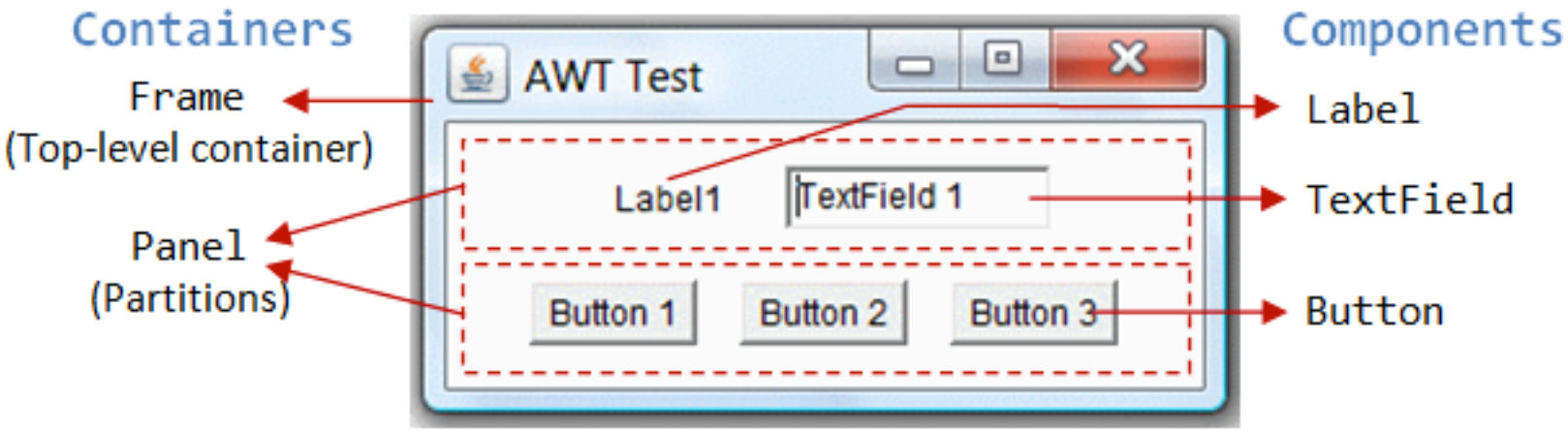
Java Swing

Università di Modena e Reggio Emilia
Prof. Nicola Bicocchi (nicola.bicocchi@unimore.it)

Containers and components



Package java.awt.*

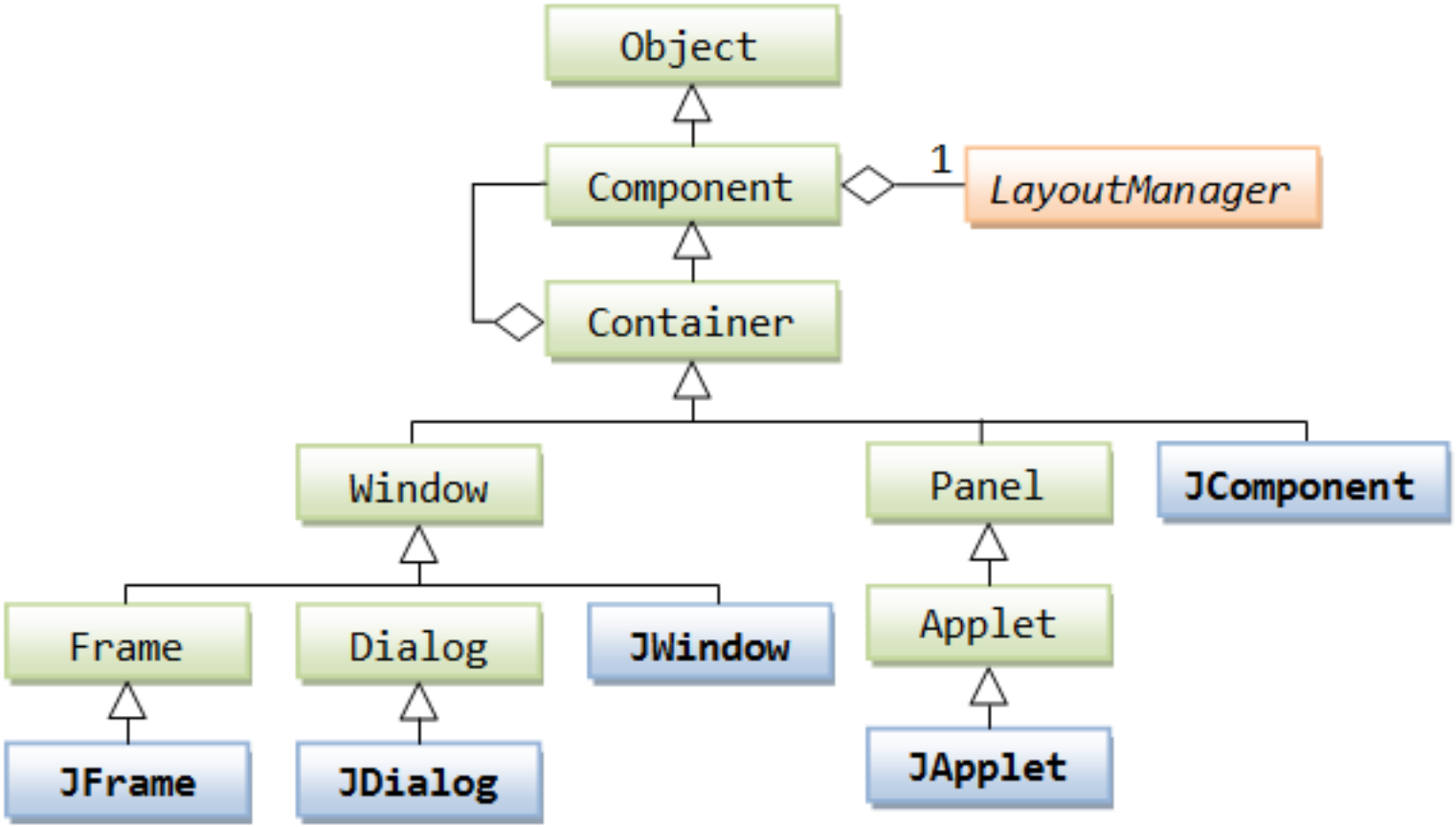
- Provides:
  - Components *(button, checkbox, scrollbar, etc.)*
  - Containers *(they are still components)*
  - Event management:
    - System-generated events
    - UI-generated events
  - Layout management

