FICH UNL • FACULTAD DE INGENIERÍA Y CIENCIAS HÍDRICAS

Tecnologías de Programación

Paradigma Orientado a Objetos

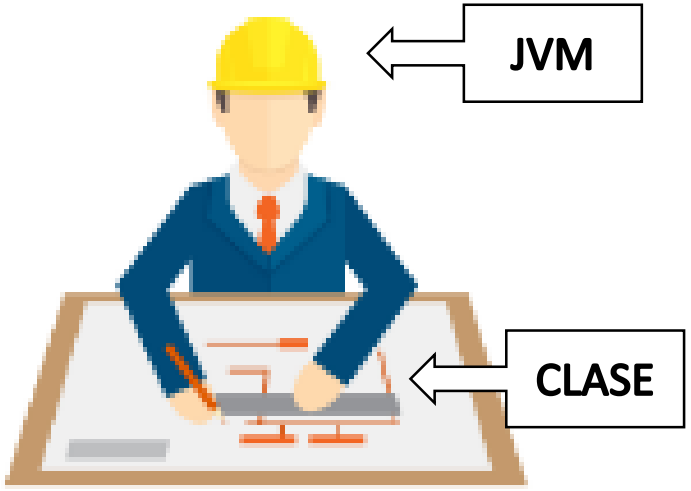
1 - Clases

“The UML Reference Manual” define a la clase como el descriptor para un conjunto de objetos que comparten los mismos atributos, operaciones, métodos, relaciones y comportamiento.

- Algunas clases tienen una contrapartida real (persona y empresa por ejemplo).
- Otras son entidades conceptuales (ecuación algebraica).
- Además, existen clases que son sólo artefactos de una implementación específica (por ejemplo, árbol binario).

1 - Clases

Podemos pensar en una clase como un molde o plantilla para un objeto. Le dice a la máquina virtual cómo crear un objeto de ese tipo en particular. Cada objeto creado a partir de esa clase puede tener sus propios valores para las variables de instancia de esa clase.



Perro
nombre raza tamaño
ladrar()

UNA CLASE

MUCHOS PERROS

2– Diagrama de Clases

- Las clases y sus relaciones se describen mediante un diagrama de clases.

- La relación entre una clase y objetos de esa clase es una relación <<instantiate>>

- Entonces: un diagrama de clases describe un conjunto de diagramas de instancias y permite representar de forma abstracta el conjunto de diagramas de instancias documentando la estructura de los datos

3 – Clases e instancias

- Un objeto se crea por la sentencia:

```
//crear un nuevo objeto
NombreClase oClase = new NombreClase();
```

Es una instancia de la clase NombreClase, y que en realidad es un objeto. Utilizamos la palabra instancia para resaltar la relación entre los objetos y las clases

Caja objeto es instancia de una clase exactamente.

Las instancias de una clase entienden y tienen la capacidad de responder los mensajes para los cuales hay métodos definidos en la clase

3 – Relaciones entre clases.

- De acuerdo con G. Booch existen tres clases básicas de relaciones:

  - Asociación (conexión entre clases)
  - Agregación/Composición (relaciones de pertenencia)
  - Generalización/especialización (relaciones de herencia)

3.1 – Asociación

- Es una relación entre clases. Diremos que dos (o más) clases tiene una relación de asociación cuando una de ellas tenga que requerir o utilizar alguno de los servicios (es decir, acceder a alguna de las propiedades o métodos) de las otras. Se da cuando una clase usa a otra clase para realizar algo.

- Las relaciones de asociación crean enlaces entre objetos. Estos enlaces no tienen por qué ser permanentes (en la mayoría de los casos, no lo son). Los objetos deben tener entidad fuera de la relación (a diferencia de las relaciones de composición).

