Introduction to Windows controls

1

Lecture Contents:

- Windows applications – basic concepts
  - OOP
  - Event driven Programming
- Form
- Controls – label, edit box, button
- Procedures - Event handlers
- Visual Studio IDE

2

Object Oriented Programming

- **Object-oriented programming** (OOP) is a programming paradigm that uses abstraction to create models based on the real world. It permits creation of entities known as UDT /User Defined data Types/ that contain data components and procedures to operate on these data components.
- Typical characteristics of OOP:
  - modularity,
  - data encapsulation (data hiding)
  - inheritance,
  - polymorphism.

3

Object Oriented Programming

- **Windows**
- **Forms**
- **Controls (labels, textboxes, listboxes, checkboxes, buttons, radio buttons, etc)**

are objects (also called instances) that belong to specific abstract user defined data types, named classes.

Analogy:	Class	>>	Object
	Basic data type	>>	Variable

4

Event Driven Programming

- **Event Driven Programming** is a style of computer programming where the flow of the program is determined by user actions (mouse clicks, key presses) or messages from other programs.
- In contrast, in **batch programming** the flow is determined by the programmer. Batch programming is the style taught in beginning programming classes while event driven programming is what is needed in any interactive program.

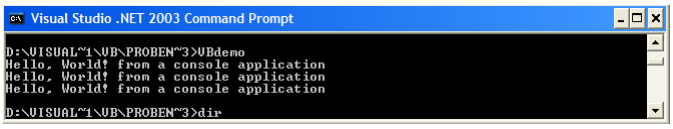
5

Windows Applications

6

Console applications

- A console application runs in a console window (or DOS box) and provides simple text-based output.



7

Windows applications

- A Windows application runs on a PC's desktop.
- Windows applications are more complex than console applications and take advantage of the GUI /Graphical User Interface/.
- VB programs are also known as applications, solutions, or projects. Each program is saved in its own folder.

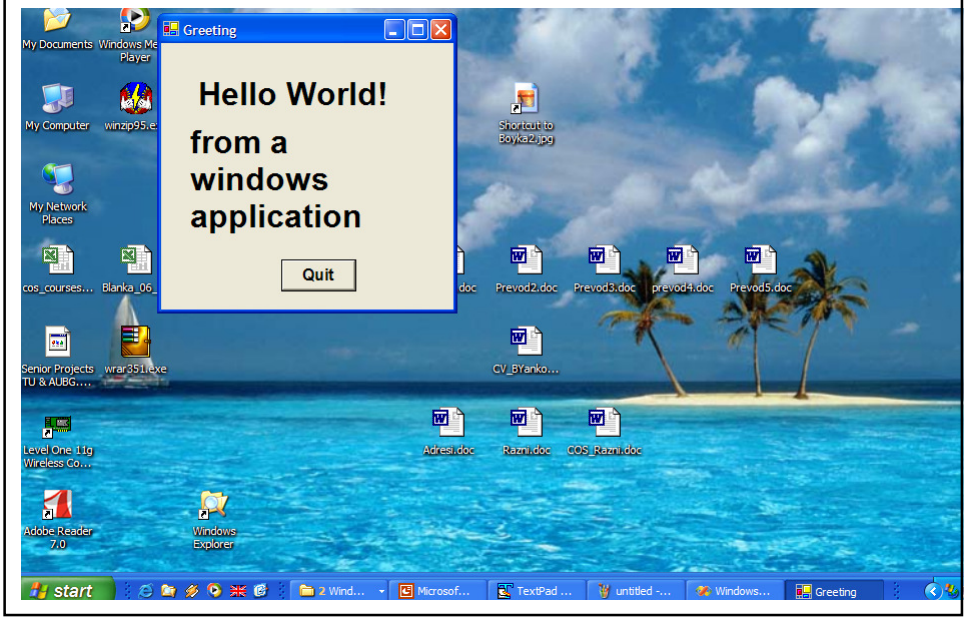
8

Windows applications

- VB programs display a Windows-style screen (called a form). Each form contains:
  - boxes (Text box, List box, Combo box, etc) into which users type and edit information.
  - buttons that users click to initiate actions.
  - labels located near a text box to tell the user what type of information is displayed in the text box.
- Boxes, buttons and labels are referred to as controls.

9

Desktop and Windows application



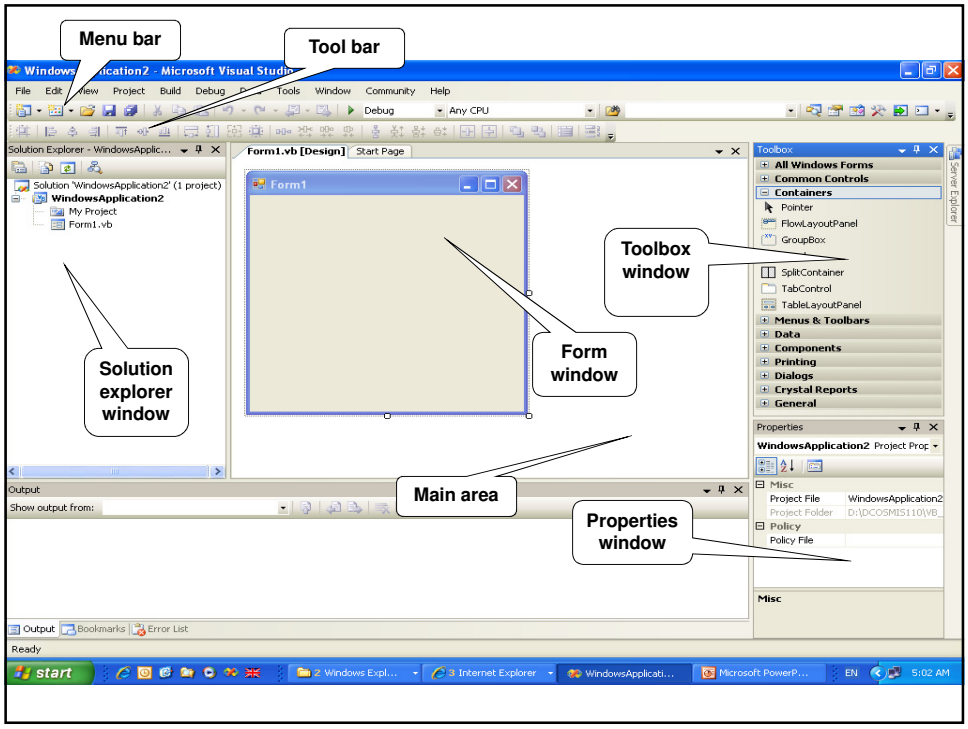
Windows application: form, label, button

11

Windows applications

For example, in *Visual Basic .NET*, selecting the Windows Application option automatically creates the windows shown on the next slide.

12

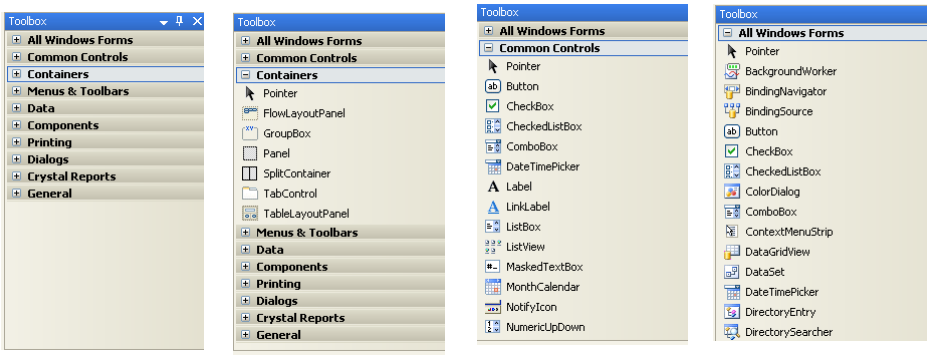


The *Integrated Development Environment (IDE)* for *VB.NET* displays following main windows:

- the **Main area window** currently holds the **Windows Form Designer.** The rectangular **Form window**, or **form** for short, becomes a Windows window when a program is executed.
- the **Solution Explorer window** is used to display various parts of a program.
- the **Properties window** is used to change how objects look and react.
- the **Toolbox window** holds icons representing controls that can be placed on the form.

14

The *ToolBox* consists of a collection of control objects or controls for short that can be created by clicking and dragging.



15

Some of the most often used controls available from The *ToolBox:*

Text boxes: You use a text box to get information from the user, referred to as input, or to display information produced by the program, referred to as output.

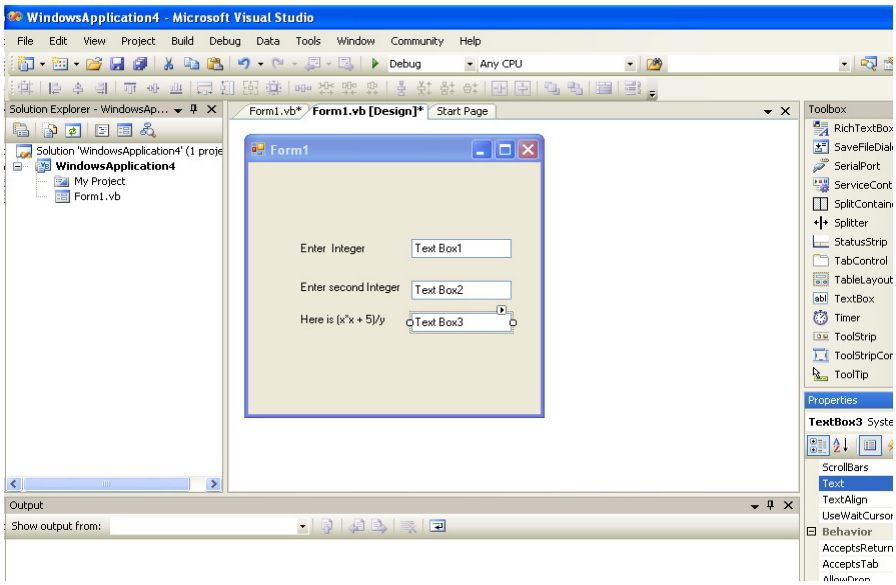
Labels: You place a label near a text box to tell the user what type of information is displayed in the text box.

Buttons: the user clicks a button to initiate an action.

