PROGRAMACIÓN ORIENTADA A OBJETOS

Es un *paradigma* (modelo de trabajo) de programación que agrupa los datos y los procedimientos para manejarlos en una única entidad: *el objeto*.

- La programación orientada a objetos es conocida como un estilo de programación que utiliza objetos y sus relaciones entre sí, a su vez estos objetos se pueden organizar en clases.

- Este modo de programar intenta representar el mundo real, donde podemos observar que cada cosa que existe es un objeto.

- Por ejemplo, si pensamos de esta manera y vemos a nuestro alrededor, nos daremos cuenta que la silla donde estamos sentados es un objeto, la mesa, un foco, un lápiz, entre otros, todo es un objeto y esto es lo que toma la programación orientada a objetos para modelar un sistema.

OBJETO

Un objeto es una unidad que engloba en sí mismo variables y funciones necesarios para el tratamiento de esos datos.

Un objeto podría ser real o abstracto, por ejemplo una organización, una factura, una figura en un dibujador, una pantalla de usuario, un avión, un vuelo de avión, etc.

En el software orientado a objeto, un objeto es cualquier cosa, real o abstracta, acerca de la cual almacenamos datos y los métodos que controlan dichos datos.

- Si volvemos a pensar en el mundo real, cada objeto tiene ciertas características que lo hacen distinto de otros objetos, estas características se conocen con el nombre de **ATRIBUTOS.**

- Otro elemento importante de los objetos son las acciones que estos pueden realizar, por ejemplo si tenemos un lápiz, las acciones que podríamos hacer con el serían dibujar, borrar, escribir… a estás acciones que realizan los objetos se conocen como **MÉTODOS** .

Objeto

Atributos:	Métodos:
Nombre	Ladrando
Color	Oliendo
Raza	Buscando
Hambriento	Meneando la cola

Calculadora

Atributos: color, teclas, pantalla, volumen, sistema electrónico.

Comportamientos: suma, resta, multiplicación, división, entre otras operaciones matemáticas.

Helicóptero

Atributos: hélices, metal, cristales, motor.

Comportamientos: volar, avanzar a gran velocidad, dar vueltas, bajar y subir, suspenderse en el aire.

ATRIBUTO

Variable perteneciente a un determinado objeto.

MÉTODO

Función perteneciente a un determinado objeto.

- Los objetos se comunican e interrelacionan entre si a través del acceso a sus atributos y del llamado a sus métodos.
- No se puede tener acceso o control de la estructura de datos excepto mediante los métodos que forman parte del tipo de objeto.

Los objetos que son similares podemos agruparlos en clases, como lo entendemos en el mundo real.

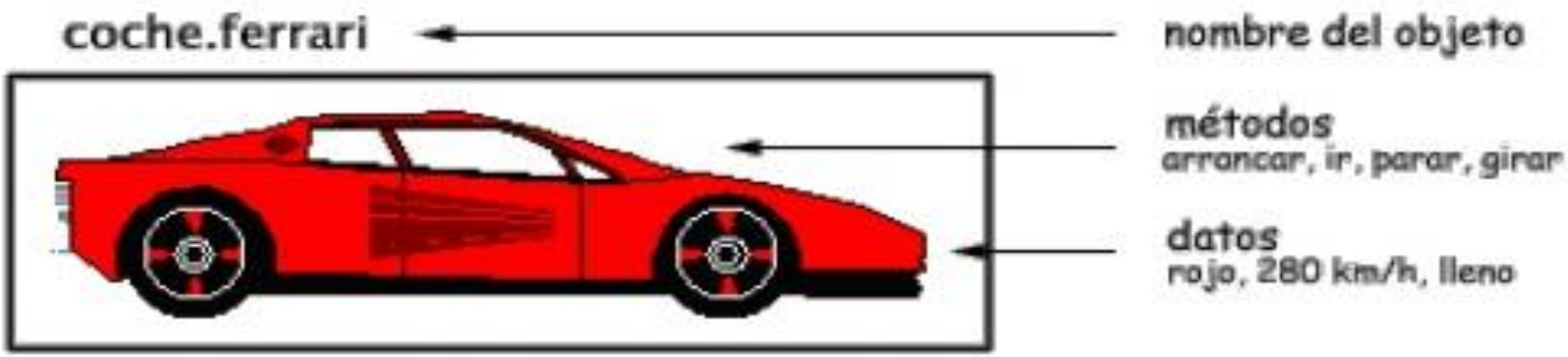
CLASE

Una clase se puede considerar como un patrón para construir objetos.

