Diseño Basado en Componentes

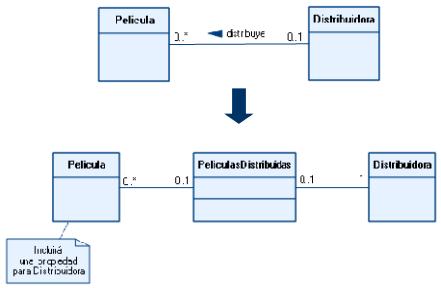
Implementación de Asociaciones en VB.NET

Ingeniería Informática
Universidad Carlos III de Madrid

Asociación 1 a * (I)

- Se implementa mediante una clase-colección para el lado *, y una propiedad en el lado 1.
- Cambiar algo en un extremo significa cambiar en el extremo contrario.

Asociación 1 a * (II)



Asociación 1 a *. Lado 1 (I)

- Se implementa mediante una simple variable miembro del tipo del extremo contrario (_distribuidora As *Distribuidora*).
- Se implementa una propiedad.
- Se añade un método *Friend* que permite sincronizar ambos extremos.
- El método *Friend* simplemente modifica el valor de la variable miembro.

```
'Clase Pelicula
Friend Sub SetDistribuidora(ByVal distribuidora As Distribuidora)
        _distribuidora = distribuidora
End Sub
```

Asociación 1 a *. Lado 1. Propiedad (I)

- El lado *Get* de la propiedad es como siempre.
- El lado *Set* es especial. En lugar de asignar simplemente el valor, hay que realizar 3 pasos:
  1. En caso de tener ya un valor (la distribuidora es distinta de *Nothing*), hay que decirle a la distribuidora actual que elimine la película de su colección. Para ello se invoca al procedimiento ***Delete***.
  2. Se asigna el nuevo valor: _distribuidora = Value
  3. Se le dice a la nueva distribuidora que añada el elemento a la colección (también en caso de ser distinta de *Nothing*). Para ello se invoca al procedimiento ***Append***.

Asociación 1 a *. Lado 1. Propiedad (II)

```
'Clase Pelicula
Public Property Distribuidora() As Distribuidora
  Get
    Return _distribuidora
  End Get
  Set(ByVal Value As Distribuidora)
    If Not _distribuidora Is Nothing Then
      _distribuidora.PeliculasDistribuidas.Delete(Me)
    End If
    _distribuidora = Value
    If Not _distribuidora Is Nothing Then
      _distribuidora.PeliculasDistribuidas.Append(Me)
    End If
  End Set
End Property
```


Asociación 1 a *. Lado *

- La clase-colección se implementa con una ***Hashtable*** y con una referencia a la clase entidad del extremo.

```
Public Class PeliculasDistribuidas
    Implements IEnumerable

  Private _distribuidora As Distribuidora
  Private _htPeliculasDistribuidas As Hashtable

  Friend Sub New(ByVal distribuidora As Distribuidora)
    _distribuidora = distribuidora
    _htPeliculasDistribuidas = New Hashtable
  End Sub
  ...
End Class
```


Asociación 1 a *. Lado *. Añadir (I)

- El método ***Add*** se encarga de añadir un elemento en la colección y de cambiar el valor del extremo opuesto.
- Para cambiar el extremo opuesto no se invoca directamente a la propiedad correspondiente (para evitar bucles infinitos), sino a un procedimiento ***Friend***.

Asociación 1 a *. Lado *. Añadir (II)

```
'Clase PeliculasDistribuidas
Public Sub Add(ByVal pelicula As Pelicula)
        If Not _htPeliculasDistribuidas.Contains(pelicula) Then
            _htPeliculasDistribuidas.Add(pelicula.Titulo, pelicula)
            pelicula.setDistribuidora(_distribuidora)
        Else
            Throw New System.Exception("Película ya registrada")
        End If
End Sub
Friend Sub Append(ByVal pelicula As Pelicula)
        _htPeliculasDistribuidas.Add(pelicula.Titulo, pelicula)
End Sub
```


Asociación 1 a *. Lado *. Borrar (I)

- También requiere al menos de 2 métodos.
- Uno público que borra en una colección y modifica el valor del extremo contrario.
- Y otro con ámbito ***Friend*** que simplemente elimina un valor de la colección.

Asociación 1 a *. Lado *. Borrar (II)

```
'Clase PeliculasDistribuidas
Public Sub Remove(ByVal pelicula As Pelicula)
        pelicula.SetDistribuidora(Nothing) 'Limpiar la referencia opuesta
        Me.Delete(pelicula) 'Eliminarlo de la colección
End Sub

Public Sub Remove(ByVal titulo As String)
        'Limpiar la referencia opuesta
        Item(titulo).SetDistribuidora(Nothing)
        Me.Delete(Item(titulo)) 'Eliminarlo de la colección
End Sub

Friend Sub Delete(ByVal pelicula As Pelicula)
        _htPeliculasDistribuidas.Remove(pelicula.Titulo)
End Sub
```


