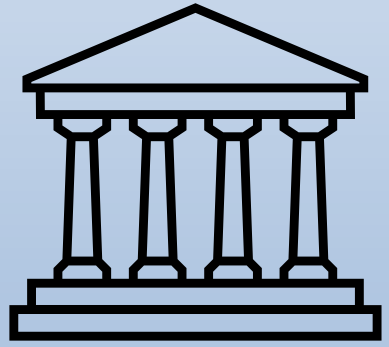
Programación Orientada a Objetos (POO)

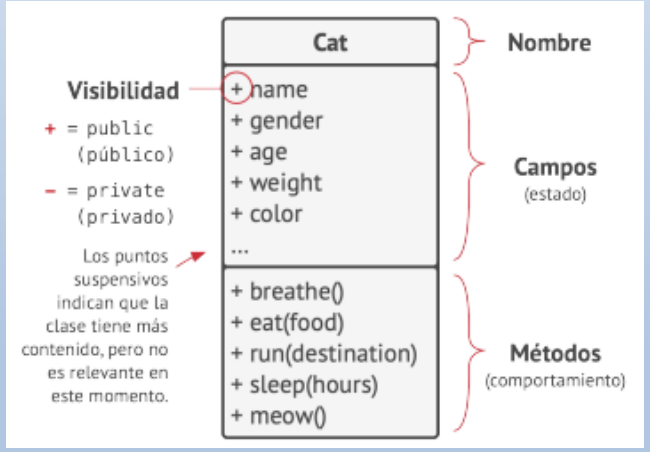


Bloque 1 – Concetos POO
2º DAW – Desarrollo Web en Entorno Servidor
Fernando Díaz-Alonso Dorado

Conceptos básicos POO

¿Qué es?

La Programación Orientada a Objetos **(POO)** es un paradigma basado en el concepto de envolver bloques de información y su comportamiento relacionado. Estos bloques los llamamos **objetos**, y se construyen a partir de un grupo de “recetas” que definimos y a las que denominamos **clases.**



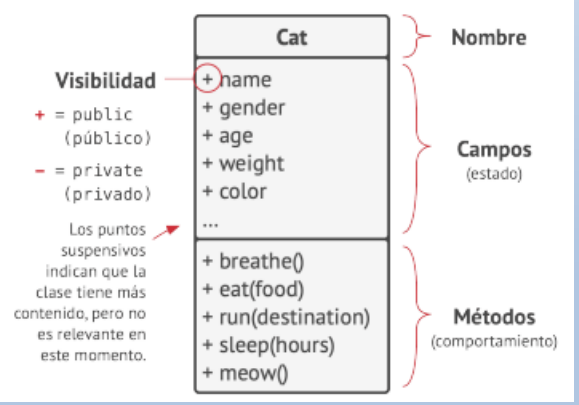
Un diagrama de clases UML

Conceptos básicos POO

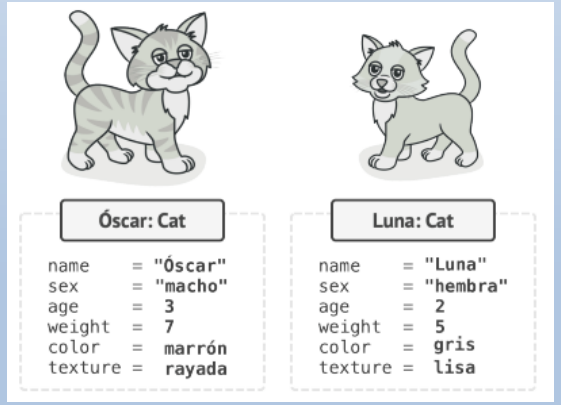
Objetos y clases

La Programación Orientada a Objetos **(POO)** es un paradigma basado en el concepto de envolver bloques de información y su comportamiento relacionado. Estos bloques los llamamos **objetos**, y se construyen a partir de un grupo de “recetas” que definimos y a las que denominamos **clases.**

