FUNDAMENTOS DE LA VISTA ESTÁTICA

INGENIERÍA DE SOFTWARE I

2º DE GRADO EN INGENIERÍA INFORMÁTICA
CURSO 2021/2022

Dr. Francisco José García Peñalvo / fgarcia@usal.es
Alicia García Holgado / aliciagh@usal.es
Andrea Vázquez Ingelmo / andreavazquez@usal.es

Departamento de Informática y Automática
Universidad de Salamanca

ÍNDICE

- Características
- Clases y objetos
  - Atributos
  - Operaciones
- Interfaces
- Relaciones

CARACTERÍSTICAS

Modela conceptos del dominio de la aplicación sus propiedades internas y sus relaciones

Se denomina vista estática porque no modela comportamiento del sistema dependiente del tiempo

Componentes principales

- **Clases**: describen conceptos del dominio de la aplicación o de la solución
- **Relaciones**
    - Asociación
    - Generalización
    - Dependencia: realización y uso

Diagramas utilizados

- **Diagrama de clases**: colección de elementos de modelado estáticos como clases, tipos, sus contenidos y relaciones

CLASES Y OBJETOS

Una **clase** es un clasificador que describe un conjunto de objetos que comparten la misma especificación de características, restricciones y semántica

- Una clase describe las propiedades y comportamiento de un grupo de objetos

Un **objeto** es una instancia de una clase

- Un objeto es una entidad discreta con identidad, estado y comportamiento invocable
- Los objetos representan entidades software del mundo real

Diagramas de clases: describen la vista estática de un sistema en forma de clases y relaciones entre ellas

Diagramas de objetos: muestra las instancias de las clases y los enlaces específicos entre esas instancias en un momento determinado

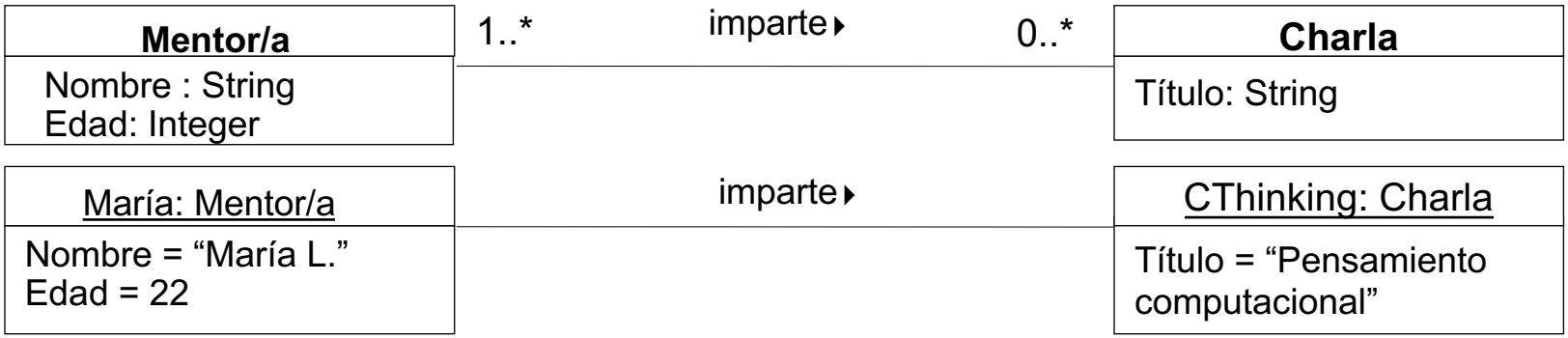
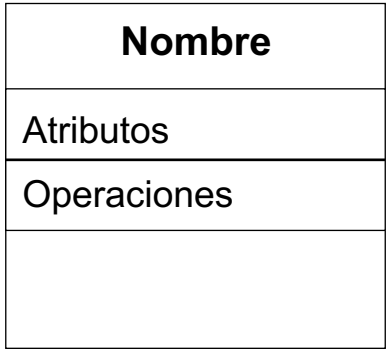


Diagrama de clases y diagrama de objetos

CLASES Y OBJETOS

Características de las clases

- Tienen un nombre único dentro de su contenedor, que es generalmente un paquete, aunque puede ser también otra clase
- Tienen una visibilidad con respecto a su contenedor, la cual especifica cómo puede ser utilizada por otras clases externas al contenedor
- Se representan por rectángulos compartimentados
  - **Compartimento del nombre:** contiene al menos el nombre de la clase (inicial en mayúscula) en negrita y centrado. Puede contener también estereotipos (ej. <<enumeration>>) y propiedades (ej. {persistent})
  - **Compartimento de los atributos:** contiene el nombre de los atributos (inicial en minúscula) alineado a la izquierda y opcionalmente información adicional
  - **Compartimento de las operaciones:** contiene el nombre de las operaciones (inicial en minúscula) alineado a la izquierda y opcionalmente información adicional
  - **Compartimentos adicionales:** para mostrar propiedades definidas por el usuario



CLASES Y OBJETOS

Compartimento de atributos (I)

- Sintaxis de descripción de un atributo

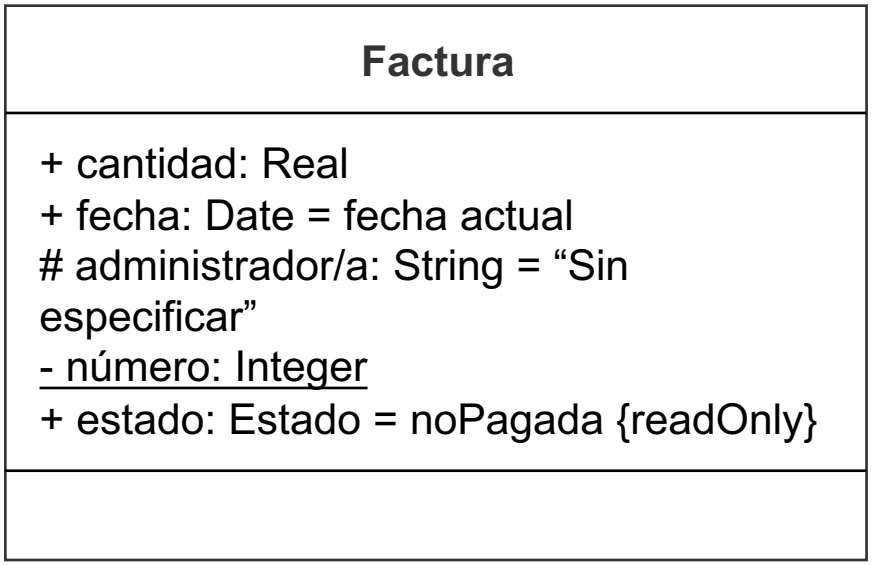
visibilidad / nombre:tipo [multiplicidad] = valorPorDefecto {cadena de propiedades}

- **Visibilidad**: expresa si los atributos son visibles a otros objetos
  - ***+ Visibilidad Pública***
  - ***# Visibilidad Protegida***
  - ***- Visibilidad Privada***
  - ***~ Visibilidad de Paquete***
  - No hay visibilidad por defecto
  - Se pueden definir otros tipos de visibilidad usando perfiles
- **/** : indica que el atributo es derivado
- **Nombre**: es una cadena que sirve para identificar al atributo
- **Tipo**: Indica el tipo o dominio del atributo
- **Multiplicidad**: número de instancias del atributo (ej. [0..1])
- **ValorPorDefecto**: si no se da este valor se omite el signo igual
- **Cadena de propiedades**: lista separada por comas de las propiedades de un atributo (readOnly, ordered, sequence…)

CLASES Y OBJETOS

Compartimento de atributos (II)

- Un atributo con alcance de clase se expresa mediante una cadena subrayada



CLASES Y OBJETOS

Compartimento de operaciones (I)

- Sintaxis de descripción de una operación

visibilidad nombre (listaParametros) : tipoRetorno {cadena de propiedades}

- **Visibilidad**: igual que para los atributos
- **Signatura de la operación**: nombre, listaParametros y tipoRetorno
- **listaParametros**: lista separada por comas de los parámetros formales. Sintaxis:

direccion nombre : tipo [multiplicidad] = valorPorDefecto {cadena de propiedades}

  - **Direccion:** indica si el parámetro se envía dentro o fuera de la operación (in, out, inout)

- **Cadena de propiedades**: lista separada por comas de las propiedades o restricciones de una operación (isQuery, isPolymorphic…)

CLASES Y OBJETOS

Compartimento de operaciones (II)

- Una operación con alcance de clase se expresa mediante una cadena subrayada
- Si la operación es abstracta se muestra en cursiva

Figura
tamaño: Tamaño pos: Posicion
+ *dibujar ( )* + escalarFigura (porcentaje: Integer = 25) + obtenerPosicion ( ): Posicion + obtenerContador: Integer

CLASES Y OBJETOS

Clases entidad, control e interfaz

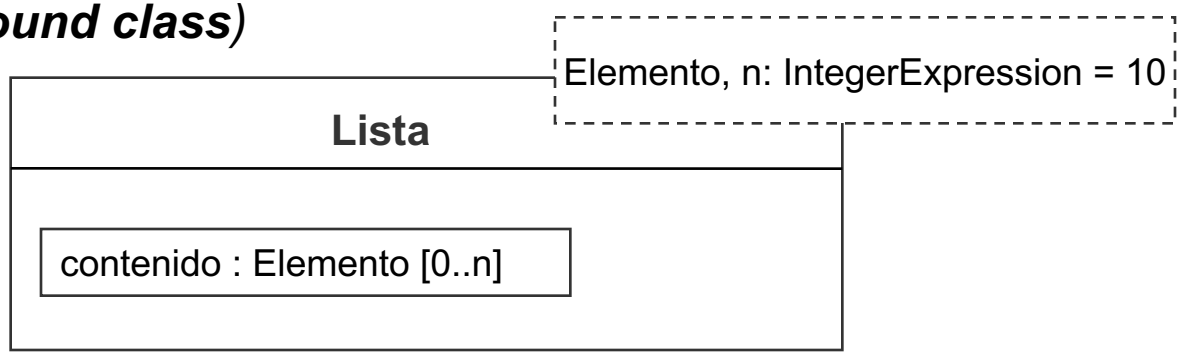
- Las clases **entidad** (***entity***) representan objetos de negocio normalmente persistentes que se almacenan en el sistema
- Las clases de **control** (***control***) se ocupan del procesamiento de la información
- Las clases **interfaz** (***boundary***) se encargan de presentar y comunicar la información al usuario o a otros sistemas

