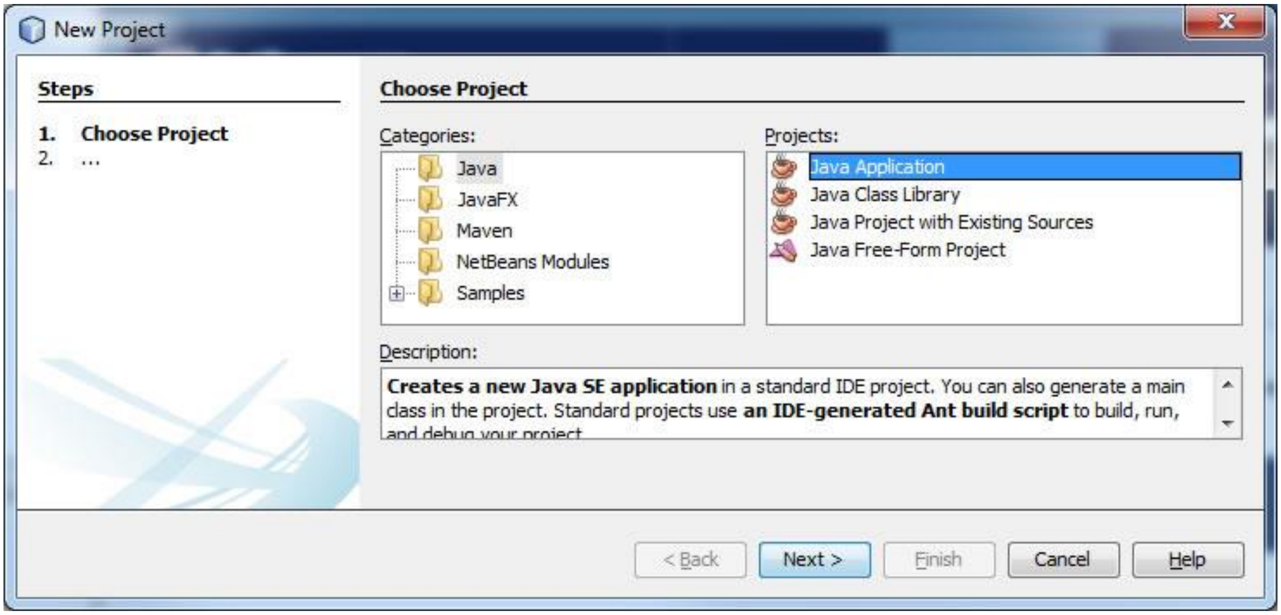
In this activity, you will learn how to create a graphical user interface, (GUI). You will learn how to accept information on a GUI from an input box, and how to output information on that same GUI. In NetBeans version 6.9 or greater, there are tools to simplify the GUI making process.
Source: https://netbeans.org/kb/docs/java/gui-functionality.html

