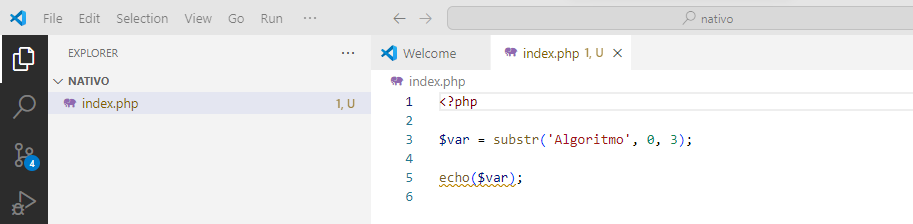
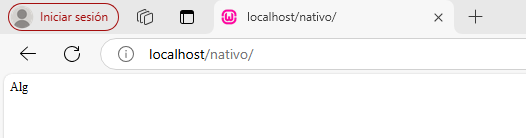
1. código php visual code



2. Mensaje desde el navegador: ejecuta las acciones de las instrucciones del código a escribir. Hacemos es hacer algoritmos personalizados y encapsular los algoritmos en subprogramas que al final del día son funciones y reutilizarlas tantas veces necesitemos y eso básicamente creo yo que permitirá construir un proyecto web. Desarrollo software es contrayendo soluciones del mundo real



3. Consultar en la documentación y editor revisando la sintaxis

4. Patrones de diseño. Mejor forma de realizar las cosas.

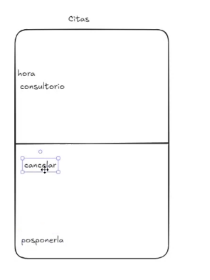
<http://refactoring.guru/es>

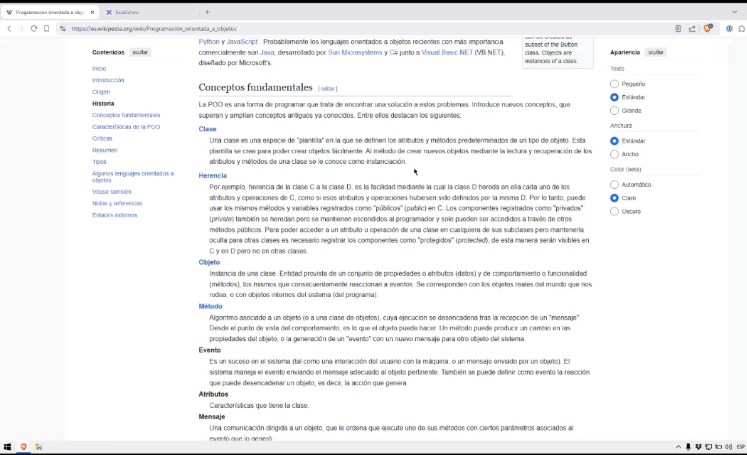
Esto es hacer las cosas de manera bien. Si necesita en su diseño de una cosa que funcione una vez con un patrón de diseño, unas reglas de hacerlo de una manera lo mejor posible para que usted cumpla y le funcione correctamente. Los patrones de diseño son creados de los muchos problemas que se han presentado. Cuando alguien tiene una solución optima la comparte con alguien más. Por ejemplo: que mi usuario no se repita y solo exista una sola vez y no me de error en el momento de seguir trabajando y se puede implementar donde sigue funcionando y nos damos cuenta de que es la mejor manera de hacerlo y decido documentarlo y final del día y exponerlo al mundo la mejor manera si se tiene este problema y que al final del día se vuelve estos patrones de diseño que son reglas o buenas practicas para la construcción del software basándonos en esos problemas que nos vamos a encontrar y todo desarrollador va encontrar y esos son los patrones que estamos viendo a continuación y así mismo como el diseño: modelo, vista y controlador. Es un modelo o diseño o estructura del código que ya ha sido utilizado y nos permite, una arquitectura que nos permite tener claridad y hacerlo de la mejor manera, es un patrón de arquitectura es el modelo mvc, es un patrón de diseño, es un patrón de arquitectura de como organizo los archivos que voy a ejecutar en términos de software. Decía, que había tres grandes ramas: modelo lógico el modelo de la base de datos de como organizo la información o la estructuro dentro de la base de datos, modelo lógico que como la organizo que estructuro cuando la este utilizando o lanzando los algoritmos, como estructura la información y el modelo visual o como lo almaceno es una arquitectura y vamos viendo esos conceptos cuando vayamos construyendo la aplicación.

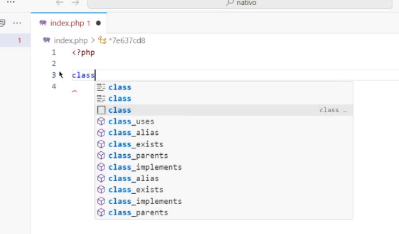
7 noviembre 2024

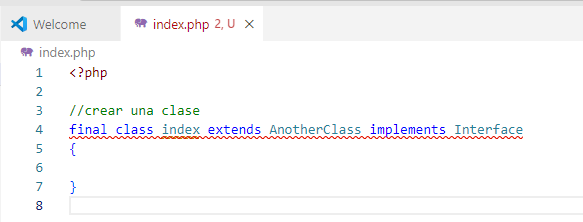
POO: https://excalidraw.com

Entonces vamos a crear un objeto. Esta clase “citas”. Es un objeto cuando se está ejecutando el programa. Yo escribo archivo php y defino los productos. La clase sigue siendo una plantilla, cuando el código se ejecuta es realmente la clase crea un hijo y crea una instancia a un hijo y tiempo de ejecución nosotros vemos como el hijo esta representado o como el sujeto esta representado.









