Curso 2020-2021

Ingeniería del Software

MODELADO

¿Qué es un modelo?

- **Abstracción o simplificación de la realidad**: divide y vencerás
- Diversos tipos de modelos:
  - estructura, electricidad, saneamiento…
  - estático, dinámico...
    - ➔**¿cómo se relacionan entre sí?**
- Modelos formales y modelos informales
  - **Modelos informales**: *ad hoc*, sin lenguaje común
  - **Modelos formales**: lenguaje universal, precisión, rigor, coherencia
- Modelado y lenguaje
  - El lenguaje es vehículo del pensamiento: ayuda a **pensar con claridad**
  - El modelado es un **elemento esencial** del proceso de desarrollo de software
  - El modelado requiere un **lenguaje adecuado**
- Modelar no es hacer diagramas sino **pensar con diagramas**

Propiedades deseables de un modelo

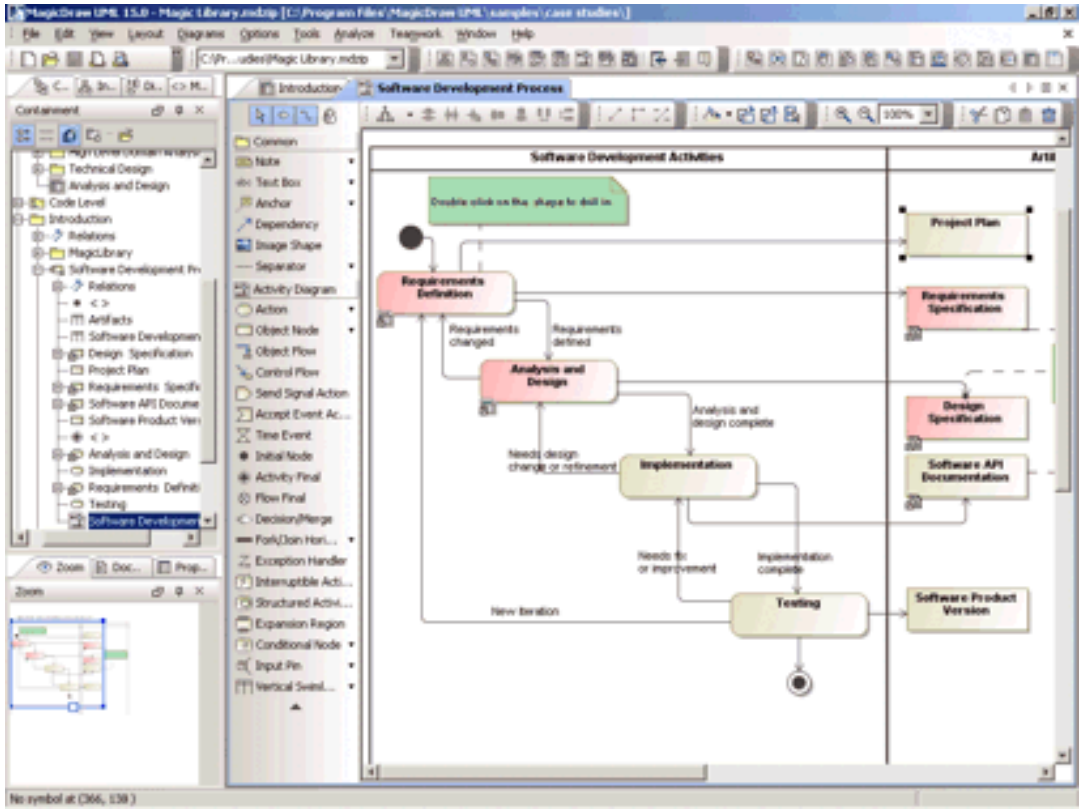
- **Comprensible**
  - Expresado de tal forma que se pueda entender fácilmente.
- **Preciso**
  - Representa fielmente el sistema modelado.
- **Predictivo**
  - Se puede utilizar para obtener conclusiones correctas sobre el sistema.
- **Barato**
  - Más económico que construir y estudiar el propio sistema.

Sistema, modelo, diagrama

- Un **sistema** es una colección de elementos organizados para cumplir una **finalidad** concreta
  - Un sistema puede estar dividido en subsistemas
- Un **modelo** es una **abstracción** de un sistema, es decir, una simplificación (completa y consistente) del sistema real, que sirve para comprenderlo mejor
  - Provisionalmente, un modelo puede ser incompleto (faltan elementos) o inconsistente (contiene contradicciones)
  - Un sistema puede estar modelado desde distintos puntos de vista complementarios, según lo que se considere relevante en cada caso
- Un **diagrama** es la representación gráfica de un conjunto de elementos interconectados, una **vista** parcial de un modelo
  - Un modelo no es meramente una colección de diagramas
  - Un modelo puede contener elementos no representados en un diagrama
  - Un modelo puede contener especificaciones textuales esenciales

Herramientas de modelado

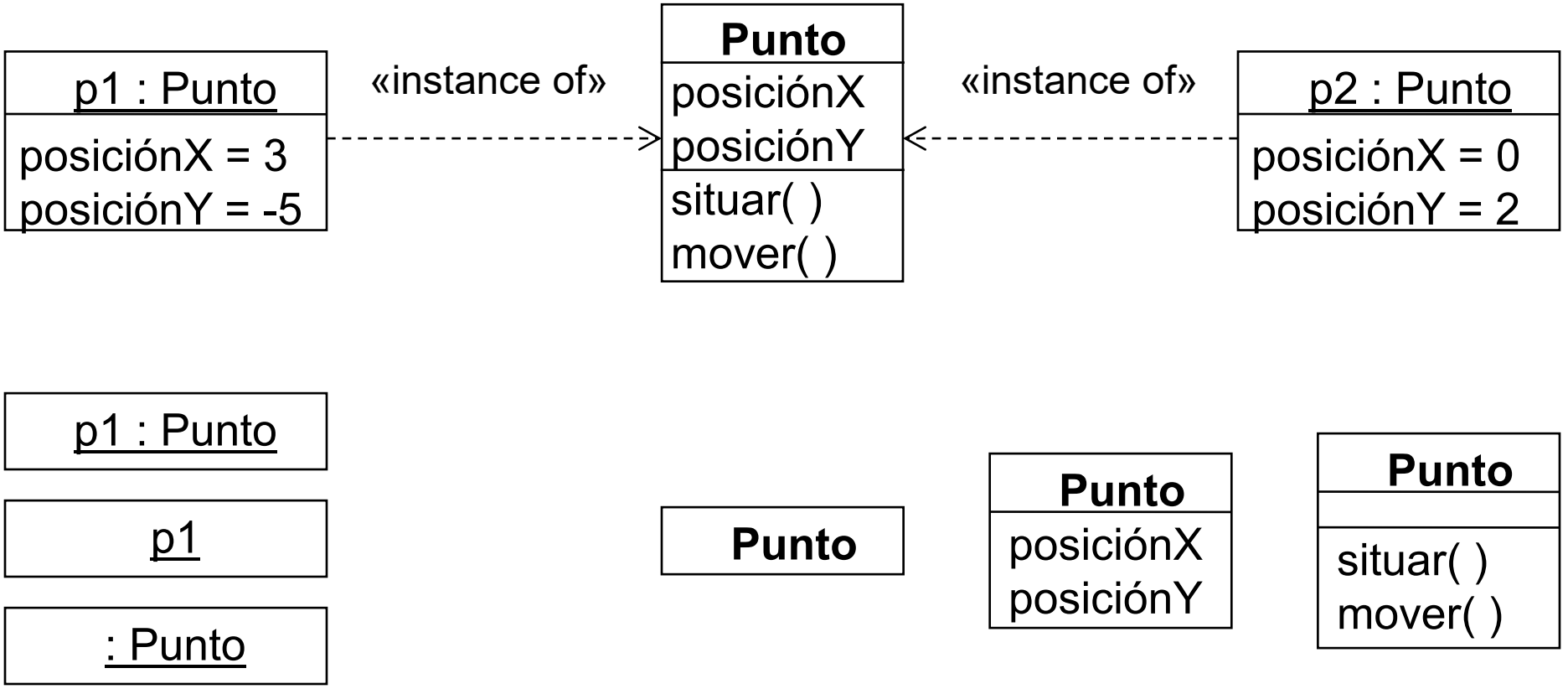
- ¿Qué puede ofrecer una herramienta CASE para UML?
  - Dibujo
  - Corrección sintáctica
  - Coherencia entre diagramas
  - Integración con otras aplicaciones
  - Trabajo multiusuario
  - Reutilización
  - Generación de código...
- Ejemplos
  - Visual Paradigm
  - Modelio
  - LucidChart



Objetos y clases

- Dos niveles de abstracción:
  - **objeto**: una **entidad concreta** con identidad, estado y comportamiento
  - **clase**: un **conjunto de entidades** con estructura y comportamiento comunes
- La relación de clasificación / instanciación
  - un objeto es **instancia** de una clase
  - la clase se usa como **plantilla** para construir (instanciar) objetos
- Objetos y clases en análisis y diseño:
  - Análisis = especificación, vista externa, caja negra
    - clases, atributos y operaciones corresponden a **conceptos del dominio**
    - es habitual usar una **notación simplificada** al máximo
  - Diseño = implementación, vista interna, caja blanca
    - clases, atributos y operaciones corresponden a **fragmentos de código**
    - nuevos artefactos y soluciones que dependen del **lenguaje** y la **plataforma** de implementación y no tienen por qué corresponder a conceptos del dominio

Notación básica de objetos y clases



Tipos de clases

- Tipos de clases según los objetos representados:
  - objetos **físicos**: avión, persona, libro...
  - objetos **lógicos**: cuenta corriente, asignatura, número complejo…
  - objetos **históricos**: asiento bancario, reserva de habitación…
- Para entender lo que es una clase, hace falta entender cuáles serán sus instancias. Un caso especial lo constituyen los objetos que representan una **colección, familia o tipo** de cosas, más que una cosa en sí misma.
- Ejemplos: raza perruna, producto a la venta, título en la biblioteca.
- Es decir, un mismo concepto del mundo real puede ser modelado como objeto o clase según el contexto:
  - “Pastor Alemán”
  - “Lata de Atún”
  - “El Lenguaje Unificado de Modelado”
- Todo objeto representa una “entidad concreta”, pero esto no significa necesariamente “entidad física” o tangible.

Clase vs. Tipo de dato

	Clase	**Tipo de dato (<<Datatype>>)**
Concepto	Representa un concepto definido **dentro de los límites del sistema**	Representa un concepto general e independiente, definido **fuera de los límites del sistema**
Población	La población es **finita** y **variable**: existen tantas instancias como explícitamente sean creadas o destruidas en tiempo de ejecución	La población es **finita o infinita, pero constante**: las instancias existen implícitamente, sin necesidad de crearlas, y no pueden ser destruidas Población finita: días, meses. Población infinita: años.
Estructura	Pueden ser **simples o estructuradas** (lo normal es que tengan varios atributos)	Pueden ser **simples o estructuradas** (ambos casos son normales): número entero, fecha dd/mm/aaaa Si es estructurado, entonces la población se obtiene como **producto cartesiano** de sus tipos de dato constitutivos, posiblemente con restricciones (fecha)
Valores	Los valores de los atributos pueden ser **fijos o variables** (más normal variables)	Los valores simples o estructurados son **fijos e inmutables**, el sistema no puede modificarlos
Comparación	Por **referencia** (“son el mismo”)	Por **valor** (“son iguales”)
Asociaciones	Puede participar en asociaciones **unidireccionales o bidireccionales**	Sólo en **asociaciones unidireccionales** (atributos): • ref. entrantes desde clase o tipo de dato propietario • si es estructurado, ref. salientes hacia tipo de dato
Copiado	Cuando se copia/clona un objeto, **no se copian los objetos referenciados**	Cuando se copia un valor estructurado, **se copian simultáneamente los valores referenciados**
Ejemplos	Usuario, ¿Día-de-Calendario?	Fecha, ¿Rol-de-Usuario?

