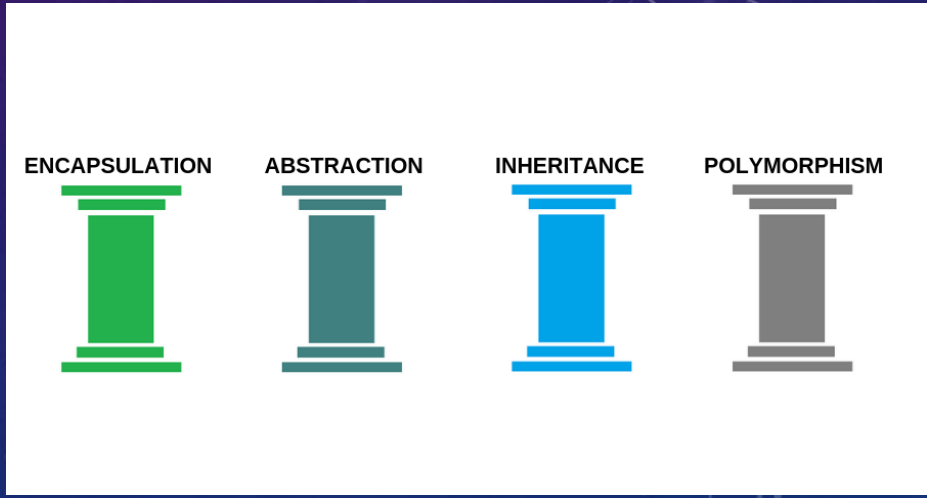
OBJECTS PROPERTIES AND METHODS

VISUAL BASIC

TABLE OF CONTENTS

LECTURE CONTEXT

- Most programs run in Object Orientated graphical environments these days such as Windows, Mountain Lion, Linux, etc.
- Graphical environments are based around a GUI model and all visible items are visual representations of some code that programmers and users can modify and utilise.
- These visible items are instances of Classes (sort of a genus name such as ‘birds’) and are called Objects.
- They may be manipulated via Properties and Methods by the user or programmer, and it is essential that students are familiar with these concepts before we move onto actual programming.



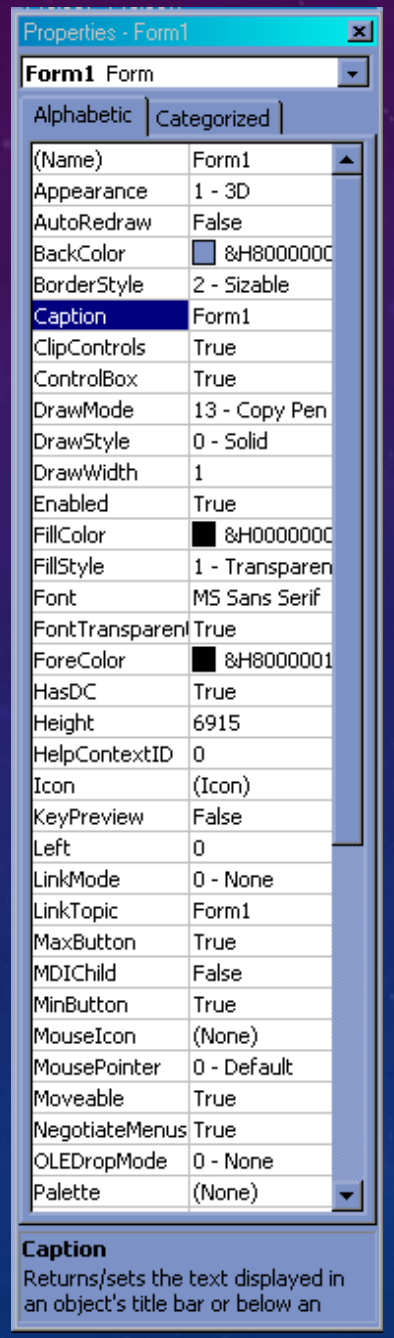
OBJECT PREFIXES

Names must start with a letter (max 255)
Capitalise and Prefix object names as follows (assessed):

