MT264 Revision Questions

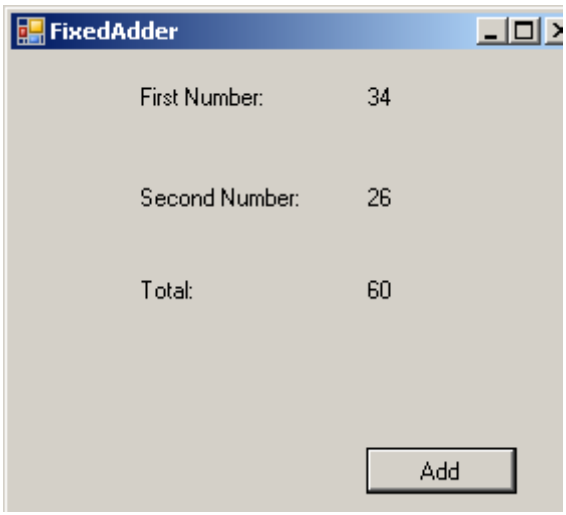
Tutor : Rifat Hamoudi
Staff No. : 00567451
Mobile No. : 0781-2796265

I have put this tutorial on the web. This tutorial can be viewed and downloaded from http://www.rifathamoudi.co.uk then selecting MT264 Tutorials

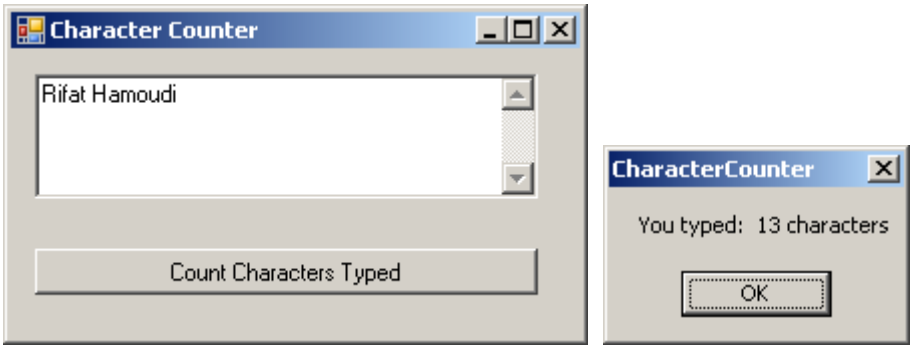
1) Explain with examples what the followings are :

a) Container
b) Controls

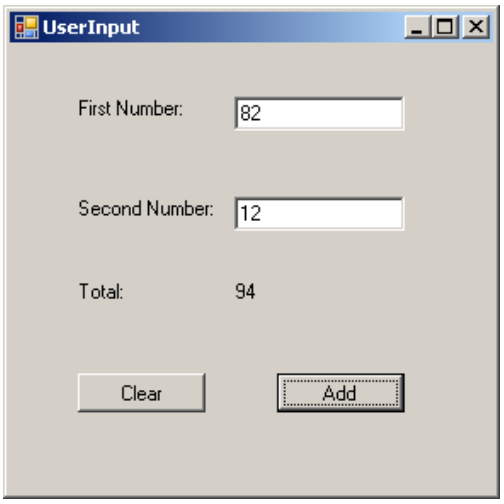
2) Design and implement a Visual Basic program that adds two numbers. An example output is as shown below :



3) Design and implement a Visual Basic program that counts the number of letters in a string. An example output is as shown below :



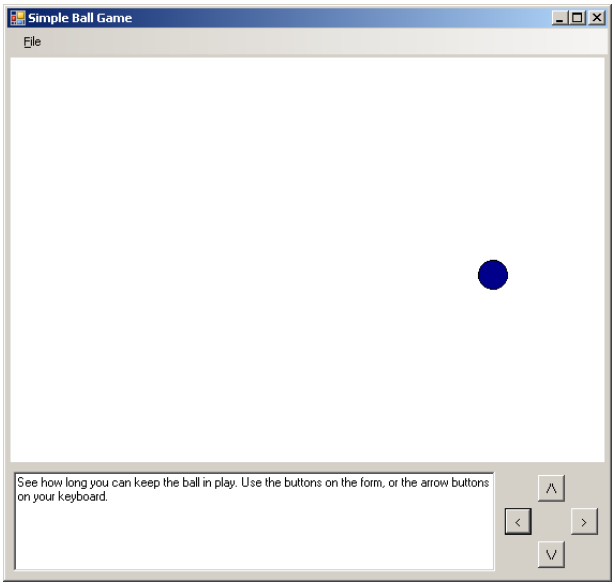
4) Design and implement a Visual Basic program that adds two numbers entered by the user. An example output is as shown below :



5) Design and implement functional code for a project entitled “Traffic Survey” with the following specification:

An application is required for collecting information about traffic from a particular location. More precisely, we wish to record the number of cars, bicycles and lorries passing (in either direction) a particular point at the roadside. The idea is that the user will record each car, bicycle and lorry as they pass. The application will maintain and display the total numbers of each type of vehicle.

6) Design and code a ball game that allows the user to control the movement of the ball. The MainForm should look as follows :



7) Design a GUI for a simple database that can be used to keep track of items by letting the user enter the item ID and the details. Make the design simple but effective.

8) Design and code a fully functional simple database software according to the following specification.

Project Specification : Simple Database

The simple database should have :

a) way to enter new item identifiers and item descriptions in the software
b) way to display the entered items
c) way to save the items entered
d) way to retrieve the items entered without having to re-enter them
e) add instructions/help to the software
f) add the About part for this software

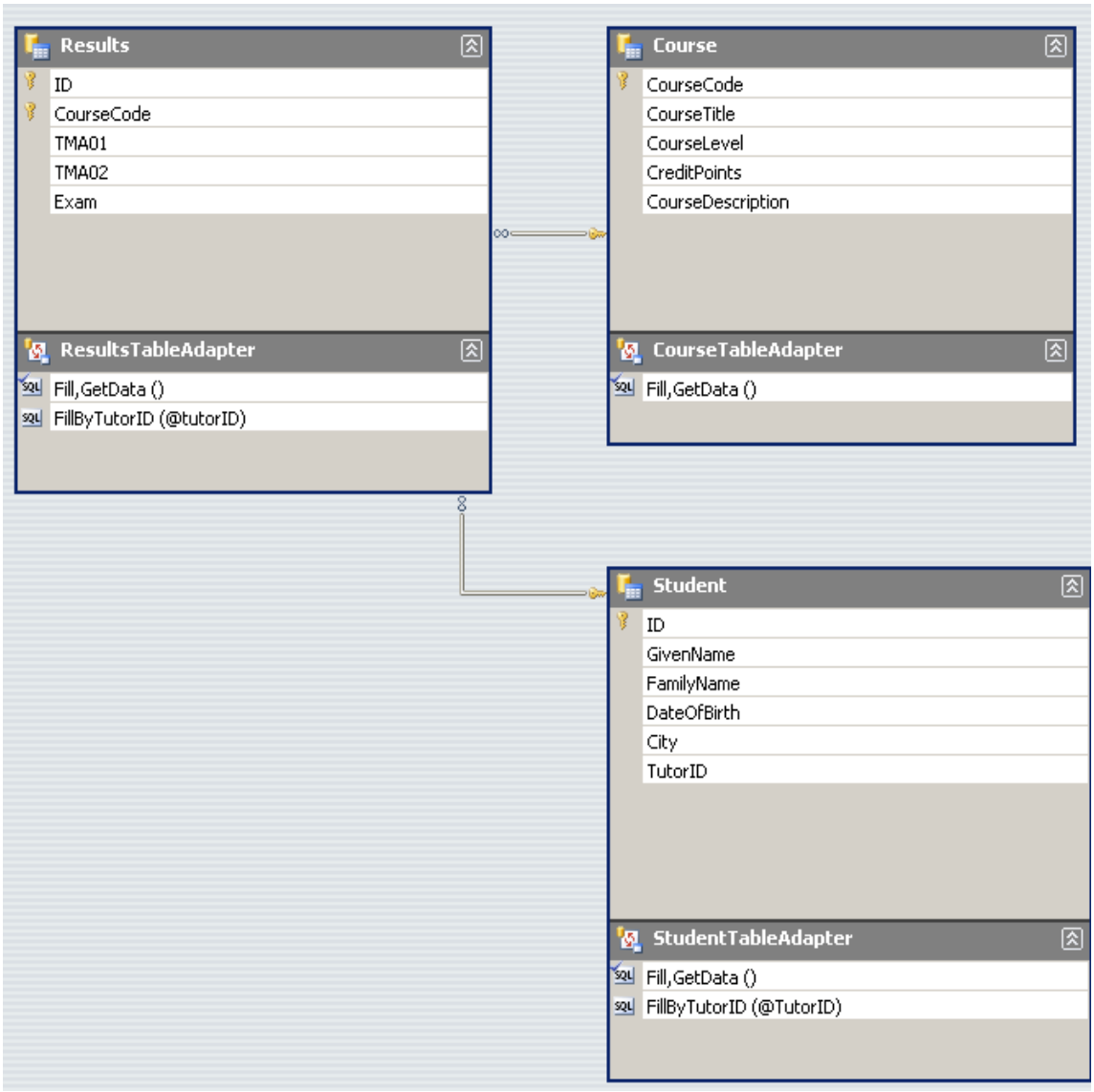
Show the software design, the GUI design and the implementation

Assume that this software is designed for a OU course management database. Test the software by entering the name of the student as item identifier and the course details and average TMA mark as the description. This way we should have a simple database model to keep track of all the students' details in a specific course.

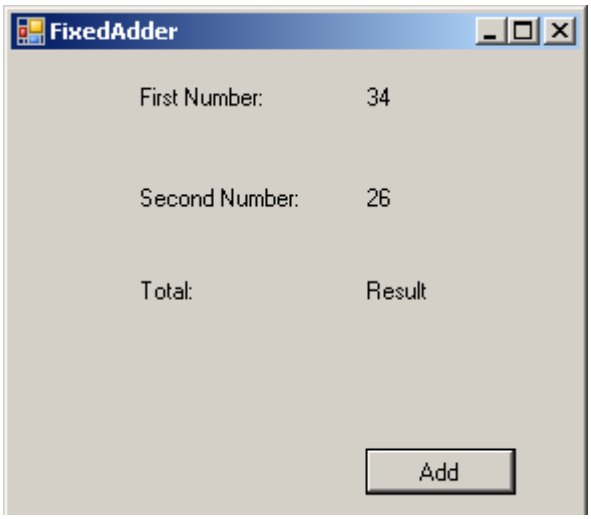
Discuss what is the problem with implementing such a flat-file "database" system.

9) Design and implement fully functional software that uses backend relational database management system to manage OU student records. The software must be able to use two valid tutor IDs T01 and T02. The entity-relationship model and tables are as follows :

Entity-relationship model



Answer to Question 1



The Form “FixedAdder” is a **Container**

There are 7 **Controls** contained within it.