Atributos

- Atributo: **propiedad compartida** por los objetos de una clase
  - cada atributo tiene un **valor** (probablemente diferente) para cada objeto
- Atributo **derivado** (concepto propio del **análisis**):
  - propiedad redundante que puede ser calculada a partir de otras
  - **/área** (= base * altura)
  - pueden implementarse como operaciones al pasar a **diseño**
- **Notación** (más importante en diseño)
  - pueden suprimirse todos los elementos excepto el nombre de atributo
    - visibilidad **nombre** multiplicidad : Tipo = valorInicial {propiedades}
  - propiedades predefinidas de los atributos: changeable, addOnly, frozen
  - ejemplos
    - saldo : Moneda = 0
    - teléfonoOficina [0..2] {addOnly}

Operaciones

- **Operación**: función o transformación que puede aplicarse a los objetos de una clase
  - pueden ser **invocadas** por otros objetos, o por el mismo objeto
  - **método**: especificación procedimental (implementación) de una operación
- **Notación** (más importante en diseño)
  - pueden suprimirse todos los elementos excepto el nombre de operación
    - visibilidad **nombre** (param: Tipo = valDef,…) : TipoRet {propiedades}
  - propiedades predefinidas de las operaciones: isQuery
  - ejemplos:
    - obtenerSaldo ( ) : Moneda {isQuery}
    - marcar (número : Teléfono; reintentos : Integer)

Enlaces y asociaciones

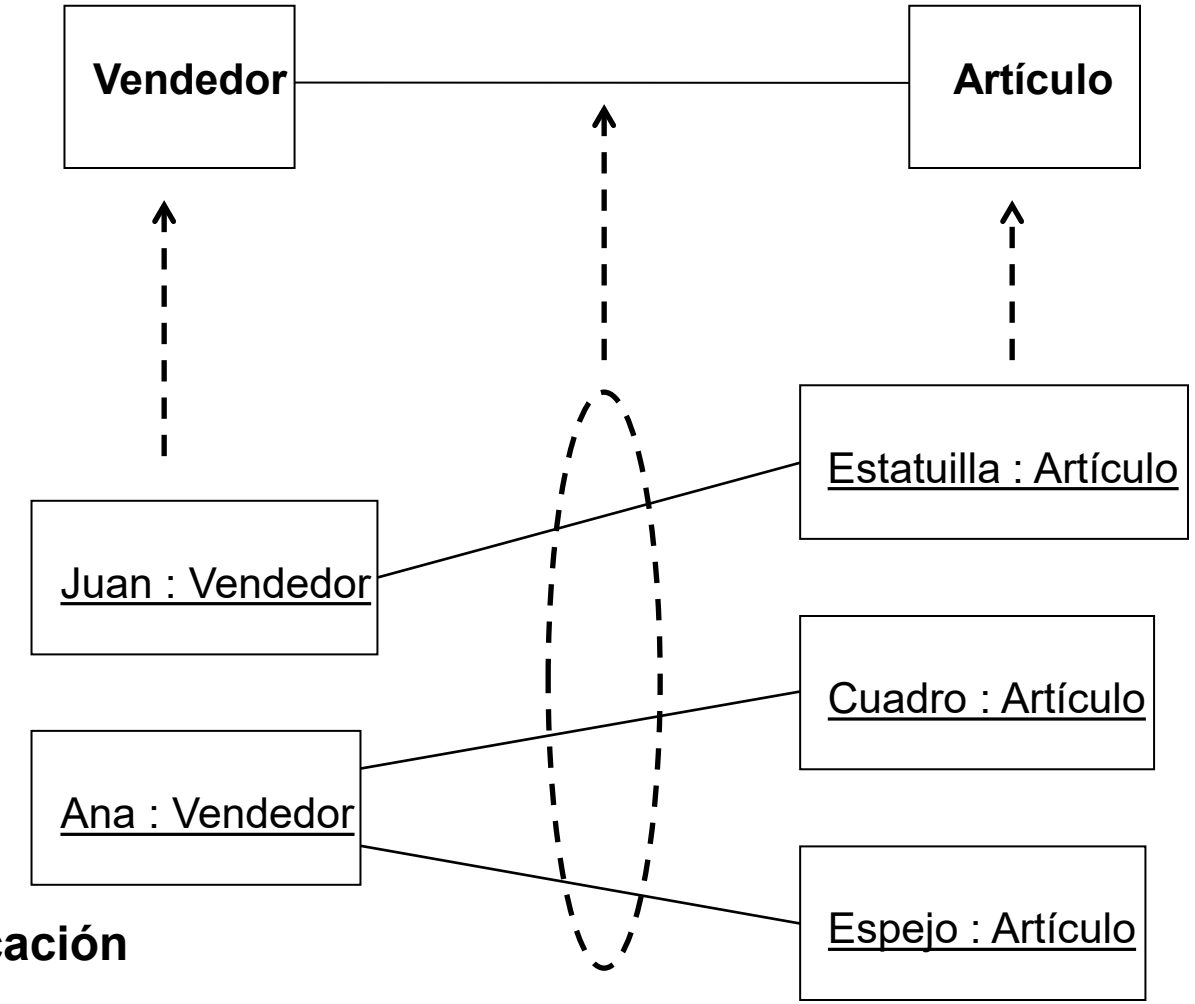
Asociación:
especificación de un conjunto de enlaces

representa la **estructura** y el **comportamiento** del sistema

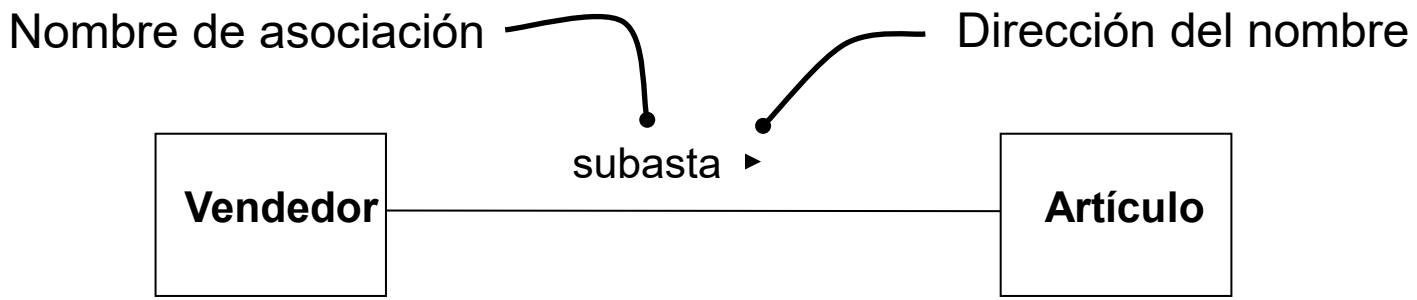
Enlace:
conexión entre objetos
determina una tupla de objetos
instancia de una asociación

estado de los objetos enlazados
estado del sistema

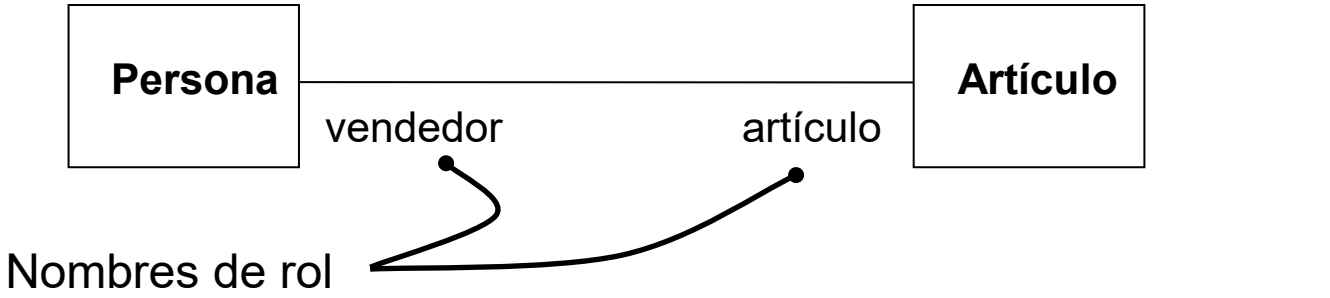
hecho + posibilidad de **comunicación**



Nombre de asociación y nombre de rol



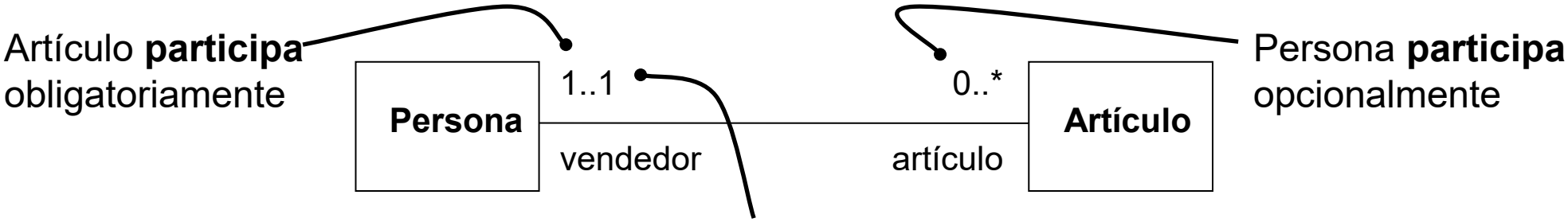
Los nombres de asociación se pueden repetir en un modelo, excepto para asociaciones entre las mismas dos clases



Los nombres de rol se pueden repetir en asociaciones distintas, y pueden ser iguales que los nombres de las clases asociadas

Multiplicidad de la asociación

- En una **asociación binaria**, la multiplicidad de un **extremo de asociación** especifica el número de instancias **destino** que pueden estar enlazadas con una única instancia **origen** a través de la asociación

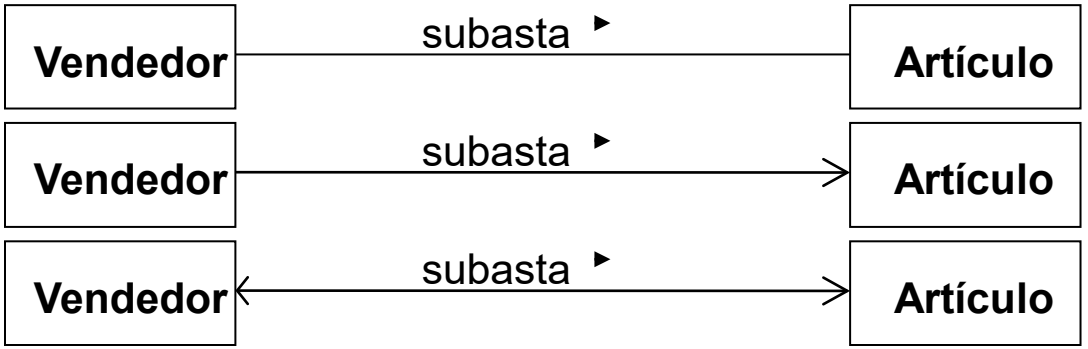


- **Valores típicos**:
  - 0..1 cero o uno
  - 1..1 uno y sólo uno (abreviado como "1")
  - 0..* desde cero hasta "muchos" (abreviado como "*")
  - 1..* desde uno hasta "muchos"
- Otros valores:
  - rangos enteros: (2..*), (0..3), etc.
  - lista de rangos separados por comas: (1, 3, 5..10, 20..*), (0, 2, 4, 8), etc.

