

# MOB 1

## Classe :

BUT initial : Représenter une famille d'objets similaires dotés de certaines propriétés communes.

## UML :

Nom →	Human
Champs / Attributs / Slots →	name : String gender : gender birth_year : unsigned
Méthodes / Fonctions membres →	age() : unsigned hello() : void

→ Ses propriétés peuvent inclure un comportement

## Cycle de Vie :

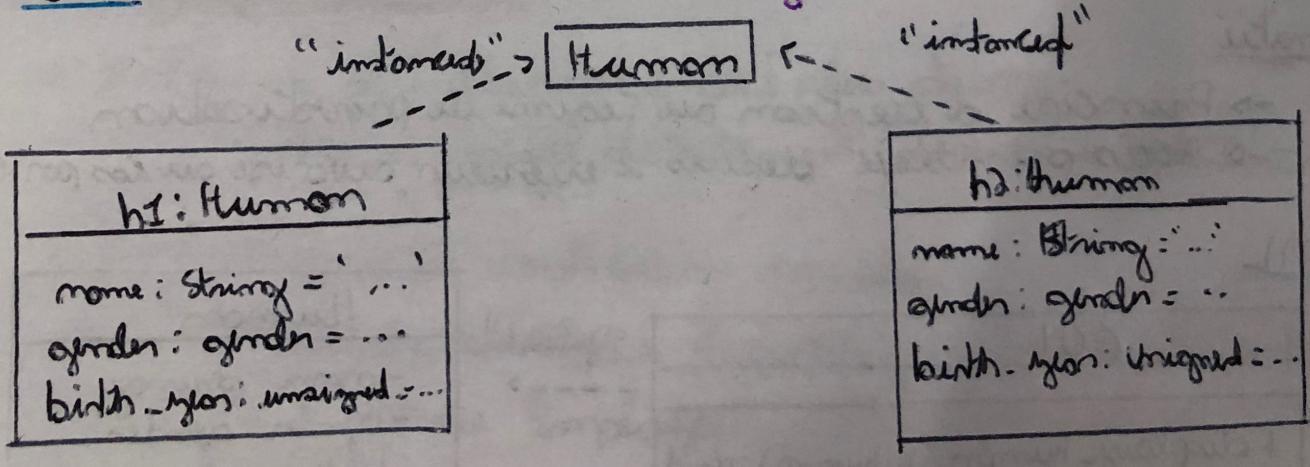
→ Statique, fixé et connu à la compilation.

## Objet :

Instantiation : Processus de création d'un objet à partir d'une classe.

- \* Une classe permet d'instantier plusieurs objets similaires
- \* Un objet n'est l'instance que d'une seule classe.

## UML :



## Cycle de Vie :

→ Dynamique

Créer, utiliser, détruire pendant l'exécution

→ Construction et Destruction

## Niveau de protection:

Champs privés précédés par un ":"

Champs publics précédés par un "+"

Exemple :

Human
- name: String
+ Human(name: String, ...)
+ hello(): Void

Attribut / méthode Privé(e) : accès restreint à la classe

Attribut / Méthode Publique : accès libre

## Notion d'Accesseur:

→ Accesseurs : getteur / Setteur

    → Consultation / modification des attributs privés

→ Interface : ensemble de l'information publique.

## Amitié

→ Principe d'exception au régime de privatisation

→ Accès privilégié depuis l'extérieur autorisé au los par los

## UML

GOT
+ displayHuman(h:Human):Void

"friend"  
----->

Human
- name: String
- gender: gender
- birthYear: unsigned

→ Concept d'amitié variable selon les langages

→ Politique de protection pas définit également

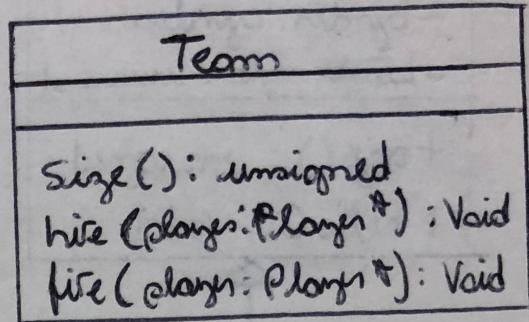
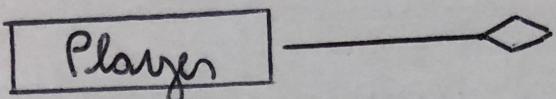
## Aggregation

→ Une forme d'inclusion

→ Aggrégation : maintient des références / pointeurs vers objets ✓

aggrégation

### UML



## Composition (Aggregation Composite)

→ Une forme d'inclusion plus strict

### UML



En autres mots, body comporte une ou plusieurs Head, d'où le nom composition.

