



Anywhere Software

Programming with B4J

A guide for High School Teachers
and Students

Leon Prokops

Version 2.0, July 2021





by Leon Prokopis, 2021, Nafplio, Greece
This work is licensed under a [Creative Commons Attribution 4.0 International License](#).

Table of Contents

Table of Contents	3
Lesson 1 – Why B4X?.....	9
Installing B4X	9
Lesson 2 – The meaning of the problem	11
The concept of the problem	11
Understanding the problem.....	11
Searching for a solution or set of solutions.....	11
Choosing the right solution	12
Implementing the solution.....	12
Checking whether this solution had the desired results.....	12
Lesson 3 - My first Program	13
Hello World.....	13
Recommended instructions for the instructor.....	13
Exercises.....	14
Lesson 4 - Variables and Range	15
How to find how many variables you need	16
Naming Variables.....	17
Declaring Variables	17
My First Variable	17
Comments.....	18
The log area and the Log() function.....	18
Mathematical Operators.....	19
Strings.....	20
Exercises.....	21
Lesson 5 – Designer.....	25
First steps with design	26
Design a wireframe	27
Visual designer.....	28
The Views Tree	28
Properties	29
Abstract Designer	29



Example 1	29
Decide on the size of the app screen.	29
Set an appropriate screen variant.	30
Create the views.	30
Excercises	32
Lesson 6 – From Designer to Code	35
Class_Globals.....	35
A deeper look at the use of variables.....	36
Passing Values to Code	36
Events	37
Writing code in Event	38
Properties	39
Exercises.....	40
Lesson 7 – Conditional Statements	41
Logical Variables.....	41
Comparative Operators	41
Logical Operators.....	42
Logical operators in programming	43
Examples of evaluation of logical sentences.....	43
If statement.....	44
If – Else	45
If – else - else if	46
Algorithms with if.....	48
Exercises.....	48
Lesson 8 – Subprograms	51
Create a subprogram in B4J.....	51
Example 1.....	51
The memory of the subprogram in B4X	53
Return a value from a subprogram.	54
Example 2	54
Exercises.....	55
Lesson 9 – Classes.....	57
Classes	57
Example of classes in B4J.....	58
Implementation methodology.....	58
The Initialize subprogram.	59

Insert a book.....	59
Show Book Items.....	60
Change book items.....	60
Use Class.....	60
Use of methods	61
Getters and Setters.....	61
Exercises.....	63
Lesson 10 – B4XPages	65
The structure of an application's folders	65
Starting an application with B4XPage.....	66
What is Root	66
Create a new B4XPage	67
Call a new B4Xpage.....	69
Close a B4XPage.....	70
Transfer information between pages	71
The Life of B4XPages.....	72
Exercises.....	73
Lesson 11 – First Project.....	75
Mobile Phones shop application.....	75
Lesson 12 – Loops	77
The Do While command	77
Examples of the command Do While.....	78
Loops with unknown repeats.....	81
The Do Until command.....	82
For Loop.....	85
Examples of For loops	86
Exercises.....	87
Lesson 13 – XUI Views	89
Libraries in B4X.....	89
The XUI Views library	90
Using XUI Views	91
Scrolling Label	91
B4XFloatTextField	91
RoundSlider	92
AnotherProgressBar.....	92
B4XImageView	93



B4XDialogs	94
MsgBoxAsync	95
Msgbox2Async.....	95
Wait For	96
CustomDialog.....	96
Templates	97
B4XDateTemplate	97
B4XColorTemplate.....	98
B4XLongTextTemplate	99
Exercises.....	100
Lesson 14 – Arrays	101
Declaring Arrays	101
Functions in arrays.....	102
Insert items into an array.....	102
Display items in a array.	103
Show items in reverse order	103
Find Total and Average Array Items	103
Find Maximum and Minimum.....	103
Search Algorithms.....	104
Serial Search.....	104
Binary Search.....	104
Sort	105
Exercises.....	108
Lesson 15 – Lists	109
Create a list	109
Insert an item in a specific location.....	110
Remove an item from a specific location	110
Change position value	110
More commands in lists	111
Exercises.....	112
Lesson 16 – Maps	115
Create a map	116
Insert items into Map	116
Use a map value.....	117
Indexes in Maps	117
The command “for each”.....	118

Check key existence.....	118
Delete Key and Empty Map.....	118
Exercises.....	119
Lesson 17 – Files.....	121
Storage Folders	121
File.DirAssets	121
xui.DefaultFolder	121
File.DirData.....	122
File.DirApp.....	122
File.DirTemp	122
Create folders	122
Check file existence	122
Create and write to a file.....	123
Writing and reading data.....	123
String-type variables	123
Lists.....	123
Maps.....	124
Exercises.....	125
Lesson 18 – Complex Data Types, Files & Views	127
The type statements	127
Show Items	128
Array of a type	128
List of Type.....	129
Maps of Type	129
Save to KVS files	130
Declare a KVS file	130
Initialize the KVS file	130
Insert items into a KVS file	131
Retrieving items from a KVS file	131
Check the existence of a record.....	132
Delete KVS file content	132
More Views	133
CustomListView	133
B4XComboBox.....	135
Exercises.....	137
Lesson 19 – Final Project	139



Create a School Library Application	139
Files	140
Data	141
Types	141
Screen Design and Functions.	142
Cheat Sheet.....	144
Lesson 20 – From B4J to B4A	147
Congratulations!	147
Application description in B4J.....	147
Designer.....	147
Transfer the app to B4A and Android.....	150
Transfer of design.....	150
Install an app on your mobile phone.	151
Appendix 1	153
Timetable.....	154
Wireframe Sketching.....	155
Classes	156
B4XPages.....	157

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>



Anywhere Software

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 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 else seeking a solution to one. 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 identifying 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

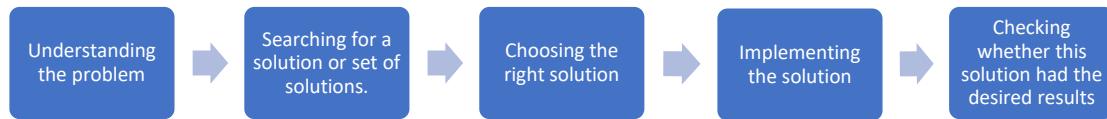
Often a problem can have more than one solution or some of them may be more efficient than others. Choosing the solution correctly leads to a shorter and more reliable outcome.

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 with the creation of a project using B4J and to become the first acquaintance with the programming environment. The following instructions in no way limit the instructor when 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	<p>Menu File -> New -> B4Xturtl Project Folder Project Name</p> <p>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</p>
2 Run Project	<p>Menu Project -> Compile & Run</p> <p>Explain what compile means and how to recognize syntax errors in the log. You don't need to provide too much information about compilation. Just the basic stuff in order to run a project.</p>
3 #Region Project Attributes	<p>Change Values in <i>MainFormWidth</i> and <i>MainFormHeight</i> to make different size application</p>
4 Sub Turtle_Start	<p><i>What does Sub mean?</i></p> <p>A small amount of code which is doing a certain activity.</p>

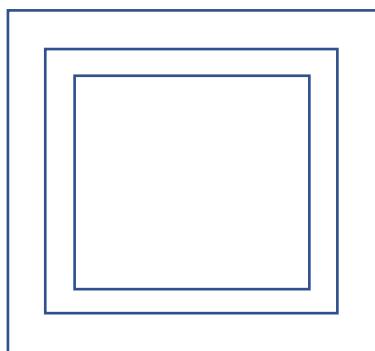


Anywhere Software

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

Exercises

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



3. 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 a memory address, and the "content" of that memory location is the "resident". On the computer often the "resident" who from now on we will call variable needs more houses, or more than one memory location, to fit into.

For a programmer to use memory, he must know the data he needs and the type of that data. These can be integers or real numbers, words or letters, some logical values (truth, false). He also needs a "home" in the computer's memory to store these data 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 sized space in memory to store values.

Because it is difficult for the developer to remember all the addresses of his data, each address can be given a human recognizable name. In fact, all programming languages recognize this need, 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's 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 identify data and the related information for 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 by thinking of the exercise in terms of keywords such as:

- Reads
- Registers
- Ask (Inputs)
- Accept
- Type (data types)

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 using keywords such as:

- Calculates
- Displays
- Writes
- Counts
- Convert

In the previous examples what are the requested inputs and outputs?

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 underscore character.
- B4X does not distinguish capital and small letters (referred to as case-insensitive).

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



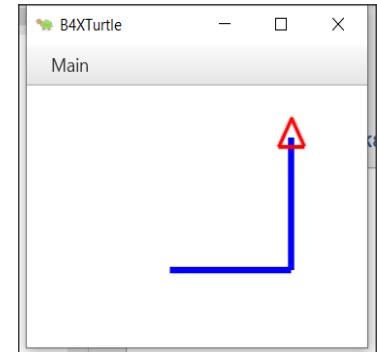
Make a program that assigns the type integer to a value and then draws with the help of the turtle a line of length as long as the value in the variable.

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 knows of its existence and thus the value it contains.

Alternatively, a variable statement that starts with the keyword Public can be known and accessed by programs or subprograms or classes etc.



The name given to a variable appears after the keyword Public or Private. 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 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 Interpreter of the language. Generally, in B4X comments should be put anywhere where 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 interpreter and indicate them on the logs panel. In order to access the logs panel we need to click on the relevant logs tab at the bottom right. The Logs panel itself is divided into two frames, the first of which displays errors and

the bottom frame displays language messages or that information we want to display using the log() function.

Using the Log() function helps the developer display messages in the log

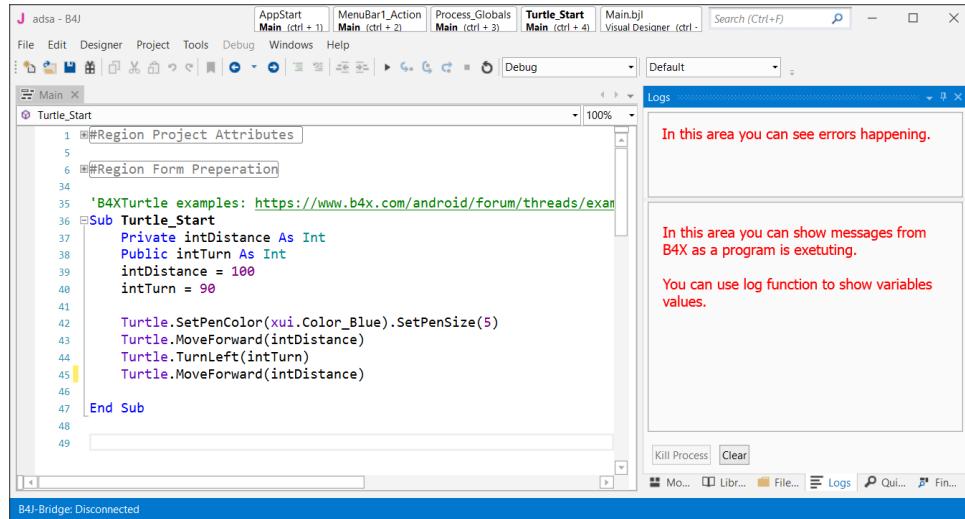


Figure 2 Logs Screen

panel while running a program as well as variable values to help ensure the program's proper operation.

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

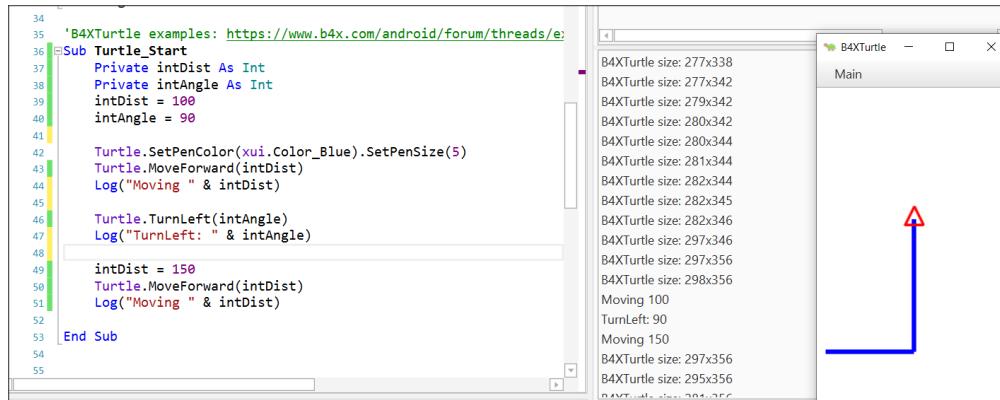


Figure 3 Using log function

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 Mod y	Modulo
Power	Power(x,y) 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           'Shows 90
Log(intS)

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

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 a 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 concatenate 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

2. 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 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 resistors R₁ and R₂ connected in series is R₁ + R₂ and parallel (R₁*R₂)/(R₁+R₂) respectively. Make a program that reads two resistance values 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:

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:
- Assigns the value of the bitcoin at the time of purchase.
 - Assigns 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:
- Assigns 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:
- Assigns the the price of the laptop in a variable.
 - Assigns 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:
- Assigns the number of years employed in a variable
 - Assigns the number of children the employee has in second variable.
 - Calculates and logs the total amount of salary the employee makes.
8. Create a program that logs 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 IDE 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 IDE 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 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 simply be functional and easy to use but information must be presented in an organized way without being confusing.

Before designing any app remember some key design elements (usabilty.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.

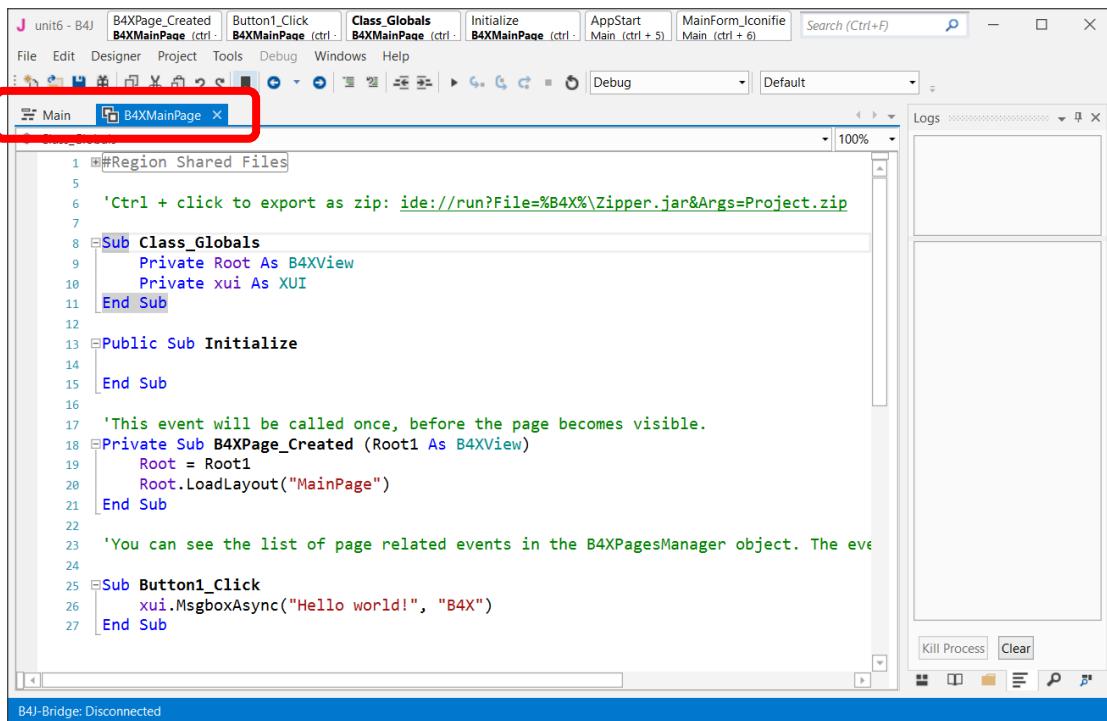
Think about the defaults. By carefully thinking about and anticipating the goals people bring to your site, you can create defaults that reduce the burden on the user. This becomes particularly important when it comes to



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 start the B4J IDE and now from file menu choose **New** and **B4XPages**. Choose a directory and enter a name for your project. You will see the code below. There are two tabs here, the first one called **Main** and the second **B4XMainPage**.



```
J unit6 - B4J B4XPage_Created [ctrl] B4XMainPage_Click [ctrl] Class_Globals [ctrl] Initialize [ctrl] B4XMainPage [ctrl] AppStart [ctrl] Main [ctrl + 5] MainForm_Iconifie [ctrl] Main [ctrl + 6] Search (Ctrl+F) 
File Edit Designer Project Tools Debug Windows Help 
Main B4XMainPage X 
Logs 
1 ##Region Shared Files 
5 
6 'Ctrl + click to export as zip: ide://run?File=%B4X%\Zipper.jar&Args=Project.zip 
7 
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 object. The eve 
24 
25 Sub Button1_Click 
26     xui.MsgboxAsync("Hello world!", "B4X") 
27 End Sub 
B4J-Bridge: Disconnected
```

Do not worry about them now. We will discuss them later. 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.

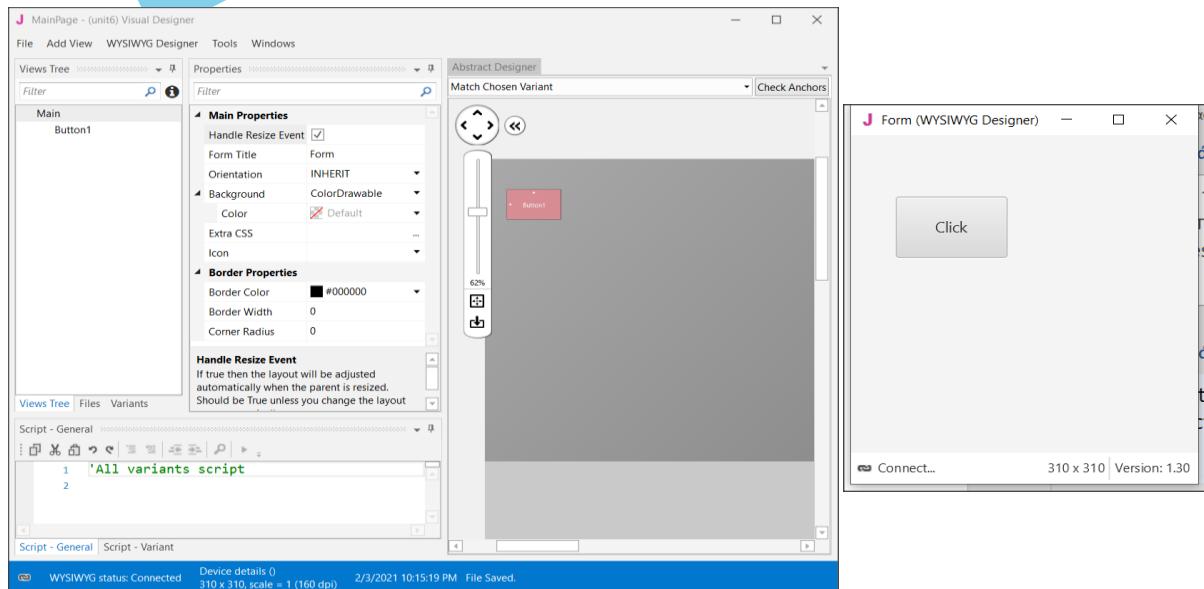


Figure 4 Designer Screen

Design a wireframe.

It is always a good idea to plan your screen layout before you start using the Visual Designer to set out your user interface. 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.

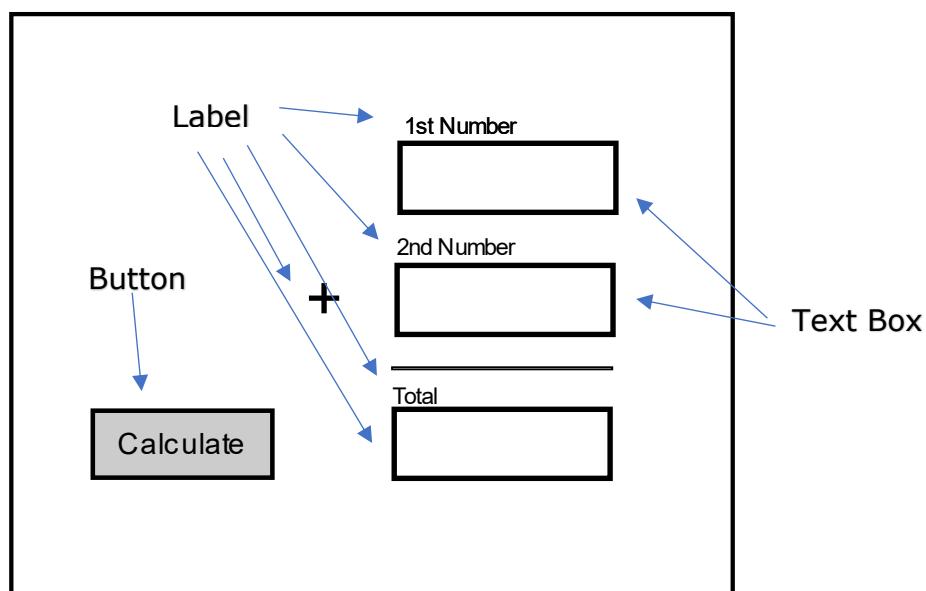


Figure 5 Wireframe



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 in the wireframe stage.



Remember

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

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.

