Programming Paradigms

1) Procedural Programming
   - Sequential flow of control
   - Traditional structured programming methods
   - Code and data are distinct from each other

2) Object Oriented Programming
   - Still sequential flow of control
   - Code and data are integrated together to form an ‘Object’.
   - It is safer, and it encourages code reuse.
   - Three main features of Object oriented programming:
     - Encapsulation  (Protect data from unauthorized use) (prevent inadvertent programming errors)
     - Inheritance    (inherit and extend the capabilities of preexisting code) (allow for reusable code)
     - Polymorphism  (Ability to overload and override methods) (makes code more readable and intuitive)

3) Event Driven Programming
   - Non Sequential flow of control.
   - User controls the flow of the program execution.
     (by causing an event)
   - An event is typically a mouse click, or a keyboard action.
VB.Net

- Is an Object Oriented and Event Driven programming language.

Procedural Applications

○ In traditional or procedural applications, the application controls which portions of code run, and the sequence in which it executes the instructions. Application execution starts with the first line of code, and follows a predefined path through the application, calling procedures as needed.

Event-Driven Applications

○ In an event-driven application, execution does not follow a predetermined path. Instead, it runs different code sections in response to events.

○ Events can be triggered by the user's actions, by messages from the system or other applications. The sequence of events determines the sequence in which the code runs. Therefore, the path through the application's code differs each time the program runs.

○ An essential part of event-driven programming is to write code that responds to the possible events that may occur in an application.

○ The following figure shows some actions that generate events to which you can respond by writing code. These events can occur in any order.
Choose a menu item

Click a button

Enter text

Address:
123 Main St.

Send
Cancel
The Object Model:

Objects in VB can be:
- Predefined (e.g. Forms or windows, Controls, etc.)
- User defined (created by the user as needed)
- All objects contain Properties and Methods.
- Some objects contain Events.

Property:
- Tell something about an object
  (name color, size, etc.)

Method:
- Actions associated with the object.
  (move, show, hide, etc.)

Event:
- Occur when the user takes an action.
  (e.g. click the mouse, press a key, etc.)

Class:
- The template for creating new objects.
- The definition of an object.
- Each object is an instance of a class.
Steps in Developing VB Projects

Planning:

1) Design the GUI (Draw the forms and their controls)
2) Plan the properties (Write down the properties that need to be set or changed.)
3) Plan the Code (Write the pseudocode to solve the problem at hand)
4) Desk check your pseudocode.

Coding:

5) Convert the design (#1 above) to Forms and Controls.

6) Set the properties of the Forms and Controls. (either manually at design time or programmatically at code time.)

7) Convert the pseudocode to Visual Basic code.

8) Test and Debug the code.
IDE or the Integrated Development Environment is a programming environment for writing and testing your VB programs.

Consists of:
- Editor (an intelligent editor!)
- Form Designer
- Compiler (To translate VB Code to an intermediate machine language known as the MS Intermediate Language (MSIL))
- Debugger (To help you with debugging your programs)
- Object/Class Browser (To view the classes, objects, properties, methods and events)
- Extensive Help facilities
1) Create a window (or FORM)

2) Using the Tool Box, select elements called ‘Controls’ and place them on the window.
   Ex: Button
       Text box
       Check Box
       Label
       Radio Button
       etc.

3) Set the properties of each control.

4) Program the ‘controls’ to react to events.
Problem: Given the radius of a circle, compute and display the **Area** and the **Circumference**.

Analysis:

# Determine the Input and Output of the program:
- Radius of a circle

# Determine the formulas, fact, etc.. needed:
- Area = PI * (Radius)^2
- Circumference = 2 * PI * Radius
- PI = 3.14159

Design:

# Design the user interface (windows and controls)
# Set the properties of the windows and controls.
# Determine the events and program them.
# Develop a list of steps to solve the problem (An Algorithm)

1) read the radius
2) compute the Area
   2.1) Area = PI * (Radius)^2
3) computer the circumference
   3.1) Circumference = 2 * PI * Radius
4) display the area and circumference
Implementation as a Console Application:

Module Module1

    Sub Main()

        Dim radius, area, circumference As Double
        Const PI = 3.14159

        Console.Write("Please enter the radius of the circle? ")
        radius = Console.ReadLine()
        area = PI * radius * radius
        circumference = 2 * PI * radius
        Console.WriteLine("Area = " & area)
        Console.WriteLine("Circumference = " & circumference)

        Console.ReadLine() 'just to pause the program.

