **HelloWorldApp.txt**

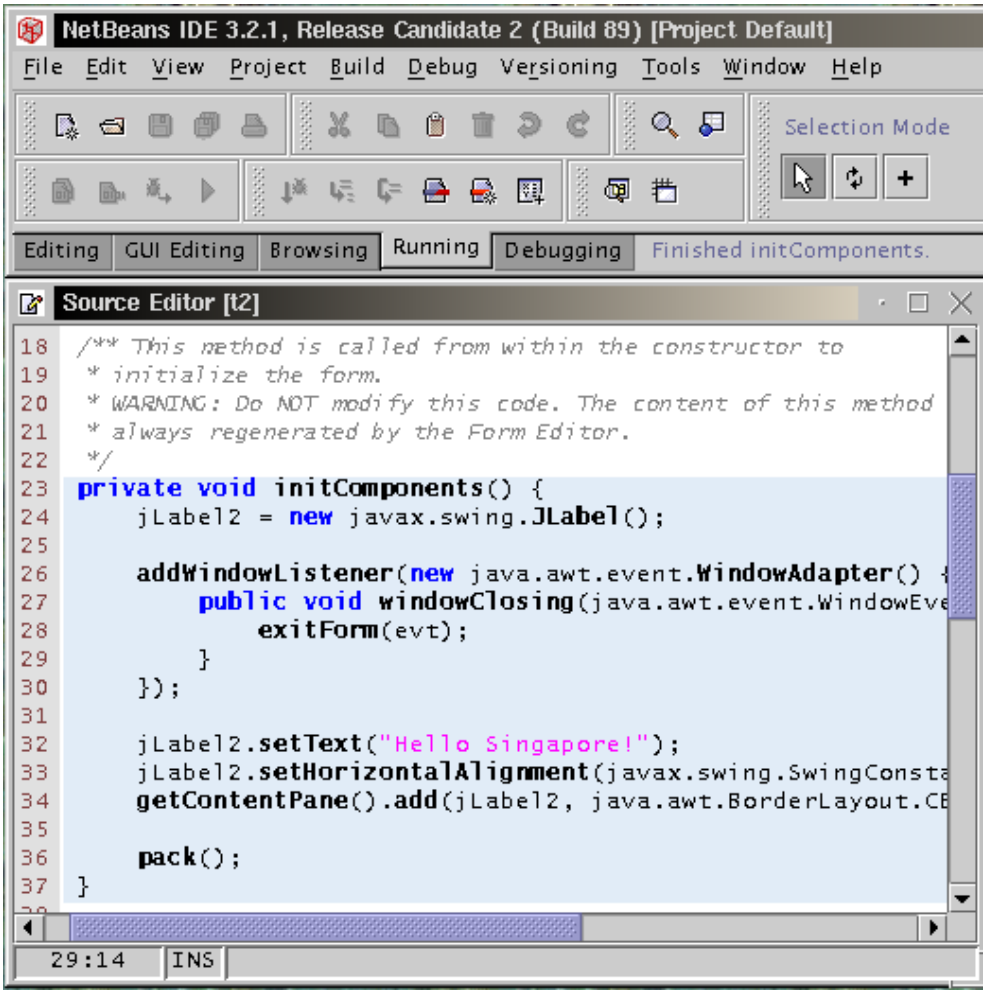
```
<BASE HREF="http://www.comp.nus.edu.sg/~hugh/swing/">
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<html>
The HelloWorld Applet <p>
<EMBED type         = "application/x-java-applet;version=1.1.2"
       java_CODE    = "HelloWorldApp.class"
       java_ARCHIVE = "applets.jar"
       WIDTH        = 400
       HEIGHT       = 50   ></EMBED>
</HTML>
```

The end result is:



5.6 Using the netbeans IDE

Simple programs like the ones just presented may be created using the GUI builder found in **net-beans** (http://www.netbeans.org), using a very small number of button presses and keystrokes. Here is a screen shot:



5.7 Summary of topics

In this module, we introduced the following topics:

- Tool sets for Java/Swing
- The relationship between JFC, Java and Swing.
- Simple first programs

Tutorial 6 - questions for week 7 (Feb 20, 2007)

1. What is meant by "*the MVC architecture*" mentioned in section 5.3?
2. Investigate how you would create a "*ToolTip*" in Tcl/Tk - give a small code segment which demonstrates the *ToolTip*.
3. Investigate how you would create a "*ToolTip*" in Java/Swing - give a small code segment which demonstrates the *ToolTip*.
4. Write a minimal Java/Swing application which puts up a single **File** menu with a **Quit** item in it.
5. The **javax.swing.UIManager** class is used to manipulate the look-and-feel of an application - as seen in section 5.3.1. How can you discover which look-and-feel strategies are implemented in the Java development environment?

Further study

- The **JFC API** at http://www.comp.nus.edu.sg/~cs3283/ftp/Java/jfcapi/
- The Netbeans **API** at http://www.comp.nus.edu.sg/~cs3283/ftp/Java/OpenAPIs/
- The **Java tutorial** at http://www.comp.nus.edu.sg/~cs3283/ftp/Java/JavaTutorial/
- **Swing Connect** at http://www.comp.nus.edu.sg/~cs3283/ftp/Java/swingConnect/
