

# SIMPLE CALCULATOR PROJECT

RICHARD C. CUPAL, LPT  
(Sepiroth X)

## Follow my socials:

Facebook: <https://facebook.com/tsadiqz3>

Instagram: <https://instagram.com/sepiroth.x/>

Twitter: <https://twitter.com/sepirothx000>

Github: <https://github.com/sepiroth-x>

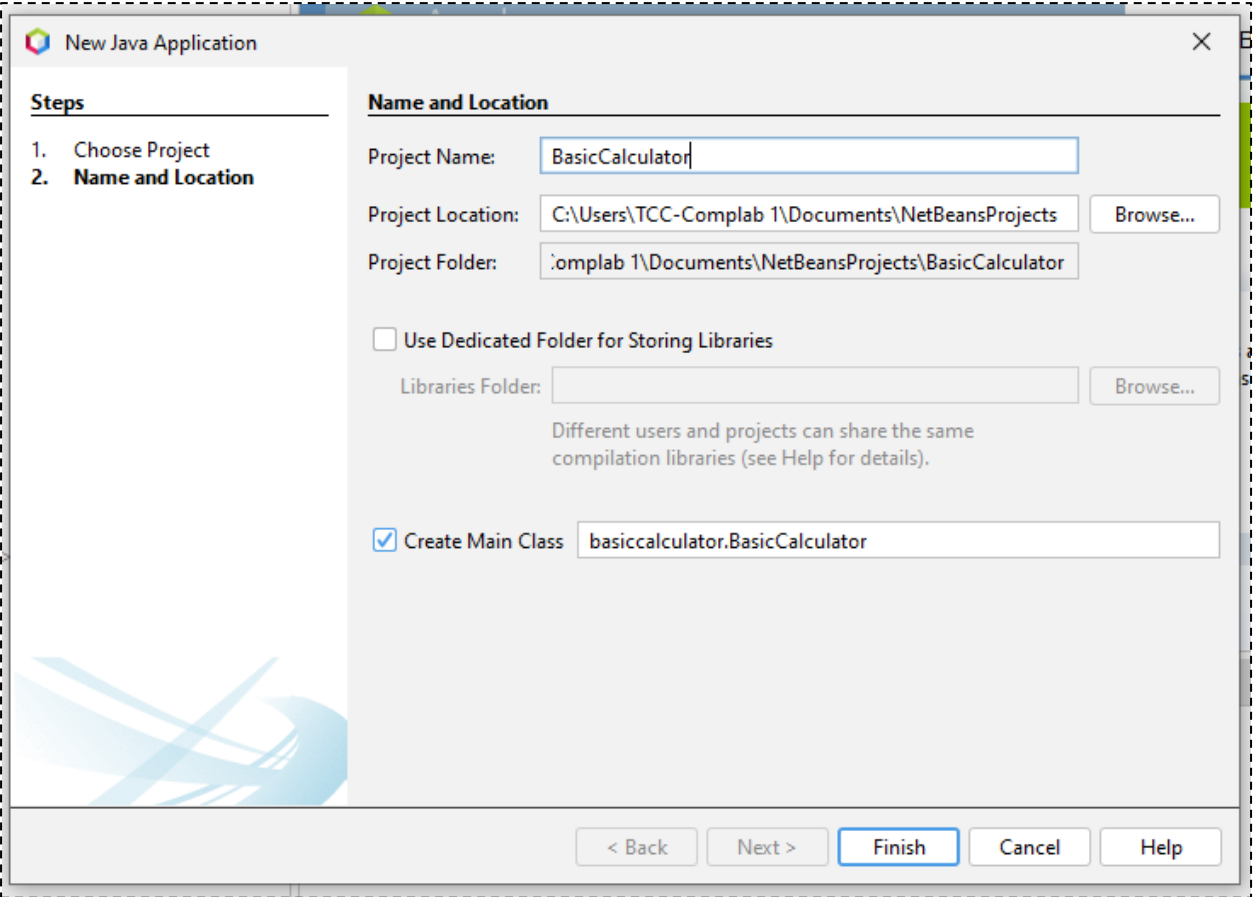
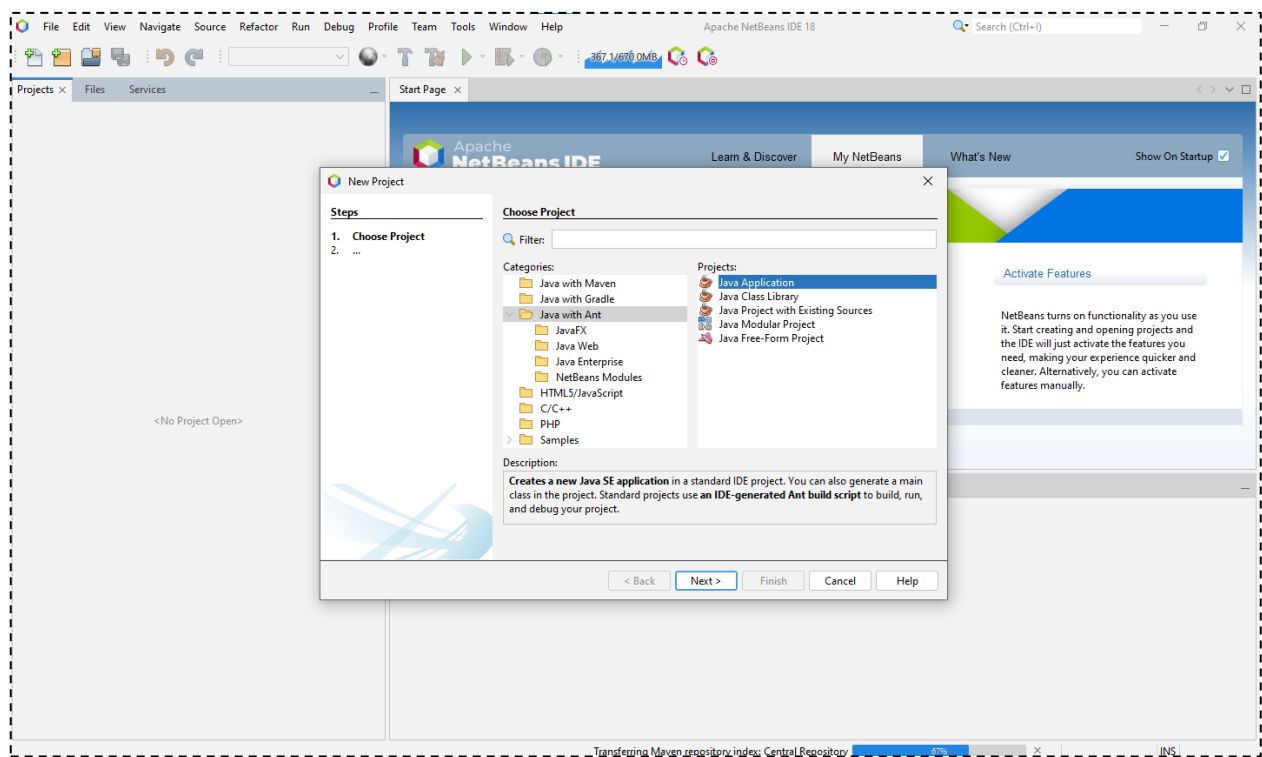
Patreon: <https://www.patreon.com/vajrayogiii>

Gcash: +639150388448

Email: [richard.cupal@ifamsocial.com](mailto:richard.cupal@ifamsocial.com) , [chardy.tsadiq@gmail.com](mailto:chardy.tsadiq@gmail.com)

# DESIGN & LAYOUT

## 1. START WITH NEW ANT PROJECT



2. Enter a project name. **Take note: Project name should start with capital letter and no spaces in between words.**

The screenshot shows the 'New Java Application' dialog box with the 'Name and Location' tab selected. The 'Project Name' field contains 'BasicCalculator'. The 'Project Location' field shows the path 'C:\Users\TCC-Complab 1\Documents\NetBeansProjects' with a 'Browse...' button. The 'Project Folder' field shows the path '.omplab 1\Documents\NetBeansProjects\BasicCalculator'. There is an unchecked checkbox for 'Use Dedicated Folder for Storing Libraries' with a 'Libraries Folder' field and a 'Browse...' button. Below this is a note: 'Different users and projects can share the same compilation libraries (see Help for details)'. There is a checked checkbox for 'Create Main Class' with the field 'basiccalculator.BasicCalculator'. At the bottom are buttons for '< Back', 'Next >', 'Finish' (highlighted), 'Cancel', and 'Help'.

This screenshot is identical to the one above but includes a red annotation. A red box highlights the 'Browse...' button next to the 'Project Location' field. A red line connects this button to the text 'SAVE LOCATION' written in large red capital letters. The rest of the dialog box, including the project name, folder paths, checkboxes, and bottom buttons, remains the same.

3. Click **'Browse'** if you want to change the save location of your project. Click **finish** once done to start working on the project.

