

## Unidad 2

# El Diagrama de Clases



**UAIOnline**  
**Ultra**»»



# El Diagrama de Clases

## Unidad 2

### ■ OBJETIVOS

- Comprender las características de una Clase.
- Diferenciar los distintos tipos de asociaciones de Clases.
- Entender las diferencias entre el paradigma estructurado y el OO.



# El Diagrama de Clases

## Unidad 2

- **HABILIDADES Y COMPETENCIAS QUE DESARROLLA LA ASIGNATURA**
- Adquirir conocimientos para representar la estructura de un sistema aplicando UML.
- Comprender las distintas concepciones acerca de las relaciones y dependencias de Clases.



# Introducción: Consideraciones Generales



**UAIOnline**  
**Ultra**»»

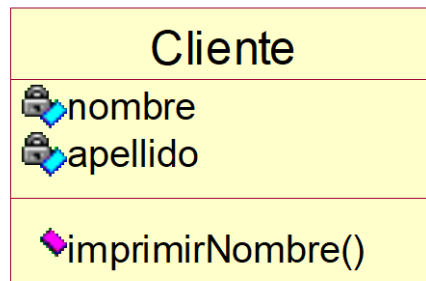


# CLASES

Las clases representan los bloques de construcción más importantes de cualquier sistema orientado a objetos.

Una clase es una descripción de un conjunto de objetos que comparten los mismos atributos, operaciones, relaciones y semántica.

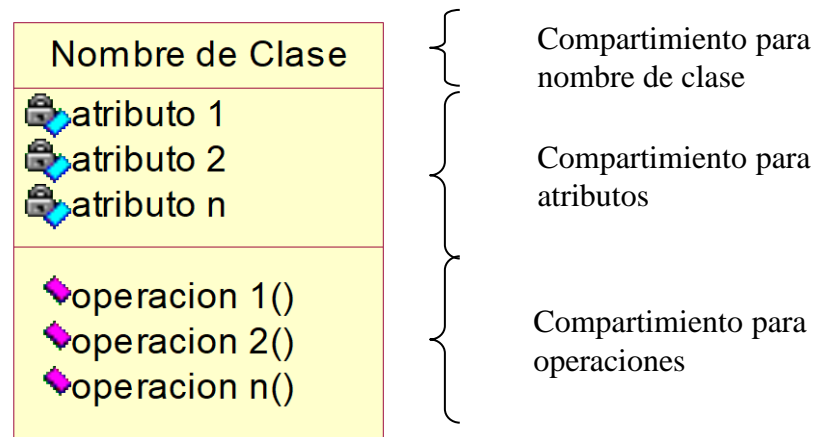
Los lenguajes de programación OO soportan directamente el concepto de clase



```
public class Cliente
{ private String nombre;
  private String apellido;
  public void imprimirNombre() { };
  ... }
```







# ATRIBUTOS Y OPERACIONES

- Un **atributo** es una propiedad de una clase que es compartida por todos los elementos de esa clase.
- Una clase puede tener cualquier número de atributos o no tener ninguno.
- Una **operación** es una abstracción de algo que se puede hacer a un objeto y que es compartido por todos los objetos de la clase.
- Una clase puede tener cualquier número de operaciones o ninguna.



# VISIBILIDAD DE ATRIBUTOS Y OPERACIONES

- Visibilidad pública (Public +): cualquier clase externa puede utilizar la característica.
- Visibilidad privada (Private -): solo la propia clase puede utilizar la característica.
- Visibilidad protegida (Protected #): cualquier descendiente de la clase puede utilizar la característica.
- Visibilidad de paquete (package ~) solo los que estén declarados dentro del paquete pueden utilizar la característica.

Nombre de Clase
 atributo 1
 atributo 2
 atributo n
 operacion 1()
 operacion 2()
 operacion n()

(-) *visibilidad privada*  
(#) *visibilidad protegida*  
(+) *visibilidad pública*  
(~) *visibilidad paquete*

# DESCRIPCIÓN DE ATRIBUTOS Y OPERACIONES

## Atributo

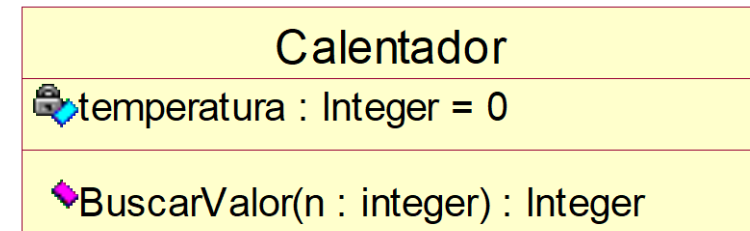
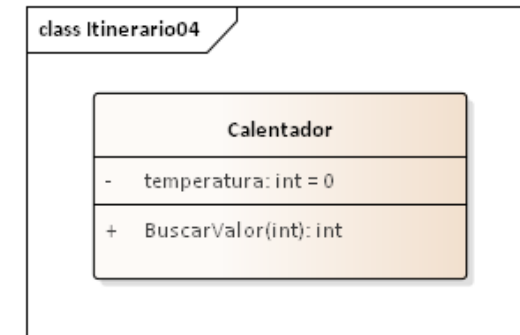
*Visibilidad [/] nombre : tipo = valor inicial*

ejemplo    - *temperatura : integer = 0*

## Operación

*Visibilidad nombre (lista parámetros) : tipo de expresión  
retornada*

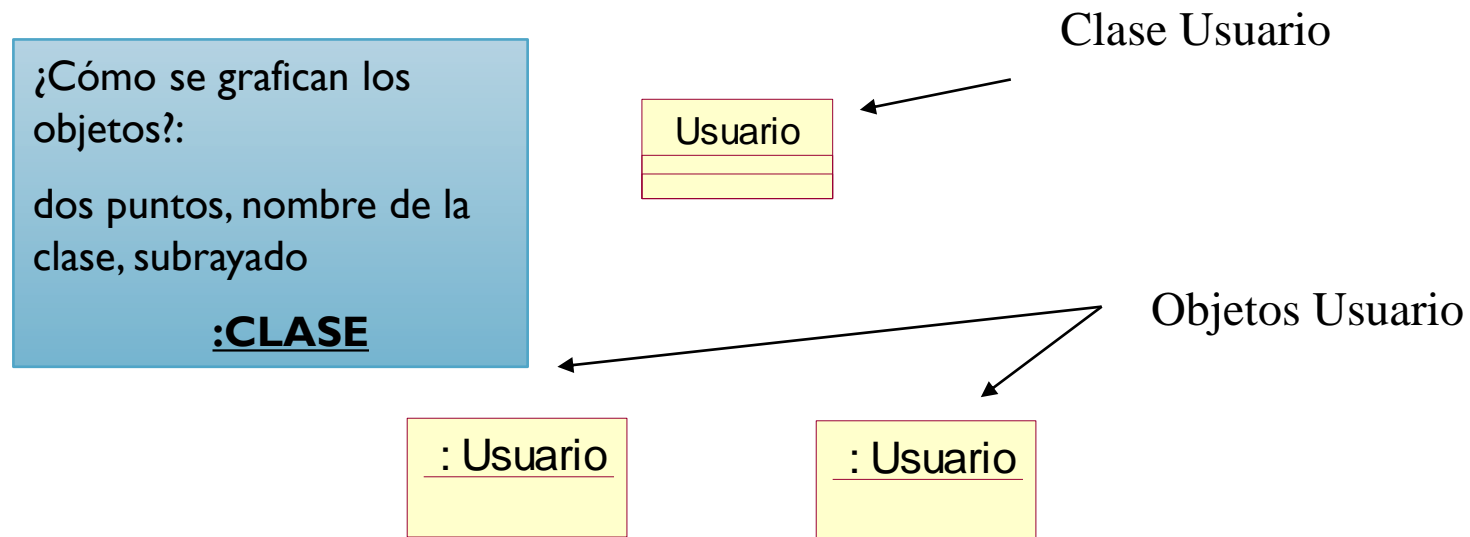
ejemplo    +*BuscarValor (n: integer) : integer*





