Unidad 3

DIPLOMATURA EN PROGRAMACION JAVA

Capítulo 4

Relaciones en .Net

Relaciones en .Net

Vectores

Declaración de vectores

Los vectores en .Net, al igual que en cualquier lenguaje, son un grupo de variables del mismo tipo agrupadas por un único nombre.

En .Net, los vectores pueden contener entre sus elementos tipos primitivos o referencias a objetos.

Ejemplo

C#

```
char[] s;
Punto[] p;
```

VB

```
Dim s() As Integer
Dim p() As Punto
```

Sin embargo, en .Net, cuando se crea un vector, este no se aloja en memoria hasta crearlo con el operador `new` en C# y `New` en VB con lo cual es, hasta ese momento, la declaración de una referencia a un objeto de tipo vector. Por lo tanto, la declaración en si misma no crea otra cosa que un espacio en memoria para alojar una referencia y esto se debe a que los vectores son objetos para el CLR.

En .Net los vectores funcionan como una clase (esto es porque internamente lo son), de la cual se debe crear una instancia con el operador `new`, como cuando se crea cualquier objeto.

Creación de vectores

Como se mencionó anteriormente, se puede crear un vector con tipos primitivos. Para esto se debe indicar a continuación del operador `new` el tipo primitivo a utilizar y seguido de éste, entre corchetes para C# y paréntesis para VB, la cantidad de elementos que poseerá el vector.

Para acceder a cada elemento del vector se utiliza el nombre del mismo con un subíndice, el cual puede ser una variable o una constante, que indica el elemento del vector con el que se quiere operar.

Ejemplo

C#

```
public char[] crearVector()
{
        char[] s;
        s = new char[26];
        s[0] = (char)('A');
```

```
            s[1] = (char)('B');
            s[2] = (char)('C');
            s[3] = (char)('D');
            s[4] = (char)('E');
          :
          :
          s[20] = (char)('U');
          s[21] = (char)('V');
          s[22] = (char)('W');
          s[23] = (char)('X');
          s[24] = (char)('Y');
          s[25] = (char)('Z');
          return s;
      }
```

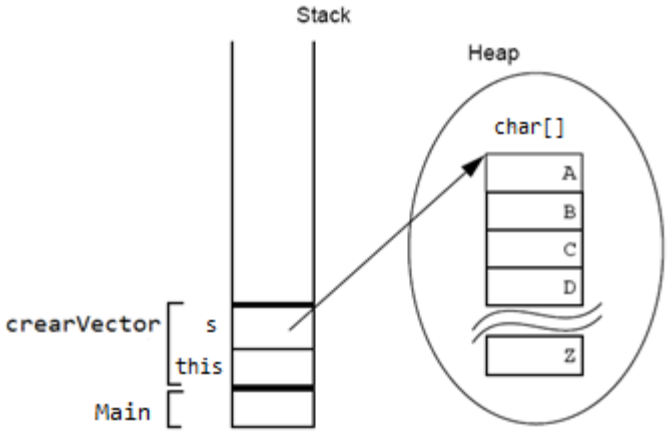
VB

```
    Function crearVector() As Char()
          Dim s As Char()
          s = New Char(26) {}
          s(0) = "A"
          s(1) = "B"
          s(2) = "C"
          s(3) = "D"
          s(4) = "E"
          :
          :
          s(20) = "U"
          s(21) = "V"
          s(22) = "W"
          s(23) = "X"
          s(24) = "Y"
          s(25) = "Z"
          Return s
    End Function
```

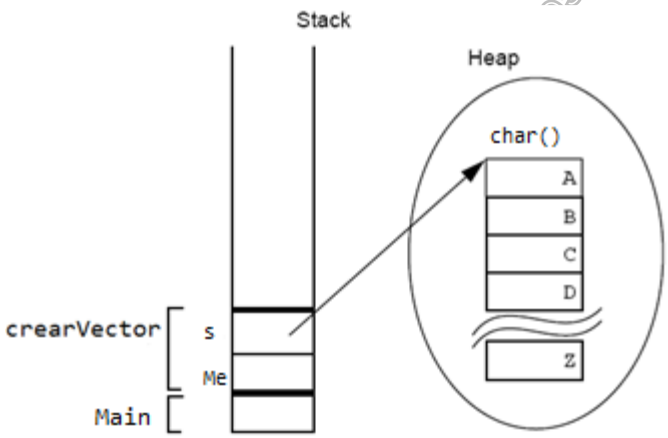

Notar que la referencia a un vector puede utilizarse como valor retornado por un método

La forma en que se aloja en memoria el vector lo muestra en las siguientes figuras

C#



VB



Notar que el contenido del vector se aloja en el heap, como los atributos de cualquier objeto.

También existen los vectores de objetos. Si bien el manejo es similar, la comprensión del concepto es un poco más confusa.

Cuando se crea un vector de objetos, al igual que un vector de tipos primitivos, se crea una referencia. Sin embargo, cuando se genera el espacio para almacenar los elementos del vector, cada uno de estos es a la vez "una referencia a objetos del tipo del cual fue declarado el vector".

Por lo tanto, por cada elemento del vector, se deberá crear un objeto cuya referencia se guardará en dicho elemento. El siguiente código muestra la operatoria de creación del vector. Para crear cada elemento se utilizará una clase de llamada Punto que recibe dos parámetros en el constructor.

C#

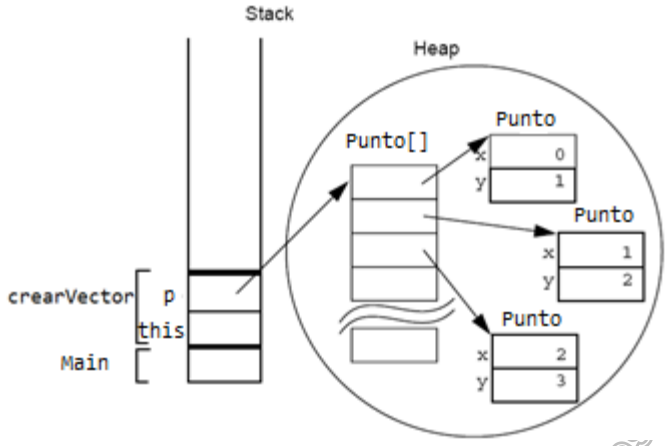
```
        public Punto[] crearVector()
        {
            Punto[] p;
            p = new Punto[26];
            p[0] = new Punto(0, 1);
            p[1] = new Punto(1, 2);
            p[2] = new Punto(2, 3);
            p[3] = new Punto(3, 4);
            p[4] = new Punto(4, 5);
            p[5] = new Punto(5, 6);
            p[6] = new Punto(6, 7);
            p[7] = new Punto(7, 8);
            p[8] = new Punto(8, 9);
            p[9] = new Punto(9, 10);
            p[10] = new Punto(10, 11);

            return p;
        }
```

VB

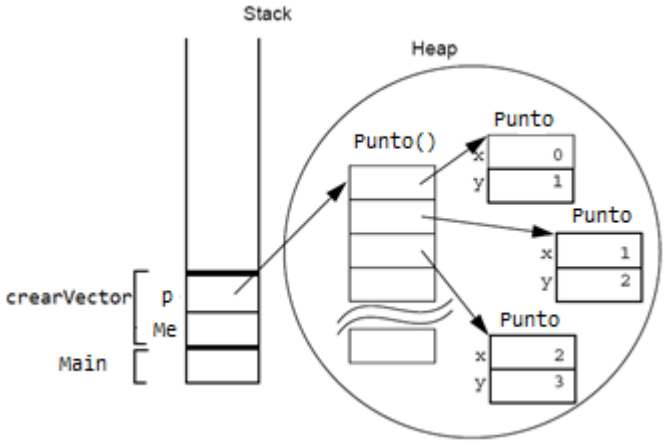
```
    Function crearVector() As Punto()
        Dim p As Punto()
        p = New Punto(26) {}
        p(0) = New Punto(0, 1)
        p(1) = New Punto(1, 2)
        p(2) = New Punto(2, 3)
        p(3) = New Punto(3, 4)
        p(4) = New Punto(4, 5)
        p(5) = New Punto(5, 6)
        p(6) = New Punto(6, 7)
        p(7) = New Punto(7, 8)
        p(8) = New Punto(8, 9)
        p(9) = New Punto(9, 10)
        p(10) = New Punto(10, 11)

        Return p
    End Function
```

Los siguientes gráficos ilustran los lugares de almacenamiento para este vector de objetos

C#



VB



Maneras de inicializar vectores

A los vectores, tanto de objetos como de tipos primitivos, se le puede asignar valores elemento a elemento o en bloque. Cuando es en bloque, se lo denomina “bloque de inicialización”, se encierra entre llaves los elementos que determinan el valor inicial de cada elemento del vector y el tamaño de este se ajusta automáticamente a la cantidad de elementos declarados en el bloque.

