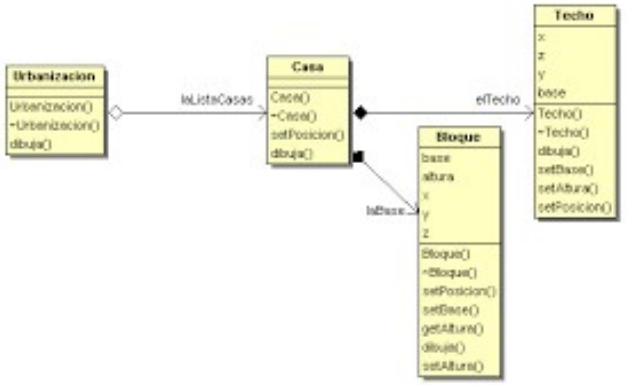
# UD5: UML. Elaboración de Diagrmas de clases

Entornos de desarrollo

Mª Carmen Safont Richarte

# INDICE

1. UML: Unified Modeling Language
2. Tipos de diagrama UML
3. Elaboración de Diagramas de clases.



# 1. UML: Unified Modeling Language

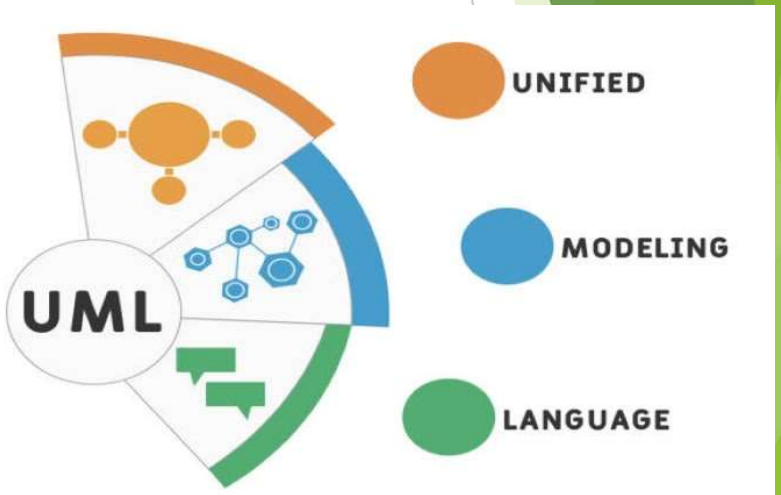
## CONTENIDOS

# 1. UML: Unified Modeling Language

## 1.1. Introducción.

*Si viajas a un país anglosajón como EEUU o Inglaterra, tendrás que hablar inglés para hacerte entender, si vas a un país hispanohablante como Argentina, tendrás que hablar español, y si vas a una empresa de desarrollo de software, tendrás que manejar UML, que es el lenguaje que se utiliza en los proyectos software orientados a objetos.*

*Por tanto, UML es una herramienta fundamental que necesitas conocer porque luego la vas a utilizar a lo largo de tu vida profesional.*



# 1. UML: Unified Modeling Language

## 1.2. ¿Qué es UML?

**Definición oficial de UML:**

*UML es un lenguaje de propósito general que ayuda a especificar, visualizar y documentar modelos de sistemas software, incluido su estructura y diseño, de tal forma que se unifiquen todos sus requerimientos*

# 1. UML: Unified Modeling Language

## 1.2. ¿Qué es UML?

- Es un lenguaje visual estandarizado de modelado.
- Está especialmente desarrollado para ayudar a todos los intervinientes en el desarrollo y modelado de un sistema o un producto software a describir, diseñar, especificar, visualizar, construir y documentar todos los artefactos que lo componen.
- Dispone de numerosos tipos de diagramas.
- Cada tipo de diagrama muestra un aspecto diferente del **modelo** de software.

*El modelado es la principal forma de visualizar el diseño de una aplicación con la finalidad de compararla con los requisitos antes de que el equipo de desarrollo comience a codificar.*

# 1. UML: Unified Modeling Language

## 1.2.1. Modelos y Diagramas

- **Modelar** consiste en diseñar aplicaciones software antes de su implementación. Es una parte esencial en proyectos de tamaño medio o grande.
- Un **modelo** es una abstracción de un elemento del sistema a desarrollar, desde un punto de vista determinado y, por lo tanto, con un nivel de detalle específico.
- Un **diagrama** consiste en una representación gráfica de una colección de modelos para representar una parte específica del sistema.
- Con el modelado del sistema podemos definir todos sus requisitos de forma totalmente a su implementación.
- Esto permite tener una **visión global** del sistema antes de iniciar su desarrollo, lo que implica:
  - Tener unos requisitos bien definidos.
  - Reducir problemas en el proceso de implementación.
  - Tener una visión global del sistema en una primera fase de su desarrollo.
  - Para poder modelar un sistema y conducir el proceso de desarrollo, es necesario el uso de una solución software

# 1. UML: Unified Modeling Language

## 1.2.1. Modelos y Diagramas

- **Los diagramas de UML** son representaciones gráficas que muestran de forma parcial un sistema de información, bien esté siendo desarrollado o ya lo haya sido. Suelen estar acompañados de documentación que les sirve de apoyo, adoptando múltiples formas. Además, UML no excluye la posibilidad de mezclar diagramas, algo que, de hecho, suele ser bastante común. Siempre teniendo en cuenta una de las máximas de UML: **Una imagen vale más que mil palabras**.
- Como **principal desventaja** ampliamente mencionada de UML podemos nombrar el hecho de que se trata de un lenguaje muy amplio, haciendo, en ocasiones, complicado utilizar todas las posibilidades que ofrece. No obstante, los analistas tienden a utilizar los diagramas de forma sencilla, consiguiendo que sean entendidos fácilmente por cualquier persona que accedan a ellos.

# 1. UML: Unified Modeling Language

## 1.2.2. ¿Por qué UML?

- Los modelos o diagramas de UML  nos ayudan a trabajar a un mayor nivel de abstracción. Permite modelar cualquier tipo de aplicación corriendo en cualquier combinación de hardware y software, sistema operativo, lenguaje de programación y red, es decir, UML es independiente de la plataforma hardware sobre la que actúa el software. Su flexibilidad permite modelar cualquier tipo de aplicación e, incluso, otros tipos de proyecto que no son puramente software.
- UML ofrece ese modelado utilizando diagramas y se denomina lenguaje por ser una forma común de expresarse por todos los analistas, desarrolladores y usuarios. Está desarrollado para ayudar a todos estos (y más) perfiles a especificar, visualizar, construir y documentar todos los componentes de un proyecto.

# 1. UML: Unified Modeling Language

## 1.2.2. ¿Por qué UML?

A pesar de que cada diagrama UML en particular aporta su visión particular al modelado, el lenguaje en su conjunto tiene algunas características que interesa resaltar:

- Es muy **sencillo**. Pese a que si es usado de forma completa puede llegar a complicarse, lo normal es que se simplifique.
- Es capaz de modelar **todo tipo de sistemas**.
- Es un lenguaje **universal**, haciendo que todos los miembros del equipo se relacionen a través de sus diagramas sean del ámbito que sean.
- Es fácilmente **extensible**. Tiene mecanismos sencillos para especializar los conceptos fundamentales.
- Es **visual** y, por lo tanto, intuitivo.
- Es **independiente** del desarrollo, del lenguaje y de la plataforma.
- Bien ejecutado aporta un conjunto considerable de **buenas prácticas**.
- **No está completo**. Utilizando los distintos diagramas no podemos estar seguros de comprender con totalidad el sistema que va a desarrollarse. Los diagramas, para facilitar su comprensión pueden (y suelen) omitir información, pueden tener partes que se entienden de distintas maneras o, incluso, pueden tener conceptos que no pueden ser representados por ningún diagrama.

