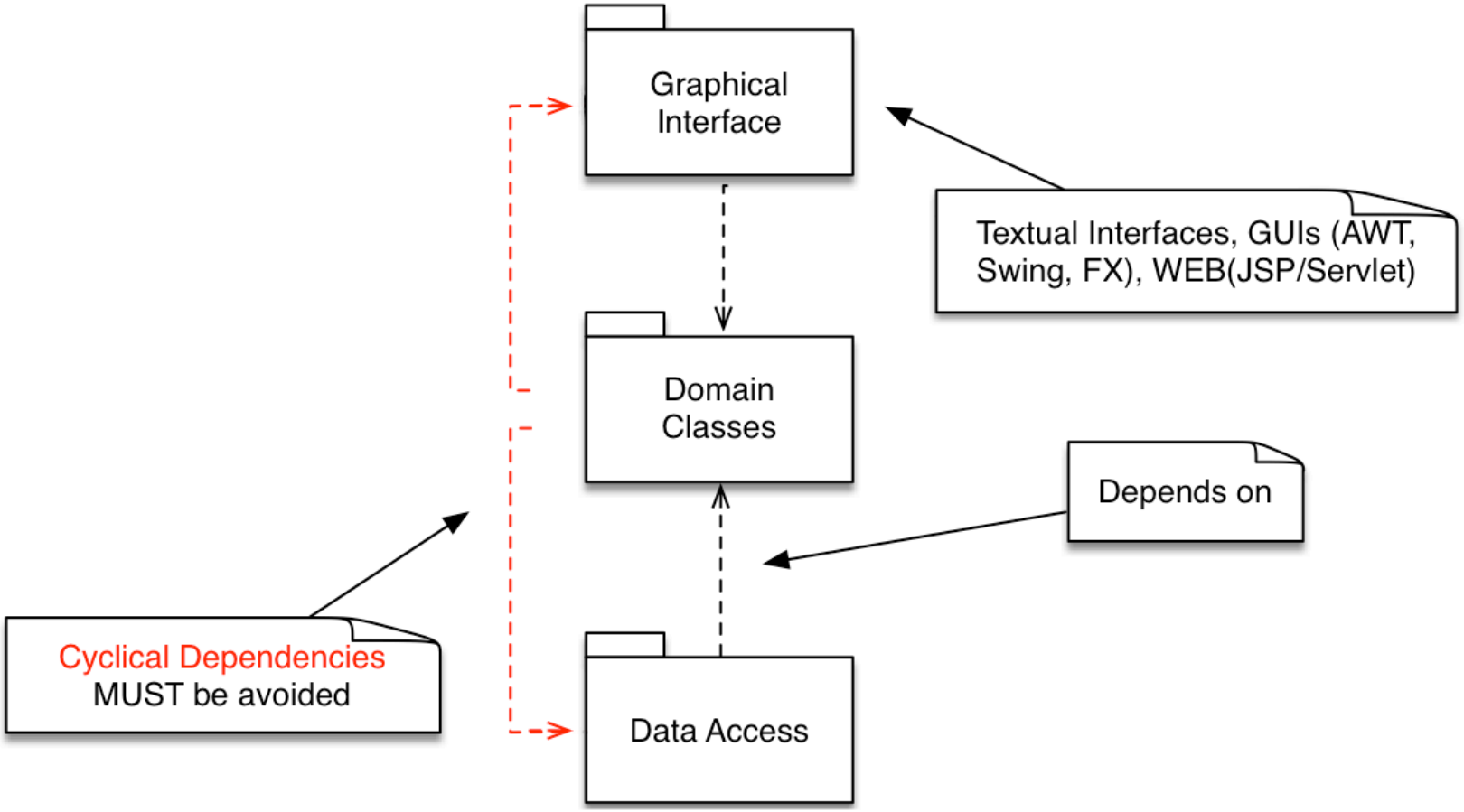
Java Swing

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

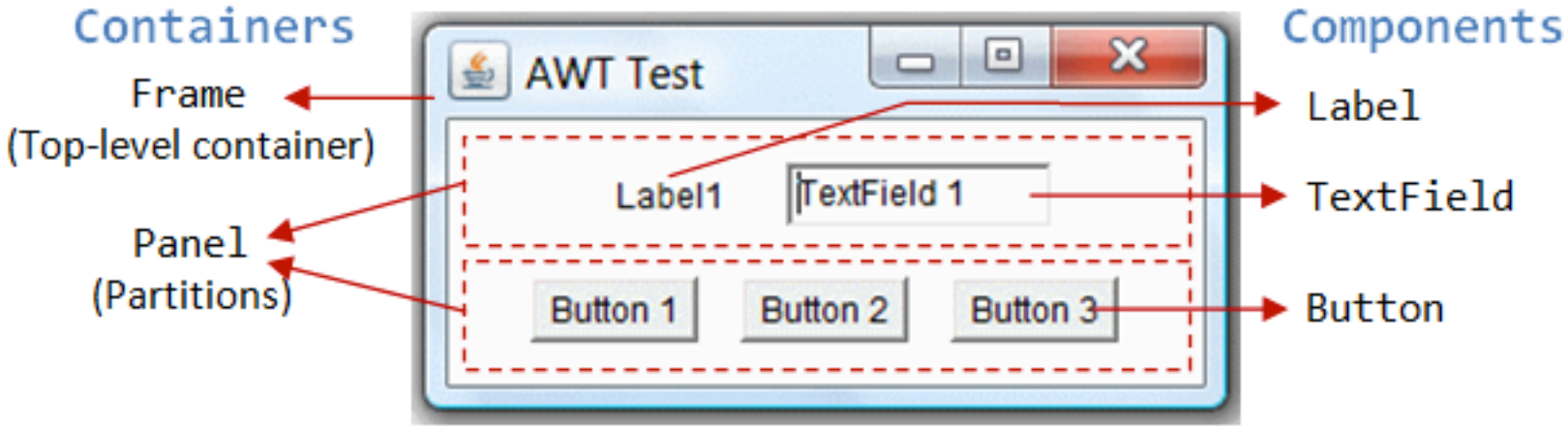
ING

Software Design



ING

Containers and components



ING

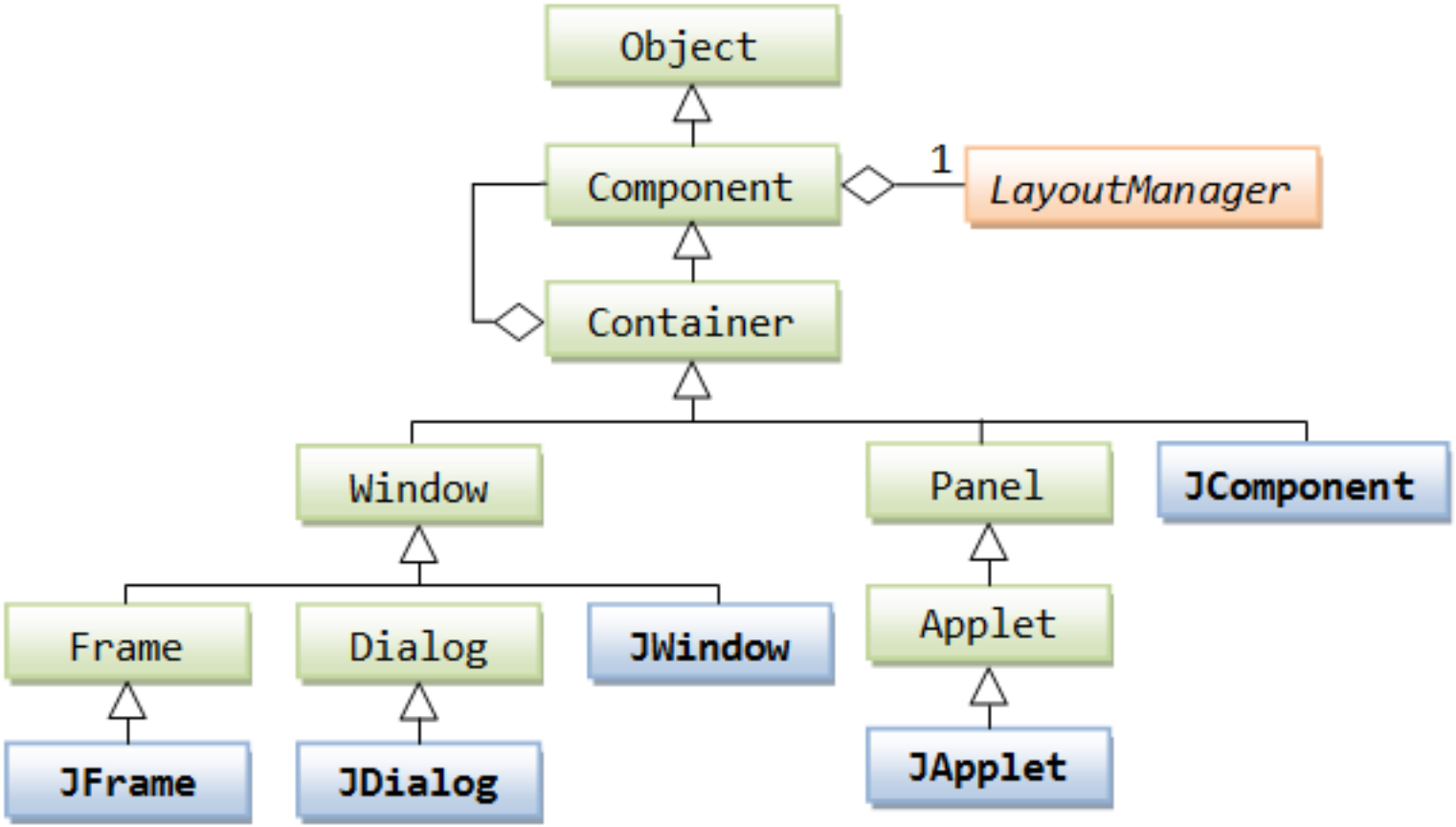
Package java.awt.*

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

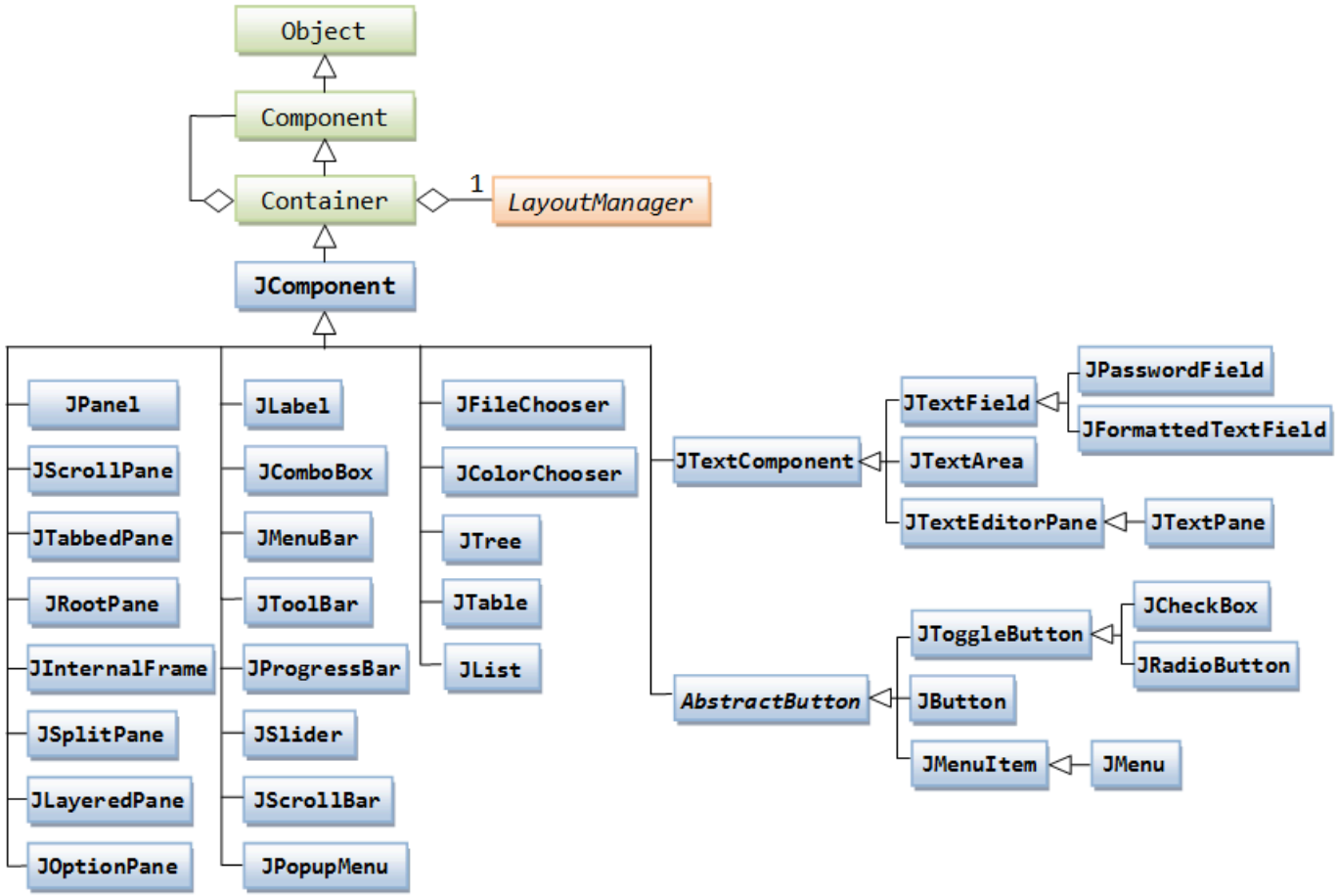
Package javax.swing.*

- 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

Class hierarchy



Class hierarchy

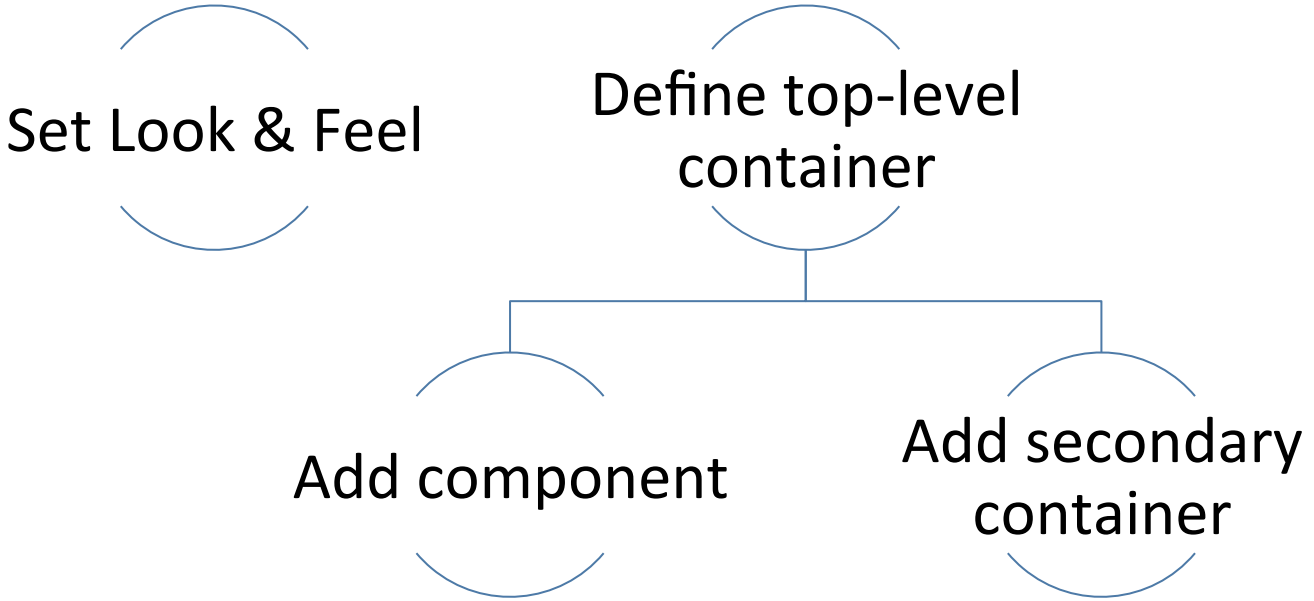