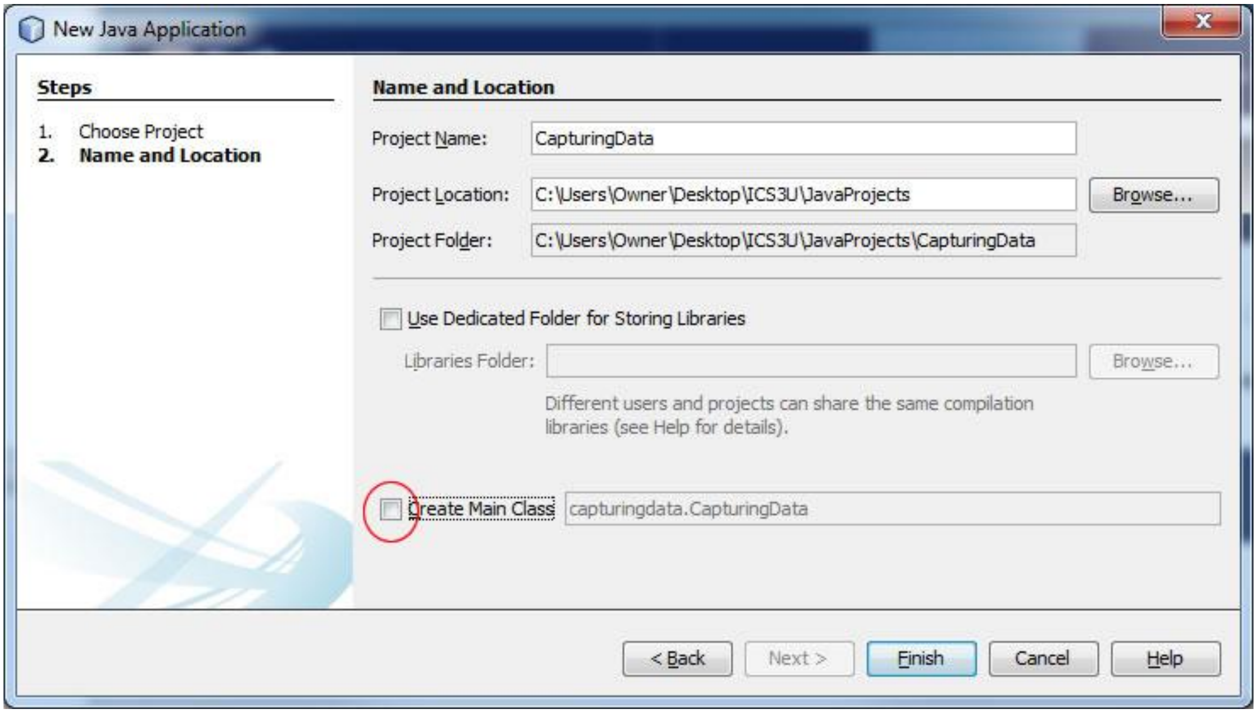
1. Creating a Project

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*.



Creates a new Java SE application in a standard IDE project. You can also generate a main class in the project. Standard projects use **an IDE-generated Ant build script** to build, run, and debug your project

3. Enter **CapturingData** in the Project Name field and specify your project location. You should be using your Unit 2 Activity 2 folder. Click Nex*t*.
4. Deselect the Create Main Class checkbox if it is selected.
5. Click Finish.



New Java Application

Steps

1. Choose Project
2. **Name and Location**

Name and Location

Project Name: CapturingData

Project Location: C:\Users\Owner\Desktop\ICS3U\JavaProjects

Project Folder: C:\Users\Owner\Desktop\ICS3U\JavaProjects\CapturingData

□ Use Dedicated Folder for Storing Libraries

Libraries Folder:

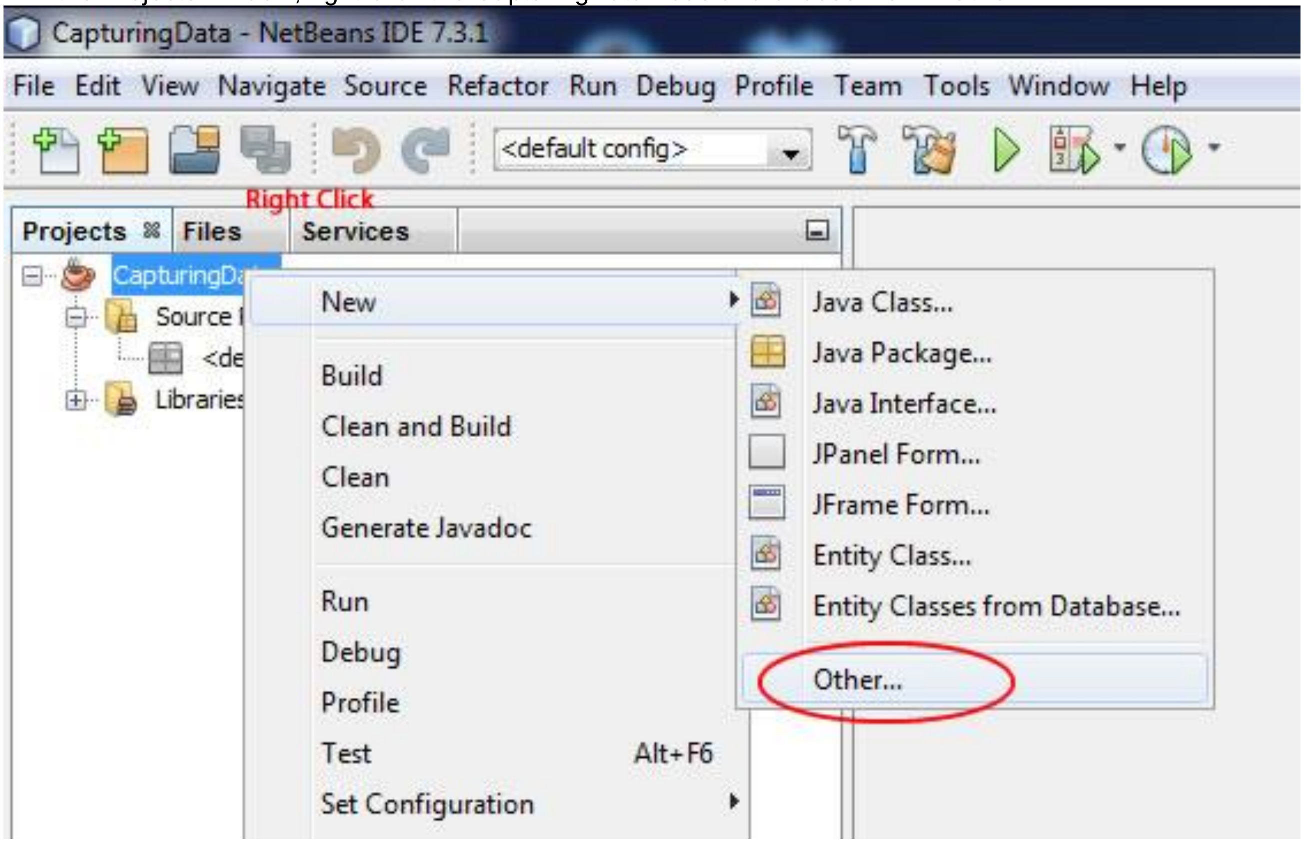
Different users and projects can share the same compilation libraries (see Help for details).

