Programming with B4X

Brief Theory

Version 1.0, February 2021

Anywhere Software

Table of Contents

Anywhere Software

Lesson 1 – Why B4X?

⏱ **1h**

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

So, the programming language must be:

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

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

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

Installing B4X

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

Lesson 2 – The meaning of the problem

⏱ **1h**

What students should know

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

The concept of the problem

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

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

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

Understanding the problem

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

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

Searching for a solution or set of solutions.

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

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

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

Choosing the right solution

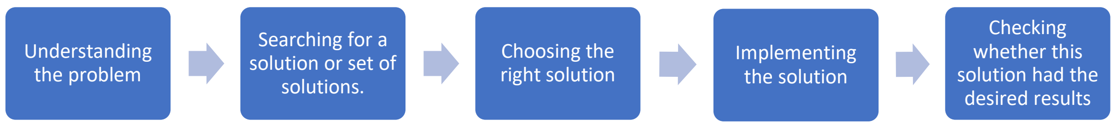
Often a problem can have more than one solution or some of them may be more efficient than others. Choosing the solution correctly leads to a shorter and more reliable outcome.

Implementing the solution

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

Checking whether this solution had the desired results.

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


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

Picture 1 The steps

Lesson 3 - My first Program

⏱ **2h**

What students should know

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

Hello World

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

Recommended instructions for the instructor

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

Teachers tip

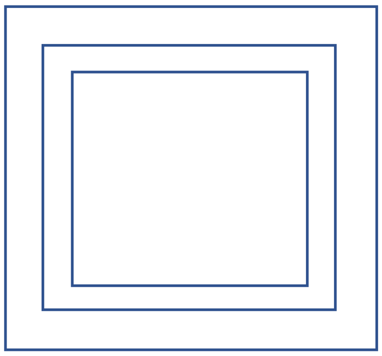
This is students first lesson. Therefore, keep it simple!

Function	Description
How to create a first project with B4J	*Menu File -> New -> B4Xturtle* *Project Folder* *Project Name* 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
Run Project	*Menu Project -> Compile & Run* Explain what means compile and how to recognize syntax errors in log. You don't need to provide to much information about compilation. Just the basic stuff in order to run a project.

Function	Description
#Region Project Attributes	Change Values in *MainFormWidth* and *MainFormHeight* to make different size application
Sub Turtle_Start	*What means Sub?* A small amount of code which is doing a certain activity.
Turtle methods	What is Turtle? What .MoveForward do? How can we find more commands? (Tell students to write "Turtle." To see the list of methods.
Errors	How to identify an error in 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 bellow.



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 memory address and the "resident" content. On the computer often the "resident" who from now on we will call variable needs more houses to fit.

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

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

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

String	array of characters

Table 1- Simple types of variables

Every type needs different space in memory to store values.

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

Remember

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

How to find how many variables you need

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

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

- Reads
- Registers
- Ask
- Accept
- Type

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

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

Information's

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

- Calculates
- Displays
- Writes
- Counts
- Convert

In the previous examples what are the requested?

Naming Variables

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

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

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

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

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

Declaring Variables

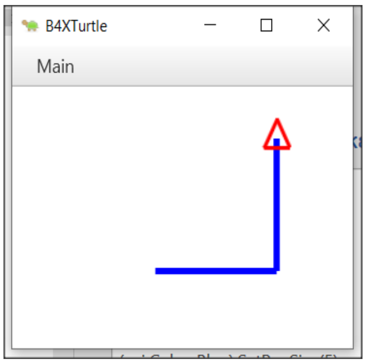
My First Variable

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

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

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

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

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



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

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

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

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

Teachers tip

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

Comments

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

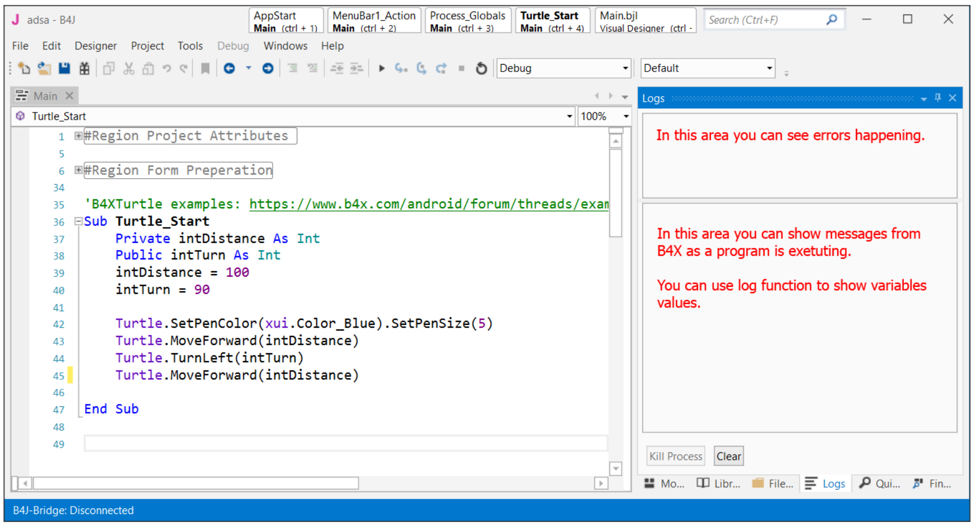
In B4X comments are inserted by writing the character ' as their first letter. From this point on it is not recognized by the translator of the language. Generally, in B4X comments you should put anywhere it is important to remember what you are doing as well as before the subprograms to explain what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

Example

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

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

The log area and the log function.

During programming various errors occur. Generally, errors in programming are divided into two categories syntax and logical. For now, we will deal with the syntax errors that are recognized by the programming language and indicate them on the logs screen. In order to access the logs screen we need to click on the relevant logs tab at the bottom right. The Logs screen itself is divided into two frames, the first of which displays errors and the bottom screen displays language messages or that information we want to display using the log() function. Using the Log() function helps the developer display messages while running a program as well as variable values to help control the program's proper operation.


adsa - B4J
AppStart Main (ctrl + 1)
MenuBar1_Action Main (ctrl + 2)
Process_Globals Main (ctrl + 3)
Turtle_Start Main (ctrl + 4)
Main.bjl Visual Designer (ctrl
Search (Ctrl+F)
File Edit Designer Project Tools Debug Windows Help
Debug
Default
Main
Turtle_Start
100%
Logs
Kill Process
Clear
Mo... Libr... File... Logs Qui... Fin...
B4J-Bridge: Disconnected

```
#Region Project Attributes

#Region Form Preperation

'B4XTurtle examples: https://www.b4x.com/android/forum/threads/exam
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
```

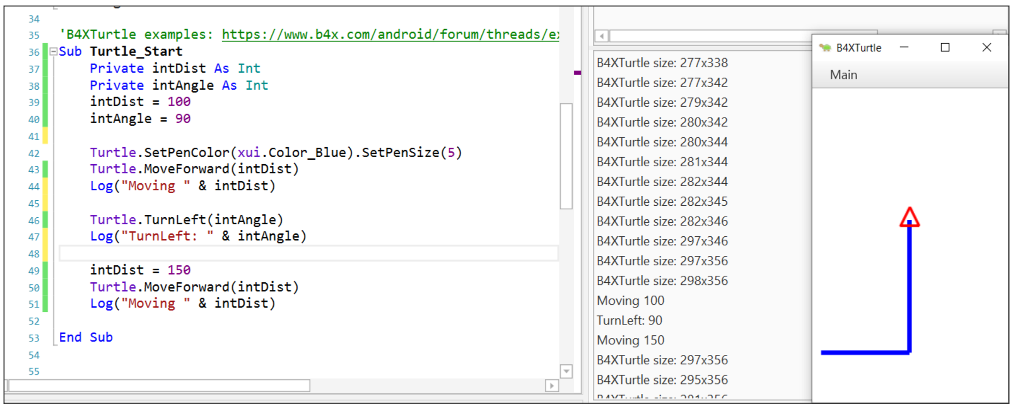
In this area you can see errors happening.

In this area you can show messages from B4X as a program is exetuting.

You can use log function to show variables values.

Figure 2 Logs Screen

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



```
'B4XTurtle examples: https://www.b4x.com/android/forum/threads/e
Sub Turtle_Start
    Private intDist As Int
    Private intAngle As Int
    intDist = 100
    intAngle = 90

    Turtle.SetPenColor(xui.Color_Blue).SetPenSize(5)
    Turtle.MoveForward(intDist)
    Log("Moving " & intDist)

    Turtle.TurnLeft(intAngle)
    Log("TurnLeft: " & intAngle)

    intDist = 150
    Turtle.MoveForward(intDist)
    Log("Moving " & intDist)

End Sub
```

```
B4XTurtle size: 277x338
B4XTurtle size: 277x342
B4XTurtle size: 279x342
B4XTurtle size: 280x342
B4XTurtle size: 280x344
B4XTurtle size: 281x344
B4XTurtle size: 282x344
B4XTurtle size: 282x345
B4XTurtle size: 282x346
B4XTurtle size: 297x346
B4XTurtle size: 297x356
B4XTurtle size: 298x356
Moving 100
TurnLeft: 90
Moving 150
B4XTurtle size: 297x356
B4XTurtle size: 295x356
```

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
Log(intS)                   'Shows 90

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

intA = intAa + 1            'Increase intA by 1
Log(intA)                   'Shows 41

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

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

Strings

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

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

```
Private strName as String
```

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

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

Also we can string together using the character &.

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

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

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

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

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

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

Teachers tip

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

Exercises

1. In the following exercises, identify the variables you need to declare. For each of them, write the relevant statement and give it an appropriate name.
    - Calculate the volume of a cylinder with a radius of one metre and a height of two metres.
    - Make a program that accepts a positive integer and calculates its square, cube, and square root.
    - Make a program that reads a sum of money in € and calculates and displays the corresponding amount in $.

- Write a program that reads the length of the sides of a rectangle from the keyboard and calculates and displays its area.
- The total resistance R of two resistances R1 and R2 connected in series is R1 + R2 and parallel (R1*R2)/(R1+R2) respectively. Male a program that it reads two values of resistant R1 and R2 and calculates the total resistance in series and parallel.

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

Name	**True**	**False**
int Age	☐	☐
_fltAmount	☐	☐
strName	☐	☐
1myAge	☐	☐
int_value	☐	☐

3. It's the end of the semester and you got your grades from three classes: Geometry, Algebra, and Physics. Create a program that: gives in 3 variables the grades of these 3 classes (Grades range from 0 - 10) Calculate the average of your grades.
4. You have bought a Bitcoin and now it's on the rise!!! Create a program that:
- Assign the value of the bitcoin at the time of purchase.
- Assign the percentage of increase (or decrease)
- Logs the total value of your bitcoin.
- Logs the increase or decrease value.

5. You now own some property, and you want to calculate the total area of the property. Create a program that:
- Assign the width and height in two variables.
- Calculate and log the area.

6. You are interested in buying a new laptop. You check the price and you see that the price is 300$ without the 10% tax. Create a program that:
- Assign the the price of the laptop in a variable.
- Assign the tax percentage in a second variable.

- Calculate and logs the total amount.

7. In a company the monthly salary of an employee is calculated by the minimum wage 400$ per month, plus 20$ multiplied by the number of years employed, plus 30$ for each child they have. Create a program that:
   - Assign the number of years employed in a variable
   - Assign the number of children the employee has in second variable.
   - Calculate and logs the total amount of salary the employee makes.
8. Create a program that log the last digit of a given integer.
9. Create two variables a and b, and initially set them each to a different number. Write a program that swaps both values.

   Example: a = 10, b = 20

   Output: a = 20, b = 10
10. Create two variables ‘a’ and ‘b’, and initially set them each to a different number. Write a program that double the Value of ‘a’ variable and increase the value of ‘b’ by 1.

    Example: a = 10, b = 20

    Output: a = 20, b = 21

Lesson 5 – Designer

⏱ **2h**

What students should know

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

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

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

Before designing any app remember some key design elements (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.

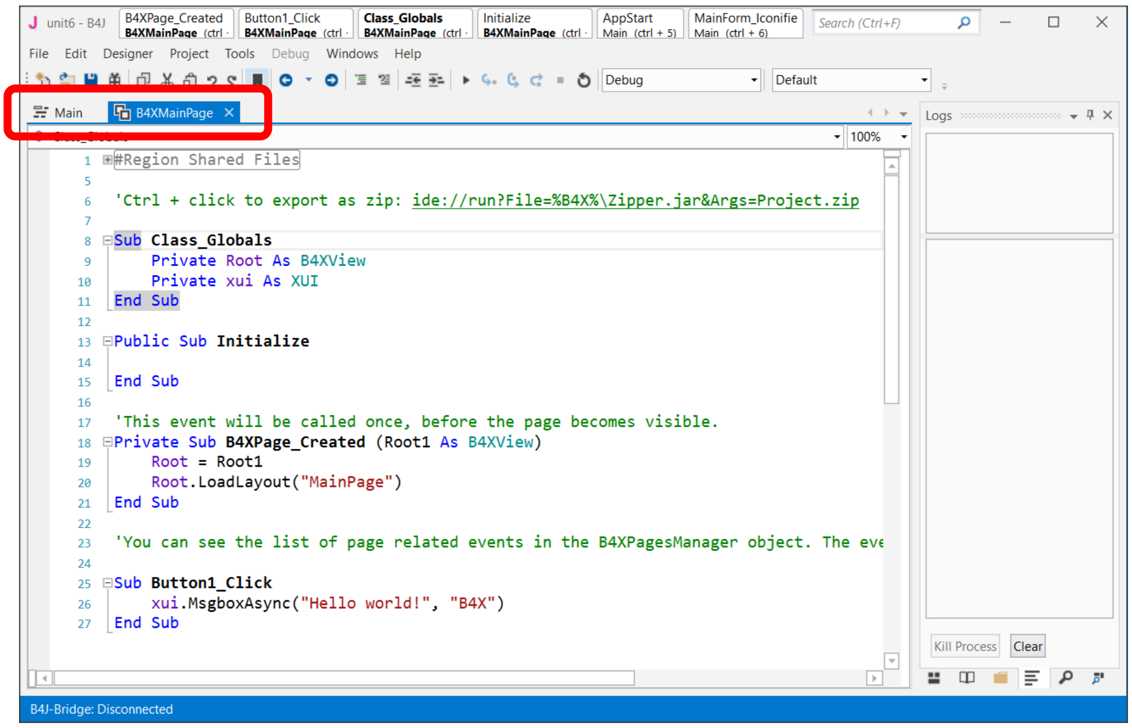
Use typography to create hierarchy and clarity. Carefully consider how you use typeface. Different sizes, fonts, and arrangement of the text to help increase scanability, legibility and readability.

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



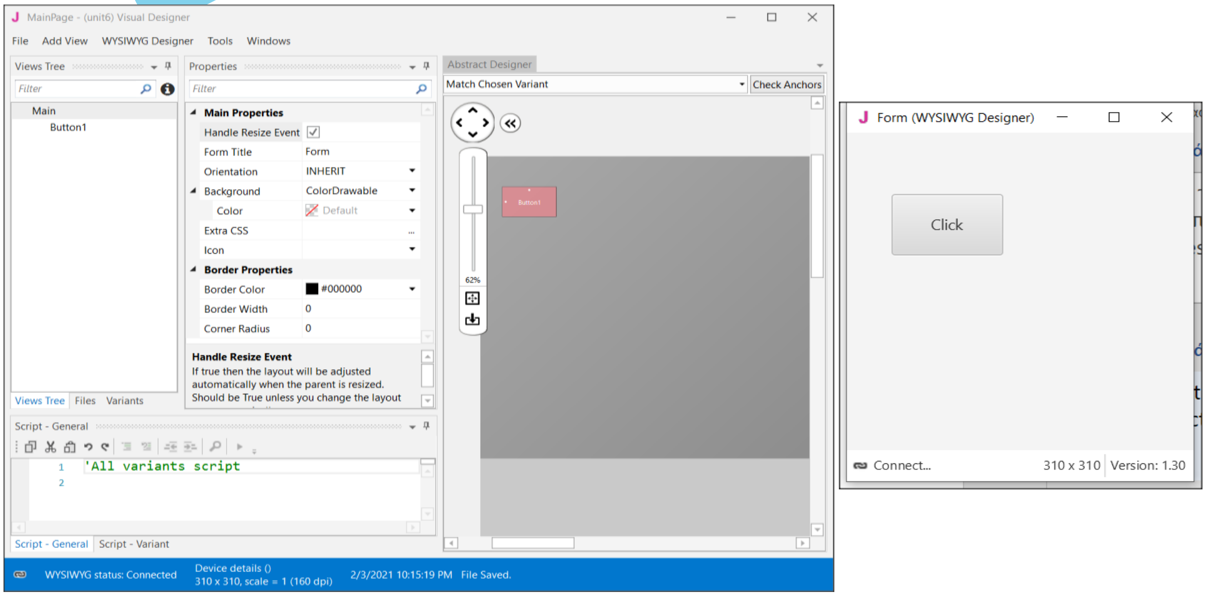
unit6 - B4J
B4XPage_Created B4XMainPage (ctrl ·
Button1_Click B4XMainPage (ctrl ·
Class_Globals B4XMainPage (ctrl ·
Initialize B4XMainPage (ctrl ·
AppStart Main (ctrl + 5)
MainForm_Iconifie Main (ctrl + 6)
Search (Ctrl+F)
File Edit Designer Project Tools Debug Windows Help
Debug
Default
Main
B4XMainPage
100%
Logs
Kill Process
Clear
B4J-Bridge: Disconnected

```
#Region Shared Files

'Ctrl + click to export as zip: ide://run?File=%B4X%\Zipper.jar&Args=Project.zip

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

Public Sub Initialize

End Sub

'This event will be called once, before the page becomes visible.
Private Sub B4XPage_Created (Root1 As B4XView)
    Root = Root1
    Root.LoadLayout("MainPage")
End Sub

'You can see the list of page related events in the B4XPagesManager object. The eve

Sub Button1_Click
    xui.MsgboxAsync("Hello world!", "B4X")
End Sub
```

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

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

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



Main Properties	
Handle Resize Event	✓
Form Title	Form
Orientation	INHERIT
Background	ColorDrawable
Color	Default
Extra CSS	
Icon	
Border Properties	
Border Color	#000000
Border Width	0
Corner Radius	0

Handle Resize Event
If true then the layout will be adjusted automatically when the parent is resized. Should be True unless you change the layout

```
'All variants script
```

WYSIWYG status: Connected — Device details () 310 x 310, scale = 1 (160 dpi) — 2/3/2021 10:15:19 PM File Saved.

