2022

School of Computational Sciences

Windows Programming Lab Manual

M.Sc. CS

Dr. Mohnish Mahamune

SUBJECT INDEX

1. Lab Exercise

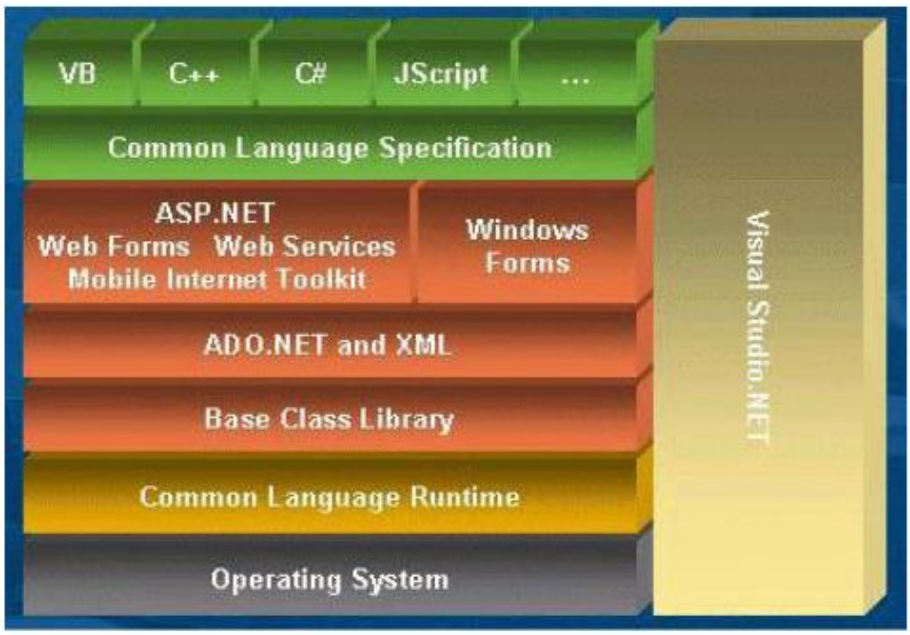
Exercise No1: (2 Hours) - 1 Study Practical

AIM - Introduction to Dot NET Framework.

POINTS TO BE COVERED

- What is Dot net?
- Describing Dot net Framework - describing its core components - Advantages of Dot net
- Dot net languages

.net (network) is a generic top-level domain (gTLD) used on the Internet's Domain Name System. The Microsoft .NET Framework is a software framework that can be installed on computers running Microsoft Windows operating systems. It includes a large library of coded solutions to common programming problems and a virtual machine that manages the execution of programs written specifically for the framework.

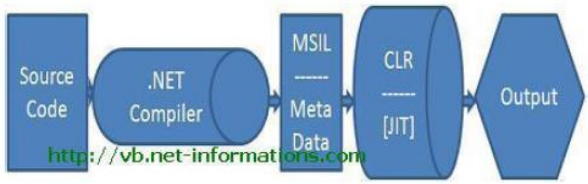


The framework's Base Class Library provides a large range of features including user interface, data and data access, database connectivity, cryptography, web application development, numeric algorithms, and

network
communications. The class library is used by programmers, who combine it with their own code to produce applications. Programs written for the .NET Framework execute in a software environment that manages the program's runtime requirements. Also part of the .NET Framework, this runtime environment is known as the Common Language Runtime (CLR). The CLR provides the appearance of an application virtual machine so that programmers need not consider the capabilities of the specific CPU that will execute the program. The CLR also provides other important services such as security, memory management, and exception handling. The class library and the CLR together constitute the .NET Framework.

How to Microsoft.net framework?

Microsoft .Net Languages Source Code are compiled into ***Microsoft Intermediate Language (MSIL)*** . MSIL we can call it as Intermediate Language (IL) or Common Intermediate Language (CIL). Microsoft Intermediate Language (MSIL) is a CPU independent set of instructions that can be converted to the native code**.** ***Metadata*** also created in the course of compile time with Microsoft Intermediate Language (MSIL) and stored it with the compiled code . Metadata is completely self-describing. Metadata is stored in a file called ***Manifest***, and it contains information about the members, types, references and all the other data that the ***Common Language Runtime (CLR)*** needs for execution.



Dig: Working of VB.NET Framework

The Common Language Runtime (CLR) uses ***metadata*** to locate and load classes, generate native code, provide security, and execute ***Managed Code***. Both ***Microsoft Intermediate Language*** (MSIL) and Metadata assembled together is known as Portable Executable (PE) file. Portable Executable (PE) is supposed to be portable across all 32-bit operating systems by Microsoft ***.Net Framework.***

During the runtime the Common Language Runtime (CLR)'s ***Just In Time*** (JIT) compiler converts the Microsoft Intermediate Language (MSIL) code into native code to the Operating System. The native code is Operating System independent and this code is known as ***Managed Code*** , that is, the language's functionality is managed by the ***.NET Framework*** . The Common Language

Runtime (CLR) provides various Just In Time (JIT) compilers, and each works on a different architecture depends on Operating Systems, that means the same

Microsoft Intermediate Language (MSIL) can be executed on different Operating Systems.

Conclusion: Hence, we have studied architecture as well as framework of .NET.