Un diagrama de clases UML



2 objetos instancias de la clase: Cat

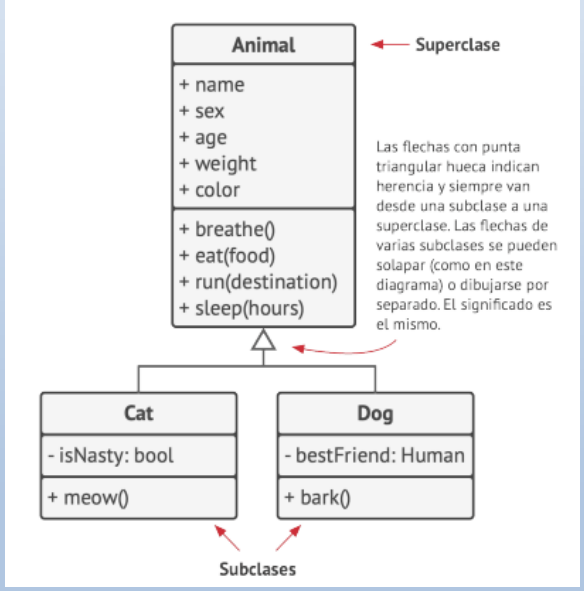


Florida Universitària

Conceptos básicos POO

Jerarquías

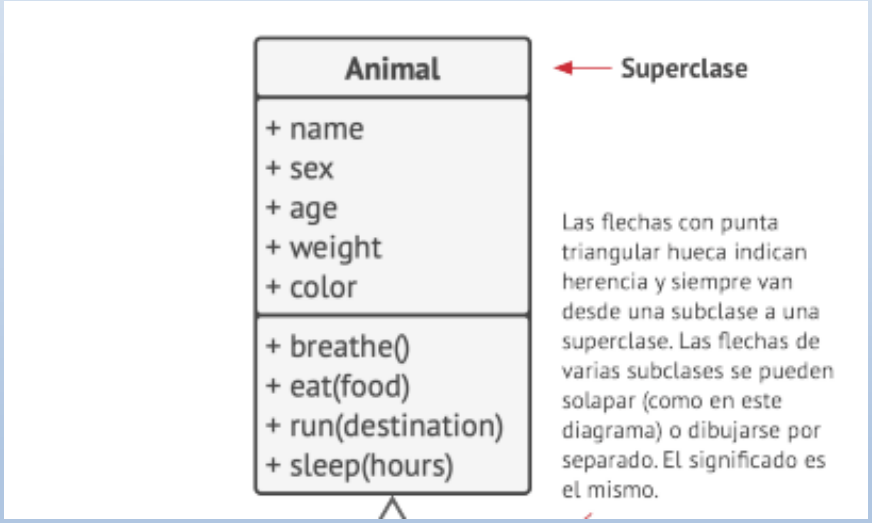
Cuando trabajas con varias clases, es normal que se organicen las clases mediante jerarquías. Lo que da lugar a **Superclases** y **subclases.**



Conceptos básicos POO

Superclases

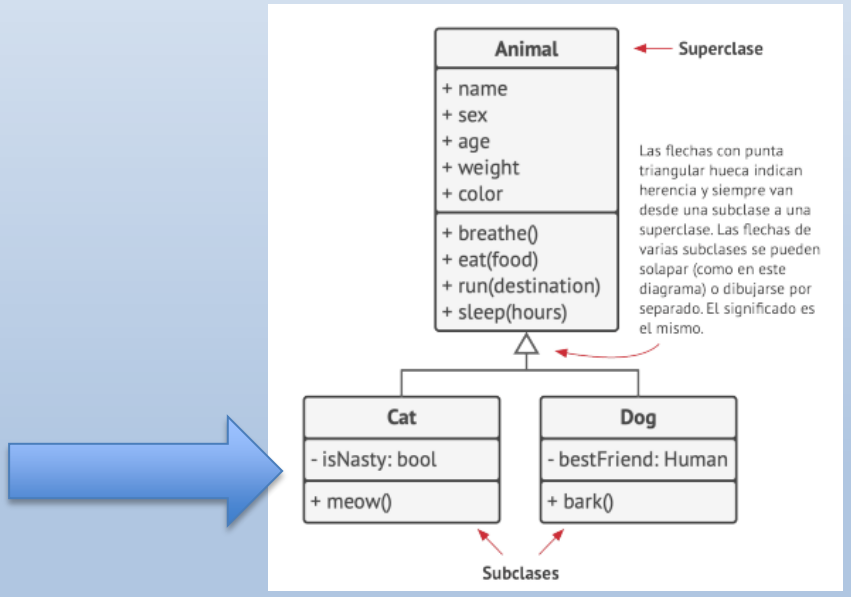
Las **superclases** o **clases padre** son aquella que engloba todos los atributos y comportamientos comunes.



