

جامعة بغداد/ كلية التربية للعلوم الصرفة ابن الهيثم/ قسم علوم الحاسبات
المرحلة الثالثة صباحي / مسائي (نظري)
السنة ٢٠٢١-٢٠٢٢
أستاذة المادة: د. شهلاء طالب

Visual Programming

Lecture 1 – Introduction

Visual programming is a type of programming language that lets humans describe processes using illustration. Whereas a typical text-based programming language makes the programmer think like a computer, a visual programming language lets the programmer describe the process in terms that make sense to humans.

- A visual programming language (VPL) is a programming language that uses graphical elements and figures to develop a program.
- A VPL employs techniques to design a software program in two or more dimensions, and includes graphical elements, text, symbols and icons within its programming context.
- A visual programming language is also known as an executable graphics language.

Visual Basic 2012 – Introduction

Microsoft launched Visual Basic 2012 in the year 2012. It is a fully object-oriented programming language implemented on the .NET Framework. Similar to the earlier version of VB.NET programming languages, VB2012 is integrated with other Microsoft Programming languages in an IDE called Visual Studio Express 2012.

The reasons for of implementing Visual Basic program are listed as follows:

- 1- It uses Integrated Development Environment (IDE) which is easier for the user to minimize code writing.
- 2- All visual programs follow the same concepts, therefore the user will become more familiar with visual approach for other visual languages.
- 3- It provides Input box and Output box as an interactive windows with user.
- 4- It is able to connect to Internet, and to call Explorer.

When you launch Visual Studio Express 2012, the start page will appear, as shown in Figure 1.1 below.

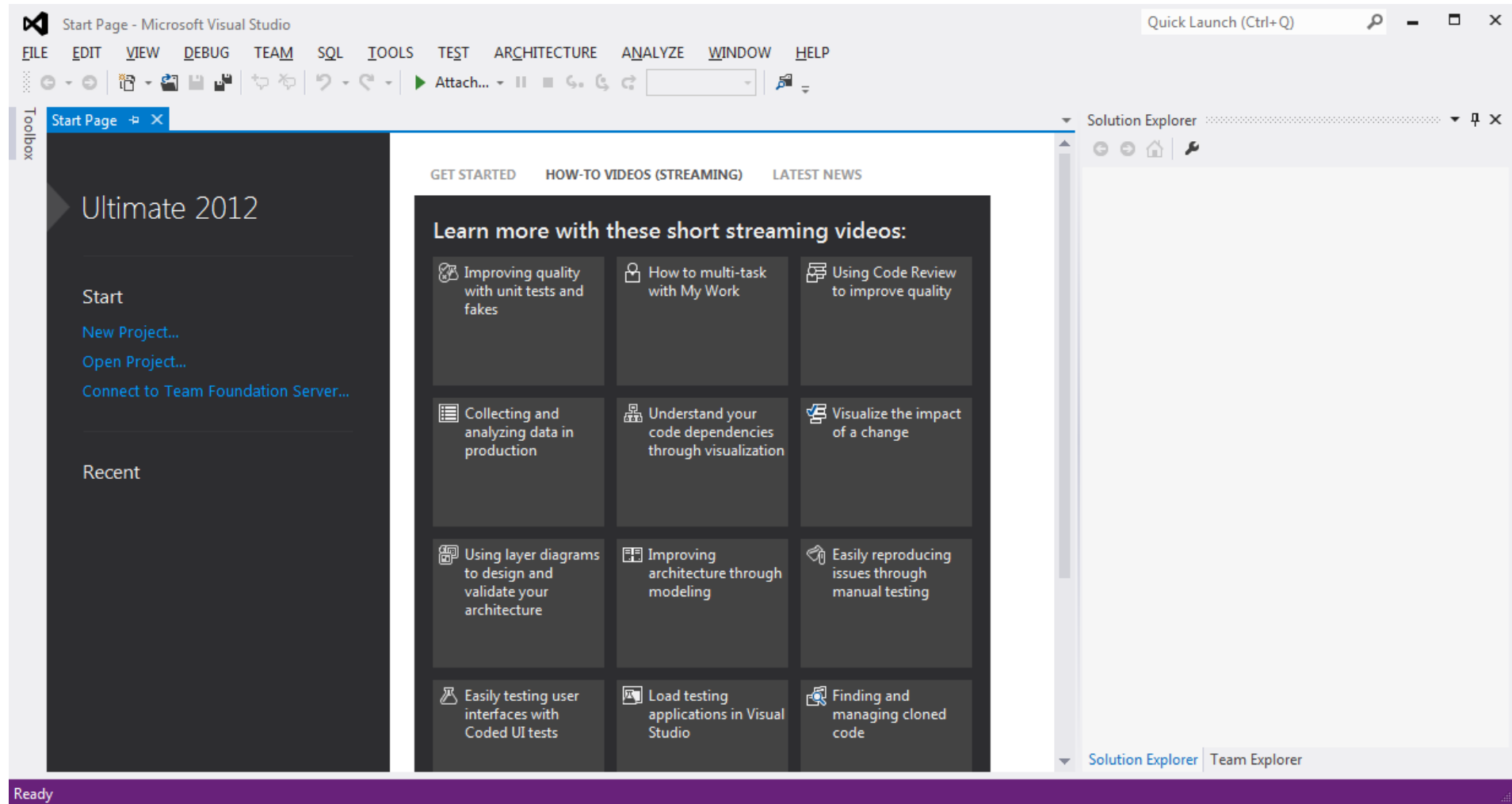


Figure 1.1: Visual Studio 2012 Start Page

To start a new Visual Studio Express 2012 project, simply click on New Project to launch the Visual Studio New Project page, as shown in Figure 1.2

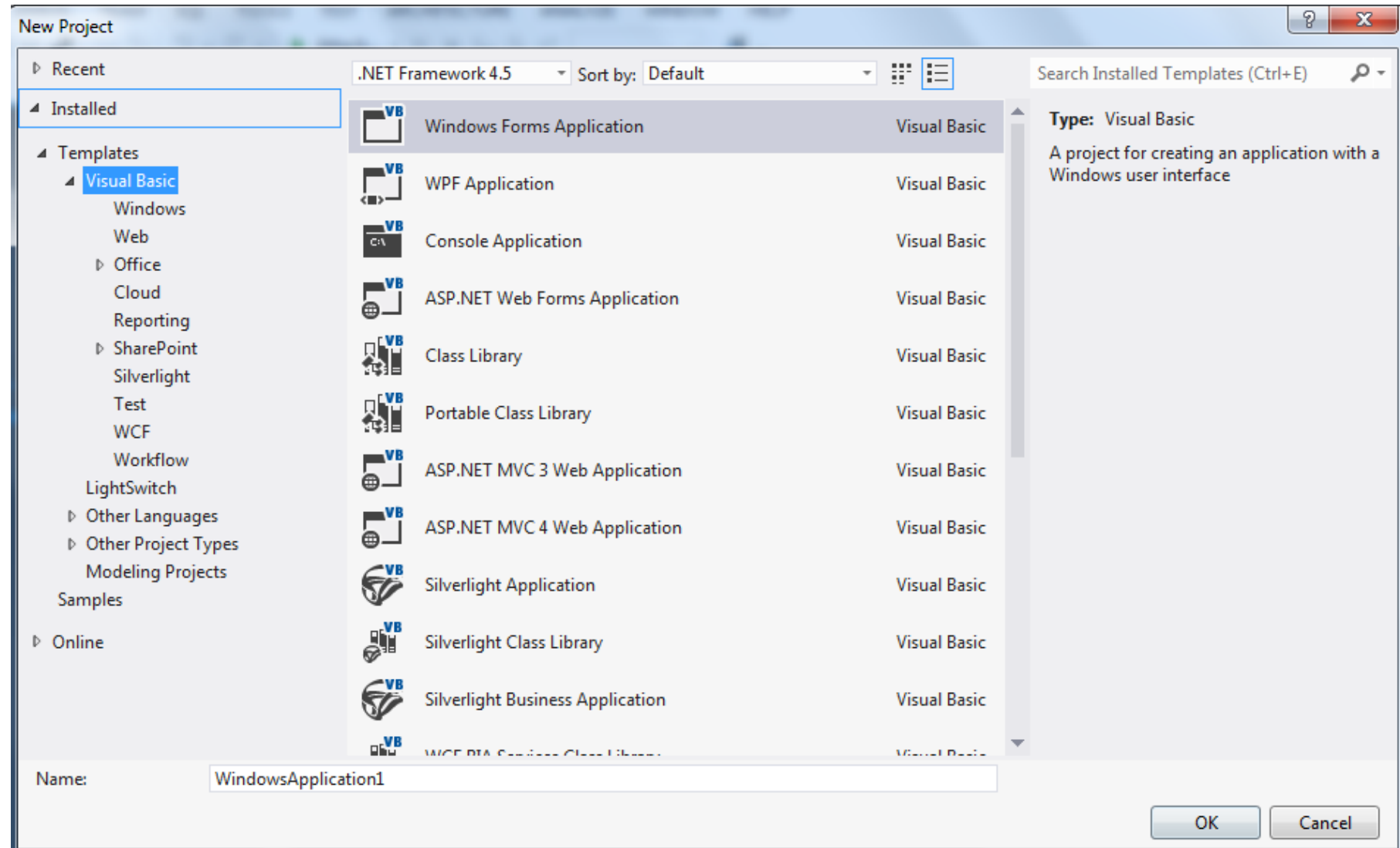


Figure 1.2: Visual Studio 2012 Project Page

- The New Project Page (Figure 1.2) comprises three templates, Visual Basic, Visual C# and Visual C++. Since we are going to learn Visual Basic 2012, we shall select Visual Basic. Visual Basic 2012 offers you four types of projects that you can create. As we are going to learn to create Windows Applications, we will select Windows Forms Application.
- At the bottom of this dialog box, you can change the default project name WindowsApplication1 to some other name you like, for example, MyFirstProgram. After you have renamed the project, click OK to continue. The Toolbox is not shown until you click on the Toolbox tab. When you click on the Toolbox tab, the common controls Toolbox will appear.

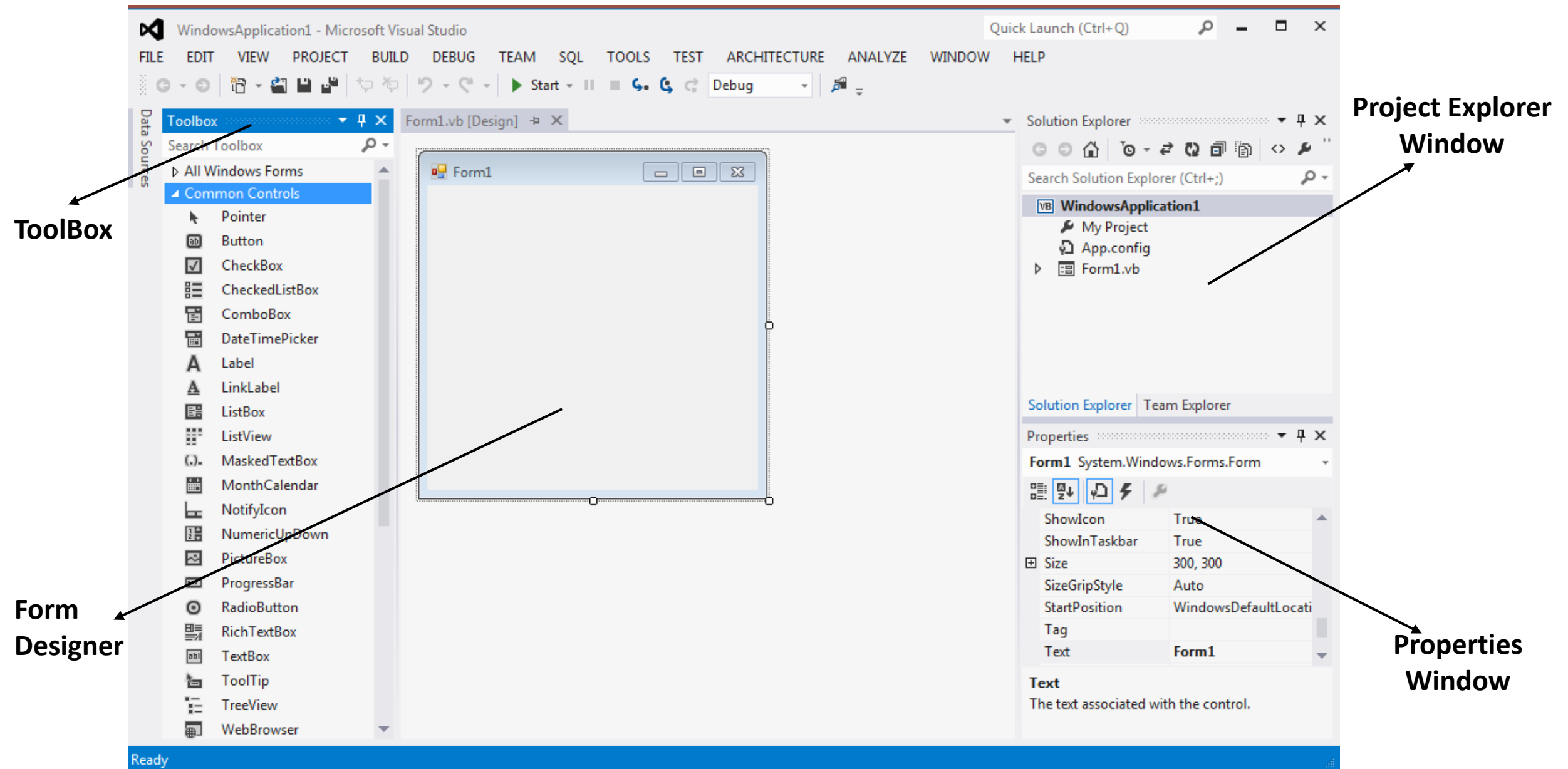


Figure 1.3: Visual Basic 2012 IDE

- Visual Basic Express 2012 IDE (Figure 1.3) comprises a few windows, the Form window, the Solution Explorer window and the Properties window. It also consists of a toolbox which contains many useful controls that allow a programmer to develop his or her VB programs.
- **Form Designer:** it is a window for each form to customize the designed interface of the application. Using the form designer, the user can add controls, graphics, and text to create the desired form appearance.
- **Project Explorer Window:** it is a list of the forms and modules for the current projects. It is a hierarchical tree- branch structure, where the project at top of tree and other parts like forms ,modules descend from this tree.
- **Properties Window:** it is a List of properties settings for a selected form or a control. These properties are characteristics (such as size, visible, or color) of the selected object it provides an easy way to set properties.
- **ToolBox:** it contains a collection of tools that are needed for project design.

What are Controls

Controls in Visual Basic 2012 are objects that can be placed on the form to perform various tasks. Figure 1.4 shows the toolbox that contains the controls. They are categorized into Common Controls, Containers, Menus, Toolbars, Data, Components, Printings and Dialogs. At the moment, we will focus on the common controls. Some frequently used common controls are Button, Label, ComboBox, ListBox, PictureBox, and TextBox. To insert a control into your form in Visual Basic 2012 IDE, drag the control from the toolbox and drop it onto the form. You can reposition and resize it as you like.

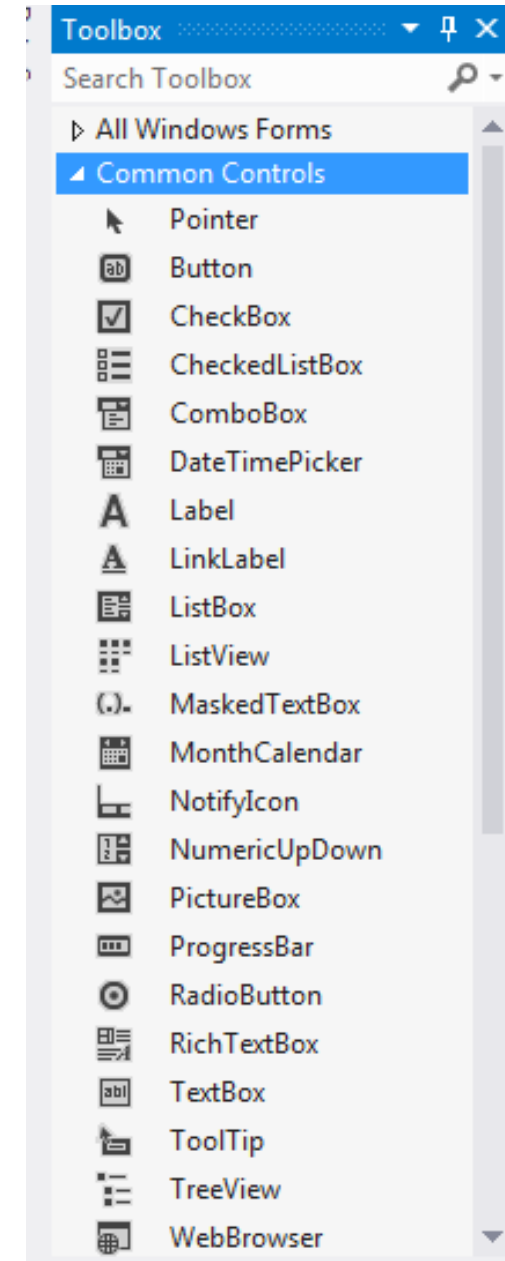


Figure 1.4: Visual Basic 2012 Toolbox

Now, we shall proceed to show you how to create your first program. First, change the text of the form to My First Program in the properties window, it will appear as the title of the program. Next, insert a button and change its text to OK. The design interface is shown in Figure 1.5.

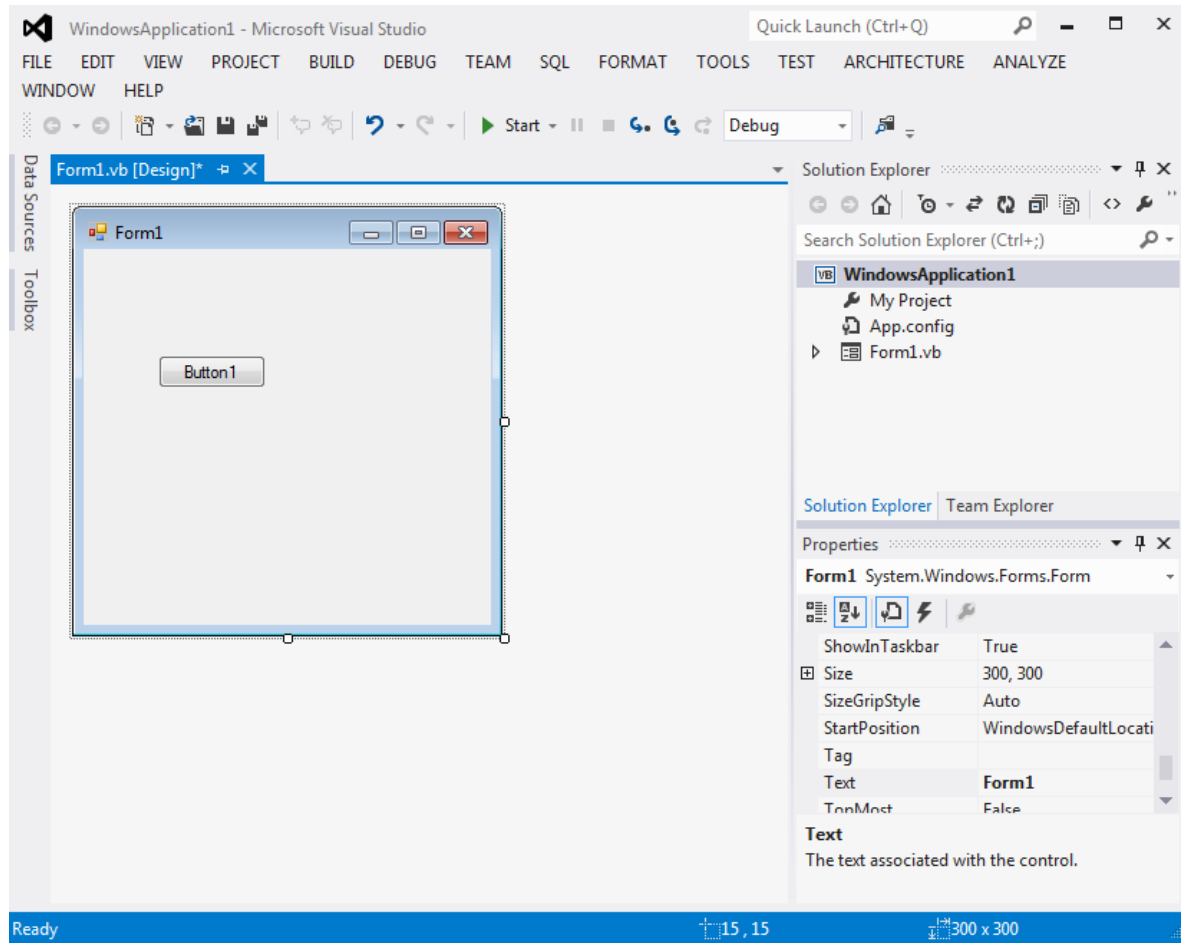


Figure 1.5: The Design Interface

Now click on the OK button to bring up the code window and enter the following statement between Private Sub and End Sub procedure, as shown in Figure 1.6.

```
MsgBox("My First Visual Basic  
2012 Program")
```

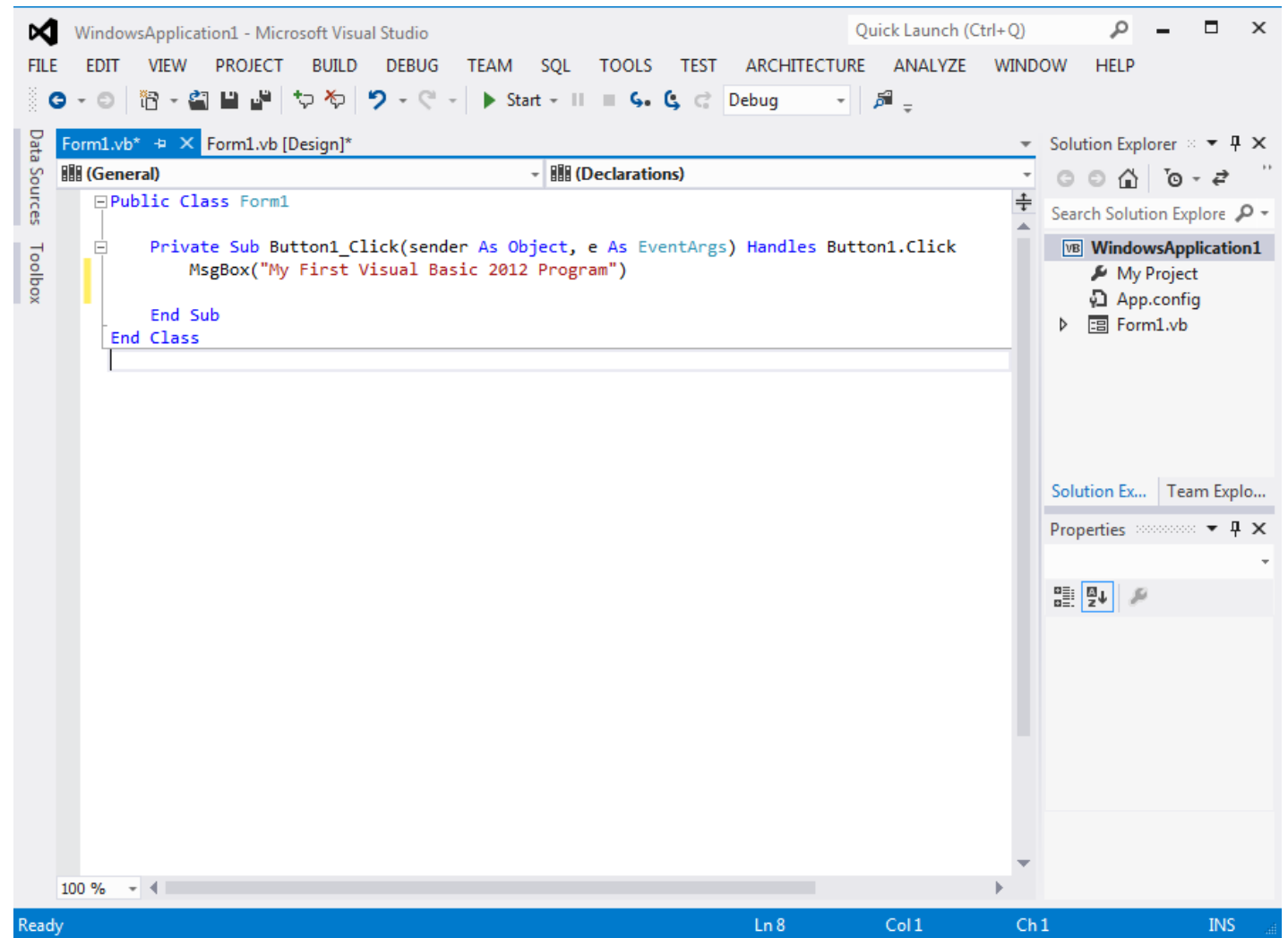


Figure 1.6: The Code Window

Now click on the Start on the toolbar to run the program then click on the OK button, a dialog box that displays the “My First Visual Basic 2012 Program” message will appear, as shown in Figure 1.7.

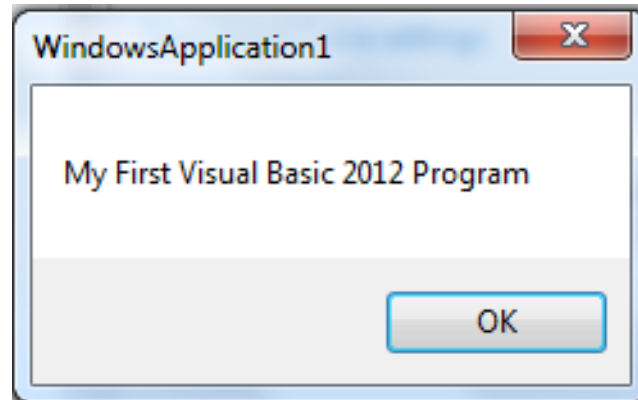


Figure 1.7: The Message Box

The function `MsgBox` is a built-in function of Visual Basic 2012 and it will display the text enclosed within the brackets.

جامعة بغداد/ كلية التربية للعلوم الصرفة ابن الهيثم/ قسم علوم الحاسبات
المرحلة: الثالثة صباحي / مسائي (نظري)
أستاذة المادة: د. شهلاء طالب

Visual Programming

Lecture 2 – Data Types, Variables, Constants and Discussion Making

Data Types Available in VB.Net

Data types refer to an extensive system used for declaring variables or functions of different types. The type of a variable determines how much space it occupies in storage and how the bit pattern stored is interpreted.

VB.Net provides a wide range of data types. The following table shows some of the data types available –

Data Type	Storage Allocation	Value Range
Boolean	Depends on implementing platform	True or False
Byte	1 byte	0 through 255 (unsigned)
Char	2 bytes	0 through 65535 (unsigned)
Date	8 bytes	0:00:00 (midnight) on January 1, 0001 through 11:59:59 PM on December 31, 9999
Decimal	16 bytes	0 through +/-79,228,162,514,264,337,593,543,950,335 (+/-7.9...E+28) with no decimal point; 0 through +/-7.9228162514264337593543950335 with 28 places to the right of the decimal
Double	8 bytes	-1.79769313486231570E+308 through -4.94065645841246544E-324, for negative values 4.94065645841246544E-324 through 1.79769313486231570E+308, for positive values
Integer	4 bytes	-2,147,483,648 through 2,147,483,647 (signed)
Long	8 bytes	-9,223,372,036,854,775,808 through 9,223,372,036,854,775,807(signed)

Variables

The following example demonstrates use of some of the types –

```
Dim b As Byte
```

```
Dim n As Integer
```

```
Dim d As Double
```

```
Dim da As Date
```

```
Dim c As Char
```

```
Dim s As String
```

```
Dim bl As Boolean
```


The Type Conversion Functions in VB.Net

Sr.No.	Functions & Description
1	CBool(expression) Converts the expression to Boolean data type.
2	CByte(expression) Converts the expression to Byte data type.
3	CChar(expression) Converts the expression to Char data type.
4	CDate(expression) Converts the expression to Date data type
5	CDbl(expression) Converts the expression to Double data type.
6	CDec(expression) Converts the expression to Decimal data type.
7	CInt(expression) Converts the expression to Integer data type.

8	CStr(expression) Converts the expression to String data type.
---	---

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

    Dim n As Integer
    Dim da As Date
    Dim b1 As Boolean
    n = 1234567
    da = Today
    Console.WriteLine(b1)
    Console.WriteLine(CByte(b1))
    Console.WriteLine(CStr(b1))
    Console.WriteLine(CStr(da))
    Console.WriteLine(CChar(CChar(CStr(n))))
    Console.WriteLine(CChar(CStr(da)))

End Sub
```

False
0
False
18/10/2021
1
1

Constant and Enumeration

The constants refer to fixed values that the program may not alter during its execution. These fixed values are also called literals.

Constants can be of any of the basic data types like an integer constant, a floating constant, a character constant, or a string literal. There are also enumeration constants as well.

The constants are treated just like regular variables except that their values cannot be modified after their definition.

An enumeration is a set of named integer constants.

Example

```
Private Sub Button2_Click(sender As Object, e As  
EventArgs) Handles Button2.Click
```

```
    Const PI = 3.14149
```

```
    Dim radius, area As Single
```

```
    radius = 7
```

```
    area = PI * radius * radius
```

```
    Console.WriteLine("Area = " & Str(area))
```

```
End Sub
```

Area = 153.933

Example

```
Private Sub Button3_Click(sender As Object, e As EventArgs) Handles  
Button3.Click
```

```
    Console.WriteLine("The Color Red is : " & Colors.red)
```

```
    Console.WriteLine("The Color Yellow is : " & Colors.yellow)
```

```
    Console.WriteLine("The Color Blue is : " & Colors.blue)
```

```
    Console.WriteLine("The Color Green is : " & Colors.green)
```

```
End Sub
```

```
Enum Colors
```

```
    red = 1
```

```
    orange = 2
```

```
    yellow = 3
```

```
    green = 4
```

```
    azure = 5
```

```
    blue = 6
```

```
    violet = 7
```

```
End Enum
```

The Color Red is : 1
The Color Yellow is : 3
The Color Blue is : 6
The Color Green is : 4

Sr.No	Statements and Description	Example
1	Dim Statement Declares and allocates storage space for one or more variables.	<pre>Dim number As Integer Dim quantity As Integer = 100 Dim message As String = "Hello!"</pre>
2	Const Statement Declares and defines one or more constants.	<pre>Const maximum As Long = 1000 Const naturalLogBase As Object = CDec(2.7182818284)</pre>
3	Enum Statement Declares an enumeration and defines the values of its members.	<pre>Enum CoffeeMugSize Jumbo ExtraLarge Large Medium Small End Enum</pre>

Discussion Making in VB

Generally, in Visual Basic 2012 the statement that needs to be executed based on the condition is known as a “**Conditional Statement**” and the statement is more likely a block of code.

Visual Basic If Statement

Syntax of Visual Basic if Statement

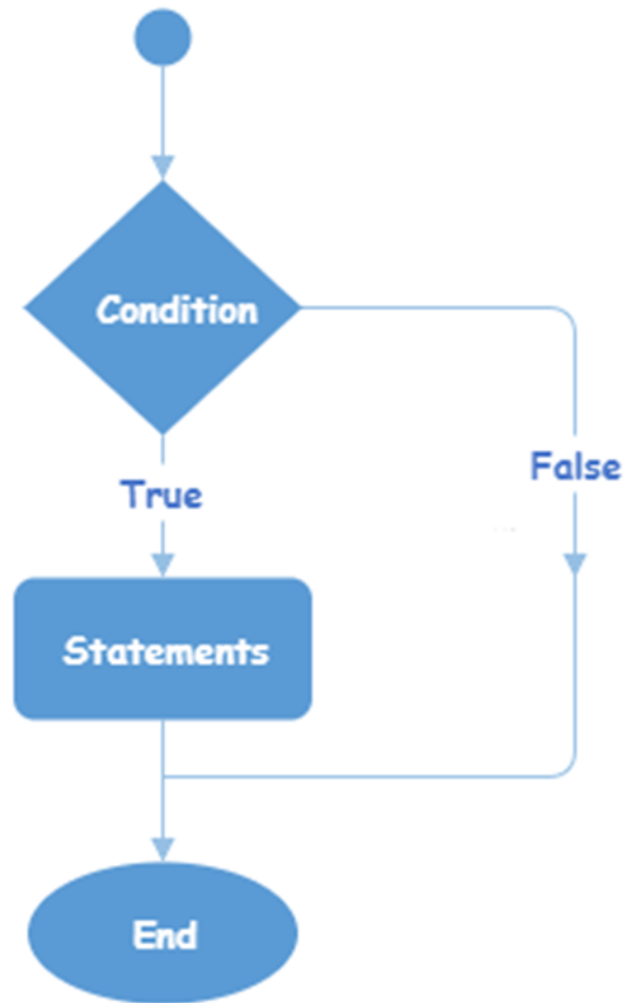
```
If bool_expression Then
```

```
// Statements to Execute if condition is true
```

```
End If
```

Visual Basic If Statement Flow Chart Diagram

Following is the flow chart diagram which will represent the process flow of **If statement** in Visual Basic programming language.



Visual Basic If Statement Example

Following is the example of defining the If statement in Visual Basic programming language to execute the block of code or statements based on a Boolean expression.

```
Private Sub Button2_Click(sender As Object, e As EventArgs)
Handles Button2.Click
    Dim x As Integer = 20, y As Integer = 10
    If x >= 10 Then
        MsgBox("x is Greater than 10")
    End If
    If y <= 5 Then
        MsgBox("y is less than or equals to 5")
    End If
    MsgBox("Press Enter Key to Exit..")
End Sub
```

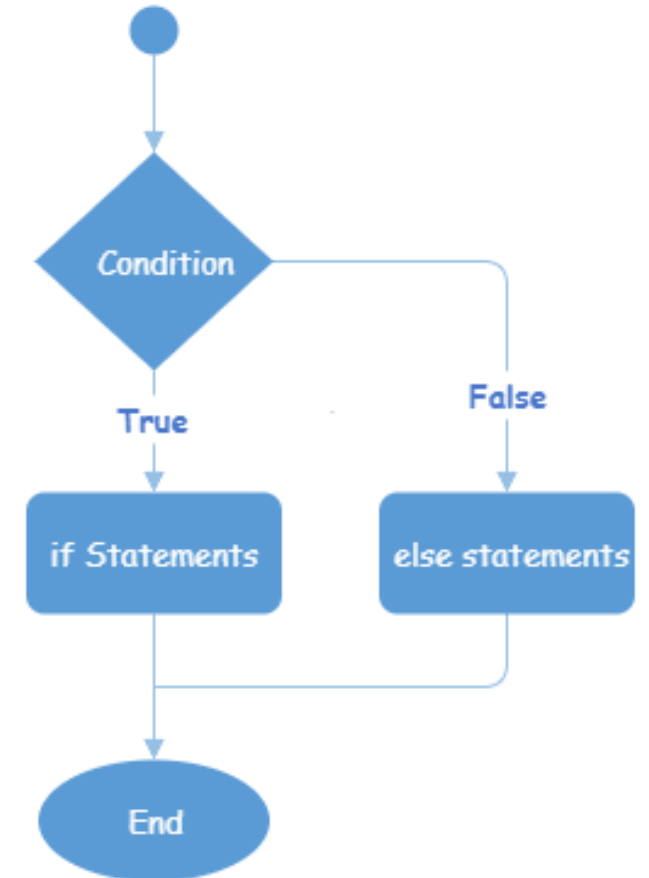
Visual Basic If Else Statement

In Visual Basic, If Else statement or condition is having an optional Else statements and the Else statements will be executed whenever the If condition fails to execute..

Generally in Visual Basic, If Else statement, whenever the boolean expression returns true, then the If statements will be executed otherwise the Else block of statements will be executed.

Syntax of Visual Basic If Else Statement

```
If boolean_expression Then
// Statements to Execute if boolean expression is True
Else
// Statements to Execute if boolean expression is False
End If
```



Visual Basic If Else Statement Example

Following is the example of defining the **If Else** statement in Visual Basic programming language to execute the block of code or statements based on a Boolean expression.

```
Private Sub Button3_Click(sender As Object, e As EventArgs) Handles  
Button3.Click  
    Dim x As Integer = 20  
    If x >= 10 Then  
        MsgBox("x is Greater than or Equals to 10")  
    Else  
        MsgBox("x is Less than 10")  
    End If  
    MsgBox("Press Enter Key to Exit..")  
  
End Sub
```

جامعة بغداد/ كلية التربية للعلوم الصرفة ابن الهيثم/قسم علوم الحاسبات
المرحلة الثالثة صباحي / مسائي (نظري)
السنة ٢٠٢١-٢٠٢٢
اساتذة المادة: شهلاء طالب

Visual Programming

Lecture 3 –Discussion Making

Discussion Making in VB

Generally, in Visual Basic the statement that needs to be executed based on the condition is known as a “**Conditional Statement**” and the statement is more likely a block of code.

Visual Basic If Statement

Syntax of Visual Basic if Statement

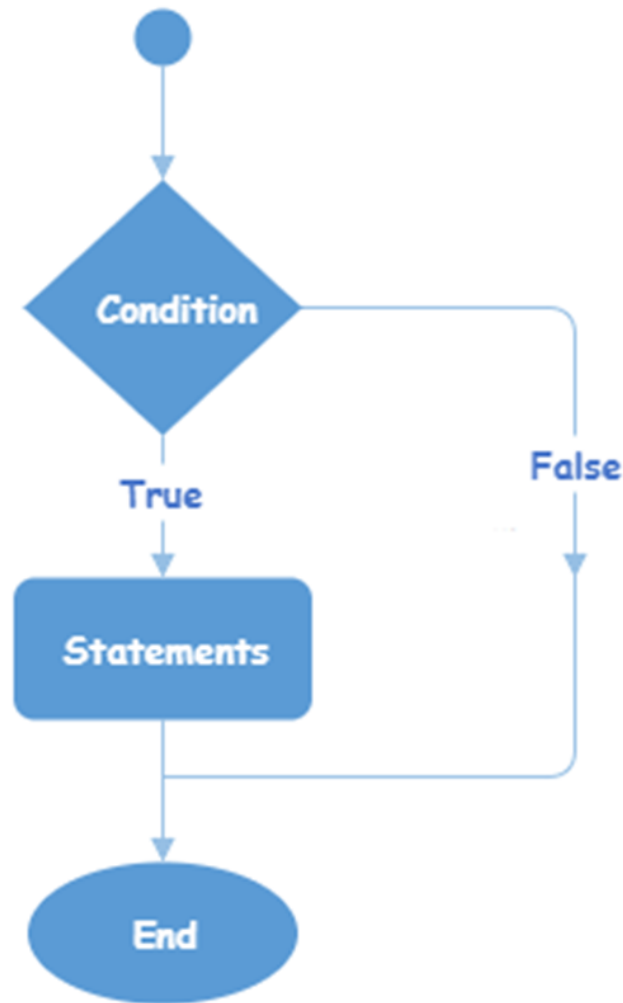
```
If bool_expression Then
```

```
// Statements to Execute if condition is true
```

```
End If
```

Visual Basic If Statement Flow Chart Diagram

Following is the flow chart diagram which will represent the process flow of **If statement** in Visual Basic programming language.



Visual Basic If Statement Example

Following is the example of defining the If statement in Visual Basic programming language to execute the block of code or statements based on a Boolean expression.

```
Private Sub Button2_Click(sender As Object, e As EventArgs)
Handles Button2.Click
    Dim x As Integer = 20, y As Integer = 10
    If x >= 10 Then
        MsgBox("x is Greater than 10")
    End If
    If y <= 5 Then
        MsgBox("y is less than or equals to 5")
    End If
    MsgBox("Press Enter Key to Exit..")
End Sub
```

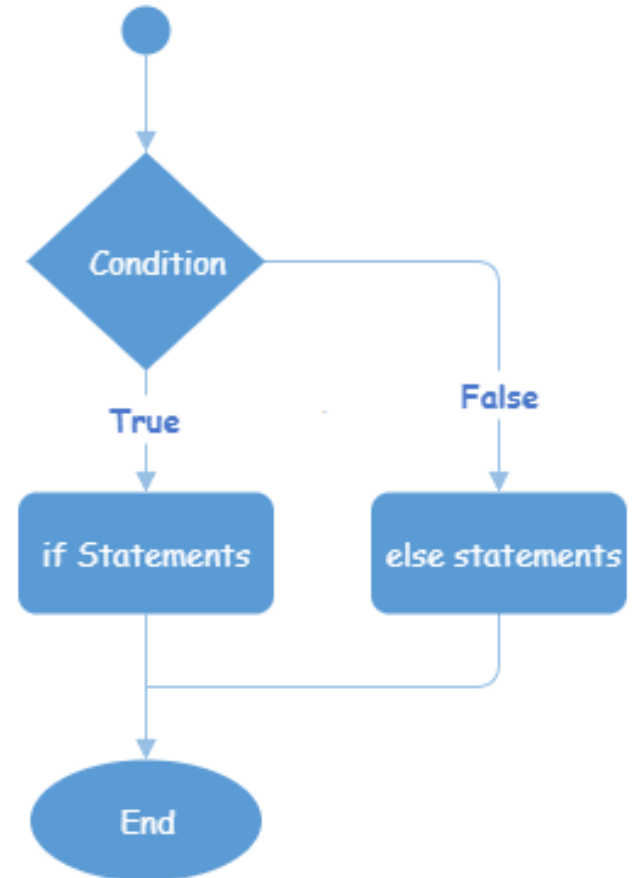
Visual Basic If Else Statement

In Visual Basic, If Else statement or condition is having an optional Else statements and the Else statements will be executed whenever the If condition fails to execute..

Generally in Visual Basic, If Else statement, whenever the boolean expression returns true, then the If statements will be executed otherwise the Else block of statements will be executed.

Syntax of Visual Basic If Else Statement

```
If boolean_expression Then
// Statements to Execute if boolean expression is True
Else
// Statements to Execute if boolean expression is False
End If
```



Visual Basic If Else Statement Example

Following is the example of defining the **If Else** statement in Visual Basic programming language to execute the block of code or statements based on a Boolean expression.

```
Private Sub Button3_Click(sender As Object, e As EventArgs) Handles
Button3.Click
    Dim x As Integer = 20
    If x >= 10 Then
        MsgBox("x is Greater than or Equals to 10")
    Else
        MsgBox("x is Less than 10")
    End If
    MsgBox("Press Enter Key to Exit..")

