ENTORNOS DE DESARROLLO UD-3

UML

DIAGRAMAS DE CLASES

ÍNDICE

[INTRODUCCIÓN 3](#_Toc127008096)

[CREACIÓN DE CLASES 3](#_Toc127008097)

[A. ATRIBUTOS 3](#_Toc127008098)

[B. MÉTODOS 4](#_Toc127008099)

[RELACIONES ENTRE CLASES 5](#_Toc127008100)

[A. CARDINALIDAD DE UNA RELACIÓN 5](#_Toc127008101)

[B. GENERALIZACIÓN (HERENCIA) 5](#_Toc127008102)

[C. AGREGACIÓN Y COMPOSICIÓN 6](#_Toc127008103)

[D. ATRIBUTOS DE ENLACE 7](#_Toc127008104)

[E. RESTRICCIONES 7](#_Toc127008105)

[PAUTAS PARA CREAR DIAGRAMAS DE CLASES 7](#_Toc127008106)

# INTRODUCCIÓN

Se considera el más importante de todos los existentes en UML y se puede encontrar lo siguiente:

* **Clases**: agrupan conjuntos de objetos con características comunes, que llamaremos atributos, y su comportamiento, que serán métodos. Los atributos y métodos tendrán una visibilidad que determinará quién puede acceder al atributo o método. Por ejemplo, una clase puede representar a un coche, sus atributos serán la cilindrada, la potencia y la velocidad, y tendrá dos métodos, uno para acelerar, que subirá la velocidad, y otro para frenar que la bajará.
* **Relaciones**: en el diagrama se representan relaciones reales entre los elementos del sistema a los que hacen referencia las clases. Pueden ser de asociación, agregación, composición y generalización. Por ejemplo, si tenemos las clases persona y coche, se puede establecer la relación conduce entre ambas. O una clase alumno puede tener una relación de generalización respecto a la clase persona.
* **Notas**: se representan como un cuadro donde podemos escribir comentarios que ayuden al entendimiento del diagrama.
* **Elementos de agrupación**: se utilizan cuando hay que modelar un sistema grande, entonces las clases y sus relaciones se agrupan en paquetes, que a su vez se relacionan entre sí.

# CREACIÓN DE CLASES

Una clase se representa en el diagrama como un rectángulo divido en tres partes:

* Arriba aparece el nombre de la clase.
* Medio los atributos.
* Abajo los métodos.

Interfaz de usuario gráfica, Texto, Chat o mensaje de texto

Descripción generada automáticamente

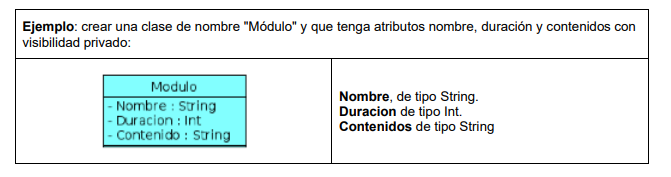
## ATRIBUTOS

Forman la parte estática de la clase. Son un conjunto de variables para las que es preciso definir:

* Su **nombre**.
* Su **tipo**, puede ser un tipo simple, que coincidirá con el tipo de dato que se seleccione en el lenguaje de programación final a usar, o compuesto, pudiendo incluir otra clase.

Además, se pueden indicar otros datos como un valor inicial o su visibilidad. La visibilidad de un atributo se puede definir como:

* **Público (+)**: se pueden acceder desde cualquier clase y cualquier parte del programa.
* **Privado (-)**: sólo se pueden acceder desde operaciones de la clase.
* **Protegido (#)**: sólo se pueden acceder desde operaciones de la clase o de clases derivadas en cualquier nivel.
* **Paquete (~)**: se puede acceder desde las operaciones de las clases que pertenecen al mismo paquete que la clase que estamos definiendo. Se usa cuando el lenguaje de implementación es Java.

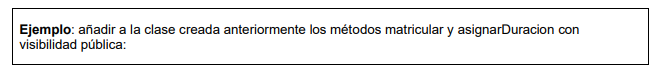


## MÉTODOS

Representan el comportamiento de una clase. Para definirlo hay que indicar la cabecera: nombre, parámetros, el return y su visibilidad.

Existen dos métodos particulares:

* El **constructor**. Tiene las siguientes características:
  + No devuelve ningún valor.
  + Tiene el mismo nombre de la clase.
  + Ejecutar las acciones necesarias cuando se instancia un nuevo objeto, es decir, reservar memoria e inicializar valores en caso de que se desee. (new).
  + Puede haber varios constructores según las necesidades. (Incluso puede existir uno vacío).
* El **destructor**. Cuando no se vaya a utilizar más el objeto, se podrá utilizar un método destructor que libere los recursos del sistema que tenía asignados.



