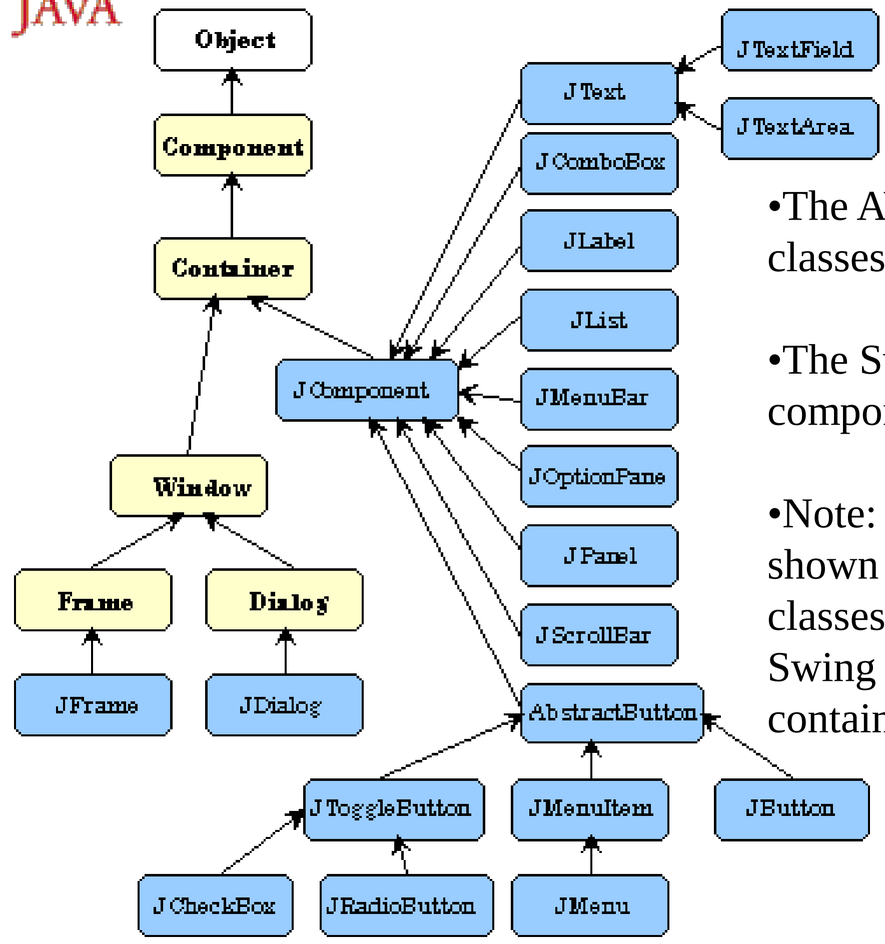
Swing Components

Ji Zhenyan & Åke Malmberg

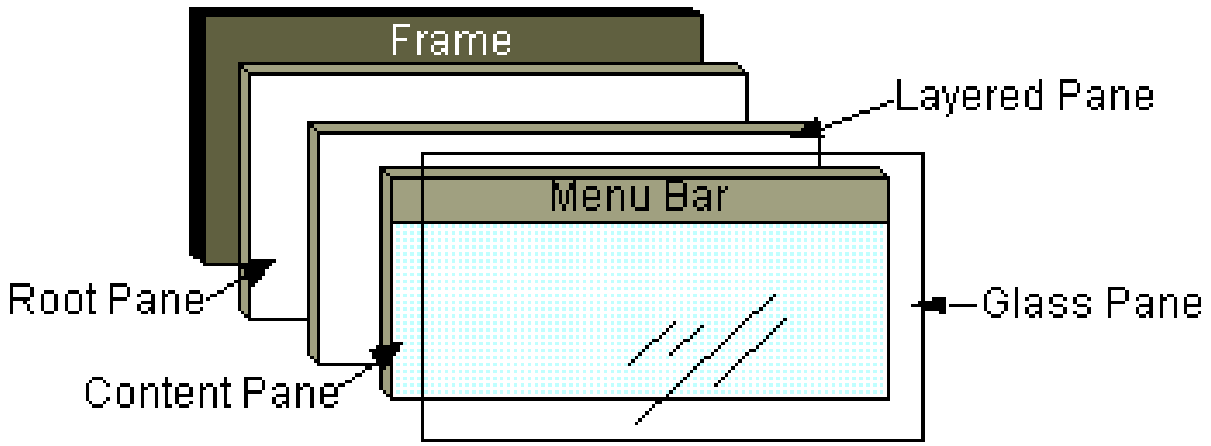


- The AWT classes are shown in yellow and the Swing classes in blue.

- The Swing components have similar names as the AWT components but begin with **J**.

- Note: There are several other Swing components not shown such as **JTable**. There are also numerous support classes that are not subclasses of **Component**.
Swing components, i.e. **JComponent** subclasses, are also containers and so can hold other Swing components.