Figure 4 Designer Screen

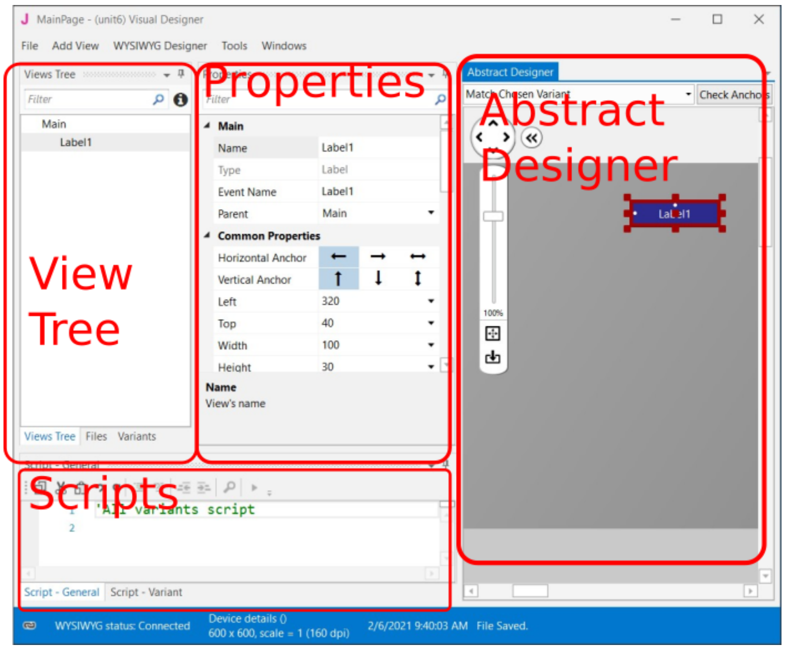
Visual designer

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

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

Remember

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



Main	
Name	Label1
Type	Label
Event Name	Label1
Parent	Main
Common Properties	
Horizontal Anchor	
Vertical Anchor	
Left	320
Top	40
Width	100
Height	30

Name
View's name

```
'All variants script
```

WYSIWYG status: Connected — Device details () 600 x 600, scale = 1 (160 dpi) — 2/6/2021 9:40:03 AM File Saved.

Figure 5 Designer's parts

The Views Tree

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

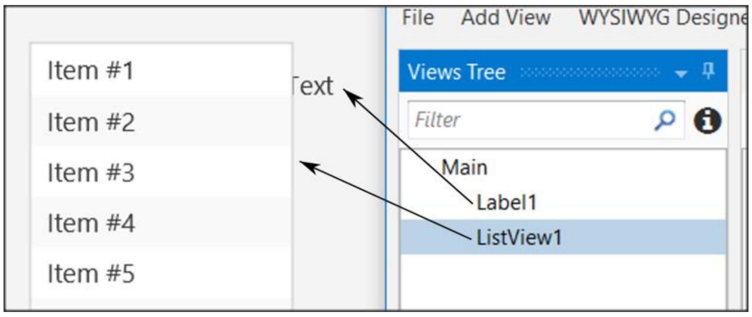


Figure 6 Views Tree

Properties

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

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

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

Table 3 Naming objects

Abstract Designer

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

Example 1

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

Decide on the size of the app screen.

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

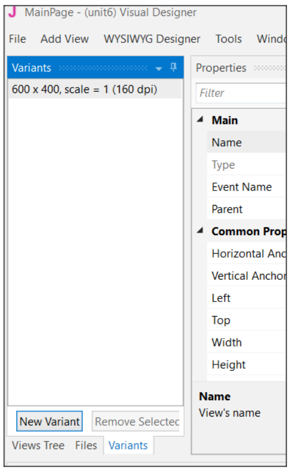
To set the application's size before beginning the Designer first go to **Main** Tab and change the first lines of code Width and Height:

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

Save your project and open Designer.

Set an appropriate variant.

Usually, you should set the variant as the MainFormWidth and MainFormHeight. This will help you plan without the risk of going outside the screen limits.
Choose Variants and then New Variant and set the width and height.
You can have as many variants as you want for different screen size but for now, we stay to only one.
Also, you can remove any variant by selecting it and choosing "Remove Variant".



600 x 400, scale = 1 (160 dpi)

Name
View's name

Figure 7 Variants Screen

Design a wireframe.

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

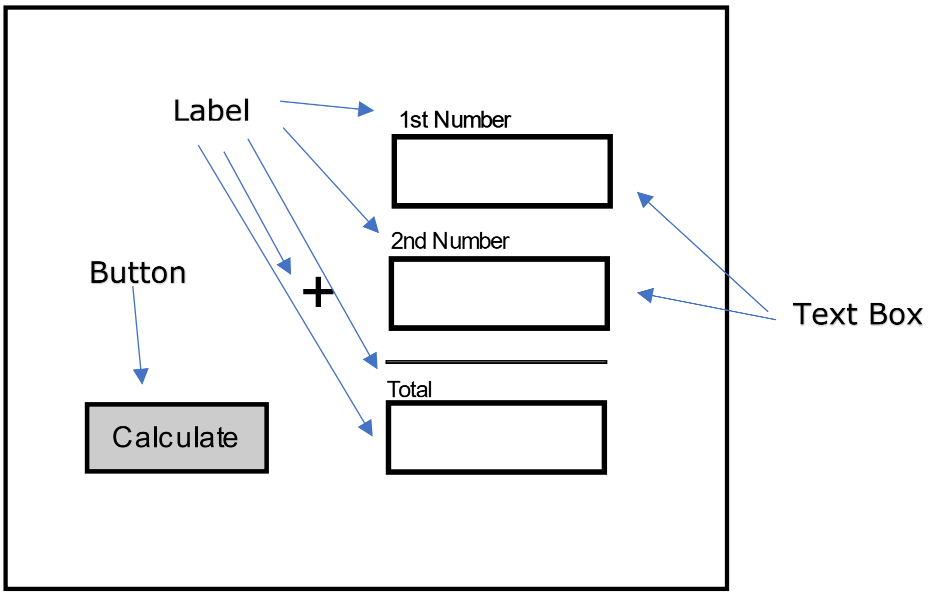


Figure 8 Wireframe

Create the views.

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

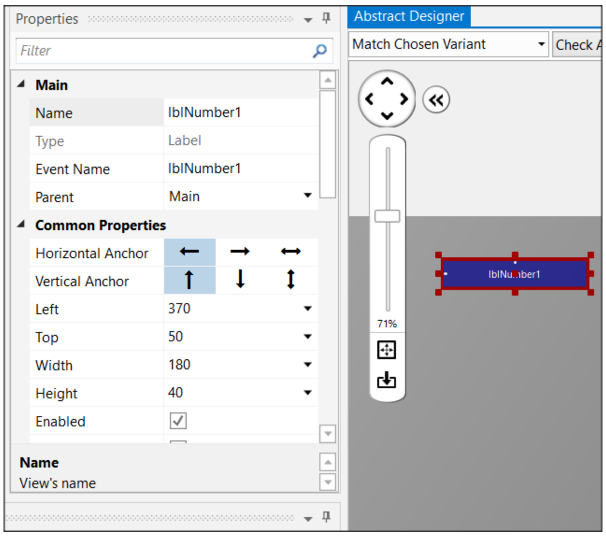
Inserting a label.

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

Now scroll down the Properties and set:

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

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



Properties

Filter

Main

Name	lblNumber1
Type	Label
Event Name	lblNumber1
Parent	Main

Common Properties

Horizontal Anchor	
Vertical Anchor	
Left	370
Top	50
Width	180
Height	40
Enabled	✓

Name

View's name

Figure 9 Labels

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

Inserting a Text Field.

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

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

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

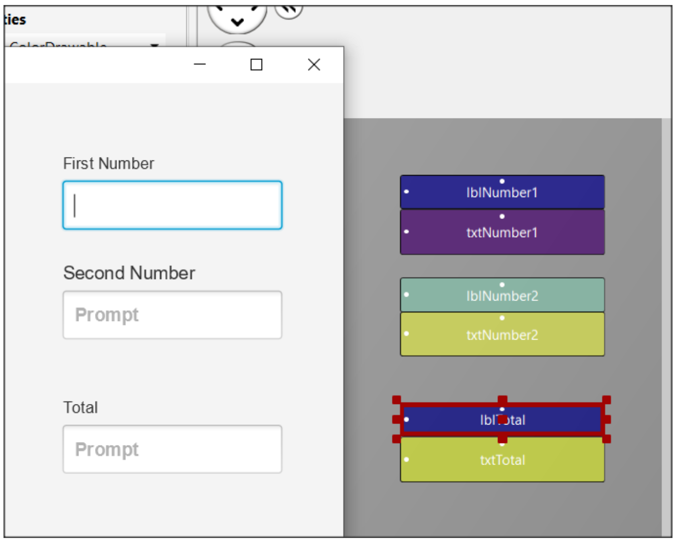


Figure 10 Text Fields

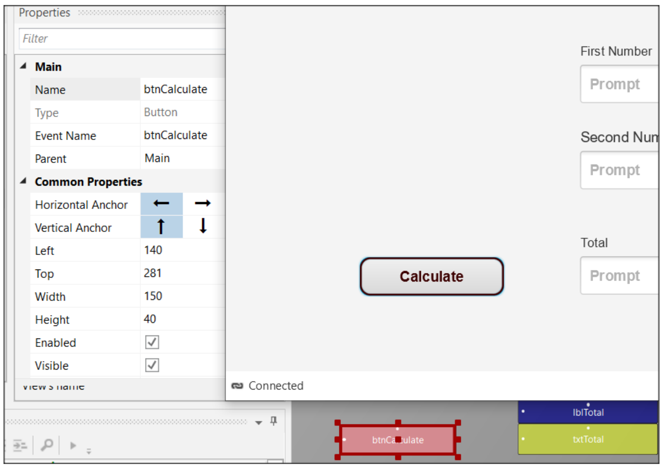
Inserting a button

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

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

From the Views menu select Button, and then set:

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



Main	
Name	btnCalculate
Type	Button
Event Name	btnCalculate
Parent	Main
Common Properties	
Horizontal Anchor	← →
Vertical Anchor	↑ ↓
Left	140
Top	281
Width	150
Height	40
Enabled	✓
Visible	✓

Figure 11 Buttons

Remember

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

Inserting a Pane

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

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

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

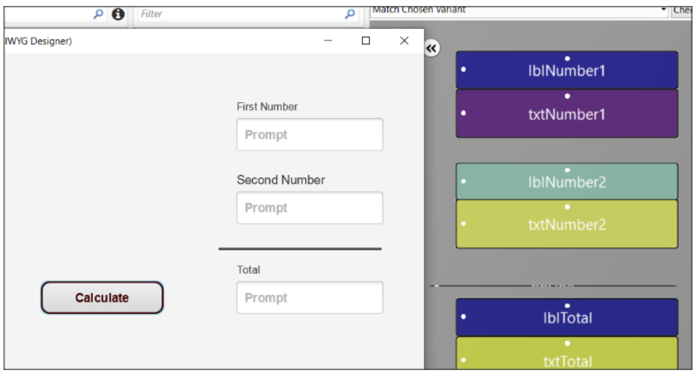


Figure 12 Pane

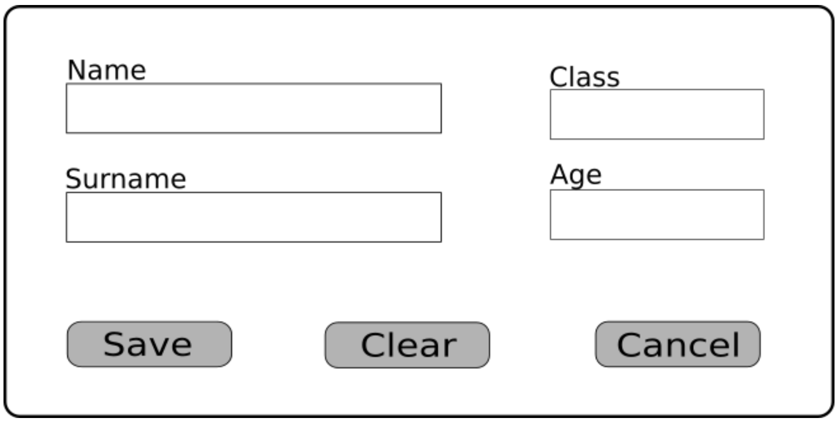
Now from menu File save the form. The form has already a name (MainPage) so you don't need to give an other name.

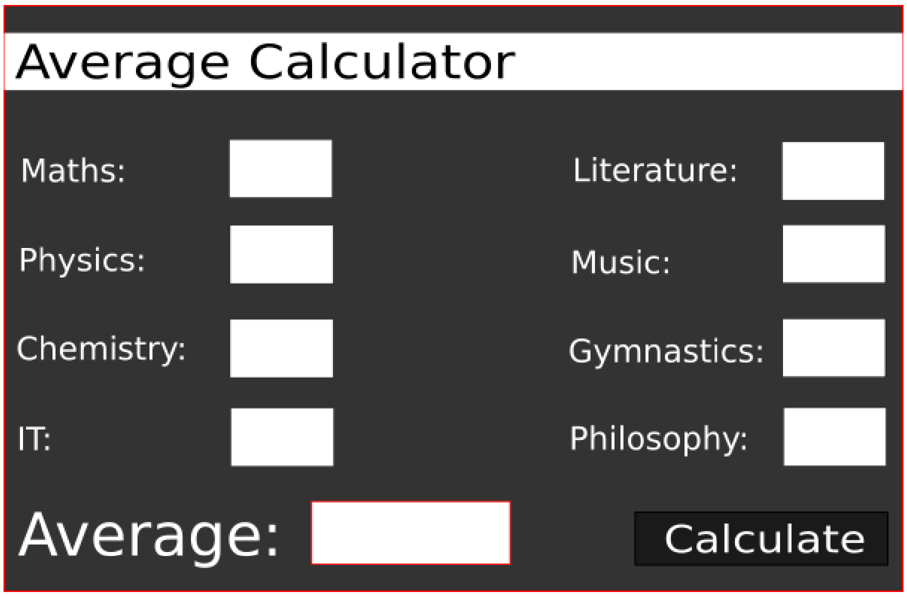
Excercises

1. Use the designer to create the following wireframes.

Teachers tip

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





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

Lesson 6 – From Designer to Code

⏱ **2h**

What students should know

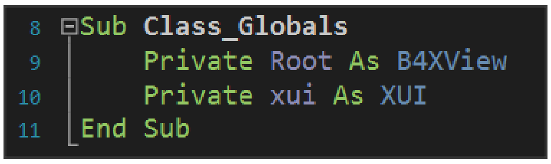
- Class_Globals
- Variables and Subs
- Passing Values to Code
- Events
- Attributes

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

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

Class_Globals

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



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

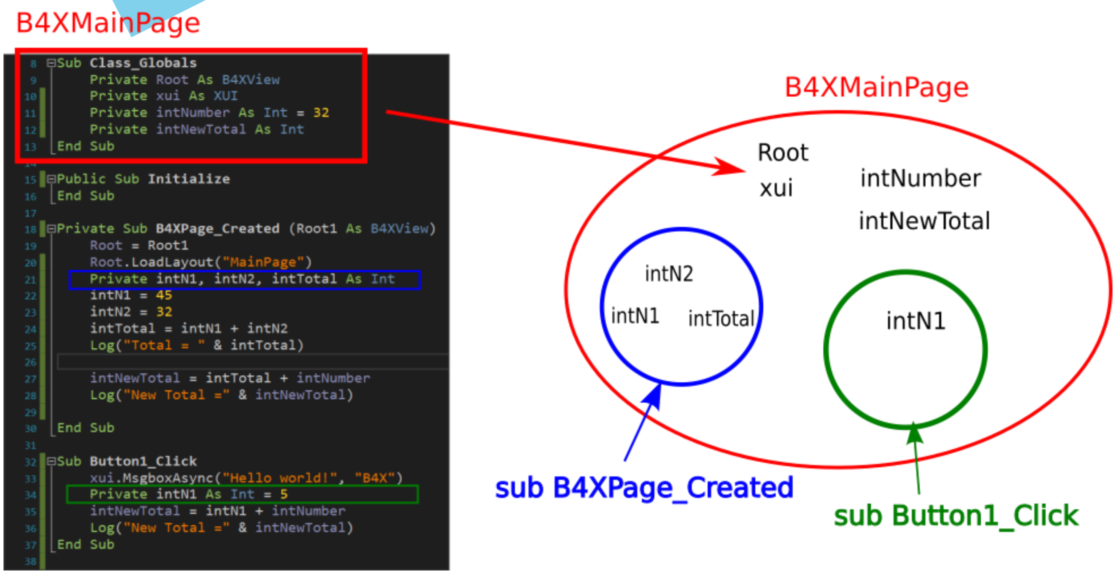
Picture 2 Sub Class_Globals

As it has already been said in 3rd Lesson Sub is a set of code that performs a specific operation. The operation of the Class_Globals is to collect the declarations of variables that we want to be known throughout the code of the tab B4XMainPage, i.e. in each subprogram.