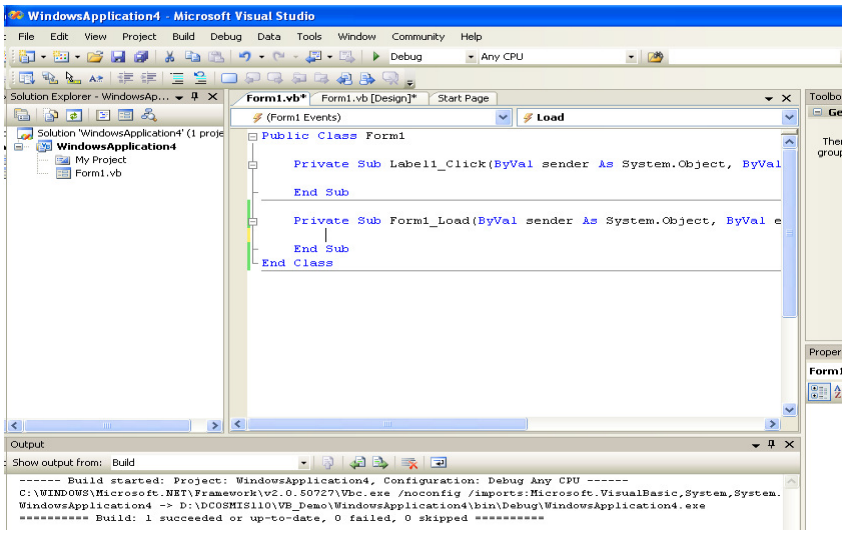
List boxes: You use a list box to display tables or several lines of text, or to make a selection.

16

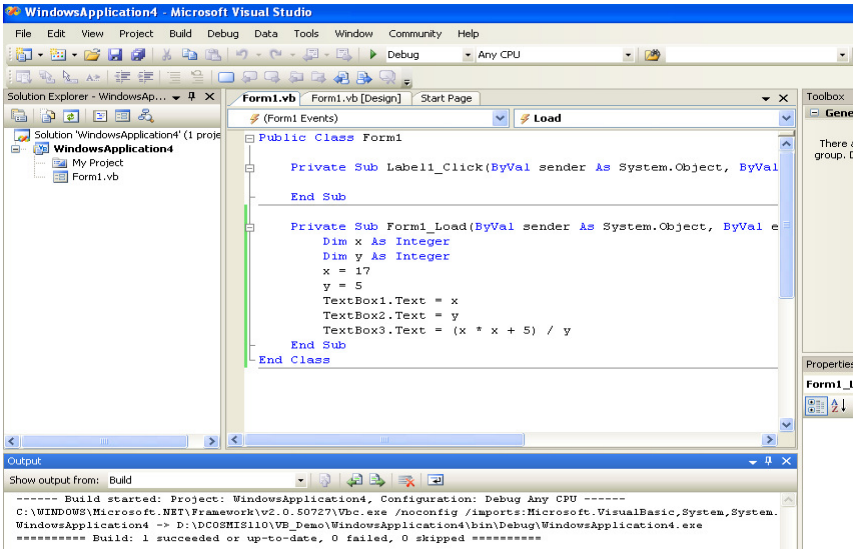
In the following, three labels and three textboxes have been created by using the *ToolBox*



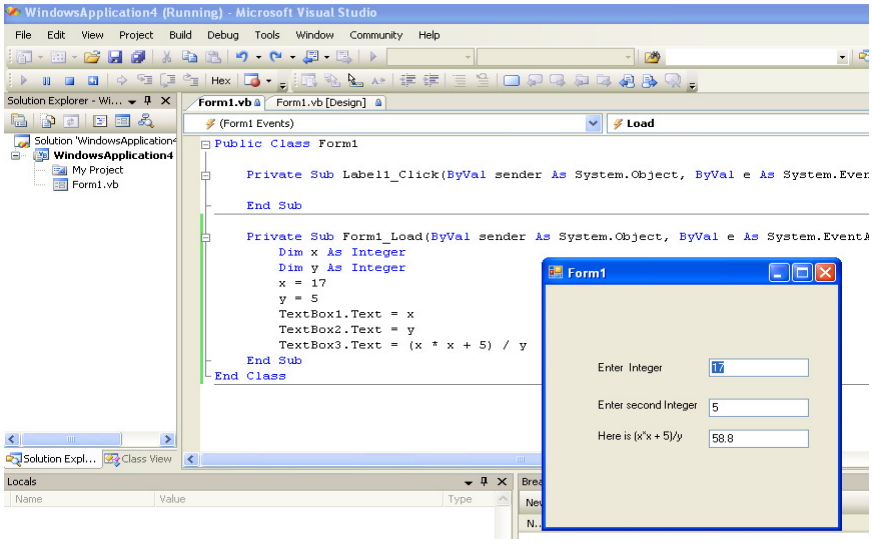
Having created a visual interface by clicking and dragging, the appropriate *Visual Basic code* is also automatically inserted into your program.



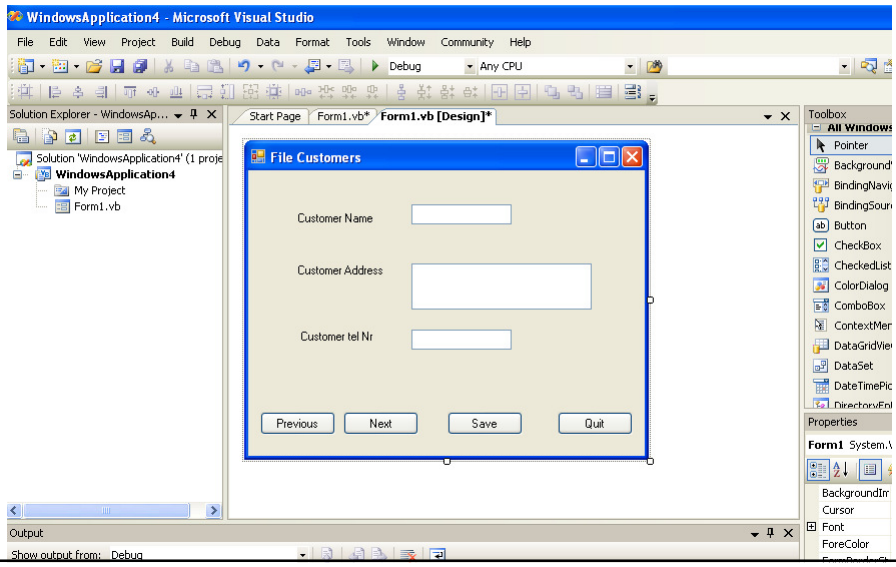
Having created a visual interface by clicking and dragging, the appropriate *Visual Basic code* is also automatically inserted into your program.



Having created a visual interface by clicking and dragging, the appropriate *Visual Basic code* is also automatically inserted into your program.



Another example of a Form design: three Labels, three blank Text Boxes and four Buttons



Visual Basic **is completely graphically (visually) oriented.**

Visual Basic **allows you to create applications using the concepts of** ***Graphical User Interface (GUI)*****.**

It has already constructed all the necessary building blocks that an application needs like windows, command buttons, list boxes and option buttons.

This means that there is much less coding needed to be done and an impressive ***Visual Basic*** **application can be created very quickly.**

22

Visual Basic **is an event-driven language**

- the code is written to respond to specific *events*, e.g. a user creates an event by clicking on a button, pressing a key, choosing an item in a list box or closing a window, etc.

In *Basic*, a program is executed without regard to events.

In *Visual Basic*, due to its windows interface, the users may click on a certain object randomly, so each object has to be programmed independently to be able to respond to those actions (events).

Therefore, a *Visual Basic* program is made up of many "subprograms", each has its own program code, and each can be executed independently, and at the same time each can be linked together in one way or another. 23

I-P-O Illustrated

- **Demo1: I-P-O model of computing process using a console application**
- Demo2: I-P-O model of computing process using a Windows application

24

Reminder: a program for a company office manager.

The company mails letters comprising a different number of sheets of paper.

The more sheets, the more stamps are required for the letter

Problem:

- How many stamps do you use when mailing a letter?
- "rule of thumb": Use one stamp for every five sheets of paper or fraction thereof.

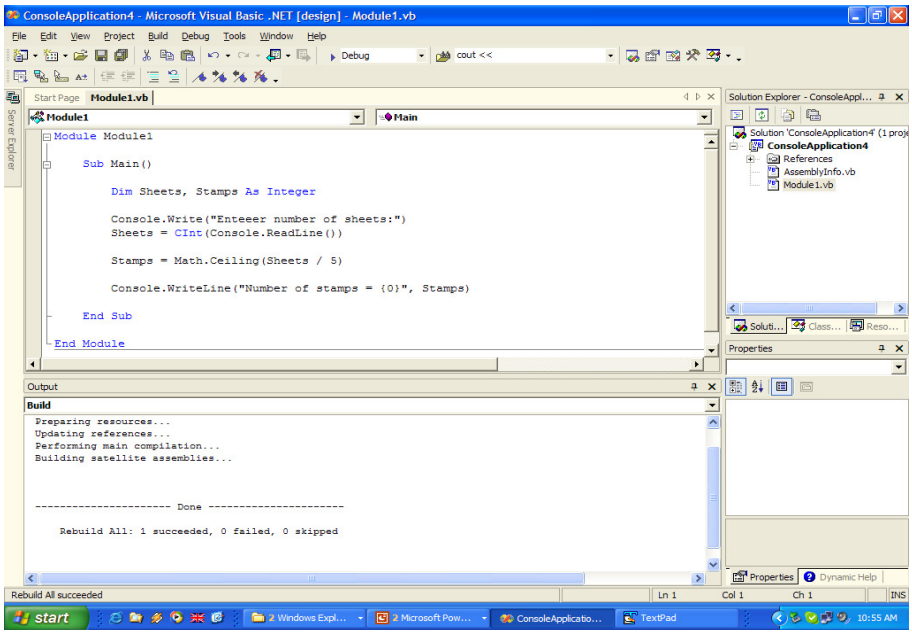
25

The Source text as a console application

```
Imports System
Module Module1

   Sub Main()

      Dim Sheets, Stamps As Integer

      Console.Write("Enter number of sheets:")
      Sheets = CInt( Console.ReadLine() )

      Stamps = Math.Ceiling(Sheets / 5)

      Console.WriteLine("Number of stamps = {0}", Stamps)

   End Sub

End Module
```

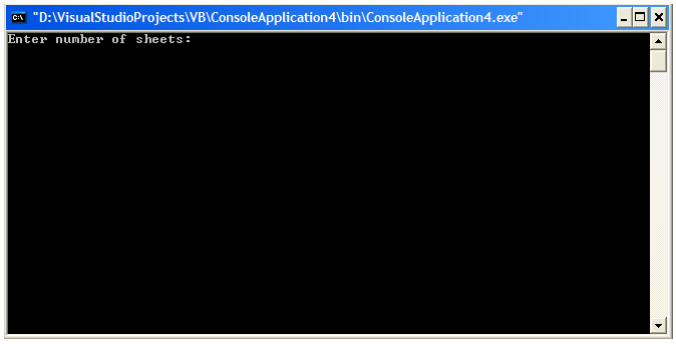
26

Compile–time: Compilation process takes place



Run–time: Execution takes place: INPUT

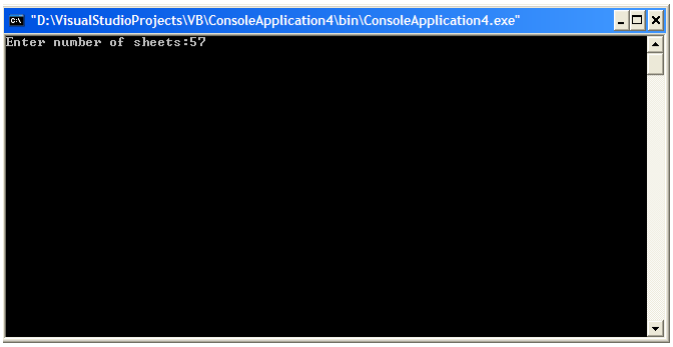
Console.Write("Enter number of sheets:")