First Number, Second Number and Total are called **Label** controls

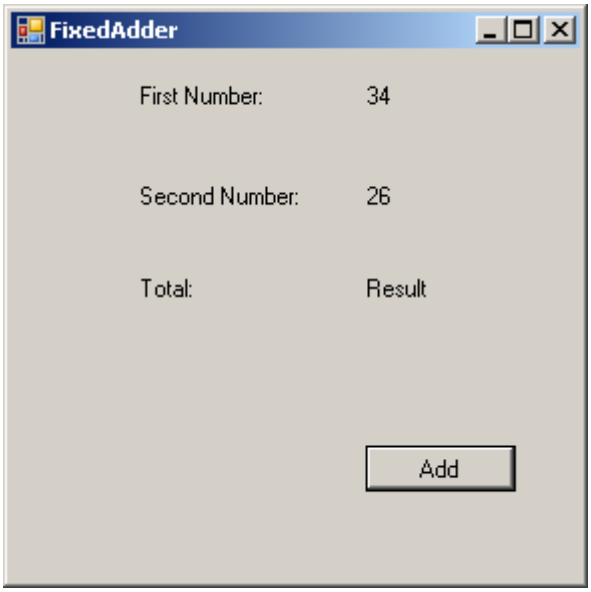
First Number **Label** control has Text property initialised to 34

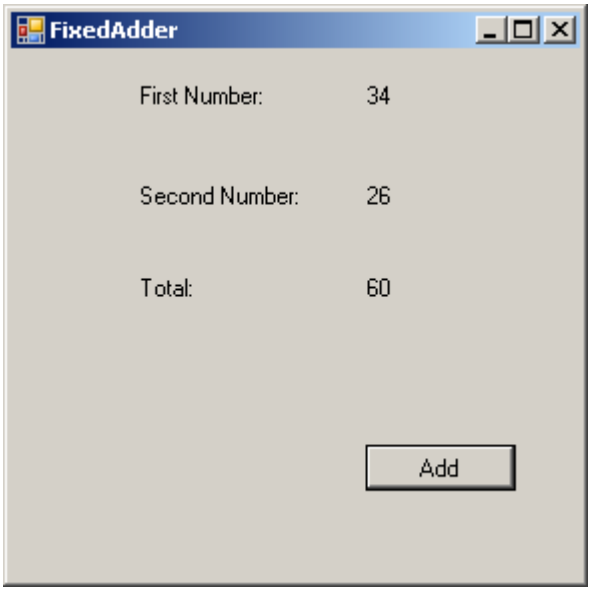
Second Number **Label** control has Text property initialised to 26

Total **Label** control has Text property initialised to “Result”

Button control is initialised to “Add”

Answer to Question 2

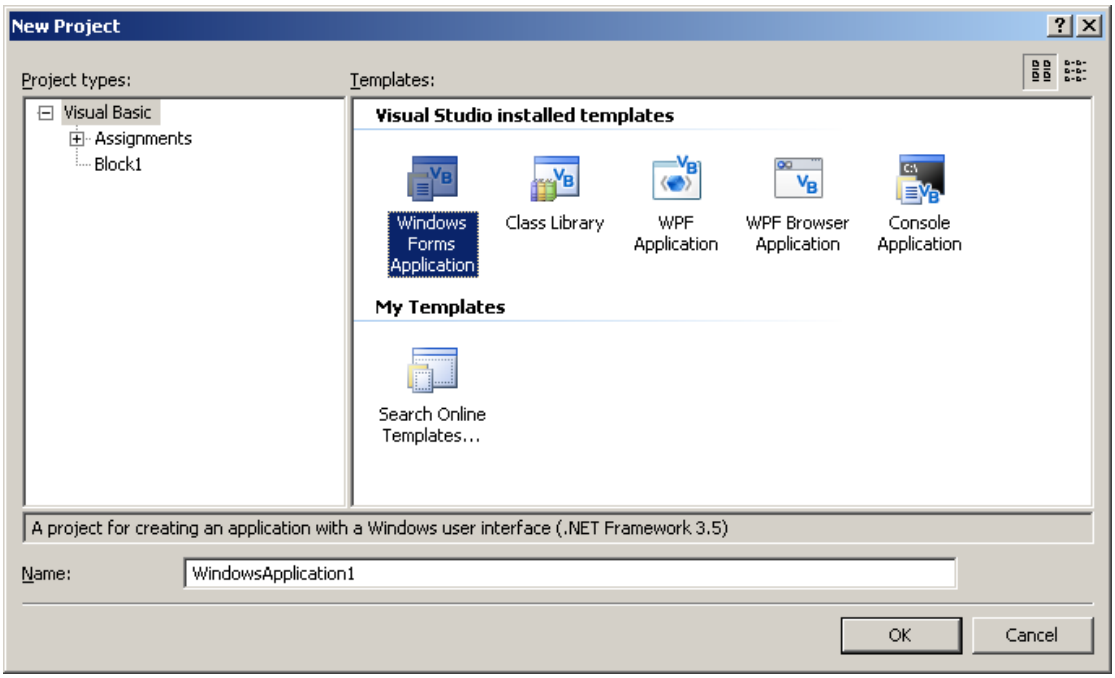





```
Public Class AddForm

    Private Sub AddButton_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles AddButton.Click

        ' A statement to assign the computed total of two labels.
        Sum.Text = Val(Num1.Text) + Val(Num2.Text)

    End Sub
End Class
```

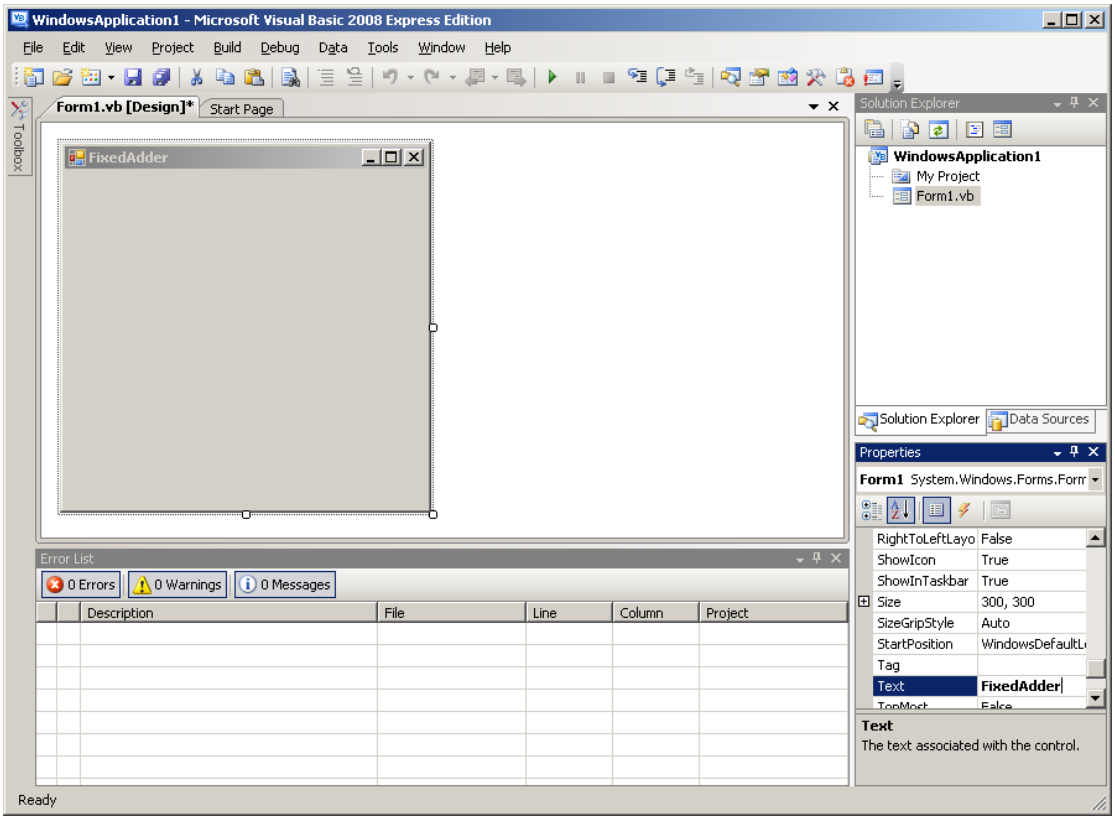
Implementation of the answer to question 2

1) File then New Project

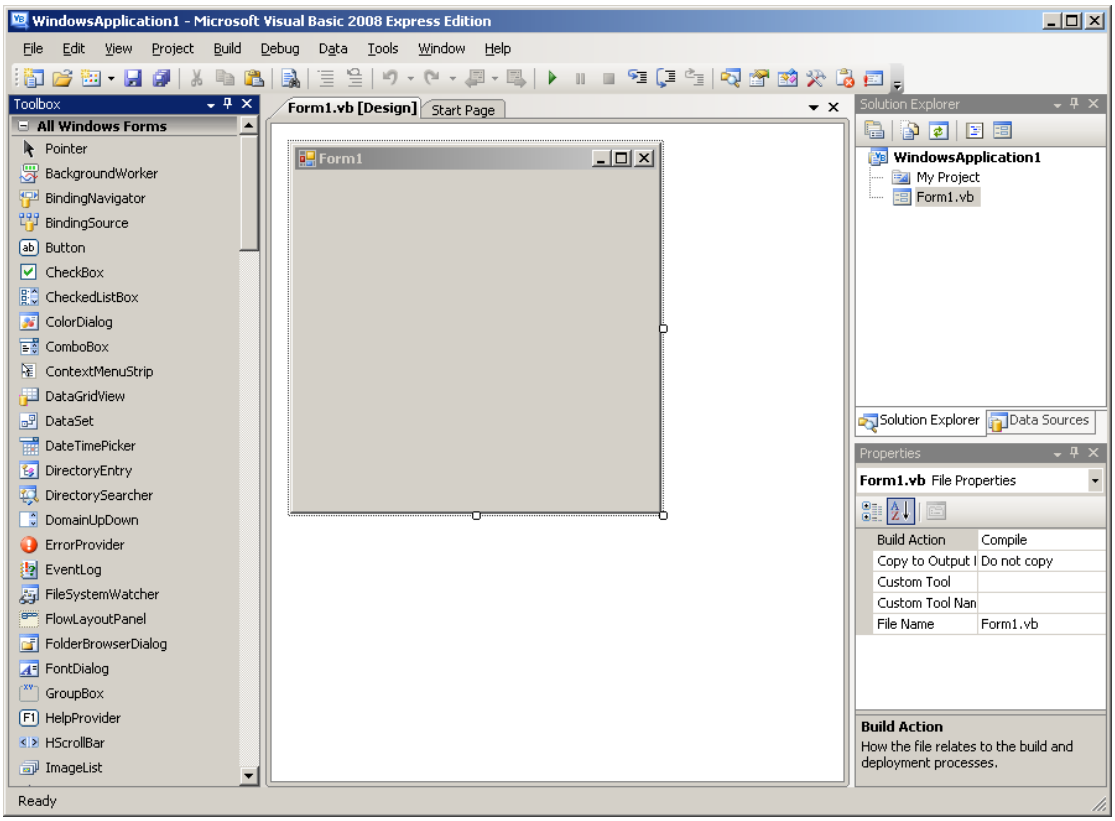


Choose “Windows Forms Application”

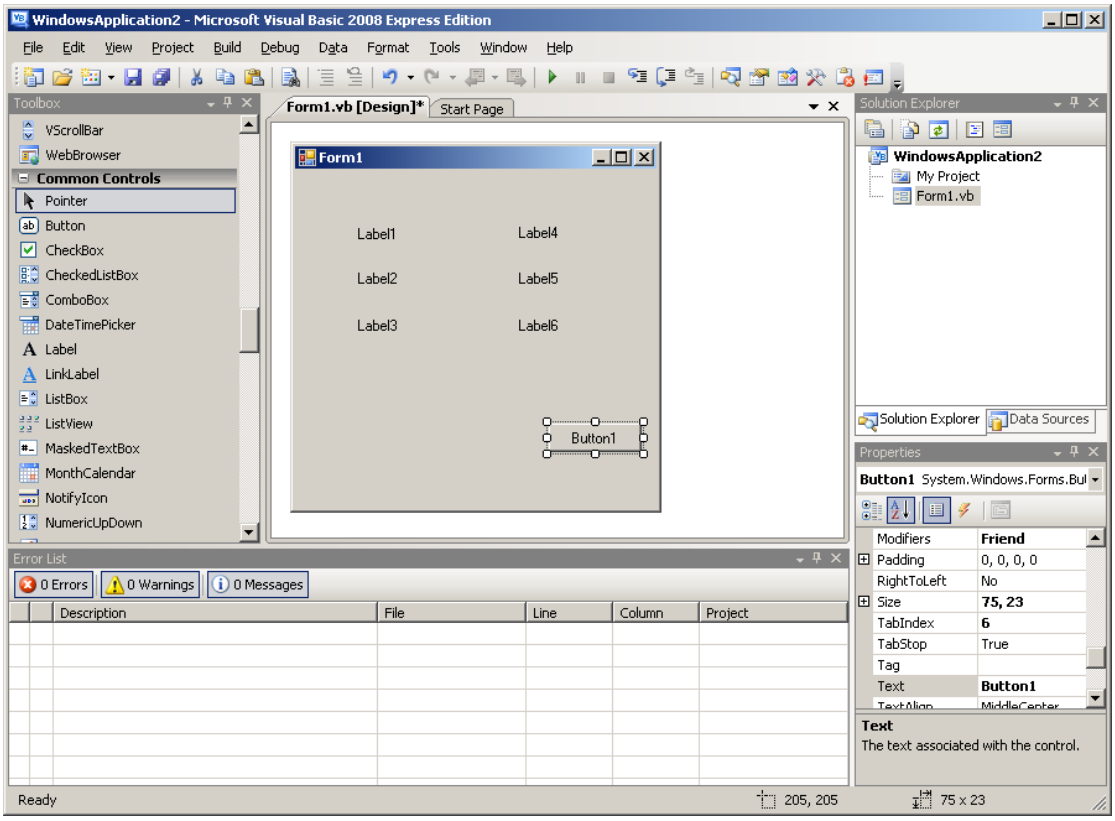
2)