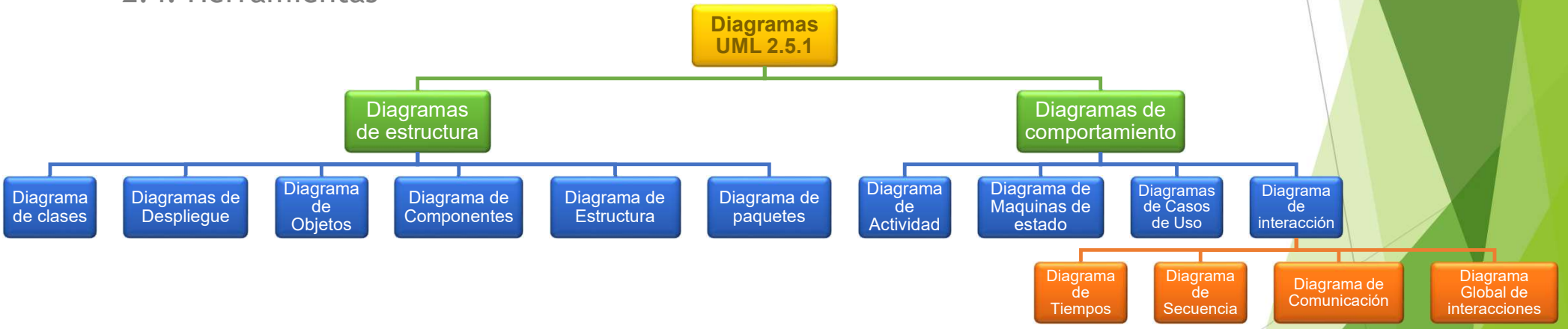
# 1. UML: Unified Modeling Language

## 1. 3. Breve historia del UML

- En un inicio, varios autores utilizan lenguajes de modelado (Booch, OMT, etc.).
- En 1995 se unifican en la versión 0.8 de UML.
- En 1997 UML se modifica con la experiencia obtenida en su utilización en empresas, lo que permite la creación de la versión **1.0.**
- En 1998, UML se aprueba por el OMG (Object Management Group), donde pasa a englobarse desde ese momento.
- Actualmente la última versión es la 2.5.1

# 2. Tipos de diagramas UML

2.1 Diagramas estructurales
2.2 Diagramas de comportamiento
2.3. Alternativas a UML
2.4. Herramientas



# 2. Tipos de diagramas UML

## 2.1. Diagramas de Estructura

- Los diagramas estructurales muestran la estructura estática del sistema y sus partes en diferentes niveles de abstracción.
- Existen un total de **siete** tipos de diagramas de estructura.
- Nosotros nos centraremos únicamente en el más importante, **los diagramas de clases**

# 2. Tipos de diagramas UML

## 2.1. Diagramas de Estructura

### Diagrama de clases

- Muestra la estructura del sistema, subsistema o componente utilizando clases con sus características, restricciones y relaciones: asociaciones, generalizaciones, dependencias, etc.

### Diagrama de componentes

- Muestra componentes y dependencias entre ellos. Este tipo de diagramas se utiliza para el desarrollo basado en componentes (CDB), para describir sistemas con arquitectura orientada a servicios (SOA).

### Diagrama de despliegue

- Muestra la arquitectura del sistema como despliegue (distribución) de artefactos de software.

### Diagrama de objetos

- Un gráfico de instancias, incluyendo objetos y valores de datos. Un diagrama de objeto estático es una instancia de un diagrama de clase; muestra una instantánea del estado detallado de un sistema en un punto en el tiempo.

### Diagrama de paquetes

- Muestra los paquetes y las relaciones entre los paquetes.

### Diagrama de perfiles

- Diagrama UML auxiliar que permite definir estereotipos personalizados, valores etiquetados y restricciones como un mecanismo de extensión ligero al estándar UML. Los perfiles permiten adaptar el metamodelo UML para diferentes plataformas o dominios.

### Diagrama de estructura compuesta

- Muestra la estructura interna (incluidas las partes y los conectores) de un clasificador estructurado.

# 2. Tipos de diagramas UML

## 2.2. Diagramas de Comportamiento

- A diferencia de los diagramas estructurales, los diagramas de comportamiento muestran cómo se comporta un sistema de información de forma dinámica. Es decir, describe los cambios que sufre un sistema a través del tiempo cuando está en ejecución.
- Hay un total de siete diagramas de comportamiento.
- Nosotros nos centraremos únicamente en los más importante, **los diagramas de caso de uso** y **los diagramas de secuencia.**

# 2. Tipos de diagramas UML

## 2.2. Diagramas de Comportamiento

### Diagrama de actividades

- Muestra la secuencia y las condiciones para coordinar los comportamientos de nivel inferior, en lugar de los clasificadores que poseen esos comportamientos. Estos son comúnmente llamados modelos de flujo de control y flujo de objetos.

### Diagrama de casos de uso

- Describe un conjunto de acciones (casos de uso) que algunos sistemas o sistemas (sujetos) deben o pueden realizar en colaboración con uno o más usuarios externos del sistema (actores) para proporcionar algunos resultados observables y valiosos a los actores u otros interesados del sistema(s).

### Diagrama de máquinas de estado

- Se utiliza para modelar el comportamiento discreto a través de transiciones de estados finitos. Además de expresar el comportamiento de una parte del sistema, las máquinas de estado también se pueden usar para expresar el protocolo de uso de parte de un sistema.

### Diagrama de interacción

- Es un subconjunto de los diagramas de comportamiento. Comprende los siguientes diagramas:

#### Diagrama de secuencia

- Es el tipo más común de diagramas de interacción y se centra en el intercambio de mensajes entre líneas de vida (objetos).

#### Diagrama tiempos

- Se centran en las condiciones que cambian dentro y entre las líneas de vida a lo largo de un eje de tiempo lineal.

#### Diagrama global de interacciones

- Los diagramas globales de interacciones brindan una descripción general del flujo de control donde los nodos del flujo son interacciones o usos de interacción.

# 2. Tipos de diagramas UML

## 2.3.Alternativas a UML

- Si bien UML es el estándar más utilizado y reconocido existen algunas alternativas que se han desarrollado para abordar diferentes enfoques o necesidades específicas en el modelado de sistemas.
- Estas alternativas pueden ser utilizadas en situaciones donde UML pueda resultar limitado o donde se necesite un enfoque más especializado para el modelado de sistemas. Sin embargo, es importante destacar que UML sigue siendo el estándar predominante y ampliamente aceptado para el modelado de sistemas de software debido a su versatilidad y amplia gama de diagramas para representar diferentes aspectos de un sistema.

| SysML (Systems Modeling Language): | BPMN (Business Process Model and Notation): | ERD (Entity-Relationship Diagrams): | Archimate: | Flowchart (Diagrama de Flujo): | DSL (Domain-Specific Languages): |
|---|---|---|---|---|---|
| • Diseñado para el modelado de sistemas de ingeniería y sistemas físicos, es una extensión de UML que se centra en el modelado de sistemas complejos. | • Especializado en modelar procesos de negocios y flujos de trabajo. Aunque no es un reemplazo directo de UML, se utiliza comúnmente en conjunto con él para representar aspectos de procesos de negocio. | • Utilizado principalmente en el modelado de datos y bases de datos. Si bien no reemplaza a UML, se enfoca en la representación de relaciones entre entidades y atributos en un contexto de bases de datos. Puedes aprender más sobre este diagrama a través de esta entrada en el blog. | •Un estándar de modelado de arquitectura empresarial que se enfoca en la representación de la arquitectura y la infraestructura empresarial, incluyendo aspectos como procesos, aplicaciones, estructuras de datos, etc. | •Aunque más simple en comparación con UML, los diagramas de flujo son útiles para representar algoritmos, flujos de trabajo simples y procesos de toma de decisiones. | •Estos son lenguajes de modelado diseñados específicamente para un dominio particular o un problema específico. A menudo, se utilizan para representar conceptos y abstracciones en un dominio específico de manera más precisa que UML. |

# 2. Tipos de diagramas UML

## 2.4.Herramientas para construir diagramas UML (online)

- La demanda de colaboración en línea combinado con las nuevas increíbles interfaces web que pueden crearse con las nuevas tecnologías (como HTML5) ha creado un nuevo mercado en rápido crecimiento de herramientas para **construir diagramas UML online gratis** y, también, de pago. De hecho, muchas de estas herramientas son más herramientas de dibujo que herramientas de modelado, pero, al menos, nos ofrecen la utilidad de dibujar y compartir nuestros modelos a través de Internet.

