Cours de Génie Logiciel

Chapitre 4 : Diagramme de classes

2022-2023

Dr. Kouah S.

Classe & Objet

Conception orientée objet : Représentation du système comme un ensemble d'objets interagissant

Diagramme de classes

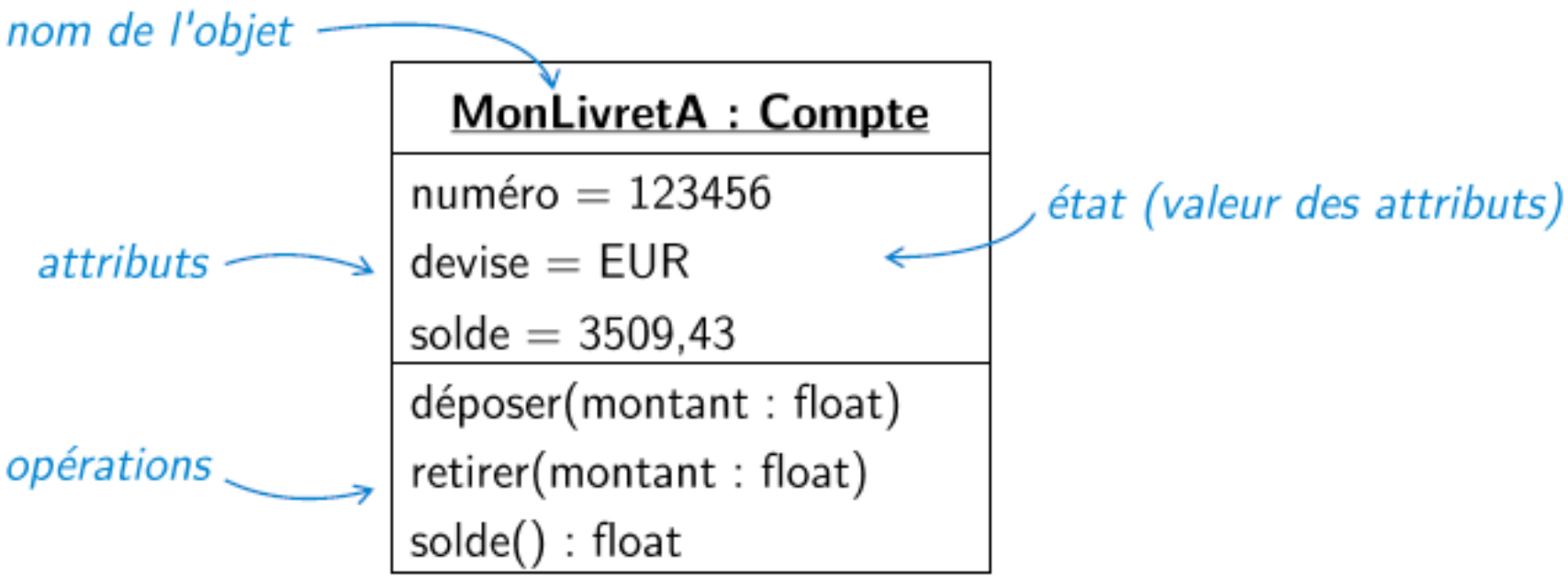
- Représentation de la structure interne du logiciel
- Utilisé surtout en conception mais peut être utilisé en analyse

Diagramme d'objets

- Représentation de l'état du logiciel (objets + relations)
- Diagramme évoluant avec l'exécution du logiciel
  - création et suppression d'objets
  - modification de l'état des objets (valeurs des atributs)
  - modification des relations entre objets

Objet

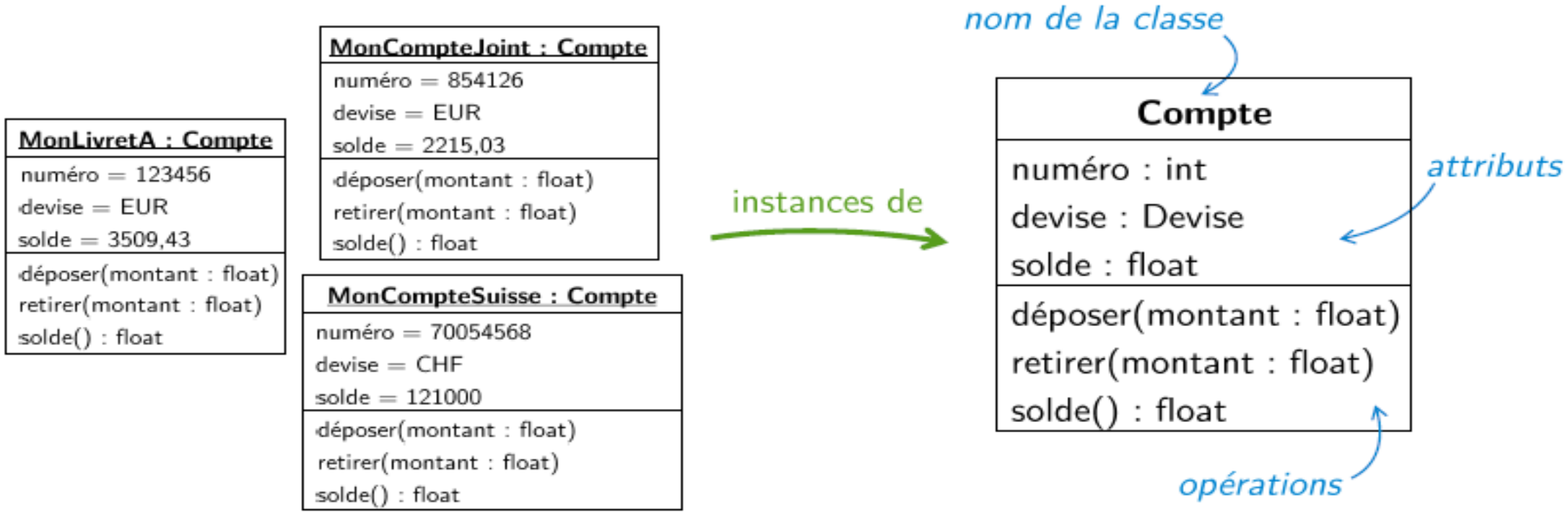
- Entité concrète ou abstraite du domaine d'application
- Décrit par : identité (adresse mémoire)
  + état (attributs)
  + comportement (opérations)



Classe

Classe : Regroupement d'objets de même nature (mêmes attributs + mêmes opérations)

Objet = instance d'une classe



Exemple

Exemple de la bibliothèque

On cherche à développer un système qui gère les emprunts et les retours dans une bibliothèque.

