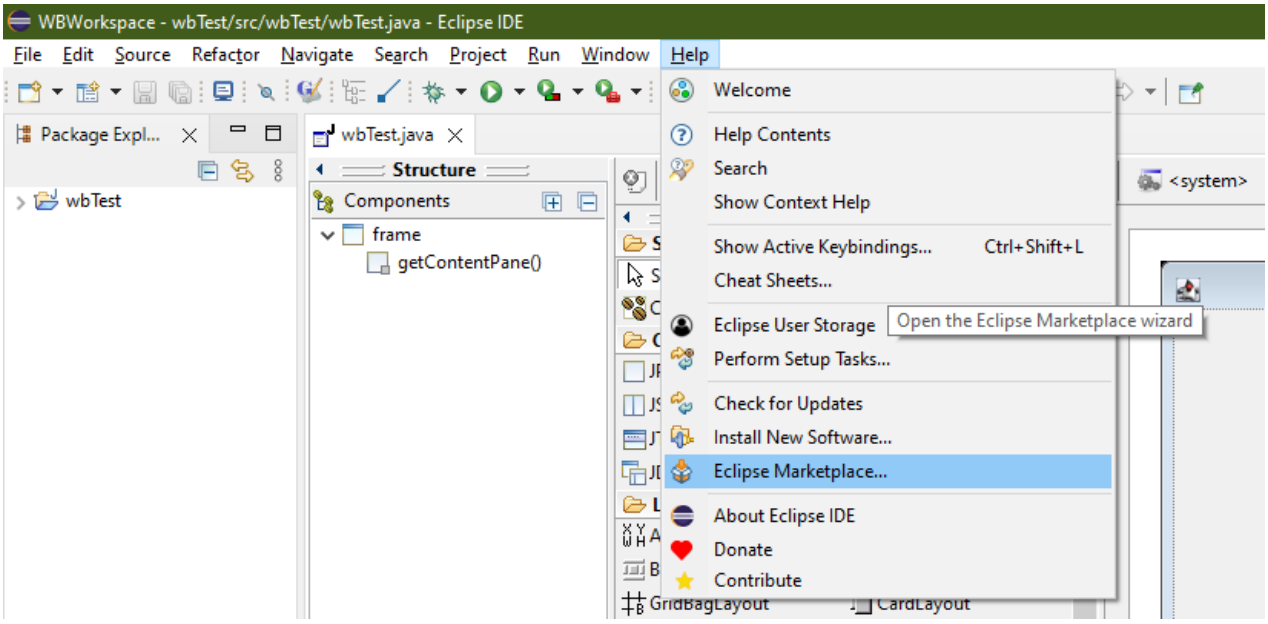
National University of Computer & Emerging Sciences, Karachi
Computer Science Department
Spring 2022, Lab Manual - 04

Course Code: SL3001	Course: Software Development and construction
Instructor(s):	Miss Nida Munawar, Miss Abeeha Sattar

