Mini Book: Entiende POO ya!

YOUDEVS

Entiende

Ya!

By: YouDevs

CREADOR DE NINGÚN OTRO LIBRO

El mini-book - v3.0.0-beta

Autor:

YOUDEVS

Sobre el autor:

Hola, mi nombre es Carlos, actualmente soy desarrollador web y creador de contenido para mis distintas redes sociales con mi marca personal llamada: YouDevs y así podrás encontrarme en YouTube, Twitch Instagram, Tiktok y Facebook

Y no puede faltar la página web: https://youdevs.com/

Me gradué de la Ingeniería en computación a mediados del 2015 y conseguí mi primer trabajo como desarrollador web casi inmediatamente (algo que para mí era impensable). Desde entonces he trabajado en varias empresas y también he ejercido como desarrollador independiente (freelancer) creando más de 30 proyectos para diversos clientes.

Mi principal pasatiempo es aprender y para mí ***enseñar en la mejor manera de aprender*** y por eso comencé en redes sociales a subir cursos y "vlogs" narrando mis experiencias y me encanta que mi contenido sea entretenido pero más que nada que sea útil y facilite el aprendizaje a los más novatos.

Es la primera vez que realizo un recurso de este estilo y pretendo hacer más **mini-books** con la intención de brindar contenido de valor, entretenido y fácil de seguir en una pocas páginas.

Soy consciente de mis limitaciones literarias y gramáticas así que con el debido respeto, solo espero que a la mayoría que lo lea le guste, le entienda y aprenda.

Créditos:

- Muchas gracias a ti que estás leyendo mi primer mini libro 💗
- Muchas gracias mí por animarme a hacer este tipo de contenido 💗
- Muchas gracias a mi comunidad por siempre estar apoyando con sus vistas, likes, comentarios, acompañándome en los streams y compartiendo mi contenido 💗
- Un crédito muy especial a esas personas que aportan $1 dólar mes a mes por medio de Patreon. Su apoyo aunque no lo parezca es ampliamente valorado y de hecho este contenido es gracias a la motivación que me brindan.💗💗

Recomendaciones

- Este recurso está pensado principalmente para personas que tienen los fundamentos de la programación y ya han realizados sus primeros programas, así que esa sería la primera recomendación: aprender a programar.
- Más adelante en el mini-book comenzaremos a desarrollar un mini proyecto con el afán de ejemplificar los conceptos que aprenderás, sería lindo que tengas una hoja a la mano para que hagas tu propio diagrama y me lo compartieras vía instagram para publicarlo en mis stories.
- El mini-book es mini, pero no por eso deberías intentar terminarlo en una sola sentada, mejor toma tu tiempo, disfrútalo e interioriza los conceptos, después de todo POO es algo que tienes que aprender una sola vez para poderlo aplicar en cualquier lenguaje de programación que aplique.

Introducción: orientada a objetos

Objeto físico

Un objeto físico y sin entrar en temas metafísicos es una cosa tangible que podemos sentir y manipular. Estamos acostumbrados a interactuar con objetos desde bebés e incluso mientras dormimos; el colchón, la piyama, la almohada y cualquier otro objeto que utilices para tu ritual de sueño.

Como seres humanos estamos limitados (la mayoría) al menos ética y moralmente a no tratar a otras personas o animales como si fueran objetos inertes con los que podemos hacer lo que sea.

Objeto en desarrollo de software

En desarrollo de software la definición de objeto no es tan diferente con las excepciones de que no es un objeto tangible y que cualquier cosa puede ser tratado como: cosa, una persona, un perro, un fantasma, tu suegra/o o tu exnovia/o.

> **Un objeto es una representación informática de un objeto real o ficticio** *(así es, a veces habrá que inventarse objetos: como una máquina del tiempo)*.