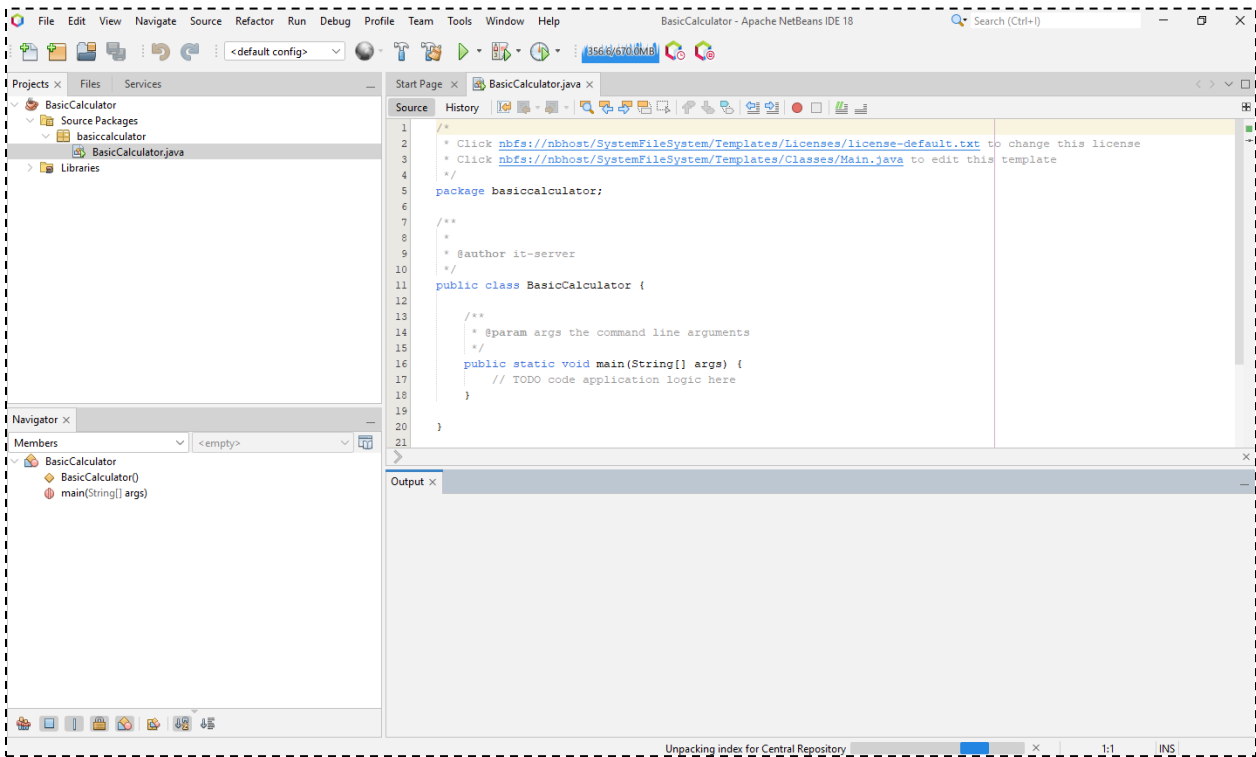
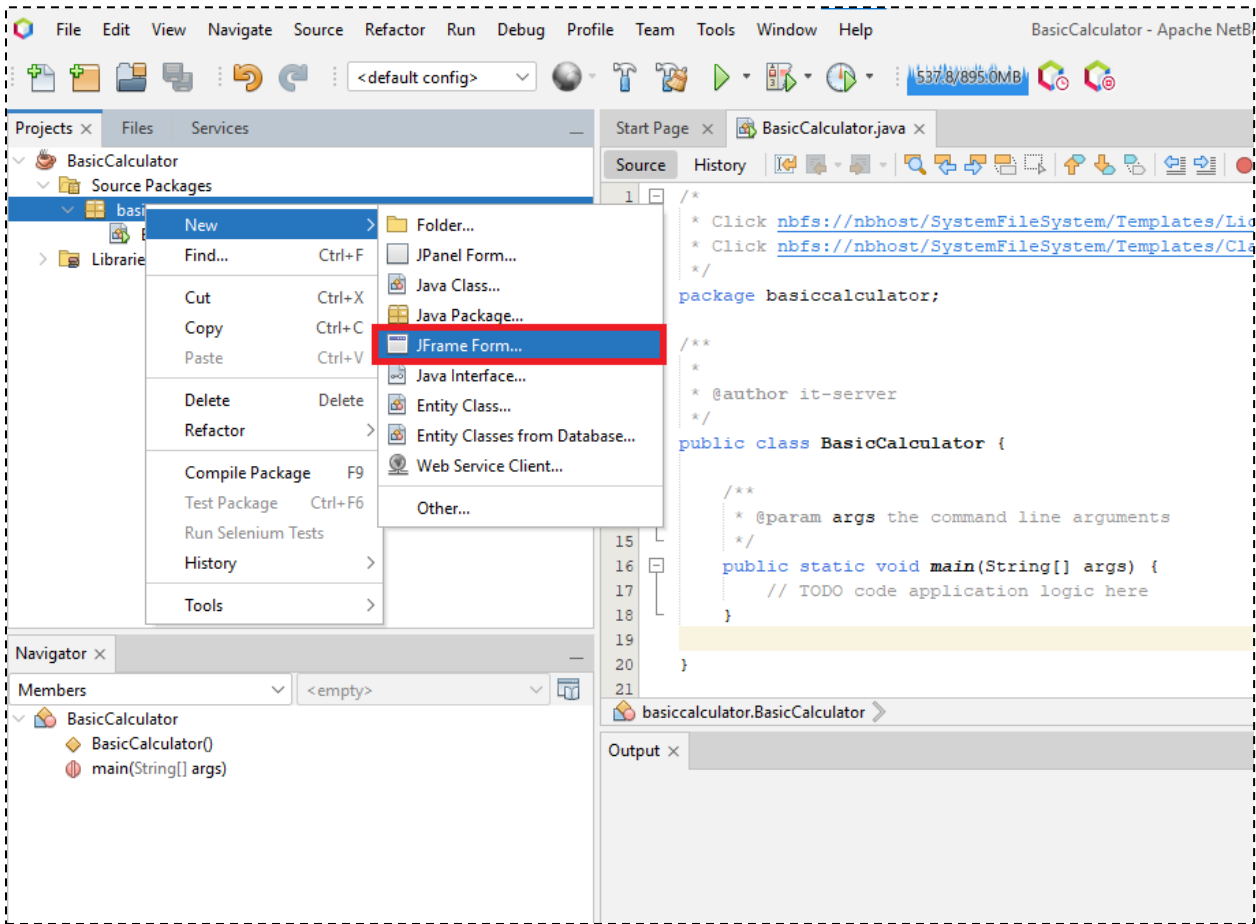
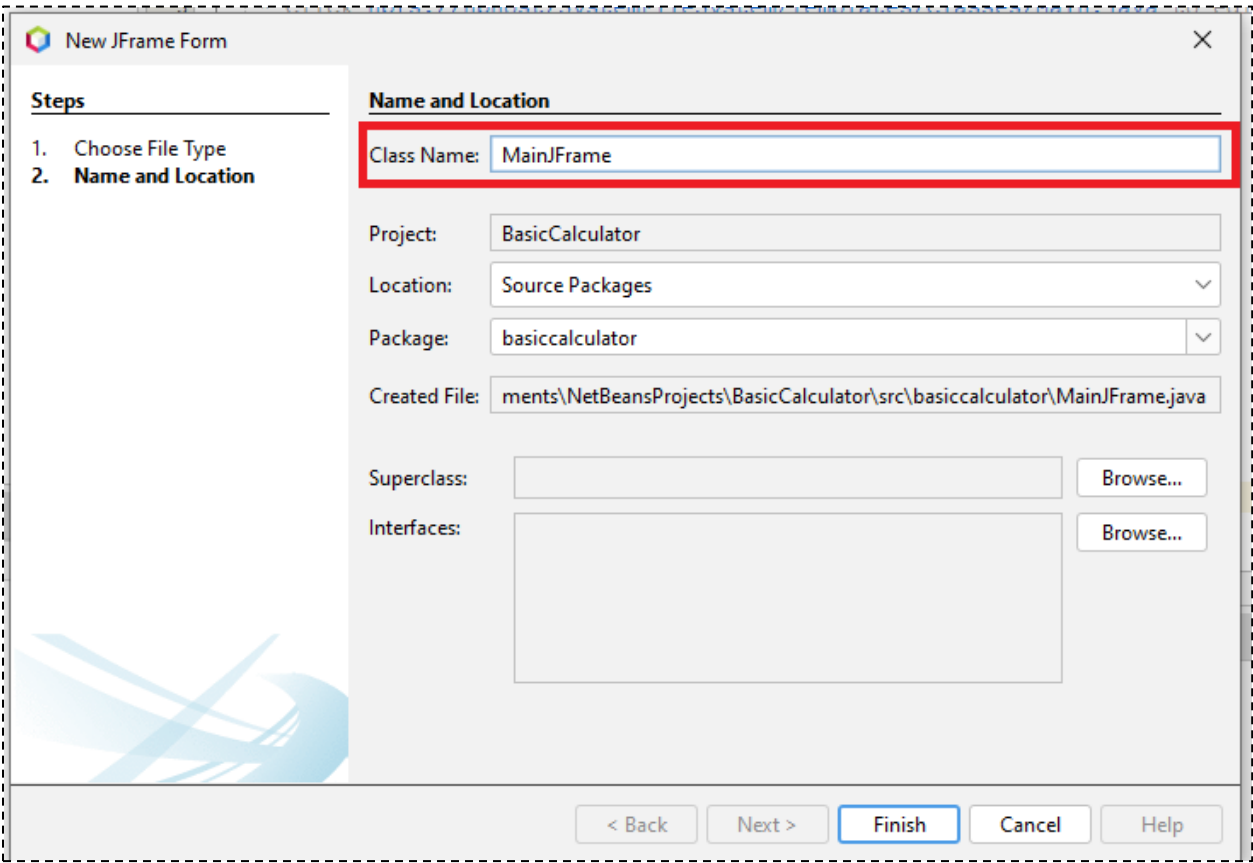


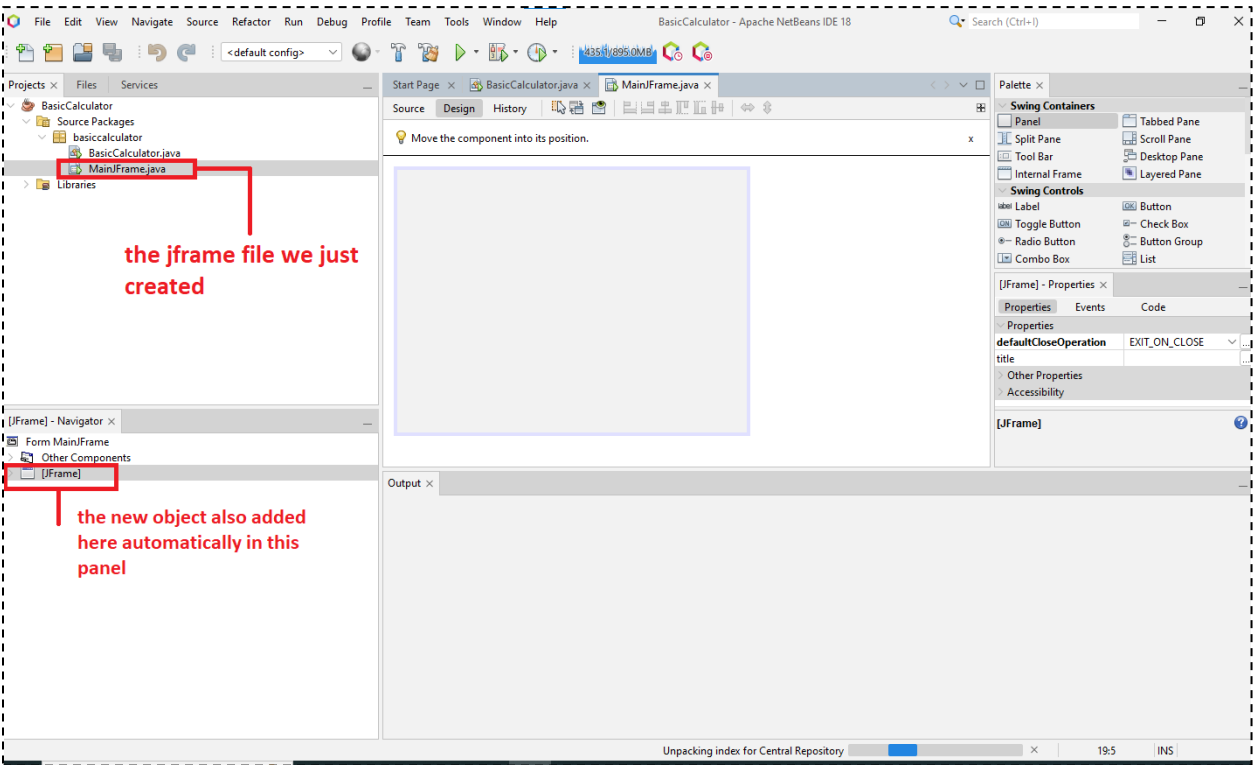
Figure 1: NetBeans IDE Interface

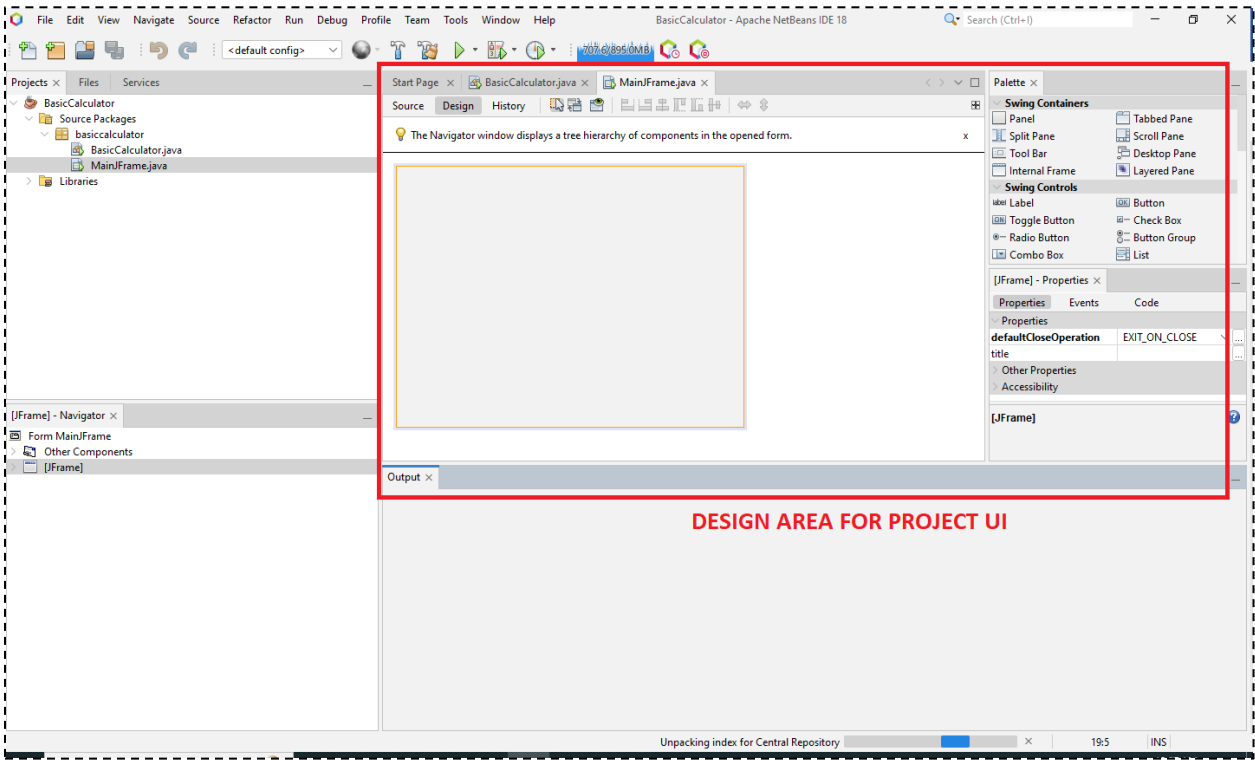


1. Create a new JFrame form file by right clicking on package inside Source Packages in your project.

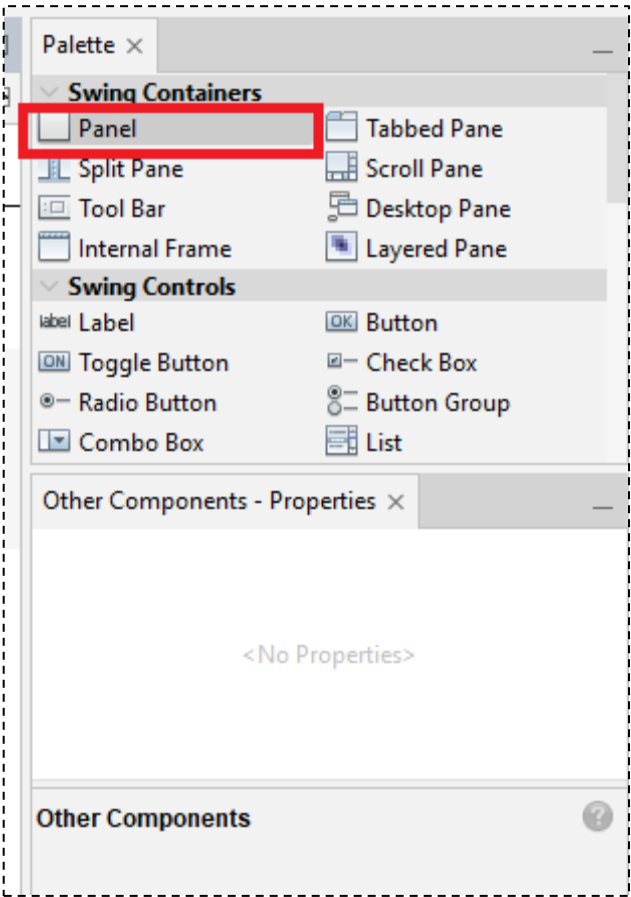


2. After clicking new *JFrame form* , a dialogue box will appear. Change the Class Name field into any name you want. However, DESCRIPTIVE name is the ideal and avoid putting spaces between words. For the sake of this project, just follow the name I used, *MainJFrame*. Please copy exact CAPITALIZATION because Java is a case sensitive language. Click '**Finish**' once done.