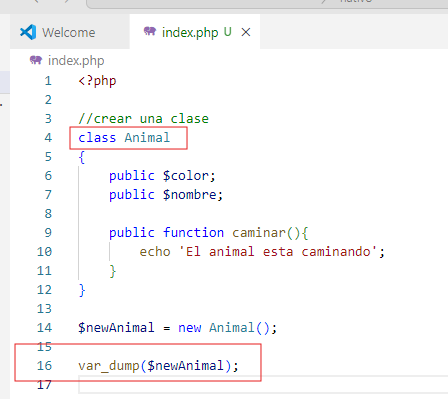
Dentro de la clase animal tenemos una propiedad publica que puede ser color, nombre. Y finalmente tenemos unos métodos con el código que estamos escribiendo puede ser funciones personalizadas de la clase como tal o de la clase encapsulada. Que yo voy crear una función: caminar. Donde creo un código personalizado y aquí en la acción podemos crear como por ejemplo el animal esta caminando. Entonces tenemos en términos de plantilla el código que identifica la clase animal. Luego voy entrar para ver cuando es un objeto donde tenemos la plantilla.



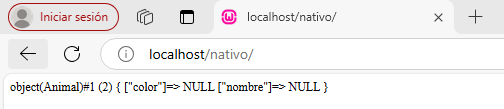
Para nosotros el tiempo de ejecución ver que tenemos un objeto hacemos lo siguiente por ejemplo nuevo animal y creamos una instancia o llamamos una plantilla para crear un objeto:



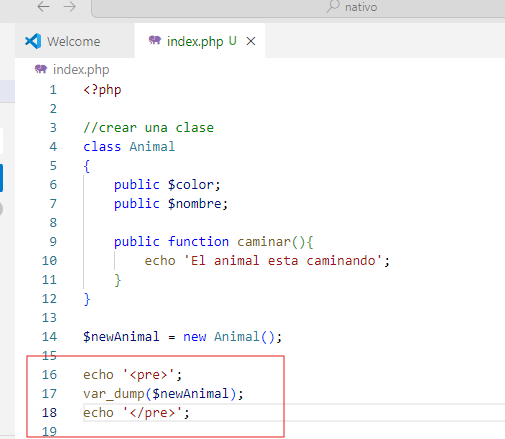
Luego se cambia el nombre de la clase siguiendo buenas prácticas, la primera letra debe ir en mayúscula y ahora en mi código me diga que la variable que es y le doy guardar.



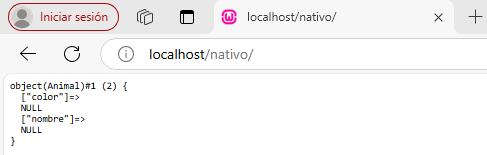
Si esto esta OK. Se ejecuta en el navegador y lo ejecuto:



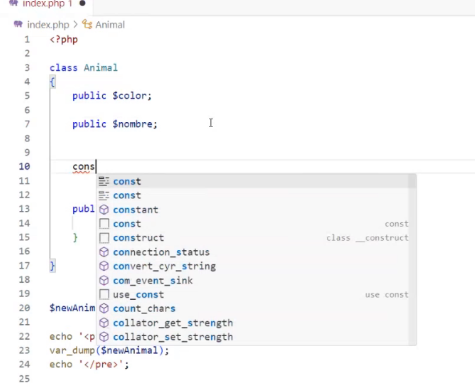
Con una etiquete “HTML” que encapsule código PHP.

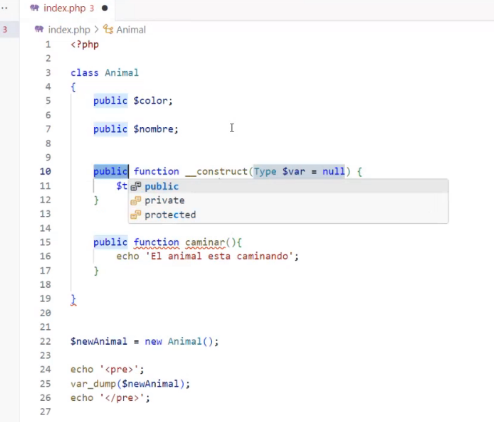


Vuelvo y ejecuto:

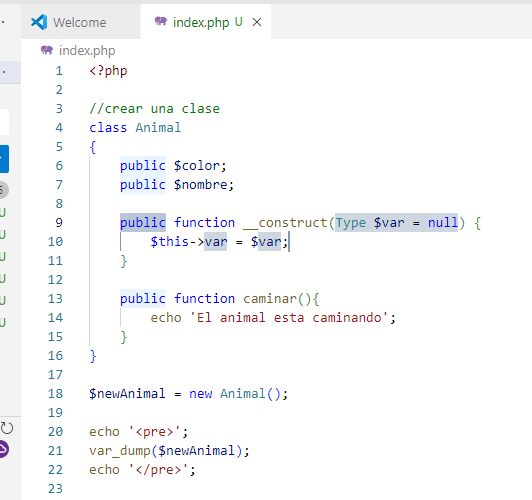


Para definir las propiedades cuando se este creando esa propiedad animal o este llamando esas propiedades deben estar establecidas. Para el caso de PHP existe y vamos a llamar un método por defecto que se llama constructor.

