Practical 8: Array and Class

NOTE: Turn on **Option Explicit** and **Option Strict** compilation options for your project.

Create a new windows form application project. Complete the following requirements:

1. The [Student] class

- Add a new class file named **Student.vb** to the project. The [**Student**] class should have 5 public properties: **Id**, **Name**, **Program**, **Year** and **CGPA**, as well as a parameterized constructor that initializes the 5 public properties.

```
Public Class Student

    Public Property Id As String
    Public Property Name As String
    Public Property Program As String
    Public Property Year As Integer
    Public Property CGPA As Decimal

    Public Sub New(id As String, name As String, program As String,
                   year As Integer, cgpa As Decimal)
        Me.Id = id
        Me.Name = name
        Me.Program = program
        Me.Year = year
        Me.CGPA = cgpa
    End Sub

End Class
```

2. The [App] module

- Add a new module file named **App.vb** to the project.
- Declare a public array that capable to hold 10 [**Student**] objects. In addition, declare a public integer that keeps track the student count (i.e. number of filled [**Student**] records).
- Create a public function that returns the next student Id based on the current student count. Valid student Ids are ranged from "88WAR00001" to "88WAR00010".

```
Module App

    Public Students(9) As Student
    Public Count As Integer = 0

    Public Function GetNextId() As String
        Return (Count + 1).ToString("88WAR00000")
    End Function

End Module
```

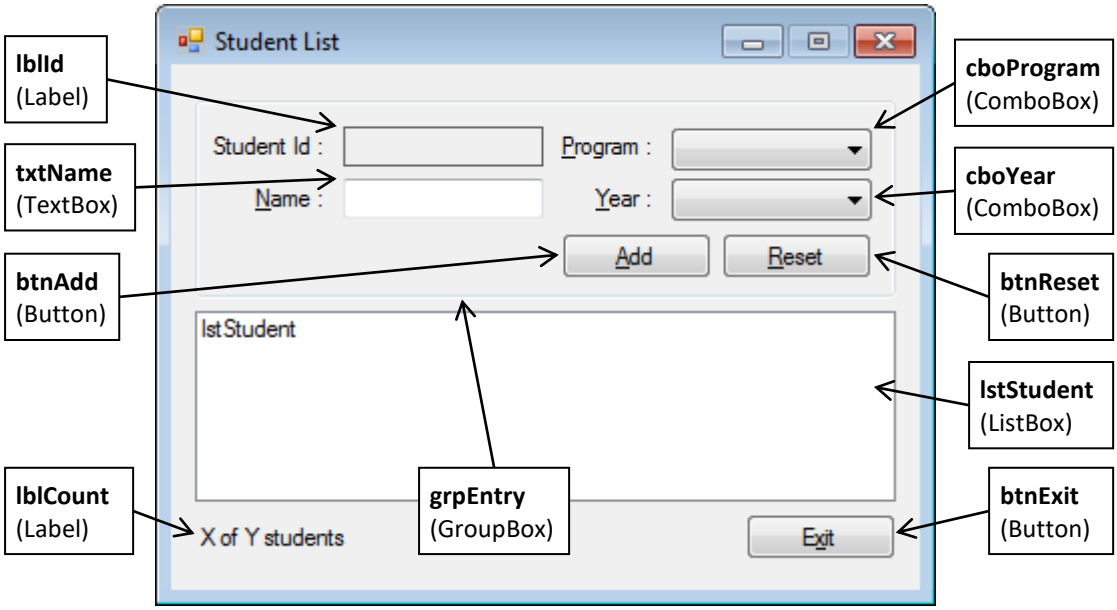
Array of [**Student**] objects (10 elements)

Keep track of the current student count

Return the next student Id (based on the count)

3. The [FrmList] form

- Add a new form named **FrmList.vb** to the project. The form has the following layout.



