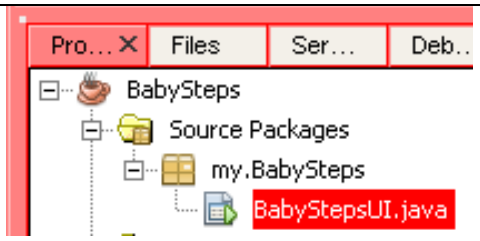
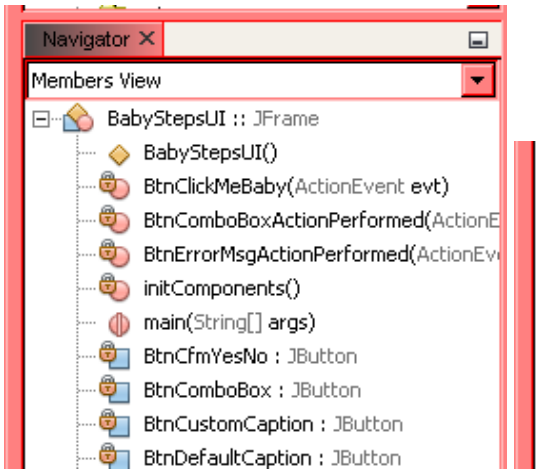
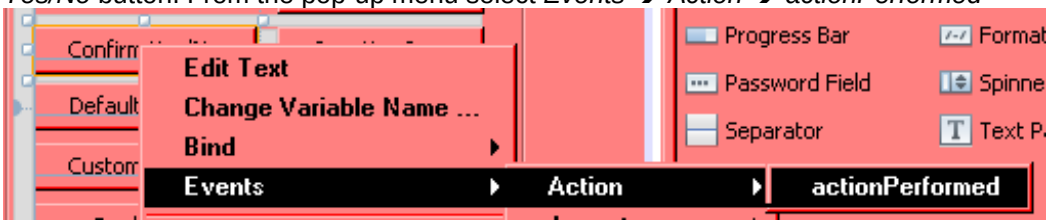
Add Message Box Events.doc

Read *Getting Started with* NetBeans to learn the first steps for creating a GUI application. The project *BabySteps* now contains a so-called JPanel overlaid with eight buttons and a text field. Each of the buttons demonstrates a different Message Box appearance and behavior. The Navigator window (below) lists all of the controls for this project..	



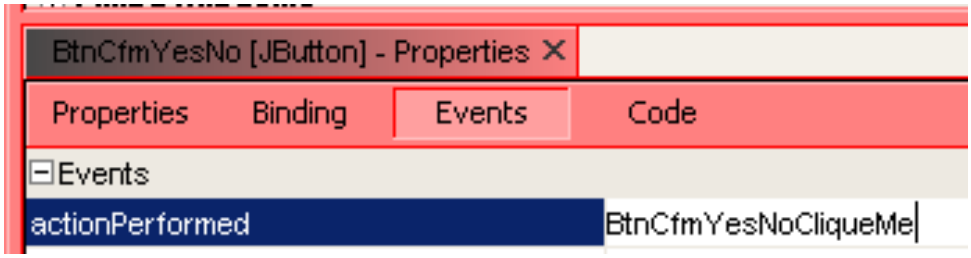
In order to make our buttons do something, we must create event handlers. Right-click the *Confirm Yes/No* button. From the pop-up menu select *Events* ➔ *Action* ➔ *actionPerformed*



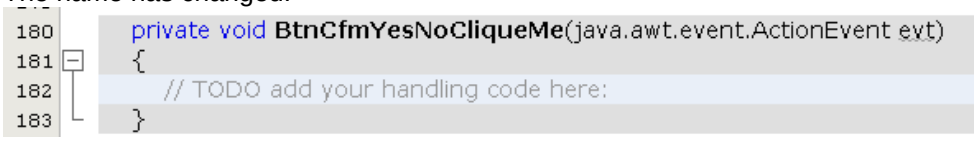
We now have an empty event handler for this button.

```
180     private void BtnCfmYesNoActionPerformed(java.awt.event.ActionEvent
181     {
182         // TODO add your handling code here:
183     }
```

