MODELADO ORIENTADO A OBJETOS CON UML

Tema 4

Ingeniería del Software
ETS Ingeniería Informática
DSIC – UPV

Curso 2024-2025

Objetivos

- Mostrar la necesidad de construir modelos para resolver problemas complejos y de grandes dimensiones
- Comprender qué es el modelado conceptual y distinguirlo claramente del diseño
- Aprender un subconjunto de UML, como notación de modelado OO
- Modelado estructural de un sistema
- Modelado del comportamiento de un sistema

Contenidos

1. Motivación.
2. Modelado OO
   - Visión de un sistema software OO
3. Notación UML
   - Diagrama de Clases. **Parte 1**
   - Diagrama de Casos de Uso. **Parte 2**
   - Diagramas de Secuencia
   - Otros diagramas

Bibliografía básica

- 🕮 Booch, G., Rumbaugh, J., Jacobson, I., UML. El Lenguaje Unificado de Modelado. UML 2.0 2ª Edición. Addison-Wesley, 2006

- 🕮 Stevens, P., Pooley, R. Utilización de UML en Ingeniería del Software con Objetos y Componentes. 2ª Edición. Addison-Wesley Iberoamericana 2007Ingeniería del Software. (8ª ed.). Addison-Wesley, 2008

- 🕮 Weitzenfeld, A., Ingeniería del Software OO con UML. Java e Internet. Thomson, 2005

- 🕮 https://www.uml.org/

Motivación

¿Qué es un modelo?

> "Un modelo es una simplificación de la realidad"

¿Por qué modelamos?

Construimos modelos para comprender mejor el sistema que estamos desarrollando

- Nos ayudan a **visualizar** cómo es o queremos que sea un sistema.
- Nos permiten **especificar** la estructura o el comportamiento de un sistema
- Nos proporcionan plantillas que nos guían en la **construcción** de un sistema
- **Documentan** las decisiones que hemos adoptado

Motivación

Modelado Orientado a Objetos

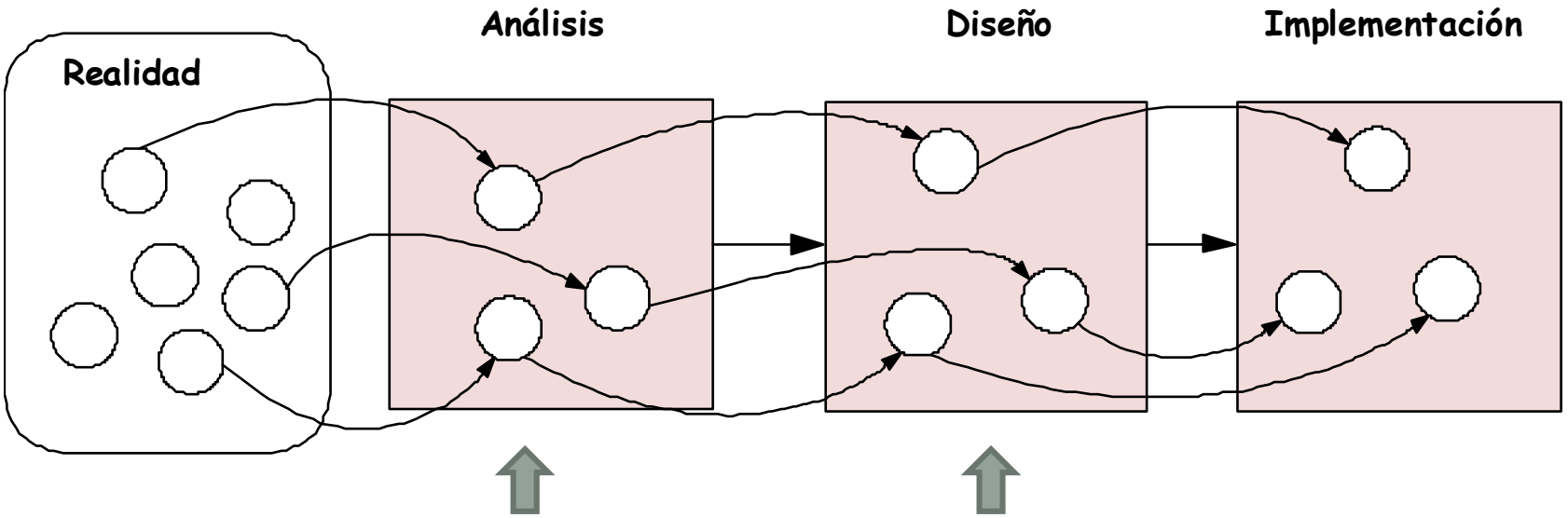
- Aparecen los lenguajes de programación OO
- Se requiere un nuevo enfoque de análisis y diseño

*"Un proceso que examina los requisitos desde la perspectiva de las **clases** y **objetos** encontrados en el vocabulario del dominio del problema" (Booch, 1994)*

1. Próximo a los **mecanismos cognitivos humanos**
2. Desarrollo **incremental** bajo una **noción** común de **objeto**

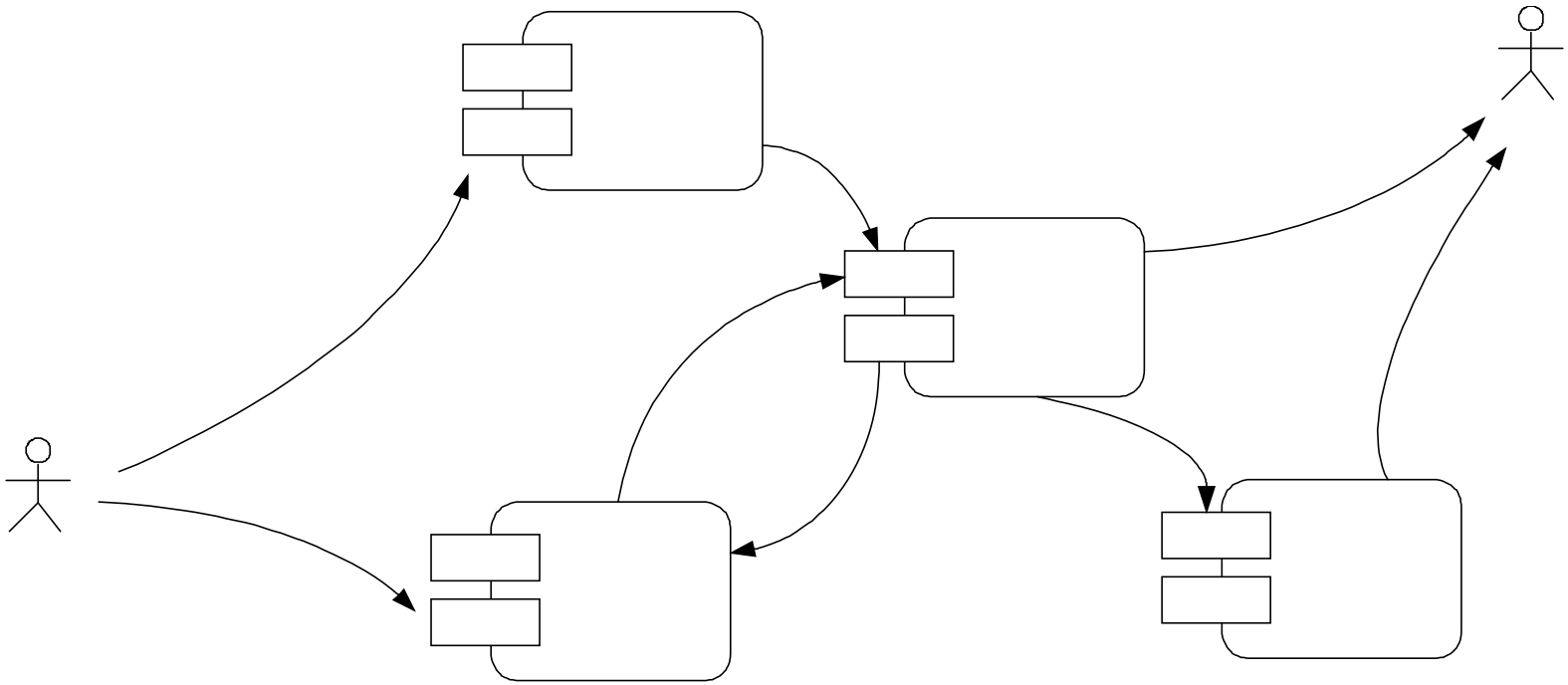
Motivación

En el enfoque OO, la descomposición del sistema se basa en los *objetos* o *clases de objeto* que se descubren en el dominio del problema



Visión de un Sistema Software OO

Visión estática + Visión dinámica



Sistema Software OO - Visión estática

Objeto:

- Entidad que existe en el mundo real
- Tiene identidad, una estructura y un estado

El avión con matrícula 1234
El avión con matrícula 6754

Clase:

- Describe un conjunto de objetos con las mismas propiedades y un comportamiento común.
- Relaciones entre clases

Avión

Sistema Software OO - Visión dinámica

- Los objetos se comunican mediante la invocación de métodos de otros objetos.