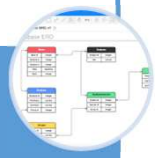
- Existen en la actualidad muchas herramientas (y más que siguen entrando al mercado) con una amplia variedad de características diferentes. Esta lista incluye exclusivamente herramientas online, desechando bibliotecas para entornos de desarrollos u otros tipos de herramienta, pero también existen otras herramientas de escritorio

# 2. Tipos de diagramas UML

## 2.4.Herramientas para construir diagramas UML (online)

### Gliffy

- Con un fuerte énfasis en los aspectos de control colaborativo y de revisión, Gliffy afirma ser «la aplicación de diagramación en línea más utilizada».
- Gliffy admite todos los diagramas UML junto con una variedad de otros tipos de diagramas, incluido un soporte sólido para los modelos de procesos BPMN.
- Viene con complementos para Confluence y Jira, así que claramente se integra muy bien con estas herramientas. Eso es importante si su equipo los usa y desea integrar sus modelos en el resto de su discusión / proceso de desarrollo.



### GenMyModel

- GenMyModel comenzó como una herramienta de modelado únicamente para UML, pero desde hace un tiempo se ha expandido para cubrir también el modelado de negocios con soporte de Archimate y BPMN.
- Ofrece un repositorio de modelos centralizado para equipos (con administración de derechos de acceso) que permite una colaboración fácil y simultáneamente de modelos.
- Al contrario de otras herramientas en esta lista, esta es más una herramienta de modelado que una herramienta de dibujo (con sus ventajas y desventajas). Como tal, una distinción clave de GenMyModel es su soporte para exportar modelos como XMI (el formato “estándar” para intercambiar modelos ) y sus capacidades de generación de código.

### Draw.io

- Destaca por su sencillez. Simplemente accediendo a draw.io en su navegador tienes un lienzo vacío para comenzar a dibujar.
- Viene con formas para modelado UML básico, ER y BPMN.
- Aun así, es un claro ejemplo de una herramienta que no entiende realmente la semántica de lo que estás dibujando, así que básicamente puedes hacer lo que quieras y utilizarlo para construir tus diagramas uml online gratis (aunque no cumplen a rajatabla las normas)
- También falla en el aspecto de colaboración, pero se integra bien con Google Drive, Dropbox, OneDrive y otros para guardar automáticamente los modelos en su ubicación preferida.

### Lucidchart

- Basado en HTML5.
- Con soporte UML y capacidades de colaboración en tiempo real.
- Puede importar archivos de Visio, lo que es definitivamente una ventaja, ya que una gran base de usuarios para estas herramientas parece provenir de entornos de modelado más similares a los de un escritorio, como esta herramienta.
- Más allá de UML, Lucidchart también viene con plantillas de dibujo para ER, modelado de procesos, diagramas de red y más.

# 2. Tipos de diagramas UML

## 2.4.Herramientas para construir diagramas UML (online)

### Creately

- Diagrama de colaboración en tiempo real.
- Con más de 50 tipos de diagramas y miles de ejemplos para comenzar.
- También se puede utilizar para trabajar fuera de línea y sincronizar su trabajo después.
- El único pero que tiene es que, ahora mismo, requiere Flash instalado para funcionar.

### Cacoo

- Su finalidad es crear «un dibujo» y no un modelado.
- Aún así, tiene algunas características interesantes, como la posibilidad de almacenar múltiples versiones de su diagrama para rastrear los cambios y la capacidad de publicar comentarios para enviar comentarios directamente sobre los diagramas.
- Cacoo admite los diagramas de casos de uso, secuencia, clase, actividad y máquina de estado.

### UMLetino

- Una herramienta UML en línea sencilla pero gratuita para crear diagramas UML de forma rápida.
- Se ejecuta en el navegador, y, por lo tanto, no requiere ninguna instalación.
- UMLetino se basa en UMLet (disponible como herramienta independiente o complemento de Eclipse).
- Los diagramas se pueden exportar como archivos XML o de imagen.
- A pesar de sus limitaciones (en comparación con otras herramientas), si estas buscando una herramienta en línea UML gratuita y fácil de usar, Esta es definitivamente una opción a considerar.

### Diagramo

- Se trata de un software de dibujo en línea HTML5 puro y de código abierto (no se requieren Flash, Java u otros complementos).
- Bajo licencia GPL.
- Fácil de descargar e instalar en tu propio servidor si así lo prefieres.
- El único «pero» es que, como un software centrado en diagramas de flujo, solo cubre un tipo de diagramas UML: máquinas de estado.
- Tampoco dispone de colaboración síncrona.
- Aun así sigue siendo una buena alternativa para construir tus diagramas uml online free.
- Los diagramas que se realizan utilizando esta web solo pueden ser guardadas como imágenes.

# 3. ELABORACIÓN DE DIAGRAMAS DE CLASES

## CONTENIDOS

# 3. ELABORACIÓN DE DIAGRAMAS DE CLASES

## 3.1. Introducción

El diagrama de clases es uno de los diagramas de estructura y se utiliza para representar los elementos que componen un sistema de información desde un punto de vista estático.

Se trata de un diagrama puramente orientado al modelo de programación orientado a objetos (POO).

Similar (salvando las distancias) al diagrama del modelo Entidad-Relación, el cual no es recogido por UML → Representa los datos y su interacción.

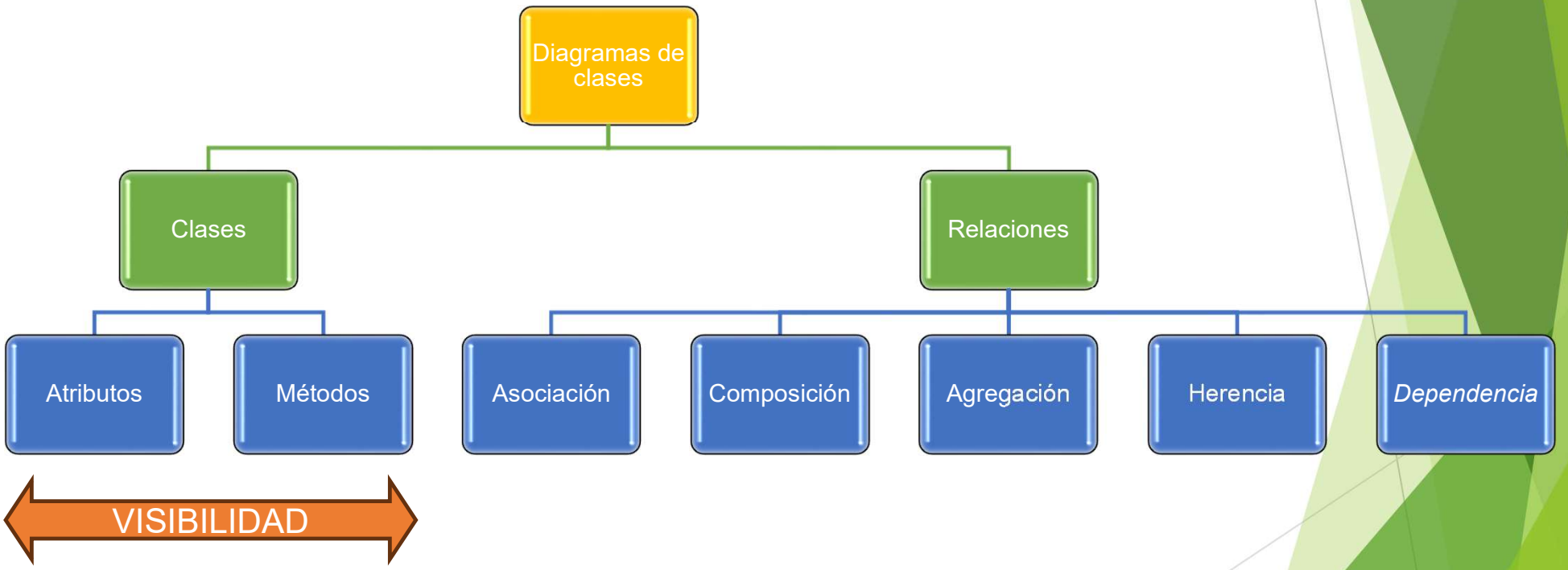
Ambos modelos muestran el modelo lógico de los datos de un sistema.