2. Lab Exercises:

Exercise No 2: (2 Hours) - 1 Practical

AIM - Design a Form to study VB.NET (Basic Concepts).

POINTS TO BE COVERED
- Integrated Development Environment (IDE) •
Module,Procedure,Sub-Procedure,Function • Data Types, Variable Declarations
- Operators, Conversion Functions

Design a form to create calculator application (operators)
-Calculator must possess the following function buttons
-+, _, *, /,log, sqr, cube, sqrt, cube root, mod, %, sin, cos, tan, cancel,1/x, pie

Steps to create windows application

-In Visual Studio, select File|New|Project. Visual studio will display the new project dialog box.

-In the new project dialog box, click the windows Application icon. In the name field, type a project name that describes the program you are binding, such as Demo program. Then, in the location field, type the name of the folder in which you want visual studio to place the projects folder and files. Click ok. Visual studio will display a design window where you can drag and drop controls onto your form.

-To display the toolbox that contains the control you can drag and drop on to your form, select View|Toolbox.Visual studio will open the Toolbox window.

-In the toolbox window, locate the various controls that you can use to design forms.

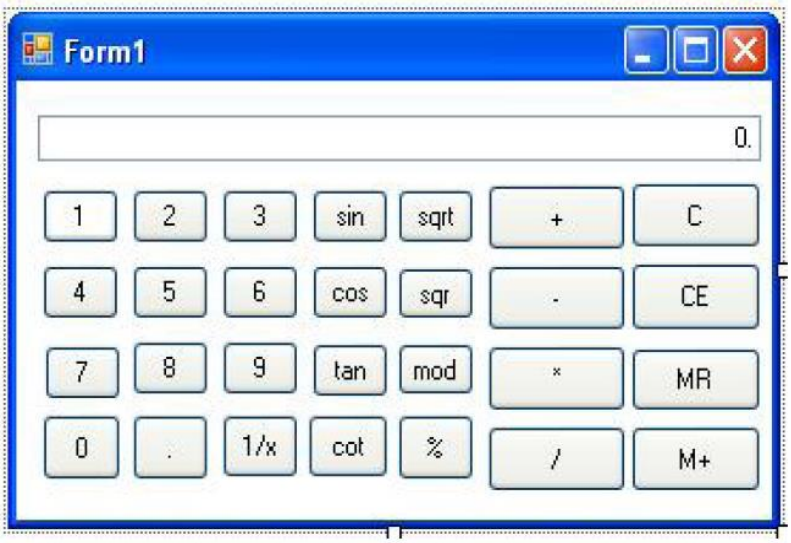
-Through the property window, we can set the properties of Various Controls.

-As mentioned above the calculator must possess the buttons

-Use operators and select case statements to execute the program.

-To run the programs select the debug menu and then press start.

Output



Conclusion: Hence, we have Studied basic concepts of VB.NET and implemented the calculator program in vb.net.

3. Lab Exercises:

Exercise No 3: (2 Hours) - 2 Practical

AIM - Design Forms using different Controls their Properties & Events.

1) Design form to Display Message in Textbox using Button Click Event.
2) Design form to Add & Remove Textbox Items from List box.
3) Design form to Add Combo box & Date Time Picker contents & display them in Listbox.
4) Design form to select image from list and display it in the picture box.

Steps to create windows application

-In Visual Studio, select File|New|Project. Visual studio will display the new project dialog box.

-In the new project dialog box, click the windows Application icon. In the name field, type a project name that describes the program you are binding, such as Demo program. Then, in the location field, type the name of the folder in which you want visual studio to place the projects folder and files. Click ok. Visual studio will display a design window where you can drag and drop controls onto your form.

-To display the toolbox that contains the control you can drag and drop on to your form, select View|Toolbox.Visual studio will open the Toolbox window.

-In the toolbox window, locate the various controls that you can use to design forms.

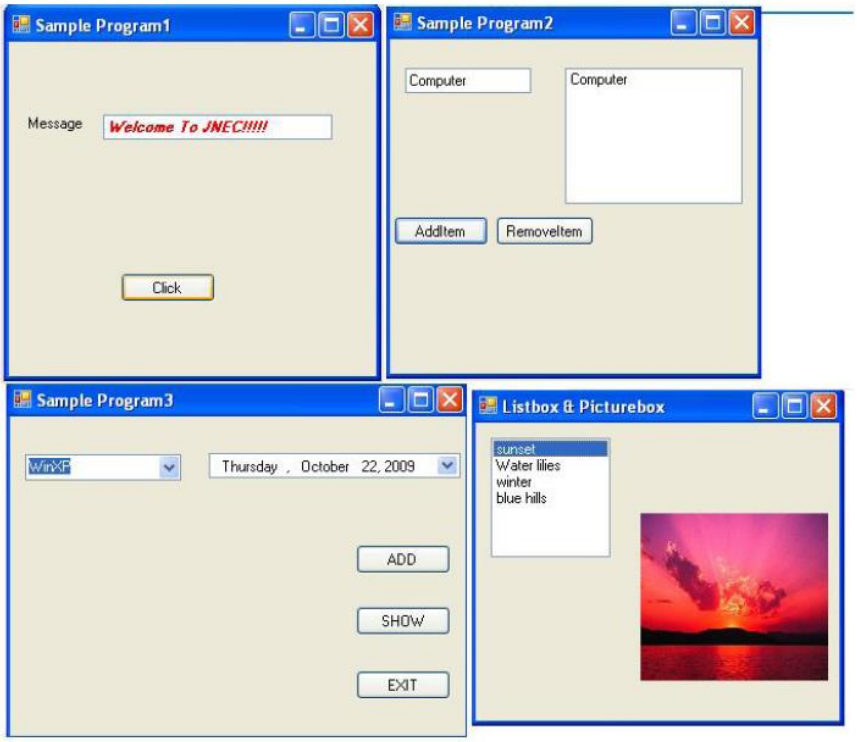
-Through the property window, we can set the properties of Various Controls.