Un pequeño ejemplo del mundo real

Un control remoto y una televisión son un perfecto ejemplo de lo que los desarrolladores quieren lograr cuando trabajan en un software utilizando POO:

- El control y la televisión son dos objetos terminados y listos para su uso

- Cada objeto está conformado por otros objetos, la televisión necesitó: de una pantalla, botones y más componentes para finalmente terminar siendo lo que es, e igualmente el control.
- Ambos objetos, aunque diferentes, pueden comunicarse.
- El control manda una señal al televisor por medio de uno de sus botones y la televisión responderá con una acción, ya sea cambiar de canal, subir o bajar el volumen o apagar/encender.

Cuando trabajes con POO la idea es que termines teniendo algo como el control y la televisión: un sistema basado en objetos que se complementen y/o puedan comunicarse entre sí.

Orientado a objetos

La POO es solo una parte de lo que podría ser todo un proceso de desarrollo orientado a objetos. También existe el análisis orientado a objetos (AOO) y el diseño orientado a objetos (DOO).

El análisis, el diseño y el desarrollo forman parte de las etapas del desarrollo:

- comienzas analizando los requerimientos,
- El análisis se pasa a un diseño o diagrama
- Y con base al diagrama comenzamos a escribir código.

Más adelante en este mini-book plantearé un desarrollo muy básico el cual vamos a representar por medio de diagramas. No escribiremos código para representar cada concepto pero visualmente tendrás una guía fiel de cómo se aplica cada uno. Podría decirse que usaremos algo de diseño orientado a objetos (DOO) para ejemplificar el desarrollo.

¿Qué es POO?

para definir POO podemos definir cada sigla con lo que sabes hasta ahora:

- **Programación**: *Escribir código con el objetivo de darle ordenes al ordenador*
- **Orientado a**: *dirigido a*
- **Objetos**: modelo informático de un objeto real/ficticio

Obtenemos que POO es: **Escribir código dirigido a modelar objetos**

Clases: las fabricas de objetos

Como todos los objetos que tenemos en casa provienen de una fabrica o quizás fueron hechos por una persona de manera artesanal. Como un un artista que dibuja y enmarca sus dibujos con la intención de venderlos. El artista seguramente cuenta con un cuarto estudio o un taller para facilitarse el proceso de creación; un lugar donde tiene todo el material para realizar sus cuadros: colores, pinceles, lienzos, un caballete (porta lienzos) y hasta los marcos.

En POO para comenzar a crear objetos también necesitamos algo como una fabrica, algo que nos permita crear muchos objetos diferentes empleando los mismos materiales *(el artista no dibuja siempre el mismo cuadro).*

Esta o estas fabricas de objetos en POO se les conoce como: **clases**

Clase y Objetos

Objeto

Es una colección de datos con comportamientos asociados con el afán de representar en código a un objeto real o ficticio.

Propiedades

A la colección de datos que conforman un objeto se les conoce como: propiedades y sirven para describir cómo lucirán los objetos; por ejemplo: color, tamaño, peso, etc...

Métodos

a los comportamientos asociados a un objeto se les conoce como métodos, cada método es algo que nuestro objeto puede realizar: una función.

Clase

En una clase se definen propiedades y métodos. Es el taller que contiene todas las herramientas necesarias para **crear objetos de un tipo específico** como un cuadro por lo que no sirve para crear un Automóvil. Si quisieras crear un Automóvil tendrías que crear una fábrica especialmente para eso.

Instancia

Crear un objeto es hacer una instancia.

Si por ejemplo tienes definida una `clase Libro` (que contiene lo necesario para crear Libros) y creas un objeto partiendo de esa clase, estás haciendo una **instancia** de la `clase Libro`

Si querer queriendo, ahora tienes un glosario de la terminología que estarás usando cuando desarrollos con POO. Faltan algunos conceptos que todavía no estamos en cuenta y que son los pilares sobre los cuales se sostiene la POO.

Los cuatro pilares de la POO

Como todo en programación cualquier detalle tiene su teoría en POO no es la excepción, más bien todo lo contrario. El propósito de este mini-book es cubrir los detalles básicos para que puedas ir a practicar en código sabiendo lo que tienes qué hacer y en qué consiste lo que estás haciendo.