28

Run–time: Execution takes place: INPUT

Sheets = CInt(Console.ReadLine())



29

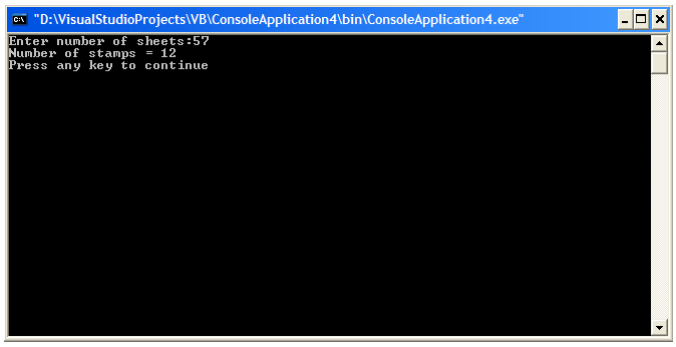
Run–time: Execution takes place: PROCESS

Stamps = Math.Ceiling(Sheets / 5)

30

Run–time: Execution takes place: OUTPUT

Console.WriteLine("Number of stamps = {0}", Stamps)



31

I-P-O Illustrated

- Demo1: I-P-O model of computing process using a console application
- **Demo2: I-P-O model of computing process using a Windows application**

32

No Compile-time, No Run–time: Design takes place:

Application includes:
1 Form

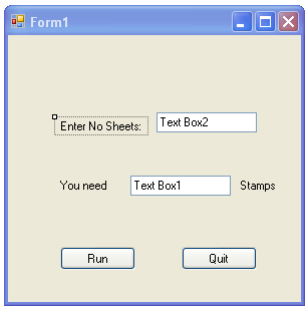
Form includes:
3 labels:
"Enter no of Sheets:"
"You need"
"Stamps"

2 text boxes:
1 for input
1 for output

2 buttons:
Run
Quit

33

No Compile-time, No Run–time: Design takes place:



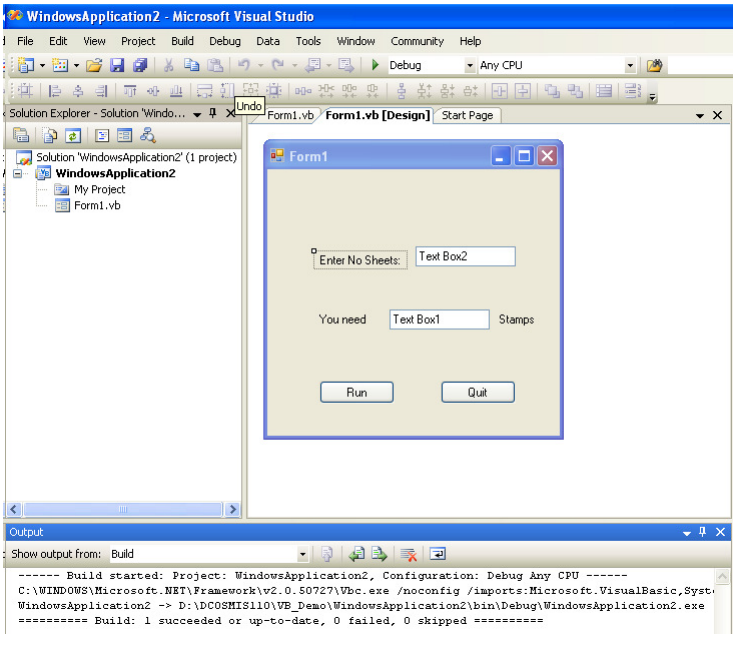
34

Source text concentrated in procedure button Run

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles Button1.Click

        Dim Sheets, Stamps As Integer

        Sheets = CInt(TextBox1.Text)

        Stamps = Math.Ceiling(Sheets / 5)       'Stamps = Sheets \ 5

        TextBox2.Text = CStr(Stamps)

