CARACTERISTICAS POO

Abstracción

Puede definirse como:las características especificas de un objeto, aquellas que lo distinguen de los demás tipos de objetos y que logran definir límites conceptuales respecto a quien está haciendo dicha abstracción del objeto.

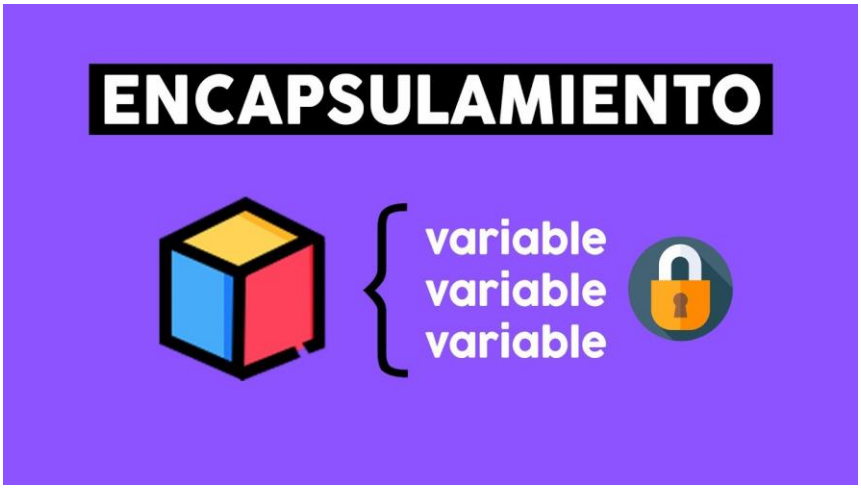
Una abstracción se enfoca en la visión externa de un objeto, separa el comportamiento específico de un objeto, a esta división que realiza se le conoce como la barrera de abstracción, la cuál se consigue aplicando el principio de mínimo compromiso

Encapsulamiento

Es el proceso de almacenar en una misma sección los elementos de una abstracción que constituyen su estructura y su comportamiento; sirve para separar el 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



Herencia:

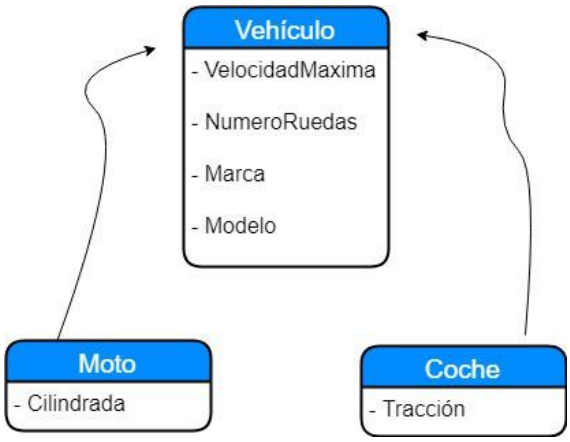
Herencia es un concepto de la programación orientada a objetos. El cual es un mecanismo que permite derivar una clase a otra clase.

En otras palabras, tendremos unas clases que serán hijos, y otras clases que serán padres.

Las clases hijas pueden utilizar tanto sus métodos y propiedades como de la clase padre, siempre que su modificador de acceso lo permita.

Por ejemplo, el siguiente código

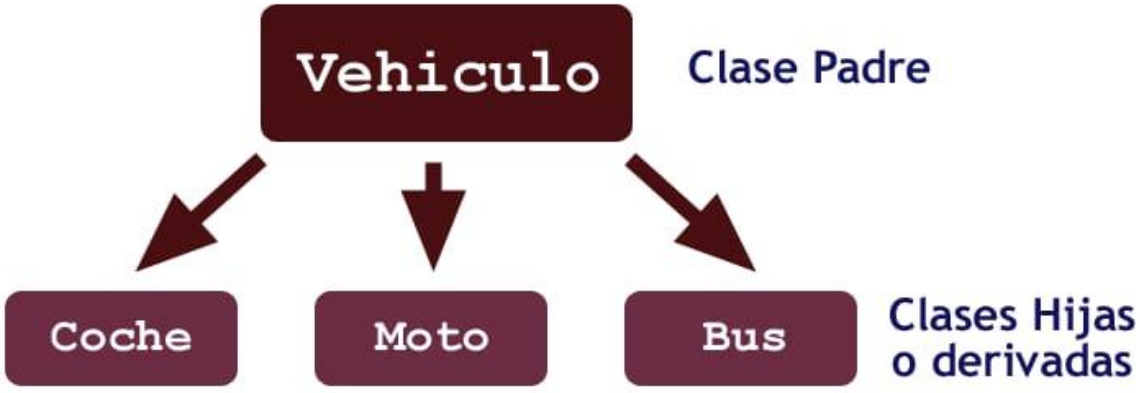
```
class Vehiculo
{
    public decimal VelocidadMaxima { get; set; }
    public int NumeroRuedas { get; set; }
    public string Marca { get; set; }
    public string Modelo { get; set; }
}
class Moto : Vehiculo
{
    public int Cilindrada { get; set; }
}
class Coche : Vehiculo
{
    public string Traccion { get; set; }
}
```



