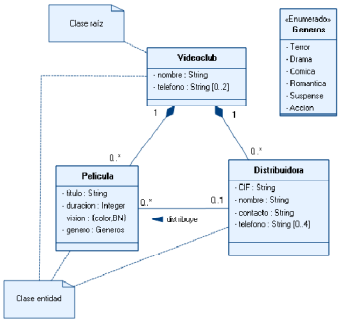
Diseño Basado en Componentes

Construcción de componentes utilizando VB.NET (Parte I)

Ingeniería Informática
Universidad Carlos III de Madrid

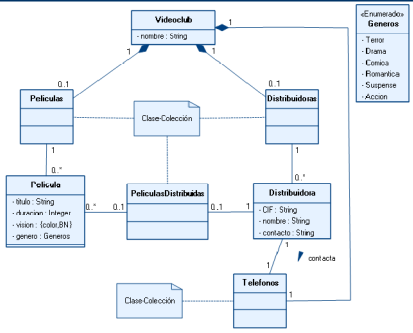
Ejemplo de Modelo



Clases entidad
Atributos miembro (I)

- Todos los atributos miembros deberían tener visibilidad privada, con una propiedad pública para su acceso.
- Deberá haber un miembro por cada atributo y por cada extremo de asociación. Si la multiplicidad máxima > 1 → el tipo del atributo será una **clase-colección**. En caso contrario será un atributo simple, ya sea un tipo básico (*String, Integer...*) u otro distinto.
- Un atributo más servirá para ligarle a la clase raíz del modelo.

Clases entidad
Atributos miembro (II)



Clases entidad
Atributos miembro (III)

```
Public Class Distribuidora
    Implements IDisposable

    Private _CIF As String
    Private _nombre As String
    Private _contacto As String
    Private _telefonos As Telefonos
    Private _peliculas As PeliculasDistribuidas
    Private _vc As Videoclub
    Private _isDisposed As Boolean
```


Clases entidad
Propiedades (I)

- Cada atributo o extremo con multiplicidad máxima 1 debe ir asociado con una propiedad pública.
- Esta propiedad deberá controlar la corrección de los datos (CIF válido, nombre no nulo...).
- Cada atributo o extremo con multiplicidad > 1 se modelará mediante una clase-colección, pero deberá crearse una propiedad (*ReadOnly*) para dar acceso a la misma.

Clases entidad
Propiedades (II)

```
'Clase Distribuidora
Public ReadOnly Property CIF() As String
        Get
            Return _CIF
        End Get
End Property
Public Property Nombre() As String
        Get
            Return _nombre
        End Get
        Set(ByVal Value As String)
            If Value Is Nothing Then
                Throw New System.ArgumentNullException
            ElseIf Value.Length = 0 Then
                Throw New System.ArgumentException("La longitud del nombre ha
    de ser mayor que cero")
            Else
                _nombre = Value
            End If
        End Set
End Property
```


Clases entidad
Propiedades (III)

```
Public Property Contacto() As String
        Get
            Return _contacto
        End Get
        Set(ByVal Value As String)
          If Value Is Nothing Then
                Throw New System.ArgumentNullException
            ElseIf Value.Length = 0 Then
              Throw New System.ArgumentException("La longitud del contacto ha de ser
    mayor que cero")
            Else
                _contacto = Value
            End If
        End Set
End Property
Public ReadOnly Property Telefonos() As Telefonos
        Get
            If _telefonos Is Nothing Then
                _telefonos = New Telefonos()
            End If
            Return _telefonos
        End Get
End Property
```

Clases entidad
Propiedades (IV)

```
Public ReadOnly Property PeliculasDistribuidas() As PeliculasDistribuidas
        Get
            If _peliculas Is Nothing Then
                _peliculas = New PeliculasDistribuidas(Me)
            End If
            Return _peliculas
        End Get
End Property

Friend ReadOnly Property VideoClub() As VideoClub
        Get
            Return _vc
        End Get
End Property
```


Clases entidad
Constructor (I)

- Método público que tendrá como parámetros todo atributo o extremo de asociación que tenga multiplicidad mínima 1.
- Debe tener como parámetro una referencia a la clase raíz a la que pertenecerá (ej. Clase Videoclub).
- Utilizando esta referencia a la raíz, cuando se cree uno de estos objetos se ha de añadir a la correspondiente colección de la clase raíz.

