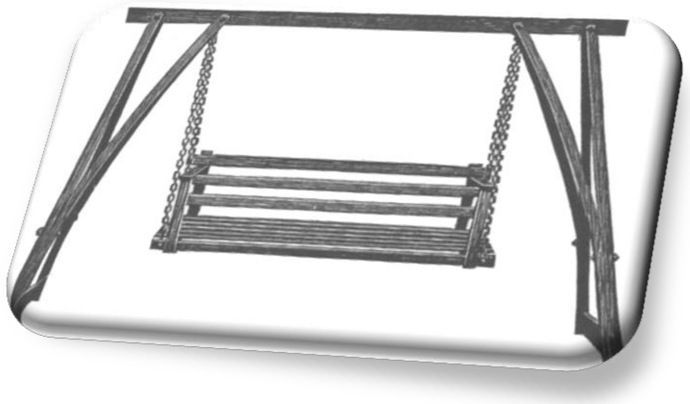


Swing

GUI Widget Toolkit

What is Swing?

- A group of 14 GUI packages
- 451 classes as of JFC 1.4
- Part of JFC Java Foundation Classes (compare now defunct MFC)

Swing Components

- General and Special Purpose Containers
- Basic Controls
- Uneditable Information Displays
- Interactive Displays

Swing and Threading

- Swing and thread are lightweight processes
- Most Swing components are not thread-safe
- Swing components executes on the 'event-dispatching' thread

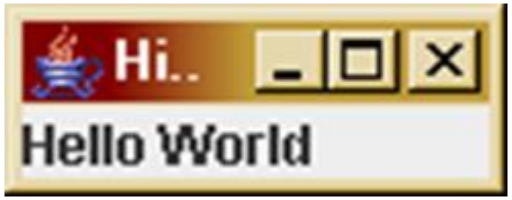
Launching Swing

- Start a Swing application using the following code:

```
public static void main(String[] args)
{
    SwingUtilities.invokeLater(new Runnable()
    {
            public void run()
            {
                    createAndShowGUI();
            }
    } );
}
```

Example Startup

```
private static void createAndShowGUI()
{
   JFrame frm = new JFrame("Hi..");
   frm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   JLabel label = new JLabel("Hello World");
   frm.getContentPane().add(label);
   frm.pack();
   frm.setVisible(true);
}
```



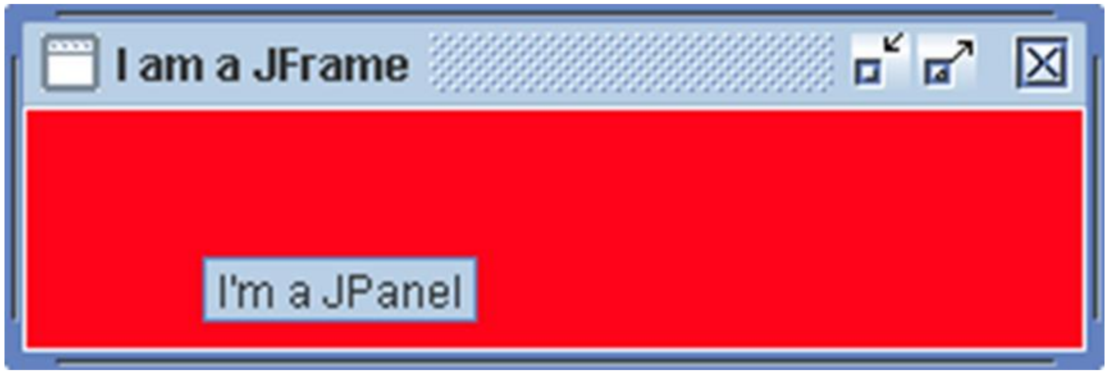
Containers

- Top Level
  - JFrame
  - JApplet
  - JDialog

- General Purpose
  - Panel
  - Scroll pane
  - Split pane
  - Tabbed pane
  - Tool bar