Antes de comenzar a descifrar cada pilar te plantearé un problema de desarrollo simulando que alguien te acaba de contratar para desarrollarle un pequeño sistema.

La dueña de una librería te acaba de contratar

Doña Gertrudis quiere integrar su negocio a la era digital *(ya va un poco tarde la doña)*

Ella sabe que tu estás aprendiendo desarrollo de software y ha decidido darte una oportunidad para desarrollar su pequeño sistema de librería.

Acordaste una junta con doña Gertrudis para hablar sobre los requerimientos del sistema.

En la junta te das cuenta que Gertrudis la tiene "re-clara", tanto así que te ha llevado una hoja con los:

Requerimientos del sistema de librería

Desarrollar un sistema en el que pueda registrar libros y consultar su información.

Características de los libros:

- autor, título, precio, stock y Id

Funciones:

- Mostrar información

Posdata: Ningún usuario puede modificar los libros, eso lo dejamos para una etapa posterior del sistema

Atentamente y con amor: doña Gertrudis

Lees los requerimientos y les das otra pasada para comenzar con el:

Análisis del sistema de librería

El objeto principal serán libros. Por lo tanto se requiere

- Modelar un libro con las propiedades mencionadas en los requerimientos
- No se pueden modificar las propiedades
- Los objetos *libro* deberán ser capaces de mostrar su información

Procedimiento: Definir una clase llamada Libro con las propiedades (autor, título, precio, stock y Id) y el método necesario para mostrar información.

Como desarrollador generalmente pasarás por las siguientes etapas

- [x] ~~Requerimientos~~
- [x] ~~Análisis~~
- [ ] Diseño / diagrama del sistema
- [ ] Desarrollo

En un equipo de desarrollo suele haber alguien encargado de recolectar los requerimientos, de hacer el análisis e incluso de diagramar el sistema dejando al desarrollador simplemente la etapa de desarrollo. En este mini-libro hemos cubierto la etapa de requerimientos, de análisis y cubriremos también la de diagramación del sistema, sin embargo la etapa de desarrollo es la que deberás aprender por medio de practica y para eso encontrarás valiosos recursos en mi canal de YouTube: YouDevs que incluso son complementos de este mini-libro.

A continuación aprenderás cada uno de los pilares de la POO a la par que diagramamos el sistema de librería que quiere doña Gertrudis.

Abstracción: el pilar donde comienza la POO.

Abstraer es: **eliminar los detalles innecesarios de un objeto para solo enfocarnos en los aspectos que son relevantes para software en desarrollo**.

El sistema en cuestión se basa en objetos reales como lo son los libros (cualquier libro), la clienta fue muy específica con los datos que desea registrar de cada libro.

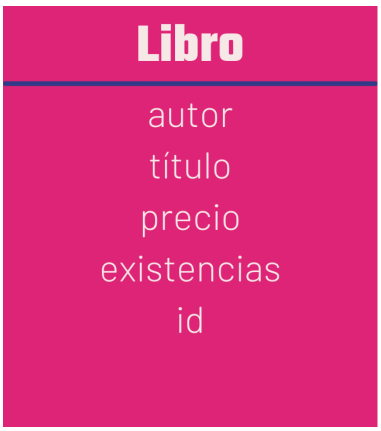
Sin querer queriendo la clienta realizó un proceso de abstracción: eliminó los detalles innecesarios de un libro real para enfocarse solo en los elementos relevantes para el sistema.

Podríamos imaginar cuáles fueron estos detalles innecesarios: la portada, el número de hojas, la editorial o el año de publicación. Todo lo anterior lo ignoró y se enfocó en lo que desea: el autor, título, precio, stock y Id.

En mi opinión el proceso de abstraer, es algo que se hace de forma natural y a veces hasta inconscientemente.

Cuando te preguntas ¿Cómo voy a comenzar con este programa? o imaginas las variables que tienes que definir para dar inicio al desarrollo... en realidad estás abstrayendo.

