

Programación

5.1 Introducción a la POO

Introducción

- Un **paradigma** de programación es un estilo de desarrollo de programas. Esto es, un modelo para resolver problemas computacionales.
- Hasta ahora hemos utilizado el **paradigma de programación estructurada**, que emplea estructuras de control (condicionales y bucles) haciendo uso de datos y funciones de manera independiente. Una de las principales desventajas de ese paradigma es que no existe un **vínculo fuerte entre funciones y datos**, lo que puede dificultar problemas más complejos. Llegados a este punto saltaremos a un nuevo paradigma, la **programación orientada a objetos** (POO) que nos dotará de nuevas herramientas que faciliten la resolución de problemas más complejos.

Introducción

- Una de las dificultades permanentes será **decidir cómo traducir el problema a resolver** en estructuras de datos y cómo interactúan entre ellas.
 - Para problemas sencillos: **cada dato** → **tipo de dato primitivo o basado directamente en primitivos** (arrays, Strings...)
 - Para problemas complejos: metodología que decida **cómo agrupar estos datos**

Bases de la POO

Abstracción:

Proceso al de extracción de las **características esenciales** de algo, ignorando los detalles superfluos.

La abstracción consiste en aislar un elemento de su contexto, centrándose en el "**qué hace**" y no en el "cómo lo hace". Permite analizar los elementos considerando sólo las propiedades esenciales

Bases de la POO

Si los ingenieros de software hubiesen nombrado el automóvil...



public class PistonCrankshaftGearWheelAssembly



- *Una buena abstracción debe introducir un vocabulario distinto al de los componentes subyacentes.*
- *Debe proteger al usuario de la complejidad subyacente, permitiéndole entender el concepto sin necesidad de conocer los detalles internos.*

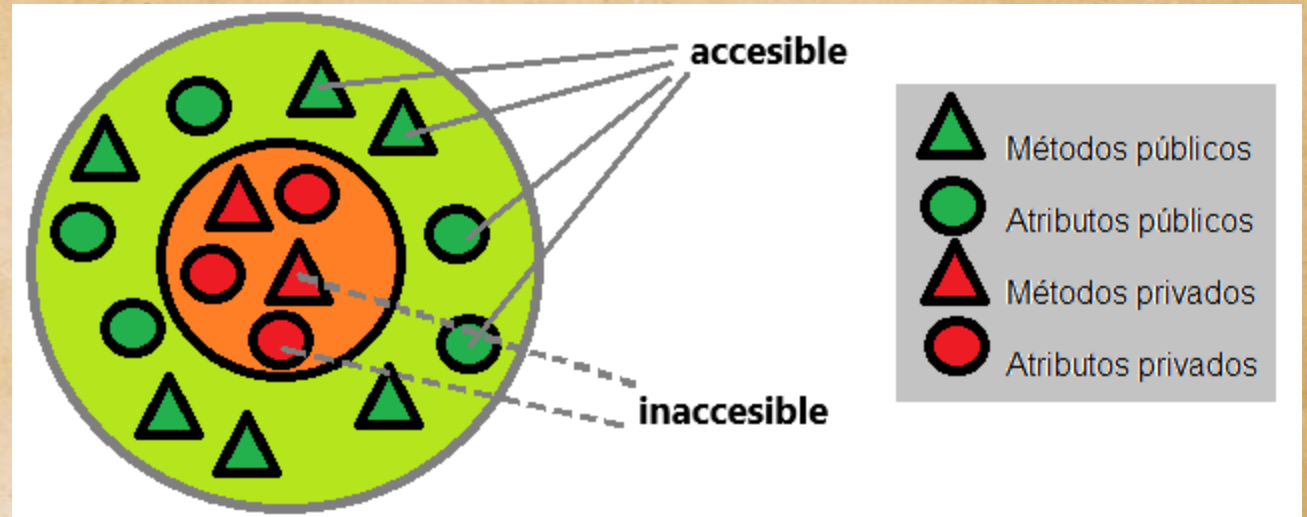
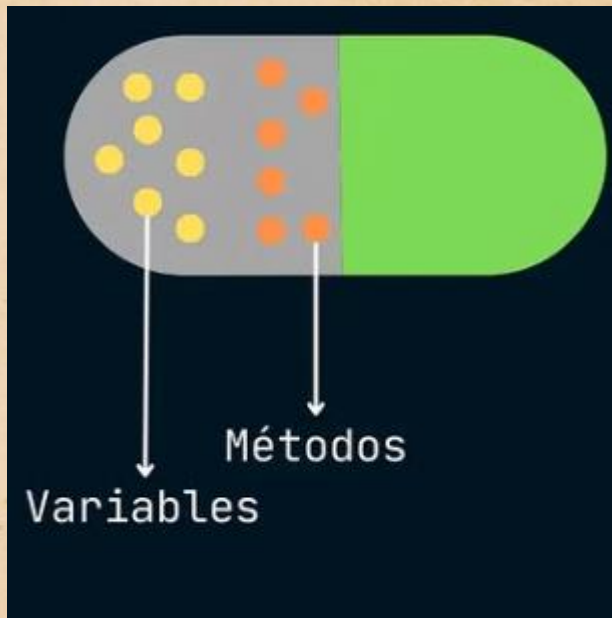
Bases de la POO

Encapsulación :

Es el empaquetamiento de información, es decir datos (atributos) y comportamiento (métodos) **en una sola unidad o componente**, ocultando el estado interno de un objeto y restringiendo el acceso a sus miembros.

Esta encapsulación de datos también implica "**ocultación de la información**" ya que **protege el estado interno** del objeto contra modificaciones imprevistas. Sólo permite el acceso mediante **métodos específicos expuestos** por la clase y por tanto garantiza la **integridad de los datos**.