Package javax.swing.*

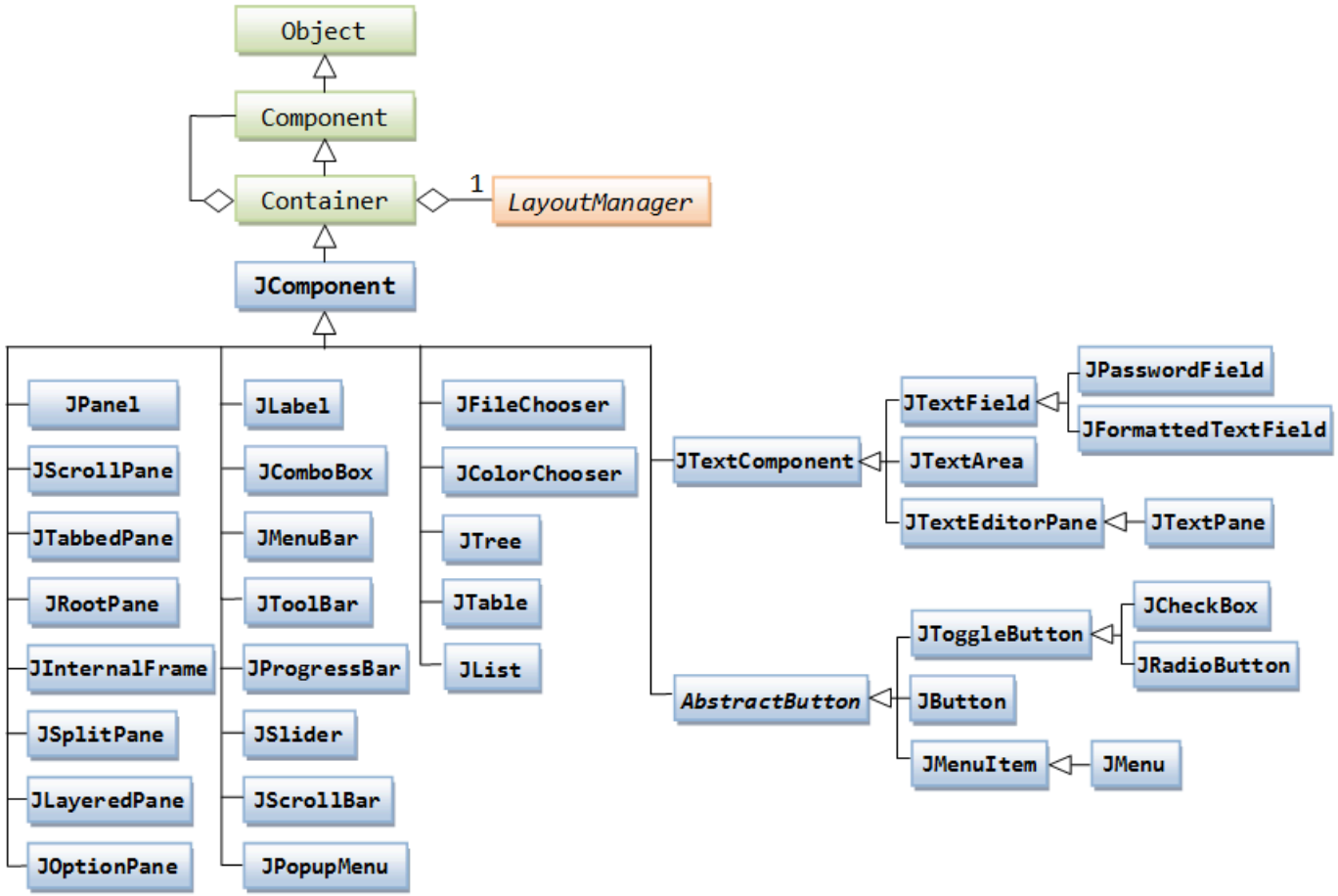
- Contains the same components of java.awt, but with different names (JButton, JFrame, etc.)
- All these components derive from JComponent
- Advantages:
  - provides a series of components light-weight with the same appearance/behavior on all platforms
  - look and feel changeable on the flight
- Swing it is an extension of AWT. However management of the events in the two packages is different