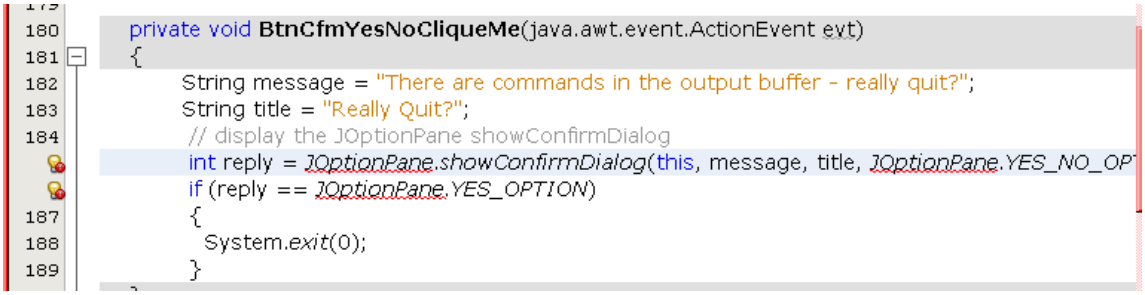
To change the event's name: Highlight *actionPerformed* in the Properties window, Enter the new name it in the Code column.



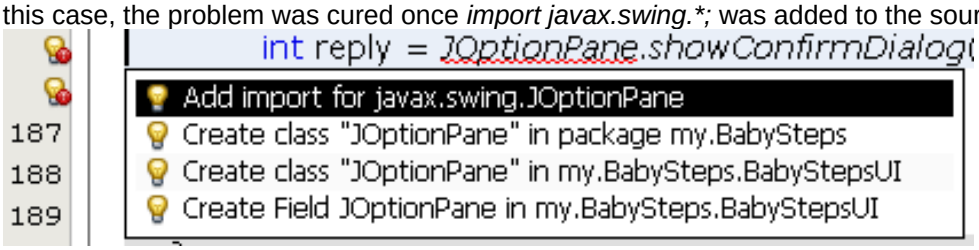
The name has changed!

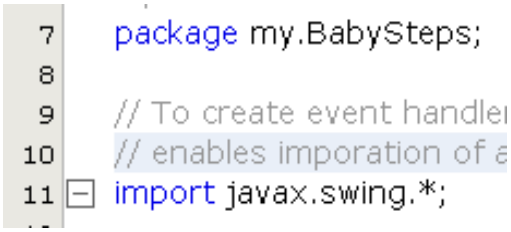


Type some code into this handler in order to make it function as a message box. But note the text underlined in red and the icons in the line number column. These indicate errors.

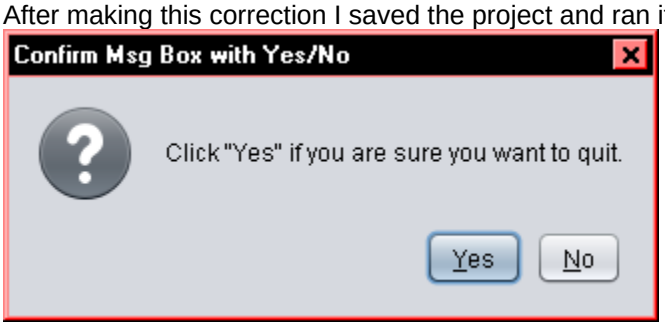


Hover the mouse over one of the icons for information about the problem along with suggested fixes. In this case, the problem was cured once *import javax.swing.*;* was added to the source code.





After making this correction I saved the project and ran it. Here is the resulting message box:



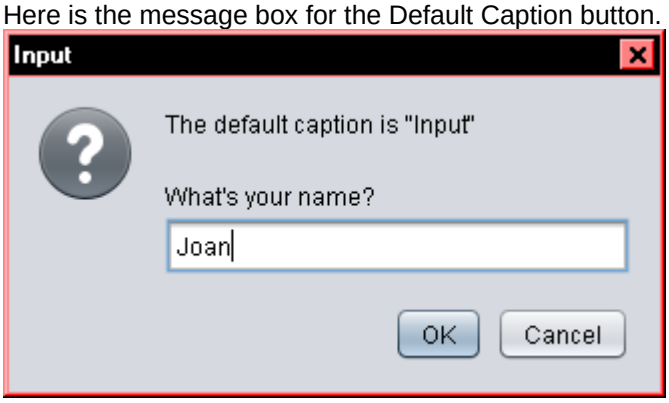
Here is the code for that message box:.

```
189     private void BtnCfmYesNoCliqueMe(java.awt.event.ActionEvent evt)
190     {
            String message = "Click \"Yes\" if you are sure you want to quit.";
192         String title = "Confirm Msg Box with Yes/No";
193          // display the JOptionPane showConfirmDialog
194          int reply = JOptionPane.showConfirmDialog(this, message, title, JOptionPane.YES_NO_OPTION);
195          if (reply == JOptionPane.YES_OPTION)
196          {
197            System.exit(0);
198          }
199     }
```

To center the running form on the monitor, add this code:

```
17      public BabyStepsUI()
18       {
19         initComponents();
20
21         // to center the form on the monit
22         this.setLocationRelativeTo(null);
23       }
```

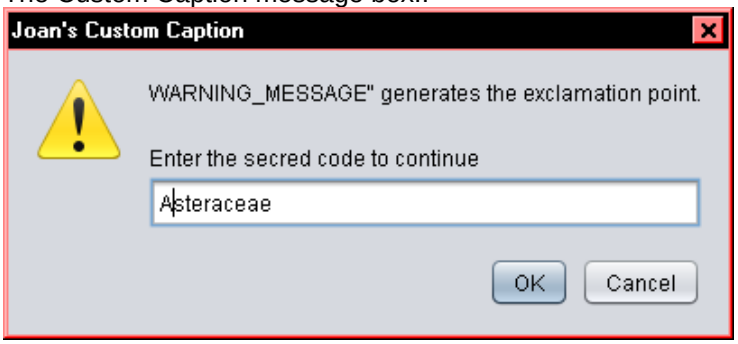
Here is the message box for the Default Caption button.



… and the code:

```
248
249     private void BtnDefaultCaptionClique(java.awt.event.ActionEvent evt)
250     {
251      // Input Dialog box #1  Default caption
252           String myMsg = "The default caption is \"Input\"\n\n";
253           myMsg = myMsg + "What's your name?";
254           String name = JOptionPane.showInputDialog(this, myMsg);
255           // display in textbox
256           txtTemp.setText(name);  //put into textbox
257
258     }
```

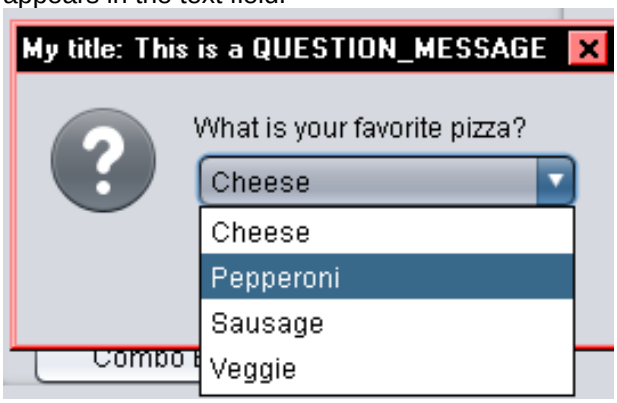
The Custom Caption message box:.

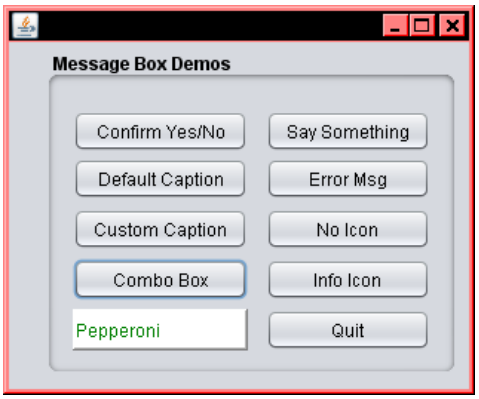


… and the code:

```
260     private void BtnCustomCaptionClique(java.awt.event.ActionEvent evt)
261     {
262
263          // Input Dialog Box #2 Custom Caption
264          String myCustom = "WARNING_MESSAGE\" generates the exclamation point.";
265          myCustom = myCustom + "\n\nEnter the secred code to continue";
266          String code = JOptionPane.showInputDialog(this,
267                                    myCustom,
268                                    "Joan's Custom Caption",
269                                    JOptionPane.WARNING_MESSAGE);
270           // display in textbox
271           txtTemp.setText(code);
272
273     }
```

The message box for the Combo Box button: When you make a selection from the combo box, it appears in the text field.