Prefix	***Object***	***Example use***
cbo	Combobox	cboUserChoice
chk	Checkbox	*chkUserSelection*
cmd	Command Button	*cmdStart*
dat	Data	*datMyDataFile*
dir	Directory list	*dirUserDirectory*
dlg	Common dialog	*dlgPrintFile*
frm	Form	*frmStartupForm*
Gr	*Graphics Object*	*grForm*
txt	Textbox	*txtDataEntryBox*
Btn	Button	btnOk

PROPERTIES

- Set characteristics and/or appearance of an Object (occasionally functionality too)
- All Objects support them and expose many to programmer
- May be manipulated/set at both ***DESIGN TIME*** (when writing code) or ***RUN TIME*** (user interaction)
- Typical: Size, Colour, Font, Text, Name....
- *Format is* ***Object.Property***
- Some properties are read only at run time (e.g. Bold and Italic for Textbox)



GENERAL PROGRAMMING UNDERSTANDING

Accessing Properties

- picPicture1 is a Class / Object in this example
- When u use the dot notation, you are accessing the internal variable inside of it
- Writing it in this manner would mean assigning the value to the inner variable
- Also known as variables in other programming languages

e.g. picP*icture1.BackColor* = Color.Red

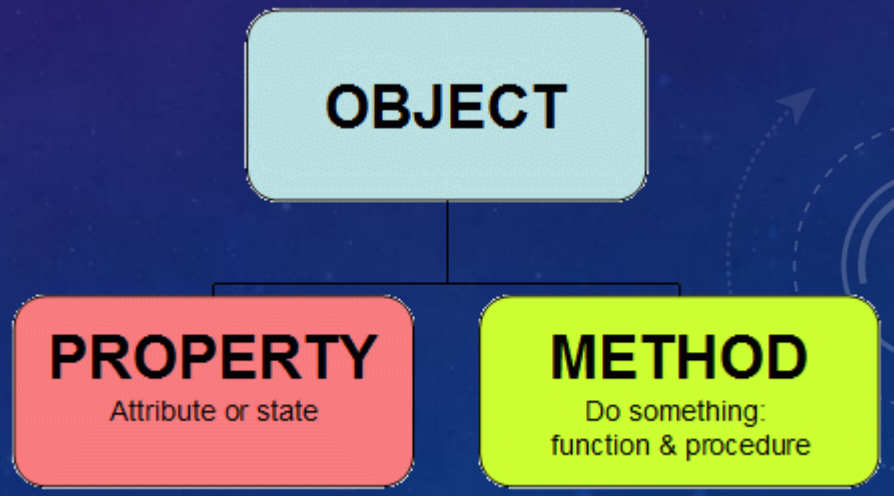
Accessing Methods

- picPicture1 is a Class / Object in this example
- When u use the brackets at the end u are trying to access a method
- Writing it in this manner would mean assigning the value to external variable
- Also known as functions in other programming languages

e.g. myResult = picPicture.getColor()