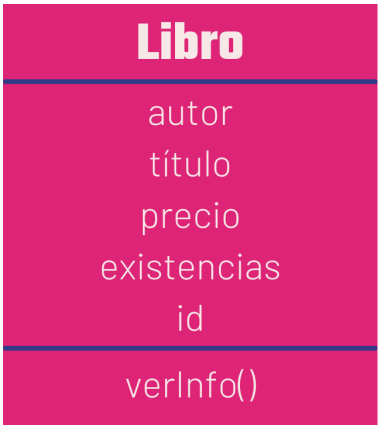
Gracias al análisis que hiciste sabes que necesitas definir una clase que a la postre te permita crear/instanciar libros. Definamos la clase con las características o propiedades requeridas:



- Tenemos una clase llamada: Libro y sus propiedades definidas

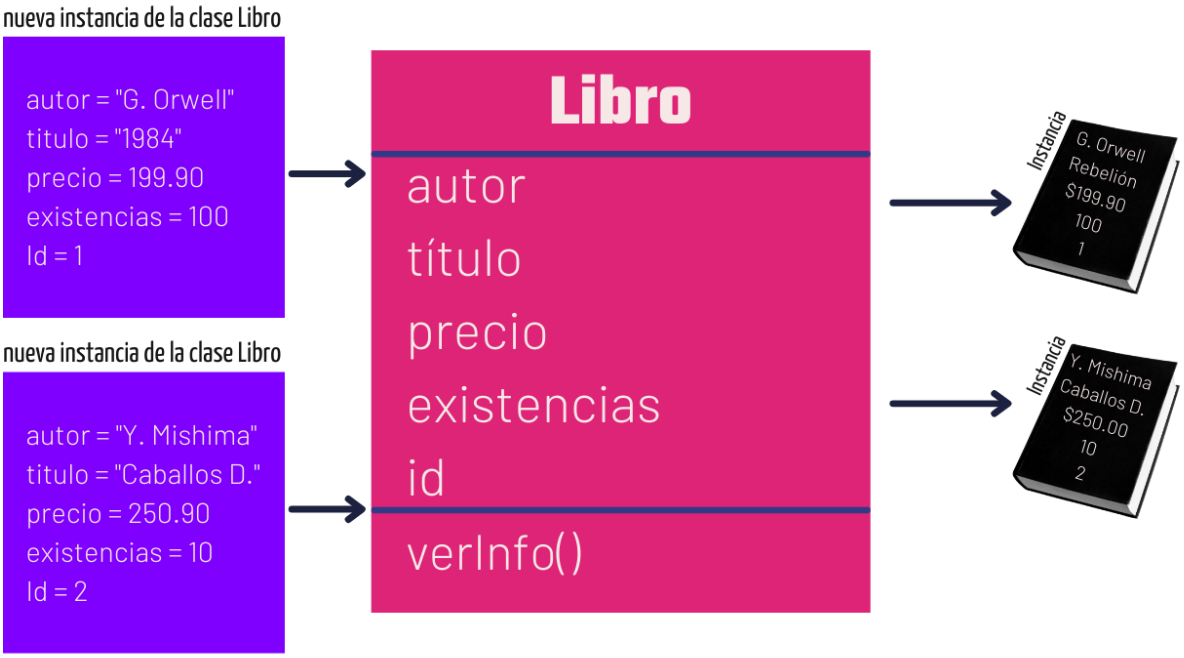
En el análisis se definió que los libros deberían ser capaces de mostrar su información. Esto es una funcionalidad y es el comportamiento que estará asociado a los objetos que se instancien de la `clase Libro`

Recordemos que el comportamiento de los objetos o las funciones que pueden hacer se les llaman Métodos. Agreguemos este método al diagrama:



- El método `verInfo()` se encargará de mostrar toda la información de un libro. Esta información no es más que las propiedades.

Listo! tenemos nuestra clase definida. Ahora podemos **instanciar objetos** de la `clase Libro`.



