GUI

Graphical User Interfaces

GUI basic

- Aim: produce graphical output and make interaction with it possible
- Means: AWT classes and Swing classes
  - using and adapting by inheritance

GUI classes

- hierarchical structure
- everything you see on screen (windows, buttons, text fields) inherits from JComponent
- a JContainer may contain other components
  - JFrame (window) and JPanel (window section)
- Helper classes (Color, Font,...)

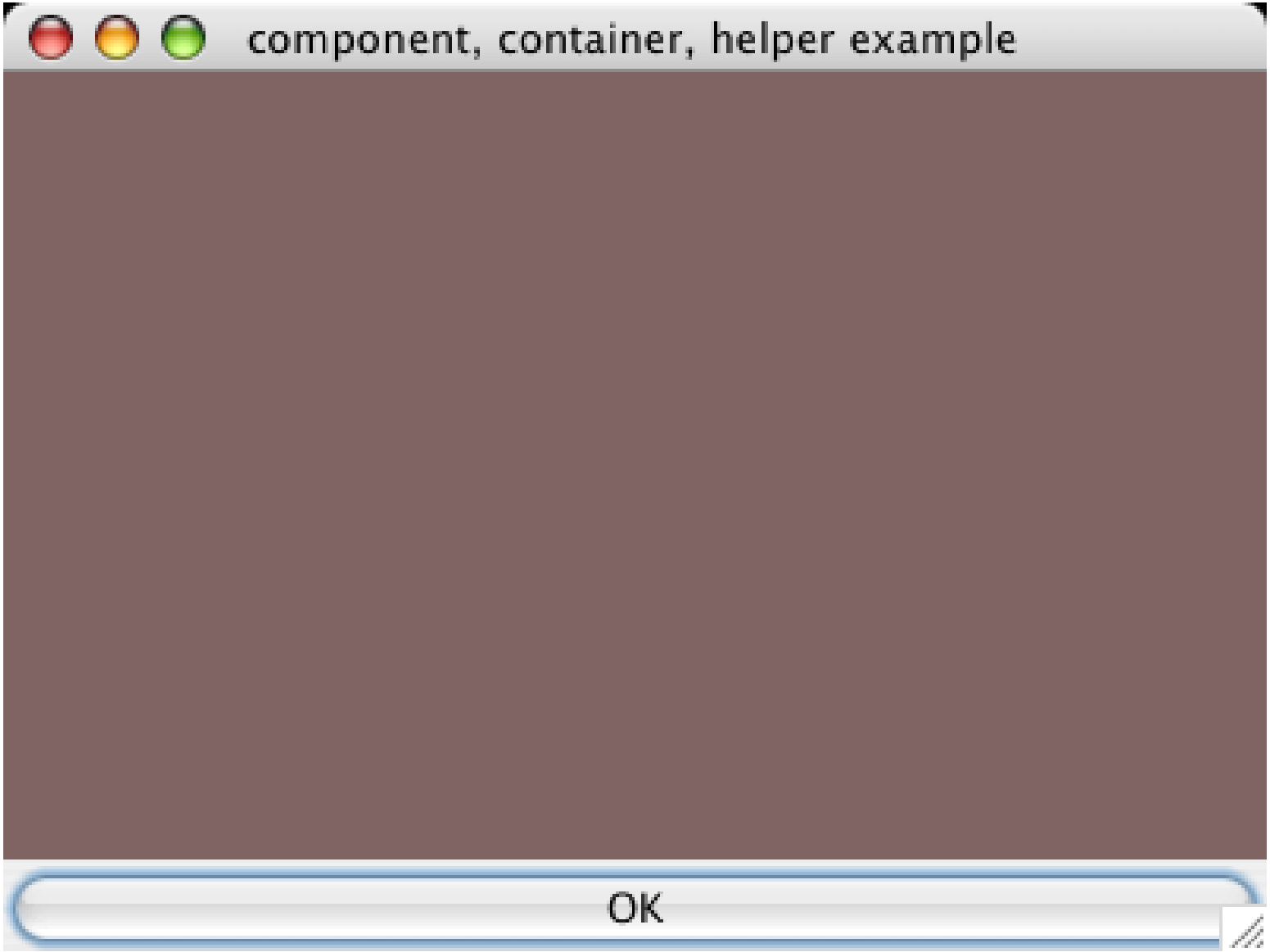
```
import java.awt.*;
import javax.swing.*;

public class MyFrame {
   void demo() {
      JFrame frame = new JFrame ("component, container, helper example");
                                                  // creates window
                                                  // not visible yet
      JButton button = new JButton("OK");      // creates button
      frame.add(button, BorderLayout.SOUTH);
                              // put component in frame: button in frame

      JPanel panel = new JPanel();             // creates another component
      frame.add(panel);     // put panel in frame

      Color color = new Color(128, 100, 100); // creates helper object:
                                                  // Color
      panel.setBackground(color);                 // colors background panel
      // standard code for frames
      frame.setSize(400, 300);                    // size of window in pixels
      frame.setVisible(true);                     // make frame visible
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                        // make close button (X) behave as expected
   }
   public static void main(String[] args) {
      new MyFrame().demo();
   }
```

