CSE 331

Visual Index of Swing GUI Components

slides created by Marty Stepp

http://www.cs.washington.edu/331/
Swing inheritance hierarchy

- Component (AWT)
  - Window
    - Frame
      - JFrame (Swing)
      - JDialog
  - Container
    - JComponent (Swing)
      - JButton
      - JComboBox
      - JMenu
      - JPopupMenu
      - JScrollPane
      - JSplitPane
      - JToolBar
      - JTextField
      - JColorChooser
      - JFileChooser
      - JButton
      - JLabel
      - JOptionPane
      - JPanel
      - JMenuBar
      - JPopupMenu
      - JProgressBar
      - JScrollPane
      - JSlider
      - JSpinner
      - JScrollPane
      - JTabbedPane
      - JTable
      - JTextField
      - JFileChooser
      - JList
      - JMenu
      - JOptionPane
      - JPanel
      - JPopupMenu
      - JProgressBar
      - JScrollPane
      - JSlider
      - JSpinner
      - JScrollPane
      - JTabbedPane
      - JTable
      - JTextField

import java.awt.*; import javax.swing.*;
Component properties

- Each has a `get` (or `is`) accessor and a `set` modifier method.
- **examples**: `getColor`, `setFont`, `setEnabled`, `isVisible`.

<table>
<thead>
<tr>
<th>name</th>
<th>type</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>background</td>
<td>Color</td>
<td>background color behind component</td>
</tr>
<tr>
<td>border</td>
<td>Border</td>
<td>border line around component</td>
</tr>
<tr>
<td>enabled</td>
<td>boolean</td>
<td>whether it can be interacted with</td>
</tr>
<tr>
<td>focusable</td>
<td>boolean</td>
<td>whether key text can be typed on it</td>
</tr>
<tr>
<td>font</td>
<td>Font</td>
<td>font used for text in component</td>
</tr>
<tr>
<td>foreground</td>
<td>Color</td>
<td>foreground color of component</td>
</tr>
<tr>
<td>height, width</td>
<td>int</td>
<td>component's current size in pixels</td>
</tr>
<tr>
<td>visible</td>
<td>boolean</td>
<td>whether component can be seen</td>
</tr>
<tr>
<td>tooltip text</td>
<td>String</td>
<td>text shown when hovering mouse</td>
</tr>
<tr>
<td>size, minimum / maximum / preferred size</td>
<td>Dimension</td>
<td>various sizes, size limits, or desired sizes that the component may take</td>
</tr>
</tbody>
</table>
JFrame

*a graphical window to hold other components*

- **public JFrame()**
  public JFrame(String title)

  Creates a frame with an optional title.

  - Call `setVisible(true)` to make a frame appear on the screen after creating it.

- **public void add(Component comp)**

  Places the given component or container inside the frame.
More JFrame

- `public void setDefaultCloseOperation(int op)`
  Makes the frame perform the given action when it closes.
  - Common value passed: `JFrame.EXIT_ON_CLOSE`
  - If not set, the program will never exit even if the frame is closed.

- `public void setSize(int width, int height)`
  Gives the frame a fixed size in pixels.

- `public void pack()`
  Resizes the frame to fit the components inside it snugly.
JButton

*a clickable region for causing actions to occur*

- **public JButton(String text)**
  Creates a new button with the given string as its text.

- **public String getText()**
  Returns the text showing on the button.

- **public void setText(String text)**
  Sets button's text to be the given string.
JLabel

*a string of text displayed on screen in a graphical program. Labels often give information or describe other components*

- `public JLabel(String text)`
  Creates a new label with the given string as its text.

- `public String getText()`
  Returns the text showing on the label.

- `public void setText(String text)`
  Sets label's text to be the given string.
JTextField, JTextArea

an input control for typing text values
(field = single line; area = multi-line)

• public JTextField(int columns)
  public JTextArea(int lines, int columns)
  Creates a new field, the given number of letters wide.

• public String getText()
  Returns the text currently in the field.

• public void setText(String text)
  Sets field's text to be the given string.
JFormattedTextField

A text box that allows special formatting and can enforce constraints about allowable text

- public JFormattedTextField(Format format)
  Creates a new field that constrains itself to the given text format.
  (e.g. DateFormat, NumberFormat, CurrencyFormat, MaskFormat)

- public Object getValue()
  public void getValue(Object value)
  The value currently set in the field, which may lag behind the text.