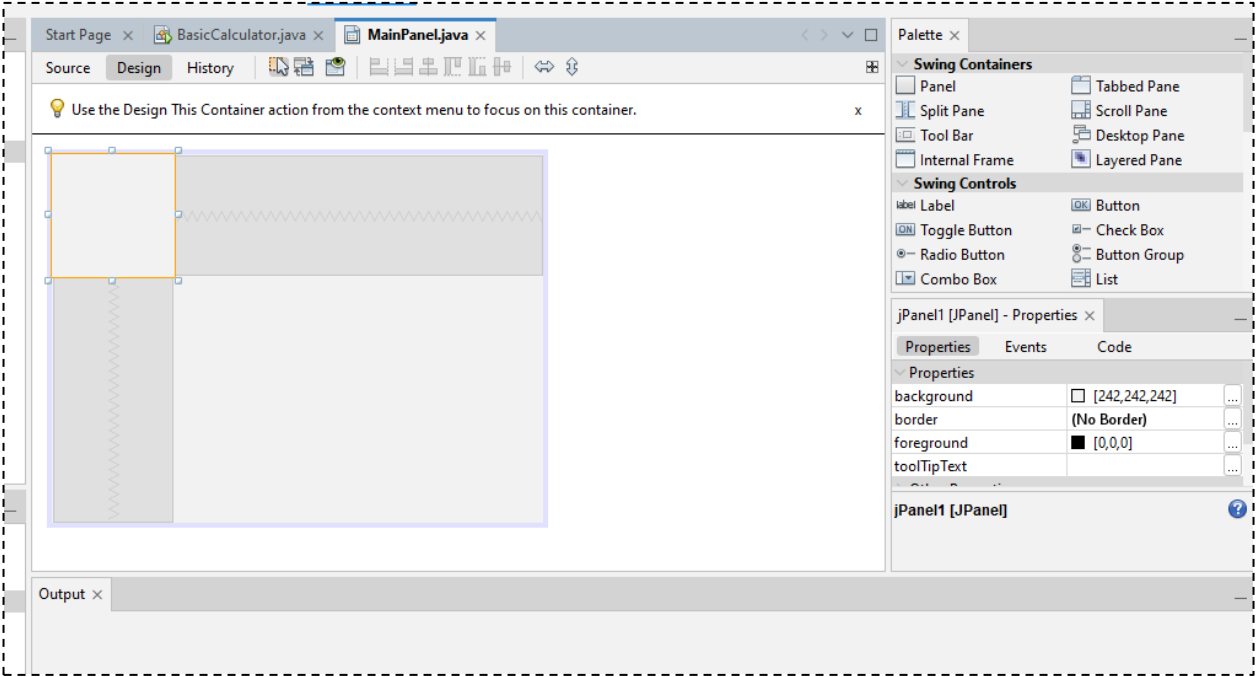




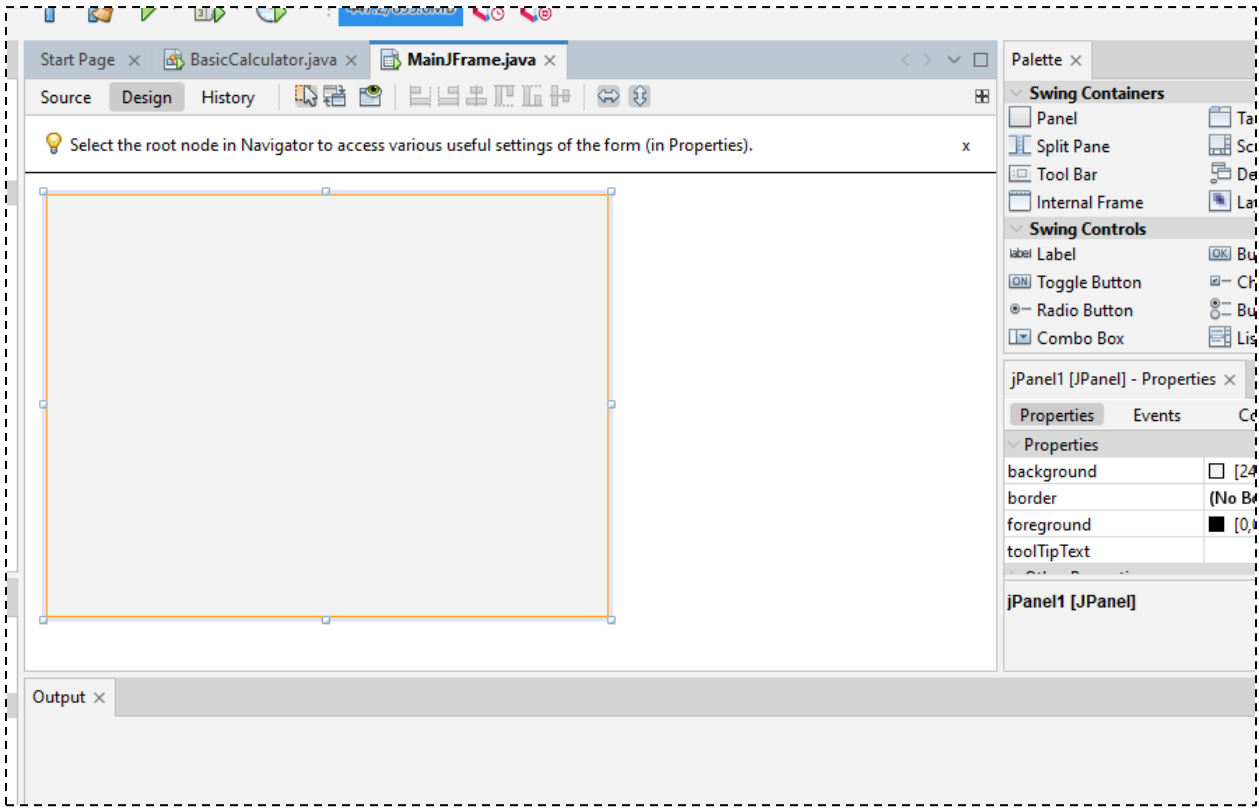
3. Highlighted in red box above is the area where we design and create the UI (User Interface) of our project. Netbeans offers easy drag & drop features for the objects we are going to use for our projects and the objects' codes will be added to our source code automatically.



4. On Palette panel, under Swing Containers, click and drag the Panel object towards the design canvas and cover it entirely.



5. Drag and adjust the Panel object until it covers the entire JFrame.

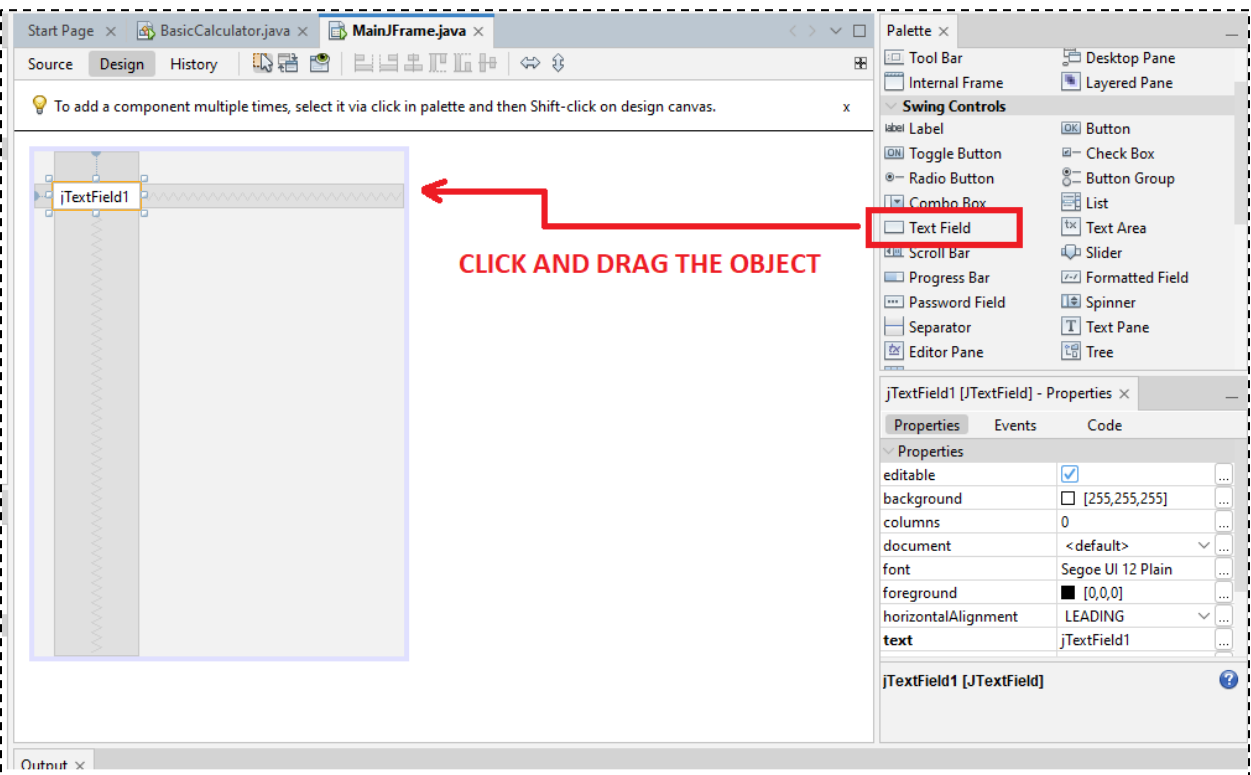
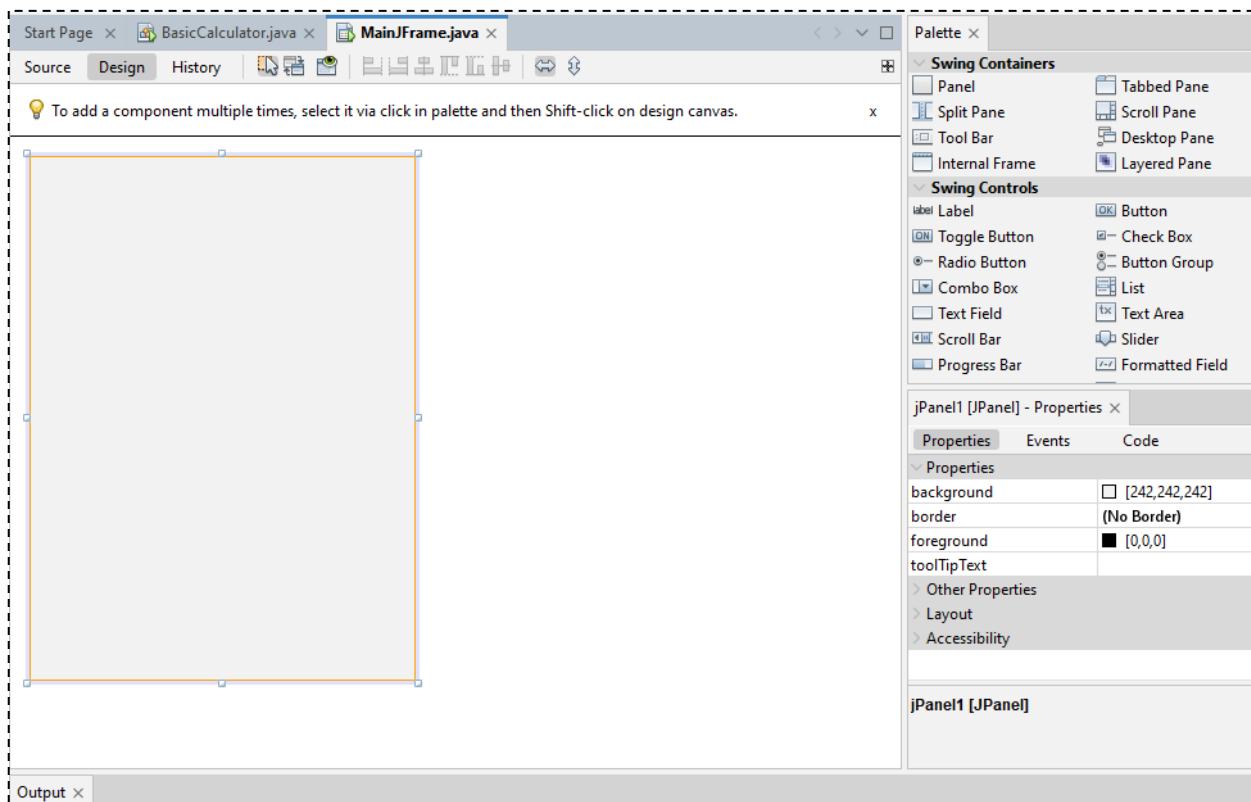


