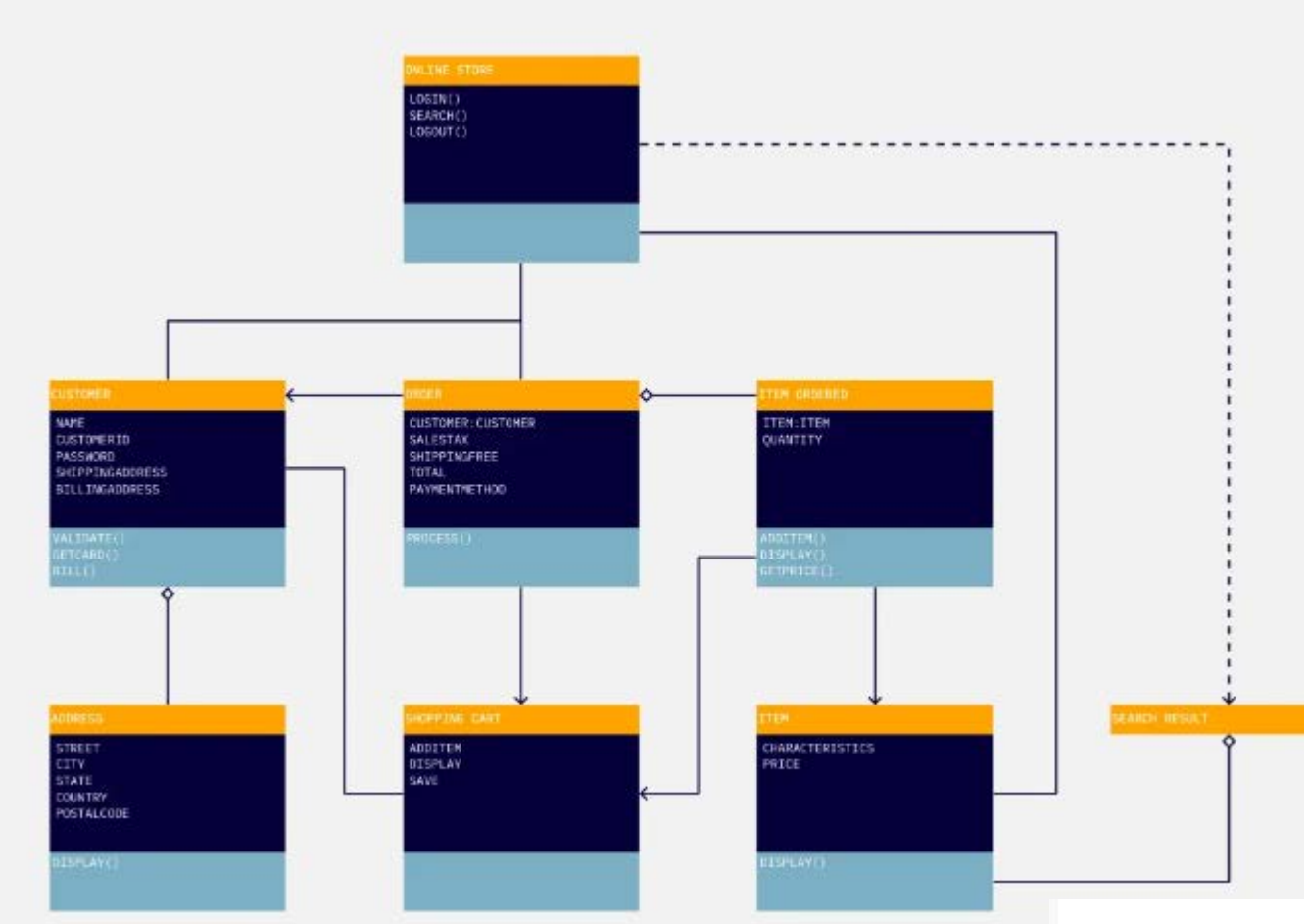
EFA
MORATALAZ

1º CFGS Desarrollo de Aplicaciones Multiplataforma

ENTORNOS DE DESARROLLO

YOLANDA MORENO Gª-MAROTO

UT4 – DIAGRAMAS DE CLASES



UNIFIED MODELING LANGUAGE™

EFA
MORATALAZ

1º Desarrollo Aplicaciones Multiplataforma

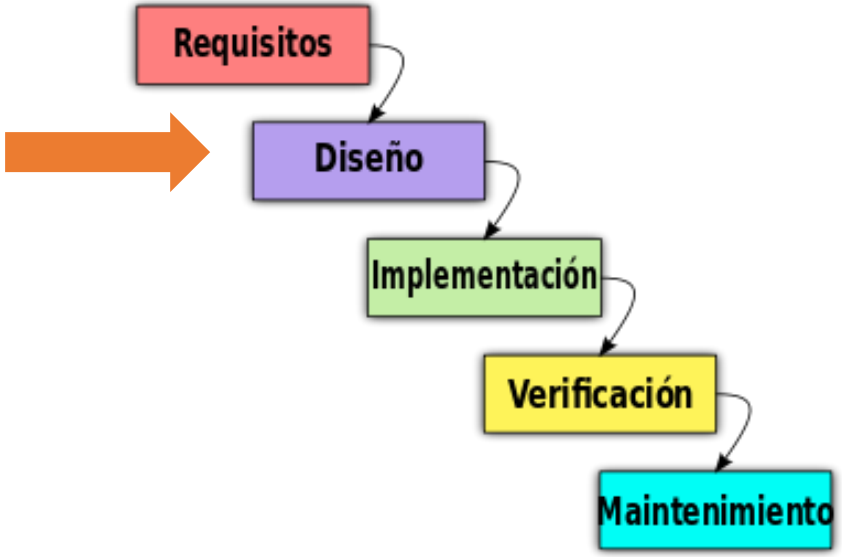
ENTORNOS DE DESARROLLO

INDICE

UT4 – DIAGRAMAS DE CLASES

INTRODUCCIÓN A UML

INTRODUCCIÓN

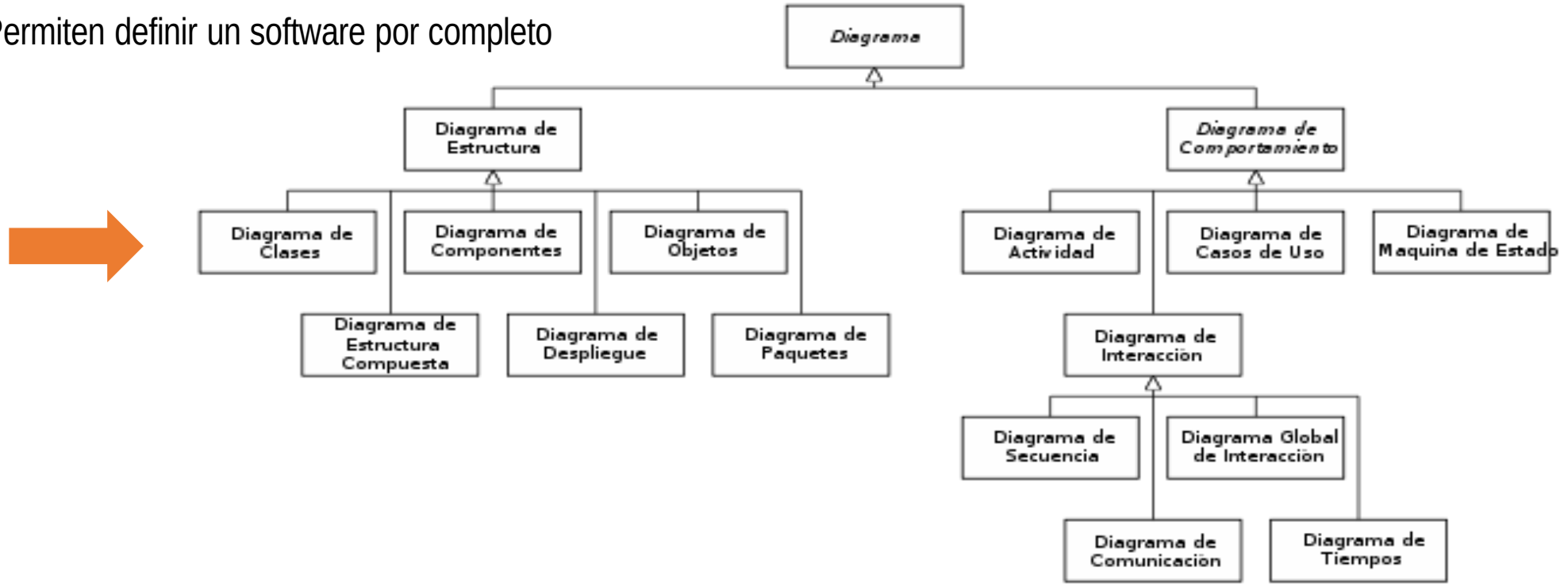


- Antiguamente, **inexistencia de diseño universal** y entendible para todos.
- Cada diseñador realizaba sus diagramas a su manera.
- Esto suponía un problema si varias personas tenían que entender los diagramas que uno o varios diseñadores les presentaban para definir el software

SOLUCIÓN:

¿Qué es UML?

- UML es un **conjunto unificado de estándares** utilizado para las diferentes necesidades y usos que un diseñador pudiera tener a la hora de plantear el programa de forma gráfica.
- Actualmente **versión 2.0.**
- No es necesario realizar todos los diagramas
- Permiten definir un software por completo



DISEÑO DE CLASES EN UML

