ALAGAPPA UNIVERSITY

[Accredited with 'A+' Grade by NAAC (CGPA:3.64) in the Third Cycle
and Graded as Category–I University by MHRD-UGC]

(A State University Established by the Government of Tamil Nadu)

KARAIKUDI – 630 003

Directorate of Distance Education

B.Sc. [Computer Science]

VI - Semester

130 64

LAB: VISUAL BASIC PROGRAMMING

Author

Dr. Preety Khatri, Assistant Professor-SOIT IMS, Noida

VIKAS® PUBLISHING HOUSE PVT. LTD.
E-28, Sector-8, Noida - 201301 (UP)
Phone: 0120-4078900 • Fax: 0120-4078999
Regd. Office: A-27, 2nd Floor, Mohan Co-operative Industrial Estate, New Delhi 1100 44
• Website: www.vikaspublishing.com • Email: helpline@vikaspublishing.com

Work Order No.AU/DDE/DE12-27/Preparation and Printing of Course Materials/2020 Dated 12.08.2020 Copies - 500

LAB: VISUAL BASIC PROGRAMMING

SYLLABI

1. Building Simple Applications
2. Working with Intrinsic Controls, Control Arrays
3. Application with Multiple Forms
4. Application with Dialogs
5. Application with Menus
6. Application using Data Controls
7. Application using Common Dialogs
8. Drag and Drop Events
9. Database Management
10. Creating ActiveX Controls
11. Database Object (DAO) and Properties
12. Active Data Objects (ADO) and OLE DB
13. Connecting to the Database, Retrieving a Record Set, Creating a Query Dynamically, Using a Parameterized Query, Using Action Queries - Adding Records, Editing Records, Closing the Database Connection
14. Simple Application Development
    - (i) Library Information System
    - (ii) Students Mark Sheet Processing
    - (iii) Telephone Directory Maintenance
    - (iv) Gas Booking and Delivering
    - (v) Electricity Bill Processing
    - (vi) Bank Transaction
    - (vii) Pay Roll Processing
    - (viii) Personal Information System
    - (ix) Question Database and Conducting Quiz
    - (x) Personal Diary

INTRODUCTION

Visual Basic (VB) is a third-generation event-driven programming language from Microsoft known for its Component Object Model (COM) programming model first released in 1991 and declared legacy during 2008. Microsoft intended Visual Basic to be relatively easy to learn and use. Visual Basic was derived from BASIC and enables the Rapid Application Development (RAD) of Graphical User Interface (GUI) applications, access to databases using Data Access Objects, Remote Data Objects, or ActiveX Data Objects, and creation of ActiveX controls and objects.

A programmer can create an application using the components provided by the Visual Basic program itself. Programs written in Visual Basic can also make use of the Windows API, which requires external functions declarations.

This lab manual, *Visual Basic Programming*, contains several programs based on Visual Basic (VB) which includes building simple applications, working with intrinsic controls, control arrays, application with multiple forms, dialogs, menus, application using data controls, common dialogs, drag and drop events, database management, creating ActiveX controls, Database Object (DAO) and properties, Active Data Objects (ADO) and OLE DB, connecting to the database, retrieving a record set, creating a query dynamically, parameterized query, action queries, simple application development, such as library information system, students mark sheet processing, telephone directory maintenance, gas booking and delivering, electricity bill processing, bank transaction, pay roll processing, personal information system, etc.

In addition, it will help students in coding and debugging their Visual Basic (VB) programs. The manual provides all logical, mathematical and conceptual programs that can help to write programs easily. These exercises shall be taken as the base reference during lab activities for students.

BLOCK 1

NOTES

This block will cover the following topics:

1. Introduction of VB and building simple applications.
2. Create, save and open the project.
3. Work with intrinsic controls and control arrays.

Visual Basic

Visual Basic (or VB) is a programming language that runs on the .NET framework and developed by Microsoft. It is a third generation event-driven programming language known for its Component Object Model (COM) programming model. It can be used to build Windows applications, web applications and Windows phone applications. Programs in VB will only run on a Windows operating system. It is easy to learn and powerful.

Building Simple Applications

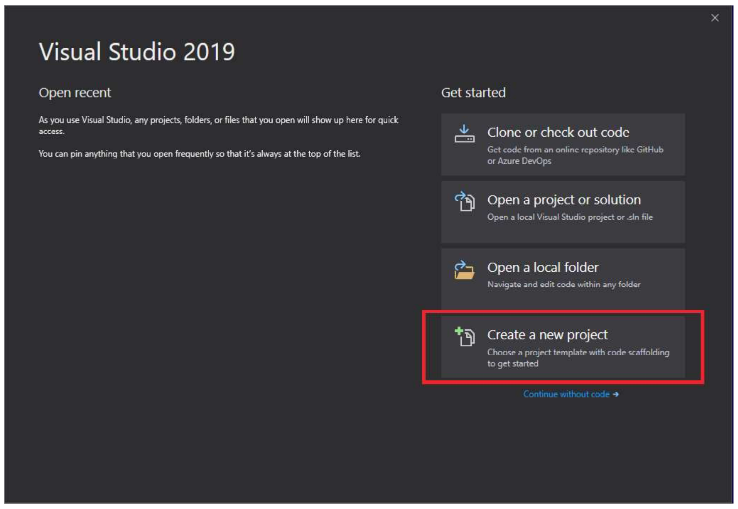
Following are the steps for building new application in Visual Basic.

Step 1: Download Visual Basic

You can download Visual Basic from Microsoft .NET (Visual Studio).

Step 2: Creating your New Project

Choose Standard EXE to enter VB integrated development environment in the New Project Dialog. In the VB IDE, a default form with the name Form1 will appear. Next, double click on Form1 to bring up the source code window for Form1, as shown in screenshot given below.



Visual Studio 2019

Open recent

As you use Visual Studio, any projects, folders, or files that you open will show up here for quick access.

You can pin anything that you open frequently so that it's always at the top of the list.

Get started

Clone or check out code
Get code from an online repository like GitHub or Azure DevOps

Open a project or solution
Open a local Visual Studio project or .sln file

Open a local folder
Navigate and edit code within any folder

Create a new project
Choose a project template with code scaffolding to get started

Continue without code →

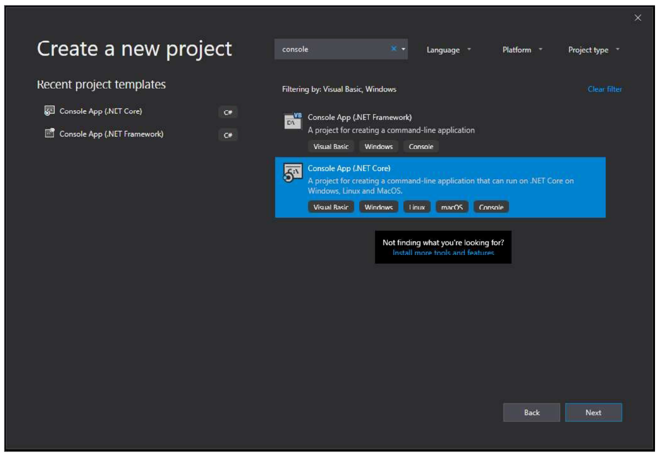
Now, follow the steps given below:

1. Open Visual Studio.
2. Choose **Create a new project** on the start Window.

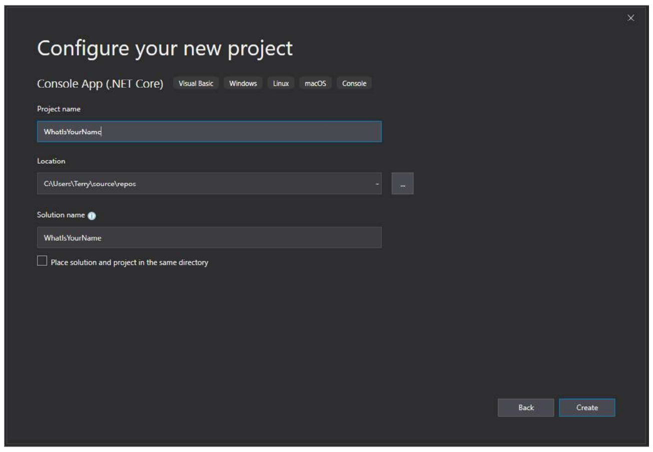
NOTES

3. Enter or type console in the search box on the Create a new project Window. After that select Visual Basic from the Language list, and then choose Windows from the Platform list.

After selecting language and platform filters, choose the Console App (.NET Core) template, and then select Next.


Create a new project
console
Language
Platform
Project type
Recent project templates
Console App (.NET Core)
C#
Console App (.NET Framework)
C#
Filtering by: Visual Basic, Windows
Clear filter
Console App (.NET Framework)
A project for creating a command-line application
Visual Basic
Windows
Console
Console App (.NET Core)
A project for creating a command-line application that can run on .NET Core on Windows, Linux and MacOS.
Visual Basic
Windows
Linux
macOS
Console
Not finding what you're looking for?
Install more tools and features
Back
Next

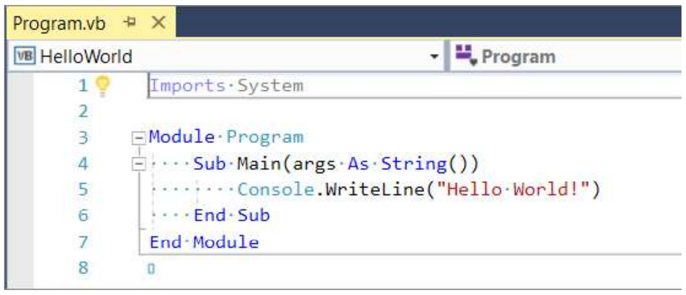
Type or enter WhatIsYourName in the Project name box in the Configure your new project window and then select Create.



Project name: WhatIsYourName
Location: C:\Users\Terry\source\repos
Solution name: WhatIsYourName
☐ Place solution and project in the same directory

Step 3: Creating Your First Application

Visual Studio creates a simple "Hello World" application for you on selecting Visual Basic project template and name of project. **WriteLine** method is called to display the literal string "**Hello World**!" in the console window.



```
Imports System

Module Program
    Sub Main(args As String())
        Console.WriteLine("Hello World!")
    End Sub
End Module
```

Now, add some code to pause the application and requesting for the user input.

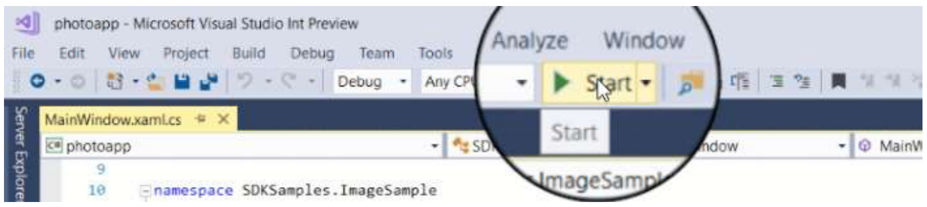
```
Console.Write("Press any key to continue...")
Console.ReadKey(true)
```

Note: Select Build → Build Solution on the menu bar.

It will compile the program in intermediate language (IL) that is converted by Just-In-Time (JIT) compiler into binary code.

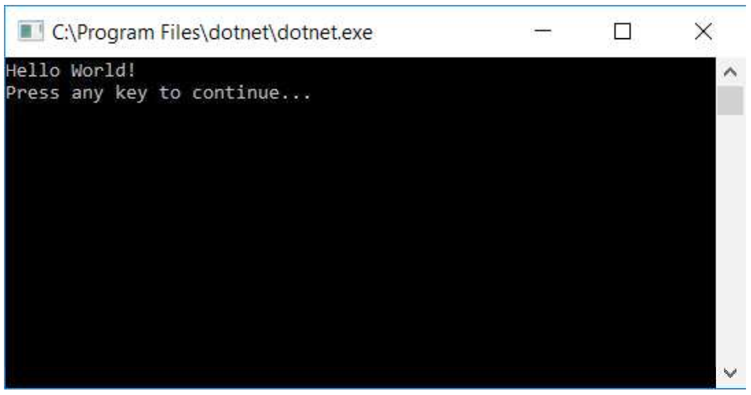
Step 4: Save and Test

Run the program in Debug mode.


photoapp - Microsoft Visual Studio Int Preview
File Edit View Project Build Debug Team Tools
Analyze Window
Debug
Any CPU
Start
Start
Server Explorer
MainWindow.xaml.cs
photoapp
ImageSampl

```
namespace SDKSamples.ImageSample
```

Press any key to close the console window.



```
Hello World!
Press any key to continue...
```

Working with Intrinsic Controls

Intrinsic controls are the basic set of twenty controls in the Toolbox. These controls exist within the Visual Basic .exe file. Intrinsic controls do not have to add to Toolbox. They can not be removed from the Toolbox. They are available during the use of VB and you can access them from the Toolbox and lists the intrinsic controls during design time.

NOTES

Table 1.1 *Intrinsic Controls and their Description*

S. NO.	Control	Prefix	Description
1	**Label**	lbl	Displays text on a form
2	**Frame**	fra	Serves as a container for other controls
3	**CheckBox**	chk	Enables users to select or deselect an option
4	**ComboBox**	Cbo	Allows users to select from a list of items or add a new value
5	**HscrollBar**	hsb	Allows users to scroll horizontally through a list of data in another control
6	**Timer**	tmr	Lets your program perform actions in real time, without user interaction
7	**DirListBox**	dir	Enables users to select a directory or folder
8	**Shape**	shp	Displays a shape on a form
9	**Image**	img	Displays graphics (images) on a form but can't be a container
10	**OLE Container**	ole	Enables you to add the functionality of another Control program to your program
11	**PictureBox**	pic	Displays graphics (images) on a form and can serve as a container
12	**TextBox**	txt	Can be used to display text but also enables users to enter or edit new or existing text
13	**CommandButton**	cmd	Enables users to initiate actions
14	**OptionButton**	opt	Lets users select one choice from a group; must be used in groups of two or more
15	**ListBox**	lst	Enables users to select from a list of items
16	**VscrollBar**	vsb	Enables users to scroll vertically through a list of data in another control
17	**DriveListBox**	drv	Lets users select a disk drive
18	**FileListBox**	fil	Lets users select a file
19	**Line**	lin	Displays a line on a form
20	**Data**	dat	Lets your program connect to a database

Control Arrays

A control array is a group of controls having the same name type and event procedures. Control arrays uses fewer resources in comparison to adding multiple control of same type at design time. They can be created at design time only. You can create the control array using the any of the three methods given below.

1. You can create a control array with only one element using a control and assigning that a numeric, non-negative value to its Index property.
2. You create two controls of the same class with an identical Name property. VB display a dialog box warning that there is already a control having same name and asks to create another control array. Click on the Yes button.
3. Select a control on the form and copy it to the clipboard, and paste a new instance of the control having the same Name property as the original one. Visual Basic shows the warning as in method second.

Control arrays are one of the most interesting features of the VB that adds flexibility to your programs. All controls in a control array have the same set of event procedures that result in reduced amount of code you have to write to respond to a user's actions. Consider an example, if you have a control array of 10 textboxes call txtField, indexed 0 to 9, and then you can use one GotFocus event among all the 10 members instead of using 10 different GotFocus events. VB will automatically pass an Index parameter to the event procedure to differentiate which member of the control array is being acted upon. The code of GotFocus event procedure for the txtField control array might look as given below:

```
Private Sub txtField_GotFocus(Index As Integer)
txtField(Index).SelStart = 0
```

```
 txtField(Index).SelLength = Len(txtField(Index).Text)
End Sub
```

Or

```
Private Sub txtField_GotFocus(Index As Integer)
With txtField(Index)
.SelStart = 0
.SelLength = Len(.Text)
End With
End Sub
```

NOTES

In Visual Basic 6, the importance of using control arrays as a means of dynamically creating new controls at run time is reduced. It has introduced a new and more powerful capability.

- Syntax for refering to a member of a control array is:

  `ControlName(Index)[.Property]`

- For events where VB already passes a parameter (for example, the textbox's KeyPress event where VB passes the KeyAscii parameter), VB will add "Index" as the first parameter, followed by the parameters that are usually passed to the event. The syntax for procedure header of the KeyPress event of the txtField control array will be:

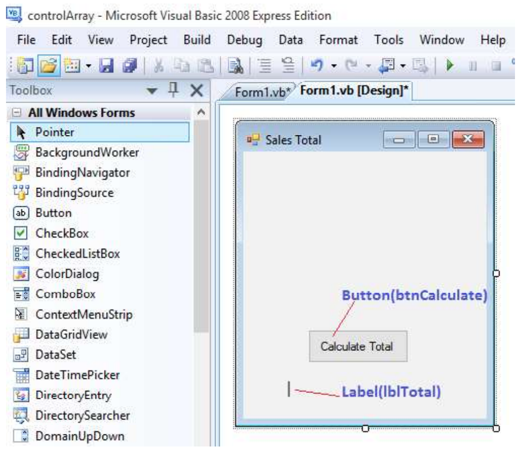
  ```
  Private Sub txtField_KeyPress(Index As Integer, KeyAscii
  As Integer)
  ```

Program 1.1: Create a form to enter total sales for five years in textboxes. All the sales values of 5 years are added and displayed in the label under the button on clicking Calculate button.

The labels and textboxes for reading sales values are created as Control Arrays of labels and textboxes respectively.

Design of Data Entry Form

Calculate Button(btnCalculate) and Label (lblTotal) for displaying the total sales of 5 years is in the design of the form.



NOTES

Data Entry Form at Runtime

Control arrays for Labels and Textboxes are declared when the form is executed. These control arrays elements are added and displayed on the form in the form_load event. When a user fills the values in textboxes and clicks the Calculate Button then a for loop is used to get values from textbox control array and add them together in a variable total. After completing loop label lblTotal's text property is set with variable total's value.

```
Public Class Form1
 Dim lblValues(4) As Label 'control array of labels
 Dim txtValues(4) As TextBox 'control array of textboxes
 Dim intCount As Integer
 Private Sub Form1_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
 For intCount = 0 To 4
```

' Delcare text box and label variable that are added as elements of control arrays

```
 Dim MyTextbox As New TextBox
 Dim MyLabel As New Label
```

'Set left and top values of labels to define display position in the form

```
    MyLabel.Left = 10
    MyLabel.Top = 10 + 25 * intCount
     'set the text of labels
    MyLabel.Text = "Year " + CStr(intCount + 1)
     'set width of the labels
    MyLabel.Width = 50
```

'add newly created label as an element at the current index defined by intCount of the label control array

```
   lblValues(intCount) = MyLabel
```

'add the label control array element on the form using controls collection

```
   Me.Controls.Add(lblValues(intCount))
```

'Set left and top values of textboxes to define display position in the form

```
   MyTextbox.Left = 100
    MyTextbox.Top = 10 + 25 * intCount
```

'add newly created textbox as an element at the current index defined by intCount of the textbox control array

```
    txtValues(intCount) = MyTextbox
```

'add the textbox control array element on the form using controls collection

```
   Me.Controls.Add(txtValues(intCount))
    Next
    End Sub
    Private Sub btnCalculate_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnCalculate.Click
```

Dim total as Double ' variable to store sum of sales while iterating through control array of textboxes

```
    For intCount = 0 To 4
```

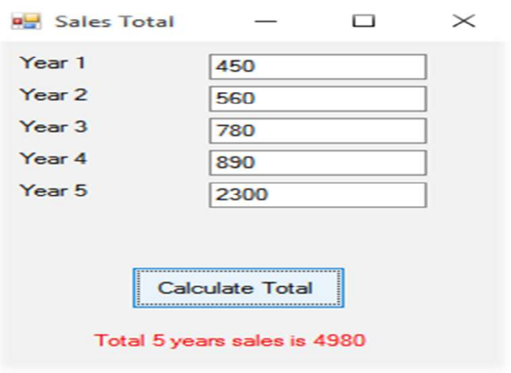
'Add values from textboxes to variable total using += assignment operator

```
     total += Val(txtValues(intCount).Text)
     Next
     'change fore color of display label
     lblTotal.ForeColor = Color.Red
```

'display total in label at form bottom

```
    lblTotal.Text = "Total 5 years sales is " + CStr(total)
     End Sub
    End Class
```

Output:



```
450
560
780
890
2300
Total 5 years sales is 4980
```

BLOCK 2

This block will cover the following topics:

1. Working with forms and dialogs.
2. Working with menus, data controls and common dialogs.

Application with Multiple Forms

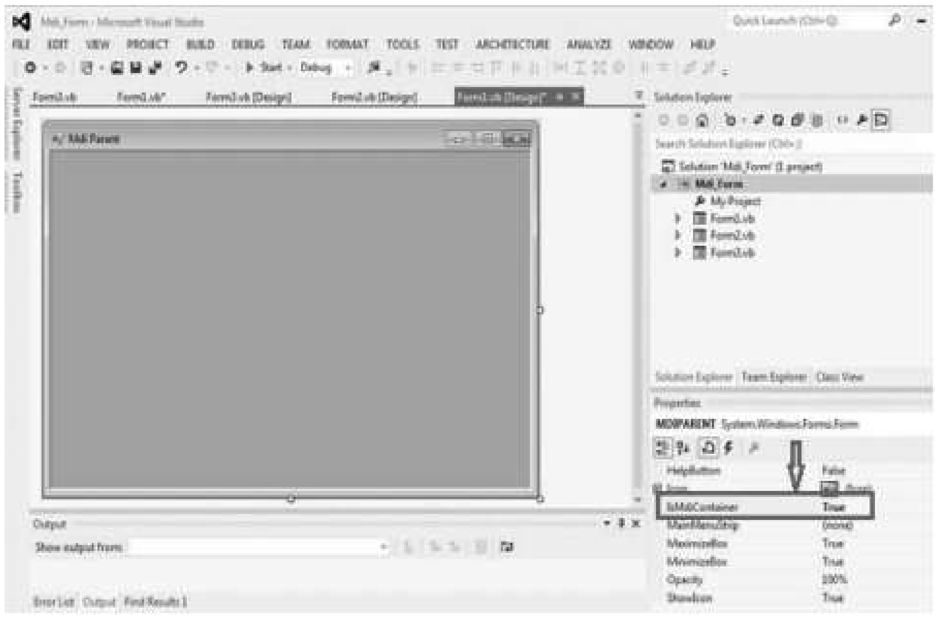
You can open one form from another one in two ways. As a modal form, if you open a second form, you can't alter the emphasis of the second form until you close it. That kind of shape is expressed by a message box. For applications that involve an answer to the second form before behavior on other forms make sense, this behaviour is convenient. Non-modally is another way in which you can open one shape from the other.In this second style, without closing the second form before returning to the first form, a user may switch back and forth between the

NOTES

forms. This kind of conduct is illustrated by the spreadsheet windows in a Microsoft Excel file. Non-modal opening forms are ideal for applications where users need the freedom to search two or more forms in any order.

Program 2.1 Write a program to demonstrate the implementation of MDI forms.

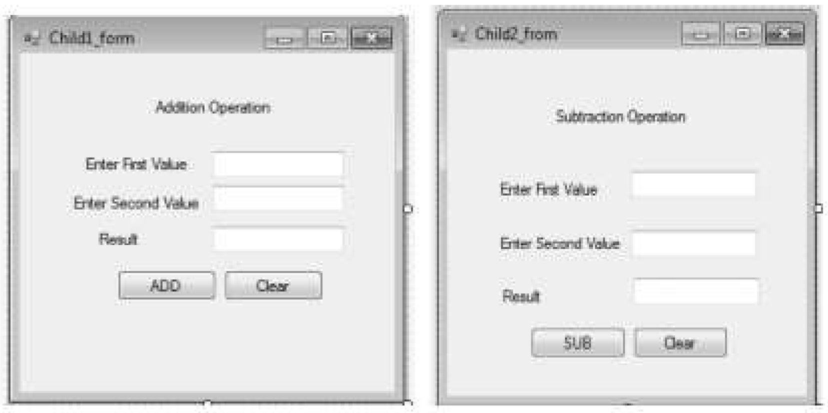
Step 1: Create new Windows forms application and name it as "Mdi_Form". A default Windows form will be displayed. Click on the Form Header and visit its properties window. From properties Window change property "IsMdiContainer to True"as shown below:


Mdi_Form - Microsoft Visual Studio
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM FORMAT TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP
Start Debug
Form1.vb [Design]
Mdi Parent
Solution Explorer
Solution 'Mdi_Form' (1 project)
Mdi_Form
My Project
Form1.vb
Form2.vb
Form3.vb
Solution Explorer Team Explorer Class View
Output
Show output from:
Error List Output Find Results 1

Properties
MDIPARENT System.Windows.Forms.Form

HelpButton	False
IsMdiContainer	True
MainMenuStrip	(none)
MaximizeBox	True
MinimizeBox	True
Opacity	100%
ShowIcon	True

Step 2: Right click on Mdi_From in solution explorer and add two forms by clicking over ADD option therein and Name them Form2 and Form3 respectively. Customize these Form2(Child1_form) and Form3(Child2_form) as shown below to make them to perform as two separate functions.



Step 3: Customise each control pasted on the Form2 and Form3 as per the functions specified on labeled as show above. After customizing these form2. vb and Form3.vb, the code behind them will be as shown below:

NOTES

Form2.vb

```
    'From2.vb
    'Program to demonestrate the use of MDI Form to perform
Addition operation
    Public Class Child1_form
    Private Sub Label1_Click(sender As Object, e As EventArgs)
Handles Label1.Click
    End Sub
    Private Sub Button1_Click(sender As Object, e As EventArgs)
Handles Button1.Click
    Dim sum As Integer
    sum  =  Convert.ToInt32(TextBox1.Text)  +  Convert.
ToInt32(TextBox2.Text)
    TextBox3.Text = sum
    End Sub
    Private Sub Button2_Click(sender As Object, e As EventArgs)
Handles Button2.Click
    TextBox1.Text = " "
    TextBox2.Text = " "
    TextBox3.Text = " "
    End Sub
    End Class
```

Form3.vb

```
    'From3.vb
    'Program to demonestrate the use of MDI Form to perfrom
Subtration Operation
    Public Class Child_form2
    Private Sub Button1_Click(sender As Object, e As EventArgs)
Handles Button1.Click
    Dim subt As Integer subt = Convert.ToInt32(TextBox2.Text)
- Convert. ToInt32(TextBox1.Text) TextBox3.Text = subt
    End Sub
    Private Sub Button2_Click(sender As Object, e As EventArgs)
Handles Button2.Click
    TextBox1.Text = " "
    TextBox2.Text = " "
    TextBox3.Text = " "
    End Sub
    End Class
```

Application with Dialogs

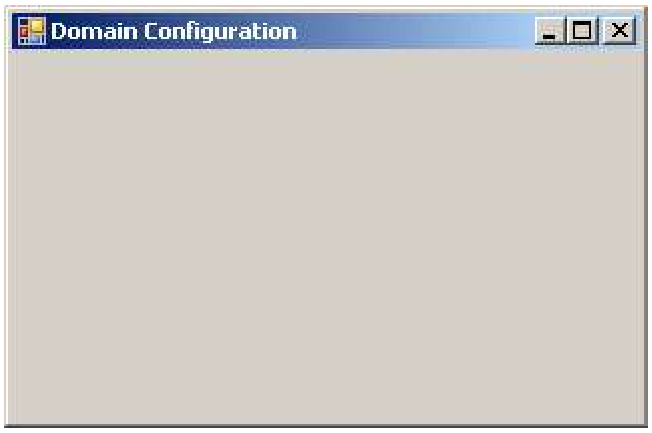
Dialog boxes are used for user interaction and information retrieval. In simple terms, a dialog box is a shape that is set to FixedDialog with its FormBorderStyle enumeration property. You can make your own custom dialog boxes using Visual Studio's Windows Forms Designer.Add controls such as Textbox, Label, and Button to customize dialog boxes as per your needs. The .NET Framework includes predefined dialog boxes, such as File Open and message boxes that can be altered for your own applications.

Dialog Box Creation

You have to start with a Form to create a dialog box, which can be obtained by creating a Windows application.

```
Imports System.Drawing
Imports System.Windows.Forms
Module Exercise
 Public Class Starter
 Inherits Form
 Dim components As System.ComponentModel.Container
 Public Sub New()
 InitializeComponent()
 End Sub
 Public Sub InitializeComponent()
 Text = "Domain Configuration"
 Width = 320
 Height = 150
 Location = New Point(140, 100)
 StartPosition = FormStartPosition.CenterScreen
 End Sub
 End Class
 Function Main() As Integer
 Dim frmStart As Starter = New Starter
 Application.Run(frmStart)
 Return 0
 End Function
End Module
```

Output:



NOTES

Types of Dialog Boxes

There are three types of dialog boxes which are given below:

1. **Modal:** They are typically used to display messages and to set program parameters. Modal dialogs come to the front of the computer, and when the modal dialog box is open, you can not use the software. The modal dialog box must be closed in order to continue using the software.
2. **System modal:** System modal dialog boxes, except that they supersede the entire desktop area, are like modal boxes. Nothing else on the computer can be tapped or picked while a device modal dialog box is open.
3. **Modeless**: Modeless dialog boxes are a different type of color, and are more like windows than dialog boxes. First, in order to ensure that dialog box messages are routed correctly, we need to change the message loop, for example:

```
while(GetMessage(&msg, NULL, 0, 0))
{
 if(!IsDialogMessage(hDlg, &msg))
  {
 TranslateMessage(&msg);
 DispatchMessage(&msg);
  }
}
```

VB.NET Dialog Box

A dialog box is a temporary window for an application that recognizes a user's mouse or keyboard response to open a file, save a file, warning updates, color, print, open a file dialog, etc. It is also beneficial to establish interaction and contact between the user and the application. In addition, when the application has to communicate with users, the dialog box appears in a type, such as when an error occurs, a warning message, a user's acknowledgment or when the program requires urgent action or if the decision is to be saved depending on the changes.

The All VB.NET dialog box inherits the CommonDialog class and overrides the base class's RunDialog() method to build the OpenFileDialog, PrintDialog, Color, and Font dialog boxes. RunDialog() method is automatically called in Windows type, when the dialog box calls its ShowDialog() function. Following functions of the **ShowDialog()** method can be called during run time in the Windows Form.

- **Abort**: It is used when a user clicks on the Abort button to return the **DialogResult.Abort** value.
- **Ignore**: It is used when a user clicks on the Ignore button to return the **DialogResult.Ignore**.
- **None**: Returns nothing, when the user clicks on the None button, and the dialog box is continued running.
- **OK**: It returns a **DialogResult.OK,** when the user clicks the **OK** button.
- **Cancel**: It returns **DialogResult.Cancel**, when the user clicks the **Cancel** button.
- **Yes**: It returns **DialogResult.Yes**.
- **Retry**: It returns a **DialogResult.Retry**,
- **No**: It returns **DialogResult.No**,

Commonly used dialog box controls in the VB.NET Windows form are as follows.

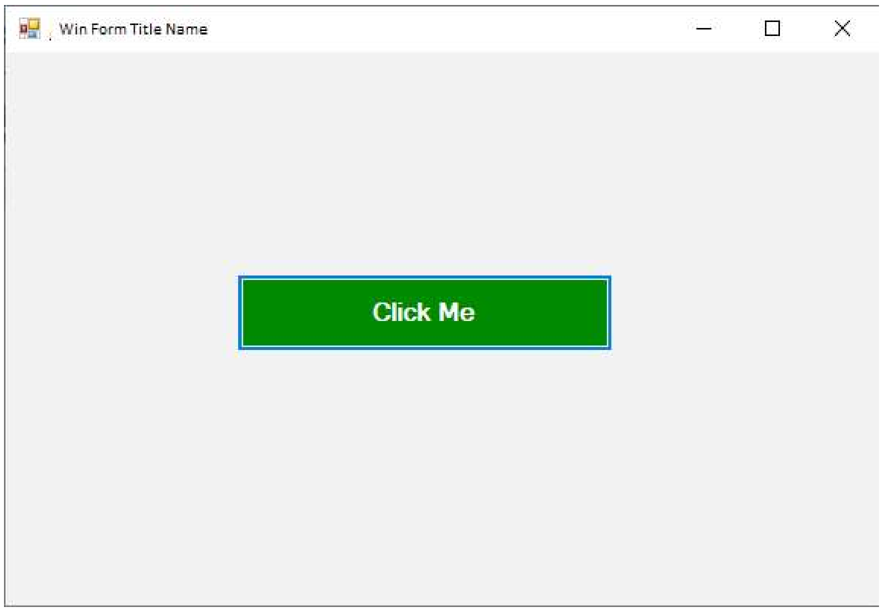
1. **Color Dialog Box**: It is used to display a color dialog that allows the user to choose a color from a predefined color or to specify a custom color.
2. **Font DialogBox**: It allows the user to choose the font, font size, color, and style to be added to the current text range.
3. **OpenFile Dialog Box**: It is used to build a prompt box that allows users to pick a file that they want to open and allows several files to be selected.
4. **Save File Dialog Box:** It prompts the user to choose a file saving location and allows the user to decide the name of the data saving file.
5. **Print Dialog Box**: This is used to create a print dialog that allows the user to print documents by selecting the printer and setting the page to be printed through the Windows application.

A sample code for creating a dialog box is given below:

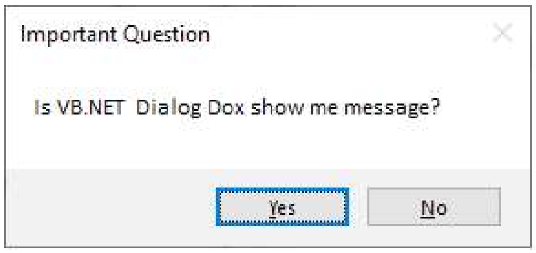
```
Public Class Dialog
 Private Sub Dialog_Load(sender As Object, e As EventArgs)
Handles MyBase.Load
```

```
      Button1.Text = "Click Me" 'Set the name of button
      Me.Text = "Win Form Title Name" ' Set the title name for
the Windows Form
      Button1.BackColor = Color.Green ' Background color of
the button
      End Sub
     Private Sub Button1_Click(sender As Object, e As EventArgs)
Handles Button1.Click
      Dim result1 As DialogResult = MessageBox.Show("Is VB.NET
Dialog Dox show me message?",
      "Important Question",
      MessageBoxButtons.YesNo)
      End Sub
     End Class
```

Compile and Run



Now, click on the Click Me button of the Windows Form, it displays the dialog box, as shown below.


Important Question

Is VB.NET Dialog Dox show me message?

NOTES

Applications with Menus

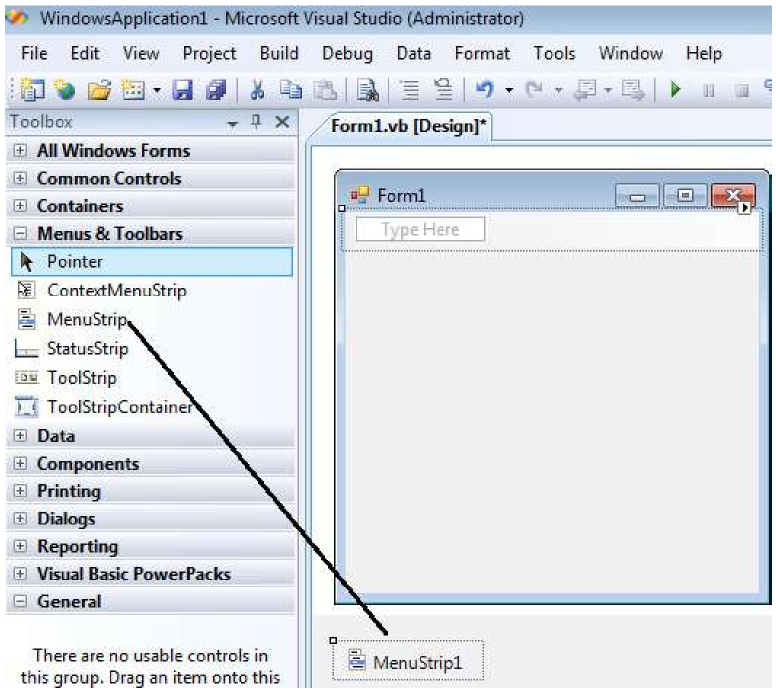
In VB6, the MainMenu feature was not present; it was added to VB.NET. But, as in previous versions of VB, you can use the methods and properties of the Menu portion to add and change menu items in either design-time or run-time.

Working with Menus in Design-Time

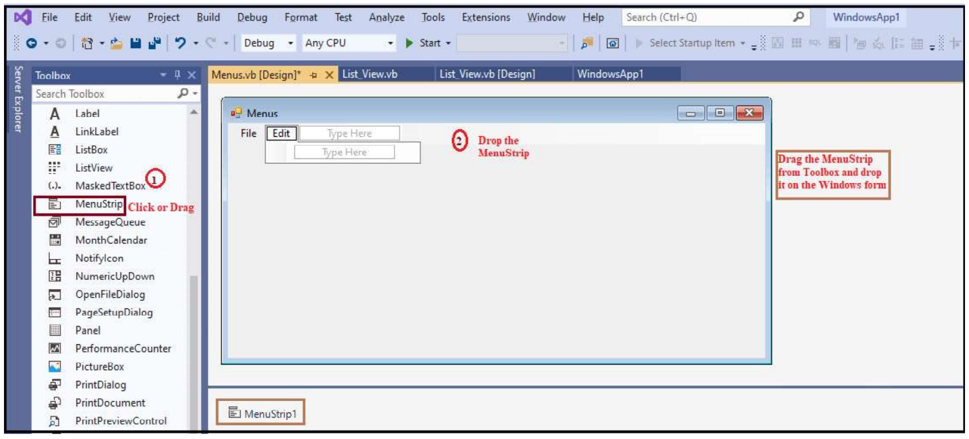
You need to add a MainMenu attribute to your form to add menus to your VB.NET application during design-time. You can build, add, and change menus and menu bars with the MainMenu control and set their properties in the Properties window. To add a MainMenu component, open the Forms Toolbox and add the MainMenu component to your form.Once you have added the control to your form, you quickly can add menus to your VB.NET windows forms.

In Windows type, a menu is used as a menu bar containing a list of related commands and is executed via MenuStrip Control also known as the VB.NET MenuStrip Control. The Menu control. Menu items are created with ToolStripMenuItem Objects. In addition, the ToolStripDropDownMenu and ToolStripMenuItem objects allow complete structure control, appearance, functionalities to create menu items, submenus, and drop-down menus in a VB.NET application.

Step 1: Drag the MenuStrip control from the toolbox and drop it on to the Form.



Step 2: We can set various properties of the Menu by clicking on the MenuStrip control once the MenuStrip is added to the form.



Properties of the MenuStrip Control

Properties	Description
CanOverflow	It gets or sets a value indicate that the MenuStrip supports overflow functionality.
GripStyle	It gets or sets the visibility of the grip used to reposition the control.
MdiWindowListItem	It gets or sets the ToolStripMenuItem that is used to display a list of Multiple-document interface (MDI) child forms.
Stretch	It gets or sets a value indicating whether the MenuStrip stretches from end to end in its container.
ShowItemToolTips	It gets or sets a value indicating whether ToolTips are shown for the MenuStrip.

Events of the MenuStrip Control

Events	Description
MenuActivate	Happen when the user accesses the menu with the keyboard or mouse.
MenuDeactivate	Happen when the MenuStrip is deactivated.

Windows Forms contain a rich set of classes for creating your own custom menus with modern appearance, look and feel. The **MenuStrip**, **ToolStripMenuItem**, **ContextMenuStrip** controls are used to create menu bars and context menus efficiently.

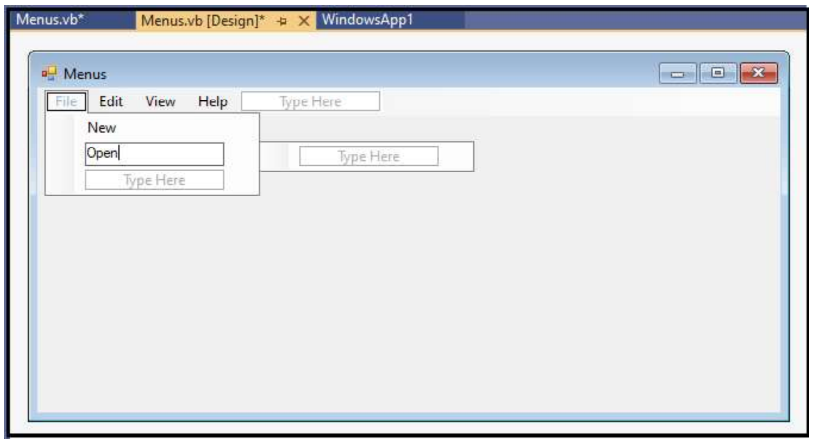
Controls	Description
MenuStrip	It provides a menu system for a form.
ContextMenuStrip	It represents a shortcut menu.
ToolStripMenuItem	It represents a selectable option displayed on a MenuStrip or ContextMenuStrip. The ToolStripMenuItem control replaces and adds functionality to the MenuItem control of previous versions.

NOTES

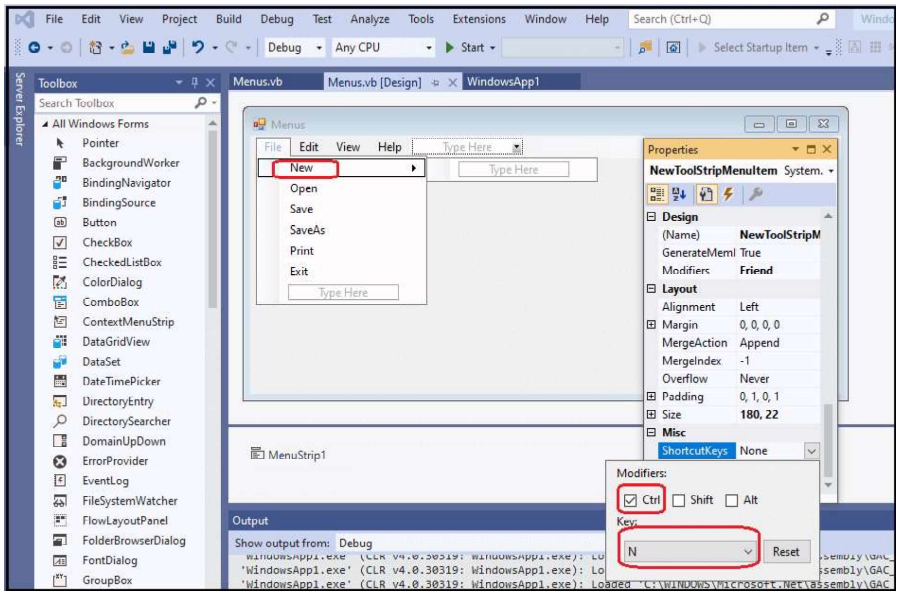
Methods of the MenuStrip Control

Methods	Description
CreateAccessibilityInstance()	It is used to create a new accessibility instance for the MenuStrip Control.
CreateDefaultItem()	The CreateDefaultItem method is used to create a ToolStripMenuItem with the specified text, image, and event handlers for the new MenuStrip.
ProcessCmdKey()	The ProcessCmdKey method is used to process the command key in the MenuStrip Control.
OnMenuActivate()	It is used to initiate the MenuActivate event in the MenuStrip control.
OnMenuDeactivate()	It is used to start the MenuDeactivate event in the MenuStrip control.

