Java Swing tutorials
Table of Contents. Good to find "How To demos".
http://docs.oracle.com/javase/tutorial/uiswing/TOC.html

Java Foundation Classes

**GUI Components: Swing & AWT**
**Pluggable Look & Feel**
**Java 2D API**
**Drag and Drop**
**Accessibility API**

---

**Pluggable Look & Feel**

<table>
<thead>
<tr>
<th>Platform</th>
<th>LookAndFeel Class</th>
<th>Default LookAndFeel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java</td>
<td>MetalLookAndFeel</td>
<td>any -- default</td>
</tr>
<tr>
<td>CDE/Motif</td>
<td>MotifLookAndFeel</td>
<td>any</td>
</tr>
<tr>
<td>MS Windows</td>
<td>WindowsLookAndFeel</td>
<td>Windows</td>
</tr>
<tr>
<td>Mac OS</td>
<td>MacLookAndFeel</td>
<td>Mac OS</td>
</tr>
</tbody>
</table>
Swing is not a replacement for AWT (Abstract Window Toolkit)
Swing classes extend AWT

<table>
<thead>
<tr>
<th>Swing Heavyweight components</th>
<th>Swing Lightweight components</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWT</td>
<td>Applet, Frame, Window, Dialog</td>
</tr>
<tr>
<td>Component, Container, Graphics, Color, Font, Toolkit, Layout Managers</td>
<td></td>
</tr>
</tbody>
</table>

Heavyweight components are opaque, windows, drawn by native window system, not as portable.

Lightweight components can have transparent backgrounds, drawn by JVM, portable.
Java UIs consist of containers using layout managers to control the geometry of UI components.

Java UI components can register listeners (or adapters) to react to events that are “fired off” by components. Components have `add*Listener()` methods for expected events. Listeners are Java interfaces.
Swing Intro

classes
containers, components, layout managers, events, adapters
JFrame, JLabel, BorderLayout, MouseAdapter

interfaces

Event listeners
ActionListener, MouseListener
interfaces are messy w/ 2+ methods.
every implementor must define all methods

adapters

convenience classes
JFC provides an adapter class for every listener
interfaces w/ 2+ methods.
Adapter extends an interface w/ methods w/ null behavior.
MouseAdapter, WindowAdapter
Java has single inheritance for classes but can implement multiple interfaces.

**Inner Classes**

- Inner class’s definition is nested within outer class
- Named or unnamed inner classes
- Scope rules allow reference of outer class variables

**Anonymous classes** – constructed and returned without a name, defined in the body, local and used only once.
The listener’s interface member functions are invoked when an event is received.

- ActionListener
- AdjustmentListener
- ContainerListener
- FocusListener
- ItemListener
- KeyListener
- MouseListener
- MouseMotionListener
- TextListener
- WindowListener

Events are represented by event classes.

- MouseEvent has getX(), getY(), getPoint(), getClickCount(), translatePoint(int x, int y), isPopupTrigger() interfaces.

Not all events have a specific class -- some event classes represent related events and use integer constants to identify the actual event.

- WindowEvent handles: activating, deactivating, closing, opening, iconifying, and de-iconifying windows with getWindow() interface
Event handling wiring "problem"

Designer of class wants to support an "application specific" behavior when a user initiates an event (request for behavior), but doesn't know what the event will be.

Provides a typed "callback slot" reference and set methods to "add" or "remove" an event handler object.

Application developer wants to use or extend existing GUI classes and be able to add "application specific" behaviors (methods) to be performed when user requests them via the GUI control.

Must provide correct type "callback object" when setting event handler.
<table>
<thead>
<tr>
<th>Event Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActionEvent</td>
<td>button activated, list item double clicked, menuItem selected, textField (enter typed)</td>
</tr>
<tr>
<td>AdjustmentEvent</td>
<td>scrollbar thumb moved</td>
</tr>
<tr>
<td>ItemEvent</td>
<td>checkbox is toggled, checkboxMenuItem selected, choice item selected, list item selected</td>
</tr>
<tr>
<td>TextEvent</td>
<td>textComponent text changes</td>
</tr>
<tr>
<td>ComponentEvent</td>
<td>component has moved (automatically handled)</td>
</tr>
<tr>
<td>ContainerEvent</td>
<td>component added / removed (auto handled)</td>
</tr>
<tr>
<td>FocusEvent</td>
<td>focus has been gained or lost</td>
</tr>
<tr>
<td>PaintEvent</td>
<td>need to repaint (automatically handled)</td>
</tr>
<tr>
<td>WindowEvent</td>
<td>window state has changed: minimized, ....</td>
</tr>
<tr>
<td>InputEvent</td>
<td></td>
</tr>
<tr>
<td>KeyEvent</td>
<td></td>
</tr>
<tr>
<td>MouseEvent</td>
<td></td>
</tr>
<tr>
<td>MouseWheelEvent</td>
<td></td>
</tr>
</tbody>
</table>
Top Level Containers

JApplet, JFrame, JDialog, JWindow

Top level containers are the root of a containment hierarchy. Interface between the native windowing environment and window manager and java application or applet.

JApplet, JFrame and JDialog contain a JRootPane which contains a contentPane container.

Components and layout managers must be added and set to the content pane not JApplet, JFrame or JDialog. (else exception thrown)
JFrame frame = new JFrame("frame title");

Constructor creates menubar and sets layout managers.

Need to get the content pane to add components to the frame

```java
frame.getContentPane().add(greenLabel, BorderLayout.CENTER);
```

Adding a menu bar to the frame

```java
frame.setJMenuBar(greyMenuBar);
```

Respond to window closing

```java
frame.addWindowListener(new WindowAdapter() {
    public void windowClosing(WindowEvent e) {System.exit(0);}

frame.pack(); // resolves all containment management
frame.setVisible(true); // displays the frame
Menus in menubars or as popup
Simple menu setup steps: (order can vary)
1. extend a JFrame for top container
2. instantiate a JMenuBar
3. instantiate a JMenu for every menu choice in the menu bar
4. instantiate each JMenuItem, JRadioButtonMenuItem, or JCheckboxMenuItem choice in the JMenu
   set JMenuItem icon
   set mnemonics and/or key accelerators
   add any submenus (pull right menus)
   add any item separators
3. add the JMenuItemItems to JMenu
4a. add a common menuItemClickListener for all (set of) menu items
4b. add an ActionListener each menu item.
5. add the JMenu to the JMenuBar
6. set the MenuBar in the frame
Swing has many pre-built dialogs

**JOptionPane** class has prebuilt, message, warning, information, question dialogs with default icons and “ok” button. Also customizable.

```java
JOptionPane.showMessageDialog( null, "Hello 585", "HI!", JOptionPane.INFORMATION_MESSAGE);
```

<< walkthrough Pipe.java >>
File chooser

`JFileChooser` class has options to display various file choosing dialogs.

```
JFileChooser.showOpenDialog(Component parent)
JFileChooser.showSaveDialog(Component parent)
JFileChooser.showDialog(Component parent,
    String title_approve)
```

After file chooser is closed (modal dialog)

- `getSelectedFile()` obtains file reference
- `setCurrentDirectory(File)`
- `setFileFilter(FileFilter)`
- `setFileSelectionMode(int)`

Accessory components

- preview, icons
Color chooser

`JColorChooser` class has options to display color choosing dialogs.

`JColorChooser.createDialog()` creates a color chooser

`JColorChooser.show(...args)` creates a chooser from `args`

`getColor()` returns selected color

Custom dialogs / choosers

585 student developed `DateChooser` class and demo