- Se describen aspectos de un sistema que cambian con el tiempo.
  - Interacciones entre objetos
  - Posibles estados de un objeto
  - Transiciones entre estados
  - Qué eventos se producen
  - Qué operaciones se ejecutan

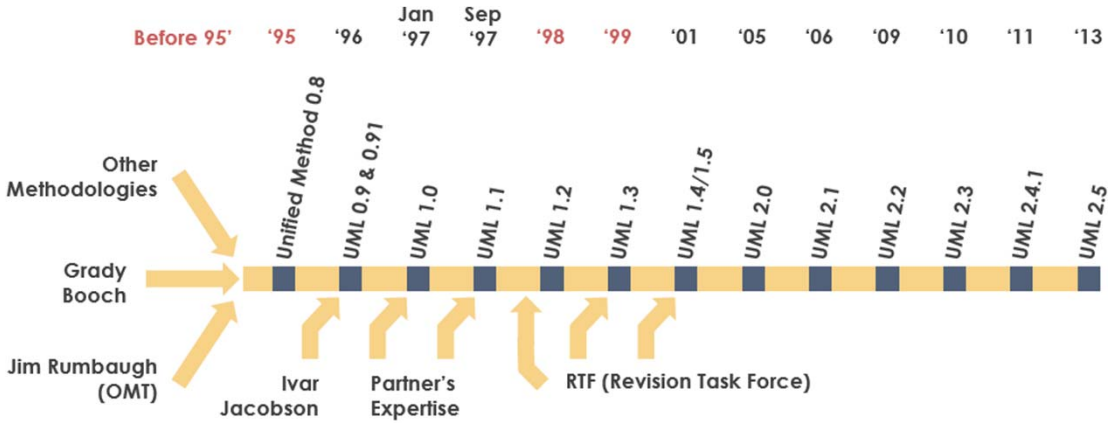
UML

Unified Modeling Language

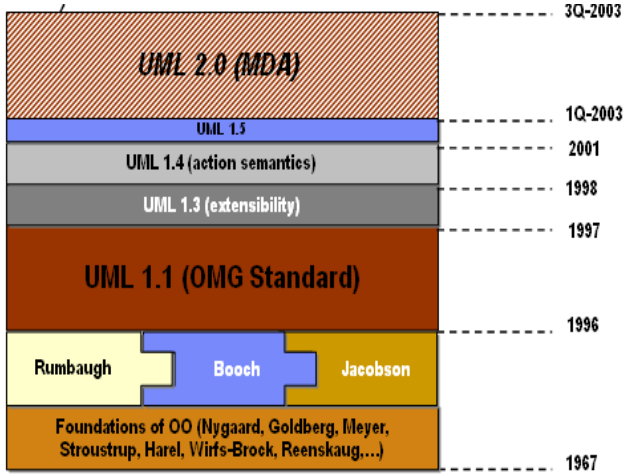
The Unified Modeling Language™ (UML™) is the industry-standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems. It simplifies the complex process of software design, creating a "blueprint" for construction.

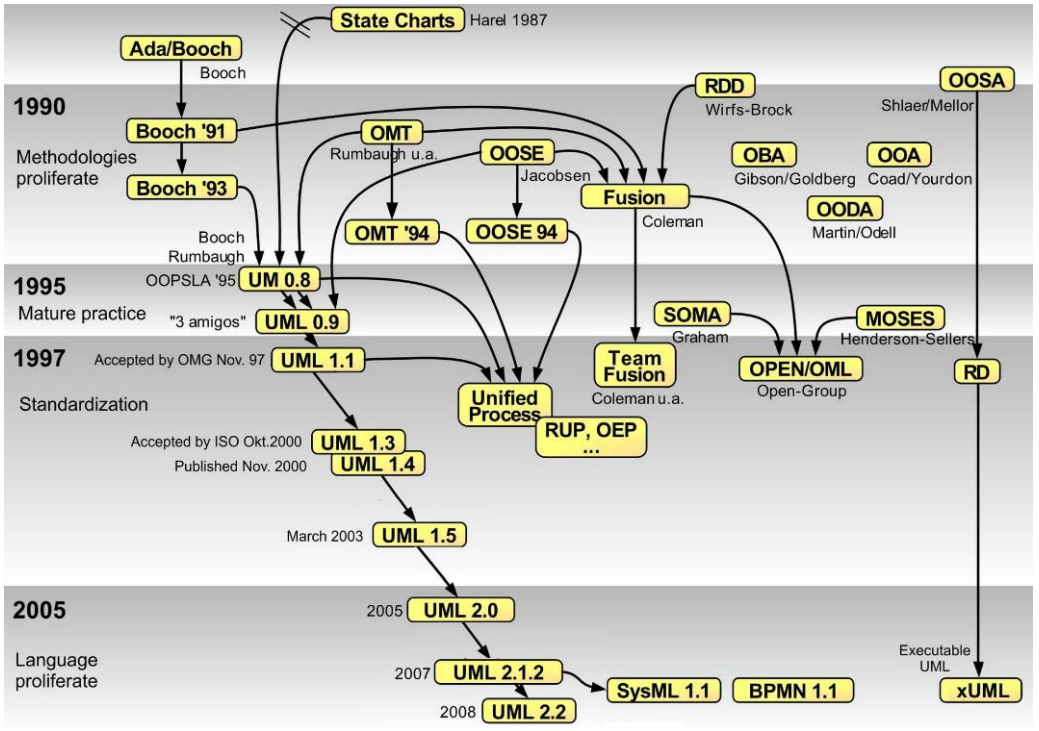
- Visualizar
- Especificar
- Construir
- Documentar

UML - Evolución



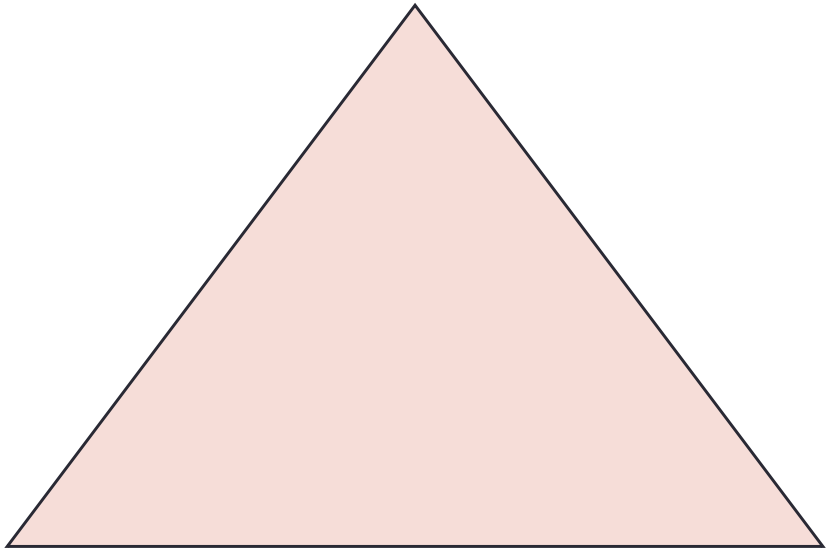
Before 95' - Fragmentation ▶ 95' - Unification ▶ 98' - Standardization ▶ 99' - Industrialization





UML - Triángulo del éxito

UML → **Notación**



Proceso (metodología)

RUP, USDP,

C. Larman, SCRUM,

XP, …

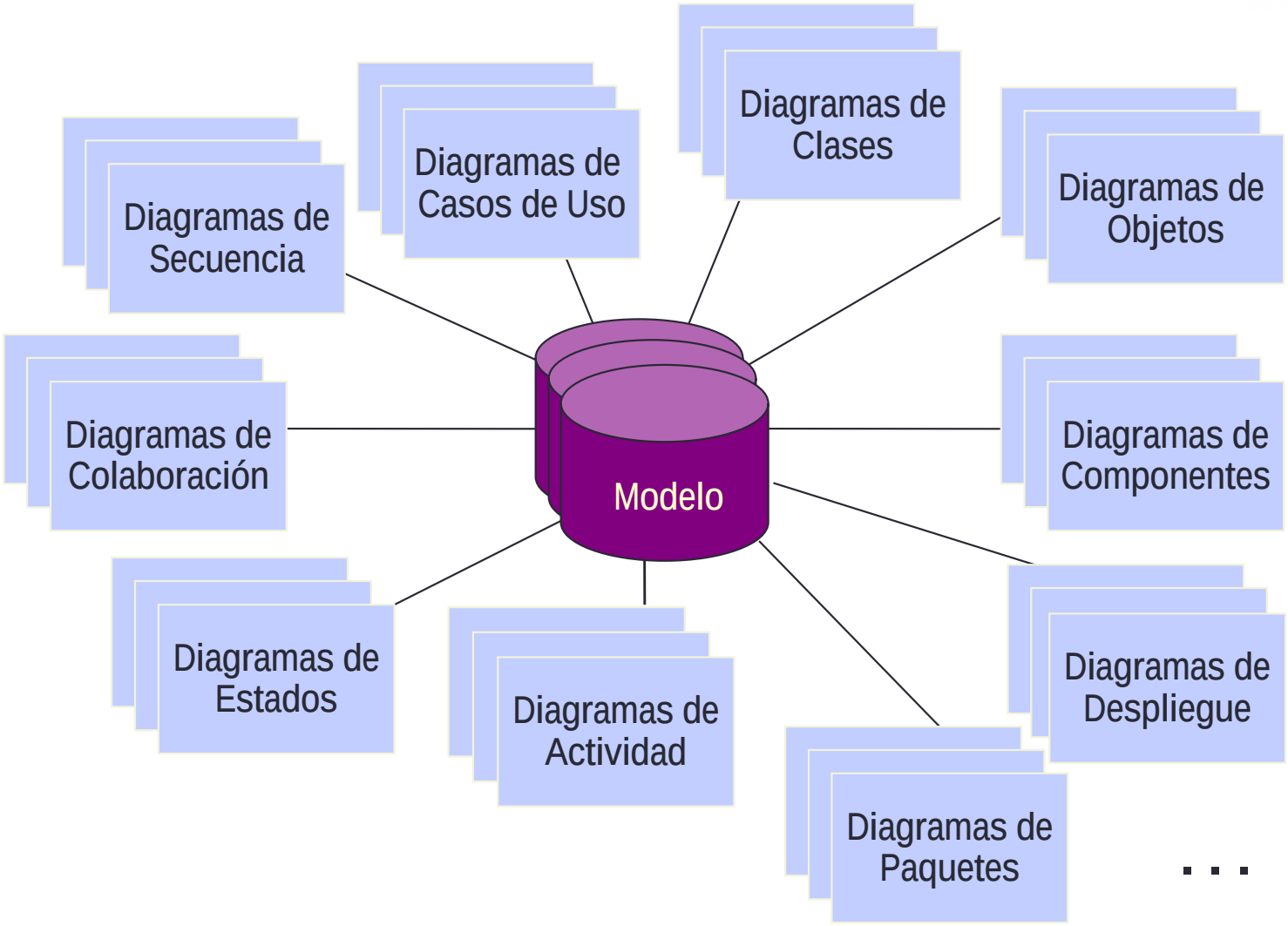
Herramienta

Rational Rose

Objecteering,

Visio, etc.

