UCA

Programación Orientada a Objetos I

UML – RELACIONES

Relaciones

- Las cosas no se encuentran aisladas, de alguna manera están conectadas. Las clases tampoco (al fin y al cabo representan cosas).
- Esas conexiones se llaman relaciones
- Las relaciones no son exclusivas de las clases. Otros diagramas de UML también las utilizan para modelar
- Gráficamente se representan como una línea por cada relación, usando diferentes tipos de línea para modelar diferente relaciones

Generalización

- Es una relación entre un elemento general, y un elemento específico
- El elemento general se llama padre o superclase, y el elemento específico se llama hijo o subclase
- Es una relación *"… es un tipo de …"* si tenemos en cuenta una semántica de lo especial a lo general
- Un objeto de la clase hija hereda los atributos y operaciones de la clase padre
- A menudo (pero no siempre) el hijo añade atributos y/u operaciones a los heredados del padre
- La implementación de una operación en una clase hija redefine la misma operación que se implementó en el padre. Esto es polimorfismo
- Un objeto de la clase hija se puede asociar a una variable de la clase padre, pero no a la inversa. Es decir, la herencia permite que los objetos de la clase hija se comporten como objetos de la clase padre, pero no al revés (la clase hija es una extensión o especialización del padre)

Generalización

- Una clase puede tener ninguno, uno o varios padres.
- Una clase sin padres, pero con hijos se llama clase raíz o clase base
- Una clase con padre pero sin hijos se llama clase hoja
- Una clase con un único padre se dice que tiene herencia simple
- Una clase con más de un padre se dice que tiene herencia múltiple
- Gráficamente se representa con como una línea continua, con una punta de flecha vacía (triangulo blanco) apuntando hacia la clase padre.

