La Güeb de Joaquín
Mi sitio de Internet

Interfaz IComparable

[Descripción general]

Hay ocasiones en las que es necesario saber si un objeto es mayor, menor, o igual que otro. Mejor dicho, cuando el VALOR del objeto es mayor, igual o menor que otro.
Esta es una situación que siempre se da cuando se necesitan ordenar los objetos contenidos por una colección. Para estos casos, Microsoft recomienda que se implemente la interfaz IComparable, que contiene una única función llamada "CompareTo"
.NET Framework define y utiliza docenas de interfaces y los programadores expertos en Visual Basic .NET deben aprender a sacar partido de todas ellas

Contenido

Introducción

Casi todas las clases Colección, (por ejemplo, System.Array, List, ArrayList, SortedDictionary (Of T), etc.) Exponen el método compartido Sort que permite ordenar los objetos contenidos en la colección, siempre y cuando sean del tipo de datos sencillos tales como números o cadenas (integer, doublé, char, string, etc.).

La ordenación de los datos se realiza por un proceso interno que compara los datos entre si y averigua si un dato es mayor igual o menor que otro y entonces se ordenan según ese valor.

Sin embargo, el método Sort, no puede ordenar objetos más complejos, por ejemplo, Persona, porque no sabe como comparar esos objetos entre sí.

El método Sort, está sobrecargado y una de sus sobrecargas admite la interfaz IComparable. Esto quiere decir que Sort sabe como ordenar a cualquier objeto que implemente la interfaz IComparable. Dicho de otra forma. Si queremos que un objeto cualquiera, por ejemplo Persona, pueda ser ordenado dentro de una colección utilizando el método Sort, deberemos implementar la interfaz IComparable.

La interfaz IComparable, expone un único método CompareTo, que recibe un objeto y devuelve -1, 0, +1, dependiendo de que el objeto actual sea menor que, igual a, o mayor que el objeto que se ha pasado como argumento.

En realidad ¿Es necesario implementar la interfaz? ¿No es suficiente con escribir la función?, pues si y pues no. El problema es que si tengo un montón de Personas en una colección y quiero ordenarlas, si está definida la interfaz IComparable todo es mucho más fácil, solo tengo que llamar al método 'Sort' de la colección, sino esta implementada tengo que realizar la ordenación a ´mano´

La clase para ver el ejemplo

Para ver el concepto vamos a trabajar con una clase Persona que se muestra a continuación

```
''' <summary>
'''   Clase Persona. EJEMPLO, para mostrar como se implementan las interfaces
''' </summary>
<Serializable()> _
Friend Class Persona
    Implements IComparable
    Implements IComparable(Of Persona)


    Private _nombre As String = String.Empty
    Private _apellidos As String = String.Empty
    Private _dni As String = String.Empty

    Public Property Nombre() As String
        Get
            Return _nombre
        End Get
        Set(ByVal value As String)
            _nombre = value
        End Set
    End Property
```

```
    Public Property Apellidos() As String
        Get
            Return _apellidos
        End Get
        Set(ByVal value As String)
            _apellidos = value
        End Set
    End Property


    Public Property DNI() As String
        Get
            Return _dni
        End Get
        Set(ByVal value As String)
            _dni = value
        End Set
    End Property


    Public Sub New(ByVal nombre As String, ByVal apellidos As String, ByVal dni As String)
        Me._nombre = nombre
        Me._apellidos = apellidos
        Me._dni = dni
    End Sub


    ' Devuelve el nombre y los apellidos
    Public ReadOnly Property ToNombreApellidos() As String
        Get
            Return String.Format(System.Globalization.CultureInfo.CurrentCulture, _
                                 " | {0,10}, {1,15} | ", _
                                 Me.Nombre, _
                                 Me.Apellidos)

        End Get
    End Property

    ' Devuelve los apellidos y el nombre
    Public ReadOnly Property ToApellidosNombre() As String
        Get
            Return String.Format(System.Globalization.CultureInfo.CurrentCulture, _
                                 " | {0,15}, {1,10} | ", _
                                 Me.Apellidos, _
                                 Me.Nombre)

        End Get
    End Property
End Class
```

La Interfaz

Firma de la interfaz

Las firmas de las interfaces son:

Public interface `IComparable`
Public interface `IComparable(Of T)`

Firma de las funciones de la interfaz

Las funciones que hay que implementar tienen la firma:

```
    Public Function CompareTo(ByVal obj As Object) As Integer _
           Implements System.IComparable.CompareTo
y
    Public Function CompareTo(ByVal other As Persona) As Integer _
           Implements System.IComparable(Of Persona).CompareTo
```

Implementando la Interfaz

Paso Uno Definir la interfaz

```
Friend Class Persona
    Implements IComparable
    Implements IComparable(Of Persona)
 . . .
 . . .
End Class
```

Paso Dos - Implementar la función de la interfaz

La función CompareTo tiene que comparar dos objetos y decidir si el primero es mayor que el segundo, si son iguales o si el primero es menor que el segundo.

Pero observa... hay que decidir si dos objetos son iguales, y esa es la tarea del método Equals. Dicho de otro modo, cuando escribamos el criterio de comparación de dos objetos, la parte del criterio que identifica a dos objetos como iguales debe ser la misma que la empleada en el método Equals.

IMPORTANTE: si mi método CompareTo tiene que funcionar igual que Equals, quiere decir que también hay que reemplazar el método Equals, lo que implica, (aunque no necesariamente) que se debe implementar la interfaz IEquatable y reemplazar el método Equals, el operador de igualdad (=) y el operador diferente(<>)

En este ejemplo, el **criterio de igualdad** empleado en la función Equals, es que el valor del campo DNI. Si dos clases tienen el mismo DNI se supone que corresponden a la misma persona y entonces son iguales.

Para escribir la función CompareTo tengo que elegir un **criterio de comparación** que no invalide la forma de trabajar de Equals. Lo que implica que para comparar dos objetos Persona, utilizare también el campo DNI

La función ´CompareTo´ mirara el valor del DNI de dos clases y en función de sus valores devolverá los valores [-1], [0], [+1]
Por ejemplo si se comparan los objetos D1 y D2 el resultado puede ser