Clases entidad
Constructor (II)

```
'Clase Distribuidora
Public Sub New (ByVal CIF As String, ByVal nombre As String, ByVal
    contacto as String, telefono As String, videoclub As Videoclub)
    `Comprobar que se trata de CIF válido
     If validarCIF(CIF) Then
          _CIF = CIF
     Else
          Throw New System.ArgumentException("CIF Inválido")
     End If
    Me.Nombre = nombre
    Me.Contacto = contacto
    Me.Telefonos.Add(telefono)
    _vc = videoclub
    _vc.Distribuidoras.Add(Me)
End Sub
```

Los 8 dígitos compondrán un número que se divide entre 23 y nos quedamos con el resto de la división. El resto se corresponde con un índice que nos dará una letra de acuerdo al siguiente conjunto de caracteres:
TRWAGMYFPDXBNJZSQVHLCKE

Clases entidad
Dispose (I)

- Interface *IDisposable*:
  - **Método público: Sub Dispose()**
- Debe ocuparse de indicar a todos los objetos asociados con él que el objeto va a destruirse.
- Por último se elimina de la colección correspondiente de la clase raíz, para lo que llama al procedimiento *Delete* que será de ámbito *Friend.*

Clases entidad
Dispose (II)

```
`Clase Distribuidora
Public Sub Dispose() Implements IDisposable.Dispose
        Dim peli As Pelicula
        'Todas sus PeliculasDistribuidas ya no son distribuidas
        'por ella
        For Each peli In Me.PeliculasDistribuidas
            peli.SetDistribuidora(Nothing)
        Next
        `Borra todos los punteros a las películas distribuidas
        PeliculasDistribuidas.Clear
        _isDisposed = True
        If Not _vc Is Nothing Then
            _vc.Distribuidoras.Delete(Me)
        End If
    End Sub
```


Clase-colección para modelar un atributo con multiplicidad máxima > 1

- Si la multiplicidad máxima nunca va a ser más de 2 ó 3 se debería implementar como varios atributos simples y no como una clase-colección. En caso de que se tengan dudas de que en un futuro este número puede aumentar, sí que se implementaría mediante una *clase-colección*.
- Si la multiplicidad máxima no es *, pueden usarse constantes para controlar dicha multiplicidad (en el ejemplo, la multiplicidad de *telefono* para una *Distribuidora* es 0..4).

Clase-colección para modelar un atributo con multiplicidad máxima > 1. Constructor

- La clase implementa el interface *IEnumerable* como toda **clase-colección**.
- Si se desea acceder a través de la posición se implementa mediante un *ArrayList*.
- El constructor simplemente inicializa el *ArrayList* y debe ser *Friend* (sólo la clase entidad correspondiente podrá crearlo).

```
Public Class Telefonos
   Implements IEnumerable

   'Se utilizar un ArrayList en lugar de un Hashtable debido a que no hay
    identificador, sino que los teléfonos se identifican por su posición.
   Private _alTelefonos As ArrayList 'System.Collections.ArrayList
   Private Const TELEFONOS_MIN As Integer = 0
   Private Const TELEFONOS_MAX As Integer = 4

   Friend Sub New()
      _alTelefonos = New ArrayList(TELEFONOS_MAX)
   End Sub
```


Clase-colección para modelar un atributo con multiplicidad máxima > 1. Añadir elementos

- Debe controlar la multiplicidad máxima.
- Debe controlar que lo que se intenta insertar es correcto (no es *Nothing* y la longitud de la cadena es > 0).
- Si todo es correcto, se inserta el elemento en la colección.

```
Public Sub Add(ByVal telefono As String)
      If telefono Is Nothing Then
         Throw New System.ArgumentNullException
      ElseIf telefono.Trim.Length = 0 Then
         Throw New System.ArgumentException _
          ("El teléfono no puede ser una cadena vacía")
      ElseIf _alTelefonos.Count = TELEFONOS_MAX Then
          'Control de multiplidad máxima
         Throw New MultiplicidadInvalidaException("Número máximo = 4")
      Else
         _alTelefonos.Add(telefono)
      End If
End Sub
```

