CS 3340 - Prog 5: Readers and Writers
Due Friday, March 11, 2016, by 5 pm

The program gives a FIFO solution to the Reader and Writer problem. There is an integer variable *total*, and all readers and writers will access the same variable. A reader doesn't update the value and can share the variable with other readers. A writer can update the variable and needs exclusive access to the variable. All the readers and writers must follow the FIFO rule.

Program Solution

- The program has three projects: DatabaseClass, ReaderWriterClasses, and Prog5GUI.

Project DatabaseClass

- This is a class library project, the assembly name must be DatabaseClass, and the root namespace must be UWPCS3340.
- The project has one class DatabaseClass.
- The DatabaseClass defines an enumeration type:

```
Public Enum DatabaseStatus
   Reading
   Writing
   Empty
End Enum
```

- The DatabaseClass has the following private variables:

```
' The data shared by all readers and writers
Private _total As Integer

' To control data access
Private ReaderCount As Integer
Private WriterCount As Integer

' To enforce mutual exclusion on ReaderCount and WriterCount
Private DataObj As New Object
```

- The DatabaseClass has a constructor that has a parameter for the initial value of _total.
- The DatabaseClass has the following public methods:

```
' Returns Reading when some readers are reading the database value,
'         Writing when a writer is writing the database value,
'         Empty otherwise.
Public ReadOnly Property TheDatabaseStatus As DatabaseStatus

' Enter Monitor before exclusive access to ReaderCount and WriterCount.
Public Sub LockDataObj()

' Exit Monitor after exclusive access to ReaderCount and WriterCount.
Public Sub ReleaseDataObj()

' Gets and sets the data value
Public Property TotalValue As Integer

' Increments the ReaderCount by one
Public Sub IncreaseReaderCount()

' Decrements the ReaderCount by one.
Public Sub DecreaseReaderCount()

' Increments the WriterCount by one.
Public Sub IncreaseWriterCount()
```

```
' Decrements the WriterCount by one.
Public Sub DecreaseWriterCount()
```

- The DatabaseClass cannot have other public members.

Project ReaderWriterClasses

- This is a class library project, the assembly name must be ReaderWriterClasses, and the root namespace must be UWPCS3340.
- The project has three classes: ReaderWriter, Reader, and Writer.
- Class ReaderWriter
  - The ReaderWriter defines the following data types:

```
Public Enum ReaderWriterType
   Reader
   Writer
End Enum

Public Enum State
   Waiting
   Working
   Finished
End Enum

Public Delegate Sub PassMessage(ByVal theID As String,
                                ByVal theState As State,
                                ByVal theTotal As Integer)
```

  - The ReaderWriter has the following shared protected members:

```
Protected Shared FIFOQueue As New Queue
Protected Shared _database As UWPCS3340.DataBaseClass
Private Shared endProgram As New AutoResetEvent(False)
```

  - The ReaderWriter has the following protected members:

```
Protected _thread As Threading.Thread
Protected _passMsg As PassMessage
Protected _mainForm As System.Windows.Forms.Form
Protected _ReaderWriterEvent As New AutoResetEvent(False)
Protected _rondomGenerator As New System.Random
```

  - The ReaderWriter has the following shared property and methods:

```
Public Shared WriteOnly Property TheDatabase() As UWPCS3340.DataBaseClass

' When a writer or a reader exits the database and no other readers/writers
' are in the database, the writer or reader wakes up the the first reader/writer
' in the waiting queue.
' If the queue is empty, sets endProgram to signalled (green), since it's
' possible that a thread is waiting for all readers/writers to finish the work.
' Mutual exclusion on the queue must be enforced.
Protected Shared Sub WakeupNextWhenExiting()

' Waits for all readers and writers to finish the work in order to terminate
' the program.
' Mutual exclusion on the DataObj and the queue must be enforced.
Public Shared Sub FinishReadWrite()
```

  - The ReaderWriter has the following property and methods:

```
Public WriteOnly Property DisplayMsg() As PassMessage

Public WriteOnly Property MainForm() As System.Windows.Forms.Form
```

```
Public MustOverride ReadOnly Property ID() As String

Public MustOverride ReadOnly Property type() As ReaderWriterType

Public Sub SpinUp()

Public Sub WakeUp()

Protected MustOverride Sub run()
```