Se debe tener en cuenta que cuando se inicializan vectores de objetos se lo hace mediante su constructor **siempre**.

Como ejemplo de asignación de valores iniciales elemento a elemento, se muestra el siguiente código.

Ejemplo

C#

```
String[] nombres;
```

```
nombres = new String[3];
nombres[0] = "Silvina";
nombres[1] = "Silvana";
nombres[2] = "Silvia";

Punto[] puntos;
puntos = new Punto[3];
puntos[0] = new Punto(22, 7);
puntos[1] = new Punto(1, 1);
puntos[2] = new Punto(22, 12);
```

VB

```
Dim nombres As String()
nombres = New String(3) {}
nombres(0) = "Silvina"
nombres(1) = "Silvana"
nombres(2) = "Silvia"

Dim puntos() As Punto
puntos = New Punto(3) {}
puntos(0) = New Punto(22, 7)
puntos(1) = New Punto(1, 1)
puntos(2) = New Punto(22, 12)
```

elEl siguiente código se muestra como ejemplo de declaración con un bloque de asignación.

Ejemplo

C#

```
String[] nombres2 = {
                    "Alejo",
                    "Alexis",
                    "Alejandro"
                    };

Punto[] puntos2 = {
                    new Punto(2, 5),
                    new Punto(12, 2),
                    new Punto(13, 8)
                    };
```

VB

```
Dim nombres2() As String = {"Alejo", "Alexis", "Alejandro"}

Dim puntos2() As Punto = { _
New Punto(2, 5), _
New Punto(12, 2), _
New Punto(13, 8) _
}
```

Asociaciones y enlaces

Para definir una relación en .Net se puede declarar un objeto dentro de una clase, pasar un objeto como parámetro a un método, devolver un objeto desde un método o declararlo como un elemento de un vector de objetos.

Salvo un tipo de relación un poco más compleja, la herencia, el resto se explicará en este módulo.

Asociaciones simples

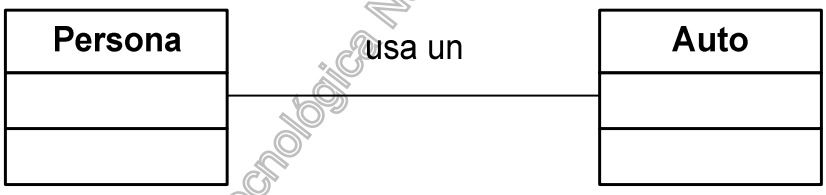
Están definidas cada vez que dos clases interaccionan entre si. Este tipo de asociaciones es la más débil de todas y por lo general se representa en el código como una variable auxiliar, un parámetro de una función o cualquier tipo de referencia que una tenga sobre la otra de manera que no cree mayor dependencia entre ellas más que por un lapso pequeño de tiempo.

La frase de verificación para una asociación simple es “usa un” o “usa una” y la relación siempre es temporal e independiente de la vida de la utiliza el servicio de la otra.

Por ejemplo, se puede suponer el siguiente problema de diseño:

Una persona usa un auto de la compañía cuando lo requiere para visitar clientes

La frase de verificación indica claramente una asociación simple entre la clase Persona y la clase Auto. El diagrama que se puede generar a partir del enunciado es el siguiente



En UML cuando se omite la multiplicidad se debe interpretar la relación de izquierda a derecha y uno a uno.

El código que se expone a continuación resuelve el problema de diseño

Ejemplo

Clase Auto

C#

```
class Auto
{
	private String marca;

	public String Marca
	{
		get { return marca; }
```

```
            set { marca = value; }
        }
        private String modelo;

         public String Modelo
        {
            get { return modelo; }
            set { modelo = value; }
        }
        private String anio;

        public String Anio
        {
            get { return anio; }
            set { anio = value; }
        }


        public Auto(String ma, String mo, String an)
        {
            marca = ma;
            modelo = mo;
            anio = an;
        }
}
```

VB

```
Public Class Auto
    Private _marca As String
    Private _modelo As String
    Private _anio As String

    Public Property Marca() As String
        Get
            Marca = _marca
        End Get
        Set(ByVal value As String)
            _marca = value
        End Set
    End Property

    Public Property Modelo() As String
        Get
            Modelo = _modelo
        End Get
        Set(ByVal value As String)
            _modelo = value
        End Set
    End Property

    Public Property Anio() As String
        Get
            Anio = _anio
        End Get
```

```
        Set(ByVal value As String)
            _anio = value
        End Set
    End Property

    Public Sub New(ByVal ma As String, ByVal mo As String, ByVal an As String)
        Marca = ma
        Modelo = mo
        Anio = an
    End Sub

End Class
```

Clase Persona

C#

```
    class Persona
    {
        private String primerNombre;

        public String PrimerNombre
        {
            get { return primerNombre; }
            set { primerNombre = value; }
        }
        private String segundoNombre;

        public String SegundoNombre
        {
            get { return segundoNombre; }
            set { segundoNombre = value; }
        }
        private String apellido;

        public String Apellido
        {
            get { return apellido; }
            set { apellido = value; }
        }
        private String documento;

        public String Documento
        {
            get { return documento; }
            set { documento = value; }
        }

        public Persona(String p,
            String s,
            String a,
            String d)
        {
            primerNombre = p;
            segundoNombre = s;
```

```
            apellido = a;
            documento = d;
        }

        public void usaAuto(Auto a)
        {
            Console.WriteLine("La persona " + primerNombre + " esta usando un " +
                              a.Marca + " modelo " + a.Modelo + " del año " + a.Anio);
        }

    }
```

VB

```
Public Class Persona
    Private _primerNombre As String
    Private _segundoNombre As String
    Private _apellido As String
    Private _documento As String
    Private _detalles As String


    Public Property PrimerNombre() As String
        Get
            PrimerNombre = _primerNombre
        End Get
        Set(ByVal value As String)
            _primerNombre = value
        End Set
    End Property

    Public Property SegundoNombre() As String
        Get
            SegundoNombre = _segundoNombre
        End Get
        Set(ByVal value As String)
            _segundoNombre = value
        End Set
    End Property

    Public Property Apellido() As String
        Get
            Apellido = _apellido
        End Get
        Set(ByVal value As String)
            _apellido = value
        End Set
    End Property

    Public Property Documento() As String
        Get
            Documento = _documento
        End Get
        Set(ByVal value As String)
            _documento = value
        End Set
```

```
    End Property

    Public Property Detalles() As String
        Get
            Detalles = _detalles
        End Get
        Set(ByVal value As String)
            _detalles = value
        End Set
    End Property

   Public Sub New(ByVal p As String, ByVal s As String, ByVal a As String, ByVal d _
                  As String)
        _primerNombre = p
        _segundoNombre = s
        _apellido = a
        _documento = d
    End Sub

    Public Sub usaAuto(ByVal a As Auto)
        Console.WriteLine("La persona esta usando un " + a.Marca _
         + " modelo " + a.Modelo + " del año " + a.Anio)
    End Sub

End Class
```

Clase UsaAsociacion

C#

```
    class UsaAsociacion
    {
        static void Main(string[] args)
        {
            Auto a = new Auto("Fiat", "Uno", "1998");
            Persona p = new Persona("Juan", "Ignacio", "Perez", "21111222");

            p.usaAuto(a);

            Console.ReadKey();

        }
    }
```

VB

```
Module UsaAsociacion

    Sub Main()
        Dim p As Persona = New Persona("Juan", "Ignacio", "Perez", "21111222")

        Dim a As Auto = New Auto("Fiat", "Uno", "1998")

        p.usaAuto(a)
```

```
        Console.ReadKey()

    End Sub

End Module
```

Asociaciones y multiplicidad

En el presente curso, por razones de simplicidad en la codificación, las multiplicidades superiores a 1 se representarán como vectores y se interaccionará con ellos elemento a elemento, sin recorrerlos.

