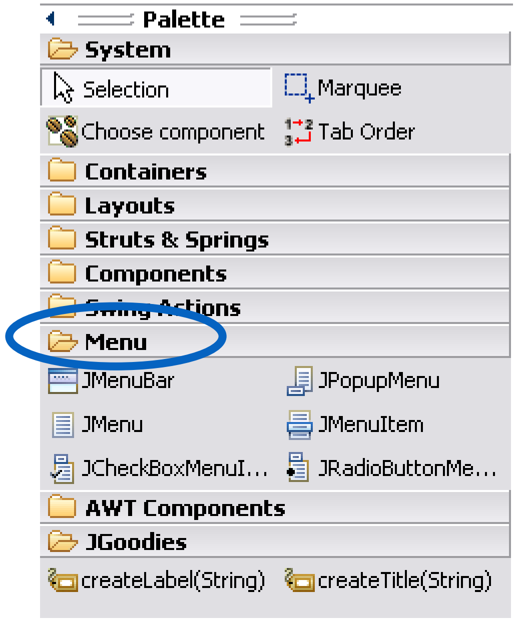
# Theory pills
# Lab 5

# Other components.

- **Menus (*JMenuBar*)**
- Lists (*JList*)
- Sliders (*JSlider*)
- Progress bars (*JProgressBar*)
- Spinners (*JSpinner*)
- Tables (*JTable*)
- Trees (*JTree*)

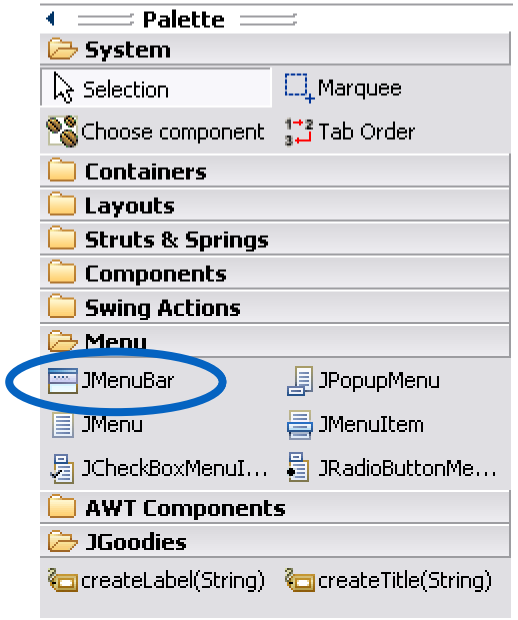
# Menus



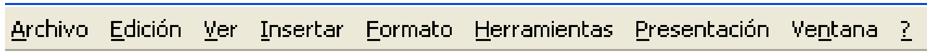
- They show a set of options (menu items) that can be ran over or selected by the user.

- They can be:
    - Drop-down
    - Submenu
    - Context menu

# Menu bars (JMenuBar)

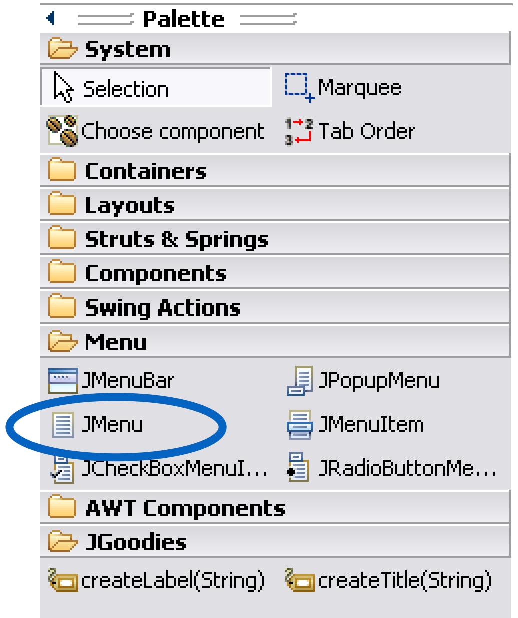


- They are in the upper side of a main window, and they contain menu titles that describe the contents of each menu.