In the screenshot below, we have created the menu and sub-items of the menu bar in the form.



Now, we write the Shortcut keys for the File subitems, such as **New→Ctrl + N, Open → Ctrl + O**, etc.



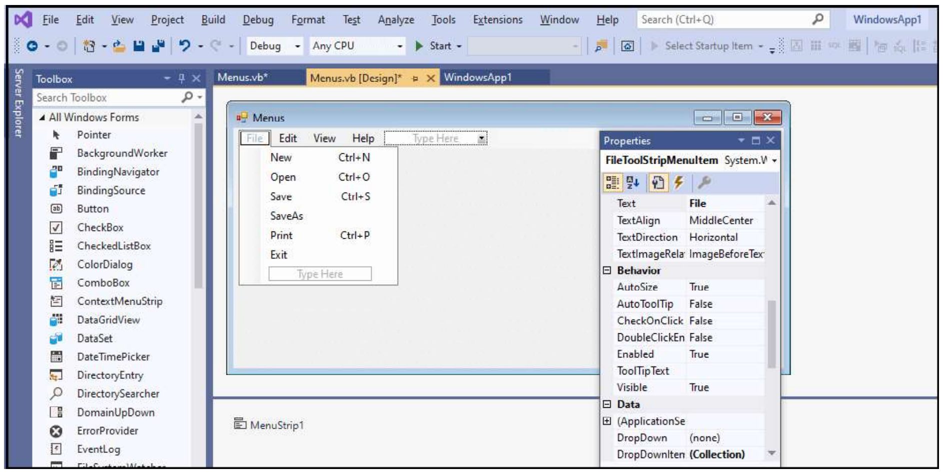
Show output from: Debug

```
'WindowsApp1.exe' (CLR v4.0.30319: WindowsApp1.exe): Lo
'WindowsApp1.exe' (CLR v4.0.30319: WindowsApp1.exe): Lo
'WindowsApp1.exe' (CLR v4.0.30319: WindowsApp1.exe): Loaded 'C:\WINDOWS\Microsoft.Net\assembly\GAC
```

Design	
(Name)	NewToolStripM
GenerateMeml	True
Modifiers	Friend
Layout	
Alignment	Left
Margin	0, 0, 0, 0
MergeAction	Append
MergeIndex	-1
Overflow	Never
Padding	0, 1, 0, 1
Size	180, 22
Misc	
ShortcutKeys	None

Modifiers: ✓ Ctrl ☐ Shift ☐ Alt
Key: N

After that, we can see the **subitems** of the Files with their Shortcut keys,



FileToolStripMenuItem System.W

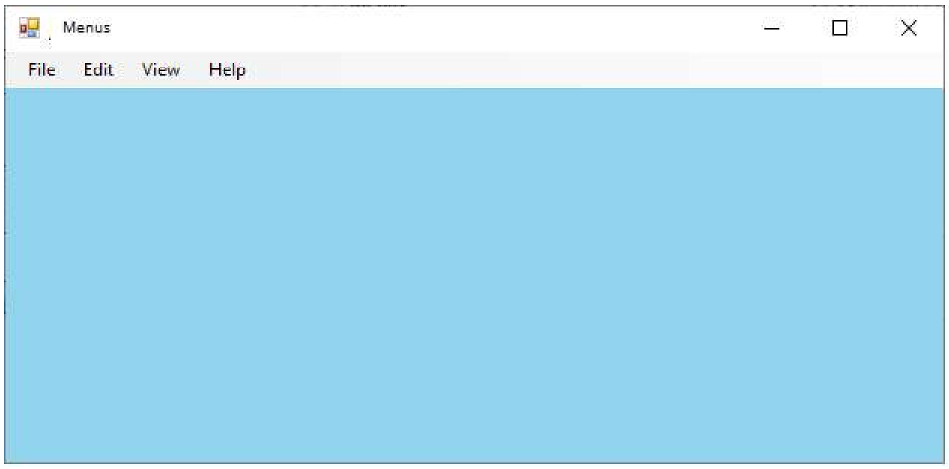
Text	File
TextAlign	MiddleCenter
TextDirection	Horizontal
TextImageRela	ImageBeforeTex
Behavior	
AutoSize	True
AutoToolTip	False
CheckOnClick	False
DoubleClickEn	False
Enabled	True
ToolTipText	
Visible	True
Data	
(ApplicationSe	
DropDown	(none)
DropDownIten	(Collection)

The code for Menus is given below:

Menus.vb

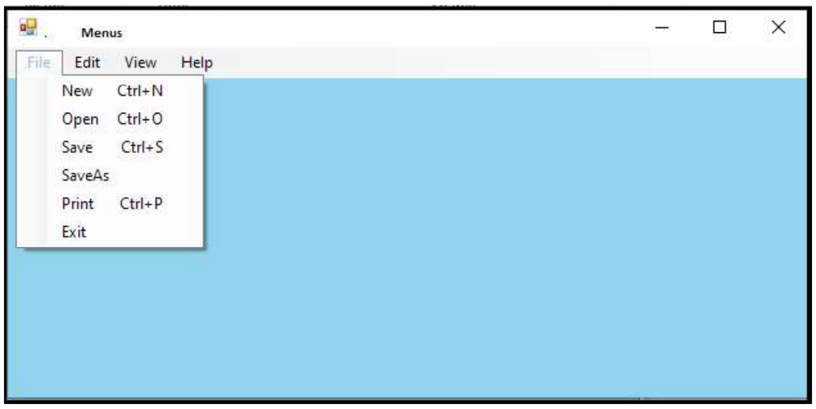
```
    Public Class Menus
    Private Sub Menus_Load(sender As Object, e As EventArgs)
Handles MyBase.Load
     Me. Text = " Menus.vb" 'set the title of the bar
     BackColor = Color.SkyBlue
     End Sub
```

Output:



NOTES

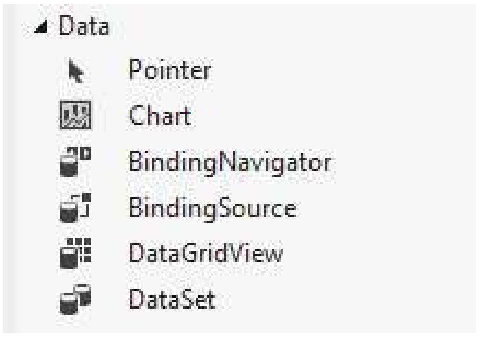
Click on the File menu that shows the multiple options related to files.



Application using Data Controls

Data Controls are used to create interfaces for manipulating and editing data from a data source. For example, TextBox or DropdownBox can be used to display and/or edit data from a data source.

The Visual Basic 2012 toolbox provides the data controls as shown in screenshot below:



The data control can be used to perform the following tasks:

1. It is used for connecting to a database.
2. To open a specified database table.
3. For creating a virtual table based on a database query.
4. Passing database fields to other Visual Basic tools.
5. Adding/updating records.
6. Identify errors that may occur while accessing data.
7. Close the database connection.

Properties of Data Controls

Data Control Properties	Description
Align	It determines where data control is displayed.
Caption	Phrase displayed on the data control.
ConnectionString	It contains the information for establishing a connection to a database.
Recordset	A set of records defined by a data control's ConnectionString and RecordSource properties. Run-time only.
LockType	It specifies the type of locks placed on records during editing (default setting makes databases read-only).
RecordSource	It determines the table (or virtual table) the data control is attached to.

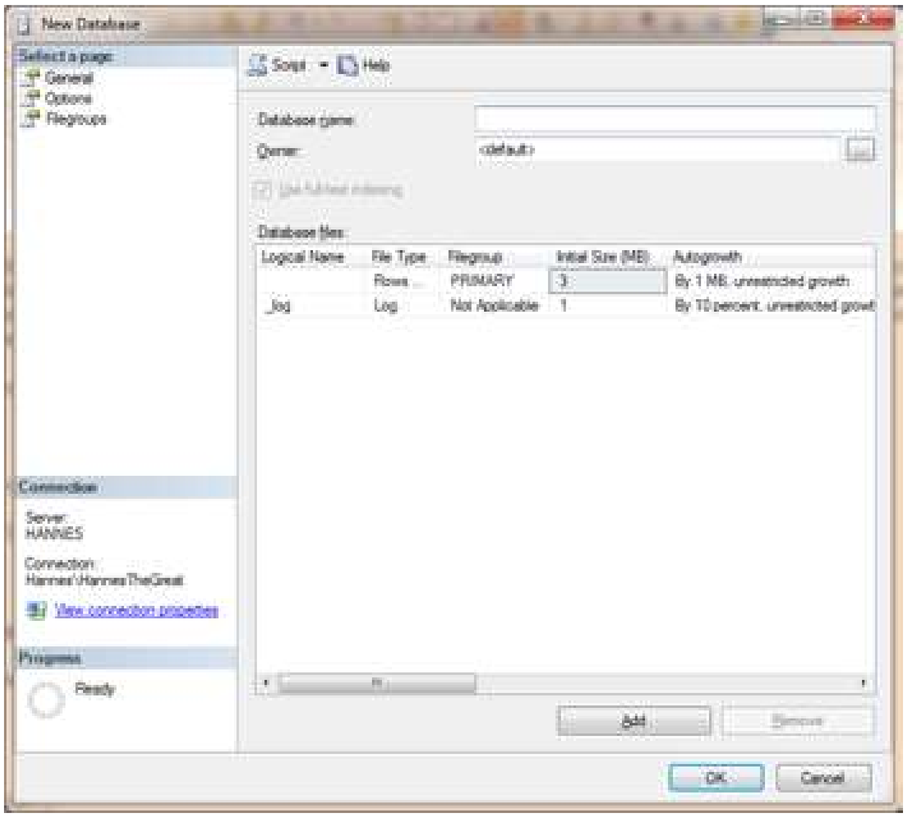
NOTES

Creating the Database

Microsoft Access or SQL Server can be used to create your database.

Following are steps for creating an SQL server database:

1. Open SQL Server Management Studio.
2. Right click on the folder named Databases and select New.



Database name:
Owner: <default>
Use full-text indexing

Database files:

Logical Name	File Type	Filegroup	Initial Size (MB)	Autogrowth
	Rows ...	PRIMARY	3	By 1 MB, unrestricted growth
_log	Log	Not Applicable	1	By 10 percent, unrestricted growt

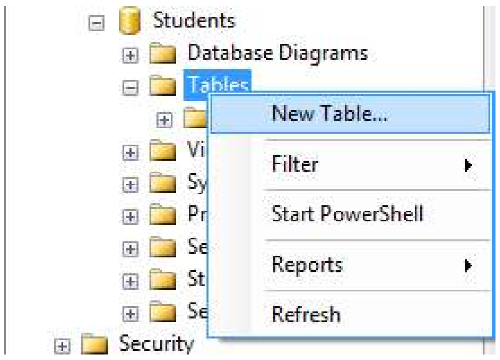
Connection
Server:
HANNES
Connection:
Hannes\HannesTheGreat
View connection properties

Progress
Ready

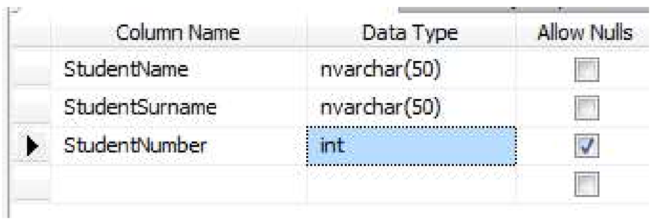
3. Give it a name of Students and click OK. After that you will now see Students in the list of databases.

NOTES

4. Expand Students in the database list, and right click on Tables, then select New Table.



5. Enter the following fields in the table.



Column Name	Data Type	Allow Nulls
StudentName	nvarchar(50)	☐
StudentSurname	nvarchar(50)	☐
StudentNumber	int	☑
		☐

6. Click on the Save button and name your Table StudentInfo.
7. In SQL Management Studio, click on New Query and write the following:

```
INSERT INTO [Students].[dbo].[StudentInfo]
 ([StudentName]
 ,[StudentSurname]
 ,[StudentNumber])
 VALUES
 ('Hannes', 'du Preez', 1)
 INSERT INTO [Students].[dbo].[StudentInfo]
 ([StudentName]
 ,[StudentSurname]
 ,[StudentNumber])
 VALUES
 ('YourName', 'YourSurname', 2)
GO
```

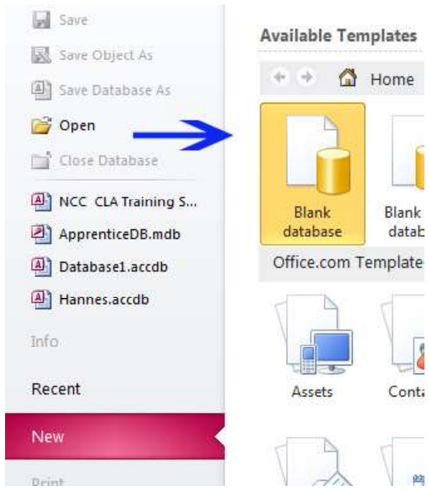
8. Click on Execute.

Creating a Microsoft Access 2010 Database

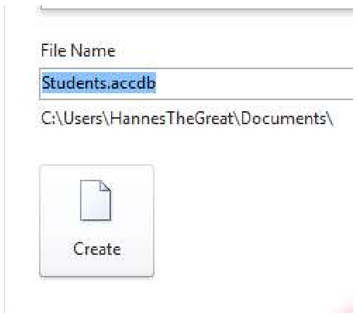
Steps for creating an MS Access 2010 database are as follows:

1. Open Microsoft Access.

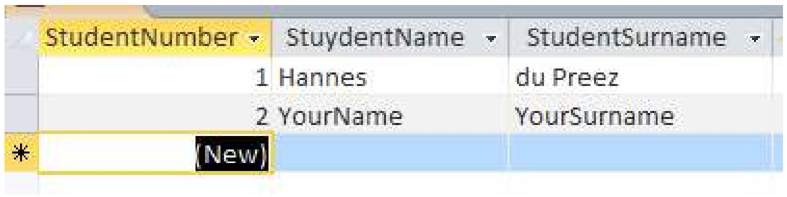
2. Select Blank Database.



3. On the right side of the screen enter the File name, Students.accdb (in this case) and click Create.



4. Inside the new Screen, edit the Columns and data to reflect.



StudentNumber	StuydentName	StudentSurname
1	Hannes	du Preez
2	YourName	YourSurname
(New)		

5. Save the table as StudentInfo.

Application using Common Dialogs

All Windows applications use standard **dialog boxes** for common operations. These dialog boxes are implemented as standard controls in the Toolbox. To use any of the **common dialog controls** in your interface, just place the appropriate control from the Dialog section of the Toolbox on your form and activate it from within your code by calling the ShowDialog method.

The Common Dialog control provides a standard set of dialog boxes for operations such as opening, saving, and printing files, as well as selecting colors and fonts and displaying help. Any six of the different dialog boxes can be displayed with just one Common Dialog control. A particular dialog box is displayed by using one of the six "Show..." methods of the Common Dialog control: **ShowOpen**, **ShowSave**, **ShowPrinter**, **ShowColor**, **ShowFont**, or **ShowHelp**.

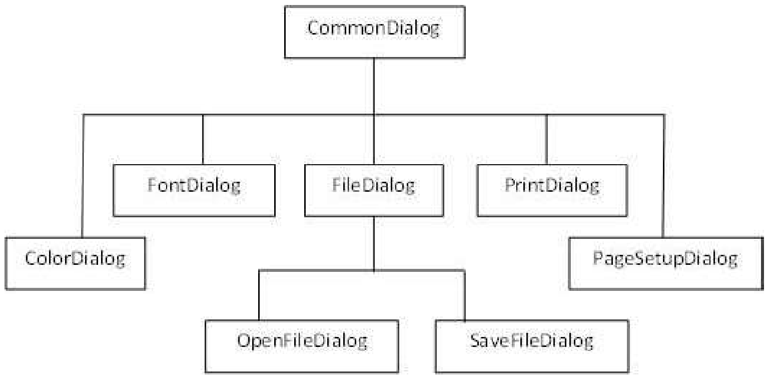
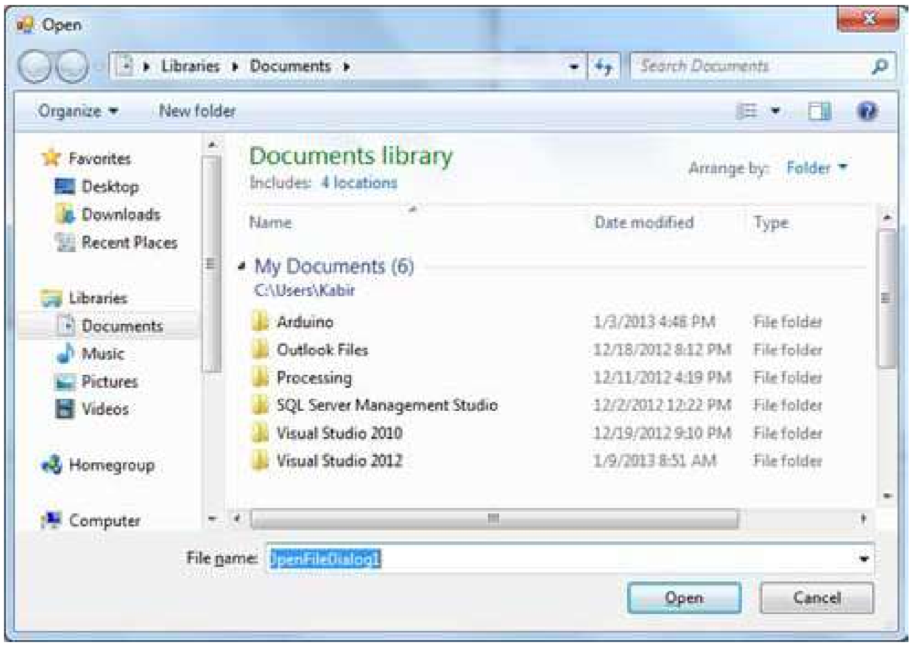


Fig. 2.1 *Common Dialog Controls*

OpenFileDialog : 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.



My Documents (6)
C:\Users\Kabir

Name	Date modified	Type
Arduino	1/3/2013 4:48 PM	File folder
Outlook Files	12/18/2012 8:12 PM	File folder
Processing	12/11/2012 4:19 PM	File folder
SQL Server Management Studio	12/2/2012 12:22 PM	File folder
Visual Studio 2010	12/19/2012 9:10 PM	File folder
Visual Studio 2012	1/9/2013 8:51 AM	File folder

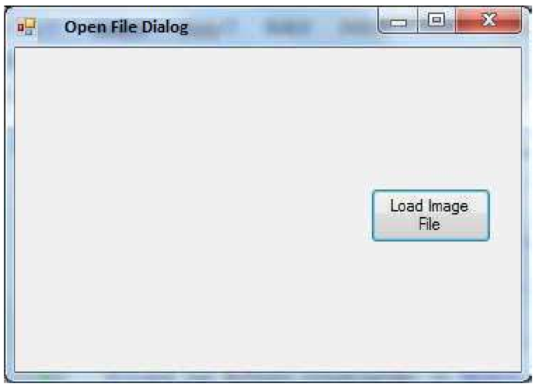
File name: OpenFileDialog1

Consider an example of loading an image file in a picture box, using the open file dialog box. Apply the following steps:

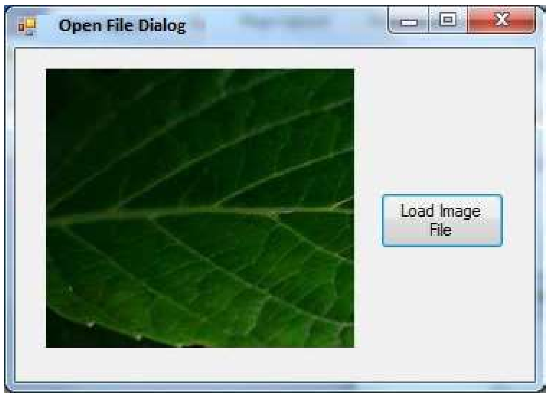
1. Drag and drop a PictureBox control, a Button control and OpenFileDialog control on the form.
2. Set the Text property of the button control to 'Load Image File'.
3. Double-click the Load Image File button and modify the code of the Click event as given below:

```
    Private Sub Button1_Click(sender As Object, e As EventArgs)
Handles Button1.Click
    If  OpenFileDialog1.ShowDialog  <>  Windows.Forms.
DialogResult.Cancel Then
    PictureBox1.Image  =  Image.FromFile  (OpenFileDialog1.
FileName)
    End If
    End Sub
```

Output:



Click on the Load Image File button to load an image stored on your computer**.**



NOTES

FontDialog: It displays a dialog box that enables users to set a font and its attribute.

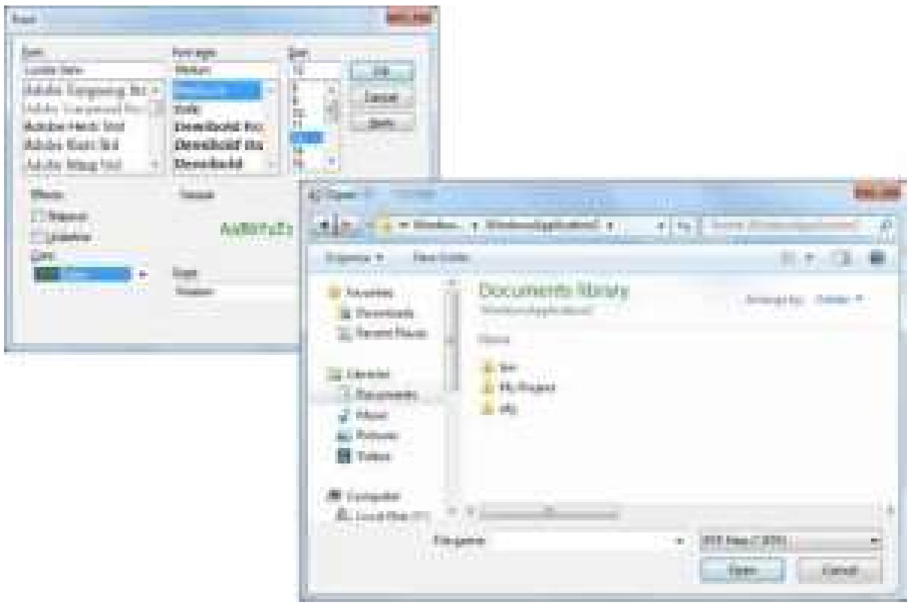
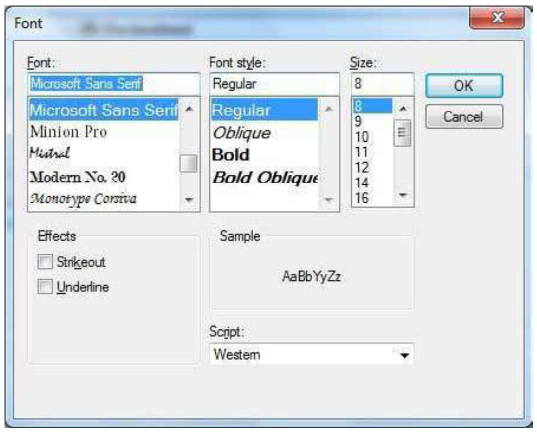


Fig. 2.2 Open and Font Common Dialog Boxes

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.



By default, the Color ComboBox is not shown on the Font dialog box. You should set the ShowColor property of the FontDialog control to be True.

Consider an example to change the font and color of the text from a rich text control using the Font dialog box. Apply the following steps:

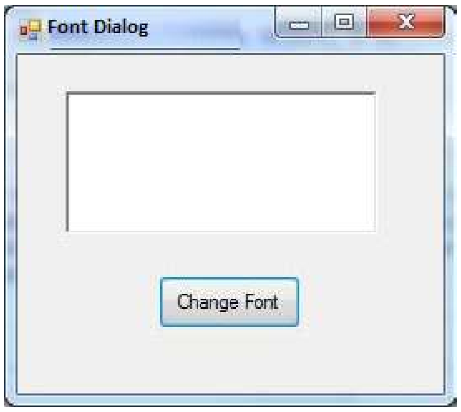
1. Drag and drop a RichTextBox control, a Button control and a FontDialog control on the form.
2. Set the Text property of the button control to 'Change Font'.

3. Set the ShowColor property of the FontDialog control to True.
4. Double-click the Change Color button and modify the code of the Click event as given below:

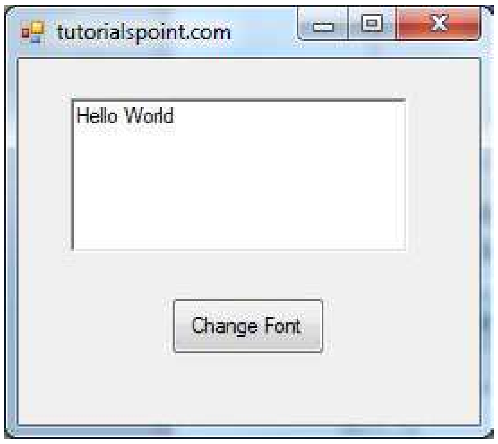
```
    Private Sub Button1_Click(sender As Object, e As EventArgs)
Handles Button1.Click
    If  FontDialog1.ShowDialog  <>  Windows.Forms.
DialogResult.Cancel Then
     RichTextBox1.ForeColor = FontDialog1.Color
     RichTextBox1.Font = FontDialog1.Font
     End If
    End Sub
```

Output:

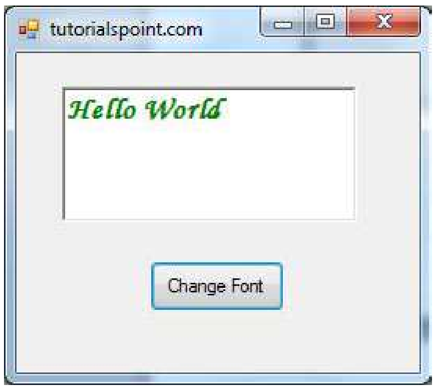
The output obtained when the application is compiled and run using Start button available at the Microsoft Visual Studio tool bar will be:



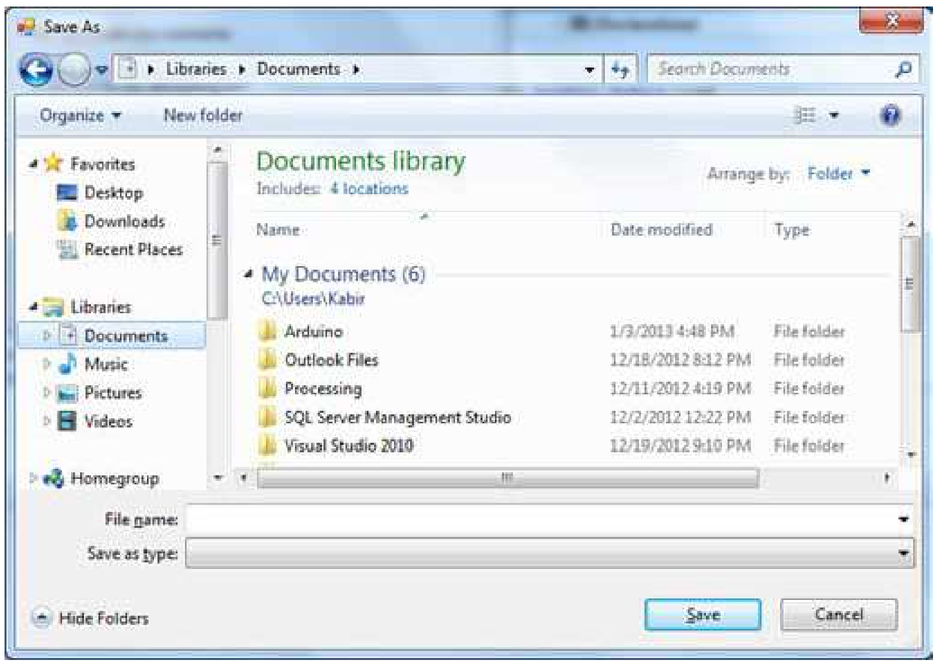
Enter some text and Click on the Change Font button.



In the Font dialog box, choose a font and a color, and then press the OK button. The font and color selected will be added as the font and foreground color of the text in the Rich text frame.



SaveFileDialog: 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. The SaveFileDialog control class inherits from the abstract class FileDialog.



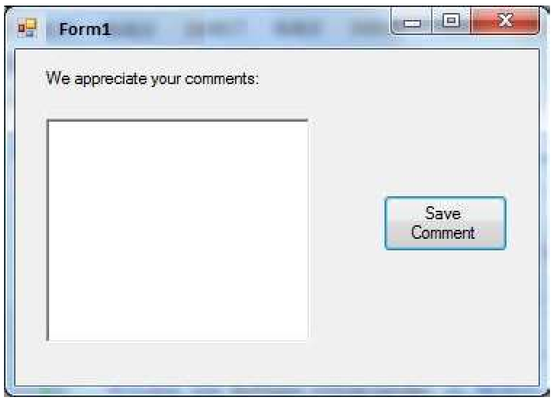
Consider an example to save the text entered into a rich text box by the user using the save file dialog box. Apply the following steps:

1. Drag and drop a Label control, a RichTextBox control, a Button control and a SaveFileDialog control on the form.
2. Set the Text property of the label and the button control to 'We appreciate your comments' and 'Save Comments', respectively.
3. Double-click the Save Comments button and modify the code as given below:

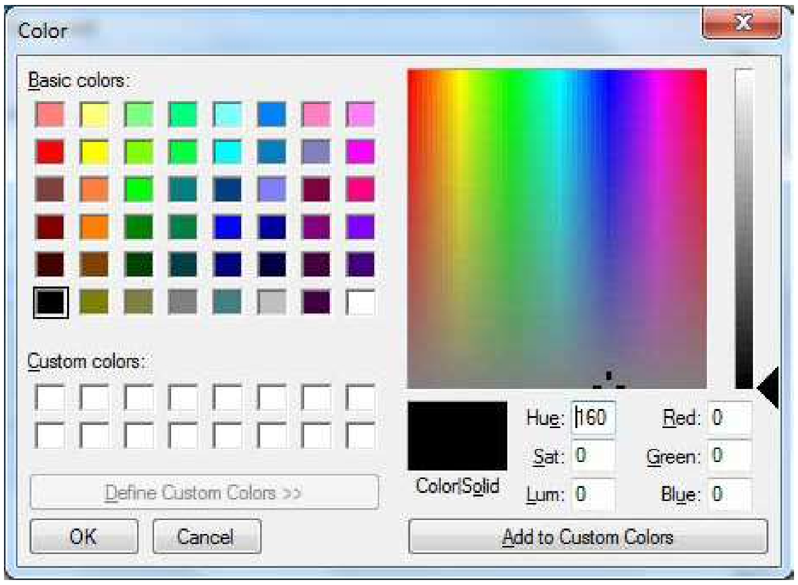
NOTES

```
    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, RichTextBox1.Text, True)
     End If
    End Sub
```

When the application is compiled and run using Start button available at the Microsoft Visual Studio tool bar, it will show the screenshot given below:



ColorDialog: It allows users to choose a color or choose custom colors from a set of predefined colors. 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.



Hue: 160
Sat: 0
Lum: 0
Red: 0
Green: 0
Blue: 0

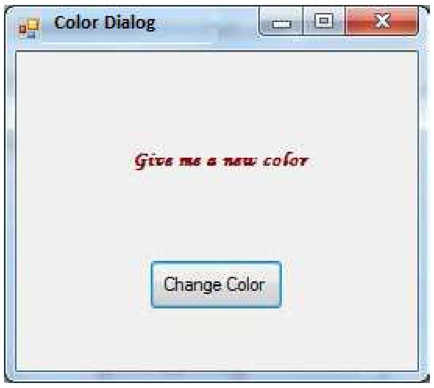
NOTES

Consider an example to change the forecolor of a label control using the color dialog box. Apply the following steps:

1. Drag and drop a label control, a button control and a ColorDialog control on the form.
2. Set the Text property of the label and the button control to 'Give me a new Color' and 'Change Color', respectively.
3. Change the font of the label as per your likings.
4. Double-click the Change Color button and modify the code of the Click event as given below.

```
     Private Sub Button1_Click(sender As Object, e As EventArgs)
Handles Button1.Click
     If ColorDialog1.ShowDialog <> Windows.Forms.
DialogResult.Cancel Then
      Label1.ForeColor = ColorDialog1.Color
      End If
     End Sub
```

When the application is compiled and run using Start button available at the Microsoft Visual Studio tool bar, it will show the following output window.



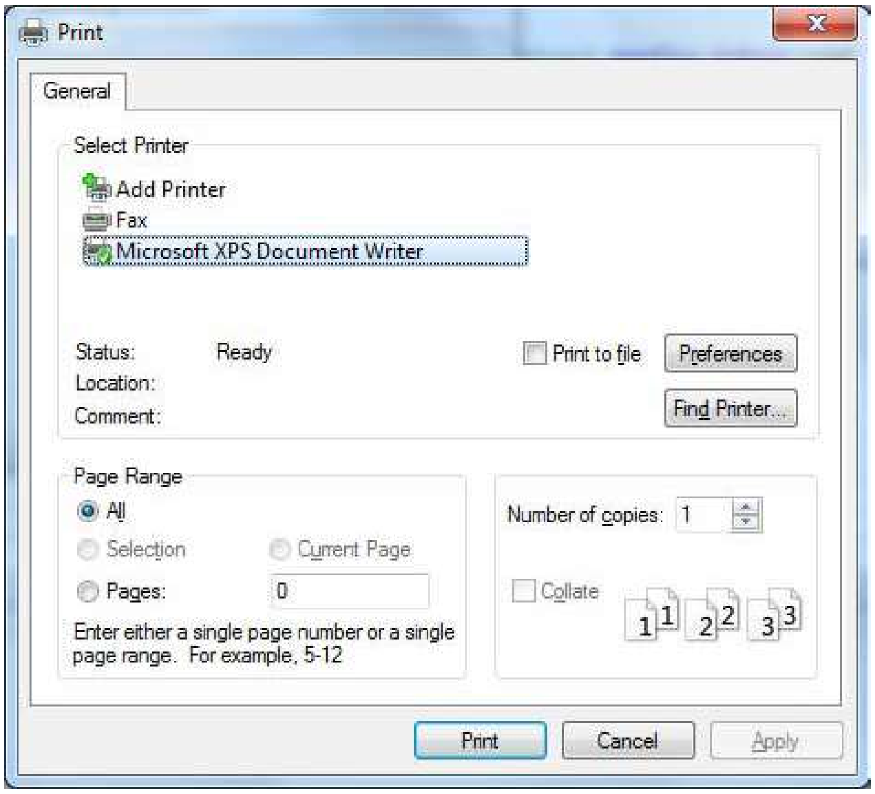
Clicking on the Change Color button, the color dialog appears, select a color and click the OK button. The selected color will be applied as the forecolor of the text of the label.

PrintDialog: It displays a dialog box that enables users to select a printer and set its attributes. There are various other controls related to printing of documents. Let us have a brief look at these controls and their purpose.

1. **PrintDocument control:** It provides support for actual events and operations of printing in Visual Basic and sets the properties for printing.
2. **PrinterSettings control:** It is used to configure how a document is printed by specifying the printer.
3. **PageSetUpDialog control:** It allows the user to specify page-related print settings including page orientation, paper size and margin size.

NOTES

4. **PrintPreviewControl control:** It represents the raw preview part of print previewing from a Windows Forms application, without any dialog boxes or buttons.
5. **PrintPreviewDialog control:** It represents a dialog box form that contains a PrintPreviewControl for printing from a Windows Forms application.



Consider an example to show a Print dialog box in a form. Apply the following steps:

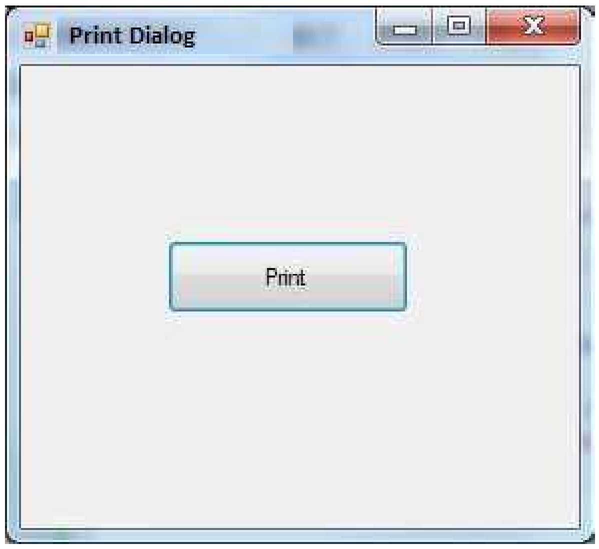
1. Add a PrintDocument control, a PrintDialog control and a Button control on the form. The PrintDocument and the PrintDialog controls are found on the Print category of the controls toolbox.
2. Change the text of the button to 'Print'.
3. Double-click the Print button and modify the code of the Click event as shown below:

```
    Private Sub Button1_Click(sender As Object, e As EventArgs)
Handles Button1.Click
     PrintDialog1.Document = PrintDocument1
    PrintDialog1.PrinterSettings = PrintDocument1.
PrinterSettings
     PrintDialog1.AllowSomePages = True
```

NOTES

```
      If PrintDialog1.ShowDialog = DialogResult.OK Then
     PrintDocument1.PrinterSettings = PrintDialog1.
PrinterSettings
      PrintDocument1.Print()
      End If
     End Sub
```

When the application is compiled and run using Start button available at the Microsoft Visual Studio tool bar, the output produced will be:



BLOCK 3

This block will cover the following topics:

1. Drag and drop events
2. Database management
3. Creating ActiveX controls
4. Database Object (DAO) and properties
5. Active Data Objects (ADO) and OLEDB

Drag and Drop Events

It is essentially a pointing interface gesture in the drag and drop case, in which the user selects a virtual object by "Grabbing" it and moving it to another position or to another virtual object.

You have certainly used drag and drop techniques as a Windows user to copy or transfer files from one folder to another, to remove a file by dragging it to the recycling bin, and to perform actions in different programs of the application. In Visual Basic, the drag-and-drop features allow you to integrate this functionality into the programs you are creating. The action of holding a mouse button down and moving a control is called **dragging**, and the action of releasing the button is called **dropping**.

Basically, a control may act as a source of a drag-and-drop process or as a destination. Visual Basic supports two drag-and-drop modes, automatic or manual. You only need to set a property in automatic mode at design time or at run time and let Visual Basic do it all. Conversely, in manual mode you have to respond to a number of events that occur while dragging is in progress, but in return you get better control over the process. To incorporate drag and drop functionality in your VB programs, you use a handful of properties, events, and methods.

Properties

The two properties involved are DragMode that specifies whether Automatic or Manual dragging will be used, and DragIcon that specifies which icon is displayed when the control is dragged.

Events

It involves two events i.e. DragDrop, which happens when a control is lowered onto the target, and DragOver, which happens when a control is dragged over the object.

Method

The Drag method starts or stops manual dragging.

Program 3.1: Create a program on drag and drop operation. For this, just create a VB.net windows application. Then design a form with Drag Drop and control &event procedure. To enable drag & drop for text, first you have to place two textboxes and set Allowdrop property of a second textbox to true and after that write the code given below:

```
    Private MouseIsDown As Boolean = False 'variable
declaration

    Private Sub TextBox1_MouseDown(ByVal sender As Object,
ByVal e As _

    System.Windows.Forms.MouseEventArgs) Handles
TextBox1.MouseDown

    'Set a flag to show that the mouse is down.

    MouseIsDown = True

    End Sub


    Private Sub TextBox1_MouseMove(ByVal sender As Object,
ByVal e As _

    System.Windows.Forms.MouseEventArgs) Handles
TextBox1.MouseMove
```

```
     If MouseIsDown Then
     'Initiate dragging.
     TextBox1.DoDragDrop(TextBox1.Text,DragDropEffects.Copy)
     End If
     MouseIsDown = False
     End Sub

     Private Sub TextBox2_DragEnter(ByVal sender As Object,
ByVal e As _
     System.Windows.Forms.DragEventArgs)  Handles
TextBox2.DragEnter
     'Check the format of the data being dropped.
     If (e.Data.GetDataPresent(DataFormats.Text)) Then
     'Display the copy cursor.
     e.Effect = DragDropEffects.Copy
     Else
     'Display the no-drop cursor.
     e.Effect = DragDropEffects.None
     End If
     End Sub

     Private Sub TextBox2_DragDrop(ByVal sender As Object,
ByVal e As _
     System.Windows.Forms.DragEventArgs)  Handles
TextBox2.DragDrop
     'Paste the text.
     TextBox2.Text = e.Data.GetData(DataFormats.Text)
     End Sub
```

From the above code, it can be seen that the DoDragDrop method is called in the MouseMove event and the MouseDown event is used to set a flag, which shows that the mouse is down. In the MouseMove event, the MouseIsDown flag is set to False. You can handle the drag in the MouseDown event also. Dring this every time a user clicks the control, and then no-drag cursor would be displayed.

