Programació

7.1 Introducció a la POO

Introducció

- Un **paradigma** de programació és un estil de desenvolupament de programes. És a dir, un model per a resoldre problemes computacionals.

- Fins ara hem utilitzat el **paradigma de programació estructurada**, que empra estructures de control (condicionals i bucles) fent ús de dades i funcions de manera independent. Un dels principals desavantatges d'eixe paradigma és que no existeix un **vincle fort entre funcions i dades**, la qual cosa pot dificultar problemes més complexos. Arribats a este punt saltarem a un nou paradigma, la **programació orientada a objectes** (POO) que ens dotarà de noves ferramentes que faciliten la resolució de problemes més complexos.

Introducció

- Una de les dificultats permanents serà **decidir com traduir el problema que es té que resoldre** en estructures de dades i com interactuen entre elles.
  - Per a problemes senzills: **cada dada → tipus de dada primitiu o basat directament en primitius** (arrays, Strings...)
  - Per a problemes complexos: metodologia que decidisca **com agrupar estes dades**

Bases de la programació

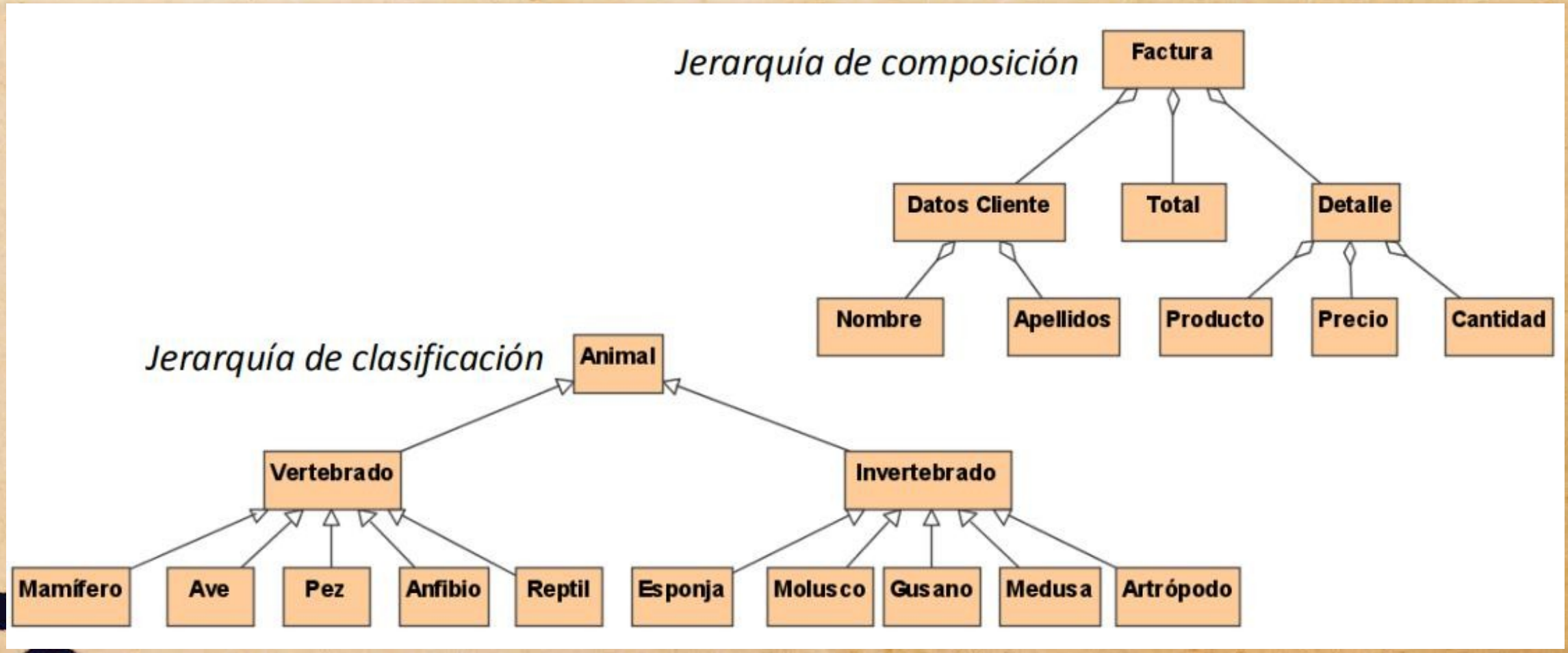
- **Abstracció**: procés mental d'extracció de les característiques essencials d'alguna cosa, ignorant els detalls superflus.