Estereotipos utilizados para representar las clases entidad, control e interfaz

CLASES Y OBJETOS

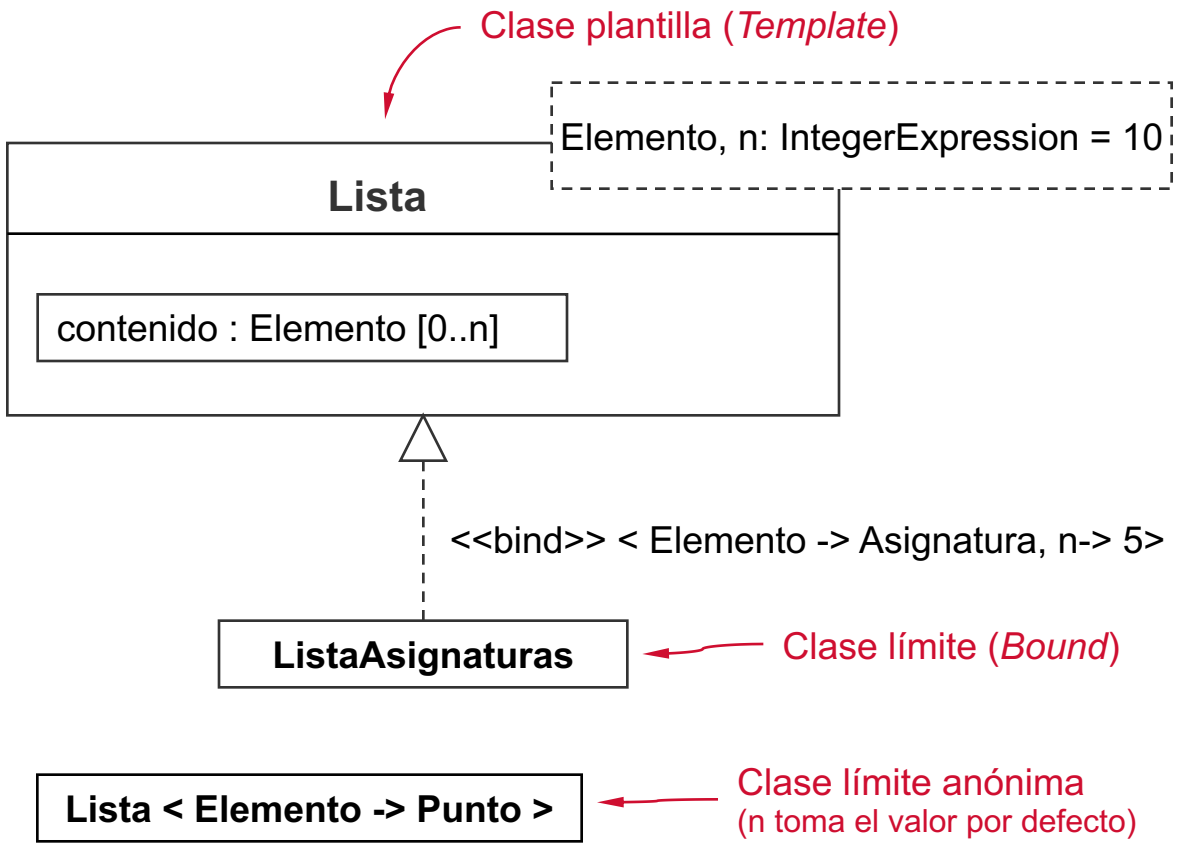
Clases plantilla (*Template Class*) (I)

- Una plantilla es un descriptor de una clase con uno o más parámetros formales sin especificar
  - Define una familia de clases
  - Cada clase se especifica asociando valores a los parámetros
- Los parámetros pueden ser
  - Clasificadores (normalmente clases)
  - Tipos primitivos
- La clase que se produce a partir de una plantilla se denomina **clase límite *(Bound class)***



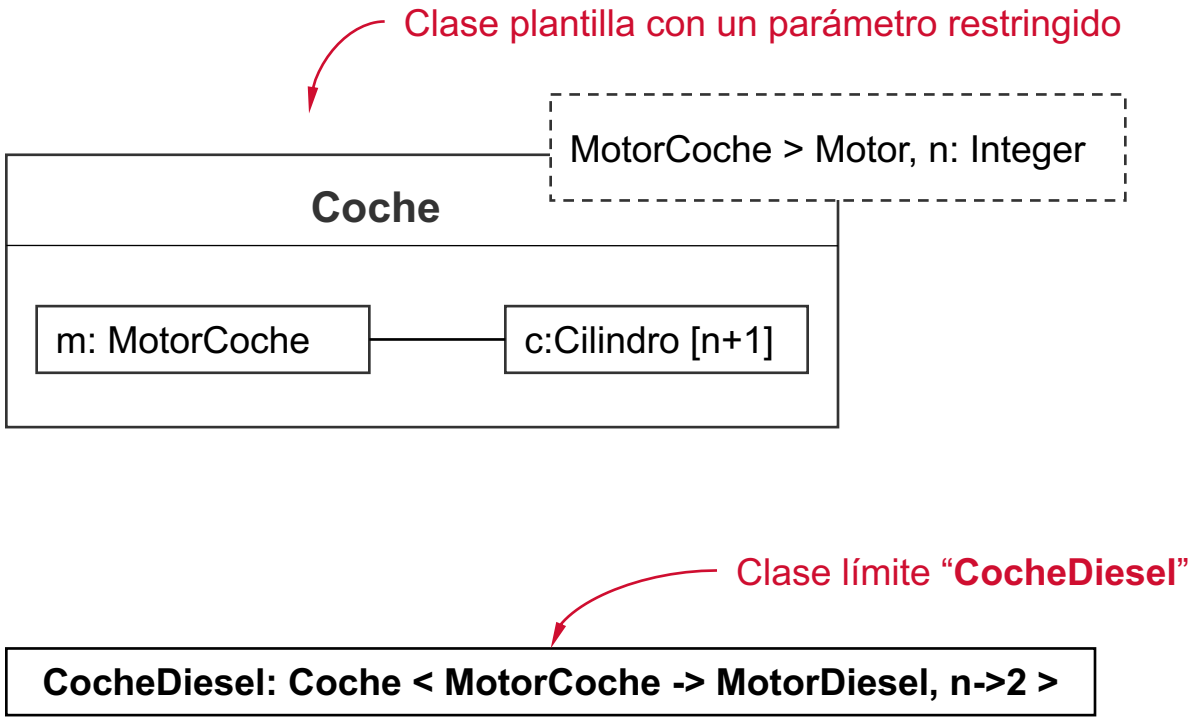
CLASES Y OBJETOS

Clases plantilla (*Template Class*) (II)



CLASES Y OBJETOS

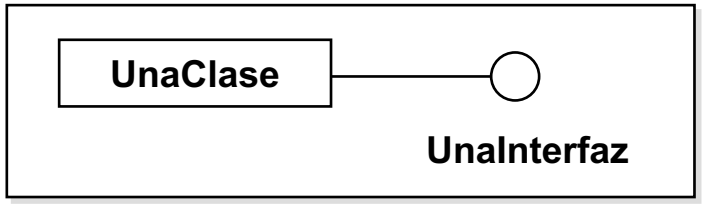
Clases plantilla (*Template Class*) (III)



INTERFACES

Una interfaz es un descriptor de las operaciones visibles externamente de una clase u otra entidad que no especifica la estructura interna

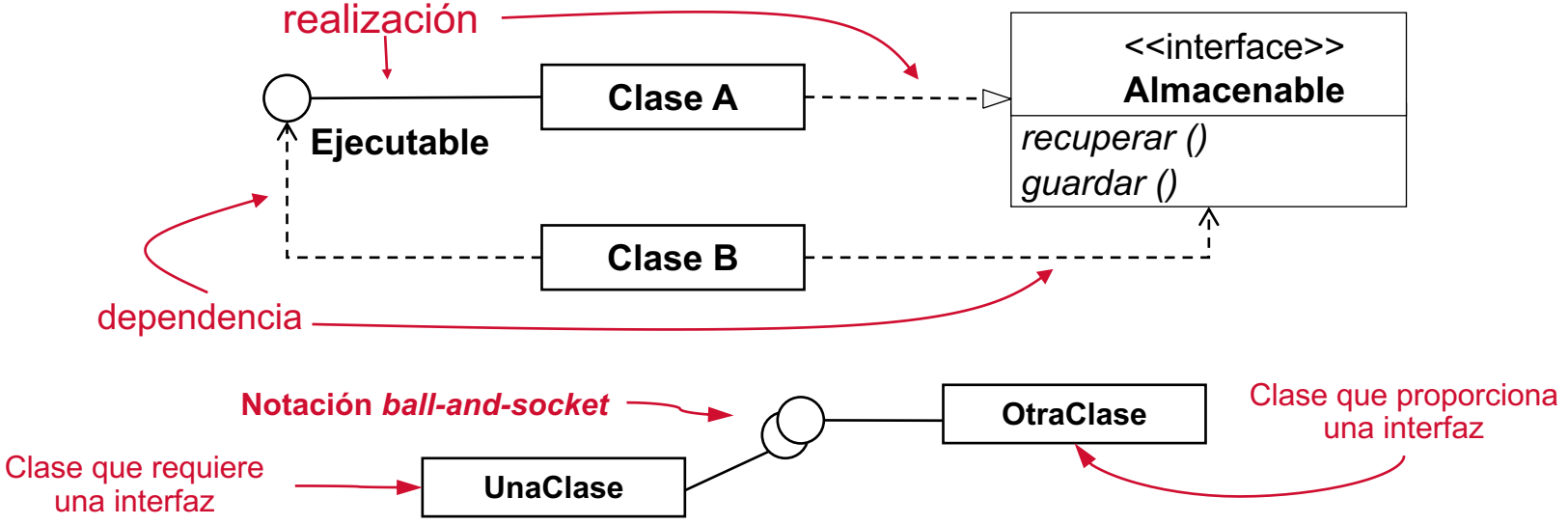
- No contienen atributos ni la implementación de las operaciones
- Las clases (o componentes) que realizan una interfaz tienen que implementar todas las operaciones de la interfaz (directamente o por derivación)
- Una clase puede admitir muchas interfaces, cuyos efectos podrán ser disjuntos o solapados
- Una interfaz no puede tener una asociación que sea navegable partiendo de la interfaz
- Las interfaces pueden tener relaciones de generalización
- Todas las operaciones de una interfaz tienen visibilidad pública
- Notación:
  - Como una clase con el estereotipo << interface>>
  - Mediante un pequeño círculo con el nombre de la interfaz situado debajo del símbolo