JPanel

- Subclass of JComponent
- Borders can be added
- Tooltips too



JSplitPane

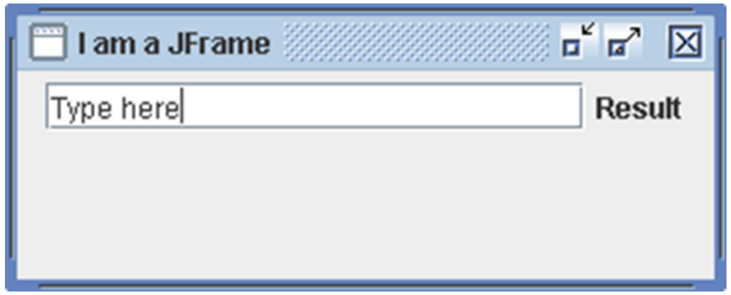
```
//Create a split pane
JSplitPane myPane = new JSplitPane();
myPane.setDividerLocation(150);
// make two panels
JPanel right = new JPanel();
right.setBackground(new Color(255,0,0));
JPanel left = new JPanel();
left.setBackground(new Color(0,255,0));
// set as left and right in split
myPane.setRightComponent(right);
myPane.setLeftComponent(left);
```

JTextField

- For single-line text input
- Methods getText, setText
- Can use ActionListener, triggered when Enter pressed

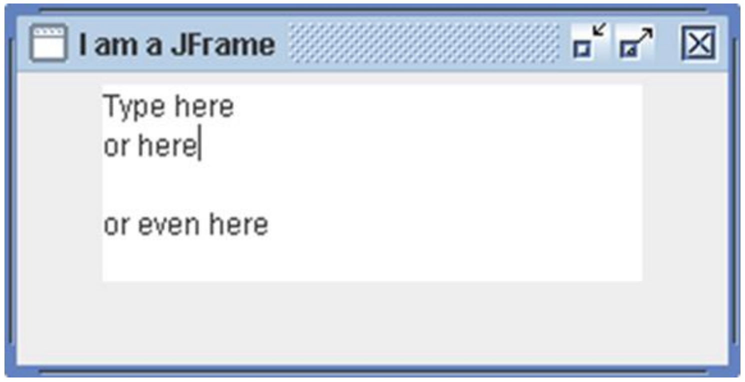
Using JTextField

- Make a panel, set as content pane
- Make and add text field
- Add actionlistener
- Make and add a label
- Program actionPerformed



JTextArea

```
JPanel myPanel = new JPanel();
app.textArea = new JTextArea("Type here",5, 20);
myPanel.add(app.textArea);
```



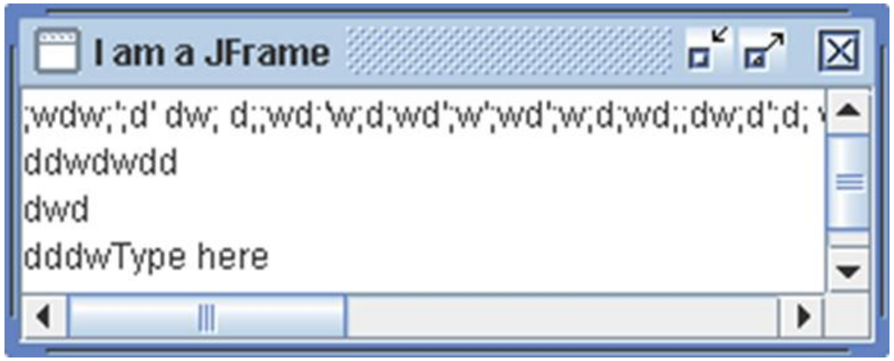
```
Type here
or here

or even here
```

TextArea expands rows and columns as needed

JScrollPane

```
JTextArea textArea = new JTextArea("Type here",5, 20);
JScrollPane scrollPane = new JScrollPane(textArea);
frame.setContentPane(scrollPane);
```



```
;wdw;';d' dw; d;;wd;'w;d;wd';w';wd';w;d;wd;;dw;d';d;
ddwdwdd
dwd
dddwType here
```

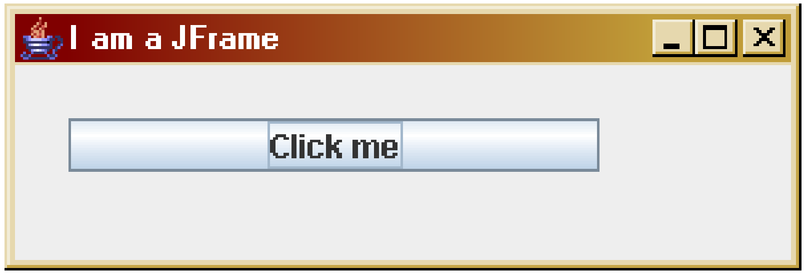
Layout Managers

- Swing uses a *LayoutManager* to control positioning of items
- There is a choice of these which work in different ways
- To do without one:

```
frame.setLayout(null);
```