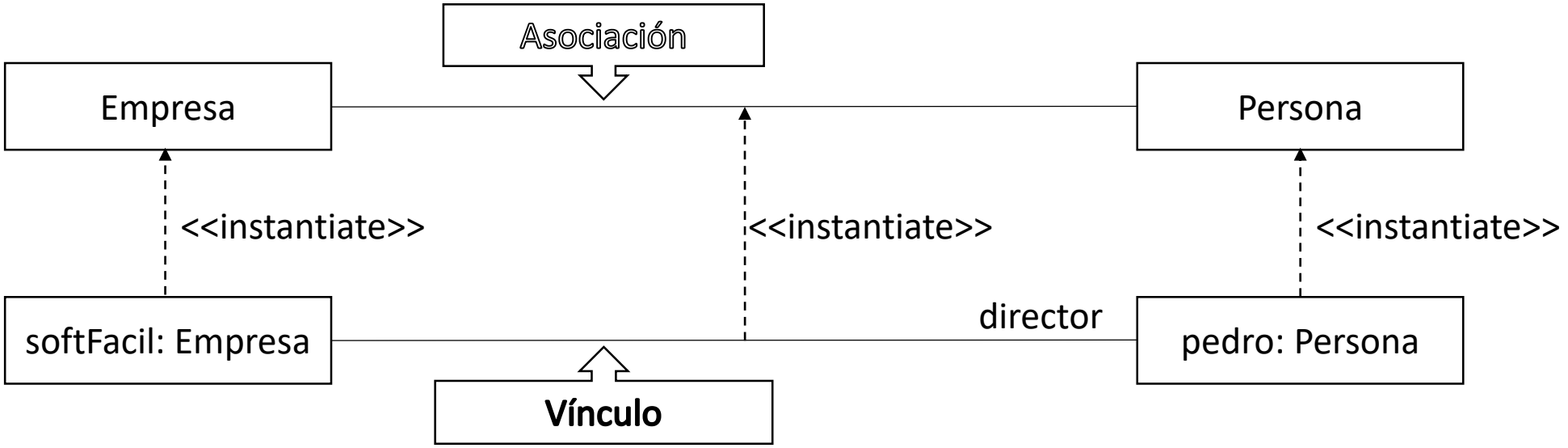
- Esta relación permite asociar objetos que colaboran entre sí. Cabe destacar que no es una relación fuerte, es decir, el tiempo de vida de un objeto no depende del otro.

- Para validar la asociación, la frase “Usa un”, debe tener sentido, por ejemplo:

  El médico *usa* **un estetoscopio**
  El **cliente** *usa* **tarjeta de crédito**.

3.1 – Asociación

El punto importante es que para que exista un vínculo entre dos objetos debe haber una asociación entre las clases de esos objetos. Para hacer explícita la semántica de la dependencia entre asociaciones. Arlow (2005)

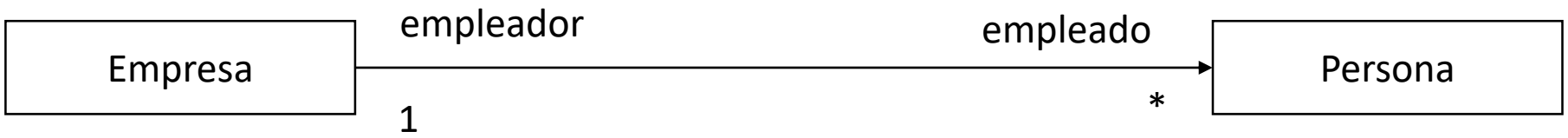


La semántica de la asociación básica es muy sencilla , una asociación entre clases indica que puede tener vínculos entre objetos de esas clases.

3.1 – Asociación - Sintaxis

Las asociaciones pueden tener:

- Un nombre: debería ser una frase verbal porque indica una acción que el objeto fuente está realizando sobre el objeto destino.
- Nombres de roles: en uno o ambos extremos de la asociación. Indican los roles que los objetos de esas clases desempeñan cuando están vinculando por instancias de esta asociación
- Multiplicidad: restringe el número de objetos de una clase que se pueden implicar en una relación determinada **en cualquier momento en el tiempo**.
- Navegabilidad: muestra que es posible pasar desde un objeto de la clase fuente a uno o más objetos de la clase destino dependiendo de la multiplicidad. Uno de los objetivos es minimizar el acoplamiento entre clases, utilizar la navegabilidad es una buena forma de hacerlo.