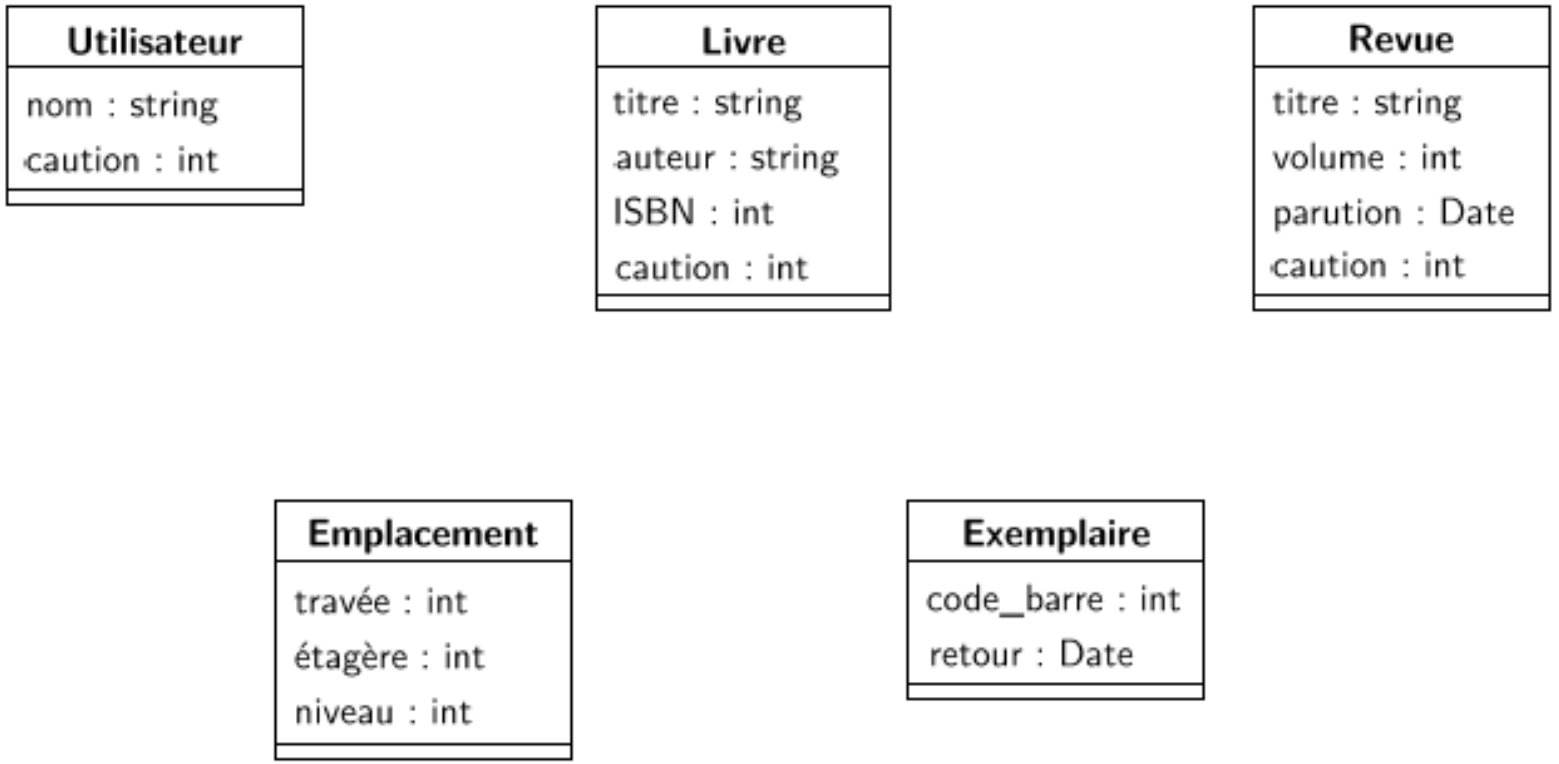
La bibliothèque gère des livres et des revues. Un livre est caractérisé par son titre, son auteur et son code ISBN. Un numéro de revue est caractérisé par le titre de la revue, un numéro de volume et sa date de parution. Chaque exemplaire d'une ressource est caractérisé par un code barre au sein de la bibliothèque.

Pour emprunter un ouvrage, un utilisateur doit être enregistré. Il s'enregistre auprès du bibliothécaire en donnant son nom et une caution. Chaque ouvrage a une caution. Un utilisateur ne peut emprunter un ouvrage que si la caution qui lui reste sur son compte est supérieure à la caution de l'ouvrage. La durée de l'emprunt est fixée à 15 jours.

