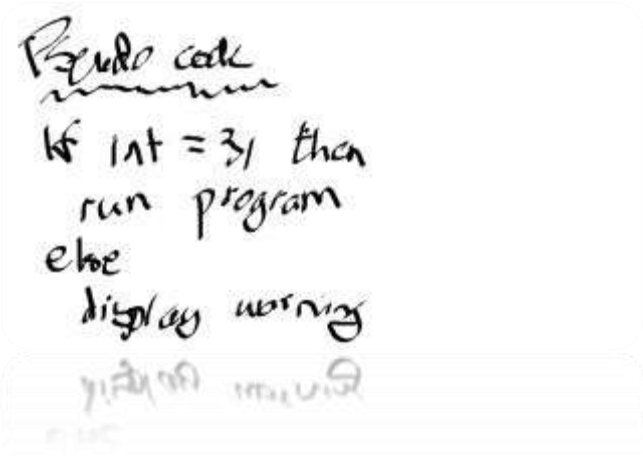
the basics of vb.net

Declaring variables

The declaration of variables is very important. We are firstly going to look at the types of variable we can have:

1. Integer - a 32 bit numerical value with no decimal places
2. String - a text value made of characters
3. Single - a numerical value up to 32 bits in size with decimal places
4. Char - a character value that represents one character alone
5. Short - a 16 bit number with no decimal places
6. Double - a 64 bit number with decimal places
7. Long - a 64 bit number without decimal places
8. Image - an image format
9. Bitmap - an image format
10. Boolean - a simple True or False value

There are more variables than these, but for this article we will only discuss these as possible variables.



Variables can be declared globally or locally. A global variable can be accessed from any section of the program whilst a local may only be accessed from the part of the program that it was declared within.

To declare a variable locally, we just use the `Dim` command. When declaring publicly or globally, we use either `Global` or `Public`.

Using variables

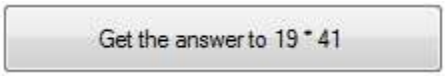
All variables can have operations performed on them. This is because underneath a variable, it is a class. A class is a collection of commands and values. So for instance, if we declare a class with the name `Sample` and then declared a sub routine underneath it with the name `SubRoutine`, we can use `Sample.SubRoutine` to perform an operation on the variable.

Controls

A control is also a class with a design, but it can be placed on the form. For instance, a button is a control which has many different properties and methods behind it. There are many controls such as:

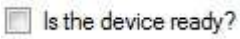
The button

Possibly the most commonly used control. The button has many different uses, but for the majority of the time, users just need to click it once to activate its code.



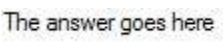
The checkbox

The checkbox allows a Boolean variable to be passed through the user as a control. The checkbox basically only has two values, on or off, true or false.



The label

The label is used to display information as a string. It is a very widely used control.



The listbox

The listbox allows the user to select an item from a list to allow an item to be worked on.

The picturebox

Does what it says on the tin. The picturebox is used for showing an image. In older versions of Visual Basic such as VB6, the computer can print (using the painting method of the system) text on to the picturebox. This was removed in VB.NET.

Events, methods and properties of controls

A **property** is used to describe the state of an object on the screen, namely a control. If we use as an example, a dog, we can clearly see the following are properties of the dog:

- Age
- Breed
- Color
- Size

In the same sense, controls have properties.

A **method** is what the object can do. So going back to that analogy of the dog, we can see that the dog can:

- Crouch
- Jump
- Run
- Walk

These are known as methods. We can make the dog perform a jump, or make it run, so it is known as a method.

Finally an **event** is what reaction something has when the object does something. For instance, if we have the event set as Jump, when the dog jumps it will trigger a response. This is known as the event. Generally speaking, events are similar to methods, so some example events for the dog could be:

- Crouch
- Jump
- Run
- Walk

However, you could take the event section a bit further and make it only trigger a response when the dog does something such as enter a building or run into the house.

Properties of a button

BackColor
ForeColor
Location
Size
Text

They are just a few properties of a button.

