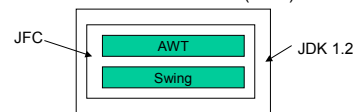


The Abstract Windowing Toolkit

- 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

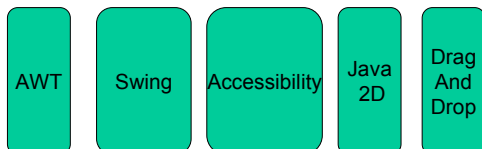
Swing

- 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"
 - it is based on the Model-View-Controller (MVC)



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.



Components

- A GUI consists of different graphic Component objects that 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 makes 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.

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.

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

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
 - For security purposes Applets are normally prevented from doing certain things (for example opening files).
- 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.

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

JFrame

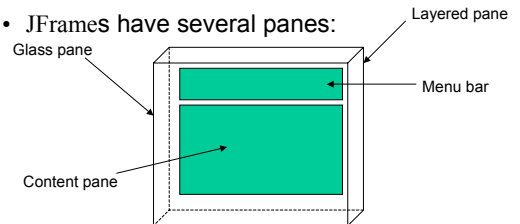
- Sizing a Frame
 - You can specify the size.
 - Height and width given in pixels.
 - The size of a pixel will vary based on the resolution of the device on which the frame is rendered.
 - The method, `pack()`, will set the size of the frame automatically based on the size of the components contained in the content pane
 - **Note** that `pack` does not look at the title bar.
- `SwingFrame2.java`

Creating a JFrame

- `SwingFrame.java`

JFrame

- JFrames have several panes:



- Components are placed in the content pane

Swing Components

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

Hello World Example

- SwingFrame3.java – Using a label.

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 retrieve and modify the value of the label.

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

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.

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

- SwingButton.java

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

Layout Managers

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

Components, Containers, and Layout Managers

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

Changing the Layout

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

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

- The layout manager should be specified 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 are added from 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' layout will change based on the new width and height.

GridLayout

- The GridLayout manager arranges components in rows and columns.
 - If the number of rows is specified
 - $\text{columns} = \text{number of components} / \text{rows}$
 - If the number of columns is specified
 - $\text{rows} = \text{number of components} / \text{columns}$
 - The number of columns is ignored unless the number of rows is zero.

Flow Layout

- SwingFlowLayout.java – Resize Window.

GridLayout

- The order in which components are added to the layout manager matters.
 - Component 1 \rightarrow (0,0), Component 2 \rightarrow (0,1),
- Components are resized to fit the row-column area.
- SwingGridLayout.java – Resize.

BorderLayout

- BorderLayout provides 5 areas to hold components.
 - The areas are named after the four different borders of the screen: North, South, East, West, and Center.
- When a Component is added to the layout, the area to place it in must be specified.
 - The order in which components is not important.

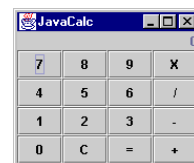
Containers

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

BoarderLayout

- The center area will always be resized to be as large as possible.
- SwingBoarder.java – Resize.

A Simple 4 Function Calculator



Swing Components