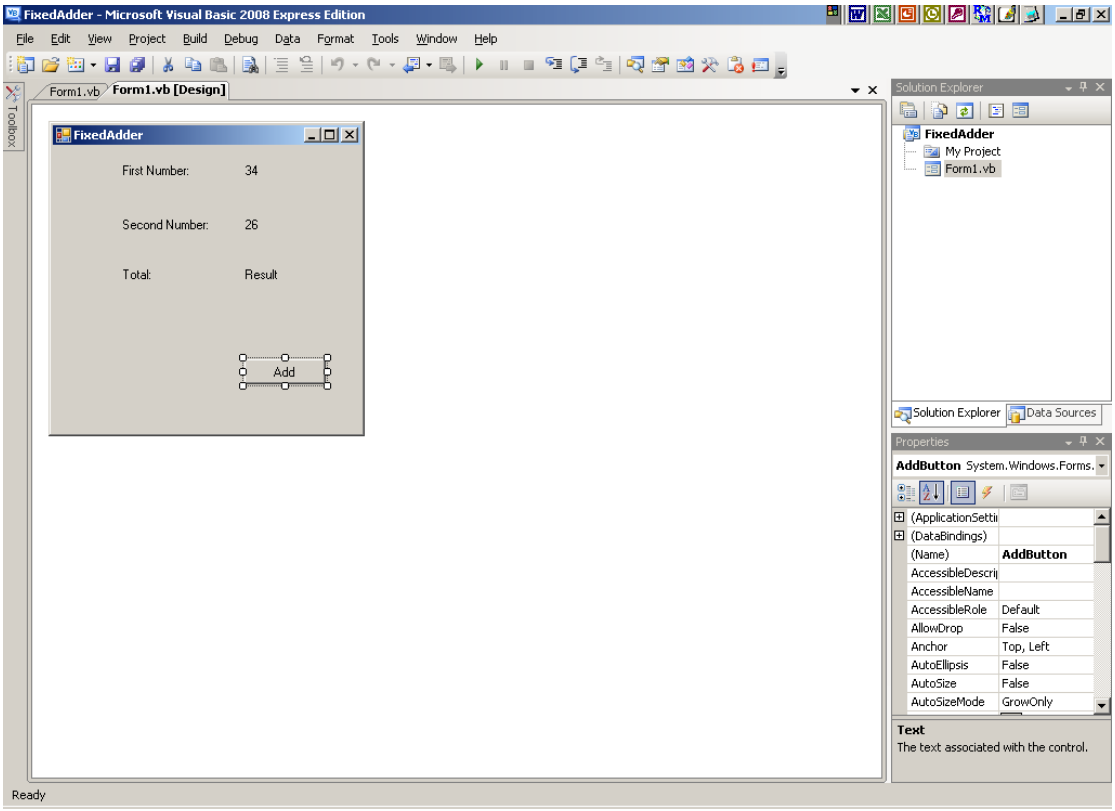
Click on Toolbar and fix it



3) In this example choose 6 Label controls and 1 Button control and drag that onto the Form1 as follows :



4) For the first 3 controls put the names by changing them from the Text property in the Label object. In Label4 change Text to a number for example 34. In the Button1 object change the Text to “Add” and the Name to “AddButton”.



5) Double click on the Add button and you get the event handler and add the following statement :

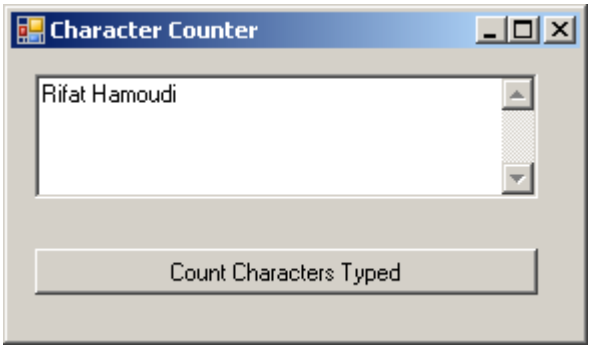
```
Sum.Text = Val(Num1.Text) + Val(Num2.Text)
```

So the final code should look like this :

```
Public Class AddForm

    Private Sub AddButton_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles AddButton.Click

        ' A statement to assign the computed total of two labels.
        Sum.Text = Val(Num1.Text) + Val(Num2.Text)

    End Sub
End Class
```

6) Press F5 or the green button to load the software. Click on the “Add” button and you should see the addition of the 2 numbers

Answer to Question 3

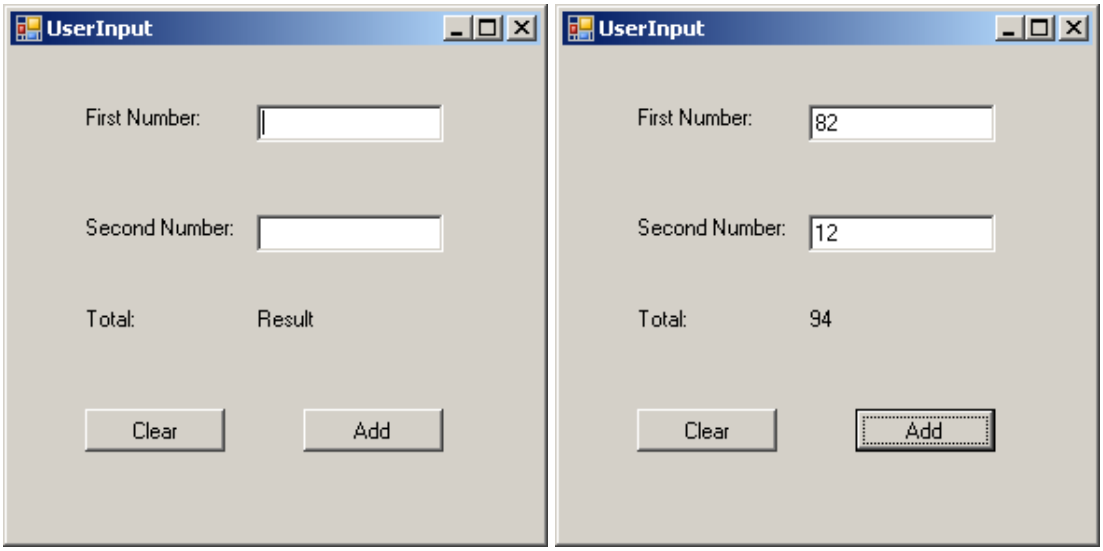



```
Public Class Form1

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

        ' Display the character count of the textbox.
        MsgBox("You typed: " & Str(Len(TextBox1.Text)) & "
characters")

    End Sub
End Class
```

The above code is the efficient type hence it is shorter but less readable. A less efficient (but more readable) code is as follows :

```
Public Class Form1

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

        Dim Lengthofstring As Integer
        Dim StringLength As String

        ' Display the character count of the textbox.

        Lengthofstring = Len(TextBox1.Text)
        StringLength = Str(Lengthofstring)

        MsgBox("You typed: " & StringLength & " characters")

    End Sub
End Class
```

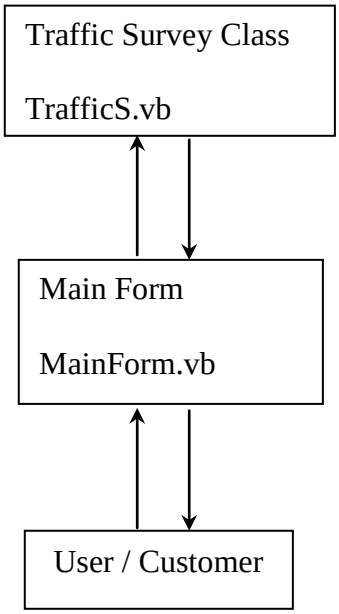
Answer to Question 4

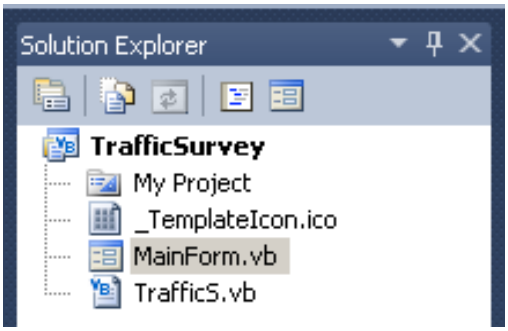



```
Public Class Form1

    Private Sub AddBtn_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles AddBtn.Click

        ' A statement to add user input values and assign the total.
        Sum.Text = Val(Num1.Text) + Val(Num2.Text)

    End Sub

    Private Sub ClearBtn_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles ClearBtn.Click

        ' Statements to resume the initial state.
        Sum.Text = "Result" : Num1.Text = "" : Num2.Text = ""

    End Sub
End Class
```