El diagrama de clases nos permite visualizar las clases que interactúan en el sistema y la relación entre ellas.

Permite especificar los atributos y métodos de la clase, indicando si son públicos, privados, etx (visibilidad de las clases).

# 3. ELABORACIÓN DE DIAGRAMAS DE CLASES

## 3.2. Elementos de un diagrama de clases



# 3. ELABORACIÓN DE DIAGRAMAS DE CLASES

## 3.2. Elementos de un diagrama de clases

### 3.2.1 Clases

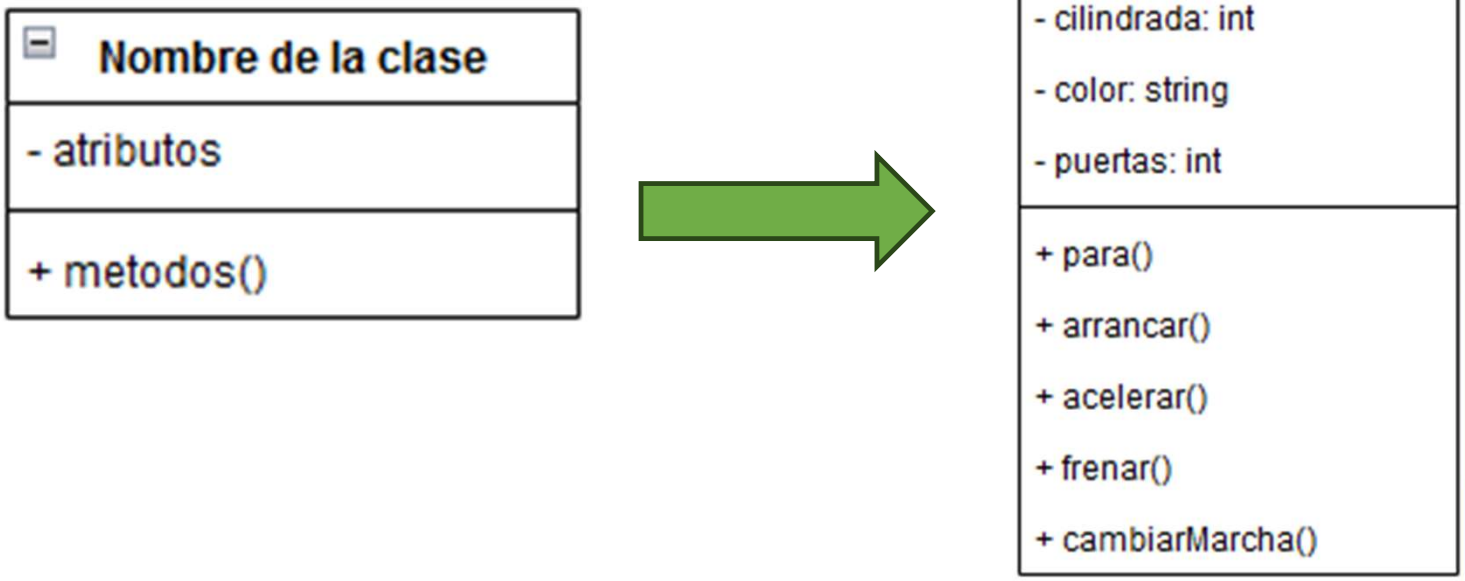
- En la POO, una **clase** representa una plantilla, basada en un elemento de la vida real, que toma sus propiedades y que realiza sus mismas acciones, de la cual se pueden derivar copias, a modo de clones.
- De esta manera, cada copia de la clase, será un **objeto** independiente con las mismas características que la clase y que puede realizar las mismas acciones.
- Se utilizan las clases para modelizar elementos de la vida real y así poderlos añadir a un programa.

# 3. ELABORACIÓN DE DIAGRAMAS DE CLASES

## 3.2. Elementos de un diagrama de clases

### 3.2.1 Clases

**Representación gráfica**



# 3. ELABORACIÓN DE DIAGRAMAS DE CLASES

## 3.2. Elementos de un diagrama de clases

### 3.2.1 Clases

**Visibilidad**

- **Pública (+)** → Representa que se puede acceder al atributo o función desde cualquier lugar de la aplicación.
- **Privada (-)** → Representa que se puede acceder al atributo o función únicamente desde la misma clase.
- **Protegida (#)** → Representa que el atributo o función puede ser accedida únicamente desde la misma clase o desde las clases que hereden de ella (clases derivadas).

| | |
|---|---|
| + | Público |
| - | Privado |
| # | Protegido |
| / | Derivado (se puede combinar con otro) |
| ~ | Paquete |

# 3. ELABORACIÓN DE DIAGRAMAS DE CLASES

## 3.2. Elementos de un diagrama de clases

### 3.2.1 Clases

Los ***objetos*** se suelen representar con los valores concretos:

| Coche1: Vehículo |
| --- |
| marca = Mazda CX30<br>cilindrada = 1200 cc<br>color = rojo<br>puertas = 5 |

El nombre del objeto va subrayado e indicando la **clase** que lo ha generado.

# 3. ELABORACIÓN DE DIAGRAMAS DE CLASES

## 3.2. Elementos de un diagrama de clases

### 3.2.2 Identificación de clases

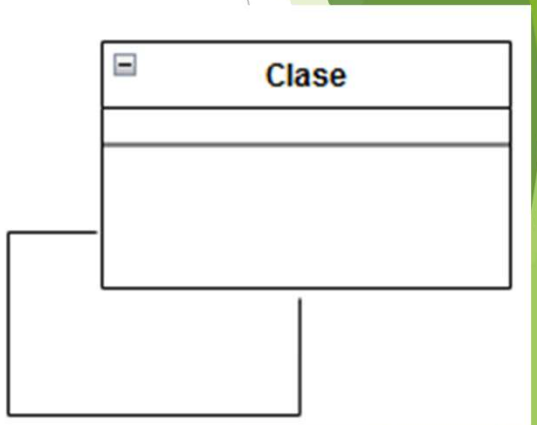
- La manera más rápida para encontrar clases sobre un enunciado, sobre una idea de negocio o, en general, sobre un tema concreto es buscar los **sustantivos** que aparecen en el mismo.
- Por poner algún ejemplo, algunas clases podrían ser: Animal, Persona, Mensaje, Expediente…
- Es un concepto muy amplio y **resulta fundamental identificar de forma efectiva estas clases**, en caso de no hacerlo correctamente se obtendrán una serie de problemas en etapas posteriores, teniendo que volver a hacer el análisis y perdiendo parte o todo el trabajo que se ha hecho hasta ese momento.

# 3. ELABORACIÓN DE DIAGRAMAS DE CLASES

## 3.2. Elementos de un diagrama de clases

### 3.2.3 Relaciones

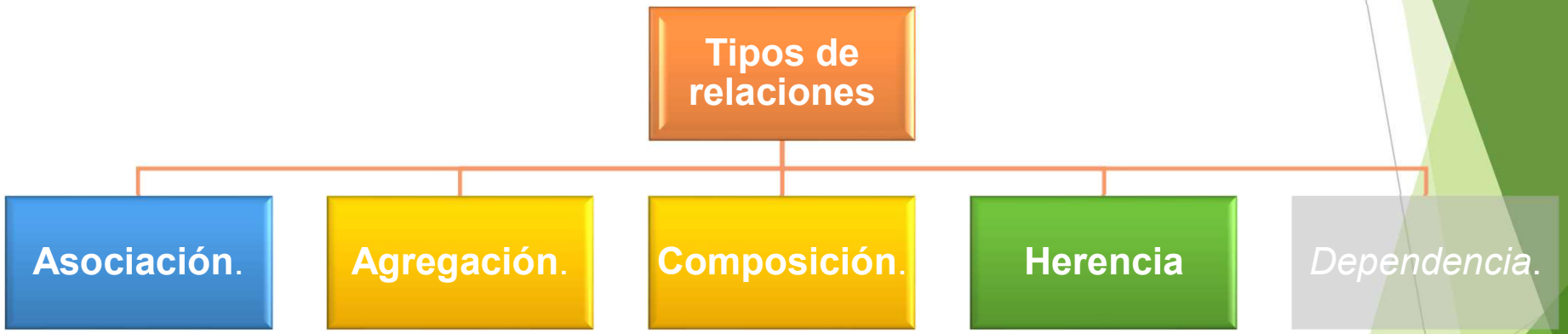
- Una relación i**dentifica una dependencia**. Esta dependencia puede ser entre dos o más clases (más común) o una clase hacía sí misma (menos común, pero existen), este último tipo de dependencia se denomina *dependencia reflexiva*.
- Las relaciones se representan con una línea que une las clases, esta línea variará dependiendo del tipo de relación.



