Theory pills
Lab 3

Main types of containers

- Frame (JFrame)
- **Dialog (JDialog)**
- Panel (JPanel)
- Scroll Panel (JScrollPane)
- Tabbed panel (JTabbedPane)
- Tool bar (JToolBar)

Dialogs (JDialog)

- Generally used to get input data from the user and to show messages.
- They derive from another component.
- The cannot have menu bar.
- Types:
  - **Modal.** Users cannot interact with the rest of the application until the dialog is closed, but they do not prevent interactions with other applications.
  - **Unmodal**. The do not prevent users from interaction with the rest of the application while opened.

JDialog: Custom dialogs

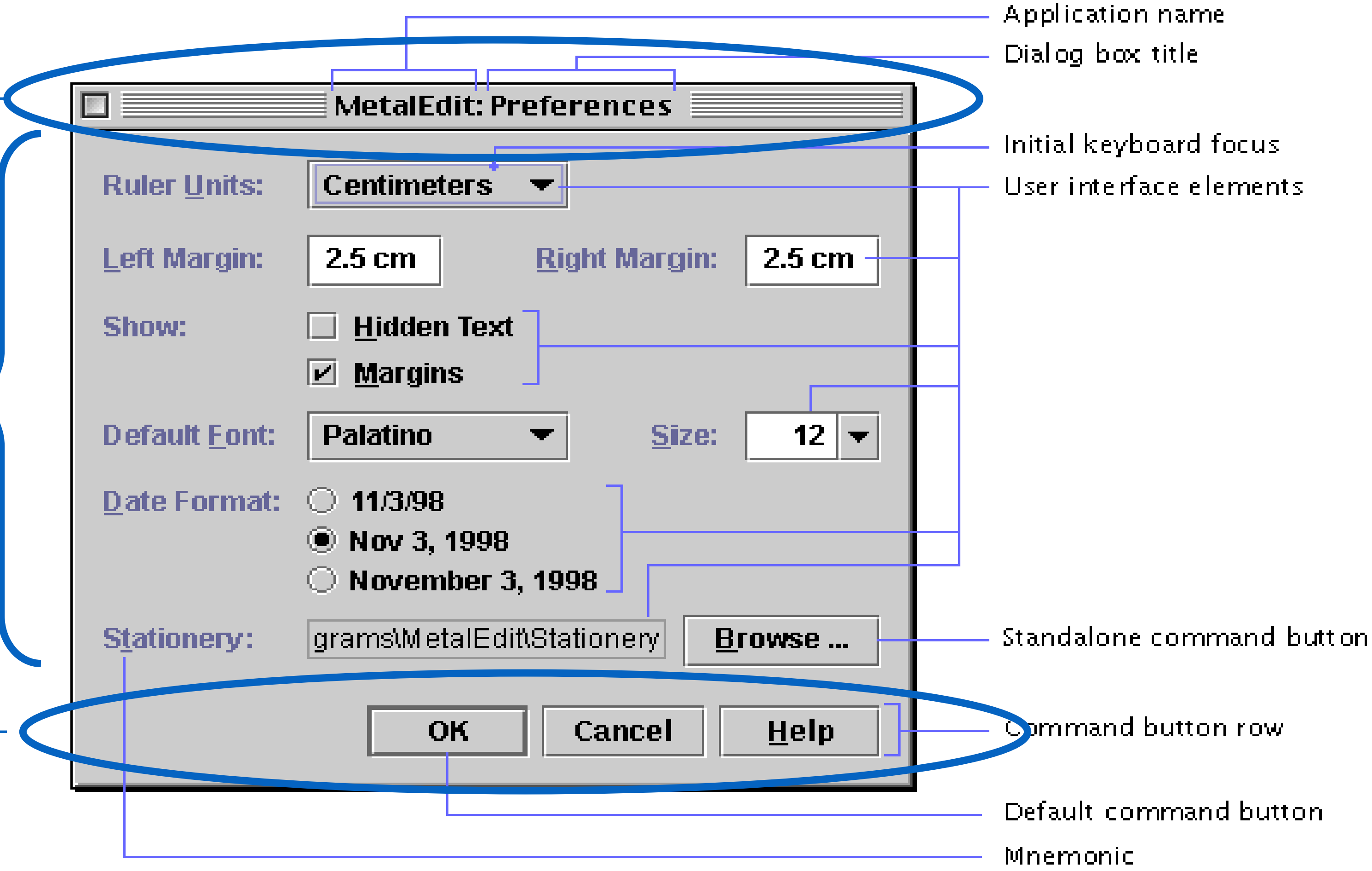
We can also design our own dialogs...