Navegabilidad de la asociación

- La navegabilidad de una **asociación binaria** especifica la capacidad que tiene una instancia de la clase **origen** de acceder a las instancias de la clase **destino** por medio de las **instancias de la asociación** que las conectan
- **Acceder** = nombrar, designar o referenciar el objeto para...
  - leer o modificar el valor de un **atributo** del objeto (desaconsejable)

  ➔ invocar una **operación** del objeto **(enviarle un mensaje)**
  - usar el objeto como **argumento** o **valor de retorno** en un mensaje
  - modificar (asignar o borrar) el **enlace** con el objeto
- No confundir:
  - **dirección del nombre** de la asociación: asimetría lingüística
  - **navegabilidad** o direccionalidad de la asociación: asimetría comunicativa



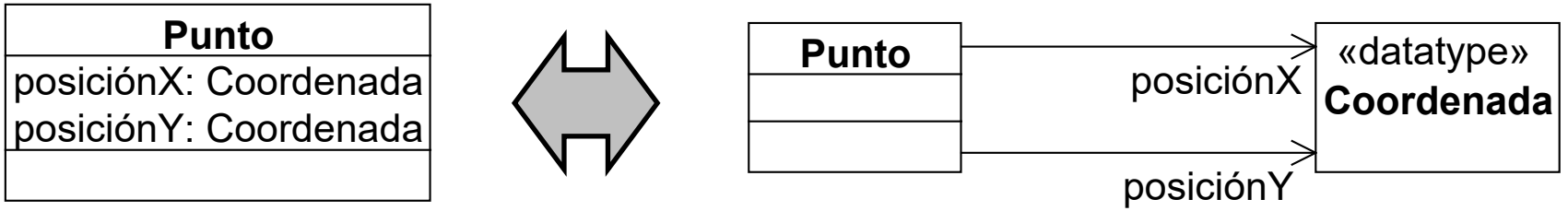
Navegabilidad no especificada

Asociación unidireccional

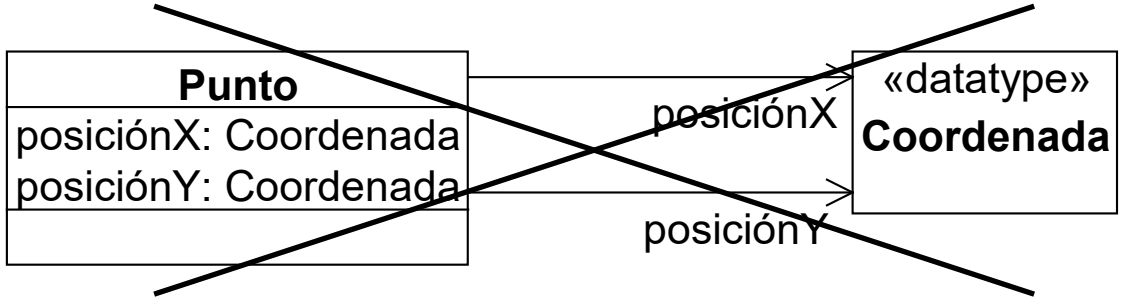
Asociación bidireccional

Asociación vs. Atributo

- Un **atributo** es equivalente a una **asociación unidireccional** (referencia hacia el tipo de dato del atributo)

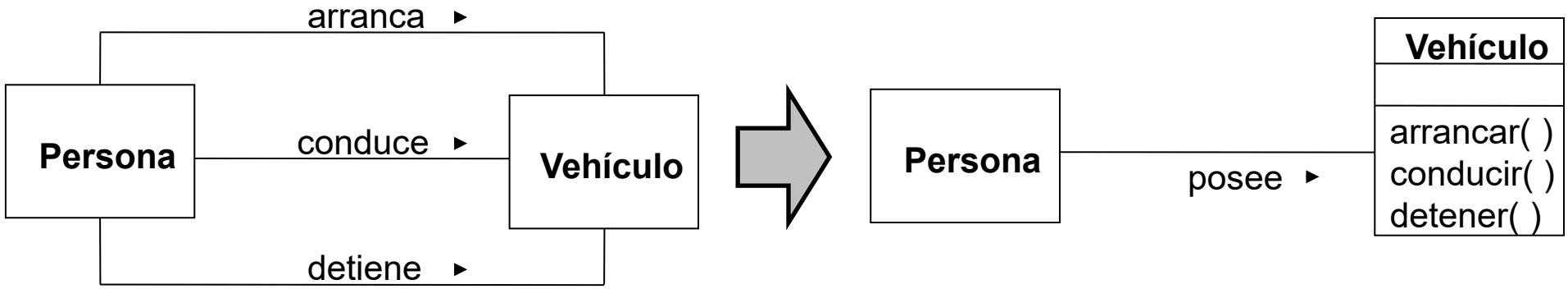


- Es conceptualmente **incorrecto** duplicar la representación



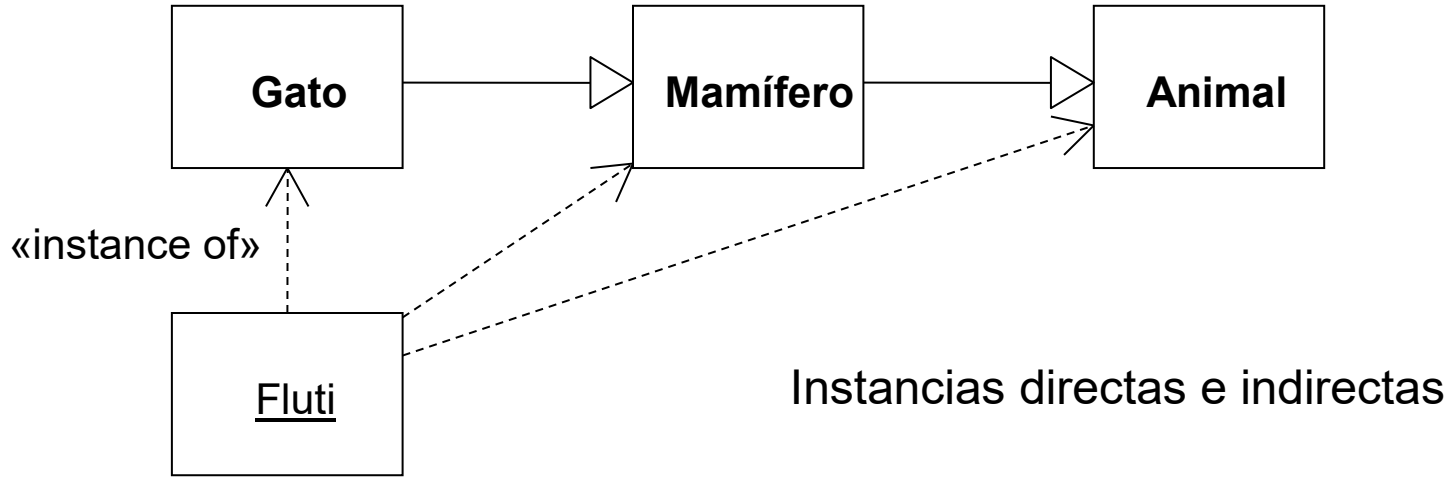
Asociación vs. Operación

- Toda asociación tiene un **doble significado**:
  - **aspecto estático**: estructura del sistema (estados posibles)
  - **aspecto dinámico**: comportamiento del sistema (interacciones posibles)
- El nombre de la asociación puede reflejar más un aspecto que el otro:
  - **nombres estáticos**: contiene, situado-en, trabaja-para, matrimonio, etc.
  - **nombres dinámicos**: subasta, publica, consulta, etc.
- Son preferibles los nombres estáticos, reservando los nombres dinámicos para **nombres de operaciones**, invocadas a través de la asociación mediante el envío de mensajes
- Una misma asociación permite la invocación de muchas operaciones



Generalización y clasificación

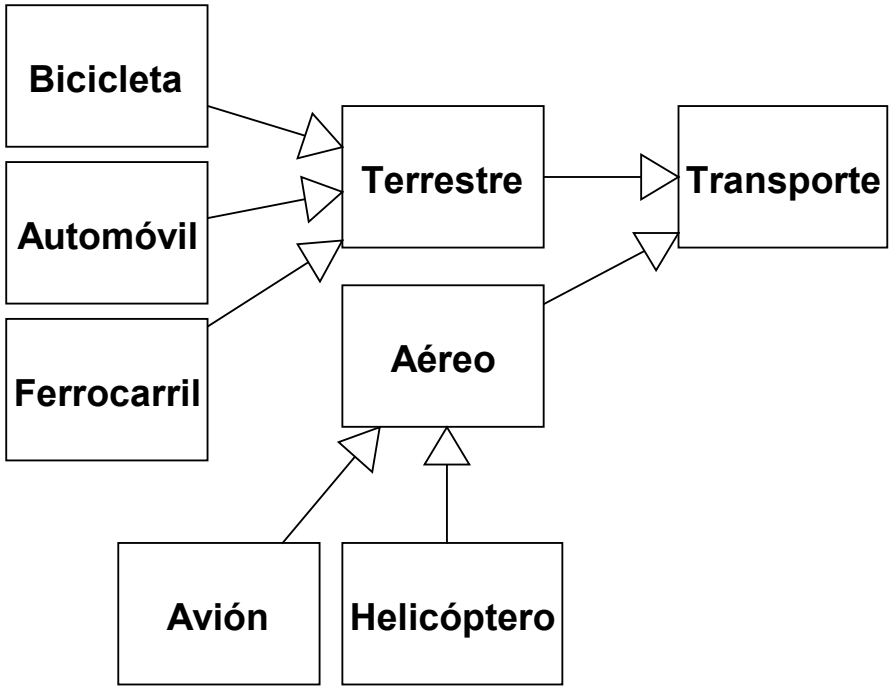
- Principio de sustitución (Barbara Liskov, 1987):
  - **Extensión**: todas los objetos de la subclase son también de la superclase.
  - **Intensión**: la definición de la superclase es aplicable a la subclase.
- Generalización: clase-clase.
  - Gato **es un tipo de** Mamífero, Mamífero **es un tipo de** Animal.
- Clasificación: objeto-clase.
  - Fluti **es un** Gato, Fluti **es un** Mamífero, Fluti **es un** Animal.