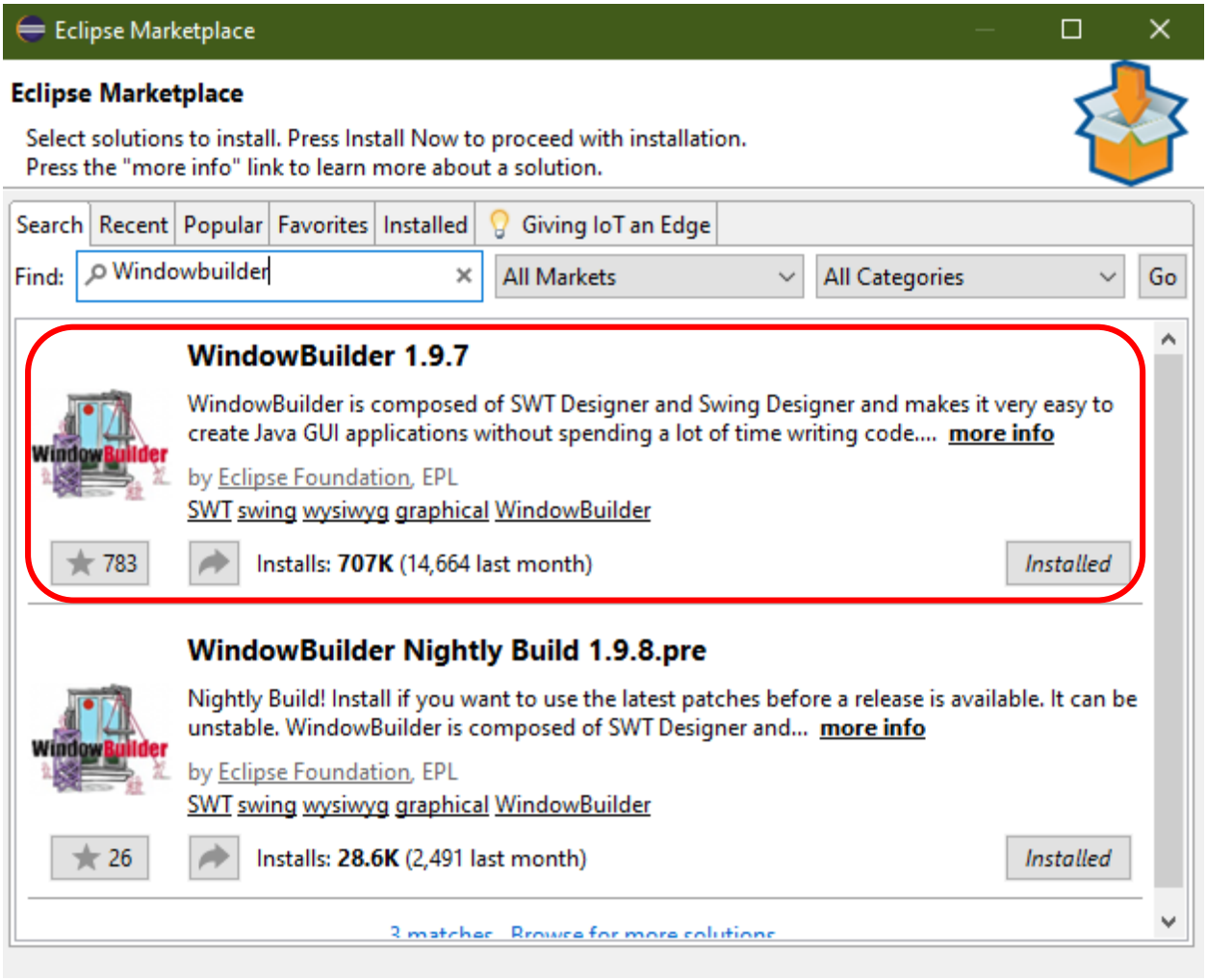
Lab # 04

Installing WindowBuilder on Eclipse:

1. First of all, open **Eclipse > Help > Eclipse Marketplace**

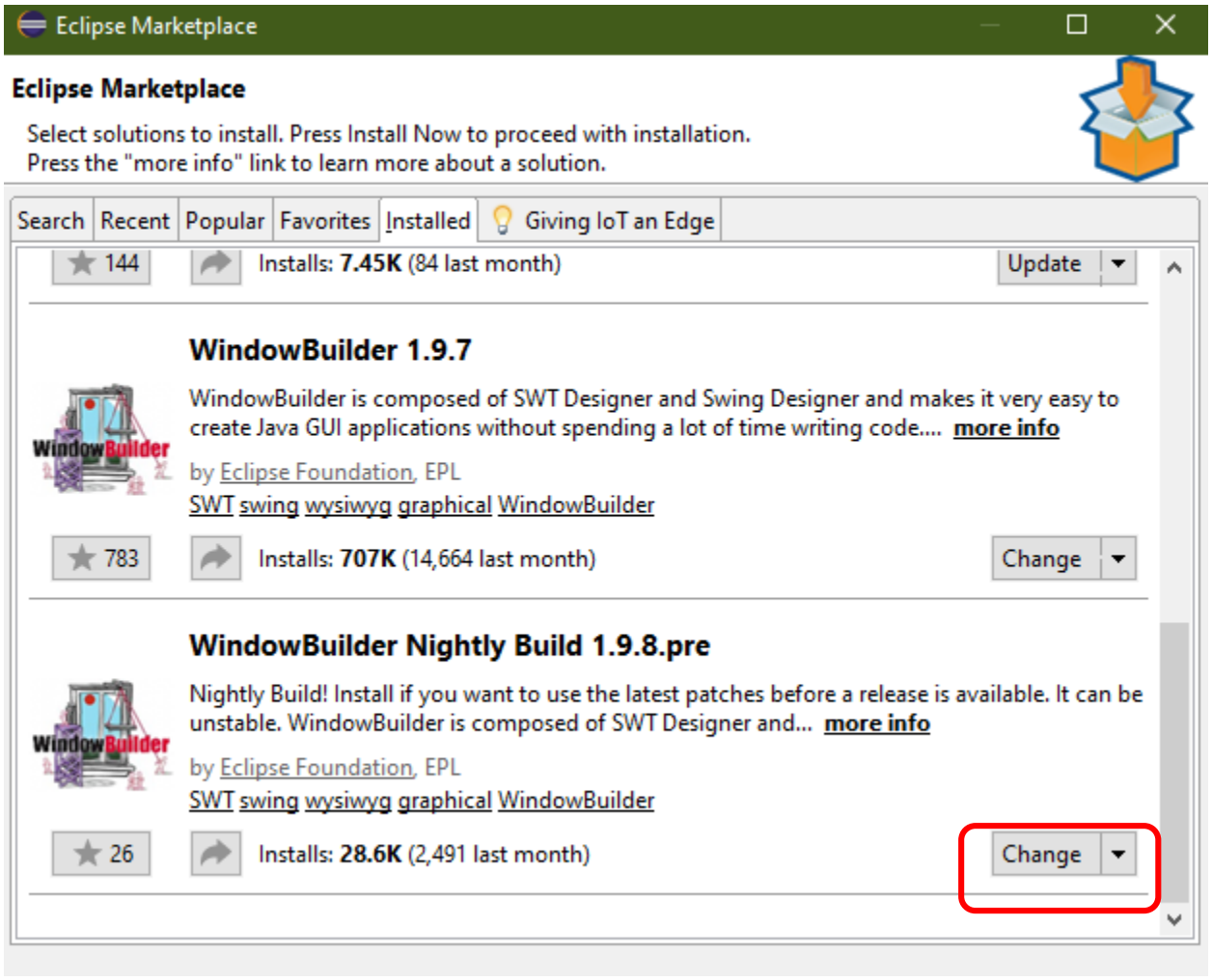


2. Type in “**WindowBuilder**” in the search bar and press **Enter**

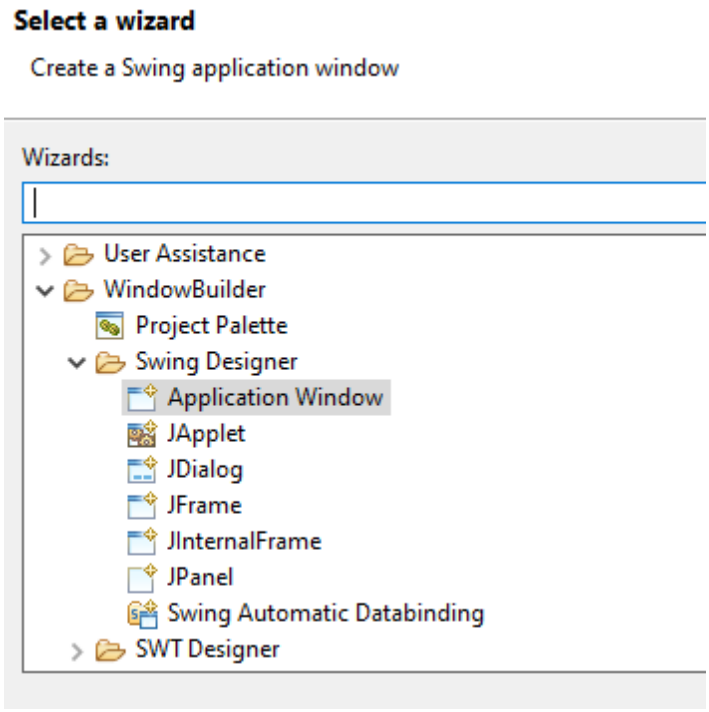


3. Install the first option that shows up.
4. Your Eclipse will restart after the installation is finished.
5. Go to **Help > Eclipse** Marketplace again and go to the “**Installed**” tab