El polimorfismo:

es una relajación del sistema de tipos, de tal manera que una referencia a una clase (atributo, parámetro o declaración local o elemento de un vector) acepta direcciones de objetos de dicha clase y de sus clases derivadas (hijas, nietas, ...)

Veremos que el polimorfismo y la herencia son dos conceptos estrechamente ligados. Conseguimos implementar polimorfismo en jerarquías de clasificación que se dan a través de la herencia. Por ejemplo, tenemos una clase vehículo y de ella dependen varias clases hijas como coche, moto, autobús, etc



Clases y objetos:

Una clase es un elemento de la programación orientada a objetos que actúa como una plantilla y va a definir las características y comportamientos de una entidad. La clase va a ser como un molde a partir del cual vamos a poder definir entidades. Una clase va a definir las características y los comportamientos de una entidad. Si yo defino, por ejemplo, la clase persona, sus características o atributos podrían ser género, es decir, si es hombre o mujer, edad y nombre. Los comportamientos o métodos que tendríamos en el caso de la clase persona podrían ser respirar, moverse, caminar, pensar, entre otras.

Una clase es un elemento de la programación orientada a objetos que actúa como una plantilla y va a definir las características y comportamientos de una entidad. La clase va a ser como un molde a partir del cual vamos a poder definir entidades. Una clase va a definir las características y los comportamientos de una entidad. Si yo defino, por ejemplo, la clase persona, sus características o atributos podrían ser género, es decir, si es hombre o mujer, edad y nombre. Los comportamientos o métodos que tendríamos en el caso de la clase persona podrían ser respirar, moverse, caminar, pensar, entre otras.

Aunque los conceptos de clase y objetos son muy diferentes, el hecho de usarlos indistintamente en algunos contextos hace que se pueda generar alguna confusión al respecto.

En la figura 2.4a y figura 2.4b se muestra, para el caso de la tienda, el correspondiente diagrama de clases y un ejemplo de un posible diagrama de objetos. Allí se puede apreciar que la clase Tienda describe todas las tiendas imaginables que vendan 4 productos.

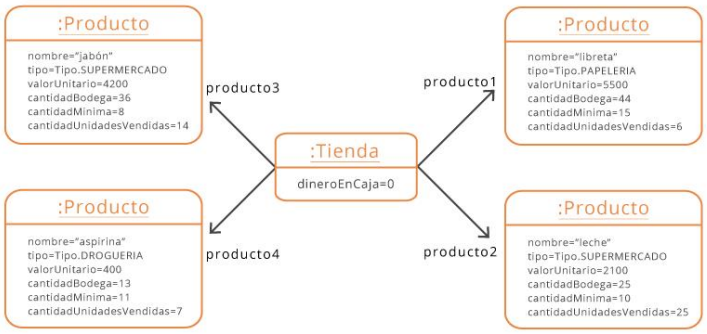
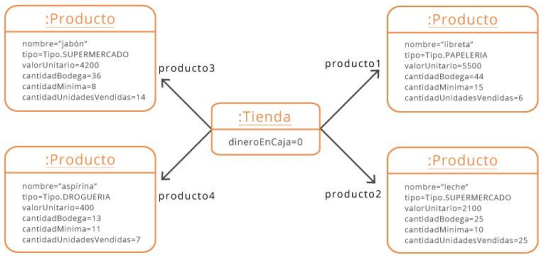


Fig. 2.4a Modelo de clases

Diagrama de clases para el caso de estudio de la tienda.

El diagrama sólo dice, por ejemplo, que producto1 debe ser un producto.