UML


Diagramas de Clases
Diagramas de Casos de Uso
Diagramas de Secuencia
Diagramas de Objetos
Diagramas de Colaboración
Modelo
Diagramas de Componentes
Diagramas de Estados
Diagramas de Actividad
Diagramas de Despliegue
Diagramas de Paquetes
. . .

UML - Tipos de diagramas

Diagrama de Clase (incluyendo Diagrama de Objetos). **Parte 1**

Diagrama de Casos de Uso. **Parte 2**

Diagramas de Comportamiento

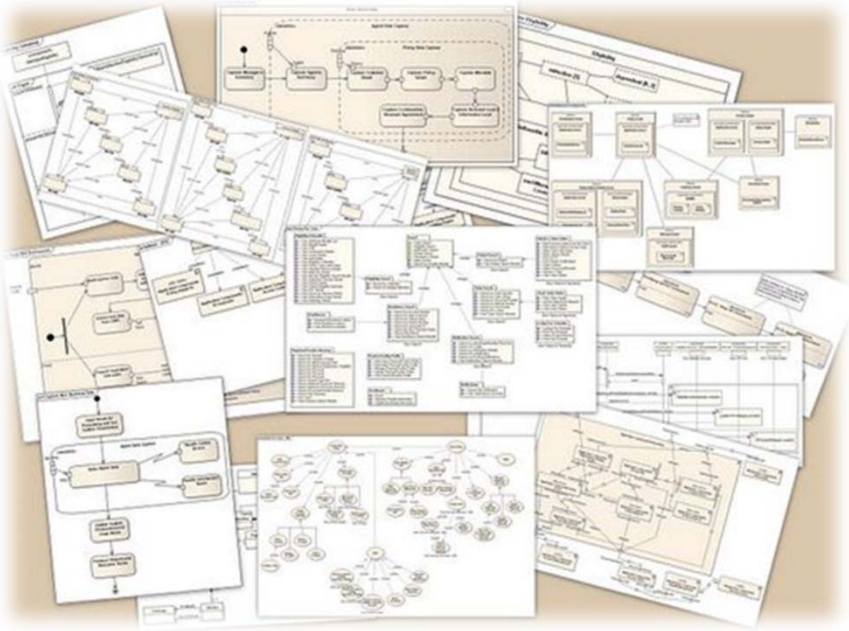
- Diagrama de Estados
- Diagrama de Actividad

Diagramas de Interacción

- Diagrama de Secuencia
- Diagrama de Colaboración

Diagramas de implementación

- Diagrama de Componentes
- Diagrama de Despliegue



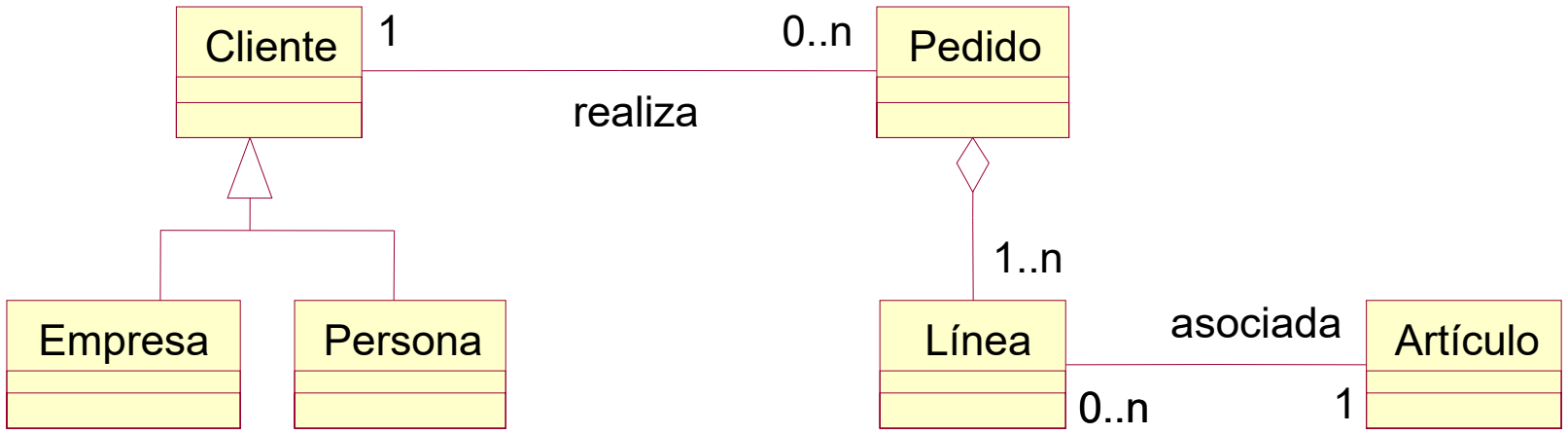
Parte 1
Diagrama de Clases

- Clases – Objetos
- Relaciones entre clases
  - Asociación
  - Agregación
  - Composición
  - Especialización/ Generalización (Herencia)

Diagrama de Clases

- Muestra la estructura estática del sistema, mostrando las clases y las relaciones entre ellas
- Es la herramienta principal de la mayor parte de los métodos OO

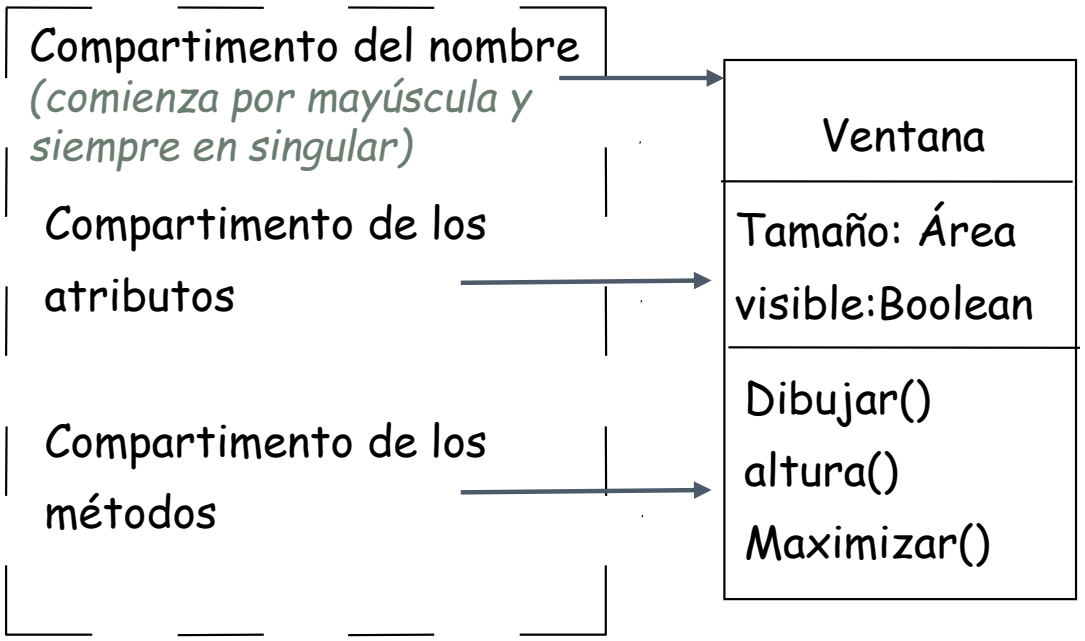
Notación



Clase

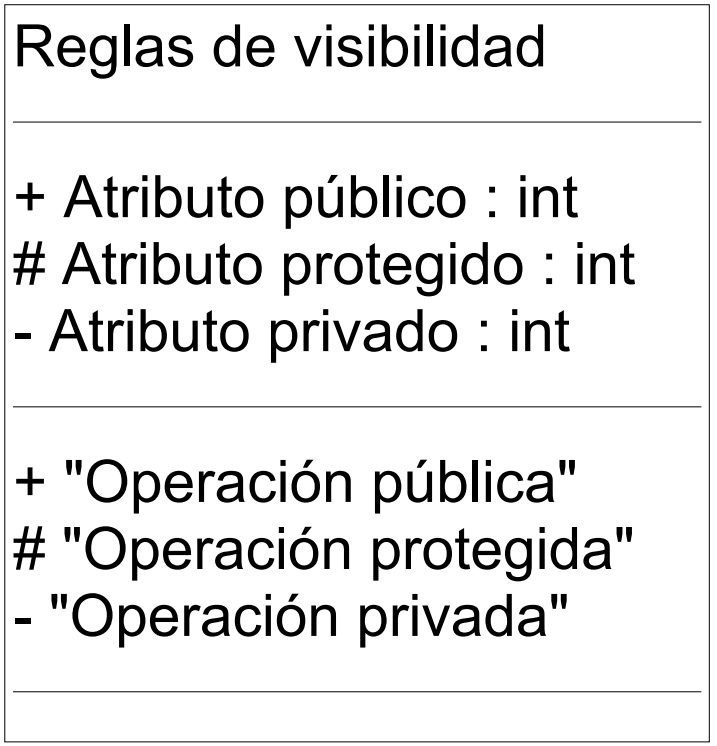
Es la descripción de un grupo de objetos con estructura, comportamiento y relaciones similares