La razón de esto es focalizar en los conceptos de la programación orientada a objetos sin entrar en los detalles de codificación propios de .Net, lo cuales se verán en otro curso.

Ejercicio 1

Los temas de los que se tratan en este ejercicio son los siguientes:

- Asociaciones
- Multiplicidad

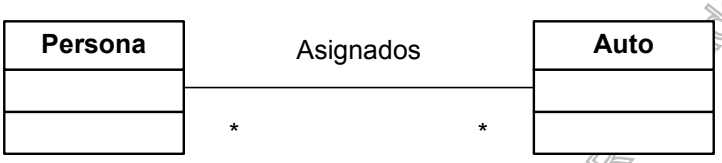
Basándose en el diseño realizado en el módulo anterior, codificar en .Net las asociaciones con su respectiva multiplicidad

Asociaciones complejas

Existen muchas situaciones en las cuales se crean relaciones de muchos a muchos en el diseño de clases. Por ejemplo, se puede suponer el siguiente problema a resolver:

> *Una persona puede tener asignados para su uso varios autos de la compañía. Los mismos son compartidos por varias personas, por lo tanto, un auto puede estar asignado a varias personas a la vez.*

El diagrama UML que representaría este enunciado es el siguiente:



Existen distintas técnicas y herramientas para resolver este tipo de relaciones, tanto de diseño como de codificación.

Por simplificación y para minimizar la cantidad de lenguaje necesario para aprender estos conceptos, el lado "muchos" de la relación se manejará con un vector cuando fuese necesario a lo largo de todas las explicaciones. Otra posibilidad son las colecciones, pero es un tema fuera del alcance de este curso.

En cuanto al diseño, las posibilidades son dos:

- Clases de asociación
- Asociaciones calificadas

Clase de asociación

Es evidente que el problema plantea la necesidad de utilizar alguna técnica de diseño o codificación para resolver la relación entre clases.

Una posible solución es:

> *Crear una clase que maneje por un lado la relación de una a muchos entre cada persona y sus posibles autos asignados y por el otro las personas asignadas a cada auto*

El siguiente código plantea una de las posibles soluciones

Ejemplo

Clase Asignados

C#

```
class Asignados
{
	private Persona persona=null;

	public Persona Persona
	{
		get { return persona; }
		set { persona = value; }
	}

	private Persona[] personas;

	public Persona[] Personas
	{
		get
		{
			if (personas.Length == 0) return null;
			else return personas;
		}
		set { personas = value; }
	}

	private Auto auto=null;

	public Auto Auto
	{
		get { return auto; }
		set { auto = value; }
	}

	private Auto[] autos;

	public Auto[] Autos
	{
		get {
			if (autos.Length == 0) return null;
			else return autos;
		}
	     set { autos = value; }
	}

	public Asignados(Persona p, Auto[] autos)
	{
		persona = p;
	     this.autos = autos;
	}

	public Asignados(Auto a, Persona[] personas)
	{
```

```
            auto = a;
            this.personas = personas;
        }
}
```

VB

```
Public Class Asignados
    Private _persona As Persona = Nothing
    Private _personas() As Persona
    Private _auto As Auto = Nothing
    Private _autos() As Auto

    Public Property Personas() As Persona()
        Get
            Personas = _personas
        End Get
        Set(ByVal value As Persona())
            _personas = value
        End Set
    End Property

    Public Property Persona() As Persona
        Get
            Persona = _persona
        End Get
        Set(ByVal value As Persona)
            _persona = value
        End Set
    End Property

    Public Property Autos() As Auto()
        Get
            Autos = _autos
        End Get
        Set(ByVal value As Auto())
            _autos = value
        End Set
    End Property

    Public Property Auto() As Auto
        Get
            Auto = _auto
        End Get
        Set(ByVal value As Auto)
            _auto = value
        End Set
    End Property

    Public Sub New(ByVal p As Persona, ByVal autos() As Auto)
        _persona = p
        Me._autos = autos
    End Sub

    Public Sub New(ByVal a As Auto, ByVal personas() As Persona)
        Me._auto = a
```

```
        Me._personas = personas
    End Sub

End Class
```

Clase Auto

C#

```
class Auto
{
       private String marca;

       public String Marca
       {
              get { return marca; }
              set { marca = value; }
       }
       private String modelo;

       public String Modelo
       {
              get { return modelo; }
              set { modelo = value; }
       }
       private String anio;

       public String Anio
       {
              get { return anio; }
              set { anio = value; }
       }

       private Persona persona;

       public Persona Persona
       {
              get { return persona; }
              set { persona = value; }
       }

       public Auto(String ma, String mo, String an)
       {
              marca = ma;
              modelo = mo;
              anio = an;
       }
}
```

VB

```
Public Class Auto
    Private _marca As String
    Private _modelo As String
    Private _anio As String
```

```
    Public Property Marca() As String
        Get
            Marca = _marca
        End Get
        Set(ByVal value As String)
            _marca = value
        End Set
    End Property

    Public Property Modelo() As String
        Get
            Modelo = _modelo
        End Get
        Set(ByVal value As String)
            _modelo = value
        End Set
    End Property

    Public Property Anio() As String
        Get
            Anio = _anio
        End Get
        Set(ByVal value As String)
            _anio = value
        End Set
    End Property

    Public Sub New(ByVal ma As String, ByVal mo As String, ByVal an As String)
        Marca = ma
        Modelo = mo
        Anio = an
    End Sub

End Class
```

Clase Persona

C#

```
class Persona
{
       private String primerNombre;

       public String PrimerNombre
       {
              get { return primerNombre; }
              set { primerNombre = value; }
       }
       private String segundoNombre;

       public String SegundoNombre
       {
              get { return segundoNombre; }
              set { segundoNombre = value; }
       }
```

```
        private String apellido;

        public String Apellido
        {
                get { return apellido; }
                set { apellido = value; }
        }
        private String documento;

        public String Documento
        {
                get { return documento; }
                set { documento = value; }
        }

        public Persona(String p,
                String s,
                String a,
                String d)
        {
                primerNombre = p;
                segundoNombre = s;
                apellido = a;
                documento = d;
        }
}
```

VB

```
Public Class Persona
    Private _primerNombre As String
    Private _segundoNombre As String
    Private _apellido As String
    Private _documento As String

    Public Property PrimerNombre() As String
        Get
            PrimerNombre = _primerNombre
        End Get
        Set(ByVal value As String)
            _primerNombre = value
        End Set
    End Property

    Public Property SegundoNombre() As String
        Get
            SegundoNombre = _segundoNombre
        End Get
        Set(ByVal value As String)
            _segundoNombre = value
        End Set
    End Property

    Public Property Apellido() As String
        Get
            Apellido = _apellido
```

```
        End Get
        Set(ByVal value As String)
            _apellido = value
        End Set
    End Property

    Public Property Documento() As String
        Get
            Documento = _documento
        End Get
        Set(ByVal value As String)
            _documento = value
        End Set
    End Property

    Public Sub New(ByVal p As String, ByVal s As String, ByVal a As String, _
                   ByVal d As String)
        _primerNombre = p
        _segundoNombre = s
        _apellido = a
        _documento = d
    End Sub

End Class
```

