Programación Orientada a Objetos

¿Qué es la POO?

Es uno de los más conocidos **paradigmas de programación**.

Entendemos por paradigma de programación, a un enfoque particular o filosofía para diseñar soluciones a los problemas del mundo de la programación.

Este enfoque **utiliza objetos como elementos fundamentales en la construcción de la solución**

El paradigma de POO inicia en 1970 y se popularizó en la década de los 90'.
Hoy es uno de los más usados.

Algunos Lenguajes POO: C++, C#, Java, **Python**, Smalltalk

Otros paradigmas de la programación

- Programación imperativa o por procedimientos
- Programación dinámica
- Programación dirigida por eventos
- Programación declarativa
- Programación funcional
- Programación lógica
- Programación con restricciones
- Programación multiparadigma
- Programación reactiva
- Lenguaje específico del dominio o DSL

Lenguajes y Paradigmas de la programación

Programación imperativa o por procedimientos	C, BASIC, Pascal
Programación dinámica	
Programación dirigida por eventos	
Programación declarativa	Lisp, Prolog
Programación funcional	Scheme (una variante de Lisp), Haskell, Python
Programación lógica	Prolog
Programación con restricciones	Prolog
Programación multiparadigma	Lisp, Python, C++, Genie, Delphi, Visual Basic, PHP, D5, Oz, Scheme,
Programación reactiva	
Lenguaje específico del dominio o DSL	SQL, Logo

Conceptos Fundamentales de la POO

Componentes de un Objeto

Clase

Mensajes

Herencia

Objeto

Método

Estado Interno

Identificación

Evento

Atributos

¿Objetos?

Un objeto es una **abstracción** de algún **hecho** o **ente** del mundo real, con **atributos** que representan sus características o propiedades, y **métodos** que emulan su comportamiento o actividad.

Todas las propiedades y métodos comunes a los objetos se encapsulan o agrupan en clases. Una **clase** es una **plantilla**, un prototipo **para crear objetos**; en general, se dice que **cada objeto es una instancia** o ejemplar **de una clase**.

Los objetos, son entonces, entidades que tienen un determinado **estado**, **comportamiento** e **identidad**.

Identidad

La identidad es una **propiedad de un objeto que lo diferencia del resto**.
Es decir, es su identificador.

Comportamiento y Estado

Los métodos (comportamiento) y atributos (estado) están estrechamente relacionados por la propiedad de conjunto.

Esta propiedad destaca que una clase requiere de métodos para poder tratar los atributos con los que cuenta. El programador debe pensar indistintamente en ambos conceptos, sin separar ni darle mayor importancia a alguno de ellos. Hacerlo podría producir el hábito erróneo de crear clases contenedoras de información por un lado y clases con métodos que manejen a las primeras por el otro. De esta manera se estaría realizando una "programación estructurada camuflada" en un lenguaje de POO.

Métodos de un Objeto

Desde el punto de vista del comportamiento, **es lo que el objeto puede hacer**.

Un método puede producir un cambio en las propiedades del objeto, o la generación de un "evento" con un nuevo mensaje para otro objeto del sistema.

Algoritmo asociado a un objeto (o a una clase de objetos), cuya ejecución se desencadena tras la recepción de un "mensaje".

Atributos o propiedades

Contenedor de un tipo de datos asociados a un objeto (o a una clase de objetos), que hace los datos visibles desde fuera del objeto y esto se define como sus características predeterminadas, y cuyo valor puede ser alterado por la ejecución de algún método.

Ejemplo: Auto

Un auto tiene atributos/propiedades, tales como:

- marca
- modelo
- color
- peso
- tipo de luces
- tipo de frenos
- cantidad de airbags
- y muchos más...

Ejemplo: Auto

También se le asocian ciertos comportamientos / métodos:

- Encender Motor
- Acelerar
- Frenar
- Apagar Motor
- Encender Luces Bajas
- Encender Luz de Giro izq.
- Encender Luz de Giro der.
- y muchos más...

Ejemplo: Auto

Ahora, también tiene **identificadores** que lo hacen “único”.

En Argentina, y en la mayoría de los países del mundo,
la **patente** se utiliza con tal fin.

