
Swing Components II

Chapter 11 - Lecture Slides

JTabbedPane, ToolTips, Borders,
Audio, JFrame, MediaTracker

JTabbedPane



JTabbedPane

- JTabbedPane allows a specific pane to be displayed based on the tab selected by the user
- Tabs can be placed on

```
JTabbedPane tabpane;  
tabpane = new JTabbedPane(JTabbedPane.LEFT);  
tabpane = new JTabbedPane(JTabbedPane.TOP);
```

- Default placements is on

```
JTabbedPane tabpane;  
tabpane = new JTabbedPane( );
```



JTabbedPane

Adding components to a pane in JTabbedPane

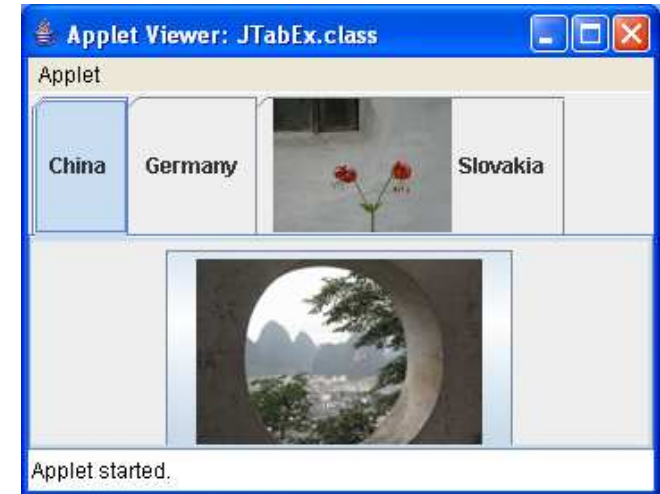
- Only `JComponent` component can be added to each tab pane
- Tab pane uses `BoxLayout`, so the component will stretch by default,
 - e.g., added using `addTab()`
- Adding tabs – use the `addTab()` method
 - `tabpane.addTab ("Tab 1", ImageIcon, panel, "tooltip text");`
 - `tabpane.addTab ("Second", object);`