Conceptos básicos POO

Subclases

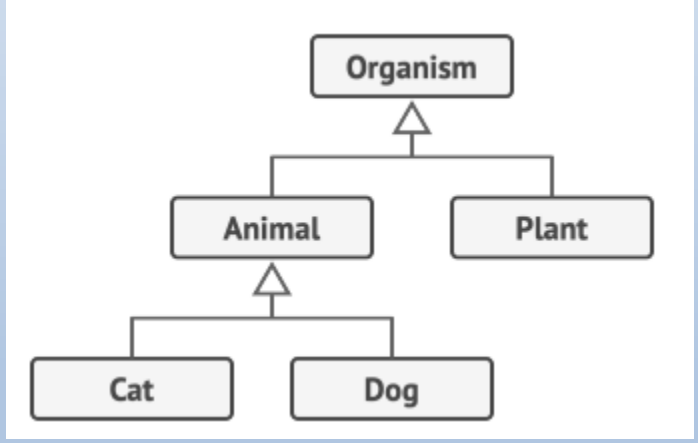
Las **subclases o hijas** son aquellas clases que heredan de una superclase, padre, el estado y el comportamiento y se encargan de definir los atributos o comportamientos diferentes.



Conceptos básicos POO

Jerarquía II

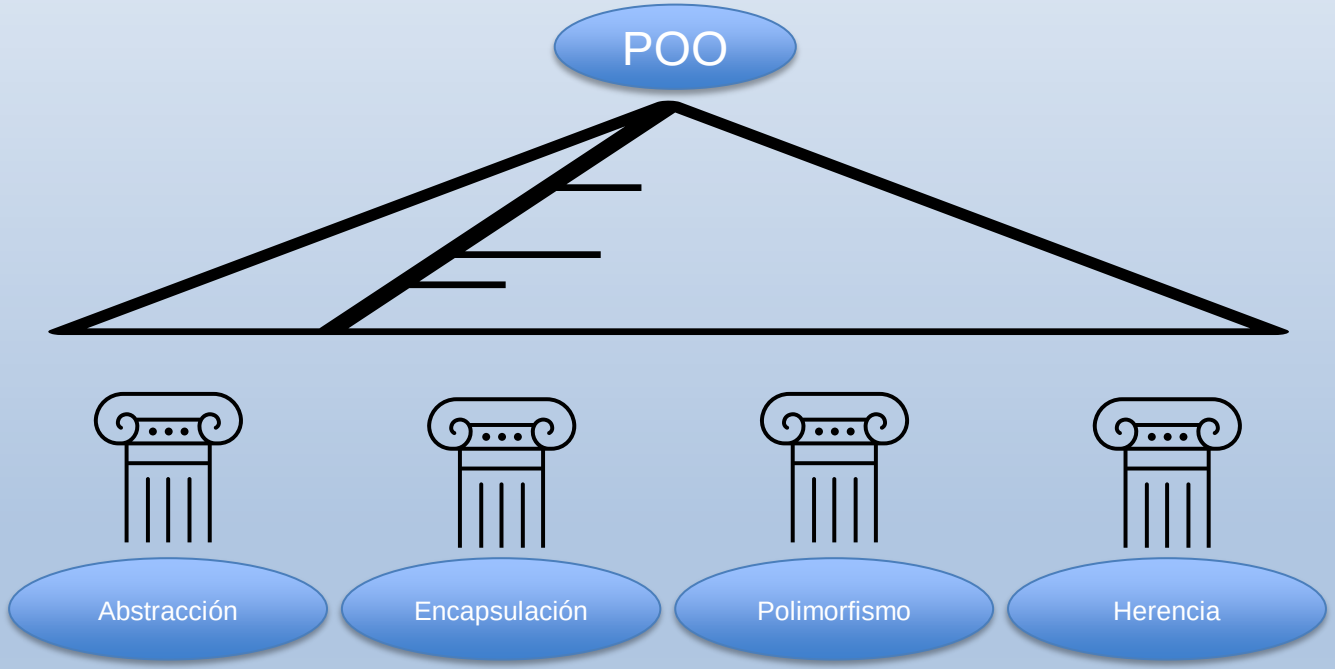
Puede darse el caso de que poseamos una superclase de la que heredemos toda la información de nuestras subclases y tengamos una jerarquía con varios niveles de jerarquía. Tipo abuelo, padre e hijos.