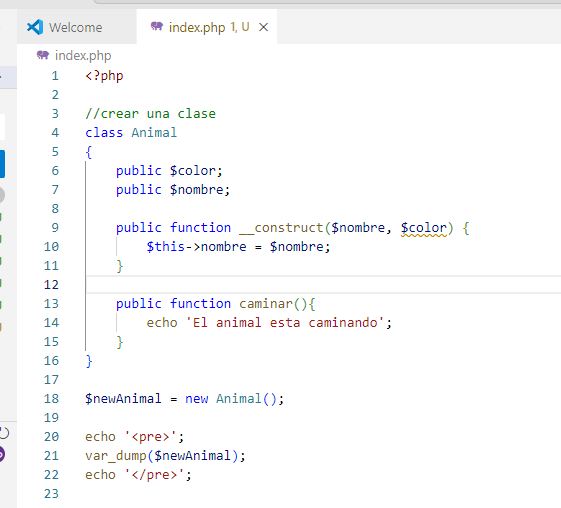




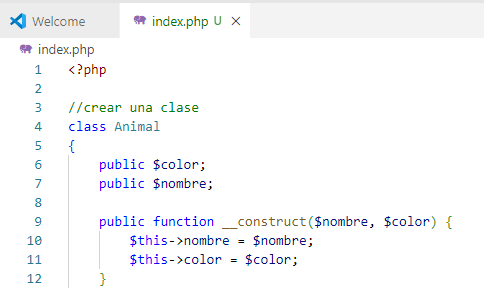
Que es lo que dice las propiedades puede ser definidas cuando se este creando o ejecutan cuando se están creando por acá. La función se crear cuando se ejecute esta línea: 



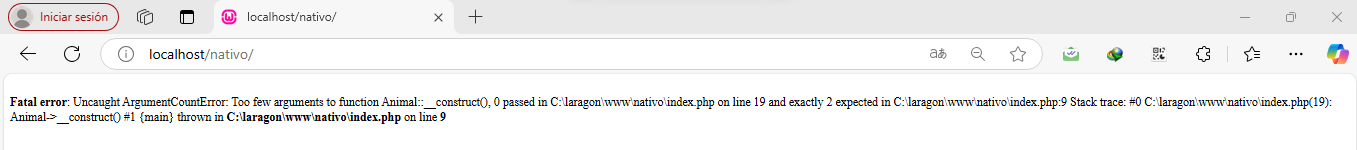
De esa manera yo le puedo decirle alguien define en el constructor el nombre ese nombre y color van ser almacenados en la propiedad anteriormente definidas pasándole los valores a estas propiedades que están definidas.



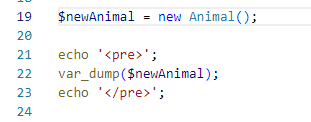
Para color hago lo mismo



Sin hacer ningún cambio adicional yo me devuelvo por acá y le doy refrescar aparece un error porque el constructor no está recibiendo los valores.



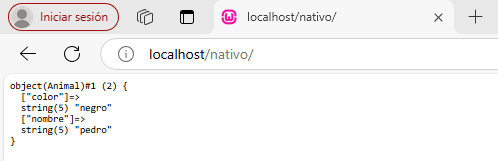
Si acá donde hay un error donde dice se espera dos argumentos nombre y color. No los estamos utilizando.



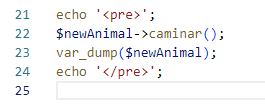


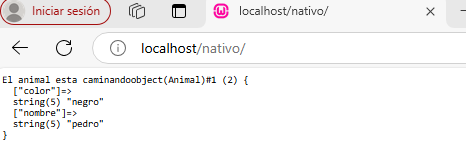
Donde tengo que pasarles esos valores para que sean propiedades ya establecidas. Si creo otro colocar se debe crear otros atributos o podemos agregarle la lógica dentro del código para que esos atributos no se repitan, pero aquí ya vemos como funciona o como esta representado un objeto. Se esta creando un objeto a partir de una plantilla como tal.



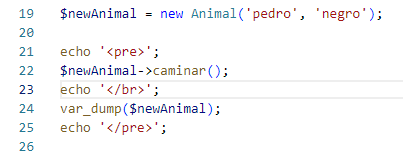


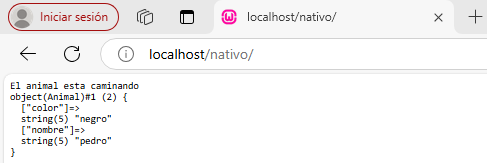
Lo otro que podemos hacer por ejemplo es llamar las acciones como lo hacemos, con un line y flecha podemos acceder a la propiedad caminar que es una función y la función se llama de esta manera que es propia del animal.



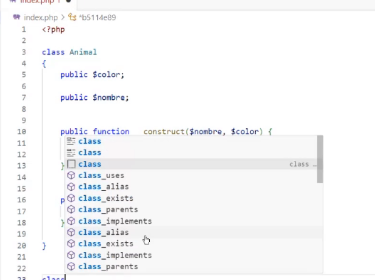


Vamos crear un html sencillo para hacer un salto de línea.