    End Sub
```

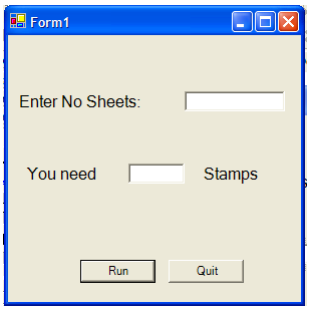
35

Compile–time: Compilation process takes place



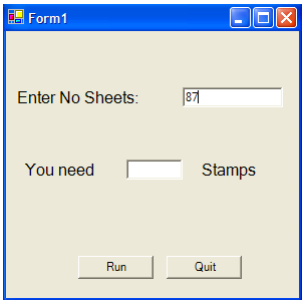
36

Run–time: Execution takes place:



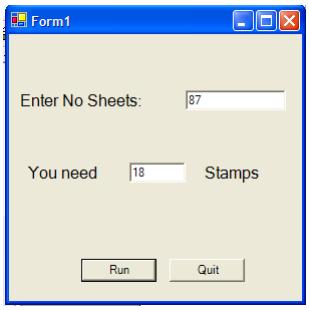
37

Run–time: Execution takes place:



38

Run–time: Execution takes place:



39

Windows Applications

40

VB.NET displays a Window-style screen (called a *form*) with:

- **boxes** into which users type information and/or in which users edit information;
- **buttons** that users click to initiate actions.

The boxes and buttons are referred to as *controls*.

The most common controls are:

- Text boxes
- Labels
- Buttons
- List boxes

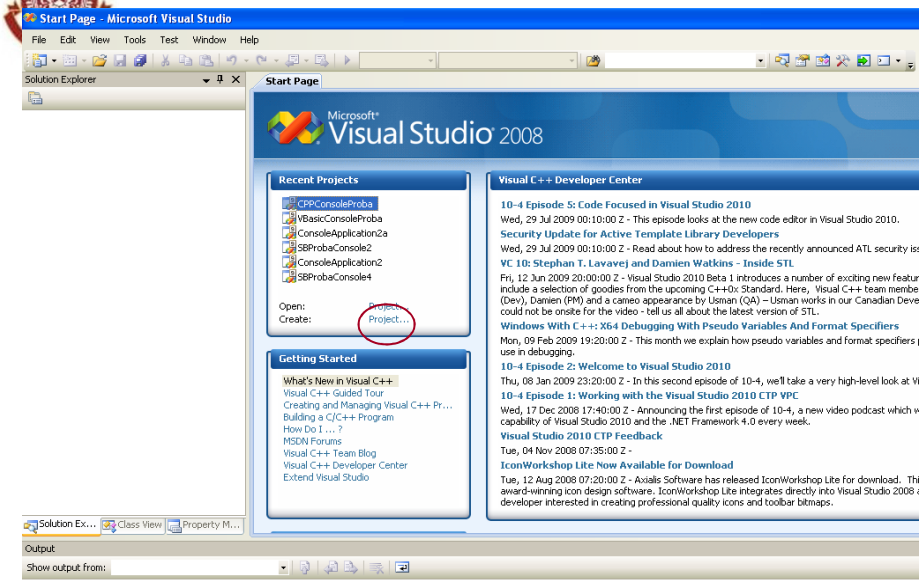
41

The three steps in creating a VB.NET program

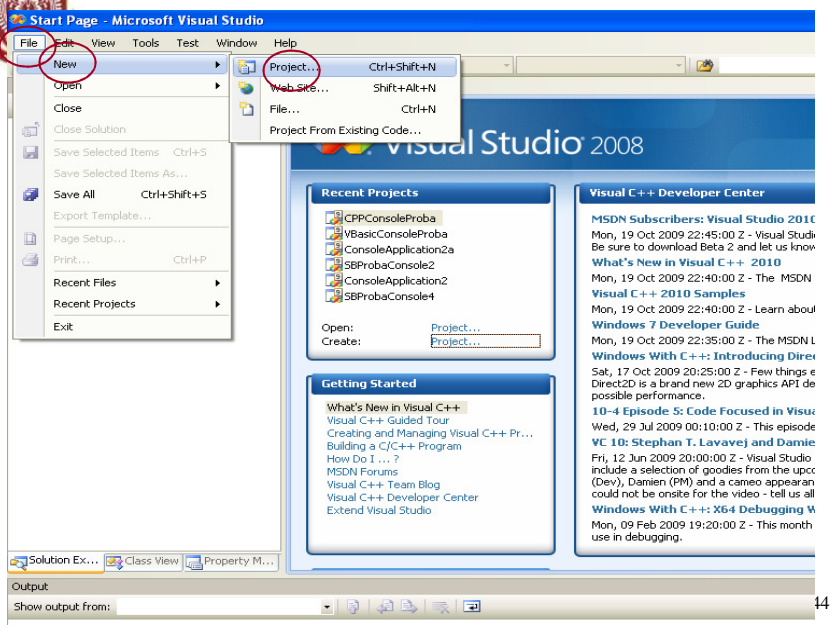
1. Create the interface; that is, generate, position, and size the objects.
2. Set properties; that is, configure the appearance of the objects.
3. Write the code that executes when events occur.

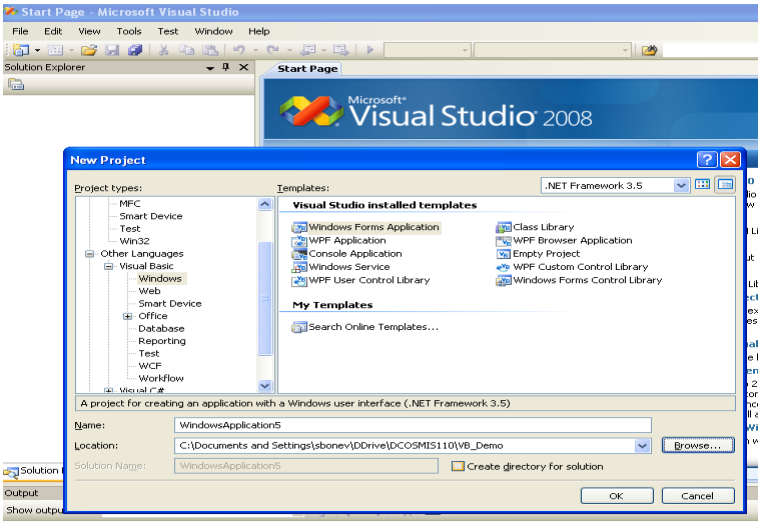
42

From the *Visual Studio Start Page*, click *Create Project*



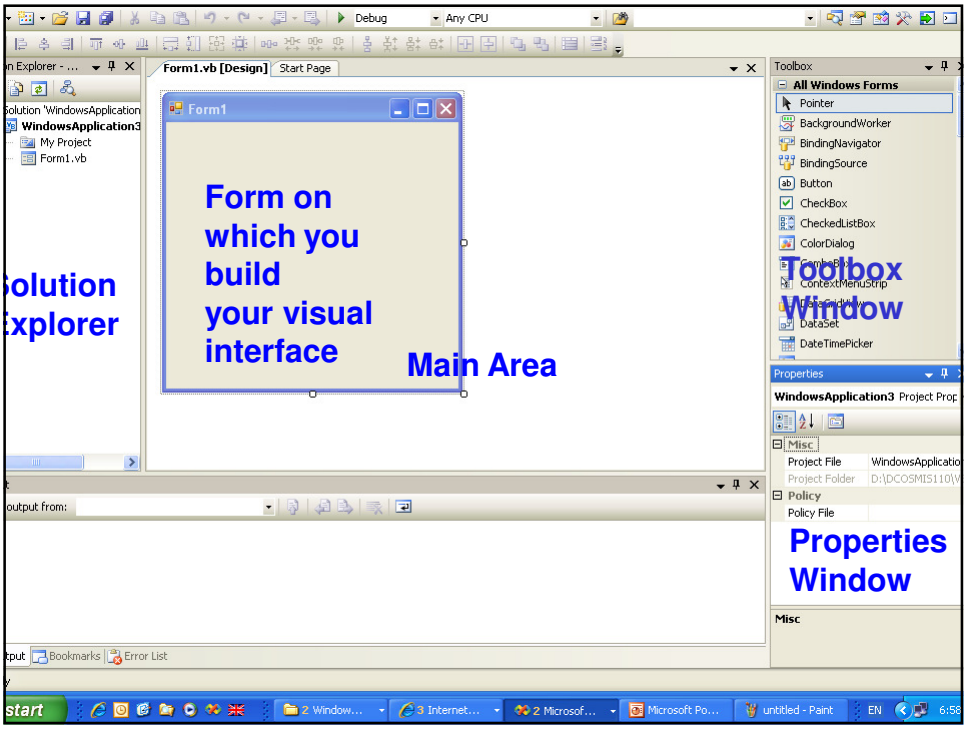
OR click *File > New > Project…*





Choose (I.e. click on) Project type: *VisualBasic > Windows* and Project Template: *Windows Forms Application*, then click OK. Also choose suitable *Name*, *Location* and *Solution Name*.

45



The *Form* is your "canvas" – here you build the visual interface that your program will present to users

- The interface comprises various *controls* that are required by your particular program.

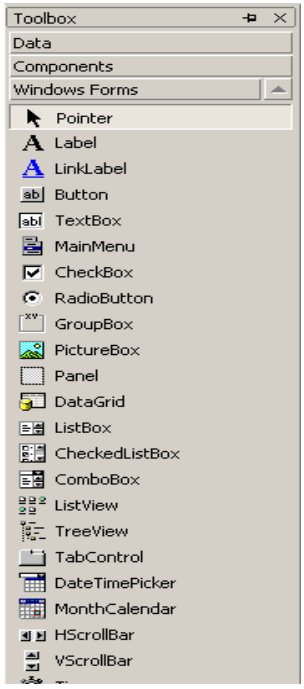
- These controls permit the user to:

- input data to the program (textboxes);
- receive back the results generated by the program and displayed on the screen in order to be examined by the user (textboxes, listboxes);
- to process specific activities (buttons);
- to display text within the form (labels).

47

The *Toolbox* contains a list of controls (icons).

Using the Toolbox, you may add controls to your "user interface", i.e. you may develop a screen for your program.

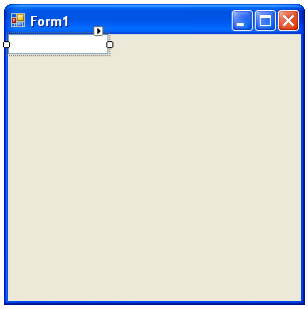


48

The TextBox control

49

Double click on the *Text Box* icon in the *Toolbox (IDE 2005/08)*



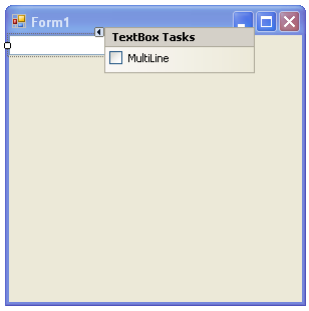
A text box appears in the top left part of the Form – by default it has the name "TextBox1" and is blank.

Drag (with the left mouse button) the text box to any part of the Form

The two small squares are known as *sizing handles* and they allow you to resize the length of the text box

50

Double click on the *Text Box* icon in the *Toolbox*



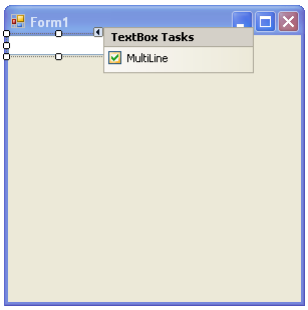
If you click the top-right square filled with triangle mark, you can modify the TextBox tasks, selecting the check box in order to transform the selected control

from SingleLine TextBox
to MultiLine TextBox

See next slide.

51

Double click on the *Text Box* icon in the *Toolbox*

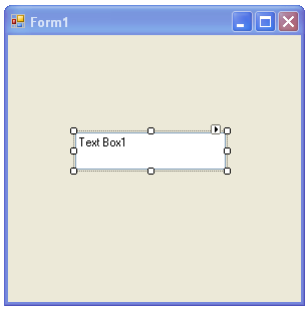


A text box appears in the top left part of the Form – by default it has the name "TextBox1"

Drag (with the left mouse button) the text box to any part of the Form

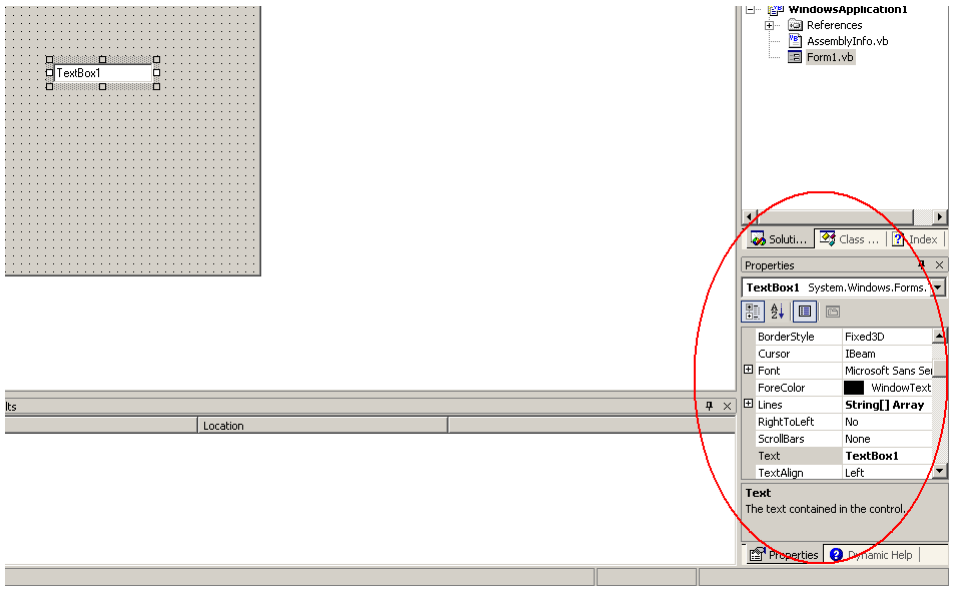
The eight small squares are known as *sizing handles* and they allow you to resize the length&width of the text box

52



When a control is "selected" (i.e. its handles are showing), its "properties" (how the controls looks and reacts) are listed in the Properties window (bottom right window of screen)

53



54

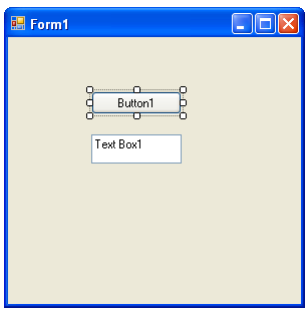
The Button control

55

Now add a button to the Form by double clicking on the Button icon in the Toolbox.

By default, it has the name "Button1"

Drag it above the text box



56

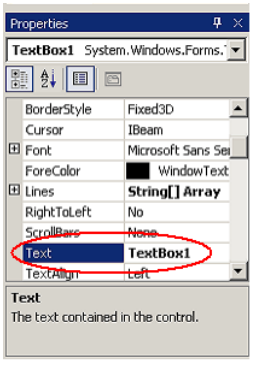
Other controls may be added to the Form in a similar way.

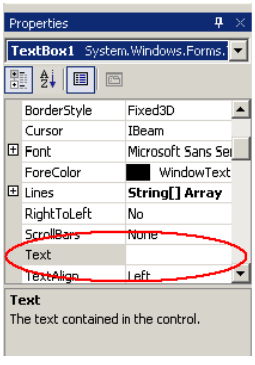
We will now develop a small program so that when the button is clicked, a message will appear in the text box.

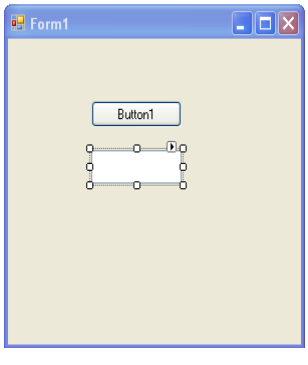
57

Click on the text box to select it.

In the Properties window, delete the Text string "Textbox1"



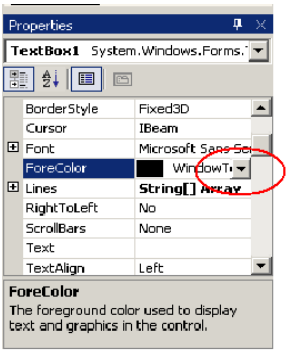


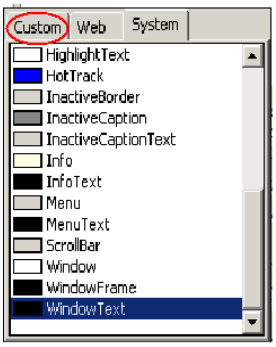


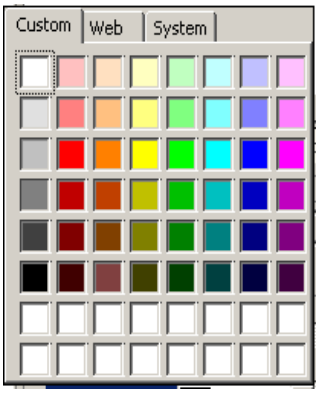
The text box will now be blank.

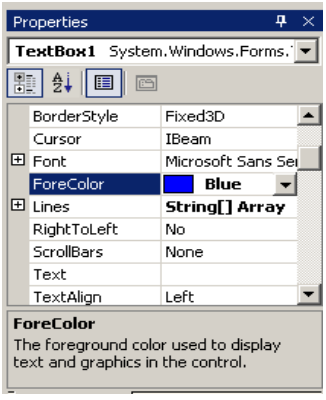
58

In the *Properties* window, scroll to find the *ForeColor* property.





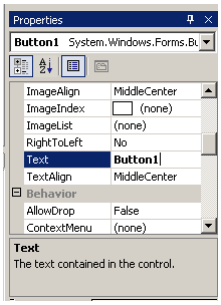


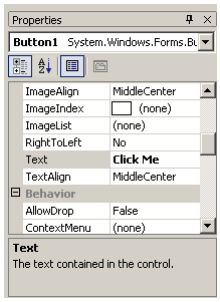


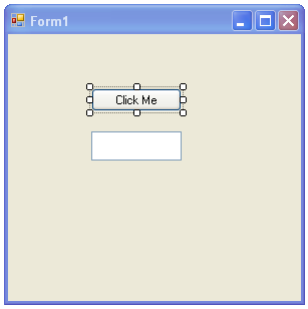
60

Click on the button to select it.

In the Properties window, change the Text string from "Button1" to "Click Me"







The button on the form will now display the message "Click Me".

61

Double click on the button.

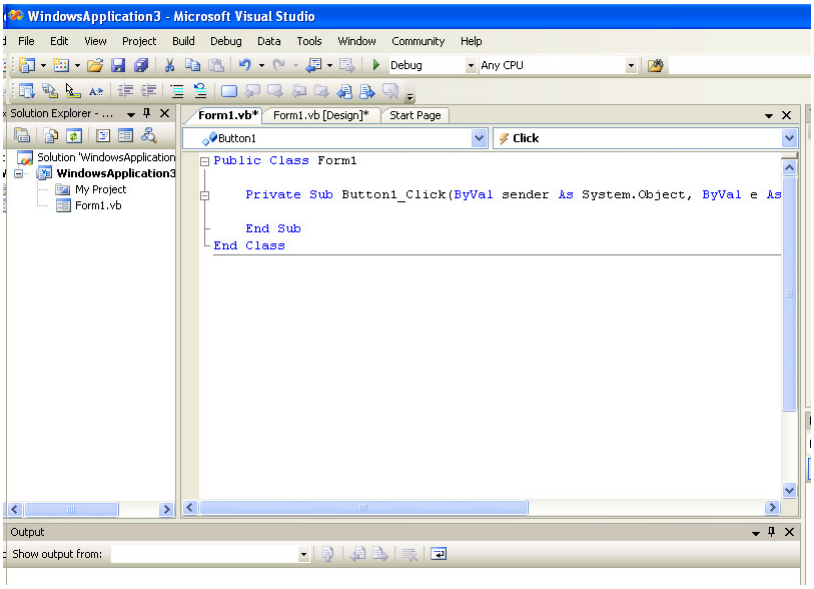
A *Code Designer* window opens up.

VB.NET automatically generates a skeleton `Sub` Procedure to handle the "event" when the button is clicked.

- When the button is clicked, this `Sub` Procedure will be called automatically.

This skeleton Visual Basic code (template) is displayed in the main area of the screen.

62



63

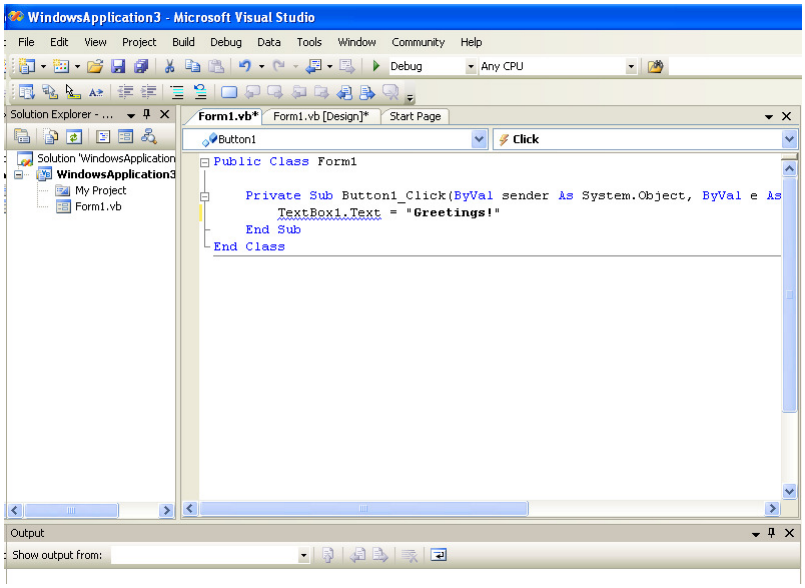
We now need to add Visual Basic statements to this `Sub` Procedure that will display a message when the button is clicked.

For the text box, we can assign a message as follows

