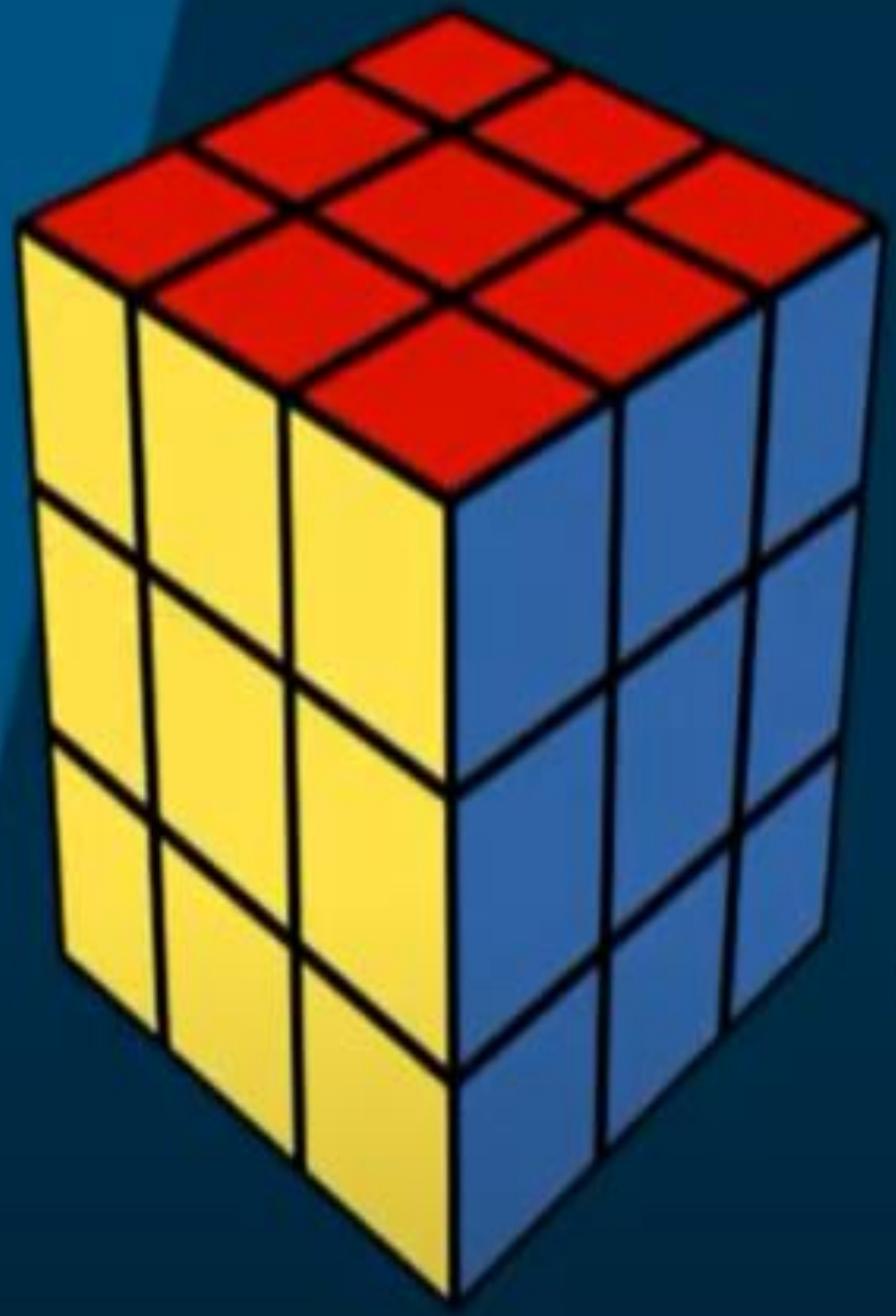
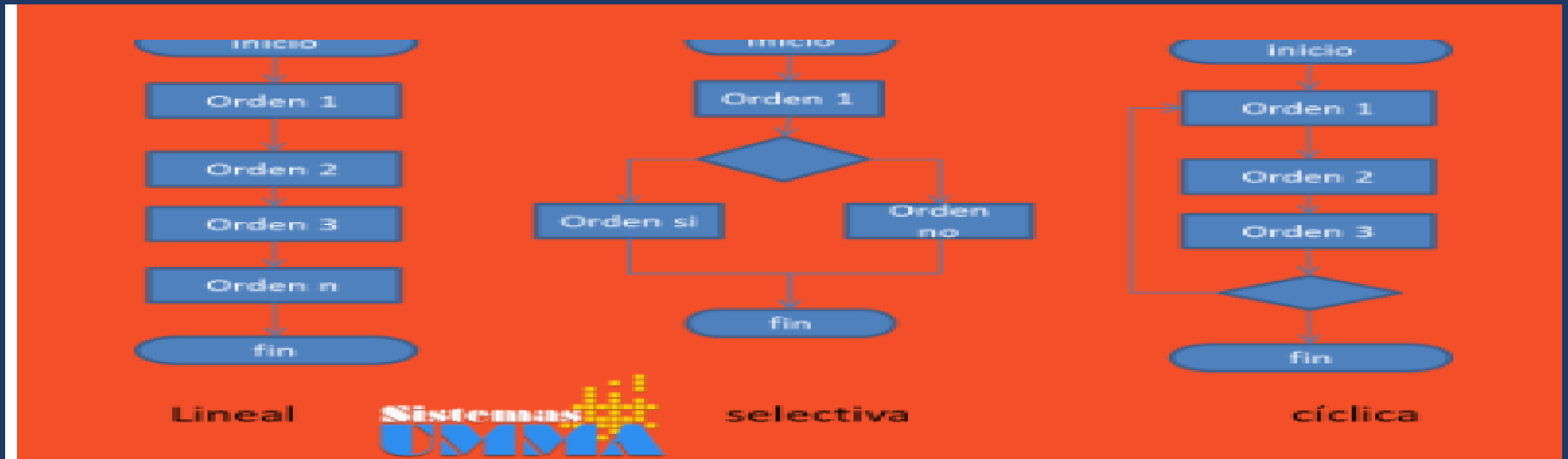
Programación Orientada a Objetos



Programación Orientada a Objetos vs Programación Estructurada

¿Qué es la Programación Estructurada?

La programación estructurada es un paradigma de programación que se basa en la ejecución secuencial de instrucciones. En este enfoque, el código se divide en bloques lógicos y se organiza en funciones y procedimientos y se utiliza un conjunto limitado de estructuras de control, como secuencias, selecciones y bucles. La programación estructurada es conocida por su simplicidad y claridad, lo que facilita la comprensión y el mantenimiento del código.

