

## Jigloo Swing Tutorial

In this simple tutorial you will create a small application with an "About" dialog, a menu bar, and the code to open and close the dialog. You will use the swing's GroupLayout to arrange the elements on both JFrame and JDialog, and swing Actions to control the JDialog.

### Topics covered:

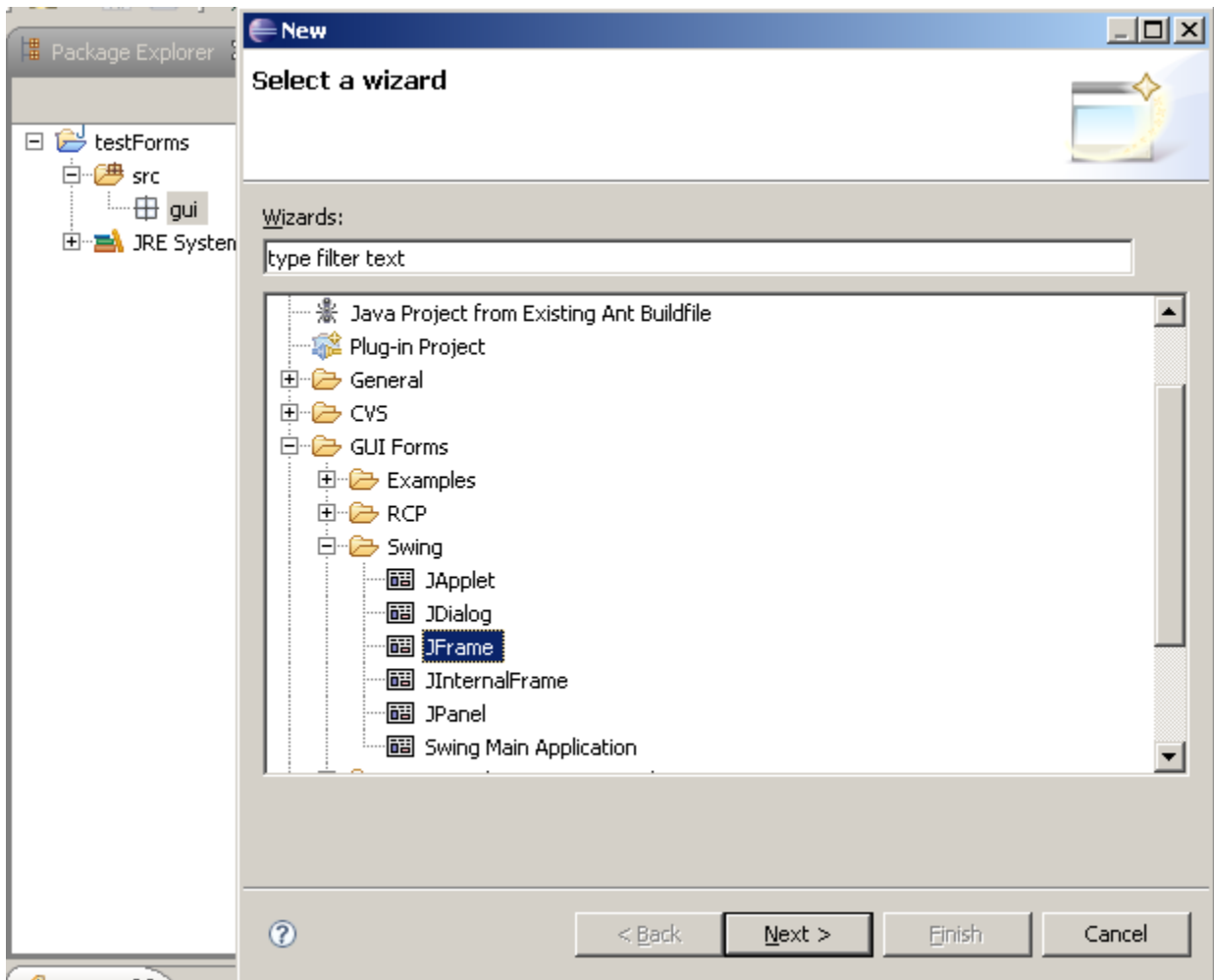
- Arranging elements using GroupLayout - anchoring and expanding by positioning or resizing or by using the drop-down menu
- Adding Actions to menu items and buttons - and associating accelerators and mnemonics with actions
- Navigating between visual elements in the tree outline and their code
- Adding multiple elements of the same type to a form
- Multi-selecting elements
- "Surround by" action - useful for times when you realise you really need your table to be in a scroll pane, etc
- Designing multiple "root" visual elements in the same form - in this case a JFrame and a JDialog
- Changing which properties appear under the "Basic" and "Expert" headings in the "GUI Properties" editor.

First you will need a Java project and (preferably) a package to hold your classes. If you don't know how to do this then hitting **Ctrl+N** is a quick way to bring up the "Create" dialog from which you can perform both these steps.