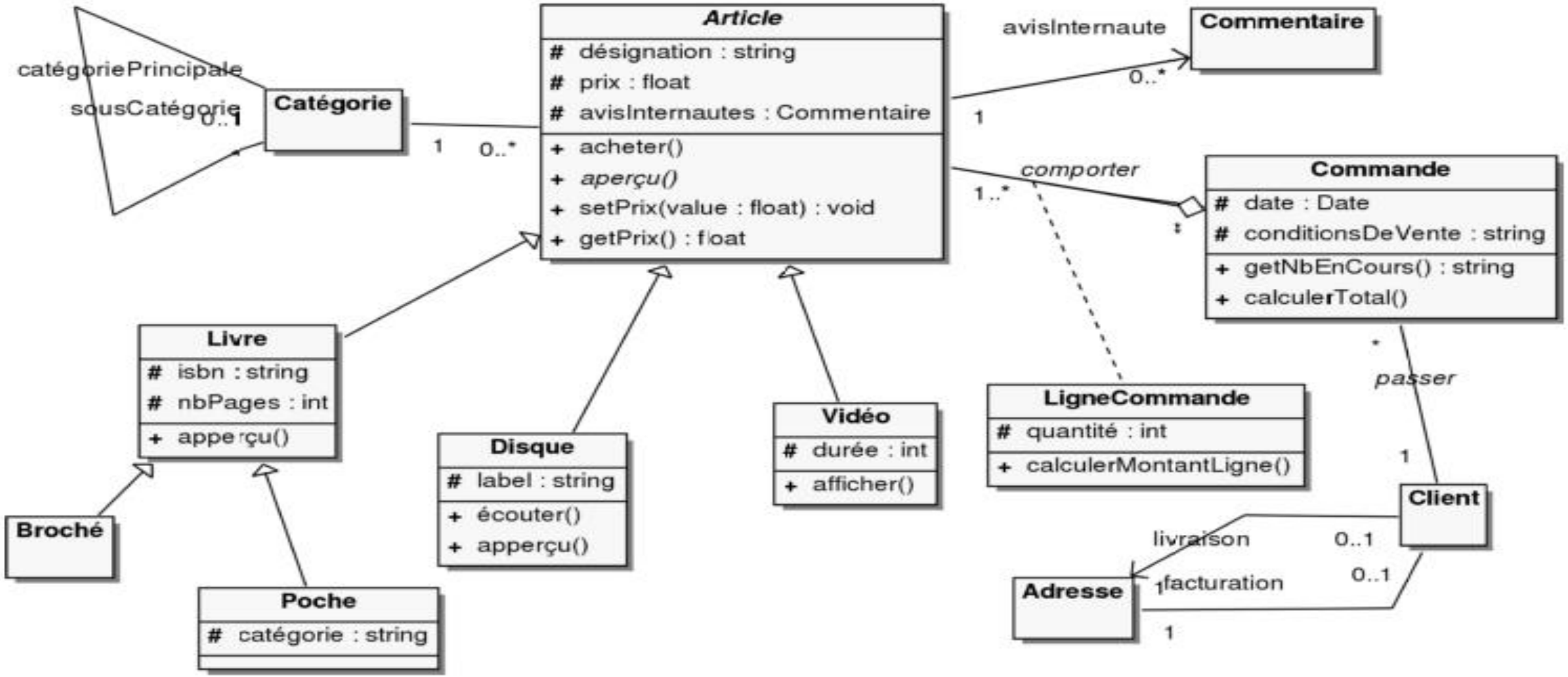
On ne peut pas emprunter plus d'un exemplaire d'une même ressource, ni emprunter une nouvelle ressource si on est en retard pour rendre une ressource.

L'emplacement de stockage d'un ouvrage dans la bibliothèque est représenté par un numéro de travée, un numéro d'étagère dans la travée, et un niveau. Différentes ressources peuvent être rangées au même emplacement, mais tous les exemplaires d'une même ressource sont stockés au même endroit.

Exemple

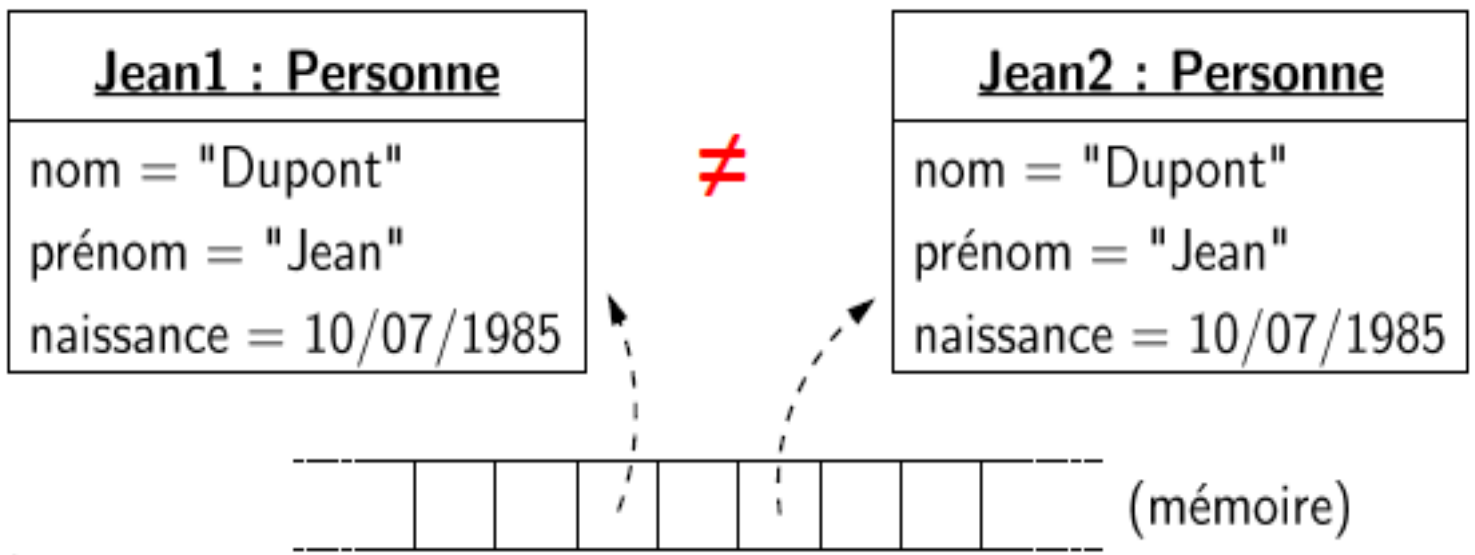

Utilisateur
nom : string
caution : int
Livre
titre : string
auteur : string
ISBN : int
caution : int
Revue
titre : string
volume : int
parution : Date
caution : int
Emplacement
travée : int
étagère : int
niveau : int
Exemplaire
code_barre : int
retour : Date

