LEARN VISUAL BASIC.NET

NOR AZLINA
NORHAYATI SA'ADAH

LEARN VISUAL BASIC .NET

NOR AZLINA BINTI IBRAHIM
NORHAYATI SA'ADAH BINTI CHE ABD RAZAK

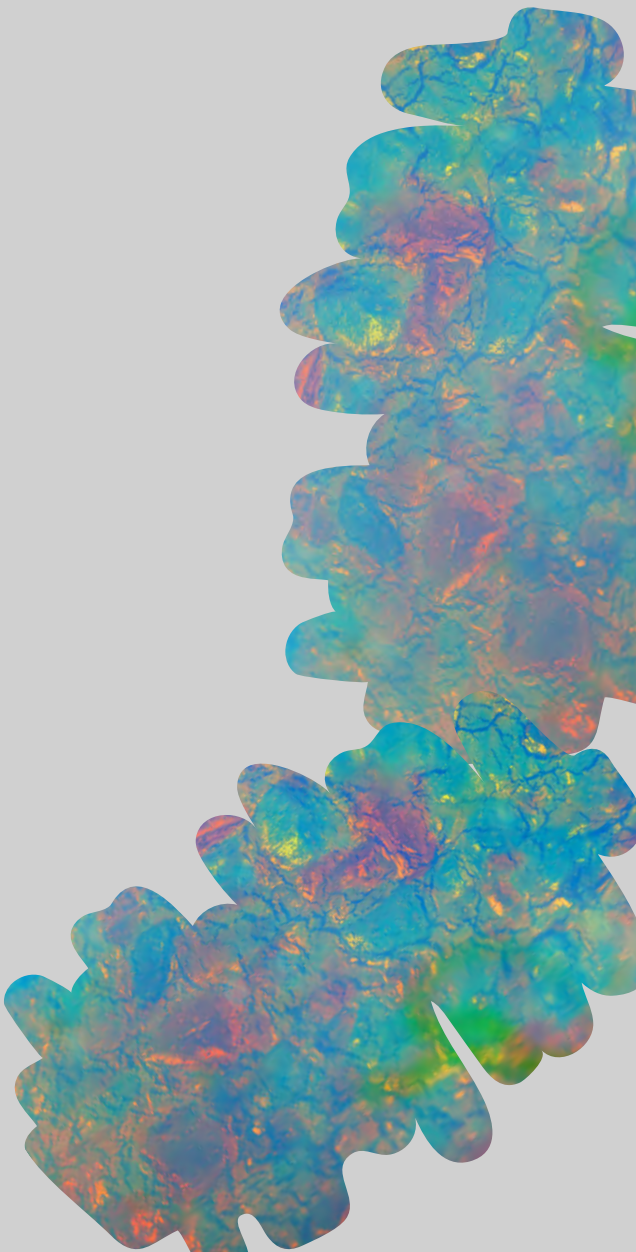
Jabatan Teknologi Maklumat dan Komunikasi
Politeknik Sultan Mizan Zainal Abidin
KM 8, Jalan Paka,
23000 Dungun
09-8400800

LEARN VISUAL BASIC .NET

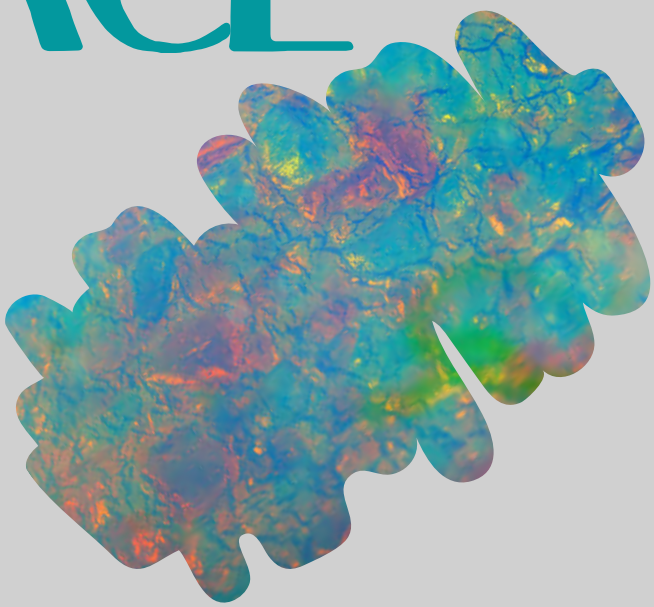
EDITION 2021
PUBLISHED BY:
POLITEKNIK SULTAN MIZAN ZAINAL ABIDIN.
KM 08, JALAN PAKA,
23000 DUNGUN, TERENGGANU DARUL IMAN.
TEL: 09-8400800
FAX: 09-8458781
WWW.PSMZA.EDU.MY

Thank you

TO OUR
BELOVED
FAMILY
AND
FRIENDS



PREFACE



LEARN
Visual Basic .net

provides students with the knowledge and skills needed to develop applications in Microsoft Visual Basic .NET for the Microsoft .NET platform. The course focuses on user interfaces programming structure, language syntax, and integration of VB.NET application development. This course introduces computer programming using the VB Programming language with object-oriented programming principles. Emphasis is on event-driven programming methods, including creating and manipulating objects, classes, and using object oriented tools such as the class debugger.

TABLE OF CONTENTS

CHAPTER 1

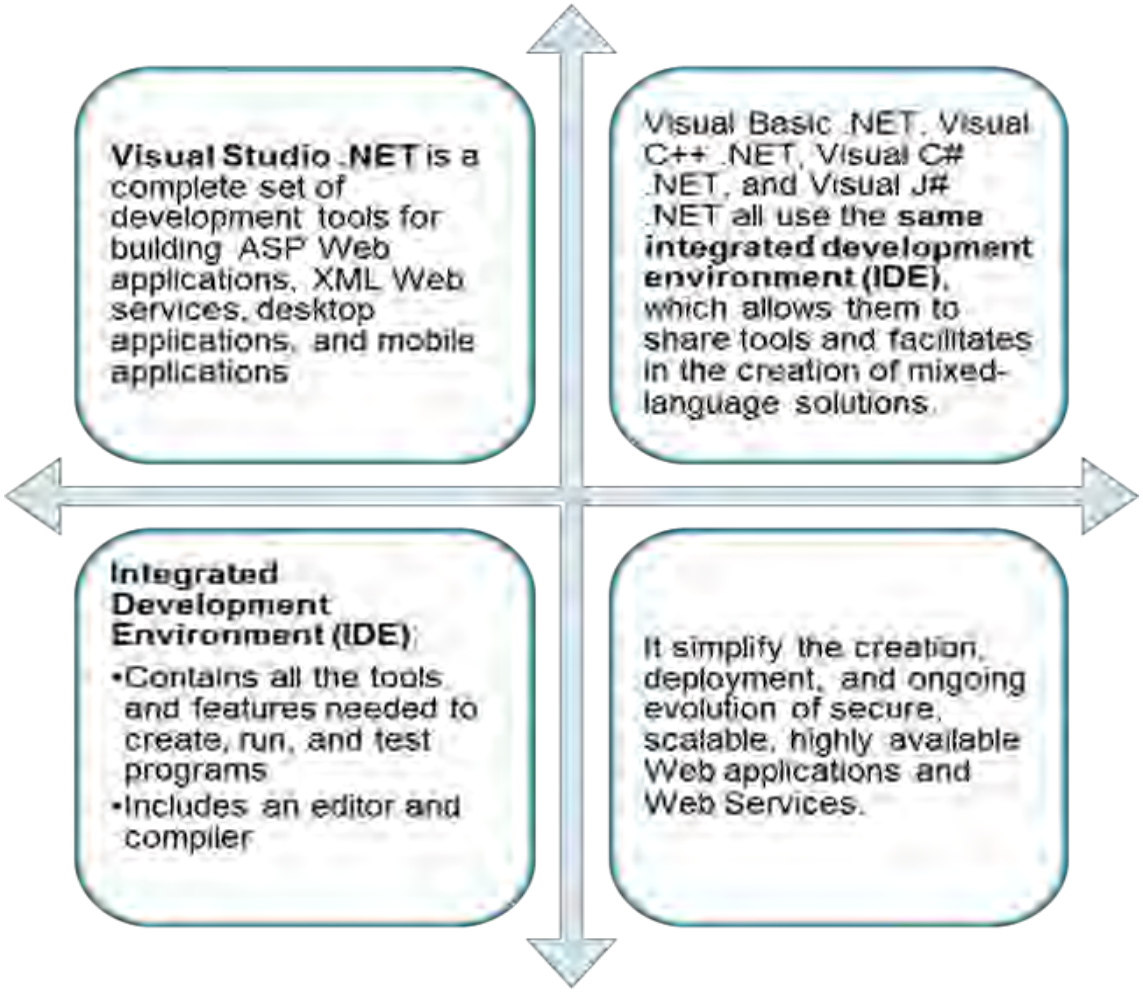
LAUNCHING VISUAL BASIC

At the completion of this chapter , you will be able to:

- Explain the .Net technology that can support multiple programming languages.
- Understand Visual Basic .NET environment.
- Navigate the Integrated Development Environment (IDE)

1

1.1 Visual Studio .NET



1.2 How To Start Visual Basic.Net

1. Click the Windows Start button
2. Locate Microsoft Visual Studio as shown as Figure 1.1.

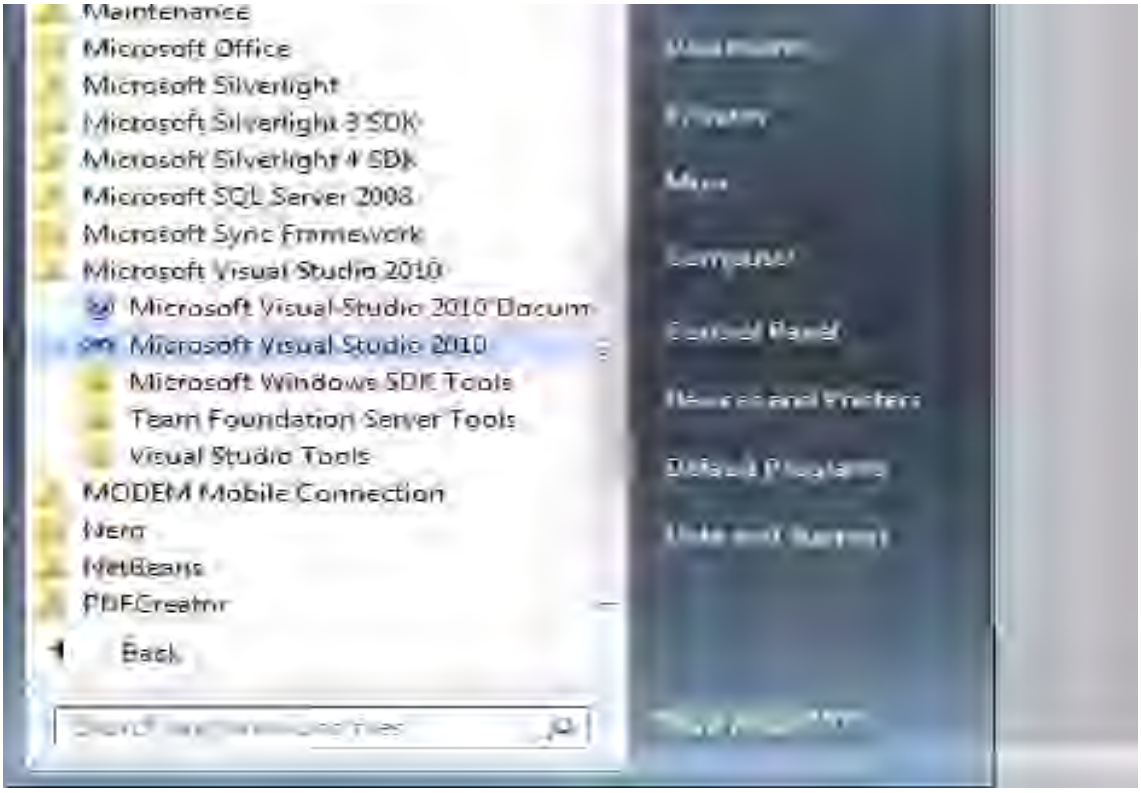


Figure 1.1 Windows Start Button

3. The first screen to display may require you to choose your Default Environment Settings. If this screen displays, click the Visual Basic Development Settings option in the list box and then click the Start Visual Studio button as shown as Figure 1.2.

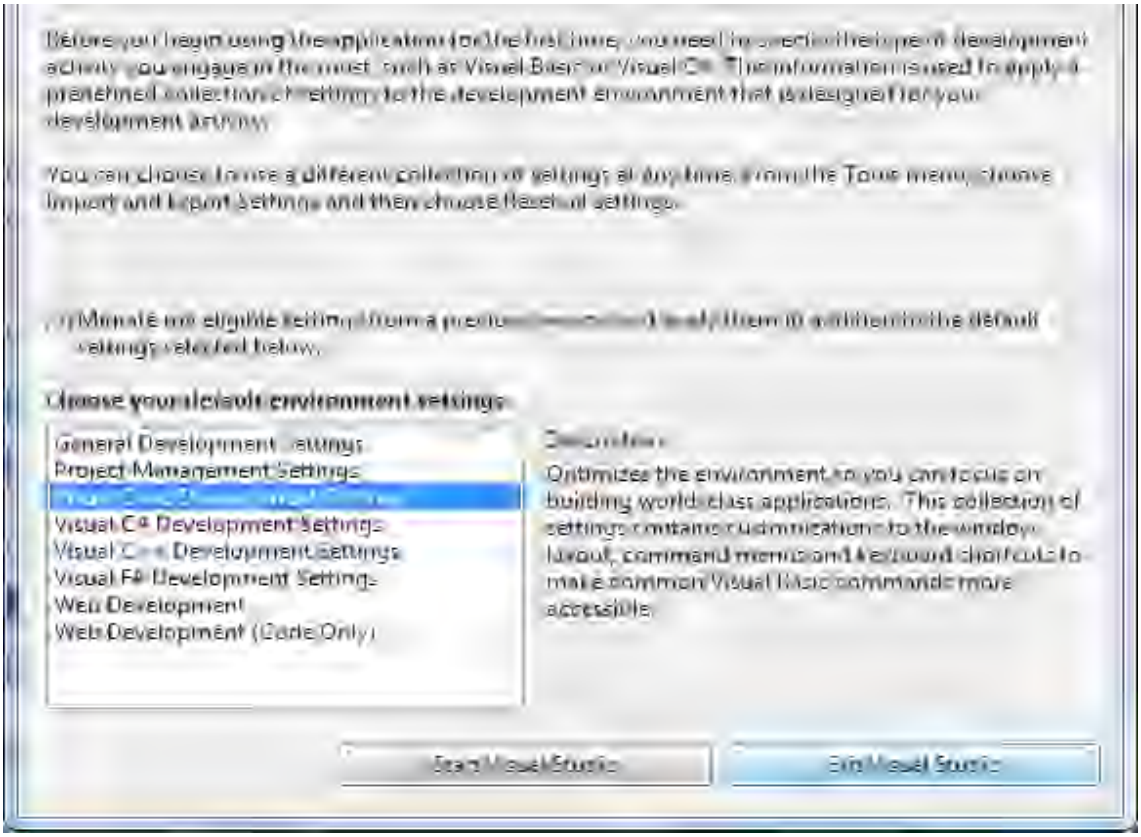


Figure 1.2 Default Environment Settings

4. The Visual Studio Start Page is shown in the

   Figure 1.3. The start page is slightly different for Visual Studio version.

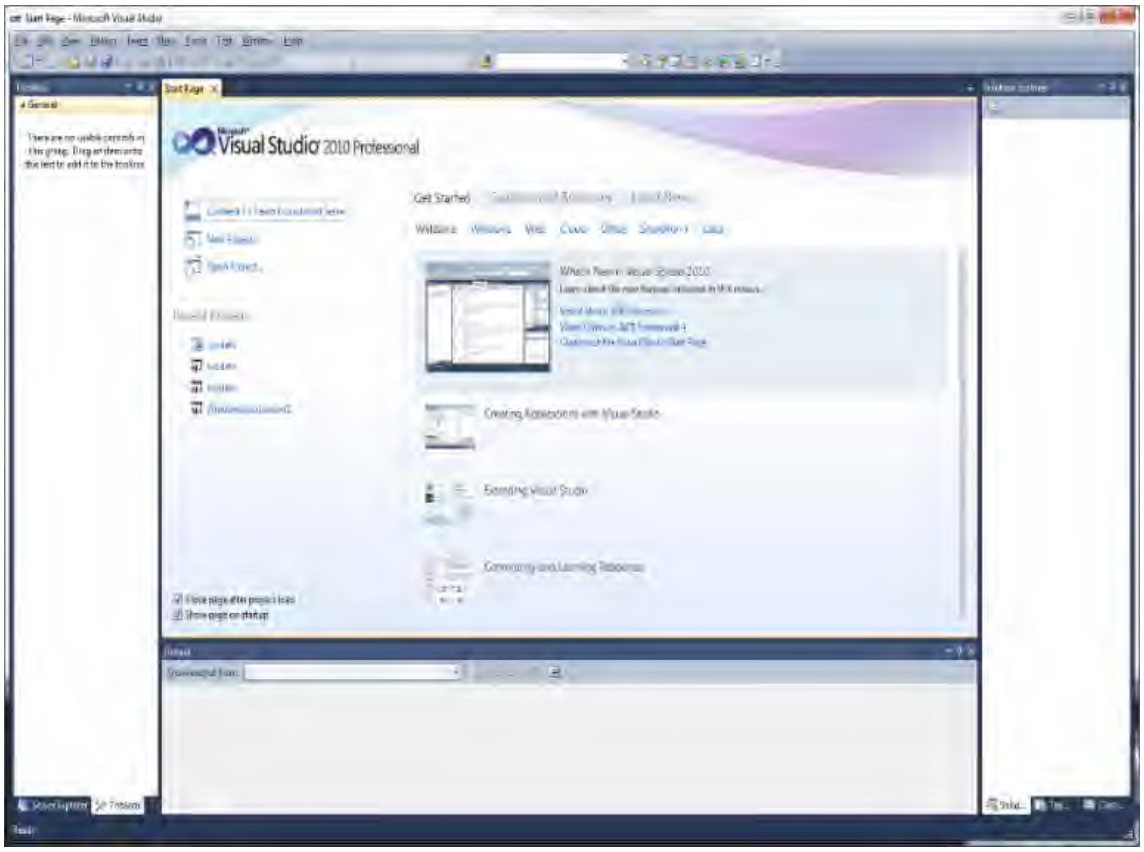


Figure 1.3 Visual Studio Start Page

5. Click the New Project link shown in the Figure 1.3. This opens the New Project dialog box.

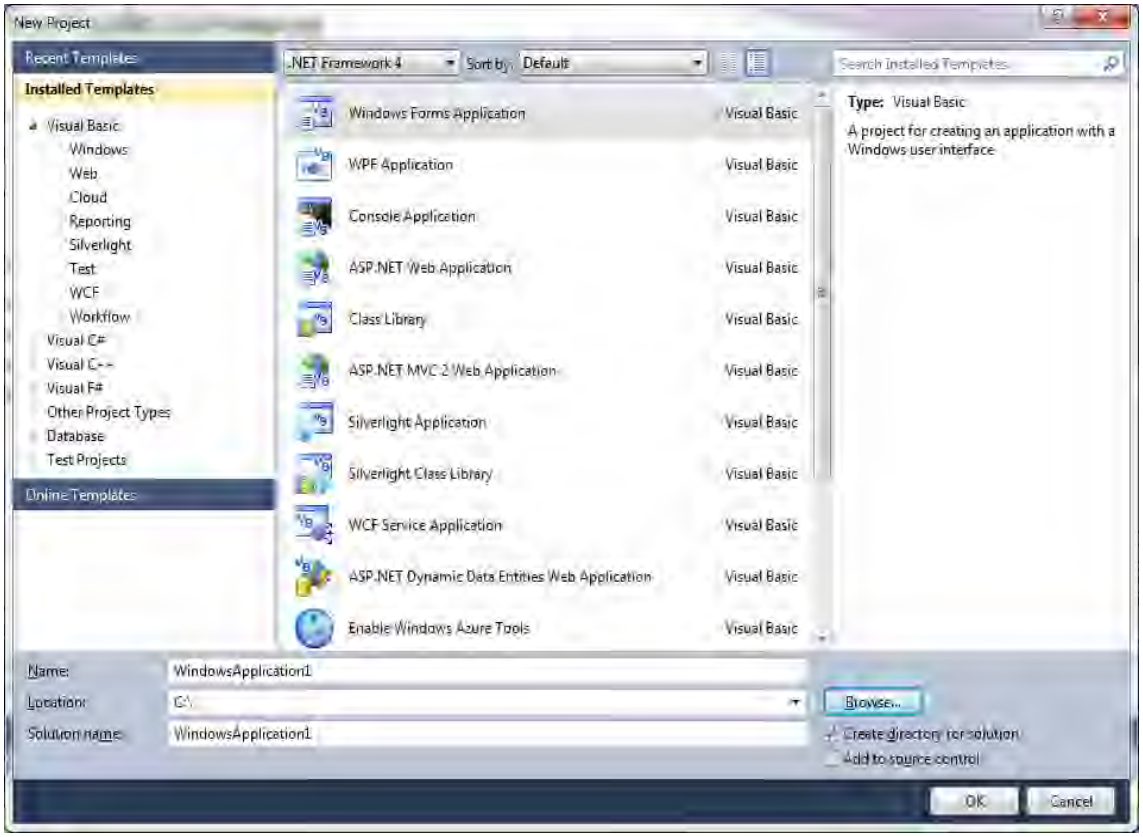


Figure 1.4 New Project Dialog Box

6. Your first project will be a Visual Basic Project using a Windows Forms Application template as shown in the Figure 1.4.

   The default name is WindowsApplication1. Change the project name to: Myinfo.

   Click the OK button.

CHAPTER 2

GRAPHICAL USER INTERFACE

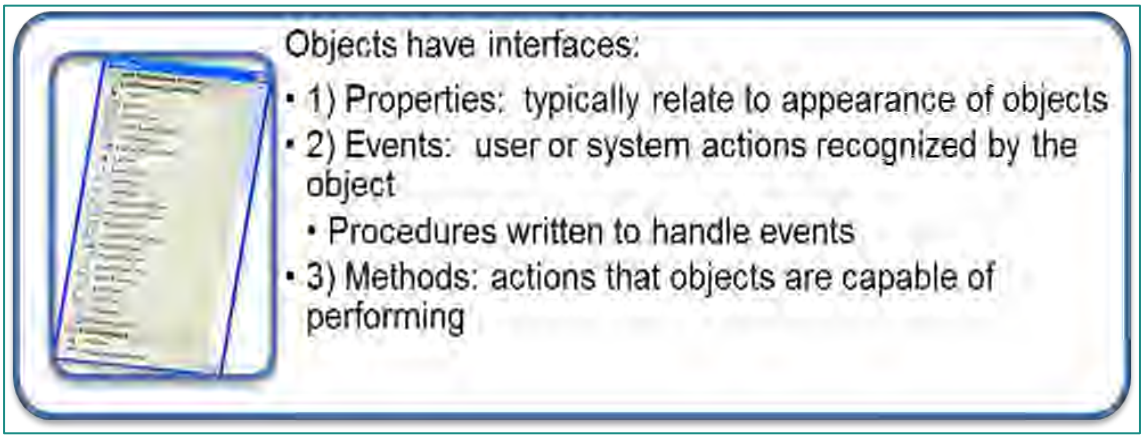
At the completion of this chapter , you will be able to:

- Use the ToolBox control.
- Write program code using code window.
- Apply Menus and Submenus, Dialog Boxes, Message Box and Input Box.