GUI
Content Pane



- With the exception of top-level containers, all Swing components whose names begin with "J" descend from the JComponent class.
  - JPanel, JScrollPane, JButton, and JTable all inherit from JComponent.
  - JFrame doesn't because it implements a top-level container.

Using Top-Level Containers

- *top-level container* exists mainly to provide a place for other Swing components to paint themselves.
- Swing provides three generally useful top-level container classes: JFrame, JDialog, and JApplet.
- a fourth top-level container, JWindow, isn't generally useful. JWindow is the Swing version of the AWT Window class, which provides a window with no controls or title that is always on top of every other window.
- Each program that uses Swing components has at least one top-level container. (To appear onscreen, every GUI component must be part of a containment hierarchy.)

- As a rule, a standalone application with a Swing-based GUI has at least one containment hierarchy with a JFrame as its root.
  - if an application has one main window and two dialogs, then the application has three containment hierarchies, and thus three top-level containers. One containment hierarchy has a JFrame as its root, and each of the other two has a JDialog object as its root.
- Each top-level container has a content pane that, generally speaking, contains the visible components in that top-level container's GUI.
- You can optionally add a menu bar to a top-level container. The menu bar is positioned within the top-level container, but outside the content pane.

Adding Components to the Content Pane

- frame.getContentPane().add(yellowLabel, BorderLayout.CENTER);
- The default content pane is a simple intermediate container that inherits from JComponent, and that uses a BorderLayout as its layout manager.
- The getContentPane method returns a Container object, not a JComponent object.
- if you want to take advantage of the content pane's JComponent features, you need to either typecast the return value or create your own component to be the content pane.
- If you create your own content pane, make sure it's opaque. A JPanel object makes a good content pane because it's simple and it's opaque, by default.

- JPanel contentPane = new JPanel(); contentPane.setLayout(new BorderLayout()); contentPane.setBorder(*someBorder*); contentPane.add(*someComponent*, BorderLayout.CENTER); contentPane.add(*anotherComponent*, BorderLayout.SOUTH); *topLevelContainer*.setContentPane(contentPane);
- the default layout manager for JPanel is FlowLayout;

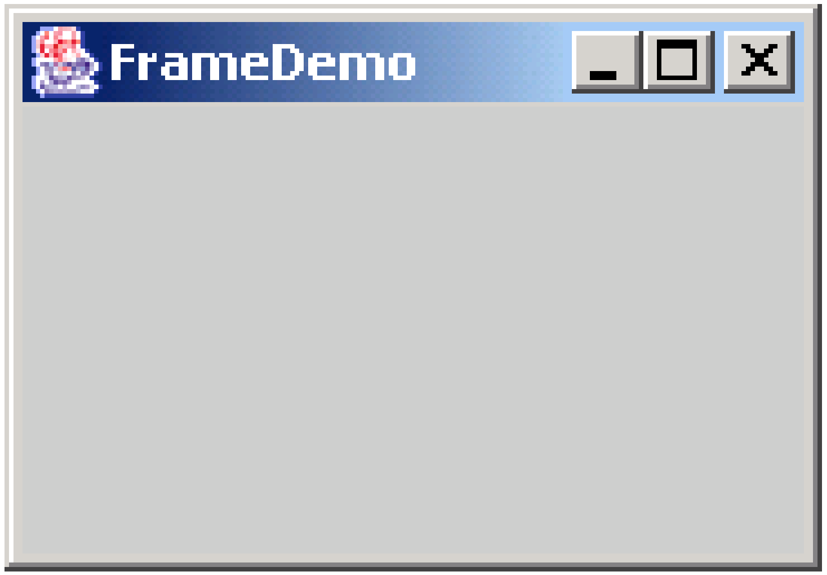
Frames

- To make a window that's dependent on another window -- disappearing when the other window is iconified, for example -- use a dialog instead of a frame.
- To make a window that appears within another window, use an internal frame.
- setDefaultCloseOperation (see interface WindowConstants)
  - DO_NOTHING_ON_CLOSE -- Don't do anything when the user's requests that the frame close. (use WindowListener)
  - HIDE_ON_CLOSE (the default) -- Hide the frame when the user closes it. This removes the frame from the screen.
  - DISPOSE_ON_CLOSE -- Hide and dispose of the frame when the user closes it. This removes the frame from the screen and frees up any resources used by it.
  - EXIT_ON_CLOSE -- Exit the application, using System.exit(0). This is recommended for applications only. If used within an applet, a SecurityException may be thrown.