Diseño de Clases: clases UML

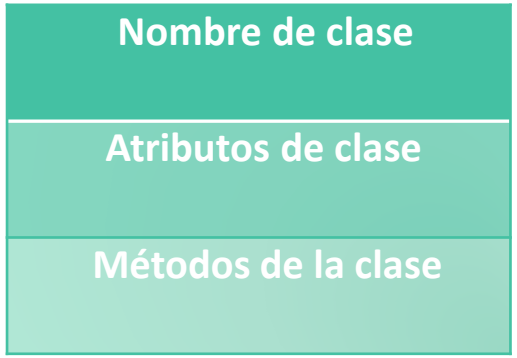
¿Qué es una clase?
Descriptor de un conjunto de objetos que **comparten** los mismos **atributos**, **métodos**, **relaciones** y **comportamiento**.

- Son las "cosas" importantes desde una visión particular que son utilizadas por los programadores/usuarios para describir el problema o la solución.
- Es una abstracción de la realidad.
- Cada una de las cosas tienen **propiedades** y **comportamientos**.
- Cada cosa sirve como **plantilla** para crear objetos.

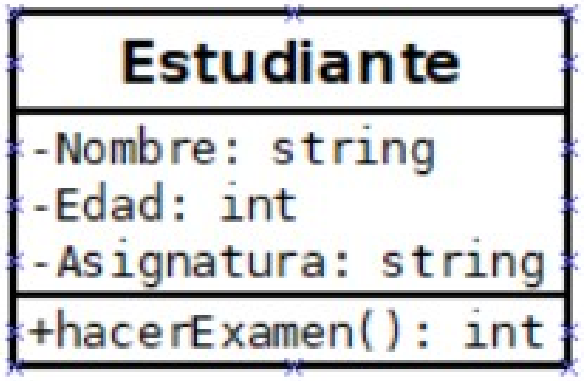
EXISTEN RELACIONES ENTRE ESAS COSAS

Diseño de Clases: clases UML

Las clases en un diagrama UML se representan en forma de caja, delimitada por 3 cajones.

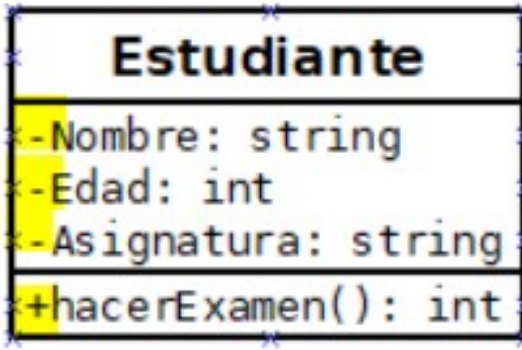


- Nombre que **identifica** a la clase.
- Propiedades y **características** que es compartida por todos los objetos de la clase.
- Acciones y **comportamientos** que realiza el objeto.



Clase: *Estudiante*.
Atributos: *Nombre:string, Edad:int, Asignatura:string.*
Métodos: *hacerExamen*().

