GUI Development in Java: SWING

A very, very, very quick overview...

FEUP, MIEIC, LPOO, 2016/17

Contents

Intro

History

- AWT - Abstract Windowing Toolkit (java.awt)
  - First editions of Java (JDK 1.0, 1.1)
  - Goal: Multi-platform GUI development
- ⇨ Swing / Java Foundation Classes (javax.swing)
  - Beyond JDK v1.2
  - New framework, on top of AWT, allowing the development of more powerful and complex GUIs.
  - Still needs some AWT features
- SWT - The Standard Widget Toolkit (org.eclipse.swt)
  - Uses OS native objects, thus is more efficient, but Eclipse IDE specific.

Common GUI features

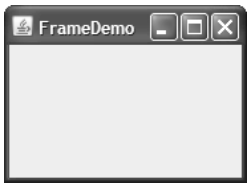
- Structure
  - Component hierarchy, from top-level windows to atomic-level controls (buttons, gauges,…)
  - *Reusable (*through configuration and extension) *Widgets (*graphical components) toolkits usage.
  - Widgets have state (*enabled/disable*, *have contents*, …*)*
- Behaviour
  - Event-driven: User acts upon the GUI triggering events.
  - Screen redrawing occurs on OS request.
- *Look and feel* and *layout*
  - Aesthetic/Visual features (not directly related to the above)

SWING component catalog

Component types (SWING)

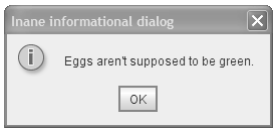
- Containers – may contain other components
  - Top-level (frames, dialogs, etc.)
  - Generic intermediate-level (panels, etc.)
  - Specific intermediate-level (tables, etc.)
- (Atomic) Components– don't contain other components.
  - Basic controls (buttons, text-boxes, etc.)
    - Get input from user, show short and simple data.
  - Non-editable data holders (labels, etc.)
    - Only to show information to the user.
  - Formatted, interactive, data holders
    - Show highly formatted data, editable (if needed) to the user.

Top-Level containers



"frame"

(JFrame)



"dialog box"

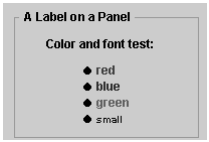
(JOptionPane, JDialog)

Applet

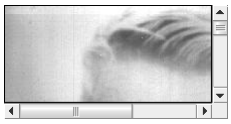
(JApplet)

Similar to "frame", but runs inside a web browser.

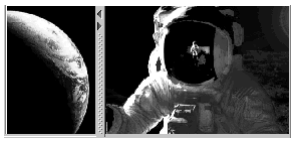
Generic intermediate-level containers



"panel"
(JPanel)

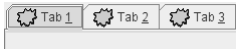


"scroll pane"
(JScrollPane)



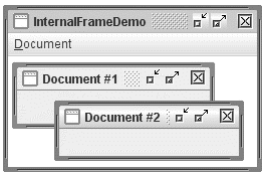
"split pane"
(JSplitPane)

"toolbar"
(JToolBar)

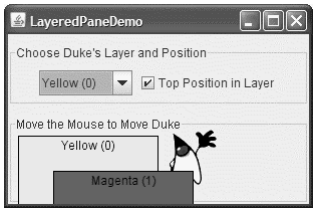


"tabbed pane"
(JTabbedPane)

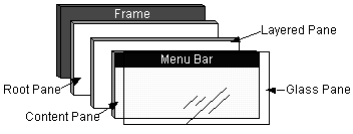
Specific intermediate-level cnt.