Exemple de diagramme de classes



Valeur des attributs : État de l'objet

- Objets différents (identités différentes) peuvent avoir mêmes attributs



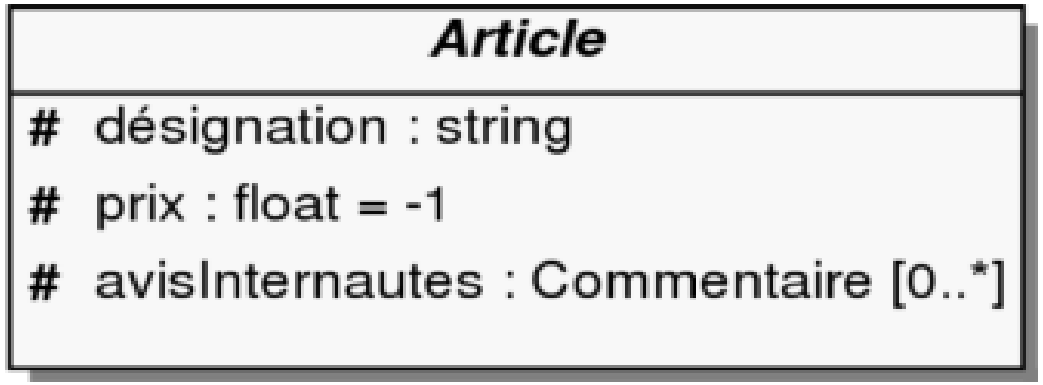
Propriété = Attribut + Operations

- Les attributs et les opérations sont les **propriétés** d'une classe. Leur nom commence par une minuscule.
  - Un **attribut** décrit une donnée de la classe.
    - Les types des attributs et leurs initialisations ainsi que les modificateurs d'accès peuvent être précisés dans le modèle
    - Les attributs prennent des valeurs lorsque la classe est instanciée : ils sont en quelque sorte des « variables » attachées aux objets.
  - Une **opération** est un service offert par la classe (un traitement que les objets correspondant peuvent effectuer).

Attribut

- Un attribut peut être initialisé et sa visibilité est définie lors de sa déclaration.
- Syntaxe de la déclaration d'un attribut :

```
modifAcces nomAtt:nomClasse[multi]=valeurInit
```



Opération

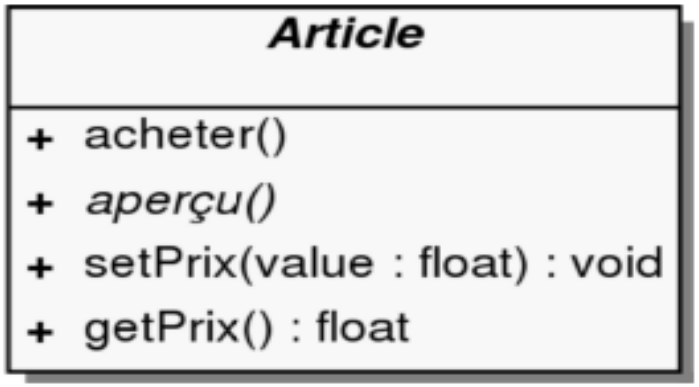
Une opération est définie par son ainsi que par les types de ses paramètres et le type de sa valeur de retour.

- La syntaxe de la déclaration d'une opération est la suivante :

```
modifAcces nomOperation(parametres):ClasseRetour
```

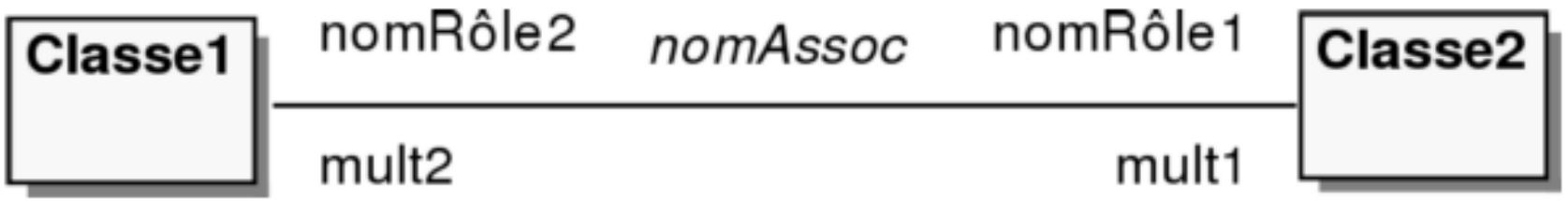
- La syntaxe de la liste des paramètres est la suivante :

```
nomClasse1 nomParam1, ... , nomClasseN nomParamN
```



Association

- Une **association** est une relation structurelle entre objets.
  - Une association est souvent utilisée pour représenter les liens posibles entre objets de classes données.
  - Elle est représentée par un trait entre classes.