- -1 si D1 < D2
- 0 si D1 = D2
- +1 si D1 > D2

Como son dos funciones que hacen el mismo trabajo, lo que hago es que la función genérica (Of T) es la que haga el trabajo real y la otra la llame para obtener el resultado

```
#Region "    Implements IComparable; IComparable(Of Persona)"

  '-------------------------------------------------------------------
    'La interfaz IComparable
    'http://msdn.microsoft.com/es-es/library/system.icomparable.compareto.aspx
    '-------------------------------------------------------------------
    Public Function CompareTo(ByVal other As Persona) As Integer _
           Implements System.IComparable(Of Persona).CompareTo
        '-------------------------------------------------------------------------
        ' A la hora de ordenar las personas, despues de sesudas y profundas
        ' meditaciones considero que debo ordenarlas por el valor del DNI
        '-------------------------------------------------------------------------

        '-----------------------------------------------------------
        ' Por definición, cualquier objeto es mayor que una referencia de
        ' objeto null (Nothing en Visual Basic) y dos
        ' referencias nulas son iguales entre sí.
        If other Is Nothing Then
            ' Mayor que cero  -  ME instancia es mayor que value.
            Return 1
        End If

        '-----------------------------------------------------------
        ' realizar la comparacion
        '-----------------------------------------------------------
        ' Menor que cero  -  ME instancia es menor que value.
        ' Cero            -  ME instancia es igual a value.
        ' Mayor que cero  -  ME instancia es mayor que value.
        '-----------------------------------------------------------
        'Como DNI es una cadena, y voy a comparar dos cadenas,
        'en lugar de escribir todo el proceso de comparación de cadenas
        'utilizo la clase StringComparer que realiza ese trabajo
        '-----------------------------------------------------------
        ' definir una clase [StringComparer] que utilice la
        ' referencia cultural actual y un sistema de comparación IgnoreCase,
        ' es decir que no distinga entre mayúsculas y minúsculas
        Dim SC As StringComparer
        SC = StringComparer.Create(System.Globalization.CultureInfo.CurrentCulture, True)

        ' realizar la comparacion de objetos
        '-----------------------------------------------------------
        ' APUNTE TACTICO
        ' Hay un problema con la comparación de cadenas cuando las cadenas tienen números.
        ' Porque se comparan como su fueran letras y se ordenan como las letras, de forma
        ' que la 'cadena' [2] está delante la 'cadena' [10] o [159]
        ' (porque una cadena que tiene menos caracteres que otra
        '  siempre es más pequeña y va delante)
        '
        ' En este caso y como solo es para ejemplo no se trata este problema
        '-----------------------------------------------------------
        Dim resultado As Integer = 0
        resultado = SC.Compare(Me.DNI, other.DNI)

        ' devolver el resultado
        Return resultado
    End Function
```

```
    '-----------------------------------------------------------------
    'La interfaz IComparable
    'http://msdn.microsoft.com/es-es/library/system.icomparable.compareto.aspx
    '-----------------------------------------------------------------
    Public Function CompareTo(ByVal obj As Object) As Integer _
              Implements System.IComparable.CompareTo
        '-------------------------------------------------------------------
        ' A la hora de ordenar las personas, despues de sesudas y profundas
        ' meditaciones considero que debo ordenarlas por el valor del DNI
        '-------------------------------------------------------------------


        '-----------------------------------------------------------
        ' El parámetro, obj, debe ser del mismo tipo que la clase o el
        ' tipo de valor que implementa esta interfaz; en caso contrario,
        ' se produce la excepción ArgumentException.
        '-----------------------------------------------------------
        If Not (obj.GetType Is Me.GetType) Then
            Throw New ArgumentException( _
            "Los objetos a comparar no son del mismo tipo")
        End If

        '-----------------------------------------------------------
        ' Por definición, cualquier objeto es mayor que una referencia de
        ' objeto null (Nothing en Visual Basic) y dos
        ' referencias nulas son iguales entre sí.
        If obj Is Nothing Then
            ' Mayor que cero  -  ME instancia es mayor que value.
            Return 1
        End If

        '-----------------------------------------------------------
        ' si llega aquí, quiere decir que los dos objetos son del mismo tipo
        ' y posiblemente los dos son objetos Persona
        '-----------------------------------------------------------

        '-----------------------------------------------------------
        ' realizar la comparacion
        '-----------------------------------------------------------
        Try
            '-------------------------------------------------------------
            ' Si alguno de los objetos no es una clase Persona
            ' se dispara la excepcion [ArgumentException]
            'convertir el parametro en un objeto Persona
            Dim personaParametro As Persona
            personaParametro = CType(obj, Persona)
            '-------------------------------------------------------------
            ' utilizar la funcion GENERICA para hacer la comparacion
            Return Me.CompareTo(personaParametro)
        Catch ex As Exception
            Throw New ArgumentException(ex.Message, ex)
        End Try
    End Function

#End Region
```

Implementación Avanzada

La biblioteca MSDN en sus instrucciones para Implementar el método Equals, e Instrucciones para implementar Equals y el operador de igualdad (==) dice lo siguiente: (Texto muy resumido del original)

- Si implementa la interfaz IComparable en un tipo dado, deberá reemplazar el método Equals en ese tipo.
- Reemplace el método Equals siempre que implemente IComparable.
- Plantéese la posibilidad de implementar la sobrecarga de los operadores de igualdad (==), distinto de (!=), menor que (<) y mayor que (>) cuando implemente IComparable.

Implementar Operadores menor que (<) y mayor que (>)

```
#Region "    Implements IComparable; IComparable(Of Persona)"

    '-------------------------------------------------------------------
    'Instrucciones para implementar Equals y el operador de igualdad (==)
    'http://msdn.microsoft.com/es-es/library/7h9bszxx.aspx
    '-------------------------------------------------------------------
    Public Overloads Shared Operator <( _
                    ByVal per1 As Persona, _
                    ByVal per2 As Persona) _
                    As Boolean

        Dim resultado As Boolean = False
        If per1.DNI < per2.DNI Then
            resultado = True
        End If
        Return resultado

    End Operator


    '-------------------------------------------------------------------
    'Instrucciones para implementar Equals y el operador de igualdad (==)
    'http://msdn.microsoft.com/es-es/library/7h9bszxx.aspx
    '-------------------------------------------------------------------
    Public Overloads Shared Operator >( _
                    ByVal per1 As Persona, _
                    ByVal per2 As Persona) _
                    As Boolean
        Dim resultado As Boolean = False
        If per1.DNI > per2.DNI Then
            resultado = True
        End If
        Return resultado
    End Operator
#End Region
```

