

# Paradigmas de programación

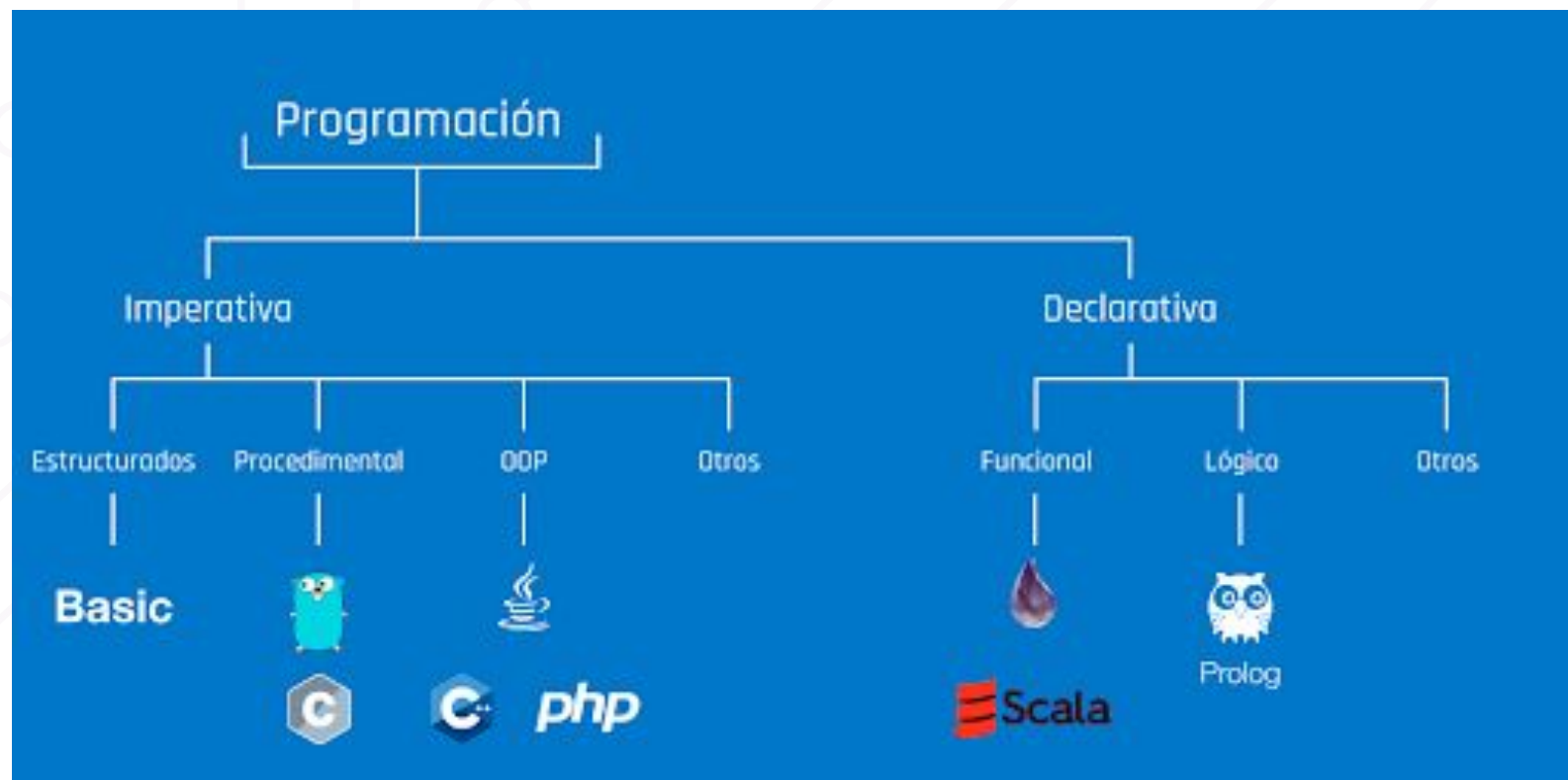
**DEV.F**  
DESARROLLAMOS(PERSONAS);

dev

The logo consists of the text 'DEV.F.' in a bold, white, sans-serif font. The 'F' is stylized with a grid-like pattern of small squares. The logo is centered within a dark blue diamond shape.

