

# ADVANCED GUI'S IN JAVA

By: Spencer Alves, Adam Singer, and  
Brandon Cox

# Layouts

- ▣ CardLayout
- ▣ GridBagLayout
- ▣ BorderLayout
- ▣ GroupLayout
- ▣ OverlayLayout
- ▣ SpringLayout
- ▣ ViewportLayout

# CardLayout

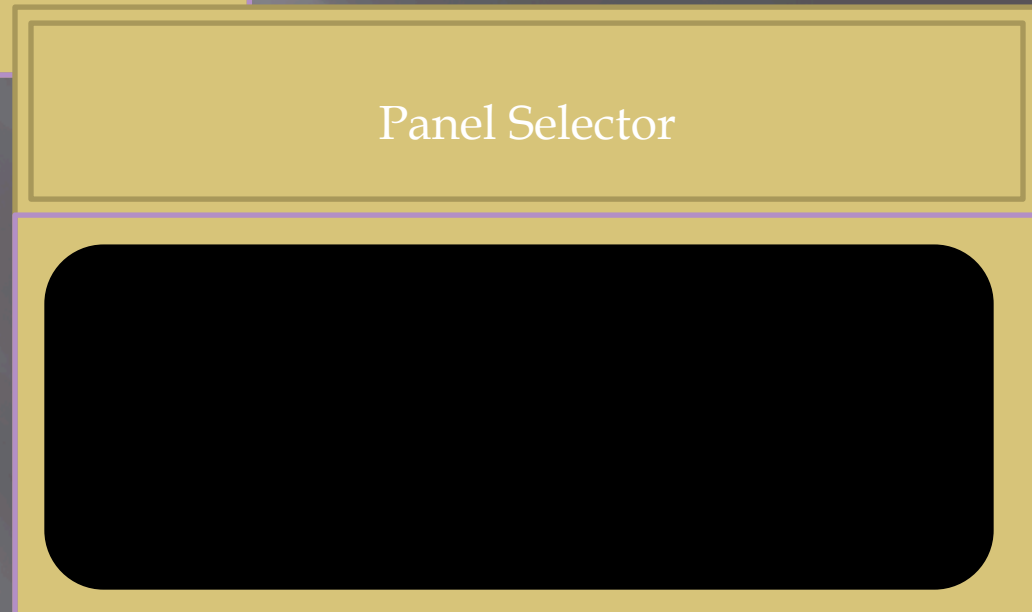
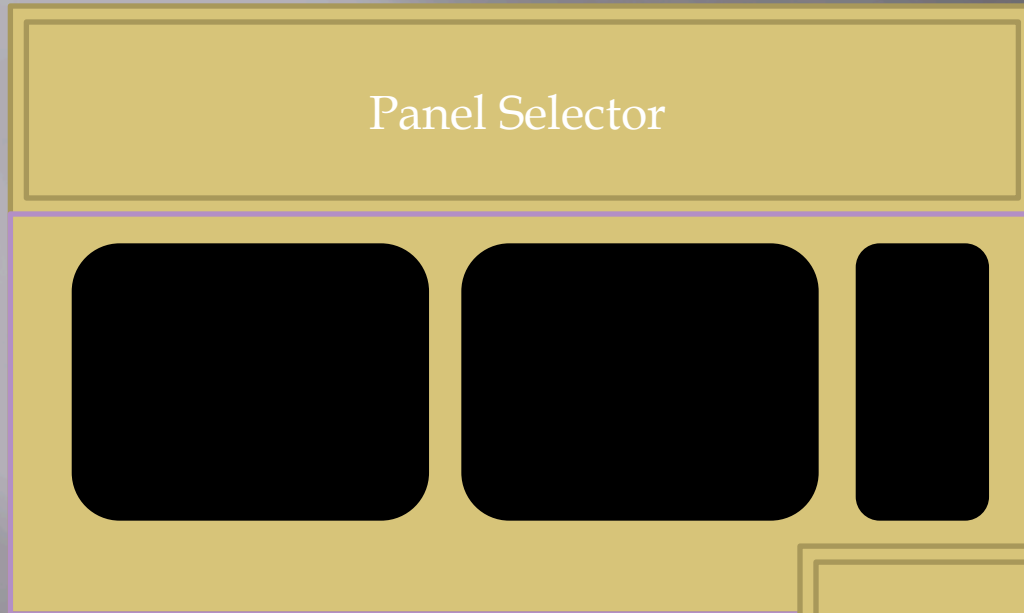
- ▣ Manages multiple components that share the same space. Similar to using a tabbed pane.
- ▣ Generally JPanels that contain multiple subcomponents.
- ▣ `Container.add(Component, Object)`
- ▣ Component is probably the panel, object is a unique identifier, probably a string.

