# Introduction to GUI Building

# This beginner tutorial is from <http://netbeans.org/kb/docs/java/gui-functionality.html> and it teaches you how to create simple graphical user interface (GUI) for an application (simple calculator), using the NetBeans IDE GUI Builder. In particular, you will see how to layout a GUI front-end that enables you to view and edit user input and how to code the behaviour of buttons and fields in a Swing form.

We will work through the layout and design of a GUI and add a few buttons and a text field. The text fields will be used for receiving user input and also for displaying the program output. The button will initiate the functionality built into the front end. The application we create will be a simple but functional calculator.

For a more comprehensive guide to the GUI Builder's design features, including video demonstrations of the various features, see *Designing a Swing GUI in NetBeans IDE* (<http://netbeans.org/kb/docs/java/quickstart-gui.html>).

**To follow this tutorial, you need the following software and resources.**

|  |  |
| --- | --- |
| [NetBeans IDE](http://www.netbeans.org/downloads/index.html) | version 6.9 or higher (8.1) |
| [Java Development Kit (JDK)](http://www.oracle.com/technetwork/java/javase/downloads/index.html) | version 6, 7 or 8 |

## Exercise 1: Creating a Project

The first step is to create an IDE project for the application that we are going to develop. We will name our project Lab8.

1. Choose *File > New Project*. Alternatively, you can click the *New Project* icon in the IDE toolbar.
2. In the Categories pane, select the *Java* node. In the Projects pane, choose *Java Application*. Click Next.
3. Type *Lab8* in the Project Name field and specify a path ,e.g., in your home directory as the project location.
4. Leave the *Use Dedicated Folder* for Storing Libraries checkbox **unselected**.
5. **Deselect** the *Create Main Class* checkbox if it is selected.
6. Click Finish.

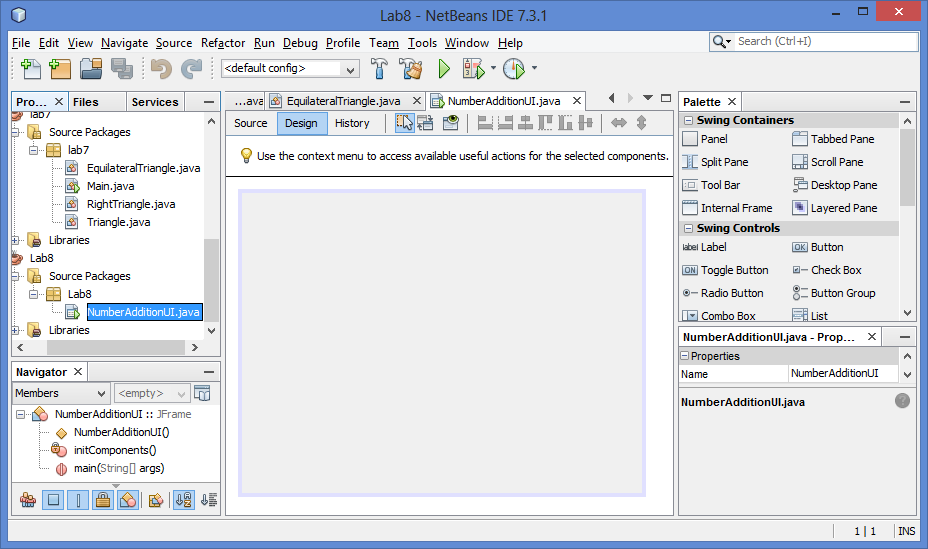
“The IDE creates *Lab8* folder in the designated location of your system. This folder contains all of the project's associated files, including its Ant script, folders for storing sources and tests, and a folder for project-specific metadata. To view the project structure, use the IDE's Files window.”

## Exercise 2: Building the Front End

To proceed with building our interface, we need to create a Java container within which we will place the other required GUI components. In this step we'll create a container using the *JFrame* component. We will place the container in a new package, which will appear within the Source Packages node.

### Create a JFrame container

1. In the Projects window, right-click the Lab8 node and choose New > Other.
2. In the New File dialog box, choose the *Swing GUI Forms* category and the *JFrame Form* file type. Click Next.
3. Enter *NumberAdditionUI* as the class name.
4. Enter *myPac*  as a package name.
5. Click Finish.

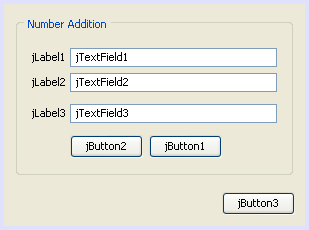
 The IDE creates the NumberAdditionUI form and the NumberAdditionUI class within the Lab8 application, and opens the NumberAdditionUI form in the GUI Builder's Design view and three additional windows appear automatically along the IDE's edges, enabling you to navigate, organize, and edit GUI forms as you build them.

