Swing

By: Dinesh Amatya

Design Patterns

- a written document that describes a general solution to a design problem that recurs repeatedly in many projects

The Model-View-Controller Pattern

- The model, stores the content
- The view, displays the content
- The controller, handles user input on view and translates that to changes in model

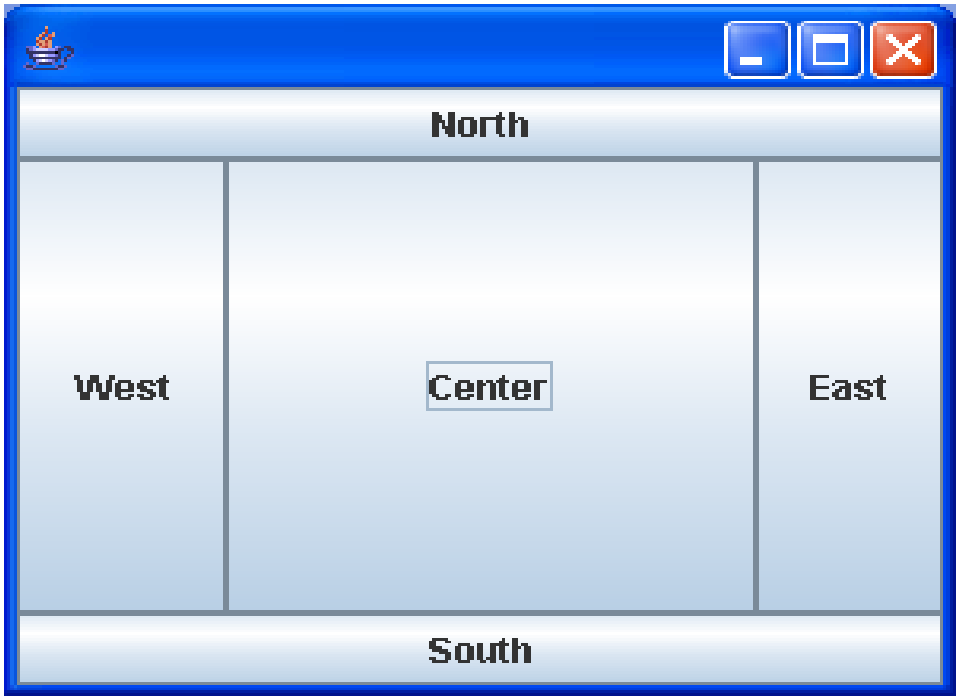
A Model-View-Controller Analysis of Swing Buttons

```
JButton button = new JButton("Blue");
ButtonModel model = button.getModel();

button.setUI(new BasicButtonUI());

button.addActionListener(new ButtonUIListener());
```

Layout Management

Border Layout

→ Default layout
→ Edge component laid out first
→ grows all component to fill available space
→ use panels to add components



Layout Management

Border Layout

```
JPanel panel = new JPanel();
panel.add(yellowButton);
frame.add(panel, BorderLayout.SOUTH);
```

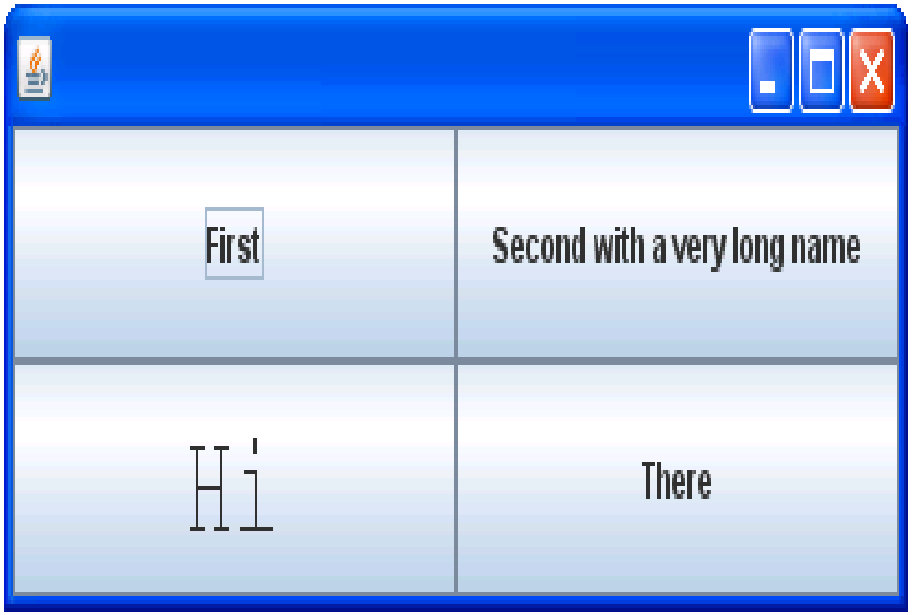
Layout Management

Grid Layout

→ arranges all components in rows and columns

→ all components are given same size

→ resizing the window grows or shrink all components with identical size



Layout Management

Grid Layout

```
panel.setLayout(new GridLayout(2, 2));
panel.add(new Jbutton("a"));
panel.add(new JButton("b"));
panel.add(new JButton("c"));
panel.add(new JTextField("d"));
```

Text Input

JtextField , JtextArea , JpasswordField

- String getText()
- void setText(String text)
- boolean isEditable()
- void setEditable(boolean b)

Jlabel

- Icon getIcon()
- void setIcon(Icon icon)

JScrollPane scrollPane = new JScrollPane(textArea);

Choice Components

JCheckBox
- boolean isSelected ()
- void setSelected(boolean state)

JRadioButton
- Icon getIcon()
- void setIcon(Icon icon)