Bases de la POO



Bases de la POO

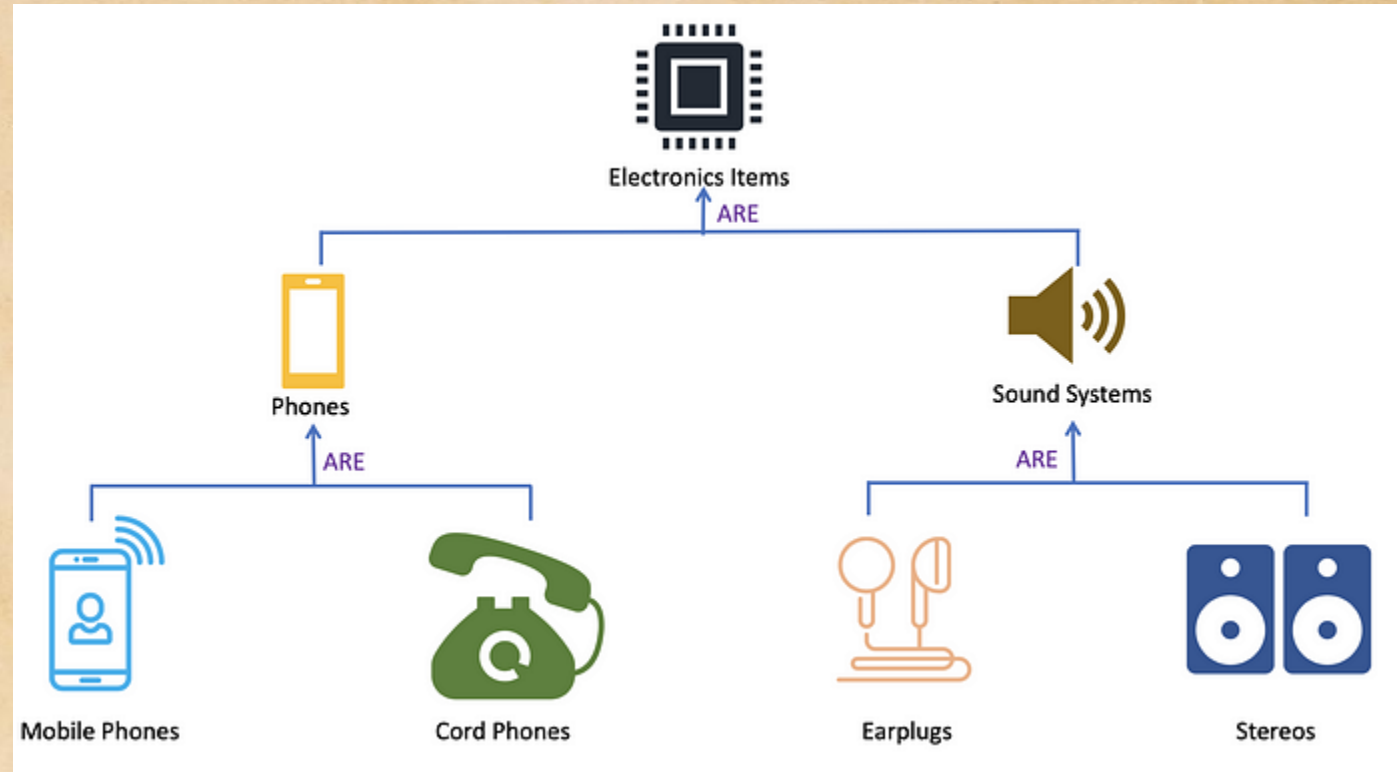
Herencia :

Es un mecanismo que permite a una clase (denominada subclase o clase hija) derivar de otra clase (denominada superclase o clase padre), adquiriendo sus métodos y atributos. **La subclase hereda el comportamiento y las características** de la superclase, permitiendo así la reutilización de código, la extensión de funcionalidades y el polimorfismo.