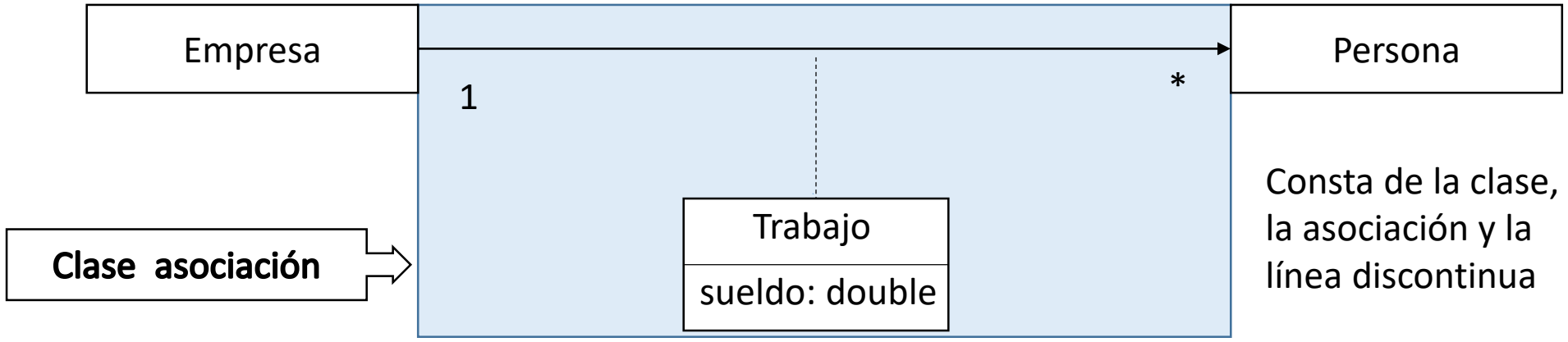


3.1 – Clase Asociación

- Una clase asociación es una asociación que es también una clase. No solo conecta dos clases como una operación, sino que define un conjunto de características que pertenecen a la propia asociación. Las clases asociación pueden tener atributos, operaciones y otras asociaciones.

- ¿Qué problema común en el modelado orientado a objetos resuelve?

  Cuando se tiene una relación muchos a muchos entre dos clases y existen atributos que no se pueden acomodar fácilmente en ninguna clase.



3.2 – Agregación/Composición

UNL • FACULTAD DE INGENIERÍA Y CIENCIAS HÍDRICAS

- Esta relación es utilizada cuando una clase se compone de otras clases, es un tipo de asociación que indica que una clase es parte de otra clase.
- Se presenta entre una clase TODO y una clase PARTE que es componente de TODO. La implementación de este tipo de relación se consigue definiendo como atributo un objeto de la otra clase que es **parte-de.**
- Los objetos de la clase TODO son objetos contenedores. Un objeto contenedor es aquel que contiene otros objetos
- Es un tipo de relación **dependiente** en donde un objeto más complejo es conformado por objetos más pequeños. En esta situación, la frase “Tiene un”, debe tener sentido:
  - **La bicicleta** tiene **ruedas**
  - **La pc** tiene **teclado**

3.2 – Agregación/Composición

Existen dos tipos de especialización de esta relación:

Agregación: implica una composición débil, si una clase se compone de otras y quitamos alguna de ellas, entonces la primera seguirá funcionando normalmente, por ejemplo un objeto de tipo Agenda tiene una lista de objetos de tipo Contacto, si quitamos algún objeto de tipo Contacto no afectamos la estructura básica del objeto de tipo Agenda.