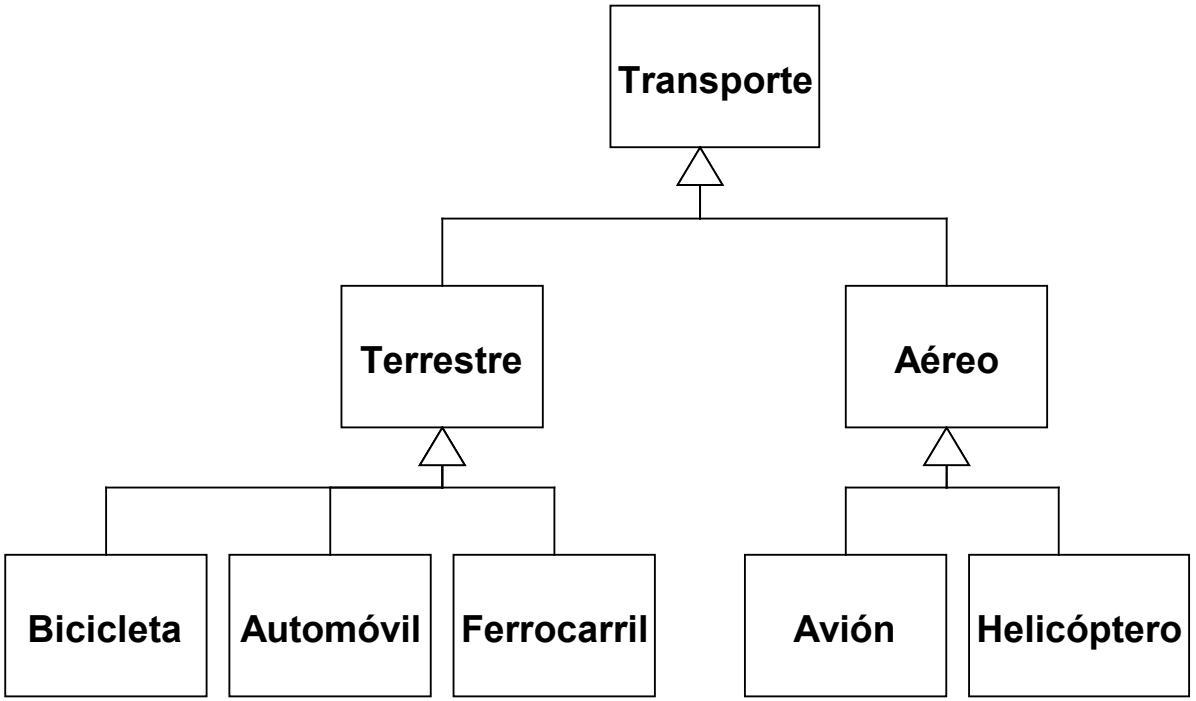
Generalización y especialización

- Dos puntos de vista complementarios:
  - Generalizar es identificar las **propiedades comunes** (atributos, asociaciones, operaciones) de varias clases y representarlas en una clase más general denominada **superclase**.
    - Elevar el nivel de abstracción, reducir la complejidad, organizar.
  - Especializar es capturar la **propiedades específicas** de un conjunto de objetos dentro de una clase dada, que aún no han sido distinguidas en ella, y representarlas en una nueva clase denominada **subclase**.
    - Reutilizar un concepto añadiendo propiedades variantes.
- Es una relación pura entre clases:
  - No tiene instancias, ni multiplicidad.
  - La subclase hereda **todas** las propiedades de la superclase.
  - Las propiedades heredadas de la superclase no se representan en la subclase (a menos que sean operaciones redefinidas).
  - Toda generalización induce una **dependencia** subclase → superclase.

Jerarquías de clases



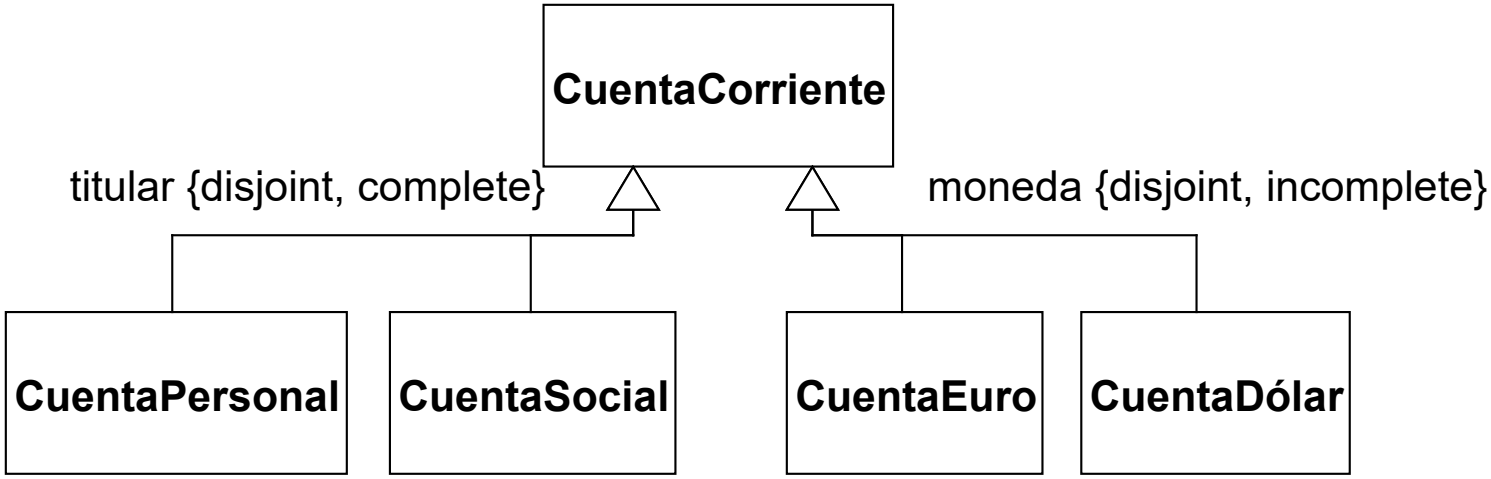
Representaciones alternativas:
- relaciones binarias
- estructura en árbol



Generalización:
- no-reflexiva
- transitiva
- asimétrica

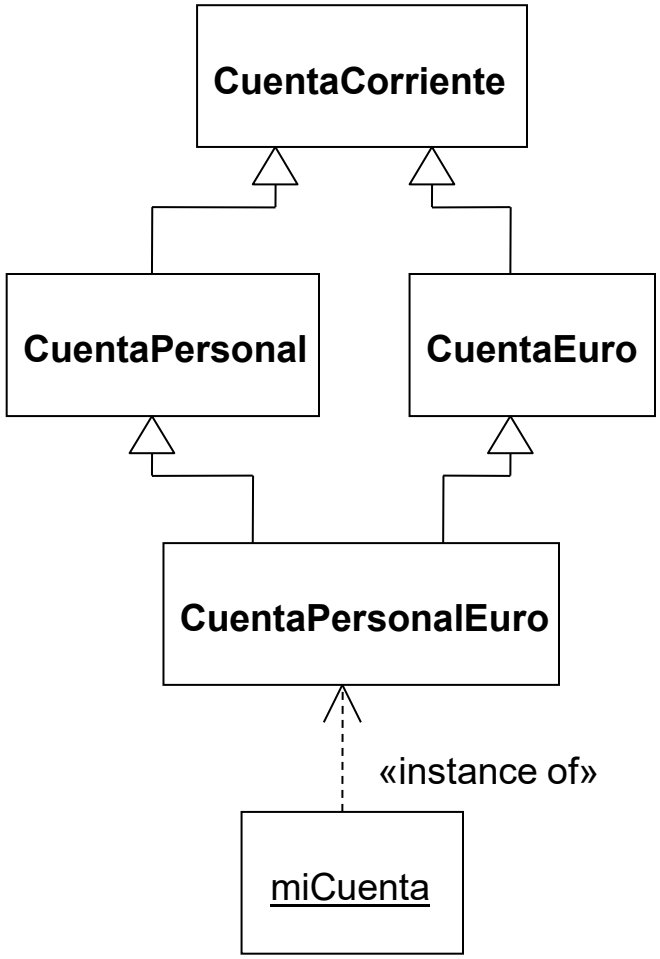
Dimensiones de especialización

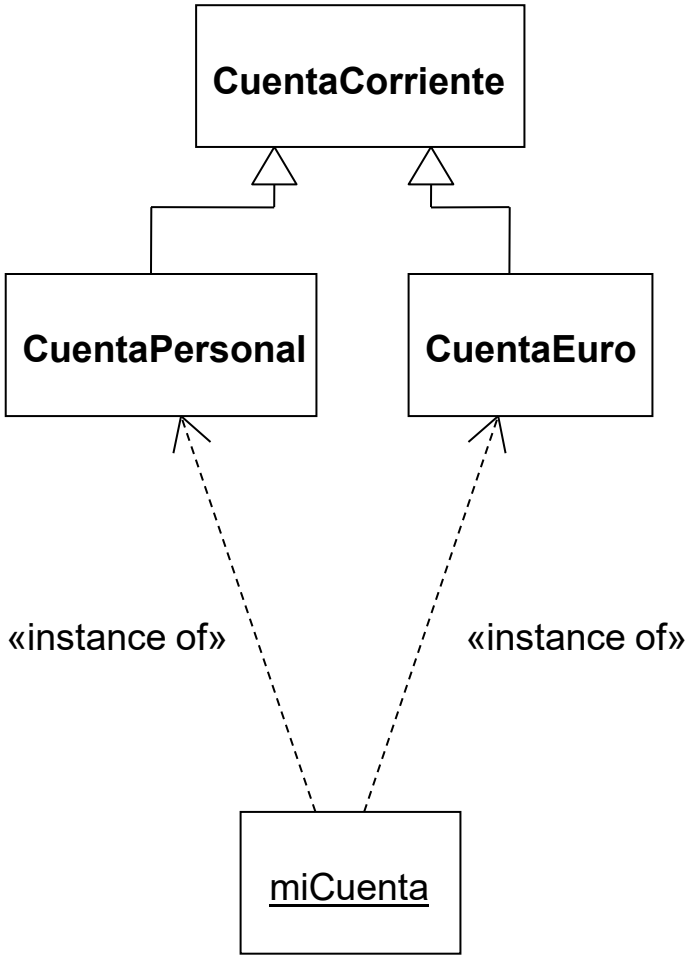
- Una superclase puede ser especializada en distintos **grupos de subclases** de acuerdo con criterios independientes: **discriminadores**.



- Restricciones:
  - **disjoint/overlapping**: las subclases no pueden tener instancias en común / o sí.
  - **complete/incomplete**: no hay otras subclases / o sí.
- Valor por defecto: disjoint, incomplete.
- **Partición propiamente dicha**: disjoint, complete.

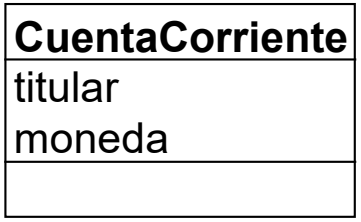
Generalización múltiple vs. Clasificación múltiple



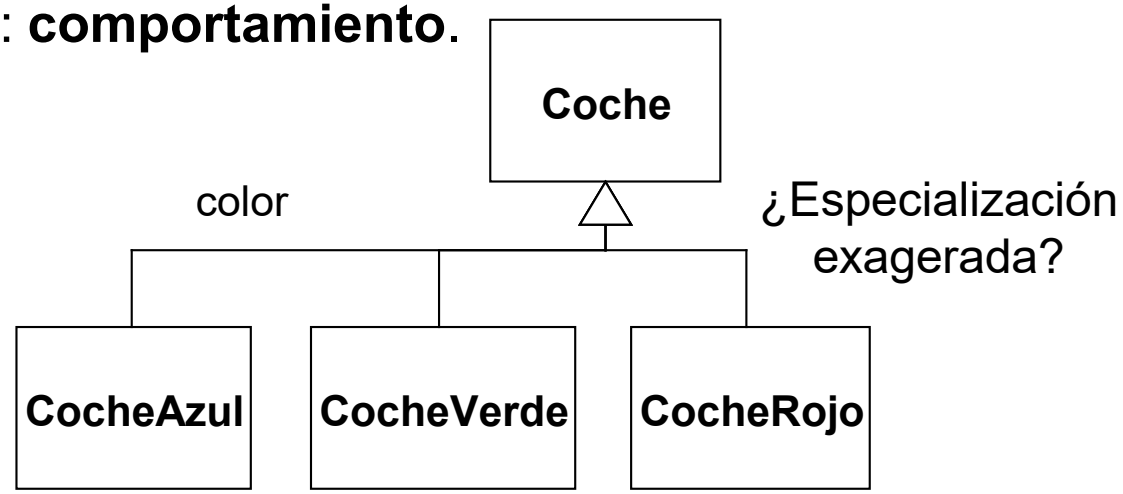


Subclase vs. Atributo

- ¿Cómo modelar las propiedades de los objetos? Regla general:
  - Propiedad cambiante o rango de valores muy grande: **atributo**.
  - Propiedad fija con valores enumerados: **especialización** (cada propiedad se traduce en un **criterio** de especialización, cada valor en una **subclase**).
    - También se puede modelar como un atributo con valor fijo.
- Problema de la **clasificación dinámica**: ¿puede un objeto cambiar de clase?
  - Permitiría modelar un cambio de propiedad como una **reclasificación** del objeto.
  - Interesante para aprovechar el **polimorfismo** (mediante un cambio de subclase).
  - No es habitual en los **lenguajes de programación**, aunque UML lo permite.
- Criterio final de especialización: **comportamiento**.