Elements:

Title

UI Interface elements

Buttons row



MetalEdit: Preferences

Ruler Units: Centimeters

Left Margin: 2.5 cm Right Margin: 2.5 cm

Show: □ Hidden Text ☑ Margins

Default Font: Palatino Size: 12

Date Format: ○ 11/3/98 ● Nov 3, 1998 ○ November 3, 1998

Stationery: grams\MetalEdit\Stationery Browse ...

OK Cancel Help

Application name

Dialog box title

Initial keyboard focus

User interface elements

Standalone command button

Command button row

Default command button

Mnemonic

Main features

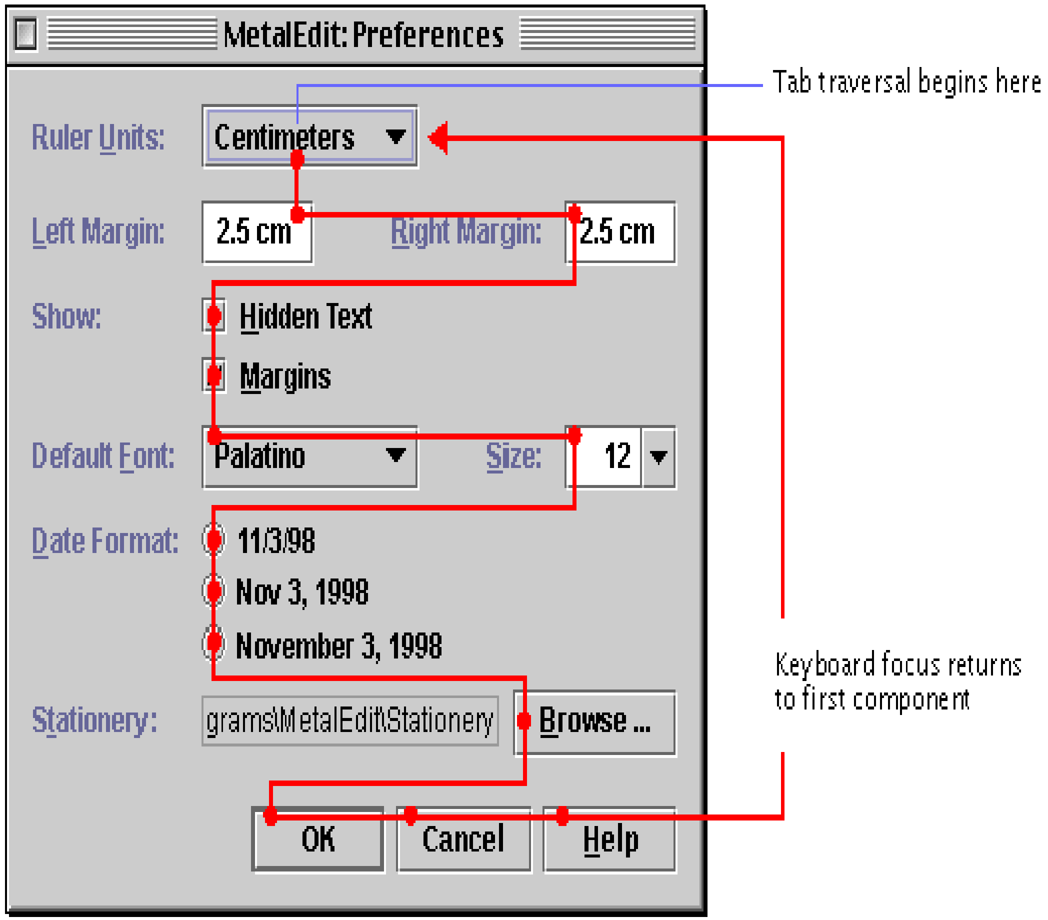
- The title should be “Application name: dialog name”
- We have to include mnemonics for every elements but
  - The default button

```
this.getRootPane().setDefaultButton(btnNext)
```

  - The Cancel button.
- When we open a dialog the focus must be on the component that the user is expected to use first.

Tab order

- It should respect the reading order of the user.
  - Be careful!: And in Arabia?