Result



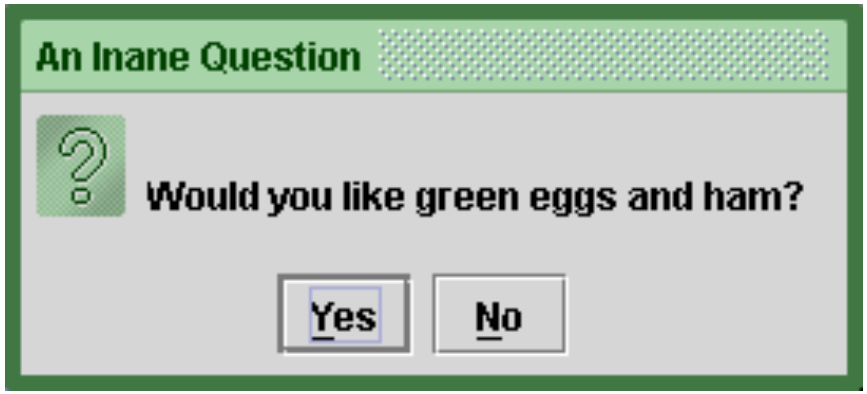
More components

- GUI classes come from several packages:
  - java.awt
  - java.awt.event (see later)
  - javax.swing
  - javax.swing.border
- overview Swing components

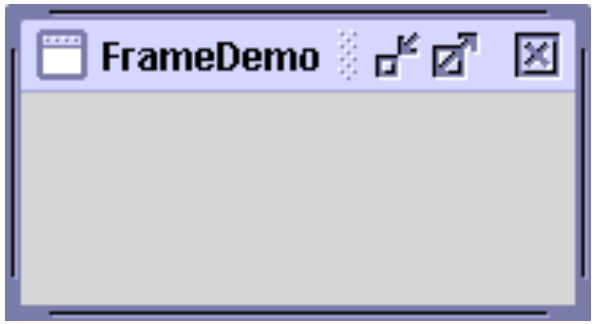
zie **http://java.sun.com/docs/books/tutorial/uiswing/components/components.html**