- **Modularització**: procés de descomposició d'un sistema en un conjunt de mòduls poc acoblats (independents) i cohesius (amb significat propi).

Bases de la programació

- **Jerarquització**: procés d'estructuració pel que es produeix una organització (jerarquia) d'un conjunt d'elements en graus o nivells de responsabilitat, de incumbència o de composició, entre d'altres. (Interrelació entre les distintes abstraccions).
- **Encapsulació**: és l'empaquetament d'informació, es a dir dades (propietats) i mètodes (comportament) en un sol component. De tal manera que el programador només ha de coneixer com invocar els mètodes sense saber com estan implementats.

Bases de la programació

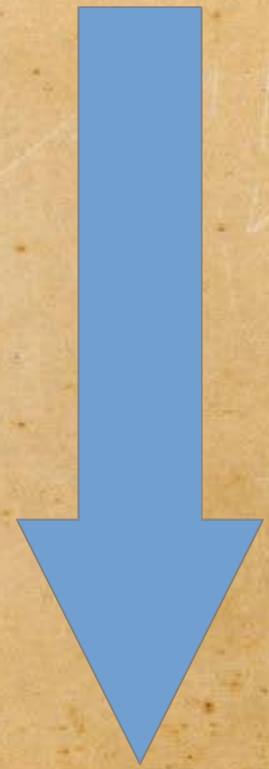


Evolució dels paradigmes de programació

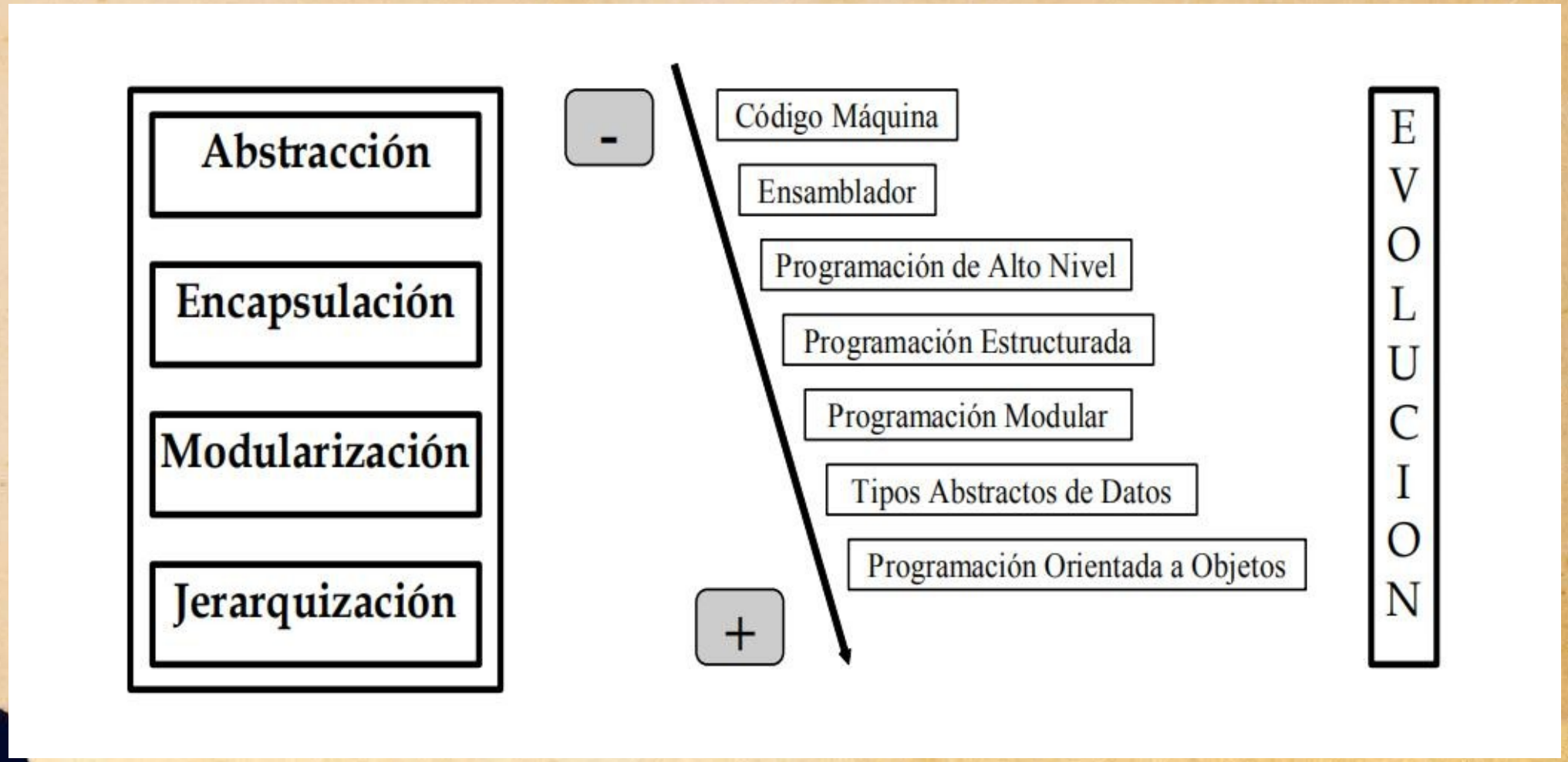
1. Codi màquina (cadenes de 0's i 1's)
2. Llenguatge assemblador (mnemotècnics)
3. Llenguatges estructurats i la programació modular (llenguatge C, Pascal, Basic...)
4. Llenguatges orientats a objectes