INTERFACES

Relaciones entre clases e interfaces

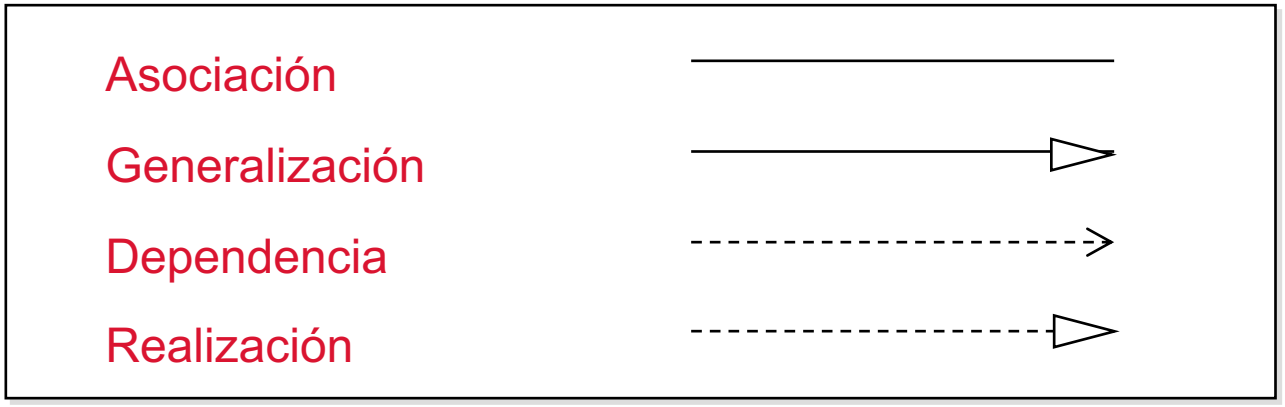
- **Dependencia**: una clase utiliza o requiere operaciones proporcionadas por la interfaz
- **Realización**: indica que la clase implementa todas las operaciones definidas en la interfaz (directamente o por derivación)
- Cuando una clase requiere una interfaz y otra clase proporciona esa interfaz se puede usar la **notación "ball-and-socket"**



RELACIONES

Una relación es una conexión semántica entre elementos de un modelo. Pueden ser de varios tipos:

- **Asociación**: relación que describe un conjunto de enlaces entre objetos
- **Generalización**: relación entre un elemento más general y otro más específico
- **Dependencia**: relación entre elementos, uno dependiente y otro independiente. Un cambio en el independiente afectará al dependiente
- **Abstracción/realización**: relación entre dos descripciones de una misma cosa a diferentes niveles

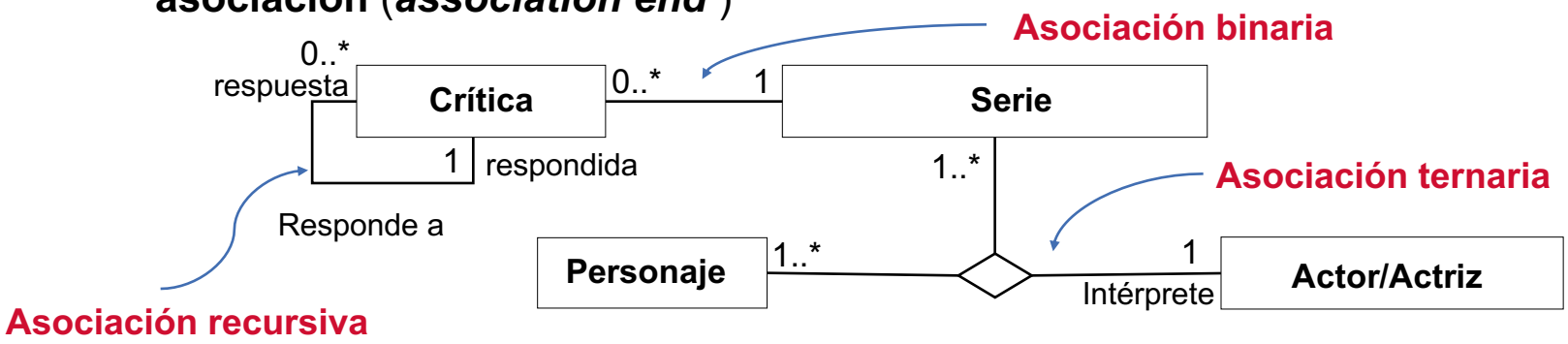


Notación de los diferentes tipos de relaciones

RELACIONES

Asociaciones

- Una asociación describe un conjunto de tuplas cuyos valores se refieren a instancias de un tipo (clase). Una instancia de una asociación es un **enlace**
- Las asociaciones llevan la información sobre relaciones entre objetos en un sistema
- Una asociación puede ser
  - **Recursiva**: conecta una clase consigo misma (conexión semántica entre objetos de la misma clase)
  - **Binaria**: conexión entre dos clases
  - **Ternaria** o de **orden superior** (**n-arias**): conexión entre tres o más clases
- Cada conexión de una asociación a una clase se llama **extremo de la asociación** (***association end***)

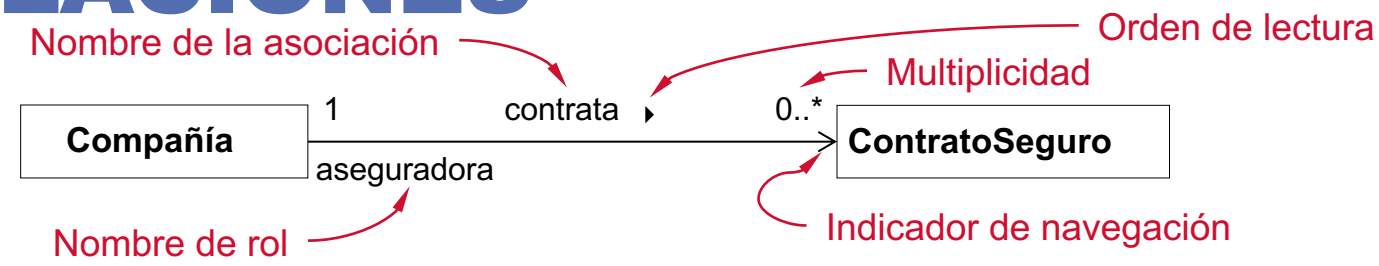


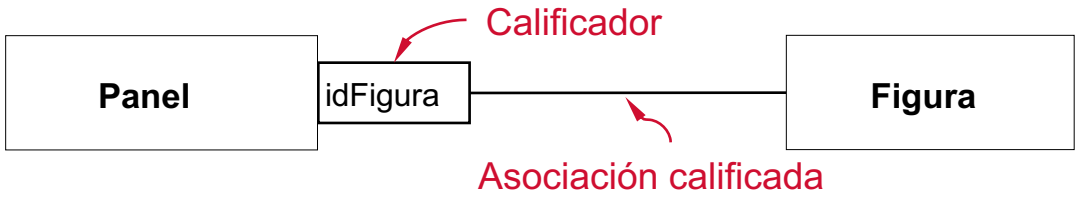
RELACIONES

Adornos de una asociación

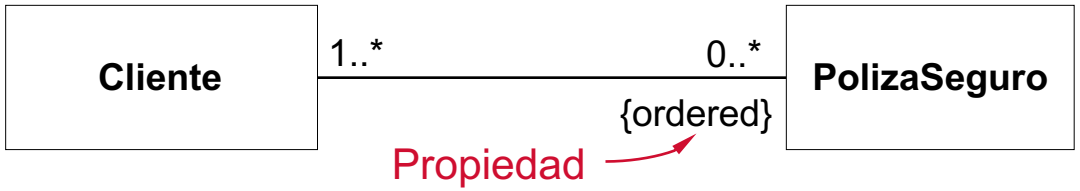
- La línea de asociación puede tener en el centro el **nombre de la asociación** y un triángulo sólido (orden de lectura)
- El **extremo de la asociación** puede contar con los siguientes *adornos*
  - **Símbolo de agregación**
  - **Nombre del extremo/rol** : representa el subconjunto de instancias del clasificador que participa en la asociación
  - **Indicador de navegación**: Una flecha indica que la asociación es navegable en un determinado sentido. Una x indica que el extremo en el que se sitúa no es navegable
  - **Multiplicidad**: Indica cuantos objetos pueden conectarse a través de una instancia de la asociación. En cada extremo de la asociación se puede indicar
    - uno: 1
    - cero o uno: 0..1
    - muchos: *
    - uno o más: 1..*
    - un número exacto: 3 (por ejemplo)
  - **Visibilidad**: se puede limitar la visibilidad en la navegación entre los objetos de una clase de la asociación y los objetos de la otra clase
  - **Calificación**: particionamiento de un conjunto de objetos relacionados con otro objeto mediante un atributo de la asociación (calificador)
  - **Cadena de propiedades**

RELACIONES

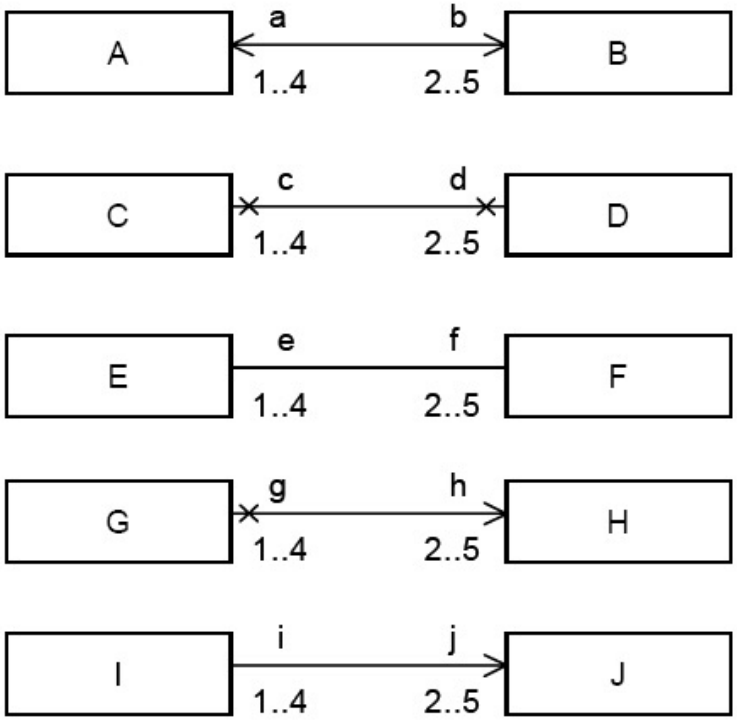

Nombre de la asociación
Orden de lectura
Multiplicidad
Compañía
1
contrata
0..*
ContratoSeguro
aseguradora
Nombre de rol
Indicador de navegación