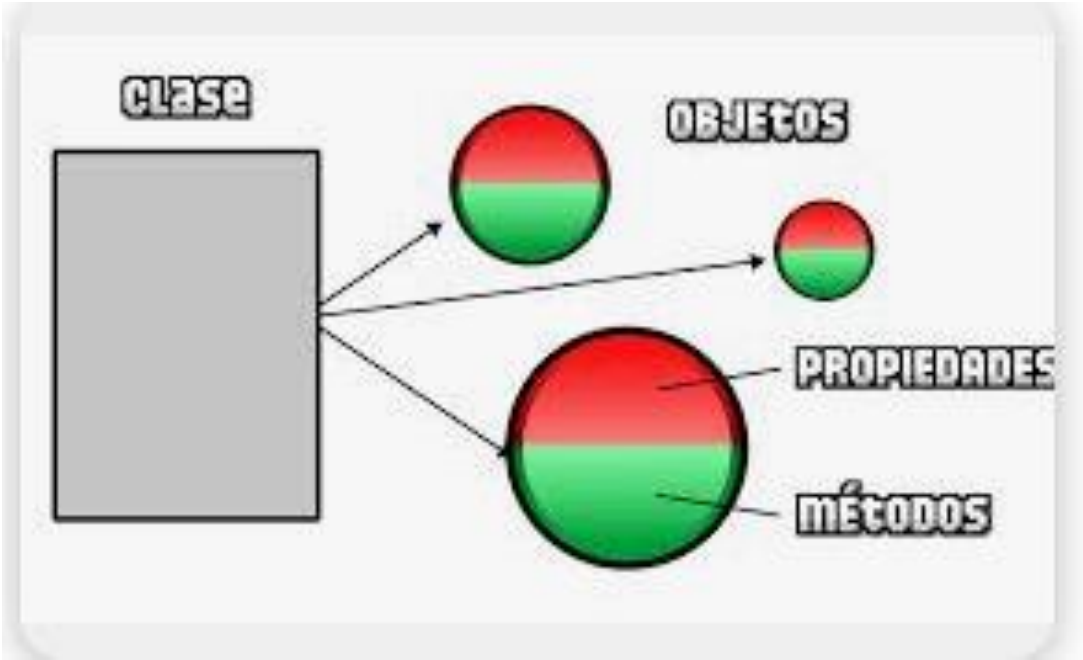


Desventajas de la Programación Estructurada

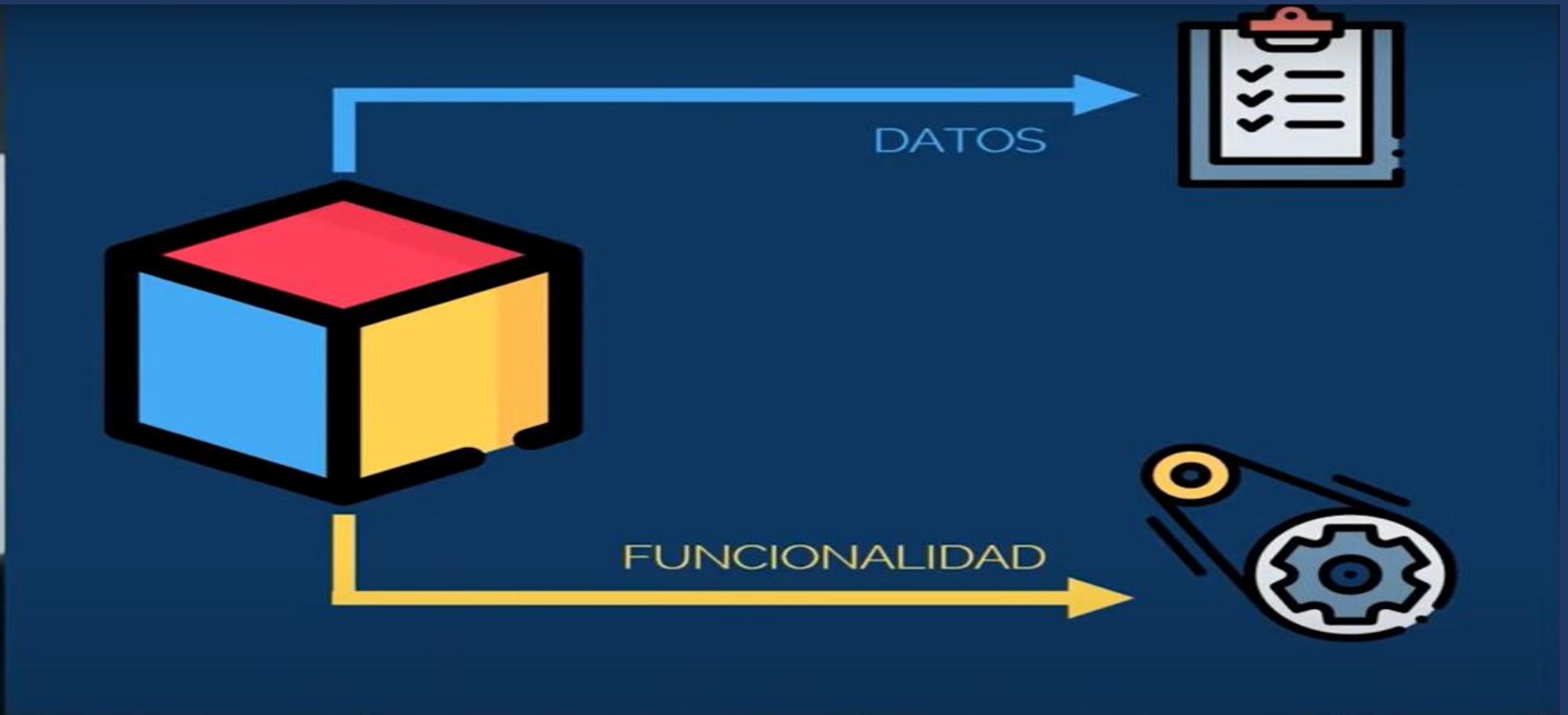
- Aunque la programación estructurada ofrece muchas ventajas, como la mejora en la legibilidad y el mantenimiento del código, también tiene algunas desventajas, especialmente cuando se compara con otros paradigmas de programación más modernos. Aquí te detallo algunas de las desventajas:
- **Dificultad para Modelar Sistemas Complejos**: La programación estructurada puede volverse complicada y difícil de manejar cuando se trata de sistemas muy grandes y complejos. El uso excesivo de funciones y procedimientos puede llevar a un código fragmentado y difícil de seguir.
- **Poca Reutilización de Código**: En comparación con la programación orientada a objetos, la programación estructurada no promueve de manera natural la reutilización del código. Los programas estructurados tienden a repetir código similar en diferentes partes del programa, lo que puede llevar a inconsistencias y errores.
- **Escalabilidad Limitada**: La falta de modularidad y encapsulamiento en la programación estructurada puede dificultar la escalabilidad del software. A medida que el programa crece, mantener la estructura y evitar la duplicación de código se vuelve cada vez más difícil.
- **Gestión de Datos Compleja**: Manejar estructuras de datos complejas, como objetos o relaciones entre datos, puede ser más engorroso en la programación estructurada. La ausencia de mecanismos como clases y objetos hace que la manipulación de datos complejos sea menos intuitiva y más propensa a errores.
- **Menor Flexibilidad en la Resolución de Problemas**: Algunos problemas pueden ser más fáciles de resolver utilizando paradigmas alternativos, como la programación orientada a objetos o la programación funcional. La programación estructurada puede no ser la mejor opción para todos los tipos de problemas, especialmente aquellos que se benefician de la abstracción y la reutilización de componentes.
- **Mantenimiento Complicado en Proyectos Grandes**: En proyectos de gran escala, mantener la coherencia y la claridad del código puede ser más difícil con la programación estructurada. A medida que crece el número de funciones y procedimientos, el código puede volverse menos manejable, lo que complica las tareas de mantenimiento y actualización.