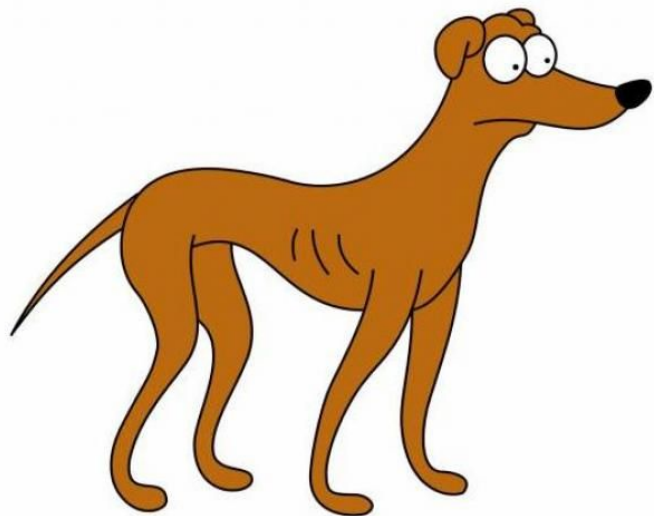
**DEV.F.**

*Un paradigma de programación  
es un estilo de desarrollo de  
programas. Es decir, un modelo  
para resolver problemas  
computacionales*



# Objetos

**DEV.F**  
DESARROLLAMOS(PERSONAS);



Metodos.

Atributos.

La estructura de un objeto literal está limitada por llaves, donde se encapsula cada identificador asignándole un valor literal, en un formato clave : valor. Si vamos a hacer uso del objeto más adelante, podemos almacenarlo en una variable asignándoselo con normalidad.

```
var nombreObjeto = {  
    identificador1: valor1,  
    identificador2: valor2,  
    identificador_n: valor_n  
}
```

# Todo como un objeto

En JavaScript, casi todo es un objeto. Todos los tipos primitivos excepto null y undefined se tratan como objetos. Pueden asignar propiedades, y tienen todas las características de los objetos.

The logo consists of the text 'DEV.FX' in a bold, white, sans-serif font. The 'X' is stylized with a grid of small squares. The logo is centered within a dark blue diamond shape.

**DEV.FX**

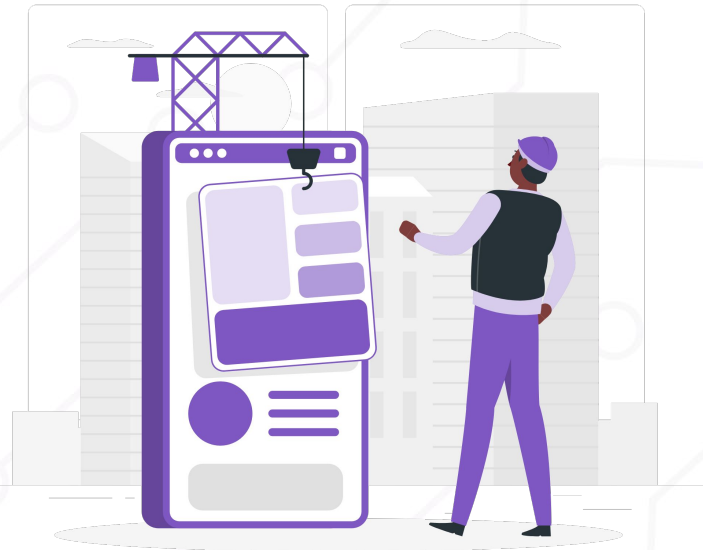
**Programación  
orientada a objetos**



# Intro a la programación orientada a objetos (POO)

La **programación Orientada a Objetos**, también llamada **POO**, se define como un paradigma de programación, con el cual podemos:

1. Invocar la manera en que se consiguen los resultados.
2. La **programación orientada a objetos** disminuye los **errores** y promociona la **reutilización del código**.
3. Es una manera **especial de programar**, que se acerca de alguna manera cómo podemos expresar las cosas en la vida real



# ¿Qué es un paradigma de programación?

Un paradigma de programación es una manera o estilo de programación de software.