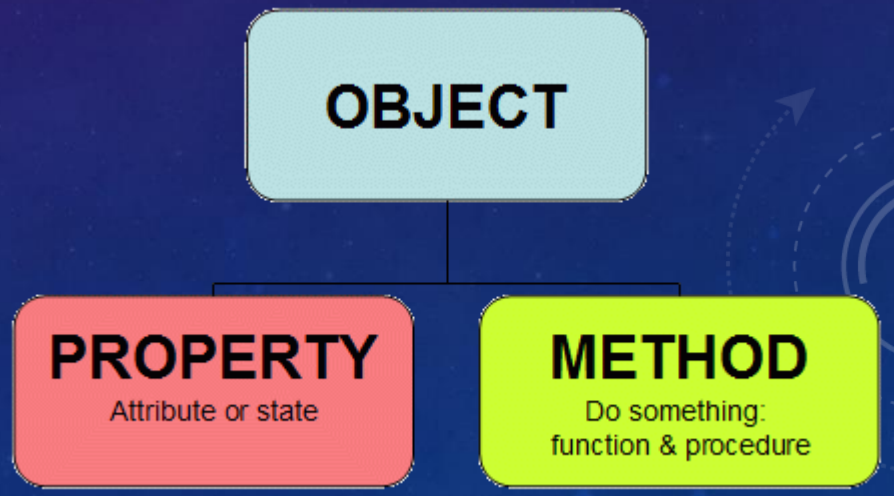
PROPERTIES

- **Name** used to access the control's properties and methods – Design Time ONLY
- **BackColor** background colour for text or graphics
  e.g. picP*icture1.BackColor* = Color.Red
- **ForeColor** usually pen or text colour
  e.g. picPicture1.ForeColor = Color.Black
- **Text** Sets a message in a textbox – or reads text from one
  *e.g. txt*TextBox.Text = "Hello World"



METHODS

- Procedures that operate on an Object
- May change the values of Properties
- Returns a value and accepts arguments
- General format is: Object.Method
- Arguments are separated by commas:
- E.g. draw a red circle with diameter 100




```
Dim grForm As Graphics = Me.CreateGraphics()
Dim ellipse_pen As New Pen(Color.Red, 3)
grForm.DrawEllipse(ellipse_pen, 0, 0, 200, 200)
```

METHOD EXAMPLE

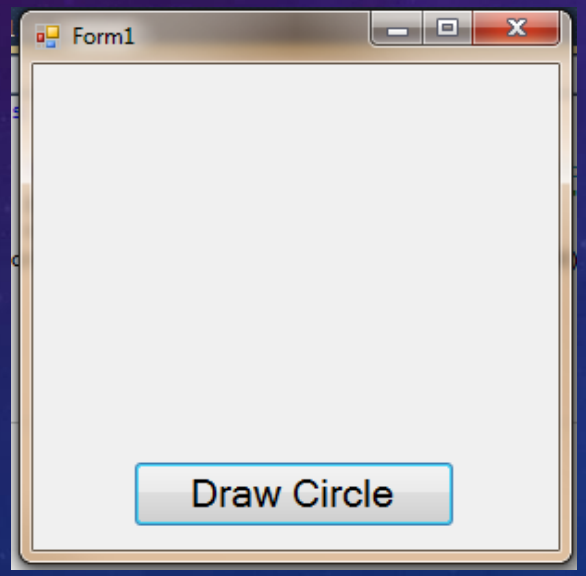
Object

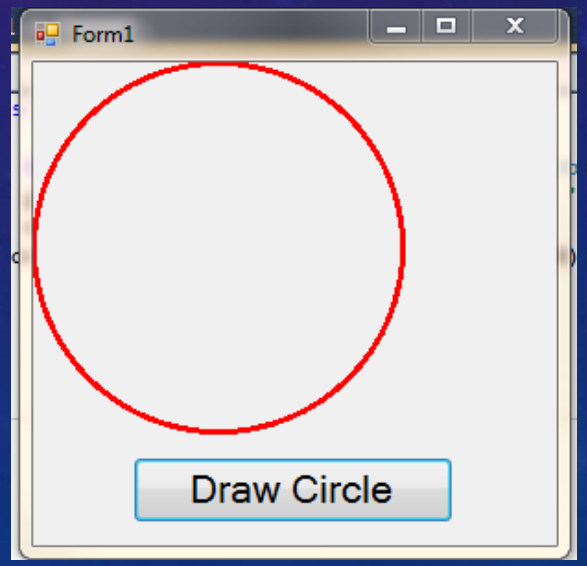
METHOD

Looks like a property doesn't it?

grForm.DrawEllipse(ellipse_pen, 0, 0, 200, 200)

Ellipse(circle) defined in a rectangle with top left coordinates 0,0 and diameter 200





EVENTS

- Trigger an object's reaction to external or internal stimuli
- Windows monitors for events and 'flags' occurrence to objects – not instantaneous
- Response code = EVENT HANDLER
- No Event Handler code= no response
- e.g clicking the command button runs:
  Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click

"Button1_Click" is just a name, "Handles Button1.Click" defines the event it handles