# 3. ELABORACIÓN DE DIAGRAMAS DE CLASES

## 3.2. Elementos de un diagrama de clases

### 3.2.3 Relaciones

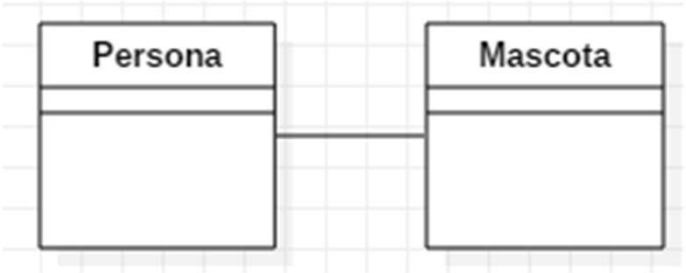


# 3. ELABORACIÓN DE DIAGRAMAS DE CLASES

## 3.2. Elementos de un diagrama de clases

### 3.2.3 Relaciones


Este tipo de relación es el más común y se utiliza para representar dependencia semántica. Se representa con una simple línea continua que une las clases que están incluidas en la asociación.
Un ejemplo de asociación podría ser: «Una mascota pertenece a una persona».



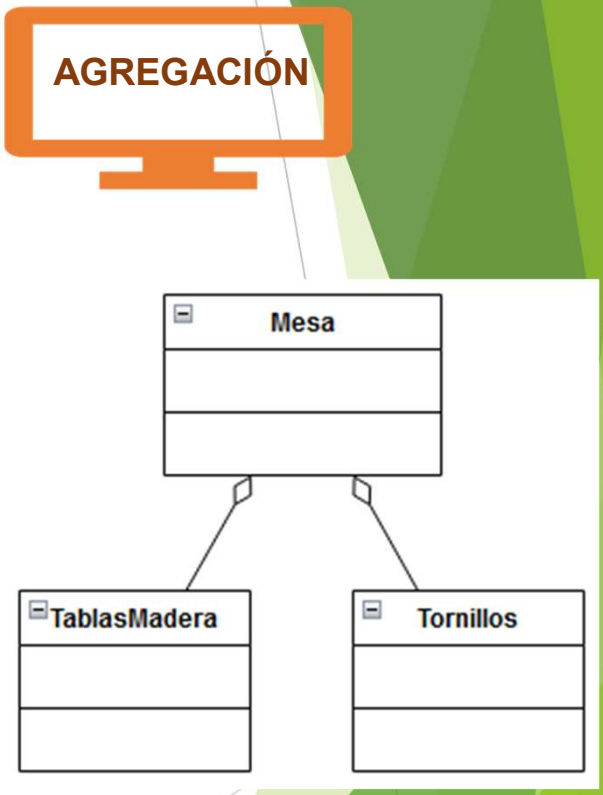

Ejemplo de asociación

# 3. ELABORACIÓN DE DIAGRAMAS DE CLASES

## 3.2. Elementos de un diagrama de clases

### 3.2.3 Relaciones

- Es una representación jerárquica que representa relaciones en las que un objeto es parte de otro, pero aun así debe tener existencia en sí mismo.
- Se representa con una línea que tiene un rombo en la parte de la clase que es una agregación de la otra clase (es decir, en la clase que contiene las otras).
- Un ejemplo de esta relación podría ser: «Las mesas están formadas por *tablas de madera* y *tornillos* o, dicho de otra manera, los tornillos y las tablas forman parte de una mesa». Como ves, el tornillo podría formar parte de más objetos, por lo que interesa especialmente su abstracción en otra clase.



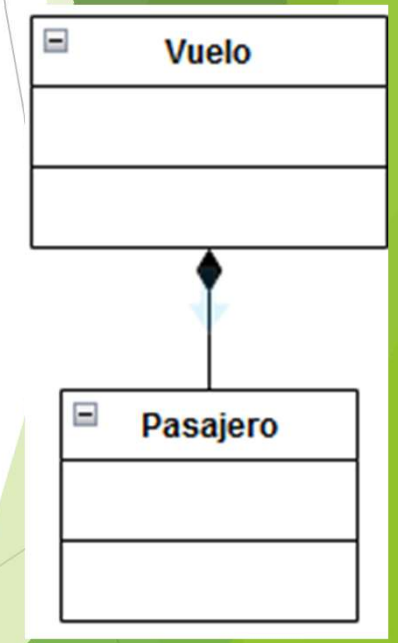
# 3. ELABORACIÓN DE DIAGRAMAS DE CLASES

## 3.2. Elementos de un diagrama de clases

### 3.2.3 Relaciones

**COMPOSICIÓN**

- Es similar a la agregación, representa una **relación jerárquica entre un objeto y las partes que lo componen**, **pero de una forma más fuerte**. En este caso, cuando el elemento que contiene los otros desaparece, deben desaparecer todos ya que no tienen sentido por sí mismos, sino que dependen del elemento que componen. Los componentes no se comparten entre varios elementos, esta es otra de las diferencias con la agregación.
- Se representa con una línea continua con un rombo relleno en la clase que es compuesta.
- Un ejemplo de esta relación sería: «Un vuelo de una compañía aérea está compuesto por pasajeros, que es lo mismo que decir que un pasajero está asignado a un vuelo»



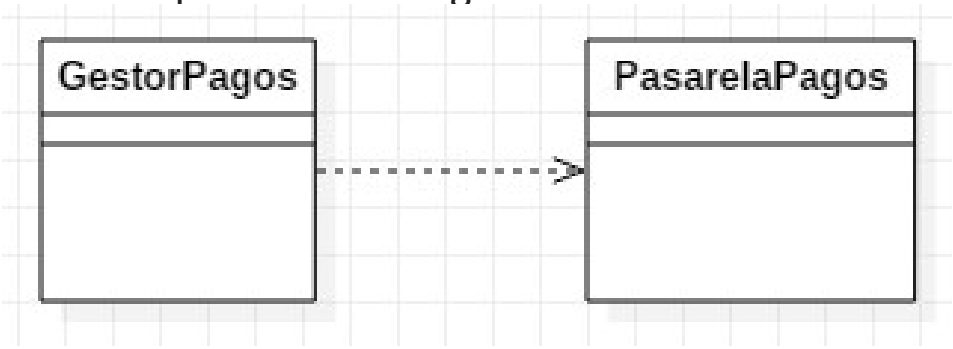
# 3. ELABORACIÓN DE DIAGRAMAS DE CLASES

## 3.2. Elementos de un diagrama de clases

### 3.2.3 Relaciones

**DEPENDENCIA**

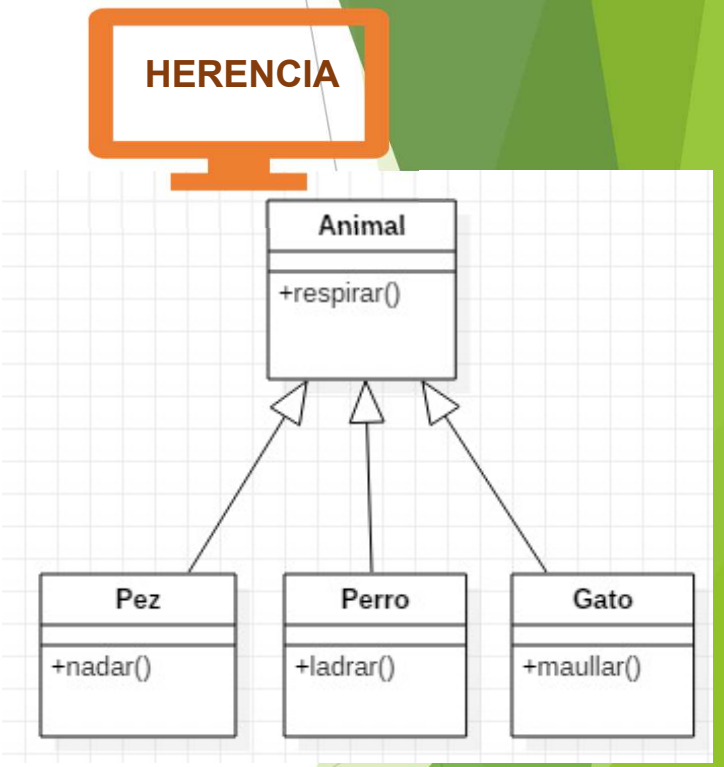
- Se utiliza este tipo de relación para representar que una clase requiere de otra para ofrecer sus funcionalidades. Es muy sencilla y se representa con una flecha discontinua que va desde la clase que necesita la utilidad de la otra flecha hasta esta misma.
- Un ejemplo de esta relación podría ser la siguiente:



# 3. ELABORACIÓN DE DIAGRAMAS DE CLASES

## 3.2. Elementos de un diagrama de clases

### 3.2.3 Relaciones




- Otra relación muy común en el diagrama de clases es la herencia. Este tipo de relaciones permiten que **una clase (clase hija o subclase) reciba los atributos y métodos de otra clase (clase padre o superclase)**. Estos atributos y métodos recibidos se suman a los que la clase tiene por sí misma. Se utiliza en relaciones «es un».
- Un ejemplo de esta relación podría ser la siguiente: Un pez, un perro y un gato son animales.
- En este ejemplo, las tres clases (Pez, Perro, Gato) podrán utilizar la función respirar, ya que lo heredan de la clase animal, pero solamente la clase Pez podrá nadar, la clase Perro ladrar y la clase Gato maullar. La clase Animal podría plantearse ser definida abstracta, aunque no es necesario.

# 3. ELABORACIÓN DE DIAGRAMAS DE CLASES

## 3.2. Elementos de un diagrama de clases

### 3.2.4 Multiplicidad

Determina cuántos objetos de cada tipo intervienen en la relación, es decir, **el número de instancias de una clase que se relacionan con UNA instancia de la otra clase.**

- Cada asociación tiene dos multiplicidades (una para cada extremo de la relación).
- Para especificar la multiplicidad de una asociación hay que indicar la multiplicidad mínima y la multiplicidad máxima (mínima..máxima).

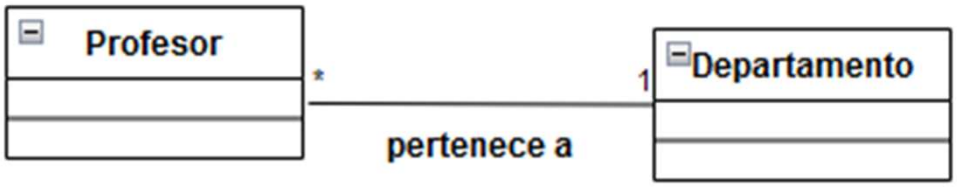
| MULTIPLICIDAD | SIGNIFICADO |
|---|---|
| 1 | Uno y sólo uno |
| 0..1 | Cero o uno |
| n..m | Desde n hasta m |
| * | Cero o varios (también sirve 0..*) |
| 1..* | Uno o varios (al menos 1) |

- Cuando la multiplicidad mínima es 0, la relación es opcional.
- Una multiplicidad mínima mayor o igual que 1 establece una relación obligatoria.

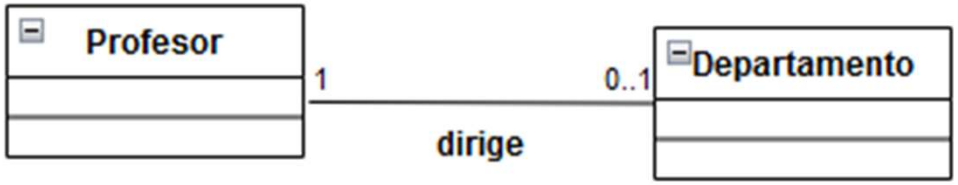
# 3. ELABORACIÓN DE DIAGRAMAS DE CLASES

## 3.2. Elementos de un diagrama de clases

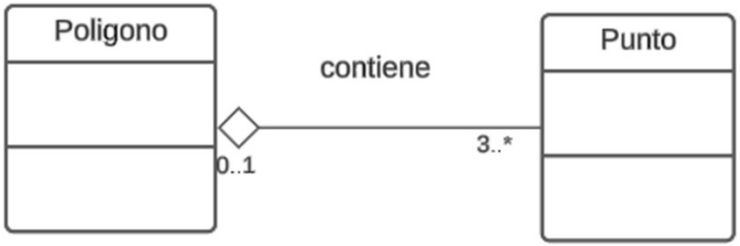
### 3.2.4 Multiplicidad




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




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




Un polígono está compuesto por al menos tres puntos. Y un punto está contenido en uno o en ningún polígono.

# 3. ELABORACIÓN DE DIAGRAMAS DE CLASES

## 3.3. Interfaces

- Una interfaz es una entidad que declara una **serie de atributos, funciones y obligaciones.** Es una especie de contrato donde toda instancia asociada a una interfaz debe de implementar los servicios que indica aquella interfaz.
- Dado que únicamente son declaraciones **no pueden ser instanciadas**.
- Las interfaces se asocian a clases. Una asociación entre una clase y una interfaz representa que esa clase cumple con el contrato que indica la interfaz, es decir, incluye aquellas funciones y atributos que indica la interfaz.
- Su representación es similar a las clases, pero indicando arriba la palabra <<interface>>.

# 3. ELABORACIÓN DE DIAGRAMAS DE CLASES

## 3.4. Cómo dibujar un diagrama de clases

Este tipo de diagramas son solicitados cuando se está describiendo la vista estática del sistema o sus funcionalidades. Unos pequeños pasos que puedes utilizar de guía para construir estos diagramas son los siguientes:

- **Identifica los nombres de las clases**. El primer paso es identificar los objetos primarios del sistema. Las clases suelen corresponder a sustantivos dentro del dominio del problema.
- **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.
- **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.

# 3. ELABORACIÓN DE DIAGRAMAS DE CLASES

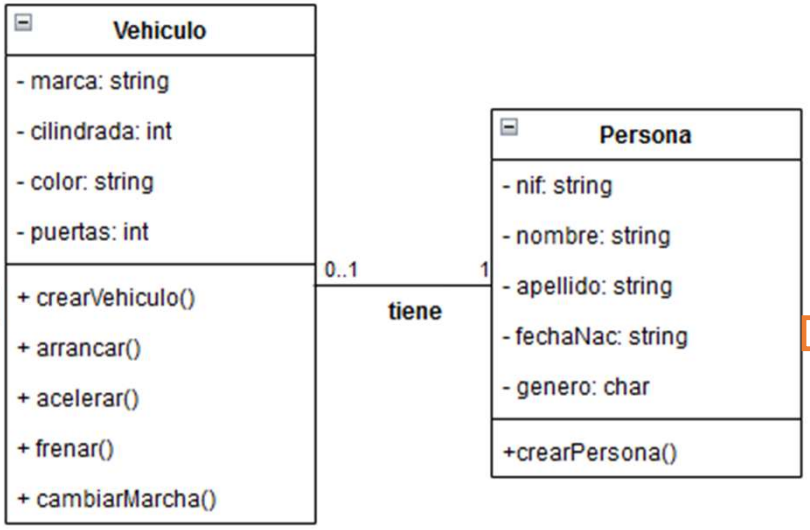
## 3.5. Buenas prácticas en el diseño del diagrama de clases

- Los diagramas de clase **pueden tender a volverse incoherentes** a medida que se expanden y crecen. Es mejor evitar la creación de diagramas grandes y **dividirlos** en otros más pequeños que se puedan vincular entre sí más adelante.
- Usando la notación de clase simple, puedes crear rápidamente **una visión general de alto nivel** de su sistema. Se puede crear un diagrama detallado por separado según sea necesario, e incluso vincularlo al primero para una referencia fácil.
- Cuantas más líneas se superpongan en sus diagramas de clase, más abarrotado se vuelve y, por tanto, más se complica utilizarlo. El lector se confundirá tratando de encontrar el camino. Asegúrate de que **no haya dos líneas cruzadas** entre sí, a no ser que no haya más remedio.
- Usa **colores** para agrupar módulos comunes. Diferentes colores en diferentes clases ayudan al lector a diferenciar entre los diversos grupos.

