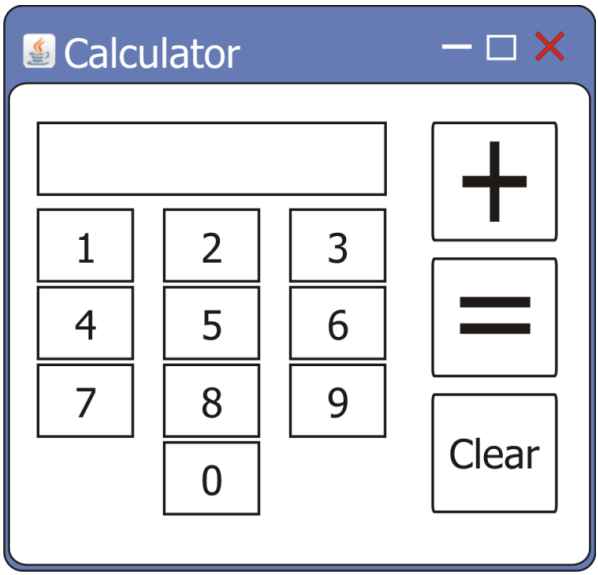
JAVA FORM

You don't have to output everything to a terminal window in Java. In this topic, you'll be writing a calculator program that makes use of forms. The form will have buttons and a textbox. We'll start off with a very simple calculator that can only add up, and then extend its capabilities so that it can divide, subtract, and multiply. The calculator will look like this:



The Significance of NetBeans when dealing with Forms

Developing a Graphic User Interface (GUI) using Java can be an art-form in itself, as there's quite a lot to get used to: components, containers, layout managers, and a whole lot more. The NetBeans development environment, however, has greatly simplified the creation of forms, and we'll use this to drag and drop controls onto a frame. Rather than go through reams of GUI Theory, we'll just jump straight into it.

Create a new project for this by clicking `File>New Project` from the NetBeans menu at the top. Select `Java>Java Application` from the list boxes, and then click the `Next` button.

On step 2 of the wizard, type `MyCalculator` as the project name. at the bottom, uncheck "`Create main class`." This is because a main method will be created for us by NetBeans when we add a form.

Click the Finish button, and NetBeans will create the project. Have a look at the Projects area on the left

in NetBeans and you will see your new projects directory.

Normally, there's a java file under the Source Packages name, but because we unchecked the "`Create main class`" box, there's no java class file there.

What we'll do is to add a form to the project. When the form is created, it will be created in its own java class file.

To add a form, right click the project name in the projects window. A menu will appear. Select `New>JFrame Form` from the menu. When you do, a wizard should appear. There you would be asked for a name for your class, and a package name. We've already created the project, and called it `MyCalculator`. The package name and class will go into the project. So for your `class name` type `JavaCalculator`, which is in the `jCalculator` package, which is in the `MyCalculator` project.

Form Views

When the wizard has finished, it will create a blank form in the main window.

Notice the two buttons at the top, `Source` and `Design`. You click the `Source` button to reveal the code behind the form. In the code, you would notice that main method has been created. When the program starts it is the main method that will be called. It's creating a new object from our form, and setting its visible property to true.

When you click on the `Design` button, you'll see your blank form again. To the right, you'll notice two areas: a Palette with a lot of controls in it, and a Properties area. However, if you can't see them, click `Windows > IDE Tools > Palette` and click `Windows > IDE Tools > Properties`.

Text Box in Java Form

Controls in the NetBeans palette can be dragged onto a form. Click on `Text Field` to select it. Now hold your

left mouse button down on the `Text Field`. Keep it held down and drag the control onto the form.

Notice that the default name for the text field is `jTextField1`. To change this, look at the Navigator area in the bottom left with the text field selected, right click on `jTextField1`. From the menu that pops up, select `Change Variable Name`. When you click on `Change Variable Name`, a dialogue box would appear. Type a new name for the `Text field`. After typing the new name, click OK.

Now in the code, you would notice the new name of your new private field variable.

Buttons in Java Forms

You add a button to a form in the same way you do for text fields– drag and drop. However as we are going to be adding lots of buttons to a control called a Panel. If you need to move the buttons, you can just move the Panel instead. All the buttons will then be moved along with the Panel.

After putting the panel control on your form, you can now begin putting your button controls. The default name of the first button is jButton1. Change the name just like you did for the text field to btnOne. The new variable name is btnOne and it is a jButton object, which is a Swing control. In the code you would see these lines of code that have been set up, even for the panel.

The text on the buttons could be edited in the properties column of the NetBeans window. You could even set the height and width of any control in the properties column. In essence, any property could be edited there.

Java Form Events

In programming terms, an event is when something special happens. For forms, this means things like buttons being clicked, the mouse moving, text being entered into text fields, the programming closing and a whole lot more.

Events are objects in Java. They come from a series of classes stored in `java.util.EventObject`.

When a button is clicked, the button is said to be the source of the event. But the very act of clicking generates an event object. The event object then looks for an object that is listening out for, say, a mouse click, or a keystroke, or any other event that can happen on a form. A text field, for example, can be listening out for keystrokes. A drop down box could be listening out for which item in the list was clicked.

Different objects (sources) fire different events. For a button, the event that is fired is the ActionListener. For a text field, it is the `keyEvent`. Think of it like this: the event object is responsible for passing messages back and forward between form objects that have had something happen to them, and objects that are waiting for something to happen.