*Note underlined section beyond scope of the module

EVENTS

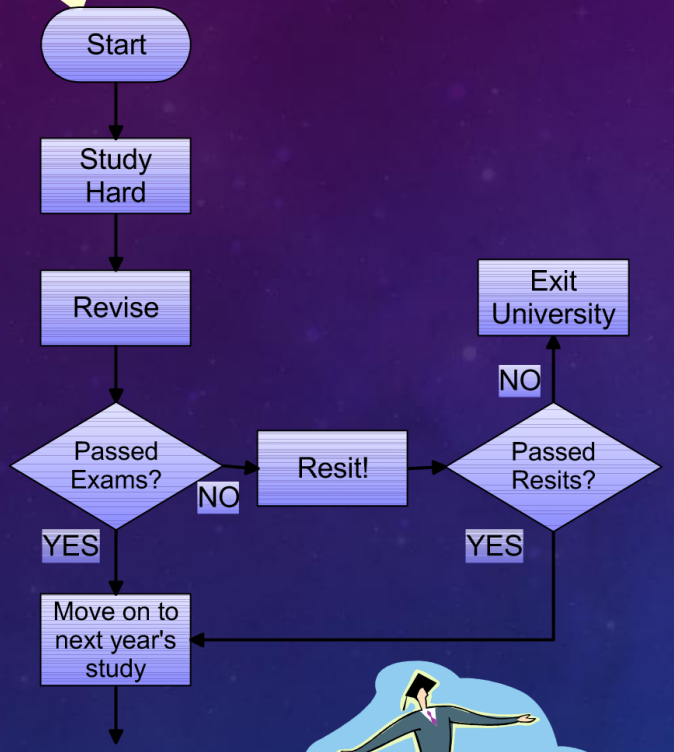
- An event handler may respond to several events which can be very useful
- Only the event Handler definitions need modification
- These visible items are instances of Classes (sort of a genus name such as ‘birds’) and are called Objects.
- E.g. to respond to 3 different button click event calls using a single event handler

Code Example

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)

Handles:

btnButton1.Click,

btnButton2.Click,

btnButton3.Click
```

Linear and Event Driven

- Linear algorithm
- Event driven algorithm



EVENTS AND HANDLERS

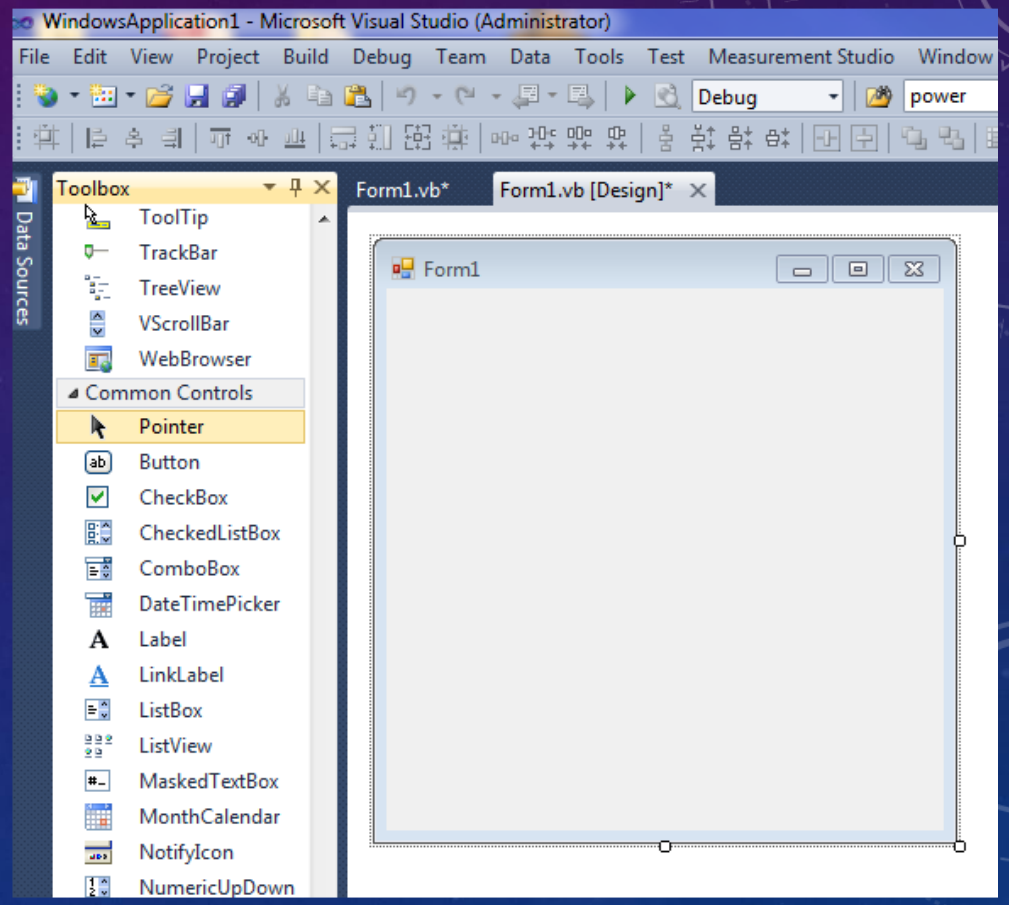
- Typical events include: Click, Double click, Mouse down/up, Key Down etc...
- Event Handler code resides in subroutines