# CardLayout

- ▣ When swapping between panels, actions must be performed on the CardLayout, which can be found with `Container.GetLayout()`. All functions also take the larger window as an argument.
- ▣ `cl.first(Component)`, `next(Component)`, `previous(Component)`, `last(Component)`
- ▣ `cl.show(Component, Object)`



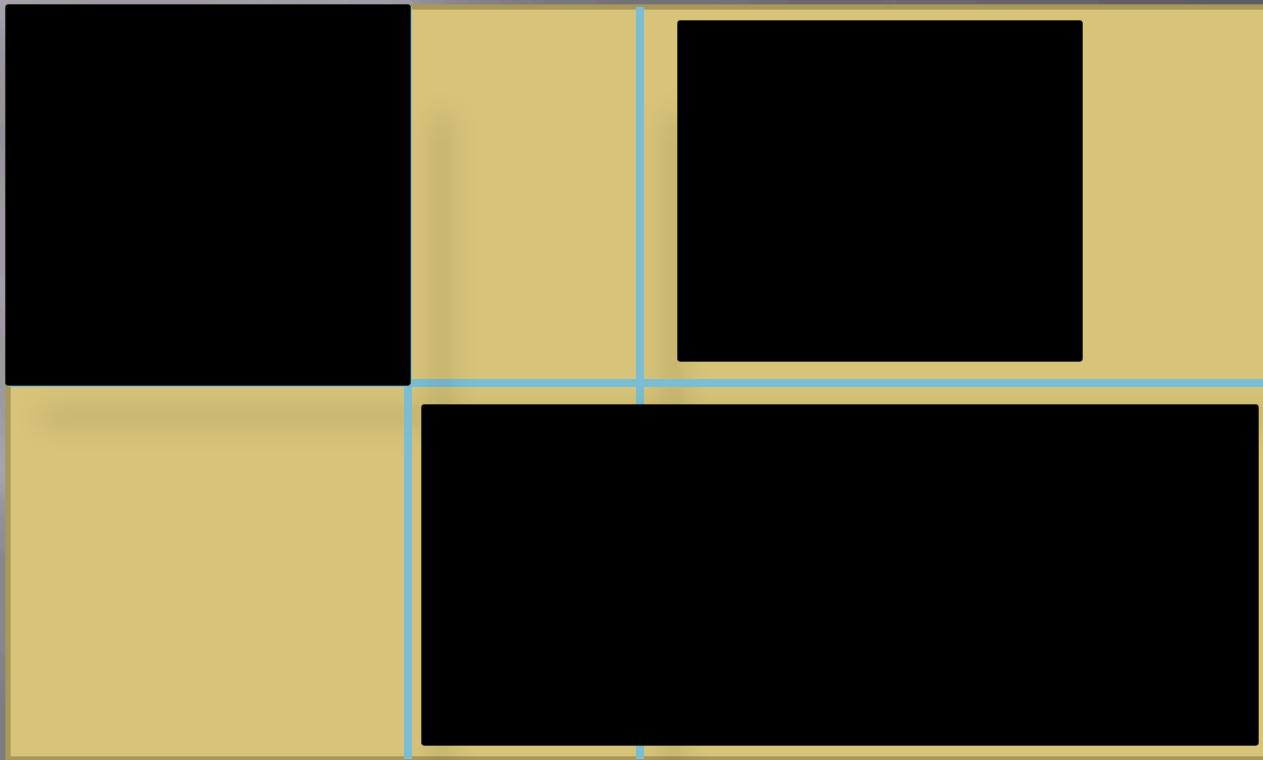
# CardLayout



# GridBagLayout

- ▣ Manages multiple components in a grid layout, but these components do not necessarily resize themselves strictly to the grid's constraints.
- ▣ `.add(Component, GridBagConstraints)`
- ▣ `GridBagConstraints` has a set of instance variables that can be set. This is what sets `GridBagLayout` apart from `GridLayout`.
- ▣ Components can set their own preferred sizes, and they can determine where their top left point is, and how many rows or columns they take up.

# GridBagLayout



# BoxLayout

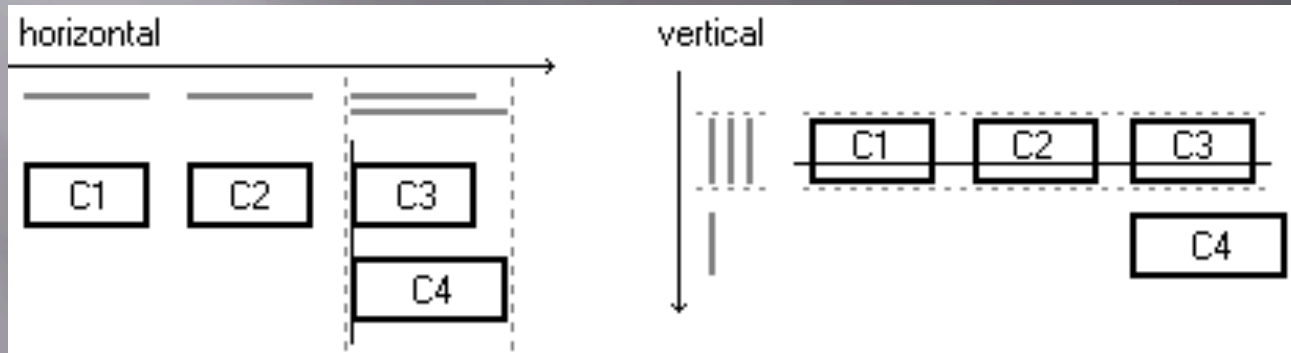
- ▣ Similar to FlowLayout.
- ▣ Allows determination of the direction in which components move across a page, as well as their alignments.
- ▣ Component. `setAlignmentX()`, `setAlignmentY()`
- ▣ Use Component.<Desired>\_ALIGNMENT constant.
- ▣ `BoxLayout.PAGE_AXIS` is top to bottom.
- ▣ `BoxLayout.LINE_AXIS` is left to right.

# GroupLayout

- ▣ Breaks creating a layout into horizontal components.
- ▣ Parallel Groups and Sequential Groups
- ▣ Parallel Groups in Sequential Groups!
- ▣ All components must be added twice.
- ▣ If a component is added in one dimension and not the other, an exception is thrown.
- ▣ Alignments can be set in each dimension separately.



# GroupLayout



- When done, set the vertical and horizontal groups using sequential and parallel groups.

# SpringLayout

- ❑ Can be visualized through a spread of components on the screen, connected by springs to adjacent components or the layout's edges.
- ❑ Vertical and Horizontal components are separate, just keep putting constraints.
- ❑ Creating many springs can be excessively long to code by hand, helper methods exist in SpringUtilities.
- ❑ Only resizable objects are resized when the window is resized.

# Data Input and Focus

- ▣ JList
- ▣ JTextArea
- ▣ JTextField
- ▣ JPasswordField

# FocusListeners

- ❑ Focus events are launched when components gain and lose focus.
- ❑ Implement the FocusListener interface in the component, and its focusGained and focusLost methods.
- ❑ Use addFocusListener.
- ❑ When a user presses enter in a JTextField or similar component, this uses an action listener, not a focus listener.