ING

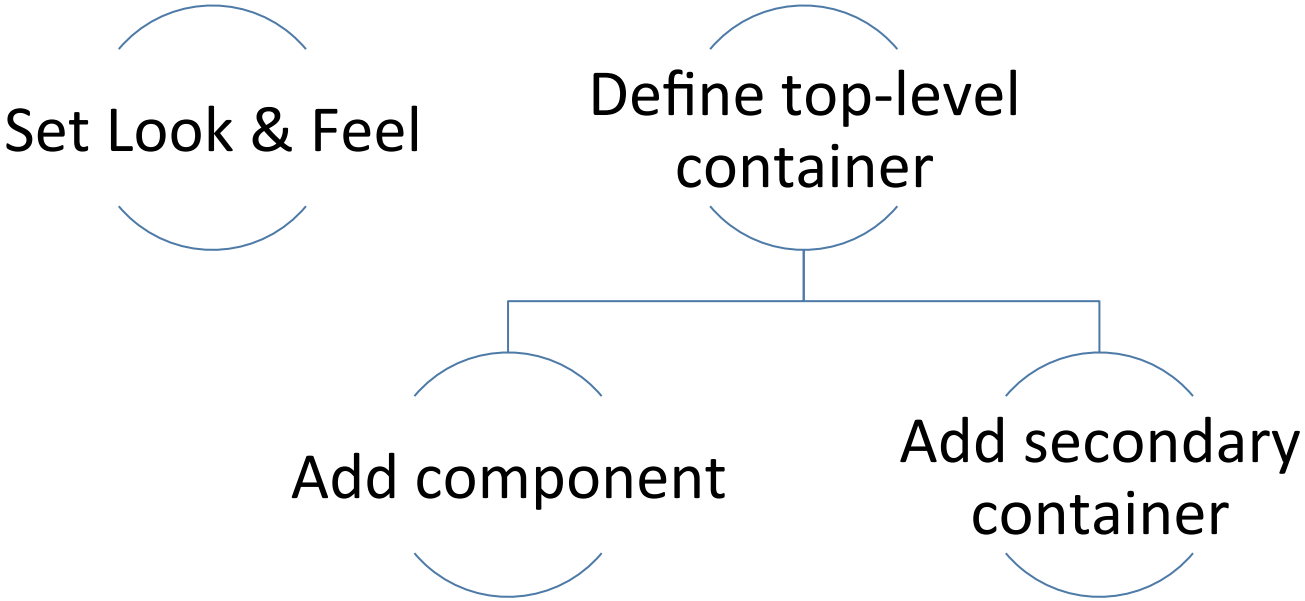
Class hierarchy



Class hierarchy



Graphical Programming

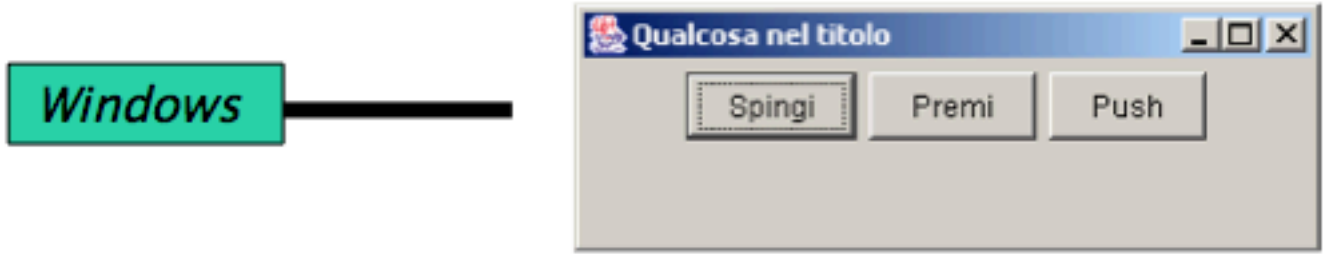


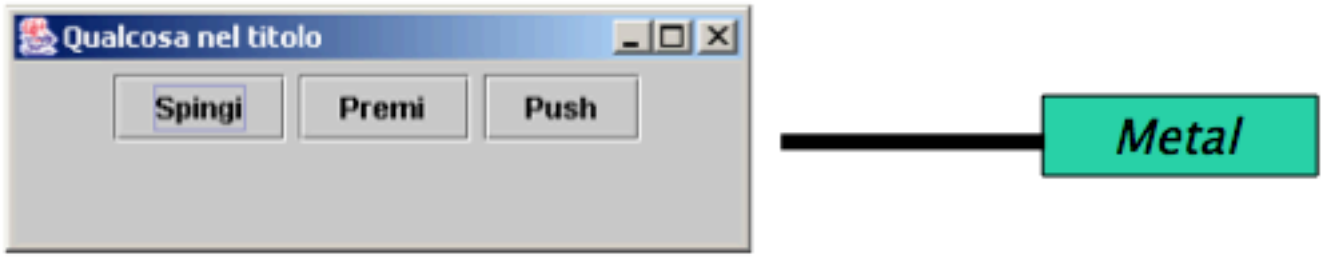
Graphical Programming

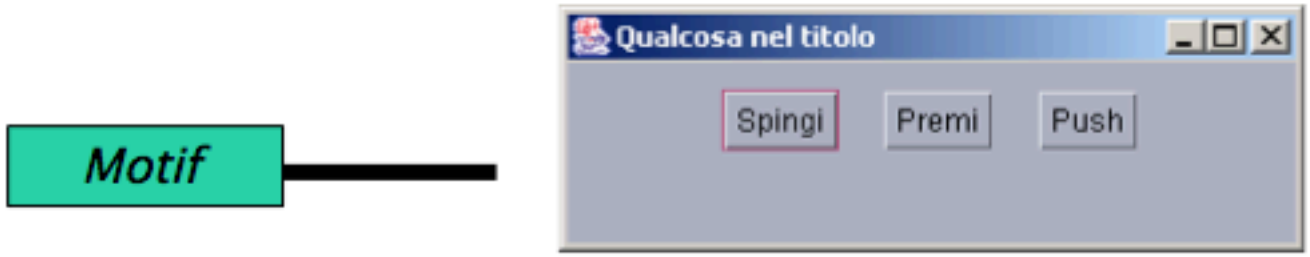
- Set a Look & Feel (= Style)
  - Microsoft Windows, Mac, Java Metal
- Define one (or more) top-level container
  - JFrame, JDialog, JApplet
- Add components to the containers
  - JButton, JComboBox, JSlider, …
- Arrange the components within layouts

ING

Look & Feel


Windows
Qualcosa nel titolo
Spingi
Premi
Push


Qualcosa nel titolo
Spingi
Premi
Push
Metal


Motif
Qualcosa nel titolo
Spingi
Premi
Push

Look & Feel

Sun's JRE provides the following L&Fs:

- **CrossPlatformLookAndFeel**: this is the "Java L&F" (also called "Metal") that looks the same on all platforms. It is part of the Java API (javax.swing.plaf.metal) and is the default.
- **SystemLookAndFeel**: here, the application uses the L&F that is native to the system it is running on.
- **Synth**: the basis for creating your own look and feel with an XML file.
- **Multiplexing**: a way to have the UI methods delegate to a number of different look and feel implementations at the same time.

ING

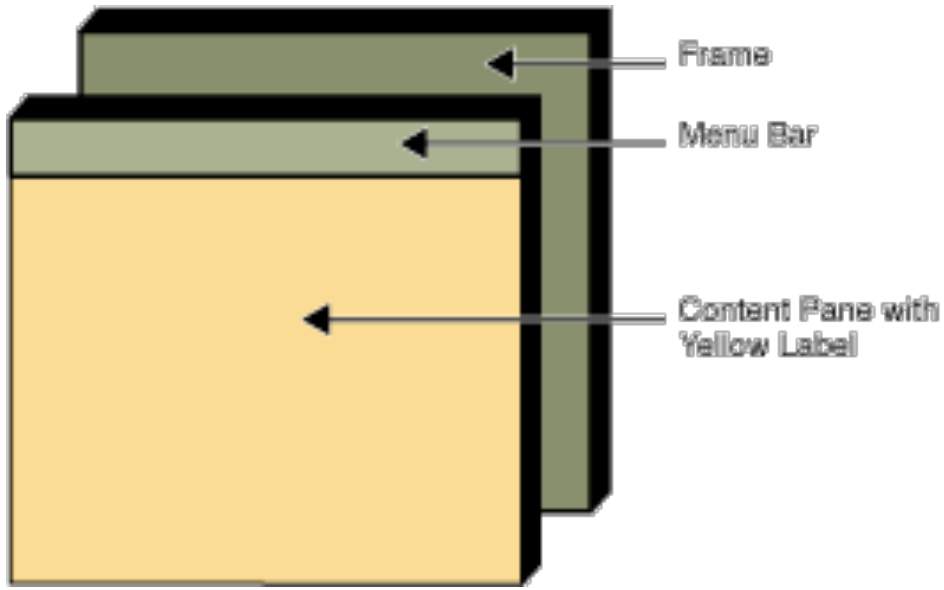
Look & Feel

UIManager manages the current look and feel, and the set of available look and feels.