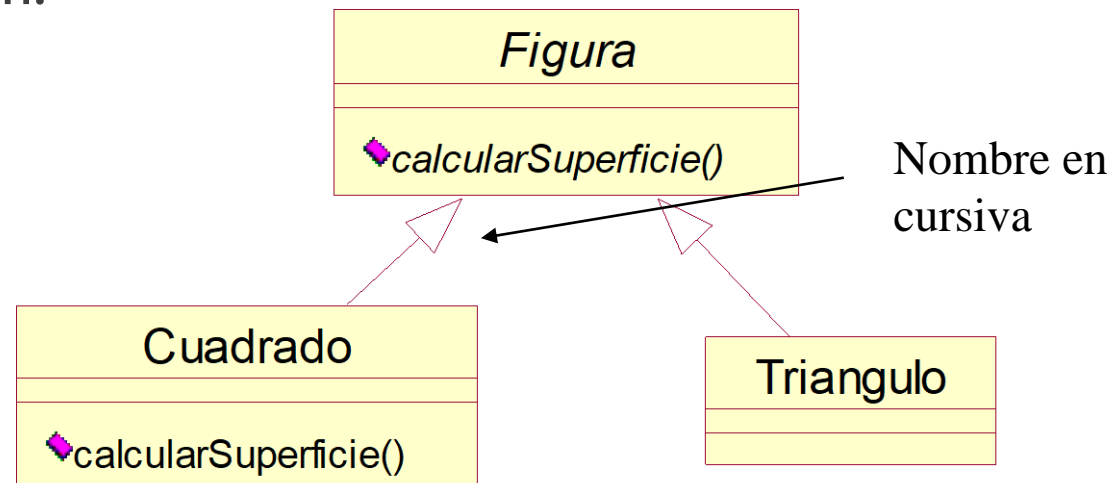
# INSTANCIAS DE CLASES: OBJETOS

- Una instancia es la manifestación concreta de una abstracción a la que se le pueden aplicar un conjunto de operaciones y posee un estado que almacena el efecto de las operaciones.



# CLASES ABSTRACTAS

- Son clases que no pueden ser instanciadas, se utilizan en jerarquías de generalización.
- Las clases abstractas tienen, al menos, una operación abstracta.
- Una operación abstracta tiene que ser implementada por algún método en un nivel más bajo de abstracción.



# INTERFAZ/ I

- Una interfaz es una colección de operaciones que especifican un servicio de una clase o un componente
- Especifica el comportamiento visible externamente de la clase
- Una interfaz define un conjunto de especificaciones de operaciones (su signatura) pero no su implementación
- La interfaz es parecida a la clase abstracta, ninguna de las dos pueden tener instancias directas, no obstante una clase abstracta puede tener operaciones concretas.
- Una interfaz es como una clase abstracta donde todas las operaciones también son abstractas

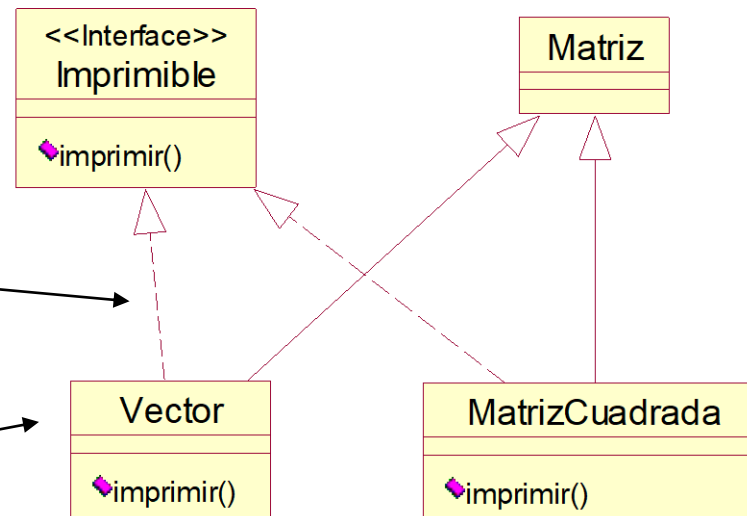
# INTERFAZ/2

- La realización es una relación entre clasificadores.
- La realización se emplea para especificar la relación que existe entre una interfaz y la clase que proporciona la operación.
- Permite “simular” herencia múltiple.

Defino la  
operación  
imprimir para  
muchas clases

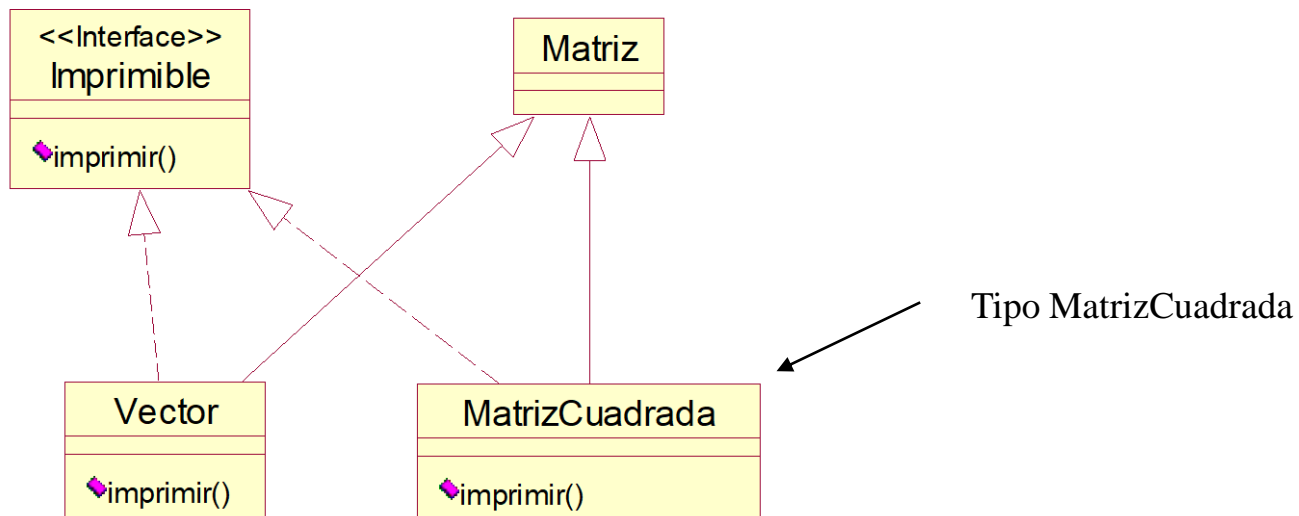
Realización

Cada clase la  
implementa  
como  
corresponde



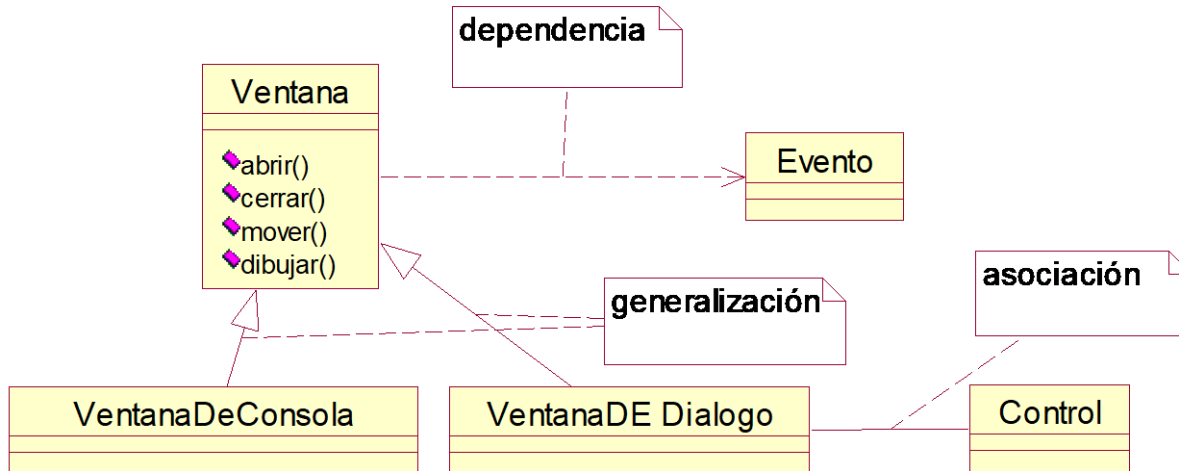
# INTERFAZ/3

- Cuando hablamos de un “**Tipo**” de objeto, nos referimos a su interfaz. Por ejemplo, si instanciamos un objeto del tipo MatrizCuadrada, estamos indicando cuales son los **mensajes** que podrá entender para luego invocar **operaciones** que serán implementadas con funciones y procedimientos,.
- Los atributos de las clases se hacen públicos por medio de **getters** y **setters**. Estos últimos se implementan con **funciones, procedimientos y propiedades** que serán parte de la interfaz de los objetos.