- public void setFocusLostBehavior(int b)
  Sets what field should do if user stops editing and value is illegal.
JScrollPane

a container that adds scrollbars around any other component

• public JScrollPane(Component comp)
  Wraps the given component with scrollbars.
  ▪ After constructing the scroll pane, you must add the scroll pane, not the original component, to the onscreen container:

  ```java
  myContainer.add(new JScrollPane(textarea), BorderLayout.CENTER);
  ```
JOptionPane

- JOptionPane.showMessageDialog(parent, message);

import javax.swing.*; JOptionPane.showMessageDialog(null, "This candidate is a dog. Invalid vote.");

- Advantages:
  - Simple; looks better than console.

- Disadvantages:
  - Created with static methods; not object-oriented.
  - Not powerful (just simple dialog boxes).
More JOptionPane

• `JOptionPane.showConfirmDialog(parent, message)`
  - Displays a message and list of choices Yes, No, Cancel.
  - Returns an `int` such as `JOptionPane.YES_OPTION` or `NO_OPTION` to indicate what button was pressed.

• `JOptionPane.showInputDialog(parent, message)`
  - Displays a message and text field for input.
  - Returns the value typed as a String (or `null` if user presses Cancel).
**JPanel**

*the default container class in Swing*

- public JPanel()
  public JPanel(LayoutManager mgr)

  Constructs a panel with the given layout (default = flow layout).

- public void add(Component comp)
  public void add(Component comp, Object info)

  Adds a component to the container, possibly giving extra information about where to place it.

- public void remove(Component comp)

- public void setLayout(LayoutManager mgr)

  Uses the given layout manager to position components.
JCheckBox, JRadioButton

a toggleable yes/no value (checkbox)
or a way choose between options (radio)

- public JCheckBox(String text)
  public JCheckBox(String text, boolean checked)
  public JRadioButton(String text)
  Creates a checked/unchecked check box with given text.

- public boolean isSelected()
  Returns true if the check box is checked.

- public void setSelected(boolean selected)
  Sets box to be checked/unchecked.
ButtonGroup

*a logical collection to ensure that exactly one radio button from a group is checked at a time*

- public ButtonGroup()
- public void add(JRadioButton button)

- The ButtonGroup is not a graphical component, just a logical group; the RadioButtons themselves also need to be added to an onscreen container to be seen.
Icon

*a picture that can appear inside a component*

• public class ImageIcon implements Icon
  ▪ public ImageIcon(String filename)
  ▪ public ImageIcon(URL address)

• in JButton, JRadioButton, JCheckBox, JLabel, etc...
  ▪ constructor that takes an Icon
  ▪ public void setIcon(Icon)
  ▪ public void setSelectedIcon(Icon)
  ▪ public void setRolloverIcon(Icon)
JComboBox

*a drop-down list of selectable items*

- public JComboBox()
- public JComboBox(Vector items)
- public JComboBox(ComboBoxModel model)

Constructs a combo box. Can optionally pass a vector or model of items. *(See DefaultComboBoxModel for a model implementation.)*

- public void addActionListener(ActionListener al)

Causes an action event to be sent to listener al when the user selects or types a new item in the combo box.
JComboBox methods

- public void addItem(Object item)
- public Object getItemAt(int index)
- public void removeAllItems()
- public void removeItem(Object item)
- public void removeItemAt(int index)
- public int getSelectedIndex()
- public Object getSelectedSelectedItem()
- public void setSelectedItem(Object item)
- public void setSelectedSelectedIndex(int index)
- public void setEnabled(boolean enabled)
- public void setEditable(boolean editable)

If editable, the user can type new arbitrary values into the box.
JList

*a list of selectable pre-defined text items*

- public JList()
  Constructs an empty JList.

- public JList(ListModel model)
  public JList(Object[] data)
  public JList(Vector data)
  Constructs a JList that displays the given data.