End Sub
```

Visual Basic If-Else-If Statement

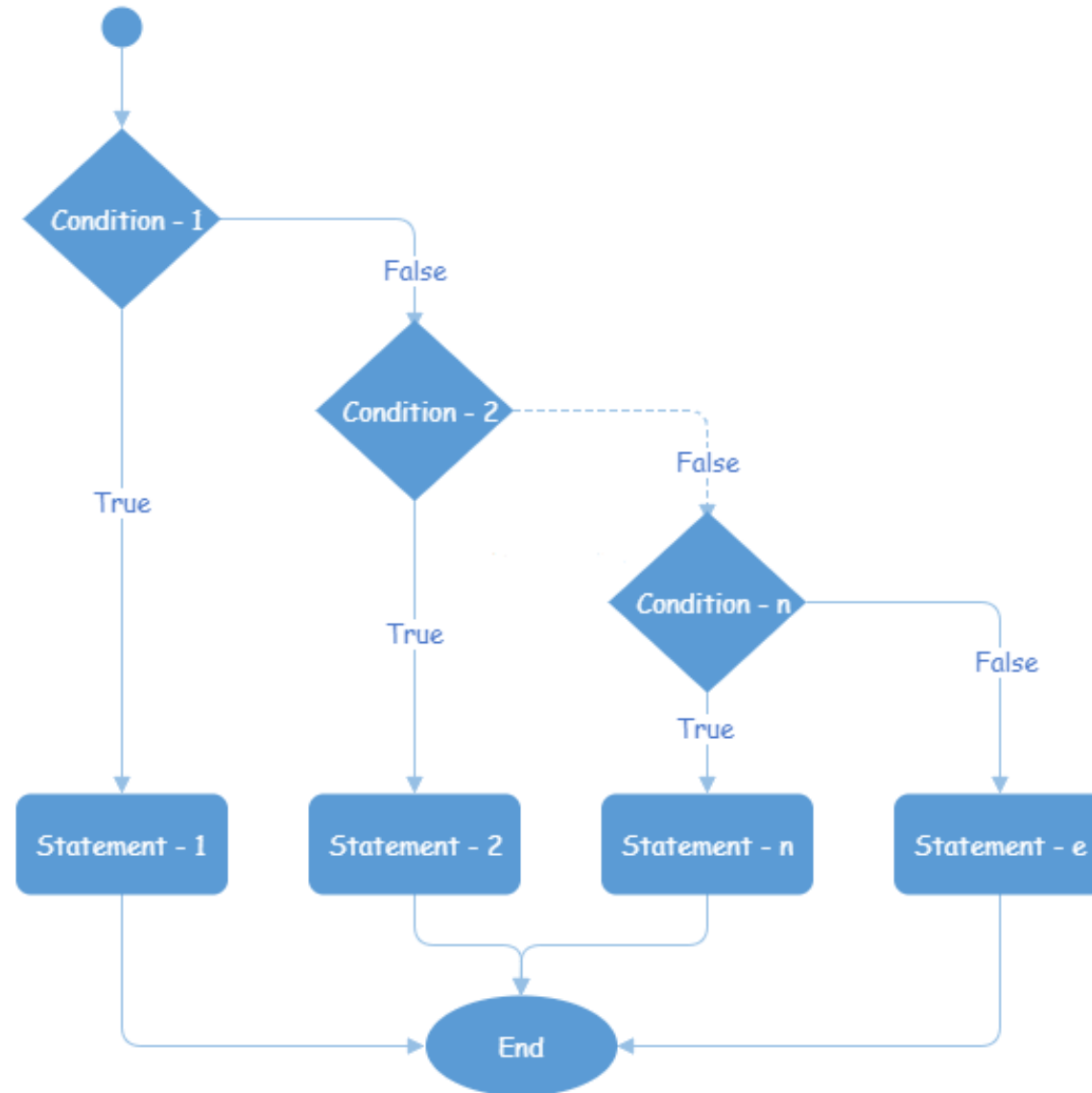
In Visual Basic, If-Else-If statement or condition is useful to define the multiple conditions and execute only the matched condition based on our requirements.

Generally, in Visual Basic if statement or if-else statement is useful when we have a one condition to validate and execute the required block of statements. In case, if we have a multiple conditions to validate and execute only one block of code, then If-Else-If statement is useful in our application.

Syntax of Visual Basic If-Else-If Statement

```
If condition_1 Then
// Statements to Execute if condition_1 is True
ElseIf condition_2 Then
// Statements to Execute if condition_2 is True
ElseIf condition_3 Then
// Statements to Execute if condition_3 is True
....
....
Else
// Statements to Execute if all conditions are False
End If
```

Visual Basic If-Else-If Statement Flow Chart



Visual Basic If-Else-If Statement Example

Following is the example of defining the **If-Else-If** statement in Visual Basic programming language to execute the block of code or statements based on the Boolean expression.

```
Private Sub Button4_Click(sender As Object, e As EventArgs) Handles
Button4.Click
    Dim x As Integer = 5
    If x = 10 Then
        MsgBox("x value equals to 10")
    ElseIf x > 10 Then
        MsgBox("x value greater than 10")
    Else
        MsgBox("x value less than 10")
    End If
    MsgBox("Press Enter Key to Exit..")

End Sub
```

Visual Basic Select Case Statement

In Visual Basic, **Select...Case** statement is useful to execute a single case statement from the group of multiple case statements based on the value of a defined expression.

By using **Select...Case** statement in Visual Basic, we can replace the functionality of [if...else if](#) statement to provide better readability for the code.

Visual Basic Select Case Statement Syntax

Generally, in Visual Basic the Select...Case statement is a collection of multiple case statements and it will execute only one single case statement based on the matching value of the defined expression.

```
Select Case variable/expresion
```

```
Case value1
```

```
// Statements to Execute
```

```
Case value2
```

```
//Statements to Execute
```

```
....
```

```
....
```

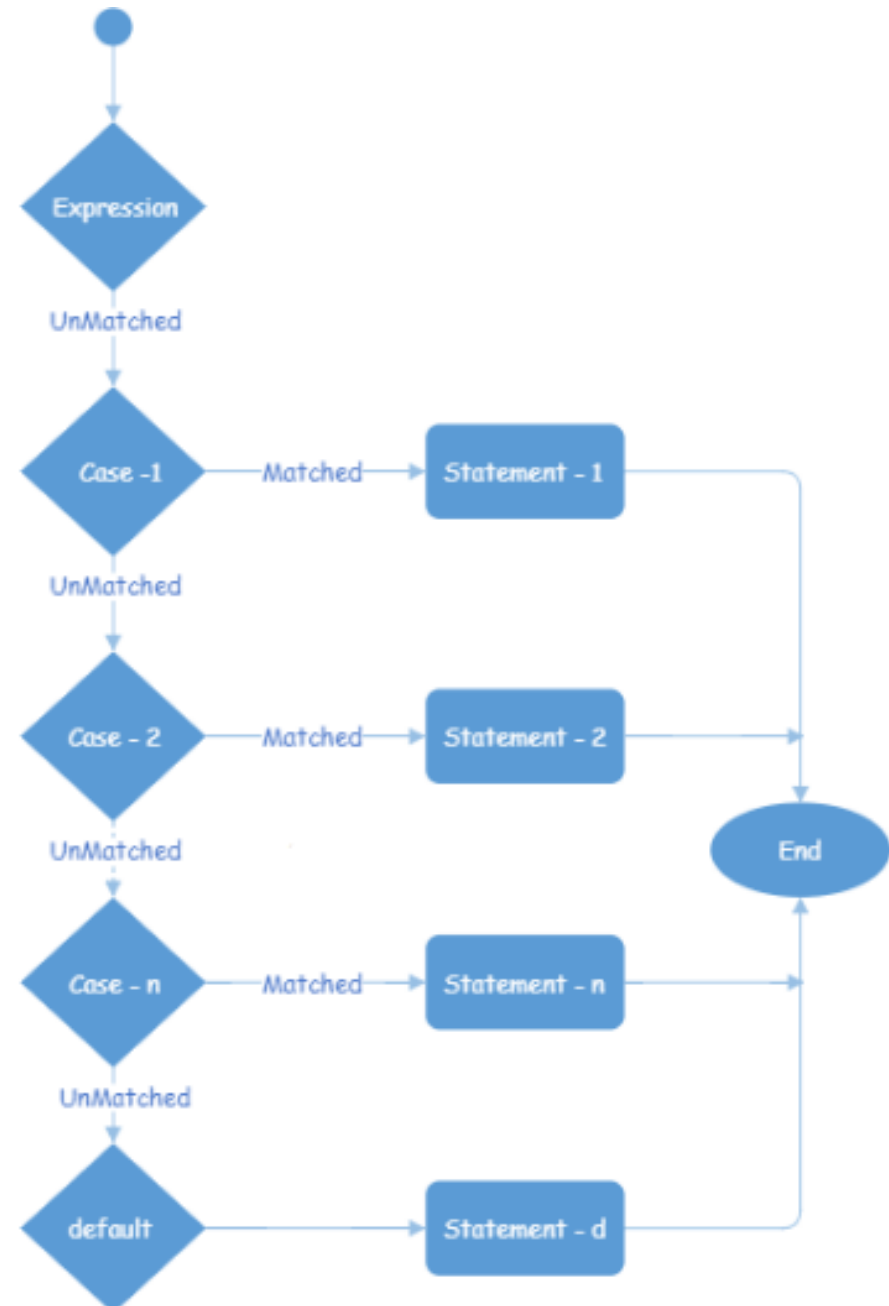
```
Case Else
```

```
// Statements to Execute if No Case Matches
```

```
End Select
```

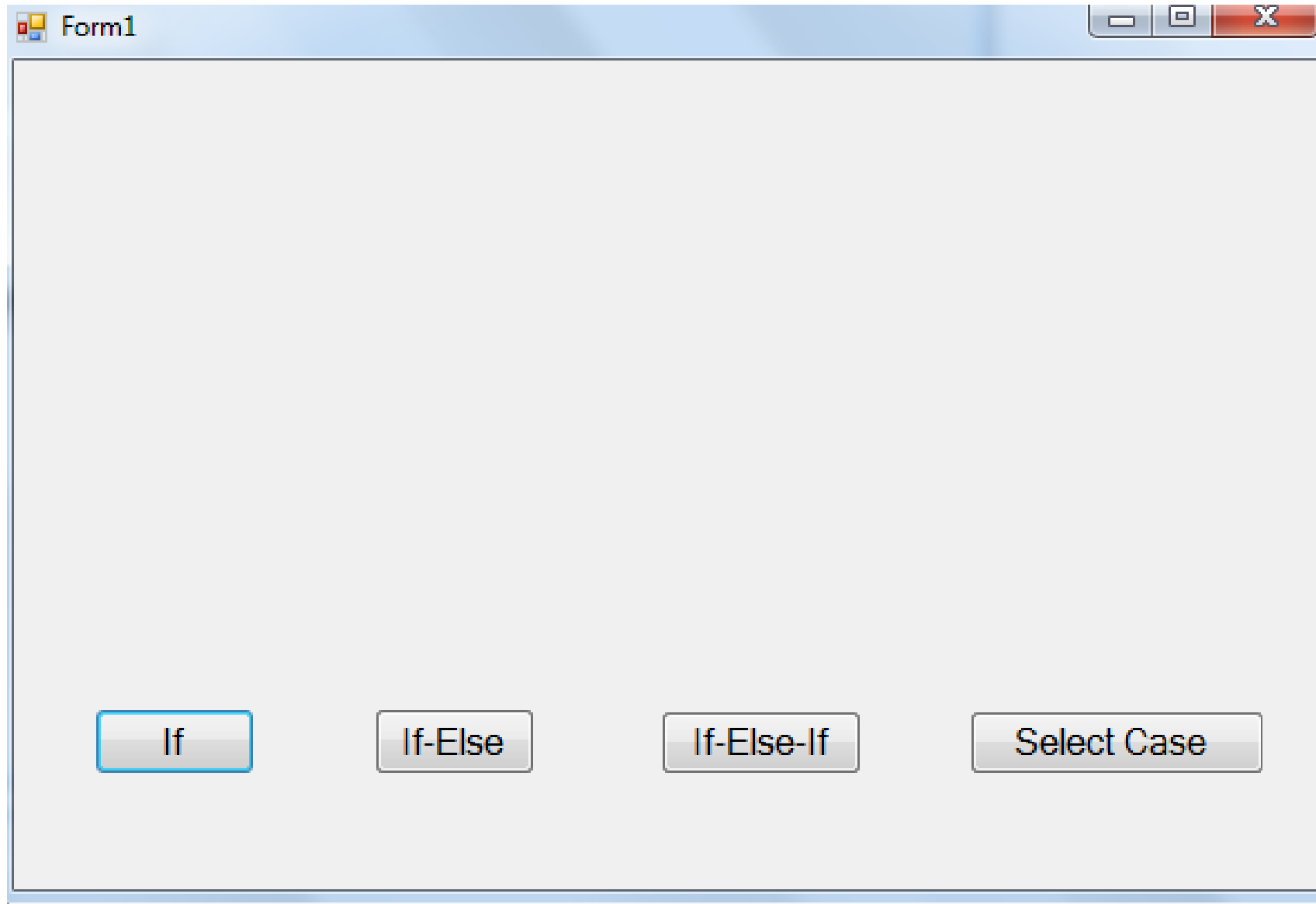
Visual Basic Select Case Statement Example

```
Private Sub Button5_Click(sender As Object,  
e As EventArgs) Handles Button5.Click  
    Dim x As Integer = 20  
    Select Case x  
        Case 10  
            MsgBox("x value is 10")  
        Case 15  
            MsgBox("x value is 15")  
        Case 20  
            MsgBox("x value is 20")  
        Case Else  
            MsgBox("Not Known")  
    End Select  
    MsgBox("Press Enter Key to Exit..")  
End Sub
```



Visual Basic Select Case Statement Flow Chart

Discussion Making in VB Program



جامعة بغداد / كلية التربية للعلوم الصرفة ابن الهيثم / قسم علوم الحاسبات
المرحلة الثالثة صباحي / مسائي (نظري)
السنة ٢٠٢١-٢٠٢٢
اساتذة المادة: شهلاء طالب

Visual Programming

Lecture 4 –Operators & Loops

Operators: An operator is a symbol that tells the compiler to perform specific mathematical or logical manipulations. VB.Net is rich in built-in operators and provides following types of commonly used operators –

- Arithmetic Operators
- Comparison Operators
- Assignment Operators
- Logical Operators

Arithmetic Operators

Following table shows all the arithmetic operators supported by VB.Net. Assume variable **A** holds 2 and variable **B** holds 7, then –

Operator	Description	Example
^	Raises one operand to the power of another	B^A will give 49
+	Adds two operands	$A + B$ will give 9
-	Subtracts second operand from the first	$A - B$ will give -5
*	Multiplies both operands	$A * B$ will give 14
/	Divides one operand by another and returns a floating point result	B / A will give 3.5
\	Divides one operand by another and returns an integer result	$B \setminus A$ will give 3
MOD	Modulus Operator and remainder of after an integer division	$B \text{ MOD } A$ will give 1

Comparison Operators

Following table shows all the comparison operators supported by VB.Net. Assume variable **A** holds 10 and variable **B** holds 20, then –

Operator	Description	Example
=	Checks if the values of two operands are equal or not; if yes, then condition becomes true.	(A = B) is not true.
<>	Checks if the values of two operands are equal or not; if values are not equal, then condition becomes true.	(A <> B) is true.
>	Checks if the value of left operand is greater than the value of right operand; if yes, then condition becomes true.	(A > B) is not true.
<	Checks if the value of left operand is less than the value of right operand; if yes, then condition becomes true.	(A < B) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand; if yes, then condition becomes true.	(A >= B) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand; if yes, then condition becomes true.	(A <= B) is true.

Logical Operators

Following table shows all the logical operators supported by VB.Net. Assume variable A holds Boolean value True and variable B holds Boolean value False, then –

Operator	Description	Example
And	It is the logical as well as bitwise AND operator. If both the operands are true, then condition becomes true. This operator does not perform short-circuiting, i.e., it evaluates both the expressions.	(A And B) is False.
Or	It is the logical as well as bitwise OR operator. If any of the two operands is true, then condition becomes true. This operator does not perform short-circuiting, i.e., it evaluates both the expressions.	(A Or B) is True.
Not	It is the logical as well as bitwise NOT operator. Use to reverses the logical state of its operand. If a condition is true, then Logical NOT operator will make false.	Not(A And B) is True.
Xor	It is the logical as well as bitwise Logical Exclusive OR operator. It returns True if both expressions are True or both expressions are False; otherwise it returns False. This operator does not perform short-circuiting, it always evaluates both expressions and there is no short-circuiting counterpart of this operator.	A Xor B is True.

Assignment Operators

There are following assignment operators supported by VB.Net –

Operator	Description	Example
=	Simple assignment operator, Assigns values from right side operands to left side operand	$C = A + B$ will assign value of $A + B$ into C
+=	Add AND assignment operator, It adds right operand to the left operand and assigns the result to left operand	$C += A$ is equivalent to $C = C + A$
-=	Subtract AND assignment operator, It subtracts right operand from the left operand and assigns the result to left operand	$C -= A$ is equivalent to $C = C - A$
*=	Multiply AND assignment operator, It multiplies right operand with the left operand and assigns the result to left operand	$C *= A$ is equivalent to $C = C * A$
/=	Divide AND assignment operator, It divides left operand with the right operand and assigns the result to left operand (floating point division)	$C /= A$ is equivalent to $C = C / A$
\=	Divide AND assignment operator, It divides left operand with the right operand and assigns the result to left operand (Integer division)	$C \setminus = A$ is equivalent to $C = C \setminus A$
^=	Exponentiation and assignment operator. It raises the left operand to the power of the right operand and assigns the result to left operand.	$C^=A$ is equivalent to $C = C ^ A$

The For...Next Loop

The syntax is:

For counter=startNumber to endNumber (Step increment)

One or more VB statements

Next

```
Dim counter As Integer
For counter = 1 To 10
    Console.WriteLine(counter)
Next
```

1
2
3
4
5
6
7
8
9
10

```
Dim counter, sum As Integer
For counter = 1 To 100 Step 10
    sum += counter
    Console.WriteLine(sum)
Next
```

1
12
33
64
105
156
217
288
369
460

Example 1: Write a program to print (hello) five times.

Sol:

```
Private Sub Button3_Click(sender As Object, e As  
EventArgs) Handles Button3.Click  
    For i = 1 To 5  
        Console.WriteLine("hello")  
    Next i  
  
End Sub
```

```
hello  
hello  
hello  
hello  
hello
```

Example 2: Write a program to print even numbers from 1 to 10.

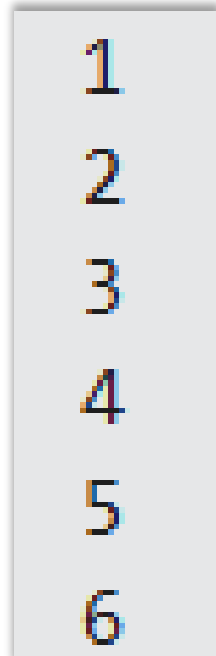
Sol:

```
Private Sub Button4_Click(sender As Object, e  
As EventArgs) Handles Button4.Click  
    For i = 2 To 10 Step 2  
        Console.WriteLine(i)  
    Next i  
  
End Sub
```

```
2  
4  
6  
8  
10
```


To exit a For....Next Loop you can place the Exit For statement within the loop; and it is normally used together with the If...Then...statement

```
Private Sub Button5_Click(sender As Object,  
e As EventArgs) Handles Button5.Click  
    Dim n As Integer  
    For n = 1 To 10  
        If n > 6 Then  
            Exit For  
        Else  
            Console.WriteLine(n)  
        End If  
    Next  
End Sub
```



1
2
3
4
5
6

The Do Loop

The Do Loop syntaxes are

a)

Do While condition

Block of one or more Visual Basic 2012 statements

Loop

b)

Do

Block of one or more Visual Basic 2012 statements

Loop While condition

c)

Do Until condition

Block of one or more Visual Basic 2012 statements

Loop

d)

Do

Block of one or more Visual Basic 2012 statements

Loop Until condition

1. Do While Loop Example

```
Private Sub Button6_Click(sender As Object,  
e As EventArgs) Handles Button6.Click  
    Dim counter As Integer  
    Do While counter <= 1000  
        Console.WriteLine(counter)  
        counter += 100  
    Loop  
End Sub
```

```
0  
100  
200  
300  
400  
500  
600  
700  
800  
900  
1000
```

Write a program to print (hello) five times with its numbering using do while loop.

```
Private Sub Button8_Click(sender As Object,  
e As EventArgs) Handles Button8.Click  
    Dim i As Integer  
    i = 1  
    Do While i <= 5  
        Console.WriteLine("hello")  
        i = i + 1  
    Loop  
  
End Sub
```

```
hello  
hello  
hello  
hello  
hello
```

Write a program to print even numbers from 1 to 10.

```
Private Sub Button9_Click(sender As Object, e
As EventArgs) Handles Button9.Click
    Dim i As Integer
    i = 2
    Do While i <= 10
        Console.WriteLine(i)
        i = i + 2
    Loop
End Sub
```



2
4
6
8
10

2. Do Loop Example

```
Private Sub Button7_Click(sender As Object, e As  
EventArgs) Handles Button7.Click
```

```
    Dim sum, n As Integer
```

```
    Console.WriteLine("n" & vbTab & "Sum")
```

```
    Console.WriteLine("-----  
-")
```

```
    Do
```

```
        n += 1
```

```
        sum += n
```

```
        Console.WriteLine(n & vbTab & sum)
```

```
        If n = 100 Then
```

```
            Exit Do
```

```
        End If
```

```
    Loop
```

```
End Sub
```

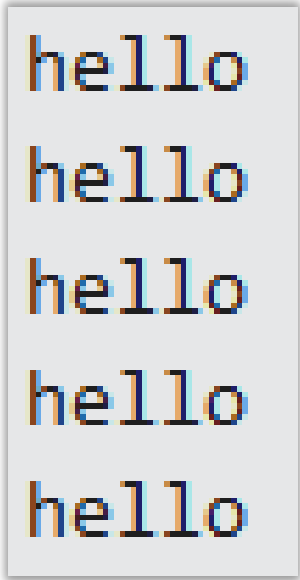
n	Sum
1	1
2	3
3	6
4	10
5	15
6	21
7	28
8	36
9	45
10	55
11	66
12	78
13	91
14	105
15	120
16	136
17	153
18	171
19	190
20	210
21	231
22	253
23	276
24	300
25	325
26	351
27	378
28	406
29	435
30	465
31	496
32	528
33	561
34	595
35	630
36	666
37	703
38	741
39	780

40	820
41	861
42	903
43	946
44	990
45	1035
46	1081
47	1128
48	1176
49	1225
50	1275
51	1326
52	1378
53	1431
54	1485
55	1540
56	1596
57	1653
58	1711
59	1770
60	1830
61	1891
62	1953
63	2016
64	2080
65	2145
66	2211
67	2278
68	2346
69	2415
70	2485
71	2556
72	2628
73	2701
74	2775
75	2850
76	2926
77	3003
78	3081
79	3160
80	3240
81	3321
82	3403
83	3486
84	3570

85	3655
86	3741
87	3828
88	3916
89	4005
90	4095
91	4186
92	4278
93	4371
94	4465
95	4560
96	4656
97	4753
98	4851
99	4950
100	5050

3. Do Until Loop.

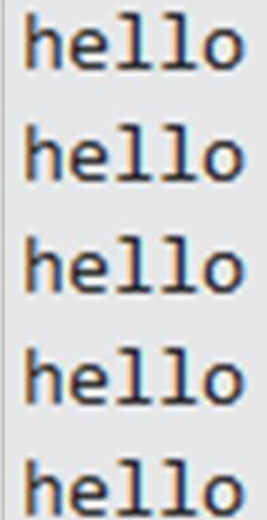
```
Private Sub Button10_Click(sender As Object, e
As EventArgs) Handles Button10.Click
    Dim i As Integer
    i = 1
    Do Until i > 5
        Console.WriteLine("hello")
        i = i + 1
    Loop
End Sub
```



```
hello
hello
hello
hello
hello
```

4. Do Loop Until.

```
Private Sub Button11_Click(sender As Object, e As  
EventArgs) Handles Button11.Click  
    Dim i As Integer  
    i = 1  
    Do  
        Console.WriteLine("hello")  
        i = i + 1  
    Loop Until i > 5  
End Sub
```



hello
hello
hello
hello
hello

جامعة بغداد / كلية التربية للعلوم الصرفة ابن الهيثم / قسم علوم الحاسبات
المرحلة الثالثة صباحي / مسائي (نظري)
السنة ٢٠٢١-٢٠٢٢
اساتذة المادة: شهلاء طالب

Visual Programming

Lecture 5 – Basic Controls

Basic Controls

VB.Net provides a huge variety of controls that help you to create rich user interface. Functionalities of all these controls are defined in the respective control classes. The control classes are defined in the **System.Windows.Forms** namespace.

The following table lists some of the commonly used controls –

Sr.No.	Widget & Description
1	Forms ↗ The container for all the controls that make up the user interface.
2	TextBox ↗ It represents a Windows text box control.
3	Label ↗ It represents a standard Windows label.
4	Button ↗ It represents a Windows button control.
5	ListBox ↗ It represents a Windows control to display a list of items.
6	ComboBox ↗ It represents a Windows combo box control.

7	<p>RadioButton ↗</p> <p>It enables the user to select a single option from a group of choices when paired with other RadioButton controls.</p>
8	<p>CheckBox ↗</p> <p>It represents a Windows CheckBox.</p>
9	<p>PictureBox ↗</p> <p>It represents a Windows picture box control for displaying an image.</p>
10	<p>ProgressBar ↗</p> <p>It represents a Windows progress bar control.</p>
11	<p>ScrollBar ↗</p> <p>It Implements the basic functionality of a scroll bar control.</p>
12	<p>DateTimePicker ↗</p> <p>It represents a Windows control that allows the user to select a date and a time and to display the date and time with a specified format.</p>

Controls Properties

Forms and controls have **properties**, **events**, and **methods**. Together they make the forms and controls useful for programmers.

You can change the appearance of the controls (and form) by setting their properties in the properties window.

Here is a shortlist of the properties we use in the course:

Property Name	Objective	Code	Stage of changing
Text	String appear in title of control	TextBox1.Text = "any text" Label1.Text = "any text" Button1.Text = "any text"	Design and Run
MultiLine	To enter more than one line in TextBox1 only	True or False	Design
BackColor	Background color for control.	TextBox1.BackColor = Color.anycolor Label1.BackColor = Color.anycolor Button1.BackColor = Color.anycolor	Design and Run
ForeColor	Color of text written on control.	TextBox1.ForeColor = Color.anycolor Label1.ForeColor = Color.anycolor Button1.ForeColor = Color.anycolor	Design and Run

TextAlign	The horizontal and vertical alignment of the text inside the control.	Left/Right/Center From properties	Design and Run
Left Top	Horizontal/vertical position of the control, counted by the number of pixels relative to the left/top side of its parent.	TextBox1.Left = no, TextBox1.Top = no Label1.Left= no, label1.Top = no Button1.Left= no , Button1.Top = no	Run
Width Height	The width or height of the control, counted by the number of pixels.	TextBox1.Width = no, TextBox1.Hight = no Label1.Width= no, label1.Hight= no Button1.Width= no , Button1.Hight = no	Run
Hide	To hide the control	TextBox1.Hide() Label1.Hide() Button1.Hide()	Run
Visible	The control appear or disappear	TextBox1 .Visible = True or False Label1. Visible = True or False Button1.Visible = True or False	Design and Run
Enabled	The control enable or disable	TextBox1.Enabled = True or False Label1. Enabled= True or False Button1. Enabled= True or False	Design and Run

Here is an example program that changes some properties of the form and the controls. To enter the code into Visual Basic IDE, you can double click Button1 in design view. Can you guess what will happen after Button1 is clicked?

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

```
    Me.Text = "Button1 Pressed!"
```

```
    Me.BackColor = Color.Pink
```

```
    Label1.Text = "Name"
```

```
    Label1.BackColor = Color.Green
```

```
    Label1.ForeColor = Color.Yellow
```

```
    Label1.Top = 80
```

```
    TextBox1.Text = "Sarah"
```

```
    TextBox1.BackColor = Color.Red
```

```
    TextBox1.Enabled = False
```

```
    TextBox1.Left = 20
```

```
    Button1.Visible = False
```

```
End Sub
```

Before

A screenshot of a Windows application window titled "Form1". The window has a light blue title bar with standard minimize, maximize, and close buttons. The main content area is light gray and contains a label "Label1" on the left, an empty text input field in the center, and a button labeled "Button1" at the bottom.

After

A screenshot of a Windows application window titled "Button1 Pressed!". The window has a light blue title bar with standard minimize, maximize, and close buttons. The main content area has a light pink background. It displays a green label "Name" above a red text input field containing the text "Sarah".

Exercise 1:

Write a program with two controls: Button1 and TextBox1. When Button1 is clicked, the following things should happen:

- (a) TextBox1 is disabled,
- (b) The background color of TextBox1 becomes yellow,
- (c) Button1 becomes visible, and
- (d) The form's background color becomes white.

Exercise 2:

Identify the mistakes in the following source code. There is one mistake in each line.

(Note: There are no mistakes with the words Me, Label1, Button1 and TextBox1.)

```
Me.Title = "Title of the form"
```

```
Label1.BackColor = Colour.Green
```

```
Button1.Visible = Ture
```

```
TextBox1.Enable = False
```

```
TextBox1.Text = Very good
```

CheckBox and RadioButton

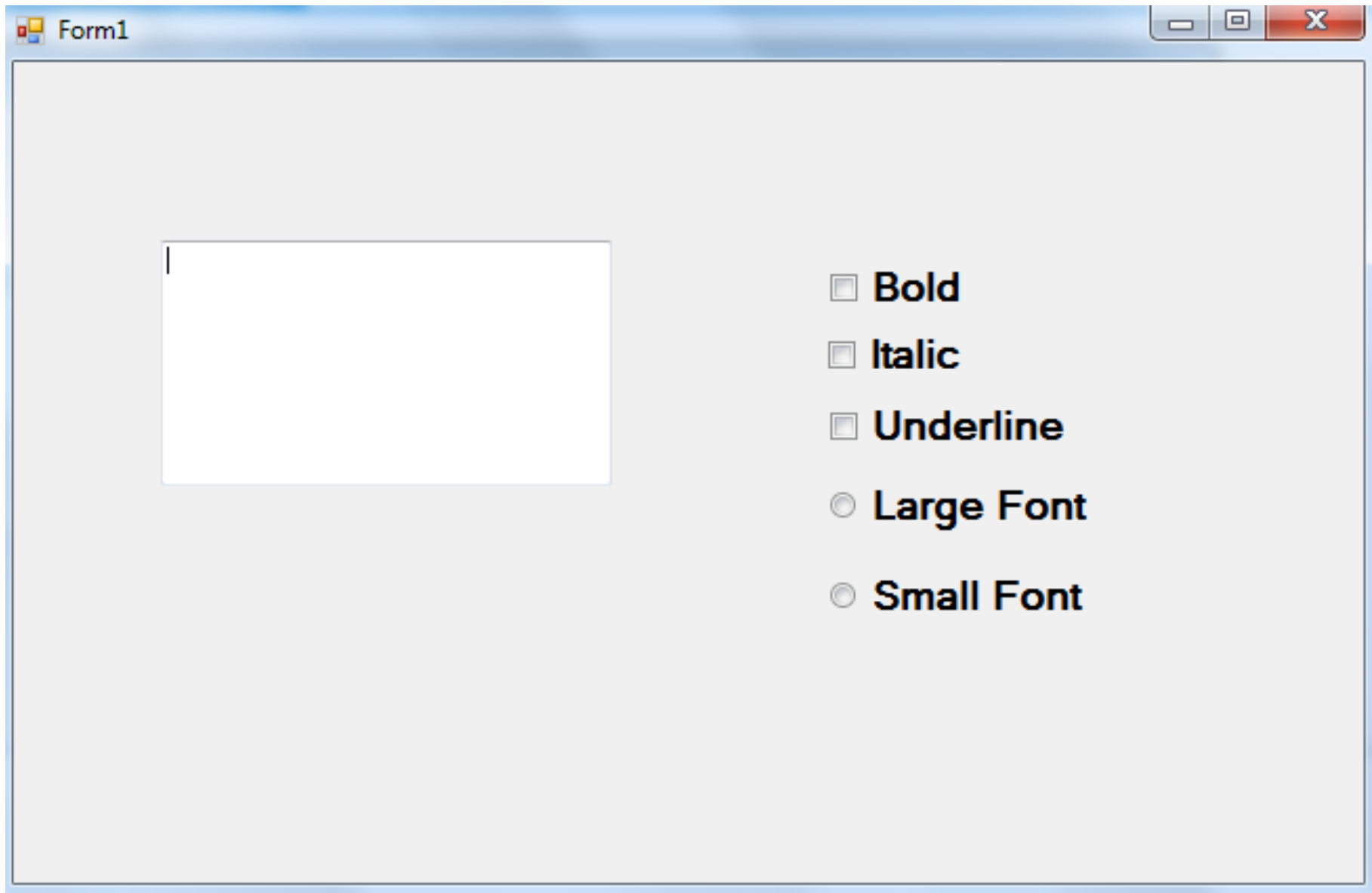
The checkbox is a control that allows the user to select multiple items.

The radio button is another control in Visual Basic 2012 that allows selection of choices. However, it operates differently from the CheckBox. While the CheckBoxes allow the user to select one or more items, radio buttons are mutually exclusive, which means the user can only choose one item only out of a number of choices.

Example

In this example, the user can enter text into a TextBox and format the font using the three CheckBoxes that represent bold, italic and underline. Also change the size of text into a TextBox by using two RadioButtons.

Form1



A window titled "Form1" with a standard Windows-style title bar (minimize, maximize, close buttons). The main area contains a text box on the left and a list of font formatting options on the right.

Bold

Italic

Underline

Large Font

Small Font

```
Public Class Form1
```

```
Private Sub CheckBox1_CheckedChanged(sender As Object, e As EventArgs) Handles CheckBox1.CheckedChanged
```

```
    If CheckBox1.Checked Then
```

```
        TextBox1.Font = New Drawing.Font("Times New Roman", 20, FontStyle.Bold)
```

```
    End If
```

```
End Sub
```

```
Private Sub CheckBox2_CheckedChanged(sender As Object, e As EventArgs) Handles CheckBox2.CheckedChanged
```

```
    If CheckBox2.Checked Then
```

```
        TextBox1.Font = New Drawing.Font("Times New Roman", 20, FontStyle.Italic)
```

```
    End If
```

```
End Sub
```

```
Private Sub CheckBox3_CheckedChanged(sender As Object, e As EventArgs) Handles CheckBox3.CheckedChanged
```

```
    If CheckBox3.Checked Then
```

```
        TextBox1.Font = New Drawing.Font("Times New Roman", 20, FontStyle.Underline)
```

```
    End If
```

```
End Sub
```

```
Private Sub RadioButton1_CheckedChanged(sender As Object, e As EventArgs) Handles RadioButton1.CheckedChanged
```

```
    TextBox1.Font = New Drawing.Font("Times New Roman", 20)
```

```
End Sub
```

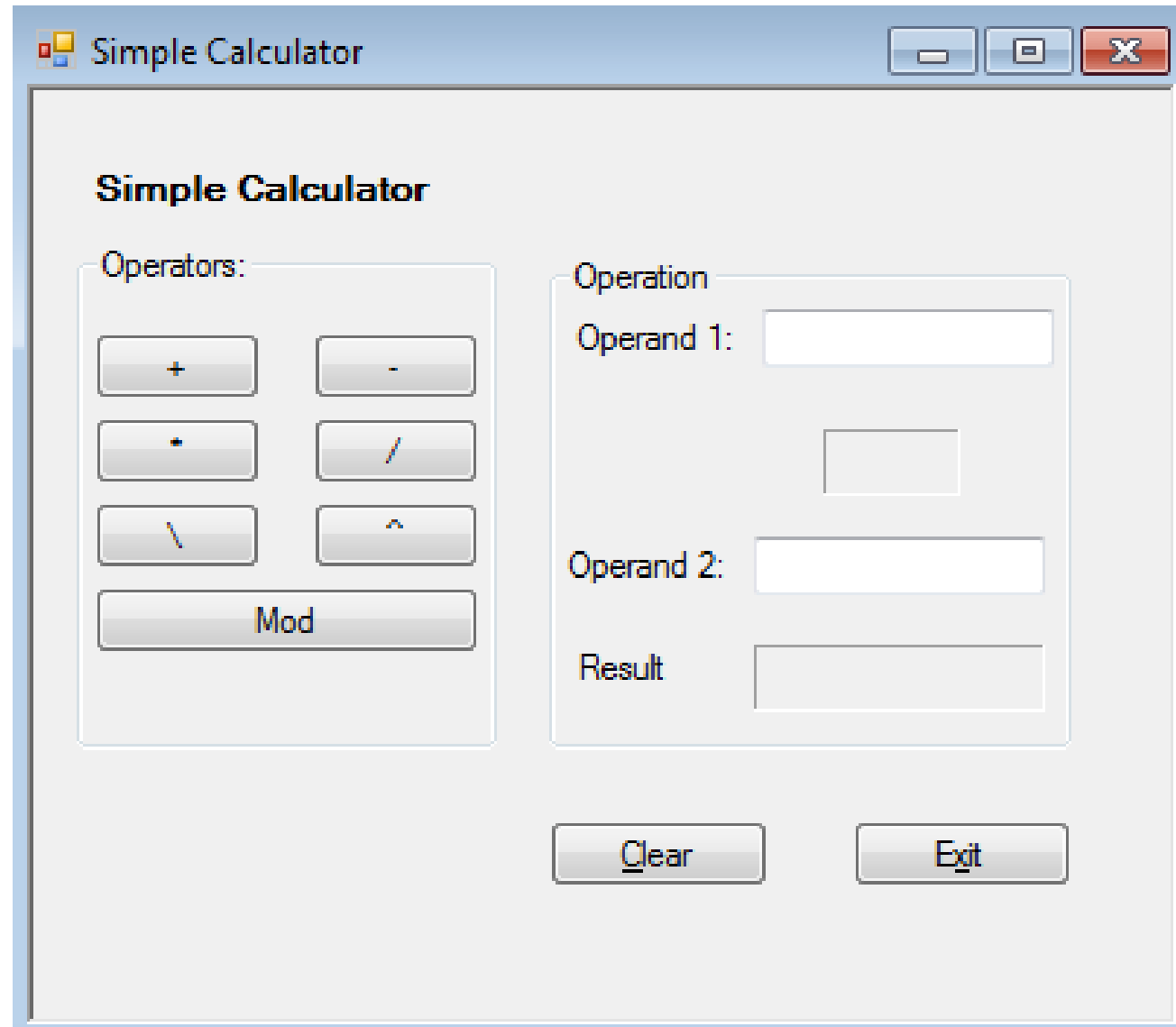
```
Private Sub RadioButton2_CheckedChanged(sender As Object, e As EventArgs) Handles RadioButton2.CheckedChanged
```

```
    TextBox1.Font = New Drawing.Font("Times New Roman", 10)
```

```
End Sub
```

```
End Class
```

Visual Basic Calculator



جامعة بغداد / كلية التربية للعلوم الصرفة ابن الهيثم / قسم علوم الحاسبات
المرحلة الثالثة صباحي / مسائي (نظري)
السنة ٢٠٢١-٢٠٢٢
اساتذة المادة: شهلاء طالب

Visual Programming

Lecture 6 – InputBox Function & ListBox Control

The InputBox() Function

An InputBox() function will display a message box where the user can enter a value or a message in the form of text.

myMessage=InputBox(Prompt, Title, default text, x-position, y-position)

myMessage is a variant data type but typically it is declared as string, which accept the message input by the users. The arguments are explained as follows:

Prompt - the message displayed normally as a question asked.

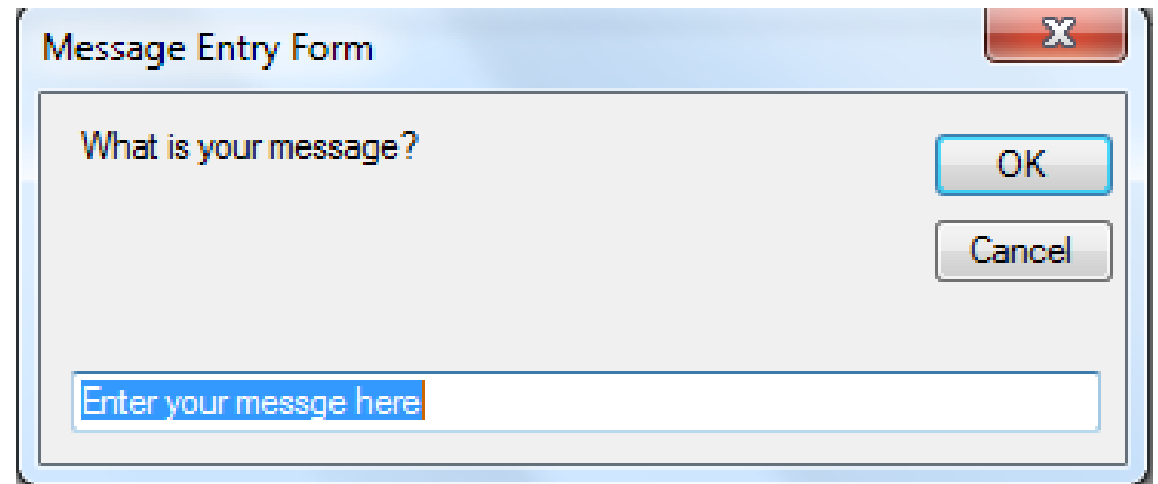
Title - The title of the Input Box.

default-text - The default text that appears in the input field where users can use it as his intended input or he may change to the message he wish to enter.

x-position and y-position - the position or the coordinates of the input box.

Example:

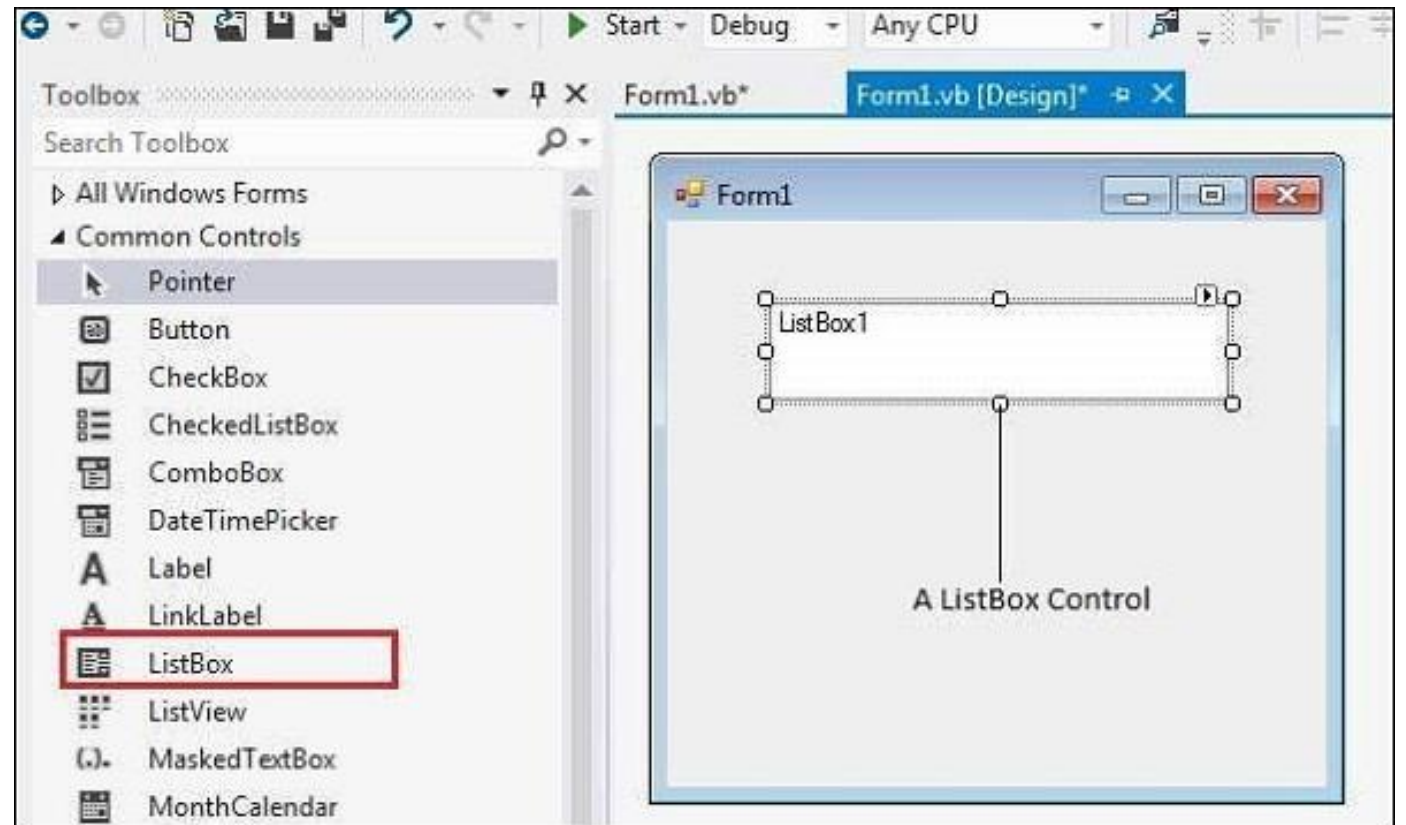
```
Private Sub Button1_Click(sender As Object, e As EventArgs)
Handles Button1.Click
    Dim userMsg As String
    userMsg = Microsoft.VisualBasic.InputBox("What is your
message?", "Message Entry Form", "Enter your messge here", 200,
300)
    If userMsg <> "" Then
        MessageBox.Show(userMsg)
    Else
        MessageBox.Show("No Message")
    End If
End Sub
```



ListBox Control

The ListBox represents a Windows control to display a list of items to a user. A user can select an item from the list. It allows the programmer to add items at design time by using the properties window or at the runtime.

You can populate the list box items either from the properties window or at runtime. To add items to a ListBox, select the ListBox control and get to the properties window, for the properties of this control. Click the (Collection) button next to the Items property. This opens the String Collection Editor dialog box, where you can enter the values one at a line.



Properties of the ListBox Control

The following are some of the commonly used properties of the ListBox control –

Methods	Description	Example
Add Item	Add an item to the ListBox &ComboBox	<code>ListBox.Item.Add("Text")</code> <code>ComboBox . Item.Add("Text")</code>
Remove Item	Removes the specified item from the ListBox &ComboBox	<code>ListBox1.Items.Remove("Text")</code> <code>ComboBox1.Item.Remove("Text")</code>
Clear	Removes all items from the ListBox &ComboBox	<code>ListBox1.Items.Clear()</code> <code>ComboBox1.Items.Clear()</code>
Select Item	Specifies the selected item in the ListBox &ComboBox.	<code>ListBox1.SelectedItem.ToString()</code>
Sorted	Boolean. Specifies whether the ListBox &ComboBox items are sorted or not.	<code>ListBox1.Sorted = True</code> <code>ComboBox1.Sorted= True</code>
ListCount	Integer. Contains the number of drop-down list items	<code>ListBox1.Items.Count</code> <code>ComboBox1.Items.Count</code>
MultiColumn	Creating multi-column in ListBox&ComboBox	<code>ListBox1.MultiColumn = True</code>

Events of the ListBox Control

The following are some of the commonly used events of the ListBox control :

Sr.No.	Event & Description
1	Click Occurs when a list box is selected.
2	SelectedIndexChanged Occurs when the SelectedIndex property of a list box is changed.

Example 1

In the following example, let us add a list box at design time and add items on it at runtime.

Take the following steps –

Drag and drop two labels, a button and a ListBox control on the form.

Set the Text property of the first label to provide the caption "Choose your favorite destination for higher studies".

Set the Text property of the second label to provide the caption "Destination". The text on this label will change at runtime when the user selects an item on the list.

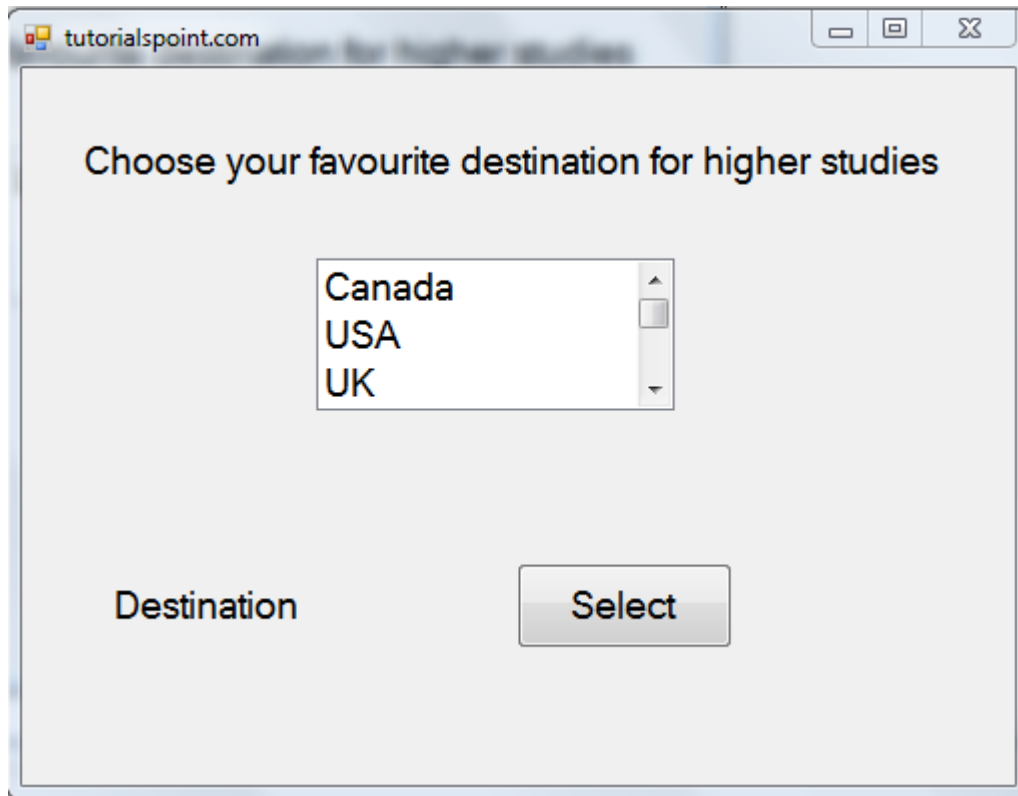
Click the listbox and the button controls to add the following codes in the code editor.