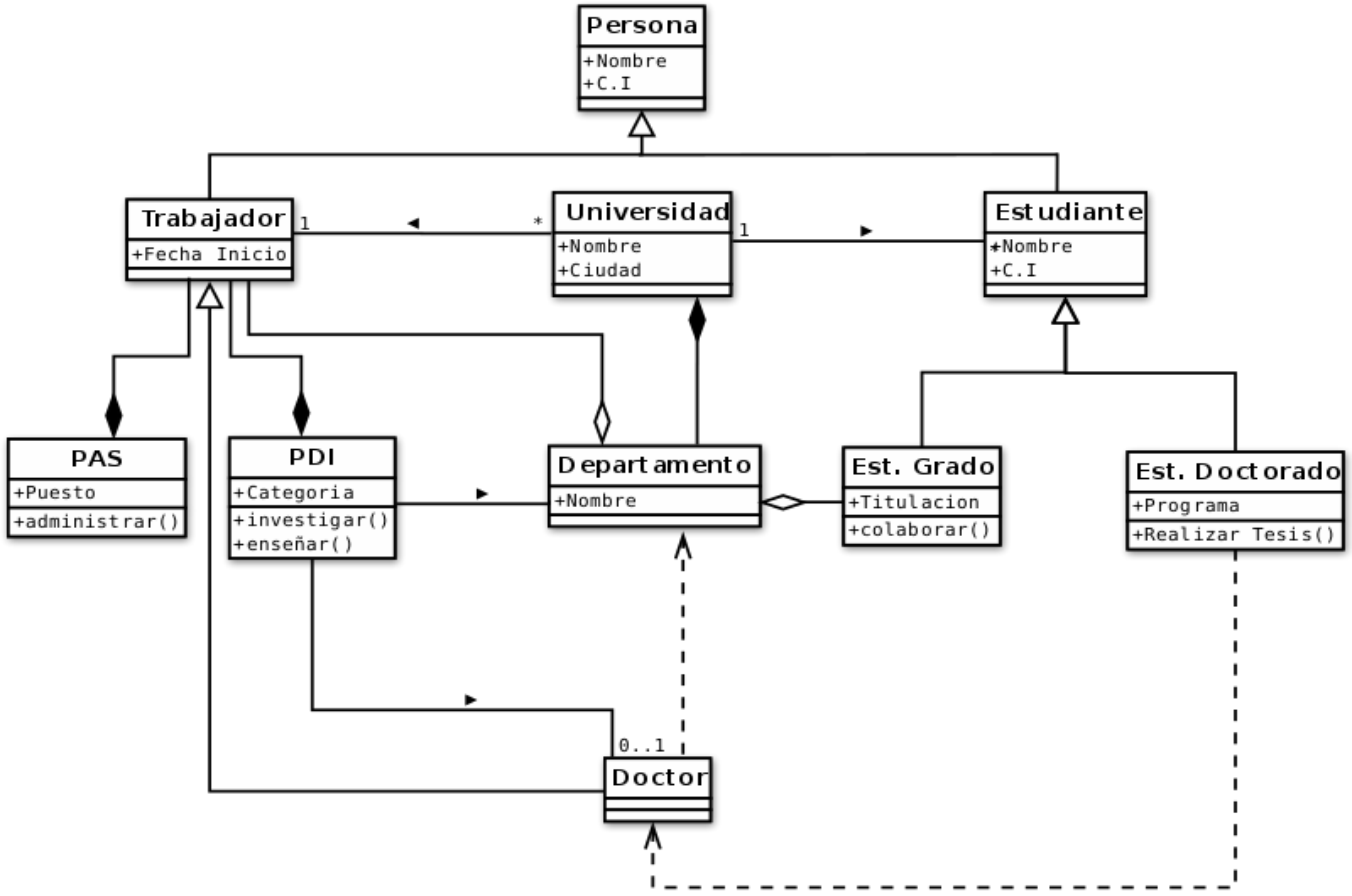
Diseño de Clases: modificadores de acceso



VISIBILIDAD	JAVA	SÍMBOLO	SIGNIFICADO
Público	public	**+**	El elemento es visible desde **dentro y fuera de la clase.**
Privado	private	**-**	El elemento solo es visible desde **dentro de la clase.**
Protegido	protected	**#**	El elemento **no es accesible desde fuera** de la clase. Pero puede ser manipulado por los demás métodos de **la clase o subclases** (clases que hereden de ella).

Diseño de Clases: diagrama de clases

Uno de los diagramas **más utilizados** dentro de UML para comprender el **alcance** y la **estructura general** de un **software**.



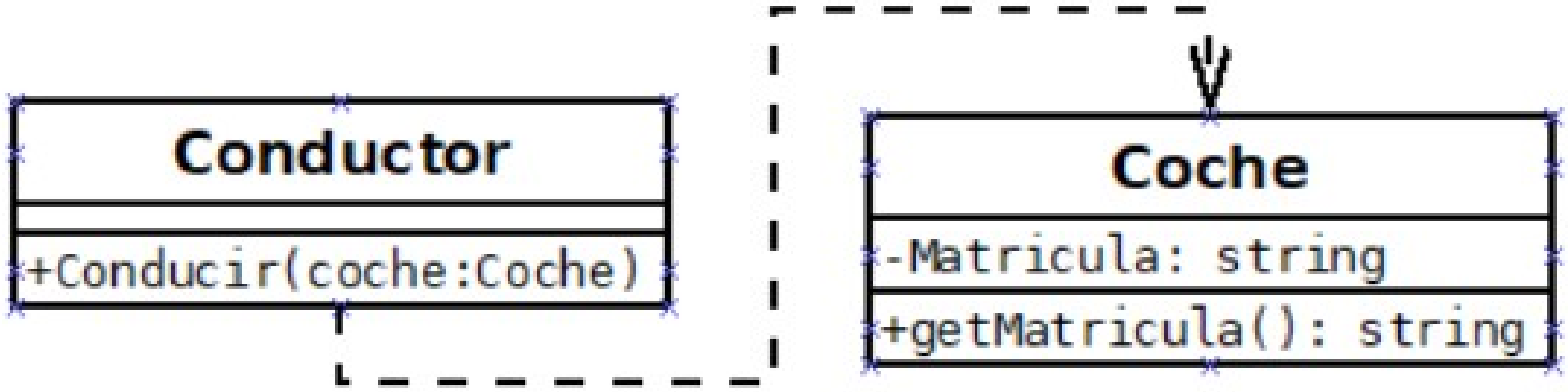
En los diagramas de clases **no** suelen aparecer:

- constructores.
- get/set.

Diseño de Clases: tipos de relaciones

- **Dependencia**: Una **clase requiere de otra** para proporcionar alguno de sus servicios. Un cambio en una clase puede afectar a otra.
  - La relación entre clases dura hasta que el método termine.

