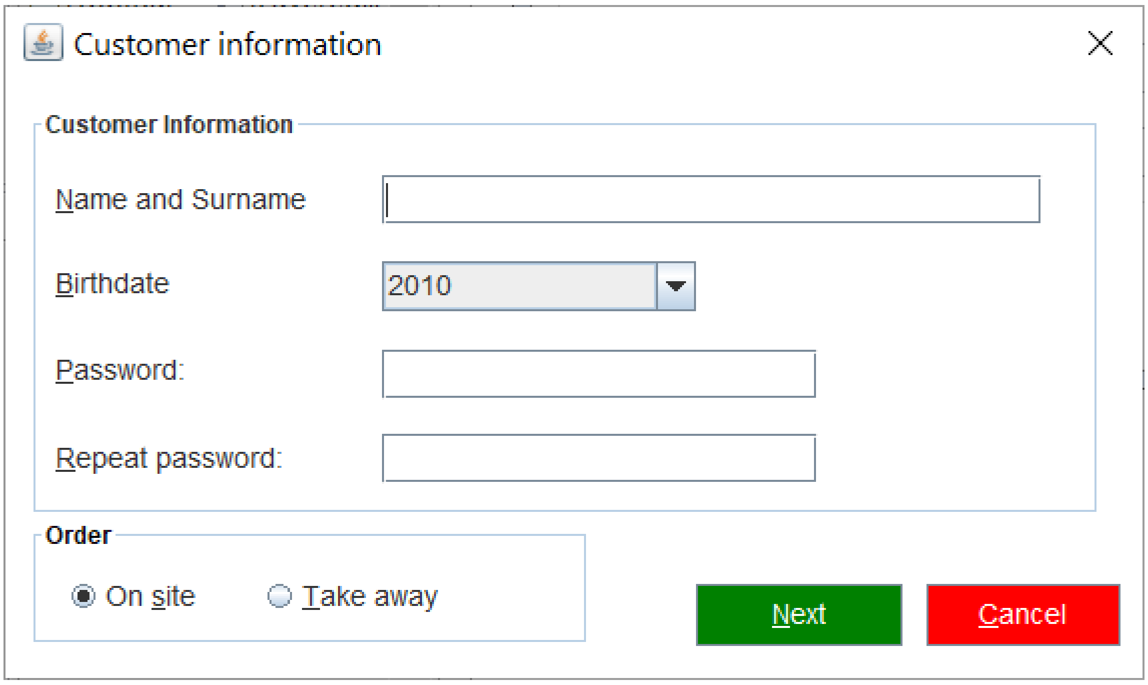
UNIVERSIDAD DE OVIEDO

ESCUELA DE INGENIERÍA INFORMÁTICA DE OVIEDO

Human-Computer Interaction

Lab 2

Creating an application with GUI using Visual Editor for Eclipse



1. Introduction

In this lab we will build a new application with a user register form. The following components will be used:

- Labels (JLabel), Buttons (JButton), Text fields (JTextField), Radio buttons (JRadioButton)
- Comboboxes (JCombobox)
- Password text fields (JPasswordField)
- Panels (JPanel)

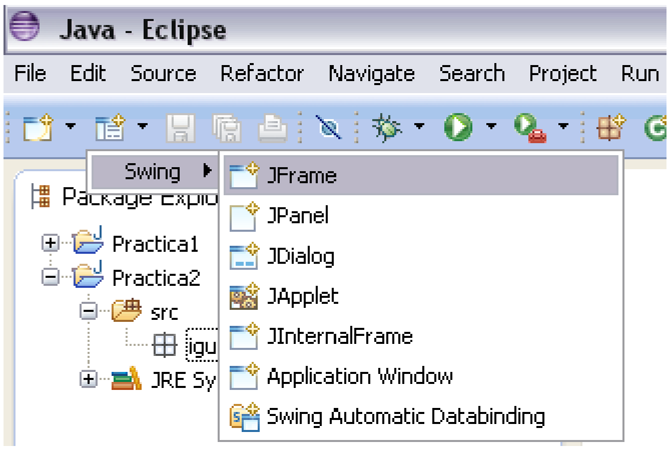
2. Creating the project in Window Builder