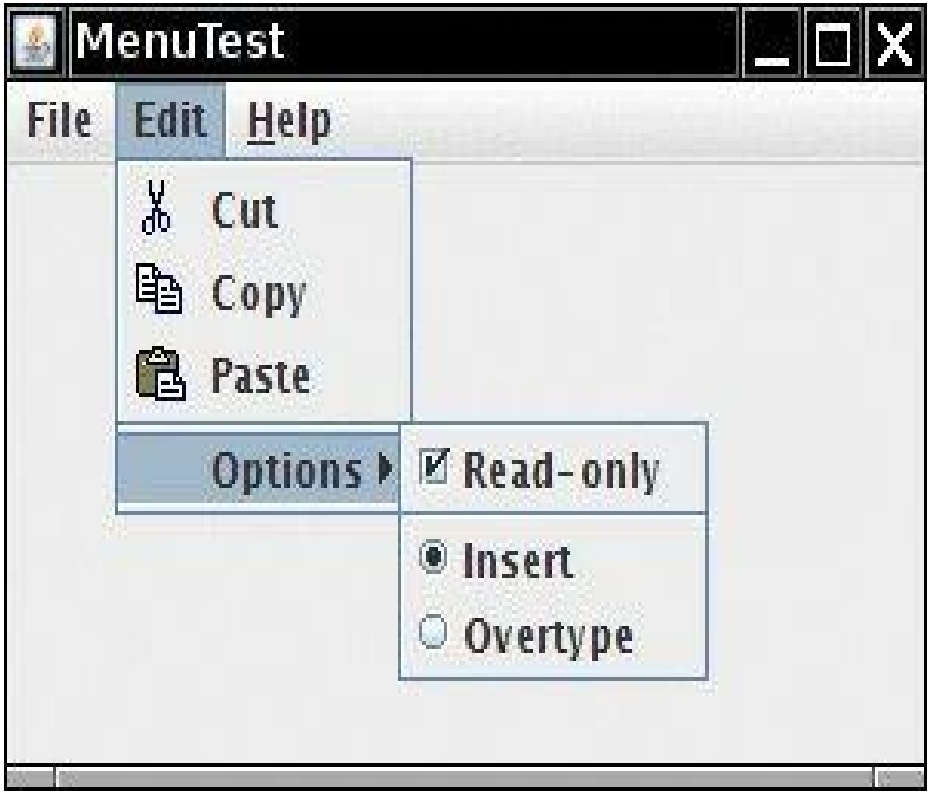
JcomboBox
- boolean isEditable()
- void setEditable(boolean b)
- void addItem(Object item)
- void insertItemAt(Object item, int index)
- void removeItem(Object item)
- void removeItemAt(int index)
- void removeAllItems()
- Object getSelectedItem()

JSlider slider = new JSlider(min, max, initialValue);

Menus

JMenuBar
JMenu
JMenuItem
JCheckBoxMenuItem
JradioButtonMenuItem
ImageIcon

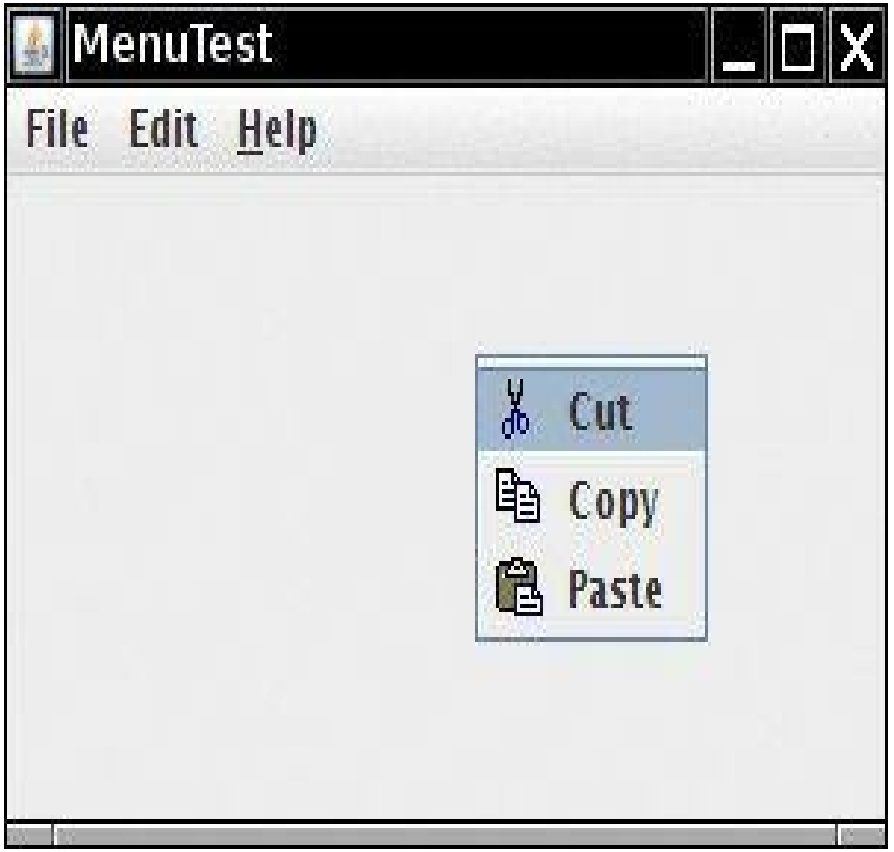
Jframe.setJMenuBar(menuBar)



Menus

JPopupMenu
JmenuItem

popup.show(panel, x, y);