Fig. 2.4b Modelo de objetos



Fíjese como cada asociación del diagrama de clases debe tener su propio objeto en el momento de la ejecución

Métodos y atributos:

Los atributos: Son las propiedades que poseen los objetos de esa clase.

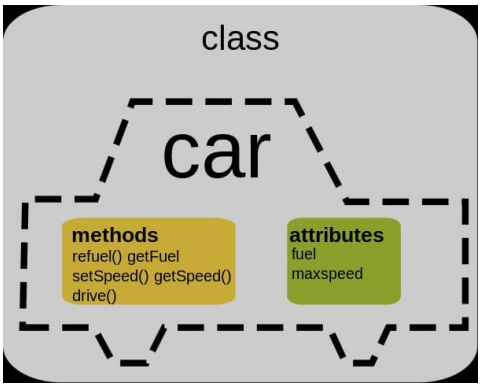
Métodos: es un modelo de programación que utiliza objetos, ligados mediante mensajes, para la solución de problemas. Puede considerarse como una extensión natural de la programación estructurada en un intento de potenciar los conceptos de modularidad y reutilización del código

Sigamos ahora el ejemplo del modelado de un coche, podemos decir que tendremos una clase llamada "Coche", que contiene en su interior varios atributos (el color, las dimensiones, el número de plazas, etc) y varios métodos que puedan ejecutar acciones (arrancar, acelerar, frenar,...).

Cuando programes, verás que los atributos se implementan usando variables y los métodos se implementan utilizando funciones y procedimientos. De esta forma, todo lo que has estudiado anteriormente lo volverás a aplicar en orientación a objetos.

Podemos decir por tanto que una Clase es un conjunto de elementos que tienen unas características comunes, es decir, que son del mismo tipo. El concepto de clase es un concepto abstracto.

Un Objeto en cambio será la concreción de uno de los elementos de una Clase, que permitirá almacenar tanto los atributos como las posibles acciones que realizará.



Modularidad:

La modularidad en la programación orientada a objetos se refiere a la capacidad de dividir un programa en módulos independientes y reutilizables. Estos módulos, también conocidos como clases, encapsulan tanto los datos como los métodos relacionados en una entidad coherente. Cada clase es responsable de realizar una única tarea específica y puede interactuar con otras clases a través de interfaces bien definidas.

La modularidad también tiene principios y son los siguientes:

Capacidad de descomponer un sistema complejo.

Capacidad de componer a través de sus módulos.

Comprensión de sistema en partes.

Pero.. ¿Qué significa cada módulo?

Capacidad de descomponer un sistema complejo

Recuerdas el principio de «Divide y Vencerás», en este procedimiento se realiza algo similar, ya qué descompones un sistema en subprogramas (recuerda llamarlos módulos), el problema en general lo divides en problemas más pequeños.

Capacidad de componer a través de sus módulos

Indica la posibilidad de componer el programa desde los problemas más pequeños complentado 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.

Comprensión de 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.

Una de las razones por la cuál es útil hacerlo modular es debido a que podemos tener los límites bien definidos y claro es lo más valioso a la hora de leer el programa, recordemos una de las buenas prácticas «Programe pensando en quien mantendrá el código».

No está de más mencionar que en algunos lenguajes orientados a objetos no existe el concepto de módulo, siendo la clase la única unidad de descomposición.

Reusabilidad:

La reusabilidad de un elemento software podemos definirla co- mo “una medida de la facilidad con la que pueden utilizarse conceptos o elementos software en nuevas situaciones”

Otra propiedad fundamental de la programación orientada a objetos es la reutilización o reusabilidad; este concepto significa que una vez que se ha creado, escrito y depurado una clase, se puede poner a disposición de otros programadores; de manera similar, al uso de las bibliotecas de funciones en un lenguaje de programación. El concepto de herencia en Java proporciona una ampliación o extensión importante a la idea de reusabilidad; una clase existente se puede ampliar añadiéndole nuevas características, atributos y operaciones.

Fundamentos de programación

Polimorfismo

Java está orientado por completo a objetos y al familiarizarse totalmente con la POO puede lograrse un desarrollo más productivo y aumentar el rendimiento y desempeño. La programación orientada a objetos (POO) es un enfoque conceptual específico para diseñar programas utilizando un lenguaje que se centra en los objetos, cuyas propiedades más importantes son:

- Abstracción,
- Encapsulamiento y ocultación de datos,
- Herencia,
- Polimorfismo y
- Reusabilidad o reutilización de código