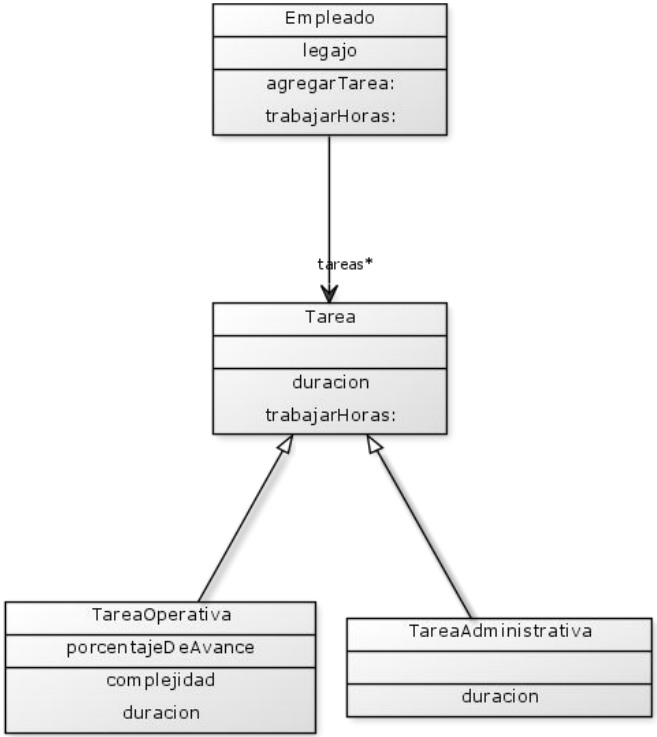
Tema 5

Elaboración de diagramas de clases

DAM 1 - Entornos de Desarrollo

Índice



Introducción

- **¿Qué es el diseño orientado a objetos?**

Un sistema que se entiende como un **conjunto de objetos** que tienen propiedades/atributos y comportamientos.

- **Vale... pero, ¿qué es un objeto?**

Un **objeto** consta de una estructura de datos y de una colección de métodos u operaciones que manipulan dichos datos. Los datos que definen un objeto son los **atributos**. Las **operaciones** definen los **comportamientos** del objeto y cambian el valor de uno o más atributos. Un objeto se comunica con otro a través del paso de **mensajes**.

- **¿Y qué es una clase?**

Una clase es una **plantilla** para crear objetos. Cuando se crea un objeto (**instancia**) se ha de especificar de qué clase es el objeto instanciado, que el compilador sepa de las características del objeto.

Conceptos orientados a objetos

El paradigma OO se basa en el concepto de **objeto**. Un **objeto** hemos dicho que es aquello que tiene estado, comportamiento e **identidad** (propiedad que lo distingue del resto de objetos). La estructura y comportamiento de objetos similares están definidos en su **clase común**. Teniendo en cuenta esto, una clase se puede definir como el conjunto de objetos que comparten estructura y comportamiento.

Clases y objetos son **conceptos similares**, pero existe una **diferencia** conceptual entre ambos. Las clases son un concepto estático definido en un programa, mientras que los objetos son entes dinámicos que existen en tiempo y espacio, y que ocupan memoria mientras se ejecuta el programa.

Ejemplo: un plano arquitectónico define las características de la casa (atributos) y cómo se comporta, pero pueden haber muchas casas iguales. Similar a los coches, todos se fabrican con los mismos componentes y tienen el mismo funcionamiento.

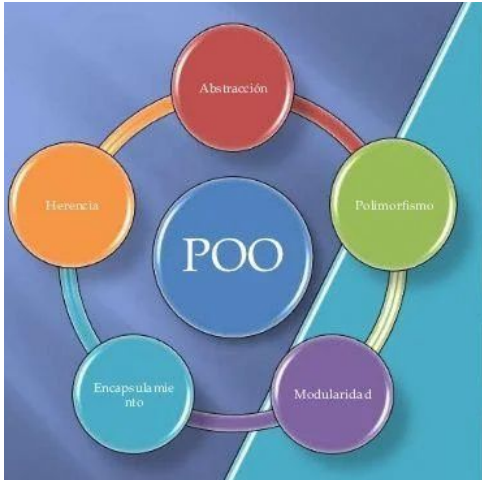
Conceptos orientados a objetos

Este modelo se basa en 5 principios:

- Abstracción
- Encapsulación
- Modularidad
- Jerarquía/herencia
- Polimorfismo

Y también , aunque en **menor medida** en estos 3:

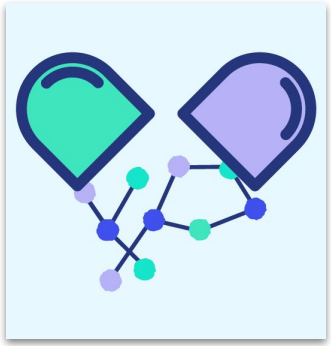
- Tipificación
- Concurrencia
- Persistencia



Conceptos orientados a objetos

- **Abstracción**

Denota las características esenciales de un objeto, donde se capturan sus comportamientos. Para ello es clave el proceso de análisis para poder llegar a componer un conjunto de clases que permitan modelar el problema.



- **Encapsulamiento**

Es el proceso de ocultar todos los detalles de un objeto que no contribuyen a esas características esenciales, es decir, separar el aspecto externo del interno. O lo que es lo mismo, ocultar atributos y métodos del objeto a otros objetos, una vez encapsulados pasan a ser atributos y métodos **privados**.

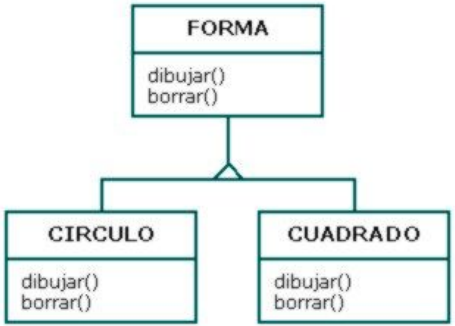
Conceptos orientados a objetos

- **Modularidad**

Propiedad de una aplicación que permite descomponer en un conjunto de módulos o partes más pequeñas coherentes e independientes. Estos se pueden compilar por separado, pero tienen cohesión con otros módulos.

- **Polimorfismo**

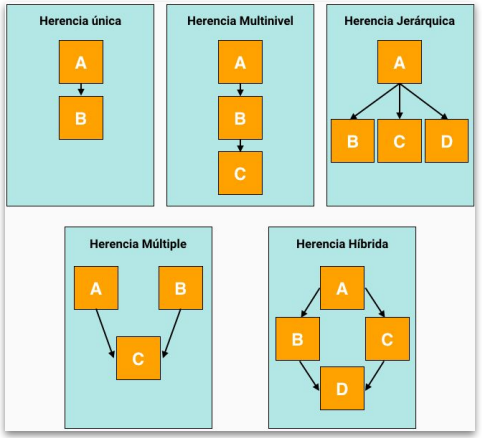
Consiste en reunir con el mismo nombre comportamientos diferentes. Es la propiedad por la cual un mismo mensaje puede originar distintas conductas al ser recibido por distintos objetos. Simplificando: dos objetos, de dos clases distintas, pueden responder a la llamada de un método con el mismo nombre, cada uno de ellos con un comportamiento encapsulado, pero que responden a una interfaz común (a través de la herencia).



Conceptos orientados a objetos

- **Herencia/jerarquía**

La programación OO permite extender las clases, produciendo clases que heredan todos el comportamiento y el código de la clase extendida. La clase original se denomina **clase padre, base o superclase**. La nueva clase se define como **clase hija, derivada o subclase**. La extensión de una clase se denomina herencia, porque la clase hija hereda todos los métodos y atributos de la clase padre que se extiende. Cada subclase estaría formada por un grupo de objetos más especializados, con características comunes que compartirán datos y operaciones. Los objetos heredan las propiedad y el comportamiento de todas las clases a las que pertenecen.



Conceptos orientados a objetos

- **Tipificación**

  Definición precisa de un objeto, de tal forma que objetos de diferentes tipos no pueden ser intercambiados.

- **Concurrencia**

  Es la propiedad que permite a varios objetos funcionar a la vez, siempre que estén activos. El objeto activo hace algo, esto se utiliza para la programación concurrente o **multihilo**.

