escuelaingenieríaindustrial
PONTIFICIA UNIVERSIDAD CATOLICA DE VALPARAISO

Diagramas de clases de UML

Franco Guidi Polanco

Escuela de Ingeniería Industrial

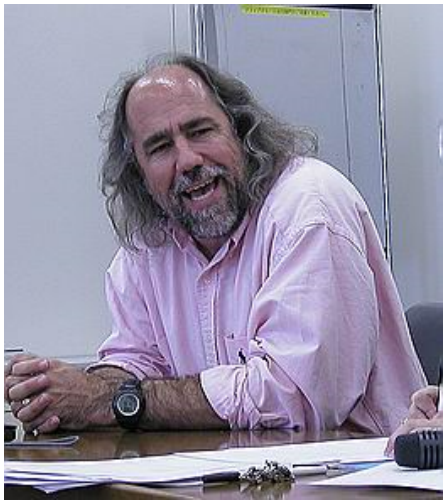
Pontificia Universidad Católica de Valparaíso, Chile

fguidi@ucv.cl

¿Qué es UML?

- UML ("Unified Modeling Language") es un lenguaje visual para crear modelos de sistemas.
- UML fue desarrollado por el trabajo conjunto de los "Tres Amigos"
- Está compuesto por distintos diagramas, para apoyar distintas etapas de desarrollo:
  - Análisis
  - Diseño
  - Instalación (deployment)

Los "Tres Amigos"



Grady Booch

Ivar Jacobson

Jim Rumbaugh

¿Por qué usar UML?

- UML es principalmente una herramienta de comunicación:
  - … con uno mismo
  - … con los miembros de un equipo de desarrollo
  - … con el cliente
- Ventajas de utilizarlo:
  - Permite capturar adecuadamente los requerimientos
  - Apoya correcta comprensión de un sistema por parte de distintos miembros de un proyecto de desarrollo

Diagramas de UML

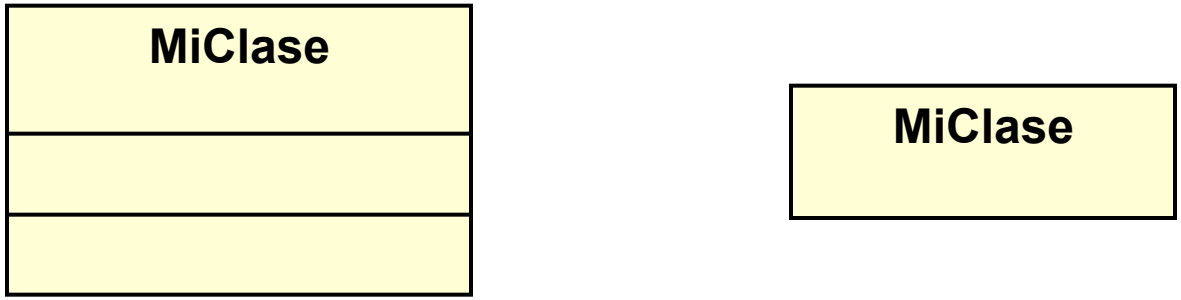
- Casos de Uso
- Clases
- Objetos
- Statechart
- Actividades
- Secuencia
- Colaboración
- Componentes

Diagrama de clases de UML

- Describe las clases y muestra las relaciones entre ellas.

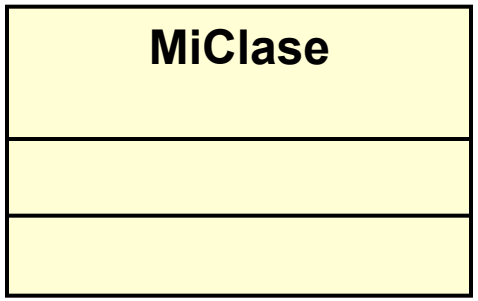
- Tipos de relaciones:
  - Is-a: una clase es del tipo de otra clase
  - Asociaciones entre clases:
    - Una clase contiene a otra clase (Has-a)
      - Agregación
      - Composición
    - Una clase usa otra clase (Uses-a)
    - Una clase crea a otra clase

Representación de clases



- La figura de la izquierda muestra el símbolo para una clase en su forma completa, y el de la derecha en su forma abreviada.
- Por convención, los nombres de clases comienzan con mayúsculas y deben estar escritos con letra de tipo **bold** en sus símbolos.

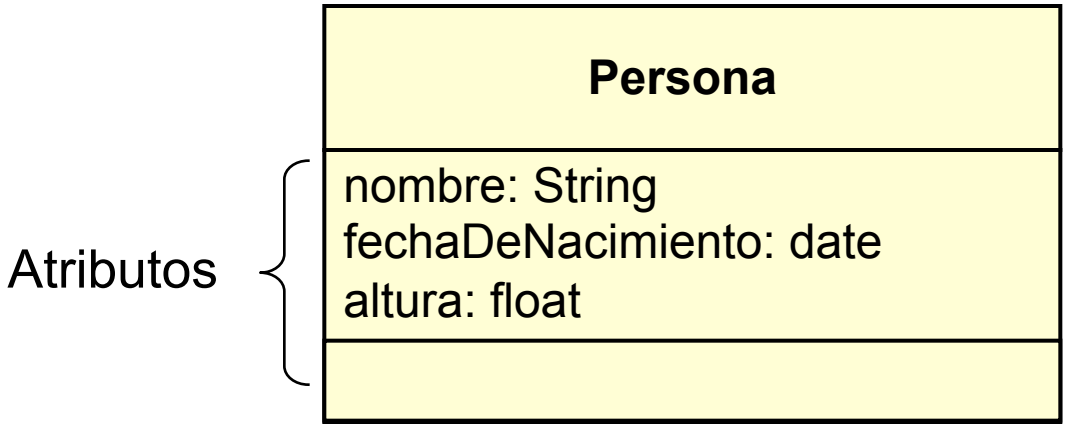
Representación de clases (II)



- En la forma completa del símbolo:
  - El compartimento superior está destinado al nombre de la clase.
  - El compartimento del medio muestra los atributos de la clase.
  - El compartimento inferior muestra las operaciones.