2.1 Controls

Controls are added to the Form from the Toolbox.

Each control has a set of properties, and a set of event procedures associated with it.



2.2 Biodata Application

Step 1: Design the interface as shown in Figure 2.1

Step 2: Set the objects and properties as shown in Table 2.1.

Step 3: Plan the event procedures as shown in Table 2.2.

Step 4: Write the code

Step 5: Save and run the project.

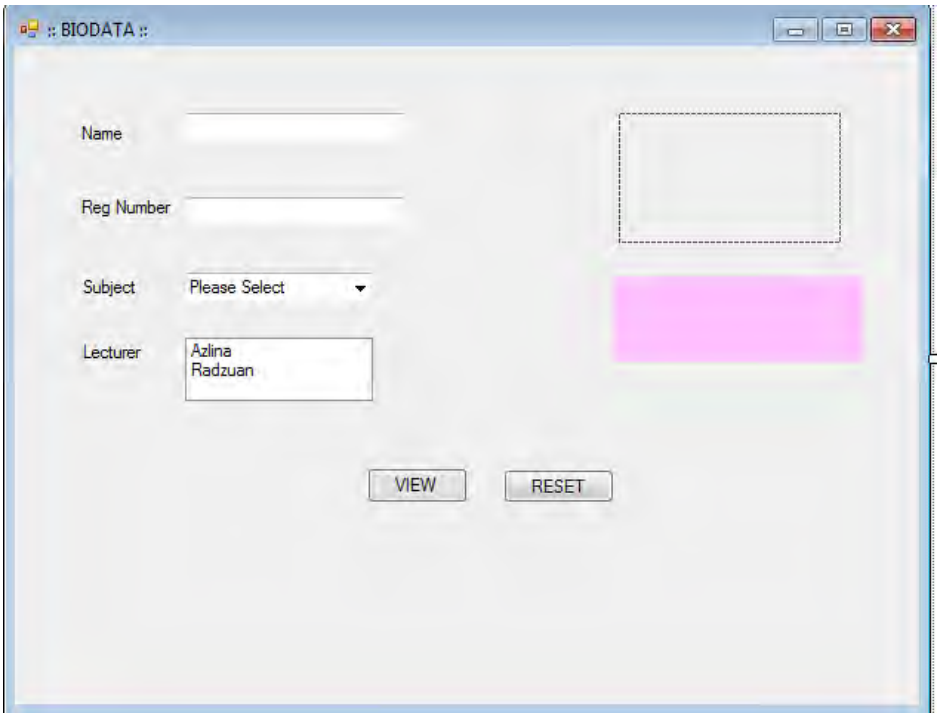


Figure 2.1 Graphical User Interface

Table 2.1 Plan the Objects and Properties

Object	Property
Label1	Name:lblname Text: Name
Label2	Name: lblreg Text: Reg Number
Label3	Name: lblsubject Text: Subject
Label4	Name: lbllecturer

	Text: Lecturer
Label5	Name: lblview Text: (blank)
TextBox1	Name:txtname Text: (blank)
TextBox2	Name:txtreg Text: (blank)
ComboBox	Name:cbosubject Items: VB.Net Project
ListBox	Name:lstlect Items: Azlina Radzuan
PictureBox	Name: picsaya
Button 1	Name: btnview Text: VIEW
Button 2	Name: btnreset Text: RESET

Table 2.2 Plan the Event Procedures

Procedure	Action
btnview	Set lblview to the name, reg number, subject and lecturer from the text boxes, combo box and list box (concatenate them)
btnreset	Clear the text boxes, label and picture box

```
Private Sub btnview_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnview.Click

      lblview.Text = "Hi " + txtname.Text + ". Your reg num is " +
      txtreg.Text + ". You choose subject " + cbosubject.Text + " and
      your lecturer is " + lstlect.Text

picsaya.Image = System.Drawing.Image.FromFile("D:\mycode\gambar.gif")

End Sub

Private Sub btnreset_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnreset.Click

        lblview.Text = String.Empty

        txtname.Text = ""

        txtreg.Text = ""

        picsaya.Visible = False

End Sub
```

2.2.1 Save the Project

Open the Visual Studio File menu and choose Save All. This option saves the current form, project, and solution files. You already selected the path for the files when you first created the project

2.2.2 Run the Project

Open the Visual Studio Debug menu and choose Start Debugging or you can choose shortcut key

2.2.3 Open the Project

You can choose one of three ways to open a saved project:

- If your project appears on the Start Page, you can open it by clicking on its name.
- Click on the Open Project button on the Start Page and browse to find your.sln file.
- Select Open Project / Solution from the Visual Studio File menu and browse to find your .sln file

2.3 Check Box And Radio Button Application

Figure 2.2 Check Box And Radio Button Interface

```
Private Sub radred_CheckedChanged(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles radred.CheckedChanged

        Me.BackColor = Color.Red

End Sub


Private Sub radyellow_CheckedChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles radyellow.CheckedChanged

        Me.BackColor = Color.Yellow

End Sub


Private Sub radblue_CheckedChanged(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles radblue.CheckedChanged

        Me.BackColor = Color.Blue

End Sub


Private Sub chkpoly_CheckedChanged(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles chkpoly.CheckedChanged

        If chkpoly.Checked = True Then

            lblpoly.Text = "PSMZA"

        Else

            lblpoly.Text = ""

        End If

    End Sub
```

```
Private Sub chkprogram_CheckedChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles chkprogram.CheckedChanged

        If chkprogram.Checked = True Then

            lblprogram.Text = "Programmer:@zlin@"

        Else

            lblprogram.Text = ""

        End If

    End Sub


Private Sub chklogo_CheckedChanged(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles chklogo.CheckedChanged

        If chklogo.Checked = True Then

        picpoly.Image = System.Drawing.Image.FromFile("D:\MODUL\VISUAL
BASIC.NET\MY

      CODES\logojpp.jpg")

            picpoly.Visible = True

        Else

            picpoly.Visible = False


        End If

End Sub
```

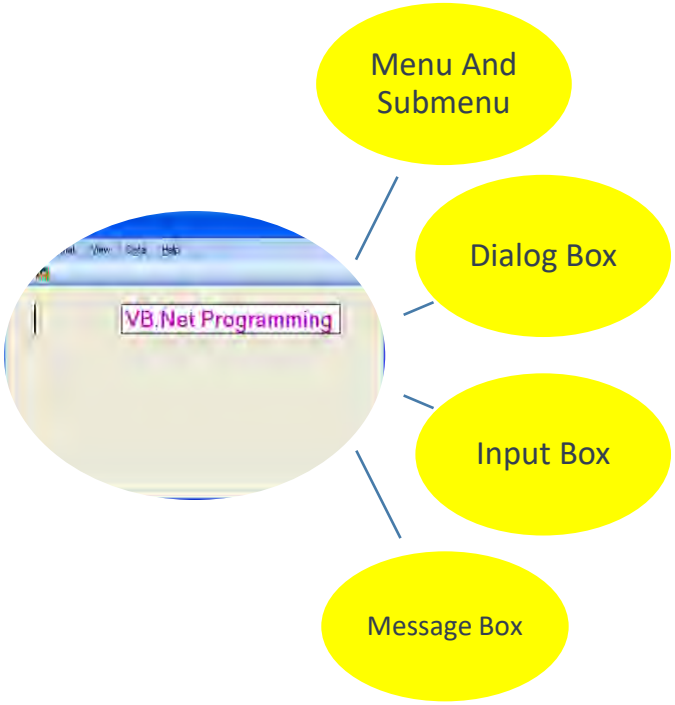
2.4 Mini Calculator Application

Figure 2.3 Mini Calculator Interface

```
Public Class frmcalculator


Private Sub btnadd_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnadd.Click

        txtresult.Text = Val(txtnum1.Text) + Val(txtnum2.Text)

 End Sub


Private Sub btnminus_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnminus.Click

        txtresult.Text = Val(txtnum1.Text) - Val(txtnum2.Text)

    End Sub


Private Sub btnmultiply_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnmultiply.Click

        txtresult.Text = Val(txtnum1.Text) * Val(txtnum2.Text)

    End Sub
```

```
Private Sub btndivide_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btndivide.Click
                txtresult.Text = Val(txtnum1.Text) / Val(txtnum2.Text)
    End Sub


Private Sub Button5_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmdreset.Click
        txtnum1.Text = ""
        txtnum2.Text = ""
        txtresult.Text = ""
    End Sub


Private Sub Button6_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmdexit.Click
        End
    End Sub
```

2.5 Menus and Submenus, Dialog Boxes, Message Box and Input Box



MENUS

* Consist of a menu bar that contains menus, each of which drops down to display a list of menu items.

* Can use menu items in place of or in addition to buttons to execute a procedure

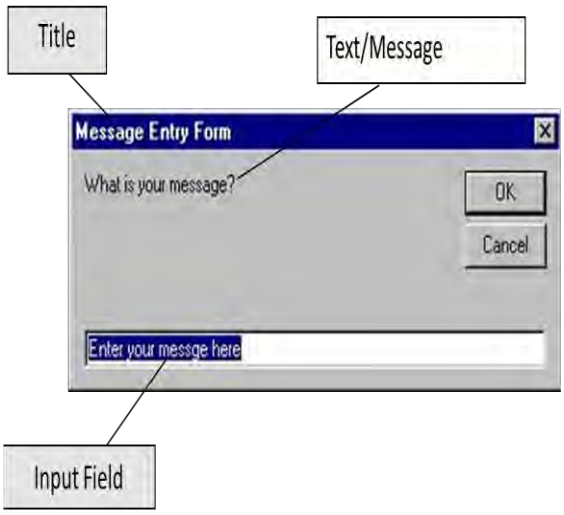
DIALOG BOX

* A control to allow the user to use the set of the standard dialog box in their project

Common dialog components provided with Visual Studio:

- Open
- Save
- Color
- Font
- Print

* DialogObject.ShowDialog( )

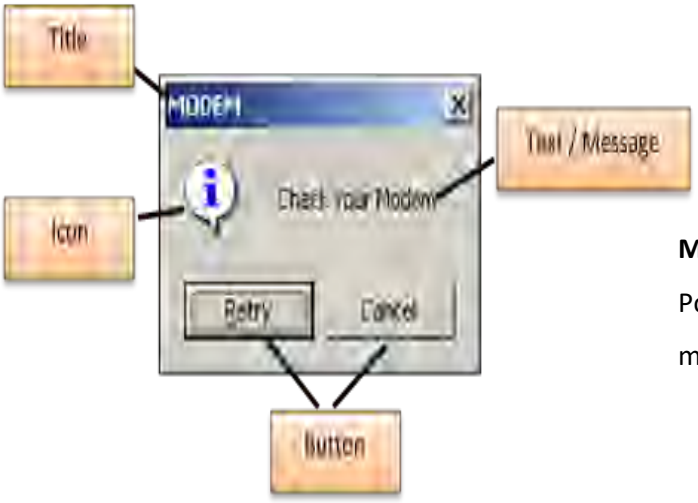


INPUT BOX

* Input box provide a simple way to gather input without placing a text box on a form.

* Quick and simple way to ask the user to enter data

* InputBox(“ Text”, “Title”, “Default”)



MESSAGE BOX

Pop up window that displays a message to the user

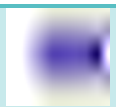
2.5.1 Message Box Format

```
MessageBox.Show(Message)
MessageBox.Show(Message, Title)
MessageBox.Show(Message, Title, Buttons)
MessageBox.Show(Message, Title, Buttons, Icon)
```

2.5.2 Message Box Button Values

Value	Description
MessageBoxButtons.AbortRetryIgnore	Displays Abort, Retry , Ignore buttons
MessageBoxButtons.OK	Displays OK buttons
MessageBoxButtons.OKCancel	Displays OK , Cancel buttons
MessageBoxButtons.RetryCancel	Displays Retry, Cancel buttons
MessageBoxButtons.YesNo	Displays Yes, No buttons
MessageBoxButtons.YesNoCancel	Displays Yes, No, Cancel buttons

2.5.3 Message Box Icons

Icons	Syntax
	MessageBoxIcon.Asterisk MessageBoxIcon.Information
	MessageBoxIcon.Error MessageBoxIcon.Hand MessageBoxIcon.Stop
	MessageBoxIcon.Exclamation MessageBoxIcon.Warning
	MessageBoxIcon.Question

2.6 Apply Menus and Submenus, Dialog Boxes, Message Box and Input Box

Start a new project. Add a MenuStrip control to your form as shown in Figure 2.2.

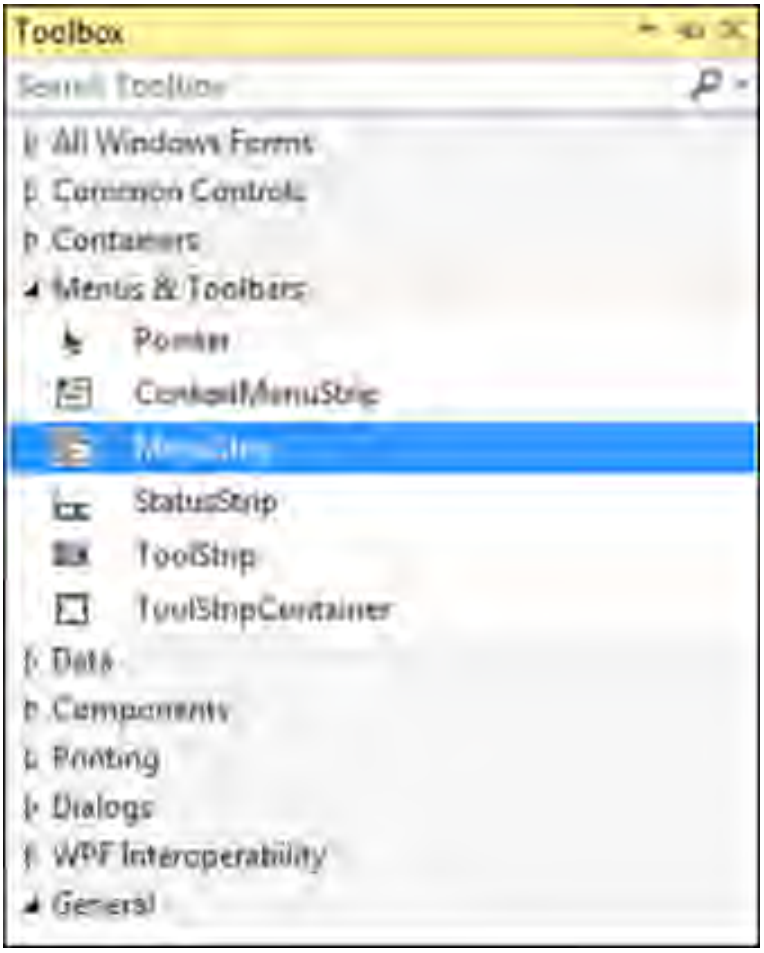


Figure 2.4 Toolbox

You will see your project as shown in Figure 2.5. The bottom of your screen is control itself. To start building your menu, click inside the area "Type Here". Type the word “File” as shown in Figure 2.6.

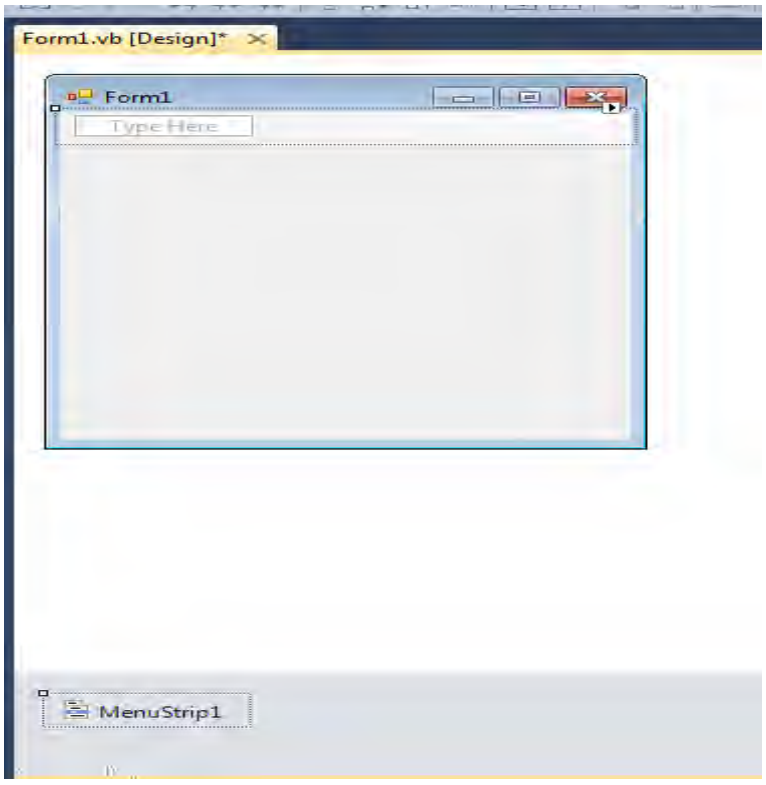


Figure 2.5 Building a Menu

Figure 2. 6 File Menu

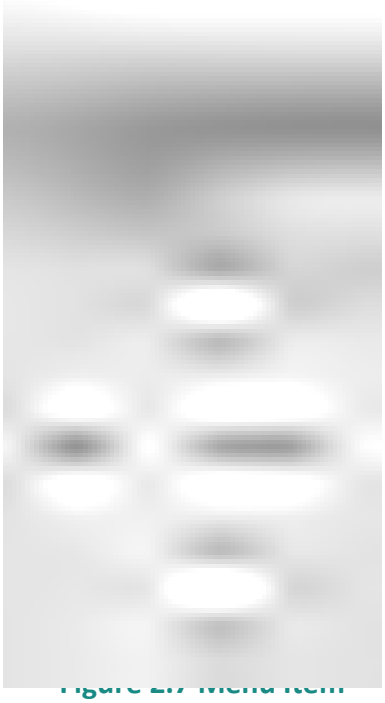


Figure 2.7 Menu Item

Add a “New” , "Open", “Close” and a "Save" item to your menu in the same way as shown in Figure 2.7.

You can add a separator between the "Save" and "Print". To add a separator, click inside the "Type Here” the minus character "-" When you hit the enter key, you'll see the separator appear. Continue your menu as shown in Figure 2.8.

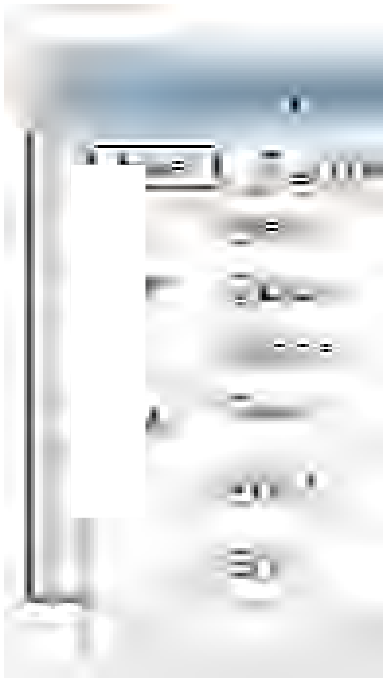


Figure 2.8 Adding Separator

When we create the menu, we can add:

a. underline shorcut
b. key combination shortcut

2.6.1 Underline Shortcut

Click on your New menu item. Position your cursor before the "N" of New .

Type an ampersand symbol (&) as shown in Figure 2.9. If you want an underline shortcut, the ampersand character should be typed before the letter you want underlined

Figure 2.9 Underline Shortcut

To use the underline shortcuts on menus, you first hold down the Alt key on your keyboard. Then type the underline character.

2.6.2 Key combination shortcuts

A key combination shortcut is one that appears at the end of a menu item as shown in Figure 2.8.

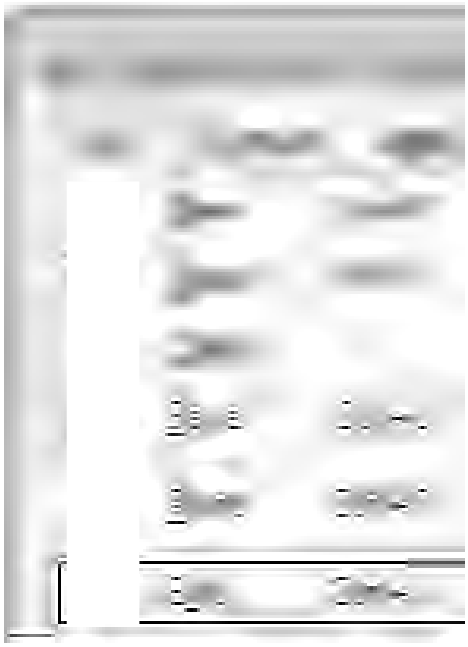


Figure 2.10 Menu Item Underline Shortcut

As example, select the Exit item on your menu. At the properties window, locate the ShortcutKeys. Click the down arrow as shown in Figure 2.11

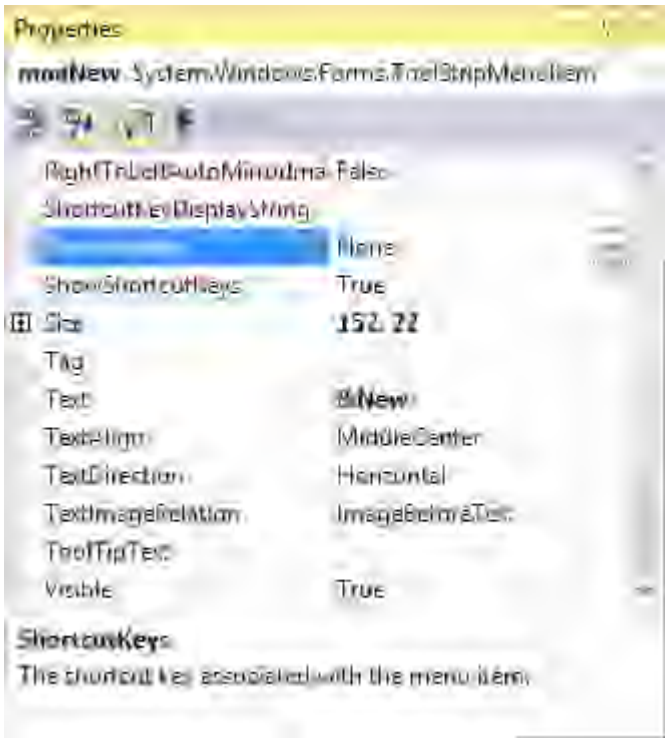


Figure 2.11 Properties Window

Check the Ctrl modifier box, then select the letter "X" from the Key dropdown list as shown in Figure 2.12.