¿Qué es la Programación Orientada a Objetos?

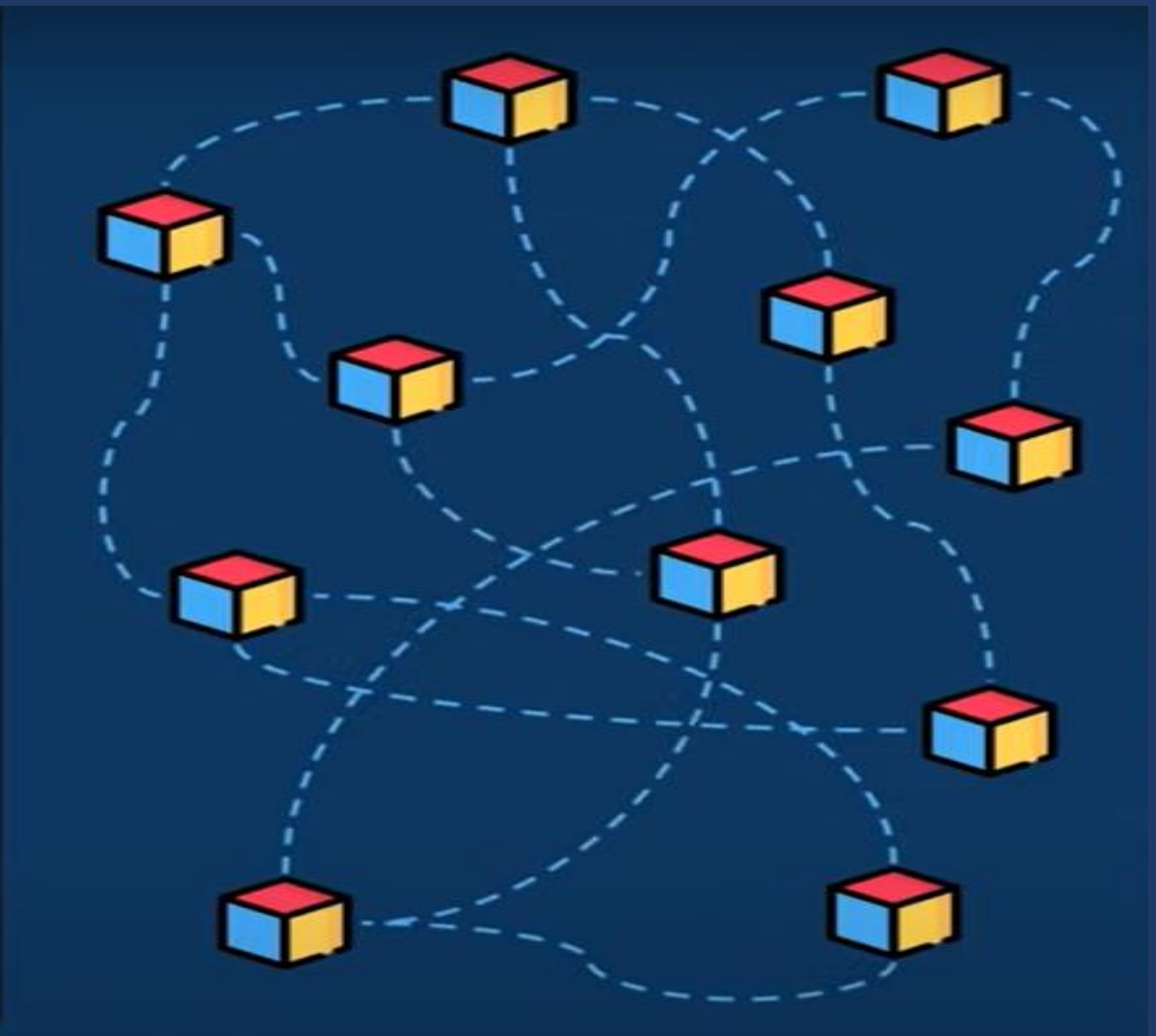
- **¿Qué es la Programación Orientada a Objetos?**
- Por otro lado, la programación orientada a objetos es un enfoque de programación que se basa en la creación de objetos que interactúan entre sí para realizar tareas. En la programación orientada a objetos, los objetos son entidades que tienen atributos y métodos, lo que permite encapsular la lógica y los datos de manera más eficiente. Este enfoque fomenta la reutilización del código y la modularidad, lo que facilita el desarrollo de aplicaciones complejas.