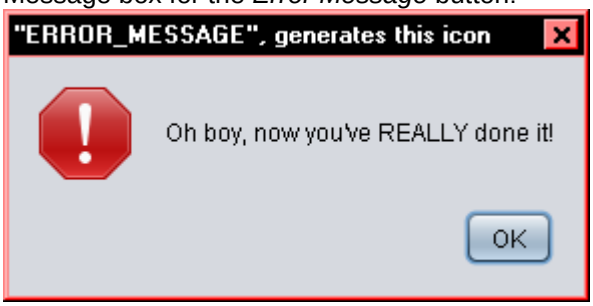




The code for the Combo Box button:

```
210
211     private void BtnComboBoxClique(java.awt.event.ActionEvent evt)
212     {
213           //  Input dialog with a combo box for choosing
214           String[] pizzas = { "Cheese", "Pepperoni", "Sausage", "Veggie"};
215           String favoritePizza = (String) JOptionPane.showInputDialog(this,
216              "What is your favorite pizza?",
217              "My title: This is a QUESTION_MESSAGE",  // Msg box title
218              JOptionPane.QUESTION_MESSAGE,
219              null,
220              pizzas,
221              pizzas[0]);
222
223           txtTemp.setText(favoritePizza);
224
225     }
```

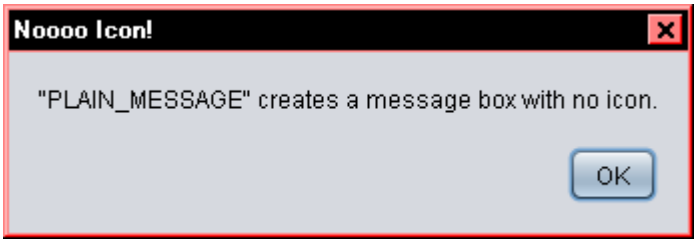
Message box for the *Error Message* button:



…and the code:

```
227     private void BtnErrorMsgClique(java.awt.event.ActionEvent evt)
228     {
229           //  with an error message
230
231           JOptionPane.showMessageDialog(this,
232              "Oh boy, now you've REALLY done it!",
233              "\"ERROR_MESSAGE\", generates this icon",
234              JOptionPane.ERROR_MESSAGE);
235     }
```

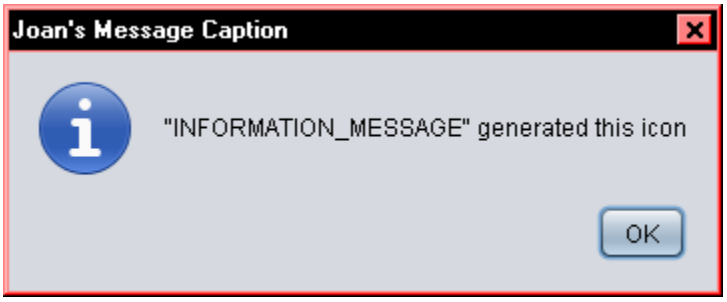
From the *No Icon* button:



… its code:

```
284
285     private void BtnNoIconClique(java.awt.event.ActionEvent evt)
286     {
287         //  no icon
288             JOptionPane.showMessageDialog(this,
289             "\"PLAIN_MESSAGE\" creates a message box with no icon.",
290             "Noooo Icon!",
291             JOptionPane.PLAIN_MESSAGE);
292
293     }
```

From the *Info Icon* button:



It's code:

```
274
275     private void BtnInformationIconClique(java.awt.event.ActionEvent evt)
276     {
277             //  Message, Caption, and Icon
278              JOptionPane.showMessageDialog(this,
279              "\"INFORMATION_MESSAGE\" generated this icon",
280              "Joan's Message Caption",
281              JOptionPane.INFORMATION_MESSAGE);
282
283     }
```