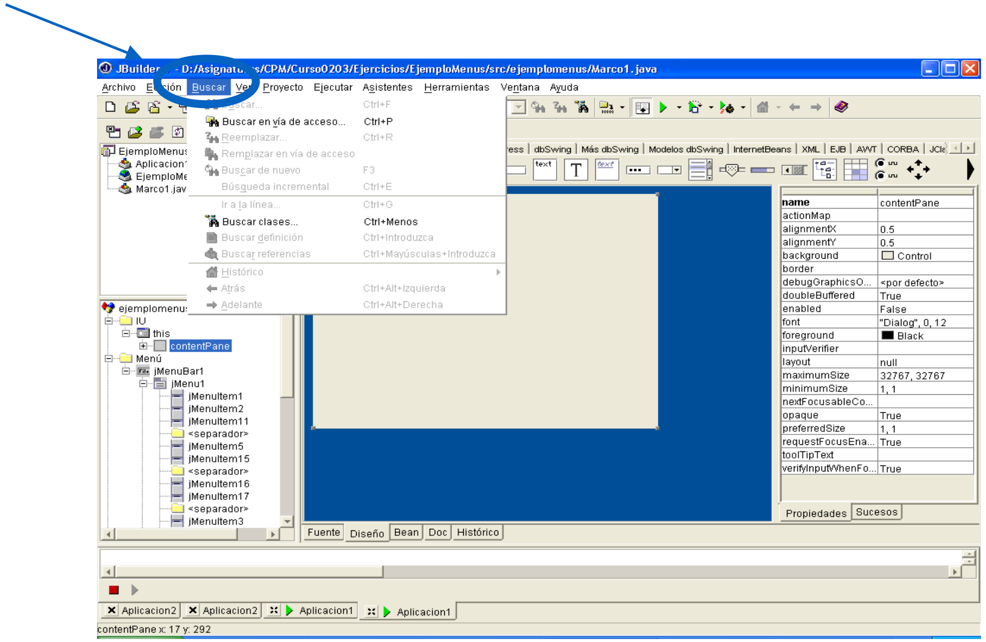


- Menu titles are generally text, but they can contain graphics or both.
- Mnemonics must be included in each title.
- Titles should be formed by simple words.

# Drop-down Menus



- It is shown whenever the user selects a menu title from the menu bar

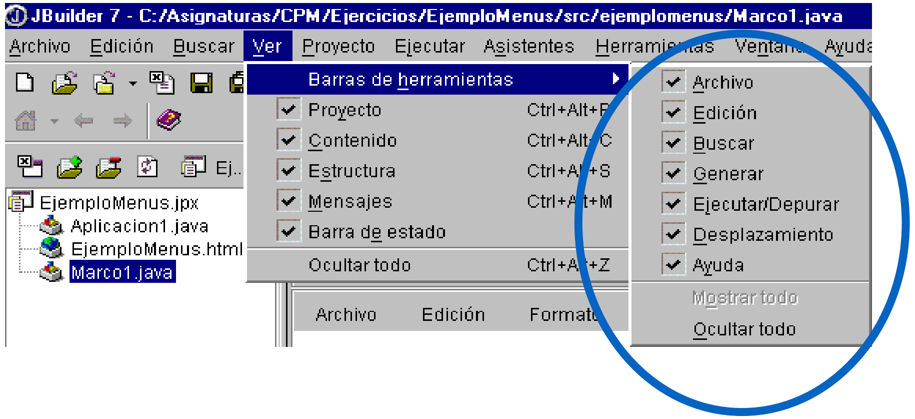


| | |
|---|---|
| **name** | contentPane |
| actionMap | |
| alignmentX | 0.5 |
| alignmentY | 0.5 |
| background | □ Control |
| border | |
| debugGraphicsO... | <por defecto> |
| doubleBuffered | True |
| enabled | False |
| font | "Dialog", 0, 12 |
| foreground | ■ Black |
| inputVerifier | |
| layout | null |
| maximumSize | 32767, 32767 |
| minimumSize | 1, 1 |
| nextFocusableCo... | |
| opaque | True |
| preferredSize | 1, 1 |
| requestFocusEna... | True |
| toolTipText | |
| verifyInputWhenFo... | True |