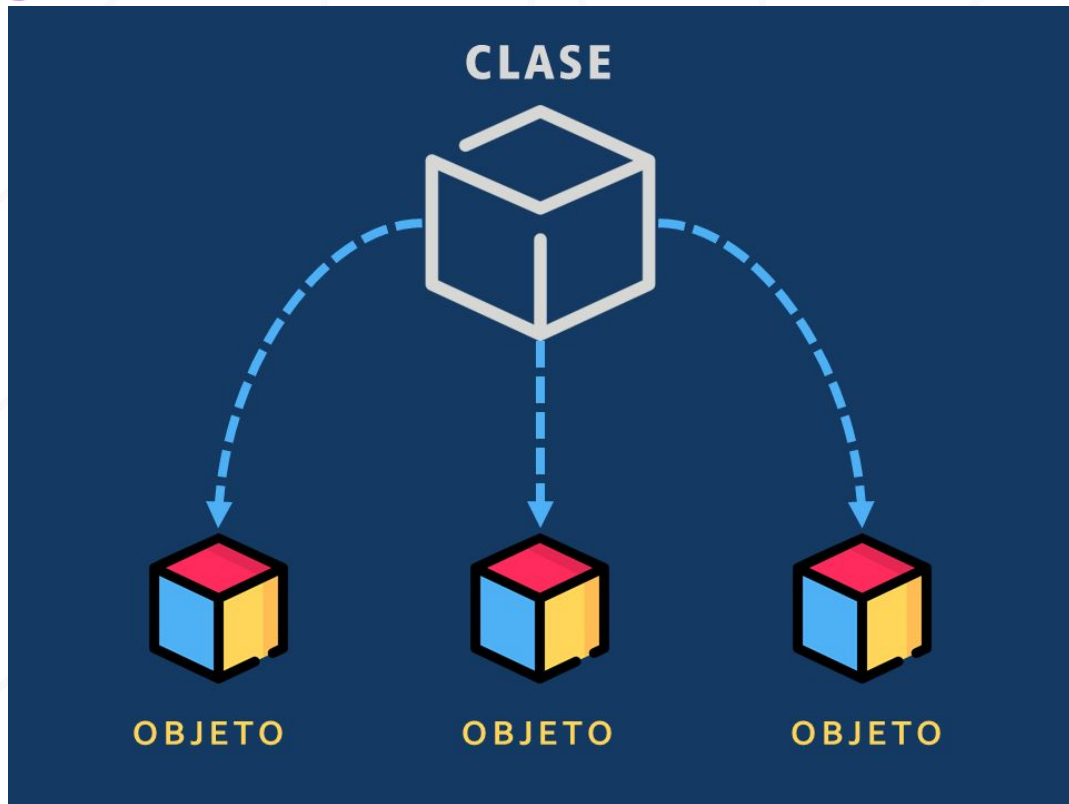
**Programación  
orientada a  
objetos**

En este modelo de paradigma se construyen modelos de objetos que representan elementos (objetos) del problema a resolver, que **tienen características y funciones**.