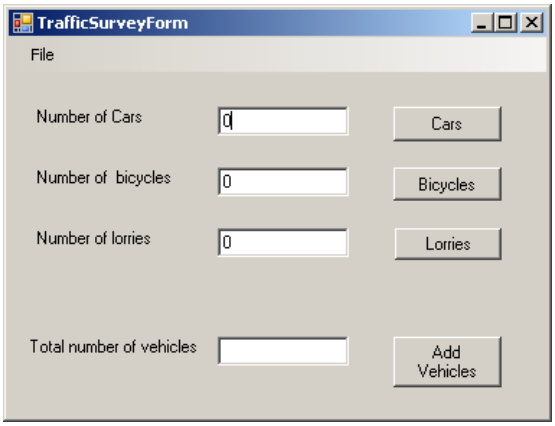
Answer to Question 5

Software design

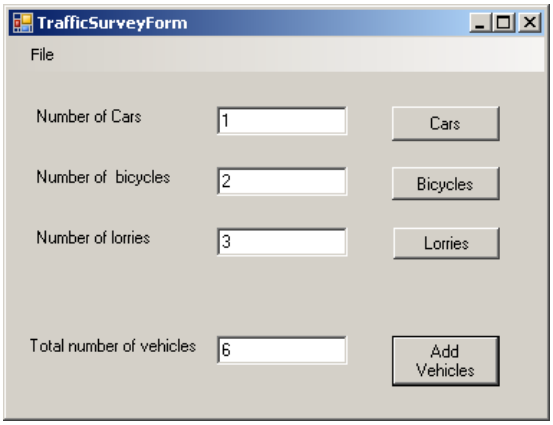




Initial state



User input state



```
Public Class TrafficS
    ' Declare the fields here.

    Private fCars As Integer
    Private fBicycles As Integer
    Private fLorries As Integer

    Public Sub New()
   ' An instance of Traffic is created with all vehicle counts set to zero.

        fCars = 0
        fBicycles = 0
        fLorries = 0

    End Sub

    ' Setters

    Public Sub addCar()
        fCars = fCars + 1
    End Sub

    Public Sub addBicycle()
        fBicycles = fBicycles + 1
    End Sub

    Public Sub addLorry()
        fLorries = fLorries + 1
    End Sub

    'Getters

    Public ReadOnly Property getCar() As Integer
        Get
            Return fCars
        End Get
    End Property

    Public ReadOnly Property getBicycle() As Integer
        Get
            Return fBicycles
        End Get
    End Property

    Public ReadOnly Property getLorry() As Integer
        Get
            Return fLorries
        End Get
    End Property


End Class
```

```
Public Class MainForm
    Private fTrafficSurvey As TrafficS

    Public Sub New()

        fTrafficSurvey = New TrafficS

        ' This call is required by the designer.
        InitializeComponent()

        ' Add any initialization after the InitializeComponent() call.
        updateView()

    End Sub

    Private Sub CarsCount_Click(sender As System.Object, e As
System.EventArgs) Handles CarsCount.Click

        fTrafficSurvey.addCar()
        Car.Text = fTrafficSurvey.getCar()
    End Sub

    Private Sub BicyclesCount_Click(sender As System.Object, e As
System.EventArgs) Handles BicyclesCount.Click

        fTrafficSurvey.addBicycle()
        Bicycle.Text = fTrafficSurvey.getBicycle()
    End Sub

    Private Sub LorriesCount_Click(sender As System.Object, e As
System.EventArgs) Handles LorriesCount.Click

        fTrafficSurvey.addLorry()
        Lorry.Text = fTrafficSurvey.getLorry()
    End Sub

    Private Sub updateView()

        Car.Text = fTrafficSurvey.getCar()
        Bicycle.Text = fTrafficSurvey.getBicycle()
        Lorry.Text = fTrafficSurvey.getLorry()
    End Sub

    Private Sub AddVehicles_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles AddVehicles.Click

        Total.Text = fTrafficSurvey.getCar() + fTrafficSurvey.getBicycle()
+ fTrafficSurvey.getLorry()

    End Sub

    Private Sub ExitToolStripMenuItem_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles ExitToolStripMenuItem.Click
        'The application is closed.
        Me.Close()
    End Sub
End Class
```

Answer to Question 6

BallAdmin.vb

```
Imports MT264Sprites
Public Class BallAdmin
    Private fStep As Integer
    Private fHVSprite As HVSprite
    Private fGameImage As Bitmap
    Private fGameArea As Rectangle
    Private fTimeInterval As Integer
    Private fRunning As Boolean
    Public ReadOnly Property TimeInterval() As Integer
        Get
            Return fTimeInterval
        End Get
    End Property
    Public ReadOnly Property Running() As Boolean
        Get
            Return fRunning
        End Get
    End Property
    Public ReadOnly Property GameImage() As Bitmap
        Get
            Return fGameImage
        End Get
    End Property

    Public Sub New()
        'Preconditions: none
        'Postconditions: A BallAdmin object is created using the
parametrised
        'constructor, but with a game area of width 200 and height
200.
        MyClass.New(200, 200)
    End Sub
    Public Sub New(ByVal width As Integer, ByVal height As Integer)
        'Preconditions: width > 0 and height > 0.
        'Postconditions: A BallAdmin object is created. The game area
is created of the
        'given width and height. GameImage is created with the same
width and height.
        'A ball is placed so that it will appear in the middle of the
game area.
        'TimeInterval is set to 100. The ball is drawn on the game
area and Running
        ' is set to True.
        Dim spritePoint As Point
        fGameArea = New Rectangle(0, 0, width, height)
        fGameImage = New Bitmap(fGameArea.Width, fGameArea.Height)
        spritePoint = New Point(fGameArea.Width \ 2, fGameArea.Height
\ 2)
        fStep = 3
        fHVSprite = New HVSprite(spritePoint, fGameArea.Width \ 20,
fGameArea.Width \ 20, fStep, 0)
        fTimeInterval = 100
        fRunning = True
        updateGameImage()
    End Sub
```

```
    Public Sub setDirection(ByVal direction As Char)
        'Preconditions: none
        'Postconditions: Decreases TimeInterval while > 10 and the
ball's
        'direction of movement is set according to the parameter.
        If TimeInterval > 10 Then
            fTimeInterval = TimeInterval - 2
        End If
        Select Case direction
            Case "u"c : fHVSprite.goUp(fStep)
            Case "d"c : fHVSprite.goDown(fStep)
            Case "l"c : fHVSprite.goLeft(fStep)
            Case "r"c : fHVSprite.goRight(fStep)
        End Select
    End Sub
    Public Sub nextMove()
        'Preconditions: none
        'Postconditions: If Running is True, the ball is moved and
GameImage is
        'updated, and if the ball has gone through the game area's
boundaries, then
        'Running is set to False.
        If Running Then
            fHVSprite.move()
            updateGameImage()
        End If
        If Not fGameArea.Contains(fHVSprite.BoundingBox) Then
            fRunning = False
        End If
    End Sub
    Private Sub updateGameImage()
        'Preconditions: none
        'Postconditions: GameImage is updated to match the current
position of the ball.
        Dim g As Graphics
        g = Graphics.FromImage(GameImage)
        g.Clear(Color.White)
        fHVSprite.draw(g)
        g.Dispose()
    End Sub
    Public Sub draw(ByVal g As Graphics)
        'Preconditions: none
        'Postconditions: The current state of the game is drawn on g.
        g.DrawImage(GameImage, 0, 0)
    End Sub
    Public Sub dispose()
        GameImage.Dispose()
    End Sub
End Class
```

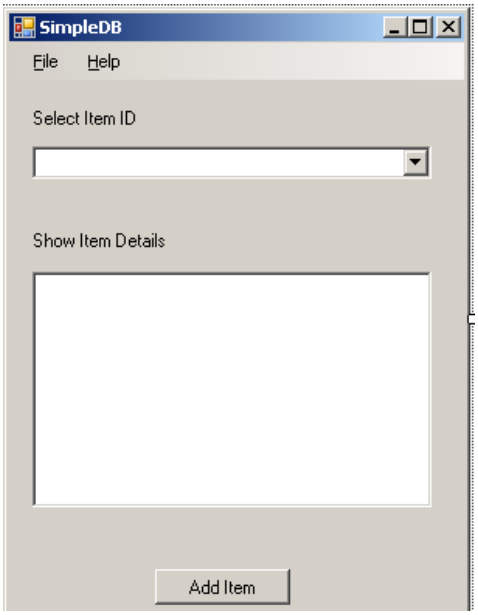
MainForm.vb

```
Public Class MainForm
    Private fBallAdmin As BallAdmin

    Private Sub exitMenuItem_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles exitMenuItem.Click
        Close()
    End Sub

    Public Sub New()

        ' This call is required by the Windows Form Designer.
        InitializeComponent()

        ' Add any initialization after the InitializeComponent()
call.
        fBallAdmin = New BallAdmin(gamePanel.Width, gamePanel.Height)
        updateView()
    End Sub
    Private Sub updateView()
        gameTimer.Enabled = fBallAdmin.Running
        gameTimer.Interval = fBallAdmin.TimeInterval
        gamePanel.Refresh() 'Note that only the panel needs to be
repainted.
    End Sub

    Private Sub gamePanel_Paint(ByVal sender As System.Object, ByVal
e As System.Windows.Forms.PaintEventArgs) Handles gamePanel.Paint
        'The current state of the game is displayed on the panel's
Graphics object.
        'Two alternatives are given for displaying the game.
        Dim g As Graphics
        g = gamePanel.CreateGraphics()
        fBallAdmin.draw(g) 'first alternative
        'g.DrawImage(fBallAdmin.GameImage, 0, 0) 'second alternative
        g.Dispose()
    End Sub

    Private Sub gameTimer_Tick(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles gameTimer.Tick
        'The ball is moved and a check is made for collisions with
the walls.
        fBallAdmin.nextMove()
        updateView()
    End Sub

    Private Sub upButton_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles upButton.Click
        'Direction of travel is set to upwards, increasing the game's
speed.
        fBallAdmin.setDirection("u"c)
        updateView()
    End Sub

    Private Sub leftButton_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles leftButton.Click
        'Direction of travel is set to left, increasing the game's
speed.
        fBallAdmin.setDirection("l"c)
        updateView()
```

```
    End Sub

    Private Sub rightButton_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles rightButton.Click
        'Direction of travel is set to right, increasing the game's
speed.
        fBallAdmin.setDirection("r"c)
        updateView()
    End Sub

    Private Sub downButton_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles downButton.Click
        'Direction of travel is set to down, increasing the game's
speed.
        fBallAdmin.setDirection("d"c)
        updateView()
    End Sub


    Private Sub MainForm_KeyDown(ByVal sender As System.Object, ByVal
e As System.Windows.Forms.KeyEventArgs) Handles MyBase.KeyDown
        'Direction of travel is set according to key pressed,
increasing the game's speed.
        Select Case e.KeyCode
            Case Keys.Up : fBallAdmin.setDirection("u"c)
            Case Keys.Down : fBallAdmin.setDirection("d"c)
            Case Keys.Left : fBallAdmin.setDirection("l"c)
            Case Keys.Right : fBallAdmin.setDirection("r"c)
        End Select
        e.Handled = True
        updateView()
    End Sub

    Private Sub upButton_PreviewKeyDown(ByVal sender As
System.Object, ByVal e As
System.Windows.Forms.PreviewKeyDownEventArgs) Handles
upButton.PreviewKeyDown, rightButton.PreviewKeyDown,
leftButton.PreviewKeyDown, downButton.PreviewKeyDown
        Select Case e.KeyCode
            Case Keys.Down, Keys.Left, Keys.Right, Keys.Up
                e.IsInputKey = True
        End Select
    End Sub

    Private Sub MainForm_Resize(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles MyBase.Resize
        'The game area is changed; if this happens during a game, the
game is reset.
        If fBallAdmin IsNot Nothing Then
            fBallAdmin.dispose()
        End If
        fBallAdmin = New BallAdmin(gamePanel.Width, gamePanel.Height)
        updateView()
    End Sub
End Class
```

Answer to Question 7

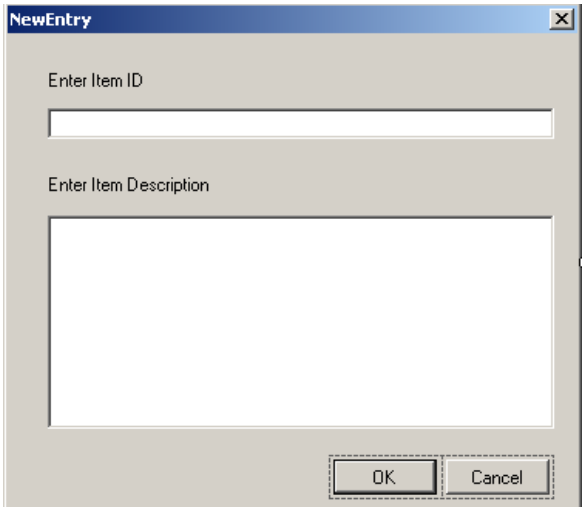
For the main form we can design something like the following :



Here we have :