- Manu bars contain every drop-down menu and submenu of the application

# Submenus

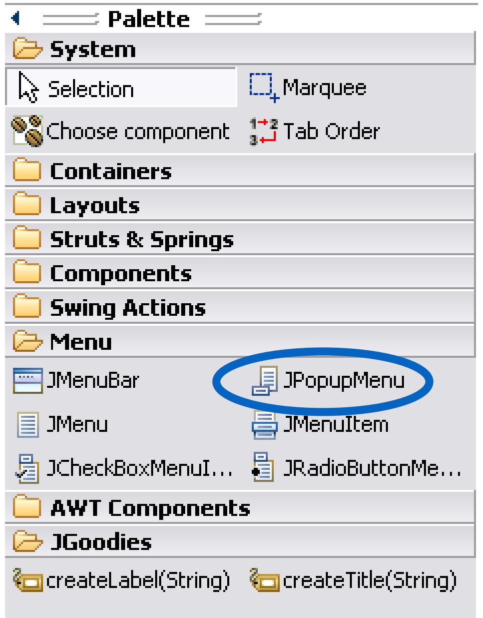
- Menus that users open clicking or slipping the mouse on a menu element



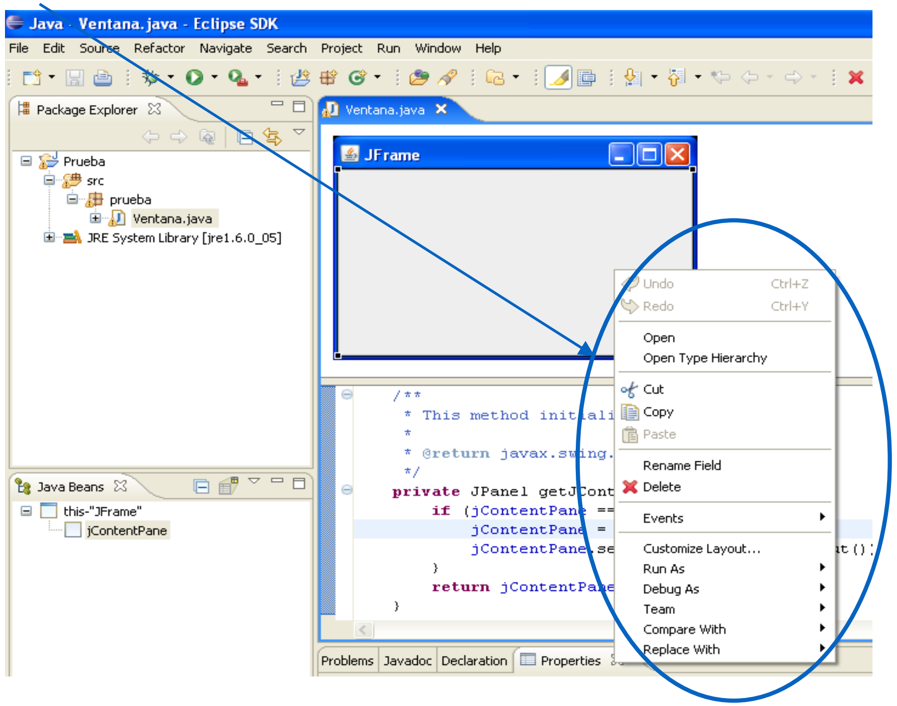
- They also include **mnemonics** and **shortcuts**
- **Second submenu levels should be avoided**.
    - We can replace them with a new dialog

*Notice that!*

# Context Menus (JPopupMenu)



- Pop-up menus
- They provide menu elements that can be applied to the object or region pointed by the mouse.



```
/**
 * This method initiali
 *
 * @return javax.swing.
 */
private JPanel getJCont
    if (jContentPane ==
        jContentPane =
        jContentPane.se                    ut();
    }
    return jContentPane
}
```

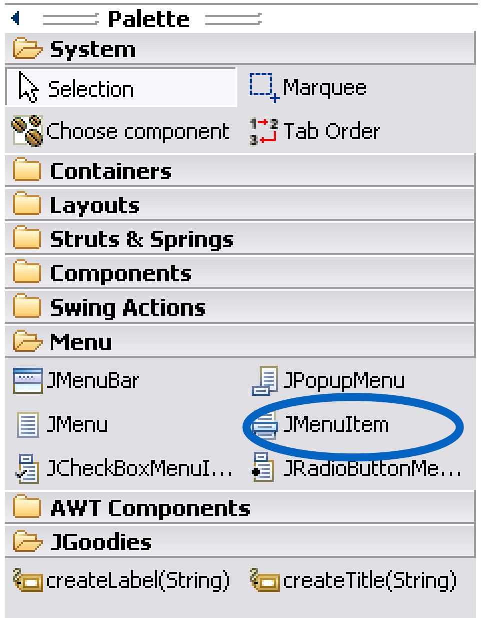
# Context Menus (II)