# RELACIONES/I

- Las clases no se encuentran aisladas, existen tres tipos principales de relaciones:
  - **Dependencias:** relaciones de uso entre clases
  - **Asociaciones:** relaciones estructurales entre clases
  - **Generalizaciones:** conectan clases generales con sus especializaciones



## RELACIONES/2

Asociación



Agregación



Composición



Generalización



Dependencia

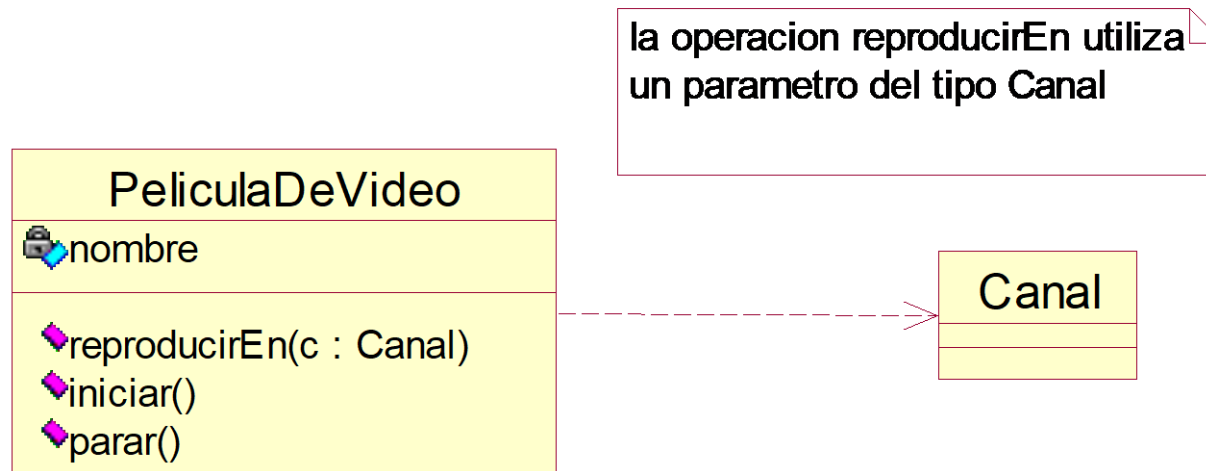


Realización



# DEPENDENCIAS

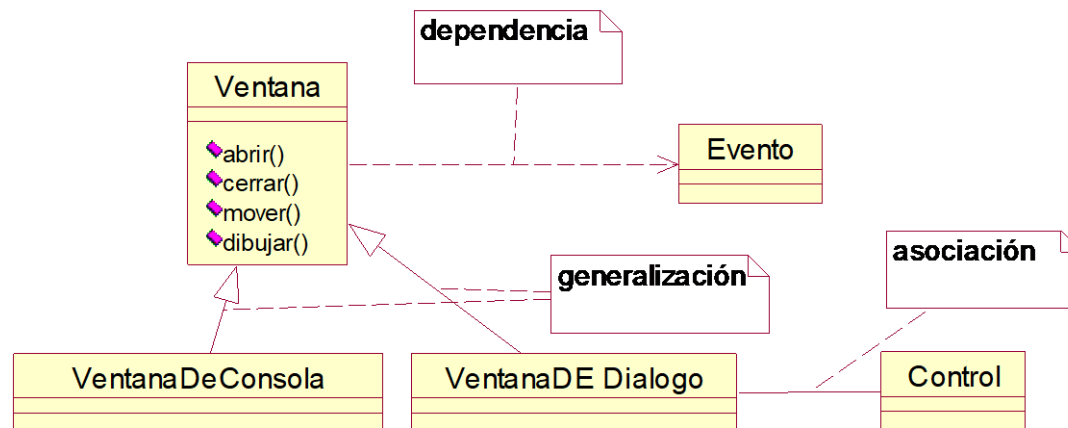
- Una dependencia es una relación de uso que declara que un elemento utiliza información de otro elemento
- En general se utilizan en el contexto de las clases para indicar que una clase utiliza las operaciones de otra o utiliza variables o parámetros cuyo tipo viene dado por la otra clase
- En los diagramas de clase, se utilizan para describir la visibilidad entre clases que no es de tipo atributo





# ASOCIACIONES

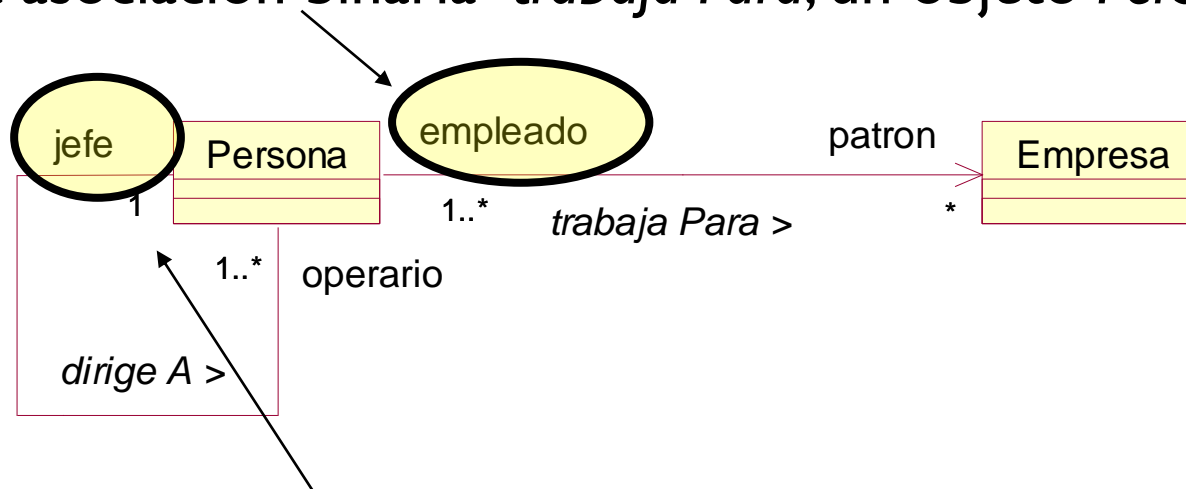
- Una asociación es una relación estructural que especifica que los objetos de una clase están conectados con objetos de otra (que puede ser la misma)
- Son relaciones entre clases que indica alguna conexión significativa que deseamos preservar durante algún tiempo.