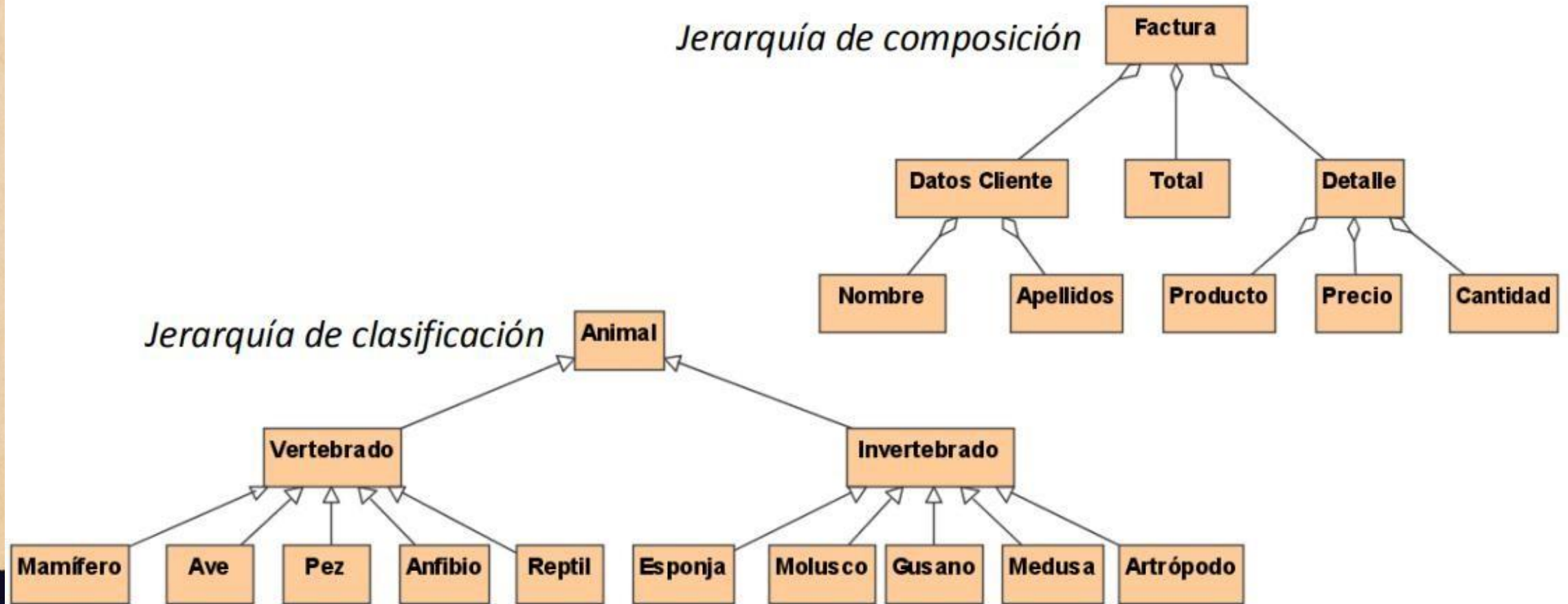


*La herencia es una técnica poderosa que va más allá de la simple "copia y pegamento" del código. Proporciona una estructura para **organizar, reutilizar y extender código**, permitiendo la implementación de **relaciones de subtipo y el polimorfismo**, aunque su uso debe ser equilibrado con otras técnicas como la **composición** para conseguir un diseño de software robusto y mantenible.*

Bases de la POO



Bases de la POO

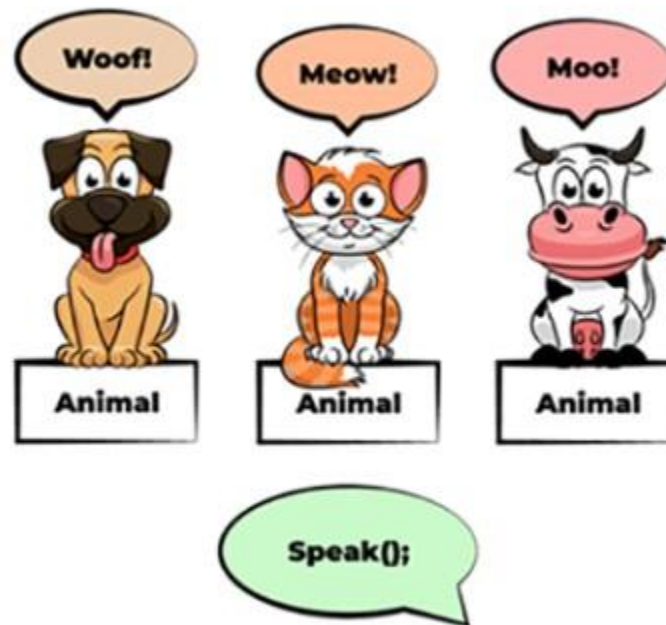


Bases de la POO

Polimorfismo:

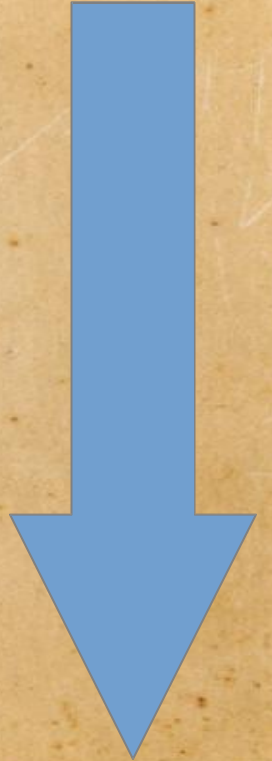
El polimorfismo, que significa "de muchas formas", es una técnica que permite a objetos de distintas clases **responder a un mismo mensaje (método) de forma específica para cada uno**, facilitando la reutilización, flexibilidad y abstracción.

Bases de la POO

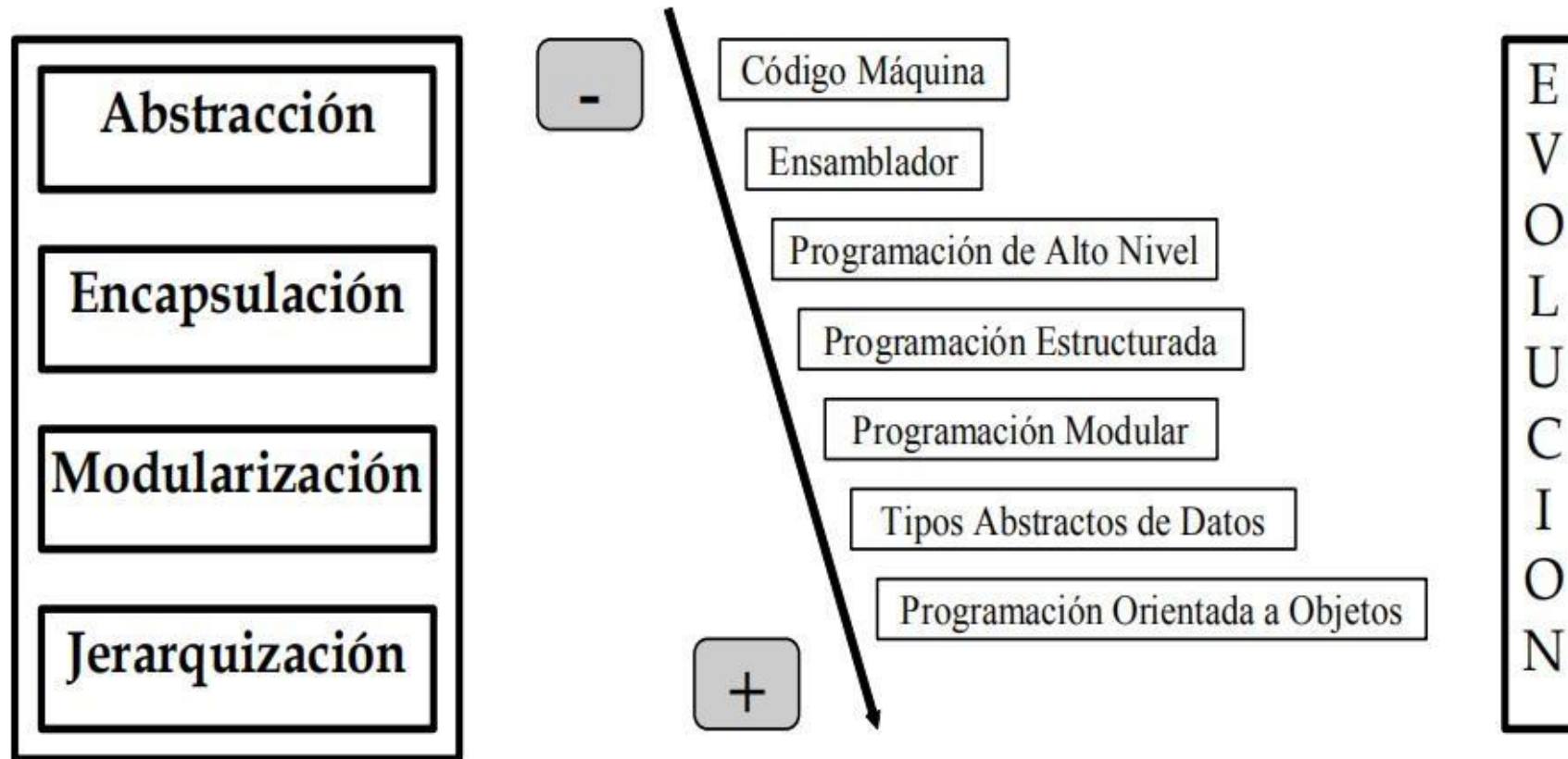


Evolución de los paradigmas de programación

1. Código máquina (cadenas de 0's y 1's)
2. Lenguaje ensamblador (mnemotécnicos)
3. Lenguajes estructurados y la programación modular (lenguaje C, Pascal, Basic...)
4. Lenguajes orientados a objetos (Java, C#,...)
- ...



Evolución de los paradigmas de programación

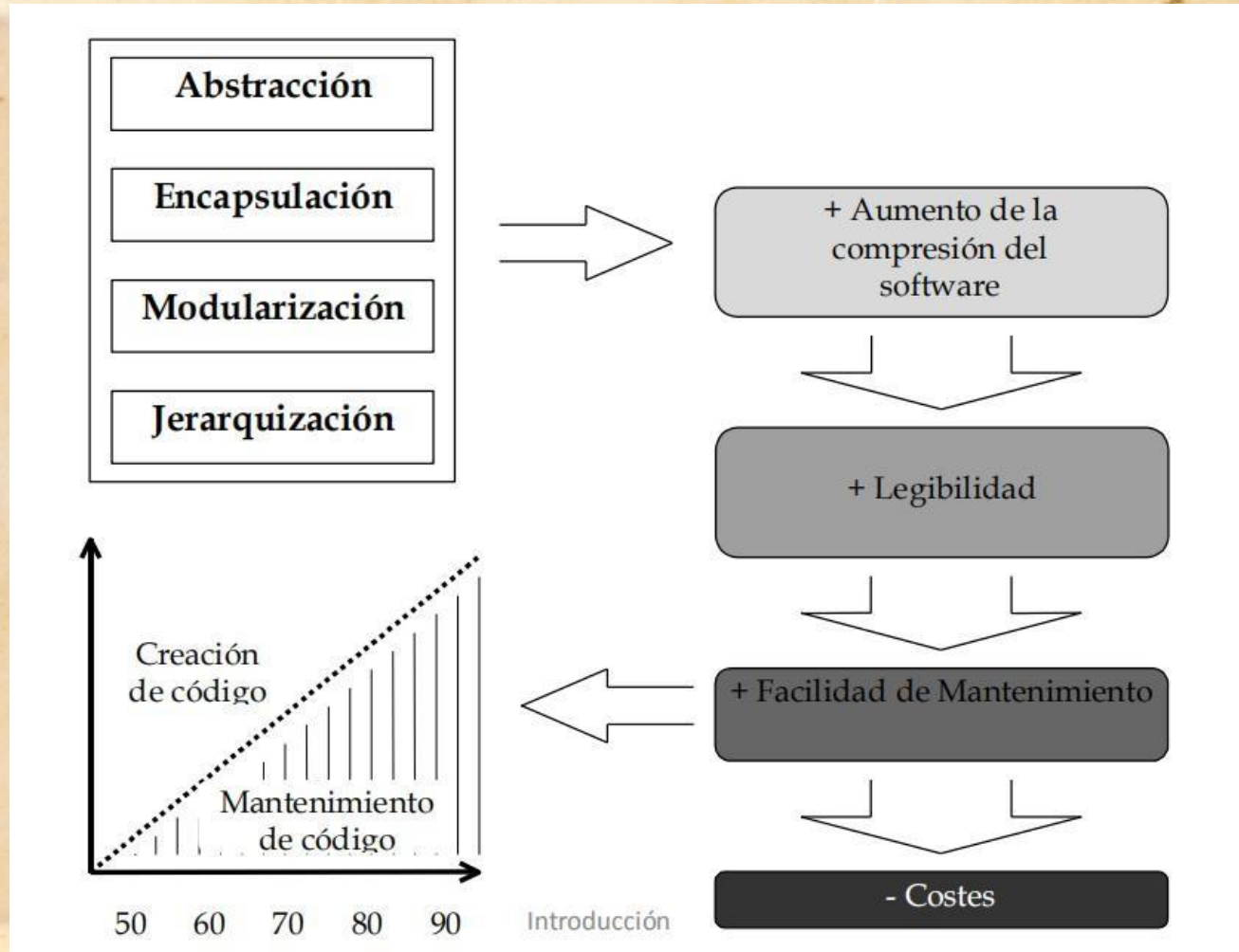


Evolución de los paradigmas de programación

Objetivos de la POO

- 1. El incremento de la abstracción, encapsulación, modularidad y jerarquización** aumenta la comprensión, la escalabilidad y la flexibilidad del software.
- 2 . El incremento de la comprensión, escalabilidad y flexibilidad del software** reduce los costes del mantenimiento del software (correctivo, adaptativo y perfectivo)
- 3 . La reducción de los costes del manteniendoreduce** los costes del desarrollo del software

Evolución de los paradigmas de programación



Programa orientado a objetos

- Simulación de un escenario del mundo real, en el que un conjunto de elementos (los objetos) interactúan entre ellos para llevar a cabo una tarea que queremos resolver.

OBJETO: Cámara digital



CARACTERÍSTICAS:

- El sensor de imagen
- El monitor LCD

FUNCIONES:

- Fotografiar una imagen

Elementos de la POO

Clase : descripción de **los datos y las operaciones** que describen el comportamiento de un cierto conjunto de elementos homogéneos.

También podemos considerar una clase como un **tipos de dato** definido por el programador, donde se agrupa información (datos y funciones)

Ejemplo: Clase Persona

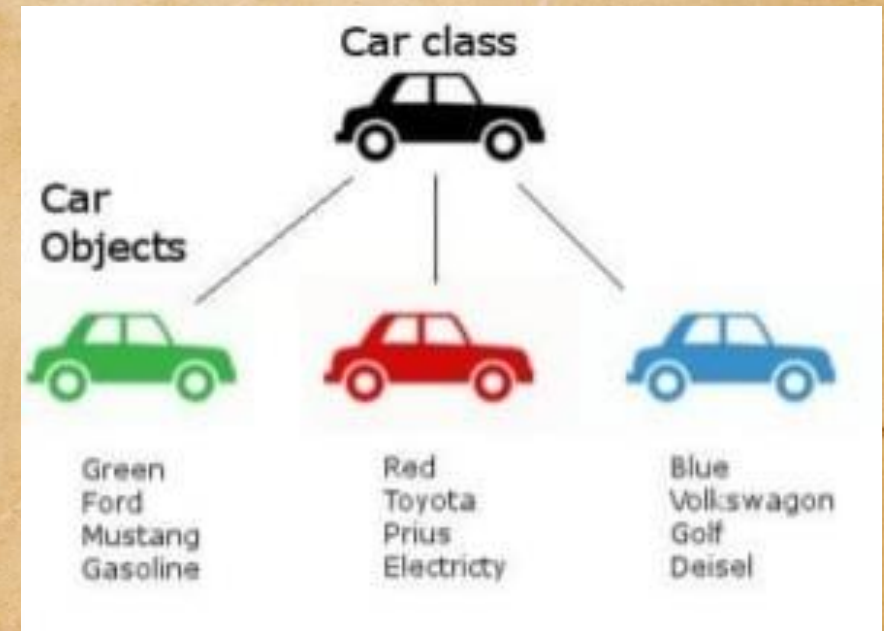
- datos: edad, nombre, sexo
- operaciones: saltar, correr, andar, etc...

Elementos de la POO

Objeto: ejemplar concreto (instancia) de una clase, que responde al comportamiento definido por las operaciones de la clase a la que pertenece, adecuándose el estado a sus datos particulares.

Ejemplo: Objetos de la clase
Persona

- un cliente concreto de un gimnasio
- un paciente concreto de un hospital



Elementos de la POO

Atributo (*): cada uno de los datos de una clase, y por tanto, presente en todos los objetos de esa clase. Una clase puede tener cualquier número de atributos, o no tener ninguna.

Ej. Atributos de la clase Persona

- edad
- sexo
- nombre

()NOTA: Muchas veces veremos que los atributos también se llaman "propiedades". Mejor hablar directamente de atributos ya que se puede confundir con conceptos que veremos más adelante cuando hablamos de "entidades" y donde se diferencia el acceso directo al atributo (pasa a llamarse "campo" o "field") y el acceso mediante métodos get/set (donde lo llamaremos "propiedad" o "property")*

Elementos de la POO

Estado: conjunto de los valores de los atributos que tiene un objeto particular en un instante concreto.

Ej. Estados de dos objetos distintos de la clase `Persona`

- Juan de 18 años y sexo hombre
- María de 45 años y sexo mujer

Si en vez de crear un nuevo objeto de la clase "Persona" lo que hacemos es modificar la edad del objeto "Joan" diremos que el objeto ha cambiado de estado.

Elementos de la POO

Método: definición de una operación o comportamiento de una clase.

Ejemplo: Métodos de la clase Persona

- **saltar:** flexionar las piernas y tomar impulsos hacia arriba
- **comida:** coger el alimento y llevarlo a la boca

Elementos de la POO

Mensaje: invocación de un método sobre un objeto. Un objeto es el agente activo que lanza el mensaje, y otro objeto es el agente pasivo que recibe el mensaje. El objeto receptor del mensaje debe contemplar esta operación entre las definidas en su clase.

Ejemplo: Mensajes a objetos de la ClasePersona

clienteGimnasio.saltar()

pacienteHospital.respirar(2);

Bases de la orientación a objetos

1. **Todo es un objeto**, con una entidad propia.
2. Un programa es un **conjunto de objetos que interactúan** entre ellos.
3. Un objeto puede estar **formado por objetos más simples**.
4. Cada objeto **pertenece a un tipo** concreto (una clase).
5. Los objetos del mismo tipo (de la misma clase) tienen un **comportamiento idéntico**.



El juego del Monopoly. Descripción (I)



- Los jugadores gestionan bienes inmuebles con el objetivo de arruinar al resto de jugadores.
- Cada jugador dispone de un dinero inicial, a forma de billetes y es identificado por una ficha del tablero.
- Esta ficha avanza de acuerdo con el resultado del lanzamiento de dos dados.

El juego del Monopoly. Descripción (II)



- Cada vez que se cae a una casilla, el jugador puede optar por comprarla pagando el precio que marca la casilla. Si lo hace, recibe un título de propiedad.
- Cuando un jugador cae en una casilla que es propiedad de otro jugador, paga al jugador una cantidad (especificada en el título).
- Hay un jugador que gestiona el dinero llamado „banca“ que no tiene ficha (gestiona billetes y propiedades no asignadas).

El juego del Monopoly. Descripción (III)



- Cuando un jugador tiene todas las casillas del mismo color puede edificar hasta cuatro casas (se representan con piezas) y después un hotel (pagando el dinero correspondiente).
- La descripción es más amplia (robar cartas, casilla de prisión, etc...).
- Vamos a aplicar las bases de la orientación a objetos.

Todo es un objeto, con identidad propia

- Aplicar la orientación a objetos en un programa es equivalente a intentar crear la simulación de un escenario del mundo real, pero dentro del ordenador.
- Estructuramos con un conjunto de **elementos**, cada uno de los cuales tiene unas **propiedades** y un **comportamiento** concreto que intente imitar al elemento del mundo real que representa.

Todo es un objeto, con identidad propia

- Los **elementos** serán los objetos
- Las **propiedades** son las cualidades que son importantes de cuantificar y definen el aspecto o estado del objeto. Formalmente se llaman **atributos**

**Cada objeto dentro de cada programa es ÚNICO,
aunque se puedan crear más objetos con idénticas
propiedades**

Todo es un objeto, con identidad propia (Monopoly)

- Cada objeto se corresponde con un elemento que interviene en la partida: tablero, dados, jugador, fichas que se mueven por el tablero, los títulos de la propiedad, las tarjetas, las casas hoteleras, los billetes...
- Algunos son individuales: tablero
- Otros múltiples con atributos idénticos (billete de 100) o varios (cada título de propiedad).



JugadorN

etc...

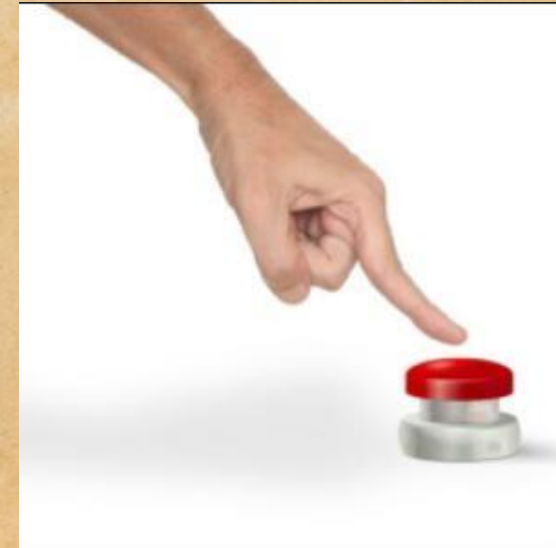
Todo es un objeto, con identidad propia (Monopoly)

Billete1	
color	naranja
valor	500

Un programa es un conjunto de objetos que interactúan

- La ejecución del programa vendrá dada por el conjunto de interacciones entre los objetos que lo componen.

Ejemplo: **Pulsar un botón** (acción del objeto) y nterac tendrá con otros objetos, transmitiendo el orden de la acción que se desea hacer.

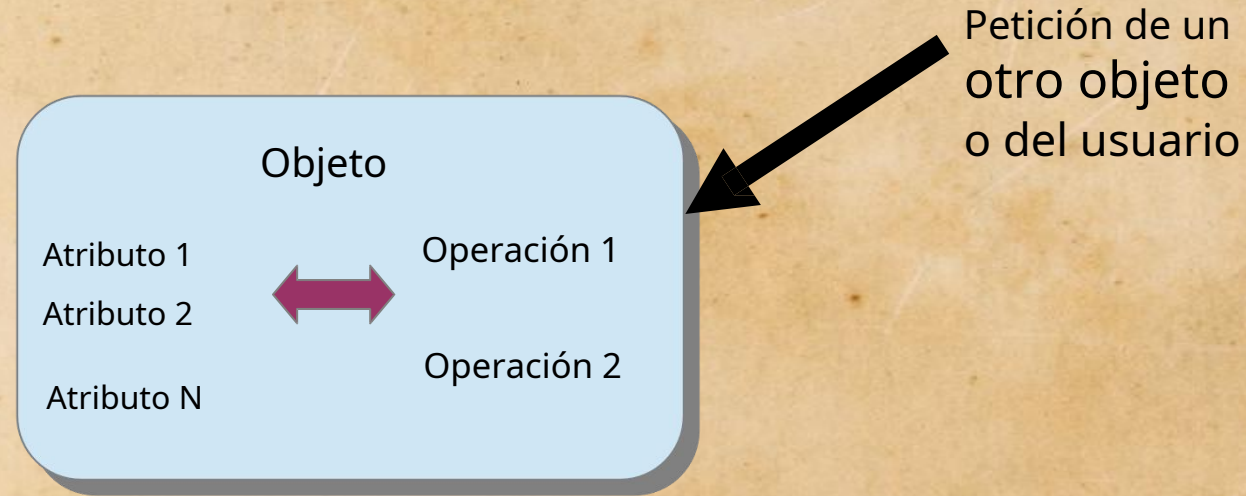



Un programa es un conjunto de objetos que interactúan

- Lo que define el comportamiento de cada objeto es la **lista de las posibles interacciones** que puede recibir.
- Cada interacción estará relacionada con una **tarea** que puede realizar **un cambio de estado**
 - Apagar o encender una luz
 - Haga clic en un botón
 - Cerrar una ventana
 - Etc....

Un programa es un conjunto de objetos que interactúan

- A cada interacción que un objeto puede recibir le llamamos **operación**





Un programa es un conjunto de objetos que interactúan (Monopoly)

- Algunas interacciones que pueden ser iniciadas por un objeto **Jugador**
 - Pagar una deuda a otro jugador(invocamos el método "pagar" de un objeto "jugador")
 - Tirar los dados(invocamos el método "tirar" de los dos objetos "dado")
 - Comprar propiedad en la banca(invocamos el método "comprar" del objeto "Banca")

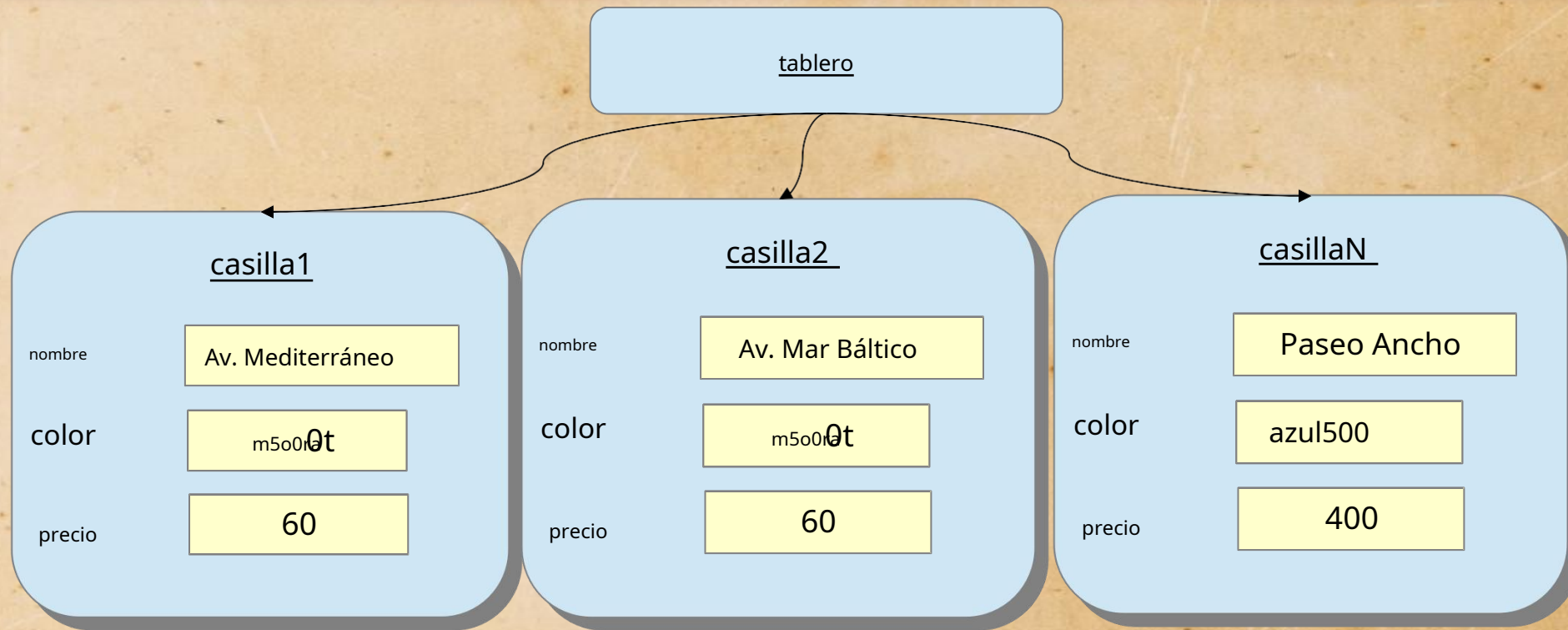
Un objeto puede componerse de otros objetos más simples

- Misma filosofía que la **descomposición modular**

Ejemplo: Motor de un coche → Alternador, Colector de admisión, Líquido de dirección...

- Es posible interactuar directamente con el objeto complejo (Motor) como con los subelementos (Alternador, Colector, etc...)