```
Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
    Me.Text = "tutorialspoint.com"
    ListBox1.Items.Add("Canada")
    ListBox1.Items.Add("USA")
    ListBox1.Items.Add("UK")
    ListBox1.Items.Add("Japan")
    ListBox1.Items.Add("Russia")
    ListBox1.Items.Add("China")
    ListBox1.Items.Add("India")
End Sub
```

```
Private Sub Button2_Click(sender As Object, e As EventArgs) Handles
Button2.Click
    MsgBox("You have selected " + ListBox1.SelectedItem.ToString())
End Sub
```

```
Private Sub ListBox1_SelectedIndexChanged(sender As Object, e As EventArgs)
Handles ListBox1.SelectedIndexChanged
    Label2.Text = ListBox1.SelectedItem.ToString()
End Sub
```

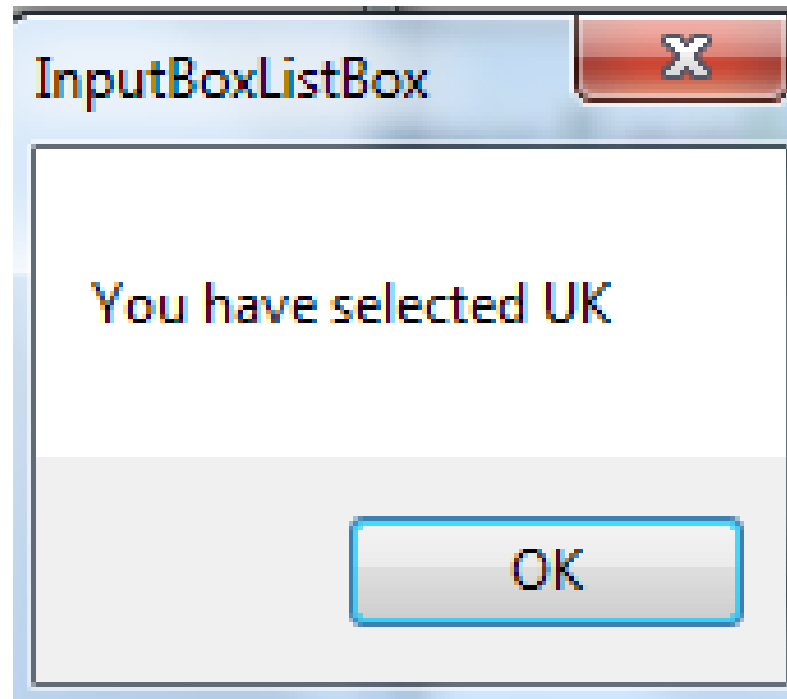
When the above code is executed and run using Start button available at the Microsoft Visual Studio tool bar, it will show the following window –



When the user chooses a destination, the text in the second label changes –



Clicking the Select button displays a message box with the user's choice –



Example 2

In this example, we will fill up a list box with items, retrieve the total number of items in the list box, sort the list box, remove some items and clear the entire list box.

Design the Form –

The screenshot shows a Windows form titled "Form1" with a standard title bar (minimize, maximize, close buttons). The form has a light gray background and contains the following elements:

- Section Header:** "Wish List for 2021" centered at the top.
- List Box:** A rectangular list box on the left side, labeled "ListBox1" in its top-left corner. It is currently empty.
- Buttons:** Five buttons are arranged on the right side of the form:
 - "Fill": A button with a gradient background, positioned to the right of the list box.
 - "Sort": A button with a gradient background, positioned below "Fill".
 - "Clear": A button with a gradient background, positioned below "Sort".
 - "Count": A button with a gradient background, positioned below the list box.
 - "Remove Items": A button with a gradient background, positioned to the right of "Count".
- Labels:** Two labels are positioned at the bottom of the form:
 - "Display total items": Located below the "Count" button.
 - "Your Selection": Located below the "Remove Items" button.


```
Public Class Form1
```

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

```
        ' Set the caption bar text of the form.
```

```
        Me.Text = "tutorialspoint.com"
```

```
        ' creating multi-column and multiselect list box
```

```
        ListBox1.MultiColumn = True
```

```
        ListBox1.SelectionMode =
```

```
SelectionMode.MultiExtended
```

```
    End Sub
```

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

```
        'populates the list
```

```
        ListBox1.Items.Add("Safety")
```

```
        ListBox1.Items.Add("Security")
```

```
        ListBox1.Items.Add("Governance")
```

```
        ListBox1.Items.Add("Good Music")
```

```
        ListBox1.Items.Add("Good Movies")
```

```
        ListBox1.Items.Add("Good Books")
```

```
        ListBox1.Items.Add("Education")
```

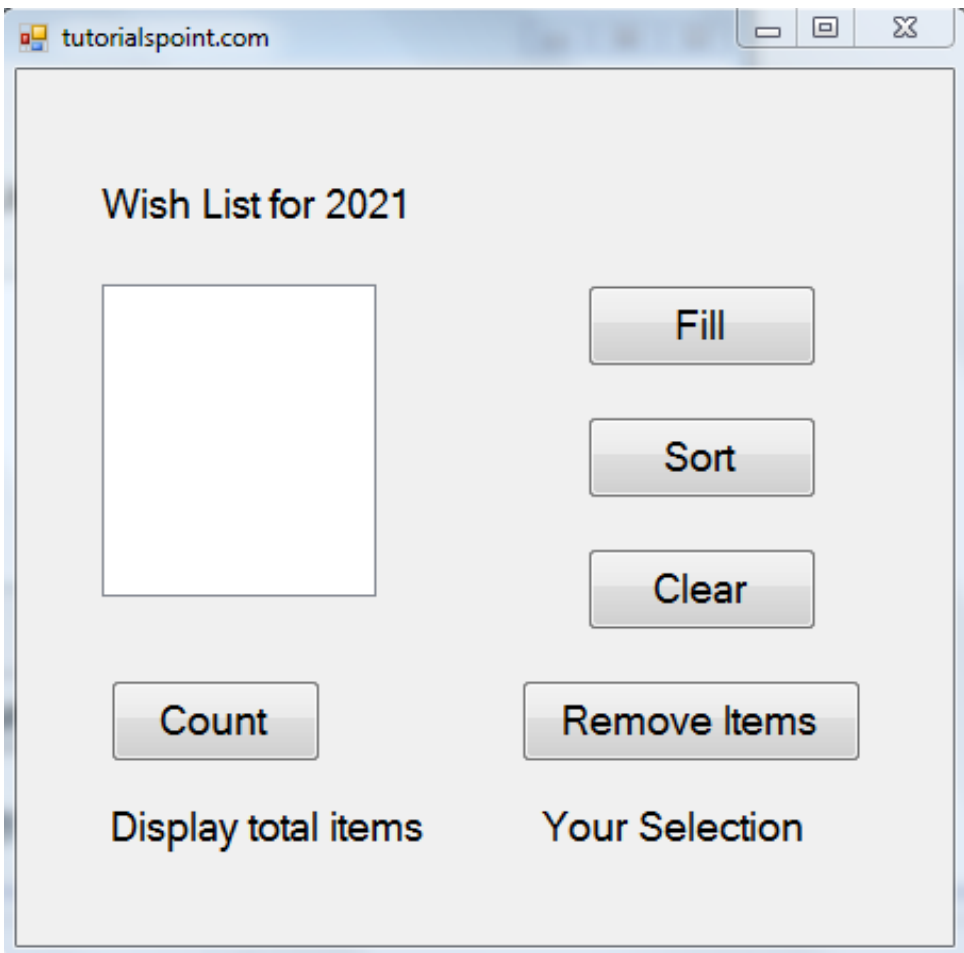
```
        ListBox1.Items.Add("Roads")
```

```
        ListBox1.Items.Add("Health")
```

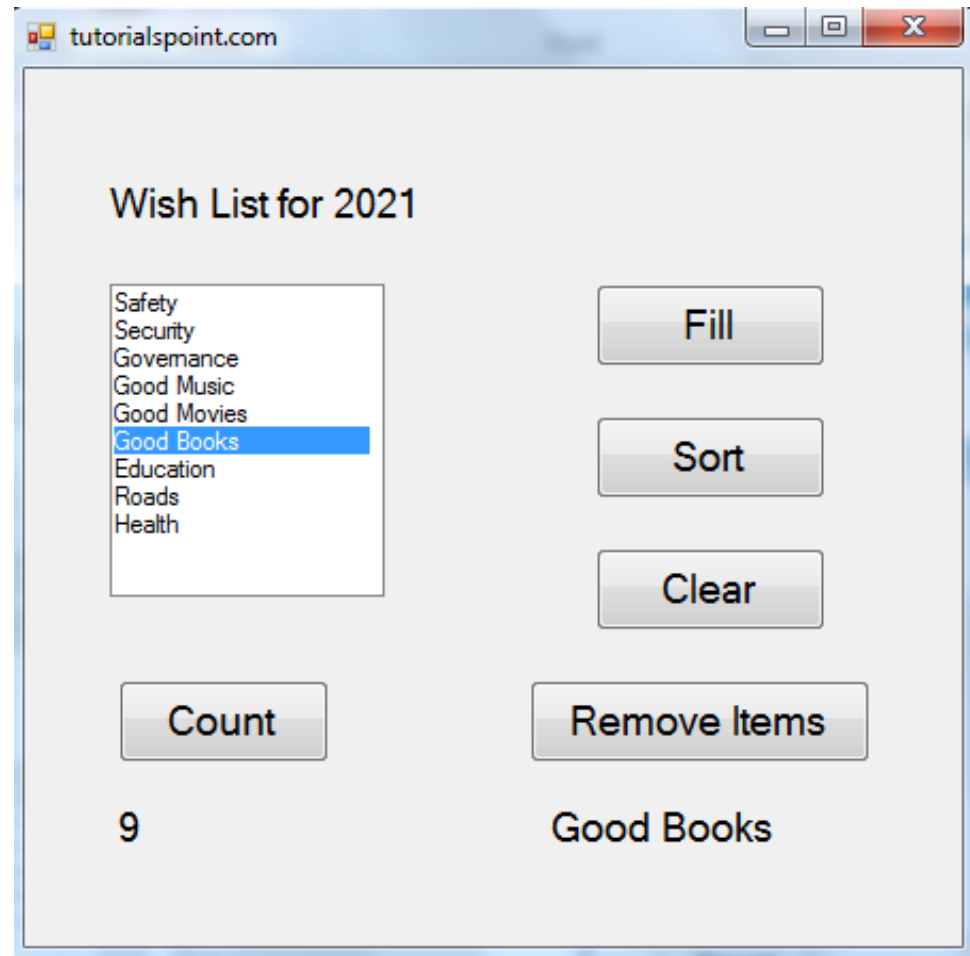
```
    End Sub
```

```
Private Sub Button2_Click(sender As Object, e As EventArgs) Handles Button2.Click
    ListBox1.Sorted = True
End Sub
Private Sub Button3_Click(sender As Object, e As EventArgs) Handles Button3.Click
    ListBox1.Items.Clear()
End Sub
Private Sub Button5_Click(sender As Object, e As EventArgs) Handles Button5.Click
    ListBox1.Items.RemoveAt(ListBox1.SelectedIndex())
End Sub
Private Sub Button4_Click(sender As Object, e As EventArgs) Handles Button4.Click
    Label1.Text = ListBox1.Items.Count
End Sub
Private Sub ListBox1_Click(sender As Object, e As EventArgs) Handles
ListBox1.Click
    Label13.Text = ListBox1.SelectedItem.ToString()
End SubEnd Class
```

When the above code is executed and run using Start button available at the Microsoft Visual Studio tool bar, it will show the following window –



Fill the list and check workings of other buttons –



جامعة بغداد/ كلية التربية للعلوم الصرفة ابن الهيثم/قسم علوم الحاسبات
المرحلة الثالثة صباحي / مسائي (نظري)
السنة ٢٠٢١-٢٠٢٢
استاذة المادة: شهلاء طالب

Visual Programming

Lecture 7 – MsgBox

MsgBox () Function

The objective of MsgBox is to produce a pop-up message box and prompt the user to click on a command button before he /she can continue. This format is as follows:

yourMsg=MsgBox (Prompt, Style Value, Title)

The first argument, Prompt, will display the message in the message box. The Style Value will determine what type of command buttons appear on the message box, please refer to Table 1 for types of command button displayed. The Title argument will display the title of the message board.

Style Value	Named Constant	Buttons Displayed
0	vbOkOnly	Ok button
1	vbOkCancel	Ok and Cancel buttons
2	vbAbortRetryIgnore	Abort, Retry and Ignore buttons.
3	vbYesNoCancel	Yes, No and Cancel buttons
4	vbYesNo	Yes and No buttons
5	vbRetryCancel	Retry and Cancel buttons

Table 1: Style Values

yourMsg is a variable that holds values that are returned by the MsgBox () function. The values are determined by the type of buttons being clicked by the users. It has to be declared as Integer data type in the procedure or in the general declaration section. Table 2 shows the values, the corresponding named constant and buttons.

Value	Named Constant	Button Clicked
1	vbOk	Ok button
2	vbCancel	Cancel button
3	vbAbort	Abort button
4	vbRetry	Retry button
5	vbIgnore	Ignore button
6	vbYes	Yes button
7	vbNo	No button

Table 2 : Return Values and Command Buttons

To make the message box look more sophisticated, you can add an icon besides the message. There are four types of icons available in VB as shown in Table 3.





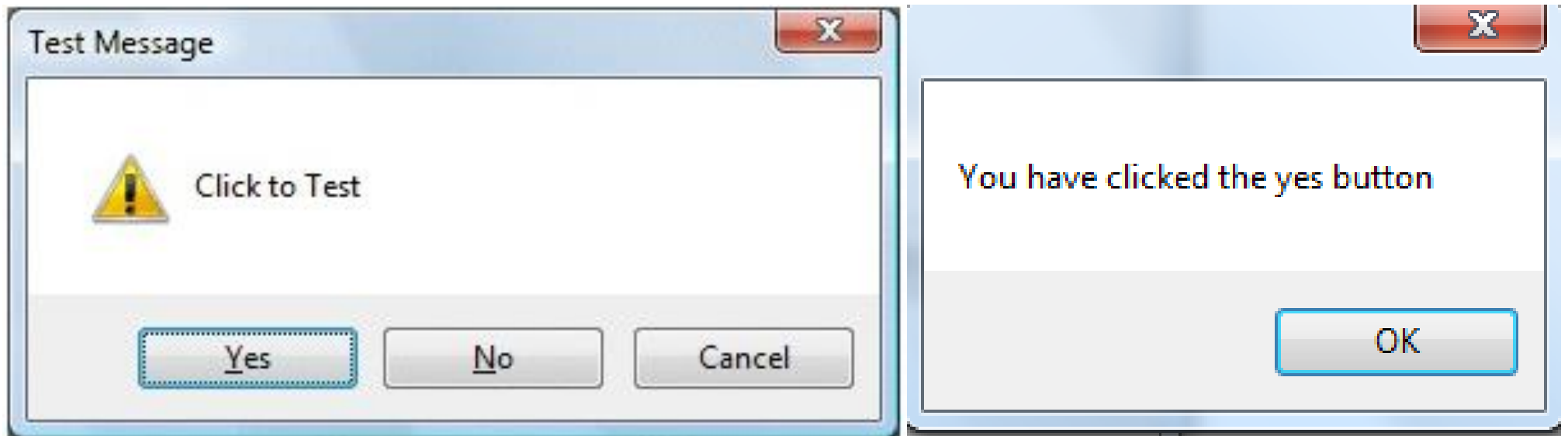
Value	Named Constant	Icon
16	vbCritical	
3	vbQuestion	
48	vbExclamation	
64	vbInformation	

Table 3: Types of Icons

```
Private Sub Button6_Click(sender As Object, e As EventArgs) Handles Button6.Click
    Dim testMsg As Integer
    testMsg = MsgBox("Click to Test", vbYesNoCancel + vbExclamation, "Test Message")
    If testMsg = 6 Then
        MessageBox.Show("You have clicked the yes button")
    ElseIf testMsg = 7 Then
        MessageBox.Show("You have clicked the NO button")
    Else
        MessageBox.Show("You have clicked the Cancel button")
    End If
End Sub
```



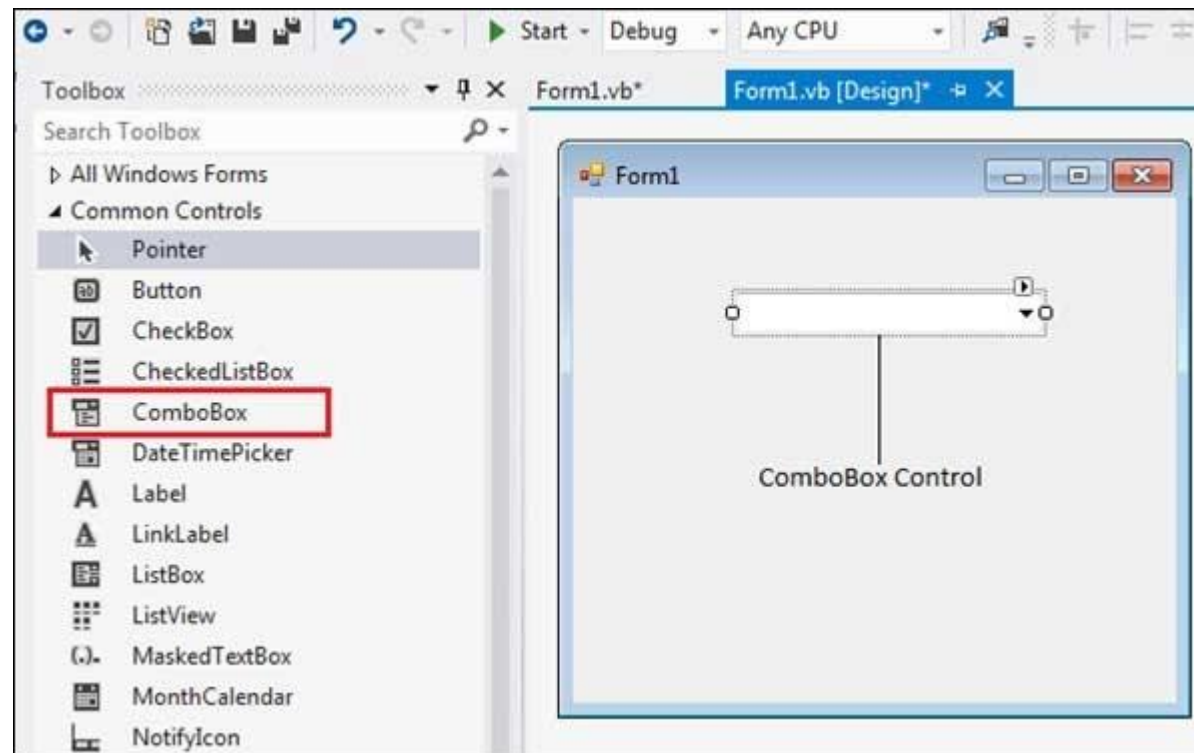
جامعة بغداد/ كلية التربية للعلوم الصرفة ابن الهيثم/قسم علوم الحاسبات
المرحلة الثالثة صباحي / مسائي (نظري)
السنة ٢٠٢١-٢٠٢٢
استاذة المادة: شهلاء طالب

Visual Programming

Lecture 8 –ComboBox Control & String Manipulation Functions

ComboBox Control

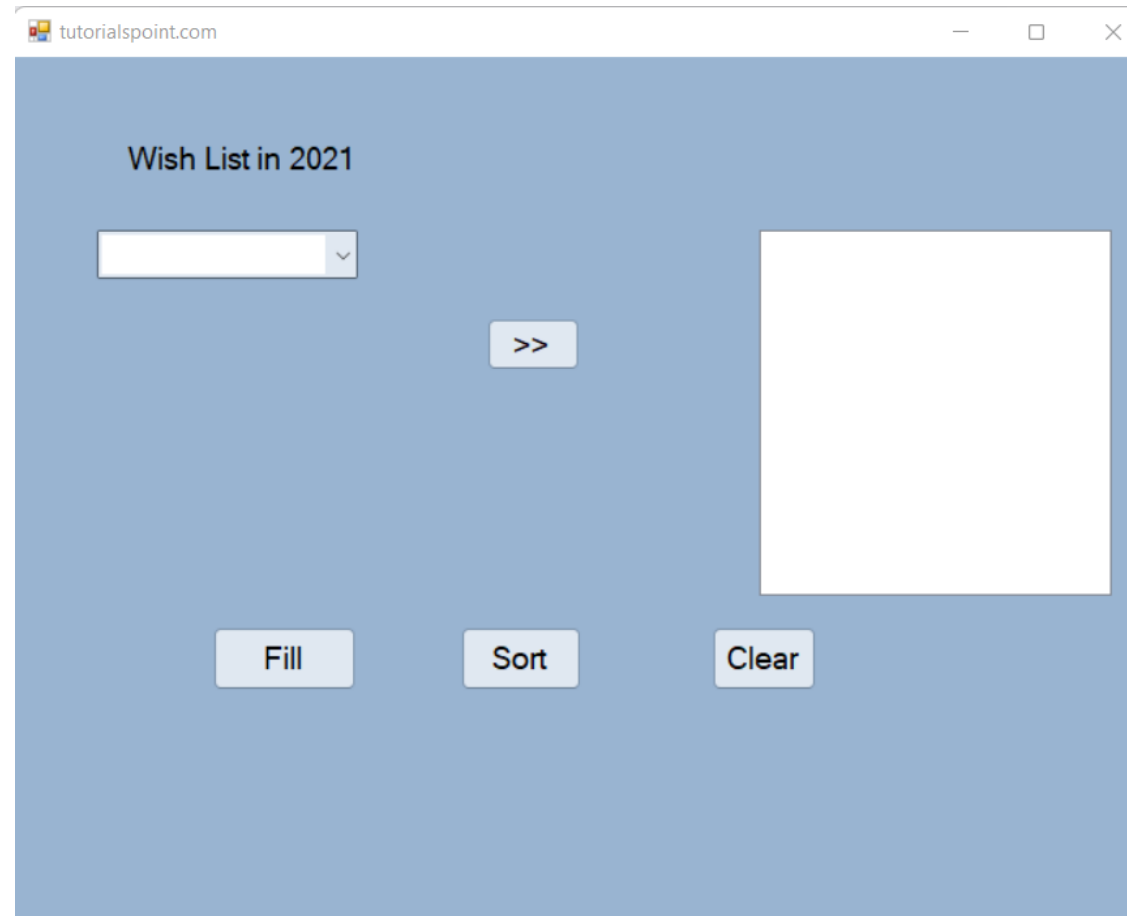
The ComboBox control is used to display a drop-down list of various items. It is a combination of a text box in which the user enters an item and a drop-down list from which the user selects an item.



In this example, let us fill a combo box with various items, get the selected items in the combo box and show them in a list box and sort the items.

Drag and drop a combo box to store the items, a list box to display the selected items, four button controls to add to the list box with selected items, to fill the combo box, to sort the items and to clear the combo box list, respectively.


Add a label control that would display the selected item.



```
Private Sub Button2_Click(sender As Object, e As EventArgs) Handles  
Button2.Click
```

```
    ComboBox1.Items.Clear()  
    ComboBox1.Items.Add("Safety")  
    ComboBox1.Items.Add("Security")  
    ComboBox1.Items.Add("Governance")  
    ComboBox1.Items.Add("Good Music")  
    ComboBox1.Items.Add("Good Movies")  
    ComboBox1.Items.Add("Good Books")  
    ComboBox1.Items.Add("Education")  
    ComboBox1.Text = "Select from..."  
    ComboBox1.Items.Add("Roads")  
    ComboBox1.Items.Add("Health")  
    ComboBox1.Items.Add("Food for all")  
    ComboBox1.Items.Add("Shelter for all")  
    ComboBox1.Items.Add("Industrialisation")  
    ComboBox1.Items.Add("Peace")  
    ComboBox1.Items.Add("Liberty")  
    ComboBox1.Items.Add("Freedom of Speech")
```

```
End Sub
```



Add items to
ComboBox

```
Private Sub Button3_Click(sender As Object, e As EventArgs) Handles  
Button3.Click
```

```
    ComboBox1.Sorted = True
```

```
End Sub
```

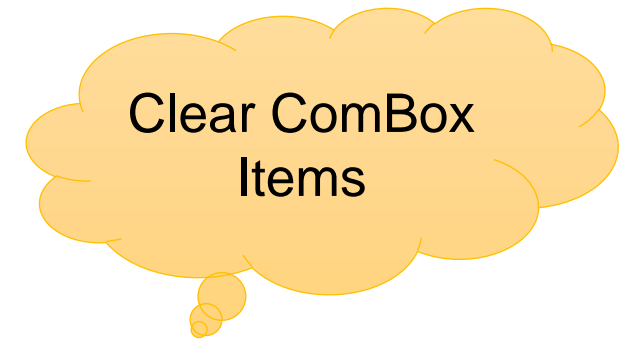


```
Private Sub Button4_Click(sender As Object, e As EventArgs)  
Handles Button4.Click
```

```
    ComboBox1.Items.Clear()
```

```
    ComboBox1.Text = ""
```

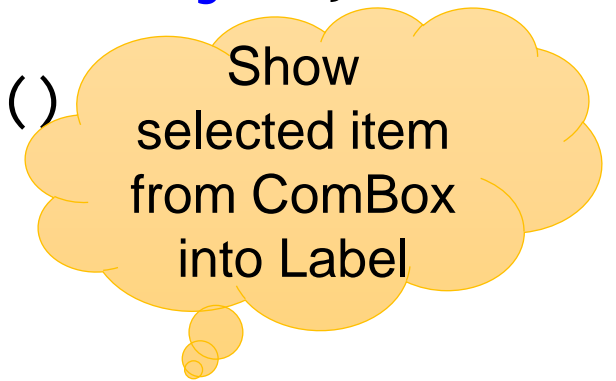
```
End Sub
```



```
Private Sub ComboBox1_SelectedIndexChanged(sender As Object, e As  
EventArgs) Handles ComboBox1.SelectedIndexChanged
```

```
    Label1.Text = ComboBox1.SelectedItem.ToString()
```

```
End Sub
```



```
Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
    ' Set the caption bar text of the form.
    Me.Text = "tutorialspoint.com"
```

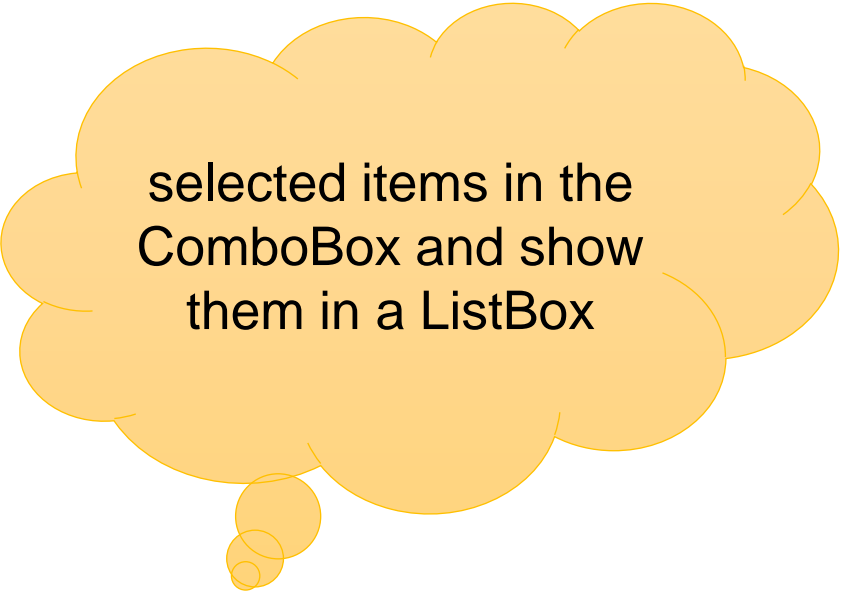
```
End Sub
```

```
Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
    If ComboBox1.SelectedIndex > -1 Then
        Dim sindex As Integer
        sindex = ComboBox1.SelectedIndex
        Dim sitem As Object
        sitem = ComboBox1.SelectedItem
        ListBox1.Items.Add(sitem & " " & sindex)
    Else
        MsgBox("you did not select")
    End If
```

```
End Sub
```

A yellow thought bubble with a black outline, containing the text "Change Form Text".

Change
Form Text

A large yellow thought bubble with a black outline, containing the text "selected items in the ComboBox and show them in a ListBox".

selected items in the
ComboBox and show
them in a ListBox

String Manipulation Functions

The Len Function	The Len function returns an integer value which is the length of a phrase or a sentence, including the empty spaces.	Len ("Phrase")
The Right Function	The Right function extracts the right portion of a phrase.	Right ("Phrase", n)
The Left Function	The Left function extract the left portion of a phrase.	Left("Phrase", n)
The Ltrim Function	The Ltrim function trims the empty spaces of the left portion of the phrase.	Ltrim("Phrase")
The Rtrim Function	The Rtrim function trims the empty spaces of the right portion of the phrase.	Rtrim("Phrase")
The Trim function	The Trim function trims the empty spaces on both side of the phrase.	Trim("Phrase")
The Mid Function	The Mid function extracts a substring from the original phrase or string.	Mid(phrase, position, n)
The InStr function	The InStr function looks for a phrase that is embedded within the original phrase and returns the starting position of the embedded phrase	Instr (n, original phrase, embedded phrase)

The Ucase and the Lcase functions	The Ucase function converts all the characters of a string to capital letters. On the other hand, the Lcase function converts all the characters of a string to small letters.	Ucase (phrase) Lcase (phrase)
The Str and Val functions	The Str is the function that converts a number to a string while the Val function converts a string to a number.	Str(number) Val(phrase)
The Chr and the Asc functions	The Chr function returns the string that corresponds to an ASCII code while the Asc function converts an ASCII character or symbol to the corresponding ASCII code.	Chr(charcode) Asc(Character)
The vbCrLf Named Constant	The vbCrLf named constant is a combination of two abbreviations Cr and Lf. Cr has numeric code Chr(13) which represents carriage return and Lf has numeric code Chr(10) which represents line feed. Carriage return means move the cursor to the left of the text field, and line feed means move down one row in the text field. By combining Cr and Lf, vbCrLf make it possible to display multiple lines in a text field such as in a message box	vbCrLf

Example

Form1



String Functions

Len

Trim

Str

Right

Mid

Val

Left

InStr

Chr

Rtrim

Ucase

AscW

Ltrim

LCase

vbCrLf

```
Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
    MsgBox(Len("computer "))
End Sub
```

```
Private Sub Button2_Click(sender As Object, e As EventArgs) Handles Button2.Click
    Dim x As String
    Dim z As String = "homework"
    x = Microsoft.VisualBasic.Right(z, 4)
    MsgBox(x)
End Sub
```

```
Private Sub Button3_Click(sender As Object, e As EventArgs) Handles Button3.Click
    Dim x As String
    Dim z As String = "computer"
    x = Microsoft.VisualBasic.Left(z, 2)
    MsgBox(x)
End Sub
```

```
Private Sub Button4_Click(sender As Object, e As EventArgs) Handles Button4.Click
    Dim x As String
    x = Mid("clever", 3)
    MsgBox(x)
End Sub
```

```
Private Sub Button5_Click(sender As Object, e As EventArgs) Handles Button5.Click
    Dim x As String = Trim("      I win      my      ")
    MsgBox(x)
End Sub
```

```
Private Sub Button6_Click(sender As Object, e As EventArgs) Handles Button6.Click
    Dim x As String = UCase("good")
    MsgBox(x)
End Sub
```

```
Private Sub Button7_Click(sender As Object, e As EventArgs) Handles Button7.Click
    Dim x As String = LCase("sMART")
    MsgBox(x)
End Sub
```

```
Private Sub Button8_Click(sender As Object, e As EventArgs) Handles Button8.Click
    Dim x As String = AscW("Agfhkjkg")
    MsgBox(x)
End Sub
```

```
Private Sub Button10_Click(sender As Object, e As EventArgs) Handles Button10.Click
    MsgBox(RTrim("Visual Basic "))
End Sub
```

```
Private Sub Button11_Click(sender As Object, e As EventArgs) Handles Button11.Click
    MsgBox(RTrim(" Visual Basic"))
End Sub
```

```
Private Sub Button12_Click(sender As Object, e As EventArgs) Handles Button12.Click
    MsgBox(InStr(1, "Visual Basic", " Basic"))
End Sub
```

```
Private Sub Button12_Click(sender As Object, e As EventArgs) Handles Button12.Click
    MsgBox(InStr(1, "Visual Basic", " Basic"))
End Sub
```

```
Private Sub Button13_Click(sender As Object, e As EventArgs) Handles Button13.Click
    MsgBox(Str("123"))
End Sub
```

```
Private Sub Button14_Click(sender As Object, e As EventArgs) Handles Button14.Click
    MsgBox(Val(123))
End Sub
```

```
Private Sub Button15_Click(sender As Object, e As EventArgs) Handles Button15.Click
    'Chr(65)=A, Chr(122)=z, Chr(37)=% , Asc("B")=66, Asc("&")=38
    MsgBox(Chr(65))
End Sub
```

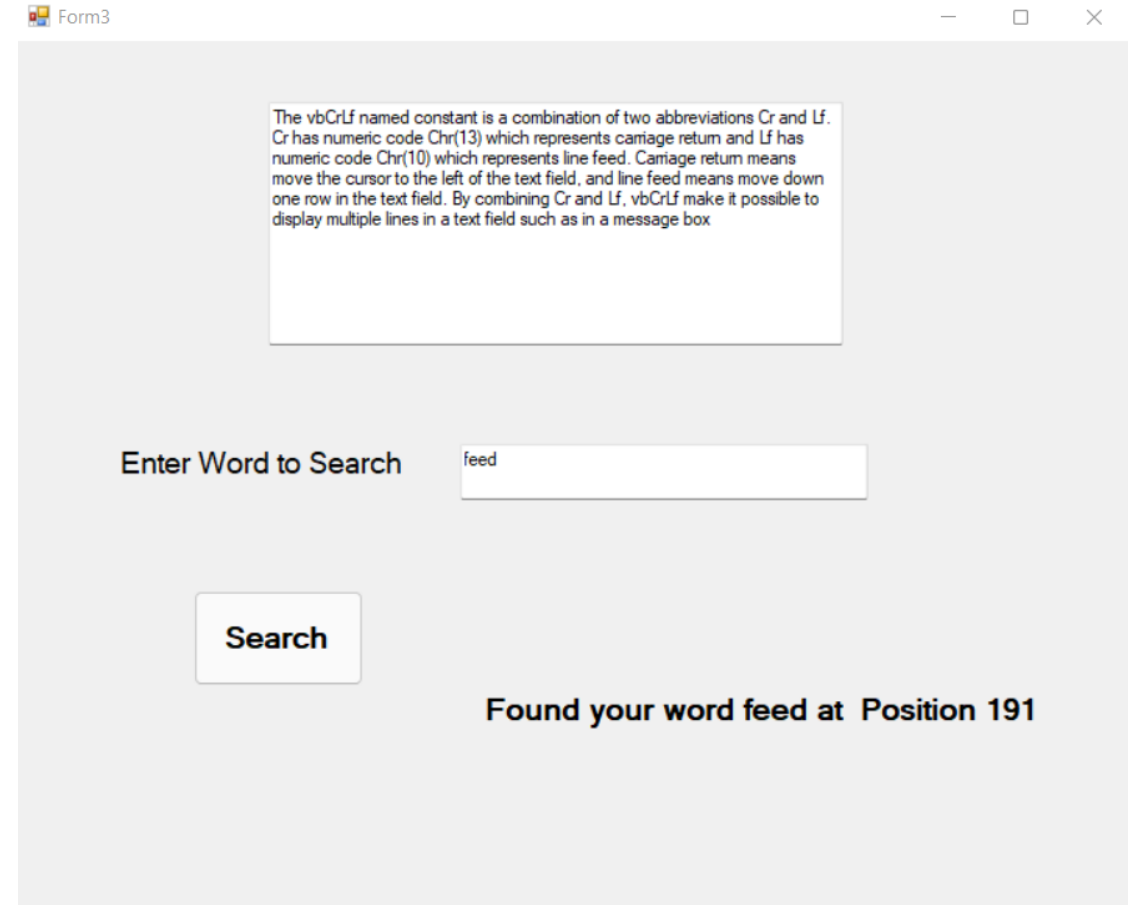
```
Private Sub Button17_Click(sender As Object, e As EventArgs) Handles Button17.Click
    MsgBox("Visual" & vbCrLf & "Basic")
End Sub
```

```
Class
```

Performing Word Search

We can make use of various string functions to perform word search from a textbox.

In the following example, we insert a textbox and set the multiline property to true. We also insert a textbox for the user to enter the word to search and a button to perform the search. Besides that, we also include a label control to display the result. In the code, we use the set Focus property to highlight the word found. In addition, we also use the SelectionStart to set the starting point of text selected.



```
Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
    Dim As Integer
    Dim m1, myWord As String
    m1 = TextBox1.Text
    myWord = TextBox3.Text
    n = InStr(1, m1, myWord)
    If n = 0 Then
        Label1.Text = "Your word not found, try again."
    Else
        Label1.Text = "Found your word " & myWord & " at " & " Position " & n
        TextBox1.Focus()
        TextBox1.SelectionStart = n - 1
        TextBox1.SelectionLength = Len(myWord)
    End If
End Sub
```

جامعة بغداد / كلية التربية للعلوم الصرفة ابن الهيثم / قسم علوم الحاسبات
المرحلة الثالثة صباحي / مسائي (نظري)
السنة ٢٠٢١-٢٠٢٢
استاذة المادة: شهلاء طالب

Visual Programming

Lecture 9 – Math Functions

Math Functions

The Value

The Result

Abs

Rnd

Exp

Round

Fix

Sqrt

Int

Max

Log

Min

The Abs function

The Abs function returns the absolute value of a given number.