Pilares de la POO

Los pilares de la POO

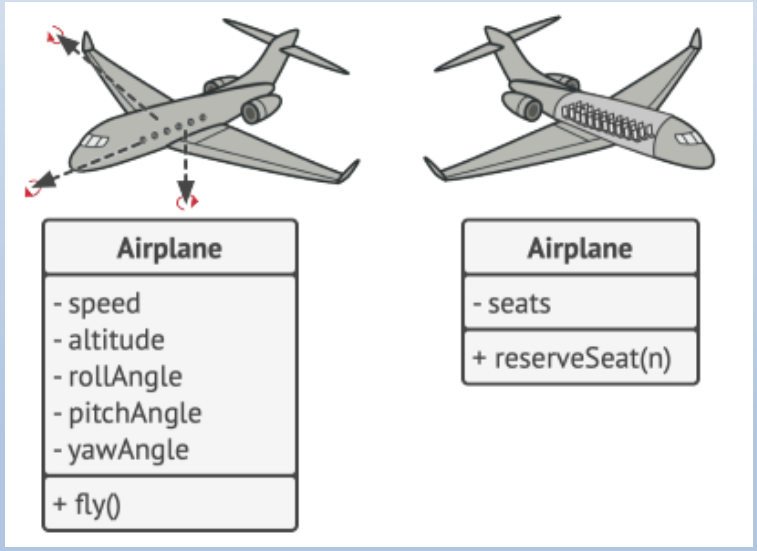
La POO se basa principalmente en cuatro pilares, que son los conceptos que la diferencian de otros paradigmas de programación.



Pilares de la POO

Abstracción

La **abstracción** de un objeto es el modelado de un objeto del mundo real según las necesidades y basadas en el contexto específico que es requerido, representando únicamente los datos que necesitamos y omitiendo el resto.



Pilares de la POO

Encapsulación

Mediante la **encapsulación** definimos la capacidad de un objeto de esconder estados y comportamientos a otros objetos, mostrando únicamente una interfaz limitada al resto del programa. Siendo la única forma de interactuar con el objeto.

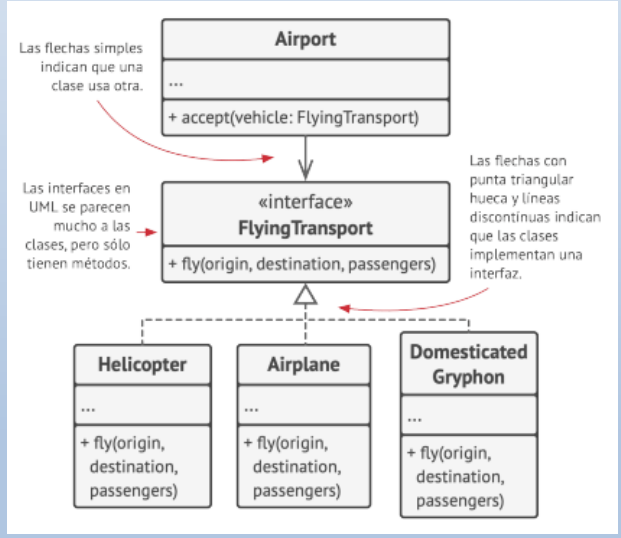
Para encapsular los métodos los hacemos privados o protegidos.

En la vida real sería el ejemplo del volante de un coche que sería la interfaz de coche para las acciones de giro de cada una de las ruedas, transmisiones, etc…

Pilares de la POO

Interfaces, clases y métodos abstractos

Las **interfaces,** las **clases y métodos abstractos** se definen en los lenguajes de programación mediante los conceptos anteriores de **Abstracción** y de **Encapsulación**.