Clase UsaClaseAsociacion

C#

```
class UsaClaseAsociacion
{
       static void Main(string[] args)
       {
              int i = 0;
              int j = 0;

              Persona[] vecPerA1 = new Persona[1];
              Persona[] vecPerA2 = new Persona[3];
              Persona[] vecPerA3 = new Persona[2];
              Persona[] vecPerA4 = new Persona[2];
              Persona[] vecPerA5 = new Persona[2];

              Persona p1 = new Persona("Juan", "Ignacio", "Perez", "21111222");
              Persona p2 = new Persona("Pedro", "Alberto", "Almeida", "23222111");
              Persona p3 = new Persona("Carlos", "Alejo", "Garcia", "21444555");
              Persona p4 = new Persona("Sergio", "Ricardo", "Burr", "21666777");

              vecPerA1[0] = p1;

              vecPerA2[0] = p1;
              vecPerA2[1] = p3;
              vecPerA2[2] = p4;

              vecPerA3[0] = p1;
              vecPerA3[1] = p2;
```

```
vecPerA4[0] = p2;
vecPerA4[1] = p3;

vecPerA5[0] = p2;
vecPerA5[1] = p4;


Auto[] vecAuP1 = new Auto[3];
Auto[] vecAuP2 = new Auto[3];
Auto[] vecAuP3 = new Auto[2];
Auto[] vecAuP4 = new Auto[2];

Auto a1 = new Auto("Fiat", "Uno", "1996");
Auto a2 = new Auto("Fiat", "Dos", "1997");
Auto a3 = new Auto("Fiat", "Tres", "1998");
Auto a4 = new Auto("Fiat", "Cuatro", "1999");
Auto a5 = new Auto("Fiat", "Cuatro", "2000");

vecAuP1[0] = a1;
vecAuP1[1] = a2;
vecAuP1[2] = a3;

vecAuP2[0] = a3;
vecAuP2[1] = a4;
vecAuP2[2] = a5;

vecAuP3[0] = a2;
vecAuP3[1] = a4;

vecAuP4[0] = a2;
vecAuP4[1] = a5;


Asignados[] autosPersona = new Asignados[4];

autosPersona[0] = new Asignados(p1, vecAuP1);
autosPersona[1] = new Asignados(p2, vecAuP2);
autosPersona[2] = new Asignados(p3, vecAuP3);
autosPersona[3] = new Asignados(p4, vecAuP4);

Asignados[] personasAuto = new Asignados[5];

personasAuto[0] = new Asignados(a1, vecPerA1);
personasAuto[1] = new Asignados(a2, vecPerA2);
personasAuto[2] = new Asignados(a3, vecPerA3);
personasAuto[3] = new Asignados(a4, vecPerA4);
personasAuto[4] = new Asignados(a5, vecPerA5);

for (i = 0; i < autosPersona.Length; i++)
{
       Console.WriteLine("Autos Asignados a la Persona " );
       Console.WriteLine("Apellido: " +
              autosPersona[i].Persona.Apellido);
```

```
                    Console.WriteLine("Primer Nombre: " +
                           autosPersona[i].Persona.PrimerNombre);
                    Console.WriteLine("Segundo Nombre: " +
                           autosPersona[i].Persona.SegundoNombre);
                    Console.WriteLine("Documento: " +
                           autosPersona[i].Persona.Documento);

                    Auto[] autos = autosPersona[i].Autos;

                    for (j = 0; j < autos.Length; j++)
                    {
                           Console.WriteLine("=================================");
                           Console.WriteLine("Marca: " + autos[j].Marca
                                  + " Modelo: " + autos[j].Modelo
                                  + " Año: " + autos[j].Anio);
                           Console.WriteLine("=================================");
                    }
             Console.WriteLine();
             }

             Console.ReadKey();
             Console.WriteLine();

             for (i = 0; i < personasAuto.Length; i++)
             {
                    Console.WriteLine("Marca: " + personasAuto[i].Auto.Marca
                           + " Modelo: " + personasAuto[i].Auto.Modelo
                           + " Año: " + personasAuto[i].Auto.Anio);

                    Persona[] personas = personasAuto[i].Personas;

                    for (j = 0; j < personas.Length; j++)
                    {
                           Console.WriteLine("=================================");
                           Console.WriteLine("Persona Asignados al Auto ");
                           Console.WriteLine("Apellido: " + personas[j].Apellido);
                           Console.WriteLine("Primer Nombre: " +
                                         personas[j].PrimerNombre);
                           Console.WriteLine("Segundo Nombre: " +
                                         personas[j].SegundoNombre);
                           Console.WriteLine("Documento: " + personas[j].Documento);
                           Console.WriteLine("=================================");
                    }
                    Console.WriteLine();
             }
             Console.ReadKey();
      }
}
```

VB

```
Sub Main()
      Dim i As Integer = 0
      Dim j As Integer = 0

      Dim vecPerA1(1) As Persona
```

```
Dim vecPerA2(3) As Persona
Dim vecPerA3(2) As Persona
Dim vecPerA4(2) As Persona
Dim vecPerA5(2) As Persona

Dim p1 As Persona = New Persona("Juan", "Ignacio", "Perez", "21111222")
Dim p2 As Persona = New Persona("Pedro", "Alberto", "Almeida", "23222111")
Dim p3 As Persona = New Persona("Carlos", "Alejo", "Garcia", "21444555")
Dim p4 As Persona = New Persona("Sergio", "Ricardo", "Burr", "21666777")

vecPerA1(0) = p1

vecPerA2(0) = p1
vecPerA2(1) = p3
vecPerA2(2) = p4

vecPerA3(0) = p1
vecPerA3(1) = p2

vecPerA4(0) = p2
vecPerA4(1) = p3

vecPerA5(0) = p2
vecPerA5(1) = p4


Dim vecAuP1(3) As Auto
Dim vecAuP2(3) As Auto
Dim vecAuP3(2) As Auto
Dim vecAuP4(2) As Auto

Dim a1 As Auto = New Auto("Fiat", "Uno", "1996")
Dim a2 As Auto = New Auto("Fiat", "Dos", "1997")
Dim a3 As Auto = New Auto("Fiat", "Tres", "1998")
Dim a4 As Auto = New Auto("Fiat", "Cuatro", "1999")
Dim a5 As Auto = New Auto("Fiat", "Cuatro", "2000")

vecAuP1(0) = a1
vecAuP1(1) = a2
vecAuP1(2) = a3

vecAuP2(0) = a3
vecAuP2(1) = a4
vecAuP2(2) = a5

vecAuP3(0) = a2
vecAuP3(1) = a4

vecAuP4(0) = a2
vecAuP4(1) = a5


Dim autosPersona(4) As Asignados

autosPersona(0) = New Asignados(p1, vecAuP1)
```

```
autosPersona(1) = New Asignados(p2, vecAuP2)
autosPersona(2) = New Asignados(p3, vecAuP3)
autosPersona(3) = New Asignados(p4, vecAuP4)

Dim personasAuto(5) As Asignados
personasAuto(0) = New Asignados(a1, vecPerA1)
personasAuto(1) = New Asignados(a2, vecPerA2)
personasAuto(2) = New Asignados(a3, vecPerA3)
personasAuto(3) = New Asignados(a4, vecPerA4)
personasAuto(4) = New Asignados(a5, vecPerA5)

For i = 0 To autosPersona.Length - 2
       Console.WriteLine("Autos Asignados a la Persona ")
       Console.WriteLine("Apellido: " + autosPersona(i).Persona.Apellido)
       Console.WriteLine("Primer Nombre: " + _
              autosPersona(i).Persona.PrimerNombre)
       Console.WriteLine("Segundo Nombre: " + _
              autosPersona(i).Persona.SegundoNombre)
       Console.WriteLine("Documento: " + autosPersona(i).Persona.Documento)

       Dim autos() As Auto = autosPersona(i).Autos

       For j = 0 To autos.Length - 2
              Console.WriteLine("======================================")
              Console.WriteLine("Marca: " + autos(j).Marca _
                     + " Modelo: " + autos(j).Modelo _
                     + " Año: " + autos(j).Anio)
              Console.WriteLine("======================================")
       Next
       Console.WriteLine()
Next

Console.ReadKey()
Console.WriteLine()

For i = 0 To personasAuto.Length - 2
       Console.WriteLine("Marca: " + personasAuto(i).Auto.Marca _
       + " Modelo: " + personasAuto(i).Auto.Modelo _
       + " Año: " + personasAuto(i).Auto.Anio)

       Dim personas() As Persona = personasAuto(i).Personas

       For j = 0 To personas.Length - 2
              Console.WriteLine("======================================")
              Console.WriteLine("Persona Asignados al Auto ")
              Console.WriteLine("Apellido: " + personas(j).Apellido)
              Console.WriteLine("Primer Nombre: " + personas(j).PrimerNombre)
              Console.WriteLine("Segundo Nombre: " + _
                     personas(j).SegundoNombre)
              Console.WriteLine("Documento: " + personas(j).Documento)
              Console.WriteLine("======================================")
       Next
       Console.WriteLine()
Next
Console.ReadKey()
```

```
End Sub
```