Top-Level Containers

Applet

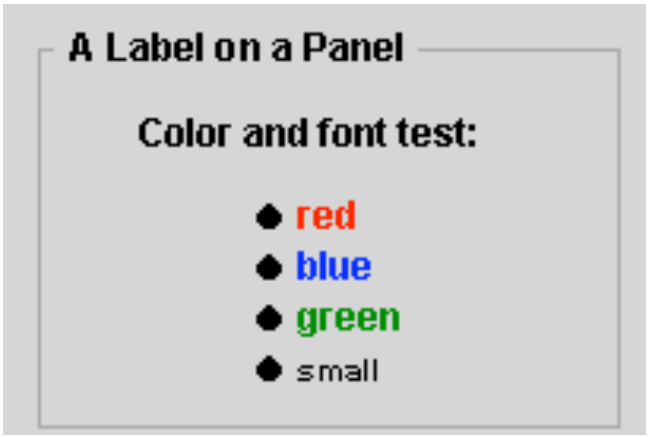


Dialog

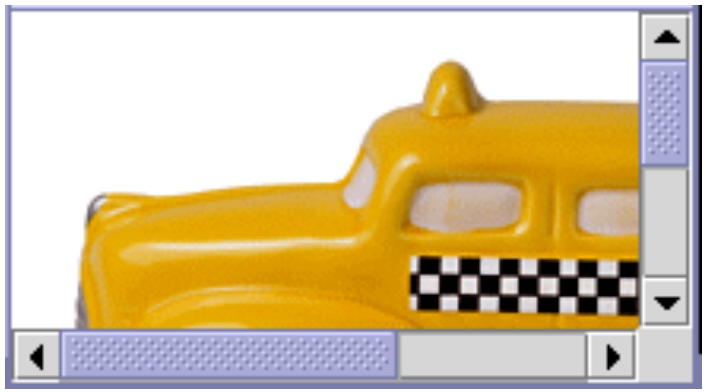


Frame

General-Purpose Containers



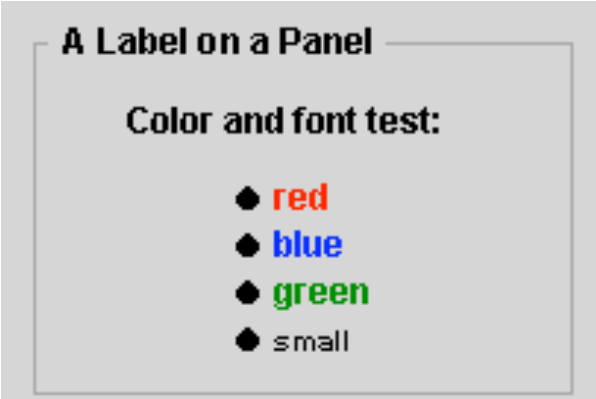
Panel



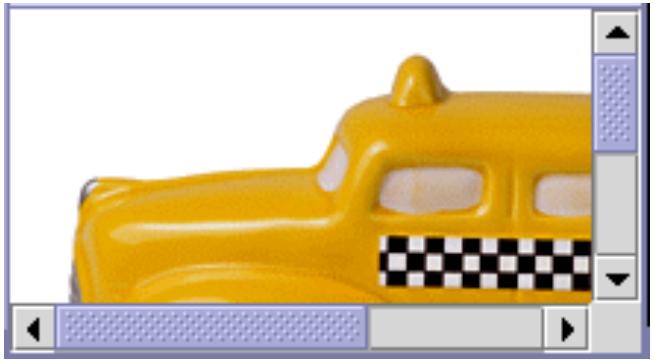
Scroll pane

Applet Dialog Frame

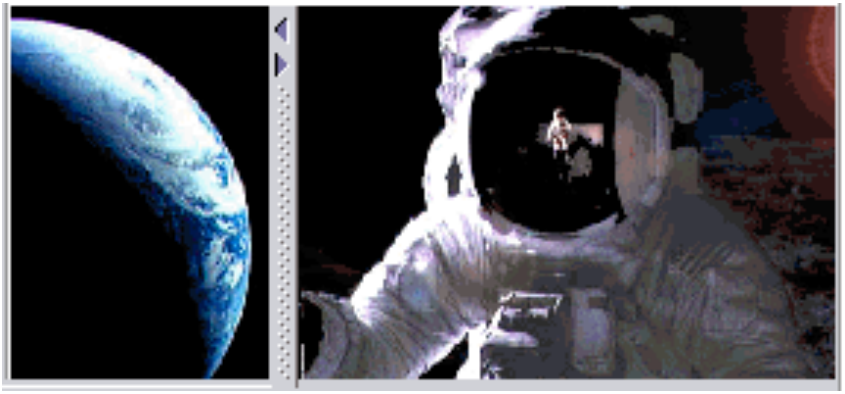
General-Purpose Containers



Panel



Scroll pane

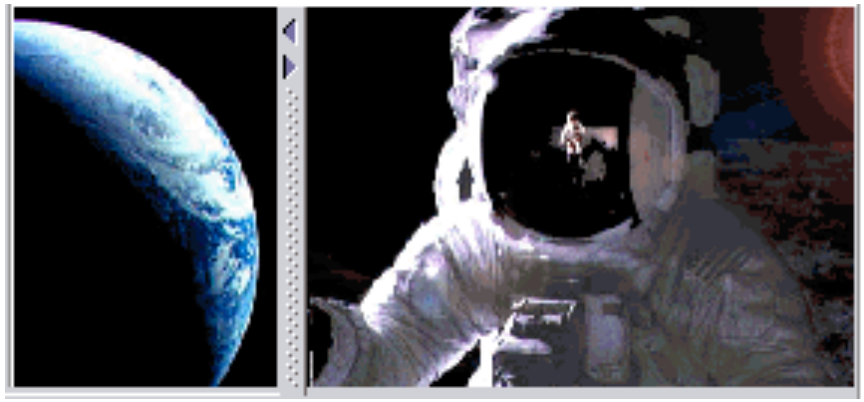


Split pane

Tabbed pane

Tool bar