## ASOCIACIONES – NOMBRE DE ROL

- Nombre de rol: cada objeto juega un rol específico en la asociación

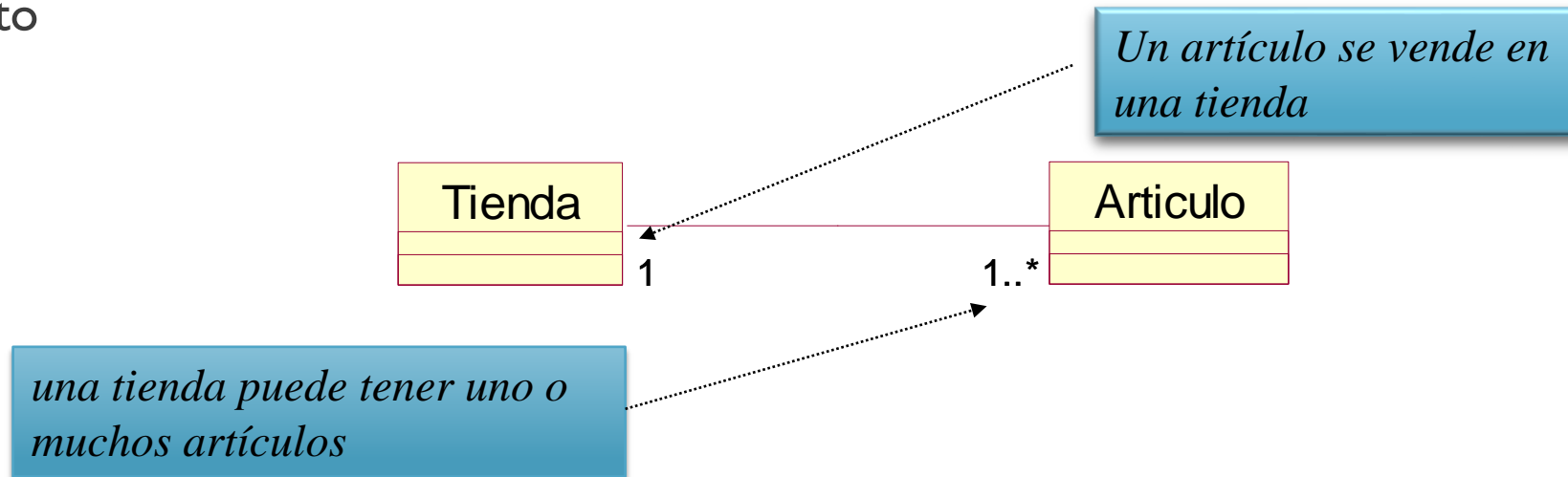
Por ejemplo, en la asociación binaria *trabaja Para*, un objeto *Persona*, juega el rol de empleado.



En la asociación unaria *dirige A*, un objeto *Persona* puede jugar el rol de operario o de jefe

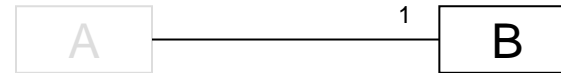
# ASOCIACIONES – MULTIPLICIDAD DE ROL/I

- Multiplicidad: define cuántas instancias de una clase A pueden asociarse con una instancia de una clase B
- Representa un rango de enteros que especifican el tamaño posible del conjunto de objetos relacionados (v gr.: 0..1; 1..1; 0..\*; 1..\*)
- El valor de la multiplicidad indica cuántas instancias se pueden asociar con otras en un momento concreto

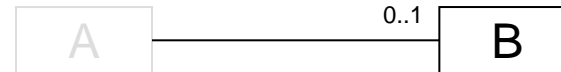


# ASOCIACIONES – MULTIPLICIDAD DE ROL/2

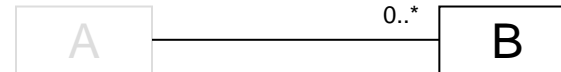
Exactamente  
Uno:



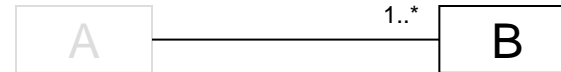
Cero o Uno:



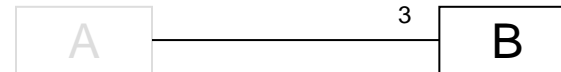
Cero o Muchos:



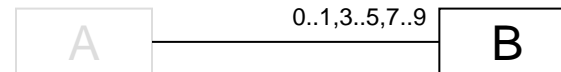
Uno o Más:



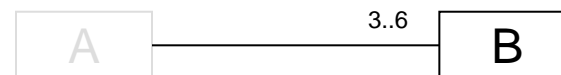
Número Exacto:



Lista:

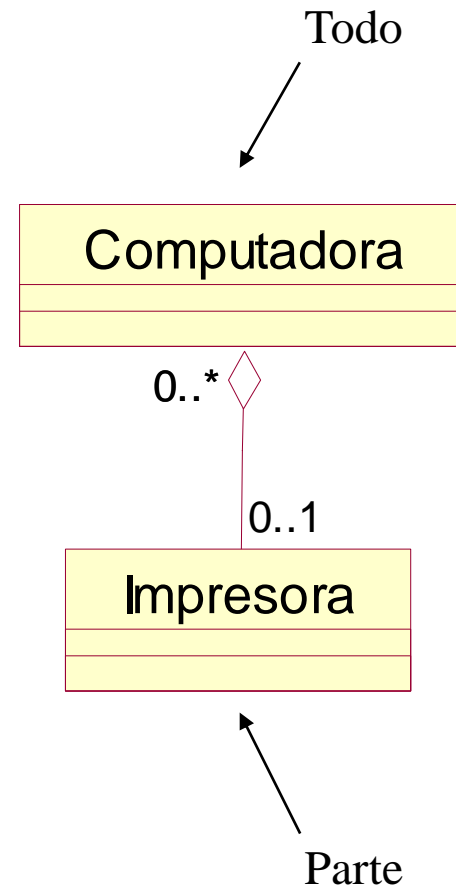


Rango:



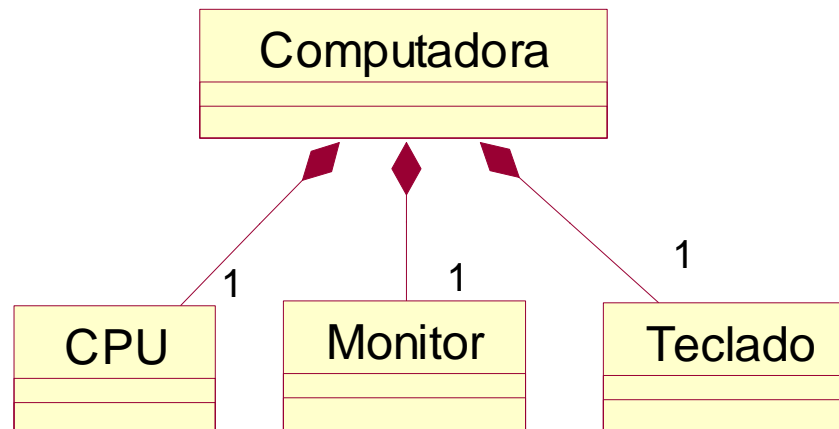
# ASOCIACIONES - AGREGACIÓN

- *Este tipo especial de asociación modela una relación TODO/PARTE*
- *Es un tipo de asociación más fuerte*
- *Es una relación no simétrica entre clases donde uno de los extremos cumple un rol dominante.*



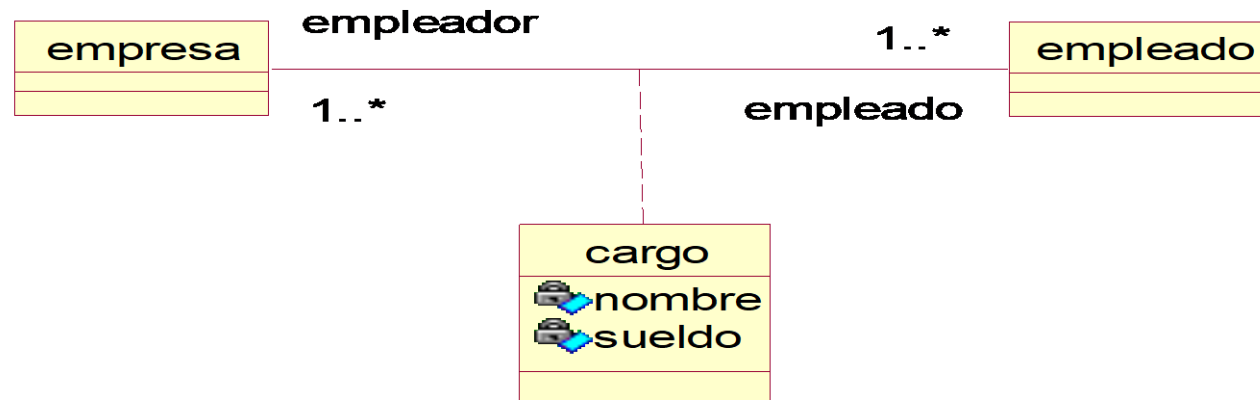
# ASOCIACIONES - COMPOSICIÓN

- *La composición es una forma de agregación con una fuerte relación de pertenencia y vidas coincidentes de la parte con el todo*
- *Dependencia existencial. El elemento dependiente desaparece al destruirse el que lo contiene.*
- *Hay una pertenencia fuerte. Se puede decir que el objeto contenido es parte constitutiva y vital del que lo contiene.*
- *Los objetos contenidos no son compartidos*