(Java, C#,...)

...



Evolució dels paradigmes de programació

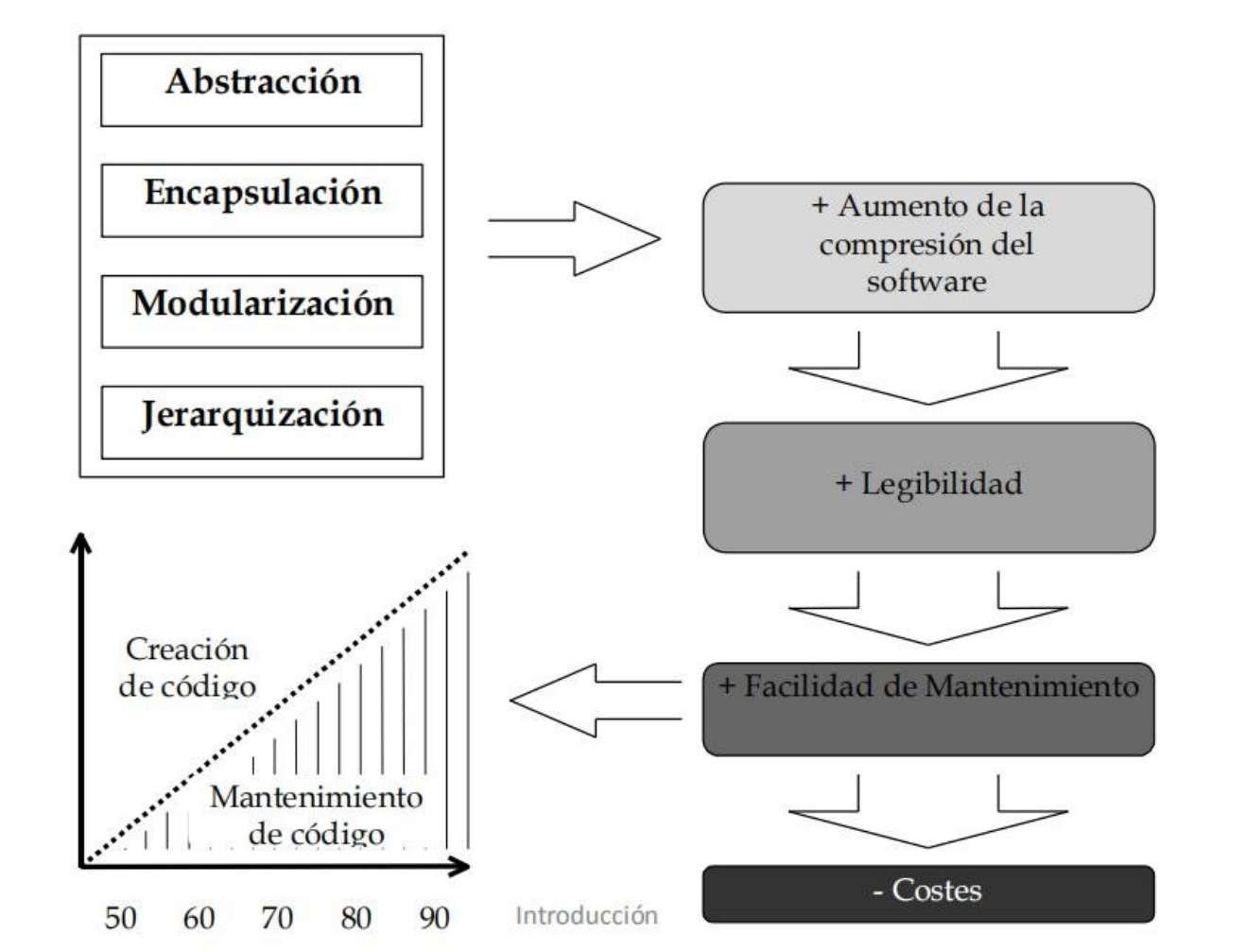


Evolució de els paradigmes de programació

Objectius de la POO

1. **L'increment de l'abstracció, encapsulació, modularitat i jerarquització** augmenta la comprensió, l'escalabilitat i la flexibilitat del programari.
2. **L'increment de la comprensió, escalabilitat i flexibilitat del programari** redueix els costos del manteniment del programari (correctiu, adaptatiu i perfectiu)
3. La **reducció dels costos del mantenint** redueix els costos del desenvolupament del programari

Evolució de els paradigmes de programació



Programa orientat a objectes

- Simulació d'un escenari del món real, en que un conjunt d'elements (els objectes) interactuen entre ells per dur a terme una tasca que volem resoldre.

OBJECTE: Càmera digital

CARACTERÍSTIQUES:

- El sensor d'imatge

- El monitor LCD

FUNCIONS:

- Fotografiar una imatge

Elements de la POO

Classe: descripció de **les dades i les operacions** que descriuen el comportament d'un cert conjunt d'elements homogenis.

També podem considerar una classe com un **tipus de dada** definit pel programador, on s'agrupa informació (dades i funcions)