2.1. Creating the project

Name it *Lab2*

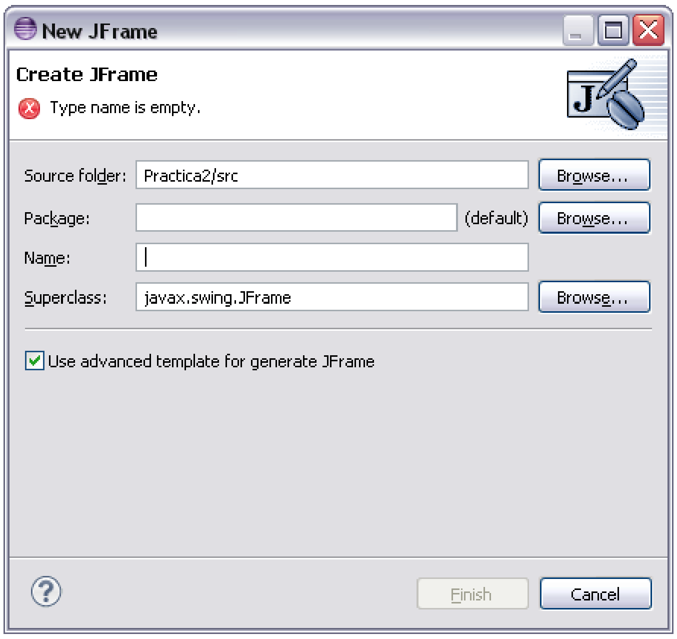
2.2. Add the visual class

Select *File-New-WindowBuilder-Swing Designer-JFrame:*



In the next dialog, define the new visual class:

1. Insert *ou.cpm.p1.ui* in the field package
2. Insert a name for the class. We will call it *RegistryForm*



Type name is empty.

Source folder: Practica2/src

Package: (default)

Name:

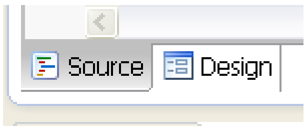
Superclass: javax.swing.JFrame

Use advanced template for generate JFrame [✓]

