

How to Create a Swing GUI in Java

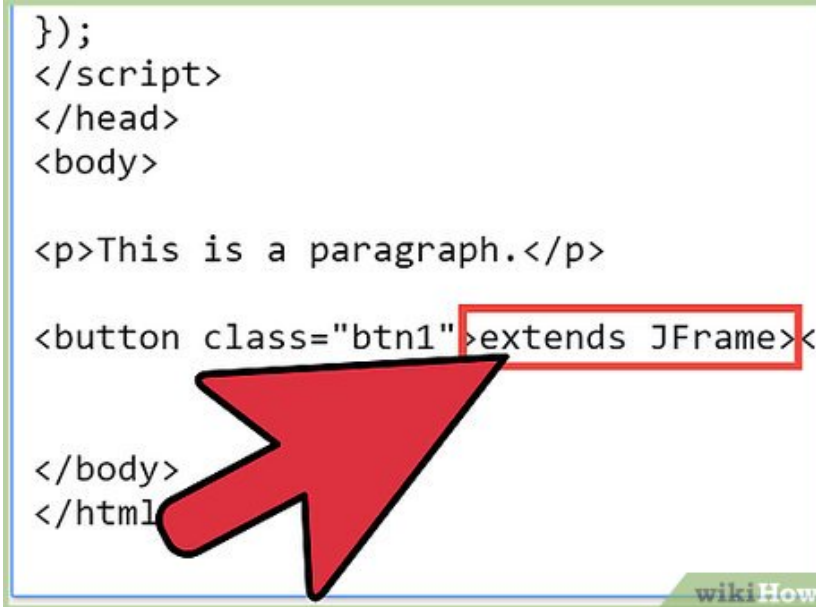
This article explains how to create simple application that is shown in the figure on the right, giving its source code as well. To place buttons, text labels and other components on the program window, you need to understand about JPanel....

Method 1 of 2:

Making the Overall Frame

```
});  
</script>  
</head>  
<body>  
  
<p>This is a paragraph.</p>  
  
<button class="btn1">extends JFrame<  
  
</body>  
</html>
```


1.



Create a class that extends the *JFrame* class. This class will hold all of your GUI components, such as buttons and text fields.

2.

```
public static void main(String[] args) {
    try {
        // Set cross-platform Java L&F (also called "Metal")
        UIManager.setLookAndFeel(
            UIManager.getCrossPlatformLookAndFeelClassName());
    }
    catch (UnsupportedLookAndFeelException e) {
        // handle exception
    }
    catch (ClassNotFoundException e) {
        // handle exception
    }
    catch (InstantiationException e) {
        // handle exception
    }
    catch (IllegalAccessException e) {
        // handle exception
    }
}
```



Plan the overall layout of your first application. A good start could be a central panel with *BorderLayout* with another panel at the bottom (*BorderLayout.South*). This second panel may have the *FlowLayout* and contain several buttons, check boxes and other similar controls. Finally, place the big *JTextArea* into the center of the central component. You will be able to use its *getText()* and *setText()* methods to do some text-based interaction with the user.


3.

```
<!DOCTYPE html>
<html>
<body>

<h1>TEST</h1>

<button type="button"
(myFrame.getContentPane().add(myLargePanel,
BorderLayout.Center)
<p id="demo"></p>

</body>
</html>
```



Write constructor to your class. This constructor must create all panels and components you plan, place them properly into each other and add the final panel the "holds all" to you frame (*myFrame.getContentPane().add(myLargePanel, BorderLayout.Center*).

4.

```
<!DOCTYPE html>
<html>
<body>

<p id="demo"></p>

<script>
var x1 = new Object(); // A new Object object
var x2 = new String(); // A new String object
var x3 = new Number(); // new Number object
var x4 = new Boolean(); // new Boolean object
var x5 = new Array(); // Array object
var x6 = new RegExp(); // RegExp object
var x7 = new Function(); // Function object
var x8 = new Date(); // Date object

document.getElementById("demo").innerHTML =
"x1: " + tvneof x1 + "<br>" +
```

Write the main method which will be the program's entry point. In this method, create an instance of your frame, set the initial size and location (use `.setSize(x,y)` and `.setLocation(width, height)`) and make it to appear on the screen by calling `.setVisible(true)`.