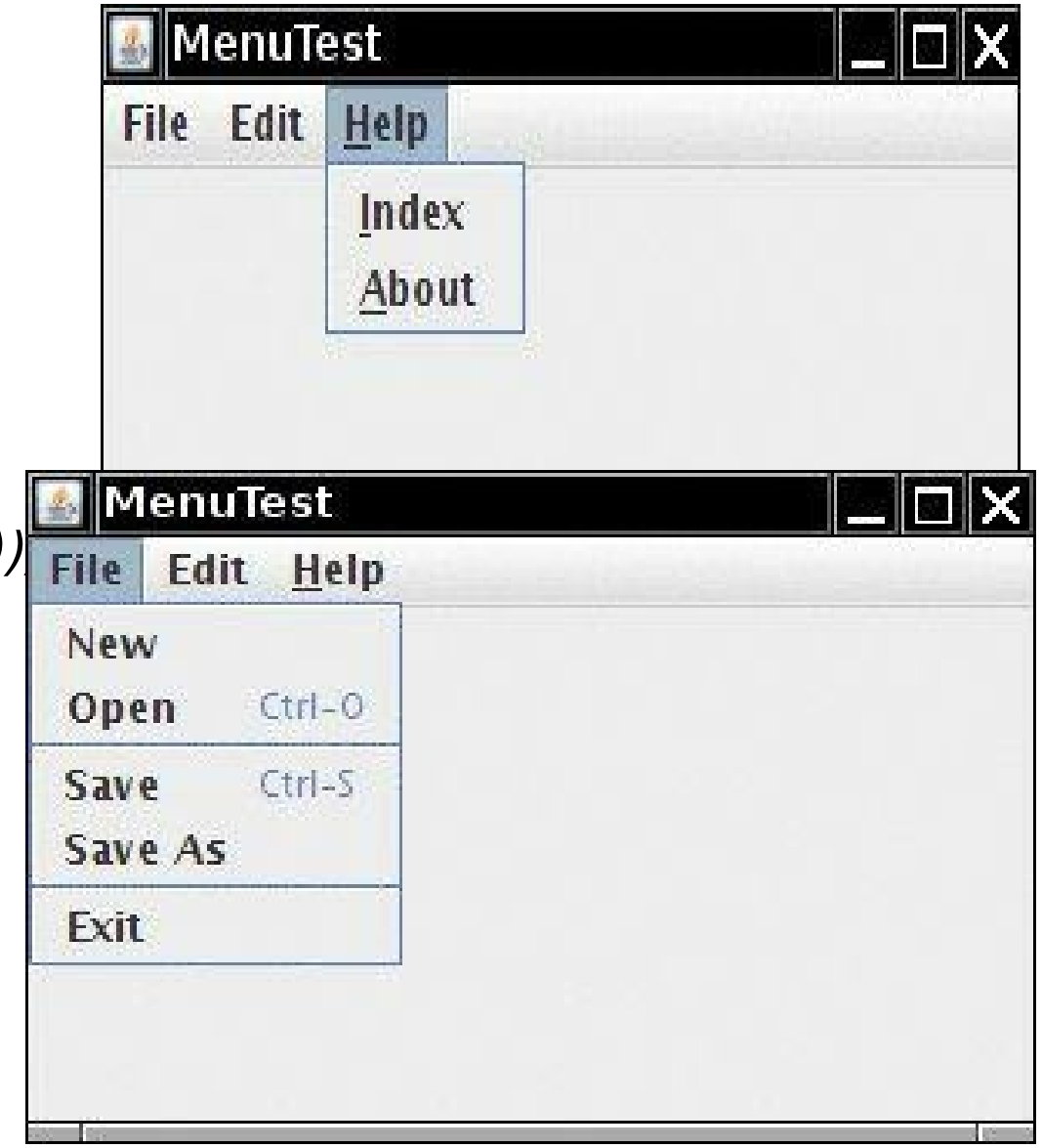
Menus

Keyboard Mnemonics and Accelerators

JMenuItem aboutItem =
new JMenuItem("About", 'A');

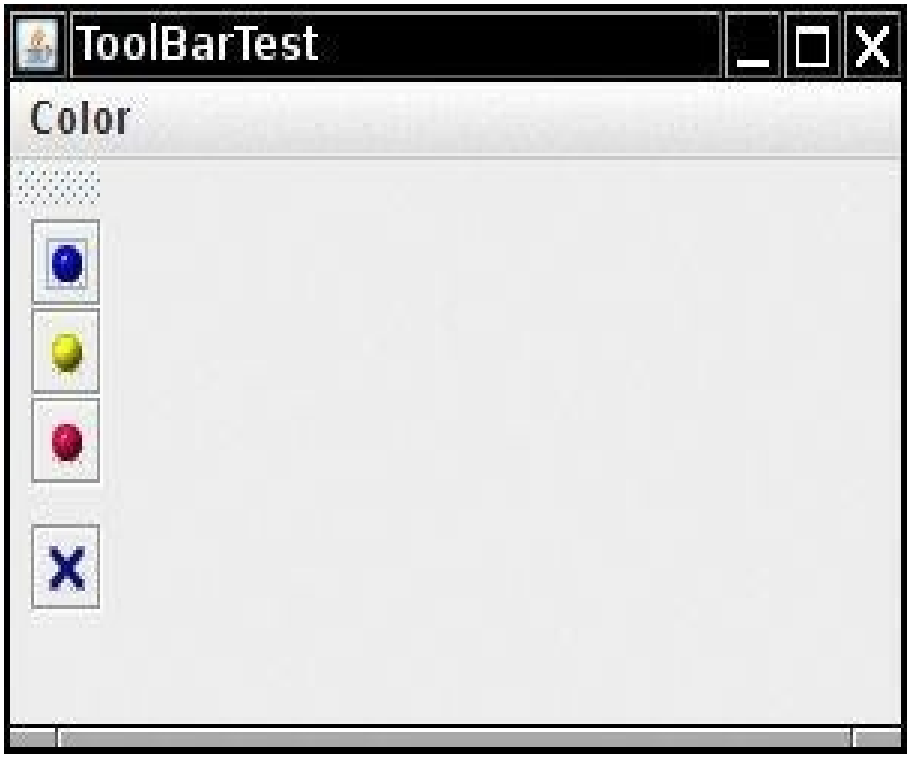
menu.setMnemonic(''H);