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

A deeper look at the use of variables.

In the following code you see three subprograms



```
Sub Class_Globals
    Private Root As B4XView
    Private xui As XUI
    Private intNumber As Int = 32
    Private intNewTotal As Int
End Sub

Public Sub Initialize
End Sub

Private Sub B4XPage_Created (Root1 As B4XView)
    Root = Root1
    Root.LoadLayout("MainPage")
    Private intN1, intN2, intTotal As Int
    intN1 = 45
    intN2 = 32
    intTotal = intN1 + intN2
    Log("Total = " & intTotal)

    intNewTotal = intTotal + intNumber
    Log("New Total =" & intNewTotal)

End Sub

Sub Button1_Click
    xui.MsgboxAsync("Hello world!", "B4X")
    Private intN1 As Int = 5
    intNewTotal = intN1 + intNumber
    Log("New Total =" & intNewTotal)
End Sub
```

Picture 3 Variables and Range

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

On the contrary, variables intN1, intN2, intTotal 'live' within the sub-program B4XPage_Created and for this reason No another subprogram may use them. At the same time, the variable **intN1** who lives in the **Button1_Click is not the same** with the one in **B4XPage_Created**, both has its own memory space, and both can have the same name.

Teachers tip

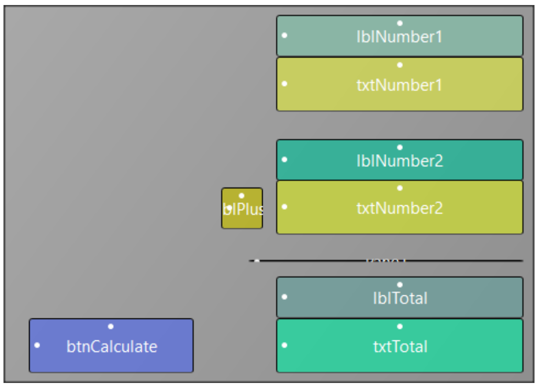
The use and range of variables is something that confuses new developers. Depending on your class, it is recommended to make examples of familiarity in their use.

Passing Values to Code

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

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

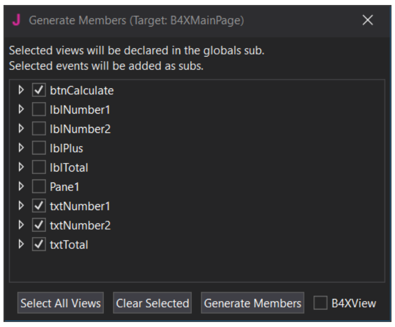
So, all "labels" items in this application do not have to be managed by code as well as the pane element. While button and textFields must be added to schedule functions on them.



Picture 4 Example 1 Designer View

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

From the Tools menu select Generate Members



Generate Members (Target: B4XMainPage)

Selected views will be declared in the globals sub.
Selected events will be added as subs.

- [✓] btnCalculate
- [ ] lblNumber1
- [ ] lblNumber2
- [ ] lblPlus
- [ ] lblTotal
- [ ] Pane1
- [✓] txtNumber1
- [✓] txtNumber2
- [✓] txtTotal

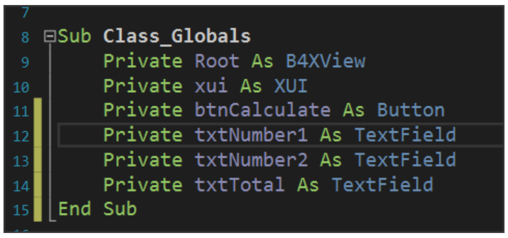
Picture 5 Generate Members

In Screen Generate Members just click on objects

- btnCalculate
- txtNumber1
- txtNumber2
- txtTotal

and then click Generate Members.

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



```
Sub Class_Globals
    Private Root As B4XView
    Private xui As XUI
    Private btnCalculate As Button
    Private txtNumber1 As TextField
    Private txtNumber2 As TextField
    Private txtTotal As TextField
End Sub
```

Picture 6 Class_Globals

Remember

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

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

Events

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

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

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

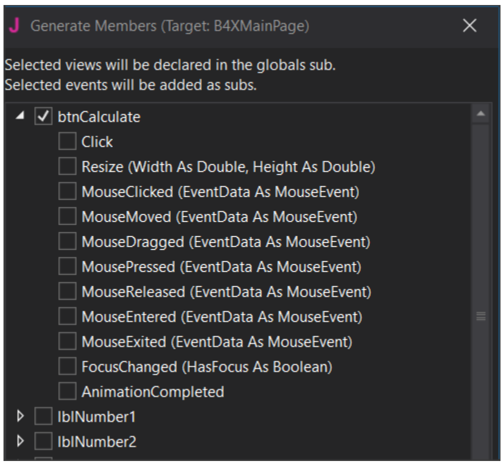
Remember

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

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

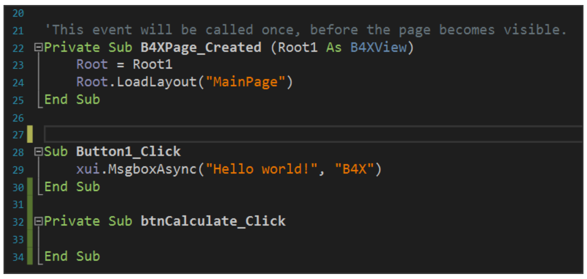
Open its list btnCalculate and several Events we'll be available to choose. Just click on "Click" and "Generate Members".

The subprogram for the **btnCalculate_**Click event has **already appeared.** Within it you will write the program code to complete.



```
Generate Members (Target: B4XMainPage)

Selected views will be declared in the globals sub.
Selected events will be added as subs.

[✓] btnCalculate
    [ ] Click
    [ ] Resize (Width As Double, Height As Double)
    [ ] MouseClicked (EventData As MouseEvent)
    [ ] MouseMoved (EventData As MouseEvent)
    [ ] MouseDragged (EventData As MouseEvent)
    [ ] MousePressed (EventData As MouseEvent)
    [ ] MouseReleased (EventData As MouseEvent)
    [ ] MouseEntered (EventData As MouseEvent)
    [ ] MouseExited (EventData As MouseEvent)
    [ ] FocusChanged (HasFocus As Boolean)
    [ ] AnimationCompleted
[ ] lblNumber1
[ ] lblNumber2
```

Picture 7 Setting Event

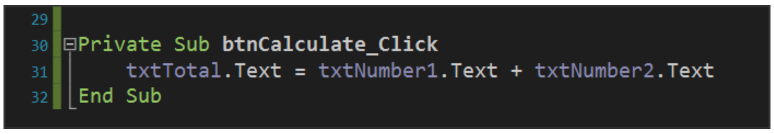


```
'This event will be called once, before the page becomes visible.
Private Sub B4XPage_Created (Root1 As B4XView)
    Root = Root1
    Root.LoadLayout("MainPage")
End Sub

Sub Button1_Click
    xui.MsgboxAsync("Hello world!", "B4X")
End Sub

Private Sub btnCalculate_Click

End Sub
```

Picture 8 The Click Event

Writing code in Event

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



```
Private Sub btnCalculate_Click
    txtTotal.Text = txtNumber1.Text + txtNumber2.Text
End Sub
```

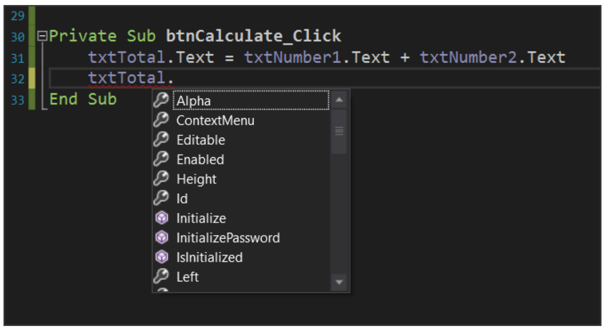
Picture 9 Calculating textFields

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

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

Properties

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



```
Private Sub btnCalculate_Click
    txtTotal.Text = txtNumber1.Text + txtNumber2.Text
    txtTotal.
End Sub
```

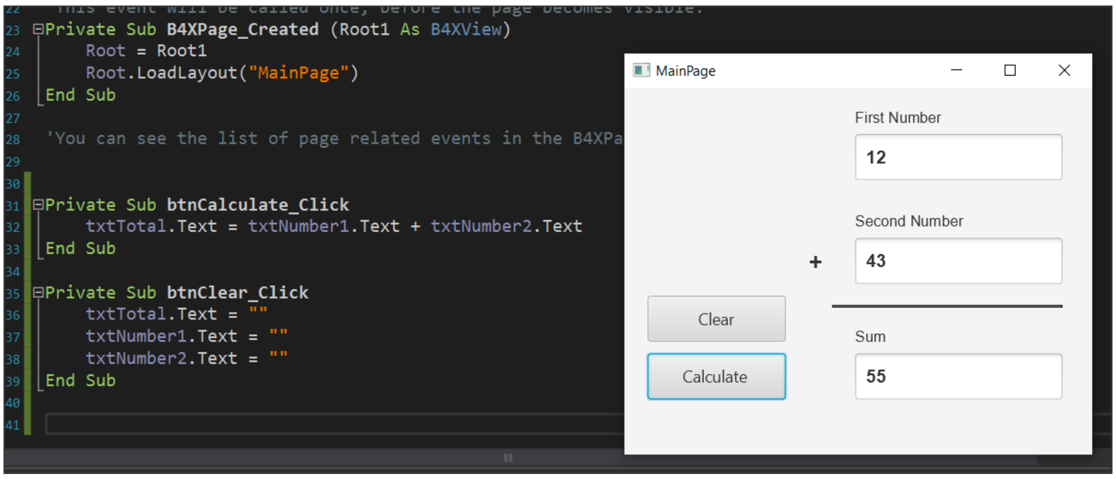
Picture 10 Object Properties

Remember

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

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

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



```
'This event will be called once, before the page becomes visible.
Private Sub B4XPage_Created (Root1 As B4XView)
    Root = Root1
    Root.LoadLayout("MainPage")
End Sub

'You can see the list of page related events in the B4XPa

Private Sub btnCalculate_Click
    txtTotal.Text = txtNumber1.Text + txtNumber2.Text
End Sub

Private Sub btnClear_Click
    txtTotal.Text = ""
    txtNumber1.Text = ""
    txtNumber2.Text = ""
End Sub
```

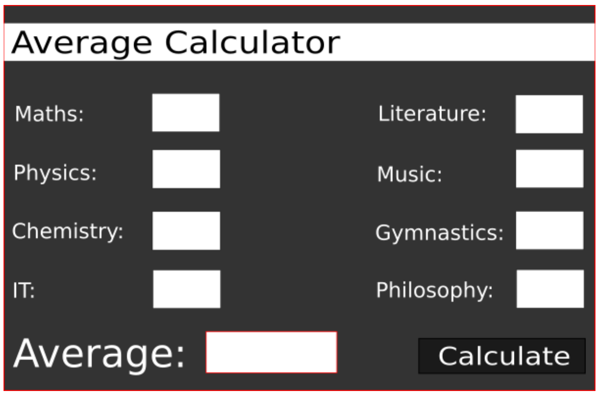
Picture 11 Clear Button and Code

When you press clear, the form will clear.

Exercises

1. Extend the example to perform the four operations: Add, subtract, multiply, and divide at the click of an appropriate button. Add the appropriate objects to the designer and then complete the code.

2. In exercise 2 of the previous course continue and complete the application by activating its functions.



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

Lesson 7 – Conditional Statements

⏱ **4h**

What students should know

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

Logical Variables

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

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

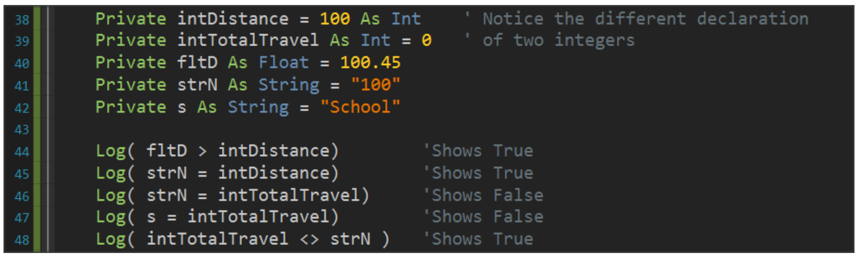
```
Private intDistance = 100, intTotalTravel = 0 As  Int
Private blnFlag  As  Boolean  = False
Private blnDone  As  Boolean
```

Comparative Operators

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

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

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



```
Private intDistance = 100 As Int    ' Notice the different declaration
Private intTotalTravel As Int = 0   ' of two integers
Private fltD As Float = 100.45
Private strN As String = "100"
Private s As String = "School"

Log( fltD > intDistance)        'Shows True
Log( strN = intDistance)        'Shows True
Log( strN = intTotalTravel)     'Shows False
Log( s = intTotalTravel)        'Shows False
Log( intTotalTravel <> strN )   'Shows True
```

Picture 12 Variable Comparisons

Remember

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

Logical Operators

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

Mathematician George Boole developed an algebra based on logical sentences.

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

Remember

AND (conjunction), denoted x AND y, satisfies x AND y = 1 if x = y = 1, and xANDy = 0 otherwise.

OR (disjunction), denoted x OR y, satisfies x OR y = 0 if x = y = 0, and x OR y = 1 otherwise.

NOT (negation), denoted NOT x, satisfies Not x = 0 if x = 1 and Not x = 1 if x = 0.

Example:

1st Sentence: Today it rains, **2nd Sentence**: *I am going to school*.

Today it's raining. (P1)	***I am going to school.* (P2)**	**P1 AND P2**	**P1 or P2**	**NO P1**

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

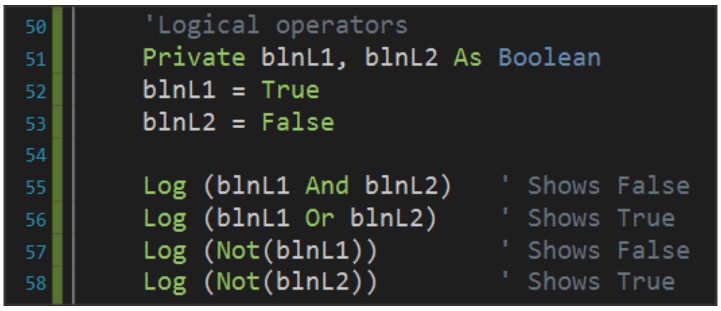
From the table we observe that

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

Logical operators in programming

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

In B4X logical variables are used as below.



```
'Logical operators
Private blnL1, blnL2 As Boolean
blnL1 = True
blnL2 = False

Log (blnL1 And blnL2)    ' Shows False
Log (blnL1 Or blnL2)     ' Shows True
Log (Not(blnL1))         ' Shows False
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			

4.

int A + int B	>=	intC	**OR**	(blnA	**AND**	blnB)
10 + 20		30		True		False
	True				False	
			True			

5.

strName1	=	“Georg	**OR**	strName	**=**	“John”

		e”		e2		
George		George		Georgia		John
	True				False	
			True			

If command

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

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

Its basic form is as follows:

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

Where condition enter a comparative or logical expression studied above.

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

Examples:

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

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

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

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

If – Else

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

Its basic form is as follows:

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

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

Examples:

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

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

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

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

If – else - else if

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

How to write:

```
If ( condition1 )  Then
   Commands
Else If (condition2 )   Then
   Commands
Else If (  condition3 )  Then
   Commands
Else If (  condition4 )  Then
   Commands
...
Else
   Commands
End If
```

The function of the multiple if is summarized as follows:

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

Example 3

A fast-food chain has these meals:

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

Create a program that:

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

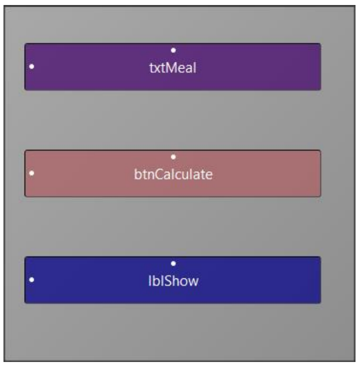
Solution

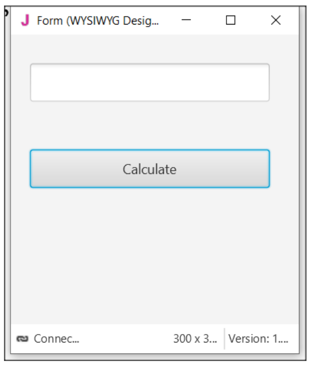
Step 1

Start a new project and give dimensions 300 x 300.

Step 2

In the designer design the app screen

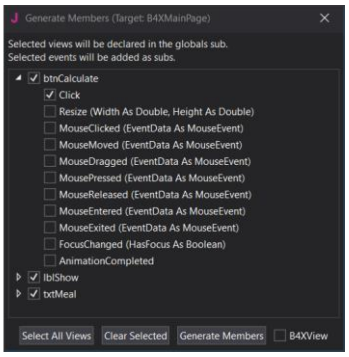




Picture 14 The app screen

Step 3

Enter txtMetal , btnCalculate, lblShow , and btnCalculate_Click.



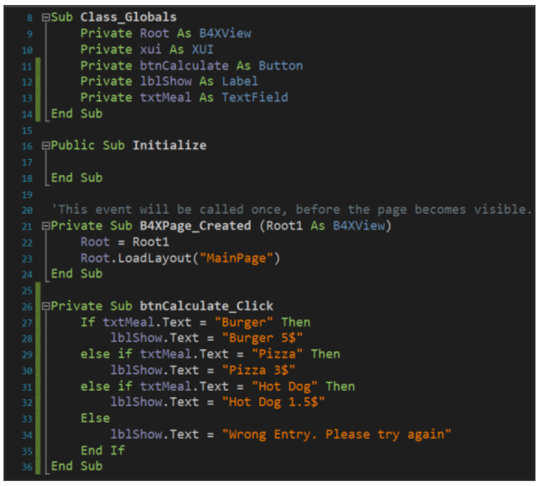
Generate Members (Target: B4XMainPage)

Selected views will be declared in the globals sub.
Selected events will be added as subs.

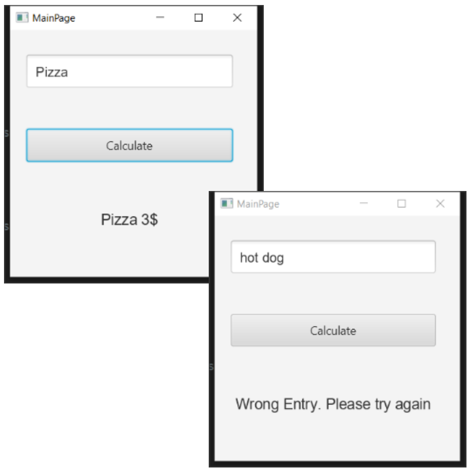
- [✓] btnCalculate
  - [✓] Click
  - [ ] Resize (Width As Double, Height As Double)
  - [ ] MouseClicked (EventData As MouseEvent)
  - [ ] MouseMoved (EventData As MouseEvent)
  - [ ] MouseDragged (EventData As MouseEvent)
  - [ ] MousePressed (EventData As MouseEvent)
  - [ ] MouseReleased (EventData As MouseEvent)
  - [ ] MouseEntered (EventData As MouseEvent)
  - [ ] MouseExited (EventData As MouseEvent)
  - [ ] FocusChanged (HasFocus As Boolean)
  - [ ] AnimationCompleted
- [✓] lblShow
- [✓] txtMeal

Step 4

The code you need to write is for btnCalculate_Click



```
Sub Class_Globals
    Private Root As B4XView
    Private xui As XUI
    Private btnCalculate As Button
    Private lblShow As Label
    Private txtMeal As TextField
End Sub

Public Sub Initialize

End Sub

'This event will be called once, before the page becomes visible.
Private Sub B4XPage_Created (Root1 As B4XView)
    Root = Root1
    Root.LoadLayout("MainPage")
End Sub

Private Sub btnCalculate_Click
    If txtMeal.Text = "Burger" Then
        lblShow.Text = "Burger 5$"
    else if txtMeal.Text = "Pizza" Then
        lblShow.Text = "Pizza 3$"
    else if txtMeal.Text = "Hot Dog" Then
        lblShow.Text = "Hot Dog 1.5$"
    Else
        lblShow.Text = "Wrong Entry. Please try again"
    End If
End Sub
```



Pizza

Pizza 3$

hot dog

Wrong Entry. Please try again

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

Algorithms with if

Find Maximum Number

Read 3 integers and find the largest in three different ways

Method 1 – Simple If Statement

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

Method 2 – Nested If

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

Method 3 – Max Algorithm

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

Exercises

1. Make the following suggestions in logical expressions.
    - 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 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. no (E<=C) or (D<=C)
    - iv. no ((B<=C) and (D<2))
    - v. no (no (B<=E) or not (C<=B))
    - vi. ((E<=A) and (E>=C)) and not (C>=A)
    - vii. no (no (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 needs.

   Prints the cost of the meal.

   Input example: "Hot Dog", 2

   Output: " 2 x Hot Dog 3$"

2. You have consumed X amount of Mbps on Wikipedia and Y amount of Mbps on memes. The cost of visiting Wikipedia is 0,10$ per Mb and the cost for watching memes is 0,05$ per Mb. If total consumption is more than 100$ print "Too much consumption". If watching meme consumption is greater than reading Wikipedia consumption print "WOW MANY MEMES", "SUCH LOL"(in new line). Create a program that:
   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 proper messages
3. An internet cafe has 2 ways of charging. If the user is a member pays 2$/hour, Else the user pays 5$. Find if someone is a member or not and calculate the price based on how many hours the user spend. If the user is a member the tax is 10% else the tax is 20%. Create a program that:
   a. Reads how many hours the user spend
   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".

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

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

Create a program that:

   a. Reads how many seconds was the calls duration.
   b. Calculates the monthly bill for the subscriber.
   c. Prints the total amount.
   d. Output: "total amount: 48$"
   e. Notice that that the charge for the first 500 seconds it's 0,01$ then for the next 501 to 800 seconds it's 0,008 and then it's 0,005$

7. In the qualifying races in the long jump at the Olympic Games, an athlete makes 3 initial attempts and if he has a performance of more than 7.50 meters, then he is entitled to continue and make another 3 more attempts. Read the first 3 attempts of an athlete and print a message whether he is entitled to continue or not and if he is entitled to find and print the best effort of the athlete.

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

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

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

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

Lesson 8 – Subroutines

⏱ **3h**

What students should know

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

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

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

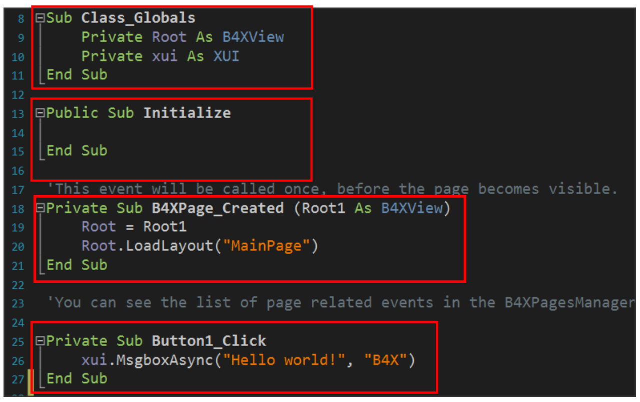
The subprogram should only do one task.

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

Have such a name that refers to its function.

Create a subprogram in B4J

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



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

Public Sub Initialize

End Sub

'This event will be called once, before the page becomes visible.
Private Sub B4XPage_Created (Root1 As B4XView)
    Root = Root1
    Root.LoadLayout("MainPage")
End Sub

'You can see the list of page related events in the B4XPagesManager

Private Sub Button1_Click
    xui.MsgboxAsync("Hello world!", "B4X")
End Sub
```

Picture 15 Subprograms

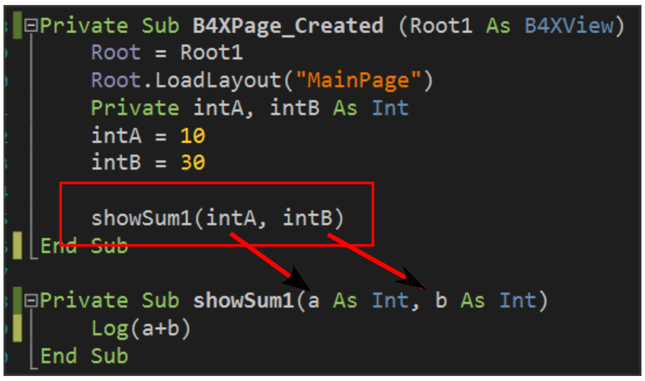
Example 1

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

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

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

The subprogram is written before or after the current subprogram:



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

    showSum1(intA, intB)
End Sub

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

Picture 16 Calling a subprogram

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

The Sub statement which means subroutine and

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

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

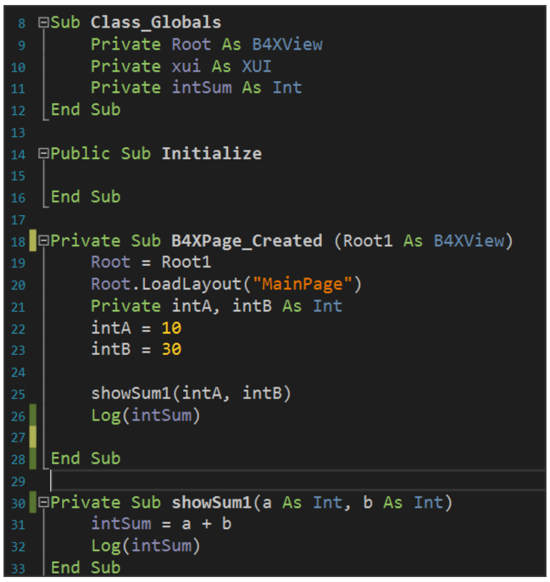
Remember

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

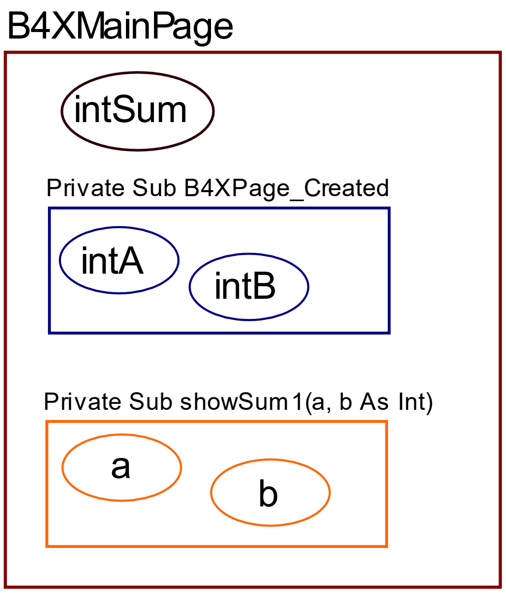
The subprogram now works with data contained within parameters a and b not the intA and IntB.

The memory of the subprogram in B4X

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



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

Public Sub Initialize

End Sub

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

    showSum1(intA, intB)
    Log(intSum)

End Sub

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


B4XMainPage
intSum
Private Sub B4XPage_Created
intA
intB
Private Sub showSum1(a, b As Int)
a
b

Picture 17 The Memory

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

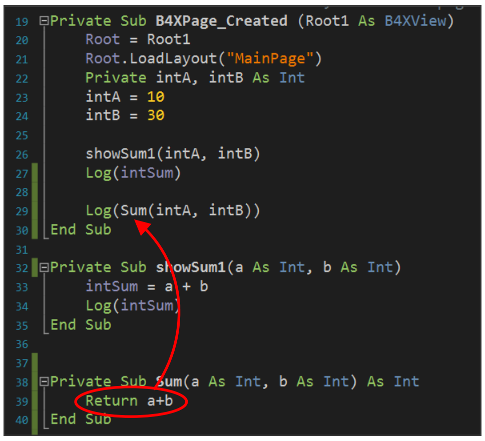
Remember

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

The variables declared within the subprograms are called **local** and are not known in the rest 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:



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

    showSum1(intA, intB)
    Log(intSum)

    Log(Sum(intA, intB))
End Sub

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


Private Sub Sum(a As Int, b As Int) As Int
    Return a+b
End Sub
```

Picture 18 Returning Values back to program.

The program must be declared as a variable type.

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

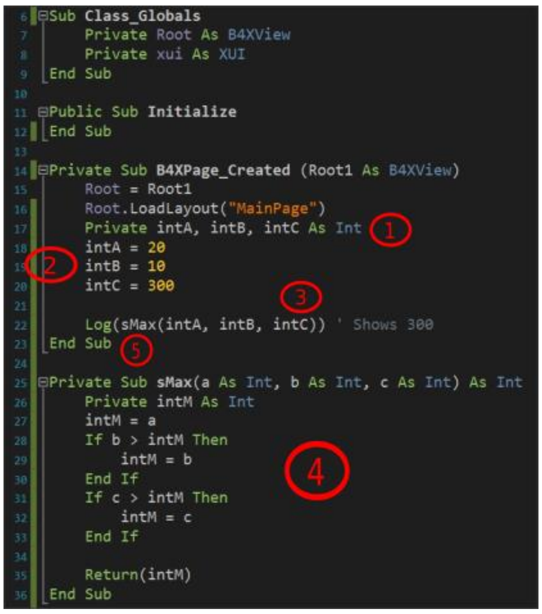
Finally, the code that called the subprogram accepts back the value calculated and can use it like any variable.

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.



```
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")
    Private intA, intB, intC As Int
    intA = 20
    intB = 10
    intC = 300

    Log(sMax(intA, intB, intC)) ' Shows 300
End Sub

Private Sub sMax(a As Int, b As Int, c As Int) As Int
    Private intM As Int
    intM = a
    If b > intM Then
        intM = b
    End If
    If c > intM Then
        intM = c
    End If

    Return(intM)
End Sub
```

Picture 19 Example 2

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.

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.

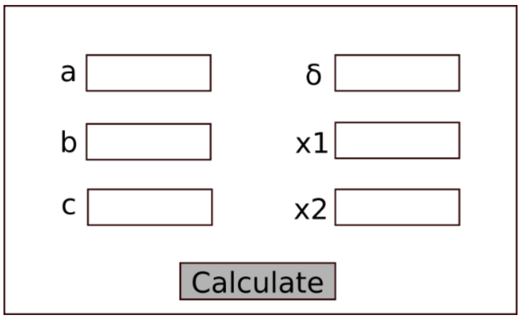
Exercises

Teachers tip

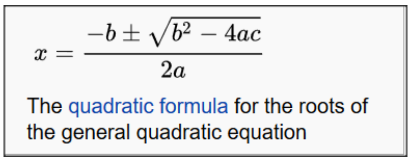
Encourage students to solve all exercises and discus the solution in class. Dedicate at least 1 hour to explain them.

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")**



Picture 21 Wireframe



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

The quadratic formula for the roots of the general quadratic equation

Picture 20 Source: Wikipedia.org

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

4. One theatre has three categories of tickets, Arena, Gallery, Proscenium. Each ticket costs 20, 30, 40€. Make a program that:

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

Lesson 9 – Classes

⏱ **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	**Student 2**	**Student 3**
125310	125311	125312
Augusta	Maria	Muḥammad

Anywhere Software

Ada Byron	Curie	al-Khwarizmi
London	Warszawa	Khwarazm
37535795	678433	646456456
ada@lon.uk	maria@wars.pol	algor@khwa.pe

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

Remember

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

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

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

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

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

Example of class in B4J

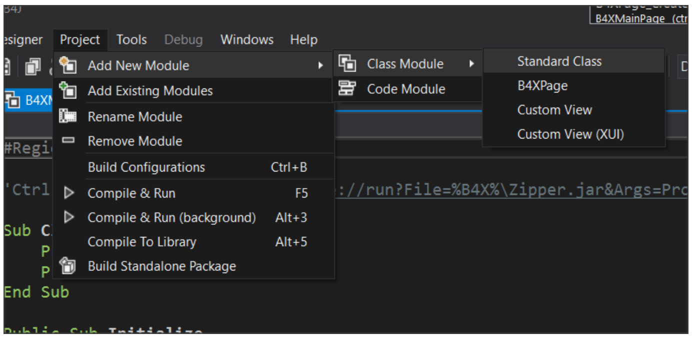
A library has a set of books which it lends to readers who subscribed in the library. Each book has features such as title, author, publisher, year of publication. Also, the Insert Book, Show Book Items, Change Book Items functions apply to each book.

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

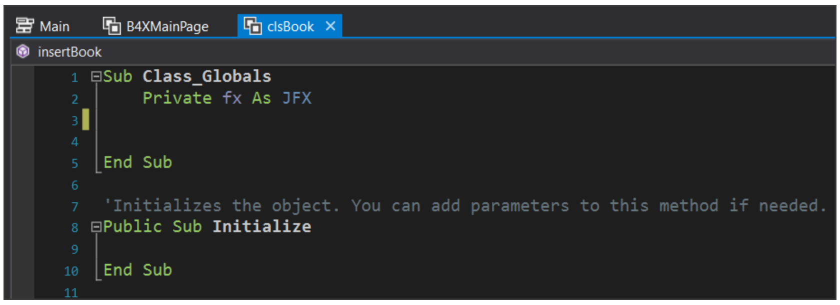
Implementation methodology

1. Create a new application B4XPages and give the name “library”.

2. From menu Project – Add New Module – Class Module menu select Standard Class.

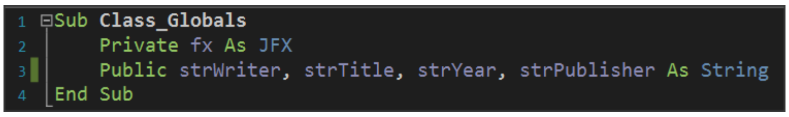


3. In the dialog box, name clsBook (i.e. class Book)
4. A new code tab named clsBook will be created.



```
Sub Class_Globals
    Private fx As JFX


End Sub

 'Initializes the object. You can add parameters to this method if needed.
Public Sub Initialize

End Sub
```

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



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

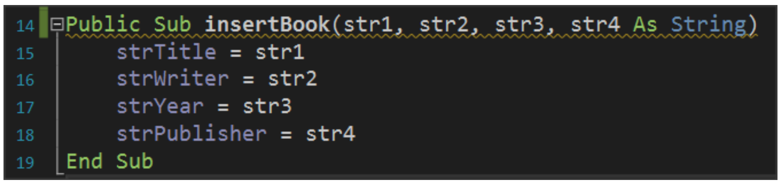
6. Finally, you must implement the subprograms that will implement the methods:
   a. Insert a book,
   b. Show Book Items,
   c. Change Book Items

Teachers tip

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

Insert a book.

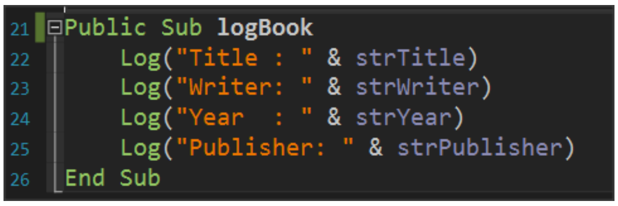
This is a subprogram that accepts 4 string data as parameters and then assigns it in the order that it is inserted into the subprogram into the variables strTitle, strWiter, strYear, strPublisher.