- Mnemonics and shortcuts must be defined **and consistent with the corresponding to the same option in the Drop-Down menu!**

- We must ensure that every option available through the context menu are also accessible by means of a more visible alternative, like drop-down menus.

# Menu Items (MenuItems)

## Composition and creation



- A menu item represents a command or option to be done (copy, cut, open, etc).
- They must be **brief and take up just one line**
- Mnemonics **must be included** in every menu item
- Shortcuts **must be offered** at least for the most frequently used options
- Shortcuts **must be consistent** with the other references to the same command.

# Menu Items

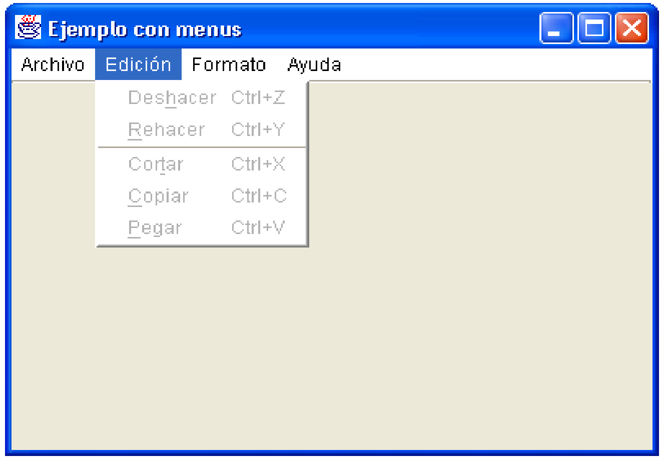
## Composition and Creation (II)

- We must use “…” to point out that the command **requires more information** to be executed.
  - Example: ***Save as…***
- We **must not** use “…” to mean that a secondary window will be shown.
  - Example: properties,  About.

# Menu Items
## Availability

- If one specific feature is not available in a window, but users can do something that could make it available, we must show the menu item as ***disabled***

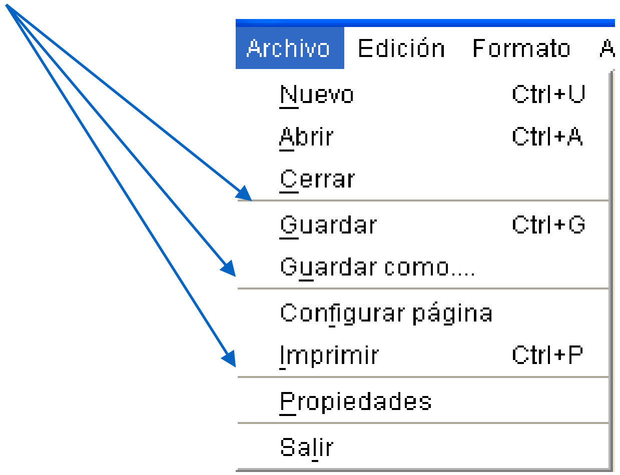
- Even when all the options of a menu are disabled, the **menu must be enabled** so the user can see all the options it contains.



# Menu Items

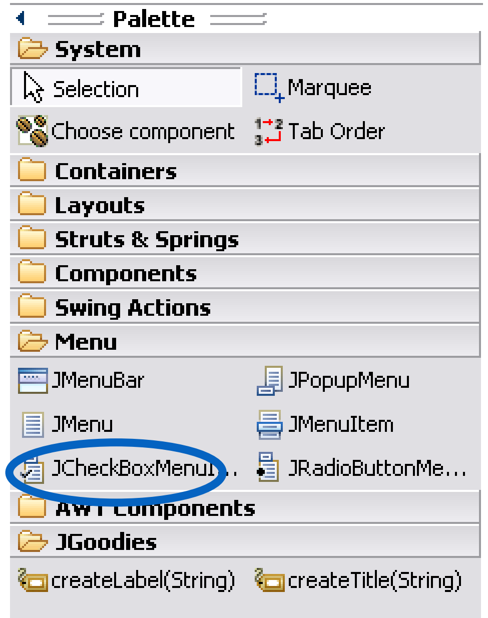
## Organization

- Separators must be used to help the users to understand and localize the different options.