    End Sub

End Module
Implementation as a Windows Application:

1) Design the User Interface

2) Set the properties for each of the controls:
   i.e. - Give the name ‘Calculate’ and ‘Exit’ to the buttons
   - Change the font and size of the ‘labels’
   - Change the ‘editable’ property of the text boxes.
     (Only for Area and Circumference text boxes.)
   - etc.
3) Determine the ‘control’ and the ‘events’ which should cause the program to calculate the ‘Area’ and ‘Circumference’ of the circle.

   (i.e. the ‘click’ event for the ‘Calculate’ control should be programmed.)

4) Code the event associate with the control.

   - When the user takes an action such as ‘click’ the ‘Calculate’ button, an event is created.
   - Programmer must write code to handle the event or the event is lost!!
   - The code written to handle the event is called an **event procedure**.
   - The syntax for the sub-procedure is shown below:

   ```vbnet
   Private Sub Procedure_Name(...) zero or more parameters...
   ......
   End sub
   ```
Example:

Public Class Form1

    Inherits System.Windows.Forms.Form

    *** SOME CODE GENERATED BY THE WINDOWS FORM DESIGNER ***

    Private Sub BtnCalculate_Click(ByVal sender As System.Object,
    ByVal e As System.EventArgs) Handles BtnCalculate.Click

        Dim Radius, Area, Circumference As Double
        Const PI = 3.14159
        Radius = CDbl(TxtRadius.Text)  'Convert the string to double
        Area = PI * Radius * Radius
        Circumference = 2 * PI * Radius
        TxtArea.Text = CStr(Area)      'Convert to string
        TxtCircumference.Text = CStr(Circumference)
    End Sub

    Private Sub BtnExit_Click(ByVal sender As System.Object,
    ByVal e As System.EventArgs) Handles BtnExit.Click

        Me.Close()
    End Sub

End Class
Now for the Code Generated by the FORM DESIGNER

Public Class Form1
    Inherits System.Windows.Forms.Form

#Region " Windows Form Designer generated code "
Public Sub New()
    MyBase.New()
    'This call is required by the Windows Form Designer.
    InitializeComponent()
    'Add any initialization after the InitializeComponent() call
End Sub

'Form overrides dispose to clean up the component list.
Protected Overloads Overrides Sub Dispose(ByVal disposing As Boolean)
    If disposing Then
        If Not (components Is Nothing) Then
            components.Dispose()
        End If
    End If
    MyBase.Dispose(disposing)
End Sub

'Required by the Windows Form Designer
Private components As System.ComponentModel.IContainer

'NOTE: The following procedure is required by the Windows Form Designer
'It can be modified using the Windows Form Designer.
'Do not modify it using the code editor.
Friend WithEvents Label1 As System.Windows.Forms.Label
Friend WithEvents Label2 As System.Windows.Forms.Label
Friend WithEvents BtnCalculate As System.Windows.Forms.Button
Friend WithEvents BtnExit As System.Windows.Forms.Button
Friend WithEvents Label3 As System.Windows.Forms.Label
Friend WithEvents Label4 As System.Windows.Forms.Label
Friend WithEvents TxtArea As System.Windows.Forms.TextBox
Friend WithEvents TxtCircumference As System.Windows.Forms.TextBox
Friend WithEvents TxtRadius As System.Windows.Forms.TextBox
<System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()
    Me.Label1 = New System.Windows.Forms.Label
End Sub
Me.TxtRadius = New System.Windows.Forms.TextBox
Me.Label3 = New System.Windows.Forms.Label
Me.TxtArea = New System.Windows.Forms.TextBox
Me.TxtCircumference = New System.Windows.Forms.TextBox
Me.SuspendLayout()
'
'Label1
'
Me.Label1.Location = New System.Drawing.Point(16, 24)
Me.Label1.Name = "Label1"
Me.Label1.TabIndex = 0
Me.Label1.Text = "Compute the Area and Circumference of a Circle"
'
'Label2
'
Me.Label2.Location = New System.Drawing.Point(24, 88)
Me.Label2.Name = "Label2"
Me.Label2.Size = New System.Drawing.Size(80, 24)
Me.Label2.TabIndex = 1
Me.Label2.Text = "Radius:"'
'TxtRadius
'
Me.TxtRadius.Location = New System.Drawing.Point(120, 88)
Me.TxtRadius.Name = "TxtRadius"
Me.TxtRadius.Size = New System.Drawing.Size(96, 20)
Me.TxtRadius.TabIndex = 2
Me.TxtRadius.Text = "Enter the radius"
'
'BtnCalculate
Me.BtnCalculate.Location = New System.Drawing.Point(88, 248)
Me.BtnCalculate.Name = "BtnCalculate"
Me.BtnCalculate.Size = New System.Drawing.Size(96, 40)
Me.BtnCalculate.TabIndex = 3
Me.BtnCalculate.Text = "Calculate"