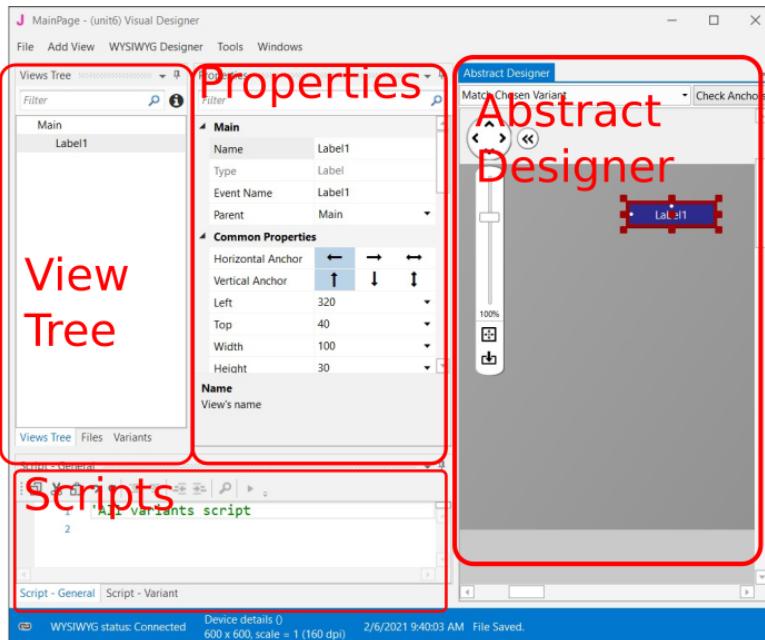


Figure 7 Designer's parts

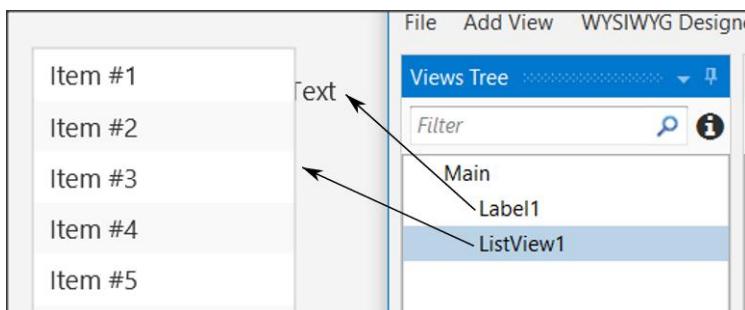


Figure 6 Views Tree

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 examples 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 in 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 screen 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 use only one. Also, you can remove any variant by selecting it and choosing "Remove Variant".

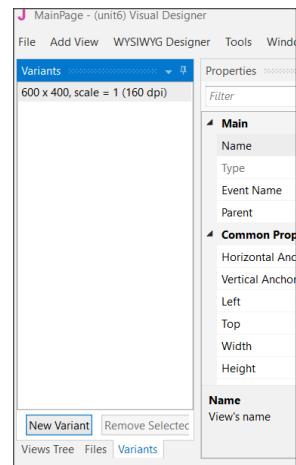


Figure 8 Variants Screen

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 decided 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".

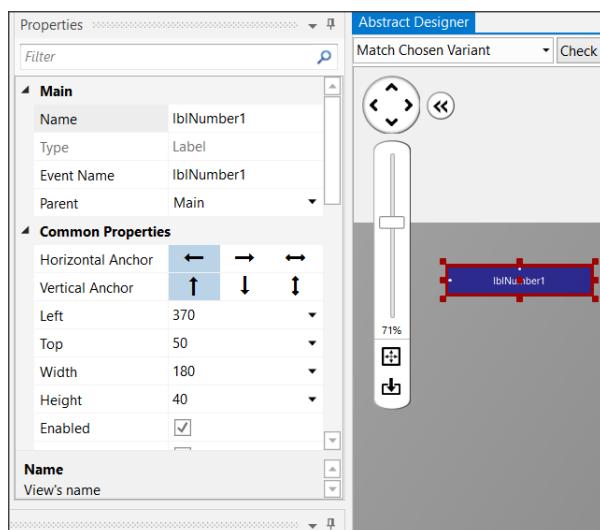


Figure 9 Labels

Inserting a Text Field.

Text Fields are used to input data into the program. There is no restriction on the type of data you can input. 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

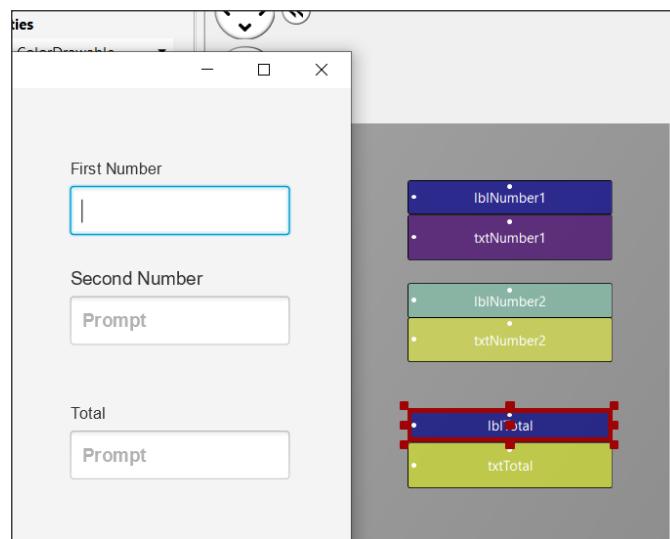


Figure 10 Text Fields

Inserting a button

Buttons in an app are used to invoke 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

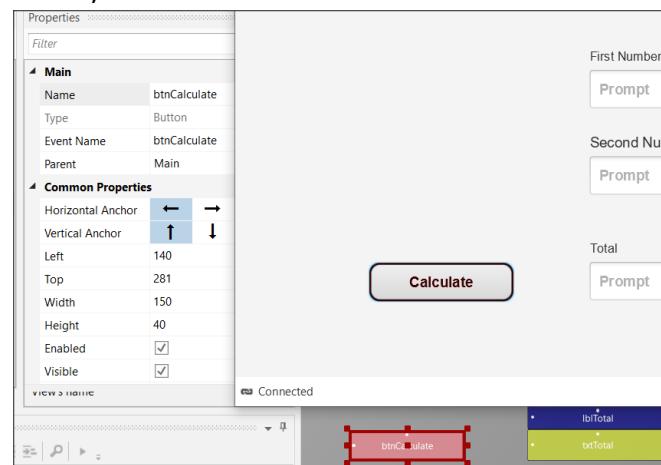


Figure 11 Buttons



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

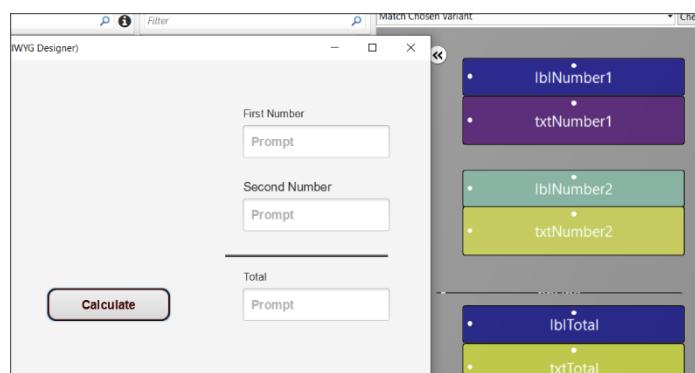


Figure 12 Pane

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



Teachers tip

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

Excercises

1. Use the designer to create the following wireframes.

Name <input type="text"/>	Class <input type="text"/>	
Surname <input type="text"/>	Age <input type="text"/>	
Save	Clear	Cancel

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:	<input type="text"/>	Calculate	

3. 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, creating the 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

When the B4XMainPage tab is clicked a code page is revealed. At the beginning of the code 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 has already been said in the 3rd Lesson Sub is a set of code that performs a specific operation. The intent of the Class_Globals is to collect the declarations of variables that we want to be known throughout

the code within the B4XMainPage module, i.e. in each subprogram.

In addition, if a variable statement starts with the Public it will be available from within other program modules.

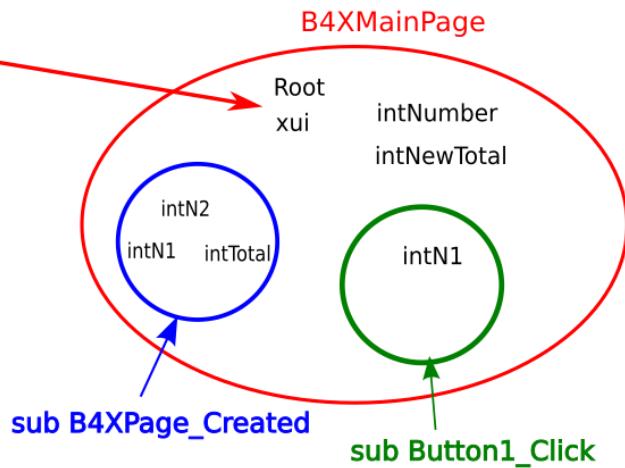


A deeper look at the use of variables.

In the following code you see three subprograms

B4XMainPage

```
8  @Sub Class_Globals
9    Private Root As B4XView
10   Private xui As XUI
11   Private intNumber As Int = 32
12   Private intNewTotal As Int
13  End Sub
14
15 @Public Sub Initialize
16  End Sub
17
18 @Private Sub B4XPage_Created (Root1 As B4XView)
19  Root = Root1
20  Root.LoadLayout(" MainPage")
21  Private intN1, intN2, intTotal As Int
22  intN1 = 45
23  intN2 = 32
24  intTotal = intN1 + intN2
25  Log("Total = " & intTotal)
26
27  intNewTotal = intTotal + intNumber
28  Log("New Total = " & intNewTotal)
29
30 End Sub
31
32 @Sub Button1_Click
33  xui.MsgboxAsync("Hello world!", "B4X")
34  Private intN1 As Int = 5
35  intNewTotal = intN1 + intNumber
36  Log("New Total = " & intNewTotal)
37 End Sub
38
```



Picture 3 Variables and Range

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

In contrast, 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** lives in the **Button1_Click** is **not the same** the one in **B4XPage_Created**, each occupies its own memory space, and both can have the same name. This is referred to as the scope of the variables meaning how widely within a module or the whole program a variable is accessible, or, to where access to a variable is restricted.



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

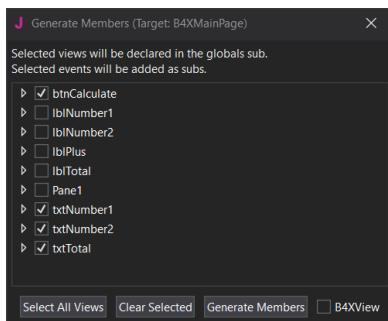
The screen you have already prepared contains objects that you must declare as variables in `Class _ Globals` to use in the program.

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. While button and textFields which will have programmable characteristics, must be added into the code editor in order to program their functions.

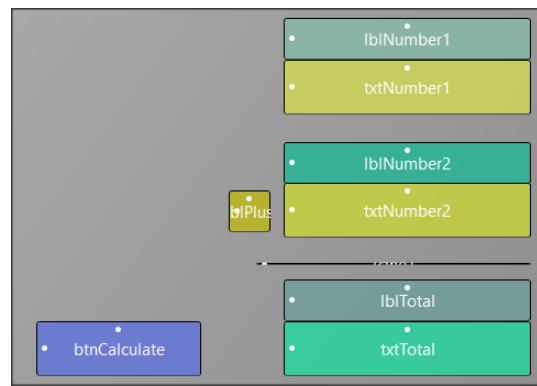
There are two ways to insert your objects into the code. The first is easily done through the designer with the following functions:

From the Tools menu select Generate Members



Picture 5 Generate Members

Your code in the Class_Globals subprogram will be updated automatically with the variables.



Picture 4 Example 1 Designer View

In Screen Generate Members just click on objects

- btnCalculate
- txtNumber1
- txtNumber2
- txtTotal

and then click Generate Members.

```

7  Sub Class_Globals
8  Private Root As B4XView
9  Private xui As XUI
10 Private btnCalculate As Button
11 Private txtNumber1 As TextField
12 Private txtNumber2 As TextField
13 Private txtTotal As TextField
14
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 type the declarations for the variables yourself, paying attention so that the names of the objects in the designer screen match.

Events

After you have declared the variables of the objects, the final stage is to write the program code for the functions of the form.

This will depend on what features and functions you have decided your app will have. In the example with the two numbers there is a button called Calculate which performs the addition when clicked and triggers the presentation of the result on your screen.



The operation is triggered by a process called an "**Event**". The programmer must detect the event of pressing the Calculate key 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.

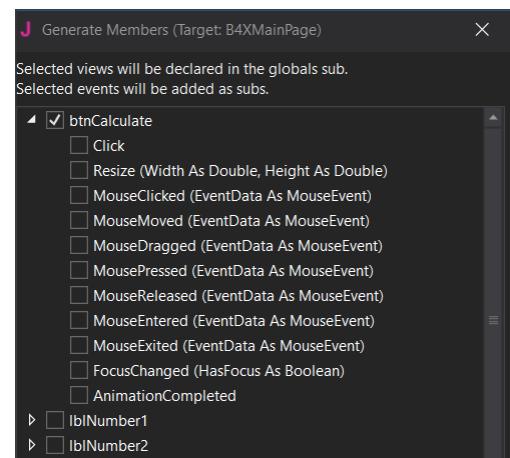
To see the Events associated with a program object open its list btnCalculate and several Events will be available to choose from. Just click on "Click" and "Generate Members".

Your code in the Class_Globals subprogram will be updated automatically with the variable declarations. Within it you will write the program code to complete.

```

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

```



Picture 7 Setting Event

Picture 8 The Click Event

Writing code in Event

All the functions associated with an object are performed within the Event subprogram.

```

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

```

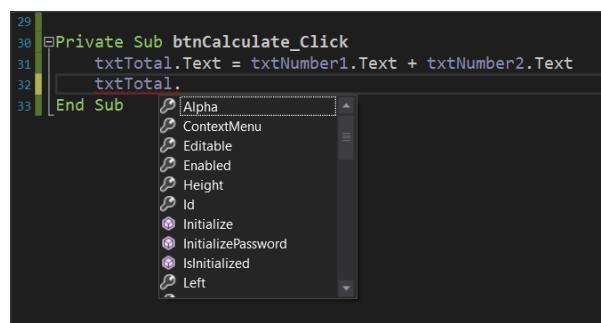
Picture 9 Calculating textFields

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 characteristics. 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 it. This information is string type.



Picture 10 Object Properties

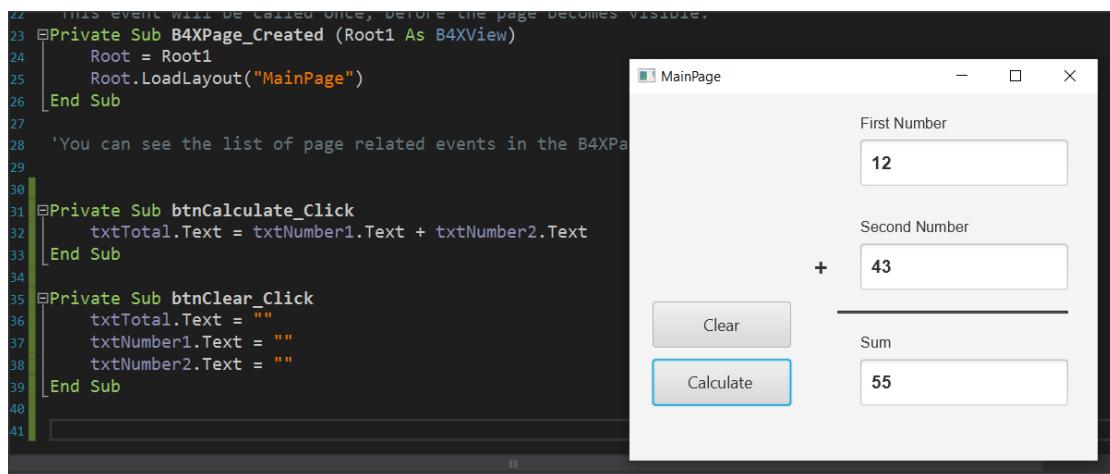


Remember

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 button 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.



Anywhere Software

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 lesson 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:	<input type="text"/>	Calculate	

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 a logical variable is extremely simple because it can accept only two values: True – False which in fact internally on the computer are translated into states 1 and 0.

The declaration 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 used in programming languages. They are the known mathematical symbols of inequalities but when programming they 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 can be either true or 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 **Not**.



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
rainning.
(P1)**

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

**P1 AND
P2**

P1 OR P2

NOT 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

- Comparing the two sentences using the logical AND evaluates to the value TRUE only when BOTH are true : It is raining (true) / I am going to school (true).
- Comparing the two sentences using the logical OR evaluates to the value TRUE when any one sentence is true : It is raining (true) / I am NOT going to school (false). Similarly : It is NOT raining (false) / I am going to school (true).
- The logical NOT reverses/converts true to false.

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(blnL1))          ' Shows False
58 |     Log (Not(blnL2))          ' 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			False		
			True			

5.

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

If statement

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 statement** is used to make the corresponding questions:

Its basic form is as follows:

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

Where condition appears in this example it would be replaced by a comparative or logical expression as 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 if-Else statement adds the ability to **if-else** to execute code if its condition is not true.

Its basic form is as follows:

```

If ( condition ) Then
    Execute this code
Else
    Execute this code
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** statement 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 evaluated, and if it is true, then the code it contains runs and the if is completed.
2. If the first **if** is false then the second is evaluated and if it is true it executes the code it contains and completed.
3. Other checks are always performed when the previous ones are false.
4. If none of the checks are true then the **else** statement executed.

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\$"

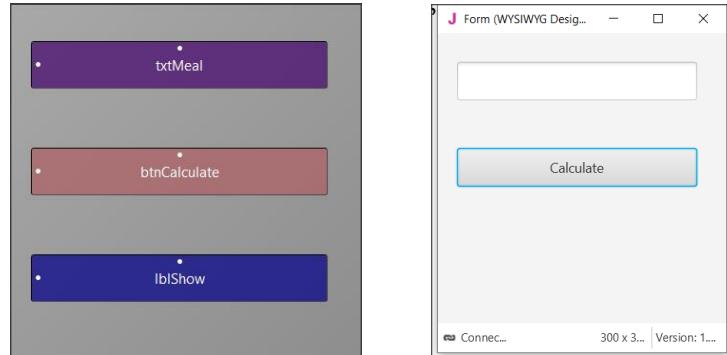
Solution

Step 1

Start a new project and give dimensions 300 x 300.

Step 2

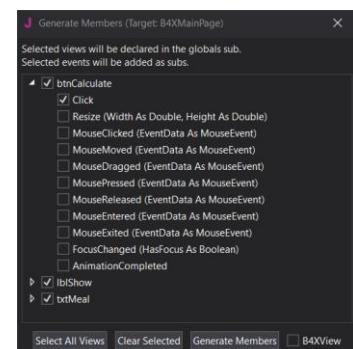
In the designer design the app screen



Picture 14 The app screen

Step 3

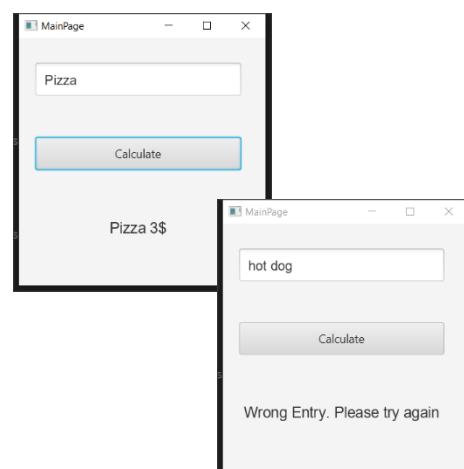
Enter txtMetal , btnCalculate , lblShow , and btnCalculate_Click.



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
26 Private Sub btnCalculate_Click
27   If txtMeal.Text = "Burger" Then
28     lblShow.Text = "Burger $5"
29   else if txtMeal.Text = "Pizza" Then
30     lblShow.Text = "Pizza 3$"
31   else if txtMeal.Text = "Hot Dog" Then
32     lblShow.Text = "Hot Dog 1.5$"
33   Else
34     lblShow.Text = "Wrong Entry. Please try again"
35   End If
36 End Sub
```



Note that the code considers capitals to be different from small letters. So Hot Dog is different from Hot Dog and HOT DOG and a user will have to input the correctly formated meal-name. Consider this when coding your program.



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

1. Turn logical expressions into code.
 - i. 'A' belongs to space [-5, 6).
 - ii. 'A' is less than 3 or more than 15.
 - iii. 'A' is equal to 'B' and 'C'.
 - iv. 'A' does not have a value of 3.
 - v. 'A' is less than 2 or 'B' is greater than 78.
 - vi. 'A' and 'B' true and 'C' false.
 - vii. the 'A' true and one of the 'B', 'C' are true.
2. 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
 - i. (A>B) OR (D=10)
 - ii. (D >= B) AND (E <> C)
 - iii. NOT (E<=C) OR (D<=C)
 - iv. NOT ((B<=C) AND (D<2))
 - v. NOT (NOT (B<=E) OR NOT (C<=B))
 - vi. ((E<=A) and (E>=C)) AND NOT (C>=A)
 - vii. NOT (NOT (A >= 2) AND (C <>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 ordered.

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:
 - a. Reads X (Wikipedia Mb consumption) and Y(watching meme Mb consumption)
 - b. Calculates the total consumption.
 - c. If total consumption greater than 100\$ print proper message If watching meme consumption is greater than reading Wikipedia articles print appropriate 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 spent. If the user is a member the tax is 10% else the tax is 20%. Create a program that:
 - a. Reads how many hours the user spent.
 - b. Check if is a member.
 - c. Add the proper tax fee.
 - d. 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:
 - a. Reads the cost of the product.
 - b. Reads your location.
 - c. Print the amount of money you must pay.
 - d. Output: "You have to pay 23\$, 20\$ for the product and 3\$ for shipping cost".



- 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.
- 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 the charge for the first 500 seconds is 0,01\$ then for the next 501 to 800 seconds it's 0,008 and then it's 0,005\$
- 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.

- 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 – Subprograms

⌚ 3h

What students should know

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

In computer programming, a subprogram (also it's called subroutine or routine) 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.

Subprograms may be defined within programs, or separately in libraries that can be used by many programs. In different programming languages, a subprogram 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.
- It should 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 already encountered the subroutine Button1_Click() and which was inserted by Generate Members. 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 subprogram 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 typing the name of the subprogram (which the developer decides) and then in parentheses the variables whose values it must know to function.

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

The Sub statement which means subprogram 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.