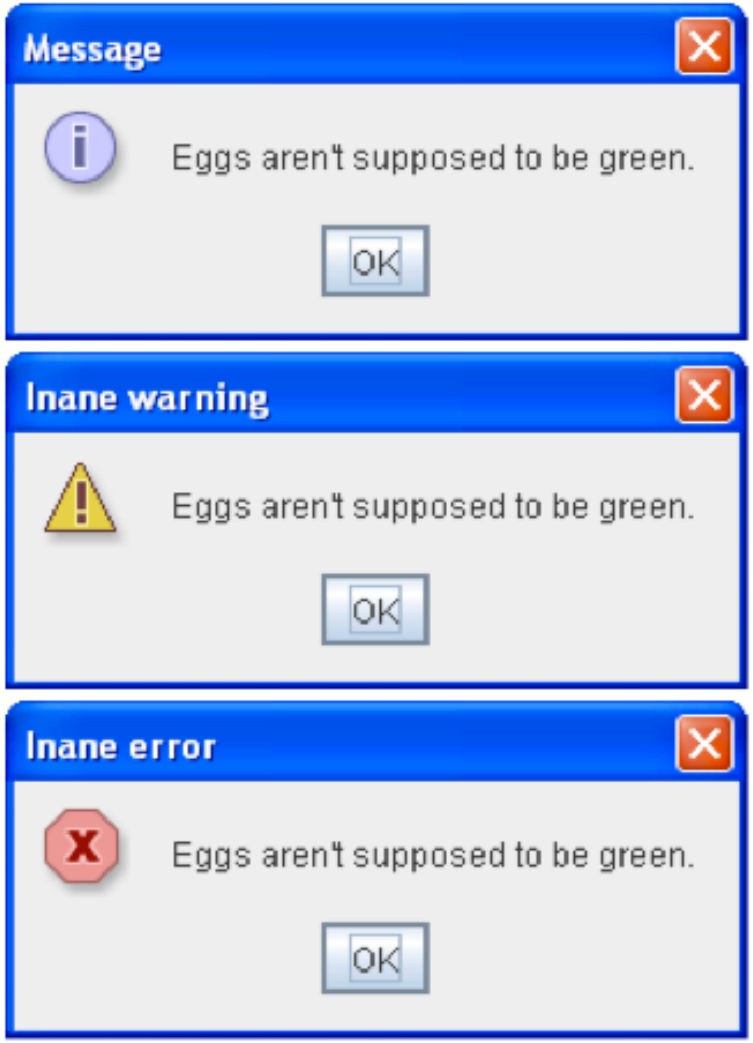
```
// Set cross-platform Java L&F (also called "Metal")
UIManager.setLookAndFeel("javax.swing.plaf.metal.MetalLookAndFeel");

// Set Motif L&F
UIManager.setLookAndFeel("com.sun.java.swing.plaf.motif.MotifLookAndFeel");

// Set Windows L&F
UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WIndowsLookAndFeel");
```

ING

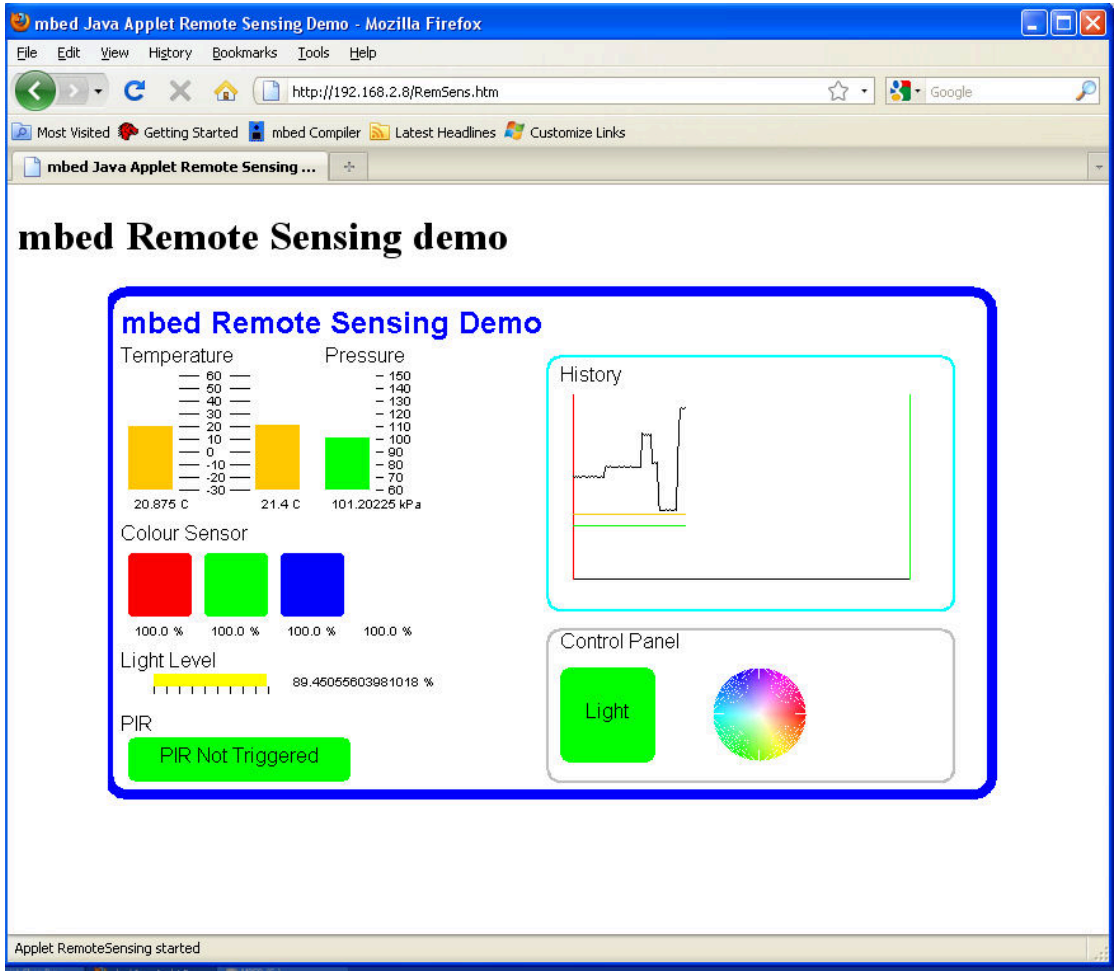
Top-level container: JFrame



Top-level container: JDialog



Top-level container: JApplet

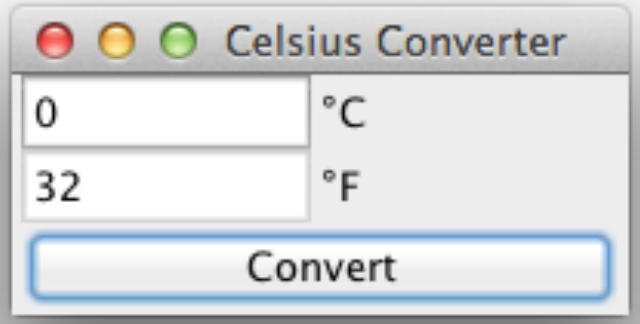


Set a top-level container

- The class must extend the container chosen (Class my_container extends JFrame)
- It is necessary to create at first a secondary container.
  - JPanel p = new JPanel(); setcontentpane (p);
- components must be added to the secondary container:
  - p.add(new JButton());