Clase-colección para modelar un atributo con multiplicidad máxima > 1. *Item, Count, Clear* y *GetEnumerator*

- Basados en *Item, Count* y *GetEnumerator* de la correspondiente colección (*HashTable* o *ArrayList*) utilizada como base.

```
Public ReadOnly Property Count() As Integer
      Get
         Return _alTelefonos.Count
      End Get
End Property

Public ReadOnly Property Item(ByVal posicion As Integer) As String
      Get
         Return CType(_alTelefonos(posicion), String)
      End Get
End Property

Public Sub Clear()
     _alTelefonos.Clear()
End Sub

Public Function GetEnumerator() As System.Collections.IEnumerator Implements
    System.Collections.IEnumerable.GetEnumerator
      Return _alTelefonos.GetEnumerator
End Function
```


Clase-colección para modelar un atributo con multiplicidad máxima > 1. Borrado de elementos

- Se debe implementar un método *Remove* que pida por parámetro el elemento a eliminar.
- Si se implementó mediante un *ArrayList* también debe implementarse un *RemoveAt* que borra el elemento de una posición dada.
- Ambos borrados se basan en los procedimientos de borrado propios de la colección usada como base *ArrayList* o *Hashtable*, pero controlando la multiplicidad mínima.

```
Public Sub RemoveAt(ByVal posicion As Integer)
      If _alTelefonos.Count = TELEFONOS_MIN Then
         'Control de multiplicidad mínima
         Throw New MultiplicidadInvalidaException("No se pueden borrar más elementos de la lista")
      End If
      _alTelefonos.RemoveAt(posicion)
   End Sub

   Public Sub Remove(ByVal telefono As String)
      If _alTelefonos.Count = TELEFONOS_MIN Then
         'Control de multiplicidad mínima
         Throw New MultiplicidadInvalidaException("No se pueden borrar más elementos de la lista")
      End If
      _alTelefonos.Remove(telefono)
End Sub
```


Clase-colección para ligar una entidad del negocio a la clase raíz (I)

- Por cada una de las clases entidad se debe crear una clase colección que le vincule a la clase raíz.
- Esta clase debe tener una variable miembro que referencia a la clase raíz (en el caso del ejemplo, del tipo *Videoclub*).
- Debe tener otro miembro, bien sea de tipo ***ArrayList*** (para colecciones ordenadas) o ***Hashtable***.
- Esta clase debe implementar el interface ***IEnumerable***.

Clase-colección para ligar una entidad del negocio a la clase raíz (II)

```
Public Class Distribuidoras
    Implements IEnumerable
   Private _vc As Videoclub
   Private _htDistribuidoras As Hashtable
   ...
End Class
```

Clase-colección para ligar una entidad del negocio a la clase raíz. Constructor

- Necesita como parámetro la referencia a la clase raíz, que se asigna a su correspondiente variable miembro.
- Debe ser con ámbito ***Friend*** porque sólo la clase raíz los puede crear.

```
Public Class Distribuidoras
   Implements IEnumerable

   Private _vc As Videoclub
   Private _htDistribuidoras As Hashtable

   Friend Sub New(ByVal videoclub As Videoclub)
         _vc = videoclub
         _htDistribuidoras = New Hashtable
    End Sub
    ...
End Class
```


Clase-colección para ligar una entidad del negocio a la clase raíz. Add (I)

- Se creará al menos un método ***Friend*** invocado desde el constructor de la correspondiente clase entidad.
- El método pide por parámetro una referencia a un objeto de la clase entidad correspondiente.

```
Friend Sub Add(ByVal distribuidora As Distribuidora)
   If _htDistribuidoras.Contains(distribuidora.CIF) Then
      Throw New System.Exception _
                  ("Distribuidora ya existente")
   Else
      _htDistribuidoras.Add(distribuidora.CIF, distribuidora)
   End If
End Sub
```


Clase-colección para ligar una entidad del negocio a la clase raíz. Add (II)