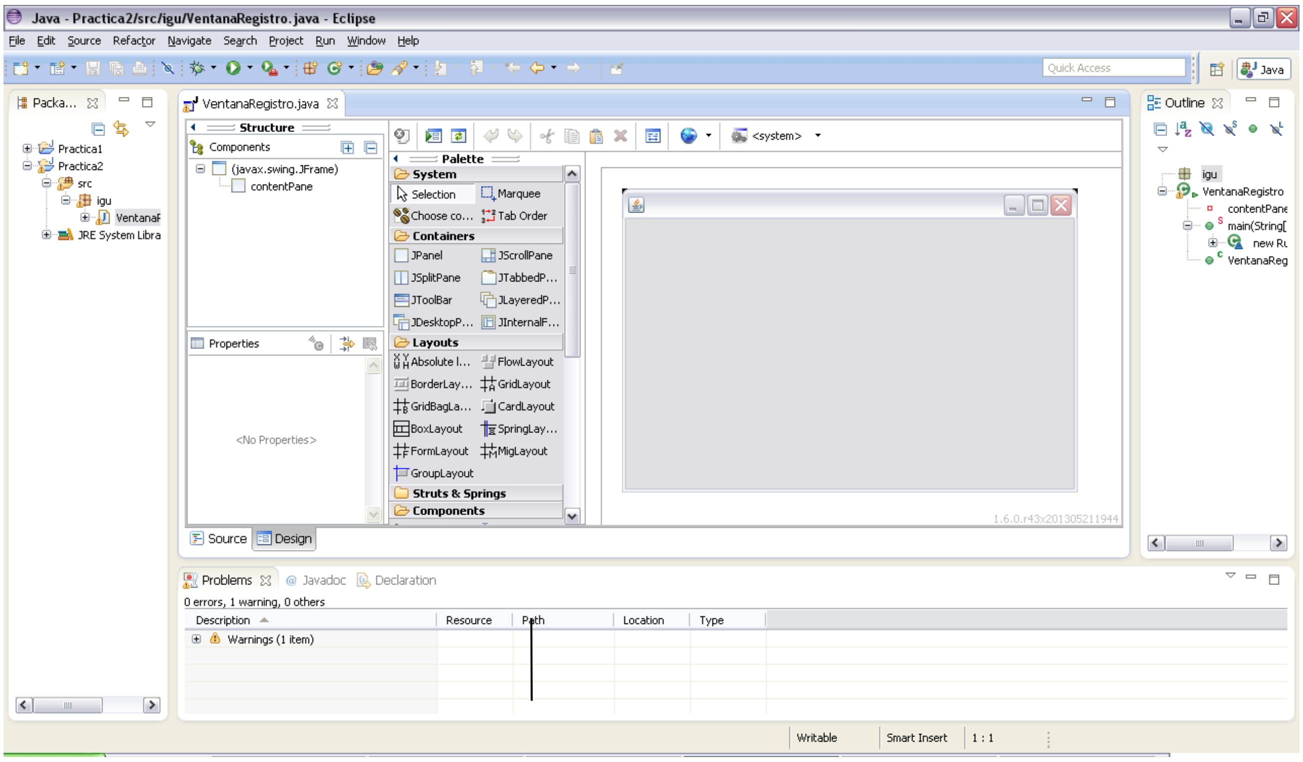
Once finished the definition of the class, Eclipse shows the code generated. This includes the following items:

- The main container as an attribute of the class (contentPane)
- In the main method, a new window is created and set visible.
- In the constructor of the window, the default action to be done by default when users press on the closing icon is set to close. Also the size of the window. The window is created, resized, the layout is set and finally, window and panel are connected.

The rest of the components will be added using the WindowBuilder visual editor. For that, we select the *Design* tab, located in the lower part of the code panel.



The aspect of Eclipse shold be like this:

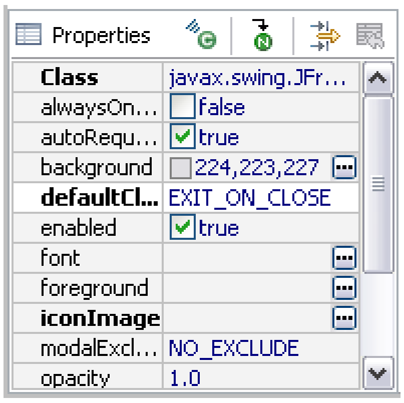


3. Modifying the default values

As shown, Window builder already included automatically a window or frame with a container (contentPane) in order to include inside every component that will make up the interface. We will modify some of these default values both in the panel and in the frame.

3.1. RegistryForm

Select the frame or window in the design panel clicking on it over the title bar. Notice that the property tab shows now the values of the attributes related to the selected JFrame object. Through this tab the values of these attributes can be modified.


Properties

Class	javax.swing.JFr...
alwaysOn...	☐ false
autoRequ...	✓ true
background	224,223,227
defaultCl...	EXIT_ON_CLOSE
enabled	✓ true
font	
foreground	
iconImage	
modalExcl...	NO_EXCLUDE
opacity	1.0

We modify:

1. ***Title***: we set it to **Customer Information**
2. ***size***: set the new size as 600,300. This can also be done "*stretching"* the window with the controls located on its borders. Previously prepare the WindowBuilder panels in order to be able to see the window with its new dimension.