ING

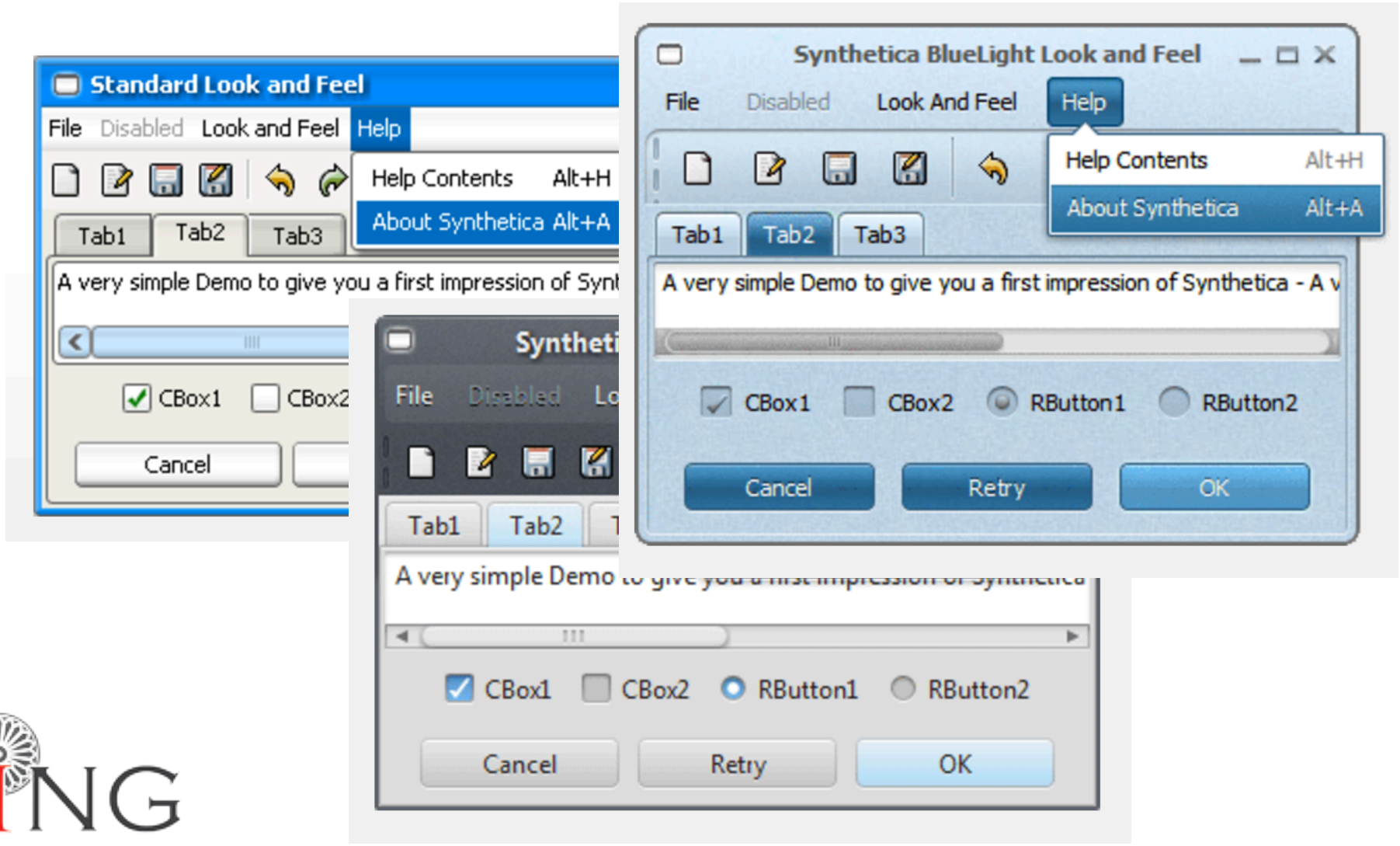
Graphical Programming

- 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

Graphical Programming



Look & Feel


Standard Look and Feel
File Disabled Look and Feel Help
Help Contents Alt+H
About Synthetica Alt+A
Tab1 Tab2 Tab3
A very simple Demo to give you a first impression of Synt
CBox1 CBox2
Cancel
Synthetica BlueLight Look and Feel
File Disabled Look And Feel Help
Help Contents Alt+H
About Synthetica Alt+A
Tab1 Tab2 Tab3