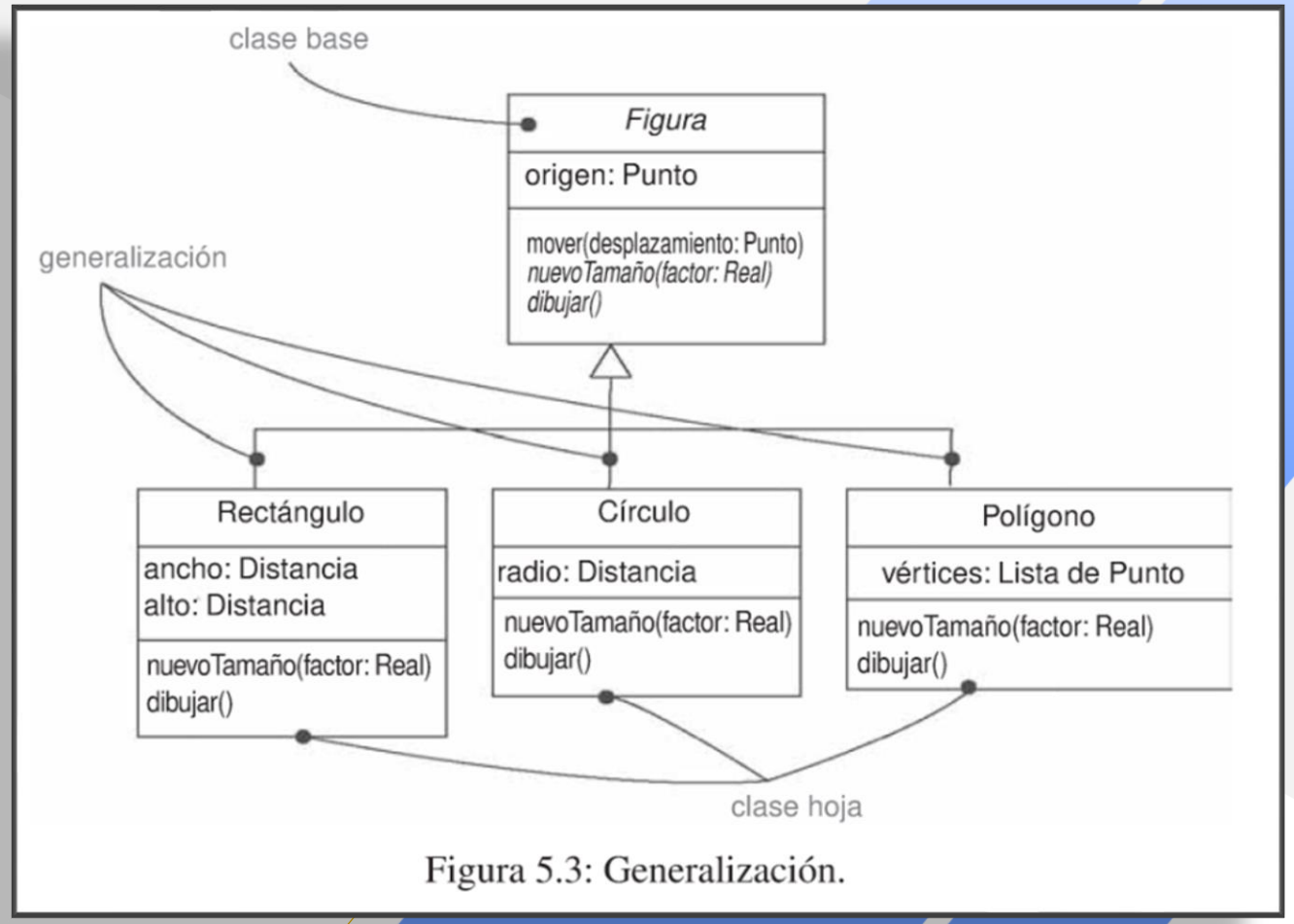
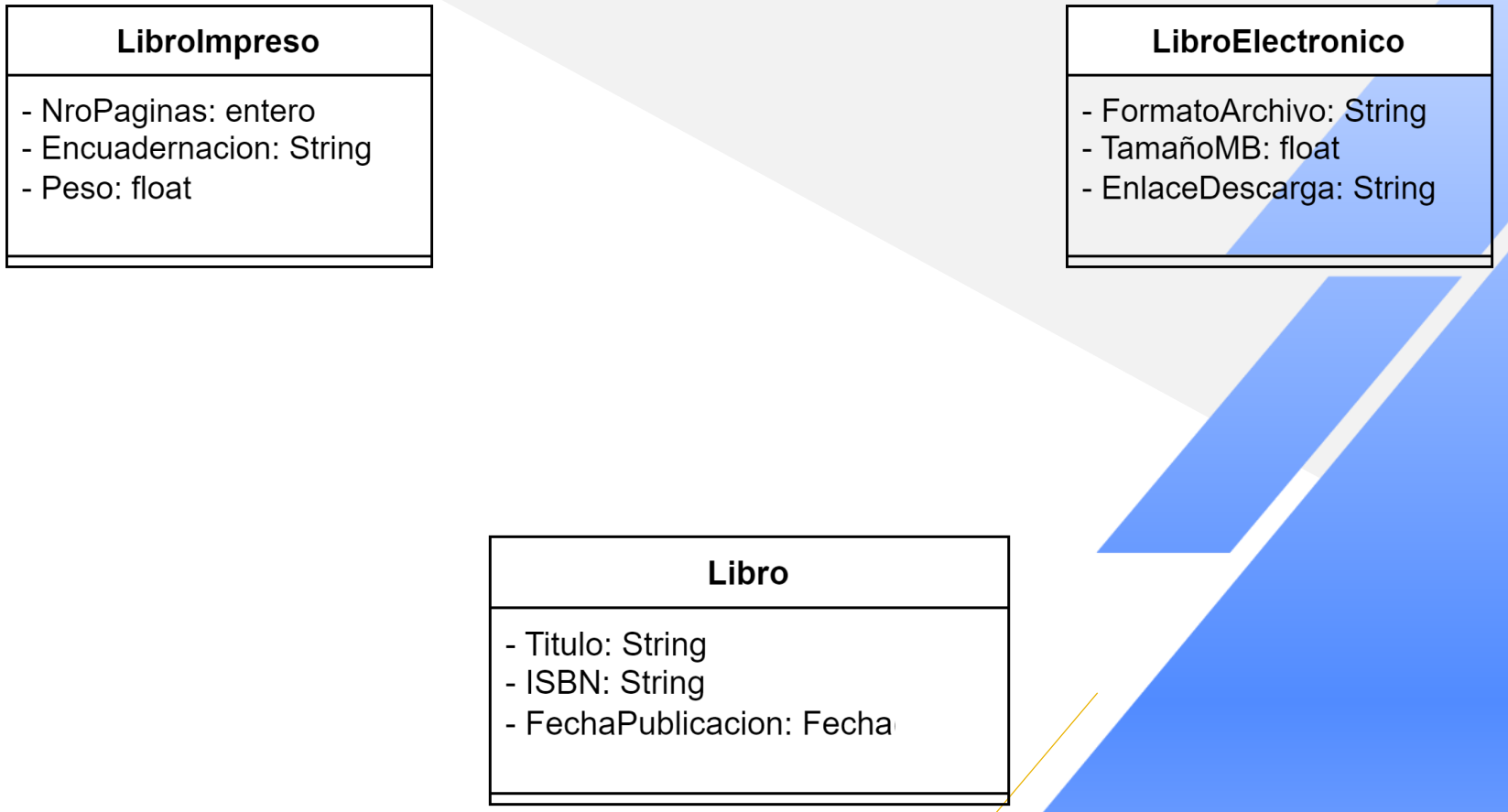
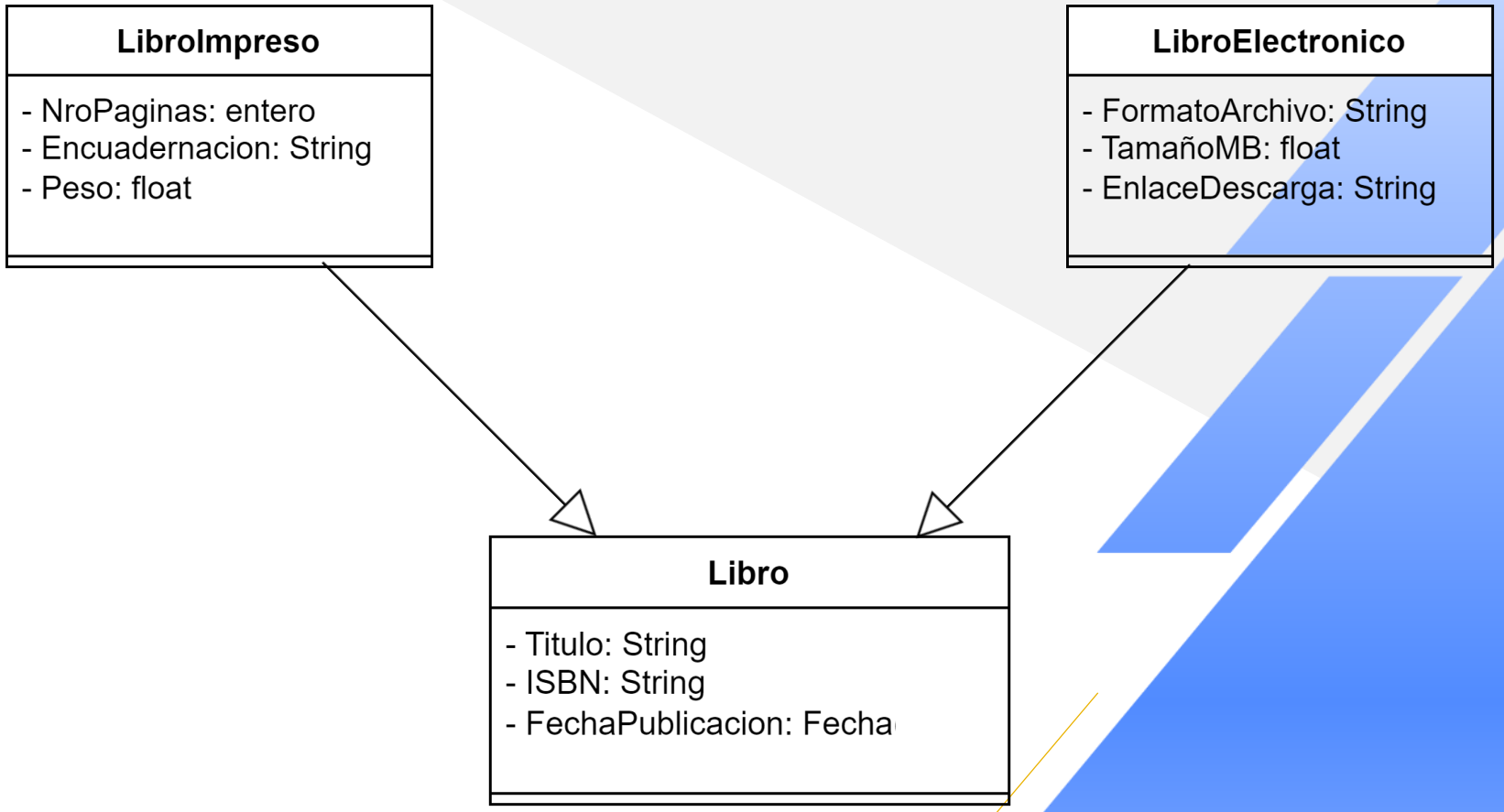


Figura 5.3: Generalización.