# 3. ELABORACIÓN DE DIAGRAMAS DE CLASES

## 3.6. Ejemplos de diagramas de clases

```
public class Vehiculo{
   private String marca;
   private int cilindrada;
   private String color;
   private int puertas
    …
   //métodos
    …
}
```

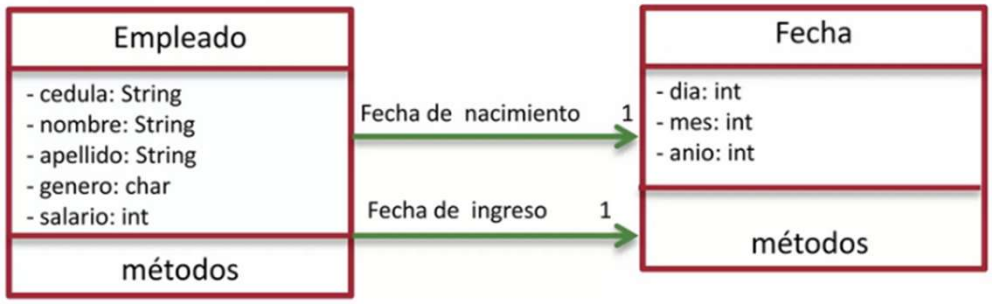



```
public class Persona{
   private String nif;
   private String nombre;
   private String apellido;
   private String fechaNac;
   private char genero;
   Private Vehículo vehículo;
    …
   //métodos
    …
}
```

La relación de asociación se convierte en atributo en la clase Persona.
El tipo será el nombre de la clase (Vehículo) con la que tiene la relación

# 3. ELABORACIÓN DE DIAGRAMAS DE CLASES

## 3.6. Ejemplos de diagramas de clases




```
public class Empleado{
   private String cedula;
   private String nombre;
   private String apellido;
   private char genero;
   private int salario;
   private Fecha fechaNac;
   private Fecha fechaIng;
}
```

```
public class Fecha{
   private int dia;
   private int mes;
   private int anio;
}
```

Atributos propios de la clase

Atributos que salen de las relaciones

# 3. ELABORACIÓN DE DIAGRAMAS DE CLASES

## 3.6. Ejemplos de diagramas de clases

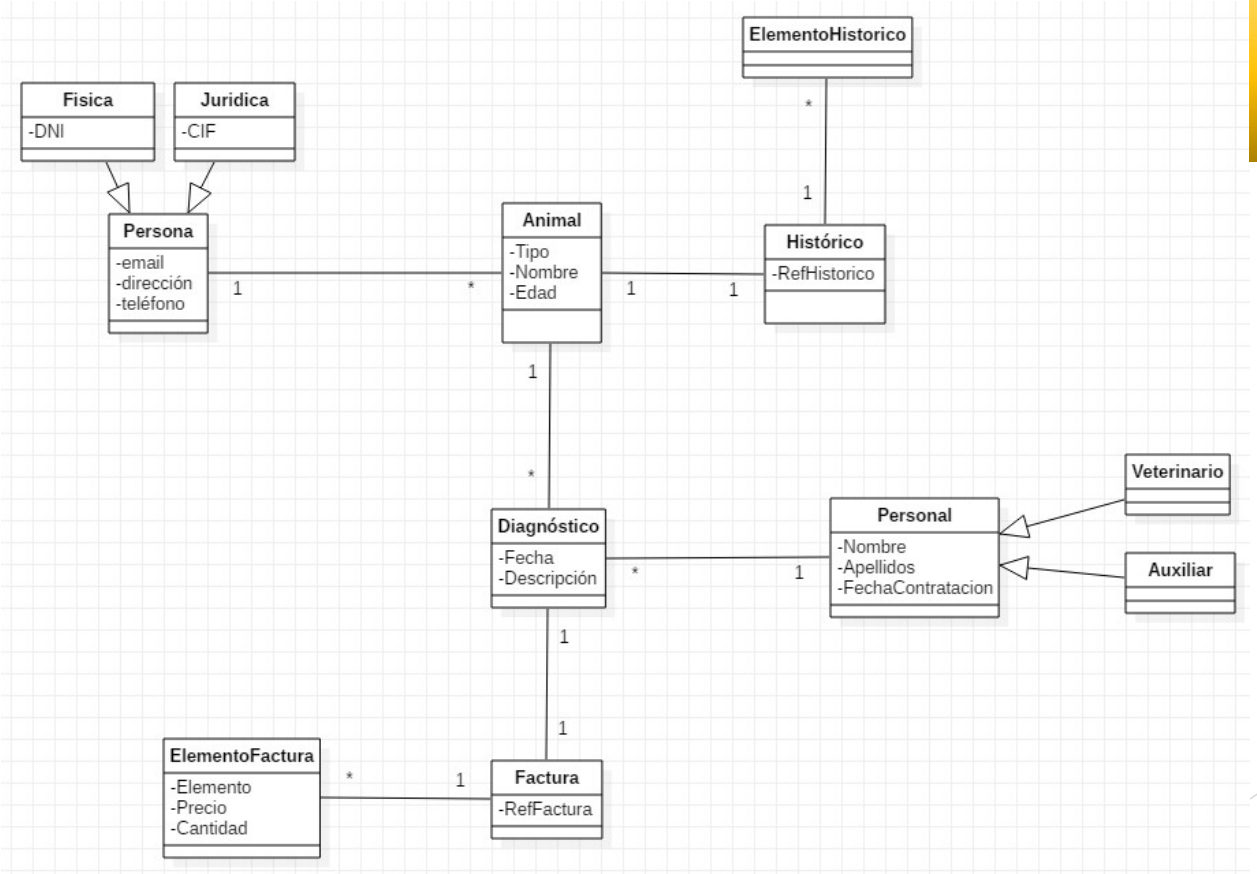
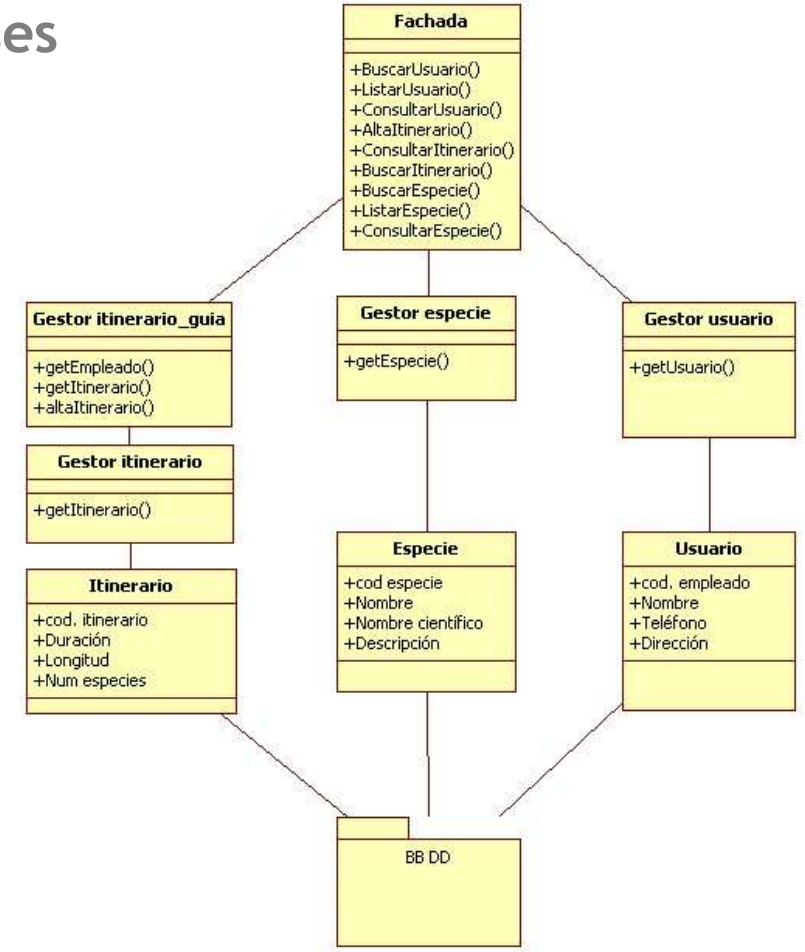