Ejemplo: Auto

Pero… Si un auto es un objeto, que hay entonces de:

- Las ruedas
- El motor
- Asientos
- y muchos más….

Todos estos, pueden ser representados como objetos usando usando nuevas clases que los representen si fuera adecuado y relevante para el dominio de la solución que estuviéramos programando.

¿Solo podemos representar a objetos físicos?

Acuerdo de Palabra

Atributos posibles:

- fecha de realización
- realizado por
- aceptado por
- fecha de firmado
- tiempo de validez
- etc

Métodos posibles

- firmar
- cancelar
- modificar
- extender
- renovar
- etc

Una relación!

Atributos posibles:

- tipo: pareja, amistad, profesional, “complicada”
- cantidad de participantes
- fecha de inicio
- fecha de fin
- etc.

Métodos posibles

- comenzar
- finalizar
- celebrar_aniversario
- regalar_presente
- etc

Clase

Es la **plantilla que define** cuales van a ser las **posibles propiedades y comportamiento de un objeto** concreto.

Formalmente, la instanciación es la lectura de estas definiciones y la creación de un objeto a partir de ella.

En otras palabras, se dice que cuando instanciamos una clase, creamos un objeto de esa clase.

Herencia

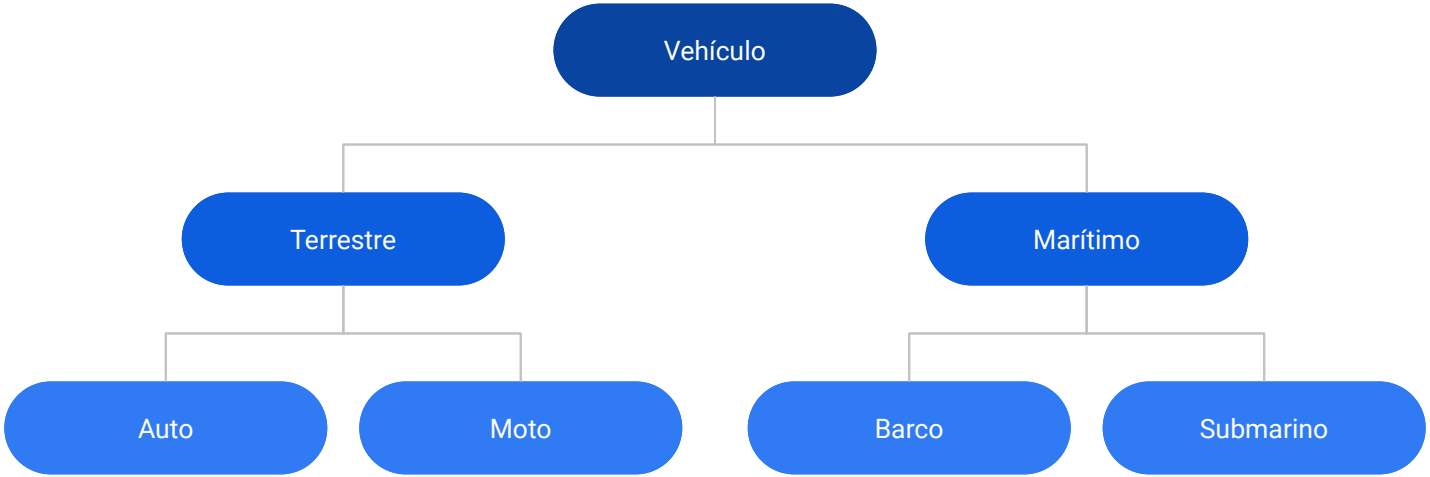
Dicho simple: Es la relación entre una clase general y otra clase más específica.

La herencia es, después de la agregación o composición, el mecanismo más utilizado para alcanzar algunos de los objetivos más preciados en el desarrollo de software como lo son la reutilización y la extensibilidad.

A través de ella, los diseñadores pueden crear nuevas clases partiendo de una clase o de una jerarquía de clases preexistente (ya comprobadas y verificadas) evitando con ello el rediseño, la modificación y verificación de la parte ya implementada. La herencia facilita la creación de objetos a partir de otros ya existentes e implica que una subclase obtiene todo el comportamiento (métodos) y eventualmente los atributos (variables) de su superclase.