Absolute Positioning

```
JFrame frame = new JFrame("I am a JFrame");
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setBounds(20,30,300,100);
frame.setLayout(null);
JButton butt=new JButton("Click me");
frame.getContentPane().add(butt);
butt.setBounds(20, 20, 200,20);
frame.setVisible(true);
```



Layout Management

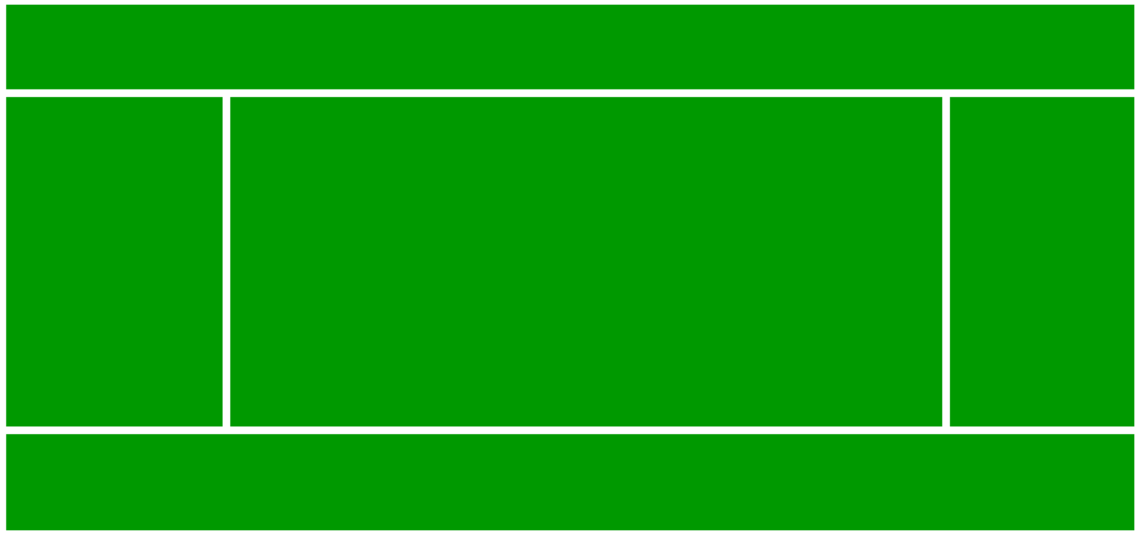
- Java supplies five commonly used layout managers:

1. BorderLayout
2. BoxLayout
3. FlowLayout
4. GridBagLayout
5. GridLayout

BorderLayout

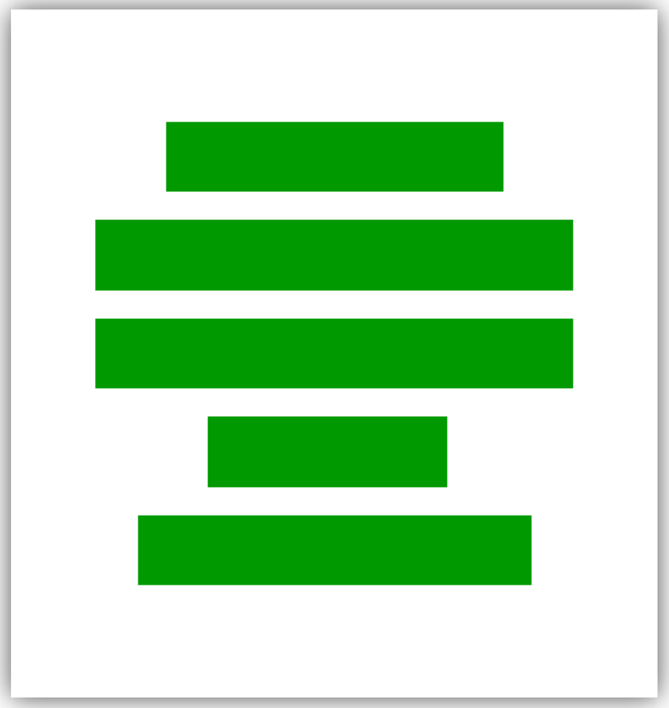
- Position must be specified:

add (“North”, myComponent)



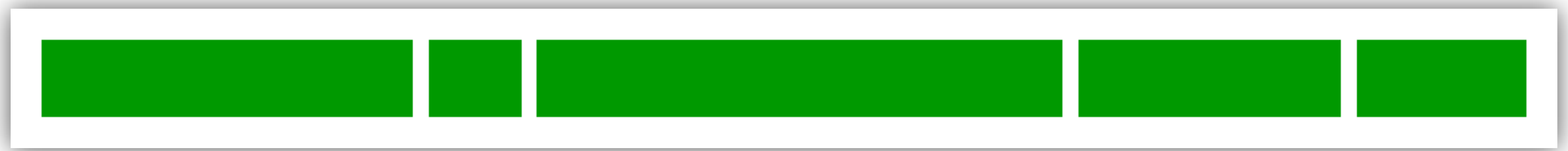
BoxLayout

- The BoxLayout class puts components in a single row or column.
- It respects the components‘ requested maximum sizes.