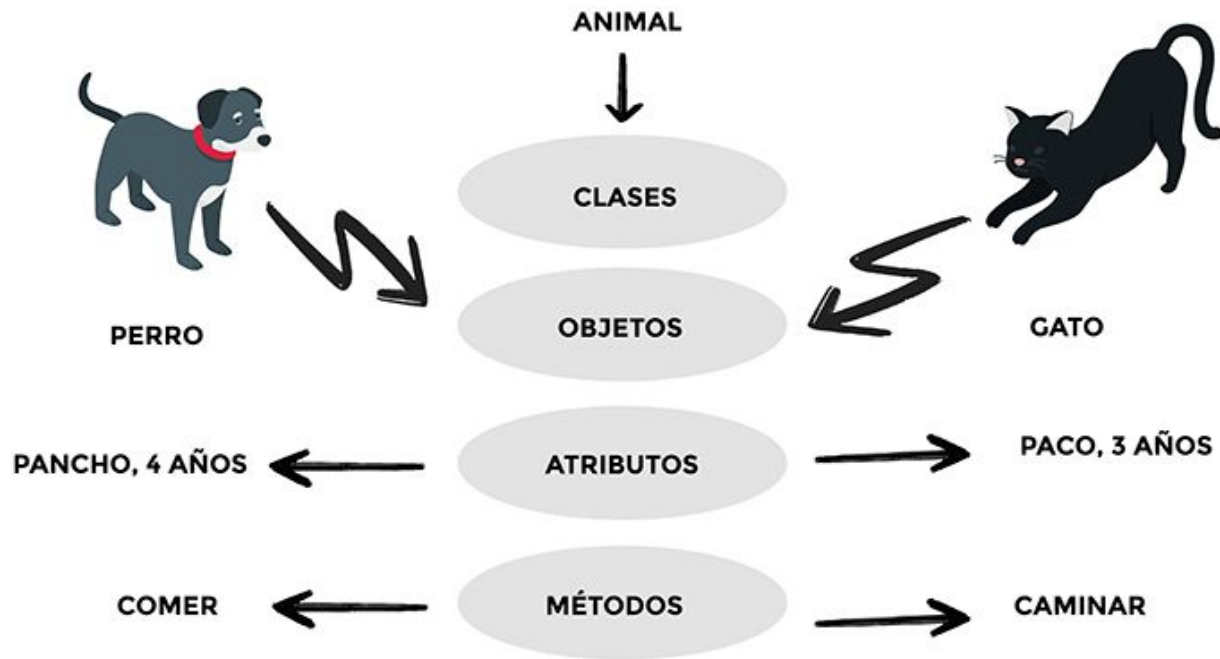
**JavaScript** es multiparadigma.

# Receta para entender POO

Los objetos se crean a partir de una plantilla llamada **clase**, cada objeto es una instancia de su clase