```
TextBox1.Text = "Greetings!"
```

i.e. for the control called `TextBox1`, we assign the text message "Greetings!" to its `Text` property.

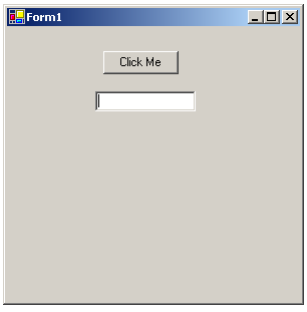
64

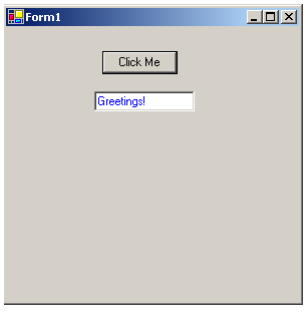


65

You are now in a position to test your first VB Windows program.

Select the *Start* option from the *Debug* pull-down menu.





You could alter further properties of the text box if necessary, e.g. font type, font size, centered, etc.

66

You are now in a position to modify your first VB Windows program.

Change/modify/update/alter the following properties of the TextBox control:

Forecolor to become green

Backcolor to become pink

Font to become Courier New

TextAlign to become centered

.

67

Other common controls:

labels: You place a label control to the left of a text box to tell the user what information should be typed in the text box

Name

list boxes: Have a number of uses, e.g. display tables or several lines of output.

68

The *Name* Property

Every control has a *Name* property.

It is used in the code to refer to the control, e.g. TextBox1, TextBox2, etc.

Common controls have ***short-form Name***

e.g.

`button`	`btn`
`label`	`lbl`
`list box`	`lst`
`text box`	`txt`

69

Working with Events

The Windows user interface relies on *events* to know when the user has done something

- when user clicks a button – `Click` event

- when user double clicks the mouse – `DoubleClick` event

- when user presses a key – `KeyPress` event

Controls may have several events, e.g. Button besides `Click` event, has `MouseUp` and `MouseDown` events

70

When a *VB.NET* program runs, the Form and its controls appear on the screen.

However, nothing happens until the user takes an action, causing an event.

We need to write *VB* code to handle these events.

Specifically, properties of controls are changed with an assignment statement

```
ControlName.Property = Value
```

71

E.g.

```
TextBox1.ForeColor = Color.Red
```

sets the color of characters in the text box

```
TextBox1.Text = "Hello"
```

displays the word "Hello" in the text box

```
btnButton.Visible = True
```

makes the button visible

72

VB.NET's Code Designer automatically generates a *code template* for handling an *event* for a *control* whenever you double click on a specific *control object* in the *Form* – *event (Sub) procedure*

E.g. Double clicking on a button control – *Code Designer* generates the following code template (in red). You (the programmer) must add *VB statements* to handle the event.

```
Private Sub Button1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
Button1.Click

        VB statements

End Sub
```

Inserted by programmer

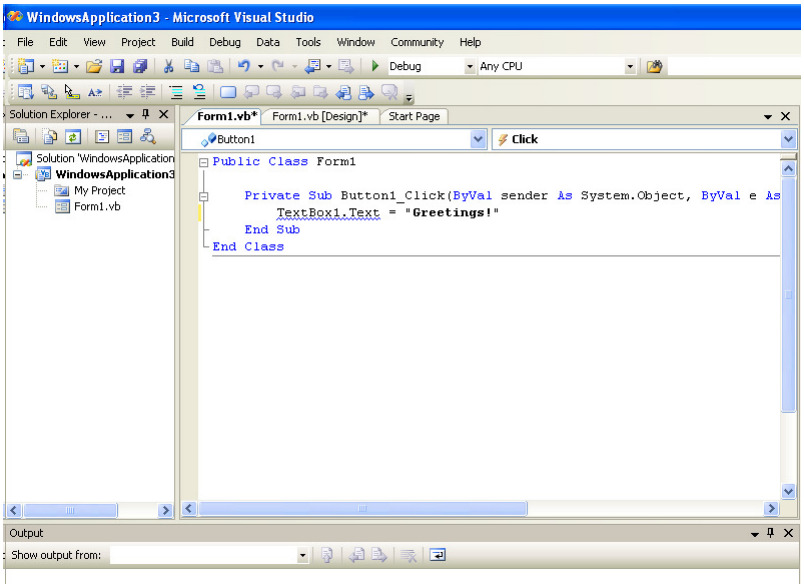
73

E.g.

```
Private Sub Button1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
Button1.Click

        TextBox1.Text = "Greetings!"

End Sub
```

displays the message "Greetings!" in the text box when the button is clicked.

74



75

The Declaration Statement of an *Event Procedure*

- A declaration statement for an *event procedure*

```
Private Sub btnOne_Click(...) Handles btnOne.Click
```

- The name can be changed at will. For example,

```
Private Sub ButtonPushed(...) Handles btnOne.Click
```

- Handling more than one event

```
Private Sub ButtonPushed(...) Handles btnOne.Click,
  btnTwo.Click
```

76

Whenever a control, say a textbox, is selected (cursor is visible in the textbox), we say that the textbox has the ***focus***.

It means that the textbox is the currently selected control and any keyboard actions will be sent directly to this control.

77

When a control loses the focus, the event

```
control.Leave
```

is triggered.

When a control gets the focus, the event

```
control.Enter
```

is triggered.

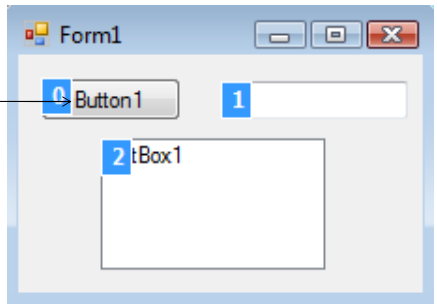
To move the focus to a specified control, use

```
control.Focus()
```

78

Tab Order /Schneider09, pp53/

Tab index



The tab indices determine the order in which controls receive the focus during tabbing.

79

You may also assign properties to the *Form* itself in the code.

Not

```
Form1.Text = "Demonstration"
```

But

```
Me.Text = "Demonstration"
```

The form is referred to by the keyword `Me`.

80

Three other useful properties:

`BackColor` - specifies the background color for the form or control.

`Visible` – setting this property to `False` causes the control to disappear when the program is run. It can be made to reappear with code.

`Enabled` – setting this property to `False` restricts its use. It appears grayed and cannot receive the focus. This means that the user will not be able to type text in the textbox. Indeed, there exist cases when you want the program to update the string in the text box but you do not want the user to be able to type text in the text box.

81

One way to end a Windows application program is by getting the user to click on a button named, say "Exit"

The following VB statement then has to be entered in the sub-procedure handling the button click event.