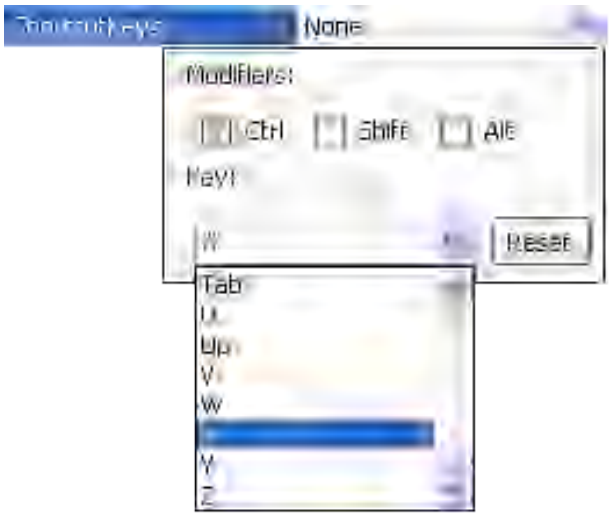


Figure 2.12 Shortcut Keys

Add the following Main Menu items to the menu bar you have already designed as shown in Figure 2.13 and Figure 2.14.

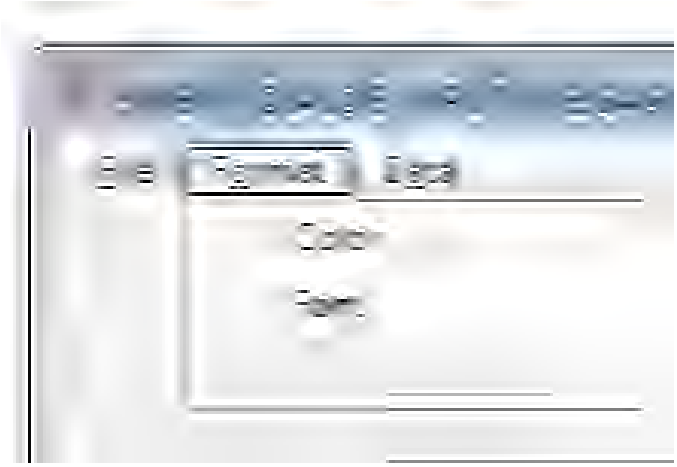


Figure 2.13 Format Menu

Figure 2.14 Data Menu

Add 2 labels to your form as shown in Figure 2.15 and the properties as shown in Table 2.3

Table 2.3 Plan the Objects and Properties

Object	Property
Label1	Name:lblinput Text: blank
Label2	Name: lbdisplay Text: VB.Net Progrramming

Figure 2.15 Adding Label to Form

Now, we will add dialog boxes in same project.

Figure 2.16 ToolBox

To your form, use the toolbox to add a Dialogs control. Double click the control:

- ColorDialog
- FontDialog
- OpenFileDialog
- SaveFileDialog
- PrintDialog

The bottom of your screen will look as shown in Figure 2.17 and name the dialog with the name dlgcolor, dlgfont, dlgopen, dlgsave and dlgprint.

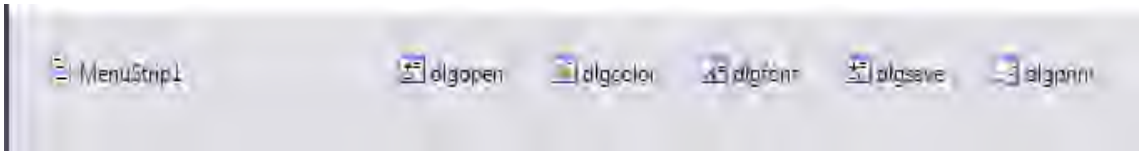


Figure 2.17 Bottom of Screen

```
Private Sub mnuopen_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles mnuopen.Click

dlgopen.ShowDialog()

End Sub

Private Sub mnusave_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles mnusave.Click

dlgsave.ShowDialog()

End Sub

Private Sub mnuprint_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles mnuprint.Click

dlgprint.ShowDialog()

End Sub

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

        dlgcolor.ShowDialog()

        lbldisplay.ForeColor = dlgcolor.Color

        Me.BackColor = dlgcolor.Color

End Sub
```

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

        dlgfont.ShowDialog()

        lbldisplay.Font = dlgfont.Font

End Sub

Private Sub mnuinput_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles mnuinput.Click

lblinput.Text = InputBox("What is your name", "BIODATA")

End Sub


Private Sub mnuexit_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles mnuexit.Click

      Dim intResult As Integer

      intResult = MessageBox.Show("Are you sure?", "EXIT",
      MessageBoxButtons.YesNo, MessageBoxIcon.Information)

      If intResult = Windows.Forms.DialogResult.Yes Then

      End

      ElseIf intResult = Windows.Forms.DialogResult.No Then

      Me.Show()

End If
```

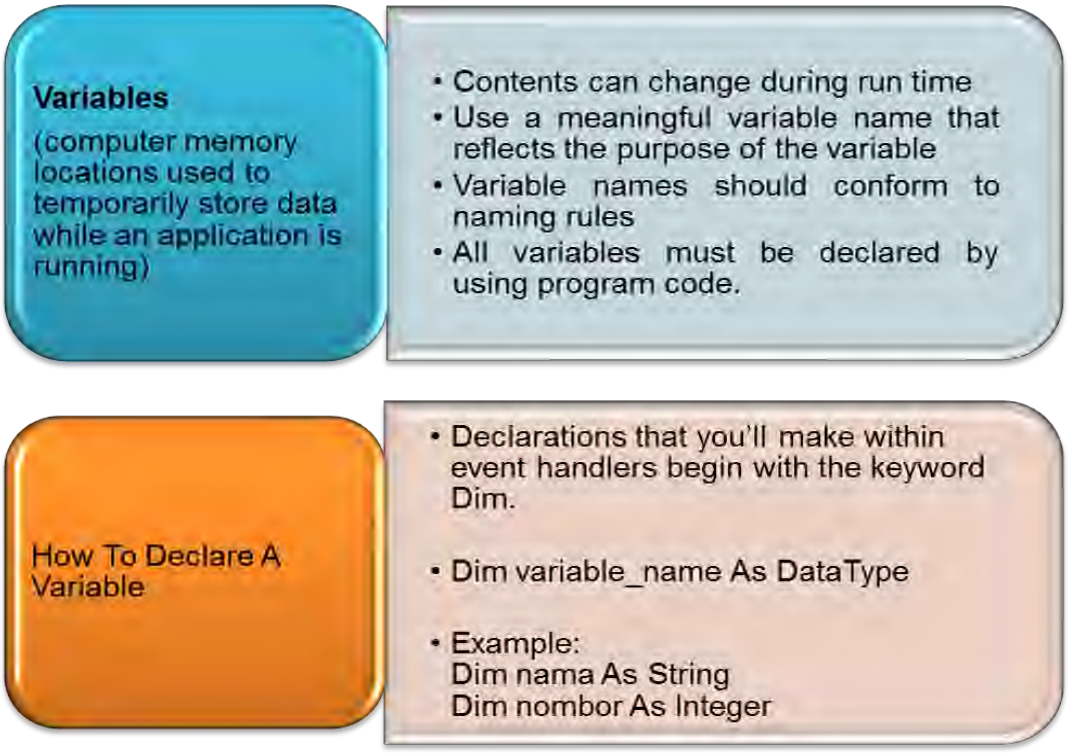
CHAPTER 3

VARIABLE & CONSTANTS

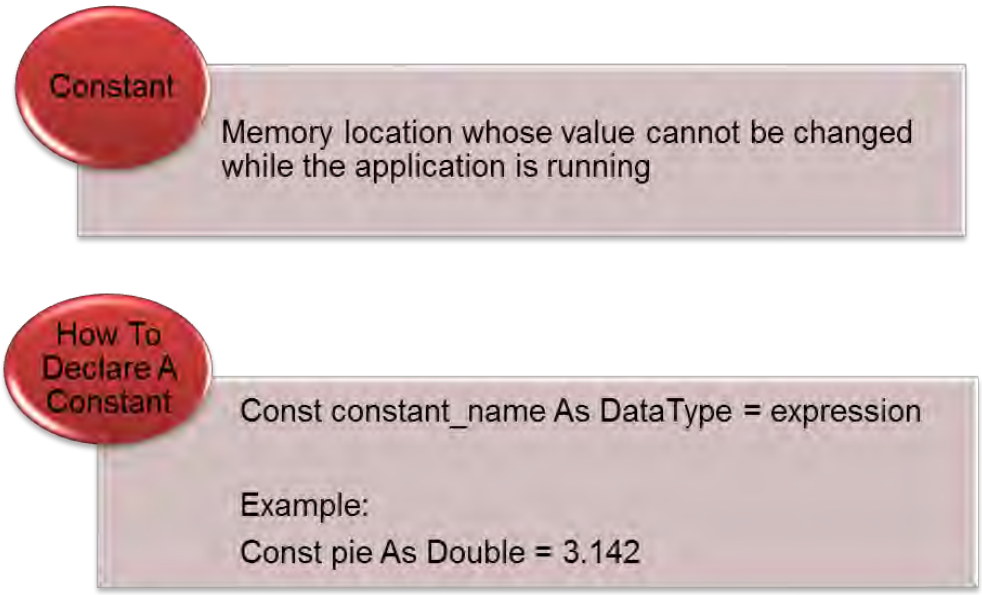
At the completion of this chapter , you will be able to:

- Explain variables and constants.
- Use variables and constant to store data in programs.

3.1 Variable



3.2 Constant



3.3 Area Of Circle Application

Step 1: Design the interface as shown in Figure 3.1

Step 2: Set the objects and properties as shown in Table 3.1.

Step 3: Plan the event procedures as shown in

Table 3.2.

Step 4: Write the code

Step 5: Save and run the project.

Figure 3.1 Graphical User Interface

Table 3.1 Plan the Objects and Properties

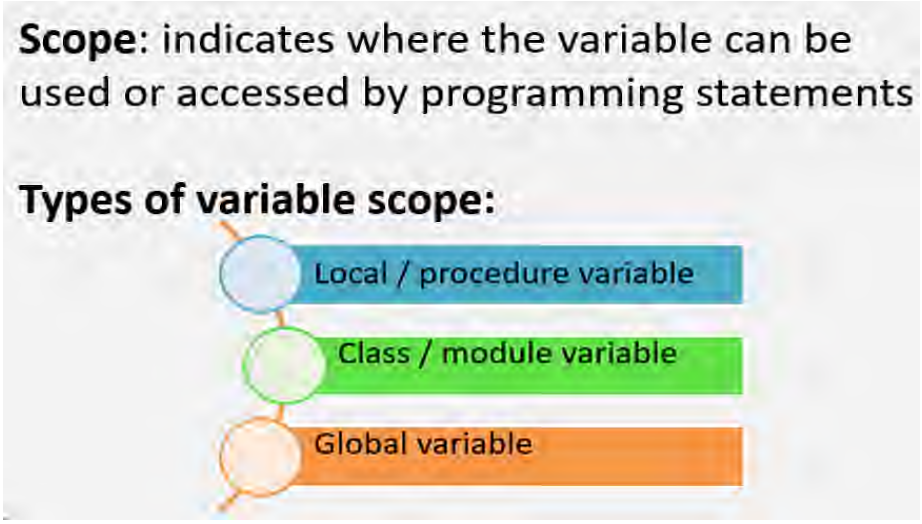
Object	Property
TextBox1	Name:txtradius Text: -
Label1	Name: lblarea Text: -
Button1	Name:btnarea Text: CALCULATE AREA

Table 3.2 Plan the Event Procedures

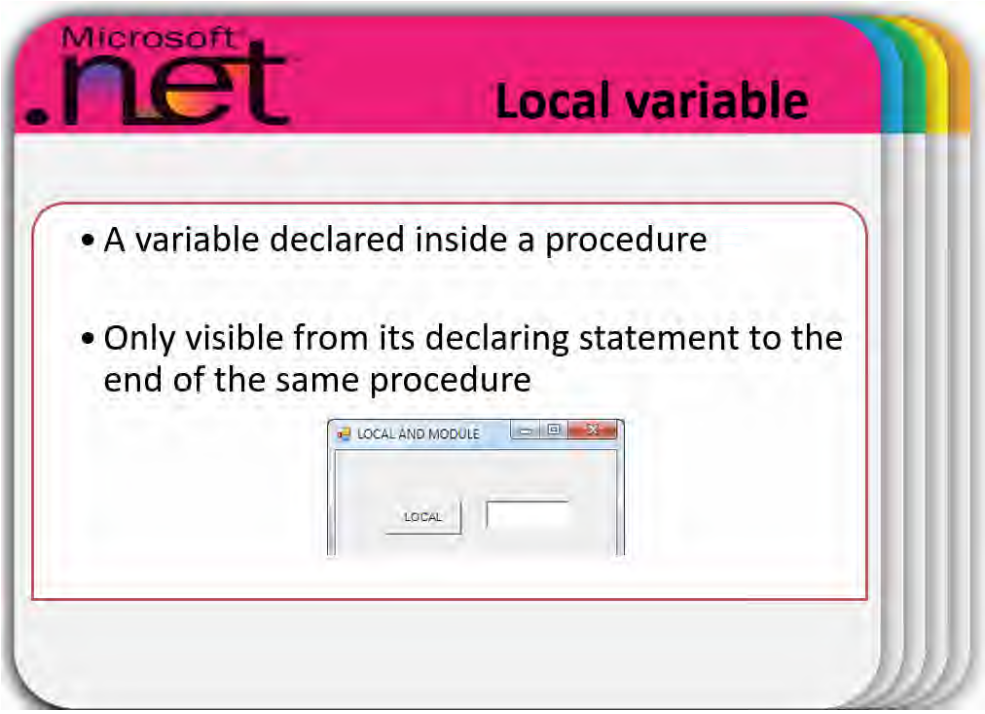
Procedure	Action
btnarea	Calculate an area of circle

```
Private Sub btnarea_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnarea.Click
        Const pi As Double = 3.1415
        Dim r As Double
        Dim area As Double
        r = txtradius.Text

        area = pi * r * r
        lblarea.Text = area
    End Sub
```

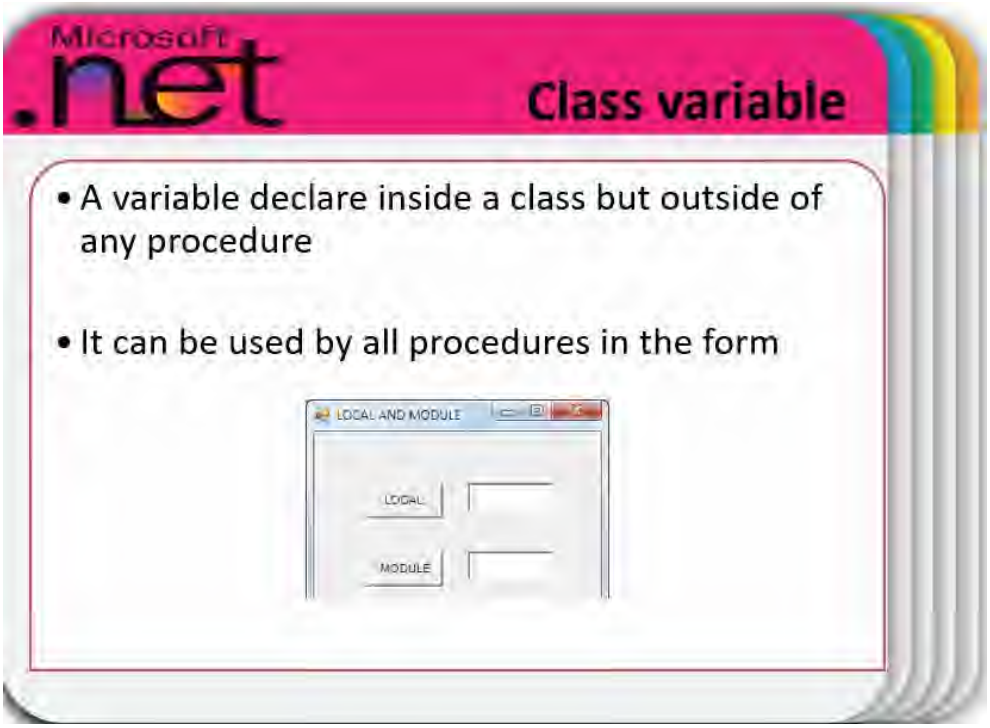
3.4 Scope Variable



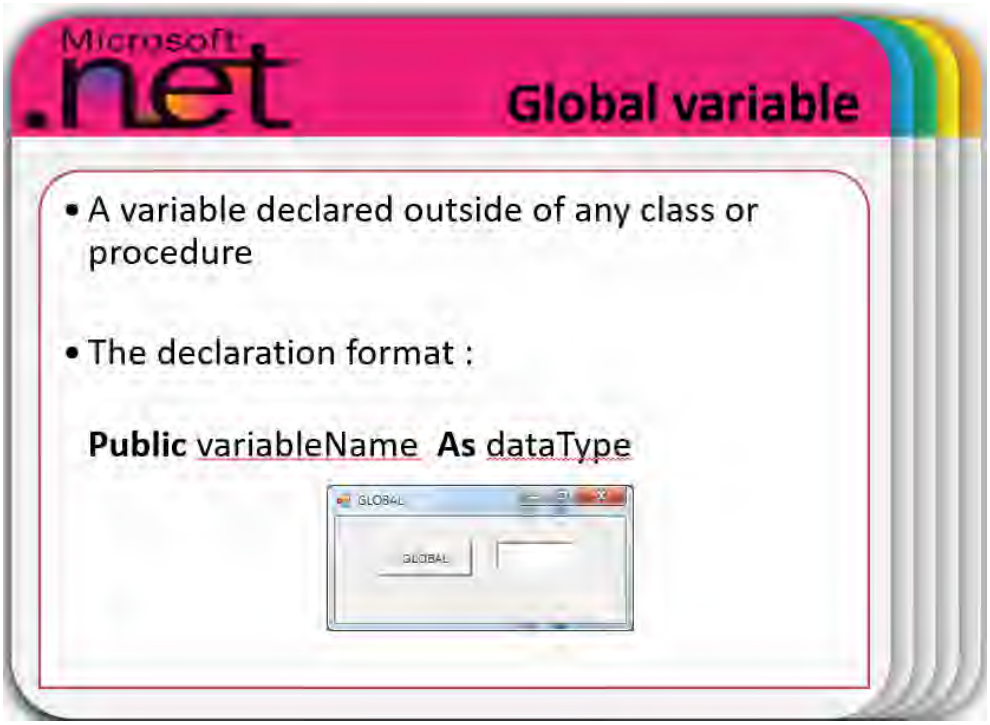
3.4.1 Local Variable



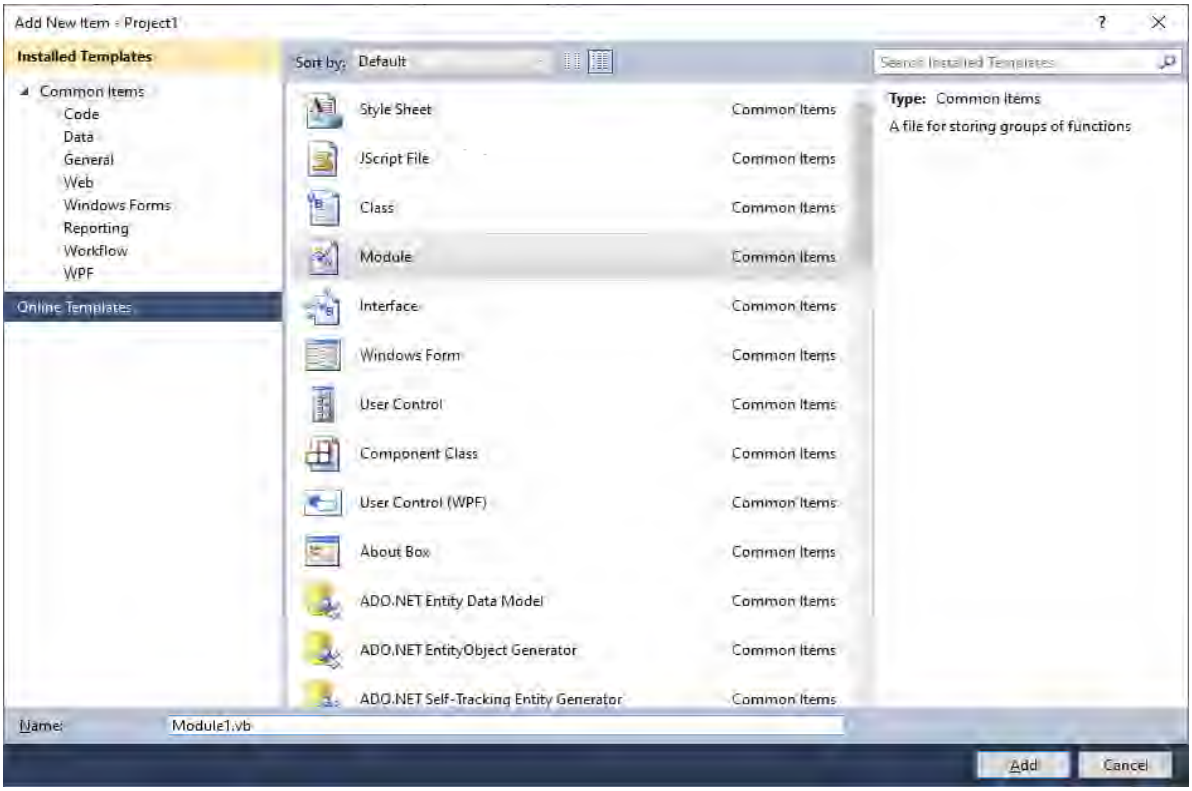
3.4.2 Class Variable



3.4.3 Global Variable



3.4.5 Scope Variable Application



```
'LOCAL VARIABLE

Private Sub btnlocal_Click(ByVal eventSender As System.Object, ByVal
eventArgs As System.EventArgs) Handles btnlocal.Click

Dim x As Integer

x = 30

Text1.Text = x

End Sub
```

```
Class Form 1

'MODULE VARIABLE

Dim x As Integer


Private Sub btnmodule_Click(ByVal eventSender As System.Object, ByVal
eventArgs As System.EventArgs) Handles btnmodule.Click