A very simple Demo to give you a first impression of Synthetica - A v
CBox1 CBox2 RButton1 RButton2
Cancel Retry OK
Synthetica
File Disabled Lo
Tab1 Tab2
CBox1 CBox2 RButton1 RButton2
Cancel Retry OK

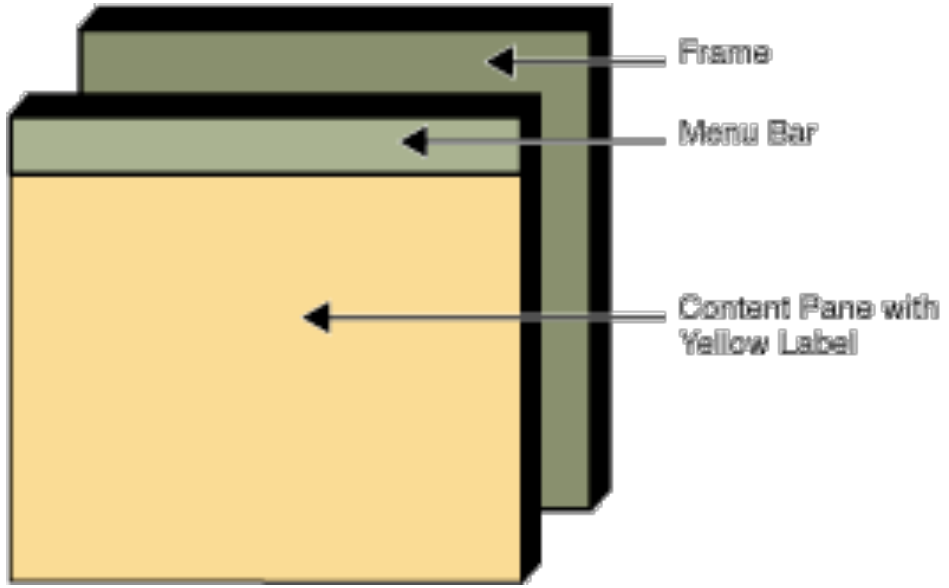
Look & Feel

UIManager manages the current look and feel!

*http://www.jyloo.com/synthetica/themes/

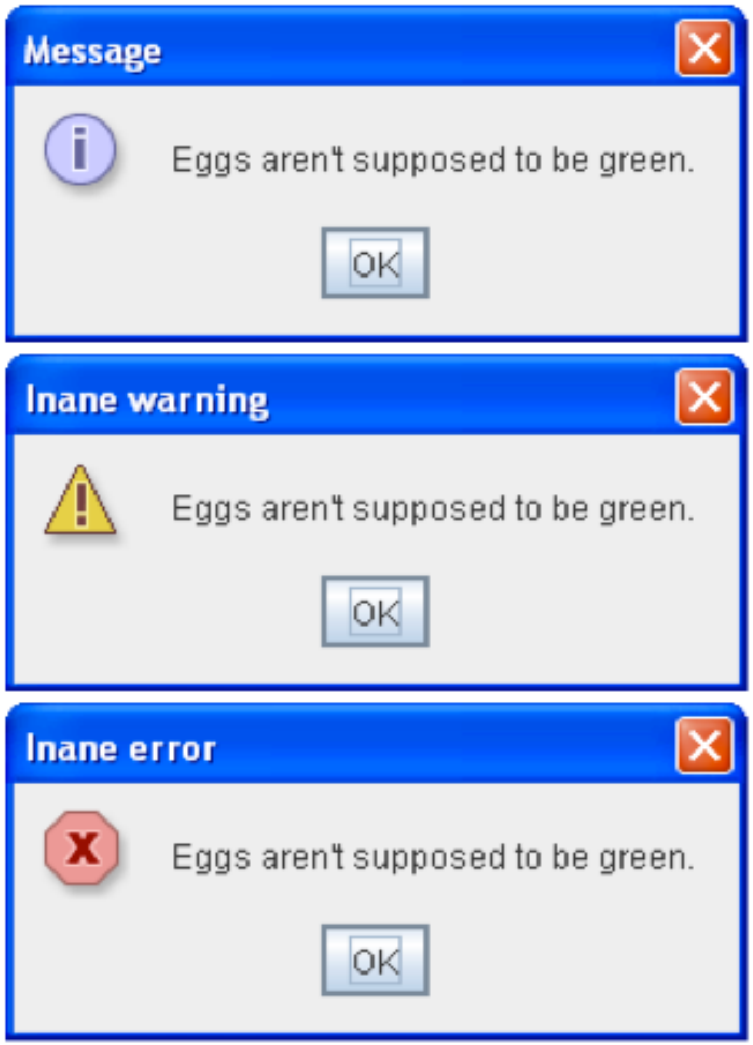
```
// Set Metal L&F
UIManager.setLookAndFeel("javax.swing.plaf.metal.MetalLookAndFeel");

// 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");
```