Notación



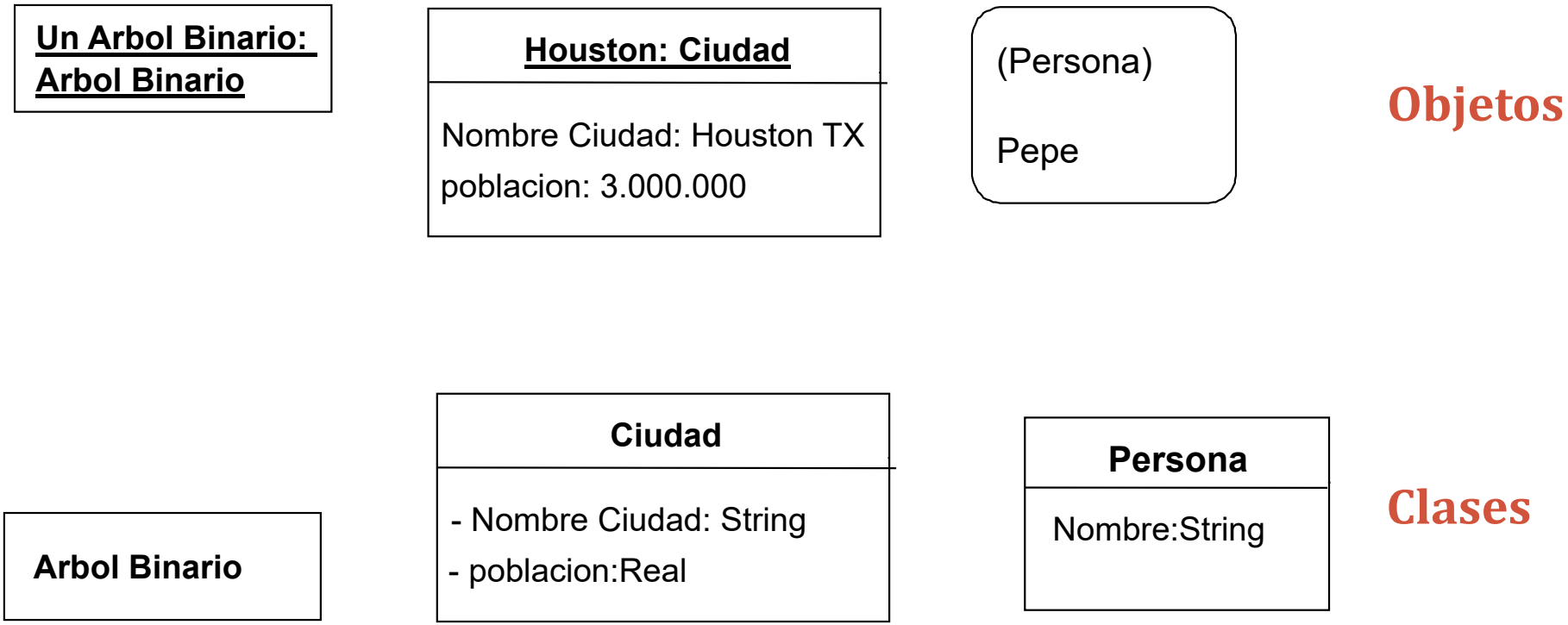
Clase

- Los atributos/operaciones pueden ser:
  - **(-) Privados**
  - **(#) Protegidos**
  - **(+) Públicos**



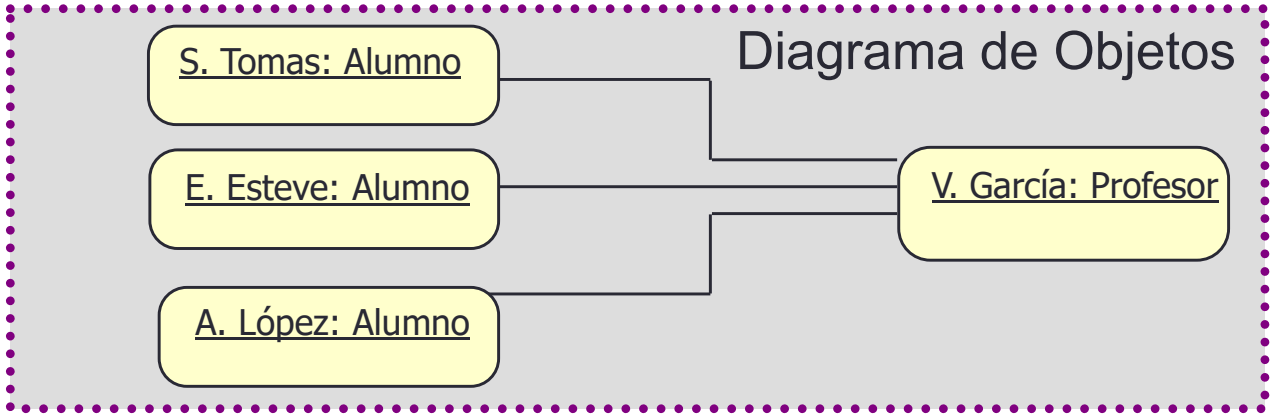
- Los atributos se pueden representar mostrando únicamente el nombre
- Los atributos no incluyen referencia a otros objetos, estas referencias se representan mediante enlaces
- Un **atributo derivado** se representa como /Atributo : Tipo
- Un método es la implementación de una operación

Clases / Objetos

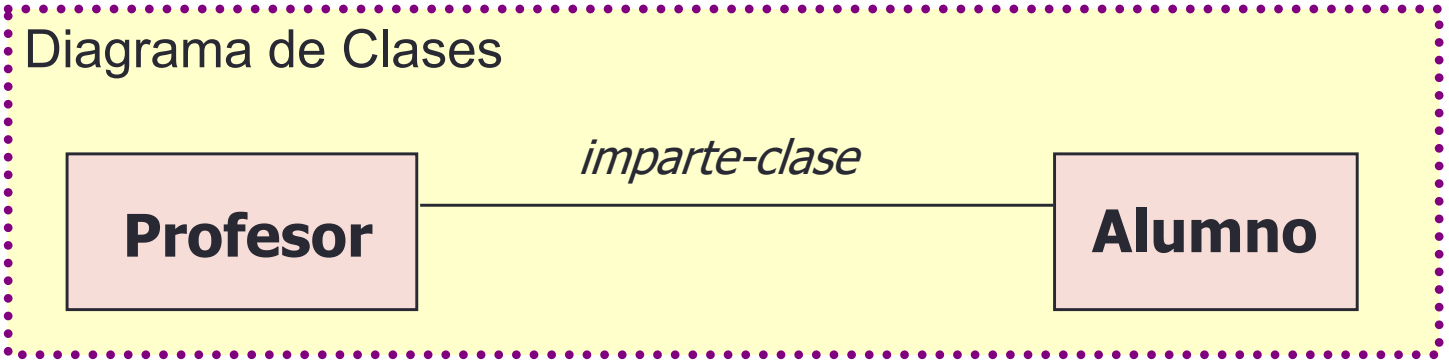


Asociaciones

- Un enlace es una conexión física o conceptual entre objetos
- Una asociación es una relación estructural que especifica que los objetos de un elemento están conectados con los objetos de otro