Special-Purpose Containers

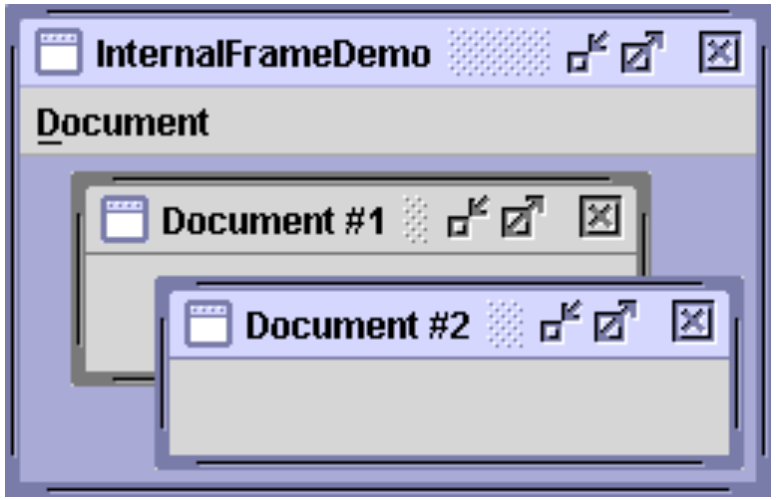


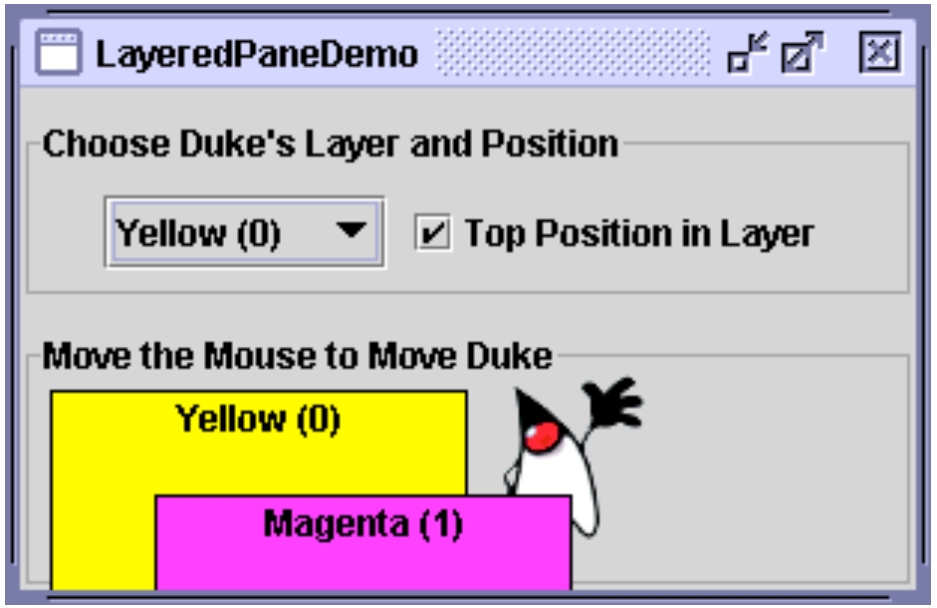
Split pane

Tabbed pane

Tool bar

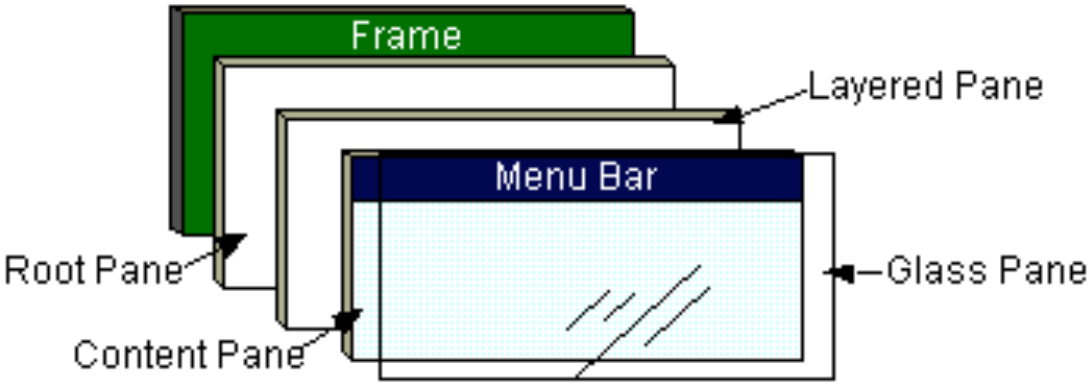
Special-Purpose Containers





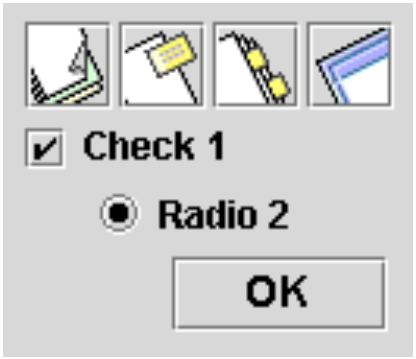
Internal frame

Layered pane

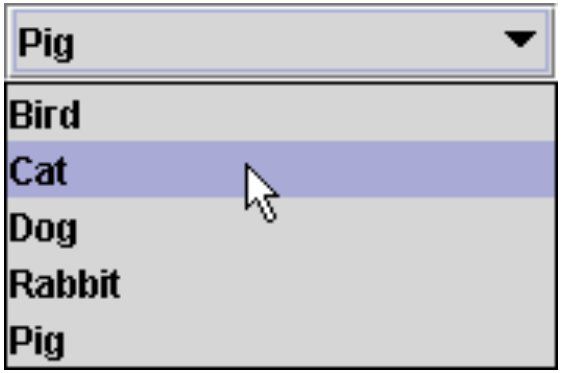


Root pane

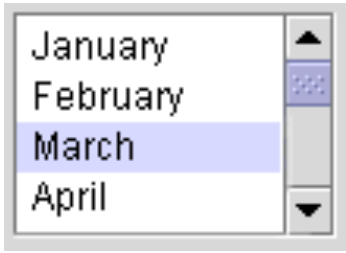
Basic Controls



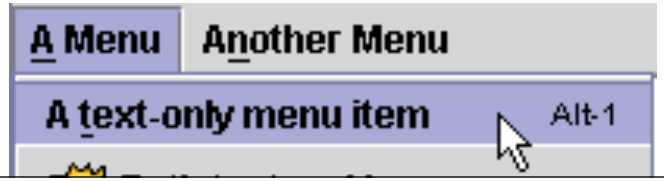
Buttons