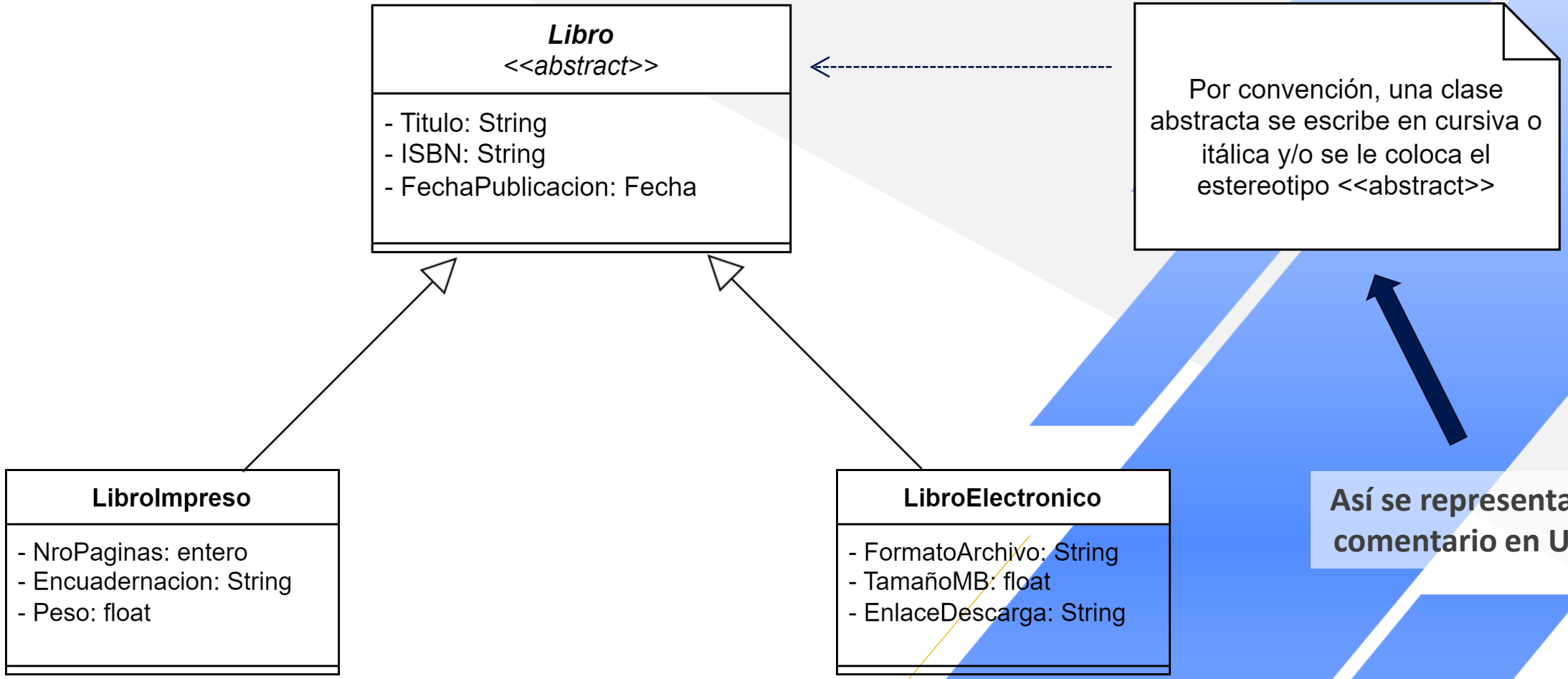
Ejemplo de generalización



Ejemplo de generalización



Ejemplo de generalización



Asociación

- Es una relación estructural que especifica que los objetos de un elemento están conectados con los del otro objeto
- Puede conectarse a si misma, es decir que un objeto de una clase podría estar conectado con otro objeto de la misma clase
- Es un tipo de relación *"… se relaciona con… "*, pero puede especificarse mediante nombres y roles
- Gráficamente se representa con una línea continua que conecta las clases

Asociación: adornos

- **Nombre**: Una asociación puede tener un nombre que describe la naturaleza de la relación. Para evitar la ambigüedad se puede dar un sentido de flecha en el sentido de lectura

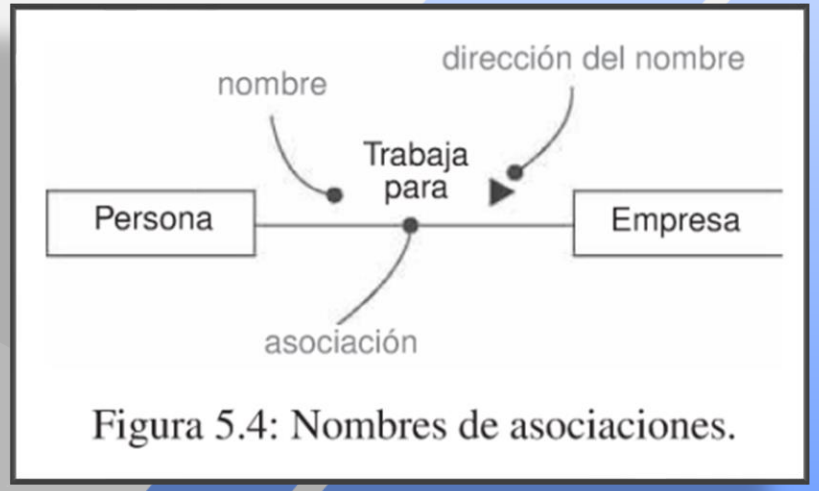


Figura 5.4: Nombres de asociaciones.

- **Rol**: Cuando una clase participa en una asociación tiene un rol en esa relación. Se puede dar un nombre a esos roles en cada uno de los extremos

