Características POO

1. ¿Qué es POO?
2. Encapsulamiento
3. Abstracción
4. Herencia
5. Polimorfismo
6. Clases y objetos
7. Métodos y objetos
8. Modularidad
9. Reusabilidad

¿Qué es POO?

La Programación Orientada a Objetos (POO) es un modelo fundamental en la ingeniería de software, que establece un enfoque basado en la organización de código en clases y objetos. Este modelo ofrece una estructura sólida para crear software modular y reutilizable.

A lo largo de la historia de la programación, han surgido varios paradigmas. En un extremo, se encuentran los lenguajes secuenciales como COBOL, y los procedimentales como Basic o C, que se centran en la secuencia de instrucciones y la manipulación de datos. En el otro extremo, están los lenguajes modernos como Java, C# y Python, que adoptan paradigmas más orientados a la definición y organización del programa, siendo la POO el modelo más popular y ampliamente utilizado en la actualidad. Link

Encapsulamiento: Es el proceso de almacenar un grupo de elementos que contribuyen a una estructura y su comportamiento, sirve para separar la interfaz contractual de una abstracción y su implantación.

Existen tres niveles de acceso para el encapsulamiento, los cuales son:

Público (Public): Todos pueden acceder a los datos o métodos de una clase que se definen con este nivel, este es el nivel más bajo, esto es lo que tu quieres que la parte externa vea.

Protegido (Protected): Podemos decir que estás no son de acceso público, solamente son accesibles dentro de su clase y por subclases.

Privado (Private): En este nivel se puede declarar miembros accesibles sólo para la propia clase. Link

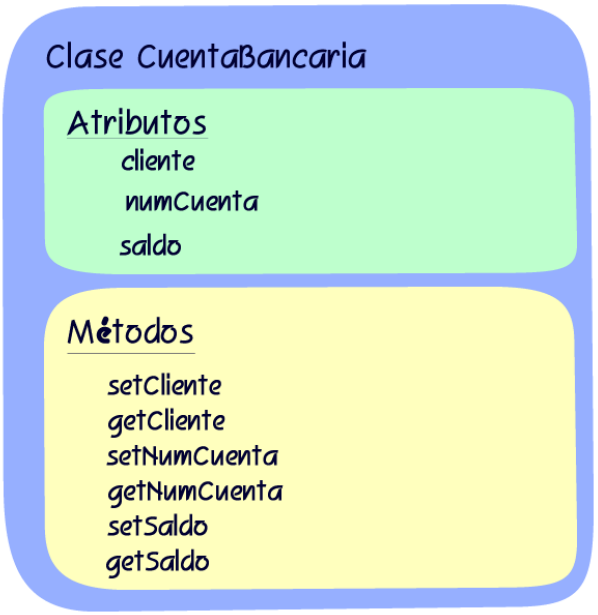
Abstracción:

Una abstracción se enfoca en la visión externa de un objeto, separa el comportamiento específico de un objeto, a esta división que realiza se le conoce como la barrera de abstracción, la cuál se consigue aplicando el principio de mínimo compromiso.

Hay una alta gama de abstracciones que existen desde los objetos que modelan muy cerca de entidades, a objetos que no tienen razón para existir, vamos a hacer una rápida mención de ello.

Tipos:

- Abstracción de Entidades: Es un objeto que representa un modelo útil de una entidad que se desea.
- Abstracción de Acciones: Un objeto que representa un conjunto de operaciones y todas ellas desempeñan funciones del mismo tipo.
- Abstracción de Máquinas virtuales: Un objeto que agrupa operaciones, todas ellas virtuales, utilizadas por algún nivel superior de control u operaciones (entre ellos podríamos hablar de un circuito).
- Abstracción de coincidencia: Un objeto que almacena un conjunto de operaciones que no tienen relación entre sí. Link



Herencia:

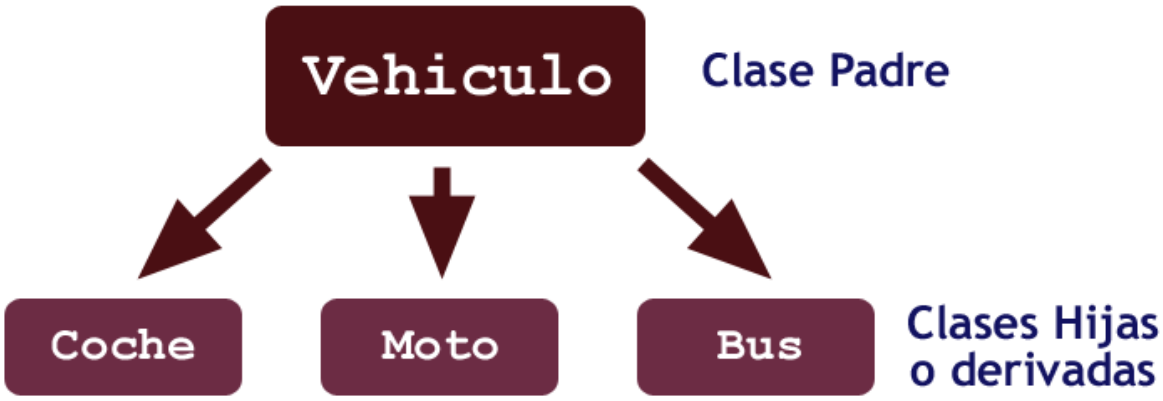
La herencia permite que se puedan definir nuevas clases basadas de unas ya existentes a fin de reutilizar el código, generando así una jerarquía de clases dentro de una aplicación. Si una clase deriva de otra, esta hereda sus atributos y métodos y puede añadir nuevos atributos, métodos o redefinir los heredados.

El concepto de herencia ofrece mucho juego. Gracias a esto, lograremos un código mucho más limpio, estructurado y con menos líneas de código, lo que lo hace más legible. Link

Polimorfismos:

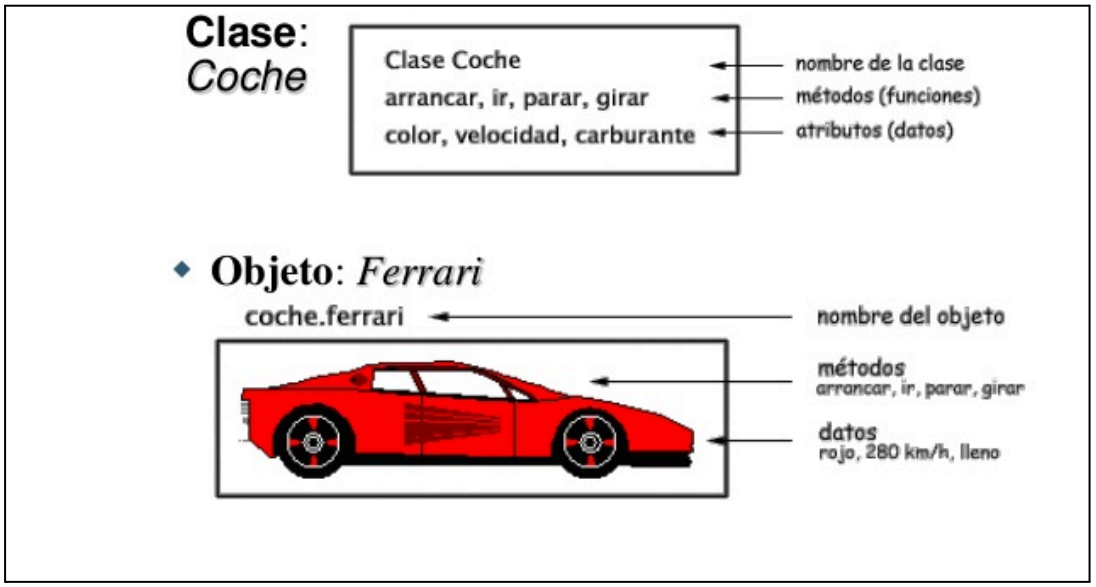
El concepto de polimorfismo es en realidad algo muy básico. Realmente, cuando estamos aprendiendo Programación Orientada a Objetos (también conocida por sus siglas POO / OOP) muchos estudiantes nos hacemos un embolado tremendo al tratar de entender el concepto, pero en su base es algo extremadamente sencillo.

Trataremos de explicarlo en este artículo con palabras sencillas, pero para los valientes, aquí va una primera definición que no es mía y que carece de la prometida sencillez. Pero no te preocupes, pues la entiendas o no, luego lo explicaré todo de manera más llana. Link



Clases y Objetos:

Las clases y los objetos son los pilares fundamentales de la Programación Orientada a Objetos (POO), y su relación es esencial para entender cómo funciona este paradigma de programación. Los objetos se crean a partir de clases, que actúan como moldes o plantillas para definir las características y comportamientos de esos objetos. A los objetos también se les conoce como instancias de una clase.