Pilares de la POO

Herencia

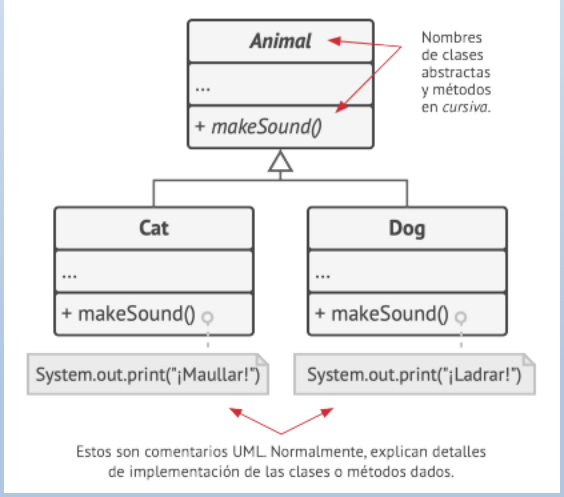
Es la capacidad de crear nuevas clases a partir de otras existentes. Su principal ventaja es la reutilización del código.

Mediante la herencia creamos clases hijas que usan la interfaz de la clase padre. Por tanto, todas las hijas están obligadas a implementar las interfaces del padre las usen o no.

Pilares de la POO

Polimorfismo

Es la capacidad que tiene un programa de detectar la verdadera clase de un objeto e invocar su implementación. Podríamos decir que es la capacidad de una clase de “fingir” que son otra clase, al extender o implementando una interfaz de una clase padre.



Relaciones entre Objetos

Dependencia

Es la relación básica y más débil que existe entre clases.

Las modificaciones en una de las clases pueden provocar cambios en la otra clase.

Esto es cuando utilizas nombres de clases concretas, al instanciar un objeto mediante el constructor, etc…

- **La clase A puede verse afectada por cambios en la clase B.**

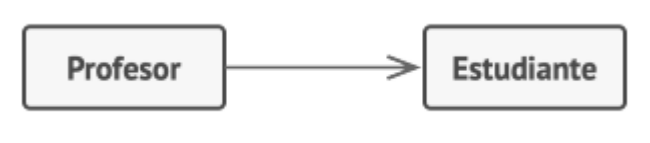
Relaciones entre Objetos

Asociación

Es una relación en la que un objeto utiliza o interactúa con otro. Pueden ser bidireccionales.

La asociación es una dependencia especializada donde un objeto tiene acceso a los objetos con los que interactúa, en una dependencia simple no se establece un vínculo permanente.

- **El objeto A conoce el objeto B. La clase A depende de la B.**



Relaciones entre Objetos

Agregación

Es un tipo especializado de asociación que representa relaciones “uno a muchos”, “muchos a muchos”.

Generalmente, con la agregación, un objeto “tiene” un grupo de otros objetos y sirve como contenedor o colección. El componente puede existir sin el contenedor y puede vincularse a varios contenedores a la vez.

- **El objeto A conoce el objeto B y consiste en B. La clase A depende de B.**

Relaciones entre Objetos

Composición

Es un tipo especializado de agregación en la que un objeto se compone de una o más instancias del otro. El componente solamente puede existir como parte del contenedor.

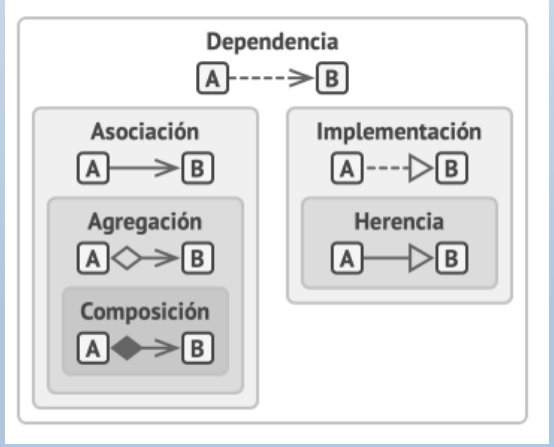
- **El objeto A conoce el objeto B, consiste en B y gestiona el ciclo vital de B. La clase A depende de B.**

Relaciones entre Objetos

Resumen de las relaciones

Las relaciones se conectan entre sí como se muestra en el esquema.

- **Implementación** → La clase A define métodos declarados en la interfaz B. Los objetos A pueden tratarse como B. La clase A depende de B.
- **Herencia** → La clase A hereda la interfaz y la implementación de la clase B, pero puede extenderla. El objeto A puede tratarse como B. La clase A depende de B.



Programación Orientada a Objetos

Bibliografía

- Patrones de Diseño – Alexander Shvets