Asociación calificada

Es otra técnica para resolver asociaciones complejas

Se realiza mediante la declaración de un atributo que maneje la complejidad de la relación. El atributo deberá almacenar o tener la capacidad de referenciar los elementos individuales de la relación compleja.

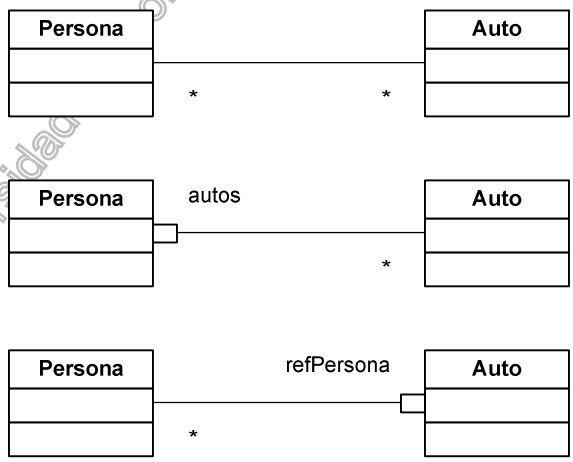
Existen diferentes técnicas para lograr este objetivo. Como en el curso se utiliza un vector para simplificar el código, la clase que lo implementa debe generar un servicio que permita interactuar con los elementos que éste almacena.

Utilizando un ejemplo similar al de la clase de asociación para que se puedan comparar ambas técnicas, se realiza el siguiente enunciado de problema

> *Crear una clase que maneje por un lado la relación de una a muchos entre cada persona y sus posibles autos asignados y en otra clase las personas asignadas a cada auto*

Por la limitación impuesta que el manejo de la relación se encuentre para el caso de "muchos" en clases diferentes, la solución evidente es la de la asociación calificada. Las candidatas para manejar este extremo de la relación en cada caso son las mismas clases que intervienen en ella y no una diferente y aparte como en el caso de la clase de asociación.

El siguiente gráfico muestra el diseño resultante



El diseño deriva en el siguiente código. Por simplicidad, se limito el tamaño del vector para no manejar la complejidad de un posible cambio de tamaño ya que .Net exige que para cambiar el

tamaño de un vector, se cree uno nuevo con más elementos y se copie los que estaban almacenados en el vector original. Para peor, esto se considera una mala práctica de programación y es uno de los argumentos de la creación de colecciones

Ejemplo

Clase Auto

C#

```
class Auto
{
	private String marca;

	public String Marca
	{
		get { return marca; }
		set { marca = value; }
	}
	private String modelo;

	public String Modelo
	{
		get { return modelo; }
		set { modelo = value; }
	}
	private String anio;

	public String Anio
	{
		get { return anio; }
		set { anio = value; }
	}

	private Persona[] refPersona = new Persona[10];

	public Persona[] RefPersona
	{
		get { return refPersona; }
		set { refPersona = value; }
	}

	private int i;

	public Auto(String ma, String mo, String an)
	{
		marca = ma;
		modelo = mo;
		anio = an;
	}

	public void setPersona(Persona persona)
	{
		refPersona[i] = persona;
```

```
            i = i + 1;
      }
      public Persona getPersona(int p)
      {
            return refPersona[p];
      }
}
```

VB

```
Public Class Auto
    Private _marca As String
    Private _modelo As String
    Private _anio As String
    Private _personas(10) As Persona

    Public Property Personas() As Persona()
        Get
            Personas = _personas
        End Get
        Set(ByVal value As Persona())
            _personas = value
        End Set
    End Property

    Public Property Marca() As String
        Get
            Marca = _marca
        End Get
        Set(ByVal value As String)
            _marca = value
        End Set
    End Property

    Public Property Modelo() As String
        Get
            Modelo = _modelo
        End Get
        Set(ByVal value As String)
            _modelo = value
        End Set
    End Property

    Public Property Anio() As String
        Get
            Anio = _anio
        End Get
        Set(ByVal value As String)
            _anio = value
        End Set
    End Property

    Public Sub New(ByVal ma As String, ByVal mo As String, ByVal an As String)
        Marca = ma
        Modelo = mo
        Anio = an
```

```
    End Sub

End Class
```

Clase Persona

C#

```
class Persona
{
	private String primerNombre;

	public String PrimerNombre
	{
		get { return primerNombre; }
		set { primerNombre = value; }
	}
	private String segundoNombre;

	public String SegundoNombre
	{
		get { return segundoNombre; }
		set { segundoNombre = value; }
	}
	private String apellido;

	public String Apellido
	{
		get { return apellido; }
		set { apellido = value; }
	}
	private String documento;

	public String Documento
	{
		get { return documento; }
		set { documento = value; }
	}

	private Auto[] autos;

	public Auto[] Autos
	{
		get { return autos; }
		set { autos = value; }
	}

	private int i;

	public Persona(String p,
		String s,
		String a,
		String d)
	{
		primerNombre = p;
```

```
                segundoNombre = s;
                apellido = a;
                documento = d;
                i = 0;
                autos = new Auto[10];
        }

        public void setAuto(Auto a)
        {
                autos[i] = a;
                i = i + 1;
        }

        public Auto getAuto(int p)
        {
                return autos[p];
        }
}
```

VB

```
Public Class Persona
    Private _primerNombre As String
    Private _segundoNombre As String
    Private _apellido As String
    Private _documento As String
    Private _autos(10) As Auto

    Public Property Autos() As Auto()
        Get
            Autos = _autos
        End Get
        Set(ByVal value As Auto())
            _autos = value
        End Set
    End Property

    Public Property PrimerNombre() As String
        Get
            PrimerNombre = _primerNombre
        End Get
        Set(ByVal value As String)
            _primerNombre = value
        End Set
    End Property

    Public Property SegundoNombre() As String
        Get
            SegundoNombre = _segundoNombre
        End Get
        Set(ByVal value As String)
            _segundoNombre = value
        End Set
    End Property

    Public Property Apellido() As String
```

```
        Get
            Apellido = _apellido
        End Get
        Set(ByVal value As String)
            _apellido = value
        End Set
    End Property

    Public Property Documento() As String
        Get
            Documento = _documento
        End Get
        Set(ByVal value As String)
            _documento = value
        End Set
    End Property

    Public Sub New(ByVal p As String, ByVal s As String, ByVal a As String, _
              ByVal d As String)
        _primerNombre = p
        _segundoNombre = s
        _apellido = a
        _documento = d
    End Sub

End Class
```