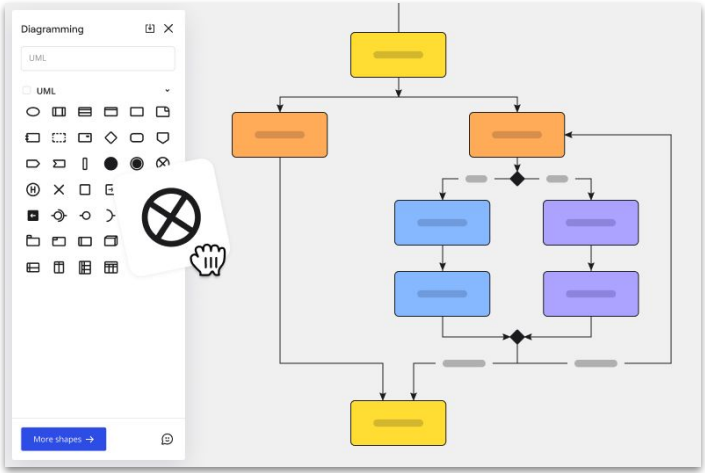
- **Persistencia**

  Es la propiedad de un objeto a través de la cual su existencia trasciende el tiempo y/o el espacio. Se refiere a objetos de clases asociadas a **bases de datos**.

¿Qué es UML?

El **UML (Unified Modeling Language o Lenguaje de Modelado Unificado)** es un lenguaje **gráfico** para visualizar, especificar y documentar cada una de las partes que comprende el desarrollo de software. Este lenguaje se puede utilizar para el modelado tanto sistemas de software, como de hardware, como organizaciones del mundo real.

Para ello utiliza una serie de **diagramas** en los que se representan los distintos puntos de vista de modelado. Podemos decir que es un lenguaje que se usa para **documentar**.



¿Qué es UML?

Un diagramas UML se compone de cuatro tipos de elementos:

- **Estructuras**: Son los nodos del grafo y definen el tipo de diagrama.
- **Relaciones**: Son los arcos del grafo que se establecen entre los elementos estructurales.
- **Notas**: Se representan como un cuadro donde podemos escribir comentarios que nos ayuden a entender algún concepto que queramos representar.
- **Agrupaciones**: Se utilizan cuando modelamos sistemas grandes para facilitar su desarrollo por bloques.

y se clasifican en:

- **Diagramas de estructura**: Representan la visión estática del sistema. Especifican clases y objetos y cómo se distribuyen físicamente en el sistema.
- **Diagramas de comportamiento**: muestran la conducta en tiempo de ejecución del sistema, tanto desde el punto de vista del sistema completo como de las instancias u objetos que lo integran. Dentro de este grupo están los **diagramas de interacción**.

En la imagen de la siguiente diapositiva aparecen todos los diagramas organizados según su categoría.

¿Qué es UML?

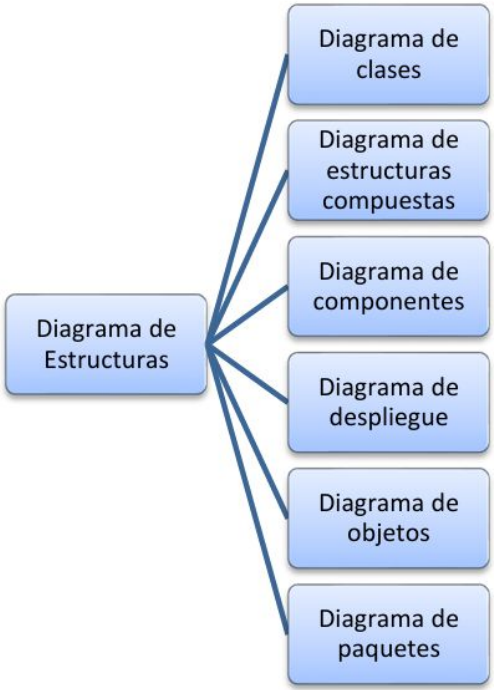


Diagrama de Estructuras
Diagrama de clases
Diagrama de estructuras compuestas
Diagrama de componentes
Diagrama de despliegue
Diagrama de objetos
Diagrama de paquetes