FrmList: GUI layout

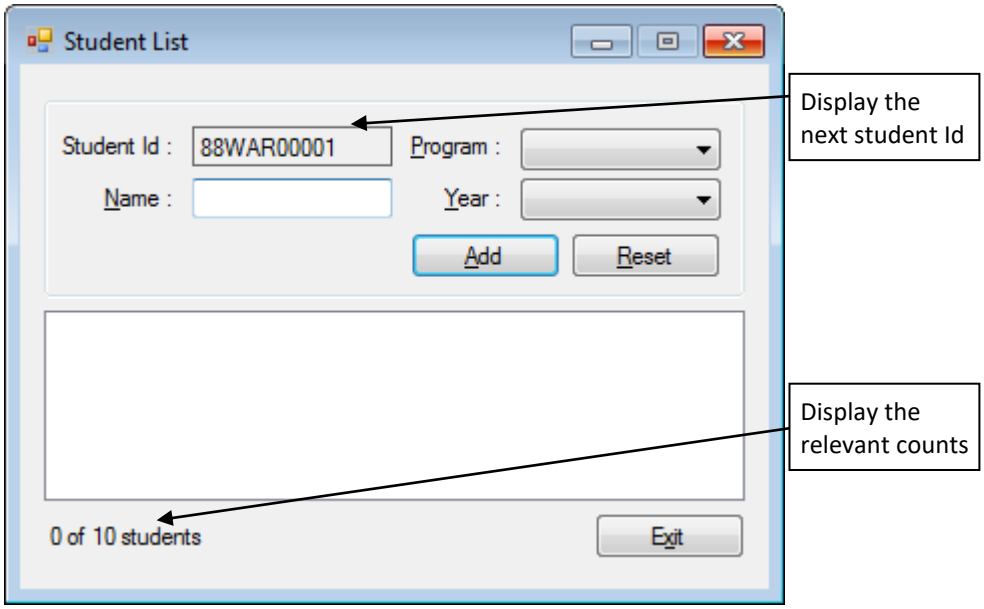
- Add the following items (in design-time) to **cboProgram** and **cboYear**.

cboProgram	cboYear
• REI • RIS • RIT • RSD • RSF • RST	• 1 • 2 • 3

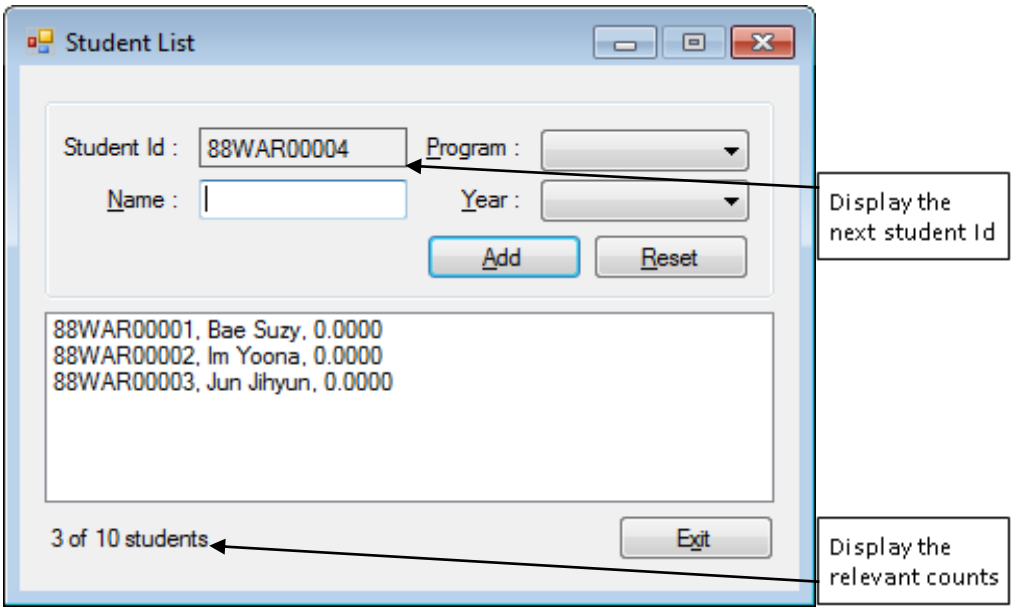
- Configure the form with appropriate usability features: access keys, tab order, accept button, cancel button, etc.

- Create a sub procedure named **UpdateList()** which:
  - Clear all items from **lstStudent**.
  - Display all [**Student**] objects stored in **App.Students** array in **lstStudent** (excluding empty elements). Show only student Id, name and CGPA (separated by commas).
  - Update **lblCount** with relevant counts.

- When the form is **loaded**, call the **UpdateList()** sub procedure and display the next student Id in **lblId** (by calling the **App.GetNextId()** function).