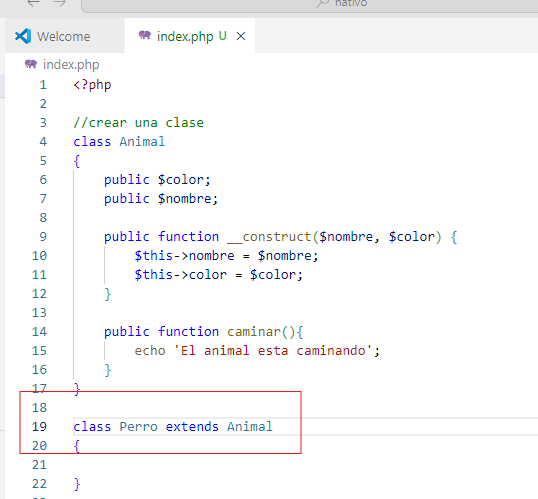




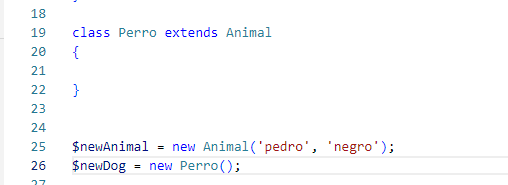
Concepto de la herencia. en el caso POO o lenguaje PHP podemos extender o crear una clase a partir de otra tenemos la herencia como tal. Podemos crear una nueva clase



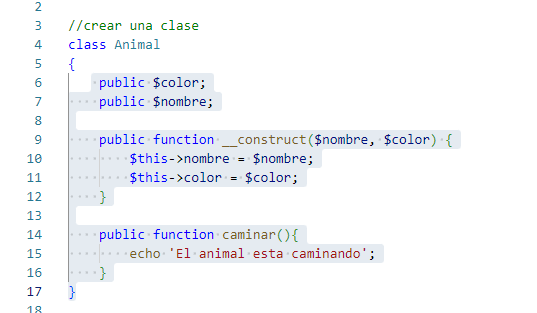
Por ejemplo, la clase perro. Lo que permite compartir los atributos o métodos que viene de la clase padre:



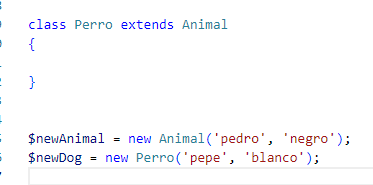
Entonces aca ejecuto este código sin colocar el nombre está esperando



esta lógica que está aquí donde esta heredando todo esto:

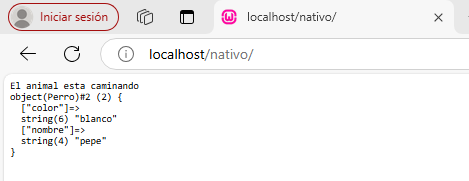


Para crear una instancia de la clase perro debo pasar un nombre y color:

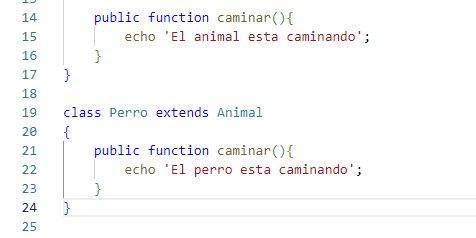


Puedo depurarlo o alinearlo:

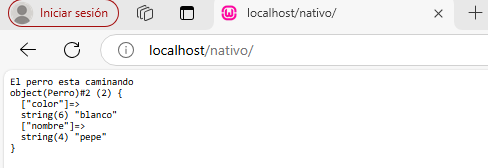




Crear una función para la clase perro:



Se puede modificar para que el comportamiento del método sea personalizado, pero sigue conservando los atributos eso sería la herencia.



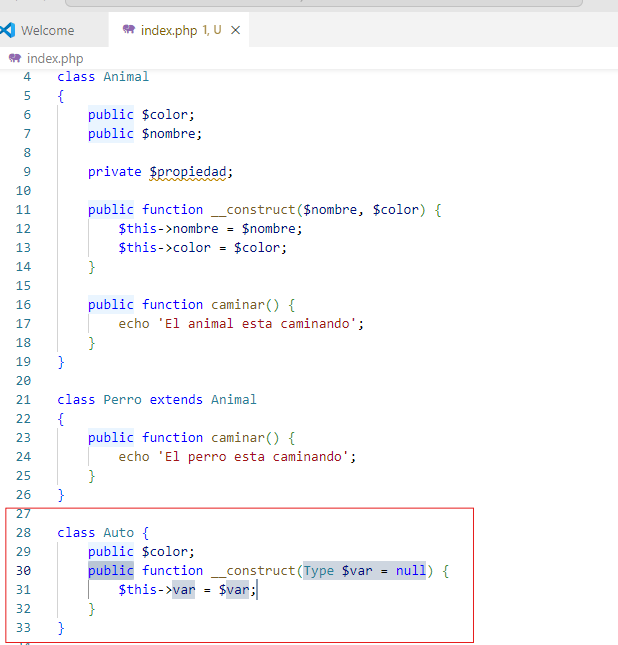
Otras características que forman parte precisamente de la POO en objetos para crear plantillas de las clases o crear platillas del código y no tener que repetir tanto código como tal. Eso finalmente le va permitir utilizar la POO.

Hay otros conceptos como los son Abstracción, encapsulamiento, polimorfismo, modularidad, ocultación, recolección de basura. El que podemos utilizar encapsulamiento. Tiene que ver como se esta viendo las propiedades y las funciones. Por ejemplo: Si aquí definimos una propiedad. Tenemos una propiedad protegida esto le va cambiar el nivel de acceso a esta propiedad cuando estemos en tiempo de ejecución y me refiero que el lenguaje de programación hemos exponer, ocultar o cancelar los datos que estamos utilizando dentro de las clases. Otro tipo de datos que puede ser publico o protegido es privado que nos va ha limitar cuando estemos utilizando el código para las acciones que podemos alterar para esta propiedad. Es este caso la propiedad no puede ser modificada por otra clase a menos que sea la misma clase animal. Eso lo vamos ha ver en tiempo de ejecución cuando haiga mas sentido y se encapsulen ciertas cosas en el caso de laravel para que visualmente lo pueda observar en el caso del usuario la contraseña es un método o propiedad privada que nadie la pueda acceder y uno como usuario puede modificar su contraseña nadie fuera de su contesto de su posibilidad lo puede modificar.

