VB.NET

* VB.NET :—

VB.NET is a programming language developed by microsoft. It is an object oriented programming language. VB.NET is an extension of visual Basic 6.0. VB 6.0 lacks the object oriented feature.
VB.NET is one of the language available under .NET framework. .NET framework is a software development platform which supports multi languages such as VB.NET, C#.NET, ASP.NET, Visual Java, NET etc.

VB.NET allows to develop following various types of application :—

i) Console Application
ii) GUI - Application
iii) Web- Application

VB.NET provides visual programming environment where all the components required to design GUI are available as ready to use component. A programmer need not write any code to

develop and create GUI component. Hence, the software is developed very easily quickly due to this reason VB.NET is call RAD (Rapid Application Development) tool of microsoft.

* Visual Studio :-

It is an IDE (Integrated development Environment) of .NET framework. Any application under .NET framework is developed through visual studio. The latest version of visual studio is microsoft visual studio 2010.

✓ How to start visual Studio

All program → Microsoft visual studio 2010 → Microsoft visual Studio 2010.

✓ Structure of V.B.NET application

```
Module <name>
  Sub Main()
  ____
  ____
  End Sub
End Module.
```

* Variable

Memory location that holds data during the program execution is called Variable.

Variable Declaration in VB.NET

Syntax

Dim <Variable_Name> As <data_type>

ex:- Dim a As Integer

Data type	Memory	Range of value
Byte	1 byte	0 to 255
SByte	1 byte	-128 to +127
Short	2 bytes	-32,768 to +32767
Ushort	2 bytes	0 to 65535
Integer	4 bytes	-2,147,483,648 to +2,147,483,647
UInteger	4 bytes	0 to +4,294,967,295
Char	2 bytes	0 to 65535
Long	8 bytes	-9,223,372,036,854,775,808 to +9,223,372,036,854,775,807
ULong	8 bytes	0 to 18,446,744,073,709,551,615
Date	8 bytes	Date value 1-1-0001 to 31-12-9999
Single	4 bytes	-3.4e+38 to +3.4+38 (fractional no)
Double	8 bytes	-1.7e+308 to +1.7e+308 (fractional no)

Data type	Memory	Range of value
Decimal	16 bytes	upto 28 decimal places after decimal point.
Boolean	1 bit	True / false
Object	4 (or) 8 bytes	Any value
String	4 (or) 8 bytes	upto 2-billion characters

Operators in VB.NET

1. Arithmetic Operators
   - `+` – Addition
   - `-` – Subtraction
   - `*` – Multiplication
   - `/` – Fractional division
   - `\` – Integer division
   - `^` – Power
   - `MOD` – Remainder

exp	15+2	15-2	15x2	15/2	15\2	15^2	15 MOD 2
Ans	17	13	30	7.5	7	225	1

2) Relational Operators

- > → greater than
- < → Less than
- >= → greater than equal to
- <= → Less than equal to
- = → Equal to
- <> → Not equal to

3) Assignment Operator

- = → Set value
- += → Add & Assign
- -= → Subtract & Assign
- /= → Divide & Assign quotient
- \= → Divide & Assign quotient
- ^= → Power & Assign
- *= → Multiply & Assign.

4) Logical Operators

- AND — Logical And
- OR — Logical OR
- NOT — Logical NOT
- XOR — Exclusive OR

T	T	F
F	T	T
T	F	T
F	F	F

* I/O - Statement

i) System.Console.WriteLine():-

Display output on console. Breck line.

```
System.Console.WriteLine("Hello");
System.Console.WriteLine("Good Morning");
```

Output

```
Hello
Good Morning
```

ii) System.Console.Write():- Does not breck line.

```
System.Conole.Write("Hello");
System.Console.Write("Good Morning");
```

Output

Hello. Good Morning

iii) System.Console.ReadLine():- Take input from user.

Q. Write VB.NET Console program to add 2 nos input by user.

Sol^n

```
Module M1
Sub Main()
Dim a As Integer
Dim b As Integer
Dim c As Integer
System.Console.Write("Input 1st no-")
a= System.console.ReadLine()
System.Console.Write("Input 2nd no-")
```

```
b= System.Console.ReadLine()
c= a+b
System.Conside.Writeline("Addition = {0}", c)
System.Console.Readline()
End Sub
End Module.
```

Q Write VB.Net Console application to calculate area and circumference of a circle for given radius.

```
Module Module1
Sub Main()
Dim r, a, c As Single
System.Console.Write("Enter Radius in cm:")
r = System.Console.Readline()
a = 3.14 * r * r
c = 2 * 3.14 * r
System.Console.Writeline("Area = {0} sq cm", a)
System.Console.Writeline("Circumference = {0} cm", c)
System.Console.Readline()
End Sub
End Module
```

* Decision Control Statement in VB.NET

1. If --- End If

Syntax

```
If <condition> then
```

< Statement - block >

End If

2. If — Else — End if

Syntax

If <condition> Then

Else

End If

3. If — Else If — Else — Endif

Syntax

If <condition> then

Else If <condition> then

Else

End If

4. Select — End Select

Syntax

```
Select Case <exp>
  Case <value 1>
    ==
  Case <value 2>
    ==
  Case <value n>
    ==
  Case else
    ==
End Select.
```

Q Write VB.Net Conside program to check a number is odd and even.

```
Module Module 1
    Sub Main()
  Dim a As Integer
  System.Console.Write("Enter a Whole no:")
a= System.Console.Readline()
  If a MOD 2 = 0 Then
    System.Console.Writeline("Even no")
Else
  System.Console.Writeline("Odd no.")
End If
System.Console.Readline()
  End Sub
  End Module
```

Q. Write VB.NET Console program that accept a year from user and checks the year is leap year or not.

```
Module Module1
    Sub Main()
        Dim a As Integer
        System.Console.Write("Enter year:")
        a = system.Console.Readline()
        If (a Mod 4 = 0 And a MOD 100 <> 0) OR
        a Mod 400 = 0 then
        System.Conside.Writeline("Leap year")
        Else
            System.Console.Writeline("Not a leap year")
        End If
        System.Console.Readline()
    End Sub
End Module
```

Q Write VB.NET Program that accept a value in range 1 to 7 and displays day of week.

```
Imports System.Console
Module Module1
    Sub Main()
        Dim a As Integer
        Write("Enter a value (1-7):")
        a = Readline()
```

```
Select Case a
    Case 1
        Write ("Sunday")
    Case 2
        Write ("Monday")
    Case 3
        Write ("Tuesday")
    Case 4
        Write ("Wednesday")
    Case 5
        Write ("Thursday")
    Case 6
        Write ("Friday")
    Case 7
        Write ("Saturday")
    Case else
        Write ("Invalid Value")
    End Select
System.Console.Readline()
    End Sub
    End Module
```

* Loop - Statement in VB.Net :—

While :—

Syntax

Loop Terminating Statement

Exit while :- Terminates while --- End ~~while~~ while.

Exit Do :- Terminates Do — loop.

Exit for :- Terminates for --- next.

Q. Write VB.Net Console program to print all odd nos. in bet 1-15.

All odd nos. between 1-15
1 3 5 7 9 11 13 15

```
Imports system.Console
Module Module1
'using while --- End while
Sub Main()
```

```
Dim i As integer
Console.WriteLine("All odd nos. in bet. 1-15")
    i=1
While i<=15
    Console.Write(i & " ")
    i+=2
End While
Console.ReadLine()
End Sub
End Module
```

✓ 'Using Do While ... End do while Loop

```
Sub Main()
Dim i As integer
Console.WriteLine("All odd nos. in bet. 1-15")
    i=1
Do While i<=15
    Console.Write(i & " ")
    i+=2
Loop
Console.ReadLine()
End Sub
End Module
```

✓ 'Using Do until ... Loop

```
Sub Main()
Dim i As integer
Console.WriteLine("All odd nos. in bet 1-15")
    i=1
```

```
Do Until j>15
    Write (j & " ")
    j+=2
Loop
Readline()
End Sub
End Module
```

✓ 'Using for ... Next

```
Sub Main()
Dim i As Integer
Writeline ("All odd nos. in bet. 1-15")
for i=1 to 15 step 2
    Write(i & " ")
    Next i
    Readline()
End sub
    End Module
```

Q Write a VB.Net program to print table of a number.

```
Imports System.Console
    Module Module1
'Table of a number
    Sub Main()
Dim i, n As Integer
Write ("Enter a number:")
    n = Readline()
Writeline ("Table of " & n)
```

```
for(
    for j = 1 to 10
        Writeline (n * j)
Next
Readline()
    End sub
        End Module
```

Q. Print all leap years between 1900 to 2019.

```
Imports System.Console
Module Module1
    Sub Main()
Dim j As Integer
Writeline("All leap years bet. 1900-2019")
    for j = 1900 to 2019
If (j Mod 4 = 0 And j Mod 100 <> 0) or j Mod
    400 = 0 then
        Writeline (j)
    End if
Next
    Readline()
End Sub
    End Module
```

Q. Write VB.Net program to print factorial of a number.

```
Imports system.Console
Module Module1
    Sub Main()
Dim n As integer
Dim fact As long = 1
Write(" Enter a number:")
n = Readline()
While n > 1
fact = fact * n
n = n-1
End While
Writeline(" Factorial = " & fact)
Readline()
End Sub
End Module
```

Q. Write VB.Net program to print sum of digits in n number.

```
Imports system.Console
Module Module1
    Sub Main()
Dim n As long
Dim r, sum As short
system Write(" Enter a whole no.")
n = Readline()
While n > 0
r = n mod 10
sum = sum + r
n = n \ 10
```

```
End While
Writeline ("Sum of digit =" & sum)
Readline()
End Sub
End Module
```

Q Write VB.Net program to test prime no.

```
Imports System.Console
Module Module1
    Sub Main()
Dim n, i, p As Integer
Write("Enter a whole no.:")
n = Readline()
For i=2 to n\2
    If n Mod i=0 then
        p=1
    Exit for
    End if
    Next
    If p=0 then
        Writeline ("Prime no.")
    Else
        Writeline ("Not a prime no.")
    End if
    Readline()
    End Sub
    End Module
```

Q. Print Square root of no. for 1 to 10

Number Square Root
1
2
=
10

```
Module Module1
    Sub Main()
        Dim i As Integer
        System.Console.WriteLine("Number  Sq. root")
        for i = 1 to 10
        System.Console.ReadLine() WriteLine("Log {0}: {1}", i,
                Math.Sqrt(i))
        Next
        System.Console.ReadLine()
    End Sub
end Module
```

* Mathematical function

abs(n)	Pow()
Min(x,y)	Log() - Natural
Max(x,y)	Log10()
Sqrt(x,y)	Round()
Sin()	PI()
Cos()	Floor()
Tan()	Ceiling()

Page No. 60
Date 16-11-19

Q. Print

```
*
* *
* * *
* * * *
* * * * *
```

```
Module Module1
    Sub Main()
        Dim i, j As Integer
        for j=1 To 5
            for j=1 To 5
                System.Console.Write("* ")
            Next j
            System.Console.WriteLine()
        Next j
        System.Console.Readline()
    End Sub
End Module
```

* Print

```
A
A B
A B C
A B C D
A B C D E
```

```
Module Module1
    Sub Main()
        Dim i, j As Integer
```

```
for i=65 to 69
    for j=65 to i
System.Console.Write(Chr(j) & " ")
    Next j
System.Console.WriteLine()
    Next i
System.Console.ReadLine()
    End Sub
End Module
```

ARRAY

Collection of variables identified by common name.