The GetDataPresent method checks the format of the data being dragged in case of DragEnter event. In our case it is text, so the Effect property is set to Copy, which in turn displays the copy cursor. The GetData method is used to retrieve the text from the DataObject. In case of DragDrop event it also assigns it to the target TextBox.

The example code given below draggs a different type of data and provides support for both cutting and copying. For these just add two picturebox controls and write the code given below:

```
    Private Sub Form1_Load(ByVal sender As System.Object,
ByVal e As _
    System.EventArgs) Handles MyBase.Load
    'Enable dropping.
    PictureBox2.AllowDrop = True
    End Sub

    Private Sub PictureBox1_MouseDown(ByVal sender As Object,
ByVal e As _
    System.Windows.Forms.MouseEventArgs) Handles
PictureBox1.MouseDown
    If Not PictureBox1.Image Is Nothing Then
    'Set a flag to show that the mouse is down.
    m_MouseIsDown = True
    End If
    End Sub

    Private Sub PictureBox1_MouseMove(ByVal sender As Object,
ByVal e As _
    System.Windows.Forms.MouseEventArgs) Handles
PictureBox1.MouseMove
    If m_MouseIsDown Then
    'Initiate dragging and allow either copy or move.
    PictureBox1.DoDragDrop(PictureBox1.Image,
DragDropEffects.Copy Or _
    DragDropEffects.Move)
    End If
    m_MouseIsDown = False
    End Sub

    Private Sub PictureBox2_DragEnter(ByVal sender As Object,
ByVal e As _
    System.Windows.Forms.DragEventArgs) Handles
PictureBox2.DragEnter
    If e.Data.GetDataPresent(DataFormats.Bitmap) Then
    'Check for the CTRL key.
    If e.KeyState = 9 Then
```

NOTES

```
    e.Effect = DragDropEffects.Copy
    Else
    e.Effect = DragDropEffects.Move
    End If
    Else
    e.Effect = DragDropEffects.None
    End if
    End sub

    Private Sub PictureBox2_DragDrop(ByVal sender As Object,
ByVal e As _
    System.Windows.Forms.DragEventArgs) Handles
PictureBox2.DragDrop
    'Assign the image to the PictureBox.
    PictureBox2.Image = e.Data.GetData(DataFormats.Bitmap)
    'If the CTRL key is not pressed, delete the source picture.
    If Not e.KeyState = 8 Then
    PictureBox1.Image = Nothing
    End If
    End Sub
```

The AllowDrop property for the second PictureBox control is set in the Form1_Load event. In both the DragEnter and DragDrop events, the code checks to see if the CTRL key is pressed to determine whether to copy or move the picture.

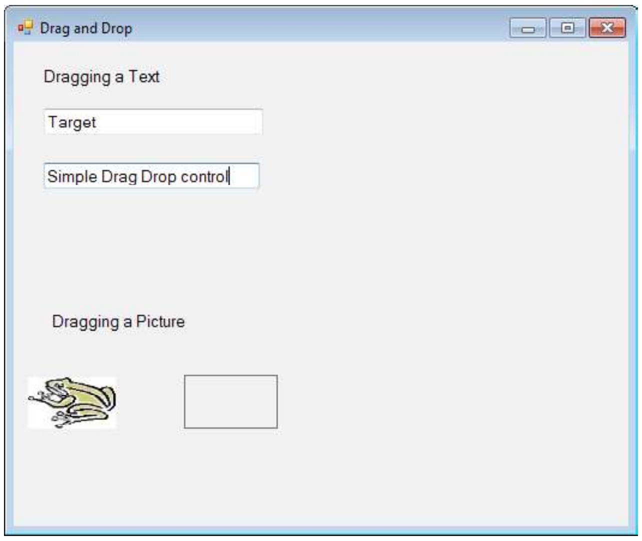


Fig. 3.1 *Control before being dragged to a target*

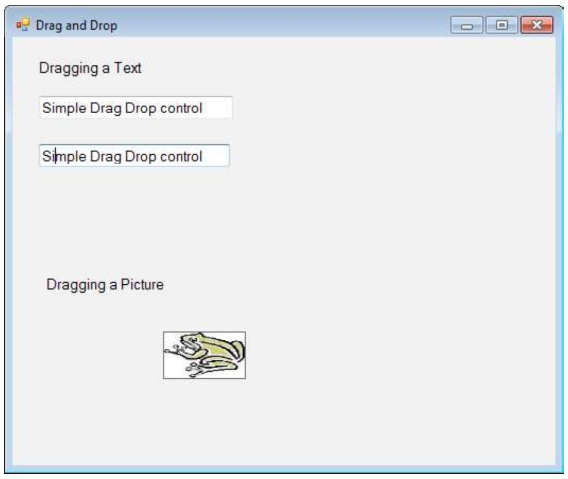


Fig. 3.2 *Control after being dragged to a target*

Database Management

Database means a place where data can be stored in a structured manner. It is a shared collection or batch of data that is logically related, along with their descriptions designed to meet the information requirements of an organization.

Database Management System (DBMS) is a software system that allows users to not only define and create a database but also maintain it and control its access. A database management system can be called a collection of interrelated data (usually called database) and a collection or set of programs that helps in accessing, updating and managing that data (which form part of a database management system).

The primary benefit of using a DBMS is to impose a logical and structured organization on data. A DBMS provides simple mechanisms for processing huge volumes of data because it is optimized for operations of this type. The two basic operations performed by the DBMS are as follows:

1. Management of data in the database
2. Management of users associated with the database

Management of the data means specifying how data will be stored, structured and accessed in the database. This includes the following:

- **Defining:** Specifying data types and structures, and constraints for data to be stored.
- **Constructing:** Storing data in a storage medium.
- **Manipulating:** Involves querying, updating and generating reports.
- **Sharing:** Allowing multiple users and programs to access data simultaneously.

Further, the database management system must offer safety and security of the information stored, in case unauthorized access is attempted or the system crashes. If data is required to be shared among many users, the system must ensure that possible anomalous results are avoided.

NOTES

Management of database users means managing the users in such a way that they are able to perform any desired operations on the database. A DBMS also ensures that a user cannot perform any operation for which he is not authorized.

In short, a DBMS is a collection of programs performing all necessary actions associated with a database. There are many DBMSs available in the market, such as MySQL, Sybase, Oracle, MongoDB, Informix, PostgreSQL, SQL Server etc.

How to Create Active X Control

ActiveX Controls were previously known as OLE controls. To render web pages more interactive, an ActiveX control can be put on web pages. Much like you put a Java applet on a page on the internet. To bring advanced features to the user experience, app developers have used ActiveX controls on their web pages.

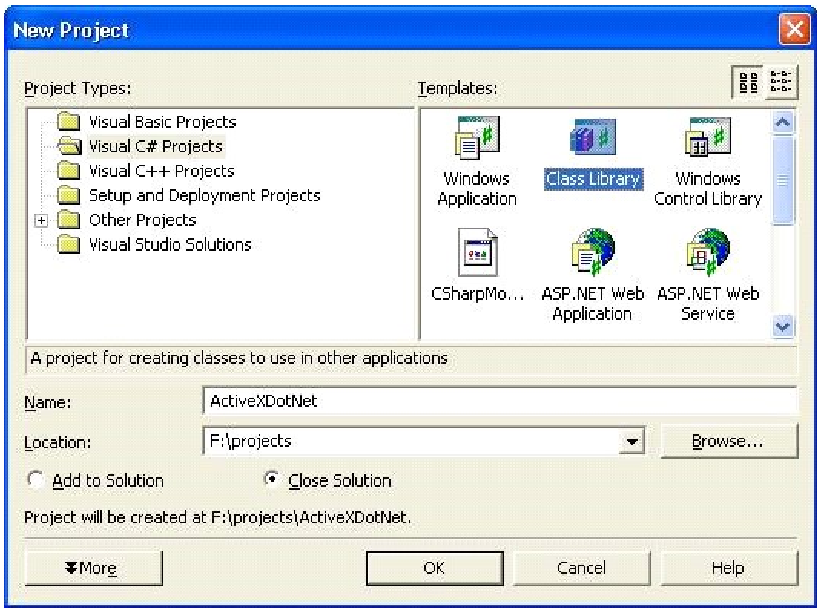
Program 3.2: To create an ActiveX control that will show a simple user interface and accept input from a web page. Following are the steps for creating an ActiveX control.

1. Create an assembly (class library project) containing an item of type User Control.
2. Expose an interface for the control.
3. Embed the user control into a web page.
4. Transfer data from a web form to the control and display the data on the control.

First, we will create a simple ActiveX control to get an overall idea about how to create ActiveX controls.

Step 1: Create an assembly

Create a new project of type Class Library. Name the class library ActiveXDotNET.



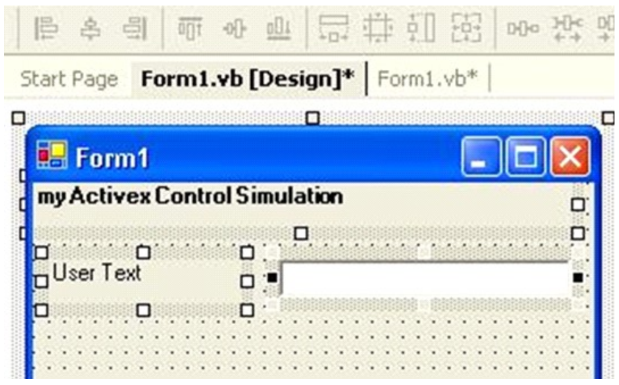
A project for creating classes to use in other applications

Name: ActiveXDotNet

Location: F:\projects

Project will be created at F:\projects\ActiveXDotNet.

Delete the Class1.cs file from your project once the project is developed, as it won't be required. Next, by right-clicking the project in your Solution Explorer, add User Control to the project, select Add, then User Control. Name your control as "myControl".



On the user control, add some UI elements, and a text box control named txtUserText. The txtUserText control will display the user data that is typed into the web form. This will demonstrate how to pass data to your User Control.

When you are done adding your user interface to the control we now have to add a key element to the control, an Interface. The interface will allow COM/COM+ objects to know what properties they can use. In this case, we are going to expose one public property named UserText. That property will allow us to set the value of the text box control.

Step 2: Expose the interface for the control

First, create a private String to hold the data passed from the web form to the control:

```
private Dim mStr_UserText as String
```

Place this String just inside the Class myControl.

Next, we will create a public property. The web page will use this property to pass text back to your control. This property will allow reading and writing of the value mStr_UserText.

```
Public Property UserText() As [String]
Get
Return mStr_UserText
End Get
Set(ByVal Value As [String])
mStr_UserText = value
'Update the text box control value also.
txtUserText.Text = value
End Set
```

End Property

In this example, you will note the extra code in the set section of the public property. We will set the private String value equal to the value passed to the property when a value is passed from the web form to the control. We are simply going to modify the value of the Text Box control directly. Typically you would not do this. Instead, you would raise an event and then validate the data being passed by examining the private variable mStr_UserText. Then you would set the value of the Text Box Control. However, it would add significant code to this example and for simplicity sake we are omitting that security precaution.

Now, you have a public property that .NET assemblies can use, you need to make that property available to the COM world. This can be done by creating an interface and making the myControl class inherit the interface. It allows COM objects to see what properties are made available. Now, the code will be:

```
Namespace ActiveX.NET
{
Public Interface AxMyControl
Property UserText() As String
End Property
End Interface 'AxMyControl
Public Class myControl
Inherits System.Windows.Forms.UserControl, AxMyControl
Private mStr_UserText As [String]
Public Property UserText() As String
Get
Return mStr_UserText
End Get
Set(ByVal Value As String)
mStr_UserText = value
'Update the text box control value also.
txtUserText.Text = value
End Set
End Property
End Class 'myControl
```

Notice that, we have an interface defined. The interface tells COM/COM+ that there is a public property available for use that is of type String and is readable (get) and writeable (set). All we do now is have the Class myControl inherit the interface and viola! We have a .NET assembly that acts like an ActiveX Control.

Step 3: Embed the user control in a web page

The last thing we do now is use the control in an example web page.

```
    <html>
    <body color=white>
    <hr>
    <font face=arial size=1>
    <OBJECT  id=”myControl1"  name=”myControl1"
classid=”ActiveX.NET.dll#ActiveXDotNET.myControl”width=288
height=72>
    </OBJECT>
    </font>
    <form name=”frm” id=”frm”>
    <input  type=”text”  name=”txt”  value=”enter  text
here”><input type=button value=”Click me”onClick=”doScript();”>
    </form>
    <hr>
    </body>
    <script language=”javascript”>
    function doScript()
    {
    myControl1.UserText = frm.txt.value;
    }
    </script>
    </html>
```

You will notice in the HTML code above, that you call your .NET assembly very similar to an ActiveX control; however there is no GUID, and no .OCX file. Your CLASSID is now the path to your DLL and the Namespace.Classname identifier. Refer to the code above to understand the syntax of the CLASSID object tag property. Place the HTML file and your DLL in the same directory on your web server and navigate to the HTML document. (Do not load the HTML document by double clicking on it, navigate to it in your browser by using the Fully Qualified URL.) *NOTE: You might need to add your web server to your Trusted Sites list in your Internet Explorer browser.

Step 4: Transfer data from the web form to the user control

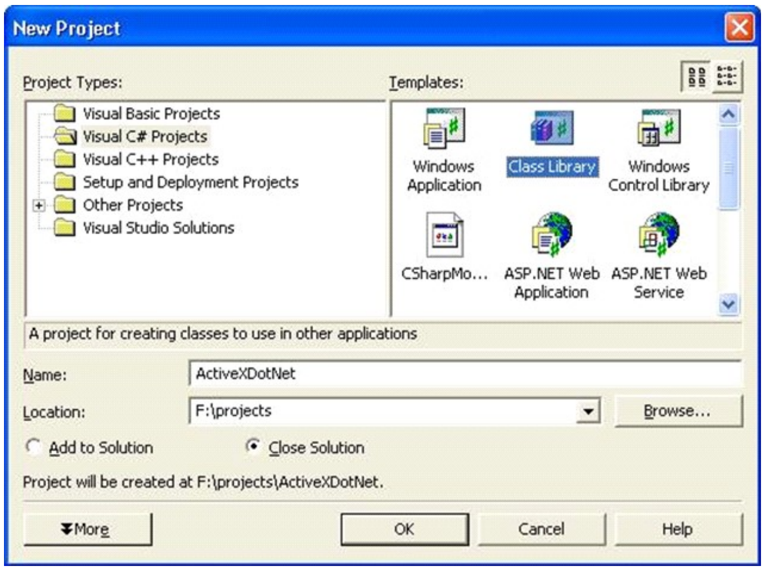
When you load the HTML page, your control should load into the page and you will see a web form with a text box and a button. In this example, if you type some text into the text box and click the button, it will use JavaScript to send the text from the web page form, to the User Control that you just built. Your User Control will then display the text in the Text Box control that I on the form.

NOTES

Program 3.3 Another program of ActiveX control.

Step 1: Create an assembly

First, you create a new project of type Class Library. Name the class library ActiveXDotNET.



A project for creating classes to use in other applications

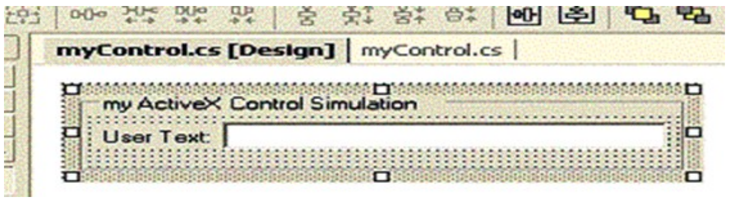
Name: ActiveXDotNet

Location: F:\projects

Add to Solution [✓ Close Solution]

Project will be created at F:\projects\ActiveXDotNet.

Once the project is created, delete the Class1.cs file from your project as it will not be necessary. Next, add a User Control to the project by right clicking on the project in your solution explorer, choose Add, then User Control. Name your user control myControl.



On the user control, add some UI elements, and a text box control named txtUserText. The txtUserText control will display the user data that is typed into the web form. This will demonstrate how to pass data to your User Control.

When you are done adding your user interface to the control we now have to add a key element to the control, an Interface. The interface will allow COM/ COM+ objects to know what properties they can use. In this case, we are going to expose one public property named UserText. That property will allow us to set the value of the text box control.

Step 2: Expose the Interface for the control

First, create a private String to hold the data passed from the web form to the control:

```
private String mStr_UserText;
```

Place this String just inside the Class myControl.

NOTES

Next, we will create a public property. The web page will use this property to pass text back to your control. This property will allow reading and writing of the value mStr_UserText.

```
public String UserText {
 get {
 return mStr_UserText;
 }
 set {
 mStr_UserText = value;
 //Update the text box control value also.
 txtUserText.Text = value;
 }
}
```

In this example, you will note the extra code in the set section of the public property. When a value is passed from the web form to the control we will set the private String value equal to the value passed to the property. In addition, we are simply going to modify the value of the Text Box control directly. Typically you would NOT do this. Instead, you would raise an event and then validate the data being passed by examining the private variable mStr_UserText. Then you would set the value of the Text Box Control. However, that would add significant code to this example and for simplicity sake we are omitting that security precaution.

Now that you have a public property that .NET assemblies can use, you need to make that property available to the COM world. We do this by creating an Interface and making the myControl class inherit the interface. This will allow COM objects to see what properties we have made available.

Your code will now look like this:

```
namespace ActiveXDotNET {
 public interface AxMyControl {
 String UserText {
 set;
 get;
 }
 }
 public class myControl: System.Windows.Forms.UserControl,
 AxMyControl {
 private String mStr_UserText;
 public String UserText {
 get {
 return mStr_UserText;
```

NOTES

```
 }
 set {
mStr_UserText = value;
 //Update the text box control value also.
 txtUserText.Text = value;
 }
 }
```

Notice that we now have an interface defined, the interface tells COM/ COM+ that there is a public property available for use that is of type String and is readable (get) and writeable (set). All we do now is have the Class myControl inherit the interface and viola! We have a .NET assembly that acts like an ActiveX Control.

Step 3: Embed the user control in a web page

The last thing we do now is use the control in an example web page.

```
<html>
<body color=white>
<hr>
<font face=arial size=1>
<OBJECT id="myControl1" name="myControl1" classid="ActiveX.NET.dll#ActiveX.NET.myControl" width=288 height=72>
</OBJECT>
</font>
<form name="frm" id="frm">
<input type="text" name="txt" value="enter text here"><input type=button value="Click me" onClick="doScript();">
</form>
<hr>
</body>
<script language="javascript">
 function doScript()
 {
 myControl1.UserText = frm.txt.value;
 }
</script>
</html>
```

You will notice in the HTML code above, that you call your .NET assembly very similar to an ActiveX control; however there is no GUID, and no .OCX file. Your CLASSID is now the path to your DLL and the Namespace.Classname identifier. Refer to the code above to understand the syntax of the CLASSID object tag property. Place the HTML file and your DLL in the same directory on your web server and navigate to the HTML document. (Do not load the HTML document by double clicking on it, navigate to it in your browser by using the Fully Qualified URL.) *NOTE: You might need to add your web server to your Trusted Sites list in your Internet Explorer browser.

Step 4: Transfer data from the web form to the user control

When you load the HTML page, your control should load into the page and you will see a web form with a text box and a button. In this example, if you type some text into the text box and click the button, it will use JavaScript to send the text from the web page form, to the User Control that you just built. Your User Control will then display the text in the Text Box control that I on the form.

Database Object (DAO) and Properties

DAO pattern or Data Access Object pattern is used to separate low level data accessing API or operations from high level business services. The participants in Data Access Object Pattern are as follows.

1. **DAO Interface:** It defines the standard operations to be performed on a model object(s).
2. **DAO Concrete Class:** It is responsible to get data from a data source.
3. **Model or Value Object:** It is simple POJO containing get/set methods to store data retrieved using DAO class.

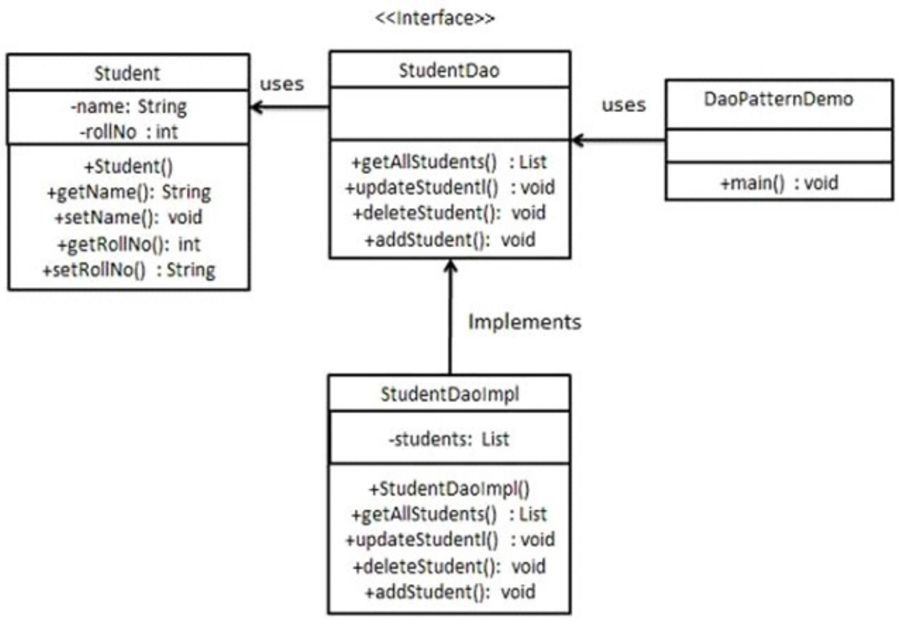
When it comes to implementing a data access solution in your VB applications, you currently have three choices: Data Access Objects (DAO), Remote Data Objects (RDO), and ActiveX Data Objects (ADO).

DAO was created before RDO and ADO, is a set of objects that enables client applications to programmatically access data. DAO not only allows you to access data but also helps in controling and managing local and remote databases in different formats. DAO can be used create and modify the database structure; create tables, queries, relationships, and indexes; retrieve, add, update, and remove data; implement security; work with different file formats; and link tables to other tables.

Implementation

Program 3.4: We are going to create a Student object acting as a Model or Value Object. StudentDao is Data Access Object Interface.StudentDaoImpl is concrete class implementing Data Access Object Interface. DaoPatternDemo, our demo class, will use StudentDao to demonstrate the use of Data Access Object pattern.

NOTES



Step 1: Create Value Object.

```
public class Student {
 private String name;
 private int rollNo;
 Student(String name, int rollNo){
 this.name = name;
 this.rollNo = rollNo;
 }

 public String getName() {
 return name;
 }
 public void setName(String name) {
 this.name = name;
 }
 public int getRollNo() {
 return rollNo;
 }
 public void setRollNo(int rollNo) {
 this.rollNo = rollNo;
 }
}
```

Step 2: Create Data Access Object Interface.

```
import java.util.List;
public interface StudentDao
{
 public List<Student> getAllStudents();
 public Student getStudent(int rollNo);
 public void updateStudent(Student student);
 public void deleteStudent(Student student);
}
```

Step 3: Create concrete class implementing above interface.

```
public class DaoPatternDemo
{
 public static void main(String[] args)
{
 StudentDao studentDao = new StudentDaoImpl();

 //print all students
 for (Student student : studentDao.getAllStudents())
{
  System.out.println(“Student: [RollNo : “ +
student.getRollNo() + “, Name : “ + student.getName() + “ ]”);
 }

 //update student
 Student student =studentDao.getAllStudents().get(0);
 student.setName(“Michael”);
 studentDao.updateStudent(student);

 //get the student
 studentDao.getStudent(0);
  System.out.println(“Student: [RollNo : “ +
student.getRollNo() + “, Name : “ + student.getName() + “ ]”);

 }
}
```

Step 4: Use the *StudentDao* to demonstrate Data Access Object pattern usage.

```
public class DaoPatternDemo {
 public static void main(String[] args) {
```

NOTES

```
      StudentDao studentDao = new StudentDaoImpl();
      //print all students
      for (Student student : studentDao.getAllStudents()) {
       System.out.println("Student: [RollNo : " +
student.getRollNo() + ", Name : " + student.getName() + " ]");
       }
      //update student
      Student student =studentDao.getAllStudents().get(0);
      student.setName("Michael");
      studentDao.updateStudent(student);
      //get the student
      studentDao.getStudent(0);
       System.out.println("Student: [RollNo : " +
student.getRollNo() + ", Name : " + student.getName() + " ]");

       }
     }
```

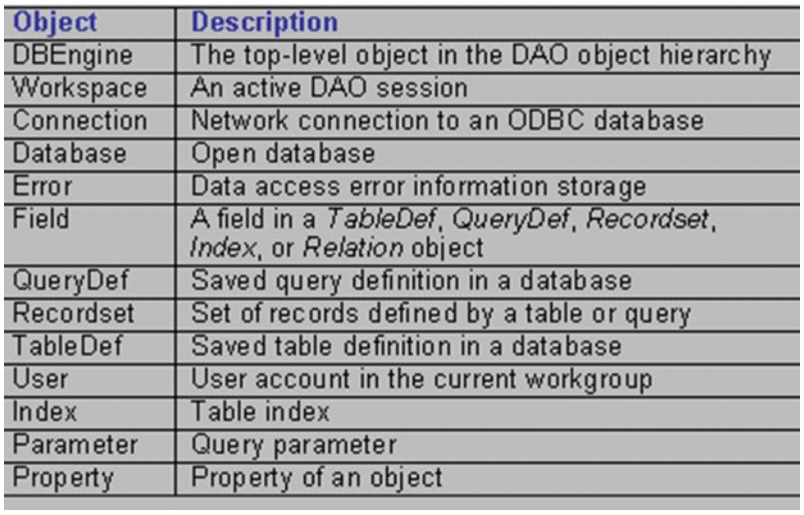
Step 5: Verify the output.

```
Student: [RollNo : 0, Name : Robert ]
Student: [RollNo : 1, Name : John ]
Student: Roll No 0, updated in the database
Student: [RollNo : 0, Name : Michael ]
```

DAO objects

Let us have a look at the DAO objects to understand DAO better.

Table 3.1 *Names and descriptions of common DAO objects*



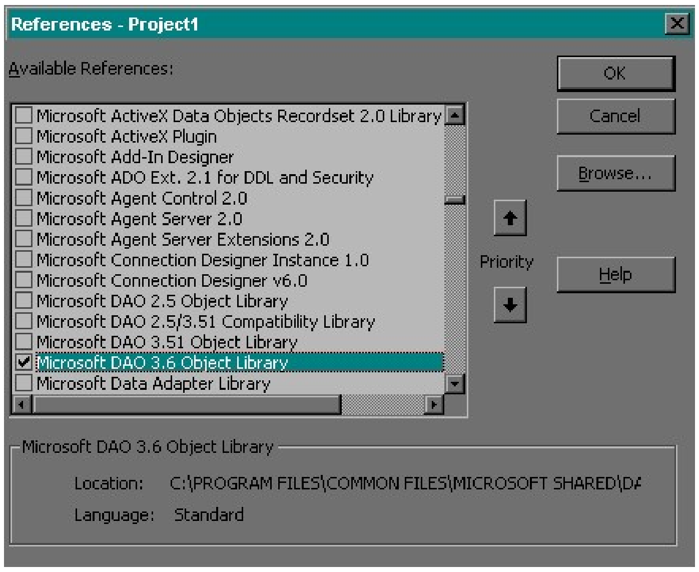
Object	Description
DBEngine	The top-level object in the DAO object hierarchy
Workspace	An active DAO session
Connection	Network connection to an ODBC database
Database	Open database
Error	Data access error information storage
Field	A field in a *TableDef*, *QueryDef*, *Recordset*, *Index*, or *Relation* object
QueryDef	Saved query definition in a database
Recordset	Set of records defined by a table or query
TableDef	Saved table definition in a database
User	User account in the current workgroup
Index	Table index
Parameter	Query parameter
Property	Property of an object

The DBEngine is the highest-level object in the DAO object model. It contains all other objects and collections. The Database object is the member of the Databases collection of the default Workspace object, which is a member of the Workspaces collection of the DBEngine object.

NOTES

Program 3.5: Let's create a simple VB project to access the data stored in Microsoft's sample Northwind database to demonstrate how you might put DAO to work.

1. Open VB and start a new project.
2. Go to Project References and select Microsoft DAO 3.6 Object Library (depending on the version of VB you are using), as shown in Figure given below.


References - Project1

Available References:

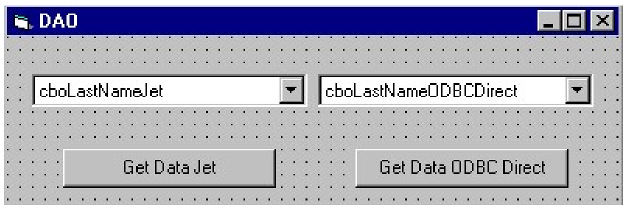
- [ ] Microsoft ActiveX Data Objects Recordset 2.0 Library
- [ ] Microsoft ActiveX Plugin
- [ ] Microsoft Add-In Designer
- [ ] Microsoft ADO Ext. 2.1 for DDL and Security
- [ ] Microsoft Agent Control 2.0
- [ ] Microsoft Agent Server 2.0
- [ ] Microsoft Agent Server Extensions 2.0
- [ ] Microsoft Connection Designer Instance 1.0
- [ ] Microsoft Connection Designer v6.0
- [ ] Microsoft DAO 2.5 Object Library
- [ ] Microsoft DAO 2.5/3.51 Compatibility Library
- [ ] Microsoft DAO 3.51 Object Library
- [✓] Microsoft DAO 3.6 Object Library
- [ ] Microsoft Data Adapter Library

Microsoft DAO 3.6 Object Library

Location: C:\PROGRAM FILES\COMMON FILES\MICROSOFT SHARED\D

Language: Standard

Add two combo boxes (cboLastNameJet and cboLastNameODBCDirect) and two command buttons (cmdGetDataJet and cmdGetDataODBCDirect) to the form as shown below.



Add the following code in to the cmdGetDataJet_Click() event.

```
Private Sub cmdGetDataJet_Click()
 Dim wrkJet As DAO.Workspace
 Dim dbJet As DAO.Database
 Dim rsJet As DAO.Recordset
 Dim strLocation As String
```

```
      'location of the Northwind.mdb database to be used for
Microsoft Jet connection
      strLocation = "D:\Program Files\Microsoft
Office\Office\Samples\"
      'Open Microsoft Jet workspace
      Set wrkJet = CreateWorkspace("", "admin", "", dbUseJet)
      'Open Microsoft Jet database
      Set dbJet = wrkJet.OpenDatabase(strLocation &
"Northwind.mdb")
      'Open Microsoft Jet read-only recordset
      Set rsJet = dbJet.OpenRecordset("SELECT LastName FROM
Employees", dbOpenDynaset, dbReadOnly)
      With cboLastNameJet
      If rsJet.EOF And rsJet.BOF Then
      'no data - disable combo box
      .Enabled = False
      Else
      'clear the combo box
      .Clear
      'move the recordset to the first row
      rsJet.MoveFirst
      Do Until rsJet.EOF
      .AddItem Trim(rsJet("LastName"))
      'move the recordset to the next row
      rsJet.MoveNext
      Loop
      'select the first item in the combo box
      .ListIndex = 0
      'close recordset
      rsJet.Close
      End If
      End With
      'close database
      dbJet.Close
      'close workspace
      wrkJet.Close
      'release objects
      Set rsJet = Nothing
```

```
 Set dbJet = Nothing
 Set wrkJet = Nothing
End Sub
```

Add the following code to the Private SubcmdGetDataODBCDirect _Click() event.

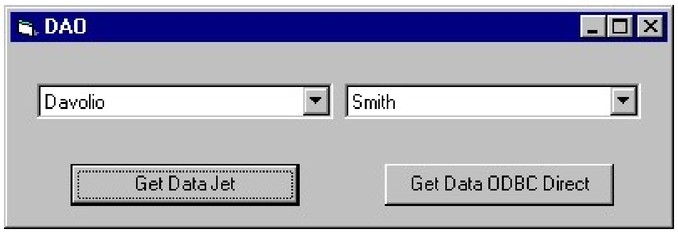
NOTES

```
Private Sub cmdGetDataODBCDirect_Click()
 Dim wrkJet As DAO.Workspace, wrkODBC As DAO.Workspace
 Dim conODBCDirect As DAO.Connection
 Dim rsODBCDirect As DAO.Recordset
 Dim strConn As String
 'connection string for
strConn = "ODBC;DATABASE=employee_records;
UID=admin;PWD=sql;DSN=employee"

 'Open ODBCDirect workspace
 Set wrkODBC = CreateWorkspace("", "admin", "", dbUseODBC)
 'Open ODBCDirect connection
  Set conODBCDirect = wrkODBC.OpenConnection("", , ,
strConn)
 'Open ODBCDirect dynamic recordset
 Set rsODBCDirect = conODBCDirect.OpenRecordset("SELECT
LastName FROM Employees", dbOpenDynamic)
 With cboLastNameODBCDirect
 If rsODBCDirect.EOF And rsODBCDirect.BOF Then
 'no data - disable combo box
 .Enabled = False
 Else
 'clear the combo box
 .Clear
 'move the recordset to the first row
 rsODBCDirect.MoveFirst
 Do Until rsODBCDirect.EOF
 .AddItem Trim(rsODBCDirect("LastName"))
 'move the recordset to the next row
 rsODBCDirect.MoveNext
 Loop
 'select the first item in the combo box
 .ListIndex = 0
```

NOTES

```
 'close recordset
 rsODBCDirect.Close
 End If
 End With
 'close workspace
 wrkODBC.Close
 'release objects
 Set rsODBCDirect = Nothing
 Set wrkODBC = Nothing
 Set conODBCDirect = Nothing
End Sub
```

3. Modify strLocation to reflect the location of the Northwind database on your computer or use another .mdb database and modify Set dbJet = wrkJet.OpenDatabase(strLocation & "Northwind.mdb") to reflect the name of the database.
4. Modify strConn to reflect the DSN name, password, and UID of a remote database.
5. Modify the query in Set rsODBCDirect = conODBCDirect.OpenRecordset ("SELECT LastName FROM Employees", dbOpenDynamic) to reflect the query you'd like to run.
6. Press [Ctrl][F5] to run the project.
7. Click the Get Data Jet button and the Get Data ODBC Direct button to obtain data using Microsoft Jet and ODBCDirect, respectively.
8. You will observe a screen as shown below:



Set DAO properties for DAO objects

Refer to the object in the DAO hierarchy to set a property that is defined by the Access database engine. The easiest and fastest way of doing that is to create object variables that represent the different objects require work with, and refer to the object variables in subsequent steps in your code. Consider the following example code to create a new TableDef object and sets its Name property.

```
Dim dbs As DAO.Database
Dim tdf As DAO.TableDef
```

```
Set dbs = CurrentDb
Set tdf = dbs.CreateTableDef
tdf.Name = "Contacts"
```

The following table provides some guidelines for determining the setting of the Type property.

If the property setting is	The Type property setting should be
A string	dbText
True / False	dbBoolean
An integer	dbInteger

The following table lists some Microsoft Access-defined properties that apply to DAO objects.

DAO object	Microsoft Access-defined properties
Database	AppTitle, AppIcon, StartupShowDBWindow, StartupShowStatusBar, AllowShortcutMenus, AllowFullMenus, AllowBuiltInToolbars, AllowToolbarChanges, AllowBreakIntoCode, AllowSpecialKeys, Replicable, ReplicationConflictFunction
SummaryInfo Container	Title, Subject, Author, Manager, Company, Category, Keywords, Comments, Hyperlink Base (See the Summary tab of the DatabaseName Properties dialog box, available by selecting Database Properties on the File menu.)
UserDefined Container	(See the Summary tab of the DatabaseName Properties dialog box, available by selecting Database Properties on the File menu.)
TableDef	DatasheetBackColor, DatasheetCellsEffect, DatasheetFontHeight, DatasheetFontItalic, DatasheetFontName, DatasheetFontUnderline, DatasheetFontWeight, DatasheetForeColor, DatasheetGridlinesBehavior, DatasheetGridlinesColor, Description, FrozenColumns, RowHeight, ShowGrid
QueryDef	DatasheetBackColor, DatasheetCellsEffect, DatasheetFontHeight, DatasheetFontItalic, DatasheetFontName, DatasheetFontUnderline, DatasheetFontWeight, DatasheetForeColor, DatasheetGridlinesBehavior, DatasheetGridlinesColor, Description, FailOnError, FrozenColumns, LogMessages, MaxRecords, RecordLocks, RowHeight, ShowGrid, UseTransaction
Field	Caption, ColumnHidden, ColumnOrder, ColumnWidth, DecimalPlaces, Description, Format, InputMask

Program 3.6: To create a new Database object and opens an existing Database object in the default Workspace objects. Then it enumerates the Database collection and the Properties collection of each Database object.

```
Sub DatabaseObjectX()
 Dim wrkAcc As Workspace
 Dim dbsNorthwind As Database
 Dim dbsNew As Database
 Dim dbsLoop As Database
 Dim prpLoop As Property
```

NOTES

NOTES

```
Set wrkAcc = CreateWorkspace("AccessWorkspace", "admin", _
 "", dbUseJet)
 ' Make sure there isn't already a file with the name of
 ' the new database.
 If Dir("NewDB.mdb") <> "" Then Kill "NewDB.mdb"

 ' Create a new database with the specified
 ' collating order.
 Set dbsNew = wrkAcc.CreateDatabase("NewDB.mdb", _
 dbLangGeneral)
 Set dbsNorthwind = wrkAcc.OpenDatabase("Northwind.mdb")
 ' Enumerate the Databases collection.
 For Each dbsLoop In wrkAcc.Databases
 With dbsLoop
 Debug.Print "Properties of " & .Name
 ' Enumerate the Properties collection of each
 ' Database object.
 For Each prpLoop In .Properties
 If prpLoop <> "" Then Debug.Print " " & _
 prpLoop.Name & " = " & prpLoop
 Next prpLoop
 End With
 Next dbsLoop
 dbsNew.Close
 dbsNorthwind.Close
 wrkAcc.Close
 End Sub
```

This example uses CreateDatabase to create a new, encrypted Database object.

```
Sub CreateDatabaseX()
 Dim wrkDefault As Workspace
 Dim dbsNew As DATABASE
 Dim prpLoop As Property
 ' Get default Workspace.
 Set wrkDefault = DBEngine.Workspaces(0)
```

```
      ' Make sure there isn't already a file with the name of
the new database.
      If Dir("NewDB.mdb") <> "" Then Kill "NewDB.mdb"
      ' Create a new encrypted database with the specified
      ' collating order.
      Set dbsNew = wrkDefault.CreateDatabase("NewDB.mdb", _
      dbLangGeneral, dbEncrypt)
      With dbsNew
      Debug.Print "Properties of " & .Name
      ' Enumerate the Properties collection of the new
      ' Database object.
      For Each prpLoop In .Properties
      If prpLoop <> "" Then Debug.Print " " & _
      prpLoop.Name & " = " & prpLoop
      Next prpLoop
      End With
      dbsNew.Close
      End Sub
```

Why Use DAO?

Visual Basic programmers highly recommend ADO as their preferred object model for accessing databases. Although ADO is an excellent model with its own unique benefits, in the context of Access databases, it doesn't have the benefit of native database connectivity, which is where DAO has the distinct advantage.

Applications written in other programming languages, such as Visual Basic, Delphi, and the like, must explicitly connect to the data source they intend to manipulate, and they must do so every time they need to manipulate the data or underlying schema. That's because, unlike Access, these applications do not have an inherent connection to the data source. When used in Access, DAO enables you to manipulate data and schema through an implicit connection that Access maintains to whichever Access database engine, ODBC, or ISAM data source it happens to be connected to.

Active Data Objects ADO and OLEDB

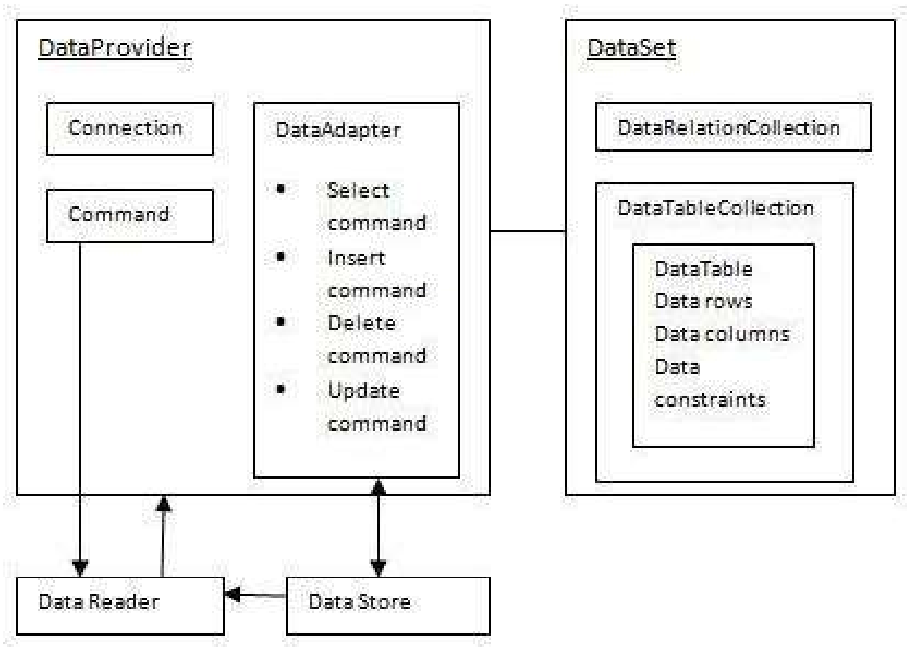
Database Access

Applications access the database to retrieve and display the data stored and also to insert, update and delete data in the database. Microsoft ActiveX Data Objects.NET (ADO.NET) is a model that is used to access and update data from .NET applications.

NOTES

ADO.NET Object Model

It is nothing but the structured process flow through various components. Data provider is used to retrieve the data residing in the database. The object model can be pictorially described as given below:



An application accesses data either through a dataset or a data reader.