```
Dim x As Integer =  
Math.Abs(Val(TextBox1.Text))  
x = Math.Abs(3)  
TextBox2.Text = x
```

The Exp function

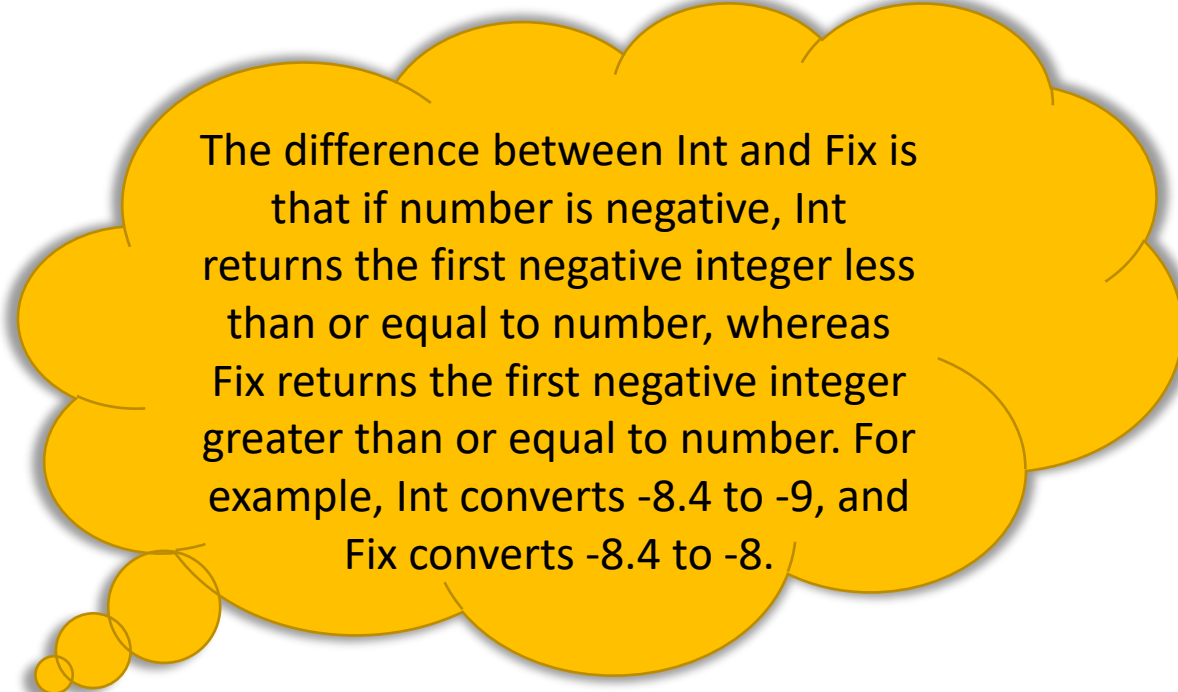
The Exp of a number x is the exponential value of x , i.e. e^x . For example, $\text{Exp}(1)=e=2.71828182$

```
Dim num1, num2 As Single  
num1 = Val(TextBox1.Text)  
num2 = Math.Exp(num1)  
TextBox2.Text = num2
```

The Fix Function

The Fix function truncates the decimal part of a positive number and returns the largest integer smaller than the number. However, when the number is negative, it will return smallest integer larger than the number. For example, $\text{Fix}(9.2)=9$ but $\text{Fix}(-9.4)=-9$

```
Dim num1, num2 As Single
num1 = Val(TextBox1.Text)
num2 = Fix(num1)
Textbox2.Text = num2
```



The difference between Int and Fix is that if number is negative, Int returns the first negative integer less than or equal to number, whereas Fix returns the first negative integer greater than or equal to number. For example, Int converts -8.4 to -9, and Fix converts -8.4 to -8.

The Int Function

The Int is a function that converts a number into an integer by truncating its decimal part and the resulting integer is the largest integer that is smaller than the number. For example

$\text{Int}(2.4)=2$, $\text{Int}(6.9)=6$, $\text{Int}(-5.7)=-6$,
 $\text{Int}(-99.8)=-100$

```
TextBox2.Text =  
Int(Val(TextBox1.Text))
```

The Log Function

The Log function is the function that returns the natural logarithm of a number. For example,
 $\text{Log}(10)=2.302585$

```
Dim num1, num2 As Single  
num1 = Val(TextBox1.Text)  
num2 = Math.Log(num1)  
TextBox2.Text = num2
```

The Rnd() Function

Rnd is a very useful function in Visual Basic 2012 . We use the Rnd function to write code that involves chance and probability. The Rnd function returns a random value between 0 and 1. Random numbers in their original form are not very useful in programming until we convert them to integers. For example, if we need to obtain a random output of 6 integers ranging from 1 to 6, which makes the program behave like a virtual dice, we need to convert the random numbers to integers using the formula $\text{Int}(\text{Rnd} * 6) + 1$.

```
Dim num As Double
```

```
num = Int(Rnd() * 6) + 1
```

```
Textbox2.Text = num
```

In this example, $\text{Int}(\text{Rnd} * 6)$ will generate a random integer between 0 and 5 because the function Int truncates the decimal part of the random number and returns an integer. After adding 1, you will get a random number between 1 and 6 every time you click the button. For example, let say the random number generated is 0.98, after multiplying it by 6, it becomes 5.88, and using the integer function $\text{Int}(5.88)$ will convert the number to 5; and after adding 1 you will get 6.

The Round Function

The Round function is the function that rounds up a number to a certain number of decimal places. The syntax is

Round (n, m)

which means to round a number n to m decimal places. For example, Math.Round (7.2567, 2) =7.26

```
Private Sub Button4_Click(sender As Object, e As EventArgs)
Handles Button4.Click
    Dim num1, num2 As Single
    num1 = TextBox1.Text
    num2 = Math.Round(num1, 2)
    Label1.Text = num2
End Sub
```

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

```
    Dim x As Integer = Math.Sqrt(25)
```

```
    MsgBox(x)
```

```
End Sub
```

```
Private Sub Button2_Click(sender As Object, e As EventArgs)  
Handles Button2.Click
```

```
    Dim x As Integer = Math.Max(10, 5)
```

```
    MsgBox(x)
```

```
End Sub
```

```
Private Sub Button3_Click(sender As Object, e As EventArgs)  
Handles Button3.Click
```

```
    Dim x As Integer = Math.Min(2, 5)
```

```
    MsgBox(x)
```

```
End Sub
```

جامعة بغداد / كلية التربية للعلوم الصرفة ابن الهيثم / قسم علوم الحاسبات
المرحلة الثالثة صباحي / مسائي (نظري)
السنة ٢٠٢٢-٢٠٢١
اساتذة المادة: شهلاء طالب

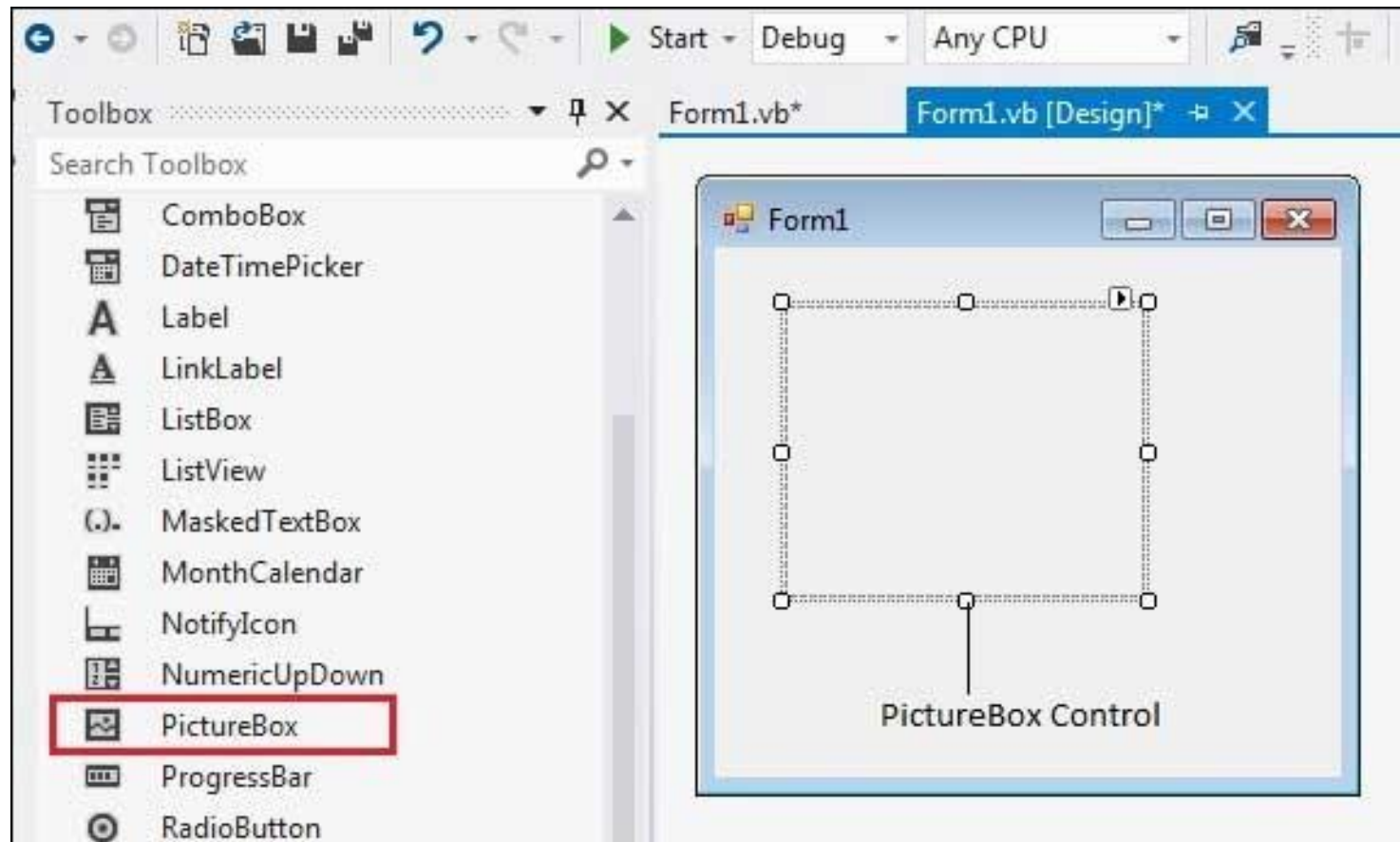
Visual Programming

Lecture 10 – PictureBox Control & Windows Media Player Control

PictureBox Control

The PictureBox control is used for displaying images on the form. The Image property of the control allows you to set an image both at design time or at run time.

Let's create a picture box by dragging a PictureBox control from the Toolbox and dropping it on the form.



PictureBox Control

You can set the Image property to the Image you want to display, either at design time or at run time. You can programmatically change the image displayed in a picture box, which is particularly useful when you use a single form to display different pieces of information.

```
PictureBox1.Image = Image.FromFile("C:\testImage.jpg")
```

Properties of the PictureBox Control

The `SizeMode` property, which is set to values in the `PictureBoxSizeMode` enumeration, controls the clipping and positioning of the image in the display area.

```
pictureBox1.SizeMode = PictureBoxSizeMode.StretchImage
```

here are five different `PictureBoxSizeMode` is available to `PictureBox` control.

`AutoSize` - Sizes the picture box to the image.

`CenterImage` - Centers the image in the picture box.

`Normal` - Places the upper-left corner of the image at upper left in the picture box.

`StretchImage` - Allows you to stretch the image in code.

You can change the size of the display area at run time with the `ClientSize` property.

You can change the size of the display area at run time with the `ClientSize` property.

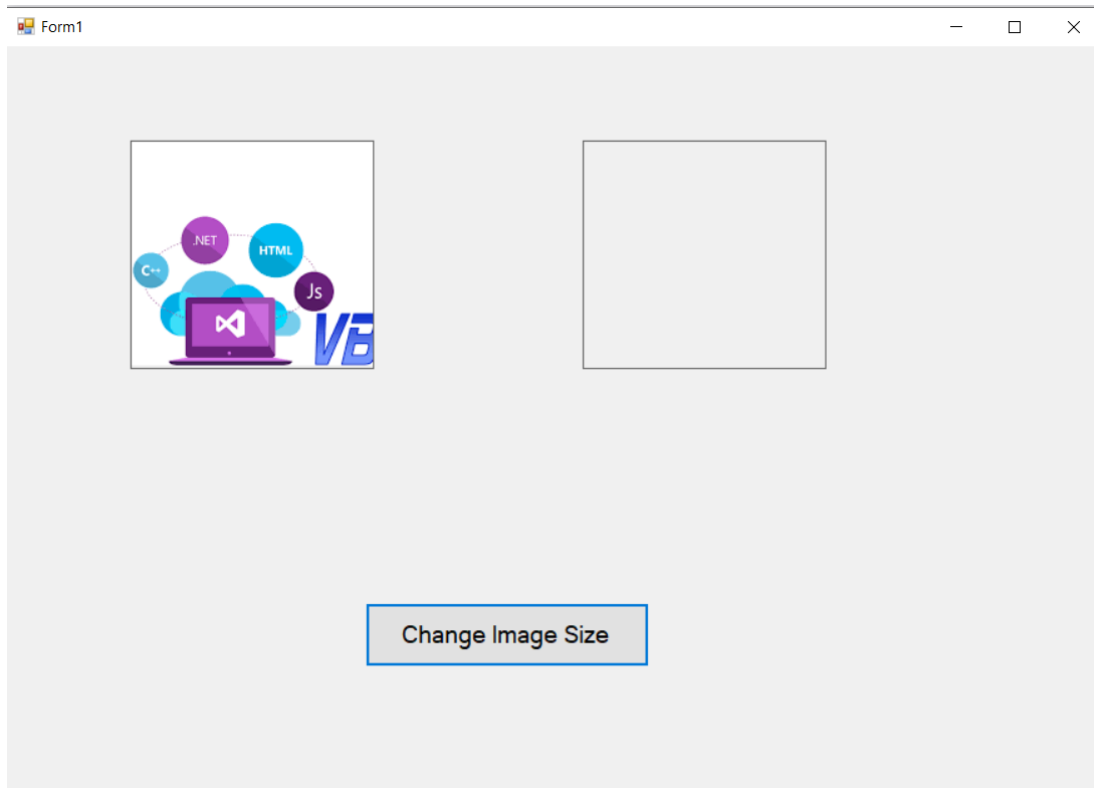
```
pictureBox1.ClientSize = New Size(xSize, ySize)
```

Example

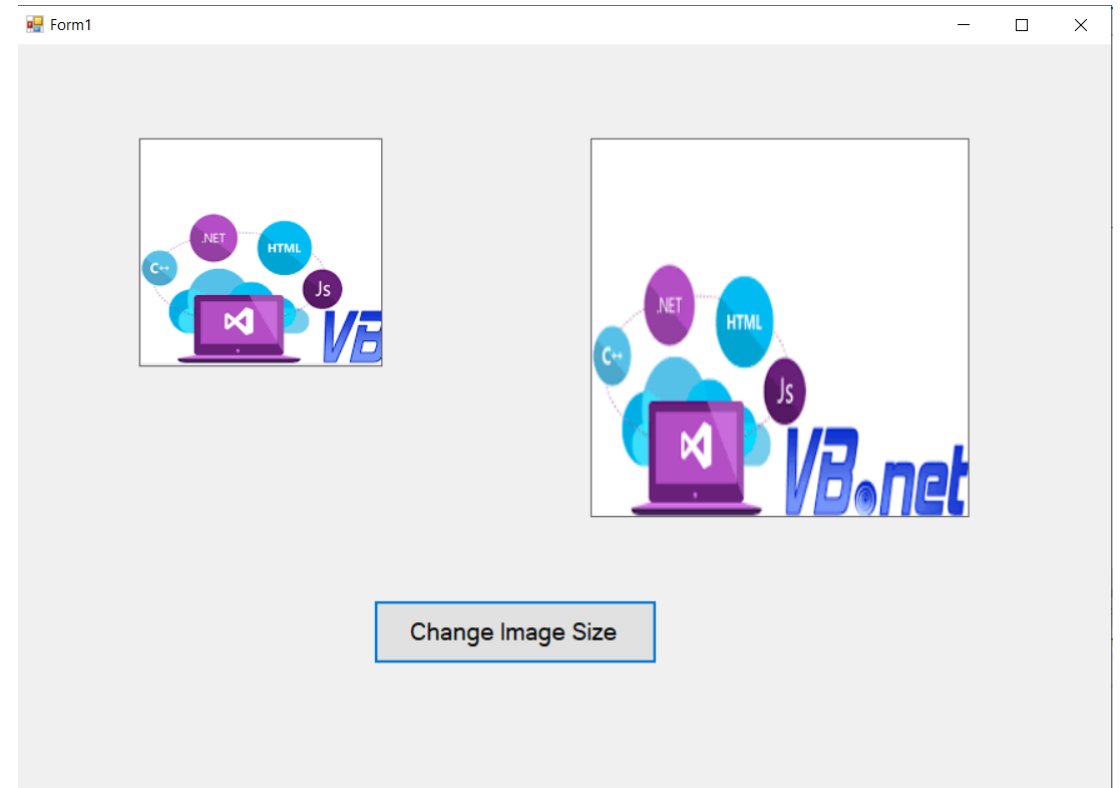
In this example, let us put two PictureBox and a button control on the form. We set the image property of the picture box to 11.png, as we used before. The Click event of the button named Button1 is to show the change of properties in PictureBox2.

```
PictureBox2.Image = Image.FromFile("C:\Users\user\Desktop\2020-2021\vb\11.png")
PictureBox2.ClientSize = New Size(300, 300)
PictureBox2.SizeMode = PictureBoxSizeMode.StretchImage
```

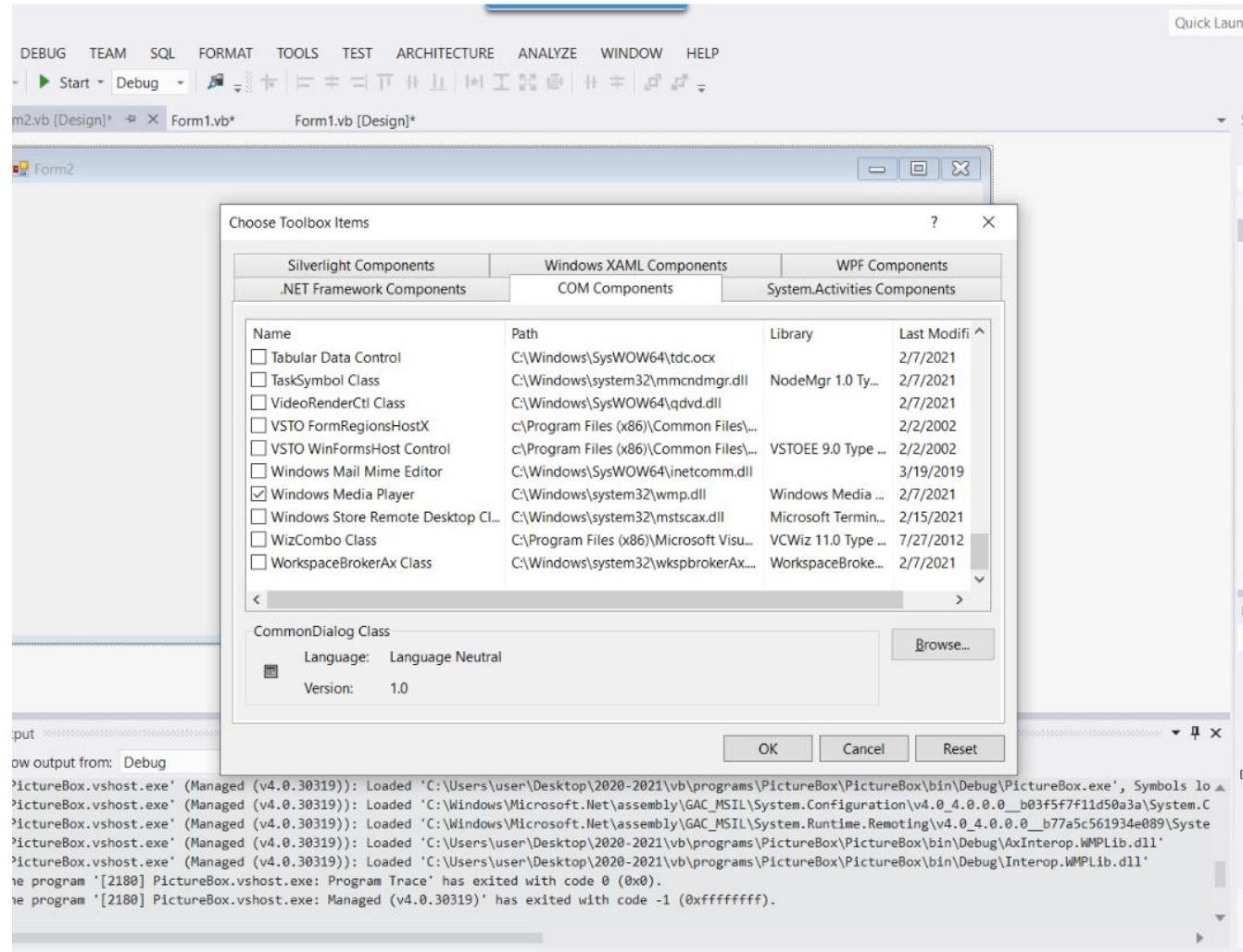
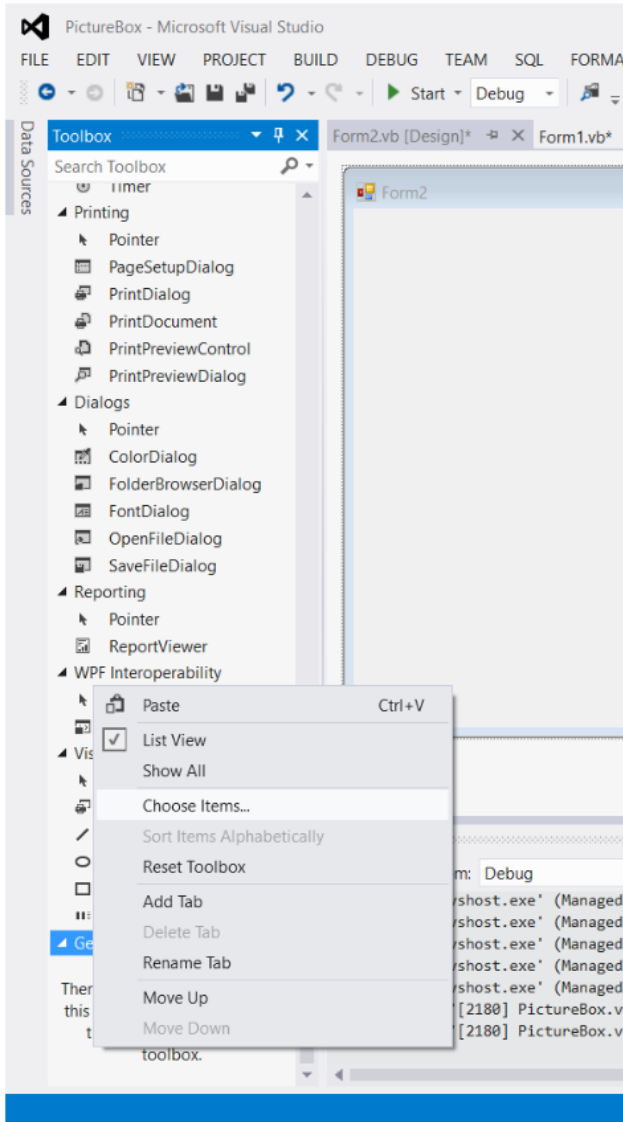
Before pressing Button1



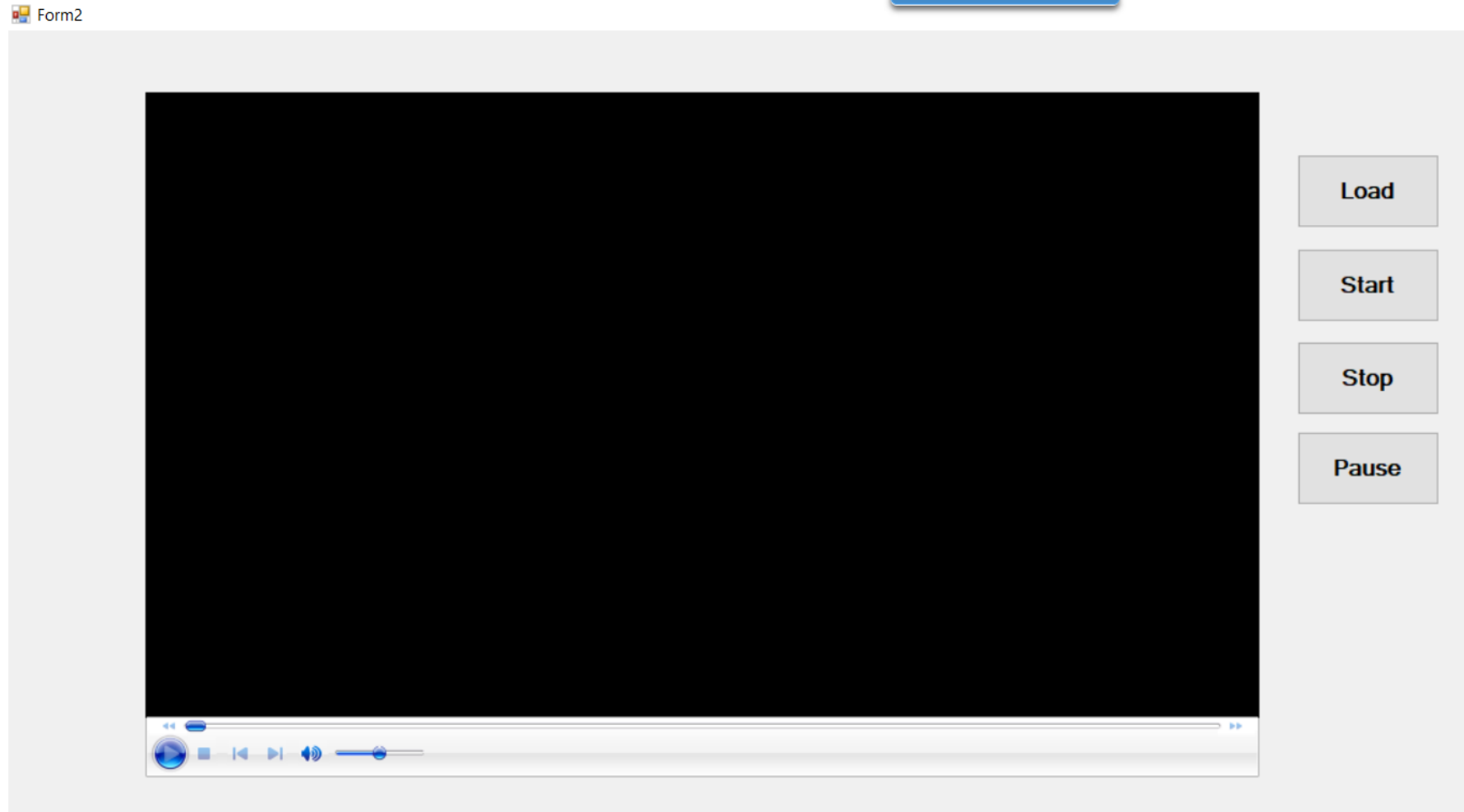
After pressing Button1



Windows Media Player Control



Example



The Code

```
Public Class Form2
```

```
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click  
        AxWindowsMediaPlayer1.URL = "C:\Users\user\Desktop\TOM&JERRY.mp4"
```

```
    End Sub
```

```
    Private Sub Button2_Click(sender As Object, e As EventArgs) Handles Button2.Click  
        AxWindowsMediaPlayer1.Ctlcontrols.play()
```

```
    End Sub
```

```
    Private Sub Button3_Click(sender As Object, e As EventArgs) Handles Button3.Click  
        AxWindowsMediaPlayer1.Ctlcontrols.stop()
```

```
    End Sub
```


```
    Private Sub Button4_Click(sender As Object, e As EventArgs) Handles Button4.Click  
        AxWindowsMediaPlayer1.Ctlcontrols.pause()
```

```
    End Sub
```

```
End Class
```

The Implementation

Form2



Playing 'TOM&JERRY': 1792 K bits/second

Pause

Load

Start

Stop

Pause

جامعة بغداد / كلية التربية للعلوم الصرفة ابن الهيثم / قسم علوم الحاسبات
المرحلة الثالثة صباحي / مسائي (نظري)
السنة ٢٠٢١-٢٠٢٢
اساتذة المادة: شهلاء طالب

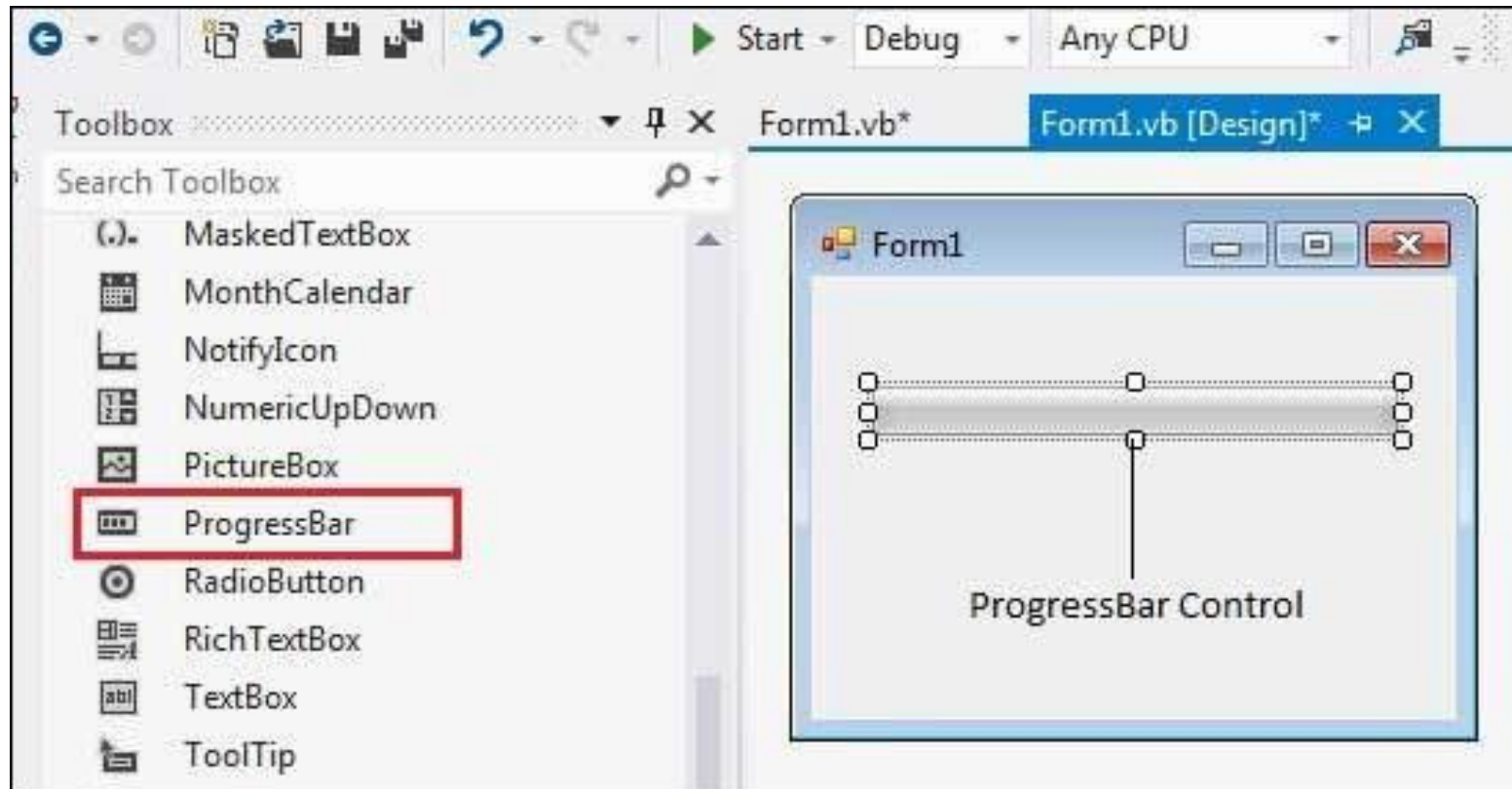
Visual Programming

Lecture 11 – ProgressBar, ScrollBars, DateTimePicker and MenuStrip Controls

ProgressBar Control

It is used to provide visual feedback to your users about the status of some task. It shows a bar that fills in from left to right as the operation progresses.

The ProgressBar control is used by the user to acknowledge the progress status of some defined tasks, such as downloading a large file from the web, copying files, installing software, calculating complex results, and more.



ProgressBar Properties

The Maximum and Minimum properties define the range of values to represent the progress of a task.

Minimum : Sets the lower value for the range of valid values for progress.

Maximum : Sets the upper value for the range of valid values for progress.

Value : This property obtains or sets the current level of progress.

By default, Minimum and Maximum are set to 0 and 100. As the task proceeds, the ProgressBar fills in from the left to the right.

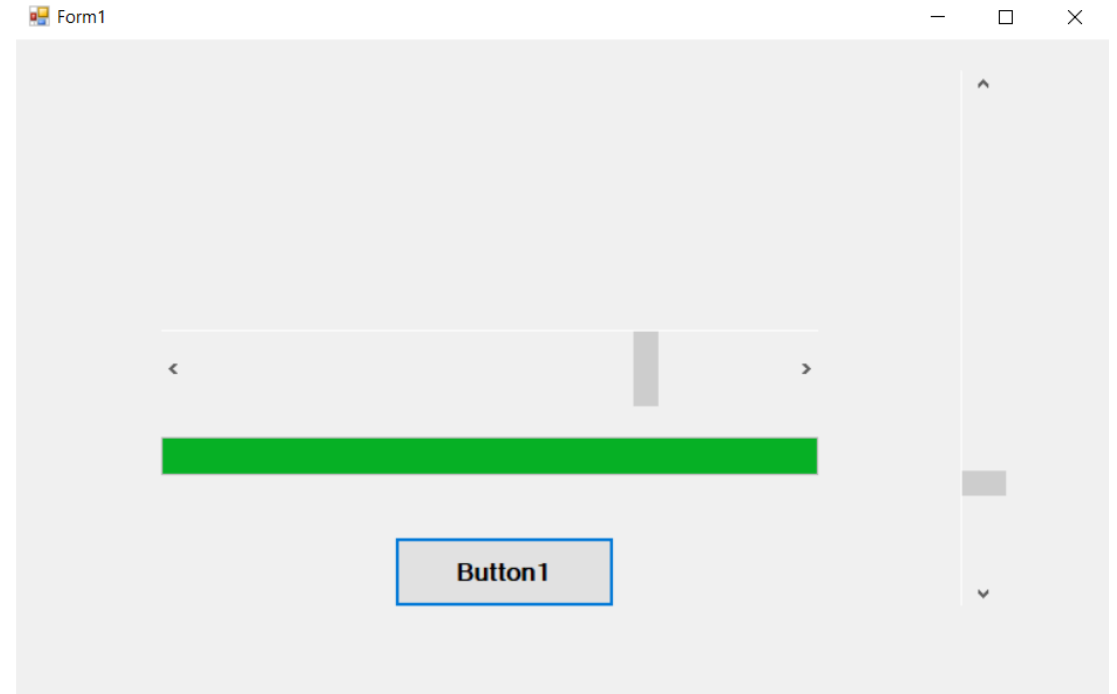
ScrollBars Control

A ScrollBar control is used to create and display vertical and horizontal scroll bars on the Windows form. It is used when we have large information in a form, and we are unable to see all the data. Therefore, we used VB.NET ScrollBar control. Generally, ScrollBar is of two types: HScrollBar for displaying scroll bars and VScrollBar for displaying Vertical Scroll bars.

Example

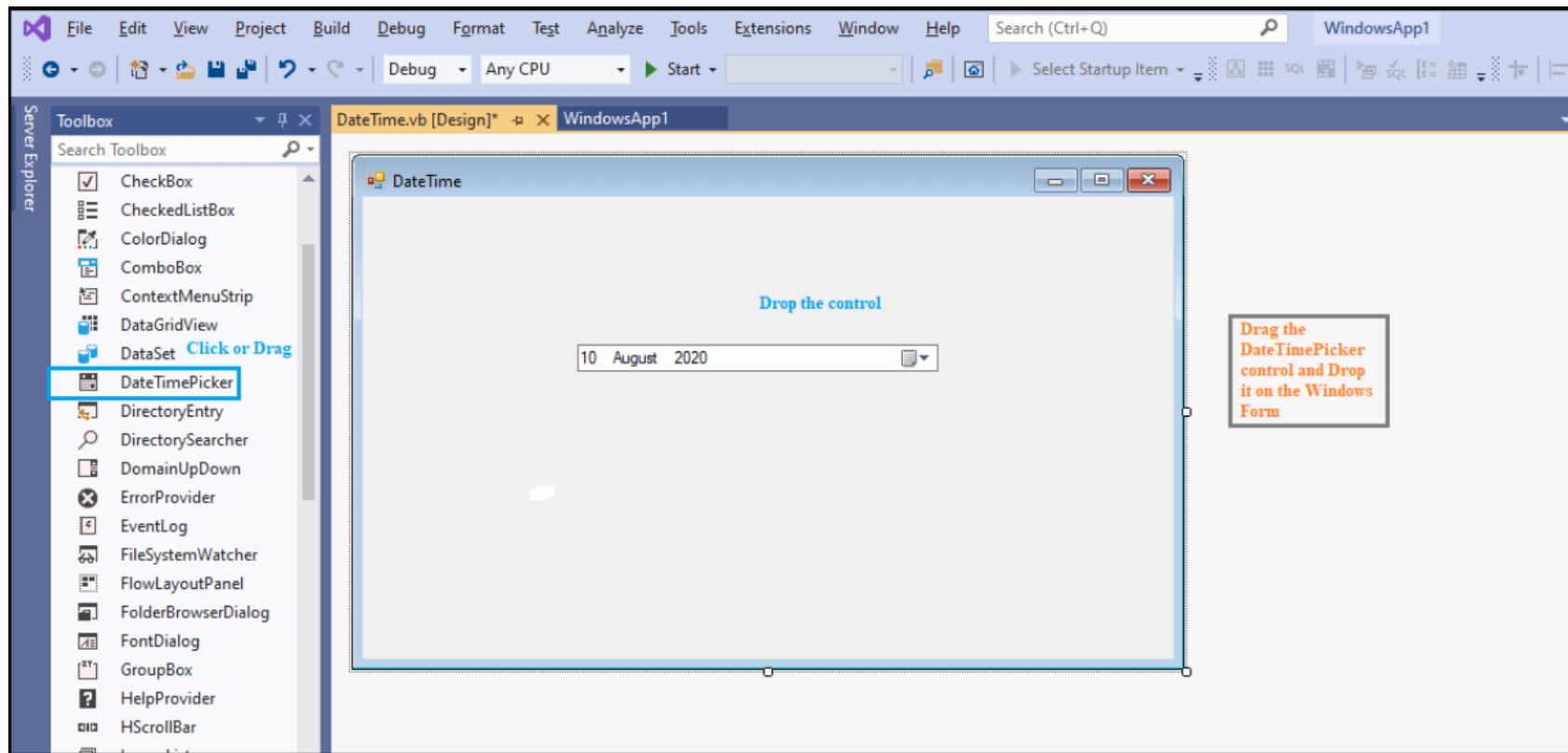
```
Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
    ProgressBar1.Visible = True
    Dim i As Integer
    ProgressBar1.Minimum = 0
    ProgressBar1.Maximum = 300

    For i = 0 To 300 Step 100
        ProgressBar1.Value = i
        HScrollBar1.Value = i
        If i > ProgressBar1.Maximum Then
            i = ProgressBar1.Maximum
        End If
    Next
    MsgBox("Successfully Completed")
    VScrollBar1.Value = i
End Sub
```



DateTimePicker Control

The DateTimePicker control allows the user to select or display date and time values with a specified format in Windows Forms. Furthermore, we can determine the current date and time using the Value property of the DateTimePicker control. By default, the Value property returns the current date and time in the DateTimePicker.



DateTimePicker Control


- The DateTimePicker control prompts the user for a date or time using a graphical calendar with scroll arrows. The most important property of the DateTimePicker is the Value property, which holds the selected date and time.
- The Value property is set to the current date by default. You can use the Text property or the appropriate member of Value to get the date and time value.
- The control can display one of several styles, depending on its property values. The values can be displayed in four formats, which are set by the Format property: Long, Short, Time, or Custom.

Label1

Label3

Monday , April 19, 2021 

Label4

Monday , April 19, 2021 

Button1

Label2


```
Public Class Form3
```

```
    Private Sub Form3_Load(sender As Object, e As EventArgs) Handles MyBase.Load
```

```
        Button1.Text = "Calculate Days"
```

```
        Label1.Text = "Calculate the total days from your date of birth to the current date."
```

```
        Label2.Text = "Total Days"
```

```
        Label3.Text = "Select the DOB"
```

```
        Label4.Text = "Current Date"
```

```
        DateTimePicker1.Format = DateTimePickerFormat.Long
```

```
    End Sub
```

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

```
        Dim dp As Date = DateTimePicker1.Value
```

```
        Dim dp2 As Date = DateTimePicker2.Value
```

```
        Dim result As TimeSpan = dp.Subtract(dp2)
```

```
        Dim ds As Integer = result.TotalDays
```

```
        TextBox1.Text = ds
```

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