Declaration Syntax

Dim <array-Name> (size) [As <data-type>]

Eg:-

Dim x(10) As Integer

x(0)
x(1)
x(2)
=
x(10)

Dim y(10)

y(0)
—
=
y(10)

} variable type

Q. Write VB.Net Console program that accept 10 - Whole Nos. from user & prints the nos. with sum, total odd & even.

```
Imports System.Console
Module Module1
    Sub Main()
Dim x(9) As Integer
Dim i, sum, odd, even As Integer
Writeline ("Input 10 Nos.")
    for j=0 To 9
Write ("Number {0}:", i+1)
  x(i)= Readline ()
Next
    Writeline ("List of nos")
    for i=0 To 9
      Writeline (x(i) & vbTab)
If x(i) Mod 2 =0 then
    even = even+1
    Else
      Odd = odd+1
End if
    sum = sum + x(i)
Next
Writeline ()
Writeline ("Sum of all nos = " & sum)
Writeline ("Total odd nos = " & odd)
Writeline ("Total even nos. =' & even)
    Readline()
```

End Sub
End Module

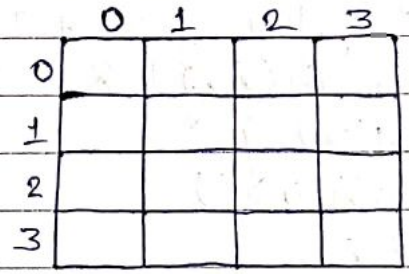
Initializing Array

Dim x() As Integer = {5,10,15,20,25}

Two Dimensional Array

Syntax
Dim <array Name (rows, cols) As <data type>

ex:- Dim A(3,3) As Integer



Initialize

Dim A(,) As Integer = {{1,3,5}, {2,3,6}, {4,5,7}}

Q Write VB.Net Console program that accept values for a 3×3 matrix & finally prints the matrix.

```
Imports System.Console
Module Module1
    Sub Main()
Dim m(2,2) As Integer
Dim i, j As Integer
Writeline("Input values for a 3x3 matrix")
    for i=0 to 2
        for j=0 to 2
        Write("Input for "&i+1&" row "&j+1&" col:"
            i+1, j+1)
            m(i,j) = Readline()
        Next j
    Next i
    Writeline("3x3 Matrix.")
        for i=0 to 2
            for j=0 to 2
    Write(m(i,j) & " ")
        Next j
        Writeline()
    Next i
        Readline()
        End Sub
        End Module
```

FUNCTION & PROCEDURE

Function: It contains set of instruction which get execute when function is called to complete a task. It always returns value to caller.

Defining function

function <function-Name> (Arg list) As <return type>

==

Return <value>
End function

Procedure: — It contains set of instruction which get executed when function is called to complete a small task. It does not return any value to caller.

Defining Procedure

Sub <procedure-Name> (Arg-list)

==

End sub

Q. Transpose 4x4 Matrix.

```
Imports System.Console
Module Module1
    Sub Main()
Dim a(4,4), m(4,4) As Integer
Dim i,j As Integer
Writeline("Input value of 4x4 Matrix")
    for i=0 to 3
        for j=0 to 3
Write("Input for row "&i+1&" col:" i+1,
                j+1)
        a(4,4)
        a(i,j) = Readline()
    Next j
    Next i
    Readline()
Writeline("3x3 Matrix")
for (i=0 to 3
    for j=0 to 3

    Writeline("
        Write(a(i,j) & " ")
        Readline()
Write("Transpose Matrix")
    for i=0 to 3
        for j=0 to 3
Write(m(j,i)+j" ") (m(i,j) & " ")
    Readline()
    End Sub End Module
```

Add two Matrix of each 3×3

```
Imports System.Console
Module Module1
    Sub Main()
Dim a(3,3), b(3,3), C(3,3) As Integer
    Dim i, j As Integer
Writeline("Input value of 3×3 for 1st 3×3
        Matrix --")
    for i=0 to 2
        for j=0 to 2
Write("Input for [0] row [1] col: "i+1, j+1)
        a(i,j) = Readline()
        Next j
    Next i
Writeline("3×3 1st Matrix")
    for i=0 to 2
        for j=0 to 2
Write(a(i,j) & " ")
        Next j
    Writeline()
    Next i
    Readline()
Writeline("Input value for 2nd 3×3 Matrix--")
    for i=0 to 2
        for j=0 to 2
Write("Input for [0] row [1] col: "i+1, j+1)
        b(i,j) = Readline()
        Next j
    Next i
```

```
Writeline(" 3x3 2nd Matrix")
    for i=0 to 2
        for j=0 to 2
Write( b(i,j) & " ")
            Next j   Writeline()
        Next i
    Readline()
```

Addition

```
for i=0 to 2
    for j=0 to 2
c(i+j) = a(i+j) + b(i+j)  Next j  Next i
Writeline(" 3x3 1st Matrix")
    for i=0 to 2
        for j=0 to 2
Write( a(i,j) & " ")
            Next j   Writeline
        Next i
    Readline()
Writeline(" 3x3 2nd Matrix")
    for i=0 to 2
        for j=0 to 2
Write( b(i,j) & " ")
            Next j
        Writeline()
        Next i
    Readline()
Writeline(" Addition of 2 - 3x3 Matrix ")
```

```
for i=0 to 2
    for j=0 to 2
        Write(c(i,j) & " ")
    Next j
    Writeline() Next i
                Readline()
            End Sub
        End Module
```

Q Input name of 10 students & finally prints

```
Imports system.Console
Module Module1
    Sub Main()
Dim n(10) As Integer
 Dim i As Integer

Writeline("Enter 10 Students Name :")
    for i=0 to 9

    Write("Students Name {0}:", i+1)
     n(i) =Readline()
    Next
Writeline("10 Students Name")
    for i=0 to 9
Writeline(n(i) &" ") Next i
    Readline()
    End Sub
End Module
```

Q Multiple two Matrix of each 3×3

Same process for input of Add Matrix

```
for i=0 to 2
    for j=0 to 2
c(i,j) = a(i,j) * b(i,j)
    Next j
Next i
```

9 Write VB.Net console program to calculate factorial of a number using user defined function.

```
Imports System.Console
Module Module1
    Sub Main()
        Dim n As Integer
        Dim fact As long
        Write("Enter a number:")
        n = Readline()
        fact = factorial(n)
        WriteLine("Factorial=" & fact)
        Readline()
    End Sub
Function factorial(ByVal x As Integer) As long
    Dim f As long = 1
    While x > 1
        f = f * x
        x = x - 1
    End While
    Return f
End function
End Module
```

9 Write VB.Net console program to calculate factorial of a number using user-defined procedure.

```
Imports System.Console
```

```
Module Module1
    Sub Main()
Dim n As Integer
    Console("Enter a number:")
        n = Readline()
        factorial(n)
        Readline()
    End Sub
Sub factorial(ByVal x As Integer)
    Dim f As Long = 1
        while x > 1
            f = f * x
            x = x - 1
        End while
Console.Writeline("Factorial =" & f)
    End Sub
End Module
```

Argument Passing Technique

- Pass - By - Value
- Pass - By - Reference

Pass - By - Value :- In this technique, any changes made in called function / procedure are not seen in calling function / procedure. This is default argument passing technique.

The ByVal Keyword is used for passing argument by value.

Ex:-

```
Imports System.Cosole
Module Module1
Sub Main()
Dim n As Integer
n=100
Writeline("Value of n before change =" & n)
call change(n)
Writeline("Value of n after change =" & n)
Readline()
End Sub
Sub change(Byval n As Integer)
n = n+20
End Sub
End Module
```

Output

Value of n before change = 100
Value of n after change = 100

Pass-By-Reference :— In this argument passing technique, any changes made in called function are also seen in calling function.

The ByRef Keyword is used to achieve Pass-By-Reference technique.

Eg:—

```
Imports System.Console
Module Module1
    Sub Main()
        Dim n As Integer
        n = 100
        WriteLine("Value of n before change=" & n)
        call change (n)
        WriteLine("Value of n after change=" & n)
        ReadLine()
    End Sub
    Sub change (ByRef n As Integer)
        n = n + 20
    End Sub
End Module
```

* Output:—

Value of n before change = 100
Value of n after change = 120

* Object - Oriented Programming in VB.Net

Class: It encapsulates the properties & methods / behaviours of objects.

Defining Class

Syntax

Class <class-Name>
<Access-specifier> <data-type>

<Access_Specifier> <function>

End Class

* Creating Object

Syntax

Dim <object> As New <Class_Name>

* Accessing Member of Class

<object>.<member_Name>

Q. Write an object-oriented program in VB.Net to perform I/O of details include Roll, Name & Course.

```
Imports System.Console
Module Module1
Class Student
Private roll As Integer
Private name As String
Private Cs As String
Public Sub getdata()
Write("Enter Roll:")
roll = Readline()
Write("Enter Name:")
name = Readline()
Write("Enter Course:")
Cs = Readline()
End Sub
```

```
Public sub show()
Writeline(" Roll Number =" & roll)
Writeline(" Name = " & name)
Writeline(" Course =" & cs)
    End sub
    End class
    Sub Main()
Dim Subj As New Student
    Subj.getdata()
    Subj.show()
        Readline()
    End Sub
    End Module
```

CONSTRUCTOR

It is a special method (function) define with name New(). It executes automatically when object of class is created. It is used to initialize object.

Syntax

```
Class XYZ
{
Public sub New() Constructor
{
}
}
```

Ex:-

```
Imports System.Console
Modul Module1
    Class Student
Private roll As Integer
Private Name As String
Private Course As String
Public Sub New()'Constructor
roll = 101
Name = "Shivam Tiwari"
Course = "BCA"
End Sub
Public Sub show()
Writeline("Roll Number = " & roll)
Writeline(" Name = " & Name)
Writeline(" Course = " & Course)
End Sub
End Class
Sub Main()
Dim sobj As New student
sobj.show()
Readline()
End Sub
End Module
```

Parameterized Constructor (Argumented Constructor)

A Constructor can be overloads means a class can here more than one constructor

each having diff. no. of arguments & its data type.
A constructor with no argument is called default constructor.

Q Design a Student Class having 3-data members to store roll, name & course. Define Constructor to set value into data member.

```
Imports System.Console
Module Module1
    Class Student
Private roll As Integer
Private name As String
Private cs As String

Public Sub New() 'Default Constructor
    roll = 1001
    name = "Shivam Tiwari"
    cs = "BCA"
    End Sub
Public Sub new (Byval rn As String, Byval sn
As String, Byval c As String) '3-arg
constructor
    roll = rn
    name = sn
    cs = c
    End Sub
```

```
Public Sub show()
Writeline("Roll Number = " & roll)
Writeline("Name=" & cs)
    End sub
    End class
    Sub Main()
Dim sobj1 As New Student
Dim sobj2 As New student(1002, "Sim", "MBA")
Dim sobj3 As New student(1003, "Shah", "BCA")
    Writeline("Record of 1st Student")
        sobj1.show()
    Writeline("Record of 2nd Student")
        sobj2.show()
    Writeline("Record of 3rd Student")
        sobj3.show()
            Readline()
        End sub
        End Module
```

DESTRUCTOR

A destructor is a method which execute automatically when object is released from memory. ~~A destructor is defined with name~~ A destructor is define with name Finalize

Syntax

```
Protect Overrides Sub Finalize()
    ===
    ===
End Sub
```

Method Overloading

It is a feature of OOP that allows to define set of methods with same name but each method should have different numbers of argument or its data type. Overloading of method is not done with return a type

Q. Overload Area() method to calculate area of circle & rectangle.

```
Imports System.Console
Module Module1
    Class MLoad
Public overloads Sub Area (Byval r As Single)
```

```
Dim a As Single
    a = 3.14 * r * r
Writeline("Area of Circle = " & a & "Sq cm")
End Sub
Public Overloads Sub Area(ByVal l as Integer, ByVal
b As Integer)
    Dim a As Integer
    a = l * b
Writeline("Area of Rectangle = " & a & "Sq cm")
End Sub    End Class
Sub Main()
Dim Cr As New MLoad
    Cr.Area(5,6)
    Cr.Area(5:3)
    Readline()
    End sub()
    End Module
```

INHERITANCE

It is a feature of OOP that allows to reuse the properties and method of old class in new class. The old class whose properties are inherited is called base class or super class. The new class that inherits of property of old class is called ~~Sub~~ ~~class~~. child or derived class.