"internal frame"
(JInternalFrame)

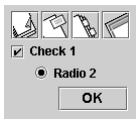


"Layered pane" (z-order)
(JLayerPane)



"aRoot pane
(JRootPane)
automatically created by top-level containers

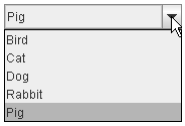
Basic controls



"buttons"
(JButton, JCheckBox, JRadioButton)

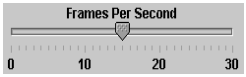
"text boxes"
(JTextField, JFormattedTextField)

JList

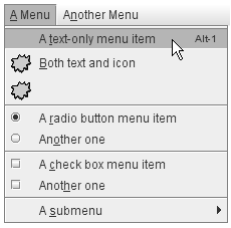


JComboBox

JSpinner

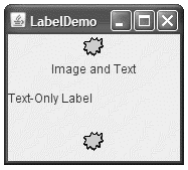


JSlider



"menus"
(JMenuBar, JMenu, JMenuItem, etc.)

Non-editable data holders

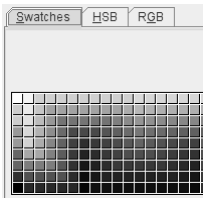


"label"
(JLabel)

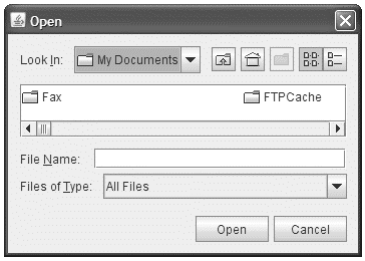
"progress bar"
JProgressBar)

"tool tip"
(JToolTip)

Formatted, interactive, data holders



JColorChooser



JFileChooser

JTable

Texto

(JTextArea, JEditorPane, etc.)

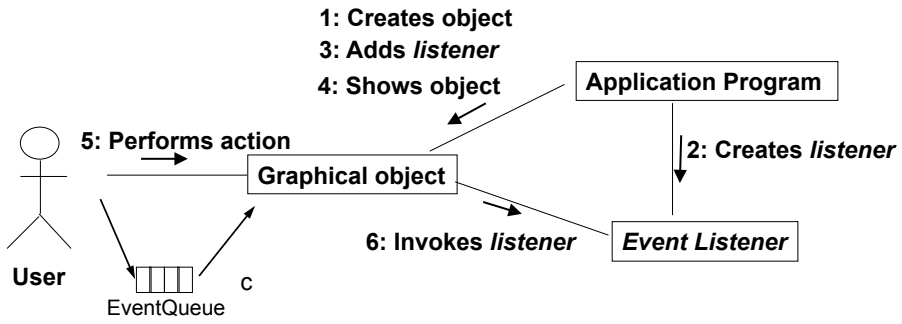
JTree

Modal, non-modal windows

- ***Modal window*** : User can't interact with other windows, while this remains open.
  - Typical scenario: Dialog boxes created using ***JOptionPane***
- ***Non-model window*** : User can change focus to other windows, without the nedd to close this one.
  - Typical scenario: top-level JFrame windows.

Event-driven mechanism

Event-driven behaviour (AWT)

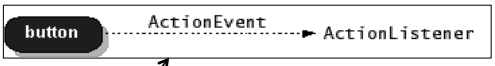


Swing automatic event processing cycle:
- Pop next event on queue
- Identifies target object
- Dispatches event to target object

(UML Communication diagram)

Event-type class hierarchy

java.lang.Object
 +--java.util.EventObject
  +--java.awt.AWTEvent
   +--java.awt.event.ActionEvent
   +--java.awt.event.TextEvent
   +--java.awt.event.ComponentEvent
    +--java.awt.event.FocusEvent
    +--java.awt.event.WindowEvent
    +--java.awt.event.InputEvent
     +--java.awt.event.KeyEvent
     +--java.awt.event.MouseEvent



Look and feel and layout managers

Layout managers (AWT)