□ Create Main Class capturingdata.CapturingData

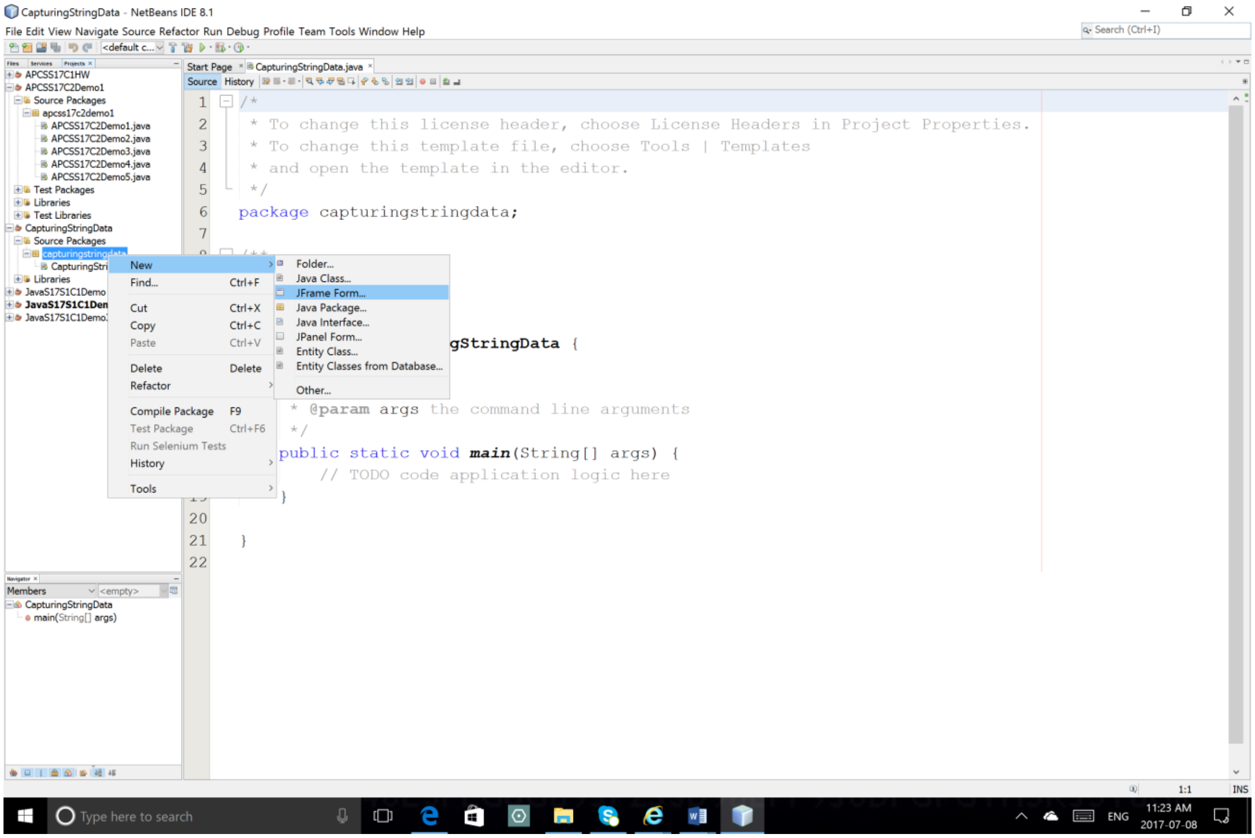
2. Building the Front End

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

1. In the Projects window, right-click the CapturingData node and choose New > Other.



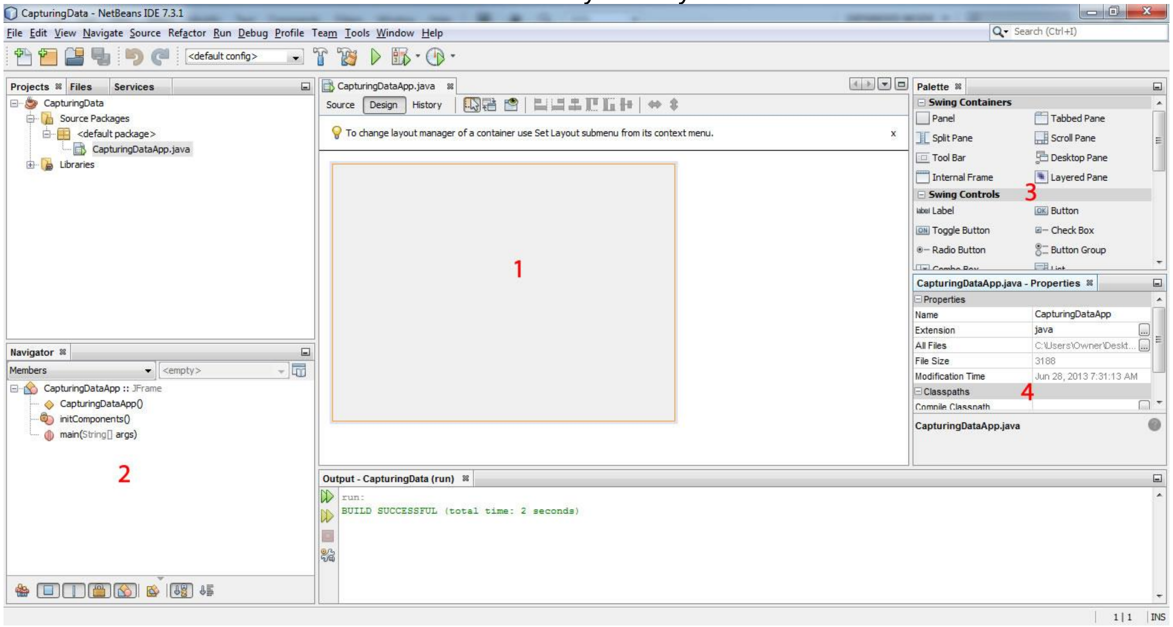
2. In the New File dialog box, choose the Swing GUI Forms category and the JFrame Form file type. Click *Next*.



```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template file in the editor.
 */
package capturingstringdata;

gStringData {

 * @param args the command line arguments
 */
public static void main(String[] args) {
    // TODO code application logic here
}

}
```

3. Enter **CapturingDataApp** as the class name
4. Click *Finish*.

You will notice that your NetBeans IDE looks a little different now than it did in previous programs you created. Since you are creating a program with a GUI, NetBeans attempts to help you by switching to the GUI Builder's interface. This interface has four key areas you need to know about.