3.2 – Agregación/Composición

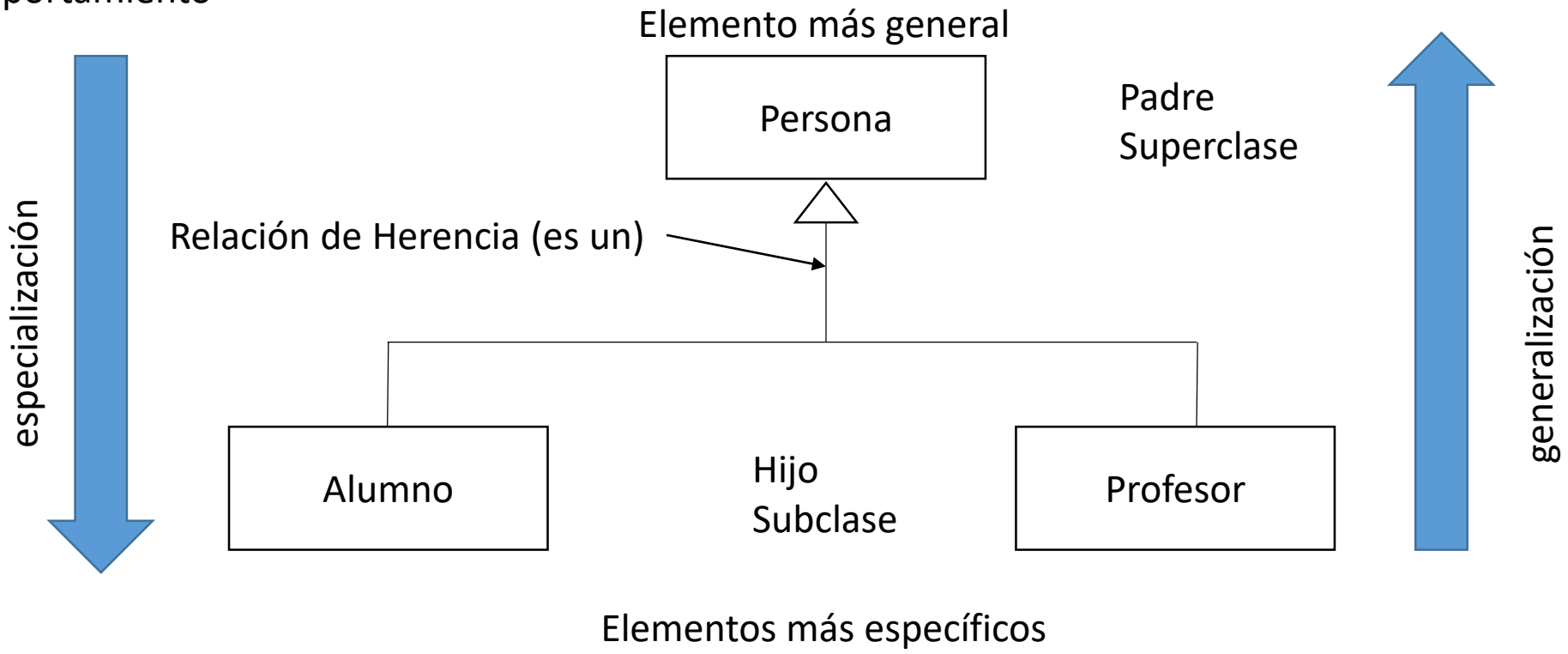
Existen dos tipos de especialización de esta relación:

Composición: La Composición es una forma fuerte de composición, donde la vida de la clase contenida debe coincidir con la vida de la clase contenedor. Los componentes constituyen una parte del objeto compuesto.

El tiempo de vida de un objeto está condicionado por el tiempo de vida del objeto que lo incluye, por ejemplo si tenemos una objeto de tipo Mesa, que está compuesto de cuatro objetos de tipo Pata, si eliminamos un objeto de tipo Pata, quedará con tres, lo cual rompe la definición original que tenía el tipo Mesa, la mesa no puede existir sin sus patas.

3.3 – Generalización/Especialización

De todas las relaciones posibles entre las distintas clases y objetos, hay que destacar por su importancia en O.O la relación de **herencia**, ésta es una relación entre clases que comparten su estructura y el comportamiento



4 – Reutilización

Existen dos mecanismos para construir clases utilizando otras clases.

- **Composición:** Una clase posee objetos de otras clases (relación *tiene un*). Se puede reutilizar los atributos y métodos de otras clases, a través de la invocación de los mensajes correspondientes.

- **Herencia:** Se pueden crear clases nuevas a partir de clases preexistentes (relación *es un*). Se puede reutilizar los atributos y métodos de otras clases como si fueran propios.

5 – Herencia

La ***herencia*** es un mecanismo que permite la definición de una clase a partir de la definición de otra ya existente. Es la característica clave de los sistemas orientados a objeto para propiciar entre otros aspectos la reusabilidad.

En general el concepto de herencia, se refiere al mecanismo por el cual los objetos comparten comportamiento.

Mediante una clase (molde, plantilla) es posible definir la estructura y el comportamiento de un conjunto de objetos, en donde se detallan los atributos y métodos que compartirán todas las instancias de la misma determinando su estado y comportamiento.

Las clases se organizan jerárquicamente, y una nueva clase puede reutilizar la estructura y comportamiento de otras previamente definidas.