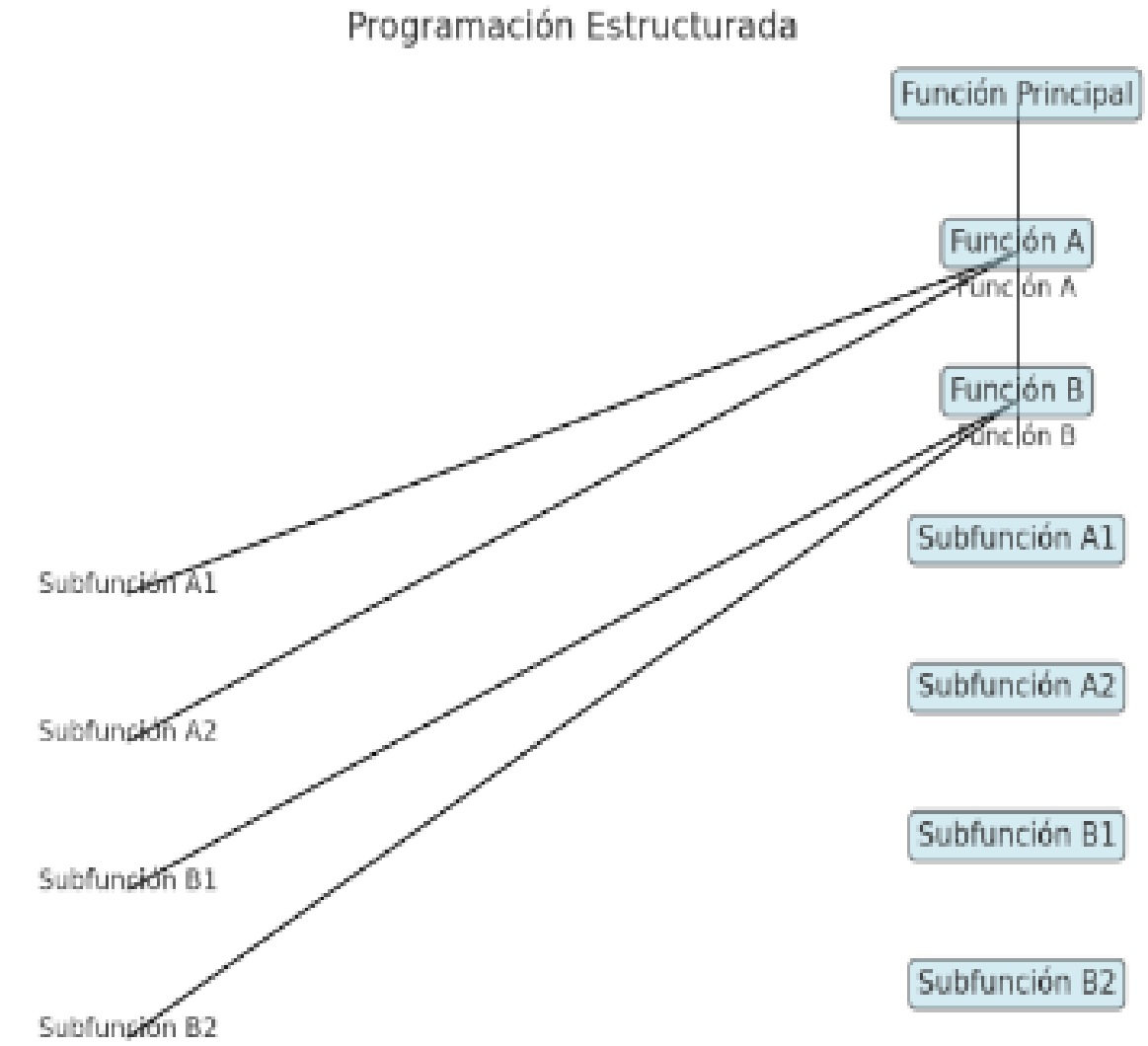
Qué es la Programación Orientada a Objetos