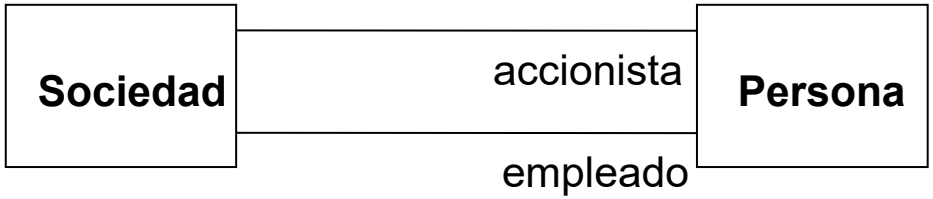
Alternativa a la doble especialización

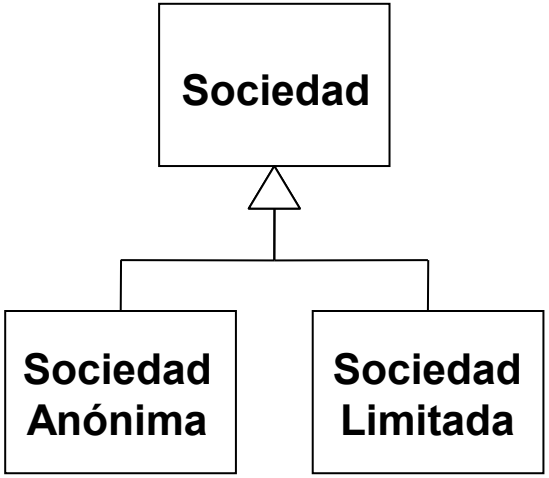


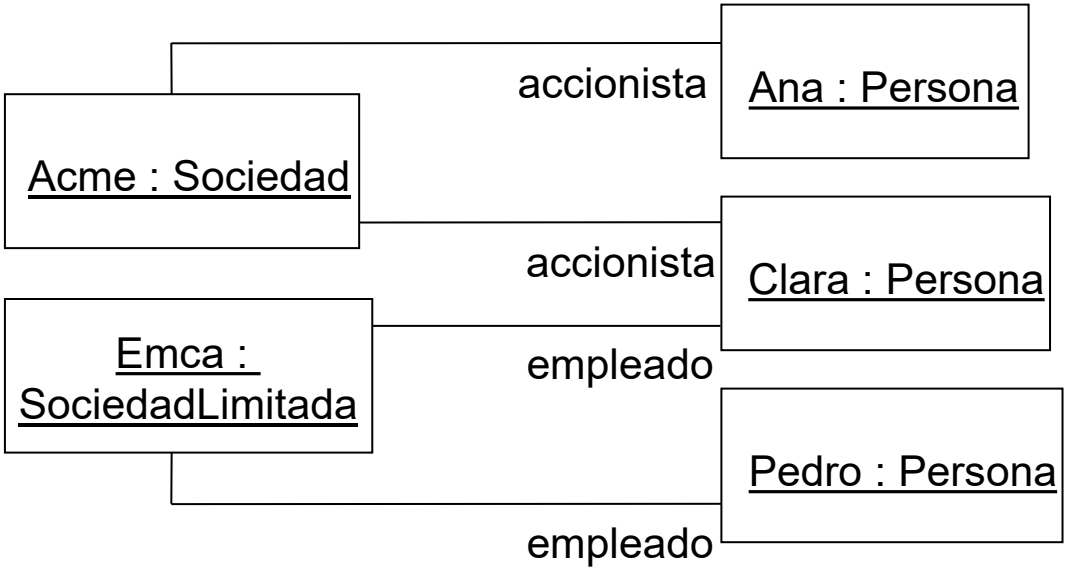
Diagramas de clases y de objetos

- Diagrama de clases
  - captura y **especifica** el vocabulario del sistema:
    - **elementos**: clases, atributos, operaciones...
    - **relaciones**: asociaciones, generalizaciones...
    - **estructura** del sistema, fundamento de su **comportamiento**
  - sugerencias para **mejorar la comunicación**:
    - nombres adecuados: clases, atributos, operaciones, asociaciones, roles
    - distribución espacial de los elementos
    - evitar cruces de líneas
    - distinto nivel de detalle según el propósito y nivel de abstracción
- Diagrama de objetos
  - **ilustra** la estructura del sistema mediante situaciones particulares
  - "fotografía" del sistema: objetos, valores de atributos; enlaces
  - las instancias deben **conformarse** a sus especificaciones
    - objetos, enlaces → clases, asociaciones
    - las especificaciones pueden estar representadas en **distintos diagramas**

Diagrama de clases vs. Diagrama de objetos

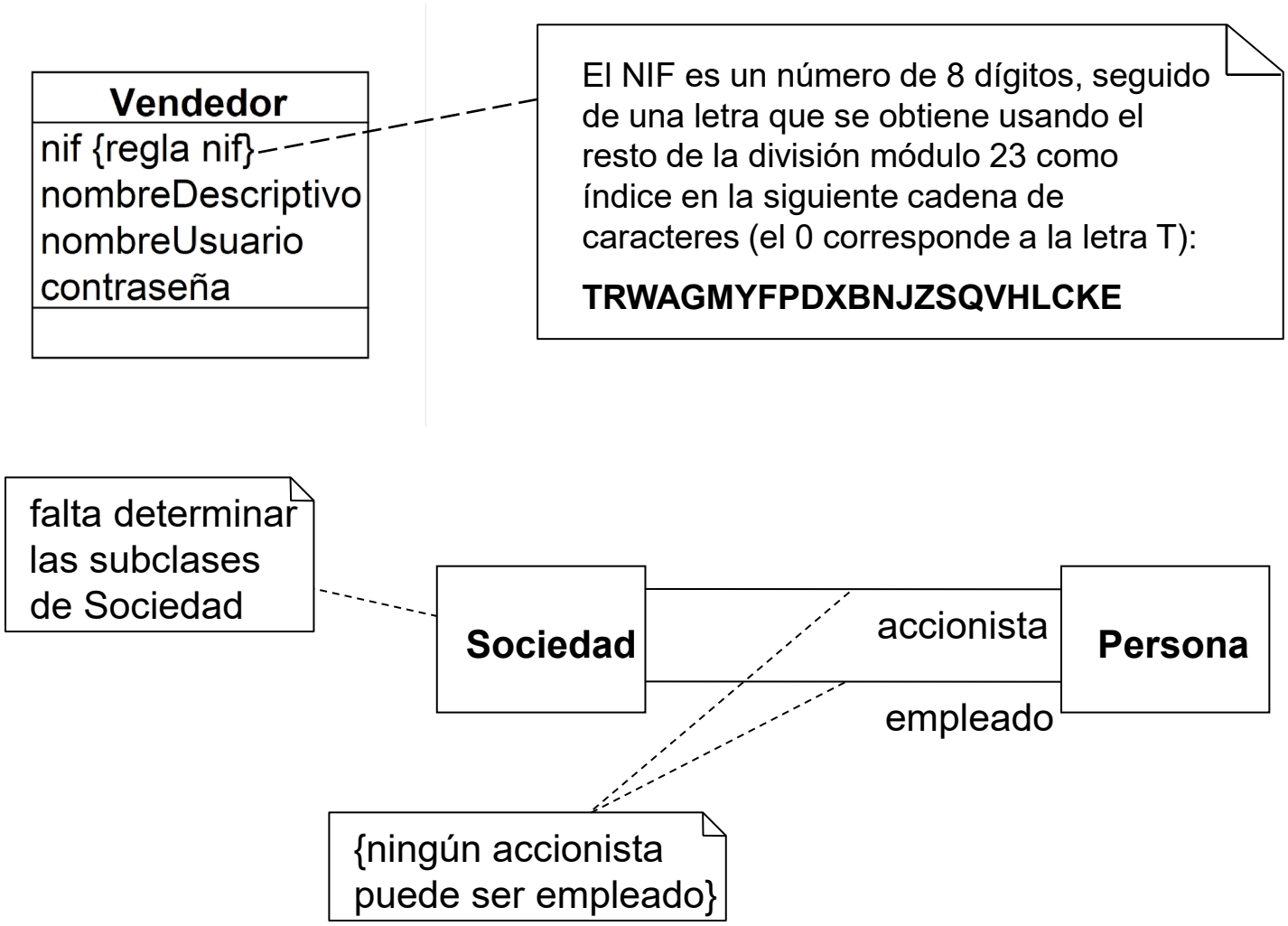




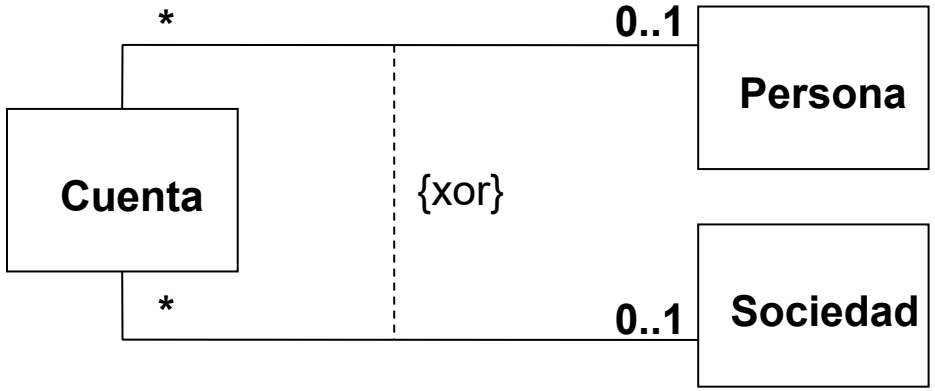


No es correcto decir que “un diagrama de objetos es una instancia de un diagrama de clases”