JTabbedPane

Images on tab

- Need to specify while calling
- Needs to be an `ImageIcon` object

```
JTabbedPane tabpane;  
Image img;  
ImageIcon icon;  
tabpane = new JTabbedPane( );  
img = getImage( getCodeBase( ), "photo.gif" );  
Icon = new ImageIcon( img );  
tabpane.addTab( "Tab 1", icon, panel );
```



```

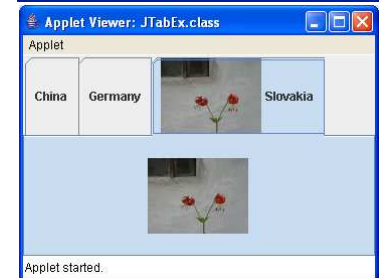
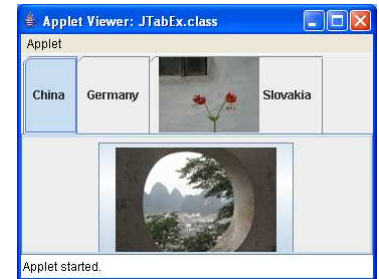
import java.awt.*;
import javax.swing.*;
public class JTabEx extends JApplet
{
    JButton b_China, b_Germany;
    JLabel flowers;
    JPanel panel;
    JTabbedPane tabpane;
    Image img1, img2, img3;
    ImageIcon iconWindow, iconStatue, iconFlowers;
    public void init( )
    {
        tabpane = new JTabbedPane( );
        img1 = getImage(getCodeBase( ), "Window.jpg" );
        img2 = getImage(getCodeBase( ), "Statue.jpg" );
        img3 = getImage(getCodeBase( ), "Flowers.jpg" );
        iconWindow = new ImageIcon(img1);
        iconStatue = new ImageIcon(img2);
        iconFlowers = new ImageIcon(img3);
        b_China = new JButton( "", iconWindow );
        b_Germany = new JButton( "What an idea!",
                                iconStatue );

        flowers = new JLabel( iconFlowers );
        panel = new JPanel( );
        panel.add(b_China);

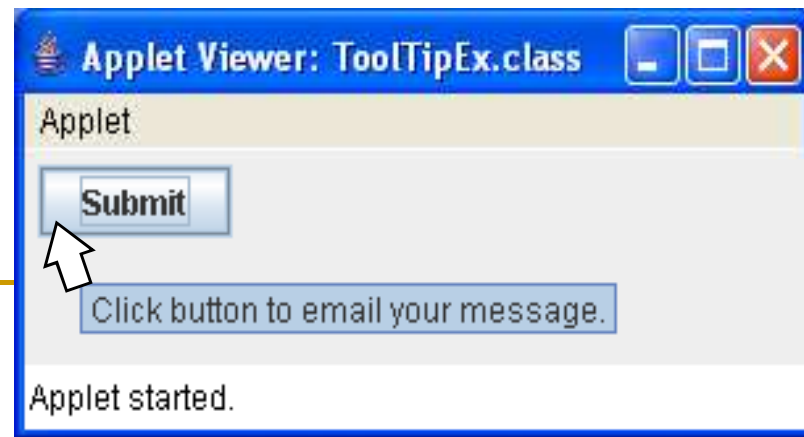
        // add tabs
        tabpane.addTab( "China", panel );
        tabpane.addTab( "Germany", b_Germany );
        tabpane.addTab( "Slovakia", iconFlowers, flowers, "Slovakia" );

        add( tabpane, BorderLayout.CENTER ); // add tabbed pane to applet
    }
}

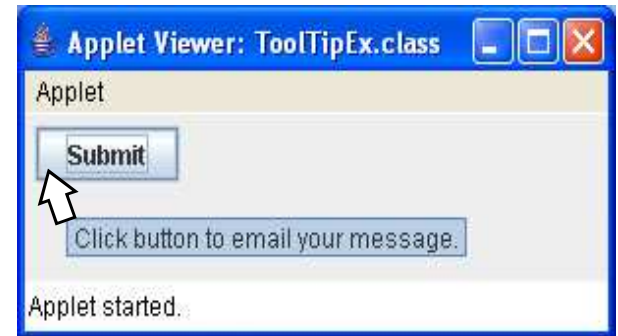
```



Tooltips



ToolTips



- Works on
 - JLabel, JTextField, JButton, etc.
- Can use `setToolTipText()` for the tooltip text
- Just add method call `obj.setToolTipText()` to object

```
obj.setToolTipText( text );
```

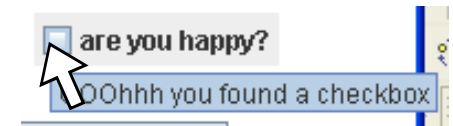
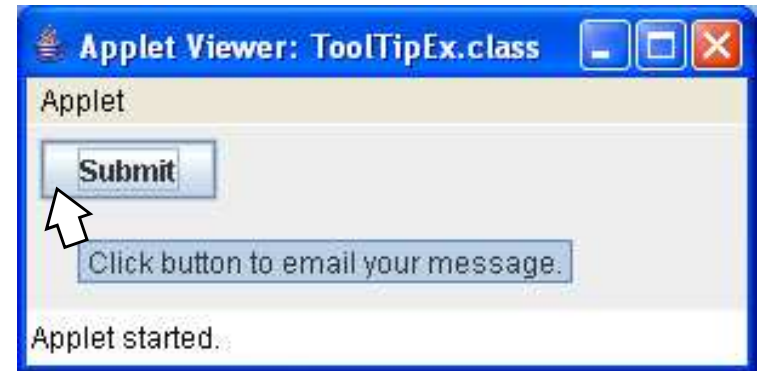
ToolTips

■ Example

```
JLabel label;  
JButton button;  
JCheckBox cb;
```

```
label = new JLabel( "My Example" );  
button = new JButton( "Submit" );  
cb = new JCheckBox( "are you happy?" );
```

```
label.setToolTipText( "<HTML>Just a <I>label</I>" );  
button.setToolTipText( "Click button to email your message" );  
cb.setToolTipText( "OOOhhh you found a checkbox" );
```