USA/UTILIZA A

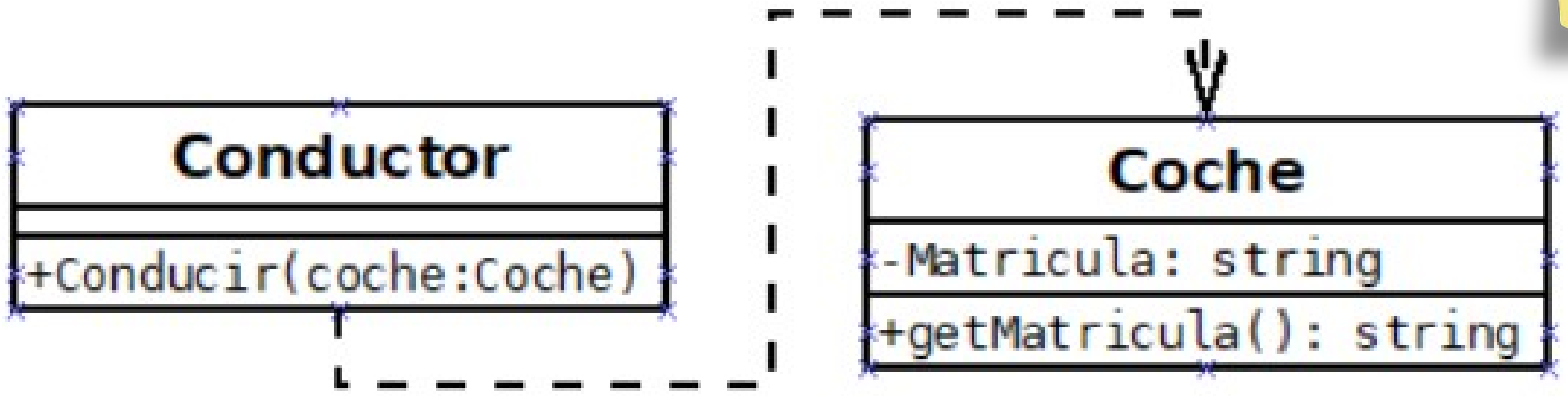


- Se usa cuando:
  - Una clase utiliza a objetos de otra clase como PARÁMETROS de una operación.
  - Una clase utiliza **objetos** de **otra** clase en alguno de sus **métodos**.

Diseño de Clases: tipos de relaciones

*"La clase Conductor **utiliza** la clase Coche."*

La clase **independiente** (Coche) no tiene conocimiento de la clase **dependiente** (Conductor)



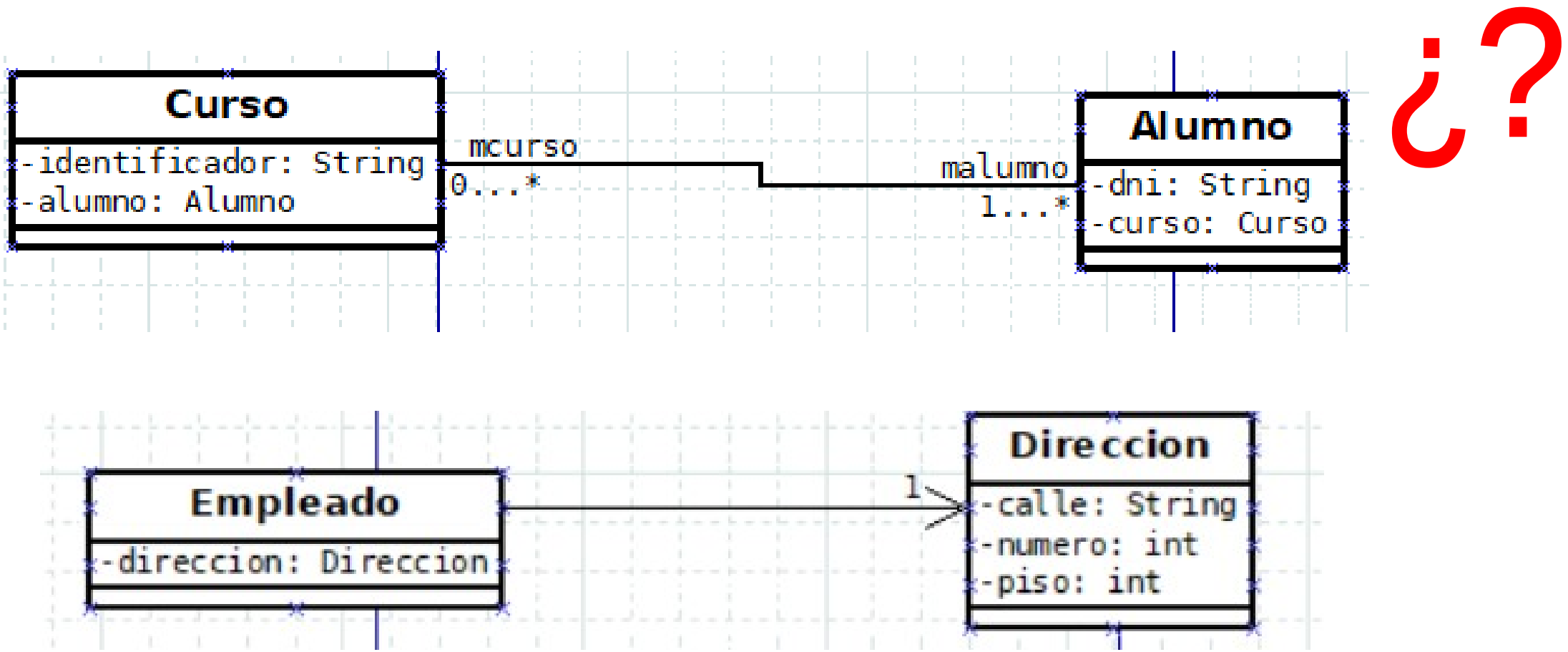
Diseño de Clases: tipos de relaciones

- **Asociación**: especifica que los **objetos de una clase están conectados con los objetos de otra**.
  - Las asociaciones entre 2 clases se llaman **binarias**.
    - Pueden existir asociaciones de N clases, siendo N>2.
  - Las asociaciones pueden ser:
    - **Bidireccionales.**
    - **Unidireccionales.**
  - En las relaciones de asociación **AMBAS** clases dependen una de la otra. En las relaciones de dependencia y generalización, la clase independiente no tiene conocimiento de la clase dependiente.

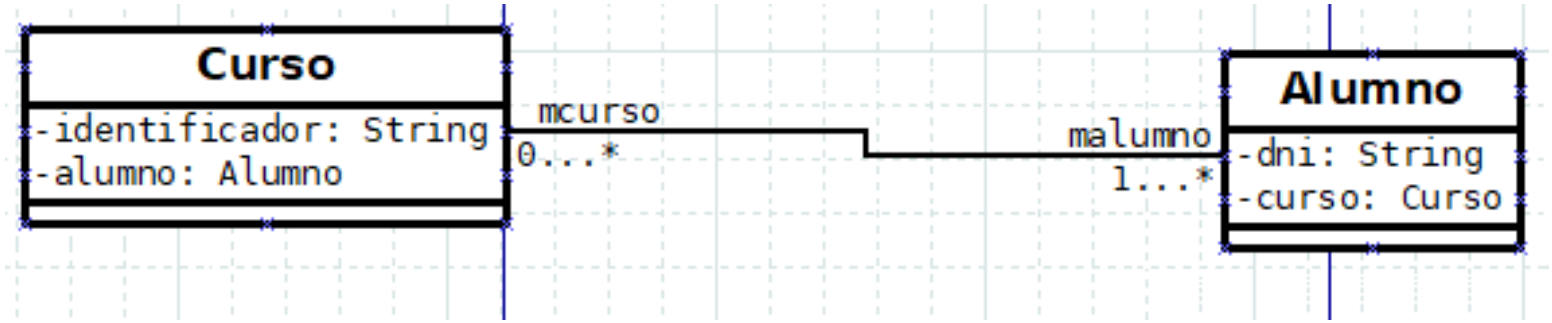
Verbo distinto a los vistos pe: estudia, cursa, tiene…

- Se usa cuando:
  - Es necesario navegar desde los objetos de una clase hacia los de otra.
  - Cuando en una clase existe un ATRIBUTO de la otra clase.