### Creating the main JFrame

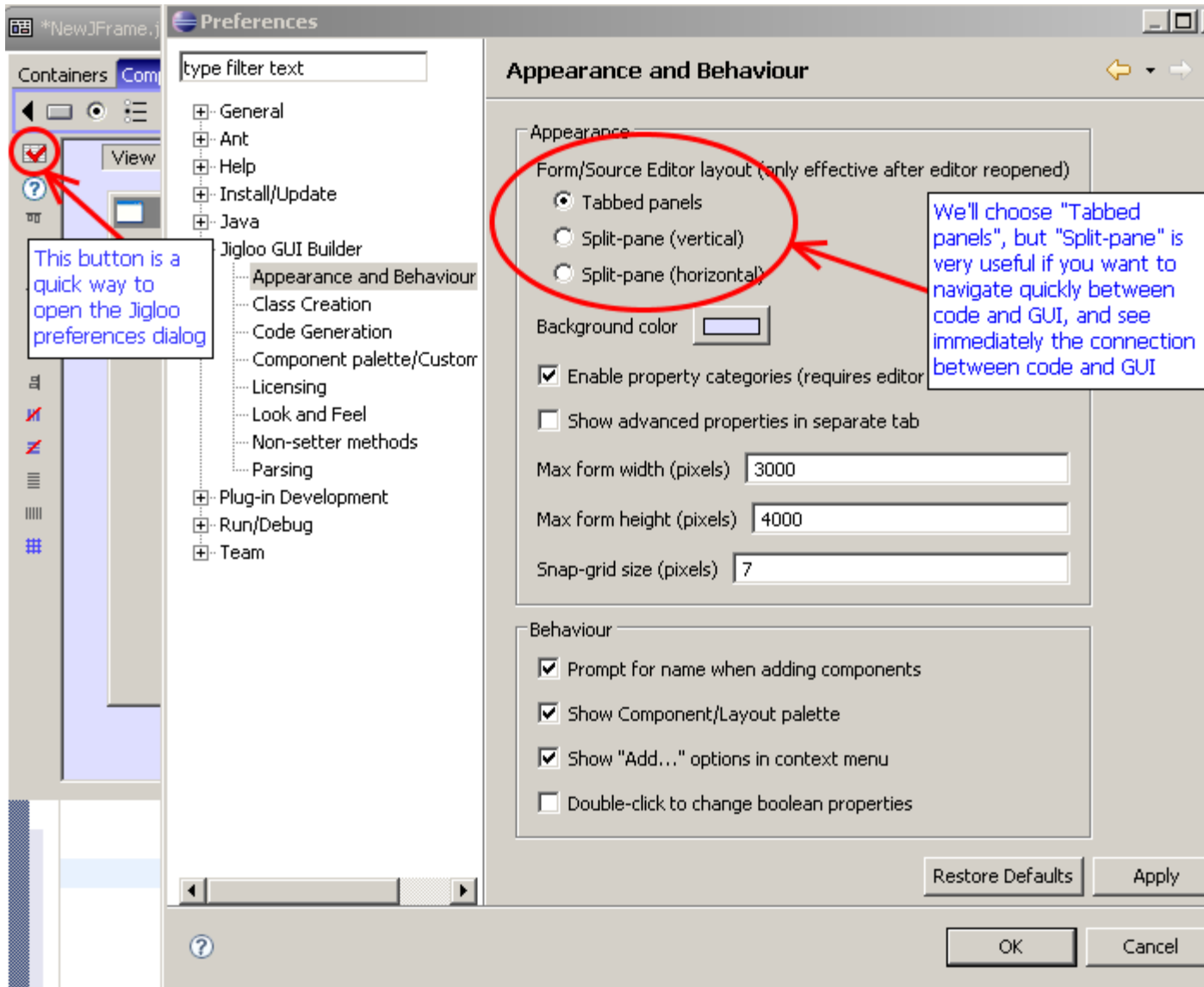
#### Create new JFrame

...again, Ctrl+N will show the "create" dialog, from which you should select "GUI forms->Swing->JFrame"

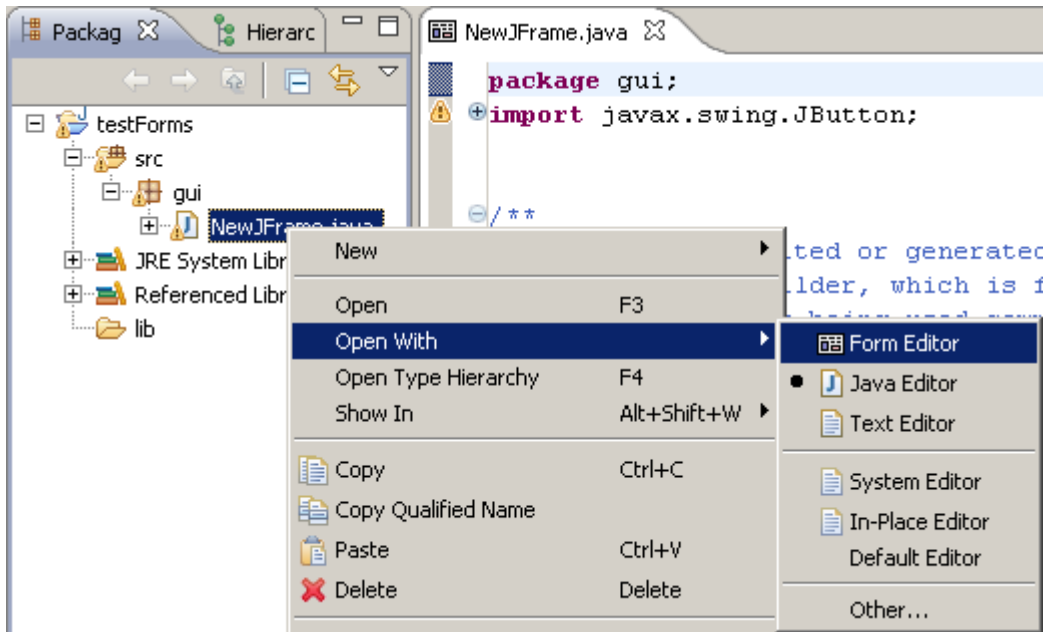


### Choose your editor preferences

Now we've got a Jigloo editor open, let's change how it looks. Click on the "Open Jigloo preferences editor" button in the toolbar to the left of the Jigloo editor. The Eclipse preferences window appears with Jigloo selected. Choose "Appearance and Behaviour" and then "Tabbed panels". This is useful when you want to maximize your design area, but "Split-pane" can be useful if you want to see immediately the connection between code and GUI.