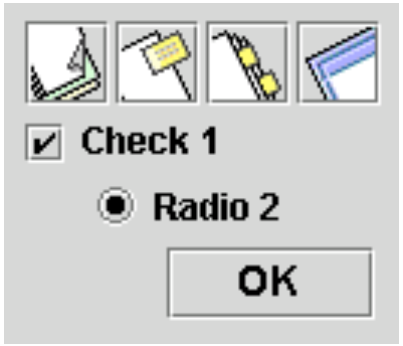
Combo box



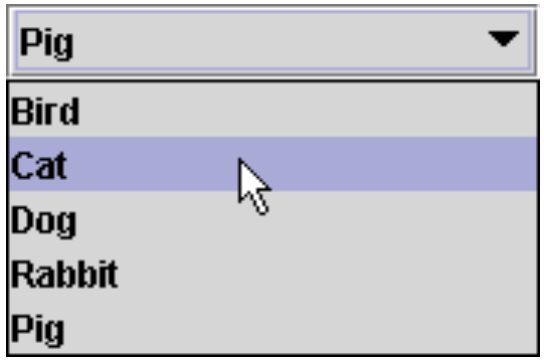
List



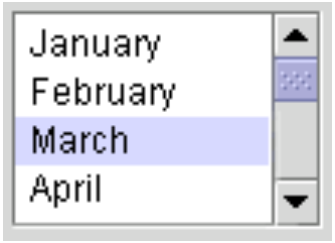
Basic Controls



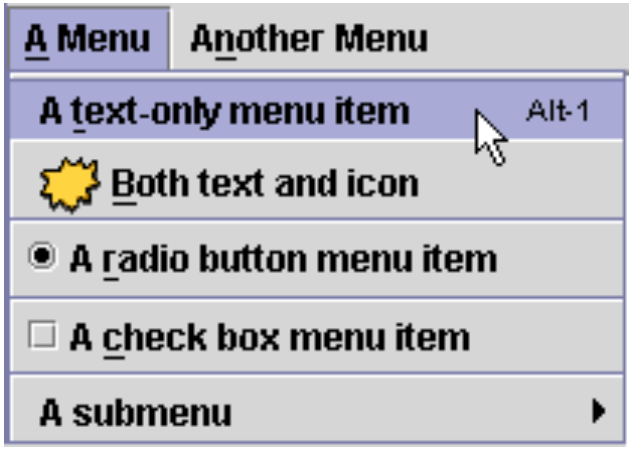
Buttons



Combo box

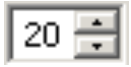


List



Menu

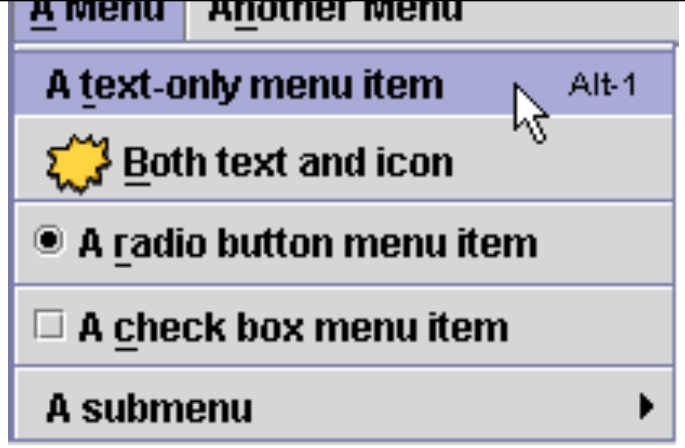
Slider



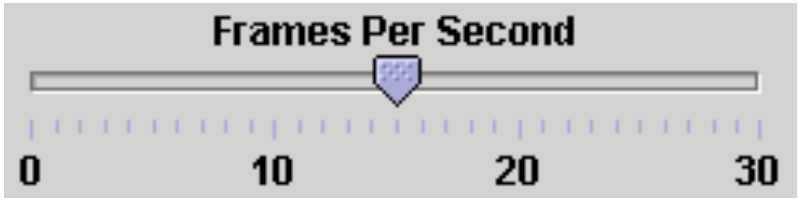
Spinner

Text field or Formatted text field

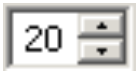