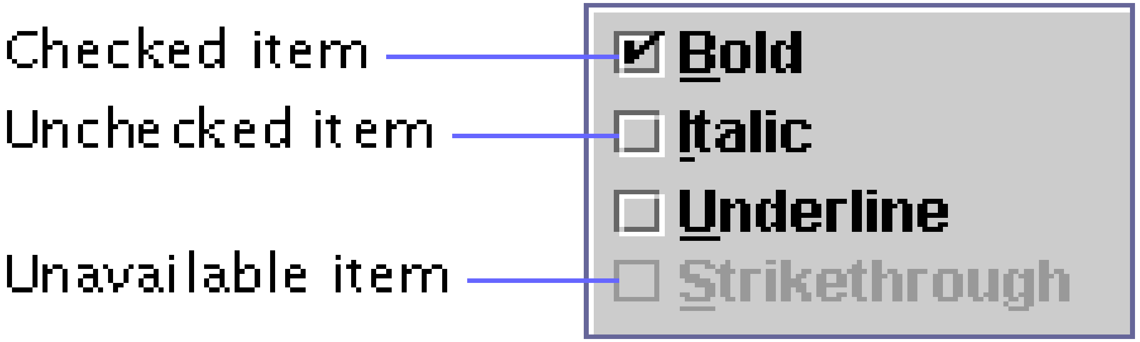


- If the number of items were to big, we must use an specific layout (*grid layout*) to show all the options in multiple columns.

# Checkbox Menu Items



- We use them to show non-exclusive options
  - They must be used carefully, given that when the user chooses and option, the menu is automatically closed, so it must be opened again if we want to choose another option.
  - So, if we predict that users will choose more that one options, using a dialog would be more interesting.

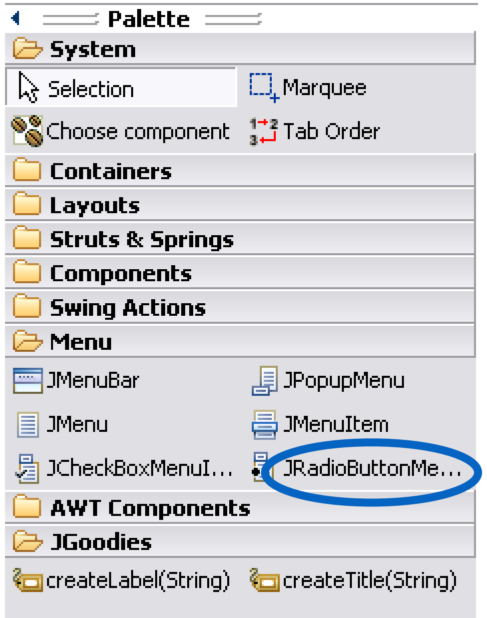


Checked item

Unchecked item

Unavailable item

# Radio Button Menu Items



- We use them to show exclusive-options
- Separators must be used in order to group the radio buttons and inform the users that they are related.
- As happens with the checkbox menu items, we must use them carefully.