EJEMPLO DE CLASES Y OBJETOS

Clase: *Coche*

- **Objeto**: *Ferrari*



INSTANCIAR

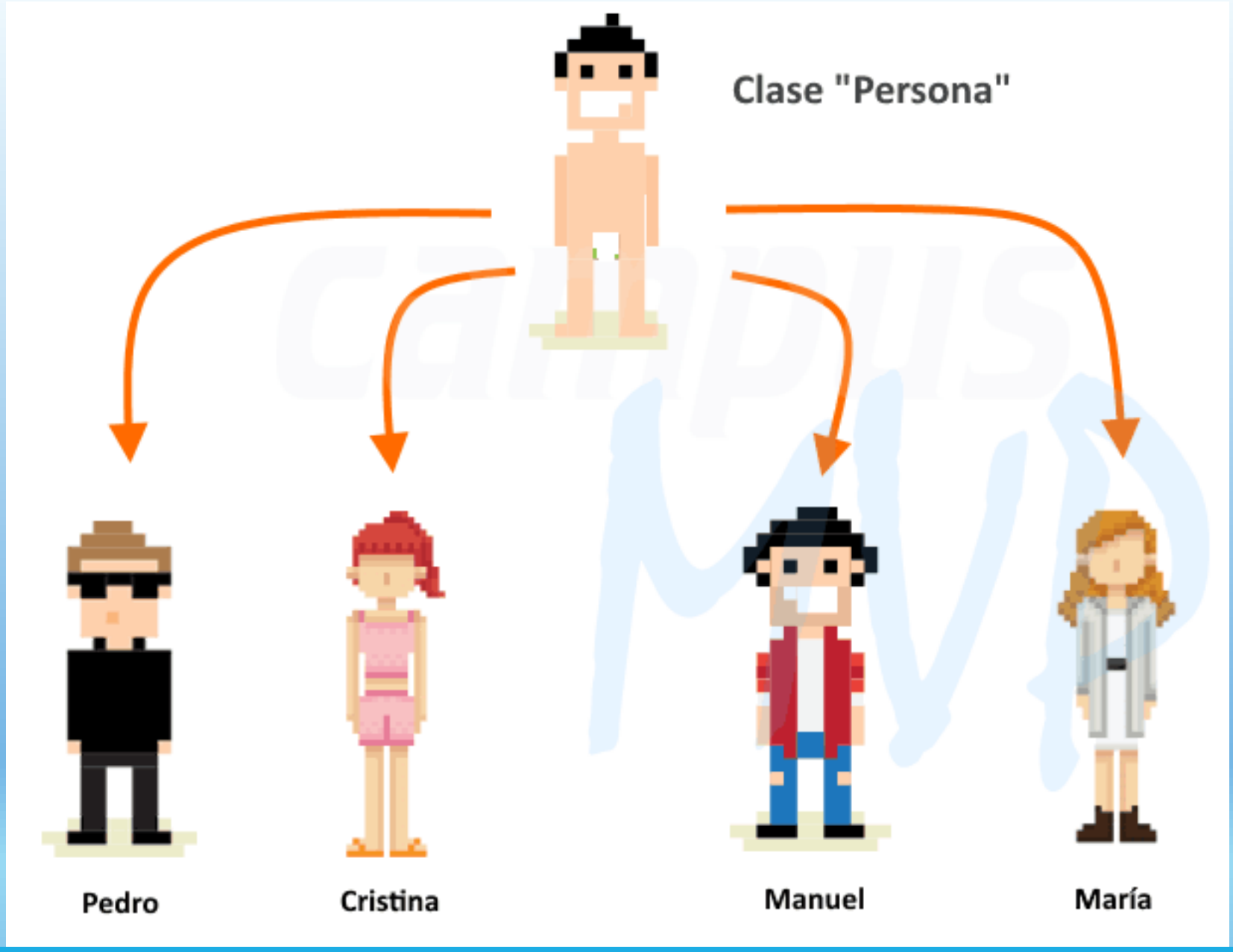
- Una clase por sí sola no sirve de nada, pues no es más que un concepto, sin entidad real.