1. **Datasets** store data in a disconnected cache and the application retrieves data from it.
2. **Data readers** provide data to the application in a read-only and forward-only mode.

Connecting to a Database

There are two two types of Connection classes in the .NET Framework.

1. **SqlConnection**: It is designed for connecting to Microsoft SQL Server.
2. **OleDbConnection**: It is designed for connecting to a wide range of databases, like Microsoft Access and Oracle.

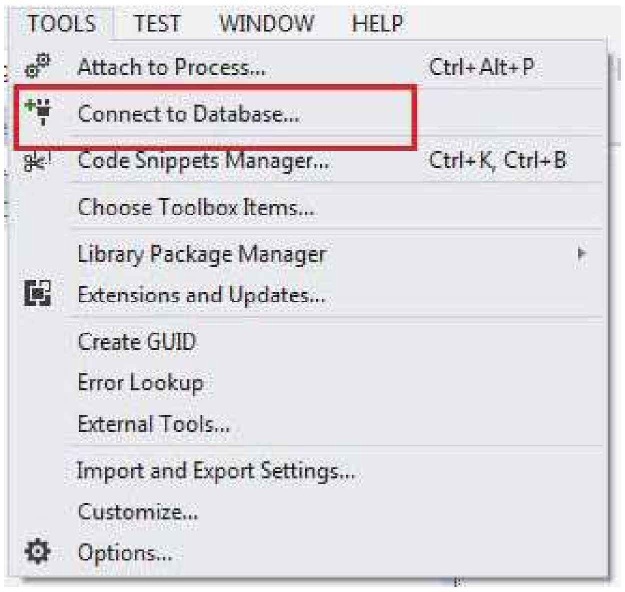
Program 3.7: To illustrate the connection with the database.

The Customers table is stored in Microsoft SQL Server in a database named as testDB.

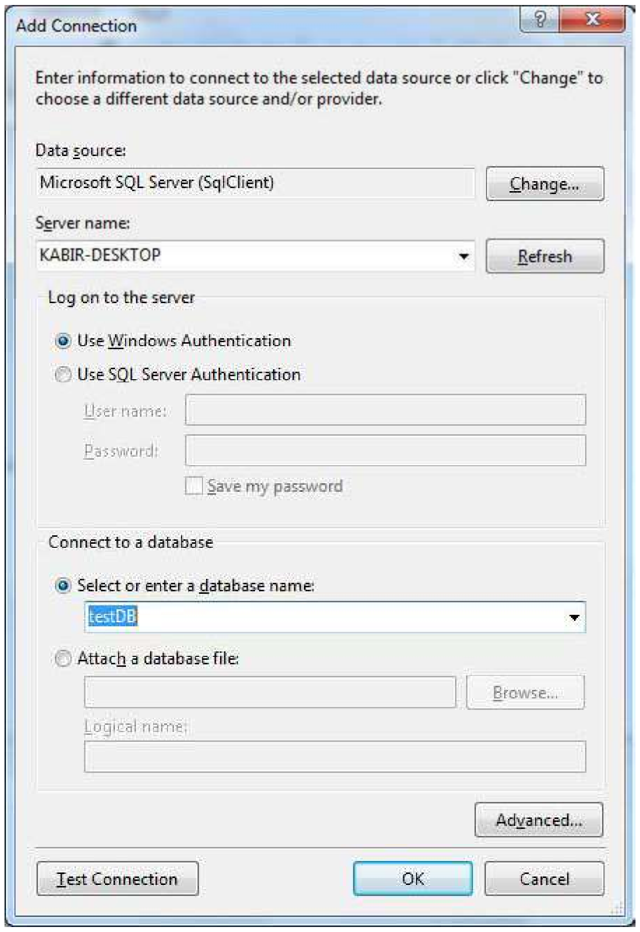
Apply the following steps to connect with the database:

1. Select TOOLS ’! Connect to Database

NOTES



2. Select the server name and database name in the Add Connection dialog box.


Add Connection

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

Data source:
Microsoft SQL Server (SqlClient)
Change...

Server name:
KABIR-DESKTOP
Refresh

Log on to the server
[✓ Use Windows Authentication]
Use SQL Server Authentication
User name:
Password:
Save my password

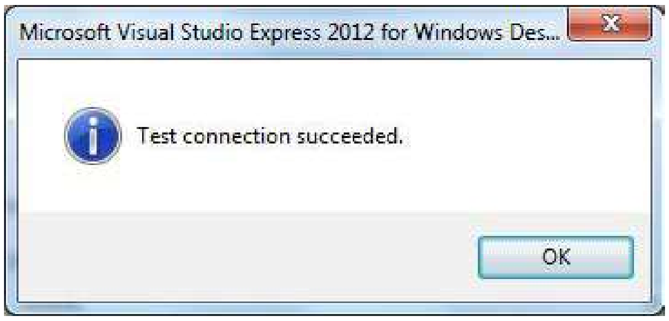
Connect to a database
[✓ Select or enter a database name:]
testDB
Attach a database file:
Browse...
Logical name:

Advanced...

Test Connection
OK
Cancel

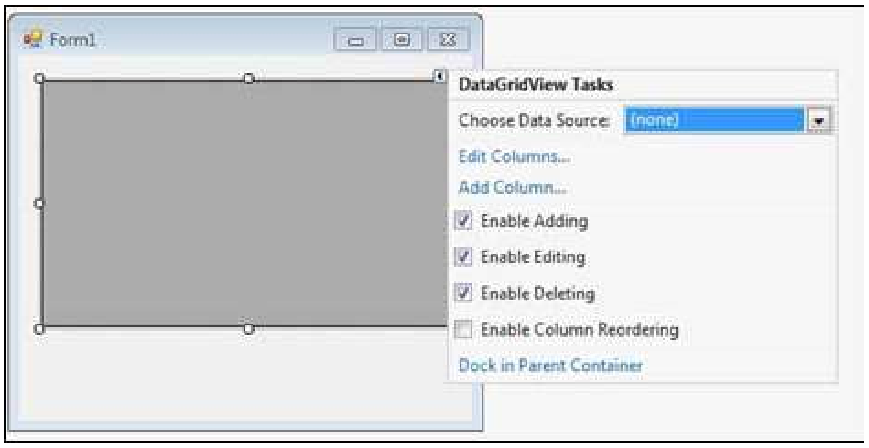
NOTES

3. Click on the Test Connection button to check for the connection succeeded.

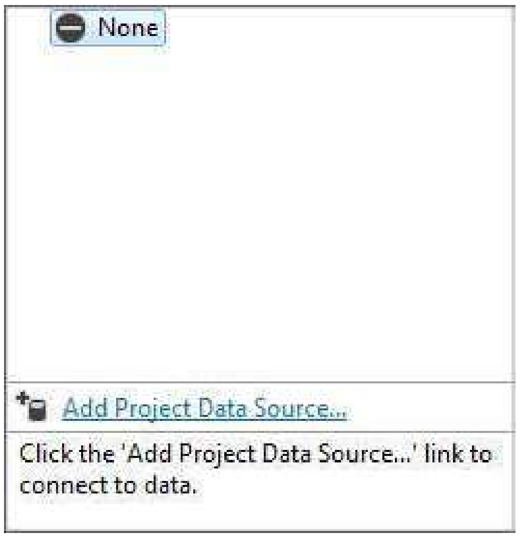

Microsoft Visual Studio Express 2012 for Windows Des...

Test connection succeeded.

4. Add a DataGridView on the form



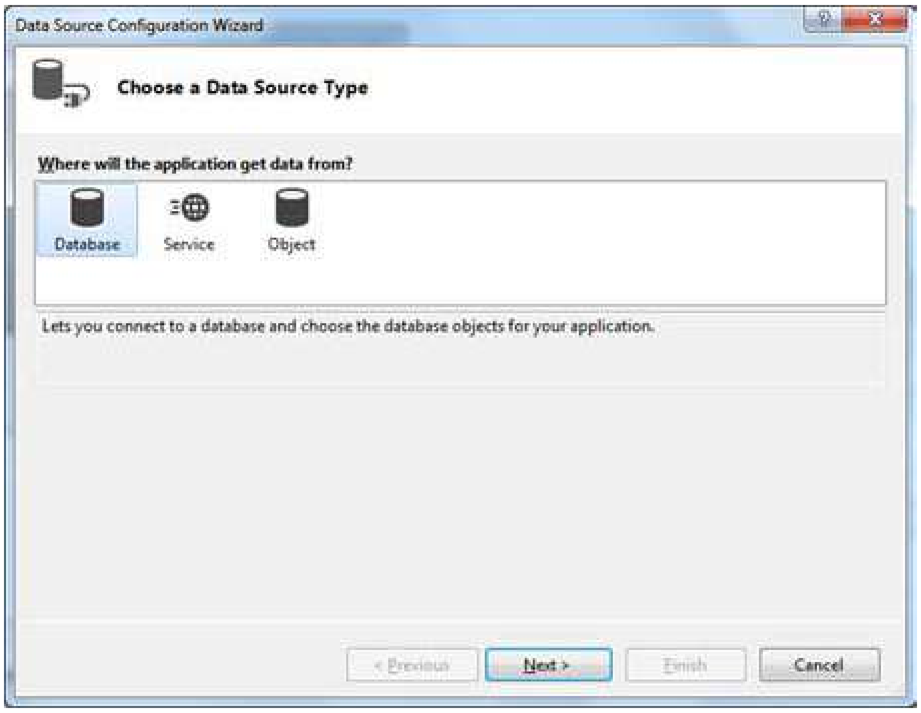
5. Click on the Choose Data Source combo box.
6. Click on the Add Project Data Source link. This will open the Data Source Configuration Wizard.



Click the 'Add Project Data Source...' link to connect to data.

NOTES

7. Select Database as the data source type

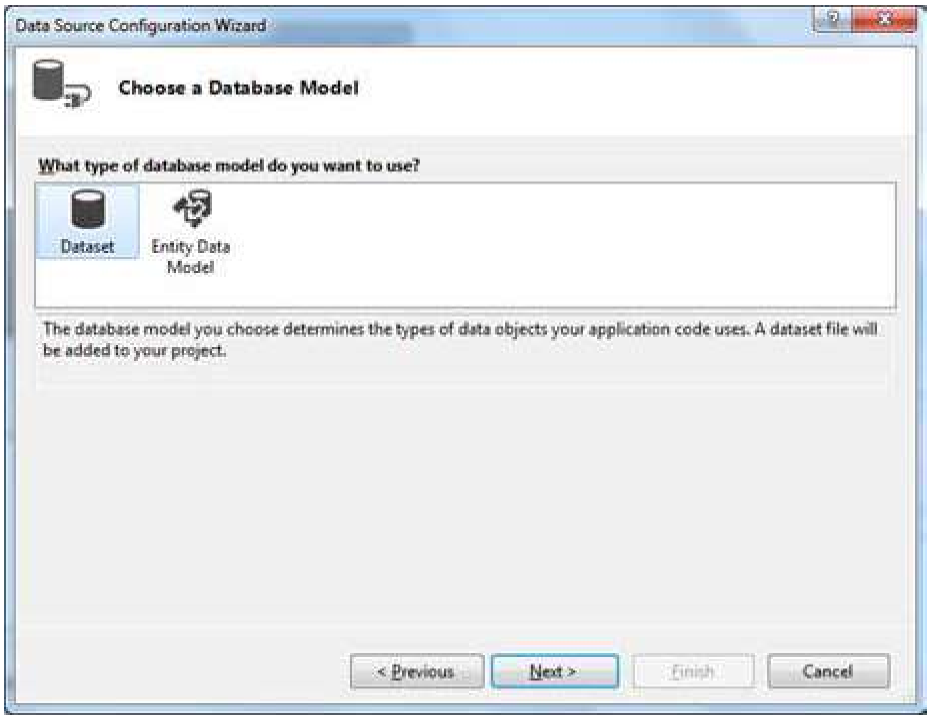


Choose a Data Source Type

Where will the application get data from?

Lets you connect to a database and choose the database objects for your application.

8. Choose DataSet as the database model.



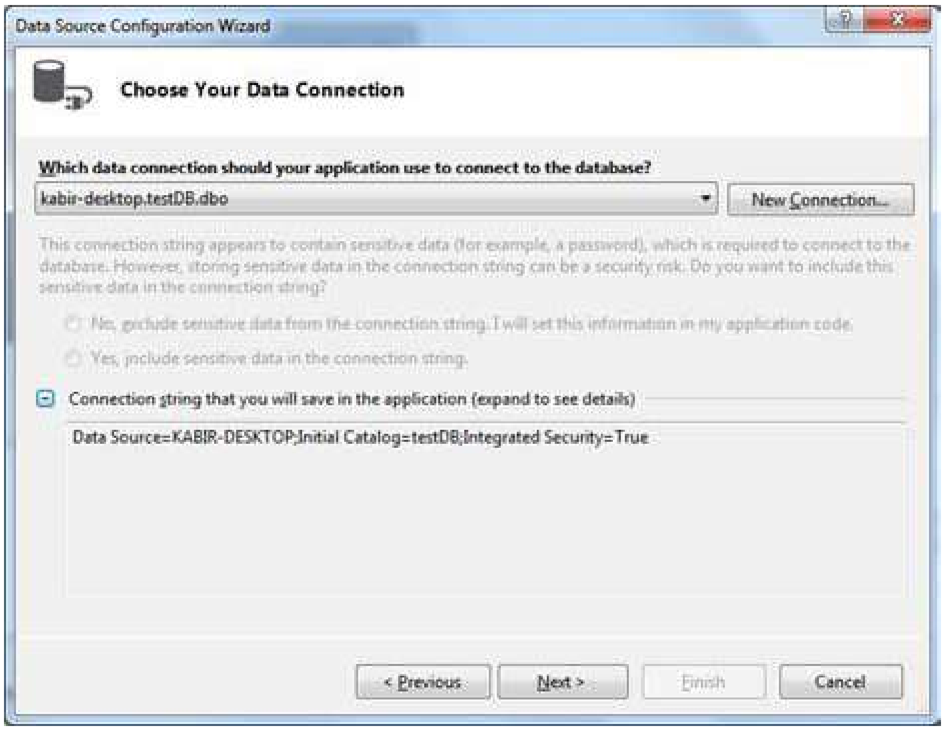
Choose a Database Model

What type of database model do you want to use?

The database model you choose determines the types of data objects your application code uses. A dataset file will be added to your project.

NOTES

9. Choose the connection already set up.



Which data connection should your application use to connect to the database?

kabir-desktop.testDB.dbo

This connection string appears to contain sensitive data (for example, a password), which is required to connect to the database. However, storing sensitive data in the connection string can be a security risk. Do you want to include this sensitive data in the connection string?

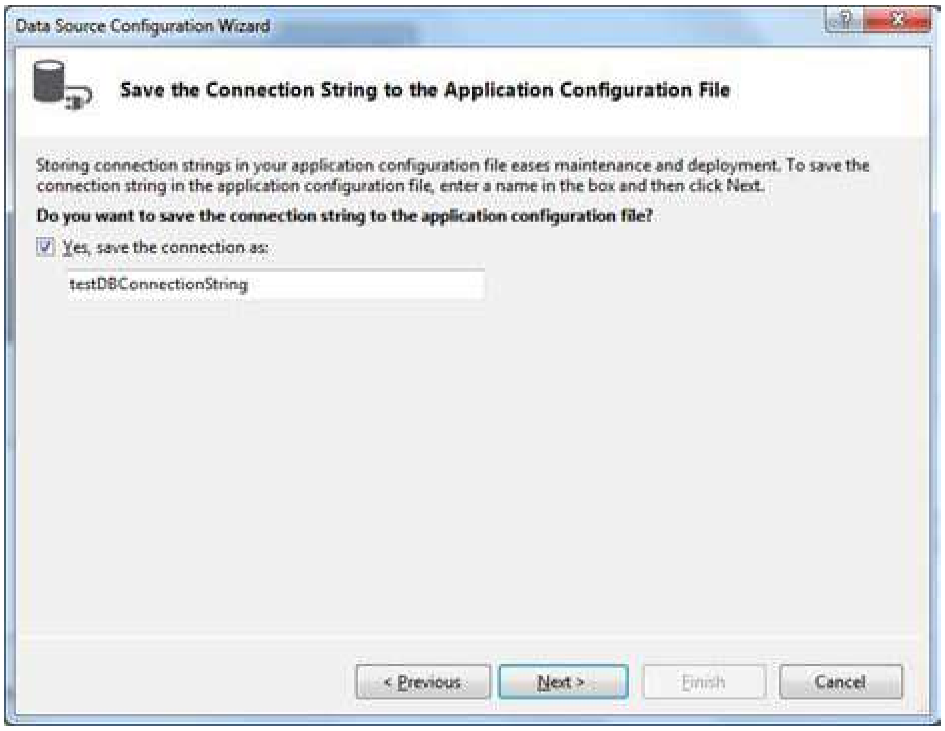
No, exclude sensitive data from the connection string. I will set this information in my application code.

Yes, include sensitive data in the connection string.

Connection string that you will save in the application (expand to see details)

Data Source=KABIR-DESKTOP;Initial Catalog=testDB;Integrated Security=True

10. Save the connection string.


Data Source Configuration Wizard
Save the Connection String to the Application Configuration File
< Previous
Next >
Finish
Cancel

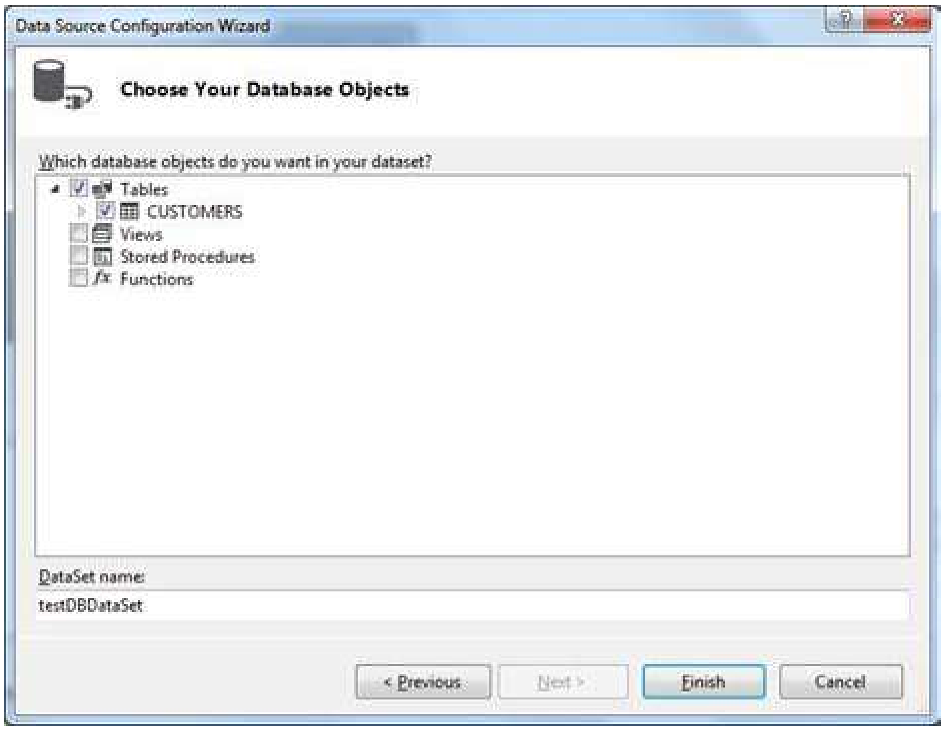
Storing connection strings in your application configuration file eases maintenance and deployment. To save the connection string in the application configuration file, enter a name in the box and then click Next.

Do you want to save the connection string to the application configuration file?

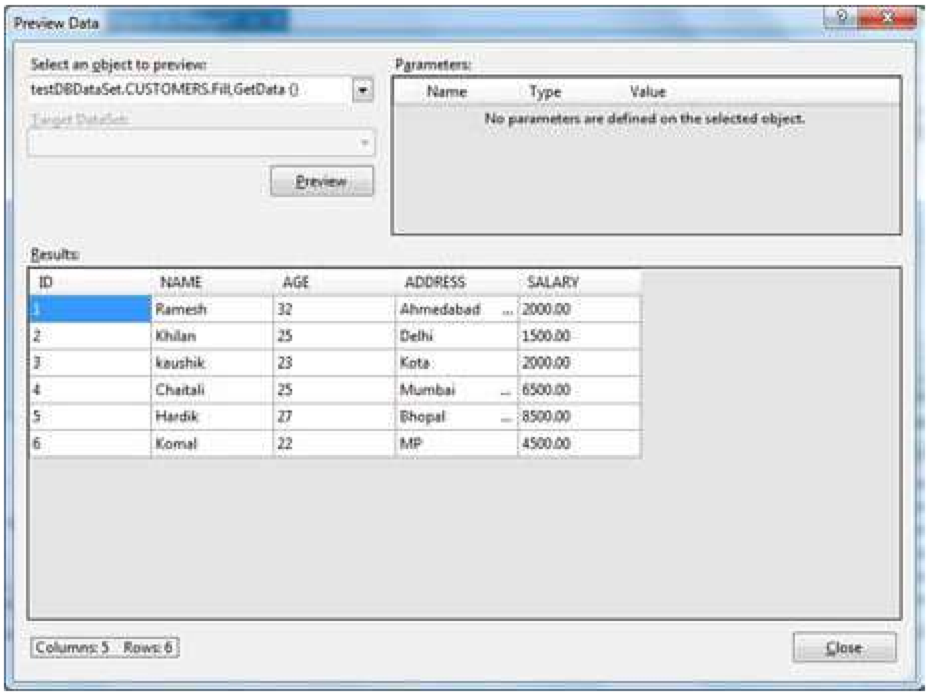
☑ Yes, save the connection as:

testDBConnectionString

11. Choose the database object that is Customers table in our example and then click the Finish button.


Data Source Configuration Wizard
Choose Your Database Objects
Which database objects do you want in your dataset?
Tables
CUSTOMERS
Views
Stored Procedures
Functions
DataSet name:
testDBDataSet
< Previous
Next >
Finish
Cancel

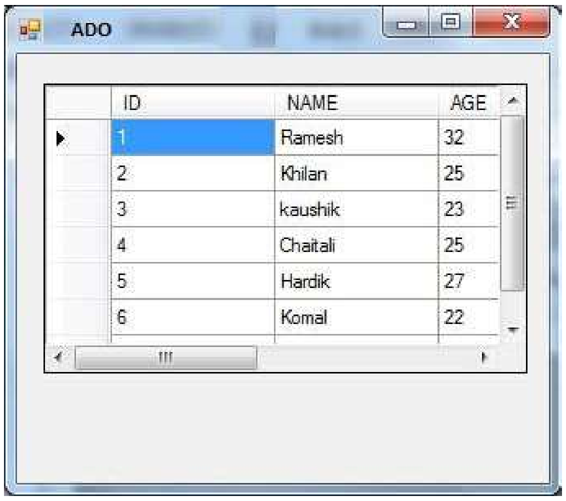
12. Select the Preview Data link to see the data in the Results grid as shown below.


Preview Data
Select an object to preview:
testDBDataSet.CUSTOMERS.Fill,GetData ()
Target DataSet:
Preview
Parameters:
Name
Type
Value
No parameters are defined on the selected object.
Results:
Columns: 5 Rows: 6
Close

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad ...	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai ...	6500.00
5	Hardik	27	Bhopal ...	8500.00
6	Komal	22	MP	4500.00

NOTES

It will produce the following output when you run the application using Start button available at the Microsoft Visual Studio tool bar.

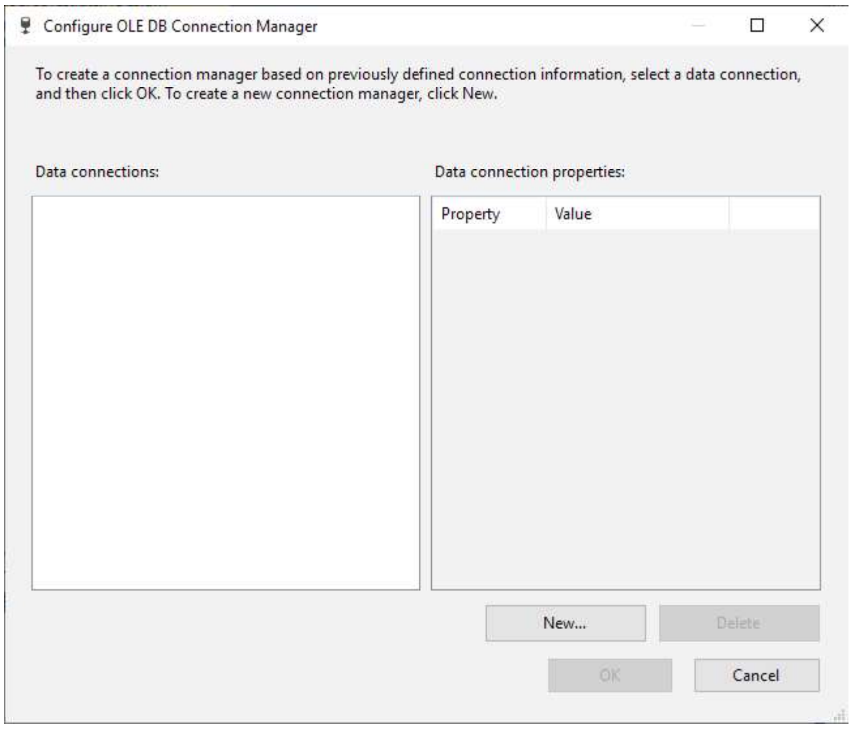


ID	NAME	AGE
1	Ramesh	32
2	Khilan	25
3	kaushik	23
4	Chaitali	25
5	Hardik	27
6	Komal	22

OLE DB Connection Manager

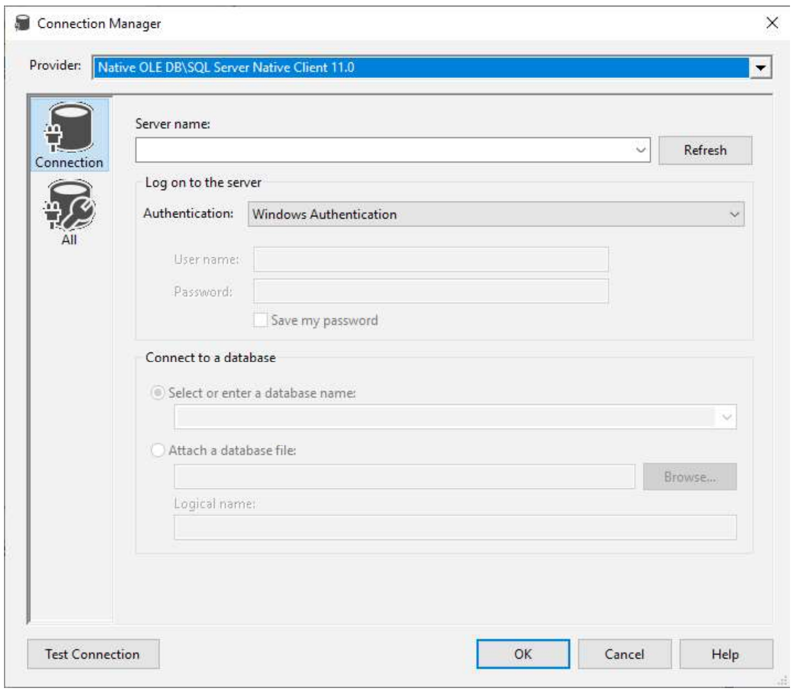
OLE DB (Object **Linking and Embedding, Database)** is an API designed by Microsoft. It allows users to access a variety of data sources in a uniform manner. OLE DB connection managers are the most popular between all SSIS connection managers.

The following window appears, when you click on Add OLE DB connection in the context menu.


Configure OLE DB Connection Manager
Data connections:
Data connection properties:
Property
Value
New...
Delete
OK
Cancel

To create a connection manager based on previously defined connection information, select a data connection, and then click OK. To create a new connection manager, click New.

All previously defined connections are listed with their properties in this window. You have to click on New button to add a new connection. The following screenshot shows the main OLE DB connection configuration form.



Provider: Native OLE DB\SQL Server Native Client 11.0

Server name:

Log on to the server

Authentication: Windows Authentication

User name:

Password:

Save my password

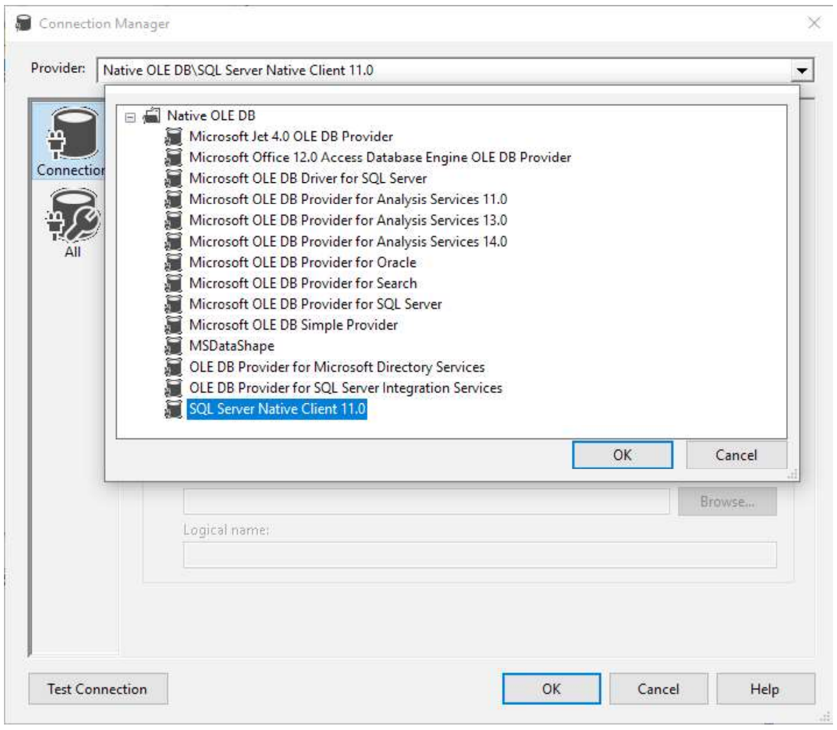
Connect to a database

Select or enter a database name:

Attach a database file:

Logical name:

When, we click on the Provider drop-down list, all available data sources providers will be displayed as shown below:



Provider: Native OLE DB\SQL Server Native Client 11.0

Native OLE DB
- Microsoft Jet 4.0 OLE DB Provider
- Microsoft Office 12.0 Access Database Engine OLE DB Provider
- Microsoft OLE DB Driver for SQL Server
- Microsoft OLE DB Provider for Analysis Services 11.0
- Microsoft OLE DB Provider for Analysis Services 13.0
- Microsoft OLE DB Provider for Analysis Services 14.0
- Microsoft OLE DB Provider for Oracle
- Microsoft OLE DB Provider for Search
- Microsoft OLE DB Provider for SQL Server
- Microsoft OLE DB Simple Provider
- MSDataShape
- OLE DB Provider for Microsoft Directory Services
- OLE DB Provider for SQL Server Integration Services
- SQL Server Native Client 11.0

Logical name:

Following are the main OLE DB connection properties:

1. **Provider:** It is used to connect to the data source.
2. **Server name:** The Server that you want to connect with.
3. **Authentication type:** It includes security parameters used to establish the connection.
4. **Database name:** The database name that we want to connect with.

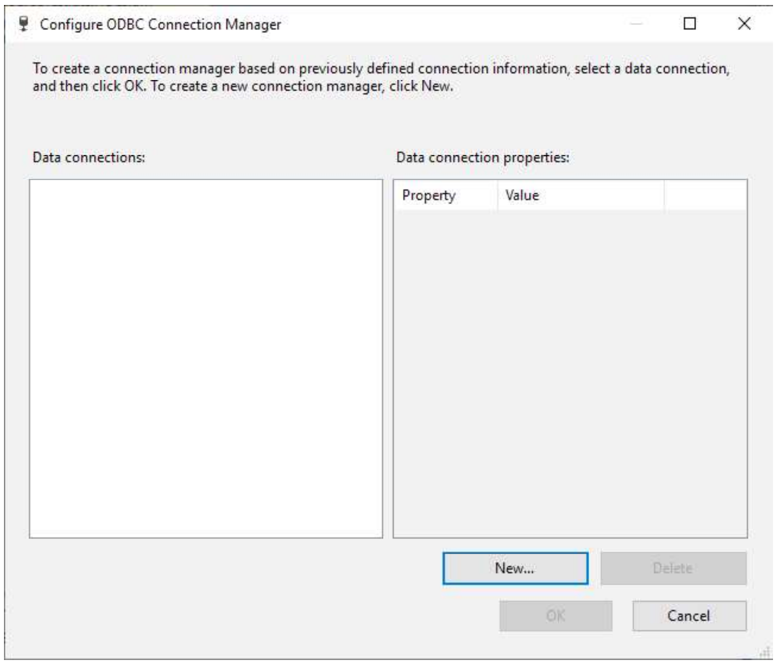
Generally, OLE DB connection manager is used in all tasks and components used to connect to an external database such as:

- Execute SQL Task
- Execute T-SQL Task
- OLE DB Source
- OLE DB Destination
- OLE DB command
- Look up Transformation
- ODBC connection manager

ODBC (Open Database Connectivity) is a standard API used to access database.It provides access only to relational databases which are used by OLE DB to access SQL-based data sources.

ODBC SSIS connection managers are also popular which are used when data sources are defined as DSN (Database Source Name) in the operating system.

Right-click inside the connection manager tab panel to add an ODBC connection manager. Click on **New Connection** button as shown below:



Configure ODBC Connection Manager

To create a connection manager based on previously defined connection information, select a data connection, and then click OK. To create a new connection manager, click New.

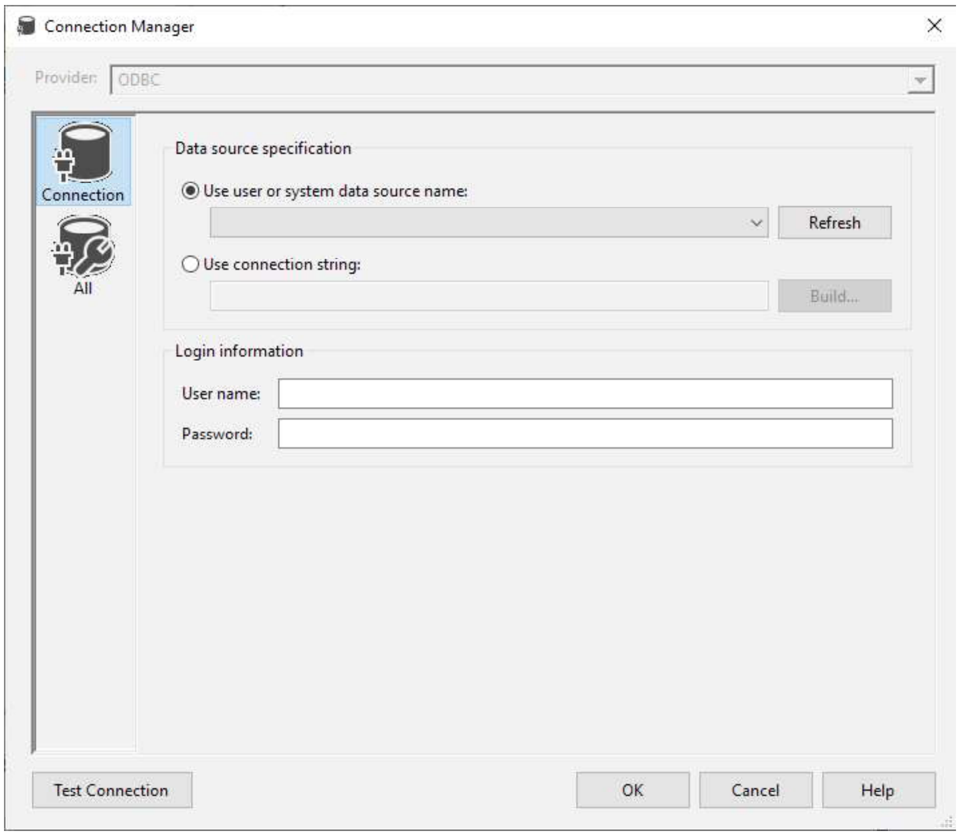
Data connections:

Data connection properties:

Property	Value

NOTES

This form contains all ODBC connections added earlier. Click on New button to add a new one. The following screenshot shows the ODBC connection manager configuration form.


Connection Manager

Provider: ODBC

Data source specification

(●) Use user or system data source name:

Refresh

( ) Use connection string:

Build...

Login information

User name:

Password:

When, we click on the Provider drop-down list, all available data sources providers will be displayed as shown below.

Connection should be closed and release the resources once the use of database is over. The Close() method is used to close the Database Connection. It rolls back any pending transactions and releases the connection from the database connected by the ODBC Data Provider. The code for ADO.NET ODBC connection will be:

```
      Imports System.Data.Odbc
      Public Class Form1
       Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button1.Click
       Dim connetionString As String
       Dim cnn As OdbcConnection
        connetionString = "Driver={Microsoft Access Driver
(*.mdb)};DBQ=yourdatabasename.mdb;"
       cnn = New OdbcConnection(connetionString)
       Try
```

```
 cnn.Open()
 MsgBox("Connection Open ! ")
 cnn.Close()
 Catch ex As Exception
 MsgBox("Can not open connection ! ")
 End Try
 End Sub
End Class
```

BLOCK 4

This block will cover the following topics:

1. Connect with the database
2. Using SQL server and DataReaders
3. Retrieving, inserting, updating and deleting the records in the database

Connecting Databases Using ADO.NET in VB.NET

Connecting and communicating with a database is a necessary part of any type of application. In other words, you can say an application requires accessing the database. ADO.NET (ActiveX Database Objects.NET) is a model provided by the .NET framework that helps in retrieving, inserting, updating, or deleting data from a database.

VB .NET uses ADO .NET (Active X Data Object) for data access and manipulation protocol that also helps us to work with data on the Internet.

ADO.NET Data Architecture

Connection

It is used for establishing a connection between database and application. **SqlConnection** class is used for the **MS-SQL** database. **OleDBConnectionclass** is used for a database like an oracle and MS-Access.

Command

It is used to execute a command (Query). **SqlCommand** class is used for the MS-SQL database. **OleDBCommand** class is used for a database like an oracle and MS-Access.

DataSet

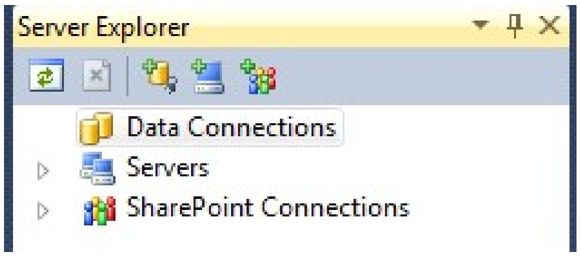
It provides a copy of the original database tables.

DataAdapter

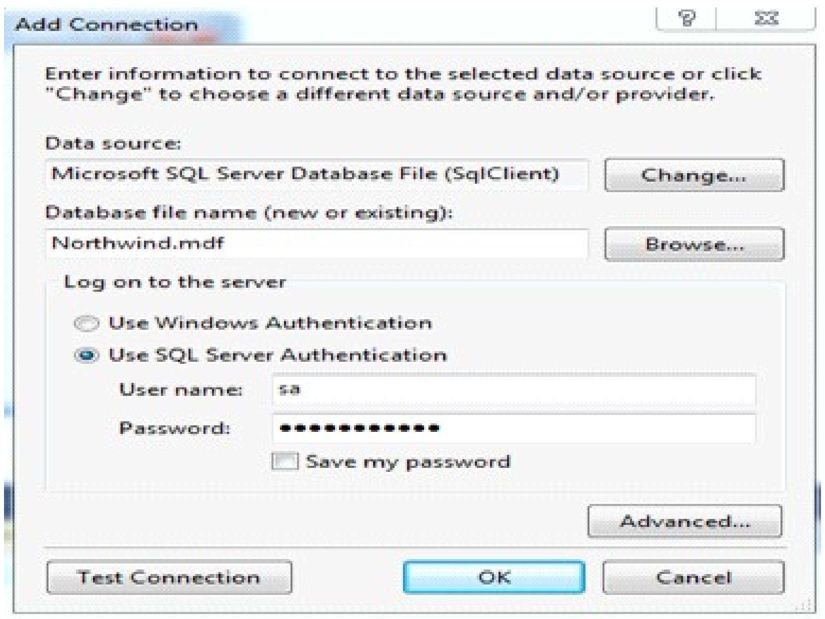
It is used to retrieve data from the database and update **DataSet**. **SqlDataAdapter** class is used for the **MS-SQL** database. **OleDBDataAdapter** class is used for a database like an **oracle** and **MS-Access**.

Data Access with Server Explorer

VB allows us to work with db in two ways i.e. visually and code. Server Explorer enables us to work with connections across different data sources visually. The window that is displayed is the Server Explorer that helps us to create and examine data connections. Server Explorer can be viewed by selecting View'!Server Explorer from the main menu or by pressing Ctrl+Alt+S on the keyboard as shown below.



We will work with SQL Server, the default provider for .NET. We'll be displaying data from Customers table in sample Northwind database in SQL Server. It requires establishing a connection to this database. You need to right-click on the Data Connections icon in Server Explorer and select Add Connection item that opens the Data Link Properties dialog. It allows us to enter the name of the server with which we want to work along with login name and password as shown below.



Add Connection

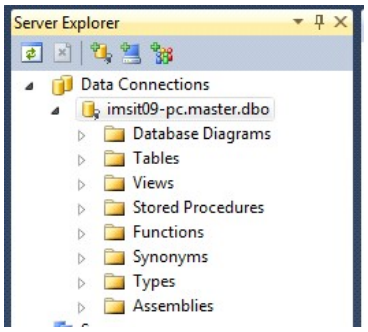
Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source:
Microsoft SQL Server Database File (SqlClient)

Database file name (new or existing):
Northwind.mdf

Log on to the server
Use Windows Authentication
[✓ Use SQL Server Authentication]
User name: sa
Password: ••••••••••••
Save my password

Now select Northwind database from the list. After that click on the Test Connection tab, If the connection is successful, a message "Test Connection Succeeded" is displayed. Click OK and close the Data Link Properties or add connection when connection to the database is set. It displays the Tables, Views and Stored Procedures in that Northwind sample database when you expand the connection node that is ("+" sign). Expanding the Tables node will display all the tables available in the database.