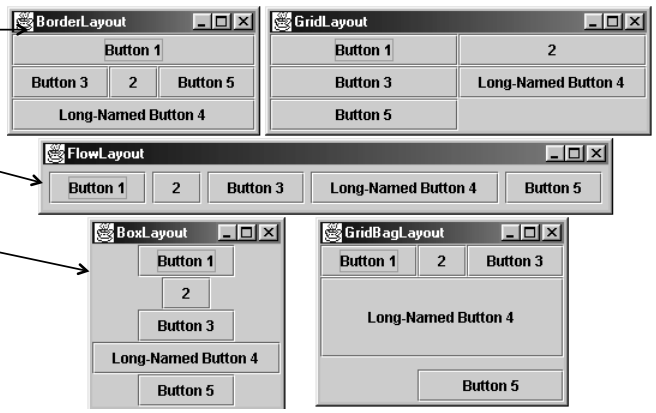
- Special objects that decide where to place (and resize) components inside a container (panel, etc.) according to a set of constraints.

North, South, East, West, Center

Top-down, Left to right

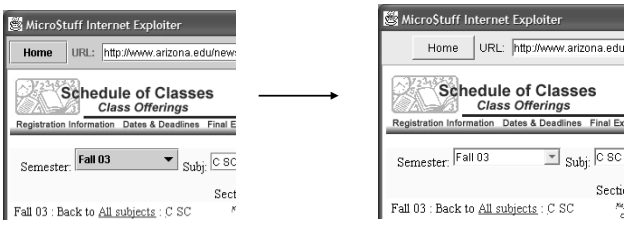
Horizontal or vertical

null Layout: absolute x, y



Look and Feel

- SWING is multi-platform, thus it can have different visuals (look and feel).
  - Macintosh, Windows, Motif, etc.
- Default *look & feel* (metal) can be changed to the current OS:
  - `UIManager.setLookAndFeel( UIManager.getSystemLookAndFeelClassName());`

