Lab 3 Due next lab period – week of May 2-5

Purpose

1. Create a simple Windows GUI contact manager application using Visual Studio and VB.NET
2. Understand the `DataGridView` control
3. Understand VB.NET Collection objects: `List` and `Dictionary`
4. Practise VB.NET programming with IO streams

What to submit

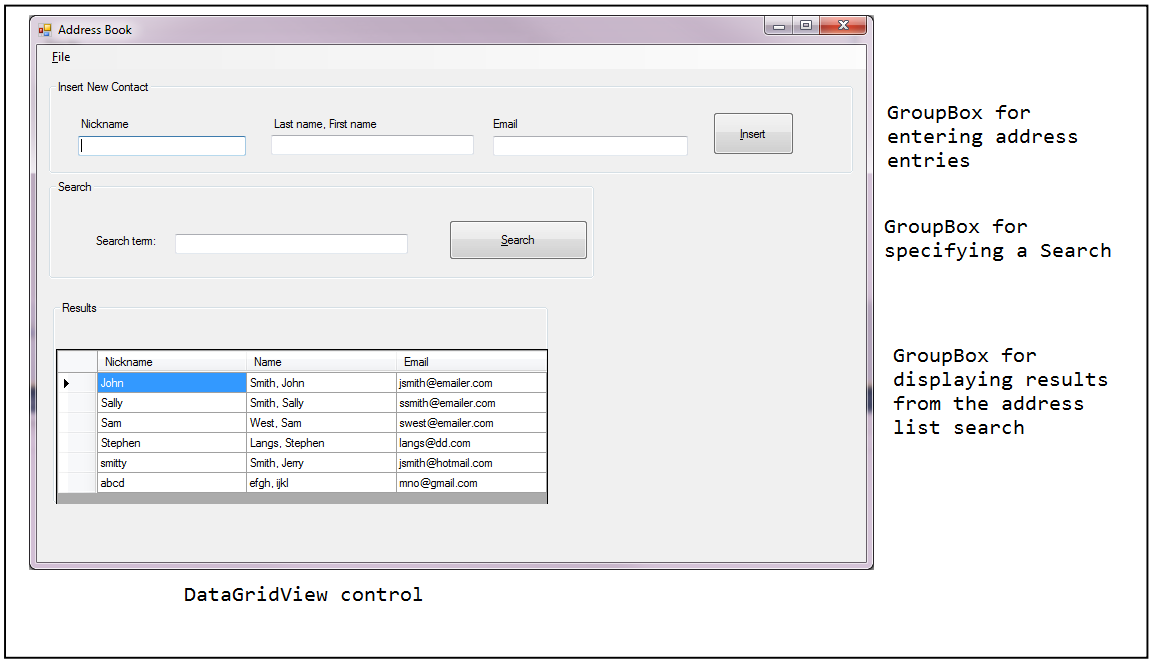
1. Demo will be done in lab, in your lab section.
2. Upload your project compressed into a .zip file to the Dropbox on D2L.

Evaluation (21 marks):

1. Be prepared to demonstrate your solution at the start of next lab. If you are not ready at the start of lab or are not present, a mark of 0 will be given.
2. Your solution will be assessed based on (3 marks each):
    - proper form layout of the controls (no overlapping controls, sufficient white space)
    - coding the insert button to insert new address entries into the address file
    - coding the search button to correctly search for address entries
    - error detection in the code (e.g. blank values detected, no blank searches)
    - menu options work correctly
    - VB.NET code style – variables named appropriately, comments used in the code where needed, code is readable and understandable.
    - Correctly answering questions about your solution.
3. **All work must be individual.** Any plagiarism will result in a mark of 0 and possibly additional penalties.

Description

In this lab you will build a prototype Windows GUI application for managing contacts similar to the following:



The example Windows executable demonstrates what functionality the completed project can be found on D2L in Lab3.zip.

The form is separated into three GroupBox controls: the top GroupBox contains three Labels and three Textboxes and an Insert button to enter new contact information, the middle GroupBox contains a Label and a TextBox used to specify a search within the address list. The bottom GroupBox shows the results of the search, if any, in a DataGridView control. Initially, the application will show all the addresses in a *bound* DataGridView control. The MenuStrip control at the top left has only 3 menu options: Clear Search, Save, and Exit.

