Fundamentos de la Programación Orientada a Objetos (POO)

Asistente de IA

22 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 **inicializar** el nuevo objeto. Esto implica:
	• **Asignar valores iniciales** a los atributos (variables) del objeto.
	• Realizar cualquier otra **configuración** 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 **estado** y **comportamiento**.
	• **Estado:** Valores de sus **atributos** (datos/propiedades).
	• **Comportamiento:** Sus **métodos** (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)**
Organización	Se centra en **objetos** que agrupan **datos** y **funciones** (métodos).	Se centra en la **lógica** y las **funciones** (procedimientos).
Datos vs. Lógica	Datos y lógica **encapsulados** dentro de los objetos (*Encapsulamiento*).	Datos y lógica a menudo **separados**. Funciones manipulan estructuras de datos.
Reutilización	**Alta** gracias a **Herencia** y **Polimorfismo**.	Se logra a través de la llamada a **funciones** (procedimientos).
Mantenimiento	**Más fácil** de mantener y escalar, cambios localizados.	Puede ser **difícil** 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**
`public`	Accesible desde **cualquier lugar**.
`private`	Accesible **solo dentro de la propia clase** donde fue declarado.
`protected`	Accesible dentro de la propia clase **y por sus subclases** (clases que heredan de ella).

Ejemplo:

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