Class A (Base Class
Class B (Sub-Class)

Types of Inheritance

i) Single Inheritance
ii) Multiple Inheritance
iii) Multilevel Inheritance
iv) Hierarchical Inheritance

Single Inheritance :— Sub class has only one base class.

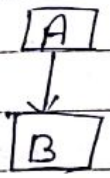


fig: Single

Multiple Inheritance :— Sub class has more than one base class.

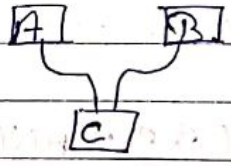


fig: Multiple

Multilevel Inheritance :— Sub class has base which is derived class.

Hierarchical Inheritance :- Multiple Sub Class has same base class

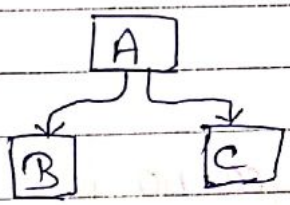


fig : Hierarchical

Note

Multiple inheritance is not supported in VB.Net.
Multiple inheritance in VB.Net is achived through "Interface".

Defining Sub Class

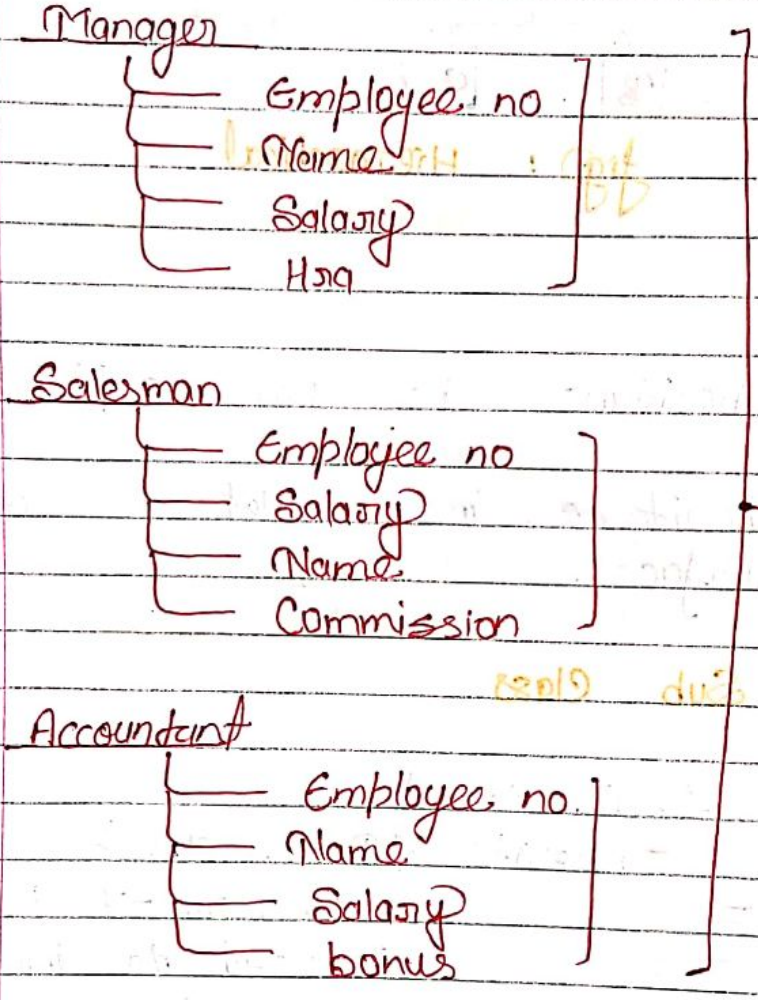
```
Class <Sub. Class>
    Inherits <Base. Class>
    ==========
    End Class
```

The inherit keyword is used to inherit class.

eg:-

```
Class A
Private x As Integer
public Sub xyz()
    ==
End Sub
Class B          → Sub Class
Inherit A        → Base class
    ==
End Class
```

Q. Write an object-oriented program in VB.Net to maintain details of various categories of employee :—




```
Imports System.Console
Module Module1
    Class Employee
Protected eno As Integer
Protected name As String
Protected sal As Single
    End class
    Class Manager
```

```
Inherits Employee
Private hra As Integer
Public Sub getdata()
Write("Employee No:")
eno = Readline()
Write("Name:")
name = Readline()
Write("Salary:")
Sal = Readline
Write("HRA:")
hra = Readline()
End Sub
Public Sub show()
Writeline("Employee No=" & eno)
Writeline("Name=" & name)
Writeline("Salary=" & Sal)
Writeline("HRA=" & hra)
End Sub
End Class
Class Salesman
Inherits Employee
Private com As Single
Public Sub getdata()
Write("Employee No:")
eno = Readline()
Write("Name:")
name = Readline()
Write("Salary:")
sal = Readline()
Write("Commision:")
```

```
com = Readline()
    End Sub
    Public Sub show()
Writeline ("Employee=" & eno)
Writeline ("Name =" & name)
Writeline ("Salary =" & sal)
Writeline ("Commision-" & comm)
    End Sub
    End Class
    Class Accountant
        Inherits Employee
    Private bonus As Integer
    Public Sub getdata()
    Write (" Employee :")
     eno = readline()
  write (" Name :")
   name = readline()
  write (" Salary :")
   Sal = readline()
  write (" Bonus :")
    Bonus = readline()
    End sub
Public Show()
writeline (" Employee No =" & eno)
Writeline (" Name = " & name)
writeline (" Salary = " & Sal)
writeline (" Bonus = " & Bonus)
    End Sub
    End class
```

```
Sub Main()
Dim M As New Manager
Dim S As New Salesman
Dim A As New Accountant
Writeline(" Enter Record of Manager ")
    M.getdata()
Writeline(" Enter record of Salesman")
    S.getdata()
Writeline(" Enter record of Accountant")
    A.getdata()
Writeline(" = = = = = = = = = = ")
Writeline(" Record of Manager ")
    M.Show()
Writeline(" = = = = = = = = = ")
Writeline(" Record of Salesman ")
    S.Show()
Writeline(" = = = = = = = = = ")
Writeline(" Record of Accountant")
    A.Show()
    Readline()
End Sub
End Module
```

* Constructor in Inheritance

When the base class and derive class both contains constructor then the constructor of base class is executed befor the constructor of sub class.

Example :—

```
Imports System.Console
Module Module1
    Class Teacher
        Public Sub New()
            Writeline(" I am teacher")
        End Sub
    End Class
    Class Student
        Inherits Teacher
        Public Sub New()
            Writeline(" I am Student")
        End Sub
    End Class
        Sub Main()
            Dim s As New Student
            Readline()
        End Sub
    End Module
```

Output

```
I am Teacher
I am Student
```

* Argumented Constructor in Inheritance

When the base class & derived class both has argumented constructor then all arguments are first passed to derived class & then required no. of arguments is passed to base class using Mybase keyword.

Syntax

MyBase.New (Arguments to pass to base class)

Example :—

```
Imports System.Console
Module Module1
    Class Teacher
    Private tid As Integer
    Private tname As String
Public Sub New(ByVal tid1 As Integer, ByVal
    tname1 As Integer String)
        tid = tid1
        tname = tname1
        End Sub
    Public Sub ShowTeacher()
        WriteLine(" Teacher id = " & tid)
        WriteLine(" Teacher name = " & tname)
        End Sub
    End Class
    Class Student
        Inherits Teacher
```

```
Private roll as Integer
Private sname As String
Public Sub New(ByVal r As Integer, ByVal
s As string , ByVal tid1 As Integer,
ByVal tname1 As String)
MyBase.New (tid1, tname1)
roll = r
sname = s
End Sub
Public Shows Sub showstudent()
Console.Writeline ("Roll No = " & roll)
Console.Writeline (" Student name = " & sname)
End Sub
End Class
Sub Main()
Dim s As New student (1001, " Atul kumar",
420, "Mr. C.K Mishra")
s.showstudent()
s.showteacher()
Readline()
End sub
End Module
```

Output

```
Roll No = 1001
Student Name = Atul kumar
Teacher id = 420
Teacher Name = Mr. C.K Mishra
```

METHOD OVERRIDING

When the base class and derived class both contains ~~constructor~~ method of same signature then the method of derived class overrides the method of base class.

The base class method must be defined with overridable keyword and the derived class method must be defined override keywords.

Ex:—

```
Imports System.Console
Module Module1
    Class Teacher
Public Overridable Sub show()
Writeline("I am Teacher")
End Sub
    End Class
Class Student
    Inherits Teacher
Public Overrides Sub show()
Writeline("I am Student")
End sub
    End class
Sub Main()
Dim s As New Student
s.show()
```

```
Readline()
End Sub
End Module
```

Output

I am Student

MustInherit :- This keyword is used to define abstract class. A class which can only be inherited by its objects can not be created is called abstract class.

Syntax

```
MustInherit Class Employee
==
End Class
```

Not Inheritable :-

This keyword is used to define a class that can't be inheritable.

Syntax

```
NotInheritable Class Book
==
End Class
```

Mustoverride

This keyword allows to declare a method/ procedure is base class that must be overridden in sub class.

Syntax

```
Mastoverride Sub Show()
```

Note

A class containing declaration of mustoverride method must be defined with mustinherit Keyword.

Example:—

```
Imports System.Console
    Module Module1
Must Inherit Class Book
    Mustoverride Sub show()
End Class
    Class MyBook
        Inherits Book
Public Overrides Sub show()
Writeline ("hello! Good Morning")
End Sub
    End Class
    Sub Main()
```

```
Dim x As New Book
x.show()
Readline()
End sub
End Module
```

Notoverridable

A method / procedure defined with Notoverridable can't be overriden in sub class.

Syntax

```
Notoverridable Sub cy2()
=========
End Sub
```

My Base :—

It is used access the member of base class from within sub-class.

Syntax

```
MyBase.member of base class
```

* Interface

Interface in VB.Net is similar to class with a difference that

interface contains only method declaration and its body is defined in sub class that inherits the interface.
And Interface is used to achive multiple inheritance more than one interface can be inherited.

Defining Interface

The Interface keyword is used to define interface.

Syntax

```
Interface <name>
    ===
    ===
End Interface
```

Ex:—

```
Interface MyInterface
    Sub First()
    Sub Second()
End Interface()
```

Inheriting Interface

The Implements keyword is used to inherits interface.