Herencia

En el ejemplo del Auto, podemos definir la siguiente jerarquía:



Estado interno

Es una **variable** que se declara privada, **que puede ser únicamente accedida y alterada por un método del objeto**, y que se utiliza para indicar distintas situaciones posibles para el objeto (o clase de objetos).

No es visible al programador que maneja una instancia de la clase.

Características Fundamentales de la POO

Principio de Ocultación

Encapsulamiento

Modularidad

Herencia

Abstracción

Polimorfismo

Recolección de Basura

Abstracción

Denota las características esenciales de un objeto, donde se capturan sus comportamientos. Cada objeto en el sistema sirve como modelo de un "agente" abstracto que puede realizar trabajo, informar y cambiar su estado, y "comunicarse" con otros objetos en el sistema **sin revelar "cómo" se implementan estas características.**

Abstracción

En el ejemplo del auto, para “encender el auto”, se debe hacer uso de otros métodos, como:

- iniciar_circuitos
- arrancar_motor
- iniciar_servofreno
- activar_airbags
- iniciar_abs
- iniciar_control_de_estabilidad
- iniciar_control_de_tracción

Pero… realmente al que maneja, no le interesan estos detalles.

Encapsulamiento

Significa reunir todos los elementos que pueden considerarse pertenecientes a una misma entidad, al mismo nivel de abstracción. Es decir, agrupar todo lo que tiene sentido para tal abstracción.

Esto permite aumentar la cohesión (diseño estructurado) de los componentes de sistema.

No confundir con principio de ocultación.

Cohesión: La cohesión se refiere al grado en que los elementos de un módulo permanecen juntos. Por lo tanto, la cohesión mide la fuerza de la relación entre las piezas de funcionalidad dentro de un módulo dado.

Principio de Ocultación

En programación orientada a objetos el principio de ocultación hace referencia a que los atributos privados de un objeto no pueden ser modificados ni obtenidos a no ser que se haga a través del paso de un mensaje (invocación a métodos, ya sean estos funciones o procedimientos).

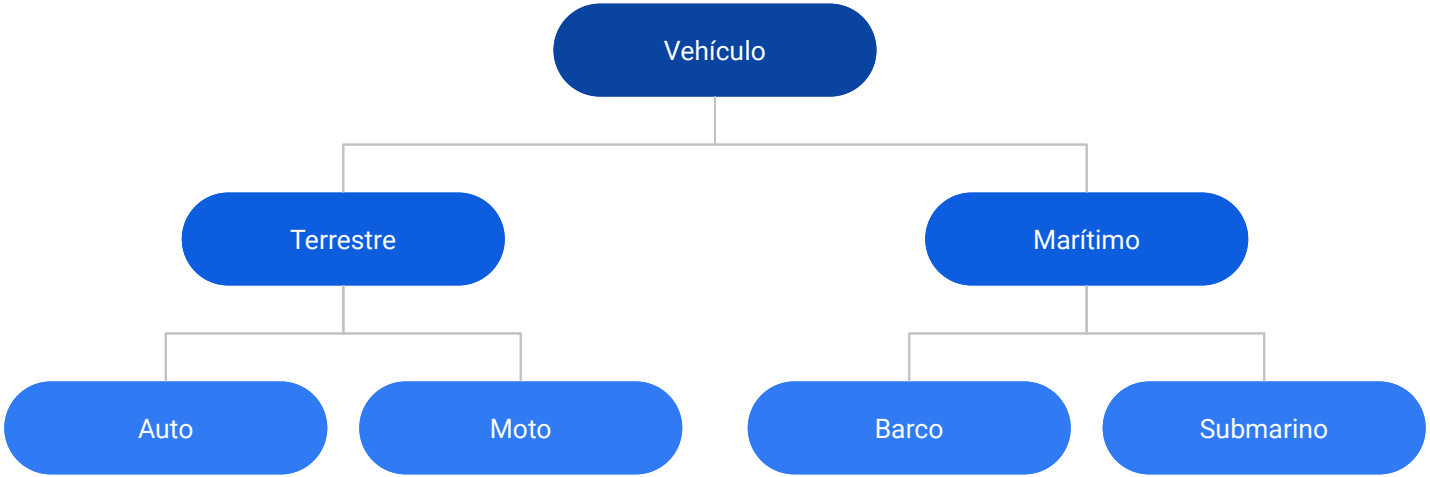
Polimorfismo

Comportamientos diferentes, asociados a objetos distintos, pueden **compartir el mismo nombre**; al llamarlos por ese nombre se utilizará el comportamiento correspondiente al objeto que se esté usando.