- public void addListSelectionListener(
  ListSelectionListener lsl)
  Adds the given listener to be informed when the selected index changes for this list.
JFileChooser

a dialog box that allows the user to browse for a file to read/write

- public JFileChooser()
- public JFileChooser(String currentDir)
- public int showOpenDialog(Component parent)
- public int showSaveDialog(Component parent)
- public File getSelectedFile()
- public static int APPROVE_OPTION, CANCEL_OPTION

Possible result values from showXxxDialog(…)

JColorChooser

*a dialog box that allows the user to choose a color from a palette*

- public JColorChooser()
- public JColorChooser(Color initial)
- public Color showDialog(Component parent, String title, Color initialColor)
  - returns null if user chooses the Cancel button
JMenuBar

*a drop-down menu of commands*

- public JMenuBar()
- public void add(JMenu menu)

**Usage:** in `JFrame`, the following method exists:
- public void setJMenuBar(JMenuBar bar)
JTabbedPane

*a container that holds subcontainers, each with a "tab" label and content*

- `public JTabbedPane()`
- `public JTabbedPane(int tabAlignment)`

Constructs a new tabbed pane. Defaults to having the tabs on top; can be set to `JTabbedPane.BOTTOM, LEFT, RIGHT, etc.`

- `public void addTab(String title, Component comp)`
- `public void insertTab(...)`
- `public void remove(Component comp)`
- `public void remove(int index)`
- `public void removeAll()`
- `public void setSelectedComponent(Component c)`
- `public void setSelectedIndex(int index)`
JToolBar

*a movable dock container to hold common app buttons and commands*

- public JToolBar()
- public JToolBar(int orientation)
- public JToolBar(String title)
- public JToolBar(String title, int orientation)

Constructs a new tool bar, with optional title and orientation; can be `JToolBar.HORIZONTAL` or `VERTICAL`, default horizontal

- public void add(Component comp)

Adds the given component to this tool bar.

- Note: If using JToolBar, don't put other components in N/E/S/W.
JMenu

a sub-menu of commands with a JMenuBar

• `public JMenu(String text)`
• `public void add(JMenuItem item)`
• `public void addSeparator()`
• `public void setMnemonic(int key)`
JMenuItem

An entry within a JMenu that can be clicked to execute a command

- public JMenuItem(String text)
- public JMenuItem(String text, Icon icon)
- public JMenuItem(String text, int mnemonic)
- public void setAccelerator(KeyStroke ks)
- public void setEnabled(boolean b)
- public void setMnemonic(int mnemonic)
- public void addActionListener(ActionListener al)
J(CheckBox|RadioButton)MenuItem

a JMenuItem with a check box or radio circle

- public JMenuItem(String text)
- public JMenuItem(String text, boolean selected)
- public JMenuItem(String text, Icon icon)
- public JMenuItem(String text, Icon icon, boolean selected)
- public void addActionListener(ActionListener al)
- public boolean isSelected()
- public void setSelected(boolean b)

Recall: in a ButtonGroup, the following method exists:

- public void add(AbstractButton button)
- These two classes extend AbstractButton.
Mnemonics

- **mnemonic**: A context-sensitive menu hotkey assigned to a specific button or other graphical component.
  - Usually visible as an underlined key, activated by pressing Alt+key.
  - Only works when input focus is on the appropriate component.

- **usage**: call `setMnemonic(char)` method
  - Menu items also have a constructor that takes a mnemonic.

```java
myQuitButton.setMnemonic('Q');
JMenuItem myNewItem = new JMenuItem("New", 'N');
// or: myNewItem.setMnemonic('N');
```
Accelerators

- **accelerator**: A global hotkey that performs an action (ex: Alt-X to exit the program) even on components that aren't in focus / visible.
  - Can be run at any time in the application.
  - Can optionally include modifiers like Shift, Alt.
  - To create an accelerator:
    - Call the static `getKeyStroke` factory method of the `KeyStroke` class.
    - Pass its result to the `setAccelerator` method of the component.

```
menuItem.setAccelerator(
    KeyStroke.getKeyStroke('T', KeyEvent.ALT_MASK));
```
a dialog box is a sub-window connected to a given main window frame that pops up for a short time

- public JDialog(Frame parent, String title, boolean modal)
  Constructs a new dialog with the given parent and title. If modal is set, this dialog is a child of the parent and the parent will be locked until the dialog is closed.

- JDialog has most all JFrame methods: getContentPane(), setJMenuBar, setVisible, setTitle(String), ...