Text2.Text = x

End Sub
```

```
Module Module1

'GLOBAL VARIABLE


Public x As Integer

End Module
```

Microsoft .net Formatting Numeric Output

Formatting: specifying the number of decimal places and any special characters to display
`ToString` method of a variable can be used to format a number

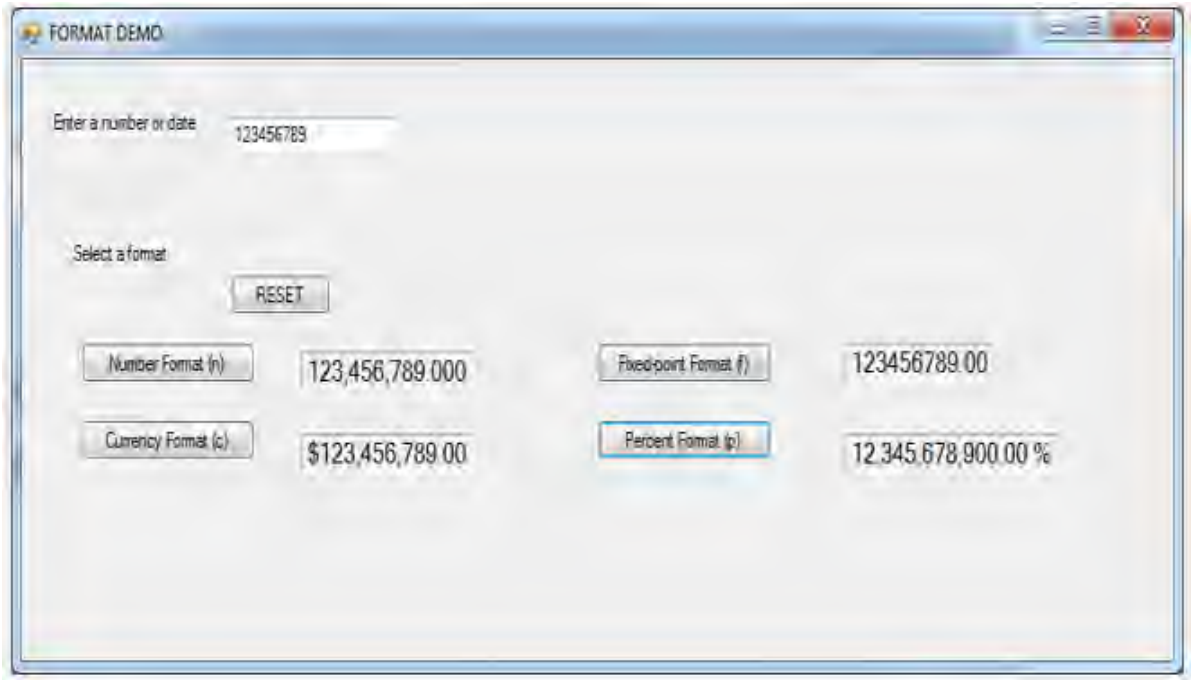
Precision specifier: controls the number of significant digits or zeros to the right of the decimal point

Microsoft .net How To Format a Number

HOW TO Format a Number

Syntax
variableName.**ToString**(*formatString*)

Format specifier (Name)	Description
C or c (Currency)	displays the string with a dollar sign; includes a thousands separator (if appropriate); negative values are enclosed in parentheses
N or n (Number)	similar to the Currency format, but does not include a dollar sign and negative values are preceded by a minus sign
F or f (Fixed-point)	same as the Number format, but does not include a thousands separator
P or p (Percent)	multiplies the value by 100 and displays the result with a percent sign; negative values are preceded by a minus sign


FORMAT DEMO
Enter a number or date
123456789
Select a format
RESET
Number Format (n)
123,456,789.000
Fixed-point Format (f)
123456789.00
Currency Format (c)
$123,456,789.00
Percent Format (p)
12,345,678,900.00 %

```
Public Class Form1

Dim input As Double

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btn_n.Click


        'convert string to numeric data type

        Double.TryParse(txtinput.Text, input)


        'if user input string data type, an error will occured

        input = txtinput.Text

        lbl_n.Text = input.ToString("n3")

End Sub


Private Sub btn_f_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btn_f.Click

        lbl_f.Text = input.ToString("f2")

End Sub


Private Sub Button1_Click_1(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btn_c.Click

        lbl_c.Text = input.ToString("c")

End Sub


Private Sub btn_p_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btn_p.Click

        lbl_p.Text = input.ToString("p")

End Sub


Private Sub btnreset_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnreset.Click

        txtinput.Text = String.Empty

        lbl_f.Text = String.Empty

        lbl_c.Text = String.Empty

        lbl_p.Text = String.Empty

        lbl_n.Text = String.Empty

End Sub
```

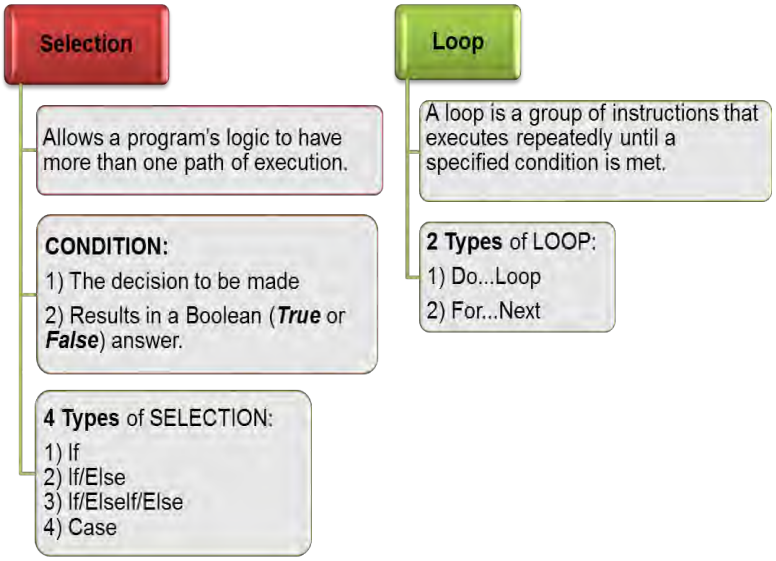
CHAPTER 4

CONTROL PROGRAM FLOWS

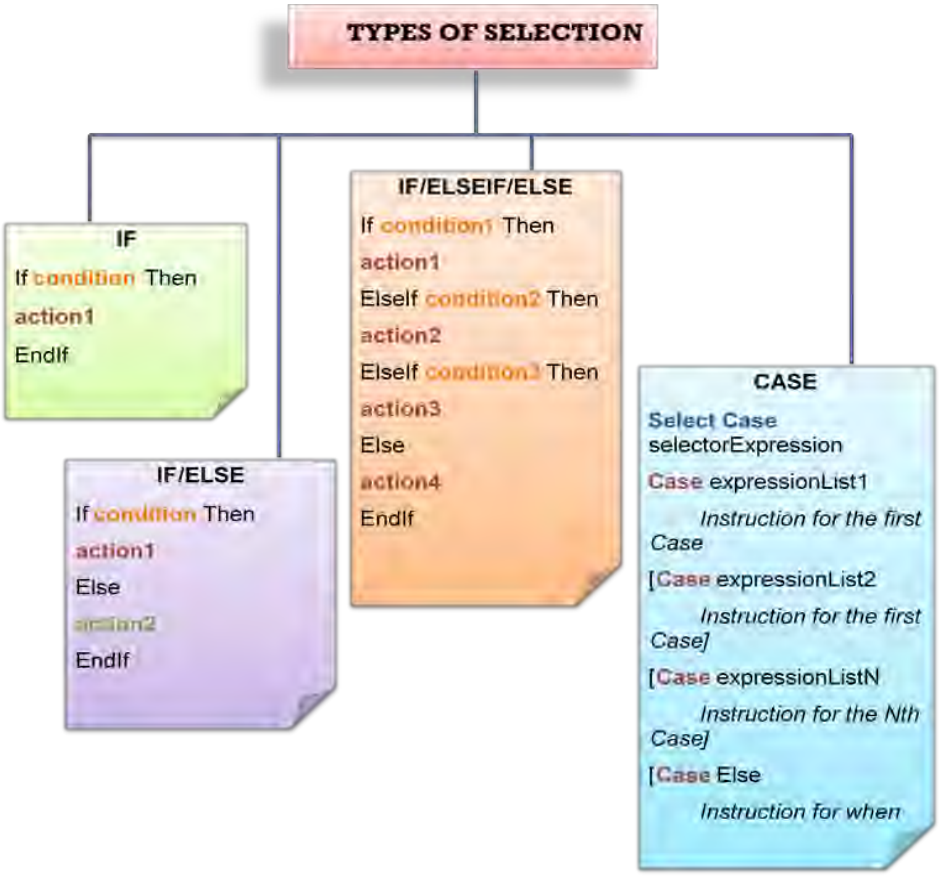
At the completion of this chapter , you will be able to:

- Identify the syntax of selection and looping statements
- Use selection and looping statements in application
- Differentiate between the various type of looping statements

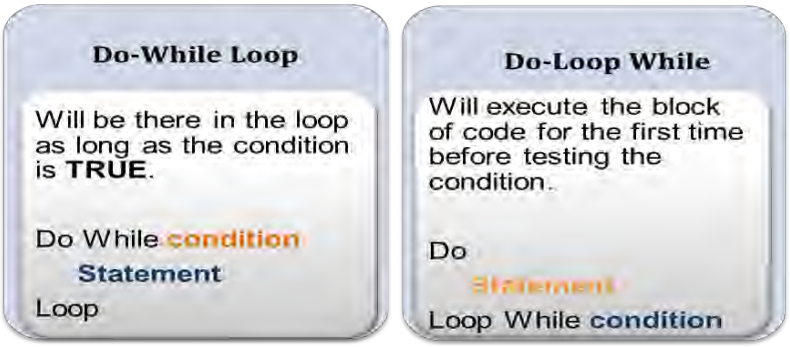
4.1 Control Structure

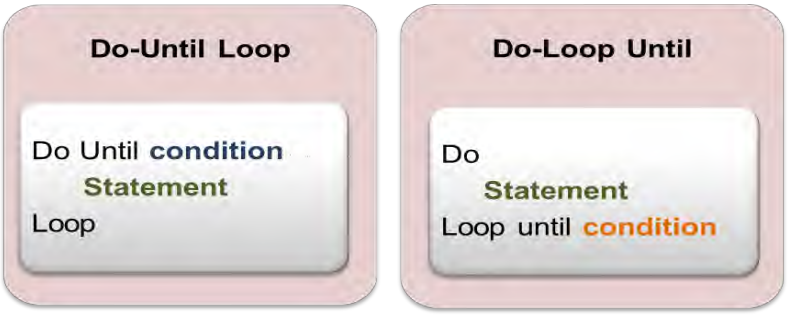


4.2 Types Of Selection



4.3 Types Of Loop





Use the number line technique tu trace the output

SCAN ME

URL: https://qrinfopoint.com/ iTKmdPG/

.

4.4 Grading Application

Step 1: Design the interface as shown in Figure 4.1

Step 2: Set the objects and properties as shown in

Table 4.1.

Step 3: Plan the event procedures as shown in

Table 4.2.

Step 4: Write the code

Step 5: Save and run the project.

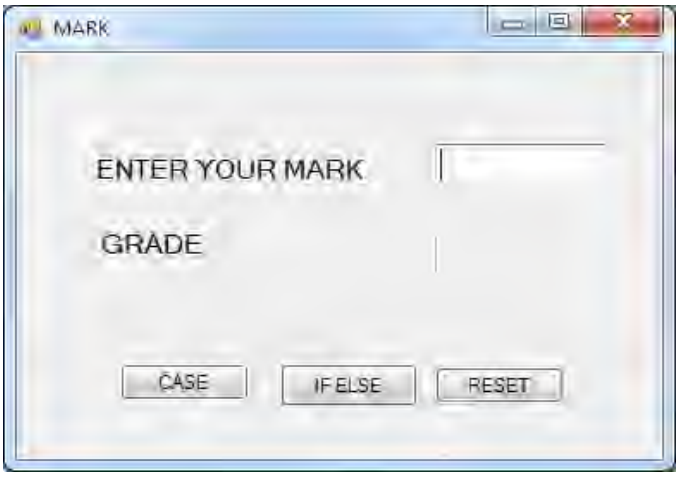


Figure 4.1 Graphical User Interface

Table 4.1 Plan the Objects and Properties

Object	Property
TextBox1	Name:txtmark Text: -
Label1	Name: lblgred Text: -
Button1	Name:btncase Text: CASE
Button2	Name:btnif Text: IF ELSE
Button3	Name:btnreset Text: RESET

Table 4.2 Plan the Event Procedures

Procedure	Action
btncase	Grade mark using case
btnif	Grade mark using if else
btnreset	Clear the text boxes and label

```
Private Sub btncase_Click(sender As
System.Object, e As System.EventArgs) Handles
btnresult.Click

        Dim markah As Integer

        markah = txtmark.Text

        Select Case (markah)

            Case 80 To 100

                lblgrade.Text = "A"

            Case 60 To 79

                lblgrade.Text = "B"

            Case 50 To 59

                lblgrade.Text = "C"

            Case 40 To 49

                lblgrade.Text = "D"

            Case 0 To 39

                lblgrade.Text = "FAIL"

            Case Else

                lblgrade.Text = "WRONG INPUT"

        End Select

    End Sub
```

```
Private Sub btnif_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles Button2.Click

        Dim markah As Integer

        markah = txtmark.Text

      If markah >= 80 And markah <= 100 Then

            lblgrade.Text = "You got A"

    ElseIf markah >= 60 And markah <= 79 Then

            lblgrade.Text = "You got B"

    ElseIf markah >= 50 And markah <= 59 Then

            lblgrade.Text = "You got C"

    ElseIf markah >= 40 And markah <= 49 Then

            lblgrade.Text = "you got D"

     ElseIf markah >= 0 And markah <= 39 Then

            lblgrade.Text = "FAIL"

        Else

            lblgrade.Text = "WRONG INPUT"

        End If

    End Sub
```

CHAPTER 5

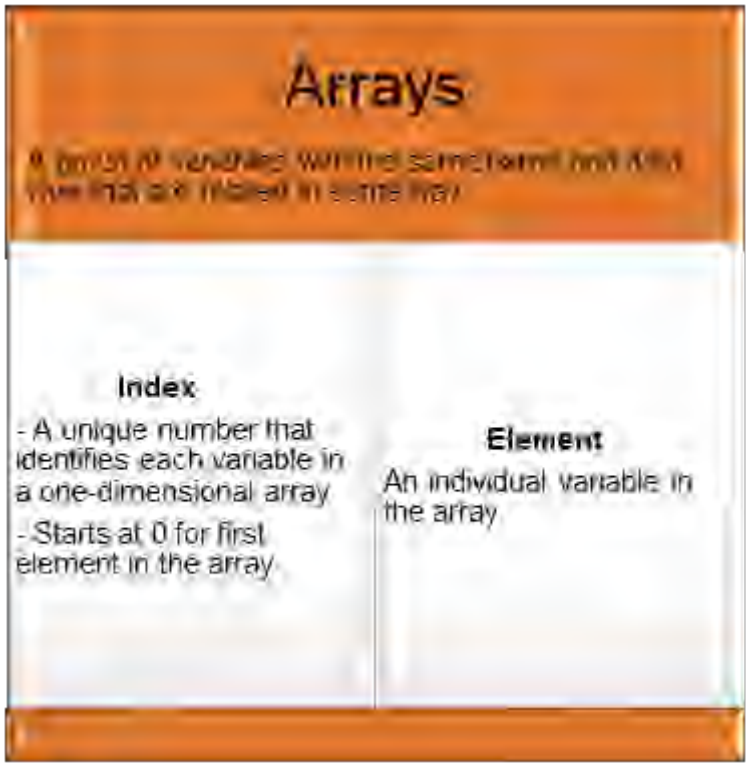
ARRAYS

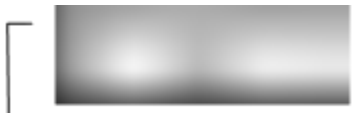
At the completion of this chapter , you will be able to:

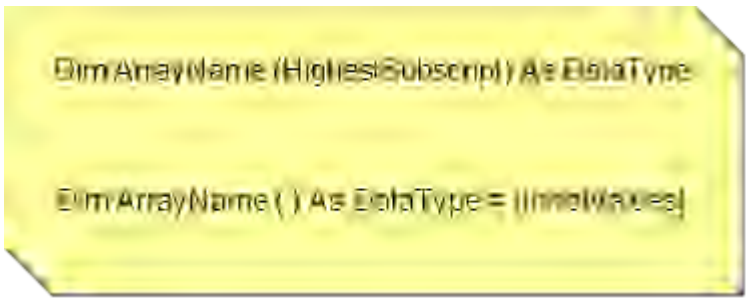
- Define a control array and illustrate with an example.
- Declare a single and multidimenional array.
- Use arrays in application.

EXERCISES

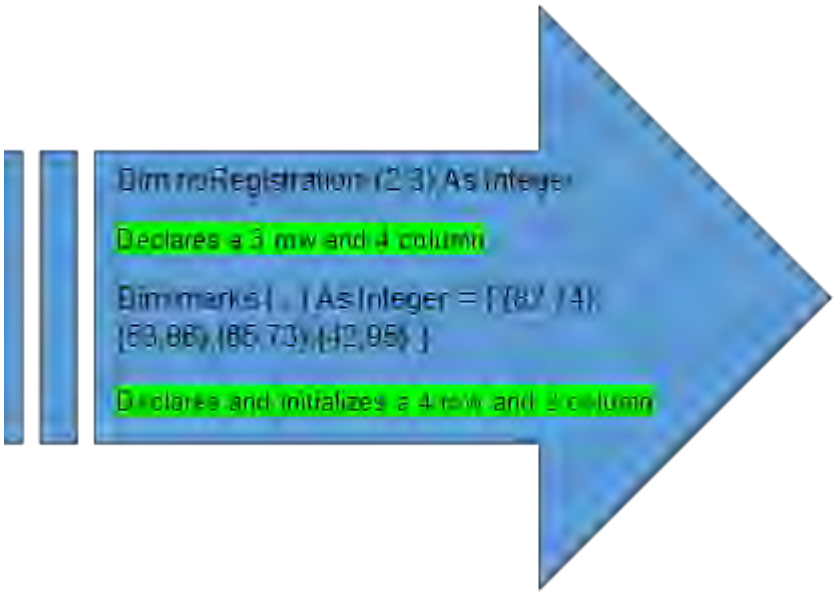
5.1 Control Arrays







How To Declare A 2D Array


Dim noRegistration (2,3) As Integer
Declares a 3 row and 4 column
Dim marks (,) As Integer = {{80, 74},
{63,86},{85,73},{42,95}}
Declares and initializes a 4 row and 2 column

5.2 Array 1D and 2D Application

Step 1: Design the interface as shown in Figure 5.1

Step 2: Set the objects and properties as shown in

Table 5.1.

Step 3: Plan the event procedures as shown in

Table 5.2.

Step 4: Write the code

Step 5: Save and run the project.

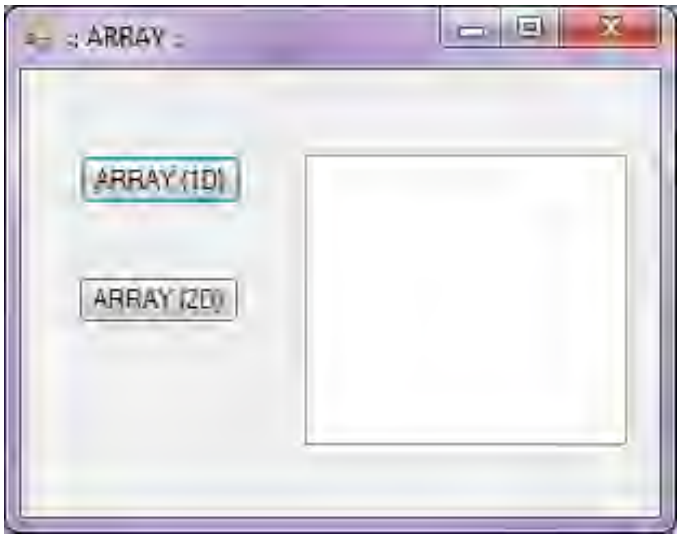


Figure 5.1 Graphical User Interface

Table 5.1 Plan the Objects and Properties

Object	Property
Button1	Name:btnone Text: ARRAY (1D)
Button2	Name: btntwo Text: ARRAY (2D)
ListBox	Name:lstoutput Text: -

Table 5.2 Plan the Event Procedures

Procedure	Action
btnone	View 1D array example
btntwo	View 2D array example

```
Private Sub btnone_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles Button1.Click

        Dim num(2) As Integer

        Dim x As Integer

        num(0) = 10

        num(1) = 27

        num(2) = 30

        For x = 0 To 2

            lstoutput.Items.Add(num(x))

        Next x

End Sub
```

```
Private Sub btntwo_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles Button2.Click

        Dim msg(1, 1) As String

        Dim i, j As Integer

        msg(0, 0) = "Visual Programming"

        msg(0, 1) = "Java"

        msg(1, 0) = "Web"

        msg(1, 1) = "C++"

        For i = 0 To 1

            For j = 0 To 1

                lstoutput.Items.Add(msg(i,
j))