```
Me.Close()
```

OR only

```
Close()
```

OR simply

```
End
```

82

Add one more button to this demo program, name it "quit" or "exit".

Insert in the body of the corresponding event handler one of the following statements:

```
Me.Close()
```
OR

```
Close()
```
OR

```
End
```

83

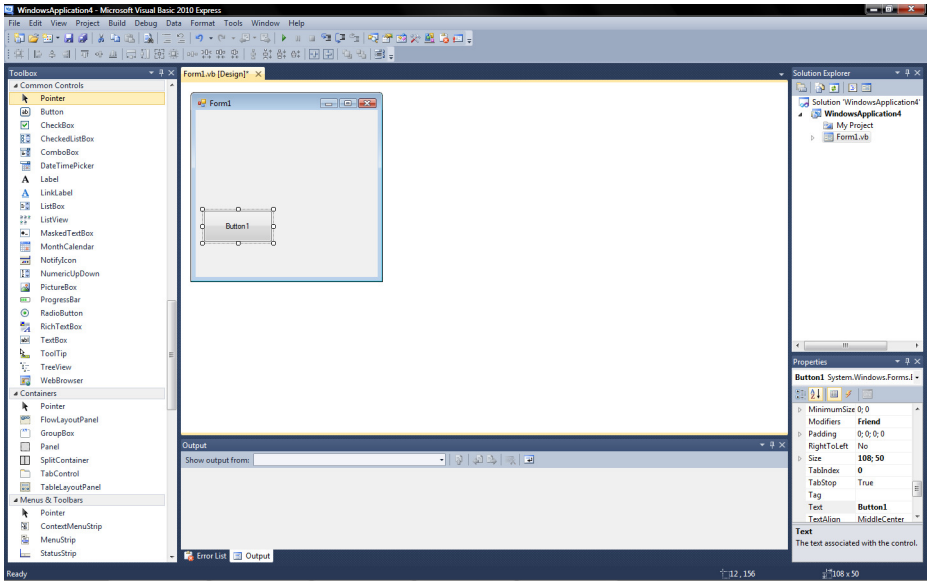
PRACTICAL TASKS

84

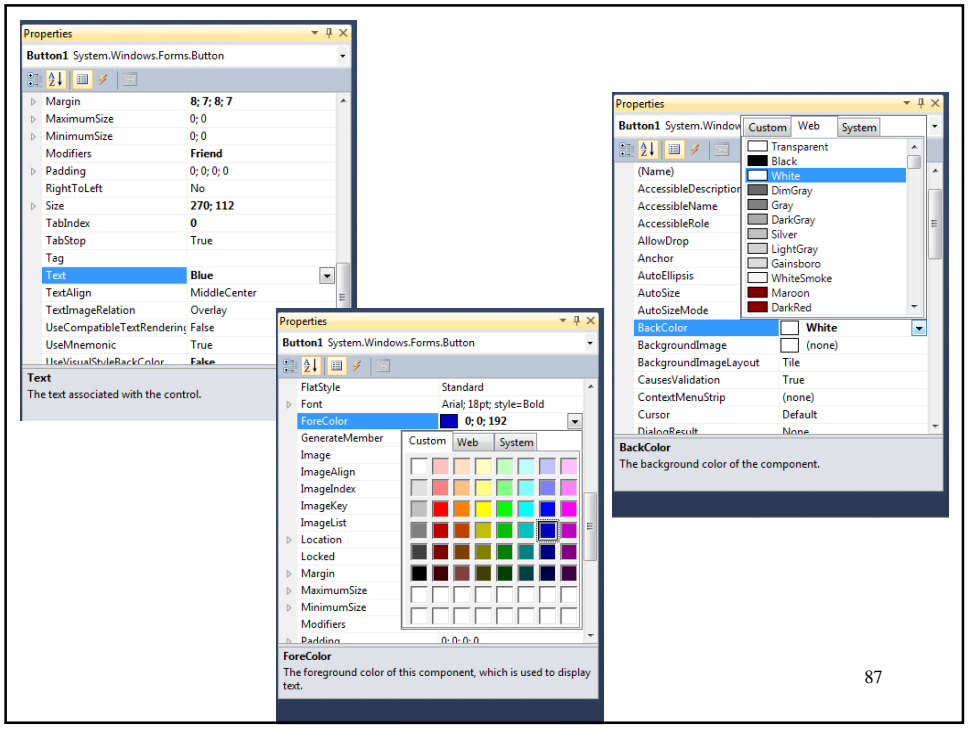
Task 1

- Develop Windows application with 2 buttons that allows the user to change the application form background color alternatively.

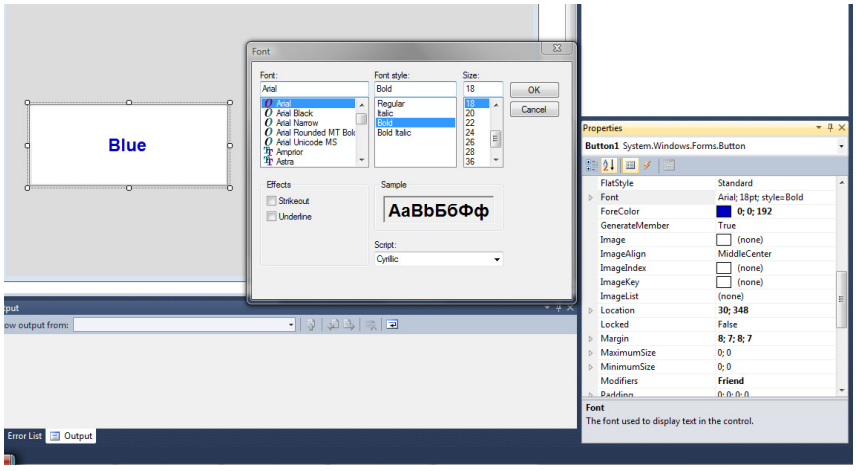
85



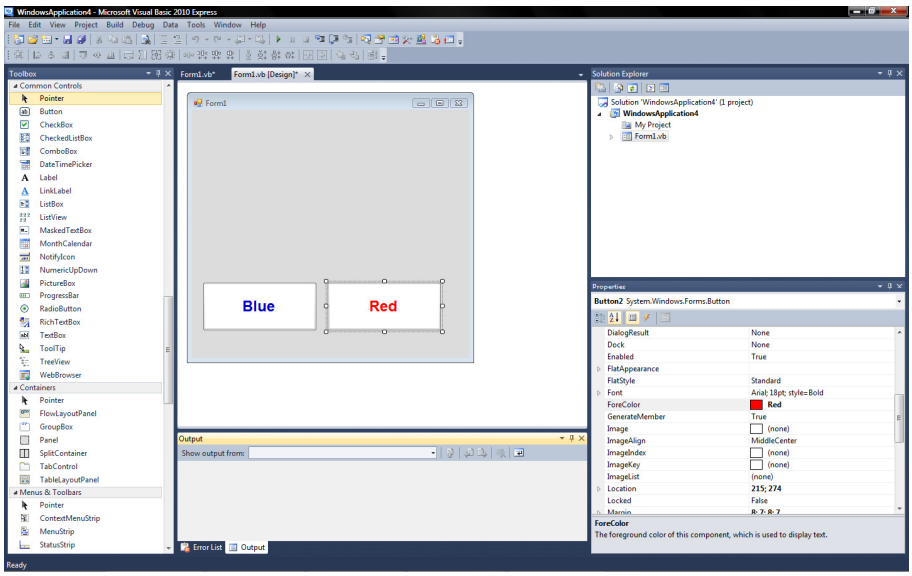
86


Properties
Button1 System.Windows.Forms.Button
Margin 8; 7; 8; 7
MaximumSize 0; 0
MinimumSize 0; 0
Modifiers Friend
Padding 0; 0; 0; 0
RightToLeft No
Size 270; 112
TabIndex 0
TabStop True
Tag
Text Blue
TextAlign MiddleCenter
TextImageRelation Overlay
UseCompatibleTextRendering False
UseMnemonic True
Text
The text associated with the control.
Properties
Button1 System.Windows.Forms.Button
FlatStyle Standard
Font Arial; 18pt; style=Bold
ForeColor 0; 0; 192
GenerateMember
Image
ImageAlign
ImageIndex
ImageKey
ImageList
Location
Locked
Margin
MaximumSize
MinimumSize
Modifiers
Custom
Web
System
ForeColor
The foreground color of this component, which is used to display text.
Properties
Button1 System.Windo
Custom
Web
System
Transparent
Black
White
DimGray
Gray
DarkGray
Silver
LightGray
Gainsboro
WhiteSmoke
Maroon
DarkRed
(Name)
AccessibleDescription
AccessibleName
AccessibleRole
AllowDrop
Anchor
AutoEllipsis
AutoSize
AutoSizeMode
BackColor White
BackgroundImage (none)
BackgroundImageLayout Tile
CausesValidation True
ContextMenuStrip (none)
Cursor Default
BackColor
The background color of the component.

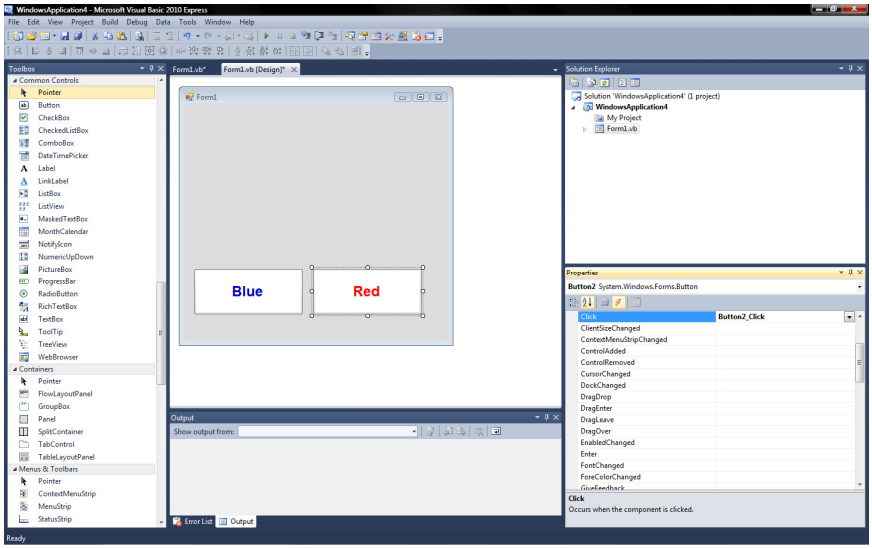
87


Blue
Font
Font:
Arial
Arial Black
Arial Narrow
Arial Rounded MT Bold
Arial Unicode MS
Amprior
Astra
Font style:
Bold
Regular
Italic
Bold
Bold Italic
Size:
18
20
22
24
26
28
36
OK
Cancel
Effects
Strikeout
Underline
Sample
АаВbБбФф
Script:
Cyrillic
Properties
Button1 System.Windows.Forms.Button
FlatStyle Standard
Font Arial; 18pt; style=Bold
ForeColor 0; 0; 192
GenerateMember True
Image (none)
ImageAlign MiddleCenter
ImageIndex (none)
ImageKey (none)
ImageList (none)
Location 30; 348
Locked False
Margin 8; 7; 8; 7
MaximumSize 0; 0
MinimumSize 0; 0
Modifiers Friend
Font
The font used to display text in the control.
Error List
Output

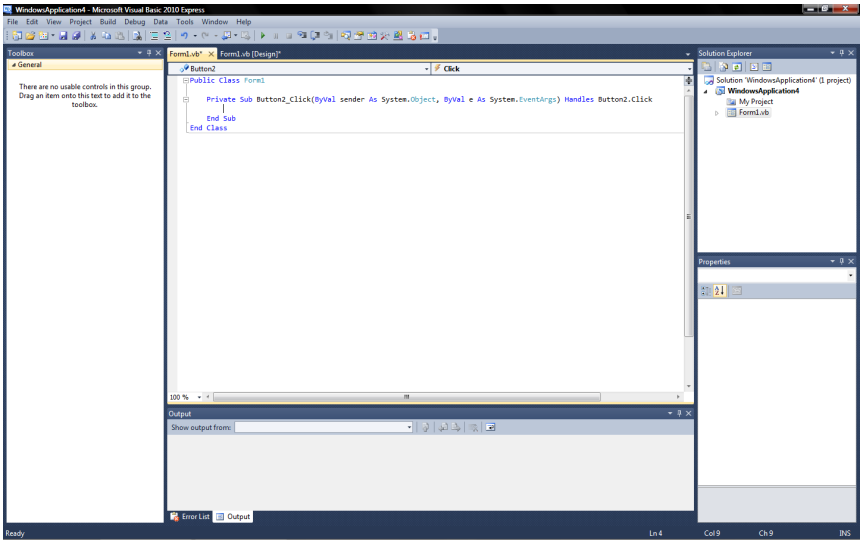
88


WindowsApplication4 - Microsoft Visual Basic 2010 Express
