The Abstract Windowing Toolkit

- Since Java was first released, its user interface facilities have been a significant weakness
  - The Abstract Windowing Toolkit (AWT) was part of the JDK from the beginning, but it really was not sufficient to support a complex user interface
- JDK 1.1 fixed a number of problems, and most notably, it introduced a new event model. It did not make any major additions to the basic components

Java Foundation Classes

- In April 1997, JavaSoft announced the Java Foundation Classes (JFC).
  - A major part of the JFC is a new set of user interface components called Swing.

Swing

- The Swing classes are used to build GUIs
  - Swing does not stand for anything
  - Swing is built on top of the 1.1/1.2 AWT libraries
- Swing makes 3 major improvements on the AWT
  - does not rely on the platform’s native components
  - it supports “Pluggable Look-and-Feel” or PLAF
  - it is based on the Model-View-Controller (MVC)
GUI Packages

- **AWT**
  - java.awt
  - java.awt.color
  - java.awt.datatransfer
  - java.awt.event
  - java.awt.font
  - java.awt.geom
  - java.awt.image
  - ...

- **Swing**
  - javax.accessibility
  - javax.swing
  - javax.swing.colorchooser
  - javax.swing.event
  - javax.swing.filechooser
  - javax.swing.plaf
  - javax.swing.table
  - javax.swing.text.html
  - javax.swing.tree
  - ...

Components

- A GUI consists of different graphic Component objects which are combined into a hierarchy using Container objects.
- **Component class**
  - An abstract class for GUI components such as menus, buttons, labels, lists, etc.
- **Container**
  - An abstract class that extends Component. Containers can hold multiple components.

Weighing Components

- Sun make a distinction between **lightweight** and **heavyweight** components
  - Lightweight components are not dependent on native peers to render themselves. They are coded in Java.
  - Heavyweight components are rendered by the host operating system. They are resources managed by the underlying window manager.
Heavyweight Components

• Heavyweight components were unwieldy for two reasons
  – Equivalent components on different platforms do not necessarily act alike.
  – The look and feel of each component was tied to the host operating system
• Almost all Swing components are lightweight except
  – JApplet, JFrame, JDialog, and JWindow

Additional Swing Features

• Swing also provides
  – A wide variety of components (tables, trees, sliders, progress bars, internal frame, …)
  – Swing components can have tooltips placed over them.
  – Arbitrary keyboard events can be bound to components.
  – Additional debugging support.
  – Support for parsing and displaying HTML based information.

Applets versus Applications

• Using Swing it is possible to create two different types of GUI programs
  – Standalone applications
    • Programs that are started from the command line
    • Code resides on the machine on which they are run
  – Applets
    • Programs run inside a web browser
    • Code is downloaded from a web server
    • JVM is contained inside the web browser
    • Applets are normally prevented from doing certain things
• For now we will write standalone applications
**JFrames**

- A **JFrame** is a Window with all of the adornments added.
- A **JFrame** provides the basic building block for screen-oriented applications.

```java
JFrame win = new JFrame( "title" );
```

**Creating a JFrame**

```java
import javax.swing.*;
public class SwingFrame {
    public static void main( String args[] ) {
        JFrame win = new JFrame( "My First GUI Program" );
        win.setVisible( true );
    }
} // SwingFrame
```

**Sizing a Frame**

- You can specify the size
  - Height and width given in pixels
  - The size of a pixel will vary based on the resolution of the device on which the frame is rendered
- The method, `pack()`, will set the size of the frame automatically based on the size of the components contained in the content pane
  - Note that `pack()` does not look at the title bar
Creating a JFrame

```java
import javax.swing.*;

public class SwingFrame {
    public static void main(String args[]) {
        JFrame win = new JFrame("My First GUI Program");
        win.setSize(250, 150);
        win.setVisible(true);
    }
}
```

JFrame

- JFrames have several panes:
  - Glass pane
  - Layered pane
  - Menu bar
  - Content pane

Components are placed in the content pane

Swing Components

- JComponent
  - JComboBox, JLabel, JList, JMenuBar, JPanel, JPopupMenu, JScrollPane, JTable, JTree, JFrame, JInternalFrame, JOptionPane, JProgressBar, JRootPane, JSeparator, JSlider, JSplitPane, JTabbedPane, JToolBar, JToolTip, JViewport, JColorChooser, JTextComponent, ...

**JLabels**

- **JLabels** are components that you can put text into.
- When creating a label you can specify the initial value and the alignment you wish to use within the label.
- You can use `getText()` and `setText()` to get and change the value of the label.

```java
JLabel label = new JLabel( "text", JLabel.RIGHT );
```

**Hello World**

```java
import javax.swing.*;

public class SwingFrame {
    public static void main( String args[] ) {
        JFrame win = new JFrame( "My First GUI Program" );
        JLabel label = new JLabel( "Hello World" );
        win.getContentPane().add( label );
        win.pack();
        win.setVisible( true );
    }
} // SwingFrame
```

**JButtons**

- **JButton** extends Component, displays a string and delivers an `ActionEvent` for each mouse click.
- Normally buttons are displayed with a border.
- In addition to text, JButtons can also display icons.

```java
JButton button = new JButton( "text" );
```
Buttons

```java
import javax.swing.*;

public class SwingFrame {
    public static void main( String args[] ) {
        JFrame win = new JFrame( "My First GUI Program" );
        JButton button = new JButton( "Click Me!!" );
        win.getContentPane().add( button );
        win.pack();
        win.setVisible( true );
    }
} // SwingFrame
```