-As mentioned above the calculator must possess the buttons

-Use operators and select case statements to execute the program.

-To run the programs select the debug menu and then press start.

Output:



Conclusion: Hence we have Design Forms using different Controls their Properties and Events

4. Lab Exercises:

Exercise No 4: (2 Hours) - 3 Practical

AIM-Design Forms using different Control Structures in VB.NET.

1) Design a Login form and validate it using IF-ELSE Statement.
2) Design a form to Move Item from one list box to another list box using FOR Loop.
3) Design form to demonstrate WHILE Loop.
4) Design form to demonstrate SLECT CASE Statement.

Steps to create windows application

-In Visual Studio, select File|New|Project. Visual studio will display the new project dialog box.

-In the new project dialog box, click the windows Application icon. In the name field, type a project name that describes the program you are binding, such as Demo program. Then, in the location field, type the name of the folder in which you want visual studio to place the projects folder and files. Click ok. Visual studio will display a design window where you can drag and drop controls onto your form.

- To display the toolbox that contains the control you can drag and drop on to your form, select View|Toolbox.Visual studio will open the Toolbox window.

-In the toolbox window, locate the various controls that you can use to design forms.

-Through the property window, we can set the properties of Various Controls.

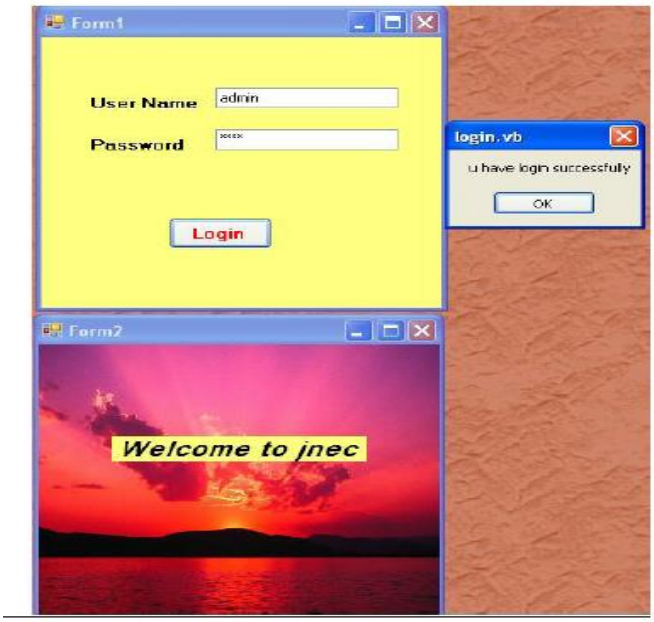
-As mentioned above the calculator must possess the buttons

-Use operators and select case statements to execute the program.

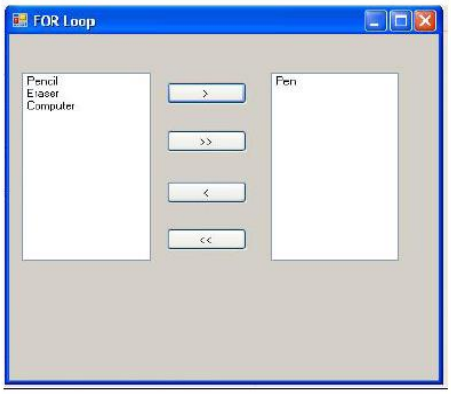
-To run the programs select the debug menu and then press start.

-Your output must look as shown.

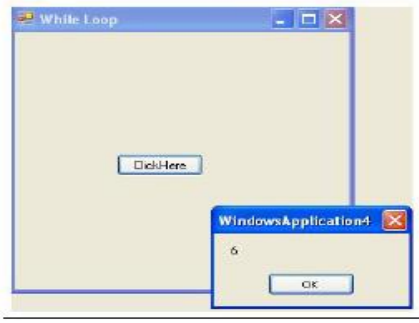
OUTPUT: IF-ELSE Statement



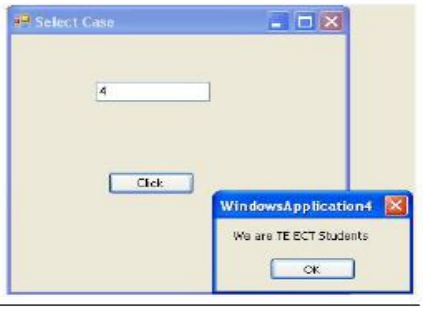
FOR Loop:



While Loop:



Select Case Statement:



Conclusion: Hence we have design Forms using different Control Structures in VB.NET

5. Lab Exercises:

Exercise No 5: (2 Hours) - 4 Practical

AIM-Design Forms to demonstrate An Array Concepts in VB.NET.