```
        MsgBox(" Days = " & ds)
```

```
    End Sub
```

```
End Class
```

Calculate the total days from your date of birth to the current date.

Select the DOB

Current Date

progressbar ×

Days = 10

OK

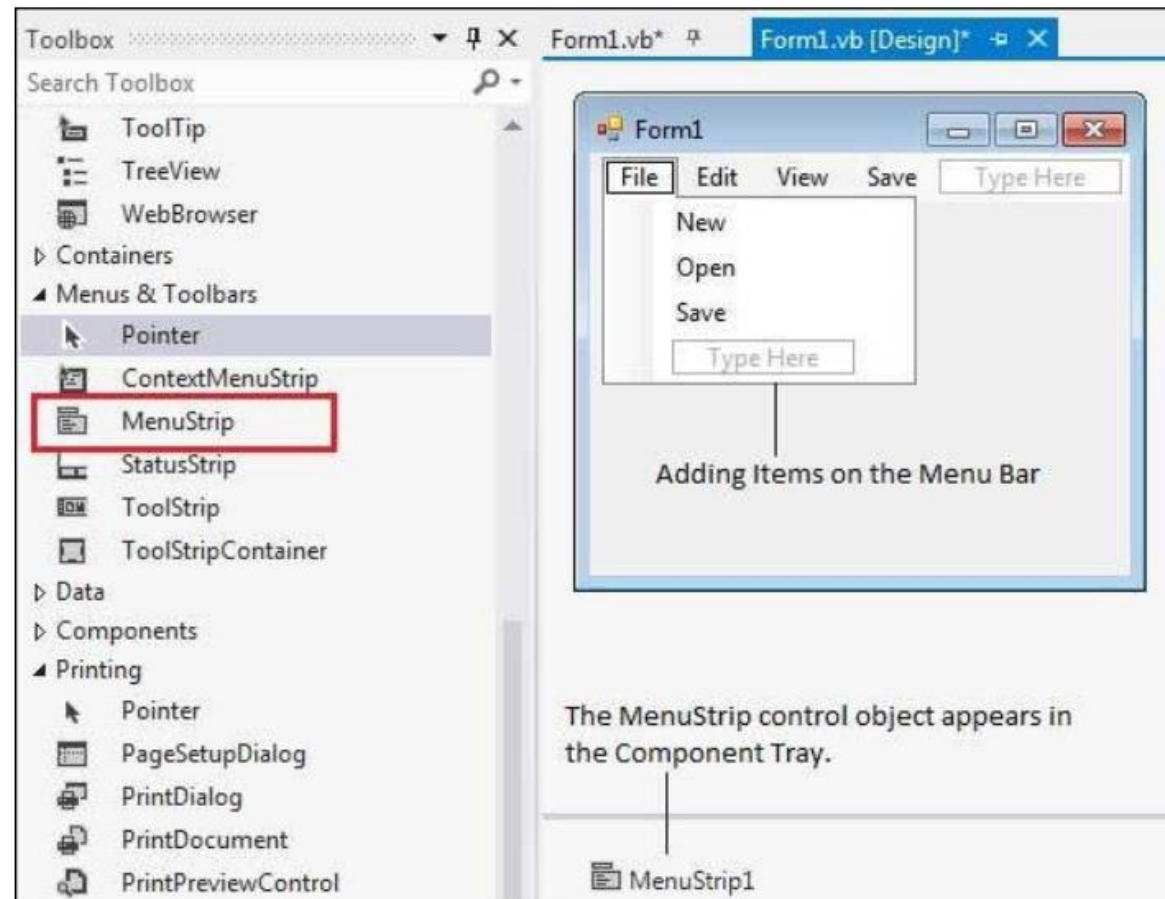
Calculate

Total Days

10

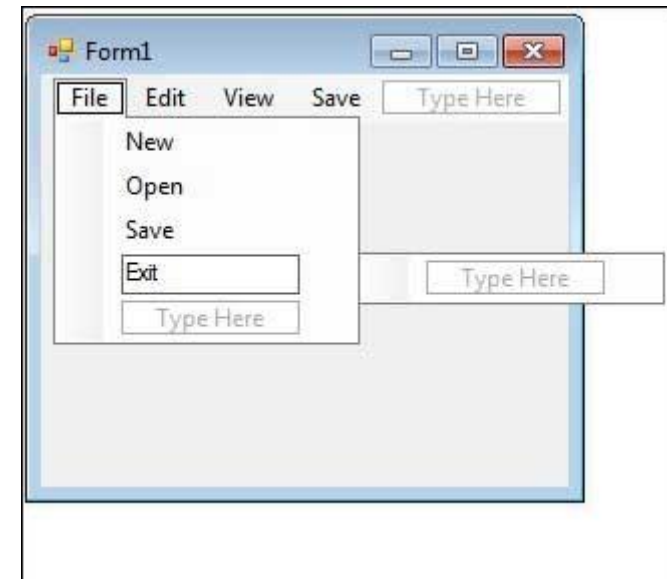
MenuStrip Control

The **MenuStrip** control represents the container for the menu structure. The MenuStrip control works as the top-level container for the menu structure. The ToolStripMenuItem class and the ToolStripDropDownMenu class provide the functionalities to create menu items, sub menus and drop-down menus.



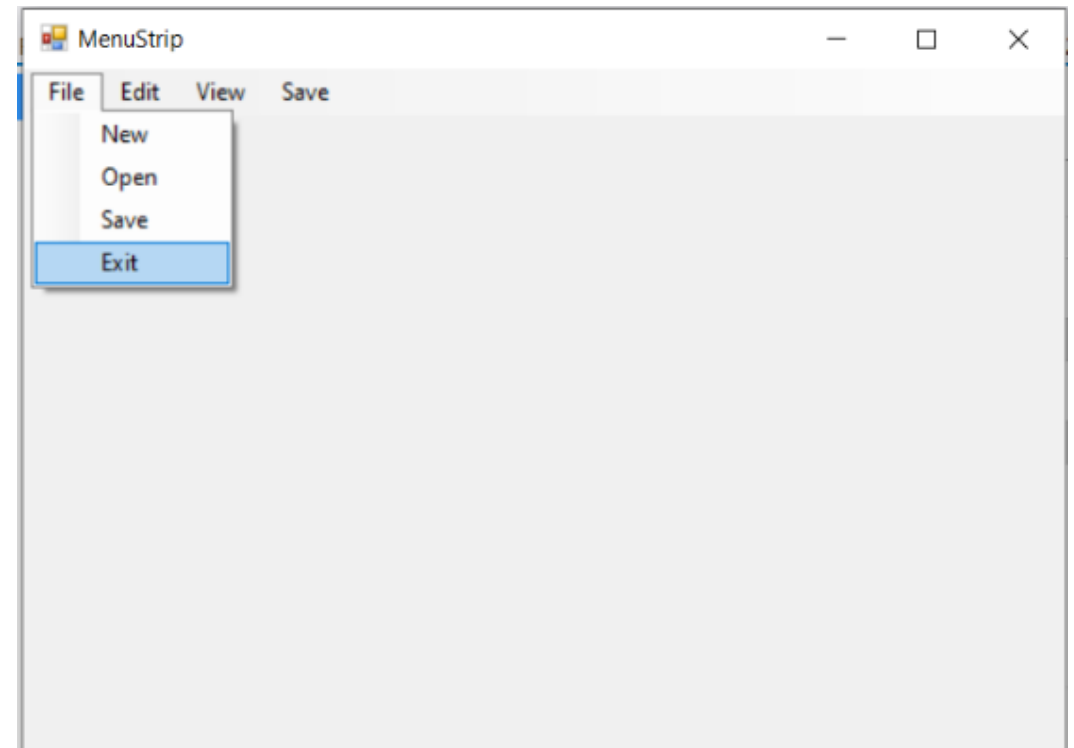
Example

- In this example, let us add menu and sub-menu items.
- Take the following steps –
- Drag and drop or double click on a MenuStrip control, to add it to the form.
- Click the Type Here text to open a text box and enter the names of the menu items or sub-menu items you want. When you add a sub-menu, another text box with 'Type Here' text opens below it.
- Complete the menu structure shown in the diagram above.
- Add a sub menu Exit under the File menu.



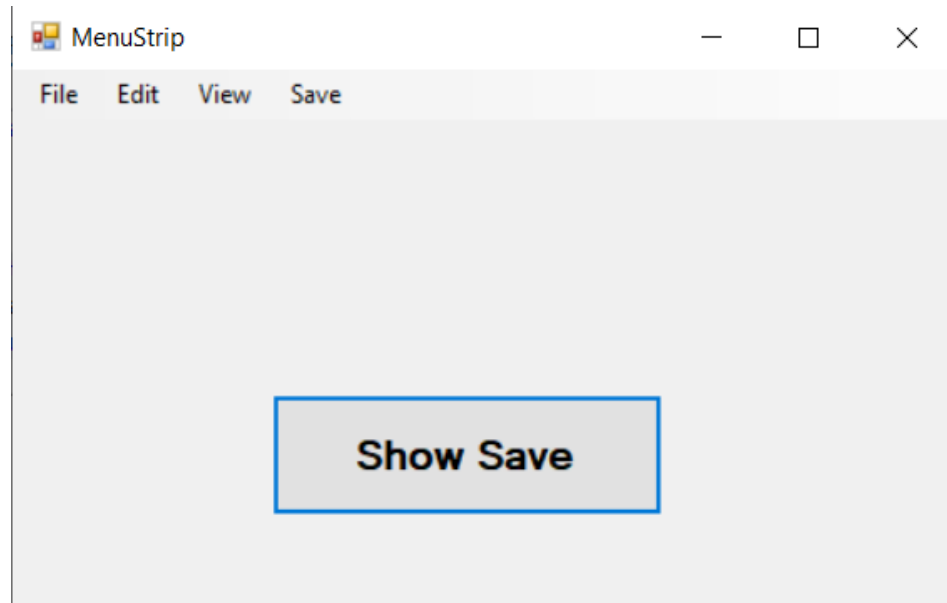
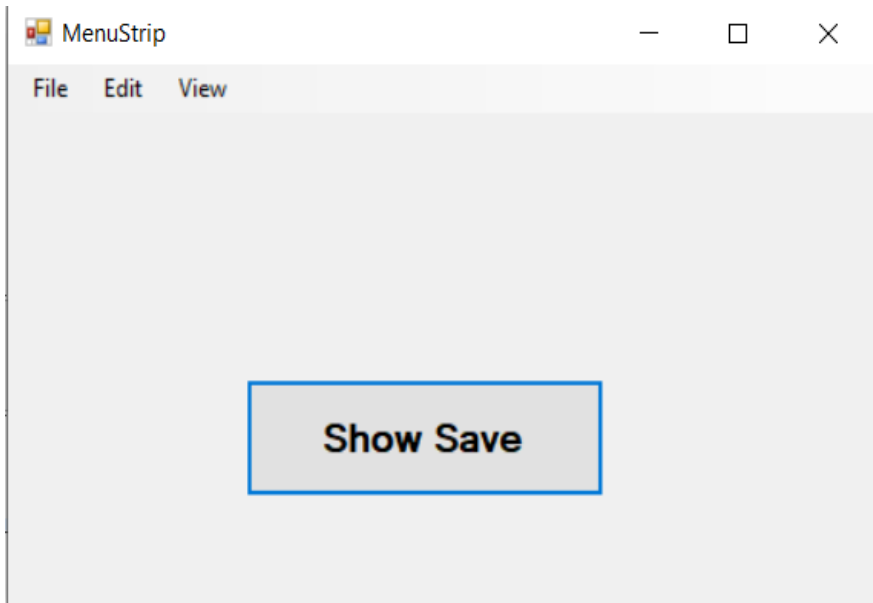
The Code

```
Private Sub ExitToolStripMenuItem_Click(sender As  
Object, e As EventArgs) Handles  
ExitToolStripMenuItem.Click  
    End  
End Sub
```

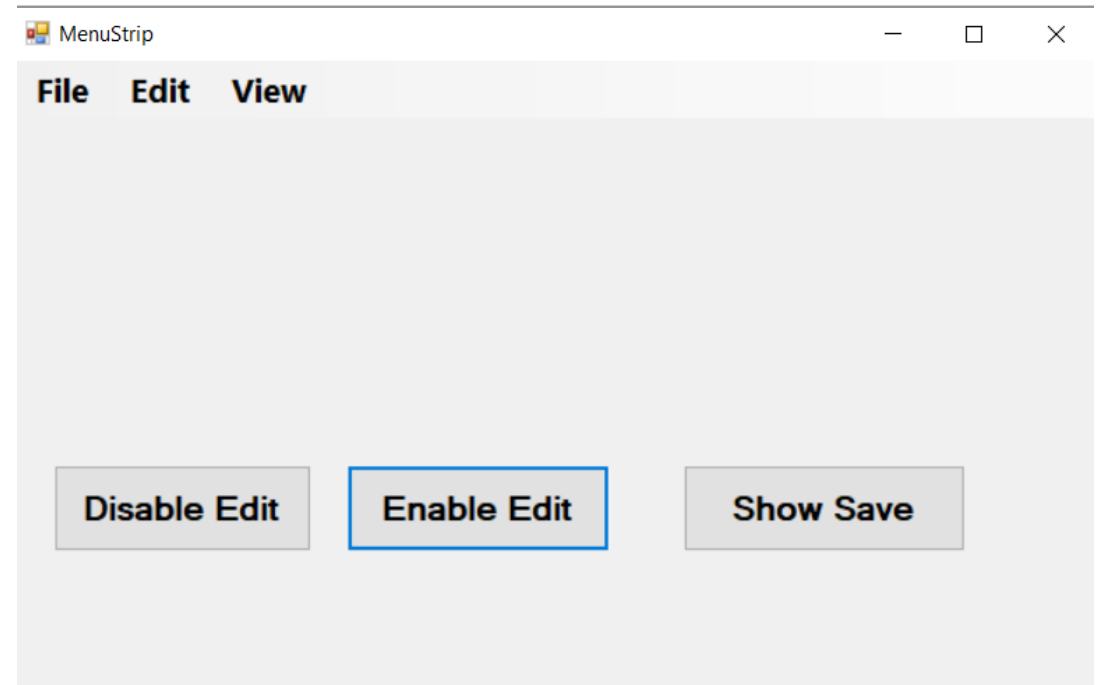
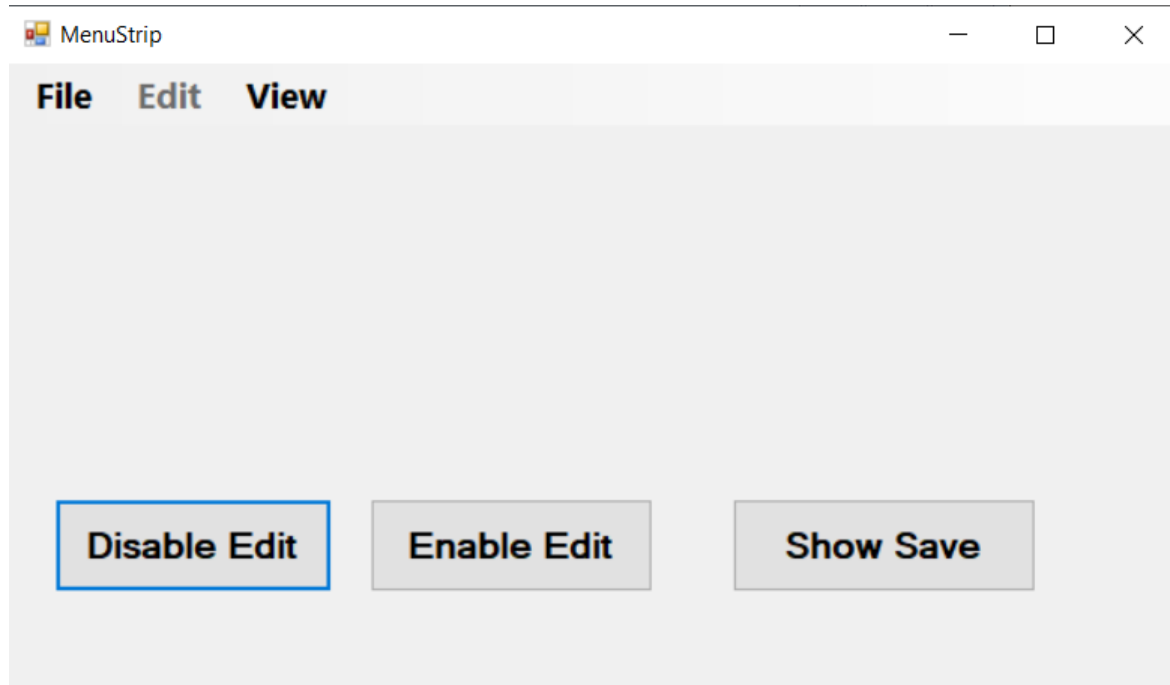


Hide and Show item in Menu Strip

```
Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
    SaveToolStripMenuItem.Visible = True
End Sub
```



Disable and Enable the Menu strip item



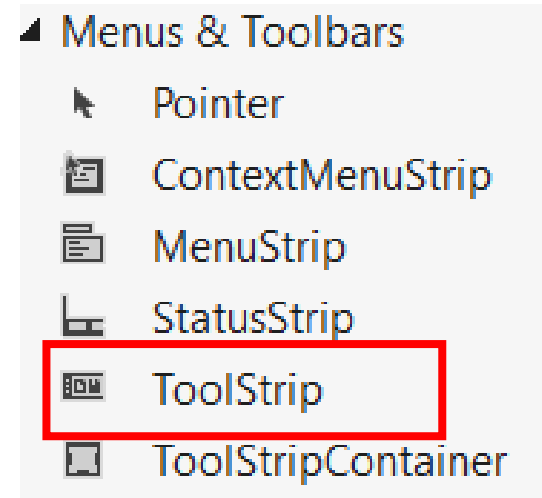
جامعة بغداد / كلية التربية للعلوم الصرفة ابن الهيثم / قسم علوم الحاسبات
المرحلة الثالثة صباحي / مسائي (نظري)
السنة ٢٠٢١-٢٠٢٢
اساتذة المادة: شهلاء طالب

Visual Programming

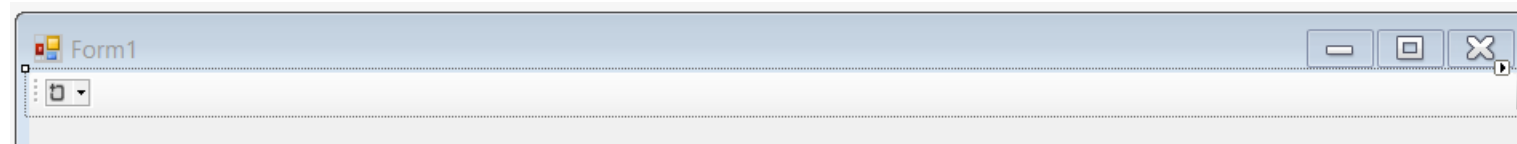
Lecture 12 – Working with ToolStrip Control & ContextMenuStrip

ToolStrip Control

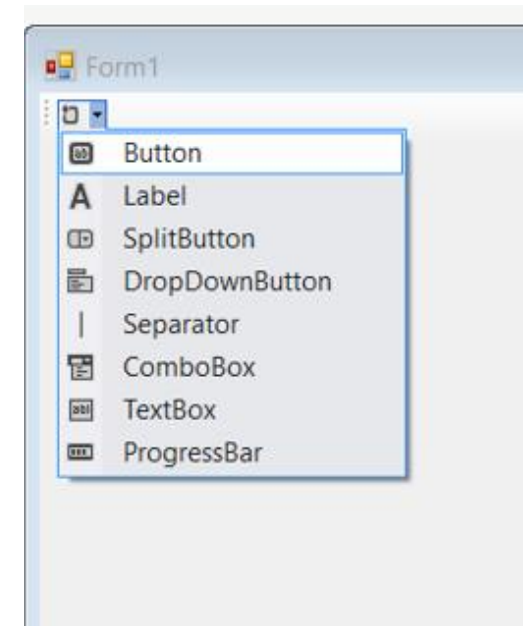
ToolStrip Control is an control object that used in the VB.NET windows application to create ToolBar. We can find it on toolbox under **Menus & Toolbar**. Double click add it into Form.



Control will stick on the top of Form.



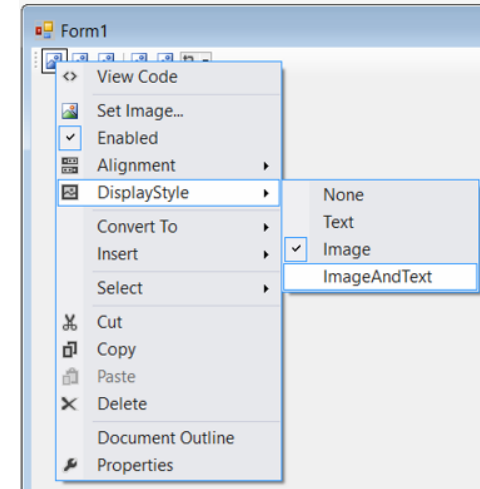
8 types of control can be added to ToolStrip.



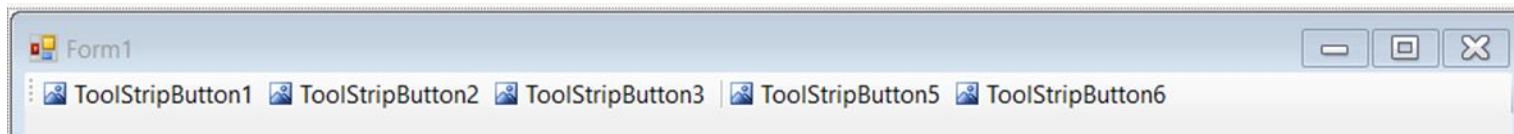
For initiation, we'll add buttons which are Save, Edit, Delete, Cancel, and Close. Set focus on toolstrip then click the dropdown and choose "button" as above image. Do it again to get 3 buttons added. After adding 3 buttons, add a Separator, then add 2 buttons again. The result should be as following:



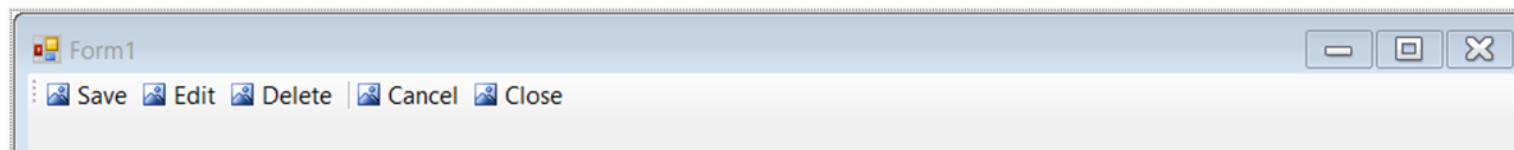
Give focus on first button image, right-click --> DisplayStyle --> ImageAndText. This step will change the button style displaying Image and Text. Do again for 4 other buttons.



Below image shows the result.

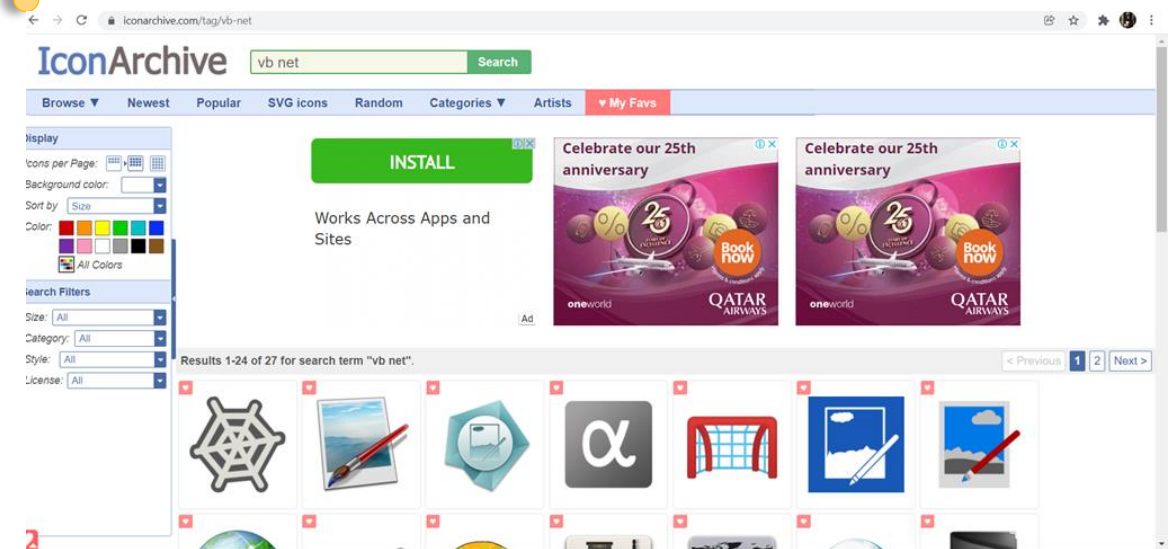


Next, we'll change the button text. Give focus on button then go to properties box to change Text. Changes each button's text with Save, Edit, Delete, Cancel, and Close.

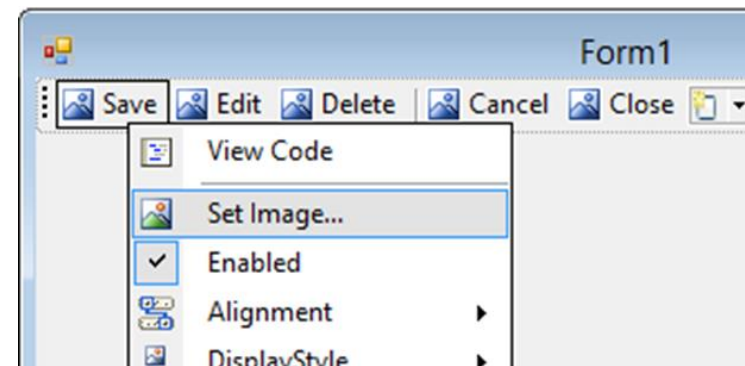


The next step is to set an image for each button. Set focus on the button, right-click and click **Set Image...**

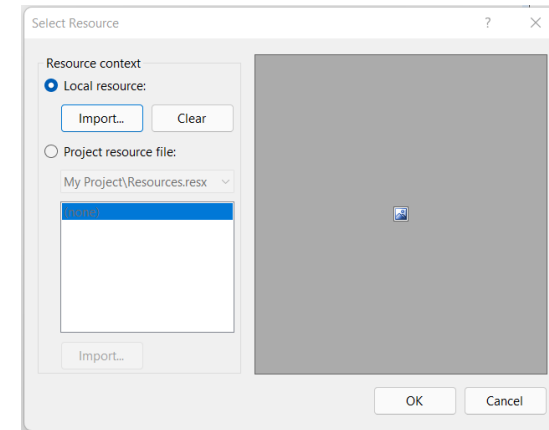
<https://iconarchive.com/tag/vb-net>



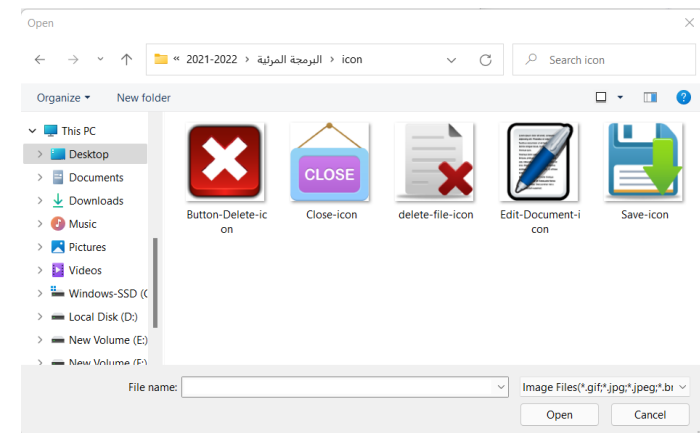
The next step is to set an image for each button. Set focus on the button, right-click and click **Set Image...**



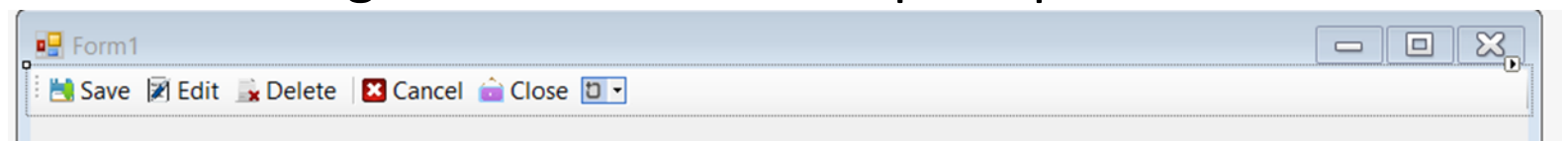
A "Select Source" dialog will be prompted. Choose option "Local resource" then click Import button.



Chose image files that represent the button's function, consider the appropriate size of images. When the image has been chosen click **Open...**



The chosen image will be set as the image of the button. Repeat process for 4 other buttons.



```
Private Sub ToolStripButton1_Click(sender As Object, e As EventArgs) Handles  
ToolStripButton1.Click  
    MsgBox("Save Button Clicked")  
End Sub
```

```
Private Sub ToolStripButton2_Click(sender As Object, e As EventArgs) Handles  
ToolStripButton2.Click  
    MsgBox("Edit Button Clicked")  
End Sub
```

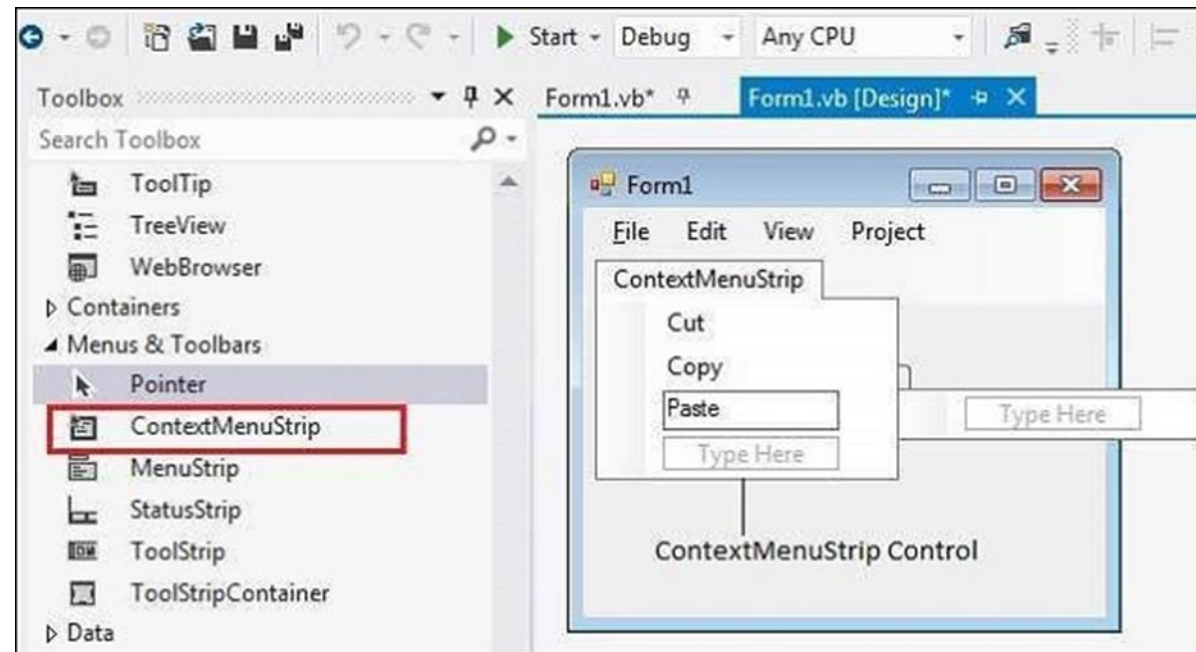
```
Private Sub ToolStripButton3_Click(sender As Object, e As EventArgs) Handles  
ToolStripButton3.Click  
    MsgBox("Delete Button Clicked")  
End Sub
```

```
Private Sub ToolStripButton5_Click(sender As Object, e As EventArgs) Handles  
ToolStripButton5.Click  
    MsgBox("Cancel Button Clicked")  
End Sub
```

```
Private Sub ToolStripButton6_Click(sender As Object, e As EventArgs) Handles  
ToolStripButton6.Click  
    MsgBox("Close Button Clicked")  
End Sub
```

ContextMenuStrip

The **ContextMenuStrip** control represents a shortcut menu that pops up over controls, usually when you right click them. They appear in context of some specific controls, so are called context menus. For example, Cut, Copy or Paste options.



Example

In this example, let us add a content menu with the menu items Cut, Copy and Paste.

Take the following steps –

Drag and drop or double click on a ControlMenuStrip control to add it to the form.

Add the menu items, Cut, Copy and Paste to it.

Add a RichTextBox control on the form.

Set the ContextMenuStrip property of the RichTextBox to ContextMenuStrip1 using the properties window.

Double Click the menu items and add following codes in the Click event of these menus –


```
Private Sub CutToolStripMenuItem_Click(sender As Object, e As  
EventArgs) Handles CutToolStripMenuItem.Click
```

```
    RichTextBox1.Cut()
```

```
End Sub
```

```
Private Sub CopyToolStripMenuItem_Click(sender As Object, e As  
EventArgs) Handles CopyToolStripMenuItem.Click
```

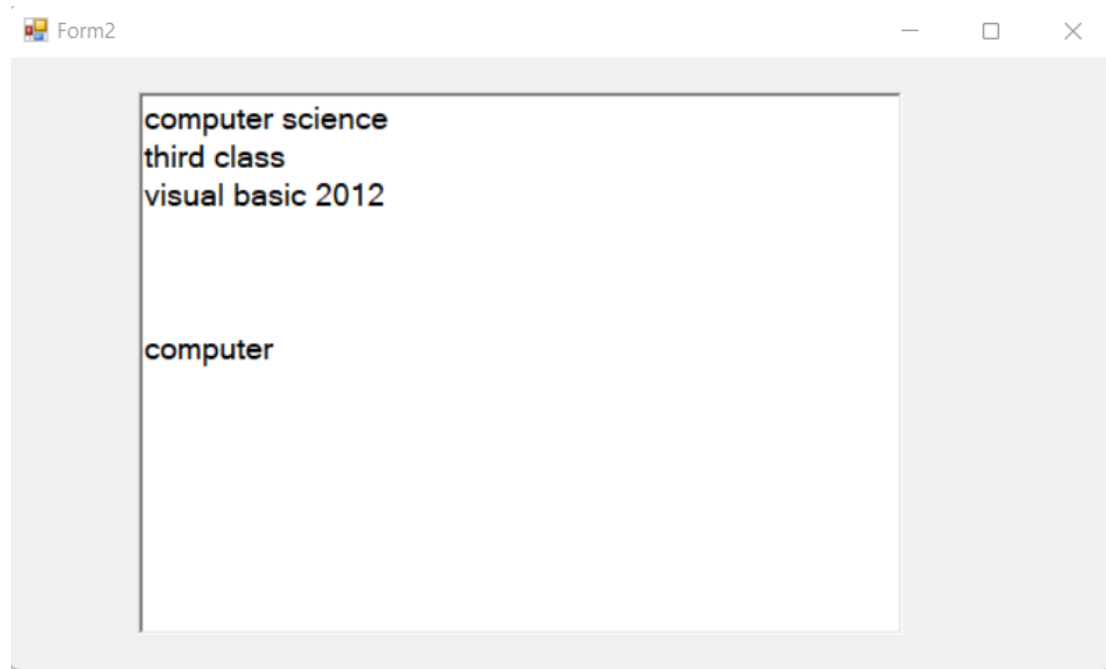
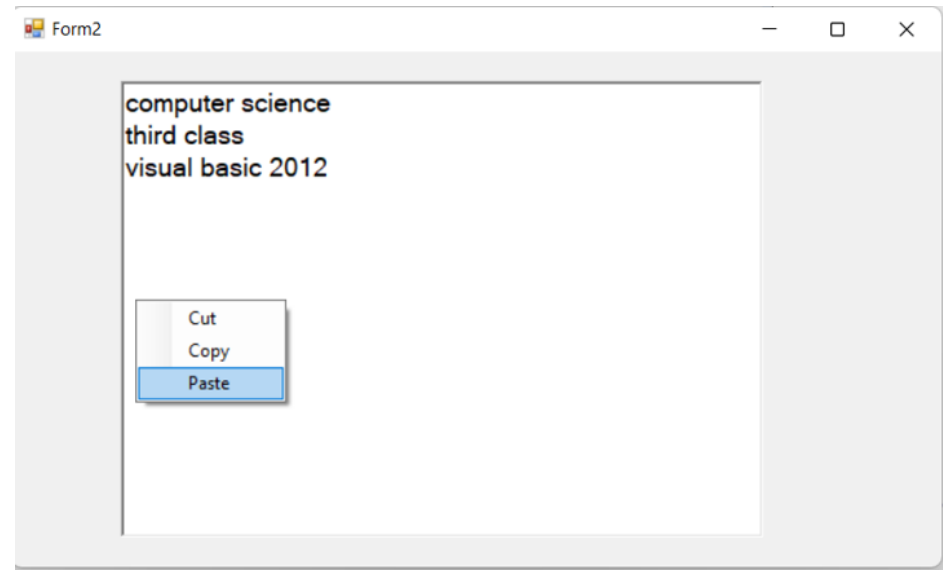
```
    RichTextBox1.Copy()
```

```
End Sub
```

```
Private Sub PasteToolStripMenuItem_Click(sender As Object, e As  
EventArgs) Handles PasteToolStripMenuItem.Click
```

```
    RichTextBox1.Paste()
```

```
End Sub
```



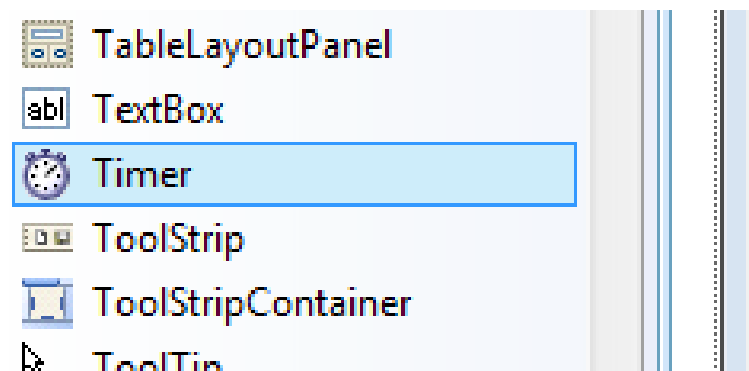
جامعة بغداد / كلية التربية للعلوم الصرفة ابن الهيثم / قسم علوم الحاسبات
المرحلة الثالثة صباحي / مسائي (نظري)
السنة 2021-2022
اساتذة المادة: شهلاء طالب

Visual Programming

Lecture 13 – Timer Control

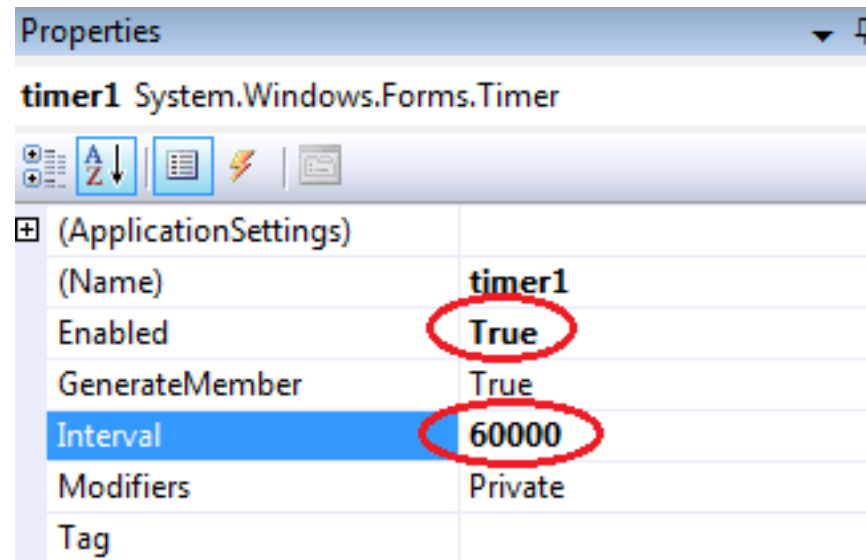
Timer Control

The timer is used to manage events that are time-related. We can use Timer Control in many situations in our development environment. If you want to run some code after a certain interval of time continuously, you can use the Timer control. As well as to start a process at a fixed time schedule, to increase or decrease the speed in an animation graphics with time schedule etc. you can use the Timer Control. The Visual Studio toolbox has a Timer Control that allowing you to drag and drop the timer controls directly onto a Windows Forms designer. At runtime it does not have a visual representation and works as a component in the background.



How to work with Timer Control ?

- With the Timer Control, we can control programs in millisecond, seconds, minutes and even in hours. The Timer Control allows us to set Interval property in milliseconds (1 second is equal to 1000 milliseconds). For example, if we want to set an interval of two minute we set the value at Interval property as 120000, means 120x1000 .
- The Timer Control starts its functioning only after its Enabled property is set to True, by default Enabled property is False.



Timer Example

The following program shows a Timer example that display current system time in a Label control. For doing this, we need one Label control and a Timer Control. Here in this program, we can see the Label Control is updated each seconds because we set Timer Interval as 1 second, that is 1000 milliseconds. After drag and drop the Timer Control in the designer form , double click the Timer control and set the DateTime.Now.ToString to Label control text property.

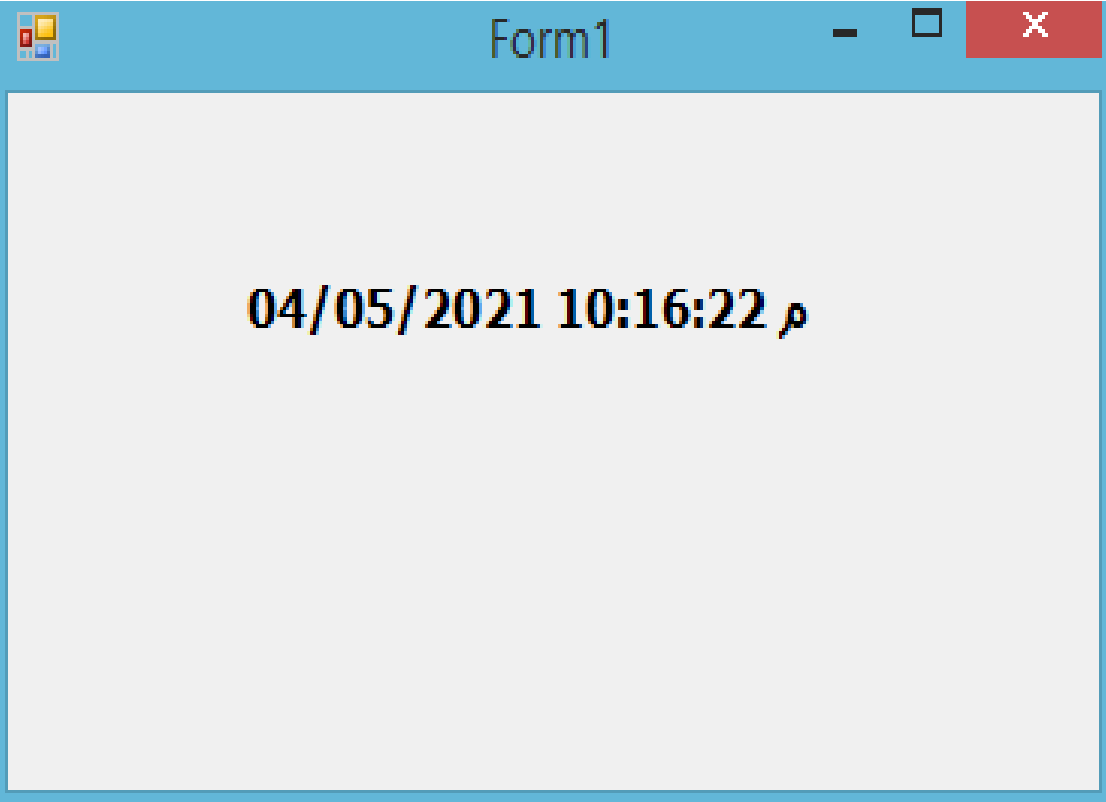
```
Private Sub Timer1_Tick(sender As Object, e As EventArgs)
    Handles Timer1.Tick
        Label1.Text = DateTime.Now.ToString
End Sub
```

Implementation

Before



After



Start and Stop Timer Control

We can control the Timer Control Object that when it start its function as well as when it stop its function. The Timer Control has a start and stop methods to perform these actions.