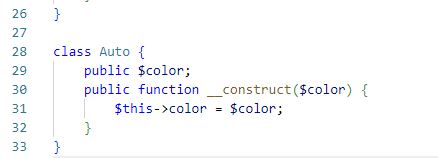
Lo mismo lo van ha ver con el polimorfismo, con el encapsulamiento y demás conceptos. En grandes rasgos básicamente lo que tenemos presente en este momento directamente de la POO.

Polimorfismo. Estructura o serie de datos. Estructura de información para ser exacto y esos pueden ser los this. También podemos utilizar otra clase que sea por ejemplo de una clase que no tenga que ver con un animal la clase autos. Tiene una propiedad publica que la pueda leer todo el mundo que tiene que ver con encapsulamiento que se llama color como lo tiene animal y aquí tenemos el constructor.

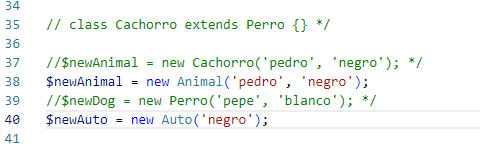




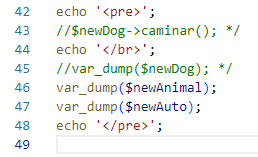
Vamos a pasar el color exactamente lo mismo. Donde podemos ver que tenemos dos clases que tienen la misma propiedad que los hago funcionar básicamente es lo mismo. Adicionalmente les voy agregar un método

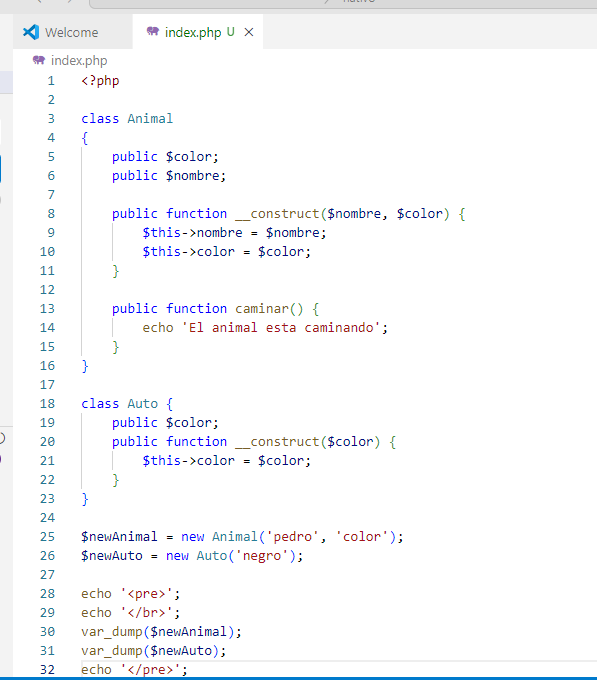


Luego vamos a crear nuevo auto

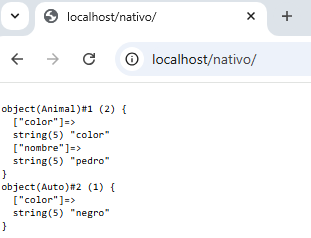


Luego

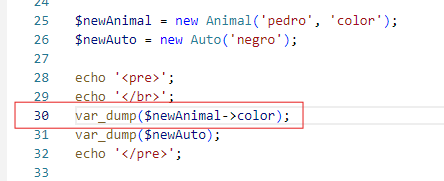


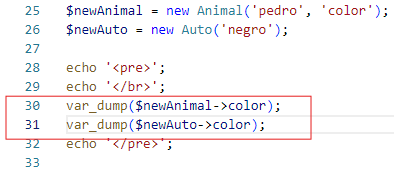


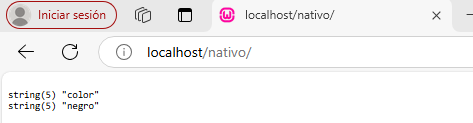
Revisa el resultado tenemos dos objetos por aparte con diferentes clases que una propiedad se comparte, eso lo que se llama polimorfismo y esa propiedad es color:



Yo solamente puedo acceder a la propiedad entonces vamos hacer con el carácter de asignación que se llama -> de esta manera.

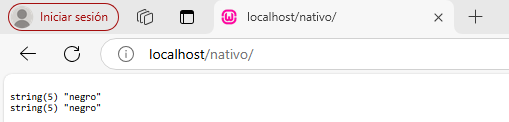




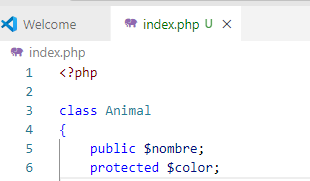


El color es negro para los dos.