**Diagrama de clases de una clínica veterinaria**

# 3. ELABORACIÓN DE DIAGRAMAS DE CLASE

## 3.6. Ejemplos de diagramas de clases

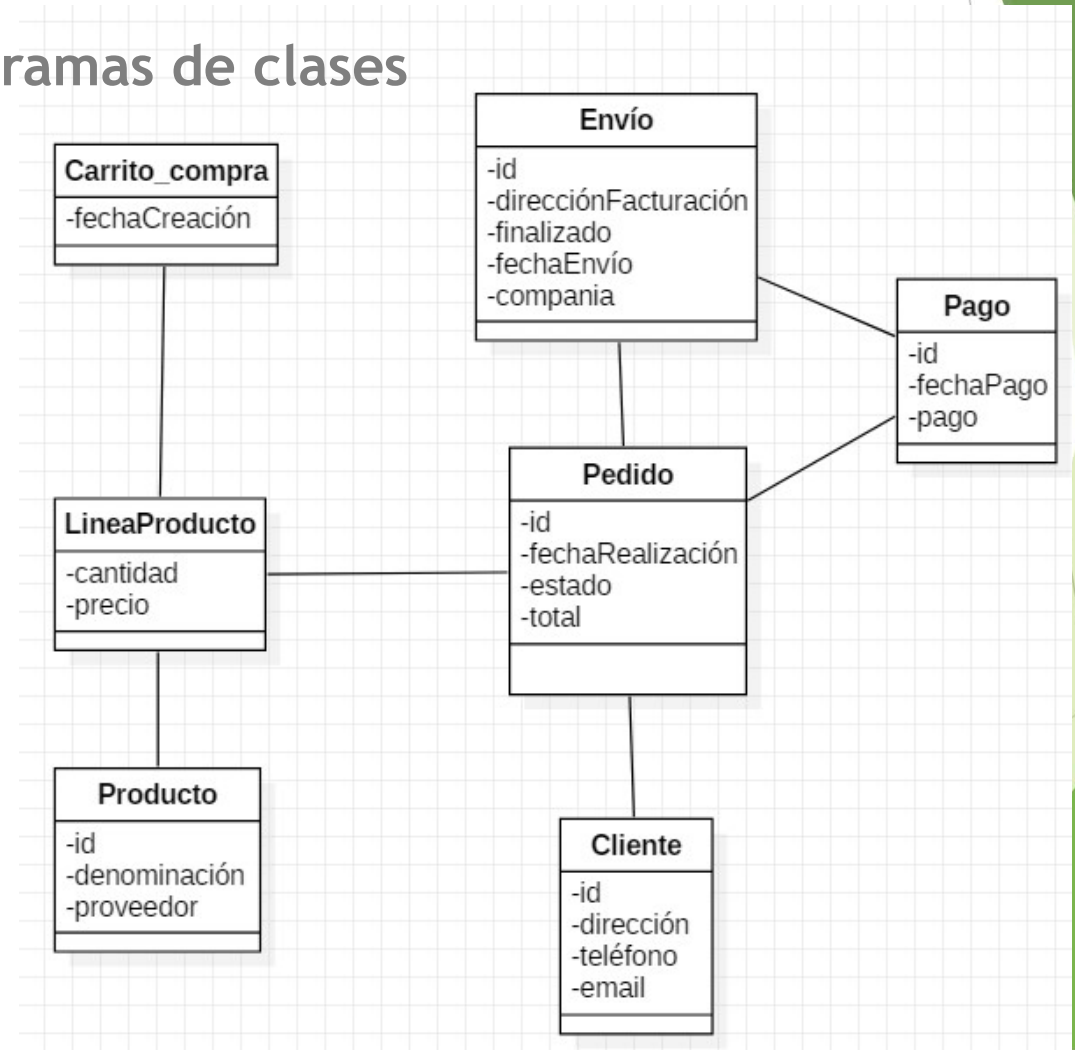
**Diagrama de clases de un zoológico**



# 3. ELABORACIÓN DE DIAGRAMAS DE CLASE

## 3.6. Ejemplos de diagramas de clases

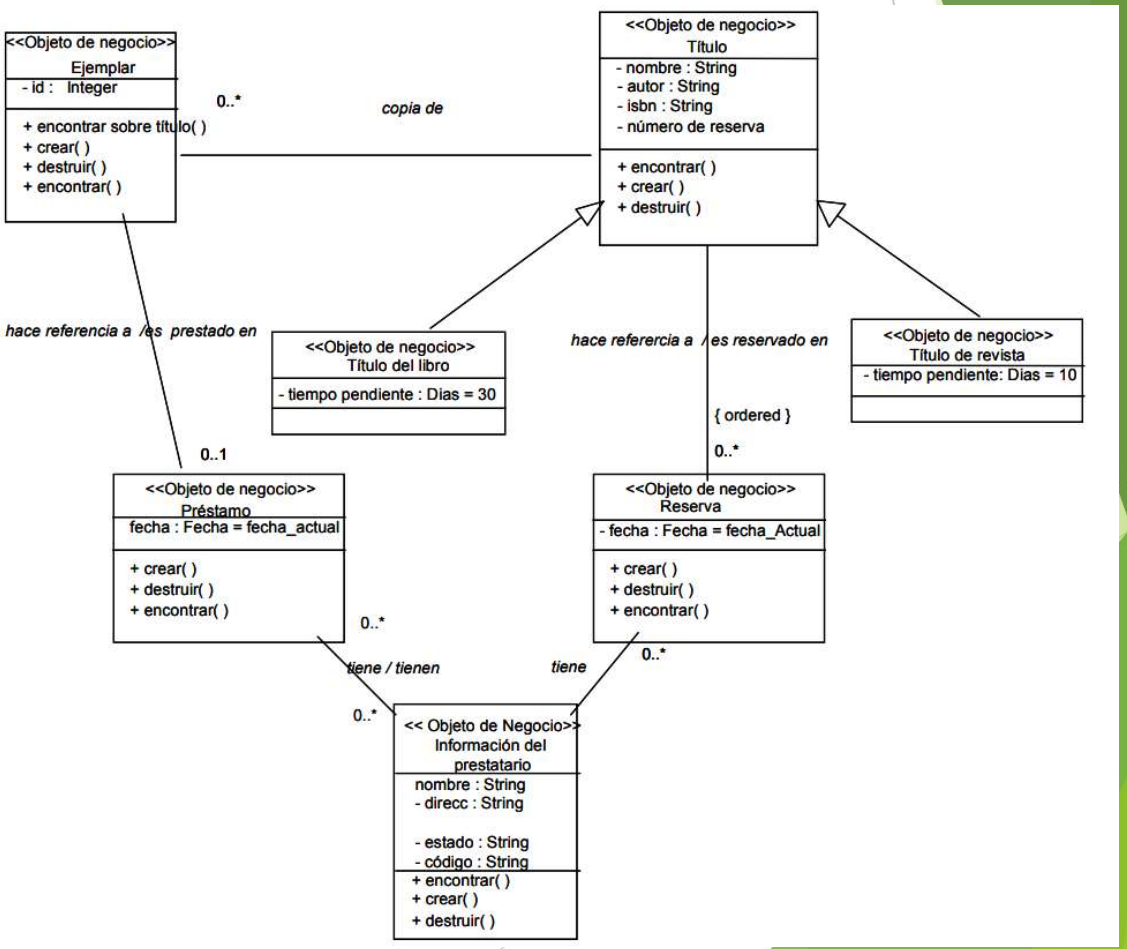
**Diagrama de clases de una tienda**



# 3. ELABORACIÓN DE DIAGRAMAS DE CLASE

## 3.6. Ejemplos de diagramas de clases

**Diagrama de clases de gestión de una biblioteca**



# Fin

Entornos de desarrollo

Mª Carmen Safont Richarte