Orientación a Objetos I - 2021

Lenguaje de Modelado UML

Diagrama de Clases

Prof. Roxana Giandini

LIFIA - Facultad de Informática - UNLP

Diagrama de Clases

Vista Estática de un Sistema O.O.

Diagrama de Clases

Definición:

Un diagrama de clases muestra las clases del sistema y sus relaciones. Describe la vista estática de un sistema.

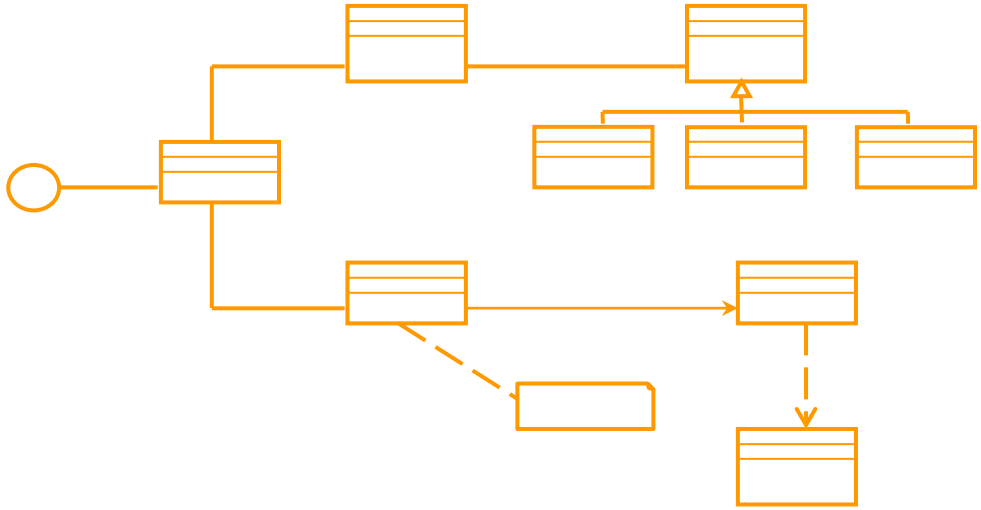
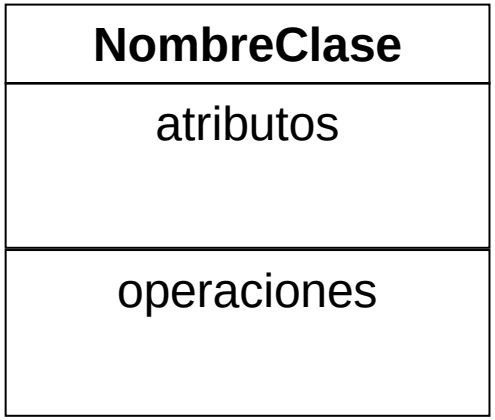


Diagrama de Clases: clase

- **¿Qué es una clase?**

  Una clase es una descripción de un conjunto de *objetos* que comparten los mismos *atributos* (estado interno de los objetos de la clase), *operaciones* (mensajes que responden los objetos), relaciones y semántica.

- **Estructura:**



Clase: atributo

Sintaxis

visibilidad **nombre** *[multiplicidad]:***tipo** *=valor inicial {propiedades}*

- Algunas partes son opcionales.
- Vamos a especificar nombre, tipo y visibilidad para los atributos
- **Visibilidad:** sirve para limitar la visibilidad del atributo a los objetos externos a la clase a la que pertenece.
  - Pública (+) : el atributo puede ser visto y usado fuera de la clase.
  - Privada (-) : el atributo no puede ser accedido por otras clases.
  - Protegida (#): es privado pero visible a las subclases de la clase donde está declarado.

NOTA: Si bien UML lo permite, ***no diseñaremos*** *clases con* ***atributos públicos*** *ya que esto contradice al concepto de encapsulamiento de los objetos.*

Clase: atributo – tipos primitivos de UML

Sintaxis

visibilidad **nombre** *[multiplicidad]*:**tipo** =*valor inicial {propiedades}*

- Algunas partes son opcionales.
- Vamos a especificar nombre, tipo y visibilidad para los atributos
- **Tipos primitivos**

Los tipos primitivos definidos en UML son:

URL del documento de especificación de UML
https://www.omg.org/spec/UML/2.5.1

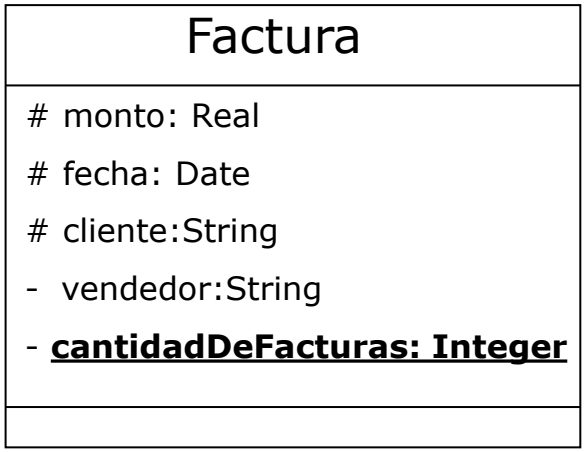
Clase: atributo

Sintaxis

Atributo de Clase:

como variable de clase. Va subrayado y significa que el valor de este atributo es el mismo para todos los objetos de la clase, ya que lo comparten.

Por ejemplo, el atributo cantidadDeFacturas es usado para contabilizar internamente las facturas en un comercio.



Clase: operación

- **Sintaxis**

visibilidad nombre([*lista de parámetros*]) :tipo de retorno {*propiedades*}

  - Algunas partes son opcionales. Vamos a especificar nombre, tipo y visibilidad para las operaciones.
  - Los parámetros se separan por comas.
  - Si la operación no tiene parámetros, igualmente van los () al final del nombre.
  - El tipo de retorno sólo va cuando la operación devuelve un valor. Si no retorna nada, no se especifica nada.

- Declaración de un parámetro:

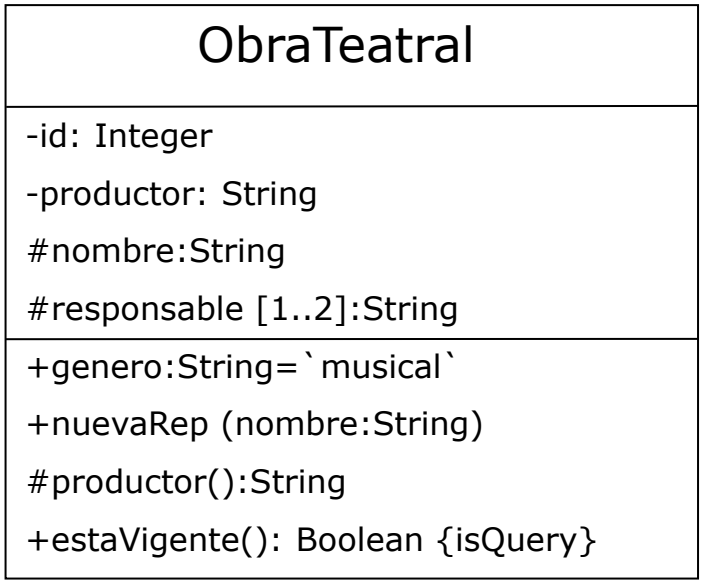
  nombre :tipo [*multiplicidad*] =valor

Clase: operación

Propiedades

- isQuery
- --

Ejemplo

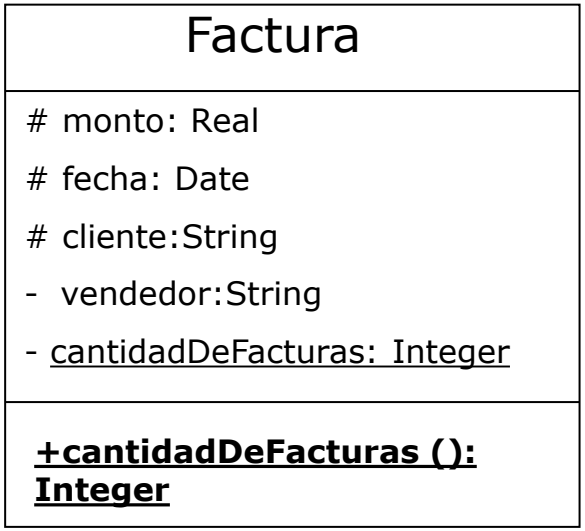


Clase: operación

Operaciones de Clase

- Mensajes de Clase, el receptor es la clase. Van subrayadas, como los atributos de clase

Ejemplo



Clase: operación

Constructores

- Mensajes de creación de objetos o instancias de la clase. Van precedidas por el estereotipo o etiqueta **<<Create>>**