Calificador
Panel
idFigura
Figura
Asociación calificada

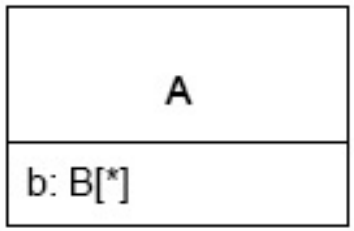

GrupoUsuarios
*
*
Usuario
1
*
clave
Visibilidad
+ usuario
+ propietario
- clave


Cliente
1..*
0..*
PolizaSeguro
{ordered}
Propiedad

RELACIONES



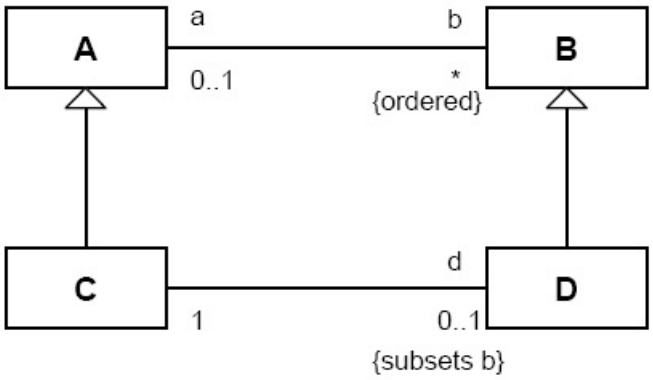
Ejemplos de extremos de navegación



Ejemplo de extremo navegable

RELACIONES

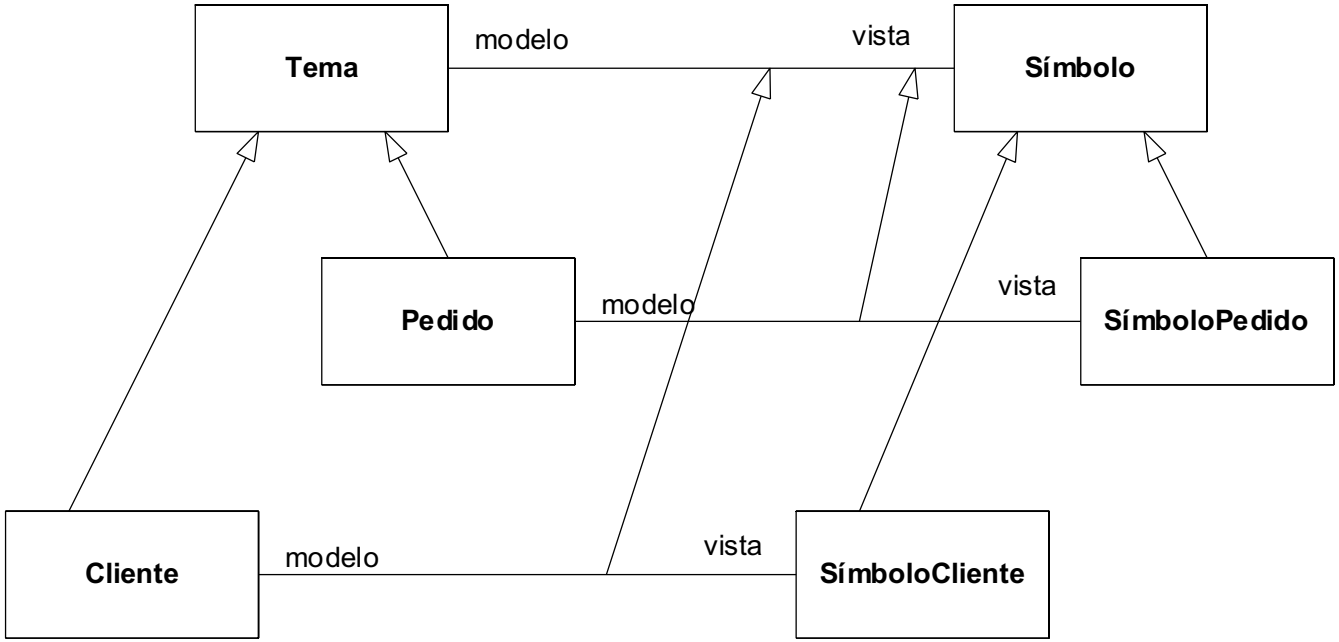
- **{subsets <nombrePropiedad>}** el extremo es un subconjunto de la propiedad llamada <nombrePropiedad>
- **{redefines <nombreExtremo>}** el extremo redefine otro extremo llamado <nombreExtremo>
- **{union}** el extremo se deriva de la unión de sus subconjuntos
- **{ordered}** el extremo representa un conjunto ordenado
- **{bag}** el extremo representa una bolsa
- **{sequence}** o **{seq}** el extremo representa una secuencia
- Si el extremo es navegable se le puede aplicar cualquier cadena de propiedades aplicable a los atributos



Cadenas de propiedades aplicables a extremos de una asociación

RELACIONES

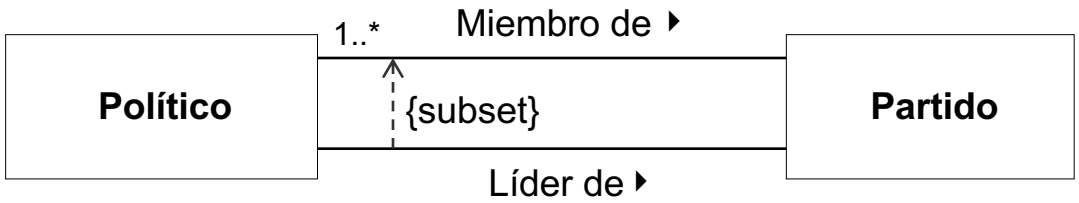
Relaciones entre asociaciones (I)

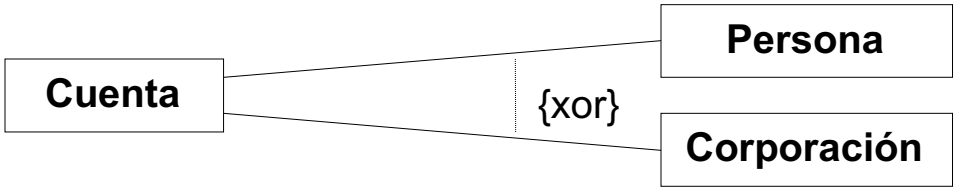


Relaciones de generalización entre asociaciones

RELACIONES

Relaciones entre asociaciones (II)





Restricciones entre asociaciones

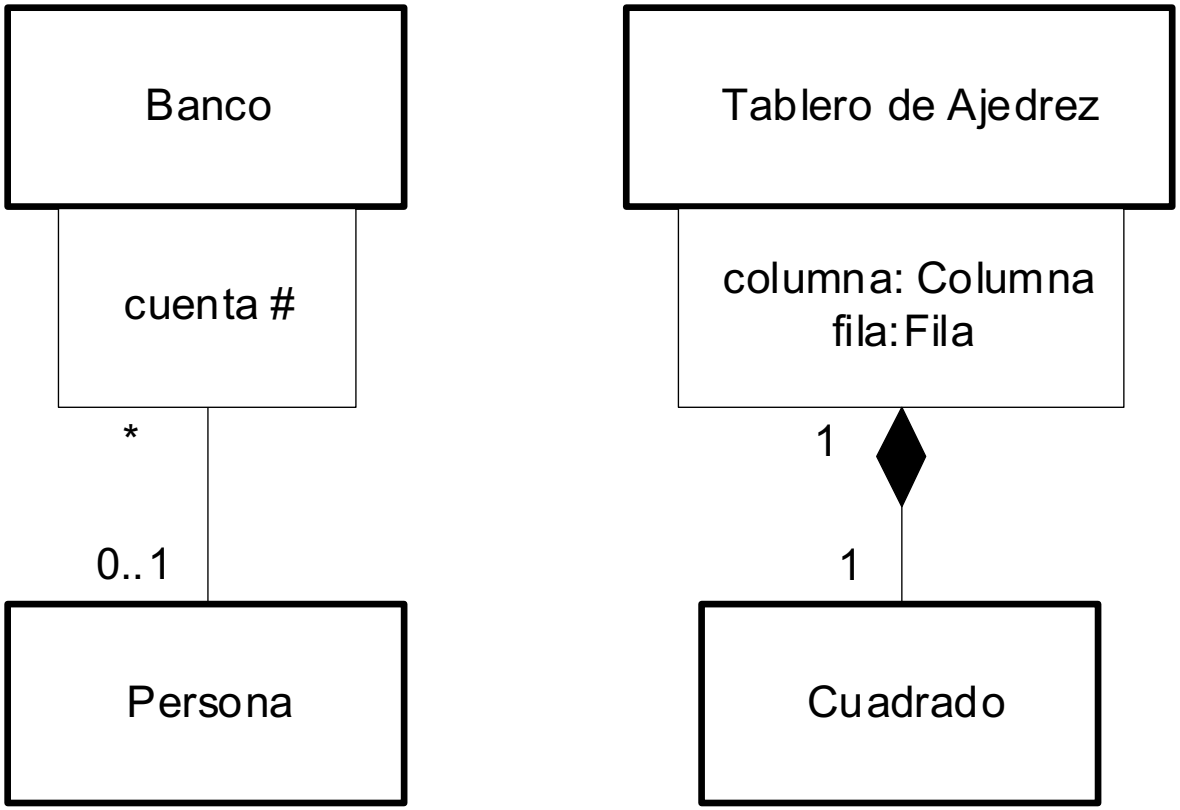
RELACIONES

Asociación calificada (I)

- Un calificador es un atributo o tupla de atributos de la asociación cuyos valores sirven para partir o clasificar un conjunto de objetos asociados mediante una asociación UML a un objeto
- Un calificador se representa como un pequeño rectángulo conectado por un lado al final de una asociación, y por el otro a la clase indicada por ese extremo de la asociación, que será la clase fuente
- El rectángulo del calificador es parte de la asociación y no parte de la clase
- Reduce la multiplicidad del rol opuesto al considerar el valor del calificador

RELACIONES

Asociación calificada (II)



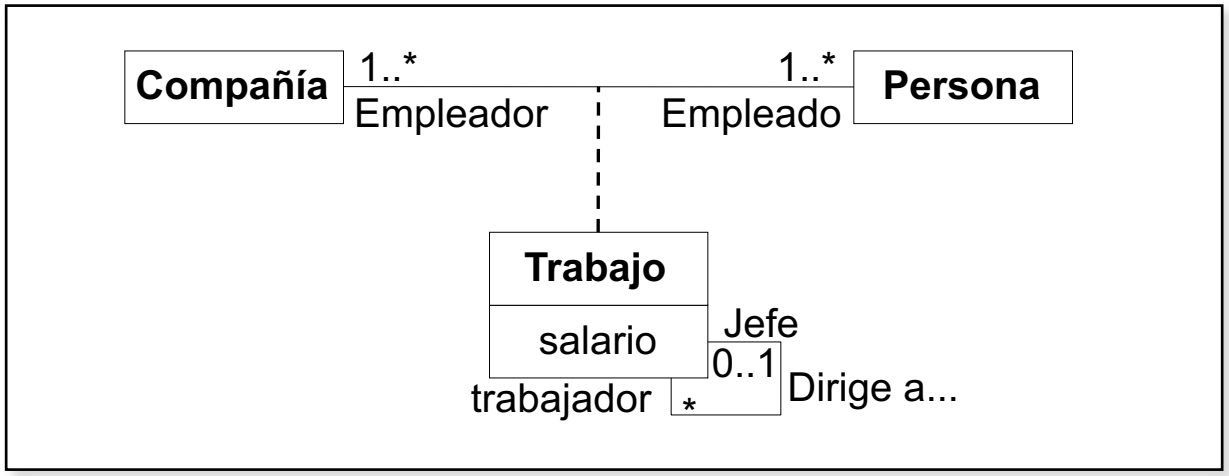
RELACIONES

Clases asociación (I)