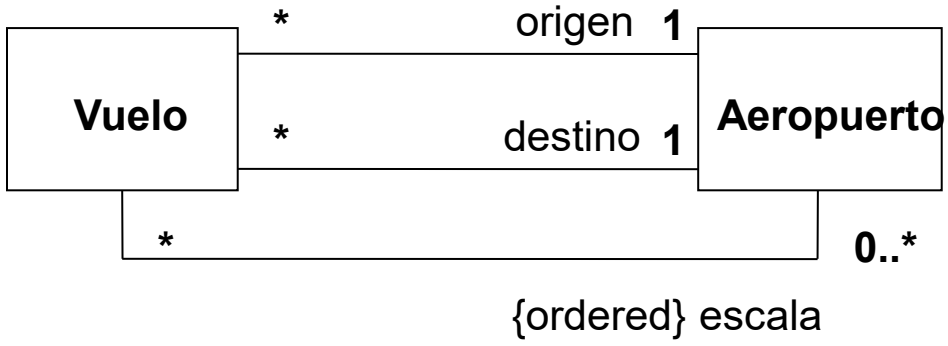
Restricciones y notas



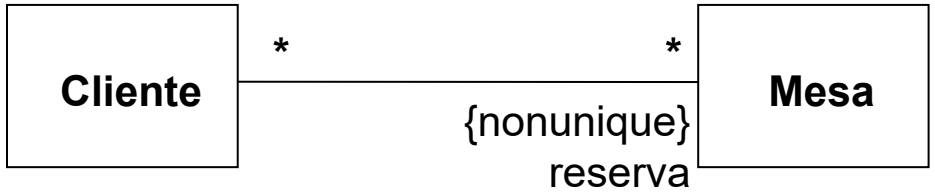
Restricciones en asociaciones



or exclusivo entre asociaciones



ordenación de los elementos



UML 1.x: una asociación no puede contener tuplas repetidas

UML 2.x: la restricción {nonunique} en un extremo permite la repetición

Legibilidad de los diagramas

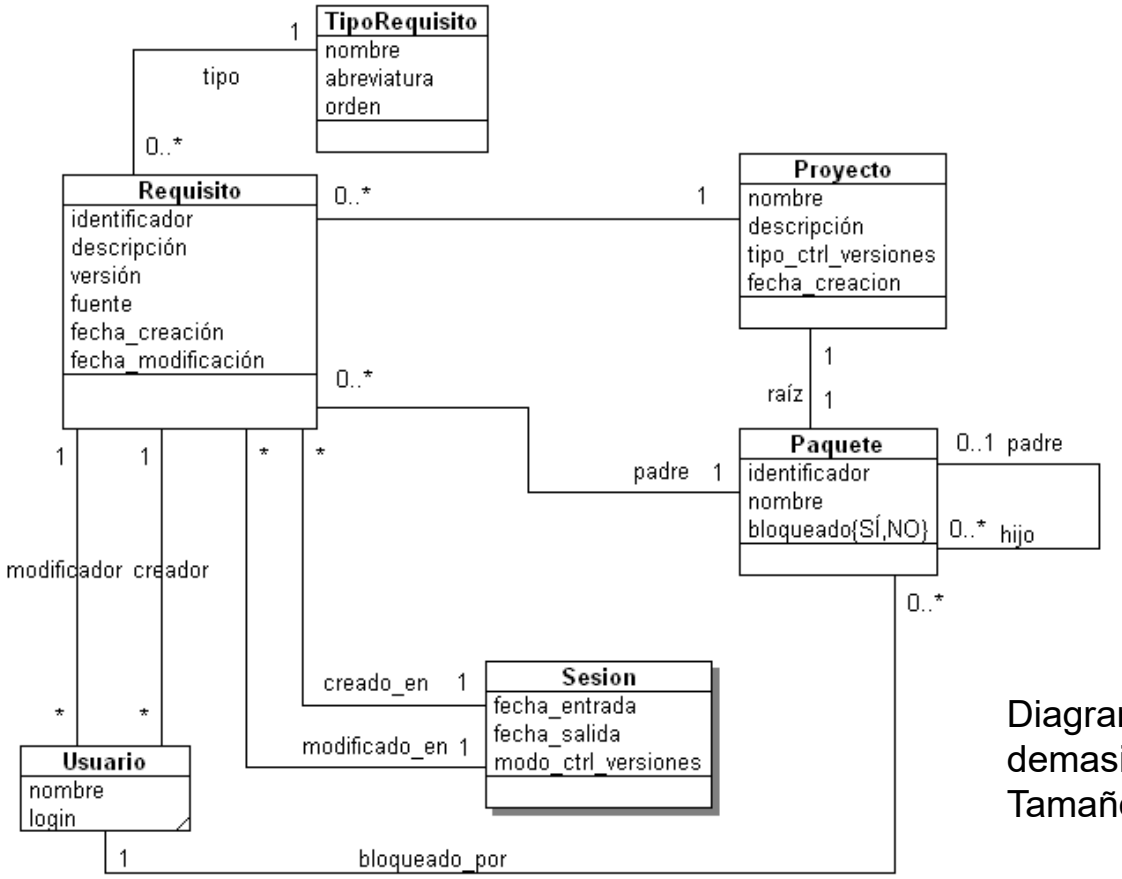
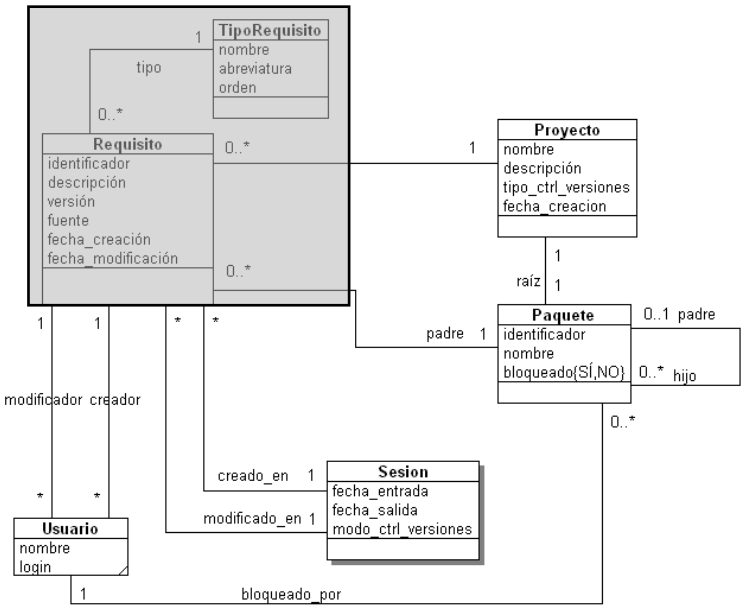
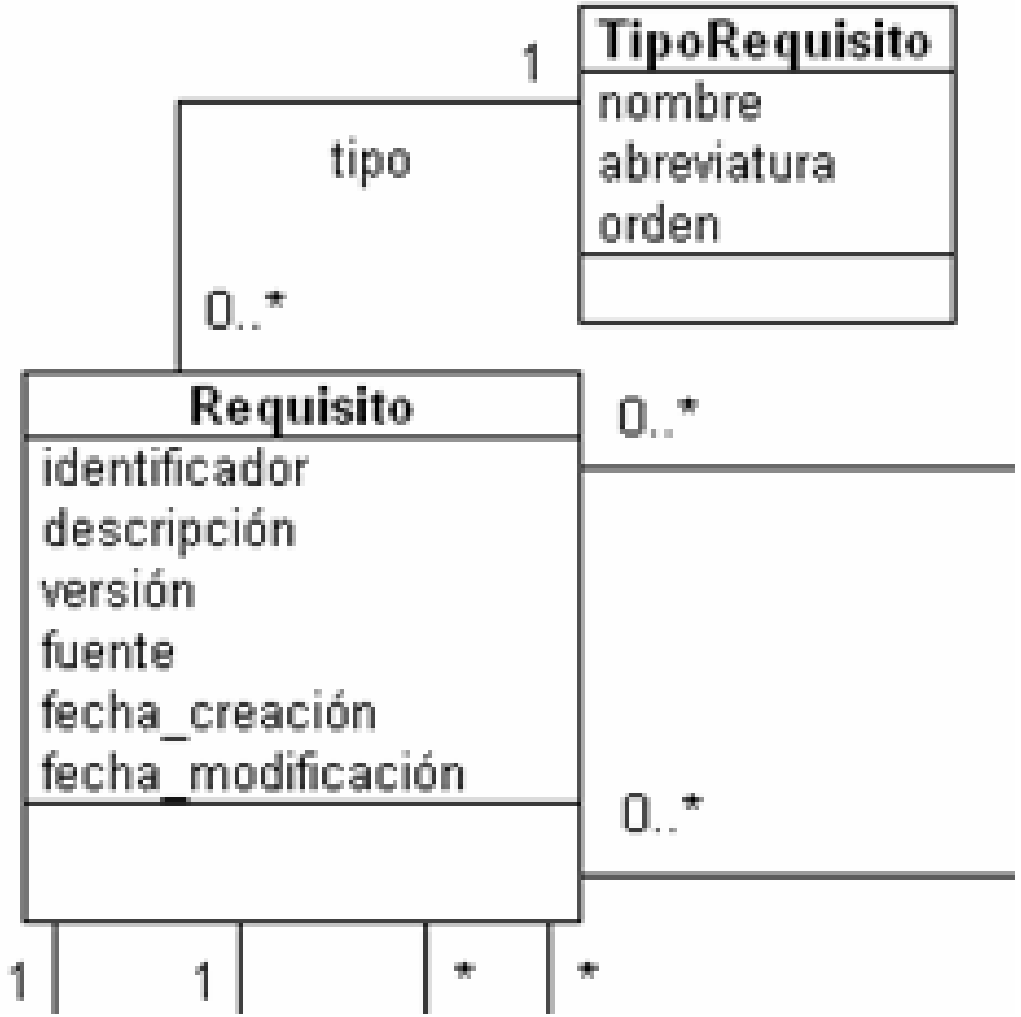


Diagrama ilegible: letra demasiado pequeña (Arial 9). Tamaño mínimo en torno a 14.

Legibilidad de los diagramas

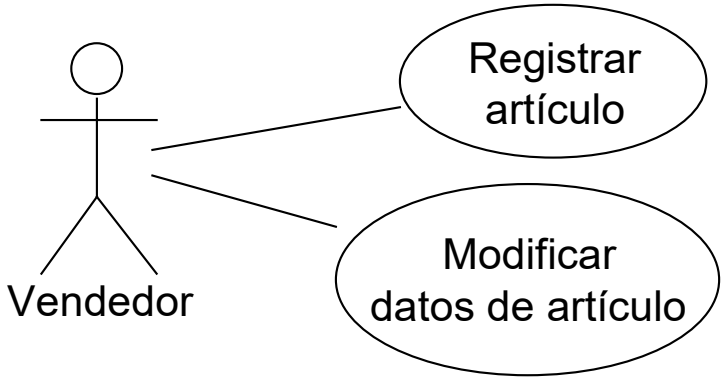


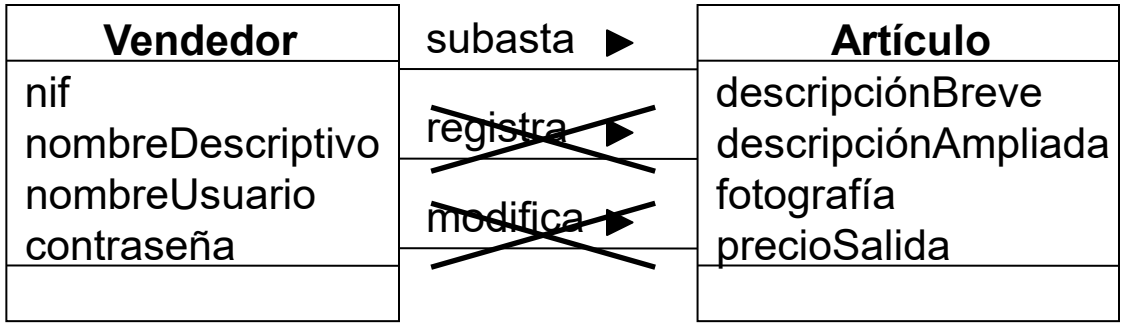


Usar “índice” en miniatura.