# JList

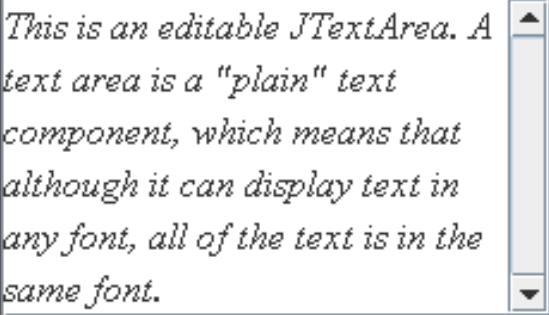
- ▣ Allows a user to select objects in a list.
- ▣ Create a ListModel, commonly a DefaultListModel
- ▣ Can insert or remove elements to or from this list model.
- ▣ `setSelectionMode`
- ▣ `setLayoutOrientation`





# JTextArea

- ▣ Allows a user to see and edit multiple lines of text.
- ▣ Only one type of font and color is allowed.
- ▣ Constructor can contain the initial text and the height and width in rows and columns of the contained text.
- ▣ Can modify various characteristics, such as whether lines are wrapped or whether the text is editable.
- ▣ Can also change the current selections in the window.



*This is an editable JTextArea. A text area is a "plain" text component, which means that although it can display text in any font, all of the text is in the same font.*

# JTextField

- ❑ Creates a small box that generally contains a single line of editable text of fixed length.
- ❑ Again, only one font type and color.
- ❑ Can set the maximum size of the field.
- ❑ Can check the text in the field on demand.
- ❑ Can fire an action event when the user indicates the text is complete, such as through pressing enter.

City:

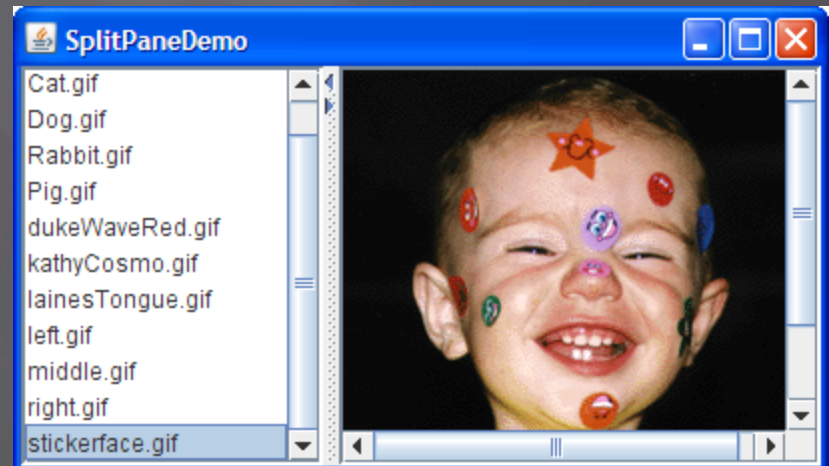
# JPasswordField

- ❑ Similar to a JTextField, but shows all characters as, by default, small black circles.
- ❑ Text is retrieved through the getPassword method rather than through getText. This is given as an array of characters rather than a string.
- ❑ The character that is shown instead of text can also be changed by default.

Enter the password:

# JSplitPane

- ❑ This type of pane displays two different components side by side that can be resized by dragging on a center divider.
- ❑ The type of split, vertical or horizontal, can be determined in the constructor.
- ❑ SplitPanels can be nested in order to add more than two components.





# JTabbedPane

- ▣ Similar to CardLayout.
- ▣ Multiple components share the same space, and the user determines which one to show by click on a set of tabs.
- ▣ After creating the pane, tabs can be added through the addTab method.