ToolTips on JTabbedPane

- Specify tooltip when adding tabs with **addTab**:

```
JTabbedPane tabs;  
tabs = new JTabbedPane( );  
tabs.addTab( "Title on Tab", null, jpanel, "Work Experience" );
```

Borders



Borders

- Not a component;
- Is a descriptor that can be added to components, to describe how to paint the edges
- Requires the use of the `JBorder` class from the `javax.swing` package
- Use the `setBorder` method on Swing components to apply a border to it
- Although usually applied to a panel (`JPanel`), you could also apply it to any Swing component, e.g, a `JLabel`

Borders

- Three things to do:



- create the

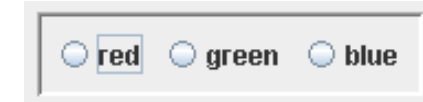
- call

- want a border around

on the object you

Borders - options

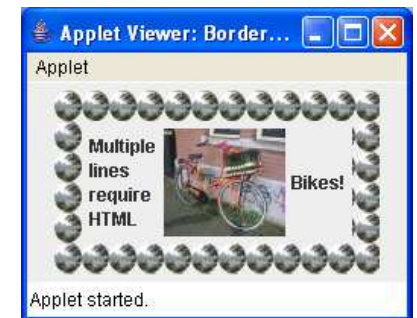
- **Line Border** simple line
- **Etched Border** etched groove
- **Bevel Border** raised (like a button) or lowered (sunk in)
- **Titled Border** a bevel with text on the border



- **Matte Border** specified size for the border, with a solid color or an image



- **Empty Border** adds a buffer of space around a component, but no “visual” border



Borders

// line border

```
Border lineborder = BorderFactory.createLineBorder( Color.RED );  
panel.setBorder(lineborder);
```

// etched border raised

```
Border etchBorder = BorderFactory.createEtchedBorder( EtchBorder.RAISED);  
panel.setBorder(etchedBorder);
```



// etched border lowered

```
Border etcBorder = BorderFactory.createEtchedBorder(EtchedBorder.LOWERED);  
panel.setBorder(etcBorder);
```

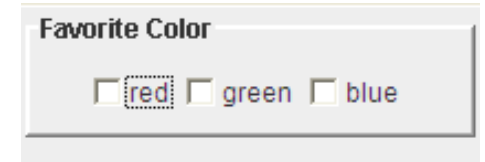


Borders

- lots of options

// titled border with raised bevel to panel

```
Border raisedbevel = BorderFactory.createRaisedBevelBorder();  
TitledBorder titled = BorderFactory.createTitledBorder( raisedbevel, "My Title");  
panel.setBorder(titled);
```



// add titled border with lowered bevel to panel

```
Border loweredbevel = BorderFactory.createLoweredBevelBorder();  
TitledBorder titled = BorderFactory.createTitledBorder( loweredbevel, "title");  
panel.setBorder(titled);
```



// matte border with an image

```
Border border = BorderFactory.createMatteBorder(-1, -1, -1, -1, ImageIcon);  
panel.setBorder( border );
```



Audio

Audio Files

- Java can play the following audio types:
 -
 -
 -
 -
 -

Audio

```
AudioClip clip;  
clip = getAudioClip( getCodeBase( ), audioFilename );  
clip.play( );
```

- There are three methods we can call on our `AudioClip`
 - `play()` play the sound file once through
 - `playLoop()` play the sound file continually
 - `stop()` stop playing the file