```
Timer1.Start() 'Timer starts functioning
```

```
Timer1.Stop() 'Timer stops functioning
```

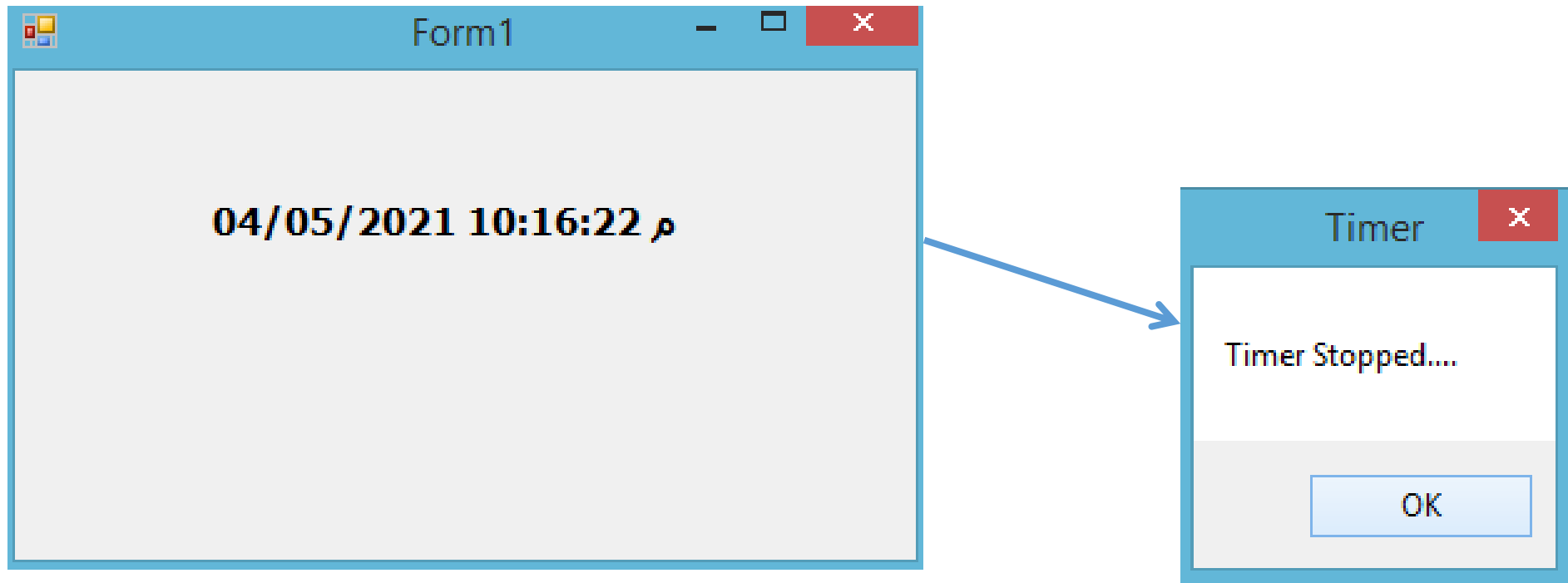
Here is an example for start and stop methods of the Timer Control. In this example we run this program only 10 seconds. So we start the Timer in the Form_Load event and stop the Timer after 10 seconds. We set timer Interval property as 1000 milliseconds (1 second) and in run time the Timer will execute 10 times its Tick event.

Code

```
Public Class Form2
    Dim second As Integer
    Private Sub Form2_Load(sender As Object, e As EventArgs) Handles
MyBase.Load
        Timer1.Interval = 1000
        Timer1.Start() 'Timer starts functioning
    End Sub

    Private Sub Timer1_Tick(sender As Object, e As EventArgs) Handles
Timer1.Tick
        Label1.Text = DateTime.Now.ToString
        Second = Second + 1
        If Second >= 10 Then
            Timer1.Stop() 'Timer stops functioning
            MsgBox("Timer Stopped....")
        End If
    End Sub
End Class
```

Implementation



Example : Creating a simple stopwatch

We can create a simple stopwatch using the Timer control. Start a new project and name it stopwatch. Change the Form1 text to Stopwatch. Insert the Timer control into the form and set its interval to 1000 which is equal to one second. Besides, set the timer Enabled property to False so that it will not start ticking when the program is started. Insert three buttons and change their names to StartBtn, StopBtn and ResetBtn respectively. Change their text to “Start”, “Stop” and “Reset” accordingly. Now, key in the code as follows:

```
Public Class Form3
```

```
    Private Sub Timer1_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick
```

```
        'To increase one unit per second
```

```
        Label1.Text = Val(Label1.Text) + 1
```

```
    End Sub
```

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

```
        'To start the Timer
```

```
        Timer1.Enabled = True
```

```
    End Sub
```

```
    Private Sub Button2_Click(sender As Object, e As EventArgs) Handles Button2.Click
```

```
        'To stop the Timer
```

```
        Timer1.Enabled = False
```

```
    End Sub
```

```
    Private Sub Button3_Click(sender As Object, e As EventArgs) Handles Button3.Click
```

```
        'To reset the Timer to 0
```

```
        Label1.Text = 0
```

```
    End Sub
```

```
End Class
```

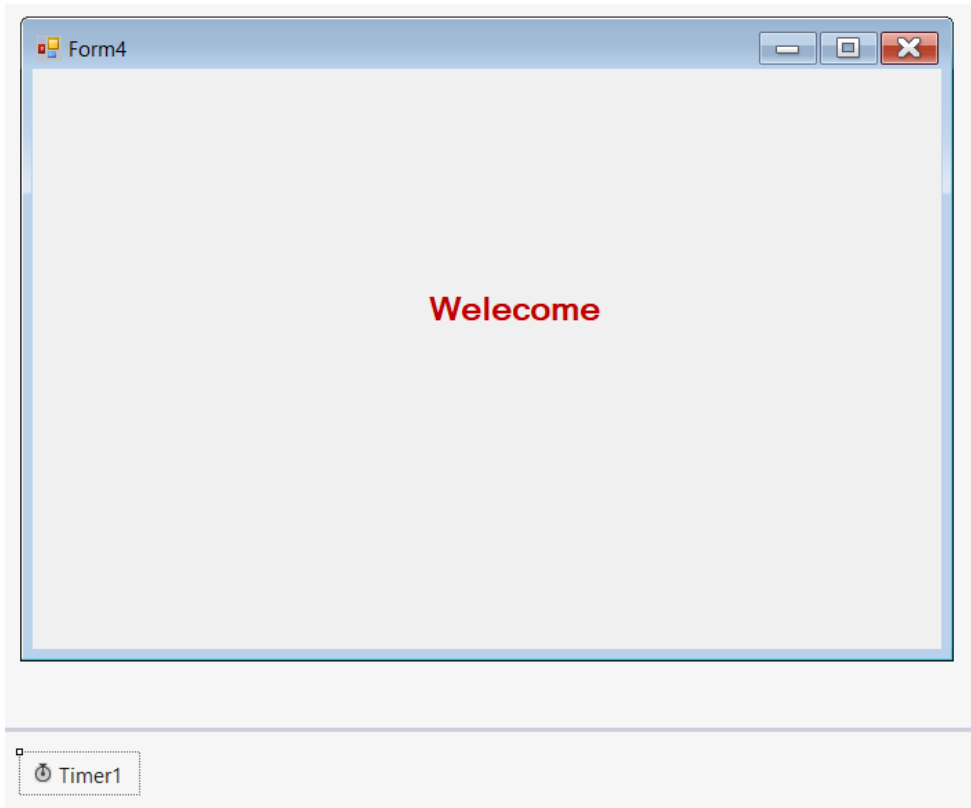
Implementation

The Interface of the Stopwatch is as shown below:

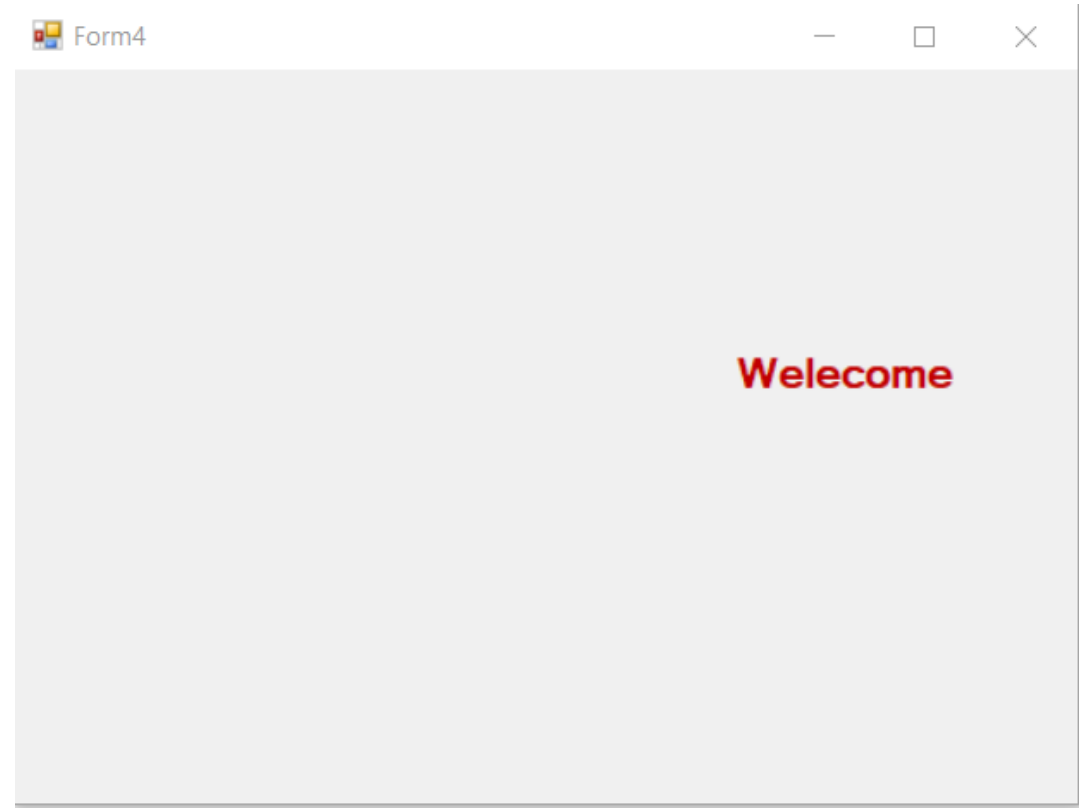


Example

Before



After



Code

```
Public Class Form3
```

```
Dim a As Integer
```

```
Private Sub Timer1_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick
```

```
    Label1.Left = Label1.Left + 5 * a
```

```
    If Label1.Left >= Me.Width Then
```

```
        a = -1
```

```
    End If
```

```
End Sub
```

```
Private Sub Form3_Load(sender As Object, e As EventArgs) Handles MyBase.Load
```

```
    a = 1
```

```
End Sub
```

```
End Class
```

Homework

Design a program that contains PictureBox1 and Timer, when Timer starts the size of PictureBox1 increase every one second after ten seconds the size of PictureBox1 to be decreases every one second and the Timer to be stopped after 20 seconds

Design a program that's contents Lable1, Button1 and Timer1 , when press the Button1 the lable1 will be a countdown , The timer stop when the Label1 equal zero and show message "The Time is Over".

جامعة بغداد / كلية التربية للعلوم الصرفة ابن الهيثم / قسم علوم الحاسبات
المرحلة الثالثة صباحي / مسائي (نظري)
السنة 2022-10
اساتذة المادة: شهلاء طالب

Visual Basic 2012

Lecture 14 – Common Dialogs

Dialog Box

A Dialog box is a temporary Window for an application that accepts user response through mouse or keyboard to open a file, save a file, notifications, alert messages, color, print, openfile dialog box, etc. It is also useful to create communication and interaction between the user and the application. Furthermore, the dialog box appears in a form when the program needs to interact with users, such as when an error occurs, an alert message, acknowledgment from the user or when the program requires immediate action or whether the decision is to be saved based on the changes.

All VB.NET Dialog box inherits the CommonDialog class and overrides the RunDialog() method of the base class to create the OpenFileDialog box, PrintDialogbox, Color, and Font Dialog box. The RunDialog() method is automatically called in a Windows form when the dialog box calls its ShowDialog() method.

ShowDialog method

There are many built-in dialog boxes to be used in Windows forms for various tasks like opening and saving files, printing a page, providing choices for colors, fonts, page setup, etc., to the user of an application. These built-in dialog boxes reduce the developer's time and workload.

All of these dialog box control classes inherit from the **CommonDialog** class and override the *RunDialog()* function of the base class to create the specific dialog box.

The *RunDialog()* function is automatically invoked when a user of a dialog box calls its *ShowDialog()* function.

The **ShowDialog** method is used to display all the dialog box controls at run-time. It returns a value of the type of **DialogResult** enumeration. The values of DialogResult enumeration are –

Abort – returns DialogResult.Abort value, when user clicks an Abort button.

Cancel – returns DialogResult.Cancel, when user clicks a Cancel button.

Ignore – returns DialogResult.Ignore, when user clicks an Ignore button.

No – returns DialogResult.No, when user clicks a No button.

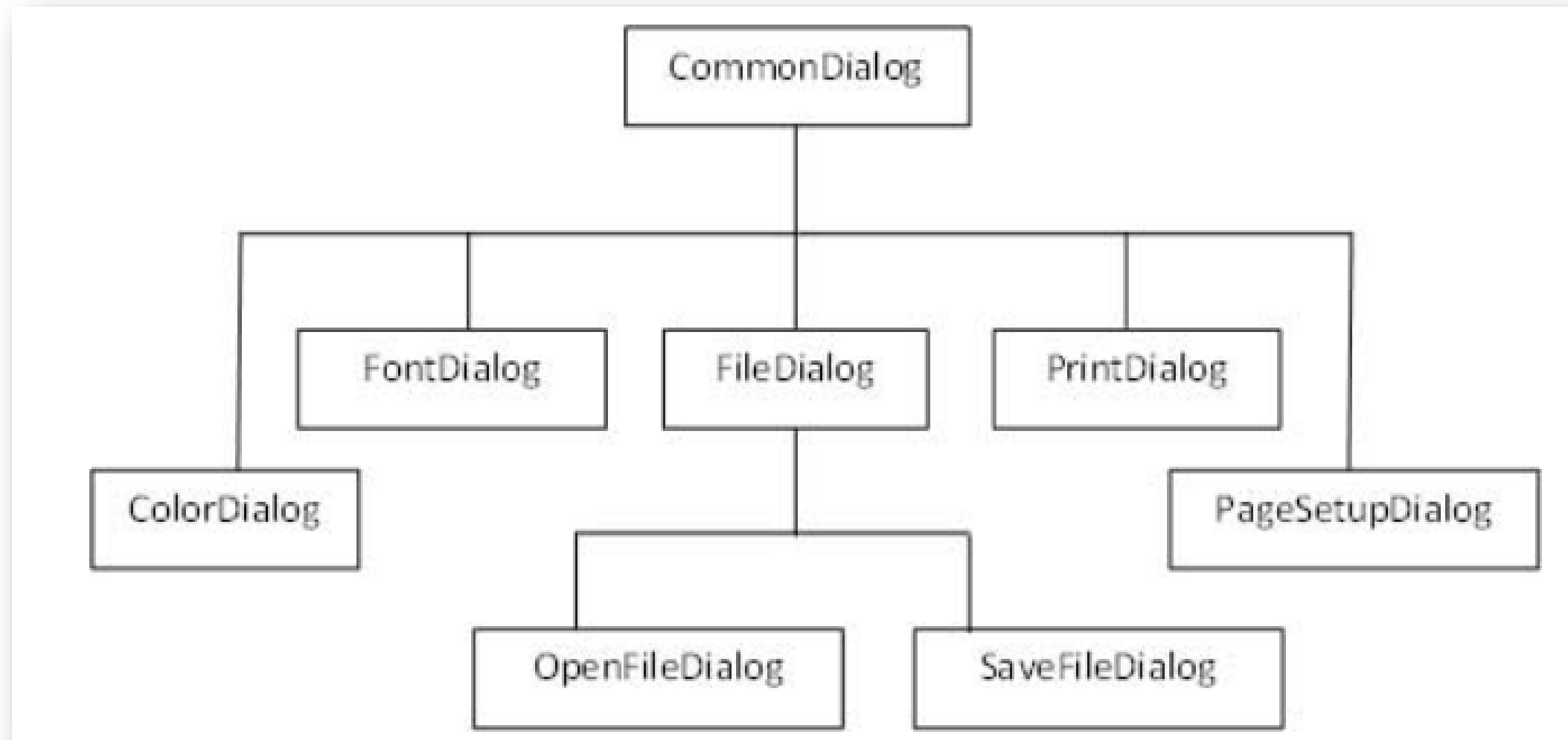
None – returns nothing and the dialog box continues running.

OK – returns DialogResult.OK, when user clicks an OK button

Retry – returns DialogResult.Retry , when user clicks an Retry button

Yes – returns DialogResult.Yes, when user clicks an Yes button

Common Dialog Class Inheritance



Common Dialogs

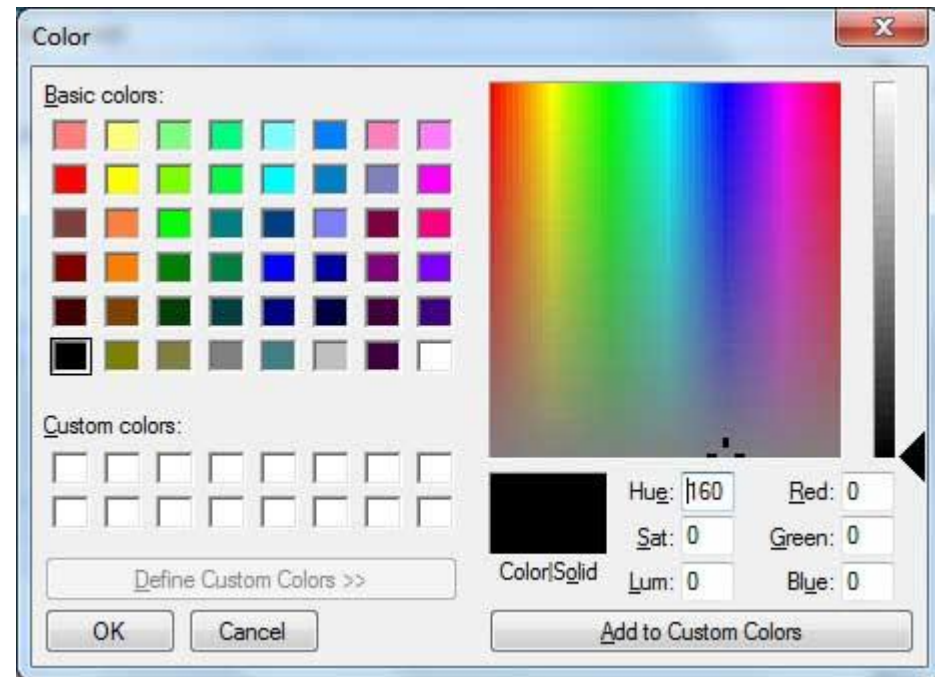
Sr.No.	Control & Description
1	<p>ColorDialog ↗</p> <p>It represents a common dialog box that displays available colors along with controls that enable the user to define custom colors.</p>
2	<p>FontDialog ↗</p> <p>It prompts the user to choose a font from among those installed on the local computer and lets the user select the font, font size, and color.</p>
3	<p>OpenFileDialog ↗</p> <p>It prompts the user to open a file and allows the user to select a file to open.</p>
4	<p>SaveFileDialog ↗</p> <p>It prompts the user to select a location for saving a file and allows the user to specify the name of the file to save data.</p>
5	<p>PrintDialog ↗</p> <p>It lets the user to print documents by selecting a printer and choosing which sections of the document to print from a Windows Forms application.</p>

ColorDialog control

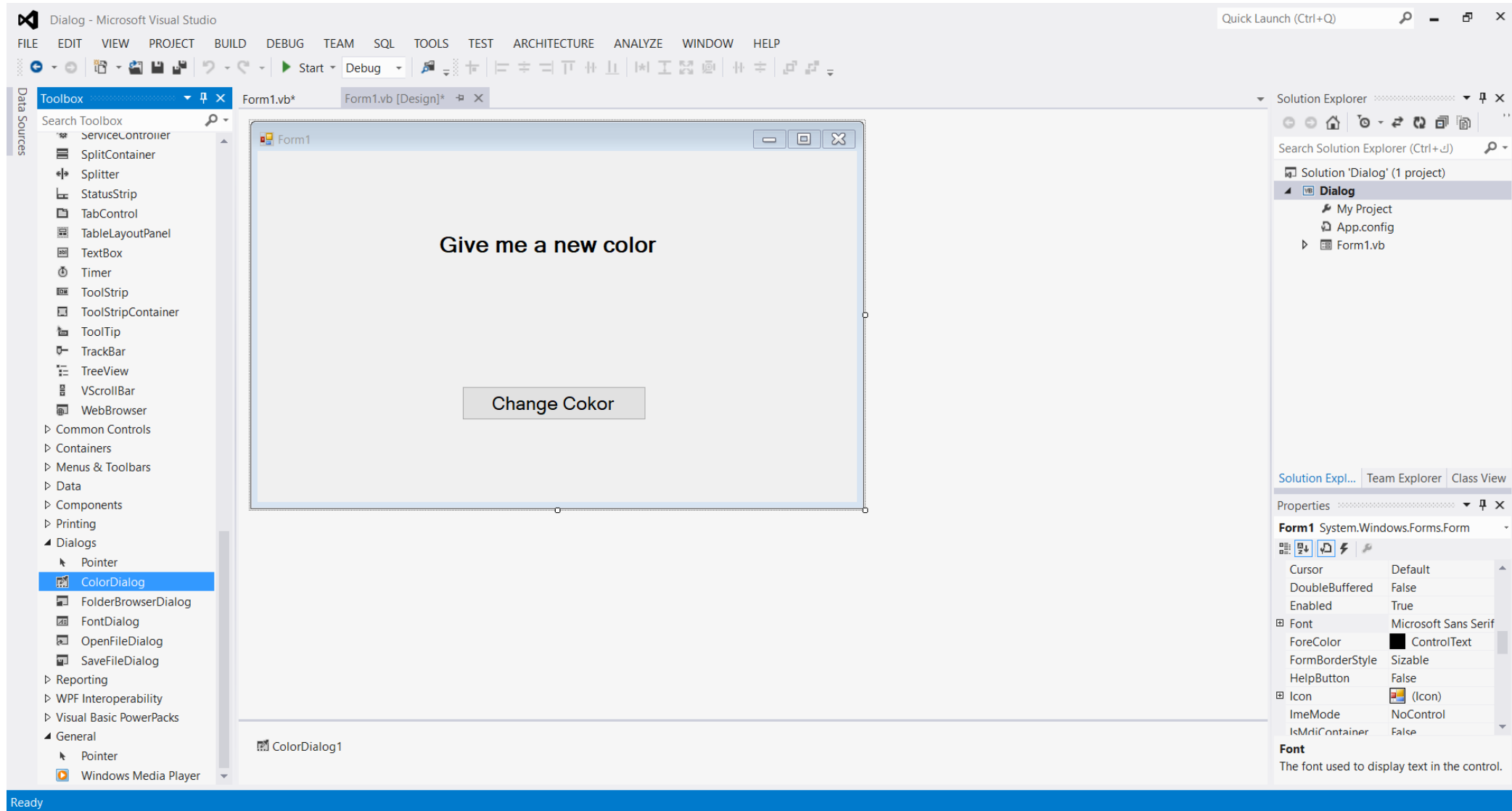
The ColorDialog control class represents a common dialog box that displays available colors along with controls that enable the user to define custom colors. It lets the user select a color.

The main property of the ColorDialog control is *Color*, which returns a **Color** object.

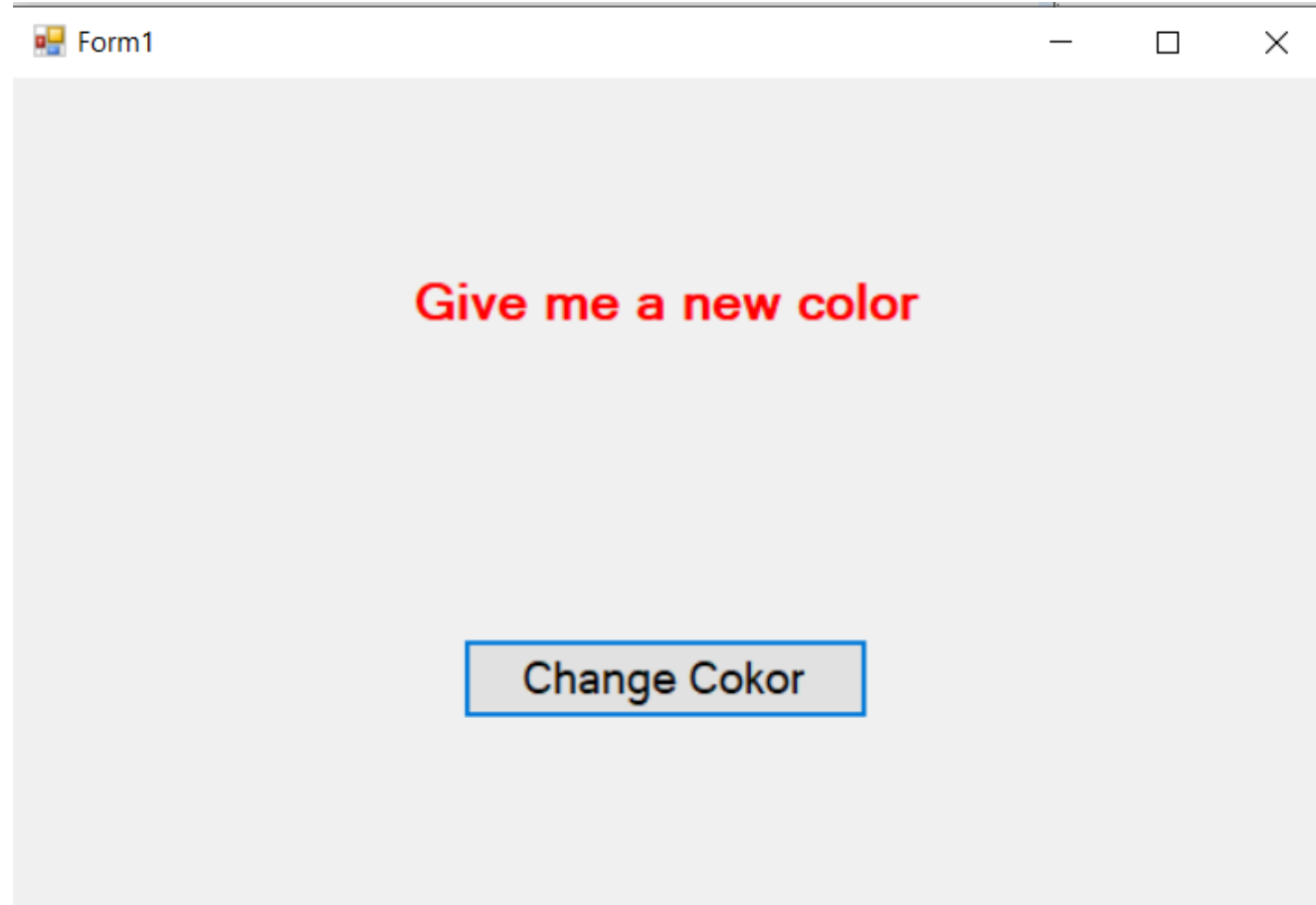
Following is the Color dialog box –



Example of ColorDialog control



Implementation



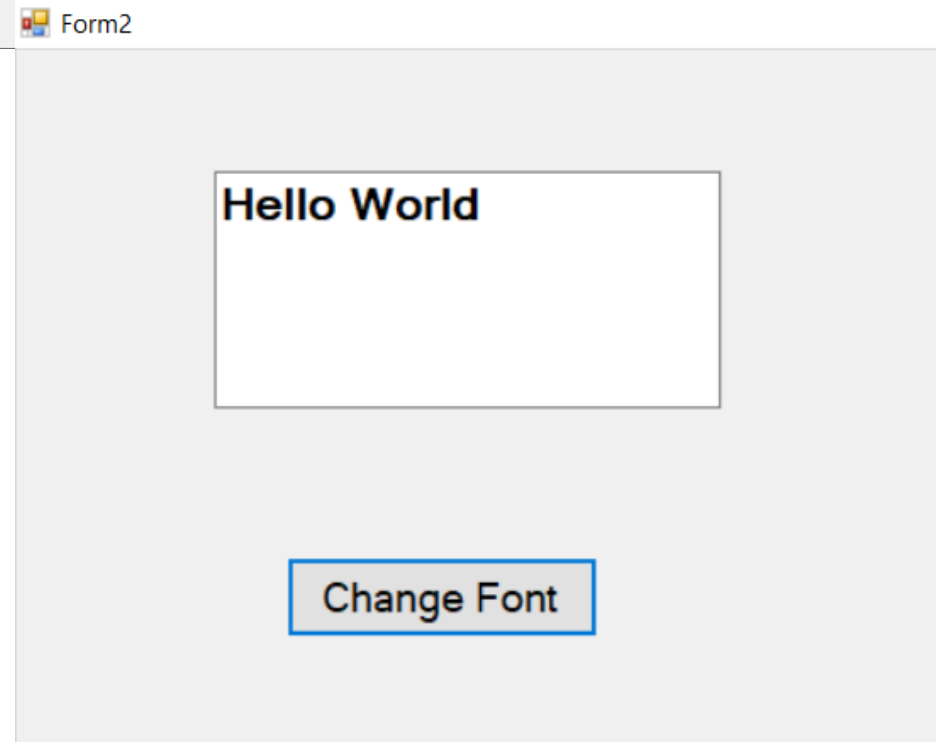
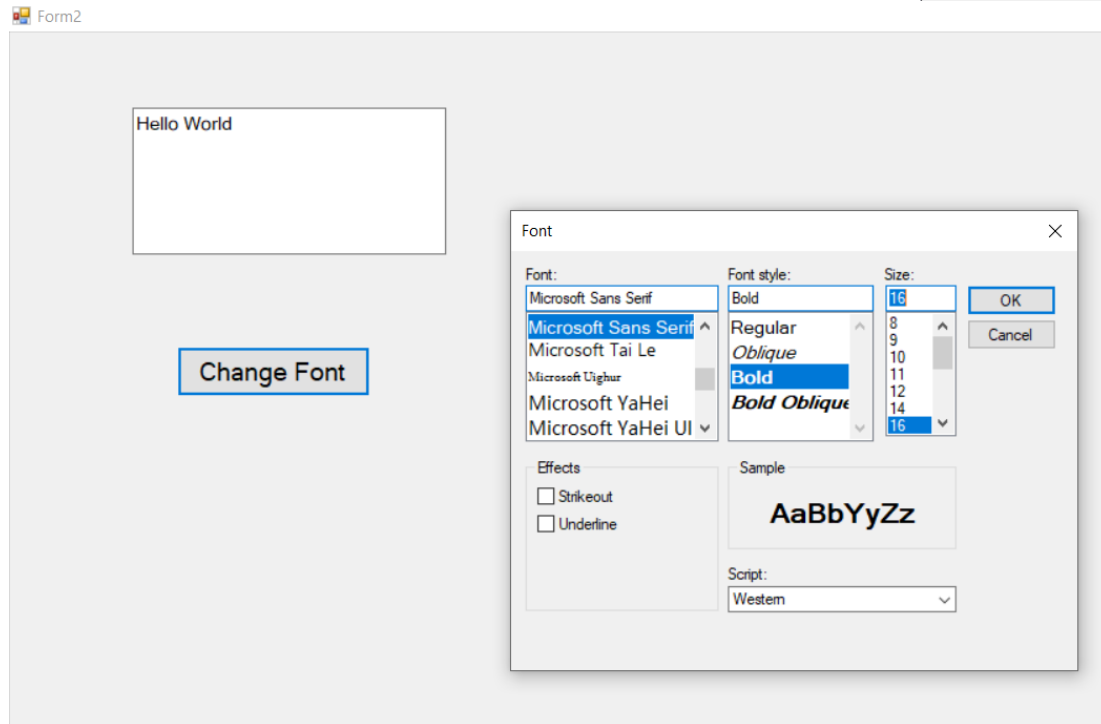
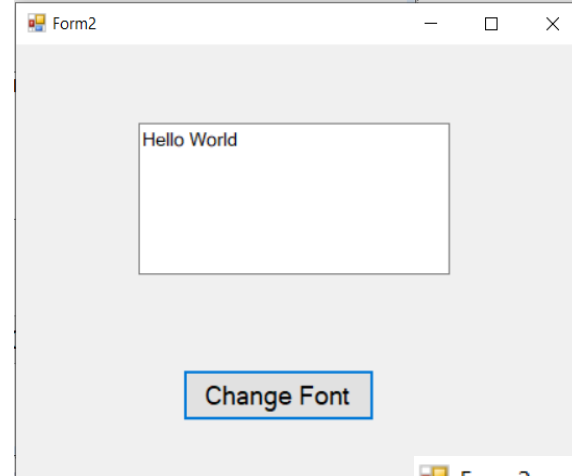
FontDialog Control

It prompts the user to choose a font from among those installed on the local computer and lets the user select the font, font size, and color. It returns the Font and Color objects.

Following is the Font dialog box –



Example of FontDialog Control



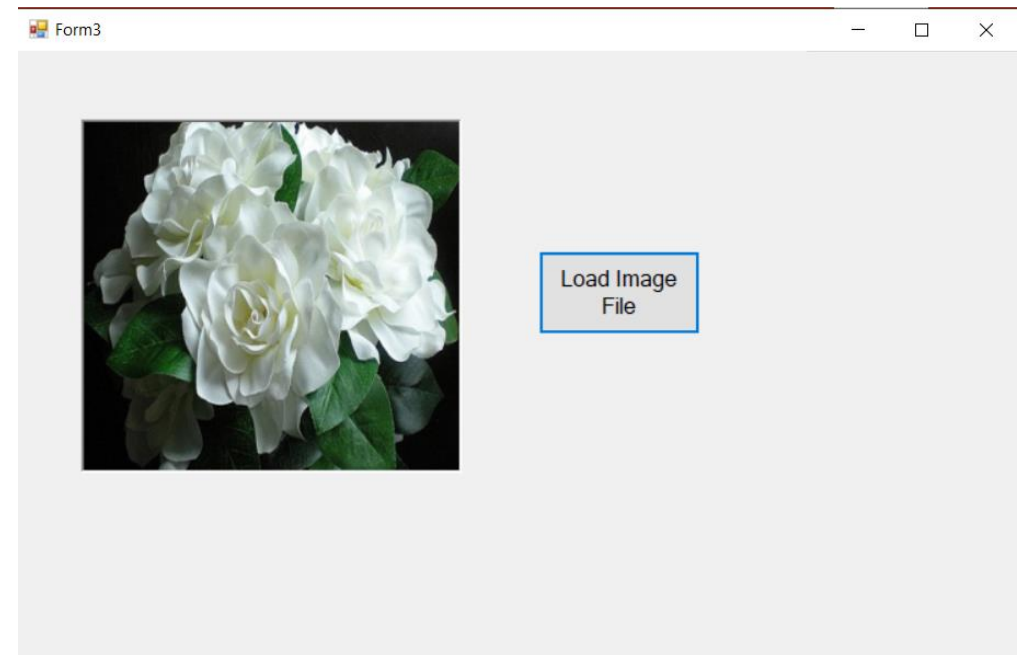
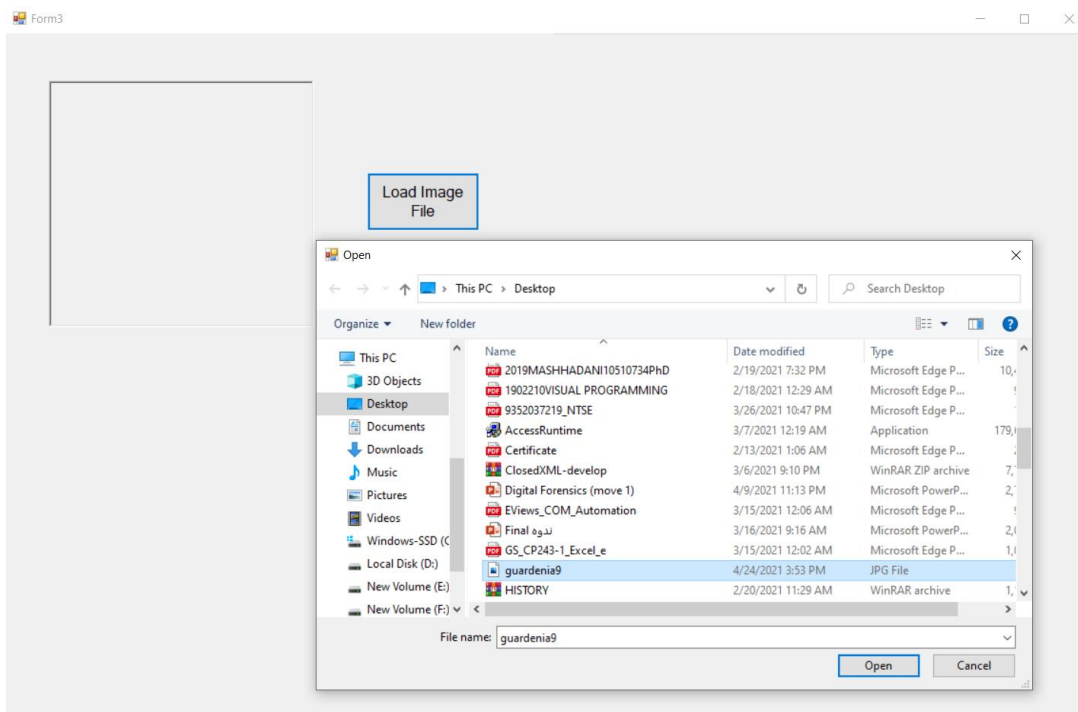
Code

```
If FontDialog1.ShowDialog <>  
Windows.Forms.DialogResult.Cancel Then  
    TextBox1.Font = FontDialog1.Font  
End If
```

OpenFileDialog Control

The **OpenFileDialog** control prompts the user to open a file and allows the user to select a file to open. The user can check if the file exists and then open it. The OpenFileDialog control class inherits from the abstract class **FileDialog**.

If the ShowReadOnly property is set to True, then a read-only check box appears in the dialog box. You can also set the ReadOnlyChecked property to True, so that the read-only check box appears checked.



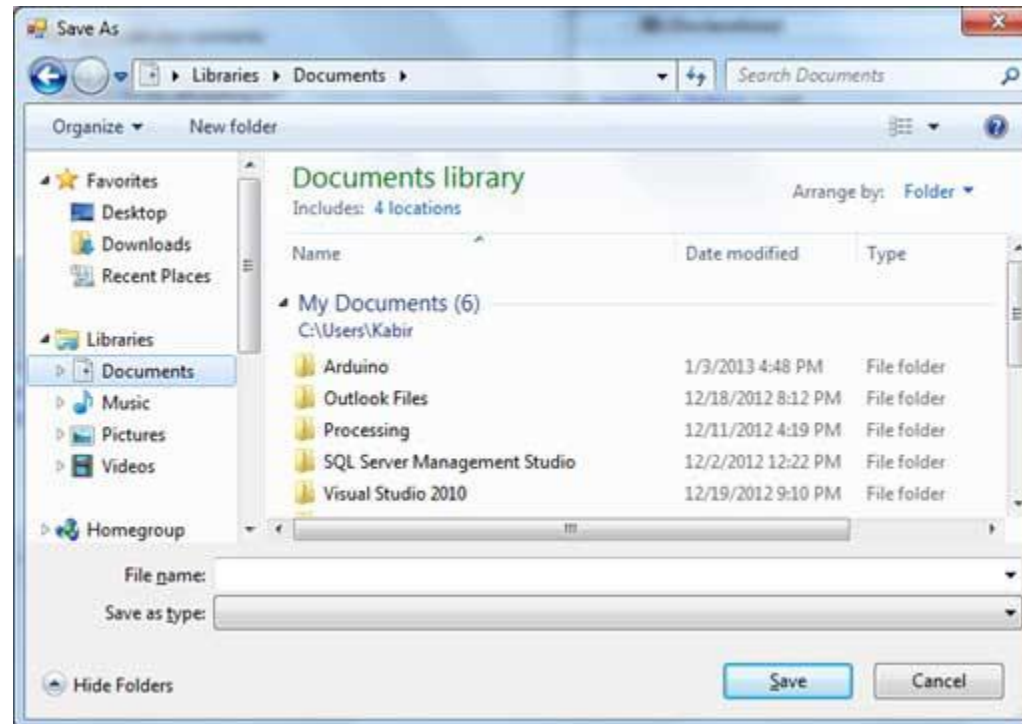
Code

```
If OpenFileDialog1.ShowDialog <>  
Windows.Forms.DialogResult.Cancel Then  
    PictureBox1.Image =  
Image.FromFile(OpenFileDialog1.FileName)  
End If
```

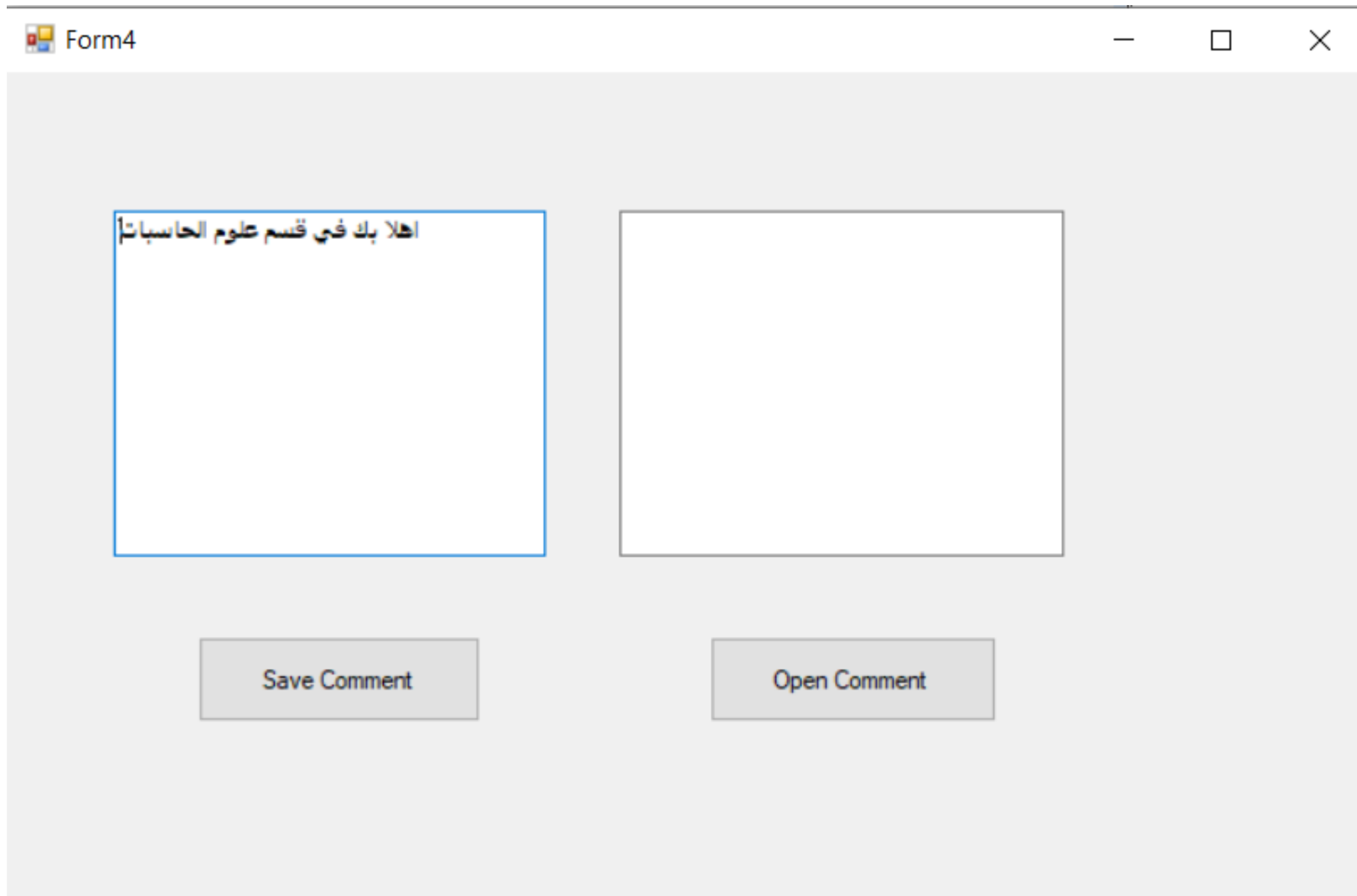
SaveFileDialog Control

The **SaveFileDialog** control prompts the user to select a location for saving a file and allows the user to specify the name of the file to save data. The **SaveFileDialog** control class inherits from the abstract class **FileDialog**.

Following is the Save File dialog box –



Example of SaveFileDialog Control and OpenFileDialog Control



اهلا بك في قسم علوم الحاسبات

Save Comment

Open Comment

Save As

< > > This PC > Desktop

Search Desktop

Organize New folder

Name	Date modified	Type	Size
.vs	3/22/2021 8:54 PM	File folder	
2020-2021	2/17/2021 11:25 PM	File folder	
Ahmed	2/28/2021 1:55 PM	File folder	
all downloads	3/13/2021 5:09 PM	File folder	
computer visiun	4/11/2021 1:26 AM	File folder	
HISTORY	2/20/2021 11:29 AM	File folder	
Mabani Al amal9-3-2021	3/9/2021 11:04 AM	File folder	
New folder	2/24/2021 12:36 AM	File folder	
New folder (2)	3/11/2021 8:56 PM	File folder	
Noor	3/11/2021 10:30 PM	File folder	

File name: computer

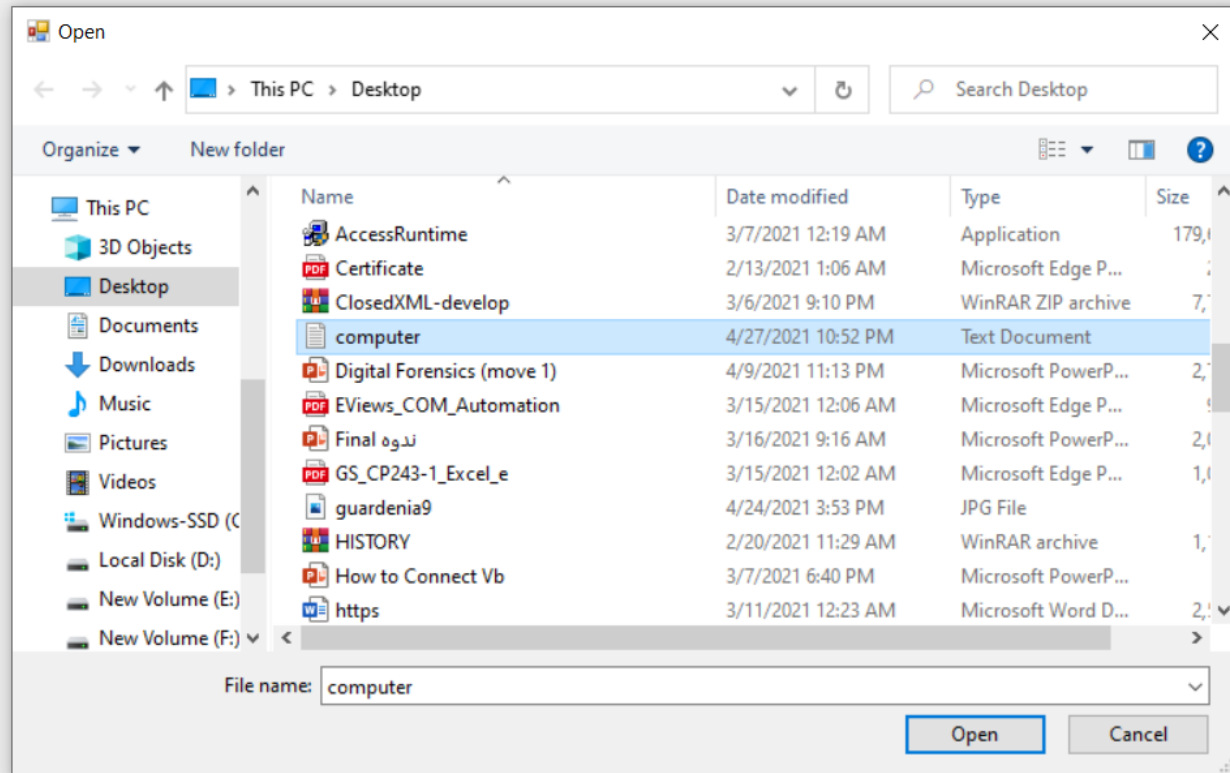
Save as type: TXT Files (*.txt)

Save Cancel

اهلا بك في قسم علوم الحاسبات

Save Comment

Open Comment



Form4

اهلا بك في قسم علوم الحاسبات

Save Comment

اهلا بك في قسم علوم الحاسبات

Open Comment

Code