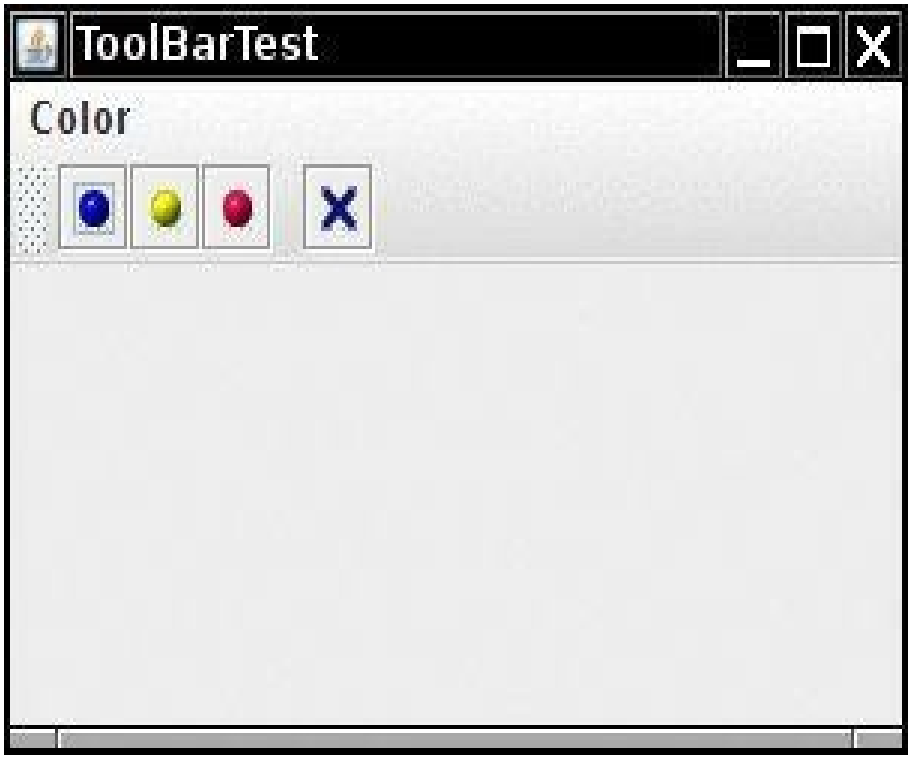
menuItem.setAccelerator
(KeyStroke.getKeyStroke("ctrl O"));



ToolBars

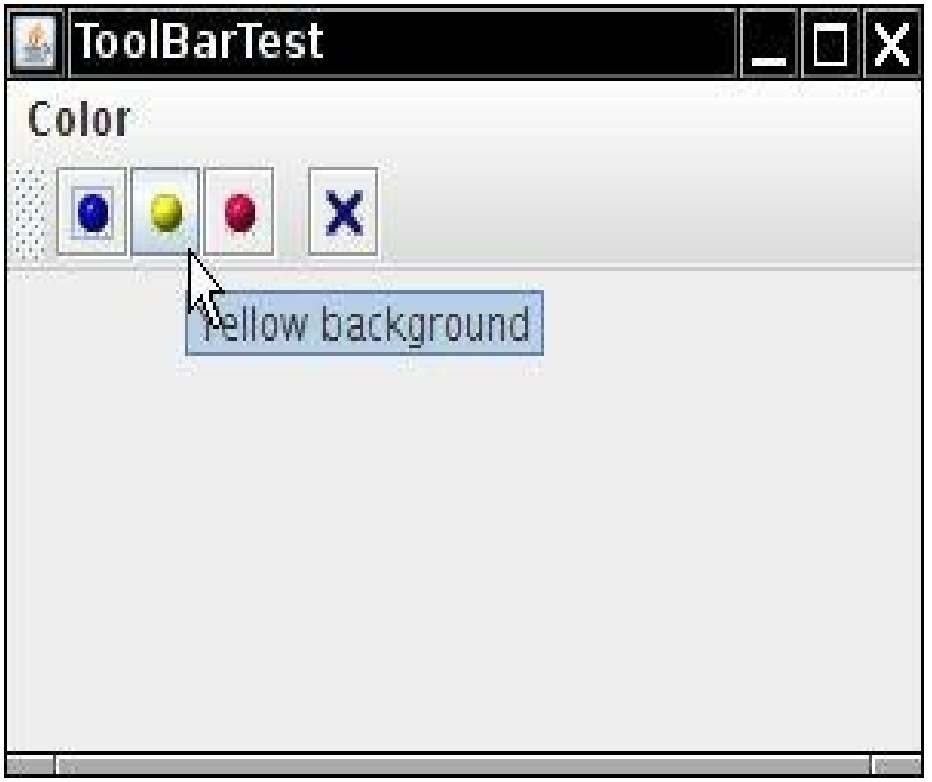
JToolBar





ToolTips

Component.setToolTipText(“”)