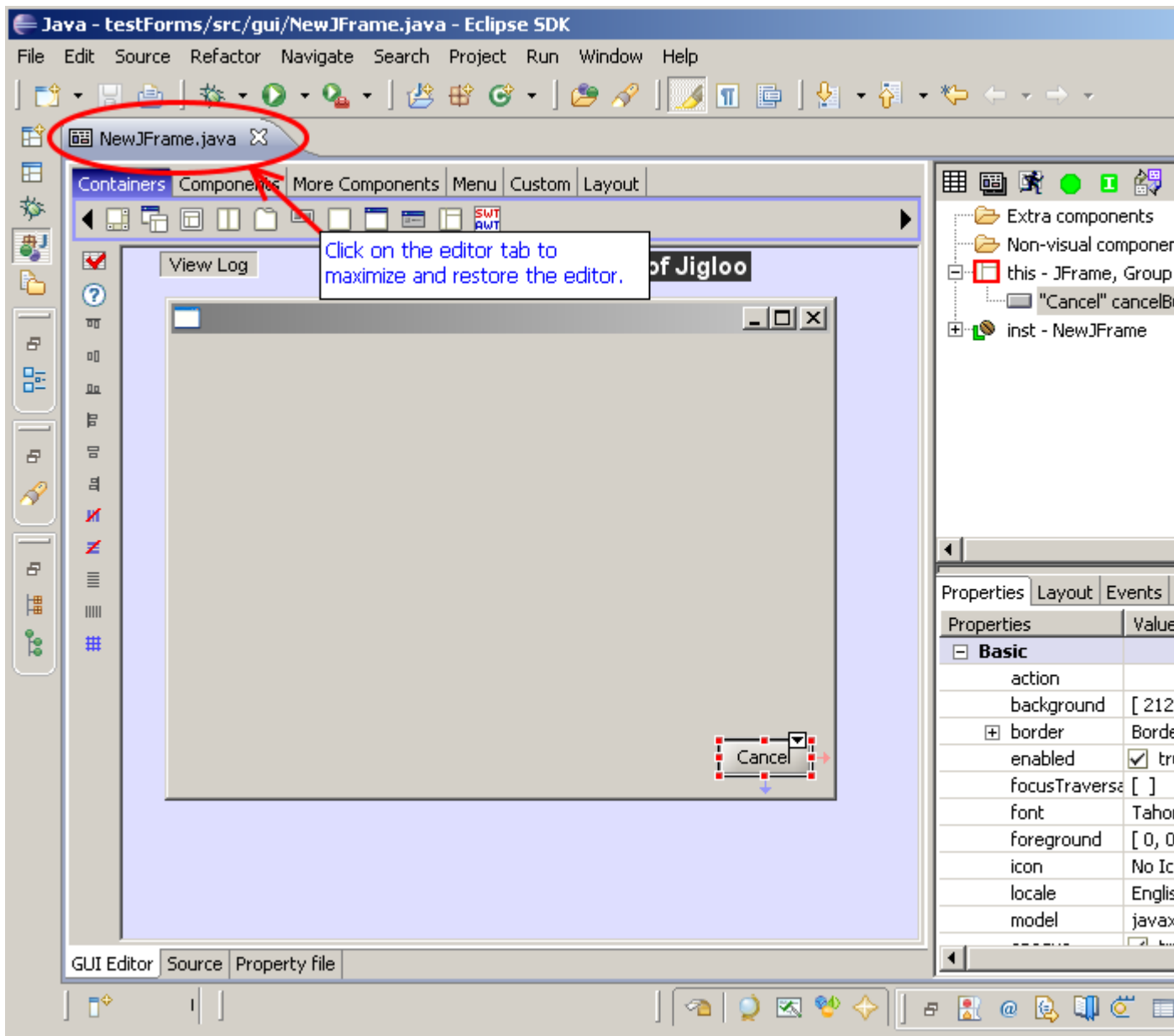


Now hit "OK" and close and re-open the Jigloo editor (you need to do this to change to tabbed panels). If your java class does not immediately re-open in the Jigloo editor, you can ensure that it uses Jigloo's Form Editor by right-clicking on the class and choosing "Open with->Form editor".



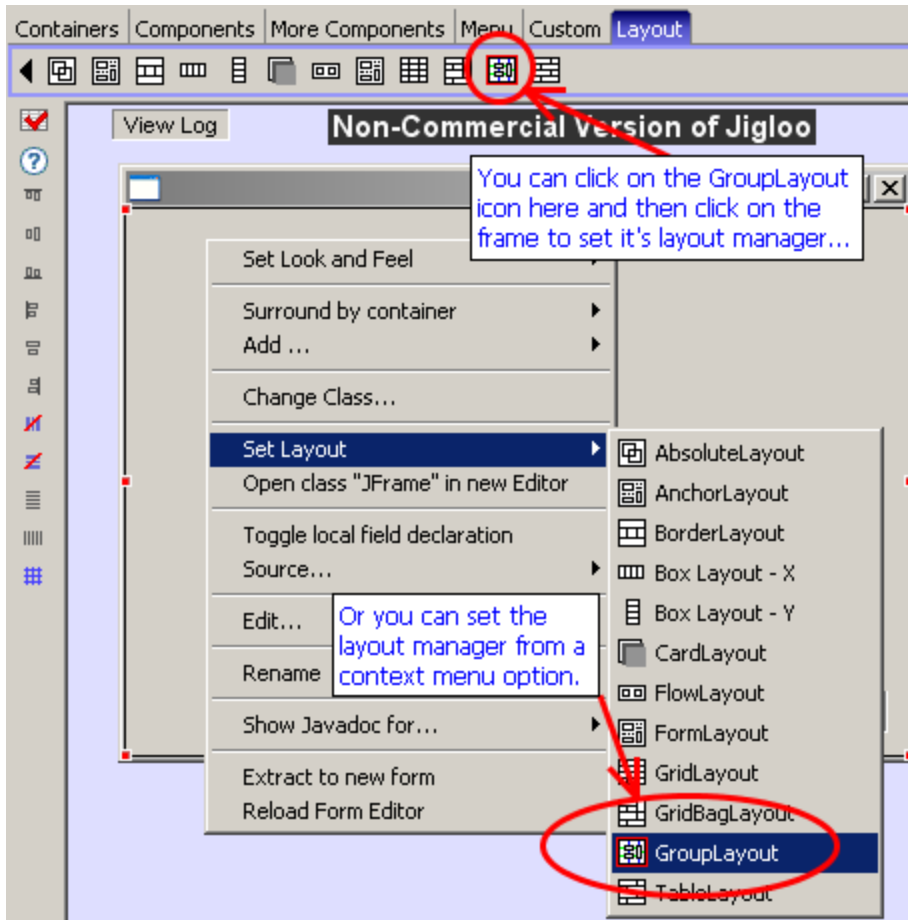
### Maximize Jigloo

You will probably find it easier to design a GUI if you can see as much of it as possible, so double-click on the editor tab and the Jigloo editor will take over most of Eclipse and arrange itself so you can see an outline of the elements, the property editor and the form designer.