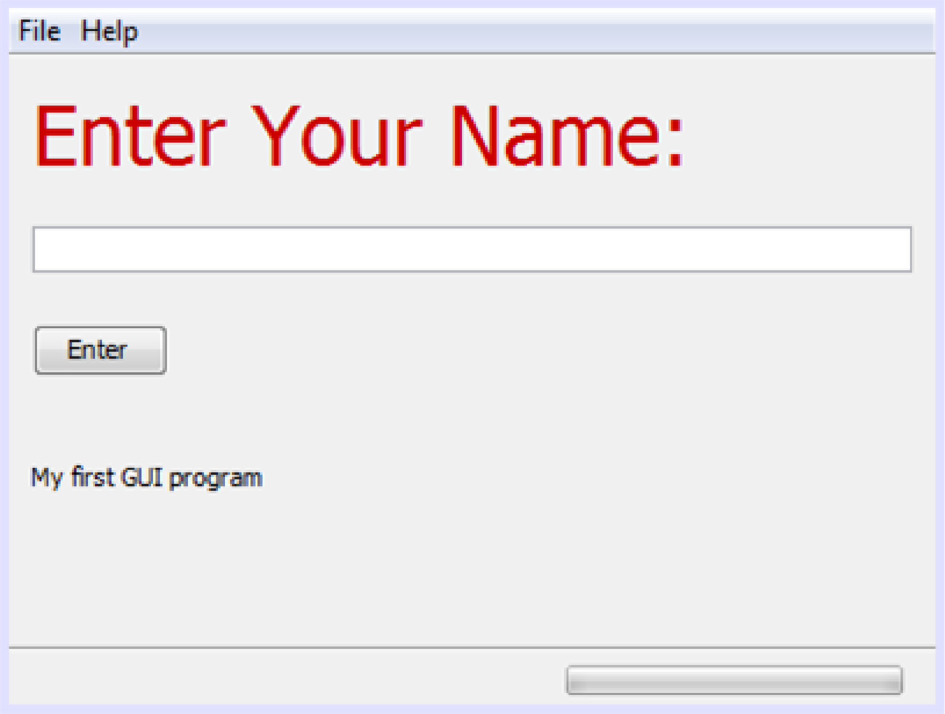
To change layout manager of a container use Set Layout submenu from its context menu.

```
run:
BUILD SUCCESSFUL (total time: 2 seconds)
```

CapturingDataApp.java - Properties	
Name	CapturingDataApp
Extension	java
All Files	C:\Users\Owner\Deskt...
File Size	3188
Modification Time	Jun 28, 2013 7:31:13 AM

1. **Design Area**: This is the main window for creating your GUI forms. A basic form will already be created for you when you start a new project. If you choose to, you can resize that initial form. You can also use the **Source** and **Design** buttons in the top left corner of the Design Area to switch between the graphical view of your program and the code that runs it.
2. **Navigator**: This area gives a list of all the components that you have added to your application. It will indicate for you what component you are working on, as well as allow you the ability to organize your components and make adjustments to them.
3. **Palette**: From here you can see a list of components that you can add to your application. To add a component to your application you simply drag it from here onto your form in the Design Area.
4. **Properties Window**: This will show a list of all the properties of the component in your application that you currently have selected.