Diseño de Clases: tipos de relaciones



Diseño de Clases: tipos de relaciones



Ambas clases pueden depender la una de la otra

En un curso hay desde 1 a varios alumnos; y un alumno está en 0 o varios cursos.

Un empleado tiene una dirección.

Diseño de Clases: tipos de relaciones

➢ **Cardinalidad.**

Las relaciones se representan mediante flechas y la cardinalidad.

- Las **flechas** pueden ser de diferentes formas indicando el tipo de relación.
- La **cardinalidad** es un símbolo que representa el número de elementos de cada clase implicados en la relación.
- El **rol** indica la función de la clase en el contexto concreto.

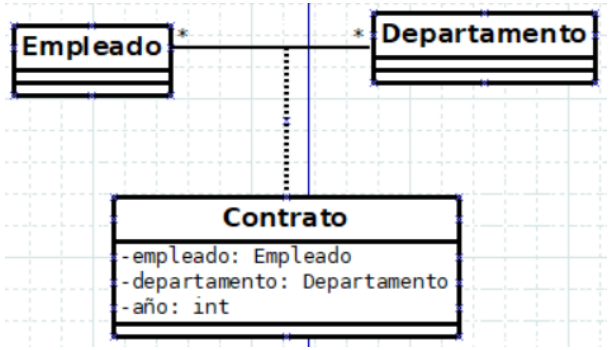
Cardinalidad	Significado
1	Uno y sólo uno
0…1	Cero o uno
N….M	Desde N hasta M
*	Cero o varios
0…*	Cero o varios
1…*	Uno o varios (al menos uno)

➢ **Clase de ASOCIACIÓN.**

Es una relación que surge cuando hay multiplicidad de **muchos a muchos** entre dos clases.

- Se **extraen los objetos involucrados** (objetos de las clases) y se ponen como **atributos dentro de la clase de asociación**.

- Se puede añadir **información común** de estos objetos a la clase en forma de atributos.

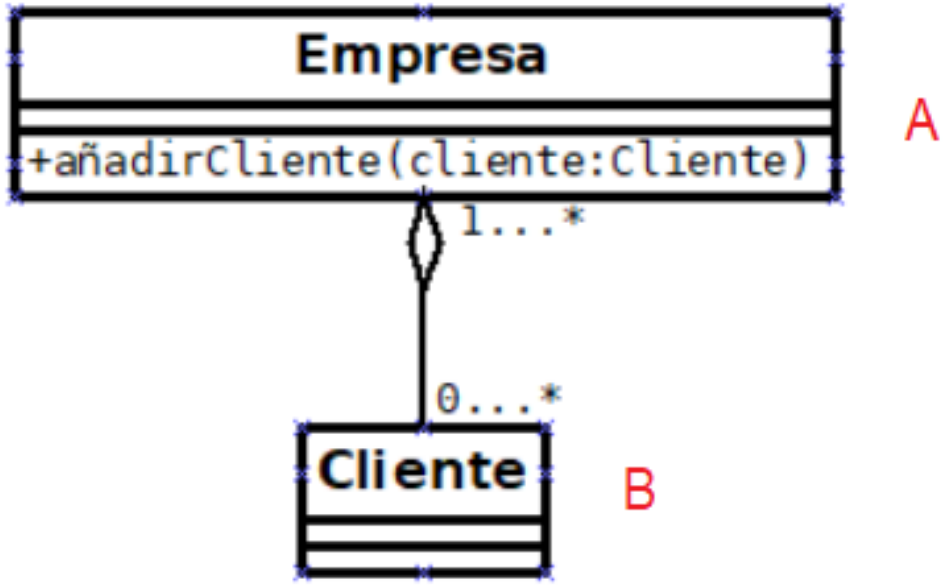


Diseño de Clases: tipos de relaciones

- **Agregación**:

Es un caso particular de asociación. Presenta a una entidad como agregado de partes.

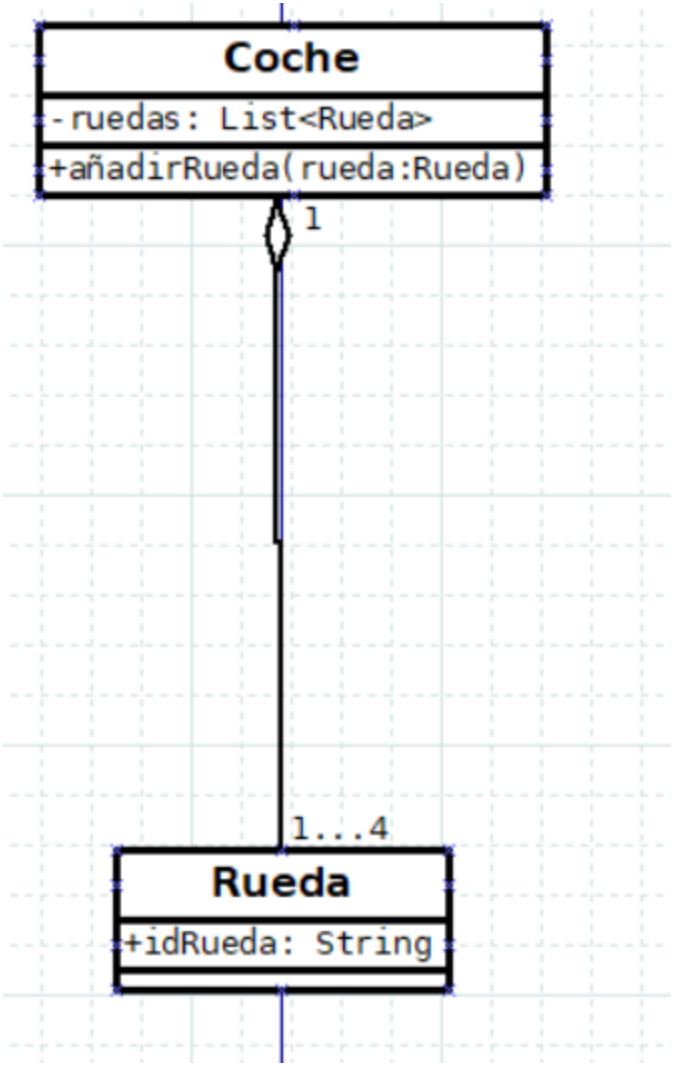
  - Los objetos de la clase B son **independientes** de los objetos de la clase A.
  - Si un objeto de la clase A desaparece, el objeto B **sigue existiendo**.
  - Los objetos de la clase B **pueden pertenecer a otras clases** distintas a la clase A.
  - Es necesario un método de la clase A para agregar objetos de la clase B.