1) MenuStrip
2) ComboBox
3) TextBox
4) Button

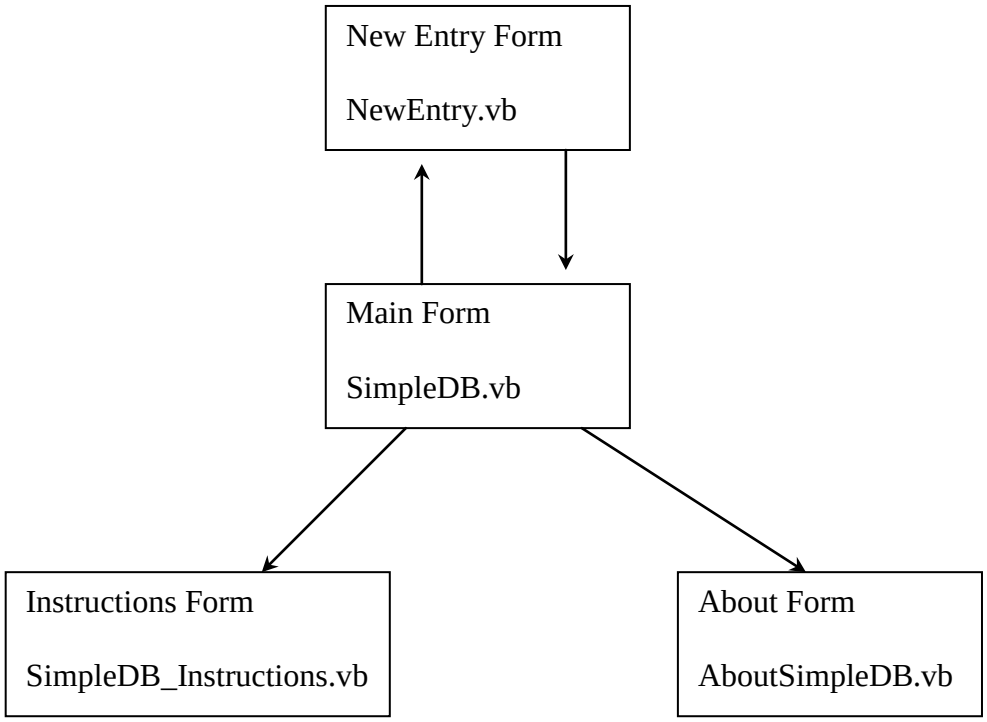
It is better to split the design so that when the user click on the Add Item button they will get another form as follows :



Here we have 2 textboxes, the bottom one is multilane

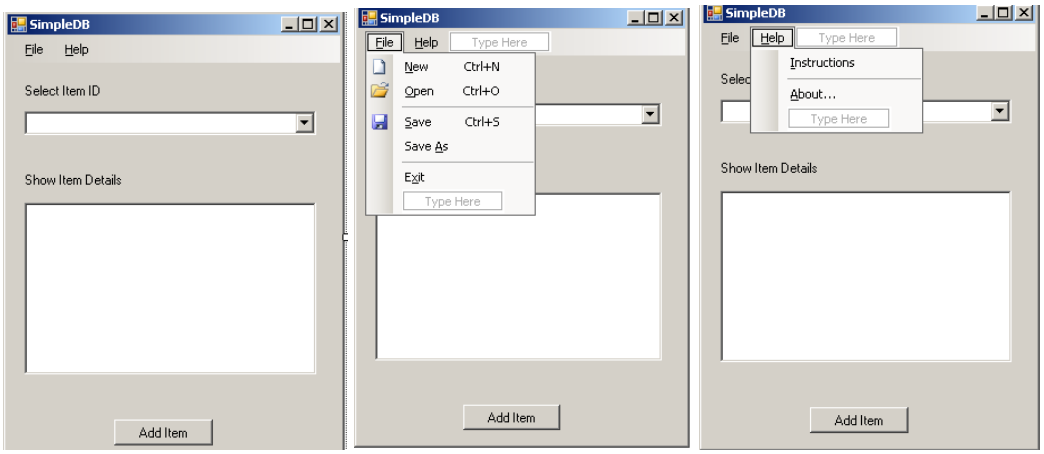
Answer to Question 8

Software design



GUI Design

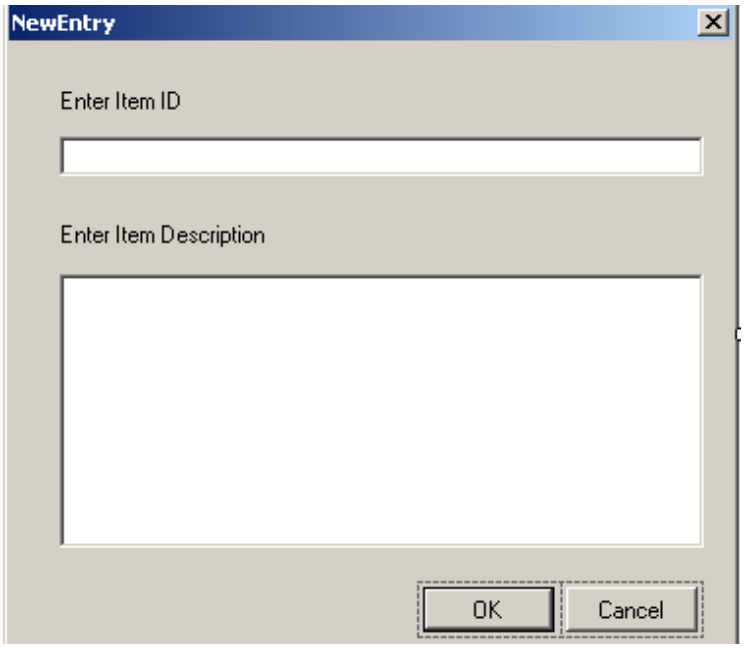
Main Form : SimpleDB.vb



Property Table : Display the entered items

	Name	Property	Initial Value
Container			
Form	SimpleDB	Text	Display the entered items
Controls			
ComboBox	itemComboBox		Scroll through the items entered
TextBox	definitionTextBox	Text	Show the properties of the items entered
MenuStrip	MenuStrip1	File	This is to show the File properties that includes : New, Open, Save, Save As and Exit
MenuStrip	MenuStrip1	Help	This is to show the Help properties that includes : Instructions and About information
Button	AddItemBut	Text	Loading the New Entry Form

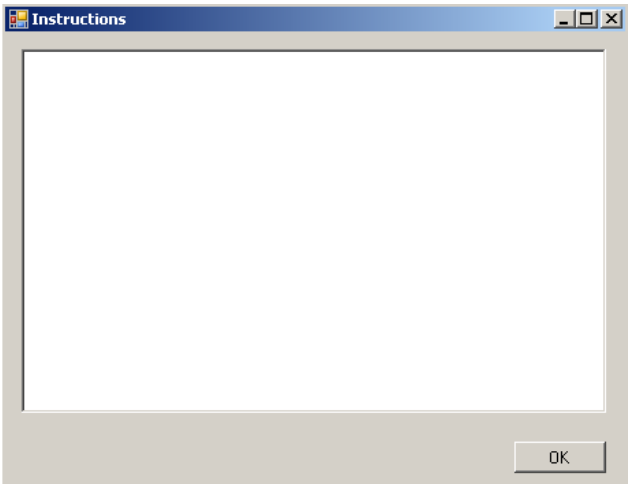
Entry Form : NewEntry.vb



Property Table : Enter new items

	Name	Property	Initial Value
Container			
Form	NewEntry	Text	Enter new items
Controls			
TextBox	itemTextBox	Text	Enter the Item's identifier
TextBox	descriptionTextBox	Text	Enter the Item's description
Button	OK_Button	Text	OK to enter the item's details
Button	Cancel_Button	Text	Cancel

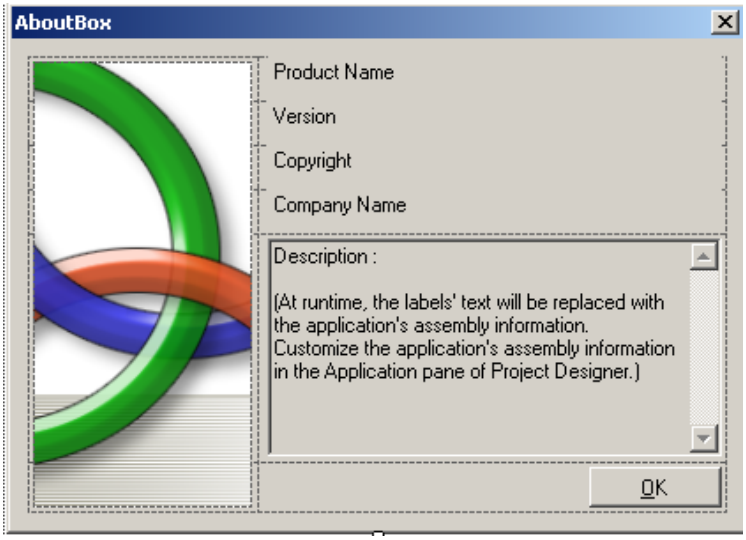
Instructions Form : SimpleDB_Instructions



Property Table : Enter new items

	Name	Property	Initial Value
Container			
Form	SimpleDB_Instructions	Text	Display the instructions
Controls			
TextBox	instructTextBox	Text	Display the instructions
Button	OK_Button	Text	OK to close the form

About DialogBox : AboutSimpleDB



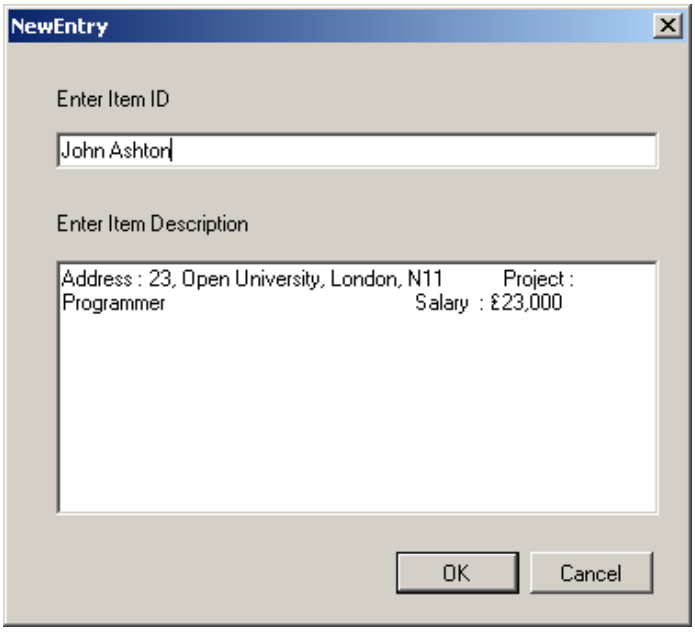
Property Table : About the software information

	Name	Property	Initial Value
Container			
DialogBox	AboutSimpleDB	Text	Show the software's information
Controls			
LogoPictureBox	LogoPictureBox	Image	Display the logo picture
Button	OK_Button	Text	OK to close the form
TextBox	LabelProductName	Text	Display the software name
TextBox	LabelVersion	Text	Display the label version
TextBox	LabelCopyright	Text	Display the copyright
TextBox	LabelCompanyName	Text	Display the company name