Ejemplo

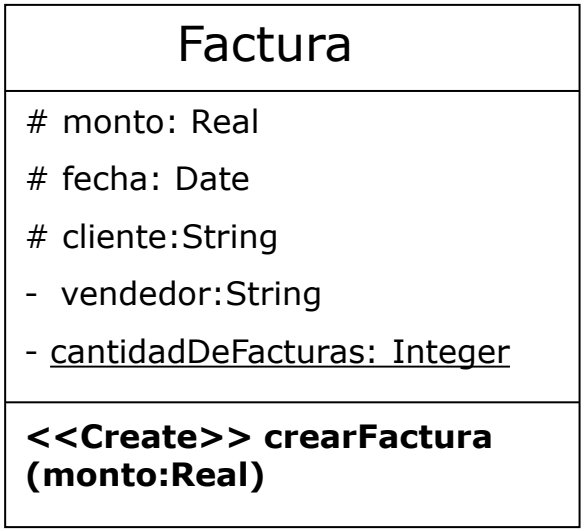


Diagrama de clases: relaciones

¿Qué son las relaciones entre clases?

Una relación es una conexión semántica entre objetos. Proveen un camino de comunicación entre ellos.

¿Qué tipos de relaciones usaremos?

- **Asociación** (relación de conocimiento)
- **Generalización** (relación de herencia)
- **Interfaz** (relación entre una clase interfaz y otra que la implementa)

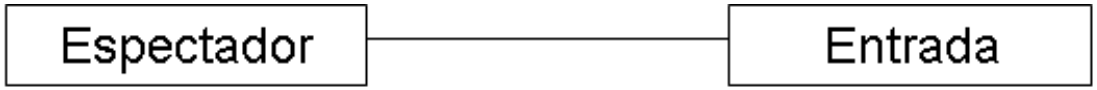
Relaciones: Asociación

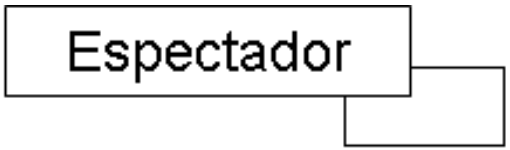
¿Qué es una Asociación?

Es una relación estructural que especifica que los objetos de un elemento (clase, caso de uso, etc.) están conectados con los objetos de otro.

Si no ponemos multiplicidad en los extremos, asume 1.

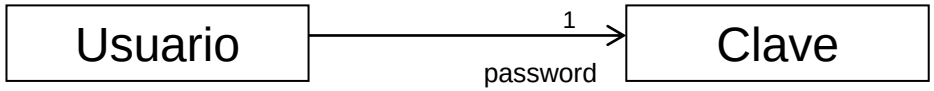
Ejemplo



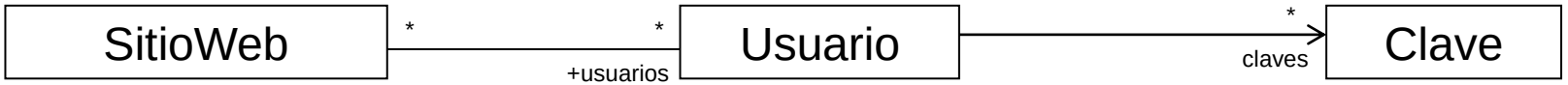


Asociación: propiedades

- **Rol:** son las caras que presentan las clases a las demás.
- **Navegabilidad**: sirve para limitar la navegación a una sola dirección.

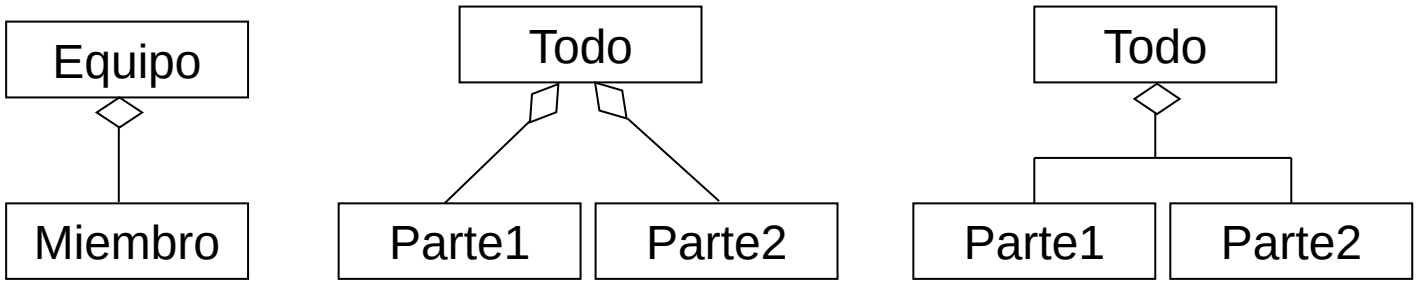


- **Visibilidad:** limita el alcance de los objetos que se relacionan. Similar al alcance de atributos.
  - Pública (+)
  - Protegida (#)
  - Privada (-)