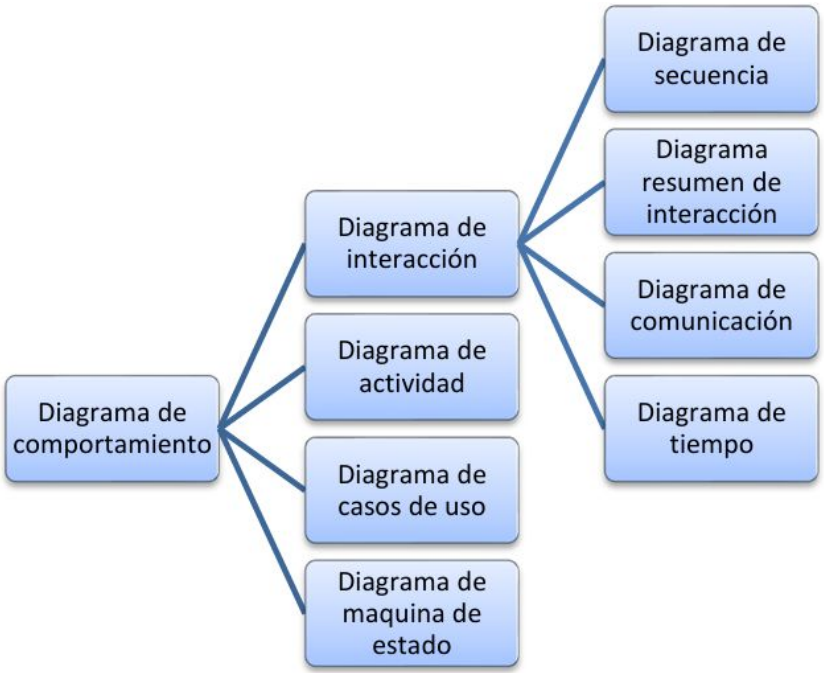


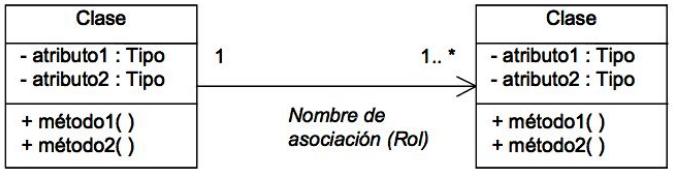
Diagrama de comportamiento
Diagrama de interacción
Diagrama de actividad
Diagrama de casos de uso
Diagrama de maquina de estado
Diagrama de secuencia
Diagrama resumen de interacción
Diagrama de comunicación
Diagrama de tiempo

Diagramas de clases

Un **diagrama de clases** es un tipo de diagrama de estructuras (estático) que describe la estructura de un sistema mostrando sus clases y asociaciones entre ellas. Sirve para visualizar las relaciones entre las clases que componen el sistema.

Un diagrama de clases está compuesto por:

- **Clases**: atributos, métodos y visibilidad.
- **Relaciones**: asociación, herencia, agregación, composición, realización y dependencia.



Diagramas de clases

- **Clases**

Las clases son la unidad básica que encapsula toda la información de un objeto. A través de ella podemos modelar el entorno en estudio (un empleado, un departamento, una cuenta bancaria...). En UML, una clase se representa como un rectángulo que posee tres divisiones:

- **Parte superior:** contiene el nombre de la clase.
- **Intermedio**: contiene los atributos (o variables) que caracterizan a la clase.
- **Parte inferior**: contiene los métodos u operaciones, los cuales son la forma como interactúa el objeto con su entorno.