The GUI Builder's various windows include:

* **Design Area.** The GUI Builder's primary window for creating and editing Java GUI forms. The toolbar's ***Source*** button enables you to view a class's source code, the ***Design*** button allows you to view a graphical view of the GUI components, the ***History*** button - to access the local history of changes of the file. The additional toolbar buttons provide convenient access to common commands, such as choosing between Selection and Connection modes, aligning components, setting component auto-resizing behaviour, and previewing forms.
* **Navigator.** Provides a representation of all the components, both visual and non-visual, in your application as a tree hierarchy. The Navigator also provides visual feedback about what component in the tree is currently being edited in the GUI Builder as well as allows you to organize components in the available panels.
* **Palette.** A customizable list of available components containing tabs for JFC/Swing, AWT, and JavaBeans components, as well as layout managers. In addition, you can create, remove, and rearrange the categories displayed in the Palette using the customizer.
* **Properties Window.** Displays the properties of the component currently selected in the GUI Builder, Inspector window, Projects window, or Files window.

If you click the *Source* button, the IDE displays the application's Java source code in the Editor with sections of code that are automatically generated by the GUI Builder indicated by grey areas (they become blue when selected), called Guarded Blocks. Guarded blocks are protected areas that are not editable in Source view. You can only edit code appearing in the white areas of the Editor when in Source view. If you need to make changes to the code within a Guarded Block, clicking the Design button returns the IDE's Editor to the GUI Builder where you can make the necessary adjustments to the form. When you save your changes, the IDE updates the file's sources.

### Adding Components: Making the Front End

Next we will use the Palette to populate our application's front end with a JPanel. Then we will add three JLabels, three JTextFields, and three JButtons. If you have not used the GUI Builder before, you might find information in the *Designing a Swing GUI in NetBeans IDE* ([netbeans.org/kb/docs/java/quickstart-gui.html](http://netbeans.org/kb/docs/java/quickstart-gui.html)) tutorial on positioning components useful. Once you are done dragging and positioning the aforementioned components, the JFrame should look something like this screenshot.

If you do not see the Palette window in the upper right corner of the IDE, choose *Window > Palette*.

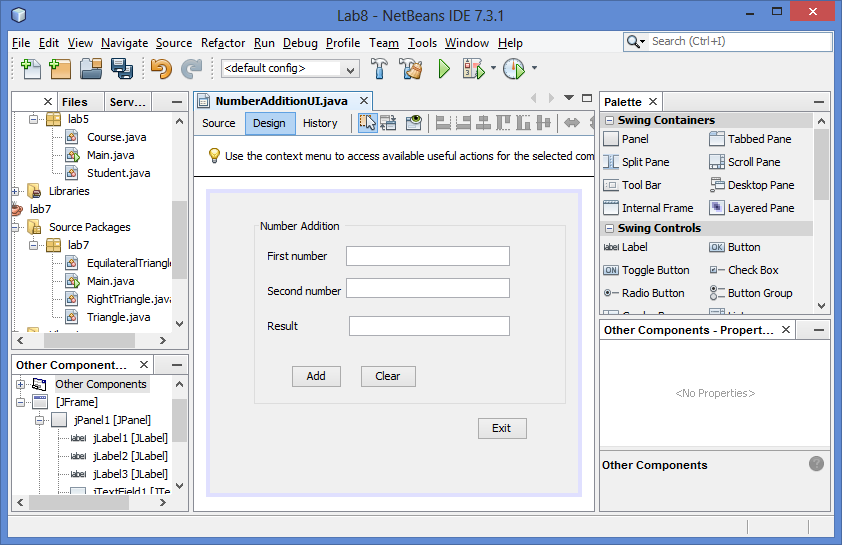
1. Start by selecting a *Panel*  from the Palette and drop it onto the Frame.
2. While the *Panel* is highlighted, go to the *Properties* window and click the ellipsis (...) button next to *Border* to choose a border style.
3. In the Border dialog, select *TitledBorder* from the list, and type in *Number Addition* in the Title field. Click OK to save the changes and exit the dialog.
4. You should now see an empty titled Frame that says *Number Addition*. Take a look at the screenshot above and add three *JLabels*, three *JTextFields* and three *JButtons* to your frame.

### Renaming the Components

In this step we are going to rename the display text of the components that were just added to the JFrame.

