

Programming with B4X

Brief Theory

Version 1.0, February 2021




Anywhere Software

Table of Contents

Lesson 1 – Why B4X?	1
Installing B4X	1
Lesson 2 – The meaning of the problem	2
The concept of the problem	2
Understanding the problem	2
Searching for a solution or set of solutions.	2
Choosing the right solution	3
Implementing the solution	3
Checking whether this solution had the desired results.	3
Lesson 3 - My first Program	4
Hello World	4
Recommended instructions for the instructor	4
Exercises	5
Lesson 4 - Variables and Range	6
How to find how many variables you need	7
Naming Variables	8
Declaring Variables	8
My First Variable	8
Comments	9
The log area and the log function.	9
Mathematical Operators	10
Strings	11
Exercises	12
Lesson 5 – Designer	15
First steps with design	16
Visual designer	17
The Views Tree	18
Properties	18
Abstract Designer	18
Example 1	18
Decide on the size of the app screen.	18
Set an appropriate variant.	19
Design a wireframe.	19
Create the views.	19



Exercices	22
Lesson 6 – From Designer to Code	23
Class_Globals	23
A deeper look at the use of variables.	24
Passing Values to Code	24
Events.....	25
Writing code in Event	26
Properties.....	27
Exercises	28
Lesson 7 – Conditional Statements	29
Logical Variables.....	29
Comparative Operators	29
Logical Operators.....	30
Logical operators in programming	31
Examples of evaluation of logical sentences.....	31
If command.....	32
If – Else	33
If – else - else if	33
Algorithms with if	36
Exercises	36
Lesson 8 – Subroutines	39
Create a subprogram in B4J.....	39
Example 1	39
The memory of the subprogram in B4X	41
Return a value from a subprogram.	42
Example 2.....	42
Exercises	43
Lesson 9 – Classes	45
Classes.....	45
Example of class in B4J	46
Implementation methodology	46
Insert a book.	47
Show Book Items.....	48
Change book items.....	48
The Initialize subprogram.	48
Use Class	48



Use of methods	49
Exercises	49
Lesson 10 – B4XPages	51
The structure of an application's folders	51
Starting an application with B4XPage.	52
What is Root	52
Create a new B4XPage	53
Call a new B4Xpage.	55
Close a B4XPage.....	56
Transfer information between pages	57
The Life of B4XPages	58
Exercises	59



Lesson 1 – Why B4X?

🕒 1h

Many new developers wonder in which programming language to invest their time and effort. Each programming language has advantages but also disadvantages that should be considered. Often the selection of a language also determines the subsequent evolution of the developer. Even more, when it comes to using language for educational purposes, there are individual elements that need to be considered.

So, the programming language must be:

- Modern and structured
- Be easy to learn for kids and new programmers.
- Provide an integrated development environment without confusing.
- To be able for the student to develop applications on different platforms like Windows, Android, IOS, Linux, Raspberry Pi, Arduino etc.
- Provide all modern data structures as lists, maps etc.
- To keep students interested by providing the possibility of developing different types of applications for example games etc.

The **B4X** programming language provides all the above features and is also a language for developing high-level applications which can be a springboard for future professional development. In addition, it has a very enthusiastic audience that gladly help in any kind of question.

For more information and material you can visit the website at <https://www.b4x.com/learn.html>

Installing B4X

The most updated information about installing will always be here: <https://www.b4x.com/b4j.html>

Lesson 2 – The meaning of the problem

🕒 1h

What students should know

- What is a problem?
- What is data?
- What is information?
- What steps need to be taken to resolve a problem.
- What is algorithm.

The concept of the problem

Every day in our lives we have problems. Simple problems of everyday life and often even bigger. As a problem we usually define a situation that we are experiencing and which makes it difficult for us to deal with it or solve (Cambridge, 2021).

When we are thinking about a problem these are steps become unconscious in our minds:

- Understanding the problem
- Searching for a solution or set of solutions.
- Choosing the right solution
- Implementing the solution
- Checking whether this solution had the desired results.

Understanding the problem

Understanding the problem is the first step in designing and solving a problem. It is a particularly critical stage because this will also affect the development of the solution. Includes the following steps:

1. Description of the problem
The description of the problem is usually done by ourselves or someone who has it. In this step it is important to clarify all its 'dark' points and to have no doubt as to its wording.
2. Find the data.
Finding the data means what these elements are based on in order to solve a problem, for example in an equation $ax + b = 0$ data are the factors a and b .
3. Find the requested.
The information that we need to find to deal with the problem. Continuing the previous example information is the x of the equation $ax+b=0$.

Searching for a solution or set of solutions.

Once we have recognised the data and what is requested, a solution must be found to solve the problem. Often that's not easy. Thus, it is necessary to look for a method or **a logical set of steps leading to the solution.**

These steps must lead to a solution **whenever** the same problem arises and, moreover, be done in **a relatively short period** of time.

In mathematics and computer science, an algorithm is a finite sequence of well-defined, computer-implementable instructions, typically to solve a class of problems or to perform a computation.

Choosing the right solution

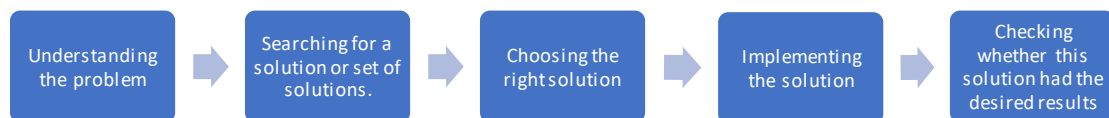
Often a problem can have more than one solution or some of them may be more efficient than others. Choosing the solution correctly leads to a shorter and more reliable outcome.

Implementing the solution

After the resolution method is selected, a series of actions must be taken to implement it. In other words, apply **the Algorithm**.

Checking whether this solution had the desired results.

Finally, after applying the solution it is necessary to test with various data, to examine whether it leads to correct results each time it is performed without problematic situations.



Picture 1 The steps

Lesson 3 - My first Program

🕒 2h

What students should know

- How to start B4J
- How to create and save a new project
- How to run a project
- What is error screen.
- How to see Turtle commands
- How to write a new project using Turtle

Hello World



Make a program that draws a straight line with the help of the turtle on the computer screen.

Recommended instructions for the instructor

The aim of the above exercise is to familiarize students in the creation of a project with B4J and to become the first acquaintance with the programming environment. The following instructions in no way limit the instructor to adapting his course to the relevant educational conditions.



Teachers tip

This is students first lesson. Therefore, keep it simple!

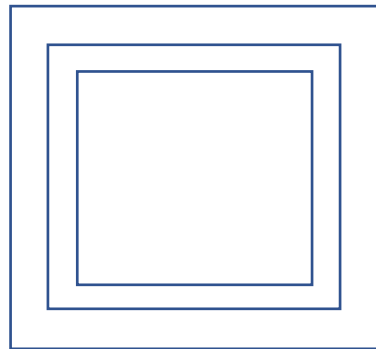
Function	Description
1 How to create a first project with B4J	<i>Menu File -> New -> B4Xturtle Project Folder Project Name</i> Explain to students the importance of the right names in each project they create and the value of correct storage in folders in a structured way
2 Run Project	<i>Menu Project -> Compile & Run</i> Explain what means compile and how to recognize syntax errors in log. You don't need to provide to much information about compilation. Just the basic stuff in order to run a project.
3 #Region Project Attributes	Change Values in <i>MainFormWidth</i> and <i>MainFormHeight</i> to make different size application
4 Sub Turtle_Start	<i>What means Sub?</i> A small amount of code which is doing a certain activity.



Function	Description
5 Turtle methods	What is Turtle? What .MoveForward do? How can we find more commands? (Tell students to write "Turtle." To see the list of methods.
6 Errors	How to identify an error in log screen

Exercises

- Using turtle and methods **MoveForward**, **TurnLeft**, **TurnRight** draw a square with any size you want.
- Using Previous commands and **PenUp**, **PenDown** and **Move** draw 3 squares like the image bellow.



- Using previous commands design a sketch of your choice. Give a name to your sketch and explain how you did it.

Lesson 4 - Variables and Range

🕒 3h

What students should know

- Explain how a computer stores data in ram.
- Explain what a variable is.
- How to name a variable
- Assign a value to a variable.
- Use Mathematical Operators
- Use log command to display a variable

Imagine that you live on a street with a few million houses all in a row; each house has as you all know its address which starts at number 1 and ends at the last house; in order to be able to locate a friend who lives on that street you need to know the number of the house; so we have on the one hand a house number and on the other the friend who lives in that house.

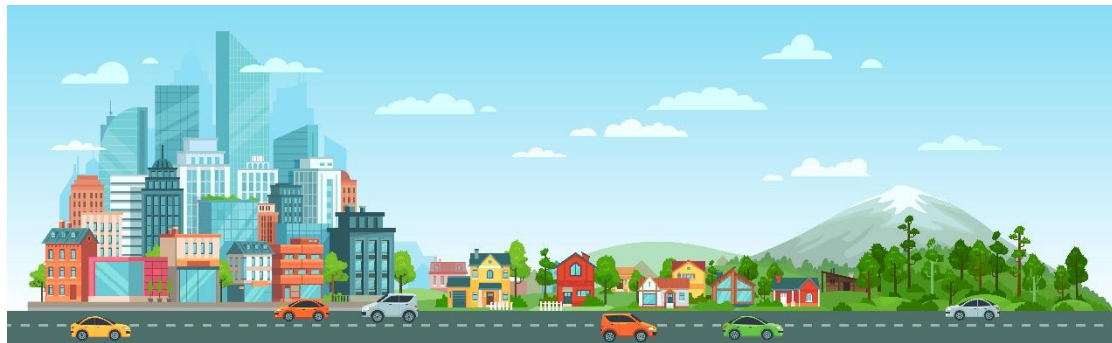


Figure 1 Computer Memory (<https://www.freepik.com>)

The computer's central memory works in much the same way. There are many houses each with its address and a "resident" within each house. This address is called memory address and the "resident" content. On the computer often the "resident" who from now on we will call variable needs more houses to fit.

For a programmer to use memory, he must know the data he needs and the type of data. These can be integers or real numbers, words or letters, some reasonable values (truth, false). He also needs a "home" in the computer memory to store them represented by address.

In B4X the data can be stored in different types as:

B4X	Type
Boolean	boolean
Byte	integer 8 bits
Short	integer 16 bits
Int	integer 32 bits
Long	long integer 64 bits
Float	floating point number 32 bits
Double	double precision number 64 bits
Char	character



String

array of characters

Table 1- Simple types of variables

Every type needs different space in memory to store values.

Because it is difficult for the developer to remember all the addresses of his data, each address corresponds to a name. Fortunately, in fact this is done by the programming language itself and all it takes is to think of a good name for his data. For example, a data that is an integer for age could be called "age". Now, there is a "home" called age in computer memory.



Remember

Variables are used to store information to be referenced and manipulated in a computer program. They also provide a way of labeling data with a descriptive name, so our programs can be understood more clearly by the reader and ourselves. It is helpful to think of variables as containers that hold information.

How to find how many variables you need

In any programming problem that a developer encounters, they should be able to locate the data and the information's of the problem.

In programming we name all those elements that we need to know to move forward in solving a problem. Usually in a programming problem we find them in the pronunciation of the exercise with the help of keywords such as:

- Reads
- Registers
- Ask
- Accept
- Type

Example 1: Write a program that converts the euros we type into dollars.

Example 2: Make a program that accepts a positive integer and calculates its square, cube, and square root.

Information's

Information's in programming are all the elements that we need to calculate after processing our data. We usually find them in pronunciation using keywords such as:

- Calculates
- Displays
- Writes
- Counts
- Convert

In the previous examples what are the requested?



Naming Variables

The names of the variables in B4X must follow specific rules.

- They must start with a capital or small character.
- They can then have digits or the character underscore.
- B4X does not distinguish capital and small letters.

Also, it's a good practice to name variables beginning with 3 small letters indicating the kind of a variable and continue with 1 uppercase letter and a meaningful word. For example:

- Dim **intAge** as Int
- Dim **fltAmount** as Float
- Dim **strName** as String

This practice helps a lot when you find variables in the code to recognize the type and the value it stores.

Declaring Variables

My First Variable



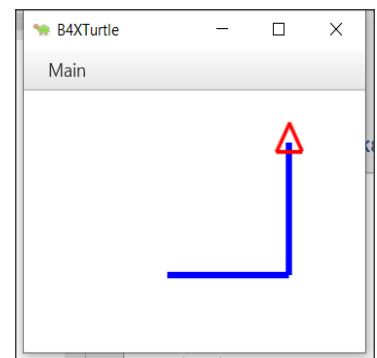
Make a program that assigns a value into integer and then draws with the help of the turtle a line of length as long as the value in the variable.

