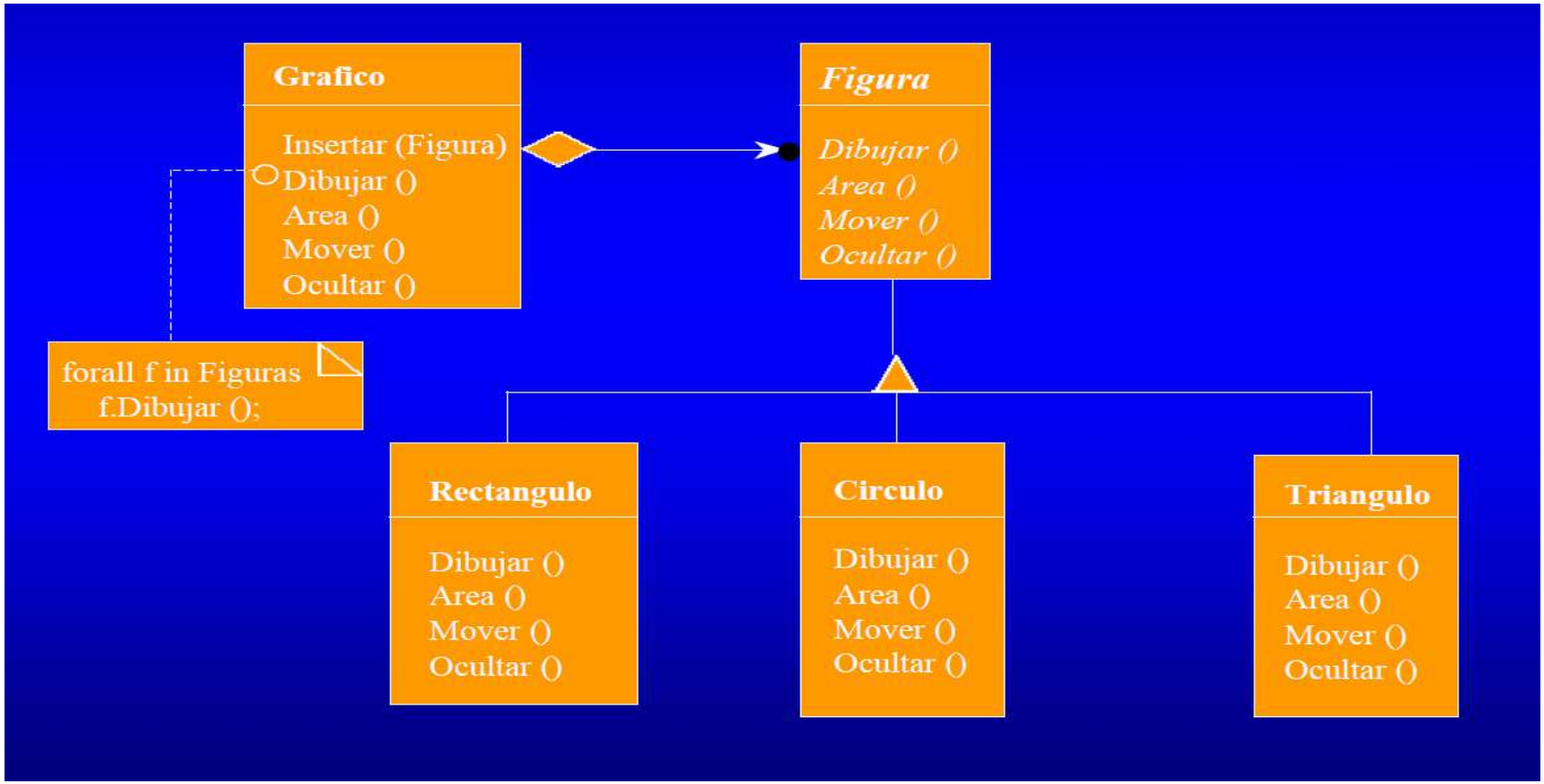
Relación de composición / agregación

