## 💡 ¿Qué es la Herencia en POO?

La **herencia** es un mecanismo que permite a una clase (subclase) adquirir las propiedades y métodos de otra clase (superclase). Esto permite reutilizar código y crear estructuras jerárquicas.

---

## 🔗 ¿Para qué sirve?

- Reutilización de código.
- Estructuración de clases en una jerarquía lógica.
- Extender funcionalidades sin modificar el código base.

---

## 💻 Ejemplo práctico

js

```
// Clase base
class Animal {
  constructor(nombre) {
    this.nombre = nombre;
  }

  mover() {
    console.log(`${this.nombre} se está moviendo.`);
  }
}

// Subclase que hereda de Animal
class Perro extends Animal {
  ladrar() {
    console.log(`${this.nombre} está ladrando.`);
  }
}

const miPerro = new Perro("Max");
miPerro.mover();    // Max se está moviendo.
miPerro.ladrar();   // Max está ladrando.
```

---

🔗 **Referencias:**

- MDN - Herencia en Clases
- MDN - super()

---

## 🤓 Preguntas orientadoras

1. ¿Qué beneficios aporta la herencia al desarrollo de software?
2. ¿Una subclase puede acceder a métodos privados de la superclase?
3. ¿Es posible sobreescribir un método de la clase padre en una subclase?
4. ¿Para qué se utiliza `super()` en el constructor?
5. ¿Qué sucede si no llamás a `super()` en el constructor de una subclase?

## 💡 ¿Qué es el Polimorfismo?

El **polimorfismo** permite que diferentes objetos respondan al mismo método de formas distintas, dependiendo de su clase.

---

## 🔗 ¿Para qué sirve?

- Crear interfaces genéricas para múltiples tipos de objetos.
- Simplificar código cuando se trabaja con múltiples subclases.

---

## 💻 Ejemplo práctico

js

```
class Figura {
  dibujar() {
    console.log("Dibujando una figura...");
  }
}
```

```
class Circulo extends Figura {
  dibujar() {
    console.log("Dibujando un círculo...");
  }
}

class Cuadrado extends Figura {
  dibujar() {
    console.log("Dibujando un cuadrado...");
  }
}

const figuras = [new Figura(), new Circulo(), new Cuadrado()];
figuras.forEach(figura => figura.dibujar());
```

---

🔗 **Referencias:**

MDN - Polimorfismo

## 🤓 Preguntas orientadoras

1. ¿Qué ventajas aporta el polimorfismo al recorrer listas de objetos?
2. ¿Podemos utilizar un método común para subclases distintas?
3. ¿Cómo se decide qué método se ejecuta en un contexto polimórfico?
4. ¿Qué pasaría si una subclase no sobrescribe un método de la clase base?
5. ¿El polimorfismo mejora la escalabilidad del código? ¿Por qué?

---

## 📎 Recursos

- Documentación de Mozilla Developer Network (MDN)