A complete example

```
public class CelsiusConverter extends JFrame implements ActionListener {
        private JButton convertButton;
        private JTextField fahrenheitLabel;
        private JTextField tempTextField;

        public CelsiusConverter() {
                tempTextField = new JTextField("0");
                fahrenheitLabel = new JTextField("32”);
                fahrenheitLabel.setEditable(false);
                convertButton = new JButton("Convert");

                JPanel p = new JPanel(new GridLayout(2, 2));
                p.add(tempTextField); p.add(new JLabel("°C"));
                p.add(fahrenheitLabel); p.add(new JLabel("°F"));

                setLayout(new BorderLayout());
                add(p, BorderLayout.CENTER);
                add(convertButton, BorderLayout.SOUTH);

                setDefaultCloseOperation(
                        WindowConstants.EXIT_ON_CLOSE);
                setTitle("Celsius Converter");
                setSize(200, 100);
                setVisible(true);
        }
}
```



Basic functions

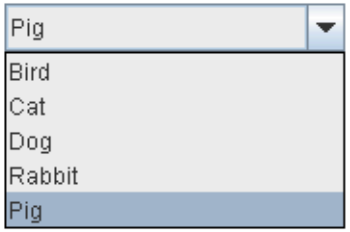
- setDefaultCloseOperation(msg)
  - EXIT_ON_CLOSE
  - DO_NOTHING_ON_CLOSE
  - DISPOSE_ON_CLOSE
  - HIDE_ON_CLOSE
- setSize(int base, int height)
  - defines the dimensions of the panel outside
- setBounds (int xSupSin, int ySupSin, int base, int height)
  - specifies the position in which it is initially the panel
- primarycont.setcontentpane (secondarycont)
  - insert the secondary container in primary

ING

Components, a visual guide

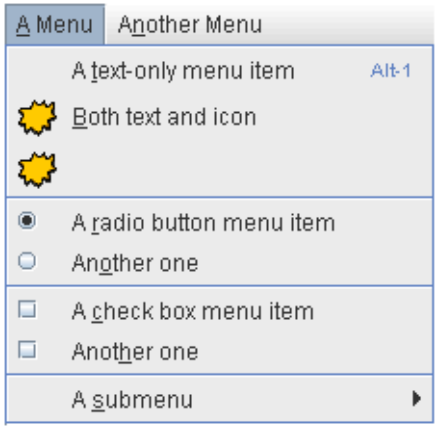
JButton

JCheckBox



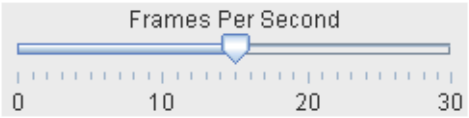
JComboBox

JList



JMenu

JRadioButton



JSlider

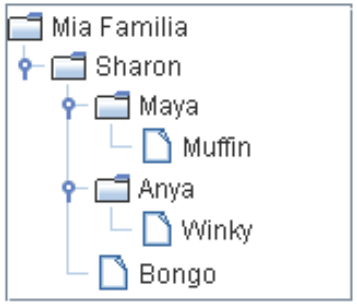
Components, a visual guide

Host	User	Password	Last Modified
Biocca Games	Freddy	!#asf6Awwzb	Mar 16, 2006
zabble	ichabod	Tazb!34$fZ	Mar 6, 2006
Sun Developer	fraz@hotmail.co...	AasW541!fbZ	Feb 22, 2006
Heirloom Seeds	shams@gmail....	bkz[ADF78!	Jul 29, 2005
Pacific Zoo Shop	seal@hotmail.c...	vbAf124%z	Feb 22, 2006

JTable

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.

JTextArea



JTree

Date: 07/2006

JSpinner

City: Santa Rosa

JTextField

Enter the password: •••••••

JPasswordField

Components, a visual guide

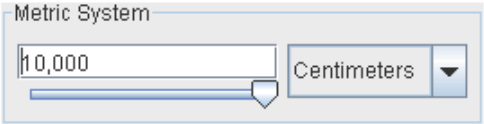
JLabel

JProgressBar

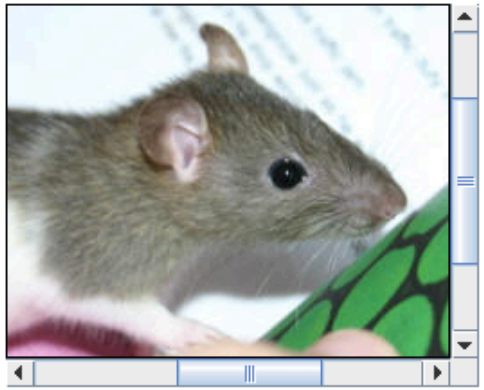
JSeparator

JToolTip

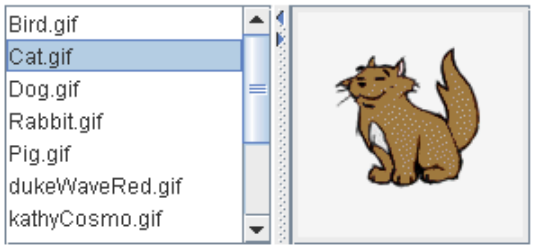
Components, a visual guide



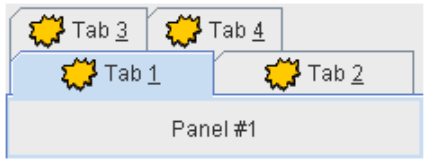
JPanel



JScrollPane



JSplitPane



JTabbedPane

JToolBar

JDialog

- They're used to receive input, provide information, advise the user, etc.
  - Every dialog is dependent on a top-level container.
  - A swing JDialog class inherits this behavior from the AWT Dialog class.
  - Dialogs are all instances of JDialog, even though the majority is done using helper classes (e.g., JOptionPane).