In B4X to use a variable we must first inform the language of its existence in order to commit space in the computer's memory to store its value.

For example, in the following code, the statement is as follows:

```
'Program: My First Variable
Sub Turtle_Start
  Private intDistance As Int
  Public intTurn As Int
  intDistance = 100
  intTurn = 90

  Turtle.SetPenColor(xui.Color_Blue).SetPenSize(5)
  Turtle.MoveForward(intDistance)
  Turtle.TurnLeft(intTurn)
  Turtle.MoveForward(intDistance)
End Sub
```



The declaration of variables begins with the keyword Private or Public.

Private means that the variable is known only in the specific space declared and no other program or subprogram does not know its existence and thus the value it contains.

Instead, a variable statement that starts with the keyword Public can be known to other programs or subprograms or classes etc.

After the keyword Private or Public follows the name of the variable we chose to give. This is where the rules discussed above apply. Finally, the type of the variable follows. For simple variables these are all those described in the *Table 1- Simple types of variables*.



Teachers tip

You don't have to explain all the variables already as well as their use. For your students to start programming, the basics of integer, float, string are enough. As you progress through the courses you can include other types according to your needs.

Comments

In computer programming, a comment is a programmer-readable explanation or annotation in the source code of a computer program. They are added with the purpose of making the source code easier for humans to understand and are generally ignored by compilers and interpreters. The syntax of comments in various programming languages varies considerably. (Wikipedia, 2021)

In B4X comments are inserted by writing the character ' as their first letter. From this point on it is not recognized by the translator of the language. Generally, in B4X comments you should put anywhere it is important to remember what you are doing as well as before the subprograms to explain what their job is. Comments are easily distinguished in code from the green color given to them by the programming environment (IDE).

Example

```
'Program: My First Variable
'This program draws a right angle, with sides as much as the
'value of the intDistance variable
Sub Turtle_Start
  Private intDistance As Int
  Public intTurn As Int
  intDistance = 100      'The sides of the right angle
  intTurn = 90           '90o angle

  Turtle.SetPenColor(xui.Color_Blue).SetPenSize(5)
  Turtle.MoveForward(intDistance)
  Turtle.TurnLeft(intTurn)
  Turtle.MoveForward(intDistance)
End Sub
```

The log area and the log function.

During programming various errors occur. Generally, errors in programming are divided into two categories syntax and logical. For now, we will deal with the syntax errors that are recognized by the programming language and indicate them on the logs screen. In order to access the logs screen we need to click on the relevant logs tab at the bottom right. The Logs screen itself is divided into two frames, the first of

which displays errors and the bottom screen displays language messages or that information we want to display using the log() function. Using the Log() function helps the developer display messages while running a program as well as variable values to help control the program's proper operation.

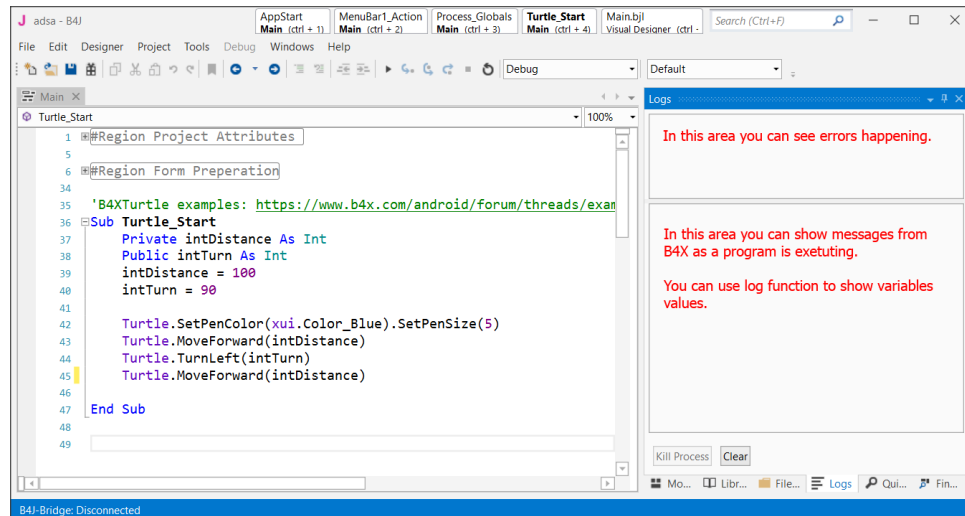


Figure 2 Logs Screen

To display any information on the screen it is sufficient to use the log() function as the example of the following image.

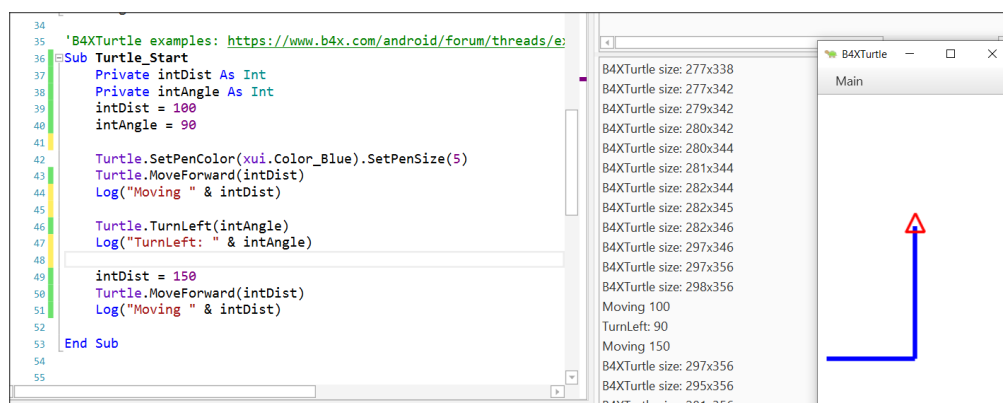


Figure 3 Using log function

Mathematical Operators

B4X supports all known mathematical operations:

Operator	Example	Operation
+	$x + y$	Addition
-	$x - y$	Subtraction
*	$x * y$	Multiplication
/	x / y	Division
Mod	$x \text{ Mod } y$	Modulo
Power	$\text{Power}(x, y) \quad x^y$	Power of

Examples:

```
Private intA, intB, intC, intS As Int
Private fltD, fltM As Float
intA = 40
intB = 20
intC = 30

intS = intA + intB + intC
Log(intS)                \Shows 90

fltD = intS / 3
Log(fltD)                \Shows 30

intA = intA + 1           \Increase intA by 1
Log(intA)                \Shows 41

intS = Power(intA - 11, 2) \ 302
Log(intS)                \Shows 900

fltM = intA mod 2         \41 modulo 2
Log(fltM)                \Shows 1
```

Strings

In computer programming, a string is traditionally a sequence of characters, either as a literal constant or as variable. The latter may allow its elements to be mutated and the length changed, or it may be fixed (after creation) (Wikipedia, Wikipedia - Strings, 2021).

A string is declared like the other variables using the String statement.

```
Private strName as String
```

Assigning value to a string can be done with the = symbol or by reading a value from the user (something we'll see later).

```
Private strName, strSurName as String
strName = "George"
strSurName = "Smith"
```

Also we can string together using the character &.

```
Private strName, strSurName as String
strName = "George"
strSurName = "Smith"
Private strPerson as String

strPerson = strName & " " & strSurName
log(strPerson)                \ shows George Smith in log screen

Private strName2 as String
strName2 = "John"
strName2 = strName2 & " Smith"
```

There are also a lot of functions regarding strings that makes our life easier when we are dealing with them:

CharAt (Index)	Returns the character at the given index.
CompareTo (Other)	Lexicographically compares the string with the Other string.
Contains (SearchFor)	Tests whether the string contains the given SearchFor string.
EndsWith (Suffix)	Returns True if the string ends with the given Suffix substring.
EqualsIgnoreCase (Other)	Returns True if both strings are equal ignoring their case.
Length	Returns the length, number of characters, of the string.
Replace (Target, Replacement)	Returns a new string resulting from the replacement of all the occurrences of Target with Replacement.
StartsWith (Prefix)	Returns True if this string starts with the given Prefix.
ToLowerCase	Returns a new string which is the result of lower casing this string.
ToUpperCase	Returns a new string which is the result of upper casing this string.
Trim	Returns a copy of the original string without any leading or trailing white spaces.

Table 2 String Functions (<https://www.b4x.com/android/documentation.html>)



Teachers tip

You can find more information about string manipulation in language booklets at (<https://www.b4x.com/android/documentation.html>)

Exercises

1. In the following exercises, identify the variables you need to declare. For each of them, write the relevant statement and give it an appropriate name.
 - Calculate the volume of a cylinder with a radius of one metre and a height of two metres.
 - Make a program that accepts a positive integer and calculates its square, cube, and square root.
 - Make a program that reads a sum of money in € and calculates and displays the corresponding amount in \$.
 - Write a program that reads the length of the sides of a rectangle from the keyboard and calculates and displays its area.
 - The total resistance R of two resistances R_1 and R_2 connected in series is $R_1 + R_2$ and parallel $(R_1 * R_2) / (R_1 + R_2)$




respectively. Make a program that it reads two values of resistant R1 and R2 and calculates the total resistance in series and parallel.

2. In the following variable names, select which are correct and which are not:
intAge

Name	True	False
int Age	<input type="checkbox"/>	<input type="checkbox"/>
_fltAmount	<input type="checkbox"/>	<input type="checkbox"/>
strName	<input type="checkbox"/>	<input type="checkbox"/>
1myAge	<input type="checkbox"/>	<input type="checkbox"/>
int_value	<input type="checkbox"/>	<input type="checkbox"/>

3. It's the end of the semester and you got your grades from three classes: Geometry, Algebra, and Physics. Create a program that: gives in 3 variables the grades of these 3 classes (Grades range from 0 - 10) Calculate the average of your grades.
4. You have bought a Bitcoin and now it's on the rise!!! Create a program that:
- Assign the value of the bitcoin at the time of purchase.
 - Assign the percentage of increase (or decrease)
 - Logs the total value of your bitcoin.
 - Logs the increase or decrease value.
5. You now own some property, and you want to calculate the total area of the property. Create a program that:
- Assign the width and height in two variables.
 - Calculate and log the area.
6. You are interested in buying a new laptop. You check the price and you see that the price is 300\$ without the 10% tax. Create a program that:
- Assign the the price of the laptop in a variable.
 - Assign the tax percentage in a second variable.
 - Calculate and logs the total amount.
7. In a company the monthly salary of an employee is calculated by the minimum wage 400\$ per month, plus 20\$ multiplied by the number of years employed, plus 30\$ for each child they have. Create a program that:
- Assign the number of years employed in a variable
 - Assign the number of children the employee has in second variable.

- 
- Calculate and logs the total amount of salary the employee makes.
8. Create a program that log the last digit of a given integer.
 9. Create two variables a and b, and initially set them each to a different number. Write a program that swaps both values.

Example: a = 10, b = 20

Output: a = 20, b = 10

10. Create two variables 'a' and 'b', and initially set them each to a different number. Write a program that double the Value of 'a' variable and increase the value of 'b' by 1.

Example: a = 10, b = 20

Output: a = 20, b = 21

Lesson 5 – Designer

🕒 2h

What students should know

- Talking about Designer
- Design the first Screen.
- Inserting and customizing Views: Labels, TextFields, Buttons, Panes
- Saving forms
- Design their own Main Screen using wireframes.

Until now you have used the turtle to move it through the screen, and the log command to display information on the language log screen. What if you ask the program user to enter values? Or what happens when you want to display information to the user? The B4X has a special interface screen design environment. Through it you can design the appearance of the screens and generally communicate with the users of your application.

Every time you must design an app you should keep in mind that the look of your app is what will attract users to it. In other words, it is not enough to be simple functional but also easy to use as well as to offer information in an organized way without confusing.

Before designing any app remember some key design elements (usability.org, 2021):

Keep the interface simple. The best interfaces are almost invisible to the user. They avoid unnecessary elements and are clear in the language they use on labels and in messaging.

Create consistency and use common UI elements. By using common elements in your UI, users feel more comfortable and can get things done more quickly.

Strategically use color and texture. You can direct attention toward or redirect attention away from items using color, light, contrast, and texture to your advantage.

Use typography to create hierarchy and clarity. Carefully consider how you use typeface. Different sizes, fonts, and arrangement of the text to help increase scanability, legibility and readability.

