UT5 – Elaboración de diagramas de clases

ENTORNOS DE DESARROLLO

Marcos Fernández Sellers

Índice de la Unidad

1. Objetivos

- Comprender los conceptos básicos del modelado visual.
- Conocer el lenguaje UML y los distintos tipos de diagramas.
- Aprender a realizar diagramas de clases.

2. Modelado Visual

- Un modelo es una simplificación de la realidad.
- El objetivo del modelado de un sistema es capturar las partes esenciales del sistema.
- Para facilitar este modelado, se realiza una abstracción y se plasma en una notación gráfica. Esto se conoce como modelado visual.

2. Modelado Visual

- Otro objetivo de este modelado visual es que sea **independiente del lenguaje de implementación**, y los diseños realizados se puedan implementar en cualquier lenguaje que soporte estas posibilidades (principalmente lenguajes orientados a objetos).

- El lenguaje de modelado más extendido es UML.

2. Modelado Visual

- El modelado visual permite manejar la complejidad de los sistemas a analizar o diseñar. De la misma forma que para construir una choza no hace falta un modelo, cuando se intenta construir un sistema complejo, como un rascacielos, es necesario abstraer la complejidad en modelos que el ser humano pueda entender.

3. Herramientas

draw.io

Visual Paradigm

ArgoUML

PlantUML

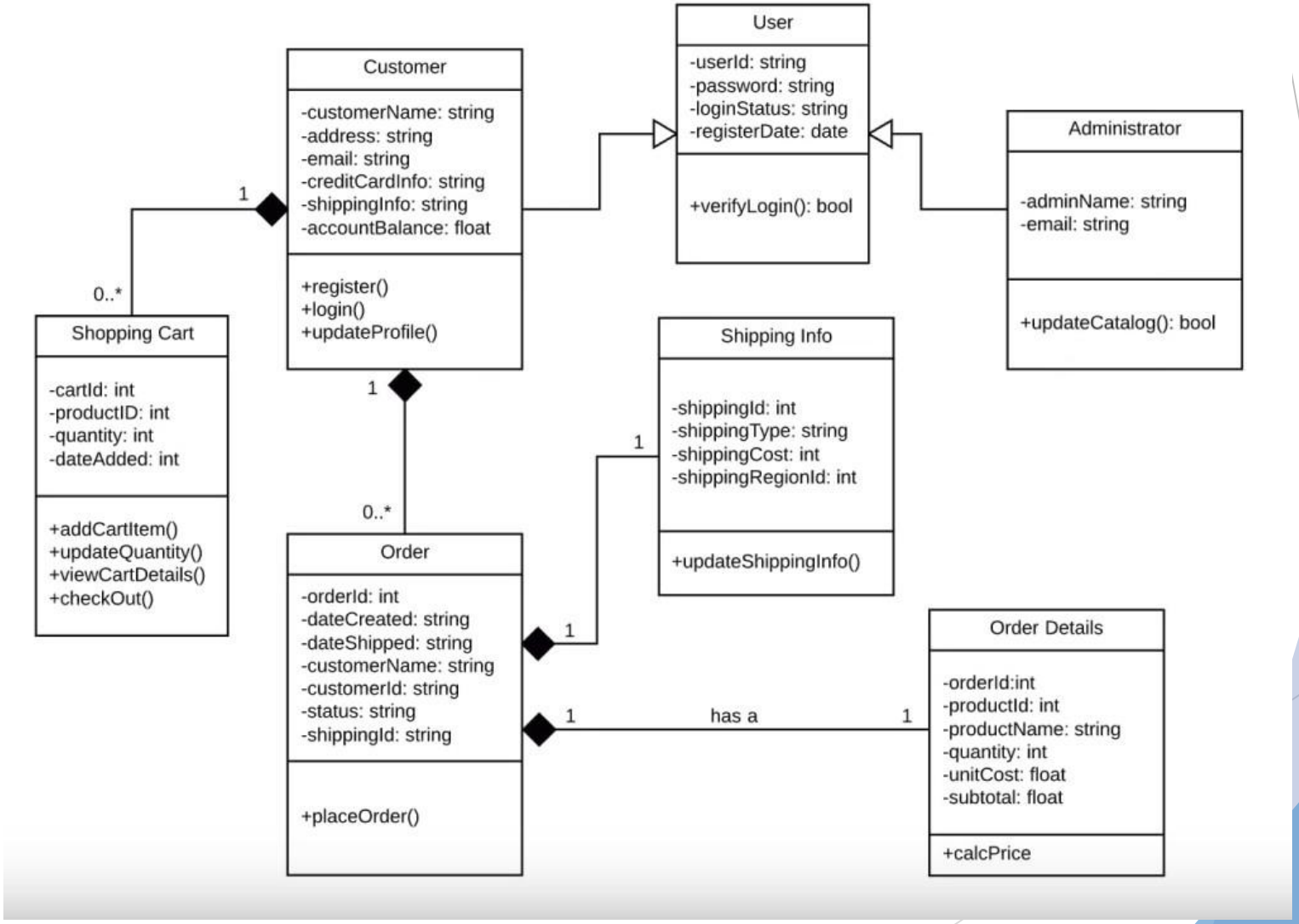
4. UML

- **UML** (Unified Modeling Language o Lenguaje Unificado de Modelado) es un conjunto de herramientas que permite modelar, construir y documentar los elementos que forman un sistema software orientado a objetos.
- Es un estándar que se ha adoptado a nivel internacional por numerosos organismos para crear esquemas o diagramas de programas informáticos para ver su estructura y comportamiento.
- El **objetivo** de UML es expresar visualmente los diagramas de forma que todos puedan entenderlos fácilmente.

4. UML

- UML permite a los desarrolladores visualizar el producto de su trabajo en esquemas o diagramas estandarizados denominados modelos que representan el sistema desde diferentes perspectivas.
- Modelo: representación gráfica o esquemática de una realidad, sirve para organizar y comunicar de forma clara los elementos que involucran un todo.