- Para poder utilizar una clase en un programa lo que hay que hacer es **instanciarla**.

- **INSTANCIAR** una clase consiste en **crear un nuevo objeto concreto de la misma**.

- Es decir, **un objeto es ya una entidad concreta** que se crea a partir de la plantilla que es la clase.

- Así un objeto puede ser una persona que se llama Cristina López, de 37 años y que en nuestro programa podría hablar, caminar o comer, que son los comportamientos que están definidos en la clase.


Clase "Persona"
Pedro
Cristina
Manuel
María

MENSAJE

- Para que un objeto haga algo, le enviamos una solicitud. Esta hace que se produzca una operación.

- La operación ejecuta el método apropiado y, de manera opcional, produce una respuesta.

- El mensaje que constituye la solicitud contiene el nombre del objeto, el nombre de una operación y, a veces, un grupo de parámetros.

Un mensaje es una solicitud para que se lleve a cabo la operación indicada y se produzca el resultado.


CLASE
Propiedad y comportamiento de un objeto concreto
ATRIBUTO
Propiedad del objeto
MÉTODO
Lo que un objeto puede hacer (algoritmo)
OBJETO
Instancia de una clase
MENSAJE
Comunicación dirigida a un objeto ordenándole que ejecute uno de sus métodos

INTERFAZ

- Es la parte del objeto que es visible para el resto de los objetos.

- Es decir, el conjunto de métodos y atributos que dispone un objeto para comunicarse con él.

- Un objeto es sólo una instancia (un ejemplar) de una clase determinada.

- Las clases tienen partes públicas y partes privadas.

- Generalmente, llamaremos a la parte pública de una clase su interfaz.

MODIFICADORES DE ACCESO

Son palabras clave que se utilizan para controlar el acceso a los campos y métodos de una clase.

Estos modificadores determinan cómo otras partes del programa pueden acceder y modificar los atributos y métodos.

. Los lenguajes de programación suelen tener diferentes tipos de modificadores de acceso, aunque los más comunes son:

1.Public: Este es el modificador menos restrictivo. Los campos y métodos declarados como públicos pueden ser accesibles desde cualquier clase.

2.Private: Los campos y métodos declarados como privados solo son accesibles dentro de la propia clase en la que se definen. No pueden accederse desde fuera de la clase, ni siquiera desde sus clases derivadas.

3.Protected: Las clases y métodos protegidos permiten el acceso desde la propia clase y también desde sus clases derivadas. Sin embargo, no se puede acceder a los elementos desde fuera de la jerarquía de herencia.

4.Paquete o Default: Existe un modificador de acceso adicional que se aplica por defecto si no se especifica ningún otro modificador. Este modificador permite el acceso a los campos y métodos desde otras clases en el mismo paquete, pero restringe el acceso desde fuera del paquete. c

Pilares de la POO


Poliformismo
Permite dar la misma orden a varios objetos de distintas clases.
Abstracción
Proceso en el que se definen los atributos y métodos de una clase.
Encapsulamiento
Protege los datos de manipulaciones no aurotizadas.
Herencia
Las clases pueden heredar los atributos de las clases bases.
Principios de la Programación orientada a objetos

ABSTRACCIÓN

- El proceso de abstracción es pensar que atributos y qué métodos debe tener un objeto.
- Cuando se desarrolla con OOP, es necesario entender el problema y centrarse en identificar los objetos relevantes, las propiedades y métodos que necesitaremos para solucionarlo, esto ayuda a reducir la complejidad.
- Por ejemplo, si queremos realizar un personaje de un juego necesitamos una serie de atributos, como puede ser un nombre, fuerza, cantidad de vidas, etc. En cambio, sus métodos podrían ser moverse, atacar, entre otros.