6. You can also adjust the JFrame in accordance to your desired height and width of your calculator interface.

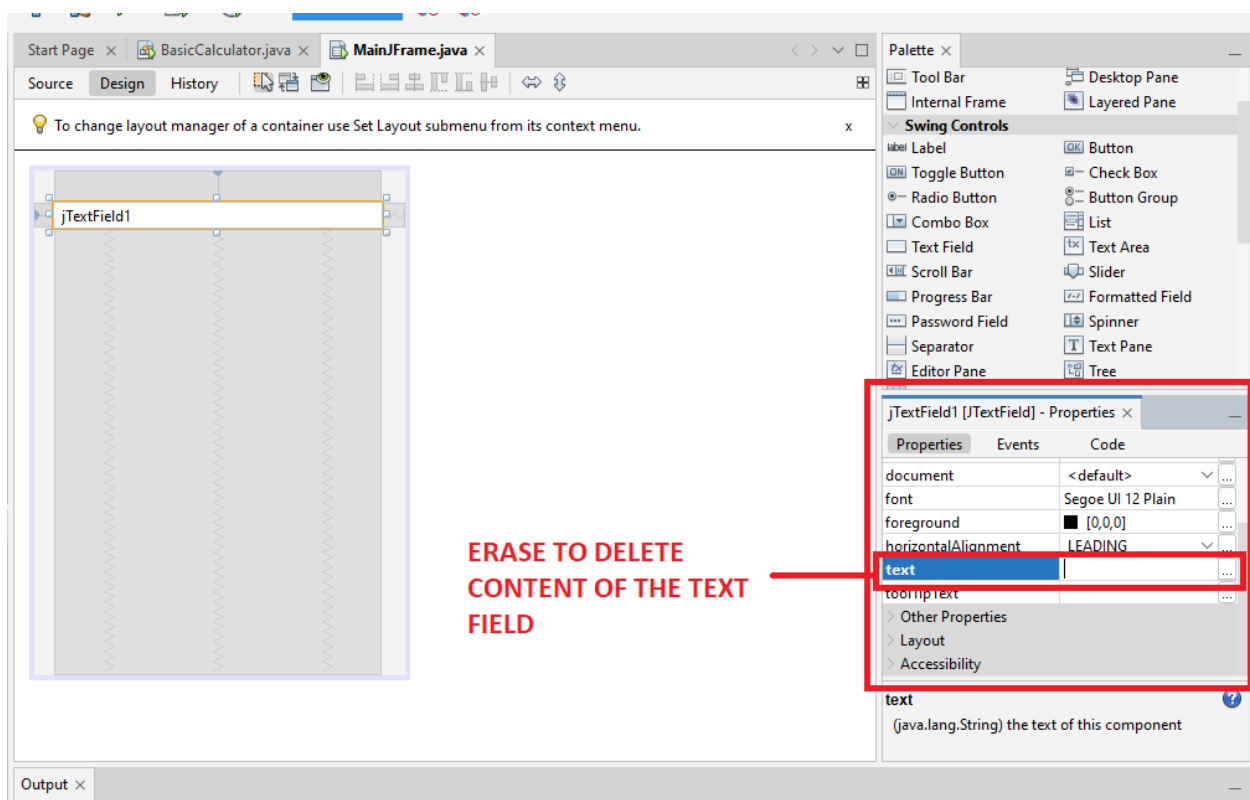
Note: JFrame contains the entire body of the project interface while JPanel contains the objects to operate / manipulate in the project. When you adjust the JFrame's width and height, the JPanel's width and height will also follow. However, if you adjust the JPanel's width and height



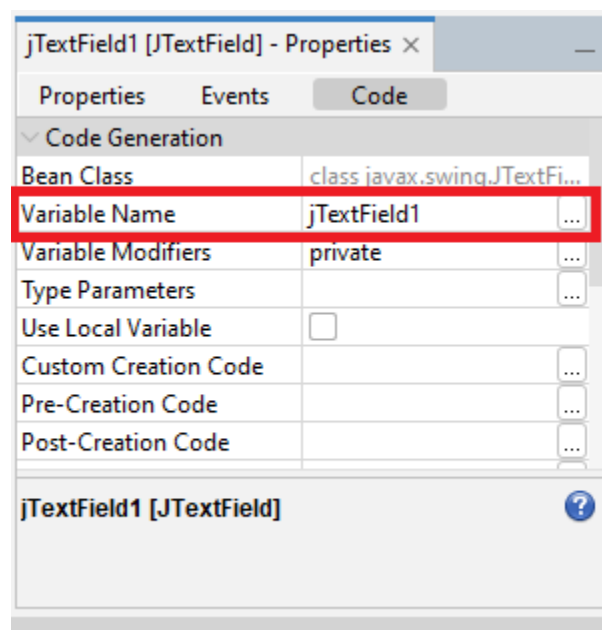
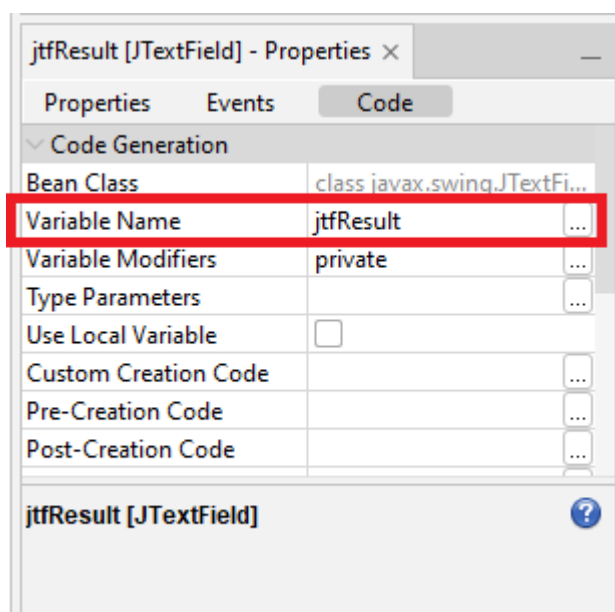
instead, the JFrame will not adjust and will just stick to its fixed size. JFrame is the container of the JPanel and JPanel is the container of the objects such as textfields, buttons and etc.

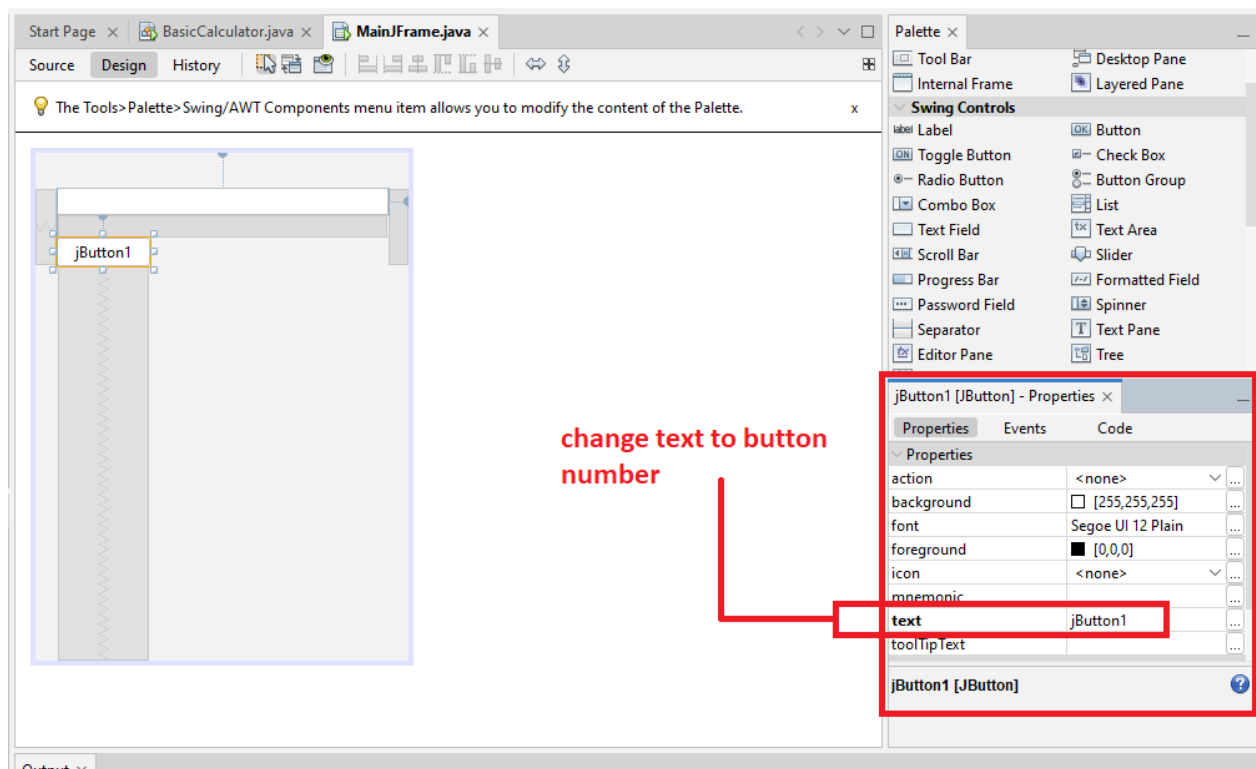


7. Click and drag the textfield inside the jpanel and adjust its width and height appropriately. We will use this textfield as the area where our input and output result.



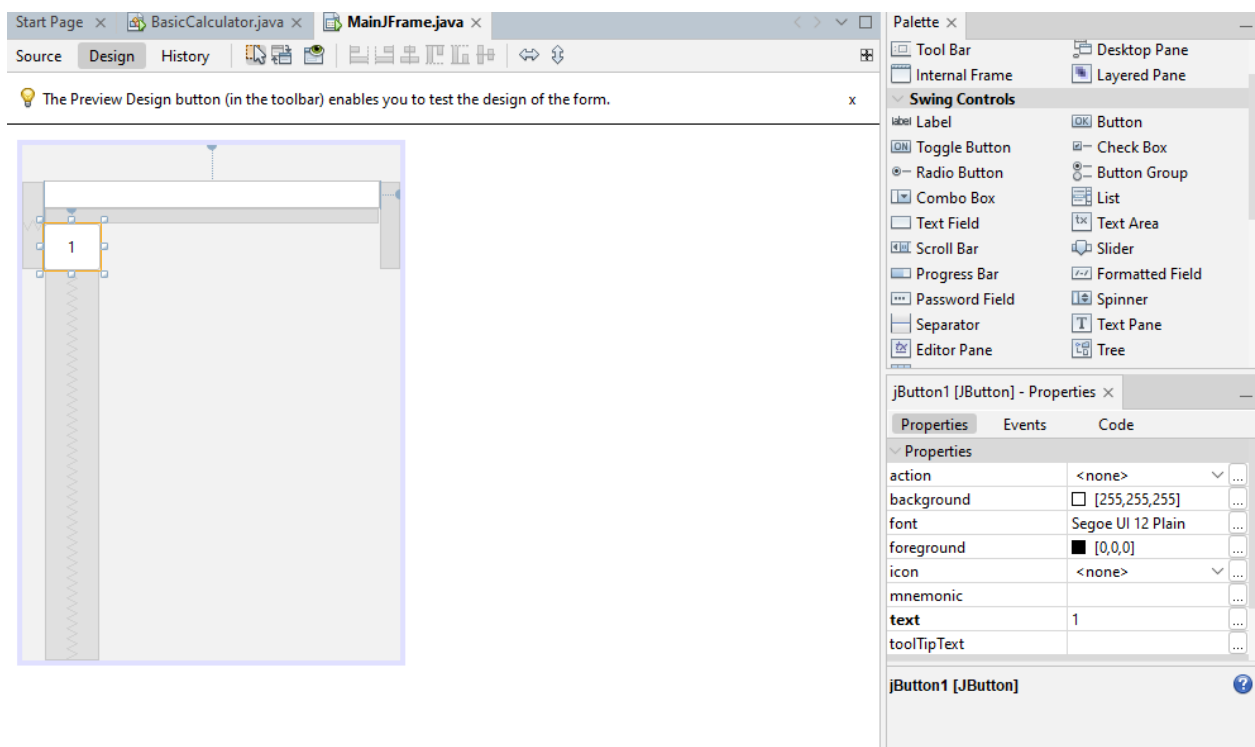
- Click on the jTextField object inside the JPanel to view its properties. Drag down in it and look for the 'text' row and erase whatever inside in the cell next to it. The purpose is to make the text field display no text data by default.
- On the same object, in the properties panel of the Jtextfield, click on the 'Code' tab and look for the Variable Name section. Change it into something descriptive that shows result of the operation. In this project, just rename it to '**jtfResult**' as seen below.