6. Scroll down and update the Nightly Build

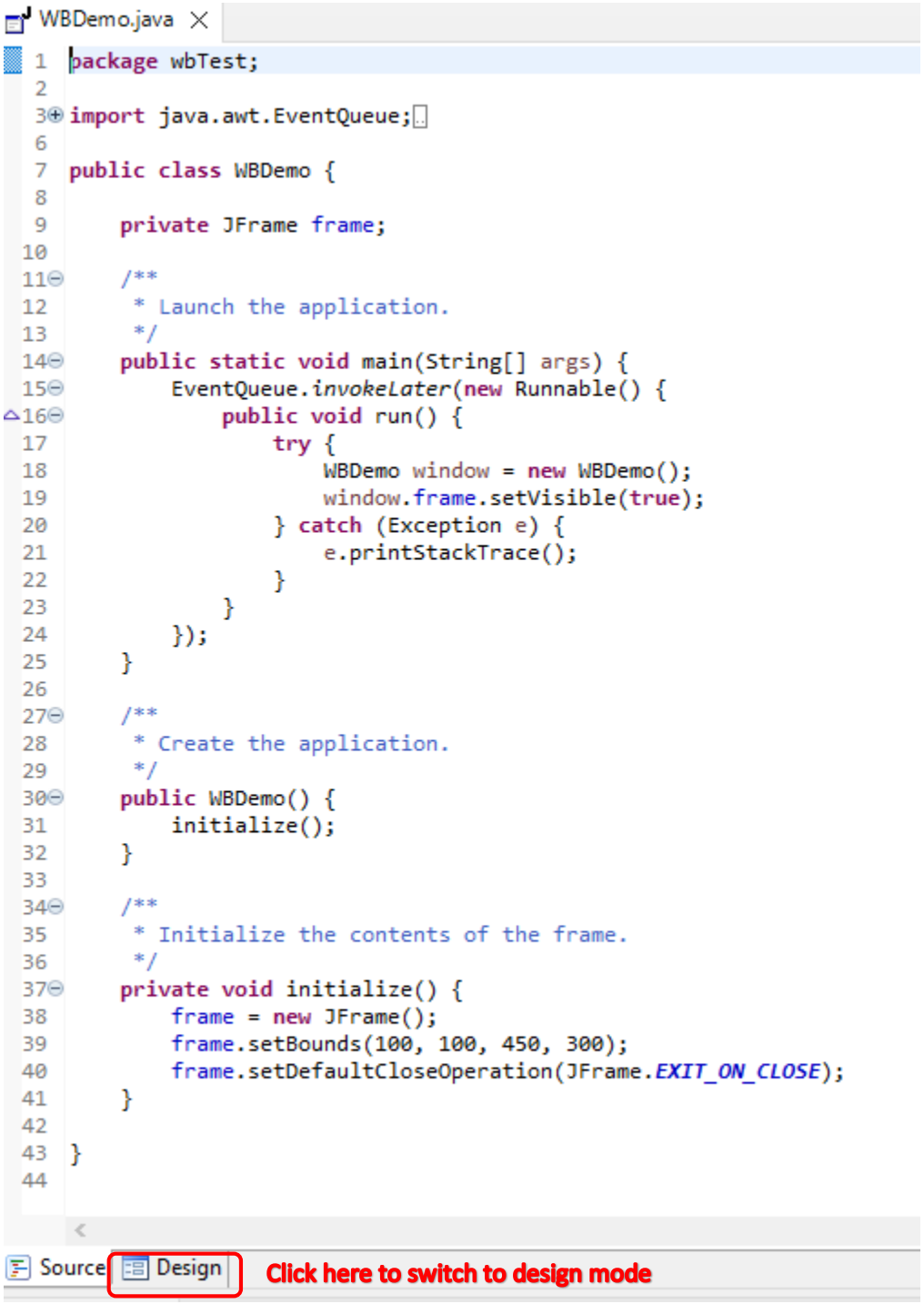


7. You can now add a WindowBuilder program to an existing project by going to **File > New > Other**. Alternatively, you can create a new Empty Project, and then add a new WindowBuilder program to it.