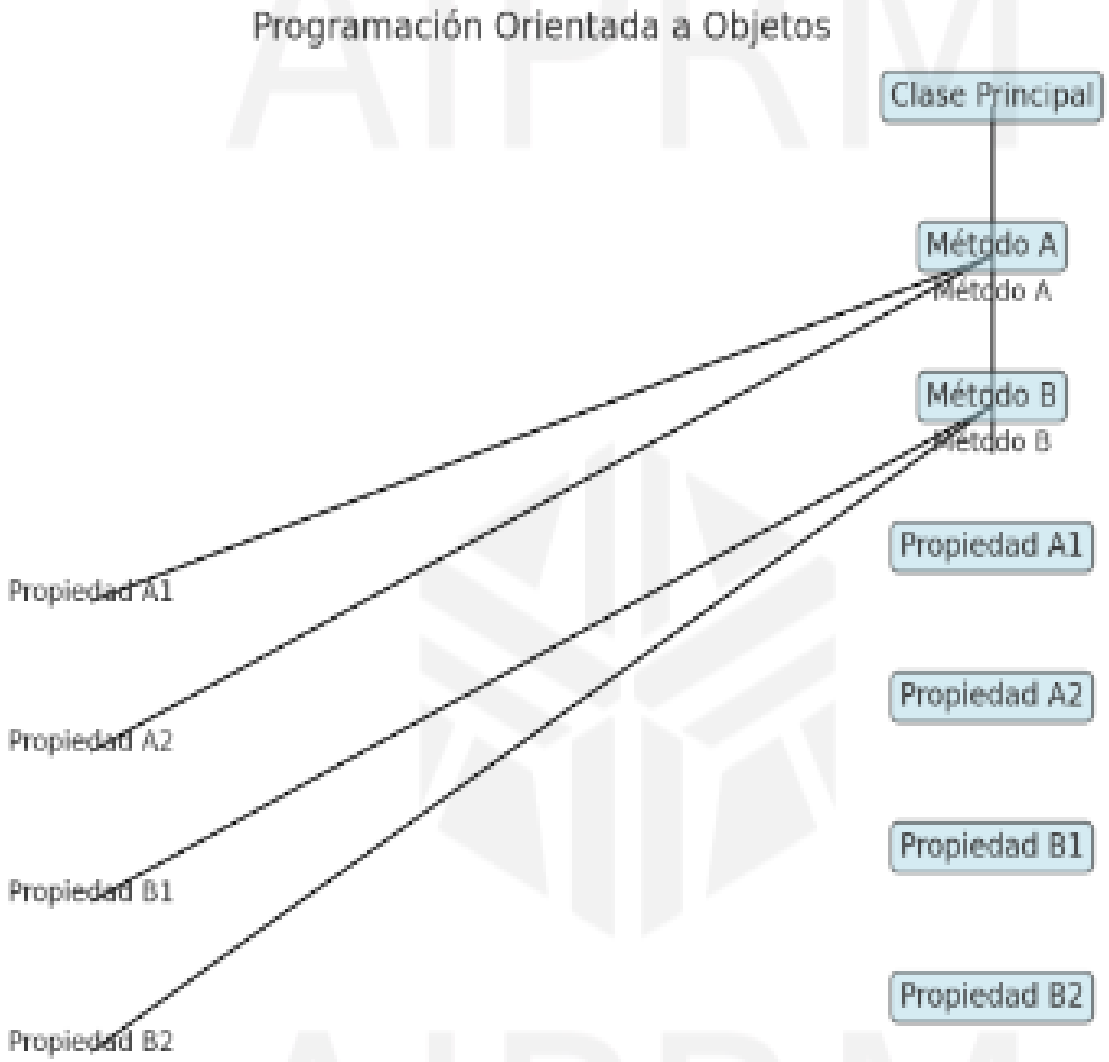


Qué es la Programación Orientada a Objetos



Diferencia entre Programación Estructurada y POO

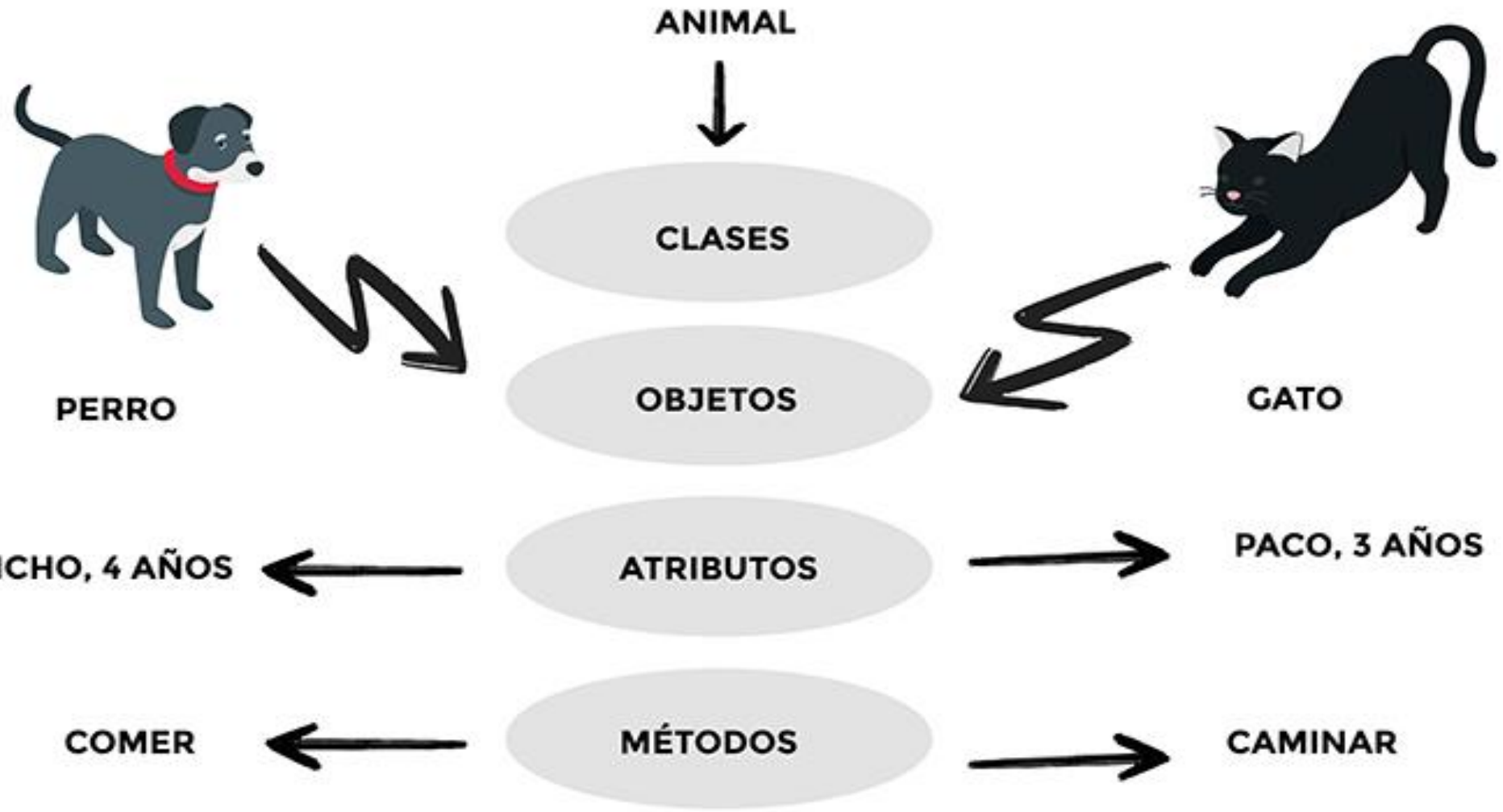




Claves de la Programación Orientada a Objetos

La POO se inspira en la forma en que percibimos y entendemos el mundo que nos rodea. Imagina que estás construyendo un sistema de gestión de una biblioteca. En lugar de pensar en términos de algoritmos y estructuras de datos, la POO te invita a considerar las entidades que existen en el contexto de la biblioteca, como libros, bibliotecarios y usuarios.

En este enfoque, cada una de estas entidades se convierte en un objeto, con propiedades (datos) y comportamientos (funcionalidades). Por ejemplo, un objeto «Libro» puede tener atributos como el título, el autor y el año de publicación, así como métodos para obtener información sobre el libro, prestarlo o devolverlo a la biblioteca.


ANIMAL
CLASES
OBJETOS
ATRIBUTOS
MÉTODOS
PERRO
GATO
PANCHO, 4 AÑOS
PACO, 3 AÑOS
COMER
CAMINAR

Beneficios de Programación Orientada a Objetos

- Reutilización del código.
- Convierte cosas complejas en estructuras simples reproducibles.
- Evita la duplicación de código.
- Al estar la clase bien estructurada permite la corrección de errores en diversos lugares del código.
- Protege la información mediante la encapsulación, pues solo se puede acceder a los datos del objeto mediante propiedades y métodos privados.
- La abstracción nos permite construir sistemas más complejos y de un modo más sencillo y organizado.

¿Que es un objeto en POO?

Un objeto es un ente que consta de un estado y de un comportamiento, que a su vez constan respectivamente de datos almacenados y de tareas realizables durante el tiempo de ejecución.
Ejemplo : Un Automovil

Propiedades :
Color
Precio
Marca
Velocidad

Funciones o Métodos:
Encender
Apagarse
Acelerar
Frenar

Qué es un Objeto?