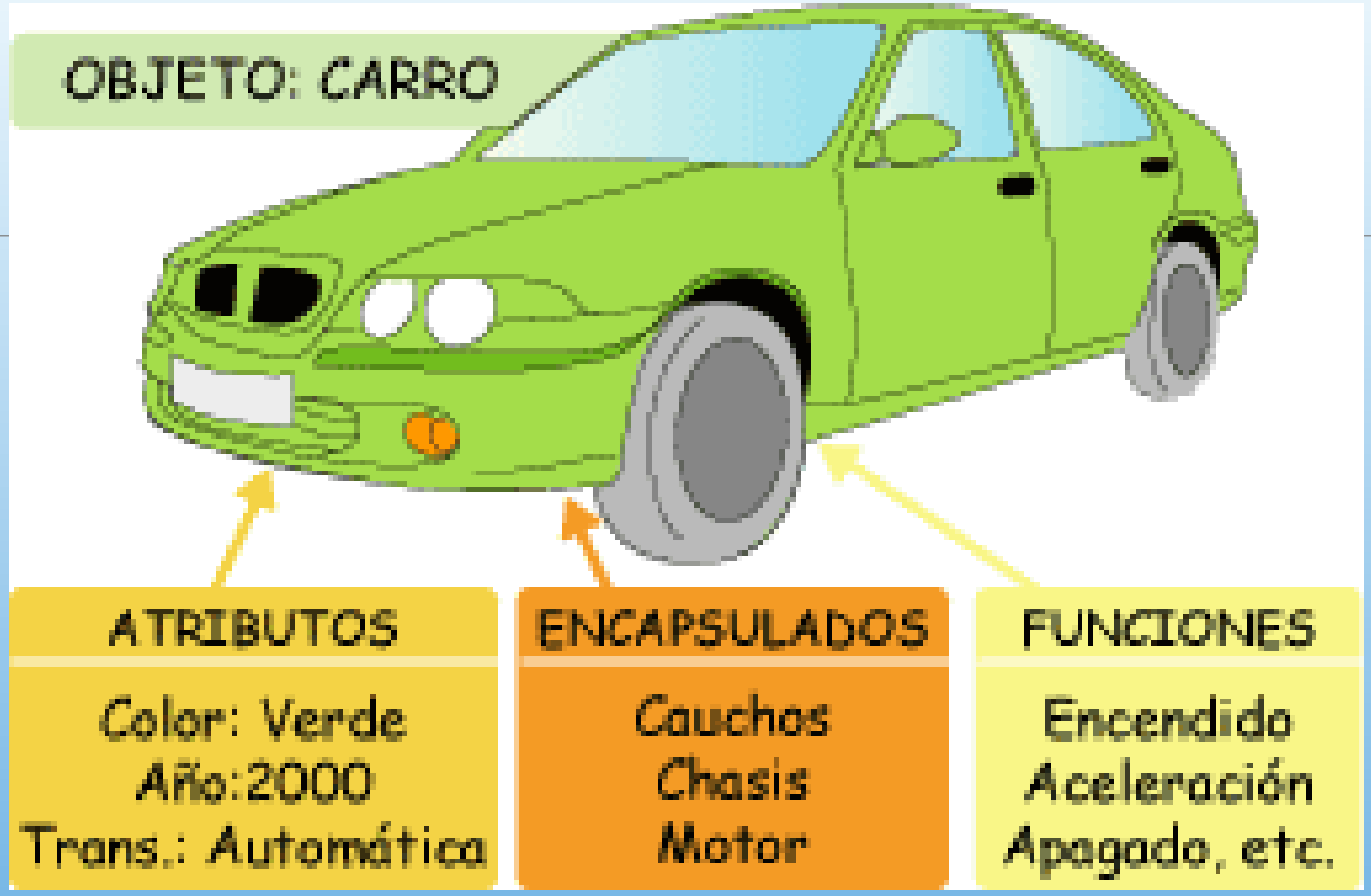
ENCAPSULAMIENTO

- Los objetos se comunican entre ellos. Esto podría traer problemas de seguridad si un objeto puede modificar los datos de cualquier otro.
- Por eso, se necesita proteger la información de manipulaciones no autorizadas. De esta manera, cuando se comunican los objetos, hay caminos que se pueden seguir y hay caminos que no, datos protegidos, datos privados o públicos, métodos para acceder a cierta información, entre otros.