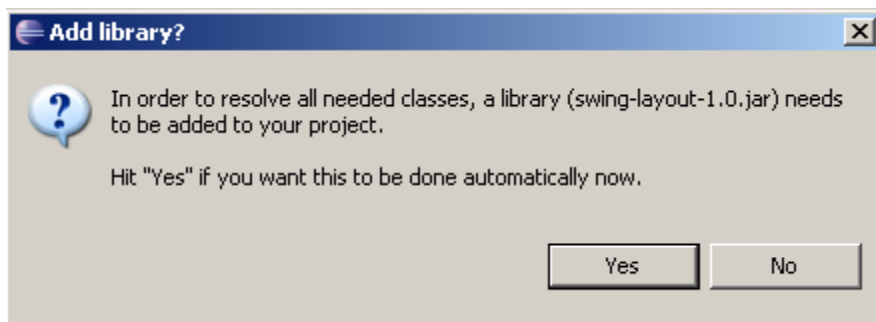
### Set layout to GroupLayout

You can either select the GroupLayout icon in the "layout" palette, or right-click on the JFrame and select "Set Layout->GroupLayout"



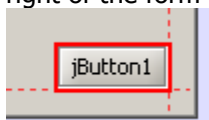
If you are using Java 6 then code will be generated for the `javax.swing.GroupLayout` which is part of Java 6.

If you are not using Java 6, Jigloo will use the `org.jdesktop.layout.GroupLayout` (which is the equivalent of the swing `GroupLayout`) and will let you know that it needs to add the `swing-layout-1.0.jar` file to your project.



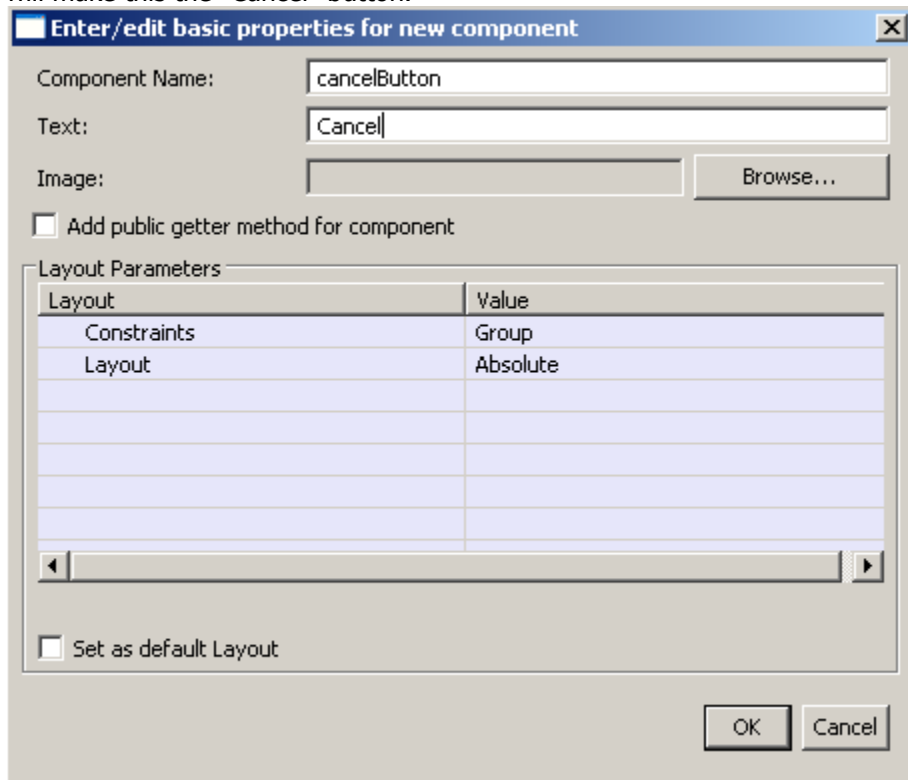
### Add OK and Cancel buttons

Select the JButton icon from the "Components" palette and move the cursor down to the bottom-right of the form until it "clicks" into place.

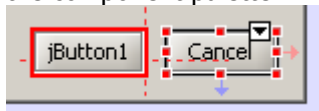


In the dialog that will open (if you have selected that option in the jigloo preferences) you will be

able to set the button's name and text (and icon if you want). You can also do all this later. We will make this the "Cancel" button.

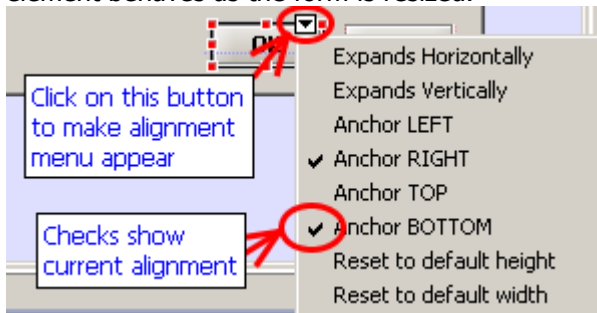


Then hold CTRL and SHIFT down and you will be able to add another button without returning to the component palette.



If you aligned the buttons with the bottom-right of the form you should see (as in the image above) a faint red arrow connecting the buttons to the right of the form and a blue arrow connecting them to the bottom of the form, indicating that they are anchored to those sides of the form.

You can also change the anchoring by either re-positioning the buttons next to edges of the form, or by using the drop-down alignment menu which you can access from the "arrow" button on the top-right of the element. Note all the menu options, which allow you to change how the element behaves as the form is resized.

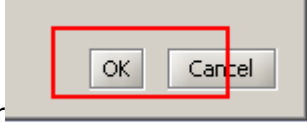


Label the buttons "OK" and "Cancel" - you can double-click on the button to edit the label, or edit

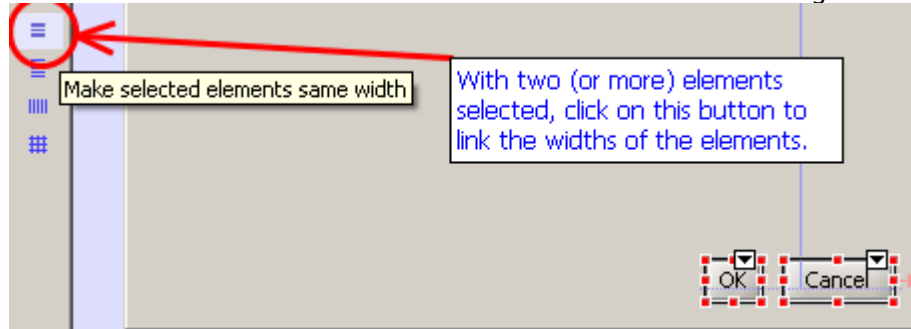
the "text" property.

### Make OK and Cancel buttons same width, and same font

Multi-select the OK and Cancel buttons - you can do this using either:

- the "rubber band" technique - hold SHIFT down then drag the mouse over both buttons,
- 
- then release the mouse, or
  - holding CTRL down as you click on the buttons

Click on the "Make selected elements the same width" icon in the alignment toolbar.