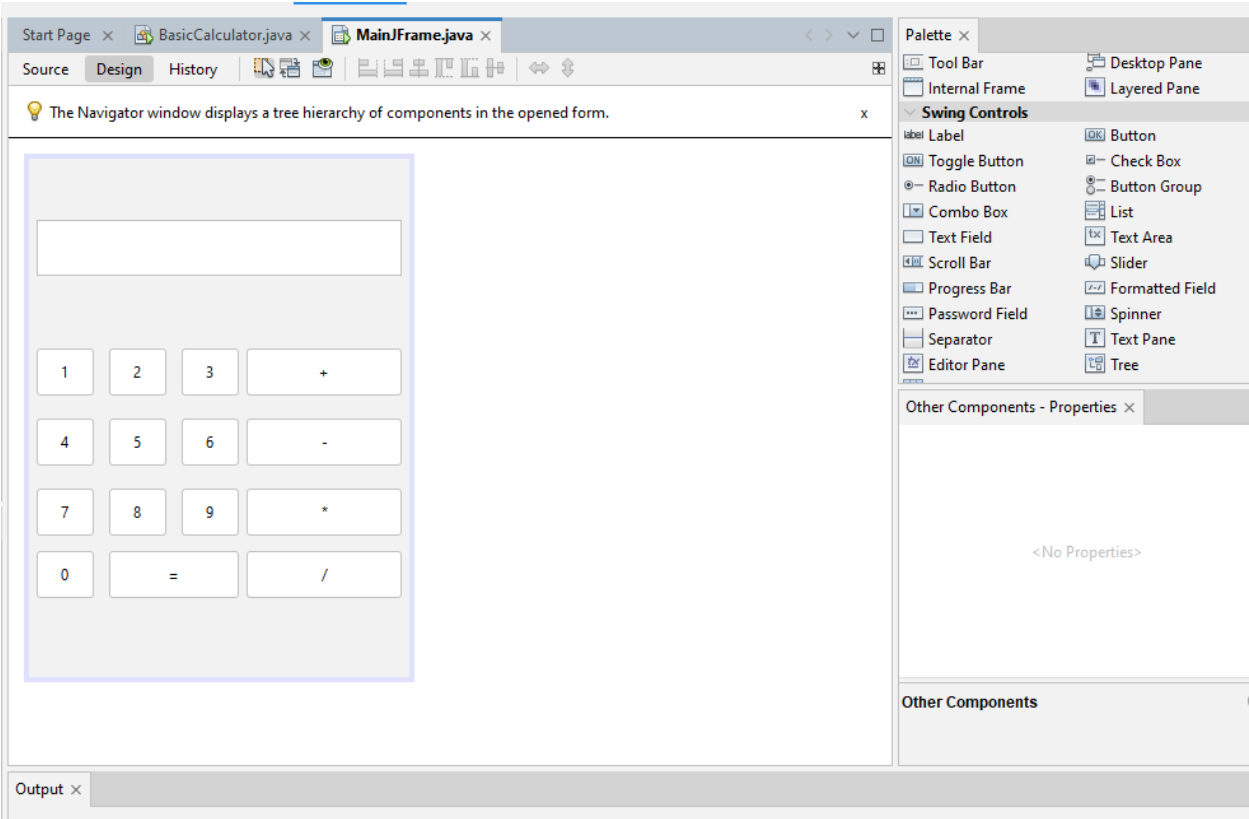


10. Add another object which is a JButton.

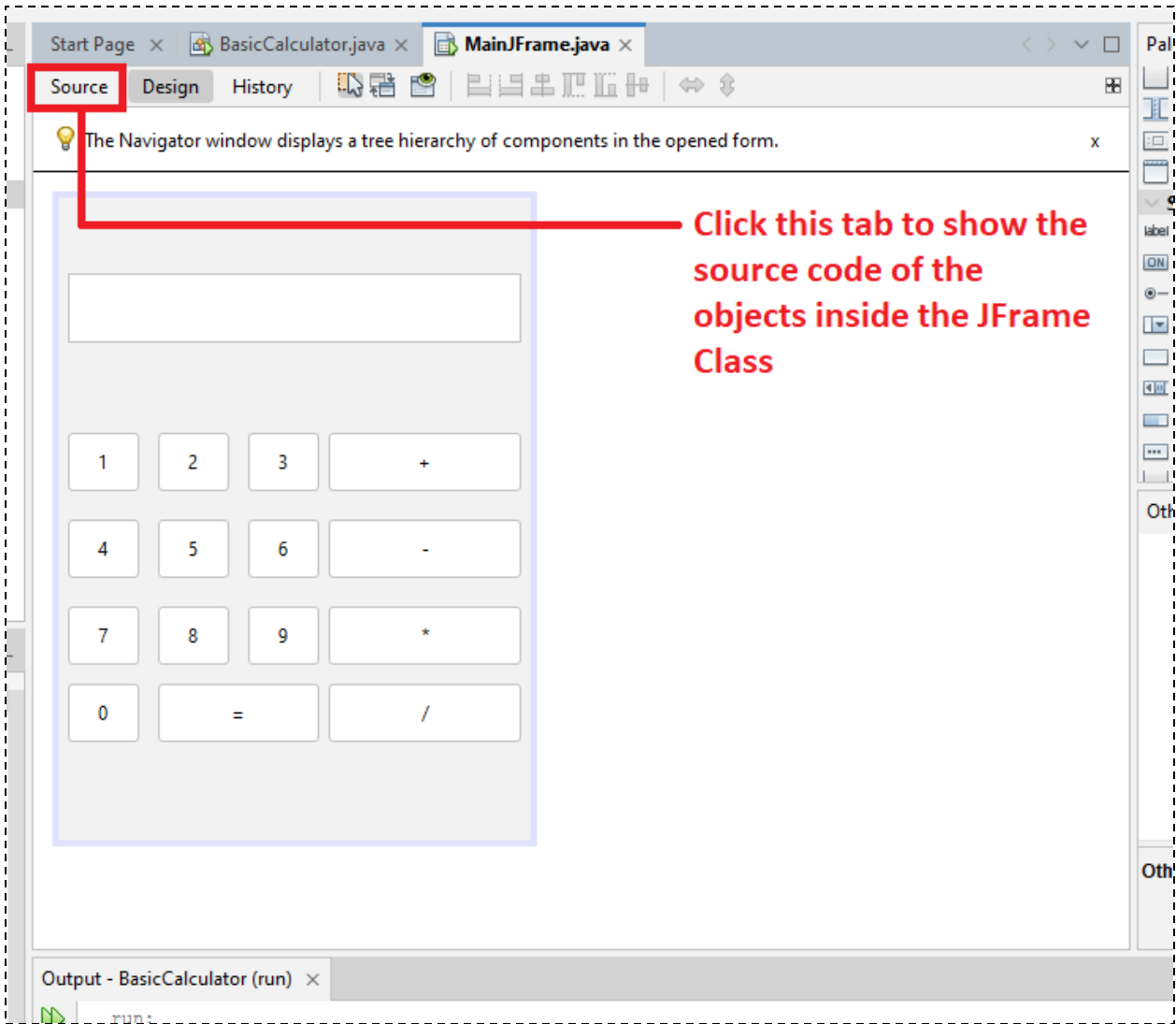
11. Change the text of the jButton into numbers for inputs for your calculator. For button 1, just put text '1'. Just do the same with number buttons.



12. Do the same with other number buttons as well. Add the operator buttons too as seen below.



13. Feel free to have your own layout as long as it looks like a calculator.



14. After the layout, click on the Source Code tab to see the code of the JFrame. Next you need to do is to set the 'visibility' attribute of the whole UI so it will appear as the project runs. Just add the method `setVisible(true)` inside the `MainJFrame` constructor. As seen below

```
1  /*
2  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this lice
3  * Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JFrame.java to edit this template
4  */
5  package basiccalculator;
6
7  /**
8   *
9   * @author it-server
10  */
11  public class MainJFrame extends javax.swing.JFrame {
12
13      /**
14       * Creates new form MainJFrame
15       */
16      public MainJFrame() {
17          initComponents();
18      }
19
20      /**
21       * This method is called from within the constructor to initialize the form.
22       * WARNING: Do NOT modify this code. The content of this method is always
23       * regenerated by the Form Editor.
24       */
25      @SuppressWarnings("unchecked")
26      Generated Code
27
28  }
```

basiccalculator.MainJFrame > MainJFrame >

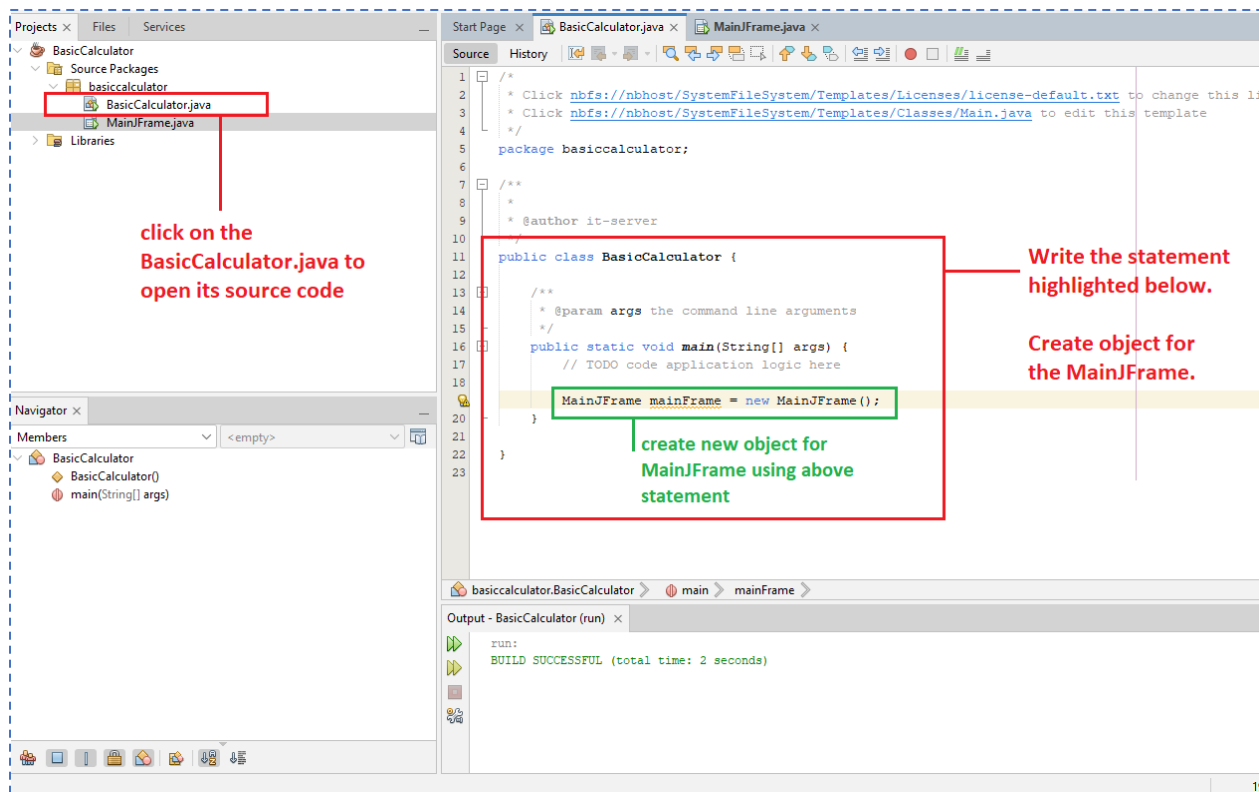
Output - BasicCalculator (run) x

run:  
BUILD SUCCESSFUL (total time: 2 seconds)

```
1  /*
2  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this lice
3  * Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JFrame.java to edit this template
4  */
5  package basiccalculator;
6
7  /**
8   *
9   * @author it-server
10  */
11  public class MainJFrame extends javax.swing.JFrame {
12
13      /**
14       * Creates new form MainJFrame
15       */
16      public MainJFrame() {
17          initComponents();
18          setVisible(true);
19      }
20
21      /**
22       * This method is called from within the constructor to initialize the form.
23       * WARNING: Do NOT modify this code. The content of this method is always
24       * regenerated by the Form Editor.
25       */
26      @SuppressWarnings("unchecked")
27      Generated Code
28
29  }
```