```
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
```

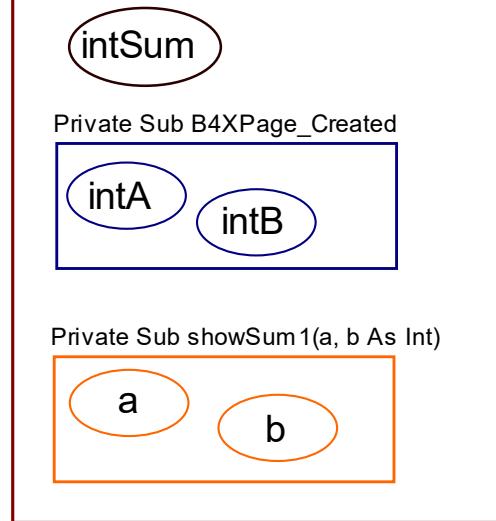
Picture 16 Calling a subprogram

The memory of the subprogram in B4X

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

```
8  Sub Class_Globals
9    Private Root As B4XView
10   Private xui As XUI
11   Private intSum As Int
12 End Sub
13
14 Public Sub Initialize
15 End Sub
16
17
18 Private Sub B4XPage_Created (Root1 As B4XView)
19   Root = Root1
20   Root.LoadLayout(" MainPage")
21   Private intA, intB As Int
22   intA = 10
23   intB = 30
24
25   showSum1(intA, intB)
26   Log(intSum)
27
28 End Sub
29
30 Private Sub showSum1(a As Int, b As Int)
31   intSum = a + b
32   Log(intSum)
33 End Sub
```

B4XMainPage



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 them easily. On the contrary, variables declared within the remaining subprogram 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 scope of the rest 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  Private Sub Sum(a As Int, b As Int) As Int
38      Return a+b
39  End Sub
40
```

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 the calculated returned value 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 1
18     intA = 20
19     intB = 10
20     intC = 300
21
22     Log(sMax(intA, intB, intC)) ' Shows 300
23 End Sub 5
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
35     Return(intM)
36 End Sub 4
```

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"/>	δ <input type="text"/>
b <input type="text"/>	x_1 <input type="text"/>
c <input type="text"/>	x_2 <input type="text"/>
Calculate	

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

The quadratic formula for the roots of the general quadratic equation

Picture 20 Source:
Wikipedia.org

Picture 21 Wireframe

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:
- Reads the code 1, 2, 3 representing respectively the categories.
 - Reads the number of seats requested.
 - Calculate the value of tickets with a subprogram that returns the amount, and which will be displayed in an appropriate textField

Lesson 9 – Classes

⌚3h

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 identical categories of data and similar actions that we can apply to them. Grouping all student data and functions into an independent and single code creates a Class. Each student record entered becomes an Object or Instance of the Class. Similarly, each Class Object has Properties characterized by variables such as student surname, registry number and so forth. The program instructions that operate on a Class's Objects are called 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.

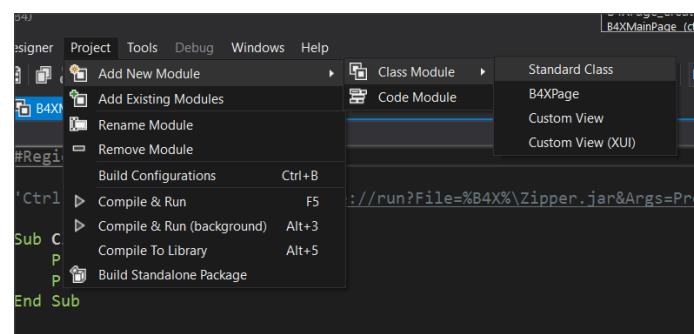
Example of classes in B4J

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

```

1  Sub Class_Globals
2      Private fx As JFX
3
4  End Sub
5
6
7  'Initializes the object. You can add parameters to this method if needed.
8  Public Sub Initialize
9
10 End Sub
11
12

```

5. Within the Class_Globals routine, add all the variables that will be the properties of the class.
- Book Title
 - Author Name
 - Publisher
 - Year of publication

```

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:
- Insert a book,
 - Show Book Items,
 - 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.

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

```

Insert a book.

This is a subprogram that accepts 4 string variables as parameters and then assigns them into the variables strTitle, strWriter, 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

```

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 XIIT
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 doing anything with an object

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

And, the logBook method displays the contents of the properties of each book.

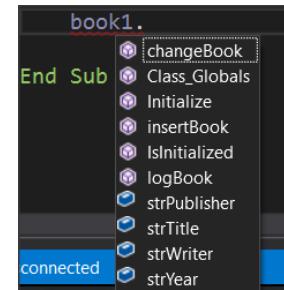


Remember

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.

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

Each time you type an object name and pressing the period B4X displays a popup dialogue 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.



Getters and Setters

Most of the time the properties of an object are kept private in order to avoid changes beyond what is allowed. This process is called encapsulation. To allow the use of private properties, the language uses public processes called get and set followed by the property name.

In the example of books, additional methods will be created as follows:



```

Sub Class_Globals
    Private fx As JFX
    Public strWriter, strTitle, strYear, strPublisher As String
    Private strWriter, strTitle, strYear, strPublisher As String
End Sub

Public Sub getWriter() As String
    Return strWriter
End Sub

Public Sub setWriter(w As String)
    strWriter = w
End Sub

Public Sub getTitle() As String
    Return strTitle
End Sub

Public Sub setTitle(t As String)
    strTitle = t
End Sub

Public Sub getYear() As String
    Return strYear
End Sub

Public Sub setYear(y As String)
    strYear = y
End Sub

Public Sub getPublisher() As String
    Return strPublisher
End Sub

Public Sub setPublisher(p As String)
    strPublisher = p
End Sub

```

Also, to insert a book using the insert method can be called:

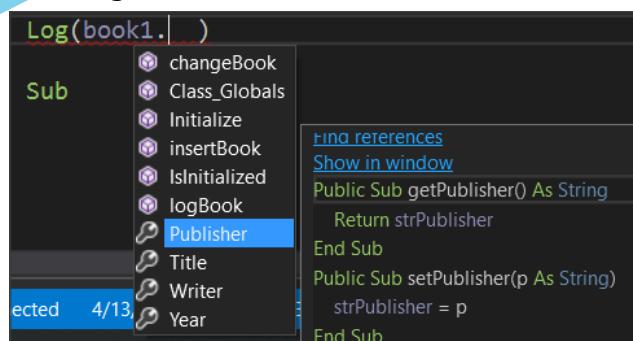
```

Public Sub insertBook(str1, str2, str3, str4 As String)
    setTitle(str1)
    setWriter(str2)
    setYear(str3)
    setPublisher(str4)
End Sub

```

Using the Set methods, you can integrate additional controls into the data you are about to enter.

Finally, when you want to use the Get and Set methods do it by typing the name that follows the get and set.



For example, to set another value to the author of book1 you can simply write:

```
book1.Writer = "Wil. Gibson"  
Log(book1.Writer)
```

Exercises

1. Referring to the first example of the lesson, 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

- a. Lesson
 - b. Class
 - c. Hours
 - d. Professor

and methods

- a. New Lesson
 - b. Change Hours
 - c. Change Professor

- d. Show Lesson
3. A store has an inventory of computers for sale. For each computer, the following are recorded:
- a. The type (desktop, laptop),
 - b. the model,
 - c. its price,
 - d. 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

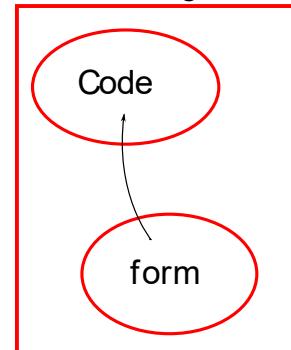
What students should know

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

B4XPages is a software library. The library includes classes and procedures to create multiple application communication forms to interface 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 B4X MainPage 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.

B4X MainPage



The structure of an application's folders

When you start a new program with B4XPages, the following folder structure is created.

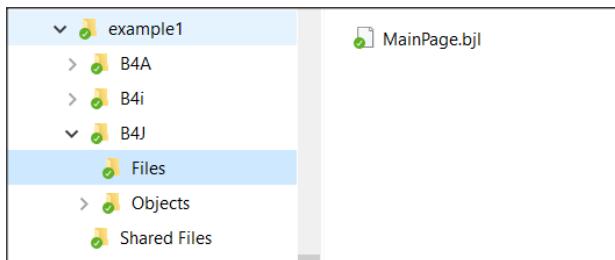


Figure 1 example1 folders

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

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

containing 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 includes files that the three different applications can share if the developer **chooses** to create an application for Android, IOS and PC.



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

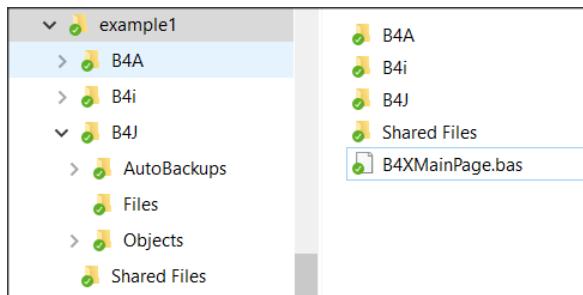


Figure 2 B4XPage files

The first page that is created must have the name "**B4X MainPage.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 IDE has already prepared the first page and as mentioned its name in the application folder is B4X MainPage.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
```

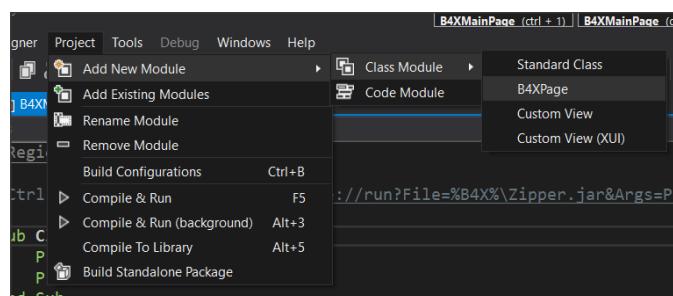
What is Root

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.

Now that the app has been created select from the menu Project – Add New Module – Class Module – B4XPage and give it the name B4XPage1.



Picture 22 Create B4XPage

A class with a name “B4XPage1” will be created, with some necessary codes to get started. The user communication screen (GUI) has not yet been created at this point. This should be done using 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

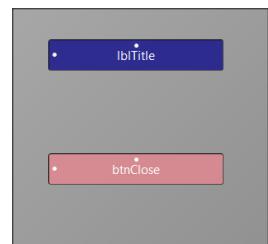
Private Sub B4XPage_Created (Root1 As
B4XView)
    Root = Root1
    'load the layout to Root
End Sub
```

Picture 23 The new B4XPage

Step 2.

Open Designer and from the menu File select New.

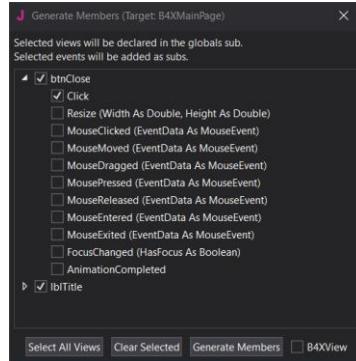
From the Variants tab, specify the dimensions of the form you want to design, and then select a label and button to insert into your form as in the Picture 24.



Picture 24 Form

Step 3.

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

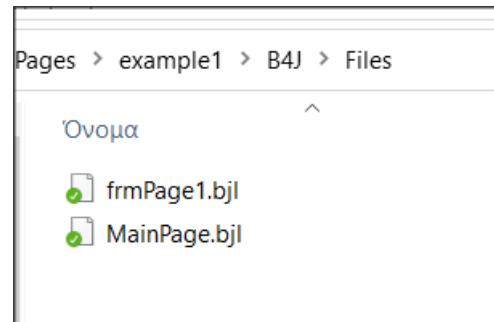


Picture 25 Generate Members

From the file menu save your form named frmPage1. (You can give any name you want).

The following code (Picture 26) will now be created, and filenames will be seen in the folder files.

```
1  Sub Class_Globals
2    Private Root As B4XView 'ignore
3    Private xui As XUI 'ignore
4    Private.btnClose As Button
5    Private lblTitle As Label
6  End Sub
7
8  'You can add more parameters here.
9  Public Sub Initialize As Object
10  	Return Me
11  End Sub
12
13  Private Sub B4XPage_Created (Root1 As B4XView)
14  	Root = Root1
15  	'load the layout to Root
16  End Sub
17
18  Private Sub btnClose_Click
19
20  End Sub
```



Picture 26 frmPage1

Step 4.

To link the form frmPage1 with the B4XPage1 you must now call the method 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 programming the methods 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 in the B4XPage1 form to return to the home page.

Call a new B4Xpage.

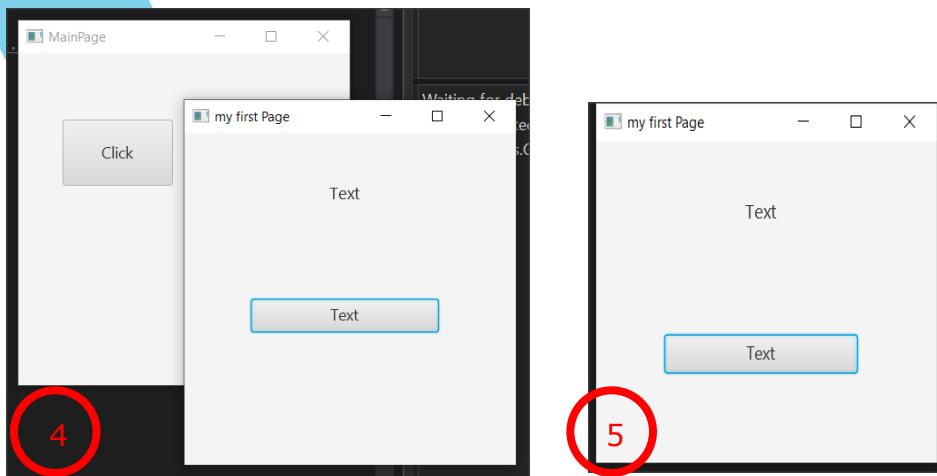
Each new B4XPage you create is a class. Therefore, before you can use it, you must create an object based on it. The creation is usually done in the B4XPage that will call the new one. So, in the previous example we write in B4XmainPage (Picture 27):

```
/ 8  ESub Class_Globals
9   Private Root As B4XView
10  Private xui As XUI
11  Private page1 As B4XPage1 ①
12 End Sub
13
14 EPublic Sub Initialize
15
16 End Sub
17
18 EPrivate Sub B4XPage_Created (Root1 As B4XView)
19   Root = Root1
20   Root.LoadLayout("MainPage")
21   page1.Initialize ②
22   B4XPages.AddPage("my first Page", page1) ③
23 End Sub
24
25 EPrivate Sub Button1_Click
26
27   ④ B4XPages.ShowPage("my first Page")
28
29   ⑤ 'B4XPages.ShowPageAndRemovePreviousPages("my first Page")
30
31 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:
B4XPages.ShowPage("my first Page")	B4XPages.ClosePage(Me)
B4XPages.ShowPageAndRemovePreviousPages("my first Page")	B4XPages.ShowPageAndRemovePreviousPages ("MainPage")

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

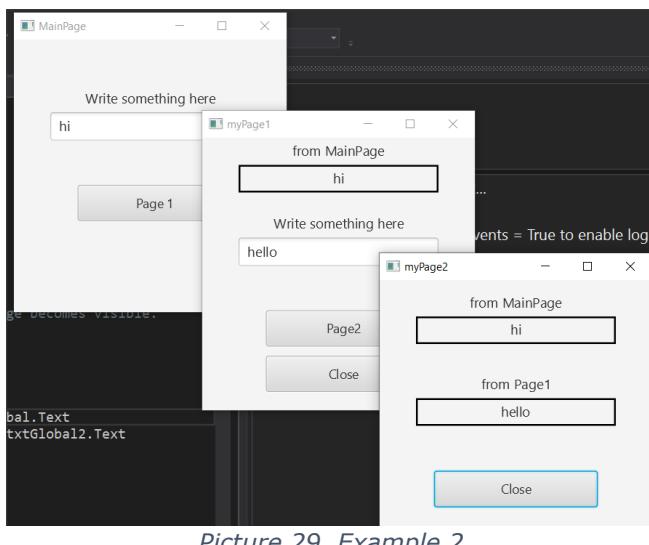
Transfer information between pages

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.



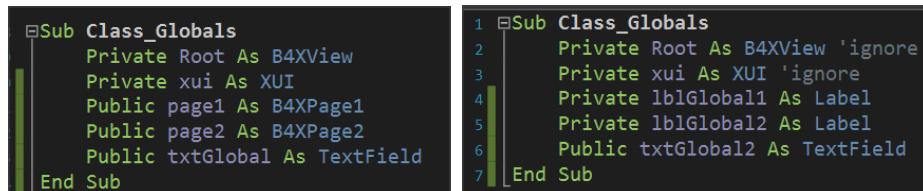
Example 2

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 in both page1 and page2 TextField is declared public.



Picture 30 Public declarations in MainPage and Page1

In order for page1 to have access to the txtGlobal1 variable, it must be referenced 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 as assigned 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
```

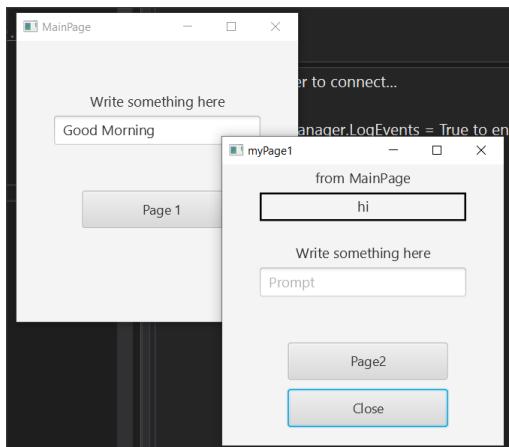


```
lblGlobal2.Text = B4XPages.MainPage.page1.txtGlobal2.Text
```

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.txtGlobal1.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 dog 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.*



Lesson 11 – First Project

⌚ 5h

What students should know

- Use previous knowledge to create your first big application.

Mobile Phones shop application.

A store sells cell phones. The following characteristics are recorded for each phone:

- Model Name
- Screen size,
- Memory capacity,
- RAM,
- Processor
- Warehouse Pieces
- Operating System (Android, IOS)
- Device Price

Build an application with the following:

1.
 - 1.1. Create a class with Phone name and properties of the above attributes.
 - 1.2. Create the methods.
 - 1.2.1. Initialize (Initialize all string properties with "" and all numeric values with 0.
 - 1.2.2. New model (with value input operations on a new object)
 - 1.2.3. Phone sale (accepts a number of pieces and subtracts this quantity from Warehouse Stock)
2. Add 8 different objects for mobile phones with the following elements (The features presented are not real but written for the needs of the task):

Model Name	Screen	Capacity	Ram	Cpu	Os	Store	Price
1 Yallomi	6.53"	64	4	Mediatek	Android	12	150\$
2 Smith	6.67"	64	4	Qualcomm	Android	4	220\$
3 Taurus	6.1	128	4	Bionic	Ios	7	780\$
4 Talisman	5.8"	64	4	Mediatek	Android	12	150\$
5 Cranberry	5.8"	32	3	Mediatek	Android	16	130\$
6 OzOn	5.8"	32	2	Mediatek	Android	16	90\$
7 H2O	5.8"	64	3	Qualcomm	Android	2	170\$
8 Zeus	6.67"	128	6	Qualcomm	Android	4	650\$

For each page below, it is recommended that you draw a sketch either on paper or with the help of the Inkscape program. You can download it for free from <https://inkscape.org/>.



Anywhere Software

- 3.** Create appropriate home screen with the following options.
 - Sell a mobile phone.
 - Warehouse Stock Status.
 - Total Store Revenue.
- 4.** If "Sell Mobile" is selected, then create new B4XPage, and show the names of the eight models and their prices. Don't forget to sketch it first.



Teachers tip

Maybe, you have to help students to call objects from main page.

- 4.1.** Clicking on each phone displays a new screen showing the device's features along with two photos of it. In addition, the feature screen also displays a button to sell the device with number of pieces and a back button.
 - 4.1.1.** If the "back" button is pressed, then return to the previous screen.
 - 4.1.2.** If the "sell" is pressed, check if the pieces are available in the warehouse or show an out of stock message. On a new screen request the buyer's details:

Customer Name: Surname Address Phone

- 4.1.3.** In the same screen display the number of pieces the buyer chose to buy as well as the total price.
- 4.1.4.** When Sell button pressed subtract the quantity purchased from the warehouse and return to the app's home screen.
- 5.** If Warehouse Stock Status is selected, display available mobile models on a new screen(B4XPage) along with the quantities in stock.
- 6.** If "Total Store Revenue" is selected, the store's receipts will be displayed on a new page(B4XPage) until this time.



Teacher's tip

In Appendix, you will find, some documents to help students to work on this first project. The project's solution steps are in the PowerPoint presentation. While the solution is not "optimal" students should understand it, with their current knowledge.

Lesson 12 – Loops

⌚ 4h

What students should know

- What are Loops?
- Do While
- Do Until
- For – Next
- Algorithms with loops

Repetitive structures are one of the most basic functions of a programming language. A loop in a computer program is a directive that repeats until a specified condition is reached. In a repeat structure, the loop the specified condition is tested on each iteration of the loop. If the answer to the "loop until" requires action, the code in the body of the loop executes. The test condition is evaluated over and over again until no further action is required, this should be when the "stop looping" test condition is met. Every time the question is asked it is called repetition or iteration..

A computer programmer who has to use the same lines of code multiple times in a program can use a loop to save time, space, ease of programming.



Remember

Repeats have to end at some point or it's a programming error. A repetition that never ends is called an endless loop. In this case the computer is trapped in a perpetual repetition of the same functions without "awareness" of vanity!

In B4X there are four repeat **commands**: For, Do While, Do Until, for Each.

The Do While command

Suppose you would like to display on the screen with the command Log() numbers from 1 to 5 below each other. The code you should write would produce this output in the log panel:

```
Log (1)  
Log (2)  
Log (3)  
Log (4)  
Log (5)
```

You realize that if you had to keep printing numbers then you would have to continue writing Log() commands for as many numbers as you need.



Anywhere Software

The command Do While is stated as follows:

Do While Condition

Commands

...

Loop

As long as the condition is true

Execute the Commands

...

Go back to condition.

Picture 32 Command Do While

In the example with the five numbers and using Do While the code is written:

```
Private i as Int = 1  
  
Do While i <= 5  
    Log(i)  
    i = i + 1  
Loop
```

Variable **i** counts how many times the loop will run.

The Condition also examines whether the contents of the counter exceeded the limit set by the developer, i.e. 5, and if this happens, the repetition stops.

Before the loop iteration end, the repeat counter is increased by +1. This value is also called Step.

The step can be positive or negative, but it can never be 0 because then the repetition would run forever.



Remember

When the counter starts at a value less than the final value then the **step must** be **positive**.

When the counter starts from a value greater than the final value then the **step must** be **negative**.

Examples of the command Do While.



Example 1

Show all integers from 100 to 1

```

Private i as Int = 100

Do While i >= 1
  Log(i)
  i = i - 1
Loop

```

In the example the repeat counter starts at a large value (100) and ends at a small (0) before the repetition is complete. Attention in this case to the **Step** is **Negative** (-1).

Example 2

Show all even numbers from 1 to 100

```

Private i as Int = 1

Do While i <= 100
  If i mod 2 = 0
  then
    Log(i)
  End if
  i = i + 1
Loop

```

```

Private i as Int = 2

Do While i <= 100
  Log(i)
  i = i + 2
Loop

```

Here are two solutions presented the first uses a command if in loop to check whether the number is even and only then executes the command Log. In the second algorithm has changed the **starting and step values** of the i in 2 and the **Step** increases by 2 which creates a faster algorithm to run.