- Different Functions related to an Array (UBOUND, LBOUND, GETLENTH ect)

Steps to create windows application

-In Visual Studio, select File|New|Project. Visual studio will display the new project dialog box.

-In the new project dialog box, click the windows Application icon. In the name field, type a project name that describes the program you are binding, such as Demo program. Then, in the location field, type the name of the folder in which you want visual studio to place the projects folder and files. Click ok. Visual studio will display a design window where you can drag and drop controls onto your form.

-To display the toolbox that contains the control you can drag and drop on to your form, select View|Toolbox.Visual studio will open the Toolbox window.

-In the toolbox window, locate the various controls that you can use to design forms.

-Through the property window, we can set the properties of Various Controls.

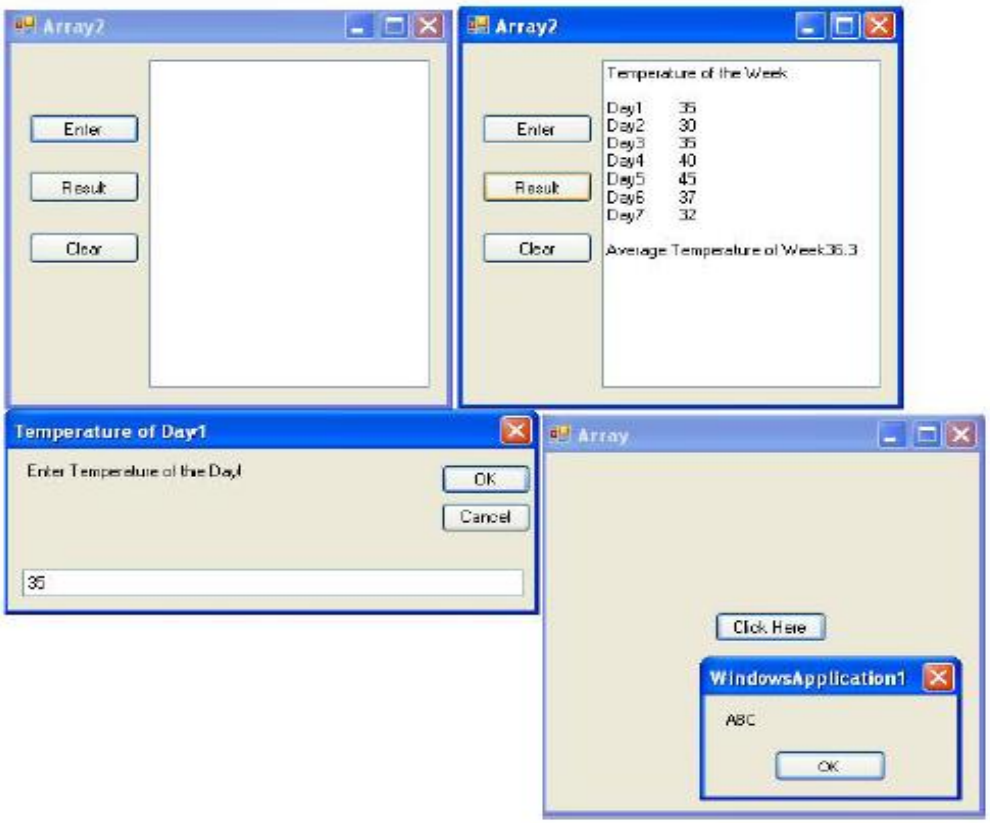
-As mentioned above the calculator must possess the buttons

-Use operators and select case statements to execute the program.

-To run the programs select the debug menu and then press start.

-Your output must look as shown.

OUTPUT:



Conclusion: Hence, we have design forms to demonstrate an Array concept in VB.NET.

6. Lab Exercises:

Exercise No 6: (2 Hours) - 5 Practical

AIM - Design Form using Menus & Dialog boxes.

- Design a form using menus
- Create submenus to open, save files, font, color using open save font and color dialog boxes.

Menus (File, Edit, Format etc in all windows applications) are those that allow us to make a selection when we want to perform some action with the application, for example, to format the text, open a new file, print and so on. In VB .NET ***MainMenu*** is the container for the Menu structure of the form. Menus are made of ***MenuItem*** objects that represent individual parts of a menu (like File->New, Open, Save, Save as etc). The two main classes involved in menu handling are, MainMenu and MenuItem. The MainMenu class let's us assign objects to a form's menu class and MenuItem is the class which supports the items in a menu system. Menus like File, Edit, Format etc and the items in those Menus are supported by this MenuItem class. It's this MenuItem's ***click*** event that makesMenus ***Event of the MenuItem.***

The default event of the MenuItem is the Click event which looks like this in code:

```
Private Sub MenuItem1_Click(ByVal sender As System.Object, ByVal
eAs_System.EventArgs) HandlesMenuItem1.Click

End Sub
```

Creating Menus is simple. Drag a ***MainMenu (Menustrip)*** component from the toolbar onto the form. When you add a MainMenu component to the form it appears in the component tray below the form. Windows form designer will add the MenuItem's for this by default, you need not add this. Once when you finish adding a MainMenu component to the form you will notice a ***"TypeHere"*** box towards the top-left corner of the form. To create a menu all you have to do is click on the ***"TypeHere"*** text which opens up a small textbox allowing you to enter text for the menu. You can view that in the image below. You can use the arrow keys on the keyboard to create a submenu or add other items to that menu or click on the first menu item and use the

left/right arrow keys on the keyboard to create a new menu item. That's all it takes to add a
menu to the form.