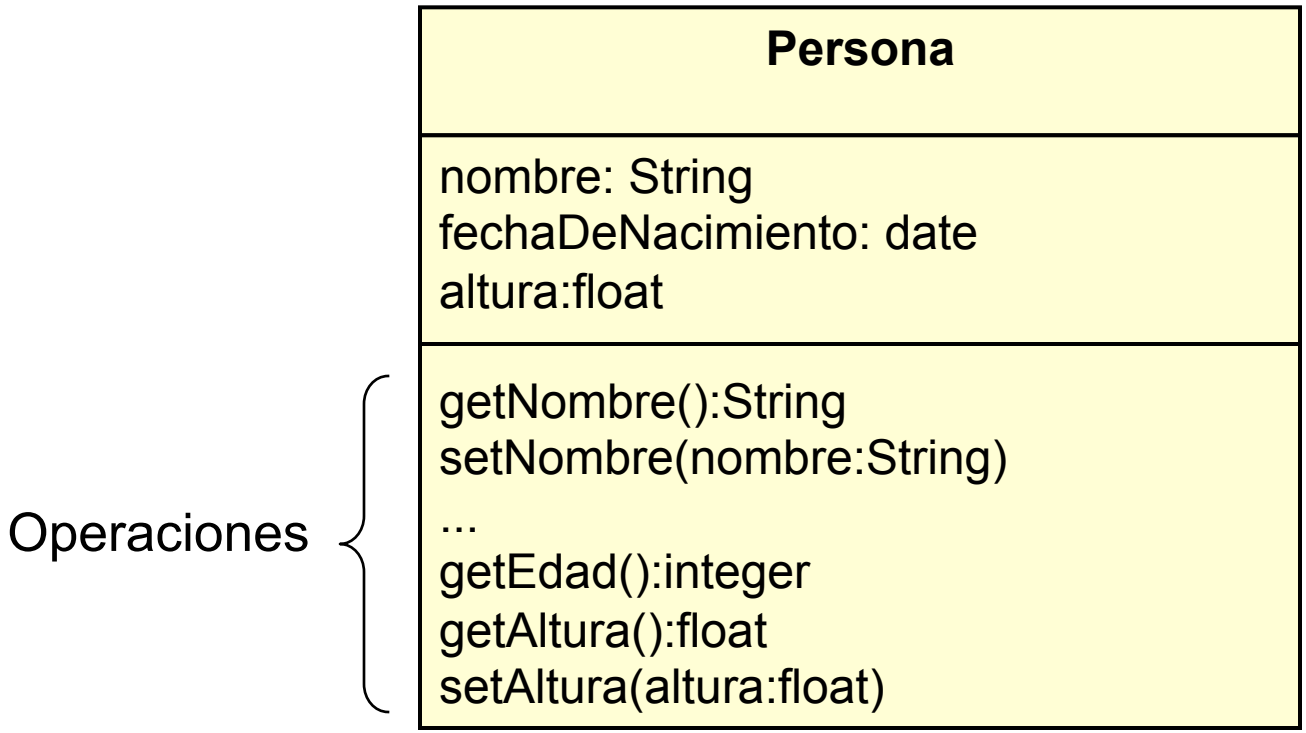
Atributos

- Los atributos representan información acerca de un objeto.
- El término atributo no es exactamente sinónimo de variable. Un atributo representa una propiedad definida en términos abstractos, mientras que una variable es el mecanismo de implementación del atributo.



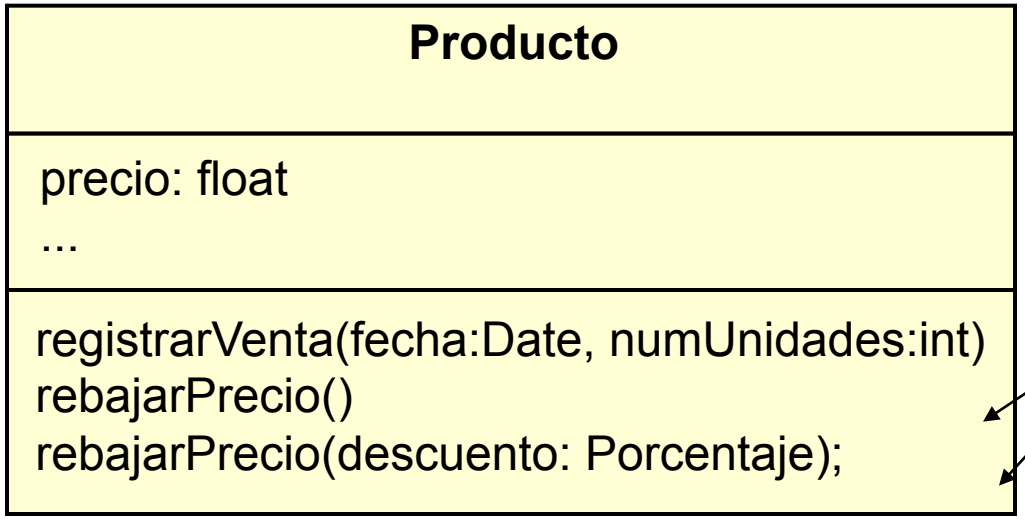
Operaciones

- Se ubican en el compartimento inferior de las clases.