```
Public Sub insertBook(str1, str2, str3, str4 As String)
    strTitle = str1
    strWriter = str2
    strYear = str3
    strPublisher = str4
End Sub
```

Show Book Items.

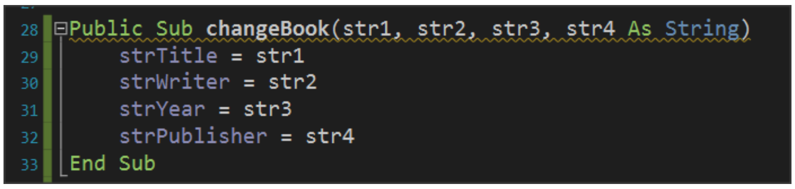
The subprogram displays with log command the properties of the Books class or in other words the variables that describe the class.



```
Public Sub logBook
    Log("Title : " & strTitle)
    Log("Writer: " & strWriter)
    Log("Year  : " & strYear)
    Log("Publisher: " & strPublisher)
End Sub
```

Change book items.

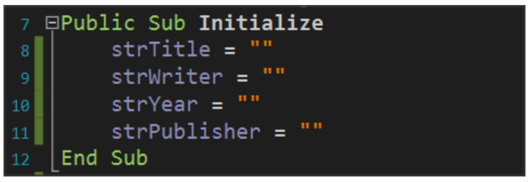
The method of changing items accepts as parameters new elements for the class and changes the property values of the corresponding   properties.



```
Public Sub changeBook(str1, str2, str3, str4 As String)
    strTitle = str1
    strWriter = str2
    strYear = str3
    strPublisher = str4
End Sub
```

The Initialize subprogram.

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

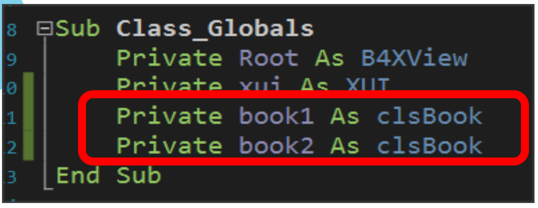


```
Public Sub Initialize
    strTitle = ""
    strWriter = ""
    strYear = ""
    strPublisher = ""
End Sub
```

Use Class

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

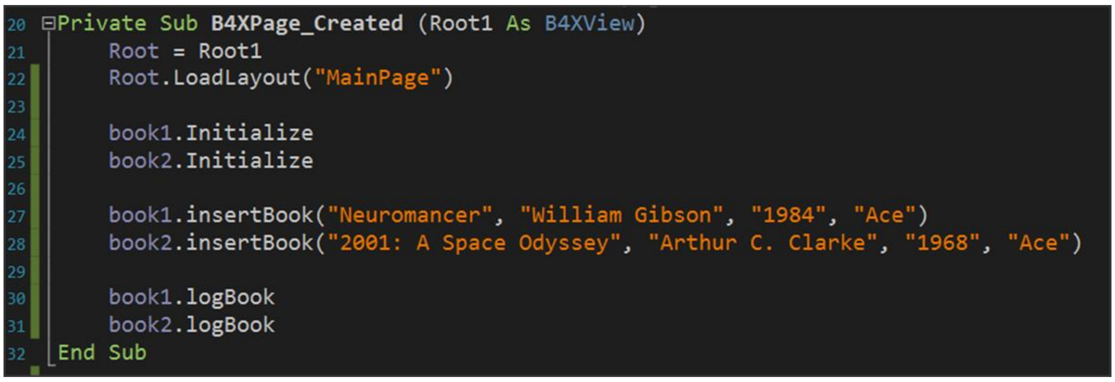
1. First create clsBook objects.



```
Sub Class_Globals
    Private Root As B4XView
    Private xui As XUI
    Private book1 As clsBook
    Private book2 As clsBook
End Sub
```

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

Use of methods



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

    book1.Initialize
    book2.Initialize

    book1.insertBook("Neuromancer", "William Gibson", "1984", "Ace")
    book2.insertBook("2001: A Space Odyssey", "Arthur C. Clarke", "1968", "Ace")

    book1.logBook
    book2.logBook
End Sub
```

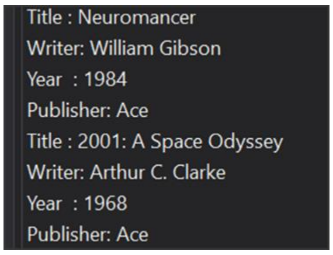
The first method to be used in objects is the initialize method.

Remember

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

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

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

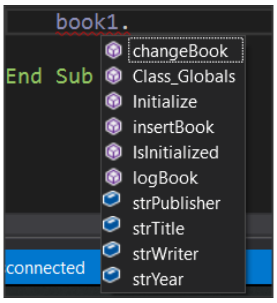


```
Title : Neuromancer
Writer: William Gibson
Year  : 1984
Publisher: Ace
Title : 2001: A Space Odyssey
Writer: Arthur C. Clarke
Year  : 1968
Publisher: Ace
```

Remember

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

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



```
book1.
End Sub
```

Exercises

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

and methods

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

2. If only one teacher teaches a specific course in a school implement the Course class with properties
   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 computers for sale. For each computer 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.

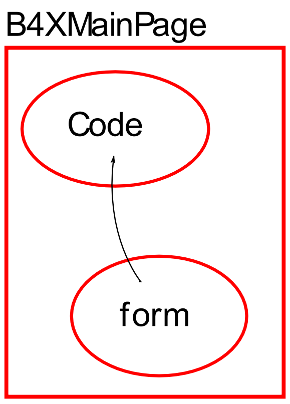
Lesson 10 – B4XPages

⏱ **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. Includes classes and procedures to create multiple application communication forms with the user. Also, it helps to transfer applications to different platforms using the B4A, B4i and B4J tools.



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

The structure of an application's folders

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

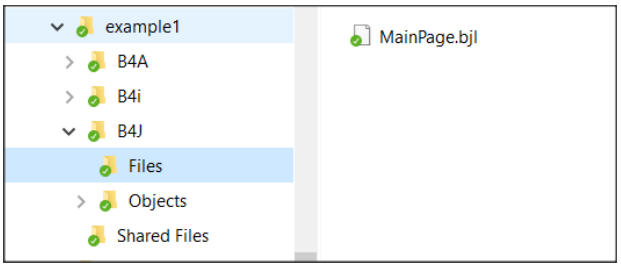


Figure 1 example1 folders

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

Specifically, in the **B4J** folder there is the “Files” folder where it contains all the files created with the Designer as well as any other files that must be used during the execution of the code e.g. images. **MainPage**.**bjl** is the file that was created automatically during the creation of the application and is its home screen. The Shared Files folder also includes files that the three different applications can share if the developer **chooses** to create an application for both Android, IOS and PC.

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

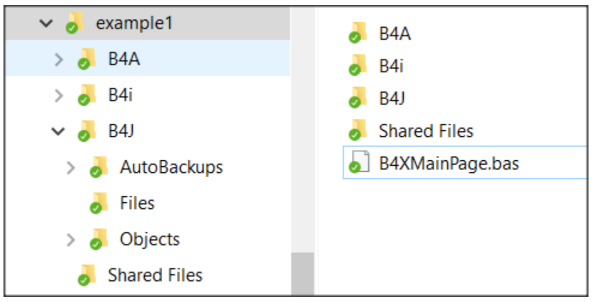


Figure 2 B4XPage files

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

Starting an application with B4XPage.

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

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

Public Sub Initialize

End Sub

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

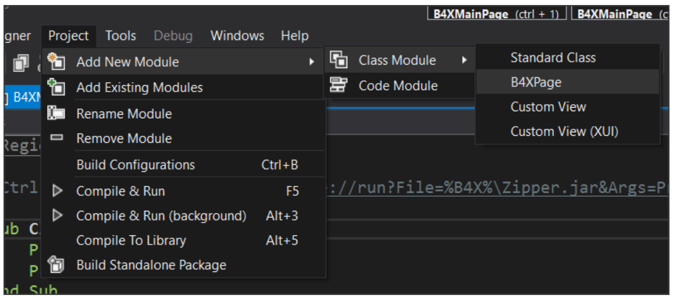
What is Root

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

Create a new B4XPage

Step 1.

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



Picture 22 Create B4XPage

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

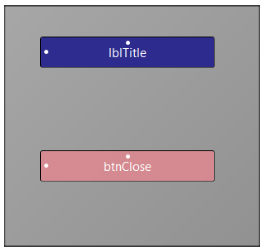
```
Sub Class_Globals
   Private Root As B4XView 'ignore
   Private xui As XUI 'ignore
End Sub

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

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

Picture 23 The new B4XPage

Step 2.

Open Designer and from the menu File select New.

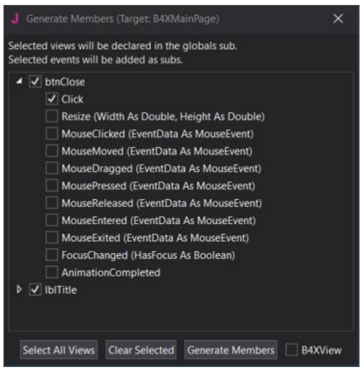
From the Variants tab, specify the dimensions of the form you want to design, and then select a label and button to insert into your form as in the Picture 24.



Picture 24 Form

Step 3.

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


Generate Members (Target: B4XMainPage)

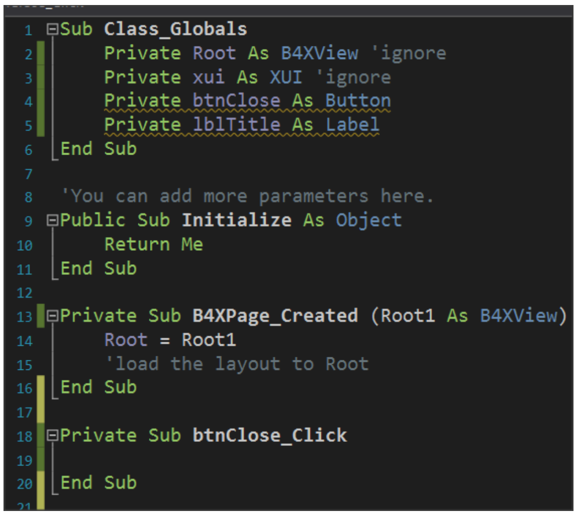
Selected views will be declared in the globals sub.
Selected events will be added as subs.

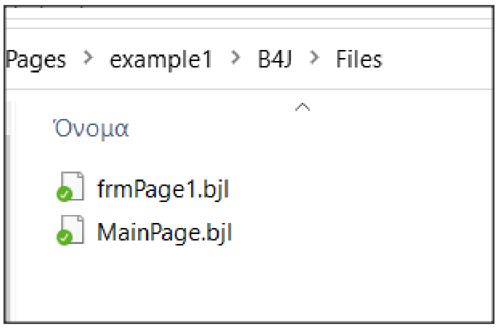
- [✓] btnClose
  - [✓] Click
  - [ ] Resize (Width As Double, Height As Double)
  - [ ] MouseClicked (EventData As MouseEvent)
  - [ ] MouseMoved (EventData As MouseEvent)
  - [ ] MouseDragged (EventData As MouseEvent)
  - [ ] MousePressed (EventData As MouseEvent)
  - [ ] MouseReleased (EventData As MouseEvent)
  - [ ] MouseEntered (EventData As MouseEvent)
  - [ ] MouseExited (EventData As MouseEvent)
  - [ ] FocusChanged (HasFocus As Boolean)
  - [ ] AnimationCompleted
- [✓] lblTitle

Picture 25 Generate Members

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

The following code (Picture 26) will now be created, and the file will have been displayed frmPage1.bjl in the folder files.


```
Sub Class_Globals
    Private Root As B4XView 'ignore
    Private xui As XUI 'ignore
    Private btnClose As Button
    Private lblTitle As Label
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

Private Sub btnClose_Click

End Sub
```


Pages › example1 › B4J › Files

Όνομα

frmPage1.bjl
MainPage.bjl

Picture 26 frmPage1

Step 4.

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

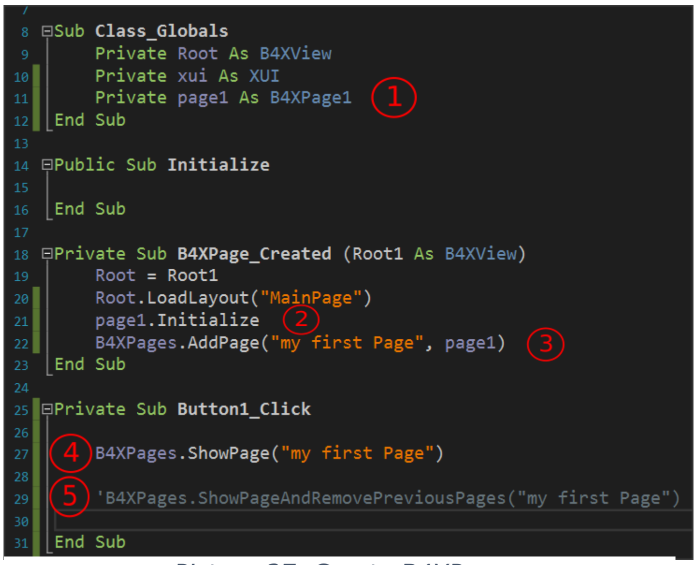
```
Private Sub B4XPage_Created  (Root1  As
B4XView)
   Root =  Root1
   'load the layout to Root
   Root.LoadLayout("frmPage1")
End Sub
```

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

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

Call a new B4Xpage.

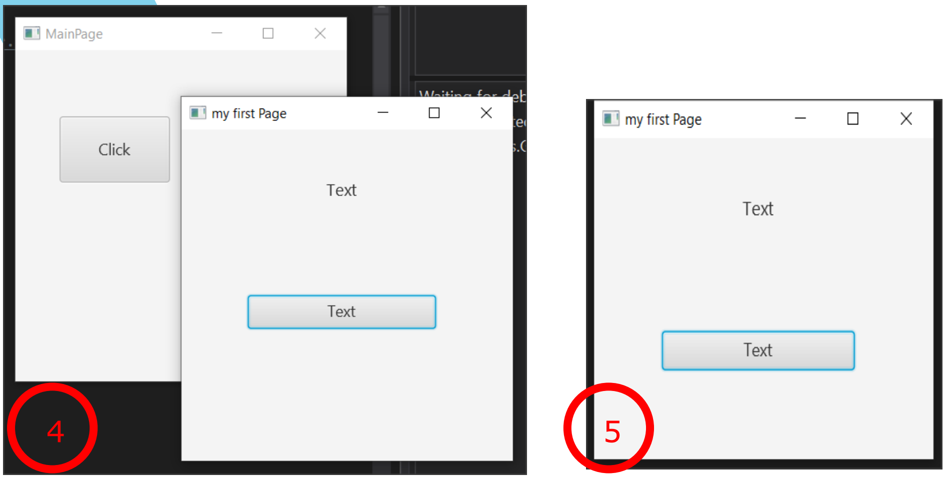
Each new B4XPage you create is a class. Therefore, before you can use it, you must create an object based on it. The creation is usually done in the B4XPage that will call the new one. So, in the previous example we write in B4XmainPage (Picture 27):



```
Sub Class_Globals
    Private Root As B4XView
    Private xui As XUI
    Private page1 As B4XPage1
End Sub

Public Sub Initialize

End Sub

Private Sub B4XPage_Created (Root1 As B4XView)
    Root = Root1
    Root.LoadLayout("MainPage")
    page1.Initialize
    B4XPages.AddPage("my first Page", page1)
End Sub

Private Sub Button1_Click

    B4XPages.ShowPage("my first Page")

    'B4XPages.ShowPageAndRemovePreviousPages("my first Page")

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")

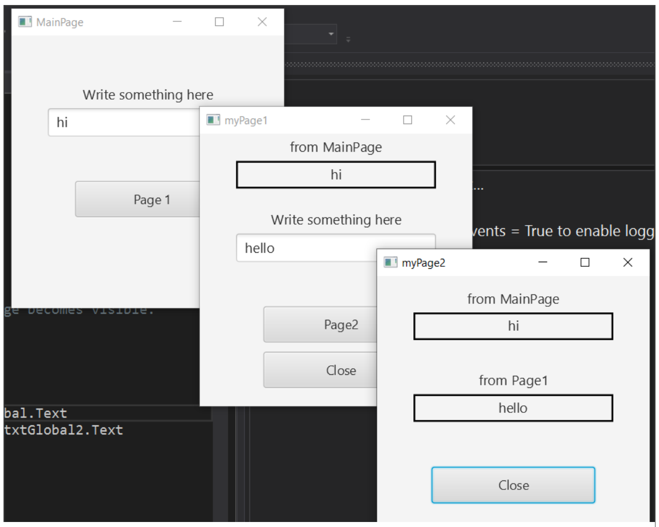
The first way closes the current form while the second way essentially calls the home page again while closing the current one.

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.

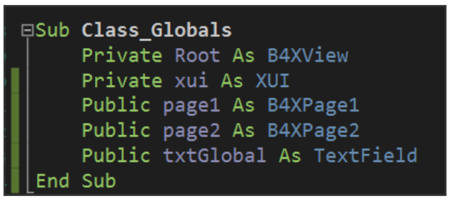
Example 2

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.

