

Programació



UT6.3. Polimorfisme

Definició

- Terme d'origen grec que significa: **moltes formes**
 - **Moltes formes de pagar** (targeta de crèdit, paypal, bizum, efectiu...)
 - **Venda de bitllets per finestreta** (una persona) o a través d'una màquina
 - **Una impressora pot imprimir** a través de diferents drivers d'impressores
 - **Un navegador mostra continguts de diferent origen** (text, imatges, vídeo...)

Limitacions a la POO

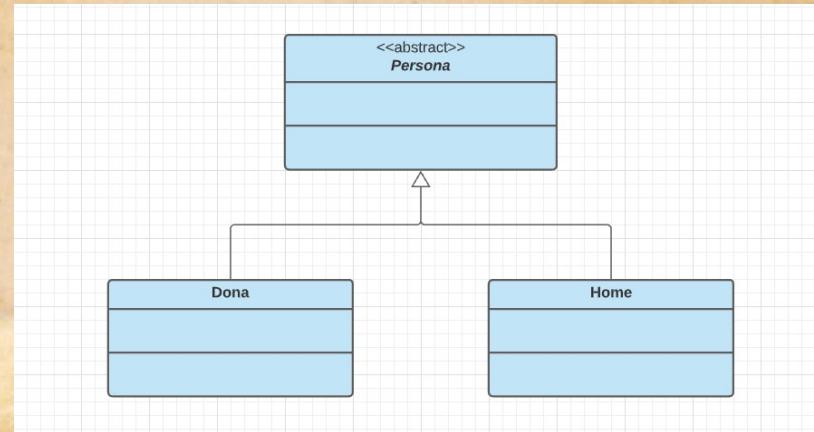
- En POO no parlem de que:
 - “alguna cosa canvia de forma”
 - “alguna cosa siga dos coses a la vegada”
- Una persona no es converteix en màquina expenedora de bitllets
- Una persona no és a la vegada una màquina expenedora de bitllets
- Un bitllet, per tant, pot ser venut per una màquina o una persona en funció del moment en què es venga

Definició formal

- És una **relaxació del sistema de tipus** (el sistema que indica el rang de literals que admet cada variable) de tal manera que una referència a una classe **accepta adreces d'objectes de la pròpia classe i de les seues classes derivades** (filles, netes...).
- Per a que existisca polimorfisme, és **imprescindible que hi haja una jerarquia de classificació**.
- Encara que una jerarquia de classificació **no obliga** a realitzar tractaments polimòrfics.

Ús de classes abstractes

- Amb la incorporació del polimorfisme, té sentit declarar **referències a classes abstractes** amb la intenció d'emmagatzemar **adreses d'objectes de classes concretes derivades**, NO de classes abstractes (les abstractes **NO són instanciables**)
- **Exemple:** Al llarg de la història en este món existeix la jerarquia de persones (dones i homes). Però en certs moments i en certs territoris no existeix polimorfisme (el lloc d'una dona no el pot ocupar un home i el lloc d'un home no el pot ocupar una dona) mentre que en altres moments o altres territoris el lloc d'una persona es indiferentment ocupat per una dona o per un home.



Exemple

```
Dona dona = new Dona();
Dona donaIncorrecta = new Home(); // ERROR 1
Dona donaPitjor = new Persona(); // ERROR 2
Home home = new Home();
Home homeIncorrecte = new Dona(); // ERROR 3
Home homePitjor = new Persona(); // ERROR 4
Persona persona1 = new Dona(); // POLIMORFISME
Persona persona2 = new Home(); // POLIMORFISME
Persona personaMal = new Persona(); // ERROR 5|
```