Diagrama

Descripción generada automáticamente con confianza media

# RELACIONES ENTRE CLASES

Es una conexión entre dos clases.

Se representan con una línea continua y determinan que entre las clases se envían mensajes (métodos) normalmente en ambas direcciones, si sólo van en una dirección, se finaliza la línea con una punta de flecha.

Se debe representar una cardinalidad.

Diagrama

Descripción generada automáticamente

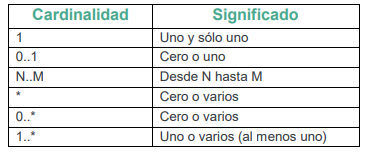
Es posible realizar relaciones reflexivas si la clase establece relaciones consigo misma:

Diagrama

Descripción generada automáticamente

## CARDINALIDAD DE UNA RELACIÓN

La cardinalidad de una relación, representa cuantos objetos de una clase se van a relacionar con objetos de otra clase. En una relación hay dos cardinalidades, una para cada extremo de la relación y pueden tener los siguientes valores:



## GENERALIZACIÓN (HERENCIA)

La generalización es una propiedad que permite a los objetos ser construidos a partir de otros objetos, es decir, la capacidad de un objeto para utilizar estructuras de datos y métodos presentes en sus antepasados. También recibe el nombre de herencia.

El objetivo principal de la generalización es la reutilización, poder utilizar código desarrollado con anterioridad. La herencia supone una clase base y una jerarquía de clases que contiene las clases derivadas. Las clases derivadas pueden heredar el código y los datos de su clase base, añadiendo su propio código especial y datos, incluso cambiar aquellos elementos de la clase base que necesitan ser diferentes.

Tipos:

* **Herencia simple**: una clase puede tener sólo un ascendente inmediato. Es decir, una subclase puede heredar datos y métodos de una única clase base.
* **Herencia múltiple**: una clase puede tener más de un ascendente inmediato, adquirir datos y métodos de más de una clase.

Diagrama

Descripción generada automáticamente

## AGREGACIÓN Y COMPOSICIÓN

* La agregación es una asociación binaria que representa una relación todo-parte (pertenece a, tiene un, es parte de).

Diagrama

Descripción generada automáticamente

* La composición es una agregación fuerte en la que una instancia ‘parte’ está relacionada, como máximo, con una instancia ‘todo’ en un momento dado, de forma que cuando un objeto ‘todo’ es eliminado, también son eliminados sus objetos ‘parte’.

Diagrama

Descripción generada automáticamente

Estas relaciones se representan con un rombo en el extremo de la entidad contenedora (todo):

* En el caso de la agregación es de color blanco.
* Para la composición negro.

Como en toda relación hay que indicar la cardinalidad.

## ATRIBUTOS DE ENLACE

Es posible que tengamos alguna relación en la que sea necesario añadir algún tipo de información que la complete de alguna manera. Cuando esto ocurre podemos añadir atributos a la relación.

Diagrama

Descripción generada automáticamente

## RESTRICCIONES

En ocasiones la relación entre dos clases está condicionada al cumplimiento de algún requisito o un parámetro de una clase que tiene un valor constante...

Cuando se precisa reflejar una condición que aparece en el enunciado y no disponemos de una notación particular para que quede reflejada en el diagrama de clases, es posible mostrarla mediante una restricción.

Las restricciones se incluyen mediante una descripción textual encerrada entre llaves.

# PAUTAS PARA CREAR DIAGRAMAS DE CLASES

A la hora de crear diagramas de clases, la clave está en hacer una buena elección de las clases que sugiere el problema. Para identificar las clases candidatas a formar parte del diagrama, es recomendable subrayar cada nombre o sintagma nominal que aparece en el enunciado. Cuando tengamos la lista completa habrá que estudiar cada clase potencial para ver si, finalmente, es incluida en el diagrama. Para ayudarnos a decidir, podemos utilizar los siguientes criterios:

* La información de la clase es necesaria para que el sistema funcione.
* La clase posee un **conjunto de atributos** que podemos encontrar en cualquier ocurrencia de sus objetos. Si sólo aparece un atributo normalmente se rechazará y será añadido como atributo de otra clase.
* La clase tiene un **conjunto de operaciones** identificables que pueden cambiar el valor de sus atributos y son comunes en cualquiera de sus objetos.
* Es una entidad externa que consume o produce información esencial para la producción de cualquier solución en el sistema.

La clase se considera si cumple todos (o casi todos) los criterios.

Se debe tener en cuenta que **la lista obtenida no incluye todo**, habrá que añadir clases adicionales para completar el modelo y también, que diferentes descripciones del problema pueden provocar la toma de diferentes decisiones al seleccionar las clases y sus atributos.