La **encapsulación** es un mecanismo que consiste en organizar datos y métodos de una estructura, conciliando el modo en que el **objeto** se implementa, es decir, evitando el acceso a datos por cualquier otro medio distinto a los especificados.

Por ejemplo, en el caso de las personas, toda la información sobre éstas (nombre, apellidos, edad... y cualquier otro dato interno que se utilice y que no necesariamente se ve desde el exterior del objeto) está circunscrito al ámbito de dicha persona.

Así, internamente tenemos un dato que es el nombre de la persona y accedemos a él a través de la propiedad pública Nombre que define la clase que representa a las personas. De este modo damos acceso sólo a lo que nos interese y del modo que nos interese.


OBJETO: CARRO
ATRIBUTOS
Color: Verde
Año:2000
Trans.: Automática
ENCAPSULADOS
Cauchos
Chasis
Motor
FUNCIONES
Encendido
Aceleración
Apagado, etc.

POLIMORFISMO

- El **polimorfismo** se refiere a la propiedad por la que es posible enviar mensajes sintácticamente iguales a **objetos** de tipos distintos.

- El único requisito que deben cumplir los **objetos** que se utilizan de manera polimórfica es saber responder al mensaje que se les envía.

- Supongamos que en nuestro juego tenemos un montón de personajes que están juntos en un mismo escenario. Hay varios piratas, algunos estrategas y un montón de otros tipos de personas.

- En un momento dado necesitamos que todos se pongan a hablar. Cada uno lo hace de una forma diferente, ya que son tipos de personajes distintos. Sería algo bastante tedioso tener que localizar primero a los de un tipo y hacerlos hablar, lo luego a los de otro y así sucesivamente.

- La idea es que puedas tratarlos a todos como personas, independientemente del tipo específico de persona que sean y simplemente decirles que hablen.

- Al derivar todos de la clase Persona todos pueden hablar, y al llamar al método Hablar() de cada uno de ellos se utilizará el proceso adecuado según el tipo (los piratas meterán sus expresiones adicionales que hemos visto, los pilotos dirán "Entrando en pista" o lo que sea, y los estrategas añadirán a todo "Déjame que lo piense bien"). Todo esto de manera transparente para el programador. Esto es el polimorfismo