- Para instanciar un objeto primero tendrás que especificar que quieres instanciar en este caso de la `clase Libro`
- Al momento de instanciar será necesario asignar valores para cada propiedad
- Como puedes apreciar, hemos instanciado dos objetos similares (libros) pero con diferentes propiedades que hacen que cada objeto sea diferente del otro a pesar de pertenecer a la misma clase.

Es así como una clase nos permite emplear las mismas herramientas para crear objetos similares (de la misma clase) que lucen diferente.

Hemos definido el proceso de abstracción e incluso definimos nuestra primer clase e instanciamos dos objetos. Simplemente genial!

Vayamos a aprender y a diagramar el siguiente pilar para avanzar en nuestro desarrollo:

Encapsulamiento: el pilar del ocultamiento

> Encapsular es: ocultar los detalles que no son relevantes para el exterior de la clase. Consta de agrupar propiedades y métodos que no sean necesarios para el exterior; de manera que el acceso sea restringido desde fuera de la clase.

Normalmente tendremos propiedades y métodos que sirven solo para procesos internos de nuestros objetos por lo que no es necesario que el resto del programa (lo que está por fuera de la clase) tenga acceso o conocimiento de estas propiedades y métodos.

Modificadores de acceso

Los lenguajes de programación que aceptan POO suelen contar con palabras reservadas conocidas como modificadores de acceso. Estas palabras reservadas sirven para especificar si queremos que el acceso de una propiedad o método sea público o sea privado y precisamente los dos modificadores de acceso más populares son: `public` y `private`

- Público (`public`): se puede acceder desde cualquier parte del código
- Private (`private`): solo es accesible desde dentro de la clase que lo define.

Gertrudis específico que no desea que nadie pueda modificar la data de sus libros y es normal, como no hay un sistema de usuarios y permisos el dejar abierto al público que

se puedan modificar propiedades podría ocasionar que cualquier usuario cambiara el precio de un libro a $0.00.

En realidad todas las propiedades son datos sensibles, no quisiéramos que cualquier pudiera modificar el título ni el autor, podría ocasionar un caos y el resto igual.

Así que especifiquemos mediante el diagrama que queremos que las propiedades sean privadas:

- Todas las propiedades han sido definidas como privadas: esto limita el acceso desde fuera de la clase y ahora no se pueden acceder o modificar desde fuera de la misma.
-
- El método `verInfo()` ha sido definido como público debido a que solo se limitará a mostrar la información.
    - El método está definido dentro de la clase y por lo tanto tiene acceso a las propiedades privadas
    - Esta es la forma en la que el exterior podrá consultar la información de cada libro sin darle opción a modificar ninguna propiedad.

Realicemos un pequeño pseudocódigo para imaginar cómo sería intentar modificar propiedades privadas y ejecutar el método público:

- Creamos una instancia de la clase Libro, el objeto se llama: libro_1
- la clase recibe valores para cada propiedad

```
libro_1 = nueva instancia Libro(
                "autor" = "Geroge O.",
                "titulo" = "1984",
                "precio" = 199.90,
                "stock" = 100,
                "id" = 1
              );
```

- desde el objeto se intenta modificar una propiedad privada como: precio. El precio que se asignó al objeto cuando se instanció fue de 199.90

```
libro_1.precio = 0.00
```

El resultado sería un error como este: `ERROR, no puede modificar una propiedad privada`

- Ahora intenta ejecutar el método `verInfo()` que es público:

```
libro_1.mostrarInfo()
```

Simulamos que el resultado es algo como:

```
"""
  Autor: G. Orwell
  Título: 1984
  Precio: $199.90
  En stock: 100
  Id: 1
"""
```

Como dato curioso: no todos los lenguajes cuentan con modificadores de acceso formales como r y `private` . Python, por ejemplo, en su lugar emplea una sintaxis especial para replicar este comportamiento.

Encapsular nos permite tener un sistema más seguro y menos agobiante para el usuario, al cual la mayoría de las veces no le interesa cómo funcionan internamente las cosas, solo quiere poder hacer lo que sabe que puede hacer el objeto en cuestión. Así como una persona promedio que sabe manejar un automóvil sin la necesidad de saber cómo trabaja internamente el motor, incluso si el motor tuviera expuesto todo su cableado y componentes internos podría ser peligroso para el usuario curioso y el automóvil mismo, por eso es mejor encapsular los detalles que no son relevantes.