Syntax

Class xyz
Implements <interface1>, <interface2>
=
End Class

Example:-

```
Imports System.Console
Module Module1
Interface Iface1
    Sub put1()
End Interface
Interface Iface2()
End Interface
Class Testing
    Implements Iface1, Iface2
Public Sub put1() Implements Iface1.put1
WriteLine("Hello! Good Morning")
End Sub
Public Sub put2() Implements Iface2.put2
WriteLine("Good Nigh")
End Sub
End Class
Sub Main()
Dim x As New Testing
x.put1()
x.put2()
Readline()
End Sub
End Module
```

WINFORM

This feature of VB.Net allows to develop windows application. An application which run in GUI environment is called windows application. VB.Net provides ready to use GUI Component and therefore a programmer need not write any code for creating GUI. Even resizing and positioning of component is done using drag & drop operation

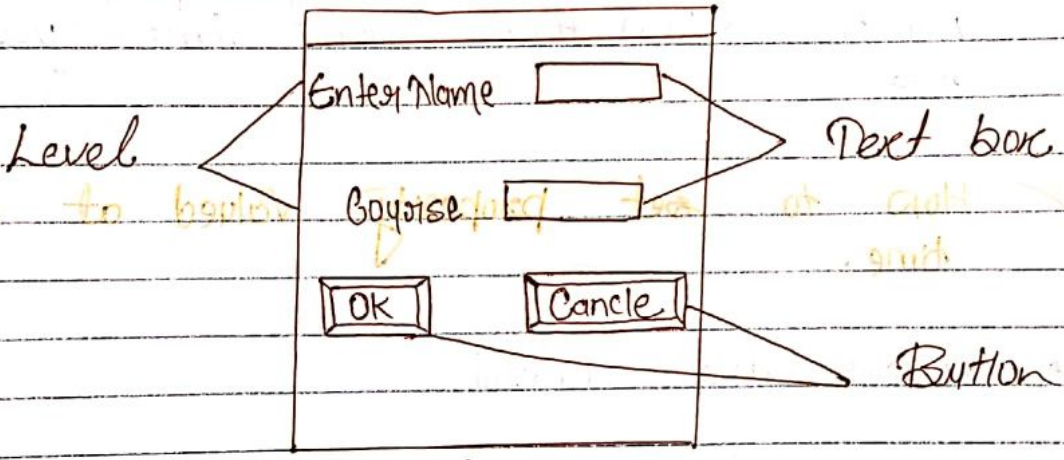


Fig: GUI

Control

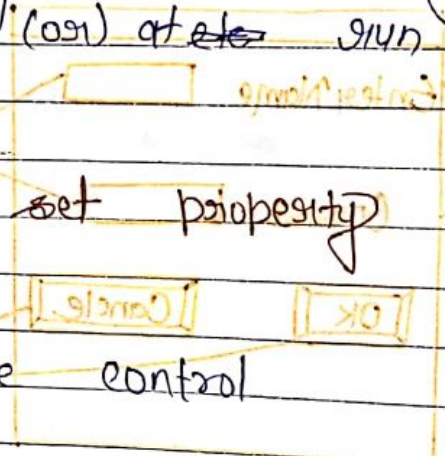
Any GUI Components such as text box, radio button, button etc are called Controls.
following 3 terms are associated with any control :—

i) Properties
ii) Method
iii) Event

Properties:—

The characteristics of control are called properties. The default appearance of a control can be changed by setting new valued to properties.

The properties can be changed either at design time through properties window (or) at run time using code.



How to set property valued at design time.

- Select the control
- Set new value in property window of property.

How to set property value at run time.

<Control-Name>.<property-Name> = <value>

Method :—

The in-built code which is run to complete a task.

Syntax

Control_Name.method_Name

Event :—

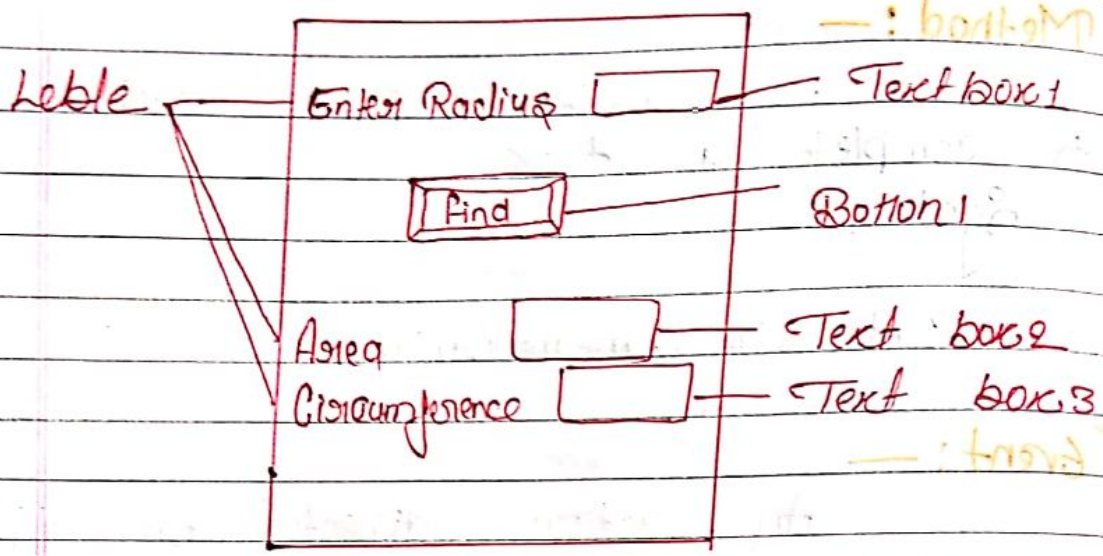
An action directed on application is called event.
All GUI - application are called event-driven application as any task is performed on occurance of event.

The codes / instruction that executes on the occurance of event is called event handler
A procedure that is called on event procedure.

Syntax

```
Sub Control_Name_event name ()
    ==========
    ==========
End Sub
```

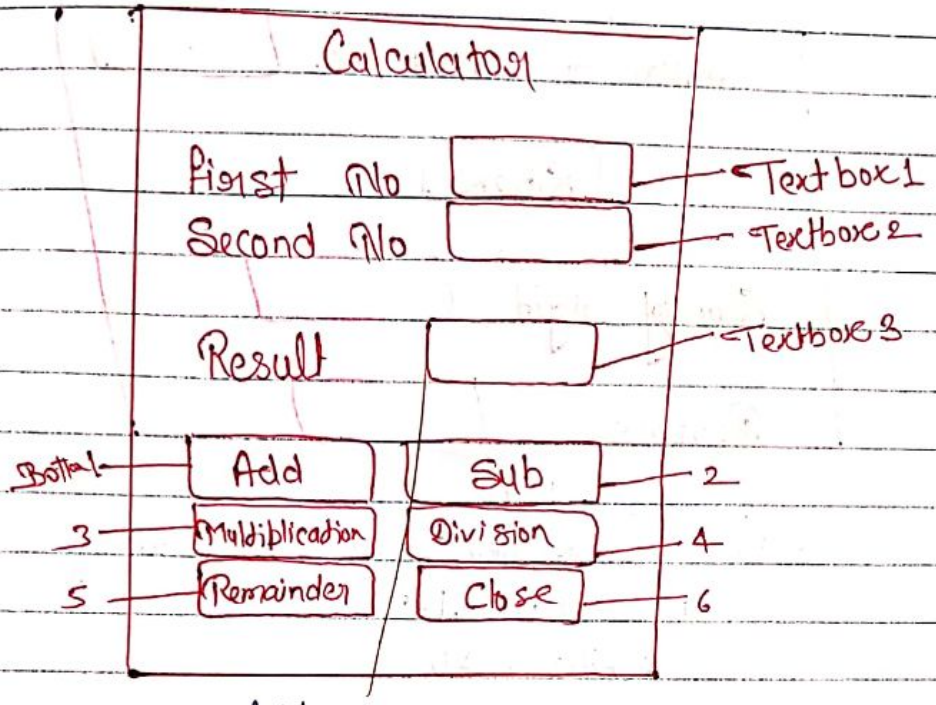
Q. Develop GUI- application in VB.Net to calculate area & circumference of a circle.



Sol~ Public Class form1
Private Sub Button1_Click(Byval

Dim r, a, c As Single
r = Textbox1.Text
a = 3.14 * r * r
c = 2 * 3.14 * r
Textbox2.Text = a
Textbox3.Text = c
End Sub
End class

Q Create a calculator program in VB.Net



Add

Textbox3.Text = Val(TextBox1.Text) + Val(TextBox2.Text)

Sub

Textbox3.Text = Textbox1.Text - Textbox2.Text

Multiplication

Textbox3.Text = Textbox1.Text * Textbox2.Text

Remainder

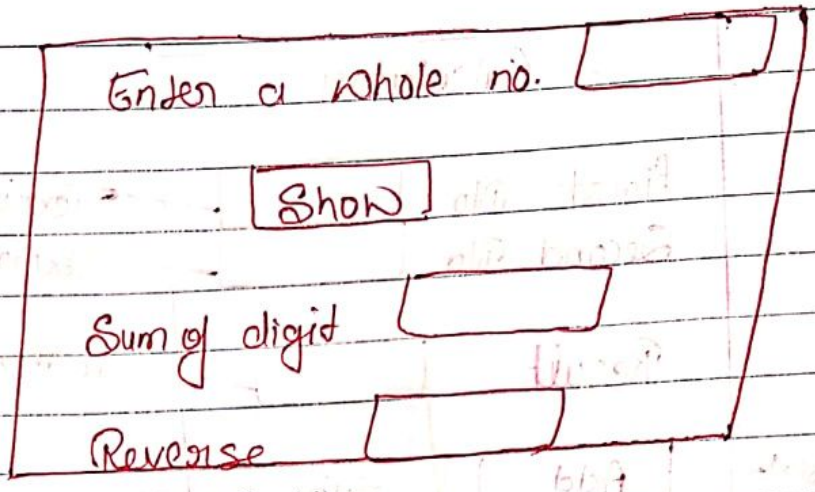
Textbox3.Text = Textbox1.Text Mod Textbox2.Text

Division

Textbox3.Text = Textbox1.Text / Textbox2.Text

Close

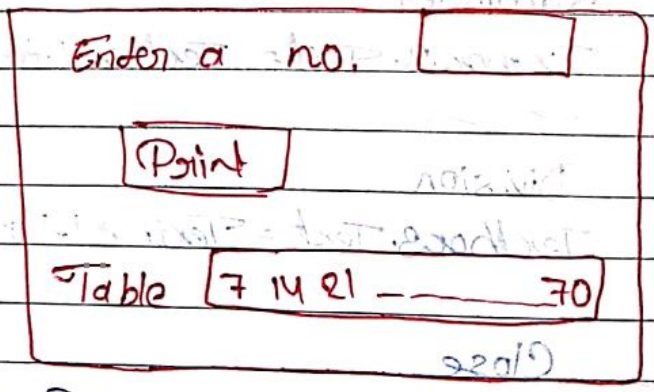
End



Show Button

```
Dim n, r, rev, sum As Integer
n = Textbox1.Text
While n > 0
r = n Mod 10
sum = sum + r
rev = rev * 10 + r
n = n \ 10
End While
Textbox2.Text = sum
Textbox3.Text = rev
```