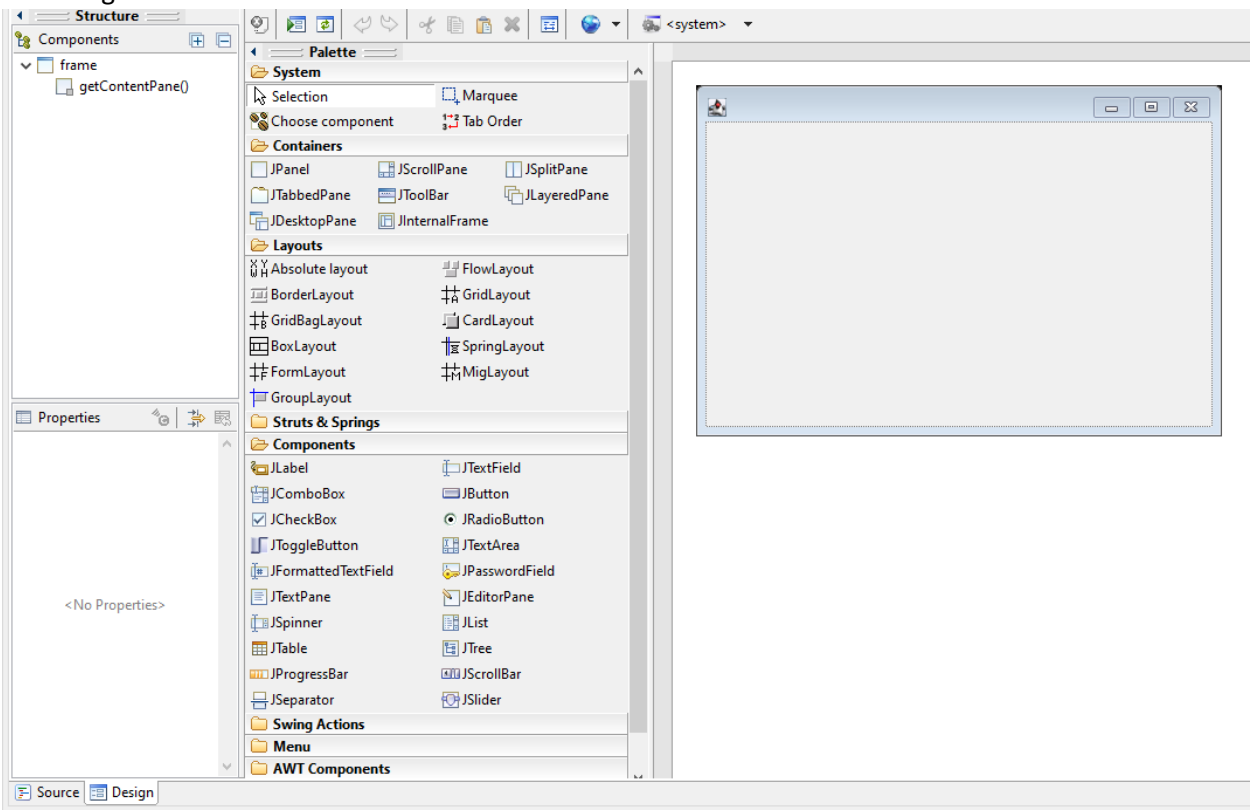
8. Select "**Application Window**" and select a name for your application. For example, **WBDemo**