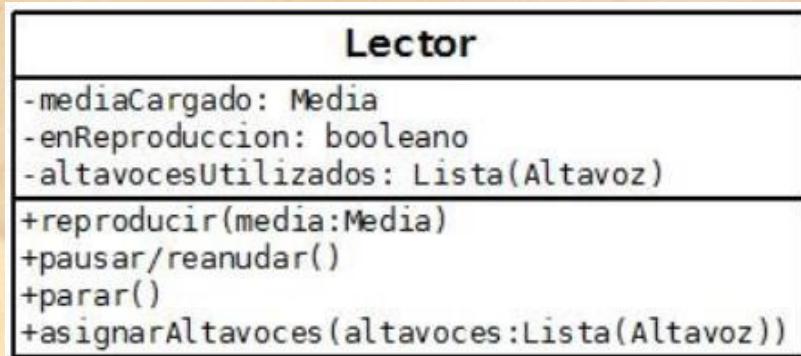
No es contempla que una dona es convertisca en home o un home en dona; ni es contempla que algú siga dona i home a la volta. El que es contempla és que una referencia a persona apunte a un objecte de la classe dona o de la classe home.

Efectes de l'herència

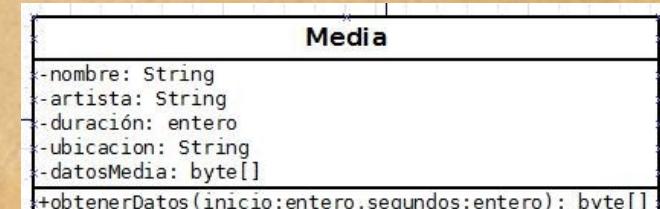
Donada una classe específica que hereta d'una altra, no pertanyerà només a ella sinó també a la seu classe pare, així com altres classes ascendents.

Exemple1: Els objectes MP3 pertanyen alhora a la classe MP3 i a la classe Media.

Exemple2: El mètode "reproducir" de la classe Lector funcionaria també amb un objecte MP3



abstracte



és un

Pèrdua d'identitat

- En el moment en que un objecte pot pertànyer a més d'una classe (pròpia i derivades), apareix la problemàtica anomenada **pèrdua d'identitat**.
- Tornant al exemple del mètode reproduir de Lector:
 - *Dins del mètode de "reproducir", el paràmetre rebut seria una referència a Media. Només es podran invocar, per tant, mètodes de la classe Media.*
- Però, quan invoquem un mètode des d'una variable de tipus Media, quin codi s'executa?

Comportament

- Quan es llança un missatge a un objecte a través d'una referència polimòrfica **s'executa el mètode prescrit a la classe de l'objecte que rep el missatge.**

Exemple:

- *Si llancem el mètode “reproducir” passant un objecte MP3 de una cançó invocarem a Lector.reproducir(mp3)*
- *Dins de “reproducir” ens interessa obtindre les dades de la cançó i invoquem a media.obtenerDatos()*
- *A pesar de que l'objecte és de la classe Media, s'obtindrien les dades de la cançó que és un MP3*

Limitació

- Quan es llança un missatge a un objecte a través d'una referencia polimòrfica, el mètode ha d'estar **contemplat a la interfície de la classe de la que es va declarar la referència** sense contemplar els possibles mètodes afegits a la classe de l'objecte apuntat.

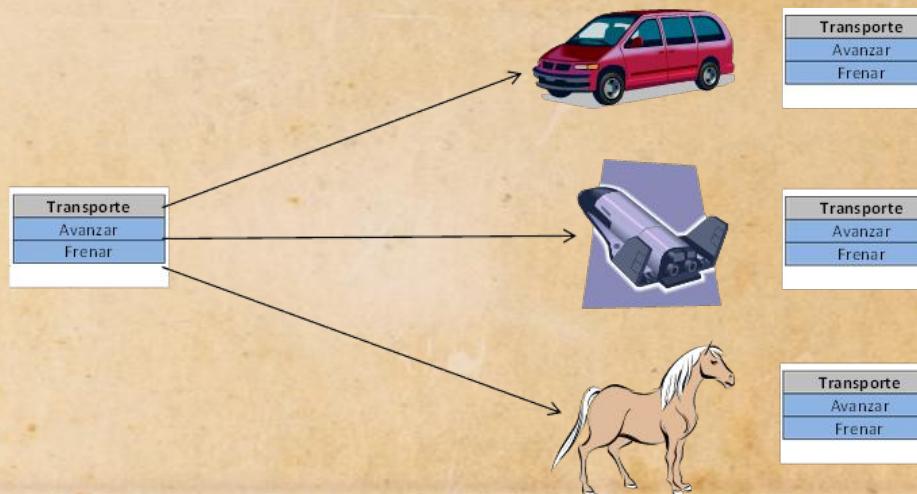
Exemple: *No tindrem accés dins de "reproducir" al mètode obtenerCalidadAudio() que és propi d'MP3*