SimpleDatabase software testing

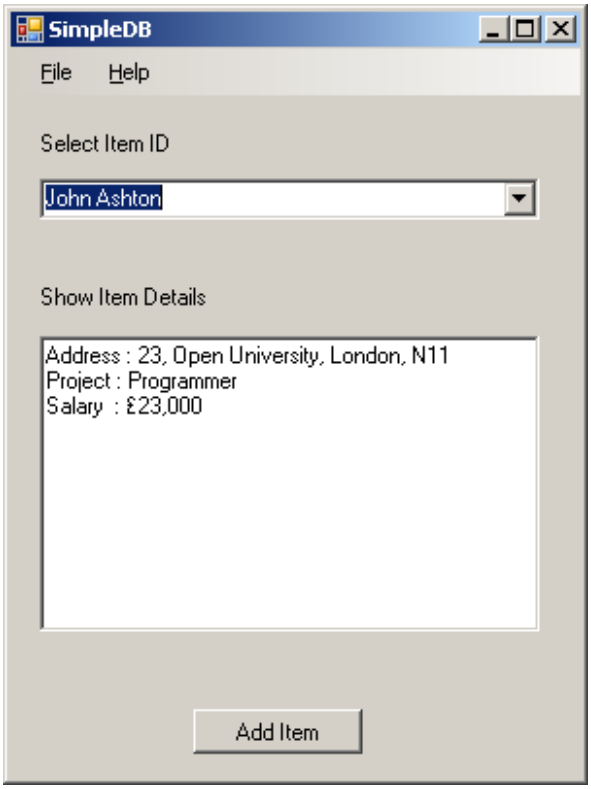
1) Entering new data

Load the software and click on “Add Item” then type the item’s details



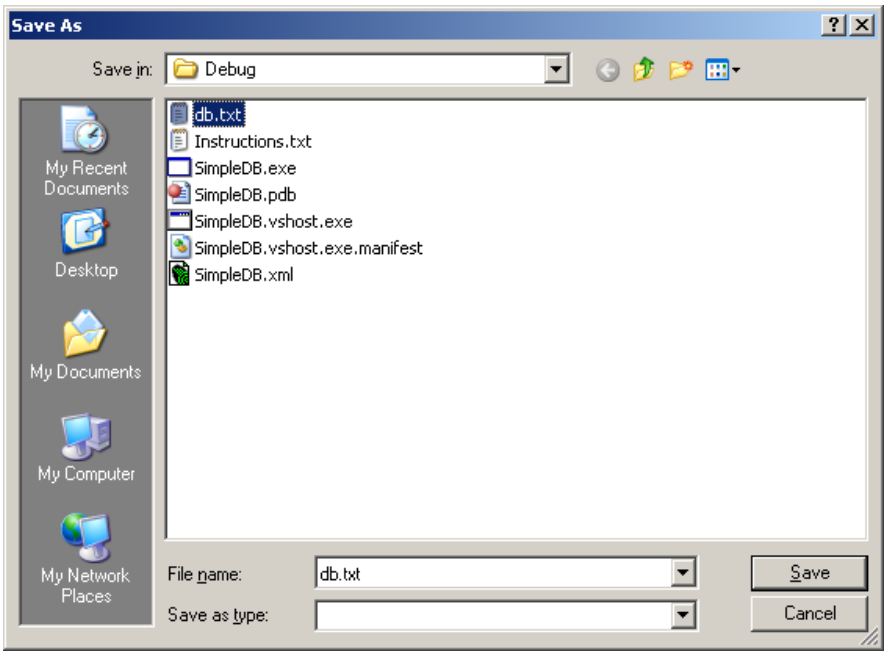
2) Displaying new data

Load the software and click on the combo box to display the item’s details



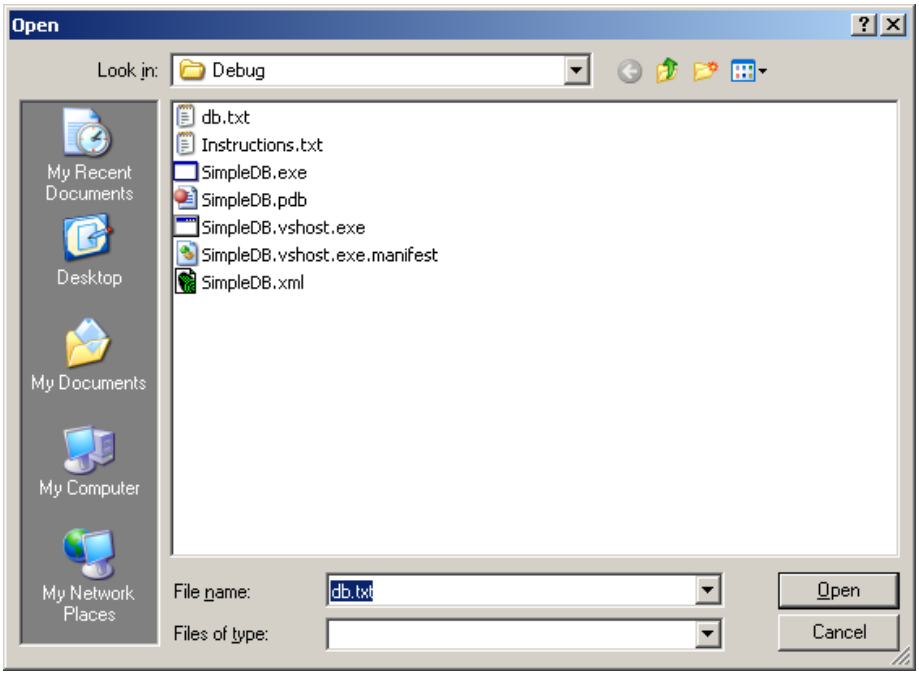
3) Saving new data

Click on File then Save and give a filename to save the data in. In this case the file is called db.txt



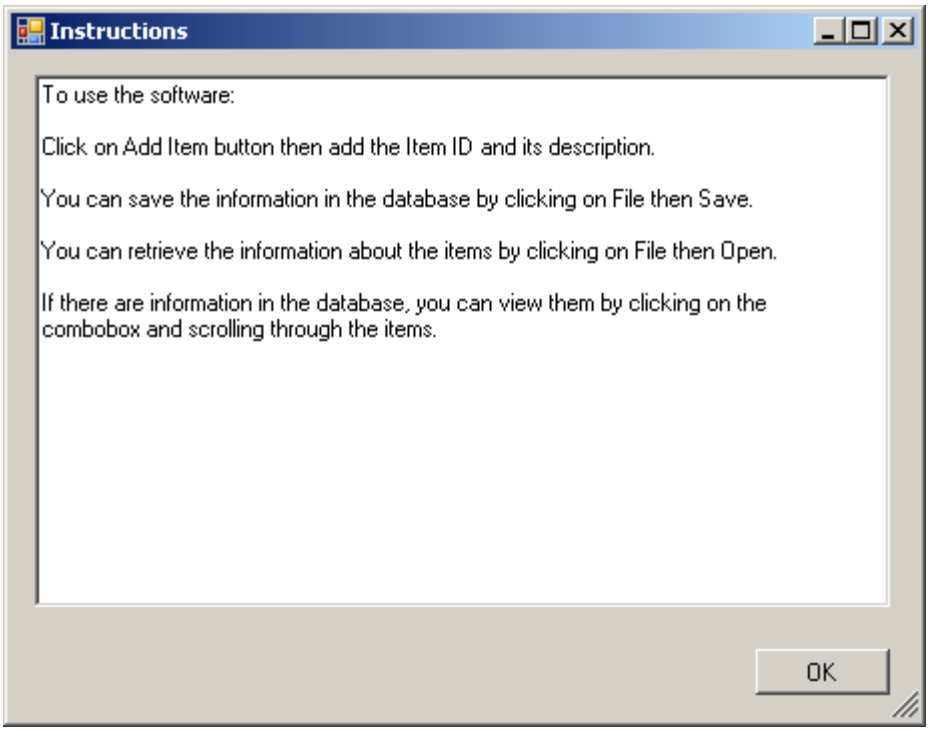
4) Retrieving data

Click on File then Open and give a filename to retrieve the data in. In this case the file is called db.txt



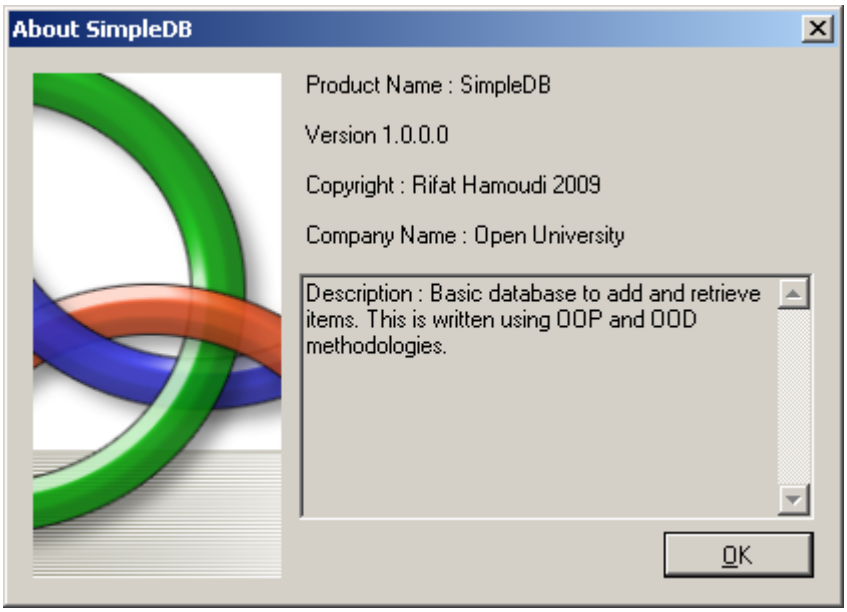
5) Show instructions on how to use the software

Click on Help then Instructions



6) Show About dialog for the SimpleDB software

Click on Help then Instructions



Software implementation

SimpleDatabase class definition and implementation

```
Imports System.IO

Public Class SimpleDatabase
    Private fItem As Dictionary(Of String, String)
    Private fModified As Boolean
    Private fFileName As String

    Public Sub New()
        'Preconditions: none
        'Postconditions: a SimpleDatabase object has been created
with the
        'glossary initially empty, FileName set to "" and Modified
set to False.
        FileName = ""
        fItem = New Dictionary(Of String, String)
        fModified = False
    End Sub

    Public Property FileName() As String
        Get
            Return fFileName
        End Get
        Set(ByVal value As String)
            fFileName = value
        End Set
    End Property

    Public ReadOnly Property Modified() As Boolean
        Get
            Return fModified
        End Get
    End Property

    Public ReadOnly Property WordList() As Dictionary(Of String,
String).KeyCollection
        'Preconditions: none
        'Postconditions: The database item entries are returned as a
collection of keys.
        Get
            Return fItem.Keys
        End Get
    End Property

    Public ReadOnly Property Description(ByVal word As String) As
String
        'Preconditions: item is an entry in the database
        'Postconditions: The description associated with the item
entry
        'for the item is returned.
        Get
            Return fItem.Item(word)
        End Get
    End Property
```

```
    Public Sub add(ByVal word As String, ByVal description As String)
        'Preconditions: item and description are not empty strings
and item is not
        'already an entry in the database.
        'Postconditions: A new entry for item with associated
description is added.
        'Modified is set to True.
        fItem.Add(word, description)
        fModified = True
    End Sub


    Public Sub load()
        'Preconditions: none
        'Postconditions: If FileName is a valid path for a text file
and this file is of
        'appropriate format, then the glossary is loaded from this
file. Otherwise the
        'glossary is cleared and an exception is thrown. It is the
responsibility of
        'client code to handle the exception. Modified is set to
False in either case.
        Dim aReader As StreamReader
        fItem.Clear()
        Try
            aReader = New StreamReader(FileName)
            While Not (aReader.EndOfStream)
                fItem.Add(aReader.ReadLine(), aReader.ReadLine())
            End While
        Catch ex As ArgumentException
            fItem.Clear()
            Throw New ArgumentException("There was a problem opening
the file " + FileName)
        Catch ex As IOException
            fItem.Clear()
            Throw New IOException("There was an input-output error
for " + FileName)
        Catch ex As UnauthorizedAccessException
            fItem.Clear()
            Throw New UnauthorizedAccessException("There was an
unauthorised access error for " + FileName)
        Finally
            fModified = False
            If aReader IsNot Nothing Then
                aReader.Close()
            End If
        End Try
    End Sub
```

```
    Public Sub save()
        'Preconditions: none
        'Postconditions: If FileName is a valid path for a text file,
then the items of the
        'glossary are stored in the file and Modified is set to
False. Otherwise an exception
        'is thrown. It is the responsibility of client code to handle
the exception.
        Dim aWriter As StreamWriter
        Try
            aWriter = New StreamWriter(FileName)
            For Each key As String In WordList
                aWriter.WriteLine(key)
                aWriter.WriteLine(Description(key))
            Next
            fModified = False
        Catch ex As ArgumentException
            fItem.Clear()
            Throw New ArgumentException("There was a problem opening
the file " + FileName)
        Catch ex As IOException
            fItem.Clear()
            Throw New IOException("There was an input-output error
for " + FileName)
        Catch ex As UnauthorizedAccessException
            fItem.Clear()
            Throw New UnauthorizedAccessException("There was an
unauthorised access error for " + FileName)
        Finally
            If aWriter IsNot Nothing Then
                aWriter.Close()
            End If
        End Try
    End Sub
End Class
```