Operaciones sobrecargadas

- Las operaciones sobrecargadas aparecen varias veces en el símbolo de la clase (en cada ocasión con diferente cantidad o tipo de argumentos).



Una de las versiones de la operación rebajarPrecio reduce el precio del producto en una cantidad predeterminada y la otra recibe un porcentaje de descuento.

Visibilidad de atributos y operaciones

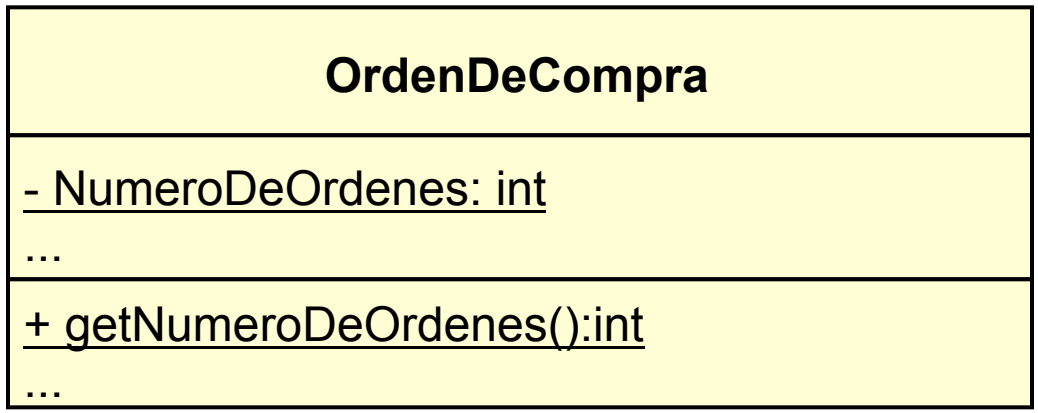
- UML añade un prefijo a las operaciones y atributos para indicar su visibilidad:
  - + para atributos y operaciones públicas .
  - # para atributos y operaciones protegidas.
  - – para atributos y operaciones privadas.
- Si se omite el prefijo, se asume que el atributo u operación es pública.

Atributos y operaciones de clases

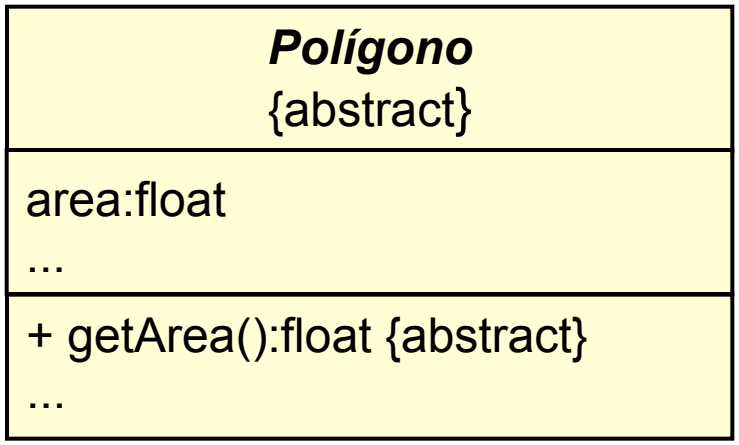
- Los atributos y operaciones de clase (aquellos que no pertenecen a una instancia en particular sino que son compartidos por toda la clase) se representan en UML subrayados.

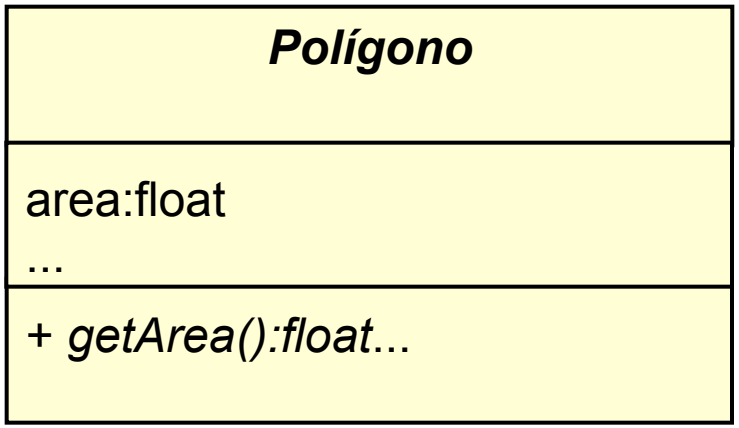
Registra el número de órdenes de compra creadas.

Obtiene en número de órdenes de compra creadas.