Propiedades o Atributos:

- Tamaño
- Peso
- Color
- Numero de Dientes

Funciones o metodos:

- Oler
- Ladrar
- Caminar
- Comer

ATRIBUTOS

Nombres
Apellidos
Correo
Contraseña
Premium

MÉTODOS

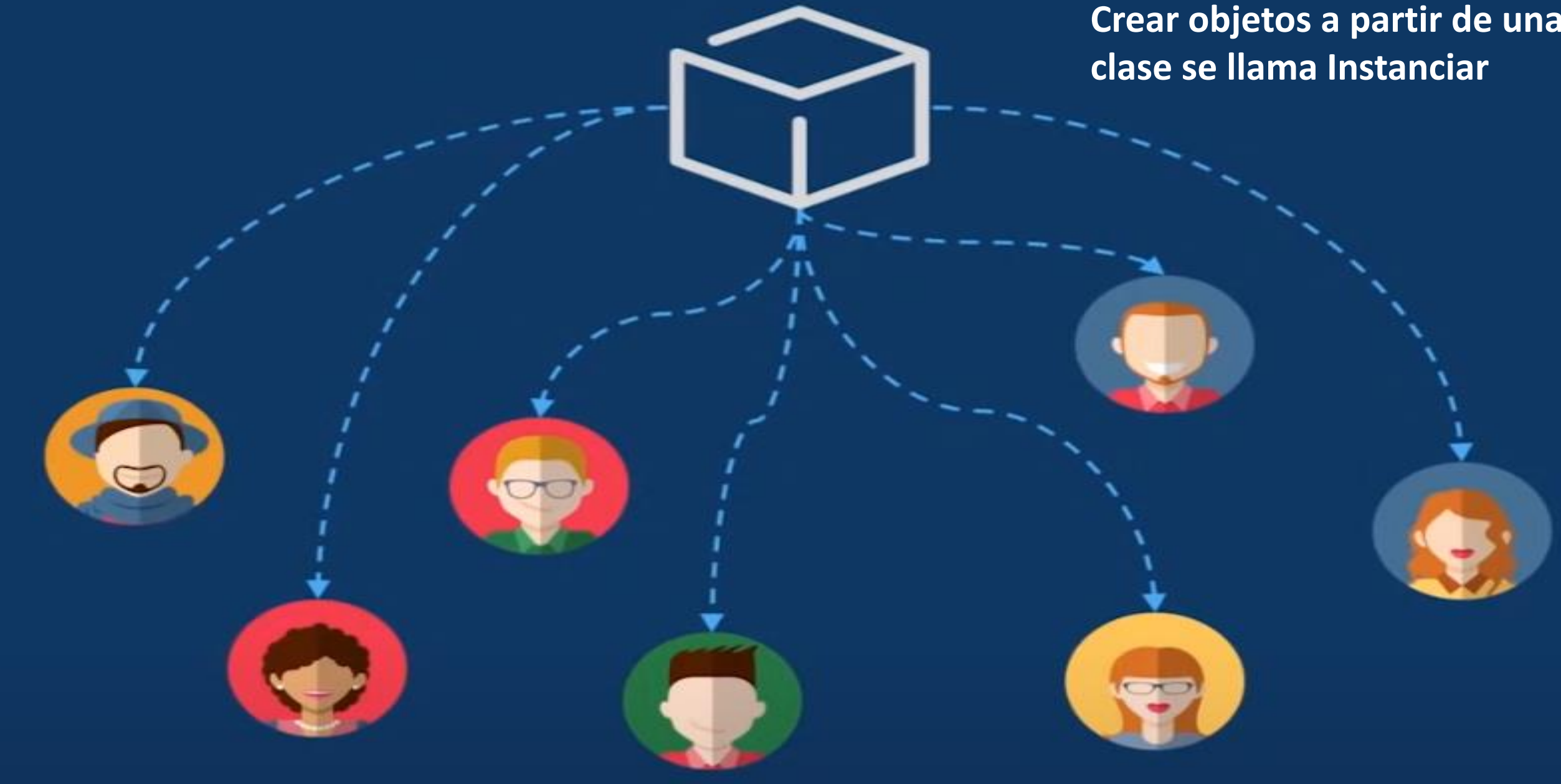
Iniciar sesion
Cerrar sesion
Editar perfil
Cambiar contraseña
Pasar a premium
Publicar en comunidad

Qué es una Clase ?

Qué es una Clase ?

'Plantilla' que tiene propiedades y funciones y sirve para definir objetos.

class Usuario



Crear objetos a partir de una clase se llama Instanciar

Qué es una Clase ?

CLASE AUTOMOVIL:
Atributos: Color, Precio, Marca
Funciones: Encender, Acelerar, Frenar

Atributos:
- Color
- Precio
- Marca

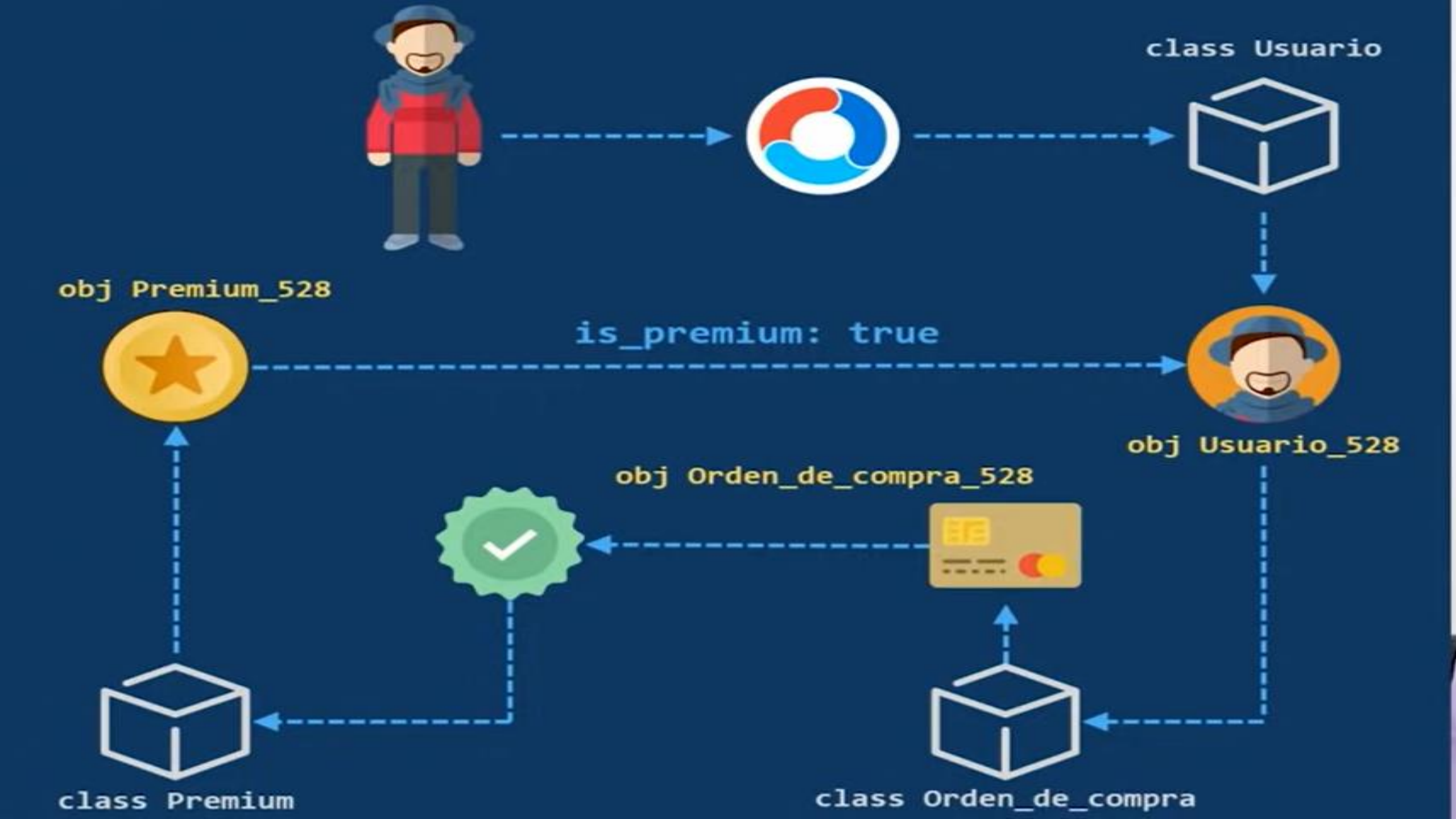
Funciones:
- Encender
- Acelerar
- Frenar