basiccalculator.MainJFrame > MainJFrame >

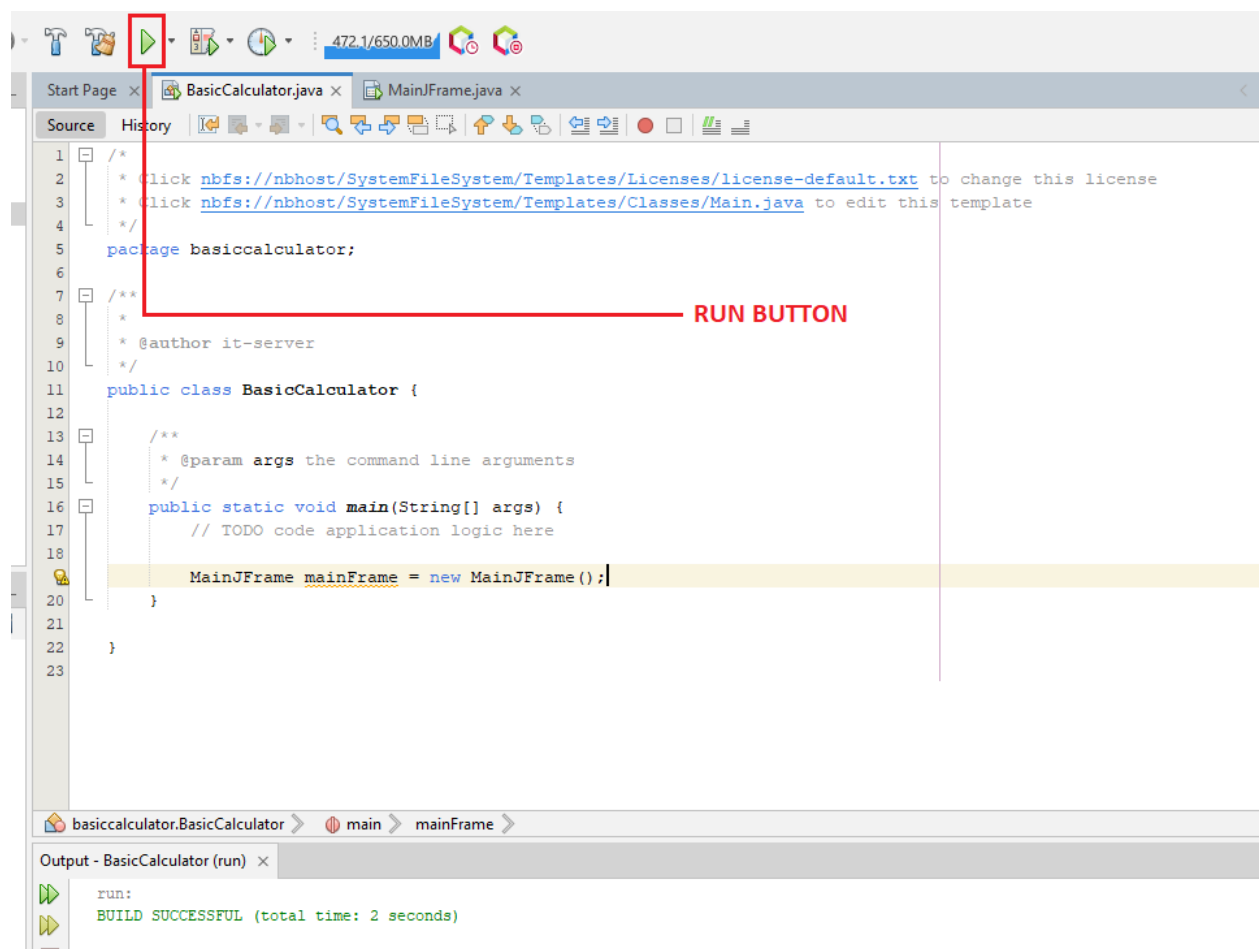
Output - BasicCalculator (run) x

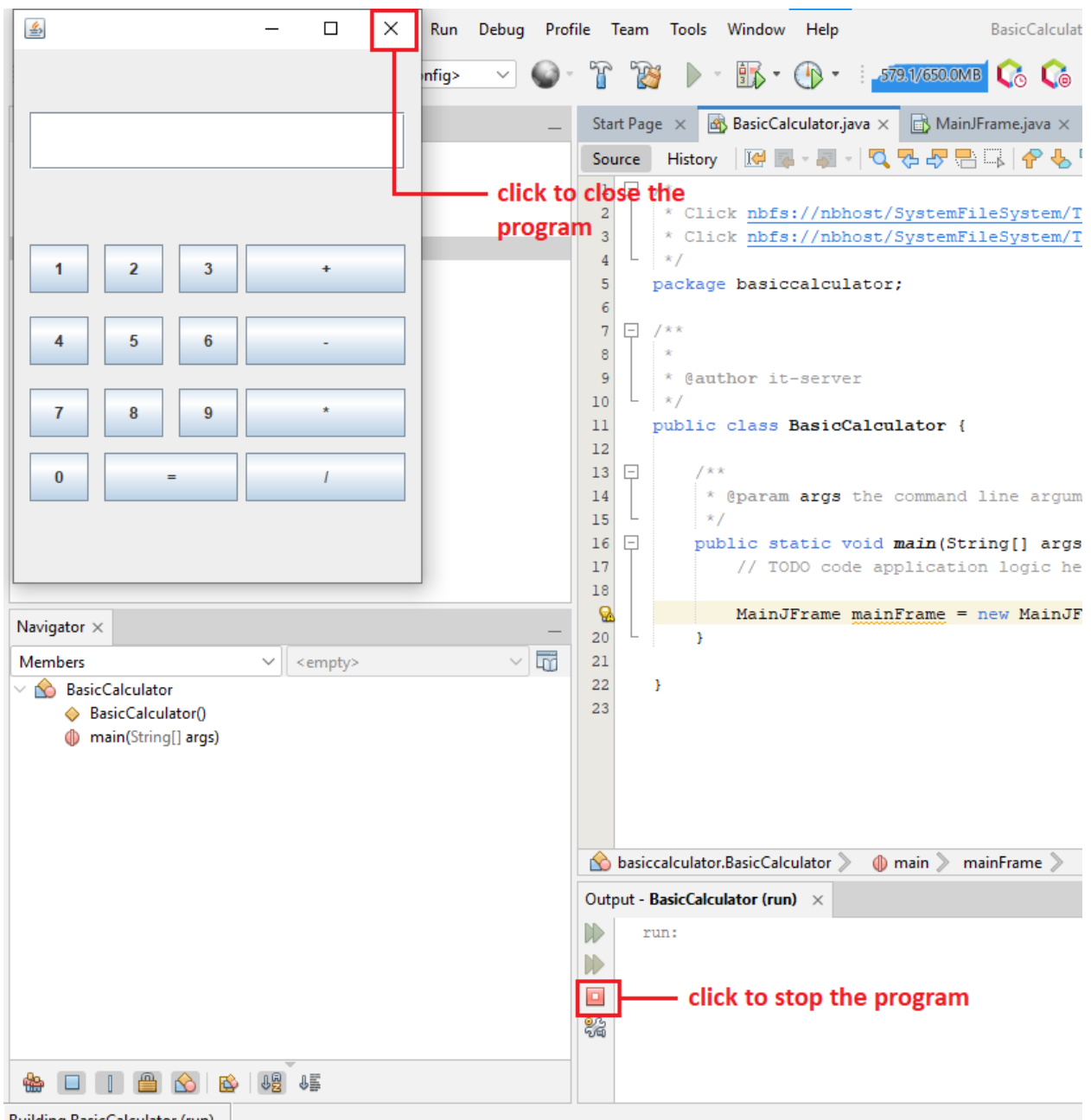


15. Double click on the BasicCalculator.java to open its source code and add the statements as seen above. Just add this statement tho:

```
MainJFrame mainframe = new MainJFrame();
```

16. After writing the statement, hit **ctrl + s** on the keyboard to save the changes in the source code and then click the run button above in the IDE.





17. If you notice, the calculator interface appears at the upper left corner of the screen. Let's set the User interface of the calculator to appear in the middle once it is launched by adding the statement below:

```
setLocationRelativeTo(null);
```

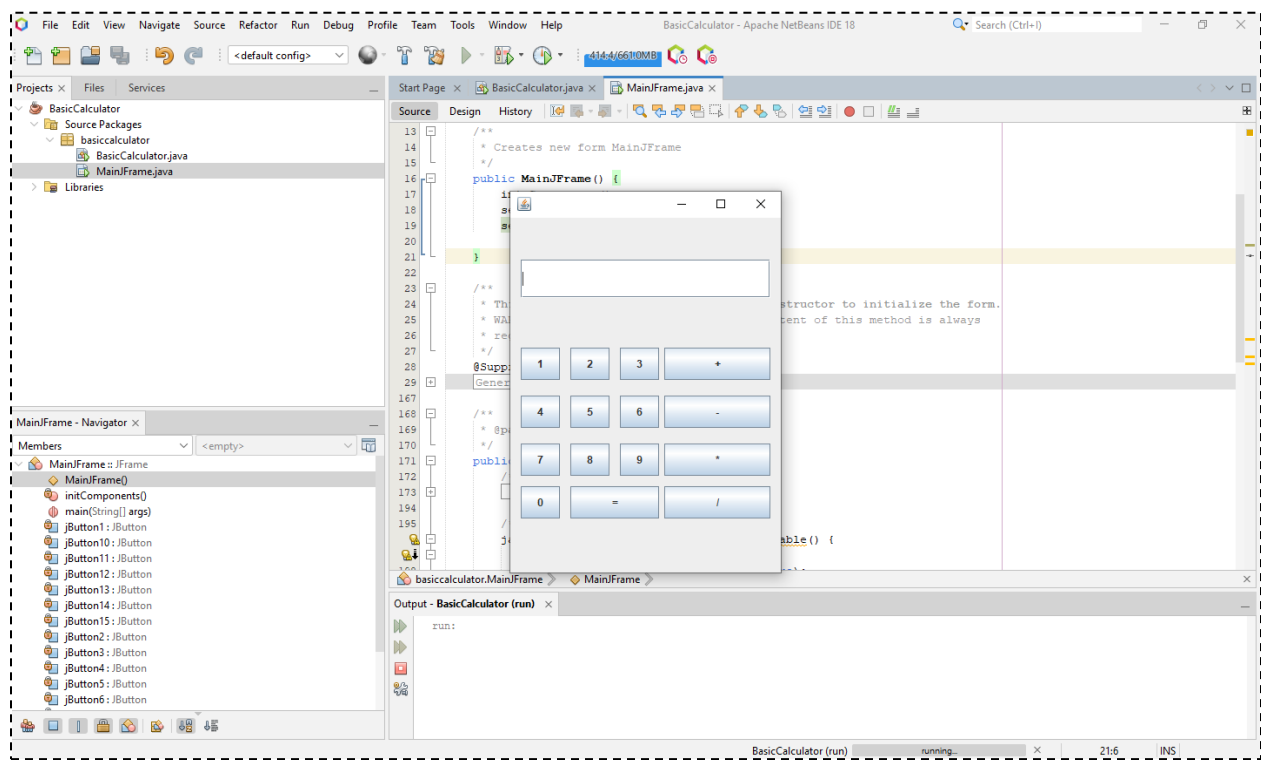
```

*/
public class MainJFrame extends javax.swing.JFrame {

    /**
     * Creates new form MainJFrame
     */
    public MainJFrame() {
        initComponents();
        setVisible(b: true);
        setLocationRelativeTo(c: null);
    }

```

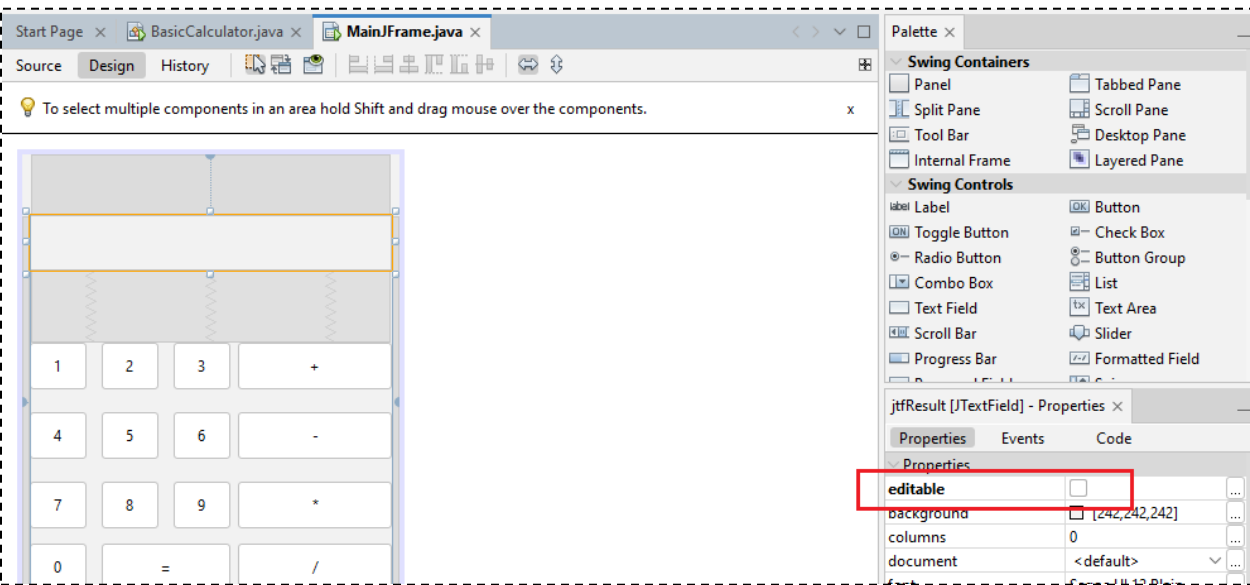
18. After adding the setLocationRelativeTo(null); statement, click on the run button again and notice the changes. The Calculator UI now starts to display in the middle of the screen.



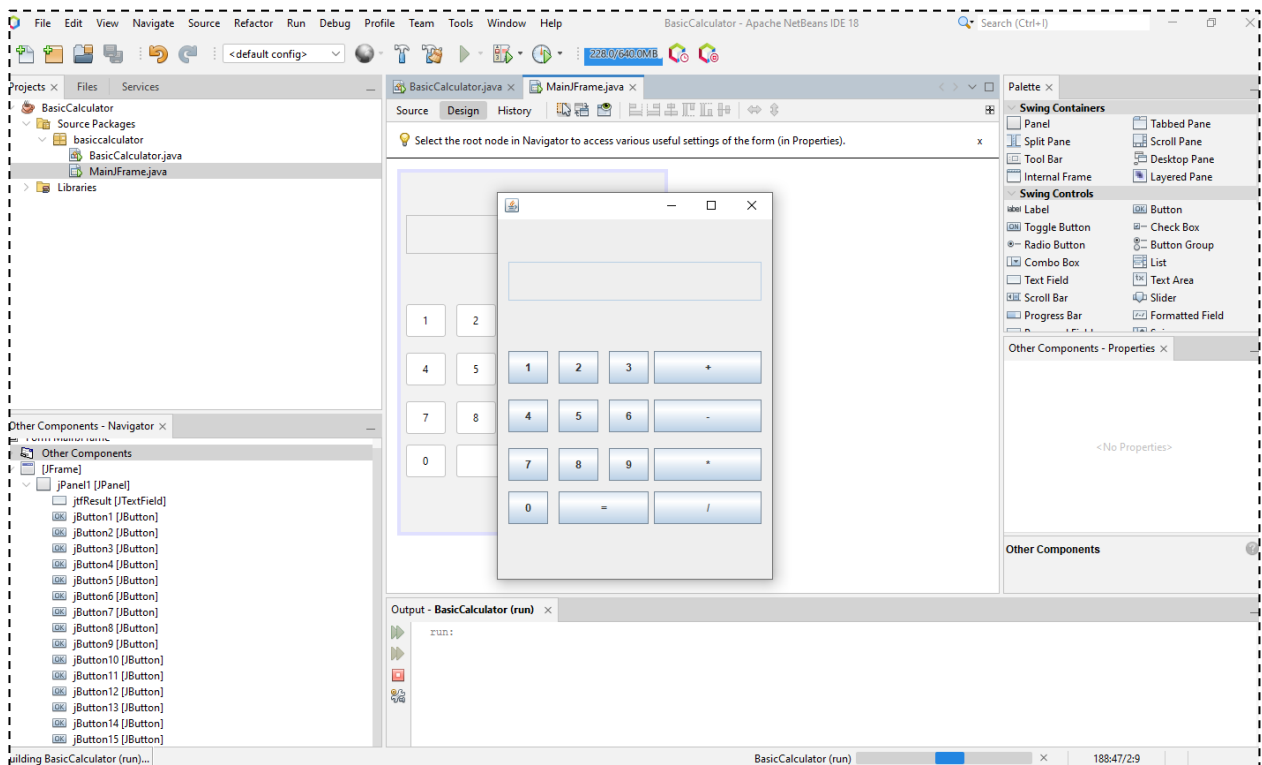
19. Stop the project. If you notice, the text field has an active blinking for typing. We need to disable it so that the input data will be coming from the buttons only once the buttons are pressed / clicked, just like what a calculator does. To do it, click on the MainJFrame.java tab, and click on the 'Design' sub-tab as seen below:



20. Uncheck the 'editable' property of text field so that by default it isn't writable.







- 21. Try to click the 'Run' button again to check the actual output. You will notice the text field is now not editable and this is what we want.
- 22. The next thing we need to do is to add functionalities to the buttons once they are pressed, they are to display the texts as numbers and somehow, we can operate the numbers to display results. Before anything else, let's prepare the variables we are to use for the operations.