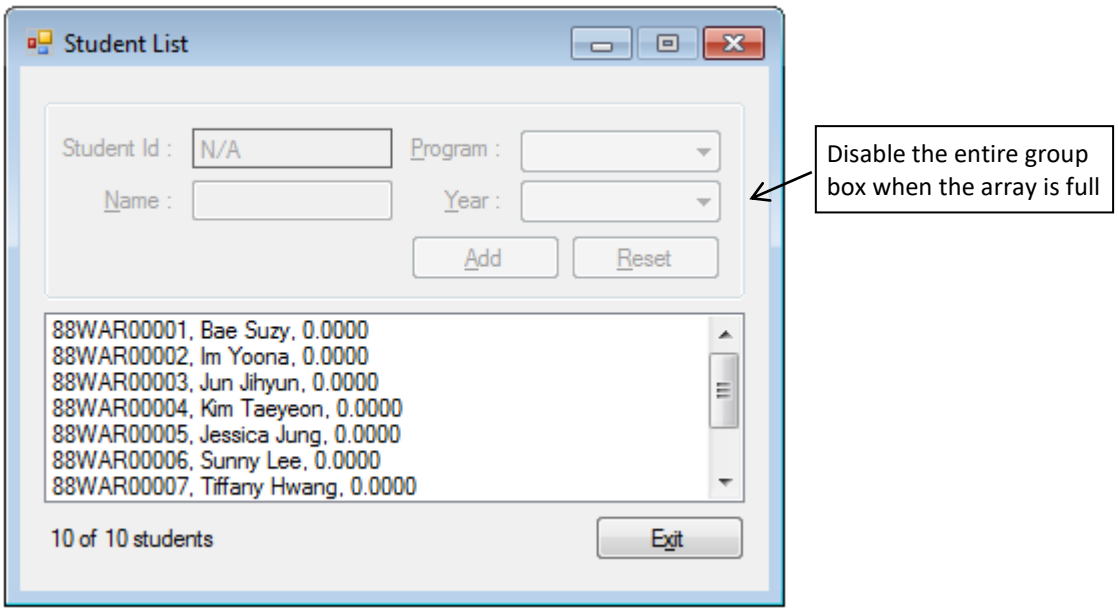
FrmList: When the form is loaded (and the list is empty)



FrmList: When there are 3 items in the list

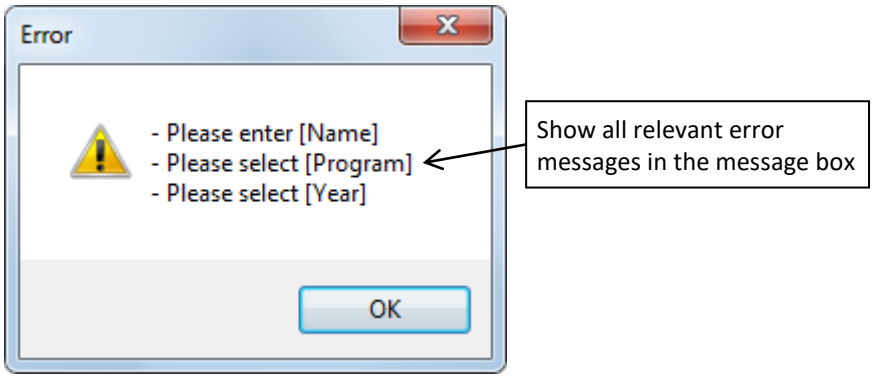
- When **btnExit** is clicked, close the form.

- When **btnReset** is clicked, perform the following:
  - Check the current student count (**App.Count**). If there are still spaces for new [**Student**] objects, display the next student Id in **lblCount**. Otherwise, display "N/A" in **lblId** and disable the entire group box (**grpEntry**).
  - Clear the inputs in **txtName**, **cboProgram** and **cboYear**.
  - Focus on **txtName**.



FrmList: When the array is full

- When **btnAdd** is clicked, perform the following:
  - Read and validate the inputs (refer to **Practical 7** if you forget how).
  - If there are input errors, display the collective error messages in a message box.
  - If there are no input errors, add the student record to the **App.Students** array, with a default CGPA of 0. Remember to increase the student count (**App.Count**).
  - Call the **UpdateList()** sub procedure to display the updated student list.
  - Perform a form reset (by calling the **btnReset_Click()** event handler).