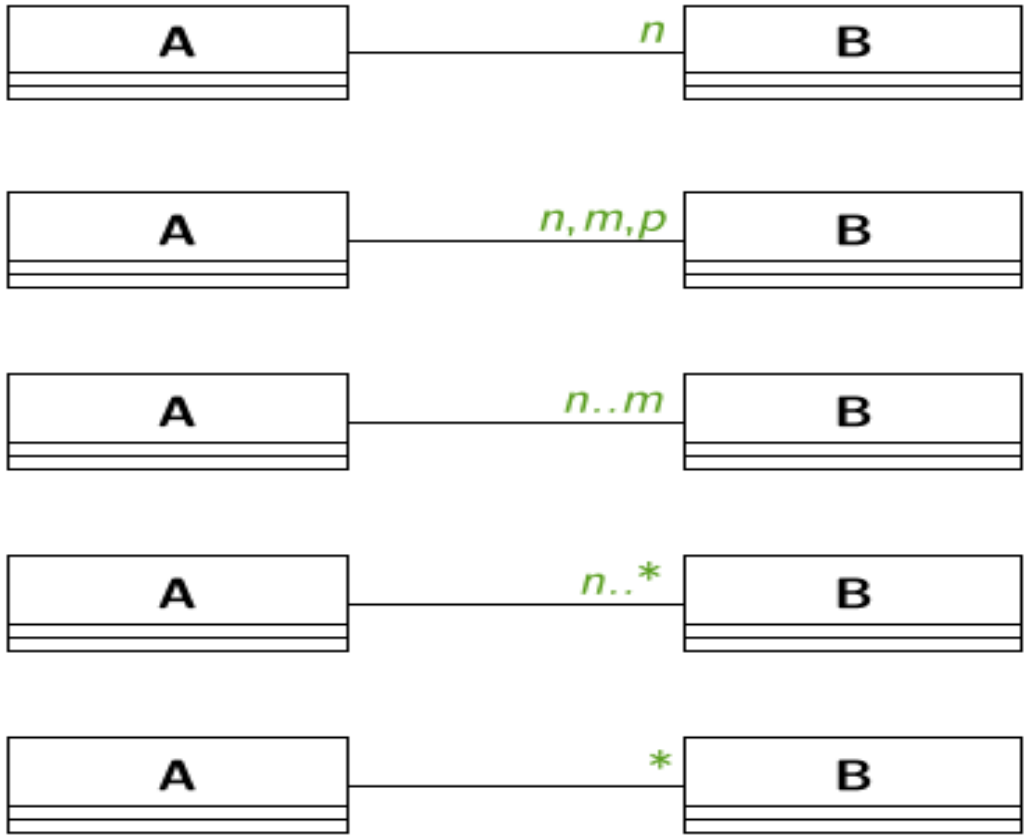
Multiplicité

- La notion de **multiplicité** permet le contrôle du nombre d'objets intervenant dans chaque instance d'une association.
  - **Exemple :** un article n'appartient qu'à une seule catégorie (1) ; une catégorie concerne plus de 0 articles, sans maximum (*).

- La syntaxe est MultMin..MultMax.
  - « * » à la place de MultMax signifie « plusieurs » sans préciser de nombre.
  - « n..n » se note aussi « n », et « 0..* » se note « * ».

Multiplicité

Nombre d'objets de la classe B associés à un objet de la classe A



Exactement *n*

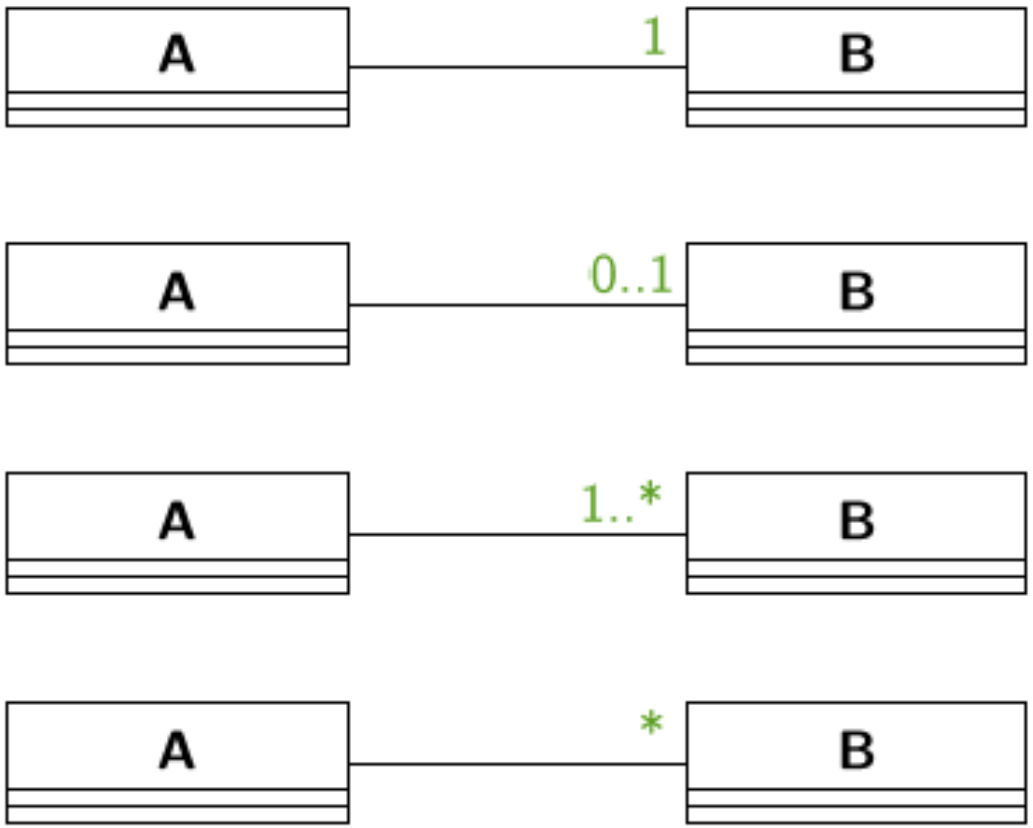
Exactement *n* ou *m* ou *p*

Entre *n* et *m*

Au moins *n*

Plusieurs (0 ou plus)