Asociaciones actor-sistema y clase-clase

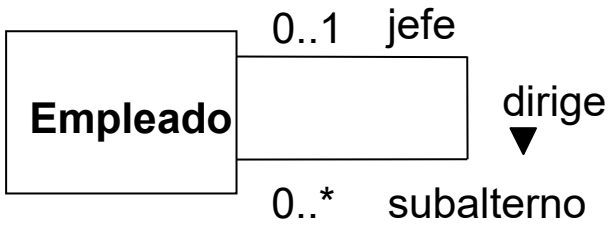
- Un mismo concepto puede ser modelado a la vez como actor y como clase:
  - **Actor**: representa entidades externas al sistema.
  - **Clase**: representa entidades modeladas dentro del sistema.
- No confundir asociaciones actor-sistema (casos de uso, relaciones con el exterior) con asociaciones clase-clase (relaciones internas):
  - "Para **subastar** algún **artículo** es necesario darse de alta como **vendedor**, introduciendo el NIF, un nombre descriptivo (largo), un nombre de usuario (breve) y una contraseña de acceso. Una vez que el vendedor está dado de alta, puede **registrar** artículos en la subasta o **modificar** alguno de sus datos: descripción breve, descripción ampliada, fotografía en formato JPEG, y precio de salida."

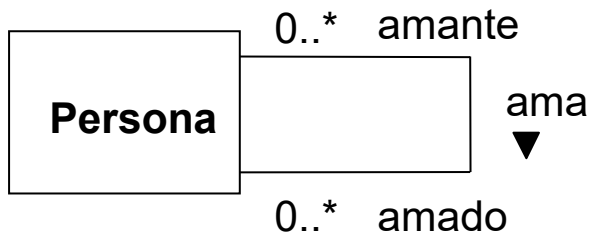




Asociaciones reflexivas

- Una asociación reflexiva (o **recursiva**) es aquella en la que los dos extremos de la asociación están unidos a la misma clase.
- Los enlaces pueden conectar **dos instancias diferentes** de la misma clase, o incluso **una instancia consigo misma**.
- En una asociación reflexiva los **nombres de rol** son obligatorios, para poder distinguir los dos extremos de la asociación.
- Una asociación reflexiva **no es simétrica**: los extremos son distinguibles, aunque la asociación quiera significar equivalencia: es-amigo-de, es-igual-a...



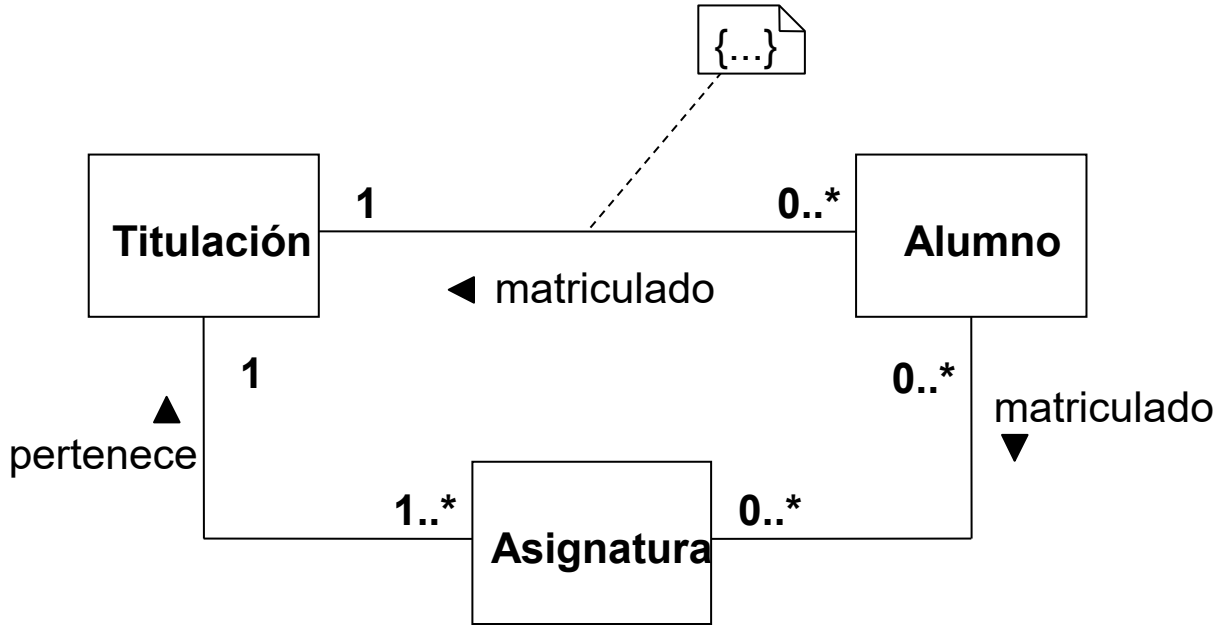


dirige(Ana, Juan) ≠ dirige(Juan, Ana)

ama(Pedro, Clara) ≠ ama(Clara, Pedro)

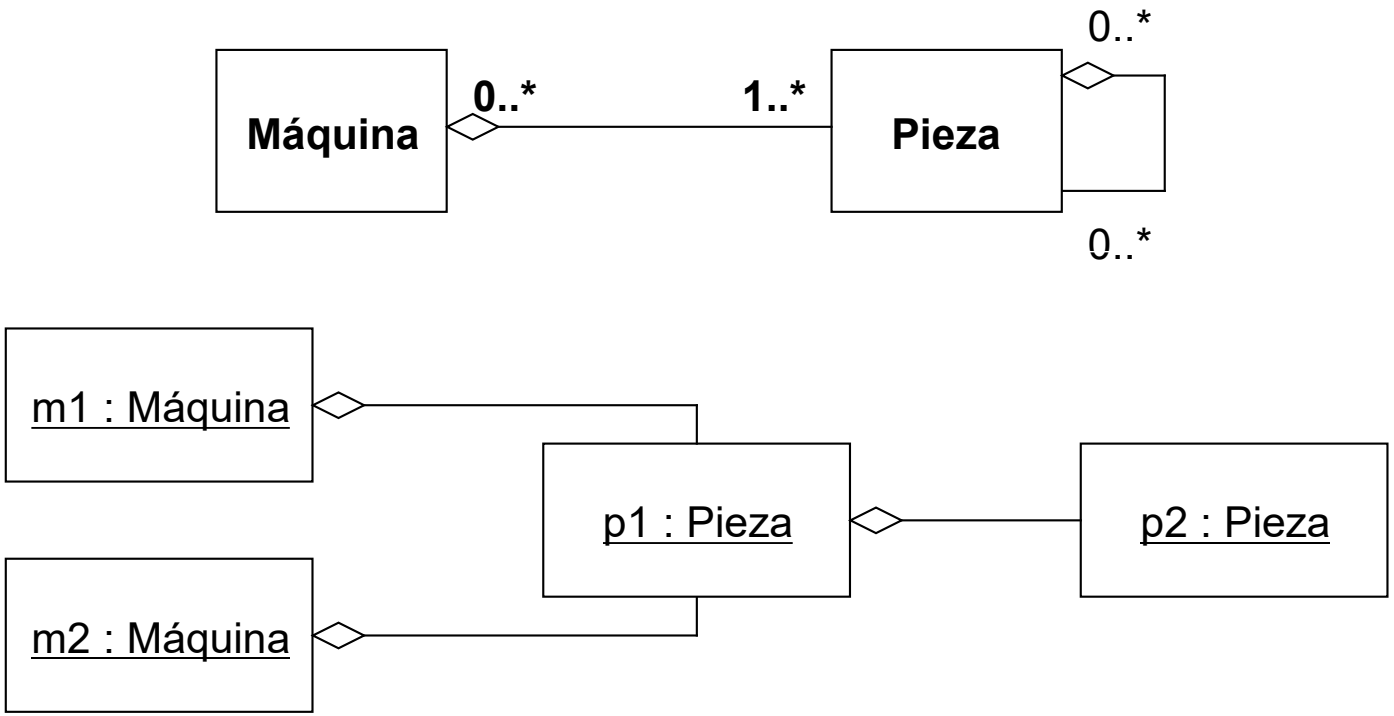
Ciclos de asociaciones y asociaciones derivadas

- ¿Puede un alumno matricularse en asignaturas que no son de su titulación?
- Posibles interpretaciones alternativas del diagrama: la titulación matriculada…
  - se obtiene a partir de las asignaturas: **asociación derivada**.
  - actúa como **restricción** para elegir asignatura matriculada.
  - actúa como **sugerencia** para elegir asignatura matriculada.
  - titulación matriculada y asignatura matriculada son **independientes**.



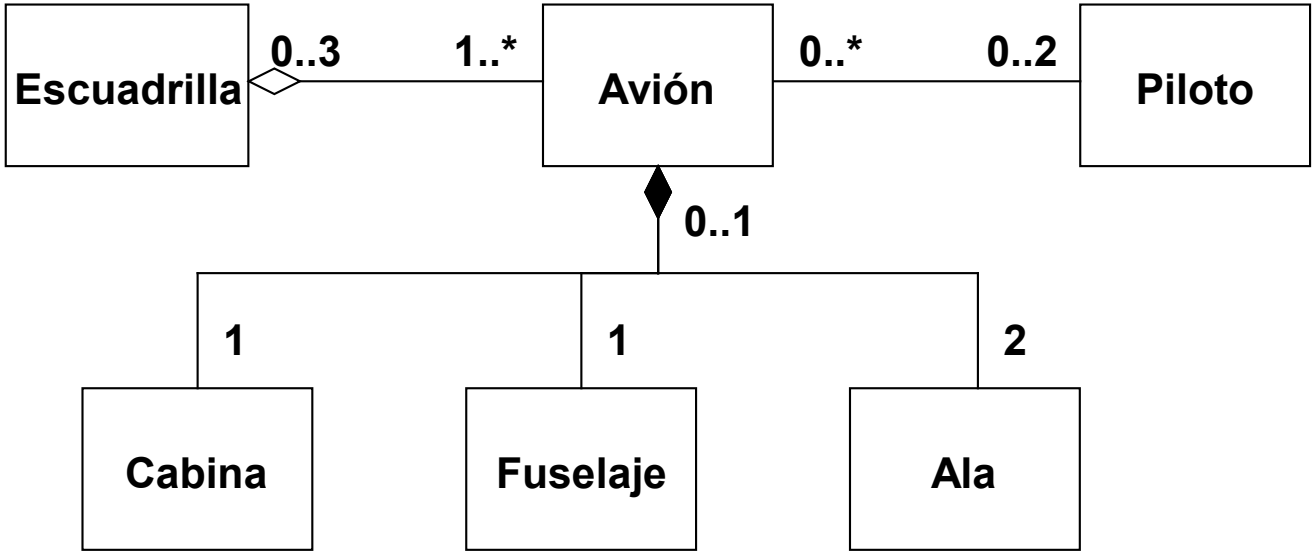
Agregación

- Es un tipo especial de asociación que representa una **relación todo-parte**, transitiva y asimétrica
  - no impone ninguna restricción especial sobre la **multiplicidad**
  - puede ser **reflexiva** para las clases, pero no para las instancias



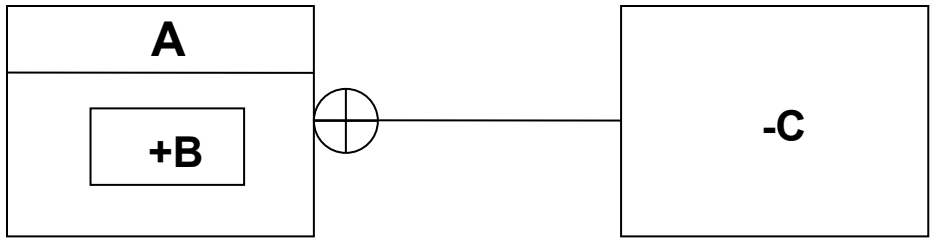
Composición

- Es un tipo especial de agregación **no compartida**
  - la **multiplicidad** sólo puede ser 0..1 ó 1..1
  - el todo es responsable de la **existencia** y **almacenamiento** de las partes
  - **propagación** de las operaciones de copiado y borrado
- Composición **no significa** encapsulamiento ni acceso restringido