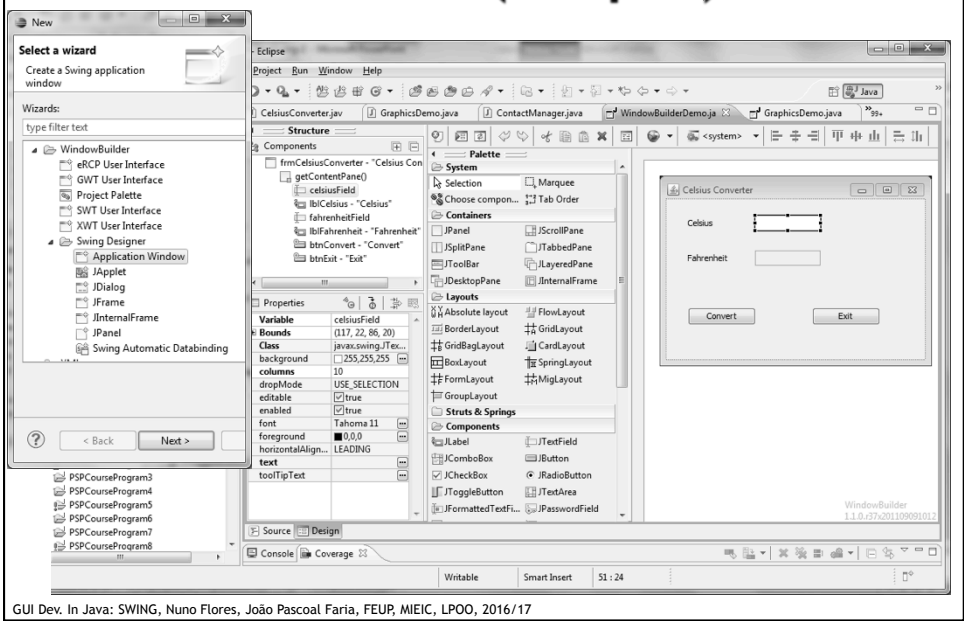


Visual Editors

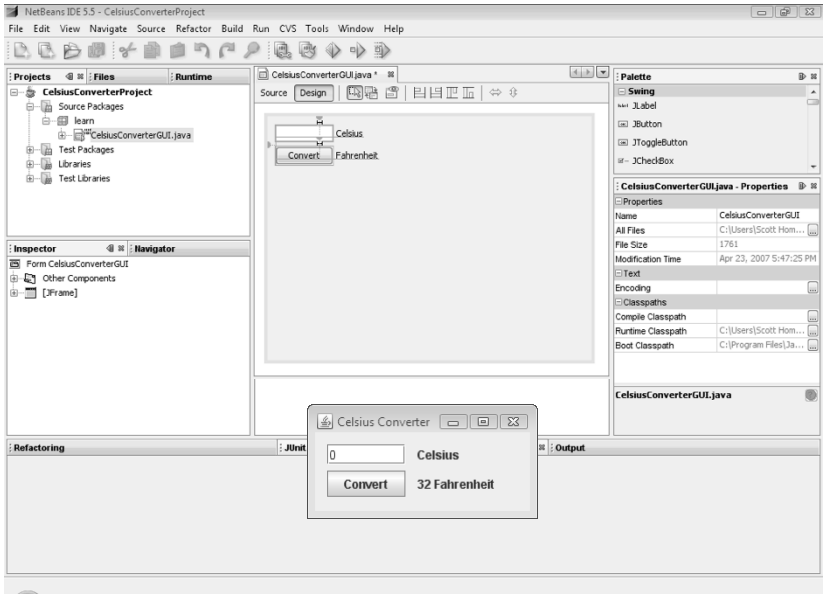
Paradigm

- Made popular by Visual Basic
- Aliases:
  - Interface builder
  - Visual editor
  - Interface designer
- Common features:
  - Graphical components palette (*drag and drop*)
  - Wysiwig (what you see is what you get) editor
  - Source-code generators
  - Current object properties list

WindowBuilder (Eclipse)