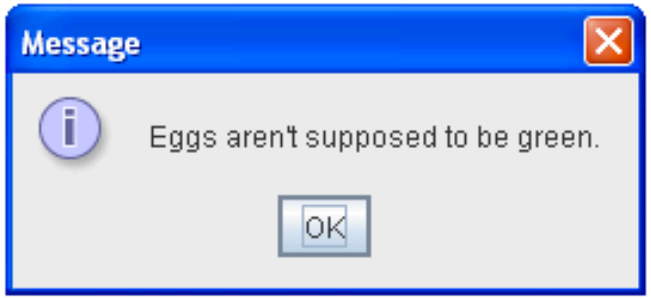
ING

JOptionPane

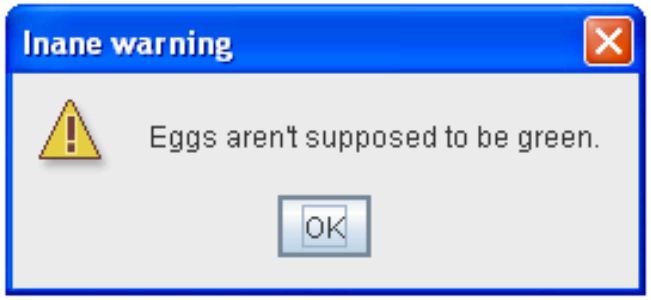
Using JOptionPane, you can quickly create and customize several different kinds of dialogs. JOptionPane provides support for laying out standard dialogs, providing icons, specifying the dialog title and text, and customizing the button text.

Icon description	Java look and feel	Windows look and feel
question		
information		
warning		
error		

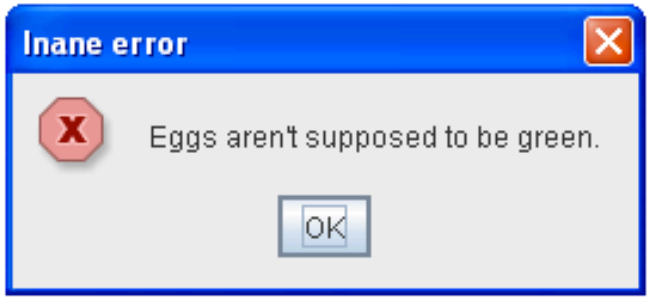
JOptionPane.showMessageDialog()




```
//default title and icon
JOptionPane.showMessageDialog(frame,
    "Eggs are not supposed to be green.");
```




```
//custom title, warning icon
JOptionPane.showMessageDialog(frame,
    "Eggs are not supposed to be green.",
    "Inane warning",
    JOptionPane.WARNING_MESSAGE);
```



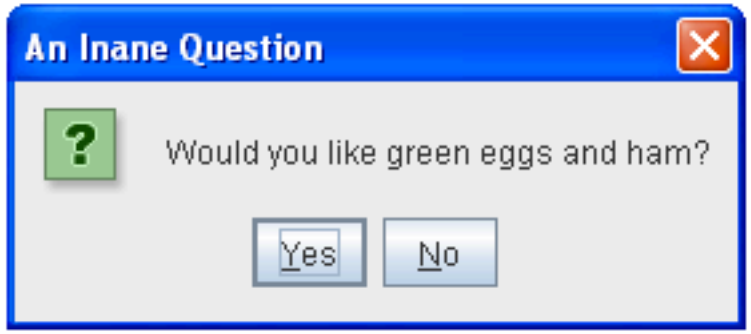

```
//custom title, error icon
JOptionPane.showMessageDialog(frame,
    "Eggs are not supposed to be green.",
    "Inane error",
    JOptionPane.ERROR_MESSAGE);
```

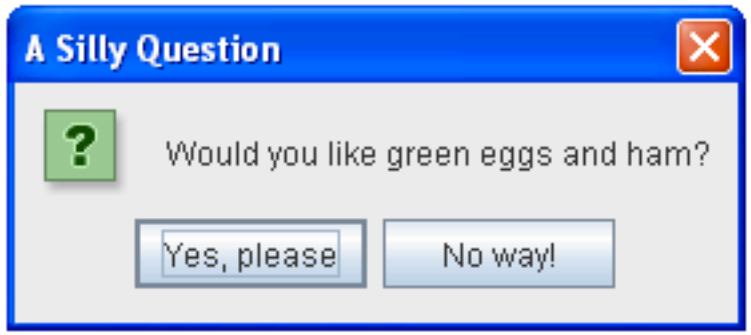
JOptionPane.showOptionDialog()

- Displays a modal dialog with the specified buttons, icons, message, title, and so on. With this method, you can change the text that appears on the buttons of standard dialogs. You can also perform many other kinds
- of customization.


```
//Custom button text
Object[] options = {"Yes, please",
                    "No, thanks",
                    "No eggs, no ham!"};
int n = JOptionPane.showOptionDialog(frame,
    "Would you like some green eggs to go "
    + "with that ham?",
    "A Silly Question",
    JOptionPane.YES_NO_CANCEL_OPTION,
    JOptionPane.QUESTION_MESSAGE,
    null,
    options,
    options[2]);
```

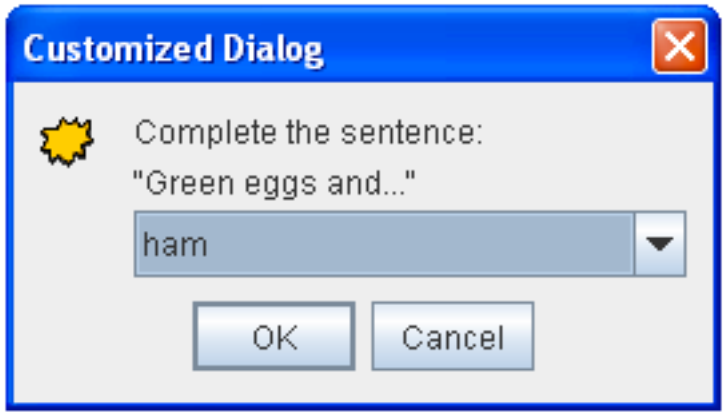
JOptionPane.showConfirmationDialog()






```
//default icon, custom title
int n = JOptionPane.showConfirmDialog(
    frame,
    "Would you like green eggs and ham?",
    "An Inane Question",
    JOptionPane.YES_NO_OPTION);


Object[] options = {"Yes, please",
                    "No way!"};
int n = JOptionPane.showOptionDialog(frame,
    "Would you like green eggs and ham?",
    "A Silly Question",
    JOptionPane.YES_NO_OPTION,
    JOptionPane.QUESTION_MESSAGE,
    null,     //do not use a custom Icon
    options,  //the titles of buttons
    options[0]); //default button title
```