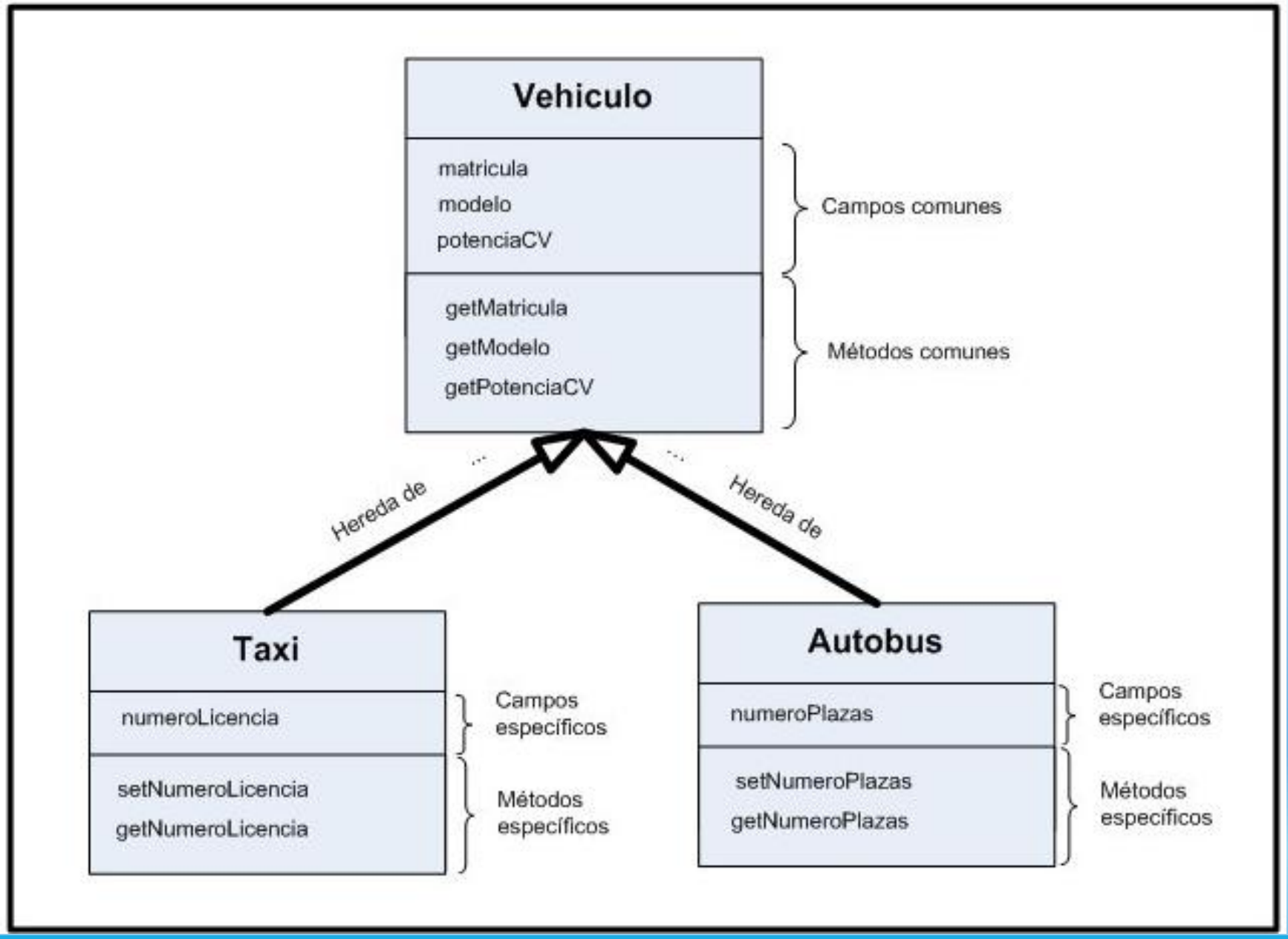
> El polimorfismo nos permite utilizar a los objetos de manera genérica, aunque internamente se comporten según su variedad específica.

HERENCIA

- La *herencia* (a la que habitualmente se denomina *subclases*) proviene del hecho de que la subclase (la nueva clase creada) contiene los atributos y métodos de la clase primaria.

- La principal ventaja de la herencia es la capacidad para definir atributos y métodos nuevos para la subclase, que luego se aplican a los atributos y métodos heredados.

- Esta particularidad permite crear una estructura jerárquica de clases cada vez más especializad


Vehiculo
matricula
modelo
potenciaCV
Campos comunes
getMatricula
getModelo
getPotenciaCV
Métodos comunes
Hereda de
Hereda de
Taxi
numeroLicencia
Campos específicos
setNumeroLicencia
getNumeroLicencia
Métodos específicos
Autobus
numeroPlazas
Campos específicos
setNumeroPlazas
getNumeroPlazas
Métodos específicos

EJERCICIO

- Realizar un programa que conste de una clase llamada Estudiante, que tenga como atributos el nombre y la nota del alumno. Definir los métodos para inicializar sus atributos, imprimirlos y mostrar un mensaje con el resultado de la nota y si ha aprobado o no.

- Crea una clase “Persona”. Con atributos nombre y edad. Además, hay que crear un método “cumpleaños”, que calcule la edad de la persona.