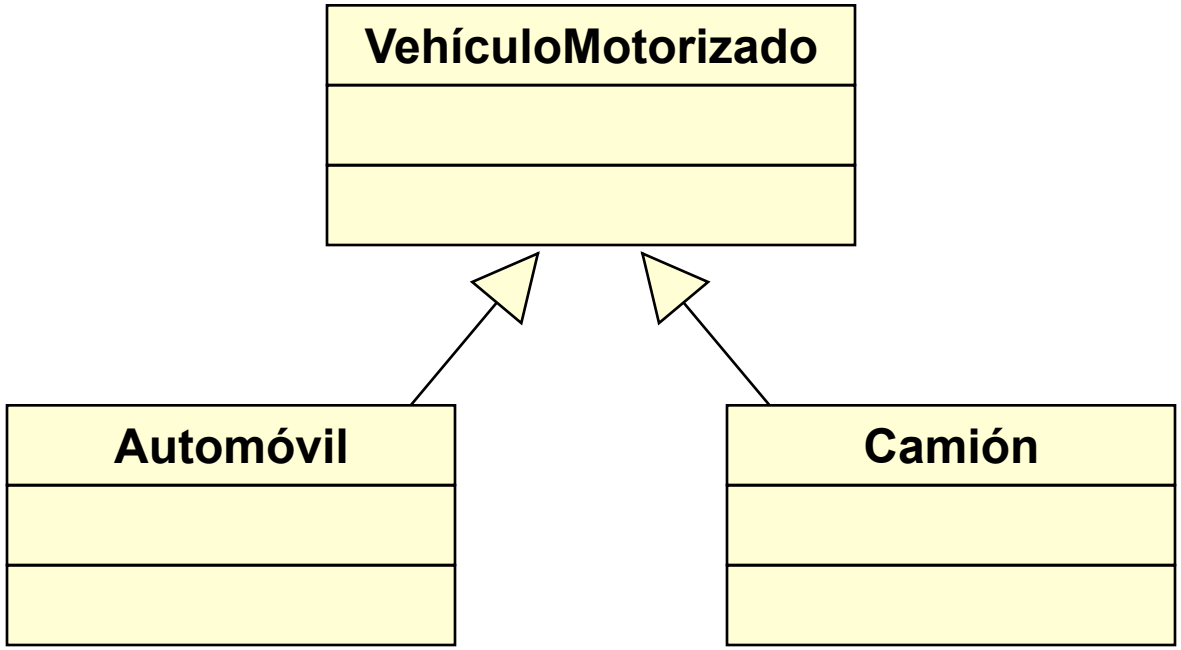
Operaciones y clases abstractas





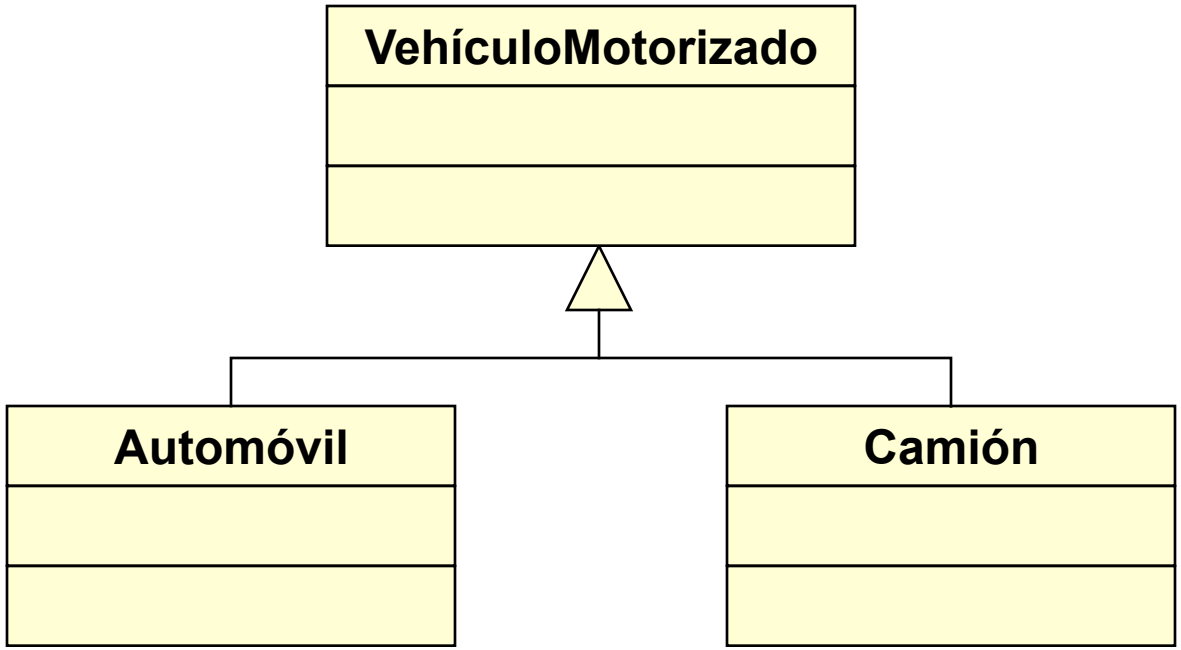
- El nombre de una clase abstracta debe estar en estilo *itálico* o con la indicación {abstract}.
- Las operaciones abstractas también deben estar en estilo *itálico* o con la indicación {abstract}.

Generalización: Herencia simple



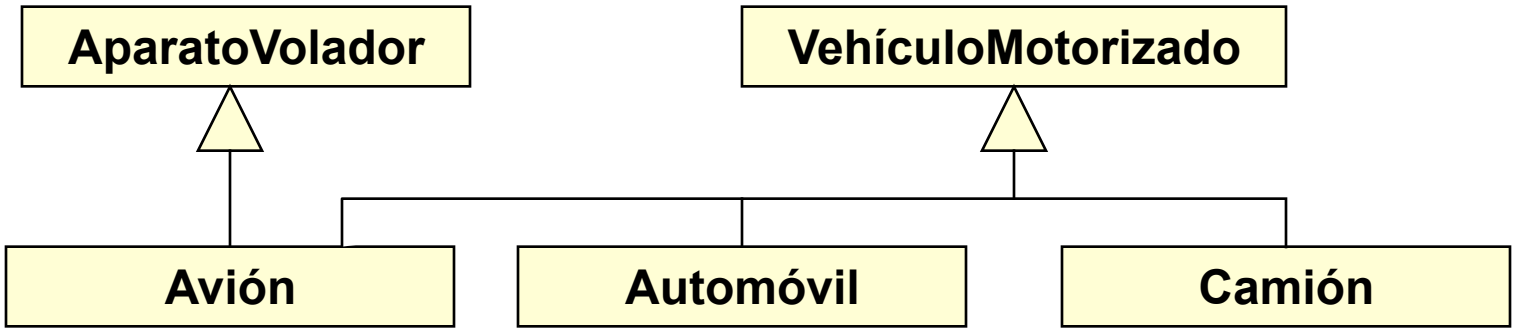
- Una jerarquía de herencia se muestra utilizando flechas que apuntan hacia arriba en la jerarquía (en el ejemplo: Automóvil y Camión son subclases de VehículoMotorizado).