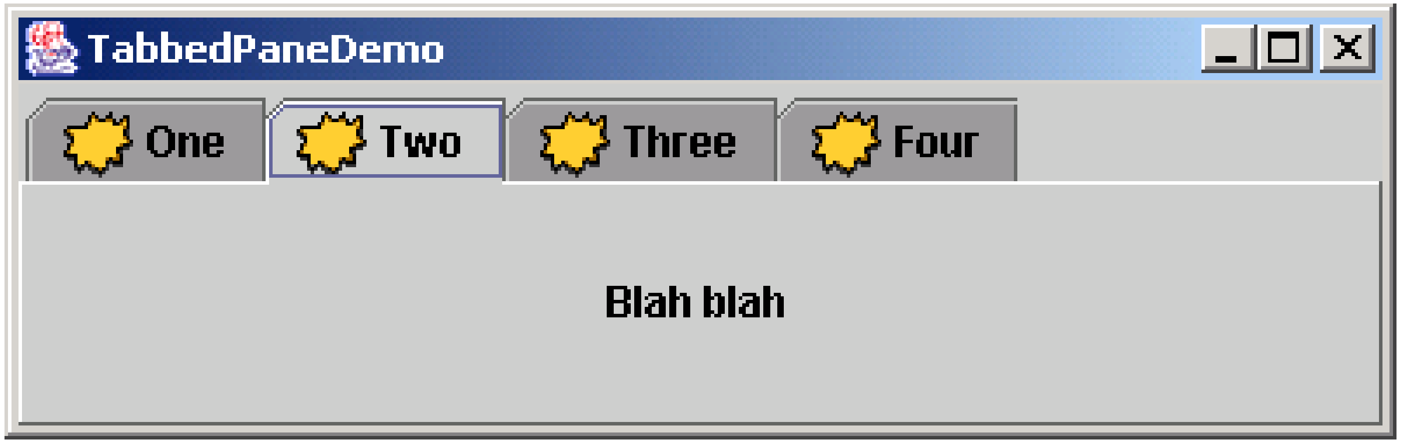
FrameDemoSimple.java

FrameDemoEmpty.java

FrameDemoBetter.java

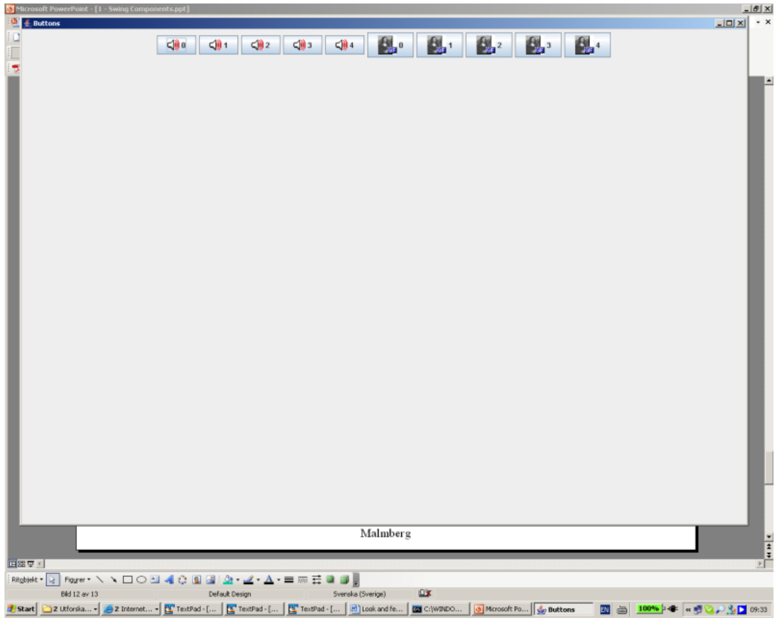


TabbedPaneDemo.java

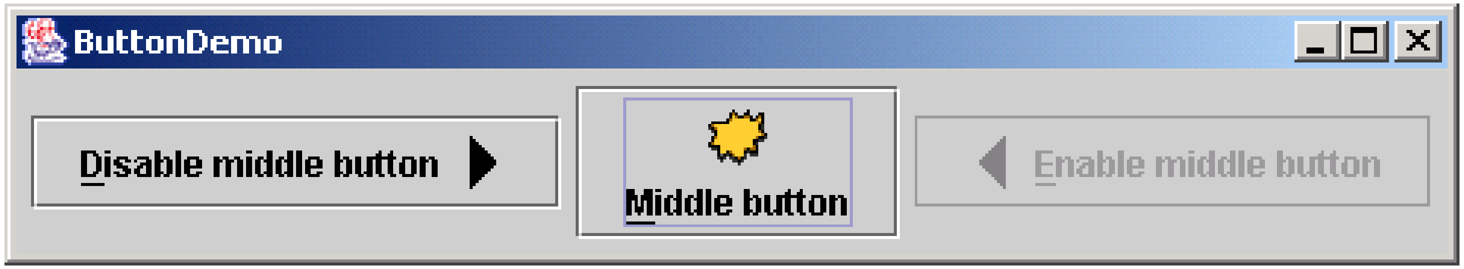


Blah blah

Buttons.java



ButtonDemo.java


ButtonDemo
Disable middle button
Middle button
Enable middle button