- Son asociaciones que también son clases
- Tienen propiedades tanto de las clases como de las asociaciones
- Se utilizan cuando cada enlace debe tener sus propios valores para los atributos, operaciones propias o sus propias referencias a objetos
- Pueden tener operaciones que modifiquen los atributos del enlace o que añadan o eliminen enlaces al propio enlace
- Pueden participar en otras asociaciones
- Cada instancia de una clase asociación tiene referencias a objetos, así como valores para los atributos especificados por la parte de la clase
- Una clase asociación **C** que conecta dos clases **A** y **B** no es lo mismo que una clase **D** con asociación binaria a cada una de las clases **A** y **B**
    - Una asociación, incluyendo a las clases asociación, es un conjunto de tuplas y no tiene duplicados entre sus referencias a objetos, mientras que una relación implícita (clase **D**) es más como una bolsa, que puede tener duplicados

RELACIONES

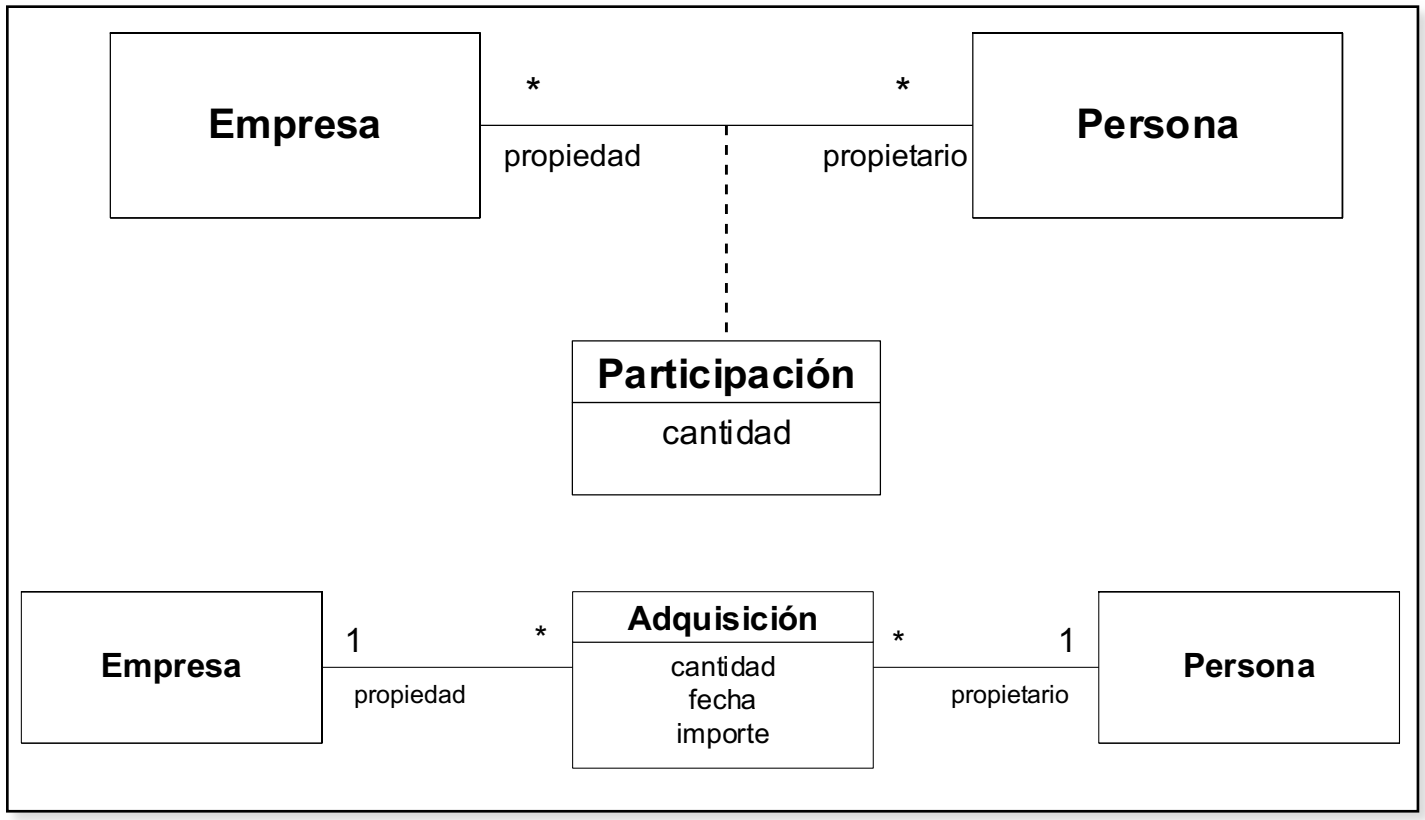
Clases asociación (II)



Ejemplo de clase asociación

RELACIONES

Clases asociación (III)



Clase asociación vs asociación modelada como clase

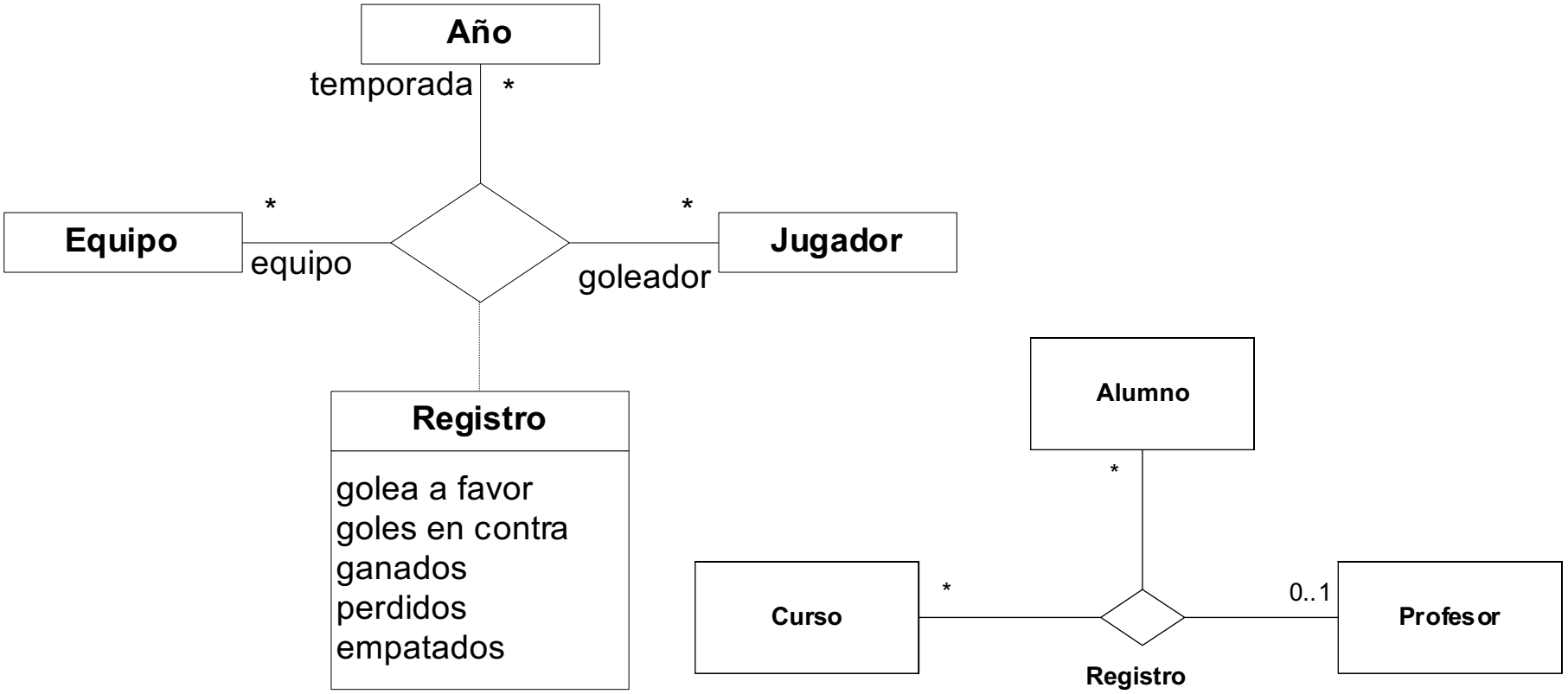
RELACIONES

Asociación n-aria (I)

- Son asociaciones que se establecen entre más de dos clases
  - Cada clase podría aparecer varias veces desempeñando cada vez distintos roles
- Las asociaciones n-arias se representan a través de rombo que se une a través de un camino con cada una de las clases que asocia
- La multiplicidad en las relaciones n-arias se puede especificar, pero es menos obvio que en el caso de las relaciones binarias
- La multiplicidad al final de una asociación representa el número potencial de instancias, cuando los otros n-1 están fijados
  - La multiplicidad con que una clase participa en una relación n-aria se define con respecto a las otras n-1 clases
  - Así, dada una relación entre las clases (**A**, **B** y **C**)
    - La multiplicidad de **C** indica cuántas instancias de **C** pueden asociarse con una pareja cualquiera de instancias de **A** y **B**. Si por ejemplo la multiplicidad es (**muchos, muchos, uno**), significa que para cada pareja posible (**A, B**) existe una única instancia de **C**. Para una pareja (**B, C**) dada, existen varias instancias de **A**
  - No se puede definir la multiplicidad con respecto a una única clase porque la multiplicidad sería varios para cualquier asociación n-aria
- Una asociación n-aria no debe de contener la marca de agregación/composición conectando alguno de sus papeles

RELACIONES

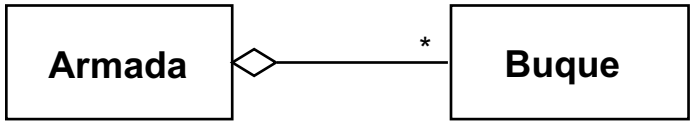
Asociación n-aria (II)



RELACIONES

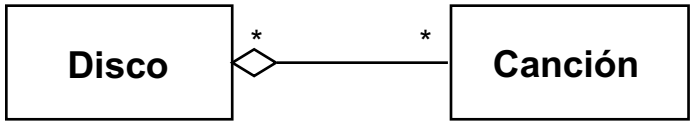
Agregación

- Es una forma de asociación que representa una relación ***todo-parte*** entre un agregado (el todo) y las partes que los componen