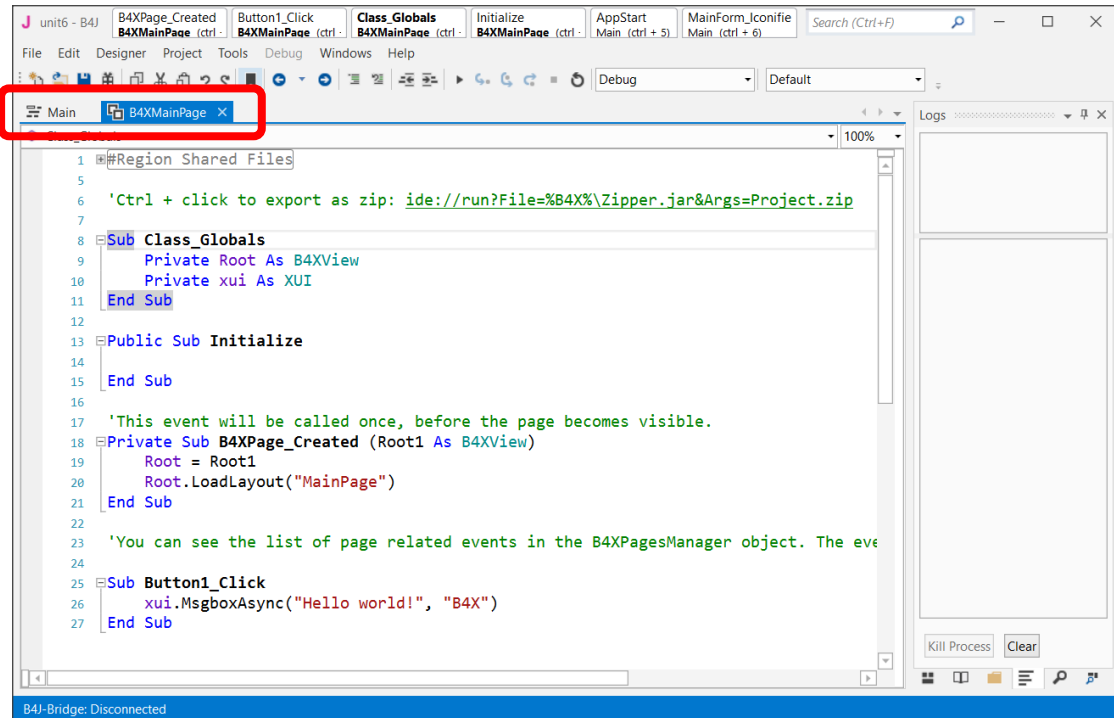
Make sure that the system communicates what's happening. Always inform your users of location, actions, changes in state, or errors.

Think about the defaults. By carefully thinking about and anticipating the goals people bring to your site, you can create defaults that reduce the burden on the user. This becomes particularly important when it comes to form design where you might have an opportunity to have some fields pre-chosen or filled out.



First steps with design

First of all, you should begin the B4J and now from file menu choose **New** and **B4XPages**. Choose a directory and write a name for your project. You will see the code bellow. There are two tabs of code here, the first one called **Main** and the second **B4XMainPage**.



Do not worry about them now. We will discuss later about them. Now all you need to know is that inside the B4XMainPage all the beautiful things happen for our code!

Now from the **Designer menu** select **Open Internal Designer**.

This is where the design process begins. Two windows will open, the first is the designer and the second is the preview of the screen you are designing.

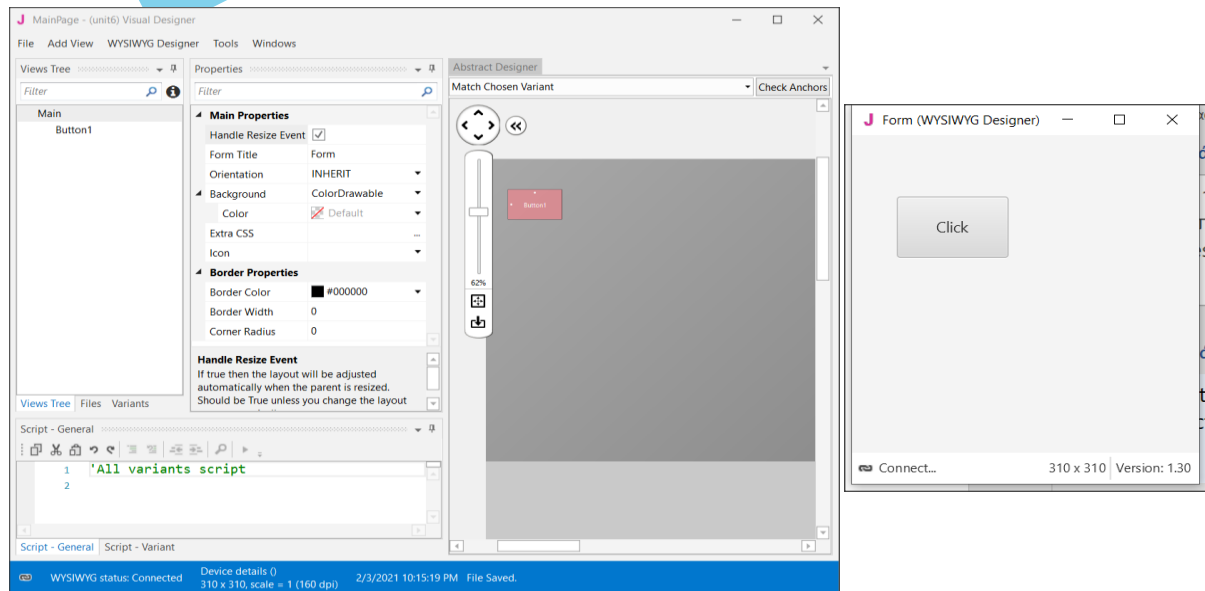


Figure 4 Designer Screen

Visual designer

The AddView menu includes all the objects needed to create our screen.

Select a label from the menu, and then move it to the preview screen where you decided before in the wireframe stage.



Remember

You can move all objects around the view by selecting them and holding the left mouse key.

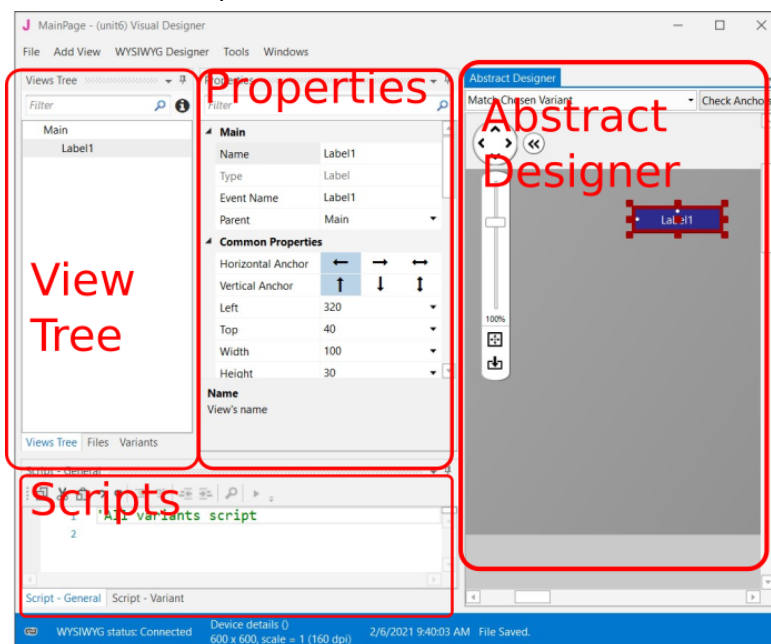


Figure 5 Designer's parts

The Views Tree

Here you see all the objects in your design. Keep in mind that the objects above the list are placed behind the next ones.

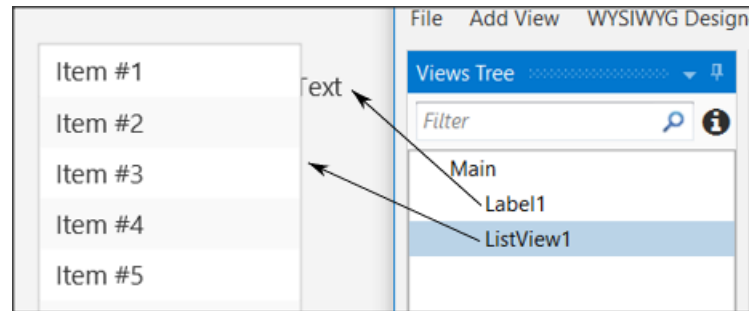


Figure 6 Views Tree

Properties

Each object has properties such as size, screen position, colors, font, etc. Each property can be changed either through the properties option or later through the program code.

One of the most important properties is the name of the object. This, like variables, should follow specific rules to indicate their type. For example in *Table 3* Naming objects some cases of names.

Type	Prefix	Example
Label	lbl	lblName
Button	btn	btnSave
TextField	txt	txtAge
Spinner	spn	spnYears
Pane	pn	pnLine1

Table 3 Naming objects

Abstract Designer

The Abstract Designer allows to select position and resize Views. It is a very useful function for quickly placing objects in the correct position (however the most accurate placement is made by the Properties tab by setting the relevant values).

Example 1



Imagine that you want to make a program that reads from the screen two Integers, calculates, and shows the sum.

Decide on the size of the app screen.

This depends on the amount of information we must display as well as on the individual items such as menus, graphics etc.

To set the application's size before beginning the Designer first go to **Main** Tab and change the first lines of code Width and Height:

```
#Region Project Attributes
    #MainFormWidth: 600
    #MainFormHeight: 400
#End Region
```

Save your project and open Designer.

Set an appropriate variant.

Usually, you should set the variant as the MainFormWidth and MainFormHeight. This will help you plan without the risk of going outside the screen limits.

Choose Variants and then New Variant and set the width and height.

You can have as many variants as you want for different screen size but for now, we stay to only one. Also, you can remove any variant by selecting it and choosing "Remove Variant".

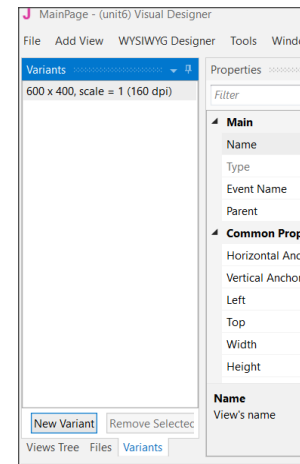


Figure 7 Variants Screen

Design a wireframe.

For small applications, this step is optional, but it is a good habit to have decided from the beginning where you want to display your details. You can use a simple sheet of paper or several programs to help create previews.

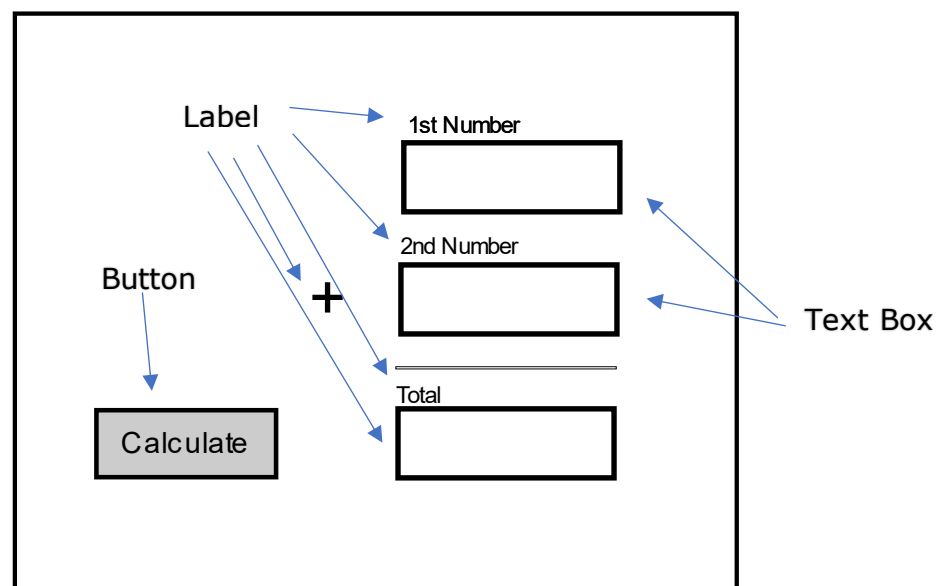


Figure 8 Wireframe

Create the views.

Now that you know what you need and where to place it, use the Designer tools to complete the process.

Inserting a label.

From View menu select label and you will see a label object in your View Tree and in the Abstract Designer. Move it in the place you decide in wireframing and choose an appropriate name from properties.

