**Herencia y polimorfismo**

[**Hector Benitez**](https://platzi.com/%40hectorbenitez)

27 de Diciembre de 2018

Cuando trabajamos con objetos y clases, algunas veces podemos encontrarnos con clases que son muy similares, incluso que comparten algunos métodos o propiedades, pero que no son completamente iguales.

En este punto hablaremos de la Herencia, un concepto que nos permitirá reutilizar todo las partes que son comunes y nos permitirá tener lo que no es común en clases separadas.

La Herencia funciona como una cadena de herencia, es decir podemos tener una clase y generar una “clase hija” a partir de ella, la clase “hija” reutilizara todas las propiedades y métodos de la clase “padre” y además le permitirá implementar esas partes que la hacen diferente.

Por ejemplo, pensemos que estamos construyendo un sistema de comercio electrónico que maneja libros digitales y álbumes musicales, para esto podríamos generar una cadena de herencia como la siguiente:



Product(id, title, price, description)
Book(isbn, publisher, author, pages, profitBonus) extends Product
Album(company, artist, duration, genre) extends Product

En este ejemplo, un libro es diferente a un álbum en algunas cosas, sin embargo existen ciertas propiedades que se comparten a través de la clase padre Product, de este modo ambas clases comparten las propiedades y métodos de Product pero además de eso implementan propiedades y métodos únicos.

Ahora vamos a hablar de un concepto adicional, el cual también es muy importante, el término es **polimorfismo** y significa “muchas formas”.

Vamos a pensar que queremos calcular la ganancia que obtendremos de la venta de ciertos productos, y en este caso los libros y los álbumes manejan diferentes porcentajes de ganancia, si generamos un método getProfit en la clase Product este método podría definir cuánto ganaremos de cada producto. Por ejemplo pensemos que ganamos 10%.

public function getProfit() {
return $price \* 0.1;
}

El agregar este método dentro de Product nos permitirá usarlo en objetos de la clase Product y también en objetos basados en las clases hijas de Product, ahora, vamos a pensar que los libros manejan una fórmula diferente porque maneja un valor de bonus adicional, en este caso podríamos tener el método getProfit pero ahora declarado dentro de la clase Book y utilizando la lógica única de esta clase:

public function getProfit() {
return $price \* (0.1 + $this->profitBonus);
}

Este concepto es un tipo de polimorfismo el cual llamamos Sobreescritura y lo que nos permite es reemplazar algo que ya estaba definido en una clase padre.

Un ejemplo de uso para esta cadena de herencia es, por ejemplo, si tenemos una lista de productos, algunos de ellos son libros y otros álbumes, y si queremos saber las ganancias totales, simplemente tenemos que recorrer los elementos e ir sumando el resultado del método getProfit y en cada caso el objeto sabrá cuál fórmula utilizar porque está definida dentro de su clase.

En resumen la herencia nos permite reutilizar código entre nuestras clases y el polimorfismo, en este ejemplo la sobreescritura, nos ayudará a que las clases puedan reaccionar de una manera diferente a métodos con el mismo nombre.