Herencia

> Es el mecanismo por el cual la POO nos permite reutilizar código sin necesidad de hacer copy-paste. Nos permite relacionar clases de tal forma que una clase hereda propiedades y métodos de otra clase.

Así de simple como suena, una clase hereda o extiende su funcionalidad desde otra clase. La clase que hereda u obtiene desde otra clase se le conoce como **subclase** o clase hija y la clase que proporciona su código es conocida como **superclase** o clase padre.

La herencia permite crear objetos más específicos partiendo de una clase general.

Gertrudis necesita agregar un objeto a su sistema

Haz recibido una llamada de doña Gertrudis para solicitarte una nueva feature para el sistema.

Su negocio está creciendo y ahora además de vender libros también venderá Comics.

Gertrudis no deja nada al azar y ha puesto manos a la obra de una nueva hoja de requerimientos:

Requerimientos del sistema de librería: Integrar registro de Comics.

> Integrar al sistema el registro de Comics, los cuales al igual que los libros se pueden consultar pero no modificar.
>
> **Características de comics**:
>
> - las mismas que en los libros
> - ilustradores
> - volumen
>
> **Funciones**:
>
> - Mostrar información
>
> Atentamente y con amor: doña Gertrudis

Después de leer y analizar los nuevos requerimientos te haz percatado que las propiedades entre libros y comics son casi las mismas salvo que los comics cuentan con dos más: ilustradores y volumen, además la funcionalidad también es la misma: `verInfo()`

Podrías pensar que la solución es tan fácil como:

- definir una nueva clase llamada: Comic
- copiar y pegar propiedades y métodos de la `clase Libro` en la `clase Comic`
- y solo agregar las dos nuevas propiedades que hacen falta para un comic.

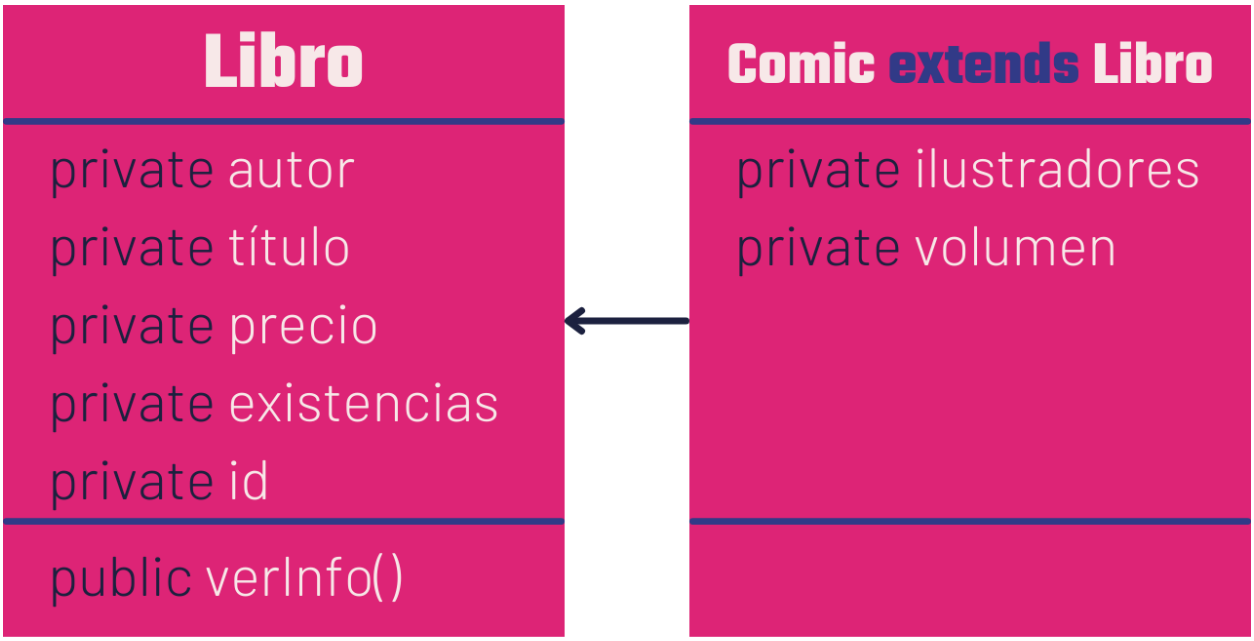
Y si, efectivamente tu sistema funcionaría bien si lo haces así, pero copiar y pegar código rompe uno de los principios de la programación: No te repitas *(DRY: don't repeat yourself)*