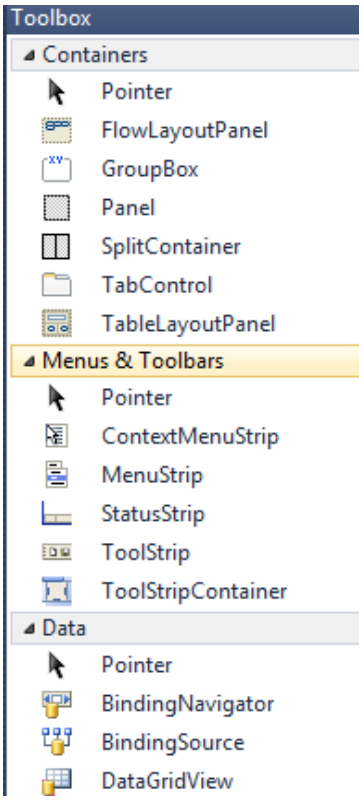


Figure 1 GroupBox, MenuStrip, DataGridView controls

The address book application uses a flat text file named `address.txt` to store personal contact information such as a friend or colleague's nickname, full name and email address. If the address file does not exist, then the application has no records to read and will show nothing in the results datagridview control. The location of the text file will be by default in the Visual Studio's project \bin\Debug folder.

The insert GroupBox contains three Label controls above three TextBox controls and a Button to perform the insert process. The application does not check for a valid email format and it should not accept blanks for any of the three input boxes. The first name and last name textbox does not check for a missing first or last name. The personal address is "keyed" by the nickname which must be unique. If an existing nickname is entered with new name and/or new email, then the application will overwrite the existing values with new values. Duplicate values for name and email entries are allowed. Any spaces before or after the text values should be trimmed. There is no functionality for deleting entries.

The search GroupBox contains a single Label control above a TextBox control and a Button to perform a search. A search for a blank term is not permitted. A single asterisk in the search TextBox returns all address entries. The search is case-sensitive. The search will look for the entered search term in all of the nickname, name and email values and display any matches in the DataGridView control.

The results GroupBox contains the DataGridView control showing the results of the search.

When the form is manually closed by clicking on the form's X in the top right, all the address list entries are written to the text file, `address.txt`, which is stored in the same folder as the application executable file. The address entries appear one per line separated by commas, example:

```
Joe,  Smith, Joseph, jsmith@gmail.com
Sue, West, Susan, suewest@gmail.com
```

"`Joe`" is the nickname, "`Smith, Joseph`" is the name, "`jsmith@gmail.com`" is the email address for example.

The implementation of the VB Form application should use a `Dictionary` Collection object to store the address entries. The keys of the `Dictionary` will be type `String` (nickname), and the values will be a two element `List` of `String` (the name and the email address).

You can assume the contents of the `address.txt` file is valid (e.g. no extra commas on a line entry, or missing data elements). No XML!

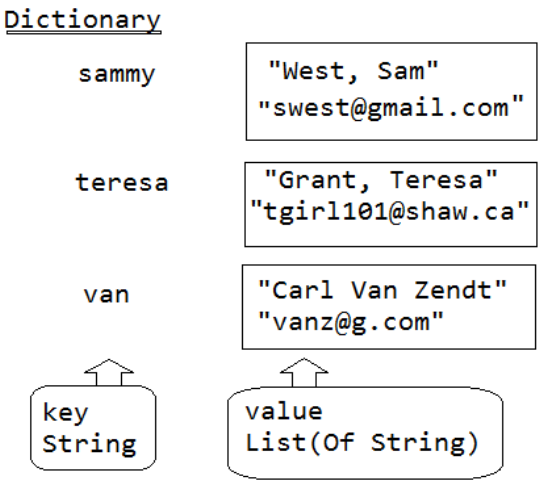


Figure 2 Example of a Dictionary collection with three elements.

Steps

REVIEW THIS SECTION COMPLETELY BEFORE STARTING.

1. Design the form layout with all the required controls in place and appropriately named:
    a. Set the Form's `MinimumSize` to 700 x 500 pixels
    b. Set the `DataGridView` `Anchor` property to Top, Bottom, Left, Right. Set its `AllowUserToAddRows` and `AllowUserToDeleteRows` both to `false`. Set the `AutoSizeColumnsMode` to `Fill` and `AutoSizeRowsMode` to `AllCells`. You can find the `DataGridView` control in the Toolbox panel under the Data section.
    c. Add Option Strict Off at the top of the Form1.vb code.

