

# Fundamentos de la Programación Orientada a Objetos (POO)

Diego Alexander Huertas Sosa

23 de octubre de 2025

## 1. Conceptos Fundamentales de POO

### 1.1. El Constructor: Definición y Función

El constructor es un método especial dentro de una clase que se invoca automáticamente al momento de crear (instanciar) un nuevo objeto de esa clase.

Cuadro 1: Función Principal del Constructor

Concepto	Función Principal
Constructor	Su función principal es <b>inicializar</b> el nuevo objeto. Esto implica:
	• <b>Asignar valores iniciales</b> a los atributos (variables) del objeto.
	• Realizar cualquier otra <b>configuración</b> necesaria para que el objeto esté listo para su uso.

#### Ejemplo:

```
class Persona {
    String nombre;
    int edad;

    // Constructor
    Persona(String n, int e) {
        nombre = n;
        edad = e;
    }
}
```

### 1.2. Objeto: Definición y Función

Un objeto es una instancia de una clase. Es la entidad fundamental en la POO y representa un elemento del mundo real (o conceptual) dentro del programa.

#### Ejemplo:

Cuadro 2: Función Principal del Objeto

Concepto	Función Principal
Objeto	Representa una entidad con <b>estado</b> y <b>comportamiento</b> .
	• <b>Estado:</b> Valores de sus <b>atributos</b> (datos/propiedades).
	• <b>Comportamiento:</b> Sus <b>métodos</b> (funciones/acciones que puede realizar).

```
Persona persona1 = new Persona("Ana", 30);
```

### 1.3. Clase: Definición y Función

Una clase es una plantilla, molde o tipo para crear objetos. Define la estructura (atributos) y el comportamiento (métodos) comunes que tendrán todos los objetos de ese tipo.

- ★ La clase es la abstracción que define las características que tendrán sus objetos.
- ★ Define la estructura de datos y el código que compartirán todas las instancias.

## 2. Diferencias y Modificadores

Ejemplo:

```
class Persona {
    String nombre;
    int edad;

    void saludar() {
        System.out.println("Hola, mi nombre es " + nombre);
    }
}
```

### 2.1. Diferencias Destacadas: POO vs. Programación Estructurada (PE)

Las principales diferencias radican en la organización del código y el manejo de los datos.

### 2.2. Uso y Razón de los Modificadores de Acceso

Los modificadores de acceso establecen la visibilidad y el alcance de los miembros de una clase (atributos y métodos).

Cuadro 3: Comparación POO vs. Programación Estructurada

Característica	Programación Orientada a Objetos (POO)	Programación Estructurada (PE)
<b>Organización</b>	Se centra en <b>objetos</b> que agrupan <b>datos</b> y <b>funciones</b> (métodos).	Se centra en la <b>lógica</b> y las <b>funciones</b> (procedimientos).
<b>Datos vs. Lógica</b>	Datos y lógica <b>encapsulados</b> dentro de los objetos ( <i>Encapsulamiento</i> ).	Datos y lógica a menudo <b>separados</b> . Funciones manipulan estructuras de datos.
<b>Reutilización</b>	<b>Alta</b> gracias a <b>Herencia</b> y <b>Polimorfismo</b> .	Se logra a través de la llamada a <b>funciones</b> (procedimientos).
<b>Mantenimiento</b>	<b>Más fácil</b> de mantener y es- calar, cambios localizados.	Puede ser <b>difícil</b> en proyectos grandes debido a datos globales.

### 2.2.1. Propósito (¿Para qué y por qué se usan?)

Se usan para implementar el principio de **Encapsulamiento**, esencial para la POO.

1. **Protección de Datos:** Evitan que el estado interno de un objeto (atributos) sea modificado de forma no controlada.
2. **Integridad y Coherencia:** Al restringir el acceso ('private'), se obliga a usar métodos públicos que pueden validar los datos, asegurando que el objeto permanezca en un estado válido.
3. **Interfaz Clara:** Ocultan los detalles de implementación al mundo exterior.

### 2.2.2. Ejemplos Comunes

Cuadro 4: Ejemplos de Modificadores de Acceso

Modificador	Alcance de la Visibilidad
<code>public</code>	Accesible desde <b>cualquier lugar</b> .
<code>private</code>	Accesible <b>solo dentro de la propia clase</b> donde fue declarado.
<code>protected</code>	Accesible dentro de la propia clase <b>y por sus sub-clases</b> (clases que heredan de ella).

**Ejemplo:**

```
class Persona {  
    private String nombre; // protegido  
    public void setNombre(String n) { nombre = n; } // acceso controlado  
}
```