## Héritage / Dérivation :

→ Une forme d'inclusion encore plus stricte

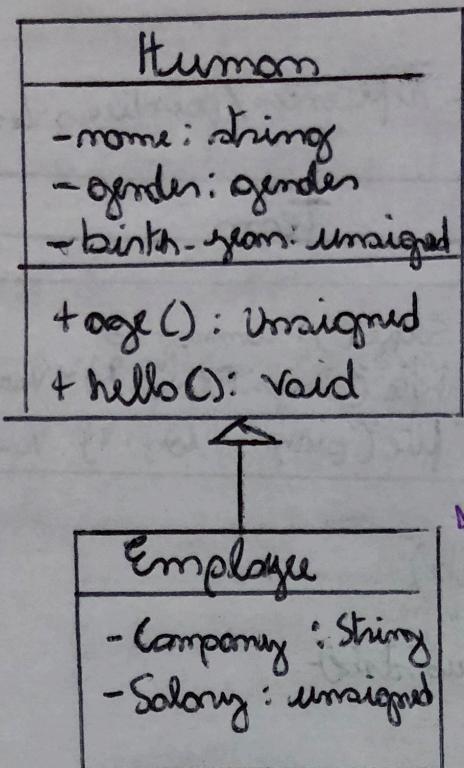
→ Contenu aggregé directement incorporé à la classe

→ Pas de risque de duplication manuelle

→ Le contenu de la classe Human fait implicitement partie de la classe Employee

→ La classe Employee « hérite » ou « dérive » de la classe Human

## UML



Classe mère ou base

Classe fille au dérivé

→ Classe fille contient tous ce qui il y a dans Human.

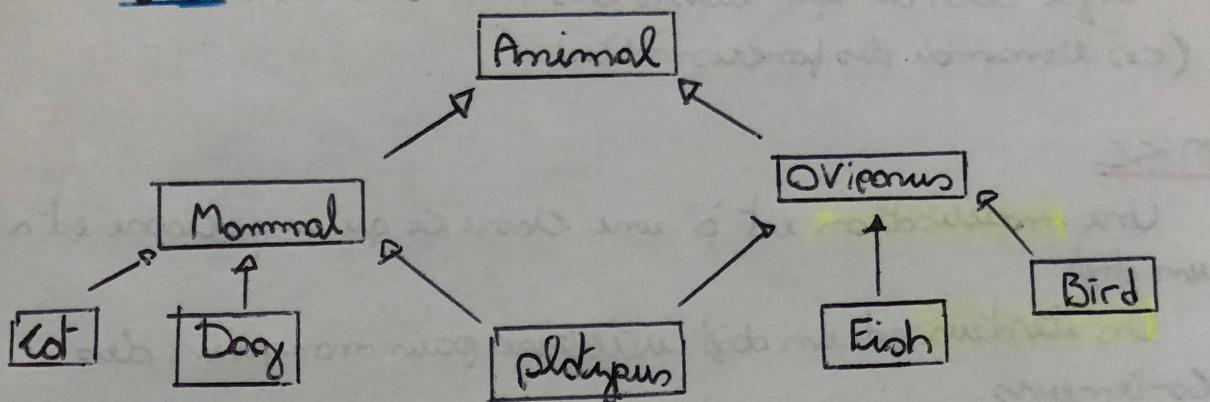
On dit que un employé est un humain, d'où l'héritage.

## Caractéristiques de l'héritage

- Acquisition : Relation de type « a un »  
→ Une équipe a des joueurs, un corps à une tête.
- Héritage : Relation de type « est un »  
→ Un employé est un humain  
Exemple :  
Santony / Président. Santony est un B
- Généralisation : Relation de type « à »  
Exemple : Humain à Animal
- Instantiation : Relation de type « et »  
Exemple : Paris et Ville

## Hierarchies de Classes

UML



→ L'héritage est une relation transitive

→ Héritage multiple (pas toujours dispo) : plusieurs super-<sup>Classe</sup>V

→ Hiérarchie de classes : arbre (au graph) orienté héritage

## Surcharge

n'est pas caractéristique de Loo

## Encapsulation

Type de données un peu plus sécurisé

## Polymorphisme

## Interface

Type abstrait qui illustre des méthodes abstraites  
( $\Leftrightarrow$  l'ensemble des fonctionnalités)

## Misc

Une **modélisation** est à une classe ce qu'une classe est à un objet.

Un **itérateur** est un objet utilitaire pour manipuler des conteneurs.

Une classe abstraite est instanciable. Faux

UML = Unified Modeling Language

Type d'attribut ou méthode : public, private, protected.

L'héritage a une notion « a un » FAUX

Une méthode constante peut être appellée sur un objet non constant. VRAI

Les trois axes d'un logiciel sont l'axe statique, l'axe dynamique et l'**axe fonctionnel**.

Un diagramme de cas d'utilisation est comme un film : il y a des acteurs. Vrai