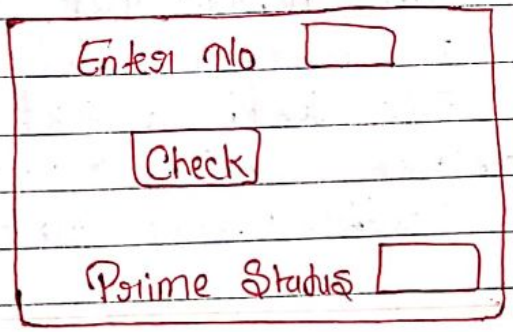
Display table of a number



Show Button

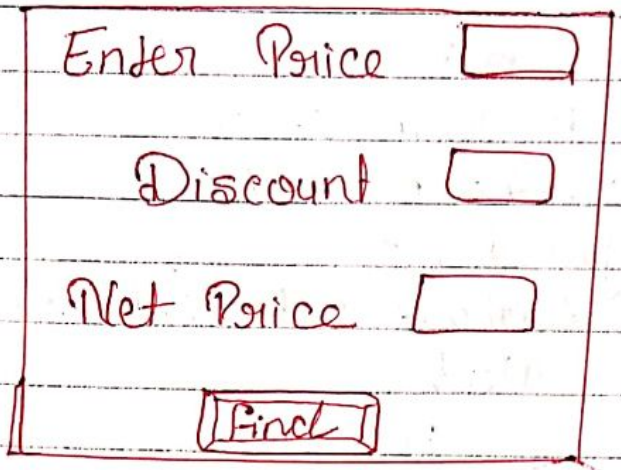
```
Dim n,i As Integer
    n= textbox1.Text
    Textbox2.Text = ""
    for i=1 to 10
    textbox2.Text=Textbox2.text + str((n*i))
    Next
        End Sub
        End Class
```

Q.




```
Dim n,j,p As Integer
    n = Textbox1.text
    for j=2 to n/2
      If n Mod j=0 then
        P=1
      End for
        End If
    If p=0 then
      Textbox2.text = ("yes")
    Else
      Textbox2.text = ("No")
    End if
```

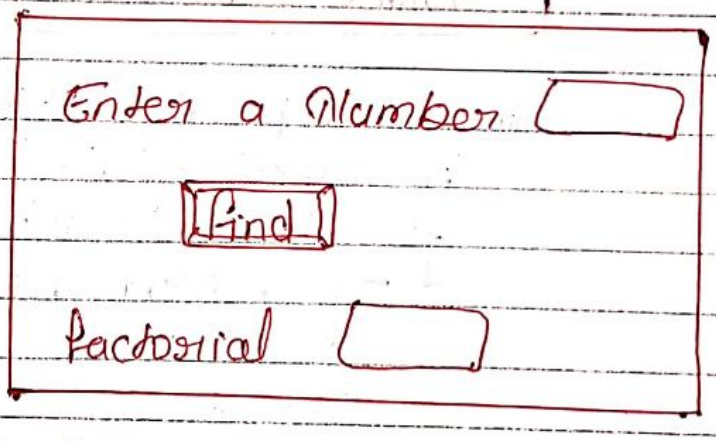
Q.



Find Botton

```
Dim n, Dis, Net, Tdis As Integer
n = textbox1.text
Dis = textbox2.text
Tdis Net = n/100 * Dise / Net = n - Tdis
Textbox3.text = Net
```

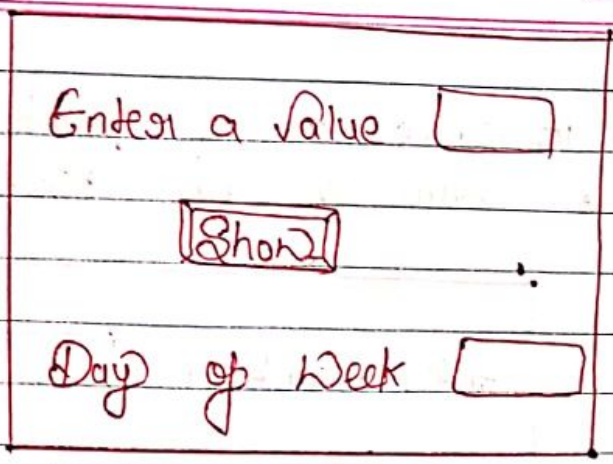
Q.



Find Botton

```
Dim n, fact As Integer
n = textbox1.text
While > 1
fact = fact * n
n = n - 1
End While
Textbox2.text = fact
```

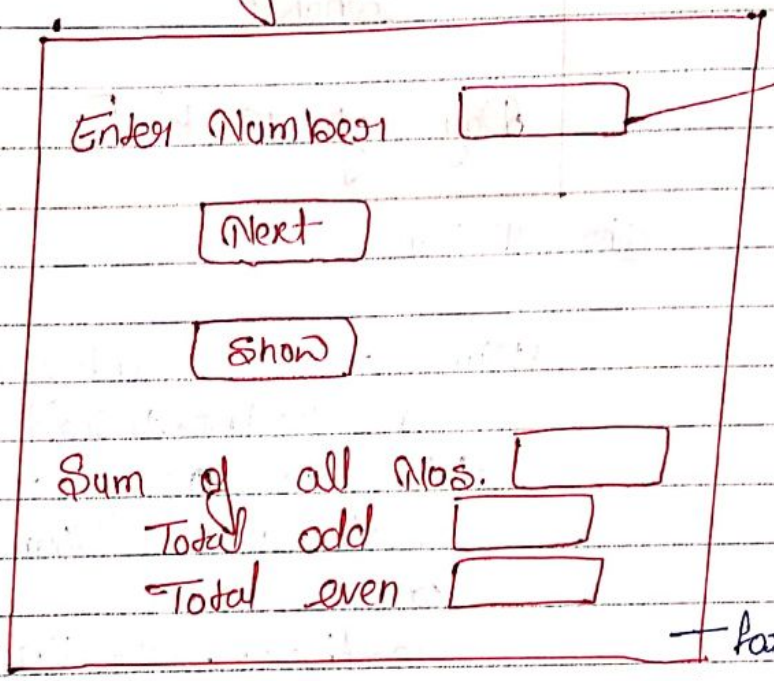
Q.



Show Button

```
Dim n As Integer
n= textbox1.text
Select Case n . Case1
 Textbox2.text=("Sunday")
 Case 2
 Textbox2.text=("Monday")
 Case 3
 Textbox2.text = ("Tuesday")
 Case 4
  Textbox2.text = ("Wednessday")
 Case 5
 Textbox2.text =("Thursday")
 Case 6
  Textbox2.text = ("Friday")
 Case 7
 Textbox2.text = ("Saturday")
```

Q. Input 10 whole nos. & prints total odd, even & sum of all nos.




```
Dim num(9) As Integer
Dim i As Integer

Private Sub Form1_Load(ByVal sender, object,
ByVal e As System.EventArgs) Handles MyBase.Load
    cmdshow.Enbled = False
End Sub

cmdNext_Click
    num(i) = TextBox1.text
    i = i+1
    Textbox1.Text = ""
    Textbox1.Focus()
    If i = 10 then
```

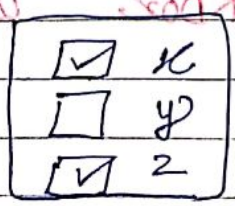
```
cmdNext.Enabled = false
cmd Show. Enabled = True
    End If
End Sub
```

Cmd Show

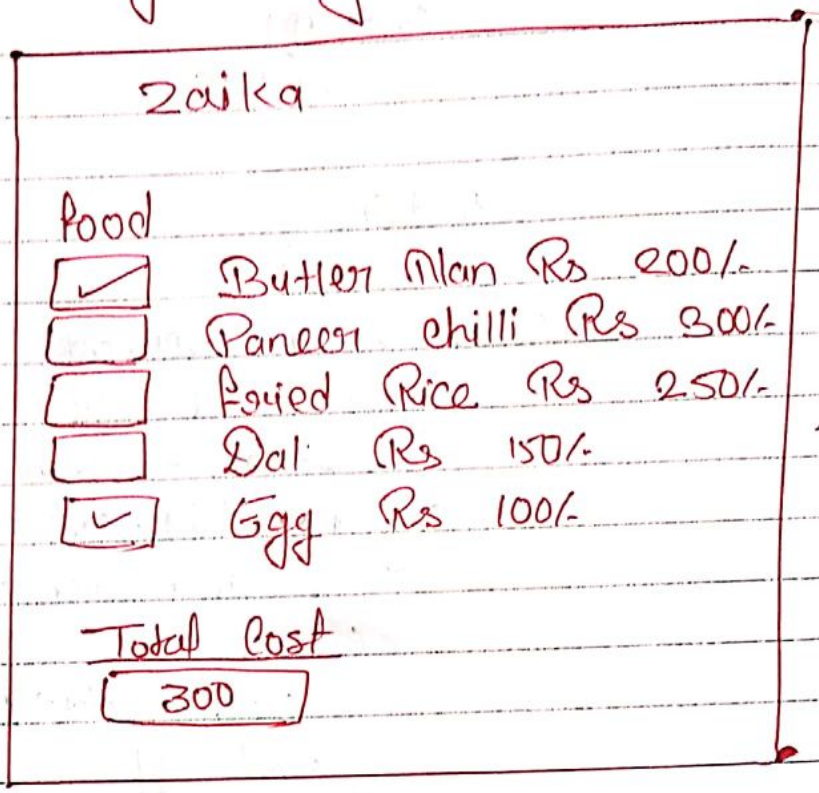
```
Dim sum, even, odd As Integer
for i = 0 to 9
    sum = sum + num(i)
If num(i) Mod 2 =0 then
    even = even + 1
Else
    odd = odd + 1
End If
Next
Textbox2.Text = Sum
Textbox3.Text = odd
Textbox4.Text = even
End Sub
End Class
```

* Checkbox Control

Provides list of items from which one or more items can be selected.



Develop program to calculate Total cost of food of a restaurant.




```
Dim bill As Integer
Checkbox 1.

If Checkbox1.Checked = True then
    bill = bill + 200
Else
    bill = bill - 200
End If
Textbox1.text = bill
```

Same

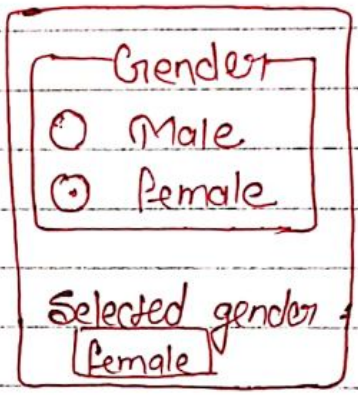
All checkbox use same condition

Radio Button

It provides multiple options from which only one option can be selected. A dot (·) appears in hollow circle of selected item.

○ Male ⊙ Female

⊖ Write VB.Net program to Display name of selected gender.