4. UML (Ejemplo)



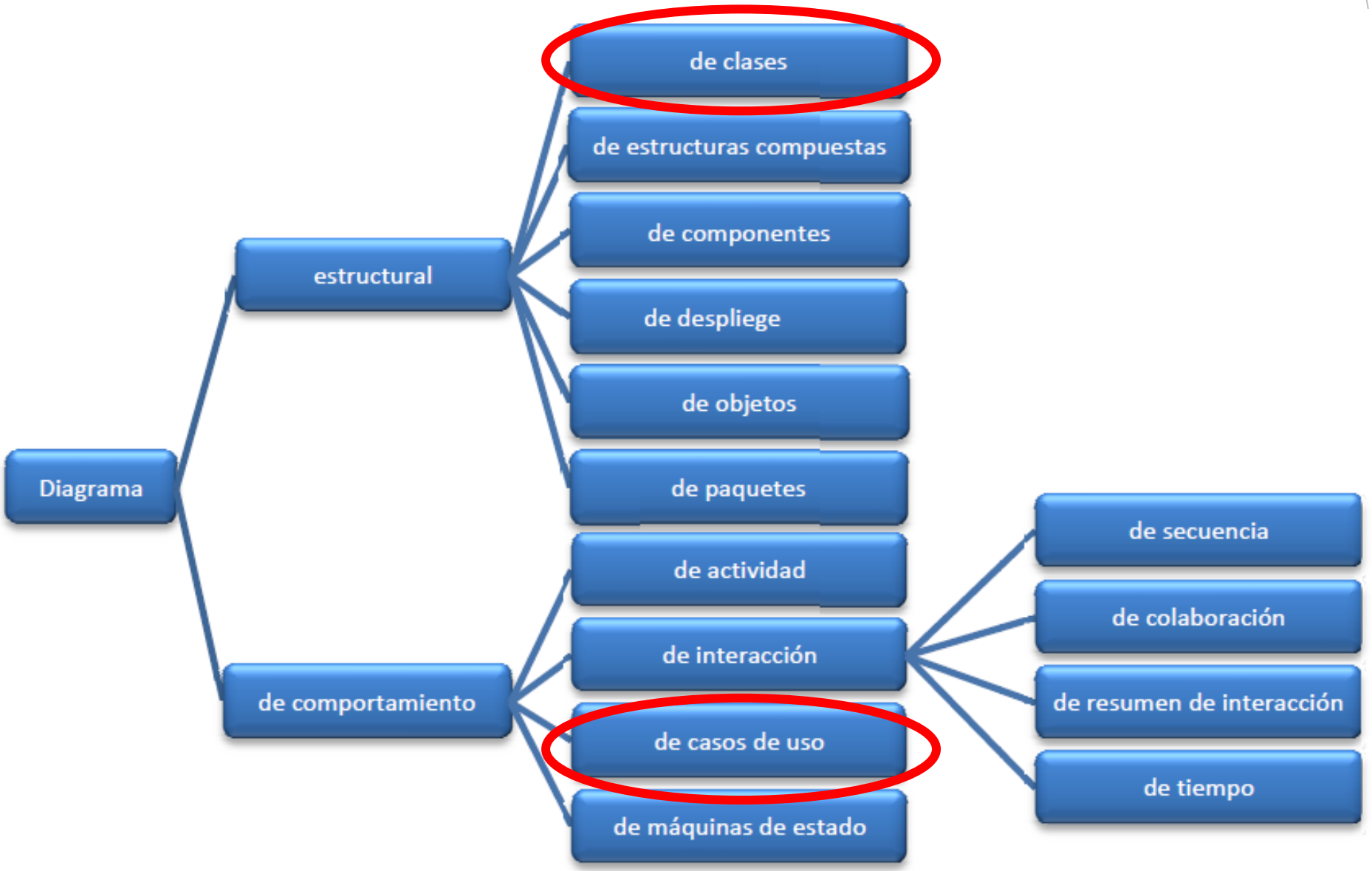
4.1. Beneficios de UML

- Simplifica la complejidad
- Mantiene las líneas de comunicación abiertas
- Automatiza la producción de software y los procesos
- Ayuda a resolver problemas de diseño persistentes
- Aumenta la calidad del trabajo
- Reduce los costes y el tiempo de comercialización

4.2. Tipos de UML

- Existen dos tipos: Unos para representar la estructura de los programas y otros para representar su comportamiento.
- En este tema vamos a ver los **Diagramas Estructurales**, que **representan la estructura estática de un programa de software o sistema**, y también **muestran los diferentes niveles de abstracción e implementación**. Se utilizan para poder visualizar las distintas estructuras que componen un sistema, como una base de datos o una aplicación. Muestran la jerarquía de los componentes o módulos, y la forma en que se conectan o interactúan entre sí.

4.2. Tipos de UML



Práctica 1

- Elige cuatro de los distintos Diagramas Estructurales UML que quieras (**distintos del diagrama de clases**), di en qué consisten y pon un ejemplo de dichos diagramas.
- Haz lo mismo pero ahora usando Diagramas de comportamientos (**distintos del diagrama de casos de usos**).

4.3. Diagrama de clases

- Es el tipo de diagrama UML más utilizado.
- Es el bloque de construcción principal de cualquier solución orientada a objetos.
- Muestra:
  - Clases de un sistema: Atributos y funciones o métodos
  - Relaciones entre clases (tipos de relación)

4.3.1. Cómo crear un diagrama de clases

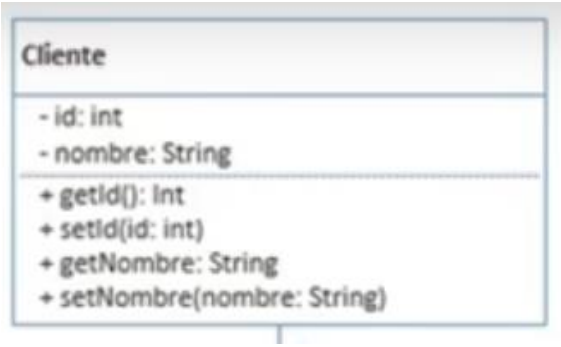
1. Identifica los nombres de las clase. El primer paso es identificar los objetos primarios del sistema. Las clases suelen corresponder a sustantivos dentro del dominio del problema.
2. Distingue las relaciones. El siguiente paso es determinar cómo cada una de las clases u objetos están relacionados entre sí. Busca los puntos en común y las abstracciones entre ellos; esto te ayudará a agruparlos al dibujar el diagrama de clase.
3. Crea la estructura. Primero, agrega los nombres de clase y vincúlalos con los conectores apropiados, prestando especial atención a la cardinalidad o las herencias. Deja los atributos y funciones para más tarde, una vez que esté la estructura del diagrama resuelta.