Asociación 1-*. Clear

- El método ***Clear*** es invocado en el método ***Dispose*** de la clase entidad (ej., clase Distribuidora).
- Se encarga de vaciar los elementos de la clase-colección.

```
'Clase PeliculasDistribuidas
Public Sub Clear()
    Dim en As IDictionaryEnumerator
    Dim titulo As String
    en = _htPeliculasDistribuidas.GetEnumerator
    While en.MoveNext()
        titulo = CType(en.Key, String)
        Me.Remove(titulo)
    End While
    _htPeliculasDistribuidas.Clear()
End Sub
```

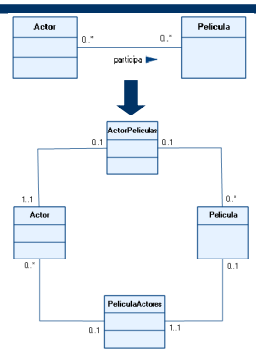
Asociación 1 a *. Lado *. Otros elementos

- Propiedades ***Item*** y ***Count***
- Función ***GetEnumerator***

Asociación *-* (I)

- Se implementa mediante dos clases-colección (una para cada extremo).
- Ambas deben estar siempre sincronizadas y su código es idéntico (cambiando el nombre de una de las clases entidad por la del otro extremo y viceversa).
- Como toda clase-colección implementa el interface ***IEnumerable***, tiene un miembro del tipo ***ArrayList*** o ***Hashtable***, y tiene otro segundo miembro que apunta al objeto al que se adscribe.

Asociación *-* (II)



Asociación *-* (III)

```
Public Class ActorPeliculas
    Implements IEnumerable

    Private _actor As Actor
    Private _htActorPeliculas As Hashtable
    ...
End class
```

Asociación *-*. Constructor

- Su constructor, como toda clase-colección, es ***Friend*** (sólo se invoca desde uno de los extremos de la asociación, es decir, desde una de las clases entidad) y pide por parámetro el objeto al que se adscribe la colección.

```
'Clase ActorPeliculas
Friend Sub New(ByVal actor As Actor)
        _actor = actor
        _htActorPeliculas = New Hashtable
    End Sub
```


Asociación *-*. Añadir (I)

- Al añadir a este extremo hay que añadir también al extremo opuesto para que ambos estén siempre sincronizados.
- Se implementa mediante un método público ***Add*** que realiza las dos inserciones (en ambos extremos).
- Si desde el ***Add*** de una de las colecciones se llama al ***Add*** de la otra colección (y viceversa) se podría entrar en bucle infinito (efecto *ping-pong*).
- Para evitarlo se implementa un segundo método ***Append*** (invocado por el ***Add***) y que simplemente introduce al elemento dentro de la colección.

Asociación *-*. Añadir (II)

```
'Clase ActorPeliculas
Public Sub Add(ByVal pelicula As Pelicula)
        If pelicula Is Nothing Then
            Throw New System.ArgumentNullException
        End If
        If _htActorPeliculas.Contains(pelicula) Then
            Throw New System.ArgumentException("Película ya registrada")
        Else
            Me.Append(pelicula)
            pelicula.Actores.Append(_actor)
        End If
    End Sub

Friend Sub Append(ByVal pelicula As Pelicula)
    If Not _htActorPeliculas.Contains(pelicula.Titulo) Then
            _htActorPeliculas.Add(pelicula.Titulo, pelicula)
        Else
            Throw New System.ArgumentException("Película ya registrada")
        End If
End Sub
```

Propiedad de la clase Pelicula de tipo colección PeliculaActores

Asociación *-*. Borrar (I)

- Al igual que al añadir, el borrado debe ser simétrico, si se borra de un extremo hay que borrar del extremo opuesto.
- Se implementará un método público (***Remove***) y otro con ámbito ***Friend*** (***Delete***)
- ***Remove*** hace el borrado en las dos colecciones invocando a los dos ***Delete***.
- ***Delete*** hace el borrado físico en las colecciones

Asociación *-*. Borrar (II)

```
'Clase ActorPeliculas
Public Sub Remove(ByVal pelicula As Pelicula)
        If pelicula Is Nothing Then
            Throw New System.ArgumentNullException
        End If
        If _htActorPeliculas.Contains(pelicula.Titulo) Then
            Me.Delete(pelicula)
            pelicula.Actores.Delete(_actor)
        Else
            Throw New System.ArgumentException _
                                    ("Película inexistente")
        End If
End Sub

Friend Sub Delete(ByVal pelicula As Pelicula)
        _htActorPeliculas.Remove(pelicula.Titulo)
End Sub
```


Asociación *-*. Clear

- El método ***Clear*** es invocado en el método ***Dispose*** de la clase entidad (ej., clase Actor).
- Se encarga de vaciar los elementos de la clase-colección.

```
'Clase ActorPeliculas
Public Sub Clear()
    Dim en As IDictionaryEnumerator
    Dim titulo As String
    en = _htActorPeliculas.GetEnumerator
    While en.MoveNext()
        titulo = CType(en.Key, String)
        Me.Remove(CType(_htActorPeliculas.Item(titulo), Pelicula))
    End While
    _htActorPeliculas.Clear()
End Sub
```


Asociación *-*. Otros elementos

- Propiedades ***Item*** y ***Count***
- Función ***GetEnumerator***

Diseño Basado en Componentes

Implementación de Asociaciones en VB.NET

Information Engineering

Ingeniería Informática
Universidad Carlos III de Madrid