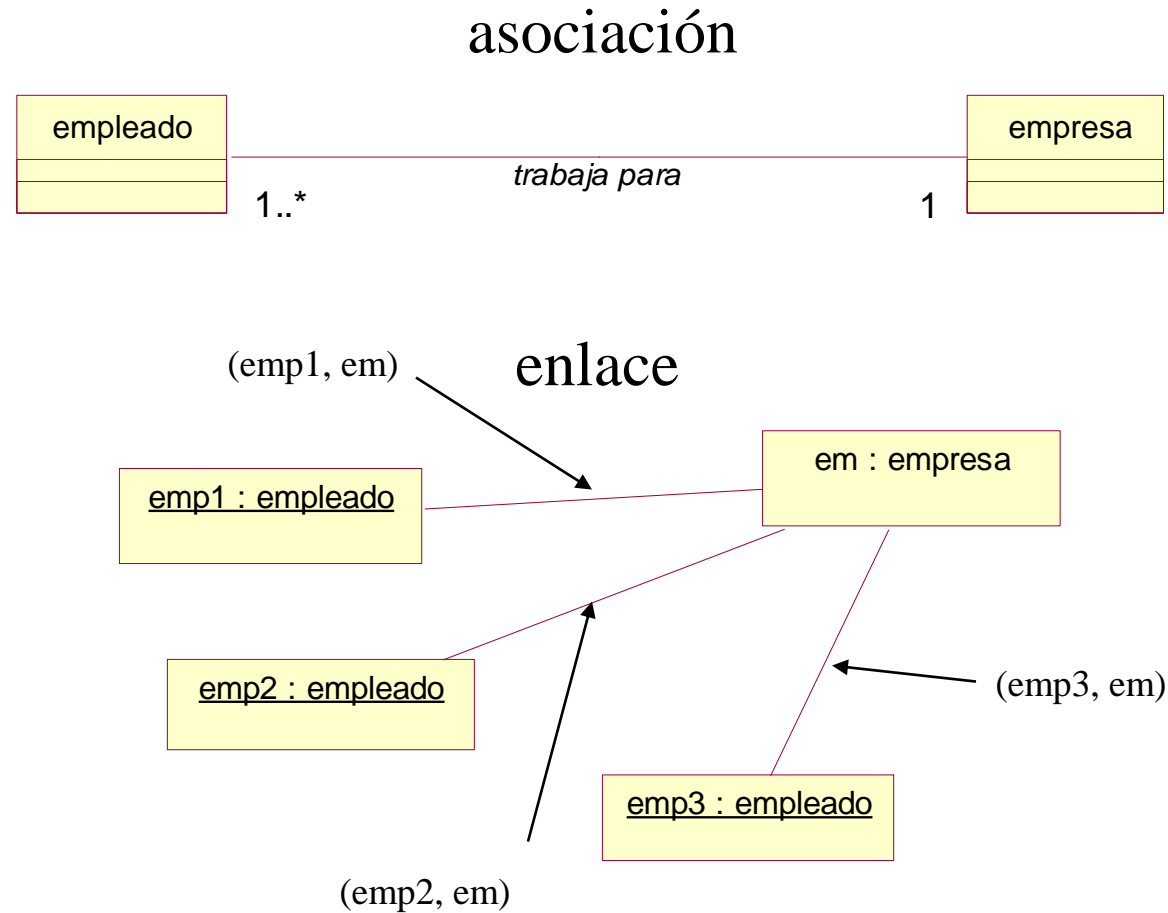
# CLASE ASOCIACIÓN

- Una asociación puede representarse por medio de una clase que permite añadir, por ejemplo, atributos y operaciones



Por ejemplo, debido a la multiplicidad, el nombre del cargo y el sueldo no pueden pertenecer a la empresa o al empleado; son atributos de la asociación

# INSTANCIA DE ASOCIACIÓN: ENLACE

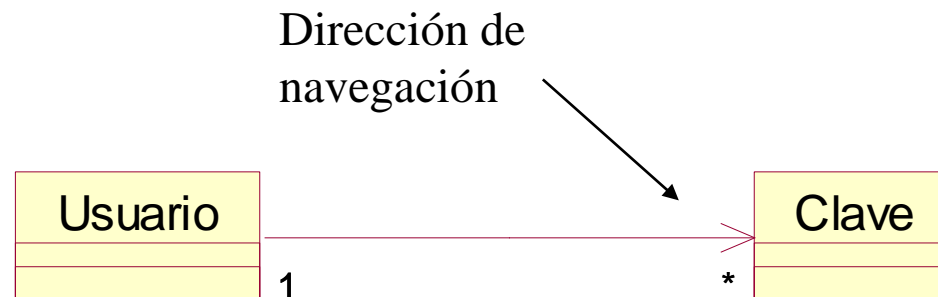


*cada instancia de una asociación (enlace) es una tupla de referencias a objetos*



# ASOCIACIONES – NAVEGACIÓN

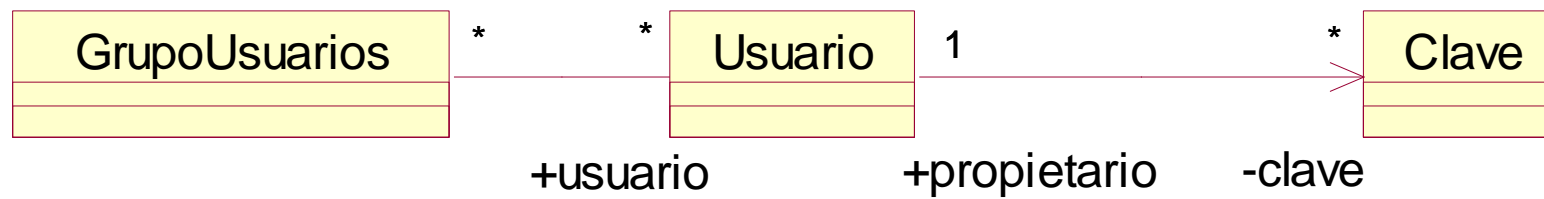
- La navegación indica que en una asociación es posible navegar de los objetos de un tipo a los de otro. Esto es debido a que el objeto inicial almacena alguna referencia del objeto navegable
- La navegación es el enunciado del conocimiento de una clase respecto a otra



# ASOCIACIONES – VISIBILIDAD DE ROL

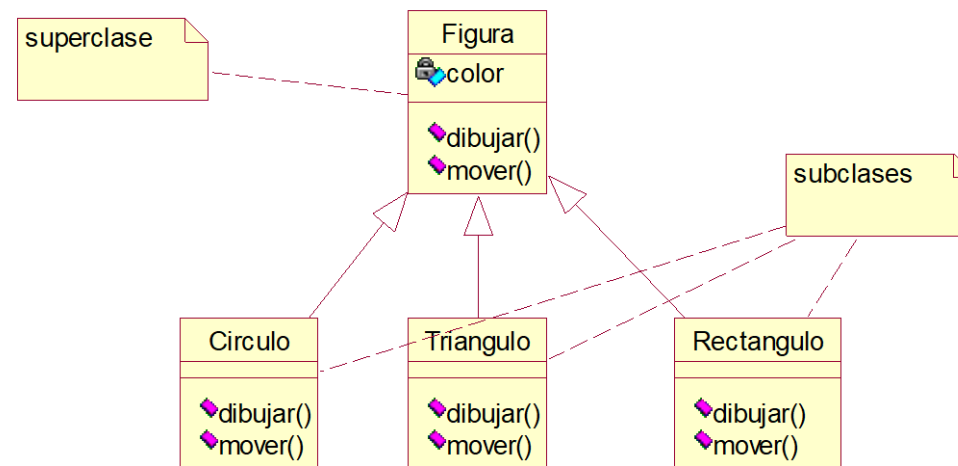
- Dada una asociación entre clases, los objetos de una clase pueden ver y navegar hasta los objetos de otra a menos que se restrinja específicamente
- La visibilidad privada indica que los objetos de ese extremo no son accesibles a ningún objeto externo a la asociación

Por ejemplo, un objeto Clave es accesible a un objeto Usuario pero no a un objeto GrupoUsuarios