4.3.2. Clases

- Las clases son el elemento principal del diagrama y representa una clase dentro del paradigma de la orientación a objetos.
- Una clase define un grupo de objetos que comparten características, condiciones y significado.
- Ejemplo de clases: Animal, Persona, alumno, coche, clientes, …

4.3.2.1. Clases. Elementos de una clase

- La primera de las zonas se utiliza para el nombre de la clase.
- La segunda de las zonas se utiliza para escribir los atributos de la clase, uno por línea y poniendo el nombre : y el tipo.
- La última de las zonas incluye cada una de las funciones que ofrece la clase (Métodos).

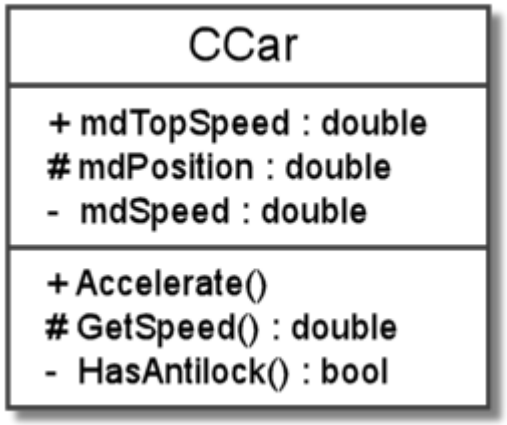


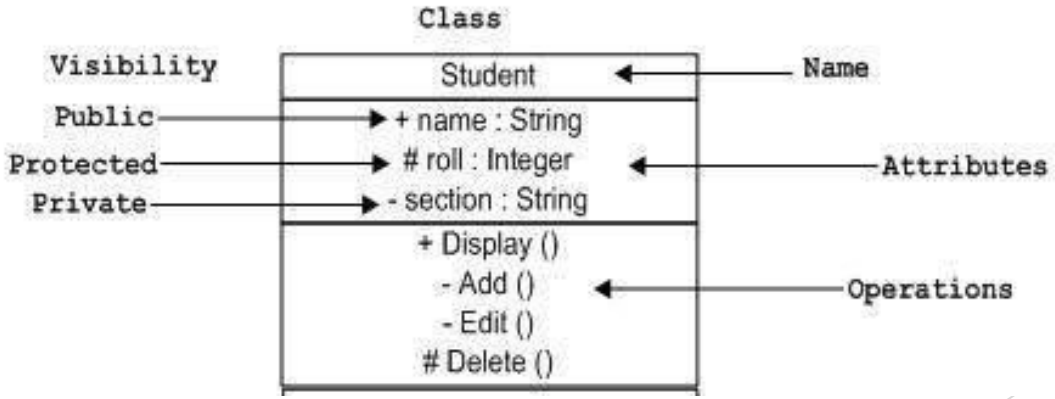
4.3.2.2. Clases. Tipos de atributos

- Los atributos de una clase pueden ser de tres tipos, que definen su visibilidad dentro de la clase:
  - Public (+). Atributo visible tanto dentro como fuera de la clase.
  - Protected (#). Atributo accesible desde dentro de la clase y sus posibles subclases heredadas.
  - Private (-). Atributo accesible solo desde dentro de la clase.

4.3.2.3. Clases. Tipos de métodos

- De manera similar a los atributos, también los diferenciamos en Public, Private o Protected.
- Es necesario incluir el tipo de datos que devuelven y sus parámetros