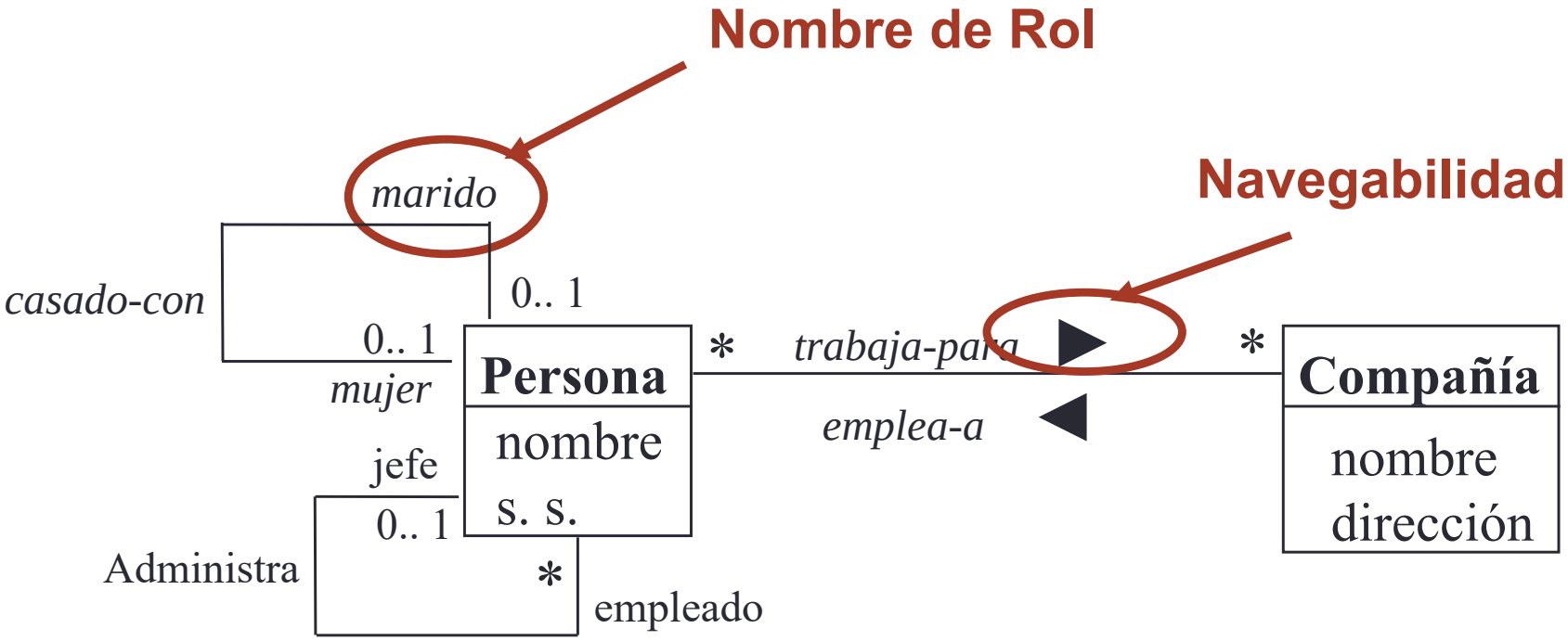
Enlace



Asociación

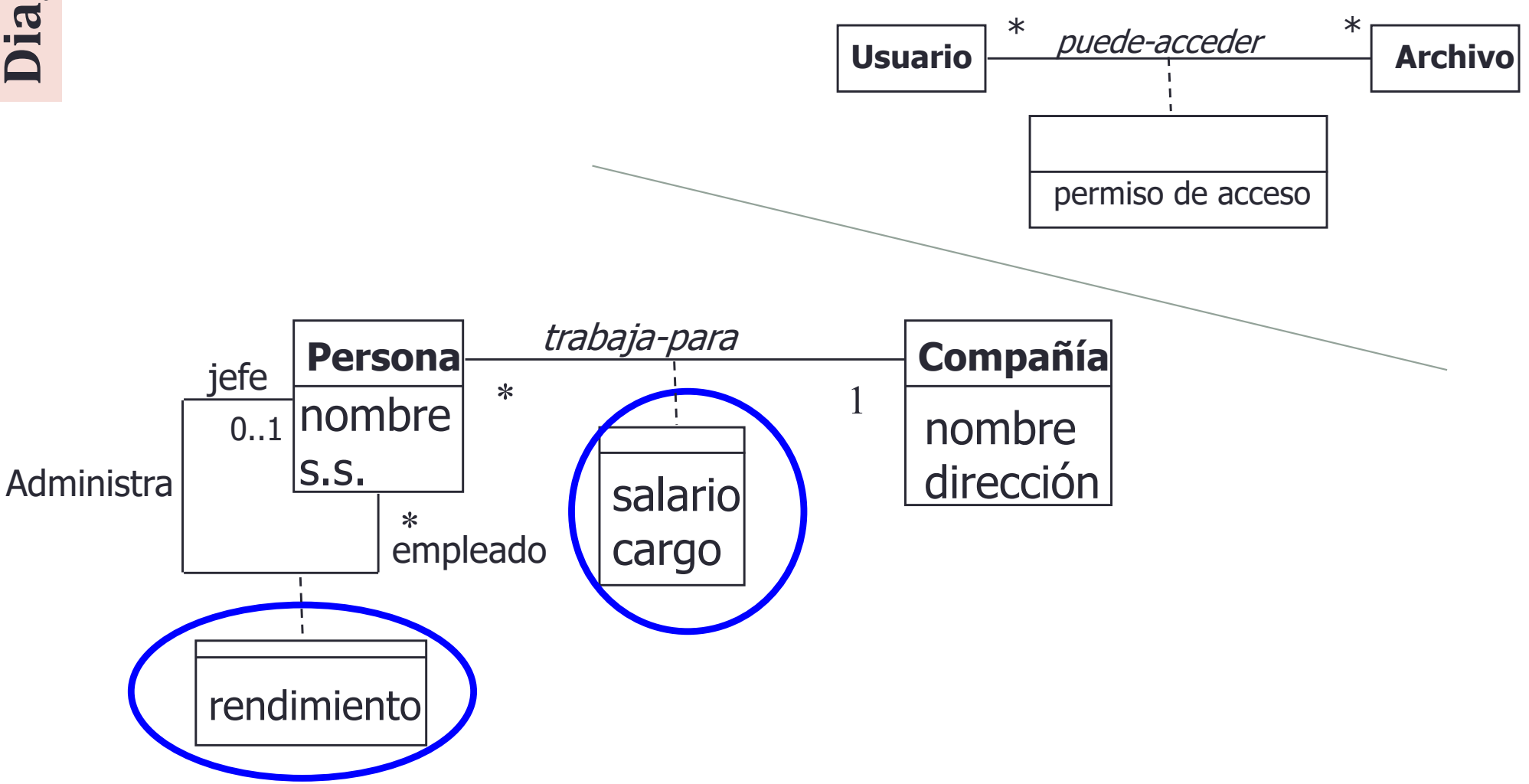
Asociaciones

- Toda asociación es bidireccional, es decir, se puede navegar en los dos sentidos, desde objetos de una clase a objetos de la otra.
- Tiene un nombre
- Puede tener nombres de rol en los extremos (obligatorio en asociaciones reflexivas)
- Multiplicidad: 1, 0..1, 0..N (*), 1..N, M..N



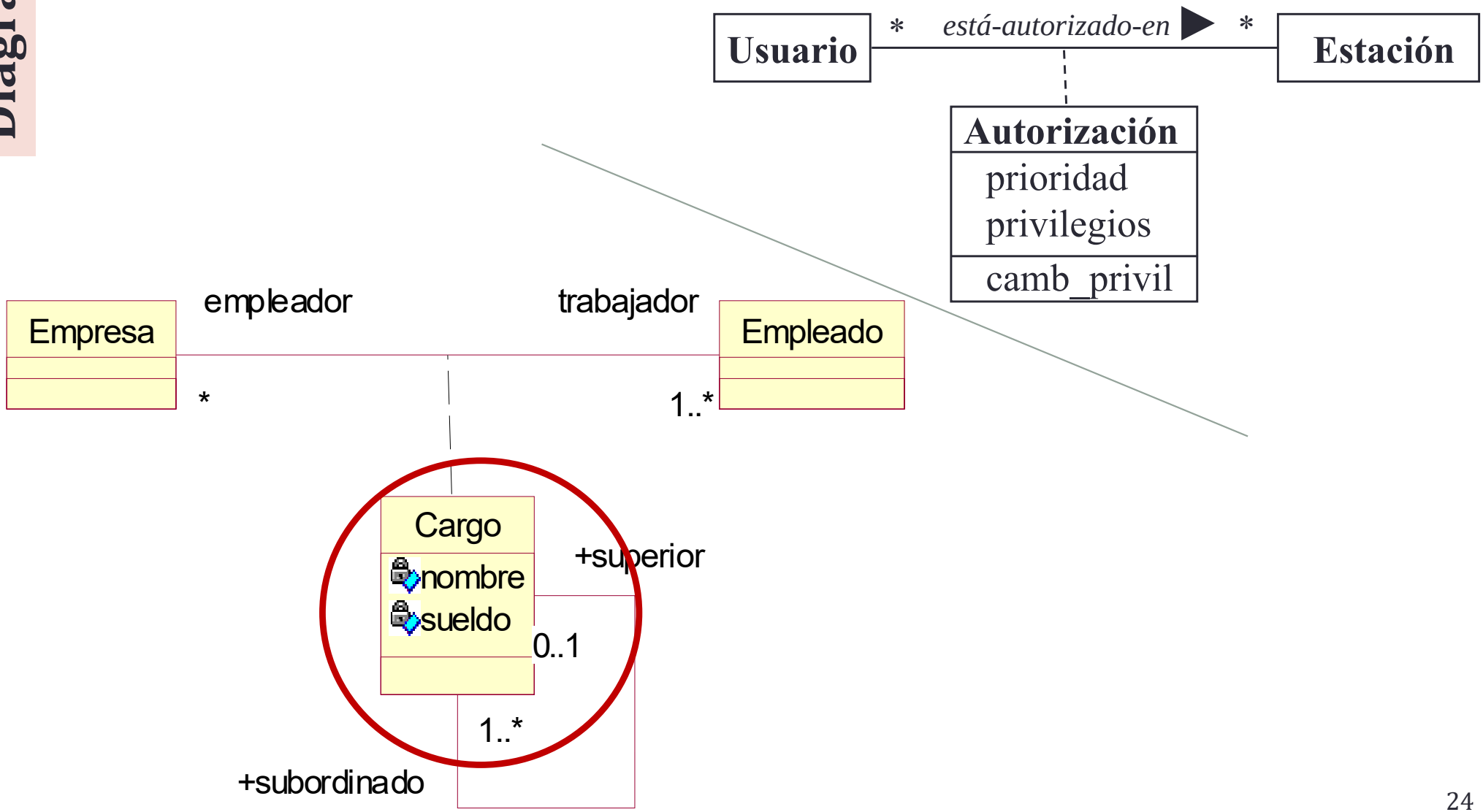
Asociaciones - Atributos de Enlace

En una asociación entre clases, la propia relación puede tener propiedades, denominados atributos de enlace