AGRUPA/TIENE/ESTÁ COMPUESTO

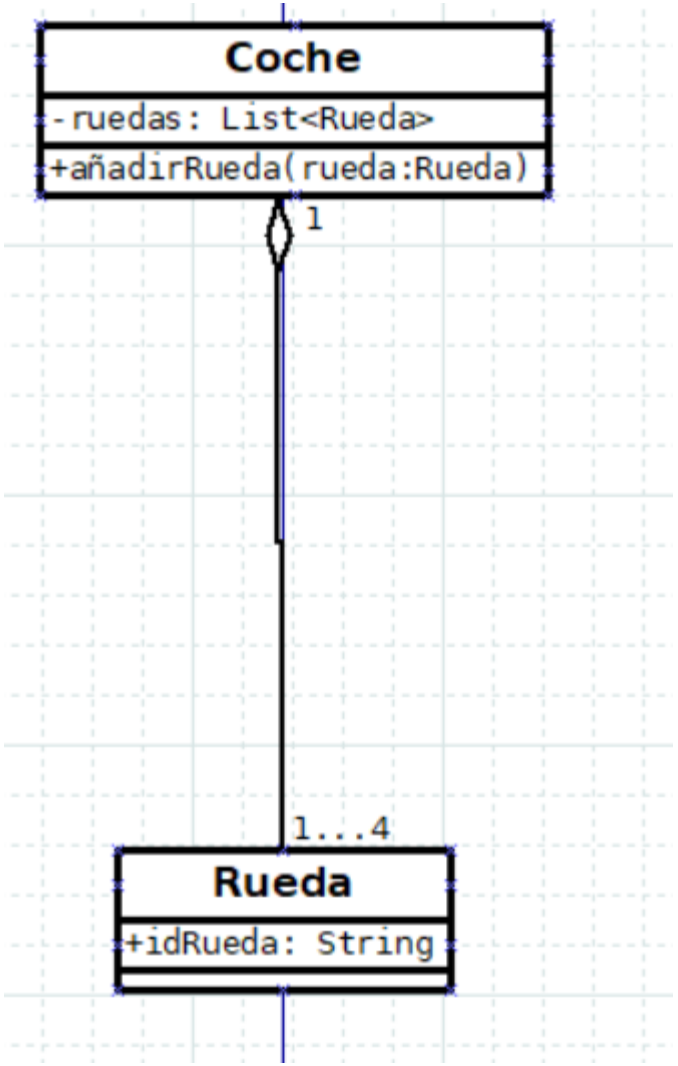
Diseño de Clases: tipos de relaciones

¿?



Representa relaciones todo/partes, compuesto/componentes.

Diseño de Clases: tipos de relaciones



Un coche tiene entre 1 y 4 ruedas.

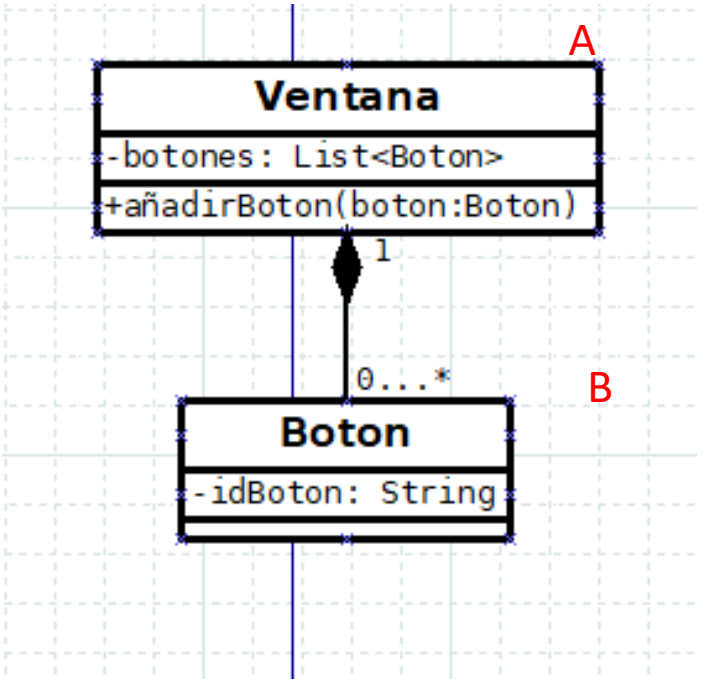
Una rueda puede ponerse en 1 coche.

Diseño de Clases: tipos de relaciones

- **Composición**:

Es un caso particular de asociación. Presenta a una entidad como agregado de partes.

- Los objetos de la clase B son **dependientes** de los objetos de la clase A.
- Si un objeto de la clase A desaparece, el objeto B **no puede seguir existiendo**.
- Los objetos de la clase B **no pueden pertenecer a otras clases** distintas a la clase A.
- Es necesario un método de la clase A para agregar objetos de la clase B.