Uneditable Information Displays



Menu



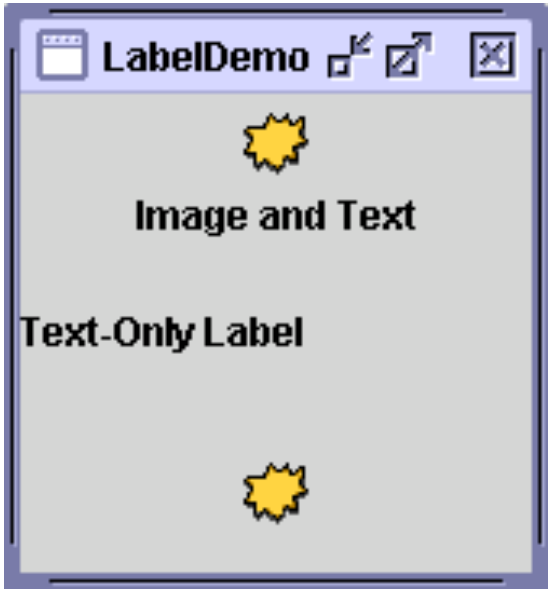
Slider



Spinner

Text field or Formatted text field

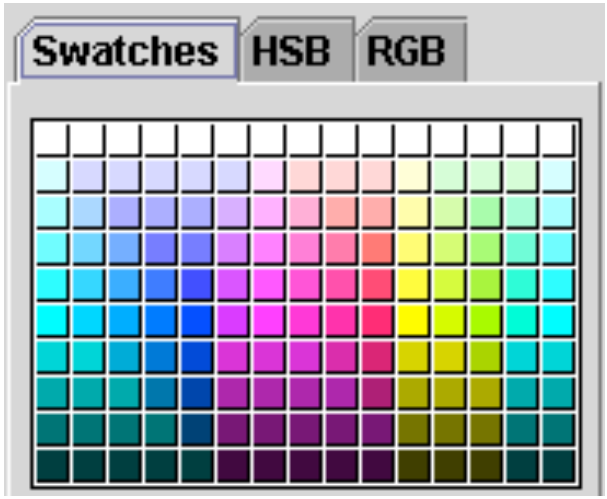
Uneditable Information Displays



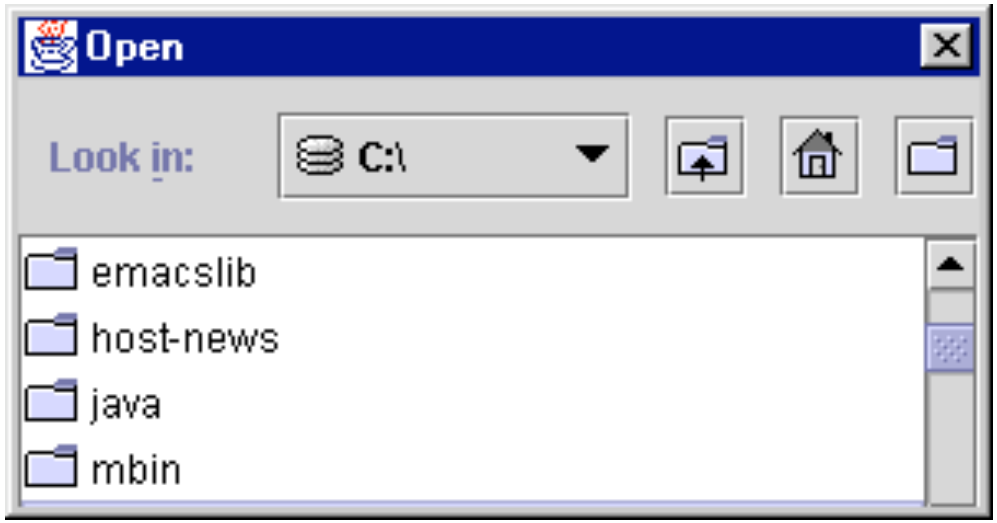
Label

Progress bar

Tool tip



Color chooser

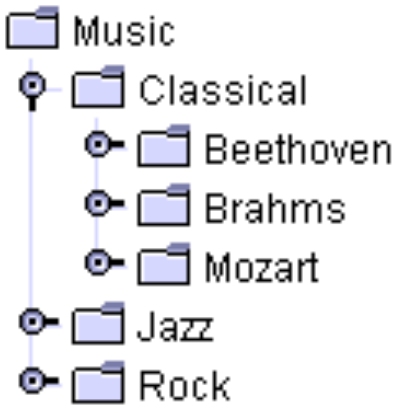


File chooser

First Name	Last Name	Favorite Food
Jeff	Dinkins	
Ewan	Dinkins	
Amy	Fowler	
Hania	Gajewska	
David	Geary	