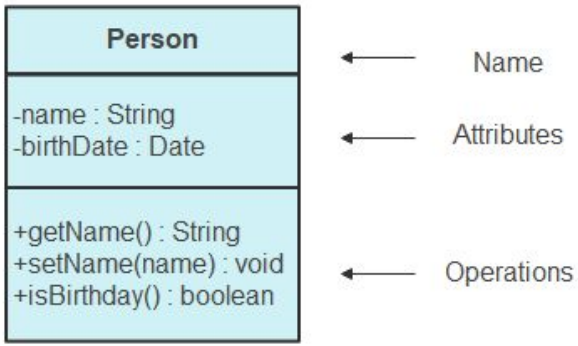
Diagramas de clases

- **Clases**

Un **atributo** representa alguna **propiedad** de la clase que se encuentra en todas las instancia de la clase. Los atributos pueden representarse solo mostrando su nombre, o mostrando su nombre y su tipo, en incluso valor por defecto.

Algunos ejemplos:

- nombre : String
- salario : Float
- ID : int



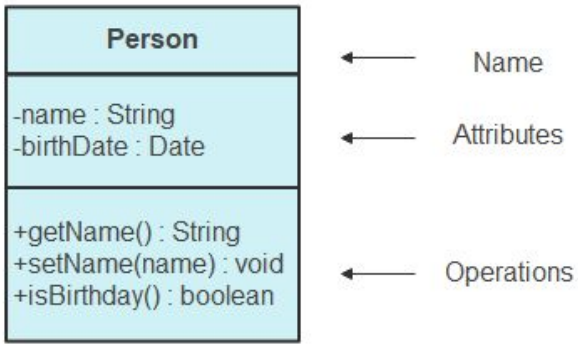
Diagramas de clases

- **Clases**

Un **método**, también denominado **operación**, es la implementación de un servicio de la clase que muestra un **comportamiento común** a todos los objetos. Definen como interactúa con el entorno.

Algunos ejemplos:

- calcularIVA() : int
- getID() : int
- getNombre() : String
- setNombre(nombre) : void



Diagramas de clases

- **Clases**

Tanto en los atributos como en los métodos tenemos que tener en cuenta **encapsulamiento** o **visibilidad** de estos.

Se distinguen 4 tipos:

- **Público (*public*):** se representa con “+”. El atributo o método es accesible desde **dentro** y **fuera** de la clase.
- **Privado (*private*):** se representa con “-”. El atributo o método es accesible **sólo** desde **dentro** de la clase.
- **Protegido (protected):** se representa con “#”. El atributo o método **no** es accesible desde **fuera** de la clase, pero sí por sus propios métodos, y los métodos de las **subclases** que la deriven.
- **Paquete (package):** se representa con “ ~ “. El atributo o método es visible para las clases del **mismo paquete**.

Diagramas de clases

- **Relaciones**

En la vida muchos objetos están vinculados o relacionados entre sí, los vínculos se corresponden con asociaciones entre los objetos, por ejemplo: **un empleado y la compañía para la que trabaja.** En UML, estos vínculos se describen mediante asociaciones, de igual modo que los objetos se describen mediante clases.

Las asociaciones tienen un nombre. y son el reflejo de los elementos de la asociación. También poseen una cardinalidad llamada **multiplicidad** que representa el número de instancias de una clase que se relacionan con las instancias de otra clase, esa multiplicidad es similar a la utilizada en el modelo Entidad/Relación.

Se sitúa en un extremo de una asociación indicando cuántas instancias de la clase situada en ese mismo extremo está vinculada una instancia de la clase del extremo opuesto.

Diagramas de clases

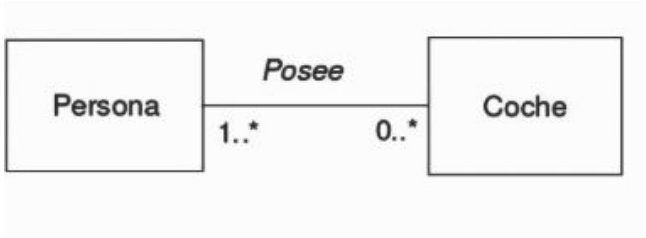
- **Relaciones**

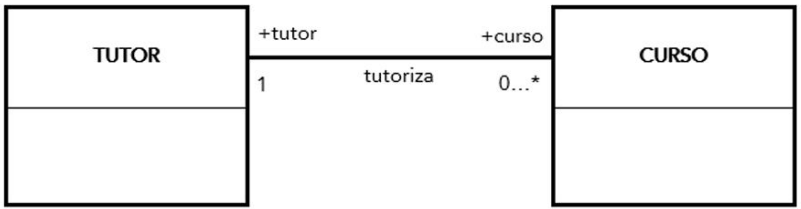
En cada extremo es posible indicar la multiplicidad mínima y máxima para indicar el intervalo de valores al que tendrá que pertenecer siempre la multiplicidad. Se utiliza la siguiente notación:

Notación	Cardinalidad / Multiplicidad
0..1	Cero o una instancia
1	Una instancia
*****	De cero a varias
1..*	De una a varias
M..N	Entre M y N
N	N instancias

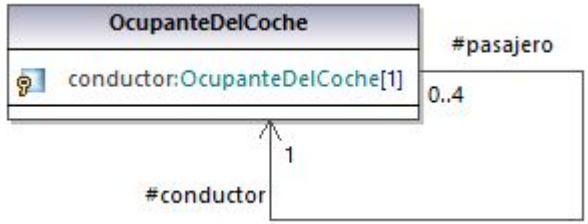
Diagramas de clases

- Relaciones





Diagramas de clases



- **Relaciones. Tipos.**

ASOCIACIÓN

Puede ser **bidireccional** o **unidireccional**, dependiendo de si ambas conocen la existencia la una de la otra o no.

Dentro de una relación de asociación, cada clase juega un **papel**.

Si se convierte a Java dos clases unidas con una asociación bidireccional, cada una de ellas tendrá un objeto o un set de objetos de la otra. En cambio, en la asociación unidireccional, la clase destino no sabe de la existencia de la otra.

La **navegabilidad** entre clases nos muestra que es posible pasar desde un objeto de la clase origen a uno o más objetos de la clase destino dependiendo de la multiplicidad.

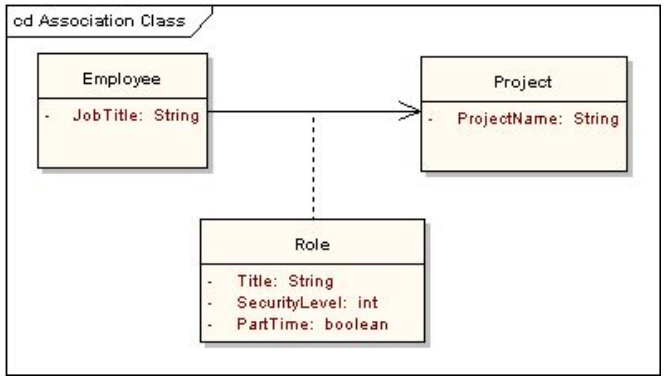
También existe la **asociación reflexiva**, que unen entre sí instancias de una misma clase.

Diagramas de clases

- **Relaciones. Tipos.**

CLASE ASOCIACIÓN

Una asociación entre clases puede llevar información necesaria para esa asociación, a esto se le llama **clase asociación**, es como una relación M:N con atributos del modelo entidad/relación. Estas clases puede estar dotadas de **atributos** y **métodos**.

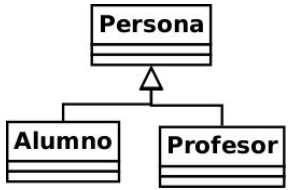


Diagramas de clases

Ejemplo 1. Realizar un diagrama de clases para representar las relaciones entre empleados y departamentos.

- Consideramos que un empleado trabaja en un departamento y en el departamento trabajan muchos empleados.
- Datos de los empleados: código, nombre, oficio, salario.
- Datos de los departamentos: código, nombre y localidad
- Además un empleado puede ser jefe de varios empleados.
- Se necesitan los métodos para asignar y devolver datos a los empleados y departamentos.

Diagramas de clases



- **Relaciones. Tipos.**

HERENCIA

Las clases se pueden organizar de **forma jerárquica**, mediante la **herencia**. La generalización define una relación entre una clase más general y una o más versiones refinadas de ella. Esta indica que una clase (subclase) hereda los atributos y métodos de otra (superclase). Por tanto, la **superclase generaliza** a las subclases, y las **subclases especializan** a la superclase.

Para representar esa asociación se utiliza una flecha, el extremo de la flecha apunta a la superclase.