Steps to create windows application

-In Visual Studio, select File|New|Project. Visual studio will display the new project dialog box.

-In the new project dialog box, click the windows Application icon. In the name field, type a project name that describes the program you are binding, such as Demo program. Then, in the location field, type the name of the folder in which you want visual studio to place the projects folder and files. Click ok. Visual studio will display a design window where you can drag and drop controls onto your form.

-To display the toolbox that contains the control you can drag and drop on to your form, select View|Toolbox.Visual studio will open the Toolbox window.

-In the toolbox window, drag and drop menustrip, context menu control

-Create submenus like cut, copy, paste, undo.

-Create submenus like new, open, save, save as, font, color, exit etc.

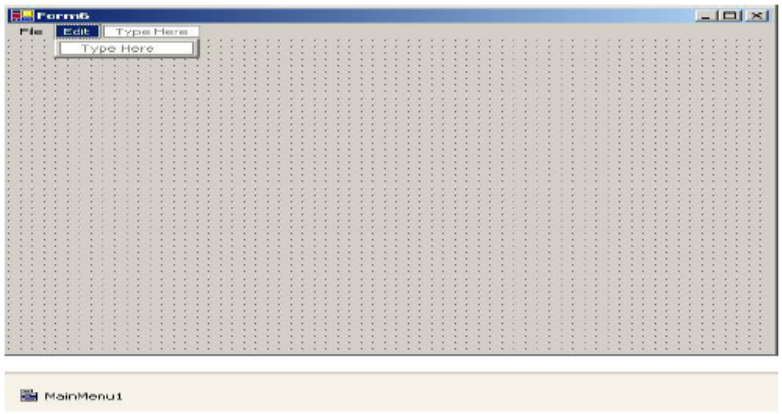
-Drag and drop opendialogbox, savedialogbox, fontdialogbox, colordialogbox on to the form.

-Write the code using various methods to open, save, font, color dialogboxes.

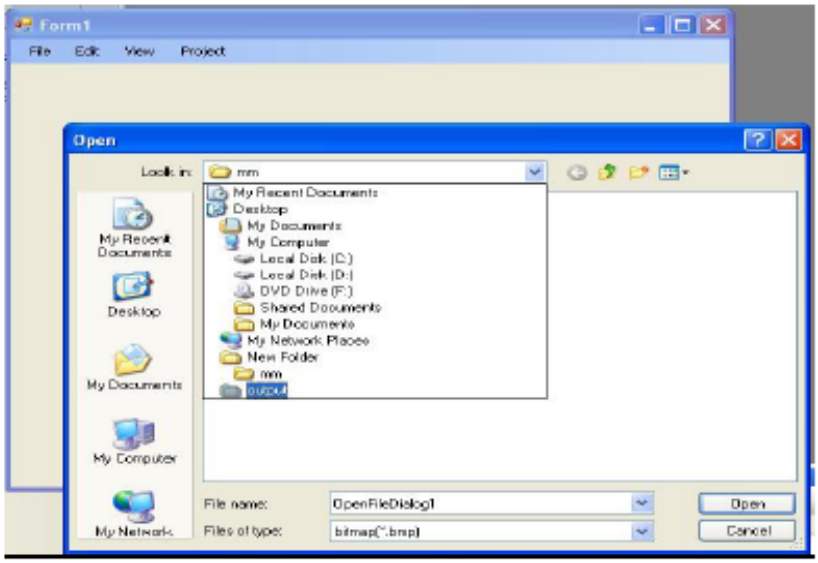
-Select Debug menu and then start to run the program.

-Your program output should look as shown.

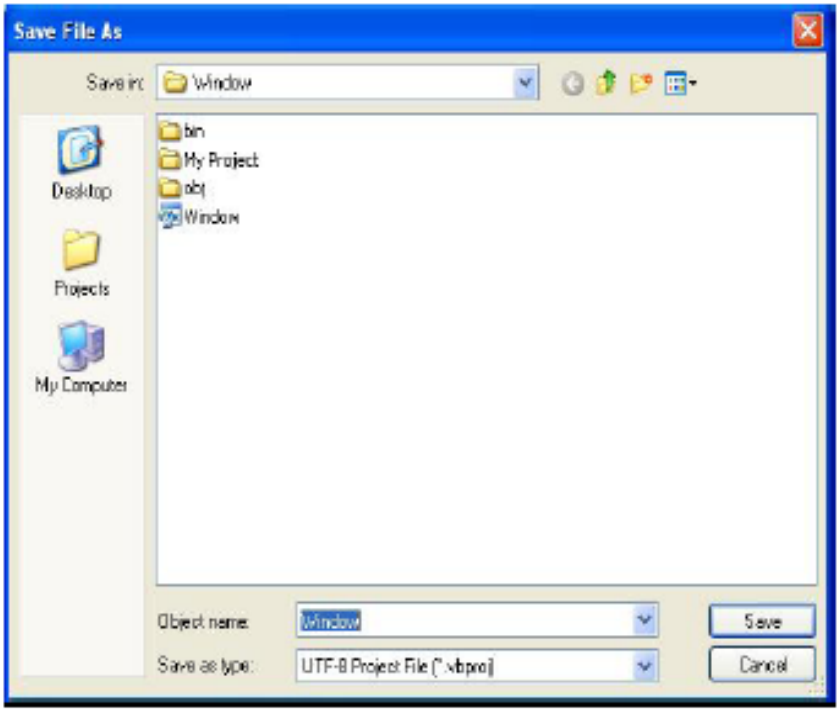
OUTPUT:



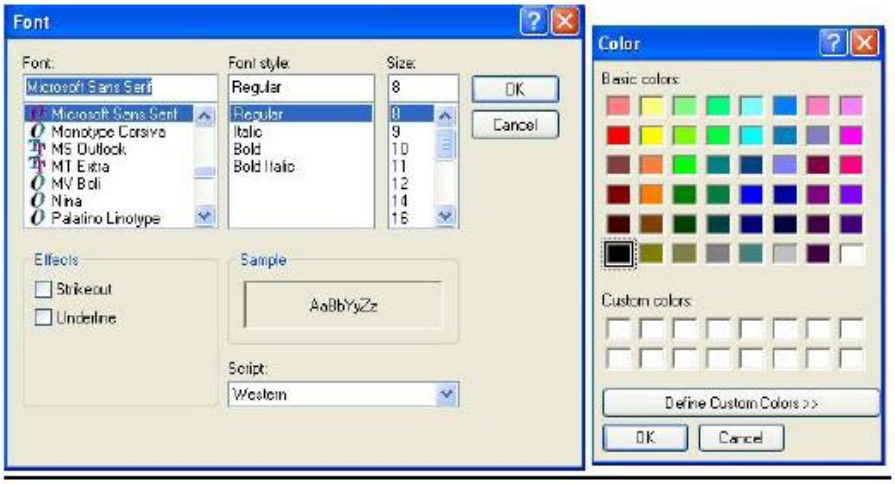
OpenfileDialogbox:



SavefileDialogbox:



FontDialogbox and ColorDialogbox:



CONCLUSION: Hence, we have studied menus using dialog boxes.

7. Lab Exercises:

Exercise No 7: (2 Hours) - 6 Practical

AIM - Object Oriented Programming Concepts in VB.NET.

- Classes & Objects,
- Constructors & Destructors,
- Inheritance,
- Polymorphism.

Coding:

Program for Class & Object

```
Imports System.Console
Module Module1

Sub Main()
Dim obj As New test()
'creating a object obj for test class
obj.disp()
'calling the disp method using obj
Read()

End Sub

End Module
Public Class test
'creating a class name test
Sub disp()
    'a method named disp in the class
                        WriteLine("Welcome to OOP")
 End Sub

End Class
```

Output: Welcome to OOP

Program for Constructor

```
Imports System.Console
Module Module1

Sub Main()
Dim con As New constructor(10)
WriteLine(con.display())
Read()
End Sub

End Module

Public Class constructor
                    Public x As Integer
Public Sub New(ByVal value As Integer)
```

```
                        x   =  value
End  Sub
Public Function display()As Integer
                        Return  x
    End  Function
End  Class
```

Output: 10

Program for Destructor

```
Imports System.Console
Module  Module1
Sub Main()
Dim obj As New destructor()
End Sub
End Module
Public Class destructor
Protected Overrides  Sub  finalize()
                        Write("Hello!!!")

End Sub
End Class
```

Output: Hello!!!

Program for Inheritance

```
Imports System.Console
Module Module1

Sub Main()
Dim ss As New two()
Writeine(ss.sum())
Read()

End  ub

End Module
Public Class one
                    'base class
Public i As  Integer   =   10
Public j As  Integer   =   20
    Public Function add()  As  Integer
    Return  i   +  j
    End  Function
End  Class
Public  Class  two
                    Inherits  one
'derive class
Public k As Integer   =   100
Public Function sum()  As  Integer
                        eturn  i   +  j   +  k
End Function
End Class
```

Output: 130

Program for Polymorphism

```
Imports System.Console
Module Module1

Sub Main()
Dim two As New one()
WriteLine(two.add(10))
'calls the function with one argument
WriteLine(two.add(10, 20))
'calls the function with two arguments
WriteLine(two.add(10, 20, 30))
'calls the function with three arguments

Read()

End Sub

End Module
Public Class one

Public i, j, k As Integer

Public Function add(ByVal i As Integer) As
Integer
                        'function with one
argument
Return i
End Function
Public Function add(ByVal i As Integer, ByVal j As Integer)
As                                                  Integer
                       'function with two arguments
Return i + j
End Function
    Public Function add(ByVal i As Integer, ByVal j As
Integer, ByVal k As Integer) As Integer
'function with three arguments
Return i + j + k
End Function
End Class
```