# Choices

- ▣ JCheckBox
- ▣ JRadioButton
- ▣ JComboBox
- ▣ JColorChooser
- ▣ JSpinner

# JCheckBox



- ❑ “An item that can be selected or deselected, and which displays its state to the user”
- ❑ Can have text or an icon next to it, just like other buttons
- ❑ Constructors: No arguments, a title, an icon, or both
- ❑ Check if it's selected with `isSelected` (inherited from `AbstractButton`) and set selection state with `setSelected`
- ❑ Change text with `setText`

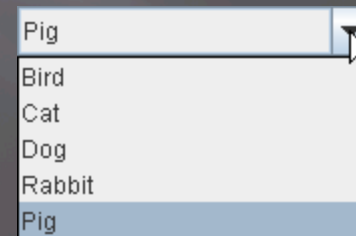
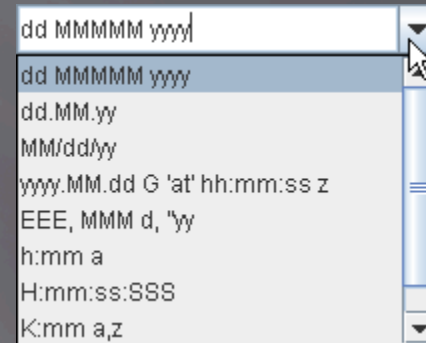
# JRadioButton

- This is a radio button
- All three are in a group
- This is deselected

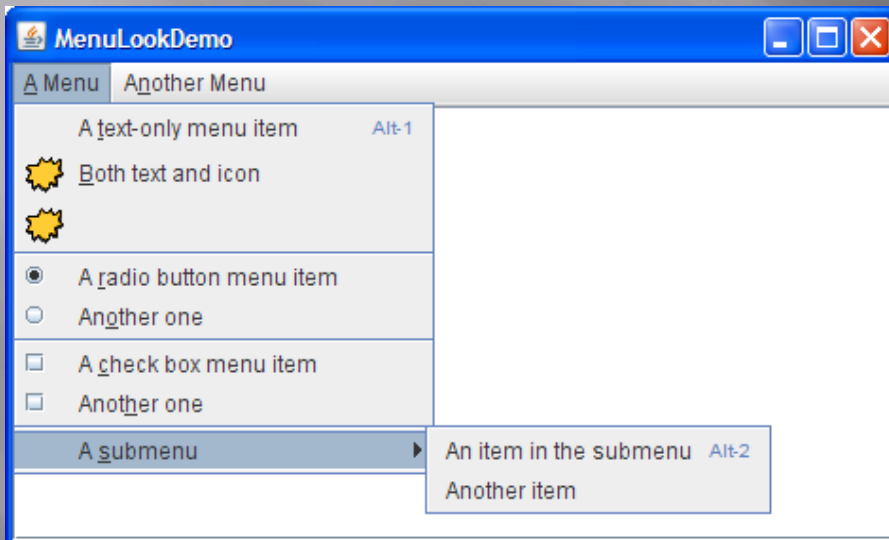
- ▣ Very similar to check box
- ▣ One is in a group, and only one in a group can be selected at a time
- ▣ Steps:
  - Create a few radio buttons
  - Configure each (with actions, mnemonics)
  - Create a ButtonGroup object
  - `group.add(each button)`
  - Add action listeners

# JComboBox

- Kind of like radio buttons, but save space
- Editable or non-editable
- Steps:
  - Create an array of strings
  - Create combo box
  - Set editable if needed
  - Configure box
  - Set selected item (by index)