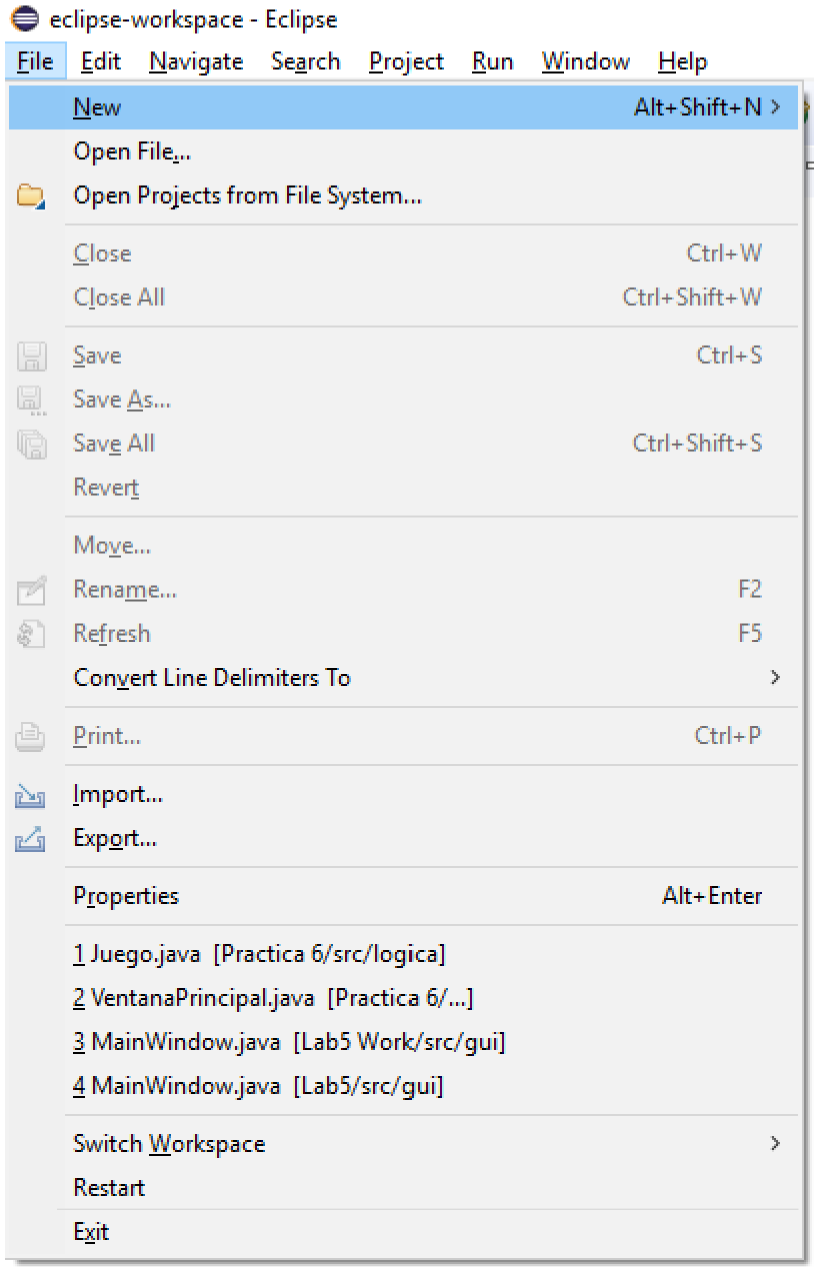
# Popular Menus Organization

- There are popular menus like File, Edit or Help that are present y most of the applications.

- If the applications uses them, the order must be:
  - *File, Edit, View, Format, […], Help*
  - The other options will be placed between *Format* and *Help*