            Next j

        Next i

    End Sub
```

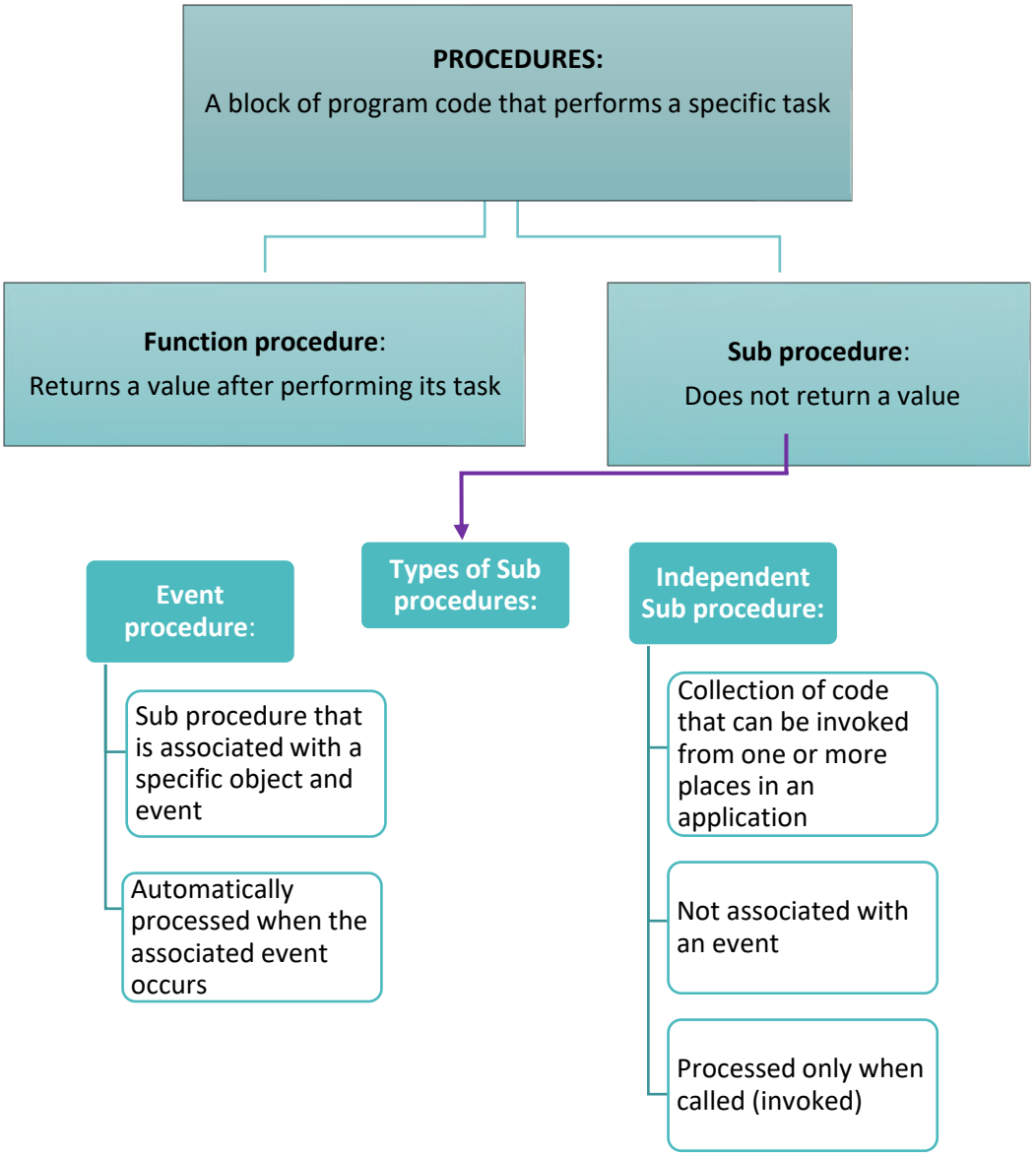
CHAPTER 6

PROCEDURES & FUNCTION

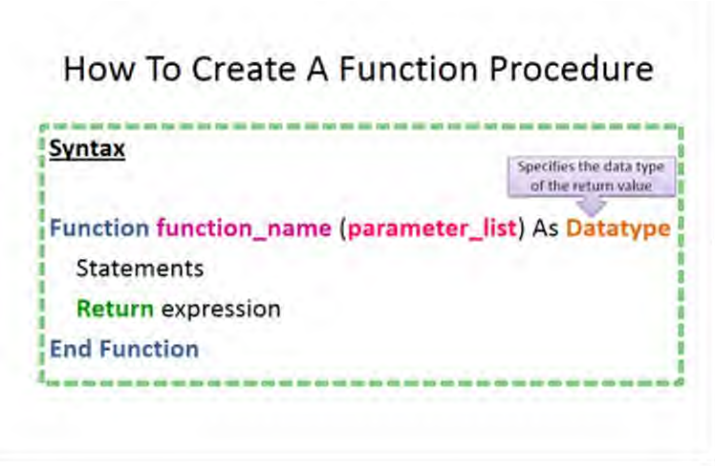
At the completion of this chapter, you will able to:

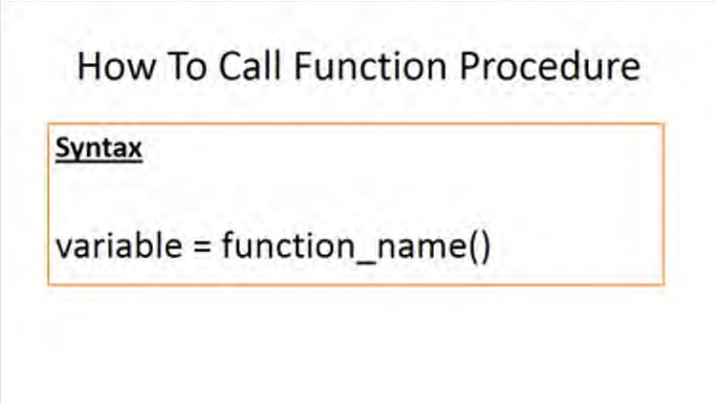
- Explain the difference between Sub Procedures and Function Procedures
- Implement Sub Procedures and Function Procedures.
- Pass information to a procedure
- Explain the difference between passing data by value and passing data by reference

6.1 Procedures



6.2 Function


How To Create A Function Procedure
Syntax
Specifies the data type of the return value
Function function_name (parameter_list) As Datatype
Statements
Return expression
End Function


How To Call Function Procedure
Syntax
variable = function_name()

6.3 Area Of Circle Using Function Application

Step 1: Design the interface as shown in Figure 6.1

Step 2: Set the objects and properties as shown in Table 6.1.

Step 3: Plan the event procedures as shown in Table 6.2.

Step 4: Write the code

Step 5: Save and run the project.

Figure 6.1 Graphical User Interface

Table 6.1 Plan the Objects and Properties

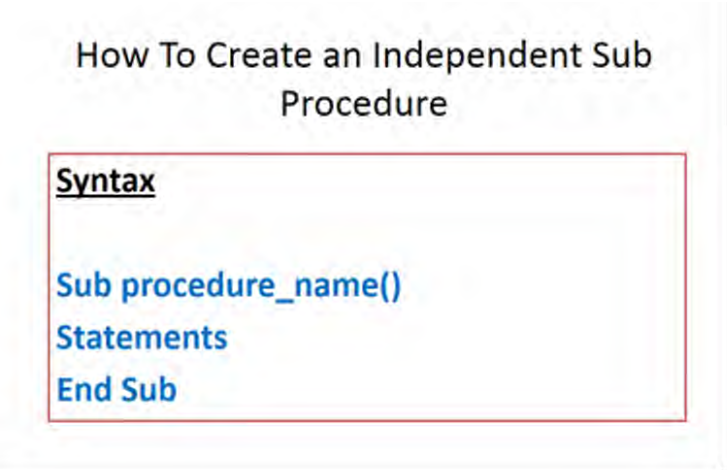
Object	Property
TextBox1	Name:txtradius Text: -
Label1	Name: lblarea Text: -
Button1	Name:btnarea Text: CALCULATE AREA

Table 6.2 Plan the Event Procedures

Procedure	Action
btnarea	Calculate an area of circle

```
Function GetCircleArea(ByVal rad As Integer) As Double
        Dim luas As Double
        luas = 3.142 * rad * rad
        Return luas
End Function

Private Sub btncal_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btncal.Click
        Dim radius As Integer
        Dim kira As Double
        radius = txtradius.Text
        kira = GetCircleArea(radius)
        lblarea.Text = kira.ToString("n2")
End Sub
```

6.4 Independent Sub Procedure



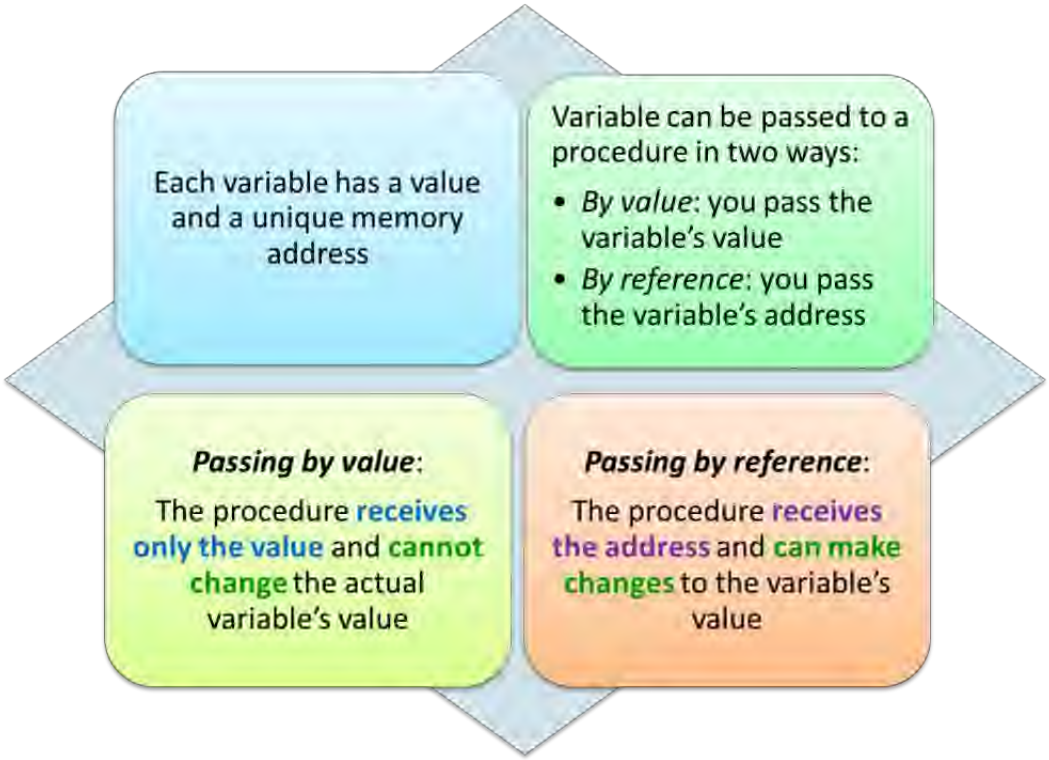
6.5 Independent Sub Procedure Application

Add listbox and button to your form and type a code below.

```
Private Sub btngo_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btngo.Click

      lstoutput.Items.Add("Hello from GO button")

      lstoutput.Items.Add("Now I am calling
      DisplayMessage procedure")

      DisplayMessage()

      lstoutput.Items.Add("Now I am back in GO button")

    End Sub
```

```
Sub DisplayMessage()

  lstoutput.Items.Add("")

  lstoutput.Items.Add("Hi from DisplayMessage
procedure")

  lstoutput.Items.Add("")

End Sub
```

6.6 Passing Argument By Val and By Ref



6.6 By Val and By RApplication

Add listbox and button to your form and type a code below.

```
Private Sub btnval_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles btnview.Click
   Dim num As Integer = 100
   lstval.Items.Add("Inside View Button.
   The value of num is  " & num)
   Change(num)
   lstval.Items.Add("Now back in View Button.
   The value of num is  " & num)
End Sub
Sub Change(ByVal nombor As Integer)
   lstval.Items.Add("Inside Change Procedure.
   The value of num is  " & nombor)
   nombor = 5
  lstval.Items.Add("The number is now  " &
  nombor)
  End Sub
```

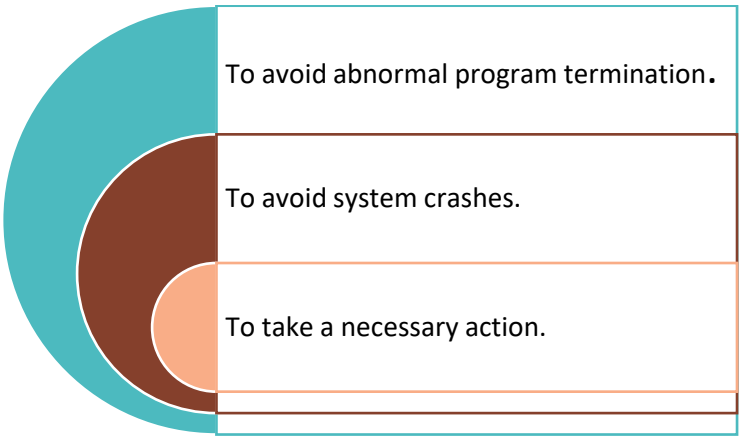
CHAPTER 7

EXCEPTION HANDLING

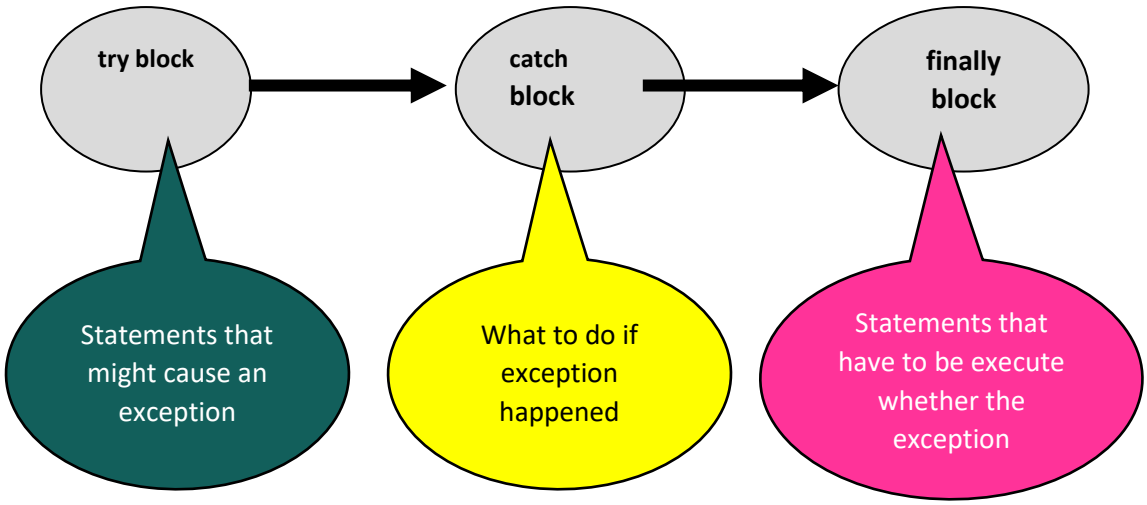
At the completion of this chapter , you will be able to:

- Define a control array and illustrate with an example.
- Declare a single and multidimenional array.
- Use arrays in application.

7.1 Exception Handling



7.2 Try, Catch, Finally



7.3 Error Description Application

Step 1: Design the interface as shown in Figure 7.1

Step 2: Set the objects and properties as shown in Table 7.1.

Step 3: Plan the event procedures as shown in Table 7.2.

Step 4: Write the code

Step 5: Save and run the project.

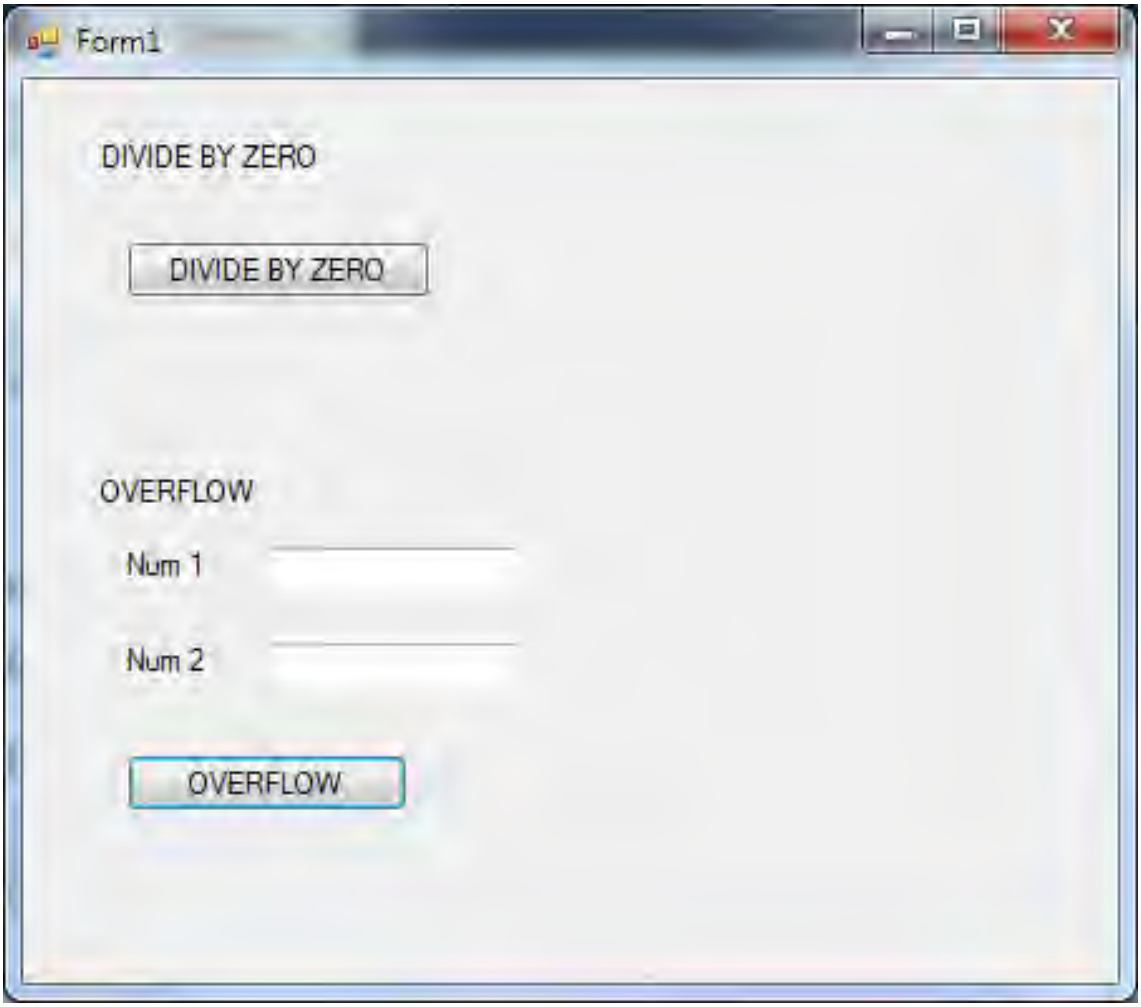


Figure 7.1 Graphical User Interface

Table 7.1 Plan the Objects and Properties

Object	Property
Button1	Name:btnzero Text: DIVIDE BY ZERO
Button2	Name: btnoverflow Text: OVERFLOW
TextBox1	Name:txtnum1 Text: -
TextBox2	Name:txtnum2 Text: -

```
Private Sub btnoverflow_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnoverflow.Click
        Dim jaw As Short
        Try
            jaw = Val(txtnum1.Text) + Val(txtnum2.Text)
            MessageBox.Show(jaw)

        Catch When Err.Number
            MessageBox.Show(Err.Description)
        End Try
    End Sub
```

Table 7.2 Plan the Event Procedures

Procedure	Action
btnzero	View divide by zero exception
btnoverflow	View overflow exception

```
Private Sub btnzero_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnzero.Click


Try

      Dim a, b, c As Integer

            a = 3

            b = 0

            c = a Mod b

      Catch When Err.Number

      MessageBox.Show(Err.Description & "Try again",
"ERROR")

        End Try

End Sub
```

```
Private Sub btnoverflow_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnoverflow.Click

        Dim jaw As Short

        Try

            jaw = Val(txtnum1.Text) + Val(txtnum2.Text)

            MessageBox.Show(jaw)


        Catch When Err.Number

            MessageBox.Show(Err.Description)

        End Try

    End Sub
```

CHAPTER 8

STRING FUNCTION

At the completion of this chapter, you will able to:

- Describe Strings and String object.
- Implement the String object to handle text.
- Implement the StringBuilder object to manipulate strings of text.

8.1 String Definition

A string is a data type used in programming, such as an integer and floating point unit, but is used to represent text rather than numbers.

It is comprised of a set of characters that can also contain spaces and numbers.

8.2 String Method

Method	General Syntax
1. IsNumeric	IsNumeric(StringExpression)
2. ToLower	StringExpression.ToLower()
3. ToUpper	StringExpression.ToUpper()
4. Length	StringExpression.Length
5. TrimStart	StringExpression.TrimStart()
6. TrimEnd	StringExpression.TrimEnd()
7. Trim	StringExpression.Trim()
8. Substring	StringExpression.SubString(Start) StringExpression.SubString(Start, Length)
9. IndexOf	StringExpression.IndexOf(SearchString)

8.3 String Application

Step 1: Design the interface as shown in Figure 8.1

Step 2: Set the objects and properties as shown in Table 8.1.

Step 3: Plan the event procedures as shown in Table 8.2.

Step 4: Write the code

Step 5: Save and run the project.

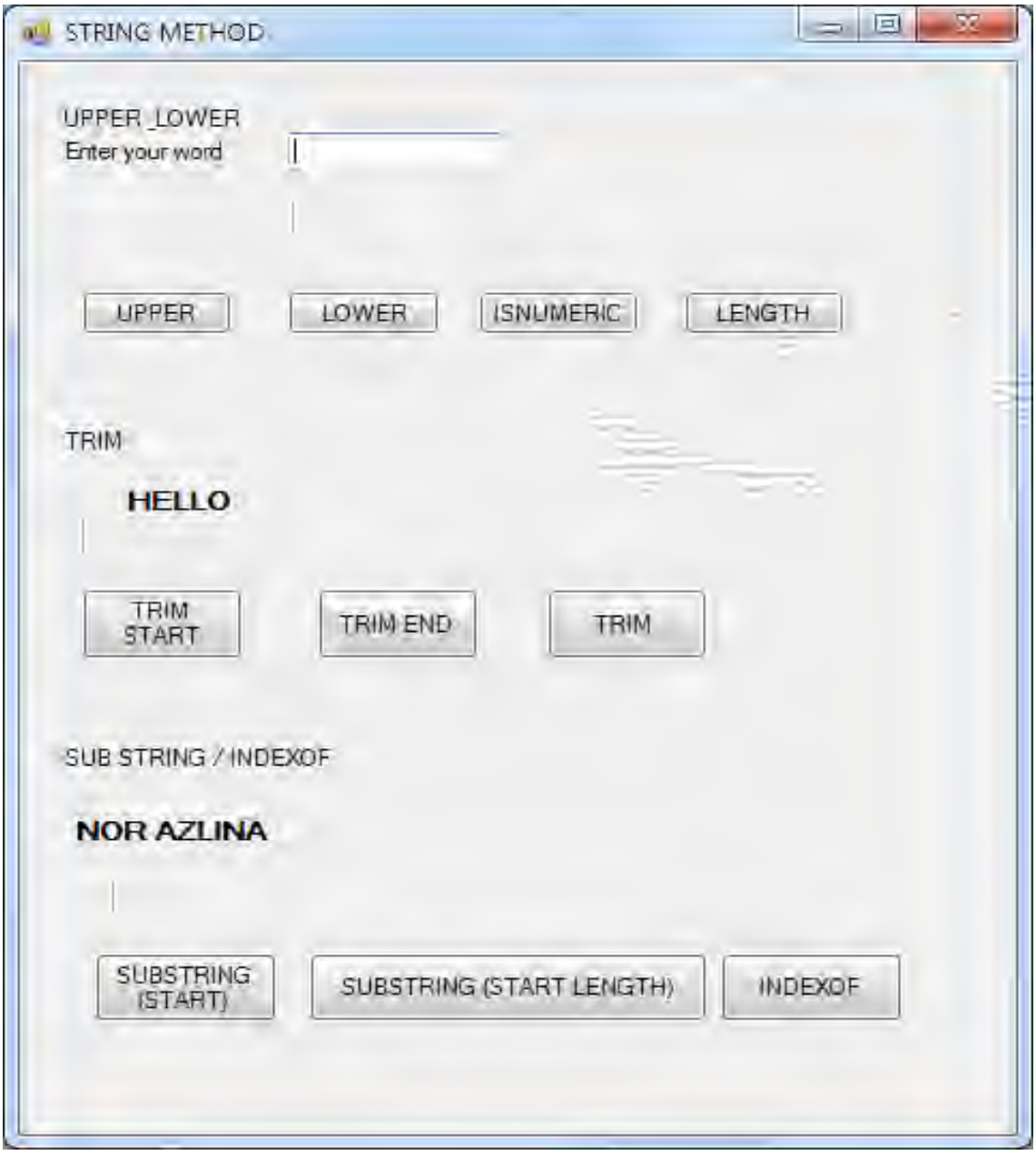


Figure 8.1 Graphical User Interface

Table 8.1 Plan the Objects and Properties

Object	Property
Button1	Name:btnup Text: UPPER
Button2	Name: btnlow Text:LOWER
Button3	Name:btnnumeric Text:ISNUMERIC
Button4	Name: btnlength Text:LENGTH
Button5	Name:btntrimstart Text: TRIM START
Button6	Name: btntrimend Text: TRIM END
Button7	Name:btntrim Text: TRIM
Button8	Name: btnsubstart Text: SUBSTRING (START)
Button9	Name:btnsublength Text: SUBSTRING (START LENGTH)
Button10	Name: btnindex Text: INDEXOF

```
Public Class Form1

Private Sub btnup_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles btnup.Click

        lbloutput.Text=txtinput.Text.ToUpper()

End Sub

Private Sub btnlow_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles btnlow.Click

      lbloutput.Text = txtinput.Text.ToLower()

End Sub

Private Sub btnnumeric_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnnumeric.Click

        If IsNumeric(txtinput.Text) Then
            lbloutput.Text = "It's a number"
        Else
            lbloutput.Text = "It's NOT a number"
        End If
End Sub
```

Table 8.2 Plan the Event Procedures

Procedure	Action
Button1	Returns the uppercase equivalent of a string
Button2	Returns the lowercase equivalent of a string
Button3	Accepts a string as its argument and returns True if the string contains a number
Button4	Returns the number of character in a string
Button5	Returns a copy of a string without leading spaces
Button6	Returns a copy of a string without trailing spaces
Button7	Returns a copy of a string without leading or trailing spaces
Button8	Search for the first occurrence of a character or string within a string
Button9	Extracts a string from within a string
Button10	Returns the index of the first occurrence of the specified substring.