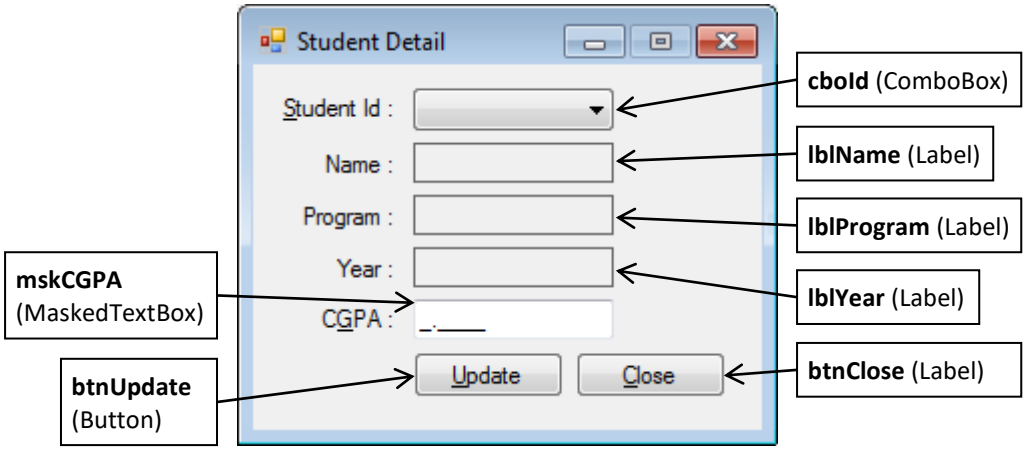
FrmList: Example of error messages

- When **lstStudent** is **double-clicked**, perform the following:
  - If an item is selected in **lstStudent**, pass its index to **FrmDetail**, show **FrmDetail** as a modal dialog, and finally call the **UpdateList()** sub procedure to display the updated student list.
  - Do nothing if no item is selected in **lstStudent**.

NOTE: You may want to program **FrmDetail** first before program for this double-click event.

4. The [FrmDetail] form

- Add a new form named **FrmDetail.vb** to the project. The form has the following layout.



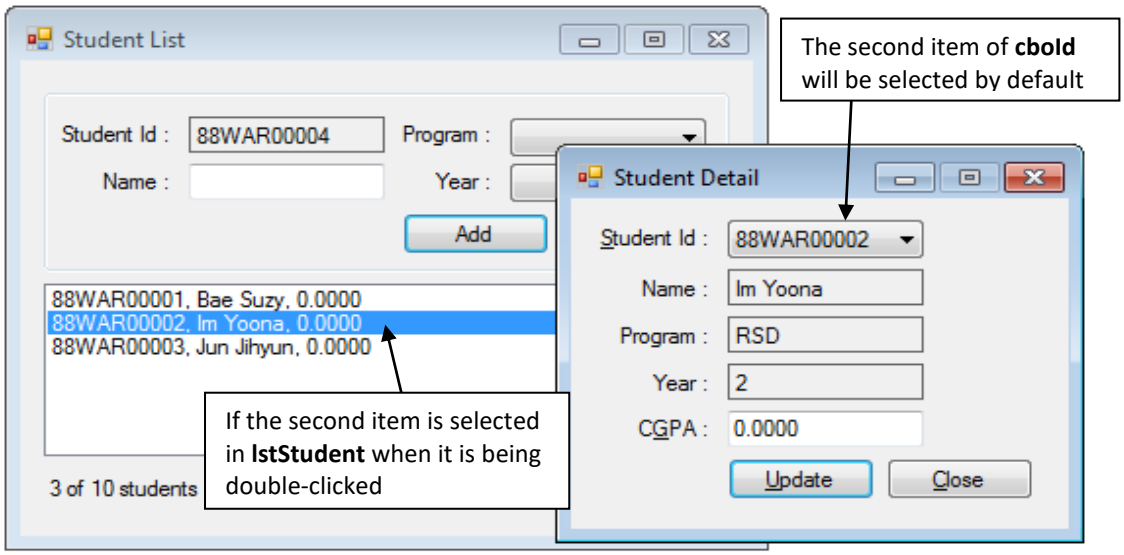
FrmDetail: GUI layout

- Set the mask of **mskCGPA** to "**0.9999**" (i.e. 1 required digit before the decimal point and 4 optional digits after the decimal point).

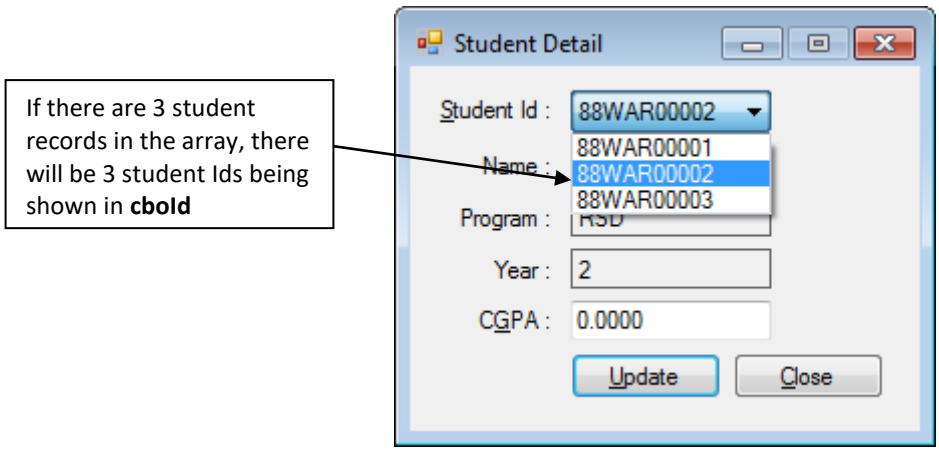
- Configure the form with appropriate usability features: access keys, tab order, accept button, cancel button, etc.