Top-level container: JFrame



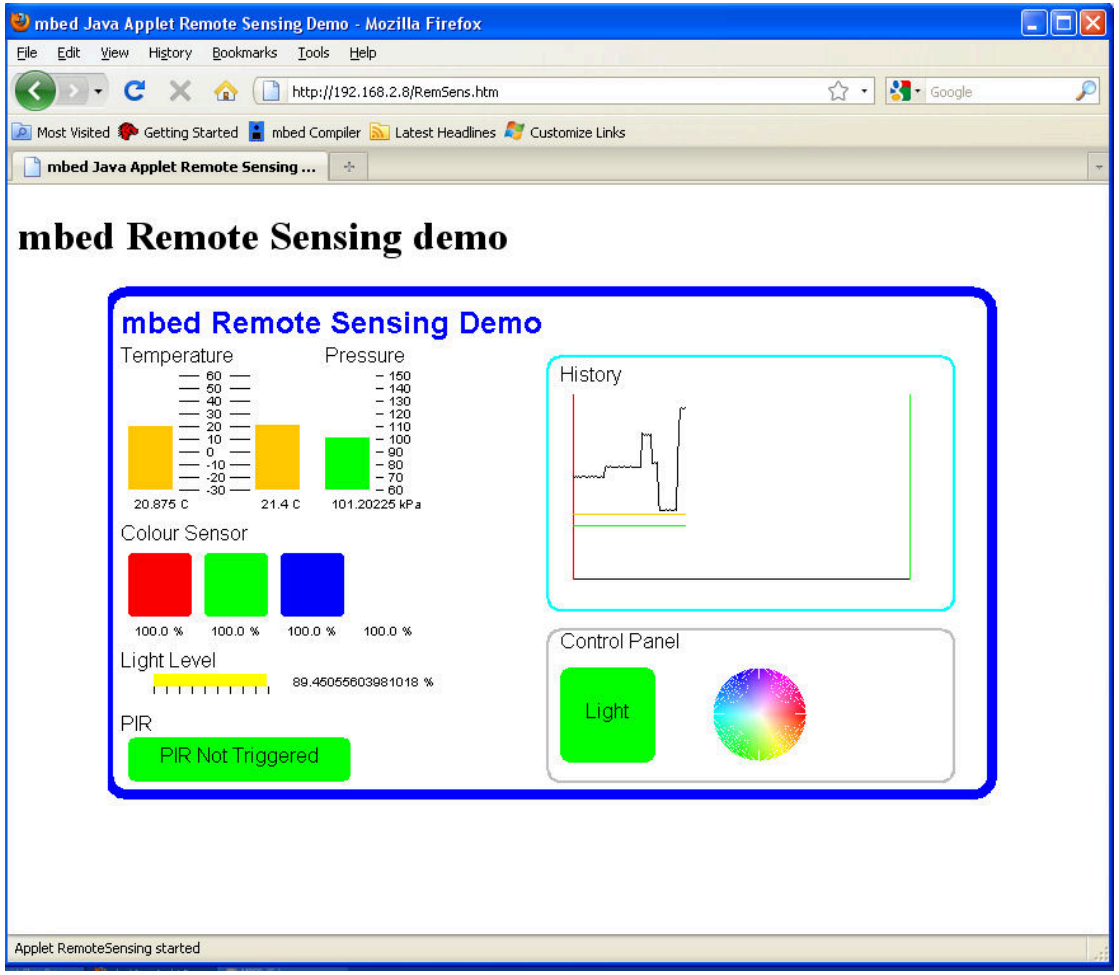
ING

Top-level container: JDialog



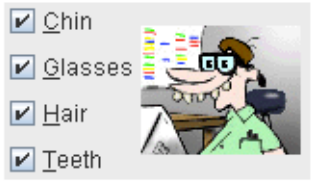
ING

Top-level container: JApplet *(deprecated)*

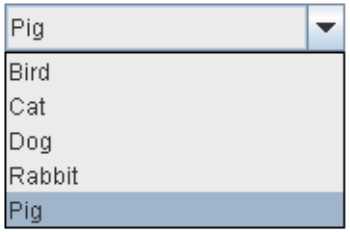


Components, a visual guide

JButton

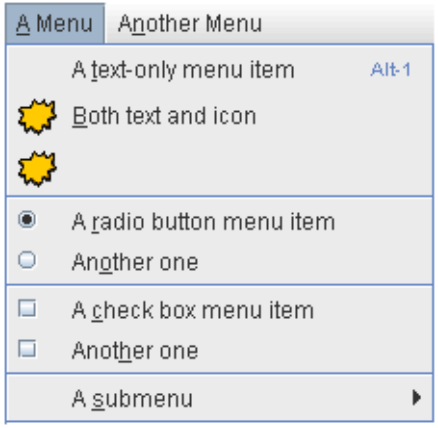


JCheckBox



JComboBox

JList



JMenu

JRadioButton

JSlider

http://web.mit.edu/6.005/www/sp14/psets/ps4/java-6-tutorial/components.html

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

JSpinner

JTextField

JPasswordField

http://web.mit.edu/6.005/www/sp14/psets/ps4/java-6-tutorial/components.html

Components, a visual guide

JLabel

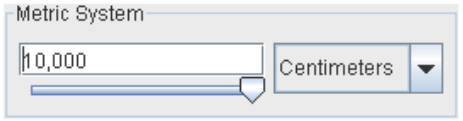
JProgressBar

JSeparator

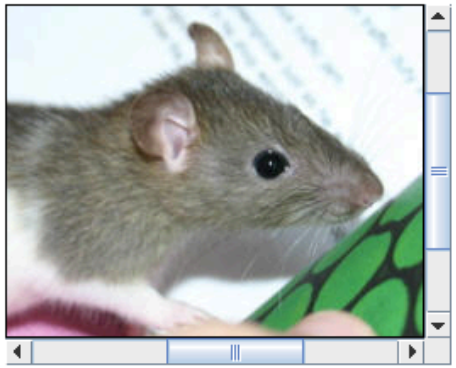
JToolTip

http://web.mit.edu/6.005/www/sp14/psets/ps4/java-6-tutorial/components.html

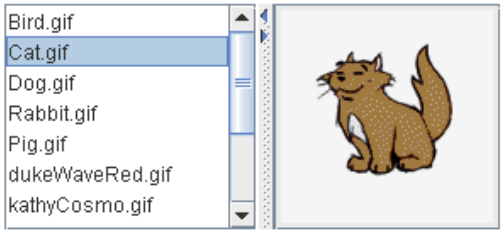
Components, a visual guide



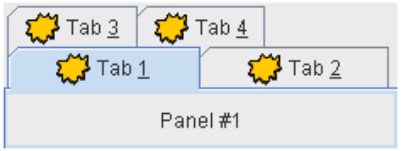
JPanel



JScrollPane



JSplitPane



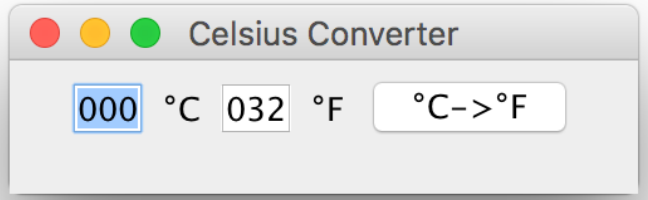
JTabbedPane

JToolBar

http://web.mit.edu/6.005/www/sp14/psets/ps4/java-6-tutorial/components.html

A complete example

```
public class CelsiusConverterBasic extends JFrame {
      private static final long serialVersionUID = 1L;
      private JButton CFButton;
      private JTextField fahrenheitTF, celsiusTF;

      public CelsiusConverterBasic() {
            super("Celsius Converter");
            celsiusTF = new JTextField("000");
            fahrenheitTF = new JTextField("032");
            CFButton = new JButton("°C->°F");

            JPanel p1 = new JPanel();
            p1.add(celsiusTF);
            p1.add(new JLabel("°C"));
            p1.add(fahrenheitTF);
            p1.add(new JLabel("°F"));
            p1.add(CFButton);

            add(p1);
            setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
            setSize(250, 75);
            setVisible(true);
      }
}
```