Si tu forma de desarrollo se basa en copiar y pegar código de diferentes partes y tener mucho código repetido, poco a poco tu sistema se volverá muy difícil de darle

mantenimiento y de hacerlo crecer hacía otras funcionalidades, porqué probablemente siempre tengas que ir a copiar la nueva funcionalidad de un lado a otro y eso se volvería insostenible en un sistema mediano-grande.

En cambio la herencia, nos permite reutilizar código sin la necesidad de hacer copy-paste.

Veamos cómo quedaría nuestro diagrama si creamos una `clase Comic` que herede de la `clase Libro`:

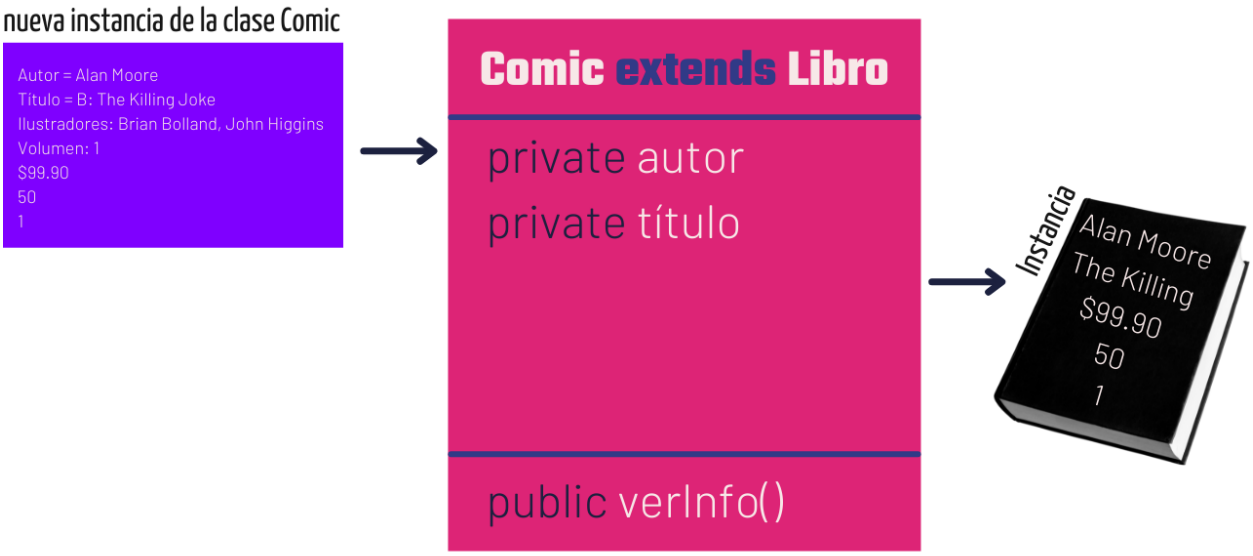


Cuando hablamos de herencia en POO hablamos de **extender la funcionalidad de una clase partiendo de otra**, por eso la palabra más común para relacionar clases en los lenguajes de programación será la palabra **extends.**

- Ahora la `clase Comic` extiende de Libro
  - La `clase Comic` hereda propiedades y métodos de la `clase Libro` por lo que ya no necesitamos definirlos de nuevo en la `clase Comic`.
  - Gracias a la herencia ahora solo necesitamos definir las propiedades y métodos que diferencian a un objeto comic de un libro: ilustradores y volumen.