Me.BtnExit.Location = New System.Drawing.Point(216, 248)
Me.BtnExit.Name = "BtnExit"
Me.BtnExit.TabIndex = 4
Me.BtnExit.Text = "Exit"

Me.Label3.Location = New System.Drawing.Point(192, 144)
Me.Label3.Name = "Label3"
Me.Label3.Size = New System.Drawing.Size(64, 32)
Me.Label3.TabIndex = 5
Me.Label3.Text = "Area"

Me.Label4.Location = New System.Drawing.Point(120, 184)
Me.Label4.Name = "Label4"
Me.Label4.TabIndex = 6
Me.Label4.Text = "Circumference"
'TxtArea

Me.TxtArea.Location = New System.Drawing.Point(264, 144)
Me.TxtArea.Name = "TxtArea"
Me.TxtArea.ReadOnly = True
Me.TxtArea.TabIndex = 7
Me.TxtArea.Text = ""

'TxtCircumference

Me.TxtCircumference.Location = New System.Drawing.Point(264, 184)
Me.TxtCircumference.Name = "TxtCircumference"
Me.TxtCircumference.ReadOnly = True
Me.TxtCircumference.TabIndex = 8
Me.TxtCircumference.Text = ""

'Form1

Me.AutoScaleBaseSize = New System.Drawing.Size(5, 13)
Me.ClientSize = New System.Drawing.Size(424, 301)
Me.Controls.Add(Me.TxtCircumference)
Me.Controls.Add(Me.TxtArea)
Me.Controls.Add(Me.Label4)
Me.Controls.Add(Me.Label3)
Me.Controls.Add(Me.BtnExit)
Me.Controls.Add(Me.BtnCalculate)
Me.Controls.Add(Me.TxtRadius)
Me.Controls.Add(Me.Label2)
Me.Controls.Add(Me.Label1)
Me.Name = "Form1"
Me.Text = "Form1"
Me.ResumeLayout(False)
End Sub

#End Region
Private Sub BtnCalculate_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
    Handles BtnCalculate.Click
    Dim Radius, Area, Circumference As Double
    Const PI = 3.14159

    Radius = CDbl(TxtRadius.Text) 'Convert the string to double

    Area = PI * Radius * Radius
    Circumference = 2 * PI * Radius

    TxtArea.Text = CStr(Area) 'Convert to string
    TxtCircumference.Text = CStr(Circumference)
End Sub

Private Sub BtnExit_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
    Handles BtnExit.Click
    Me.Close()
End Sub
End Class
Various Types of Errors

- Syntax Errors
- Runtime Errors
- Logic Errors

Syntax Errors:
- When VB’s rules for punctuation, format, or spelling is violated.
- Most Syntax errors are detected by the editor in the IDE.

Runtime Errors
- If your program halts or crashes during execution.
- Example:
  - Divide by zero.
  - Finding the square root of a negative number.
  - Trying to read from a non-existing file.
- Runtime errors are known as ‘EXCEPTIONS’

Logic Errors
- Program runs but produces incorrect results.