1. Double-click *jLabel1* and change the text property to *First Number*
2. Double-click *jLabel2* and change the text to *Second Number*
3. Double-click *jLabel3* and change the text to *Result*
4. Delete the sample text from *jTextField1*. You can make the display text editable by clicking the text field, pausing, and then clicking the text field again. You may have to resize the *jTextField1* to its original size. Repeat this step for *jTextField2* and *jTextField3*.
5. Rename the display text of *jButton1* to *Clear*. (You can edit a button's text by right-clicking the button and choosing Edit Text. Or you can click the button, pause, and then click again.)
6. Rename the display text of *jButton2* to *Add*.
7. Rename the display text of *jButton3* to *Exit*.

Your Finished GUI should now look like the one in the following screenshot:



## Exercise 3: Adding Functionality

In this exercise we are going to assign functionality to the *Add*, *Clear*, and *Exit* buttons. The *jTextField1* and *jTextField2* boxes will be used for user input and *jTextField3* for program output - what we are creating is a very simple calculator.

### Making the Exit Button Work

In order to make the buttons alive, we have to assign an event handler to each, enabling them to respond to events. In our case we want to know when the button is pressed, either by mouse click or via the keyboard. So, we will use *ActionListener* responding to *ActionEvent*.

1. Right click the *Exit* button. From the pop-up menu choose *Events > Action > ActionPerformed*. Note that the menu contains many more events you can respond to! When you select the *actionPerformed* event, the IDE will automatically add an *ActionListener* to the *Exit* button and generate a handler method for handling the listener's *actionPerformed* method.
2. The IDE will open up the Source Code window and scroll to where you implement the action you want the button to do when the button is pressed (either by mouse click or via keyboard). Your Source Code window should contain the following lines:

private void **jButton3ActionPerformed**(java.awt.event.ActionEvent evt) {

//TODO: Add your handling code here:

}

1. We are now going to add code for what we want the Exit Button to do. You will have to type *System.exit(0);*

to the above code, replacing the *//TODO:* line.

Your finished Exit button code should look like this:

private void **jButton3ActionPerformed**(java.awt.event.ActionEvent evt) {

System.exit(0);

}

### Making the Clear Button Work

1. Click the *Design* tab at the top of your work area to go back to the Form Design
2. Right click the *Clear* button (jButton1). From the pop-up menu select *Events > Action > actionPerformed*.
3. We are going to have the *Clear* button erase all text from the jTextFields. To do this, you will add some code like above. Your finished source code should look like this:

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt){

jTextField1.setText("");

jTextField2.setText("");

jTextField3.setText("");

}

The above code changes the text in all three of our JTextFields to nothing, in essence it is overwriting the existing Text with a blank.

### Making the Add Button Work

The Add button will perform three actions.

1. It is going to accept user input from *jTextField1* and *jTextField2* and convert the input from a type String to a float.
2. It will then perform addition of the two numbers and finally,
3. It will convert the sum to a type String and place it in *jTextField3*.

Let’s get started!

1. Click the Design tab at the top of your work area to go back to the Form Design.
2. Right-click the *Add* button (jButton2). From the pop-up menu, select *Events > Action > actionPerformed*
3. We are going to add some code to have our *Add* button work. The finished source code shall look like this:

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt){

// First we define float variables.

float num1, num2, result;

// We have to parse the text to a type float.

num1 = Float.parseFloat(jTextField1.getText());

num2 = Float.parseFloat(jTextField2.getText());

// Now we can perform the addition.

result = num1+num2;

// We will now pass the value of result to jTextField3.

// At the same time, we are going to change

// the value of result from a float to a string.

jTextField3.setText(String.valueOf(result));

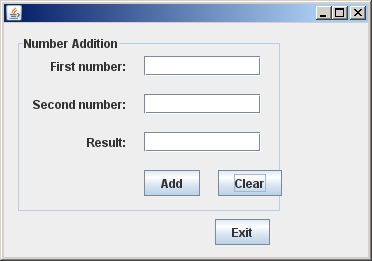
}

You can copy/paste the above code.

Our program is complete and we can now build and run it to see it in action.

## Exercise 4: Running the Program

The final step is to build and run the program.

**To run the program in the IDE:**

1. Choose *Run > Run Project* (Lab8)
2. If you get a window informing you that Project Lab8 does not have a main class set, then you should select Lab8.NumberAdditionUI as the main class in the same window and click the OK button.
3. Your created program is now running.

**To run the program outside the IDE:**

1. Choose *Run > Clean and Build Main Project* (Shift-F11) to build the application JAR file.
2. Using your system's file explorer or file manager, navigate to the *Lab8/dist* directory.
3. Double-click the *Lab8.jar* file.