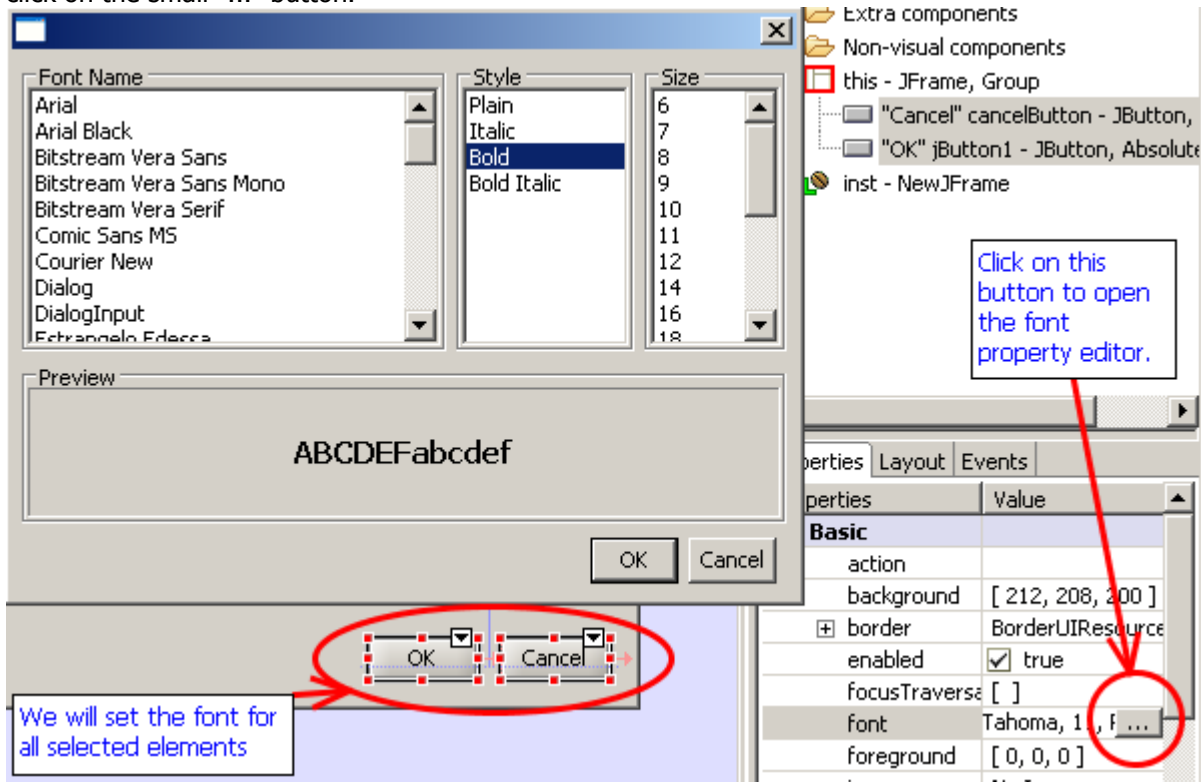


Then select the OK button - notice that now the "same widths" icon has changed to "Unlink the selected element's width from all other elements". This allows you to "unlink" an individual element after it has been linked.

Now multi-select both buttons again and (just for fun) change the font to "bold" by clicking on the "font" property in the "GUI Properties" view and using the dialog that will appear after you



click on the small "..." button.

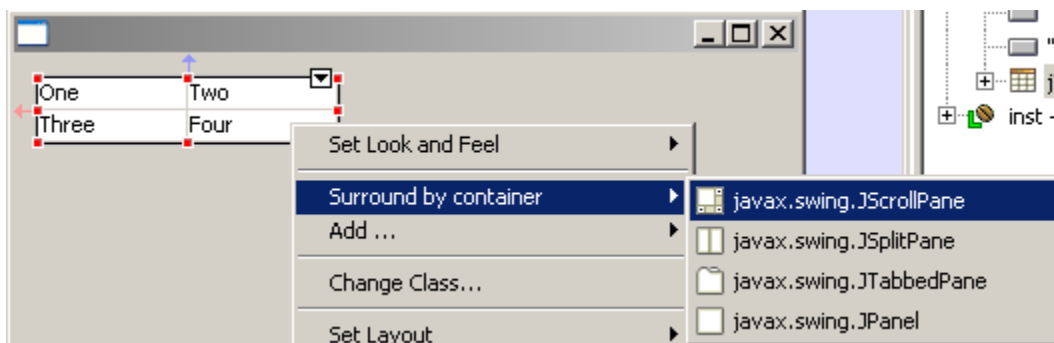


### Add JTable

using the "Components" palette - add it to the top-left of the form.

### Surround by JScrollPane and stretch it across the form.

Oops - you just remembered you really meant to add a JScrollPane first, for the table to go inside, but instead of undoing what you did, you can right-click on the JTable and choose "Surround by->JScrollPane".



Now select the JScrollPane (either in the outline or by clicking on its scrollbars, or its edges - if a scrollpane's child element occupies all of the area of a scrollpane and there are no scrollbars, clicking near the edge of the scrollpane will select the scrollpane). Then drag the side or corner of the JScrollPane so that it stretches from the left to the right of the form and is a bit taller. Again, red and blue arrows should indicate that it is connected to both the left and right sides of the form.

If you open the drop-down menu you will see that "Expands Horizontally" is checked, and as

before you can use that menu to change any of the listed properties. Note that you will need to uncheck "Expands horizontally" in order to be able to anchor it left or right.

### Add two JTextField (using quick-repeat) and JButton

Just to get a feel for GroupLayout, add two JTextFields and a JButton, then select them all and surround by a JPanel and set the JPanel's layout to GroupLayout. You can resize the panel and move/resize the elements till they look similar to what is shown below.

Then edit the JPanel's "Border" property to create a titled border with some text.