Methods of a button

CreateGraphics
Show
Visible = True
Visible = False

Events of a button

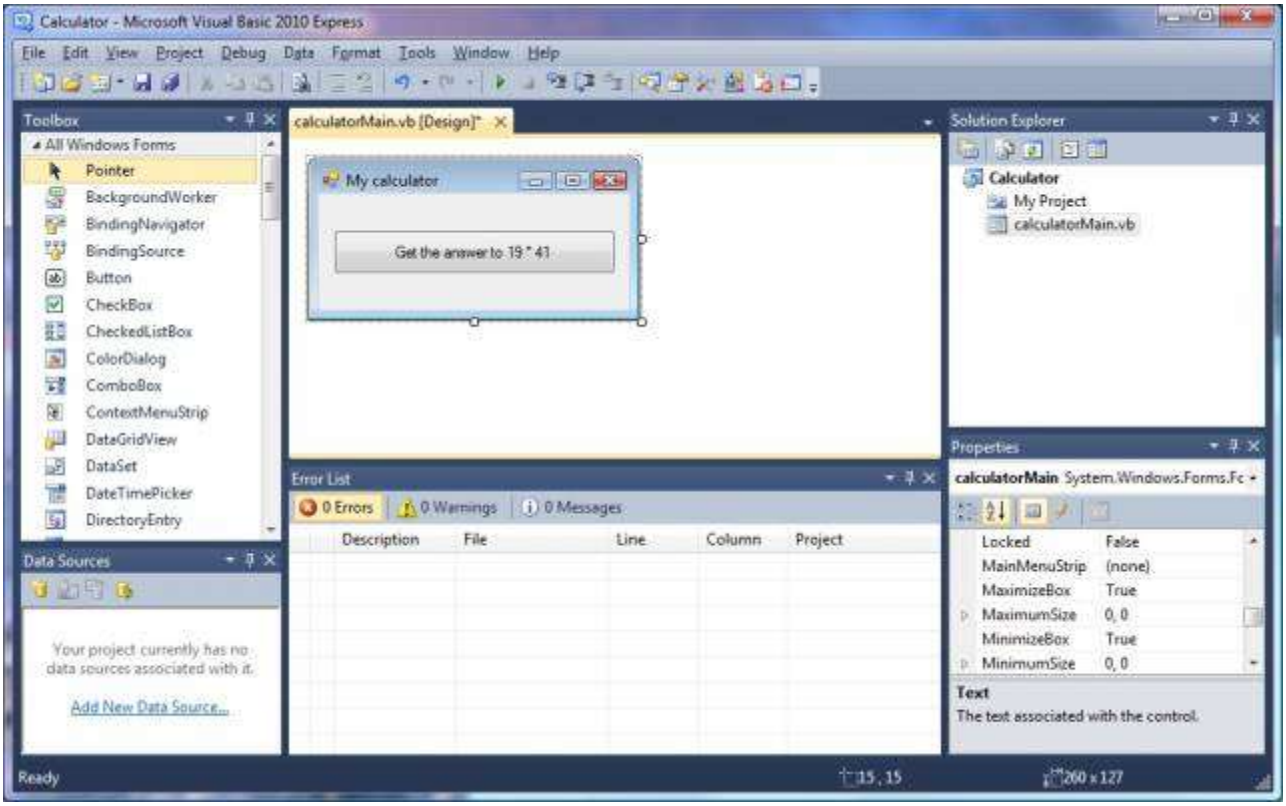
GotFocus
KeyDown
MouseClick
MouseDown
MouseMove
MouseUp

Designing a form

A form is a class which has a design around it. What this means is that the form will actually have an appearance on screen. The window that you are currently using to view this page is a form.

A form can have many different controls on it, such as those which were described earlier.

Below is an example of Visual Basic 2010 Express Edition in the middle of creating a form.



Visual Basic conceals a text editor behind this interface which is called the Designer. The Designer is responsible for creating the form itself. It hides code from the user to prevent accidental damage to the form design. Visual Basic itself is known as the **Integrated Development Environment (IDE)**, not to be confused with the hardware version; Integrated Drive Electronics.

A form, like a control, has many properties such as BackColor and Text. We also have properties such as MaximizeBox (as Boolean), MinimizeBox (as Boolean), Opacity (as Single) and TopMost (as Boolean) all with their own uses.

It might be a good idea to familiarise yourself with the IDE and form development before proceeding.

The first application - a simple calculator

The first application to be made in this tutorial is a simple calculator that will add two variables. Our variables will be number1 and number2. We will also have a variable with the name answer. The code for this application is shown below:

```
Dim number1 As Integer = 19
Dim number2 As Integer = 41
Dim answer As Integer = 0

answer = number1 * number2
```

This will give us an answer, but the user will be unable to see it. This is where we can now go further than just working on variables. The class that we are going to use is already inherited into every project that we work on. It is the `Microsoft` class. We are going to use the subroutine within this class called `MsgBox`. We could use `MessageBox` but `MsgBox` is fine for this sample.

```
Dim number1 As Integer = 19
Dim number2 As Integer = 41
Dim answer As Integer = 0

answer = number1 * number2

'this will show the answer in a message dialog
```

```
MsgBox(answer)
```

Looking at the code above, we can clearly see that the program multiplies 19 by 41 and then returns an answer as an integer and displays it through a message dialog.

Commenting code

In the previous sample application, above the last line, there is a green line of text. The line starts with an apostrophe. This represents what is known as a comment. A comment is not executed as code when the program compiles. The purpose of a comment is to inform the programmer what the line below or above does. It is only visible in the text editor.

Commenting is a very useful way of reminding yourself of what code does when you haven't programmed on the same project for a long time. Therefore, I recommend commenting the code.