2. Create the Address Class:
    a. From the Menu, select Project and select the option for Add Class... Enter the name `Address.vb` in the Add New Item dialog. The class `Address` has three `Private` members: strNickname, strName and strEmail. Define the three public `Property` methods for each member, as in:

```
Public Property Nickname() As String
  Get
      Return strNickname
  End Get
   Set (ByVal value As String
           strNickname = value
   End Set
End Property
```

And define the constructor for this class having the three parameters:

```
Sub New(ByVal pstrNickname As String, ...
   MyBase.New()
   strNickname = pstrNickname
  ...
End Sub
```

With this new class you are able to declare and instantiate new Address objects as

```
Dim address As Address

address = New Address("joe", "Joe Smith", "jsmith@gmail.com")
…
address.Nickname  ' returns the nickname of the address object
```

3. Connect the datagrid to the data:
   a. The address entries will be defined in the Form1.vb code in a Dictionary Collection object. You'll need to declare this object as a class variable.

```
Private addressDList As Dictionary(Of String, List(Of String))
```

   b. The first parameter in the Dictionary's Add method is a key (the nickname) and the second parameter is a List Collection object of two strings: the name and the email. When the form loads, you'll need to instantiate this object:

```
addressDList = New Dictionary(Of String, List(Of String))
```

   c. And later, to access and add to this dictionary, you can do:

```
strKey = ... ' The nickname.

lstValue.Add( strName ) ' The full name.
lstValue.Add( strEmail ) ' The email.

addressDList.Add(strKey, lstValue)  ' adding a new address
                                    ' entry to Dictionary List
```

   d. The DataGridView control in the Form1 has its DataSource property set (in code, see below) not to the data file, and not to the Dictionary List but to a List object. The DataSource property must be set to an object that implements the IList interface, which the List Collection does. The List object you need to define will be a List of Address objects. This is not a ListBox control but a Collection object used only in the VB.NET code. You'll need to declare this as a class variable:

```
Private lstAddress As List(Of Address)
```

   e. And in the form load method, instantiate it:

```
lstAddress = New List(Of Address)
```

   f. Then later, you can access and add to it as follows

```
...
... some process to add in all address entries from file...
```

```
lstAddress.Add( address )
...
```

g. Finally, once the list of addresses is set up, you can set the DataGridView's datasource to it:
`DataGridView1.DataSource = lstAddress`

4. A function called `FindAddress` in the Form1.vb code returns the `Address` object matching a given nickname value

```
Private Function FindAddress(ByVal strNickname As String) _
                                                  As Address
  For Each a As Address In lstAddress
      If a.Nickname = strNickname Then
          Return a
      End If
  Next
  Return Nothing
End Function
```

This function will be called within the Form1.vb's `Insert` process to determine if the nickname is already used in the address list.

5. The Form's `Load` Event should instantiate the Dictionary List and the address List

```
addressDList = New Dictionary(Of String, List(Of String))
lstAddress = New List(Of Address)
```

a. Declare a class variable to hold the file name (address.txt). If the data file exists, then attempt to open it within a `Try Catch` block. Use `FileSystem.FileExists(strFilename)` which will require `Imports Microsoft.VisualBasic.FileIO` at the very top of the source code file.

b. Declare a class variable to hold a StreamReader object, called sr. Use a call to the StreamReader object to read in the address entries one line at a time (when there is no more input to read from the file then the Peek method returns -1).

```
sr = New StreamReader(strFileName)
While sr.Peek <> -1
    Dim strLine As String
    strLine = sr.ReadLine
     Dim strKey As String
```

c. The `StreamReader` and `StreamWriter` objects require `Imports System.IO`, and can be declared as class variables

d. Each line in the data file contains four data elements separated by commas: a nickname, last name, first name and email address. You can use any method you like to parse them out. The String `Split` method can come in handy here. For example, assign the `strLine.Split(",")` to `strArray`, where `strArray` is a `String` array. A `String` array can be declared in Visual Basic like this:

```
Dim strArray() As String
```

The first element in a VB array is index zero, e.g. `strArray(0)`.

e. Use the ContainsKey method for Dictionary to check if the key value (the nickname) is already in the Dictionary addressDList. If it is not, then add in the new entry to the Dictionary.
f. Add the address entry to the address List (lstAddress). This will require the creation of a new Address object that will be added to the List.
g. Don't forget to close the StreamReader object sr after the while loop but within the Try block.
h. Set the DataGridView control DataSource to the address List (lstAddress), after the if statement that checked if the file existed
i. Save your work (press ctrl-shift-S or select File | Save All.
j. Run your application (press F5 or select Debug | Start Debugging or click the green arrow on the toolbar). Verify the form loads the sample address.txt data into the DataGridView when you start the application. If it doesn't, confirm the location of the address.txt file is where the .exe file is located (under the folder bin\Debug).