Implementar la interfaz IEquatable

En estos casos lo mejor es implementar la interfaz IEquatable y escribir las funciones que sugiere esa interfaz:

```
#Region " Implements IEquatable(Of Persona)"

    '-------------------------------------------------------------------
    'Interfaz IEquatable(Of T)
    'http://msdn.microsoft.com/es-es/library/ms131187.aspx
    'Determinar si dos objetos tienen el mismo VALOR.
    '-------------------------------------------------------------------
    'La interfaz IEquatable
    '-------------------------------------------------------------------
    Public Overloads Function Equals(ByVal other As Persona) As Boolean _
              Implements System.IEquatable(Of Persona).Equals

    '-------------------------------------------------------------------
    '  Sombreado Object.Equals
    Public Overloads Overrides Function Equals(Object) As Boolean
```

```
    '--------------------------------------------------------------------
    '  Sombreado Object.Equals
    '--------------------------------------------------------
    ' Aquí es donde se hace TODO el trabajo de comparación
    '--------------------------------------------------------
    Public Overloads Shared Function Equals( _
                    ByVal per1 As Persona, _
                    ByVal per2 As Persona) _
                    As Boolean
        Dim resultado As Boolean = False

        If per1.DNI = per2.DNI Then
            resultado = True
        End If
        Return resultado
    End Function


    '--------------------------------------------------------------------
    ' Implementar el operador de igualdad (=)
    '--------------------------------------------------------------------
    Public Overloads Shared Operator =(Persona, Persona) As Boolean

    '--------------------------------------------------------------------
    ' Implementar el operador DiferenteDe (<>)
    Public Overloads Shared Operator <>(Persona, Persona) As Boolean

    '--------------------------------------------------------------------
    'Sombreado Object.GetHashCode
    'http://msdn.microsoft.com/es-es/library/system.object.gethashcode.aspx
    Public Overrides Function GetHashCode() As Integer

#End Region
```