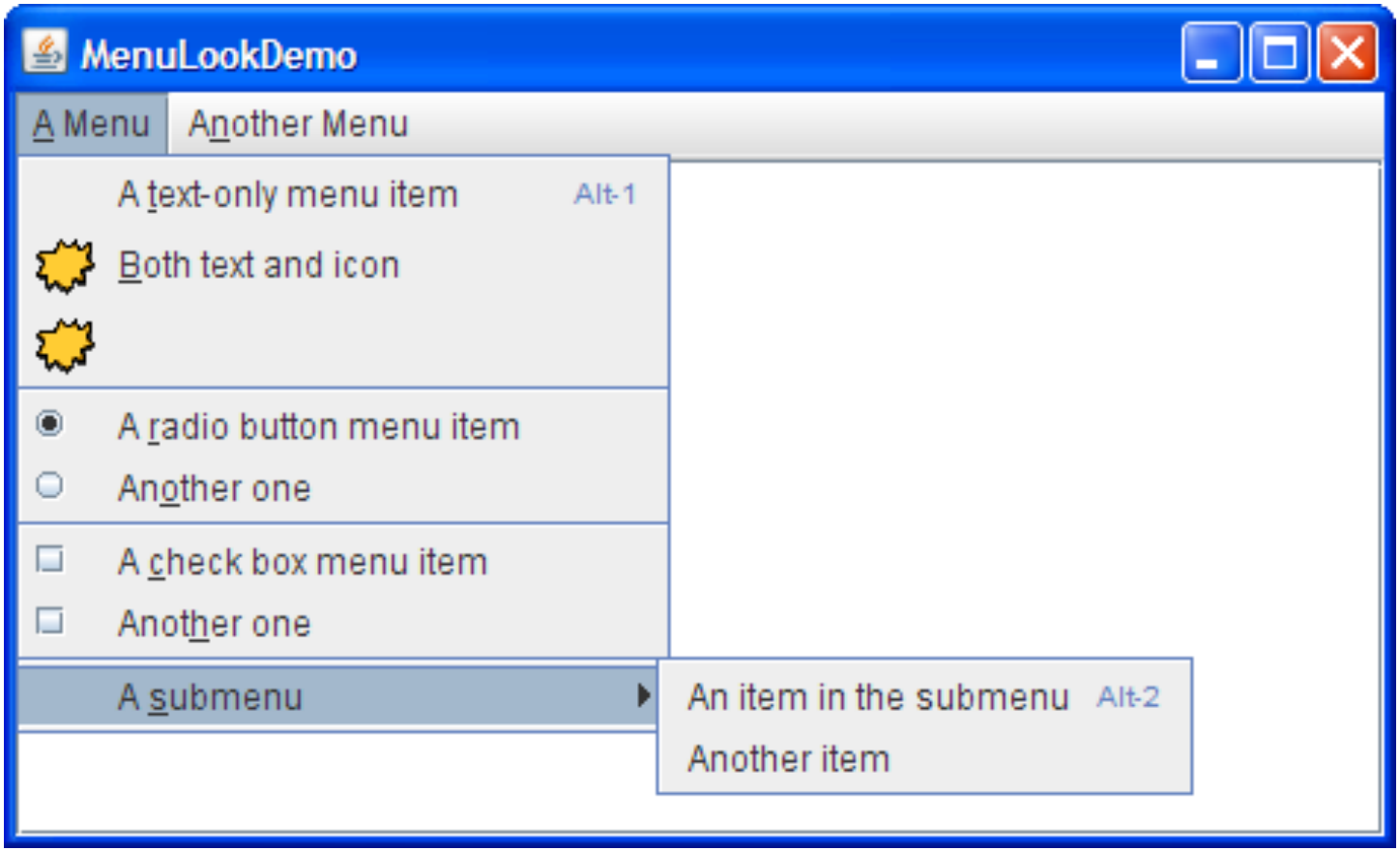
JFrame basic functions

- setDefaultCloseOperation(WindowConstants)
  - EXIT_ON_CLOSE
  - DO_NOTHING_ON_CLOSE
  - DISPOSE_ON_CLOSE
  - HIDE_ON_CLOSE
- setSize(int base, int height)
  - defines the dimensions of the component
- setVisible(boolean visibility)
  - defines the visibility status of the component
- primarycontainer.setcontentpane(secondarycontainer)
  - insert the secondary container in primary

Running it!

```
// Ok
public static void main(String[] args) {
    new CelsiusConverter();
}

// Better
public static void main(String[] args) {
    EventQueue.invokeLater(new Runnable() {
        @Override
        public void run() {
            new CelsiusConverter();
        }
    });
}
```

Swing Menus



ING

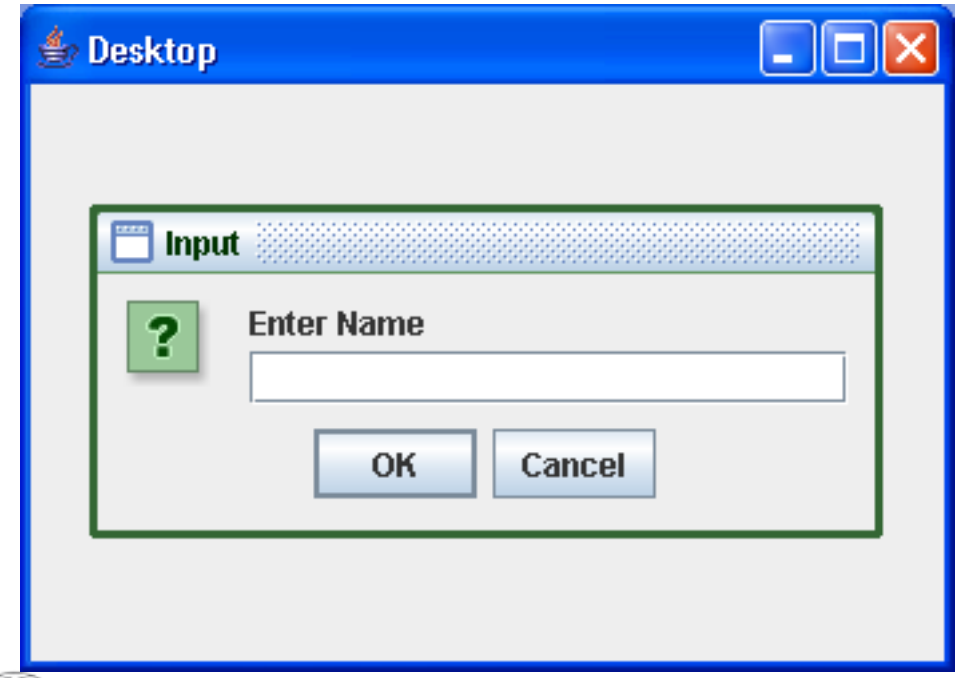
Swing Menus

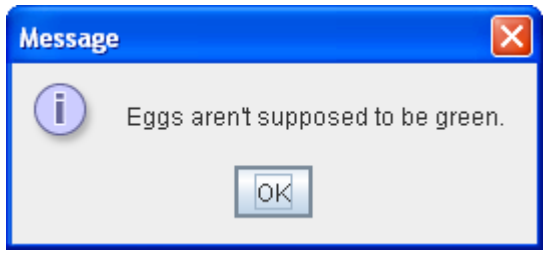
- Three components are involved in a hierarchical fashion:
  - JMenuBar, JMenu, JMenuItem