Main Form : SimpleDB.vb

```
Imports System.IO

Public Class SimpleDB

    Private fSimpleDatabase As SimpleDatabase
    Public Sub New()
        fSimpleDatabase = New SimpleDatabase
        ' This call is required by the Windows Form Designer.
        InitializeComponent()

        ' Add any initialization after the InitializeComponent()
call.
        updateView()
    End Sub

    Public Sub updateView()
        itemComboBox.Items.Clear()
        definitionTextBox.Clear()
        For Each key As String In fSimpleDatabase.WordList
            itemComboBox.Items.Add(key)
        Next
    End Sub

    Private Sub AboutToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
AboutToolStripMenuItem.Click
        AboutSimpleDB.ShowDialog()
    End Sub

    Private Sub itemComboBox_SelectedIndexChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
itemComboBox.SelectedIndexChanged
        'The description of the selected word is displayed.
        definitionTextBox.Text = _

fSimpleDatabase.Description(itemComboBox.SelectedItem.ToString())
    End Sub

    Private Sub ContentsToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
ContentsToolStripMenuItem.Click
        ' This event handler displays a dialog box with the
instructions
        ' for playing the game.
        Dim anInstructDialog As SimpleDB_Instructions
        anInstructDialog = New SimpleDB_Instructions
        anInstructDialog.ShowDialog()
        anInstructDialog.Dispose()
    End Sub
```

```
    Private Sub saveMenuItem_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles saveMenuItem.Click
        'If the file name is empty, the event handler for
saveAsMenuItem is used.
        'Otherwise the glossary is saved and the view is updated. If
an exception is
        'caught, the file name is set to "" and an error message is
displayed.
        'Add your code here
        If fSimpleDatabase.FileName = "" Then
            saveAsMenuItem_Click(sender, e)
        Else
            Try
                fSimpleDatabase.save()
                updateView()
            Catch ex As ArgumentException
                fSimpleDatabase.FileName = ""
                MessageBox.Show(ex.Message)
            Catch ex As IOException
                fSimpleDatabase.FileName = ""
                MessageBox.Show(ex.Message)
            Catch ex As UnauthorizedAccessException
                fSimpleDatabase.FileName = ""
                MessageBox.Show(ex.Message)
            Catch ex As Exception
                fSimpleDatabase.FileName = ""
                MessageBox.Show(ex.Message)
            End Try
        End If
    End Sub

    Private Sub saveAsMenuItem_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles saveAsMenuItem.Click
        If (aSaveFileDialog.ShowDialog() = DialogResult.OK) Then
            fSimpleDatabase.FileName = aSaveFileDialog.FileName
            saveMenuItem_Click(sender, e) 'Reuse code to save the
file
        End If
    End Sub

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

```
    Private Sub openMenuItem_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles openMenuItem.Click
        If fSimpleDatabase.Modified Then
            If MessageBox.Show("Database has been modified, save?",
"SimpleDatabase", _
                  MessageBoxButtons.YesNo) = DialogResult.Yes Then
                saveMenuItem_Click(sender, e)  'Reuse code to save
the file
            End If
        End If

        If (anOpenFileDialog.ShowDialog() = DialogResult.OK) Then
            Try
                fSimpleDatabase.FileName = anOpenFileDialog.FileName
                fSimpleDatabase.load()
            Catch ex As ArgumentException
                fSimpleDatabase.FileName = ""
                MessageBox.Show(ex.Message)
            Catch ex As IOException
                fSimpleDatabase.FileName = ""
                MessageBox.Show(ex.Message)
            Catch ex As UnauthorizedAccessException
                fSimpleDatabase.FileName = ""
                MessageBox.Show(ex.Message)
            Catch ex As Exception
                fSimpleDatabase.FileName = ""
                MessageBox.Show(ex.Message)
            End Try
        End If
        updateView()
    End Sub

    Private Sub AddItemBut_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles AddItemBut.Click
        'A dialogue box for a new entry is displayed. If an
appropriate entry is
        'made and OK is clicked, the entry is added to the glossary.
If the entry
        'already exists, a warning is displayed instead. The GUI is
updated.
        Dim aNewEntryDialog As NewEntry
        Dim word As String
        Dim description As String
        aNewEntryDialog = New NewEntry
        If (aNewEntryDialog.ShowDialog() = DialogResult.OK) Then
            word = aNewEntryDialog.itemTextBox.Text
            description = aNewEntryDialog.descriptionTextBox.Text
            If fSimpleDatabase.WordList.Contains(word) Then
                MessageBox.Show("Cannot add new entry - item already
contained in database")
            Else
                fSimpleDatabase.add(word, description)
            End If
        End If
        aNewEntryDialog.Dispose()
        updateView()
    End Sub

End Class
```

Entry Form : NewEntry.vb

```
Imports System.Windows.Forms

Public Class NewEntry

    Private Sub OK_Button_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles OK_Button.Click
        Dim aItem As String
        Dim description As String
        aItem = Me.itemTextBox.Text
        description = Me.descriptionTextBox.Text
        If aItem = "" Or description = "" Then
            Me.DialogResult = Windows.Forms.DialogResult.None
        Else
            Me.DialogResult = System.Windows.Forms.DialogResult.OK
            Me.Close()
        End If
    End Sub

    Private Sub Cancel_Button_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Cancel_Button.Click
        Me.DialogResult = System.Windows.Forms.DialogResult.Cancel
        Me.Close()
    End Sub

End Class
```

Instructions Form : SimpleDB_Instructions

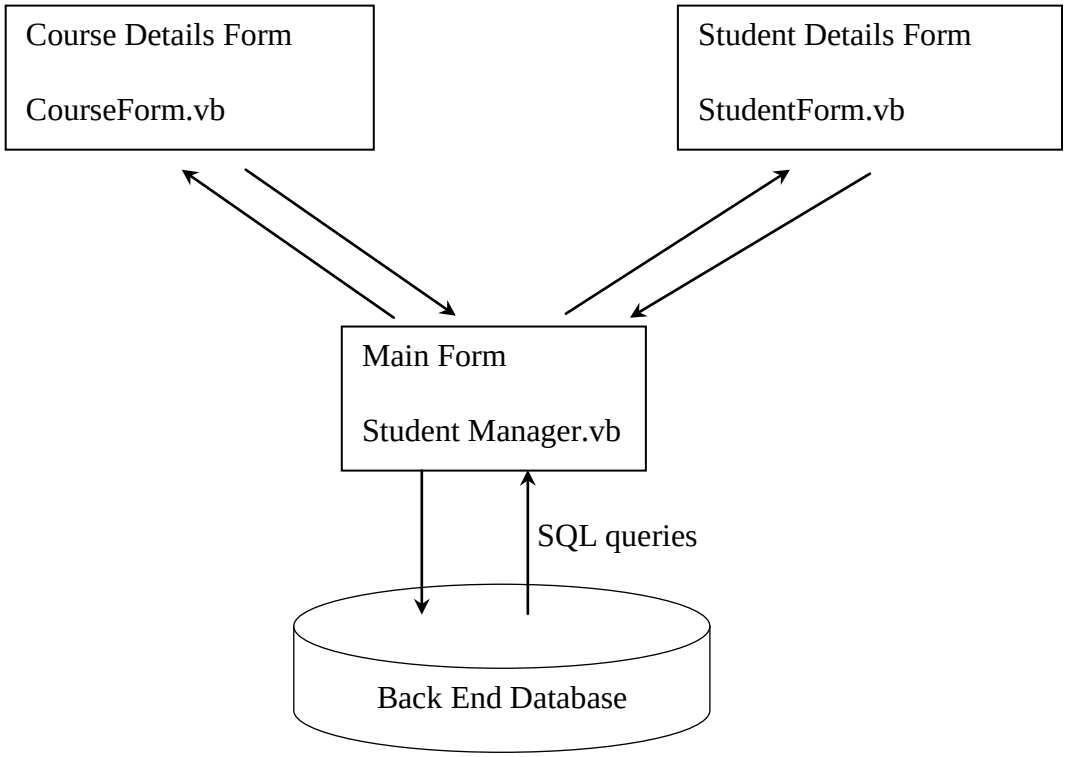
```
Public Class SimpleDB_Instructions

    Private Sub SimpleDB_Instructions_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        ' The file to be loaded is plain text, so we have to specify
the type.
        instructTextBox.LoadFile("Instructions.txt",
RichTextBoxStreamType.PlainText)
    End Sub

    Private Sub OK_Button_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles OK_Button.Click
        Me.DialogResult = System.Windows.Forms.DialogResult.OK
        Me.Close()
    End Sub
End Class
```

About DialogBox : AboutSimpleDB

```
Public NotInheritable Class AboutSimpleDB

    Private Sub AboutBox_Load(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles MyBase.Load
        ' Set the title of the form.
        Dim ApplicationTitle As String
        If My.Application.Info.Title <> "" Then
            ApplicationTitle = My.Application.Info.Title
        Else
            ApplicationTitle =
System.IO.Path.GetFileNameWithoutExtension(My.Application.Info.Assemb
lyName)
        End If
        Me.Text = String.Format("About {0}", ApplicationTitle)
        ' Initialize all of the text displayed on the About Box.
        ' TODO: Customize the application's assembly information in
the "Application" pane of the project
        '    properties dialog (under the "Project" menu).
        Me.LabelProductName.Text = "Product Name : SimpleDB"
        Me.LabelVersion.Text = String.Format("Version {0}",
My.Application.Info.Version.ToString)
        Me.LabelCopyright.Text = "Copyright : Rifat Hamoudi 2009"
        Me.LabelCompanyName.Text = "Company Name : Open University"
        Me.TextBoxDescription.Text = "Description : Basic database to
add and retrieve items. This is written using OOP and OOD
methodologies."
    End Sub