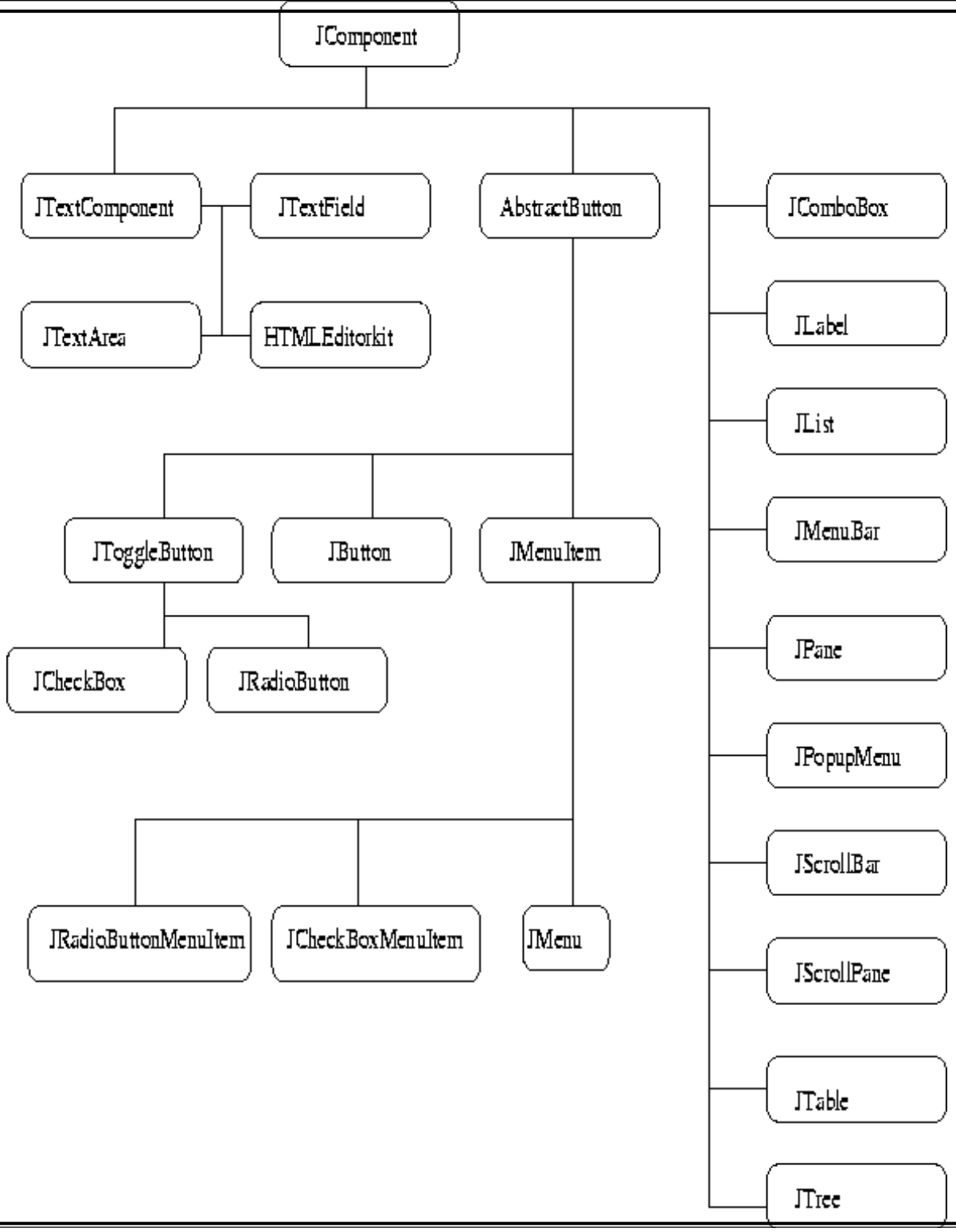
Table

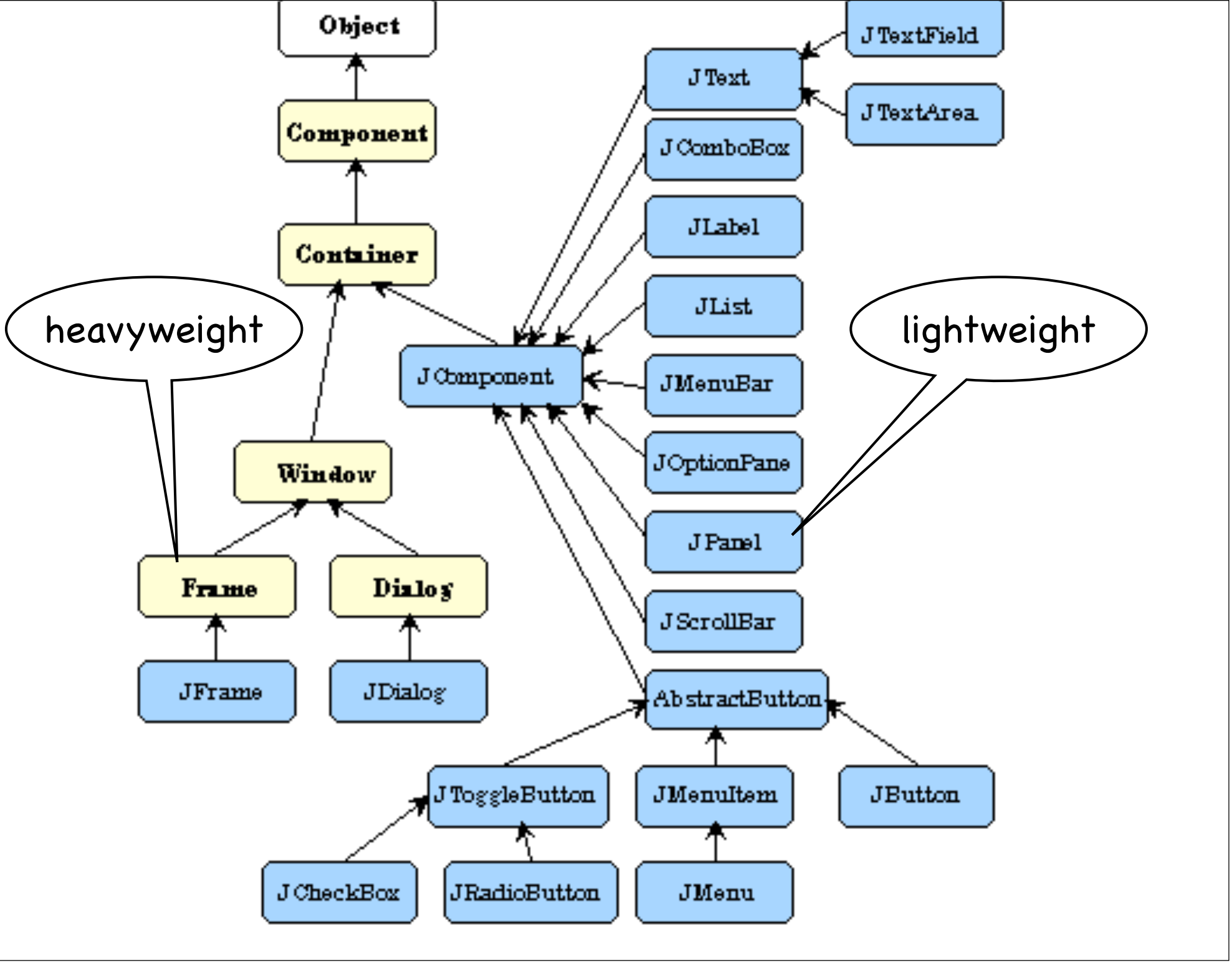
Text



Tree

Swing class hierarchy




Object
Component
Container
heavyweight
Window
Frame
Dialog
JFrame
JDialog
JComponent
JText
JTextField
JTextArea
JComboBox
JLabel
JList
JMenuBar
JOptionPane
JPanel
JScrollBar
AbstractButton
JToggleButton
JMenuItem
JButton
JCheckBox
JRadioButton
JMenu
lightweight

Layout managers

In Java, the placing of the components in the window is performed during run-time. Reason:

- optimal size of component depends on computer (size of text in buttons, e.g.)
- user can resize window on the fly

The strategy of how to place components is determined by the l*ayout manager*. There are several layout managers. Examples:

- FlowLayout (the default layout manager of JPanel)
- BorderLayout (the default layout manager of JFrame)
- “null layout”: doing everything yourself

In Netbeans you design the layout in a more WYSIWYG-way, using the "Design" panel

```
// Example of FlowLayout --  resize the window and see what happens
import javax.swing.*;

public class LayoutExample extends JPanel {

    LayoutExample() {        // this is a constructor
        JButton ontw = new JButton("Ontwaakt,");
        JButton verw = new JButton("verworpenen");
        JButton dera = new JButton("der aarde");

        this.add(ontw);
        this.add(verw);
        this.add(dera);
     }

    public static void main(String[] args) {
        JFrame fraEe = new JFrame("Layout example");
        LayoutExample panel = new LayoutExample();
        frame.add(panel);

        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(400, 200);
        frame.setVisible(true);
    }
}
```

```
// Example of BorderLayout --  resize the window and see what happens
import javax.swing.*;
import java.awt.*;

public class LayoutExample2 extends JPanel {

    LayoutExample2() {        // this is a constructor
        this.setLayout( new BorderLayout() );
        JButton ontw = new JButton("Ontwaakt,");
        JButton verw = new JButton("verworpenen");
        JButton dera = new JButton("der aarde");