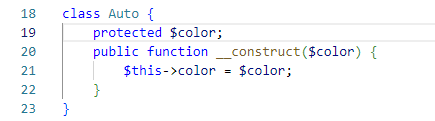




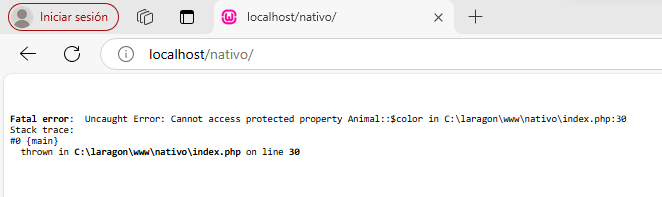
Vamos usar el concepto de encapsulamiento y lo vamos ha decir que el color en este caso que nombre es público. Vamos ha definir nombre pero el color en este caso va ser protegido.



En el segundo caso también.



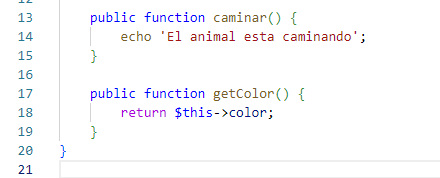
Esto es un concepto de encapsulamiento no puede nadie por fuera de la clase acceder a esa propiedad y quiere decir que tengo que hacer algo como lo siguiente. Como clase animal como esto es protegido nadie puede por afuera del código animal de acceder a esa característica.



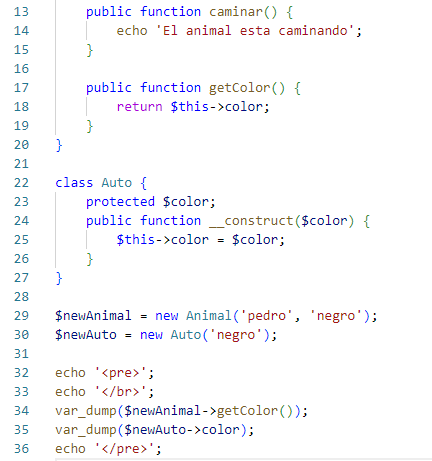
Entonces vamos a crear una función o un método como tal. En este caso se va a llamar “getColor”.

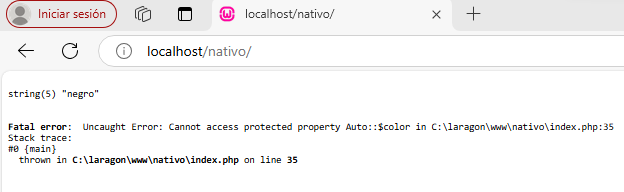


Objeto o clase animal tiene que llamar a la función color para poder leer.

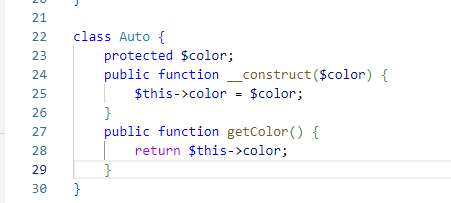


Me voy animal coloco:

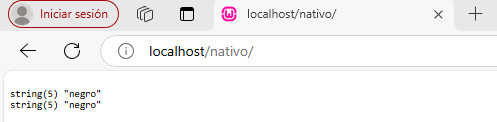




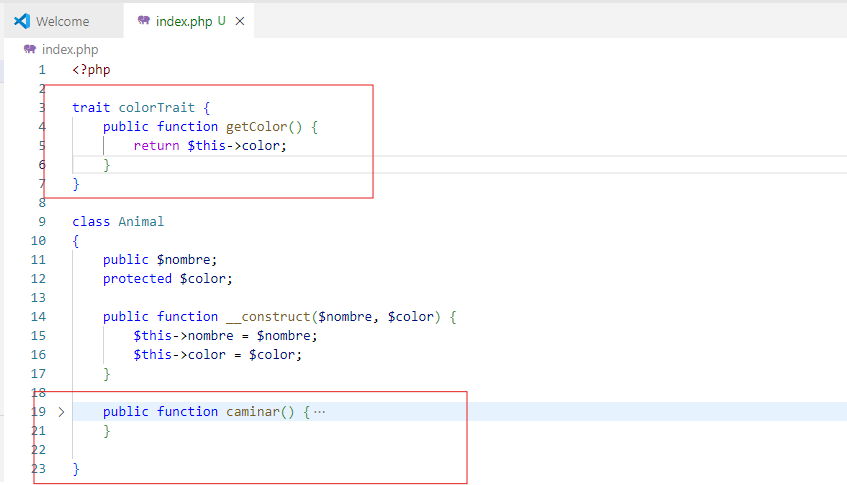
Ya salió la información. Estoy validando en este caso pueda acceder desde el código y leer internamente y esto obedece a otra lógica que nos falta validar.

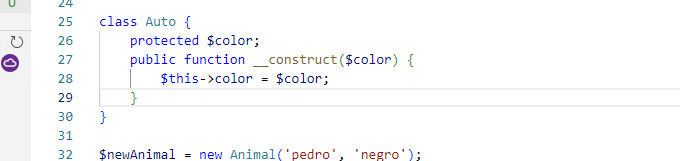






Cual es el problema en acceder a las propiedades personalizadas las dos son de color negro una es de animal y otras es de auto. Como podemos aplicar otro concepto adicional polimorfismo para darle este mismo comportamiento para las dos clases que son diferentes. Entonces yo puedo crear un nuevo tipo de estructura de información dentro de php que se llama trait. Vamos llamar ColorTrait y voy decir que si alguien me utiliza como Trait tenga esta función. Copio el código y borro el antes estaba. Si este Trait si alguien lo tiene que lo ejecute. Le quite el código a animal y auto para que pueda llamar a color.