Layout Manager

- Layout Manager
  - An interface that defines methods for positioning and sizing objects within a container. Java defines several default implementations of `LayoutManager`.
- Geometrical placement in a Container is controlled by a `LayoutManager` object

Components, Containers, and Layout Managers

- Containers may contain components (which means containers can contain containers!!).
- All containers come equipped with a layout manager which positions and shapes (lays out) the container’s components.
- Much of the action in the AWT occurs between components, containers, and their layout managers.
Layout Managers

- Layouts allow you to format components on the screen in a platform independent way.
- The standard JDK provides five classes that implement the `LayoutManager` interface:
  - `FlowLayout`
  - `GridLayout`
  - `BorderLayout`
  - `CardLayout`
  - `GridBagLayout`
- Layout managers are defined in the AWT package.

Changing the Layout

- To change the layout used in a container you first need to create the layout.
- Then the `setLayout()` method is invoked on the container is used to use the new layout.

```java
JPanel p = new JPanel();
p.setLayout(new FlowLayout());
```

- The layout manager should be established before any components are added to the container.

FlowLayout

- `FlowLayout` is the default layout for the `JPanel` class.
- When you add components to the screen, they flow left to right (centered) based on the order added and the width of the screen.
- Very similar to word wrap and full justification on a word processor.
- If the screen is resized, the components' flow will change based on the new width and height.
FlowLayout

```java
import javax.swing.*;
import java.awt.*;

public class FlowLayout {
    private static String labels[] = {
        "one", "two", "three", "four", "five", "six",
        "seven", "eight", "nine", "ten", "eleven", "twelve" };

    public static void main( String args[] ) {
        JFrame win = new JFrame( "My First GUI Program" );
        win.getContentPane().setLayout( new FlowLayout() );
        for ( int i = 0; i < labels.length; i++ )
            win.getContentPane().add( new JButton( labels[ i ] ) );
        win.pack();
        win.setVisible( true );
    }
}
```

GridLayout

- Arranges components in rows and columns
  - If the number of rows is specified
    - columns = number of components / rows
  - If the number of columns is specified
    - Rows = number of components / columns
  - The number of columns is ignored unless the number of rows is zero.
- The order in which you add components matters
  - Component 1 \( \rightarrow (0,0) \), Component 2 \( \rightarrow (0,1) \), ...
- Components are resized to fit the row-column area
BorderLayout

- BorderLayout provides 5 areas to hold components. These are named after the four different borders of the screen, North, South, East, West, and Center.
- When a Component is added to the layout, you must specify which area to place it in. The order in which components is not important.
- The center area will always be resized to be as large as possible.

```java
import javax.swing.*;
import java.awt.*;

public class SwingFrame {
    public static void main( String args[] ) {
        JFrame win = new JFrame( "My First GUI Program" );
        Container content = win.getContentPane();
        content.setLayout( new BorderLayout());
        content.add( BorderLayout.NORTH, new JButton( "North" ) );
        content.add( BorderLayout.SOUTH, new JButton( "South" ) );
        content.add( BorderLayout.EAST, new JButton( "East" ) );
        content.add( BorderLayout.WEST, new JButton( "West" ) );
        content.add( BorderLayout.CENTER, new JButton( "Center" ) );
        win.pack();  win.setVisible( true );
    }
} // SwingFrame
```
Containers

- A JFrame is not the only type of container that you can use in Swing
- The subclasses of Container are:
  - JPanel
  - JWindow
  - JApplet
- Window is subclassed as follows:
  - JDialog
  - JFrame

A Simple 4 Function Calculator
CalcGui.java

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class CalcGui implements ActionListener {
    // Labels for the buttons
    private static final String labels = "789X456/123-0C=+";
    private static final int NUMROWS = 4;
    private static final int NUMCOLS = 4;
    private JLabel display;  // The display
    public CalcGui( String name ) {
        // A Frame for the calculator
        JFrame win = new JFrame(name);
        // Create the button panel
        JPanel buttons = new JPanel();
        buttons.setLayout(new GridLayout(NUMROWS, NUMCOLS));
        JButton b;
        for ( int i = 0 ; i < labels.length() ; i++ ) {
            b = new JButton( labels.substring( i, i + 1 ) );
            buttons.add( b );
        }
        // Create the display
        display = new JLabel( "0", JLabel.RIGHT )
        // Add the panel to the frame
        win.add( buttons, BorderLayout.CENTER );
        win.add( display, BorderLayout.SOUTH );
        win.pack();
        win.setVisible(true);
    }
    // Handle clicks on the buttons
    public void actionPerformed(ActionEvent e) {
        String s = display.getText();
        if ( s.length() < 10 ) {
            s = s + e.getActionCommand();
            display.setText( s );
        }
    }
    public static void main(String[] args) {
        new CalcGui("Calculator");
    }
}
CalcGui.java

// "Assemble" the calculator
Container content = win.getContentPane();
content.setLayout( new BorderLayout() );
content.add( BorderLayout.NORTH, display );
content.add( BorderLayout.CENTER, buttons );
// Display it and let the user run with it :-) 
win.pack();
win.setVisible( true );
}