MetalEdit: Preferences

Ruler Units: Centimeters

Left Margin: 2.5 cm

Right Margin: 2.5 cm

Show: □ Hidden Text

□ Margins

Default Font: Palatino

Size: 12

Date Format: ○ 11/3/98

○ Nov 3, 1998

○ November 3, 1998

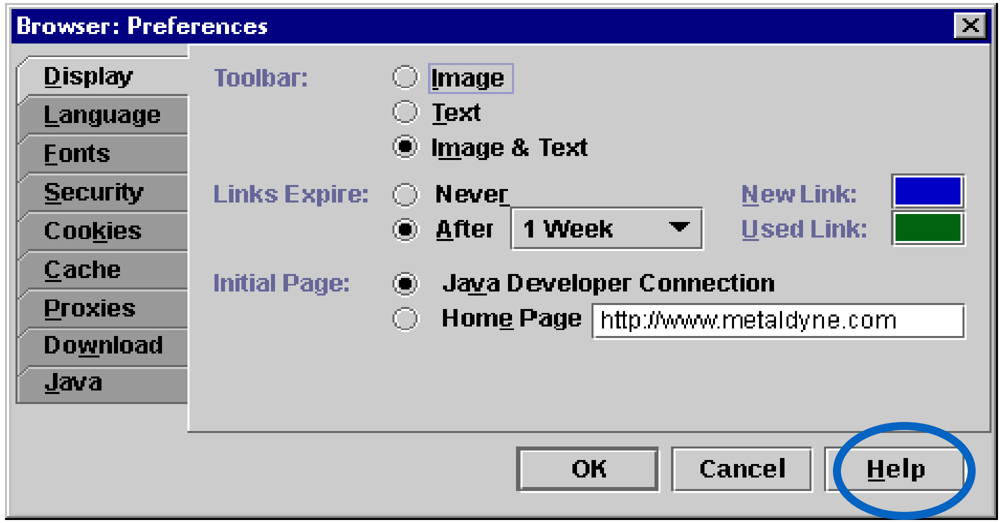
Stationery: grams\MetalEdit\Stationery

Tab traversal begins here

Keyboard focus returns to first component

Where should we place the Command Buttons?

- All those buttons whose effect affects the whole dialog must be placed in a row on the **bottom of the dialog**, and **aligned to the right**
- If Help button was used to show additional information regarding the dialog, It must be the last on the right.



Toolbar: ○ Image ○ Text ● Image & Text

Links Expire: ○ Never ● After 1 Week

New Link: 🟦

Used Link: 🟩

Initial Page: ● Java Developer Connection ○ Home Page http://www.metaldyne.com

Default button in a dialog

- It is activated whenever the user presses Intro. It executes the actions linked to that button (must be the most usual).
- An ***unsafe option*** (that could, for example, lead to user data lost) should **never** be the default button.

- The default button **does not need to have the focus** when the user presses intro.
- If the dialog has default button, **it must be the first command button** of the line.
- It does **not** need ***mnemonic***
- In Java: getRootPane().setDefaultButton(*buttonName*);

Cancel button

- It is activated with the Esc key. It fires the actions associated with the Cancel button.
- This behavior **must be implemented manually**, there is no way to determine which is the Cancel button.
  - Solution: Check continuously if the pressed key is Esc and activate the logic of the Cancel button whenever it was detected (keyboard events management)

Dialog (II)

- In Swing, several standard classes supporting dialogs:
  - JOptionPane
  - JColorChooser
  - JFileChooser
- All of them are **modal**

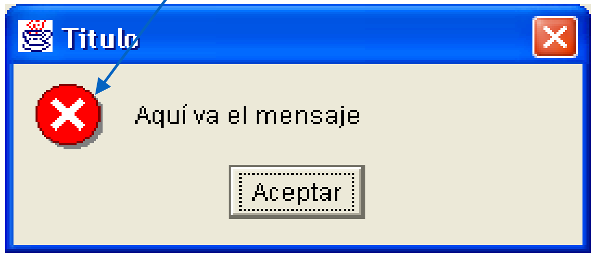
JOptionPane

- It allow us to create and customize several types of dialogs, specifying…
  - Icons (custom, none or a standard one)
  - Title,
  - Text
  - Buttons text
  - Location in the screen.
- Standard icons: *question, information, warning* y *error*
- Main static methods.
  - *showMessageDialog*
  - *showConfirmDialog*
  - *showInputDialog y*
  - *showOptionDialog*

JOptionPane.showMessageDialog

- It shows a modal dialog with one only Ok button.
- We can customize message, icon and title.
- Examples:
  - `JOptionPane.showMessageDialog(this, “Mensaje”);`
  - `JOptionPane.showMessageDialog(this,”Mensaje”,”Titulo”,JOptionPane.WARNING_MESSAGE);`
  - `JOptionPane.showMessageDialog(this,”Mensaje”,”Titulo”,JOptionPane.ERROR_MESSAGE);`
  - `JOptionPane.showMessageDialog(this,”Mensaje”,”Titulo”,JOptionPane.INFORMATION_MESSAGE);`
  - `JOptionPane.showMessageDialog(this,”Mensaje”,”Titulo”,JOptionPane.QUESTION_MESSAGE);`
  - `JOptionPane.showMessageDialog(this,”Mensaje”,”Titulo”,JOptionPane.PLAIN_MESSAGE);`

Specify the parent component. It can be the frame or any other component. The dialog will be shown in the middle of the parent component.