```
Private Sub btnlength_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnlength.Click

      lbloutput.Text = txtinput.Text.Length

End Sub

Private Sub btnstart_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btbts.Click

      lblmsgtrim.Text =lblgreeting.Text.TrimStart()

End Sub

Private Sub btntend_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnte.Click

      lblmsgtrim.Text = lblgreeting.Text.TrimEnd()

End Sub

Private Sub btntrim_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnt.Click

      lblmsgtrim.Text = lblgreeting.Text.Trim()

End Sub
```

```
Private Sub btnsub_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnsub.Click

      lblsubout.Text = lblsubin.Text.Substring(2)

End Sub

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnsubsl.Click

      lblsubout.Text =lblsubin.Text.Substring(0, 5)

End Sub

Private Sub btnindex_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnindex.Click

'If search string caps lock or small from original, the output display
-1

      lblsubout.Text = lblsubin.Text.IndexOf("Z")

End Sub

End Class
```

CHAPTER 9

OBJECT ORINTED PROGRAMMING

At the completion of this chapter, you will able to:

- Describe Strings and String object.
- Implement the String object to handle text.
- Implement the StringBuilder object to manipulate strings of text.

8.1 String Definition

A string is a data type used in programming, such as an integer and floating-point unit, but is used to represent text rather than numbers.

It is comprised of a set of characters that can also contain spaces and numbers.

8.2 String Method

Method	General Syntax
1. IsNumeric	IsNumeric(StringExpression)
2. ToLower	StringExpression.ToLower()
3. ToUpper	StringExpression.ToUpper()
4. Length	StringExpression.Length
5. TrimStart	StringExpression.TrimStart()
6. TrimEnd	StringExpression.TrimEnd()
7. Trim	StringExpression.Trim()
8. Substring	StringExpression.SubString(Start) StringExpression.SubString(Start, Length)
9. IndexOf	StringExpression.IndexOf(SearchString)

8.3 String Application

Step 1: Design the interface as shown in Figure 8.1

Step 2: Set the objects and properties as shown in Table 8.1.

Step 3: Plan the event procedures as shown in Table 8.2.

Step 4: Write the code

Step 5: Save and run the project.

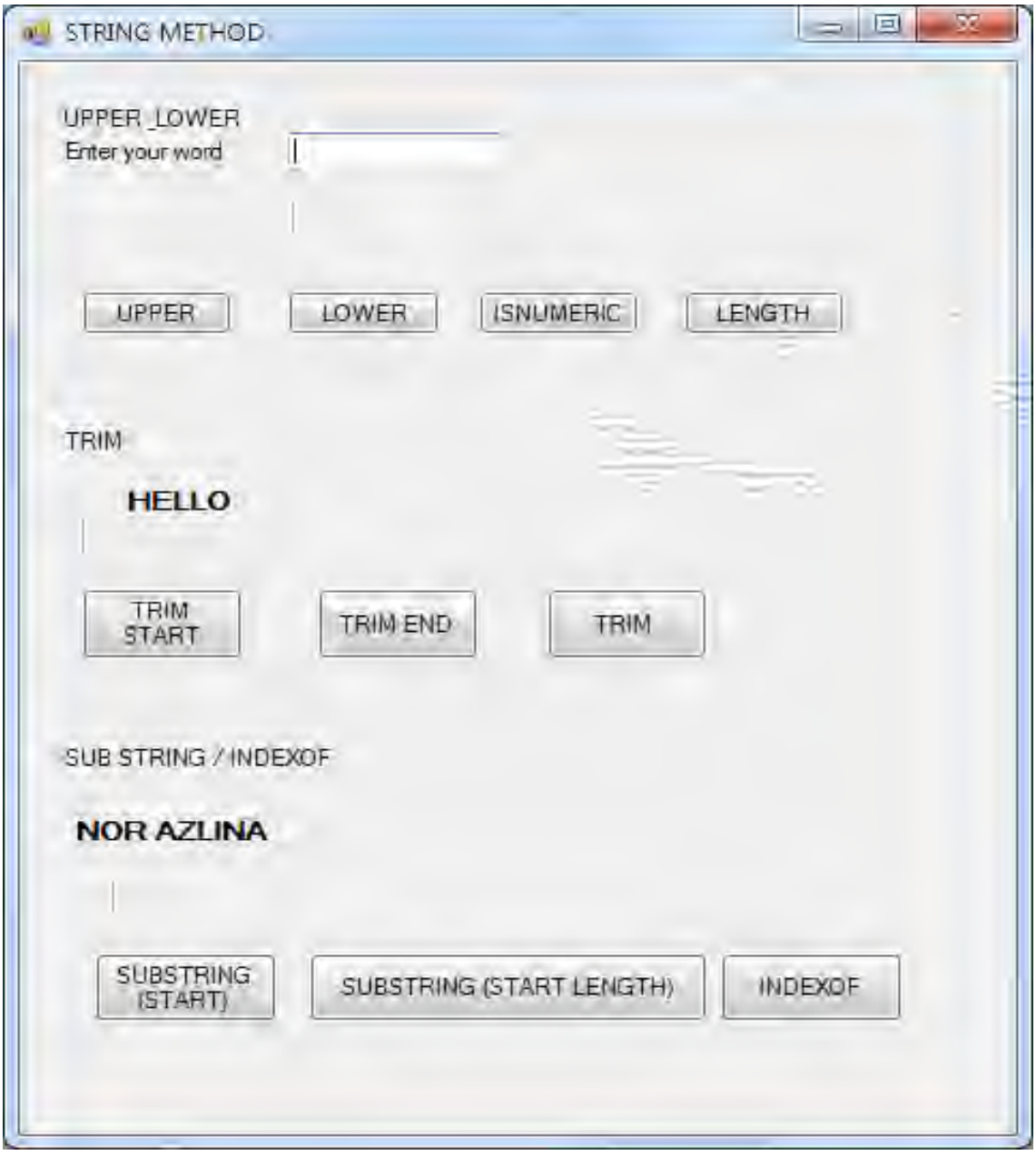


Figure 8.1 Graphical User Interface

Table 8.1 Plan the Objects and Properties

Object	Property
Button1	Name:btnup Text: UPPER
Button2	Name: btnlow Text:LOWER
Button3	Name:btnnumeric Text:ISNUMERIC
Button4	Name: btnlength Text:LENGTH
Button5	Name:btntrimstart Text: TRIM START
Button6	Name: btntrimend Text: TRIM END
Button7	Name:btntrim Text: TRIM
Button8	Name: btnsubstart Text: SUBSTRING (START)
Button9	Name:btnsublength Text: SUBSTRING (START LENGTH)
Button10	Name: btnindex Text: INDEXOF

```
Public Class Form1

Private Sub btnup_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles btnup.Click

        lbloutput.Text=txtinput.Text.ToUpper()

End Sub

Private Sub btnlow_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles btnlow.Click

      lbloutput.Text = txtinput.Text.ToLower()

End Sub

Private Sub btnnumeric_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnnumeric.Click

        If IsNumeric(txtinput.Text) Then
            lbloutput.Text = "It's a number"
        Else
            lbloutput.Text = "It's NOT a number"
        End If
End Sub
```

Table 8.2 Plan the Event Procedures

Procedure	Action
Button1	Returns the uppercase equivalent of a string
Button2	Returns the lowercase equivalent of a string
Button3	Accepts a string as its argument and returns True if the string contains a number
Button4	Returns the number of character in a string
Button5	Returns a copy of a string without leading spaces
Button6	Returns a copy of a string without trailing spaces
Button7	Returns a copy of a string without leading or trailing spaces
Button8	Search for the first occurrence of a character or string within a string
Button9	Extracts a string from within a string
Button10	Returns the index of the first occurrence of the specified substring.

```
Private Sub btnlength_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnlength.Click

      lbloutput.Text = txtinput.Text.Length

End Sub

Private Sub btnstart_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btbts.Click

      lblmsgtrim.Text =lblgreeting.Text.TrimStart()

End Sub

Private Sub btntend_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnte.Click

      lblmsgtrim.Text = lblgreeting.Text.TrimEnd()

End Sub

Private Sub btntrim_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnt.Click

      lblmsgtrim.Text = lblgreeting.Text.Trim()

End Sub
```

```
Private Sub btnsub_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnsub.Click

      lblsubout.Text = lblsubin.Text.Substring(2)

End Sub

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnsubsl.Click

      lblsubout.Text =lblsubin.Text.Substring(0, 5)

End Sub

Private Sub btnindex_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnindex.Click

'If search string caps lock or small from original, the output display -1

      lblsubout.Text = lblsubin.Text.IndexOf("Z")

End Sub

End Class
```

```
Public ReadOnly Property testgrade() As Char
        Get
            Dim chrgrade As Char
            If intmark >= 80 Then
                chrgrade = "A"
            ElseIf intmark >= 70 Then
                chrgrade = "B"
            ElseIf intmark >= 60 Then
                chrgrade = "C"
            ElseIf intmark >= 50 Then
                chrgrade = "D"
            Else
                chrgrade = "E"
            End If
            Return chrgrade
        End Get
    End Property
```

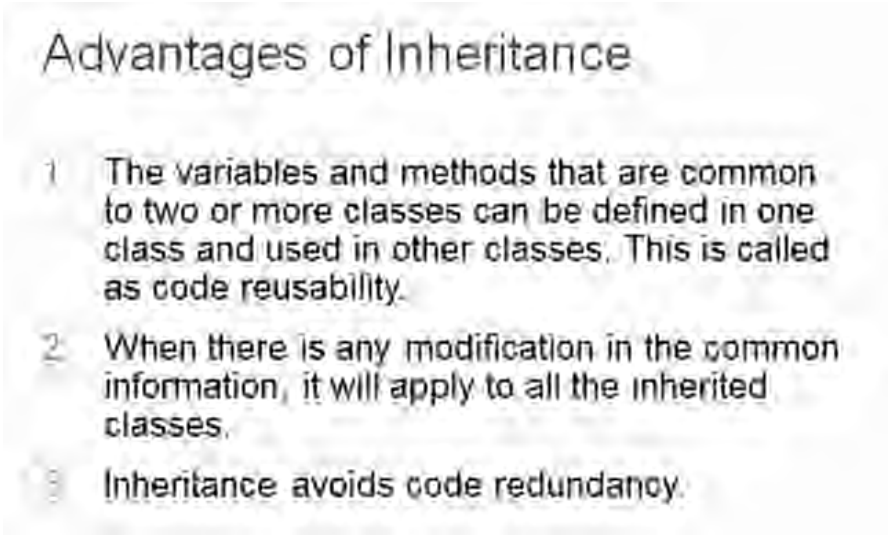
```
Public Class Form1

    Private Sub btnsave_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btnsave.Click

        Dim objstudent As Student

        objstudent = New Student

        GetData(objstudent)

        txtgrade.Text = objstudent.testgrade

    End Sub


    Private Sub GetData(ByVal objstudent As Student)

        objstudent.name = txtname.Text

        objstudent.regnum = txtreg.Text

        objstudent.mark = txtmark.Text

    End Sub


    Private Sub btnexit_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btnexit.Click

        End

    End Sub

End Class
```

9.3 Inheritance

Process by which objects can acquire the properties of objects of other class.

The new class will have combined features of both the classes

In OOP, inheritance provides reusability, like, adding additional features to an existing class without modifying it



9.4 Inheritance Application

Step 1: Design the interface as shown in Figure 9.3. Add two classes and named it as First.vb and Second.vb

Step 2: Set the objects and properties as shown in

Table 9.3.

Step 3: Plan the event procedures as shown in

Table 9.4.

Step 4: Write the code

Step 5: Save and run the project.

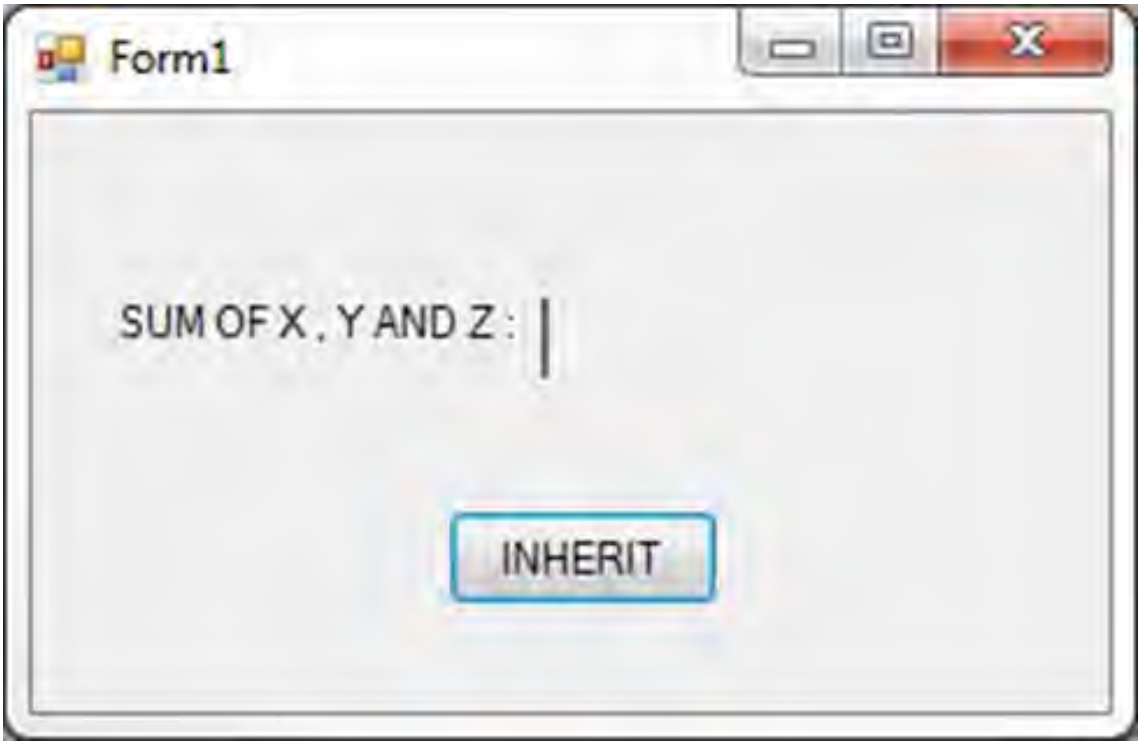


Figure 9.3 Graphical User Interface

Table 9.3 Plan the Objects and Properties

Object	Property
Button1	Name:btninherit Text: INHERIT
Label1	Name: lblsum Text: -

Table 9.4 Plan the Event Procedures

Procedure	Action
btninherit	Total the number

```
Public Class Form1

    Private Sub btninherit_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Button1.Click

        Dim obj As New Second

        obj.X = 100

        obj.Y = 50

        obj.Z = 80

        lblsum.Text = obj.getSum()

    End Sub

End Class
```

```
Public Class First

    Private nilai_x As Integer

    Private nilai_y As Integer

    Public Property X() As Integer

        Get

            Return nilai_x

        End Get

        Set(ByVal value As Integer)

            nilai_x = value

        End Set

    End Property

    Public Property Y() As Integer

        Get

            Return nilai_y

        End Get

        Set(ByVal value As Integer)

            nilai_y = value

        End Set
```

```
Public Class Second
    Inherits First
    Private nilai_z As Integer
    Public Property Z() As Integer
        Get
            Return nilai_z
        End Get
        Set(ByVal value As Integer)
            nilai_z = value
        End Set
    End Property
    Public Function getSum() As Integer
        Dim total As Integer
        total = X + Y + Z
        Return total
    End Function
End Class
```

9.5 Polymorphism

Figure 9.4 Polymorphism

9.6 Polymorphism Application

Step 1: Design the interface as shown in Figure 9.4. Add class and named it Operation.vb.

Step 2: Set the objects and properties as shown in Table 9.5.

Step 3: Plan the event procedures as shown in Table 9.6.

Step 4: Write the code

Step 5: Save and run the project.

Figure 9.5 Graphical User Interface

Table 9.5 Plan the Objects and Properties

Object	Property
Label1	Name: lblresult1 Text: -
Label2	Name: lblresult2 Text: -
Label3	Name: lblresult3 Text: -
Button1	Name: btnresult Text: RESULT

Table 9.6 Plan the Event Procedures

Procedure	Action
btnresult	View the result

```
Public Class Form1

Private Sub btnresult_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnresult.Click

      Dim objpoly As New Operation()

            lblresult1.Text= "Value for i+k: " &objpoly.add(10, 30)

            lblresult2.Text= "Value for i+j: " &objpoly.add(10.1, 20.5)

            lblresult3.Text= "Value for i+j+k: "&objpoly.add(10, 20, 30)

    End Sub

End Class
```

```
Public Class Operation

    Public i, j, k As Integer

    Public Function add(ByVal i As Integer, ByVal k As Integer) As
Integer

        Return i + k

    End Function

    Public Function add(ByVal i As Double, ByVal j As Double) As Double

        Return i + j

    End Function

    Public Function add(ByVal i As Integer, ByVal j As Integer, ByVal k
As Integer) As Integer

        Return i + j + k

    End Function

End Class
```

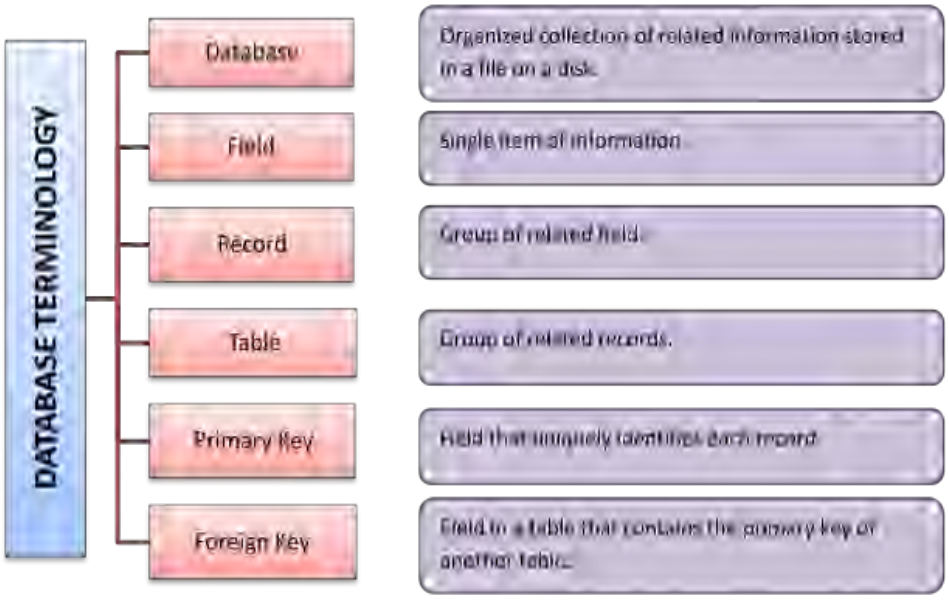
CHAPTER 10

DATABASE

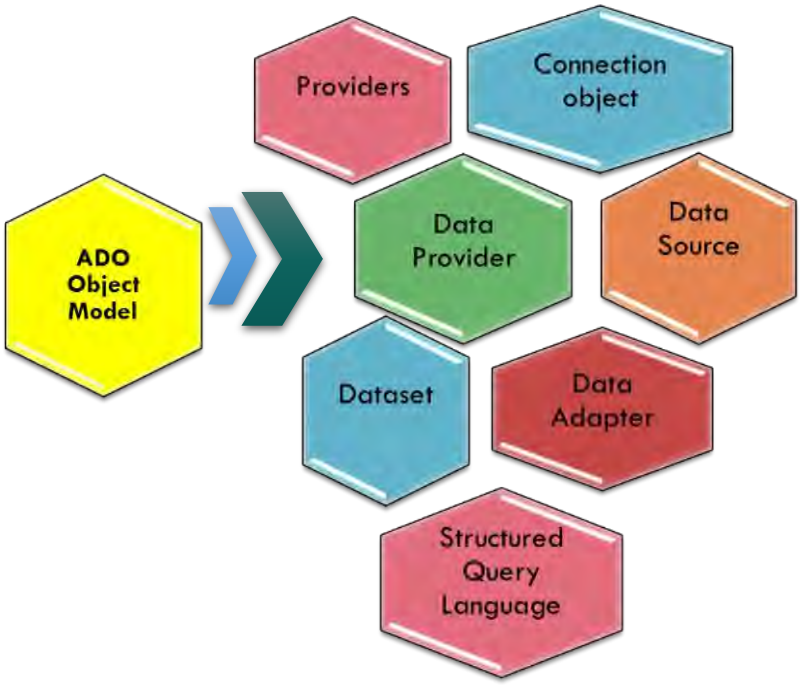
At the completion of this chapter, you will able to:

- Define relational database
- Define database terminology
- Describe ADO.NET
- Create a database program

10.1 Database Terminology



10.2 ADO Object Model



- **Providers** are objects that retrieve data from a database.
- The ***Connection* object** contains information about the database, such as the server name, the name of the database, and the user name.
- **Data Provider** objects use the *Connection* object to connect to the database.
- **Data Source:** the original source/ location of database data;

- **Dataset**: copy of the fields and records stored in the computer's internal memory, which the application can access
- The **DataAdapter** uses a provider which translates the request into a language the database understands
- **Structured Query Language (SQL)**: A set of commands that allows you to access and manipulate the data stored in many database management systems on computers of all sizes

10.3 Data Access Application

Step 1: Design the interface as shown in Figure 10.1

Step 2: Set the objects and properties as shown in

Table 10.1.

Step 3: Plan the event procedures as shown in

Table 10.2.

Step 4: Write the code

Step 5: Save and run the project.

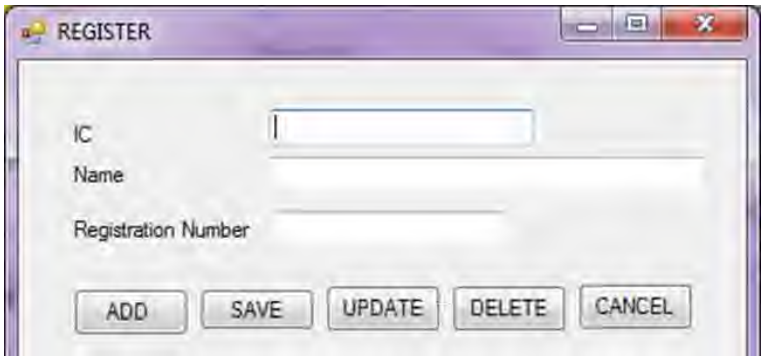


Table 10.1 Plan the Objects and Properties

Object	Property
Button1	Name:btnadd Text: ADD
Button2	Name: btnsave Text: SAVE
Button3	Name: btnupdate Text: UPDATE
Button4	Name: btndelete Text: DELETE
Button5	Name: btncancel Text: CANCEL

TextBox1	Name:txtic Text: -
TextBox2	Name:txtName Text: -
TextBox3	Name:txtReg Text: -

Table 10.2 Plan the Event Procedures

Procedure	Action
btnadd	Add new data
btnsave	Save the data
btnupdate	Update the data
btndelete	Delete the data
btncancel	Clear all input from textbox