In this tutorial you learned how you hook up functionality to GUI components with the NetBeans GUI Builder.

## How Event Handling Works

This tutorial showed how to respond to a simple button event. There are many more events you can have your application respond to. The IDE can help you find the list of available events your GUI components can handle:

1. Go back to the file *NumberAdditionUI.java* in the Editor. Click the Design tab to see the GUI's layout in the GUI Builder.
2. Right-click any GUI component, and select Events from the pop-up menu. For now, just browse the menu to see what's there, you don't need to select anything.
3. Alternatively, you can select Properties from the Window menu. In the Properties window, click the Events tab. In the Events tab, you can view and edit events handlers associated with the currently active GUI component.
4. You can have your application respond to key presses; single, double and triple mouse clicks; mouse motion; window size and focus changes. You can generate event handlers for all of them from the Events menu. The most common event you will use is an Action event. (Learn [best practices for Event handling](http://docs.oracle.com/javase/tutorial/uiswing/events/generalrules.html#twokinds) from Sun's [Java Events Tutorial](http://docs.oracle.com/javase/tutorial/uiswing/events/index.html).)

How does event handling work? Every time you select an event from the Event menu, the IDE automatically creates a so-called *event listener* for you, and hooks it up to your component. Go through the following steps to see how the event handling works.

1. Go back to the file NumberAdditionUI.java in the Editor. Click the Source tab to see the GUI's source.
2. Scroll down and note the methods jButton1ActionPerformed(), jButton2ActionPerformed(), and jButton3ActionPerformed() that you just implemented. These methods are called event handlers.
3. Now scroll to a method called initComponents(). If you do not see this method, look for a line that says Generated Code; click the + sign next to it to expand the collapsed initComponents() method.
4. First, note the gray block around the initComponents() method. This code was auto-generated by the IDE and you cannot edit it.
5. Now, browse through the initComponents() method. Among other things, it contains the code that initializes and places your GUI components on the form. This code is generated and updated automatically while you place and edit components in the Design view.
6. In initComponents(), scroll down to where it says

jButton3.setText("Exit");

jButton3.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton3ActionPerformed(evt);

}

});

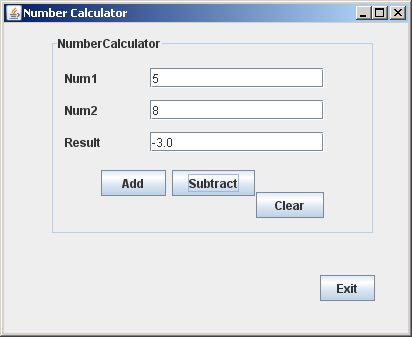
This is the spot where an event listener object is added to the GUI component; in this case, you register an ActionListener to the jButton3. The ActionListener interface has an actionPerformed method taking ActionEvent object which is implemented simply by calling your jButton3ActionPerformed event handler. The button is now listening for action events. Everytime it is pressed, an ActionEvent is generated and passed to the listener's actionPerformed method, which in turn executes the code provided by you in the event handler for this event.

Generally speaking, to be able to respond, each interactive GUI component needs to register to an event listener and needs to implement an event handler. As you can see, NetBeans IDE handles hooking up the event listener for you, so you can concentrate on implementing the actual business logic that should be triggered by the event.

Now, if you still have time, add one more button that will do subtraction (sample solution follow).

This applet is created using the NetBeans GUI.

It is a simple calculator performing addition and subtraction operations.

public class Main {

public static void main(String args[]) {

NumCalcUI objUI = new NumCalcUI();

objUI.setVisible(true);

/\* java.awt.EventQueue.invokeLater ( new Runnable() {

public void run() {

new NumCalcUI().setVisible(true);

}

});

} \*/

}

}

public class NumCalcUI extends javax.swing.JFrame {

/\*\* Creates new form NumCalcUI \*/

public NumCalcUI() {

initComponents();

}

/\*\* This method is called from within the constructor to initialize the form.

\* WARNING: Do NOT modify this code. The content of this method is

\* always regenerated by the Form Editor.