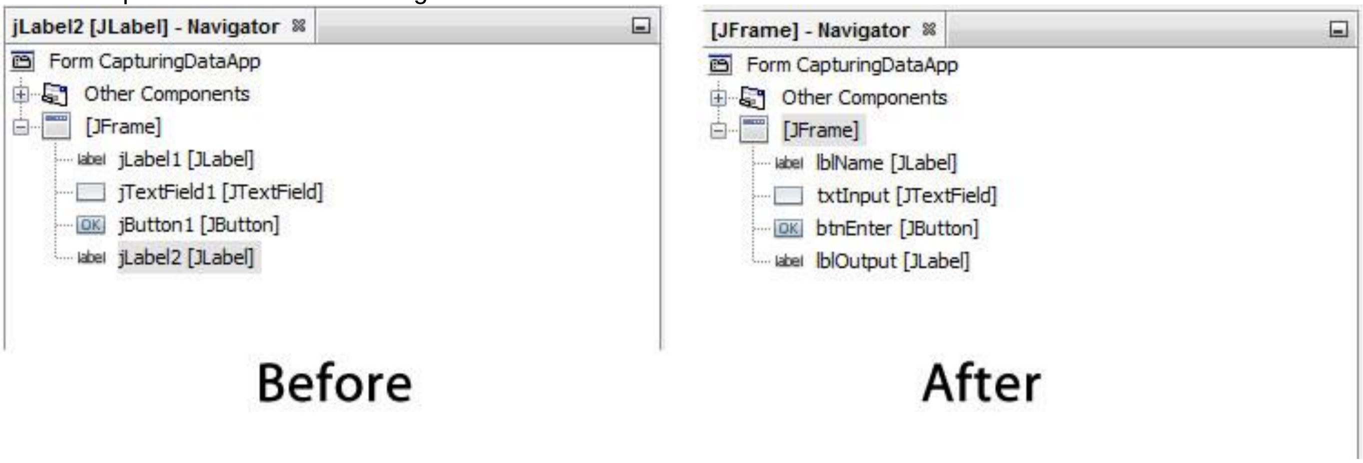
Building your GUI



1. Drag a **Label** from the **Palette** to your application window. Place it near the top left. Double click on it and change the text to read, "**Enter Your Name:**". Now take a look at the **Properties** window. Find the **font** property. Click the "**…**" button next to it and select a font size of 36. The next property down is the **foreground** property. Click the "**…**" button next to it and change the colour to a deep red.
2. Drag a **Text Field** from the **Palette** to your application form and place it right under your previous label. Notice how NetBeans will help you line up your new component with previous components by showing lines on the screen to indicate position. In the **Properties** window find the **text** property and remove any text you find there so that your field is blank to start with.
3. Drag a **Button** from the **Palette** to your application window and place it as shown in the image above. In the **Properties** window find the **text** property and change it to "**Enter**".
4. Drag another **Label** from the **Palette** to your application and place it under the button. Change the **text** property to "My first GUI program".

You should now have an application window that looks like the image above. In the process of creating the application so far, you have had to alter some of the properties of the various components. Take a moment to select one of those components and then scroll through the list of properties to see what else is available. At this time, there will be a lot of properties that won't mean anything to you, but there will be others that you can probably guess what they do based on their names. Feel free to play around a bit. When you are done looking around, continue on with the next set of instructions.

5. Take a look at the **Navigator** and you will see that all the components you added to your application form, are listed there. However, the names of these components are very generic and don't match how they are used. You need to rename them so that they reflect their use. The component names should be changed to: **lblName**, **txtInput**, **btnEnter**, and **lblOutput**. Below is an image of what the **Navigator** shows before changing the names, and what the **Navigator** will show after you have changed the names. You can change the names by *Right Clicking* on the component and select "Change Variable Name..."



*****Naming Controls best practice:*****
We will attach a prefix to each of the component name. It is also recommened that you use **Camel Notation** (lowercase first letter and use uppercase for the first letter of each subsequent words) as well. Please note that you don't exactly have to follow my prefixes. You can come up with your own, as long as it is consistent throughout your program.
Using the following chart:

Component Type	**prefix**	**Variable Name Example**
Label	lbl	lblFlag, lblFirstAnswer, lblMessage
Button	btn	btnNumber, btnUser, btnChoice
Radio Button	rad	radFirstChoice, radGrade
Button Group	grp	grpClass, grpFruit, grpRacers
List	lst	lstClass, lstValueOfTemp
Text Field	txt	txtPopulation, txtNameOfChild
Menu Bar	mnu	mnuPrice, mnuPriceOfGrain, mnuSpeed

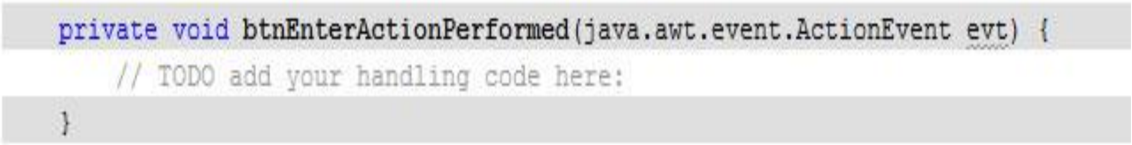
etc...