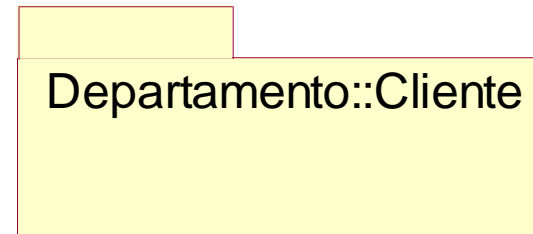
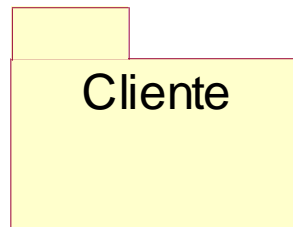
# GENERALIZACIÓN

- La generalización es una relación entre un elemento general (llamado superclase o padre ) y un tipo más específico de ese elemento (llamado subclase o hijo)
- El hijo puede añadir nueva estructura y comportamiento o modificar el comportamiento del padre
- La generalización consiste en factorizar los elementos comunes de un conjunto de clases en una clase más general llamada superclase



# PAQUETES/ I

- Un paquete es un mecanismo de propósito general para organizar el modelo de manera jerárquica
- Ayudan a organizar los elementos de modelado con el fin de comprenderlos. Los paquetes pueden tener dentro otros paquetes
- Los paquetes tienen un nombre que los identifica, el nombre puede ser simple o calificado (cuando le precede el nombre del paquete donde se encuentra)



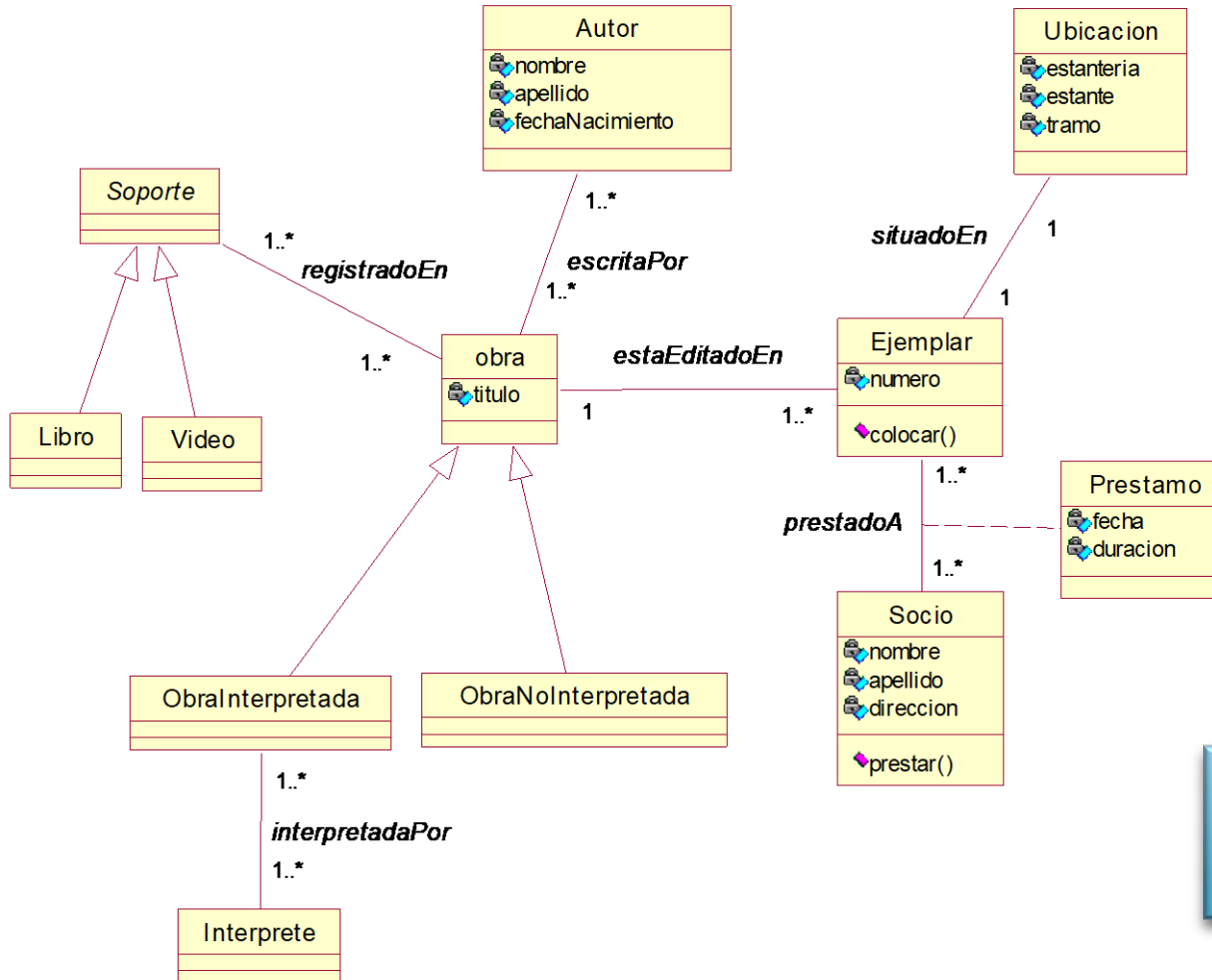
## PAQUETES/2

### ¿Cuáles son las ventajas de utilizar Paquetes?

- Evitar conflictos de nombres.
- Facilitar el reconocimiento y búsqueda de elementos de acuerdo a una funcionalidad específica.
- Controlar el acceso a sus contenidos.
- Todos los mecanismos de extensibilidad de UML se aplican a los paquetes.

# EJEMPLO

SEÑALADOR



En un modelo del dominio de la aplicación solo modelamos clases que hacen referencia a abstracciones de la realidad que son útiles a nuestros fines

# SUGERENCIAS

En la descripción de las clases del dominio del problema

- Identificar las clases y sus asociaciones más importantes
- Incluir los atributos más importantes
- No preocuparse inicialmente por las operaciones (corresponde a la etapa de diseño)
- No pensar en jerarquías (al principio...)

## SUGERENCIAS/2

En una etapa de refinamiento (ver proceso de desarrollo) incluir

- Relaciones de jerarquía Clase/subclase
- Agregaciones y composiciones
- Patrones de diseño Larman
- Patrones de diseño Gamma (opcional)
- Restricciones OCL (opcional)