Generalización: Herencia simple (II)



- Otro estilo para mostrar una jerarquía de herencia.

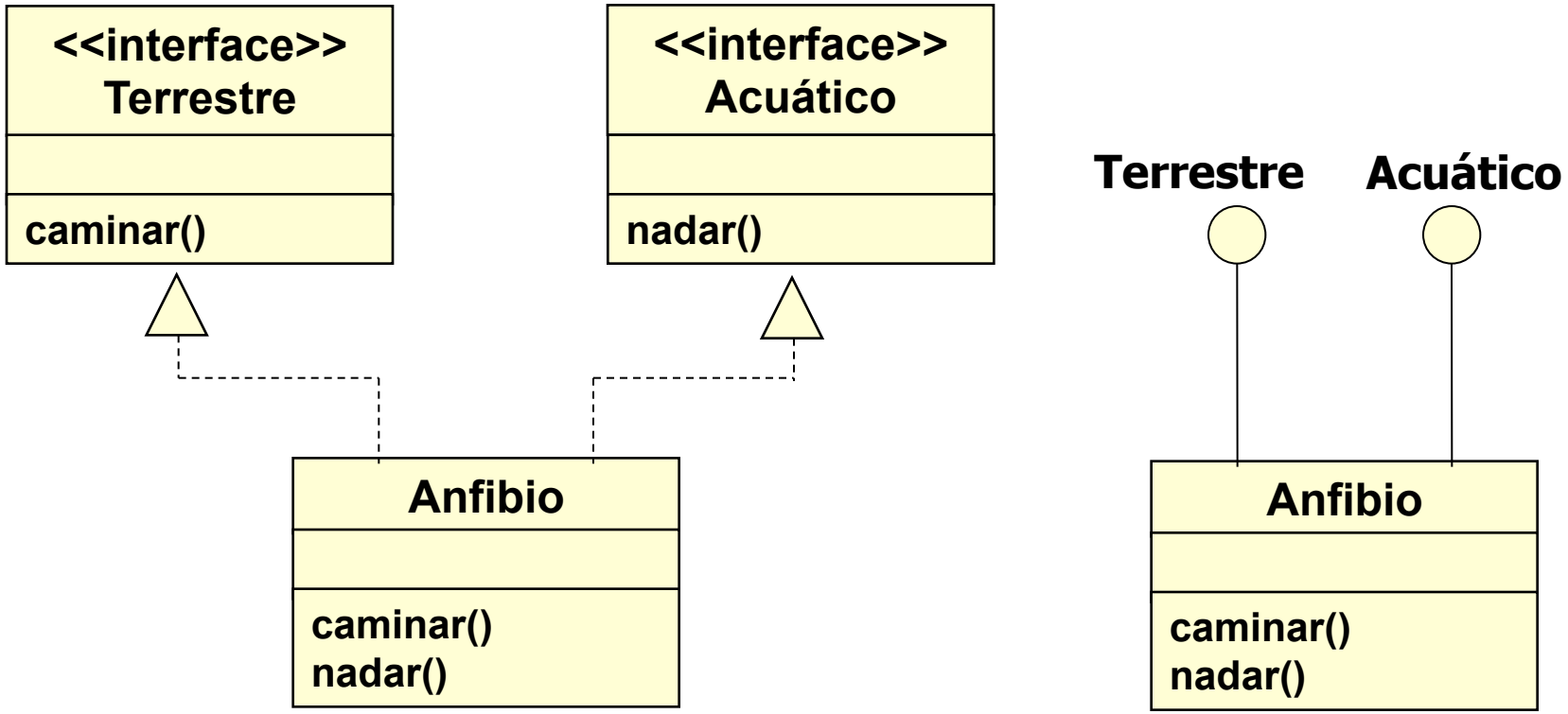
Generalización: Herencia múltiple



- UML permite mostrar herencia múltiple (cuando una clase hereda directamente de más de una superclase).
- En el ejemplo, un Avión es un AparatoVolador y un VehículoMotorizado.

Implementación de interfaces

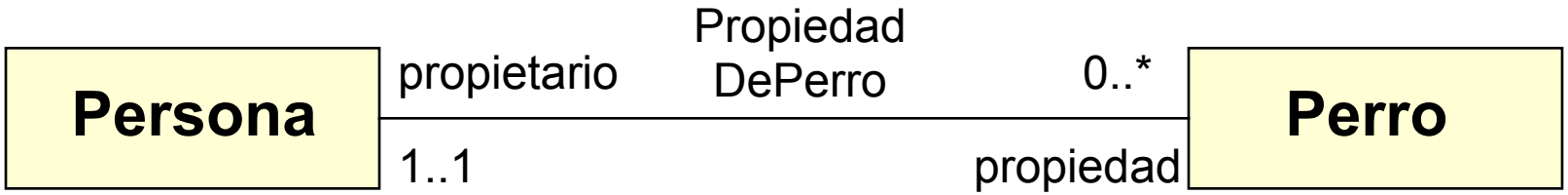
- Notaciones alternativas:



Asociaciones

- Una asociación caracteriza un cierto tipo de relación que puede darse entre instancias de determinadas clases.
- Por ejemplo, si tenemos las clases Persona y Perro, las siguientes relaciones podrían darse entre sus instancias:
  - Juan es propietario de Fido
  - Pedro es propietario de Rintintín
  - Pedro es propietario de Lassie

Asociaciones (II)