The screenshot shows the NetBeans IDE interface. The main window displays a GUI design with a table and a group of components. The table has two columns, "Column 1" and "Column 2", and two rows, "One" and "Three". Below the table is a group of components labeled "group one" containing two text fields, "jTextField2" and "jTextField1", and a button, "jButton2". A red circle highlights the "group one" components, and a text box with an arrow pointing to it says "Added some elements then surrounded them with a JPanel and set the JPanel's layout to GroupLayout." Another text box with an arrow pointing to the "border" property in the Properties window says "Set the JPanel's border to 'TitledBorder' and then set the title to 'group one'". The Properties window shows the "Basic" tab with the following properties:

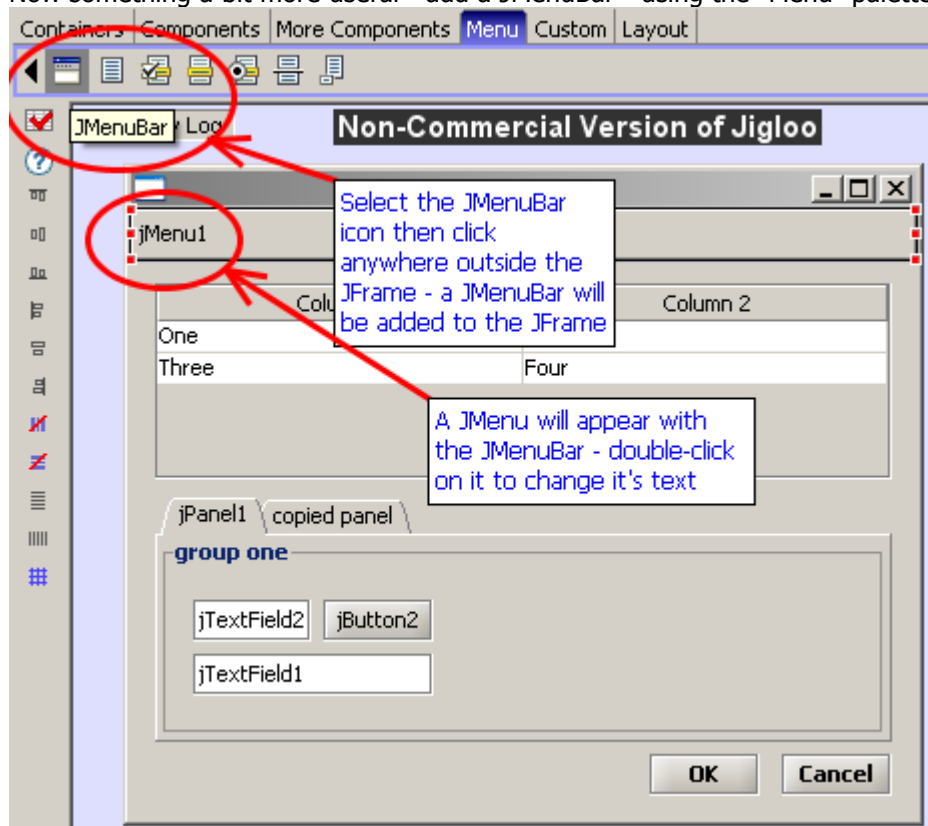
Properties	Value
background	[ 0, 0, 0 ]
border*	TitledBorder
border	No Border
title*	group one
titleColor	[ 10, 36, 106 ]
titleFont	Tahoma, 11, Bold
titleJustificati	LEADING
titlePosition	TOP
enabled	<input type="checkbox"/> false
focusTraversalPc	[ ]

### Surround JPanel by JTabbedPane then copy and paste the JPanel into the JTabbedPane

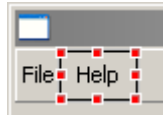
This just demonstrates copy and paste - or you could try cutting and pasting multiple components too

### Add JMenuBar with File and Help menus

Now something a bit more useful - add a JMenuBar - using the "Menu" palette.

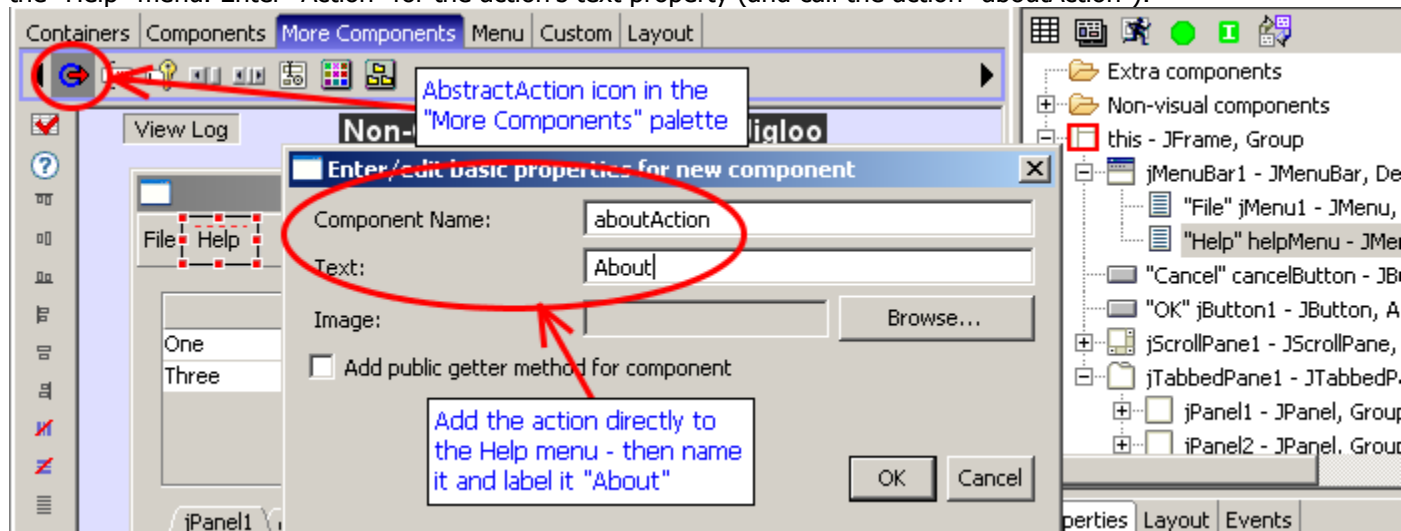


Now double-click on the "jMenu1" menu and change the text to "File", then add a new "Help" JMenu to the menu bar.

