# Programación Orientada a Objetos

*4 de Marzo de 2018 a las 03:16*

Este artículo es complementario del episodio 22 del podcast Tomando Un Café que trata sobre Programación Orientada a Objetos(POO). El objetivo es dar una introducción y dar a conocer diversos conceptos desde un punto de visto teórico y no enfocado a un lenguaje de programación en particular.

Este artículo esta enfocada a usuario noveles en la programación y que sirva como primer contacto sobre este tema, habrá muchos conceptos que no se verán y si el lector quiere profundizar debera buscar mas contenido.

## ¿Qué es POO?

POO es un paradigma de programación, describe un enfoque a la hora de diseñar una solución, las aplicaciones que siguen este paradigma tienen una estructura definida con los caracteriza, un programa desarrollado en POO se distingue facilmente. Hay mas paradigmas de programación como; funcional, declarativa, dirigida a aspectos, reactiva,imperativa...etc.

Los lenguajes de programación, con los cuales desarrollaremos el código donde implementaremos la solución diseñada, pueden soportar uno o varios paradigma de programación. Por ejemplo, *Python* soporta POO, programación imperativa y funcional.

POO esta compuesta por una serie de características que consideramos como principales, que están defirnirdas por una serie de conceptos , que son los siguientes:

- **Cohesión**: Permite conocer el grado de relación entre los elementos de un módulo, una cohesión alta es lo deseado porque implica un buen diseño.
- **Acoplamiento**: Muy relacionado con el anterior, define las relación entre los diferentes módulos de una aplicación, un acoplamiento bajo implica que hay una relación baja entre los módulos. Esto facilita la modificación o sustitución, cualquier de estas tareas no afecta al resto.
- **Abstracción**: Indica que tenemos que centranos en la funcionalidad, en *¿Qué hace?*, y no centrarnos en el detalle, *¿Como se hace?*.
- **Encapsulación**: Define que la información debe ocultarse(encapsulamiento) no se debe tener acceso directo a los datos de un componente, y el acceso sera restringido.
- **Polimorfismo**: Define que un determinado ccomponente puede funcionar de

diferentes formas dependiendo de como se llame. Muy relacionado con el concepto de Herencia.

- **Herencia**: Define la posibilidad de definir nuevos componentes reutilizando elementos de componentes creado anteriormiente. Mas adelante se tratara este concetpos

Como su nombre indica, en POO el concepto clave es el *objeto*, una aplicación estará compuesta por una serie de objetos que se podrán comunicar entre ellos, que tienen una características y acciones definidas. Una aplicación desarrollada en POO ejecuta un conjunto de objetos.

A continuación se muestra un glosario de terminos relacionadas con la POO, donde se describiran diferentes componentes que podemos utilizar. Algunos de los componentes descritos pueden no estan incluido en un lenguaje, Python no utiliza *Interfaces*.

Los componentes mostrados en el glosario se pueden considerar como los mas importantes dentro POO.

## Glosario de términos

**Objeto**: Representa una entidad dentro del contexto de la problema , que es utilizada por una aplicación para crear un modelo que permita solucionar un problema. Puede representar tanto una entidad real, una casa o coche, como una entidad abstracto, cliente o persona.

**Clase**: Permite definir un objeto, tanto sus caracteristicas como sus acciones, en un lenguaje de programación, define la estructura del objeto. Hay que tener en cuenta que para poder utilizar un objeto, debe ser definido por una clase.

**Atributos**: Definen las características de un objeto, una clase puede contener una conjunto de atributos que permite describir un objeto.

**Metodos**: Definen las acciones que podemos realizaar con ese objeto, son funciones que pertenecen (su ámbito) a una clase.

**Instancia**: Asignación de una serie de valores a los atributos de una clase. Para poder utilizar un objeto se debe crear a partir de su clase y ,para ello, debe asignarse una serie de valores a sus atributos. Podemos crear diversas instancias de una misma clase, incluso con los mismo datos, que representarán objetos diferentes.

**Constructor**: Es un método especial, que define como crear un objeto de la clase. Cuando se crea una instancia el constructor índica que atributos debemos asignarle valores.

**Herencia**: Es una caracteristica principal de POO, define un mecanismo que permite

crear una clase partiendo de otras clases existentes, se denomina *herencia múltiples* si derivan de varias clases, reutilizando los atributos y métodos de otras clases, permitiendo añadir su propios atributos y métodos. Con este mecanismo podemos reutilizar diversas clases para crear otras nuevas, evitando rediseños innecesarios.

**Clases abstractas**: Son un tipo de clase diseñada para la herencia, esta clases es utilizada para que otras clases la utilizen y deriven de ella sus atributos y métodos. Con son especificas para ser utilizadas en la herencias, estas clases no pueden instanciarse, no generan objetos.

**Interfaz**: Es un conjunto de métodos , cuyo objetivo es que sirvan de plantilla para crear otras clases. Define solo los métodos, tipo y parametros, no hay código dentro de ellos, las clases que lo implementan deberán desarrollar su funcionalidad . Una interfaz no puede crear un objetos, por ese mótivo se le denominan *métodos abstractos*.

**Clases anónimas**: Son un tipo de clase que no tiene un nombre definido, se pueden crear objetos con ella(instancia), y heredar, igual que en las clases "normales". Pueden utilizarse en diversos ambitos, depende de como esten implementadas en el lenguaje, crear objetos únicos y sencillos, ser utilizadas como parámetros o gestión de eventos.

## Conclusión

La Programación Orientada a Objetos es el paradigma de programación mas utilizado , con diferencia, muchos lenguajes de programación lo soportan y muchos framework lo utilizan para el desarrollo de aplicaciones. Por lo que cualquier programador debe conocerlo y utilizarlo, facilitara su trabajo en el desarrollo de código.

Como he comentado anteriormente, este artículo sirve de complemento al episodio del podcast sobre POO, aunque no es obligatorio y el artículo por si mismo puede servir como introducción a POO.

Espero que después de leer el artículo os den ganas de profundizar y utilizar este paradigma de programación.