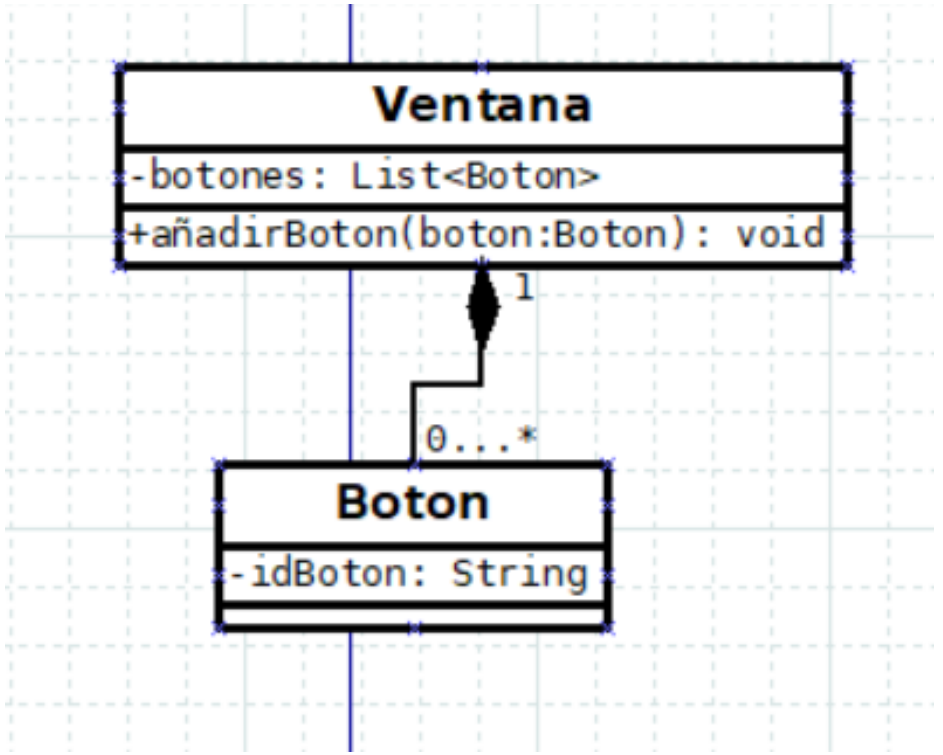
AGRUPA/ESTAR FORMADO/ESTAR COMPUESTO

¿?

Diseño de Clases: tipos de relaciones

Si la clase independiente se quita, no pasa nada.

Si la parte dependiente se quita, la independiente deja de existir.



*Una ventana **tiene** de 0 a varios botones.*

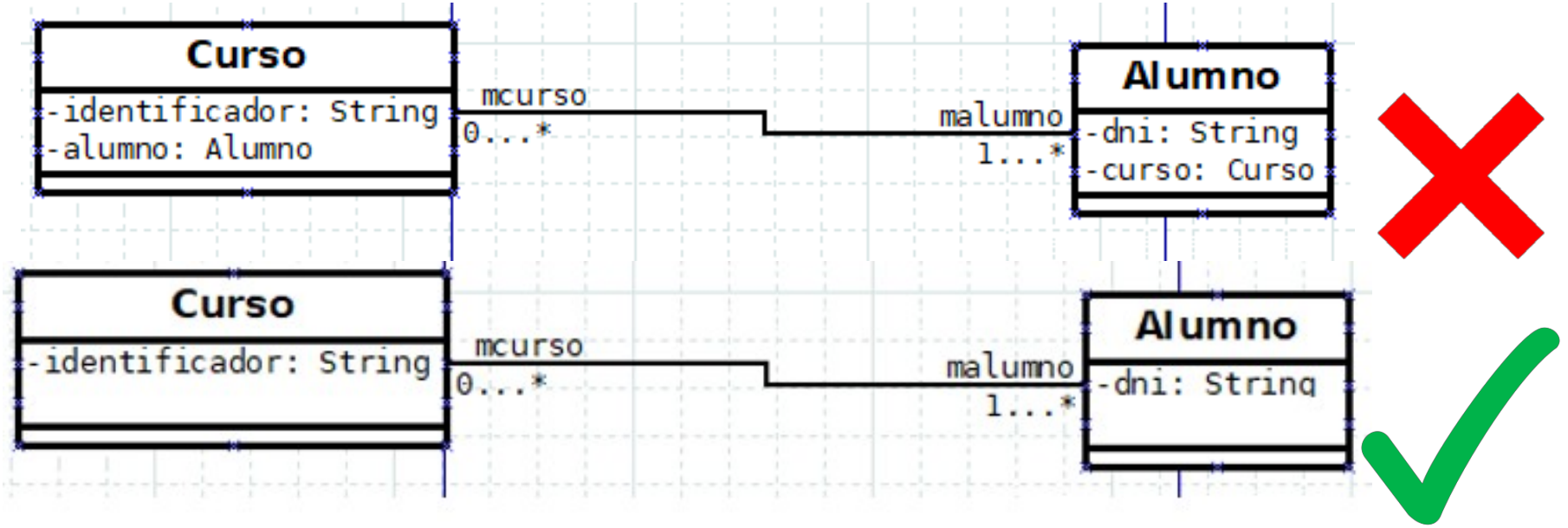
Diseño de Clases: tipos de relaciones

Para **simplificar** **No** se ponen los objetos fruto de la relación entre clases en el diagrama.

¡¡¡Son detalles de implementación. No de diseño!!!

Si es importante poner el **tipo de retorno** de los métodos

Por ejemplo:



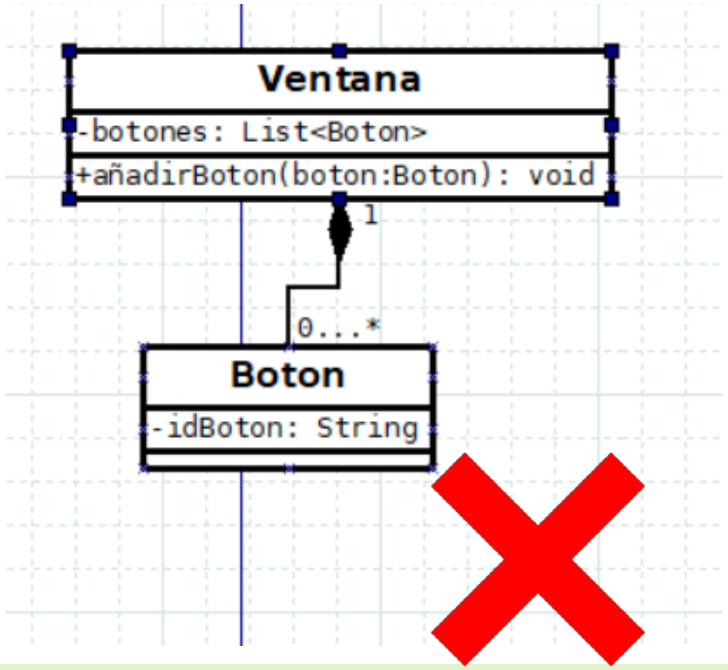
Diseño de Clases: tipos de relaciones

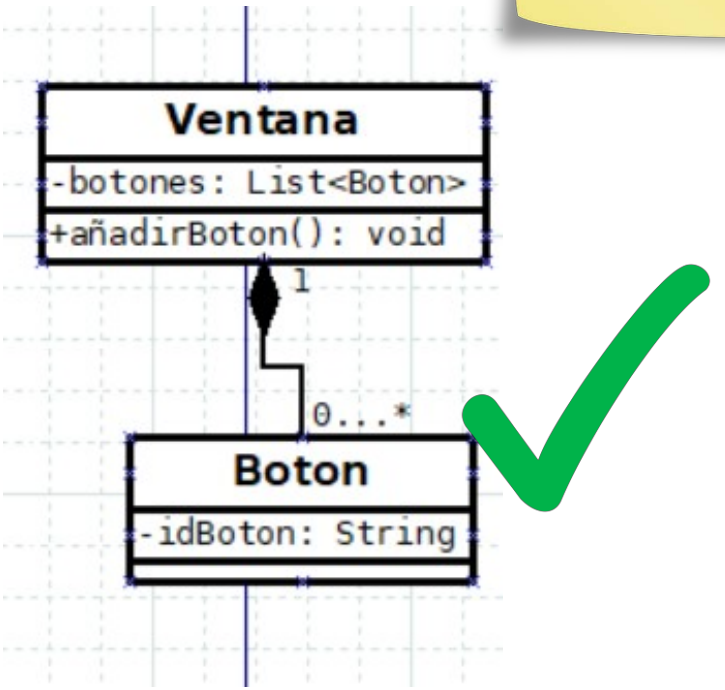
Para **simplificar** No se ponen los objetos fruto de la relación entre clases en el diagrama.