Agregación compartida

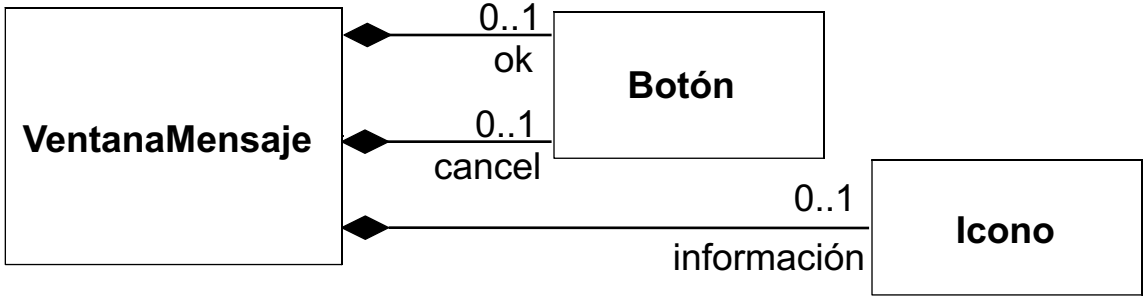
- Agregación en la cual las partes pueden pertenecer a cualquiera de los agregados



RELACIONES

Agregación compuesta (composición)

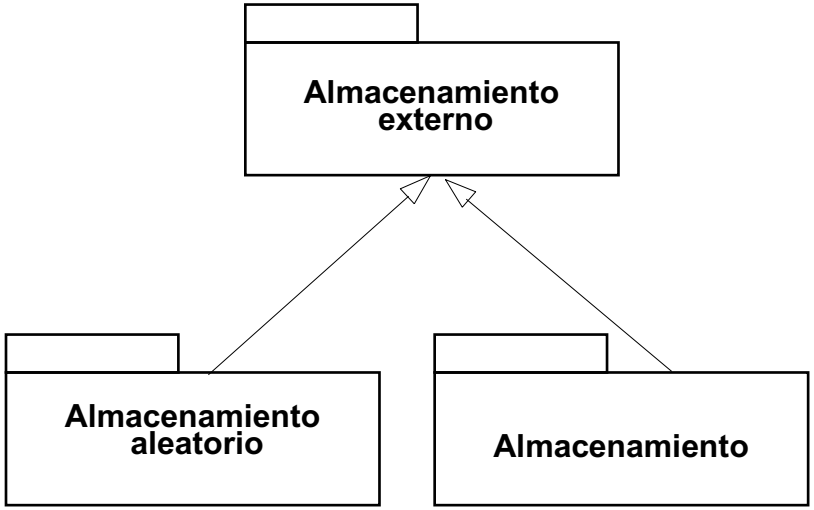
- Es una asociación de agregación con las restricciones adicionales de que un objeto solo puede ser parte de un compuesto a la vez y que el objeto compuesto es el único responsable de la disponibilidad de todas sus partes



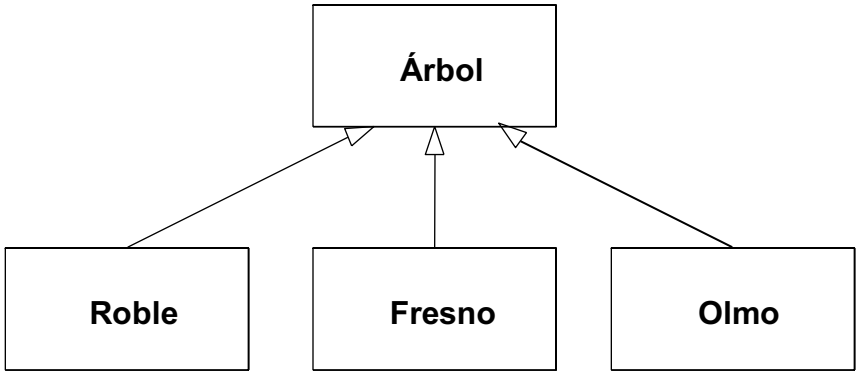
RELACIONES

Generalización

- Es una relación de taxonomía entre un elemento general y otro más específico que es plenamente consistente con el primer elemento y que le añade información adicional
- Cada instancia del clasificador específico es una instancia indirecta del clasificador general



Relación de generalización entre paquetes

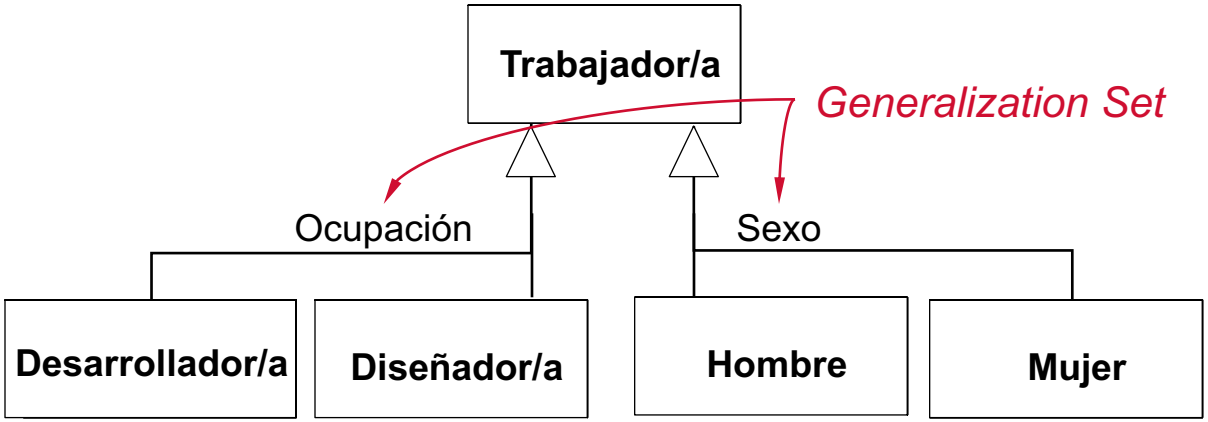


Relación de generalización entre clases

RELACIONES

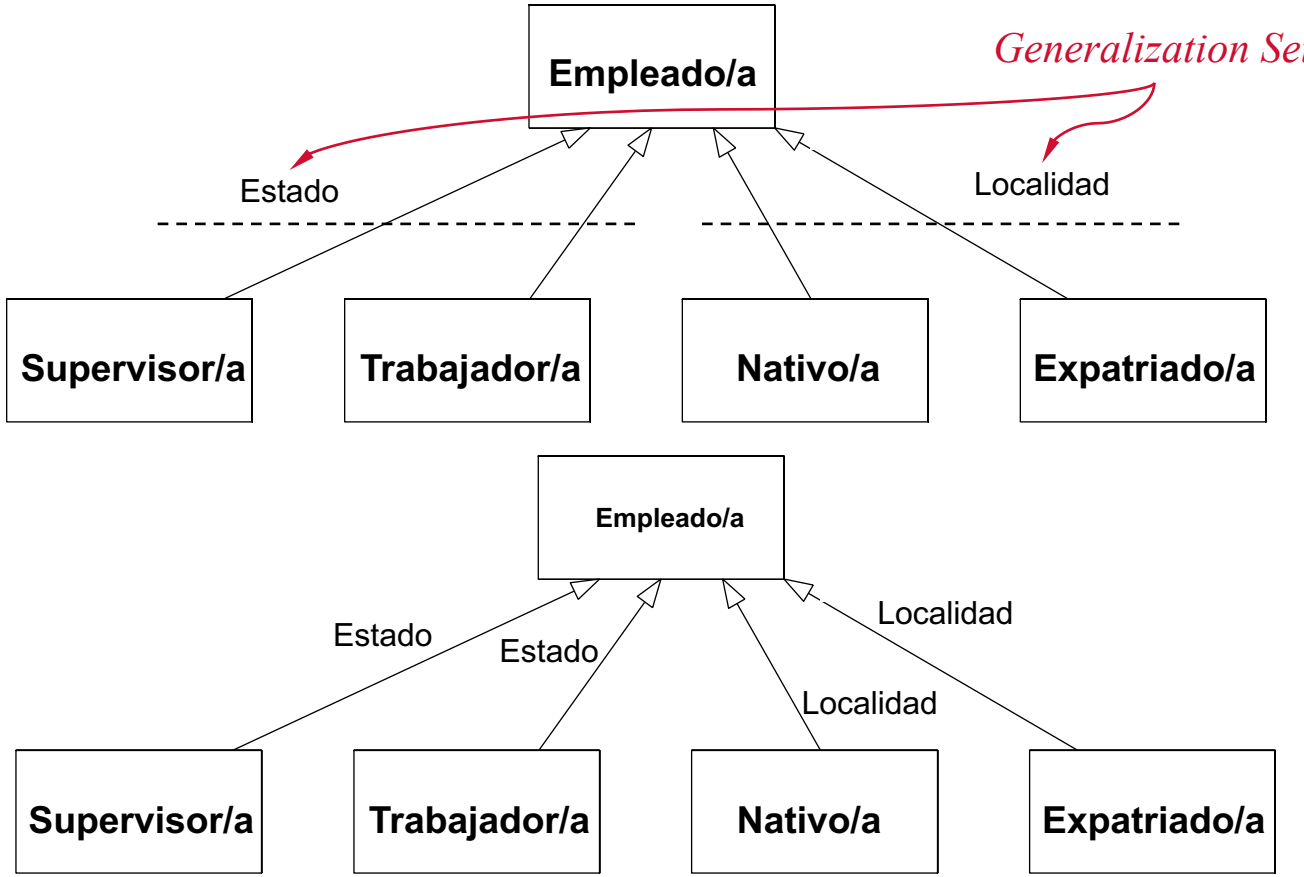
Conjunto de generalización *(Generalization Set)* (I)

- Se utiliza para diferenciar diferentes relaciones de generalización de un clasificador



RELACIONES

Conjunto de generalización (II)



