

The Abstract Windowing Toolkit

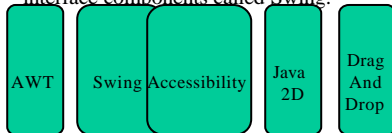
- Since Java was first released, its user interface facilities have been a significant weakness
 - The Abstract Windowing Toolkit (AWT) was part of the JDK from the beginning, but it really was not sufficient to support a complex user interface
- JDK 1.1 fixed a number of problems, and most notably,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

ICSS235 - Swing

1

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.

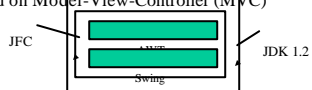


ICSS235 - Swing

2

Swing

- Swing classes are used to build GUIs
 - Swing does not stand for anything
 - Swing is built on top of the 1.1 and 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
 - based on Model-View-Controller (MVC)



ICSS235 - Swing

3

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
 - ...

ICSS235 - Swing

4

Components

- A graphical user interface 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. Classes derived from Container, most notably Panel, Applet, Window, Dialog, Frame, can contain multiple components.

ICSS235 - Swing

5

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.

ICSS235 - Swing

6

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

ICSS235 - Swing

7

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.

ICSS235 - Swing

8

Applets versus Applications

- It is possible to create two different types of GUIs
 - 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

ICSS235 - Swing

9

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" );
```

ICSS235 - Swing

10

Creating a JFrame

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



ICSS235 - Swing

11

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

ICSS235 - Swing

12

Creating a JFrame

```
import javax.swing.*;

public class SwingFrame {
    public static void main( String args[] ) {
        JFrame win = new JFrame( "My First GUI Program" );

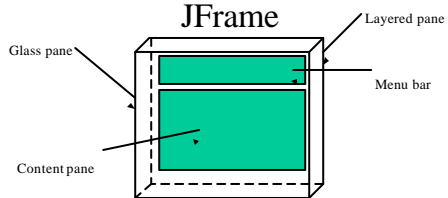
        win.setSize( 250, 150 );
        win.show();
    }
} // SwingFrame
```



ICSS235 - Swing

13

JFrame



- The content pane is where the components will be placed
- The entire collection of panes is called the RootPane

ICSS235 - Swing

14

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, ...

ICSS235 - Swing

15

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.

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

ICSS235 - Swing

16

Hello World

```
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.show();
    }
} // SwingFrame
```



ICSS235 - Swing

17

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.

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

ICSS235 - Swing

18

Buttons

```
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.show();
    }
} // SwingFrame
```



ICSS235 - Swing

19

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

ICSS235 - Swing

20

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.

ICSS235 - Swing

21

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

ICSS235 - Swing

22

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.

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

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

ICSS235 - Swing

23

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

ICSS235 - Swing

24

Flow Layout

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

public class SwingFrame {
    public static void main( String args[] ) {
        JFrame win = new JFrame( "My First GUI Program" );

        win.getContentPane().setLayout( new FlowLayout() );

        for ( int i = 0; i < 10; i++ )
            win.getContentPane().add(
                new JButton( String.valueOf( i ) ) );

        win.pack();
        win.show();
    }
} // SwingFrame
```

ICSS235 - Swing

25

FlowLayout



ICSS235 - Swing

26

GridLayout


- Arranges components in rows/columns.
 - If the number of rows is specified, the number of columns will be the number of components divided by the rows
 - If the number of columns is specified, the number of rows will be the number of components divided by the columns
 - Specifying the number of columns affects the layout only when the number of rows is set to zero.
 - The order in which you add components is relevant.

ICSS235 - Swing

27

GridLayout

```
GridLayout ( 2, 4 )
```



```
GridLayout ( 0, 4 )    GridLayout ( 4, 4 )    GridLayout ( 10, 10 )
```

ICSS235 - Swing 28

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

BorderLayout

```
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( "North", new JButton( "North" ) );
        content.add( "South", new JButton( "South" ) );
        content.add( "East", new JButton( "East" ) );
        content.add( "West", new JButton( "West" ) );
        content.add( "South", new JButton( "South" ) );
        content.add( "Center", new JButton( "Center" ) );

        win.pack(); win.show();
    } // SwingFrame
}
```

ICSS235 - Swing 30

BorderLayout



ICSS235 - Swing

31

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`

ICSS235 - Swing

32

A Simple 4 Function Calculator



ICSS235 - Swing

33

Swing Components



ICSS235 - Swing

34

CalcGui.java

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

public class CalcGui implements {
    // Labels for the buttons
    private static final String labels = "789X/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);
```

ICSS235 - Swing

35

CalcGui.java

```
// 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 )
```

ICSS235 - Swing

36

CalcGui.java

```
// "Assemble" the calculator
Container content = win.getContentPane();
content.setLayout( new BorderLayout() );
content.add( "North", display );
content.add( "Center", buttons );

// Display it and let the user run with it :-)
win.pack();
win.show();
}
```