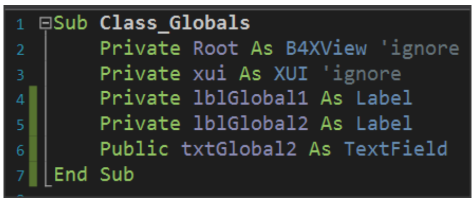


Picture 29 Example 2

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



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



```
Sub Class_Globals
    Private Root As B4XView 'ignore
    Private xui As XUI 'ignore
    Private lblGlobal1 As Label
    Private lblGlobal2 As Label
    Public txtGlobal2 As TextField
End Sub
```

Picture 30 Public declarations in MainPage and Page1

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

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

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

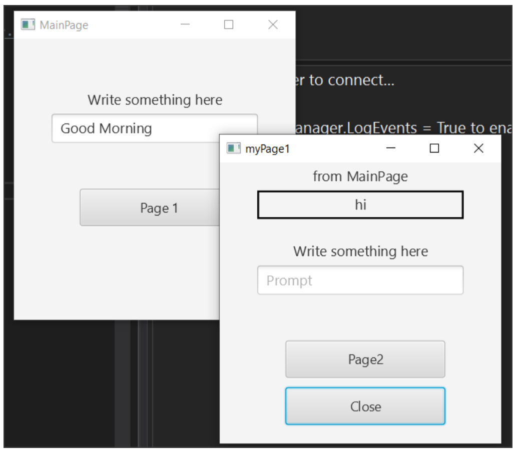
Similarly, Page2 has access to MainPage's txtGlobal1 and Page1's txtGlobal2 variables as follows:

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

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

The Life of B4XPages

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



Picture 31 B4XPage life

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

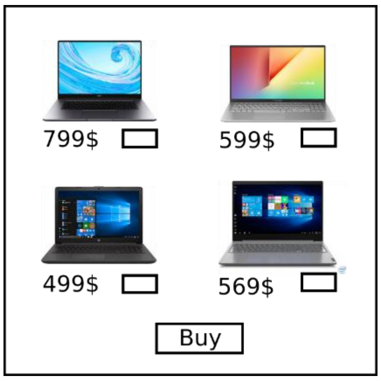
```
Private Sub B4XPage_Appear
   lblGlobal1.Text = B4XPages.MainPage.txtGlobal.Text
End Sub
```

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

Exercises

1. The little encyclopedia of dogs. Create an application where three different breeds of dogs are displayed on a home page and after the user clicks on the corresponding name display information about the breed along with two photos.
   You can use TextArea in designer to make bigger text areas with scroll bar.
2. Build an application that solves:
   a. the primary equation ax+b=0,
   b. the secondary equation ax a^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 and 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 the total quantity of the model)

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/.](https://inkscape.org/)

3. Create appropriate home screen with the following options.
- Sell a mobile phone.
- Warehouse 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 error 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 he 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 Status is selected, display mobile models on a new screen(B4XPage) along with the rest quantities in the warehouse.

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

At the next few pages, you will find, some documents to help students to work we this first project. In power point presentation there are all solution steps and you can find the code in folder lesson11. The solution is not "optimal" but a solution that students should understand.

Project	
Date	
Student	

Organize your time - What to do and when

Hours

	1	2	3	4	5	6	7	8	9

Wireframe Sketching

Form Name	
Date	
Student	

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:	
Public Variables:	1.
	2.
	3.
	4.
	5.
	6.
	7.
	8.
	9.
	10.
Event	
Description:	
Event	
Description:	
Event	
Description:	

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 poses a question. If the answer requires action, it runs. The same question is asked over and over again until no further action is required. Every time the question is asked it is called repetition.

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 to B4J would be as follows:

```
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.

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 it 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 value** 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:

Remember

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

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

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

Do While  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 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(-100, 100)
  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
                                   1
A = Rnd(-100, 100)
intMax = A         2
intMin = A
Do While  i  <=  9
  A = Rnd(-100, 100)

  If intMax  < A  then
    intMax = A
  End If                 3

  If intMin  > A  then
    intMax = A
  End If

  i = i +  1
Loop
```

1. First read a number outside the while.
2. Set as the starting value in the intMax end 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 unknow 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. 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)      2
  intSum = intSum + A
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(-100, 100)
   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(-100, 100)
intMax = A
intMin = A
Do Until  I  >  9
   A = Rnd(-100, 100)

   If intMax  < A  then
      intMax = A
   End If

   If intMin  > A  then
      intMax = 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:

- Do not needs to initialize the counter.
- Do not needs 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 for

Example 1

Show all numbers from 100 to 1

```
Private I As Int

For i = 100 to 1 step -1
   Log(i)
Next
```

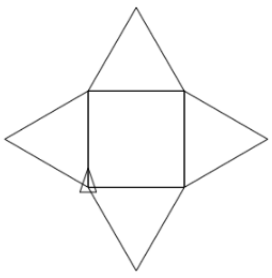
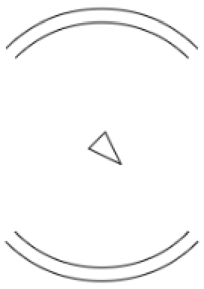
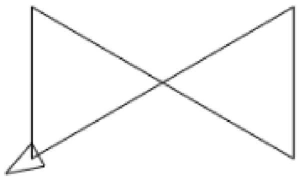
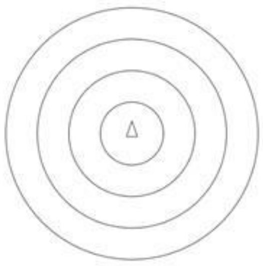
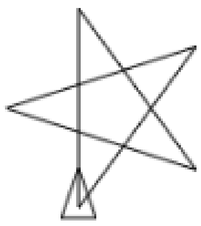
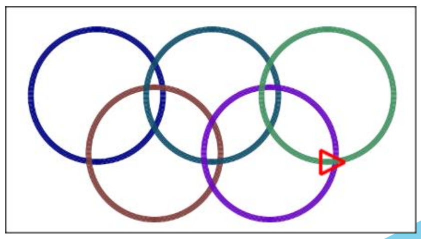
Example 2

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+...+K.
2. Write a program that the user will give numbers and calculate the average of the above numbers. The program ends by entering zero.
3. A consumer wants to buy toys for gifts to 10 children. It also wants their total price not to exceed 200€. Make a program that:
    a. Enter the cost of the game (random number between 10-50 ,command Rnd(10, 50))
    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 enter in an appropriate field. The program will include a design start button and a screen cleaning button that when pressed cleans the screen and the turtle will placed 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

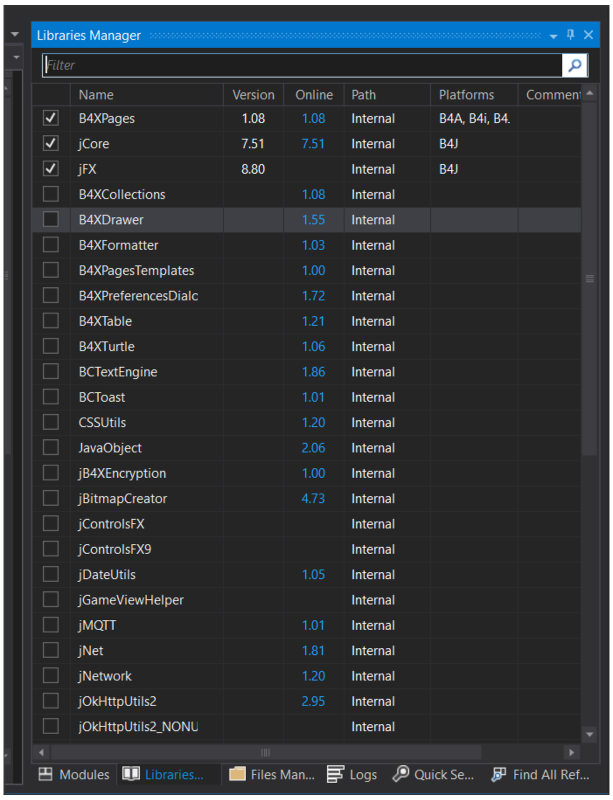
Each programming language has libraries. In general, a library includes collections of code, structures, classes, methods that can be used by developers to facilitate the process of developing a program.

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.


Libraries Manager

	Name	Version	Online	Path	Platforms	Comment
✓	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

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.

External libraries should always be copied all in 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 B4X :)
  - 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 her 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

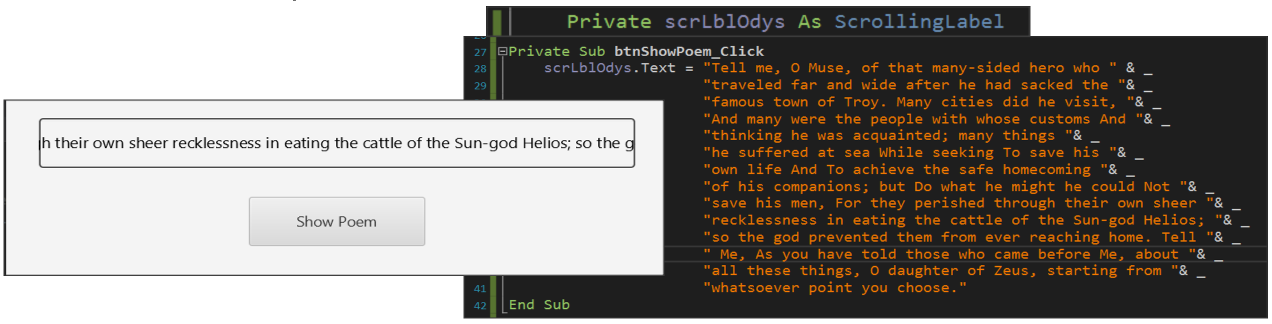
AnotherProgressBar

B4XSwitch

B4XImageView

Scrolling Label

Scrolling label is a text label where it can move text so that it can appear in its entirety.



```
Private scrLblOdys As ScrollingLabel
```

```
Private Sub btnShowPoem_Click
    scrLblOdys.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."
End Sub
```

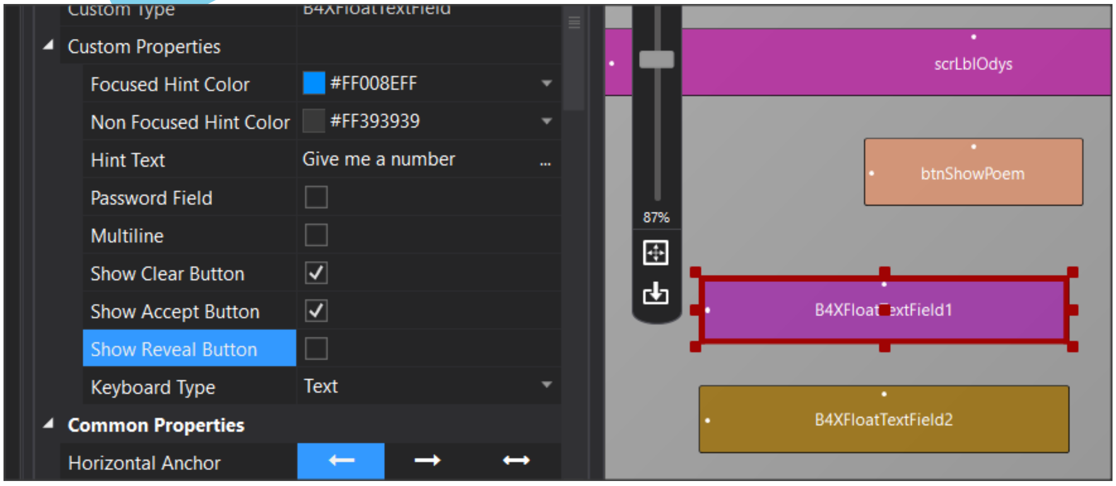
h their own sheer recklessness in eating the cattle of the Sun-god Helios; so the g

Picture 34 Scrolling Label

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

B4XFloatTextField

B4XFolatTextField 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).

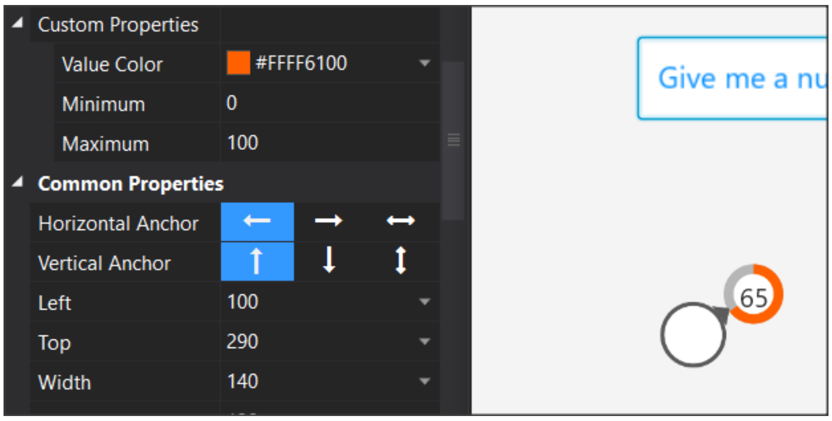

Custom Type B4XFloatTextField
Custom Properties
Focused Hint Color #FF008EFF
Non Focused Hint Color #FF393939
Hint Text Give me a number
Password Field ☐
Multiline ☐
Show Clear Button ✓
Show Accept Button ✓
Show Reveal Button ☐
Keyboard Type Text
Common Properties
Horizontal Anchor

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. 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.


Custom Properties
Value Color #FFFF6100
Minimum 0
Maximum 100
Common Properties
Horizontal Anchor
Vertical Anchor
Left 100
Top 290
Width 140

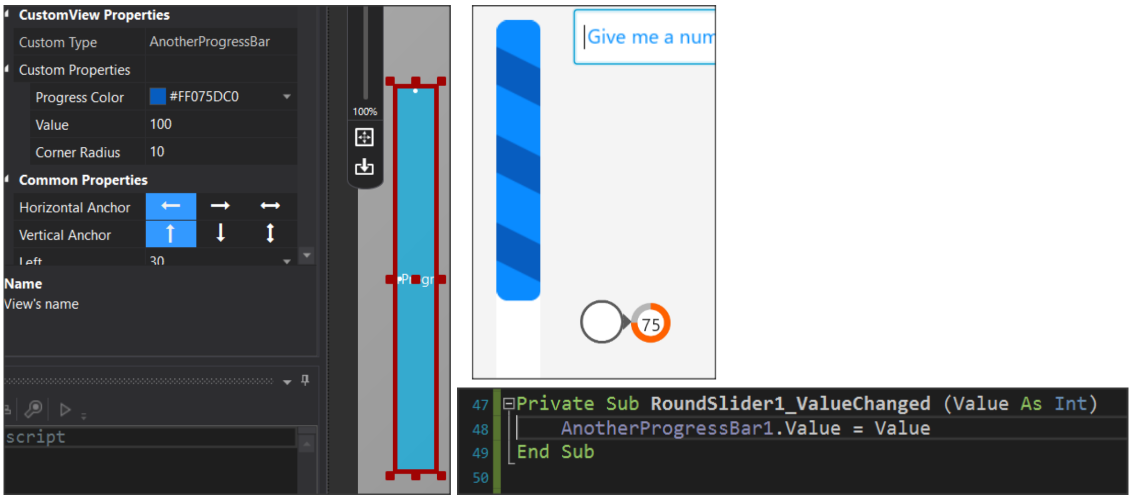
Picture 36 RoundSlider

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 vary 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 of the image, progressBar vary depending on the value it receives from RoundSlider.



CustomView Properties

Custom Type	AnotherProgressBar
Custom Properties	
Progress Color	#FF075DC0
Value	100
Corner Radius	10
Common Properties	
Horizontal Anchor	
Vertical Anchor	
Left	30

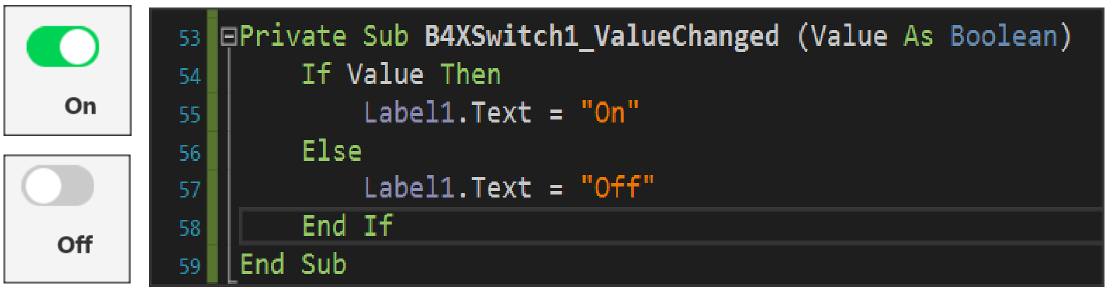
Name
View's name

```
Private Sub RoundSlider1_ValueChanged (Value As Int)
    AnotherProgressBar1.Value = Value
End Sub
```

Picture 37 AnotherProgressBar

B4XSwitch

B4XSwitch creates a sliding button on the screen.