RELACIONES

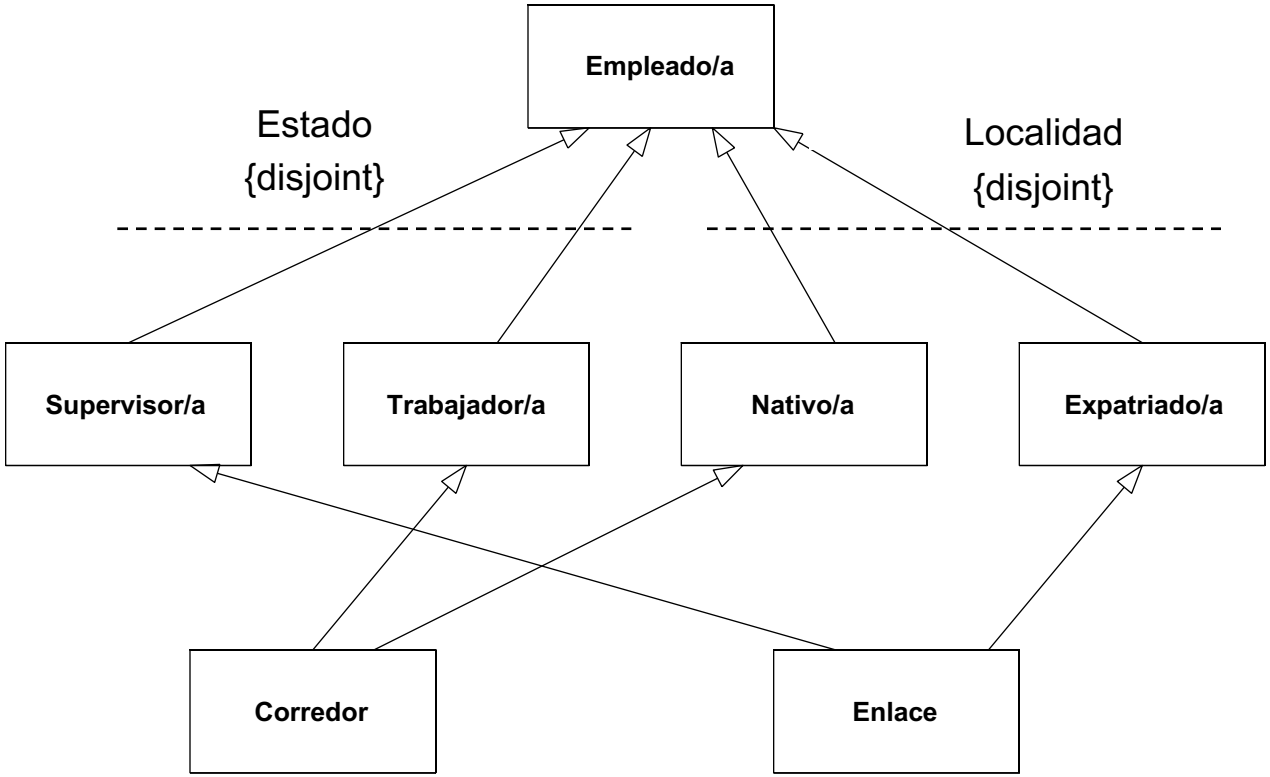
Restricciones de los conjuntos de generalización (I)

- ***Disjoint*** (disjunto) – Ningún elemento puede tener dos hijos en el conjunto como antecesores (en una situación de generalización múltiple). Ninguna instancia puede ser una instancia directa o indirecta de dos de los hijos (en una semántica múltiple de la clasificación)
- ***Overlapping*** (solapado) – Un elemento puede tener dos o más hijos en el conjunto de antecesores. Una instancia puede ser una instancia de dos o más hijos
- ***Complete*** (completo) – Todos los hijos posibles se han enumerado en el conjunto y no puede ser agregado ninguna más
- ***Incomplete*** (incompleto) – No se ha enumerado todavía todos los hijos posibles en el conjunto. Se esperan más hijos o se conocen pero no se han declarado aún

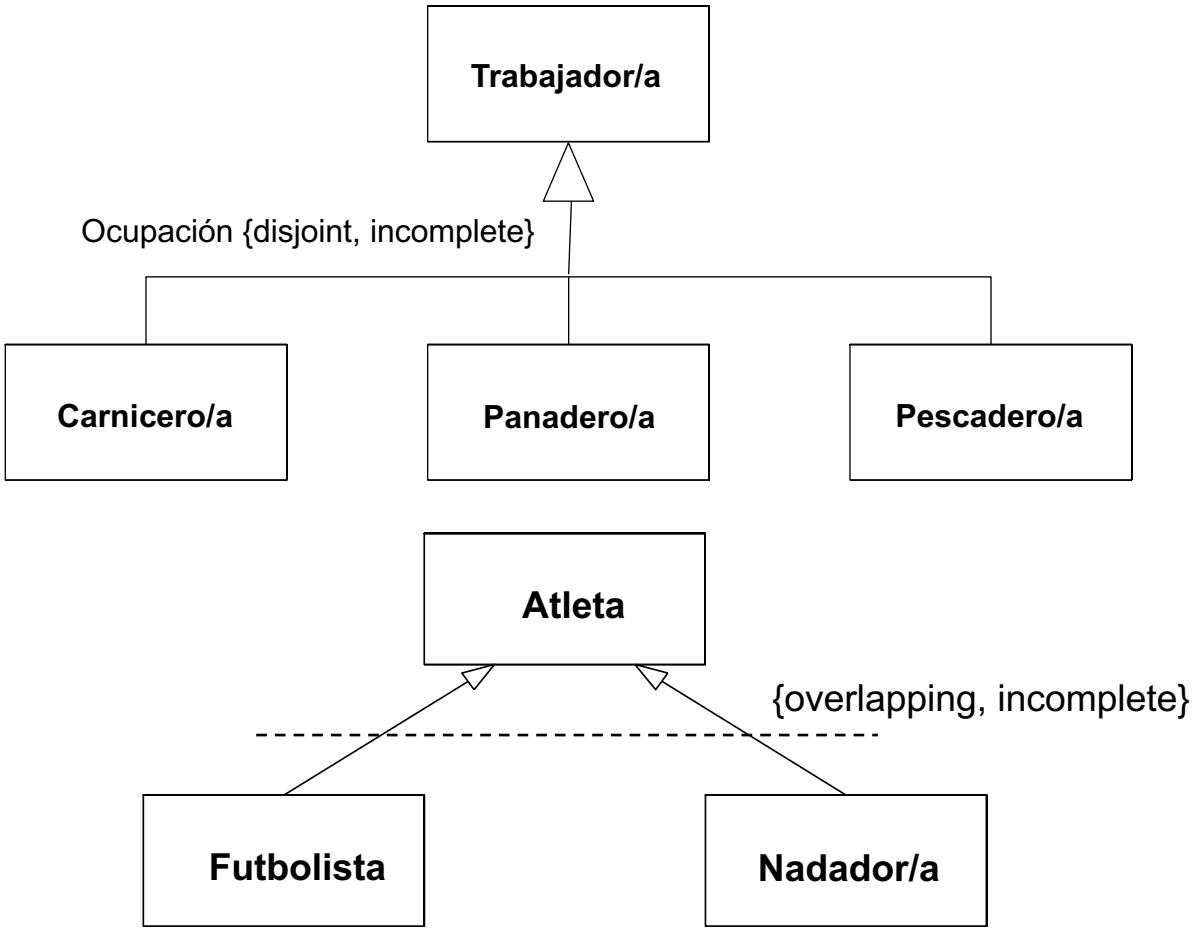
Por defecto: **{incomplete, disjoint}**

RELACIONES

Restricciones de los conjuntos de generalización (II)



RELACIONES

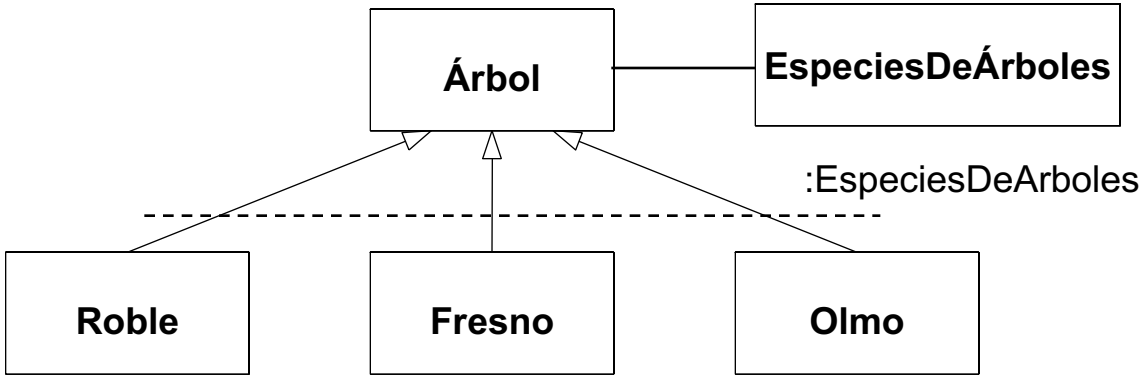


Formas de representar las restricciones de los conjuntos de generalización

RELACIONES

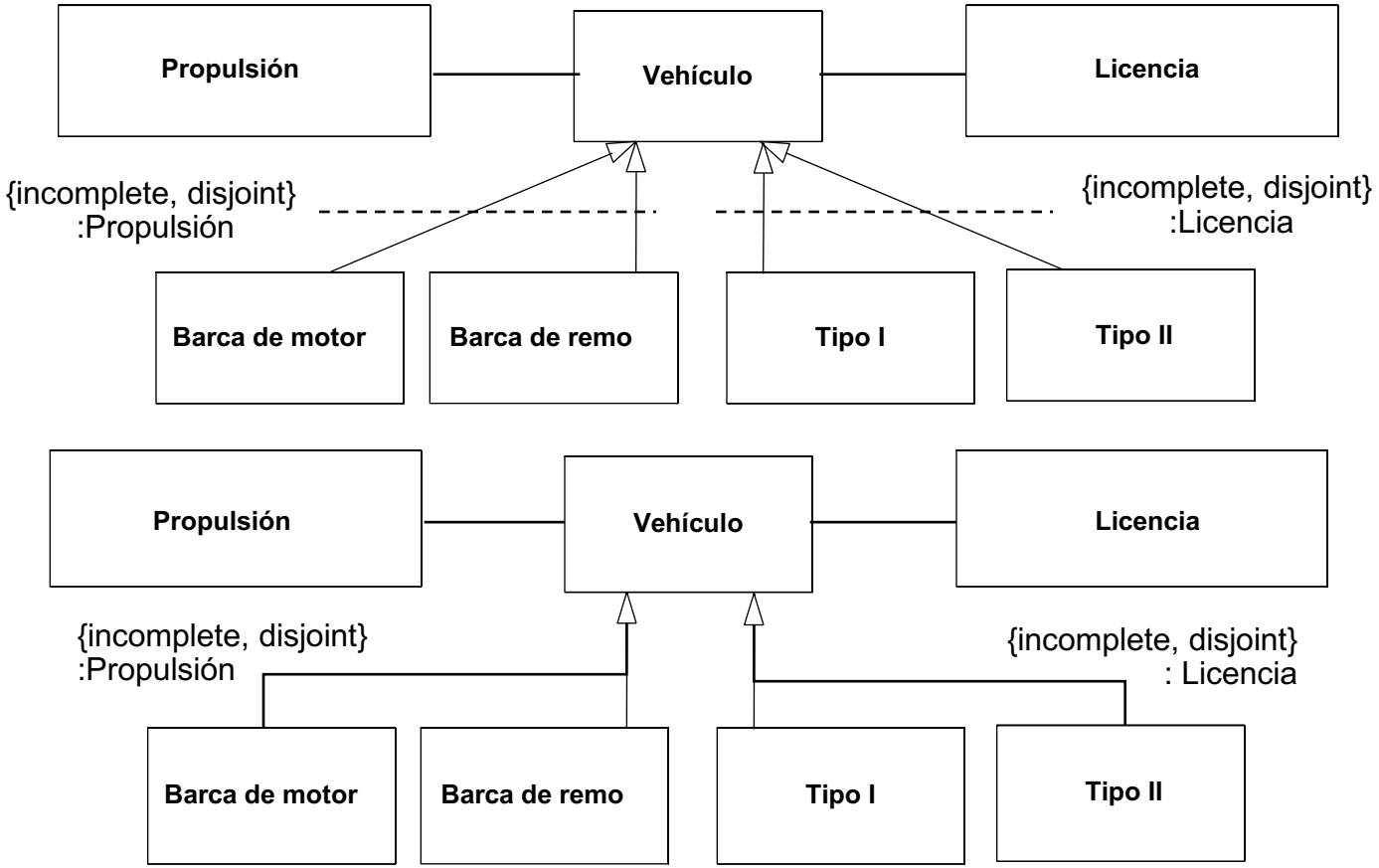
Supratipo (*powertype*) (I)