```
Imports System.Data.OleDb

Public Class frmRegister

    Dim inc As Integer

    Dim MaxRows As Integer

    Dim con As New OleDb.OleDbConnection

    Dim dbProvider As String

    Dim dbSource As String

    Dim ds As New DataSet

    Dim da As OleDb.OleDbDataAdapter

    Dim sql As String


Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load

     dbProvider = "PROVIDER=Microsoft.Ace.OLEDB.12.0;"

     dbSource = "Data Source = J:\VISUAL BASIC.NET\MY

     CODES\DbaseRegistration_ACCDB _USE\Daftar.accdb"

     con.ConnectionString = dbProvider & dbSource

     con.Open()

     sql = "SELECT * FROM Pelajar"

     da = New OleDb.OleDbDataAdapter(sql, con)

     da.Fill(ds, "Daftar")

     con.Close()

     MaxRows = ds.Tables("Daftar").Rows.Count

     inc = -1

End Sub
```

```
Private Sub btnadd_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btnadd.Click

        txtic.Clear()

        txtName.Clear()

        txtReg.Clear()

        txtic.Focus()

        inc = 0

End Sub

Private Sub btnupdate_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles btnupdate.Click

Dim cb As New OleDb.OleDbCommandBuilder(da)

ds.Tables("Daftar").Rows(inc).Item(0) = txtic.Text

ds.Tables("Daftar").Rows(inc).Item(1) = txtName.Text

ds.Tables("Daftar").Rows(inc).Item(2) = txtReg.Text

       da.Update(ds, "Daftar")

       MsgBox("Data updated successfully")


End Sub
```

```
Private Sub btnsave_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btncommit.Click

        If inc <> -1 Then

Dim cb As New OleDb.OleDbCommandBuilder(da)

Dim dsNewRow As DataRow

  dsNewRow ds.Tables("Daftar").NewRow()

  dsNewRow.Item("IC") = txtic.Text

  dsNewRow.Item("FirstName") = txtName.Text

  dsNewRow.Item("RegNum") = txtReg.Text

        ds.Tables("Daftar").Rows.Add(dsNewRow)

da.Update(ds, "Daftar")

 MsgBox("New Record added to the database")

       txtName.Clear()

       txtReg.Clear()

       txtic.Clear()

        End If

End Sub
```

CHAPTER

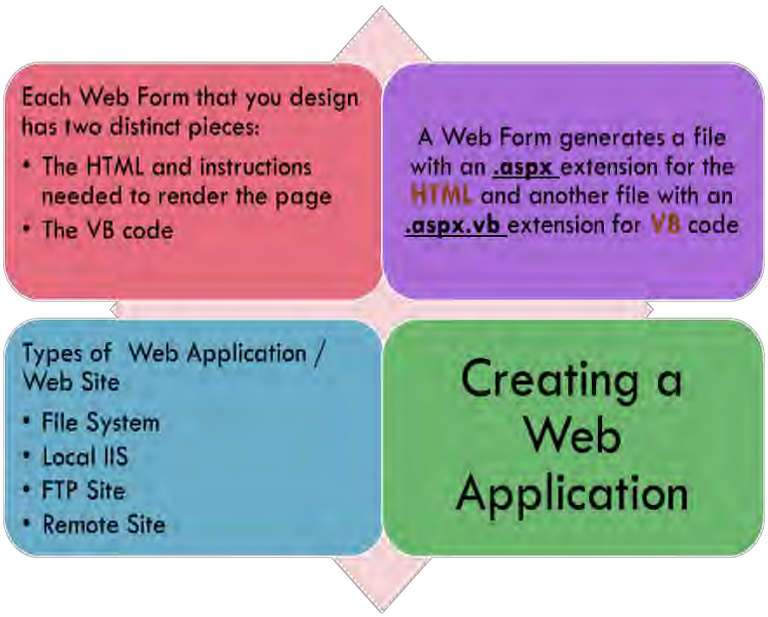
11

WEBPAGE

At the completion of this chapter, you will able to:

- Illustrate the relationship between client and server.
- Create a dynamic web page using VB.NET.

11.1 Create A Web Page



11.2 Create A File System Web Site Project

Open Microsoft Visual Studio. On the File menu, select New Web Site. The New Web Site dialog box appears as shown in Figure 11.1.

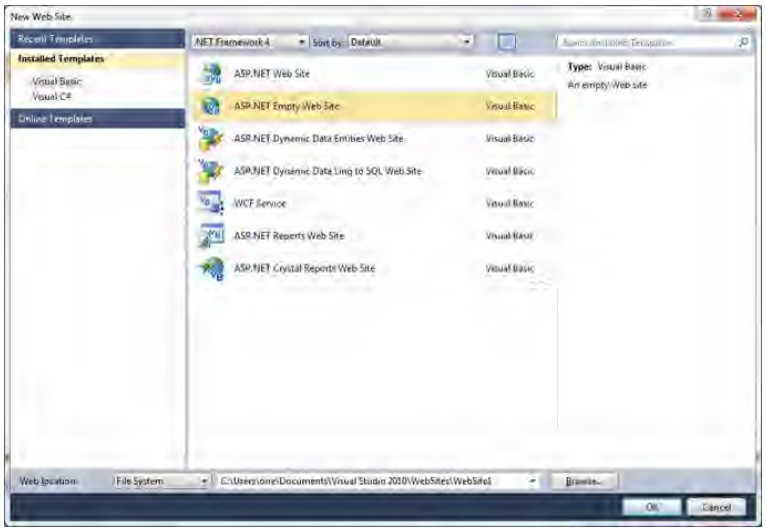


Figure 11.1 New Web Site dialog box

Select ASP.NET Empty Web Site and in the Web Location box, select File System, and then enter the name of the folder where you want to keep the pages of your Web site. Click OK.

11.2.1 Add a page to the Web site

On the Website menu, click Add New Item as shown in Figure 11.2. In the template list, select Web Form. In the Name box, index.aspx. Then click Add.

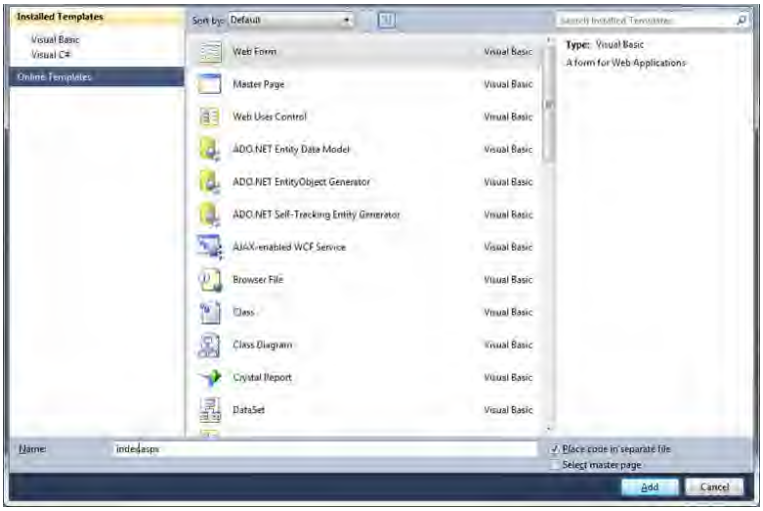


Figure 11.2 Add new Item

At the bottom of the document window, click the Design tab to switch to Design view. Design view displays the page that you are working as shown in Figure 11.3. Click inside the rectangle that is outlined by a dashed line.

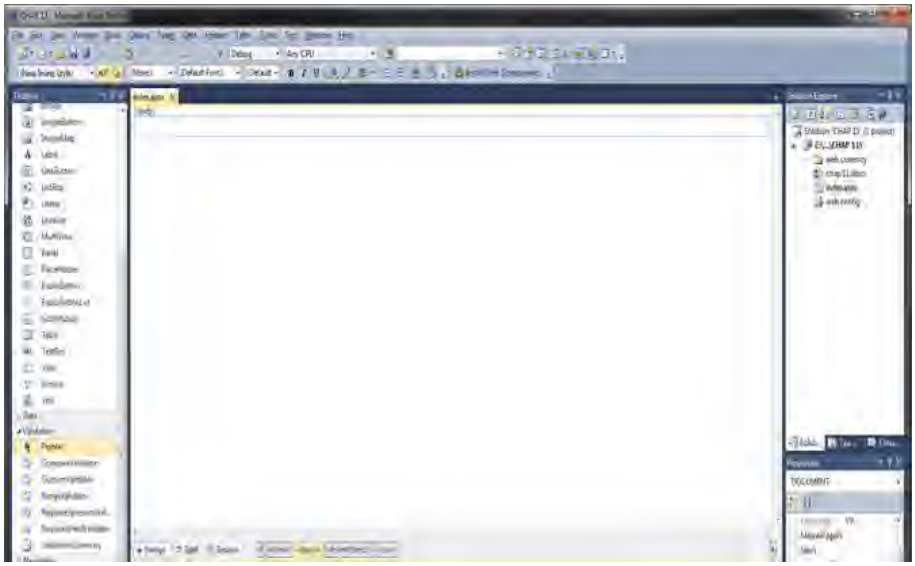


Figure 11.3 index.aspx Web Form

Design the web form as shown in Figure 11.4

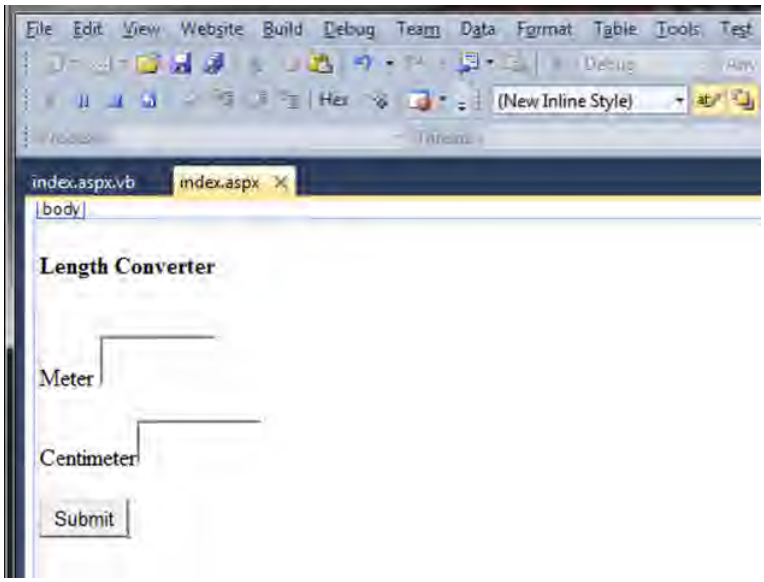


Figure 11.4 Graphical User Interface

Table 11.1 Plan the Objects and Properties

Object	Property
Label1	Text: Length Converter ID: lbllength
Label2	Text: Meter ID: lblmeter
Label3	Text:Centimeter ID: lblcm
TextBox1	Text: (blank) ID: txtmeter
TextBox2	Text: (blank) ID: txtcm
Button	Text: Submit ID: btnsubmit

Table 11.2 Plan the Event Procedures

Procedure	Action
btnsubmit	Convert meter value to cm value

11.3 Validation Controls

Validation controls are used to validate the data of an input control or a validator is a control that checks one input control for a specific type of error condition and displays a description of that problem. These are the following validation controls as shown in Table11.3.

```
Protected Sub btnsubmit_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnsubmit.Click
        Const rate As Double = 100
        txtcm.Text = txtmeter.Text * rate
End Sub
```

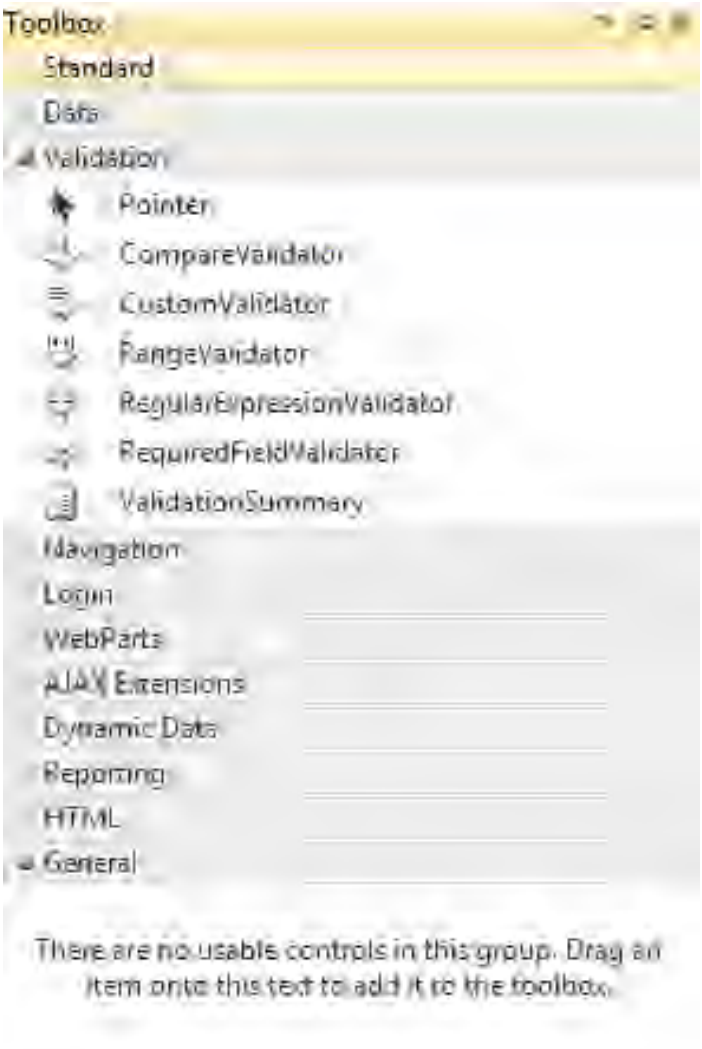


Figure 11.5 Validation Control Toolbox

Table 11.3 Validation Control

Validation Control	Description
RequiredFieldValidator	To ensures that the user does not skip or blank an entry of the control.
Compare Validator	Used to compare the entry value of one control to another control using comparison operator (less than, equal to, greater than and others).
Range Validator	Checks that a user's entry is between specified lower and upper boundaries. You can check ranges within pairs of numbers, alphabetic characters, or dates. Boundaries can be expressed as constants.
RegularExpression Validator	Checks that the entry matches a pattern defined by a regular expression. This type of validation allows you to check for predictable sequences of characters, such as those in social security numbers, e-mail addresses, telephone numbers, postal codes, and so on.
CustomValidator	Checks the user's entry using validation logic that you code yourself. This type of validation allows you to check for values derived at run time.
ValidationSummary	Displays the validation errors in summary form for all of the validators on a page.

11.3.1 Apply Validation Control In A Web Form

Add RequiredFieldValidator to a web form as shown in Figure 11.6. Set the properties as shown in Table11.4.

Figure 11.6 Required Field Validator Control

Table 11.4 RequiredFieldValidator Properties

Property	Set
ErrorMessage	Please enter the number
ControlToValidate	Choose txtmeter

Run the page and do not insert value in txtmeter. The error message will appear as shown in Figure 11.7.

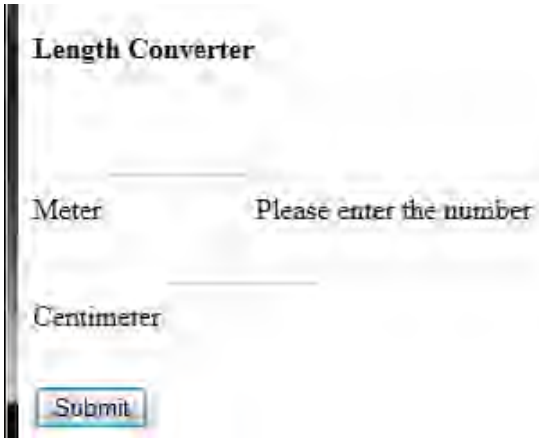


Figure 11.7 RequiredFieldValidator Output

Add RangeValidator to a web form. Set the properties as shown in Table11.5.

Table 11.5 RangeValidator Properties

Property	Set
ErrorMessage	Please insert number 1 – 1000 only
ControlToValidate	Choose txtmeter
MaximumValue	1000
MinimumValue	1

Run the page and insert 0 or 1001 in txtmeter. The error message will appear as shown in Figure 11.8 and Figure 11.9. It is because we have set the maximum and minimum value.

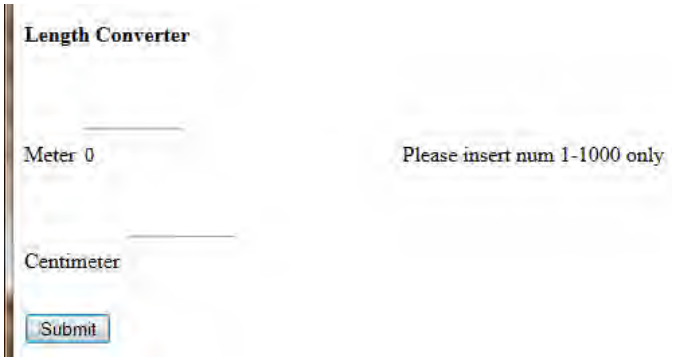


Figure 11.8 Minimum value

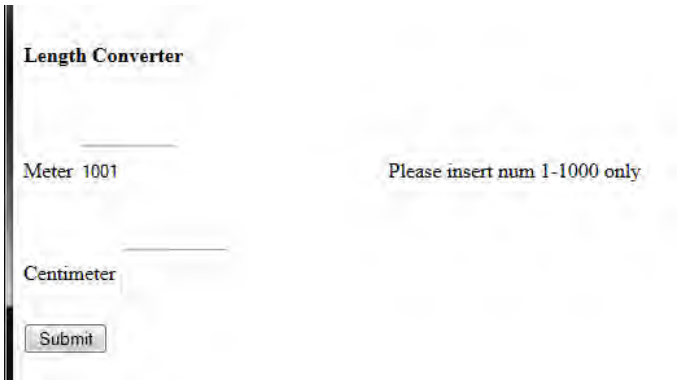


Figure 11.8 Maximum value

EXERCICES

1. Choose the **CORRECT** abbreviation for IDE.

A. Integrated Drive Environment

B. Integrated Desktop Environment

C. Intuitive Development Environment

D. Integrated Development Environment

2. Choose the **CORRECT** generation of .Net for Visual Studio 2010 tool.

A. 3.0

B. 3.5

C. 4.0

D. 4.5

3. Select the window that can be used to write code.

A. Immediate Window

B. Locals Window

C. Code Editor Window

D. Solution Explorer Window

4. Which of the following statement declares the operator symbol, operands, and code that define an operator procedure on a class or structure?

A. Sub

B. Declare

C. Operator

D. Property

5. Determine the debugger in VB.Net.

A. A computer program enabling the user to enter or alter text.

B. A computer program that converts instructions into a machine-code.

C. A computer program that can analyze and execute a program line by line.

D. A computer program that assists in the detection and correction of errors in other computer programs.

6. Select the default data type for Visual Basic.

A. Variant

B. Integer

C. Boolean

D. String

7. Choose the **CORRECT** name for this object 

.

A. Combo box

B. List box

C. Check box

D. Text box

8. Choose the **CORRECT** statement to terminate all forms.

A. `End`

B. `Me.Close ( )`

C. `Me.Exit ( )`

D. `Application.Close ( )`

9. Select which is **NOT** a main component of the Visual Studio IDE.

A. Tool Box

B. Start Menu

C. Solution Explorer

D. Properties Window

10. Calculate the value for Profit using the following expression

```
Profit = 100 * (1 + 1.0) ^ 2
```

A. 200

B. 400

C. 20000

D. 40000

11. Choose the **CORRECT** output that will generate when the codes in Figure 1 is executed and the value of TempInteger is 50.