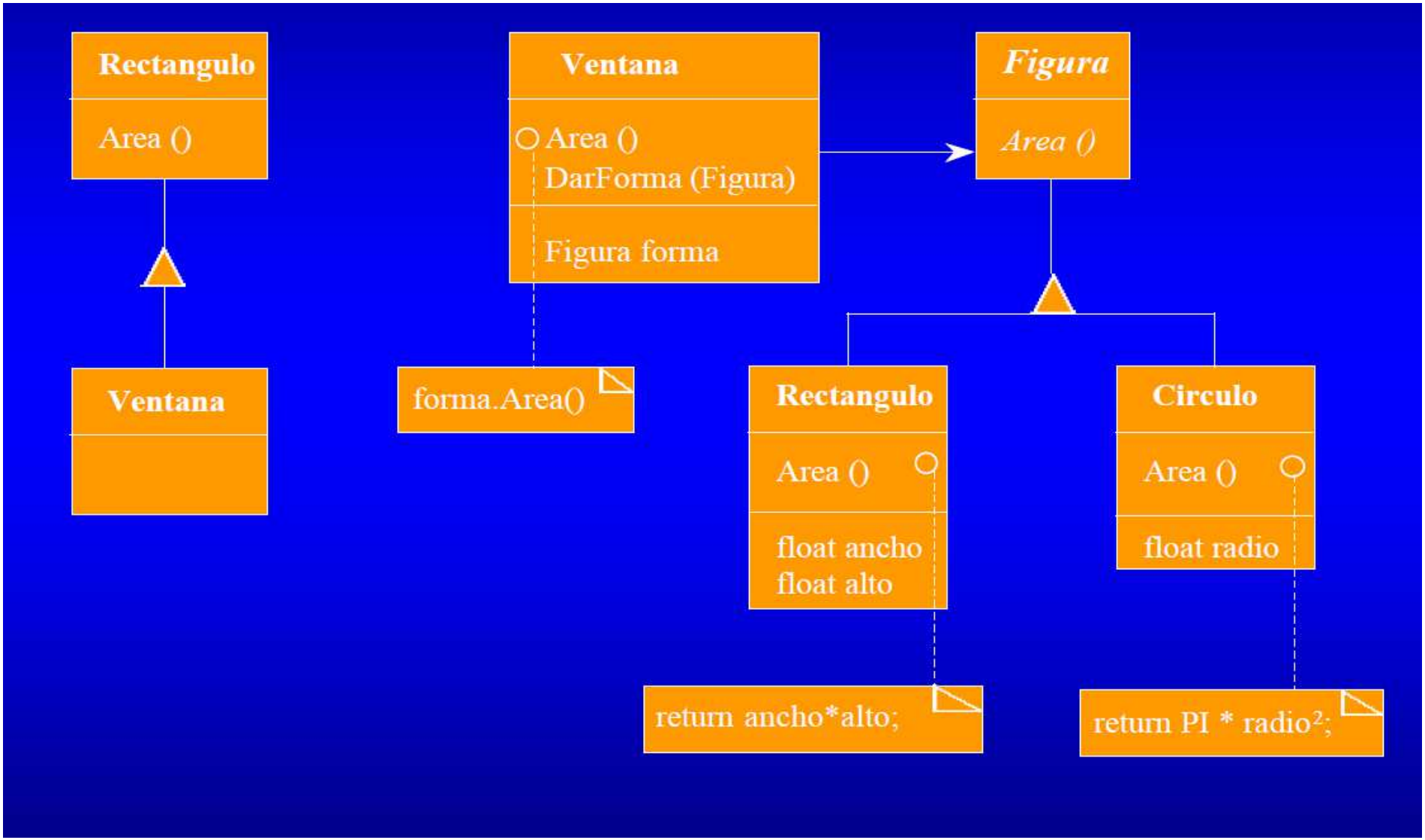
- Un *Grafico* se compone de varios objetos *Figura*.