Multiplicité



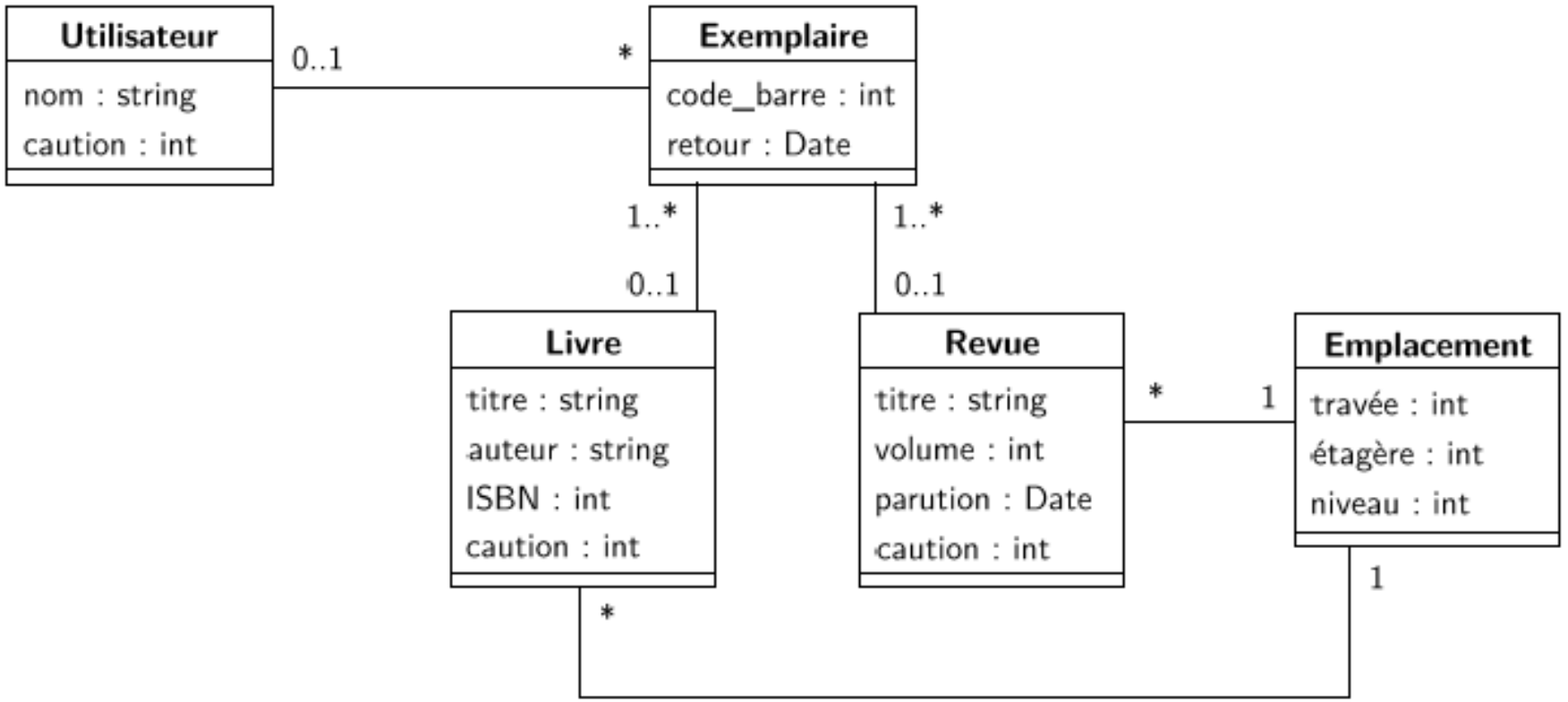
Exactement 1

Au plus 1 (0 ou 1)

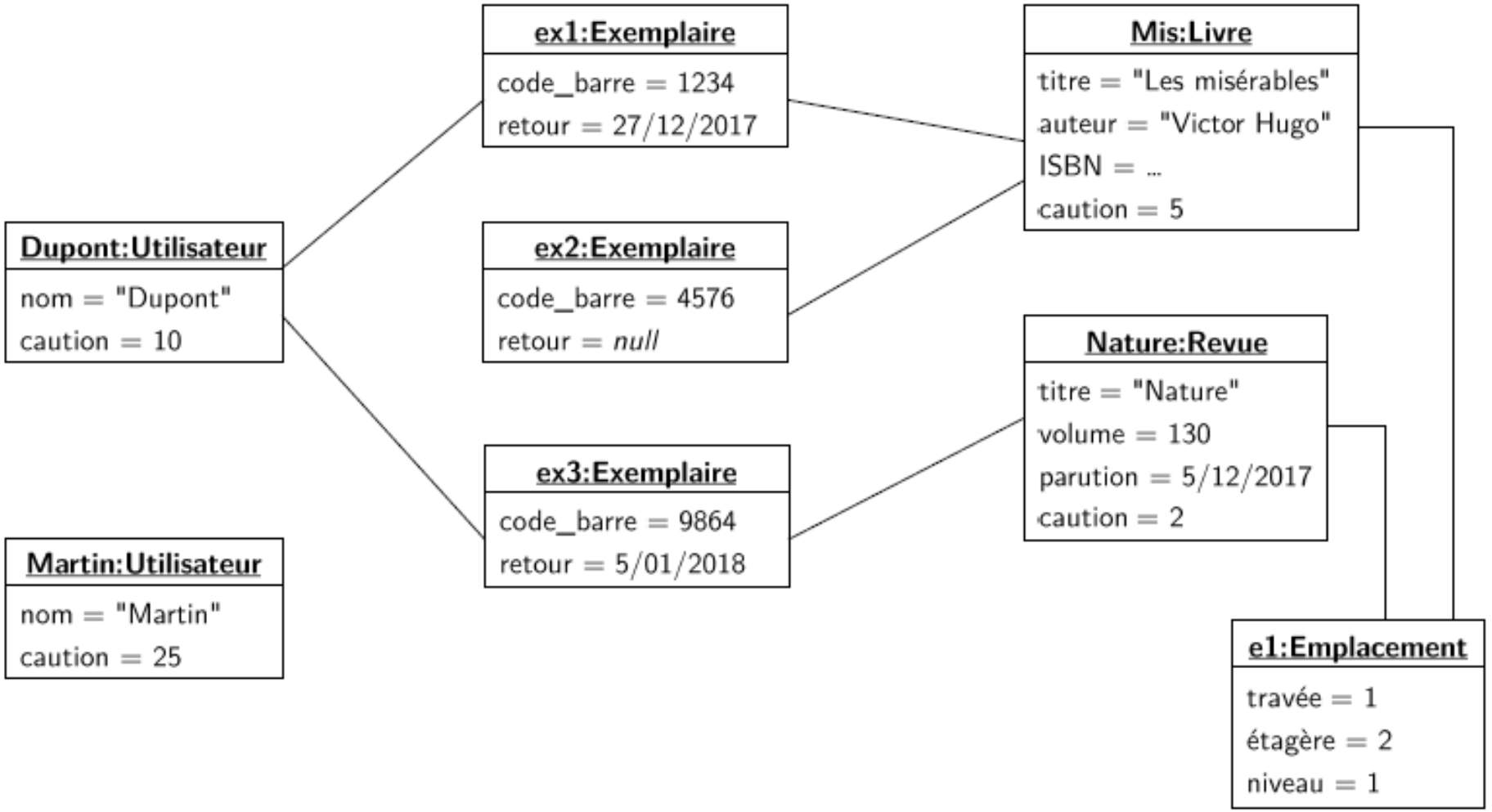
Au moins 1 (jamais 0)