Now scroll down the Properties and set:

- **Width:** 180
- **Height:** 30
- **Text:** First Number
- **Alignment:** CENTER_LEFT
- **Font:** SansSerif
- **Size:** 13

Experiment with the other settings and see it displayed in the preview pane.

Insert a second label or you can also duplicate the first one. Select it and press Ctrl-D. The second method gives a same label as the first one with the same properties except Name Property. Set "lblNumber2" as name and "Second Number" as Text and Create a third label with name "lblTotal" and Text: "Total".

Inserting a Text Field.

Text Fields are used to import data into the program. There is no restriction on the type of data you can import. From View Menu choose TextField and set:

- **Name:** txtNumber1
- **Width:** 180
- **Height:** 40
- **Font:** SansSerif
- **Bold:** checked

Place the textField underneath "First Number" label. Now put a same TextField with name "txtNumber2" and put it underneath label "Second Number". At the end create a third TextField with Name "txtTotal". You will probably see something like the Figure 10 Text Fields

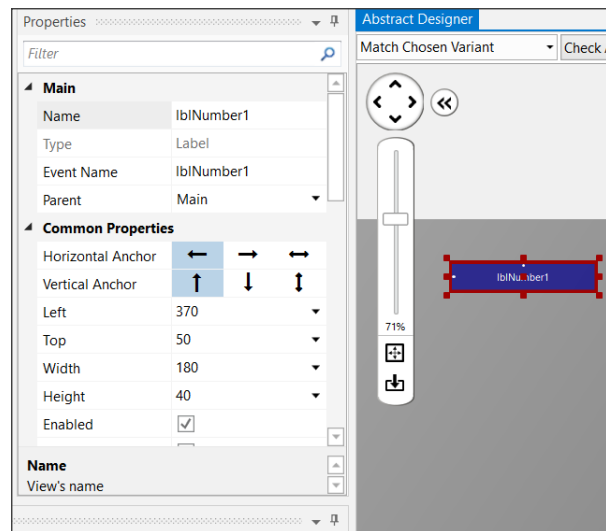


Figure 9 Labels

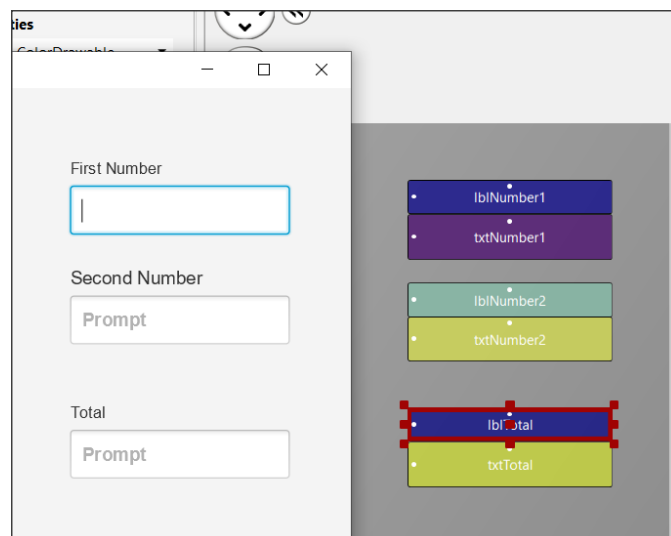


Figure 10 Text Fields

Inserting a button

Buttons in an app are used to enable functions. The program detects the click and then executes appropriate commands depending on the button pressed.

For each button you can set different features such as size, color, shape, etc. to stand out on your screen and be easily detected by users of your app.

From the Views menu select Button, and then set:

- **Name:** btnCalculate
- **Width:** 150
- **Height:** 40
- **Border Color:** #3C0000
- **Border Width:** 2
- **Corner Radius:** 20
- **Text:** Calculate
- **Text Color:** #FF3C0000
- **Font:** SansSerif
- **Size:** 15
- **Bold:** checked

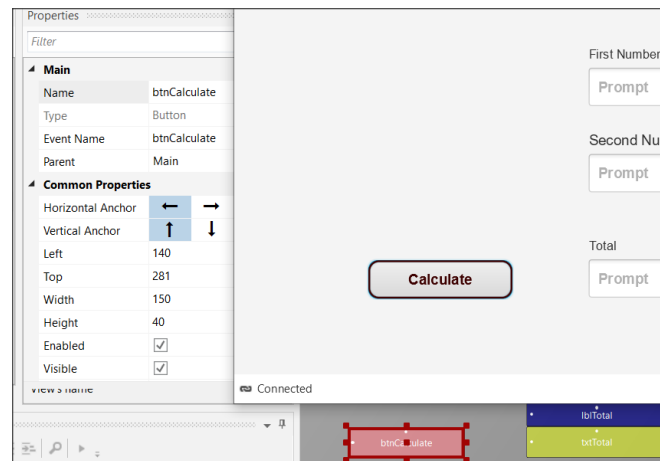


Figure 11 Buttons



Remember

File -> Save (or Ctrl - S) every time you make something valuable!

Inserting a Pane

You can use a Pane to visually group specific objects on the screen you're drawing. The pane displays a frame, and you can specify properties such as color, border, fill, etc. You can also use it at a very small height (1 or 2) to display a single line on your screen.

This example is used to draw a line before the total. From menu AddView insert Pane and set:

- **Name:** pnLine
- **Width:** 180
- **Height:** 1
- **Border Color:** #000000
- **Border Width:** 2

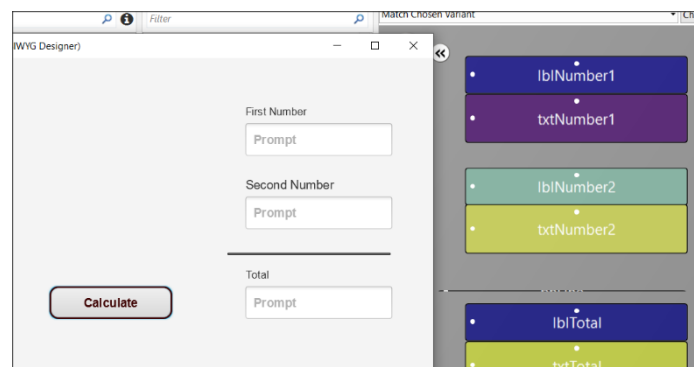


Figure 12 Pane

Now from menu File save the form. The form has already a name (MainPage) so you don't need to give an other name.

Exercises

1. Use the designer to create the following wireframes.



Teachers tip

This is the fun part. You can also leave students free to experiment with views.

A wireframe of a form for student information. It contains four text input fields arranged in a 2x2 grid. The labels 'Name', 'Class', 'Surname', and 'Age' are positioned above their respective input fields. Below the input fields are three buttons: 'Save', 'Clear', and 'Cancel'.

A wireframe of an 'Average Calculator' application. The title 'Average Calculator' is at the top. Below it, there are eight input fields arranged in two columns. The left column has labels 'Maths:', 'Physics:', 'Chemistry:', and 'IT:'. The right column has labels 'Literature:', 'Music:', 'Gymnastics:', and 'Philosophy:'. At the bottom left, there is a label 'Average:' followed by a larger input field. At the bottom right, there is a 'Calculate' button.

2. Think and design your own Dream Application. Give it a name, create a wireframe in your notebook and create the Design View.

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 2 Sub Class_Globals

As it has already been said in 3rd 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.

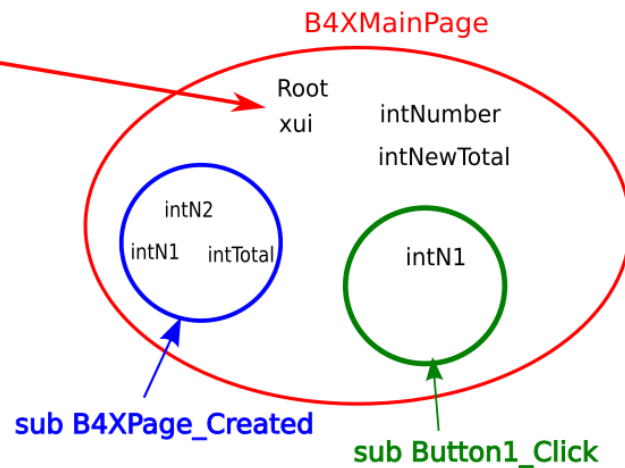
In addition, if a variable statement starts with the public it will be available from other program "tabs".

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
34
```



Picture 3 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.

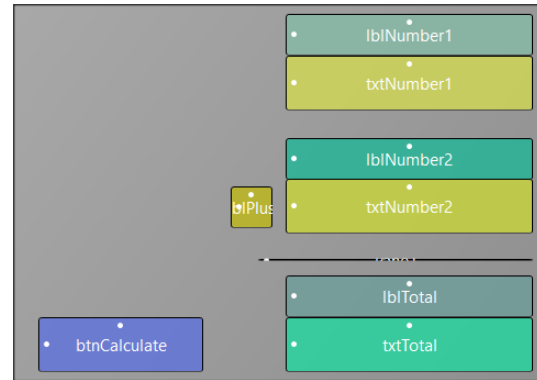
Passing Values to Code

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.

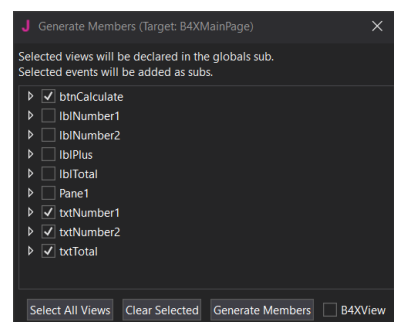
So, all "labels" items in this application do not have to be managed by code as well as the pane element. While button and textFields must be added to schedule functions on them.



Picture 4 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 5 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 6 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.

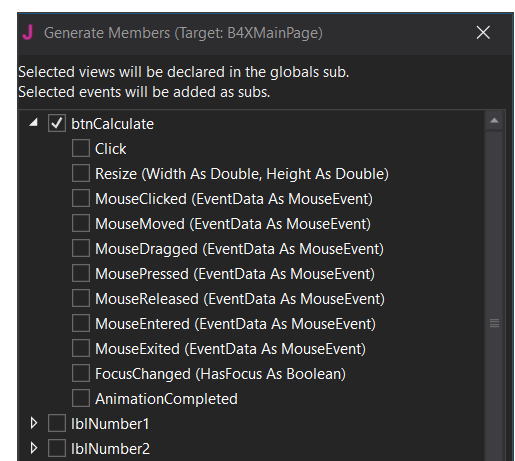
The detection of the event is easy and you just need to create a new subprogram that has the name of the event. This can be done at the same time as the declaration of the variables of our objects through the Designer.

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 8 The Click Event



Picture 7 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 9 Calculating textFields

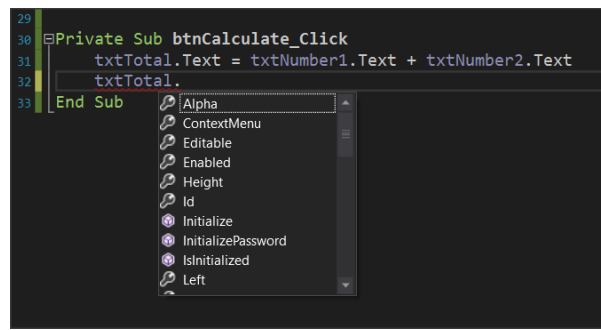
Essentially both are achieved with a single command. This is the command of Picture 9 Calculating textFields follows:

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 10 Object Properties

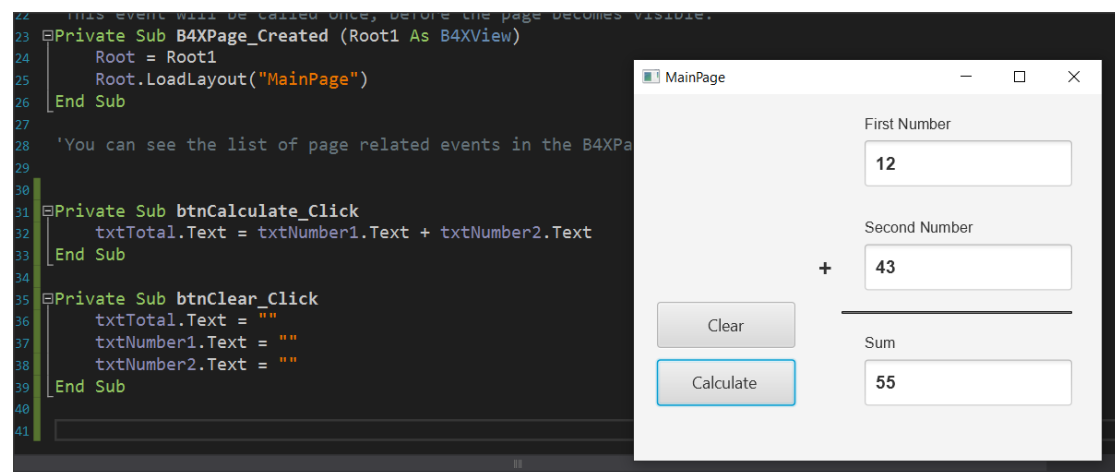


Remember

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 "".

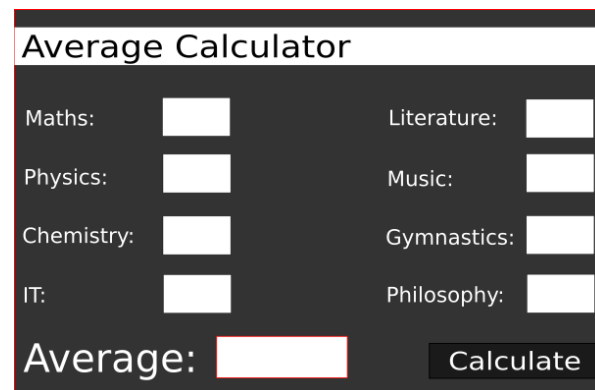


Picture 11 Clear Button and Code

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:

Physics:

Chemistry:

IT:

Literature:

Music:

Gymnastics:

Philosophy:

Average:

Calculate

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

Lesson 7 – Conditional Statements

🕒 4h

What students should know

- Boolean Variables
- Relational Operators
- Logical Operators
- If Statement
- If-Else Statement
- If-Else-Else If Statement
- MAX Algorithms

Logical Variables

So far, we have seen three types of variables integer, real, and string. As a type it is extremely simple because it can accept only two different values: True – False which in fact internally on the computer are translated into states 1 and 0 (leaked by current or not).

The statement of a logic (from now on we will call it Boolean) becomes like the other simple variables we have known:

```
Private intDistance = 100, intTotalTravel = 0 As Int
Private blnFlag As Boolean = False
Private blnDone As Boolean
```

Comparative Operators

Comparative operators are used to make comparisons between values in the programming languages. They are the known mathematical symbols of inequalities only that on the computer are written in a slightly different way.

Mathematical Symbol	B4X	Meaning
=	=	Equality
≤	<=	Smaller or Equal
≥	>=	Greater or Equal
≠	<>	Different
<	<	Smaller
>	>	Larger



Generally, to make a comparison you must compare variables or values of the same type. Eg. Integer values with integer values, real with real prices, etc. Also, in B4X you can compare numeric variables, such as integer variable and float, or Strings (strings) and numeric variables because internally the language converts strings into numbers.

```
38 Private intDistance = 100 As Int ' Notice the different declaration
39 Private intTotalTravel As Int = 0 ' of two integers
40 Private fltD As Float = 100.45
41 Private strN As String = "100"
42 Private s As String = "School"
43
44 Log( fltD > intDistance) 'Shows True
45 Log( strN = intDistance) 'Shows True
46 Log( strN = intTotalTravel) 'Shows False
47 Log( s = intTotalTravel) 'Shows False
48 Log( intTotalTravel <> strN ) 'Shows True
```

Picture 12 Variable Comparisons



Remember

The result of a comparison is always a logical value of True or False.

Logical Operators

Consider a sentence such as 'I am going to school now'.

Mathematician George Boole developed an algebra based on logical sentences.

In Mathematics and Mathematical Logic, Boolean Algebra is the algebra where the values of the variables are the true and false, usually represented by 1 and 0 respectively. Unlike elementary algebra where the values of variables are numbers and the main acts are addition and multiplication, in Boolean there are three main acts **And**, **OR** and Denial **No**.



Remember

AND (conjunction), denoted $x \text{ AND } y$, satisfies $x \text{ AND } y = 1$ if $x = y = 1$, and $x \text{ AND } y = 0$ otherwise.

OR (disjunction), denoted $x \text{ OR } y$, satisfies $x \text{ OR } y = 0$ if $x = y = 0$, and $x \text{ OR } y = 1$ otherwise.

NOT (negation), denoted $\text{NOT } x$, satisfies $\text{Not } x = 0$ if $x = 1$ and $\text{Not } x = 1$ if $x = 0$.

Example:

1st Sentence: Today it rains,

2nd Sentence: *I am going to school.*

**Today it's
raining.
(P1)**

***I am going to
school.
(P2)***

**P1 AND
P2**

P1 or P2

NO P1



True	True	True	True	False
True	False	False	True	False
False	True	False	True	True
False	False	False	False	True

From the table we observe that

- Two sentences that unite with the logical **AND** are true when it unites two truths only.
- Two sentences that are united by logical **OR** are true when even one sentence is true.
- The **logical NO** reverses the truth or lie of a sentence.

Logical operators in programming

Logical operators are used in programming to create complex comparative expressions. This helps the developer optimize their code with fewer lines and simpler code.

In B4X logical variables are used as below.

```

50 'Logical operators
51 Private blnL1, blnL2 As Boolean
52 blnL1 = True
53 blnL2 = False
54
55 Log (blnL1 And blnL2) ' Shows False
56 Log (blnL1 Or blnL2) ' Shows True
57 Log (Not(bl nL1)) ' Shows False
58 Log (Not(bl nL2)) ' Shows True

```

Picture 13 Use of Logical Acts

Notice that logical operations must link logical variables or logical expressions as we will see later.

Examples of evaluation of logical sentences.

The following variables are given with their values:

```

Private intA = 10, intB = 20, = 30 As int
Private strName1 = "George", strName2 = "Georgia" As String
Private blnA = True, blnB = False blnC = False As Boolean

```

Calculate the value of the following logical expressions.

1.

blnA	AND	blnB
True		False
False		

2.

Inta	>	intC	AND	blnA
10		30		True
False		False		

3.

intA + intB	>=	intC	AND	(blnA	OR	blnB)
10 + 20		30		True		False
True		True		True		

4.

intA + intB	>=	intC	OR	(blnA	AND	blnB)
10 + 20		30		True		False
True		True		False		

5.

strName1	=	"George"	OR	strName2	=	"John"
George		George		Georgia		John
True		True		False		

If command

Often as in life we ask questions so in programming there is a need for the developer to ask questions to check values or change the continuity of the program in different directions.

The **if command** is used to make the corresponding questions:

Its basic form is as follows:

```
If ( condition ) Then
    Commands
End If
```

Where condition enter a comparative or logical expression studied above.

The meaning is: If condition is **TRUE** execute the commands between Then and End If

Examples:

```
Private intA = 10, intB = 20 As Int
Private fltA As Float

If intA > 0 Then
    Log(intA & " is positive Number")
End If
```



```

If intA > 10 Or intB > 10 Then
    Log("One or both numbers are greater than 10")
End If

If intA Mod 2 = 0 Then
    Log(intA & " is Even number")
End If

```

If – Else

The Else command adds the ability to **if** to execute code if its condition is not true.

Its basic form is as follows:

```

If ( condition ) Then
    Commands
Else
    Commands
End If

```

The meaning is: If condition is **TRUE** execute the commands between Then and Else otherwise Execute the commands between Else and End If.

Examples:

```

Private intA = 10, intB = 20 As Int
Private fltA As Float

If intA > 0 Then
    Log(intA & " is possitive Number")
Else
    Log(intA & " is not possitive Number")
End If

If intA > 10 Or intB > 10 Then
    Log("One or both numbers are greater than 10")
Else
    Log("None of the two numbers are greater than 10")
End If

If intA Mod 2 = 0 Then
    Log(intA & " is Even number")
Else
    Log(intA & " is Odd number")
End If

```

If – else - else if

Multiple **if** further extends the functionality of an **if** command by adding more than 1 control to the structure.

How to write:

```

If ( condition1 ) Then

```



```

Commands
Else If (condition2 ) Then
  Commands
Else If ( condition3 ) Then
  Commands
Else If ( condition4 ) Then
  Commands
...
Else
  Commands
End If

```

The function of the multiple if is summarized as follows:

1. The first condition runs, and if it is true, then the code it contains runs and if it is completed.
2. If the first **if** is false then the second is executed and if it is true it executes the code it contains and if it is completed.
3. Other checks are always performed when the previous ones are false.
4. If none of the checks are true then the **else** command is run which is optional.

Example 3

A fast-food chain has these meals:

Meal	Price
Burger	5\$
Pizza	3\$
Hot Dog	1,5\$

Create a program that:

Reads the meal the customer wants. Prints the cost of the meal. Input example: "Hot Dog", Output: "Hot Dog 1,50\$"

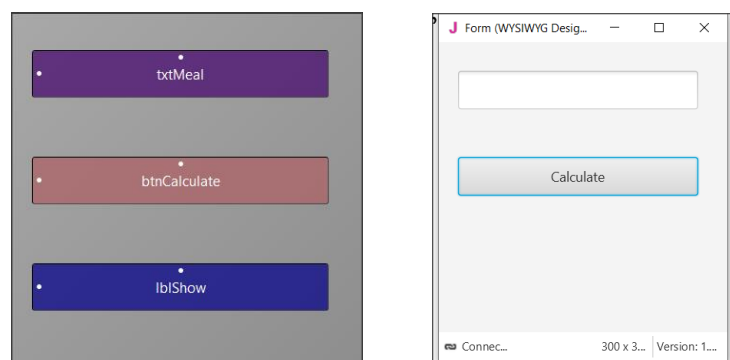
Solution

Step 1

Start a new project and give dimensions 300 x 300.

Step 2

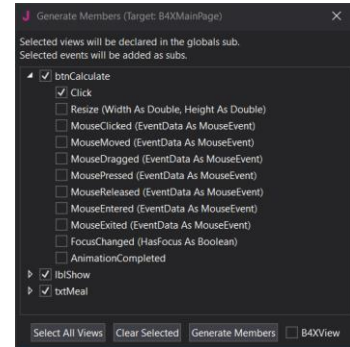
In the designer design the app screen



Picture 14 The app screen

Step 3

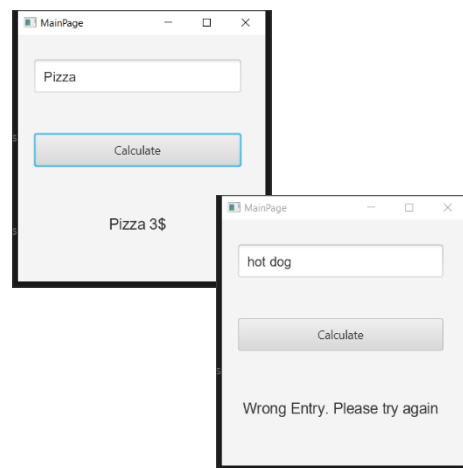
Enter txtMeal , btnCalculate, lblShow , and btnCalculate_Click.



Step 4

The code you need to write is for btnCalculate_Click

```
8 Sub Class_Globals
9     Private Root As B4XView
10    Private xui As XUI
11    Private btnCalculate As Button
12    Private lblShow As Label
13    Private txtMeal As TextField
14 End Sub
15
16 Public Sub Initialize
17 End Sub
18
19 'This event will be called once, before the page becomes visible.
20 Private Sub B4XPage_Created (Root1 As B4XView)
21     Root = Root1
22     Root.LoadLayout("MainPage")
23 End Sub
24
25 Private Sub btnCalculate_Click
26     If txtMeal.Text = "Burger" Then
27         lblShow.Text = "Burger 5$"
28     else if txtMeal.Text = "Pizza" Then
29         lblShow.Text = "Pizza 3$"
30     else if txtMeal.Text = "Hot Dog" Then
31         lblShow.Text = "Hot Dog 1.5$"
32     Else
33         lblShow.Text = "Wrong Entry. Please try again"
34     End If
35 End Sub
```



Note that the code considers capitals to be different from small letters. So he doesn't recognize the hot dog as a meal that should be written as a Hot Dog.

Algorithms with if

Find Maximum Number

Read 3 integers and find the largest in three different ways

Method 1 – Simple If Statement

```
If Inta > intB AND Inta >
intC then
    Log(Inta)
End If
If intB > Inta AND intB >
intC then
    Log(intB)
End If
If intC > Inta AND IntC >
```

Method 2 – Nested If

```
If Inta > intB then
    If Inta > intC then
        Log(Inta)
    End If
End If
If IntB > IntA then
    If IntB > intC then
        Log(IntB)
    End If
End
If IntC > IntA then
    If IntC > IntA then
        Log(IntC)
    End If
End If
```

Method 3 – Max Algorithm