Clase UsaAsociacionCalificada

C#

```
class UsaAsociacionCalificada
{
       static void Main(string[] args)
       {
              Persona p1 = new Persona("Juan", "Ignacio", "Perez","21111222");
              Persona p2 = new Persona("Pedro", "Alberto", "Almeida","23222111");
              Persona p3 = new Persona("Carlos", "Alejo", "Garcia","21444555");
              Persona p4 = new Persona("Sergio", "Ricardo", "Burr","21666777");

              Auto a1 = new Auto("Fiat", "Uno","1996");
              Auto a2 = new Auto("Fiat", "Dos","1997");
              Auto a3 = new Auto("Fiat", "Tres","1998");
              Auto a4 = new Auto("Fiat", "Cuatro","1999");
              Auto a5 = new Auto("Fiat", "Cinco","2000");


              //Asignar los autos de la persona 1
              p1.setAuto(a1);
              p1.setAuto(a3);
              // Registrar que el auto fue asignado a
              // la persona 1
              a1.setPersona(p1);
              a3.setPersona(p1);
```

```
//Asignar los autos de la persona 2
p2.setAuto(a2);
p2.setAuto(a4);
// Registrar que el auto fue asignado a
// la persona 2
a2.setPersona(p2);
a4.setPersona(p2);

//Asignar los autos de la persona 3
p3.setAuto(a1);
p3.setAuto(a5);
// Registrar que el auto fue asignado a
// la persona 3
a1.setPersona(p3);
a5.setPersona(p3);

//Asignar los autos de la persona 4
p4.setAuto(a2);
p4.setAuto(a4);
// Registrar que el auto fue asignado a
// la persona 4
a2.setPersona(p4);
a4.setPersona(p4);

Console.WriteLine("Autos Asignados a la Persona 1");
Console.WriteLine("Marca: " + p1.getAuto(0).Marca
       + " Modelo: " + p1.getAuto(0).Modelo
       + " Año: " + p1.getAuto(0).Anio);
Console.WriteLine("Marca: " + p1.getAuto(1).Marca
       + " Modelo: " + p1.getAuto(1).Modelo
       + " Año: " + p1.getAuto(1).Anio);
Console.WriteLine("------------------------------");
Console.WriteLine("Autos Asignados a la Persona 2");
Console.WriteLine("Marca: " + p2.getAuto(0).Marca
       + " Modelo: " + p2.getAuto(0).Modelo
       + " Año: " + p2.getAuto(0).Anio);
Console.WriteLine("Marca: " + p2.getAuto(1).Marca
       + " Modelo: " + p2.getAuto(1).Modelo
       + " Año: " + p2.getAuto(1).Anio);
Console.WriteLine("------------------------------");
Console.WriteLine("Autos Asignados a la Persona 3");
Console.WriteLine("Marca: " + p3.getAuto(0).Marca
       + " Modelo: " + p3.getAuto(0).Modelo
       + " Año: " + p3.getAuto(0).Anio);
Console.WriteLine("Marca: " + p3.getAuto(1).Marca
       + " Modelo: " + p3.getAuto(1).Modelo
       + " Año: " + p3.getAuto(1).Anio);
Console.WriteLine("------------------------------");
Console.WriteLine("Autos Asignados a la Persona 4");
Console.WriteLine("Marca: " + p4.getAuto(0).Marca
       + " Modelo: " + p4.getAuto(0).Modelo
       + " Año: " + p4.getAuto(0).Anio);
Console.WriteLine("Marca: " + p4.getAuto(1).Marca
       + " Modelo: " + p4.getAuto(1).Modelo
       + " Año: " + p4.getAuto(1).Anio);
```

```
            Console.WriteLine("-----------------------------");
            Console.WriteLine("*****************************");
            Console.WriteLine("-----------------------------");
            Console.WriteLine("Personas Asignadas al auto 1");
            Console.WriteLine("Nombre: " + a1.getPersona(0).PrimerNombre
                  + " " + a1.getPersona(0).SegundoNombre
                  + " " + a1.getPersona(0).Apellido);
            Console.WriteLine("Nombre: " + a1.getPersona(1).PrimerNombre
                  + " " + a1.getPersona(1).SegundoNombre
                  + " " + a1.getPersona(1).Apellido);
            Console.WriteLine("-----------------------------");
            Console.WriteLine("Personas Asignadas al auto 2");
            Console.WriteLine("Nombre: " + a2.getPersona(0).PrimerNombre
                  + " " + a2.getPersona(0).SegundoNombre
                  + " " + a2.getPersona(0).Apellido);
            Console.WriteLine("Nombre: " + a2.getPersona(1).PrimerNombre
                  + " " + a2.getPersona(1).SegundoNombre
                  + " " + a2.getPersona(1).Apellido);
            Console.WriteLine("-----------------------------");
            Console.WriteLine("Personas Asignadas al auto 3");
            Console.WriteLine("Nombre: " + a3.getPersona(0).PrimerNombre
                  + " " + a3.getPersona(0).SegundoNombre
                  + " " + a3.getPersona(0).Apellido);
            Console.WriteLine("-----------------------------");
            Console.WriteLine("Personas Asignadas al auto 4");
            Console.WriteLine("Nombre: " + a4.getPersona(0).PrimerNombre
                  + " " + a4.getPersona(0).SegundoNombre
                  + " " + a4.getPersona(0).Apellido);
            Console.WriteLine("Nombre: " + a4.getPersona(1).PrimerNombre
                  + " " + a4.getPersona(1).SegundoNombre
                  + " " + a4.getPersona(1).Apellido);
            Console.WriteLine("-----------------------------");
            Console.WriteLine("Personas Asignadas al auto 5");
            Console.WriteLine("Nombre: " + a5.getPersona(0).PrimerNombre
                  + " " + a5.getPersona(0).SegundoNombre
                  + " " + a5.getPersona(0).Apellido);

            Console.ReadKey();
      }
}
```