A modo de resumen

La implementación de la interfaz IComparable implica a su vez la implementación de la interfaz IEquatable.

```
#Region " Implements IEquatable(Of Persona)"
    '-------------------------------------------------------------------
    'Interfaz IEquatable(Of T)
    'http://msdn.microsoft.com/es-es/library/ms131187.aspx
    'Determinar si dos objetos tienen el mismo VALOR.
    '-------------------------------------------------------------------
    'Para determinar si tienen la misma REFERENCIA utilizar la función
    'Object.ReferenceEquals (Método)
    'http://msdn.microsoft.com/es-es/library/system.object.referenceequals.aspx
    '-------------------------------------------------------------------

    '-------------------------------------------------------------------
    'La interfaz IEquatable
    'http://msdn.microsoft.com/es-es/library/ms131190.aspx
    '-------------------------------------------------------------------
    Public Overloads Function Equals(ByVal other As Persona) As Boolean _
              Implements System.IEquatable(Of Persona).Equals
        Return Equals(Me, other)
    End Function


    '-------------------------------------------------------------------
    '  Sombreado Object.Equals
    '  http://msdn.microsoft.com/es-es/library/system.object.equals.aspx
    '-------------------------------------------------------------------
    Public Overloads Overrides Function Equals(ByVal obj As Object) As Boolean
        '-----------------------------------------------------------
        ' El parámetro, obj, debe ser del mismo tipo que la clase o el
        ' tipo de valor que implementa esta interfaz
        '-----------------------------------------------------------
        If Not (obj.GetType Is Me.GetType) Then
            Return False
            '"Los objetos a comparar no son del mismo tipo")
        End If
        '-----------------------------------------------------------
        Dim unapersona As Persona
        unapersona = CType(obj, Persona)
        Return Equals(Me, unapersona)
    End Function


    '-------------------------------------------------------------------
    '  Sombreado Object.Equals
    '  http://msdn.microsoft.com/es-es/library/system.object.equals.aspx
    '-------------------------------------------------------------------
    '-------------------------------------------------------
    ' Aquí es donde se hace TODO el trabajo de comparación
    '-------------------------------------------------------
    Public Overloads Shared Function Equals( _
                    ByVal per1 As Persona, _
                    ByVal per2 As Persona) _
                    As Boolean
        Dim resultado As Boolean = False

        If per1.DNI = per2.DNI Then
            resultado = True
        End If
        Return resultado
    End Function
```

```
    '-----------------------------------------------------------------------
    'Instrucciones para implementar Equals y el operador de igualdad (==)
    'http://msdn.microsoft.com/es-es/library/7h9bszxx.aspx
    '-----------------------------------------------------------------------
    Public Overloads Shared Operator =( _
                    ByVal per1 As Persona, _
                    ByVal per2 As Persona) _
                    As Boolean
        Return Equals(per1, per2)
    End Operator


    '-----------------------------------------------------------------------
    'Instrucciones para implementar Equals y el operador de igualdad (==)
    'http://msdn.microsoft.com/es-es/library/7h9bszxx.aspx
    '-----------------------------------------------------------------------
    Public Overloads Shared Operator <>( _
                    ByVal per1 As Persona, _
                    ByVal per2 As Persona) _
                    As Boolean
        Return Not (Equals(per1, per2))
    End Operator


    '-----------------------------------------------------------------------
    'Sombreado Object.GetHashCode
    'http://msdn.microsoft.com/es-es/library/system.object.gethashcode.aspx
    'http://msdn.microsoft.com/es-es/library/7h9bszxx.aspx
    '-----------------------------------------------------------------------
    Public Overrides Function GetHashCode() As Integer
        Return _dni.GetHashCode()
    End Function


#End Region


#Region "    Implements IComparable; IComparable(Of Persona)"

    '-----------------------------------------------------------------------
    'Instrucciones para implementar Equals y el operador de igualdad (==)
    'http://msdn.microsoft.com/es-es/library/7h9bszxx.aspx
    '-----------------------------------------------------------------------
    Public Overloads Shared Operator <( _
                    ByVal per1 As Persona, _
                    ByVal per2 As Persona) _
                    As Boolean

        Dim resultado As Boolean = False
        If per1.DNI < per2.DNI Then
            resultado = True
        End If
        Return resultado

    End Operator


    '-----------------------------------------------------------------------
    'Instrucciones para implementar Equals y el operador de igualdad (==)
    'http://msdn.microsoft.com/es-es/library/7h9bszxx.aspx
    '-----------------------------------------------------------------------
    Public Overloads Shared Operator >( _
                    ByVal per1 As Persona, _
                    ByVal per2 As Persona) _
                    As Boolean
        Dim resultado As Boolean = False
        If per1.DNI > per2.DNI Then
            resultado = True
        End If
        Return resultado
    End Operator
```

```
'-------------------------------------------------------------------
'La interfaz IComparable
'http://msdn.microsoft.com/es-es/library/system.icomparable.compareto.aspx
'-------------------------------------------------------------------
Public Function CompareTo(ByVal obj As Object) As Integer _
          Implements System.IComparable.CompareTo
    '-------------------------------------------------------------------
    ' A la hora de ordenar las personas, despues de sesudas y profundas
    ' meditaciones considero que debo ordenarlas por el valor del DNI
    '-------------------------------------------------------------------


    '-------------------------------------------------------------
    ' El parámetro, obj, debe ser del mismo tipo que la clase o el
    ' tipo de valor que implementa esta interfaz; en caso contrario,
    ' se produce la excepción ArgumentException.
    '-------------------------------------------------------------
    If Not (obj.GetType Is Me.GetType) Then
        Throw New ArgumentException( _
        "Los objetos a comparar no son del mismo tipo")
    End If

    '-------------------------------------------------------------
    ' Por definición, cualquier objeto es mayor que una referencia de
    ' objeto null (Nothing en Visual Basic) y dos
    ' referencias nulas son iguales entre sí.
    If obj Is Nothing Then
        ' Mayor que cero  -  ME instancia es mayor que value.
        Return 1
    End If


    '-------------------------------------------------------------
    ' si llega aquí, quiere decir que los dos objetos son del mismo tipo
    ' los dos son objetos Persona
    '-------------------------------------------------------------
    'convertirlo a el parametro a objeto Persona
    Dim personaParametro As Persona
    personaParametro = CType(obj, Persona)

    '-------------------------------------------------------------
    ' realizar la comparacion
    '-------------------------------------------------------------
    ' utilizar la funcion GENERICA para hacer la comparacion
    Return Me.CompareTo(personaParametro)
End Function



'-------------------------------------------------------------------
'La interfaz IComparable(Of Persona)
'http://msdn.microsoft.com/es-es/library/43hc6wht.aspx
'-------------------------------------------------------------------
Public Function CompareTo(ByVal other As Persona) As Integer _
          Implements System.IComparable(Of Persona).CompareTo
    '-------------------------------------------------------------------
    ' A la hora de ordenar las personas, despues de sesudas y profundas
    ' meditaciones considero que debo ordenarlas por el valor del DNI
    '-------------------------------------------------------------------


    '-------------------------------------------------------------
    ' Por definición, cualquier objeto es mayor que una referencia de
    ' objeto null (Nothing en Visual Basic) y dos
    ' referencias nulas son iguales entre sí.
    If other Is Nothing Then
        ' Mayor que cero  -  ME instancia es mayor que value.
        Return 1
    End If
```

```
        '-------------------------------------------------------------
        ' realizar la comparacion
        '-------------------------------------------------------------
        ' Menor que cero  -  ME instancia es menor que value.
        ' Cero            -  ME instancia es igual a value.
        ' Mayor que cero  -  ME instancia es mayor que value.
        '-------------------------------------------------------------
        'Como DNI es una cadena, y voy a comparar dos cadenas,
        'en lugar de escribir todo el proceso de comparación de cadenas
        'utilizo la clase StringComparer que realiza ese trabajo
        '-------------------------------------------------------------
        ' definir una clase [StringComparer] que utilice la
        ' referencia cultural actual y un sistema de comparación IgnoreCase,
        ' es decir que no distinga entre mayúsculas y minúsculas
        Dim SC As StringComparer
        SC = StringComparer.Create(System.Globalization.CultureInfo.CurrentCulture, True)

        ' realizar la comparacion de objetos
        '-------------------------------------------------------------
        ' APUNTE TACTICO
        ' Hay un problema con la comparación de cadenas cuando las cadenas tienen números.
        ' Porque se comparan como su fueran letras y se ordenan como las letras, de forma
        ' que la 'cadena' [2] está delante la 'cadena' [10] o [159]
        ' (porque una cadena que tiene menos caracteres que otra siempre es más pequeña y va
delante)
        '
        ' En este caso y como solo es para ejemplo no se trata este problema
        '-------------------------------------------------------------
        Dim resultado As Integer = 0
        resultado = SC.Compare(Me.DNI, other.DNI)

        ' devolver el resultado
        Return resultado
    End Function

#End Region
```