Titulo

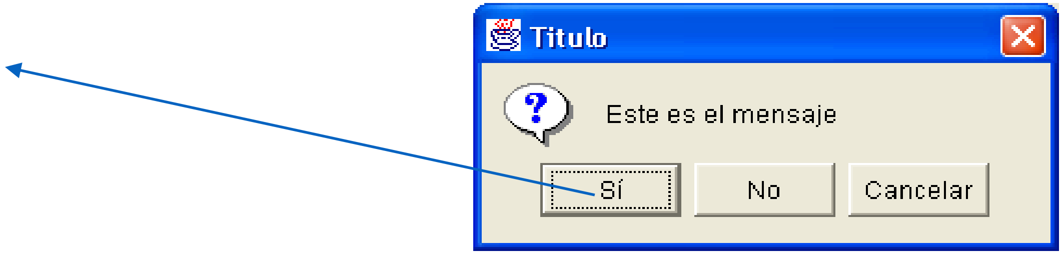
Aquí va el mensaje

JOptionPane.showConfirmDialog

- It shows a modal dialogs to ask for confirmation.
- It allow us to specify: messange, icon, title and the combination of buttons
- Examples:

```
int resp = JOptionPane.showConfirmDialog(this, “Mensaje”);
int resp = JOptionPane.showConfirmDialog(this,”Mensaje”,”Titulo”, JOptionPane.YES_NO_OPTION);
int resp = JOptionPane.showConfirmDialog(this,”Mensaje”,”Titulo”, JOptionPane.YES_NO_CANCEL_OPTION);
int resp = JOptionPane.showConfirmDialog(this,”Mensaje”,”Titulo”, JOptionPane.OK_CANCEL_OPTION);
int resp = JOptionPane.showConfirmDialog(this,”Mensaje”,”Titulo”, JOptionPane.DEFAULT_OPTION);
```

```
 if (resp == JOptionPane.YES_OPTION) {
      . . .
}
```

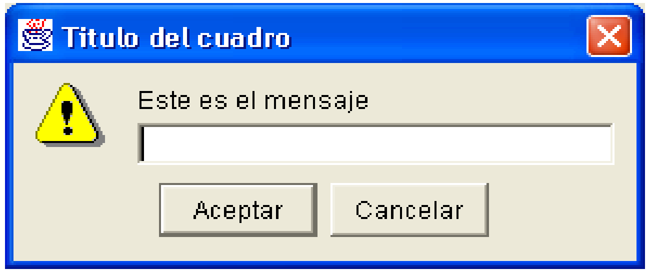


Este es el mensaje

JOptionPane.showInputDialog

- Modal dialog that allows users to send a String to the system.
- This dialog must be user carefully, given that data validation can be done only after closing the dialog.
- Examples:

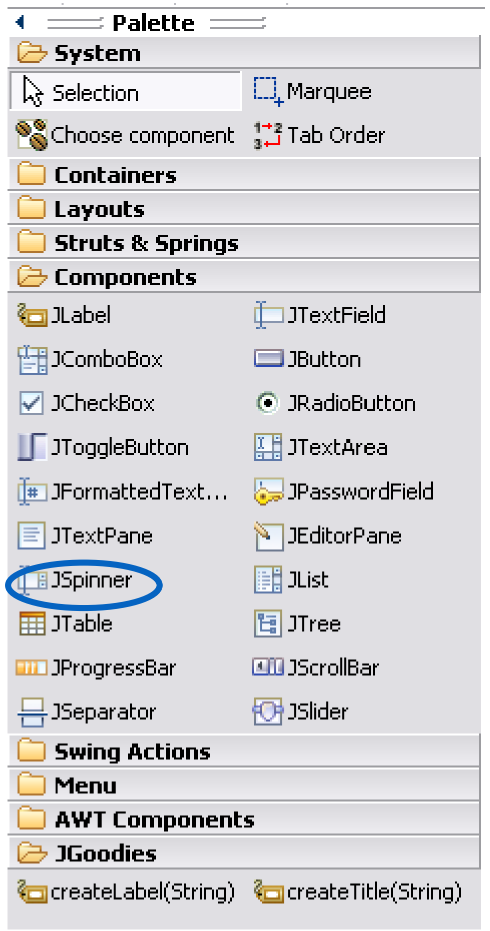
- `String valor = JOptionPane.showInputDialog(this, mensaje);`
- `String valor = JOptionPane.showInputDialog(this,”Mensaje”,”Titulo”,JOptionPane.PLAIN_MESSAGE);`
- `String valor = JOptionPane.showInputDialog(this,”Mensaje”,”Titulo”,JOptionPane.INFORMATION_MESSAGE);`
- `String valor = JOptionPane.showInputDialog(this,”Mensaje”,”Titulo”,JOptionPane.WARNING_MESSAGE);`
- `String valor = JOptionPane.showInputDialog(this,”Mensaje”,”Titulo”,JOptionPane.QUESTION_MESSAGE);`
- `String valor = JOptionPane.showInputDialog(this,”Mensaje”,”Titulo”,JOptionPane.ERROR_MESSAGE);`



