PROGRAMACIÓN ORIENTADA A OBJETOS

La Programación Orientada a Objetos (POO) es un paradigma de programación que se basa en el concepto de "objetos", los cuales son entidades que encapsulan datos y comportamientos. Esta metodología de desarrollo de software se centra en organizar el código en torno a objetos que interactúan entre sí, permitiendo una mejor modularidad, reutilización y mantenimiento del código.

Principios fundamentales

1. **Abstracción**: la POO permite abstraer los detalles complejos de un sistema, enfocándose en las características esenciales relevantes para el problema que se está resolviendo.

2. **Encapsulamiento**: los objetos encapsulan sus datos y comportamientos, lo que significa que solo revelan una interfaz pública mientras ocultan los detalles internos de su implementación. Esto promueve la modularidad y reduce la complejidad del sistema.

3. **Herencia**: la herencia permite la creación de nuevas clases basadas en clases existentes, lo que facilita la reutilización del código y la creación de relaciones jerárquicas entre los objetos.

4. **Polimorfismo**: permite que un objeto pueda comportarse de diferentes maneras dependiendo del contexto. Esto se logra mediante el uso de métodos con el mismo nombre pero con diferentes implementaciones en las clases derivadas.

Componentes de la POO

1. **Clases**: son plantillas para la creación de objetos. Definen las propiedades (atributos) y comportamientos (métodos) comunes a un conjunto de objetos.

2. **Objetos**: son instancias de una clase específica. Cada objeto tiene su propio conjunto único de valores para los atributos de la clase y puede invocar los métodos definidos en esa clase.

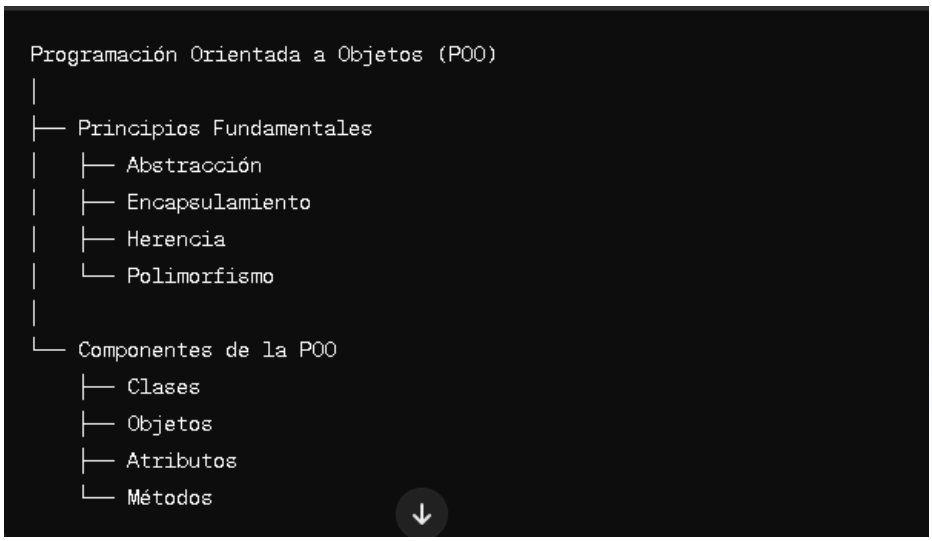
3. **Atributos**: representan las características de un objeto y se definen como variables dentro de una clase.

4. **Métodos**: son funciones asociadas a una clase que definen su comportamiento. Los métodos pueden acceder y modificar los atributos de un objeto, así como realizar otras acciones específicas.

Ejemplo práctico

Supongamos que queremos modelar un sistema de gestión de empleados. Podríamos tener una clase **Empleado** que representa a un empleado con atributos como nombre, edad y salario. Podríamos definir métodos para calcular el salario anual, aumentar el salario, etc. Luego, podríamos crear objetos individuales de esta clase para cada empleado en nuestra organización.

Esquema



En la Programación Orientada a Objetos (POO), las clases pueden tener propiedades, métodos y eventos que definen su comportamiento y características. Aquí está una explicación de cada uno de estos conceptos:

Propiedades

Las propiedades son variables asociadas a un objeto que representan su estado o características. Estas propiedades definen los datos que un objeto puede contener. Por ejemplo, en una clase **Coche**, las propiedades podrían incluir la marca, el modelo, el año de fabricación y el color. Las propiedades también pueden tener modificadores de acceso que especifican si pueden ser accedidas o modificadas desde fuera de la clase.

Métodos

Los métodos son funciones asociadas a una clase que definen su comportamiento. Estos métodos pueden realizar acciones específicas sobre las propiedades de un objeto o realizar operaciones más complejas. Por ejemplo, en una clase **Coche**, podríamos tener métodos como **arrancar()**, **detener()**, **acelerar()**, **frenar()** que controlan el funcionamiento del vehículo. Los métodos también pueden tener parámetros que les permiten recibir información necesaria para realizar su tarea.

Eventos

Los eventos son señales que indican que algo ha ocurrido dentro de un objeto. Estos eventos pueden ser generados por el propio objeto o por acciones externas. Por ejemplo, en una interfaz gráfica de usuario, un botón podría generar un evento de clic cuando se presiona. En la POO, los eventos se utilizan para permitir que los objetos interactúen entre sí y con el entorno externo de manera dinámica. Los eventos pueden ser manejados por métodos específicos llamados "manejadores de eventos" que responden a la ocurrencia de un evento particular.

Ejemplo práctico

Continuando con el ejemplo del **Coche**, podríamos tener propiedades como **marca**, **modelo**, **año**, **color**, y métodos como **arrancar()**, **detener()**, **acelerar()**, **frenar()**. Además, podríamos tener eventos como **puertaAbierta**, **lucesEncendidas**, que se activan cuando ocurren ciertas acciones, como abrir una puerta o encender las luces del coche.

Referencias:

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley.

Eckel, B. (2003). Thinking in Java. Prentice Hall.

Booch, G., Rumbaugh, J., & Jacobson, I. (2005). Unified Modeling Language User Guide, The (2nd Edition). Addison-Wesley Professional.