Example 3 – Sum Algorithm

Make a program that for ten numbers entered, the program calculates their sum. These numbers are considered to range between -100 and 100



The function will be used for the purposes of the example. It is used as follows:

```
Private I As Int = 1
Private A as Int
Private intSum as Int = 0

Do While i <= 10
    A = Rnd(-100, 101)
    intSum = intSum + A

    i = i + 1
Loop
```



Remember

The **Rnd command (First Value, Last Value)** returns a number between the First (inclusive) and Last (exclusive) value.

e.g. A = Rnd(1, 10) returns a number between 1 and 9 in variable A



Example 4 – Algorithm Count.

Make a program that for ten numbers entered, the program end calculates the number of negatives. These numbers are considered to range between -1000 and 1000

```
Private I As Int = 1
Private A as Int
Private intCounter as Int = 0

Do While i <= 10
    A = Rnd(-1000, 1001)
    If A < 0 then
        intCounter = intCounter + 1
    End If
    i = i + 1
Loop
```



Example 5 - Maximum Algorithm - Minimum

Make a program that for ten numbers entered the program counts the largest and smallest numbers. These numbers are considered to range between -1000 and 1000

```
Private I As Int = 1
Private A as Int
Private intMax, intMin as Int

A = Rnd(-1000, 1001) ①
intMax = A
intMin = A
Do While i <= 9
    A = Rnd(-100, 100)

    If intMax < A then
        intMax = A
    End If ②

    If intMin > A then
        intMin = A
    End If

    i = i + 1
Loop
```

1. First read a number outside the while.
2. Set as the starting value in the intMax and intMin variable the first number since there is no one else to compare.
3. For each new number it is checked if it is less than intMin then intMin replaced by A if the new A is greater than intMax the intMax is replaced with the new A.

Loops with unknown repeats.

Often in programming we do not know from the beginning the number of repetitions. This usually happens when the repeat depends on the value taken from user or on values calculated during the repeat.



Example 6 – Non-specific number of repetitions

Create a program that constantly reads numbers and calculates the Averages of the even values. The program stops when a number less than 0 is entered.



```

Private A  as Int
Private intCount  as Int = 0

A = Rnd(-100, 100) 1
Do While A > 0 2
    If A mod 2 = 0 then
        intCount = intCount + 1
    End If

    A = Rnd(-100, 100) 3
Loop

```

1. First read a number outside the replay.
2. The DoWhile condition checks whether the number A is within the limits.
3. A new number is read before the end of the iteration.



Example 7 – Non-specific number of repetitions

Create a program that reads numbers and calculates their sum. The program stops when the sum exceeds 200.

```

Private A  as Int
Private intSum  as Int = 0

Do While intSum <= 200 1
    A = Rnd(-100, 100)
    intSum = intSum + A 2
Loop

```

1. The condition checks whether the total has not reached the limit of 200.
2. A number is read and then program calculates the sum. Condition checked again by Do While.

The Do Until command.

The operation of the Do Until is like the Do While. The only difference is the condition which controls when a repetition will end as opposed to the Do While which controls how long it will work. In both cases the check shall be carried out at the beginning, in which case the code shall not be executed if the Condition is False.



Remember

The condition of Do Until is the denial of Do While.



Example 1

Show all numbers from 100 to 1

```
Private i as Int = 100  
  
Do Until i < 1  
    Log(i)  
    i = i - 1  
Loop
```



Example 2

Show all even numbers from 1 to 100

```
Private i as Int = 1  
  
Do Until i > 100  
    If i mod 2 = 0 then  
        Log(i)  
    End if  
    i = i + 1  
Loop
```

```
Private i as Int = 2  
  
Do Until i > 100  
    Log(i)  
    i = i + 2  
Loop
```



Example 3 – Sum Algorithm

Make a program that for ten numbers entered, the program calculates their sum. These numbers are considered to range between -100 and 100.

```
Private I As Int = 1  
Private A as Int  
Private intSum as Int = 0  
  
Do Until I > 10  
    A = Rnd(-100, 100)  
    intSum = intSum + A  
  
    i = i + 1  
Loop
```





Example 4 – Algorithm Count

Make a program that for ten numbers entered, the program calculates the number of negatives. These numbers are considered to range between -1000 and 1000.

```
Private i as Int = 1
Private A as Int
Private intCounter as Int = 0

Do Until i > 10
    A = Rnd(-1000, 1001)
    If A < 0 then
        intCounter = intCounter + 1
    End If
    i = i + 1
Loop
```



Example 5 - Maximum Algorithm - Minimum

Make a program that for ten numbers entered, the program counts the largest and smallest numbers. These numbers are considered to range between -1000 and 1000.

```
Private I As Int = 1
Private A as Int
Private intMax, intMin as Int

A = Rnd(-1000, 1001)
intMax = A
intMin = A
Do Until I > 9
    A = Rnd(-1000, 1001)

    If intMax < A then
        intMax = A
    End If

    If intMin > A then
        intMin = A
    End If

    i = i + 1
Loop
```



Example 6 – Non-specific number of repetitions

Create a program that constantly reads numbers and calculates the Averages of the even. The program stops when a number less than 0 is entered.

```
Private A  as Int
Private intCount  as Int  = 0

A = Rnd(-100, 100)
Do Until A < 0

    If A mod 2 = 0  then
        intCount = intCount + 1
    End If

    A = Rnd(-100, 100)
Loop
```



Example 7 – Non-specific number of repetitions

Create a program that reads numbers and calculates their sum. The program stops when the sum exceeds 200.

```
Private A  as Int
Private intSum  as Int  = 0

Do Until intSum > 200
    A = Rnd(-100, 100)
    intSum = intSum + A
Loop
```

For Loop

The for loop is probably the simplest repeat command.

```
for i = n1 to n2 Step n3
    Commands
    ...
Next
```

Where:

i the Counter variable

n1 the initial value of the Counter

n2 the final value

n3 step of repetition



- At the beginning of repetition, variable 'i' accepts n1.
- Executes commands within the repetition.
- At the end of the repetition the value of 'i' increases or decreases by n3
- Check if 'i' has exceeded the final value n2.
 - if the step is positive and 'i' is less than or equal to the final value then the repeat runs again
 - If the step is negative and 'i' is greater than or equal to the final value, then the repeat runs again.



Remember

In relation to **Do While** and **Do Until** the **For** command:

- Does not need to initialize the counter.
- Does not need to set the step change operation.
- It is always for measurable repeats (however, you can use the exit command to get out of it at any time).

Examples of For loops



Example 1

Show all numbers from 100 to 1

```
Private I As Int
For i = 100 to 1 step -1
  Log(i)
Next
```



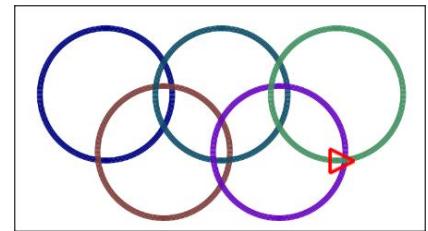
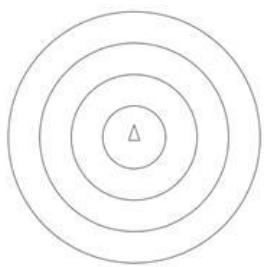
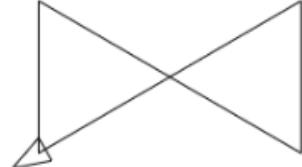
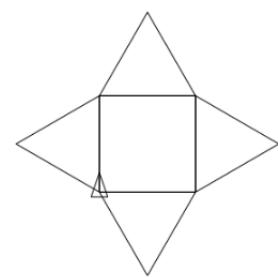
Example 2

Show all even numbers from 1 to 100

```
Private I As Int
For i = 1 to 100
  If i mod 2 = 0 then
    Log(i)
  End If
Next
```

Exercises

1. Write a program that reads an integer K between 1 and 200 and calculates the sum of $1+2+3+\dots+K$.
2. Write a program in which the user enters numbers and calculate the average of the above numbers. The program ends by entering zero.
3. A consumer wants to buy games for gifts to 10 children and also wants their total price not to exceed 200€. Make a program to:
 - a. Enter the cost of the game (random number between 10-50, command Rnd(10, 51))
 - b. Calculate the total cost of games purchased so far,
 - c. Check if the money has run out.
 - d. Finally show the total cost of the games and their number.
4. A leap year (also known as an intercalary year or bissextile year) is a calendar year that contains an additional day (or, in the case of a lunisolar calendar, a month) added to keep the calendar year synchronized with the astronomical year or seasonal year. A year is a leap year when:
if (year is not divisible by 4) then (it is a common year)
else if (year is not divisible by 100) then (it is a leap year)
else if (year is not divisible by 400) then (it is a common year)
else (it is a leap year)
Make a program that shows the leap years between 1900 and 2100
(https://en.wikipedia.org/wiki/Leap_year)
5. Build a program with the help of the turtle that creates polygons with a number of angles that the user enters in an appropriate field. The program will include a design start button and a screen clearing button that when pressed clears the screen and places the turtle in center of screen.
6. Draw the following shapes with the turtle:



Lesson 13 – XUI Views

⌚3h

What students should know

- What is a library.
- XUI library.
- Dialogs
- Templates

Each programming language has libraries. In general, a library includes collections of code, structures, classes, methods that can be used by developers to facilitate the process of developing a program.

Libraries in B4X

Libraries are generally divided into internal libraries and are those that are installed together with the language and external ones created by the developer himself or found from other sources (e.g. Github). To use a library, you only need to select it from the corresponding Libraries list in B4X.

The list of libraries contains useful information. These are:

- The current version of the library that is installed,
- The latest version published in order to update our language.
- Whether it is an internal library or an external library.
- Which platforms it concerns.

To use a library, you must have been informed about the functions it performs as well as the data and methods it uses. For each library there is relevant information about its use through the language website at <https://www.b4x.com/android/documentation.html>.

Name	Version	Online	Path	Platforms	Comments
B4XPages	1.08	1.08	Internal	B4A, B4i, B4	
jCore	7.51	7.51	Internal	B4J	
jFX	8.80		Internal	B4J	
B4XCollections		1.08	Internal		
B4XDrawer		1.55	Internal		
B4XFormatter		1.03	Internal		
B4XPagesTemplates		1.00	Internal		
B4XPreferencesDialc		1.72	Internal		
B4XTable		1.21	Internal		
B4XTurtle		1.06	Internal		
BCTextEngine		1.86	Internal		
BCToast		1.01	Internal		
CSSUtils		1.20	Internal		
JavaObject		2.06	Internal		
jB4XEncryption		1.00	Internal		
jBitmapCreator		4.73	Internal		
jControlsFX			Internal		
jControlsFX9			Internal		
jDateUtils		1.05	Internal		
jGameViewHelper			Internal		
jMQTT		1.01	Internal		
jNet		1.81	Internal		
jNetwork		1.20	Internal		
jOKHttpUtils2		2.95	Internal		
jOKHttpUtils2_NONU			Internal		

Picture 33 Libraries

through the language website at <https://www.b4x.com/android/documentation.html>.



Anywhere Software

External libraries should always be copied into a specific folder on your computer. From Menu "Tools", "Configure Paths", "Additional Libraries" select the folder in which they will be stored.



Teacher's tip

Below are some important views. Other very important ones are absent from this list, such as ComboBox, which we will refer to in more extensively in the next chapters.

The XUI Views library

The purpose of the XUI Views library is to offer a common way to create applications for b4j, b4a and b4i.

This library is associated with creating forms and views that are included in them. Includes:

- **Views** (objects)
 - B4XComboBox - Cross platform ComboBox / Spinner / ActionSheet.
 - ScrollingLabel - A label that scrolls the text when it is wider than the label.
 - AnotherProgressBar - Vertical or horizontal animated progress bar.
 - B4XLoadingIndicator - 6 different animated loading indicators.
 - RoundSlider - A round slider.
 - SwiftButton - 3d button
 - AnimatedCounter
 - B4XFloatTextField - A TextField / EditText with a floating hint
 - B4XSwitch - Nice looking two state control.
 - B4XPlusMinus - Allows the user to select a number or item from a previously set list.
 - B4XBreadCrumb - Navigation control.
 - B4XSeekBar - Horizontal or vertical seek bar / slider.
 - MadeWithLove - Show your love to B4 :)
 - B4XImageView - ImageView with useful resize modes.
 - XUIViewsUtils - Static code module with various utility methods.
- **Dialogs** (Dialog Boxes)
 - A class that provides the features required to show a dialog. There are three methods to show dialogs: Show - Shows a simple dialog with text, ShowCustom - Allows you to pass a layout of your own and show it as a dialog, ShowTemplate - Shows a dialog based on a template class. See the source code for the template structure. It is quite simple.
- **Templates**
 - B4XDateTemplate - Based on AnotherDatePicker.
 - B4XColorTemplate - Color picker.

- B4XLongTextTemplate - Scrollable text.
- B4XListTemplate - A list of items. The user can choose one of the items.
- B4XSignatureTemplate - Captures the user signature and adds a timestamp to the bitmap.
- B4XInputTemplate - Template for text and numeric inputs.
- B4XSearchTemplate - A list with a search field.
- B4XTimedTemplate - A template that wraps other templates and creates a dialog that closes automatically after the set time with a nice, animated progress bar.

Using XUI Views

To use the XUI Views first you need to check it's name on the library tab. Then go to the Designer to create the objects you want.



Example

The following program shows the use of:

ScrollingLabel

B4XFloatTextField

RoundSlider

AnotherProgressBar

B4XSwitch

B4XImageView

Scrolling Label

Scrolling label is a text label where it can move text (left or right direction) so that it can appear in its entirety.

```

Private scrLbl0dys As ScrollingLabel
Private Sub btnShowPoem_Click
    scrLbl0dys.Text = "Tell me, O Muse, of that many-sided hero who " & _
        "traveled far and wide after he had sacked the "& _
        "famous town of Troy. Many cities did he visit, "& _
        "And many were the people with whose customs And "& _
        "thinking he was acquainted; many things "& _
        "he suffered at sea While seeking To save his "& _
        "own life And To achieve the safe homecoming "& _
        "of his companions; but Do what he might he could Not "& _
        "save his men, For they perished through their own sheer "& _
        "recklessness in eating the cattle of the Sun-god Helios; "& _
        "so the god prevented them from ever reaching home. Tell "& _
        "Me, As you have told those who came before Me, about "& _
        "all these things, O daughter of Zeus, starting from "& _
        "whatsoever point you choose."

```

Picture 34 Scrolling Label

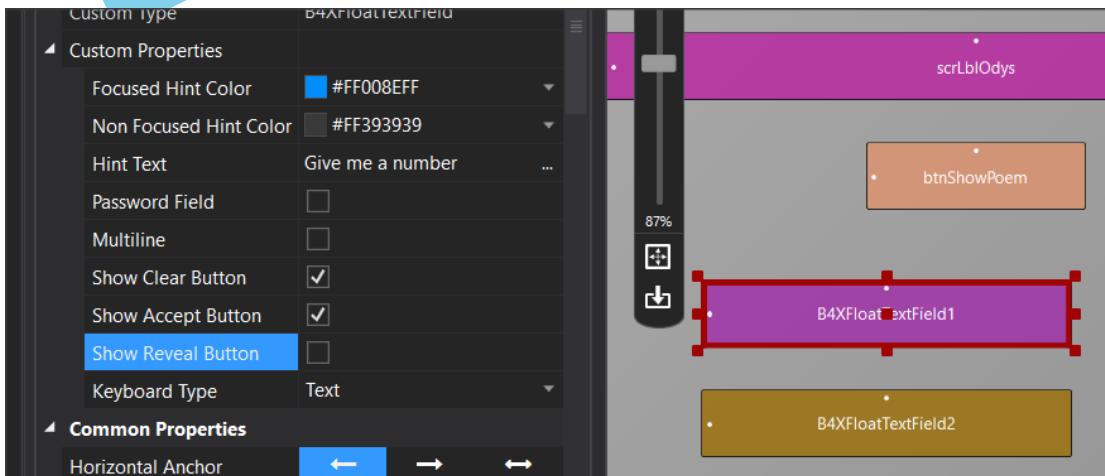
It is mainly used for very long texts or to display a running message on the screen.

B4XFloatTextField

B4XFloatingTextField creates a text box on the screen to insert data. Its use is similar to the simple text box, but in addition it displays a label that helps



the user identify the field without inserting an additional label before the field (Hint Text).

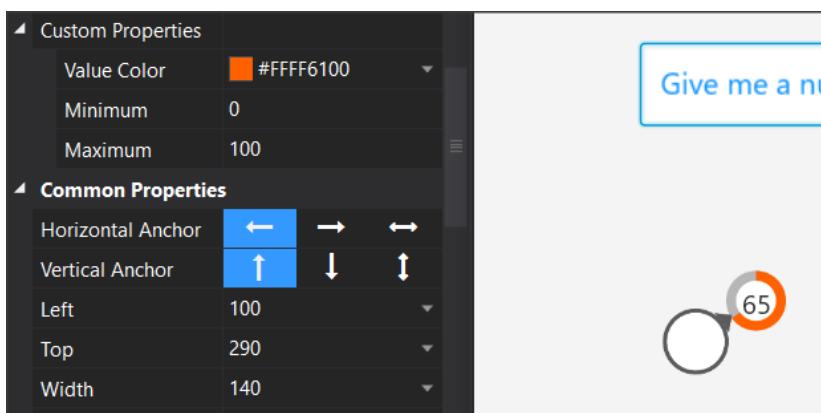


Picture 35 B4XFloatTeextField

Additionally, it displays the appropriate controls to delete or enable this field (Show Clear Button, Show Accept button). Finally, it allows the Keyboard Type property to enter text, either integer numbers or decimal numbers.

RoundSlider

RoundSlider is a controller that moves with the mouse around a circle. At each point in the cycle, it displays a value from a range of values that you have selected in the Minimum – Maximum properties in Designer.



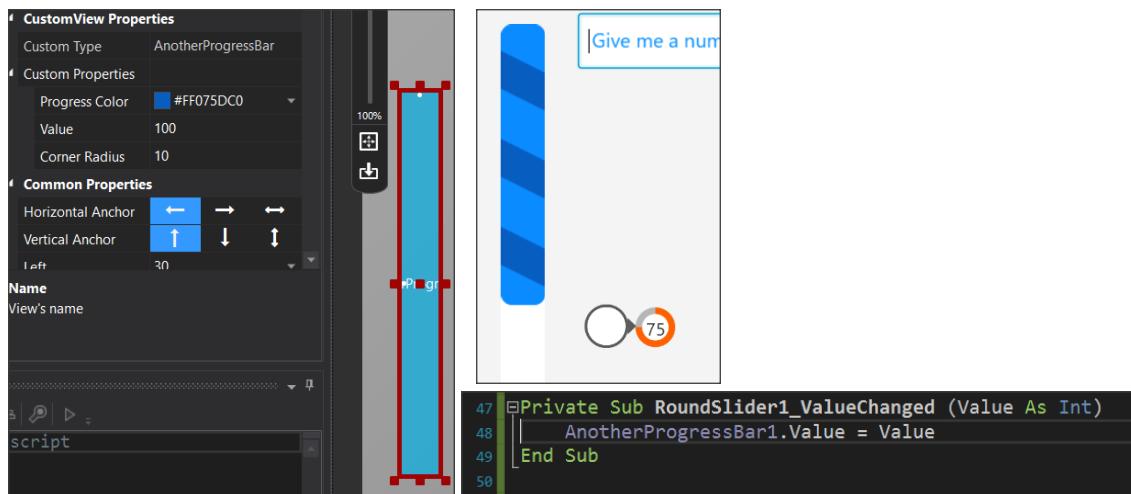
Picture 36 RoundSlider

Each time the RoundSlider value changes, the "_ValueChanged" event is triggered, and gives value of which the developer can use.

AnotherProgressBar

AnotherProgressBar displays a bar that varies between 0 and 100 and is suitable for displaying percentages or other proportions. Using it is simple, after you declare the variable that corresponds to the item, use the property ".value" to set a value.

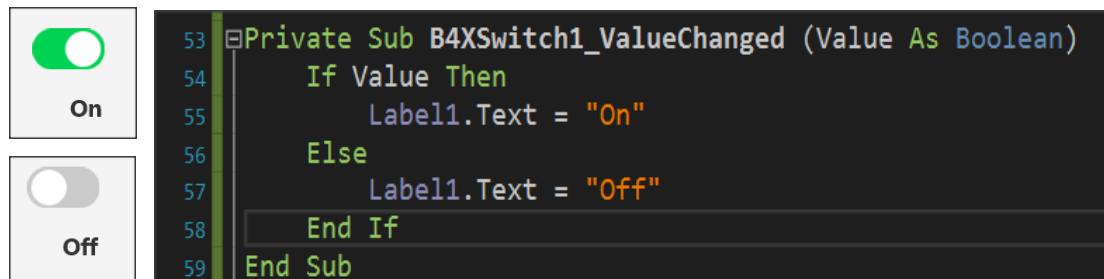
In the example image below, the progress bar varies depending on the value it receives from RoundSlider.



Picture 37 AnotherProgressBar

B4XSwitch

B4XSwitch creates a sliding button on the screen.

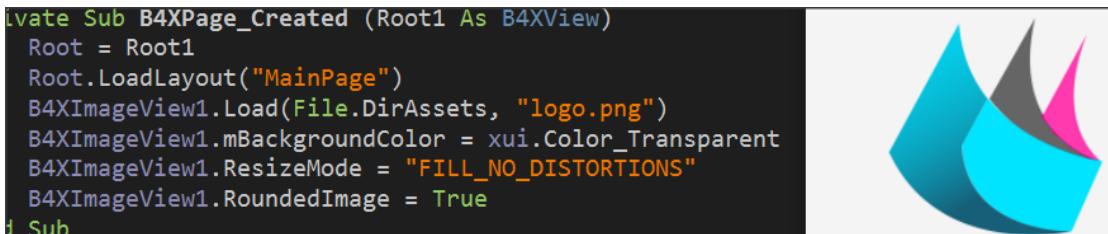
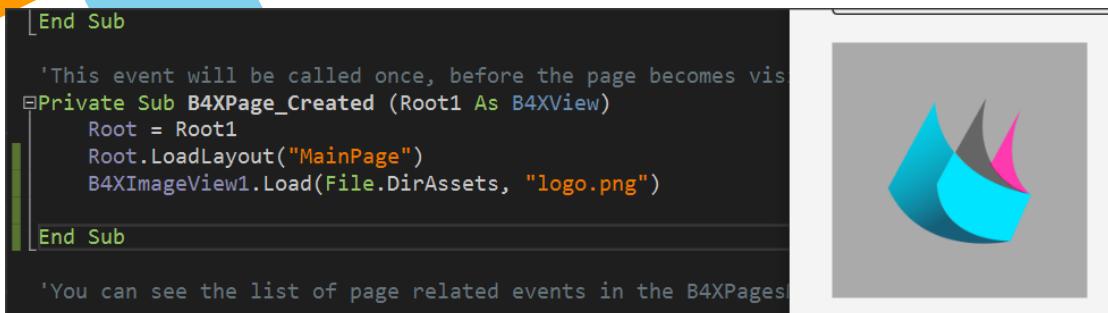


Picture 38 B4XSwitch

In the example image above when the returned value is true then a label displays the text "On" otherwise displays "Off".