Audio

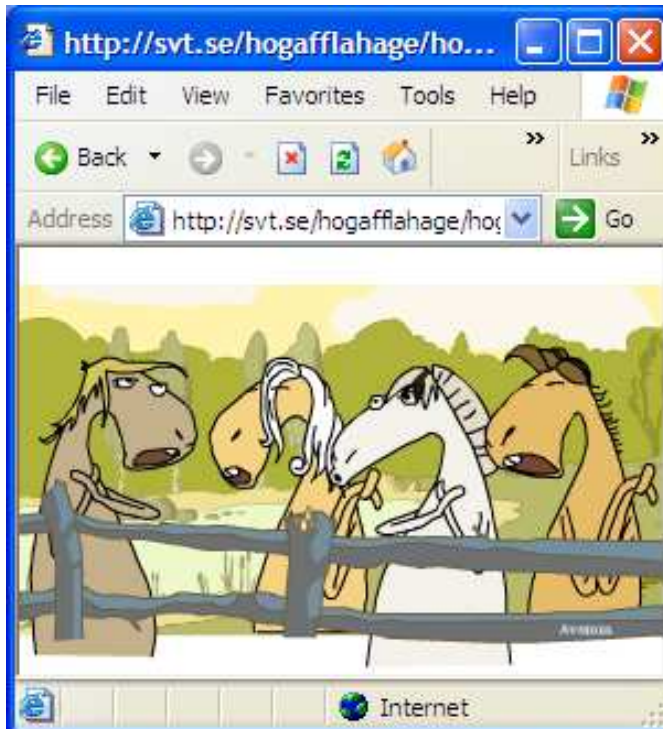
```
import java.awt.*;
import java.applet.*;
import javax.swing.*;
public class AudioPlay extends JApplet
{
    String audioFilename = "mySounds.mid";
    AudioClip ac;
    public void init( )
    {
        ac = getAudioClip( getCodeBase( ), audioFilename );
        ac.play( );
    }
}
```

Audio

```
import java.awt.*;
import java.applet.*;
import javax.swing.*;
public class AudioPlay extends JApplet
{
    String audioFilename = "mySounds.mid";
    AudioClip ac;
    public void init( )
    {
        ac = getAudioClip( getCodeBase( ), audioFilename );
        ac.loop( );
    }
}
```

Audio: Need for stop ()

- See Example
- Each button starts an audio file playing
- If the previous one isn't stopped first, then the new one plays on TOP of the other one
- This is usually not desired



However, some applications want this technique

(see example of Singing Horses on Internet at:

http://svt.se/hogafflahage/hogafflaHage_site/Kor/hestekor.swf

stop method

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
import javax.swing.*;
public class LoopAudio extends JApplet
{
    String audioFilename = "happyDaze.wav";
    AudioClip ac;
    public void init( )
    {
        ac = getAudioClip( getCodeBase( ), audioFilename );
        ac.loop( );      // loop instead of play
    }
    public void stop( )
    {
        // can't stop it if it isn't running, check first
        if( ac != null )
            ac.stop( );
    }
}
```

JFrame windows



Window

- Mostly used for pop-ups
- Generic type that **Frame** and **Dialog** inherit
- 4 important methods:
 - **dispose()** – eliminate resources when done
 - **setSize()** – resize to fit added components
 - **setVisible()** – make the window appear
 - **setLocation()** – position window on screen

JFrame

- Uses BorderLayout by default
- Possible ornaments:
 -
 -
 -
 -
 -
- Methods:
 - Constructors:
 - `new JFrame()`
 - `new JFrame(String title)`
 - `setBounds(w, h)`
 - `setSize(w, h)`



JFrame

- Create a JFrame window

```
JFrame frame;  
frame = new JFrame( text );
```

- Set the location on the screen (optional)

```
frame.setLocation( 500, 600 );
```

- **Major Steps**

1. Create the frame.

```
JFrame frame;  
frame = new JFrame("FrameDemo");
```

2. Set layout manager

```
frame.setLayout( new BorderLayout( ) );
```

3. Create components and put them in the frame.

```
JButton go = new JButton( "Go for it!" );  
frame.add( go, BorderLayout.SOUTH);
```

4. Size the frame.

```
frame.pack( );
```

OR

```
frame.setSize( 200, 500 );
```

5. Show it.