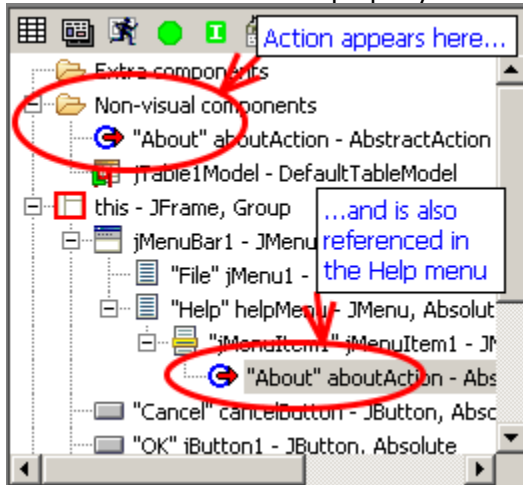


### Add an "About" action (and menu item) to the Help menu

Select the "AbstractAction" icon from the "More components" palette, and then click directly on the "Help" menu. Enter "Action" for the action's text property (and call the action "aboutAction").



If you look at the Outline view you will see that a JMenuItem has been added to the Help menu and the aboutAction has been associated with it. The JMenuItem's text will also take on the value of the AbstractAction's text property.

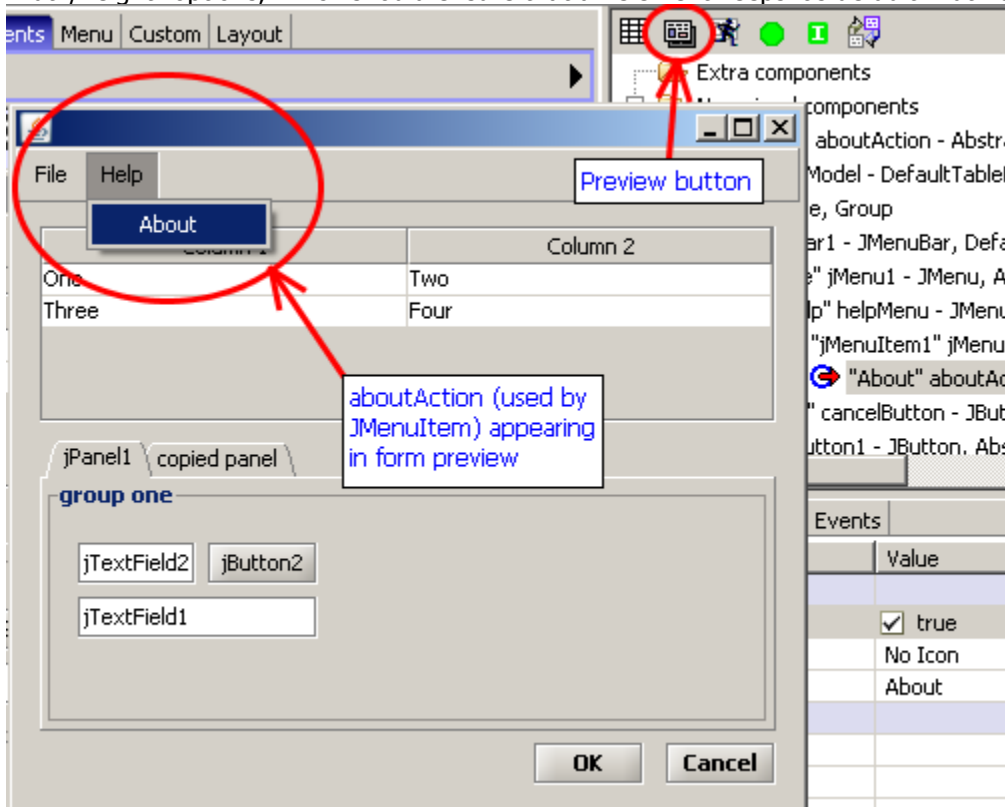


### Test out the form using the "Preview" button

Hit the "Preview" button in the Outline view and click on the "Help" menu - the "About" item should show itself.

You can also resize the form and check out how it behaves.

If an element expands horizontally or vertically but you would rather it not, then make sure the appropriate "Expands" option is not selected, and then also try the "Reset to default width/height" options, which should ensure that an element keeps it's default width or height.

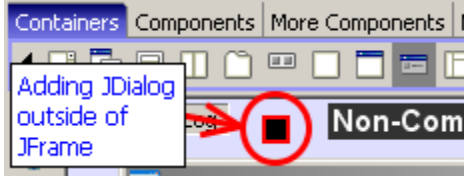


## Creating the "About" dialog and linking it to the main frame

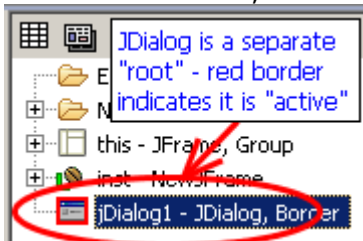
### Add JDialog

Either right-click on a region outside of the main JFrame and choose "Add container->JDialog" or choose the JDialog icon from the "Containers" menu and click somewhere outside of the main JFrame.

This will add a totally separate JDialog to the class.



Note the JDialog node on the Outline view. The red border indicates that this element is the "active" root element, and is being shown in the form editor.

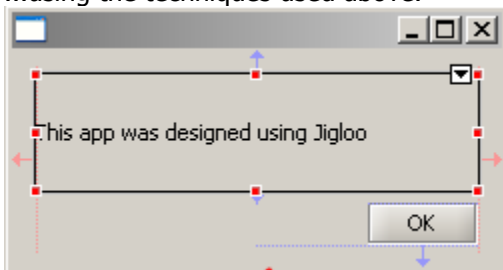


Try clicking on the "this" node - the main form will appear. Then click on the "jDialog1" node and get back to the JDialog.

This is how you can use Jigloo to design multiple "root" elements in the same Java class.

### Set GroupLayout and add label and OK button

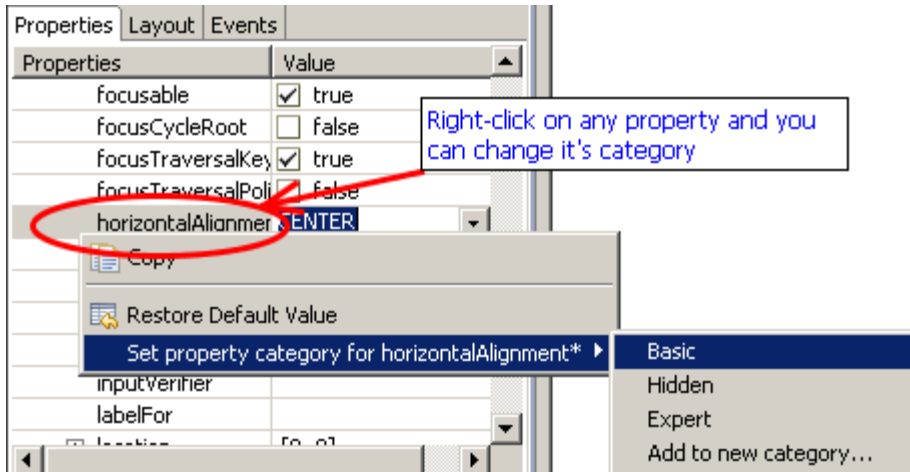
...using the techniques used above.