¡¡¡Son detalles de implementación. No de diseño!!!

Si es importante poner el **tipo de retorno** de los métodos

Por ejemplo:





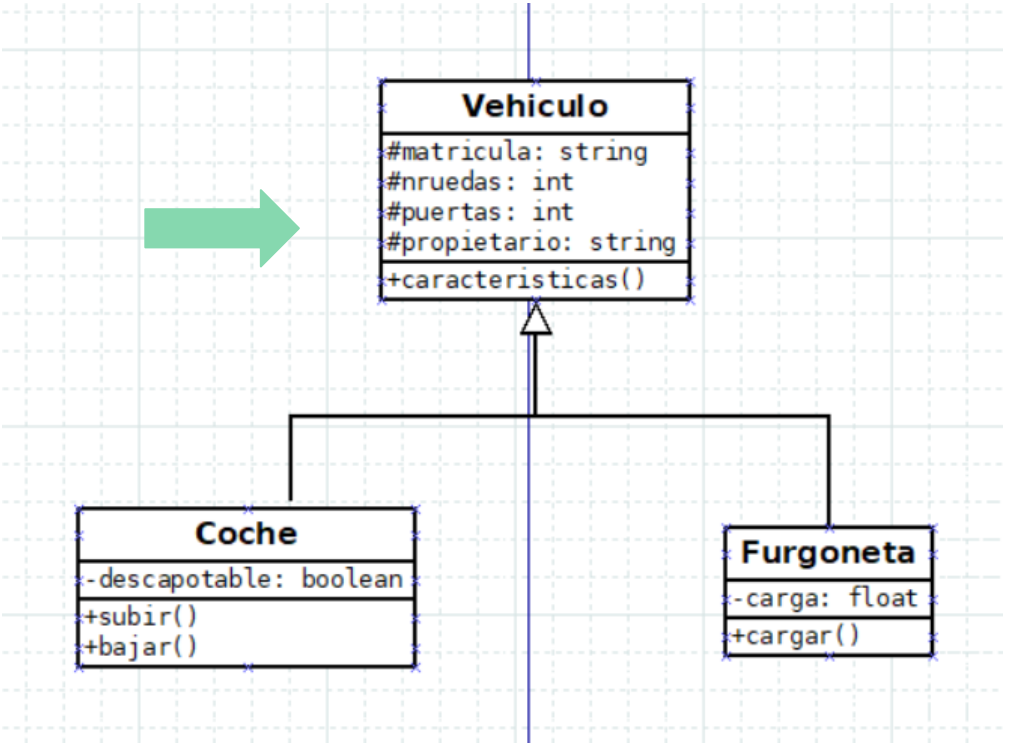
Diseño de Clases: tipos de relaciones

- **Generalización/Herencia**:

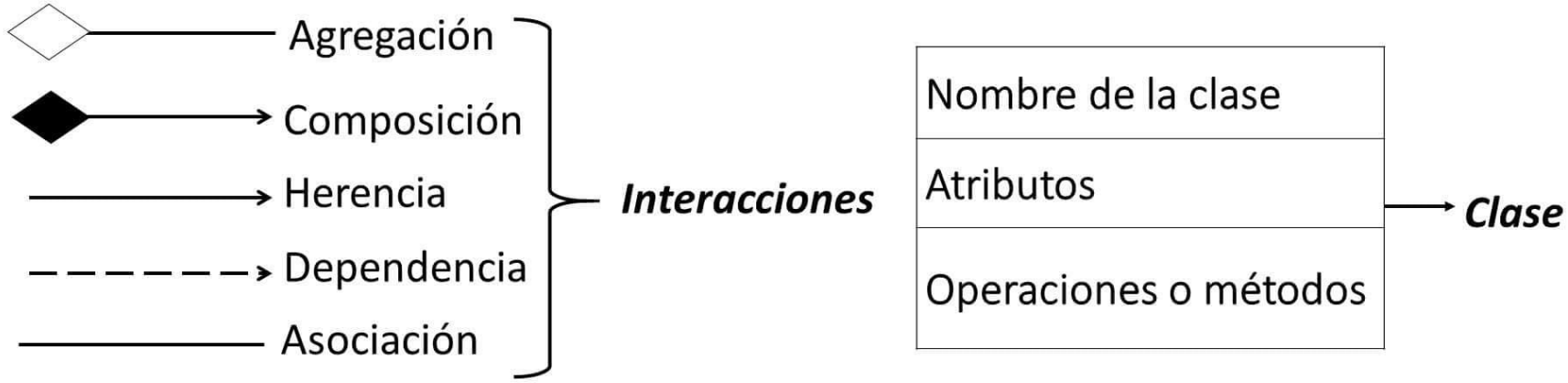
  Se da entre dos entidades: una principal o **clase padre** y otras sub-entidades o **clases hijas**. Las clases hijas son clases especializadas de la clase padre.

  - Las clases hijas **heredan** las características y comportamientos de las clases padres.
  - **Añaden** características y comportamientos nuevos que las diferencian entre sí.

ES UN TIPO DE...



Elementos y símbolos en los diagramas de clases UML



1 no mas de uno
0..1 cero o uno
0..* cero o muchos
1..* uno o mucho
* muchos

Multiplicidad
(Numero de elementos relacionados)

\+ Publico
\- Privado
\# Protegido

Visibilidad
(niveles de acceso)

Herramientas útiles

- Herramientas modelado y código fuente:
  - Visual Paradigm.
  - Argo UML.
  - Dia + Dia2Code.

- Herramientas de modelado:
  - Dia.

- Ingeniería inversa:

  Es posible hacer Ingeniería inversa a través del código fuente.

La **ingeniería inversa** es el proceso que identifica las propiedades de un objeto físico mediante la realización de un análisis exhaustivo de su estructura, funciones y operaciones.