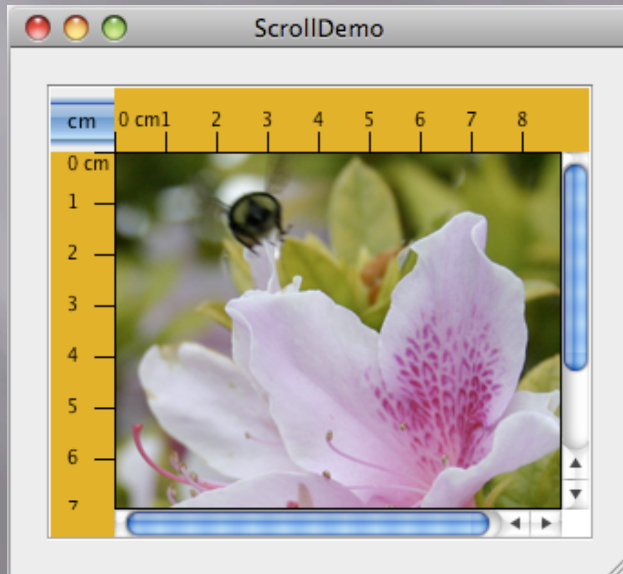
# JMenuBar, JMenu, and JMenuItem



- A JMenuBar is at the top of the screen, but a JPopupMenu is like a drop-down button.
- A JMenu is a single list of items, like the “file menu” or “edit menu,” or it can be a submenu
- Steps to create a menu hierarchy:
  - Create a top-level JMenuBar or JPopupMenu
  - Add JMenus to the Bar or Popup
  - Add JMenuItemS to the Jmenus
    - Could be just text, could have an icon, or you could use JCheckboxMenuItem or JRadioButtonMenuItem
    - Remember JRadioButtonMenuItems must be part of a ButtonGroup
  - Can also call addSeparator



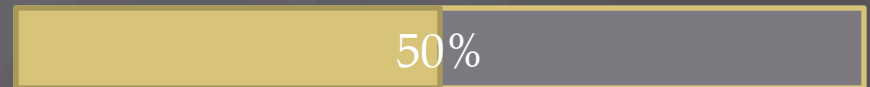
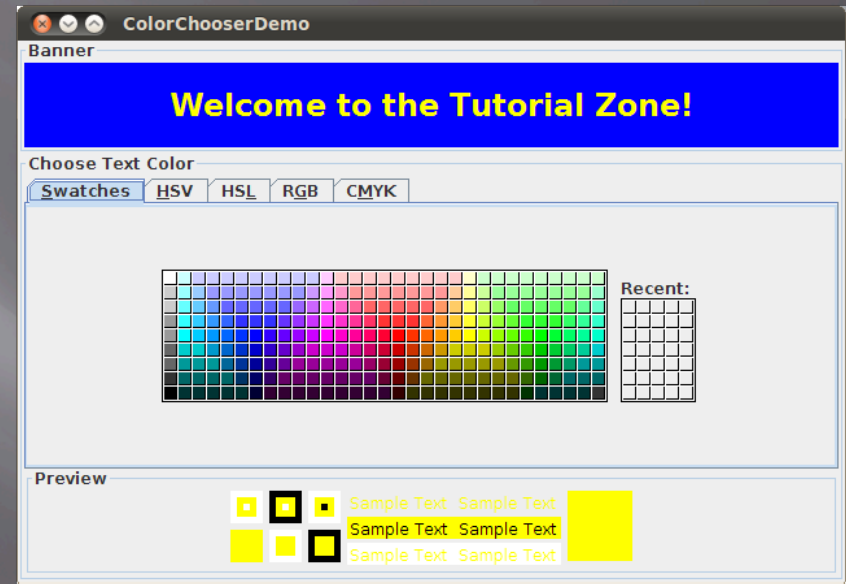
# JScrollBar and JScrollPane



- JScrollBar doesn't do much by itself
  - Can set min/max, position, orientation, increment
- JScrollPane
  - Uses JViewport to manage inside area
  - Can use ScrollablePicture or your own Component

# JColorChooser, JProgressBar, and JSpinner

- JColorChooser
  - Presents a modal dialog
  - getColor/setColor
  - Can also define dialog
- JProgressBar
  - Construct with start and end points
  - Update with setValue
  - Can also have a string on top (use setString, setStringPainted)
  - Can also be indeterminate (setIndeterminate)
- JSpinner
  - A text box with up/down arrows, used to select values
  - Can store a number, list, date, or any custom format



# Works Cited

- ▣ *The Java Tutorials*. Oracle, 2011. Web. 15 Oct. 2011.