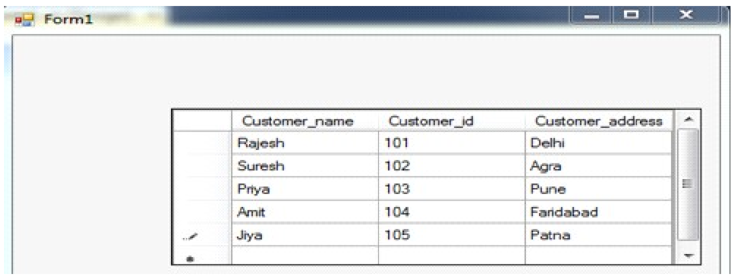
We will work with Customers table to display its data in our example. Now, drag Customers table onto the form from the Server Explorer. It creates SQLConnection1 and SQLDataAdapter1 objects that are the data connection and data adapter objects used to work with data. They are displayed on the component tray. After that, generate the dataset which holds data from the data adapter. To do that select Data'!Generate DataSet from the main menu or rightclick SQLDataAdapter1 object and select generate DataSet menu. It will displays the generate Dataset dialogbox.

Select the radio button with New option to create a new dataset once the dialogbox is displayed. Make sure Customers table is checked and click OK. It adds a dataset, DataSet to the component tray. After that, drag a DataGrid from toolbox to display Customers table. Set the data grid's DataSource property to DataSet and it's DataMember property to Customers. Next, we need to fill the dataset with data from the data adapter. The code for that is given below:

```
Private Sub Form1_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs)_
Handles MyBase.Load
DataSet.Clear()
SqlDataAdapter1.Fill(DataSet)
'filling the dataset with the dataadapter's fill method
End Sub
```

The output of the above code is given below:

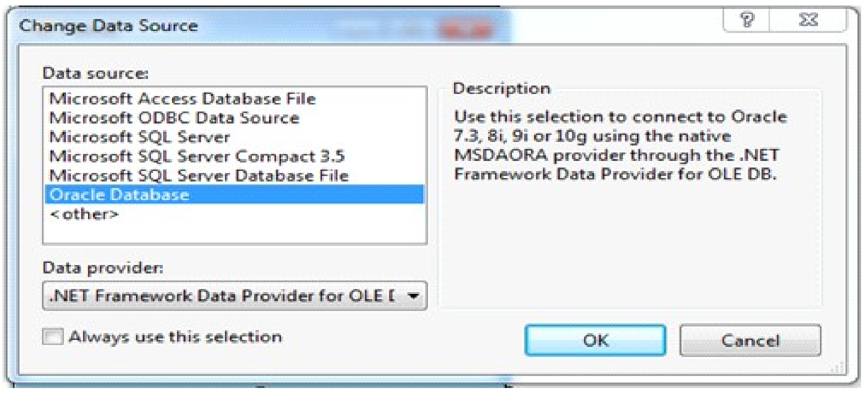
Customers table is displayed in the data grid once the application is executed. That's one of the simplest ways of displaying data using the Server Explorer window.



Customer_name	Customer_id	Customer_address
Rajesh	101	Delhi
Suresh	102	Agra
Priya	103	Pune
Amit	104	Faridabad
Jiya	105	Patna

Microsoft Access and Oracle Database

You need to select Microsoft OLE DB Provider for Oracle from the Provider tab in the DataLink dialog when working with Oracle. The process is same when working with Oracle or MS Access but has some minor changes. It requires appropriate Username and password.


Change Data Source

Data source:
Microsoft Access Database File
Microsoft ODBC Data Source
Microsoft SQL Server
Microsoft SQL Server Compact 3.5
Microsoft SQL Server Database File
Oracle Database
<other>

Description
Use this selection to connect to Oracle 7.3, 8i, 9i or 10g using the native MSDAORA provider through the .NET Framework Data Provider for OLE DB.

Data provider:
.NET Framework Data Provider for OLE [

Always use this selection

Using DataReaders, SQL Server

Here, we will work with ADO .NET objects in code to create connections and read data by using the data reader. The namespace that requires to be imported when working with SQL Connections is System.Data.SqlClient. Here, we will check that how to connect by using our own connection objects. We will also check how to use the command object.

1. Working with SQL Server

The classes used while working with SQL server are discussed below:

(a) **The SqlConnection Class:** This class shows the connection to SQL Server data source. We will use OleDB connection object when working with databases instead of SQL Server. Sqlconnections is 70% faster than OleDb connections.

(b) **The SqlCommand Class:** This class represents a SQL statement or stored procedure for use in a database with SQL Server.

(c) **The SqlDataAdapter Class:** This class represents a bridge between SQL Server database and dataset. It includes the Select, Insert, Update and Delete commands for loading and updating the data.

(d) **The SqlDataReader Class:** This class creates a data reader to be used with SQL Server.

2. DataReaders

A DataReader is a lightweight object which provides forward-only, read-only data in a very efficient and fast way. Data access with DataReader is read-only, if we cannot make any changes (update) to data and forward-only, which means we cannot go back to the previous record which was accessed. A DataReader requires the use of an active connection for the entire time. We can instantiate a DataReader by making a call to a Command object’s ExecuteReader command. When the

DataReader is first returned, it is positioned before the first record of the result set. To make the first record available, we need to call the Read method. If a record is available, then Read method moves the DataReader to next record and returns True. If a record is not available the Read method returns False.

Program 4.1: To retrieve data using Select command (to display data from Discounts table in Pubs sample database).

```
     Imports System.Data.SqlClient
     Public Class Form1 Inherits System.Windows.Forms.Form
     Dim myConnection As SqlConnection
     Dim myCommand As SqlCommand
     Dim dr As New SqlDataReader()
     'declaring the objects
     Private Sub Form1_Load(ByVal sender As System.Object,
ByVal e As
     System.EventArgs)_
     Handles MyBase.Load
     myConnection = New
     SqlConnection("server=localhost;uid=sa;pwd=;database=pubs")
     'establishing connection. you need to provide password
for sql server
     Try
     myConnection.Open()
     'opening the connection
     myCommand = New SqlCommand("Select * from discounts",
     myConnection)
     'executing the command and assigning it to connection
     dr = myCommand.ExecuteReader()
     While dr.Read()
     'reading from the datareader
     MessageBox.Show("discounttype" & dr(0).ToString())
     MessageBox.Show("stor_id" & dr(1).ToString())
     MessageBox.Show("lowqty" & dr(2).ToString())
     MessageBox.Show("highqty" & dr(3).ToString())
     MessageBox.Show("discount" & dr(4).ToString())
     'displaying the data from the table
     End While
     dr.Close()
     myConnection.Close()
```

```
Catch e As Exception
End Try
End Sub
End Class
```

The above code displays records from discounts table in MessageBoxes.

Retrieving records with a Console Application

```
Imports System.Data.SqlClient
Imports System.Console
Module Module1
Dim myConnection As SqlConnection
Dim myCommand As SqlCommand
Dim dr As SqlDataReader
Sub Main()
Try
myConnection = New
SqlConnection("server=localhost;uid=sa;pwd=;database=pubs")
'you need to provide password for sql server
myConnection.Open()
myCommand = New SqlCommand("Select * from discounts",
myConnection)
dr = myCommand.ExecuteReader
Do
While dr.Read()
WriteLine(dr(0))
WriteLine(dr(1))
WriteLine(dr(2))
WriteLine(dr(3))
WriteLine(dr(4))
' writing to console
End While
Loop While dr.NextResult()
Catch
End Try
dr.Close()
myConnection.Close()
End Sub
End Module
```

Inserting Records

Example 4.2: To insert a Record into the Jobs table in Pubs sample database.

```
     Imports System.Data.SqlClient
     Public Class Form2 Inherits System.Windows.Forms.Form
     Dim myConnection As SqlConnection
     Dim myCommand As SqlCommand
     Dim ra as Integer
     'integer holds the number of records inserted
     Private Sub Form2_Load(ByVal sender As System.Object,
ByVal e_
     As System.EventArgs) Handles MyBase.Load
     End Sub
     Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e_As System.EventArgs) Handles Button1.Click
     myConnection  =  New  SqlConnection("server=localhost;
uid=sa;pwd=;database=pubs")
     'you need to provide password for sql server
     myConnection.Open()
     myCommand = New SqlCommand("Insert into Jobs values 12,'IT
     Manager',100,300,_
     myConnection)
     ra=myCommand.ExecuteNonQuery()
     MessageBox.Show("New Row Inserted" & ra)
     myConnection.Close()
     End Sub
     End Class
```

Deleting a Record

Example 4.3: For deleting a record, we will use Authors table in Pubs sample database to work with this code. Drag a button onto the form and place the following code.

```
     Imports System.Data.SqlClient
     Public Class Form3 Inherits System.Windows.Forms.Form
     Dim myConnection As SqlConnection
     Dim myCommand As SqlCommand
     Dim ra as Integer
     Private Sub Form3_Load(ByVal sender As System.Object,
ByVal e_
     As System.EventArgs) Handles MyBase.Load
     End Sub
     Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e_
     As System.EventArgs) Handles Button1.Click
     myConnection = New
     SqlConnection("server=localhost;uid=sa;pwd=;database=pubs")
     'you need to provide password for sql server
     myConnection.Open()
     myCommand = New SqlCommand("Delete from Authors where
     city='Oakland'",_
     myConnection)
     'since no value is returned we use ExecuteNonQuery
     ra=myCommand.ExecuteNonQuery()
     MessageBox.Show("Records affected" & ra)
     myConnection.Close()
     End Sub
     End Class
```

NOTES

Updating Records

Example 4.4: For updating a record, we will update a row in Authors table. Drag a button onto the form and write the following code.

```
     Imports System.Data.SqlClient
     Public Class Form4 Inherits System.Windows.Forms.Form
     Dim myConnection As SqlConnection
     Dim myCommand As SqlCommand
     Dim ra as Integer
     Private Sub Form4_Load(ByVal sender As System.Object,
ByVal e_
     As System.EventArgs) Handles MyBase.Load
     End Sub
     Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e_
     As System.EventArgs) Handles Button1.Click
     myConnection = New
     SqlConnection("server=localhost;uid=sa;pwd=;database=pubs")
     'you need to provide password for sql server
     myConnection.Open()
     myCommand  =  New  SqlCommand("Update  Authors  Set
city='Oakland'
     where city=_
     'San Jose' ",myConnection)
```

NOTES

```
    ra=myCommand.ExecuteNonQuery()
    MessageBox.Show("Records affected" & ra)
    myConnection.Close()
    End Sub
    End Class
```

Using OleDb Provider

The classes of the OleDb provider with which we work are as follows:

1. **The OleDbConnection Class:** The OleDbConnection class allows a connection to OleDb data source. OleDbconnections are used to connect to most databases.
2. **The OleDbCommand Class:** The OleDbCommand class shows a SQL statement or stored procedure which is to be executed in a database by an OLEDB provider.
3. **The OleDbDataAdapter Class:** The OleDbDataAdapter class represents as an intermediate between OleDb data source and datasets. We use the Select, Insert, Delete and Update commands for loading and updating the data.
4. **The OleDbDataReader Class:** The OleDbDataReader class creates a datareader for use with an OleDb data provider. 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. It is used to read a row of data from the database.

Program 4.5: To retrieve the records. In the code below, we are working with Emp table in Oracle.

```
    Imports System.Data.OleDB
    Public Class Form1 Inherits System.Windows.Forms.Form
    Dim myConnection As OleDbConnection
    Dim myCommand As OleDbCommand
    Dim dr As New OleDbDataReader()
    'declaration
    Private Sub Form1_Load(ByVal sender As System.Object,
ByVal e As
    System.EventArgs)_
    Handles MyBase.Load
    myConnection = New OleDbConnection_
    ("Provider=MSDAORA.1;UserID=scott;password=tiger;
database=ora")
    'MSDORA is the provider when working with Oracle
    Try
```

```
myConnection.Open()
'opening the connection
myCommand = New OleDbCommand("Select * from emp",
myConnection)
'executing the command and assigning it to connection
dr = myCommand.ExecuteReader()
While dr.Read()
'reading from the datareader
MessageBox.Show("EmpNo" & dr(0))
MessageBox.Show("EName" & dr(1))
MessageBox.Show("Job" & dr(2))
MessageBox.Show("Mgr" & dr(3))
MessageBox.Show("HireDate" & dr(4))
'displaying data from the table
End While
dr.Close()
myConnection.Close()
Catch e As Exception
End Try
End Sub
End Class
```

The above code displays first 5 columns from the Emp table in Oracle.

Inserting Records

Program 4.6: Drag a Button from the toolbox onto the Form. When this Button is clicked the values specified in code will be inserted into the Emp table.

```
    Imports System.Data.OleDb
    Public Class Form2 Inherits System.Windows.Forms.Form
    Dim myConnection As OleDbConnection
    Dim myCommand As OleDbCommand
    Dim ra as Integer
    'integer holds the number of records inserted
    Private Sub Form2_Load(ByVal sender As System.Object,
ByVal e As_
    System.EventArgs) Handles MyBase.Load
    End Sub
    Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As _
```

NOTES

```
     System.EventArgs) Handles Button1.Click
     myConnection = New
     OleDbConnection("”Provider=MSDAORA.1;User_
ID=scott;password=tiger;database=ora”
     )
     Try
     myConnection.Open()
     myCommand = New OleDbCommand("Insert into emp values
12,’Ben’,’Salesman’,300
     12-10-2001,3000,500,10 ", myConnection)
     ‘emp table has 8 columns. You can work only with the
columns you want
     ra=myCommand.ExecuteNonQuery()
     MessageBox.Show("Records Inserted” & ra)
     myConnection.Close()
     Catch
     End Try
     End Sub
     End Class
```

Updating Records

Program 4.7: Drag a Button on a new form and write the following code.

```
     Imports System.Data.OleDb
     Public Class Form4 Inherits System.Windows.Forms.Form
     Dim myConnection As OleDbConnection
     Dim myCommand As OleDbCommand
     Dim ra as Integer
     Private Sub Form4_Load(ByVal sender As System.Object,
ByVal e As_
     System.EventArgs) Handles MyBase.Load
     End Sub
     Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e_
     As System.EventArgs) Handles Button1.Click
     Try
     myConnection = New OleDbConnection("”Provider=
MSDAORA.1;User_
     ID=scott;password=tiger;database=ora”)
     myConnection.Open()
     myCommand = New OleDbCommand("Update emp Set DeptNo=65
     where DeptNo=793410",_ myConnection)
     ra=myCommand.ExecuteNonQuery()
     MessageBox.Show("Records Updated" & ra)
     myConnection.Close()
     Catch
     End Try
     End Sub
     End Class
```

Deleting Records

Program 4.8: Drag a Button on a new form and write the following code.

```
     Imports System.Data.OleDb
     Public Class Form3 Inherits System.Windows.Forms.Form
     Dim myConnection As OleDbConnection
     Dim myCommand As OleDbCommand
     Dim ra as Integer
     Private Sub Form3_Load(ByVal sender As System.Object,
ByVal e As_
     System.EventArgs) Handles MyBase.Load
     End Sub
     Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e_
     As System.EventArgs) Handles Button1.Click
     Try
     myConnection = New OleDbConnection(""Provider=MSDAORA.
1;User_
     ID=scott;password=tiger;database=ora")
     myConnection.Open()
     myCommand = New OleDbCommand("Delete from emp where
     DeptNo=790220",_
     myConnection)
     ra=myCommand.ExecuteNonQuery()
     MessageBox.Show("Records Deleted" & ra)
     myConnection.Close()
     Catch
     End Try
     End Sub
     End Class
```

Data Access using MSAccess

Program 4.9: In this program, create a database named Emp in Microsoft Access in the C drive of your computer. In the Emp database, create a table, Table1 with EmpNo, EName and Department as columns, insert some values in the table and close it. Drag three TextBoxes and a Button. The following code will assume that TextBox1 is for EmpNo, TextBox2 is for EName and TextBox3 is for Department. Our intention is to retrieve data from Table1 in the Emp Database and display the values in these TextBoxes without binding, when the Button is clicked.

```
     Imports System.Data.OleDb
     Public Class Form1 Inherits System.Windows.Forms.Form
     Dim cn As OleDbConnection
     Dim cmd As OleDbCommand
     Dim dr As OleDbDataReader
     Private Sub Form1_Load(ByVal sender As System.Object,
ByVal e as _
     System.EventArgs) Handles MyBase.Load
     End Sub
     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;_
     Data Source=C:\emp.mdb;")
     'provider to be used when working with access database
     cn.Open()
     cmd = New OleDbCommand("select * 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
     End Try
     dr.Close()
     cn.Close()
     End Sub
     End Class
```

When you run the code and click the Button, records from Table1 of the Emp database will be displayed in the TextBoxes.

Program 4.10: Write a code for retrieving records with a Console Application.

```
Imports System.Data.OleDb
Imports System.Console
Module Module1
Dim cn As OleDbConnection
Dim cmd As OleDbCommand
Dim dr As OleDbDataReader
Sub Main()
Try
cn = New OleDbConnection("Provider=Microsoft.
Jet.OLEDB.4.0;Data
Source=C:\emp.mdb;_
Persist Security Info=False")
cn.Open()
cmd = New OleDbCommand("select * from table1", cn)
dr = cmd.ExecuteReader
While dr.Read()
WriteLine(dr(0))
WriteLine(dr(1))
WriteLine(dr(2))
'writing to console
End While
Catch
End Try
dr.Close()
cn.Close()
End Sub
End Module
```

Code for Inserting a Record

```
Imports System.Data.OleDb
Public Class Form2 Inherits System.Windows.Forms.Form
Dim cn As OleDbConnection
Dim cmd As OleDbCommand
Dim dr As OleDbDataReader
Dim icount As Integer
Dim str As String
```

NOTES

```
     Private Sub Form2_Load(ByVal sender As System.Object,
ByVal e As_
     System.EventArgs) Handles MyBase.Load
     End Sub
     Private Sub Button1_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:\emp.mdb;")
     cn.Open()
     str = "insert into table1 values(" & CInt(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
     MessageBox.Show(icount)
     'displays number of records inserted
     Catch
     End Try
     cn.Close()
     End Sub
     End Class
```

BLOCK 5
SIMPLE APPLICATION DEVELOPMENT

This block will cover the development of following simple applications:

1. Library information system
2. Students marksheet processing
3. Telephone directory maintenance
4. Gas booking and delivering
5. Electricity bill processing
6. Bank Transaction

7. Pay roll processing
8. Personal information system
9. Question database and conducting Quiz
10. Personal diary

NOTES

1. Library Information System

Add Books:

```
     Public Class AddBooks
      Public NameFrm, NameTo As String
      Private Sub Button9_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button9.Click
      Me.Close()
      End Sub

      Private Sub AddBooks_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
      Call generateyear()
      Call disablethem()
      Call readData()
      Call GroupID_Combo()
      End Sub
      Sub GroupID_Combo()
      Try
       If objcon.State = ConnectionState.Closed Then
objcon.Open()
      com = New OleDb.OleDbCommand("Select GroupID from GroupD",
objcon)
      dr = com.ExecuteReader
      While dr.Read
      ComboBox1.Items.Add(dr.Item(0))
      End While
      dr.Close()
      objcon.Close()
      Catch ex As Exception

      End Try
      End Sub
      Sub generateyear()
      Dim YearNow As Integer
```

NOTES

```
      YearNow = Int(My.Computer.Clock.LocalTime.Year.ToString)
      Dim a, b, c As Integer
      a = YearNow - 5
      b = YearNow
      For c = a To b
      ComboBox2.Items.Add(c)
      Next
      End Sub
      Private Sub ComboBox1_LostFocus(ByVal sender As Object,
ByVal e As System.EventArgs) Handles ComboBox1.LostFocus
      ComboBox1.Text = ComboBox1.Text.ToUpper()
      End Sub
      Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button1.Click
      ComboBox3.Text = "Available"
      Call enablethem()
     End Sub
      Private Sub TextBox2_LostFocus(ByVal sender As Object,
ByVal e As System.EventArgs) Handles TextBox2.LostFocus
      NameFrm = TextBox2.Text
      Call Sentence()
      TextBox2.Text = NameTo
      End Sub
      Sub disablethem()
      'TextBox1.Enabled = False
      TextBox2.Enabled = False
      TextBox3.Enabled = False
      ComboBox1.Enabled = False
      TextBox4.Enabled = False
      TextBox5.Enabled = False
      TextBox6.Enabled = False
      ComboBox2.Enabled = False
      ComboBox3.Enabled = False
      End Sub
      Sub enablethem()
      TextBox1.Enabled = True
      TextBox2.Enabled = True
      TextBox3.Enabled = True
```

NOTES

```
ComboBox1.Enabled = True
TextBox4.Enabled = True
TextBox5.Enabled = True
TextBox6.Enabled = True
ComboBox2.Enabled = True
ComboBox3.Enabled = True
TextBox1.Clear()
TextBox2.Clear()
TextBox3.Clear()
TextBox4.Clear()
TextBox5.Clear()
TextBox6.Clear()
ComboBox1.Text = "”
ComboBox2.Text = "”
ComboBox3.Text = "”
End Sub
Sub Sentence()
Dim a, b As Integer
a = NameFrm.Length
NameTo = "”
For b = 0 To a - 1
If b = 0 Then
If Char.IsLower(NameFrm(0)) Then
NameTo = Char.ToUpper(NameFrm(0))
Else
NameTo = NameFrm(0)
End If
Else
If NameFrm(b - 1) = " " Then
NameTo = NameTo + Char.ToUpper(NameFrm(b))
Else
NameTo = NameTo + NameFrm(b)
End If
End If
Next
End Sub
Private Sub TextBox3_LostFocus(ByVal sender As Object,
ByVal e As System.EventArgs) Handles TextBox3.LostFocus
```

```
      NameFrm = TextBox3.Text
      Call Sentence()
      TextBox3.Text = NameTo
      End Sub
       Private Sub TextBox3_TextChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
TextBox3.TextChanged
      End Sub
      Private Sub TextBox4_LostFocus(ByVal sender As Object,
ByVal e As System.EventArgs) Handles TextBox4.LostFocus
      NameFrm = TextBox4.Text
      Call Sentence()
      TextBox4.Text = NameTo
      End Sub
       Private Sub TextBox4_TextChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
TextBox4.TextChanged
     End Sub
      Private Sub TextBox5_LostFocus(ByVal sender As Object,
ByVal e As System.EventArgs) Handles TextBox5.LostFocus
      NameFrm = TextBox5.Text
      Call Sentence()
      TextBox5.Text = NameTo
      End Sub
       Private Sub TextBox5_TextChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
TextBox5.TextChanged
      End Sub
      Private Sub Button2_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button2.Click
      If TextBox1.Text = “” Then
      MsgBox(“Please enter the Book ID!”, 0, “”)
      Else
      Try
     If objcon.State = ConnectionState.Closed Then objcon.Open()
      com = New OleDb.OleDbCommand(“INSERT INTO Books VALUES(‘“
& TextBox1.Text & “‘,’” & ComboBox1.Text & “‘,’” & TextBox2.Text
& “‘,’” & TextBox3.Text & “‘,’” & TextBox4.Text & “‘,’” &
ComboBox2.Text & “‘,’” & TextBox5.Text & “‘,’” & TextBox6.Text
& “‘,’” & ComboBox3.Text & “‘)”, objcon)
      com.ExecuteNonQuery()
      Call readData()
```

NOTES

```
          MsgBox("Saved successfully", 0, "SUCCESS")
          objcon.Close()
          Catch ex As Exception
          MsgBox(ex.Message, 0, "")
          End Try
          End If
          End Sub
          Sub readData()
          ListView1.Clear()
         ListView1.Columns.Add("BOOK    ID",    90,
HorizontalAlignment.Center)
         ListView1.Columns.Add("GROUP    ID",    90,
HorizontalAlignment.Center)
         ListView1.Columns.Add("BOOK   NAME",   310,
HorizontalAlignment.Center)
         ListView1.Columns.Add("PUBLISHER",      90,
HorizontalAlignment.Center)
         ListView1.Columns.Add("AUTHOR",         90,
HorizontalAlignment.Center)
         ListView1.Columns.Add("PUBLISHING YEAR", 130,
HorizontalAlignment.Center)
         ListView1.Columns.Add("EDITION",        90,
HorizontalAlignment.Center)
         ListView1.Columns.Add("PRICE",          90,
HorizontalAlignment.Center)
         ListView1.Columns.Add("STATUS",         90,
HorizontalAlignment.Center)
          ListView1.View = View.Details
          Try
           If  (objcon.State  =  ConnectionState.Closed)  Then
objcon.Open()
          com = New OleDb.OleDbCommand("SELECT * FROM Books ",
objcon)
          dr = com.ExecuteReader
          While dr.Read()
          Call adddatatolistview(ListView1, dr(0), dr(1), dr(2),
dr(3), dr(4), dr(5), dr(6), dr(7), dr(8))
          End While
          dr.Close()
          objcon.Close()
          Catch
```

```
      'MsgBox("Please Refresh", MsgBoxStyle.Information, "")
      End Try
      End Sub
      Public Sub adddatatolistview(ByVal lvw As ListView, ByVal
BookID As String, ByVal GroupID As String, ByVal BookName As
String, ByVal Publisher As String, ByVal Author As String,
ByVal PubYear As String, ByVal edi As String, ByVal pric As
String, ByVal st As String)
      Dim lv As New ListViewItem
      lvw.Items.Add(lv)
      lv.Text = BookID
      lv.SubItems.Add(GroupID)
      lv.SubItems.Add(BookName)
      lv.SubItems.Add(Publisher)
      lv.SubItems.Add(Author)
      lv.SubItems.Add(PubYear)
      lv.SubItems.Add(edi)
      lv.SubItems.Add(pric)
      lv.SubItems.Add(st)
      End Sub
      Private Sub Button8_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button8.Click
      Try
       If objcon.State = ConnectionState.Closed Then
objcon.Open()
       If MessageBox.Show("Do you really want to delete?",
"ARE YOU SURE", MessageBoxButtons.YesNo) =
Windows.Forms.DialogResult.Yes Then
      com = New OleDb.OleDbCommand("DELETE FROM Books WHERE
BookID='" & TextBox1.Text & "'", objcon)
      com.ExecuteNonQuery()
      objcon.Close()
      MsgBox("Deleted successfully", 0, "SUCCESS")
      End If
      Catch ex As Exception


      End Try
      End Sub
      Sub fill_list()
       com = New OleDb.OleDbCommand("Select * from Books",
objcon)
```

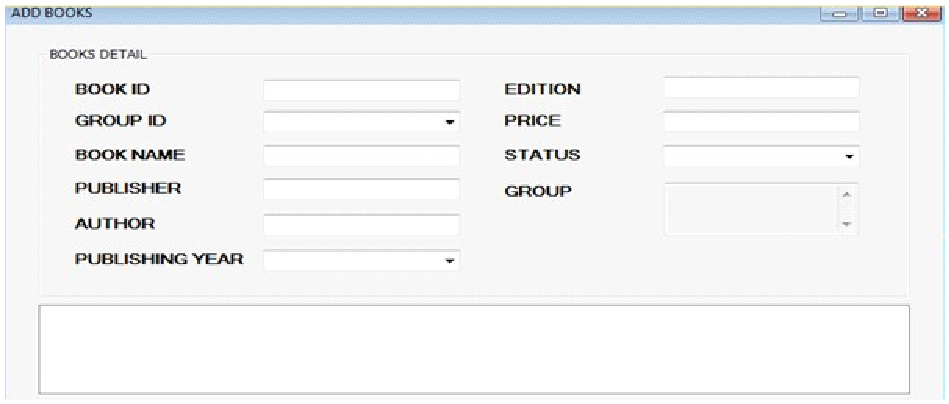
```
      Dim dr As OleDb.OleDbDataReader
      dr = com.ExecuteReader
      dr.Read()
      While (dr.NextResult)
      End While
      End Sub
      Private Sub GroupBox1_Enter(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles GroupBox1.Enter
      End Sub
       Private  Sub  TextBox1_TextChanged(ByVal  sender  As
System.Object,  ByVal  e  As  System.EventArgs)  Handles
TextBox1.TextChanged
      Dim i As Integer
      ListView1.SelectedItems.Clear()
      TextBox1.Focus()
      Try
      If Me.TextBox1.Text = "" Then
      TextBox2.Text = ""
      Else
      For i = 0 To ListView1.Items.Count - 1
      If TextBox1.Text = ListView1.Items(i).SubItems(0).Text
Then
      ComboBox1.Text = ListView1.Items(i).SubItems(1).Text
      TextBox2.Text = ListView1.Items(i).SubItems(2).Text
      TextBox3.Text = ListView1.Items(i).SubItems(3).Text
      TextBox4.Text = ListView1.Items(i).SubItems(4).Text
      ComboBox2.Text = ListView1.Items(i).SubItems(5).Text
      TextBox5.Text = ListView1.Items(i).SubItems(6).Text
      TextBox6.Text = ListView1.Items(i).SubItems(7).Text
      ComboBox3.Text = ListView1.Items(i).SubItems(8).Text
      ListView1.Items(i).Selected = True
      Exit For
      End If
      Next
      End If
      Catch

      End Try
      End Sub
```

NOTES

NOTES

```
      Private Sub ListView1_SelectedIndexChanged(ByVal sender
As System.Object, ByVal e As System.EventArgs) Handles
ListView1.SelectedIndexChanged
      Dim i As Integer
      For i = 0 To ListView1.Items.Count - 1
      If ListView1.Items(i).Selected = True Then
      TextBox1.Text = ListView1.Items(i).SubItems(0).Text
      TextBox7.Clear()
      Exit For
      End If
      Next
      ListView1.Focus()
      ListView1.FullRowSelect = True
      End Sub

      Private Sub Button6_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button6.Click
      Try
      Dim i As Integer
      For i = 0 To ListView1.Items.Count - 1
      If ListView1.Items(i).Selected = True Then
      TextBox1.Text = ListView1.Items(i + 1).SubItems(0).Text
      Exit For
      End If
      Next
      ListView1.Focus()
      ListView1.FullRowSelect = True
      Catch ex As Exception

      End Try
      End Sub
      Private Sub ComboBox1_SelectedIndexChanged(ByVal sender
As System.Object, ByVal e As System.EventArgs) Handles
ComboBox1.SelectedIndexChanged
      Call GroupNameCom()
      End Sub
      Sub GroupNameCom()
      Try
      If  objcon.State  =  ConnectionState.Closed  Then
objcon.Open()
      com = New OleDb.OleDbCommand("Select * from GroupD",
objcon)
      dr = com.ExecuteReader
      While dr.Read
      If dr.Item(0) = ComboBox1.Text Then
      TextBox7.Text = dr.Item(1)
      End If

      End While
      dr.Close()
      objcon.Close()
      Catch ex As Exception
      End Try
      End Sub

      Private Sub ComboBox1_TextUpdate(ByVal sender As Object,
ByVal e As System.EventArgs) Handles ComboBox1.TextUpdate
      Call GroupNameCom()
      End Sub

      Private Sub Button5_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button5.Click
      Try
      Dim i As Integer
      For i = 0 To ListView1.Items.Count - 1
      If ListView1.Items(i).Selected = True Then
      TextBox1.Text = ListView1.Items(i - 1).SubItems(0).Text
      Exit For
      End If
      Next
      ListView1.Focus()
      ListView1.FullRowSelect = True
      Catch ex As Exception

      End Try
      End Sub
     End Class
```

NOTES



Book Details:

```
    Public Class BookDetail
     Dim sel As Integer
     Private Sub ComboBox1_SelectedIndexChanged(ByVal sender
As System.Object, ByVal e As System.EventArgs) Handles
ComboBox1.SelectedIndexChanged
     Label1.Text = ComboBox1.Text
     Label1.Visible = True
     If Label1.Text = "STATUS" Then
     ComboBox2.Enabled = True
     ComboBox2.Visible = True
     TextBox1.Visible = False

     Else
     ComboBox2.Enabled = False
     ComboBox2.Visible = False
     TextBox1.Visible = True

     End If
     Call forselect()
     End Sub
     Sub forselect()
     If ComboBox1.Text = "BOOK ID" Then
     sel = 1
     ElseIf ComboBox1.Text = "BOOK NAME" Then
     sel = 2
     ElseIf ComboBox1.Text = "AUTHOR" Then
     sel = 3
```

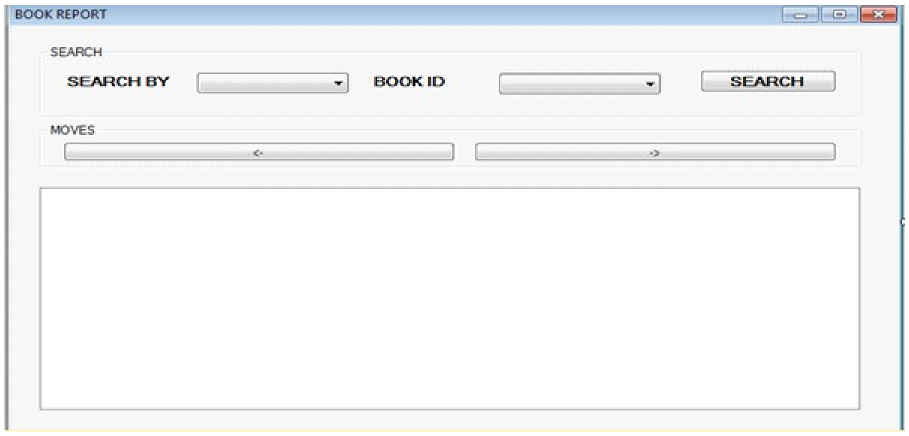
```
      ElseIf ComboBox1.Text = “STATUS” Then
      sel = 8
      End If
      End Sub

      Private Sub BookDetail_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
      ComboBox2.Visible = False
      TextBox1.Visible = False
      Label1.Visible = False
      Call readData()
      End Sub
      Sub readData()
      ListView1.Clear()
     ListView1.Columns.Add(“BOOK ID”, 90,
HorizontalAlignment.Center)
     ListView1.Columns.Add(“GROUP ID”, 90,
HorizontalAlignment.Center)
     ListView1.Columns.Add(“BOOK NAME”, 310,
HorizontalAlignment.Center)
     ListView1.Columns.Add(“PUBLISHER”, 90,
HorizontalAlignment.Center)
     ListView1.Columns.Add(“AUTHOR”, 90,
HorizontalAlignment.Center)
     ListView1.Columns.Add(“PUBLISHING YEAR”, 130,
HorizontalAlignment.Center)
     ListView1.Columns.Add(“EDITION”, 90,
HorizontalAlignment.Center)
     ListView1.Columns.Add(“PRICE”, 90,
HorizontalAlignment.Center)
     ListView1.Columns.Add(“STATUS”, 90,
HorizontalAlignment.Center)
      ListView1.View = View.Details
      sel = 5
      ‘Call whenclick()
      End Sub
      Sub whenclick()
      Try
      While dr.Read()
```

NOTES

```
      Call adddatatolistview(ListView1, dr(0), dr(1), dr(2),
dr(3), dr(4), dr(5), dr(6), dr(7), dr(8))
      End While
      dr.Close()
      objcon.Close()
      Catch
      'MsgBox("Please Refresh", MsgBoxStyle.Information, "")
      End Try
      End Sub
      Public Sub adddatatolistview(ByVal lvw As ListView, ByVal
BookID As String, ByVal GroupID As String, ByVal BookName As
String, ByVal publisher As String, ByVal author As String,
ByVal pubyear As String, ByVal edi As String, ByVal pric As
String, ByVal status As String)
      Dim lv As New ListViewItem
      lvw.Items.Add(lv)
      lv.Text = BookID
      lv.SubItems.Add(GroupID)
      lv.SubItems.Add(BookName)
      lv.SubItems.Add(publisher)
      lv.SubItems.Add(author)
      lv.SubItems.Add(pubyear)
      lv.SubItems.Add(edi)
      lv.SubItems.Add(pric)
      lv.SubItems.Add(status)
      End Sub


      Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button1.Click
       If objcon.State = ConnectionState.Closed Then
objcon.Open()
      Select Case (sel)
      Case 1
      com = New OleDb.OleDbCommand("select * from Books where
BookID='" & TextBox1.Text & "'", objcon)
      dr = com.ExecuteReader
      Case 2
      com = New OleDb.OleDbCommand("select * from Books where
BookName='" & TextBox1.Text & "'", objcon)
      dr = com.ExecuteReader
      Case 3
      com = New OleDb.OleDbCommand("select * from Books where
Author='" & TextBox1.Text & "'", objcon)
      dr = com.ExecuteReader
      Case 5
      com = New OleDb.OleDbCommand("select * from Books",
objcon)
      dr = com.ExecuteReader
      Case 8
      com = New OleDb.OleDbCommand("select * from Books where
Status='" & ComboBox2.Text & "'", objcon)
      dr = com.ExecuteReader
      End Select
      Call readData()
      Call whenclick()
      objcon.Close()
      End Sub
      Private Sub ListView1_SelectedIndexChanged(ByVal sender
As System.Object, ByVal e As System.EventArgs) Handles
ListView1.SelectedIndexChanged
      End Sub
      Private Sub Button6_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button6.Click
      Try
      Dim i As Integer
      For i = 0 To ListView1.Items.Count - 1
      If ListView1.Items(i).Selected = True Then
      TextBox1.Text = ListView1.Items(i + 1).SubItems(0).Text
      Exit For
      End If
      Next
      ListView1.Focus()
      ListView1.FullRowSelect = True
      Catch ex As Exception

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

NOTES

```
 Try
 Dim i As Integer
 For i = 0 To ListView1.Items.Count - 1
 If ListView1.Items(i).Selected = True Then
 TextBox1.Text = ListView1.Items(i - 1).SubItems(0).Text
 Exit For
 End If
 Next
 ListView1.Focus()
 ListView1.FullRowSelect = True
 Catch ex As Exception

 End Try
 End Sub
End Class
```



Issue Book:

```
Public Class IssueBook

 Private Sub Button9_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button9.Click
 Me.Close()
 End Sub

  Private  Sub  PictureBox1_Click(ByVal  sender  As
System.Object, ByVal e As System.EventArgs)

 End Sub
```

```
      Private Sub IssueBook_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
      Call Retrive_C()
      Call BookID_Combo()
      Call readData()
      End Sub
      Sub Retrive_C()
      Try
       If  objcon.State  =  ConnectionState.Closed  Then
objcon.Open()
      com = New OleDb.OleDbCommand("Select CID from Customer",
objcon)
      dr = com.ExecuteReader
      While dr.Read
      ComboBox5.Items.Add(dr.Item(0))
      End While
      dr.Close()
      objcon.Close()
      Catch ex As Exception

      End Try
      End Sub
      Sub BookID_Combo()
      Try
       If  objcon.State  =  ConnectionState.Closed  Then
objcon.Open()
      com = New OleDb.OleDbCommand("Select BookID from Books
WHERE status='Available'", objcon)
      dr = com.ExecuteReader
      While dr.Read
      ComboBox1.Items.Add(dr.Item(0))
      End While
      dr.Close()
      objcon.Close()
      Catch ex As Exception

      End Try
      End Sub
      Sub readData()
```

NOTES

```
      ListView1.Clear()
         ListView1.Columns.Add(“BOOK   ID”,   90,
HorizontalAlignment.Center)
         ListView1.Columns.Add(“GROUP   ID”,   90,
HorizontalAlignment.Center)
        ListView1.Columns.Add(“BOOK  NAME”,  310,
HorizontalAlignment.Center)
         ListView1.Columns.Add(“PUBLISHER”,   90,
HorizontalAlignment.Center)
           ListView1.Columns.Add(“AUTHOR”,     90,
HorizontalAlignment.Center)
       ListView1.Columns.Add(“PUBLISHING YEAR”, 130,
HorizontalAlignment.Center)
          ListView1.Columns.Add(“EDITION”,     90,
HorizontalAlignment.Center)
           ListView1.Columns.Add(“PRICE”,      90,
HorizontalAlignment.Center)
           ListView1.Columns.Add(“STATUS”,     90,
HorizontalAlignment.Center)
      ListView1.GridLines = True
      ListView1.View = View.Details
      Try


       If  (objcon.State  =  ConnectionState.Closed)  Then
objcon.Open()
      com = New OleDb.OleDbCommand(“SELECT * FROM Books WHERE
status=’Available’”, objcon)
      dr = com.ExecuteReader
      While dr.Read()
      Call adddatatolistview(ListView1, dr(0), dr(1), dr(2),
dr(3), dr(4), dr(5), dr(6), dr(7), dr(8))
      End While
      dr.Close()
      objcon.Close()
      Catch
      ‘MsgBox(“Please Refresh”, MsgBoxStyle.Information, “”)
      End Try
      End Sub
      Public Sub adddatatolistview(ByVal lvw As ListView, ByVal
BookID As String, ByVal GroupID As String, ByVal BookName As
String, ByVal Publisher As String, ByVal Author As String,
ByVal PubYear As String, ByVal edi As String, ByVal pric As
String, ByVal st As String)
      Dim lv As New ListViewItem
      lvw.Items.Add(lv)
      lv.Text = BookID
      lv.SubItems.Add(GroupID)
      lv.SubItems.Add(BookName)
      lv.SubItems.Add(Publisher)
      lv.SubItems.Add(Author)
      lv.SubItems.Add(PubYear)
      lv.SubItems.Add(edi)
      lv.SubItems.Add(pric)
      lv.SubItems.Add(st)
      End Sub
      Sub Retrive()
      objcon.Open()
      com = New OleDb.OleDbCommand("SELECT * FROM Books",
objcon)
      com.ExecuteNonQuery()
      dr = com.ExecuteReader
      dr.Read()
      While (dr.NextResult)
      ComboBox1.Items.Add(dr(1))
      End While
      objcon.Close()
      End Sub