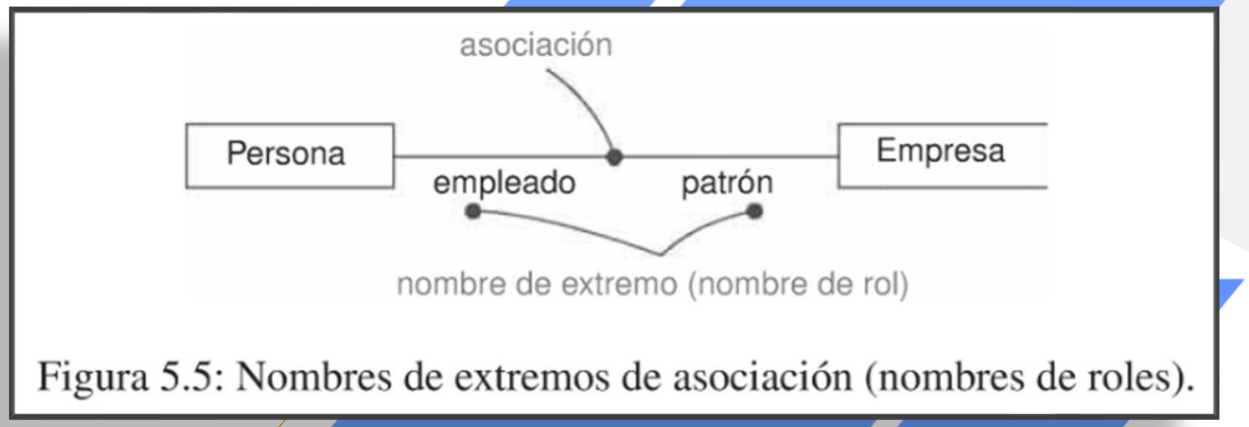
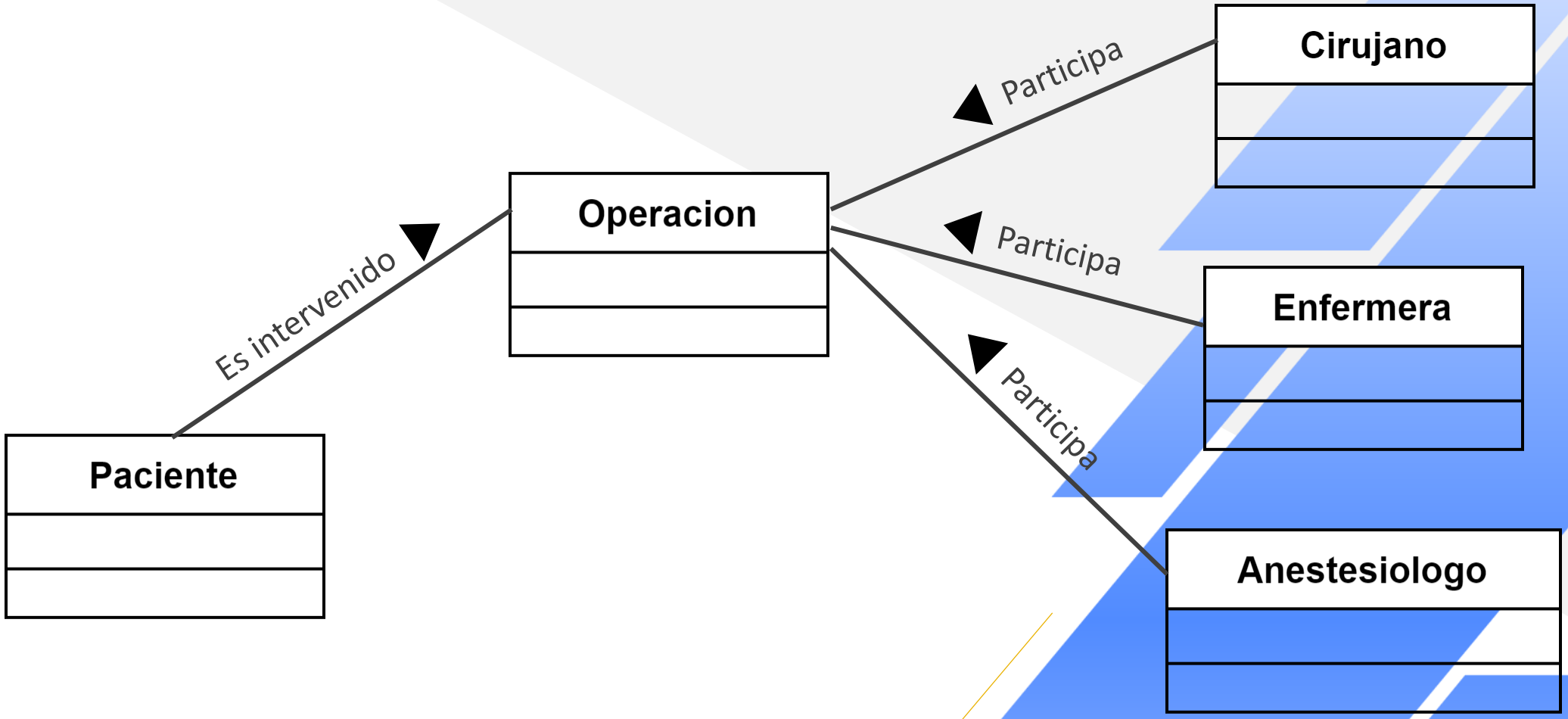


Figura 5.5: Nombres de extremos de asociación (nombres de roles).

Ejemplo


Cirujano
Participa
Operacion
Participa
Enfermera
Es intervenido
Participa
Paciente
Anestesiologo

Asociación: adornos

- **Multiplicidad**: A veces es útil o necesario señalar *cuantos* objetos pueden conectarse a través de la asociación. Eso es la multiplicidad (o cardinalidad)
- Muestra la cantidad de objetos que puede haber de esta clase, por cada objeto de la otra clase
- Se puede indicar:
  - Como un valor. Ejemplo: 1
  - Como un rango entre valor mínimo y máximo. Ejemplo: 0..4
  - Cuando queremos decir “muchos” sin establecer un límite se usa el asterisco. Ejemplo: 1..*
  - El asterisco solo es lo mismo que poner 0..*

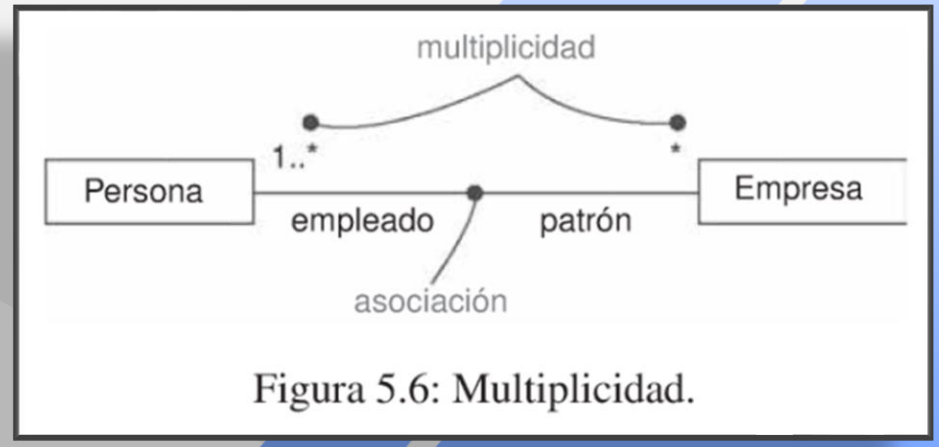
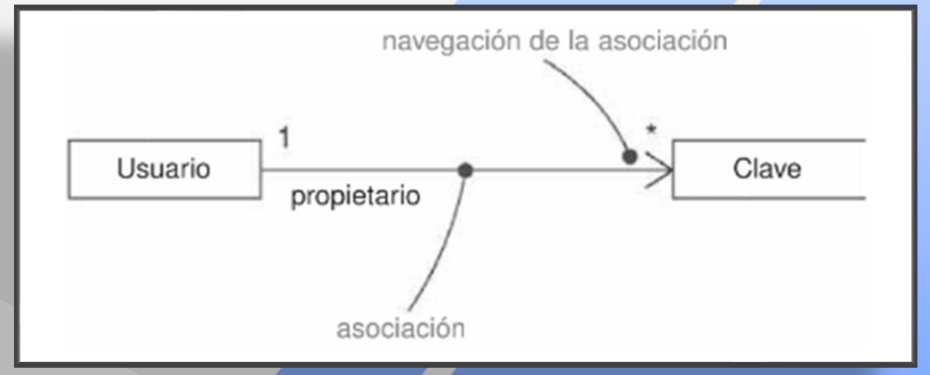


Figura 5.6: Multiplicidad.