Blue
Red

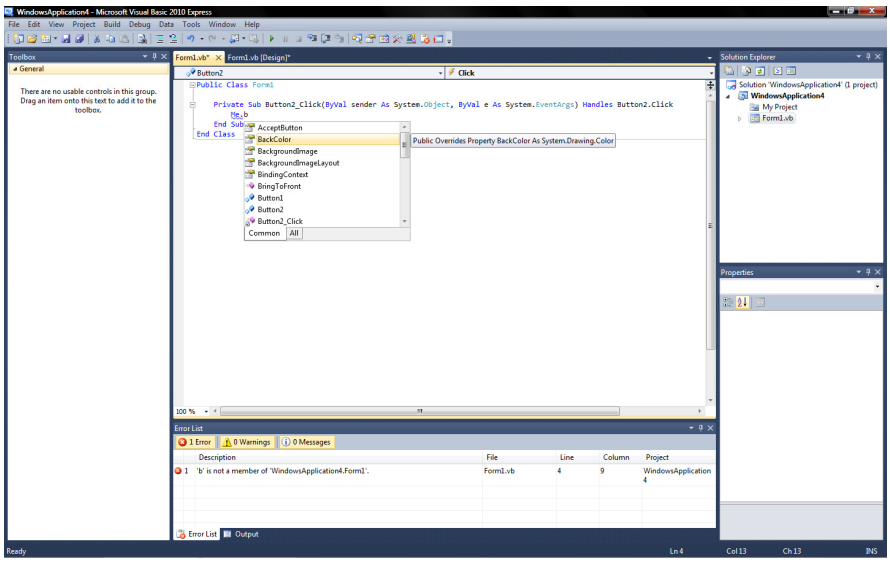
89


WindowsApplication4 - Microsoft Visual Basic 2010 Express
Blue
Red

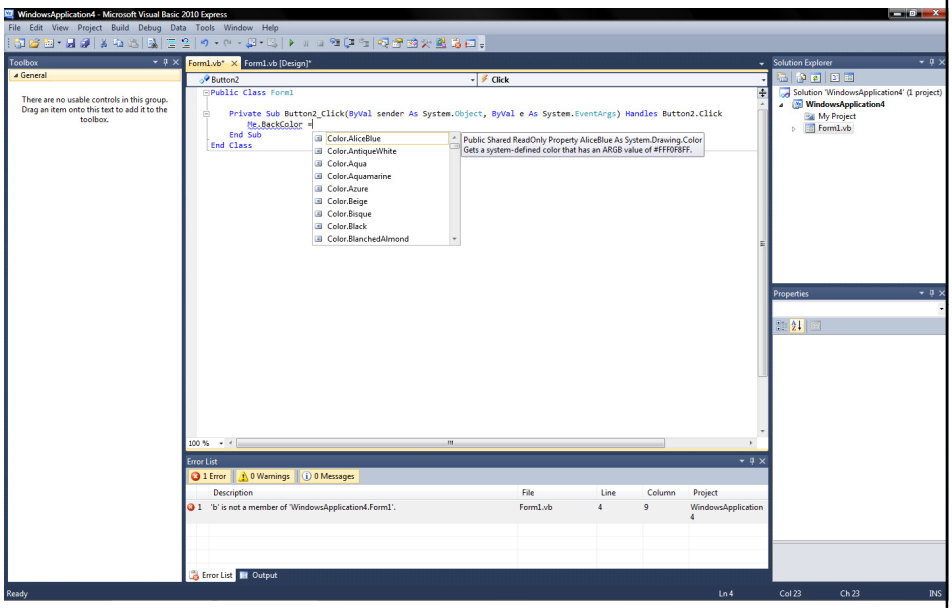
90


WindowsApplication4 - Microsoft Visual Basic 2010 Express
File Edit View Project Build Debug Data Tools Window Help
Toolbox
General
There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.
Form1.vb* Form1.vb [Design]*
Button2
Click
Public Class Form1
Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click
End Sub
End Class
Solution Explorer
Solution 'WindowsApplication4' (1 project)
WindowsApplication4
My Project
Form1.vb
Properties
Output
Show output from:
Error List
Output
Ready
Ln 4
Col 9
Ch 9
INS

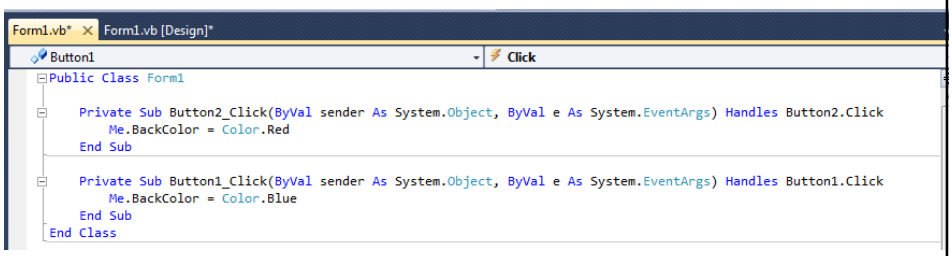
91


WindowsApplication4 - Microsoft Visual Basic 2010 Express
Toolbox
General
There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.
Form1.vb* Form1.vb [Design]*
Button2
Click
Public Class Form1
Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click
Me.b
End Sub
End Class
AcceptButton
BackColor
BackgroundImage
BackgroundImageLayout
BindingContext
BringToFront
Button1
Button2
Button2_Click
Common
All
Public Overrides Property BackColor As System.Drawing.Color
Solution Explorer
Solution 'WindowsApplication4' (1 project)
WindowsApplication4
My Project
Form1.vb
Properties
Error List
1 Error
0 Warnings
0 Messages
Description
File
Line
Column
Project
1 'b' is not a member of 'WindowsApplication4.Form1'.
Form1.vb
4
9
WindowsApplication4
Error List
Output
Ready
Ln 4
Col 13
Ch 13
INS

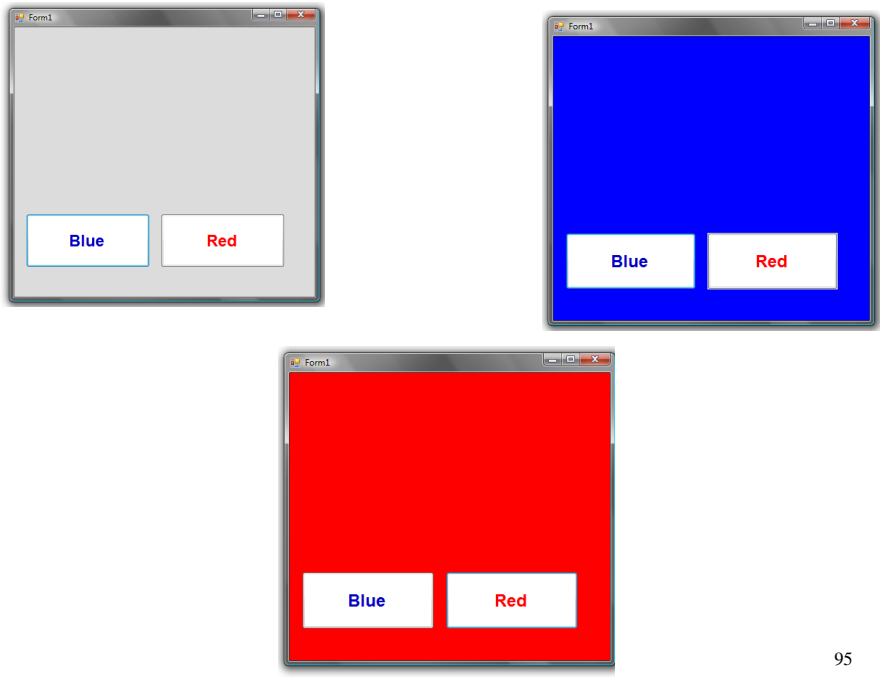
92


WindowsApplication4 - Microsoft Visual Basic 2010 Express
Public Class Form1
Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click
Me.BackColor =
End Sub
End Class
Color.AliceBlue
Color.AntiqueWhite
Color.Aqua
Color.Aquamarine
Color.Azure
Color.Beige
Color.Bisque
Color.Black
Color.BlanchedAlmond
Public Shared ReadOnly Property AliceBlue As System.Drawing.Color
Gets a system-defined color that has an ARGB value of #FFF0F8FF.
1 Error
0 Warnings
0 Messages
Description
File
Line
Column
Project
1 'b' is not a member of 'WindowsApplication4.Form1'.
Form1.vb
4
9
WindowsApplication4

93


Form1.vb*
Form1.vb [Design]*
Button1
Click
Public Class Form1
Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click
Me.BackColor = Color.Red
End Sub
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
Me.BackColor = Color.Blue
End Sub
End Class

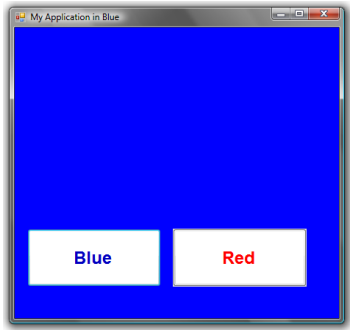
94

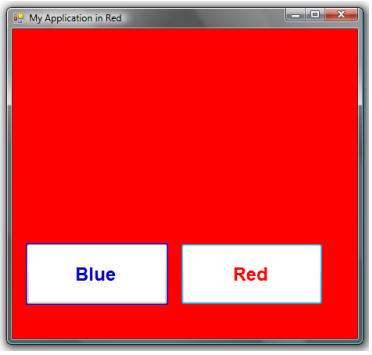


95