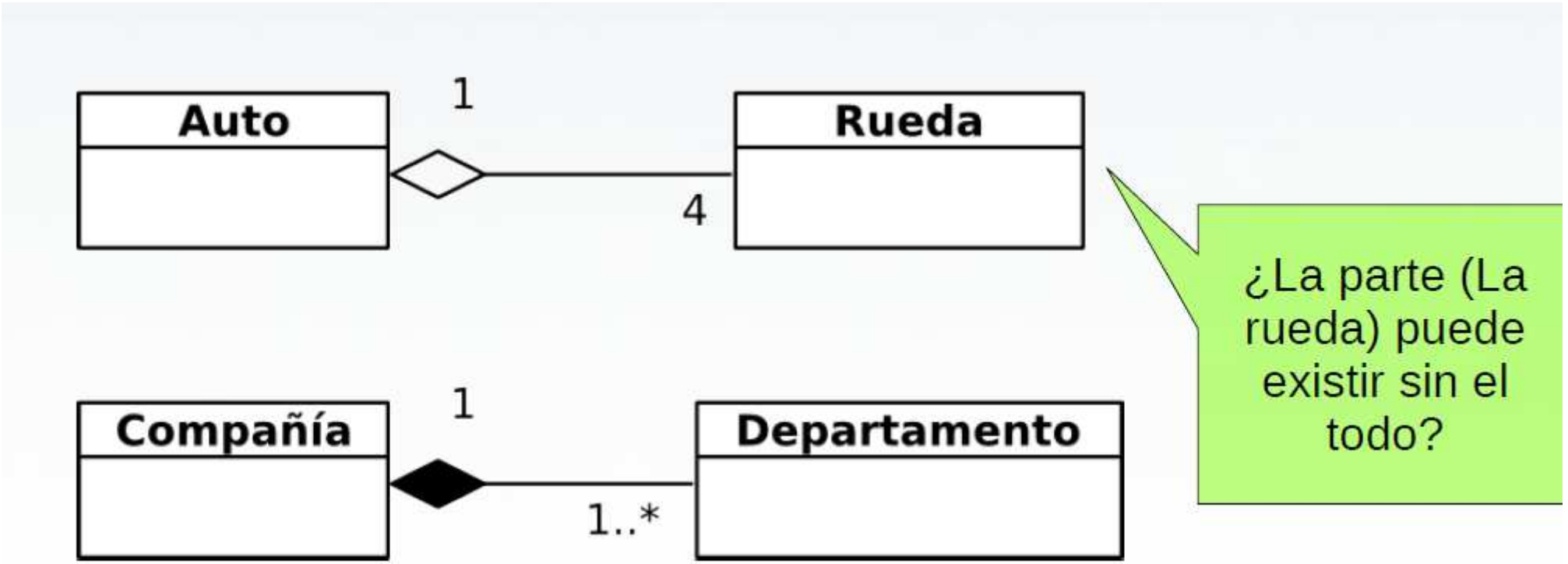


Relación de composición / agregación

- El código de los métodos del agregado (Grafico) delega parte de su implementación en métodos de sus partes (Figura).
- El agregado puede tener la responsabilidad de liberar la memoria de sus partes (ejemplo anterior) o no (siguiente ejemplo).
- La diferencia entre composición y agregación está en que en la composición existe dependencia de existencia, la composición se representa con un rombo vacío y la agregación con un rombo relleno.

Reutilización de código: herencia de clase vs composición




Auto
1
Rueda
4
¿La parte (La rueda) puede existir sin el todo?
Compañía
1
Departamento
1..*

Reutilización de código: herencia de clase vs composición

	Ventaja	Desventaja
Herencia	• Fácil de utilizar. • Fácil de modificar la implementación heredada.	• Establece relaciones estáticas. • Se rompe la encapsulación.
Composición	• Establece relaciones dinámicas. • Se mejora la encapsulación.	• Mayor número de objetos. • El comportamiento del sistema depende de las relaciones entre objetos, en vez de estar concentrado en una clase.

Reutilización de código: herencia de clase vs composición

- Consejos :
  - Favorecer la composición frente a la herencia de clase.
  - Evitar excesivas relaciones entre clases (las clases deben estar débilmente acopladas).
  - Evitar jerarquías de clases excesivamente complejas.

Ejercicios

- Modela y relaciona una factura con las lineas de factura.
- El guerrero lleva consigo una mochila en la que transporta sus armas. Crea una jerarquía de armas utilizando herencia y relaciona la mochila con las armas.

Dependencia

- La dependencia es la relación menos importante. Simplemente refleja que la implementación de una clase depende de otra.
- Una dependencia puede indicar la utilización de un objeto de una clase como argumento de una operación de otra o en su implementación, o bien que una clase instancia objetos de otra.