\*/

// <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents

private void initComponents() {

jPanel1 = new javax.swing.JPanel();

jLabel1 = new javax.swing.JLabel();

jLabel2 = new javax.swing.JLabel();

jLabel3 = new javax.swing.JLabel();

jTextField1 = new javax.swing.JTextField();

jTextField2 = new javax.swing.JTextField();

jTextField3 = new javax.swing.JTextField();

jButton1 = new javax.swing.JButton();

jButton2 = new javax.swing.JButton();

jButton4 = new javax.swing.JButton();

jButton3 = new javax.swing.JButton();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

setTitle("Number Calculator");

jPanel1.setBorder(javax.swing.BorderFactory.createTitledBorder("NumberCalculator"));

jLabel1.setText("Num1");

jLabel2.setText("Num2");

jLabel3.setText("Result");

jTextField1.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jTextField1ActionPerformed(evt);

}

});

jButton1.setText("Clear");

jButton1.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton1ActionPerformed(evt);

}

});

jButton2.setText("Add");

jButton2.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton2ActionPerformed(evt);

}

});

jButton4.setText("Subtract");

jButton4.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton4ActionPerformed(evt);

}

});

javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);

jPanel1.setLayout(jPanel1Layout);

jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addGap(10, 10, 10)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)

.addComponent(jButton1, javax.swing.GroupLayout.PREFERRED\_SIZE, 68, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(jButton2, javax.swing.GroupLayout.PREFERRED\_SIZE, 65, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(jButton4)

.addGap(69, 69, 69))

.addGroup(javax.swing.GroupLayout.Alignment.LEADING, jPanel1Layout.createSequentialGroup()

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jLabel1)

.addComponent(jLabel2)

.addComponent(jLabel3))

.addGap(49, 49, 49)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

.addComponent(jTextField3)

.addComponent(jTextField2)

.addComponent(jTextField1, javax.swing.GroupLayout.DEFAULT\_SIZE, 174, Short.MAX\_VALUE)))))

.addContainerGap(46, Short.MAX\_VALUE))

);

jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addContainerGap()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jLabel1)

.addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jLabel2)

.addComponent(jTextField2, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jLabel3)

.addComponent(jTextField3, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addGap(18, 18, 18)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jButton2)

.addComponent(jButton4))

.addContainerGap(34, Short.MAX\_VALUE))

.addGroup(jPanel1Layout.createSequentialGroup()

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(jButton1)

.addContainerGap())))

);

jButton3.setText("Exit");

jButton3.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton3ActionPerformed(evt);

}

});

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addGap(46, 46, 46)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)

.addComponent(jButton3)

.addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addContainerGap(33, Short.MAX\_VALUE))

);

layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addContainerGap()

.addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 40, Short.MAX\_VALUE)

.addComponent(jButton3)

.addGap(32, 32, 32))

);

pack();

}// </editor-fold>//GEN-END:initComponents

private void jTextField1ActionPerformed (java.awt.event.ActionEvent evt) { //GEN- FIRST:event\_jTextField1ActionPerformed

jTextField1.setText("");

jTextField2.setText("");

jTextField3.setText("");

}//GEN-LAST:event\_jTextField1ActionPerformed

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_jButton2ActionPerformed

// First we define float variables.

float num1, num2, result;

// We have to parse the text to a type float.

num1 = Float.parseFloat(jTextField1.getText());

num2 = Float.parseFloat(jTextField2.getText());

// Now we can perform the addition.

result = num1+num2;

// We will now pass the value of result to jTextField3.

// At the same time, we are going to change the value of *result* from float to string.

jTextField3.setText(String.valueOf(result));

}//GEN-LAST:event\_jButton2ActionPerformed

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_jButton3ActionPerformed

System.exit(0);

}//GEN-LAST:event\_jButton3ActionPerformed

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_jButton1ActionPerformed

jTextField1.setText("");

jTextField2.setText("");

jTextField3.setText("");

}//GEN-LAST:event\_jButton1ActionPerformed

private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_jButton4ActionPerformed

// First we define float variables.

float num1, num2, result;

// We have to parse the text to type float.

num1 = Float.parseFloat(jTextField1.getText());

num2 = Float.parseFloat(jTextField2.getText());

// Now we can perform the addition.

result = num1-num2;

// We will now pass the value of result to jTextField3.

// At the same time, we are going to change the value of result from float to string.

jTextField3.setText(String.valueOf(result));

}//GEN-LAST:event\_jButton4ActionPerformed

// Variables declaration - do not modify//GEN-BEGIN:variables

private javax.swing.JButton jButton1;

private javax.swing.JButton jButton2;

private javax.swing.JButton jButton3;

private javax.swing.JButton jButton4;

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel2;

private javax.swing.JLabel jLabel3;

private javax.swing.JPanel jPanel1;

private javax.swing.JTextField jTextField1;

private javax.swing.JTextField jTextField2;

private javax.swing.JTextField jTextField3;

// End of variables declaration//GEN-END:variable

}

# If you still have time, go to *Designing a Swing GUI in NetBeans IDE* to learn more about creating a GUI with Swing components (<http://netbeans.org/kb/docs/java/quickstart-gui.html>).