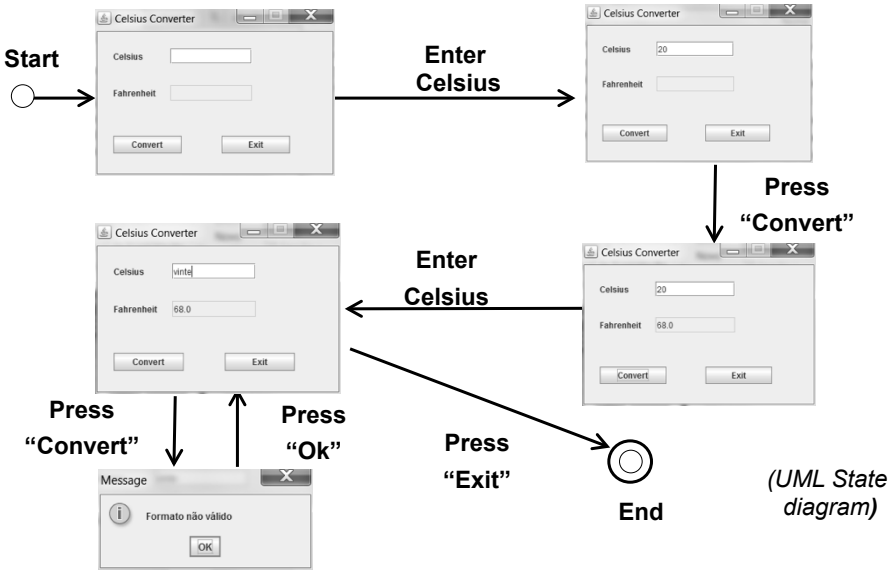


Net Beans IDE



Example: Celsius to Fahrenheit converter

Required usage



Step #1 – Create main window

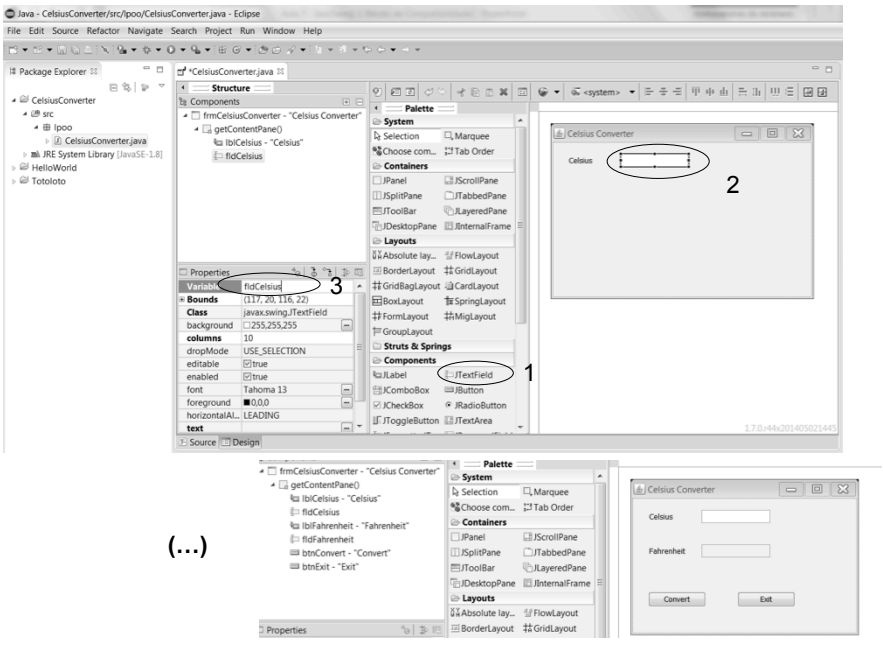
On Eclipse (withWindow Builder):

- File -> New -> Java Project -> [Project Name = CelsiusConverter]
- File -> New -> Other … -> Window Builder -> Swing Designer -> Application Window > [Package = lpoo, Name = CelsiusConverter]
- CelsiusConverter.java -> Open With -> WindowBuilder Editor -> Design
  - Change title of main window (*frame*) to "Celsius Converter"
  - Disable window resizing (*resizable*) of main window (*frame*)
  - In the contents area, (*getContentPane*), select absolut layout

Step #2 – Create object

At the "Design" view:

- Add label (JLabel) "Celsius"
- Add text field (JTextField), named "fldCelsius", for entering Celsius value by the user
- Add label (JLabel) "Fahrenheit"
- Add text field (JTextField), named "fldFahrenheit", non-editable, to show Fahrenheit value.
- Add button (JButton) "Convert"
- Add button (JButton) "Exit"



Step 3 – Define "Exit" button action

On the "Design" view, double-click on button "Exit", and add instruction "System.exit(0", on the *listener* method body.

anonymous inner class

```
JButton btnExit = new JButton("Exit");
btnExit.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        System.exit(0);
    }
});
btnExit.setBounds(178, 158, 97, 25);
frmCelsiusConverter.getContentPane().add(btnExit);
```

(creates, on-the-fly, anonymous inner class that implements ActionListener interface)

Step #4 – Define "Convert" button action

At the "Design" view, double-clink on button"Convert" and add the following code to the *listener* method body

```
JButton btnConvert = new JButton("Convert");
btnConvert.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        double tempCelsius;
        try {
            tempCelsius =
                Double.parseDouble(fldCelsius.getText());
        }
        catch (NumberFormatException ex) {
            JOptionPane.showMessageDialog(frmCelsiusConverter,
                "Formato não válido");
            return;
        }
        double tempFahr = tempCelsius * 1.8 + 32;
        fldFahrenheit.setText(Double.toString(tempFahr));

    }
});
```

Refs. and other sources

- http://docs.oracle.com/javase/tutorial/uiswing/index.html
- Thinking in Java, 3rd, Ch. 14: Creating Windows & Applets