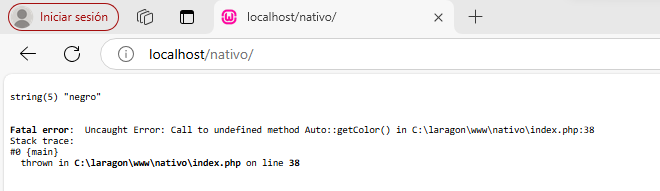




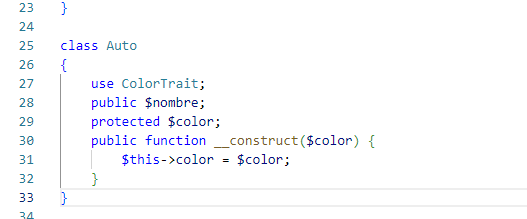
Para que no de error le agregar a la clase animal y le voy decir que utilice esta estructura de datos.

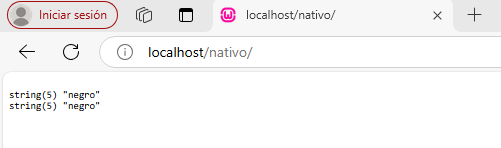


Ya puedo utilizarlo utilizando el concepto de polimorfismo puedo tener el comportamiento, la acción o método getColor se lo agregó a la clase animal sin importar que sea la clase animal.



Y exactamente puedo hacer lo mismo con Auto. Y es más eficiente que repetir código. Substraje el código del color de una propiedad personalizada se lo definí para que este representado una sola vez en el archivo y es utilizado en las dos clases que estamos utilizando. Eso aplica para el concepto de polimorfismo. Porque le estoy dando el comportamiento a dos clases diferentes.

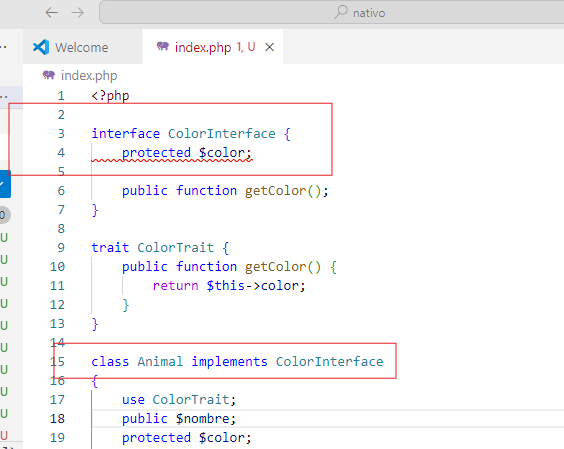




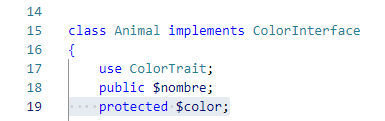
Hay otro tipo de elemento que obedece en hacer las clases en POO en php el elemento mas importante de la unidad de manipulación de la información de las clases y entonces toda gira en darle los comportamientos o características a las clases como tal. Existe otro tipo de archivo u otro tipo de estructura de datos que podemos encontrar en PHP que se llama las interfaces y en este caso vamos a crear una interfaz. Cuando nosotros definimos ese tipo de clase o ese tipo de elemento o ese tipo de estructura básicamente lo que hacemos que el código se asegure y cumple la estructura que necesito sin importar el comportamiento solo es la estructura. Entonces para eso vamos hacer lo siguiente en crear una propiedad protegida que se llame color y vamos a crear una función get color. Que esa interfaz sea utilizada en esta clase de animal.

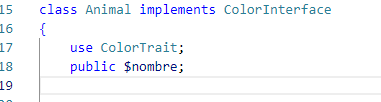


Y esto va se lo siguiente. Estamos diciendo que implemente la clase animal la interfaz color.

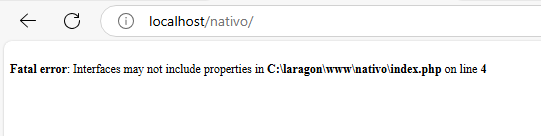


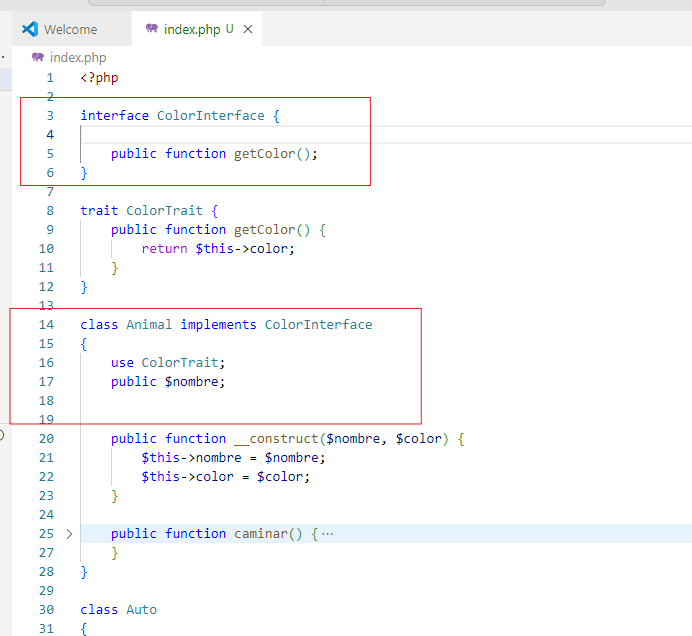
Que sucede si quito lo que tenemos por aca:

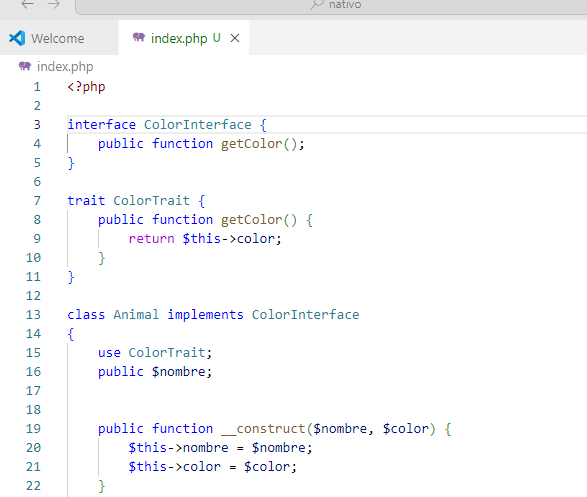


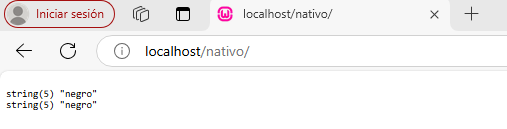


Ya funcionaria porque no cumple de la estructura que había definido

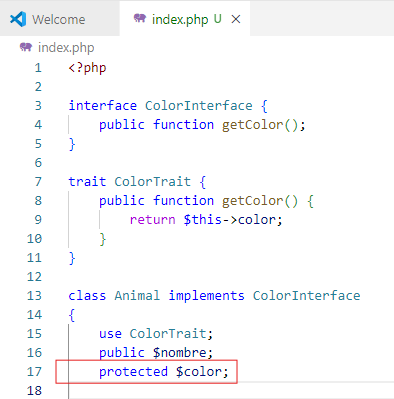


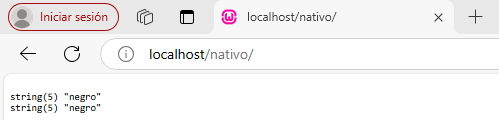




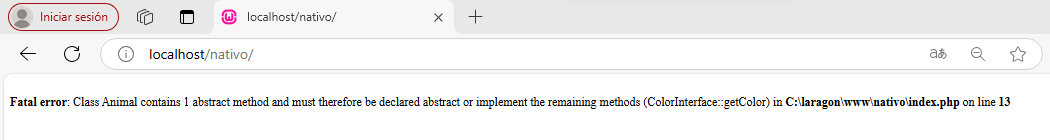


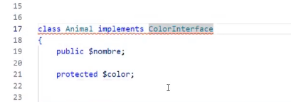
Vamos a retornar el atributo



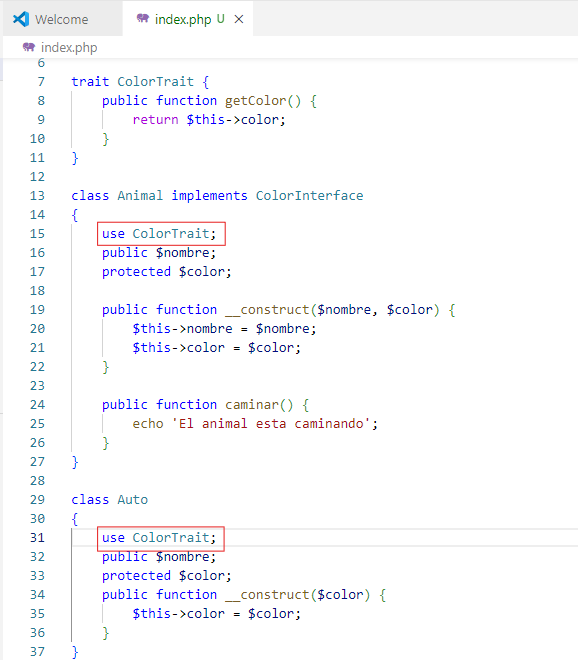


Quitamos     use ColorTrait; da este error porque no cumple con la estructura que esta definida.

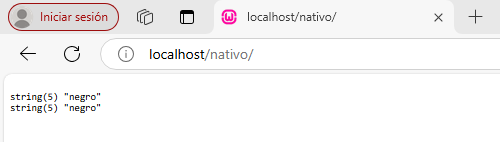




Clase animal a través de la interfaz tiene que tener un método que se llame color sin importar en el código que este en el método. Simplemente tienes que existir un método que se llame color. Hay podemos hacer dos acciones. Agregar el método de manera directa o utilizar el Trait para agregarle el polimorfismo eso solo funciona en el error que tenemos en ese momento. Fijese que eso aplica en ese caso una característica más al código:



De asegurarse de que esta la estructura y que cumpla con todos los conceptos utilizados de la mejor manera posible y esta es la estructura que tiene que tener la clase sin importar el código interno que tiene cada método y este caso el trait es simplemente es compartir esos métodos sin importar si es una clase o no o tipo de clase que sea a través de todas las clases que existe. Al quitarlo daba error y agregarlo esto funciona.



Eso le va a ver más sentido cuando vamos a usar modelos más complejos, clase más complejas, pero mas o menos tenga una referencia de como se aplican los conceptos o encargado de utilizar esos conceptos y entonces aquí están claros la herencia, esta claro el encapsulamiento, esta claro el polimorfismo, está claro la abstracción del código como tal.



Hay tenemos los objetos que se estuvieron ejecutando.

Repositorios