Exemple: Classe `Persona`

- dades: edat, nom, sexe
- operacions: saltar, córrer, caminar, etc...

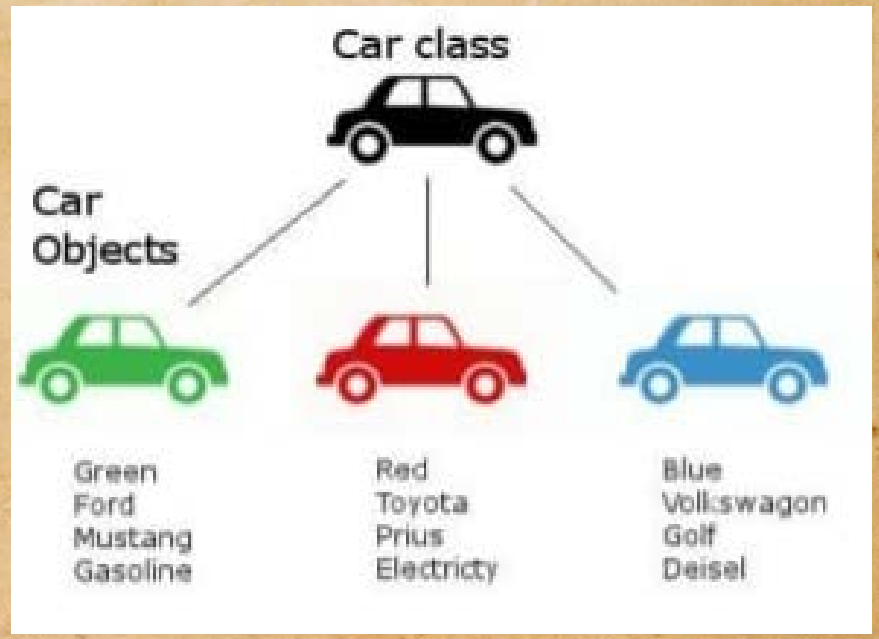
Elements de la POO

Objecte: exemplar concret (instància) d'una classe, que respon al comportament definit per les operacions de la classe a la que pertany, adequant-se l'estat a les seues dades particulars.

Exemple: Objectes de la clase `Persona`

- un client concret d'un gimnàs

- un pacient concret d'un hospital



Elements de la POO

Atribut (*): cadascuna de les dades d'una classe, i per tant, present a tots els objectes d'eixa clase. Una classe pot tindre qualsevol nombre d'atributs, o no tindre'n cap.

Ex. Atributs de la classe `Persona`

- `edat`
- `sexe`
- `nom`

(*)NOTA: Moltes voltes vorem que els atributs també s'anomenen "propietats". Millor parlar directament d'atributs ja que es pot confondre amb conceptes que vorem més endavant quan parlem "d'entitats" i on es diferencia l'accés directe a l'atribut (passa a anomenar-se "camp" o "field") i l'accés mitjançant mètodes get/set (on l'anomenarem "propietat" o "property")

Elements de la POO

Estat: conjunt dels valors dels atributs que té un objecte particular en un instant concret.