      Private Sub Button2_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button2.Click
      Try
      If objcon.State = ConnectionState.Closed Then
objcon.Open()
      com = New OleDb.OleDbCommand("UPDATE Books SET
status='Rented' WHERE BookID='" & ComboBox1.Text & "'", objcon)
      com.ExecuteNonQuery()
      objcon.Close()
      Call readData()
      If objcon.State = ConnectionState.Closed Then
objcon.Open()
```

NOTES

```
      com = New OleDb.OleDbCommand("INSERT INTO Issue VALUES('"
& ComboBox1.Text & "','" & ComboBox2.Text & "','" &
TextBox2.Text & "','" & ComboBox5.Text & "','" & TextBox1.Text
& "','" & DateTimePicker1.Text & "','" & DateTimePicker2.Text
& "')", objcon)
      com.ExecuteNonQuery()
      MsgBox("Book has been Issued!", 0, "")
      Call readData()
      objcon.Close()
      Catch ex As Exception
      MsgBox(ex.Message, 0, "")
      End Try
      End Sub

      Private Sub ComboBox1_SelectedIndexChanged(ByVal sender
As System.Object, ByVal e As System.EventArgs) Handles
ComboBox1.SelectedIndexChanged
      Dim i As Integer
      ListView1.SelectedItems.Clear()
      TextBox1.Focus()
      Try
      If Me.ComboBox1.Text = "" Then
      TextBox2.Text = ""
      Else
      For i = 0 To ListView1.Items.Count - 1
      If ComboBox1.Text = ListView1.Items(i).SubItems(0).Text
Then
      ComboBox2.Text = ListView1.Items(i).SubItems(1).Text
      TextBox2.Text = ListView1.Items(i).SubItems(2).Text
      TextBox3.Text = ListView1.Items(i).SubItems(3).Text
      TextBox4.Text = ListView1.Items(i).SubItems(4).Text
      ComboBox3.Text = ListView1.Items(i).SubItems(5).Text
      TextBox5.Text = ListView1.Items(i).SubItems(6).Text
      TextBox6.Text = ListView1.Items(i).SubItems(7).Text
      ComboBox4.Text = ListView1.Items(i).SubItems(8).Text

      ListView1.Items(i).Selected = True
      Exit For
      End If
```

NOTES

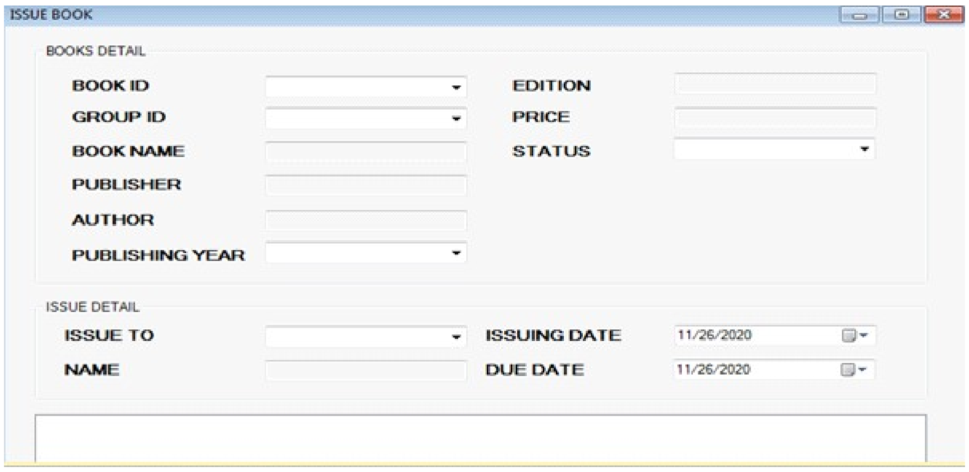
```
      Next
      End If
      Catch

      End Try
      End Sub

      Private Sub Button8_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button8.Click
      Try
      If ComboBox1.Text = "" Then
      MsgBox("Please mention the BookID", 0, "")
      Else
      If objcon.State = ConnectionState.Closed Then
      com = New OleDb.OleDbCommand("delete from Issue where
BookID='" & ComboBox1.Text & "'", objcon)
       If MsgBox("Do you really want to delete?",
MsgBoxStyle.YesNo, "Are you sure?") =
Windows.Forms.DialogResult.Yes Then
      com.ExecuteNonQuery()
      End If
      objcon.Close()
      End If
      End If
      Catch ex As Exception

      End Try
      End Sub

      Private Sub ListView1_SelectedIndexChanged(ByVal sender
As System.Object, ByVal e As System.EventArgs) Handles
ListView1.SelectedIndexChanged
      Dim i As Integer
      For i = 0 To ListView1.Items.Count - 1
      If ListView1.Items(i).Selected = True Then
      ComboBox1.Text = ListView1.Items(i).SubItems(0).Text
      Exit For
      End If
      Next
```

```
ListView1.Focus()
ListView1.FullRowSelect = True
End Sub
```

NOTES

```
Private Sub ComboBox5_SelectedIndexChanged(ByVal sender
As System.Object, ByVal e As System.EventArgs) Handles
ComboBox5.SelectedIndexChanged
Try
 If objcon.State = ConnectionState.Closed Then
objcon.Open()
 com = New OleDb.OleDbCommand("Select CID,CName from
Customer", objcon)
dr = com.ExecuteReader
While dr.Read
If dr.Item(0) = ComboBox5.Text Then
TextBox1.Text = dr.Item(1)
End If

End While
dr.Close()
objcon.Close()
Catch ex As Exception

End Try
End Sub

Private Sub Button6_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button6.Click
Try
Dim i As Integer
For i = 0 To ListView1.Items.Count - 1
If ListView1.Items(i).Selected = True Then
TextBox1.Text = ListView1.Items(i + 1).SubItems(0).Text
Exit For
End If
Next
ListView1.Focus()
ListView1.FullRowSelect = True
Catch ex As Exception
```

```
      End Try
      End Sub

      Private Sub Button5_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button5.Click
      Try
      Dim i As Integer
      For i = 0 To ListView1.Items.Count - 1
      If ListView1.Items(i).Selected = True Then
      TextBox1.Text = ListView1.Items(i - 1).SubItems(0).Text
      Exit For
      End If
      Next
      ListView1.Focus()
      ListView1.FullRowSelect = True
      Catch ex As Exception

      End Try
      End Sub
     End Class
```


ISSUE BOOK

BOOKS DETAIL

BOOK ID		EDITION	
GROUP ID		PRICE	
BOOK NAME		STATUS	
PUBLISHER			
AUTHOR			
PUBLISHING YEAR			

ISSUE DETAIL

ISSUE TO		ISSUING DATE	11/26/2020
NAME		DUE DATE	11/26/2020

Return Book:

```
     Public Class ReturnBook

      Private Sub Button9_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button9.Click
      Me.Close()
      End Sub
      Private Sub Button2_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button2.Click
      If ComboBox1.Text = "" Then
      MsgBox("Please mention the Book ID", 0, "")
      Else
      Try
       If objcon.State = ConnectionState.Closed Then
objcon.Open()
       com = New OleDb.OleDbCommand("UPDATE Books SET
status='Available' WHERE BookID='" & ComboBox1.Text & "'",
objcon)
      com.ExecuteNonQuery()
      objcon.Close()
      Call readData()
       If objcon.State = ConnectionState.Closed Then
objcon.Open()
       com = New OleDb.OleDbCommand("INSERT INTO Returns
VALUES('" & ComboBox1.Text & "','" & ComboBox2.Text & "','" &
TextBox2.Text & "','" & ComboBox5.Text & "','" & TextBox1.Text
& "','" & TextBox3.Text & "','" & TextBox7.Text & "','" &
DateTimePicker2.Text & "','" & TextBox6.Text & "')", objcon)
      com.ExecuteNonQuery()
      MsgBox("Book has been returned!", 0, "")
      objcon.Close()
      Catch ex As Exception
      MsgBox(ex.Message, 0, "")
      End Try
      End If
      End Sub

      Private Sub Button8_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button8.Click
      If ComboBox1.Text = "" Then
      MsgBox("Please mention a Book ID", 0, "")
      Else

      Try
       If objcon.State = ConnectionState.Closed Then
objcon.Open()
      com = New OleDb.OleDbCommand("DELETE FROM Returns WHERE
BookID='" & ComboBox1.Text & "'", objcon)
```

```
      com.ExecuteNonQuery()
      MsgBox("Deleted Success!", 0, "")
      Call ClearThem()
      objcon.Close()
      Catch ex As Exception

      End Try
      End If
      End Sub
      Sub ClearThem()
      ComboBox1.TabIndex = ""
      ComboBox2.Text = ""
      TextBox2.Text = ""
      TextBox3.Text = ""
      TextBox6.Text = ""
      ComboBox5.Text = ""
      TextBox1.Text = ""
      TextBox7.Text = ""
      DateTimePicker2.Refresh()
      End Sub

      Private Sub ReturnBook_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
      Call BookID_Combo()
      Call readData()

      End Sub
      Sub BookID_Combo()
      Try
       If objcon.State = ConnectionState.Closed Then
objcon.Open()
      com = New OleDb.OleDbCommand("Select BookID from Books
WHERE status='Rented'", objcon)
      dr = com.ExecuteReader
      While dr.Read
      ComboBox1.Items.Add(dr.Item(0))
      End While
      dr.Close()
```

NOTES

```
objcon.Close()
Catch ex As Exception

End Try
End Sub
Sub readData()
ListView1.Clear()
ListView1.Columns.Add("BOOK ID", 90, HorizontalAlignment.Center)
ListView1.Columns.Add("GROUP ID", 90, HorizontalAlignment.Center)
ListView1.Columns.Add("BOOK NAME", 310, HorizontalAlignment.Center)
ListView1.Columns.Add("PUBLISHER", 90, HorizontalAlignment.Center)
ListView1.Columns.Add("AUTHOR", 90, HorizontalAlignment.Center)
ListView1.Columns.Add("PUBLISHING YEAR", 130, HorizontalAlignment.Center)
ListView1.Columns.Add("EDITION", 90, HorizontalAlignment.Center)
ListView1.Columns.Add("PRICE", 90, HorizontalAlignment.Center)
ListView1.Columns.Add("STATUS", 90, HorizontalAlignment.Center)
ListView1.View = View.Details
Try

If (objcon.State = ConnectionState.Closed) Then objcon.Open()
com = New OleDb.OleDbCommand("SELECT * FROM Books WHERE status='Rented'", objcon)
dr = com.ExecuteReader
While dr.Read()
Call adddatatolistview(ListView1, dr(0), dr(1), dr(2), dr(3), dr(4), dr(5), dr(6), dr(7), dr(8))
End While
dr.Close()
objcon.Close()
Catch
```

```
      'MsgBox("Please Refresh", MsgBoxStyle.Information, "")
      End Try
      End Sub
      Public Sub adddatatolistview(ByVal lvw As ListView, ByVal
BookID As String, ByVal GroupID As String, ByVal BookName As
String, ByVal Publisher As String, ByVal Author As String,
ByVal PubYear As String, ByVal edi As String, ByVal pric As
String, ByVal st As String)
      Dim lv As New ListViewItem
      lvw.Items.Add(lv)
      lv.Text = BookID
      lv.SubItems.Add(GroupID)
      lv.SubItems.Add(BookName)
      lv.SubItems.Add(Publisher)
      lv.SubItems.Add(Author)
      lv.SubItems.Add(PubYear)
      lv.SubItems.Add(edi)
      lv.SubItems.Add(pric)
      lv.SubItems.Add(st)
      End Sub

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

      Private Sub ListView1_SelectedIndexChanged(ByVal sender
As  System.Object,  ByVal  e  As  System.EventArgs)  Handles
ListView1.SelectedIndexChanged
      Dim i As Integer
      For i = 0 To ListView1.Items.Count - 1
      If ListView1.Items(i).Selected = True Then
      ComboBox1.Text = ListView1.Items(i).SubItems(0).Text
      Exit For
      End If
      Next
      ListView1.Focus()
      ListView1.FullRowSelect = True
      End Sub
```

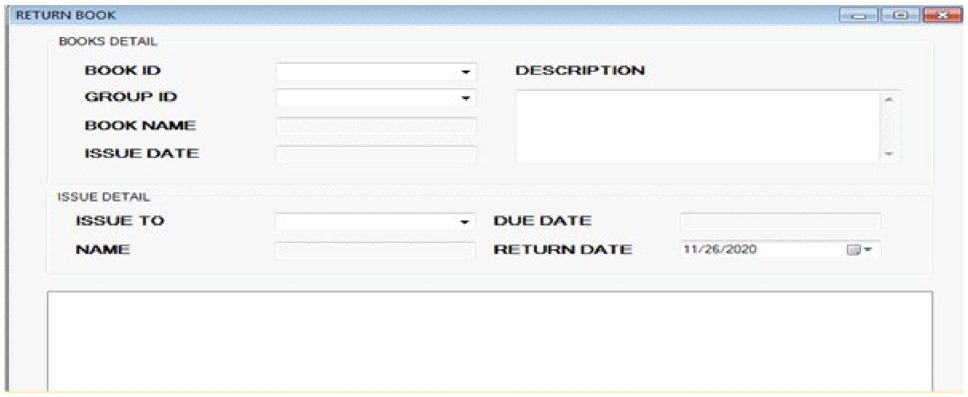
NOTES

NOTES

```
      Private Sub ComboBox1_SelectedIndexChanged(ByVal sender
As System.Object, ByVal e As System.EventArgs) Handles
ComboBox1.SelectedIndexChanged
      Dim i As Integer
      ListView1.SelectedItems.Clear()
      TextBox1.Focus()
      Try
      If Me.ComboBox1.Text = "" Then
      TextBox2.Text = ""
      Else
      For i = 0 To ListView1.Items.Count - 1
      If ComboBox1.Text = ListView1.Items(i).SubItems(0).Text
Then
      ComboBox2.Text = ListView1.Items(i).SubItems(1).Text
      TextBox2.Text = ListView1.Items(i).SubItems(2).Text
      ListView1.Items(i).Selected = True
      Exit For
      End If
      Next
      End If
      Catch

      End Try
      Call IssueDetail()
      End Sub
      Sub IssueDetail() '
      Try
       If objcon.State = ConnectionState.Closed Then
objcon.Open()
      com = New OleDb.OleDbCommand("Select IssueDate, IssueName,
IssueTo, DueDate from Issue WHERE BookID='" & ComboBox1.Text
& "'", objcon)
      dr = com.ExecuteReader
      While dr.Read
      ComboBox5.Text = dr.Item(2)
      TextBox1.Text = dr.Item(1)
      TextBox3.Text = dr.Item(0)
      TextBox7.Text = dr.Item(3)
      End While
      dr.Close()
      objcon.Close()
      Catch ex As Exception

      End Try
      End Sub

      Private Sub Button6_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button6.Click
      Try
      Dim i As Integer
      For i = 0 To ListView1.Items.Count - 1
      If ListView1.Items(i).Selected = True Then
      TextBox1.Text = ListView1.Items(i + 1).SubItems(0).Text
      Exit For
      End If
      Next
      ListView1.Focus()
      ListView1.FullRowSelect = True
      Catch ex As Exception

      End Try
      End Sub

      Private Sub Button5_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button5.Click
      Try
      Dim i As Integer
      For i = 0 To ListView1.Items.Count - 1
      If ListView1.Items(i).Selected = True Then
      TextBox1.Text = ListView1.Items(i + 1).SubItems(0).Text
      Exit For
      End If
      Next
      ListView1.Focus()
      ListView1.FullRowSelect = True
      Catch ex As Exception
```

```
  End Try
  End Sub
End Class
```

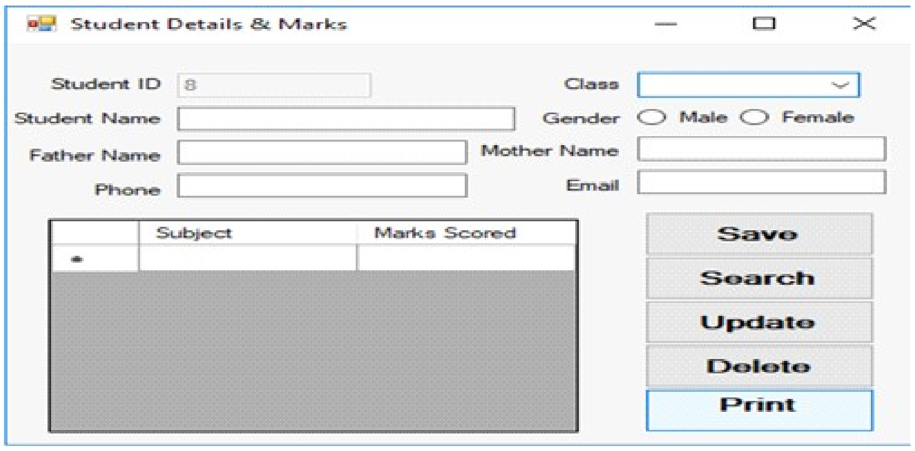
NOTES



2. Student Marksheet Processing

```
      Public conDB As New OleDb.OleDbConnection
       Public Sub connectDB()
       If conDB.State = ConnectionState.Closed Then
              conDB.ConnectionString =
"Provider=Microsoft.ACE.OLEDB.12.0; Data Source=" &
Application.StartupPath & "\stuDB.accdb"
       conDB.Open()
       End If
       End Sub
       Function getNewID(tblName As String, fldName As String)
As String
       Dim strVal, sql As String
       Dim cmd As OleDb.OleDbCommand
       connectDB()
       sql = "select max(" & fldName & ") from " & tblName
       cmd = New OleDb.OleDbCommand(sql, conDB)
       strVal = Convert.ToString(cmd.ExecuteScalar())
       If strVal = "" Then
       strVal = "1"
       Else
       strVal = Convert.ToString(CInt(strVal) + 1)
       End If
       Return strVal
       End Function
```

NOTES



Button Click:

```
      Dim strSQL As String
       Dim gndr As String
       Dim i As Integer
       If rdbFemale.Checked = True Then
       gndr = "Female"
       Else
       gndr = "Male"
       End If
        strSQL = "insert into studentmaster values(" &
txtStuID.Text & ",'" & cboClass.Text & "','" & txtStuName.Text
& "','" & txtFName.Text & "','" & txtMName.Text & "','" & gndr
& "','" & txtPhone.Text & "','" & txtEmail.Text & "')"
       cmd = New OleDb.OleDbCommand(strSQL, conDB)
       cmd.ExecuteNonQuery()
       For i = 0 To dgvMarks.RowCount - 2
       strSQL = "insert into studentmarks values(" & txtStuID.Text
& ",'" & dgvMarks.Item(0, i).Value & "'," & dgvMarks.Item(1,
i).Value & ")"
       cmd = New OleDb.OleDbCommand(strSQL, conDB)
       cmd.ExecuteNonQuery()
       Next
```

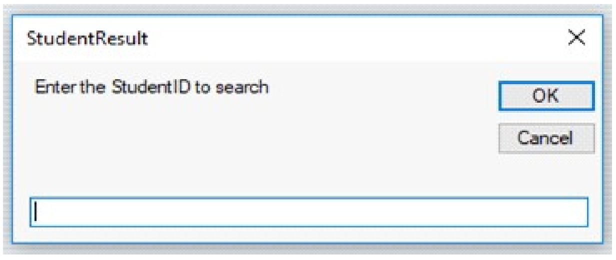
Search Button:

```
      Dim sid, cnt As Integer
       Dim drl As OleDb.OleDbDataReader
       Dim cmdl As New OleDb.OleDbCommand
       sid = CInt(InputBox("Enter the StudentID to search"))
       cmdl = New OleDbPress Ctrl+V to copy the following code
```

NOTES

```
     Dim sid, cnt As Integer
      Dim drl As OleDb.OleDbDataReader
      Dim cmdl As New OleDb.OleDbCommand
      sid = CInt(InputBox("Enter the StudentID to search"))
      cmdl = New OleDb.OleDbCommand("select * from studentmaster
where stuid=" & sid, conDB)
      drl = cmdl.ExecuteReader()
      If drl.Read() Then
      txtStuID.Text = drl.Item(0)
      cboClass.Text = drl.Item(1)
      txtStuName.Text = drl.Item(2)
      txtFName.Text = drl.Item(3)
      txtMName.Text = drl.Item(4)
      If drl.Item(5) = "Female" Then
      rdbFemale.Checked = True
      Else
      rdbMale.Checked = True
      End If
      txtPhone.Text = drl.Item(6)
      txtEmail.Text = drl.Item(7)
      drl.Close()
      cmdl = New OleDb.OleDbCommand("select subject, marks
from studentmarks where stuid=" & sid, conDB)
      drl = cmdl.ExecuteReader()
      dgvMarks.Rows.Clear()
      cnt = 0
      While drl.Read()
      dgvMarks.Rows.Add()
      dgvMarks.Item(0, cnt).Value = Convert.ToString
(drl.Item(0))
      dgvMarks.Item(1, cnt).Value = Convert.ToString
(drl.Item(1))
      cnt = cnt + 1
      End While
      Else
      MsgBox("No student with this ID")
      End If
     .OleDbCommand("select * from studentmaster where stuid="
& sid, conDB)
      drl = cmdl.ExecuteReader()
      If drl.Read() Then
      txtStuID.Text = drl.Item(0)
      cboClass.Text = drl.Item(1)
      txtStuName.Text = drl.Item(2)
      txtFName.Text = drl.Item(3)
      txtMName.Text = drl.Item(4)
      If drl.Item(5) = "Female" Then
      rdbFemale.Checked = True
      Else
      rdbMale.Checked = True
      End If
      txtPhone.Text = drl.Item(6)
      txtEmail.Text = drl.Item(7)
      drl.Close()
      cmdl = New OleDb.OleDbCommand("select subject, marks
from studentmarks where stuid=" & sid, conDB)
      drl = cmdl.ExecuteReader()
      dgvMarks.Rows.Clear()
      cnt = 0
      While drl.Read()
      dgvMarks.Rows.Add()
      dgvMarks.Item(0, cnt).Value = Convert.ToString
(drl.Item(0))
      dgvMarks.Item(1, cnt).Value = Convert.ToString
(drl.Item(1))
      cnt = cnt + 1
      End While
      Else
      MsgBox("No student with this ID")
      End If
```

NOTES


StudentResult

Enter the StudentID to search

NOTES

Button Update:

```
     Dim strSQL As String
      Dim gndr As String
      Dim i As Integer
      If rdbFemale.Checked = True Then
      gndr = "Female"
      Else
      gndr = "Male"
      End If
       strSQL = "update studentmaster set stuClass='" &
cboClass.Text & "', StuName='" & txtStuName.Text & "',
StuFname='" _
     & txtFName.Text & "',StuMName='" & txtMName.Text &
"',StuGender='" & gndr & "',StuPhone='" & txtPhone.Text _
     & "',StuEmail='" & txtEmail.Text & "' where StuID=" &
CInt(txtStuID.Text)
      cmd = New OleDb.OleDbCommand(strSQL, conDB)
      cmd.ExecuteNonQuery()
      ' delete all records from marks table to add the new
marks and subjects
      strSQL = "delete * from studentmarks where StuID=" &
CInt(txtStuID.Text)
      cmd = New OleDb.OleDbCommand(strSQL, conDB)
      cmd.ExecuteNonQuery()
      ' Insert the new subjects and marks for the student
      For i = 0 To dgvMarks.RowCount - 2
      strSQL = "insert into studentmarks values(" & txtStuID.Text
& ",'" & dgvMarks.Item(0, i).Value & "'," & dgvMarks.Item(1,
i).Value & ")"
      cmd = New OleDb.OleDbCommand(strSQL, conDB)
      cmd.ExecuteNonQuery()
      Next
```

Button Delete:

```
     Dim strSQL As String
      ' delete the record of student from master table
      strSQL = "delete * from studentmaster where StuID=" &
CInt(txtStuID.Text)
      cmd = New OleDb.OleDbCommand(strSQL, conDB)
      cmd.ExecuteNonQuery()
      ' delete all records from marks table
      strSQL = "delete * from studentmarks where StuID=" &
CInt(txtStuID.Text)
      cmd = New OleDb.OleDbCommand(strSQL, conDB)
      cmd.ExecuteNonQuery()
```

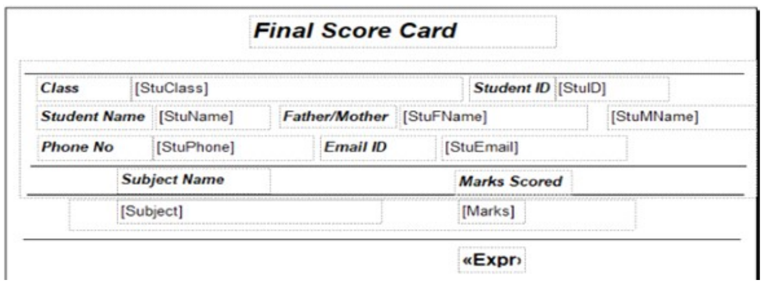
NOTES

Print Button:

```
     Dim frm As New Form2 ' creates an object of form containing
the reportviewer
      frm.Show()' displays the report
```

Report Viewer:

```
     Private Sub Form2_Load(sender As Object, e As EventArgs)
Handles MyBase.Load
      Dim dt1, dt2 As New DataTable
      Dim sid As Integer
      connectDB()
      sid = CInt(frmStuDetails.Controls("txtStuID").Text)
       Dim cmd1 As New OleDb.OleDbCommand("SELECT * from
StudentMarks where stuid=" & sid, conDB)
      cmd1.CommandTimeout = 4096
      Dim ta1 As New OleDb.OleDbDataAdapter(cmd1)
      ta1.Fill(dt1)
       Dim cmd2 As New OleDb.OleDbCommand("SELECT * from
StudentMaster where stuid=" & sid, conDB)
      cmd2.CommandTimeout = 4096
      Dim ta2 As New OleDb.OleDbDataAdapter(cmd2)
      ta2.Fill(dt2)
      With Me.ReportViewer1.LocalReport
      .DataSources.Clear()
     .DataSources.Add(New Microsoft.Reporting.WinForms.
ReportDataSource("DataSet1", dt1))
     .DataSources.Add(New Microsoft.Reporting.WinForms.
ReportDataSource("DataSet2", dt2))
      End With
      Me.ReportViewer1.RefreshReport()
      End Sub
```



Final Score Card

Class	[StuClass]		Student ID	[StuID]	
Student Name	[StuName]	Father/Mother	[StuFName]		[StuMName]
Phone No	[StuPhone]	Email ID	[StuEmail]		

Subject Name	Marks Scored
[Subject]	[Marks]

«Expr»

NOTES

3. Telephone Directory Maintenance

```
Imports System.IO
Imports System.IO.Directory
Imports System.IO.DirectoryInfo
Imports System.IO.Path
Imports System.Environment
Imports System.IO.FileStream
Imports System.IO.File
Imports System.IO.FileInfo
Imports System.Data.SqlClient
Imports System.Data
Imports System.Data.OleDb

Public Class frmPonBuk

 Dim strPath As String
 Dim dsContact As New DataSet
 Dim dsContactNam As New DataSet
 Dim daContact As New OleDbDataAdapter
 Dim daContactNam As New OleDbDataAdapter
 Dim sqlCommand As New OleDbCommand
 Dim strAction As String
 Dim strSQL As String
 Dim dt As New DataTable
 Dim dtContact As New DataTable
 Dim dtSearch As New DataTable
 Dim daSearch As New OleDbDataAdapter
 Dim dsSearch As New DataSet
 Dim drDSRow As DataRow
 Dim drNewRow As DataRow
 Dim cnPhoneBook As New OleDbConnection

 Private Sub frmPonBuk_KeyDown(ByVal sender As Object,
ByVal e As System.Windows.Forms.KeyEventArgs) Handles
Me.KeyDown
 'code for short cut key, note this will work if you
 'set the form's keypreview property to true
```

NOTES

```
      Select Case e.KeyCode
      Case Keys.F8
      If Me.cmdAdd.Enabled = True Then
      Me.cmdAdd_Click(sender, e)
      End If
      Case Keys.F9
      If Me.cmdEdit.Enabled = True Then
      Me.cmdEdit_Click(sender, e)
      End If
      Case Keys.F10
      If Me.cmdDelete.Enabled = True Then
      Me.cmdDelete_Click(sender, e)
      End If
      Case Keys.F11
      If Me.cmdUpdate.Enabled = True Then
      Me.cmdUpdate_Click(sender, e)
      End If
      Case Keys.F12
      If Me.cmdCancel.Enabled = True Then
      Me.cmdCancel_Click(sender, e)
      End If
      Case Keys.Enter
      SendKeys.Send("{TAB}")

      End Select
      End Sub

      Private Sub frmPonBuk_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
      'Dim strPath As String
       'you can use this method in order to get your
database(Access) path
       'strPath = System.Environment.CurrentDirectory &
"\Data\PhoneBook.accdb"
      'cnPhoneBook.ConnectionString = " Provider=Microsoft.
ACE.OLEDB.12.0;Data Source=" & specialName & ";Persist Security
Info=False;"
     cnPhoneBook.ConnectionString = " Provider=Microsoft.ACE.
OLEDB.12.0;Data Source=../Data/PhoneBook.accdb;Persist Security
Info=False;"
```

NOTES

```
      strSQL = " SELECT [LastName]+', '+[FirstName]+' '+[MiddleName] AS Name, TblContact.* FROM TblContact ORDER BY [LastName]+', '+[FirstName]+' '+[MiddleName];"
      daContact.SelectCommand = New OleDbCommand(strSQL, cnPhoneBook)
      daContact.Fill(dsContact, "TblContact")
      Me.dtContact = dsContact.Tables("TblContact")
      'binding controls to dataset
      Me.txtLstNam.DataBindings.Add("Text", dsContact, "TblContact.LastName")
      Me.txtFstNam.DataBindings.Add("Text", dsContact, "TblContact.FirstName")
      Me.txtMidNam.DataBindings.Add("Text", dsContact, "TblContact.MiddleName")
      Me.txtHomAdr.DataBindings.Add("Text", dsContact, "TblContact.HomeAdr")
      Me.txtBusAdr.DataBindings.Add("Text", dsContact, "TblContact.BusAdr")
      Me.txtTelNo.DataBindings.Add("Text", dsContact, "TblContact.TelNo")
      Me.txtMobNo.DataBindings.Add("Text", dsContact, "TblContact.MobNo")
      Me.txtEml.DataBindings.Add("Text", dsContact, "TblContact.EMail")

      'setting datagrid properties
      Me.dtgContact.DataSource = dsContact
      Me.dtgContact.DataMember = "TblContact"
      Me.dtgContact.Columns(0).HeaderText = "Name"
      Me.dtgContact.Columns(1).Visible = False
      Me.dtgContact.Columns(2).Visible = False
      Me.dtgContact.Columns(3).Visible = False
      Me.dtgContact.Columns(4).Visible = False
      Me.dtgContact.Columns(5).HeaderText = "Home Address"
      Me.dtgContact.Columns(6).HeaderText = "Bus. Address"
      Me.dtgContact.Columns(7).HeaderText = "Telephone"
      Me.dtgContact.Columns(8).HeaderText = "Mobile"
      Me.dtgContact.Columns(9).HeaderText = "E-Mail"

      'Used SQL statement for Combo box to display the name of contact person
      strSQL = " SELECT TblContact.ContactID, [LastName]+',
'+[FirstName]+' '+[MiddleName] AS Name FROM TblContact ORDER
BY [LastName]+', '+[FirstName]+' '+[MiddleName];"
      daContactNam.SelectCommand = New OleDbCommand(strSQL,
cnPhoneBook)
      daContactNam.Fill(dsContactNam, "TblContact")
      'datatable for combo box
      Me.dt = dsContactNam.Tables("TblContact")
      Me.cmbSearch.DataSource = dt
      Me.cmbSearch.DisplayMember = "Name"
      Me.cmbSearch.ValueMember = "ContactID"
      Me.cmbSearch.SelectedIndex = -1
       Me.txtRecPos.Text = "Contact Record " &
Me.BindingContext(dsContact, "TblContact").Position + 1 & "
of : " & dsContact.Tables("TblContact").Rows.Count
      ' call procedure to lock the text field
      lockField()
      ' call procedure to disabled update
      UpdtOff()
      End Sub

     Private Sub cmdFstRec_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles cmdFstRec.Click
     Me.BindingContext(dsContact, "TblContact").Position = 0
     Me.txtRecPos.Text = "Contact Record " &
Me.BindingContext(dsContact, "TblContact").Position + 1 & "
of : " & dsContact.Tables("TblContact").Rows.Count
      End Sub
      Private Sub cmdPrv_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles cmdPrv.Click
      Me.BindingContext(dsContact, "TblContact").Position =
Me.BindingContext(dsContact, "TblContact").Position - 1
       Me.txtRecPos.Text = "Contact Record " &
Me.BindingContext(dsContact, "TblContact").Position + 1 & "
of : " & dsContact.Tables("TblContact").Rows.Count
      End Sub

      Private Sub cmdNext_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles cmdNext.Click
      Me.BindingContext(dsContact, "TblContact").Position =
Me.BindingContext(dsContact, "TblContact").Position + 1
```

NOTES

NOTES

```
      Me.txtRecPos.Text = "Contact Record " &
Me.BindingContext(dsContact, "TblContact").Position + 1 & "
of : " & dsContact.Tables("TblContact").Rows.Count
      End Sub

      Private Sub cmdLst_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles cmdLst.Click
      Me.BindingContext(dsContact, "TblContact").Position =
Me.BindingContext(dsContact, "TblContact").Count - 1
      Me.txtRecPos.Text = "Contact Record " &
Me.BindingContext(dsContact, "TblContact").Position + 1 & "
of : " & dsContact.Tables("TblContact").Rows.Count
      End Sub
      Private Sub UnlockField()

      Me.txtFstNam.ReadOnly = False
      Me.txtLstNam.ReadOnly = False
      Me.txtMidNam.ReadOnly = False
      Me.txtHomAdr.ReadOnly = False
      Me.txtBusAdr.ReadOnly = False
      Me.txtTelNo.ReadOnly = False
      Me.txtMobNo.ReadOnly = False
      Me.txtEml.ReadOnly = False

      End Sub
      Private Sub lockField()

      Me.txtFstNam.ReadOnly = True
      Me.txtLstNam.ReadOnly = True
      Me.txtMidNam.ReadOnly = True
      Me.txtHomAdr.ReadOnly = True
      Me.txtBusAdr.ReadOnly = True
      Me.txtTelNo.ReadOnly = True
      Me.txtMobNo.ReadOnly = True
      Me.txtEml.ReadOnly = True

      End Sub
      Private Sub UpdtOff()
```

NOTES

```
      Me.cmdAdd.Enabled = True
      Me.cmdEdit.Enabled = True
      Me.cmdDelete.Enabled = True
      Me.cmdUpdate.Enabled = False
      Me.cmdCancel.Enabled = False

      Me.cmdAdd.BackColor = Color.Tan
      Me.cmdEdit.BackColor = Color.Tan
      Me.cmdDelete.BackColor = Color.Tan
      Me.cmdUpdate.BackColor = Color.Black
      Me.cmdCancel.BackColor = Color.Black
      End Sub
      Private Sub UpdtOn()

      Me.cmdAdd.Enabled = False
      Me.cmdEdit.Enabled = False
      Me.cmdDelete.Enabled = False
      Me.cmdUpdate.Enabled = True
      Me.cmdCancel.Enabled = True

      Me.cmdAdd.BackColor = Color.Black
      Me.cmdEdit.BackColor = Color.Black
      Me.cmdDelete.BackColor = Color.Black
      Me.cmdUpdate.BackColor = Color.Tan
      Me.cmdCancel.BackColor = Color.Tan

      End Sub

      Private Sub cmdAdd_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles cmdAdd.Click
      strAction = "ADD"
      UpdtOn()
      UnlockField()
      Me.BindingContext(dsContact, "TblContact").AddNew()
      Me.txtLstNam.Focus()
      End Sub
```

NOTES

```
      Private Sub cmdEdit_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles cmdEdit.Click
      strAction = "EDIT"
      UpdtOn()
      UnlockField()
      End Sub

      Private Sub cmdDelete_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles cmdDelete.Click
      Dim delCommand As New OleDbCommand
      Dim intPos As Integer
      Dim intContactID As Integer
      Dim strUsrRsp As String
        intPos = Me.BindingContext(dsContact,
"TblContact").Position
      intContactID = dtContact.Rows(intPos).Item(1)
      strUsrRsp = MsgBox("Do you want to delete this record",
MsgBoxStyle.YesNo + MsgBoxStyle.Question +
MsgBoxStyle.ApplicationModal, "Phone Book")
      If strUsrRsp = MsgBoxResult.Yes Then
      Try
      cnPhoneBook.Open()
      strSQL = "Delete from TblContact where (ContactID = " &
intContactID & ")"

      sqlCommand = New OleDbCommand(strSQL, cnPhoneBook)

      sqlCommand.ExecuteNonQuery()

      cnPhoneBook.Close()
      dsContact.Clear()
      daContact.Fill(dsContact, "TblContact")
      MsgBox("Record has been deleted", MsgBoxStyle.OkOnly +
MsgBoxStyle.Information + MsgBoxStyle.ApplicationModal, "Phone
Book")
      Catch ex As Exception
      MsgBox(Err.Description)
      End Try
      Else
      End If
      dsContactNam.Clear()
      daContactNam.Fill(dsContactNam, "TblContact")
      cmbSearch.SelectedIndex = -1
      End Sub

      Private Sub cmdUpdate_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles cmdUpdate.Click
      Dim SubPos As Integer
      Dim intPos As Integer
      Dim intContactID As Integer

      Try
      Select Case strAction
      Case "ADD"
     Me.BindingContext(dsContact, "TblContact").
EndCurrentEdit()
      cnPhoneBook.Open()
      strSQL = "INSERT INTO TblContact (LastName, FirstName,
MiddleName, HomeAdr, BusAdr, TelNo, MobNo, EMail) "
      strSQL = strSQL & " VALUES ('" & Me.txtLstNam.Text &
"','" & Me.txtFstNam.Text & "','" & Me.txtMidNam.Text & "','"
& Me.txtHomAdr.Text & "','" & Me.txtBusAdr.Text & "','" &
Me.txtTelNo.Text & "','" & Me.txtMobNo.Text & "','" &
Me.txtEml.Text & "');"
      sqlCommand = New OleDbCommand(strSQL, cnPhoneBook)
      sqlCommand.ExecuteNonQuery()

      cnPhoneBook.Close()
      dsContact.Clear()
      daContact.Fill(dsContact, "TblContact")

      Case "EDIT"
     intPos = Me.BindingContext(dsContact, "TblContact").
Position
      intContactID = dtContact.Rows(intPos).Item(1)

     Me.BindingContext(dsContact, "TblContact").
EndCurrentEdit()
      cnPhoneBook.Open()
```

NOTES

```
     strSQL = "UPDATE TblContact SET LastName = '" &
Me.txtLstNam.Text & "', FirstName = '" & Me.txtFstNam.Text &
"', MiddleName = '" & Me.txtMidNam.Text & "', HomeAdr = '" &
Me.txtHomAdr.Text & "', "
     strSQL = strSQL & " BusAdr = '" & Me.txtBusAdr.Text & "',
TelNo = '" & Me.txtTelNo.Text & "', MobNo = '" & Me.txtMobNo.Text
& "', EMail = '" & Me.txtEml.Text & "' WHERE
(((TblContact.ContactID)=" & intContactID & "));"
      sqlCommand = New OleDbCommand(strSQL, cnPhoneBook)


      sqlCommand.ExecuteNonQuery()
      cnPhoneBook.Close()
        SubPos = Me.BindingContext(dsContact,
"TblContact").Position
      dsContact.Clear()
      daContact.Fill(dsContact, "TblContact")
      Me.BindingContext(dsContact, "TblContact").Position =
SubPos


      End Select
      UpdtOff()
      lockField()
      Catch ex As Exception
      MsgBox(strSQL)
      End Try
      dsContactNam.Clear()
      daContactNam.Fill(dsContactNam, "TblContact")
     Me.txtRecPos.Text = "Contact Record " &
Me.BindingContext(dsContact, "TblContact").Position + 1 & "
of : " & dsContact.Tables("TblContact").Rows.Count
      End Sub
      Private Sub cmdCancel_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles cmdCancel.Click
     Me.BindingContext(dsContact, "TblContact").
CancelCurrentEdit()
      UpdtOff()
      lockField()
       Me.txtRecPos.Text = "Contact Record " &
Me.BindingContext(dsContact, "TblContact").Position + 1 & "
of : " & dsContact.Tables("TblContact").Rows.Count


      End Sub
```

NOTES

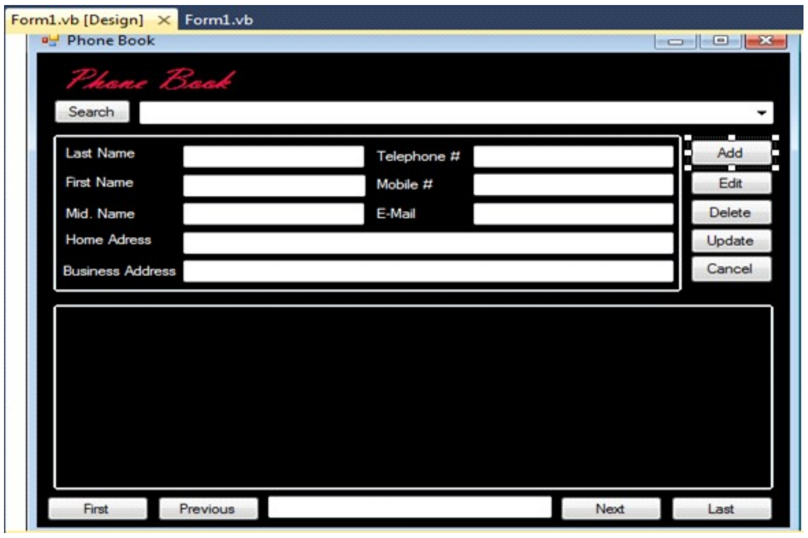
```
      Private Sub cmdSearch_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles cmdSearch.Click
      Dim ContactIDSrh As Integer
      Dim ColNum As Integer
      Dim RowNum As Integer
      Dim RecCount As Integer
      ColNum = 0
      RowNum = 0
      'Check Combo box if it has a value
      If Me.cmbSearch.SelectedValue <> 0 Then
       RecCount  =  Me.BindingContext(dsContact,
"TblContact").Count
      ContactIDSrh = Me.cmbSearch.SelectedValue
      'move at first record
      Me.BindingContext(dsContact, "TblContact").Position =
0
      'loop until we find the desired Contact Person
      Do While ContactIDSrh <> dtContact.Rows(RowNum).Item(1)
      If RowNum <> RecCount Then
      'move record position
      Me.BindingContext(dsContact, "TblContact").Position =
RowNum + 1
      RowNum = RowNum + 1
      Else
      'exit loop if record found
      Exit Do
      End If
      Loop
      Else
      MsgBox("Please Select the Student name to be searched")
      End If
      End Sub