```
Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
    SaveFileDialog1.Filter = "TXT Files (*.txt)|*.txt"
    If SaveFileDialog1.ShowDialog = Windows.Forms.DialogResult.OK _
        Then
        My.Computer.FileSystem.WriteAllText _
            (SaveFileDialog1.FileName, TextBox1.Text, True)
    End If
End Sub

Private Sub Button2_Click(sender As Object, e As EventArgs) Handles
Button2.Click
    If OpenFileDialog1.ShowDialog <> Windows.Forms.DialogResult.Cancel Then
        TextBox2.Text =
My.Computer.FileSystem.ReadAllText(OpenFileDialog1.FileName)
    End If
End Sub
```

جامعة بغداد / كلية التربية للعلوم الصرفة ابن الهيثم / قسم علوم الحاسبات
المرحلة الثالثة صباحي / مسائي (نظري)
السنة 2021-2022
اساتذة المادة: شهلاء طالب

Visual Programming

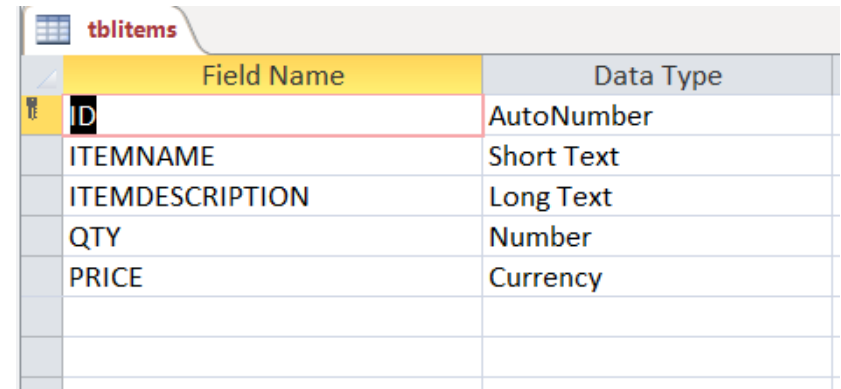
Lecture 15 – How to Connect Access Database in VB.Net

Steps How to Connect Access Database in VB.Net

Step 1: Create an MS Access Database. Open an MS Access Database in your Computer and Create a Blank Database and Save it as *“inventorydb.accdb”*.

Step 2: Create a Database Table.

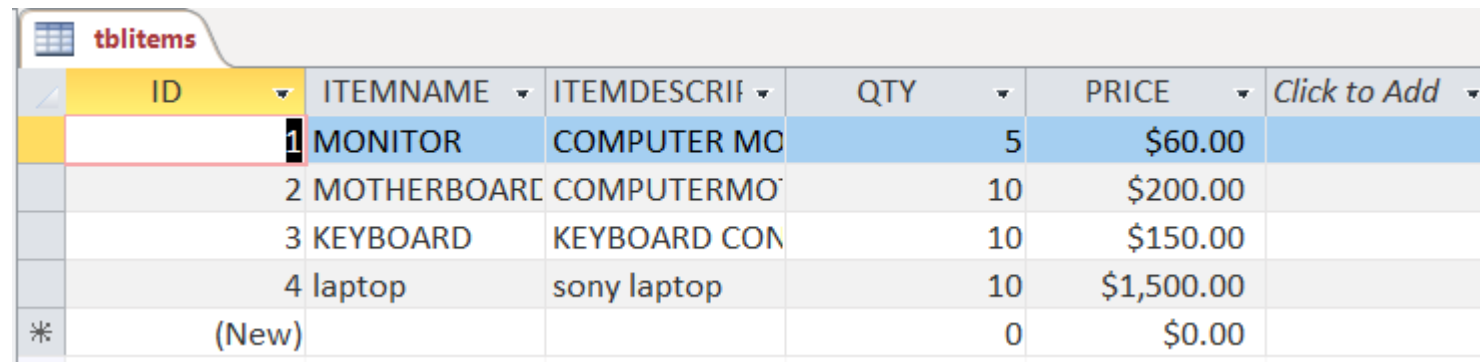
To create a table, follow the image below and save it as *“tblitems”*.



The screenshot shows the design view of a table named 'tblitems'. The table has five fields: ID (AutoNumber), ITEMNAME (Short Text), ITEMDESCRIPTION (Long Text), QTY (Number), and PRICE (Currency). The ID field is highlighted as the primary key.

Field Name	Data Type
ID	AutoNumber
ITEMNAME	Short Text
ITEMDESCRIPTION	Long Text
QTY	Number
PRICE	Currency

Step 3: Populate the table. Add sample records in the table. follow the sample records in the image below.



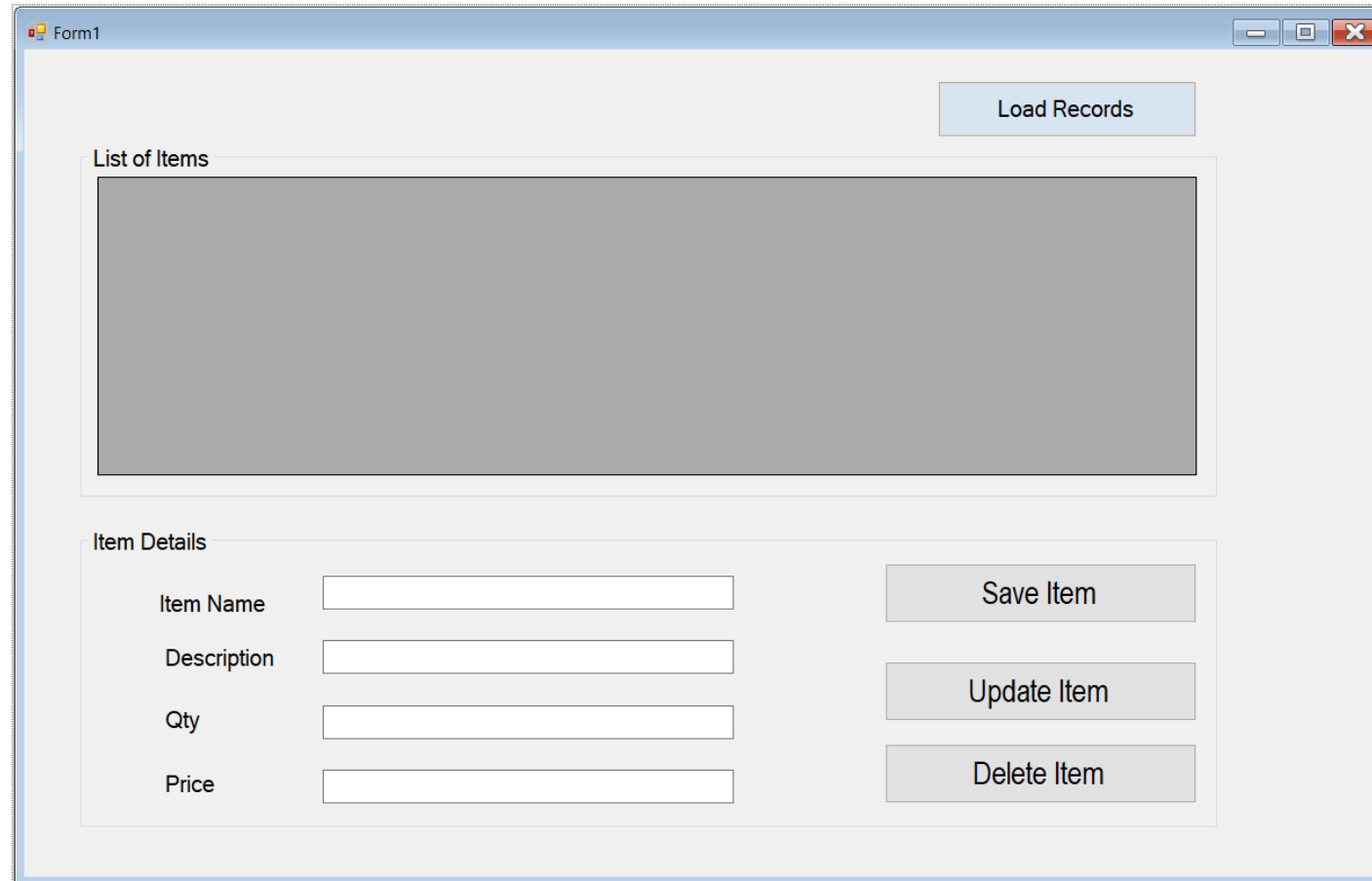
The screenshot shows the datasheet view of the 'tblitems' table. It contains five records, including a new record marked with an asterisk.

ID	ITEMNAME	ITEMDESCRIF	QTY	PRICE	Click to Add
1	MONITOR	COMPUTER MO	5	\$60.00	
2	MOTHERBOARD	COMPUTERMO	10	\$200.00	
3	KEYBOARD	KEYBOARD CON	10	\$150.00	
4	laptop	sony laptop	10	\$1,500.00	
*	(New)		0	\$0.00	

Steps How to Connect Access Database in VB.Net

Step 4: Create a VB.Net Application. Open Visual Studio and Create a Visual Basic Application project and Save it as *“connectvbaccess”*.

Step 5: Design the user interface. To design the form, you need to follow the image below.



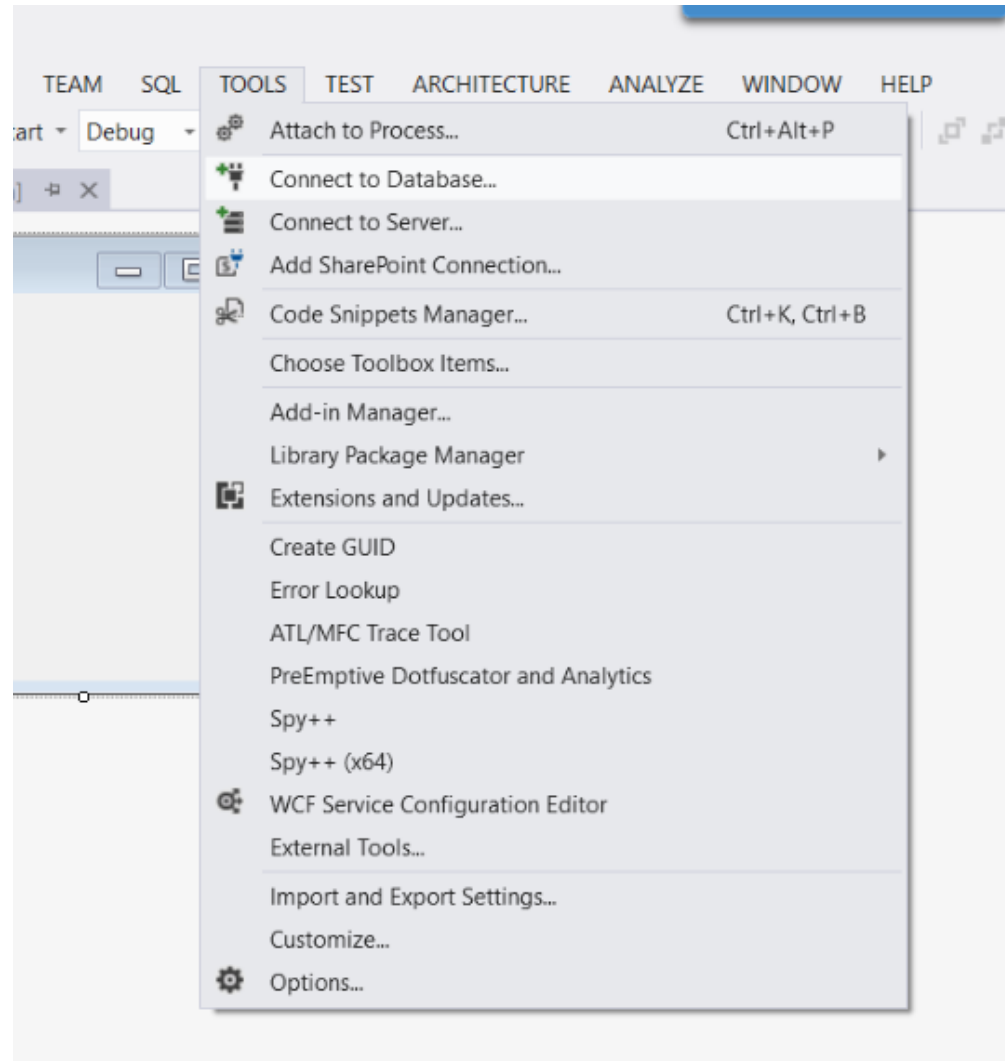
The screenshot displays a Windows form titled "Form1" with a standard title bar. The form is divided into two main sections:

- List of Items:** A large, empty rectangular area intended for displaying a list of records.
- Item Details:** A section containing four text input fields labeled "Item Name", "Description", "Qty", and "Price". To the right of these fields are three buttons: "Save Item", "Update Item", and "Delete Item".

At the top right of the form, there is a "Load Records" button. The form has a light gray background and a blue border.

Steps How to Connect Access Database in VB.Net

Step 6: Select TOOLS from Menu Bar → Connect to Database



Steps How to Connect Access Database in VB.Net

Step 7: Select the database name from Add Connection dialog box.

Add Connection ? X

Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source:
Microsoft Access Database File (OLE DB) Change...

Database file name:
Browse...

Log on to the database

User name: Admin

Password:

Save my password

Advanced...

Test Connection OK Cancel

Add Connection ? X

Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source:
Microsoft Access Database File (OLE DB) Change...

Database file name:
C:\Users\user\Documents\Database1.accdb Browse...

Log on to the database

User name: Admin

Password:

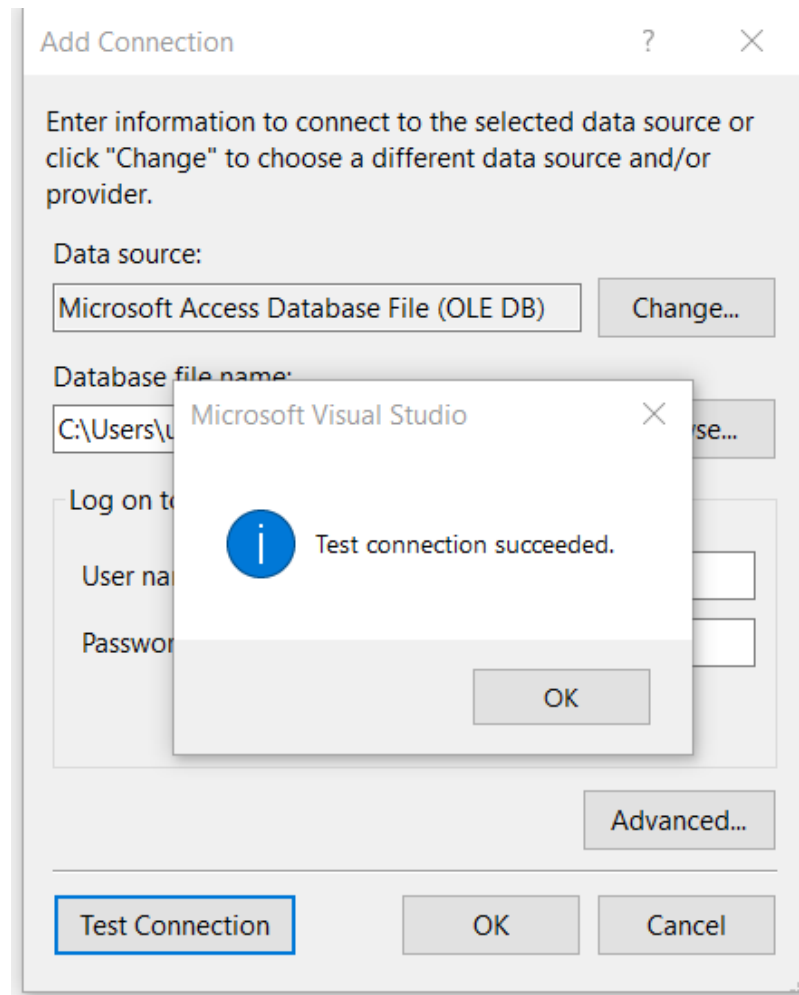
Save my password

Advanced...

Test Connection OK Cancel

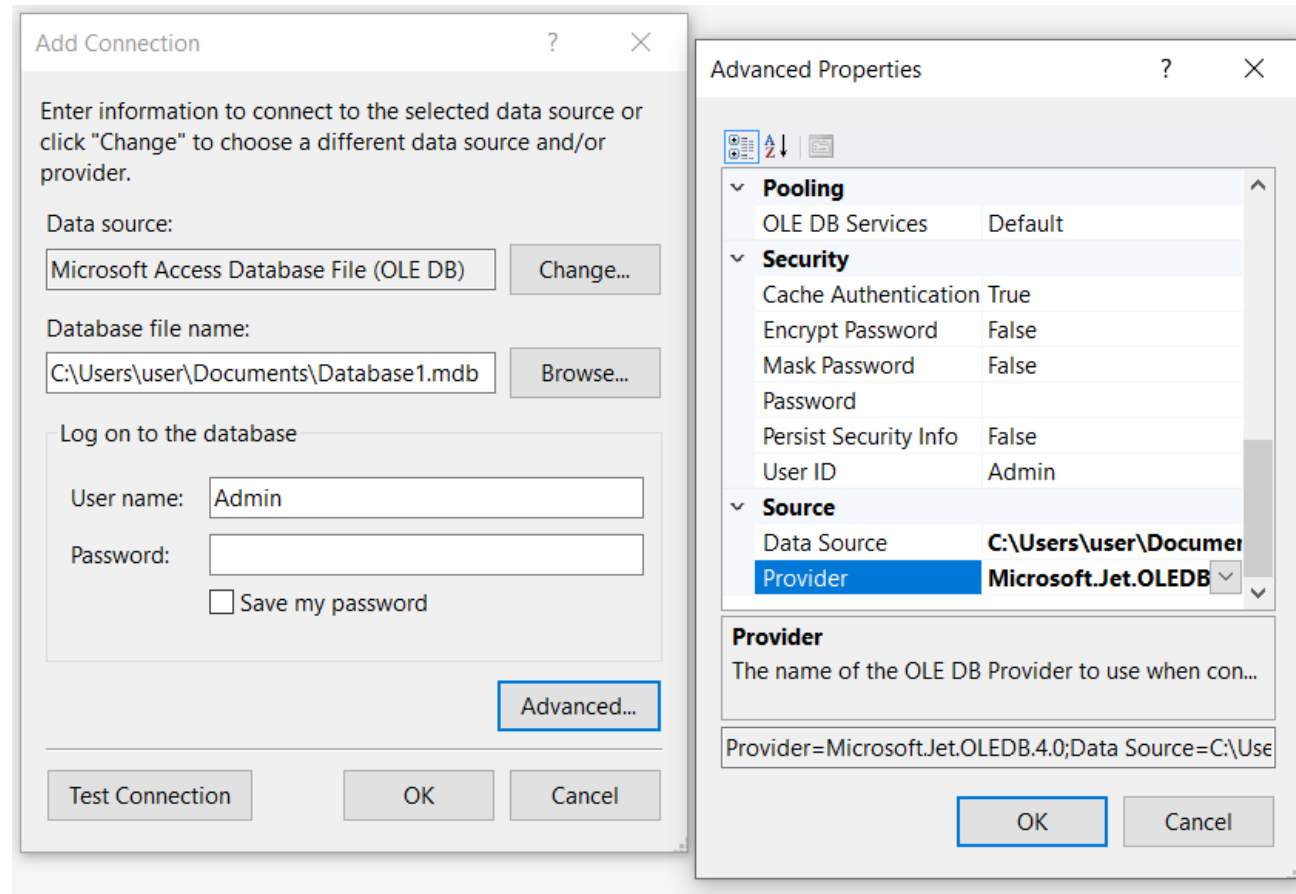
Steps How to Connect Access Database in VB.Net

Step 8: Test Connection To test the connection, click the “Test Connection” button, finally click “OK” button at the side of the “Cancel” button.



Steps How to Connect Access Database in VB.Net

Step 9: Copy the Connection String. Copy the connection string so that we can use this in our next step.



Code To Connect Access Database in VB.Net

- Add Imports System.Data.OleDb before **Public class Form1**
- Add the following code under **“Public Class Form1”**.

Dim con As New OleDbConnection

```
Dim c As String = "Provider=Microsoft.ACE.OLEDB.12.0;Data  
Source=C:\Users\lenovo 1\Documents\Database6.accdb"
```

Double Click on the Form1 and add the following code

```
con.ConnectionString = c
```

The Code above started with a **Declaration of Variable** name **“con”** with an **Ole Object Type OleDbConnection**.

Inside **OleDbConnection**, we pasted the connection string we copied from the **“Step 9”** instructions.

Test the Connection of Access database in VB.Net

To test the **Connection between ms access database and VB.Net**, Double click the **form1** add the following code under "Form1_Load" events.

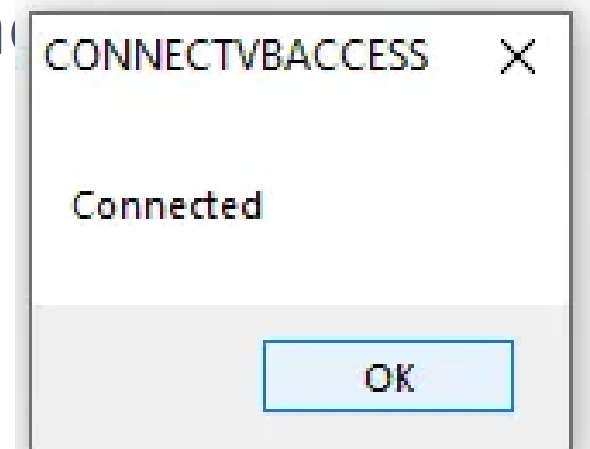
```
con.ConnectionString = c
    Try
        con.Open()
        If con.State = ConnectionState.Open Then
            MsgBox("connected")
        Else
            MsgBox("not connected")
        End If
    Catch ex As Exception
        MsgBox("ex.message")
    Finally
        con.Close()
    End Try
```

Test the Connection of Access database in VB.Net

- We use try-catch to the exceptions that may occur during runtime.
- open the connection
- check using if statement if the connection is open
- 'Display a message box if successfully connected or Not
- close the connection

Press "F5" to run the Project.

When you run the project it will give you this message



How to Load Record from Access Database to Datagridview In VB.Net

Try

```
Dim sql As String
Dim cmd As New OleDbCommand
Dim dt As New DataTable
Dim dp As New OleDbDataAdapter
con.Open()
sql = "Select * from tblitems"
cmd.Connection = con
cmd.CommandText = sql
dp.SelectCommand = cmd
dp.Fill(dt)
DataGridView1.DataSource = dt
Catch ex As Exception
    MsgBox("ex.message")
Finally
    con.Close()
End Try
```

After adding the code, you may press **F5** or click the Start debugging button to test the code. The output should look like as shown below.

List of Items

	ID	ITEMNAME	ITEMDESCF	QTY	PRICE
▶	1	MONITOR	COMPUTE...	5	60
	2	MOTHER...	COMPUTE...	10	200
	3	KEYBOARD	KEYBOAR...	10	150
	4	laptop	sony laptop	10	1500
•					

Item Details

Item Name

Description

Qty

Price

جامعة بغداد / كلية التربية للعلوم الصرفة ابن الهيثم / قسم علوم الحاسبات
المرحلة الثالثة صباحي / مسائي (نظري)
السنة 2021-2022
اساتذة المادة: شهلاء طالب

Visual Programming

Lecture 16 – How to Connect Access Database in VB.Net (Insert, Update and Delete)

- **What is Data provider?** The data provider is used to connecting to a database. Executing commands and retrieving data, storing it in a **Dataset**, reading the retrieved data and updating the database.
- **What is Command?** A command is a SQL statement or a stored procedure used to retrieve, insert, delete or modify data in a data source.
- **What is DataAdapter?** This is integral to the working of ADO.Net since data is transferred to and from a database through a data adapter. It retrieves data from a database into a dataset and updates the database. When changes are made to the dataset, the changes in the database are actually done by the data adapter.
- **What is DataTable?** Datatable consists of the DataRow and DataColumn objects. The DataTable objects are case-sensitive.
- **What is OleDbConnection?** **OleDbConnection** is designed for connecting to a wide range of databases, like Microsoft Access and Oracle.

List of Items

	ID	ITEMNAME	ITEMDESCF	QTY	PRICE
▶	1	MONITOR	COMPUTE...	5	60
	2	MOTHER...	COMPUTE...	10	200
	3	KEYBOARD	KEYBOAR...	10	150
	4	laptop	sony laptop	10	1500
•					

Item Details

Item Name

Description

Qty

Price

Save Record in Access Database using VB.net

We learn **how to save the record in the access database using vb.net**. To do this, double click the **“Save Item”** button and add the following code.

Implementation

Form1

Load Records

List of Items

ID	ITEMNAM	ITEMDESC	QTY	PRICE
1	MONITOR	COMPU...	5	60
2	MOTHE...	COMPU...	10	200
3	KEYBO...	KEYBO...	10	150
4	laptop	sony lapt...	10	1500

Item Details

Item Name

Description

QTY

Price

Save Item

Update Item

Delete Item

Form1

Load Records

List of Items

ID	ITEMNAM	ITEMDESC	QTY	PRICE
1	MONITOR	COMPU...	5	60
2	MOTHE...	COMPU...	10	200
3	KEYBO...	KEYBO...	10	150
4	laptop	sony lapt...	10	1500

Item Details

Item Name

Description

QTY

Price

Save Item

Update Item

Delete Item

Insert

New record has been inserted successfully!

OK

Implementation

Form1



Load Records

List of Items

ID	ITEMNAM	ITEMDESC	QTY	PRICE
1	MONITOR	COMPU...	5	60
2	MOTHE...	COMPU...	10	200
3	KEYBO...	KEYBO...	10	150
4	laptop	sony lapt	10	1500
12	Scanner	HP Scan...	4	400

Item Details

Item Name

Description

QTY

Price

Save Item

Update Item

Delete Item

Try

```
Dim i As Integer
Dim sql As String
Dim cmd As New OleDb.OleDbCommand
con.Open()
sql = "Insert Into
tblitems(ItemName,ITEMDESCRIPTION,Qty,Price)
VALUES ('" & TextBox1.Text & "', '" &
TextBox2.Text & "', '" & TextBox3.Text &
"', '" & TextBox4.Text & "')"
cmd.Connection = con
cmd.CommandText = sql
i = cmd.ExecuteNonQuery
If i > 0 Then
    MsgBox("New record has been
inserted successfully!")
    dt.Clear()
    sql = "Select * from tblitems"
```

```
cmd.Connection = con
cmd.CommandText = sql
dp.SelectCommand = cmd
dp.Fill(dt)
DataGridView1.DataSource = dt

Else
    MsgBox("No record has been
inserted successfully!")
End If

Catch ex As Exception
    MsgBox(ex.Message)
Finally
    con.Close()
End Try
```

Updating of Records from Access Database In VB.Net

we will learn **how to update records from an access database using vb.net**. In order for us to proceed in **updating the record**, we will add first a code to pass value from datagridview to textboxes.

To start with, go back to form design and double click the datagridview. And Change the Event to **“CellClick”** from **“CellContentClick”**. It means that every time the user clicks the selected data in the Datagrid view, the value will automatically pass to the textboxes. So here’s the following code.

Implementation

4

Load Records

List of Items

ID	ITEMNAM	ITEMDESC	QTY	PRICE
1	MONITOR	COMPU...	5	60
2	MOTHE...	COMPU...	10	200
3	KEYBO...	KEYBO...	10	150
4	laptop	sony lapt...	10	1500
12	Scanner	HP Scan...	4	400
*				

Item Details

Item Name

Description

QTY

Price

Save Item

Update Item

Delete Item

4

Load Records

List of Items

ID	ITEMNAM	ITEMDESC	QTY	PRICE
1	MONITOR	COMPU...	5	60
2	MOTHE...	COMPU...	10	200
3	KEYBO...	KEYBO...	10	150
4	laptop	sony lapt...	10	1500
12	Scanner	HP Scan...	4	400
*				

Item Details

Item Name

Description

QTY

Price

Save Item

Update Item

Delete Item

Insert

Record has been UPDATED successfully!

OK

Implementation

Load Records

List of Items

	ID	ITEMNAM	ITEMDESC	QTY	PRICE
▶	1	MONITOR	COMPU...	5	60
	2	MOTHE...	COMPU...	10	200
	3	KEYBO...	KEYBO...	10	150
	4	laptop	sony lapt...	100	1500
	12	Scanner	HP Scan...	4	400
*					

Item Details

Item Name

Description

QTY

Price

Save Item

Update Item

Delete Item

```
Private Sub DataGridView1_CellClick(sender As Object, e As
DataGridViewCellEventArgs) Handles DataGridView1.CellClick
    Me.Text = DataGridView1.CurrentRow.Cells(0).Value
    TextBox1.Text =
DataGridView1.CurrentRow.Cells(1).Value
    TextBox2.Text =
DataGridView1.CurrentRow.Cells(2).Value
    TextBox3.Text =
DataGridView1.CurrentRow.Cells(3).Value
    TextBox4.Text =
DataGridView1.CurrentRow.Cells(4).Value
End Sub
```

And here's the Following code for Updating the record from access database using vb.net.

Try

```
    Dim sql As String
    Dim cmd As New
OleDb.OleDbCommand
    con.Open()
    sql = "UPDATE tblitems SET
ItemName='" & TextBox1.Text & "',
ITEMDESCRIPTION='" & TextBox2.Text &
"', Qty='" & Val(TextBox3.Text) & "',
Price='" & Val(TextBox4.Text) & "'
WHERE Id='" & Val(Me.Text) & '"
    cmd.Connection = con
    cmd.CommandText = sql
    i = cmd.ExecuteNonQuery
    If i > 0 Then
        MsgBox("Record has been
UPDATED successfully!")
        dt.Clear()
    sql = "Select * from tblitems"
```

```
cmd.Connection = con
cmd.CommandText = sql
dp.SelectCommand = cmd
dp.Fill(dt)
```

```
DataGridView1.DataSource = dt
    Else
        MsgBox("No record has been
UPDATED!")
    End If
Catch ex As Exception
    MsgBox(ex.Message)
Finally
    con.Close()
End Try
```

Deleting of Records from Access Database In VB.Net

For deleting of records from the access database in vb.net, we will still use the same code in inserting and updating the record from access using vb.net.

Go back to Form design and double click the “*Delete Item*” button. Then add the following code.

Implementation

12

Load Records

List of Items

ID	ITEMNAM	ITEMDESC	QTY	PRICE
1	MONITOR	COMPU...	5	60
2	MOTHE...	COMPU...	10	200
3	KEYBO...	KEYBO...	10	150
4	laptop	sony lapt...	100	1500
▶ 12	Scanner	HP Scan...	4	400
*				

Item Details

Item Name

Description

QTY

Price

Save Item

Update Item

Delete Item

12

Load Records

List of Items

ID	ITEMNAM	ITEMDESC	QTY	PRICE
1	MONITOR	COMPU...	5	60
2	MOTHE...	COMPU...	10	200
3	KEYBO...	KEYBO...	10	150
4	laptop	sony lapt...	100	1500
▶ 12	Scanner	HP Scan...	4	400
*				

Item Details

Item Name

Description

QTY

Price

Save Item

Update Item

Delete Item

Insert

Record has been deleted successfully!

OK

Implementation

Load Records

List of Items

	ID	ITEMNAM	ITEMDESC	QTY	PRICE
▶	1	MONITOR	COMPU...	5	60
	2	MOTHE...	COMPU...	10	200
	3	KEYBO...	KEYBO...	10	150
	4	laptop	sony lapt...	100	1500
	*				

Item Details

Item Name

Description

QTY

Price

Save Item

Update Item

Delete Item

Try

```
    Dim sql As String
    Dim cmd As New
OleDb.OleDbCommand
    con.Open()
    sql = "Delete * from
tblitems WHERE Id=" & Val(Me.Text) &
""
    cmd.Connection = con
    cmd.CommandText = sql
    i = cmd.ExecuteNonQuery
    If i > 0 Then
        MsgBox("Record has
been deleted successfully!")
    dt.Clear()
    sql = "Select * from tblitems"
```

```
cmd.Connection = con
cmd.CommandText = sql
dp.SelectCommand = cmd
dp.Fill(dt)
```

```
DataGridView1.DataSource = dt
```

```
    Else
MsgBox("No record has been
deleted!")
    End If
Catch ex As Exception
    MsgBox(ex.Message)
Finally
    con.Close()

End Try
```


جامعة بغداد / كلية التربية للعلوم الصرفة ابن الهيثم / قسم علوم الحاسبات
المرحلة الثالثة صباحي / مسائي (نظري)
السنة 2021-2022
اساتذة المادة: شهلاء طالب

Visual Programming

Lecture 17– How to Create a Quick Search using VB.net and MS Access

Steps how to search record in vb.net using ms access

Step 1: First, use the database that you created before.

Step 2: Next, we will now create a *Visual basic Project* named “Bookfinder” then, *extract* the **download database** and put it inside the **Bin folder**.

Step 3: After creating a project in Visual Basic. Let’s now design the form. Follow the Image below on the form looks like.

Item Name:

Description:

QTY:

Price:

Search (OR)

Search (AND)

Use Quick Search

Use TextBox Filter

Result

Step 4: Next, Let's add **functionality** to our application by adding some code to our **objects**.

Step 5: First, we will add declaration under public class: and here's the code:

```
Imports System.Data.OleDb
Public Class Form1
    Dim i As Integer
    Dim con As New OleDbConnection
    Dim sql As String
    Dim cmd As New OleDbCommand
    Dim dt As New DataTable
    Dim dp As New OleDbDataAdapter
    Dim c As String = "Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=C:\Users\user\Documents\Database1.mdb"
```

Step 6: Next, Double click the form, and on the form, load adds the following code:

```
Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
    TextBox5.Visible = False
    Button3.Enabled = False
    Label5.Visible = False
End Sub
```

Step 7: Double the “Search OR” button and add the following code:

```
dt.Clear()  
    Try  
        con.ConnectionString = c  
        con.Open()  
        sql = "Select * from tblitems where ItemName='" & TextBox1.Text & "' or  
ITEMDESCRIPTION='" & TextBox2.Text & "' or QTY='" & TextBox3.Text & "' or  
Price='" & TextBox4.Text & "' "  
  
        cmd.Connection = con  
        cmd.CommandText = sql  
        dp.SelectCommand = cmd  
        dp.Fill(dt)  
        DataGridView1.DataSource = dt  
  
    Catch ex As Exception  
        MsgBox(ex.Message)  
    Finally  
        con.Close()  
    End Try
```

Implementation

Search Information

Item Name:

Description:

QTY:

Price:

Quick Search:

Result

	ID	ITEMNAME	ITEMDESCRIPTIO	QTY	PRICE
▶	13	scanner	HP	4	400
	14	laptop	hp	1	1000
*					

Step 8: Next double click the “**Use Quick Search**” button. And ad the following code:

```
Button2.Enabled = False
TextBox5.Visible = True
TextBox1.BackColor = Color.Aqua
TextBox2.BackColor = Color.Aqua
TextBox3.BackColor = Color.Aqua
TextBox4.BackColor = Color.Aqua
TextBox1.Enabled = False
TextBox2.Enabled = False
TextBox3.Enabled = False
TextBox4.Enabled = False
Button3.Enabled = True
Label15.Visible = True
```

The screenshot shows a Windows application window titled "Search Information". The window contains a search interface with the following elements:

- Four input fields labeled "Item Name:", "Description:", "QTY:", and "Price:". Each field has a red rectangular highlight.
- Two buttons labeled "Search (OR)" and "Search (AND)" positioned to the right of the input fields.
- Two buttons labeled "Use Quick Search" and "Use TextBox Filter" positioned below the search buttons.
- A "Quick Search:" label followed by a text input field.
- A large gray rectangular area labeled "Result" at the bottom of the window.

Step 9: Then, double click the **textbox** under “**Quick Search**” Label make sure you will be redirected to “**Texhchanged *Event***”.

This allows you to perform a quick search because every time you type on the textbox provided it will **automatically give** you the results on the **datagridview** based on the keyword inputted by the user.

And here's add the following code:

```
Private Sub TextBox5_TextChanged(sender As Object, e As EventArgs) Handles
TextBox5.TextChanged
    dt.Clear()
    Try
        con.ConnectionString = c
        con.Open()
        sql = "Select * from tblitems where ItemName LIKE '%" & TextBox5.Text &
            "%' or ITEMDESCRIPTION LIKE '%" & TextBox5.Text & "%' "
        cmd.Connection = con
        cmd.CommandText = sql
        dp.SelectCommand = cmd
        dp.Fill(dt)
        DataGridView1.DataSource = dt
    Catch ex As Exception
        MsgBox(ex.Message)
    Finally
        con.Close()
    End Try
End Sub
```

Implementation

Search Information

Item Name:

Description:

QTY:

Price:

Quick Search:

Search (OR) Search (AND)

Use Quick Search

Use TextBox Filter

Result

	ID	ITEMNAME	ITEMDESCRIPTIO	QTY	PRICE
▶	4	laptop	sony laptop	10	1500
	14	laptop	hp	1	1000
	15	laptop	lenovo	1	1200
*					

Step 10: Double the “**Search AND**” button and add the following code:
dt.Clear()

Try

```
con.ConnectionString = c
```

```
con.Open()
```

```
sql = "Select * from tblitems where ItemName='" & TextBox1.Text & "'  
and ITEMDESCRIPTION='" & TextBox2.Text & "' and QTY='" & TextBox3.Text  
& "' and Price='" & TextBox4.Text & "' "
```