6. The Search will be triggered by clicking the Search button. Its process should follow this sequence:
   a. Declare a strSearch String variable and assign it to the Search textbox Text, apply the Trim method
      ```
      strSearch = txtSearch.Text.Trim
      ```
   b. If the strSearch is blank, show a message telling the user that the search text is blank
   c. Else, set the List lstAddress to Nothing (because you are going to rebuild it with entries that match the provided search string)
   d. Declare lstAddress equal to New List(Of Address). This List object will contain all the address entries from the Dictionary List that successfully match the search.
   e. Declare address as an Address object
   f. Start a For Each loop to iterate through all address entries in the address Dictionary List:
      ```
      For Each kvp As KeyValuePair(Of String, List(Of String)) In addressDList
      ```
      - Declare strNickname as String
      - Declare lstValues as List(Of String) ' this is the two elements of name and email
        Set lstAddress to New List(Of Address)
        Assign kvp.Key to strNickname. This is the key part of the Dictionary List element
      - Assign kvp.Value to lstValues. This is the value part of the Dictionary List element – this is not a String but a List of two elements: the full name and the email
      - Check if the strSearch is found in either strNickname or either of the list elements in lstValues (you can use the IndexOf method and see if it does not return -1) or is equal to "*"
      - If the strSearch is found, then create a new Address object using the values of strNickname and the two list elements. The first element of the List lstValues is lstValues.Item(0).
      - Add the new Address object to the lstAddress List
   g. Set the DataGridView control's DataSource to Nothing (this resets the datagridview)
   h. Set the DataGridView control's DataSource to lstAddress

7. In the Form's Closing Event make a call to a new Sub you will write called SaveAllAddress() defined as follows: (the code that will write out each entry in the addressDList object to the address.txt file)

   a. Declare a Try Catch block.
   b. Within the Try block start a For Each loop to iterate through each entry in the addressDList

```
For Each kvp As KeyValuePair(Of String, List(Of String)) _
                                In addressDList
```

c. Declare `addressEntry` of type `Address`

d. Create the new `addressEntry` object based on the `kvp.Key, kvp.Value.Item(0)`, and `kvp.Value.Item(1)`, which are the nickname, name and email values.

e.
```
sw = New StreamWriter(strFileName, True)

sw.WriteLine(addressEntry.Nickname + "," + _
             addressEntry.Name + "," + _
             addressEntry.Email)
```

f. Close the StreamWriter (within the For loop):
```
sw.Close()
```

g. In the Catch block, display a useful message

```
MessageBox.Show("Unable to write new address entry. Reason : "
            + vbCrLf + ex.Message.ToString())
```

The `Exception` object `ex` contains the error message.

8. The `Insert` process should
   a. first confirm that all the textbox values are non-blank. If they are, then construct a new `Address` object using the three values.
   b. If the nickname is found in the Dictionary List (use the `ContainsKey` method), then remove that `Dictionary` element (use the `Remove` method) and remove the Address object from the `lstAddress` List (use the function `FindAddress` to find a possible address object match).
   c. Once any elements are removed from the Dictionary List and the address List, then add the new address entry to them both.
   d. Set the DataGridView DataSource to `Nothing`, then set it to the `lstAddress` List so the new entry will appear in the `DataGridView`.

9. For the menu tool strip drag the `MenuStrip` control to the form's top left. Name the top level menu option as File, then enter three submenu items: Clear Search, Save, and Exit.
   a. Double click on the Clear Search menu option to bring up the Click event for that menu item. This Click event can do two things: set the Search TextBox Text to a "*", then call the `btnSearch_Click(sender, e)` sub to perform the Search.
   b. Double click on the Save menu option, and write the event handler. You should be able to use code your wrote above.
   c. Double click on the Exit menu option, and write the event handler. Your application should save the addresses, and close.

Information on DataGridView and DataSource. The sample code in this MSDN help file is an example of an *unbound* DataGridView (ie the programmer manually applies the data to the DataGridView).
http://msdn.microsoft.com/en-us/library/system.windows.forms.datagridview.aspx

Information on List:
http://msdn.microsoft.com/en-us/library/6sh2ey19.aspx#Y11430

Information on Dictionary:

http://msdn.microsoft.com/en-us/library/xfhwa508.aspx

Information on File Stream:

http://msdn.microsoft.com/en-us/library/system.io.filestream.aspx