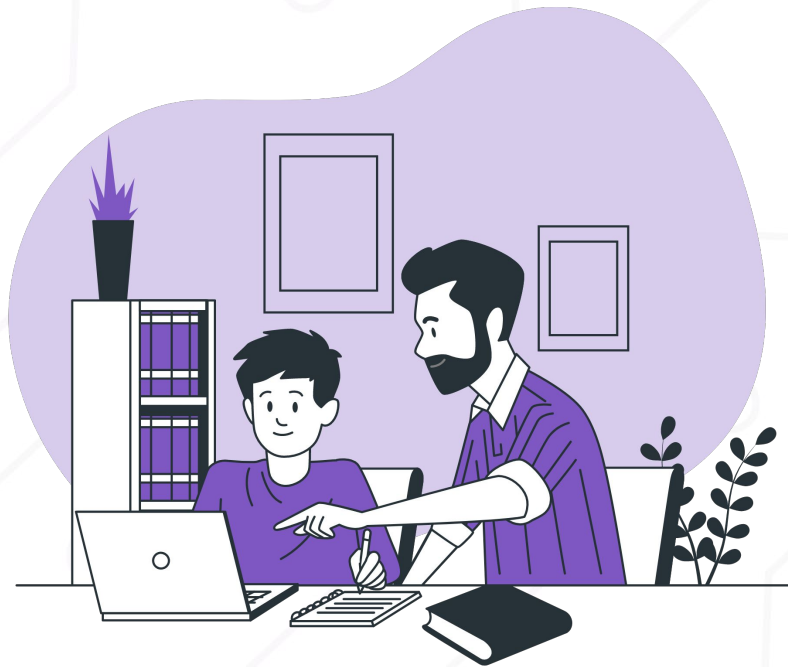


# Descripción Gráfica



# PARA QUE USAMOS POO

La idea básica de la **Programación Orientada a Objetos (POO)** es que usamos objetos para modelar cosas del mundo real que queremos representar en nuestros programas, y/o proveemos una simple manera para acceder a la funcionalidad que, de otra manera, sería difícil o imposible de usar.



# Información que cura

Los objetos pueden contener **información** y **código relacionados**, los cuales representan información acerca de lo que estás tratando de modelar, y la funcionalidad o comportamiento que deseas que tenga.

Los datos de un Objeto (y frecuentemente, también las funciones) se pueden almacenar ordenadamente (la palabra oficial es **encapsular**).

Los objetos también se usan comúnmente como **almacenes de datos** que se pueden enviar fácilmente a través de la red.



# Abstracción del objeto

Definiciones de las propiedades y comportamiento de un tipo de objeto concreto.



## ■ Atributos:

- color
- velocidad
- ruedas
- motor

## ■ Métodos:

- arranca()
- frena()
- dobla()

# Abstracción Visual

C O C I N A

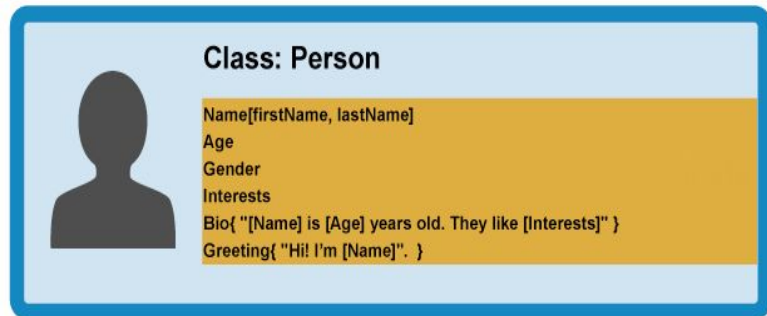




# Abstracción

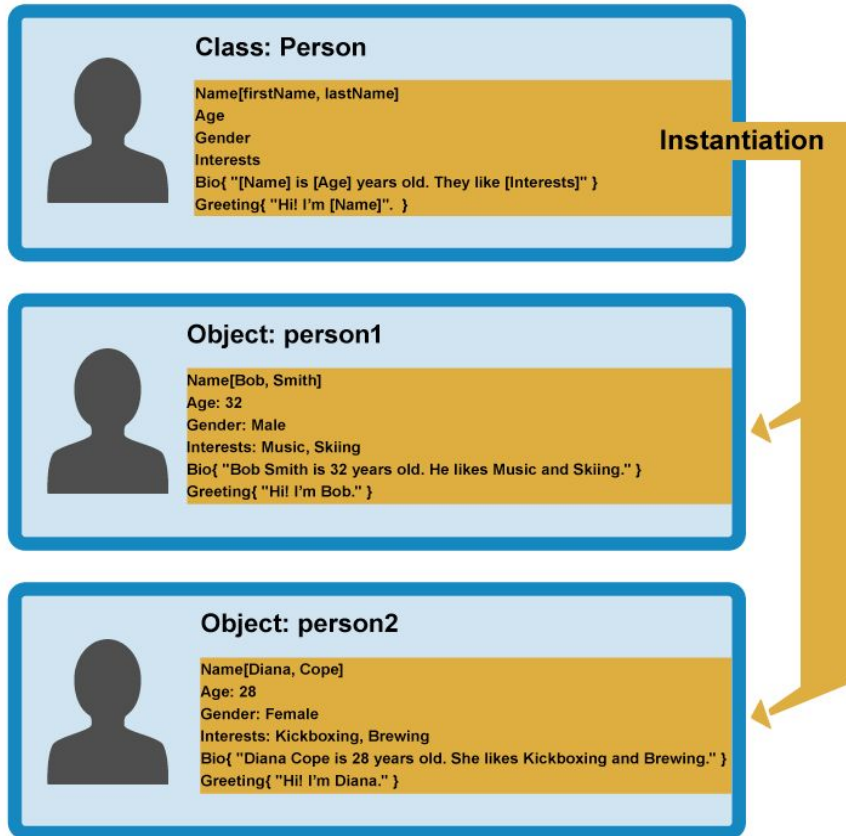
En este caso, solo estamos interesados en mostrar su **nombre**, **apellido**, **calificación final** y **si está cursa el semestre**, además de una pequeña introducción sobre este individuo basada en los datos anteriores.