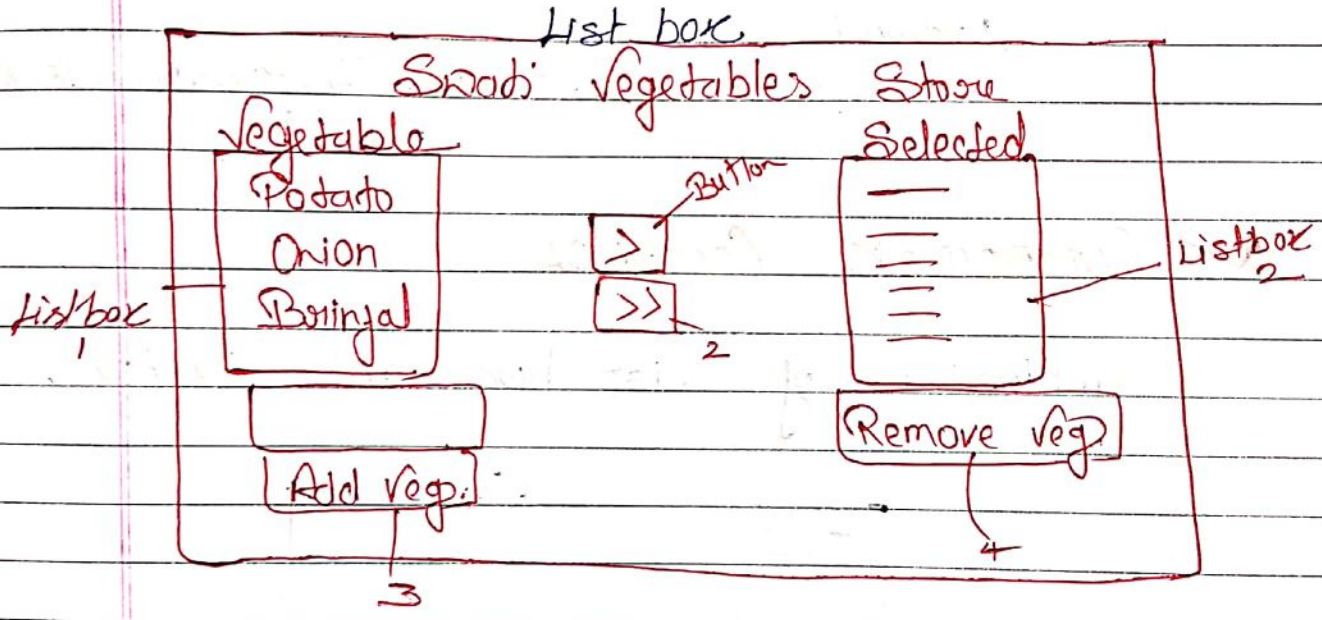

```
RadioButton1
If RadioButton1.checked = True then
    TextBox1.Text = "Male"
End if

RadioButton2
If RadioButton2.checked = True then
    TextBox1.Text = "female"
End if
```

LIST BOX

provides list of options in rectangular box and allows the user to select one or more options.

List box



Form1 Load

```
Listbox1.Items.Add("Potato")
Listbox1.Items.Add("Onion")
Listbox1.Items.Add("Tomato")
Listbox1.Items.Add("Brinjal")
```

Button1

```
Listbox2.Items.Add(ListBox1.selectedItem)
```

Button2

```
Dim i As Integer
for i=0 to Listbox1.Items.count -1
Listbox2.Item.Add (Listbox1.items(i))
Next
```

Button3

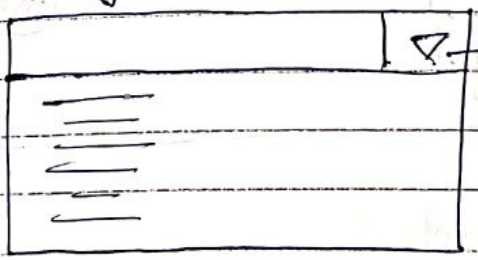
```
Listbox1.Items.Add (TextBox1.text)
TextBox1.clear()
```

Button4

```
Listbox2.Items.Remove (ListBox2.selectd
                       Item)
```

COMBOBOX CONTROL

Combination of list box and textbox.



Q Display fee of course selected by user.

Select Course [BCA]

Fee [ ]

Course	Fee
MCA	21800
BCA	15050
BBA	17000
MBA	52000
DCA	3500
ADCA	8400

```
Form1_Load
    Combox1.Items.Add("BCA")
    Combox1.Items.Add("MCA")
    Combox1.Items.Aeld("BBA")
    Combox1.Items.Add("MBA")
    Combox1.Items.Add("DCA")
    Combox1.SelectedIndex = 0

Combox1_SelectedIndexChanged
    If Combbox1.Text = "BCA" then
        Textbox1.Text = 15050
    Else If Combbox1.Text = "MCA" then
        The Textbox1.Text = 21800
    Else If Comboboxl.Text = "BBA" then
        Textbox1.Text = 17000
    Else If Comboboxl.Text = "MBA" then
        Textboxl.Text = 52000
    Else If Comboboxl.Text = "DCA") then
        Textboxl.Text = 3500
    End If
```

TIMER CONTROL

This Control automatically triggers event based on time. It calls tick event after the given time interval.

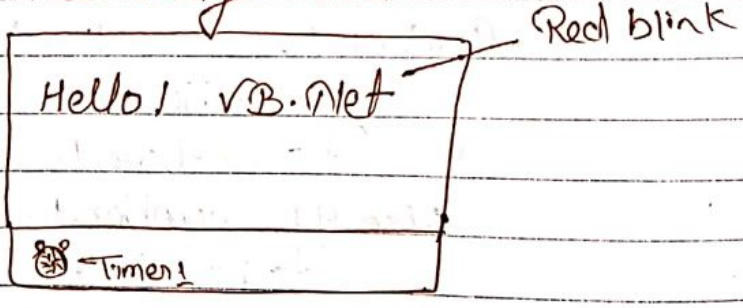
Property

- .Interval — Allows to set time.

Method

.Start() — Call tick event
.Stop() — Stop tick event

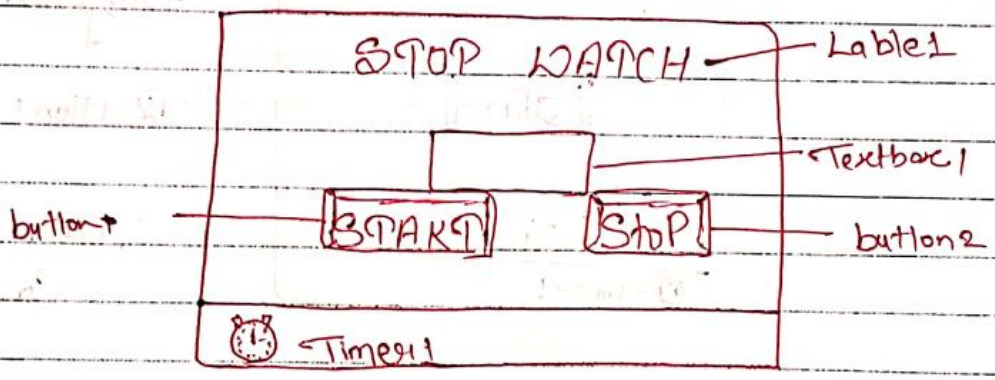
Q. Create a blinking text




```
Public Class Form1
    Dim x As Integer
Form1_Load
    Timer1.Interval = 100
    Timer1.Start()
End Sub
Timer1_Tick
    If x=0 then
```

```
Lable1.Visible = False
    c=1
Else If x=1 then
    Label1.Visible = True
        x=0
    End If
End Sub
    End Class
```

Q Create Stop-Watch program in VB.Net.

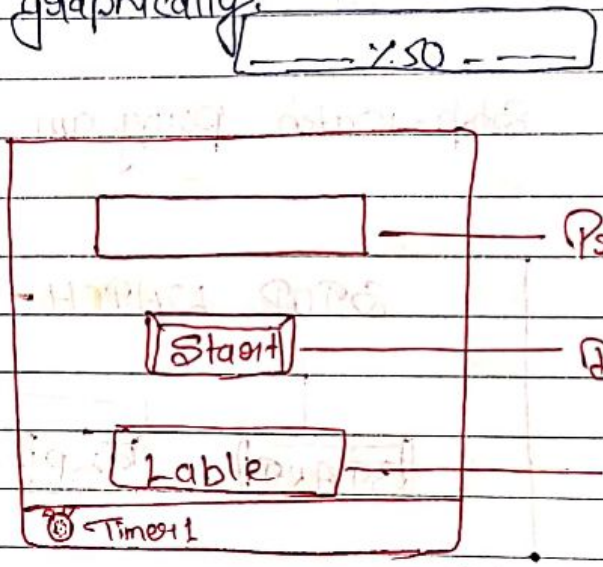



```
Public Class form1
    Dim x As Integer
Load
    Timer1.Interval = 100
Button2_1
Timer1.Start()
    Timer1_Tick
    Textbox1.text = x
        x = x+1
    If x = 3001 Then
        Timer1.Stop()
```

Button2
Timer1.Stop()
End Class

PROGRESSBAR CONTROL

It shows the progress of task being done graphically.

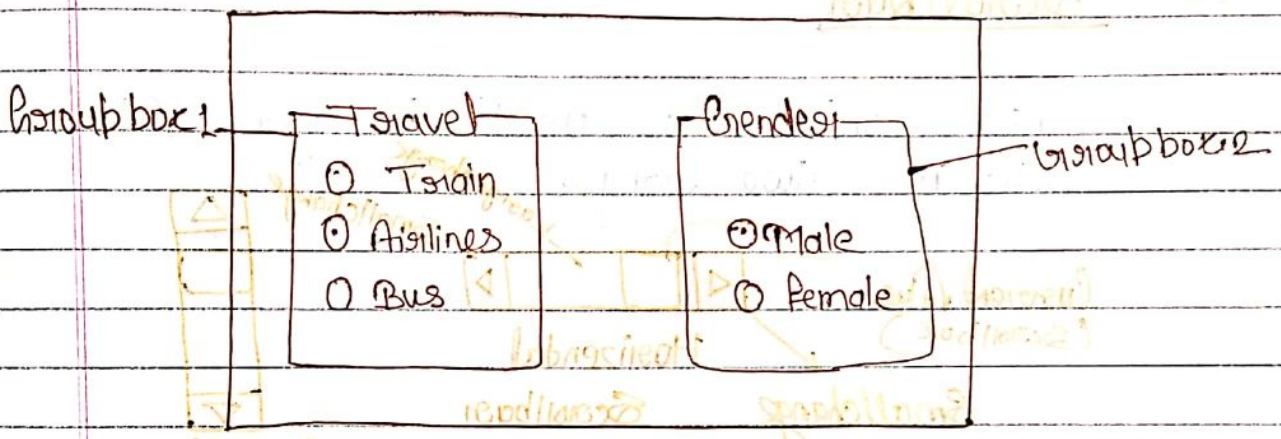


Public class Form1
Dim x As Integer
Form1.Load
progressBar1.Minimum = 0
progressBar1.Maximum = 200
Timer1.Interval = 500
Button1
Timer1.Start()
Timer1.Tick
progressbar1.Value = x
x = x + 1
If x = 201 then

```
Timer1.Stop()
Lable1.Text = "Task Completed"
End If
```

GROUPBOX CONTROL

This control is used to make grouping of option/Radio button.



PICTUREBOX CONTROL

This control allows to insert / display / image / picture.

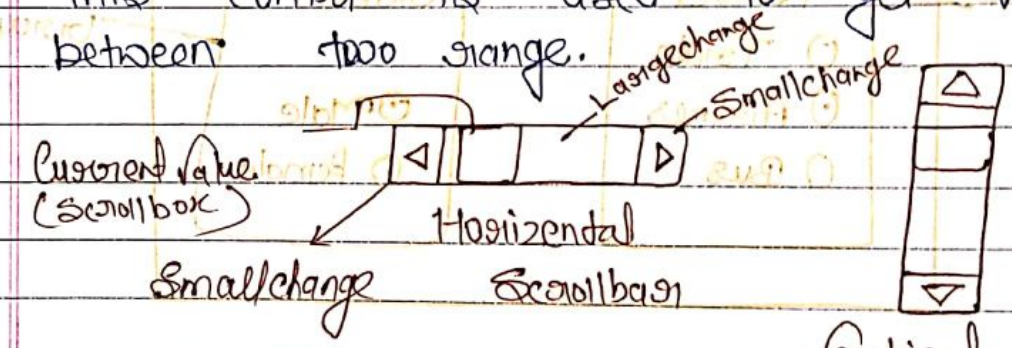
Property

Image :- Allows to specify image file to display.