- Se puede crear otro ***Add*** público y opcional que pida los mismos parámetros que el constructor de su clase entidad.
  - Este método debe llamar al constructor de su clase entidad.
  - Usa un parámetro más que es la referencia al objeto-raíz (variable miembro _vc).

```
Public Function Add (ByVal CIF As String, ByVal nombre As _
      String, ByVal contacto as String, telefono As String) _
      As Distribuidora
      Return New Distribuidora(CIF, nombre, _
            contacto, telefono, _vc)
End Function
```


Clase-colección para ligar una entidad del negocio a la clase raíz. Item y Count (I)

- Se han de ofrecer 2 propiedades *ReadOnly* denominadas ***Item*** y ***Count***.
- El contenido de estas propiedades se basa en ***Item*** y ***Count*** de la colección utilizada como base (***ArrayList*** o ***Hashtable***).
- ***Item*** tiene un parámetro que debe ser un entero si se ha usado un ***ArrayList***, o bien, el atributo identificativo de la clase (CIF, código de película...) si se ha utilizado una Hashtable. Devuelve un objeto del tipo de su clase entidad.

Clase-colección para ligar una entidad del negocio a la clase raíz. Item y Count (II)

```
Public ReadOnly Property Item(ByVal CIF As String) _
                                        As Distribuidora
    Get
        Return CType(_htDistribuidoras.Item(CIF), _
                        Distribuidora)
    End Get
End Property

Public ReadOnly Property Count() As Integer
    Get
        Return _htDistribuidoras.Count
    End Get
End Property
```


Clase-colección para ligar una entidad del negocio a la clase raíz. Borrado (I)

- Antes del borrado físico de un elemento (método ***Remove***) deben eliminarse todas sus referencias llamando al método ***Dispose*** de la clase entidad.
- El método ***Remove*** pide por parámetro una referencia a un objeto de la clase entidad a borrar.
- En caso de una colección no ordenada (es decir, ***Hashtable***) un segundo ***Remove*** (que puede ser opcional) pide por parámetro el atributo identificativo de la clase entidad (ej., CIF).
- En caso de una colección ordenada (es decir, ***ArrayList***) se implementa un ***RemoveAt*** que borra el elemento en la posición que se ha indicado como parámetro.
- Un último procedimiento, ***Delete***, con ámbito ***Friend***, es llamado desde el ***Dispose*** de la clase para efectuar el borrado físico de la colección: _htDistribuidoras.*Remove(distribuidora.CIF)*

Clase-colección para ligar una entidad del negocio a la clase raíz. Borrado (II)

```
Public Sub Remove(ByVal distribuidora As Distribuidora)
      distribuidora.Dispose()
End Sub

Public Sub Remove(ByVal CIF As String)
      If _htDistribuidoras.Contains(CIF) Then

   CType(_htDistribuidoras.Item(CIF),Distribuidora).dispose()
      End If
End Sub

Friend Sub Delete(ByVal distribuidora As Distribuidora)
      _htDistribuidoras.Remove(distribuidora.CIF)
End Sub
```


Clase-colección para ligar una entidad del negocio a la clase raíz. Clear

- Se encarga de eliminar todos los elementos de la colección.
- Utilizando ***Hashtable***:

```
Public Sub Clear()
  Dim en As IDictionaryEnumerator
  Dim cif As String
  en = _htDistribuidoras.GetEnumerator
  While en.MoveNext()
     cif = CType(en.Key, String)
     Me.Remove(cif)
  End While
 _htDistribuidoras.Clear()
End Sub
```

- Utilizando **ArrayList**:

```
Public Sub Clear()
  While _alDistribuidoras.Count > 0
     Me.RemoveAt(0)
  End While
End Sub
```

Clase-colección para ligar una entidad del negocio a la clase raíz. GetEnumerator

- Como toda clase-colección se debe implementar la función ***GetEnumerator***.
- Utilizando ***Hashtable***:

```
Public Function GetEnumerator() As _
        IEnumerator Implements IEnumerable.GetEnumerator
    Return _htDistribuidoras.Values.GetEnumerator
End Function
```

- Utilizando ***ArrayList***:

```
Public Function GetEnumerator() As _
        IEnumerator Implements IEnumerable.GetEnumerator
    Return _alDistribuidoras.GetEnumerator
End Function
```

Diseño Basado en Componentes

Construcción de componentes utilizando VB.NET (Parte I)

Ingeniería Informática
Universidad Carlos III de Madrid