```
Max = inta
If intB > Max then
    MAX = intB
End If
If intC > Max then
    MAX = intC
End If
Log(Max)
```

Exercises

- Make the following suggestions in logical expressions.
 - A belongs to space $[-5, 6)$.
 - a is less than 3 or more than 15.
 - a is equal to b and c.
 - a does not have a value of 3.
 - A is less than 2 or b is greater than 78.
 - a and b true and c false.
 - the a true and one of the b, c true.
- What is the logical result (true or false) of performing the following operations if the following variables have the values below?
A = 10, B = 2, C = -4, D = 9 and E = 1
 - $(A > B)$ or $(D = 10)$
 - $(D \geq B)$ and $(E \neq C)$
 - no $(E \leq C)$ or $(D \leq C)$
 - no $((B \leq C)$ and $(D < 2))$
 - no $(\text{no } (B \leq E) \text{ or not } (C \leq B))$
 - $((E \leq A)$ and $(E \geq C))$ and not $(C \geq A)$
 - no $(\text{no } (A \geq 2) \text{ and } (C \neq 9))$

1. A fast-food chain has these meals:

Meal	Price
Burger	5\$
Pizza	3\$
Hot Dog	1,5\$

Create a program that:

Reads the meal the customer wants and second how many items of this meal needs.

Prints the cost of the meal.

Input example: "Hot Dog", 2

Output: " 2 x Hot Dog 3\$"

2. You have consumed X amount of Mbps on Wikipedia and Y amount of Mbps on memes. The cost of visiting Wikipedia is 0,10\$ per Mb and the cost for watching memes is 0,05\$ per Mb. If total consumption is more than 100\$ print "Too much consumption". If watching meme consumption is greater than reading Wikipedia consumption print "WOW MANY MEMES", "SUCH LOL"(in new line). Create a program that:
- Reads X (Wikipedia Mb consumption) and Y(watching meme Mb consumption)
 - Calculates the total consumption.
 - If total consumption greater than 100\$ print proper message If watching meme consumption is greater than reading Wikipedia articles print proper messages
3. An internet cafe has 2 ways of charging. If the user is a member pays 2\$/hour, Else the user pays 5\$. Find if someone is a member or not and calculate the price based on how many hours the user spend. If the user is a member the tax is 10% else the tax is 20%. Create a program that:
- Reads how many hours the user spend
 - Check if is a member.
 - Add the proper tax fee.
 - Print the total amount the user must pay Output: "The user is a member stayed 2 hours for 2\$/hour plus the 10% the total amount is 4.4\$"
4. You want to buy something from Amazon. The seller charges different prices for shipping cost based on location. For US it's 5\$ for Europe it's 7\$ for Canada it's 3\$ for other places it's 9\$. Create a program that:
- Reads the cost of the product.
 - Reads your location.
 - Print the amount of money you must pay.
 - Output: "You have to pay 23\$, 20\$ for the product and 3\$ for shipping cost".



5. A company sells a product for 0.70 € a piece if up to 200 pieces are ordered and for 0.50 € a piece if more than 200 pieces are ordered. Read the number of pieces ordered and calculate their value.
6. A cell phone company has the following billing policy.

Fixed cost 25\$	
Call duration(in seconds)	Charge(\$/per second)
1-500	0,01
501-800	0,008
801+	0,005

Create a program that:

- Reads how many seconds was the calls duration.
 - Calculates the monthly bill for the subscriber.
 - Prints the total amount.
 - Output: "total amount: 48\$"
 - Notice that that the charge for the first 500 seconds it's 0,01\$ then for the next 501 to 800 seconds it's 0,008 and then it's 0,005\$
7. In the qualifying races in the long jump at the Olympic Games, an athlete makes 3 initial attempts and if he has a performance of more than 7.50 meters, then he is entitled to continue and make another 3 more attempts. Read the first 3 attempts of an athlete and print a message whether he is entitled to continue or not and if he is entitled to find and print the best effort of the athlete.

You can use method Visible = True or False to hide or show labels, textFields and Buttons.

8. In a municipality there are parking spaces for a short period of time. Parking charges are staggered, as shown in the table below:

Parking time	Charge per hour
Up to 1 hour	3.50 €
The next 2 hours	8.00 €
The next 2 hours	12.00€
Over 5 hours	15.00 €

Make a program that reads the duration someone left his car in parking and calculates the total cost.

Lesson 8 – Subroutines

🕒 3h

What students should know

- What is a subroutine.
- Declaring a Sub
- Passing Values
- Returning Values from a sub

In computer programming, a subroutine is a sequence of program instructions that performs a specific task, packaged as a unit. This unit can then be used in programs wherever that task should be performed.

Subroutines may be defined within programs, or separately in libraries that can be used by many programs. In different programming languages, a subroutine may be called a routine, subprogram, function, method, or procedure. In general, to create a subprogram, the developer should keep the following in mind:

The subprogram should only do one task.

Be relatively small and ideally no larger than a screen so that it can be read easily.

Have such a name that refers to its function.

Create a subprogram in B4J

You have Button1_Click already encountered subprogram in B4J that the language has prepared. Notice that events like Button1_Click are also a routine to serve the event.

Example 1

Suppose a function needs to be performed, such as adding two numbers given by the user. As a program it is very simple and generally does not need a subprogram for such a function. Here we will use a subprogram to understand how it is used and operated.

The two integers intA and intB have been declared in the program, values have been assigned, and then the showSum1 routine is called.

```
8 Sub Class_Globals
9     Private Root As B4XView
10    Private xui As XUI
11 End Sub
12
13 Public Sub Initialize
14
15 End Sub
16
17 'This event will be called once, before the page becomes visible.
18 Private Sub B4XPage_Created (Root1 As B4XView)
19     Root = Root1
20     Root.LoadLayout("MainPage")
21 End Sub
22
23 'You can see the list of page related events in the B4XPagesManager
24
25 Private Sub Button1_Click
26     xui.MsgboxAsync("Hello world!", "B4X")
27 End Sub
```

Picture 15 Subprograms

This is done by writing the name of the routine (which the developer decides) and then in parentheses the variables whose values it must know to function.

The subprogram is written before or after the current subprogram:

```
Private Sub B4XPage_Created (Root1 As B4XView)
    Root = Root1
    Root.LoadLayout("MainPage")
    Private intA, intB As Int
    intA = 10
    intB = 30

    showSum1(intA, intB)
End Sub

Private Sub showSum1(a As Int, b As Int)
    Log(a+b)
End Sub
```

Picture 16 Calling a subprogram

It always starts with the **Private** statement which means that it is a subprogram that will be known only in the code (B4XmainPage) or **Public** for the subprogram to be known in other parts of our application.

The Sub statement which means subroutine and

In parentheses, the names of the variables that will receive the data from the call point are indicated.

Notice in the image that this data enters in the order written when calling the routine, i.e. the value of the intA will enter in variable a and the value of the variable intB will enter in variable b.



Remember

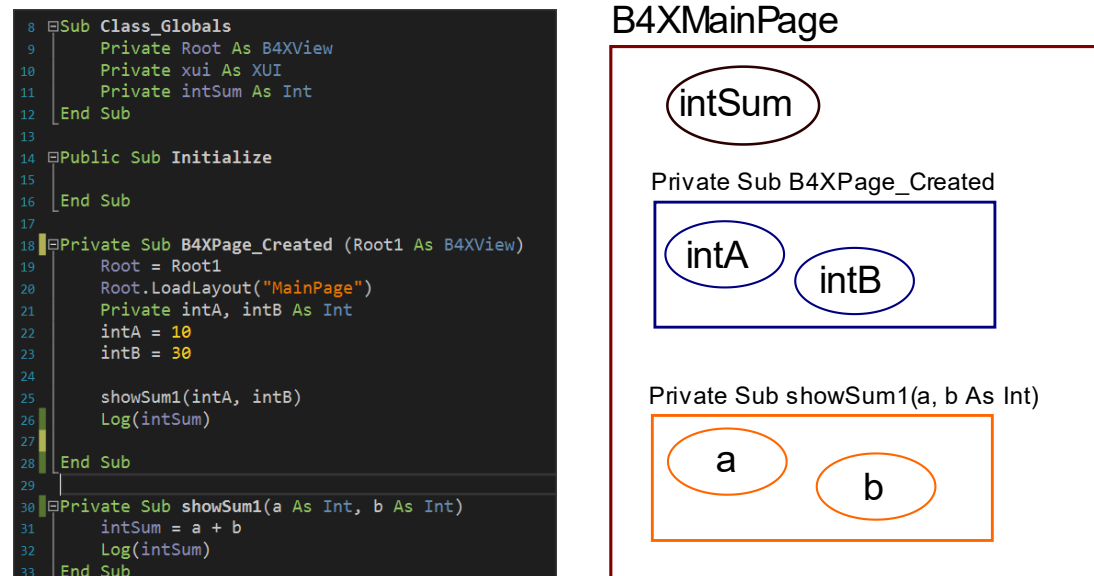
Variables exchanged between subprograms during their call are called **parameters**.

The subprogram now works with data contained within parameters a and b not the intA and IntB.



The memory of the subprogram in B4X

Each subprogram has its own memory space to store their variables. An exception to this rule is the **Class_Globals** subprogram, whose data is known to all B4XMainPage routines and can be reported to them by name.



Picture 17 The Memory

From the image code you can notice that the variable **intSum** is known in both subprograms and is used simply by writing its name. In B4X these variables are displayed in a different color so that the developer can distinguish easily. On the contrary, variables declared within the remaining variables live only within them and another subprogram cannot use them by name.

Remember



Variables declared within **Class_Globals** are known in all subprograms and are called **Global**.

The variables declared within the subprograms are called **local** and are not known in the rest of the program.

Return a value from a subprogram.

A subprogram can return a value to the code that calls it. This is done through the subprogram name itself as follows:

```
19 Private Sub B4XPage_Created (Root1 As B4XView)
20     Root = Root1
21     Root.LoadLayout("MainPage")
22     Private intA, intB As Int
23     intA = 10
24     intB = 30
25
26     showSum1(intA, intB)
27     Log(intSum)
28
29     Log(Sum(intA, intB))
30 End Sub
31
32 Private Sub showSum1(a As Int, b As Int)
33     intSum = a + b
34     Log(intSum)
35 End Sub
36
37
38 Private Sub Sum(a As Int, b As Int) As Int
39     Return a+b
40 End Sub
```

The program must be declared as a variable type.

In addition, the return command must be used within the subprogram to return the calculated value.

Finally, the code that called the subprogram accepts back the value calculated and can use it like any variable.

Picture 18 Returning Values back to program.



Remember

Often routines that return values to the code that calls them **are** also called Functions.

Example 2

Write a subprogram that accepts 3 integer variables and returns the largest value.