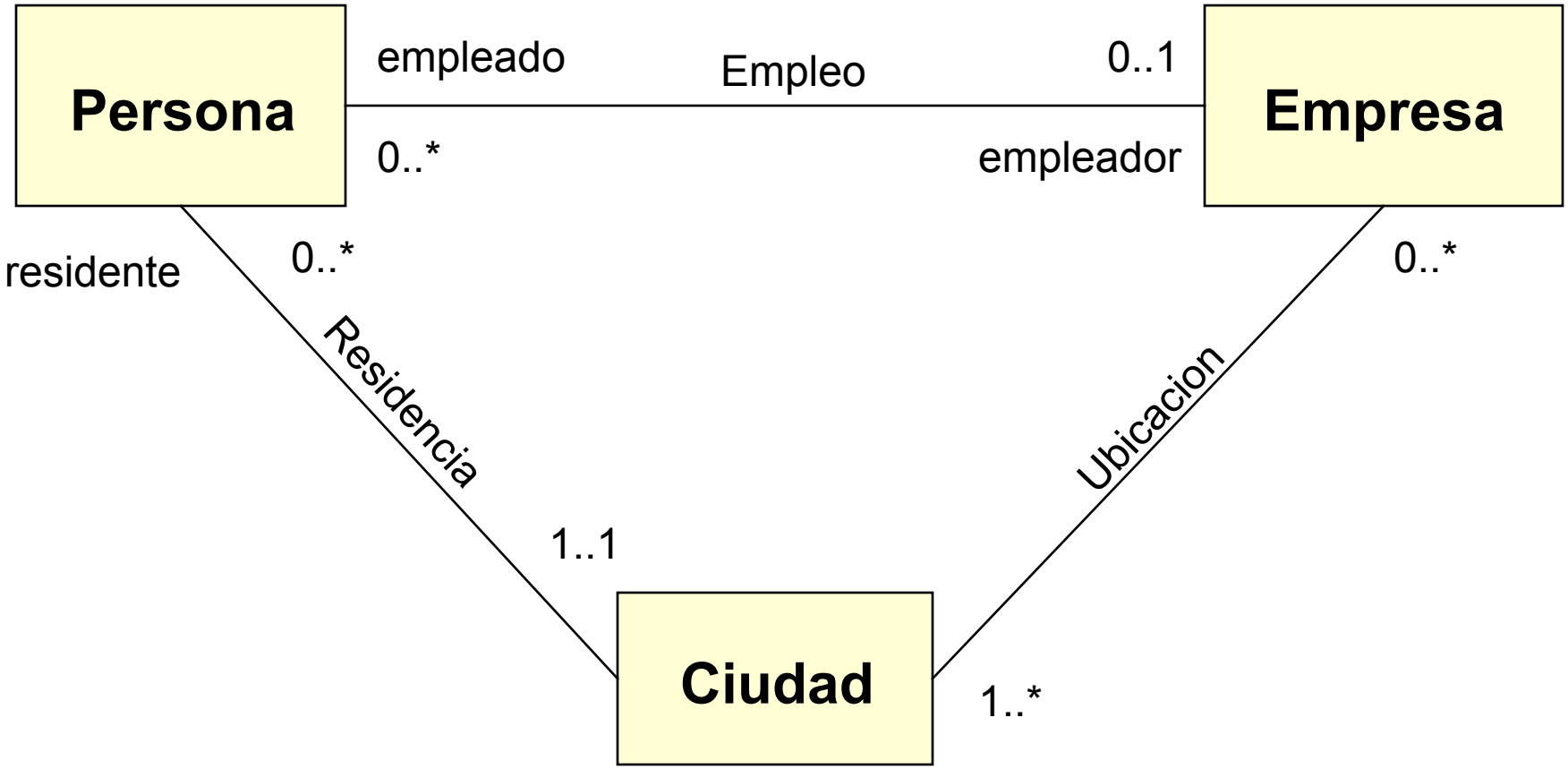


- La asociación muestra que existe una relación de propiedad entre personas y perros, por la cual una persona puede ser propietario de cero o más perros y un perro es propiedad de una única persona.

Asociaciones (III)

- Cada asociación se muestra como una línea entre dos clases.
- El nombre de la asociación aparece en la línea.
- El rol de cada clase en la asociación aparece al lado de la clase, al final de la línea.
- La multiplicidad de la asociación también aparece al final de la línea.