```
frame.setVisible(true);
```

JFrame – Simple Example

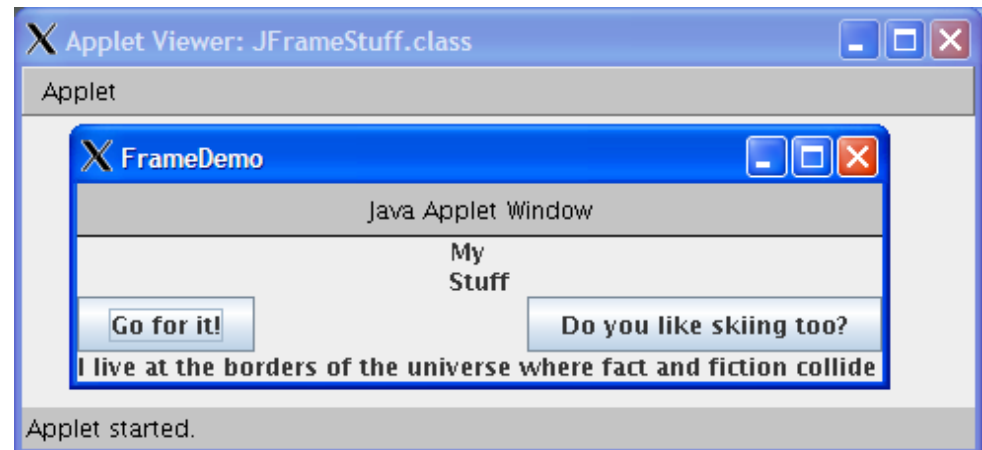
```
import javax.swing.*;
public class JFrameDefault extends JApplet
{
    JFrame frame;
    public void init( )
    {
        frame = new JFrame( "Testing JFrame stuff" );
        frame.setSize( 200,100 );
        frame.setVisible( true );
    }
}
```



JFrame – Example with components

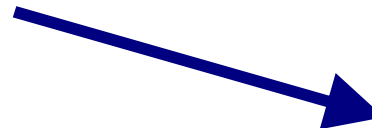
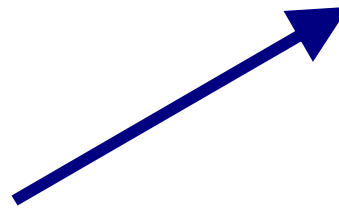
```
import java.awt.*;
import javax.swing.*;
public class JFrameStuff extends JApplet
{
    JFrame frame;
    JLabel myname, universe;
    JButton go, skiing;
    public void init( )
    {
        //1. Create the frame.
        frame = new JFrame("FrameDemo");
        //2. Set layout manager
        frame.setLayout( new BorderLayout( ) );
        //3. Create components and put them in the frame.
        myname = new JLabel( "<HTML>My <BR>Stuff", JLabel.CENTER );
        go = new JButton( "Go for it!" );
        skiing = new JButton( "Do you like skiing too?" );
        universe = new JLabel( "I live at the borders of the universe where fact and fiction collide" );
        frame.add(myname, BorderLayout.NORTH);
        frame.add(go, BorderLayout.WEST);
        frame.add(skiing, BorderLayout.EAST);
        frame.add(universe, BorderLayout.SOUTH);