Conclusió

- Fins ara, l'herència ha implicat especificar nous atributs i operacions a la classe derivada, i eixos membres s'afegeixen als heretats de la classe pare.
- Ara el que pretenem és modificar el comportament d'una operació prèviament definida en una classe pare. Aquí entra el joc el polimorfisme.

Una altra definició de polimorfisme

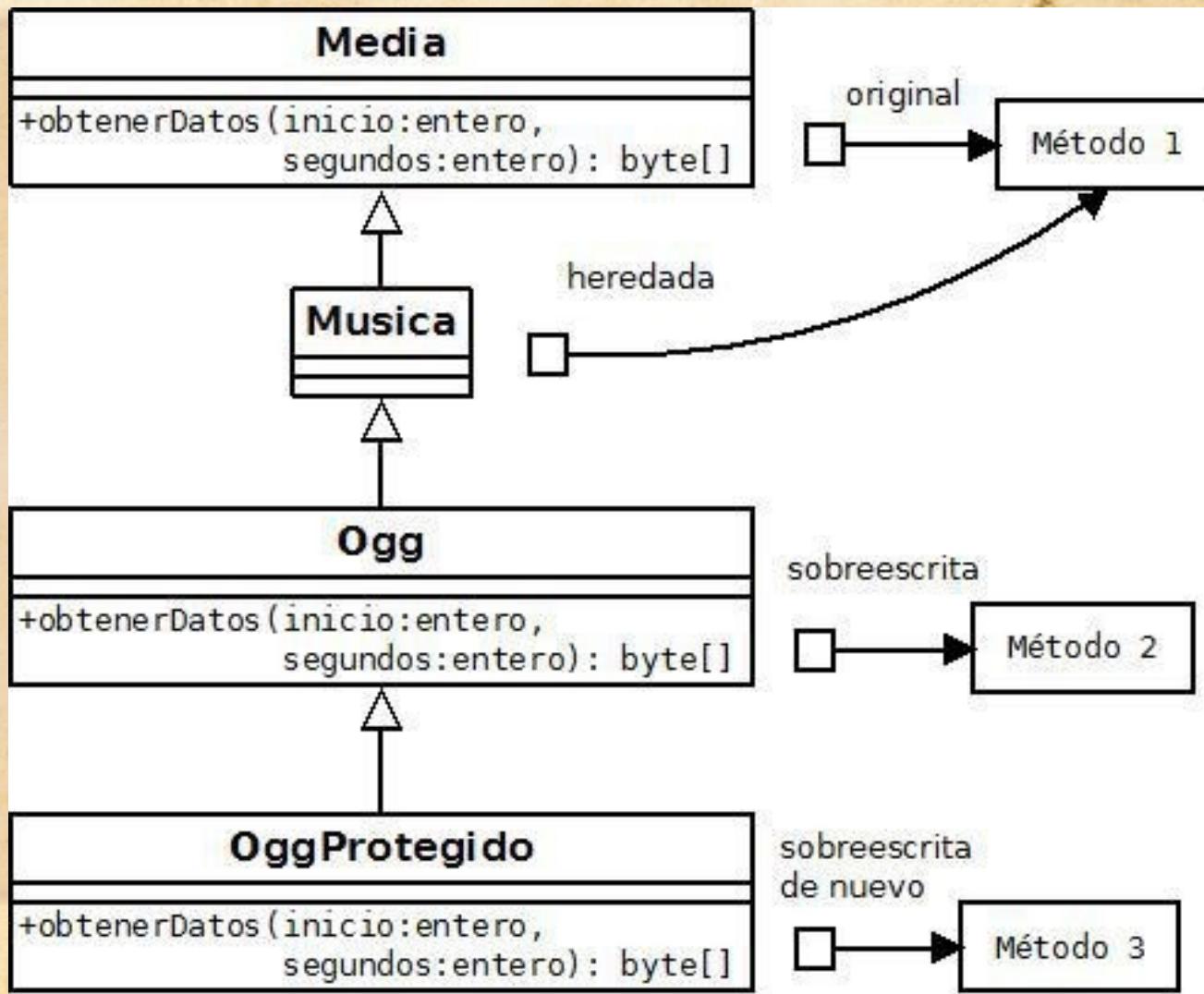
- El polimorfisme consisteix en la possibilitat d'aplicar una mateixa operació a objectes de diferents classes, però invocant a una implementació diferente segons la classe a la que pertanya l'objecte a qui passem el missatge.