      Private Sub txtLstNam_LostFocus(ByVal sender As Object,
ByVal e As System.EventArgs) Handles txtLstNam.LostFocus
      'this will trigger if the txtLstNam has lost the focus
and during adding new or editting existing record
      If strAction = "ADD" Or strAction = "EDIT" Then
      'transform the string into proper case
```

NOTES

```
       Me.txtLstNam.Text  =  StrConv(Me.txtLstNam.Text,
VbStrConv.ProperCase)
      End If
      End Sub

      Private Sub txtFstNam_LostFocus(ByVal sender As Object,
ByVal e As System.EventArgs) Handles txtFstNam.LostFocus
      If strAction = "ADD" Or strAction = "EDIT" Then
       Me.txtFstNam.Text  =  StrConv(Me.txtFstNam.Text,
VbStrConv.ProperCase)
      End If
      End Sub

      Private Sub txtMidNam_LostFocus(ByVal sender As Object,
ByVal e As System.EventArgs) Handles txtMidNam.LostFocus
      If strAction = "ADD" Or strAction = "EDIT" Then
       Me.txtMidNam.Text  =  StrConv(Me.txtMidNam.Text,
VbStrConv.ProperCase)
      End If
      End Sub

      Private Sub txtHomAdr_LostFocus(ByVal sender As Object,
ByVal e As System.EventArgs) Handles txtHomAdr.LostFocus
      If strAction = "ADD" Or strAction = "EDIT" Then
       Me.txtHomAdr.Text  =  StrConv(Me.txtHomAdr.Text,
VbStrConv.ProperCase)
      End If
      End Sub

      Private Sub txtBusAdr_LostFocus(ByVal sender As Object,
ByVal e As System.EventArgs) Handles txtBusAdr.LostFocus
      If strAction = "ADD" Or strAction = "EDIT" Then
       Me.txtBusAdr.Text  =  StrConv(Me.txtBusAdr.Text,
VbStrConv.ProperCase)
      End If
      End Sub

      Private Sub txtTelNo_LostFocus(ByVal sender As Object,
ByVal e As System.EventArgs) Handles txtTelNo.LostFocus
      If strAction = "ADD" Or strAction = "EDIT" Then
```

NOTES

```
      If Len(Me.txtTelNo.Text) = 7 Then
      Me.txtTelNo.Text = Mid(Me.txtTelNo.Text, 1, 3) & "-" &
Mid(Me.txtTelNo.Text, 4, 2) & "-" & Mid(Me.txtTelNo.Text, 6,
2)
      End If
      End If
      End Sub

      Private Sub dtgContact_CellClick(ByVal sender As Object,
ByVal e As System.Windows.Forms.DataGridViewCellEventArgs)
Handles dtgContact.CellClick
       Me.txtRecPos.Text = "Contact Record " &
Me.BindingContext(dsContact, "TblContact").Position + 1 & "
of : " & dsContact.Tables("TblContact").Rows.Count
      End Sub

     End Class
```



4. Gas Booking and Delivering

Main:

```
     Private Sub Command1_Click() Handles Command1.Click
     '#Const Compile_Command1_Click = True
     #If Compile_Command1_Click Or CompileAll_Form1 Then
      Form2.Load()
      Form2.Show()
      Close()
     #End If' Compile_Command1_Click
```

NOTES

```
      End Sub
      Private Sub Command2_Click(ByVal sender As Object, ByVal
e As System.EventArgs) Handles Command2.Click
     '#Const Compile_Command2_Click = True
     #If Compile_Command2_Click Or CompileAll_Form1 Then
      Form15.Load()
      Form15.Show()
      Close()
     #End If' Compile_Command2_Click
      End Sub

      Private Sub Command3_Click(ByVal sender As Object, ByVal
e As System.EventArgs) Handles Command3.Click
     '#Const Compile_Command3_Click = True
     #If Compile_Command3_Click Or CompileAll_Form1 Then
      'b = InputBox("Enter Record No", "Find to Modify")
      Form6.Load()
      Form6.Show()
      Close()
     #End If' Compile_Command3_Click
      End Sub

      Private Sub Command4_Click() Handles Command4.Click
     '#Const Compile_Command4_Click = True
     #If Compile_Command4_Click Or CompileAll_Form1 Then
      Form16.Load()
      Form16.Show()
      Close()
     #End If' Compile_Command4_Click
      End Sub

      Private Sub Command5_Click(ByVal sender As Object, ByVal
e As System.EventArgs) Handles Command5.Click
     '#Const Compile_Command5_Click = True
     #If Compile_Command5_Click Or CompileAll_Form1 Then
      Form5.Load()
      Form5.Show()
      Close()
```

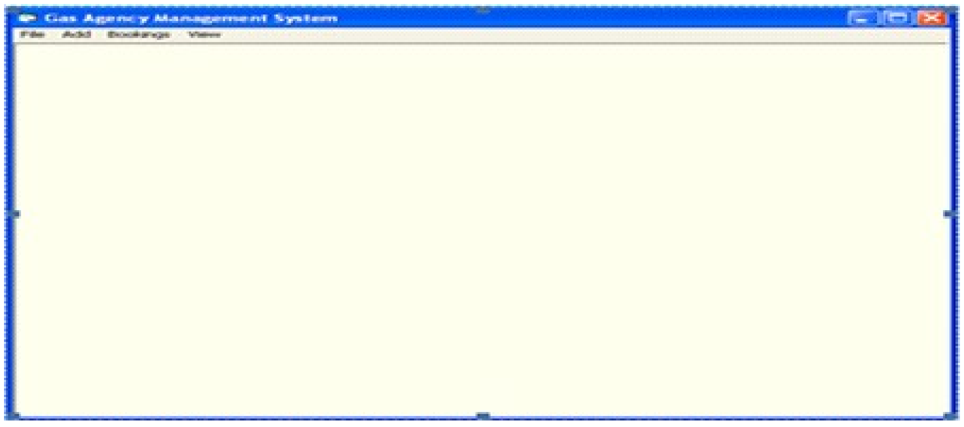
```
#End If' Compile_Command5_Click
 End Sub

 Private Sub Command6_Click(ByVal sender As Object, ByVal
e As System.EventArgs) Handles Command6.Click
'#Const Compile_Command6_Click = True
#If Compile_Command6_Click Or CompileAll_Form1 Then
 Form7.Load()
 Form7.Show()
 Close()
#End If' Compile_Command6_Click
 End Sub

 Private Sub Command7_Click() Handles Command7.Click
'#Const Compile_Command7_Click = True
#If Compile_Command7_Click Or CompileAll_Form1 Then
 Application.Exit()
#End If' Compile_Command7_Click
 End Sub

 Private Sub Command8_Click() Handles Command8.Click
'#Const Compile_Command8_Click = True
#If Compile_Command8_Click Or CompileAll_Form1 Then
 Form8.Load()
 Form8.Show()
 Close()
#End If' Compile_Command8_Click
 End Sub

 Private Sub Command9_Click(ByVal sender As Object, ByVal
e As System.EventArgs) Handles Command9.Click
'#Const Compile_Command9_Click = True
#If Compile_Command9_Click Or CompileAll_Form1 Then
 Form14.Load()
 Form14.Show()
 Close()
#End If' Compile_Command9_Click
 End Sub
```

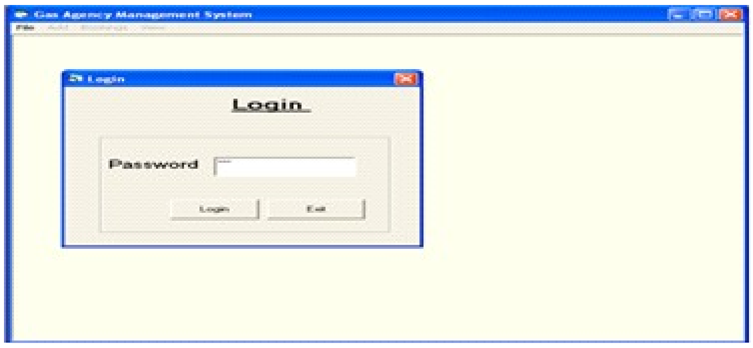
NOTES

```
     Private Sub Form1_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
     '#Const Compile_Form_Load = True
     #If Compile_Form_Load Or CompileAll_Form1 Then
      Timer1.Interval = 50
     #End If ' Compile_Form_Load
      End Sub
      Private Sub Timer1_Tick(ByVal sender As Object, ByVal e
As System.EventArgs) Handles Timer1.Tick
     '#Const Compile_Timer1_Timer = True
     #If Compile_Timer1_Timer Or CompileAll_Form1 Then
      l1.Top -= 60
      If l1.Top<=100 Then
      l1.Top = 13000
      End If

      L2.Top -= 60
      If L2.Top<=100 Then
      L2.Top = 13000
      End If

      L3.Top -= 60
      If L3.Top<=100 Then
      L3.Top = 13000
      End If

      L4.Top -= 60
      If L4.Top<=100 Then
      L4.Top = 13000
      End If

      L5.Top -= 60
      If L5.Top<=100 Then
      L5.Top = 13000
      End If

      L6.Top -= 60
      If L6.Top<=60 Then
      L6.Top = 13000
      End If
```

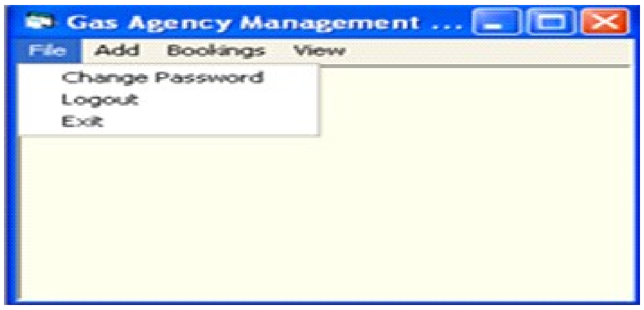
NOTES

```
       l7.Top -= 60
       If l7.Top<=60 Then
       l7.Top = 13000
       End If

       l8.Top -= 60
       If l8.Top<=60 Then
       l8.Top = 13000
       End If

       l9.Top -= 60
       If l9.Top<=60 Then
       l9.Top = 13000
       End If
      #End If' Compile_Timer1_Timer
       End Sub

       Private Sub Timer2_Tick(ByVal sender As Object, ByVal e
As System.EventArgs) Handles Timer2.Tick
      '#Const Compile_Timer2_Timer = True
      #If Compile_Timer2_Timer Or CompileAll_Form1 Then
      Label4.ForeColor = ColorTranslator.FromOle
(QBColor(Rnd()*15))
      Label5.ForeColor = ColorTranslator.FromOle
(QBColor(Rnd()*15))
      #End If' Compile_Timer2_Timer
       End Sub

      End Class
```

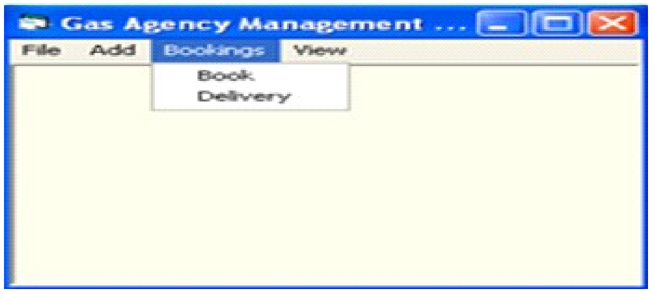




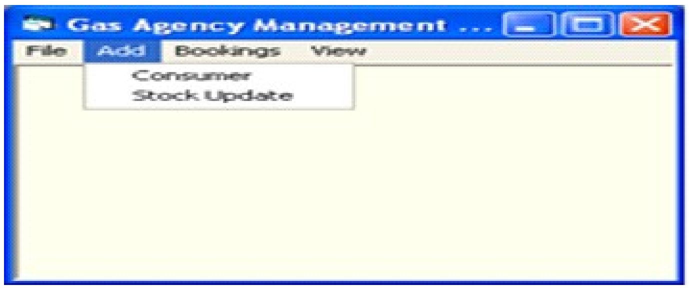
File Menu:



Booking Menu:



Add Menu:



5. Electricity Bill Management

Main form:

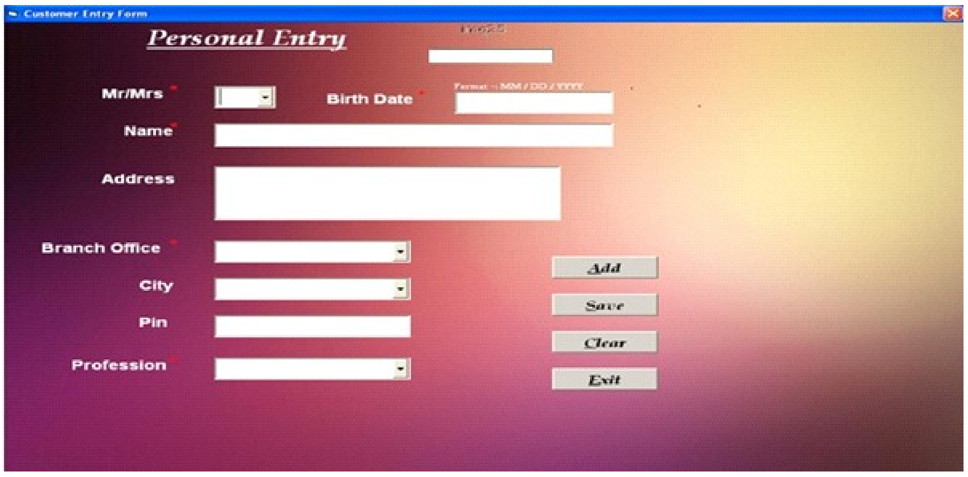
```
Private Sub Cmdexit_Click()
End
End Sub
```

```
     Private Sub Cmd1_Click()
     txtuser.Text = UCase(txtuser)
     txtpass.Text = UCase(txtpass) '& LCase(txtpass)
     If txtuser.Text = "ELECTRICITY" And txtpass = "KULKARNI"
Then
     Main.Show
     Me.Hide
     Else
     MsgBox ("Please try again")
     txtuser.SetFocus
     End If
     End Sub

     Private Sub Cmd2_Click()
     End
     End Sub

     Customer Form:
     Private Sub Cmdadd_Click()
     Adodc1.Refresh
     Adodc1.Recordset.AddNew
     End Sub

     Private Sub cmdclear_Click()
     Adodc1.Refresh
     cmbgn.Text = ""
     txtnm.Text = ""
     txtad.Text = ""
     cmbec.Text = ""
     cmbct.Text = ""
     Txtpn.Text = ""
     cmbpro.Text = ""
     Txtdob.Text = ""
     End Sub

     Private Sub cmdsv_Click()
     If cmbgn.Text = "" Or txtnm.Text = "" Or cmbec.Text = ""
Or cmbpro.Text = "" Or Txtdob.Text = "" Then
```

NOTES

```
MsgBox “Please Fill Requireds Fields Then Save Your Record”
Else
Adodc1.Recordset.Fields(0) = cmbgn.Text
Adodc1.Recordset.Fields(1) = txtnm.Text
Adodc1.Recordset.Fields(2) = txtad.Text
Adodc1.Recordset.Fields(3) = cmbec.Text
Adodc1.Recordset.Fields(4) = cmbct.Text
Adodc1.Recordset.Fields(5) = Txtpn.Text
Adodc1.Recordset.Fields(6) = cmbpro.Text
Adodc1.Recordset.Fields(7) = Text1.Text ‘lbldt.Caption
Adodc1.Recordset.Fields(8) = Txtdob.Text
Adodc1.Recordset.Save
Adodc1.Refresh
MsgBox “Record Save Successfully”

cmbgn.Text = “”
txtnm.Text = “”
txtad.Text = “”
cmbec.Text = “”
cmbct.Text = “”
Txtpn.Text = “”
cmbpro.Text = “”
Txtdob.Text = “”
End If
End Sub

Private Sub Command5_Click()
Unload Me
End Sub


Private Sub Form_Load()
‘Adodc1.Refresh
cmbgn.Text = “”
txtnm.Text = “”
txtad.Text = “”
cmbec.Text = “”
cmbct.Text = “”
```

```
    Txtdob.Text = ""
    Txtpn.Text = ""
    cmbpro.Text = ""
    Text1.Text = Date

    'FormatDateTime((DateTime.Day) & ("-") & (DateTime.Month)
& ("-") & (DateTime.Year))
    'd & "/" & m & "/" & y
    'lbldt.Caption = FormatDateTime(DateTime.Date, vbLongDate)
    'vbGeneralDate
    'DateTime.Date
    End Sub
```



Bill:

```
    Private Sub Cmbnm2_LostFocus()
    'On Error Resume Next
    'Adodc1.Refresh
    'While Not Adodc1.Recordset.EOF = True
    'If Adodc1.Recordset!Name = Cmbnm2.Text Then
    'txtadd.Text = Adodc1.Recordset!Add 'ress
    'Txtex.Text = Adodc1.Recordset!Exchange
    'Txtpin.Text = Adodc1.Recordset!pincode
    'Else
    ''
    ''Exit Do
    'End If
    'Loop
    Adodc1.Refresh
```

NOTES

```
Adodc2.Refresh
Do While Adodc1.Recordset.EOF = False
If Adodc1.Recordset!Name = Cmbnm2.Text Then
Txtadd.Text = Adodc1.Recordset!Add
Txtex.Text = Adodc1.Recordset!Exchange
Txtpin.Text = Adodc1.Recordset!pincode
Text1.Text = Adodc1.Recordset!plan
Exit Do
End If
'End If

Adodc1.Recordset.MoveNext
'Adodc2.Recordset.MoveNext
Loop

'Do While Adodc2.Recordset.EOF = False
'If Adodc2.Recordset!planname = Text1.Text Then
'Txtmcc.Text = Adodc2.Recordset!MonthlyCharges
'txtfc.Text = Adodc2.Recordset!free_calls
'Exit Do
'End If
'Adodc2.Recordset.MoveNext
'Loop


'Adodc2.Recordset.MoveNext

Txtn2.Text = Cmbnm2.Text
Txtn3.Text = Txtadd.Text
Txtn4.Text = txtcust.Text
Txtn5.Text = Txttel.Text
Txtn6.Text = Txtex.Text
Txtn7.Text = Txtpin.Text
Txtdb.Text = Txtfmc.Text
Txtdb1.Text = txtfc.Text
'Wend
End Sub
```

```
Private Sub Cmdadd_Click()
Adodc3.Refresh
Adodc3.Recordset.MoveNext
Adodc3.Recordset.AddNew
Cmdadd.Visible = False
cmdsv.Visible = True
End Sub

Private Sub cmdcalc_Click()
'Txtgmc.Text = Val(Txtcmr.Text) - Val(Txtomr.Text)
'Txtncc.Text = Val(Txtgmc.Text) - Val(txtfc.Text)
'If Txtncc.Text <= 0 Then
'Txtncc.Text = "0"
'Txtmcc.Text = Txtncc.Text
'Else
'Txtmcc.Text = Txtncc.Text
'End If
End Sub

Private Sub cmdsv_Click()
'Txtn2.Text = Cmbnm2.Text
'Txtn3.Text = txtadd.Text
'Txtn4.Text = txtcust.Text
'Txtn5.Text = txttel.Text
'Txtn6.Text = Txtex.Text
'Txtn7.Text = txtpin.Text
'Txtdb.Text = Txtfmc.Text
'Txtdb1.Text = txtfc.Text
Adodc3.Recordset.Fields(0) = Txtn2.Text
Adodc3.Recordset.Fields(1) = Txtn4.Text
Adodc3.Recordset.Fields(2) = Txtn5.Text
Adodc3.Recordset.Fields(3) = Txtn6.Text
Adodc3.Recordset.Fields(4) = Txtn7.Text
Adodc3.Recordset.Fields(5) = Txtn3.Text
Adodc3.Recordset.Fields(6) = Txtomr.Text
Adodc3.Recordset.Fields(7) = Txtcmr.Text
Adodc3.Recordset.Fields(8) = Txtgmc.Text
Adodc3.Recordset.Fields(9) = txtfc.Text
```

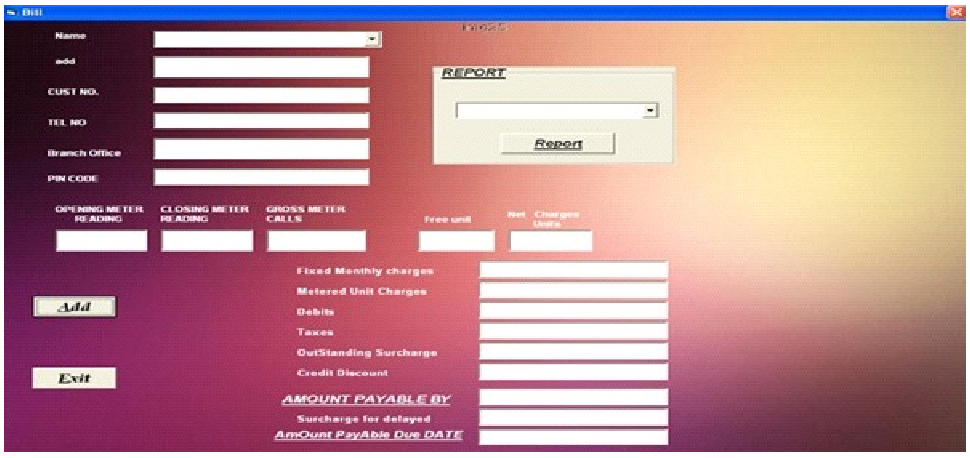
NOTES

```
Adodc3.Recordset.Fields(10) = Txtncc.Text
Adodc3.Recordset.Fields(11) = Txtfmc.Text
Adodc3.Recordset.Fields(12) = Txtmcc.Text
'Adodc3.Recordset.Fields(13) = Txtdb.Text
Adodc3.Recordset.Fields(14) = Txttx.Text
'Adodc3.Recordset.Fields(15) = Txtdb1.Text
Adodc3.Recordset.Fields(18) = Txtapb.Text
Adodc3.Recordset.Fields(19) = Txtsfdp.Text
Adodc3.Recordset.Fields(20) = Txtapdd.Text
'Adodc1.Recordset.Save
'Adodc2.Recordset.Save
Adodc3.Recordset.Save
MsgBox "BILL SAVE Successfully"
Adodc3.Refresh
While Adodc3.Recordset.EOF = False
Combo1.AddItem (Adodc3.Recordset!Name)
Adodc3.Recordset.MoveNext
Wend

'Val(Txtgmc.Text) = Val(Txtcmr.Text) - Val(Txtomr.Text)
'End
End Sub

Private Sub cmdx_Click()
Unload Me
End Sub

Private Sub Combo1_LostFocus()
'Text2.Text = Combo1.Text
'Adodc3.Refresh
'On Error Resume Next
'If DataEnvironment1.con1.State = 1 Then
DataEnvironment1.con1.Open
'DataEnvironment1.con1.Close
'DataEnvironment1.con1.Open
'DataEnvironment1.Bill_details (Text2.Text)
''DataReport3.Show
'BillReport.Show
End Sub
```

```
     Private Sub Command1_Click()
     Text2.Text = Combo1.Text
     Adodc3.Refresh
     On Error Resume Next
     If DataEnvironment1.con1.State = 1 Then
DataEnvironment1.con1.Open
     DataEnvironment1.con1.Close
     DataEnvironment1.con1.Open
     DataEnvironment1.Bill_details (Text2.Text)
     'DataReport3.Show
     BillReport.Show
     End Sub

     Private Sub Form_Load()
     While Adodc1.Recordset.EOF = False
     Cmbnm2.AddItem (Adodc1.Recordset!Name)
     Adodc1.Recordset.MoveNext
     Wend
     txtfc.Text = ""
     Txtfmc.Text = ""

     'Label5.Caption = DateTime.Month(Date) & "/" &
DateTime.Year(Date)
     Adodc3.Refresh
     While Adodc3.Recordset.EOF = False
     Combo1.AddItem (Adodc3.Recordset!Name)
     Adodc3.Recordset.MoveNext
     Wend

     cmdsv.Visible = False
     'Val(Txtgmc.Text) = Val(Txtcmr.Text) - Val(Txtomr.Text)
     End Sub

     Private Sub Frame1_DragDrop(Source As Control, X As Single,
Y As Single)
     'BillReport.Show
     End Sub

     Private Sub Label5_Click()
     End Sub

     Private Sub Txtgmc_GotFocus()
     Txtgmc.Text = Val(Txtcmr.Text) - Val(Txtomr.Text)
```

NOTES

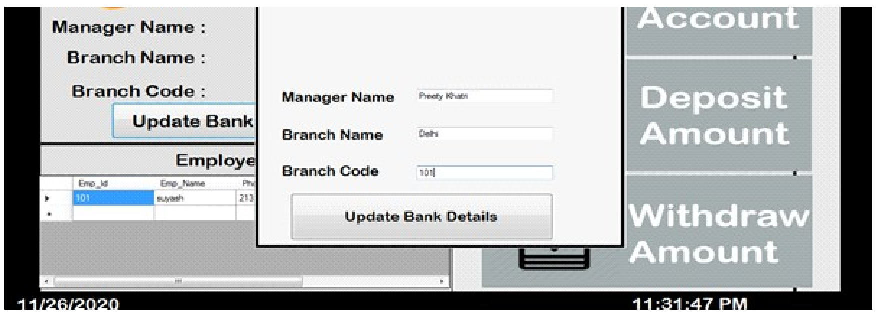
```
     Txtncc.Text = Val(Txtgmc.Text) - Val(txtfc.Text)
     If Txtncc.Text <= 0 Then
     Txtncc.Text = "0"
     Txtmcc.Text = Txtncc.Text
     Else
     Txtmcc.Text = Txtncc.Text
     End If
     Txttx.Text = (Val(Txtfmc.Text) + Val(Txtmcc.Text)) * 0.1023
     Txttx.Text = Round(Txttx.Text)
     Txtapb.Text  =  Val(Txttx.Text)  +  Val(Txtfmc.Text)  +
Val(Txtmcc.Text)
     If Val(Txtapb.Text) > 0 Then
     Txtsfdp.Text = "10"
     Txtapdd.Text = Val(Txtapb.Text) + Val(Txtsfdp.Text)
     Else
     MsgBox "Wrong Bill Amount"
     End If
     End Sub

     Private Sub Txtomr_GotFocus()
     Do While Adodc2.Recordset.EOF = False
     If Adodc2.Recordset!planname = Text1.Text Then
     'Txtmcc.Text = Adodc2.Recordset!MonthlyCharges
     txtfc.Text = Adodc2.Recordset!free_calls
     Exit Do
     End If
     Adodc2.Recordset.MoveNext
     Loop
     End Sub
```



6. Bank Transaction System

Bank Details:

```
     Public Class bankd
      Private Sub Label2_Click(sender As Object, e As EventArgs)
      End Sub

       Private Sub PictureBox1_Click(sender As Object, e As
EventArgs)
      End Sub

      Private Sub cls_Click(sender As Object, e As EventArgs)
Handles cls.Click
      Me.Close()
      End Sub

       Private  Sub  Button1_Click(sender  As  Object,  e  As
EventArgs) Handles Button1.Click
      home.managername.Text = TextBox1.Text
      home.brnamee.Text = TextBox2.Text
      home.Label6.Text = TextBox3.Text
      MsgBox("Bank Details Updated")
      End Sub

      Private Sub RectangleShape1_Click(sender As Object, e
As EventArgs) Handles RectangleShape1.Click
      End Sub
       Private  Sub  PictureBox1_Click_1(ByVal  sender  As
System.Object, ByVal e As System.EventArgs)
      End Sub
     End Class
```



Manager Name : Preety Khatri

Branch Name : Delhi

Branch Code : 101

NOTES

Deposit:

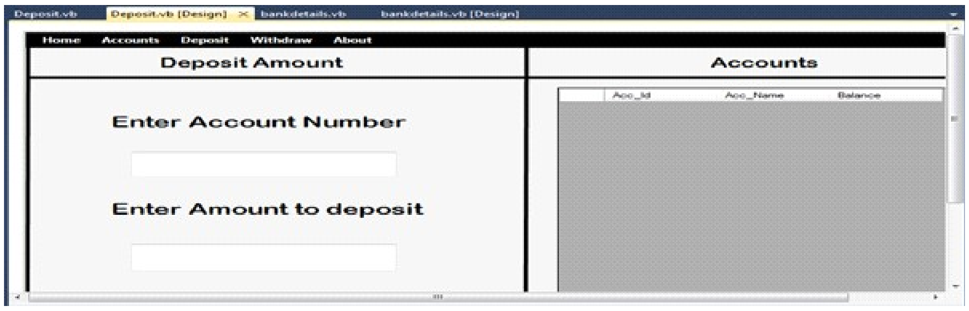
```
    Public Class Deposit
     Private Sub cls_Click(sender As Object, e As EventArgs)
Handles cls.Click
     Me.Hide()
     End Sub

     Private Sub Deposit_Load(sender As Object, e As EventArgs)
Handles MyBase.Load
      'TODO: This line of code loads data into the
'BankaccountsDataSet.baccounts' table. You can move, or remove
it, as needed.
     Me.BaccountsTableAdapter.Fill(Me.BankaccountsDataSet.baccounts)
     dat.Text = Date.Now.ToString("MM/dd/yyyy")
     Timer1.Start()
     End Sub

     Private Sub Label3_Click(sender As Object, e As EventArgs)
Handles Label3.Click
     End Sub

     Private Sub dat_Click(sender As Object, e As EventArgs)
Handles dat.Click
     End Sub

      Private Sub Button1_Click(sender As Object, e As
EventArgs) Handles Button1.Click
     Me.Close()
     End Sub

     Private Sub Timer1_Tick(sender As Object, e As EventArgs)
Handles Timer1.Tick
     clock.Text = TimeOfDay
     End Sub

      Private Sub Button2_Click(sender As Object, e As
EventArgs) Handles Button2.Click
     Dim con As New OleDb.OleDbConnection
     con.ConnectionString = "PROVIDER = Microsoft. Ace. OLEDB.
12.0; Data Source =F:\Sem.4\extra vs code\bankmanagementsystem\
bankmanagementsystem\project\ BankManageMentSystem\
BankManageMentSystem\bankaccounts.accdb"
```

```
    Dim SqlString As String = "update [baccounts] set [Balance]
= Balance+@TextBox2.Text where [Acc_Id] = @TextBox1.Text"

    Using conn As New OleDb.OleDbConnection(con.
ConnectionString)
      Using cmd As New OleDb.OleDbCommand(SqlString, con)
      cmd.CommandType = CommandType.Text
      cmd.Parameters.AddWithValue("column", TextBox2.Text)
      cmd.Parameters.AddWithValue("column", TextBox1.Text)
      con.Open()
      MsgBox("Amount Deposited Successfully")
      cmd.ExecuteNonQuery()
      Me.DataGridView1.Refresh()
      TextBox2.Text = ""
      TextBox1.Text = ""
      End Using
      End Using
      End Sub

      Private Sub HomeToolStripMenuItem_Click(sender As Object,
e As EventArgs) Handles HomeToolStripMenuItem.Click
      home.Show()
      End Sub

      Private Sub AccountsToolStripMenuItem_Click(sender As
Object, e As EventArgs) Handles AccountsToolStripMenuItem.Click
      End Sub

      Private Sub AddAccountToolStripMenuItem_Click(sender As
Object, e As EventArgs) Handles AddAccountToolStripMenuItem.
Click
      addaccount.Show()
      End Sub

      Private Sub UpdateAccountToolStripMenuItem_Click(sender
As Object, e As EventArgs) Handles UpdateAccountToolStrip
MenuItem.Click
      updateaccount.Show()
      End Sub

      Private Sub DeleteAccountToolStripMenuItem_Click(sender
As Object, e As EventArgs) Handles DeleteAccount
ToolStripMenuItem.Click
```

NOTES

NOTES

```
        deleteaccount.Show()
        End Sub

        Private Sub DepositToolStripMenuItem_Click(sender As
Object, e As EventArgs) Handles DepositToolStripMenuItem.Click
        Me.Show()
        End Sub

        Private Sub WithdrawToolStripMenuItem_Click(sender As
Object, e As EventArgs) Handles WithdrawToolStripMenuItem.Click
        Withdraw.Show()

        End Sub

       Private Sub RegisterProductToolStripMenuItem_Click(sender
As Object, e As EventArgs) Handles RegisterProduct
ToolStripMenuItem.Click
        Register.Show()
        End Sub

        Private Sub CreditsToolStripMenuItem_Click(sender As
Object, e As EventArgs) Handles CreditsToolStripMenuItem.Click
        about.Show()
        End Sub

        Private Sub HelpToolStripMenuItem_Click(sender As Object,
e As EventArgs) Handles HelpToolStripMenuItem.Click
        Help.Show()
        End Sub

        Private Sub AboutToolStripMenuItem_Click(sender As
Object, e As EventArgs) Handles AboutToolStripMenuItem.Click
        End Sub
       End Class
```



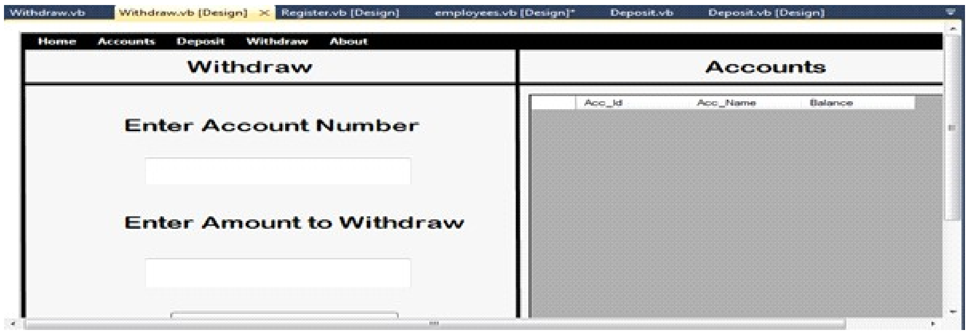
Withdraw:

```
    Public Class Withdraw
     Private Sub AddAccountToolStripMenuItem_Click(sender As
Object, e As EventArgs) Handles AddAccountToolS
tripMenuItem.Click
     End Sub

     Private Sub cls_Click(sender As Object, e As EventArgs)
Handles cls.Click
     Me.Hide()
     End Sub

      Private Sub Withdraw_Load(sender As Object, e As
EventArgs) Handles MyBase.Load
      'TODO: This line of code loads data into the
'BankaccountsDataSet.baccounts' table. You can move, or remove
it, as needed.
     Me.BaccountsTableAdapter.Fill(Me.BankaccountsDataSet.baccounts)
     dat.Text = Date.Now.ToString("MM/dd/yyyy")
     Timer1.Start()
     End Sub

     Private Sub dat_Click(sender As Object, e As EventArgs)
Handles dat.Click
     End Sub

     Private Sub Timer1_Tick(sender As Object, e As EventArgs)
Handles Timer1.Tick
     clock.Text = TimeOfDay
     End Sub

      Private Sub Button2_Click(sender As Object, e As
EventArgs) Handles Button2.Click
     Dim con As New OleDb.OleDbConnection
     con.ConnectionString = "PROVIDER = Microsoft.Ace.OLEDB.
12.0; Data Source =F:\Sem.4\extra vs code\bankmanagementsystem\
bankmanagementsystem\project\BankManageMentSystem
\BankManageMentSystem\ bankaccounts.accdb"
    Dim SqlString As String = "update [baccounts] set [Balance]
= Balance-@TextBox2.Text where [Acc_Id] = @TextBox1.Text"
    Using conn As New OleDb.OleDbConnection
(con.ConnectionString)
```

NOTES

NOTES

```
    Using cmd As New OleDb.OleDbCommand(SqlString, con)
    cmd.CommandType = CommandType.Text
    cmd.Parameters.AddWithValue("column", TextBox2.Text)
    cmd.Parameters.AddWithValue("column", TextBox1.Text)
    con.Open()
    MsgBox("Amount Withdrawn Successfully")
    cmd.ExecuteNonQuery()
    Me.DataGridView1.Refresh()
    TextBox2.Text = ""
    TextBox1.Text = ""
    End Using
    End Using
    End Sub
   End Class
```



7. Payroll Processing

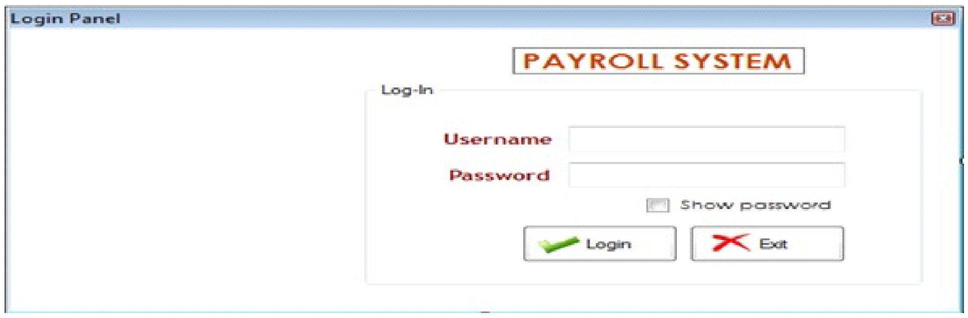
Login:

```
    Imports System.Data.OleDb
    Public Class frmloginA


    Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button1.Click
    Dim con As New System.Data.OleDb.OleDbConnection("Provider
= Microsoft.jet.OleDB.4.0;Data Source = " & Application.
StartupPath & "\datastorage.mdb;")
     Dim cmd As OleDbCommand = New OleDbCommand( _
     "SELECT * FROM logininfo WHERE Username = '" & _
     TextBox1.Text & "' AND [Password] = '" & txtPassword.Text
& "' ", con)
     con.Open()
     Dim sdr As OleDbDataReader = cmd.ExecuteReader()
```

```
      If (sdr.Read() = True) Then
      MessageBox.Show("You are Now Logged In")
      frmMainA.Show()
      TextBox1.Focus()
      TextBox1.Clear()
      txtPassword.Clear()
      Me.Hide()
      Else
      MessageBox.Show("Invalid Username or Password!")
      End If
      End Sub

      Private Sub Button2_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button2.Click
      If MsgBox("Do you want to switch user?", vbYesNo +
vbQuestion) = vbYes Then
      Me.Hide()
      TextBox1.Clear()
      txtPassword.Clear()
      Frmchoose.Show()
      End If
      End Sub

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

      Private Sub CheckBox1_CheckedChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
CheckBox1.CheckedChanged
      If CheckBox1.Checked = True Then
      txtPassword.PasswordChar = ""
      Else
      txtPassword.PasswordChar = "•"
      End If
      End Sub

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

NOTES

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

      Private Sub GroupBox1_Enter(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles GroupBox1.Enter
      End Sub

      Private Sub PictureBox1_Click_1(ByVal sender As
System.Object, ByVal e As System.EventArgs)
      End Sub
     End Class
```



Form Main:

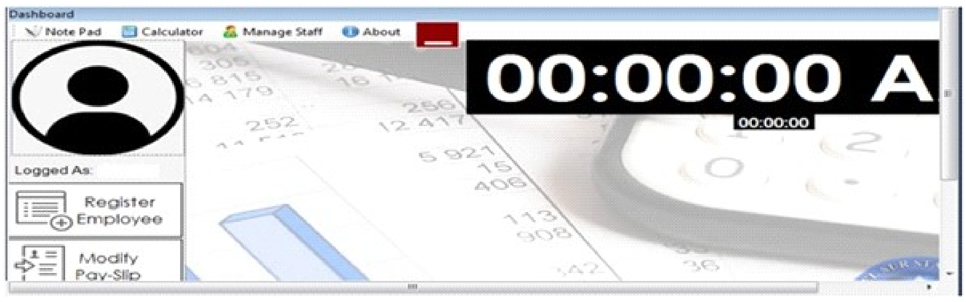
```
     Imports System.IO
     Public Class frmMainA
      Private Sub Timer1_Tick(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Timer1.Tick
      lblTime.Text = DateTime.Now.ToString("hh:mm:ss tt")
      lblDate.Text = DateTime.Now.ToString("MMMM dd yyyy")
      End Sub

      Private Sub frmmainuser_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
      Label2.Text = frmloginA.TextBox1.Text
      Timer1.Start()
      End Sub

      Private Sub btnMaintenance_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnMaintenance.Click
      Try
      Dim fbd As New FolderBrowserDialog
      If fbd.ShowDialog() = vbOK Then
      File.Copy("GenerallPayroll.accdb", fbd.SelectedPath &
"\GenerallPayroll.accdb")
      MsgBox("Done...")
      End If
      Catch ex As Exception
      MsgBox(ex.Message)
      End Try
      End Sub

      Private Sub btnMini_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnMini.Click
      Me.WindowState = FormWindowState.Minimized
      End Sub

      Private Sub btnLogout_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnLogout.Click
      btnLogout.BackColor = Color.White
      btnLogout.ForeColor = Color.Black
      If MsgBox("Do you want to switch user?", vbYesNo +
vbQuestion) = vbYes Then
      Me.Hide()
      Frmchoose.Show()
      End If
      End Sub

      Private Sub NotePadToolStripMenuItem_Click(ByVal sender
As System.Object, ByVal e As System.EventArgs) Handles
NotePadToolStripMenuItem.Click
      Try
      System.Diagnostics.Process.Start("Notepad.exe")
      Catch ex As Exception
          MessageBox.Show(ex.Message, "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
      End Try
      End Sub