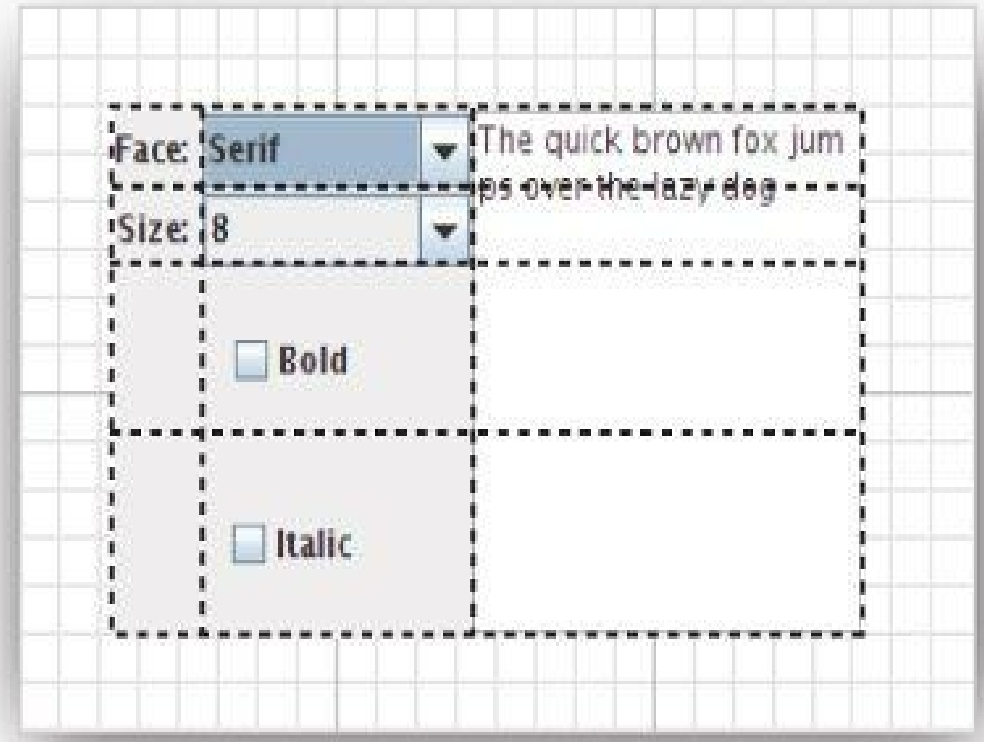
GridBag Layout

```
JPanel pane = new JPanel(new GridBagLayout());
GridBagConstraints c = new GridBagConstraints();

pane.add(theComponent, c);
```

→ gridx, gridy, gridwidth, and gridheight
→ weightx and weighty
→ fill and anchor

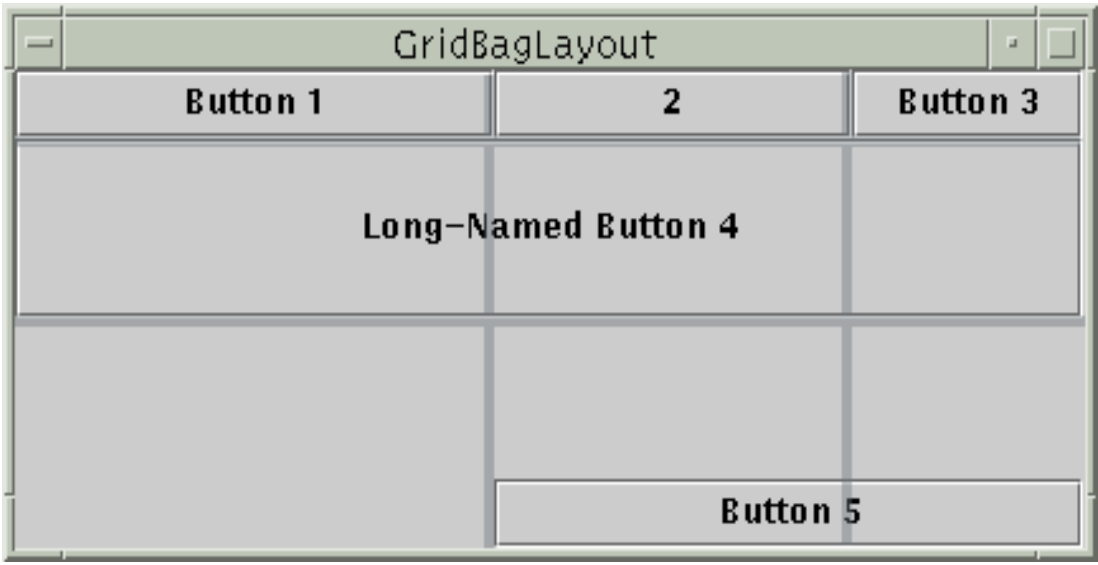
```
-------------------------------------------------
|FIRST_LINE_START    PAGE_START     FIRST_LINE_END|
|                                                 |
|                                                 |
|LINE_START            CENTER           LINE_END|
|                                                 |
|                                                 |
|LAST_LINE_START      PAGE_END       LAST_LINE_END|
-------------------------------------------------
```