Clase:

Puedes pensar en una clase como un concepto abstracto, similar a la idea de una persona. Cuando hablamos de una persona, no nos referimos a una persona específica, sino a un conjunto de características y comportamientos que definen lo que significa ser una persona. Estas características podrían incluir el nombre, la edad, la estatura, entre otros aspectos. Esto es lo que representa la clase: una descripción general de un tipo de objeto.

Objeto:

Por otro lado, los objetos son casos concretos o ejemplares de esa clase. Son como individuos específicos que encarnan las características definidas por la clase. Por ejemplo, tú, tu mamá, el presidente de tu país, Alan Turing, Nelson Mandela, son todos objetos de la clase "persona". Cada uno de estos objetos tiene sus propios valores para las características definidas en la clase, como el nombre, la edad y la estatura. Link

Métodos y atributos:

En Java, los atributos pueden clasificarse en varios tipos según sus propiedades y comportamientos.En Java, los atributos de una clase pueden inicializarse de varias formas, ya sea directamente en la declaración de la variable, dentro de un constructor o en un bloque de inicialización.

- Atributos de instancia: Son específicos de cada instancia de una clase. Cada objeto tiene su propia copia de estos atributos y su valor puede variar de un objeto a otro.
- Atributos estáticos: Son compartidos por todas las instancias de una clase. Pertenecen a la clase en lugar de a una instancia particular, y su valor es el mismo para todas las instancias de esa clase.
- Atributos estáticos: Son compartidos por todas las instancias de una clase. Pertenecen a la clase en lugar de a una instancia particular, y su valor es el mismo para todas las instancias de esa clase.
- Atributos de clase: Son similares a los atributos estáticos y son compartidos por todas las instancias de una clase. Sin embargo, se pueden heredar y sobrescribir en subclases. Link

Modularidad:

La modularidad consiste en dividir un programa en módulos que puedan compilarse por separado, sin embargo tendrá conexiones con otros módulos.

La modularidad también tiene principios y son los siguientes:

1. Capacidad de descomponer un sistema complejo.

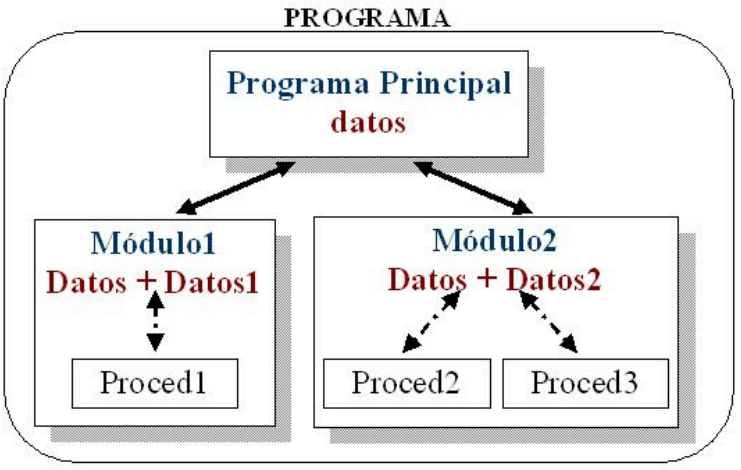
Descompone un sistema en subprogramas (recuerda llamarlos módulos), el problema en general lo divides en problemas más pequeños.

2. Capacidad de componer a través de sus módulos.

Indica la posibilidad de componer el programa desde los problemas más pequeños completando y resolviendo el problema en general, particularmente cuando se crea software se utilizan algunos módulos existentes para poder formar lo que nos solicitan, estos módulos que se integran a la aplicación deben de ser diseñados para ser reusables.

3. Comprensión del sistema en partes.

El poder tener cada parte separada nos ayuda a la comprensión del código y del sistema, también a la modificación del mismo, recordemos que si el sistema necesita modificaciones y no hemos trabajado con módulos definitivamente eso será un caos.



Reusabilidad:

Una vez que una clase ha sido escrita, creada y depurada, se puede distribuir a otros programadores para utilizar en sus propios programas. Esta propiedad se llama reusabilidad o reutilización. Su concepto es similar a las funciones incluidas en las bibliotecas de funciones de un lenguaje procedimental como C que se pueden incorporar en diferentes programas. Link