

## Lesson 6 – From Designer to Code

🕒 2h

### What students should know

- Class\_Globals
- Variables and Subs
- Passing Values to Code
- Events
- Attributes

Designing the app's appearance in Designer is the first step in the manufacturing stages. Often new developers turn to it to redesign, correct, or add individual information.

When the screen is ready the developer goes to the next stage of programming the functions. That is, everything that elements included in the design to gain functionality. In other words, text boxes can record data, buttons can activate functions, lists can display data, etc.

### Class\_Globals

At the beginning of the code on tab B4XMainPage there is a set of variable declarations between Sub Class\_Globals and End Sub.

```
8 Sub Class_Globals
9     Private Root As B4XView
10    Private xui As XUI
11 End Sub
```

*Picture 1 Sub Class\_Globals*

As it has already been said in 3<sup>rd</sup> Lesson Sub is a set of code that performs a specific operation. The operation of the Class\_Globals is to collect the declarations of variables that we want to be known

throughout the code of the tab B4XMainPage, i.e. in each subprogram.

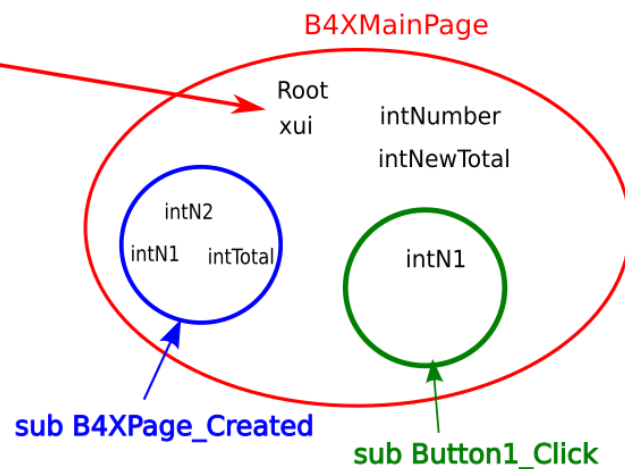
In addition, if a variable statement starts with the public it will be available from other program "tabs".

A deeper look at the use of variables.

In the following code you see three subprograms

### B4XMainPage

```
6 @Sub Class_Globals
7   Private Root As B4XView
8   Private xui As XUI
9   Private intNumber As Int = 32
10  Private intNewTotal As Int
11 End Sub
12
13 @Public Sub Initialize
14 End Sub
15
16 @Private Sub B4XPage_Created (Root1 As B4XView)
17   Root = Root1
18   Root.LoadLayout("MainPage")
19   Private intN1, intN2, intTotal As Int
20   intN1 = 45
21   intN2 = 32
22   intTotal = intN1 + intN2
23   Log("Total = " & intTotal)
24   intNewTotal = intTotal + intNumber
25   Log("New Total = " & intNewTotal)
26 End Sub
27
28 @Sub Button1_Click
29   xui.MsgboxAsync("Hello world!", "B4X")
30   Private intN1 As Int = 5
31   intNewTotal = intN1 + intNumber
32   Log("New Total = " & intNewTotal)
33 End Sub
```



Picture 2 Variables and Range

The `intNumber`, `intNewTotal`, `Root`, and `xui` variables declared within `Class_Globals` "live" within Module B4 XMainPage.

On the contrary, variables `intN1`, `intN2`, `intTotal` 'live' within the sub-program `B4XPage_Created` and for this reason No another subprogram may use them. At the same time, the variable **intN1** who lives in the **Button1\_Click** **is not the same** with the one in **B4XPage\_Created**, both has its own memory space, and both can have the same name.



### Teachers tip

The use and range of variables is something that confuses new developers. Depending on your class, it is recommended to make examples of familiarity in their use.

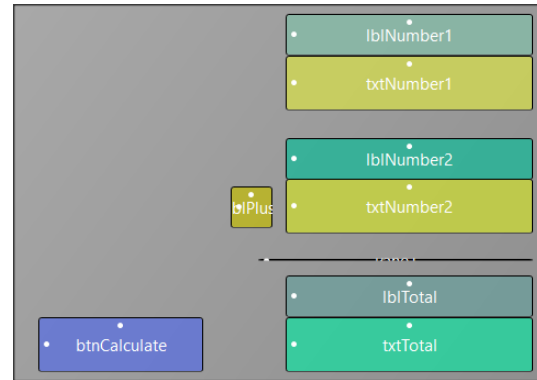
## Passing Values to Code

The screen you have already prepared contains objects that you must declare as variables in `Class _ Globals` to use in the program.