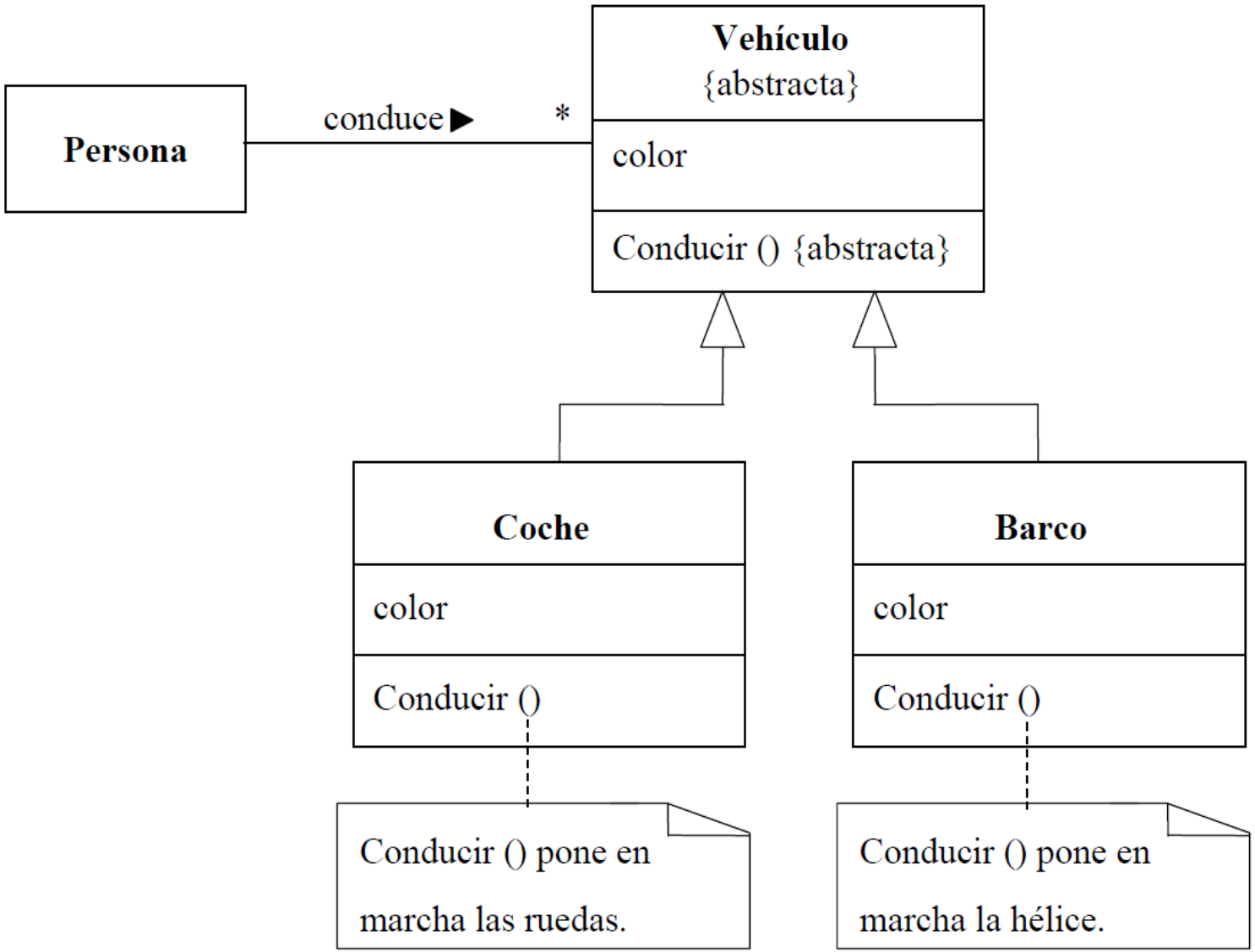
Asociación

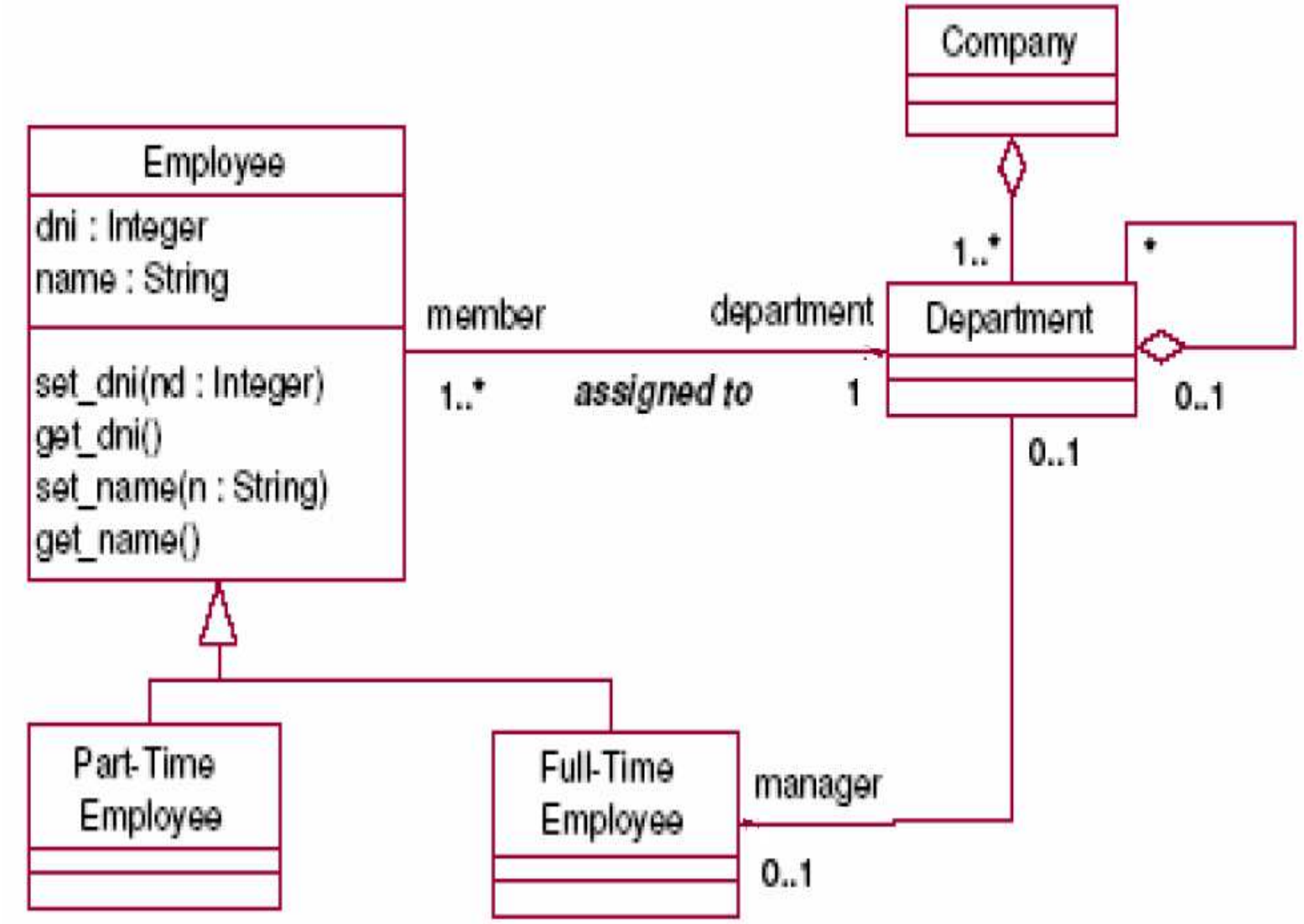
- En cambio, la asociación es la relación más importante y común. Refleja una relación entre dos clases independientes que se mantiene durante la vida de los objetos de dichas clases o al menos durante un tiempo prolongado.
- En UML suele indicarse el nombre de la relación, el sentido de dicha relación y las cardinalidades en los dos extremos.

Dependencia o asociación

- Encontrareis numerosa bibliografía explicando la semántica profunda de las dependencias y asociaciones con ejemplos alejados de la realidad y explicaciones densas y poco claras… os lo voy a poner muy fácil:
  - Si tenemos que guardar un atributo con una referencia a un objeto de otra clase, haremos una asociación.
  - Si usamos un objeto de otra clase, pero se usa momentáneamente en un método o sólo se recibe como argumento, pero no lo guardamos como atributo, estableceremos una dependencia.


Departamento
1
1..*
Profesor
Estudiante
0..*
0..*
Asignatura
Departamento
1
1
Secretaria


Persona
conduce ▶
*
Vehículo
{abstracta}
color
Conducir () {abstracta}
Coche
color
Conducir ()
Barco
color
Conducir ()
Conducir () pone en marcha las ruedas.
Conducir () pone en marcha la hélice.


Company
Employee
dni : Integer
name : String
set_dni(nd : Integer)
get_dni()
set_name(n : String)
get_name()
member
department
1..*
assigned to
1
1..*
*
Department
0..1
0..1
Part-Time
Employee
Full-Time
Employee
manager
0..1