3.2. Content pane

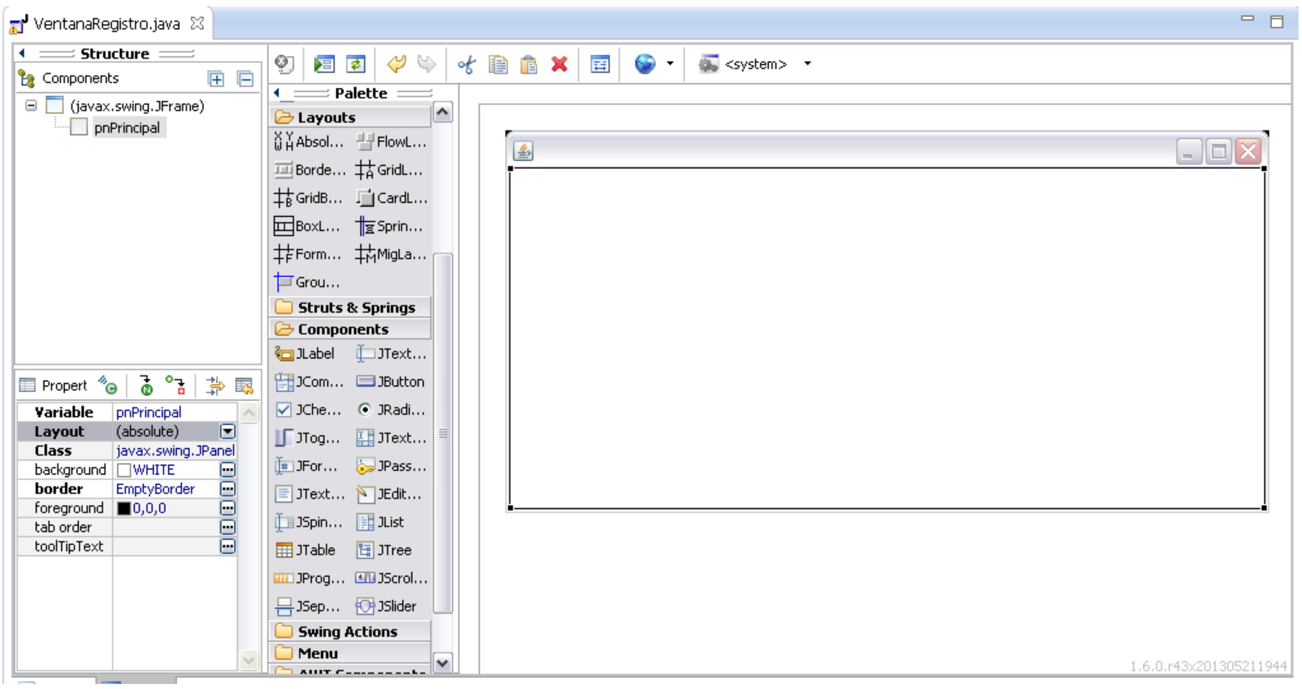
As we sais before, WindowBuilder already included in the window a main container or content panel (contentPane) to contain every component that will form the interface. Changing the name of the component is advisable. For that, select the content panel in the design panel clicking in the center of the window under design. In the property tab we change:

1. **Layout:** select *AbsoluteLayout* from the combo box.
2. **Background**: Click on the three points to open the dialog that allow us to select a new color, and chose white.

