Taller de conceptos y principios de programación orientada a objetos GA4-220501095-AA2-EV01

Aprendices:

Julián Mateo Ríos Gallego

Allison Ivana Vecino Rivero

Álvaro Antonio Quintero Cuervo

Haiver Darío Díaz Toro

Servicio Nacional de Aprendizaje - SENA

Análisis y Desarrollo de Software (2721490)

Instructor:

Juan Fernando Jiménez Rodríguez

Tunja, colombia

INTRODUCCIÓN

La programación orientada a objetos (POO) es un paradigma que se basa en la organización del código en torno a objetos, entidades que contienen datos y funciones. Sus principios fundamentales son la encapsulación, la herencia y el polimorfismo. La encapsulación permite ocultar los detalles internos de un objeto, la herencia facilita la creación de nuevas clases basadas en clases existentes y el polimorfismo permite que objetos de diferentes clases sean tratados de manera uniforme. Estos principios proporcionan un enfoque estructurado, reutilizable y modular para el desarrollo de software lo veremos a continuación en el desarrollo de la evidencia

Conceptos de la POO

1. **Clases:**

- Las clases son plantillas que definen las propiedades y comportamientos de los objetos. Son como moldes para crear objetos.

2. **Objetos**

- Los objetos son instancias de una clase, es decir, son entidades concretas que contienen datos (llamados atributos) y funcionalidades (llamadas métodos).

3. **Atributos:**

- Son las características o datos que describen un objeto. Por ejemplo, en una clase "Coche", los atributos podrían ser el color, la marca, el modelo, etc.

4. **Métodos:**

- Son funciones o comportamientos que pueden realizar los objetos. Por ejemplo, en la misma clase "Coche", los métodos podrían ser "arrancar()", "detenerse()", etc.

Principios de la POO

Encapsulación:
Este principio se refiere a la ocultación de los detalles internos de un objeto, protegiendo los datos mediante el uso de modificadores de acceso (público, privado, protegido) para controlar la exposición y manipulación de los atributos y métodos de una clase.

Herencia:
La herencia permite que una clase herede atributos y métodos de otra clase. La clase que hereda se denomina subclase o clase hija, mientras que la clase de la que hereda se conoce como superclase o clase padre. Facilita la reutilización del código y la creación de jerarquías de clases.

Polimorfismo:
Es la capacidad de objetos de distintas clases de responder al mismo mensaje (método) de maneras diferentes. Se puede lograr a través de la sobrecarga de métodos, la sobreescritura de métodos y el uso de interfaces y clases abstractas.

Características de la POO

Abstracción:
La abstracción permite centrarse en los aspectos esenciales de un objeto, ignorando los detalles menos relevantes. Las clases abstraen entidades del mundo real, definiendo solo las propiedades y comportamientos relevantes para el sistema.

Modularidad:
La POO fomenta la creación de código modular, dividido en partes más pequeñas (clases). Esto facilita el mantenimiento, la comprensión y la reutilización del código al separar responsabilidades y funcionalidades.

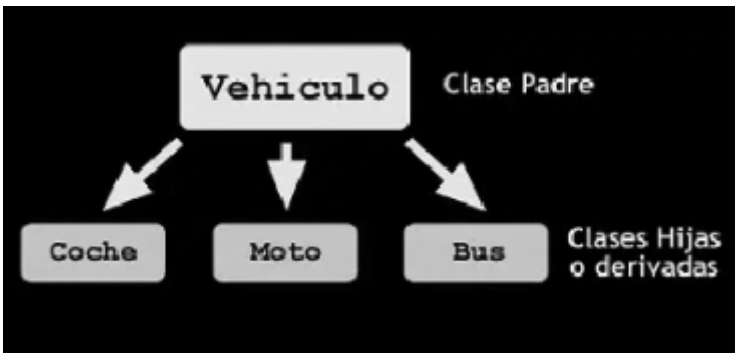
Reusabilidad:
Gracias a la herencia y la creación de clases genéricas, se promueve la reutilización del código. Las clases pueden extender o especializar funcionalidades existentes, lo que ahorra tiempo y esfuerzo en el desarrollo.

Estos conceptos, principios y características son fundamentales en la POO y proporcionan un marco sólido para la creación de sistemas de software más estructurados, flexibles y fáciles de mantener.

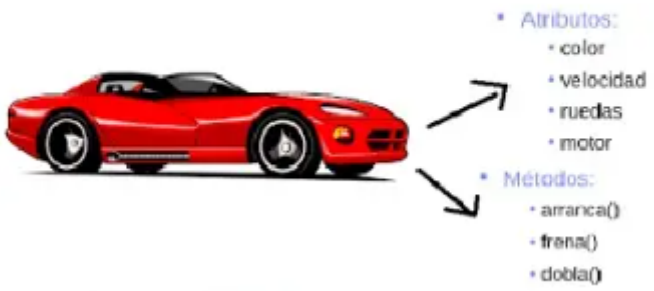
GLOSARIO CON SU USO

1. Clase:
   - Definición: Son plantillas o moldes que definen la estructura y el comportamiento de los objetos. Una clase puede tener atributos (variables) y métodos (funciones). Las clases son la base de la Programación Orientada a Objetos y permiten organizar y reutilizar el código de manera eficiente

   - Uso: Las clases son los planos para la creación de objetos y definen la estructura y funcionalidad de los mismos.



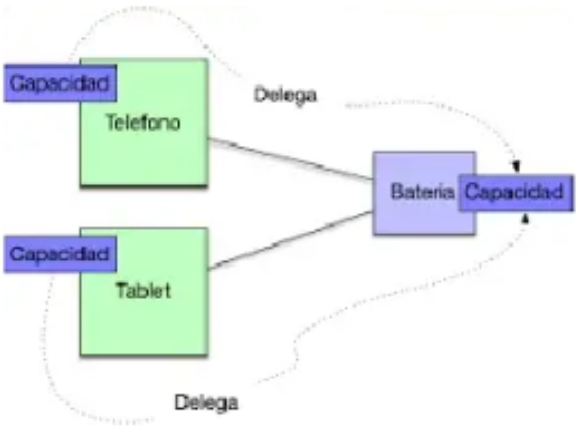
2. Objeto:
   - Definición:Son instancias de una clase, es decir, son elementos concretos que se crean a partir de una plantilla (clase). Los objetos tienen atributos y pueden realizar acciones a través de métodos. Cada objeto tiene su propio estado y comportamiento, pero comparte la estructura y funcionalidad definida en su clase.
  - Uso: Los objetos representan entidades del mundo real y permiten trabajar con datos y funciones específicas.



3. Atributos:
- Definición: Son variables que pertenecen a una clase u objeto. Los atributos almacenan información que caracteriza a la clase u objeto.Pueden ser variables simples (como números o cadenas de texto) o incluso otros objetos. Los atributos definen el estado de un objeto y pueden ser modificados o consultados a través de los métodos
- Uso: Los atributos almacenan datos que caracterizan a un objeto, como nombre, edad, color, etc.

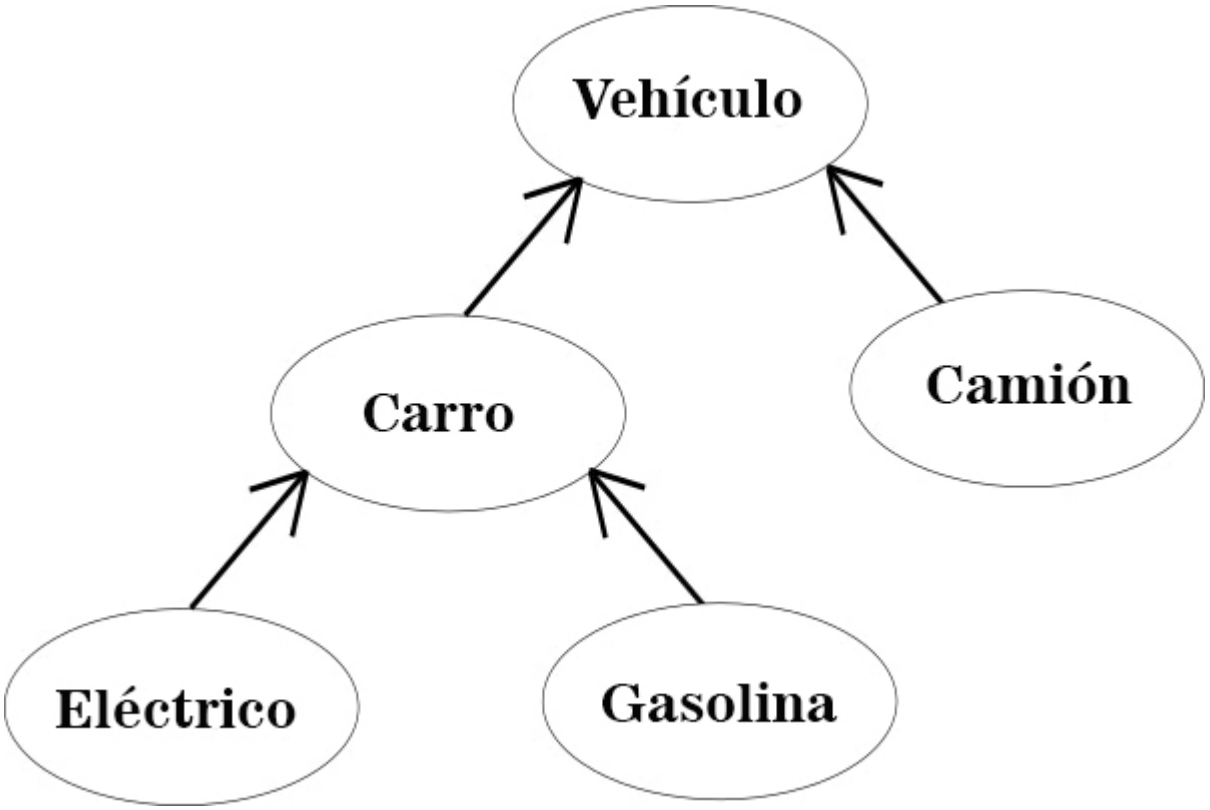
4. Métodos:
- Definición: Son funciones o procedimientos que definen el comportamiento de una clase. Los métodos pueden recibir parámetros y devolver resultados. Los métodos permiten a los objetos realizar acciones específicas y manipular sus atributos. También pueden interactuar con otros objetos y realizar operaciones más complejas
- Uso: Los métodos permiten a los objetos realizar tareas específicas, como cálculos, interacciones o modificaciones de datos.

5. Encapsulación:
- Definición: Es el principio de la POO que consiste en ocultar los detalles internos de una clase y proporcionar una interfaz pública para interactuar con ella. El encapsulamiento permite proteger los datos y el comportamiento de una clase, evitando el acceso directo a sus atributos y métodos. Solo se exponen los elementos necesarios para interactuar con la clase de manera segura y controlada
- Uso: La encapsulación protege los datos y asegura que solo se acceda a ellos de manera controlada, mejorando la seguridad y la integridad del código.

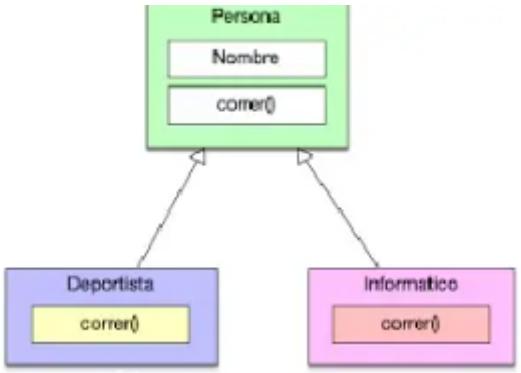


6. Herencia:

- Definición: Es un mecanismo que permite la creación de nuevas clases a partir de clases existentes, aprovechando y extendiendo su funcionalidad. La clase que se hereda se conoce como clase padre o superclase, y la clase que hereda se conoce como clase hija o subclase. La herencia permite crear jerarquías de clases y facilita la reutilización de código
- Uso: La herencia promueve la reutilización de código y la creación de jerarquías de clases, lo que simplifica el diseño y el mantenimiento del software.



7. Polimorfismo:
- Definición: Es la capacidad de un objeto de tomar diferentes formas o comportamientos según el contexto. El polimorfismo permite tratar objetos de diferentes clases de manera uniforme, siempre y cuando compartan una interfaz común. Esto facilita la flexibilidad y la extensibilidad del código, ya que se pueden agregar nuevas clases sin modificar el código existente
- Uso: El polimorfismo permite tratar objetos de manera uniforme, lo que facilita la creación de interfaces genéricas y flexibles.



8. Abstracción:
- Definición: Es el proceso de identificar las características esenciales de un objeto y representarlas en una clase. La abstracción permite simplificar y organizar el código, ocultando los detalles innecesarios. En lugar de preocuparse por las implementaciones internas, los programadores se
Se centran en los aspectos conceptuales y funcionales de los objetos.
- Uso: La abstracción simplifica la representación de objetos del mundo real, permitiendo definir solo lo relevante para el sistema.

9. Modularidad:
- Definición: La subdivisión del código en módulos o clases más pequeñas y autocontenidas.
- Uso: La modularidad mejora la organización y la legibilidad del código al separar las responsabilidades y funcionalidades en unidades más manejables.

10. Reusabilidad:
- Definición: La capacidad de reutilizar clases y componentes en diferentes partes de una aplicación o en otros proyectos.
- Uso: La reutilización ahorra tiempo y esfuerzo al aprovechar el código existente para construir nuevas funcionalidades.

CONCLUSIÓN

La Programación Orientada a Objetos (POO) es un paradigma fundamental en el desarrollo de software que proporciona una estructura poderosa y eficiente para la creación de aplicaciones complejas como lo vimos en el desarrollo de la evidencia. A través de la representación de entidades del mundo real como objetos con propiedades y comportamientos, la POO permite un enfoque modular y flexible para el diseño de sistemas.

En la POO, las clases actúan como plantillas para la creación de objetos, definiendo sus atributos (características) y métodos (comportamientos). Este enfoque facilita la abstracción, permitiendo a los desarrolladores centrarse en los aspectos relevantes de un sistema y omitir los detalles menos esenciales. Además, la encapsulación, uno de sus principios fundamentales, garantiza la seguridad y la integridad del código al ocultar los detalles internos de un objeto y exponer sólo lo necesario a través de modificadores de acceso.

En resumen podemos concluir que la Programación Orientada a Objetos no solo se trata de una manera de escribir código, sino de un enfoque integral que promueve la eficiencia, el mantenimiento y la escalabilidad del software. Comprender y aplicar los conceptos, principios y características de la POO es fundamental para desarrollar sistemas sólidos, flexibles y fáciles de mantener, lo que se traduce en un proceso de desarrollo más efectivo y en la creación de aplicaciones robustas y confiables.

BIBLIOGRAFÍA

- ➔ Herencia en programación: características, tipos, ejemplos (lifeder.com)
- ➔ Programación Orientada a Objetos (POO) | AppMaster
- ➔ ¿Qué es la Programación Orientada a Objetos (POO)? | EDteam
- ➔ Programación orientada a objetos: qué es, conceptos y lenguajes (assemblerinstitute.com)