Ejemplo de asociaciones



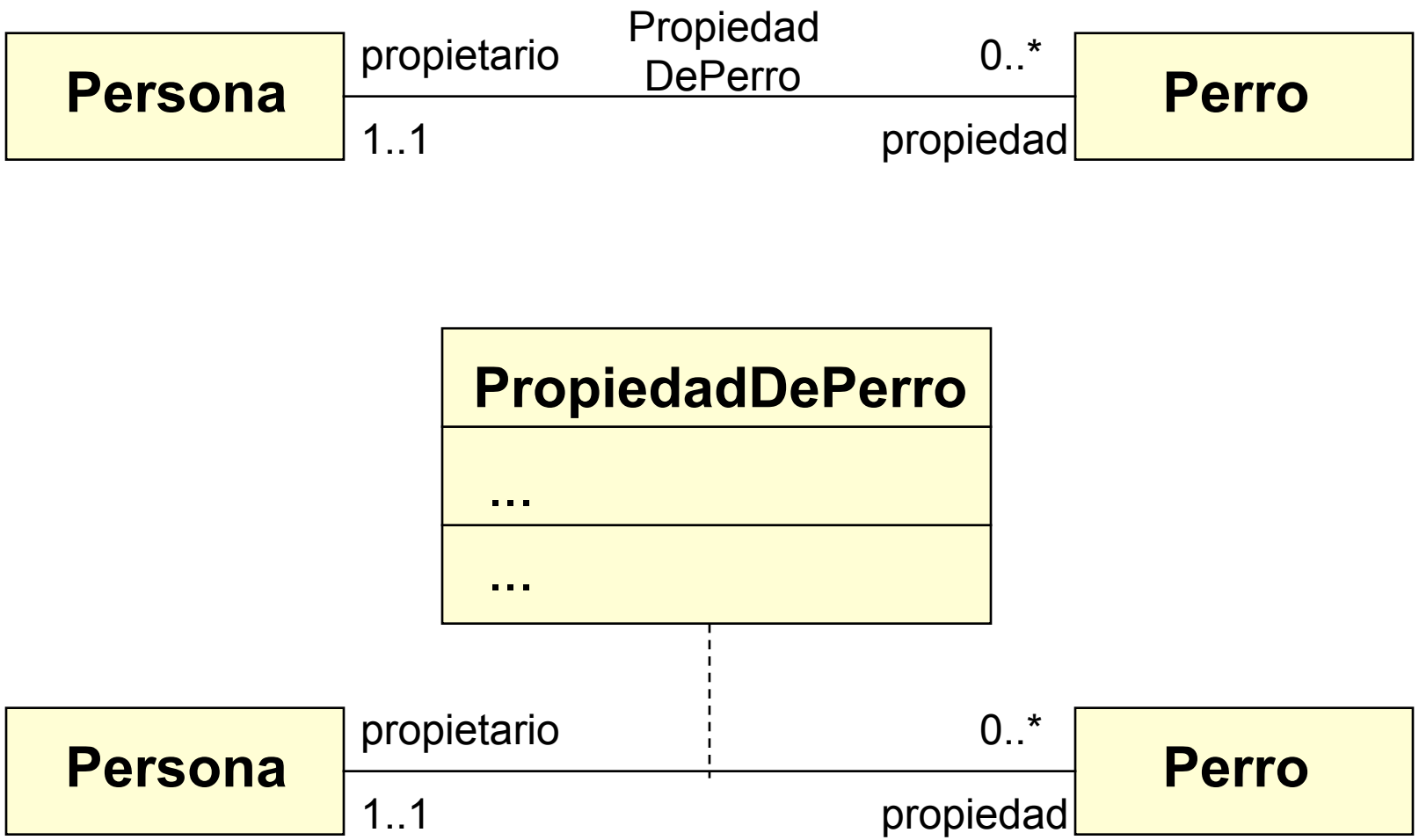
Ejemplo de asociaciones (II)

- Como las empresas emplean personas, existe una asociación entre las clases Empresa y Persona. El nombre de esta asociación es Empleo.
- El rol de la persona es el de empleado en la asociación de empleo.
- Una persona puede ser empleada en 0 ó 1 empresas (0..1); una empresa puede emplear a 0 o más personas (0..*).

Más sobre asociaciones

- No es obligatorio poner nombres a las asociaciones. Sin embargo es recomendable (se nombran con un sustantivo singular).
- No es necesario poner nombres de roles tampoco.
- La multiplicidad en un diagrama puede ser debatible, depende de lo que interese representar en el modelo.
- Puede existir más de una asociación entre un par de clases. Asimismo, una clase puede tener una asociación consigo misma.

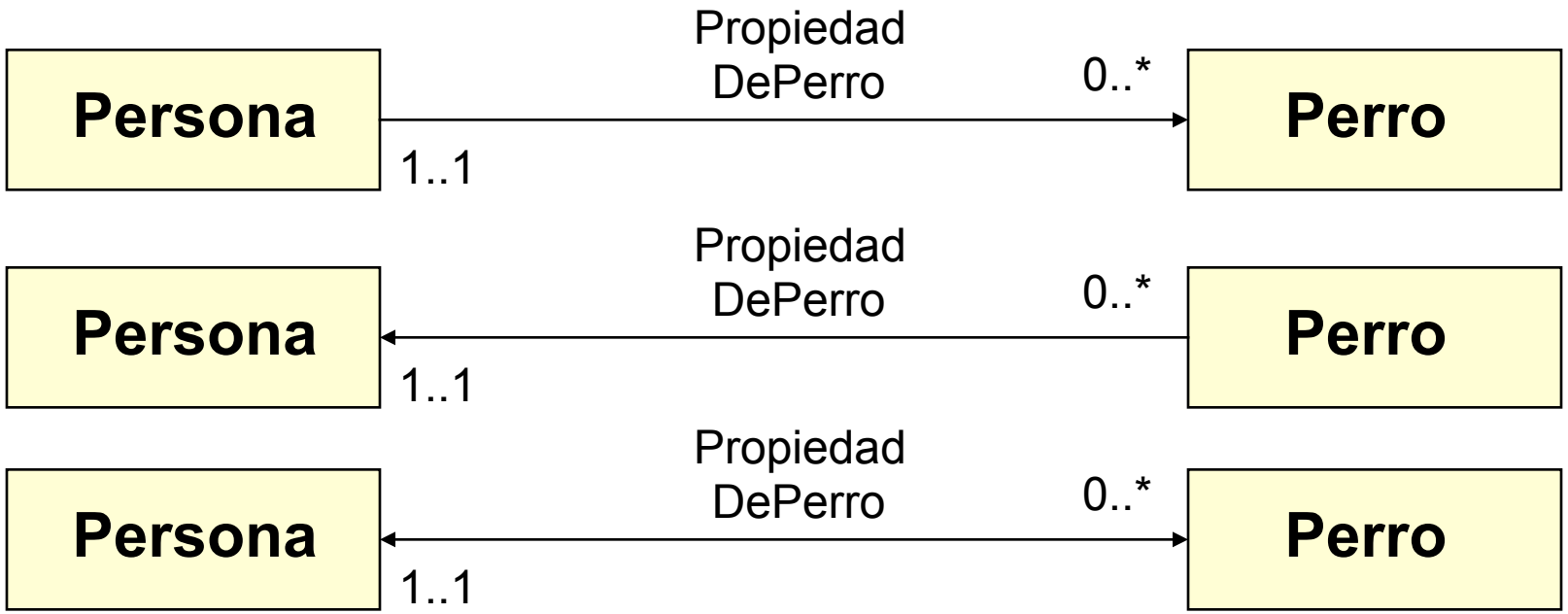
Asociaciones representadas como clases



Asociaciones como clases (II)