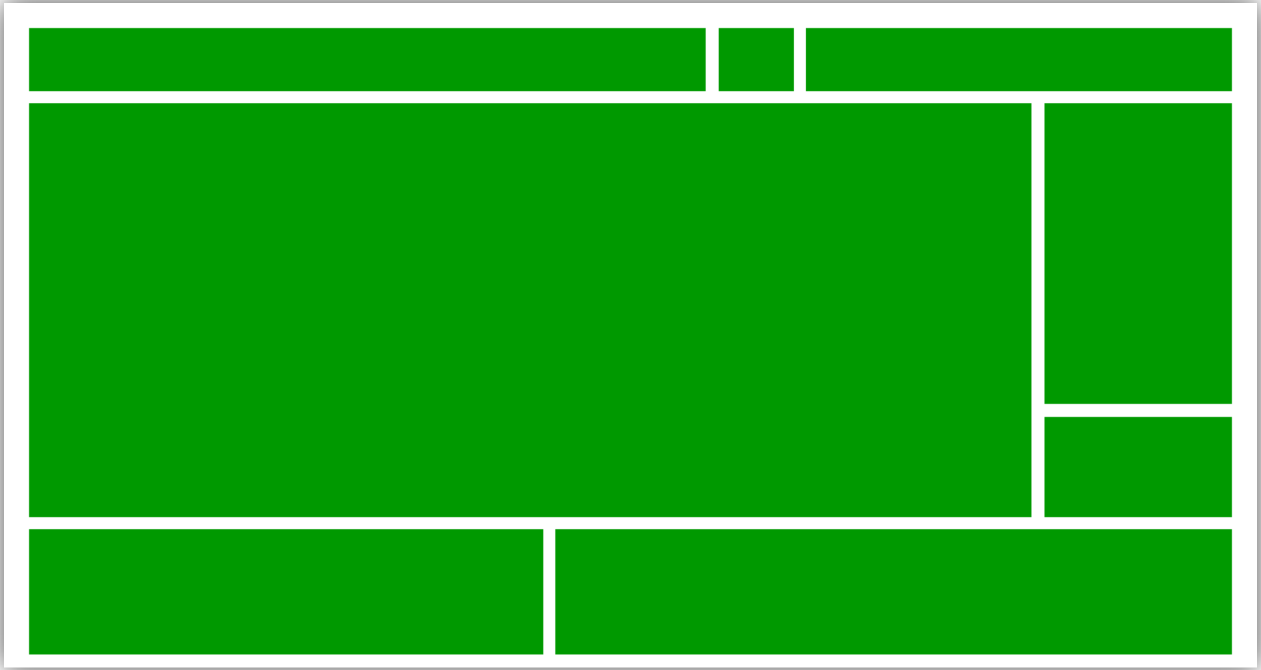
FlowLayout

- FlowLayout is the default layout manager for every JPanel.
- It simply lays out components from left to right, starting new rows if necessary