        this.add(ontw, BorderLayout.NORTH);
        this.add(verw);
        this.add(dera, BorderLayout.SOUTH);
     }

    public static void main(String[] args) {
        JFrame frame = new JFrame("Layoutvoorbeeld");
        LayoutExample2 panel = new LayoutExample2();
        frame.getContentPane().add(panel);

        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(400, 200);
        frame.setVisible(true);
    }
}
```

- For more on Swing components: see http://java.sun.com/docs/books/tutorial/uiswing/components

- overview...

Netbeans

- Netbeans helps in creating GUI applications
- A "Design" view on a GUI gives a WYSIWYG editor
- This will generate Java code that takes care of:
  - creation of components and setting properties
  - dealing with layout managers
  - adding components to containers
  - linking to code that reacts to events (you still have to write the reaction itself!)

Netbeans (2)

How to achieve this?

1. Create new JFrame Form (menu New File, Java GUI Forms)
2. Click on desired component in the Swing Palette
3. Click in Design window to place component. Resize, reposition, etc.

Netbeans (3)

Writing reaction to, e.g., a button:

1. Select the button in the Design window
   (if you forget this, you don't get the right menu; this happened to me in the demo at the lecture)

2. Double-click on the button; choose Events->Action->actionPerformed

3. You are taken to the method `jButton...ActionPerformed` in the code; write there the code that should be performed when the user clicks the button

Netbeans (4)

- Alternative:

  1. Select the button in the Design window
  2. In the properties panel, choose the Events tab
  3. At the event "actionPerformed", select <none> and press Enter
  4. You are taken to the method `jButton…ActionPerformed` in the code; write there the code that should be performed when the user clicks the button

Netbeans (5)

- See also the help pages of Netbeans:
  - from Help menu: Quick Start Guide
  - from there: Java GUI Applications
  - from there: Adding functionality to Buttons
  - etc.