0 ou plus

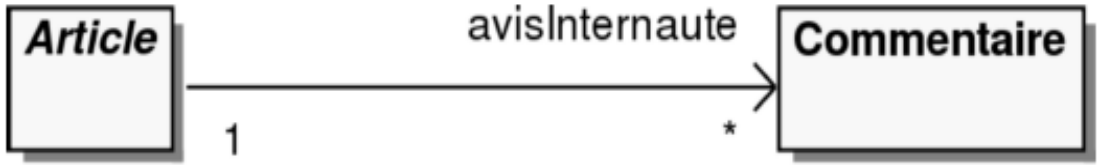
Exemple de diagramme de classe 2

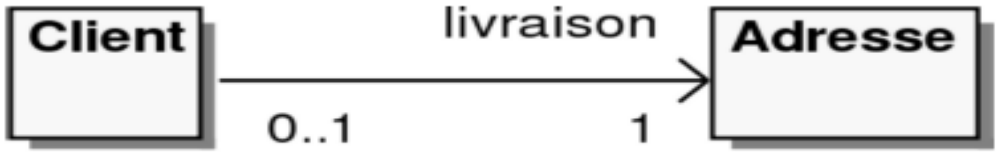


Exemple de diagramme d'objets

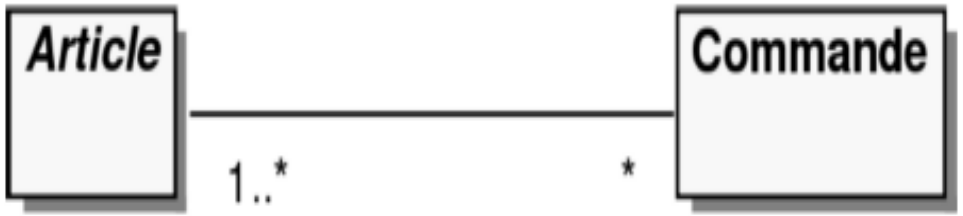


Association unidirectionnelle





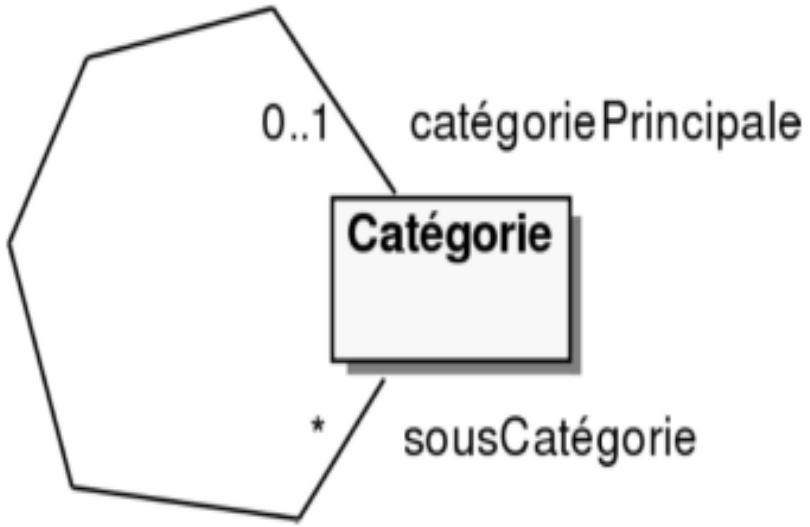
Association binaire



Plus difficile : gérer à la fois la cohérence et les collections

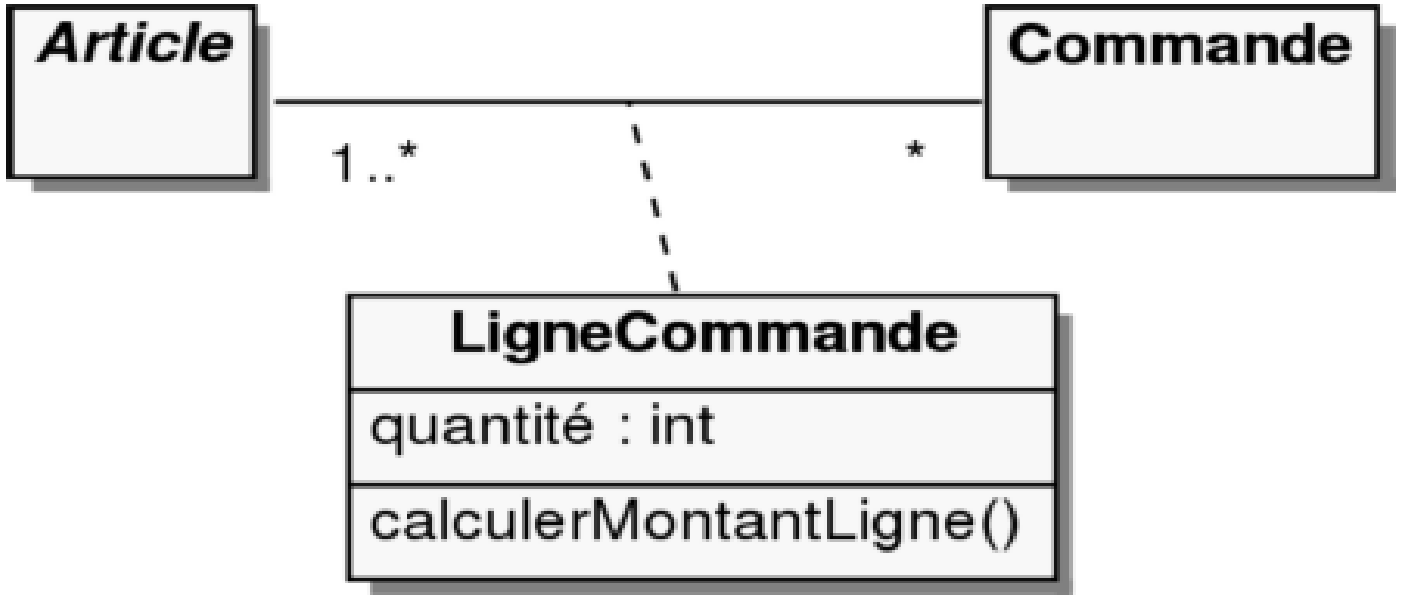
Association Réflexive

- L'association la plus utilisée est l'association binaire (reliant deux classs).
- Parfois, les deux extrémités de l'association pointent vers le même classeur. Dans ce cas, l'association est dite « **réflexive** ».



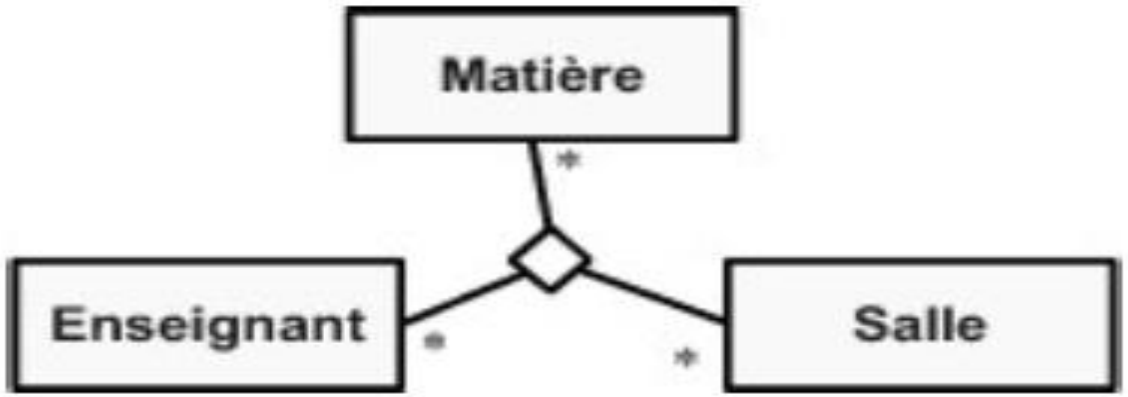
Classe Association

- Une association peut être raffinée et avoir ses propres attributs, qui ne sont disponibles dans aucune des classes qu'elle lie.