  - The class cannot have other public members.
- Class Reader
  - This is a sub-class of ReaderWriter
  - Class Reader must override the ID property to return ReaderWriterType.Reader.
  - Class Reader must override the Type property to return a string in the format format "Reader_XXnn", where "XX" is your initials and "nn" is the harsh code of the thread object, e.g., Reader_QY22.
  - Class Reader must override the run method according to the following pseudo-code:

```
Lock DataObj of the database
    Lock the FIFO queue
    If the queue is not empty or the database status is writing
       Add itself to the queue
       Invoke delegate _passMsg on the main form
       Release DataObj and the queue
       Wait for _ReaderWriterEvent
       Wakeup next reader before accessing the database
    Otherwise
       Release DataObj and the queue

Lock DataObj
Increase reader count by one
    Release DataObj

Invoke delegate _passMsg on the main form
Generate a random integer between 2000 and 3000
Work (sleep) for for the generated milliseconds
Invoke delegate _passMsg on the main form

Lock DataObj
Decrease reader count by one
If the database status is empty
   Wakeup next one in the queue
Release DataObj
```

- Class Writer
  - This is a sub-class of ReaderWriter
  - Class Writer must override the ID property to return ReaderWriterType.Writer.
  - Class Writer must override the Type property to return a string in the format format "Writer_XXnn", where "XX" is your initials and "nn" is the harsh code of the thread object, e.g., Writer_QY23.
  - Class Writer must override the run method according to the following pseudo-code:

```
Lock DataObj of the database
    Lock the FIFO queue
    If the queue is not empty or the database status is not empty
       Add itself to the queue
       Invoke delegate _passMsg on the main form
       Release DataObj and the queue
       Wait for _ReaderWriterEvent
    Otherwise
       Release DataObj and the queue

Lock DataObj
Increase writer count by one
    Release DataObj
Invoke delegate _passMsg on the main form
```

```
Generate a random integer between 3000 and 4000
Work (sleep) for for the generated milliseconds
    Generate a random integer between -10 and 10
    Update the database value using the generated integer
Invoke delegate _passMsg on the main form

Lock DataObj
Decrease writer count by one
Wakeup next one in the queue
Release DataObj
```

Project Prog5GUI

- This is a Windows Forms Application project.
- The project has two classes: Prog5 and ReaderWriterForm, and the startup object must be Sub Main or Prog5.
- Prog5 cannot have any variables, even inside the Sub Main.
- The title of ReaderWriterForm is "Readers and Writers - FIFO" followed by your first and last names. The border style is Fixed3D and its size is (850, 460).
- The form has a small textbox txtTotal with a label and a large textbox txtLog in the middle. Both textboxes are read only; at the beginning, txtTotal displays the initial value of 1000, and txtLog is empty.
- On the right side of the textboxes, there is a listbox, lstWaiting with a label of text "Waiting." On the left side, there is another listbox, lstWorking with a label of text "Working."
- There are 3 command buttons at the bottom with text "New Reader," "New Writer," and "Exit," respectively.
- Clicking button "New Reader" will create a new object of class Reader.
- Clicking button "New Writer" will create a new object of class Writer.
- The listbox lstWaiting displays all waiting readers and writers in the FIFO order.
- The listbox lstWorking displays all working readers and writers.
- The textbox txtLog displays messages from all readers and writers, with the latest ones shown at the bottom.
- Clicking button "Exit" will disable all the buttons and wait for all working and waiting readers/writers to finish their work. Then a message box will be displayed to ask if the user really wants to terminate the program. The program will be terminated if the user chooses Yes. Otherwise, the program will continue to run with the buttons enabled.
- When waiting for the readers and writers to finish their work, the messages from the readers and writers will be processed promptly and displayed in the textboxes on the form.

Submission

1. Name your solution folder as UserName_Prog5, using your UWP UserName.
2. Drop your solution folder to folder Prog5 of the class DROP folder by the due time.
3. You may lose up to two points for incorrect submission.
4. You must follow the programming rules, and you may lose up to five points on style.