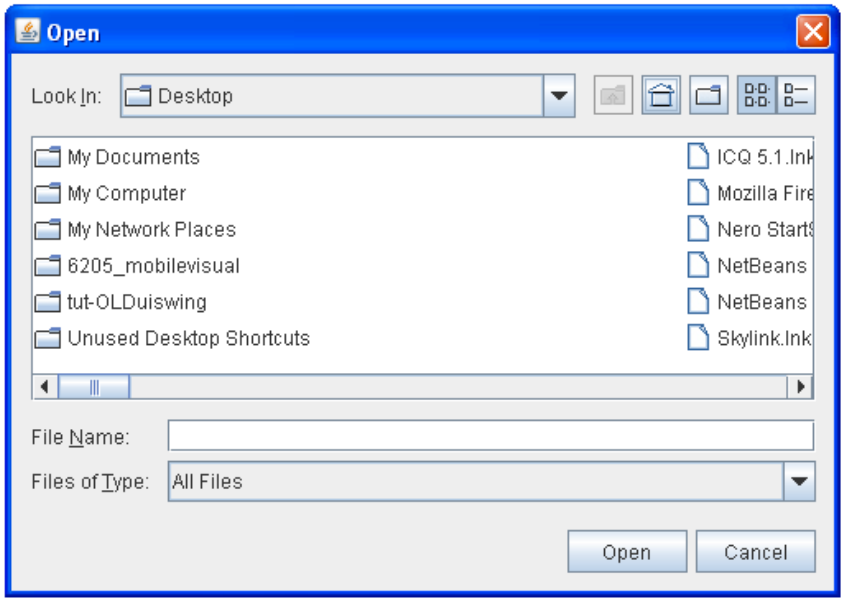
JOptionPane.showInputDialog()




```
Object[] possibilities = {"ham", "spam", "yam"};
String s = (String)JOptionPane.showInputDialog(
                    frame,
                    "Complete the sentence:\n"
                    + "\"Green eggs and...\"",
                    "Customized Dialog",
                    JOptionPane.PLAIN_MESSAGE,
                    icon,
                    possibilities,
                    "ham");

//If a string was returned, say so.
if ((s != null) && (s.length() > 0)) {
    setLabel("Green eggs and... " + s + "!");
    return;
}

//If you're here, the return value was null/empty.
setLabel("Come on, finish the sentence!");
```

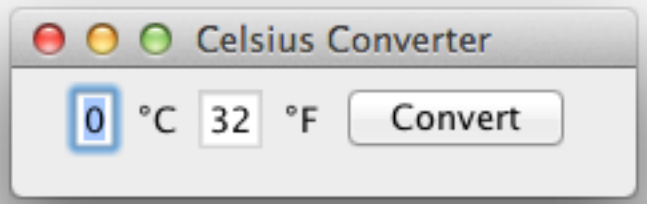
JFileChooser

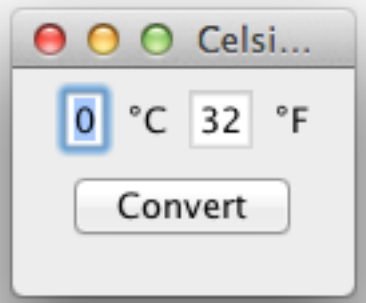
- File choosers provide a GUI for navigating the file system. Could be used as both:
  - static method (modal)
  - instance of JFileChooser



What is a layout?

- All the former examples graphs, when resized, allow the relocation of the components:
  - this behavior is a necessity: Java adapts to many platforms (display in different ways)





Layout Manager

- A layout manager determines the disposal of the components (java.awt.*) on a container
- Panels are containers supporting layouts
  - different panels can have different layouts
- Methodology:

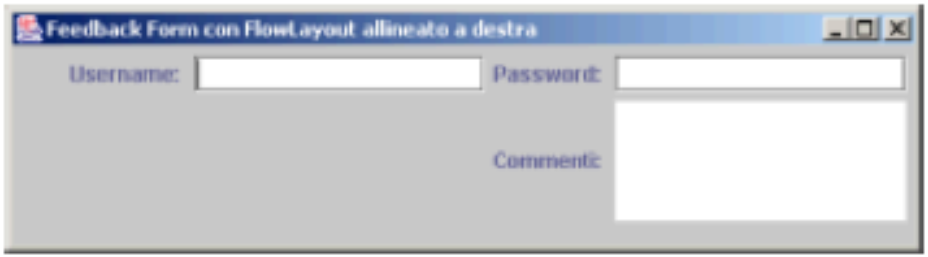
```
JPanel panel = new JPanel(new GridLayout(2,2));
panel.add(JButton); (…)
```

Layout Manager - FlowLayout

- It is the default layout (e.g., new JPanel())
  - Disposes components from left to right, starting from the left most corner in the top
- Constructors:
  - `FlowLayout f = new FlowLayout();`
  - `FlowLayout f = new FlowLayout(int align);`
  - `FlowLayout f = new FlowLayout(int align, int hgap, int vgap);`
- Constructors parameters:
  - align: Alignment of basis (FlowLayout.LEFT, FlowLayout.RIGHT, FlowLayout.CENTER)
  - hgap: Horizontal space between components (default: 3 pixel)
  - vgap: Vertical space between components (default: 3 pixel)

ING

Layout Manager - FlowLayout



(FlowLayout.RIGHT)

(FlowLayout.CENTER,20,20)