In the example, in the previous lesson, some of the objects on the screen do not have to be included in the code.

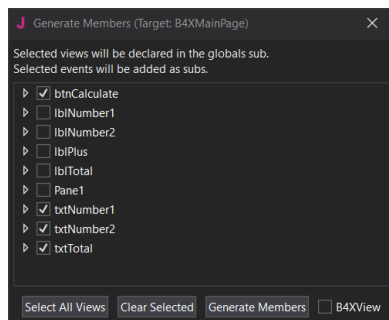
So, all "labels" items in this application do not have to be managed by code as well as the pane element. While button and textFields must be added to program functions on them.



Picture 3 Example 1 Designer View

There are two ways to insert your objects into the code. The first is easily done through the designer with the following functions:

From the Tools menu select Generate Members



Picture 4 Generate Members

In Screen Generate Members just click on objects

- btnCalculate
- txtNumber1
- txtNumber2
- txtTotal

and then click Generate Members.

Your code in the Class\_Globals subprogram will be updated automatically with the variables.

```
7
8 Sub Class_Globals
9     Private Root As B4XView
10    Private xui As XUI
11    Private btnCalculate As Button
12    Private txtNumber1 As TextField
13    Private txtNumber2 As TextField
14    Private txtTotal As TextField
15 End Sub
```

Picture 5 Class\_Globals



### Remember

Each object we import is of a certain type as well as the types of variables.

The second input solution is to write the variables yourself, paying attention so that the names of the objects on your screen are the same as those you write as a variable.

## Events

After you have declared the variables of the objects, the final stage is to enable the functions of the form.

This depends on how you've decided that each app works. In the example with the two numbers there is a button called Calculate and it is what will activate the addition as well as the appearance of the result on our screen.

The operation is triggered by a process called "**Event**". The programmer must detect the event of pressing the key Calculate and when this is done then do the relevant calculations and display the result.



### Remember

There are hundreds of different events happening in one application; the developer determines how the program will react to each of them.

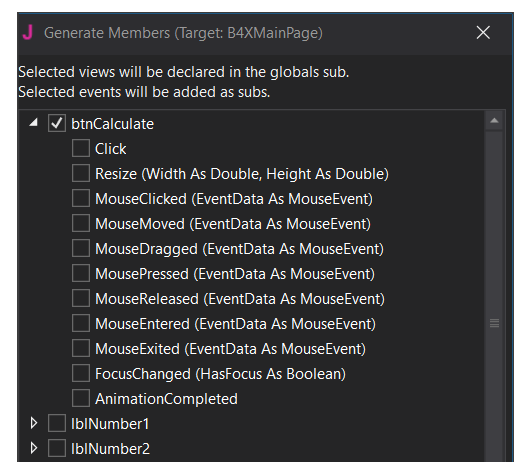
The detection of the event is easy and you just need to create a new subprogram that has the name of the event. This can be done at the same time as the declaration of the variables of our objects through the Designer.

Open its list btnCalculate and several Events we'll be available to choose. Just click on "Click" and "Generate Members".

The subprogram for the **btnCalculate\_Click** event has **already appeared**. Within it you will write the program code to complete.

```
20
21 'This event will be called once, before the page becomes visible.
22 @Private Sub B4XPage_Created (Root1 As B4XView)
23     Root = Root1
24     Root.LoadLayout("MainPage")
25 End Sub
26
27
28 @Sub Button1_Click
29     xui.MsgboxAsync("Hello world!", "B4X")
30 End Sub
31
32 @Private Sub btnCalculate_Click
33
34 End Sub
```

Picture 7 The Click Event



Picture 6 Setting Event

## Writing code in Event

Within an Event subprogram, all the functions associated with it are

usually performed.

```
29
30 @Private Sub btnCalculate_Click
31     txtTotal.Text = txtNumber1.Text + txtNumber2.Text
32 End Sub
```

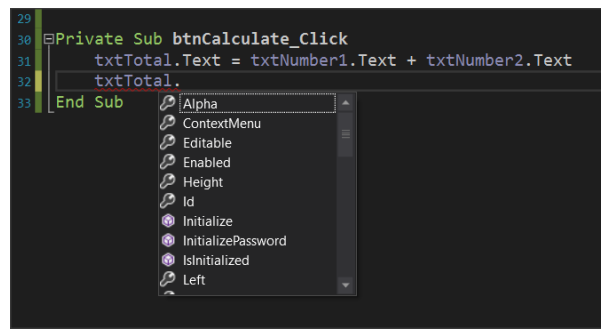
Picture 8 Calculating textFields

Essentially both are achieved with a single command. This is the command of Picture 8 Calculating textFields follows:

*The content of textField txtTotal is equal to the contents of txtNumer1 and txtNumber2.*

## Properties

Each object you insert into your code has a number of different properties. For example properties can be the color, size, location content as they have been Described and in the previous lesson. These properties can provide information or change display issues. For example, the property `txtNumber1.text` provides information on the content of the object `txtNumber1` or can set a value as content depending on how you use the. This information is type string.



Picture 9 Object Properties

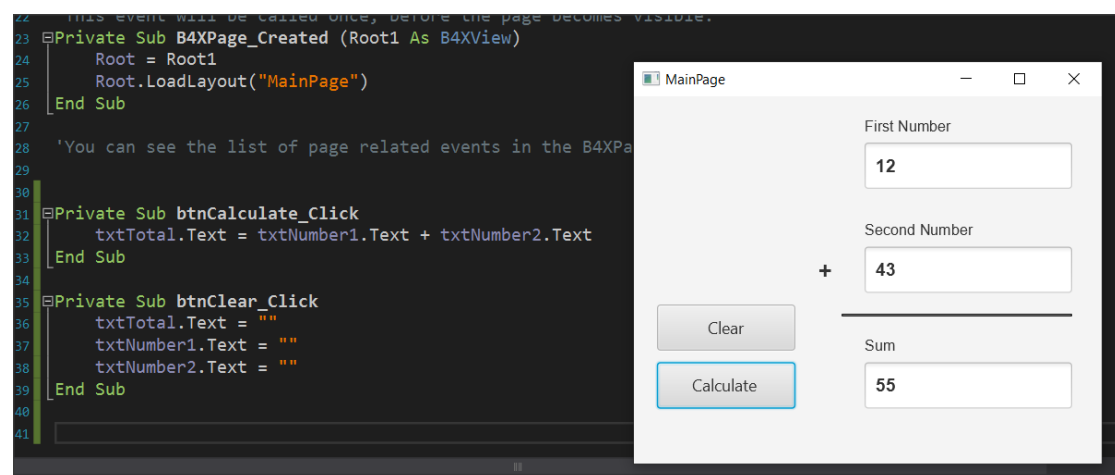


### Remember

You can do operations with string contents when they describe numbers.

Let us now assume that we want to create a new function in the example where another key clears the form to write new numbers. The functions you will perform are as follows:

- Open Designer and add a new button named e.g. `btnClear`.
- Set it as a variable in `Class_Global`.
- Enter the event `btnClear_Click`.
- Set the text properties of `txtNumber1`, `txtNumber2`, `txtTotal` to "".

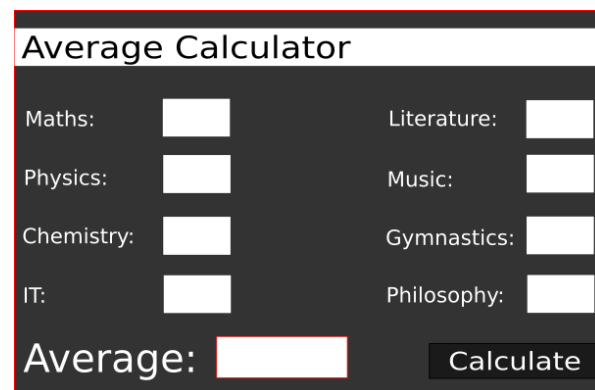


Picture 10 Clear Button and Code

When you press clear, the form will clear.

## Exercises

1. Extend the example to perform the four operations: Add, subtract, multiply, and divide at the click of an appropriate button. Add the appropriate objects to the designer and then complete the code.
2. In exercise 2 of the previous course continue and complete the application by activating its functions.



Average Calculator

Maths:	<input type="text"/>	Literature:	<input type="text"/>
Physics:	<input type="text"/>	Music:	<input type="text"/>
Chemistry:	<input type="text"/>	Gymnastics:	<input type="text"/>
IT:	<input type="text"/>	Philosophy:	<input type="text"/>

Average:

Take care that the numbers in lessons must be between 0-100. If not, you should show an error message.