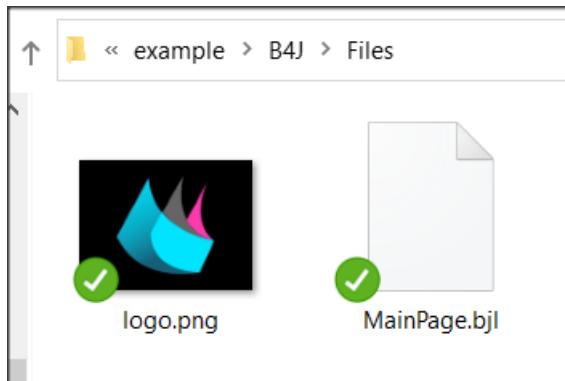
B4XImageView

B4XImageView displays an image and provides the ability to choose how the image is displayed either by code or by setting the object's options in Designer.



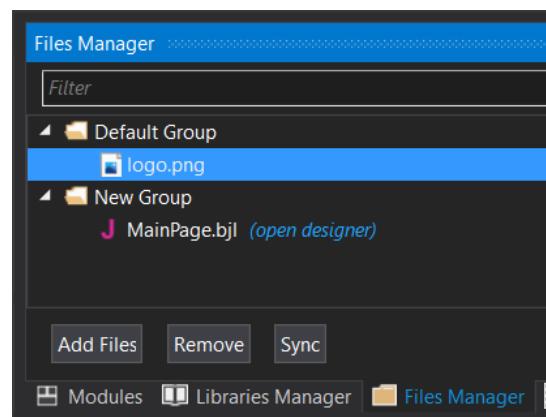
Picture 39 B4XImageView

In the B4XImageView.Load command, in addition to the image name, the folder where the image is stored must be declared first. In this case, the Files (Files.DirAsset) folder is declared as in the image below:



Picture 40 Files Folder (DirAsset)

This folder is not accessible for writing new files when the application is created. As the lesson/s continue we will refer in greater detail to the folders.



Picture 41 Files Manager

B4XDialogs

Dialog boxes are screens that popup to inform the user of an event or to get data from users without the need to create another B4XPage.

MsgBoxAsync

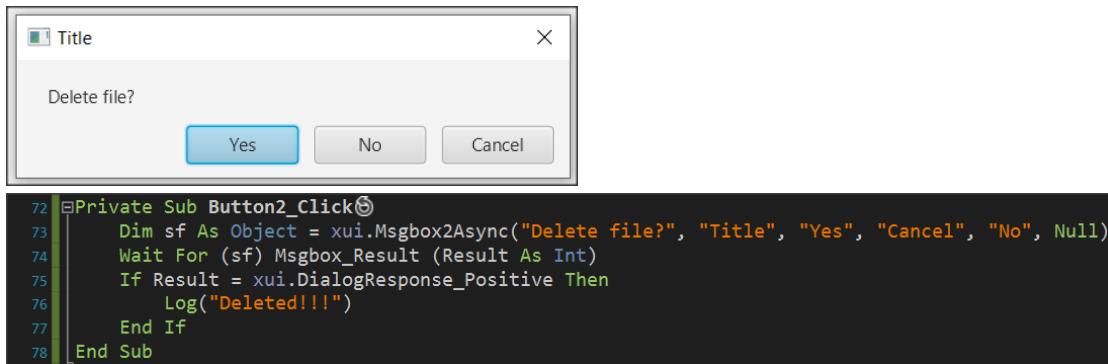
Already in previous lessons you used the command `xui.MsgboxAsync` which displays a simple message on the screen as shown in Picture 42. The command also displays an OK button to close the message and return to the program.



Picture 42 MsgBoxAsync

Msgbox2Async

Using `Msgbox2Async` is more complex and allows the programmers to display a message and then choose between three different functions by pressing a corresponding button.



Picture 43 MsgBox2Async

`Msgbox2Async` returns an object ("sf" in the example) that waits for a response event from user. The result will be one of these internal constants:

- **DialogResponse_Positive** (if "Yes" was pressed in the example)
- **DialogResponse_Negative** (if "No" key was pressed)
- **DialogResponse_Cancel** (if "Cancel" was pressed)

The above values are controlled by an if statement and appropriate actions are performed accordingly.

The words that appear on the action buttons can be changed but always have the same order as a "Positive", "Cancel", "Negative".

You can skip a button simply by writing a blank string "" at the corresponding point and the button will not appear in the dialog box.

The last parameter allows you to display an icon to the left of the message, and Null displays nothing.



Wait For

Wait For freezes the execution of an operation until an event is activated.

```
Wait For (<sender>) <event signature>
```

Where **sender** is the object for which an event is waiting to be activated and

Event signature the event that was activated where in the case of MsgBox2Async are the DialogResponse_Positive, DialogResponse_Cancel, DialogResponse_Negative.

The event is then checked with an if statement.

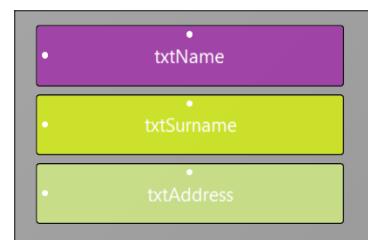
CustomDialog

Creating a Custom dialog box includes several steps that start with the Designer.



Picture 44 Steps to create Custom Dialog.

Continuing the first example create a new form in Designer and save under the name frmInsStudent. Also create corresponding variables for Text Fields from Generate Members.

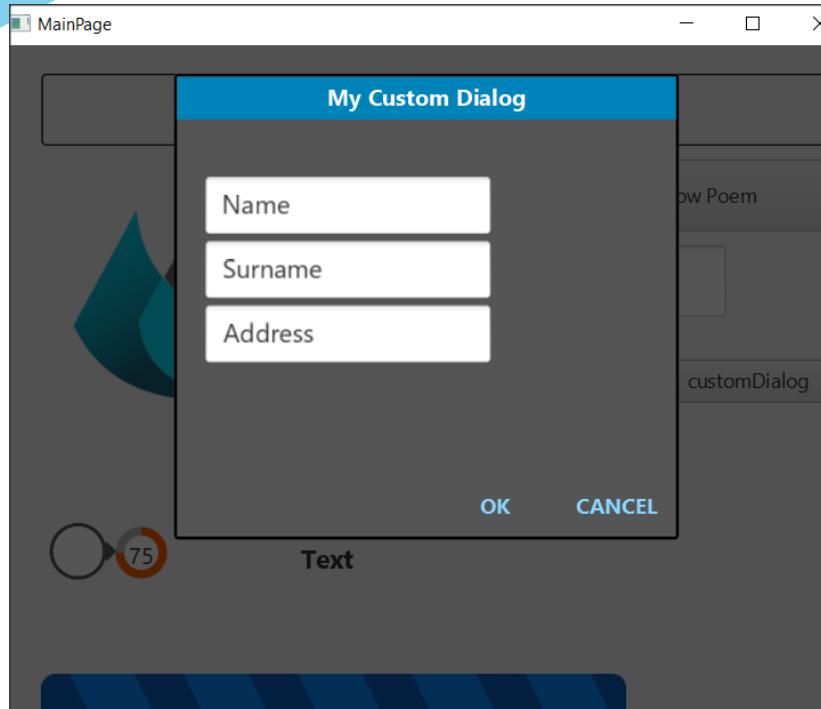


```
87 | Private Sub Button3_Click()
88 |     dialog.Initialize (Root)
89 |     dialog.Title = "My Custom Dialog" 1
90 |
91 |     Dim p As B4XView = xui.CreatePanel("")
92 |     p.setLayoutAnimated(0, 0, 0, 350dip, 250dip) 2
93 |
94 |     p.LoadLayout("frmInsStudent")
95 |     dialog.PutAtTop = True 'put the dialog at the top of the screen 3
96 |
97 |     Wait For (dialog.ShowCustom(p, "OK", "", "CANCEL")) Complete (Result As Int) 4
98 |     If Result = xui.DialogResponse_Positive Then
99 |         Log(txtName.text & " " & txtSurname.text & " " & txtAddress.text)
100 |     End If
101 | End Sub
```

Picture 45 Custom Dialog

1. Set an object named dialog type B4XDialog within globalSub, and then use the command “dialog.Initialize(Root)” to initialize it. “dialog.Title =” sets a title in the dialog box that you are about to create.
2. Set a pane in which to display the items on the form you designed and set its dimensions in pixels.
3. Load your form with the “p.LoadLayout” command where “p” is the pane you set before and then set to appear at the top above all other windows.

4. Wait for a button to click and check the results.



Picture 46 The dialog box

Templates

The B4X has ready-made templates for creating dialog boxes, some of which are:

- B4XDateTemplate
- B4XColorTemplate
- B4XLongTextTemplate

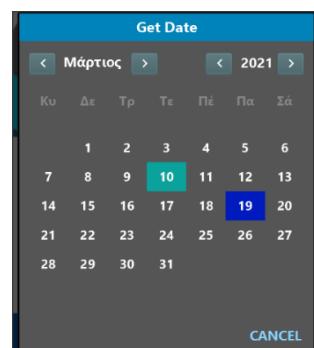


Teacher's tip.

Additional functions will be used in next chapters as concepts such as lists, dictionaries, etc. have not yet been discussed.

B4XDateTemplate

Creates a Dialog box to select a date. This box only needs a cancel button because when a date is selected it automatically returns the relevant value to the program. When the form opens, it has already marked the current date in a different color.



B4XDateTemplate based on a dialogue template that should be declared and initialized in the same manner as for Custom Dialogs

1. Initialize and set a title in the dialog. In the example initialization is in

```
108 | Private Sub btnDate_Click()
109 |     dialog.Title = "Get Date" 1
110 |     DateTemplate.Initialize
111 |     DateTemplate.MinYear = 2010 2
112 |     DateTemplate.MaxYear = 2030
113 |     'only CANCEL needed
114 |     Wait For (dialog.ShowTemplate(DateTemplate, "", "", "CANCEL")) Complete (Result As Int)
115 |     If Result = xui.DialogResponse_Positive Then
116 |         btnDate.Text = DateTime.Date(DateTemplate.Date)
117 |     End If
118 | End Sub
```

Picture 47 Create Date Dialog Box

Class_Globals Sub.

2. Specify whether you want years limits.
3. Wait for the date to be selected, and if selected, display it as a name on the "btnDate" or use it according to your needs.



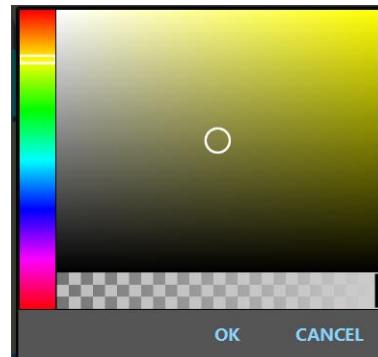
Remember

The date is returned to Ticks which represent milliseconds from 1/1/1970. These milliseconds are converted to a date with the "DateTime.Date" command.

B4XColorTemplate

B4XColorTemplate is similar to that for the calendar/date function. Once again, the code is waiting with the Wait For command to make a color selection, and when pressed OK returns a color number.

The example uses the color to change the background of the home page with the "Root.Color".



```
123 | Private Sub btnColor_Click()
124 |     ColorTemplate.Initialize
125 |     Wait For (dialog.ShowTemplate(ColorTemplate, "OK", "", "CANCEL")) Complete (Result As Int)
126 |     If Result = xui.DialogResponse_Positive Then
127 |         Root.Color = ColorTemplate.SelectedColor
128 |     End If
129 | End Sub
```

Picture 48 Create Color Template

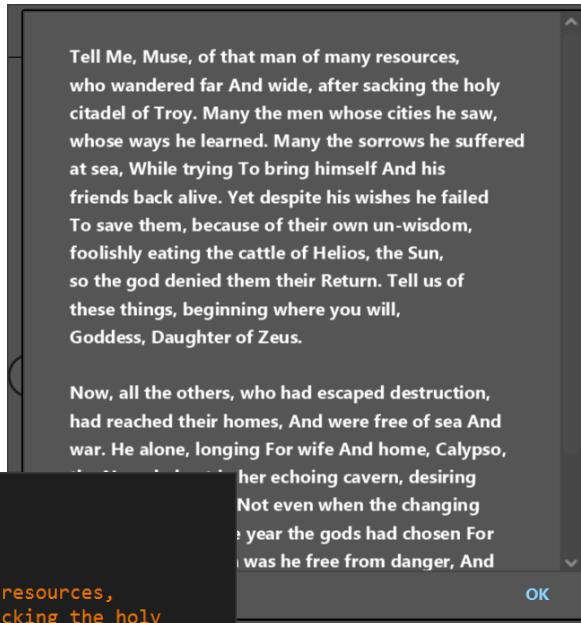
B4XLongTextTemplate

B4XLongTextTemplate displays a window on the screen to display long text. Also, it displays a navigation bar to scroll the text.

As in previous templates, set a variable of type B4XLongTextTemplate, and then after you initialize it you can set a size for the window that you are going to create with the "Resize" command.

Finally, set the Text property to

```
134 Private Sub btnReadPoem_Click
135     LongTextTemplate.Initialize
136     LongTextTemplate.Resize(500, 500)
137     LongTextTemplate.Text = "$"
138     Tell Me, Muse, of that man of many resources,
139     who wandered far And wide, after sacking the holy
140     citadel of Troy. Many the men whose cities he saw,
141     whose ways he learned. Many the sorrows he suffered
142     at sea, While trying To bring himself And his
143     friends back alive. Yet despite his wishes he failed
144     To save them, because of their own un-wisdom,
145     foolishly eating the cattle of Helios, the Sun,
146     so the god denied them their Return. Tell us of
147     these things, beginning where you will,
148     Goddess, Daughter of Zeus.
149     "$"
150     dialog.ShowTemplate(LongTextTemplate, "OK", "", "")
151 End Sub
```



the text you want to display and call "ShowTemplate".

Picture 49 LongTextTemplate



Exercises

1. Create a program that asks for two dates between 2000 and 2021 and shows the distance between them in days, months and years.

Tip:

```
DateUtils.PeriodBetween(date1, date2).Days  
DateUtils.PeriodBetween(date1, date2).Months  
DateUtils.PeriodBetween(date1, date2).Years
```

2. Create a program that reads the following customer information:

- a. Name
- b. Surname
- c. Phone
- d. Phone Account (in €)

Then create a button on the form that says payment and if pressed displays the message "Do you want to pay?" If answered yes set 0 to the variable corresponding to the Phone Account.

3. Create a program that for 5 books displays a summary of them in an appropriate window. Books must be selected from an equal number of buttons with book's images on them. The summaries and images of 5 books can be found in the supporting material of the exercise.
4. Create a form that contains the following items:

- a. Name
- b. Surname
- c. Phone

It also includes a switch that when the form is open has a light_Gray background color with text fields in White while when closed it has dark gray background with light_Gray text Fields. Also include a label to say day or night with black or white letters each.

Tip: Use JFX library to set floatTextFields colors

```
Private fx As JFX ' in Class_Globals  
lblDayNight.TextColor = fx.Colors.Black ' Where you want to change color
```

5. A water tank is 5 meters height and has a base of 3x3 meters. Make a program that,
 - A. continues reads the level height in meters and displays a bar to represent the height. Text field will not allow any text but only decimals values.
 - B. If water height is above 5 meters show message "Danger Overflow!".
 - C. If the level Is less than 0.5 meters display a message. "Warning there is not enough water in the tank.
 - D. In every change show the total amount of water in cubic meters.

Lesson 14 – Arrays

⌚ 4h

What students should know

- One dimensional Arrays
- Basic Operations with arrays
- MAX – MIN item
- Linear search
- Binary search
- Sorting with Bubble Sort
- Sorting with Selection Sort

A common problem in computer programming is managing a large amount of data. When a problem consists of 6-7 variables, it is easy to declare and use them. What happens when there is a need to use multiple similar data at the same time? For example, how can two hundred variables with names and grades be declared and how can a programmer manage so many data?

One of the solutions that computer science provides to said problem is the use of arrays. In general, the array term defines a set of data of the same type that is placed one after the other in computer memory. So, the developer can find them simply by moving from one location to another without specifying separate names.

Grades	98	76	86	45	32	77	56	99	34	71	47	82	69	88
0														13

Picture 50 Array named "Grades" and 14 places for grades

In Picture 50 you can see an example array with 14 student grades in a lesson. All points are placed in continuous positions and there is only one name (Grade). The programmer, to refer to memory locations in a array, should just indicate its name and then type, in parentheses, the number of the cell in which the data they need is located. Thus, for example, to display the first element of the array, they simply type "Grades(0)" where 0 is the first position of the array.

Declaring Arrays

An array is declared like other variables.

```
Private Grades(14) As Int
```

Here, Grades is the name of the array and the number in parentheses represents the number of positions in the array. Once an array is declared with a certain size, it cannot be changed within the code unless it is re-declared with a new size.



Functions in arrays

Insert items into an array.

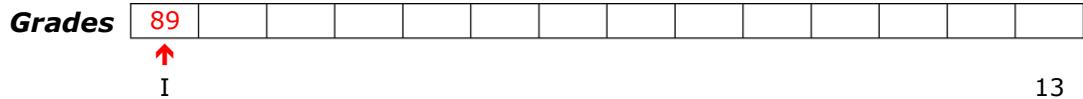
To insert an item into an array, you only need to assign a value to the corresponding place in the array. E.g.

```
Grades(0) = 89
```

The process can continue the same way, but it is easier to create a repeating (loop) process to fill the array. In the case of the Grade array in B4X it could be:

```
Private Grades(14) As Int
For i = 0 To 13
    Grades(i) = Rnd(1,100)
Next
```

The above code fills the array with random numbers from 1 to 100. Notice that the position measurement starts at number 0. The variable **i** identifies



Picture 51 Insert items into a array

the position in the array as the loop iterates and is called the **INDEX** of the array. Moving the index **i** with a repeat command you can access each location in the array.

A second way to insert items into an array is as follows:

```
Private Grades() As Int
Grades = Array As Int(19,43,12,65,23,87,45,65,87,23,56)
```

This size of the array is not specified in its statement, but during a fill by inserting items with the **Array command**.

It is obvious that you can have arrays of any type, for example strings, floats etc, but never mix up the types in an array.

Display items in a array.

Using the Log command, you can easily print an item in a array or the entire array using one iteration.

```
Log(Grades(0)) ' Shows the 1st item of Array Grades  
  
For i = 0 to 13  
    Log(i & ":" & Grades(i))  
Next
```

```
0: 26  
1: 95  
2: 72  
3: 17  
4: 82  
5: 76  
6: 72  
7: 10  
8: 91  
9: 79  
10: 19  
11: 86  
12: 28  
13: 30
```

The example above starts an iteration that uses an index (i) to display the current index value of the array.

Attempting to use an out-of-bounds index leads to a collapse of the program, so it is very important that you pay attention to the use of indexes and array boundaries.

Show items in reverse order

The following code shows the array items from end to beginning:

```
For i = 13 to 0 Step -1  
    Log(i & ":" & Grades(i))  
Next
```

```
13: 39  
12: 5  
11: 88  
10: 34  
9: 82  
8: 99  
7: 5  
6: 48  
5: 63  
4: 62  
3: 16  
2: 49  
1: 93  
0: 55
```

Generally, you can move your index as you want in a repeat display or use any items in the array you want.

to

Find Total and Average Array Items

The rules applicable to repetitive procedures for algorithmic techniques generally apply to arrays with few variations. Thus, the sum of the elements requires an extra variable that will hold the sum, which we usually call "sum", and a repeat in the array.

```
Private intSum As Int = 0  
Private fltAverage as Float  
For i = 0 to 13  
    intSum = intSum + Grades(i)  
Next  
fltAverage = intSum / 14  
Log(intSum & fltAverage)
```

Find Maximum and Minimum

Finding the maximum or minimum item of a array makes use of additional variables commonly called Max and Min, respectively. The first value in the array (here this is Grades) is set as the start value, and, then, the remaining values in the array are tested against this first value. If an element greater than Max (or less than Min) is found, Max (or Min) is replaced with the new item.



```

Private intMax, intMin As Int
intMax = Grades(0)
intMin = Grades(0)
For i = 0 To 13
    If intMax < Grades(i) Then
        intMax = Grades(i)
    End If
    If intMin > Grades(i) Then
        intMin = Grades(i)
    End If
Next
Log("Max = " & intMax)
Log("Min = " & intMin)

```

Search Algorithms

Searching for an item in an array refers to scanning it in search of an item that meets a specific condition.

In this unit, we will discuss serial and binary search algorithms.

Serial Search

Serial searching is the easiest but also the slowest way to search. It involves scanning all items in a array to find the item being searched. The following code shows the positions in the array that contain the key value.

```

'Find all positions with key value
For i = 0 To 999
    If Grades(i) = key Then
        Log("found in : " & i & "position")
    End If
Next

```

When it is necessary to get the first location in which a specified value is found, a logical variable (found) should be declared, the value of which we will reverse if the item is found.

```

Private found As Boolean = False
i = 0
Do While Not (found) And i <= 999
    If Grades(i) = key Then
        Log("found in : " & i & "position")
        found = True
    End If
    i = i + 1
Loop
If Not(found) then
    Log("Not found")
End If

```

Binary Search

Binary Search applies only to sorted arrays. The basic philosophy of the method includes examining the middle value. If the item, we are looking for

is smaller than the central one then the search continues on the upper half of the array. Contrarywise, if it is greater, the bottom half is searched. In the following example, there is an array containing Grades with 10 values sorted in ascending order.