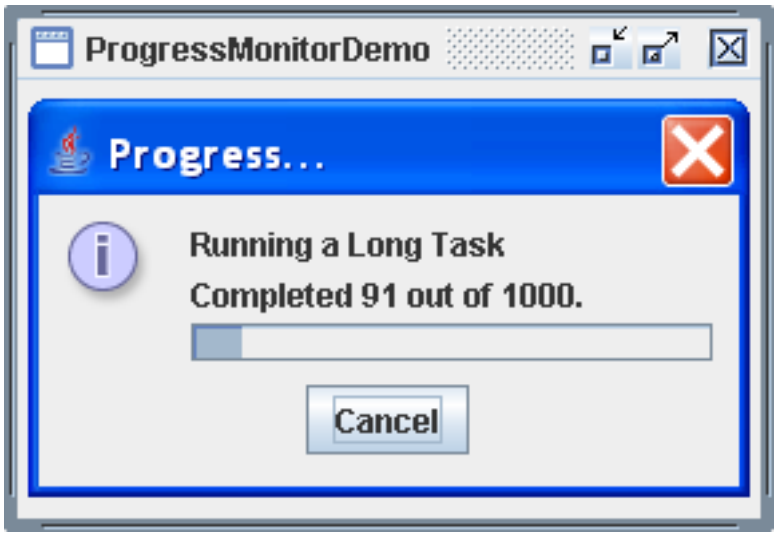
```
JMenuItem openFile = new JMenuItem("Open");
JMenuItem closeFile = new JMenuItem("Close");

JMenu file = new JMenu("File");
file.add(openFile);
file.add(closeFile);

JMenuBar menuBar = new JMenuBar();
menuBar.add(file)

setJMenuBar(menuBar);
```

JDialog

- Applications need to provide information, advise the user, etc.







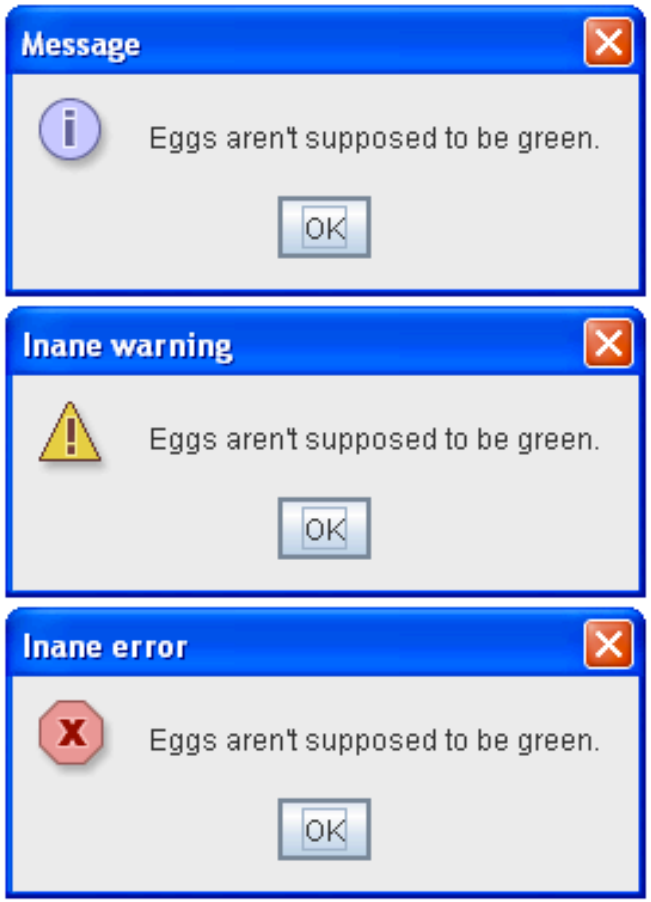
JDialog

- Dialogs are a better choice than instantiating other JFrames!
  - Every dialog is dependent on a top-level container.
  - Dialogs are all instances of JDialog, even though the majority is done using helper classes (e.g., JOptionPane).

How to make Dialogs

- Subclassing JDialog (top-level container) and defining your own layouts.
- 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.

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);
```

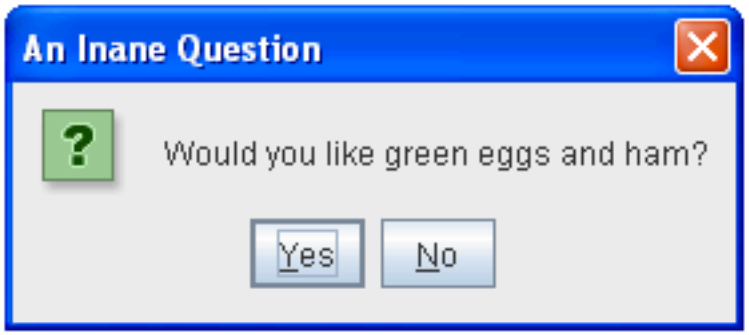
```
//custom title, error icon
JOptionPane.showMessageDialog(frame,
    "Eggs are not supposed to be green.",
    "Inane error",
    JOptionPane.ERROR_MESSAGE);
```

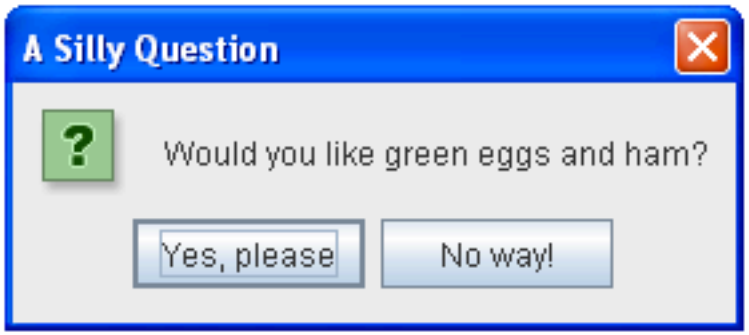
JOptionPane.showOptionDialog()

- 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()



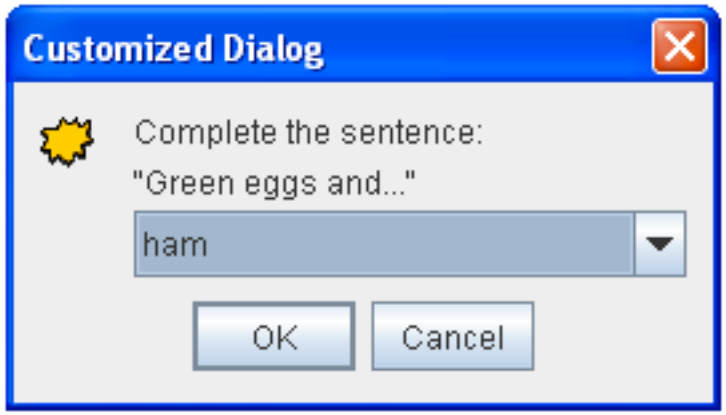



```
//default icon, custom title
int n = JOptionPane.showConfirmDialog(
    frame,
    "Would you like green eggs and ham?",
    "An Inane Question",
    JOptionPane.YES_NO_OPTION);
```

```
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
```

ING

JOptionPane.showInputDialog()

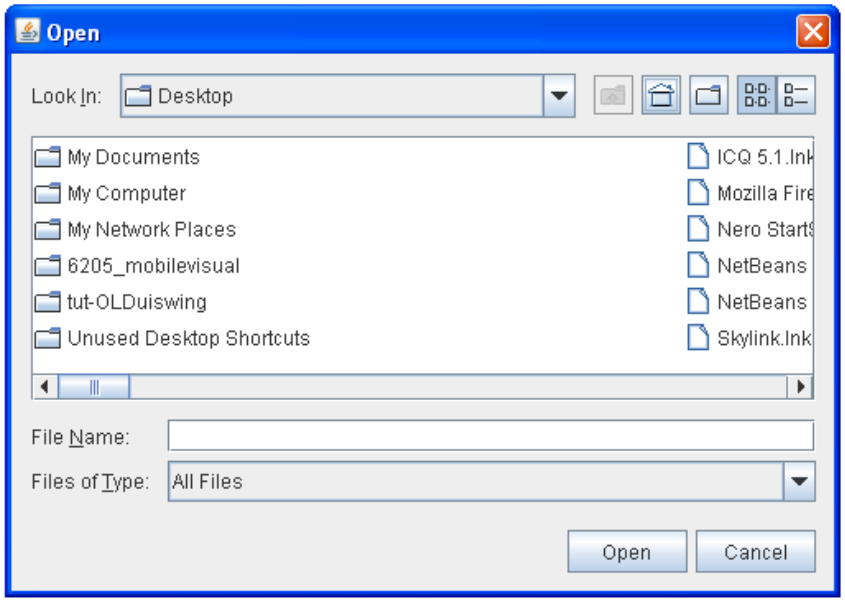