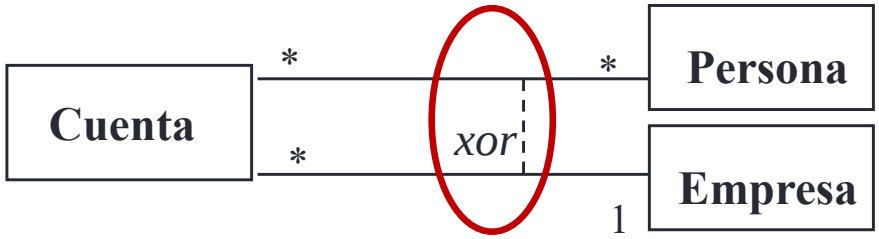
Asociaciones - Clase Asociación

El atributo de enlace puede ser una clase



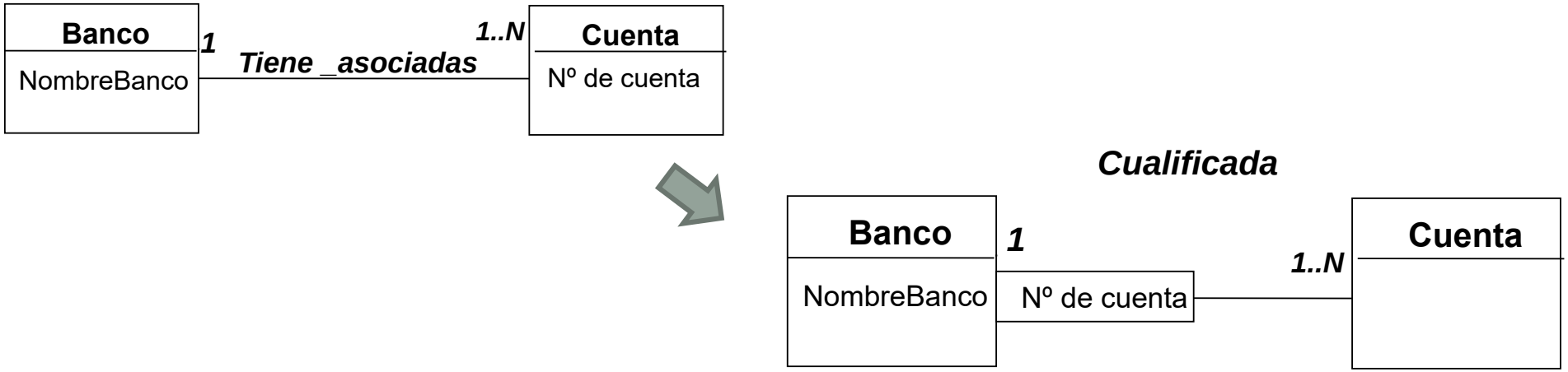
Asociaciones - Asociación excluyente

XOR



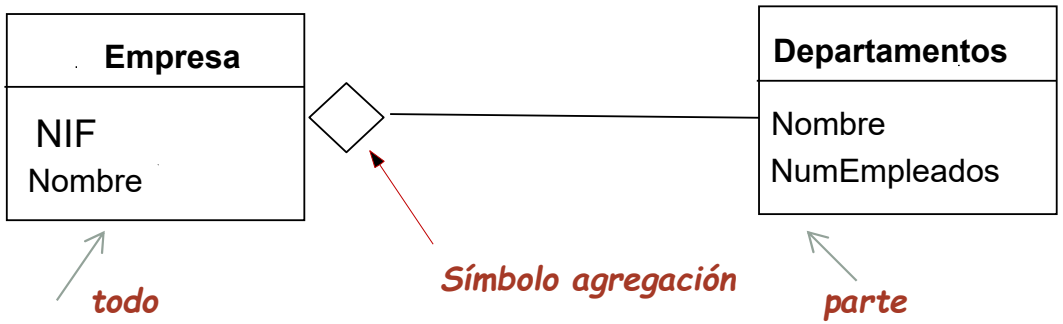
Asociaciones - Asociación cualificadas

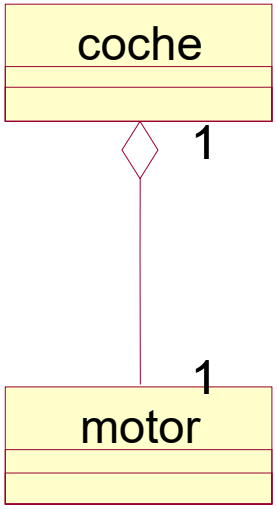
Cualificadores o calificadores nos sirven para refinar más el modelo, indicando el índice para recorrer la relación *(¿Cómo identificar un objeto o conjunto de objetos en el otro extremo?)*

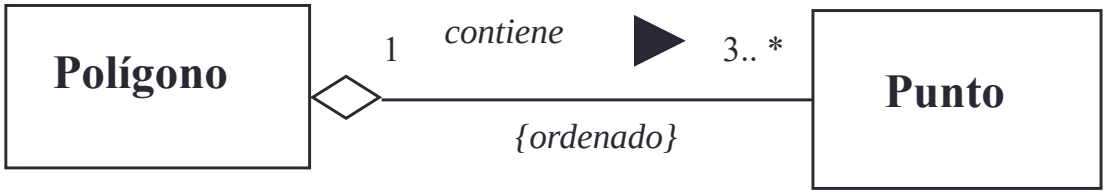


Agregación

- Es una asociación con unas propiedades semánticas adicionales.
- "*está formado por*"







Agregación

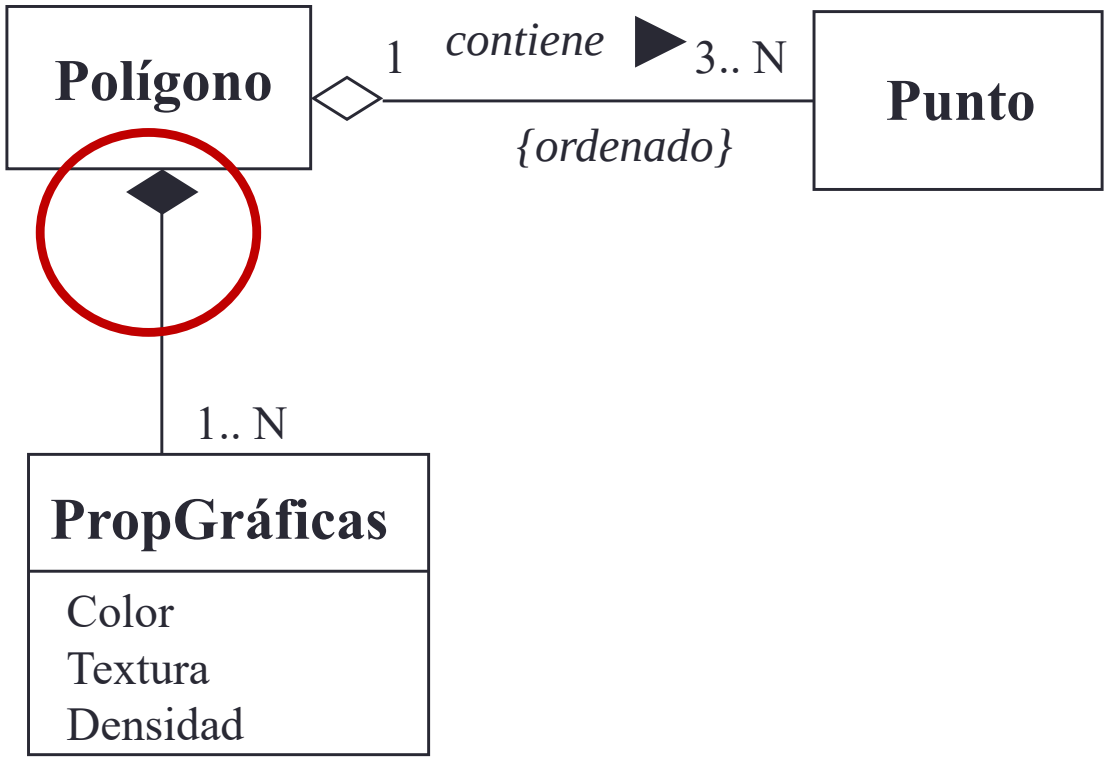
Tipos de agregación:

Inclusiva o física: cada componente puede pertenecer a lo sumo a un compuesto. La destrucción del compuesto implica la destrucción de las partes.

Referencial o de catálogo: los componentes son reutilizables a lo largo de distintos compuestos. No están relacionados los tiempos de vida.

Composición

- Es una agregación inclusiva o física
- "*está compuesta por*"