Binary Search key value = 80									
1	47	47	47	47	47	47	47	47	47
2	58	58	58	58	58	58	58	58	58
3	62	62	62	62	62	62	62	62	62
4	69	69	69	69	69	69	69	69	69
5	74	74	74	74	74	74	74	74	74
6	79	79	79	79	79	79	79	79	79
7	80	80	80	80	80	80	80	80	80
8	83	83	83	83	83	83	83	83	83
9	88	88	88	88	88	88	88	88	88
10	95	95	95	95	95	95	95	95	95
	1	2	3	4					

- Initially the first and last cell of the array are entered in the Top and Bot variables. The center of the array is then determined as the quotient of the integer division (Top+Bot)/2.
- Check the Grades(Cent) with key and if it is smaller, the Bot is transferred above the Cent. Otherwise, the Top is transferred below the Cent.
- Repeat the steps above until the item is found or Top location to be larger than the Bot.

Sort

There are several sorting algorithms in an array that you can use. In this unit, we will discuss Bubble and Selection Sort.

Bubble sorting

Bubble sorting is a simple sorting algorithm that repeatedly steps through the array, compares adjacent elements, and swaps them if they are in the wrong order. The pass through the list is repeated until the array is sorted. The algorithm, which is a comparison sort, is named for the way smaller or larger elements "bubble" to the top of the list.

Example Bubble Sort

An array named Grades with 5 integer grades is given.

```
Private Grades() As Int
Grades = Array As Int(65,12,19,43,23)
```

1 st Pass									
1	65	65	65	65	65	65	65	65	65
2	12	12	12	12	12	12	12	12	12
3	19	19	19	19	19	19	19	19	19
4	43	43	43	43	43	43	43	43	43
5	23	23	23	23	23	23	23	23	23
	1	2	3	4					



Initially, the algorithm starts from the last position of the array and compares sequentially with the previous

1. The first comparison is made with the values of cells 5, 4 where Grades(5) is less than Grades(4) and thus the two cells swap values.
2. The next step compares cells 4 and 3 where Grades(4) is not smaller than Grade(3) and does not change anything in the array.
3. For positions 3 and 2 the values in the array do not change either.
4. And in the last step, the 2nd to the 1st position is compared and a swap occurs again.

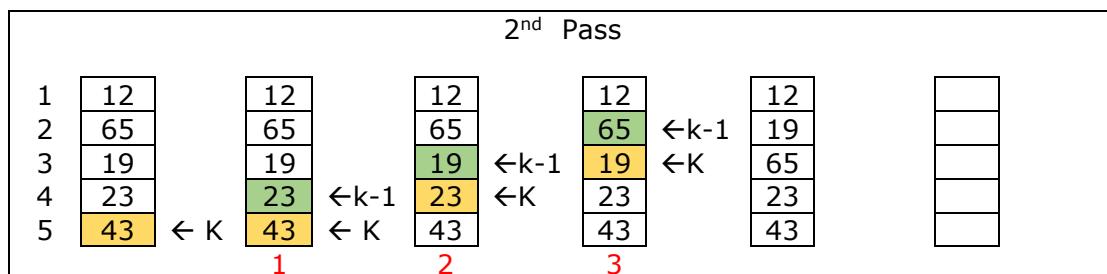
The first pass is implemented with the following code.

```

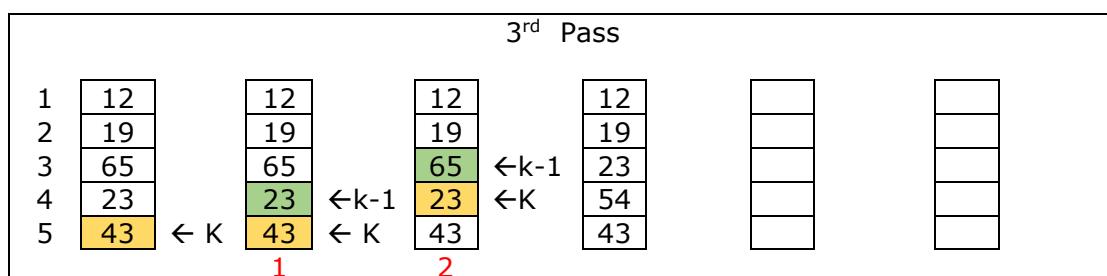
Private Grades() As Int
Private temp As Int
Grades = Array As Int(19, 43, 12, 65, 23)
For k = 4 To 1 Step -1
    If Grades(k) < Grades(k-1) Then
        temp = Grades(k)
        Grades(k) = Grades(k-1)
        Grades(k-1) = temp
    End If
Next

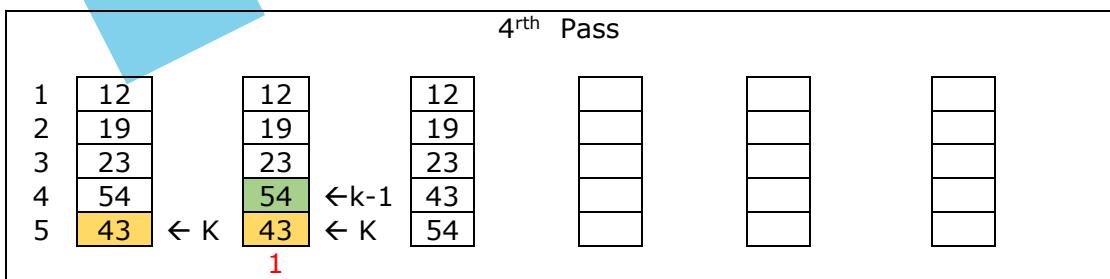
```

The smallest item has been transferred to the top of array.



In the second pass through the array, the same procedure is performed except that the loop counter size must be one less than the size of the previous pass.





Passes continue until the classification of the array is completed. Notice that each time fewer positions are checked since at each pass the smallest one rises to the surface (bubbles upwards).

Generally, these passages are as large as the size of the array -1. In the example for an array of 5 items, 4 passes were made. The completed code is as follows:

```

Private Grades() As Int
Private temp As Int
Grades = Array As Int(19, 43, 12, 65, 23)
For i = 1 To 4      'i counts the different pass
    For k = 4 To i Step -1
        If Grades(k) < Grades(k-1) Then
            temp = Grades(k)
            Grades(k) = Grades(k-1)
            Grades(k-1) = temp
        End If
    Next
Next

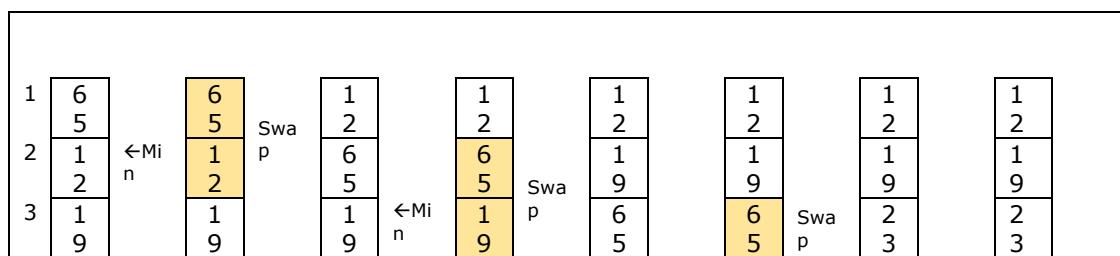
```

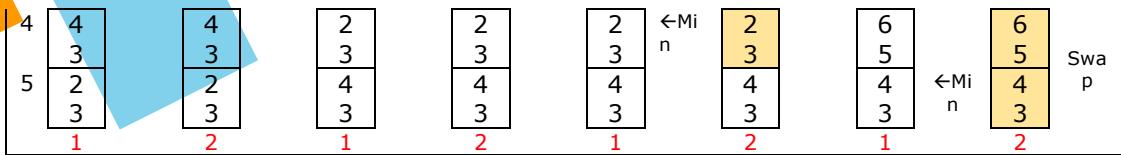
Selection Sort

The algorithm divides the array list into two parts: a sorted part of items which is built up from left to right at the front (left) of the array and a subarray of the remaining unsorted items that occupy the rest of the array. The algorithm proceeds by finding the smallest (or largest, depending on sorting order) element in the unsorted subarray, exchanging (swapping) it with the leftmost unsorted element (putting it in sorted order).

Implemented according to the steps below

1. Select the minimum item
2. Exchange of the minimum with the first item
3. Repeat steps 1 and 2 for the rest of the array items





```

Private Grades() As Int
Grades = Array As Int(65,12,19,43,23)
Private intMin, intMinPos As Int

For k = 0 To 4
    intMin = Grades(k)
    intMinPos = k
    For i = k To 4      'Find the minimum from position k to 5th
        If intMin > Grades(i) Then
            intMin = Grades(i)
            intMinPos = i
        End If
    Next
    Grades(intMinPos) = Grades(k)
    Grades(k) = intMin
Next

```

Exercises

1.

- Type a program that fills an array named **A(50)** with random integers from 1 to 100.
- Calculate and display the sum of array **A(50)** in even positions.
- If the sum of the first 25 positions in the array is equal to the sum of the last 25 items, show the message "Equal totals".
- If $A(1)=A(50)$, $A(2)=A(49)$, $A(3)=A(48)\dots A(25)=A(26)$, then displayed the message "Array symmetrical"
- Find the maximum value and the locations of the array that it is located in.
- Create a subprogram that sorts Array A
- Create a subprogram that accepts an array and an integer and applies binary search for that number to the array. Finally, return the location where the number was found or 0.
- Using the two previous subprograms, sort array A, and then search for item 67 and display appropriate messages whether it was found or not.

2.

- The average temperature per day for one year is stored in an array **Temperature(365)**. If you consider these temperatures to be integer values between 1 and 40 °C, find and display the frequency of each temperature.
- In the previous array **Temperature** find the second highest temperature of the year.

Lesson 15 – Lists

⌚ 2h

What students should know

- What is a list?
- Basic Operations with lists

A list is a set of nodes arranged linearly (one after the other). Each node contains, in addition to its data, a pointer pointing to the next node. The pointer of the last node does not point to a node and it is NULL.

Create a list

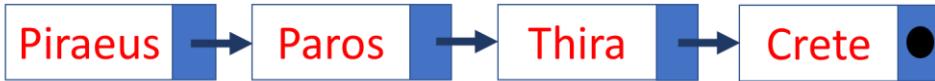
A list in B4X is declared as follows:

```
Private Islands As List  
Islands.Initialize
```

Where Islands is the name of the list created. In addition, a list to be used must be initialized using the initialize method.

Items are imported into the list using the Add method, which adds a new item to the end of the list.

```
Islands.Add("Piraeus")  
Islands.Add("Paros")  
Islands.Add("Thira")  
Islands.Add("Crete")
```



You can also insert items in a table at the end of the list with the AddAll command.

```
Islands.AddAll(Array as String("Piraeus", "Paros", "Naxos", "Crete"))
```



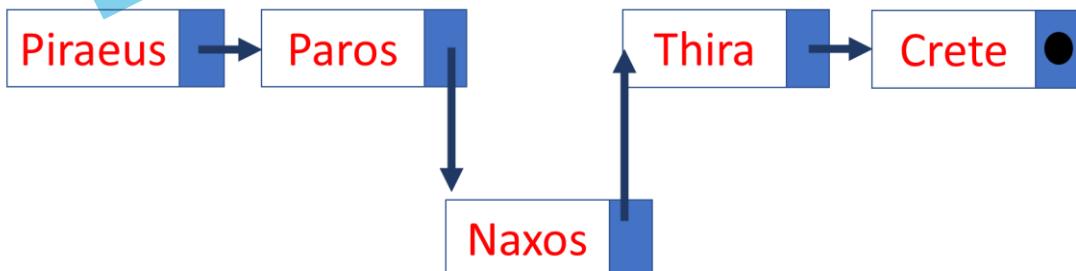
Remember

List items counts from 0. So, a list with 4 elements has the element(0) until element(3)



Insert an item in a specific location

To insert between two elements you can use the **InsertAt** property



```
Islands.InsertAt(2, "Naxos")
```

In the above example the island of Naxos is inserted between Paros and Thira.

Remove an item from a specific location

The RemoveAt command removes a list item from a specific location.



```
Islands.RemoveAt(1)
```

In the example above, Paros's island is removed from the list.

Change position value

With the Set command you can change the content of a location with another value that you enter.

```
Islands.Set(4, "Rhodes")
```



Here the value of 5th place changes to "Rhodes".

More commands in lists

1. List size.



```
Islands.Size
```

The example above returns the value 5.

2. Sort a list. You can sort a list in ascending or descending order with sort

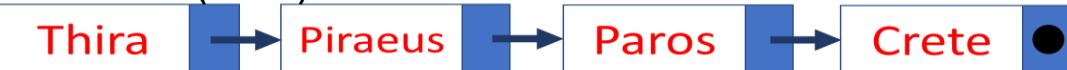
```
Islands.Sort(True) ' Sortby ascending order  
Islands.Sort(False) ' Sortin descending order
```



Islands.Sort(True)



Islands.Sort(False)



3. Empty List

A list empties with the Clear command

```
Islands.Clear
```

4. Move through a list

Moving within a list with a repetitive command

```
For i = 0 to Islands.Size - 1 'Remember it begins from 0  
    Log(islands.Get(i))  
Next
```

5. You can take advantage of the values in a list by assigning its values to other variables:

```
For i = 0 to Islands.Size - 1  
    Private isle As String  
    Isle = islands.Get(i)  
    Log(Isle)  
Next
```



6. Insert other types of items

A list can store items of any simple or complex type you want. For example, each item in a list could be an object or an entire table.

In the following example, each item in a list is an integer table:

```
'Create a list of arrays
Private StudentsGrades As List 1
StudentsGrades.Initialize

For j = 0 To 4
    Private Grades(5) As Int 2
    For i = 0 To 4
        Grades(i) = Rnd(1, 100) 'Create 5 random Grades For student
    Next
    StudentsGrades.Add(Grades) 3
Next

' Show Students Grades
For j = 0 To StudentsGrades.Size - 1
    Log("Student: " & j)
    Private stGrades(5) As Int = StudentsGrades.Get(4)
    For I = 0 To 4
        Log(stGrades(i)) 5
    Next
Next
```

1. The list is declared and initialized
2. For 5 elements in the list, an equal number of tables are created with random numbers between 1 and 100
3. Each table is placed at the end of the list with the Add command
4. For all items in the list, a table is filled by the item in the list
5. The table we got from each item is displayed.

Exercises

1.

- a. Create a list of Country and Capital Names. For example, you can use the following countries.

CUBA	HAVANA
CYPRUS	NICOSIA
CZECHIA	PRAGUE
EGYPT	CAIRO
KENYA	NAIROBI
MEXICO	MEXICO CITY
PERU	LIMA
VIETNAM	HANOI
PORTUGAL	LISBON

Source: <https://www.boldtuesday.com/pages/alphabetical-list-of-all-countries-and-capitals-shown-on-list-of-countries-poster>

- b. Display the names of countries starting with the letter "P"
- c. In an appropriate text field, enter name of a country and then search the list if that country exists and display the capital city.
- d. Make a button to show a dialog asking for new country and capital and add it to list.



Lesson 16 – Maps

⌚ 2h

What students should know

- What is a map?
- Basic Operations with maps
- For Each command

A map is a collection that contains pairs of keys and values. For example, records in a phone book or a dictionary can be a map. Other times it is also referred to by the name dictionary.



Figure 3 Yellow Pages Map

Each key is unique so you can refer to it uniquely. This means that if you add a key/value pair (entry) and the collection already contains an entry with the same key, the previous entry will be removed from the map.

The value can be any type from a simple variable to an object.



Anywhere Software

Create a map

A map is declared in B4X as below:

```
Private EnglishGreek As Map  
EnglishGreek.Initialize  
  
Private EnglishItalian As Map  
EnglishItalian.Initialize
```

Where **EnglishGreek** the name of the first map was created and **EnglishItalian** the second. In addition, a map to be used must be initialized.

Insert items into Map

Put method allows one to add keys/values to a map. For example, the two maps below use the put method.

English	Greek
Memory	Μνήμη
Screen	Οθόνη
Printer	Εκτυπωτής
Programming	Γλώσσα
Language	Προγραμματισμού

Map 1 **EgnlishGreek**

English	Italian
Memory	Memoria
Screen	Schermo
Printer	Stampante
Programming	Linguaggio di
Language	Programmazione

Map 2 **EnglishItalian**

The Put method is stated as follows:

<map name>.put(key, value)

The following example creates two Maps with key words from The English language and values in Greek and Italian.

```
Private EgnlishGreek As Map  
EgnlishGreek. Initialize  
  
Private EnglishItalian As Map  
EnglishItalian. Initialize  
  
EgnlishGreek. Put("Memory", "Memory")  
EgnlishGreek. Put("Screen", "Screen ")  
EgnlishGreek. Put("Printer ", "Printer")  
EgnlishGreek. Put("Programming Language", "ProgrammingLanguage"))  
  
EnglishItalian. Put("Memory", "Memoria")  
EnglishItalian. Put("Screen", "Schermo")  
EnglishItalian. Put("Printer", "Stampante")  
EnglishItalian. Put("Programming Language", "Linguaggio di  
Programmazione")
```

Use a map value

To use a value from a map, you only need to use the Get command.

<map name>. Get(Key As Object)

```
GRWord = EnglishGreek.Get("Screen")
Log(GRWord)      ' Shows Screen

ITWord = EnglishItalian.Get("Screen")
Log(ITWord) 'Shows Schermo
```

The variable GRWord retrieves the Value associated with the Key "Screen" from map EnglishGreek, and in the second example the variable ITWord retrieves the Value which corresponds to the Key "Screen" from map EnglishItalian..



Remember

The type of variable that the value accepts from the map must be the same type as the value. Key types must always be string or number.

If the key does not exist on the map, then Null is returned

```
ITWord = EnglishItalian.Get("Keyboard")
Log(ITWord) 'Shows null
```

Indexes in Maps

You can use indexes as in lists to access either the Key's name or the Value associated with a Key.

<map name>. GetKeyAt(Index As Int)

```
Key = EnglishItalian. GetKeyAt(2)
Log(key) ' Shows Printer
```

<map name>. GetValueAt(Index As Int)

```
Value = EnglishItalian.GetValueAt(2)
Log(Value) ' Shows Stampante
```

Also, the GetKeyAt and GetValueAt commands can be used in a iterative process to get all map values:

```
For i = 0 To EnglishGreek.Size - 1
    Log(EnglishGreek.GetValueAt(i))
Next
```

The above iteration shows the values of map EnglishGreek.



The command “for each”

One iteration command that has not been discussed so far is the command **for each**. This command creates an iteration that accepts items from a list of values such as a map for example.

```
For Each word As String In EnglishItalian.Values  
    Log(word)  
Next
```

The above iteration defines a variable which (the word in the example) will retrieve the value of the current position. You do not need a counter or step like for loop. Next steps give next map values and stops when map keys end.

```
For Each key As String In EnglishItalian.Keys  
    Log(key)  
Next
```

Displays the keys stored in the map.

To get both keys and values at the same time you can use the command for each as below:

```
For Each key As String In EnglishItalian.Keys  
    Log(key & EnglishItalian.GetValueAt(key))  
Next
```

Check key existence

If you are looking for a key, you can scan the table to find out if it exists or not, but it's easier and faster to use the **ContainsKey** command

<map name>. ContainsKey(Key As Object)

```
If EnglishGreek.ContainsKey("Keyboard") Then  
    Log("There is already an entry with this key !", "")  
Else  
    Log("There is not such a key!", "")  
End If
```

Delete Key and Empty Map

The Remove command deletes a key (and of course its value from a Map)

<map name>. Remove(Key As Object)

```
EnglishGreek. Remove("Memory")
```

Deleting all items from a Map is done with the Clear command.

```
EnglishGreek. Clear
```

Exercises

1. Create a map with Country names as keys and their capitals as values.

CUBA HAVANA

CYPRUS NICOSIA

CZECHIA PRAGUE

EGYPT CAIRO

KENYA NAIROBI

MEXICO MEXICO CITY

PERU LIMA

VIETNAM HANOI

PORTUGAL LISBON

Source: <https://www.boldtuesday.com/pages/alphabetical-list-of-all-countries-and-capitals-shown-on-list-of-countries-poster>

2. Add 3 countries with their capitals.
3. Display the names of countries and their capitals by using the **for each** command
4. Create a new map that contains the names of the capitals as keys and country's names as values.
5. In an appropriate text field, enter the name of a city and then display the country that owns the city.



Lesson 17 – Files

⌚ 2h

What students should know

- What is a file?
- File location in B4J
- File Methods

A File is a collection of data with similar content that is stored in the permanent space usually on the computer's hard disk. It is one of the most important features of programming languages since while all caching is done with central memory, after an application is completed its data should already be stored permanently.

engl_it.txt

Memory	Memoria
Screen	Schermo
Printer	Stampante

Picture 52 File engl_it.txt

Generally, the files are divided into databases and simple files which we will examine.

Storage Folders

Each operating system has many different folders to store applications' data or other files. In order for B4J to work with files independently of the operating system, it uses keywords that refer to a specific type of folder.

File.DirAssets

Includes files contained in the application files folder that have been added by B4J file management during the development phase of an application. These files are read-only and no file can be added while the application is running. Files are usually written by the developer so that after installing the application they can be copied to other folders for use.

xui.DefaultFolder

Returns the folder where the application data are stored (see File.DirData). You are first required to call the SetDataFolder command once before using it.

```
xui.SetDataFolder(AppName As String)
```



Anywhere Software

File.DirData

Returns the folder where the application data is stored and is suitable for creating files and storing data.

In a Windows environment, returns the "user data folder" that is usually in the path

`C:\Users\[user name]\AppData\Roaming\[AppName]`

On non-Windows operating systems, returns the folder where the application is installed

File.DirApp

Returns the folder where the application is installed. In Windows, this folder is usually Program Files and is read-only.

File.DirTemp

Returns the Temporary Files folder.

```
|Log(File.DirApp)
|Log(File.DirAssets)
|Log(File.DirTemp)
|Log(File.DirData("example1"))
|
Sub
```

Program started.
C:\Users\teacher1\NEXTCL~1\DOCUME~1\ANYWAR~1\LE0F89~1\example\example1\B4J\Objects
AssetsDir
C:\Users\teacher1\AppData\Local\Temp\
C:\Users\teacher1\AppData\Roaming\example1

Picture 53 Directories commands

You can use the above commands in combination to declare a new folder within the previous ones. For example,

```
Private strFolder As String = File.DirTemp & "lesson17\"  
Log(strFolder)
```

Displays C:\Users\<user name>\AppData\Local\Temp\lesson17\

Create folders

You can create a new folder with the

File.MakeDir (Parent As String, Dir)

```
File.MakeDir(File.DirTemp, "lesson17")
```

Creates a folder named lesson17 within C:\Users\teacher1\AppData\Local\Temp\ of the previous example.