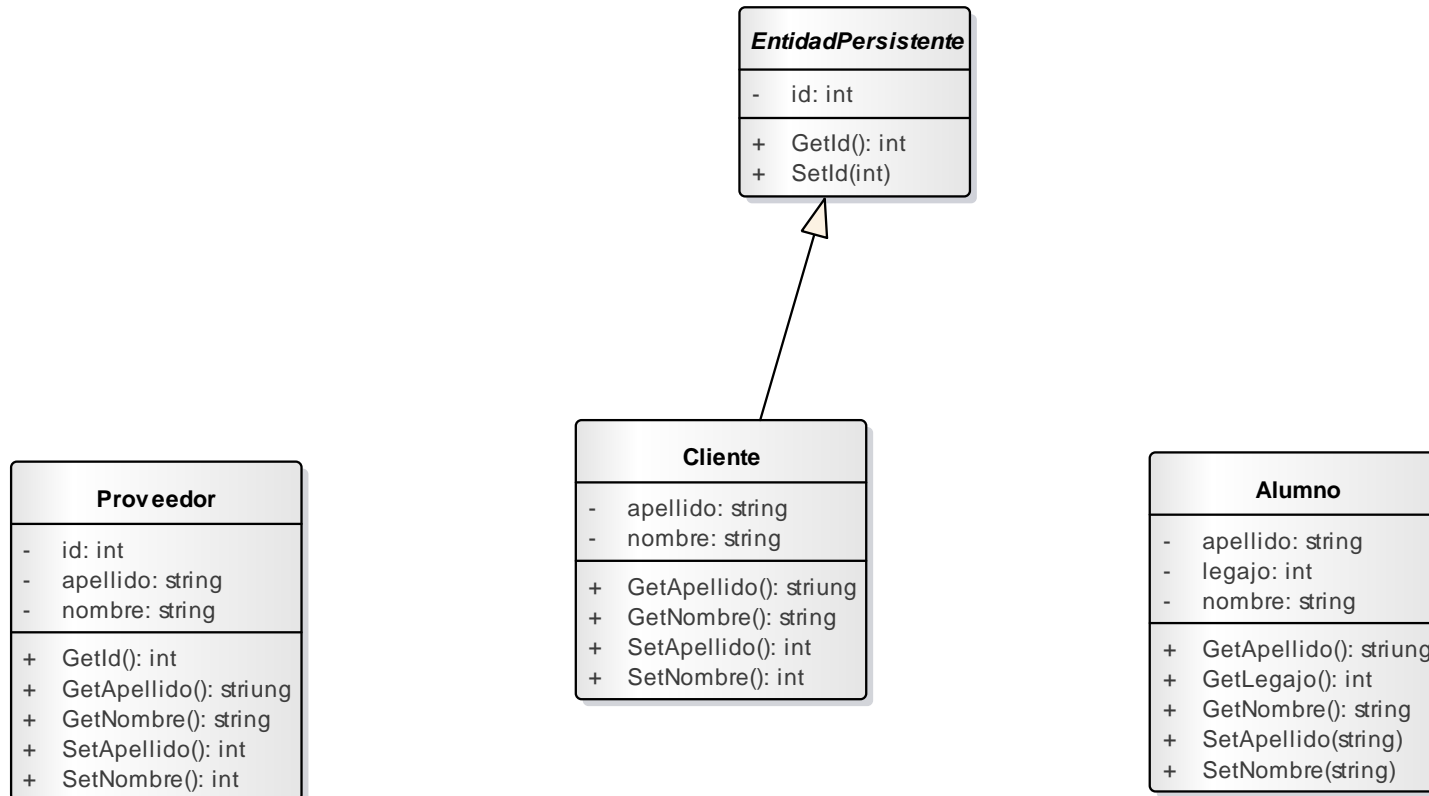


## SUGERENCIAS /2

- Al trabajar con clases en la vista de diseño, es importante definir cuales serán sus atributos y su comportamiento pensando en los conceptos de cohesión y acoplamiento.
- Se deberá pensar en utilizar diferentes técnicas de reutilización para lograr valores óptimos de estos conceptos, es decir herencia y composición de objetos para subir la cohesión y bajar el acoplamiento.
- No utilizar atributos que no describan conceptualmente los objetos, por ejemplo **ID**. Si bien es un atributo que probablemente necesitemos durante el diseño del software, no son parte de las entidades y esto debería verse reflejado.
- Es importante tratar de identificar cuales son los atributos que serán identificatorios, por ejemplo el alumno tiene un legajo y este será el que represente su identidad.

# SUGERENCIAS /3

class Modelo de Clases



# AUTO EVALUACIÓN/I

Comprendí los conceptos más importantes de la unidad 2.2. si puedo definir y dar ejemplos de:

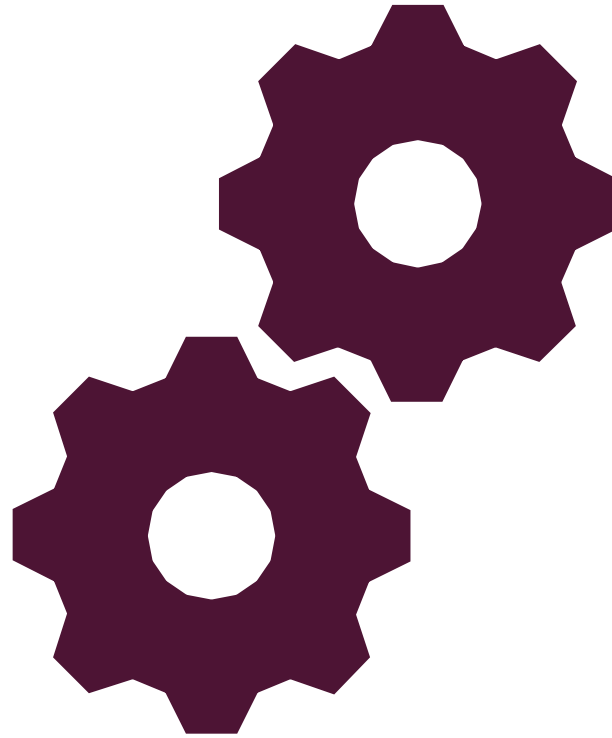
- Clase
- Clase abstracta
- Interfaz
- Operación y atributos de clases
- Dependencias
- nombre de rol y multiplicidad
- Asociación / agregación / composición
- Generalización
- Paquetes

# AUTO EVALUACIÓN/2

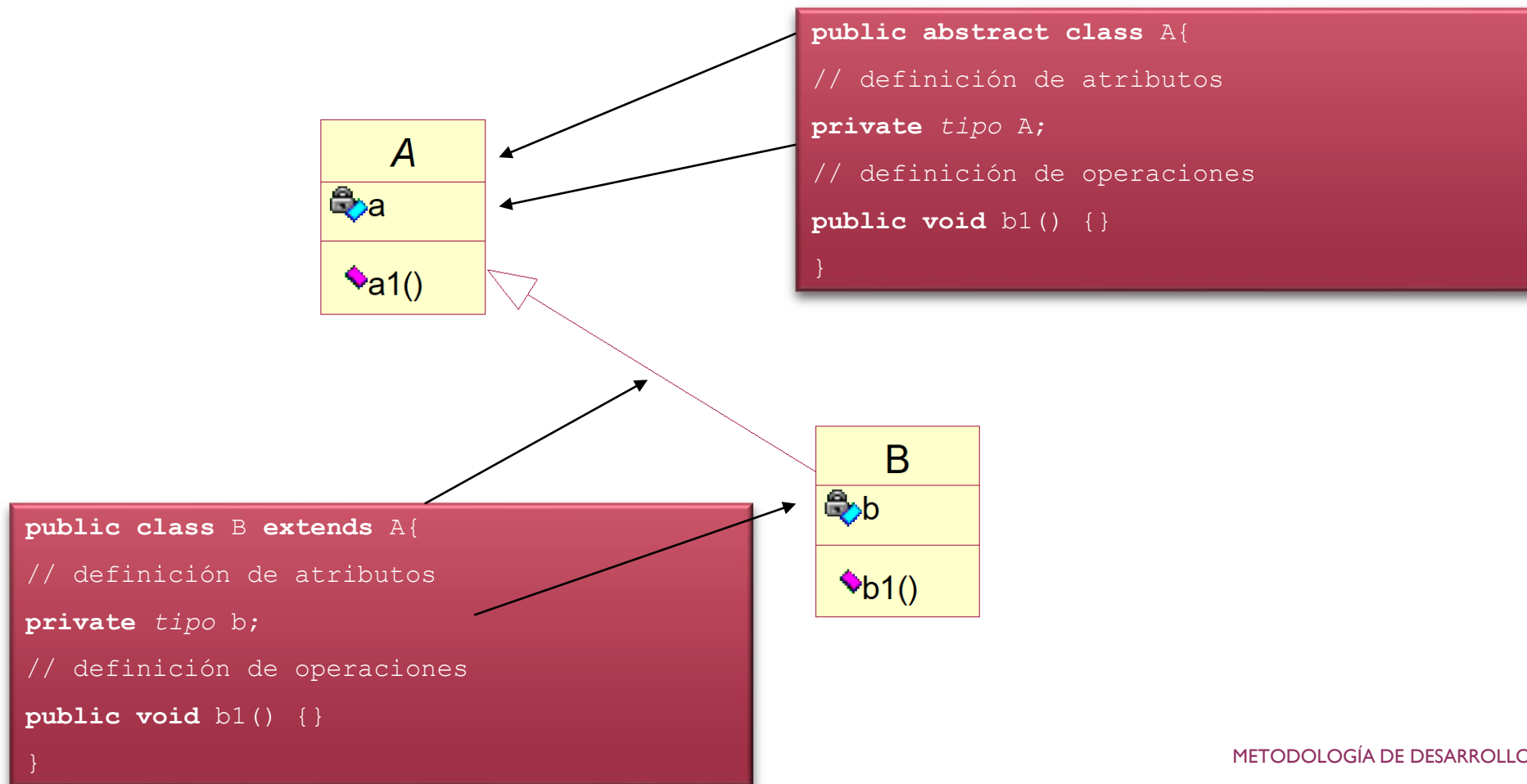
Comprendí los conceptos más importantes de la unidad 2.2, si