```
cmd.Connection = con
```

```
cmd.CommandText = sql
```

```
dp.SelectCommand = cmd
```

```
dp.Fill(dt)
```

```
DataGridView1.DataSource = dt
```

Catch ex As Exception

```
MsgBox(ex.Message)
```

Finally

```
con.Close()
```

End Try

Implementation

Search Information

Item Name:

Description:

QTY:

Price:

Quick Search:

Result

ID	ITEMNAME	ITEMDESCRIPTIO	QTY	PRICE
14	laptop	hp	1	1000
*				

Step 11: Double the “**Use TextBox Filter**” button and add the following code:

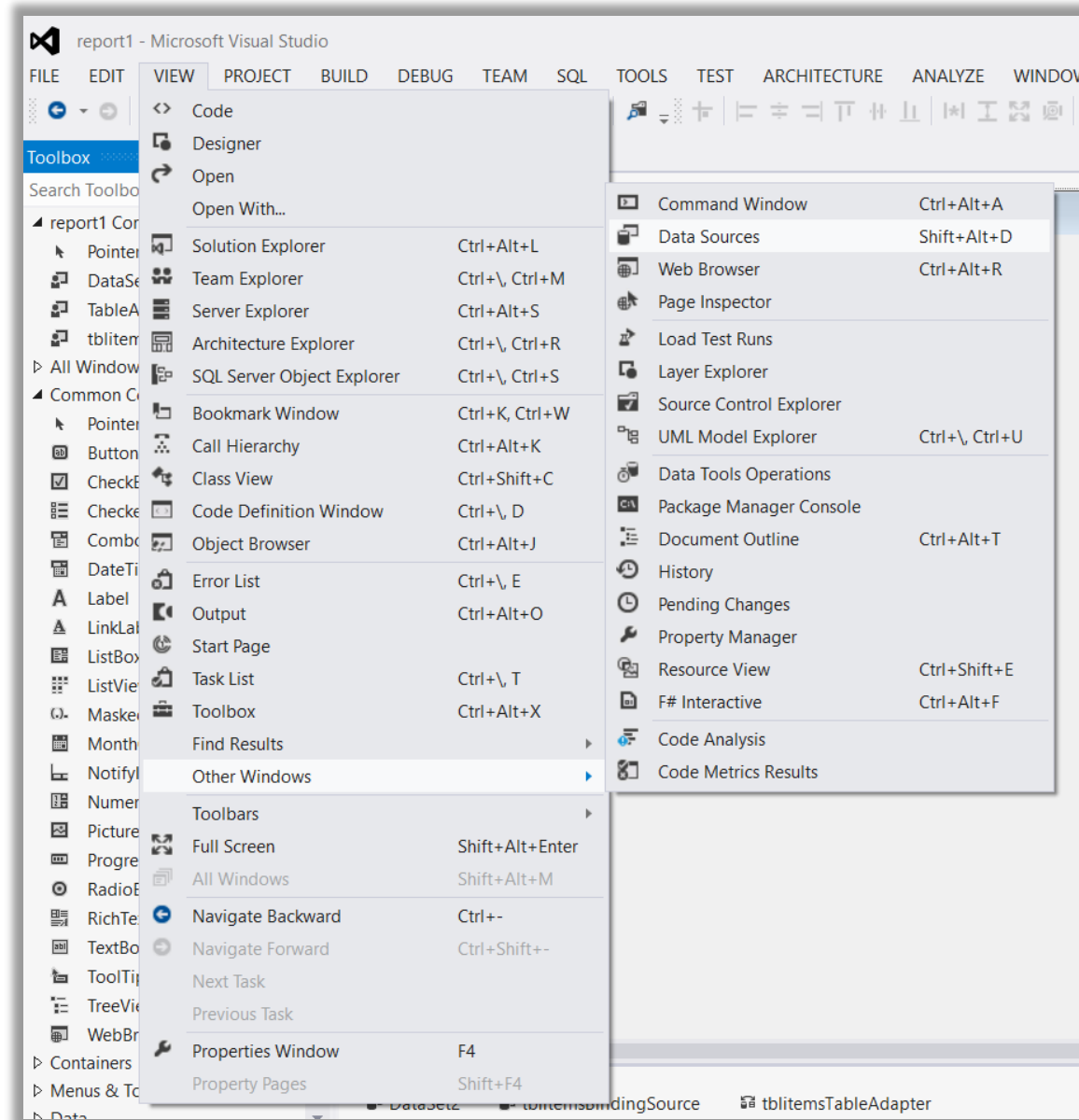
```
Button2.Enabled = True
TextBox1.BackColor = Color.White
TextBox2.BackColor = Color.White
TextBox3.BackColor = Color.White
TextBox4.BackColor = Color.White
TextBox1.Enabled = True
TextBox2.Enabled = True
TextBox3.Enabled = True
TextBox4.Enabled = True
Button3.Enabled = False
Label5.Visible = False
TextBox5.Visible = False
```

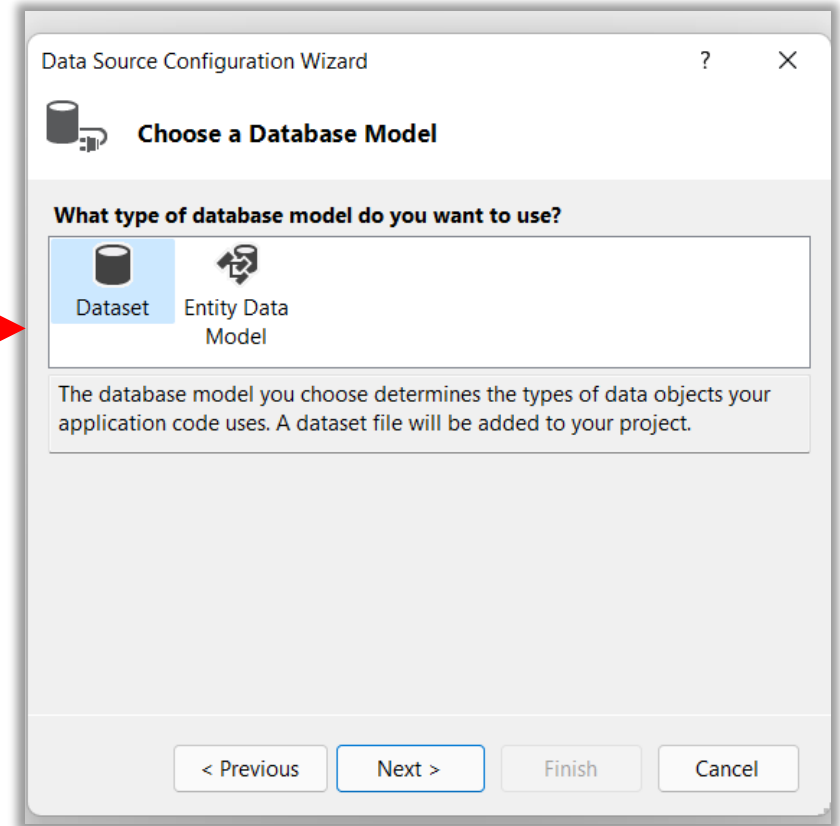
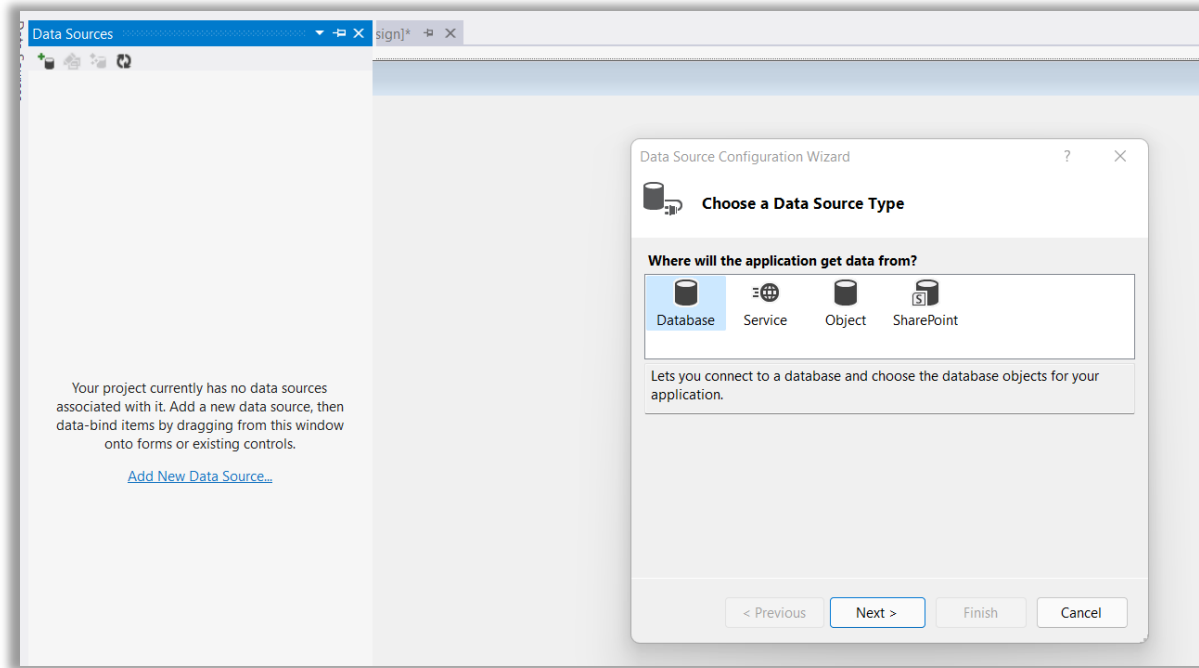
جامعة بغداد / كلية التربية للعلوم الصرفة ابن الهيثم / قسم علوم الحاسبات
المرحلة الثالثة صباحي / مسائي (نظري)
السنة 2021-2022
اساتذة المادة: شهلاء طالب

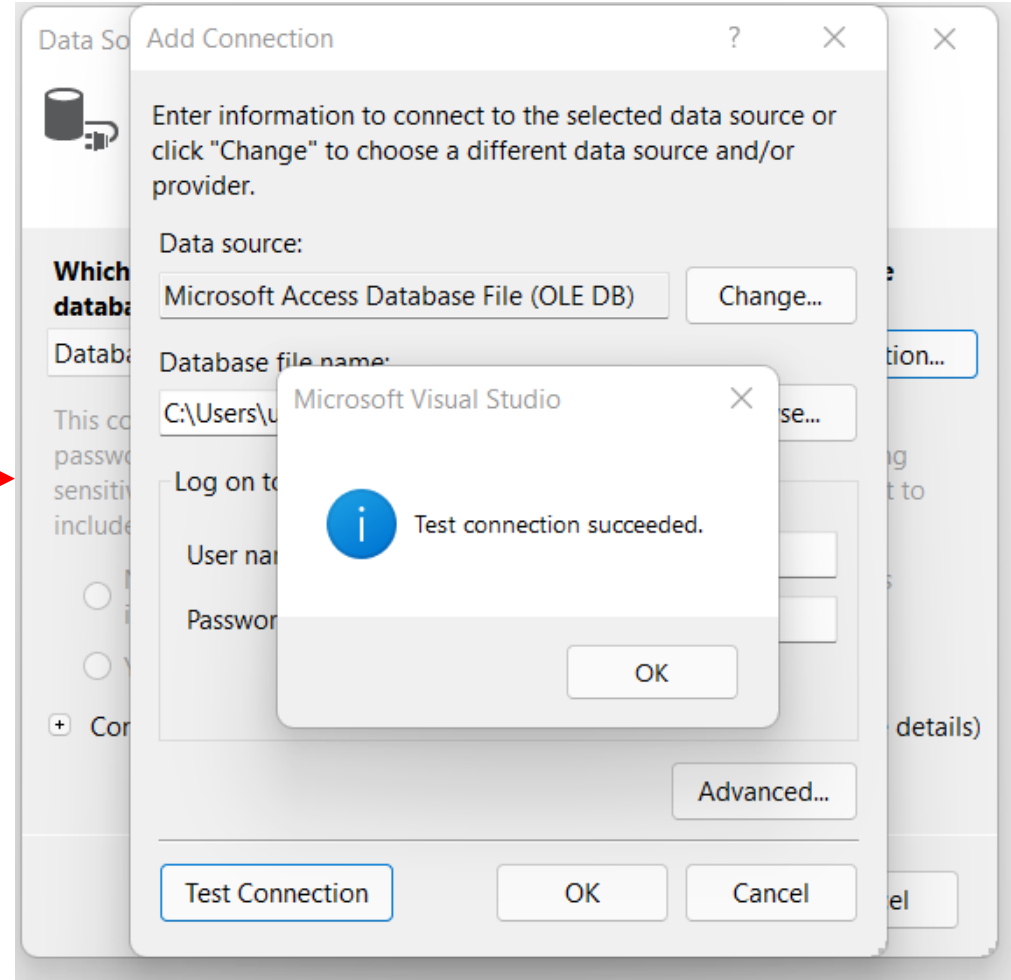
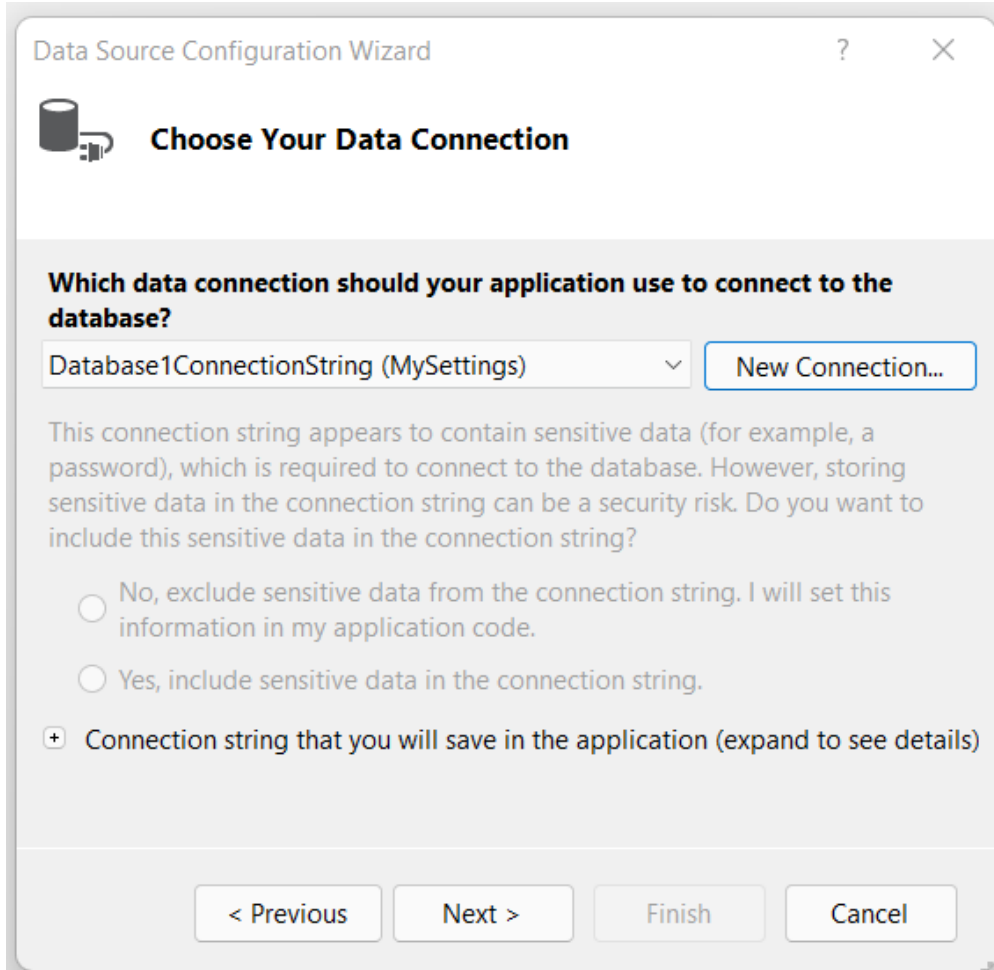
Visual Programming

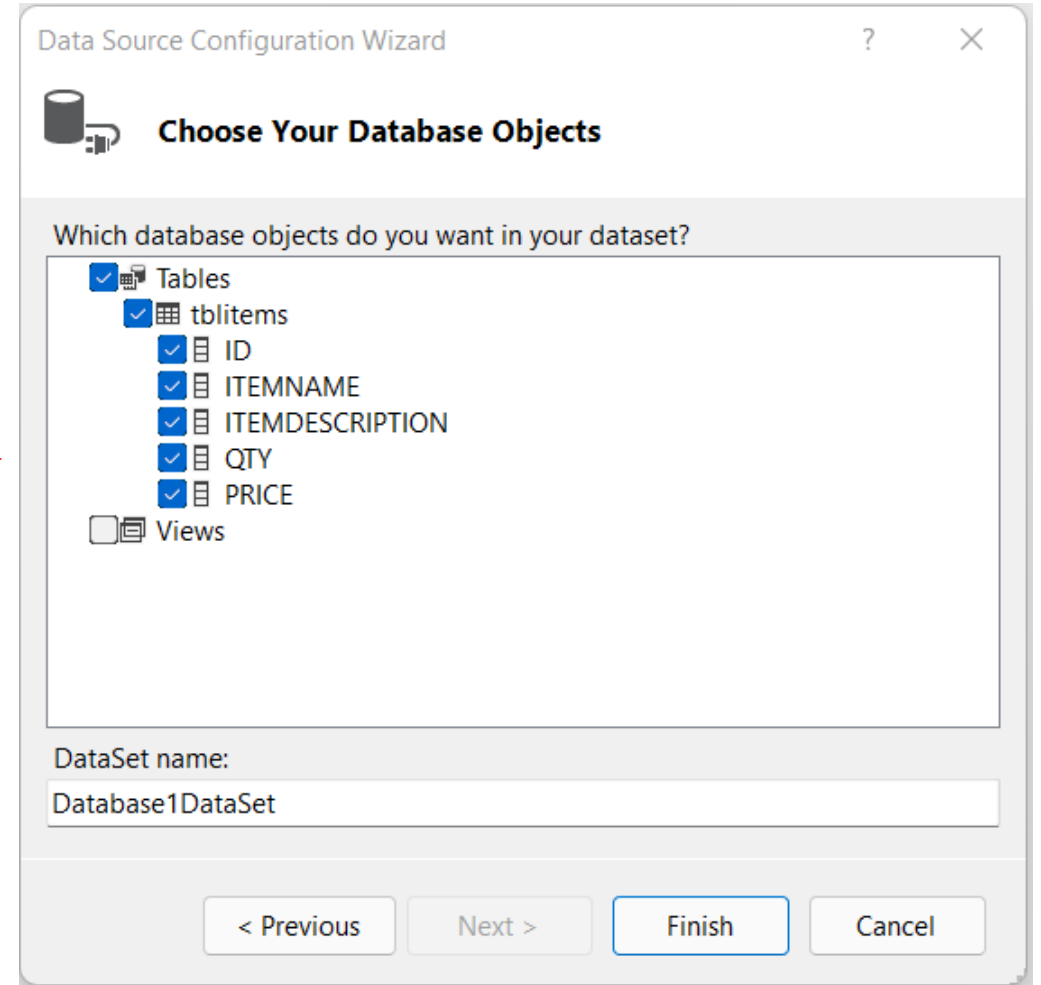
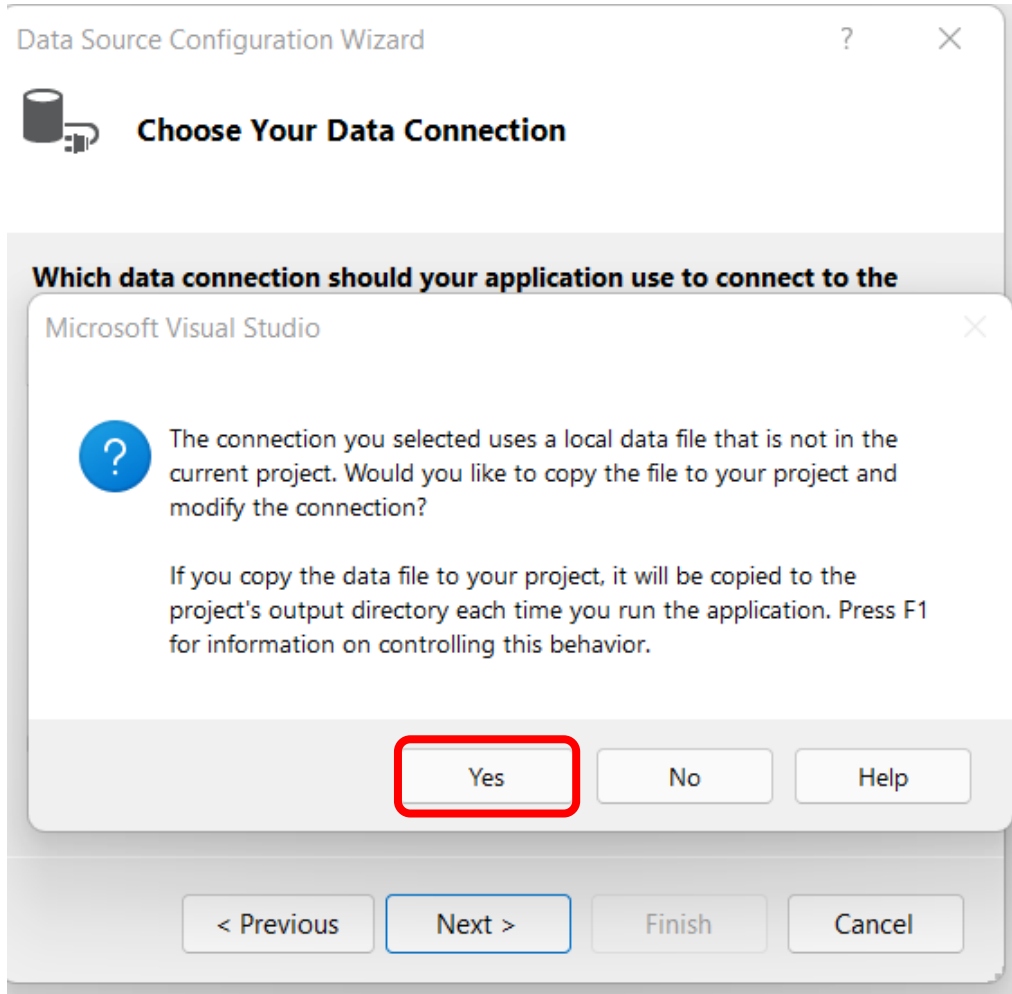
Lecture 18 – Report

Step 1: Data Sources









Data Sources

- Database1DataSet
 - tblitems
 - ID
 - ITEMNAME
 - ITEMDESCRIPTION
 - QTY
 - PRICE

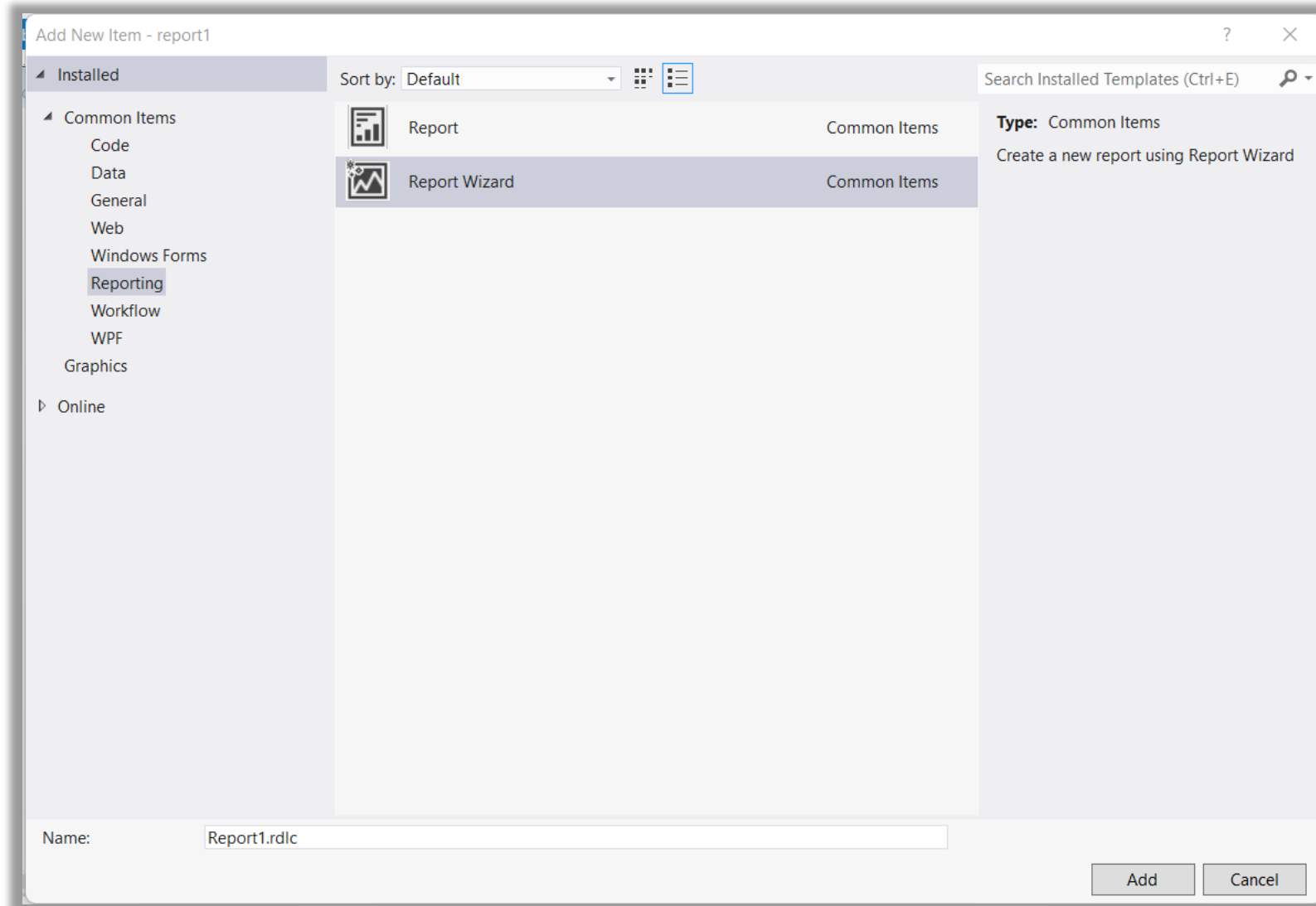


Solution Explorer

Search Solution Explorer (Ctrl+;)

- Solution 'report1' (1 project)
 - report1
 - My Project
 - app.config
 - Database1.mdb
 - Database1DataSet.xsd
 - Form1.vb

Step 2: Using Report Wizard



Report Wizard

Arrange fields

Arrange fields to group data in rows, columns, or both, and choose values to display. Data expands across the page in column groups and down the page in row groups. Use functions such as Sum, Avg, and Count on the fields in the Values box.

Available fields

- ID
- ITEMNAME
- ITEMDESCRIPTION
- QTY
- PRICE

Column groups

Row groups

- ITEMNAME

Σ Values

- PRICE

Help < Back Next > Cancel



Report Wizard

Choose the layout

If you choose to show subtotals and grand totals, you can place them above or below the group. Stepped reports show hierarchical structure with indented groups in the same column.

Options:

- Show subtotals and grand totals
 - Blocked, subtotal below
 - Blocked, subtotal above
 - Stepped, subtotal above
- Expand/collapse groups

Preview

ITEMNAME	PRICE
[ITEMNAME]	[PRICE]
Total	

Help < Back Next > Cancel

Choose a style

Styles feature different fonts and color schemes, but do not affect the basic layout. You can customize the style after you finish the wizard.

Styles:

Corporate
Forest
Generic
Mahogany
Ocean
Slate

Preview

ITEMNAME	PRICE
[ITEMNAME]	[PRICE]
Total	

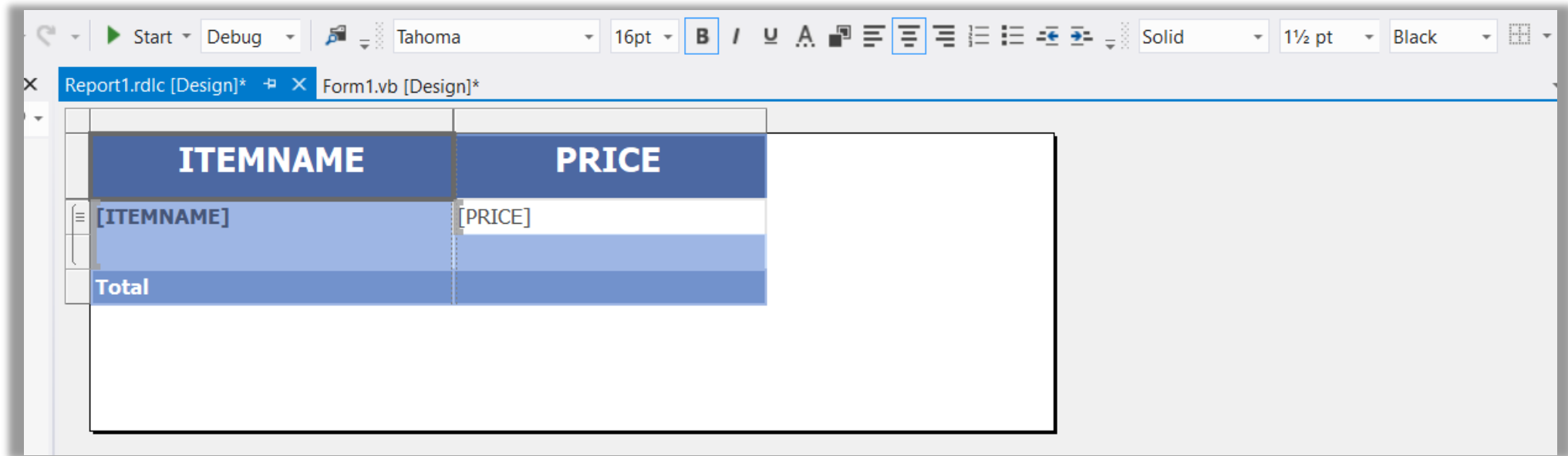
Help

< Back

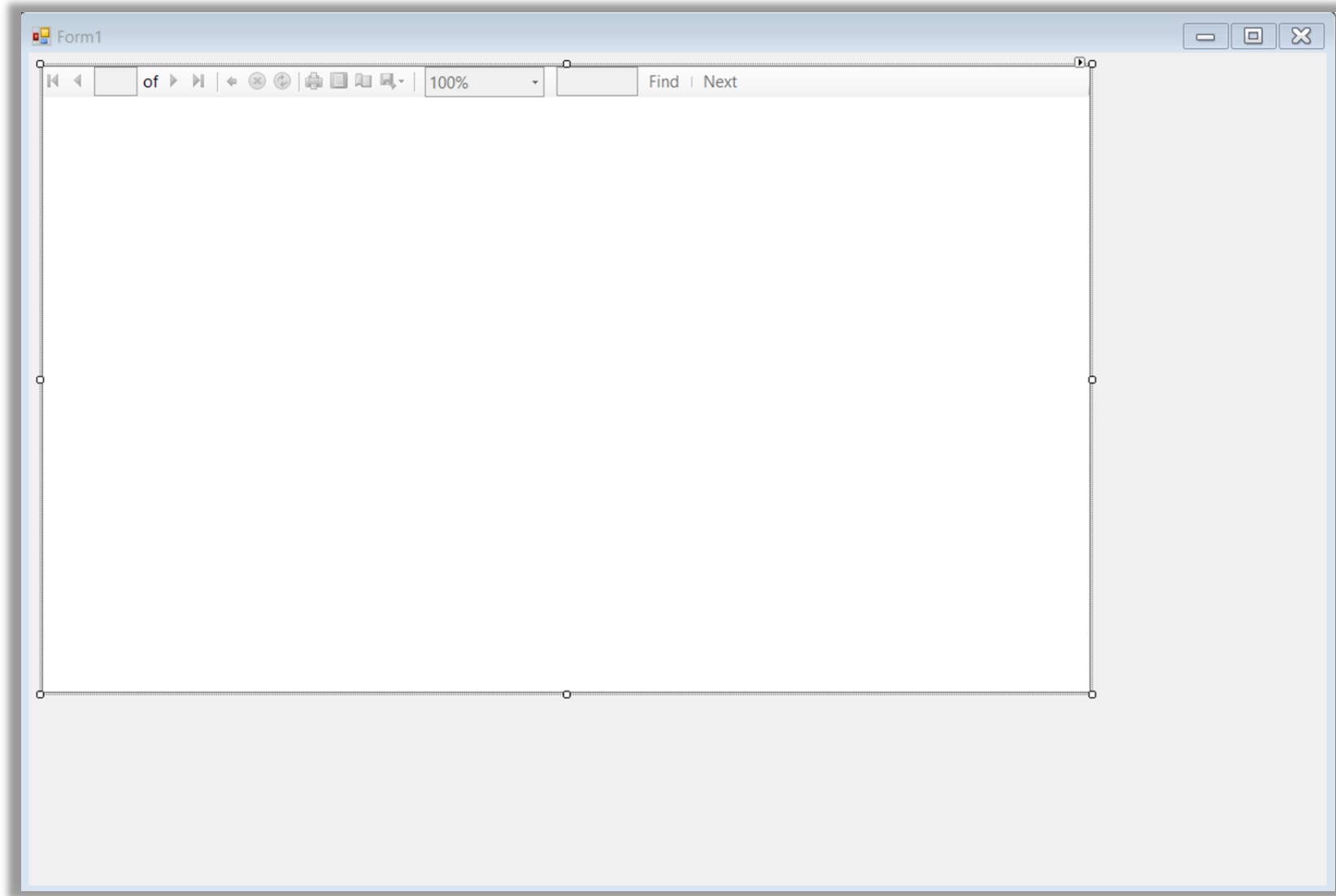
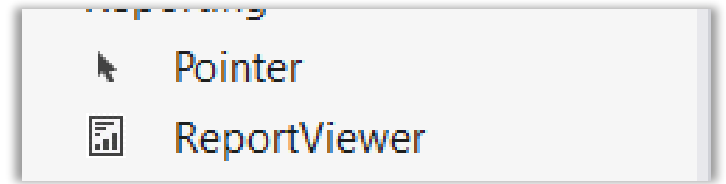
Finish >>

Cancel

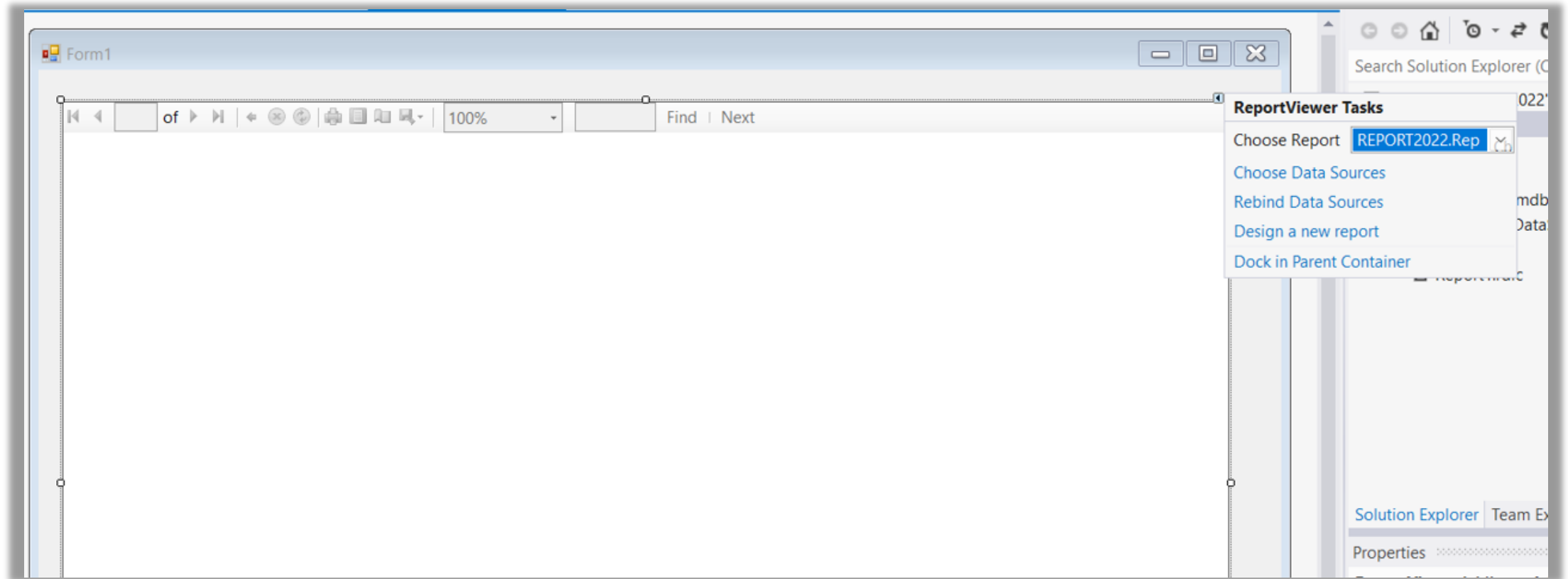
Step 3: Format the Report



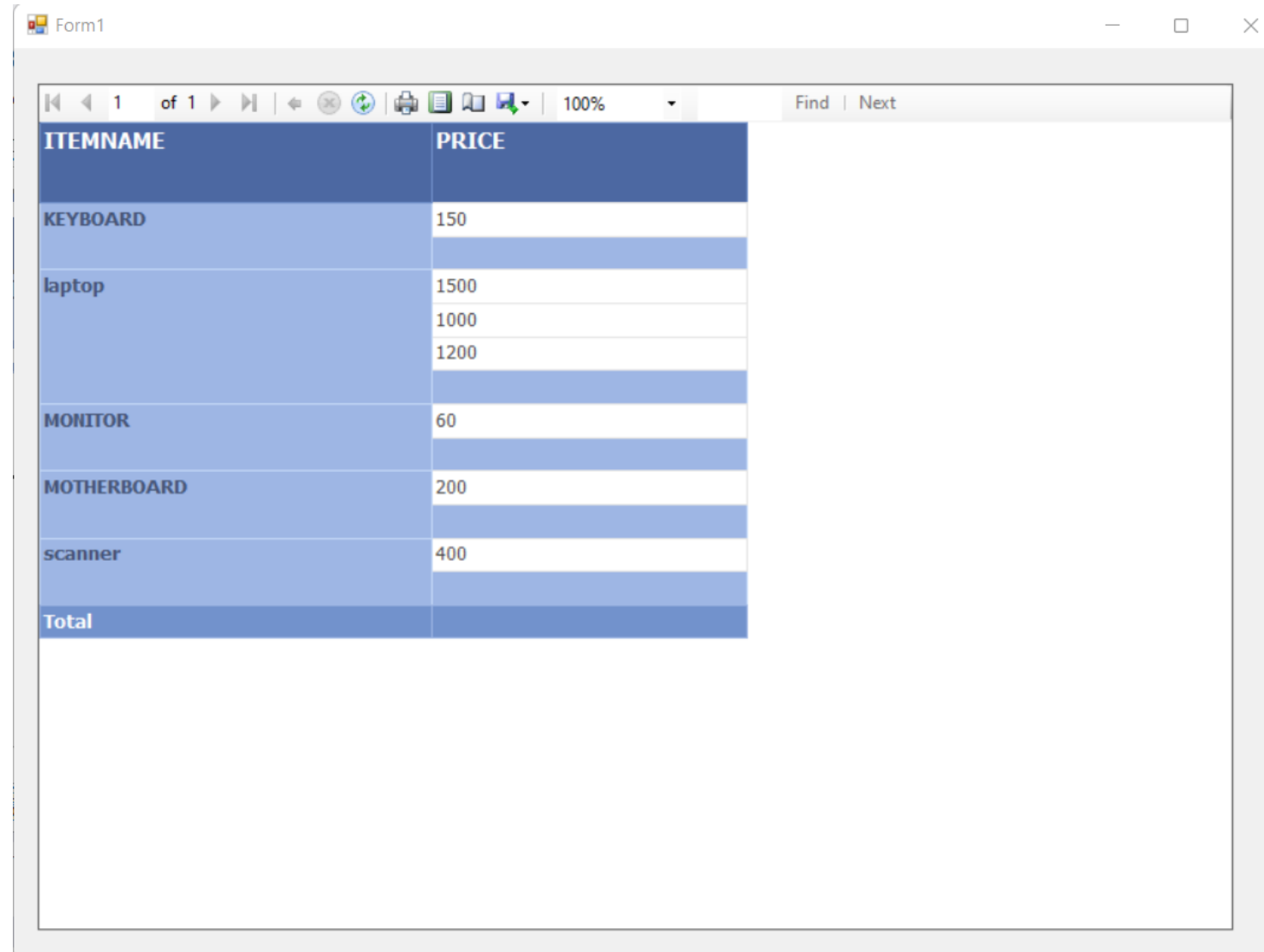
Step 4: Adding ReportViewer



Step 5: Connect the ReportViewer with Report



Step 6: Review the Report



The screenshot shows a report viewer window titled "Form1". The report displays a table with two columns: "ITEMNAME" and "PRICE". The table contains the following data:

ITEMNAME	PRICE
KEYBOARD	150
laptop	1500
	1000
	1200
MONITOR	60
MOTHERBOARD	200
scanner	400
Total	

ITEM NAME	PRICE
KEYBOARD	150
Laptop	1500
	1000
	1200
MONITOR	50
MOTHERBOARD	200
Scanner	400
Total	

جامعة بغداد / كلية التربية للعلوم الصرفة ابن الهيثم / قسم علوم الحاسبات
المرحلة الثالثة صباحي / مسائي (نظري)
السنة 2021-2022
اساتذة المادة: شهلاء طالب

Visual Basic 2012

Lecture 19 – How to navigate between Records using VB.net and MS Access

```
Public Sub nav(ByVal a As Integer)
    Try

        TextBox1.Text = dt.Rows(a).Item(1)
        TextBox2.Text = dt.Rows(a).Item(2)
        TextBox3.Text = dt.Rows(a).Item(3)
        TextBox4.Text = dt.Rows(a).Item(4)

    Catch ex As Exception
        MsgBox(ex.Message)
    End Try

End Sub
```

Show First Record

```
Private Sub Button7_Click(sender As Object, e As
EventArgs) Handles Button7.Click
    If minval <> 0 Then
        minval = 0
        nav(minval)
    End If
End Sub
```

Show Last Record

```
Private Sub Button8_Click(sender As Object, e As EventArgs)
Handles Button8.Click
    minval = dt.Rows.Count() - 1
    nav(minval)
End Sub
```

Show Next Record

```
Private Sub Button5_Click(sender As Object, e As EventArgs)
Handles Button5.Click

    minval += 1
    If minval > dt.Rows.Count() - 1 Then

        minval = dt.Rows.Count() - 1

    End If
    nav(minval)
End Sub
```


Show Previous Record

```
Private Sub Button6_Click(sender As Object, e As  
EventArgs) Handles Button6.Click
```

```
    minval -= 1  
    If minval < 0 Then  
        minval = 0  
    End If  
    nav(minval)  
End Sub
```

جامعة بغداد / كلية التربية للعلوم الصرفة ابن الهيثم / قسم علوم الحاسبات
المرحلة الثالثة صباحي / مسائي (نظري)
السنة 2021-2022
اساتذة المادة: شهلاء طالب

Visual Programming

Lecture 20 – File Handling + Function & Procedure

Read Text from Text File

```
Imports System.IO
Public Class Form1

Private Sub Button4_Click(sender As Object, e As
EventArgs) Handles Button4.Click
    Dim fileReader As String

    fileReader=My.Computer.FileSystem.ReadAllText("C:
\Users\user\Desktop\mshraa\g12\24003.txt")
    MsgBox(fileReader)
    TextBox1.Text = fileReader

End Sub
```

Write Text in Text File

```
Dim FILE_NAME As String = "C:\Users\user\Desktop\mshraa\g12\24001.txt"

If System.IO.File.Exists(FILE_NAME) = True Then

    Dim objWriter As New System.IO.StreamWriter(FILE_NAME)

    objWriter.Write(TextBox1.Text)
    objWriter.Close()
    MessageBox.Show("Text written to file")

Else

    MessageBox.Show("File Does Not Exist")

End If
```

What is Procedure in VB

A procedure is a group of statements that together perform a task when called. After the procedure is executed, the control returns to the statement calling the procedure. VB.Net has two types of procedures –

- Functions
- Sub procedures or Subs

Functions return a value, whereas Subs do not return a value.

Defining a Function

The Function statement is used to declare the name, parameter and the body of a function. The syntax for the Function statement is –

–

```
Function  FunctionName  [(ParameterList)]  As  ReturnType  
[Statements]  
End Function
```

Where,

- ***FunctionName*** – indicates the name of the function
- ***ParameterList*** – specifies the list of the parameters
- ***ReturnType*** – specifies the data type of the variable the function returns

Example

Following code snippet shows a function FindMax that takes two integer values and returns the larger of the two.

```
Sub Main()  
    Dim a As Integer = 100  
    Dim b As Integer = 200  
    Dim res As Integer  
    res = FindMax(a, b)  
    Console.WriteLine("Max value is : {0}", res)  
    Console.ReadLine()  
End Sub
```

```
Function FindMax(ByVal num1 As Integer, ByVal num2 As Integer)
As Integer
```

```
    ' local variable declaration */
```

```
    Dim result As Integer
```

```
    If (num1 > num2) Then
```

```
        result = num1
```

```
    Else
```

```
        result = num2
```

```
    End If
```

```
    FindMax = result
```

```
End Function
```


Defining Sub Procedures

The **Sub** statement is used to declare the name, parameter and the body of a sub procedure. The syntax for the Sub statement is –

```
Sub SubName [(ParameterList)] [Statements]
```

```
End Sub
```

Where,

- ***SubName*** – indicates the name of the Sub
- ***ParameterList*** – specifies the list of the parameters

Example

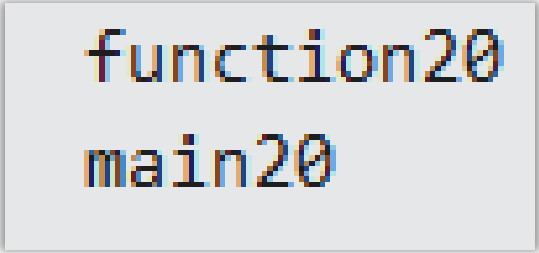
The following example demonstrates a Sub procedure CalculatePay that takes two parameters hours and wages and displays the total pay of an employee –

```
Sub Main()  
    'calling the CalculatePay Sub Procedure  
    CalculatePay(25, 10)  
    CalculatePay(40, 20)  
    CalculatePay(30, 27.5)  
    Console.ReadLine()  
End Sub  
Sub CalculatePay(ByRef hours As Double, ByRef wage As Decimal)  
    'local variable declaration  
    Dim pay As Double  
    pay = hours * wage  
    Console.WriteLine("Total Pay: {0:C}", pay)  
End Sub
```

ByRef Argument

```
Private Sub Button3_Click(sender As Object, e As EventArgs)
    Handles Button3.Click
    Dim a As Integer
    a = 10
    x(a)
    Console.WriteLine("main" & a)
End Sub

Sub x(ByRef a As Integer)
    a = 20
    Console.WriteLine("function" & a)
End Sub
```



function20
main20

ByVal Argument

```
Private Sub Button4_Click(sender As Object, e As EventArgs)  
Handles Button4.Click
```

```
    Dim a As Integer
```

```
    a = 10
```

```
    x1(a)
```

```
    Console.WriteLine("main" & a)
```

```
End Sub
```

```
Sub x1(ByVal a As Integer)
```

```
    a = 20
```

```
    Console.WriteLine("function" & a)
```

```
End Sub
```

```
function20  
main10
```