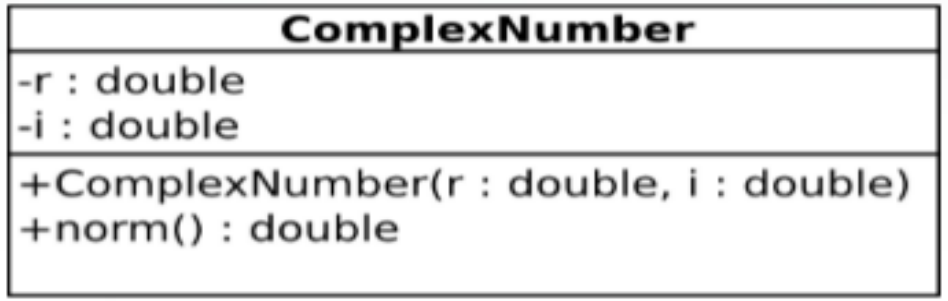




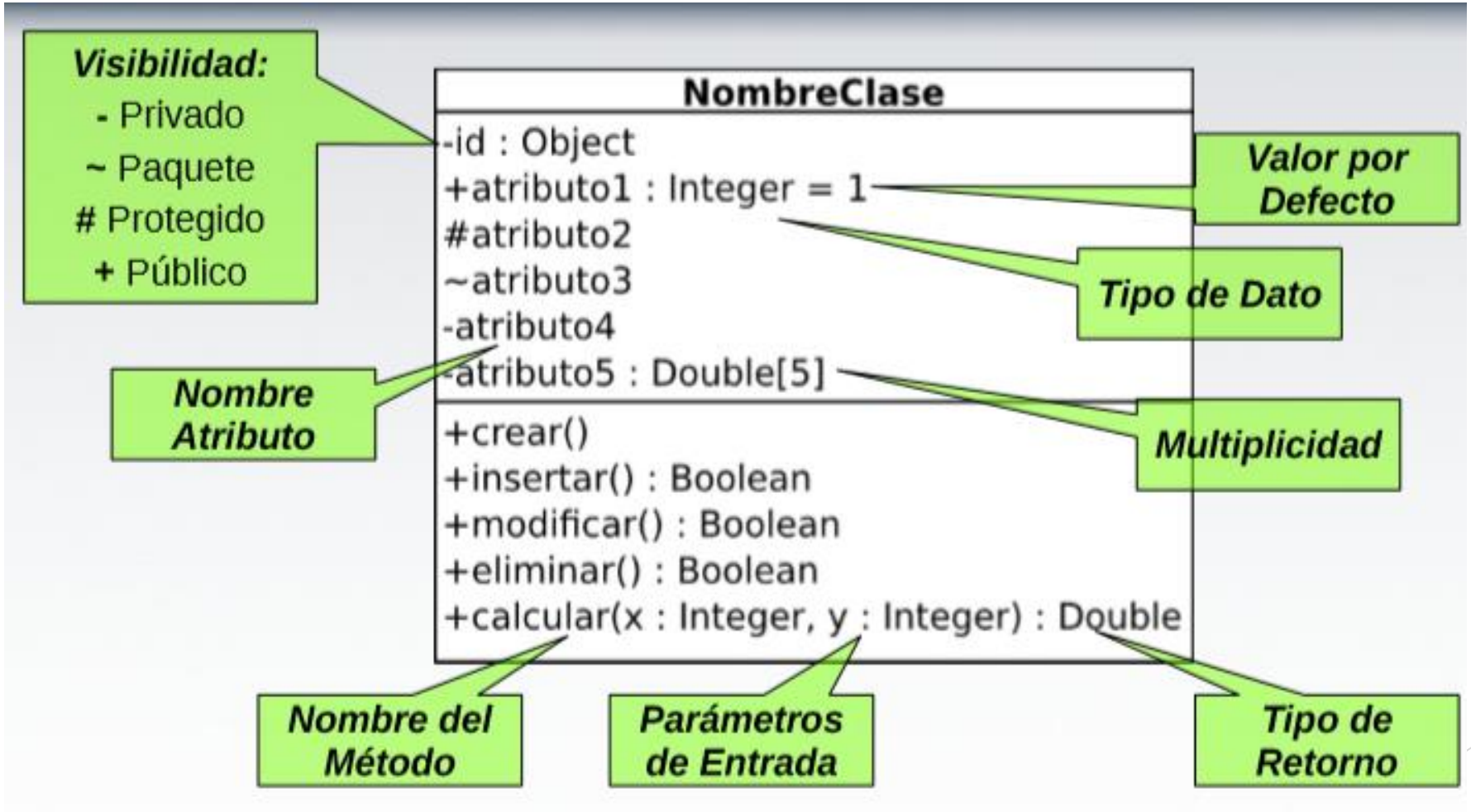
4.3.2.4. Ejemplo

```java
public class ComplexNumber {

   private double r;
   private double i;

   public ComplexNumber(double r, double i) {
      this.r = r;
      this.i = i;
   }

   public double norm() {
      return Math.sqrt(r * r + i * i);
   }
}
```

El código es Java ;-)



4.3.2.5. Resumen clases



4.3.2.6. Criterios de decisión

Cuando tengamos la lista completa habrá que estudiar cada objeto potencial para ver si, finalmente, es incluido en el diagrama. Para ayudarnos a decidir podemos utilizar los siguientes criterios:

1. La información del objeto es necesaria para que el sistema funcione.
2. El objeto posee un conjunto de atributos que podemos encontrar en cualquier ocurrencia del objeto. Si sólo aparece un atributo normalmente se rechazará y será añadido como atributo de otro objeto.

4.3.2.6. Criterios de decisión

3. El objeto tiene un conjunto de operaciones identificables que pueden cambiar el valor de sus atributos y son comunes a cualquier ocurrencia del objeto.
4. Es una entidad externa que consume o produce información esencial para la producción de cualquier solución en el sistema.

El objeto se incluye si cumple todos (o casi todos) los criterios.

Práctica 2. Alquiler de automóviles

Se desea diseñar un diagrama de clases sobre la información de las reservas de una empresa dedicada al alquiler de automóviles, teniendo en cuenta que:

- Un determinado cliente puede tener en un momento dado hechas varias reservas.
- De cada cliente se desean almacenar su DNI, nombre, dirección y teléfono, así como un código de cliente.
- Cada cliente puede ser avalado por otro cliente de la empresa.
- Una reserva la realiza un único cliente pero puede involucrar varios coches.
- Es importante registrar la fecha de inicio y final de la reserva, el precio del alquiler de cada uno de los coches, los litros de gasolina en el depósito en el momento de realizar la reserva, el precio total de la reserva y un indicador de si el coche o los coches han sido entregados.
- Todo coche tiene siempre asignado un determinado garaje que no puede cambiar. De cada coche se requiere la matricula, el modelo el color y la marca.
- Cada reserva se realiza en una determinada agencia.
- Tanto de la agencia como del garaje se guardará un código y el nombre, además de la dirección de la agencia.

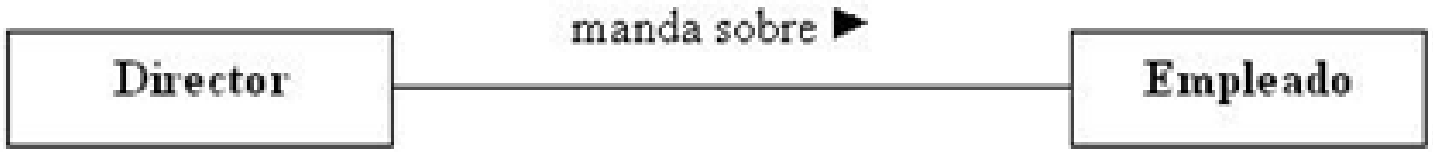
Diseña solo las clases de este diagrama en una de las herramientas mostradas

4.3.3 Tipos de relaciones

4.3.3.1. Asociaciones

- Una asociación es una relación estructural que especifica que los objetos de un elemento están conectados con los objetos de otro. Es una relación débil.