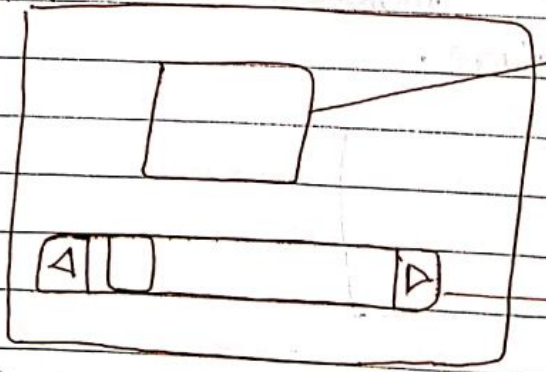
SizeMode :- Allows to specify how the image will be adjusted.
Normal
Streatch Mode
Auto size
Centere
Zoom

Scrollbar

This control is used to get value between two range.



Q. Display value in range 1 to 100 using Scrollbar.



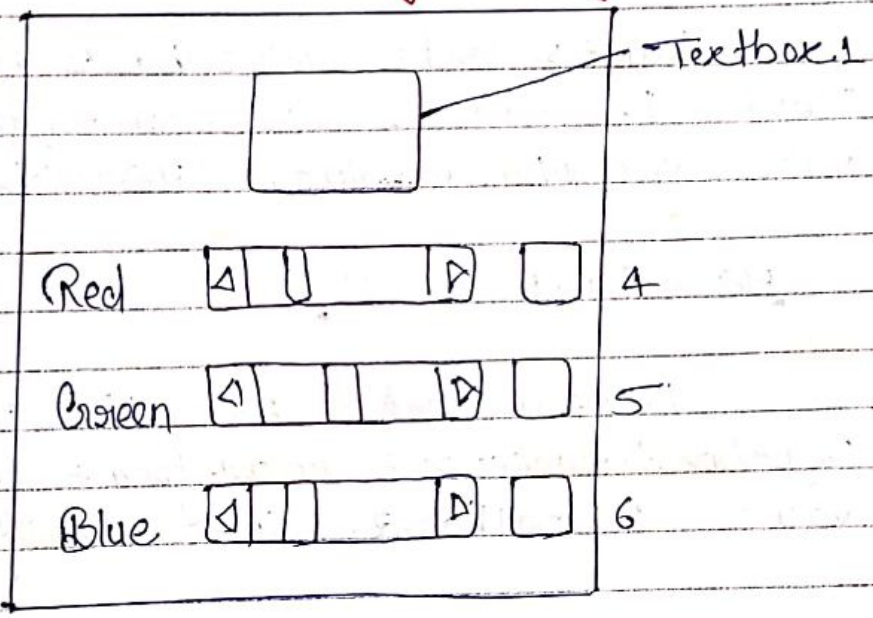
Property

Smallchange : 1
Largechange : 1
Minimum : 1
Maximum : 100

Private Sub HScrollbar1_Scroll

Textbox1.Text = HScroll1.Value
End Sub
End Class

Q. Create Colour mixing program in VB.Net



Form1_Load

HScrollbar1.Maximum = 255
HScrollbar2.Maximum = 255
HScrollbar3.Maximum = 255
HScrollbar1.LargeChange = 1
HScrollbar1.LargeChange = 1
HScrollbar1.LargeChange = 1
Lable4.Text = 0
Lable5.Text = 0
Lable6.Text = 0

HScrollbar1 - Scroll

Lable4.Text = HScrollbar1.Value
Textbox1.Backcolor = Color.fromArgb(HScrollbar1.Value, HScrollbar2.Value, HScrollbar3.Value)

HScrollbar2 - Scroll

Lable5.Text = HScrollbar2.Value
Textbox1.Backcolor = Color.fromArgb(HScrollbar1.Value, HScrollbar2.Value, HScrollbar3.Value)

HScrollbar3 - Scroll

Lable6.Text = HScrollbar3.Value
Textbox1.Backcolor = Color.fromArgb(HScrollbar1.Value, HScrollbar2.Value, HScrollbar3.Value)

Menustrip.Control

This control used to create menu.

Steps

1) Place Menustrip Control on form
2) Type Menu item in the blank place holder.
3) Set shortcut key by setting in shortkeys property.

OpenFileDialog Control

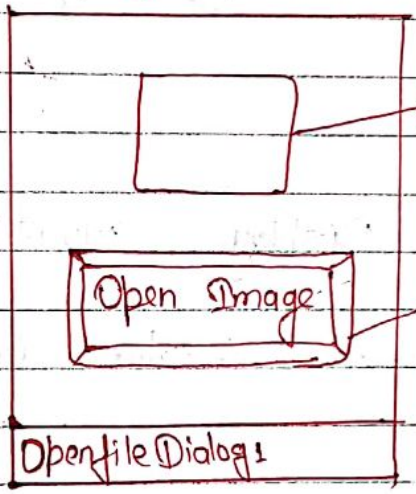
This control display the openfileDialog box and allows the user to select file to be opened.

property

- ShoDialog :— Displays open dialog box.

FileName :— Returns / sets file to open

Q Create Image viewer application in VB.Net.



Button1

OpenFileDialog1.ShowDialog()
PictureBox1.Image = Image.FromFile(OpenFileDialog1.FileName)

SaveFileDialog Control

This control displays the Save As—

dialog box & allows the user to save data into specified file.

property

- ShowDialog :- Display save Dialog box.
- FileName :- Holds file name given by user.

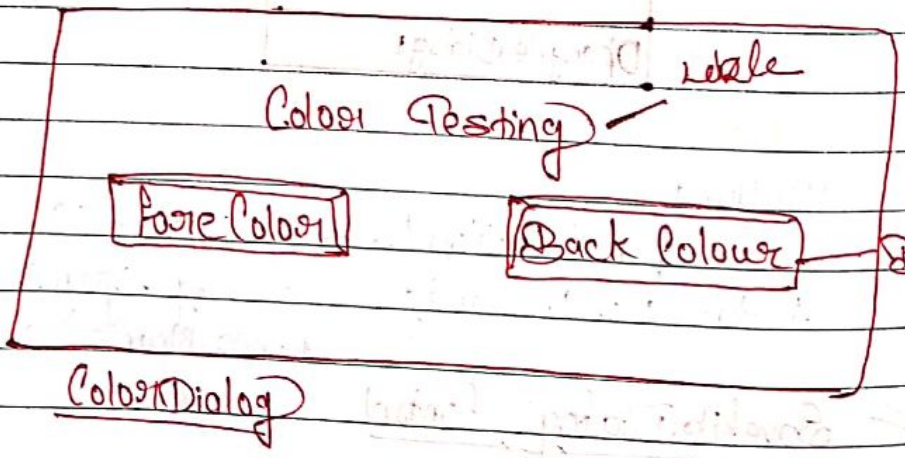
ColorDialog Control

This control displays the colour display box & allows the user to select colour.

Property

ShowDialog :- Display ColorDialog box

Color :- Return Selected Colour.



Button1

ColorDialog1.ShowDialog()
Lable1.ForeColor = ColorDialog1.Color

Button2

ColorDialog1.ShowDialog()
Lable1.BackColor = ColorDialog1.Color

Font Dialog Control

The FontDialog Control displays the font-dialog box & allows the user to select font & apply on desired component

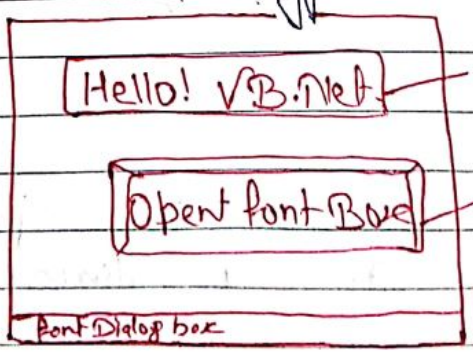
property

Font :— Return selected font

Color :— Return Selected Color.

FontName :— Return name of font.

ShowEffect :— Shows effect option.

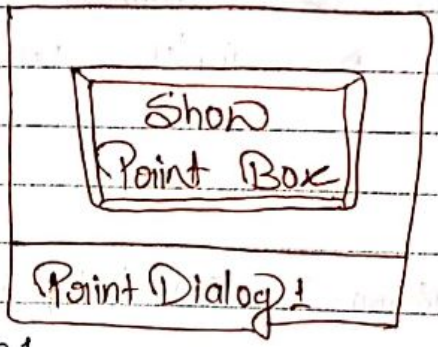


Button1_Click

```
FontDialog1.ShowDialog()
FontDialog1.ShowEffects = True
FontDialog1.ShowColor = True
Textbox1.Font = FontDialog1.Font
Textbox1.ForeColor = FontDialog1.Color
```

PrintDialog Control

This control display the dialog box.



Button1

```
PrintDialog1.ShowDialog()
```

APPLICATION

- SDI
- MDI

SDI :— Allows to open only one window at a time.

MDI :— Allows to open more than one window at a time. Outer window is called MDI - Parent & inner window is called MDI - child.

* How to make MDI - Container / Parent.

- Select form
- Set "True" to IsmdiContainer Property.

How to make MDI - Child

Syntax

<form_Name> - MDI parent = Me

✓ Ritch Textbox Control

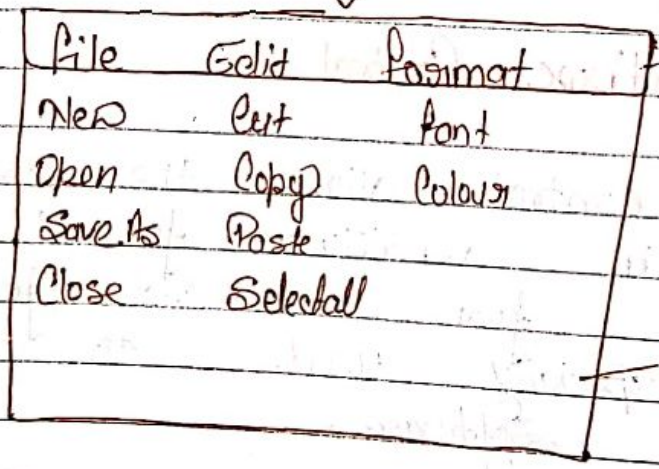
This control allow the user to type text in multiline format & provides properties for advanced formatting. It is specially used to develop editors while Software.

Property

- Forecolor
- Backcolor
- Selection Color

- Selection Font
- SaveFile | Method
- LoadFile
- WordWrap
- Length
- SelectionLength
- TextAlign
- Cut
- Copy
- paste | Method
- Select
- SelectAll
- Undo
- Redo.

Q Develop editor program in VB.Net.



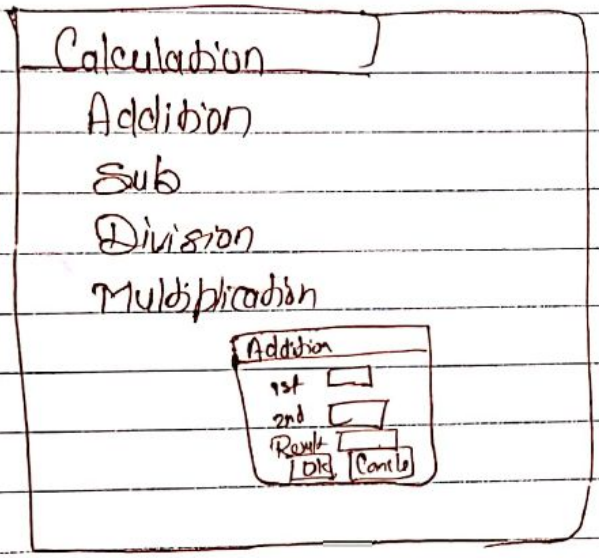
Save As

SaveFileDialog1.ShowDialog()
RichTextBox1.SaveFile(SaveFileDialog1.FileName)

Open Tool

OpenFileDialog1.ShowDialog()
Richtextbox.1.LoadFile (openfile1.FileName)

MDI - Application



Addition Tool

form2.MdiParent = Me
form2.Show()

Date 16/12/19

PAGE NO.: 309
DATE: / /

ADO.NET
(ActiveX Data Object .Net)

→ It is a module of .Net framework that allows to connect and manipulate the data of database of diversed format. It establises connection between application and data source.

ADO.NET consist of various classes which are used to connect, retreve, insert, update and delete data. All the ADO.NET classes are located in System.Data namespace

There are two main components of ADO.NET:-

(1) Data provider
(2) Dataset

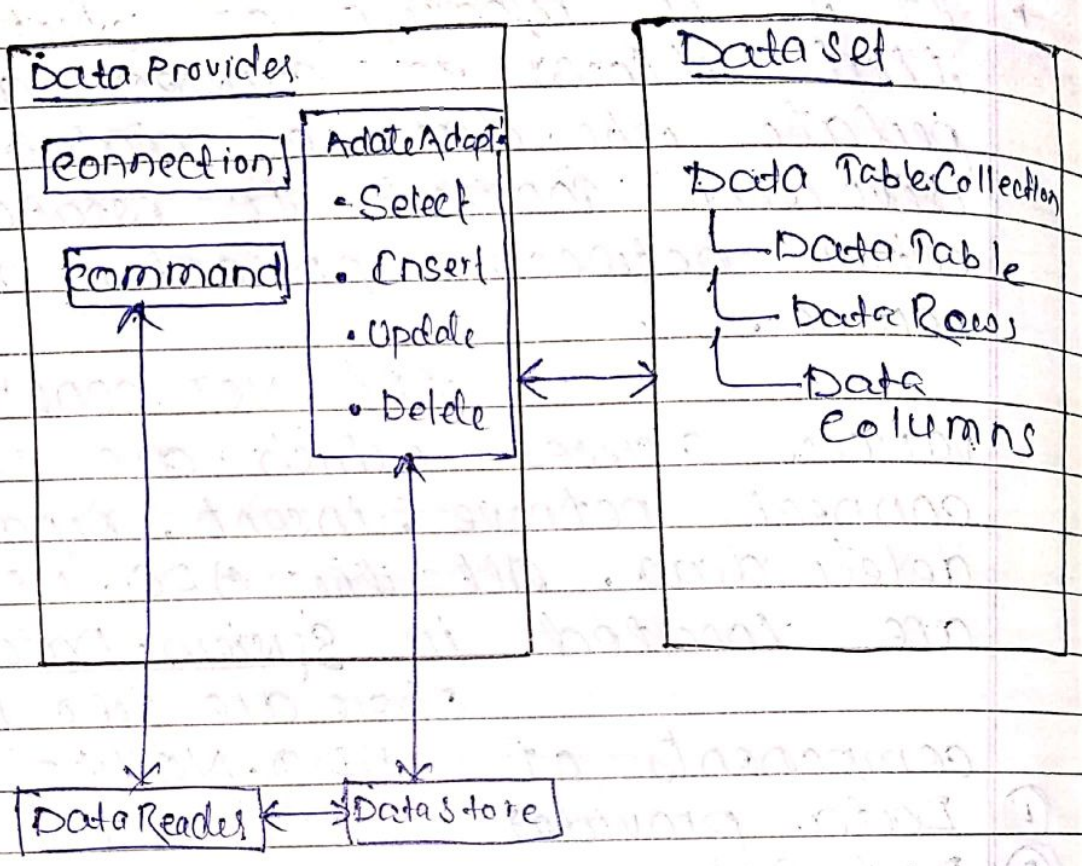
(1) Data provider:- It provides objects such as command, connection, datareader and data adapter to perform database operation.

(2) Data set:- It allows to access data independently from the data source. Dataset contains collection of objects like DataTable, DataRows, DataColumns to access the individual components of data store.

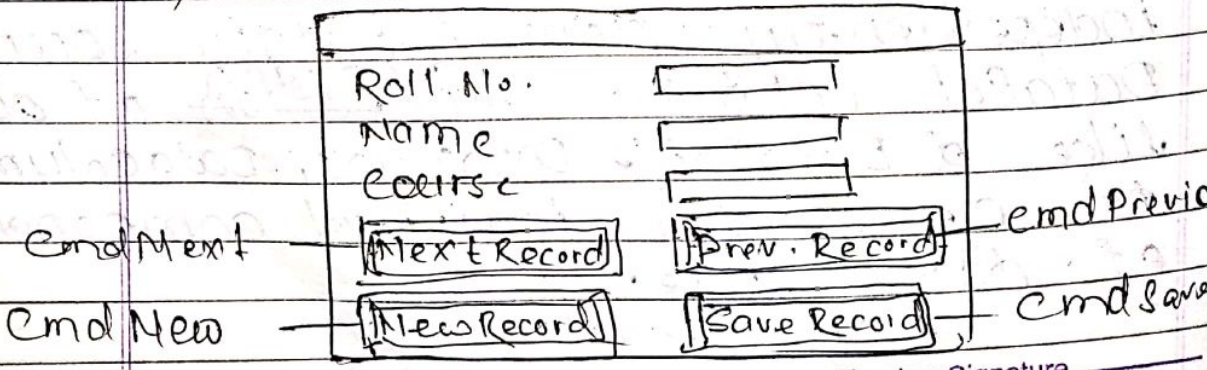
Teacher Signature

PAGE NO.: 310
DATE: / /

Following diagram shows the ADO.NET ~~architecture~~ architecture and its object



Q1- Create a database application in VB.Net to access & manipulate the record of students stored in student table within dotnet.accdb database (MS-Access) The record consists of Roll No, Name & course



Teacher Signature

Pgm:-

```
Imports System
Imports System.Data.OleDb
Public Class Form1
Dim cnn As OleDbConnection
Dim da As OleDbDataAdapter
Dim cmd As OleDbCommand
Dim ds As New Dataset
Dim dt As New DataTable
Dim row As DataRow
Dim i As Integer
Private Sub Form1_Load(ByVal---
 cnn = New OleDbConnection("Provider =
microsoft.ace.oledb.12.0; data
source = c:\81sam\dotnet.accdb")
 cnn.Open()
 da = New OleDbDataAdapter("select *
  from student", cnn)
 da.Fill(ds, "std")
  dt = ds.Tables.Item("std")
  row = dt.Rows(0)
TextBox1.Text = row.Item(0)
TextBox2.Text = row.Item(1)
TextBox3.Text = row.Item(2)
  i = 0

Private Sub cmdNext_Click(ByVal---
  Dim rcount As Integer
   rcount = dt.Rows.count
   i += 1
If i = rcount Then
  MessageBox.Show("Last Record")
```

Teacher Signature

```
        i = rcount - 1
        Exit Sub
    End If
    row = dt.Rows(i)
TextBox1.Text = row.Item(0)
TextBox2.Text = row.Item(1)
TextBox3.Text = row.Item(2)
End Sub

Private Sub cmdPrevious_Click(_
    i = i - 1
    If i < 0 Then
        MessageBox.Show("First Record")
        i = 0
        Exit Sub
    End If
    row = dt.Rows(i)
TextBox1.Text = row.Item(0)
TextBox2.Text = row.Item(1)
TextBox3.Text = row.Item(2)
End Sub

Private Sub cmdNew_Click(_
    TextBox1.Clear()
    TextBox2.Clear()
    TextBox3.Clear()
    TextBox1.Focus()
End Sub

Private Sub cmdsave_Click(_
    Dim str As String
```

```
If cnn.State = ConnectionState.Closed Then
    cnn.Open()
End If
str = "Insert into student values('" & Val
(TextBox1.Text) & "', '" & TextBox2.
Text & "', '" & TextBox3.Text & "')"
cmd = New OleDbCommand()
cmd.Connection = cnn
cmd.CommandText = str
cmd.ExecuteNonQuery()
cnn.Close()
End Sub
End Class
```

date 17/12/19

Exception Handling

An error that occures during the program execution is called "Exception" in other words, run-time error is called exception.

VB.NET provides mechanism to handle exception. Exception handling refers to taking necessary action on occurance of 'Exception'

Following keywords are used to handle exception:-

i) Try:- It contains codes that may generate exception.

ii) catch:- The exception thrown from

Teacher Signature

within Try-block is received in catch-block. It contains code to take action.

iii) Throw :- It used to generate a particular type of exception forcibly.

iv) Finally :- This block contains code that always get executed no matter whether an exception has occured or not.

Syntax

```
Try
  ===
  ===
  ===
  Exit Try
Catch Exception As (exception-type)
  ===
  ===
  Exit Try
Finally
  ===
  ===
End Try
```

Exception-class

In VB.Net that are various pre-defined classes to handle different kind of exceptions. These are derived from System.Exception class.

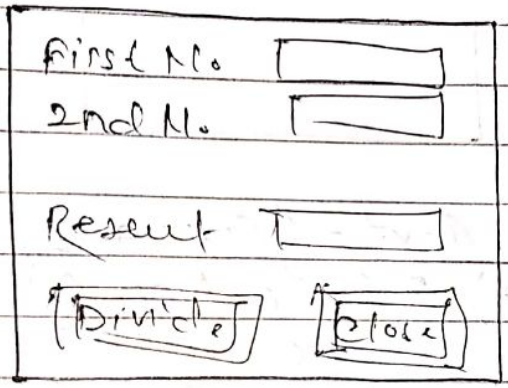
Examples

- System.DivideByZeroException
- System.ArrayIndexOutOfRangeException
- System.ArrayTypeMismatchException
- System.NullReferenceException

Teacher Signature

- System.outofMemoryException.

Q:- ~~Write~~ Create a program in VB.Net to divide no. Handles division by 0 Exception.



Soln:-

```
Public Class Form1
Private Sub Button1_Click(ByVal sender---
  Dim a, b, r As Integer
    a = TextBox1.Text
    b = Textbox2.Text
    Try
      r = a \ b
    TextBox3.Text = r
Catch e1 As System.DivideByZeroException
    MsgBox("SORRY! U CAN NOT DIVIDE A NO BY 0")
Finally
    MsgBox("Thanks")
End Try
End Sub
End Class
```

Teacher Signature_____

Q1:- Create a program in VB.Net to show date & time.

Current date/Time
30-12-19 11:11

Soln:- TextBox1.Text = "Current Date/Time" & Chr(13) & Date.Now()

OR

TextBox1.Text = Date.Now()

- Enter key use for button
  Form → Accept button → Select button
- Textbox Cursor
  Textbox → Tab Index → position select as you wish
- Highlighter button
  Button.Enabled = False/True