```
Form1.vb*  Form1.vb [Design]*
Button1                                   Click
Public Class Form1

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click
        Me.BackColor = Color.Red
        Me.Text = "My Application in Red"
    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
        Me.BackColor = Color.Blue
        Me.Text = "My Application in Blue"
    End Sub
End Class
```

96



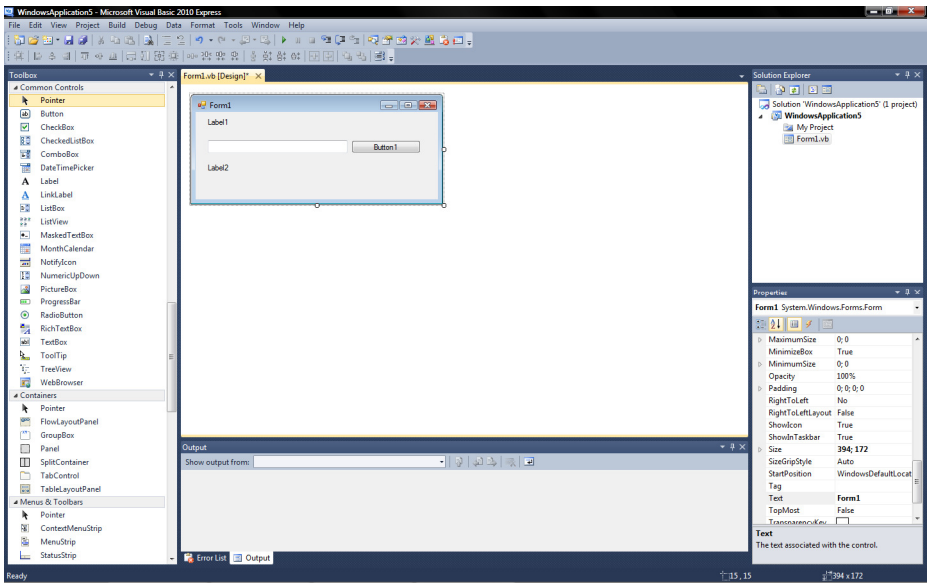


97

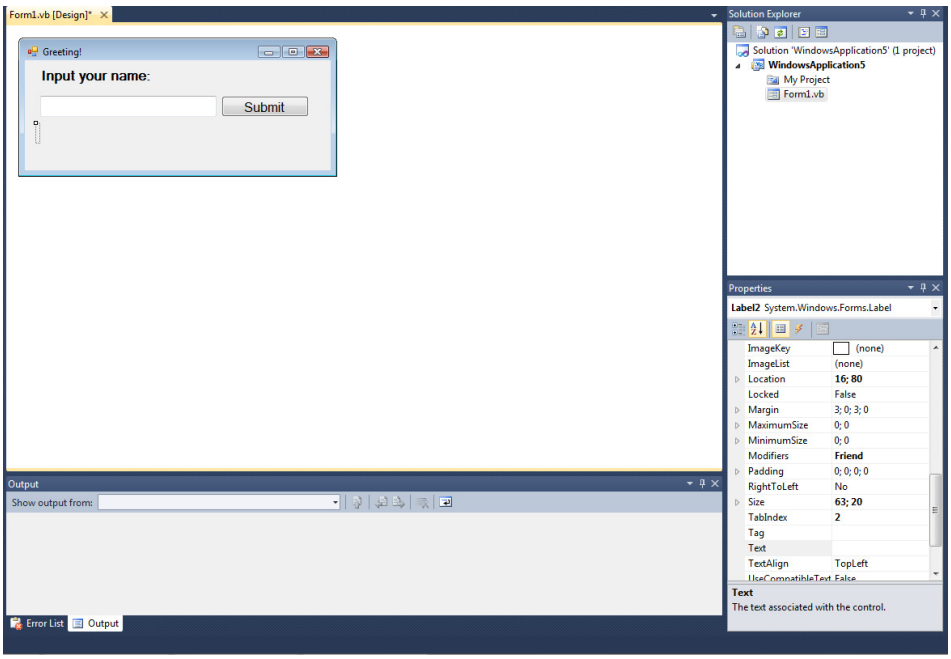
Task 2

- Develop Windows application that asks the user to input his/her name and displays greeting message.
- Components:
  - 2 labels
  - 1 textbox
  - 1 button

98


WindowsApplication5 - Microsoft Visual Basic 2010 Express
Toolbox
Form1.vb [Design]*
Form1
Label1
Button1
Label2
Solution Explorer
Solution 'WindowsApplication5' (1 project)
WindowsApplication5
My Project
Form1.vb
Properties
Output
Error List
Output
Ready

99


Form1.vb [Design]*
Greeting!
Input your name:
Submit
Solution Explorer
Solution 'WindowsApplication5' (1 project)
WindowsApplication5
My Project
Form1.vb
Properties
Label2 System.Windows.Forms.Label
Output
Show output from:
Error List
Output

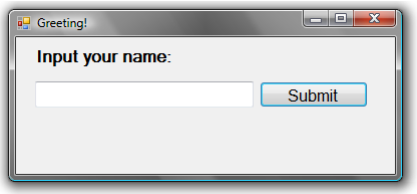
100

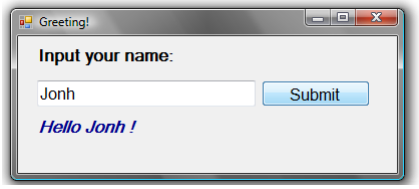
Form1.vb* | Form1.vb [Design]*

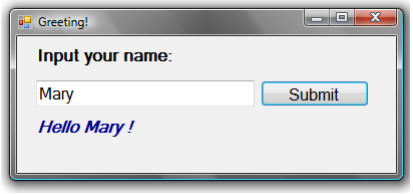
(General) | (Declarations)

```
Public Class Form1

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
        Label2.Text = "Hello " + TextBox1.Text + " !"
    End Sub
End Class
```

101







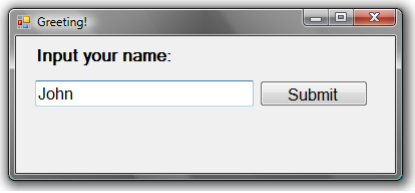
102

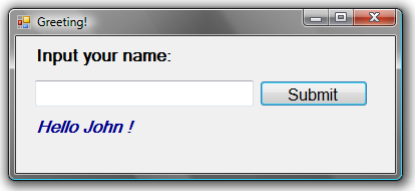
Form1.vb* | Form1.vb [Design]*

Button1 | Click

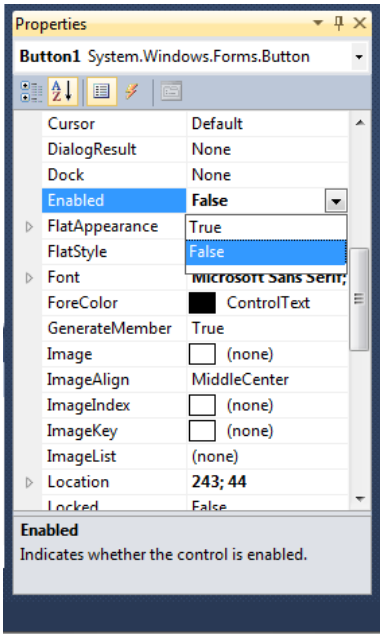
```
Public Class Form1

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
        Label2.Text = "Hello " + TextBox1.Text + " !"
        TextBox1.Text = ""
    End Sub
End Class
```

103

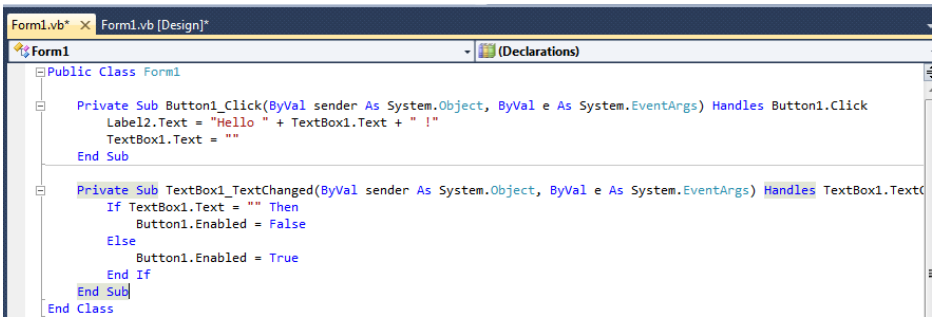




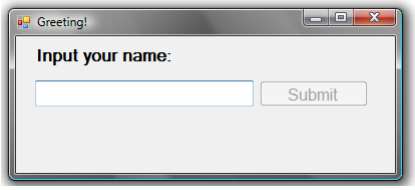
104

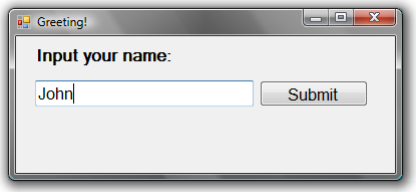

Properties
Button1 System.Windows.Forms.Button
Cursor Default
DialogResult None
Dock None
Enabled False
FlatAppearance
FlatStyle
Font
ForeColor ControlText
GenerateMember True
Image (none)
ImageAlign MiddleCenter
ImageIndex (none)
ImageKey (none)
ImageList (none)
Location 243; 44
Locked False
True
False
Enabled
Indicates whether the control is enabled.

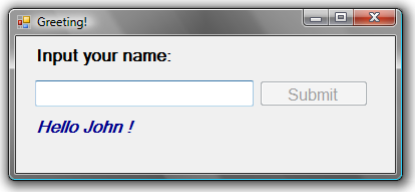
105


Form1.vb*
Form1.vb [Design]*
Form1
(Declarations)
Public Class Form1
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
Label2.Text = "Hello " + TextBox1.Text + " !"
TextBox1.Text = ""
End Sub
Private Sub TextBox1_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles TextBox1.Text
If TextBox1.Text = "" Then
Button1.Enabled = False
Else
Button1.Enabled = True
End If
End Sub
End Class

106


Greeting!
Input your name:
Submit


Greeting!
Input your name:
John
Submit


Greeting!
Input your name:
Submit
Hello John !

107