VB

```
Sub Main()
      Dim p1 As Persona = New Persona("Juan", "Ignacio", "Perez", "21111222")
      Dim p2 As Persona = New Persona("Pedro", "Alberto", "Almeida", "23222111")
      Dim p3 As Persona = New Persona("Carlos", "Alejo", "Garcia", "21444555")
      Dim p4 As Persona = New Persona("Sergio", "Ricardo", "Burr", "21666777")

      Dim a1 As Auto = New Auto("Fiat", "Uno", "1996")
      Dim a2 As Auto = New Auto("Fiat", "Dos", "1997")
      Dim a3 As Auto = New Auto("Fiat", "Tres", "1998")
      Dim a4 As Auto = New Auto("Fiat", "Cuatro", "1999")
      Dim a5 As Auto = New Auto("Fiat", "Cinco", "2000")
```

```
'Asignar los autos de la persona 1
p1.Autos(0) = a1
p1.Autos(1) = a3
' Registrar que el auto fue asignado a
' la persona 1
a1.Personas(0) = p1
a3.Personas(0) = p1

'Asignar los autos de la persona 2
p2.Autos(0) = a2
p2.Autos(1) = a4
' Registrar que el auto fue asignado a
' la persona 2
a2.Personas(0) = p2
a4.Personas(0) = p2

'Asignar los autos de la persona 3
p3.Autos(0) = a1
p3.Autos(1) = a5
' Registrar que el auto fue asignado a
' la persona 3
a1.Personas(1) = p3
a5.Personas(0) = p3

'Asignar los autos de la persona 4
p4.Autos(0) = a2
p4.Autos(1) = a4
' Registrar que el auto fue asignado a
' la persona 4
a2.Personas(1) = p4
a4.Personas(1) = p4

Console.WriteLine("Autos Asignados a la Persona 1")
Console.WriteLine("Marca: " + p1.Autos(0).Marca _
       + " Modelo: " + p1.Autos(0).Modelo _
       + " Año: " + p1.Autos(0).Anio)
Console.WriteLine("Marca: " + p1.Autos(1).Marca _
       + " Modelo: " + p1.Autos(1).Modelo _
       + " Año: " + p1.Autos(1).Anio)
Console.WriteLine("-----------------------------")
Console.WriteLine("Autos Asignados a la Persona 2")
Console.WriteLine("Marca: " + p2.Autos(0).Marca _
       + " Modelo: " + p2.Autos(0).Modelo _
       + " Año: " + p2.Autos(0).Anio)
Console.WriteLine("Marca: " + p2.Autos(1).Marca _
       + " Modelo: " + p2.Autos(1).Modelo _
       + " Año: " + p2.Autos(1).Anio)
Console.WriteLine("-----------------------------")
Console.WriteLine("Autos Asignados a la Persona 3")
Console.WriteLine("Marca: " + p3.Autos(0).Marca _
       + " Modelo: " + p3.Autos(0).Modelo _
       + " Año: " + p3.Autos(0).Anio)
Console.WriteLine("Marca: " + p3.Autos(1).Marca _
       + " Modelo: " + p3.Autos(1).Modelo _
       + " Año: " + p3.Autos(1).Anio)
```

```
        Console.WriteLine("-----------------------------")
        Console.WriteLine("Autos Asignados a la Persona 4")
        Console.WriteLine("Marca: " + p4.Autos(0).Marca _
               + " Modelo: " + p4.Autos(0).Modelo _
               + " Año: " + p4.Autos(0).Anio)
        Console.WriteLine("Marca: " + p4.Autos(1).Marca _
               + " Modelo: " + p4.Autos(1).Modelo _
               + " Año: " + p4.Autos(1).Anio)
        Console.WriteLine("-----------------------------")
        Console.WriteLine("*****************************")
        Console.WriteLine("-----------------------------")
        Console.WriteLine("Personas Asignadas al auto 1")
        Console.WriteLine("Nombre: " + a1.Personas(0).PrimerNombre _
               + " " + a1.Personas(0).SegundoNombre _
               + " " + a1.Personas(0).Apellido)
        Console.WriteLine("Nombre: " + a1.Personas(1).PrimerNombre _
               + " " + a1.Personas(1).SegundoNombre _
               + " " + a1.Personas(1).Apellido)
        Console.WriteLine("-----------------------------")
        Console.WriteLine("Personas Asignadas al auto 2")
        Console.WriteLine("Nombre: " + a2.Personas(0).PrimerNombre _
               + " " + a2.Personas(0).SegundoNombre _
               + " " + a2.Personas(0).Apellido)
        Console.WriteLine("Nombre: " + a2.Personas(1).PrimerNombre _
               + " " + a2.Personas(1).SegundoNombre _
               + " " + a2.Personas(1).Apellido)
        Console.WriteLine("-----------------------------")
        Console.WriteLine("Personas Asignadas al auto 3")
        Console.WriteLine("Nombre: " + a3.Personas(0).PrimerNombre _
               + " " + a3.Personas(0).SegundoNombre _
               + " " + a3.Personas(0).Apellido)
        Console.WriteLine("-----------------------------")
        Console.WriteLine("Personas Asignadas al auto 4")
        Console.WriteLine("Nombre: " + a4.Personas(0).PrimerNombre _
               + " " + a4.Personas(0).SegundoNombre _
               + " " + a4.Personas(0).Apellido)
        Console.WriteLine("Nombre: " + a4.Personas(1).PrimerNombre _
               + " " + a4.Personas(1).SegundoNombre _
               + " " + a4.Personas(1).Apellido)
        Console.WriteLine("-----------------------------")
        Console.WriteLine("Personas Asignadas al auto 5")
        Console.WriteLine("Nombre: " + a5.Personas(0).PrimerNombre _
               + " " + a5.Personas(0).SegundoNombre _
               + " " + a5.Personas(0).Apellido)

        Console.ReadKey()

End Sub
```

Ejercicio 2

El tema del que trata en este ejercicio es el siguiente:

- Asociaciones complejas

Basándose en el diseño realizado en el módulo anterior, codificar en .Net las asociaciones complejas que se hayan detectado

Agregaciones

Es una relación más fuerte entre clases ya que se mantiene por un período de tiempo mucho más largo que con una asociación simple. Las agregaciones son parte fundamental de la programación porque permiten el acceso a otros objetos mediante los servicios públicos que estos prestan incorporándolos como parte del objeto que lo agrega.

Al no tener la limitación temporal de la composición que necesite el objeto a lo largo de todo el tiempo de vida del que lo compone, resulta una técnica de programación mucho más flexible y proclive a los cambios dinámicos que suceden en los sistemas. Por eso la agregación es la relación más utilizada en la programación orientada a objetos, compitiendo inclusive con la herencia que, como se verá más adelante, es la base de la reutilización del código.

Se caracteriza por las palabras “tiene un” o “tiene una” y traducido a la programación esto significa que la clase declara al menos una variable de instancia que almacena la referencia al objeto agregado

Aunque no es mandatorio, la relación puede durar la vida del objeto que tiene declarada la agregación. Sin embargo, para que sea una agregación, debe existir al menos una forma en la cual el objeto agregado pueda ser remplazado por otro del mismo tipo.

Si se tiene el siguiente enunciado

Una persona tiene un auto

La resolución en UML estaría dada por el siguiente gráfico

Y la correspondiente solución en código.

Ejemplo

Clase Auto

C#

```
class Auto
{
	private String marca;

	public String Marca
	{
		get { return marca; }
```

```
            set { marca = value; }
        }
        private String modelo;

        public String Modelo
        {
            get { return modelo; }
            set { modelo = value; }
        }
        private String anio;

        public String Anio
        {
            get { return anio; }
            set { anio = value; }
        }

        public Auto(String ma, String mo, String an)
        {
            marca = ma;
            modelo = mo;
            anio = an;
        }
}
```

VB

```
Public Class Auto
    Private _marca As String
    Private _modelo As String
    Private _anio As String

    Public Property Marca() As String
        Get
            Marca = _marca
        End Get
        Set(ByVal value As String)
            _marca = value
        End Set
    End Property

    Public Property Modelo() As String
        Get
            Modelo = _modelo
        End Get
        Set(ByVal value As String)
            _modelo = value
        End Set
    End Property

    Public Property Anio() As String
        Get
            Anio = _anio
        End Get
        Set(ByVal value As String)
            _anio = value
```

```
        End Set
    End Property

    Public Sub New(ByVal ma As String, ByVal mo As String, ByVal an As String)
        Marca = ma
        Modelo = mo
        Anio = an
    End Sub

End Class
```

Clase Persona

C#

```
class Persona
{
        private String primerNombre;

        public String PrimerNombre
        {
                get { return primerNombre; }
                set { primerNombre = value; }
        }
        private String segundoNombre;

        public String SegundoNombre
        {
                get { return segundoNombre; }
                set { segundoNombre = value; }
        }
        private String apellido;

        public String Apellido
        {
                get { return apellido; }
                set { apellido = value; }
        }
        private String documento;

        public String Documento
        {
                get { return documento; }
                set { documento = value; }
        }
        private Auto auto;

        public Auto Auto
        {
                get { return auto; }
                set { auto = value; }
        }

        public Persona(String p,
                String s,
```

```
            String a,
            String d)
    {
            primerNombre = p;
            segundoNombre = s;
            apellido = a;
            documento = d;
    }
}
```