Check file existence

Before you use a file, usually the first task to do is to check for existence. The command is

File.Exists (Dir As String, FileName As String)

and returns true or false depending on whether the file exists or not.

```
Private fn as String = "mydata.txt"  
If File.Exists(File.DirTemp, fn) Then
```

```

Log("File " & fn & " Exists")
Else
    Log("There is no File: " & fn)
End If

```

Create and write to a file

The command to create a file is:

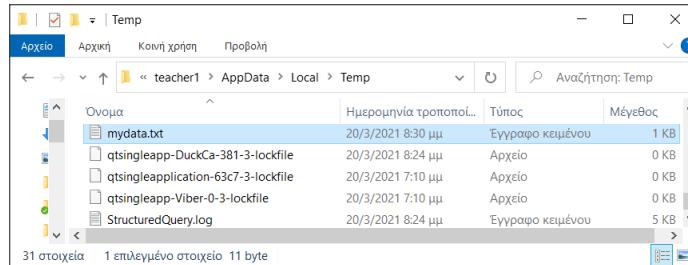
File.OpenOutput (Dir As String, FileName As String, Append As Boolean)

```

Private fn as String = "mydata.txt"
File.OpenOutput(File.DirTemp, fn, True)

```

If the file does not exist then it is created.



Picture 54 mydata.txt inside temp folder

The logical value True or False in the third parameter refers to whether the file is opened for write by erasing the older data or will add to existing records.

Writing and reading data

You can write or read data from one word to more complex structures such as lists and maps.

String-type variables

File.WriteString (Dir As String, FileName As String, Text As String)

```

Private fn as String = "mydata.txt"
Private msg As String = "Hello World"
File.WriteString(File.DirTemp, fn, msg)

```

Writes a string variable to the file. Attention the file is created from the beginning even if it already existed. So, its data are deleted.

For a file that already exists you can read its data and transfer it to a string variable with the command

File.ReadString (Dir As String, FileName As String) As String

```

Private fn as String = "mydata.txt"
Private strFileContent As String
strFileContent = File.ReadString(strFolder, fn)

```

Lists

To write a list to a file, use the command:

File.WriteLine(Dir As String, FileName As String, List As List)

```
File.WriteLine(strFolder, "mydata.txt", List1)
```

With the above command each item in the list is converted to string and a new row is created in the file.

To read a list from a file, use the command:

File.ReadList (Dir As String, FileName As String)

```
List1 = File.ReadList(File.DirRootExternal, " mydata.txt")
```

The command creates a new item in the list per line of the file.

Maps

An entire map is recorded in a file with the command

File.WriteLine(Dir As String, FileName As String, Map As Map)

```
File.WriteLine(File.DirInternal, "file.txt", map1)
```

The most common function is to create a configuration file for application.

A map is read with the command

ReadMap(Dir As String, FileName As String)

```
map1 = File.ReadMap(File.DirInternal, "file.txt")
```

The order of the items is not necessarily the same as the order of the file but this does not matter in a map.

Exercises

The Lesson17-ex1 program is given, which creates a list of football teams and a brief history of them.

1. Create a map with a key to the group name and Values containing its history.
2. Create a file that stores the above map named teams.txt
3. Create a string from all map keys and values. Use & CRLF to add a new line at the end of each line
4. Write string in a new file with name teams2.txt
5. Open with an external editor (such as notepad++) teams.txt and teams2.txt and examine their differences.



Lesson 18 – Complex Data Types, Files & Views

🕒 4

What students should know

- Type declaration
- Arrays of Type
- List of Type
- Map of Type
- KVS Files
- CustomListView
- B4XComboBox

Often a developer wants to group variables related to each other. For example, a student has a First Name, Last Name, Address, Phone, etc. One can of course create separate variables for each different element of the student but it is more convenient to create a new data type that contains all these values together.

The type statements

The described data type is called **type** and is stated as follows:

Type Student(LastName As String, FirstName As String, Address As String, PhoneNumber As String)

Notice that the keyword “Type” is first written, then a name is written for the new data type record that is created, and finally in parentheses all variables/elements included in the new type. Type declarations are always written in Class_Globals and it is Public by default.

In the student's example, the data Type statement looks like this:

```
Sub Class_Globals
    Type Student(LastName As String, FirstName As String,
                 Address As String, PhoneNumber As String)

End Sub
```

A new data type has now been created called Student and you can declare new variables based on this.

```
Sub Class_Globals
    Type Student(LastName As String, FirstName As String,
                 Address As String, PhoneNumber As String)

    Public Student1 As Student
End Sub
```



The “*Student1*” variable is now a Student-type variable, and to access its data you use the variable name with one period and then the name of the items included in the type statement e.g.

```
Student1.LastName = "Ioannidis"  
Student1.FirstName = "Alkinoos"  
Student1.Address = "Athens, Greece"  
Student1.PhoneNumber = "+303465854234"
```

You can create as many variables of the Student type as you want and assign each other.

```
Private Student2 as Student  
Student2 = Student1
```

Here the *Student2* variable now contains exactly the same data as the *Student1* variable.

Show Items

You can use the *Log()* command to display the contents of *Student1*, but the result will be something like the following:

```
Log(Student1)  
[IsInitialized=false, LastName=Ioannidis, FirstName=Alkinoos  
, Address=Athens, Greece, PhoneNumber=+303465854234]
```

To display or use all elements of a type, use the entire name along with the item.

```
Log(Student1.LastName & " " & Student1.FirstName)
```

It is usually helpful to create a routine that accepts a type variable and then displays or processes the type elements.

```
Private Sub LogStudent(st As Student)  
    Log(st.FirstName)  
    Log(st.LastName)  
    Log(st.Address)  
    Log(st.PhoneNumber)  
End Sub
```

Array of a type

You can also create arrays of Type where it will include more students. For example,

```
Sub Class_Globals  
    Type Student(LastName As String, FirstName As String,  
                Address As String, PhoneNumber As String)  
  
    Public Students(10) As Student  
End Sub
```

The Students(10) statement creates an array of 10 Students. Each item can be used with a for loop or with a separate reference to each with its index.

Students(0). LastName = "Paul"

Students(0). FirstName = "Belmond"

List of Type

A list can contain variables of Type.

```
Sub Class_Globals
    Type Student(LastName As String, FirstName As String,
                 Address As String, PhoneNumber As String)

    Public listStudents As List
End Sub

Private Sub B4XPage_Created (Root1 As B4XView)
    ...
    ...
    listStudents.Initialize
    listStudents.Add(Student1)      ' Add Student to list

    'Get list items and log
    For i = 0 To listStudents.Size-1
        Private st As Student
        st = listStudents.Get(i)
        LogStudent(st)
    Next
End Sub

Private Sub LogStudent(st As Student)
    Log(st.FirstName)
    Log(st.LastName)
    Log(st.Address)
    Log(st.PhoneNumber)
End Sub
```

Maps of Type

Using maps to store type data is also possible as long as you have a unique key for each type variable you enter.

In the case of Students it could be for example an ID number, a registration number, an email account, etc.

```
Sub Class_Globals
    Type Student(id As String, LastName As String,
                 FirstName As String, Address As String, PhoneNumber As String)

    Public mapStudents As Map
```



```

End Sub

Private Sub B4XPage_Created (Root1 As B4XView)

    Student1.ID = "FXA47345S3"
    Student1.LastName = "Ioannidis"
    Student1.FirstName = "Alkinoos"
    Student1.Address = "Athens, Greece"
    Student1.PhoneNumber = "+303465854234"

    mapStudents.Initialize
    mapStudents.Put(Student1. ID, Student1)

End Sub

```

In this way it is now easy to find a student with a key ID number by calling the Get method.

Save to KVS files

The files discussed in Chapter 17 refer to text files that often do not serve to store complex structures such as type declarations, Lists, and Maps. Another type of file supported by B4J is key value store (KVS) files.

The way they work is similar to that of Maps. All you need is a key and the structure you want to save. KVS files essentially hide a database, but the important thing is that the developer doesn't need to know any of them as long as he uses the appropriate methods.

The commands in the KVS files are as follows:

Declare a KVS file

To declare a KVS file, you must use the **KeyValueStore** library to which you can add to your project from the Libraries tab.

Then create a **KeyValueStore** variable.

```
Private kvsFile As KeyValueStore
```

Initialize the KVS file

The Initialize method defines the folder in which the file will be saved and the name it will have.

```

File.MakeDir(File.DirTemp, "lesson18")

kvsFile.Initialize(File.DirTemp & "lesson18", "kvsData.dat")

Log(File. DirTemp & "lesson18")

```

The above commands create a folder named lesson18 within the temp folder, and a file named "kvsData.dat" and finally displays the path of the file on the Log screen.



Remember

The file to be created cannot be opened with another viewer (for example notepad++) but must always be used by the program you

Insert items into a KVS file

The Put method is used to write the data into a KVS file. Each insertion that is **made** is called a Record. A prerequisite is that you have decided on a unique key so that you can refer to the record later. The following example inserts the Student1 with the student's ID number as a key.

```
kvsFile.Put(Student1.ID, Student1)
```

This way you can write any data type such as Lists, Maps, Strings, simple variables (numbers), types, and tables (only tables typed Byte or Object), as well as combinations (one list of maps for example).

Type declarations should be always be declared in the B4XMainPage module.

Maps can also be stored using the PutMapAsync method. It is also the best way to save Map elements since it enters the Map elements key as the key of the record.

```
kvsFile.PutMapAsync(mapStudents)
```

Retrieving items from a KVS file

The Get method retrieves an item from a KVS file. The returned value is an object. In other words, be careful to assign the returned value to a variable of the same object type.

The following example reads the record of a Student from a KVS file.

```
Student3 = kvsFile.Get("FS23534X21")
LogStudent(Student3)
```

The value "FS23534X21" is the key to the record and the subroutine LogStudent has described previously retrieves a Student and displays its contents with the Log () command.



Remember

If the key does not exist, then there is a problem with the operation of the program. The existence of the record must be checked before you assign it to a variable. This can be done with the ContainsKey method.



You can read an entire map using the **GetMapAsync** method. This command accepts a list of keys as a parameter and returns a map with the keys and their corresponding values.

```
Log("Show keys")
Private keys As List = kvsFile.ListKeys
For i = 0 To keys.Size-1
    Log(keys.Get(i))
Next
```

Finally, following the above code, GetMapAsync becomes:

```
Wait For (StudentFile.GetMapAsync(keys)) Complete (mapSt As Map)
```

To write a map in a KVS file use:

```
Wait For (StudentFile.PutMapAsync(keys)) Complete (Success As Boolean)
```

Check the existence of a record

To check the existence of a key in a KVS file, you must use the ContainsKey method.

```
If kvsFile.ContainsKey("FS23534X21") Then
    Student3 = kvsFile.Get("FS23534X21")
Else
    Log("Wrong id key")
End If
```

ContainsKey returns True after it finds the key in the file.

Delete KVS file content

The Remove method deletes a key along with its value from a KVS file, and the DeleteAll method deletes all data.

```
kvsFile.Remove ("FS23534X21")
kvsFile.DeleteAll
```



Remember

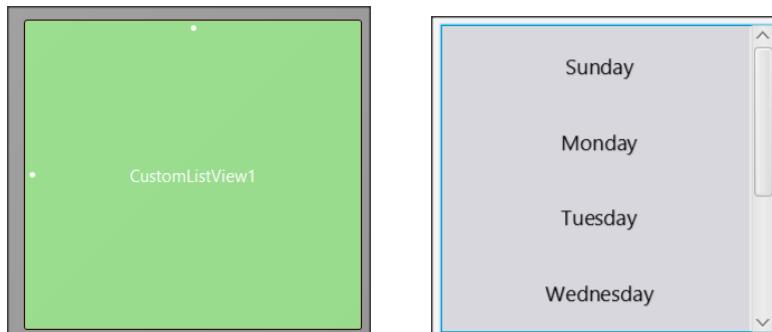
It's always a good practice to close file you don't need any more with method.close.

More Views

Two important views that will help to display or select elements from complex types, lists, and maps structures are **CustomListView** and **B4XComboBox**.

CustomListView

Creates a list of items that you can select by clicking the mouse.



The returned value is a value that you specified at the time ListView was created.

```
Sub Class_Globals
    Private Root As B4XView
    Private xui As XUI
    Private lstItems As List

    Private CustomListView1 As CustomListView 1
    Private lblDate As Label
End Sub

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

    lstItems.Initialize
    lstItems.AddAll(Array As String("Sunday", "Monday",
    "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday")) 2

    For i = 0 To lstItems.Size-1
        CustomListView1.AddTextItem(lstItems.Get(i), i) 3
    Next

End Sub

Private Sub CustomListView1_ItemClick(Index As Int, Value As Object) 4
    lblDate.Text = Value
End Sub
```

1. Declare a CustomListView variable



2. Create a list structure that contains the items that will appear in CustomListView
 3. Scan the entire list and each item is placed in CustomListView1 using the **AddTextItem** method.
- AddTextItem** accepts and displays a value you want to display. The above example corresponds to the values from 0 to 6 for the days from Sunday to Saturday.
4. When an item in the list is clicked, the **_ItemClick** event triggered and adds an index order number corresponding to. The example shows the index of the item that was clicked on the `lblDatetag`.

Additional methods of CustomListView are the following:

- **Clear As String**
Deletes all CustomListView components
- **GetValue (Index As Int) As Object**
Returns the value for the index value specified in the parenthesis.
- **Size As Int [read only]**
Returns the number of items.

Mark a list item

As mentioned earlier when clicking an item in a list, the event `_ItemClick` is triggered. In order for this item to remain marked, the developer must either use some of the ready-made tools provided by the language (e.g. the CLVSelections library) or write his own code. The following algorithm assumes that you have set a variable in Class_Global named `selectedItem` of type Int.

```
Private Sub CustomListView1_ItemClick (Index As Int, Value As Object)
If selectedItem = -1 Then
    Private p As B4XView = CustomListView1.GetPanel(Index) 1
    p.GetView(0).Color = xui.Color_Blue
    selectedItem = Index
Else
    Private p As B4XView = CustomListView1.GetPanel(selectedItem) 2
    p.GetView(0).Color = xui.Color_White
    If selectedItem = Index Then
        selectedItem = -1
    Else
        Private p As B4XView = CustomListView1.GetPanel(Index) 3
        p.GetView(0).Color = xui.Color_Blue
        selectedItem = Index
    End If
End If
End Sub
```

The original value of `selectedItem` was set to -1 where it means that nothing is selected. The code uses two commands:

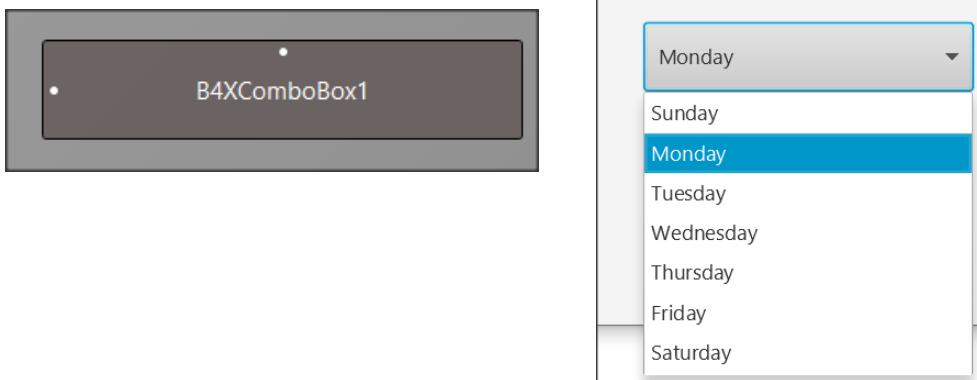
```
Private p As B4XView = CustomListView1.GetPanel(Index)
p.GetView(0).Color = xui.Color_Blue
```

Each item in a list is created by the language within a box called a Panel. You can set the color by accessing the box using the GetPanel(Index) method where Index equals the current value of the line that was clicked. Then the p.GetView(0).Color defines the color the developer wants to set. Then:

1. If an item in the list is clicked, the routine checks the value of the selectedItem, and if that is -1 then sets a Blue background color in the clicked line and sets selectedItem to the value of the Index that gives the event _ItemClick
2. If selectedItem already has a value, a white color is set as the background in the box, and then there are two cases
 - a. If Clicked the already selected item, the item stops being selected and selectedItem becomes -1
 - b. If Clicked another item, the new item gets blue color and selectedItem gets the value of the Index.

B4XComboBox

B4XComboBox displays a drop-down list of items. The user can click one of them and select it. Unlike CustomListView, the returned value is always a number for the order in which the item is placed in the ComboBox, so the developer must then be able to map the value to a structure.



```

Sub Class_Globals
    Private Root As B4XView
    Private xui As XUI
    Private lstItems As List
    Private B4XComboBox1 As B4XComboBox
    Private lblCmbDate As Label
End Sub

Private Sub B4XPage_Created (Root1 As B4XView)
    Root = Root1
    Root.LoadLayout("MainPage")
    lstItems.Initialize
    lstItems.AddAll(Array As String("Sunday", "Monday", _
    "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"))
    B4XComboBox1.SetItems(lstItems)
End Sub

Private Sub B4XComboBox1_SelectedIndexChanged(Index As Int)
    lblCmbDate.Text = Index
End Sub

```

1. Declare a type **B4XComboBox** variable
2. Create a list structure that contains the items that will appear in B4XComboBox
3. Place the items in the list in B4XComboBox. **Caution** does not require iterative structure as in CustomListView
4. If an item is selected, the **_SelectedIndexChanged** event is triggered and returns the index of the selected item.

Exercises

1. Create a type named **Customer** and variables

ID, FirstName, LastName, Phone

2. Create a list of Customers with the following items

Id	FirstName	LastName	Phone
A3473	John	Smith	4563454
B1753	Selim	Al Huarizmi	6532578
C6544	Mateo	Sandor	7345346
C6323	Lucía	Graham	1231345
F1462	Noam	Bell	6978323

3. Save the list to a KVS file
4. Build a button that when clicked fills a CustomListView with customers information.
5. Build an insert button that, when clicked, displays an appropriate DialogBox that prompts for new customer information and add them into CustomListView.
6. Save the new client to the KVS file as well.
7. When a customer is clicked display a confirmation message for deleting customer and in case of positive feedback delete the customer from CustomListView and KVS File.



Lesson 19 – Final Project

⌚ 10h

What students should know

- Use previous knowledge to create your final Project.



Dear Teacher

The solution proposed relies on knowledge gained during the course. You can extend the questions as you want depending on your class level and available time.

To study how a school library can be created you can read the book "Setting Up and Running a School Library by Nicola Baird" here <https://files.Eric.Ed.Gov/Fulltext/Ed536911.Pdf>

The names of the books and authors come from the Gutenberg Project https://www.gutenberg.org/ebooks/offline_catalogs.html

Publishers' names and release dates are random as well as students' names.

Prokopis Leon, pliroforikos[at]gmail.com

Create a School Library Application

A school library has a set of books which it lends to students in the three classes. The books are placed on shelves numbered from 1 to 60.

For each book, the following information is recorded:

- Code
- Title
- Author
- Release Year
- Publisher
- Shelf

For students, the following are recorded:

- Registry Number
- Name
- Surname
- Class
- Phone Number



Anywhere Software

Make the application described below:

Files

Two text files named are given books.txt and students.txt (inside the materials folder).

Some of the file elements are:

books.txt

```
30;The Life of Abraham Lincoln;Henry Ketcham;1866;New Public publ.;54  
31;Christopher Columbus; Mildred Stapley;1954;Cider publ.;43  
32;The Adventures of Ferdinand Count Fathom; Tobias Smollett;1982;Orange punl.;32  
33;Tales of the Jazz Age;F. Scott Fitzgerald;1944; Gutenberg publ.;5  
34;The Old Stone House;Anne March;1904;Orange punl.;50
```

Each book is described in the following fields which we distinguish from each other with the character ";":

- Id
- Title
- Author
- Year
- Publisher
- Shelf

students. txt

```
1001;Jude;Segers;A;7900209  
1002;Desire;Cid;A;7047635  
1003;Madelyn;Pittard;A;9011036  
1004;Lorita;Tomczak;A;6677490  
1005;Lynwood;Posey;A;9014379  
1006;Nella;Felps;A;8423818
```

Each student is described in the following fields which stand out from each other with the character ';':

- Id
- First Name
- Last Name
- Class
- Phone Number

Implement a function that inserts the above text files into Maps with the names mapBooks and mapStudents as the keys in the first column of each file where it represents the Id of the book and the student respectively. The values of each item will be a Type as described below.



Teachers tip

Before you can read the files you need to read all the content like a string and then check its characters one by one and place them in a table of items that you will later write into the Maps. A string is checked using the `<string>` method. **CharAt(index)**.

Example Of Student Map

Key	Type				
	Id	FirstName	LastName	Class	Phone
1001	1001	Jude	Segers	A	7900209
1002	1002	Desire	Cid	A	7047635
1003	1003	Madelyn	Pittard	A	9011036

Data

Types

1. Create Type Book

Type Book with elements:

- Id
- Title
- Author
- Year
- Publisher
- Shelf

Functions:

- Insert Book
- Display/Select Book/s
- Delete Book

2. Create Student Type

Properties:

- Id
- First Name
- Last Name
- Class
- Phone Number

Functions:

- Insert Student
- Display / Select Student
- Delete Student

Each student who has borrowed books has their own KVS file with name "ID number.dat". For example, the student with ID 21 has a file named "21.dat". The file writes the books borrowed by the student using a map structure. For example, if you have borrowed two books you should have a Map of the following structure:

("11", "03/27/2021")

("14", "04/01/2021")



Where the first number is the ID of the book borrowed and is also the key of the map while the borrowing date is saved as a value.

Screen Design and Functions.

The program must have a central menu of buttons to call appropriate administrative screens. Specific:

1. Book.

Includes a CLV list (CustomListView) of library books to be loaded from the books.txt and book insert and delete buttons.

When the Insert button is pressed, create a dialog asking for new book items to be inserted into the file.

When the Delete button is pressed, delete the book selected from the list.

2. Student

Includes a CLV list of school students that will be loaded from the students.txt file and insert and delete buttons.

When the Insert button is pressed, create a dialog asking for new student information and store it into students.txt.

When the Delete button is pressed, delete that student from the list together the books they borrowed assuming they have returned them.

3. Lending

The lending screen includes a ComboBox containing the student's name and a CLV list of library books.

There is also a button Lend when its' click triggered:

If there is a student's file

It will load the entire map from the file

It will add a new key with the ID of the book borrowed and set the date variable for the borrowing date.