Ex. Estats de dos objectes distints de la classe `Persona`

- `Joan de 18 anys i sexe home`
- `Maria de 45 anys i sexe dona`

Si en comptes de crear un nou objecte de la classe "Persona" el que fem és modificar l'edat de l'objecte « Joan » direm que l'objecte ha canviat d'estat.

Elements de la POO

Mètode: definició d'una operació o comportament d'una classe.

Exemple: Mètodes de la classe `Persona`

- **`saltar`**: flexionar les cames i prendre impulsos cap amunt
- **`menjar`**: agafar l'aliment i portar-lo a la boca

Elements de la POO

Missatge: invocació d'un mètode sobre un objecte. Un objecte és l'agent actiu que llança el missatge, i un altre objecte és el agent passiu que rep el missatge. El objecte receptor del missatge deu contemplar esta operació entre les definides a la seua classe.

Exemple: Missatges a objectes de la Classe `Persona`

```
clientGimnàs.saltar()

pacientHospital.respirar(2);
```

Bases de l'orientació a objectes

1. **Tot és un objecte**, amb una entitat pròpia.
2. Un programa és un **conjunt d'objectes que interactuen** entre ells.
3. Un objecte pot estar **format per objectes més simples**.
4. Cada objecte **pertany a un tipus** concret (una classe).
5. Els objectes del mateix tipus (de la mateixa classe) tenen un **comportament idèntic**.

El joc del Monopoly. Descripció (I)

- Els jugadors gestionen béns immobles amb l'objectiu d'arruïnar al resta de jugadors.
- Cada jugador disposa d'uns diners inicials, a forma de bitllets i és identificat per una fitxa del tauler.
- Esta fitxa avança d'acord amb el resultat del llançament de dos daus.

El joc del Monopoly. Descripció (II)

- Cada vegada que es cau a una casella, el jugador pot optar per comprar-la pagant el preu que marca la casella. Si ho fa, rep un títol de propietat.
- Quan un jugador cau a una casella que és propietat d'un altre jugador, paga al jugador una quantitat (especificada al títol).
- Hi ha un jugador que gestiona el diners anomenat „banca“ que no té fitxa (gestiona bitllets i les propietats no assignades).

El joc del Monopoly. Descripció (III)

- Quan un jugador té totes les caselles del mateix color pot edificar fins quatre cases (es representen amb peces) i després un hotel (pagant el diners corresponents).
- La descripció és més àmplia (robar cartes, la casella de la presó, etc...).
- Anem a aplicar les bases de la orientació a objectes.

Tot és un objecte, amb identitat pròpia

- Aplicar la orientació a objectes a un programa és equivalent a intentar crear la simulació de un escenari del món real, però dins del ordinador.
- Estructurem amb un conjunt d'**elements,** cada un dels quals té unes **propietats** i un **comportament** concret que intente imitar a l'element del món real que representa.

Tot és un objecte, amb identitat pròpia

- Els **elements** seran els objectes
- Les **propietats** són les qualitats que són importants de quantificar, i que defineixen l'aspecte o estat del objecte. Formalment s'anomenen **atributs**

Cada objecte dins de cada programa és ÚNIC, encara que es puguen crear més objectes amb propietats idèntiques

Tot és un objecte, amb identitat pròpia (Monopoly)

- Cada objecte es correspon amb un element que intervé a la partida: tauler, daus, jugador, fitxes que es mouen pel tauler, els títols de la propietat, les targetes, les cases i hotels, els bitllets...
- Alguns són individuals: tauler
- Altres múltiples amb atributs idèntics (bitllet de 100) o diversos (cada títol de propietat).

Tot és un objecte, amb identitat pròpia (Monopoly)