Un objeto puede componerse de otros objetos más simples (Monopoly)



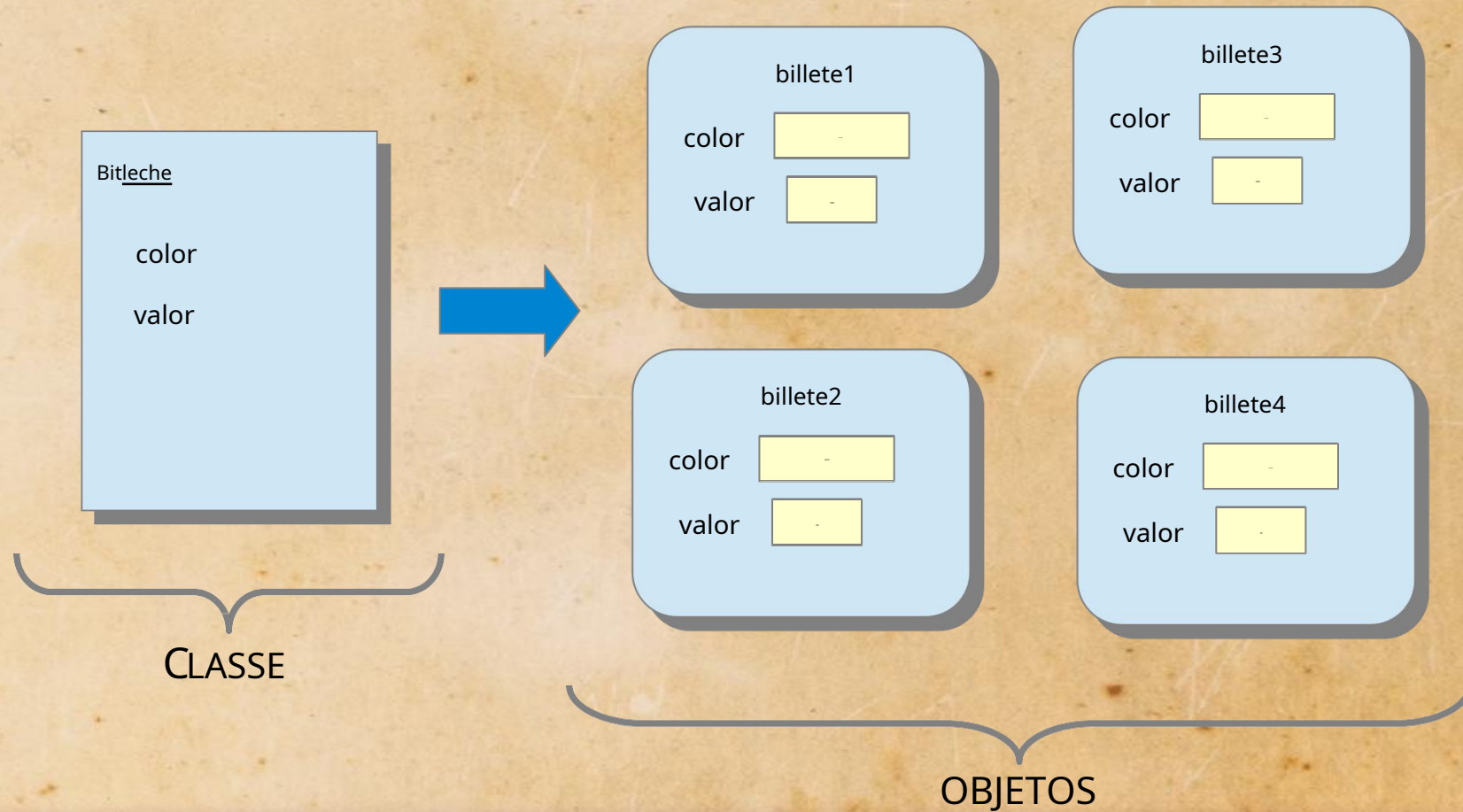
Cada objeto pertenece a un tipo concreto: una clase

- A medida que se van de finalizando los objetos que componen el programa en ejecución, algunos de ellos serán del mismo tipo (clase):
 - Tienen las mismas propiedades, aunque cada uno tendrá sus valores concretos, y lo mismo comportamiento.

Cada objeto pertenece a un tipo concreto: una clase

- Una **clase** es la especificación formal de las **propiedades** (atributos) y el **comportamiento** esperado (la lista de operaciones) de un conjunto de objetos del mismo tipo, y que actúa como una plantilla para generar cada uno de ellos.
- Formalmente se dice que un objeto es una **INSTANCIA** de una clase y que un objeto es **instanciado** cuando se crea dentro de la aplicación.

Cada objeto pertenece a un tipo concreto: una clase (Monopoly)



Los objetos del mismo tipo tienen un comportamiento idéntico

- Esta afirmación puede considerarse una extensión de todo lo que hemos visto anteriormente.
- Una clase define los atributos de objetos del mismo tipo así como el comportamiento (su lista de operaciones).
- Una **operación**, por tanto, una función o transformación que se puede aplicar a cualquier objeto de una clase.

Relación formal de los elementos de la POO

- Una **clase** es la definición de los atributos y métodos que describen el comportamiento de un cierto conjunto de objetos homogéneos.
- Un **objeto** es un ejemplar concreto de una clase que responde a los mensajes correspondientes a los métodos de ésta, adecuándose al estado de sus atributos.

Relación de los elementos de la POO

- Las clases asumen el principio de encapsulación: cuando se describe una clase, debe describirse tanto su vista pública o interfaz como su vista privada o implantación.
- **La vista pública o interfaz** describe a qué operaciones responden los objetos de esa clase, o sea, su comportamiento.
- **La vista privada o implantación** describe las estructuras de datos de la clase y cómo manipulan las operaciones los datos. De esa forma, se conjuga la abstracción inherente a la clase con la encapsulación de sus datos y de su forma de operar