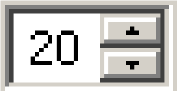
Titulo del cuadro

Este es el mensaje

Spinner (JSpinner)



- Allow to select a value within a range of possible options
- The values change pressing the slide-buttons, or can also be introduced directly.
- It has some domain logic: we must configure maximum, minimum and increment values.
- Those values can be changed at runtime.

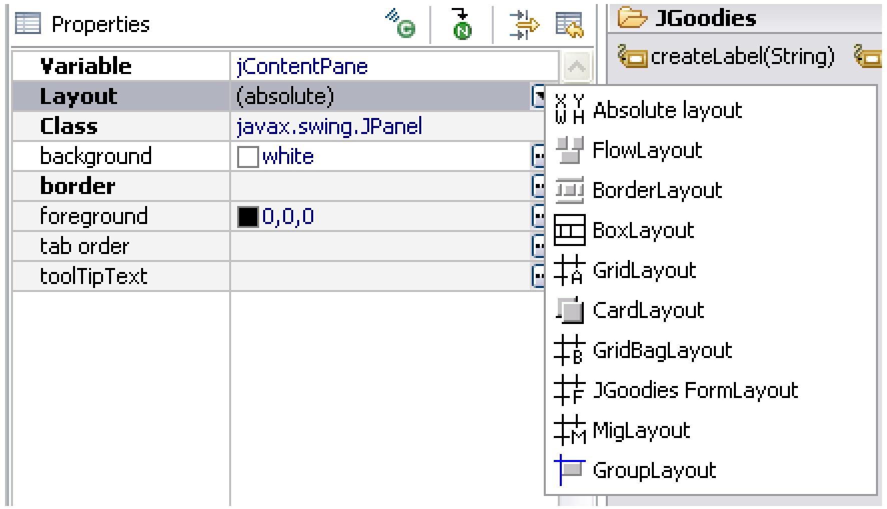


Layouts

- They determine the way the components are organized inside a container, specifying size and placement.
- WE must use the one that suits better with the needs of the application
- Steps…
  - Create the container
  - Determine the layout
  - Add the components to the container

Types of Layouts

- The most popular are…
  - **FlowLayout**
  - BorderLayout
  - CardLayout
  - GridLayout
  - BoxLayout
  - GridBagLayout

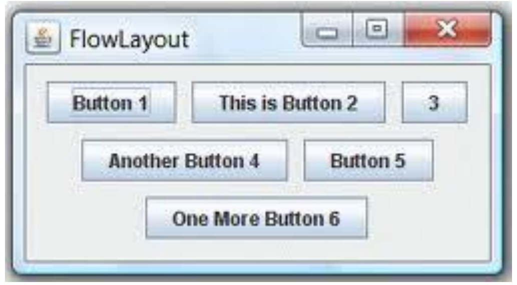


Variable	jContentPane
Layout	(absolute)
Class	javax.swing.JPanel
background	□ white
border	
foreground	■ 0,0,0
tab order	
toolTipText	

- By default…
  - JFrame, JDialog → BorderLayout
  - JPanel, JScrollPane → FlowLayout

FlowLayout

- The simplest and the default in every panel
- Components are placed in one or more rows starting from the top of the panel.
- New rows will be created if necessary
- If the size of the container is modified, the components will replace themselves.
- We can specify
  - Alignment.
    - Property *alignment: left, rigth, center*
  - Space between components.
    - Properties: *horizontalgap, verticalGap*


FlowLayout
Button 1
This is Button 2
3
Another Button 4
Button 5
One More Button 6