Computer Programming 2 : Laboratory Activity - Building a GUI with Netbeans

Objective: To be able to create a simple graphical user interface and add simple back-end functionality.

Creating a Project

1. Open Netbeans Application> Choose File > New Project . Or 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 NumberAddition in the Project Name field and specify a path, for example, in your home directory, as the project location.

(Optional) Select the Use Dedicated Folder for Storing Libraries checkbox and specify the location for the libraries folder.

4. Deselect the Create Main Class checkbox if it is selected.
5. Click Finish.

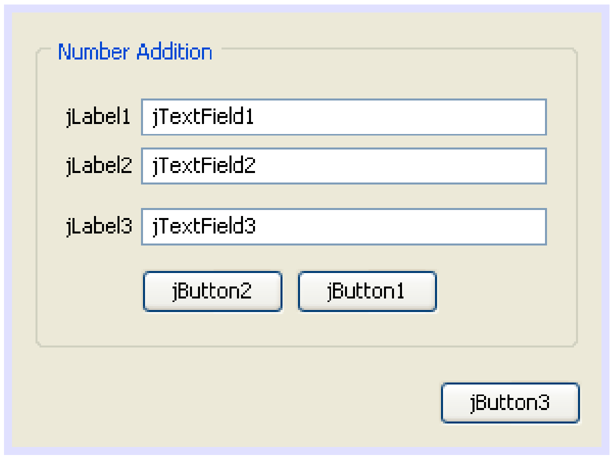
Build the Front End

Create a JFrame container

1. In the Projects window, right-click the NumberAddition 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 my.numberaddition as the package.
5. Click Finish.

Add Components

Use the Palette to populate the application's front end with a JPanel. Then add three JLabels, three JTextFields, and three JButtons. Once you are done dragging and positioning the aforementioned compo nents, the JFrame should look something like the following screenshot.



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

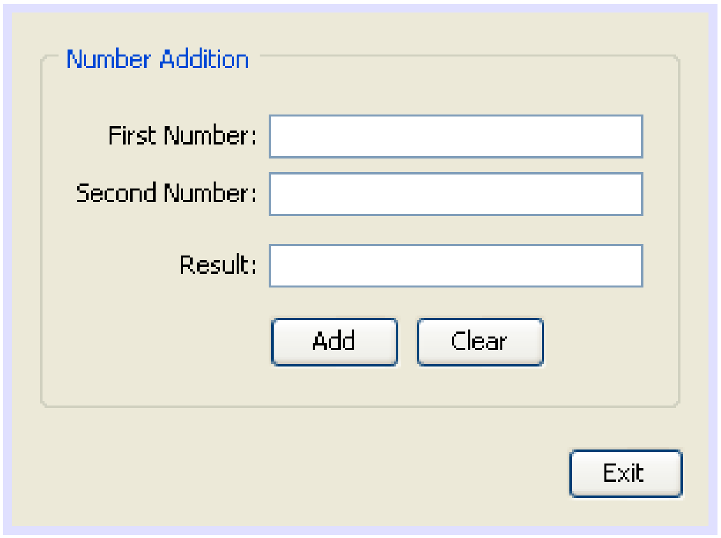
1. Start by selecting a Panel from the Swing Containers category on Palette and drop it onto the JFrame.
2. While the JPanel is highlighted, go to the Properties window and click the ellipsis (...) button next to Border to choose a border style.
3. In the Border dialog, select TitledBorder from the list, 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 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.

Rename the Components

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 right-clicking the text field and choosing Edit Text from the popup menu. 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 following screenshot:



Add Functionality

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 – creating simple calculator.

Make the Exit Button Work

In order to give function to the buttons, we have to assign an event handler to each to respond to events. 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 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: }
   ```

   To add code for what we want the Exit Button to do, replace the TODO line with System.exit(0);. Your finished Exit button code should look like this:

   ```
   private  void  jButton3ActionPerformed(java.awt.event.ActionEvent
   evt) { System.exit(0); }
   ```

The Clear Button

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. The Clear button erase all text from the jTextFields. To do this, you will add some code like the previous example. Your finished source code should look like this:

```
private  void  jButton1ActionPerformed(java.awt.event.ActionEvent
evt){
jTextField1.setText("");
jTextField2.setText("");
jTextField3.setText(""); }
```

The Add Button

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.
3. And finally, it will convert the sum to a type String and place it in `jTextField3`. To do so,

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));
}
```

Additional Tasks: Adding more Buttons with Functionality

1. Add the Multiply, Subtract and Divide buttons.
2. Add codes to each of the added buttons above with the following actions ;
   Multiply Button : performs multiplication of two numbers
   Subtract Button: performs subtraction of the two numbers
   Divide Button : performs division of two numbers.

Run and Test your application.

To run the program outside of 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 ` NumberAddition/dist` directory.
3. Double-click the `NumberAddition.jar` file.

After a few seconds, the application should start.

You can also launch the application from the command line.

To launch the application from the command line:

1. On your system, open up a command prompt or terminal window.
2. In the command prompt, change directories to the `NumberAddition/dist` directory.
3. At the command line, type the following statement:

```
java -jar  NumberAddition.jar
```

Event Handling

This activity 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.

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 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 reads

```
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.