→ ipadX, ipadY and insets



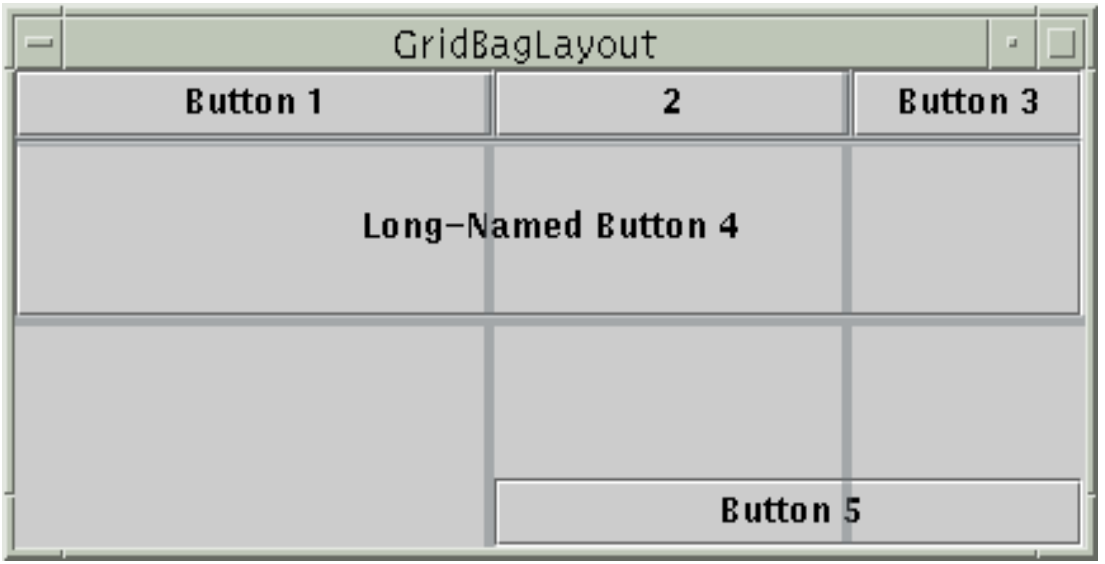
GridBag Layout

```
]GridBagConstraints c = new GridBagConstraints();
c.fill = GridBagConstraints.HORIZONTAL;

button = new JButton("Button 1");
c.weightx = 0.5;
c.gridx = 0;
c.gridy = 0;
pane.add(button, c);

button = new JButton("Button 2");
c.gridx = 1;
c.gridy = 0;
pane.add(button, c);

button = new JButton("Button 3");
c.gridx = 2;
c.gridy = 0;
pane.add(button, c);
```



GridBag Layout

```
button = new JButton("Long-Named Button 4");
c.ipady = 40;      //make this component tall
c.weightx = 0.0;
c.gridwidth = 3;
c.gridx = 0;
c.gridy = 1;
pane.add(button, c);

button = new JButton("5");
c.ipady = 0;       //reset to default
c.weighty = 1.0;   //request any extra vertical space
c.anchor = GridBagConstraints.PAGE_END; //bottom of space
c.insets = new Insets(10,0,0,0);  //top padding
c.gridx = 1;       //aligned with button 2
c.gridwidth = 2;   //2 columns wide
c.gridy = 2;       //third row
pane.add(button, c);
```



Using No Layout

1. Set the layout manager to null.
2. Add the component you want to the container.
3. Specify the position and size that you want:

```
frame.setLayout(null);
JButton ok = new JButton("OK");
frame.add(ok);
ok.setBounds(10, 10, 30, 15);
```

Using Custom Layout

→ implement the LayoutManager interface
→ override the following five methods:

```
void addLayoutComponent(String s, Component c);
void removeLayoutComponent(Component c);
Dimension preferredLayoutSize(Container parent);
Dimension minimumLayoutSize(Container parent);
void layoutContainer(Container parent);
```

Group Layout

Home Work :)

Dialog Box

→ Model
→ Modelless

Message Dialog
Input Dialog
Confirm Dialog
Option Dialog

Message Dialog

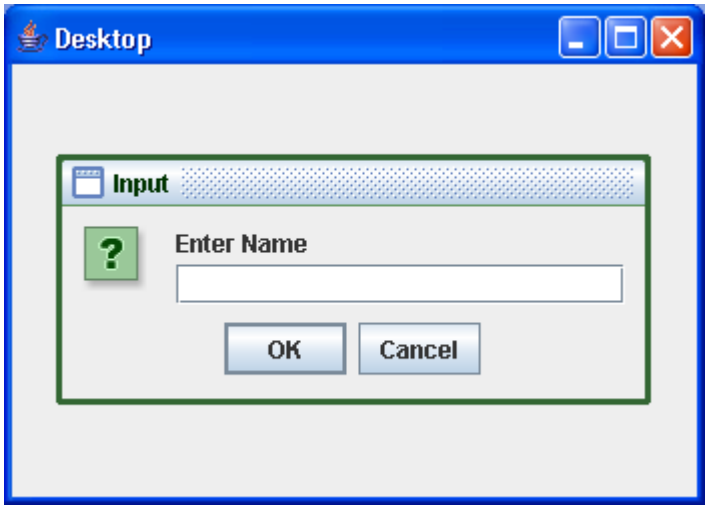
int mc = JoptionPane.WARNING_MESSAGE;
JOptionPane.showMessageDialog (null, "Message", "Title", mc);

1) Specifies the window. Null means the center of the screen.
2) The text shown in the box.
3) The title of the box
4) The icon type

The icon types are:
WARNING_MESSAGE: A yellow triangle with an exclamation mark.
QUESTION_MESSAGE: A question mark.
ERROR_MESSAGE: A stop sign with a X in it.
INFORMATION_MESSAGE: A purple circle with an I in it.
PLAIN_MESSAGE: No icon.

Input Dialog

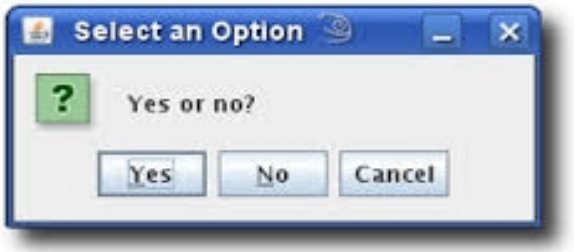
*int mc = JOptionPane.*QUESTION_MESSAGE*;*
String str = JOptionPane.showInputDialog (null, "Enter Name:", "Input", mc);



Confirm Dialog

```
int mc = JOptionPane.QUESTION_MESSAGE;
int bc = JOptionPane.YES_NO_CANCEL_OPTION;
int ch = JOptionPane.showConfirmDialog (null, "Select:",
"Title", bc, mc)
```

JOptionPane.YES_NO_OPTION: Yes, no.
JOptionPane.YES_NO_CANCEL_OPTION: Yes, no, cancel.
JOptionPane.OK_CANCEL_OPTION: Ok, cancel.