Sobreescritura d'operacions

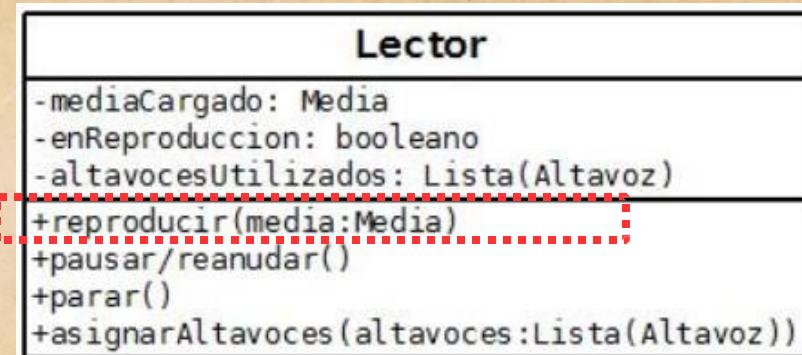
- Per aplicar el polimorfisme s'ha de fer ús de la **sobreescritura d'operacions** (override en anglès)
- Com ja vam vore, **sobreescriri** una operació vol dir tornar-la a definir en una classe subclasse, de manera que el mètode associat i, per tant, el codi que s'executa, tinga un comportament diferent al de la classe pare.

Exemple: Fitxers multimèdia



Operacions polimòrfiques

- Reprenent el mètode "reproducir" de la classe Lector:



- Recordem que dins del mètode "reproducir", s'efectuava una **pèrdua de identitat**.

Operacions polimòrfiques

- Ara bé, si dins del mètode “reproducir” s’invoca al mètode “obtenerDatos” de Media, s’executarà la implementació corresponent a l’objecte passat com a paràmetre.
- Exemple:

```
OGG medial = new OGG();  
lector.reproducir(medial);
```
- Dins de "reproducir", la trucada al mètode “obtenerDatos” executarà la implementació "Método 2" de l'esquema anterior (sobreescritura del mètode de Media).

Aplicacions del polimorfisme en classes abstractes

- Suposem que es vol donar suport a la lectura d'arxius de vídeo i donat que la lectura de dades és different, la operació `obtenerDatos` tindrà que fer tasques diferents en funció de les dades que tracte.
- Si no utilitzàrem l'herència ni el polimorfisme provocaria que es tindrien que aplicar operacions condicionals (una per a cada format a tractar) dins de la classe `Media`:
 - Poca cohesió (una mateixa classe gestiona formats de dades diferents).
 - Fa falta tornar sempre a este codi font per ampliar funcionalitats.

Aplicacions del polimorfisme en classes abstractes

- Gràcies a tractar el mètode `obtenerDatos` com a abstracte:
 - Es mantenen els principis de cohesió i d'ocultació de la informació (cada classe gestiona el seu format de música per a obtenir dades).
 - Independitzem la resta del programa davant de canvis (si es vol donar suport a un nou format, no hi ha que modificar cap classe ja existent).

Exemple de sobreescritura de mètodes

```
public class X {  
    met1 () {...codi X...}  
}  
  
public class Z extends X {  
    met1 () {...codi Z...} // Sobreescritura de met1() d'X  
    met2 () {...} // Mètode inexistent a la classe X  
}  
  
X ox = new X ();  
X oz = new Z ();  
  
ox.met1(); // (1)  
oz.met1(); // (2)  
ox.met2(); // (3)  
oz.met2(); // (4)
```