Fitxa1
FitxaN

Dau1
Dau2

Bitllet1
Bitllet2

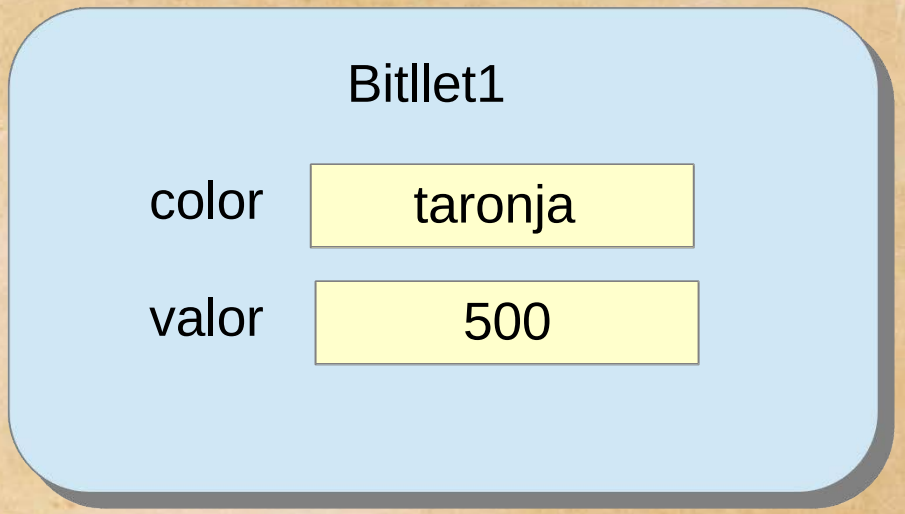
Títol1
TítolN

Casa1
CasaN

Jugador1
JugadorN

etc….

Tot és un objecte, amb identitat pròpia (Monopoly)



Un programa és un conjunt d'objectes que interactuen

- L'execució del programa vindrà donat pel conjunt d'interaccions entre els objectes que el componen.

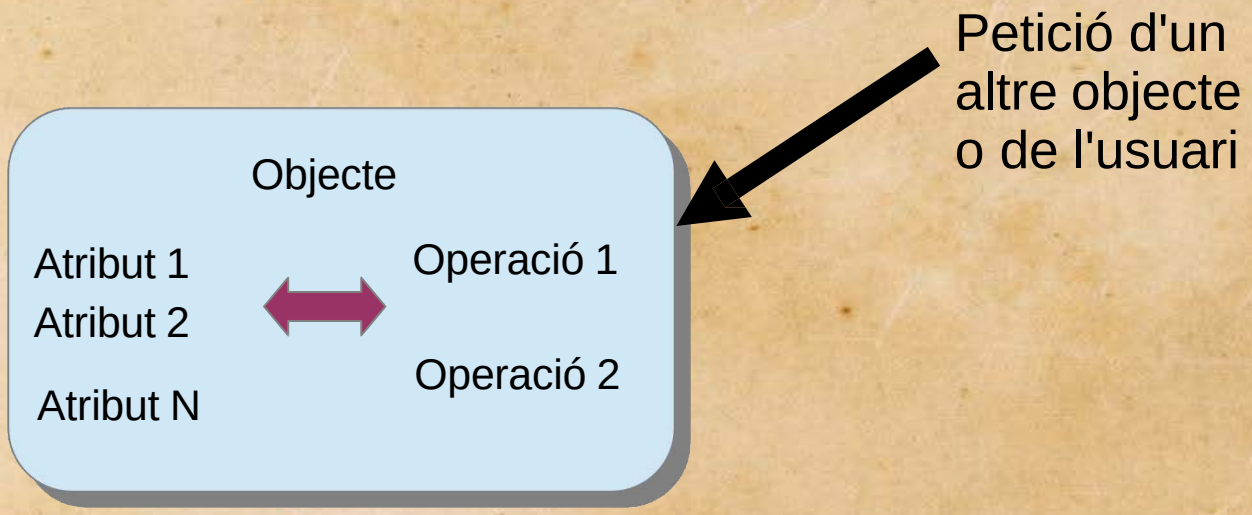
Exemple: **Prémer un botó** (acció del objecte) interactuarà amb altres objectes, transmetent la ordre de l'acció que es desitja fer.

Un programa és un conjunt d'objectes que interactuen

- El que defineix el comportament de cada objecte és la **llista de les possibles interaccions** que pot rebre.
- Cada interacció estarà relacionada amb una **tasca** que pot realitzar **o un canvi de estat**
  - Apagar o encendre una llum
  - Cliqueu un botó
  - Tancar una finestra
  - Etc....

Un programa és un conjunt d'objectes que interactuen

- A cada interacció que un objecte pot rebre l'anomenem **operació**



Un programa és un conjunt d'objectes que interactuen (Monopoly)

- Algunes interaccions que poden ser iniciades per un objecte **Jugador**
  - Pagar un deute a un altre jugador (invoquem el mètode “pagar” d’un objecte “jugador”)
  - Llençar els daus (invoquem el mètode “tirar” dels dos objectes “dau”)
  - Comprar propietat a la banca (invoquem el mètode “comprar” de l'objecte “Banca”)