```
Object[] possibilities = {"ham", "spam", "yam"};
String s = (String)JOptionPane.showInputDialog(
                    frame,
                    "Complete the sentence:\n"
                    + "\"Green eggs and...\"",
                    "Customized Dialog",
                    JOptionPane.PLAIN_MESSAGE,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

ING

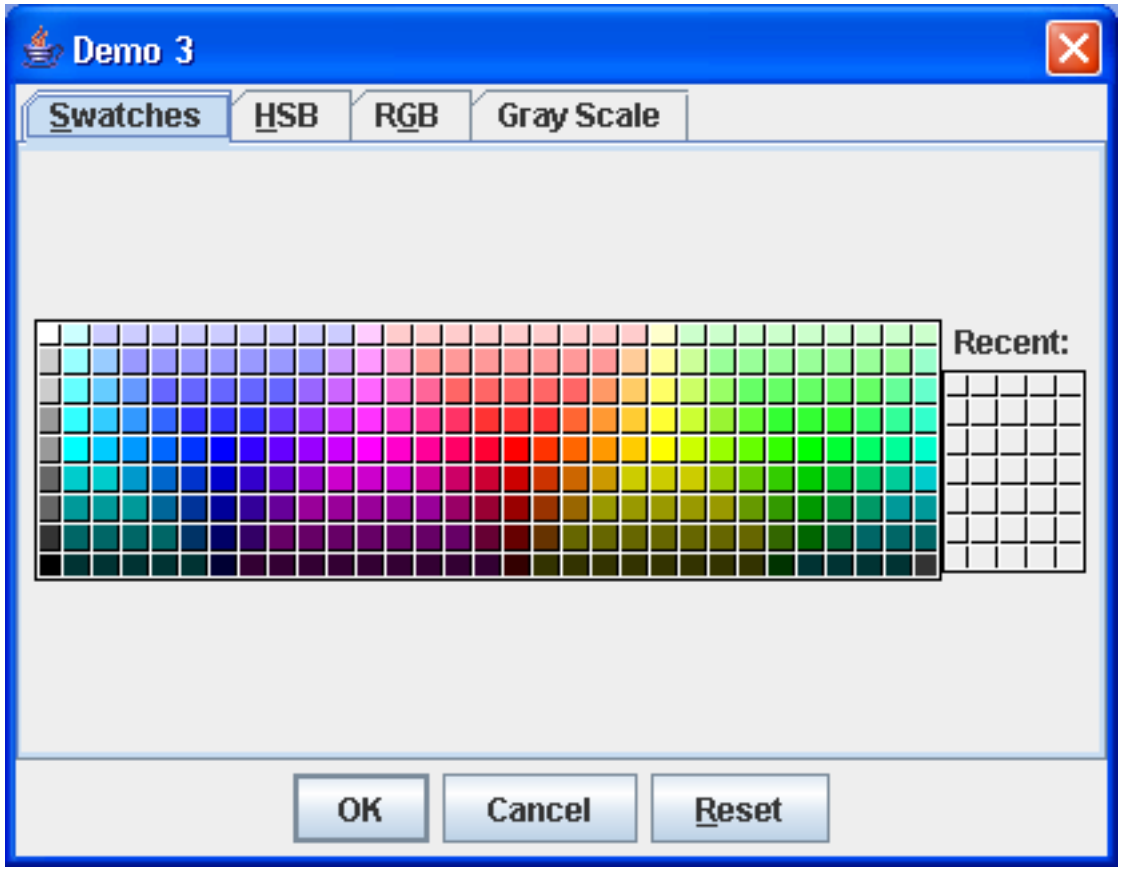
JFileChooser

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



JColorChooser

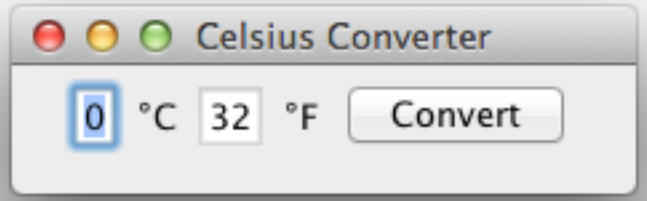
- Provides a GUI for navigating color spaces.

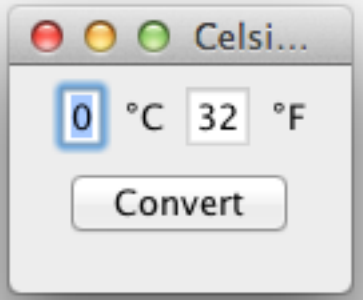


ING

What is a layout?

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





ING

Layout Manager

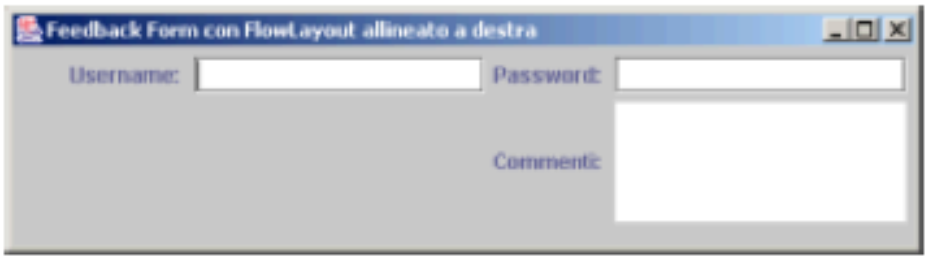
- A layout manager determines the disposal of the components in a container
  - *Flow, Border, Grid, GridBag, Card* Layouts
- Panels are containers supporting layouts
  - Different panels can have different layouts
  - Layouts are passed to constructors
- Methodology:

```
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)

Layout Manager - FlowLayout



(FlowLayout.RIGHT)

(FlowLayout.CENTER,20,20)

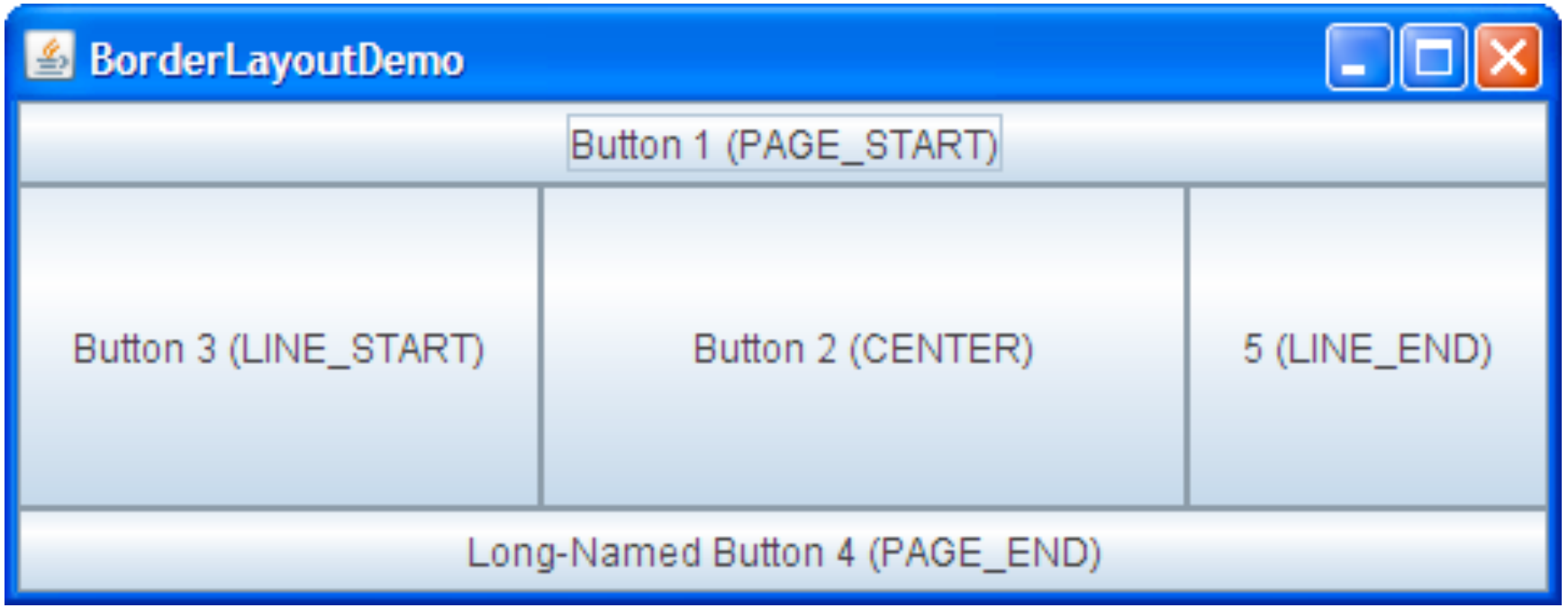
ING

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);
```