```
Private Sub Button2_Click(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnSendMessage.Click
txtTextBox1.Text = "Hello"
```

FORM OBJECT

- Basis for most VB apps.
- One may have multiple forms in an application
- Supports many events
- Most important event handler is under the Load Event

  - run at application startup
  - initialization of variables and equipment
  - similar to onInit on other programming languages



TEXTBOX CONTROL

- Gather user input or display program output – most applications use Textboxes
- Typical properties: dimensions, position, bold, italic, index (control array), password box (******), colour, scrollbars...
- Supports MULTILINE property (default=False)

TEXTBOX CONTROL

To display text,
set property

```
Private Sub Button1_Click_1(…) Handles
Button1.Click
txtTextbox1.Text = “Hello World”
End Sub
```

To read and store text, equate a variable to the object’s property

```
Private Sub Button1_Click_1(…) Handles
Button1.Click
Dim strMyString as String
strMyString = txtTextbox1.Text
End Sub
```

COMMAND BUTTON CONTROL

- Default Event trigger by clicking mouse on button
- Typical properties: dimensions, position, bold, italic, index (control array), password box (******), colour, scrollbars...
- Supports MULTILINE property (default=False)
- Other events: key up, key down, mouse up, mouse down, got focus, lost focus
- Methods: Move, refresh, drag, set focus

```
Private Sub btnCommand1_Click()
txtTextbox1.Text = "Hello World"
End Sub
```

COMMAND BUTTON CONTROL

- Supports many properties: style, back color, captions, down picture, enabled, font, size, position…
- Changing color requires style = Graphical
- All can be set at design time or dynamically at runtime

WRITING COMMENTS IN VISUAL BASIC

Visual Basic Comments Example

Following is the example of defining the comments in Visual Basic programming language.

```
Module Module1
    Sub Main()
        ' Calling Method to Show Greet Messaging
        GreetMessage()
        Console.WriteLine("Press Any Key to Exit..")
        Console.ReadLine() ' This method to read the commands from a console
    End Sub
    ' This Method will display the welcome message
    Public Sub GreetMessage()
        Console.WriteLine("Welcome to Tutlane")
    End Sub
End Module
```

https://www.tutlane.com/tutorial/visual-basic/vb-comments

DEBUGGING IN VISUAL BASIC

- Use Console.WriteLine(variableName)
- Or System.Diagnostics.Debug.WriteLine(variableName)
- To debug ur output
- In programming languages like python we would use print() to check our output

```
Public Class Form1

    Dim strMyString As String
    0 references
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles btnSendMessage.Click
        strMyString = txtTextBox1.Text
        Console.WriteLine(strMyString)
        System.Diagnostics.Debug.WriteLine(strMyString)
    End Sub
End Class
```

STRING CONCATENATION IN VISUAL BASIC

Concatenation operators join multiple strings into a single string. There are two concatenation operators, `+` and `&`. Both carry out the basic concatenation operation, as the following example shows.

VB Copy