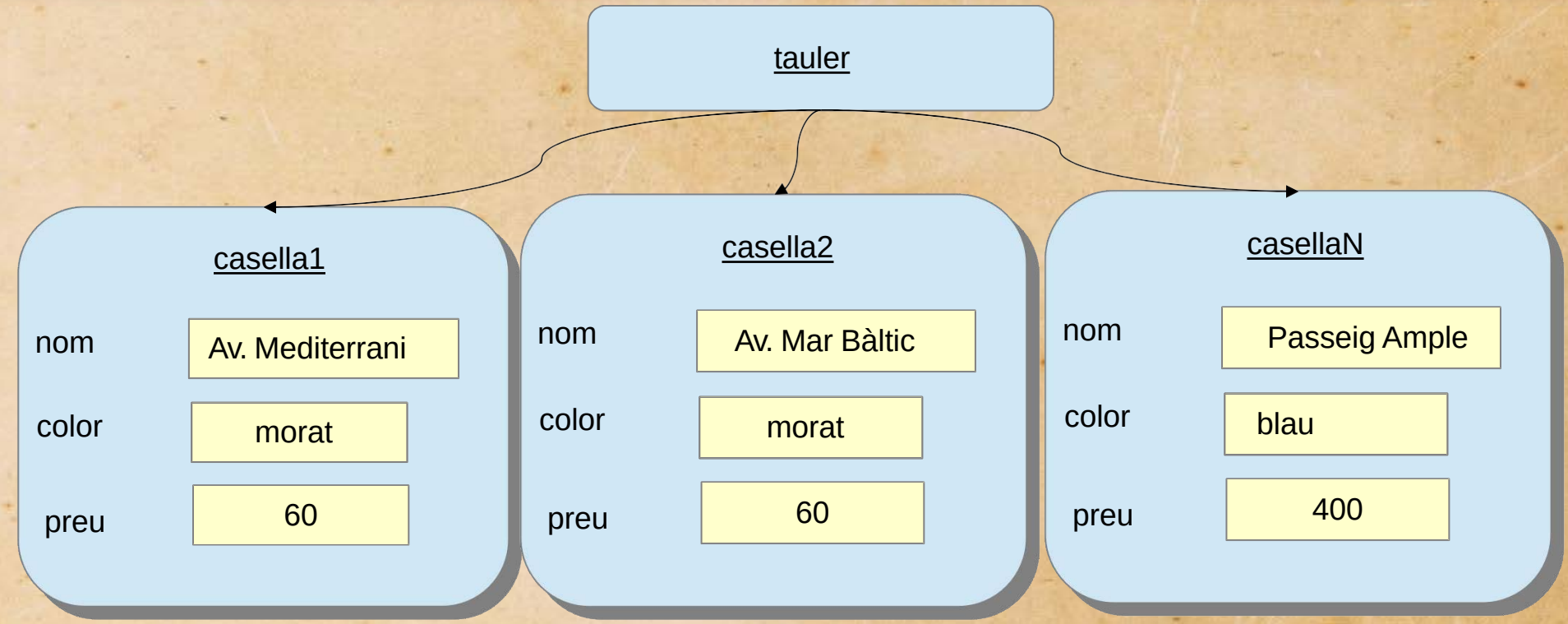
Un objecte es pot compondre d'altres objectes més simples

- Mateixa filosofia que la **descomposició modular**

Exemple: Motor d'un cotxe → Alternador, Col·lector d'admissió, Líquid de direcció...

- És possible interactuar tant directament amb l'objecte complex (Motor) com amb els subelements (Alternador, Col·lector, etc...)

Un objecte es pot compondre d'altres objectes més simples (Monopoly)