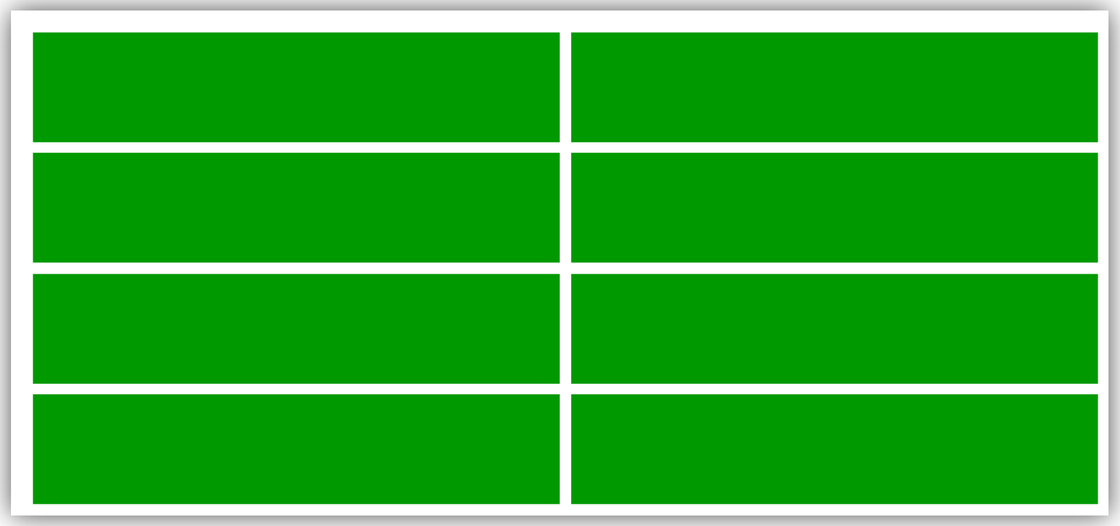
GridBagLayout

- GridBagLayout is the most sophisticated, flexible layout manager the Java platform provides.

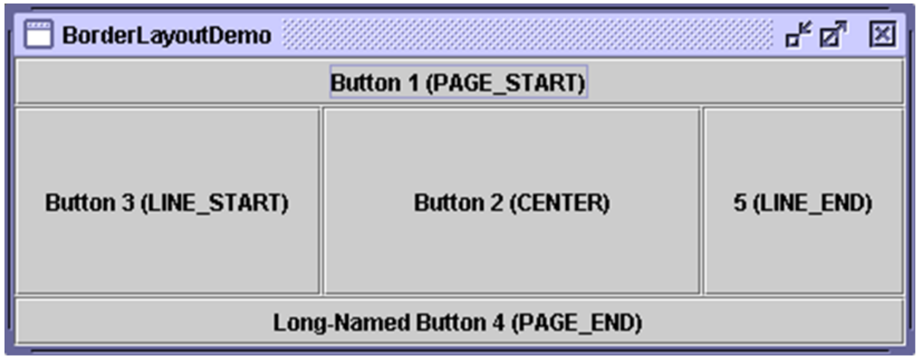


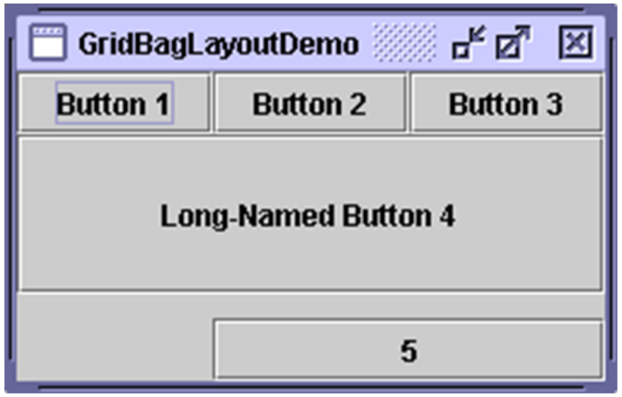
GridLayout

- GridLayout simply makes a bunch of components equal in size and displays them in the requested number of rows and columns