9. It will create a basic program that opens a window. You can view the design by clicking on the "**Design**" tab at the bottom.




```java
package wbTest;

import java.awt.EventQueue;

public class WBDemo {

    private JFrame frame;

    /**
     * Launch the application.
     */
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    WBDemo window = new WBDemo();
                    window.frame.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }

    /**
     * Create the application.
     */
    public WBDemo() {
        initialize();
    }

    /**
     * Initialize the contents of the frame.
     */
    private void initialize() {
        frame = new JFrame();
        frame.setBounds(100, 100, 450, 300);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

}
```

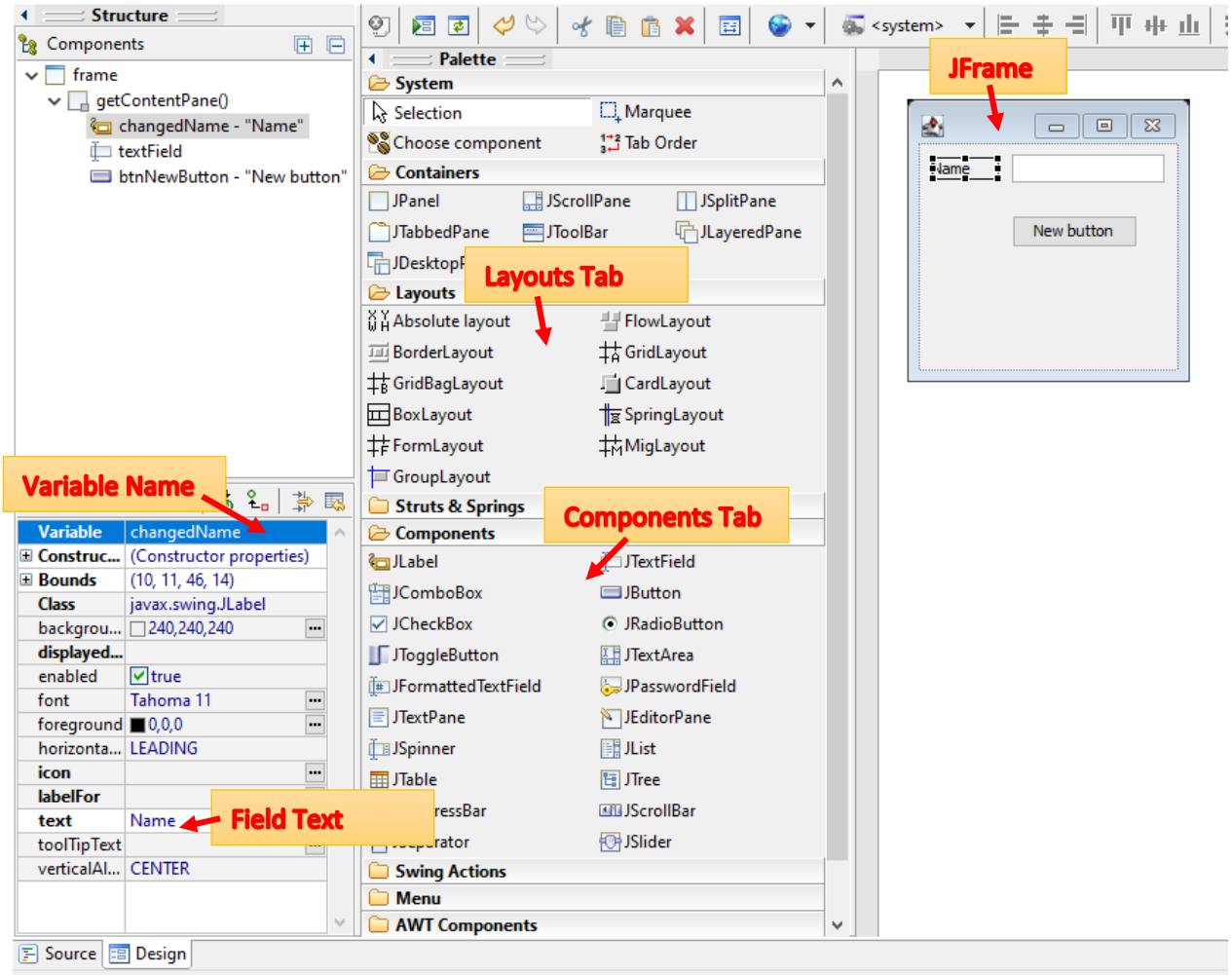
10. The design mode will look like this:



11. You can select and drop the desired components on your frame.
12. You can switch to "**Source**" to view the code that is autogenerated by your actions.