5 – Herencia

La ***herencia*** es un mecanismo que permite la definición de una clase a partir de la definición de otra ya existente. Es la característica clave de los sistemas orientados a objeto para propiciar entre otros aspectos la reusabilidad.

En general el concepto de herencia, se refiere al mecanismo por el cual los objetos comparten comportamiento.

Mediante una clase (molde, plantilla) es posible definir la estructura y el comportamiento de un conjunto de objetos, en donde se detallan los atributos y métodos que compartirán todas las instancias de la misma determinando su estado y comportamiento.

Las clases se organizan jerárquicamente, y una nueva clase puede reutilizar la estructura y comportamiento de otras previamente definidas.

6 – Polimorfismo

Polimorfismo es invocar métodos distintos con el mismo mensaje (ligadura en tiempo de ejecución). Para ello es necesaria una jerarquía de herencia: una clase base que contenga un método polimórfico, que es redefinido en las clases derivadas (no anulado).

Se permite que los métodos de los hijos puedan ser invocados mediante un mensaje que se envía al padre. Este tipo de clase que se usa para implementar el polimorfismo se conoce como ***clase abstracta***.

Un conjunto de operaciones abstractas es una forma de definir un conjunto de operaciones que todas las subclases concretas deben implementar. Esto se conoce como contrato

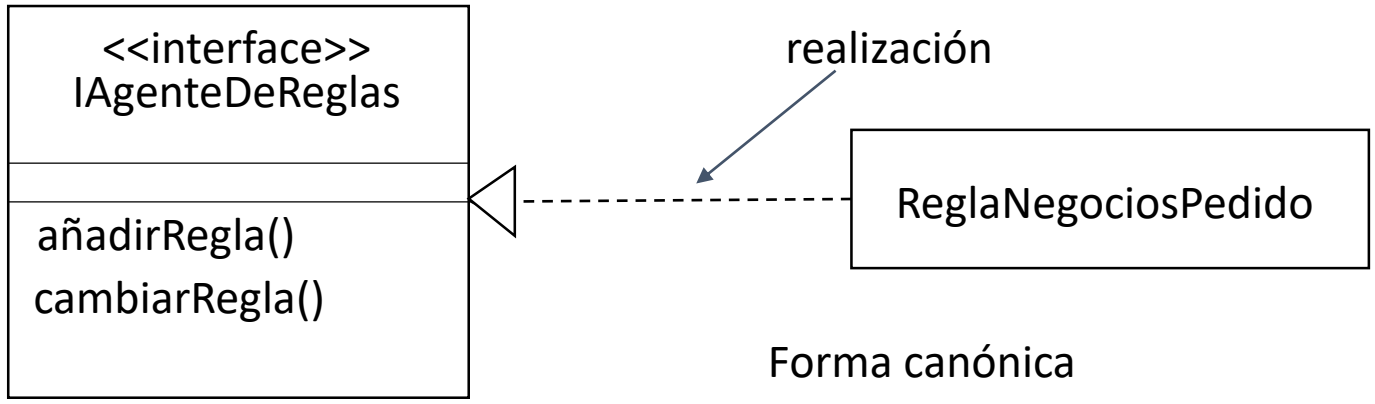
El polimorfismo permite diseñar sistemas más sencillos que se pueden acomodar más fácilmente el cambio porque le permite tratar objetos diferentes de la misma forma.

7 – Interfaces

En UML las interfaces se emplean para modelar las líneas de separación de un sistema. Una interfaz es una colección de operaciones (métodos) que sirven para especificar un servicio de una clase o un componente. Al declarar una interfaz se puede enunciar el comportamiento deseado de una abstracción independientemente de una implementación de ella

Una clase puede realizar muchas interfaces.

Permiten separar la especificación de un contrato (la propia interfaz) de su implementación (por una clase o un componente)



Forma canónica

8 – Realización

La realización es lo suficientemente diferente de la dependencia, la generalización y la asociación como para tratarla como un tipo aparte de relación. Semánticamente la realización es algo así como una mezcla entre dependencia y generalización, y su notación es una combinación de la notación de la dependencia y generalización.

La realización se utilizará bajo dos circunstancias:

- En el contexto de las interfaces
- En el contexto de las colaboraciones