Anidamiento – Clase interna

- Una clase puede anidarse dentro de otra, como los **paquetes**, con el fin de limitar la **visibilidad** desde el exterior
- La relación de anidamiento **no es una asociación**
- No tiene nada que ver con la agregación o la composición
  - la **composición** abstrae enlaces todo-parte entre objetos
  - el **anidamiento** es una pura relación de inclusión entre clases



Diseño e implementación de asociaciones binarias

- Los **lenguajes OO** no proporcionan una construcción adecuada
  - combinación de **atributos** y **operaciones** para gestionar los enlaces
  - colecciones para manejar la **multiplicidad** (comprobación de tipo...)
  - referencias cruzadas sincronizadas para manejar la **bidireccionalidad**

Diagrama de análisis

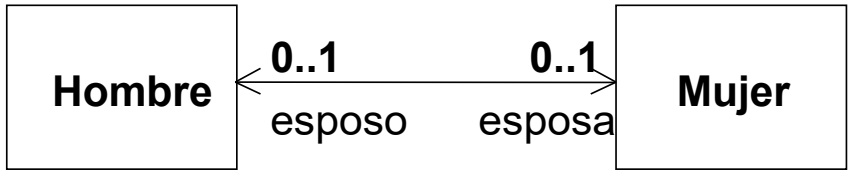
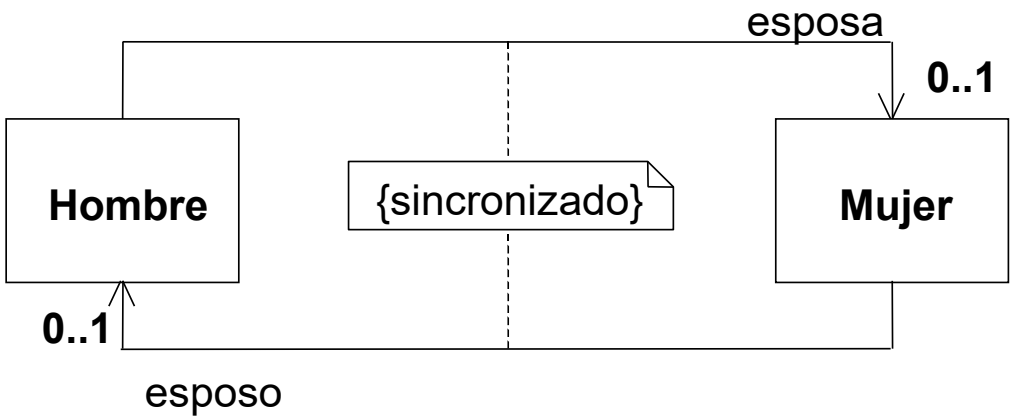


Diagrama de diseño



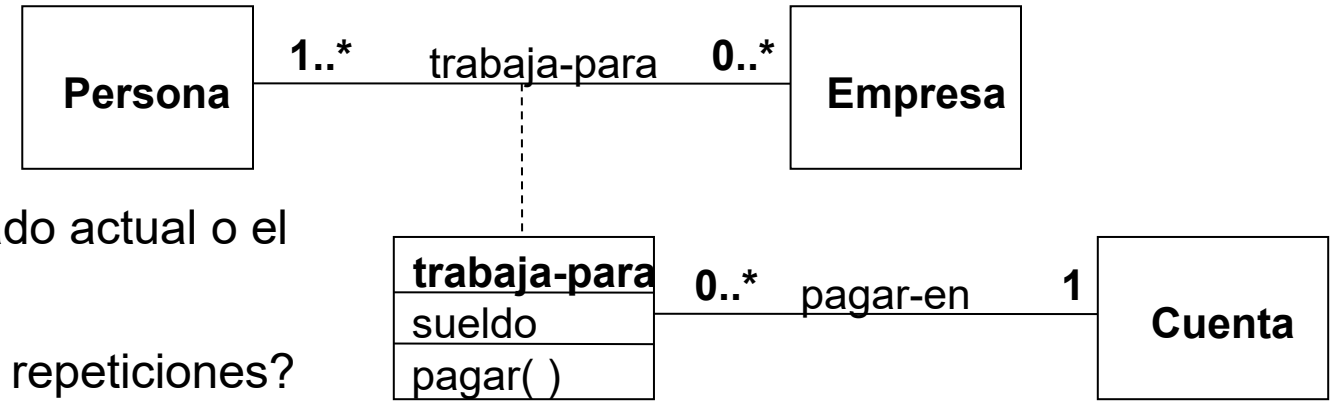
Código demasiado simple:

```
public class Hombre {
      private Mujer esposa;
      …
}

public class Mujer {
      private Hombre esposo;
      …
}
```

Clase-asociación

- Tiene todas las propiedades de una clase y de una asociación:
  - **atributos**, **operaciones** y **asociaciones** con otras clases.
  - conexión **entre clases** que especifica **enlaces** entre ellas.
  - multiplicidad, navegabilidad, agregación...
- Es un único elemento, por tanto tiene un **nombre único**.
- Como cualquier otra asociación, en UML 1.x no podía contener tuplas repetidas, aunque los valores de los atributos fueran distintos: sustituir **clase-asociación** por **clase intermedia**.

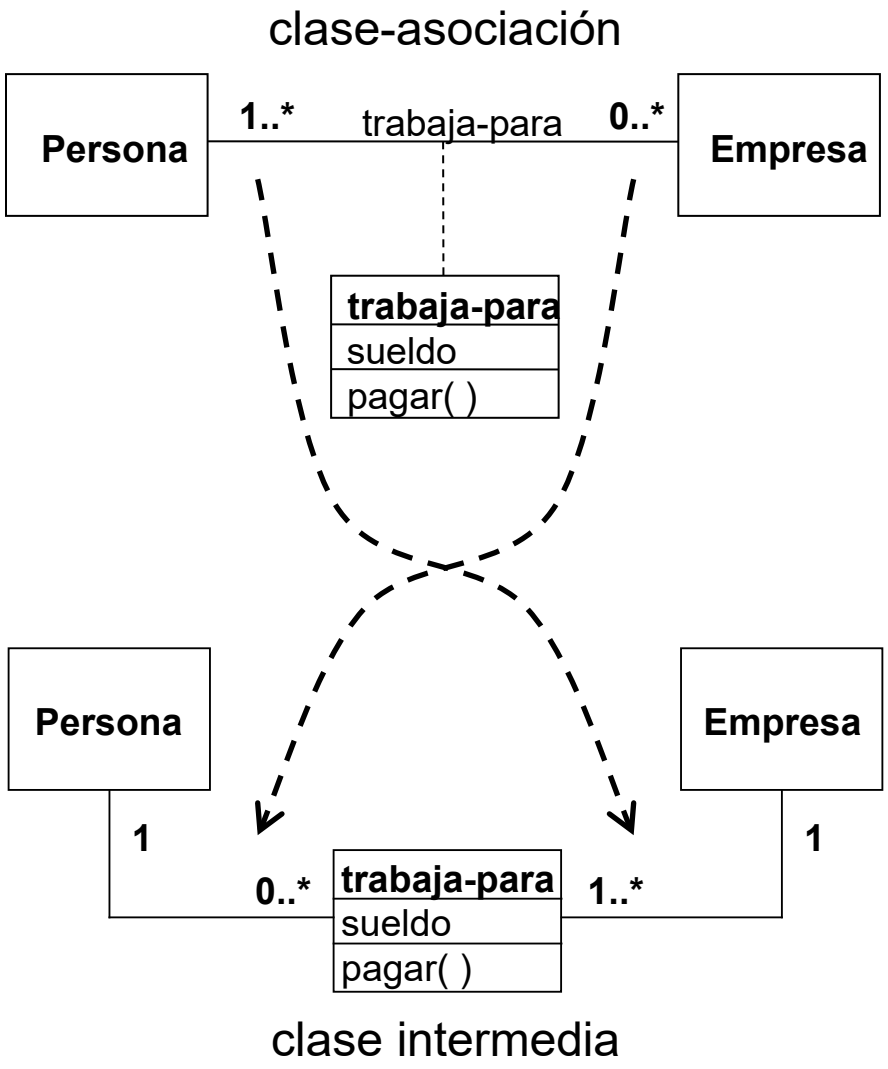


¿Representa el estado actual o el registro histórico?

¿Queremos permitir repeticiones?

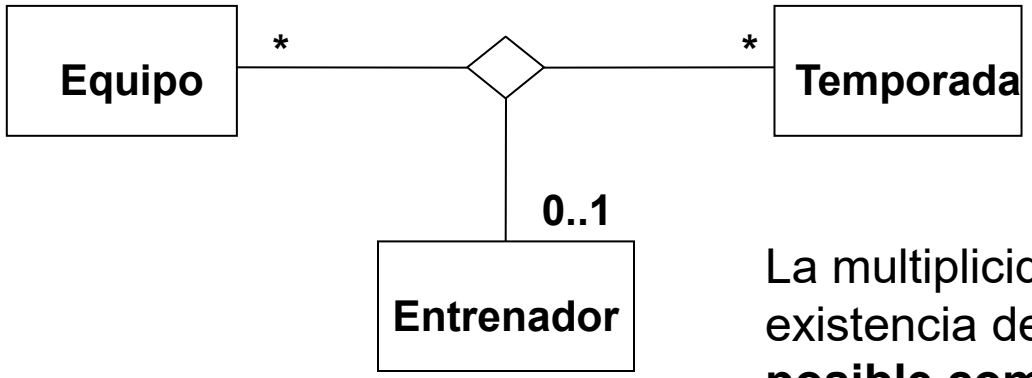
Transformación de clase-asociación en clase intermedia

- Es una forma de **implementar** la clase-asociación, que consiste en sustituir la clase-asociación por una **clase simple**, cuyas instancias **representan enlaces**.
- Las **multiplicidades originales** se **cruzan**, y aparecen otras nuevas.
- Permite automáticamente la existencia de **tuplas repetidas**; hay que añadir una **restricción adicional** si se desea evitarlas.



Asociación n-aria

- Asociación entre **N clases**: los enlaces conectan **N instancias**
  - **no permite**: dirección del nombre, agregación
  - **sí permite**: navegabilidad (aunque significado problemático), clase-asociación
- Multiplicidad engañosa:
  - número permitido de instancias para cualquier **posible combinación** de instancias de las otras n-1 clases
  - la **multiplicidad mínima** suele ser 0
  - efecto **"rebote del uno"**: la multiplicidad mínima 1 (o superior) en un extremo implica que debe existir un enlace (o más) para cualquier posible combinación de instancias de los otros extremos



Un equipo enlazado con una temporada siempre tiene un entrenador asignado: no hay **"enlaces cojos"**.

La multiplicidad mínima 1 implicaría la existencia de un enlace (o más) para **toda posible combinación** de equipo y temporada.

Transformación de asociación n-aria en clase intermedia

- Sustituir la asociación n-aria por una **clase simple**, cuyas instancias **representan enlaces**.
- Las **multiplicidades originales** se **pierden**, y aparecen otras nuevas.
- Permite la existencia de **tuplas repetidas**, cuando es necesario.
- Es una forma de **implementar** la asociación n-aria, pero hay que añadir **restricciones adicionales** para no permitir tuplas repetidas y para expresar las multiplicidades originales perdidas.