- Declare a public variable named **Index**, which allows **FrmList** to pass the selected index in **lstStudent** to the form.

- When the form is **loaded**, perform the following:
  - Clear all items from **cboId**.
  - Add the student Ids of the student records in the **App.Students** array into **cboId**.
  - Set the selected index of **cboId** by to the value of the public variable **Index** (i.e. the selected index of **lstStudent** in **FrmList**).



FrmDetail: Select the default student Id



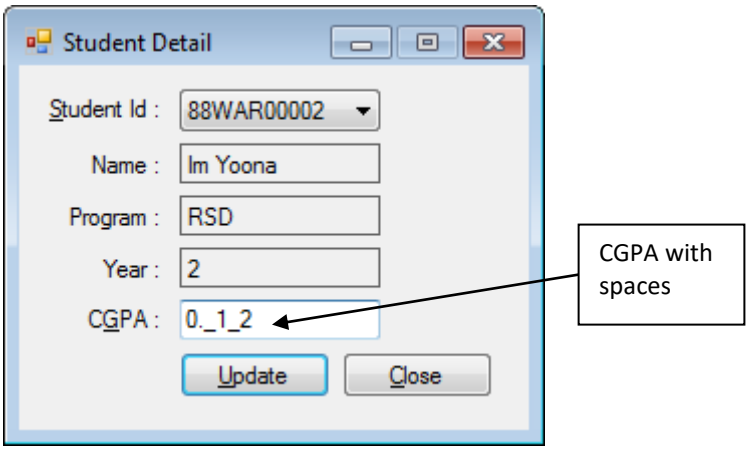
FrmDetail: Student Ids in **cboId**

- When the selected index of **cboId** is changed, retrieve and display name, program, year and CGPA of the selected student in the relevant labels and masked textbox.

- When **btnClose** is clicked, close the form.

- When **btnUpdate** is clicked, perform the following:
  - Read and validate the CGPA. CGPA must between 0.0000 and 4.0000 inclusively (refer to **Practical 7** if you forget how).
  - If there is input error, display the error message in a message box.
  - If there is no input error, update the CGPA of the selected student.
  - Close the form.

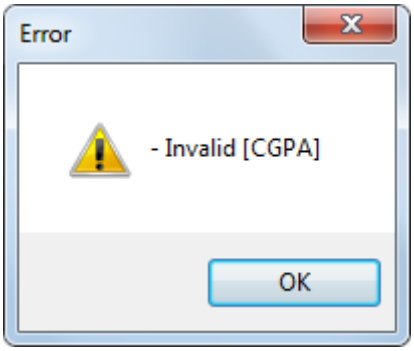
NOTE: Beware when you perform validation on CGPA. The mask "**0.9999**" allows spaces after the decimal point. For example, you may accept an input like the following:



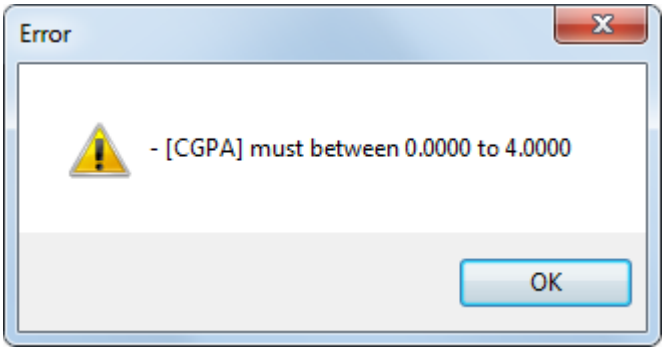
In this case, the CGPA will not be convertible to decimal type (exception will occur). One of the simple tricks to solve this problem is to replace spaces in the CGPA with zeros before the conversion:

```
CDec(mskCGPA.Text.Replace(" ", "0"))
```

Thus, an input of "0._1_2" will be treated as "0.0102" (which is logical, anyhow).



If the mask is not completed



If CGPA is out of range

- Test your solution.