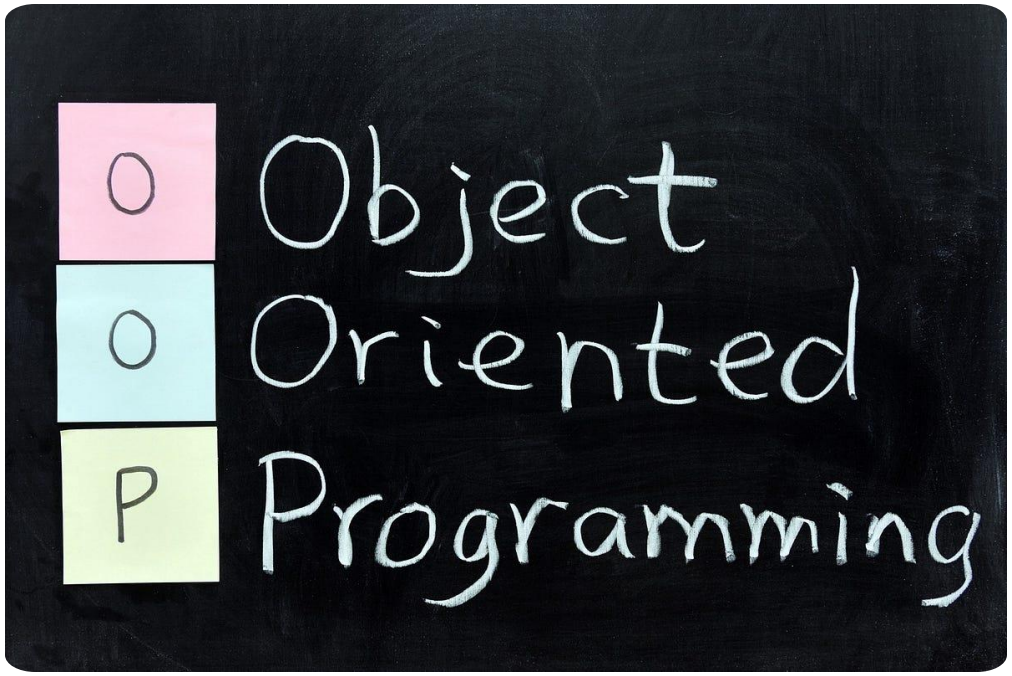
Principios de POO

EJEMPLO EN SWIFT



Conceptos

- Abstracción
- Herencia: clase y superclase
- Polimorfismo
- Encapsulamiento

POO

Clase

Propiedades

Métodos

```
// Abstracción
class Vehiculo {
    var nombre: String
    var marca: String
    var tipo: String

    init(nombre: String, marca:
        String, tipo: String) {
        self.nombre = nombre
        self.marca = marca
        self.tipo = tipo
    }

    func moverse() -> String {
        return ""
    }
}
```

Constructor

métodos especiales que se activan cuando se crea una instancia de un objeto a partir de una clase.

La **abstracción** permite representar características esenciales y relevantes de un objeto

POO

Herencia sirve para crear una clase basada en una ya existente, de tal forma que se aprovechan sus propiedades y métodos favoreciendo la reutilización de código.

Superclase (original)

```
// Abstracción
class Vehiculo {
    var nombre: String
    var marca: String
    var tipo: String

    init(nombre: String, marca:
        String, tipo: String) {
        self.nombre = nombre
        self.marca = marca
        self.tipo = tipo
    }

    func moverse() -> String {
        return ""
    }
}
```

Subclases (heredadas)

```
// Vehículo terrestre
class Terrestre: Vehiculo {
    init(nombre: String, marca: String) {
        super.init(nombre: nombre, marca: marca,
            tipo: "Terrestre")
    }

    override func moverse() -> String {
        return "Rodar"
    }
}
```

```
// Vehículo aéreo
class Aereo: Vehiculo {
    init(nombre: String, marca: String) {
        super.init(nombre: nombre, marca: marca,
            tipo: "Aéreo")
    }

    override func moverse() -> String {
        return "Volar"
    }
}
```

```
// Vehículo marítimo
class Maritimo: Vehiculo {
    init(nombre: String, marca: String) {
        super.init(nombre: nombre, marca: marca,
            tipo: "Marítimo")
    }

    override func moverse() -> String {
        return "Navegar"
    }
}
```

POO

A la función "presentar" podemos enviarle objeto de tipo Vehiculo o un objeto de tipo Aereo, Terrestre o Maritimo

```swift
// Polimorfismo
func presentar(vehiculo: Vehiculo) {
    print("\(vehiculo.nombre) de marca
        \(vehiculo.marca) es un vehículo
        \(vehiculo.tipo) y puede
        \(vehiculo.moverse()).")
}

// Uso de las clases
let avion = Aereo(nombre: "Avión", marca:
    "Boeing")
let coche = Terrestre(nombre: "Coche", marca:
    "Toyota")
let barco = Maritimo(nombre: "Barco", marca:
    "Yamaha")

presentar(vehiculo: avion)  // Avión de marca
    Boeing es un vehículo Aéreo y puede Volar.
presentar(vehiculo: coche)  // Coche de marca
    Toyota es un vehículo Terrestre y puede
    Rodar.
presentar(vehiculo: barco)  // Barco de marca
    Yamaha es un vehículo Marítimo y puede
    Navegar.
```

Polimorfismo permite que objetos de diferentes clases puedan ser tratados como objetos de una clase común, a través de la *herencia* y la implementación de *métodos sobrescritos* en las clases derivadas.

POO

Nivel de protección

Encapsulamiento se basa en solo permitir el acceso a determinados atributos y métodos para ser llamados desde fuera de la clase que los define.

```
class Vehiculo {
    var nombre: String
    var marca: String
    var tipo: String
    private var multas: Int

    init(nombre: String, marca: String, tipo:
        String) {
        self.nombre = nombre
        self.marca = marca
        self.tipo = tipo
        self.multas = 0
    }
    func getMultas() -> Int {
        return multas
    }

    func agregarMulta() {
            multas += 1
    }

    func moverse() -> String {
        return ""
    }
}
```