Representación de asociación

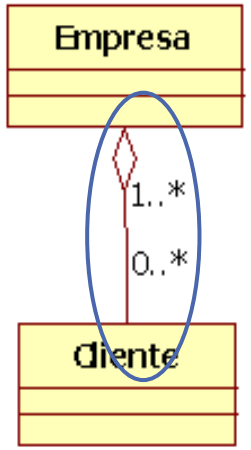


4.3.3.2. Agregación

- **La agregación es un tipo de asociación que indica que una clase es parte de otra clase (composición débil).**
- Veamos un ejemplo de agregación:

La Agregación nos dice «*tiene un*«.

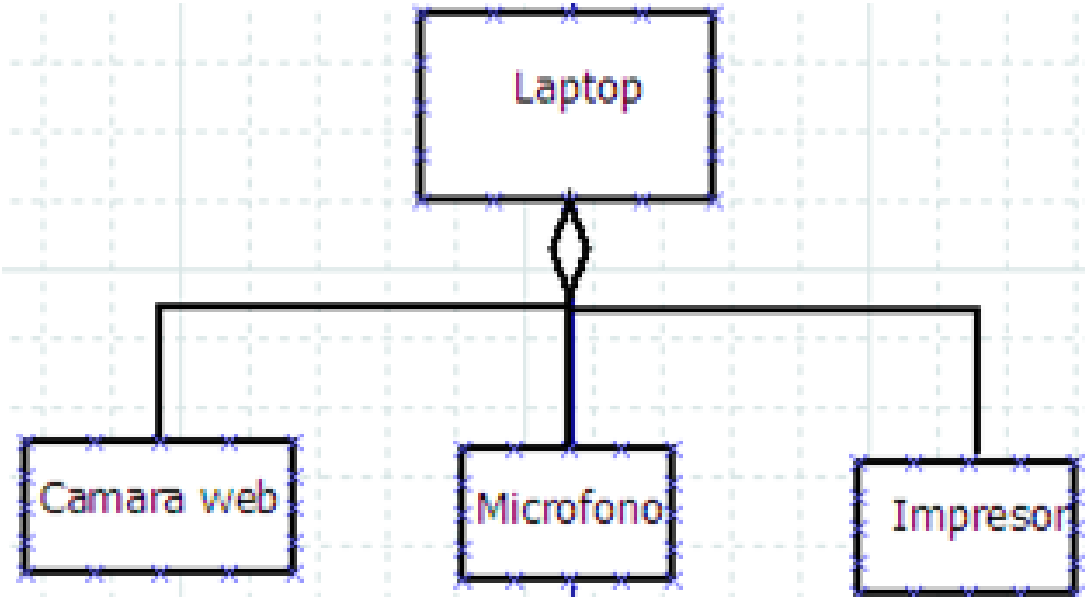
Cuando eliminamos la clase empresa, no tenemos porqué eliminar la clase Cliente, de manera tal que la clase cliente podría continuar existiendo.



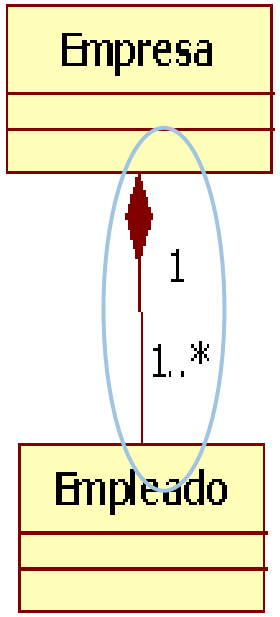
- Tenemos una clase Empresa.
- Tenemos una clase Cliente.
- Una empresa agrupa a varios clientes.

Ejemplo de Agregación

Por ejemplo sabemos que a una computadora portátil se le pueden agregar elementos como micrófono, cámara web e impresora; sin embargo **la ausencia de estos elementos no repercute en el funcionamiento básico de dicha portátil**. El símbolo de agregación es un diamante vacio colocado en el extremo de la clase que contiene las clases agregadas.



4.3.3.3. Composición



La Composición nos dice «*es parte de*«.
La ***clase es parte de la otra (Composición fuerte)***
En esta relación, no tiene sentido que la clase empleado viva de forma independiente sin formar parte de la clase Empresa.

También diremos **que al eliminar la clase empresa, eliminaremos por lo tanto, la clase empleado, ya que la relación entre ambas clases, es estrecha.**

- Tenemos una clase Empresa.
- Un objeto Empresa está a su vez compuesto por uno o varios objetos del tipo empleado.
- El tiempo de vida de los objetos Empleado depende del tiempo de vida de Empresa, ya que si no existe una Empresa no pueden existir sus empleados.

Ejemplo de Composición

La composición a diferencia de la agregación representa una clase que está compuesta por otras clases que son indispensables para que esta funcione. Por ejemplo podríamos decir que la laptop citada en el ejemplo anterior funciona si le quitamos la cámara web; sin embargo no funcionará si le faltase la pantalla, por lo que podemos decir que una laptop está compuesta básicamente de una pantalla y su unidad de procesamiento (CPU).