Atributos:
- Color
- Precio
- Marca

Funciones:
- Encender
- Acelerar
- Frenar

Atributos:
- Color
- Precio
- Marca

Funciones:
- Encender
- Acelerar
- Frenar


class Usuario
obj Premium_528
is_premium: true
obj Usuario_528
obj Orden_de_compra_528
class Premium
class Orden_de_compra

Clase

Instanciar

Objeto

Instanciar

Clase

Objeto

4 Principios de la Programación Orientada a Objetos

ABSTRACCION

La abstracción la podemos definir como el proceso de identificar aquellas características (atributos) y acciones o comportamientos (métodos) propios de un elemento que deseemos representar.

ENCAPSULAMIENTO

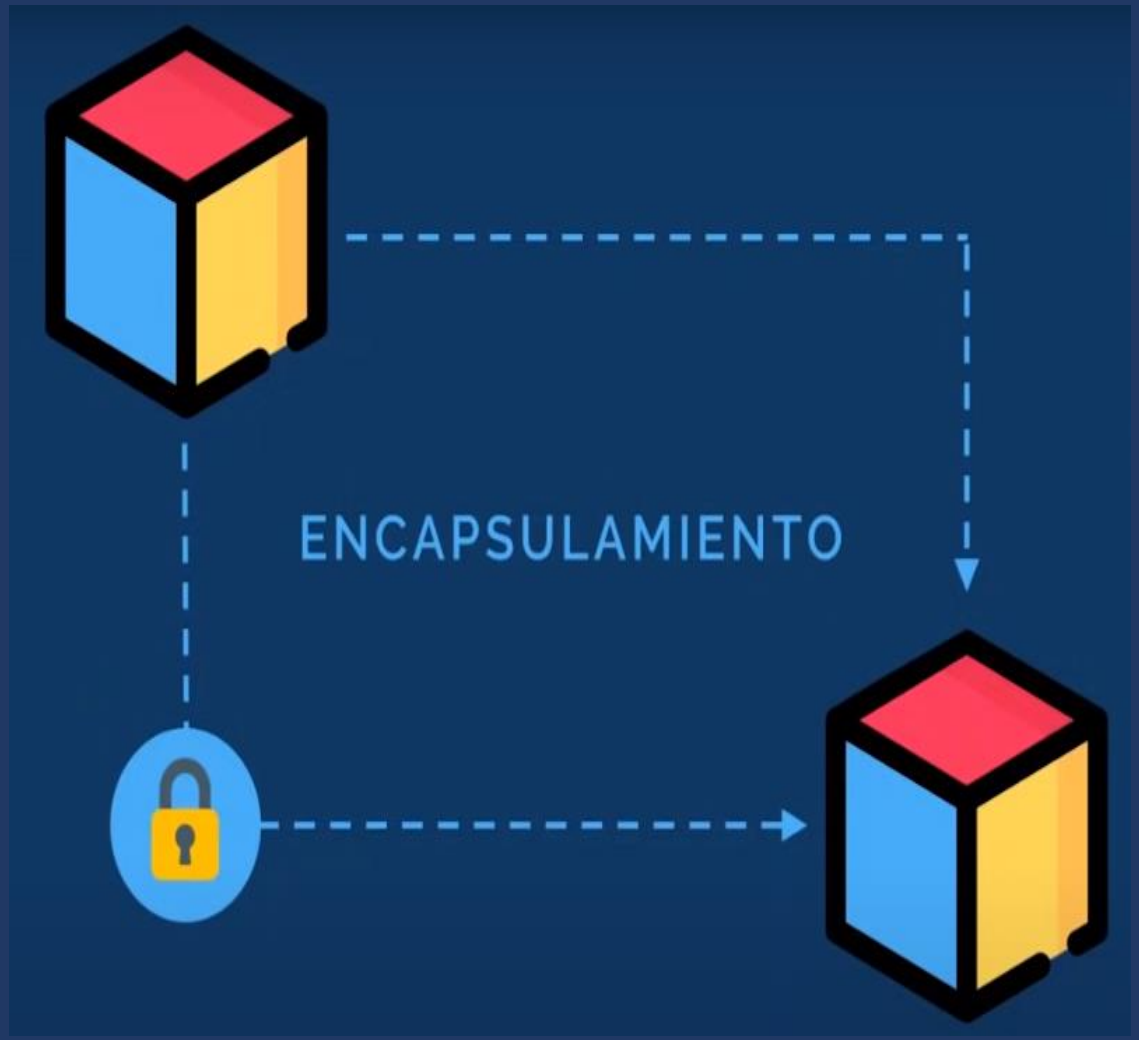
La encapsulación es un mecanismo de protección de atributos y métodos, es decir, protege a los datos asociados de un objeto contra su modificación por los objetos que no tenga derecho a acceder a ellos.

Niveles de encapsulamiento

Nivel Privado: los atributos y métodos del objeto sólo es accesible desde la misma clase.