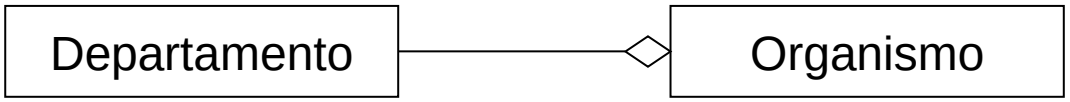


Asociación: propiedades (cont.)

Agregación: es una asociación especial, una relación del tipo "todo/parte" dentro de la cual una o más clases son partes de un todo.



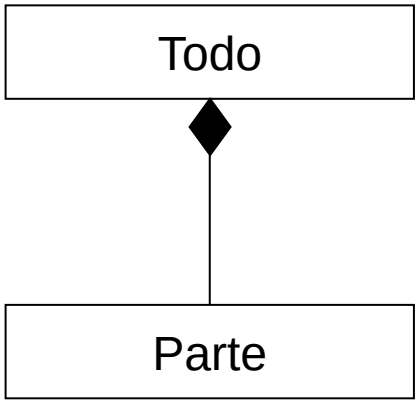
Ejemplo:



Asociación: propiedades (cont.)

Composición: es una forma de agregación, con fuerte sentido de posesión y tiempo de vida coincidentes de las partes con el conjunto.

- Una parte puede pertenecer solamente a una composición (un *todo*).
- Cuando el todo desaparece, también lo hacen sus partes.

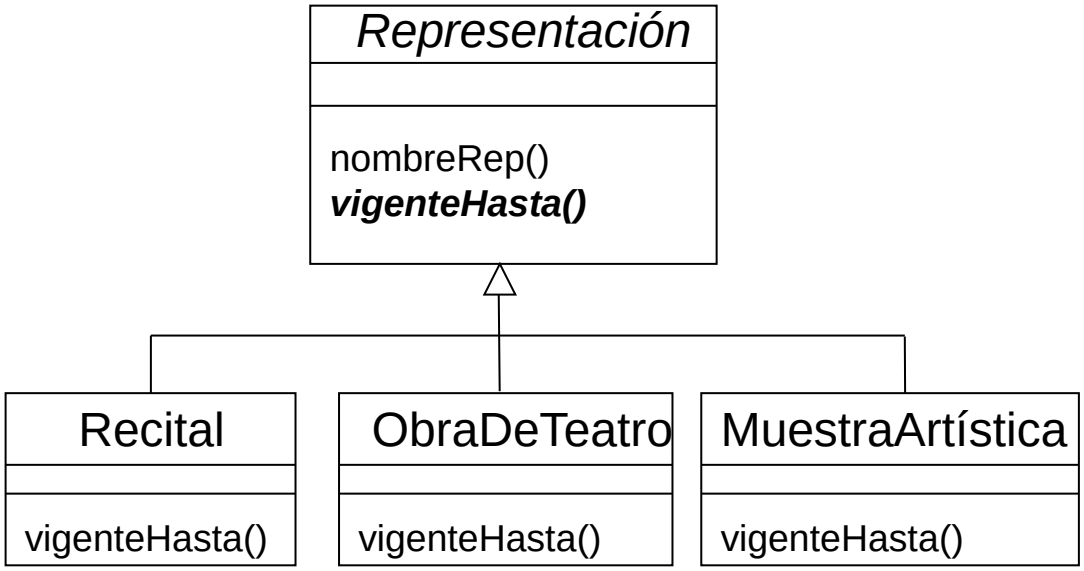


Relaciones: Generalización

¿Qué es una Generalización?

Es una relación de herencia entre un elemento general (llamado superclase) y un caso más específico de ese elemento (llamado subclase).