```
If TempInteger > 30 Then
     If TempInteger > 80 Then
          CommentLabel.Text = "Hot"
     Else
          CommentLabel.Text =
     "Moderate"
     End If
Else
     CommentLabel.Text = "Freezing"
```

Figure 1

A. `Hot`

B. `Moderate`

C. `Freezing`

D. No output

12. Choose the **CORRECT** code to display the message box in Figure 2.

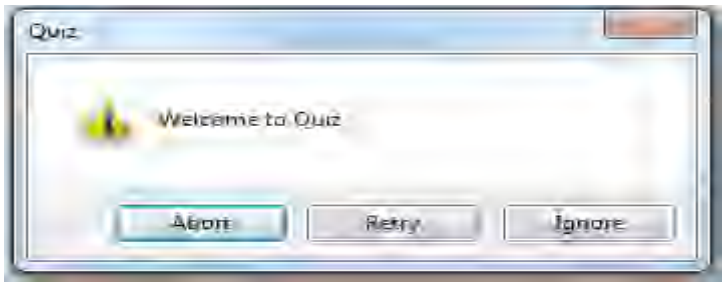


Figure 2

A.
```
MessageBox.Show ("Welcome to Quiz", "Quiz", _
MessageBoxButtons.AbortRetryIgnore, MessageBoxIcon.Question)
```

B.
```
MessageBox.Show ("Welcome to Quiz", "Quiz", _
MessageBoxButtons.RetryCancel, MessageBoxIcon.Exclamation)
```

C.
```
MessageBox.Show ("Quiz", "Welcome to Quiz”, _
MessageBoxButtons.AbortRetryIgnore,
MessageBoxIcon.Exclamation)
```

D. `MessageBox.Show ("Welcome to Quiz", "Quiz", _`
`MessageBoxButtons.AbortRetryIgnore,`
`MessageBoxIcon.Exclamation)`

13. Choose the **CORRECT** code to display a window in Figure 3.

Figure 3

A. `Msgbox ("Nama Saya Ialah")`
B. `Msgbox ("Nama Saya Ialah", "name")`
C. `Inputbox ("Nama Saya Ialah")`
D. `InputBox ("Name"," Nama Saya Ialah")`

14. Choose the **CORRECT** numeric data type.

A. Boolean
B. Char
C. Double
D. String

15. Select the suitable operator for integer division.

A. +
B. /
C. \
D. ^

16. Name the **INCORRECT** type of variable in Visual Basic.Net.

A. Case
B. Dim
C. Private
D. Public

17. Choose the **CORRECT** feature for constant.

A. It is one of the data types.

B. Its value can be changes by user.

C. Its name declaration is without any value or data.

D. Its data cannot be change during the project execution.

18. Identify the property used to modify the color of the text displayed in a control.

A. `TextColor`

B. `BackColor`

C. `LetterColor`

D. `ForeColor`

19. Select the format to change the number from 40550.90 into RM40,550.90.

A. `ToString ("P2")`

B. `ToString ("C")`

C. `ToString ("C4")`

D. `ToString ("N")`

20. Convert the variable that store a value of 3000.505 to a numeric data type.

A. `Const sum As Byte = 3000.505`

B. `Const total As String = 3000.505`

C. `Dim sum As Char = 3000.505`

D. `Dim total As Decimal = 3000.505`

21. Choose the **CORRECT** code to convert the string "500" into number 500.

A. `numResult = Val ("500")`

B. `numResult = Asc ("500")`

C. `numResult = Oct ("500")`

D. `numResult = Hex ("500")`

22. Choose the **CORRECT** statement to assign the value into a variable.

A. `myIC = txtIC.Text`

B. `my&Name = "MOHD"`

C. `myNumber = 10 to 100`

D. `2myAddress = "No 12, Taman ABC"`

23. Choose the **CORRECT** statement for the blank space ***W*** and ***X*** in Figure 4.

```
If   intTotal > 50        W
     MsgBox ("PASSED")
  X
     MsgBox ("FAILED")
End If
```

Figure 4

A. ***W*** : Next ***X*** : Else

B. ***W*** : Then ***X*** : ElseIf

C. ***W*** : Then ***X*** : Else

D. ***W*** : Then ***X*** : End

24. Transfer the Select … Case statement in Figure 5 to an IF … Then … Else statement.

```
Select Case menu
           Case "1"
           frmPersonal.Show()

       Case "2"
           frmAbout.Show()

       Case Else
           MsgBox("Wrong number")
End Select
```

Figure 5

A.

```
If menu = "1" Then
     frmPersonal.Show ( )
Else menu = "2" Then
     frmAbout.Show ( )
Else Then
     MsgBox ("Wrong Number")
End If
```

B.

```
ElseIf menu = "1" Then
      frmPersonal.Show ( )
ElseIf menu = "2" Then
      frmAbout.Show ( )
ElseIf
      MsgBox ("Wrong Number")
End
```

C.

```
If menu = "1" Then
      frmPersonal.Show ( )
ElseIf menu = "2" Then
      frmAbout.Show ( )
Else
      MsgBox ("Wrong Number")
End If
```

D.

```
If menu = "1"
      frmPersonal.Show ( )
ElseIf menu = "2"
      frmAbout.Show ( )
Else
      MsgBox ("Wrong Number")
End If
```

25. Select the looping control structure to display message box for 5 times.

A.

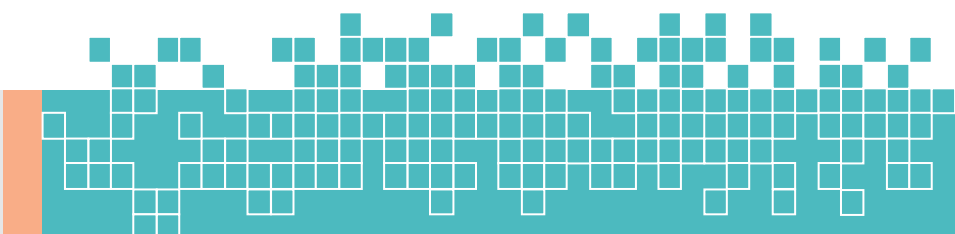
```
intCount = 0
Do While intCount < 6
      intCount = intCount + 1
            MessageBox.Show ("Good Luck")
Loop
```

B.

```
intLoop = 0
MessageBox.Show ("Good Luck")

Do
      intLoop = intLoop + 1
      MessageBox.Show ("Good Luck")
Loop Until intLoop = 5
```

C.

```
For i = 0 To 5
      MessageBox.Show ("Good Luck")
Next
```

D.

```
For i = 0 To 4
     MessageBox.Show ("Good Luck")
Next
```

26. Find the answer for TOTAL in Figure 6 by referring to the expression below.

```
lblTotal.Text = txtBag.Text + txtShoes.Text
```

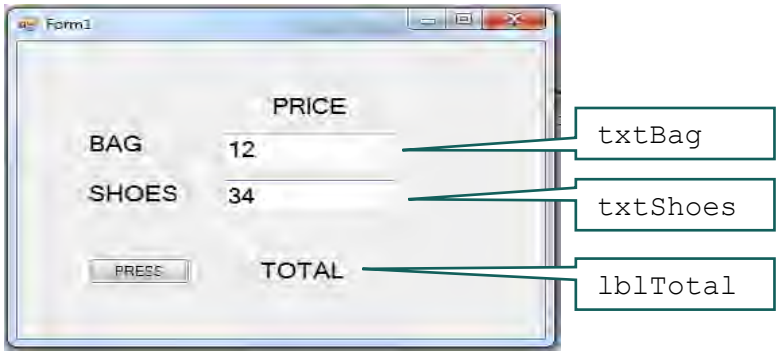


Figure 6

A. 46

B. 46.00

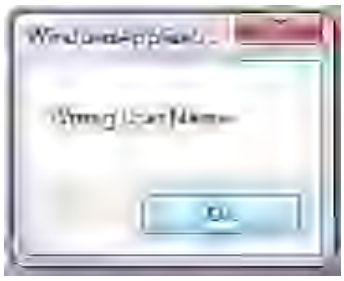
C. 3412

D. 1234

27. Figure 7 is the block of code for btnLogin. Determine the output after end user filled the `txtUserName.Text = "manan"` and `txtPassword.Text = "manan123"`.

```
Dim strUsername As String = "manan@gmail.com"
Dim strPassword As String = "manan123"
If txtUserName.Text <> strUsername And
txtPassword.Text = strPassword Then
    MsgBox("Wrong User Name")
ElseIf txtUserName.Text = strUsername And
txtPassword.Text<>strPassword Then
    MsgBox("Wrong Password")
ElseIf txtUserName.Text <> strUsername And
txtPassword.Text<>strPassword Then
    MsgBox("Both are wrong!")
Else
MsgBox("You are successful to login")
End If
```

A. 

B. 

C. 

D. 

28. Find the errors found in the VB.Net statement in Figure 8.

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

      Dim strContacts(4) As String

      For k = 0 To 4
            strContacts(k) = InputBox("Enter the name: ")

      For i = 0 To 4
            MessageBox.Show("The value for index" & i & " is " & _
            strContacts(i))

      Next
End Sub
```

Figure 8

A. 1. Looping statement was not completed.
   2. Identifier expected.

B. 1. Variable was not declared.
   2. Missing datatype.

C. 1. There is missing of End For.
   2. Wrong datatype for strContacts.

D. 1. Problem at message box statement.
   2. Wrong coding for InputBox.

29. Determine the output of the code in Figure 9 when the user entered "minis" into `textBox1` and "try" into `textBox2` in Figure 10.

```
Private Sub btnTekan_Click(ByVal sender As System.Object, ByVal e As_
System.EventArgs) Handles btnTekan.Click

      Dim strWord1 As String = TextBox1.Text
      Dim strWord2 As String = TextBox2.Text

      MessageBox.Show (displayWord(strWord1, strWord2))

End Sub

Function displayWord(ByRef s1 As String, ByVal s2 As String) As _
String
      Dim resultDisplay As String

      resultDisplay = s1.ToUpper & s2.ToUpper

      Return resultDisplay
```

Figure 9

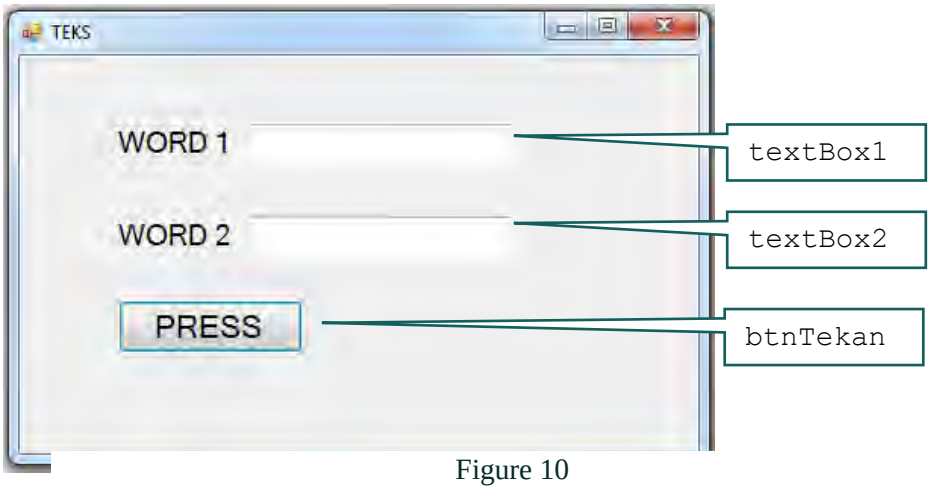


Figure 10

A. MINIS TRY
B. MINISTRY
C. ministry
D. minis try

30. Analyze the output that will be generate when the codes in Figure 11 executed.

```
Dim intVal1 As Integer = 30
Dim intVal2 As Integer = 20
Dim intVal3 As Integer = intVal1 + intVal2
Dim intVal4 As Integer = (intVal3 - 3) Mod 3

If intVal4 > 70 Then
    MsgBox("OLD")

ElseIf intVal4 > 40 And intVal4 <= 70 Then
    MsgBox("MEDIUM")

ElseIf intVal4 > 0 And intVal4 <= 40 Then
    MsgBox("YOUNG")

Else
    MsgBox("OUT OF RANGE")

End If
```

A. 

B. 

C. 

D. 

31. Choose the **CORRECT** property to change the prefix name of object.

A. Font

B. Text

C. Name

D. Tag

32. Choose the **CORRECT** property to change the form title into “Borang Aduan” in Figure 12.

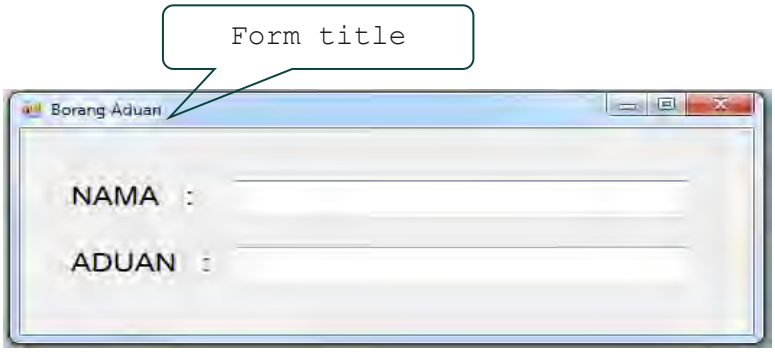


Figure 12

A. Text

B. Name

C. Label

D. Font

33. Apply the **CORRECT** function statement in Figure 13 to get the output in Figure 14.

```
Private Sub btnCalculate_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles btnCalculate.Click

    Dim strYes As String

    strYes = MessageBox.Show("DO YOU WANT CALCULATE??", -
    "Calculate", MessageBoxButtons.YesNo, MessageBoxIcon.Question)

    If strYes = vbYes Then
          MsgBox(calculate(10, 40))
    End If
End Sub
```

Figure 13

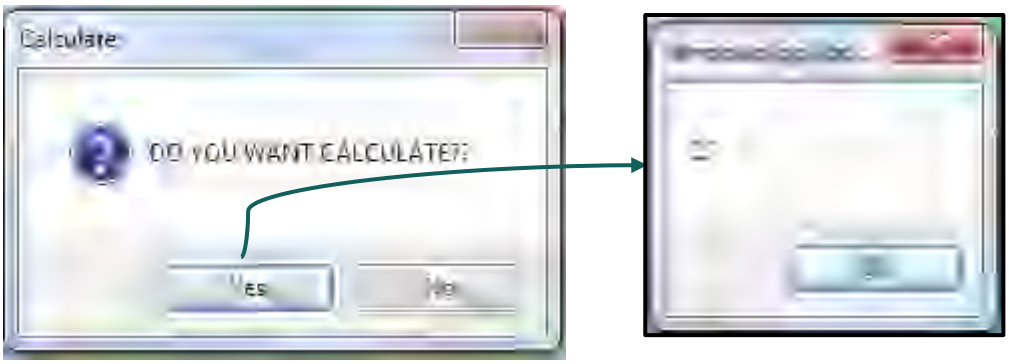


Figure 14

```
Sub Function calculate(ByVal m, ByRef n)
        Dim average As Double
        average = (m + n) / 2
End Function
```

A.

```
Function calculate(ByVal m, ByRef n) As Double
        Dim average As Double
        average = (m + n) / 2
End Sub
```

B.

```
Sub calculate(ByVal m, ByRef n)
        Dim average As integer
        average = (m + n) / 2
        Return average
End Function
```

C.

```
Function calculate(ByVal m, ByRef n) As Double
        Dim average As Double
        average = (m + n) / 2
        Return average
End Function
```

34. Determine how many times the input box will be displayed when the code in Figure 15 is executed.

```
count = 15
While count > 10
    InputBox("My Favourite Number:")
    count = count - 1
End While
```

Figure 15

A. 3
B. 4
C. 5
D. 6

35. Determine how many times the message box will be displayed when the code in Figure 16 is executed.

```
For intNum = 0 To 10 Step 3
    MsgBox("HAPPY BIRTHDAY")
Next
```

Figure 16

A. 3

B. 4

C. 10

D. 11

Figure 17

36. Based on Figure 17, choose the **CORRECT** statement to add an item if a user wants to add name "MUHD" into the list box called `lstName`.

A. `lstName.Items.Text="MUHD"`

B. `lstName.Items.Add("MUHD")`

C. `lstName.Add ("MUHD")`

D. `lstName.Add = "MUHD"`

37. Choose the one-dimension array declaration named Marks with 10 elements.

A. `Private Marks [10] As Double`

B. `Private Marks (10) As Double`

C. `Dim Marks [9] As Double`

D. `Dim Marks (9) As Double`

38. Determine the **CORRECT** declaration of two dimension with 2 columns and 3 rows for `Total` arrays.

A. `Dim Total [2,3] As Double`

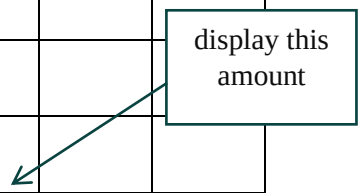
B. `Dim Total [2,3] As Double = {{50,60,45} , {55,59,47}}`

C. `Dim Total (2,3) As Double`

D. `Dim Total ( , ) As Double = {{50,60,45} , {55,59,47}}`

Table 1 : Array of Students monthly saving (dblSaving)

Monthly / Students	0	…	5	6	7	…
0						
…						
11			110	90	100	
12			95	85	96	
13			70	93	98	
…						



39. Choose the **CORRECT** syntax to display the amount as indicated by the arrow in Table 1.

A. `MessageBox.Show (dblSaving (6 , 11))`

B. `dblSaving (6 , 11) = 90`

C. `MsgBox (dblSaving [6 , 11] )`

D. `lblDisplay.Text (dblSaving (6 , 11))`

40. Declare the new object for class Vehicles.

A. `Dim New veh As Vehicles`

B. `Dim Vehicles As New veh`

C. `Dim veh As New Vehicles`

D. `veh As New Vehicles`

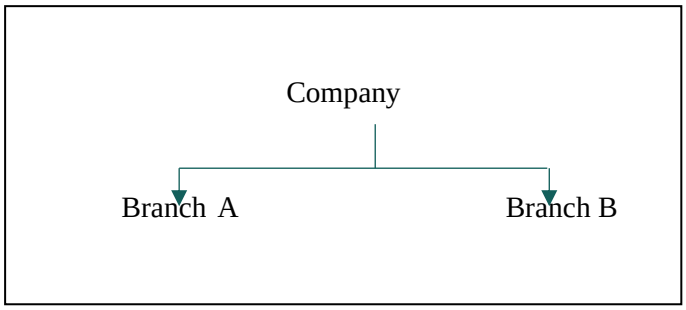


Figure 18

41. Company is a super class while Branch A and Branch B are sub classes in Figure 18. Choose the **CORRECT** statement of inheritance.

A.

```
Public Class Branch B
      Inheritance's Company
End Public
```

B.

```
Public Class Company
      Inherits Branch B
End Public
```

C.

```
Public Class Branch A
      Inherits Company
End Public
```

D.

```
Inherits Class Branch A
      Class Company
End Inherits
```

42. Based on coding in Figure 19, class Person had been created. Choose the **CORRECT** statement to call the `Display ( )` method.

```
Public Class Person
      Private name As String
      Private address As String

      Sub Display ( )
            MessageBox.Show ("Hello my friend")
      End Sub
End Class
```

Figure 19

A.

```
Sub Main ( )
      Dim friend As New Person (  )
      friend.Display (  )
End Sub
```

B.

```
Sub Main ( )
      Dim Person As New staff (  )
      Person.Display (  )
End Sub
```

C.

```
Sub Main ( )
      Dim friend As New Person (  )
      Display (  )
End Sub
```

D.

```
Sub Main ( )
      Dim friend As New Person (  )
      friend.Person.Display (  )
 End Sub
```

43. Choose the suitable validation control for text box to fill an e-mail address.

A. RequiredField validation
B. RegularExpression validation
C. Range validation
D. Comparison validation

44. User must fill the name in the web form. Select the type of validation control that programmer should set in the property.

A. RequiredField validation
B. RegularExpression validation
C. Custom validation
D. Range validation

45. Choose the **CORRECT** validation control that programmer will do his own condition to validate the user field text box.

A. Range validation
B. Validation summary
C. Custom validation
D. Comparison validation

46. Choose the **CORRECT** statement of string manipulation to display the index **7** when the variable `strDep` is declared in Figure 20.

```
Dim strDep As String = "JTMK DEPARTMENT"
```

Figure 20

A.

```
strDep = strMsg.Trim("PART")
MsgBox(strDep)
```

B.

```
strDep = strMsg.OfIndex("PART")
MsgBox(strDep)
```

C.

```
strDep = strMsg.IndexOf("PART")
MsgBox(strDep)
```

D.

```
strDep = strDep.Index("PART")
MsgBox(strDep)
```

47. The `strBanner` variable contains the following text **"WELCOME BACK".** Choose the statement that will display **"WEL"** only.

A.

```
strBanner = strBanner.Remove("COME BACK")
MsgBox(strBanner)
```

B.

```
strBanner = strBanner.Trim("COME")
MsgBox(strBanner)
```

C.

```
strBanner = strBanner.Replace("COME")
MsgBox(strBanner)
```

D.

```
strBanner = strBanner.Remove(3)
MsgBox(strBanner
```

48. The `strMalaysia` variable contains the following string "**MY COUNTRY IS MALAYSIA**". Choose the **CORRECT** statement to display the output is "**COUNT**".

A. `strMalaysia = srtMalaysia.Display(3, 5)`

B. `strMalaysia = srtMalaysia.ToString(3, 5)`

C. `strMalaysia = srtMalaysia.TrimStart`

D. `strMalaysia = srtMalaysia.Substring(3, 5)`

49. Choose **CORRECT** output for the statement in Figure 21.

```
Dim hstay1 As String = "HOMESTAY NOR"
     Dim hstay2 As String = hstay1.Insert(11, "MBO")
MessageBox.Show (hstay2)
```

Figure 21

A. HOMESTAY NOMBOR

B. HOMESTAY NORMBO

C. HOMESTAY NMBOOR

D. HOMESTAY NOR

50. Choose the best answer to complete the exception handling in Figure 22.

```
Sub Main( )
     Dim no1 As Integer = 0
          Dim no2 As Double
     Try
             y = 100 / x
      (i)  ex As ArithmeticException
             MessageBox.Show(ex.Message)
             MsgBox(y)
      (ii)
End Sub
```

Figure 22

A. i Catch, ii. End

B. i. Catch, ii. End Try

C. i. Catch, ii. Finally

D. i. Catch, ii. Lastly

REMERCIEMENTS

Alfred C. Thompson II (2018). Visual Basic. Net Fundamentals. CreateSpace Independent Publishing Platform (ISBN: 9781723535482)

Chris Goode, John Kauffman, etc all, (2018) Beginning ASP.NET 1.0 with Visual Basic.NET, John Wiley & Sons. (ISBN :9780764558665)

Conrod. P. & Tylee. L. (2019). Learn Visual Basic 2019 Edition: A Step-By-Step Programming Tutorial 16th Edition 2019. Washington, U.S: Kidware Software. (ISBN: 1951077105)

Goode. C., Kauffman, J. & Miller, C. L. (2018) Beginning ASP.NET 1.0 with Visual basic.NET, John Wiley & Sons. (ISBN: 978-0-764-55866-5)

Hoisington. C. (2017). Microsoft Visual Basic 2017 for Windows, Web and Database Applications: Comprehensive, Cengage Learning. (ISBN: 9781337102117)

Kumar. G. (2019). Learn Visual Basic .Net Programming: A Practical Approach Kindle Edition. Kindle. (ASIN: B07MYMDR5X) Additional:

LEARN

VISUAL BASIC.NET

Provides students with the knowledge and skills needed to develop applications in Microsoft Visual Basic .NET for the Microsoft .NET platform. The course focuses on user interfaces programming structure, language syntax, and integration of VB.NET application development. This course introduces computer programming using the VB Programming language with object-oriented programming principles. Emphasis is on event-driven programming methods, including creating and manipulating objects, classes, and using object oriented tools such as the class debugger.