### Set horizontalAlignment to CENTER (and make horizontalAlignment a Basic property)

Select the label, and in the "GUI Properties" editor, open the "Expert" node and scroll to "horizontalAlignment". Change this to "CENTER".

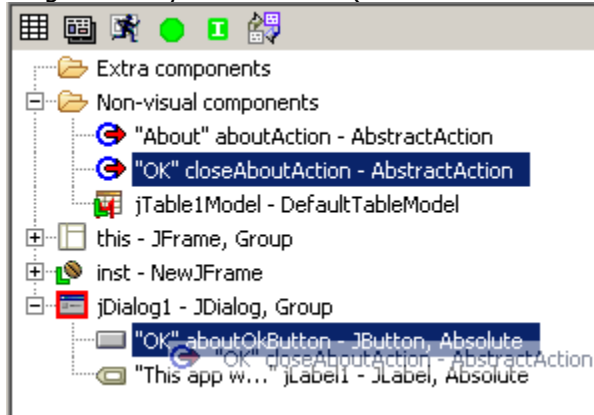
Now, if you think you will be using this property fairly often then right-click on it and choose "Set property category for horizontalAlignment->Basic".



The horizontalAlignment property (for all elements that have it) will now appear under the "Basic" property node.

### Add action to dialog's OK button with code to dispose dialog

Now, add a "Close" action to the form - again, choose "AbstractAction" from the "More Components" palette but this time (for a change) click anywhere on the editor outside of the JDialog. Set the text to "OK" and the name to closeAboutAction. Then go to the Outline view and drag the newly-added action (under the "Extra components" node) to the dialog's "OK" button.



The action should now appear twice - once under the "Extra components" node and once under the dialog's "OK" button. If you wish to disassociate the action with the OK button, just select the action under the button and hit the "delete" key. A prompt dialog will ask you if you really want to delete the action but it will really just remove the action from the button. If you really want to delete the action, select it first under the "Extra components" node.

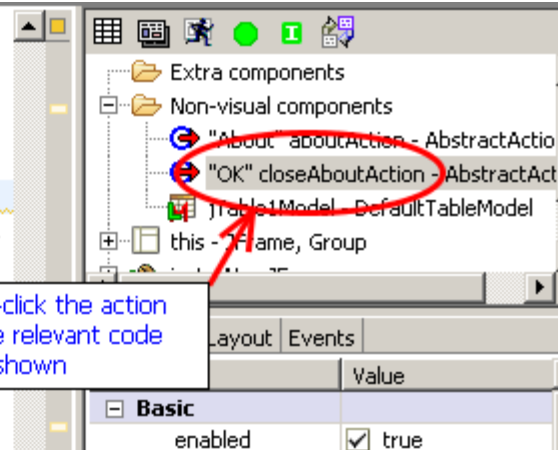
Now, find the close action under the OK button in the Outline view and double-click on it. The source code will appear and the close action's code will be highlighted.

```

return jDialog1;
}

private AbstractAction getCloseAboutAction() {
    if (closeAboutAction == null) {
        closeAboutAction = new AbstractAction("OK",
            public void actionPerformed(ActionEvent
                getJDialog1().dispose());
    }
};
}
return closeAboutAction;
}
}

```



Edit the code to add  
`getJDialog1().dispose()`  
as shown, so that the dialog will close when the OK button is hit.

### Add code to open dialog from the aboutAction

But how will the dialog appear in the first place? We will add code to the aboutAction, so double-click on the aboutAction in the Outline view. Then add the code:

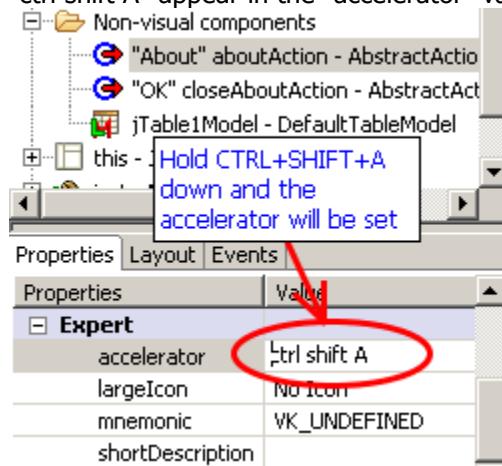
```

getJDialog1().pack();
getJDialog1().setLocationRelativeTo(null);
getJDialog1().setVisible(true);

```

### Add an accelerator for the "About" action

We want people to be able to easily see what this application is about, right? Well, maybe not, but for the purposes of this demo we will associate an "accelerator" with the about action. Select the about action, and in the GUI Properties editor, select the "accelerator" property - it will be under the "Expert" node, but you might want to move it to the "Basic" node, as above with the horizontalAlignment property. Then hold CTRL+SHIFT+A down and you should see "ctrl shift A" appear in the "accelerator" value.

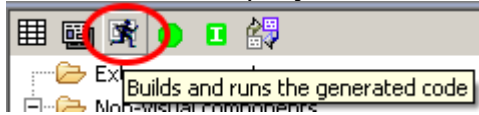


Then release the keys and hit "return" and the code will be updated.

You can also associate a mnemonic with the action in a similar way - a mnemonic allows a user to select a menu item by hitting a key (say, A in this case), but only when the parent menu is visible.

### Running the app

Congratulations! You are all done! Hit CTRL+S to save the form. A quick way to run the main method of the class you just created is to click the "Run" button in the Outline view.



The main frame should appear in the middle of the screen.

Note: If elements do not appear to be a "comfortable" size when the app is run, try selecting them in the Jigloo editor and choosing "Reset to default width/height" from the alignment menu.

Try hitting CTRL+SHIFT+A - the about menu should appear - woo hoo!  
And when your curiosity has been thoroughly satisfied you can click the "OK" button, causing the dialog to disappear.

Note: if you get an "java.lang.UnsupportedClassVersionError: Bad version number in .class file" message, make sure your compiler compliance level (set using the properties dialog for your project) is the same as the version of Java you are running your class with.

