Paradigmas de la programación Programación Orientada a Objetos (POO).

Integrantes: Camila Rivas, Vicente Saavedra, Isabel Espinoza, Luis Gutiérrez, Stephany Vargas, Juan Opazo

¿En qué consiste este paradigma de la programación?

La Programación Orientada a Objetos (POO) es un paradigma de programación que organiza el código en objetos.

Los objetos son instancias que heredan de clases y la finalidad de estos es representar entidades del mundo real.

Cada objeto tiene atributos y métodos provenientes de la clase a la que pertenecen.

Conceptos clave:

1. Clase:
    a. Son plantillas o moldes para crear objetos.
    b. Definen los atributos (datos) y métodos (acciones) que tendrán los objetos.
    c. Ejemplo: Una clase Coche con atributos como color y marca, y métodos como acelerar() y frenar().
2. Objeto:
    a. Son instancias de una clase.
    b. Cada objeto tiene sus propios valores para los atributos.
    c. Ejemplo: Un objeto miAuto de la clase Coche con color = rojo y marca = Toyota.

Por ejemplo:
La clase Persona tiene los atributos que definen a una persona como el nombre, edad, dirección.
Los métodos definidos en esta clase persona corresponden a funciones básicas que pueden realizar, como caminar, correr, sentarse, etc.

Si creo un objeto de tipo persona, esta persona tendrá los atributos y métodos provenientes de dicha clase.

La POO se basa en cuatro principios fundamentales:

1. **Abstracción:**
    a. Representa conceptos del mundo real a través de clases y objetos. Oculta detalles internos y muestra solo lo esencial para el uso del objeto.
    b. Ejemplo: Un `Smartphone` tiene muchos componentes internos, pero el usuario solo usa métodos como `hacerLlamada()`.
2. **Encapsulamiento:**
    a. Protege los datos dentro de los objetos y solo permite el acceso a través de métodos específicos.
    b. Ejemplo: Un atributo `saldo` en una cuenta bancaria que solo puede modificarse con métodos como `depositar()` o `retirar()`.
3. **Herencia:**
    a. Permite crear nuevas clases basadas en otras, reutilizando código y facilitando la extensión de funcionalidades.
    b. Ejemplo: Una clase `Vehiculo` con atributos generales (`ruedas`, `motor`), y una clase `Moto` que hereda de `Vehiculo` pero agrega `tipoDeCasco`.
4. **Polimorfismo:**
    a. Permite que un mismo método pueda comportarse de diferentes maneras según el contexto, es decir, según el objeto que lo utilice.
    b. Ejemplo: Un método `hacerSonido()` en una clase `Animal`, que puede representar un `maullido` en `Gato` y un `ladrido` en `Perro`.

¿Qué problema o problemas intenta resolver?

La POO busca mejorar la modularidad, **reutilización de código** y mantenimiento de programas complejos de forma escalable. Algunos problemas que resuelve incluyen:

1. **Código desorganizado y difícil de mantener:**
    a. Facilita la estructuración del código en módulos independientes.
2. **Duplicación de código**:
    a. Gracias a la herencia y la reutilización, se evita repetir código innecesariamente.
3. **Escalabilidad y flexibilidad limitadas:**
    a. Al ser un modelo modular, permite agregar nuevas funcionalidades sin afectar otras partes del sistema.
4. **Dificultad en modelar problemas reales:**
    a. La POO se asemeja a la forma en que pensamos en objetos y relaciones en el mundo real.

Ventajas de la POO:

- Reutilización de código mediante la herencia.
- Modularidad, lo que facilita la organización del código.
- Mantenimiento más sencillo y escalabilidad.

- Mayor seguridad gracias al encapsulamiento.

Háblanos brevemente sobre su historia.

La **Programación Orientada a Objetos (POO)** tiene sus raíces en la década de **1960**, cuando Ole-Johan Dahl y Kristen Nygaard desarrollaron el lenguaje **Simula** en Noruega. Simula introdujo el concepto de **clases y objetos**, permitiendo modelar sistemas complejos de forma más natural.

En los años 70, **Smalltalk**, creado en Xerox PARC, refinó estos conceptos al introducir **herencia y polimorfismo**, estableciendo las bases de la POO moderna.

Durante los 80 y 90, lenguajes como **C++**, **Java** y **Python** popularizaron la POO, integrándola en la industria del software. Hoy, sigue siendo un paradigma fundamental, utilizado en lenguajes modernos como **C#, Swift y Kotlin**.

¿Qué lenguajes de programación lo implementan? (recuerda que muchos lenguajes son multiparadigma).

Java: Uno de los lenguajes más conocidos por su enfoque en POO. Todo en Java está basado en clases y objetos.

C++: Extiende a C con características de POO, como clases, herencia, y polimorfismo.

Python: Aunque no es estrictamente orientado a objetos, Python soporta completamente la POO y permite trabajar con clases y objetos de manera sencilla.

C#: Similar a Java, C# está diseñado principalmente para trabajar con POO. Es un lenguaje de programación desarrollado por Microsoft.

Ruby: Es un lenguaje completamente orientado a objetos, donde incluso los tipos básicos como números y cadenas son objetos.

Swift: El lenguaje de programación de Apple para aplicaciones iOS y macOS, basado en POO.

Objective-C: Aunque tiene raíces en C, este lenguaje es conocido por su fuerte enfoque en POO, especialmente en el desarrollo para macOS e iOS.

JavaScript: A pesar de ser un lenguaje funcional, JavaScript soporta POO mediante la creación de objetos y el uso de clases (en versiones más recientes).

PHP: Desde la versión 5, PHP incluye soporte completo para POO, permitiendo la creación de clases y objetos.

Kotlin: Lenguaje moderno para aplicaciones Android que también es completamente compatible con POO.

¿En qué escenario hipotético utilizarías este paradigma?

Existen muchos escenarios hipotéticos en el que podríamos utilizar este paradigma, pero uno de ellos sería, por ejemplo, un sistema de recomendación de películas y series como en netflix;

Podríamos tener clases como ***usuario, producto y recomendación*.**

La clase ***usuario*** podría almacenar los productos que se han visto, y la clase ***producto*** tendría información como nombre, categoría y calificación. El sistema de recomendaciones podría analizar el historial del usuario y generar recomendaciones personalizadas.

Juego de rol (RPG)

- Clases: `Personaje`, `Enemigo`, `Arma`, `Habilidad`.
- Objetos: Un mago con hechizos de fuego, un guerrero con espada.
- Métodos: Atacar, defender, subir de nivel.

Sistema de gestión de una tienda online

- Clases: `Producto`, `CarritoDeCompras`, `Usuario`, `Pedido`.
- Objetos: Un usuario con un carrito lleno de productos electrónicos.
- Métodos: Agregar producto al carrito, calcular total, procesar pago.