Output:
10
30
60

8. Lab Exercises:

Exercise No 8: (2 Hours) -2 Study

AIM - Study of ADO.NET

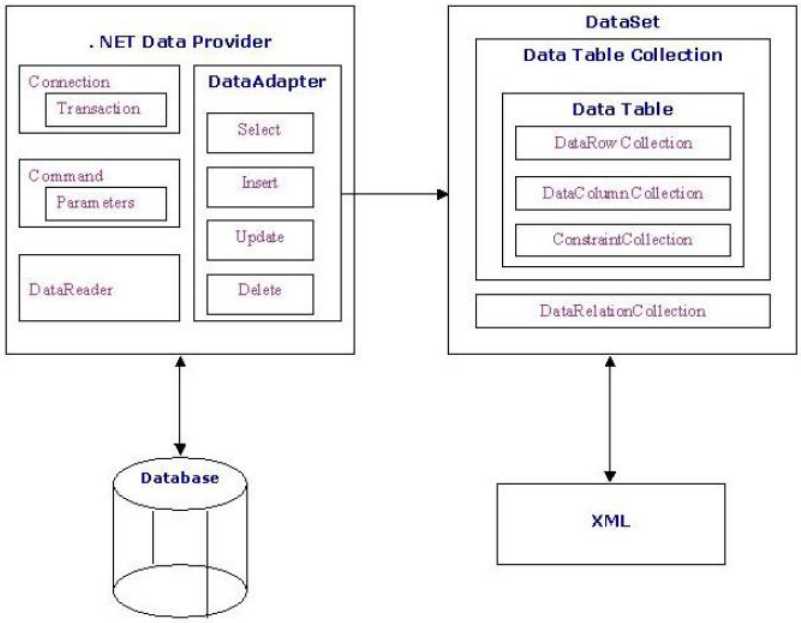
- Explain block diagram
- Describe various components
- Write all the steps to connect to a database.

ADO.NET provides consistent access to data sources such as Microsoft SQL Server, as well as data sources exposed through ***OLEDB*** and ***XML.*** Data-sharing consumer applications can use ***ADO.NET*** to connect to these data sources and retrieve, manipulate, and update data.

ADO.NET cleanly factors data access from data manipulation into discrete components that can be used separately or in tandem. ADO.NET includes ***.NET Framework data providers*** for connecting to a database, executing commands, and retrieving results. Those results are either processed directly, or placed in an ***ADO.NET Dataset*** object in order to be exposed to the user in an ad-hoc manner, combined with data from multiple sources, or remote between tiers. The ADO.NET Dataset object can also be used independently of a .NET Framework data provider to manage data local to the application or sourced from XML.

The ***ADO.NET*** classes are found in ***System.Data.dll***, and are integrated with the XML classes found in ***System.Xml.dll***. When compiling code that uses the ***System.Data*** namespace, reference both ***System.Data.dll and System.Xml.dll.*** For an example of compiling an ***ADO.NET*** application using a command line compiler.

Most applications need data access at one point of time making it a crucial component when working with applications. Data access is making the application interact with a database, where all the data is stored. Different applications have different requirements for database access. VB .NET uses ***ADO .NET (Active X Data Object)*** as its data access and manipulation protocol which also enables us to work with data on the Internet. Let's take a look why ADO .NET came into picture replacing ADO.



ADO .NET Data Architecture

The ADO.NET Data Architecture

Data Access in ADO.NET relies on two components: ***Dataset*** and ***Data Provider***.

Dataset: The dataset is a ***disconnected, in-memory*** representation of data. It

can be considered as a ***local copy*** of the relevant portions of the database. The Dataset is persisted in memory and the data in it can be manipulated and

updated independent of the database. When the use of this Dataset is finished, changes can be made back to the central database for updating. The data in Dataset can be loaded from any valid data source like Microsoft SQL server

database, an Oracle database or from a Microsoft Access database.

Data Provider: The Data Provider is responsible for ***providing*** and ***maintaining***

the connection to the database. A DataProvider is a set of related components that work together to provide data in an efficient and performance driven manner. The .NET Framework currently comes with two DataProviders: the ***SQL Data Provider*** which is designed only to work with Microsoft's SQL Server 7.0 or later and the ***OleDb DataProvider*** which allows us to connect to other

types of databases like Access and Oracle. Each DataProvider consists of the following component classes:

The ***Connection*** object which provides a connection to the database The ***Command*** object which is used to execute a command The ***DataReader*** object which provides a forward-only, read only, connected recordset. The ***Data Adapter*** object which populates a disconnected Dataset with data and performs update

Data access with ADO.NET can be summarized as follows:

A connection object establishes the connection for the application with the database. The command object provides direct execution of the command to the database. If the command returns more than a single value, the command object returns a DataReader to provide the data. Alternatively, the DataAdapter can be used to fill the Dataset object. The database can be updated Using the command object or the DataAdapter.

Component classes that make up the Data Providers

The Connection Object

The Connection object creates the connection to the database. Microsoft Visual Studio .NET provides two types of Connection classes: the ***SqlConnection*** object, which is designed specifically to connect to Microsoft SQL Server 7.0 or later, and the ***OleDbConnection*** object, which can provide connections to a wide range of database types like Microsoft Access and Oracle. The Connection object contains all of the information required to open a connection to the database.