- Cada empresa puede tener uno o más personas
- Cada persona puede trabajar para ninguna o varias empresas

Asociación: adornos

- Cada asociación tiene dos multiplicidades o cardinalidades, una a cada extremo de la relación.
- Significados:
  - 1: Uno y solo uno
  - 0..1: Uno o ninguno
  - *: Ninguno o varios
  - 0..*: Ninguno o varios
  - 1..*: Uno o varios (al menos uno)
  - 2..5: Al menos dos pero no más de cinco



Lectura de la relación:

- Un usuario es propietario de cero o varias claves
- Una clave puede pertenecer a uno usuario

O mejor aún…

- **Cada usuario puede tener o no varias claves, y a su vez cada clave solo puede pertenecer a un solo usuario**

Asociación: agregación

- La ***agregación simple*** es una relación donde hay una clase que es más importante que la otra.
- Es una relación del tipo todo/parte, una clase que representa el todo, y otra que representa una parte
- Es del tipo *"…es una parte de…"*, o en forma inversa "*… tiene un…"*
- Un objeto del "todo" tiene objetos de la "parte"
- Los objetos parte pueden pertenecer a varios objetos todo.
- Los objetos parte existen con independencia de la existencia del todo, es decir que no se destruyen con la destrucción del todo (no es responsabilidad del todo)
- Gráficamente se representa con un rombo vació en la parte del todo

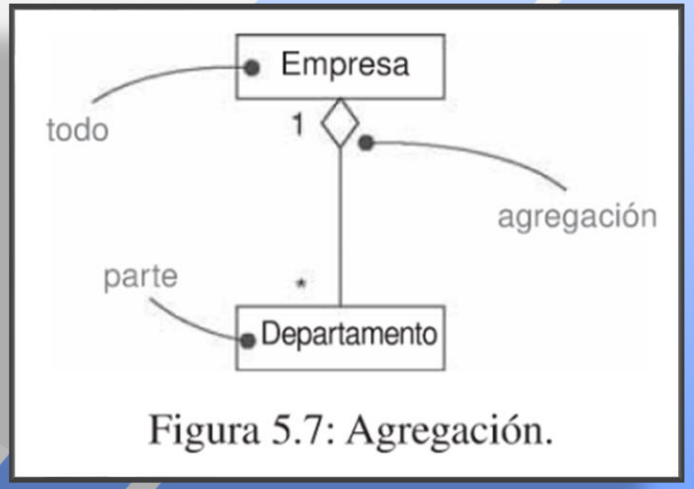


Figura 5.7: Agregación.

Asociación: composición

- La ***agregación por composición*** es similar a la agregación simple, pero fuerte relación de pertenencia y vida.
- Es del tipo *"… está compuesto por …"*
- Los objetos parte solo pueden pertenecer a un solo objeto todo (también llamado contenedor).
- Los objetos parte existen mientras exista el objeto todo.
- Es responsabilidad del objeto todo la creación y destrucción de objetos parte
- Gráficamente se representa con un rombo relleno en la parte del todo.

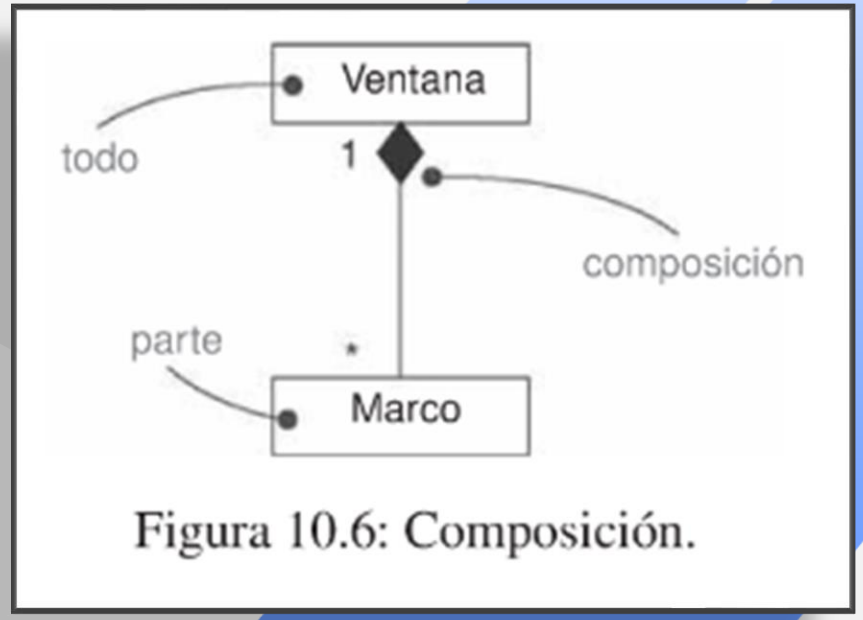
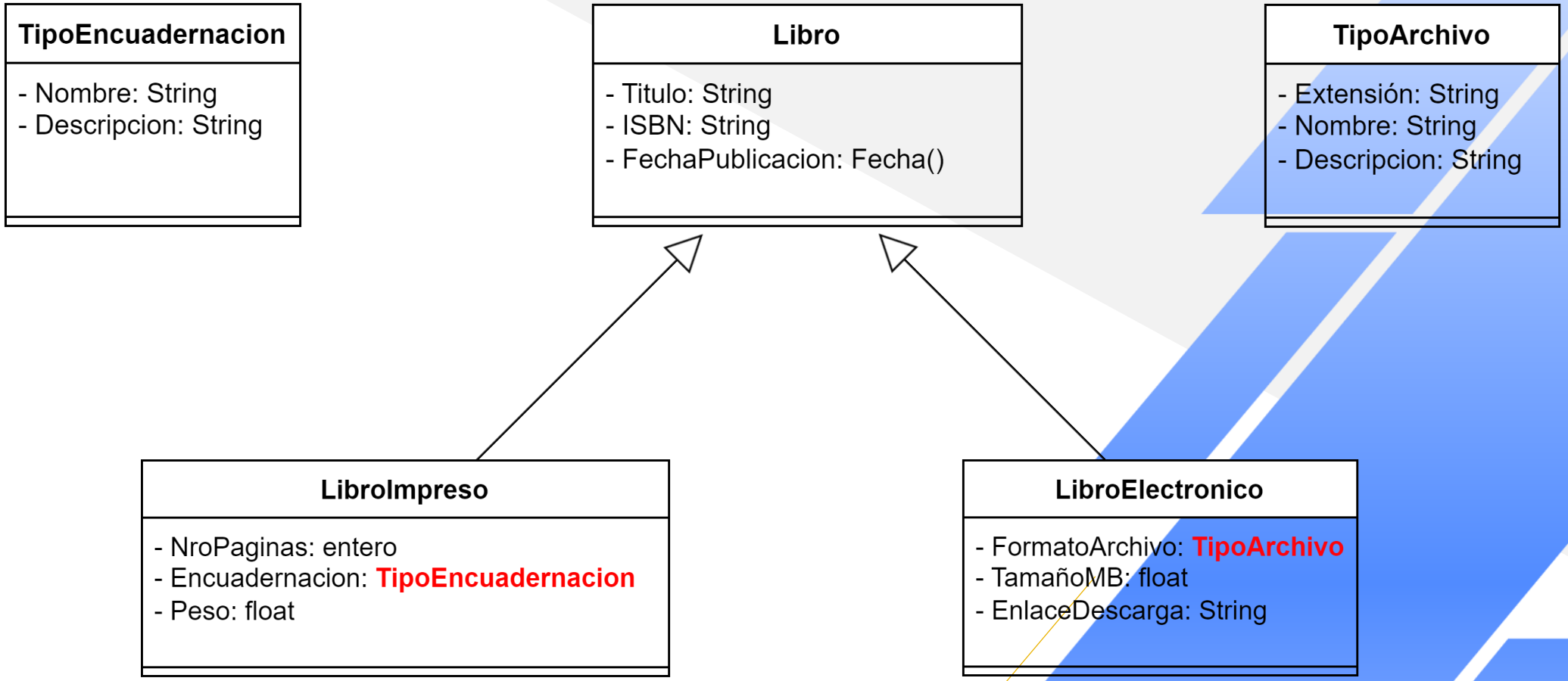
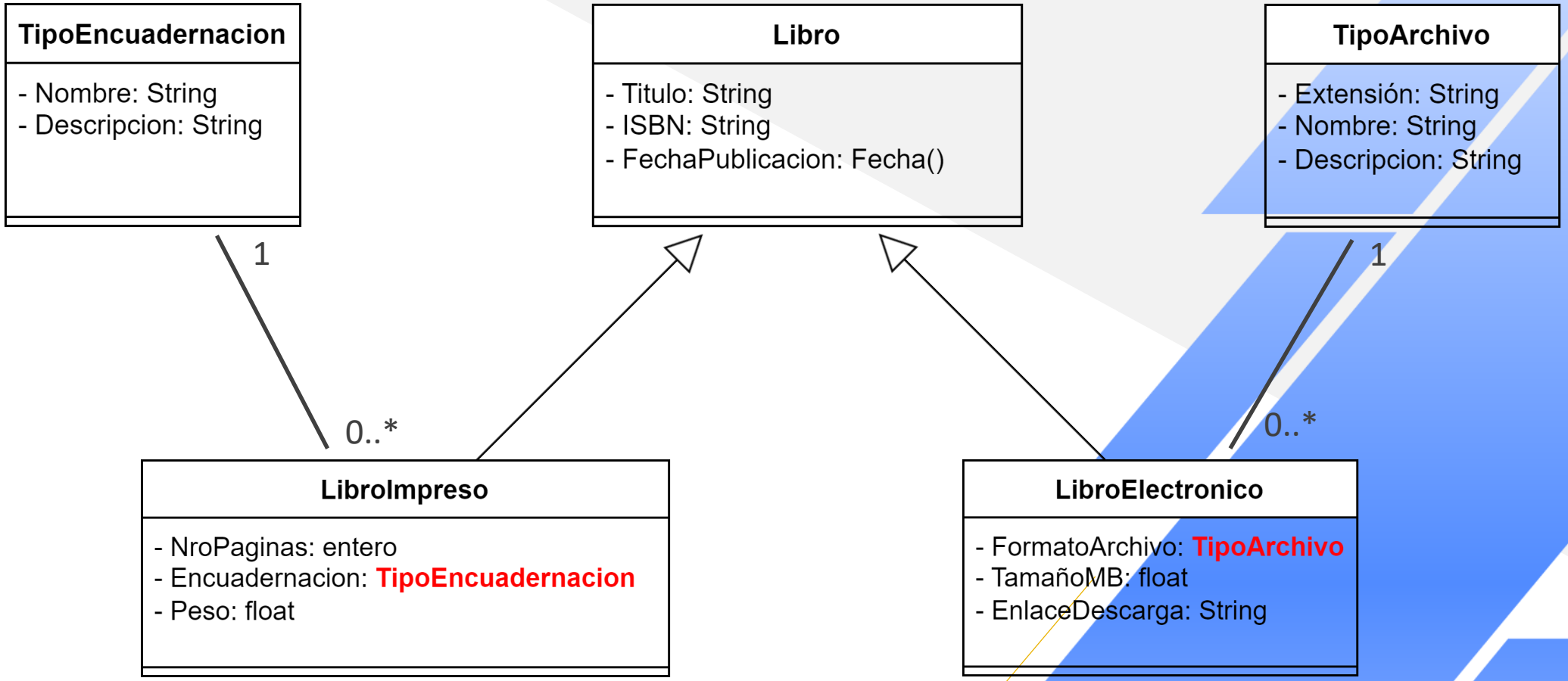


Figura 10.6: Composición.

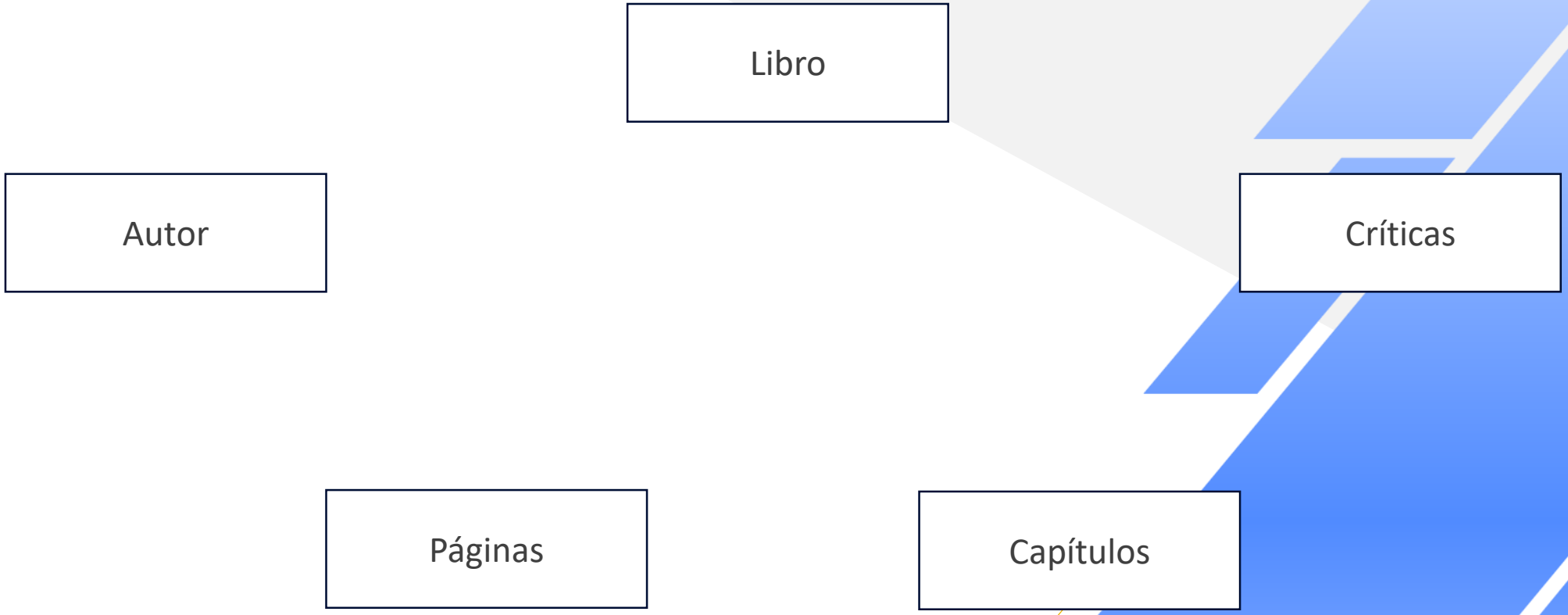
Ejemplo de asociación



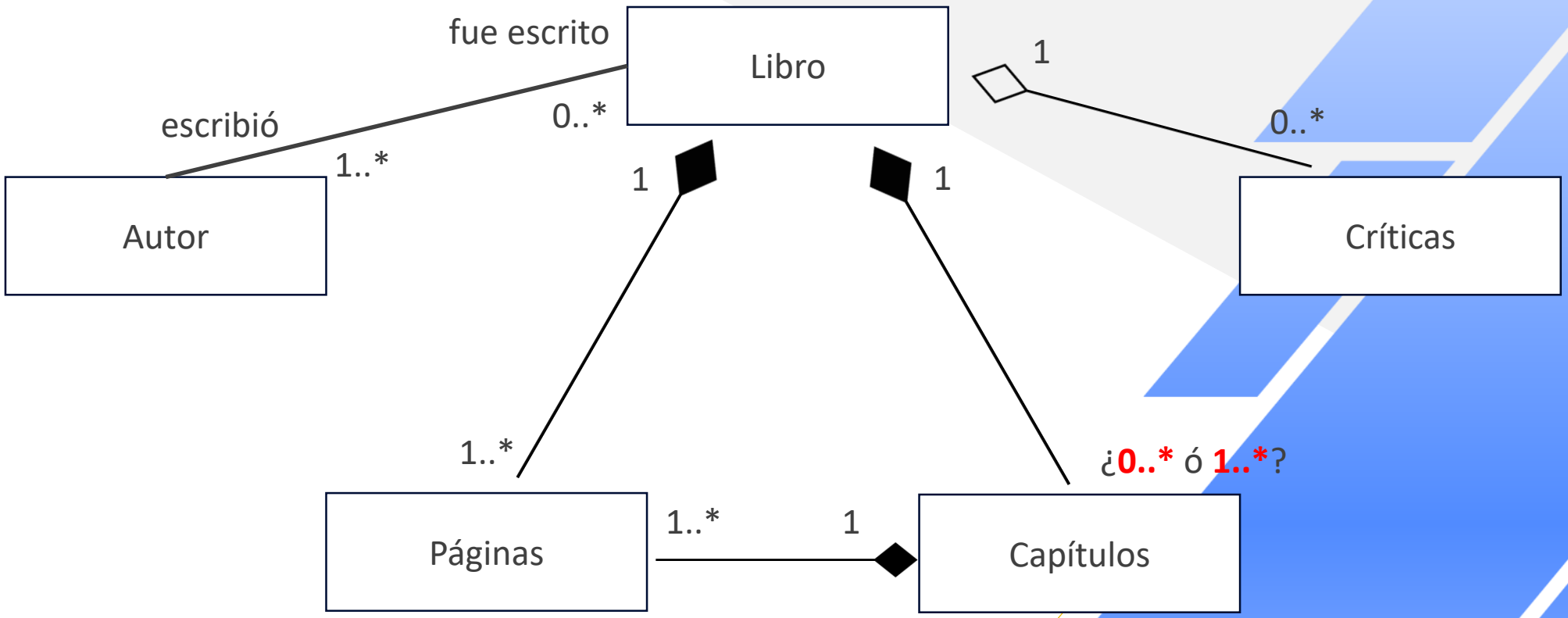
Ejemplo de asociación



Ejemplo de asociación



Ejemplo de asociación



Dependencia

- Es una relación que declara que un elemento usa y/o necesita los servicios de otro elemento, pero no necesariamente a la inversa
- Es una relación del tipo *"… depende de …", "… utiliza a …"* o *"… requiere de… "* en el sentido de la flecha
- Normalmente las utilizamos cuando queremos modelar que una clase va a necesitar los servicios (operaciones) de otra clase.
- Si la clase utilizada cambia, la operación de la otra clase podría cambiar (por eso es dependencia… DEPENDE de la otra clase)
- Se representa con una línea discontinua con punta de flecha apuntando al elemento del cual se depende

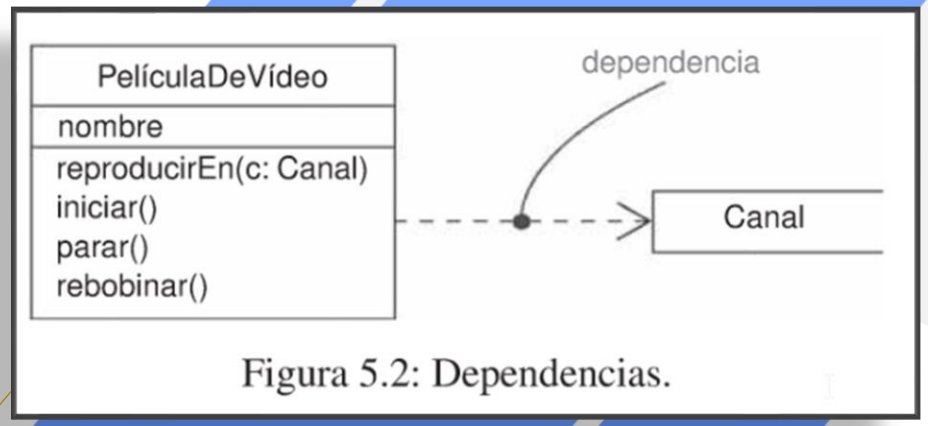


Figura 5.2: Dependencias.