- Clasificador cuyas instancias son subclases de otro clasificador. Es una metaclase



RELACIONES

Supratipo (II)



RELACIONES

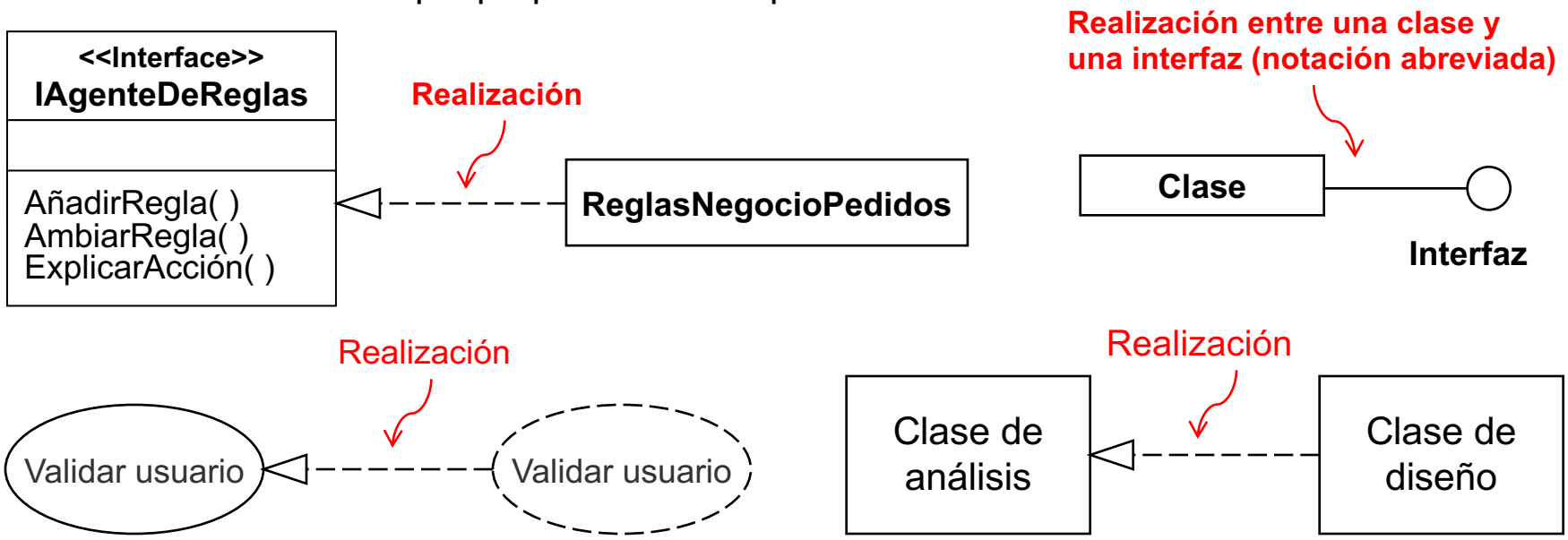
Realización (I)

- Es una relación entre una especificación y su implementación
  - El proveedor proporciona la especificación y el cliente la implementación de dicha especificación
  - El cliente debe soportar (por herencia o por declaración directa) todas las operaciones que el proveedor le proporciona
- La **especificación** describe el comportamiento o la estructura de algo sin determinar cómo se implementará el comportamiento
- Una **implementación** proporciona los detalles relativos a la forma de implementar el comportamiento
  - Un elemento puede realizar más de una especificación
- El elemento **cliente** tiene que admitir todo el comportamiento del elemento proveedor, pero no tiene por qué satisfacer su estructura o su implementación
- El **proveedor** de una realización indica qué operaciones tienen que estar presentes en el cliente, pero es el cliente el que tiene la responsabilidad de aportarlas

RELACIONES

Realización (II)

- La relación de realización es una relación de dependencia con una notación especial:
  - Línea discontinua con una punta de flecha triangular hueca en el extremo del elemento que proporciona la especificación

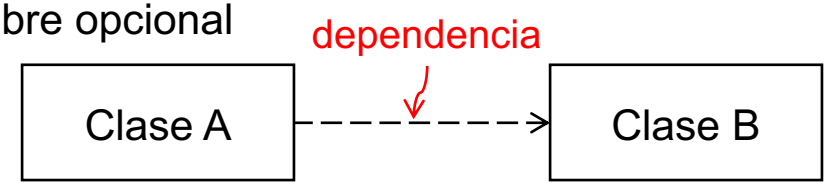


Ejemplos de relaciones de realización

RELACIONES

Dependencia

- Es una **relación semántica entre dos o más elementos del modelo**
- Indica una situación en la que un **cambio en el elemento proveedor** requiere un cambio, o una indicación de cambio en el elemento cliente
- Agrupa varios tipos de relaciones diferentes
- Una dependencia puede tener:
  - Un **nombre** para indicar su **rol** en el modelo
  - Un **estereotipo** para establecer la **naturaleza** precisa de la dependencia
- Una **dependencia entre dos paquetes** indica la presencia de, al menos, una dependencia del tipo indicado entre un elemento de cada paquete
- Una dependencia **se representa** mediante **una flecha con línea discontinua** entre dos elementos del modelo
  - El elemento de modelo en la cola de la flecha (el cliente) depende del elemento en la punta de flecha (el proveedor)
  - La flecha se puede etiquetar con la palabra clave opcional, para indicar el tipo de dependencia, y un nombre opcional

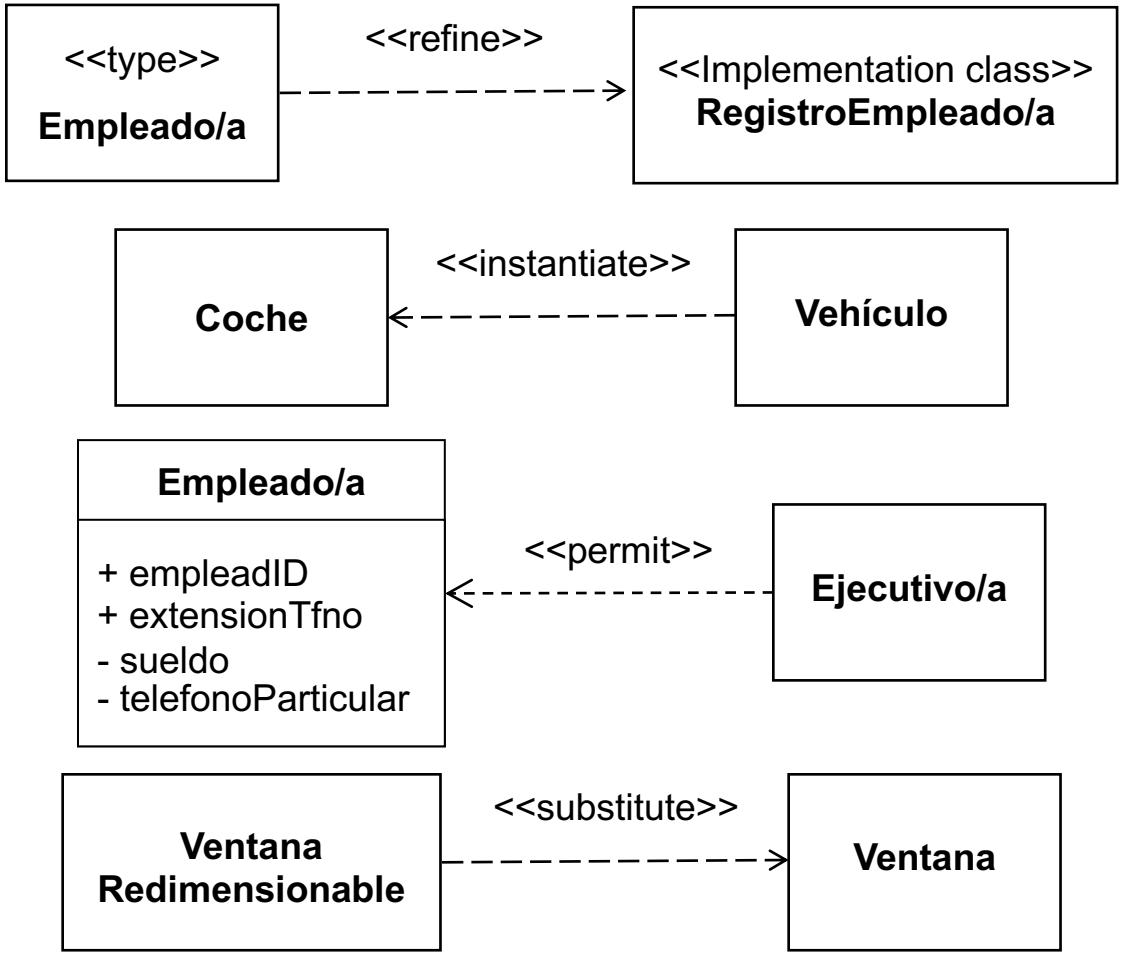


RELACIONES

Tipos de dependencia

- Las relaciones de dependencia pueden ser de varios tipos y pueden llevar asociados diferentes estereotipos
  - **Abstracción**
    - ***derive***
    - ***refine***
    - ***trace***
  - **Uso**
    - ***use***
    - ***call***
    - ***create***
    - ***instantiate***
    - ***send***
  - **Permiso**
    - ***permit***
  - **Sustitución**
    - ***substitute***

RELACIONES



Ejemplos de tipos de dependencia

BIBLIOGRAFÍA

- García-Peñalvo, F. J., Moreno García, M. N., García-Holgado, A., & Vázquez-Ingelmo, A. (2022). UML. Unified Modeling Language. In F. J. García-Peñalvo, A. García-Holgado, & A. Vázquez-Ingelmo (Eds.), *Recursos docentes de la asignatura Ingeniería de Software I. Grado en Ingeniería Informática. Curso 2021-2022*. Grupo GRIAL, Universidad de Salamanca. https://doi.org/10.5281/zenodo.5979552

FUNDAMENTOS DE LA VISTA ESTÁTICA

INGENIERÍA DE SOFTWARE I

2º DE GRADO EN INGENIERÍA INFORMÁTICA
CURSO 2021/2022

Dr. Francisco José García Peñalvo / fgarcia@usal.es
Alicia García Holgado / aliciagh@usal.es
Andrea Vázquez Ingelmo / andreavazquez@usal.es

Departamento de Informática y Automática
Universidad de Salamanca