Podemos observar en el diagrama que la `clase Comic` queda más vacía y en código pasará igual dado que automáticamente estamos reutilizando el código de la `clase Libro` al momento de heredar.

Ahora podemos instanciar objetos Comic:



Como dato curioso: hay lenguajes de programación como Java y Python que aceptan herencia múltiple y otros como PHP que solo permiten heredar de una sola clase. El concepto en sí funciona igual.

En conclusión la herencia nos permite reutilizar código y crear objetos más especializados.

Polimorfismo

Polimorfismo es un apodo despampanante para un concepto muy simple probablemente, el más simple de los cuatro pilares. Aunque su descripción también puede traicionarnos sino vemos un ejemplo claro de cómo se aplica.

La definición literal de polimorfismo es: Muchas formas.

Y en POO:

> es la capacidad de que un mismo método obtenga resultados diferentes dependiendo de la subclase desde la cual se ejecute.

Es confuso porqué... ¿Cómo logramos que una misma función/método obtenga diferentes resultados?

Respuesta: sobrescribiendo el método en la subclase y ajustando un poco.

En nuestro desarrollo tenemos un pequeño problema: el método `verInfo()` está hecho para mostrar toda la información relacionada con un Libro y no está tomando en cuenta las dos propiedades que se agregaron en la clase Comic (ilustradores y volumen).

Si por ejemplo, simulamos ejecutar el método desde un objeto Comic ahora mismo resultaría en algo como:

```
"""
  Autor = Alan Moore
  Título = B: The Killing Joke
  $99.90
  50
  1
"""
```

Nos mostraría la información de las propiedades que coinciden entre las dos clases, pero hace falta las dos propiedades especificas de la clase Comic. Esto pasa porqué en nuestra clase comic no hemos definido ningún método `verInfo()` por lo cual

automáticamente está empleando el método que hereda desde la clase Libro. Si queremos que la funcionalidad este completa tal y como la quiere Gertrudis, debemos emplear polimorfismo para implementar el mismo método `verInfo()` desde la clase Comic y ajustar la información que debe de mostrar:

- Definimos el método `verInfo()` en la `clase Comic`

Ahora cuando ejecutemos `verInfo()` desde un objeto Comic podremos mostrar toda la información, por ejemplo:

```
"""
  Autor = Alan Moore
  Título = B: The Killing Joke
  Ilustradores: Brian Bolland, John Higgins
  Volumen: 1
  $99.90
  50
  1
"""
```

- Ahora si contamos con la información de los ilustradores y el volumen

El polimorfismo no es más que sobrescribir un método que proviene de la superclase y adaptarlo para obtener el resultado deseado.

Ahora podríamos crear tantos objetos libros y comics como doña Gertrudis requiera.

Y colorin colorado este desarrollo se a terminado y quedamos atentos a los nuevos requerimientos de doña gertrudis que seguramente a futuro querrá que apliques.

Despedida

Muchas gracias por haber leído hasta aquí te agradezco tu apoyo y espero que este mini-book te haya gustado y más que nada que hayas aprendido lo que se pretendía, si fue así no dudes en compartirlo en tus redes sociales y/o tus compañeros de clase. Estoy seguro que si tu profesor te pregunta que

— ¿Qué es abstracción?

y respondes que

— Es eliminar los detalles innecesarios de un objeto para enfocarse en los aspectos que son relevantes para el contexto.

Tu profesor se quedará flipando en colores.

No olvides seguirme en todas mis redes sociales es la forma más fácil para no perderte de este tipo de contenido:

- YouTube
- Instagram
- Tiktok
- Facebook
- Twitch

Y también visita mi la página web oficial de youdevs donde encontrarás todos los recursos que he creado de manera organizada: https://youdevs.com/

Te deseo alegría y satisfacción en tu carrera como desarrollador.

```
print("Adiós mundo!")
```