```vb
Dim x As String = "Mic" & "ro" & "soft"
Dim y As String = "Mic" + "ro" + "soft"
' The preceding statements set both x and y to "Microsoft".
```

These operators can also concatenate `String` variables, as the following example shows.

VB Copy

```vb
Dim a As String = "abc"
Dim d As String = "def"
Dim z As String = a & d
Dim w As String = a + d
' The preceding statements set both z and w to "abcdef".
```

https://docs.microsoft.com/en-us/dotnet/visual-basic/programming-guide/language-features/operators-and-expressions/concatenation-operators

STRING CONCATENATION IN VISUAL BASIC

Differences Between the Two Concatenation Operators

The + Operator has the primary purpose of adding two numbers. However, it can also concatenate numeric operands with string operands. The + operator has a complex set of rules that determine whether to add, concatenate, signal a compiler error, or throw a run-time InvalidCastException exception.

The & Operator is defined only for String operands, and it always widens its operands to String, regardless of the setting of Option Strict. The & operator is recommended for string concatenation because it is defined exclusively for strings and reduces your chances of generating an unintended conversion.

Can be written in this manner, example shown below

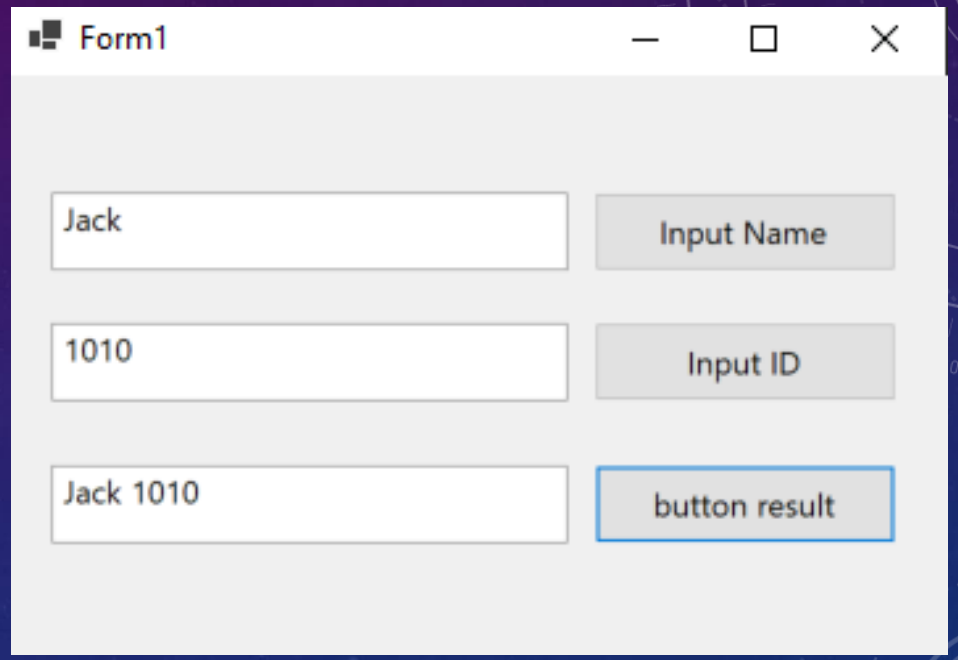
```
0 references
Private Sub Button3_Click(sender As Object, e As EventArgs) Handles button3.Click
    strMyString = txtTextBox1.Text + " " + txtTextBox2.Text
    System.Diagnostics.Debug.WriteLine(strMyString)
```

https://docs.microsoft.com/en-us/dotnet/visual-basic/programming-guide/language-features/operators-and-expressions/concatenation-operators

EXERCISE (10 – 15 MINUTES)

To create a mini application

1. Input name button on click must print in visual studio output
2. Input id button on click must print in visual studio output
3. When I click button result it should print out the name field with a space followed by the ID




```
Show output from: Debug
'exercise_1.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\sh
'exercise_1.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\sh
Jack
1010
Jack 1010
The thread 0x2b98 has exited with code 0 (0x0).
```