Here are the sample variables we can use for the calculator operations:

```
int num1, num2;  
char operator;  
String result;
```

You can declare the following variables inside the MainJFrame.java class as global variables as seen below:

```
BasicCalculator.java x MainJFrame.java x
Source Design History
7 /**
8  *
9  * @author it-server
10 */
11 public class MainJFrame extends javax.swing.JFrame {
12
13     /**
14      * Creates new form MainJFrame
15      */
16     public MainJFrame() {
17         initComponents();
18         setVisible(b: true);
19         setLocationRelativeTo(c: null);
20
21     }
22
23     int num1, num2;
24     char operator;
25     String result;
26
27
28     /**
29      * This method is called from within the constructor to initializ
30      * WARNING: Do NOT modify this code. The content of this method i
31      * regenerated by the Form Editor.
32      */
33     @SuppressWarnings("unchecked")
```

After declaring the variables, we are to put code statements inside each button. If you added all the necessary buttons, you will see something like this below on your Navigator:

Other Components - Navigator x

Other Components

JFrame

jPanel1 [JPanel]

jtfResult [JTextField]

jButton1 [JButton]

jButton2 [JButton]

jButton3 [JButton]

jButton4 [JButton]

jButton5 [JButton]

jButton6 [JButton]

jButton7 [JButton]

jButton8 [JButton]

jButton9 [JButton]

jButton10 [JButton]

jButton11 [JButton]

jButton12 [JButton]

jButton13 [JButton]

jButton14 [JButton]

jButton15 [JButton]

take note of the variable names of your objects.

example: jtfResult is the name of your JTextField object, while jButton1 is the name of the one of your JButton objects.

123+456-789\*0=/

Output x

Always take note and remember the variable names (identifiers) of your objects. If in-case you prefer to change the identifier of your object, just click on the object, go to object's Properties panel, click on the Code tab and change the Variable Name field as seen below:

The Preview Design button (in the toolbar) enables you to test the design of the form.

123+

456-

789\*

0= /

Panel

Split Pane

Tool Bar

Internal Frame

Swing Controls

Label

ToggleButton

Radio Button

Combo Box

Text Field

Scroll Bar

Progress Bar

Tabbed Pane

Scroll Pane

Desktop Pane

Layered Pane

Button

Check Box

Button Group

List

Text Area

Slider

Formatted Field

jButton1 [JButton] - Properties

Properties

Events

Code

Code Generation

Bean Class

Variable Name

Variable Modifiers

Type Parameters

Use Local Variable

Generate Mnemonics Code

Custom Creation Code

Pre-Creation Code

class javax.swing.JButton

jButton1

private

☐

☐

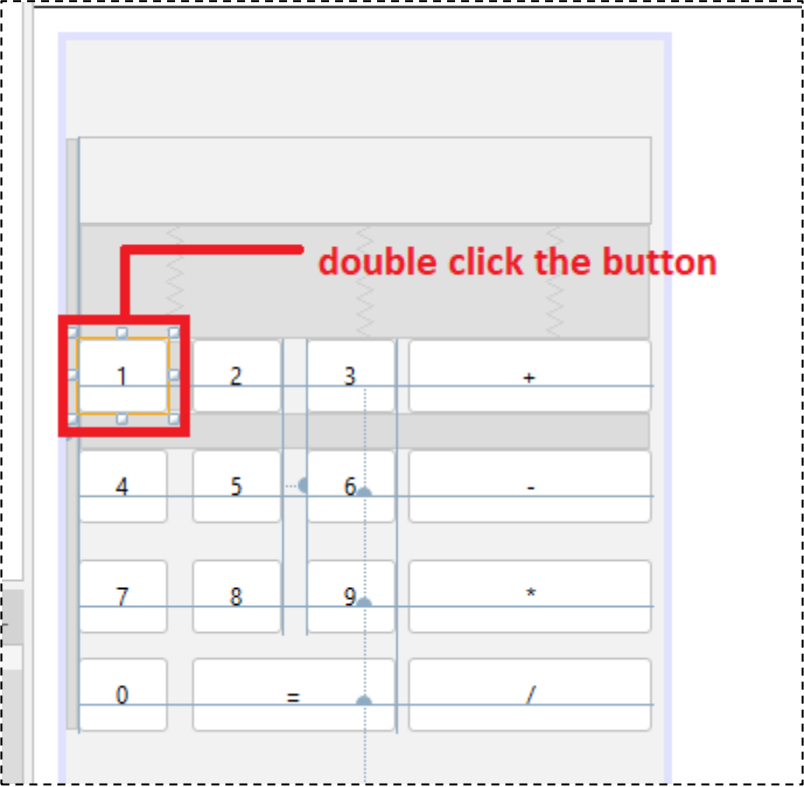
jButton1 [JButton]

the identifier of the variables are used to call and operate the variables.

Output

# CODING THE OBJECTS & EVENTS

We now need to add codes to our button object so they will do something once clicked. As for the button with text '1', it must display the data '1' in the text field once it is clicked. However, if the button is clicked multiple times, it must display the data '1' multiple times in a concatenated string form. For example, the button is clicked thrice (3 times), the text field should display '111' as data in the text field. To do this, we need to add the following statements inside the JButton1's code. To jump write away to the method definition of the JButton1, just head over to Design tab of your MainJFrame and DOUBLE CLICK on the button with text '1' written in it. You can also do the same with other buttons whenever you want to add codes to them.

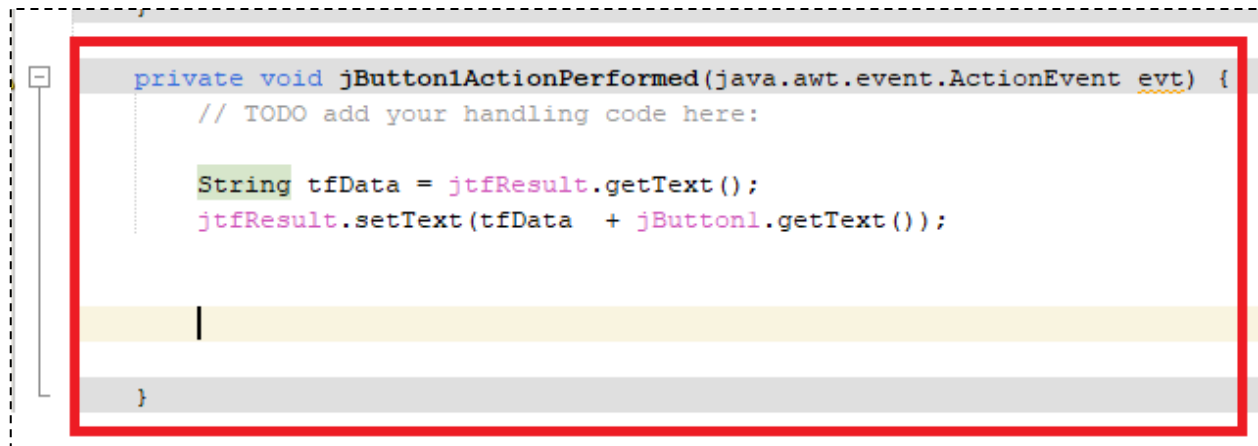


```
BasicCalculator.java x MainJFrame.java x
Source Design History
28 * WARNING: Do NOT modify this code. The content of this method is always
29 * regenerated by the Form Editor.
30 */
31 @SuppressWarnings("unchecked")
32 Generated Code
182
183 private void jTextFieldActionPerformed(java.awt.event.ActionEvent evt) {
184     // TODO add your handling code here:
185
186 }
187
188
189 private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
190     // TODO add your handling code here:
191
192
193
194
195
196 }
197
198
199 /**
200  * @param args the command line arguments
201  */
202 public static void main(String args[]) {
203     /* Set the Nimbus look and feel */
204     LookAndFeelSettingCode (optional)
```

add the following statements inside the body of JButton1

```
String tfData = jTextField1.getText();  
jTextField1.setText(tfData + jButton1.getText());
```

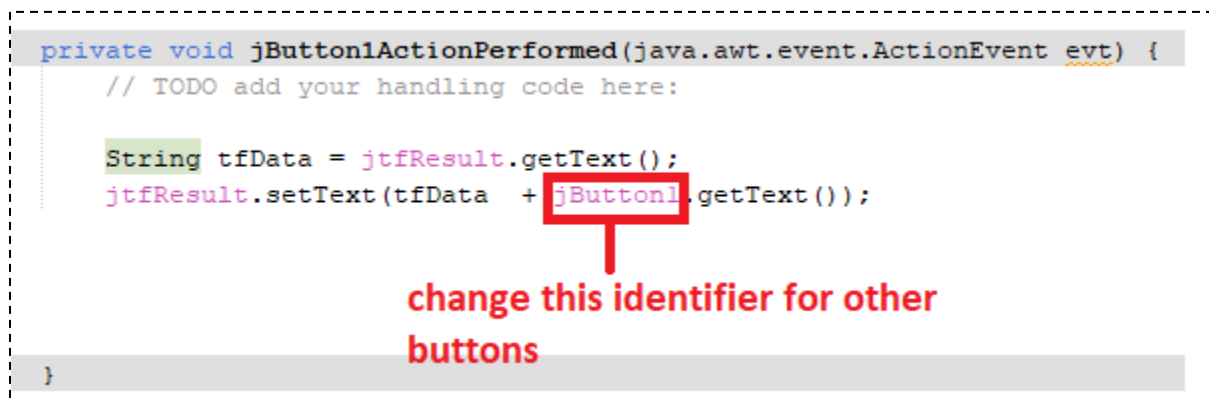
As seen below:



```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
  
    String tfData = jTextField1.getText();  
    jTextField1.setText(tfData + jButton1.getText());  
  
}
```

The above code will get the text data from the text field and concatenate the text data contained by the JButton once the button is pressed. After adding the code, you can hit **ctrl+s** in case the netbeans isn't auto-save to manually save the code. Press the run button and try pressing the button with the text '1' multiple times to see the results.

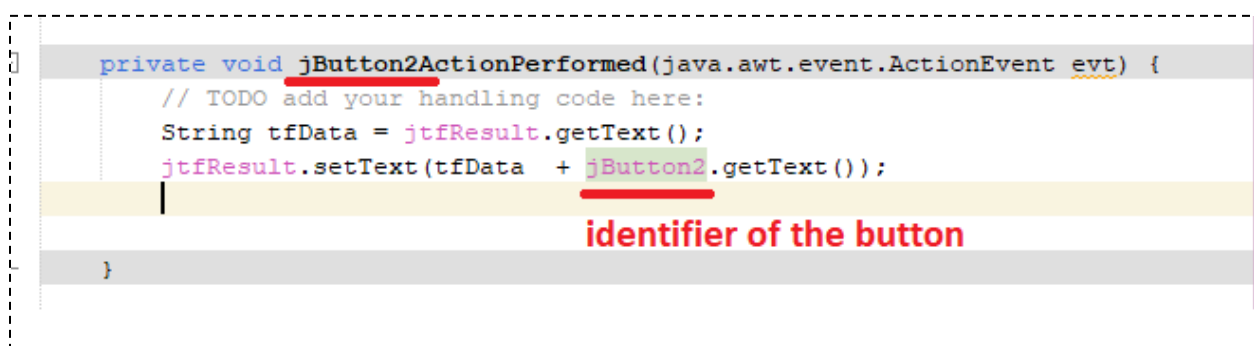
You can do the same with the other buttons however you need to change the object name in the following statement in accordance to the variable name of your JButton.



```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
  
    String tfData = jTextField1.getText();  
    jTextField1.setText(tfData + jButton1.getText());  
  
}
```

change this identifier for other buttons

For example, you want to add the code for button 2, just double on the button with text '2' just like what you did earlier with the previous button to automatically write the method definition of the button and jump right into it. Just copy the source code you pasted but this time change the identifier part of the code with the identifier of the button you're working with now. Just look at the image below:



```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    String tfData = jTextField1.getText();  
    jTextField1.setText(tfData + jButton2.getText());  
  
}
```