- Entiendo las diferencias entre clase, clase abstracta e interfaz
- Comprendo cuál es el objetivo de establecer una visibilidad determinada en atributos y operaciones y lo relaciono con el concepto de encapsulamiento
- Entiendo la diferencia conceptual entre agregación y composición y que ambas son tipos especiales de asociaciones
- Comprendo que el concepto de objeto y enlace son análogos
- Comprendo cómo y para qué utilizo las clases abstractas en las relaciones de generalización
- Entiendo cómo una clase hijo puede ampliar y/o modificar el comportamiento establecido en la clase padre
- Comprendo cómo usar los paquetes para organizar los elementos de modelado

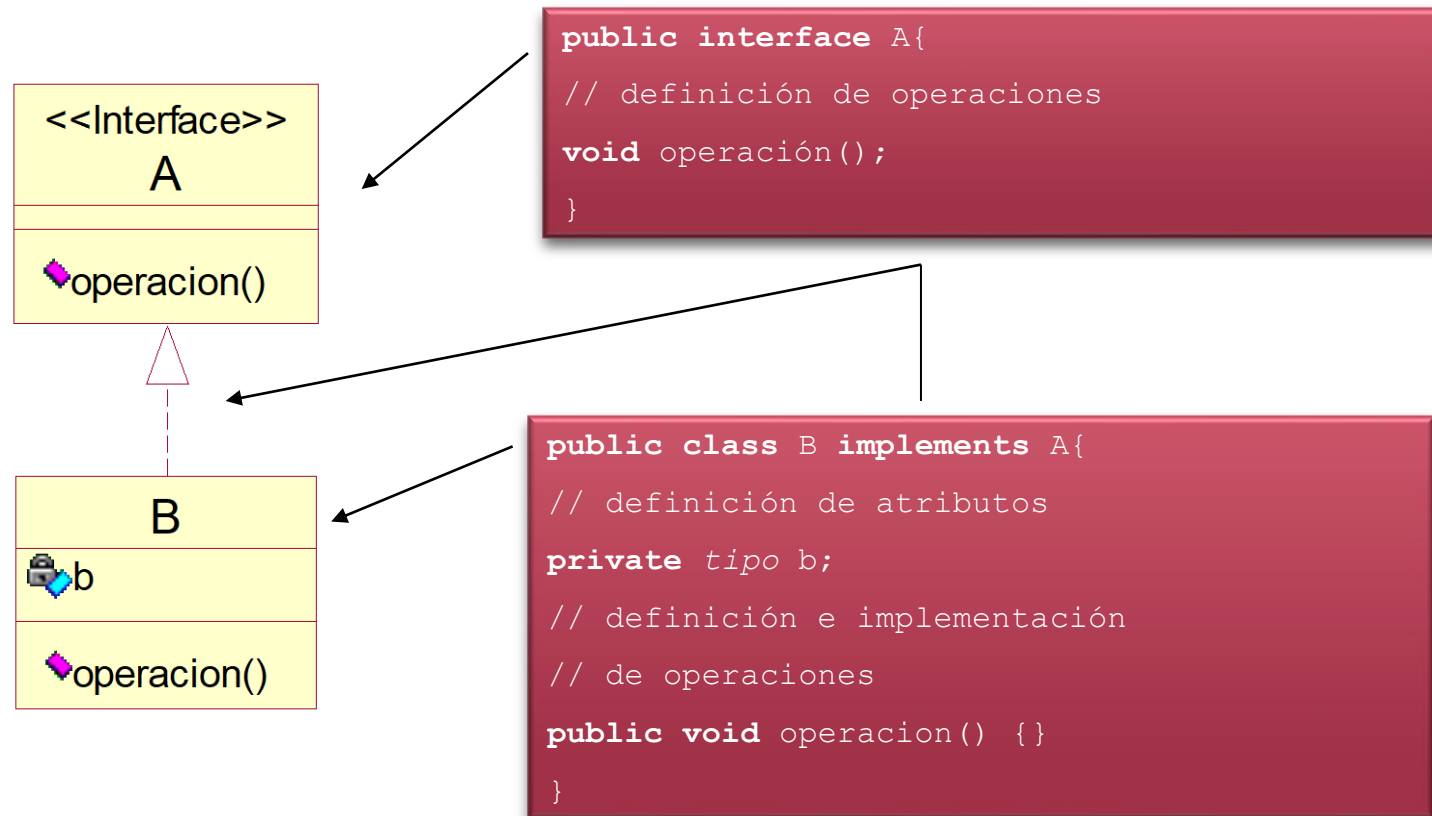
# IMPLEMENTACIÓN BÁSICAS DE LAS ABSTRACCIONES



# CLASES, CLASES ABSTRACTAS, GENERALIZACIÓN



# INTERFAZ



# ASOCIACIÓN 1..1



```
public class A{
    // definición de atributos
    private tipo a;
    // referencia a B mediante un atributo
    // del tipo B
    private B rol-b;
}
```

```
public class B{
    // definición de atributos
    private tipo b;
}
```



# ASOCIACIÓN 1 A MUCHOS



```
public class A{
// definición de atributos
private tipo a;
// * opción 1 *
// referencia al conjunto de objetos B
// mediante un vector
private Vector rol-b = new Vector();
}
```

```
public class A{
// definición de atributos
private tipo a;
// * opción 2 *
// referencia al conjunto de objetos B
// mediante un arreglo
private B rol-b[] = new B[n];
// "n" valor a definir
}
```

```
public class B{
// definición de atributos
private tipo b;
}
```

# ASOCIACIÓN MUCHOS A MUCHOS

(TIENEN PROBLEMAS DE INTEGRIDAD A RESOLVER MEDIANTE CÓDIGO)

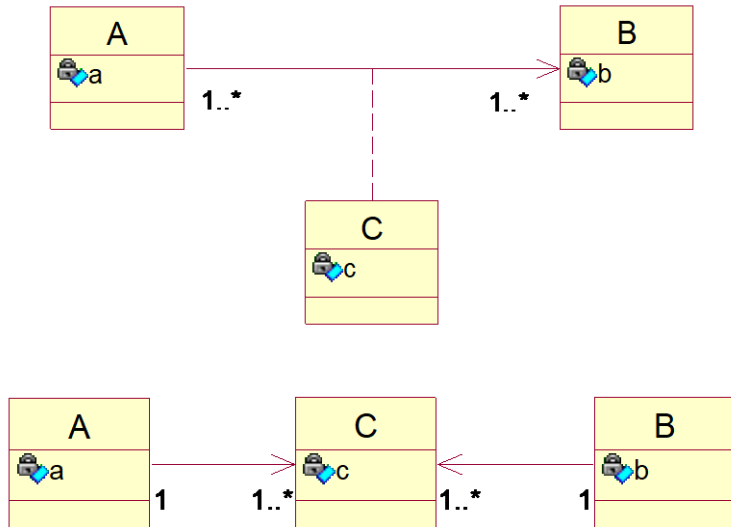


```
public class A{
// definición de atributos
private tipo b;
// referencia al conjunto de objetos C
// mediante un arreglo
private B a[] = new B[n];
}
```

```
public class B{
// definición de atributos
private tipo b;
// referencia al conjunto de objetos C
// mediante un arreglo
private A a[]= new A[n];
}
```

# CLASE ASOCIACION (UNA VERSION)

(TIENEN PROBLEMAS DE INTEGRIDAD A RESOLVER MEDIANTE CÓDIGO)



```
public class A{
    // definición de atributos
    private tipo a;
    // referencia al conjunto de
    // objetos C
    // mediante un arreglo
    private C c[] = new C[n];
}
```

```
public class B{
    // definición de atributos
    private tipo b;
    // referencia al conjunto de objetos C
    // mediante un arreglo
    private C c[]= new C[n];
}
```

```
public class C{
    // definición de atributos
    private tipo c;
    // referencia al objeto C
    // mediante atributos
    private B b;
    private A a;
}
```



Fin de la clase



**UAI**

**Universidad Abierta  
Interamericana**