

OPTATIVA 2: PROGRAMACIÓN. 2º SMR

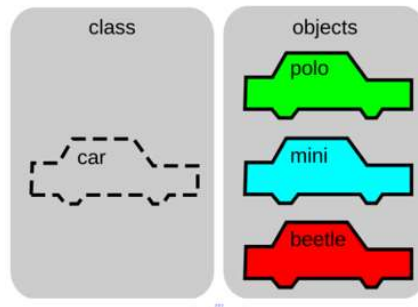
Programación Orientada a Objetos

La POO es un paradigma de programación que se basa en el uso de clases y objetos.

Una clase es una estructura que define las propiedades (atributos) y el comportamiento (métodos) de un objeto.

Un objeto es una instancia (o un caso concreto) de una clase.

Idea: Las clases se pueden considerar los moldes, plantillas o el diseño para crear objetos. Los objetos se crearían basándose en esos *moldes* o *diseños*.



En general, en las clases se definen las propiedades o atributos y los métodos que tendrán los objetos concretos que se creen basándose en una determinada clase.

El estado de un objeto dependerá de sus valores o atributos; el comportamiento de los objetos depende de los métodos definidos:

- ⦿ Métodos -> funciones.
- ⦿ Atributos -> variables.

Ejemplos (en pseudocódigo):

<pre>Clase perro{ Nombre: tipo cadena de caracteres Raza: tipo cadena de caracteres Edad: tipo número Color: tipo cadena de caracteres función ladrar(){ imprimir "Guau" } función gruñir(){ imprimir "Grrrr!!!" } }</pre>	<pre>Clase coche{ Marca: tipo cadena de caracteres Modelo: tipo cadena de caracteres Potencia: tipo número Color: tipo cadena de caracteres función arrancar(){ imprimir "Consumiendo combustible" } función pedir_mantenimiento(){ imprimir "Llévame al taller" } }</pre>
---	---

OPTATIVA 2: PROGRAMACIÓN. 2º SMR

#Ejemplo de POO. Se define la clase perro y se crea un objeto de esta clase.

```
class perro:
    def __init__(self, nombre, raza, edad, color):
        print("Creado!")
        self.nombre = nombre
        self.raza = raza
        self.edad = edad
        self.color = color

    def ladrar(self):
        print("Guau!")

    def grunir(self):
        print("GRRRRRR!")

Rocky = perro("Rocky", "Bulldog Francés", 11, "Negro")

print(Rocky.nombre)
print(Rocky.raza)
print(Rocky.edad)
print(Rocky.color)

Rocky.ladrar()
Rocky.grunir()
```

Creado!
Rocky
Bulldog Francés
11
Negro
Guau!
GRRRRRR!

Los cuatro principios de la POO:

La POO se basa en cuatro principios fundamentales:

Encapsulamiento: las variables y los métodos se definen juntos y se controla el acceso a los datos (variables) permitiendo la interacción con las variables solo con los métodos definidos en la clase.

Abstracción: consiste en ocultar los detalles que no tienen que conocerse. Se reduce la complejidad del uso porque el usuario no tiene que conocer los detalles de cómo se hacen las cosas.

Polimorfismo: capacidad de usar un mismo nombre de método para comportamientos diferentes, según el objeto que lo ejecute o el número de parámetros que se usan al llamar al método.

Herencia: mecanismo para que una clase que es hija de otra herede sus métodos y propiedades.