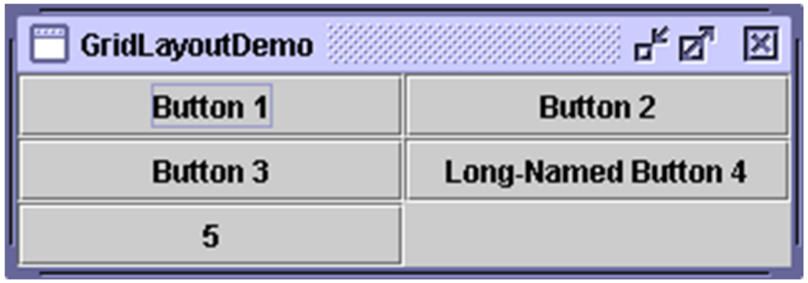


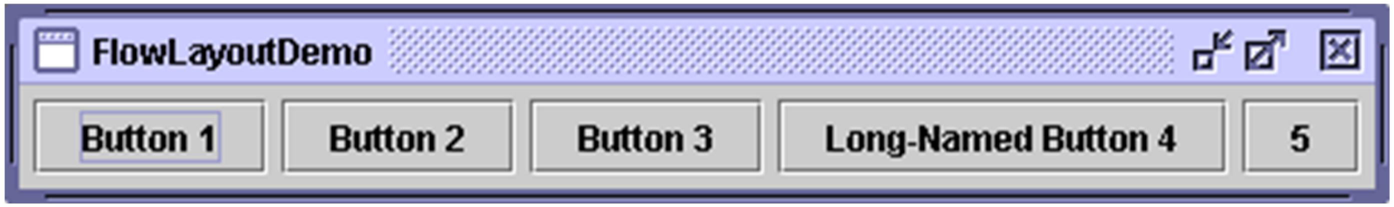
Sample Layouts





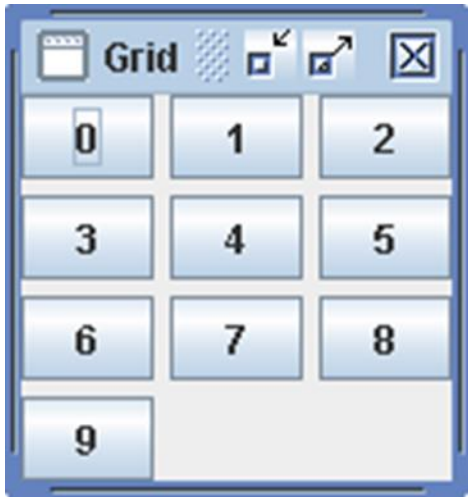
from Swing tutorial on java.sun.com





Grid

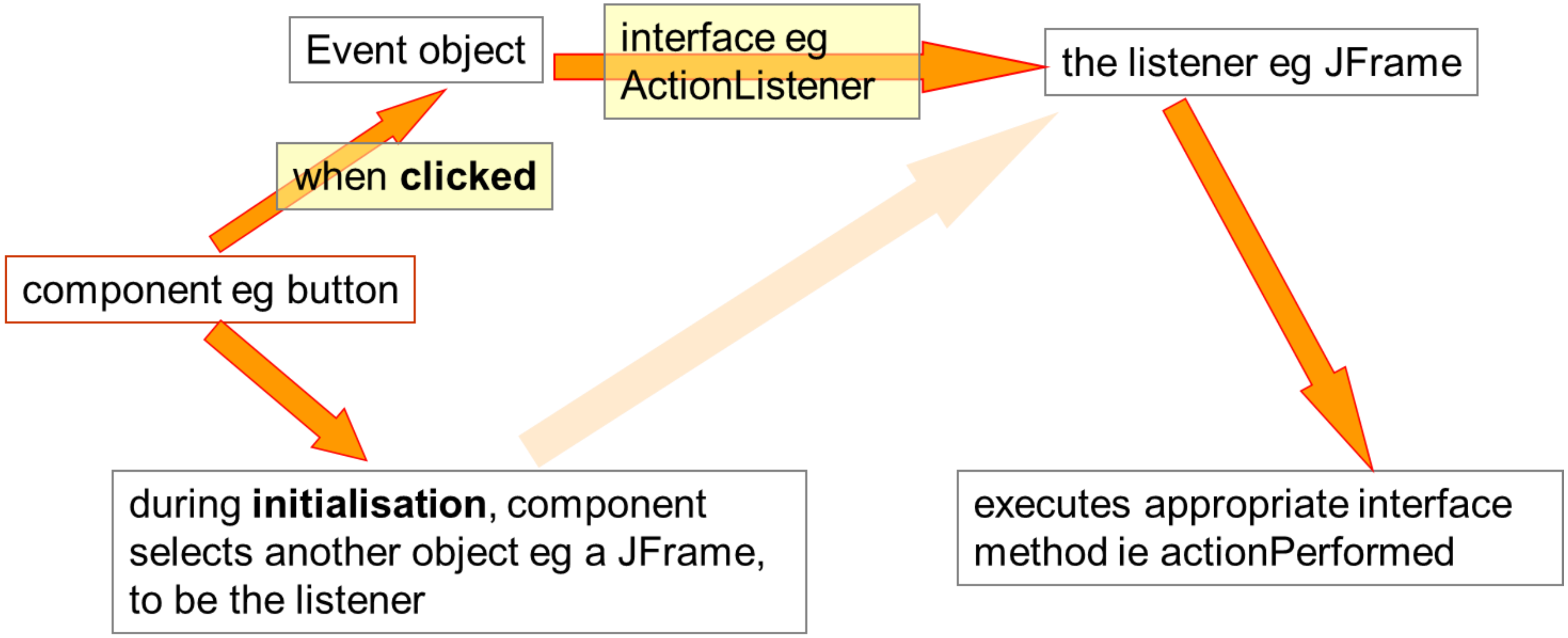
```
JFrame.setDefaultLookAndFeelDecorated(true);
JFrame frame = new JFrame("Grid");
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.getContentPane().setLayout(new GridLayout(4,3,5,5));
for (int i=0; i<10; i++)
    frame.getContentPane().add(new JButton(""+i));
frame.pack();
frame.setVisible(true);
```