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

End Class
```

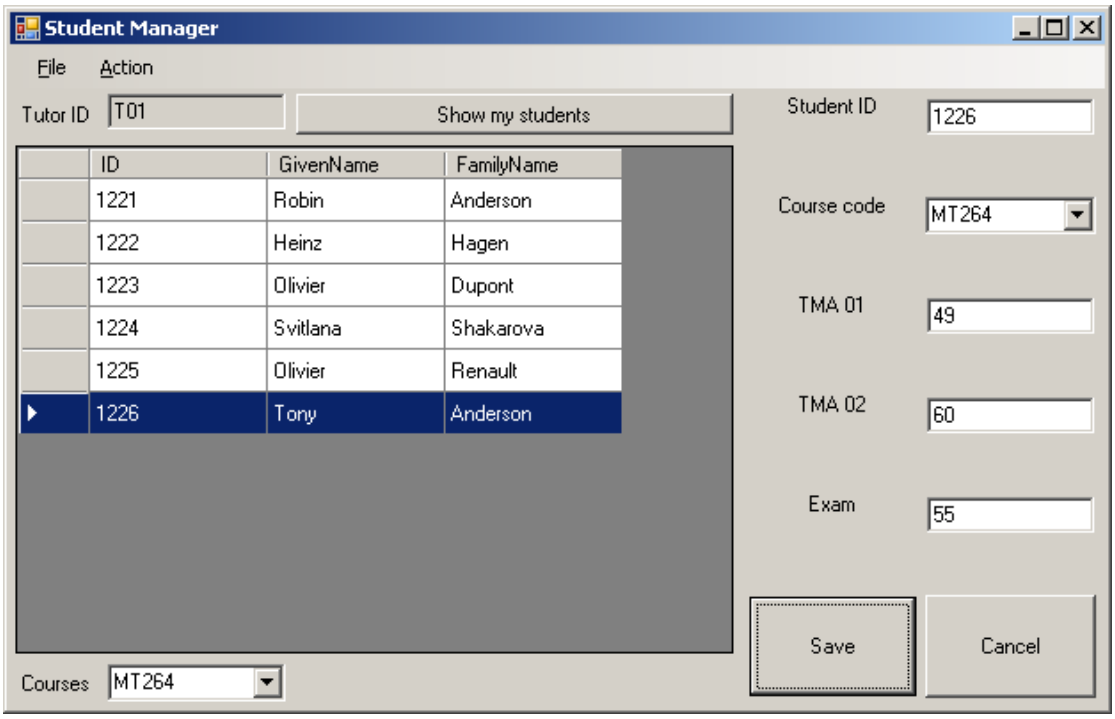
Answer to Question 9

Software design

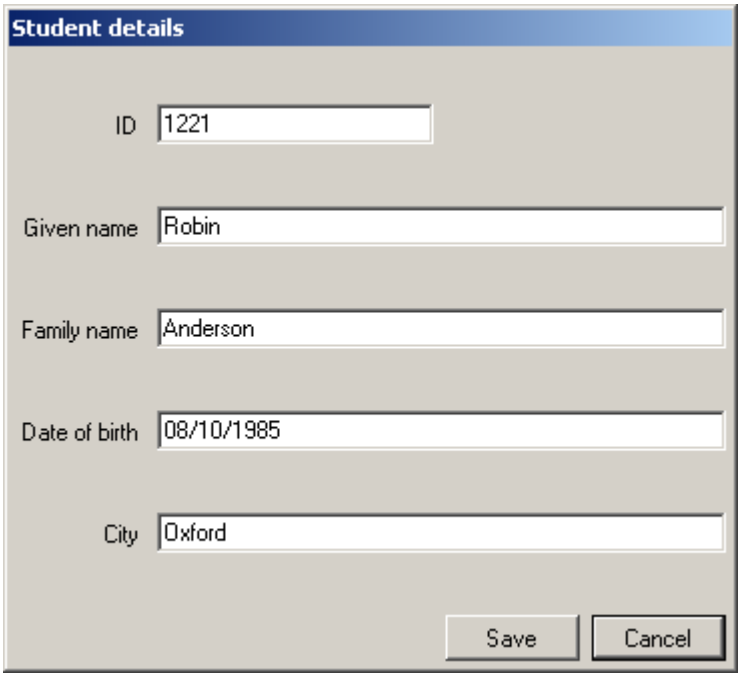


GUI Design

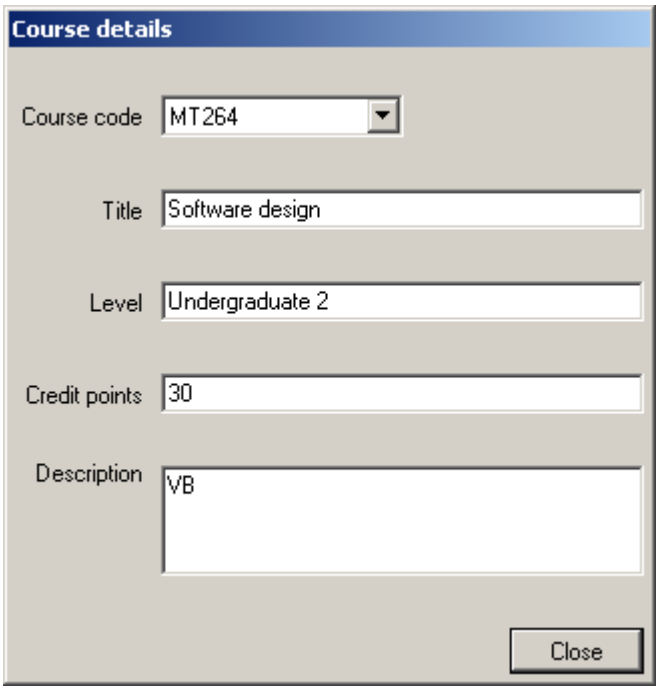
Main Form : StudentManager.vb



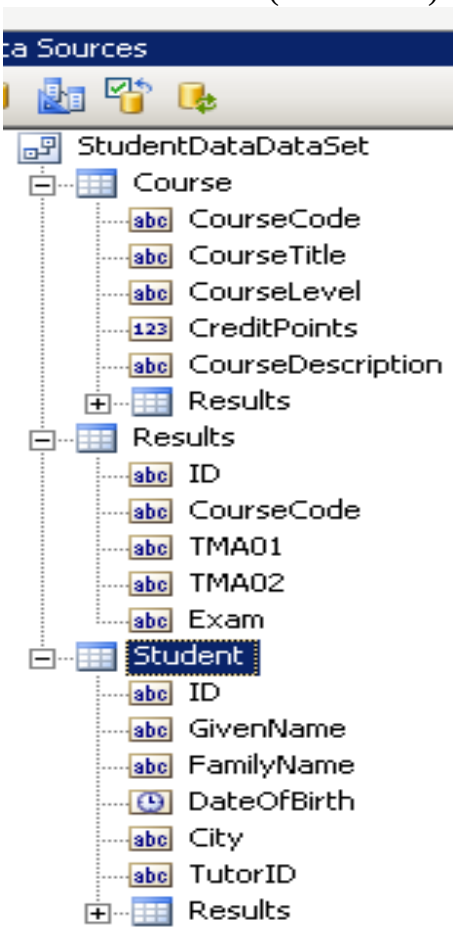
Student Details Form : StudentForm.vb



Course Details Form : CourseForm.vb



Relational Database (DataSource)



MainForm.vb

```
Public Class MainForm

    Private Sub MainForm_Load(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles MyBase.Load
        Try
            'Fill the CourseTable.

CourseForm.CourseTableAdapter.Fill(CourseForm.StudentDataDataSet.Cour
se)

        Catch ex As SqlClient.SqlException
            MessageBox.Show(ex.ToString)
        End Try
    End Sub

    Private Sub courseMenuItem_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles courseMenuItem.Click
        'The Course form is displayed.
        'CourseForm.ShowDialog()
        CourseForm.Show()
    End Sub

    Public Sub New()

        ' This call is required by the Windows Form Designer.
        InitializeComponent()

        ' Add any initialization after the InitializeComponent()
call.
        allCoursesComboBox.DataSource =
CourseForm.CourseBindingSource
        allCoursesComboBox.DisplayMember = "CourseCode"
    End Sub

    Private Sub showDataButton_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles showDataButton.Click
        'Load student data for the tutor user. Load results data.
        'Make tutor ID text box read-only.
        Try

Me.StudentTableAdapter.FillByTutorID(Me.StudentDataDataSet.Student,
tutorIdTextBox.Text)

Me.ResultsTableAdapter.FillByTutorID(Me.StudentDataDataSet.Results,
tutorIdTextBox.Text)
            tutorIdTextBox.ReadOnly = True
        Catch ex As System.Exception
            System.Windows.Forms.MessageBox.Show(ex.Message)
        End Try
    End Sub

    Private Sub studentMenuItem_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles studentMenuItem.Click
        'The Student form is displayed. If the user edits the data
and clicks on Save,
        'then the Student data is updated; otherwise any new edits
are discarded.
        If StudentForm.ShowDialog = Windows.Forms.DialogResult.OK
Then
```

```
            StudentBindingSource.EndEdit()
        End If
        StudentForm.Show()
    End Sub

    Private Sub saveButton_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles saveButton.Click
        'The local data set is updated with the amended results
record.
        FKResultsStudentBindingSource.EndEdit()
    End Sub

    Private Sub saveMenuItem_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles saveMenuItem.Click
        'Update the local Results and Student data. Then update the
database.
        Try
            StudentBindingSource.EndEdit()
            FKResultsStudentBindingSource.EndEdit()
            StudentTableAdapter.Update(StudentDataDataSet.Student)
            ResultsTableAdapter.Update(StudentDataDataSet.Results)
        Catch ex As SqlClient.SqlException
            MessageBox.Show(ex.ToString())
        End Try
    End Sub

    Private Sub MainForm_FormClosing(ByVal sender As System.Object,
ByVal e As System.Windows.Forms.FormClosingEventArgs) Handles
MyBase.FormClosing
        'All data is saved to the database.
        saveMenuItem_Click(sender, e)
    End Sub

    Private Sub clearMenuItem_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles clearMenuItem.Click
        'Save current data. Clear data from the Student and Results
tables.
        saveMenuItem_Click(sender, e)
        StudentDataDataSet.Student.Clear()
        StudentDataDataSet.Results.Clear()
        tutorIdTextBox.ReadOnly = False
    End Sub

    Private Sub addCourseMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
addCourseMenuItem.Click
        'Add selected student-course to Results table in local data
set.
        Dim row As DataRowView
        Try
            FKResultsStudentBindingSource.EndEdit()
            row = CType(FKResultsStudentBindingSource.AddNew(),
DataRowView)
            row.Item("CourseCode") = allCoursesComboBox.Text
            FKResultsStudentBindingSource.EndEdit()
        Catch ex As Data.ConstraintException
            FKResultsStudentBindingSource.CancelEdit()
            MessageBox.Show("Course was not added - is it already on
the student's list?")
        End Try
    End Sub
```

```
    Private Sub removeCourseMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
removeCourseMenuItem.Click
        'Remove currently selected student-course from Results table
in local data set.
        FKResultsStudentBindingSource.RemoveCurrent()
        FKResultsStudentBindingSource.EndEdit()
    End Sub

    Private Sub exitMenuItem_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles exitMenuItem.Click
        'The main form is closed
        Close()
    End Sub

    Private Sub cancel_Button_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles cancel_Button.Click
        'All edits made in the results panel since the data was last
saved are cancelled.
        FKResultsStudentBindingSource.CancelEdit()
    End Sub

End Class
```

CourseForm.vb

```
Public Class CourseForm

    Private Sub OK_Button_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles closeButton.Click
        Me.DialogResult = System.Windows.Forms.DialogResult.OK
        Me.Hide()
    End Sub

    Private Sub Cancel_Button_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs)
        Me.DialogResult = System.Windows.Forms.DialogResult.Cancel
        Me.Close()
    End Sub

End Class
```

StudentForm.vb

```
Public Class StudentForm
    Private iDBinding As Binding
    Private givenNameBinding As Binding
    Private familyNameBinding As Binding
    Private dobBinding As Binding
    Private cityBinding As Binding

    Private Sub saveButton_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles saveButton.Click
        MainForm.StudentBindingSource.EndEdit()
        Me.DialogResult = System.Windows.Forms.DialogResult.OK
        Me.Hide()
    End Sub

    Private Sub cancelButton_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles cancelButton1.Click,
cancelButton1.Click
        MainForm.StudentBindingSource.CancelEdit()
        Me.DialogResult = System.Windows.Forms.DialogResult.Cancel
        Me.Hide()
    End Sub

    Private Sub setBindings()
        iDBinding = New Binding("Text",
MainForm.StudentBindingSource, "ID")
        givenNameBinding = New Binding("Text",
MainForm.StudentBindingSource, "GivenName")
        familyNameBinding = New Binding("Text",
MainForm.StudentBindingSource, "FamilyName")
        dobBinding = New Binding("Text",
MainForm.StudentBindingSource, "DateOfBirth", True,
System.Windows.Forms.DataSourceUpdateMode.OnValidation, Nothing, "d")
        cityBinding = New Binding("Text",
MainForm.StudentBindingSource, "City")
        idTextBox.DataBindings.Add(iDBinding)
        givenNameTextBox.DataBindings.Add(givenNameBinding)
        familyNameTextBox.DataBindings.Add(familyNameBinding)
        dobTextBox.DataBindings.Add(dobBinding)
        cityTextBox.DataBindings.Add(cityBinding)
    End Sub

    Public Sub New()

        ' This call is required by the Windows Form Designer.
        InitializeComponent()

        ' Add any initialization after the InitializeComponent()
call.
        setBindings()
    End Sub

End Class
```