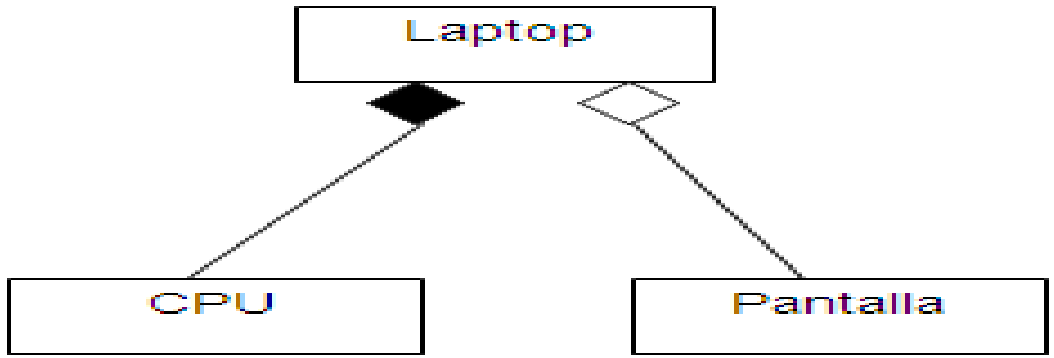
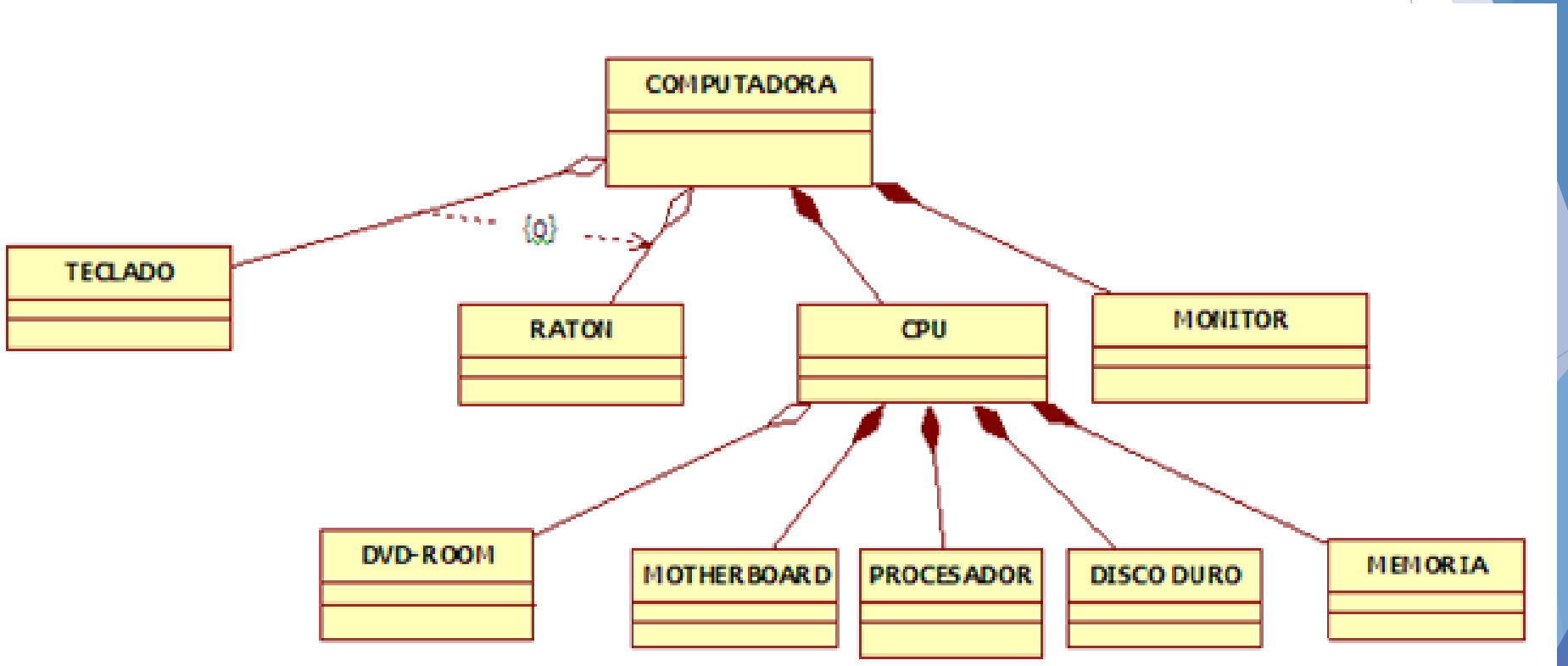


Figura. Representación de composición

Diferencia entre agregación y composición

- **Agregación es una relación débil.**
- **Composición es una relación fuerte.**



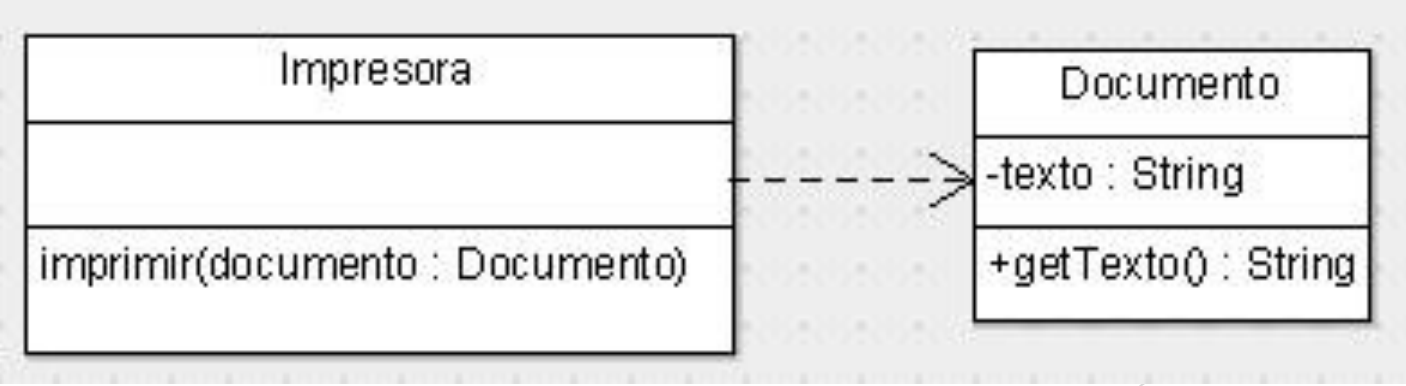
4.3.3.4. Dependencia o instanciación

- **Es una relación de uso entre dos clases (una usa a la otra).** Esta relación es la más básica entre clases y comparada con los demás tipos de relación, **la mas débil**.