```
6 Sub Class Globals
7     Private Root As B4XView
8     Private xui As XUI
9 End Sub
10
11 Public Sub Initialize
12 End Sub
13
14 Private Sub B4XPage_Created (Root1 As B4XView)
15     Root = Root1
16     Root.LoadLayout("MainPage")
17     Private intA, intB, intC As Int
18     intA = 20
19     intB = 10
20     intC = 300
21
22     Log(sMax(intA, intB, intC)) ' Shows 300
23 End Sub
24
25 Private Sub sMax(a As Int, b As Int, c As Int) As Int
26     Private intM As Int
27     intM = a
28     If b > intM Then
29         intM = b
30     End If
31     If c > intM Then
32         intM = c
33     End If
34     Return(intM)
35 End Sub
```

Three integer variables (intA, intB, intC) are declared within subprogram B4XPage_Created.

Values are assigned to the three variables.

The sMax subprogram is called with parameters the three variables.

The subprogram applies the MAX algorithm for the three variables a, b, c and returns the intM value calculated.

Finally, the highest value appears on the Log screen.

Picture 19 Example 2





Remember

We love subprograms because:

- It is boring to always write the same code.
- It is faster than anyone can type.
- It is helping not to repeat the same mistakes. You have done it once!
- Cool programmers write subprograms.

Exercises



Teachers tip

Encourage students to solve all exercises and discuss the solution in class.
Dedicate at least 1 hour to explain them.

1. Write a program that calculates the area of a circle. The user must enter the radius of a circle in an appropriate textField, and then using a subprogram to calculate and return the area.
2. Write a program that calculates the solution of the secondary equation $ax^2 + bx + c = 0$.
 - a. The user must enter the equation coefficients in appropriate TextFields.
 - b. The result appears in one, or two textField depending on the value of the Discriminant.
 - c. The Discriminant must be calculated with a subprogram that returns its value.
 - d. It will not be permissible to calculate the Discriminant factor unless all the fields of factors a, b, c are filled in.

Note to display the error message use
`xui.MsgboxAsync("Message","Title")`

a	<input type="text"/>	b	<input type="text"/>
b	<input type="text"/>	x1	<input type="text"/>
c	<input type="text"/>	x2	<input type="text"/>
<input type="button" value="Calculate"/>			

Picture 21 Wireframe


$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

The [quadratic formula](#) for the roots of the general quadratic equation

Picture 20 Source:
[Wikipedia.org](https://en.wikipedia.org/wiki/Quadratic_formula)

3. Make a program that uses the turtle and draws squares with a side given by the user. Create a subprogram that accepts the side and then draws the square starting from where the turtle is already located and moving clockwise.



- 
4. One theatre has three categories of tickets, Arena, Gallery, Proscenium. Each ticket costs 20, 30, 40€. Make a program that:
 - a. Reads the code 1, 2, 3 representing respectively the categories.
 - b. Reads the number of seats he wants.
 - c. Calculate the value of tickets with a subprogram that returns the amount, and which will be displayed in an appropriate textField



Lesson 9 – Classes



What students should know

- What is a Class?
- What is an Object?
- What are Attributes and Methods?
- Create and use simple class with B4J.

Often in programming there is the need to describe similar items in a single way. For example, pupils of a school. Each student is known to have a full name, home address, class to attend, some grades, etc. To monitor and manage this information, a programmer must organize them with systematic way.



Teachers tip

Classes are a tricky subject of programming. Avoid many details and you do not need to go into issues such as inheritance, encapsulation, etc.

Classes

In the previous example with students, we could say that each student has the following information.

Student	
1	Registry Number
2	Name
3	Surname
4	Address
5	Phone
6	Email

At the same time, we need functions such as:

- New Student Registration
- Change Student Information
- Student Transfer
- Show Student Information
- Delete Student

Thus, for three students we could have the following elements:

Student 1	
1	125310
2	Augusta
3	Ada Byron
4	London
5	37535795
6	ada@lon.uk

Student 2	
1	125311
2	Maria
3	Curie
4	Warszawa
5	678433
6	maria@wars.pol

Student 3	
1	125312
2	Muhammad
3	al-Khwarizmi
4	Khwarazm
5	646456456
6	algor@khwa.pe



From the above we conclude that essentially for students we have similar data and similar actions that we can apply to them. Grouping all student data and functions into an independent and single code is called class. Every student Called **Object** or **Instance** of the class. Also, the variables that characterize the student surname, Registry number etc. Called **Properties** and the functions Described **Methods**.



Remember

We call **class** the grouping of data and functions into a single and independent code.

Object or **Instance** of the class are all independent elements that result from the use of the class.

Variables for an object are called **properties**.

The functions applied to an object are called **methods**.

Some of the advantages of using classes are flexibility in the use of code, speed and ease in developing programs and reusing code in other programs.

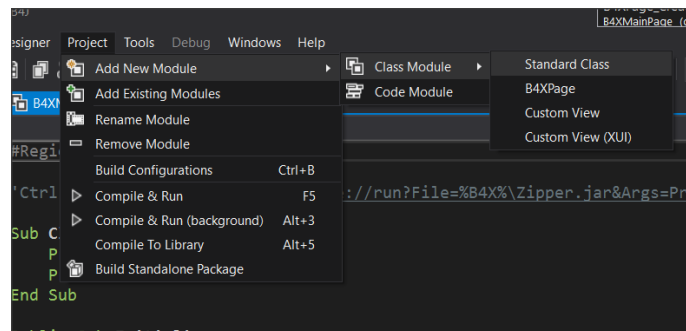
Example of class in B4J

A library has a set of books which it lends to readers who subscribed in the library. Each book has features such as title, author, publisher, year of publication. Also, the Insert Book, Show Book Items, Change Book Items functions apply to each book.

Create an application in B4J that implements the Book class with the properties and methods mentioned above.


Implementation methodology

1. Create a new application B4XPages and give the name "library".
2. From menu Project – Add New Module – Class Module menu select Standard Class.



3. In the dialog box, name clsBook (i.e. class Book)
4. A new code tab named clsBook will be created.





```
Main B4XMainPage clsBook X
insertBook
1 Sub Class_Globals
2   Private fx As JFX
3
4
5 End Sub
6
7 'Initializes the object. You can add parameters to this method if needed.
8 Public Sub Initialize
9
10 End Sub
11
```

5. Within the Class_Globals routine, add all the variables that will be the properties of the class.
 - a. Book Title
 - b. Author Name
 - c. Publisher
 - d. Release Date

```
1 Sub Class_Globals
2   Private fx As JFX
3   Public strWriter, strTitle, strYear, strPublisher As String
4 End Sub
```

6. Finally, you must implement the subprograms that will implement the methods:
 - a. Insert a book,
 - b. Show Book Items,
 - c. Change Book Items



Teachers tip

Be careful to explain that all the variables and methods you create are public so that objects can use them.

Insert a book.

This is a subprogram that accepts 4 string data as parameters and then assigns it in the order that it is inserted into the subprogram into the variables strTitle, strWiter, strYear, strPublisher.

```
14 Public Sub insertBook(str1, str2, str3, str4 As String)
15   strTitle = str1
16   strWriter = str2
17   strYear = str3
18   strPublisher = str4
19 End Sub
```

Show Book Items.

The subprogram displays with log command the properties of the Books class or in other words the variables that describe the class.

```
21 Public Sub logBook
22     Log("Title : " & strTitle)
23     Log("Writer: " & strWriter)
24     Log("Year  : " & strYear)
25     Log("Publisher: " & strPublisher)
26 End Sub
```

Change book items.

The method of changing items accepts as parameters new elements for the class and changes the property values of the corresponding properties.

```
28 Public Sub changeBook(str1, str2, str3, str4 As String)
29     strTitle = str1
30     strWriter = str2
31     strYear = str3
32     strPublisher = str4
33 End Sub
```

The Initialize subprogram.

The Initialize routine is used to give initial values to variables or do any other functions required when creating an object from a class.

```
7 Public Sub Initialize
8     strTitle = ""
9     strWriter = ""
10    strYear = ""
11    strPublisher = ""
12 End Sub
```

Use Class

Back on tab B4XMainPage it's time to use the clsBook class.

1. First create clsBook objects.

```
8 Sub Class_Globals
9     Private Root As B4XView
10    Private xui As XUI
11    Private book1 As clsBook
12    Private book2 As clsBook
13 End Sub
```

where book1, book2 are two clsBook objects with all the properties and methods discussed previously.

Use of methods

```
20 Private Sub B4XPage_Created (Root1 As B4XView)
21     Root = Root1
22     Root.LoadLayout("MainPage")
23
24     book1.Initialize
25     book2.Initialize
26
27     book1.insertBook("Neuromancer", "William Gibson", "1984", "Ace")
28     book2.insertBook("2001: A Space Odyssey", "Arthur C. Clarke", "1968", "Ace")
29
30     book1.logBook
31     book2.logBook
32 End Sub
```

The first method to be used in objects is the initialize method.



Remember

Initialize is not optional method. It's the first method you must use before do anything with an object

The InsertBook method then enters values for the two books in the object properties.

End the logbook method displays the contents of the properties of each book.

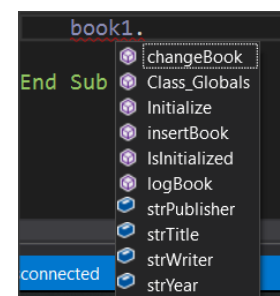
```
Title : Neuromancer
Writer: William Gibson
Year : 1984
Publisher: Ace
Title : 2001: A Space Odyssey
Writer: Arthur C. Clarke
Year : 1968
Publisher: Ace
```



Remember

Each property can be called by writing the object name, then a period and the method name. Some methods need parameters to work while others don't.

Each time you write an object name and pressing the period B4X displays a window with all available properties and methods of the class. The IsInitialized method checks whether the object is initialized and exists in all classes that you create.



Exercises

1. In the first example of the course, implement in B4J the Student class with properties:
 - Registry Number
 - Name
 - Surname
 - Class
 - Phone
 - Email





and methods

- New Student
- Show Student
- Change Class
- Change Phone

2. If only one teacher teaches a specific course in a school implement the Course class with properties

- Lesson
- Class
- Hours
- Professor

and methods

- New Lesson
- Change Hours
- Change Professor
- Show Lesson

3. A store has computers for sale. For each computer are recorded:

- The type (desktop, laptop),
- the model,
- its price,
- Its cpu (I3, i5, i7, i9)

Create a class that implements a computer with the above properties and methods that you will create. Be careful to check that the values entered in the type and cpu are those in the parenthesis.

Lesson 10 – B4XPages

🕒 3h

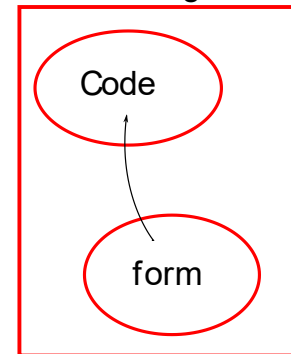
What students should know

- What is a B4XPage
- How to Create and Delete a B4XPage
- Passing Values within Pages

B4XPages is a software library. Includes classes and procedures to create multiple application communication forms with the user. Also, it helps to transfer applications to different platforms using the B4A, B4i and B4J tools.

Every application you have created so far with B4J already includes a B4XPage. This is B4XMainPage which is always the user's first contact form with the application. More generally, each B4XPage manages all the codes required for the GUI (Graphics User Interface) to work.

B4XMainPage



The structure of an application's folders

When you start a new program with B4XPage, the following folder structure is created.

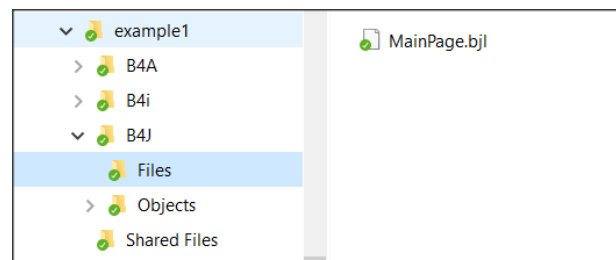


Figure 1 example1 folders

Each of the three folders B4A, B4i And B4J include the relevant codes in order to create applications, Android, Ios computers (Windows, Linux platforms) respectively.

Specifically, in the **B4J** folder there is the "Files" folder where

it contains all the files created with the Designer as well as any other files that must be used during the execution of the code e.g. images. **MainPage.bjl** is the file that was created automatically during the creation of the application and is its home screen. The Shared Files folder also includes files that the three different applications can share if the developer **chooses** to create an application for both Android, IOS and PC.

The application's home folder contains all the files that create the different B4XPages of our application.

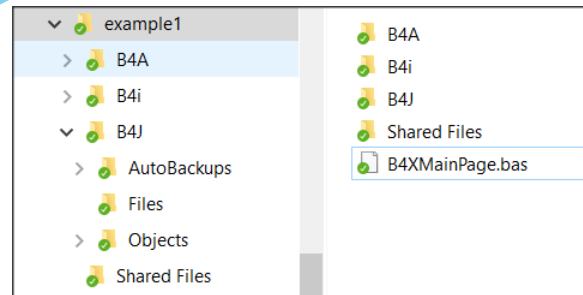


Figure 2 B4XPage files

The first page that is created must have the name **"B4XMainPage.bas"** and cannot be changed; all other pages are named by the developer.

Starting an application with B4XPage.

When creating a new application with B4Xpage, the language has already prepared the first page and as mentioned its name in the application folder is B4XMainPage.bas. Also, it has created a form (or GUI screen) to communicate with the user (named MainPage.bjl). Eventually, a mechanism called B4XPagesManager undertakes to manage the pages.

```
Sub Class_Globals
    Private Root As B4XView
    Private xui As XUI
End Sub

Public Sub Initialize
End Sub

Private Sub B4XPage_Created (Root1 As B4XView)
    Root = Root1
    Root. LoadLayout ("MainPage")
End Sub
```

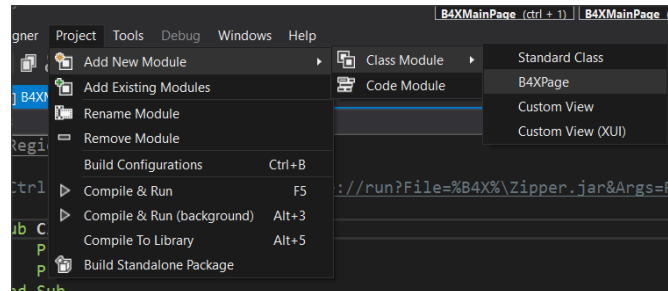
What is Root

The Root variable is an object of class B4XView. Undertakes to perform all display-related tasks in the various forms created by the developer (also associated with code sharing in B4J, B4A, B4i). Therefore, the Root object instructs the MainPage form to load with the Root.LoadLayout("MainPage" method).

Create a new B4XPage

Step 1.

After Create an app select from the menu Project – Add New Module – Class Module – B4XPage And Give the name B4XPage1.



Picture 22 Create B4XPage

A class with a name "B4XPage1" will be created, and some necessary codes to get started. The user communication screen (GUI) has not yet been created at this point. This should be done by the Designer later.