O, dicho de otro modo, las referencias y las colecciones de objetos pueden contener objetos de diferentes tipos, y la invocación de un comportamiento en una referencia producirá el comportamiento correcto para el tipo real del objeto referenciado.

Polimorfismo

¿Cómo es la implementación del método **encender_vehículo** en los distintos vehículos?

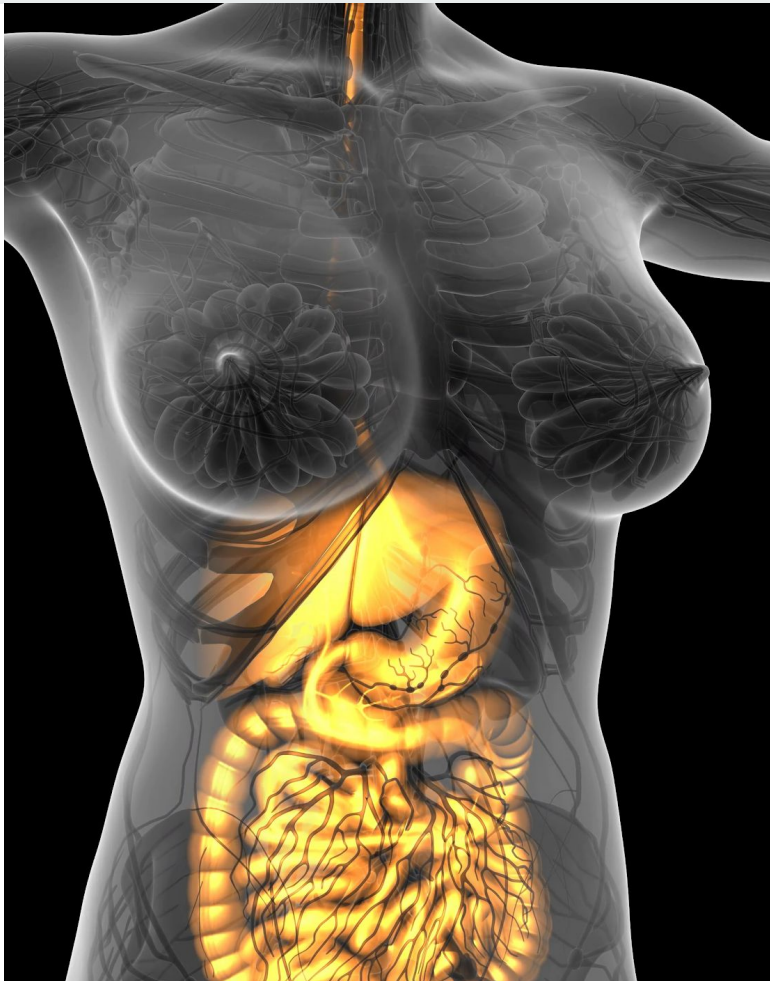


Modularidad

Se denomina "modularidad" a la propiedad que permite subdividir una aplicación en partes más pequeñas (llamadas módulos), cada una de las cuales debe ser tan independiente como sea posible de la aplicación en sí y de las restantes partes.

Estos módulos se pueden compilar por separado, pero tienen conexiones con otros módulos.

Al igual que la encapsulación, los lenguajes soportan la modularidad de diversas formas.



Recolección de Basura

La recolección de basura (garbage collection) es la técnica por la cual el entorno de objetos se encarga de destruir automáticamente, y por tanto desvincular la memoria asociada, los objetos que hayan quedado sin ninguna referencia a ellos.

Esto significa que el programador no debe preocuparse por la asignación o liberación de memoria, ya que el entorno la asignará al crear un nuevo objeto y la liberará cuando nadie lo esté usando.

En la mayoría de los lenguajes híbridos que se extendieron para soportar el Paradigma de Programación Orientada a Objetos como C++ u Object Pascal, esta característica no existe y la memoria debe desasignarse expresamente.