Task:

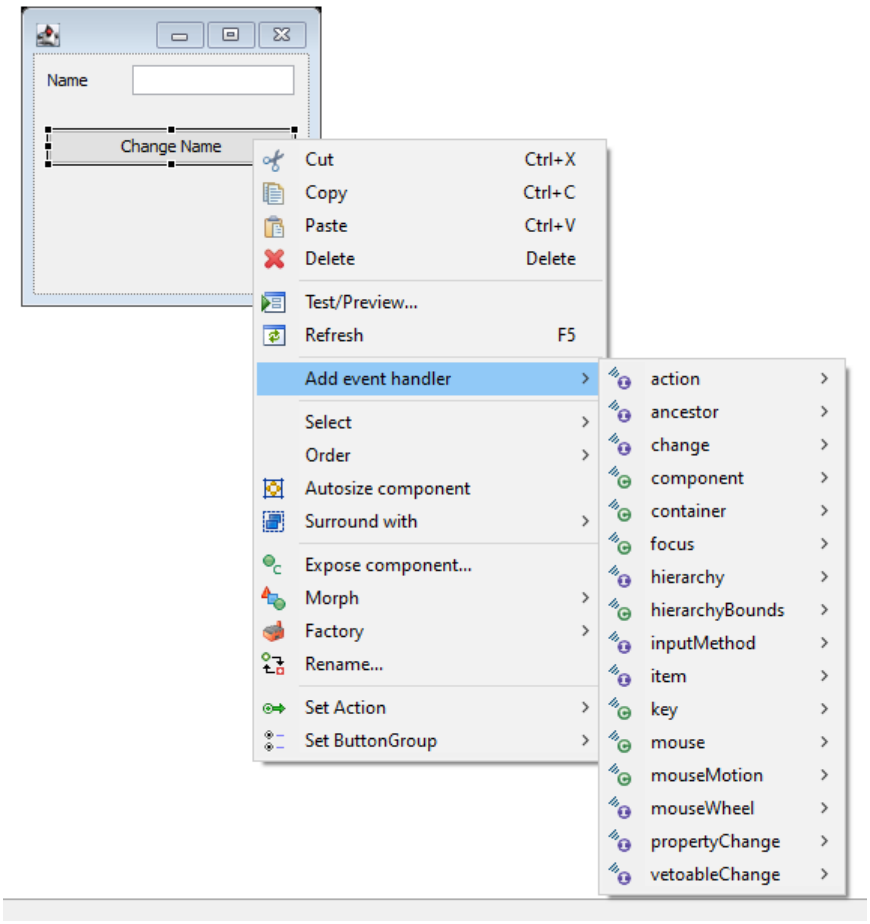
- First apply **Absolute Layout** to your JFrame. This will allow you to place your components how you want. You can do this by clicking on absolute layout, and then clicking on your JFrame.
- Add a **JLabel**, a **JTextField**, and a **JButton** to your **JFrame** by selecting them from the list and dropping them into it. Observe the generated code.
- Switch back to design mode, and click on any of the components that you added. Observe the properties tab on the left side of the screen.
- Change the variable name and text displayed by the field through the properties tab. Observe the changes in the code.



Adding Event Listeners through WindowBuilder

WindowBuilder makes it so that you don't have to add Listeners manually. Let's have a look at how we can add listeners through our Design Screen.

1. Right-click on the JButton on the previously made form and menu will appear.



2. Add an action listener for the button
3. Your code will now contain a listener and an overloaded function with an empty body

```
58          JButton btnNewButton = new JButton("Change Name");
59          btnNewButton.addActionListener(new ActionListener() {
60              public void actionPerformed(ActionEvent e) {
61              }
62          });
```

4. You can now write the body of your function for example:
   `textField.setText("Your name here");`
5. You can add other types of listeners in the same manner.