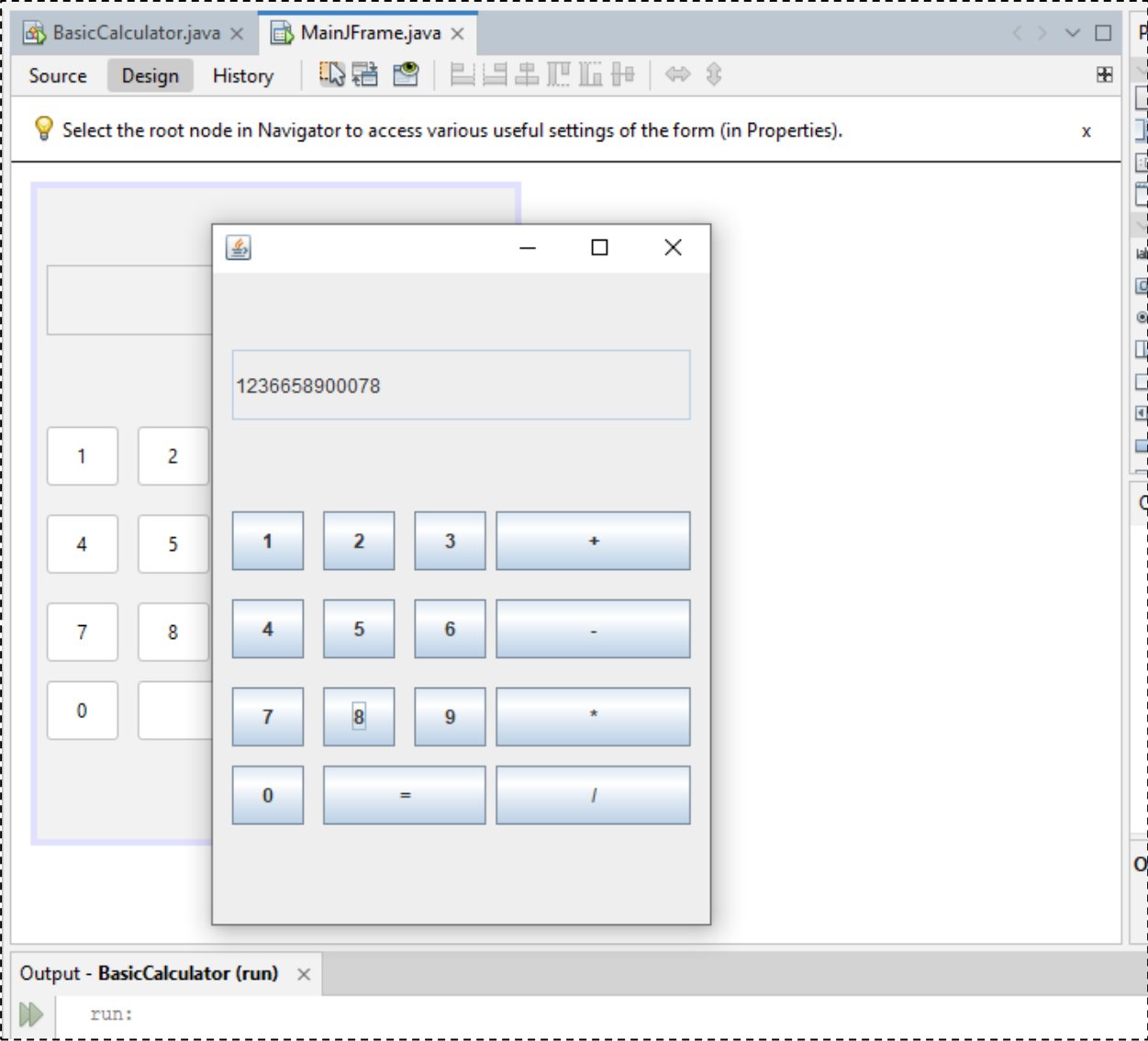
identifier of the button

Just do the same with the other buttons with numbers. Basically, from button 1, 2 . . . 9 & 0.

```
private void jButton10ActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    String tfData = jTextField1.getText();  
    jTextField1.setText(tfData + jButton10.getText());  
}
```

Take note: My source code for the button zero. My button with text '0' in it has a variable name of JButton10. That's the name I need to use together with the .getText() function in order for it to work perfectly fine.

After adding the codes for all the numerical buttons, try running the program and test it. It should work perfectly fine!



The next thing we need to do is to add the codes for the operator and for the equal sign. For the operators (addition, subtraction, multiplication & division), we need to allow it to store whatever data in the text field to the variable 'num1' we wrote in the variable declaration of source code. This variable

'num1' will hold the first operand and the other variable with identifier 'num2' will hold the second operand. The result of the operation for these two variables will be stored in the 'result' variable.

# Operand

Updated: 04/26/2017 by Computer Hope

In computer programming, an **operand** is any object capable of being manipulated. For example, in "1 + 2" the "1" and "2" are the **operands** and the plus symbol is the **operator**.

Before anything else, we need to fix / change the identifier of our operator buttons for ease of reference. I suggest you follow the identifiers I use for the following buttons:

- Plus button -> jButtonPlus
- Minus button -> JButtonMinus
- Multiply button -> JButtonMultiply
- Divide button -> JButtonDivide
- Equals button -> JButtonEquals

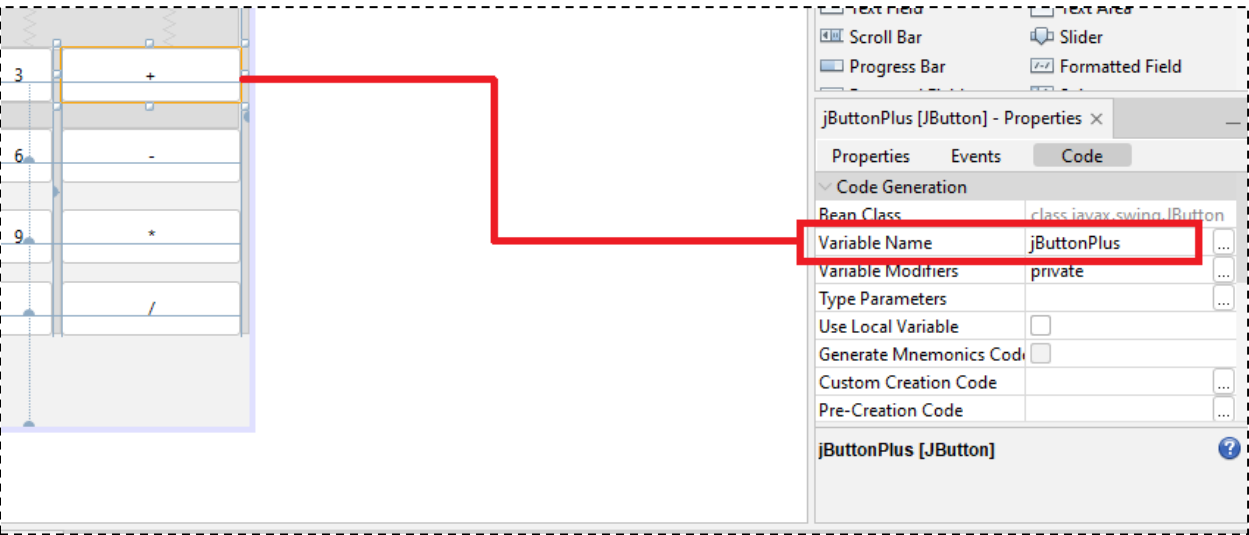


Figure 2. Change the identifier of your Operator Buttons

Write the following source code in your **jButtonPlus** as seen below:

```
private void jButtonPlusActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
  
    num1 = Integer.parseInt(jtfResult.getText());  
    jtfResult.setText(null);  
    operator = '+';  
  
}
```

Here are the source code for the buttons plus, divide, minus and multiple.

#### **Addition Button**

```
num1 = Integer.parseInt(jtfResult.getText());  
jtfResult.setText(null);  
operator = '+';
```

#### **Minus Button**

```
num1 = Integer.parseInt(jtfResult.getText());  
jtfResult.setText(null);  
operator = '-';
```

#### **Multiplication Button**

```
num1 = Integer.parseInt(jtfResult.getText());  
jtfResult.setText(null);  
operator = 'x';
```

#### **Division Button**

```
num1 = Integer.parseInt(jtfResult.getText());  
jtfResult.setText(null);  
operator = '/';
```

The final part of our code will be the production of result for the operands depending on the selected operator. Just follow the syntax for the equals button below:

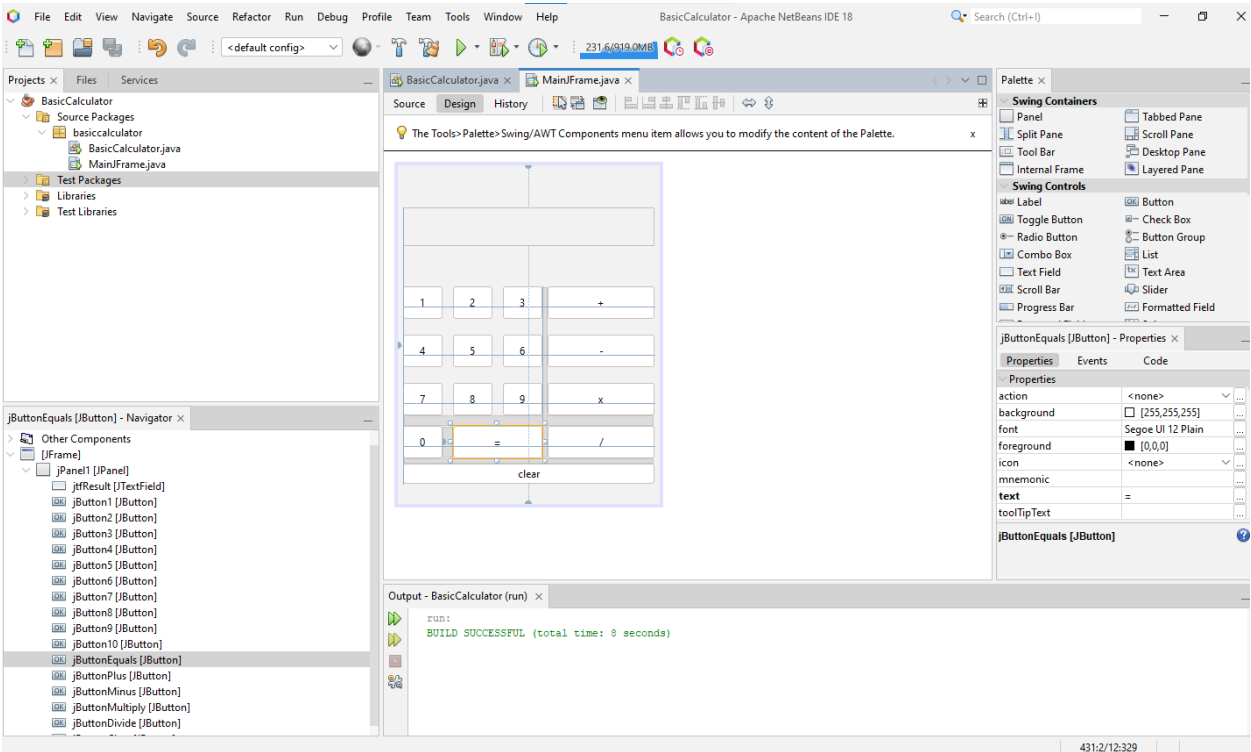


## Equals Button

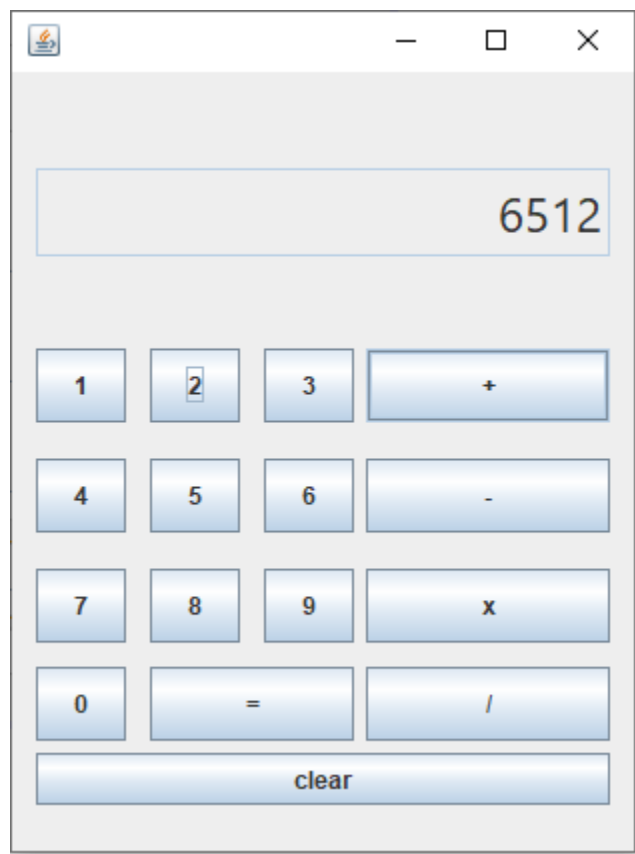
```
private void jButtonEqualsActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    num2 = Integer.parseInt(jtfResult.getText());  
    switch(operator) {  
        case '+':  
            int sum = num1 + num2;  
            result = Integer.toString(sum);  
            jtfResult.setText(result);  
            operator = '\0';  
            break;  
        case '-':  
            int difference = num1 - num2;  
            result = Integer.toString(difference);  
            jtfResult.setText(result);  
            operator = '\0';  
            break;  
        case 'x':  
            int product = num1 * num2;  
            result = Integer.toString(product);  
            jtfResult.setText(result);  
            operator = '\0';  
            break;  
        case '/':  
            float quotient = num1 / num2;  
            jtfResult.setText(Float.toString(quotient));  
            operator = '\0';  
            break;  
        default:  
            jtfResult.setText(null);  
            num1 = 0;  
            num2 = 0;  
            result = null;  
    }  
}
```

That’s basically the core functionalities of a calculator. You can also add a clear button and add the code for it so you can reset the data in the text fields and in the variables. Here’s the code:

```
private void jButtonClearActionPerformed(java.awt.event.ActionEvent evt) {  
  
    // TODO add your handling code here:  
  
    jTextFieldResult.setText(null);  
  
    operator = '\\0';  
  
    num1 = 0;  
  
    num2 = 0;  
  
    result = null;  
  
}
```



# FINAL OUTPUT!



## Follow my socials:

Facebook: <https://facebook.com/tsadiqz3>

Instagram: <https://instagram.com/sepiroth.x/>

Twitter: <https://twitter.com/sepirothx000>

Github: <https://github.com/sepiroth-x>

Patreon: <https://www.patreon.com/vajrayogiii>

Gcash: +639150388448

Email: [richard.cupal@ifamsocial.com](mailto:richard.cupal@ifamsocial.com) , [chardy.tsadiq@gmail.com](mailto:chardy.tsadiq@gmail.com)