Más información:

Limitaciones y condiciones para escribir la función CompareTo

En realidad ¿Es necesario implementar la interfaz? ¿No es suficiente con escribir la función?, pues si y pues no. El problema es que si tengo un montón de personas en una colección y quiero ordenarlas, si está definida la interfaz IComparable todo es mucho más fácil, solo tengo que llamar al método 'sort' de la colección, sino esta implementada tengo que realizar la ordenación a ´mano´

En la biblioteca MSDN tenemos la siguiente información

A) La firma de la función tiene que ser la siguiente:

```
Function CompareTo (obj as  Object) As Integer
```

- Observa que se recibe un único objeto, ¿Con quién se compara? Pues con la instancia del objeto que llama a la función es decir con 'ME'

B) El Parámetro: Object - es el Objeto que se va a comparar con esta instancia.

Valor devuelto: es un valor `Integer – un` Entero de 32 bits con signo que indica el orden relativo de los objetos que se está comparando. El valor devuelto tiene los siguientes significados:

- Menor que cero (<0) Esta instancia es menor que obj.
- Cero (=0) Esta instancia es igual a obj.
- Mayor que cero (>0) Esta instancia es mayor que obj.

C) Excepciones: La función debe disparar la excepción [ArgumentException] si el objeto recibido a través del parámetro no es del mimo tipo que esta instancia.