Ejemplo Práctico:

- Tenemos una clase *Impresora*..
- Tenemos una clase *Documento* con un atributo *texto*.
- La clase *Impresora* se encarga de imprimir los *Documentos*.
- Para esto generamos una relación de dependencia:

(La impresora usa un documento para imprimirlo)



Otro ejemplo de dependencia

(Para resolver una ecuación usa las funciones)

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Para resolver una ecuación de segundo grado hemos de recurrir a la función `sqrt` de la clase `Math` para calcular una raíz cuadrada.

4.3.4. Cardinalidad de las relaciones

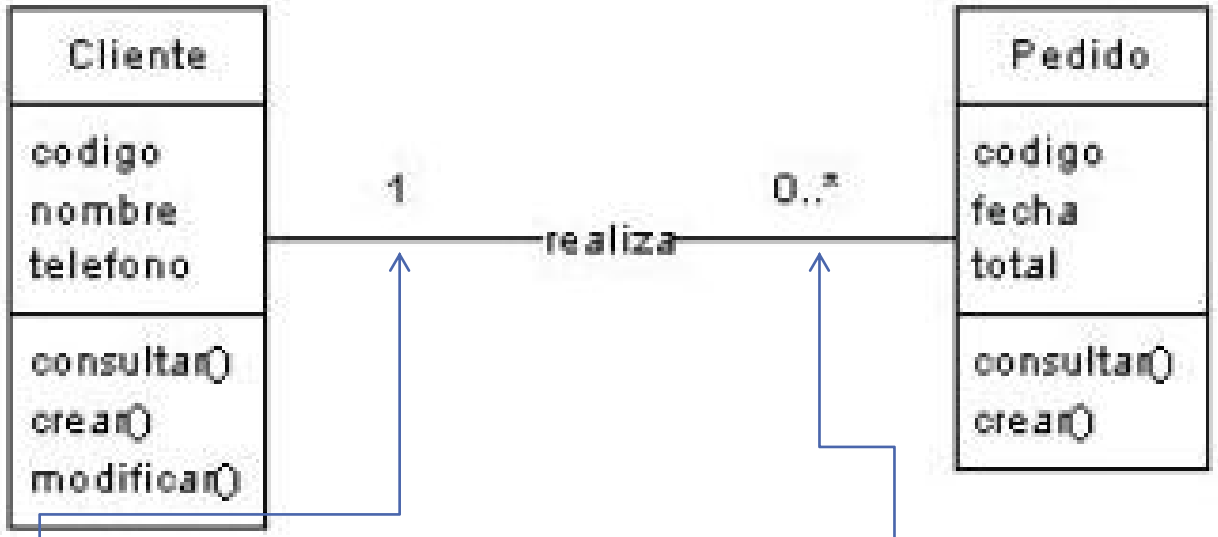
Antes es necesario explicar el concepto de cardinalidad de relaciones: En UML, la cardinalidad de las relaciones indica el grado y nivel de dependencia, se anotan en cada extremo de la relación y éstas pueden ser:

- **uno o muchos**: 1..* (1..n)
- **0 o muchos**: 0..* (0..n)
- **número fijo**: m (m denota el número)

4.3.4. Cardinalidad de las relaciones

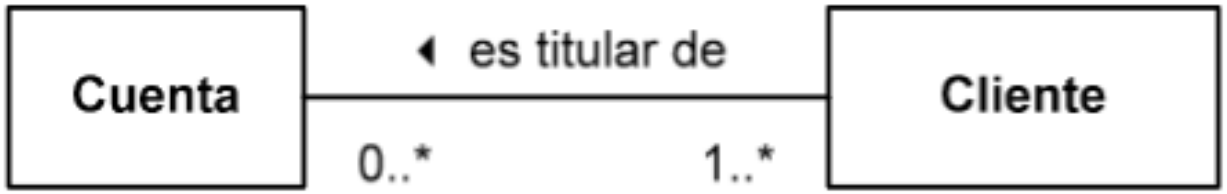
Significado de las cardinalidades.	
Cardinalidad	**Significado**
1	Uno y sólo uno
0..1	Cero o uno
N..M	Desde N hasta M
*	Cero o varios
0..*	Cero o varios
1..*	Uno o varios (al menos uno)

4.3.4.1. Ejemplos cardinalidad (I)



→Un cliente puede realizar muchos pedidos
→ Un pedido es realizado por solo un cliente (tiene un código)

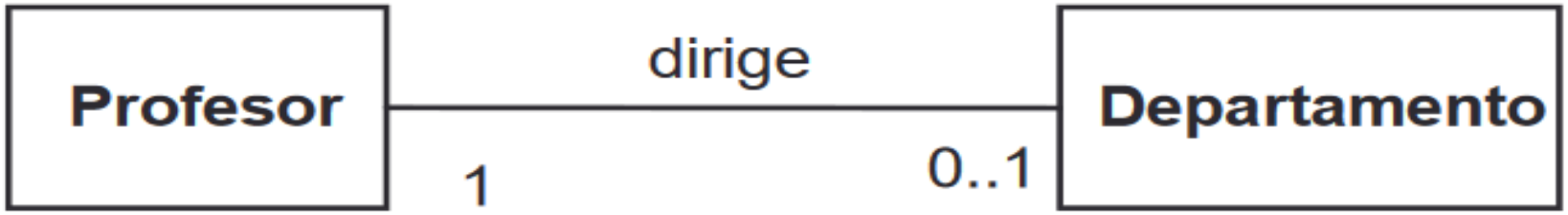
4.3.4.1. Ejemplos cardinalidad (II)



