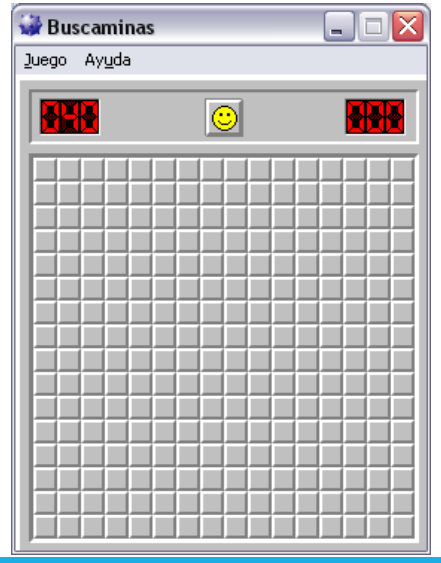
Theory pills Labs 1 & 2

Main types of containers

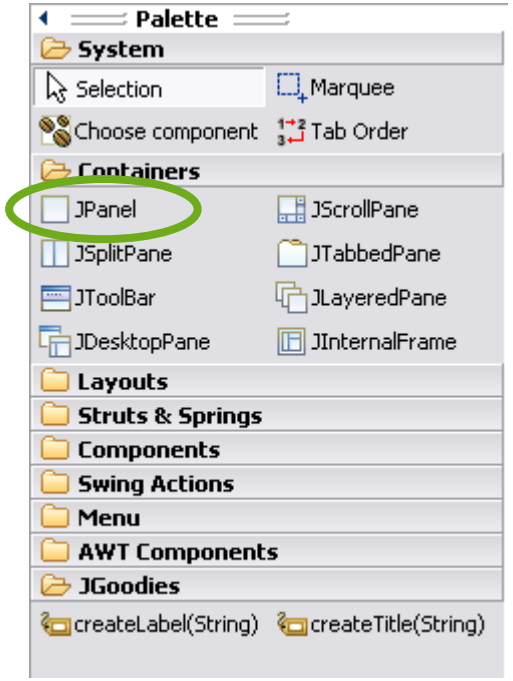
- Frame (JFrame)
- Dialog (JDialog)
- Panel (JPanel)
- Scroll Panel (JScrollPane)
- Tabbed panel (JTabbedPane)
- Tool bar (JToolBar)

Frame (JFrame)

- Window that cannot be contained by any other window. Represents the main window of the GUI.
- It has border, title, control menu, minimizing and maximizing buttons, resizing controls.
- It can contain menu bars
- The components of the window must be contained inside a Panel, never directly on the JFrame.



Panel (JPanel)



- Container that groups components inside a window or a different panel.
- *Layout managers* allow to place the components visually inside the container.
  - We can use them also to group components visually with a border (ex. Radio buttons).
- Some methods…
  - getComponentCount () : number of components in the panel
  - getComponents() : set of components included in the panel.

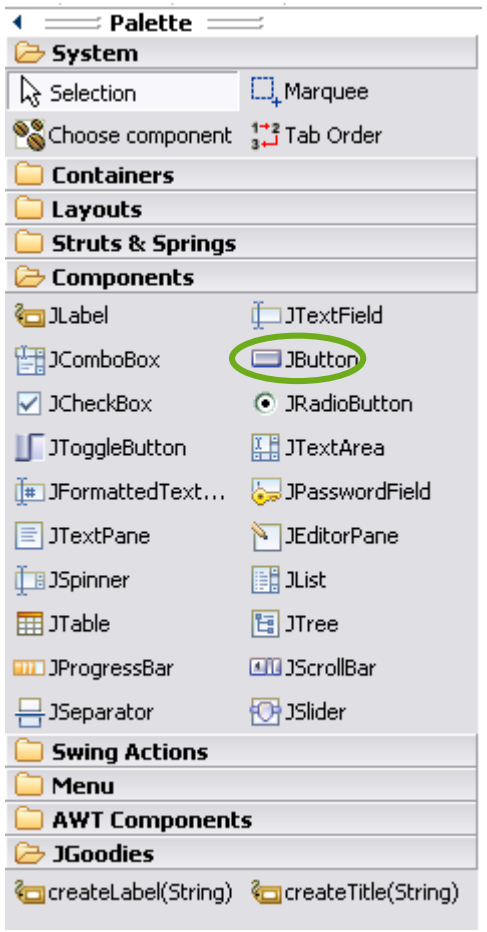
Basic components

Buttons

- *Command*
- Flip-flop (*Toggle*)
- *Check Boxes*
- *Radio Buttons*

Combo Box

Command buttons (JButton) (I)



They can contain text, graphics or both.

Generally one single word is used to identify the button.

In it has text, a mnemonic must be defined, except for the default and cancel buttons.

Those without text should have a tooltip associated describing their name or functionality.

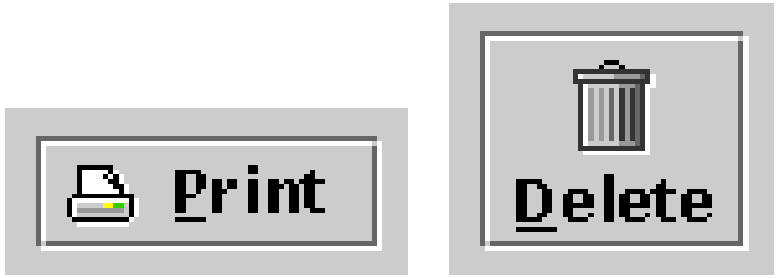
- They provide information about a component whenever the user stops on the component.
- Tooltips must be active by default, but we **must** provide the possibility of disabling them (for example, with a checkbox in the preferences window).

Command buttons(II)

The buttons containing only text must show it centered.

Buttons with text and graphics…

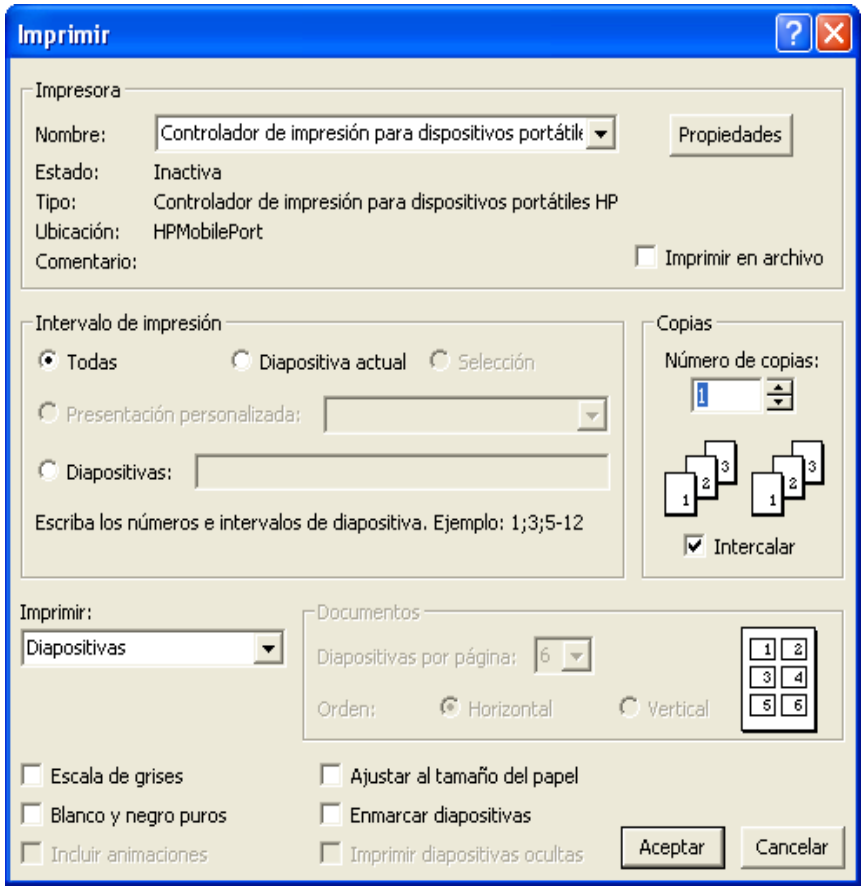
- Text must be on the right or under the graphic
- Mnemonics are necessary yet, except for default and cancel buttons



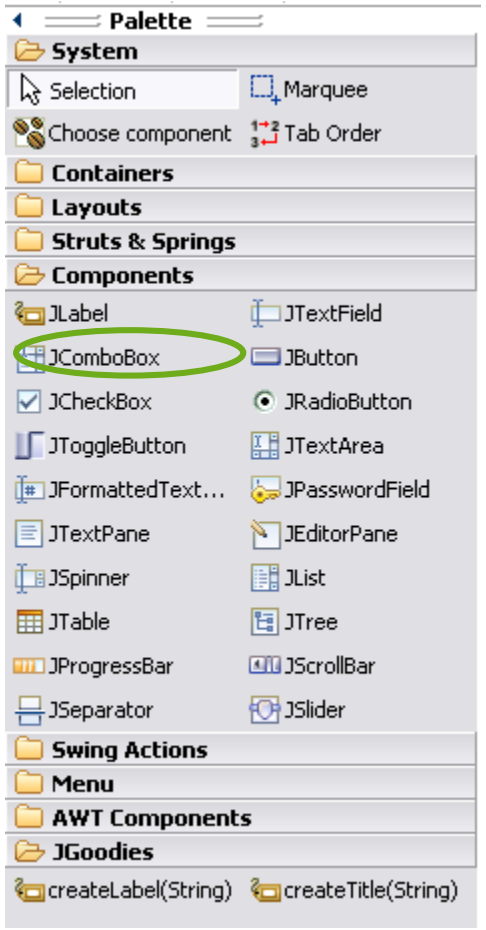
Command buttons(III)

If pressing the buttons involves interacting with another dialog, we add "…" to the text of the button.

More information is needed to complete the action



Toggle buttons (JToggleButton)



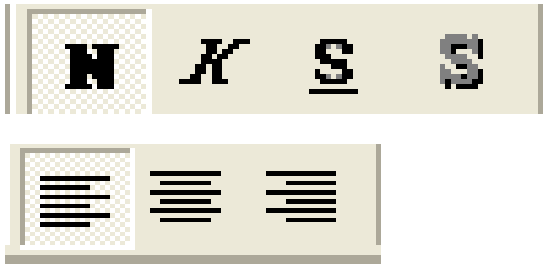
Two states: on & off.

Can have text and/or graphics.

Both text and graphics must state for both states (on/off)

They can be used to represent independent (like checkboxes) or exclusive options (like radio buttons)

Toggle buttons (II)



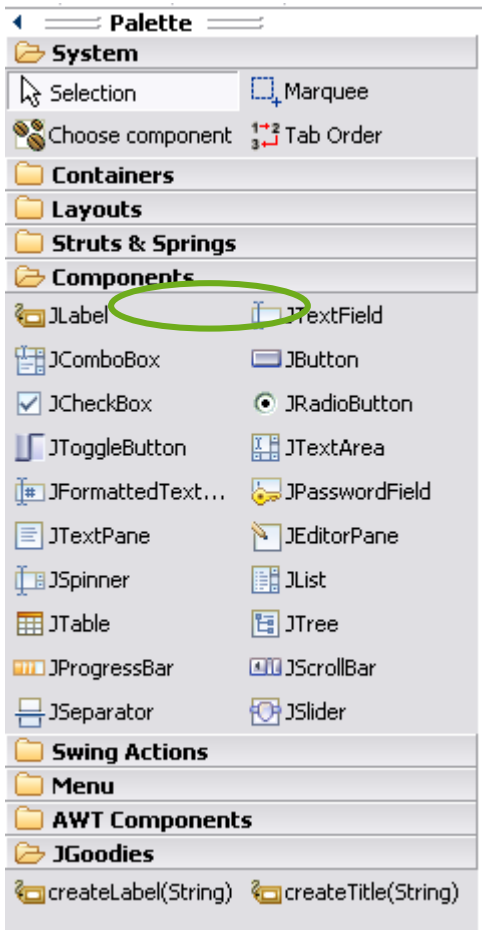
They can represent….:

- Independent options:
  - *checkboxes.*
- Exclusive options:
  - Radio buttons.
  - Both strategies are usually applied in toolbars and with the checkboxes and radio buttons inside dialogs.

Selected property:

- *true* means ON
- *false* means OFF

CheckBox (JCheckBox)

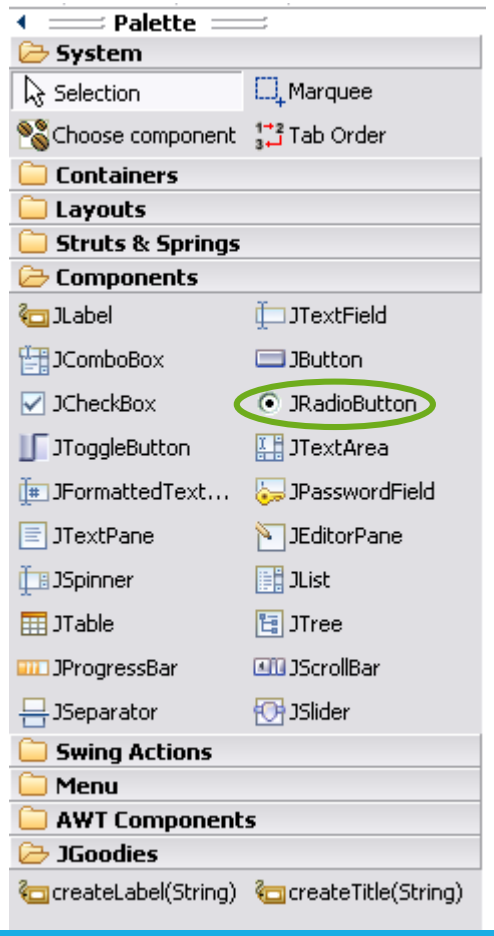


Two states: on y off

Selected property evidences if the checkbox is selected or not.

11

Radio buttons (JRadioButton)

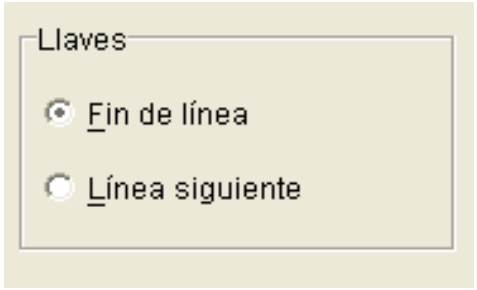


They restricts the selection to just only one of the grouped options

In java, we need a ButtonGroup

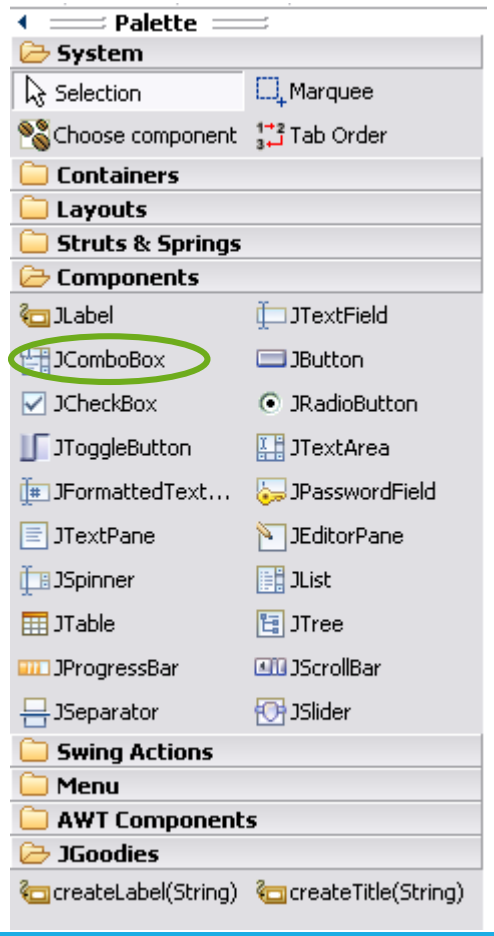
Similar to toggle buttons, but more convenient while designing dialogs

We can group them visually using titled panels.

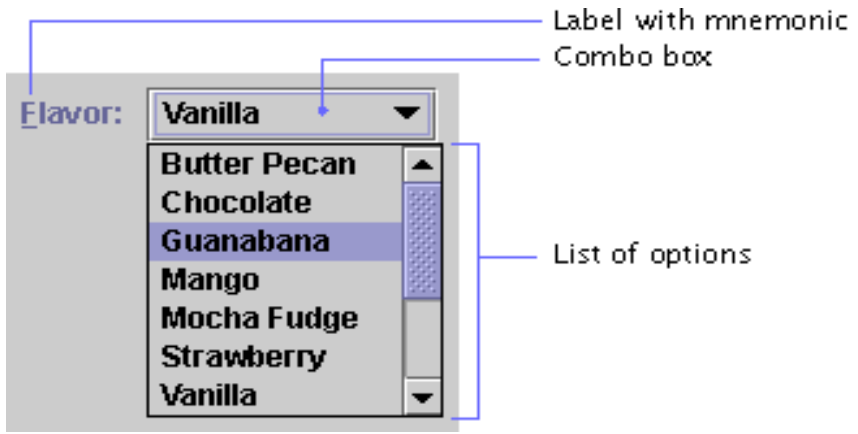


Properties and methods similar to checkboxes.

Combo Boxes (JComboBox)



To select just one option between several.



The text of the items inside the combo **must be capitalized**

Elements **must be ordered**

Mnemonics **must** be defined

Combo Boxes (II)

Not Editables

- Also called List Boxes.
- They show a list and the user can choose one element.
- We can use them instead of radio buttons when…
  - Space is limited
  - Too many options to use radio buttons

Editables

- The user can key, select or modify the text
- Can be used to help the user allowing him/her to key a value (besides selecting it from the list)

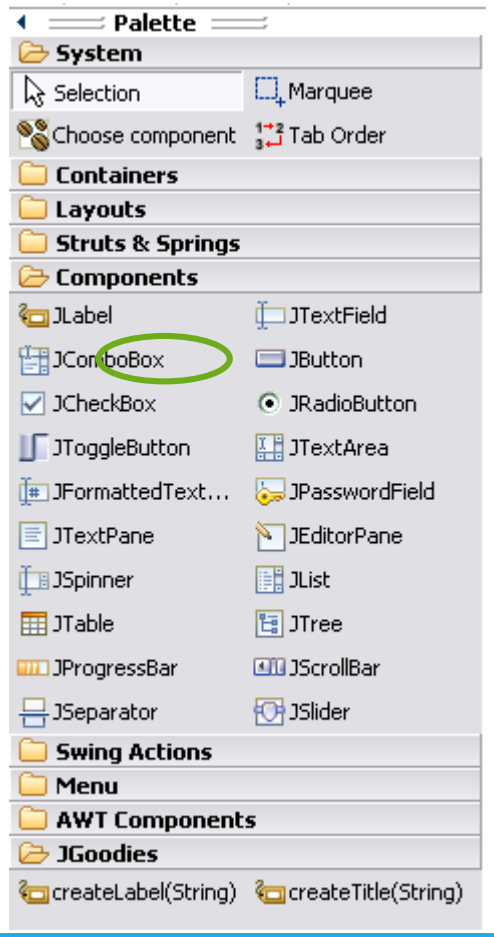
Text components

The allow the user to see and edit text

Some components…

- *JLabel*
- *JTextField*
- *JPasswordField*
- *JTextArea*

Label (JLabel)



They can contain text, graphics or both, not editable

They cannot be selected

Text **must be short,** and terminology must be familiar to users.

We can use mnemonics (*displayedMnemonic*) associating it to focus the component next to the label (*labelFor*).

The label must be inactive whenever the associated component is inactive.

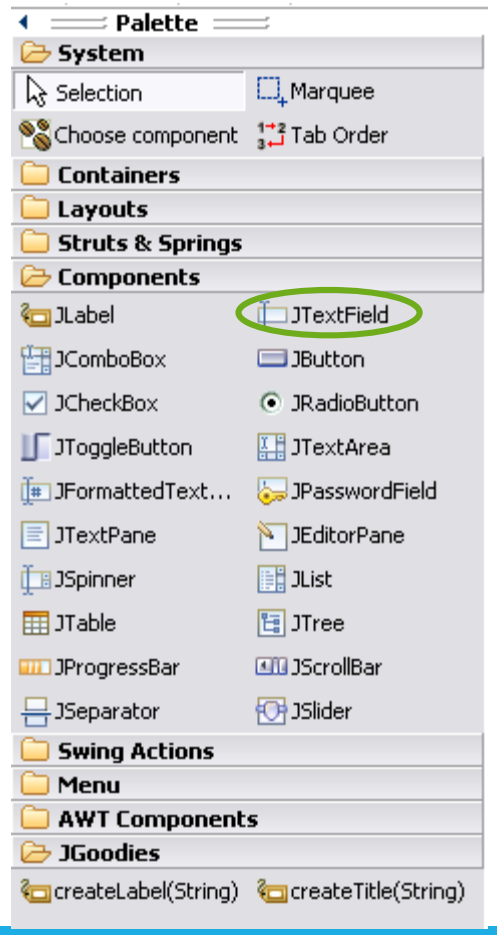
They must be before or over the component they describe

Text must be capitalized and ":" must be added at the end (when associated to a component),

Two main functions:

- Identify components
- Notify about the state or guide the users

Text Field (JTextField)



They show a text line

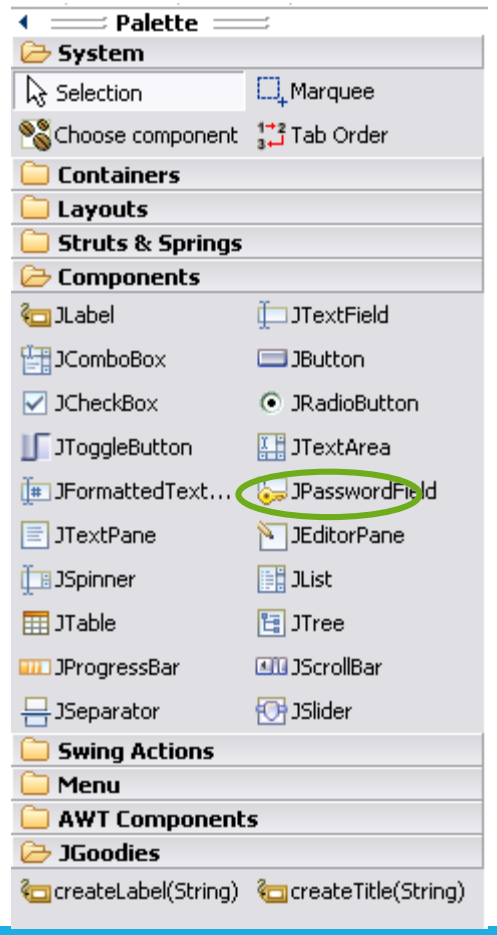
- Editable
- Not editable. Users can select and copy the text, that they cannot change it. Text can only be modified programmatically.

Mnemonics must be associated to the label.

We can associate events to…

- The user presses intro
- The focus leaves the component
- ..

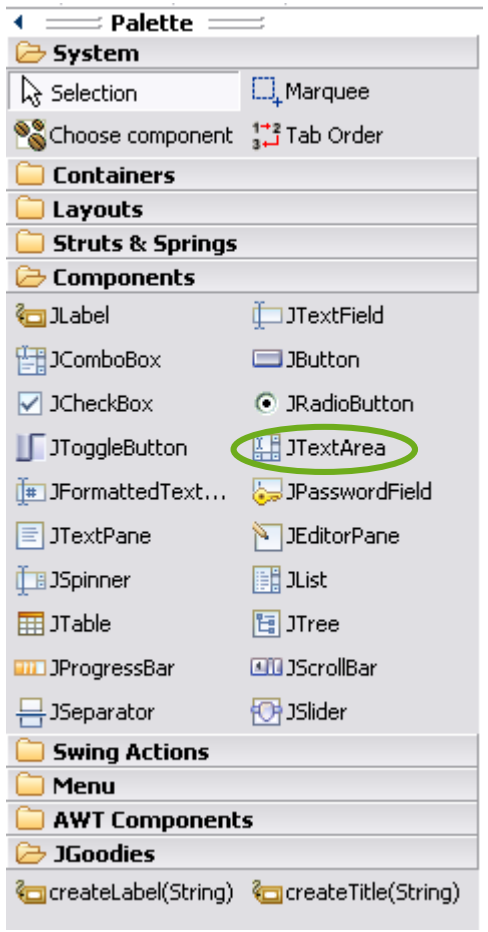
Password Field (JPasswordField)



Editable Text Field editable that shows masked characters instead of the introduced by the user.

Similar to textfield, but without copying and pasting.

Text Area (JTextArea)



Provides a space where users can see, key and edit multiple lines of text.

It has founts, size and simple style.

If we want scroll bars it must be included inside a Scroll Pane

Properties

- lineWrap->True
- wrapStyleWord-> True

Methods

- *setText* //
- *append* //
- *insert* //

That’s all folks!