Method 2 of 2:

Programming responses to the user actions

1.

```
<button id="myBtn">Try it</button>

<p id="demo"></p>

<script>
document.getElementById("myBtn").addEventListener
("click", displayDate);

function displayDate() {
    document.getElementById("demo").innerHTML =
Date();
}
</script>

</body>
</html>
```

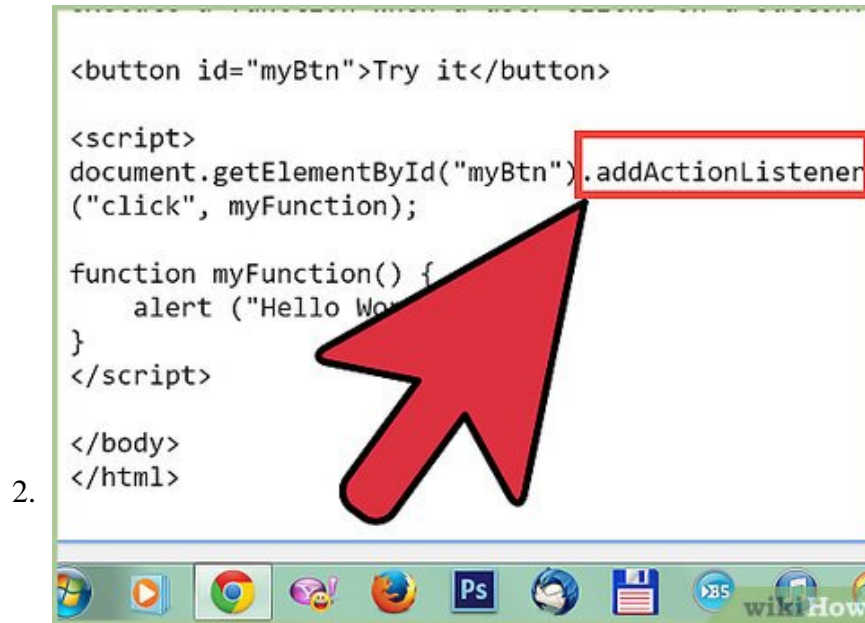
Make your frame implement the *ActionListener* interface. This will allow your class to listen to components' activities.

```
2. <button id="myBtn">Try it</button>

<script>
document.getElementById("myBtn").addActionListener
("click", myFunction);

function myFunction() {
    alert ("Hello World");
}
</script>

</body>
</html>
```



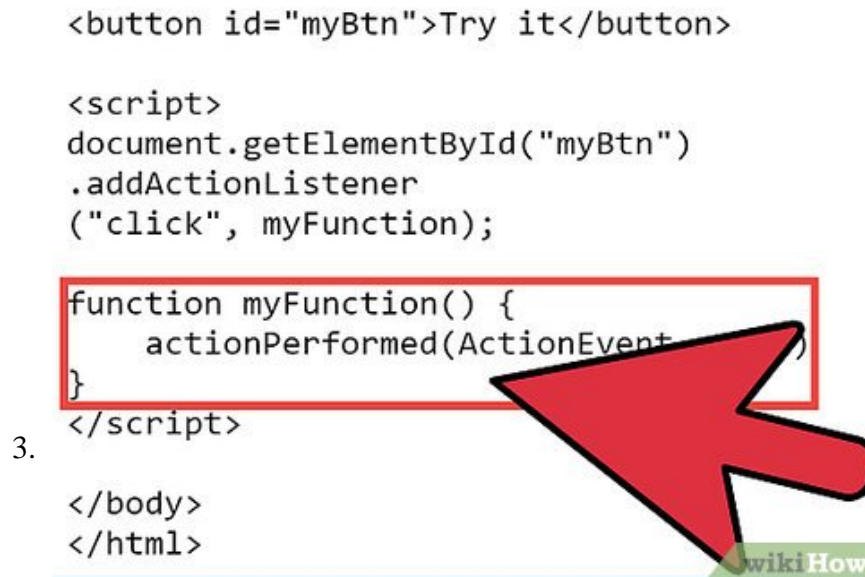
For every button, check box or other control that you have created, invoke its method `.addActionListener`, passing your frame (*this*) as parameter.

```
3. <button id="myBtn">Try it</button>

<script>
document.getElementById("myBtn")
.addActionListener
("click", myFunction);

function myFunction() {
    actionPerformed(ActionEvent event)
}
</script>

</body>
</html>
```



Override ActionListener's abstract method, `actionPerformed(ActionEvent event)`. In this method, you should put if statements checking where does the action event come from. This if statement should have a condition that says something like `"if (event.getSource() == button1)"`. This checks where the event came from and if it came from your button. Inside the if statement, do whatever needs to be done when your button is pressed.

```
4. "Clicked!<br>";  
   }  
  
   function myThirdFunction() {  
     .setText("myText")  
   }  
 </script>  
  
</body>  
</html>
```



JTextArea has a method `.setText("myText")` which seems good as the way to program some visible response on your action.

You finished reading the article "**How to Create a Swing GUI in Java**" edited by the [TipsMake](#) team. We hope this article has provided you with many useful tech tips and tricks. You can search for similar articles on tips and guides. Thank you for reading and for following us regularly.