Program 2: VB.NET Classes
25 points
Due Wednesday (Week 4), Feb. 10, 2016 by 5:00 PM

Program Description

The program manages a list of house units to be built. There are three types of houses: Platteville, Madison, and Chicago, and they are represented by three classes. The three classes are all subclasses of an abstract class House.

The solution has two projects: Project HouseClasses and Project Prog2GUI.

Project HouseClasses

1. This project is a class library project and defines the abstract class House and the three subclasses.
2. **The root namespace must be UWPCS3340, the assembly name must be HouseClasses.**
3. Abstract class House
   - The class keeps a list (an array or other collection) of all created House objects. The index must indicate the order in which the house objects are added, starting with 0.
   - You figure out other private/protected data members of the class.
   - Public shared properties

```
' Returns the total count of house objects to be built.
Public Shared ReadOnly Property TotalCount As Integer

' Returns the house at the specified index.
Public Shared ReadOnly Property HouseByIndex(ByVal index As Integer) As House
```

   - Public Properties

```
Public ReadOnly Property Type As String

Public ReadOnly Property ID As String

Public ReadOnly Property Rooms As Integer

Public ReadOnly Property Garages As Integer

Public ReadOnly Property Price As Single
```

   - Public Method

```
' Modifies number of rooms and number of garages, and hence
' the price of the house if both parameters are in the range.
' Otherwise, an exception is thrown with a message to indicate
' the invalid parameters (one or two).
' Event PriceChanged will be raised if both parameters are valid
' and the new price is different from the old price.
Public Sub Modify(ByVal numRooms As Integer, ByVal numGarages As Integer)
```

   - Public Event

```
' Raised when the price of the house changes.
Public Event PriceChanged()
```

4. The specifications of the three subclasses are given below.
   - For type Platteville, the number of rooms is between 2 and 3, the number of garages is between 1 and 2. The base price for two-room and one-garage is $200,000. The cost for the third room is $8,000, and the cost for the second garage is $2,500.
   - For type Madison, the number of rooms is between 2 and 4, the number of garages is between 1 and 3. The base price for two-room and one-garage is $300,000. The cost for an additional room is $10,000, and the cost for an additional garage is $5,000.
   - For type Chicago, the number of rooms is between 3 and 4, the number of garages is between 2 and 3. The base price for three-room and two-garage is $400,000. The price is $410,000 for three-room and three-garage, $420,000 for four-room and two-garage, and $428,000 for four-room and three-garage .
   - Each subclass must have a public constructor
     - The constructor must have a parameter for the ID.
     - A valid ID must have exactly five characters, and each character must be a digit or letter.
     - The ID must be unique.
     - If all conditions on ID are satisfied, a house object will be created with the minimum rooms and garages for the base price, and the type of the house is also set.
     - If any condition is violated, no house object will be created and an exception will be thrown.
5. The house class and the three subclasses can have other private or protected members, but they **MUST NOT** have additional

Public and Friend members.
6. Your grade will depend on how well you design the classes.
7. Remember that redudant code must be limited, and methods could be overridable.

Project Prog2GUI

1. The project is a Windows Forms Application project and has three classes: Prog2, FormClassHouse, and FormClassList.
2. Class Prog2 must be the Startup object and it must have a shared Sub Main, which should run the application with an object of FormClassHouse.
3. Class FormClassHouse
   - The GUI of the form class must be very similar to the sample program.
   - The class has a private data member _frmList, which should be set to an object of FormClassList to make it work.
   - The title of the form is "Program 2" followed by your first and last names, its size is (770, 490), and boarderstyle is Fixed3D. The form has a dropdown combobox for the user to choose the house type, a text box to enter the house ID, two groups of radiobuttons to select the number of rooms and the number of garages, a ReadOnly textbox to display the price of the house. The form also has four command buttons: btnSave, btnModify, btnList and btnExit.
   - When the form is loaded at the beginning, the dropdown combobox for the type has the focus with no item being selected, both ID and price textboxes are empty, no selection is made to rooms or garages, and btnModify is disabled.
   - Clicking btnSave will save a new house of the selected type and the specified ID to the list of houses. The client code does not validate any values for the house, but must catch any exceptions raised by the classes, including the exception when a new house has an ID that exists in the list already. After a new house is saved successfully, the ID textbox will be ReadOnly, the Type combobox will be disabled, the number of rooms and the number of the garages will be indicated in the radiobuttons, button btnSave will be disabled with the text changed to "NEW", while other buttons will be enabled.
   - Clicking btnModify will save the changes made to the house displayed on the form. The user can change the number of rooms and the number of garages, but not the ID or Type. The client code should just call the class methods without checking the input and catch any possible exceptions. When a house has been modified, the price will be modified accordingly and the new price will be displayed on the form.
   - Clicking btnSave with text "NEW" will clear all values on the form and the user can enter values for a new house, the ID textbox will not be ReadOnly, the Type combobox will be enabled, button btnModify will be disabled, and all other buttons will be enabled.
   - Clicking btnExit will terminate the project.
   - Clicking btnList will hide the form and display form _frmList in the center of the screen.
   - The price must be displayed in the currency format and aligned right.
   - **(5 points)**There must be an event handler to handle the PriceChanged event for all created House objects. The event procedure just displays a message box to indicate that the price has changed.
4. Class FormClassList
   - The following private member and public property are required:

```
' The variable must point to the FormClassHouse object that created this object of FormClassList.
Private _mainForm As FormClassHouse

Public WriteOnly Property MainForm As FormClassHouse
```

   - Your form must be very similar to the sample program.
   - The title is "All Houses" followed by your first and last names,, its size is (660, 390), and boarderstyle is FixedSingle.
   - The form has a textbox to display the total count of houses to be built.
   - The total count of houses should be correct at all times. You must use the shared property TotalCount to get the total count of houses and then display it.
   - The form has a listbox of size (277, 100) to display the ID, Type and price of all houses. The font of the listbox is Courier New with size 10.
   - You must display the houses in the listbox in the same way as the sample program.
   - The form has two command buttons: btnDisplay and btnOK.
   - Clicking btnOK will hide the form and show _mainForm.
   - Clicking btnDisplay will hide the form, show _mainForm with the data of the selected house, and _mainForm will be ready for user to modify the selected house.
   - The user should select a house before clicking btnDisplay. If no house is selected, then nothing will be changed and a message box will be displayed.
   - Hint: To format the houses in the listbox, set the font to a fixed-size font such as Courier New and use PadRight and/or PadLeft on strings and FormatCurrency on the price.

Submission

1. Name your solution folder as UserName_Prog2, where UserName is your UWP username.
2. Both projects MUST be inside the solution folder.
3. Drop your solution folder to the class DropBox by the due time.
4. You will lose two points for incorrect submission, including incorrect folder name.

5. You must follow the programming rules, and you may lose up to five points on style.
6. You will lose five (5) points if the startup object is incorrect.