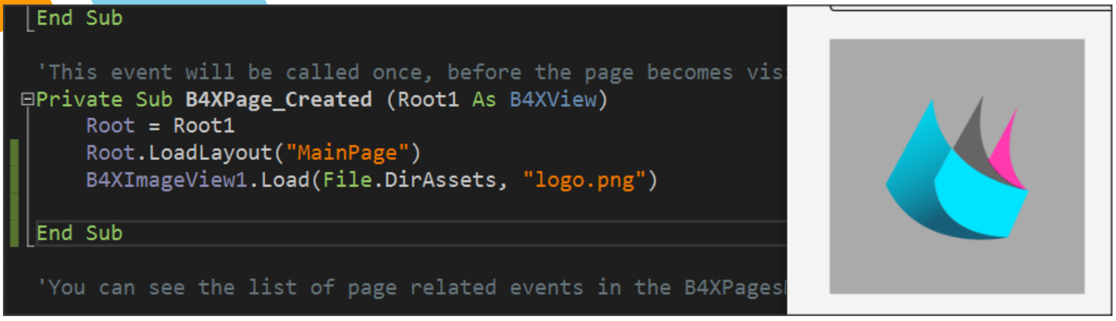
```
Private Sub B4XSwitch1_ValueChanged (Value As Boolean)
    If Value Then
        Label1.Text = "On"
    Else
        Label1.Text = "Off"
    End If
End Sub
```

Picture 38 B4XSwitch

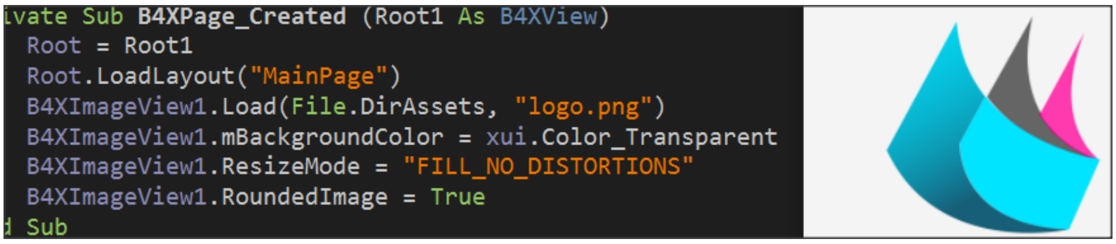
In the example of the image when the returned value is true then a label displays the text "On" otherwise displays "Off".

B4XImageView

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



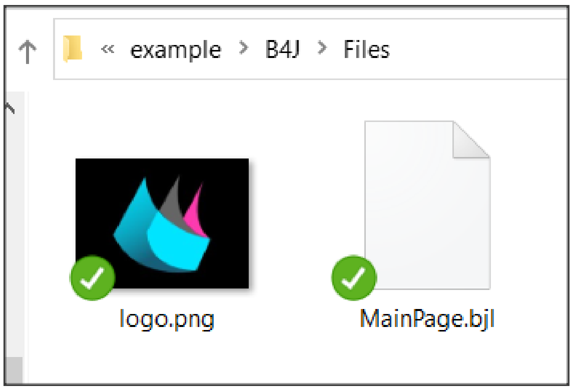
```
End Sub

 'This event will be called once, before the page becomes vis
Private Sub B4XPage_Created (Root1 As B4XView)
    Root = Root1
    Root.LoadLayout("MainPage")
    B4XImageView1.Load(File.DirAssets, "logo.png")

End Sub

 'You can see the list of page related events in the B4XPages
```



```
ivate Sub B4XPage_Created (Root1 As B4XView)
  Root = Root1
  Root.LoadLayout("MainPage")
  B4XImageView1.Load(File.DirAssets, "logo.png")
  B4XImageView1.mBackgroundColor = xui.Color_Transparent
  B4XImageView1.ResizeMode = "FILL_NO_DISTORTIONS"
  B4XImageView1.RoundedImage = True
d Sub
```

Picture 39 B4XImageView

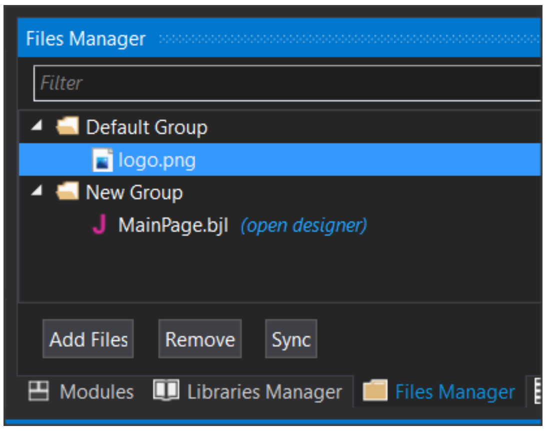
In the Load command, in addition to the image name, the folder where the image is stored must be declared first. In this case, the Files (Files.DirAsset) folder is declared as in the image below:



Picture 40 Files Folder (DirAsset)

This folder is not accessible for writing new files when the application is created. In the course of the files, we will refer in greater detail to the folders.

To display an image stored in the Files folder, it must also be declared on the File Manager tab of the language.



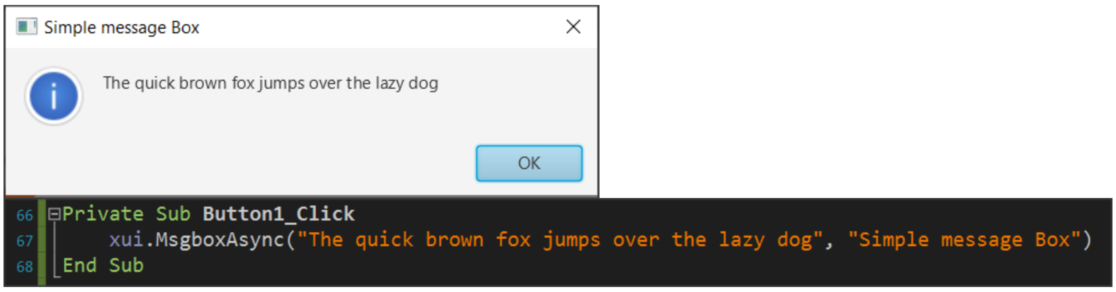
Picture 41 Files Manager

B4XDialogs

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

MsgBoxAsync

Already in previous courses you used the command xui.MSgBoxAsync which displays a simple message on the screen.



The quick brown fox jumps over the lazy dog

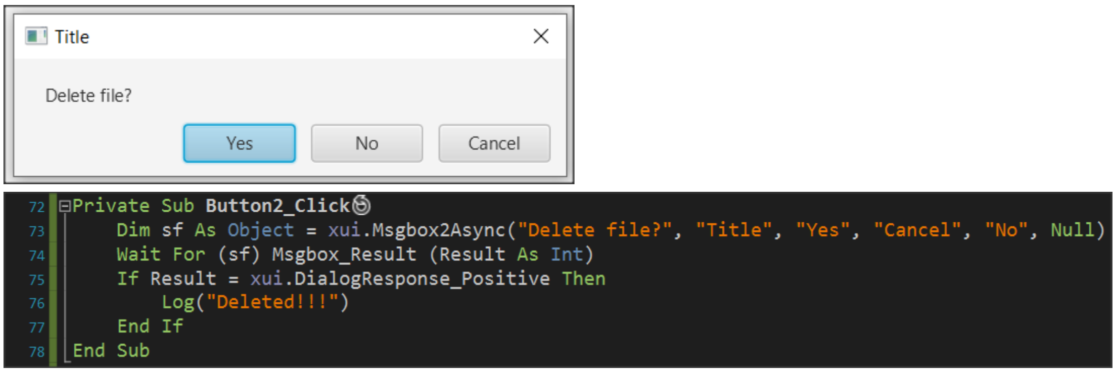
```
Private Sub Button1_Click
    xui.MsgboxAsync("The quick brown fox jumps over the lazy dog", "Simple message Box")
End Sub
```

Picture 42 MsgBoxAsync

As shown in Picture 42 window. The command also displays an OK button to close the message and return to the program.

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.



Delete file?

```
Private Sub Button2_Click
    Dim sf As Object = xui.Msgbox2Async("Delete file?", "Title", "Yes", "Cancel", "No", Null)
    Wait For (sf) Msgbox_Result (Result As Int)
    If Result = xui.DialogResponse_Positive Then
        Log("Deleted!!!")
    End If
End Sub
```

Picture 43 MsgBox2Async

MsgBox2Async returns an object ("sf" in the example) that waits for a response event from user.

- **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 command 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 when 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>
```

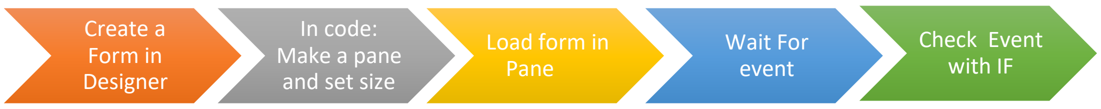
Where **sender** is the object for which an event is waiting to be activated and

Event signature the event that was activated where in the case of MsgBox2 Async are the DialogResponse_Positive, DialogResponse_Cancel, DialogResponse_Negative.

The event is then checked with an if command.

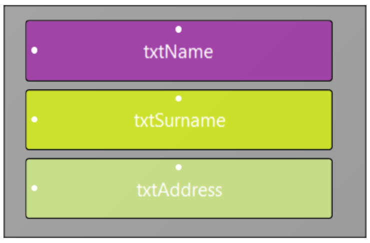
CustomDialog

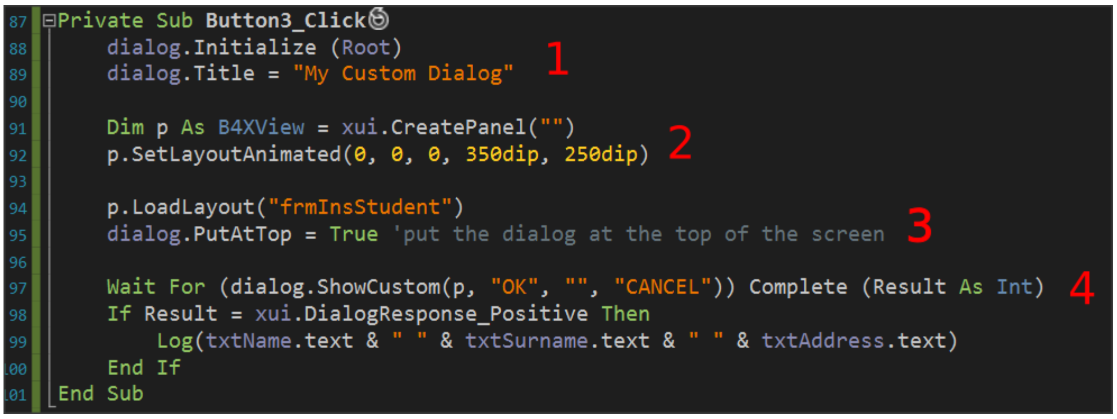
Creating a Custom dialog box includes several steps that start with the Designer.



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.





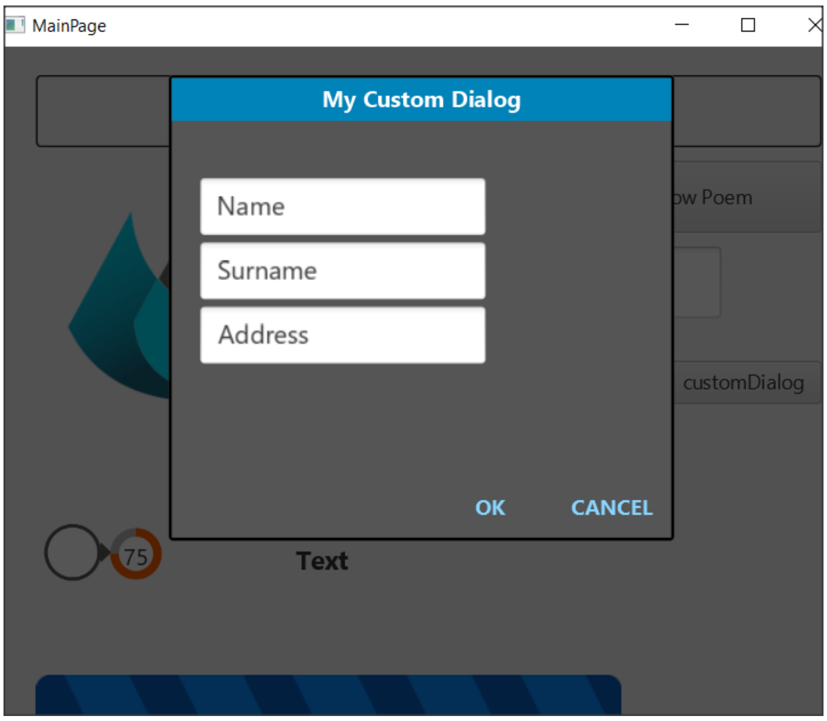
```
Private Sub Button3_Click
    dialog.Initialize (Root)
    dialog.Title = "My Custom Dialog"       1

    Dim p As B4XView = xui.CreatePanel("")
    p.SetLayoutAnimated(0, 0, 0, 350dip, 250dip)   2

    p.LoadLayout("frmInsStudent")
    dialog.PutAtTop = True 'put the dialog at the top of the screen   3

    Wait For (dialog.ShowCustom(p, "OK", "", "CANCEL")) Complete (Result As Int)   4
    If Result = xui.DialogResponse_Positive Then
        Log(txtName.text & " " & txtSurname.text & " " & txtAddress.text)
    End If
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” sets a title in the dialog box that you are about to create.

Anywhere Software

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" 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:

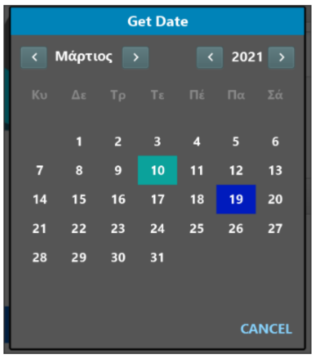
- B4XDateTemplate
- B4XColorTemplate
- B4XLongTextTemplate

Teacher's tip.

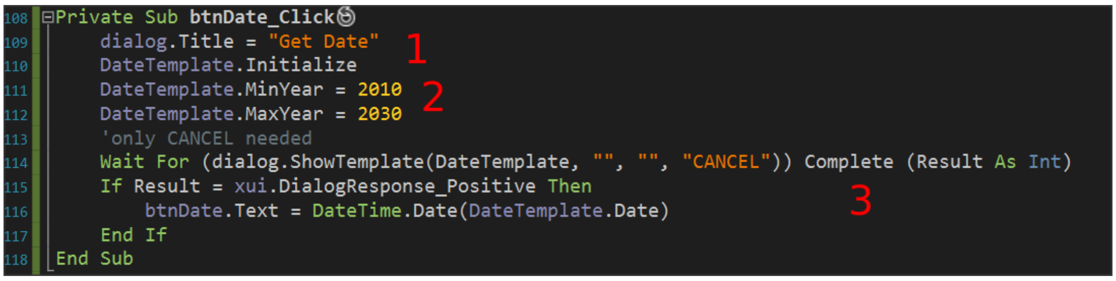
Additional functions will be used in next chapters as concepts such as lists, dictionaries, etc. have not yet been discussed.

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 initialised as in the Custom Dialogs



```
Private Sub btnDate_Click
    dialog.Title = "Get Date"
    DateTemplate.Initialize
    DateTemplate.MinYear = 2010
    DateTemplate.MaxYear = 2030
    'only CANCEL needed
    Wait For (dialog.ShowTemplate(DateTemplate, "", "", "CANCEL")) Complete (Result As Int)
    If Result = xui.DialogResponse_Positive Then
        btnDate.Text = DateTime.Date(DateTemplate.Date)
    End If
End Sub
```

Picture 47 Create Date Dialog Box

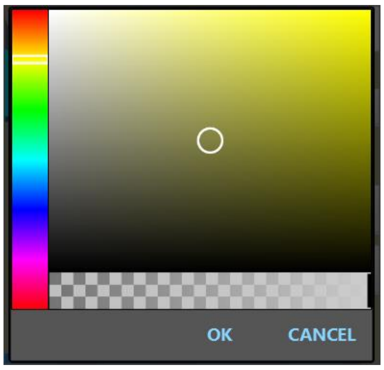
1. Initialize and set a title in the dialog. In the example initialization done in Class_Globals Sub.
2. Specify whether you want years limits.
3. Wait for the date to be selected, and if selected, display it as a name on the "btnDate" or use it according to your needs.

Remember

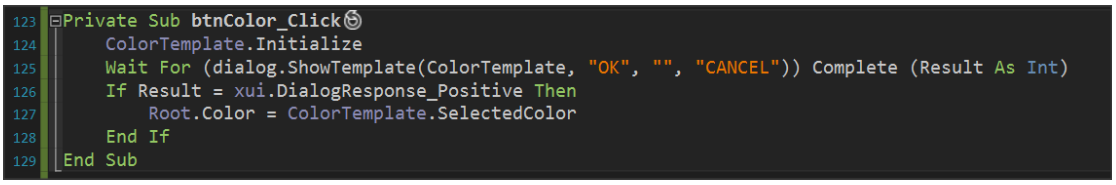
The date is returned to Ticks which represent milliseconds from 1/1/1970. These milliseconds are converted to a date with the "DateTime.Date" command.