The Command Object

The Command object is represented by two corresponding classes: ***SqlCommand*** and ***OleDbCommand.*** Command objects are used to execute commands to a Database across a data connection. The Command objects can be used to execute stored procedures on the database, SQL commands, or return complete tables directly. Command objects provide three methods that are used to execute commands on the database:

ExecuteNonQuery: Executes commands that have no return values such as INSERT, UPDATE or DELETE

ExecuteScalar: Returns a single value from a database query

ExecuteReader: Returns a result set by way of a DataReader object

The DataReader Object

The DataReader object provides a ***forward-only, read-only, connected stream*** recordset from a database. Unlike other components of the Data Provider, DataReader objects cannot be directly ***instantiated***. Rather, the DataReader is returned as the result of the Command object's ***ExecuteReader*** method. The SqlCommand.ExecuteReader method returns a SqlDataReader object, andthe OleDbCommand.ExecuteReader method returns an OleDbDataReader object. The DataReader can provide rows of data directly to application logic when you do not need to keep the data cached in memory. Because only one row is in memory at a time, the DataReader provides the lowest overhead in terms of system performance but requires the exclusive use of an open Connection object for the lifetime of the DataReader.

The DataAdapter Object

The DataAdapter is the class at the core of ADO .NET's disconnected data access. It is essentially the ***middleman*** facilitating all communication between the database and a DataSet. The DataAdapter is used either to fill a DataTable or DataSet with data from the database with its ***Fill*** method. After the memory- resident data has been manipulated, the DataAdapter can commit the changes to the database by calling the Update method. The DataAdapter provides four properties that represent database commands:

SelectCommand
InsertCommand
DeleteCommand
UpdateCommand

When the Update method is called, changes in the Dataset are copied back to the database and the appropriate Insert Command, Delete Command, or Update Command is executed.

Using OleDb Provider the Objects of the OleDb provider with which we work are:

OleDbConnection: The OleDbConnection class represents a connection to OleDb data source. OleDb connections are used to connect to most databases.

The OleDbCommand Class: The OleDbCommand class represents a SQL

statement or stored procedure that is executed in a database by an OLEDB provider.

OleDbDataAdapter class: The OleDbDataAdapter class acts as a middleman between the datasets and OleDb data source. We use the Select, Insert, Delete and Update commands for loading and updating the data.

OleDbDataReader class: The OleDbDataReader class creates a data reader for use with an OleDb data provider. It is used to read a row of data from the database. The data is read as forward-only stream which means that data is read sequentially, one row after another not allowing you to choose a row you want or going backwards.

Conclusion: Hence we have studied ADO.NET Architecture.

9. Lab Exercises:-

Exercise No 9: (2 Hours) -7practical

AIM- Design a Form to study Data Access in VB.NET.

Design forms in VB.NET and Connect the Forms with MS-Access & Oracle.

- Display table records on the Form
- Insert record into table from the form - Update Existing record.
- Navigate through the records.

Sample Code

Retrieving Records

```
Imports System.Data.OleDb


Public Class Form1
Dim CN As OleDbConnection Dim CMD As OleDbCommand Dim DR As
OleDbDataReader Dim ICOUNT As Integer
Dim STR As String
   Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
Try
   CN=New
OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;DataSource=C:\Document
s and Settings\My Documents\3454.mdb")
'PROVIDER TO BE USED WHEN WORKING WITH ACCESS DATABASE
CN.Open()
CMD=New OleDbCommand("SELECT top 1 * from table1", CN) DR =
CMD.ExecuteReader
While DR.Read()
TextBox1.Text = DR(0)
TextBox2.Text = DR(1)
TextBox3.Text = DR(2)
```

```
    'LOADING DATA INTO TEXTBOXES BY COLUMN INDEX End While

Catch e1 As Exception
    MsgBox(e1.Message.ToString)
End Try
    DR.Close()
    CN.Close()
End Sub
```

Inserting Records

```
 Private Sub Button2_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button2.Click
Try
        CN = New OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=C:\Documents and Settings\jnec\My Documents\3049.mdb")
        CN.Open()
        STR = "INSERT INTO Table1 VALUES(" &
TextBox1.Text & ",'" & TextBox2.Text & "','" & TextBox3.Text & "')"
        'STRING STORES THE COMMAND AND CINT IS USED
TO CONVERT NUMBER TO STRING
    CMD = New
OleDbCommand(STR, CN)
                    ICOUNT =
CMD.ExecuteNonQuery
                    MessageBo
x.Show(ICOUNT)
 Catch e1 As Exception
                    MsgBox(e1
.Message.ToString)
End Try
    CN.Close()
End Sub
```

Updating Records

```
    Private Sub Button3_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button3.Click
Try
        CN = New OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=C:\Documents and Settings\jnec\My Documents\3049.mdb")
        CN.Open()
        STR = "update Table1 set name= '" & TextBox2.Text &
"',class='" & TextBox3.Text & "'where rollno=" & TextBox1.Text & ""
        'STRING STORES THE COMMAND AND CINT IS USED
TO CONVERT NUMBER TO STRING
    CMD = New OleDbCommand(STR, CN)
        ICOUNT = CMD.ExecuteNonQuery
            MessageBox.Show(ICOUNT)
 Catch e1 As Exception
            MsgBox(e1.Message.ToString)
End Try
End Sub

End Class
```