Ejemplo



- En UML puede representarse herencia múltiple
- La clase abstracta (que no puede instanciarse por tener operaciones abstractas) va en cursiva o bien con <<abstract>> acompañando al nombre

Diagrama de clases - Ejemplo

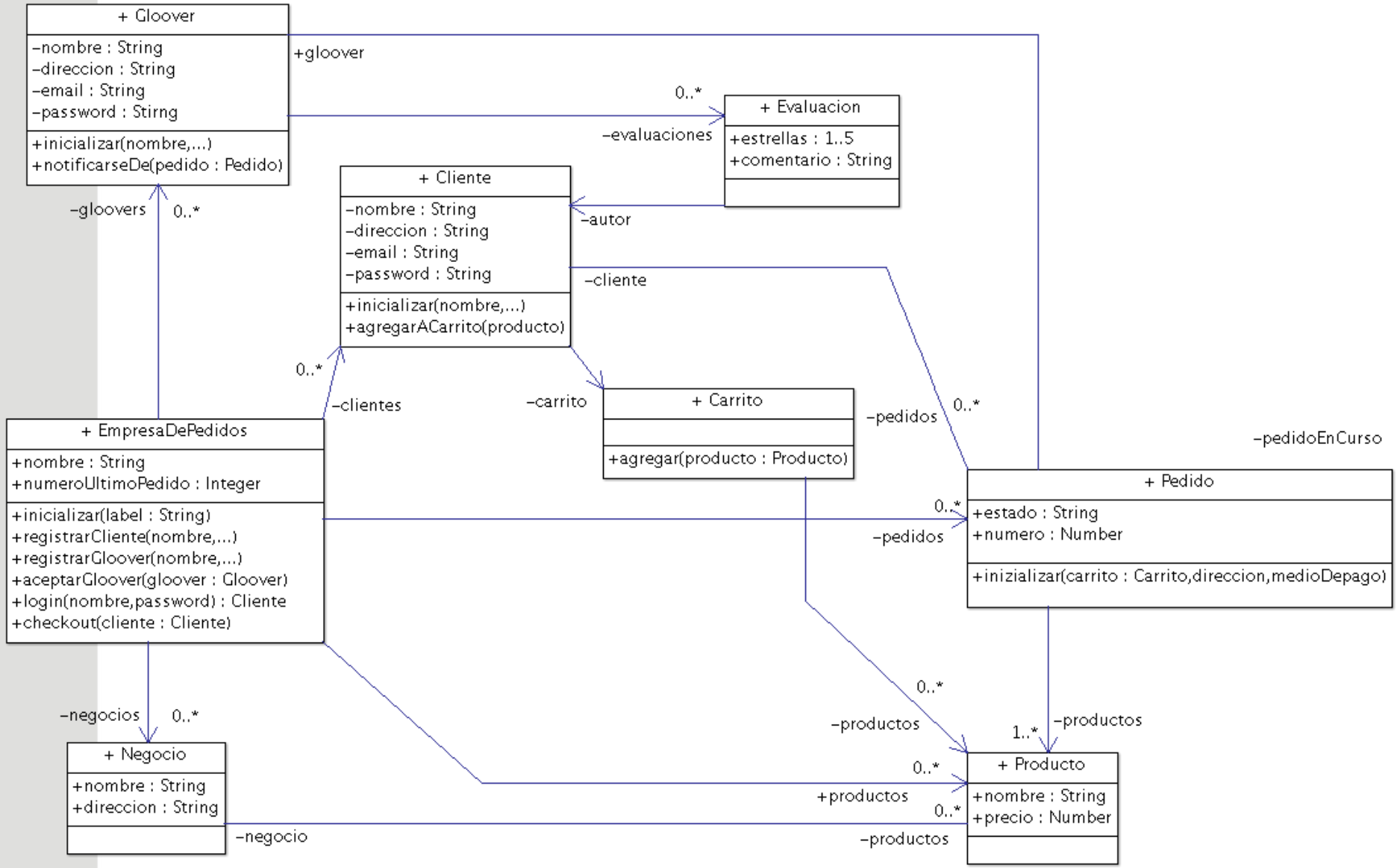


Diagrama de clases - Interfaces

Una clase Interfaz especifica una parte, visible externamente, del comportamiento de otras clases.

En UML se especifica con el estereotipo **<<Interface>>**.

Es abstracta ya que otra/s clase/s implementan sus métodos.

La clase que implementa la interfaz, implementa **todos** los métodos de la interfaz.

La relación entre la clase que la implementa y la Interfaz, se representa con línea punteada.

Diagrama de clases - Interfaces