D) Comentarios

- Se puede comparar null (Nothing en Visual Basic) con cualquier tipo sin que se genere una excepción
  - Por definición, cualquier objeto es mayor que una referencia Null (Nothing en Visual Basic)(null) se considera menor que cualquier otro objeto.
  - Dos referencias Null (Nothing en Visual Basic) son iguales entre sí.
- El parámetro, obj, debe ser del mismo tipo que la clase o el tipo de valor que implementa esta interfaz; en caso contrario, se produce la excepción ArgumentException.

E) Condiciones lógicas básicas de la función

- A.CompareTo(A) debe devolver cero.
- Si A.CompareTo(B) devuelve cero, entonces B.CompareTo(A) debe devolver cero.
- Si A.CompareTo(B) devuelve cero y B.CompareTo(C) devuelve cero, entonces A.CompareTo(C) debe devolver cero.
- Si A.CompareTo(B) devuelve un valor distinto de cero, entonces B.CompareTo(A) debe devolver un valor del signo contrario.
- Si A.CompareTo(B) devuelve un valor x distinto de cero y B.CompareTo(C) devuelve un valor y del mismo signo que x, entonces A.CompareTo(C) debe devolver un valor del mismo signo que x e y.

Código de ejemplo

En el siguiente enlace hay un fichero ZIP que contiene ejemplo

- Fichero con el código de ejemplo: [2008_10_08_SolucionInterfazIComparer.zip]
- MD5 checksum: [139F3E0FA74D626AE82DC4212DEF6671]
- MD5 checksum: Información
  - [http://www.elguille.info/colabora/MD5_checksum.aspx]

Bibliografía

- *Programación* Avanzada con Microsoft Visual Basic .NET
  Francisco Balena
  ISBN 84-481-3715-9
  McGraw-Hill

Información de este Documento

[Historial de este documento]

- *Autor*:
  - *Nombre*: Joaquín Medina Serrano
  - *Email* : joaquin@medina.name
  - *Web*: http://joaquin.medina.name

- *Publicador*:
  - *Nombre*: Joaquín Medina Serrano

- *Fechas* (Formato de fecha ISO 8610:2004. [Año-Mes-DiaTHora:Minutos])
  - *Fecha de Creación*: 2008-09-05T23:06
  - *Fecha de la última modificación*: 2008-10-08T20:51
  - *Fecha de Impresión*: 2008-10-08T20:51

- *Copyright*
  - © Joaquín 'jms32' Medina Serrano 2008 - Reservados todos los derechos."

[Direcciones Web Interesantes]

Biblioteca de clases de .NET Framework
[http://msdn.microsoft.com/es-es/library/ms229335.aspx]

- IComparable (Interfaz)
  - http://msdn.microsoft.com/es-es/library/system.icomparable.aspx
- IComparable.CompareTo (Método)
  - http://msdn.microsoft.com/es-es/library/system.icomparable.compareto.aspx
- IComparable(Of T) (Interfaz genérica)
  - http://msdn.microsoft.com/es-es/library/4d7sx9hd.aspx
- IComparable(Of T).CompareTo (Método)
  - http://msdn.microsoft.com/es-es/library/43hc6wht.aspx
- IComparer (Interfaz)
  - http://msdn.microsoft.com/es-es/library/system.collections.icomparer.aspx
- IComparer..::.Compare (Método)
  - http://msdn.microsoft.com/es-es/library/system.collections.icomparer.compare.aspx
- StringComparer (Clase)
  - http://msdn.microsoft.com/es-es/library/system.stringcomparer.aspx

Manual del programador de .NET Framework
[http://msdn.microsoft.com/es-es/library/sxe8hcf2.aspx]

- Instrucciones para implementar Equals y el operador de igualdad (==)
  - http://msdn.microsoft.com/es-es/library/7h9bszxx.aspx
  - http://msdn.microsoft.com/es-es/library/ms229035.aspx
- Implementar el método Equals
  - http://msdn.microsoft.com/es-es/library/336aedhh.aspx

- Instrucciones de uso
  - http://msdn.microsoft.com/es-es/library/ms229035.aspx

[Palabras Clave]

- Sort, IComparable, 'nterfaz IComparable', Equals , interfaz, IEquatable, 'interfaz IEquatable', 'tipos por valor', 'tipos por referencia', variables, Equals, instancia, 'misma instancia', comparación, 'comparación de objetos', GetHashCode

[Grupo de documentos]

- [Documento Index]
- [Documento Start]