4. Customizing the application UI

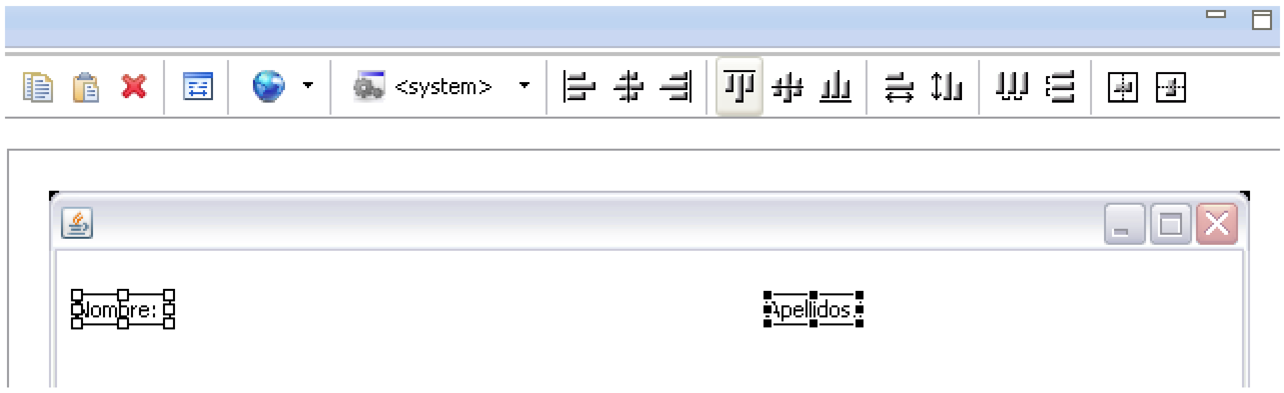
4.1. Adding new components to the application.

1. Select the folder *Components* in the components palette. The relationship of visual components will be shown.


VentanaRegistro.java
Structure
Components
(javax.swing.JFrame)
pnPrincipal
Palette
Layouts
Absol… FlowL… Borde… GridL… GridB… CardL… BoxL… Sprin… Form… MigLa… Grou…
Struts & Springs
Components
JLabel JText… JCom… JButton JChe… JRadi… JTog… JText… JFor… JPass… JText… JEdit… JSpin… JList JTable JTree JProg… JScrol… JSep… JSlider
Swing Actions
Menu
<system>
Propert
1.6.0.r43x201305211944

Variable	pnPrincipal
Layout	(absolute)
Class	javax.swing.JPanel
background	WHITE
border	EmptyBorder
foreground	0,0,0
tab order	
toolTipText	

2. Add the components you need to the window and modify the corresponding attributes. If you want to align all the components in a vertical or horizontal line, select all of them at the same time and choose the more appropriate option in the buttons shown on top of the designer.



For the comboboxes, insert their possible values modifying the attribute *model* and adding a value per line.

5. Adding functionality to a button

Now that we have a kind of application, we will see the way to link functionality to the components we added. We will start associating the button *Cancel* with the code we need to close the application.

Steps:

1. Select the button and deploy the contextual menu with the right button. Select *Add Event Handler.*

2. Select the event *ActionPerformed* and press finish in the dialog. The IDE will generate the code that defines the response to the user action. The following code will be generated:

```
btnCancel.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent arg0) {
        }
  });
```

3. Introduce the sentence that finishes the application.

```
   System.exit(0);
```

And test again!

6. Workshop Lab 2

1. **IMPORTANT**: Study the theory foundations associated to this lab. Remember that all these theory pills are part of the theory contents and will be evaluated in both the theory and lab exams.
2. **Document** (with Javadoc comments) the application and the proposed extensions code.
3. Complete the interface with all the components shown at the beginning of the lab. In the next lab we will need this window with all the components already finished.
4. Link to the event of the *Next* button the required code to check that…
    a. None of the textfields is empty
    b. Both password fields match.

   Show one or several messages whenever any of this two restrictions was violated.

Note 1: the method getText() of a JTextField component returns the string contained in the component. For the JPasswordField components the method to use is getPassword() that returns an array of characters. With *String.valueOf()* the array can be transformed into a String.

Note 2: To show a message to the user we can use:

```
        JOptionPane.showMessageDialog(null, "Text to be shown…");
```

ATTENTION: Workshop lab 2 must be ready for the next Lab. Finish at home any missing requirement.