B4XColorTemplate

B4XColorTamplate is similar to that of the day. 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".



```
Private Sub btnColor_Click
    ColorTemplate.Initialize
    Wait For (dialog.ShowTemplate(ColorTemplate, "OK", "", "CANCEL")) Complete (Result As Int)
    If Result = xui.DialogResponse_Positive Then
        Root.Color = ColorTemplate.SelectedColor
    End If
End Sub
```

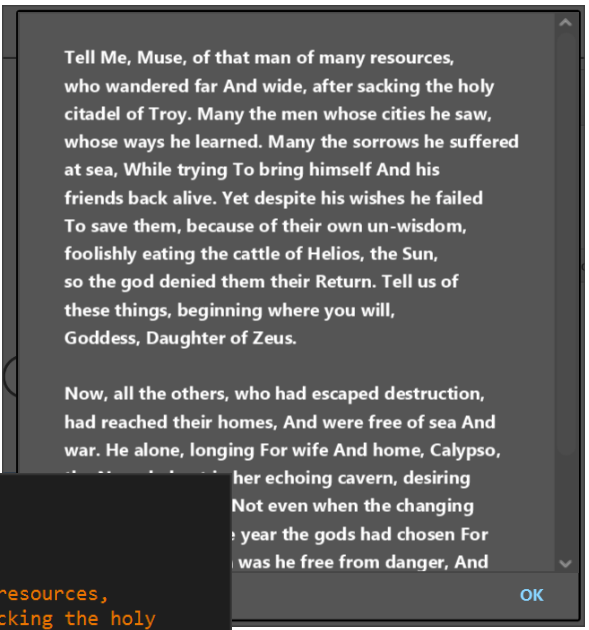
Picture 48 Create Color Template

B4XLongTextTemplate

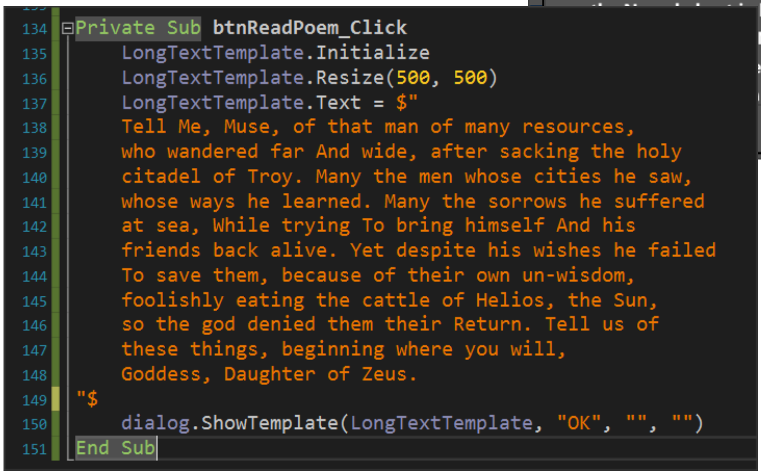
B4XLongTextTemplate displays a window on the screen to display long text. Also, it displays a navigation bar to scroll the text.

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



```
Tell Me, Muse, of that man of many resources,
who wandered far And wide, after sacking the holy
citadel of Troy. Many the men whose cities he saw,
whose ways he learned. Many the sorrows he suffered
at sea, While trying To bring himself And his
friends back alive. Yet despite his wishes he failed
To save them, because of their own un-wisdom,
foolishly eating the cattle of Helios, the Sun,
so the god denied them their Return. Tell us of
these things, beginning where you will,
Goddess, Daughter of Zeus.

Now, all the others, who had escaped destruction,
had reached their homes, And were free of sea And
war. He alone, longing For wife And home, Calypso,
her echoing cavern, desiring
Not even when the changing
year the gods had chosen For
was he free from danger, And
```



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

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 display 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 display a bar, represent the height. Text field will not allow any texts but only decimals values.
    B. If water height is above 5 meters show message "Danger Water Leak!".
    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, 100 students and 100 grades, 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 tables. In general, the table 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 Table named "Grades" and 14 places for grades

In Picture 50 you can see an example table 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 table, should just indicate its name and then write, 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 table, they simply write "Grades(0)" where 0 is the first position of the table.

Declaring Tables

A table is declared like other variables.

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

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

Functions in tables

Insert items into a table.

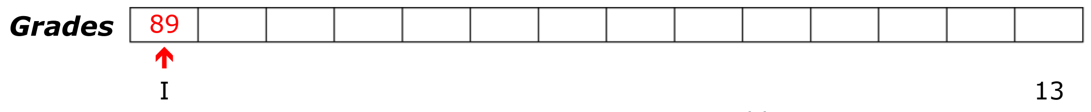
To insert an item into a table, you only need to assign a value to the corresponding place in the table. E.g.

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

The process can continue the same way, but it is easier to create a repeating process to fill the table. In the case of the Grade table 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 table with random numbers from 1 to 100. Notice that the position measurement starts at number 0. The variable **i** which



Picture 51 Insert items into a table

identifies each time the position of the table we use is called **Index** of the table. Moving the index i with a repeat command you can access each location in the table.

A second way to insert items into a table is as follows:

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

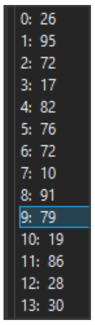
This size of the table is not specified in its statement, but during a fill by inserting items with the **Array command.**

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

Display items in a table.

Using the Log command, you can easily print an item in a table or the entire table 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
```

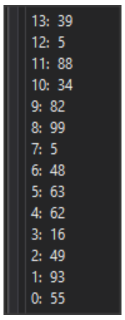
The example above starts an iteration that uses an i index to display the current index value of the table.

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 table boundaries.

Show items in reverse order

The following code shows the table 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 to display or use any items in the table you want.

Find Total and Average Table Items

The rules applicable to repetitive procedures for algorithmic techniques generally apply to tables 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 table.

```
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 table makes use of an additional variable commonly called Max or Min, respectively. This variable initially assigns the developer the first value of the table, and then checks all other values with Max (or Min). 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 a table refers to scanning a table in search of an item that meets a specific condition.

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

Serial Search

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

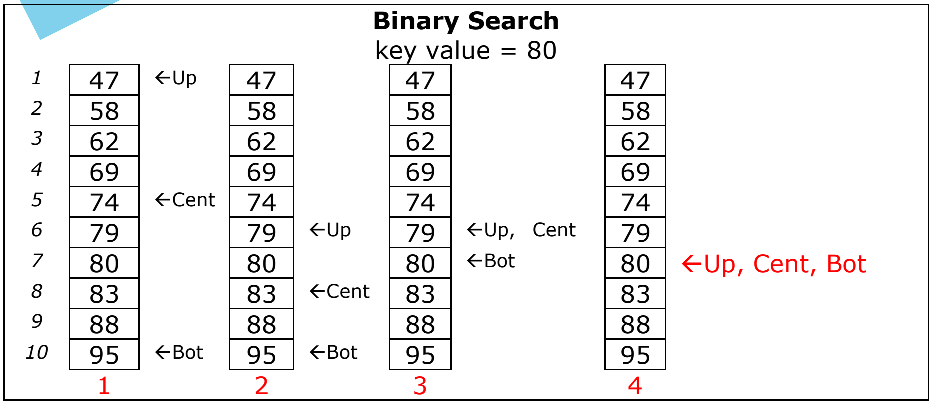
```
'Find all positions with key value
For i = 0  To  999
   If Grades(i) = key  Then
      Log("found in : " & i & "position")
   End If
Next
```

When it is necessary to find the first location that a value is displayed, 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 tables. 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 table. Contrarywise, if it is greater, the bottom half is searched. In the following example, there is a Grade table with 10 values sorted in ascending order.



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

Sort

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

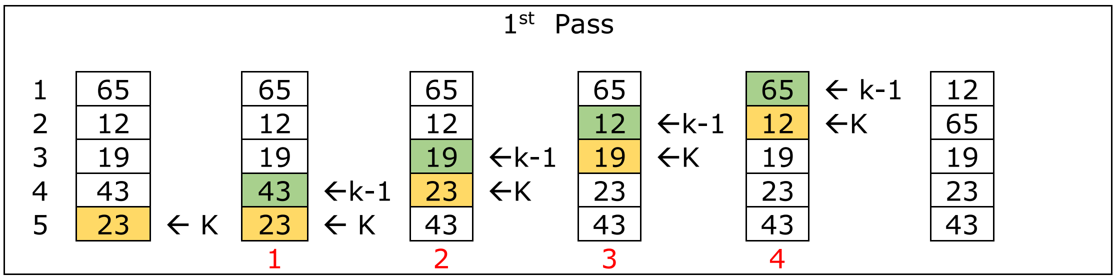
Bubble sorting

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

A table named Grades with 5 integer grades is given.

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



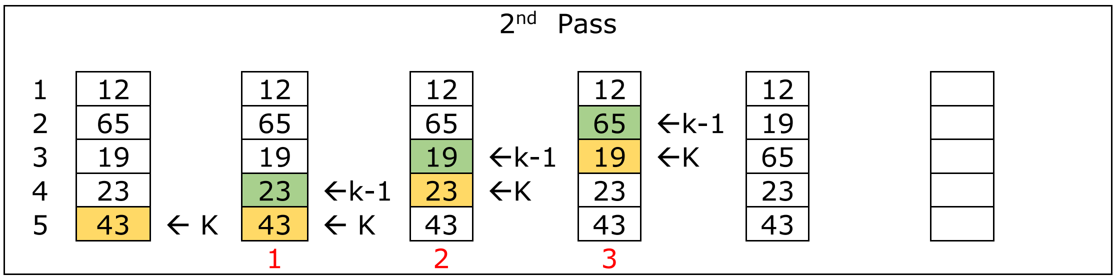
Initially, the algorithm starts from the last position of the table and compares sequentially with the previous

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

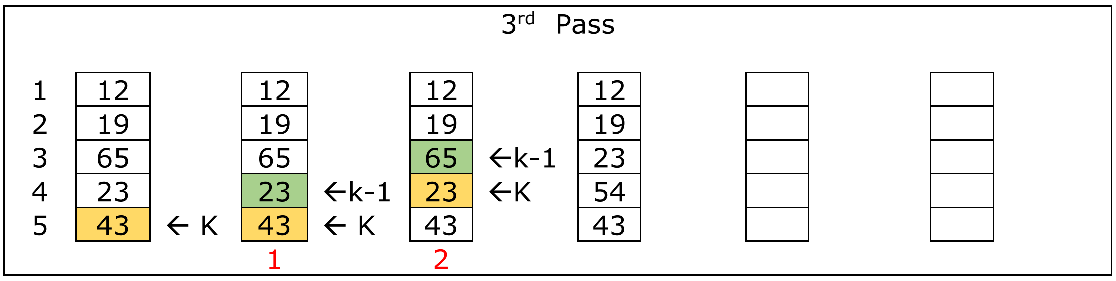
The first pass is implemented with the following code.

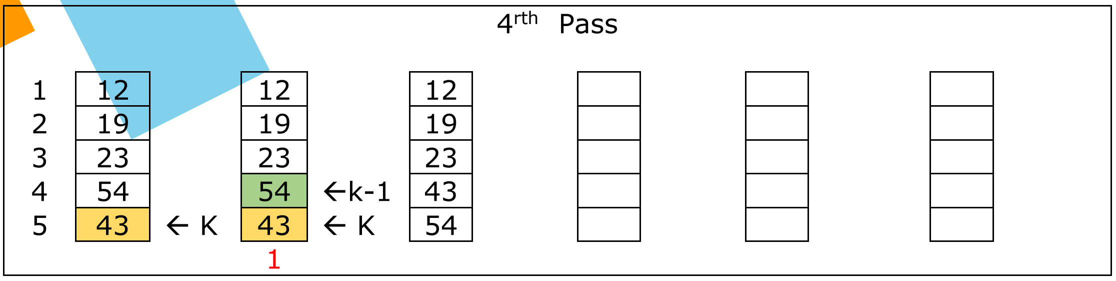
```
Private Grades()  As  Int
Private temp  As Int
Grades = Array As Int(19,43,12,65,23)
For k = 4  To  1  Step  -1
   If Grades(k)  <  Grades(k-1)  Then
      temp =  Grades(k)
      Grades(k) =  Grades(k-1)
      Grades(k-1) =  temp
   End If
Next
```

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



In the second pass of the table, the same procedure is performed except that the checks between the cells end up up to the 2nd position after the smallest element has already been climbed to the first.





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

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

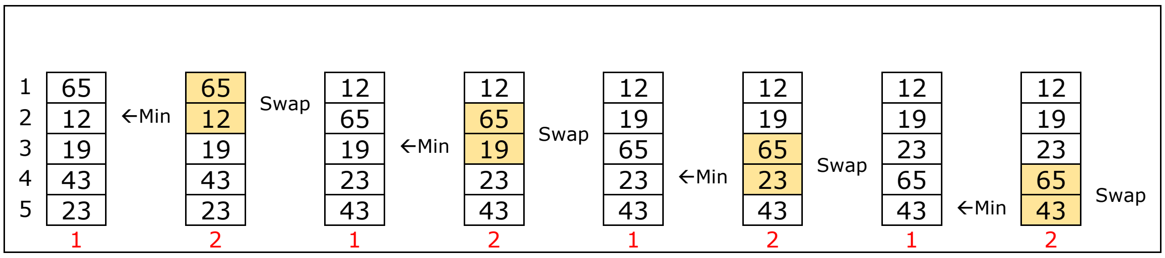
```
Private Grades()  As  Int
Private temp  As Int
Grades = Array As Int(19,43,12,65,23)
For i  = 1  to  4       'i  counts the different pass
  For k = 4  To  i  Step  -1
    If Grades(k)  <  Grades(k-1)  Then
      temp =  Grades(k)
      Grades(k) =  Grades(k-1)
      Grades(k-1) =  temp
    End If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

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

2.

   a. The average temperature per day for one year is stored in a table Temperature(365). If you consider these temperatures to be integer values between 1 and 40 ^{o}C, find and display the frequency of each temperature.
   b. In the previous table **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

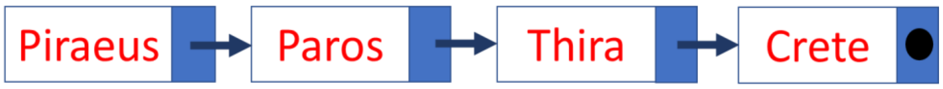
A list in B4X is declared as follows:

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

Where Islands 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.

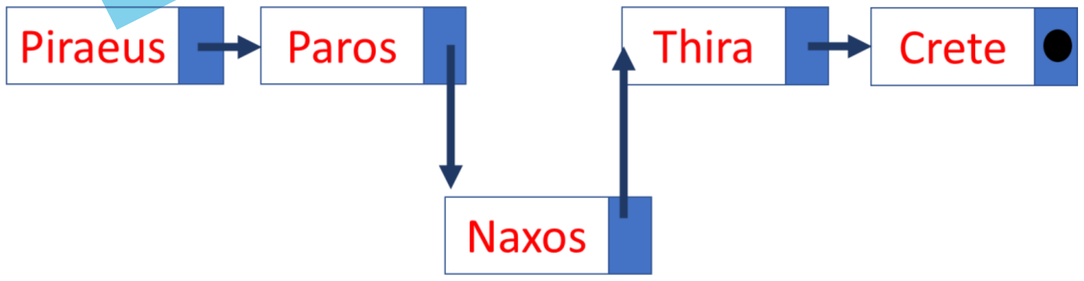
```
Islands.AddAll(Array as String("Piraeus", "Paros", "Naxos", "Crete"))
```

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 a third you can use the **InsertAt** property

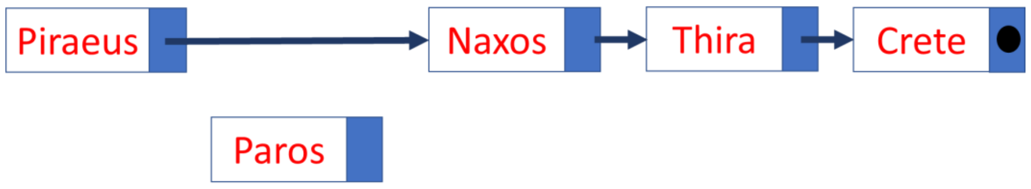


```
Islands.InsertAt(2, "Naxos")
```

In the above example is inserted between Paros and Thira the island of Naxos in 3rd place.

Remove an item from a specific location

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



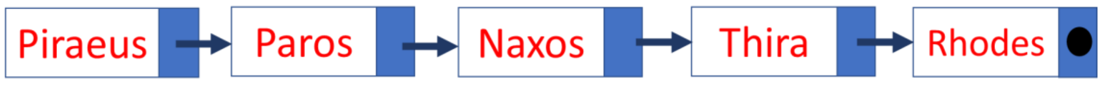
```
Islands. RemoveAt(1)
```

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

Change position value

With Set command you to change one value of a location with another value that you enter.

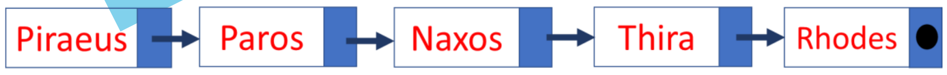
```
Islands.Set(4, "Rhodes")
```



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

More commands in lists

1. List size.

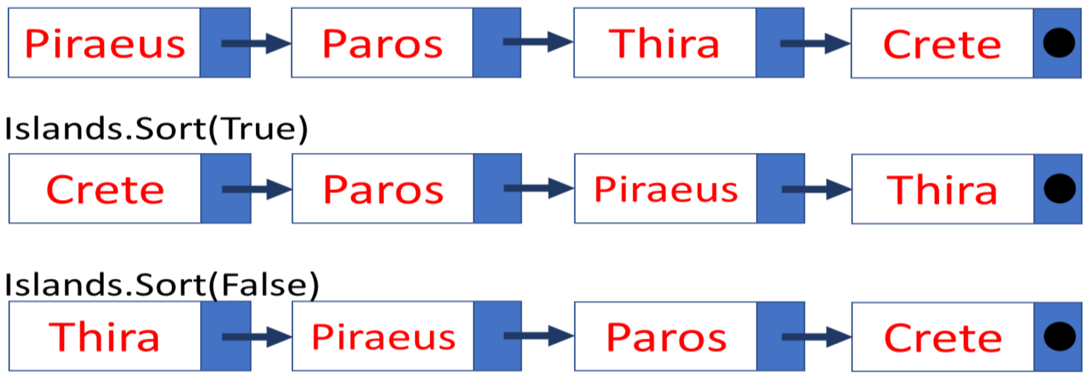


```
Islands.Size
```

In the list of the image 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
```



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(j) 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.