Adding Functionality
Now that you have created an application with a variety of components, we need to make it do something. What we will do is have the program take a person's name from the input box and then display a new message using that person's name in the output box.
Components are **event driven**. That means that they do things when certain events take place. For example, when the user presses a button with their mouse, (the **event**), the program displays a message showing the date, (the **action**).

1. Right click on the **Enter** button. From the menu that appears select *Event, Action, ActionPerformed*. This will create what is called an *ActionListener* for the **Enter** button. This means

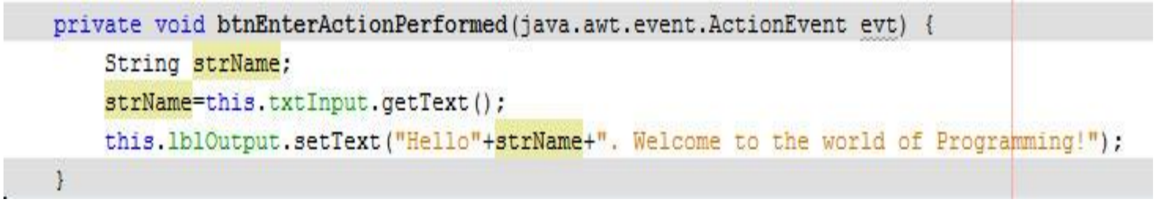
that the program will be *listening* for the *ActionPerformed* event to occur, and if it does, it will run the set of instructions that you give it within that event. The *ActionPerformed* event listens for the button to be pressed by either the mouse or the keyboard.

2. At this point the IDE will display the code that goes along with your application. Notice that there are quite a few lines of code created already. This code controls the main window, as well as all the components that you added. NetBeans automatically generated these for you and you don't need to alter them. The specific lines of code you are looking for are as follows:



```
private void btnEnterActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}
```

3. We are now going to add code that controls what the **Enter** button does. Replace the TODO line from above with the code shown below. You are probably not familiar with the keyword "**this**". It is optional to use. What it does is that it refers to the current form that you are working on. It helps you by listing a drop down menu so that it's easier to select the available component.



```
private void btnEnterActionPerformed(java.awt.event.ActionEvent evt) {
    String strName;
    strName=this.txtInput.getText();
    this.lblOutput.setText("Hello"+strName+". Welcome to the world of Programming!");
}
```

Congratulations. You have now just completed your first program with a GUI. It is now time to run it and see what happens. To do so, simply press the **Run** button on the toolbar or press **F6**.

Examining the Code

In the previous section you added code that controlled what the **Enter** button did when pressed. Let's take a step-by-step look at that code now.

String strName;

This simply created a new variable called **name** that would hold a **string** value.

strName = this.txtInput.getText();

The **getText()** method which is part of the **JTextField** class simply retrieves the text that has been entered into a components **text** property. In this line the **getText()** method is used to get the **text** that the user enters into the **txtInput** component and assigns that to the **strName** variable you previously created.

this.lblOutput.setText("Hello " + strName + ". Welcome to the world of programming!");

The **setText()** method takes the string value that is within the brackets and assigns it the **text** property of a component. In our case the string is being assigned to the **lblOutput** component. There are three parts to the string being assigned. First the word “Hello ” is put into the **text** property. Then the contents of the **strName** variable will be added to the end of the **text** property. Finally the words “Welcome to the world of programming!” are added to the end of the **text** property.

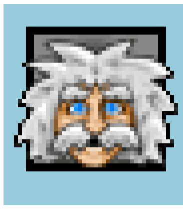
Modifying the Code

Now that you have a running program with a working GUI, it is time to experiment. Try changing the properties of some of the components. Try adding new components. Play around with the code and see what you can do.

A good place to start is by adding a new button and labeling it **Exit**. Now add an event to this button that causes the program to close when the button is pressed. Use the following code inside your *ActionPerformed* method to end the program:

System.exit(0);

Enrichment



Create another input box where the user can enter “Yes” if they have ever programmed before and “No” if they haven't. Depending on what the user enters, alter the output message appropriately. To accomplish this you will need to use if statements. You can find out how to do this by either doing an Internet search or looking ahead in this course.