```
Sub Class_Globals
    Private Root As B4XView 'ignore
    Private xui As XUI 'ignore
End Sub

'You can add more parameters here.
Public Sub Initialize As Object
    Return Me
End Sub

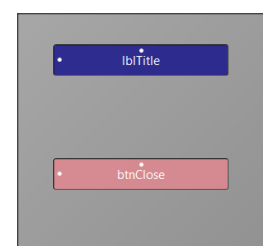
Private Sub B4XPage_Created (Root1 As B4XView)
    Root = Root1
    'load the layout to Root
End Sub
```

Picture 23 The new B4XPage

Step 2.

Open Designer and from the menu File select New.

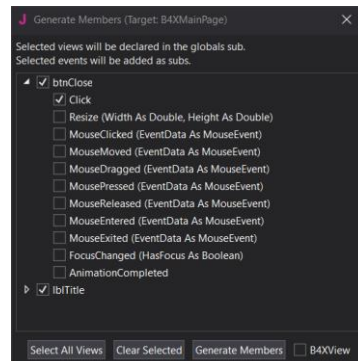
From the Variants tab, specify the dimensions of the form you want to design, and then select a label and button to insert into your form as in the Picture 3.



Picture 24 Form

Step 3.

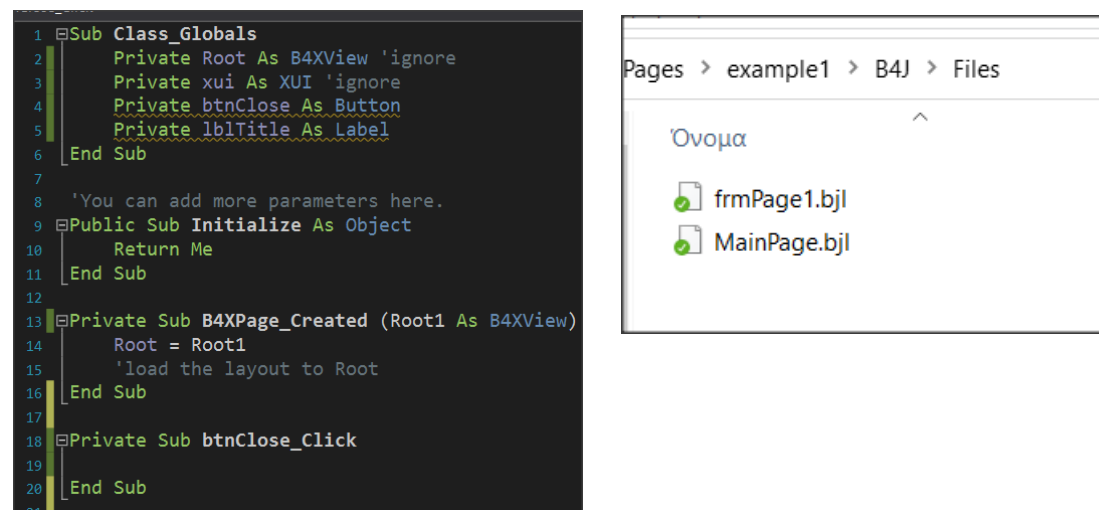
Use the menu Generate Members to insert the two objects into your code as well as the Click of the button. Remember that this action must be applied when you are in the code of the B4XPage1.



Picture 25 Generate Members

From the file menu save your form named frmPage1. (You can give any name you want, and it serves the needs of the application).

The following code (Picture 5) will now be created, and the file will have been displayed frmPage1.bjl in the folder files.



Picture 26 frmPage1

Step 4.

To link the form frmPage1 with the B4XPage1 you must now call the property `Root.LoadLayout("frmPage1")` within the event `B4XPage_Created`.

```
Private Sub B4XPage_Created (Root1 As B4XView)
    Root = Root1
    'load the layout to Root
    Root.LoadLayout("frmPage1")
End Sub
```

The next steps include method's programming and depend on the purpose of the application you are building.

In our example, suppose you want to move to the B4XPage1 screen with one click in B4XmainPage's button and then click the button you placed to return to the home page.

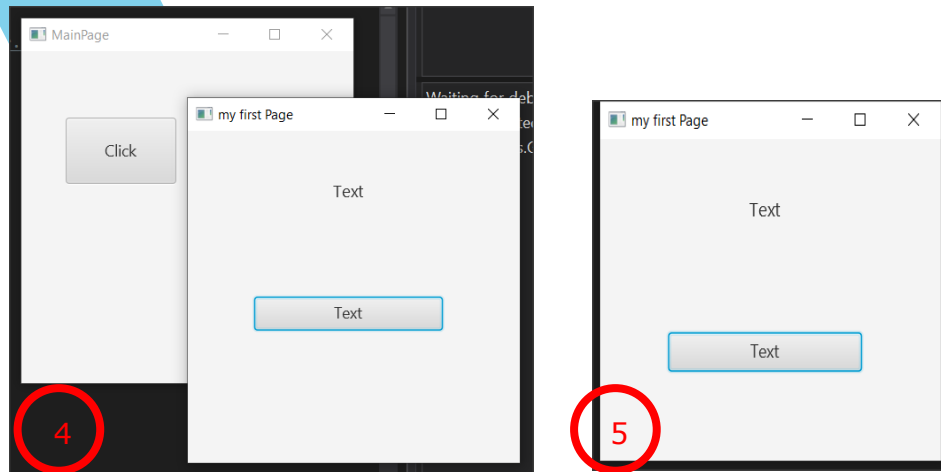
Call a new B4Xpage.

Each new B4XPage you create is a class. Therefore, before you can use it, you must create an object based on it. The creation is usually done in the B4XPage that will call the new one. So, in the previous example we write in B4XmainPage (Picture 6):

```
1 Sub Class_Globals
2     Private Root As B4XView
3     Private xui As XUI
4     Private page1 As B4XPage1 ①
5 End Sub
6
7 Public Sub Initialize
8 End Sub
9
10 Private Sub B4XPage_Created (Root1 As B4XView)
11     Root = Root1
12     Root.LoadLayout("MainPage")
13     page1.Initialize ②
14     B4XPages.AddPage("my first Page", page1) ③
15 End Sub
16
17 Private Sub Button1_Click
18     ④ B4XPages.ShowPage("my first Page")
19     ⑤ 'B4XPages.ShowPageAndRemovePreviousPages("my first Page")
20 End Sub
```

Picture 27 Create B4XPage

1. Set a class object B4XPage1.
2. Initialize page 1. Runs the Initialize routine within class B4XPage1.
3. Create an ID for the new page object. (In the example is "my first Page")
4. Call the new page while the home page remains open.
5. Second way to call a page where the home screen closes and only the new page remains open.



Picture 28 Two methods to open a B4Xpage

Close a B4XPage.

When a B4XPage is called, the program control passes to that page. Therefore, for the page to close, an event must occur, such as pressing a button or the close button in the top right of the window. More generally, it also depends on how the screen was opened:

When opened with:	Usually closes with:
<code>B4XPages.ShowPage("my first Page")</code>	<code>B4XPages.ClosePage(Me)</code>
<code>B4XPages.ShowPageAndRemovePreviousPages("my first Page")</code>	<code>B4XPages.ShowPageAndRemovePreviousPages("MainPage")</code>

The first way closes the current form while the second way essentially calls the home page again while closing the current one.

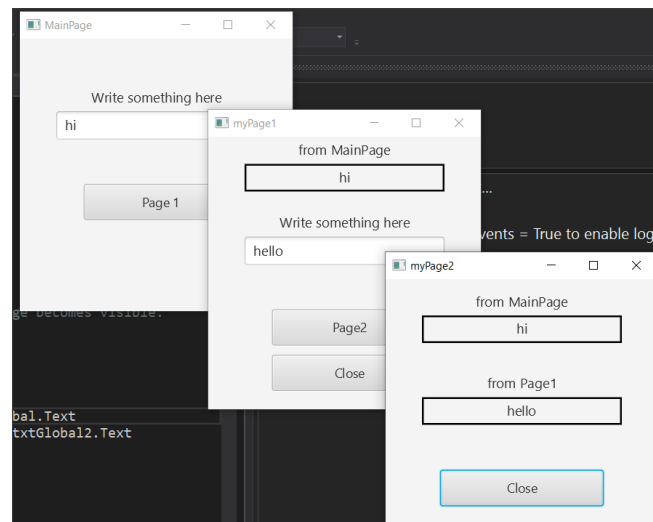
Transfer information between pages

For one page to have access to data from another, those pages must have variables declared with the keyword `Public`. The pages objects themselves when they are created either in the `MainPage` or elsewhere have also been declared as `Public`.



Example 2

For the purposes of this example, you will use the application of example 2. This includes `MainPage`, `B4XPage1` and `B4XPage2`. Forms have also been created in the Designer. Open example 2, run it, and observe its operation.



Picture 29 Example 2

As you run the application, notice that text from textFields is transferred to the following pages. This is because both `page1` and `page2` and both `TextField` are declared `public`.

```
Sub Class_Globals
    Private Root As B4XView
    Private xui As XUI
    Public page1 As B4XPage1
    Public page2 As B4XPage2
    Public txtGlobal As TextField
End Sub
```

Picture 30 Public declarations in MainPage and Page1

In order `page1` to have access to the `txtGlobal1` variable, it must also use it by indicating the name of the page on which it was created:

```
lblGlobal1.Text = B4XPages.MainPage.txtGlobal.Text
```

where `lblGlobal1` is a label that displays the content read on the `page1` screen.

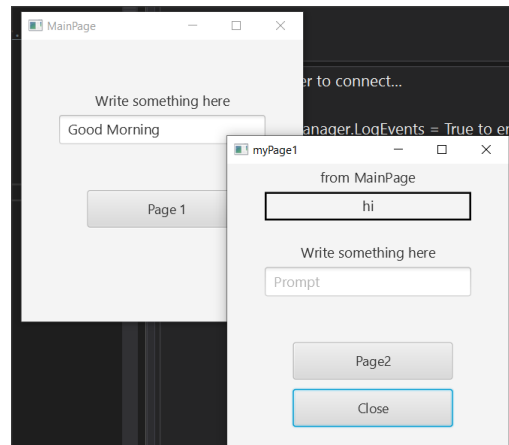
Similarly, `Page2` has access to `MainPage's txtGlobal1` and `Page1's txtGlobal2` variables as follows:

```
lblGlobal1.Text = B4XPages.MainPage.txtGlobal.Text
lblGlobal2.Text = B4XPages.MainPage.page1.txtGlobal2.Text
```

where lblGlobal1 and lblGlobal2 are two labels that display the contents of the two Public Variables on the page2 screen.

The Life of B4XPages

In the previous example, try closing all windows except MainPage and type in new text and press the button to page1. You will notice that the value displayed by the MainPage it is not the new but still shows the first one.



Picture 31 B4XPage life

This is because B4Xpages remain in computer memory and each time they are called the event B4XPage_Create does not run again. To read the global variables again from MainPage you can run event **B4XPage_Appear** and in there use the variables from MainPage:

```
Private Sub B4XPage_Appear
    lblGlobal1.Text = B4XPages.MainPage.txtGlobal.Text
End Sub
```

Unlike B4XPage_Create that runs only once on the first call of the page, B4XPage_Appear runs every time the window appears in the foreground, so you can use it whenever the page returns to transfer variables from other forms.

Exercises

1. The little encyclopedia of dogs. Create an application where three different breeds of dogs are displayed on a home page and after the user clicks on the corresponding name display information about the breed along with two photos.

You can use TextArea in designer to make bigger text areas with scroll bar.

2. Build an application that solves:
 - a. the primary equation $ax+b=0$,
 - b. the secondary equation $ax^2+bx+c=0$ and
 - c. calculates the hypotenuse of a triangle given the dimensions of the two vertical sides.

It is given that the square root of an x number is calculated by \sqrt{x} .

3. Build an app that creates a virtual store as follows: The home screen will show images of 4 different objects, such as laptops and a TextField for each item where customer writes the quantity. Then pressing the Buy button, the program on a new page displays the Total Value and quantity of items selected. *Except MainPage you will need only one more.*