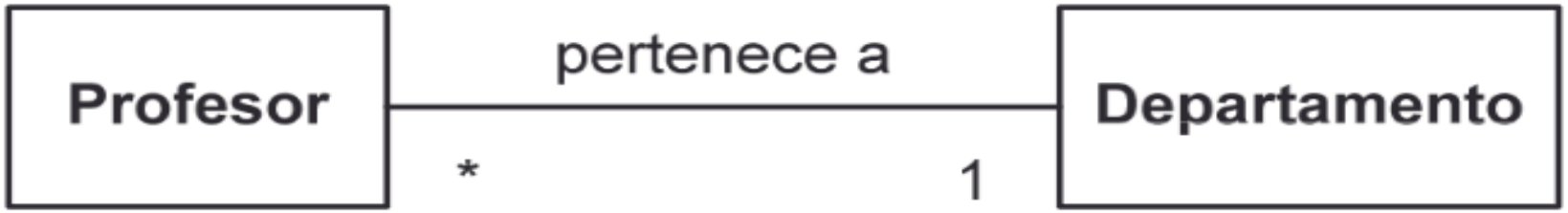
Relación opcional
Un cliente puede o no ser titular de una cuenta

Relación obligatoria
Una cuenta ha de tener un titular como mínimo

4.3.4.1. Ejemplos cardinalidad (III)

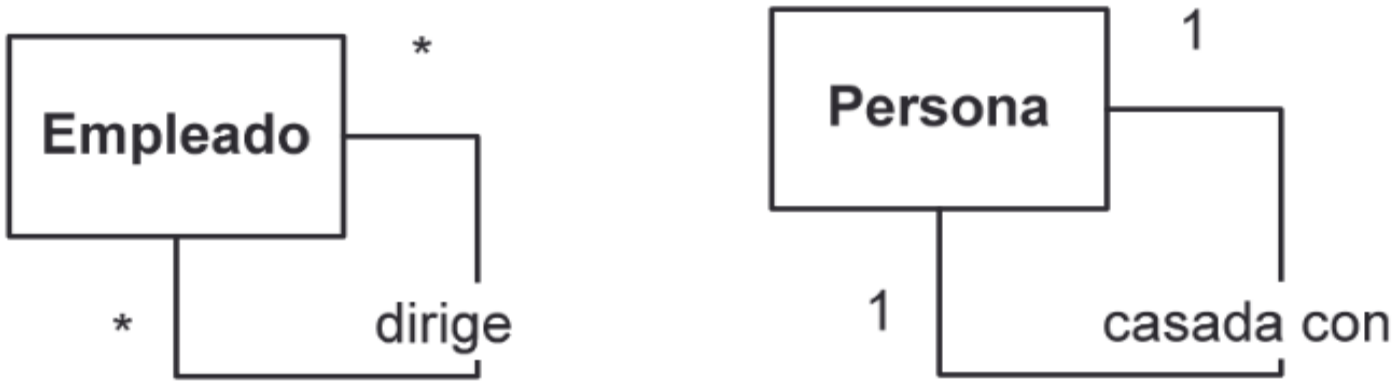


Todo departamento tiene un director.
Un profesor puede dirigir un departamento.



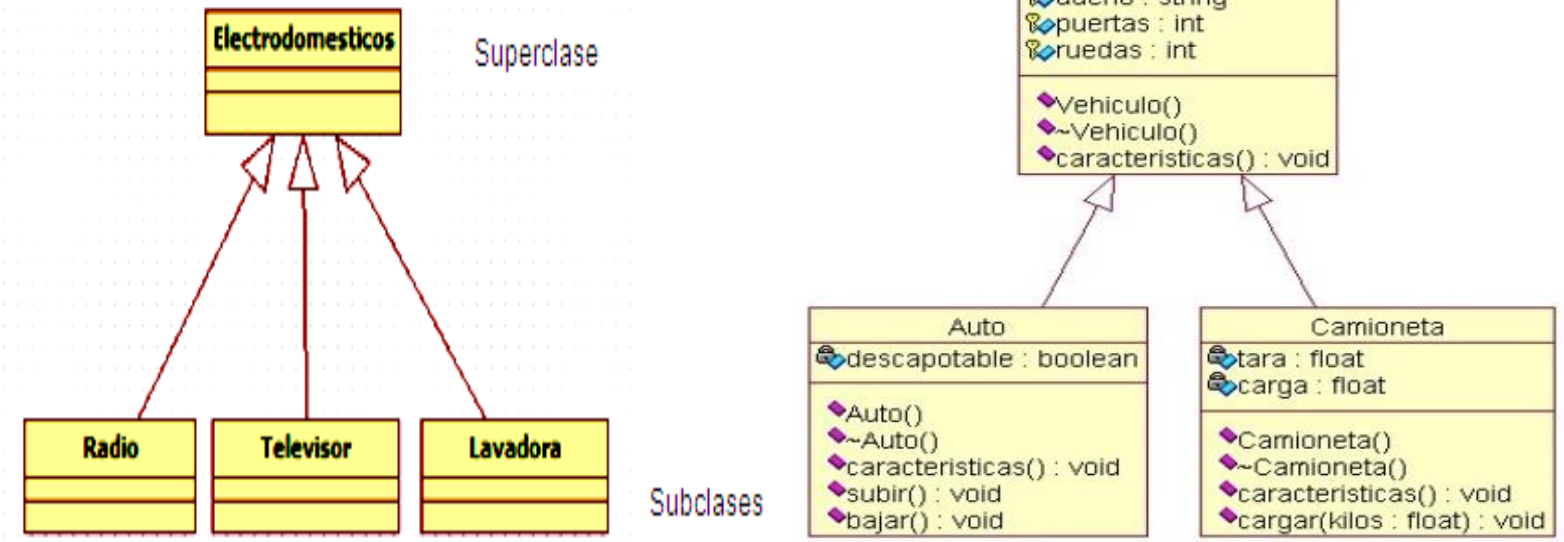
Todo profesor pertenece a un departamento.
A un departamento pueden pertenecer varios profesores.

4.3.4.1. Ejemplos cardinalidad (IV)



4.3.5. Generalización o herencia

- Indica que una subclase hereda los métodos y atributos especificados por una superclase, de esta forma la subclase además de poseer sus propios métodos y atributos, poseerá las características y atributos visibles de la superclase (public y protected).



Práctica 2 (bis). Alquiler de automóviles

Partiendo de las clases realizadas anteriormente, ahora incluye también las distintas relaciones entre las clases