Note: the function will return the button pressed.
JOptionPane.OK_OPTION: OK was clicked.
JOptionPane.CANCEL_OPTION: Cancel was clicked.
JOptionPane.YES_OPTION: Yes was clicked.
JOptionPane.NO_OPTION: No was clicked.
JOptionPane.CLOSED_OPTION: The box was closed.

Option Dialog

```
int mc = JOptionPane.QUESTION_MESSAGE;
String[] opts = {"Java", "C++", "VB", "PHP", "Perl"};
int ch = JOptionPane.showOptionDialog (null, "What
language do you prefer", "Option Dialog Box", 0, mc, null,
opts, opts[3])
```

1) Specifies the window. Null means the center of the screen.
2) The text shown in the box.
3) The title of the box.
4) Unused for OptionDialog, use 0.
5) The icon .
6) The icon object, use null for this one.
7) The buttons shown, a string array must be supplied.
8) The default button selected.

Creating Dialogs

```
public class AboutDialog extends JDialog {
    public AboutDialog(JFrame owner) {
        super(owner, "About DialogTest", true);
        add(new Jlabel(“Custom Dialog” ),BorderLayout.CENTER);
        JPanel panel = new JPanel();
        JButton ok = new JButton("OK");
        ok.addActionListener(new
                                ActionListener() {
                                    public void actionPerformed(ActionEvent event) {
                                        setVisible(false);
                                    }
                                });
        panel.add(ok);
        add(panel, BorderLayout.SOUTH);
        setSize(250, 150);
    }
}
```

1. In the constructor of your dialog box, call the constructor of the superclass JDialog.
2. Add the user interface components of the dialog box.
3. Add the event handlers.
4. Set the size for the dialog box.

Data Exchange

ClassWork

File Chooser

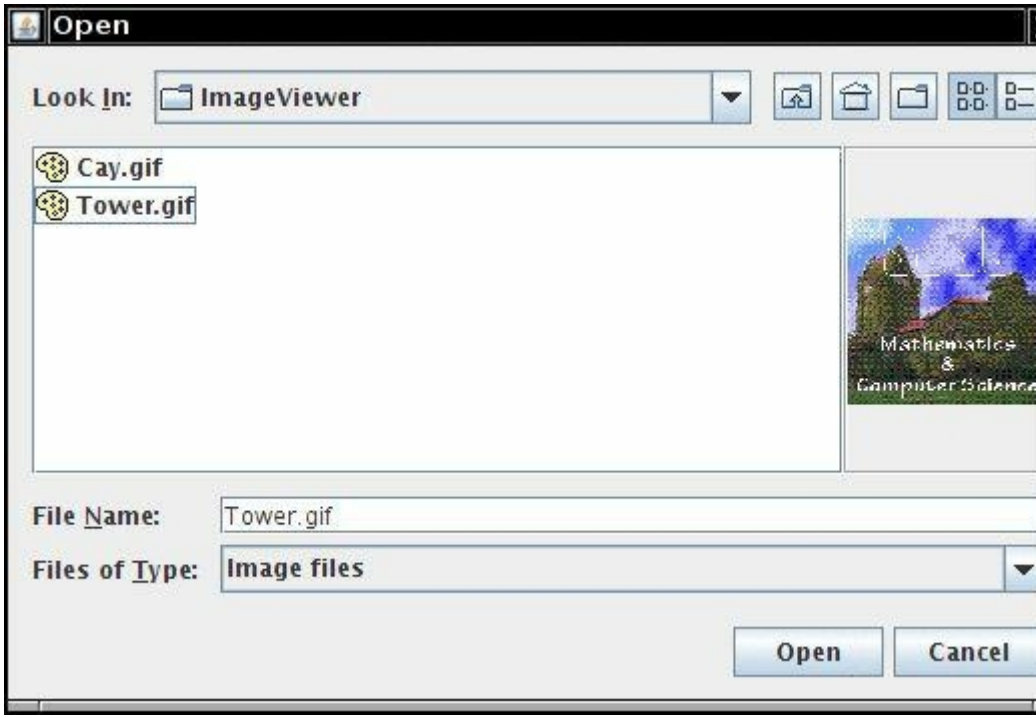
```
JFileChooser  fileDialog = new JfileChooser();
int returnVal = fileDialog.
                showOpenDialog(component);
```

JFileChooser.APPROVE_OPTION,
JFileChooser.CANCEL_OPTION, or
JfileChooser.ERROR_OPTION

```
java.io.File file = fileDialog.getSelectedFile();
```

```
fileDialog.setMultiSelectionEnabled(true);
```

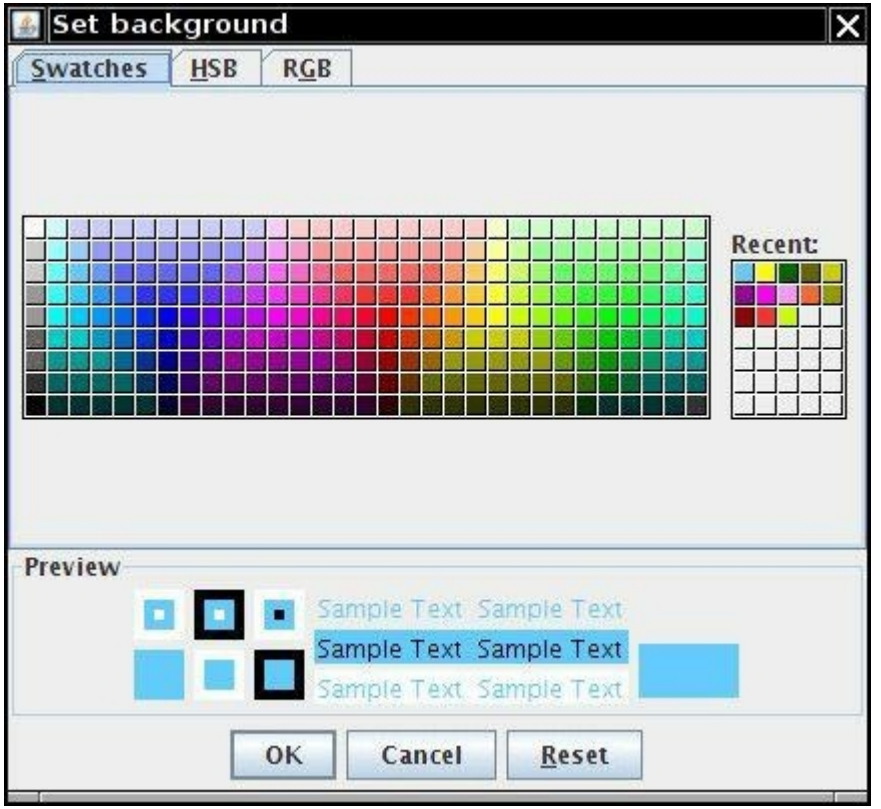
```
fileDialog.getSelectedFiles();
```




```
fileDialog.setFileFilter(new FileNameExtensionFilter("Text Files", "txt",
"csv");
```

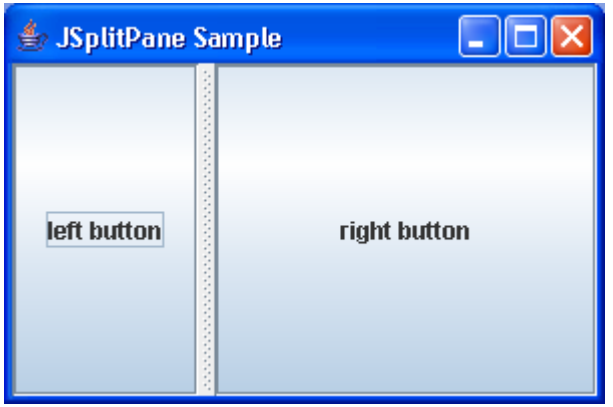
Color Chooser

```
Color backgroundColor = JColorChooser.showDialog(mainFrame,
                        "Set background", Color.white);
```

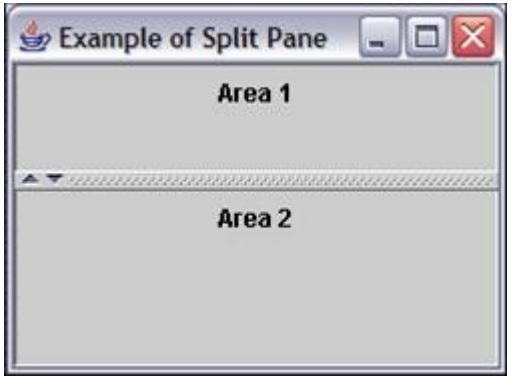


Split pane

```
JSplitPane splitPane = new JSplitPane(JSplitPane.VERTICAL_SPLIT,
                    true, component1,component2);
```




```
splitPane.setOneTouchExpandable(true);
```

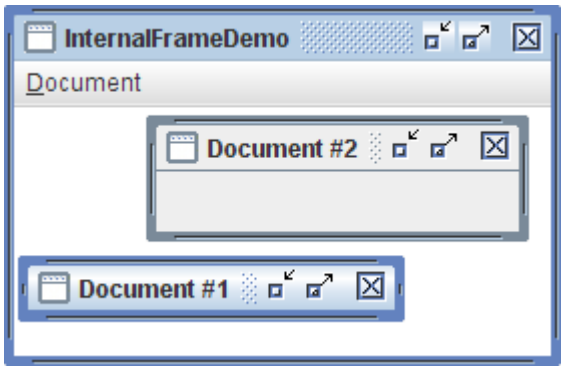


Tabbed pane

```
JTabbedPane jtp = new JTabbedPane();

jtp.addTab("Tab1", component1);
jtp.addTab("Tab2", component2);

jtp.setMnemonicAt(0, KeyEvent.VK_A);
jtp.setMnemonicAt(1, KeyEvent.VK_B);
```

Desktop pane and internal frame

```
    JdesktopPane desktop = new JdesktopPane();

JInternalFrame internalFrame = new JInternalFrame(
                                "Internal Frame", true, true, true, true);

desktop.add(internalFrame);

internalFrame.setVisible(true);
```



title
resizable
closable
maximizable
iconifiable

Cascade and Tile

HomeWork

References

- http://searchsoftwarequality.techtarget.com/definition/pattern
- http://www.cs.wcupa.edu/rkline/java/mvc-design.html
- http://bip.weizmann.ac.il/course/prog2/tutorial/uiswing/layout/gridbag.htm
- http://www.dreamincode.net/forums/topic/22739-message-dialogs-in-java/
- http://www.java-tips.org/java-se-tips/javax.swing/how-to-make-split-pane-using-swing.html
- http://docs.oracle.com/javase/tutorial/uiswing/components/splitpane.html