Events

- Based on an event-handling model
- New component implements a *Listener*
- The *Listener* object is responds to Event objects coming from the component
- The Listener object needs to implement the appropriate *interface*

Event Handling


Event object
interface eg ActionListener
the listener eg JFrame
when **clicked**
component eg button
during **initialisation**, component selects another object eg a JFrame, to be the listener
executes appropriate interface method ie actionPerformed

Interfaces

- The ActionListener interface has just one method:

```
public void actionPerformed(ActionEvent e)
```

- A class implementing the ActionListener interface defines that method
- Or it’s a compile error

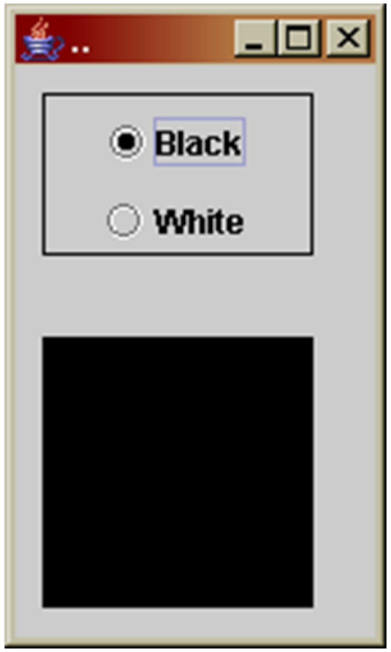
Which Button

- If have several buttons, all must link to actionPerformed.
- How to know which button was clicked?
- Use the .getSource method of the ActionEvent object

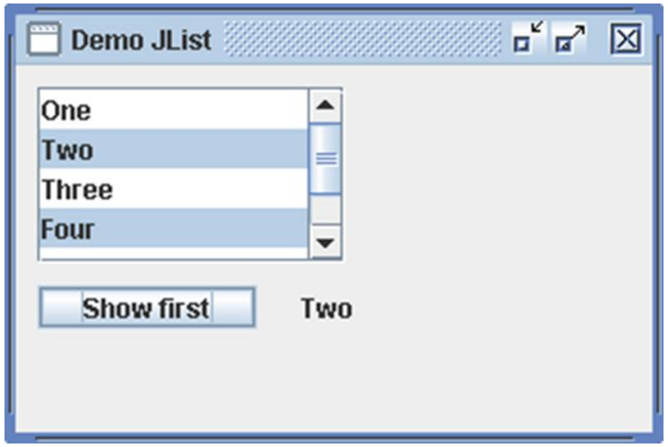
```
button1=new JButton("Button 1");
..
button2 = new JButton("Button 2");
..
public void actionPerformed(ActionEvent e)
{
  if (e.getSource()==button1)
      label.setText("Button1 clicked");
  else
      label.setText("Button2 clicked");
}
```

Groups

```
JPanel groupPanel = new JPanel();
groupPanel.setBounds(10,10,100,60);
groupPanel.setBorder(BorderFactory.createLineBorder
  (Color.black));
frame.getContentPane().add(groupPanel);
groupPanel.add(app.choice1);
groupPanel.add(app.choice2);
```