        //4. Size the frame.
        frame.pack( );
        //5. Show it.
        frame.setVisible(true);
        //6. Optional: change location.
        frame.setLocation( 30, 70 );
    }
}
```



JFrame - Events

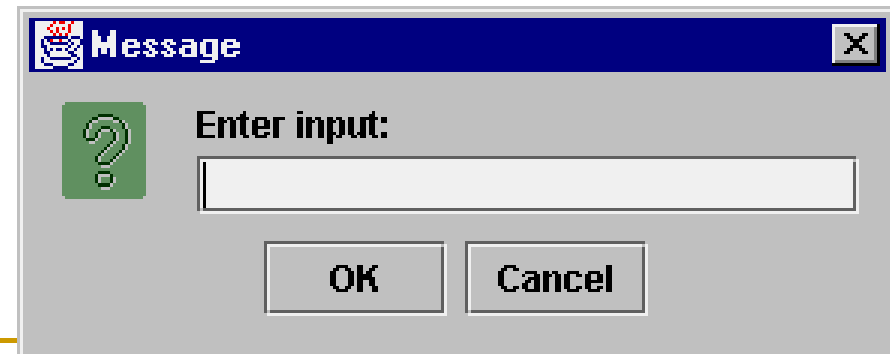
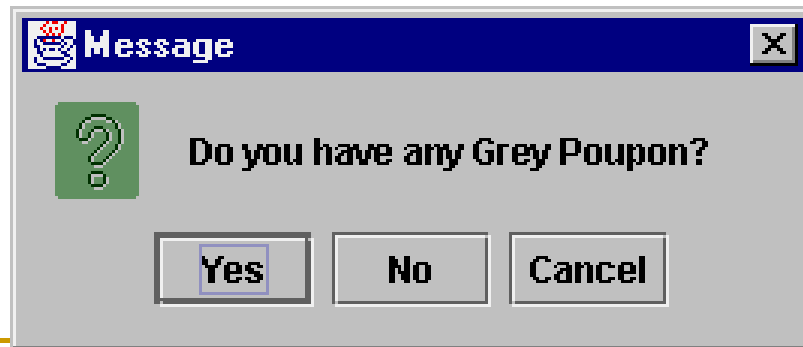
- usually want to pop open a `JFrame` based on an event (e.g. a button click)
- Example



Dialogs

(extra material)

JOptionPane Examples



Dialog

- For simple input/output with user
- Uses `JOptionPane` by default
- Are Modal by default - can change to modeless

- Constructors:
 - `new Dialog(Frame parent)`
 - `new Dialog(Frame parent, String title)`
 - `new Dialog(Frame parent, boolean isModal)`
- Call to `super()`

JFrame Closing Behaviors

- When user selects window manager Close option for **Dialog**, has default behavior
 - *Dialog* did nothing
 - *Dialog* hides itself
- **setDefaultCloseOperation (operation)**
 - DO_NOTHING_ON_CLOSE
 - HIDE_ON_CLOSE
 - DISPOSE_ON_CLOSE
 - No EXIT_ON_CLOSE operation

Dialog Example

```
JFrame frame;  
JButton button;  
Dialog dmsg;  
frame = new JFrame( );  
dmsg = new Dialog( frame, "Hi there", true );  
button = new JButton( "Okay?" );  
JPanel panel = new JPanel();  
p.add( button );  
dmsg.add( panel );  
dmsg.setSize( 400,100 );  
dmsg.setVisible( true );
```

Menu

(extra material)

Menu

- MenuBar
 - Collection of Menus
 - Associated with a frame
- Menu
 - Collection of MenuItem and separators
 - Can be a MenuItem for hierarchical menus
- MenuItem
 - Enabled or disabled
 - May be checkable

MediaTracker

(extra material)

MediaTracker

-
- Wait until all images loaded before displaying the applet
-

MediaTracker

- Declare a variable
- Create an instance
- Add an image to the tracker
- Wait until all the images load before continuing with program

```
public void start( )  
{  
    try  
    {  
        imgTracker.waitForAll( );  
    }catch (Exception e ) { }  
}
```

```

import java.awt.*;
import javax.swing.*;
public class MediaTrackerEx extends JApplet
{
    Image one, two, three, four;
    MediaTracker imgTracker;
    public void init( )
    {
        loadImages( );
    }
    public void start( )
    {
        try
        {
            imgTracker.waitForAll( );
        } catch (Exception e ) { }
    }
    public void loadImages( )
    {
        imgTracker = new MediaTracker( this );
        one = getImage( getCodeBase( ), "Amsterdam.jpg" );
        two = getImage( getCodeBase( ), "Botswana.jpg" );
        three = getImage( getCodeBase( ), "Statue.jpg" );
        four = getImage( getCodeBase( ), "Thailand.jpg" );
        imgTracker.addImage( one, 1 );
        imgTracker.addImage( two, 2 );
        imgTracker.addImage( three, 3 );
        imgTracker.addImage( four, 4 );
    }
}

```

Summary

- JTabbedPane
- Tooltips
- Borders
- Audio
- JFrame
- JOptionPaneS
- Extra Material*
 - Menu
- MediaTracker