Cada objecte pertany a un tipus concret: una classe

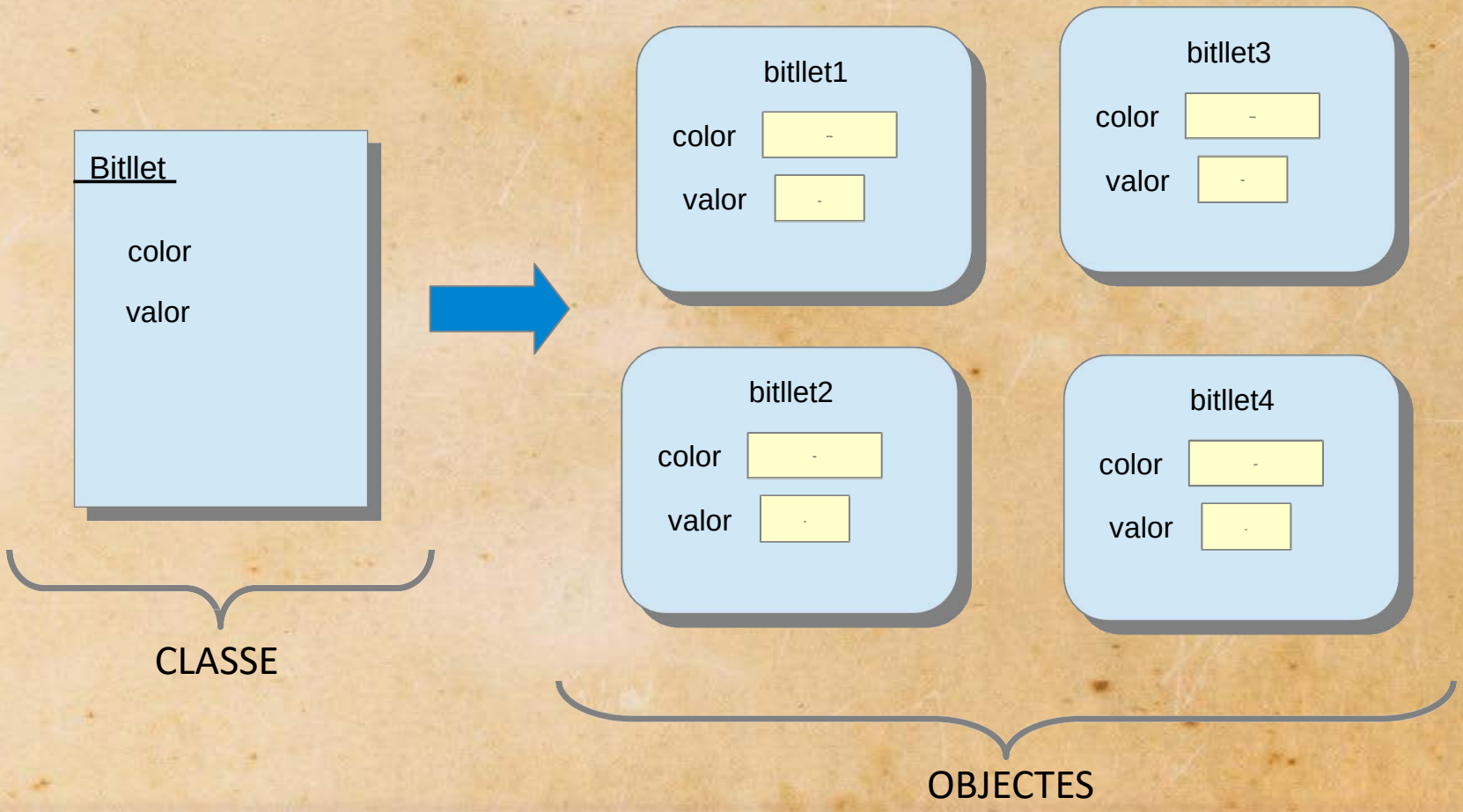
- A mesura que es van definint els objectes que componen el programa en execució, alguns d'ells seran del mateix tipus (classe):
  - Tenen les mateixes propietats, encara que cadascun tindrà els seus valors concrets, i el mateix comportament.

Cada objecte pertany a un tipus concret: una classe

- Una **clase** és l'especificació formal de les **propietats** (atributs) i el **comportament** esperat (la llista d'operacions) d' un conjunt d'objectes del mateix tipus, i que actua com una plantilla per a generar cada un d'ells.

- Formalment es diu que un objecte és una **INSTÀNCIA** d'una classe i que un objecte **és instanciat** quan es crea dins de l'aplicació.

Cada objecte pertany a un tipus concret: una classe (Monopoly)



Els objectes del mateix tipus tenen un comportament idèntic

- Esta afirmació es pot considerar una extensió de tot el que hem vist anteriorment.
- Una classe defineix els atributs d'objectes del mateix tipus així com el comportament (el seu llista de operacions).
- Una **operació** és, per tant, una funció o transformació que es pot aplicar a qualsevol objecte d'una classe.

Relació formal dels elements de la POO

- Una **classe** és la definició dels atributs i mètodes que descriuen el comportament d'un cert conjunt de objectes homogenis.
- Un **objecte** és un exemplar concret d'una classe que respon als missatges corresponents a els mètodes d'aquesta, adequant-se a l'estat dels seus atributs.

Relació de els elements de la POO

- Les classes assumeixen el principi d'encapsulació: quan es descriu una classe, es deu descriure tant la seua vista pública o interfície com la seua vista privada o implantació.
- La **vista pública o interfície** descriu a quines operacions responen els objectes d'eixa classe, o siga, el seu comportament.
- La **vista privada o implantació** descriu les estructures de dades de la classe i com manipulen les operacions els dades. D'eixa forma, es conjuga l'abstracció inherent a la classe amb l'encapsulació de les seues dades i de la seua forma d'operar