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 1 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.

```
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 2 Calling a subprogram

The subprogram is written before or after the current 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

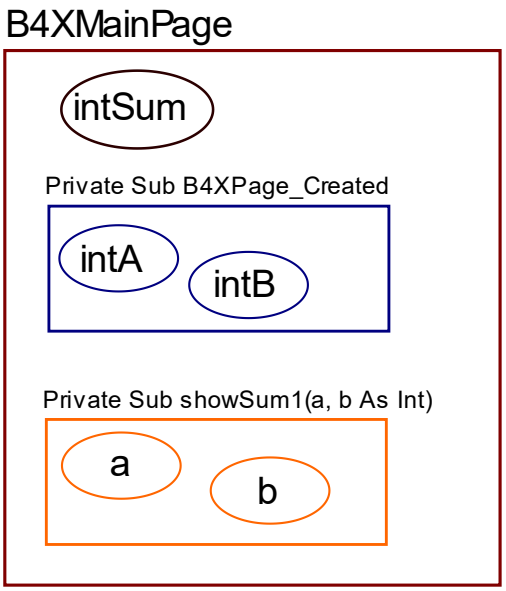
Variables exchanged between subprograms during their call are called **parameters**.

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
```



Picture 3 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.

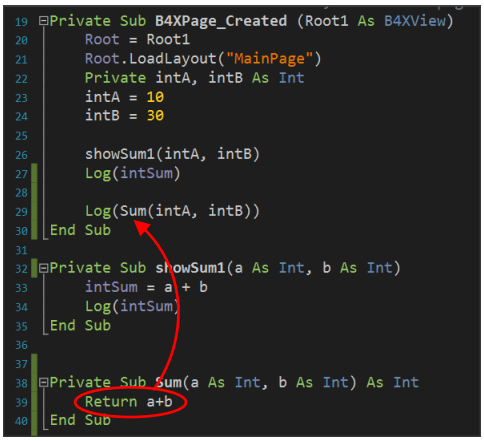
Remember

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:



Picture 4 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.

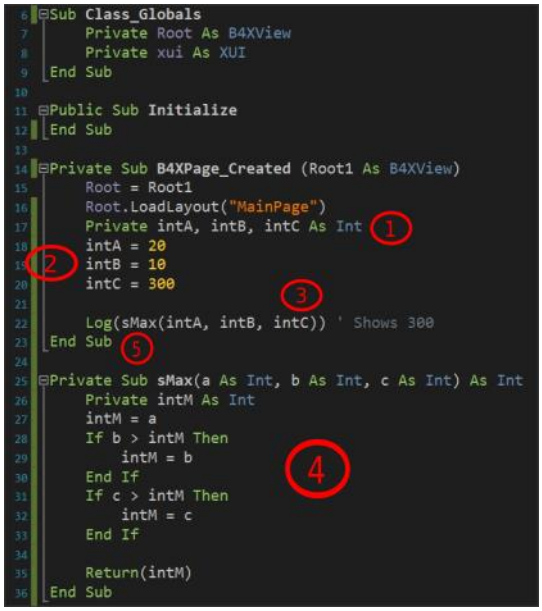
Finally, the code that called the subprogram accepts back the value calculated and can use it like any variable.

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.



Picture 5 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.

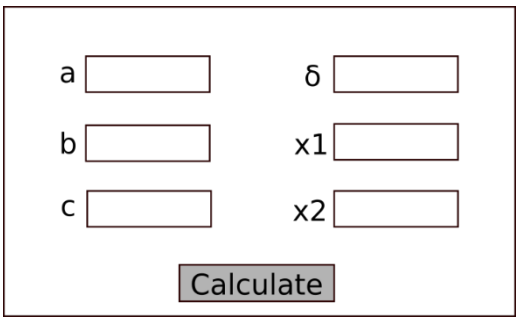
Exercises

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 7 Wireframe

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

The quadratic formula for the roots of the general quadratic equation

Picture 6 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