Otras consideraciones

- Lo que más se utilizan son:
  - Dependencias simples y sin adornos
  - Generalizaciones (herencia)
  - Asociaciones con nombre, multiplicidad y rol
- Modelar la herencia (generalización) a través de varios "niveles" de ser necesario
- Estilo de dibujo:
  - Líneas horizontales/verticales que se doblan en ángulos de 90º
  - Líneas oblicuas (con cualquier ángulo)
  - No usar líneas curvas (dificultan la lectura)
  - Cruce de líneas con salto

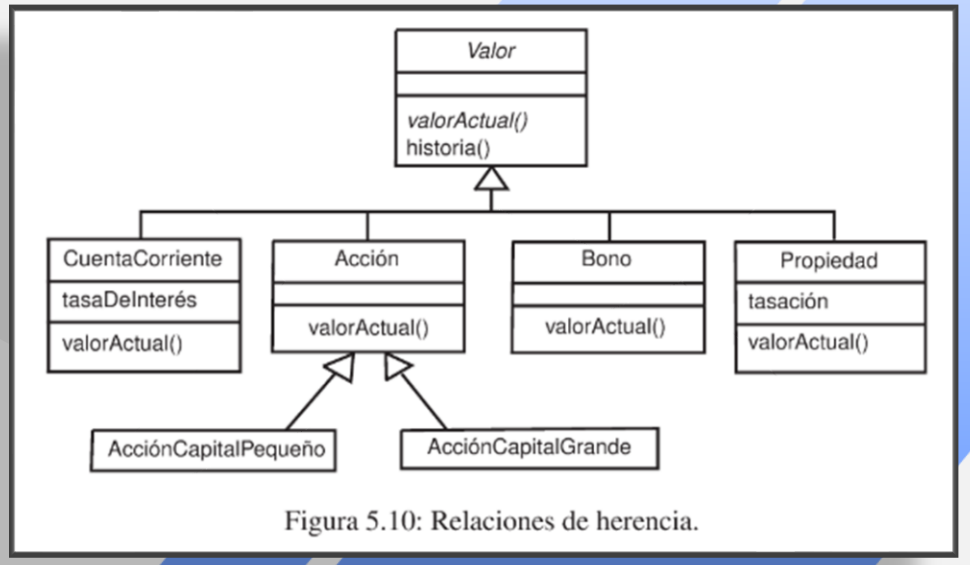


Figura 5.10: Relaciones de herencia.

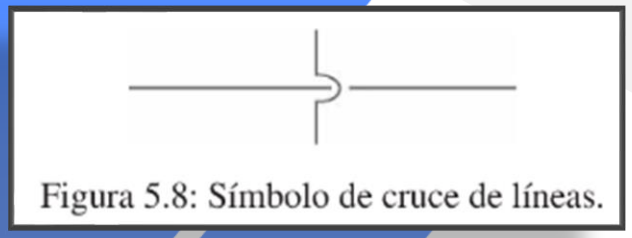


Figura 5.8: Símbolo de cruce de líneas.

Ejemplo

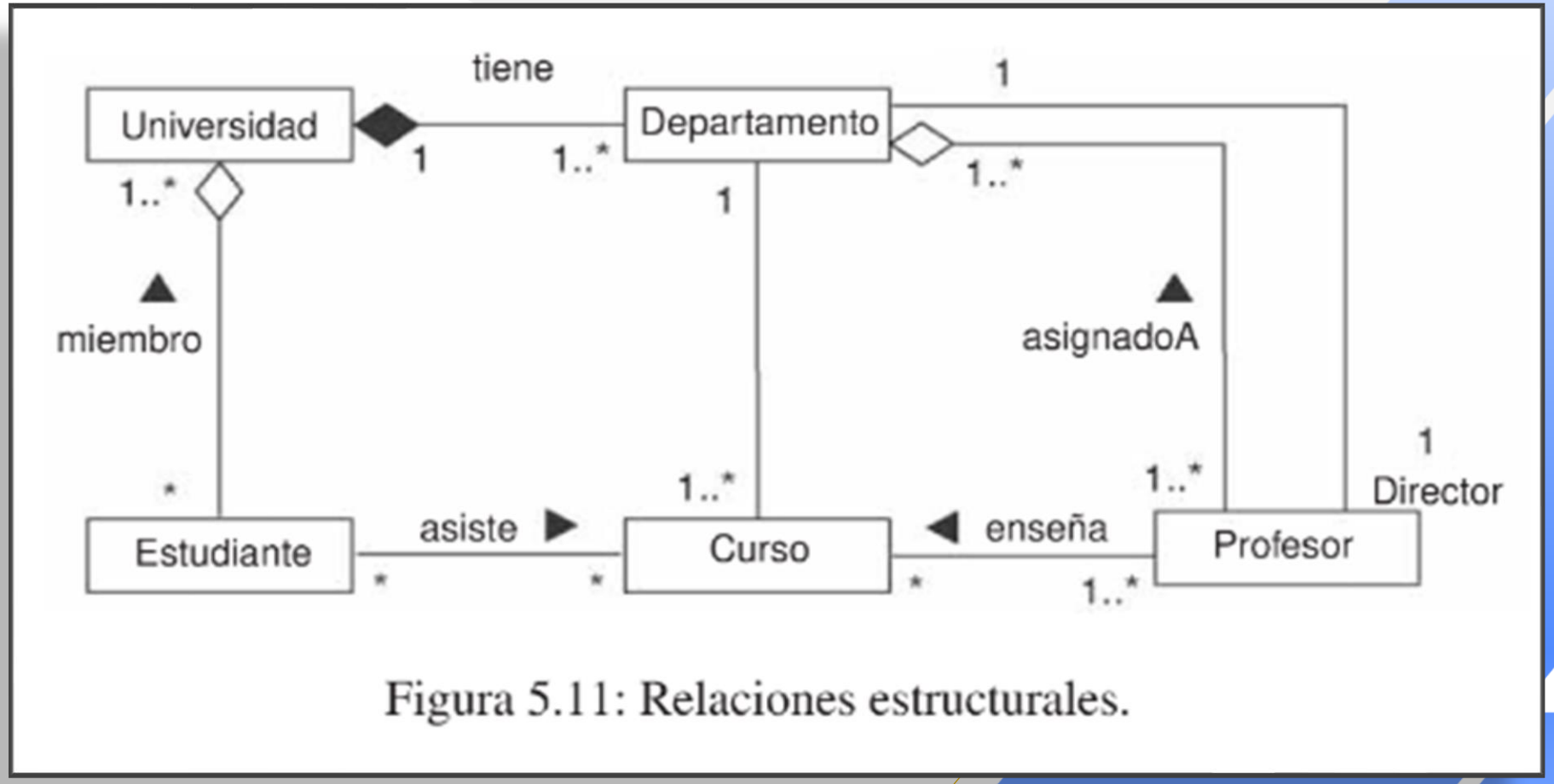


Figura 5.11: Relaciones estructurales.

Para la clase que viene (UML)

- Repasar los PPTs y leer de UML Guía de usuario:
  - UML 1: Introducción
    - Cap. 1: Por qué modelamos (12 pág)
    - Cap. 2: Presentación de UML (26 pág)
  - UML 2: Clases
    - Cap. 4: Clases (14 pág)
  - UML 3: Relaciones
    - Cap. 5 Relaciones (16 pág)
- Pueden complementar con la siguiente bibliografía:
  - UML Gota a gota: de Martin Fowler/Kendal Scott
  - Aprendiendo UML en 24 horas: Joseph Schmuller