It will re-save the map

If there is no student file

It will create the file

It creates a Map with the borrowing information(ID and date)

It will save the file

Consider that there are as many copies as you need from each book.

4. Return

The lending screen includes a B4XcomboBox containing the student's names and a CLV list of books borrowed. Also, there is a button called "Return".

If return_click triggered then it will delete this book from the student's list as well as from the student's file.

If he hasn't borrowed another book, program will delete and lending record file too.



Cheat Sheet

1. How to declare a type

```
Sub Class_Globals
    Type Student(LastName As String, FirstName As String,
                  Address As String, PhoneNumber As String)

    Private Student1 As Student
End Sub
```

2. How to read a file on a string

```
stFile = File.ReadString(File.DirAssets, "students.txt")
```

3. Check character by character a string that contains a info and place it in a list of tables.

```
Private i As Int = 0
For j = 0 To stFile.Length-1
    If stFile.CharAt(j) <> ";" And stFile.CharAt(j) <> CRLF Then
        stud(i) = stud(i) & stFile.CharAt(j)
    Else if stFile.CharAt(j) = ";" Then
        i = i + 1
    else if stFile.CharAt(j) = CRLF Then
        i = 0
        retList.Add(stud)
        Private stud(5) As String
    End If
Next
```

4. Convert a list of tables to a map

```
For i = 0 To lst.Size-1
    Private stud(5) As String
    stud = lst.Get(i)           ' Create an array from list item
    Private st As Student      ' Student is type declaration
    st.Initialize
    st.ID = stud(0)            'Put every array item into type
    St.FirstName = stud(1)
    St.LastName = stud(2)
    St.Cls = stud(3)
    St.Phone = stud(4)
    St.Borrowed = 0
    mStudent.Put(stud(0), st)   ' Insert type into map
Next
```

5. Place spaces In as string to increase its length (size) up to a number.

```
Do While s1.Length <= 5
    s1 = s1 & " "
Loop
```

6. Select – Deselect an item from a CLV List.

Each item in a list is created by the language within a box called a Panel. You can set the color by accessing the box properties using the GetPanel(Index) method where Index the current value of the line that was clicked then:

- If an item in the list is clicked, the subroutine checks the value of selectedItem, and if that is -1 then sets a Blue background color in the clicked line and sets selectedItem to the Index value that gave the _ItemClick event
- If selectedItem already has a value, a white color is set as the background in the box, and then there are two cases
- Click the already selected item, in which case the item stops being selected and selectedItem becomes -1
- Click another item in which case selectedItem gets the value of index.

```
Private Sub clvBooks_ItemClick (Index As Int, Value As Object)
    If selectedItem = -1 Then
        Private p As B4XView = clvBooks.GetPanel(Index)
        p.GetView(0).Color = xui.Color_Blue
        selectedItem = Index
    Else
        Private p As B4XView = clvBooks.GetPanel(selectedBook)
        p.GetView(0).Color = xui.Color_White
        If selectedBook = Index Then
            selectedItem = -1
        Else
            Private p As B4XView = clvBooks.GetPanel(Index)
            p.GetView(0).Color = xui.Color_Blue
            selectedItem = Index
        End If
    End If
End Sub
```

7. Load items from KVS File to map

The command must run within Wait For.

```
Wait For (StudentFILE.GetMapAsync(StudentFILE.ListKeys)) Complete (mapSt As Map)
```

The GetMapAsync command returns a map with the items. First declare mapSt As Map and initialize it.

8. Save items to a map's KVS File

```
Wait For (StudentFILE.PutMapAsync(mapSt)) Complete (Success As Boolean)
```

9. Delete a file from a folder

```
File.Delete(File.DirTemp, <file name>)
```



Lesson 20 – From B4J to B4A

⌚ 2

What students should know

- Transfer a B4J project to B4A

Congratulations!

Having completed the course's lessons up to this point you can say that you have acquired a good knowledge of the B4X language. What you've learned so far it is only the beginning of the journey of the "art" of programming.

In addition, the B4X language as discussed in previous chapters can be used to build mobile applications on both Android and IOS. With minimal changes and without having to learn new commands you can transfer an application written in Windows with B4J to Android and the B4A language.

In this last chapter you will deal with the conversion of a program (given ready) into a program for Android.



Picture 55 B4X Languages

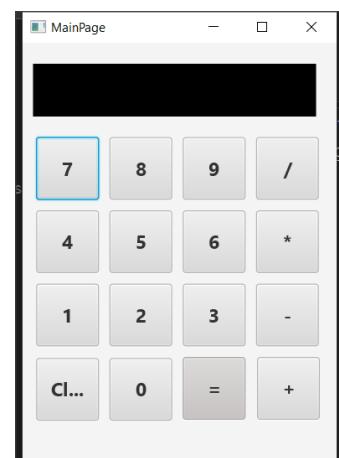
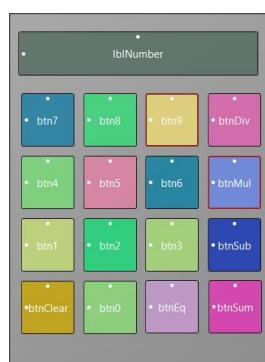
Application description in B4J

The application is a simple calculator that performs basic operations.

Designer

The app consists of 16 different buttons and a label that displays the numbers.

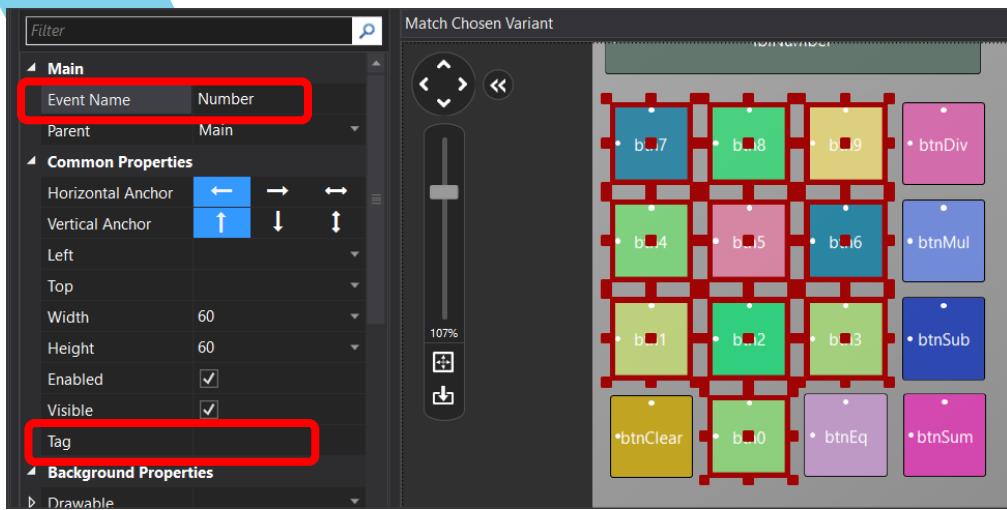
Ten of the buttons are the numbers from 0 to 9 and the rest are the basic operations, equality and clear.



Picture 56 Simple Calculator

In programs where you need to manage multiple buttons, in order not to use separate events for each event_click , it is possible to group the events_click.





For example, after selecting all the buttons listed in the numbers from 0 to 9, then set the "Event Name" properties, for example Number. Now all buttons have a common click event named Number_Click.

```
Private Sub Number_Click
    Dim b As Button
    b = Sender
    Log(b.Tag)
    If done Then
        lblNumber.Text = 0
        done = False
    End If
    lblNumber.Text = lblNumber.Text & b.Tag
End Sub
```

Also, for you to have a different value from each button you need to update the "tag"property. Assign a value to each different number. For example, in btnNumber1, set "tag" to 1, in btnNumber2 set "Tag" to 2 and so on.

Now in the event take advantage of the Tag value to enter a new number.

Sender assigns the clicked button to variable b.

The program code is as follows. Notice that no buttons have been declared as variables:

```
Sub Class_Globals
    Private Root As B4XView
    Private xui As XUI
    Private lblNumber As Label
    Private fltNumber1, fltNumber2 As Float
    Private operation As String
    Private done As Boolean      'When true an operation just finished
End Sub

Public Sub Initialize
End Sub

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

```

lblNumber.Text = ""
done = False
End Sub

'When Number button clicked add to string lblNumber new Number
'from tag
Private Sub Number_Click
    Dim b As Button
    b = Sender
    If done Then
        lblNumber.Text = 0
        done = False
    End If
    lblNumber.Text = lblNumber.Text & b.Tag
End Sub

'When Clear Button clicked clear all numbers and operations
Private Sub btnClear_Click
    lblNumber.Text = 0
    fltNumber1 = 0
    fltNumber2 = 0
End Sub

'When an operation button clicked set operation string to tag
'of clicked operation button
Private Sub operation_Click
    Dim b As Button
    b = Sender
    operation = b.Tag
    fltNumber1 = lblNumber.Text
    lblNumber.Text = 0
End Sub

'When button "=" clicked check operation string and do the operation
Private Sub btnEq_Click
    fltNumber2 = lblNumber.Text
    If operation = "+" Then
        lblNumber.Text = fltNumber1 + fltNumber2
    Else If operation = "-" Then
        lblNumber.Text = fltNumber1 - fltNumber2
    Else If operation = "*" Then
        lblNumber.Text = fltNumber1 * fltNumber2
    Else If operation = "/" Then
        lblNumber.Text = fltNumber1 / fltNumber2
    End If
    fltNumber1 = lblNumber.Text
    done = True
End Sub

```



Transfer the app to B4A and Android

Already when you start an application the appropriate folder for B4A and B4i has already been created even though you are not using it yet. These will be inside the relevant Files folder inside the project's home folder.

Documents > AnywareSoftware > Lesson20 - From B4J to B4A > Lesson20 - Solution > B4A >				
Όνομα	Ημερομηνία τροποποι... Διάστημα	Τύπος	Μέγεθος	Εργαλεία
 Files	31/3/2021 2:30 πμ	Φάκελος αρχείων		
 Lesson20 - Solution.b4a	31/3/2021 2:30 πμ	B4A Source Code	3 KB	
 Lesson20 - Solution.b4a.meta	31/3/2021 2:30 πμ	Αρχείο META	1 KB	
 Starter.bas	31/3/2021 2:30 πμ	Αρχείο BAS	2 KB	

To start the transfer, you will need to install the B4A IDE suite. For instructions on installation, see the link <https://www.b4x.com/b4a.html>.

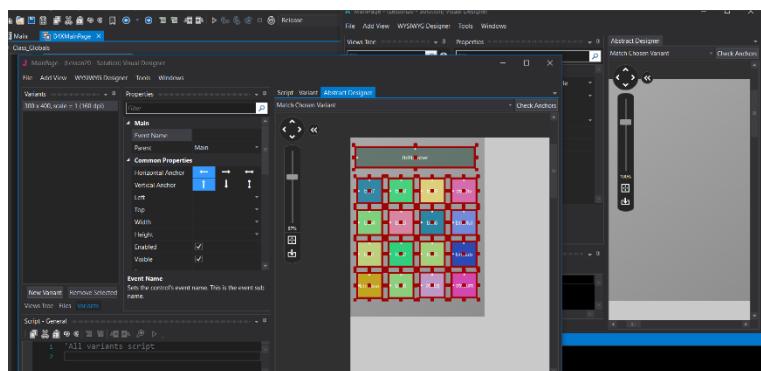
After you complete the installation do not forget to install on your mobile the application **B4A-Bridge** which will help you transfer the calculator to the mobile. The app is free to install from Google Play.

Transfer of design

Open the calculator with B4J. Similarly open the calculator with B4A. The file of interest is located within the B4A folder and has a b4a extension. Already all the code of the application exists within the B4A!

Open the Designer which visually makes no difference to that of b4j.

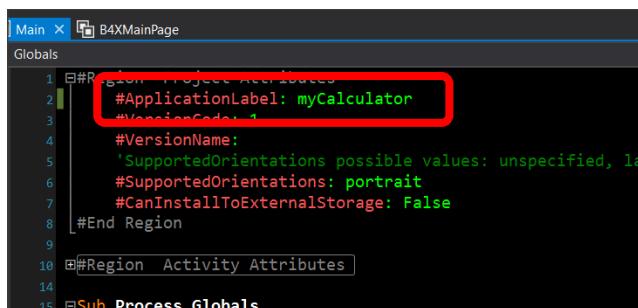
The design **has not been transferred** to the B4A but you can easily select all objects from the designer of B4J and by copying and pasting to the design will be added to the B4A project.



You can also make any changes you want to make to the "look" of your app. Finally, save the changes and return to B4A IDE.

Install an app on your mobile phone.

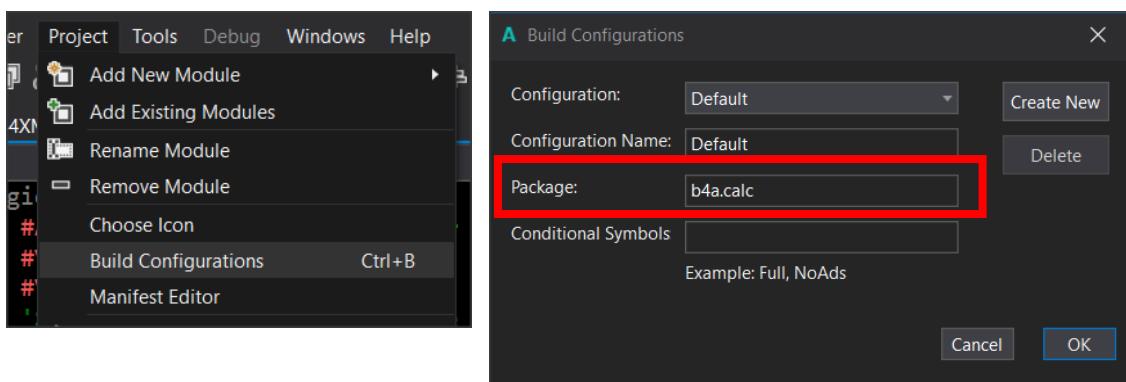
Before installing the app on a mobile device, you must specify the name of your app and the name of the "package" to be sent to the mobile.



```
Main X B4XMainPage
Globals
1  #:Region Object Attributes
2  #ApplicationLabel: myCalculator
3  #:VersionCode: 1
4  #:VersionName:
5  #:SupportedOrientations possible values: unspecified, landscape, portrait
6  #:SupportedOrientations: portrait
7  #:CanInstallToExternalStorage: False
8  #:End Region
9
10 #:Region Activity Attributes
11
12 #:Sub Process Globals
```

Picture 57 Application Label

From the tab Main specify the name you want the application to have in the directive `#ApplicationLabel`: It's the name that will appear on the screen of your mobile phone or tablet.

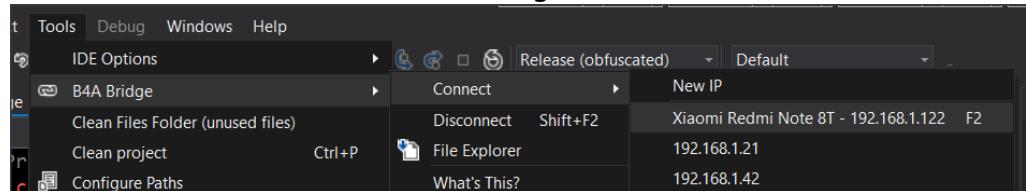


Picture 58 Build Configurations

From the Build Configurations menu, specify the name of the `Package`: This should be something unique for your device, which is why it is recommended to start with "b4a.". In the example, the package name is "b4a. calc».

Installation on the mobile phone requires that you have already installed the application B4A-Bridge (on the mobile device).

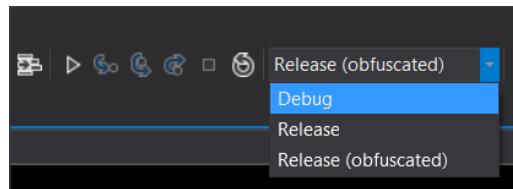
1. Launch the B4A-Bridge app on your Android device and make sure that it's connected to your local network.
2. From the Tools menu -> B4A Bridge -> Connect



Select your device that should appear in the menu. If you can't find it, check to see if it's connected to your local area network.

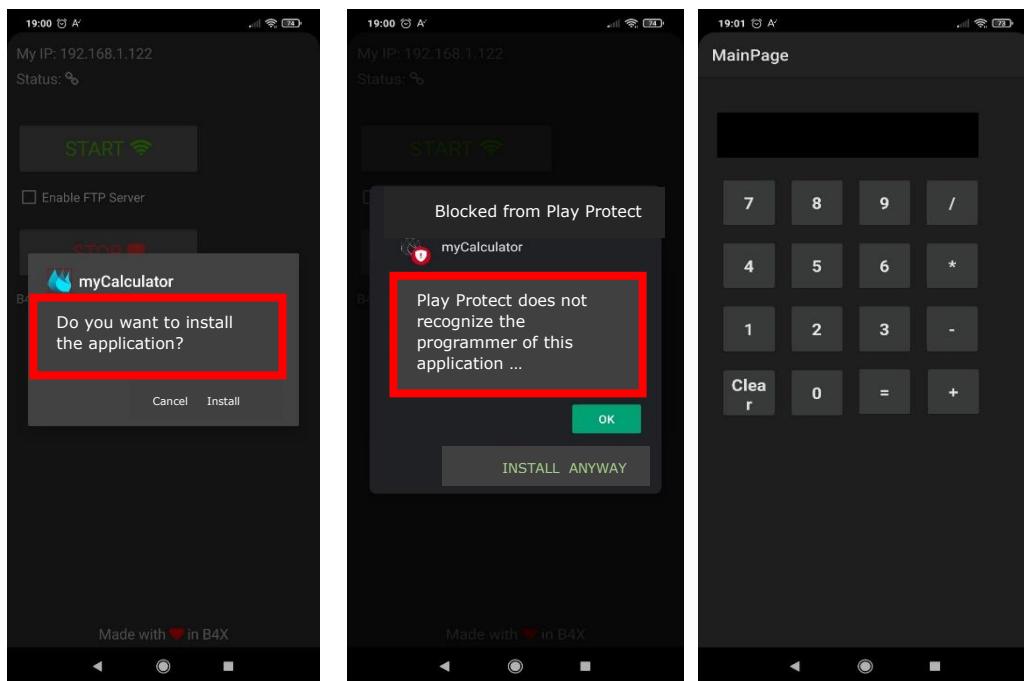


3. Select the type of installation to perform. Debug if you are in the process of debugging the application and Release once you are



finished with the deployment and want it to run independently of the development environment of B4A.

4. Tap the run icon and check your phone in B4A-Bridge.



During installation, you may receive a message from Play Protect that you can ignore and continue.

Appendix 1

Material for teams.

In next few pages you will find useful materials to help teams organize their work. You can copy and share them.



Anywhere Software

Timetable

Project _____

Date _____

Students _____

Organize your time - What to do and when

Hours

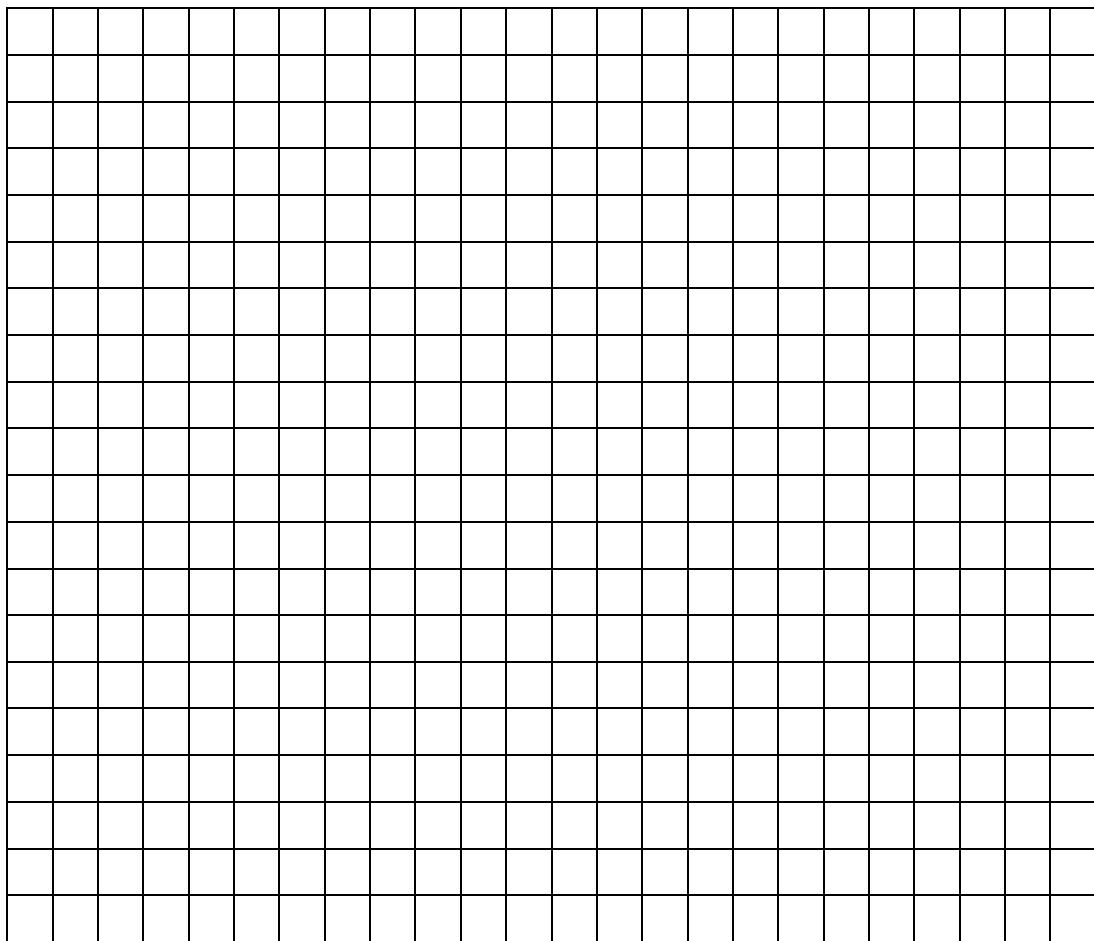
	1	2	3	4	5	6	7	8	9

Wireframe Sketching

Form Name

Date

Student



Anywhere Software

Classes

Class Name:

Date:

Student:

- Attributes:**
- 1.
 - 2.
 - 3.
 - 4.
 - 5.
 - 6.
 - 7.
 - 8.

Method

Description:

Method

Description:

Method

Description:

Method

Description:

B4XPage Name:

Date:

Student:

1.

2.

3.

4.

5.

6.

7.

8.

9.

10.

Public Variables:

Event

Description:

Event

Description:

Event

Description:

