



OOP OBJECT ORIENTED PROGRAMMING

Celia García Castilla



POR QUÉ OOP

Ya tenemos una estructura, unos estilos y unas funcionalidades en la aplicación web, ahora necesitamos una forma ágil de manejar cantidades de datos cada vez mayores, es decir necesitamos...

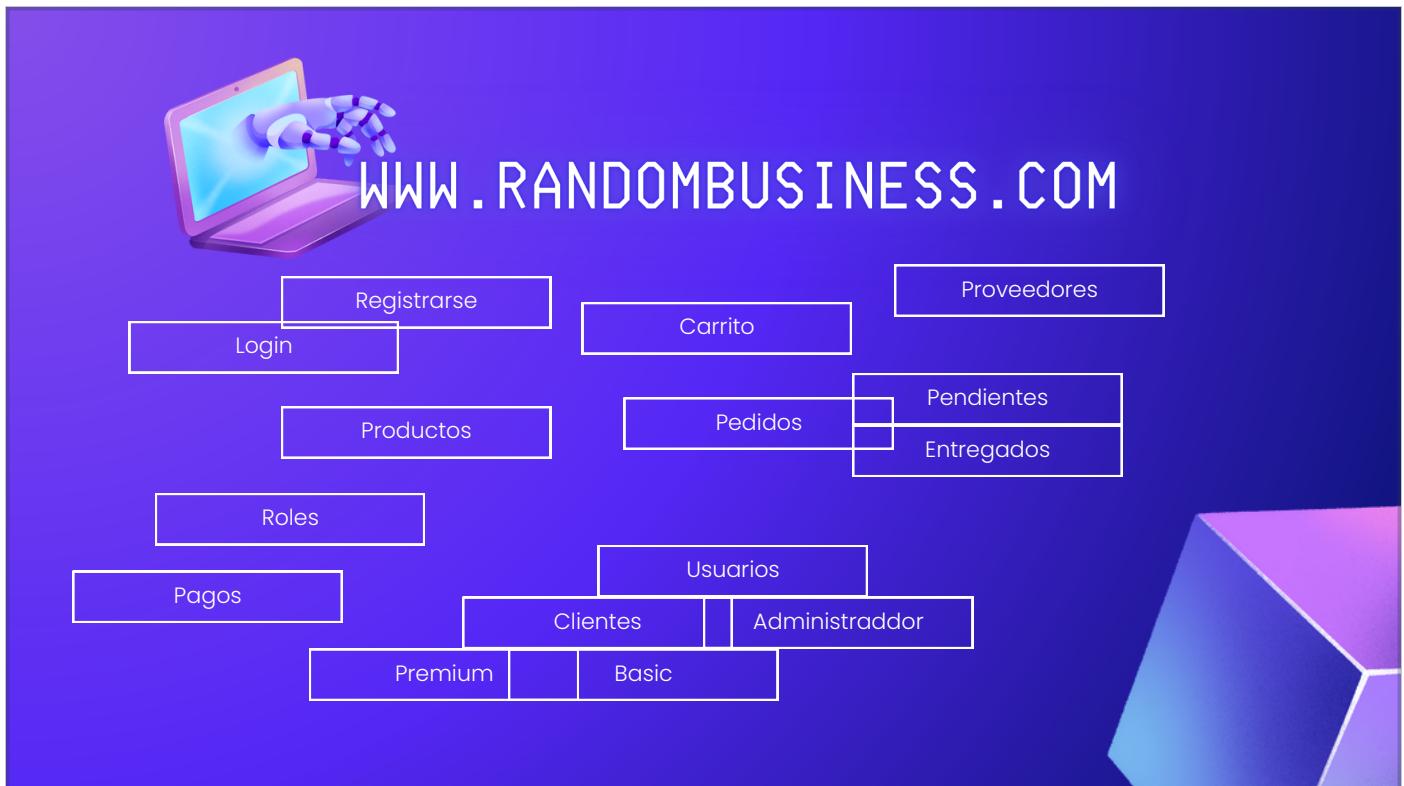
У нас уже есть структура, стили и функциональность в веб-приложении, теперь нам нужен гибкий способ обработки все больших объемов данных, то есть нам нужно...

Un paradigma de programación que no se enfoque en las funciones y la lógica necesaria para manipular datos, sino en estructurar esos datos para facilitar su manipulación

=

OOP

Парадигма программирования, которая не сосредотачивается на функциях и необходимой логике для манипулирования данными, а структурирует сами данные для облегчения их манипуляции.



Imaginemos que tenemos la web de un negocio cualquiera.

La cantidad de información que se almacena, tanto pública como interna o privada es enorme: Usuarios, usuarios cliente, usuarios administradores, roles de usuario, productos, pedidos, pagos, proveedores, etc

Представим, что у нас есть веб-сайт какого-либо бизнеса.

Объем информации, которая хранится, как публичная, так и внутренняя или конфиденциальная, огромен: пользователи, клиенты-пользователи, администраторы, роли пользователей, продукты, заказы, платежи, поставщики и т. д.



Para estructurar esta información y poder manejarla, en la OOP se realiza un proceso de abstracción que convierte esta nube de acciones y datos en objetos.

Cada objeto es un conjunto de datos, llamados ATRIBUTOS, y funcionalidades, llamadas MÉTODOS, asociados a un elemento del mundo real que constituye una entidad dentro de la aplicación web.

Este proceso tiene algo de creativo, pues aunque hay convenciones y buenas prácticas que lo hacen homogéneo, el resultado va a depender del enfoque y las habilidades del programador.

Для структурирования этой информации и ее управления в ООП выполняется процесс абстракции, который превращает эту облачность действий и данных в объекты.

Каждый объект представляет собой набор данных, называемых атрибутами, и функциональностей, называемых методами, связанных с элементом реального мира, который является сущностью веб-приложения.

Этот процесс имеет элемент творчества, поскольку, хотя существуют соглашения и best practices заставляют его однородным, результат зависит от подхода и навыков программиста.



La estructura del objeto la determinamos con un molde o plantilla llamado clase. En la clase definimos qué ATRIBUTOS Y MÉTODOS tiene un objeto, y a partir de la clase creamos tantos OBJETOS como sea necesario. Es decir, que cuando creamos un objeto estamos creando una INSTANCIA de una clase. Cada objeto o instancia de una misma clase tendrá la misma estructura, pero las propiedades y la forma en que se desarrollan los métodos variarán de un objeto a otro.

Структуру объекта мы определяем с помощью шаблона или шаблона, называемого классом. В классе мы определяем, какие АТРИБУТЫ И МЕТОДЫ имеет объект, и на основе класса мы создаемолько ОБЪЕКТОВ, сколько необходимо. Другими словами, когда мы создаем объект, мы создаем ЭКЗЕМПЛЯР класса. Каждый объект или экземпляр одного и того же класса будет иметь ту же структуру, но свойства и способ выполнения методов будут различаться от одного объекта к другому.



WWW.RANDOMSHOESHOP.COM

Objeto Zapato

Atributos: precio, marca, modelo

Métodos: mostrar información, aplicar descuento

En la web de esta zapatería va a haber un objeto "zapato", cuyos atributos serán el precio, la marca y el modelo y cuyos métodos serán una función para mostrar estos atributos o información del zapato y una segunda función que aplica un descuento en el atributo "precio".

На веб-сайте этого обувного магазина будет объект "обувь", а его атрибутами будут цена, марка и модель, а методами - функция для отображения этих атрибутов или информации об обуви и вторая функция, применяющая скидку к атрибуту "цена".

Métodos

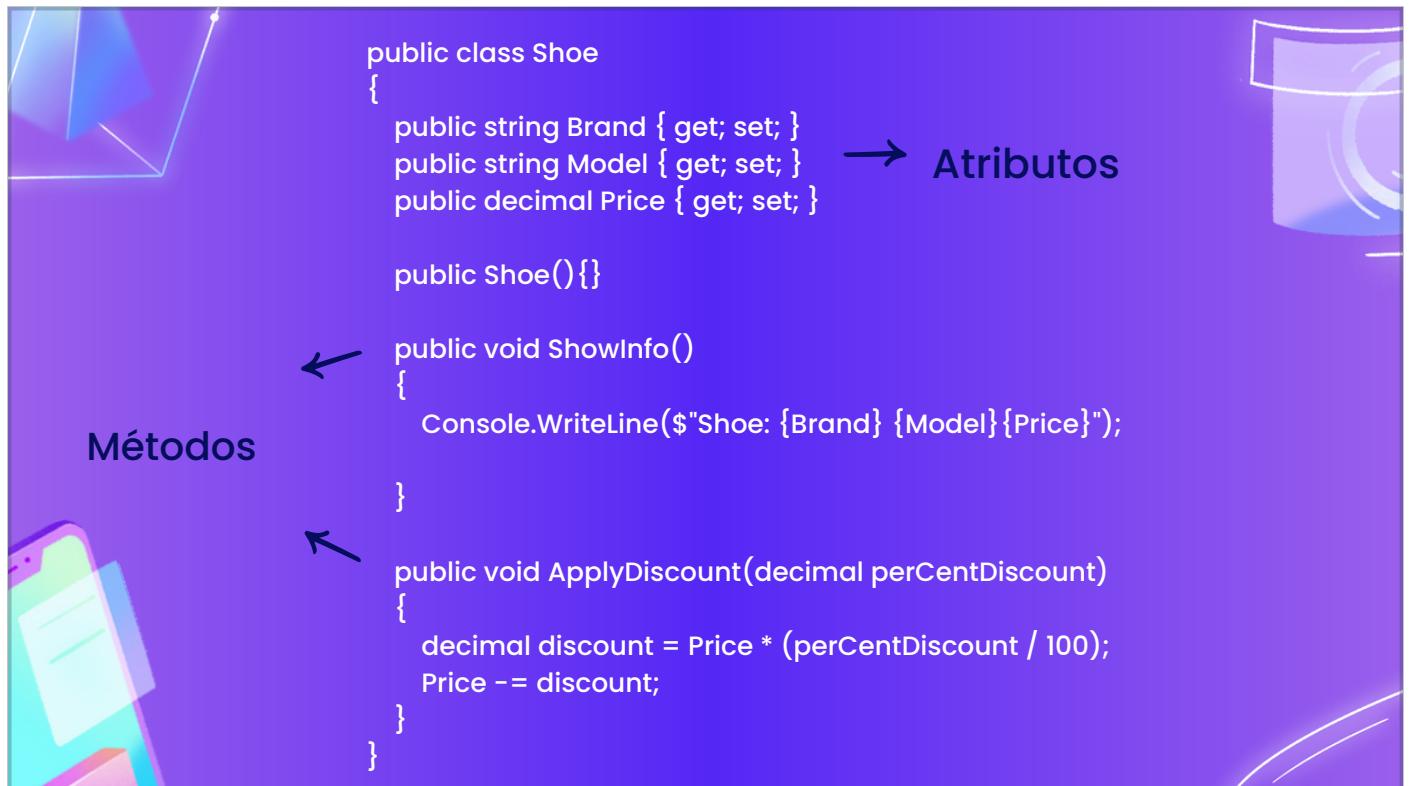
```
public class Shoe
{
    public string Brand { get; set; }
    public string Model { get; set; }
    public decimal Price { get; set; }

    public Shoe() {}

    public void ShowInfo()
    {
        Console.WriteLine($"Shoe: {Brand} {Model}{Price}");
    }

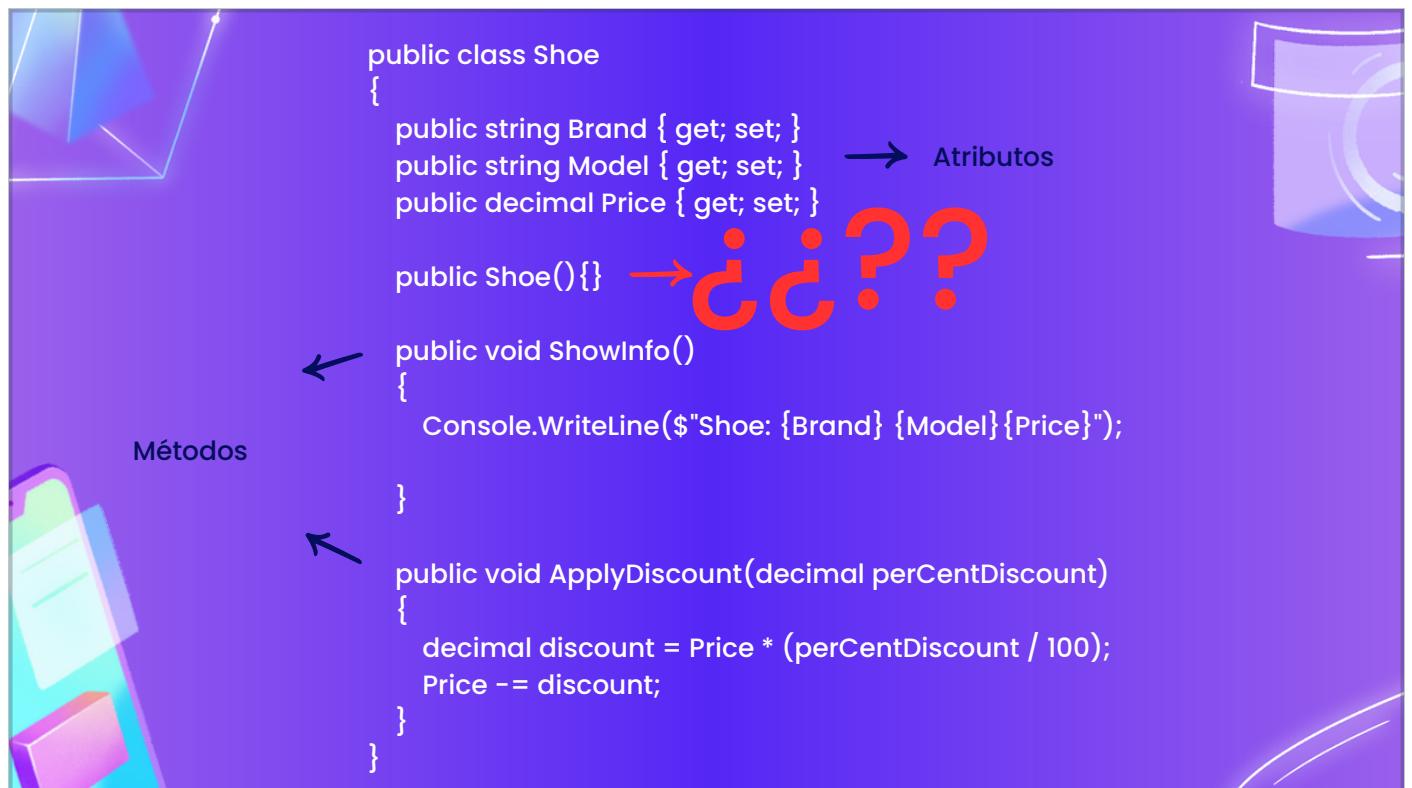
    public void ApplyDiscount(decimal perCentDiscount)
    {
        decimal discount = Price * (perCentDiscount / 100);
        Price -= discount;
    }
}
```

→ Atributos



```
public class Shoe
{
    public string Brand { get; set; }
    public string Model { get; set; }
    public decimal Price { get; set; } → Atributos
    public Shoe(){} → ¿¿¿
    public void ShowInfo()
    {
        Console.WriteLine($"Shoe: {Brand} {Model}{Price}");
    }
    public void ApplyDiscount(decimal perCentDiscount)
    {
        decimal discount = Price * (perCentDiscount / 100);
        Price -= discount;
    }
}
```

Métodos



Что это может быть? Скоро мы узнаем... а если вы не можете ждать, исследуйте это сами!

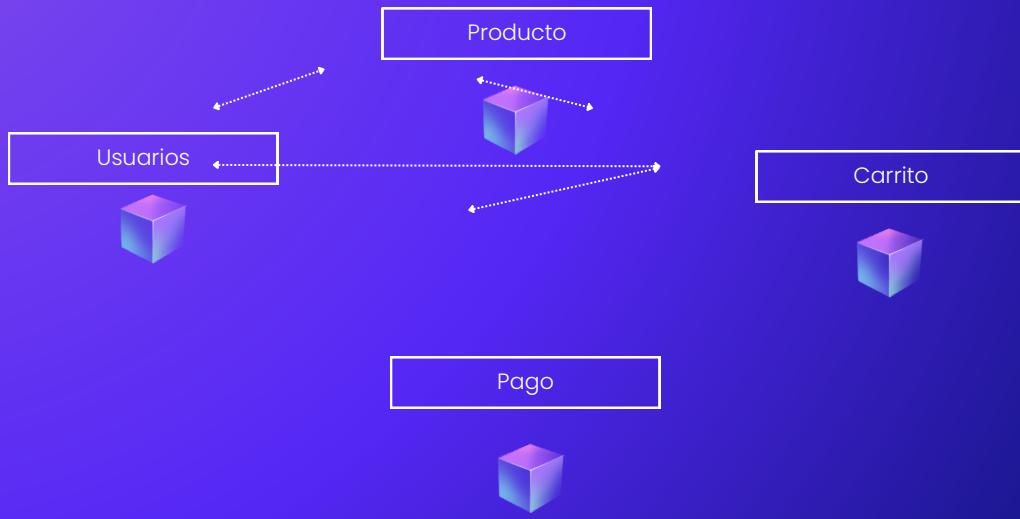
```
Shoe shoe1 = new Shoe("Nike", "Air Max", 150.00m);  
shoe1.showInfo();
```

```
Shoe shoe2 = new Shoe("Adidas", "Superstar", 120.00m);  
shoe2.ApplyDiscount(10);  
shoe2.ShowInfo();
```

Objetos, instancias
de una misma clase

Aquí tenemos dos objetos de una misma instancia, tienen la misma estructura pero las propiedades de los atributos y la forma de desarrollar los métodos son distintos, porque cada objeto corresponde a una entidad distinta del mundo real. En el proceso de abstracción determinamos que estas entidades serán objetos o instancias de una misma clase.

У нас есть два объекта одного и того же экземпляра, они имеют одинаковую структуру, но свойства атрибутов и способы разработки методов отличаются, потому что каждый объект соответствует различной сущности реального мира. В процессе абстракции мы определили, что эти сущности будут объектами или экземплярами одного и того же класса.



Así en lugar de tener en el código una cantidad enorme de funciones que se leen y ejecutan de arriba abajo, tengo una estructura modulada de objetos que interactúan entre ellos y configuran el funcionamiento de la aplicación web. Por ejemplo, un objeto usuario interactúa con un objeto producto (el zapato, por ejemplo) e instancia un objeto carrito al que se añade el producto. El carrito a su vez instancia un objeto "pago" que se asocia al usuario, etc...

Таким образом, вместо иметь в коде огромное количество функций, которые читаются и выполняются сверху вниз, у меня есть модульная структура объектов, которые взаимодействуют друг с другом и настраивают работу веб-приложения. Например, объект пользователя взаимодействует с объектом продукта (например, обувь) и создает экземпляр объекта "корзина", к которому добавляется продукт. Корзина ihrer создает экземпляр объекта "оплата", который связывается с пользователем и т.д.

CONCEPTOS BÁSICOS

Clase

Una clase es una especie de "plantilla" en la que se definen los atributos y métodos predeterminados de un tipo de objeto. Esta plantilla se crea para poder crear objetos fácilmente. Al método de crear nuevos objetos se le conoce como **instanciación**.

Herencia

Por ejemplo, herencia de la clase C a la clase D, es la facilidad mediante la cual la clase D hereda en ella cada uno de los atributos y operaciones de C, como si esos atributos y operaciones hubiesen sido definidos por la misma D. Por lo tanto, puede usar los mismos métodos y variables registrados como "públicos" (public) en C.

Objeto

Instancia de una clase. Entidad provista de un conjunto de propiedades o atributos (datos) y de comportamiento o funcionalidad (métodos), los mismos que consecuentemente reaccionan a eventos. Se corresponden con los objetos reales del mundo que nos rodea, o con objetos internos del sistema (del programa).

Класс

Класс - это своего рода "шаблон", в котором определяются предопределенные атрибуты и методы для определенного типа объекта. Этот шаблон создается для удобного создания объектов. Процесс создания новых объектов известен как **инстанциация**.

Наследование

Например, наследование класса С классу D представляет собой возможность, при которой класс D наследует все атрибуты и операции из С, как если бы эти атрибуты и операции были определены самим D. Таким образом, D может использовать те же самые методы и переменные, определенные как "public" (публичные) в С.

Объект

Это экземпляр класса. Объект представляет собой сущность, обладающую набором свойств или атрибутов (данных) и поведения или функциональности (методов), которые соответствующим образом реагируют на события. Они соответствуют реальным объектам в окружающем нас мире или внутренним объектам системы (программы).

CONCEPTOS BÁSICOS

Método

Algoritmo asociado a un objeto (o a una clase de objetos), cuya ejecución se desencadena tras la recepción de un "mensaje". Desde el punto de vista del comportamiento, es lo que el objeto puede hacer. Un método puede producir un cambio en las propiedades del objeto, o la generación de un "evento" con un nuevo mensaje para otro objeto del sistema.

Evento

Es un suceso en el sistema (tal como una interacción del usuario con la máquina, o un mensaje enviado por un objeto). El sistema maneja el evento enviando el mensaje adecuado al objeto pertinente. También se puede definir como evento la reacción que puede desencadenar un objeto; es decir, la acción que genera.

Atributos

Características que tiene la clase.

Метод

Алгоритм, связанный с объектом (или классом объектов), выполнение которого активизируется после получения "сообщения". С точки зрения поведения, это то, что объект может делать. Метод может изменять свойства объекта или генерировать "событие" с новым сообщением для другого объекта в системе.

Событие

Это событие в системе (такое как взаимодействие пользователя с машиной или сообщение, отправленное объектом). Система обрабатывает событие, отправляя соответствующее сообщение соответствующему объекту. Также можно определить событие как реакцию, которую может вызвать объект, то есть действие, которое он генерирует.

Атрибуты

Характеристики, которые имеет класс.

CONCEPTOS BÁSICOS

Propiedad o atributo

Contenedor de un tipo de dato asociado a un objeto (o a una clase de objetos), que hace los datos visibles desde fuera del objeto y esto se define como sus características predeterminadas, y cuyo valor puede ser alterado por la ejecución de algún método.

Estado interno

Es una variable que se declara privada, que puede ser únicamente accedida y alterada por un método del objeto, y que se utiliza para indicar distintas situaciones posibles para el objeto (o clase de objetos). No es visible al programador que maneja una instancia de la clase.

Identificación de un objeto

Un objeto se representa por medio de una tabla o entidad que esté compuesta por sus atributos y funciones correspondientes.

Свойство или атрибут

Контейнер для типа данных, связанный с объектом (или классом объектов), который делает данные видимыми извне объекта и определяет их предопределенные характеристики, значения которых могут быть изменены выполнением какого-либо метода.

Внутреннее состояние

Это переменная, объявленная как приватная, к которой можно получить доступ и изменить только через метод объекта, и которая используется для указания возможных различных ситуаций для объекта (или класса объектов). Она не видна программисту, который обрабатывает экземпляр класса.

Идентификация объекта

Объект представляется с помощью таблицы или сущности, состоящей из его атрибутов и соответствующих функций.



PILARES/VENTAJAS OOP

ABSTRACCIÓN Puedes representar objetos y entidades del mundo real en tu código, lo que facilita la comprensión y el mantenimiento.	MODULARIDAD Puedes dividir tu aplicación en módulos y clases reutilizables, lo que promueve la reutilización de código y facilita las actualizaciones y modificaciones futuras.
POLIMORFISMO Puedes utilizar el polimorfismo para permitir que diferentes objetos respondan de manera diferente a los mismos métodos, lo que brinda flexibilidad y extensibilidad en tu código.	ENCAPSULACIÓN Puedes ocultar la complejidad interna de tus objetos y exponer solo los métodos y propiedades necesarios, lo que mejora la seguridad y el control del acceso a los datos.
HERENCIAS Puedes crear jerarquías de clases donde las clases derivadas heredan comportamientos y propiedades de las clases base, lo que permite la reutilización de código y la creación de relaciones lógicas entre las entidades.	

Столпы/преимущества ООП

Абстракция: Вы можете представлять объекты и сущности реального мира в своем коде, что облегчает понимание и поддержку.

Модульность: Вы можете разделить свое приложение на модули и повторно используемые классы, что способствует повторному использованию кода и облегчает будущие обновления и изменения.

Инкапсуляция: Вы можете скрывать внутреннюю сложность ваших объектов и выставлять только необходимые методы и свойства, что улучшает безопасность и контроль доступа к данным.

Наследование: Вы можете создавать иерархии классов, где производные классы наследуют поведение и свойства базовых классов, что позволяет повторно использовать код и создавать логические связи между сущностями.

Полиморфизм: Вы можете использовать полиморфизм, чтобы различные объекты реагировали по-разному на одни и те же методы, что обеспечивает гибкость и расширяемость в вашем коде.

BIBLIOGRAFÍA

https://es.wikipedia.org/wiki/Programaci%C3%B3n_orientada_a_objetos

<https://universidadeuropea.com/blog/programacion-orientada-objetos/#:~:text=La%20programaci%C3%B3n%20orientada%20a%20objetos%20es%20un%20modelo%20de%20programaci%C3%B3n,l%C3%A9gica%20necesaria%20para%20esa%20manipulaci%C3%B3n.>

<https://ed.team/blog/que-es-la-programacion-orientada-a-objetos-poo>

[https://intelequia.com/blog/post/qu%C3%A9-es-la-programaci%C3%B3n-orientada-a-objetos#:~:text=La%20Programaci%C3%B3n%20Orientada%20a%20Objetos%20\(POO\)%20es%20un%20paradigma%20de,concepto%20de%20clases%20y%20objetos.](https://intelequia.com/blog/post/qu%C3%A9-es-la-programaci%C3%B3n-orientada-a-objetos#:~:text=La%20Programaci%C3%B3n%20Orientada%20a%20Objetos%20(POO)%20es%20un%20paradigma%20de,concepto%20de%20clases%20y%20objetos.)



MUCHAS GRACIAS