How to add other components to your JFrame?

A multitude of components exist which you can include inside your JFrame. Some of them are:

1. JLabel
2. JTextField
3. JTextArea
4. JPasswordField
5. JCheckBox
6. JRadioButton
7. JComboBox
8. JTable
9. JList
10. JOptionPane
11. JScrollBar
12. JMenuItem & JMenu
13. JPopupMenu
14. JCheckBoxMenuItem
15. JSeparator
16. JProgressBar
17. JTree
18. JColorChooser
19. JTabbedPane
20. JSlider
21. JSpinner
22. JDialog
23. JPanel
24. JFileChooser
25. JToggleButton
26. JToolBar
27. JViewport
28. JFrame
29. JComponent
30. JLayeredPane
31. JDesktopPane
32. JEditorPane
33. JScrollPane
34. JSplitPane
35. JTextPane
36. JRootPane

You can add any of the components using the components available on the **Palette** tab.

We have already seen how to use JLabel, JTextField and JButton. Let use try and use some other commonly used components.

JRadioButton

You can add radio buttons to the JFrame just like any other components. But in order to use them like normal radio buttons, you will need to add them to a button group.

It can be done using the following lines of code

```
ButtonGroup group = new ButtonGroup();
group.add(radioButtonName);
group.add(radioButtonName_2);
```

You can then use listeners for each of the radio buttons, as necessary.

How to Make Dialogs

JOptionPane

A dialog window is an independent sub window meant to carry temporary notice apart from the main Swing Application Window. Most Dialogs present an error message or warning to a user, but Dialogs can present images, directory trees, or just about anything compatible with the main Swing Application that manages them.

A simple message can be displayed like:

```
JOptionPane.showMessageDialog(frame, "Name Set!");
```

This can be part of a function that handles events.

You can pass an additional parameter to show if the message is an error, notification, warning, or a plain message etc.

The function calls would look like:

(Try these out for yourself!)

1. `JOptionPane.showMessageDialog(frame, "Eggs are not supposed to be green.");`

2. `JOptionPane.showMessageDialog(frame, "Eggs are not supposed to be green.", "Inane warning", JOptionPane.WARNING_MESSAGE);`

3. `JOptionPane.showMessageDialog(frame, "Eggs are not supposed to be green.", "Inane error", JOptionPane.ERROR_MESSAGE);`

4. `JOptionPane.showMessageDialog(frame, "Eggs are not supposed to be green.", "A plain message", JOptionPane.PLAIN_MESSAGE);`

JOptionPane.showOptionDialog displays a modal dialog with the specified buttons, icons, message, title, and so on. With this method, you can change the text that appears on the buttons of standard dialogs. You can also perform many other kinds of customization.

```
Object[] options = {"Yes, please",
                    "No, thanks",
                    "No eggs, no ham!"};

//will return a value based on your choice
int n = JOptionPane.showOptionDialog(frame,
    "Would you like some green eggs to go with that ham?", //text to display
    "A Silly Question", //title
    JOptionPane.YES_NO_CANCEL_OPTION, //type of options
    JOptionPane.QUESTION_MESSAGE, //icon to display [Warning, Error, etc]
    null, //icon to use
    options, //array options
    options[2]); //default choice
```

JTable are used to display the data in a tabular format. You can try out this piece of code in your **initialize()** method.

```
frame = new JFrame();
              frame.setBounds(100, 100, 663, 538);

              JPanel panel = new JPanel();
              String[][] rec = {
                                 { "1", "Steve", "AUS" },
                                 { "2", "Virat", "IND" },
                                 { "3", "Kane", "NZ" },
                                 { "4", "David", "AUS" },
                                 { "5", "Ben", "ENG" },
                                 { "6", "Eion", "ENG" },
                              };
                              String[] header = { "Rank", "Player", "Country" };
                              JTable table = new JTable(rec, header);
                              panel.add(new JScrollPane(table));
                              frame.getContentPane().add(panel);

              frame.setVisible(true);
              frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```