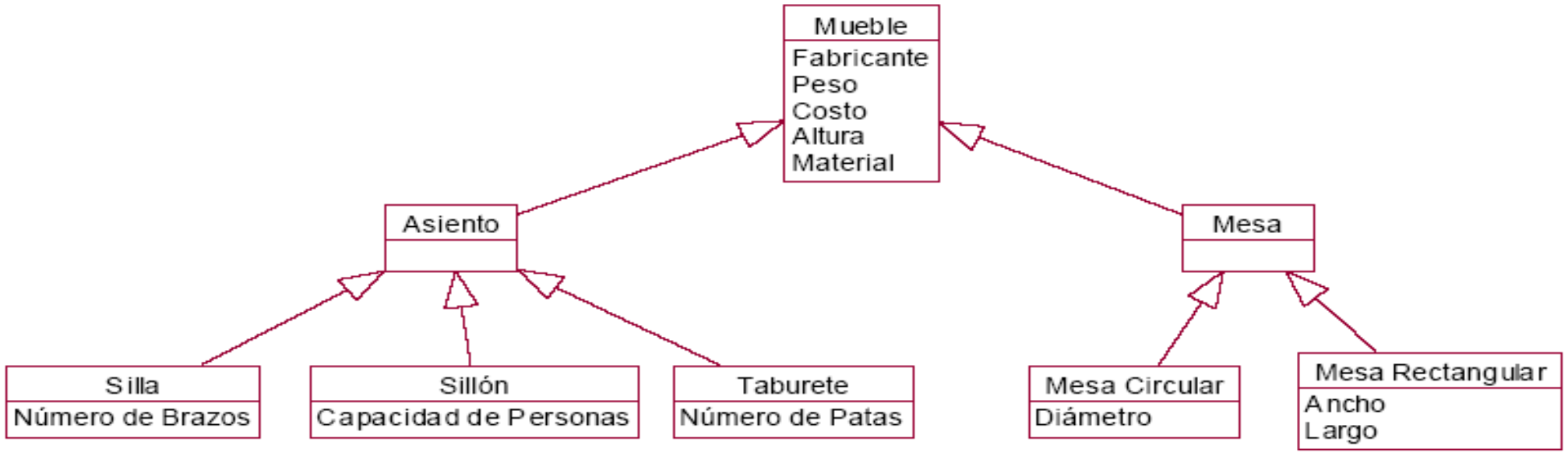


Diferencias entre Composición y Agregación

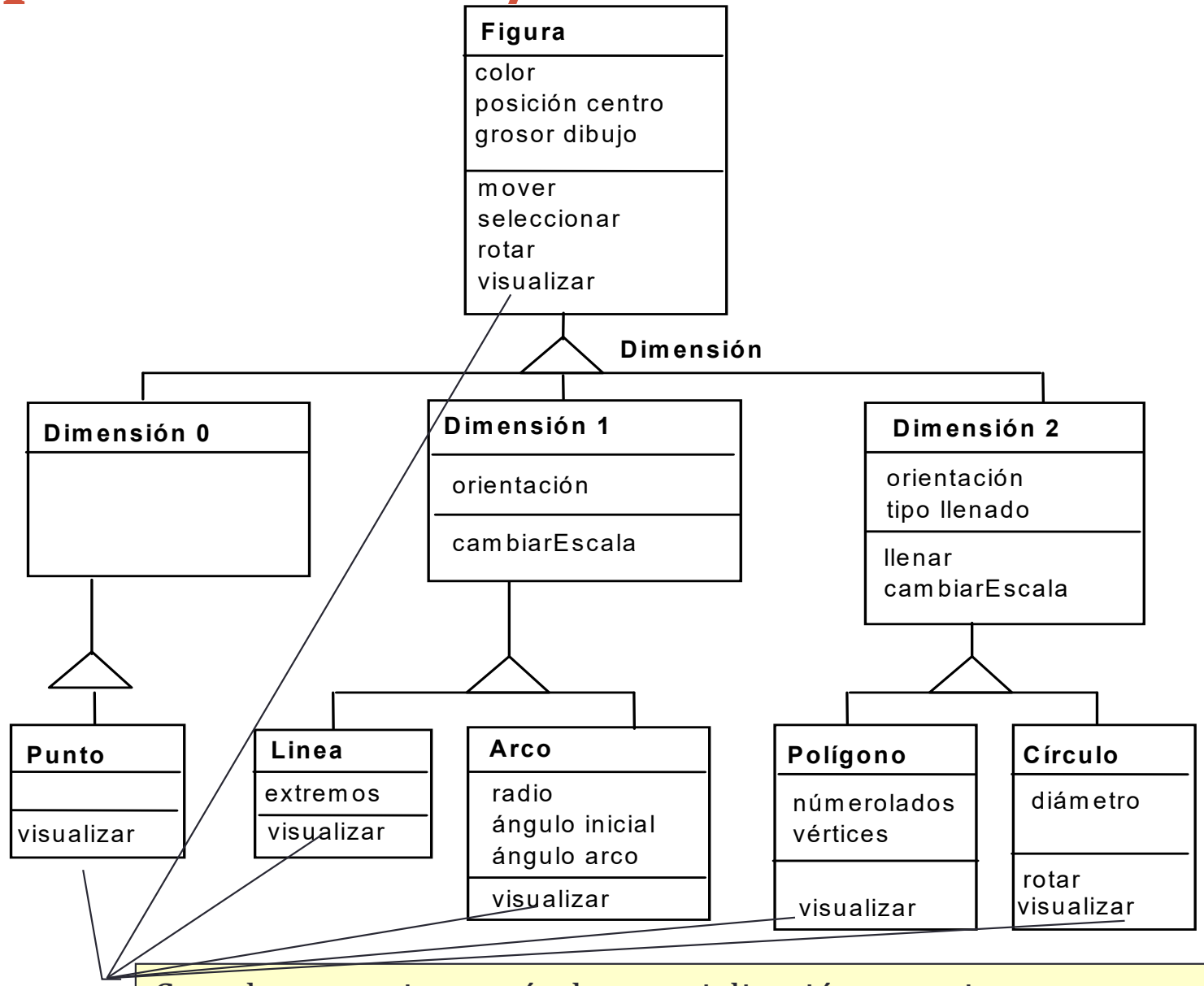
	Agregación	Composición
Varias asociaciones comparten los componentes	Sí	No
Destrucción de los componentes al destruir el compuesto	No	Sí
Cardinalidad a nivel de compuesto	Cualquiera	0..1 ó 1
Representación		

Especialización / Generalización

- Permiten definir jerarquías de clases
- Generalización: Dado un conjunto de clases, si tienen en común atributos y métodos, se puede crear una clase más general (superclase) a partir de las iniciales (subclases)
- Especialización: es la relación inversa
- "es un"



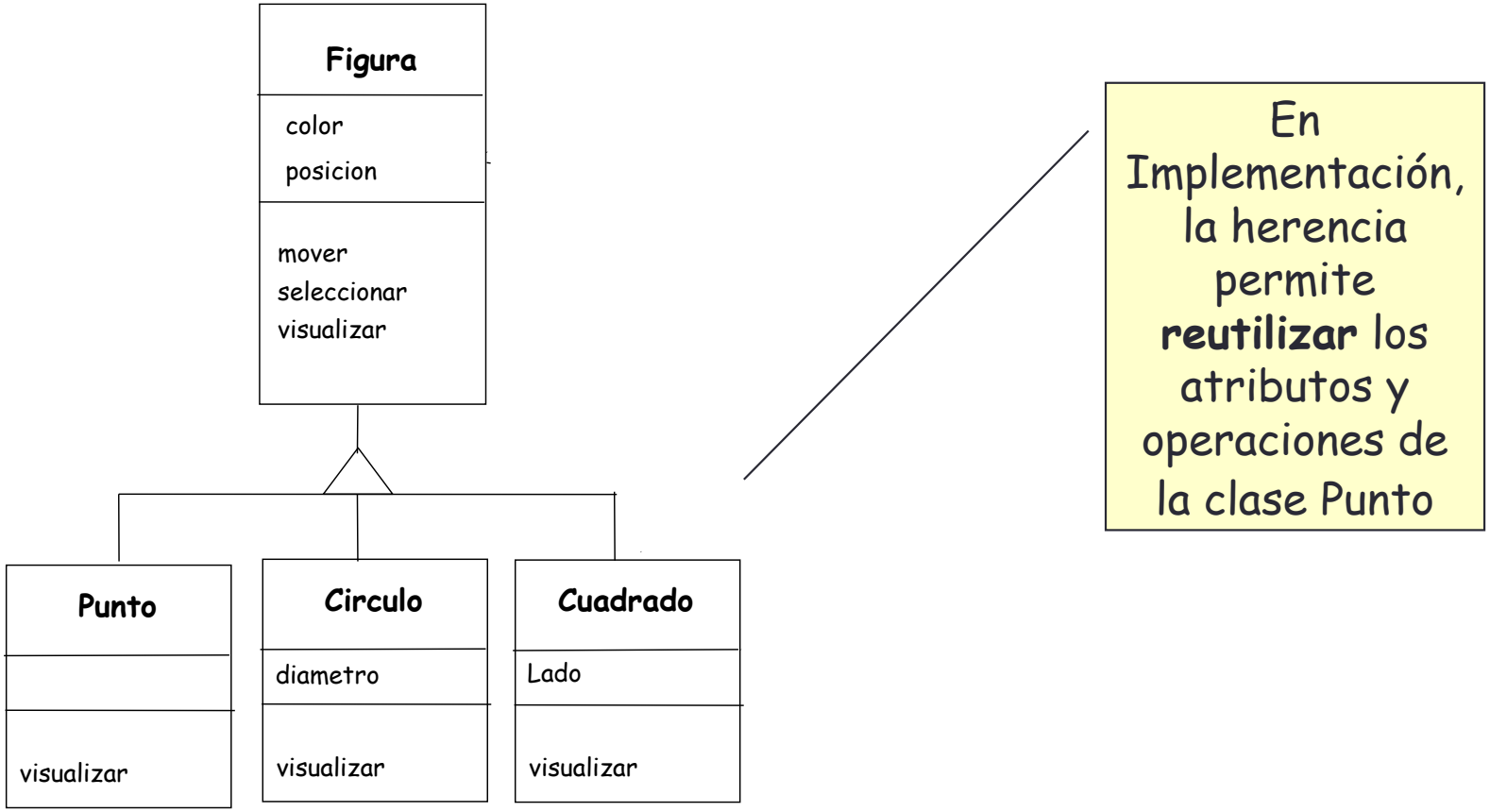
Especialización / Generalización



Cuando en una jerarquía de especialización se repite una característica de una clase (atributo u método) estamos **redefiniendo** la característica heredada

Especialización / Generalización

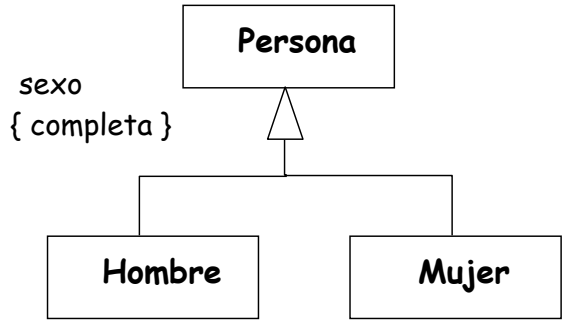
La relación de **especialización** se emplea en la fase de modelado de un sistema, mientras que la relación de **herencia** se ve como un mecanismo de reutilización de código en la fase de implementación o diseño.



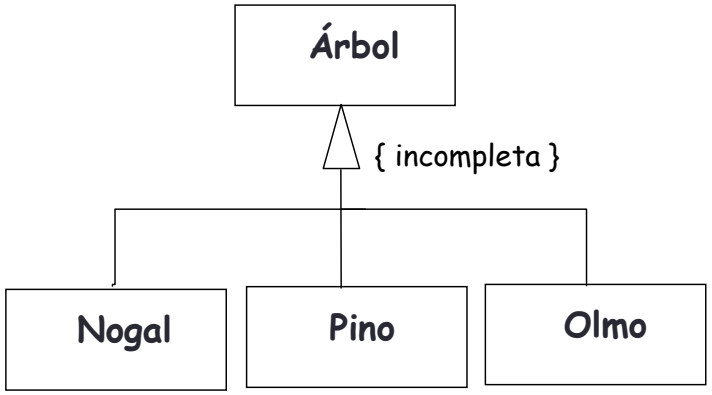
Especialización / Generalización

Dos tipos de **restricciones**:

- **Completa**: Todos los hijos de la generalización se han especificado en el modelo

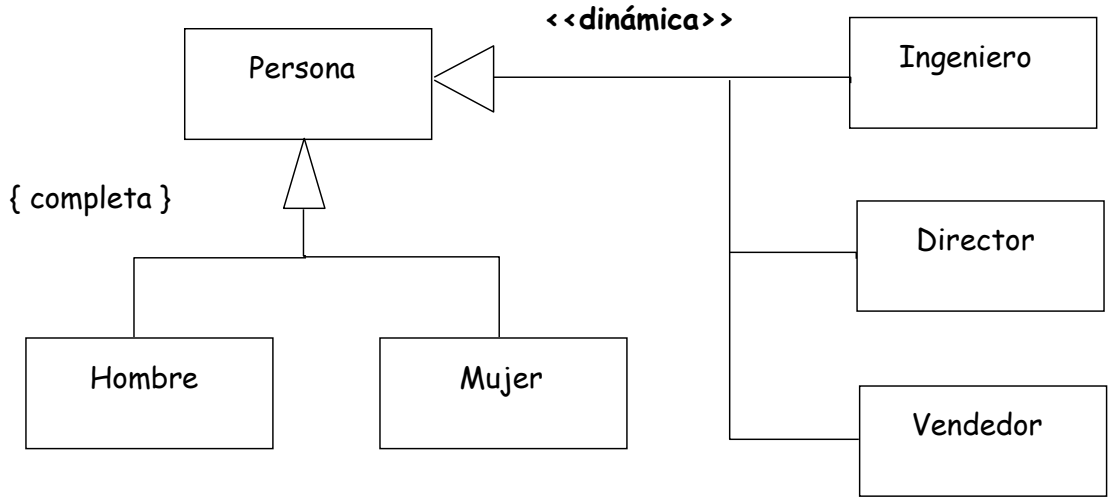


- **Incompleta**: No se han especificado todos los hijos y se permiten hijos adicionales

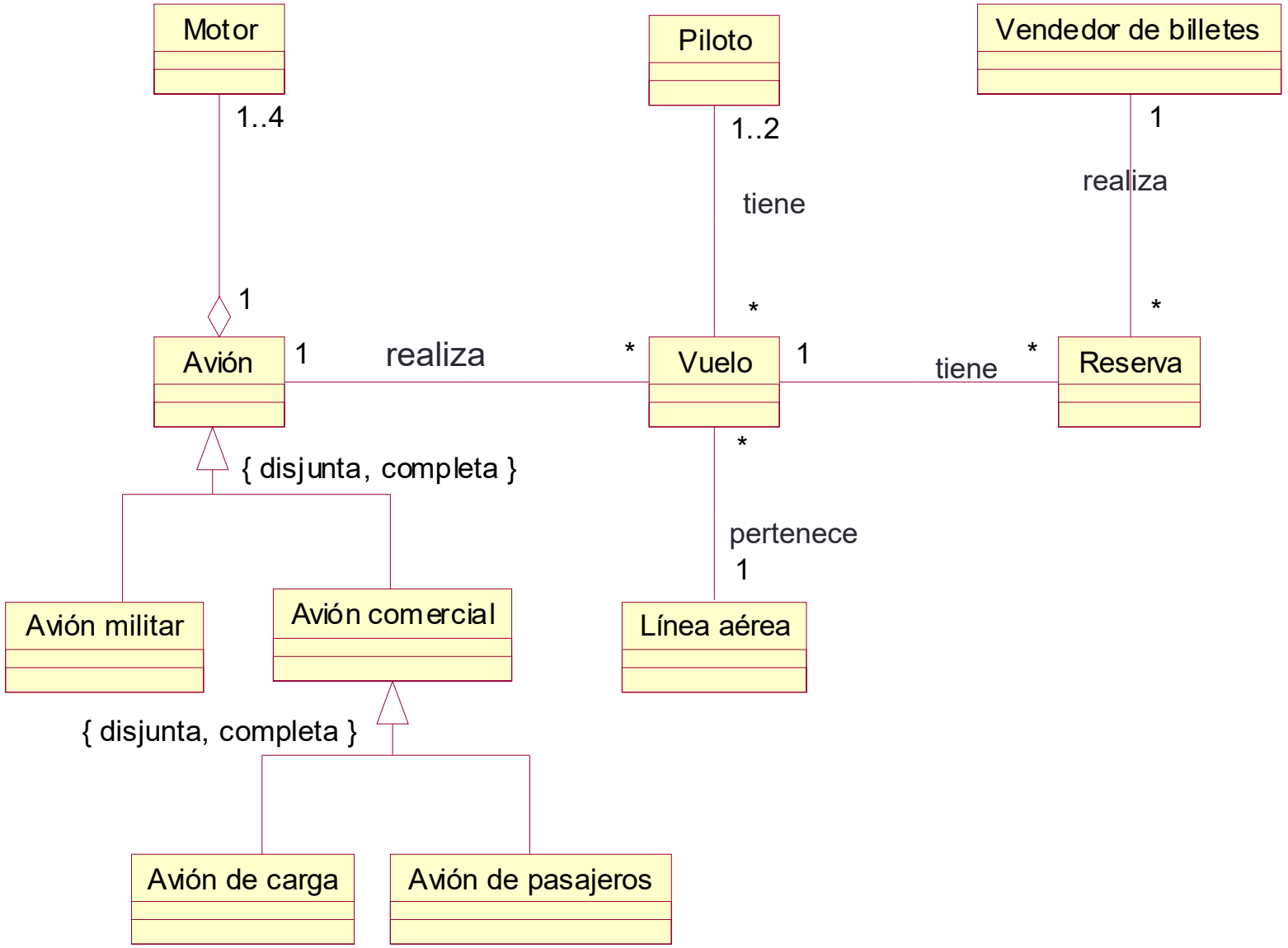


Especialización / Generalización

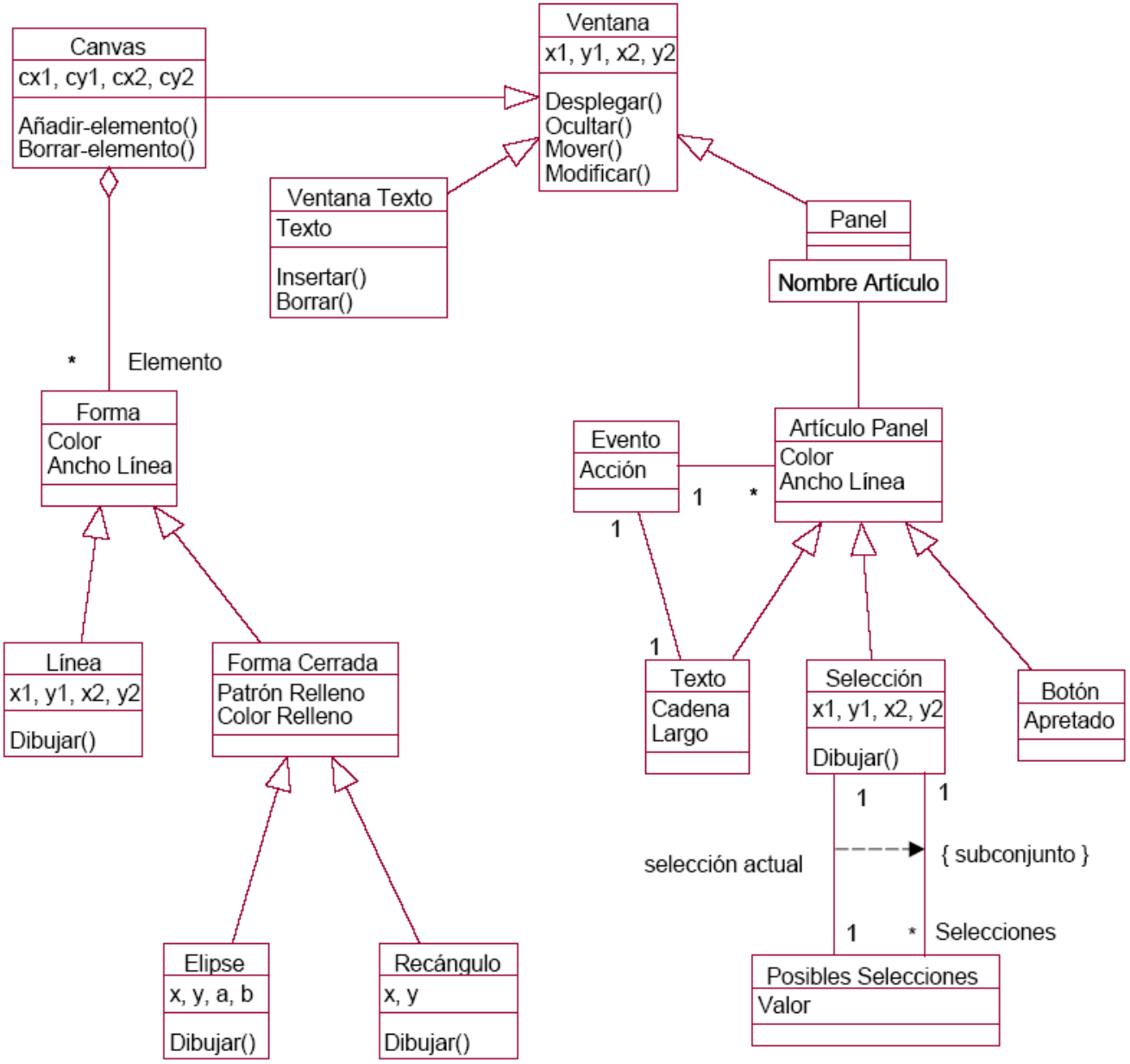
Se habla de especialización **dinámica** cuando un objeto puede cambiar de clase dentro de una jerarquía de subclases



Ejemplos


Motor
Piloto
Vendedor de billetes
1..4
1..2
1
realiza
tiene
1
*
*
Avión
1
realiza
*
Vuelo
1
tiene
*
Reserva
{ disjunta, completa }
*
pertenece
1
Avión militar
Avión comercial
Línea aérea
{ disjunta, completa }
Avión de carga
Avión de pasajeros

Ejemplos


Canvas
cx1, cy1, cx2, cy2
Añadir-elemento()
Borrar-elemento()
Ventana
x1, y1, x2, y2
Desplegar()
Ocultar()
Mover()
Modificar()
Ventana Texto
Texto
Insertar()
Borrar()
Panel
Nombre Artículo
* Elemento
Forma
Color
Ancho Línea
Evento
Acción
Artículo Panel
Color
Ancho Línea
1
*
1
1
Línea
x1, y1, x2, y2
Dibujar()
Forma Cerrada
Patrón Relleno
Color Relleno
Texto
Cadena
Largo
Selección
x1, y1, x2, y2
Dibujar()
Botón
Apretado
1
1
selección actual
{ subconjunto }
1
*
Selecciones
Elipse
x, y, a, b
Dibujar()
Recángulo
x, y
Dibujar()
Posibles Selecciones
Valor