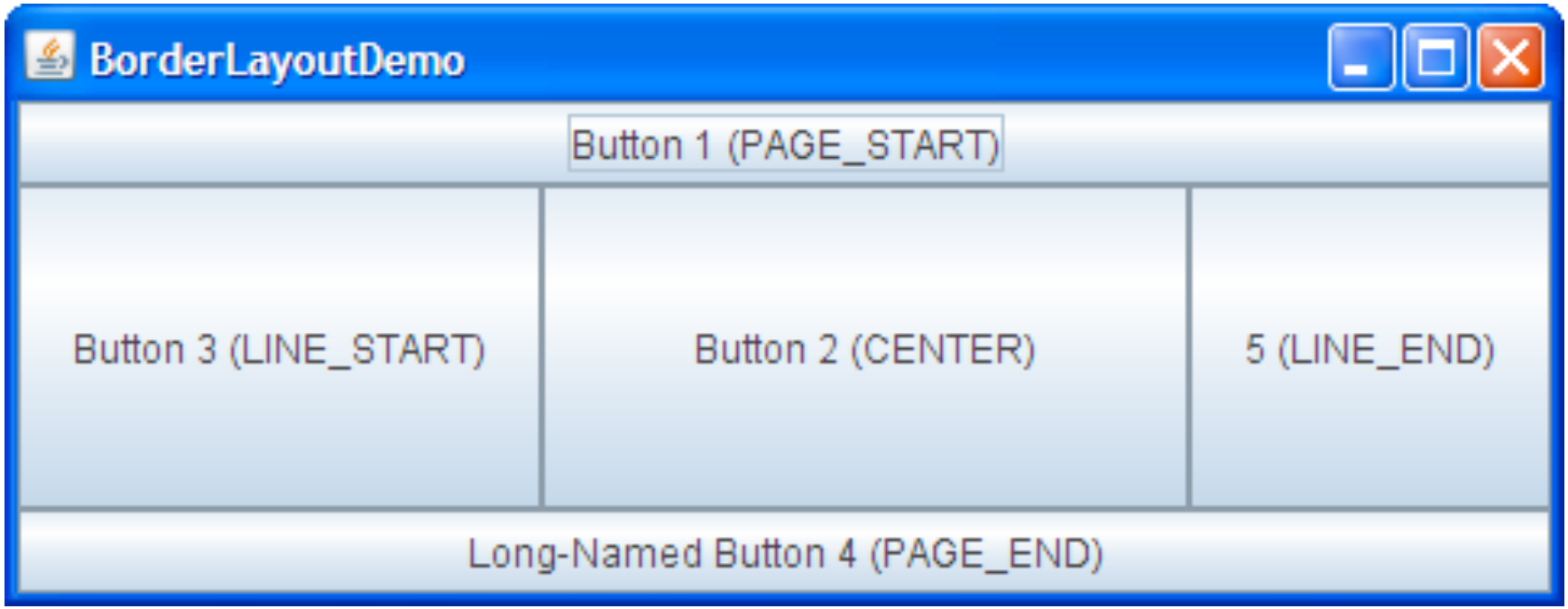
Layout Manager - BorderLayout

- Splits into five areas (“North”, “South”, “East”, “West”, “Center”).
- Constructors:
  - `BorderLayout b = new BorderLayout();`
  - `BorderLayout b = new BorderLayout(int1, int2);`
    - int1, int2 are the spaces between the components related horizontal and vertical
- The filling is “targeted”:

```
JPanel panel = new JPanel(new BorderLayout());
panel.add(BorderLayout.PAGE_START”, b1);
panel.add(BoarderLayout.PAGE_END, b2);
```

ING

Layout Manager - BorderLayout

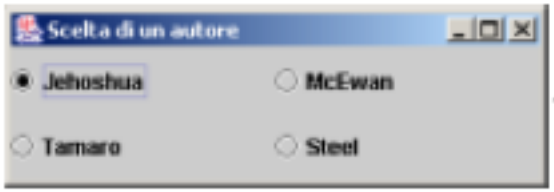


Layout Manager - GridLayout

- Splits the visual area in a grid of rows and columns
  - Starts from the box in the top left
- Constructors:
  - `GridLayout g = new GridLayout(int rows, int cols);`
  - `GridLayout g = new GridLayout(rows, cols, hgap, vgap);`
- Constructors parameters:
  - rows: number of row; cols: number of columns;
  - hgap: Spacing (in pixels) between two horizontal boxes (default: 0 pixel)
  - vgap: spacing (in pixel) between two vertical boxes (default: 0 pixel)

Layout Manager - GridLayout

4 row, 1 columns:
distance 0



2 row, 2 columns:
distance 0 pixel

(Distanza min = 0)

2 row, 2 columns:
distance 10 pixel

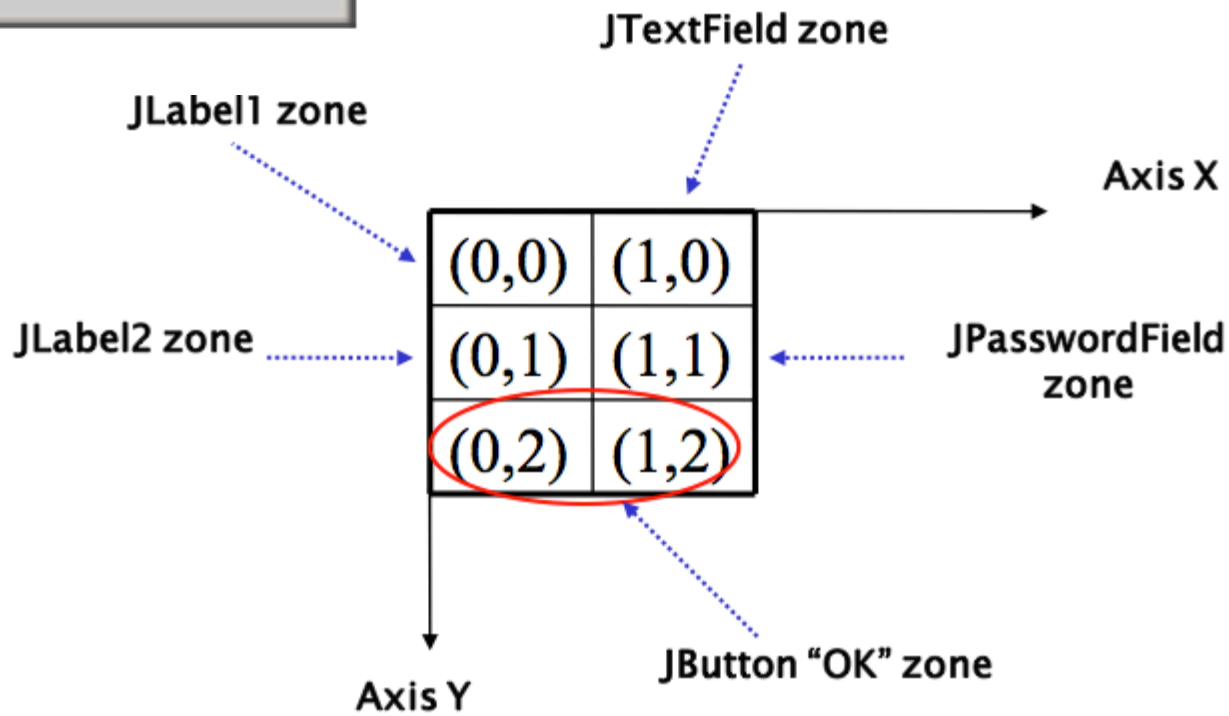
(Min distance = 10)

Layout Manager - GridBagLayout

- Extension of GridLayout. Makes it possible to adjust the elements of the grid
- Methodology:

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

//For each component to be added to this container:
//...Create the component...
//...Set instance variables in the GridBagConstraints
instance...
pane.add(theComponent, c);
```

Layout Manager - GridBagLayout



Layout Manager - CardLayout

- CardLayout allows to have different panels in the frame, but only one showed at time
  - the panels are called cards
- Methodology:

```
JPanel p = new JPanel(new CardLayout());
p.add("Panel1", new JPanel());
p.add("Panel2", new JPanel());
```