Layout Manager - BorderLayout

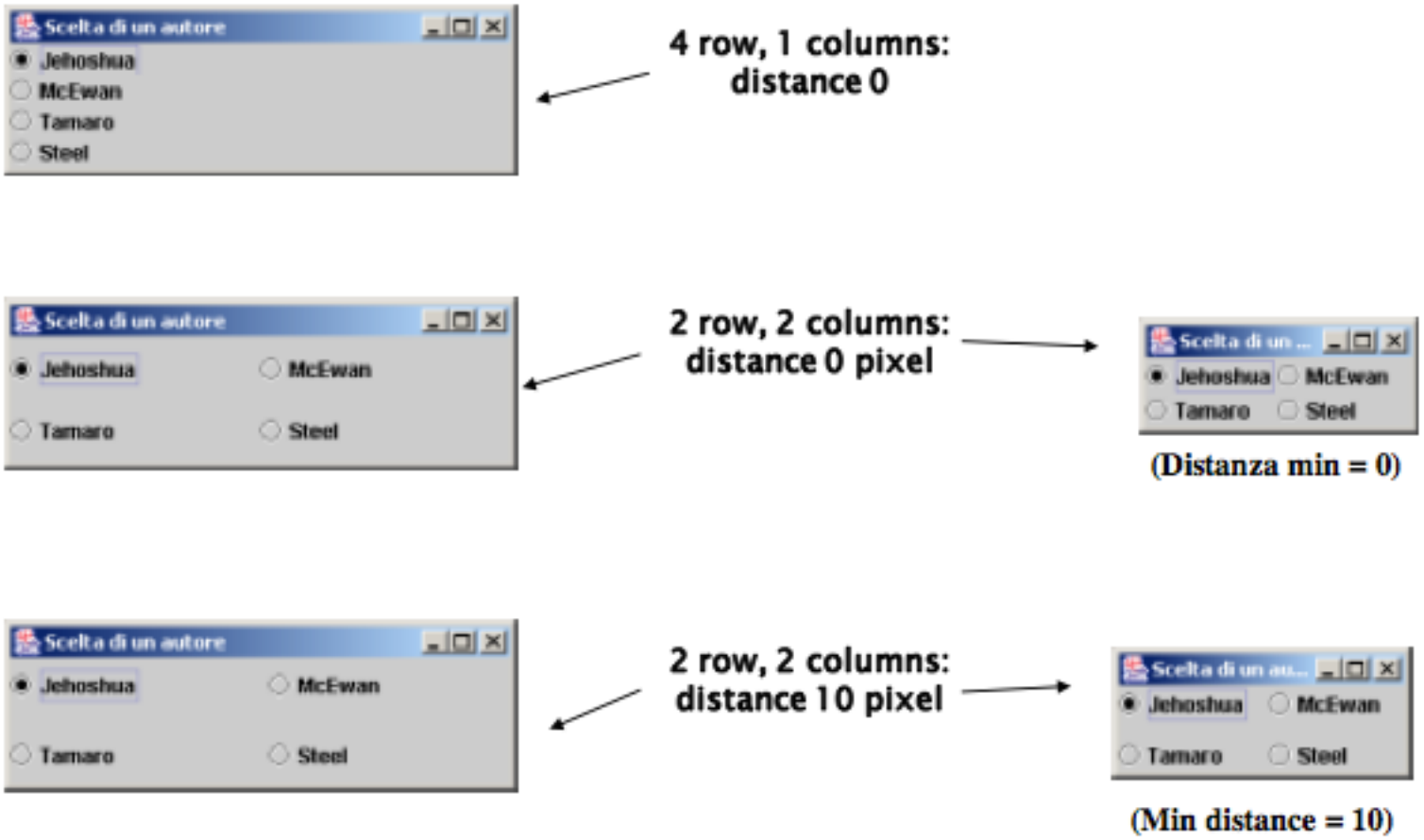


ING

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

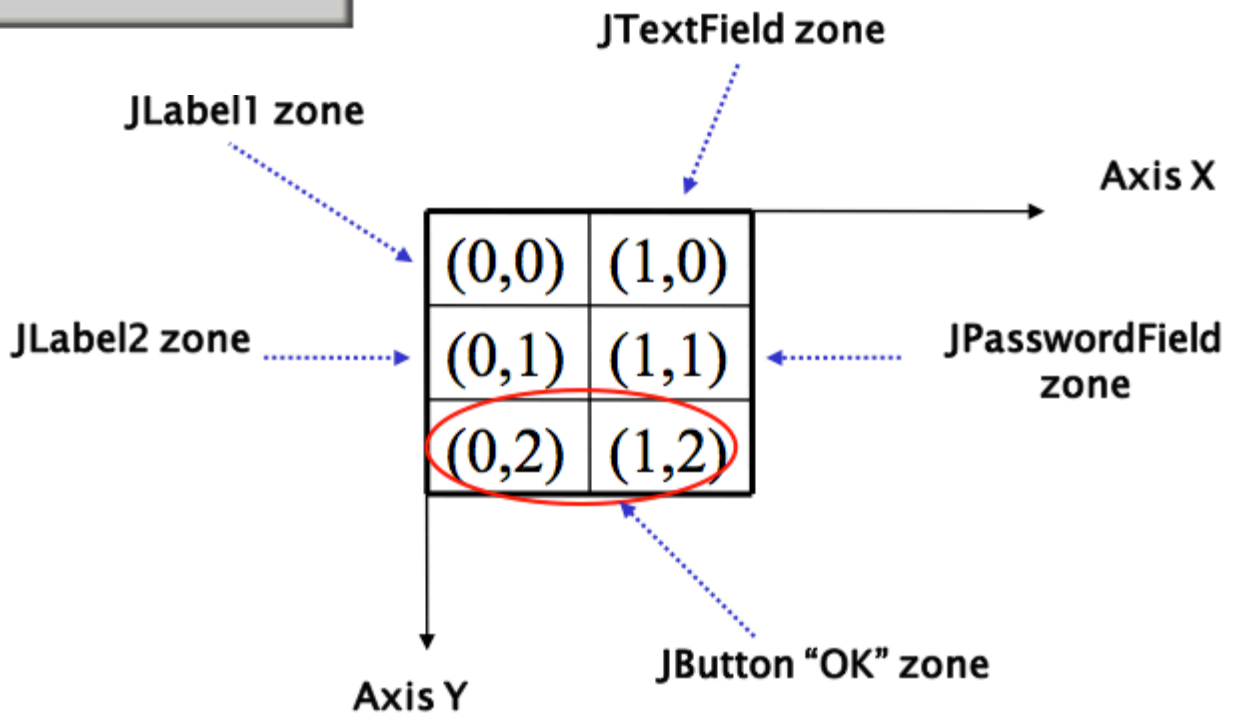


Layout Manager - GridBagLayout

- 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());
```