- La asociación PropiedadDePerro se ha promovido a una nueva clase, conectada a la asociación con una línea punteada.
- Promover una asociación a clase permite anexarle atributos y operaciones propias.
- En el ejemplo, la clase PropiedadDePerro, puede registrar la fecha en que un perro fue adquirido por una cierta persona (atributo fechaDeAdquisicion).

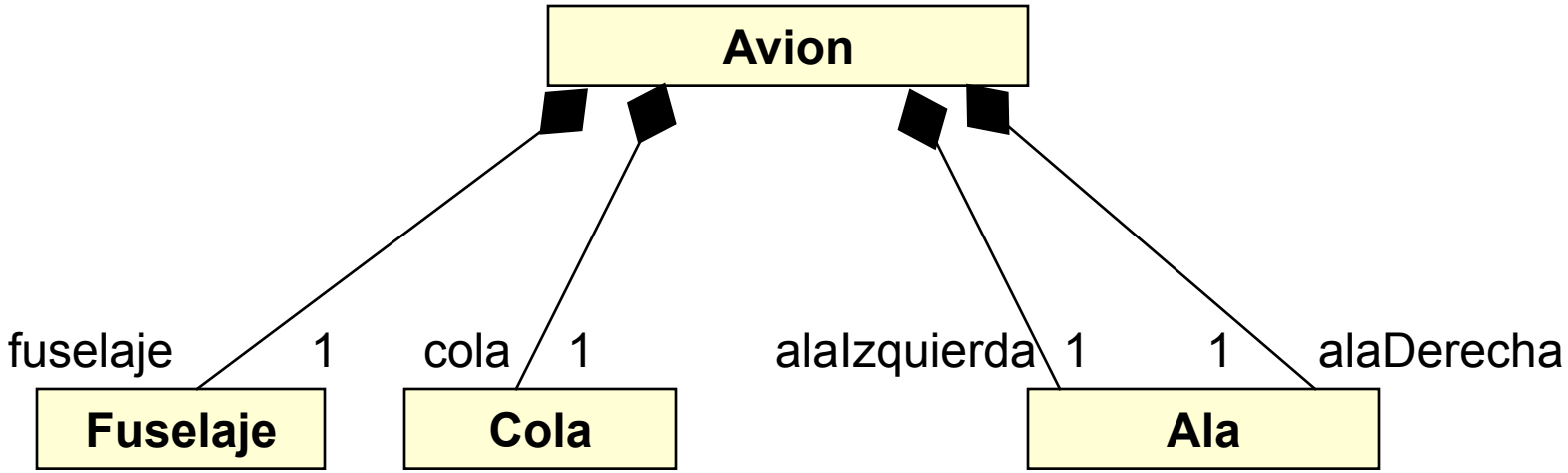
Navegabilidad de asociaciones



- Las asociaciones con cabeza de flecha muestran que existe un "link" directo desde un objeto de una clase al otro, lo que permite un acceso rápido.

Composición

- Permite expresar que un objeto se compone de otros objetos. Por ejemplo, un Avión se compone de un Fuselaje, una Cola y dos Alas (una a cada lado).



Composición (II)

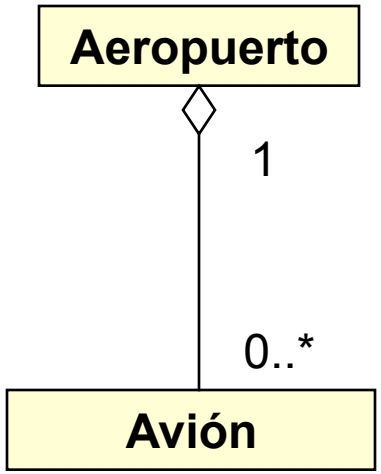
- La asociación entre el objeto compuesto y sus constituyentes se denota con un una línea con diamante relleno en el extremo del objeto compuesto.
- El rol del constituyente aparece en el extremo del constituyente de la asociación (un objeto constituyente puede jugar más de un rol).
- Debe mostrarse la multiplicidad en el extremo del constituyente de la asociación.

Composición (III)

- El objeto compuesto no existe sin sus componentes.
- Un objeto constituyente puede formar parte de solo un objeto compuesto a la vez.
- La composición suele ser heterogénea: los componentes suelen ser de distintas clases (cola, fuselaje, etc.).

Agregación

- Permite expresar que un objeto agrupa a otros objetos. Por ejemplo, un Aeropuerto contiene al conjunto de Aviones que en su loza se encuentran.



Agregación (II)

- La asociación entre el agregado y sus constituyentes se denota con un una línea con diamante abierto (no relleno) en el extremo del agregado.
- El rol del constituyente aparece en el extremo del constituyente de la asociación.
- Debe mostrarse la multiplicidad en ambos extremos de la asociación.

Agregación (III)

- El objeto agregado puede existir potencialmente sin sus objetos constituyentes.
- Un objeto constituyente puede ser parte de más de uno agregado.
- La agregación tiende a ser homogénea: los objetos constituyentes son de la misma clase.

Creación

- Se pude representar la idea que una clase es creada por otra utilizando la etiqueta <<create>>: