EJEMPLO CONCESIONARIO

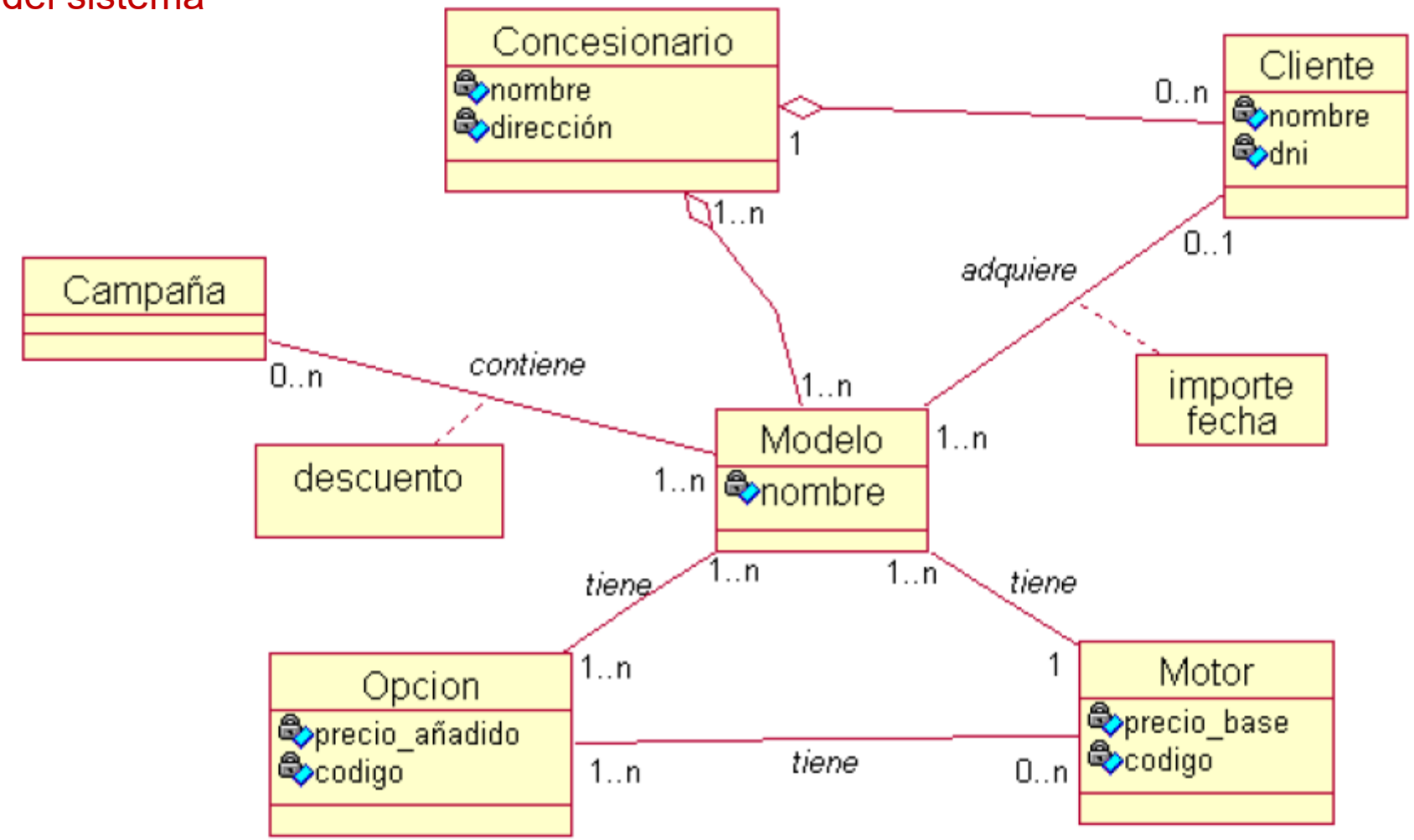
Diseño de clases y constructores
Solución parcial

Ingeniería del Software
ETS Ingeniería Informática
DSIC – UPV

Contenido

- Diagrama de clases del ejemplo
- Diseño en c# de las clases (también sus constructores):
  - Concesionario
  - Cliente
  - Modelo
  - Campaña
  - Opción
  - Motor

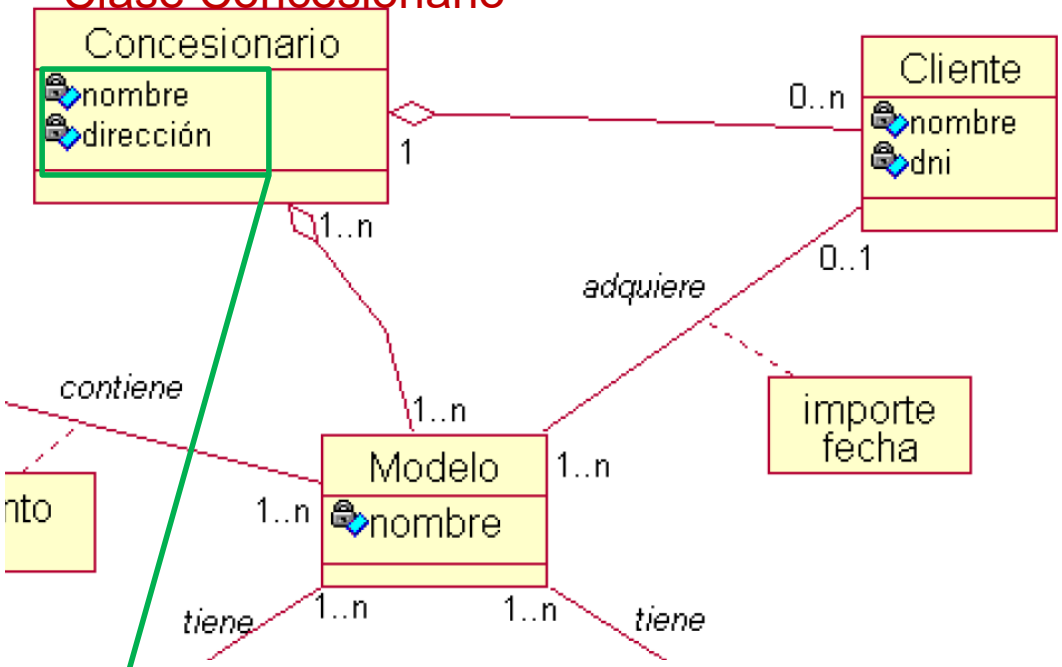
Diagrama de clases del sistema



DISEÑO DE LOS ATRIBUTOS

Cardinalidad Máxima

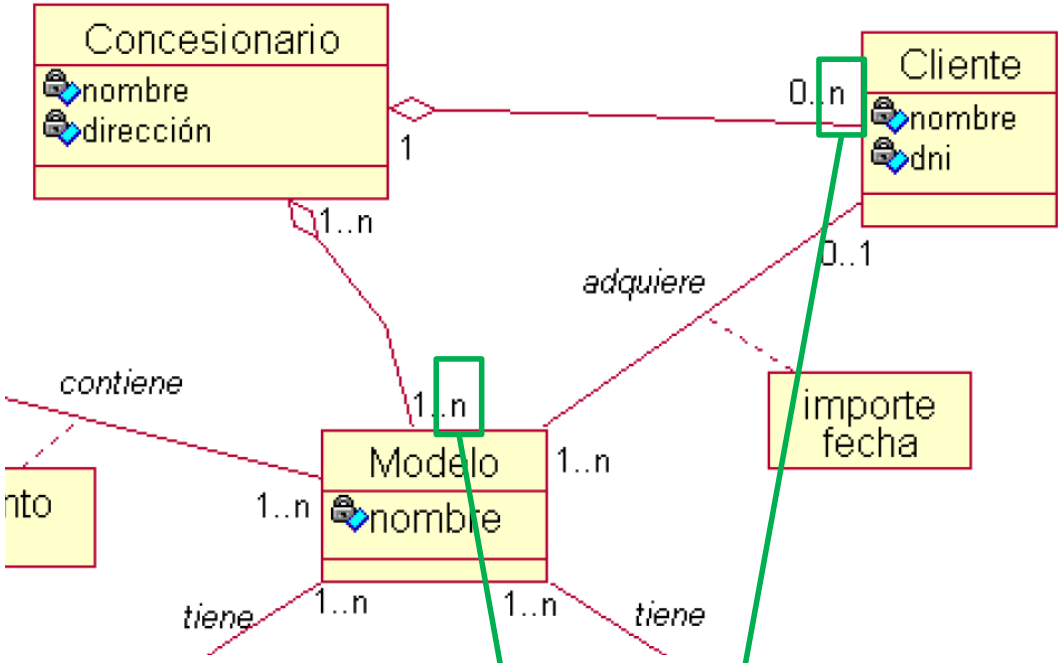
Clase Concesionario



Añadimos sus atributos

```
6 referencias
public class Concesionario{
    private string nombre;
    private string direccion;

    private ICollection <Cliente> clientes; //Tenemos una cardinalidad máxima de N, necesitamos colección
    private ICollection <Modelo> modelos;   //Tenemos una cardinalidad máxima de N, necesitamos colección
```

Clase Concesionario

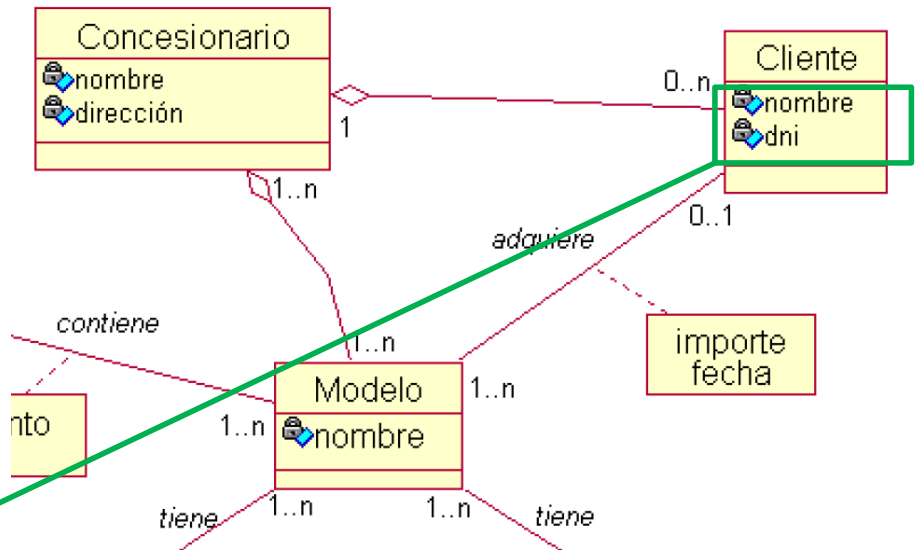


La multiplicidad máxima n nos indica que tenemos que crear dos colecciones en Concesionario

```
6 referencias
public class Concesionario{
    private string nombre;
    private string direccion;

    private ICollection <Cliente> clientes; //Tenemos una cardinalidad máxima de N, necesitamos colección
    private ICollection <Modelo> modelos;   //Tenemos una cardinalidad máxima de N, necesitamos colección
```

Clase Cliente

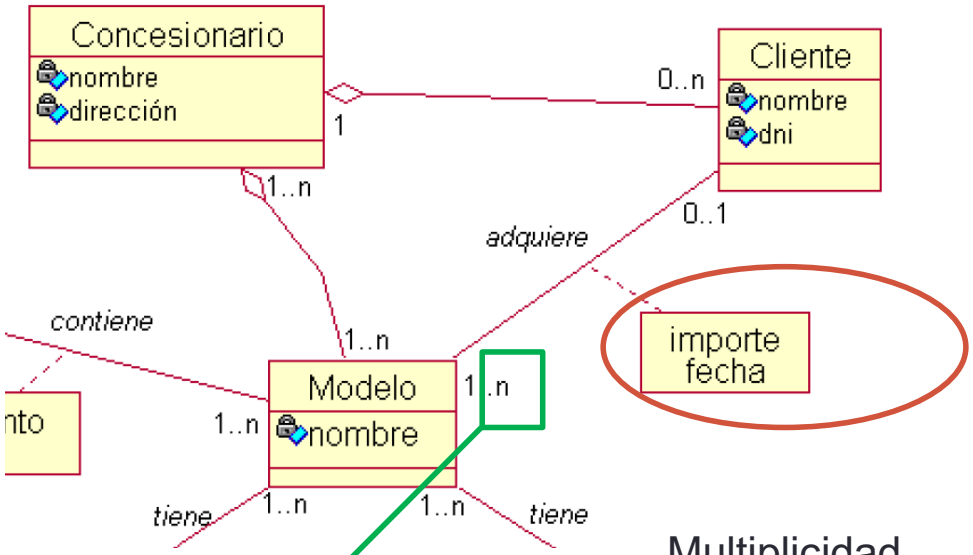
Tenemos que declarar los atributos propios




```
4 referencias
public class Cliente{
    private string nombre;
    private string dni;

    private ICollection <Modelo> modelos; //Tenemos una cardinalidad máxima de N, necesitamos colección
    private Concesionario concesionario;
```

Clase Cliente

La multiplicidad máxima n nos indica que tenemos que crear una colección

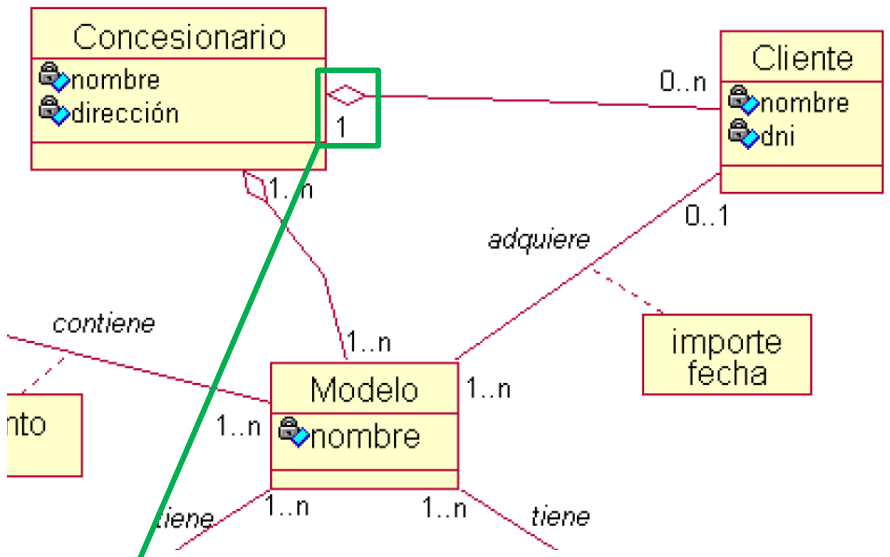


Multiplicidad máxima de 1 a N, los atributos se declarará en Modelo

```
4 referencias
public class Cliente{
    private string nombre;
    private string dni;

    private ICollection <Modelo> modelos; //Tenemos una cardinalidad máxima de N, necesitamos colección
    private Concesionario concesionario;
```

Clase Cliente

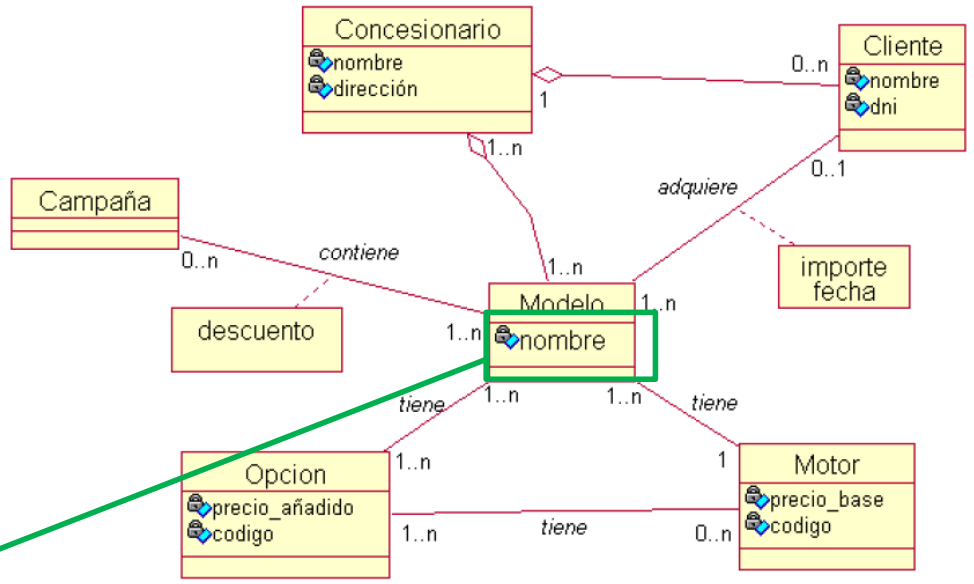
La multiplicidad máxima 1 nos indica que tenemos que añadir un atributo




```
4 referencias
public class Cliente{
    private string nombre;
    private string dni;

    private ICollection <Modelo> modelos; //Tenemos una cardinalidad máxima de N, necesitamos colección
    private Concesionario concesionario;
```

Clase Modelo

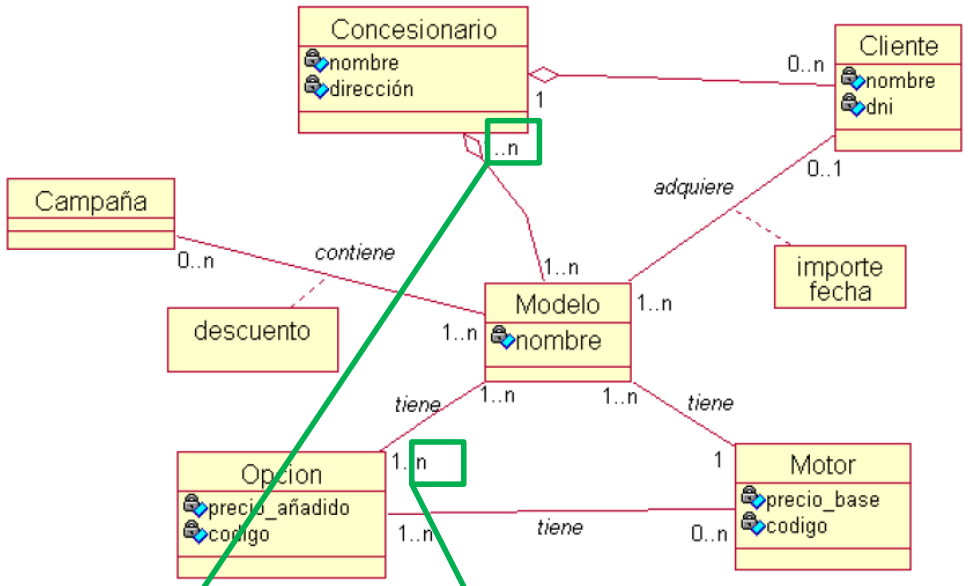
Tenemos que declarar los atributos propios




```
12 referencias
public class Modelo{
    private string nombre;

    private ICollection <Concesionario> concesionarios;  //Tenemos una cardinalidad máxima de N, necesitamos colección
    private ICollection <Contiene> contienes; //El atributo de enlace en una relación N a N promociona a clase
    private ICollection < Opcion> opciones; //Tenemos una cardinalidad máxima de N, necesitamos colección
    private Cliente cliente; //cardinalidad maxima 1, minima cero, no hay que pasarlo en el constructor
    private Motor motor;


    private double importe; //atributo de la relación con cliente, se pone en el lado de muchos
    private DateTime fecha; //atributo de la relación con cliente, se pone en el lado de muchos
```

Clase Modelo

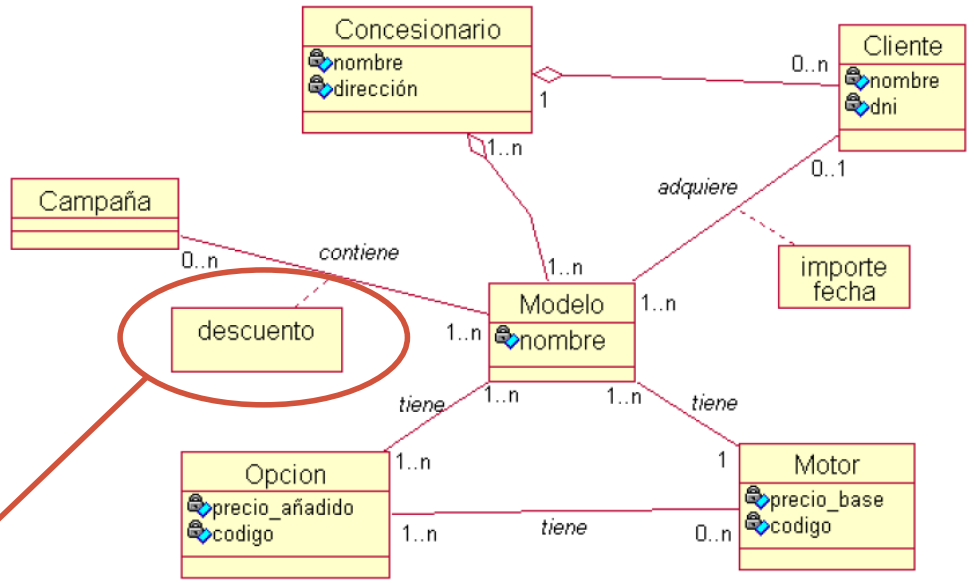
La multiplicidad máxima n nos indica que tenemos que crear una colección

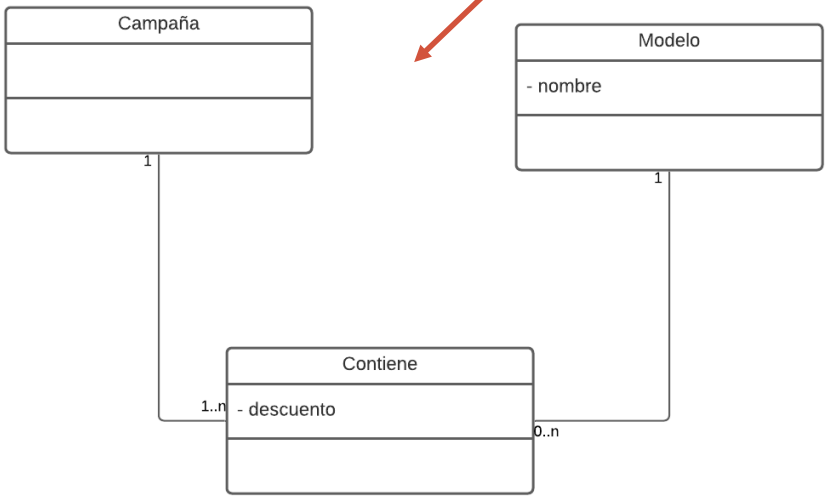



```
12 referencias
public class Modelo{
    private string nombre;

    private ICollection <Concesionario> concesionarios;  //Tenemos una cardinalidad máxima de N, necesitamos colección
    private ICollection <Contiene> contienes; //El atributo de enlace en una relación N a N promociona a clase
    private ICollection < Opcion> opciones; //Tenemos una cardinalidad máxima de N, necesitamos colección
    private Cliente cliente; //cardinalidad maxima 1, minima cero, no hay que pasarlo en el constructor
    private Motor motor;


    private double importe; //atributo de la relación con cliente, se pone en el lado de muchos
    private DateTime fecha; //atributo de la relación con cliente, se pone en el lado de muchos
```

Clase Modelo

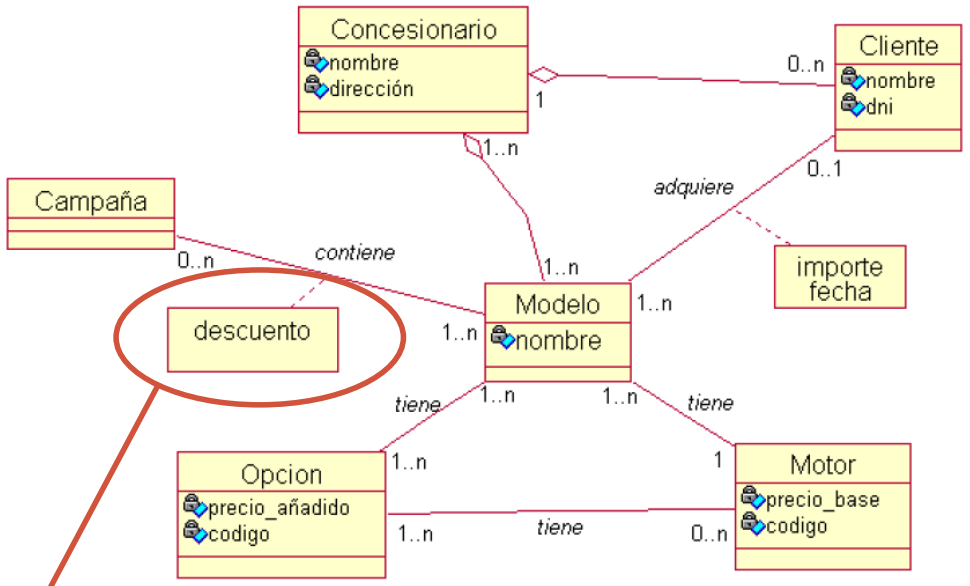
Multiplicidad máxima de N a N, el atributo de enlace promociona a clase. Modelo se relacionará con esa clase con máxima de N





Clase Modelo

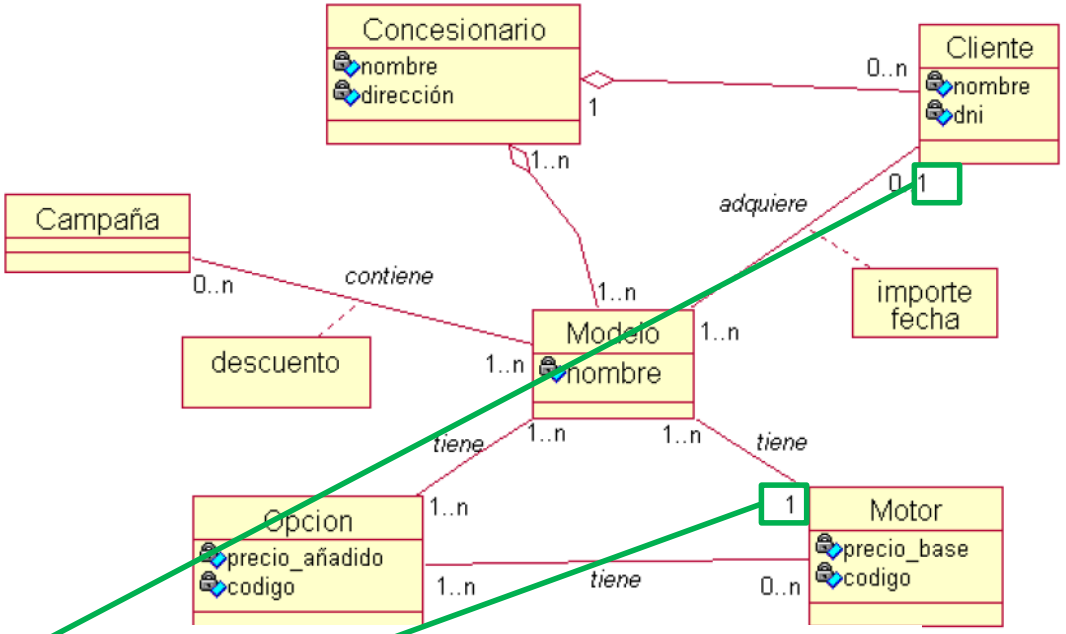
Multiplicidad máxima de N a N, el atributo de enlace promociona a clase. Modelo se relacionará con esa clase con máxima de N




```
12 referencias
public class Modelo{
    private string nombre;

    private ICollection <Concesionario> concesionarios; //Tenemos una cardinalidad máxima de N, necesitamos colección
    private ICollection <Contiene> contienes; //El atributo de enlace en una relación N a N promociona a clase
    private ICollection < Opcion> opciones; //Tenemos una cardinalidad máxima de N, necesitamos colección
    private Cliente cliente; //cardinalidad maxima 1, minima cero, no hay que pasarlo en el constructor
    private Motor motor;


    private double importe; //atributo de la relación con cliente, se pone en el lado de muchos
    private DateTime fecha; //atributo de la relación con cliente, se pone en el lado de muchos
```

Clase Modelo

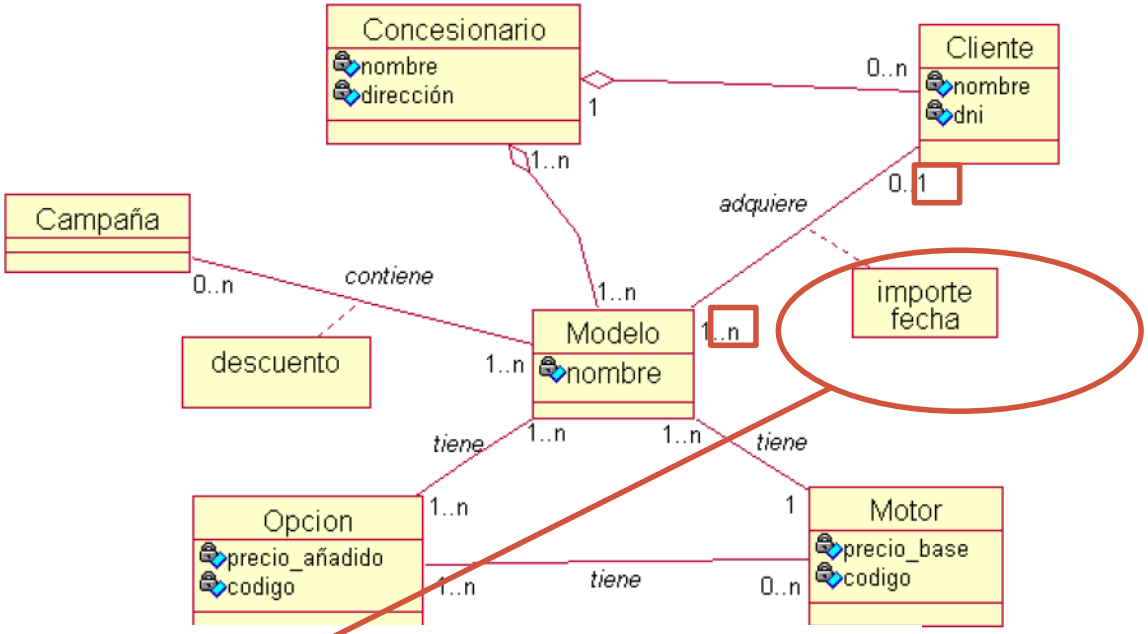
La multiplicidad máxima 1 nos indica que tenemos que añadir un atributo




```
12 referencias
public class Modelo{
    private string nombre;

    private ICollection <Concesionario> concesionarios;  //Tenemos una cardinalidad máxima de N, necesitamos colección
    private ICollection <Contiene> contienes; //El atributo de enlace en una relación N a N promociona a clase
    private ICollection <Opcion> opciones; //Tenemos una cardinalidad máxima de N, necesitamos colección
    private Cliente cliente; //cardinalidad maxima 1, minima cero, no hay que pasarlo en el constructor
    private Motor motor;


    private double importe; //atributo de la relación con cliente, se pone en el lado de muchos
    private DateTime fecha; //atributo de la relación con cliente, se pone en el lado de muchos
```

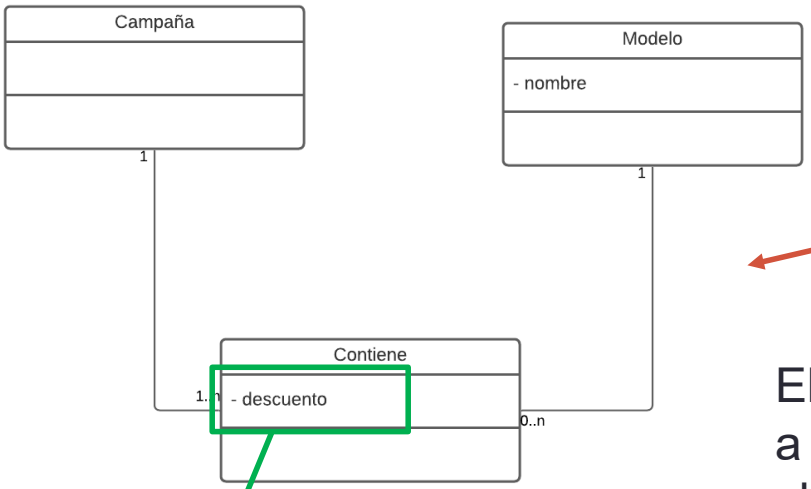
Clase Modelo

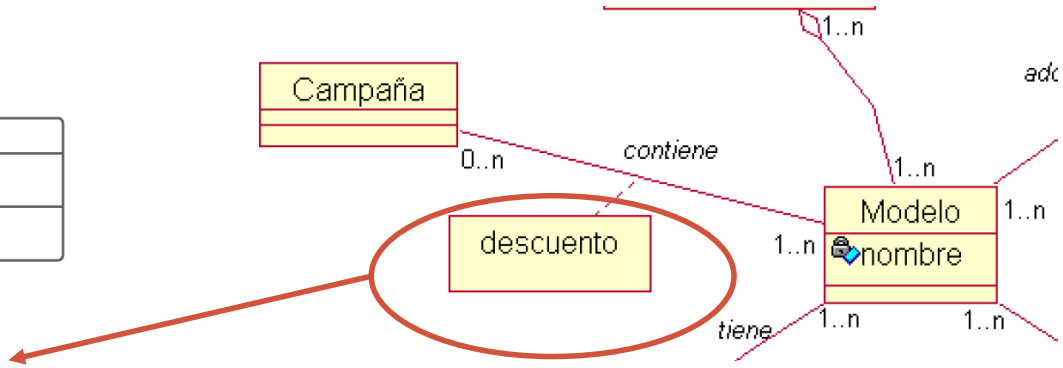
Multiplicidad máxima de 1 a N, los atributos se declaran en Modelo




```
12 referencias
public class Modelo{
    private string nombre;

    private ICollection <Concesionario> concesionarios;  //Tenemos una cardinalidad máxima de N, necesitamos colección
    private ICollection <Contiene> contienes; //El atributo de enlace en una relación N a N promociona a clase
    private ICollection < Opcion> opciones; //Tenemos una cardinalidad máxima de N, necesitamos colección
    private Cliente cliente; //cardinalidad maxima 1, minima cero, no hay que pasarlo en el constructor
    private Motor motor;


    private double importe; //atributo de la relación con cliente, se pone en el lado de muchos
    private DateTime fecha; //atributo de la relación con cliente, se pone en el lado de muchos
```

Clase Contiene

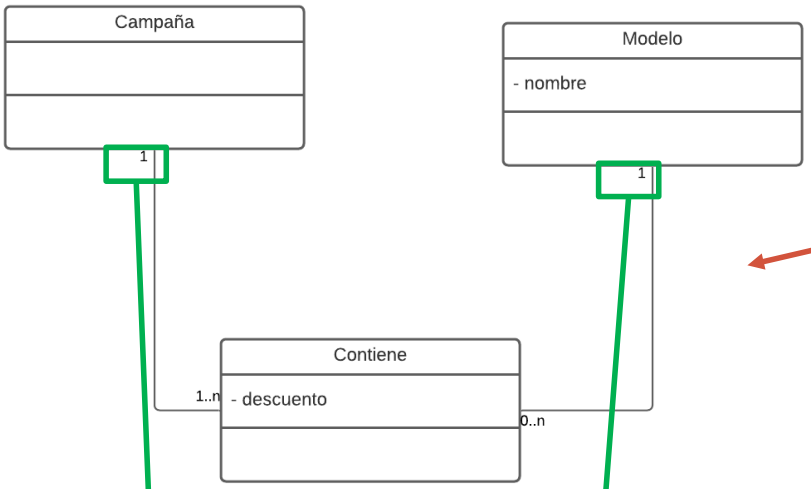


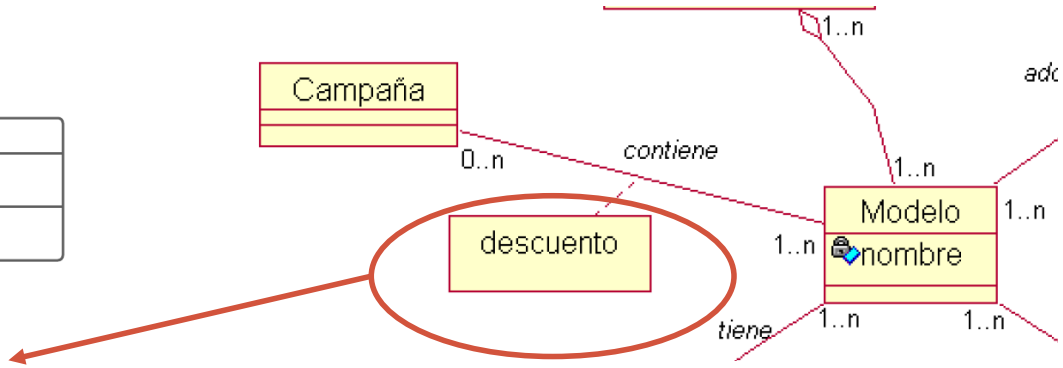


El atributo de la asociación pasa a ser un atributo de la nueva clase

```
6 referencias
public class Contiene{
    //Esta clase surge al promocionar un atributo de una relacion N a N a clase
    private double descuento;

    private Modelo modelo;
    private Campanya campanya;
```

Clase Contiene

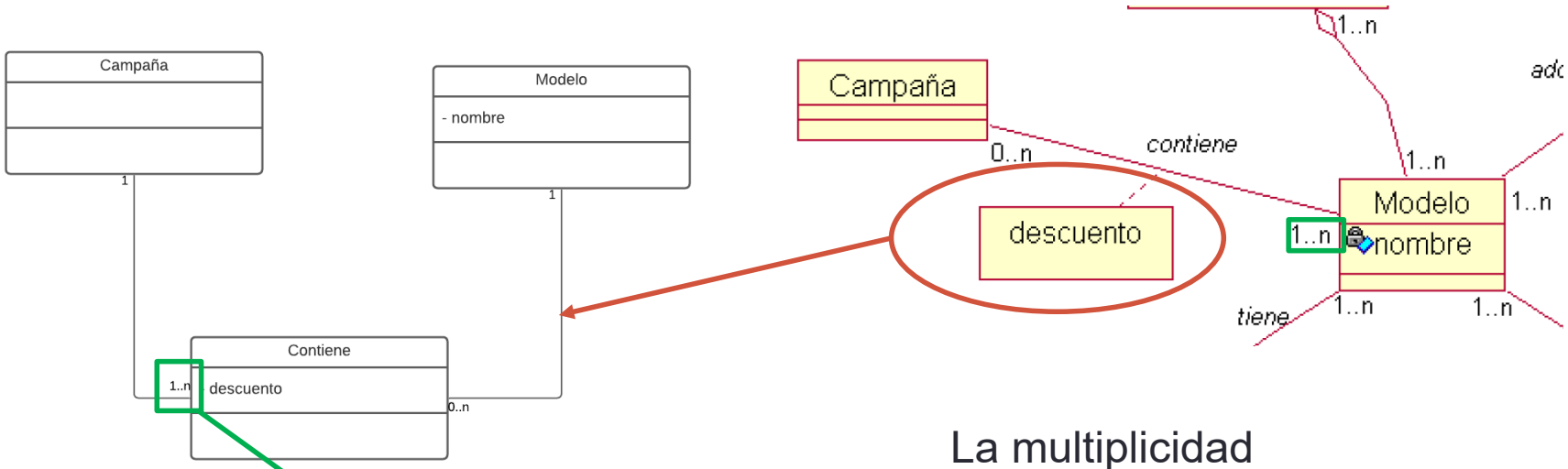




La multiplicidad máxima 1 nos indica que tenemos que añadir un atributo

```
6 referencias
public class Contiene{
    //Esta clase surge al promocionar un atributo de una relacion N a N a clase
    private double descuento;

    private Modelo modelo;
    private Campanya campanya;
```

Clase Campaña

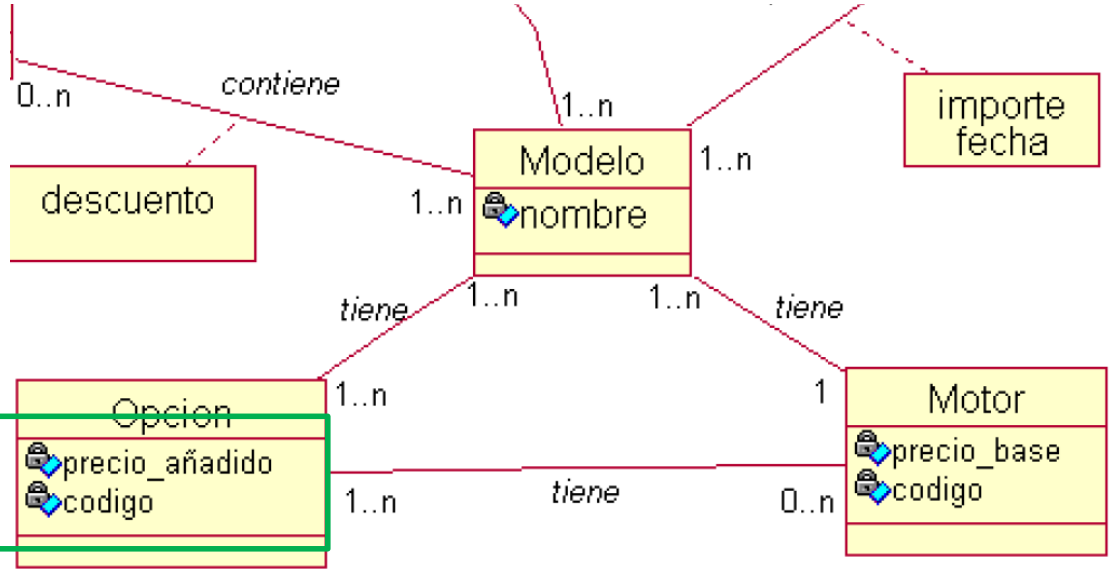


La multiplicidad máxima n nos indica que tenemos que crear una colección

```
public class Campanya{
    private ICollection <Contiene> contienes; //El atributo de enlace en una relación N a N promociona a clase
```

Clase Opción

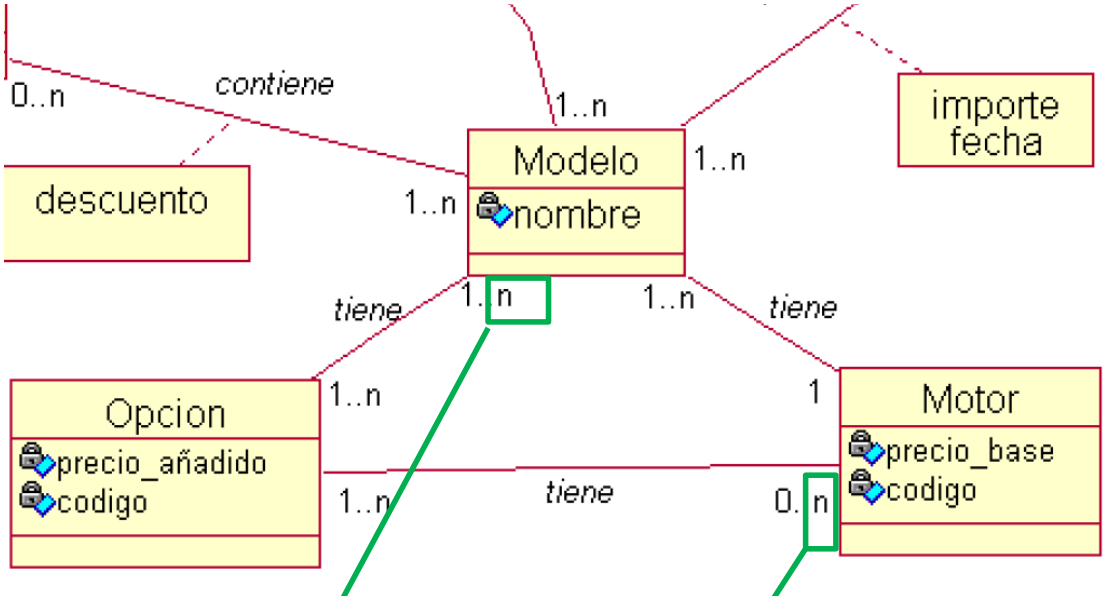
Tenemos que declarar los atributos propios




```
7 referencias
public class Opcion{
    private double precio_anyadido;
    private int codigo;
    private ICollection<Modelo> modelos;//Tenemos una cardinalidad máxima de N, necesitamos colección
    private ICollection<Motor> motores;//Tenemos una cardinalidad máxima de N, necesitamos colección
```

Clase Opción

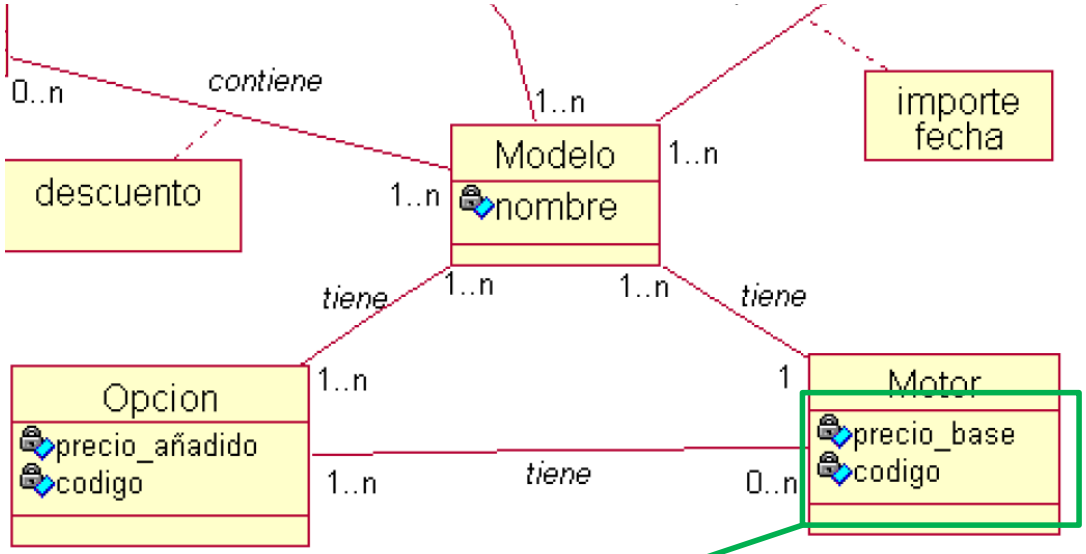
La multiplicidad máxima n nos indica que tenemos que crear una colección




```
7 referencias
public class Opcion{
    private double precio_anyadido;
    private int codigo;
    private ICollection<Modelo> modelos;//Tenemos una cardinalidad máxima de N, necesitamos colección
    private ICollection<Motor> motores;//Tenemos una cardinalidad máxima de N, necesitamos colección
```

Clase Motor

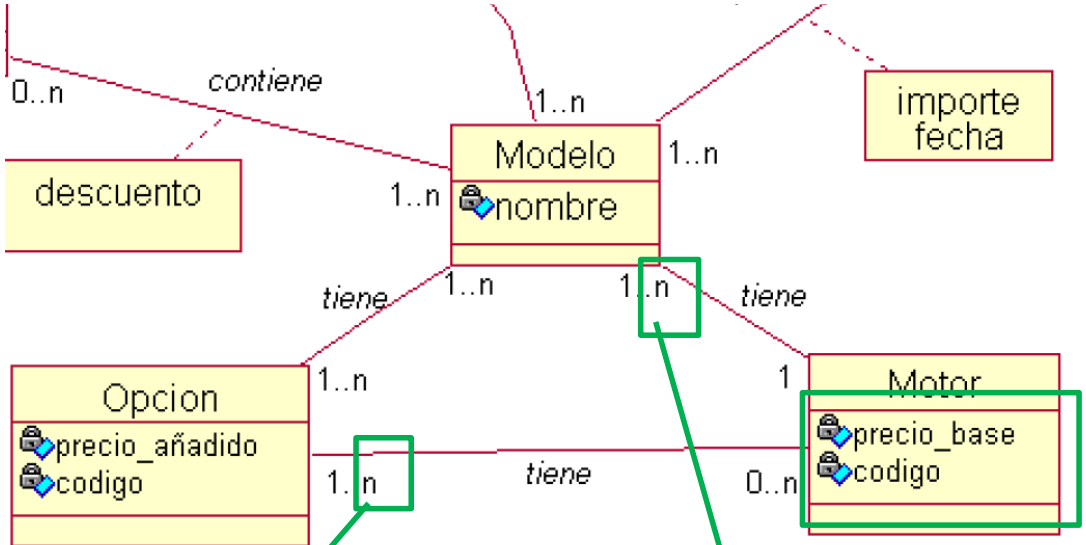
Tenemos que declarar los atributos propios




```
5 referencias
public class Motor{
    private double precio_base;
    private int codigo;

    private ICollection<Modelo> modelos;//Tenemos una cardinalidad máxima de N, necesitamos colección
    private ICollection < Opcion> opciones; //Tenemos una cardinalidad máxima de N, necesitamos colección
```

Clase Motor

La multiplicidad máxima n nos indica que tenemos que crear una colección

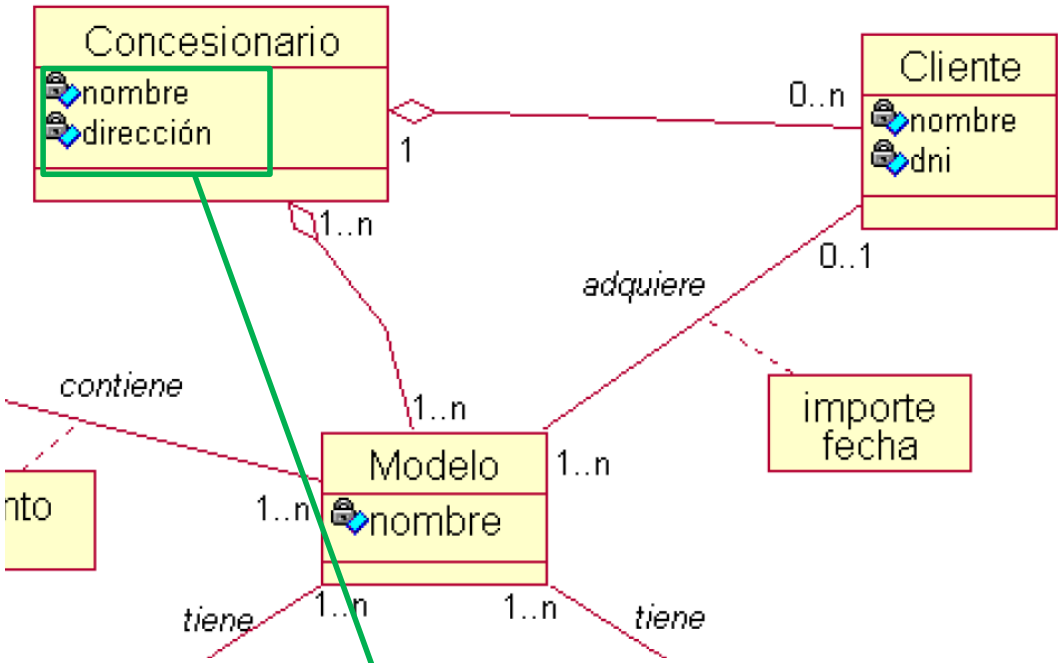



```
5 referencias
public class Motor{
    private double precio_base;
    private int codigo;

    private ICollection<Modelo> modelos;//Tenemos una cardinalidad máxima de N, necesitamos colección
    private ICollection < Opcion> opciones; //Tenemos una cardinalidad máxima de N, necesitamos colección
```

DISEÑO DE CONSTRUCTORES

Cardinalidad Mínima

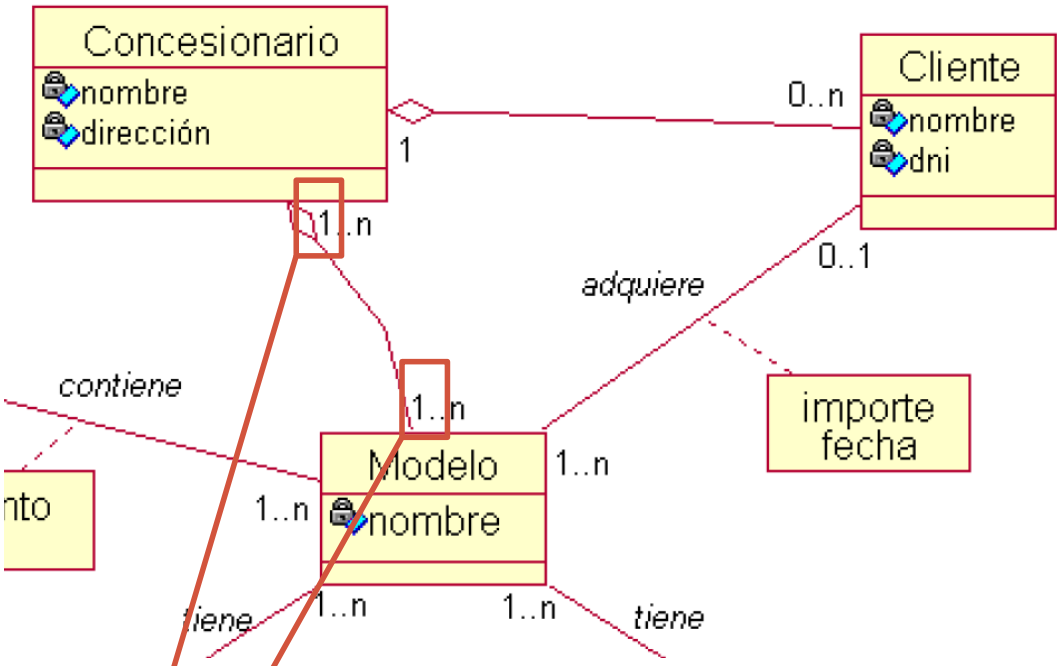
Clase Concesionario



Pasamos los valores de los atributos e inicializamos

```
0 referencias
public Concesionario(string nombre, string direccion){
    this.nombre = nombre;
    this.direccion = direccion;

    this.clientes = new List<Cliente>(); //cardinalidad mínima de cero, solo inicializar

    this.modelos = new List<Modelo>();
    //this.modelos.Add(modelo); --> tendrá que asegurarse por código
}

}
```

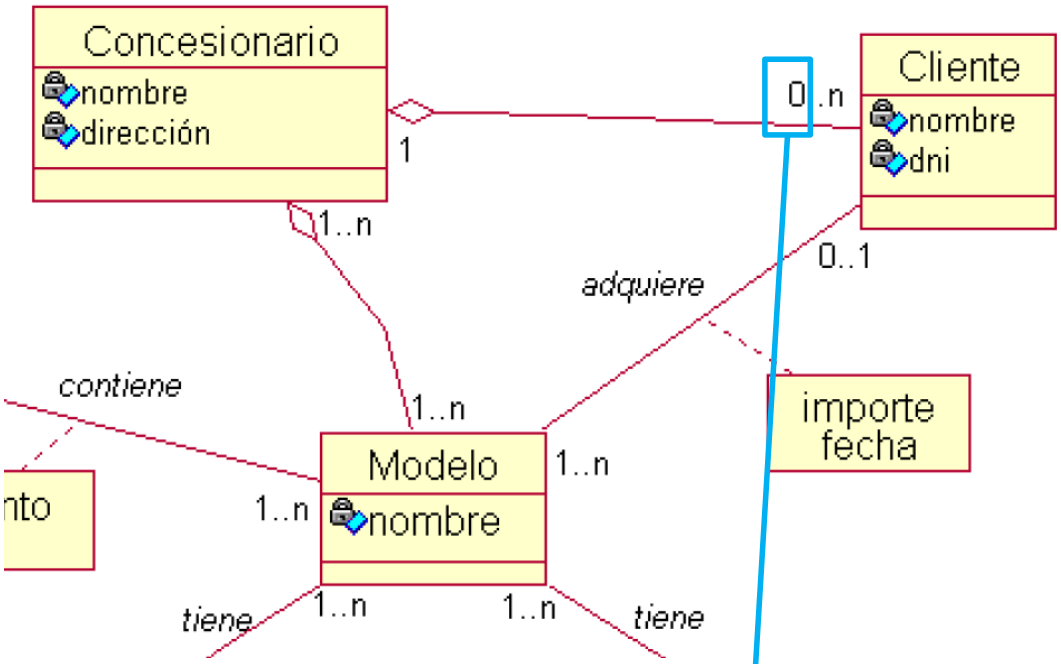
Clase Concesionario



La relación con Modelo tiene cardinalidad mínima 1 a 1
Relajaremos en esta clase, que es la que tiene la colección
En el constructor, solo inicializamos la colección

```
//Tenemos una cardinalidad mínima 1 a 1 con Modelo,  relajamos en este lado
// y eliminamos Modelo del constructor
0 referencias
public Concesionario( string nombre, string direccion){
    this.nombre = nombre;
    this.direccion = direccion;

    this.clientes = new List<Cliente>(); //cardinalidad mínima de cero, solo inicializar

    this.modelos = new List<Modelo>();
    //this.modelos.Add(modelo); --> tendrá que asegurarse por código
}

}
```

Clase Concesionario



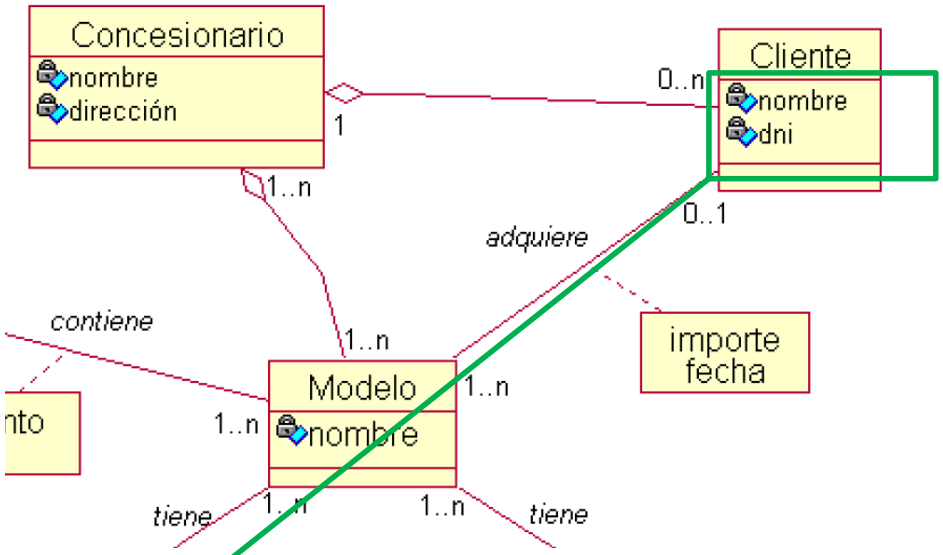
La relación con Cliente tiene cardinalidad mínima de 0

En el constructor, solo inicializamos la colección

```
    //Tenemos una cardinalidad mínima 1 a 1 con Modelo,  relajamos en este lado
    // y eliminamos Modelo del constructor
    0 referencias
    public Concesionario( string nombre, string direccion){
        this.nombre = nombre;
        this.direccion = direccion;

        this.clientes = new List<Cliente>(); //cardinalidad mínima de cero, solo inicializar

        this.modelos = new List<Modelo>();
        //this.modelos.Add(modelo); --> tendrá que asegurarse por código
    }

}
```

Clase Cliente

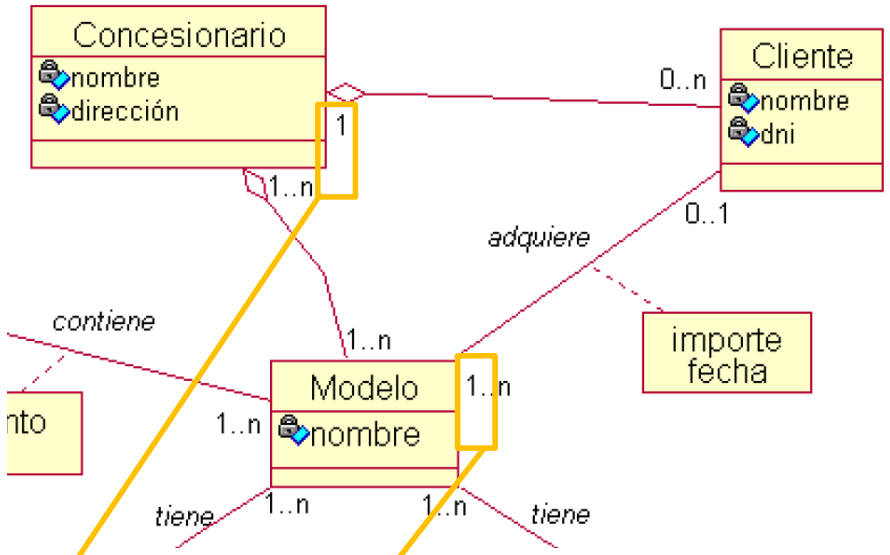
Pasamos los valores de los atributos e inicializamos




```
    //Cardinalidad minima de 1 con concesionario y con modelo, hay que pasarlos en el contructor
    0 referencias
    public Cliente(string nombre, string dni, Concesionario concesionario, Modelo modelo){
        this.nombre = nombre;
        this.dni = dni;
        this.concesionario = concesionario;

        this.modelos = new List<Modelo>();
        this.modelos.Add(modelo);

    }

}
```

Clase Cliente

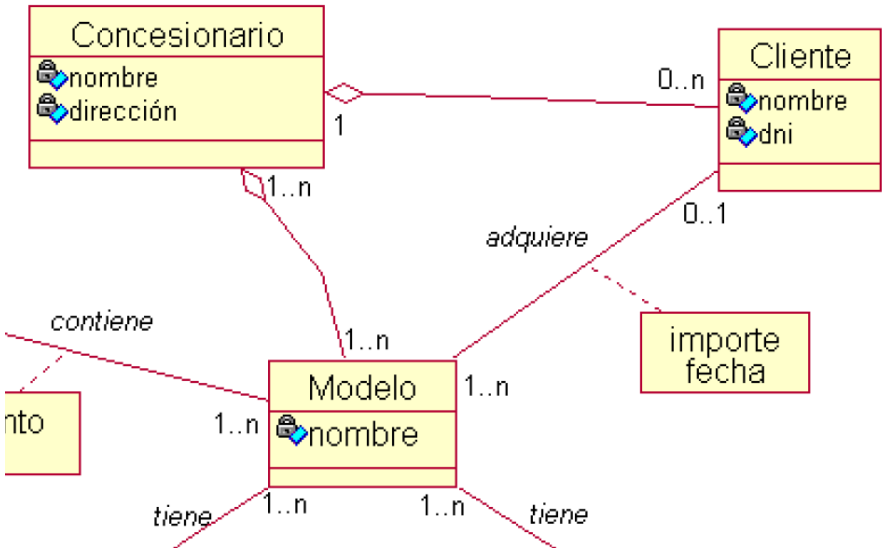
La cardinalidad mínima de 1 nos obliga a pasar al constructor un objeto de la correspondiente clase, así como añadirlo al atributo correspondiente.




```
    //Cardinalidad minima de 1 con concesionario y con modelo, hay que pasarlos en el contructor
    0 referencias
    public Cliente(string nombre, string dni, Concesionario concesionario, Modelo modelo){
        this.nombre = nombre;
        this.dni = dni;
        this.concesionario = concesionario;

        this.modelos = new List<Modelo>();
        this.modelos.Add(modelo);

    }

}
```

Clase Cliente

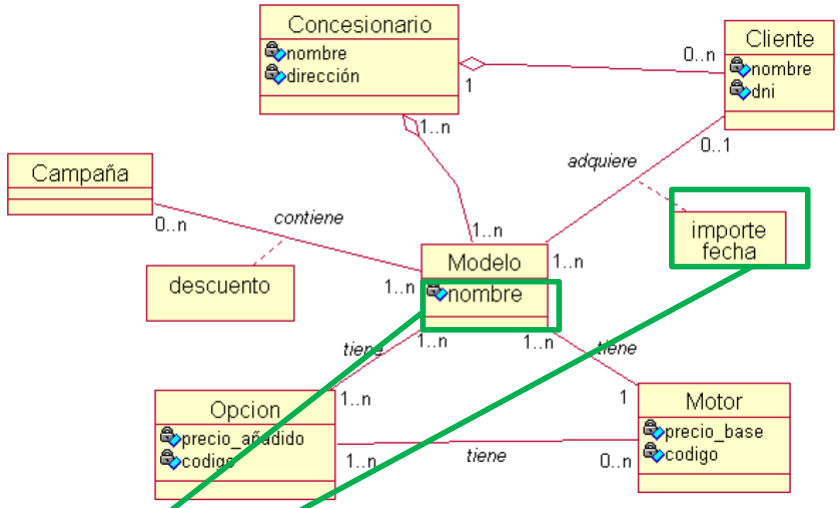
Hay que inicializar las colecciones que posee




```
    //Cardinalidad minima de 1 con concesionario y con modelo, hay que pasarlos en el contructor
    0 referencias
    public Cliente(string nombre, string dni, Concesionario concesionario, Modelo modelo){
        this.nombre = nombre;
        this.dni = dni;
        this.concesionario = concesionario;

        this.modelos = new List<Modelo>();
        this.modelos.Add(modelo);

    }

}
```

Clase Modelo

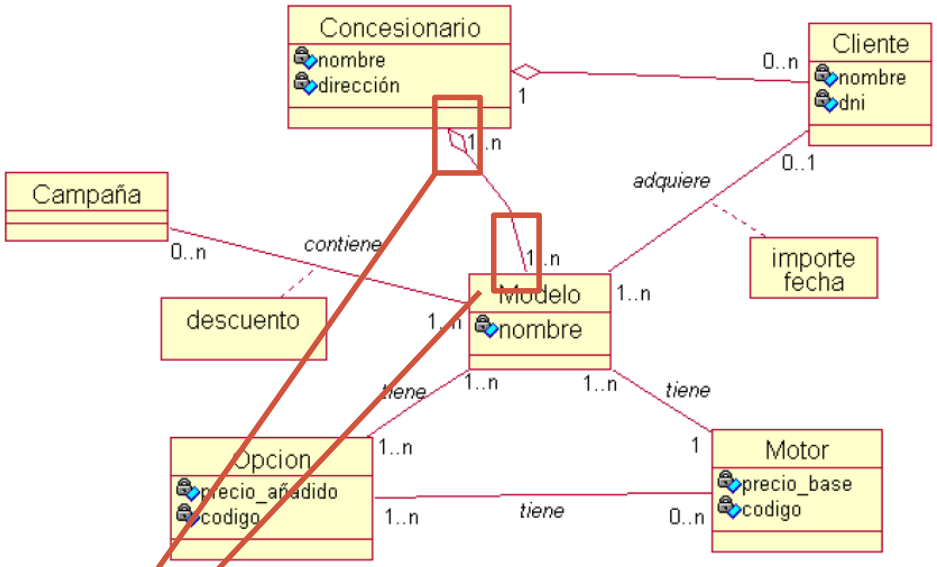
Pasamos los valores de los atributos e inicializamos. Importe y fecha vienen de los atributos del enlace




```
//Cardinalidad minima de 1 a 1 con concesionario, lo pasamos al constructor y relajamos al otro lado
//Cardinalidad mínimma de 1 a 1 con Motor, lo pasamos al contructor y relajamos al otro lado
//Cardinalidad mínima de a 1 a 1 con Opcion, lo pasamos al contructor y relajamos al otro lado
1 referencia
public Modelo(string nombre, double importe,  DateTime fecha, Concesionario concesionario, Motor motor, Opcion opcion)
{
    this.nombre = nombre;
    this.importe = importe;
    this.fecha = fecha;

    this.concesionarios = new List<Concesionario>();
    this.concesionarios.Add(concesionario);

    this.contienes = new List<Contiene>(); //cardinalidad minima de cero, solo inicializar la colección

    this.opciones = new List<Opcion>();
    this.opciones.Add(opcion);

    this.motor = motor;



}
```

Clase Modelo

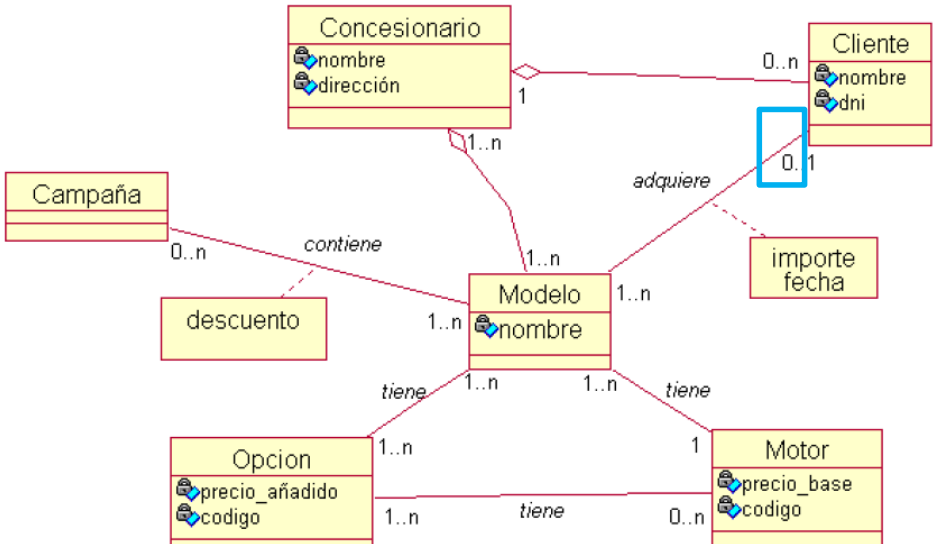
La relación con Concesionario tiene cardinalidad mínima 1 a 1.
Ya hemos relajado en concesionario.
Pasamos el objeto, inicializamos la colección y añadimos el objeto




```
//Cardinalidad minima de 1 a 1 con concesionario, lo pasam
//Cardinalidad mínimma de 1 a 1 con Motor, lo pasamos al c
//Cardinalidad mínima de a 1 a 1 con Opcion, lo pasamos al contructor y relajamos al otro lado
1 referencia
public Modelo(string nombre, double importe,  DateTime fecha, Concesionario concesionario, Motor motor, Opcion opcion)
{
    this.nombre = nombre;
    this.importe = importe;
    this.fecha = fecha;

    this.concesionarios = new List<Concesionario>();
    this.concesionarios.Add(concesionario);

    this.contienes = new List<Contiene>(); //cardinalidad minima de cero, solo inicializar la colección

    this.opciones = new List<Opcion>();
    this.opciones.Add(opcion);

    this.motor = motor;



}
```

Clase Modelo

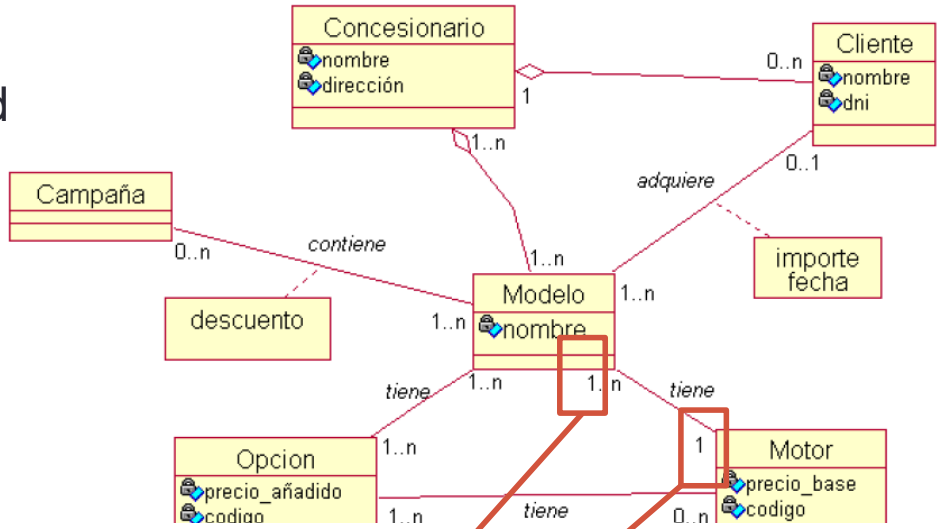
La relación con Cliente tiene cardinalidad mínima 0. No pasamos el objeto en el constructor,




```
//Cardinalidad minima de 1 a 1 con concesionario, lo pasamos al constructor y relajamos al otro lado
//Cardinalidad mínimma de 1 a 1 con Motor, lo pasamos al contructor y relajamos al otro lado
//Cardinalidad mínima de a 1 a 1 con Opcion, lo pasamos al contructor y relajamos al otro lado
1 referencia
public Modelo(string nombre, double importe,  DateTime fecha, Concesionario concesionario, Motor motor, Opcion opcion)
{
    this.nombre = nombre;
    this.importe = importe;
    this.fecha = fecha;

    this.concesionarios = new List<Concesionario>();
    this.concesionarios.Add(concesionario);

    this.contienes = new List<Contiene>(); //cardinalidad minima de cero, solo inicializar la colección

    this.opciones = new List<Opcion>();
    this.opciones.Add(opcion);

    this.motor = motor;



}
```

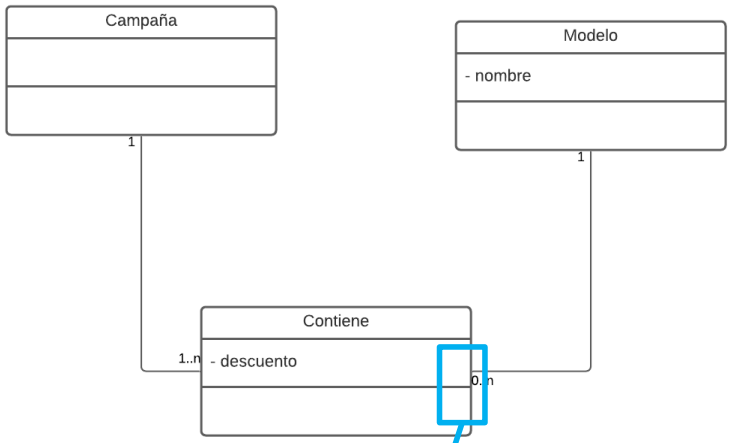
Clase Modelo

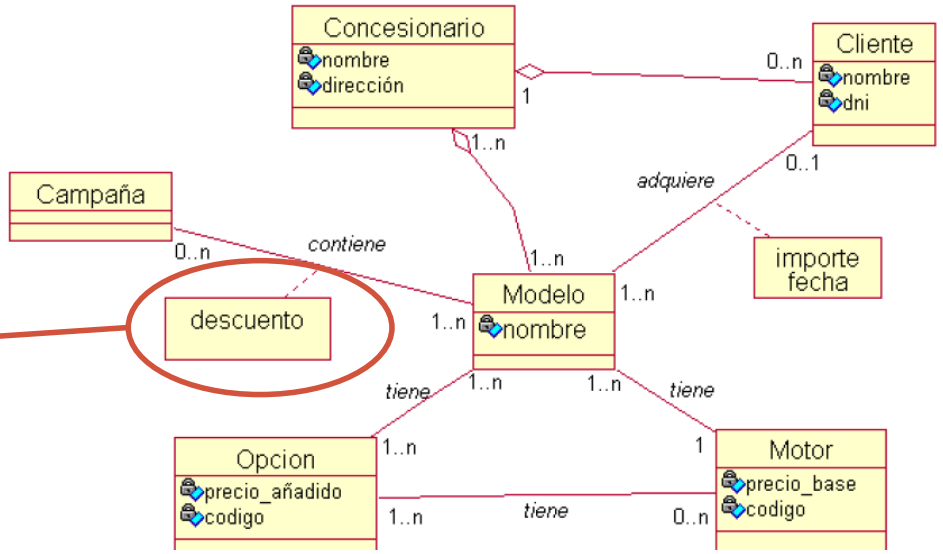
La relación con Motor tiene cardinalidad mínima 1 a 1.
Relajaremos en Motor.
Pasamos el objeto e inicializamos el atributo




```
//Cardinalidad minima de 1 a 1 con concesionario, lo pasamos al constructor y relajamos al otro lado
//Cardinalidad mínimma de 1 a 1 con Motor, lo pasamos al contructor y relajamos al otro lado
//Cardinalidad mínima de a 1 a 1 con Opcion, lo pasamos al contructor y relajamos al otro lado
1 referencia
public Modelo(string nombre, double importe,  DateTime fecha, Concesionario concesionario, Motor motor, Opcion opcion)
{
    this.nombre = nombre;
    this.importe = importe;
    this.fecha = fecha;

    this.concesionarios = new List<Concesionario>();
    this.concesionarios.Add(concesionario);

    this.contienes = new List<Contiene>(); //cardinalidad minima de cero, solo inicializar la colección

    this.opciones = new List<Opcion>();
    this.opciones.Add(opcion);

    this.motor = motor;


}
```

Clase Modelo

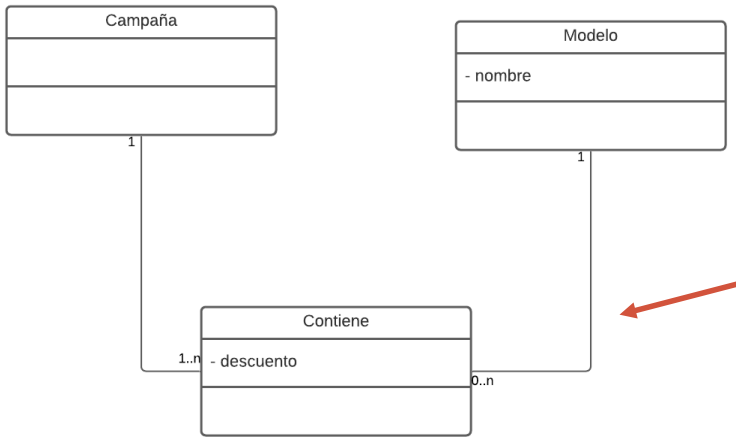


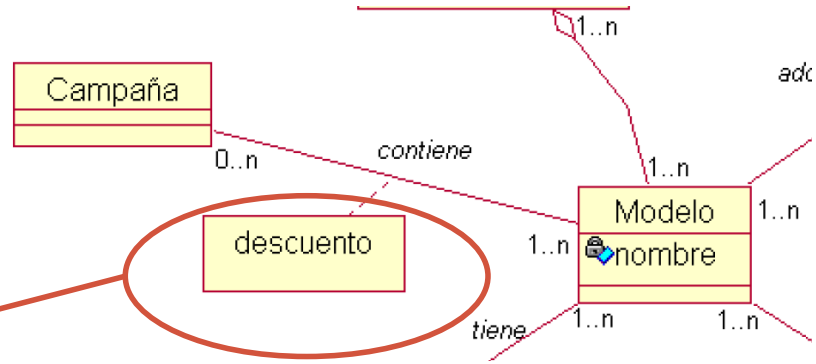



```
//Cardinalidad minima de 1 a 1 con concesionario, lo pasamos al constructor y relajamos al otro lado
//Cardinalidad mínimma de 1 a 1 con Motor, lo pasamos al contructor y relajamos al otro lado
//Cardinalidad mínima de 1 a 1 con Opcion, lo pasamos al contructor y relajamos al otro lado
1 referencia
public Modelo(string nombre, double importe,  DateTime fecha, Concesionario concesionario, Motor motor, Opcion opcion)
{
    this.nombre = nombre;
    this.importe = importe;
    this.fecha = fecha;

    this.concesionarios = new List<Concesionario>();
    this.concesionarios.Add(concesionario);

    this.contienes = new List<Contiene>(); //cardinalidad minima de cero, solo inicializar la colección

    this.opciones = new List<Opcion>();
    this.opciones.Add(opcion);

    this.motor = motor;


}
```

La relación con Contiene tiene cardinalidad mínima 0. No pasamos el objeto en el constructor e inicializamos la colección

Clase Contiene

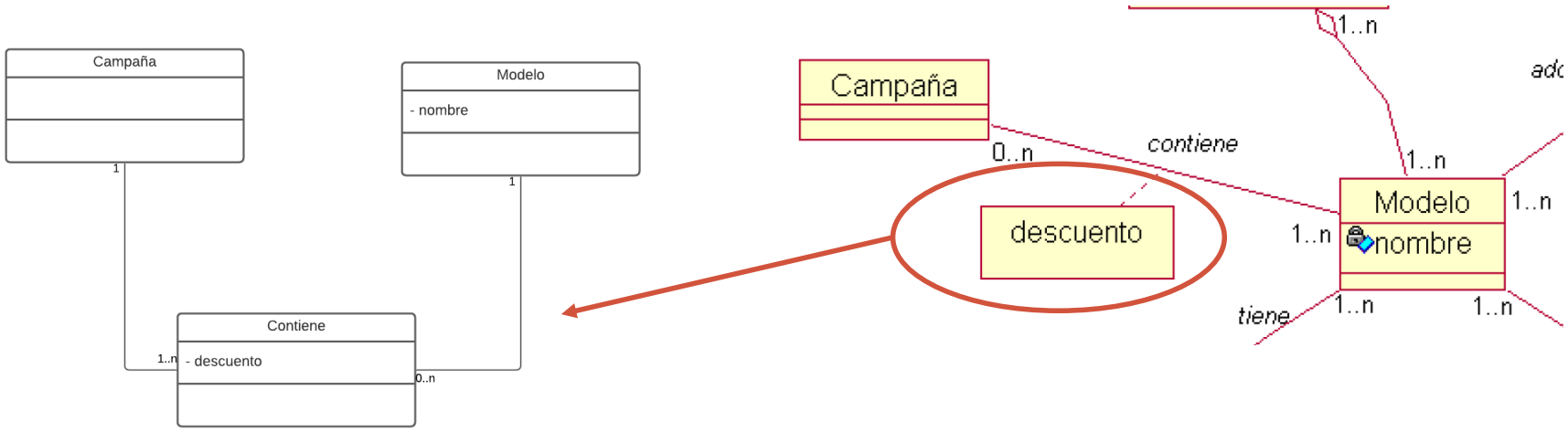




El atributo de la asociación pasa a ser un atributo de la nueva clase

```
6 referencias
public class Contiene{
    //Esta clase surge al promocionar un atributo de una relacion N a N a clase
    private double descuento;

    private Modelo modelo;
    private Campanya campanya;

    0 referencias
    public Contiene(double descuento, Modelo modelo, Campanya campanya){
        this.descuento = descuento;
        this.modelo = modelo;
        this.campanya = campanya;
    }
}
```

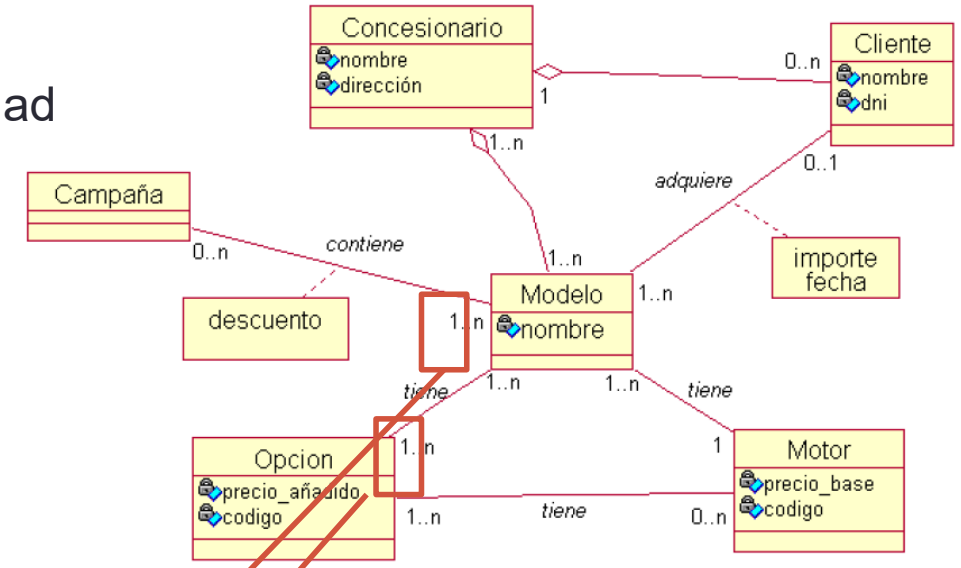
Clase Campaña




```
public class Campanya{

    private ICollection <Contiene> contienes; //El atributo de enlace en una relación N a N promociona a clase

    //Cadinalidad minima de 1 a 1 con Contiene, relajamos en este lado, eliminando del constructor
    0 references | 0 changes | 0 authors, 0 changes
    public Campanya(){
        this.contienes = new List <Contiene>();
        //this.contienes.Add(contiene); //Tendrá que asegurarse por código
    }

}
```

Clase Modelo

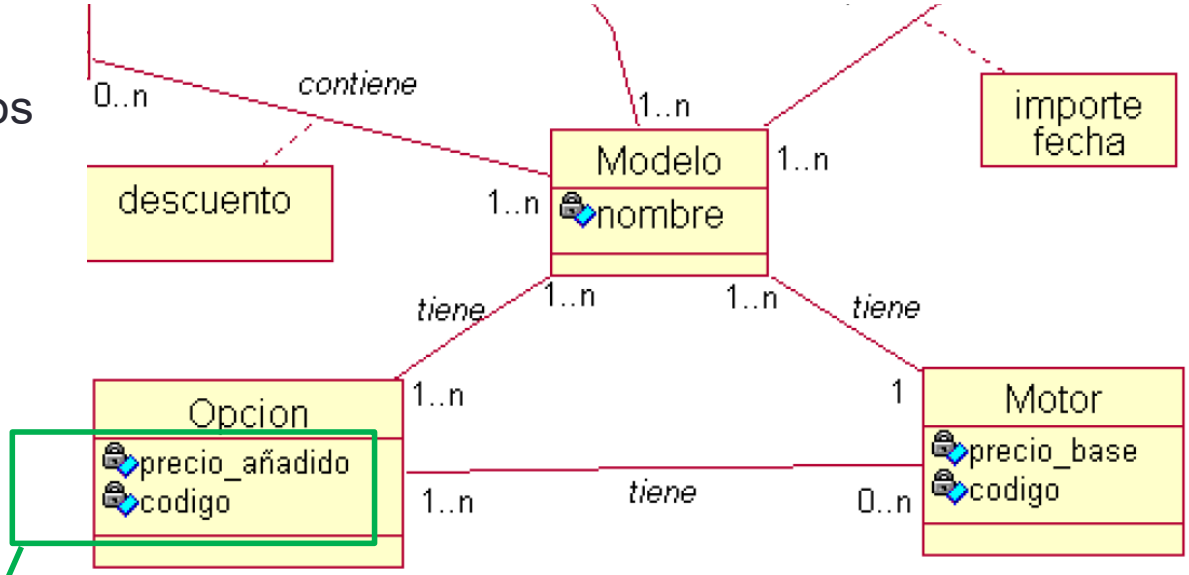
La relación con Opción tiene cardinalidad mínima 1 a 1.
Relajaremos en Opción.
Pasamos el objeto, inicializamos la colección y añadimos el objeto




```
0 referencias
public Modelo(string nombre, Concesionario concesionario, Motor motor, Opcion opcion)
{
    this.nombre = nombre;

    this.concesionarios = new List<Concesionario>();
    this.concesionarios.Add(concesionario);

    this.contienes = new List<Contiene>(); //cardinalidad minima de cero, solo inicializar la colección

    this.opciones = new List<Opcion>();
    this.opciones.Add(opcion);

    this.motor = motor;


}
}
```

Clase Opción

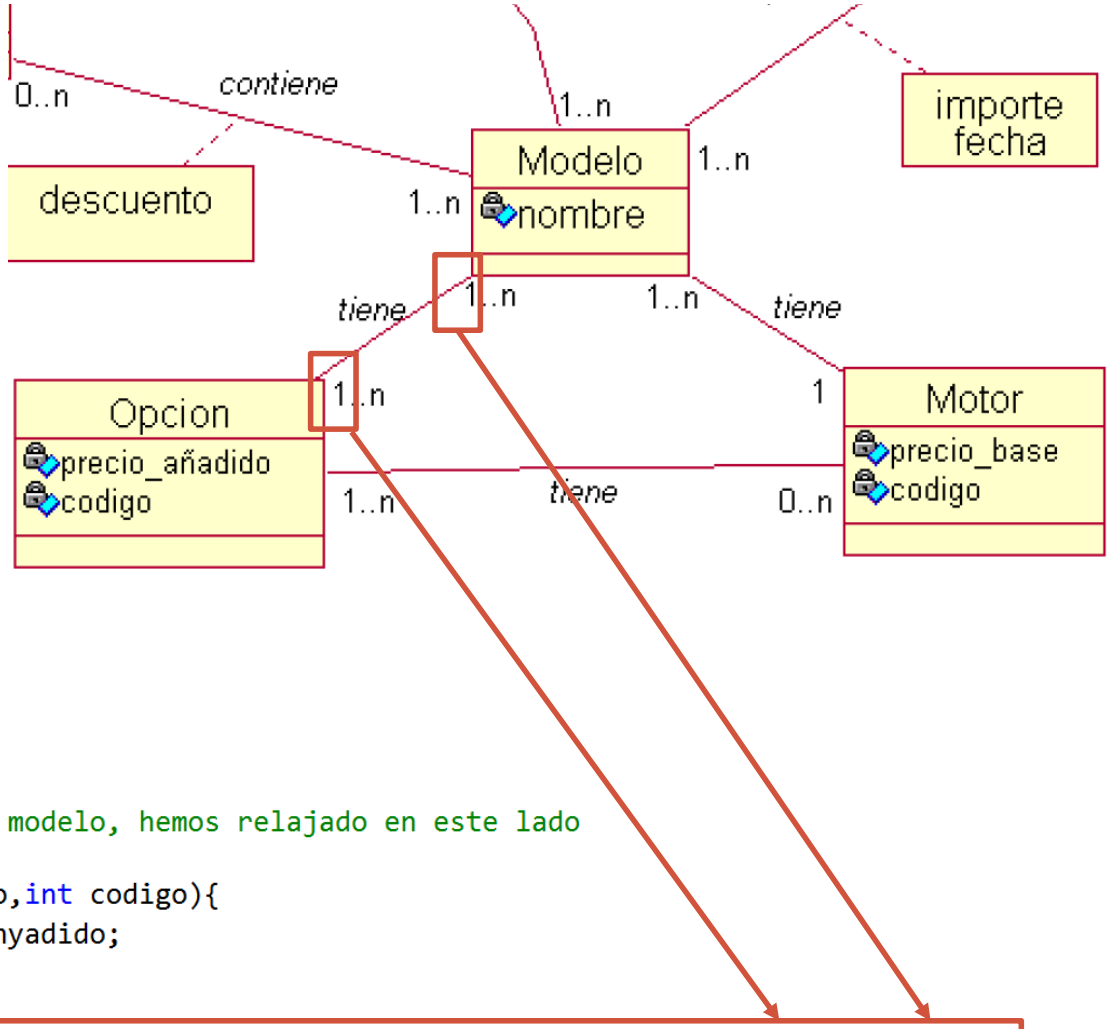
Pasamos los valores de los atributos e inicializamos




```
//Cardinalidad mínimma de 1 a 1 con modelo, hemos relajado en este lado
0 referencias
public Opcion(double precio_anyadido,int codigo){
    this.precio_anyadido = precio_anyadido;
    this.codigo = codigo;

    this.modelos = new List<Modelo>();
    //This.modelo.Add(modelo) ---> tendrá que asegurarse por código. Hemos relajado en este lado

    this.motores = new List<Motor>(); //cardinalidad minima de cero, solo inicializar la colección
}
}
```

Clase Opción

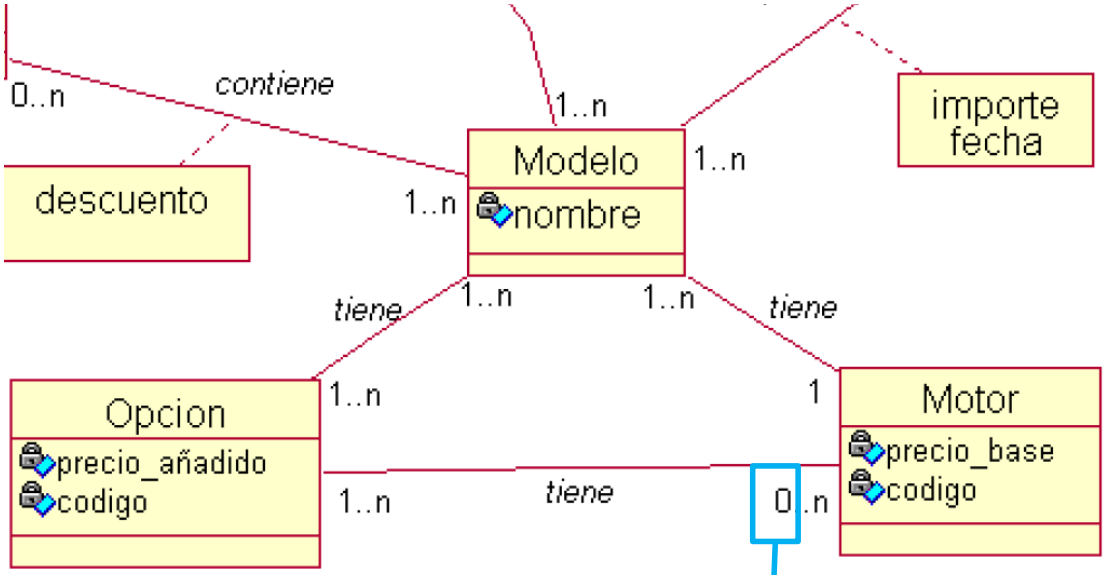
La relación con Modelo tiene cardinalidad mínima 1 a 1. Relajaremos en este lado.
No pasamos el objeto y solo inicializamos la colección




```
//Cardinalidad mínimma de 1 a 1 con modelo, hemos relajado en este lado
0 referencias
public Opcion(double precio_anyadido,int codigo){
    this.precio_anyadido = precio_anyadido;
    this.codigo = codigo;

    this.modelos = new List<Modelo>();
    //This.modelo.Add(modelo) ---> tendrá que asegurarse por código. Hemos relajado en este lado

    this.motores = new List<Motor>(); //cardinalidad minima de cero, solo inicializar la colección
}
}
```

Clase Opción

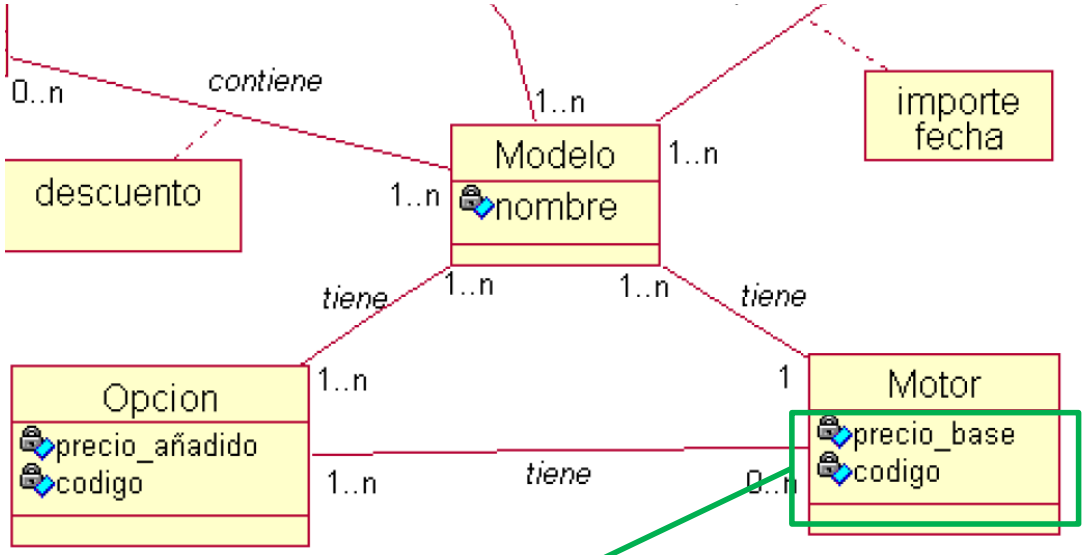
La relación con Motor tiene cardinalidad mínima 0.
No pasamos el objeto en el constructor e inicializamos la colección




```
//Cardinalidad mínimma de 1 a 1 con modelo, hemos relajado en este lado
0 referencias
public Opcion(double precio_anyadido,int codigo){
    this.precio_anyadido = precio_anyadido;
    this.codigo = codigo;

    this.modelos = new List<Modelo>();
    //This.modelo.Add(modelo) ---> tendrá que asegurarse por código. Hemos relajado en este lado

    this.motores = new List<Motor>(); //cardinalidad minima de cero, solo inicializar la colección
}
}
```

Clase Motor

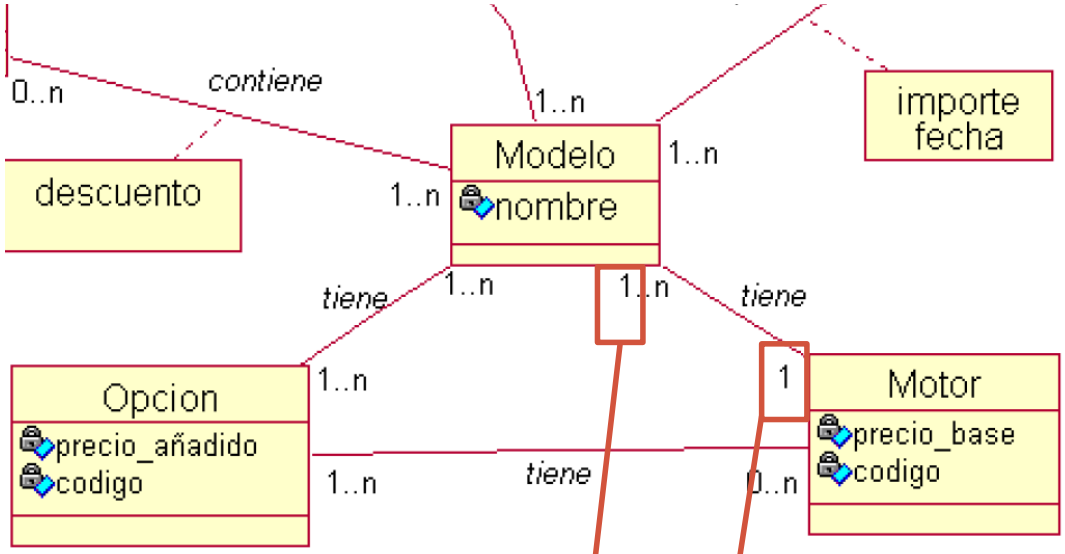
Pasamos los valores de los atributos e inicializamos




```
    //Cardinalidad mínimma de 1 a 1 con Modelo, hemos relajado en este lado
    0 referencias
    public Motor(double precio_base, int codigo, Opcion opcion){
        this.precio_base = precio_base;
        this.codigo = codigo;

        this.opciones = new List<Opcion>();
        this.opciones.Add(opcion); //carddinalidad minima de 1

        this.modelos = new List<Modelo>();
        //this.modelos.Add(modelo); --> tendrá que asegurarse por código. Hemos relajado en este lado
    }
}
```

Clase Motor

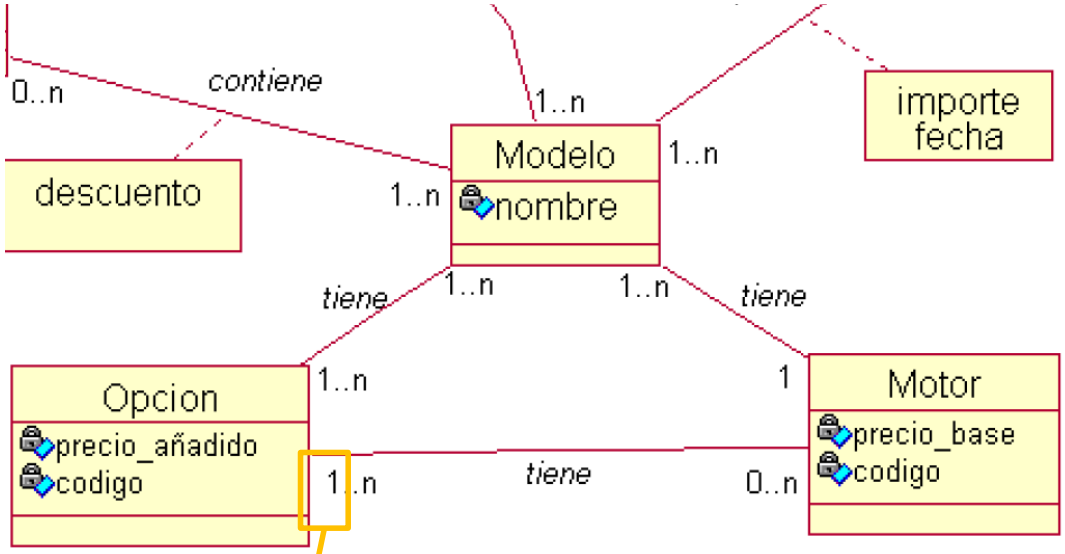
La relación con Modelo tiene cardinalidad mínima 1 a 1. Relajaremos en este lado.
No pasamos el objeto e inicializamos la colección




```
    //Cardinalidad mínimma de 1 a 1 con Modelo, hemos relajado en este lado
    0 referencias
    public Motor(double precio_base, int codigo, Opcion opcion){
        this.precio_base = precio_base;
        this.codigo = codigo;

        this.opciones = new List<Opcion>();
        this.opciones.Add(opcion); //carddinalidad minima de 1

        this.modelos = new List<Modelo>();
        //this.modelos.Add(modelo); --> tendrá que asegurarse por código. Hemos relajado en este lado
    }
}
```

Clase Motor

La relación con Opción tiene cardinalidad mínima 1. Pasamos el objeto, inicializamos la colección y añadimos el objeto




```
    //Cardinalidad mínimma de 1 a 1 con Modelo, hemos relajado en este lado
    0 referencias
    public Motor(double precio_base, int codigo, Opcion opcion){
        this.precio_base = precio_base;
        this.codigo = codigo;

        this.opciones = new List<Opcion>();
        this.opciones.Add(opcion); //carddinalidad minima de 1

        this.modelos = new List<Modelo>();
        //this.modelos.Add(modelo); --> tendrá que asegurarse por código. Hemos relajado en este lado
    }
}
```

INSTANCIACIÓN DEL SISTEMA

Añadimos las instrucciones para cumplir las restricciones por código

```
0 referencias
class Program
{
    0 referencias
    static void Main(string[] args)
    {
        Concesionario valenciaMotor = new Concesionario("Valencia Motor, S.A", "Avda. Tres Cruces, Valencia");
        Opcion opcionBlue = new Opcion(3500, 1);
        Motor motor93 = new Motor(2000, 1, opcionBlue);


        Modelo c3 = new Modelo("Citroën c3",12000,new DateTime(2020,10,30), valenciaMotor, motor93, opcionBlue);
        opcionBlue.AddModelo(c3);//Aseguramos por código la cardinalidad mínima que habíamos relajado
        motor93.AddModelo(c3); //Aseguramos por código la cardinalidad mínima que habíamos relajado
        Cliente cliente = new Cliente("sole", "11111111G", valenciaMotor, c3);
        c3.setCliente(cliente);  //Dejamos consistente el modelo

        valenciaMotor.AddCliente(cliente); //Dejamos consistente el modelo
        valenciaMotor.AddModelo(c3);  ///Aseguramos por código la cardinalidad mínima que habíamos relajado

        Campanya campanya = new Campanya();
        Contiene descuentoCampanya = new Contiene(0.10, c3, campanya);
        campanya.AddContiene(descuentoCampanya); //Aseguramos por código la cardinalidad mínima que habíamos relajado
        c3.AddContiene(descuentoCampanya); //Dejamos consistente el modelo




        |

    }
}
```