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

NOTES

```
Try
System.Diagnostics.Process.Start("Calc.exe")
Catch ex As Exception
    MessageBox.Show(ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
End Try
End Sub

Private Sub SystemInfoToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles SystemInfoToolStripMenuItem.Click
End Sub

Private Sub btnCataloging_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnCataloging.Click
frmregister.Show()
End Sub

Private Sub btnCirculation_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnCirculation.Click
frmpayslip.Show()
End Sub

Private Sub AddStaffToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles AddStaffToolStripMenuItem.Click
frmaddstaff.Show()
End Sub

Private Sub RemoveStaffToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles RemoveStaffToolStripMenuItem.Click
frmremovestaff.Show()
End Sub

Private Sub ToolStripMenuItem1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ToolStripMenuItem1.Click
About.Show()
End Sub
```

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

      Private Sub SearchRecordsToolStripMenuItem_Click(ByVal
sender As System.Object, ByVal e As System.EventArgs)
      End Sub
     End Class
```

NOTES



Print Slip:

```
     Public Class frmpayslip
     Private Sub GenPayFinalBindingNavigatorSaveItem_
Click(ByVal sender As System.Object, ByVal e As System.
EventArgs)
      Me.Validate()
      Me.GenPayFinalBindingSource.EndEdit()
      Me.TableAdapterManager.UpdateAll(Me.GenerallPayrollDataSet)
      End Sub

      Private Sub frmpayslip_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
       'TODO: This line of code loads data into the
'GenerallPayrollDataSet.GenPayFinal' table. You can move, or
remove it, as needed.
     Me.GenPayFinalTableAdapter.Fill(Me.GenerallPayrollDataSet.
GenPayFinal)
     End Sub

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

NOTES

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

    Private Sub Button5_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button5.Click
    Me.Validate()
    Me.GenPayFinalBindingSource.EndEdit()
    Me.TableAdapterManager.UpdateAll(Me.GenerallPayrollDataSet)
    MessageBox.Show("Successfully Added")
    End Sub

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

    Private Sub btnDeleteJHS_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnDeleteJHS.Click
    Try
    If PlantIDTextBox.Text = "" Then
    MessageBox.Show("Please select employee id", "Entry",
MessageBoxButtons.OK, MessageBoxIcon.Warning)
    Exit Sub
    End If
    If PlantIDTextBox.Text.Count > 0 Then
    If MessageBox.Show("Do you really want to delete the
record?" & vbCrLf & "You can not restore the record" & vbCrLf
& "It will delete record permanently" & vbCrLf & "related to
selected employee", "Warning!!!", MessageBoxButtons.YesNo,
MessageBoxIcon.Warning) = Windows.Forms.DialogResult.Yes Then
    GenPayFinalBindingSource.RemoveCurrent()
    Me.TableAdapterManager.UpdateAll(Me.GenerallPayrollDataSet)
    End If
    End If

    Catch ex As Exception
    MessageBox.Show(ex.Message, "Error", MessageBoxButtons.
OK, MessageBoxIcon.Error)
      End Try
      End Sub

      Private Sub Button2_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button2.Click
      txtReceipt.Text = ""
      txtReceipt.AppendText("" + vbNewLine)
      txtReceipt.AppendText("" + vbNewLine)
      txtReceipt.AppendText("" + vbNewLine)
      txtReceipt.AppendText("" + vbNewLine)
      txtReceipt.AppendText("" + vbNewLine)
      txtReceipt.AppendText("" + vbNewLine)
      txtReceipt.AppendText("" + vbNewLine)
      txtReceipt.AppendText(vbTab + vbTab + vbTab + vbTab +
vbTab + vbTab & "PAY-SLIP" + vbNewLine)
      txtReceipt.AppendText("" + vbNewLine)
      txtReceipt.AppendText("" + vbNewLine)
      txtReceipt.AppendText("" + vbNewLine)
      txtReceipt.AppendText("Plantilla Number: " + vbTab &
PlantIDTextBox.Text + vbTab + vbTab + vbTab + vbNewLine)
      txtReceipt.AppendText("Employee Name: " + vbTab &
EmployeeNameTextBox.Text + vbTab + vbTab + vbNewLine)
      txtReceipt.AppendText("Number: " + vbTab + vbTab &
NoTextBox.Text + vbNewLine)
      txtReceipt.AppendText("" + vbNewLine)
      txtReceipt.AppendText("" + vbNewLine)
      txtReceipt.AppendText("Basic Salary: " + vbTab &
BasicTextBox.Text + vbNewLine)
      txtReceipt.AppendText("Pera: " + vbTab + vbTab &
PERATextBox.Text + vbNewLine)
      txtReceipt.AppendText("Gross Amount: " + vbTab &
GrossAmountTextBox.Text + vbNewLine)
      txtReceipt.AppendText("= = = = = = = = = = = = = = =
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
= = = = = = = " + vbNewLine)
      txtReceipt.AppendText("" + vbNewLine)
      txtReceipt.AppendText(vbTab + vbTab & "Deductions" +
```

NOTES

NOTES

```
vbNewLine)
      txtReceipt.AppendText("" + vbNewLine)
      txtReceipt.AppendText("W/ Tax: " + vbTab + vbTab + vbTab
& WtaxTextBox.Text + vbNewLine)
      txtReceipt.AppendText("GSIS Premium: " + vbTab + vbTab
& GSISPremiumTextBox.Text + vbNewLine)
      txtReceipt.AppendText("GSIS Salary Loan: " + vbTab &
GSISSalaryLoanTextBox.Text + vbNewLine)
      txtReceipt.AppendText("GSIS EL: " + vbTab + vbTab &
GSISELTextBox.Text + vbNewLine)
      txtReceipt.AppendText("GSIS EMRGL: " + vbTab + vbTab &
GSISEMRGLTextBox.Text + vbNewLine)
      txtReceipt.AppendText("GSIS PL: " + vbTab + vbTab &
GSISPLTextBox.Text + vbNewLine)
      txtReceipt.AppendText("Pag-Ibig Premium: " + vbTab &
PagIbigPremTextBox.Text + vbNewLine)
      txtReceipt.AppendText("Pag-Ibig ML: " + vbTab + vbTab &
PagIbigMLTextBox.Text + vbNewLine)
      txtReceipt.AppendText("Pag-Ibig 2: " + vbTab + vbTab &
PagIbig2TextBox.Text + vbNewLine)
      txtReceipt.AppendText("Phil Health Premium: " + vbTab &
PhilHealthPremiunTextBox.Text + vbNewLine)
      txtReceipt.AppendText("LEAP: " + vbTab + vbTab + vbTab
& LEAPTextBox.Text + vbNewLine)
      txtReceipt.AppendText("IGP: " + vbTab + vbTab + vbTab &
IGPTextBox.Text + vbNewLine)
      txtReceipt.AppendText("Faculty Union: " + vbTab + vbTab
& FacultyUnionTextBox.Text + vbNewLine)
      txtReceipt.AppendText("Refund Disallow: " + vbTab &
RefundDisallowTextBox.Text + vbNewLine)
      txtReceipt.AppendText("Tuition: " + vbTab + vbTab +
vbTab & TuitionTextBox.Text + vbNewLine)
      txtReceipt.AppendText("LBP Payment: " + vbTab + vbTab &
LBPPaymentTextBox.Text + vbNewLine)
      txtReceipt.AppendText("City Savings: " + vbTab + vbTab
& CitySavingsTextBox.Text + vbNewLine)
      txtReceipt.AppendText("" + vbNewLine)
      txtReceipt.AppendText("Total Deductions: " + vbTab &
TotalDeductionTextBox.Text + vbTab + vbTab & "NET Amount: " +
vbTab & NETAmountTextBox.Text + vbNewLine)
```

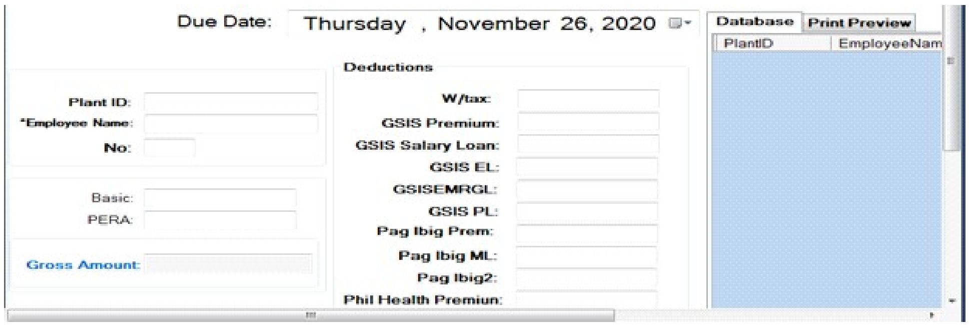
```
      txtReceipt.AppendText("= = = = = = = = = = = = = = = =
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
= = = = = = = " + vbNewLine)
      txtReceipt.AppendText(vbTab & "Due Date: " + Today &
vbTab + vbTab + vbTab + vbTab + vbTab + vbTab & "Time: " &
TimeOfDay + vbNewLine)
      txtReceipt.AppendText("= = = = = = = = = = = = = = = =
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
= = = = = = = " + vbNewLine)
      txtReceipt.AppendText("" + vbNewLine)
      txtReceipt.AppendText("" + vbNewLine)
      txtReceipt.AppendText("" + vbNewLine)
      txtReceipt.AppendText("" + vbNewLine)
      txtReceipt.AppendText(vbTab + "Recieve by:" + vbNewLine)
      txtReceipt.AppendText(vbTab + vbTab + vbTab +
"____________________" + vbNewLine)
      txtReceipt.AppendText(vbTab + vbTab + vbTab +
EmployeeNameTextBox.Text + vbNewLine)
      txtReceipt.AppendText(vbTab + vbTab + vbTab + " Employee"
+ vbNewLine)
      txtReceipt.AppendText("" + vbNewLine)
      txtReceipt.AppendText("" + vbNewLine)
      txtReceipt.AppendText("" + vbNewLine)
      txtReceipt.AppendText("= = = = = = = = = = = = = = = =
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
= = = = = = = " + vbNewLine)
      txtReceipt.AppendText(" Need Help? Contact Us: 09096510899
" + vbNewLine)
      txtReceipt.AppendText("= = = = = = = = = = = = = = = =
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
= = = = = = = " + vbNewLine)
      txtReceipt.AppendText(vbTab + vbTab + vbTab +
PictureBox1.Text + vbNewLine)
      PrintPreviewDialog1.ShowDialog()
      End Sub


      Private Sub PrintDocument1_PrintPage(ByVal sender As
System.Object, ByVal e As System.Drawing.Printing.
PrintPageEventArgs) Handles PrintDocument1.PrintPage
```

NOTES

```
      e.Graphics.DrawString(txtReceipt.Text, Font,
Brushes.Black, 140, 140)
      e.Graphics.DrawImage(Me.PictureBox1.Image, 120, 130,
PictureBox1.Width - 15, PictureBox1.Height - 25)
      e.Graphics.DrawImage(Me.PictureBox2.Image, 300, 130,
PictureBox2.Width - 15, PictureBox2.Height - 25)
      End Sub
      Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button1.Click
      TotalDeductionTextBox.Text = Val(WtaxTextBox.Text) +
Val(GSISPremiumTextBox.Text) + Val(GSISSalaryLoanTextBox.Text)
+ Val(GSISELTextBox.Text) + Val(GSISEMRGLTextBox.Text) +
Val(GSISPLTextBox.Text) + Val(PagIbigPremTextBox.Text) +
Val(PagIbigMLTextBox.Text) + Val(PagIbig2TextBox.Text) +
Val(PhilHealthPremiunTextBox.Text) + Val(LEAPTextBox.Text) +
Val(IGPTextBox.Text) + Val(FacultyUnionTextBox.Text) +
Val(RefundDisallowTextBox.Text) + Val(TuitionTextBox.Text) +
Val(LBPPaymentTextBox.Text) + Val(CitySavingsTextBox.Text)
      GrossAmountTextBox.Text = Val(BasicTextBox.Text) +
Val(PERATextBox.Text)
      NETAmountTextBox.Text = Val(GrossAmountTextBox.Text) -
Val(TotalDeductionTextBox.Text)
     NETAmountTextBox.Text = FormatCurrency(NETAmountTextBox.
Text)
     TotalDeductionTextBox.Text = FormatCurrency(Total
DeductionTextBox.Text)
     GrossAmountTextBox.Text = FormatCurrency(Gross
AmountTextBox.Text)
      MessageBox.Show("Successfully Computed")
      End Sub

      Private Sub Button9_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button9.Click
      Me.TableAdapterManager.UpdateAll(Me.GenerallPayrollDataSet)
      Me.Close()
      End Sub
```

NOTES

```
      Private Sub Button8_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button8.Click
      GenPayFinalBindingSource.MovePrevious()
      End Sub

      Private Sub Button7_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button7.Click
      GenPayFinalBindingSource.MoveNext()
      End Sub

       Private  Sub  TextBox14_TextChanged(ByVal  sender  As
System.Object,  ByVal  e  As  System.EventArgs)  Handles
TextBox14.TextChanged
      Me.GenPayFinalBindingSource.Filter = "PlantID LIKE '" &
TextBox14.Text & "%'"
      End Sub
     End Class
```


Due Date: Thursday , November 26, 2020

8. Personal Information System

Main:

```
     Imports System.Data.OleDb
     Public Class frmmain
      Dim Oledr As OleDbDataReader
      Dim Item As New ListViewItem()
      Dim ItemSearch As New ListViewItem
      Private Sub frmmain_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
      Call ListStudentColumns(lststudent)
```

NOTES

```
Call openconnection()
Call Initialized()
Call LoadListView()
Call closeconnection()
End Sub
Public Sub LoadListView()
lststudent.Items.Clear()
Call Initialized()
Oledr = OleDa.SelectCommand.ExecuteReader()
Do While Oledr.Read()
Item = lststudent.Items.Add(Oledr("studentno").ToString())
Item.SubItems.Add(Oledr("firstname").ToString())
Item.SubItems.Add(Oledr("lastname").ToString())
Item.SubItems.Add(Oledr("course").ToString())

Loop
Oledr.Close()
End Sub

Private Sub btnAdd_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnAdd.Click
frmadd.ShowDialog()
End Sub
Private Function UpdateValidateStudent() As Boolean
If lststudent.Items.Count = 0 Then
MsgBox("No records.", MsgBoxStyle.Information, "No
Records")
Return True
Exit Function
End If
If lststudent.SelectedItems.Count > 1 Then
MsgBox("Double click the record",
MsgBoxStyle.Information)
lststudent.SelectedItems.Clear()
Return True
Exit Function
End If
```

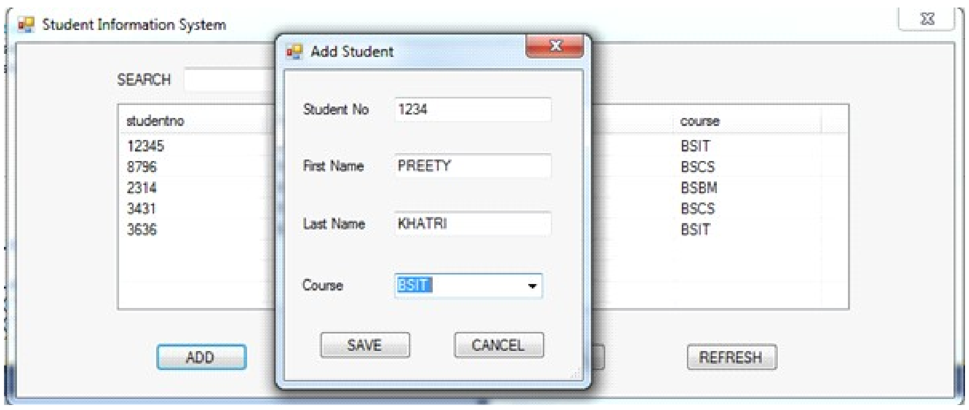
```
      If lststudent.SelectedItems.Count = 0 Then
      MsgBox("Please choose the record you want to edit",
MsgBoxStyle.Information)
      Return True
      Exit Function
      End If
      End Function

      Private Sub btnEdit_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnEdit.Click
      If UpdateValidateStudent() = True Then
      Return
      End If
      frmedit.ShowDialog()
      End Sub
      Private Function DeleteStudentValidate() As Boolean
      If lststudent.Items.Count = 0 Then
      MsgBox("No Records to delete")
      Return True
      Exit Function
      End If
      If lststudent.SelectedItems.Count = 0 Then
      MsgBox("Please choose the record you want to delete.")
      Return True
      Exit Function
      End If
      End Function

      Private Sub btnDelete_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnDelete.Click
      If DeleteStudentValidate() = True Then
      Return
      End If

      If MsgBox("Do you really want to delete this record?",
MsgBoxStyle.YesNo + MsgBoxStyle.Question, "Delete?") =
MsgBoxResult.No Then
```

NOTES

NOTES

```
MsgBox("Delete Cancelled.", MsgBoxStyle.Information)
lststudent.SelectedItems.Clear()
Exit Sub
End If
For Each Item As ListViewItem In lststudent.SelectedItems
Item.Remove()
OleDa.DeleteCommand = New OleDbCommand()
Call openconnection()
OleDa.DeleteCommand.CommandText = "DELETE FROM tblstudent
WHERE studentno = @studentno"
OleDa.DeleteCommand.Connection = OleCn
OleDa.DeleteCommand.Parameters.Add("@studentno",
OleDbType.VarChar, 50, "studentno").Value =
Item.Text.ToString()
OleDa.DeleteCommand.ExecuteNonQuery()
Call LoadListView()
Call closeconnection()
Next
MsgBox("Record Deleted")
lststudent.SelectedItems.Clear()
End Sub

Private Sub btnRefresh_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnRefresh.Click
Call openconnection()
Call Initialized()
Call LoadListView()
Call closeconnection()
txtSearch.Clear()
MsgBox("Total Records = " & lststudent.Items.Count,
MsgBoxStyle.Information, "Record")
End Sub
Private Sub SearchStudent()
lststudent.Items.Clear()
Call Initialized()
OleDa.SelectCommand.CommandText = "SELECT * FROM
tblstudent WHERE studentno Like '%%" & txtSearch.Text.
Trim.ToString() & "%%'"
```

```
      OleDa.SelectCommand.Connection = OleCn
      Oledr = OleDa.SelectCommand.ExecuteReader()
      Do While Oledr.Read()
     ItemSearch = lststudent.Items.Add(Oledr("studentno").
ToString())
      ItemSearch.SubItems.Add(Oledr("firstname").ToString())
      ItemSearch.SubItems.Add(Oledr("lastname").ToString())
      ItemSearch.SubItems.Add(Oledr("course").ToString())


      Loop
      Oledr.Close()
      End Sub


       Private Sub txtSearch_TextChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
txtSearch.TextChanged
      OleDa.SelectCommand = New OleDbCommand()
       OleDa.SelectCommand.CommandText = "SELECT * FROM
tblstudent WHERE studentno Like '%%'"
      OleDa.SelectCommand.Connection = OleCn
      Call openconnection()
      OleDa.SelectCommand.ExecuteNonQuery()
      Call SearchStudent()
      Call closeconnection()
      End Sub
     End Class
```



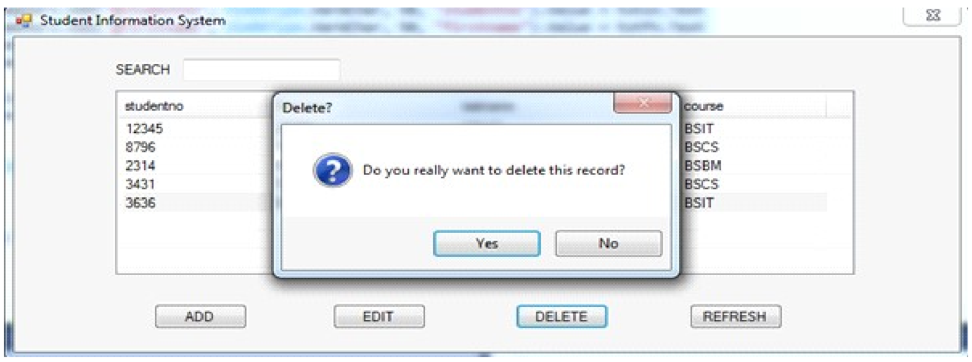
studentno	course
12345	BSIT
8796	BSCS
2314	BSBM
3431	BSCS
3636	BSIT

Student No 1234
First Name PREETY
Last Name KHATRI
Course BSIT

Add Information:

```
     Imports System.Data.OleDb
     Public Class frmadd
      Private Sub frmadd_FormClosing(ByVal sender As Object,
ByVal e As System.Windows.Forms.FormClosingEventArgs) Handles
Me.FormClosing
      Call cleartext()
      txtsn.Focus()
      frmmain.lststudent.SelectedItems.Clear()
      End Sub

      Private Sub frmadd_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
      End Sub
      Private Sub cleartext()
      Me.txtsn.Clear()
      Me.txtfn.Clear()
      Me.txtln.Clear()
      End Sub

      Private Sub btnCancel_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnCancel.Click
      Me.Close()
      End Sub

      Private Sub btnSave_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnSave.Click
      If txtsn.Text = "" Or txtfn.Text = "" Or cmbcourse.Text
= "" Then
       MsgBox("Please don't leave blank textfields",
MsgBoxStyle.Information, "Missing data")
      Exit Sub
      End If
      Try
      Call openconnection()
      OleDa.InsertCommand = New OleDbCommand()
      OleDa.InsertCommand.CommandText = "INSERT INTO tblstudent
(studentno, firstname, lastname, course)" & _
      "VALUES (@studentno , @firstname, @lastname, @course)"
      OleDa.InsertCommand.Connection = OleCn
       OleDa.InsertCommand.Parameters.Add("@studentno",
OleDbType.VarWChar, 50, "studentno").Value = txtsn.Text
       OleDa.InsertCommand.Parameters.Add("@firstname",
OleDbType.VarWChar, 50, "firstname").Value = txtfn.Text
       OleDa.InsertCommand.Parameters.Add("@lastname",
OleDbType.VarWChar, 50, "lastname").Value = txtln.Text
       OleDa.InsertCommand.Parameters.Add("@course",
OleDbType.VarWChar, 50, "course").Value = cmbcourse.Text
      OleDa.InsertCommand.ExecuteNonQuery()
      Call frmmain.LoadListView()
      Call closeconnection()
      MsgBox("Records Saved", MsgBoxStyle.Information, "Saved")
      Me.Close()
      Catch ex As Exception
      MsgBox("Cannot Save this record, Existing Student Number",
MsgBoxStyle.Information, "Error")
      Call closeconnection()
      txtsn.Focus()
      txtsn.SelectAll()
      End Try
      End Sub
     End Class
```

Delete Record:


Student Information System

SEARCH

studentno	course
12345	BSIT
8796	BSCS
2314	BSBM
3431	BSCS
3636	BSIT

Delete?

Do you really want to delete this record?

NOTES

Edit Record:

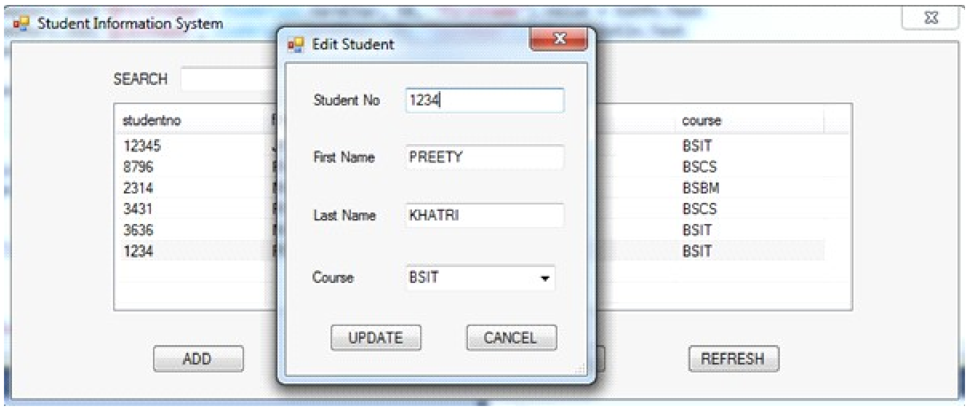
```
     Imports System.Data.OleDb
     Public Class frmedit
      Private Sub frmedit_FormClosing(ByVal sender As Object,
ByVal e As System.Windows.Forms.FormClosingEventArgs) Handles
Me.FormClosing
      Call cleartext()
      txtsn.Focus()
      frmmain.lststudent.SelectedItems.Clear()
      End Sub

      Private Sub frmedit_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
      Call openconnection()
      Call Initialized()
     txtsn.Text = CStr(frmmain.lststudent.SelectedItems(0).
Text)
      Call Fill()
      Call closeconnection()
      End Sub
      Private Sub cleartext()
      Me.txtsn.Clear()
      Me.txtfn.Clear()
      Me.txtln.Clear()
      End Sub
      Private Sub Fill()
      Dim OleDr As OleDbDataReader
      OleDa.SelectCommand = New OleDbCommand()
       OleDa.SelectCommand.CommandText = "SELECT * From
tblstudent WHERE studentno = @studentno"
       OleDa.SelectCommand.Parameters.Add("@studentno",
OleDbType.VarWChar, 50, "studentno").Value = txtsn.Text
      OleDa.SelectCommand.Connection = OleCn
      OleDr = OleDa.SelectCommand.ExecuteReader()
```

NOTES

```
      If OleDr.HasRows() Then
      OleDr.Read()
      txtsn.Text = OleDr("studentno").ToString()
      txtfn.Text = OleDr("firstname").ToString()
      txtln.Text = OleDr("lastname").ToString()
      cmbcourse.Text = OleDr("course").ToString()
      End If
      OleDr.Close()
      End Sub

      Private Sub btnCancel_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnCancel.Click
      Me.Close()
      End Sub

      Private Sub btnSave_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnSave.Click
      If txtsn.Text = "" Or txtfn.Text = "" Or txtln.Text = ""
Or cmbcourse.Text = "" Then
      MsgBox("Dont leave blank textfields")
      Exit Sub
      End If
      Try
      Call openconnection()
      OleDa.UpdateCommand = New OleDbCommand()
      OleDa.UpdateCommand.CommandText = "UPDATE tblstudent SET
studentno = @studentno, firstname = @firstname, lastname =
@lastname, course = @course WHERE studentno = ?"
      OleDa.UpdateCommand.Connection = OleCn
       OleDa.UpdateCommand.Parameters.Add("@studentno",
OleDbType.VarWChar, 50, "studentno").Value = txtsn.Text
       OleDa.UpdateCommand.Parameters.Add("@firstname",
OleDbType.VarWChar, 50, "firstname").Value = txtfn.Text
       OleDa.UpdateCommand.Parameters.Add("@lastName",
OleDbType.VarWChar, 50, "lastName").Value = txtln.Text
```

NOTES

```
        OleDa.UpdateCommand.Parameters.Add("@Course",
OleDbType.VarWChar, 50, "Course").Value = cmbcourse.Text
        OleDa.UpdateCommand.Parameters.Add(New
System.Data.OleDb.OleDbParameter("EmpID",
System.Data.OleDb.OleDbType.VarWChar, 50, _
      System.Data.ParameterDirection.Input, False, CType(0,
Byte), CType(0, Byte), "studentno", _
      System.Data.DataRowVersion.Original, Nothing)).Value =
frmmain.lststudent.SelectedItems(0).Text
      OleDa.UpdateCommand.ExecuteNonQuery()
      Call frmmain.LoadListView()
      Call closeconnection()
      MsgBox("Records Updated")
      Me.Close()
      Catch ex As Exception
      MsgBox("Cannot Update StudentNo is present")
      Call closeconnection()
      txtsn.Focus()
      txtsn.SelectAll()
      End Try
      End Sub
     End Class
```



studentno	course
12345	BSIT
8796	BSCS
2314	BSBM
3431	BSCS
3636	BSIT
1234	BSIT

Student No: 1234
First Name: PREETY
Last Name: KHATRI
Course: BSIT

9. Question Database and Conducting Quiz

Register:

```
     Public Class Form2
```

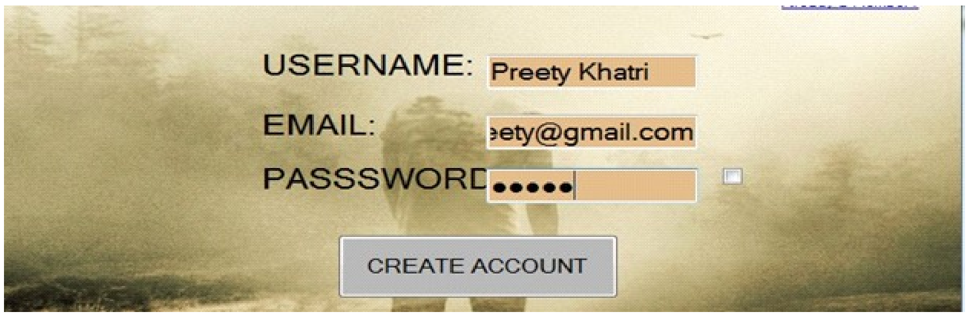
```
      Private Sub Form2_Load(sender As Object, e As EventArgs)
Handles MyBase.Load
      End Sub

      Private Sub LinkLabel1_LinkClicked(sender As Object, e
As LinkLabelLinkClickedEventArgs)
      SIGN_IN.Show()
      Me.Close()
      End Sub

       Private  Sub  Button1_Click(sender  As  Object,  e  As
EventArgs)
      Home.Show()
      Me.Close()
      End Sub

       Private  Sub  Button2_Click(sender  As  Object,  e  As
EventArgs)
      End Sub

       Private  Sub  GroupBox1_Enter(sender  As  Object,  e  As
EventArgs)
      End Sub

       Private  Sub  Button3_Click(sender  As  Object,  e  As
EventArgs)
      quest6.Show()
      End Sub

       Private  Sub  Button1_Click_1(sender  As  Object,  e  As
EventArgs) Handles Button1.Click
      My.Settings.Username = username1.Text
      My.Settings.Password = password1.Text
      My.Settings.Save()
      MsgBox("Your Account Has Been Created")
      SIGN_IN.Show()
```

NOTES

```
        Me.Close()
        End Sub
```

NOTES

```
        Private Sub LinkLabel1_LinkClicked_1(sender As Object,
e As LinkLabelLinkClickedEventArgs) Handles
LinkLabel1.LinkClicked
        SIGN_IN.Show()
        Me.Close()
        End Sub

        Private Sub Button2_Click_1(sender As Object, e As
EventArgs) Handles Button2.Click
        Form1.Show()
        End Sub

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

        If CheckBox1.Checked Then
        password1.UseSystemPasswordChar = False
        Else
        password1.UseSystemPasswordChar = True
        End If
        End Sub
       End Class
```



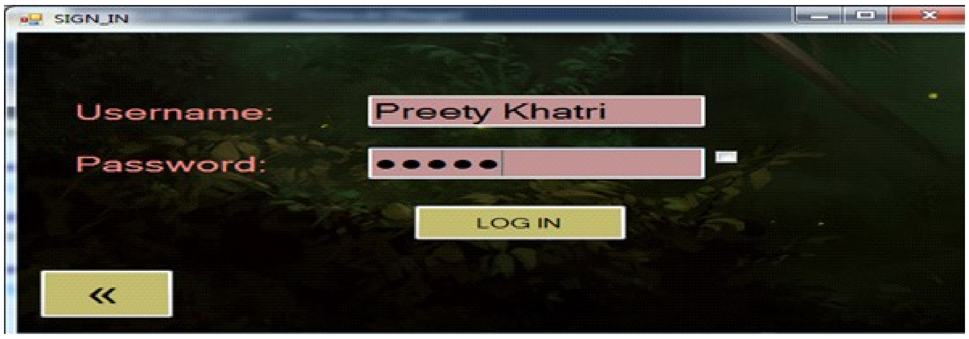
USERNAME: Preety Khatri
EMAIL: ɘety@gmail.com
PASSSWORD ●●●●●

Sign In:

```
   Public Class SIGN_IN
    Private Sub Button1_Click(sender As Object, e As
EventArgs) Handles Button1.Click
    If username2.Text = My.Settings.Username And
    password2.Text = My.Settings.Password = True Then
    Home.Show()
    Me.Close()
    Else
    MsgBox("Incorrect Username Or Password")
    username2.Clear()
    password2.Clear()
    End If
    End Sub

    Private Sub Button2_Click(sender As Object, e As
EventArgs) Handles Button2.Click
    Form1.Show()
    Me.Close()
    End Sub

    Private Sub Button3_Click(sender As Object, e As
EventArgs)
    End Sub

    Private Sub SIGN_IN_Load(sender As Object, e As EventArgs)
Handles MyBase.Load
    End Sub

    Private Sub CheckBox1_CheckedChanged(sender As Object,
e As EventArgs) Handles CheckBox1.CheckedChanged
    If CheckBox1.Checked Then
    password2.UseSystemPasswordChar = False
    Else
```

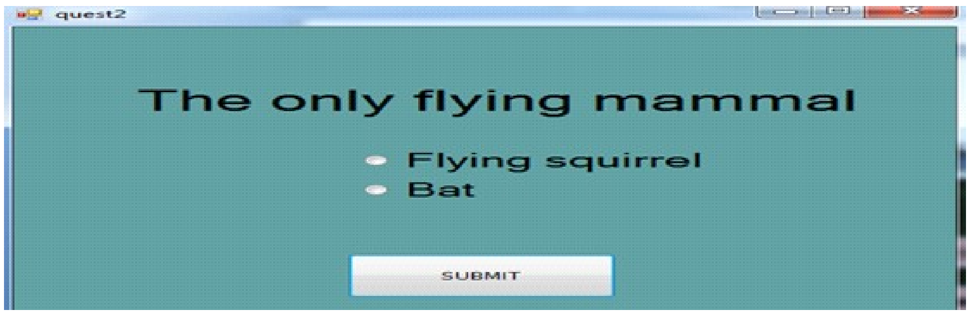
```
      password2.UseSystemPasswordChar = True
      End If
      End Sub
     End Class
```

NOTES



Question:

```
     Public Class quest2
       Private Sub Button2_Click(sender As Object, e As
EventArgs) Handles Button2.Click
      Button2.Invalidate()
      If RadioButton3.Checked Then
      MsgBox("You are correct")
      quest8.LBLRIGHT.Text = quest8.LBLRIGHT.Text + 1
      Else
      MsgBox("You are wrong")
      quest8.LBLWRONG.Text = quest8.LBLWRONG.Text + 1
      End If
      Dim quest6 As New quest2
      Dim quest2 As New quest4
      quest4.Show()
      Me.Hide()
      End Sub

      Private Sub Label2_Click(sender As Object, e As EventArgs)
Handles Label2.Click
      End Sub
      Private Sub RadioButton4_CheckedChanged(sender As Object,
e As EventArgs) Handles RadioButton4.CheckedChanged
      End Sub

      Private Sub RadioButton3_CheckedChanged(sender As Object,
e As EventArgs) Handles RadioButton3.CheckedChanged
      End Sub
     End Class
```



10. Personal Diary

Main:

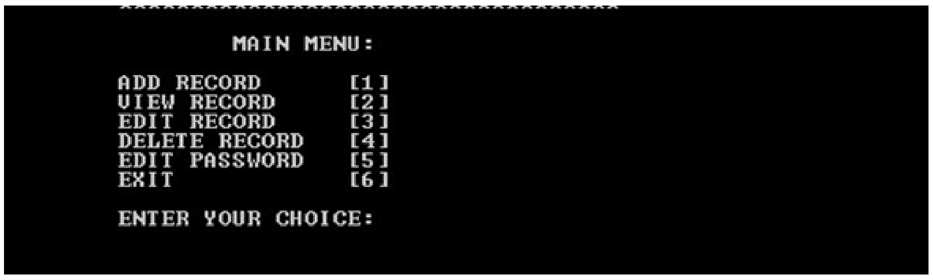
```
     Class clsEntry

      Public Property dtDateOfentry As DateTime
      Public Property strContent As String

      Public Sub New(ByVal dtDate As DateTime, _
      ByVal strText As String)
      dtDateOfentry = dtDate
      strContent = strText
      End Sub

      Public Overrides Function ToString() As String
      Return dtDateOfentry & " " & strContent
      End Function
      End Class
```

NOTES

```
Module Module1
 Sub Main(ByVal args As String())
 Dim objDiary As clsDiary = New clsDiary()
 Dim cSelection As Char = "0"c
 While cSelection <> "4"c
 objDiary.Welcome()
 Console.WriteLine()
 Console.WriteLine("MAIN MENU")
 Console.WriteLine("1 - ADD RECORD")
 Console.WriteLine("2 - VIEW RECORD")
 Console.WriteLine("3 - EDIT RECORD")
 Console.WriteLine("4 - DELETE RECORD")
 Console.WriteLine("5 - EDIT PASSWORD")
 Console.WriteLine("6 - EXIT")
Console.WriteLine("ENTER YOUR CHOICE")

 cSelection = Console.ReadKey().KeyChar
 Console.WriteLine()
 Select Case cSelection
 Case "1"c
 objDiary.Add()
 Case "2"c
 objDiary.View()
 Case "3"c
 objDiary.Edit()
Case "4"c
 objDiary.Delete()
Case "5"c
 objDiary.Edit()
 Case "6"c
 Console.WriteLine("Press any key to exit.")
 Case Else
 Console.WriteLine("Error.")
```

NOTES

```
End Select
Console.ReadKey()
End While
End Sub
End Module
```



```
**********************************
        MAIN MENU:

ADD RECORD      [1]
VIEW RECORD     [2]
EDIT RECORD     [3]
DELETE RECORD   [4]
EDIT PASSWORD   [5]
EXIT            [6]

ENTER YOUR CHOICE:
```

Add/Delete or Search Items:

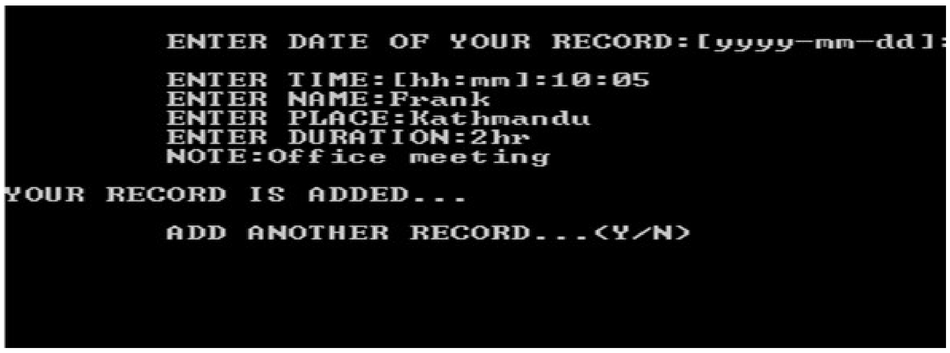
```
 Public Sub Add(ByVal dtDate As DateTime, ByVal strText _
  As String)
  lstEntries.Add(New clsEntry(dtDate, strText))
  End Sub

  Public Sub Delete(ByVal dtDate As DateTime)
  Dim lstResults As List(Of clsEntry) = Find(dtDate, True)
  For Each Entry As clsEntry In lstResults
  lstEntries.Remove(Entry)
  Next
  End Sub

 Public Function Find(ByVal dtDate As DateTime, ByVal
blnTime _
  As Boolean) As List(Of clsEntry)
   Dim lstResults As List(Of clsEntry) = New List(Of
clsEntry)()
  For Each Entry As clsEntry In lstEntries
  If ((blnTime) AndAlso (Entry.dtDateOfentry = _
  dtDate)) OrElse ((Not blnTime) AndAlso _
  (Entry.dtDateOfentry.Date = dtDate.Date))
  Then lstResults.Add(Entry)
 Next
 Return lstResults
 End Function
```

NOTES

```
Class clsDiary
 Private dbData As clsDatabase
 Public Sub New()
 dbData = New clsDatabase()
 End Sub

 Private Function GetDate() As DateTime
 Dim dtDate As DateTime
 While Not DateTime.TryParse(Console.ReadLine(), dtDate)
 Console.WriteLine("Error. Try again:")
 End While
 Return dtDate
 End Function

 Public Sub Print(ByVal dtDay As DateTime)
 Dim lstResults As List(Of clsEntry) = dbData.Find(dtDay,
_
 False)
 For Each Entry As clsEntry In lstResults
 Console.WriteLine(Entry)
 Next
 End Sub

 Public Sub Add()

 Dim dtDate As DateTime = GetDate()
 Console.WriteLine("Enter the entry text:")
 Dim strText As String = Console.ReadLine()
 dbData.Add(dtDate, strText)
 End Sub
```

```
      Public Sub Search()
      Dim dtDate As DateTime = GetDate()
      Dim lstResults As List(Of clsEntry) = dbData.Find(dtDate,
_ False)
      If lstResults.Count() > 0 Then
      Console.WriteLine("Found:")
      For Each Entry As clsEntry In lstResults
      Console.WriteLine(Entry)
      Next
      Else
      Console.WriteLine("Nothing found.")
      End If
      End Sub

      Public Sub Delete()
      Dim dtDate As DateTime = GetDate()
      dbData.Delete(dtDate)
      End Sub
      Public Sub Welcome()
      Console.Clear()
      Console.WriteLine("ENTER DATE OF YOUR RECORD: [yyyy-mm-
dd]:", DateTime.Now))
      Console.WriteLine("ENTER TIME:")
     Console.WriteLine("ENTER NAME:")
     Console.WriteLine("ENTER PLACE:")
     Console.WriteLine("ENTER DURATION:")
     Console.WriteLine("NOTE:")
      Console.WriteLine("ADD ANOTHER RECORD…<Y/N>"
      Print(DateTime.Today)
      Console.WriteLine()
      Print(DateTime.Now.AddDays(1))
      Console.WriteLine()
      End Sub
      End Class
```

NOTES



```
        ENTER DATE OF YOUR RECORD:[yyyy-mm-dd]:

        ENTER TIME:[hh:mm]:10:05
        ENTER NAME:Frank
        ENTER PLACE:Kathmandu
        ENTER DURATION:2hr
        NOTE:Office meeting
YOUR RECORD IS ADDED...

        ADD ANOTHER RECORD...(Y/N)
```