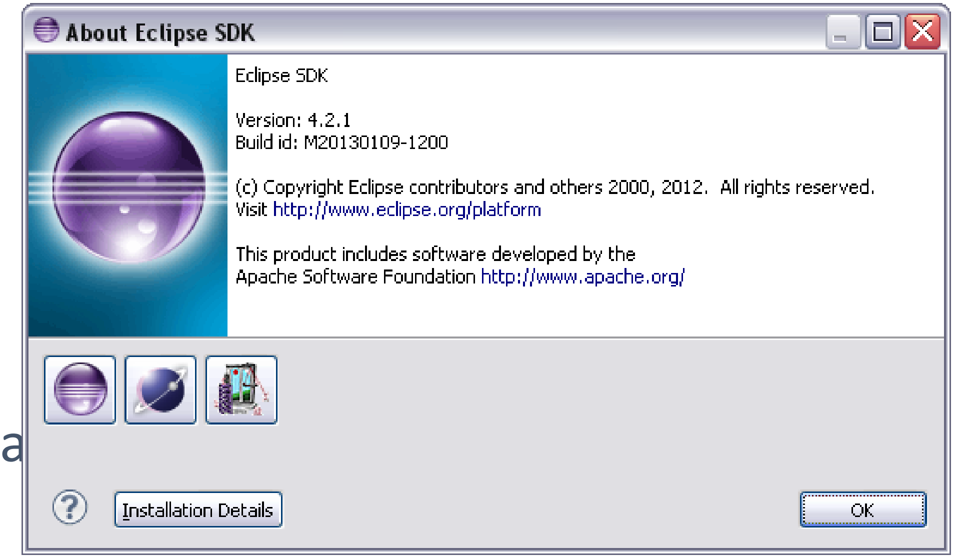
# Popular Menus
# File

- File
  - It encloses the option that can be applied to the whole document or to the application as a whole.
  - <u>If the application does not manage files</u>, we can change the name of this first option whenever the target object corresponds to the whole application. Example: Project, Game, etc.
  - The *Exit* option will always be the last the drop-down menu.
  - In case there were more options before *Exit*, **we must use a separator**



# Popular Menus
# Help

- It provides online access to information related to the application features
- It provides access to the “About” dialog that shows information about the application.
  - It includes…
    - Name of the product
    - Version
    - Company logo
    - Product logo
    - Authors
    - Etc.



Eclipse SDK

Version: 4.2.1
Build id: M20130109-1200

(c) Copyright Eclipse contributors and others 2000, 2012. All rights reserved.
Visit http://www.eclipse.org/platform

This product includes software developed by the
Apache Software Foundation http://www.apache.org/

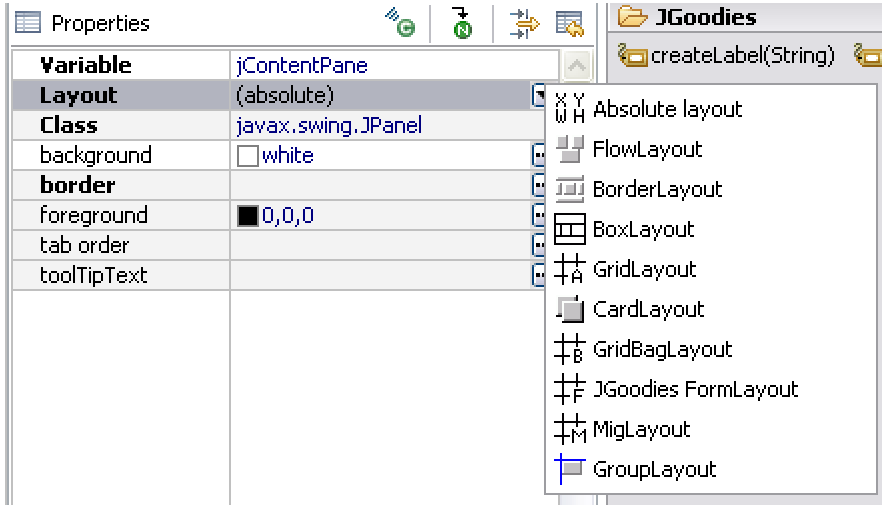
- The *About* option must be separa
  separator.

# 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



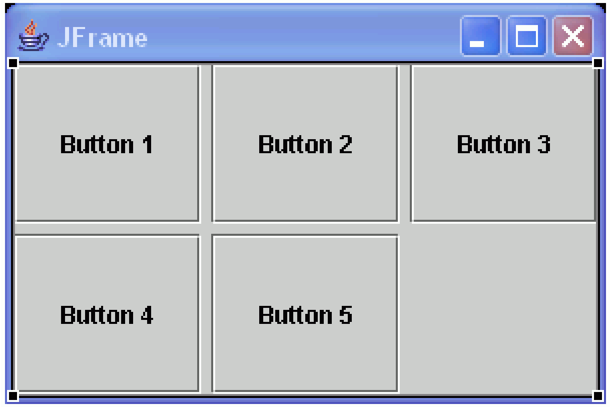
Properties

| | |
|---|---|
| **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

# GridLayout

- We configure it with a number of columns and rows.
- Components are placed I the cells sequentially (left to rigth, top to bottom).
- Cell size is homogeneous
- Every components is resized to fill all available space in the cell.
- Properties:
  - hgap, vgap
  - colums, rows



# BoxLayout

- It permits to organized the components in an horizontal or in a vertical line.
- Example: JToolBar
- Space cannot be left between components
- Properties:
  - Axis: X_AXIS, Y_AXIS