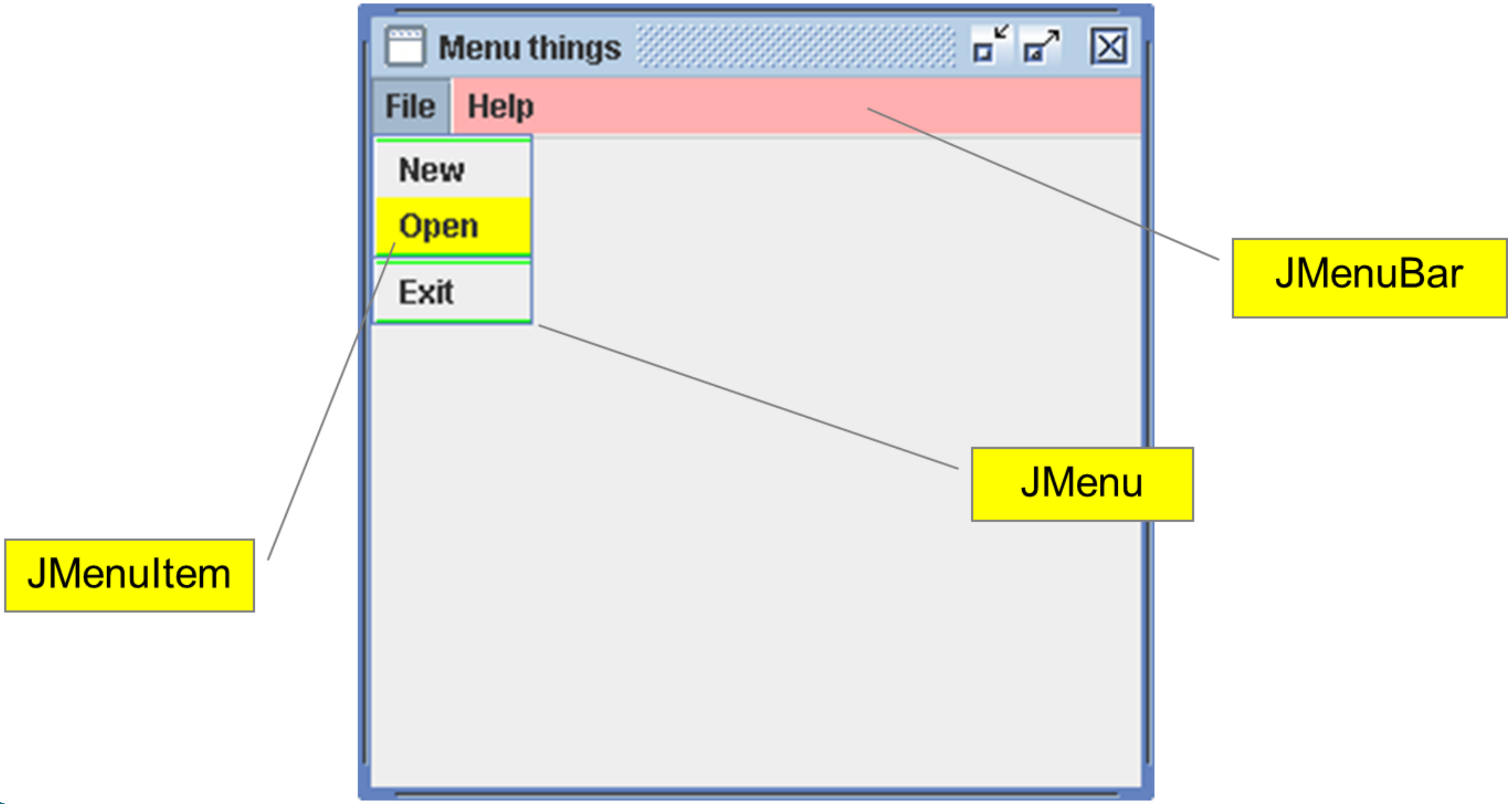
ListBox



- Data held in array
- List box shows array
- List box inside scroll pane

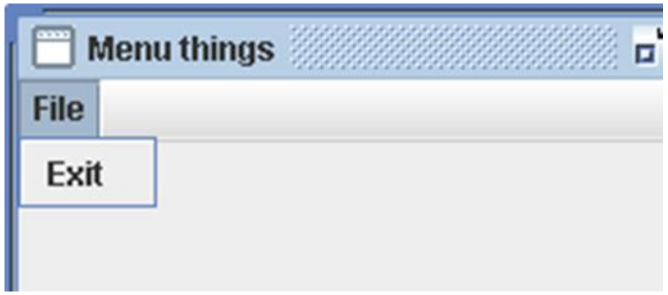
myList.getModel().getElementAt(..

Menus


Menu things
File
Help
New
Open
Exit
JMenuBar
JMenu
JMenuItem

Menu Excerpt

```
JMenuBar myMenuBar = new JMenuBar();
JMenu menu1  = new JMenu("File");
JMenuItem item = new JMenuItem("Exit");
item.addActionListener(app);
menu1.add(item);
myMenuBar.add(menu1);
frame.setJMenuBar(myMenuBar);
..
public void actionPerformed(ActionEvent e)
{
   System.exit(0);
}
```



PaintComponent

- Method of class JComponent
- Inherited to all subclasses, e.g. JPanel, JButton,…
- The place where all custom painting belongs
- Invoked by the event-scheduler or by the repaint() – method

Custom Painting

- Every class derived from Jcomponent can be used for custom drawing.
  - JPanel: Generating and displaying graphs in top of a blank or transparent background
  - JLabel: Painting on top of an image
  - JButton: custom button