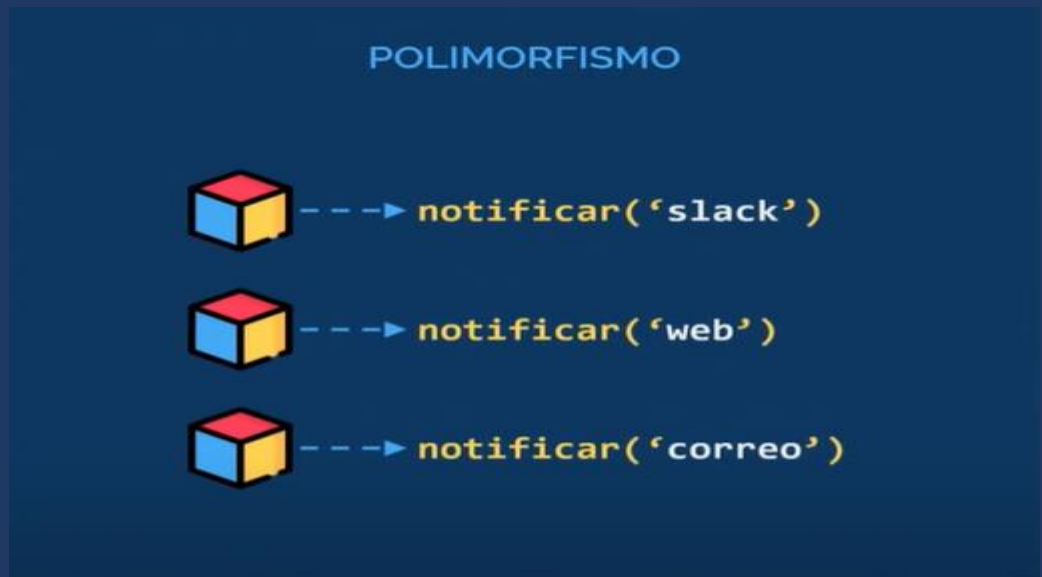
Nivel protegido: los atributos y métodos del objeto sólo es accesible desde la clase y las clases que heredan

Nivel abierto (publico): los atributos y métodos del objeto puede ser accedido desde cualquier clase.



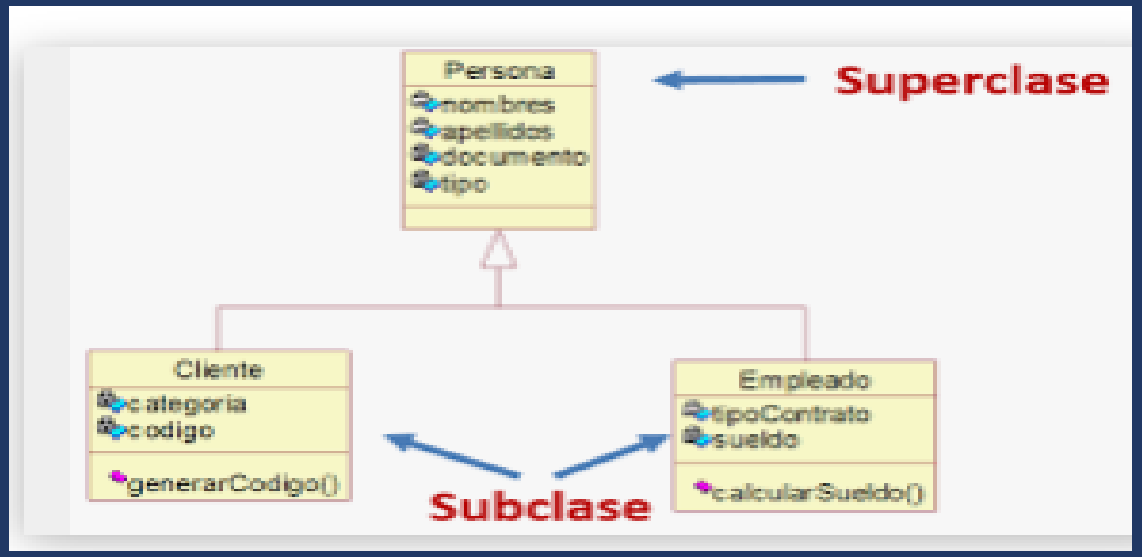
POLIMORFISMO

El polimorfismo se trata simplemente de que un único objeto puede tener múltiples estados y comportamientos, básicamente es la capacidad de los objetos de una clase, en responder de diferentes maneras a un solo mensaje.



HERENCIA

Es el mecanismo por el cual una clase permite heredar las características (atributos y métodos) de otra clase. La **herencia** permite que se puedan definir nuevas clases basadas de unas ya existentes a fin de reutilizar el código, generando así una jerarquía de clases dentro de una aplicación



Ejemplos básicos de código Python

```
edad=int(input('Introduzca su edad:\n'))
respuesta=None

if edad>=18:
    print('Es mayor de edad, puede comprar alcohol.
¿Cual desea? Introduzca un numero de opcion.')
    respuesta=input('1-ron.\n2-whisky\n3-ginebra\n')

    if respuesta=='1':
        print('Ha elegido comprar ron')
    elif respuesta=='2':
        print('Ha elegido comprar whisky' )
    elif respuesta=='3':
         print('Ha elegido comprar ginebra')
    else:
        print('Opcion no valida')
else:
       print('Es menor de edad, vuelva de aqui un tiempo o no empiece
       con el alcohol')
```

Ejemplos básicos de código Python

```
print('---------Calculadora Mejorada-----------')
def suma(a,b):
    return a+b

def Resta(a,b):
    return a-b

def Multiplicacion(a,b):
    return a*b

def Division(a,b):
    return a/b

def Modulo(a,b):
    return a%b

def Exponente(a,b):
    return a**b
```

```
while True:
    n1=float(input('Ingrese primer valor: '))
    n2=float(input('Ingrese segundo valor: '))
    print('\n')
    print('Hola, elija una opcion')
    print('1. Suma')
    print('2.Resta')
    print('3.Multiplicacion')
    print('4.Division')
    print('5.Modulo')
    print('6.Exponente')
    opcion=int(input('Teclee un numero y pulse Enter: '))

    match opcion:
        case 1:
            resultado=suma(n1,n2)
        case 2:
            resultado=Resta(n1,n2)
        case 3:
            resultado=Multiplicacion(n1,n2)
        case 4:
            resultado=Division(n1,n2)
        case 5:
                resultado=Modulo(n1,n2)
        case 6:
            resultado=Exponente(n1,n2)
        case _:
            print('Debe Ingresar una opcion Valida de 1 al 6')
```

Ejemplos básicos de código Python

```
print(f'El resultado de la operacion es : {resultado}\n')

    rpta=input('Desea continuar (s/n):  ').lower()
    print('\n')
    if rpta=='n':
        break
```

Funciones Lambda

```
m=lambda a,b:a*b
r=m(5,2)
print(r)


(lambda n1,n2:print(n1*n2))(7,5)
```