VB

```
Public Class Persona
    Private _primerNombre As String
    Private _segundoNombre As String
    Private _apellido As String
    Private _documento As String
    Private _detalles As String
    Private _auto As Auto


    Public Property PrimerNombre() As String
        Get
            PrimerNombre = _primerNombre
        End Get
        Set(ByVal value As String)
            _primerNombre = value
        End Set
    End Property

    Public Property SegundoNombre() As String
        Get
            SegundoNombre = _segundoNombre
        End Get
        Set(ByVal value As String)
            _segundoNombre = value
        End Set
    End Property

    Public Property Apellido() As String
        Get
            Apellido = _apellido
        End Get
        Set(ByVal value As String)
            _apellido = value
        End Set
    End Property

    Public Property Documento() As String
        Get
            Documento = _documento
        End Get
        Set(ByVal value As String)
            _documento = value
        End Set
    End Property
```

```
    Public Property Auto() As Auto
        Get
            Auto = _auto
        End Get
        Set(ByVal value As Auto)
            _auto = value
        End Set
    End Property

    Public Property Detalles() As String
        Get
            Detalles = _detalles
        End Get
        Set(ByVal value As String)
            _detalles = value
        End Set
    End Property


    Public Sub New(ByVal p As String, ByVal s As String, ByVal a As String, ByVal d
As String)
        _primerNombre = p
        _segundoNombre = s
        _apellido = a
        _documento = d
    End Sub


End Class
```

Clase UsaAgregacion

C#

```
class UsaAgregacion
{
       static void Main(string[] args)
       {
              Auto a = new Auto("Fiat", "Uno","1998");
              Persona p = new Persona("Juan", "Ignacio", "Perez","21111222");

              p.Auto=a;

              Console.WriteLine("La Persona tiene agregado un auto " + p.Auto.Marca
                                  +" " + p.Auto.Modelo+ " del año " + p.Auto.Anio);
              Console.ReadKey();

       }
}
```

VB

```
Sub Main()
       Dim p As Persona = New Persona("Juan", "Ignacio", "Perez", "21111222")
```

```
        Dim a As Auto = New Auto("Fiat", "Uno", "1998")

        p.Auto = a
        Console.WriteLine("La Persona tiene agregado un auto " + p.Auto.Marca _
               + " " + p.Auto.Modelo + " del año " + p.Auto.Anio)

        Console.ReadKey()

End Sub
```

Composiciones

Es la relación más fuerte entre clases (al igual que la herencia, que se verá posteriormente) ya que los objetos, el que compone y el compuesto, están ligados durante tiempo de vida, es decir, cuando se crea uno se crea el otro y cuando se destruye uno se destruye al otro. Cualquier caso fuera de esta consideración lo convierte en una agregación.

Se caracteriza por las palabras "siempre tiene" porque su relación es durante todo el tiempo de vida de ambos objetos. Debe quedar el claro que el objeto compuesto puede tener una instancia diferente e independiente al de la composición, pero cuando se crea dentro de un objeto que lo compone no puede escapar a la limitación temporal que impone la composición, dura el mismo tiempo que la vida del objeto siempre.

Si se tiene el siguiente enunciado

Una persona siempre tiene una dirección

La resolución en UML estaría dada por el siguiente gráfico

Y la correspondiente solución en código. Notar que en el caso de la composición se puede tener una referencia al objeto**, pero no se la puede cambiar**.

Ejemplo

Clase Direccion

C#

```
class Direccion
{
        private String nombreCalle;// nombre completo de la calle

        public String NombreCalle
```

```
        {
                get { return nombreCalle; }
                set { nombreCalle = value; }
        }
        private int nro;                    // nro de la casa o departamento

        public int Nro
        {
                get { return nro; }
               set { nro = value; }
        }

        public Direccion(
                String nomCalle,     // nombre completo de la calle
                int n)// nro de la casa
        {
                nombreCalle = nomCalle;
                nro = n;
        }
}
```

VB

```
Public Class Direccion
    Private _nombreCalle As String
    Private _nro As Integer

    Public Property NombreCalle() As String
        Get
            NombreCalle = _nombreCalle
        End Get
        Set(ByVal value As String)
            _nombreCalle = value
        End Set
    End Property

    Public Property Nro() As String
        Get
            Nro = _nro
        End Get
        Set(ByVal value As String)
            _nro = value
        End Set
    End Property

    Public Sub New(ByVal nombreCalle As String, ByVal nro As Integer)
        Me._nombreCalle = nombreCalle
        Me._nro = nro
    End Sub

End Class
```

Clase Persona

C#

```
class Persona
{
	private String primerNombre;

	public String PrimerNombre
	{
		get { return primerNombre; }
		set { primerNombre = value; }
	}
	private String segundoNombre;

	public String SegundoNombre
	{
		get { return segundoNombre; }
		set { segundoNombre = value; }
	}
	private String apellido;

	public String Apellido
	{
		get { return apellido; }
		set { apellido = value; }
	}
	private String documento;

	public String Documento
	{
		get { return documento; }
		set { documento = value; }
	}
	private Direccion dir;

	public Direccion Dir
	{
		get { return dir; }
	}

	public Persona(String p,
		String s,
		String a,
		String d,
		String direccion,
		int num)
	{
		primerNombre = p;
		segundoNombre = s;
		apellido = a;
		documento = d;
		dir = new Direccion(direccion, num);
	}
}
```

VB

```
Public Class Persona
    Private _primerNombre As String
    Private _segundoNombre As String
    Private _apellido As String
    Private _documento As String
    Private _detalles As String
    Private _direccion As Direccion

    Public Property PrimerNombre() As String
        Get
            PrimerNombre = _primerNombre
        End Get
        Set(ByVal value As String)
            _primerNombre = value
        End Set
    End Property

    Public Property SegundoNombre() As String
        Get
            SegundoNombre = _segundoNombre
        End Get
        Set(ByVal value As String)
            _segundoNombre = value
        End Set
    End Property

    Public Property Apellido() As String
        Get
            Apellido = _apellido
        End Get
        Set(ByVal value As String)
            _apellido = value
        End Set
    End Property

    Public Property Documento() As String
        Get
            Documento = _documento
        End Get
        Set(ByVal value As String)
            _documento = value
        End Set
    End Property

    Public Property Direccion() As Direccion
        Get
            Direccion = _direccion
        End Get
        Set(ByVal value As Direccion)
            _direccion = value
        End Set
    End Property
```

```
    Public Property Detalles() As String
        Get
            Detalles = _detalles
        End Get
        Set(ByVal value As String)
            _detalles = value
        End Set
    End Property

    Public Sub New(ByVal p As String, ByVal s As String, ByVal a As String, ByVal d
As String, _
                   ByVal direccion As String, ByVal num As Integer)
        _primerNombre = p
        _segundoNombre = s
        _apellido = a
        _documento = d
        _direccion = New Direccion(direccion, num)
    End Sub

End Class
```

Clase UsaComposicion

C#

```
class UsaComposicion
{
       static void Main(string[] args)
       {
              Persona p = new Persona("Juan", "Pedro",
                     "Diaz","22333444","Venezuela",1301);
              Console.WriteLine("La persona vive en " + p.Dir.NombreCalle + ", Nro. "
                     + p.Dir.Nro );

              Console.WriteLine("Mudate Juan!!!!");

              p.Dir.NombreCalle = "Saenz Peña";
              p.Dir.Nro = 151;
              Console.WriteLine("La persona vive en " + p.Dir.NombreCalle + ", Nro. "
                     + p.Dir.Nro);
              Console.ReadKey();
       }
}
```

VB

```
Sub Main()
       Dim p As Persona = New Persona("Juan", "Pedro", "Diaz", "22333444", _
                     "Venezuela", 1301)
       Console.WriteLine("La persona vive en " + p.Direccion.NombreCalle + _
                     ", Nro. " + p.Direccion.Nro)
       Console.ReadKey()

End Sub
```

Ejercicios 3 y 4

Los temas de los que se tratan en este ejercicio son los siguientes:

- Agregación
- Composición

En el ejercicio 3, basándose en el diseño realizado en el módulo anterior, codificar en .Net las agregaciones y composiciones que se hayan detectado.

En el ejercicio 4, codificar un diagrama UML respetando solamente el diseño del mismo sin conocer el enunciado del problema o como se analizó y diseñó