

## Adding Functionality to Buttons: A Beginners Guide

Contributed by [Saleem Gul](#) and Tomas Pavek, maintained by Ruth Kusterer

This beginner tutorial teaches you how to build a simple GUI with back-end functionality. You will learn how to develop a graphical user interface and how to add functionality to the buttons and fields used. A basic understanding of the Java Programming Language is required.

This document takes you through the fundamental concepts of GUI creation and takes the approach taken in many self learning books. We will work through the layout and design of a GUI and add a few Buttons and Text Boxes. The Text Boxes 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.

**Expected duration: 15 minutes**

### Tutorial Exercises

- [Exercise 1: Creating a Project](#)
- [Exercise 2: Building the Front End](#)
- [Exercise 3: Adding Functionality](#)
- [Exercise 4: Running the Program](#)
- [How Event Handling Works](#)
- [Next Steps](#)

### Prerequisites

- The user should have basic skills in the Java language.
- The user should be comfortable with using the NetBeans IDE.
- The user should have already completed the tutorial [GUI Building in Netbeans IDE 6.0](#).

### Software Needed for This Tutorial

The NetBeans GUI Builder comes bundled with NetBeans IDE, and no special setup is necessary.

Make sure you have the following software installed on your computer:

- NetBeans IDE 6.0 ([download](#))
- In order to install and run NetBeans IDE 6.0, you also require the Java SE Development Kit (JDK) version 5.0 or higher. You can get a version of JDK for your platform at: <http://java.sun.com>.

## Exercise 1: Creating a Project

---

The first step is to create a Project, we will name our project NumberAddition.

1. Choose File > New Project. Alternately, you can click the New Project icon in the IDE toolbar.
2. In the Categories pane, select the General node. In the Projects pane, choose Java Application. Click Next.
3. Enter NumberAddition in the Project Name field and specify a path e.g. in your home directory as the project location.
4. Ensure that the Set as Main Project checkbox is selected. Deselect Create Main Class if it is selected.
5. Click Finish.

## Exercise 2: Building the Front End

---

After creating the new application, you may have noticed that the Source Packages folder in the Projects window contains an empty `<default package>` node. 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 and place the container in a new package.

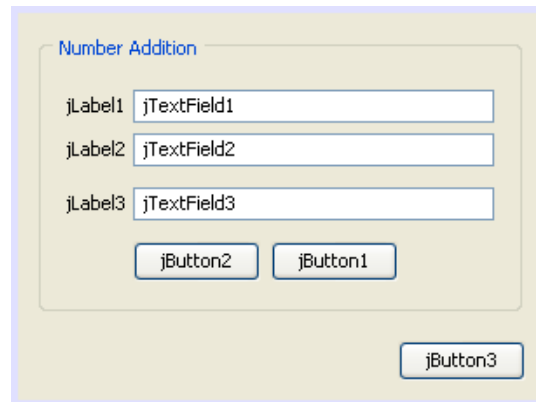
### Create a JFrame container

1. In the Projects window, right-click the NumberAddition node and choose New > JFrame Form.
2. Enter NumberAdditionUI as the Class Name.
3. Enter `my.NumberAddition` as the package.
4. Click Finish.

The IDE creates the `NumberAdditionUI` form and the `NumberAdditionUI` class within the `NumberAdditionUI.java` application, and opens the `NumberAdditionUI` form in the GUI Builder. Notice that the `my.NumberAddition` package replaces the default package.

### Adding Components: Making of the Front End

Next we are going to populate our application's front end with one `JPanel`, and three `JLabels`, `JTextFields` and `JButtons` each. The `JFrame` with the aforementioned components will look like this.



Select `Windows > Palette` to open the Palette if you don't see it. You use the Palette to drag and drop UI components to the design area.

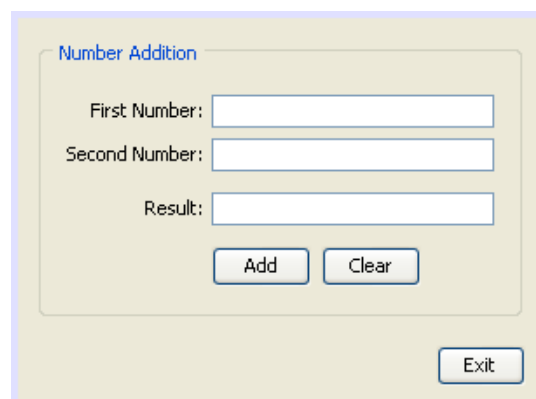
1. Start by selecting a `JPanel` from the Palette and drop it onto the `JFrame`.
2. While the `JPanel` is highlighted, go to the Properties window and click the "..." button next to `Border` to choose a border style.
3. In the `Border` dialog, select `Titled Border` from the list, and type in `Number Addition` in the `Title` field. Click `OK` to save the changes and exit the dialog.
4. You should now see an empty titled `JFrame` that says `Number Addition` like in the screenshot. Look at the screenshot and add three `JLabels`, three `JTextFields` and three `JButtons` as you see above.

### Renaming the Components

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

1. Double-click on `jLabel1` and change the text property to `First Number`
2. Double-click on `jLabel2` and change the text to `Second Number`
3. Double-click on `jLabel3` and change the text to `Result`
4. Double-click on `jTextField1` and remove the sample text in the text field. You may have to resize the `jTextField1` to its original size. Repeat this step for `jTextField2` and `jTextField3`.
5. Double-click on `jButton1` and rename it `Clear`
6. Double-click on `jButton2` and rename it `Add`
7. Double-click on `jButton3` and rename it `Exit`

Your Finished GUI should now look like this:



## Exercise 3: Adding Functionality

In this exercise we are going to give 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. Let's begin.

### Making the Exit Button Work

In order to give function to the buttons, we have to assign an event handler to each, responding to an event. In our case we want to know when the button is pressed, either by mouse click or via keyboard. So we will use ActionListener responding to ActionEvent.

1. Right Click on 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:
}
```

3. 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 on the Design tab at the top of your work area to go back to the Form Design
2. Right Click on 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 jTextField3. 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 jTextField3 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.

Lets get started!

1. Click on the Design tab at the top of your work area to go back to the Form Design.
2. Right-click on the Add button (jButton3). 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));
}

```

Our Program is now complete 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.

1. Click on Build in the Main Menu and Choose --> Build Main Project.
2. When the Build output is finished, click on Run in the Main Menu and Choose --> Run Main Project
3. If you get a window informing you that Project NumberAddition does not have a main class set, then you should select my.NumberAddition.NumberAdditionUI as the main class in the same window and click the OK button.
4. Your created program is now running.

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

## 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 Matisse.
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](#) from Sun's [Java Events Tutorial](#).)

How does event handling work? Everytime 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 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 blue 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 to action events. Everytime it is pressed an `ActionEvent` is generated and passed to the listener's `actionPerformed` method which in turn executes code that you provided 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.

## Next Steps

---

- [Matisse GUI Builder - Frequently Asked Questions](#)
- [Best practices for Event handling](#) from Sun's [Java Events Tutorial](#).
- [Building a Java Desktop Database Application](#)
- Code Samples can be found at [Java Almanac: Java Events](#)

If you have questions or need support, and want to keep yourself informed on the latest developments on the NetBeans IDE features, join the [nbusers@netbeans.org](mailto:nbusers@netbeans.org) mailing list by sending an empty message to [nbusers-digest-subscribe@netbeans.org](mailto:nbusers-digest-subscribe@netbeans.org).

[Feedback](#)