Esto es conocido como **abstracción** — crear un modelo simple de algo complejo que represente sus aspectos más importantes y que sea fácil de manipular para el propósito de nuestro programa.



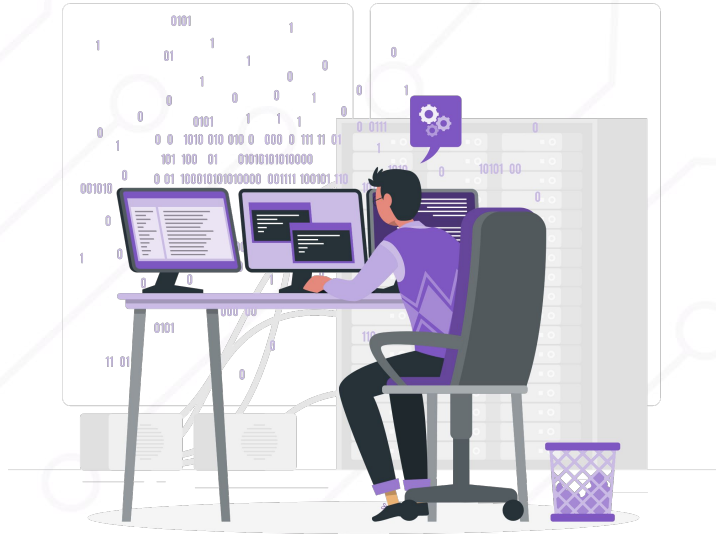
# Creando Objetos

Partiendo de nuestra clase, podemos crear **instancias de objetos** — objetos que contienen los datos y funcionalidades definidas en la clase original. Teniendo a nuestra clase **Person**, ahora podemos crear gente con características más específicas:



# De lo visual al Código.

Vamos a considerar un sencillo programa que muestra información sobre estudiantes y profesores en una escuela. Aquí daremos un vistazo a la **POO (Programación Orientada a Objetos)** en general



# Plasmar la abstracción en Código

```
// ALUMNOS
class Alumnos {

    constructor( nombre, apellido, califFinal, inscrito){
        this.nombre = nombre;
        this.apellido = apellido;
        this.califFinal = califFinal;
        this.inscrito = inscrito;
    }
}
```

# CLASES

Una manera de definir una clase es mediante una declaración de clase. Para declarar una clase, se utiliza la palabra reservada `class` y un nombre para la clase "Alumnos".

```
class Alumnos {  
    constructor(nombre, apellido, califFinal) {  
        this.nombre = nombre;  
        this.apellido = apellido;  
        this.califFinal = califFinal;  
    }  
}
```

esta es una razón por la cual es muy importante usar `this`, para que usen sus propios valores, y no algún otro valor.

# Constructor

El constructor es un método especial que se ejecuta automáticamente cuando se crea una instancia de esa clase.

Su función es inicializar el objeto y sirve para asegurarnos que los objetos siempre contengan valores iniciales válidos.