- Comme, dans le modèle objet, seules les classes peuvent avoir des attributs, cette association devient alors une classe appelée « **classe-association** ».



Association N-aire

- Une **association n-aire** lie plus de deux classes.
  - Notation avec un losange central pouvant éventuellement accueillir une classe-association.
  - La multiplicité de chaque classe s'applique à une instance du losange.

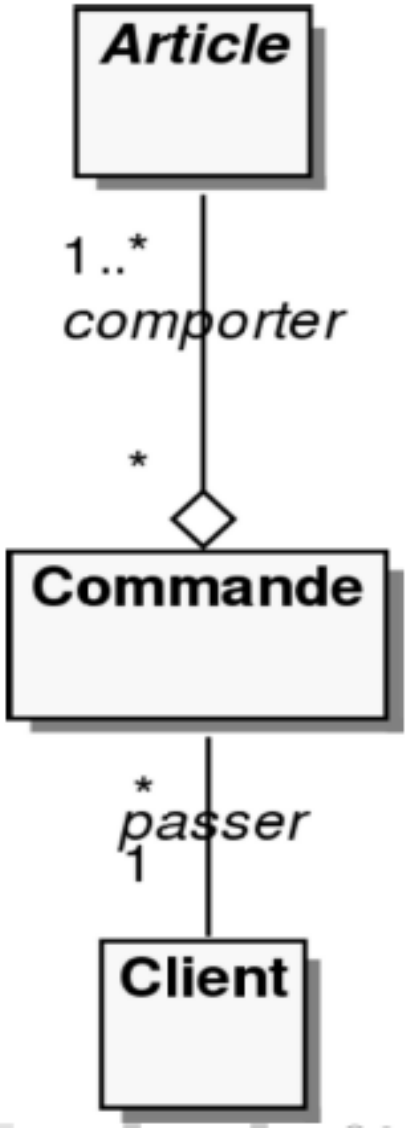


Les associations n-aires sont peu fréquentes et concernent surtout les cas où les multplicités sont toutes « * ». Dans la plupart des cas, on utilisera plus avantageusement des classes-association ou plusieurs relations binaires.

Agrégation

- Une **agrégation** est une forme particulière d'association. Elle représente la relation d'**inclusion** d'un élément dans un ensemble.
- On représente l'agrégation par l'ajout d'un losange vide du côté de l'agrégat.

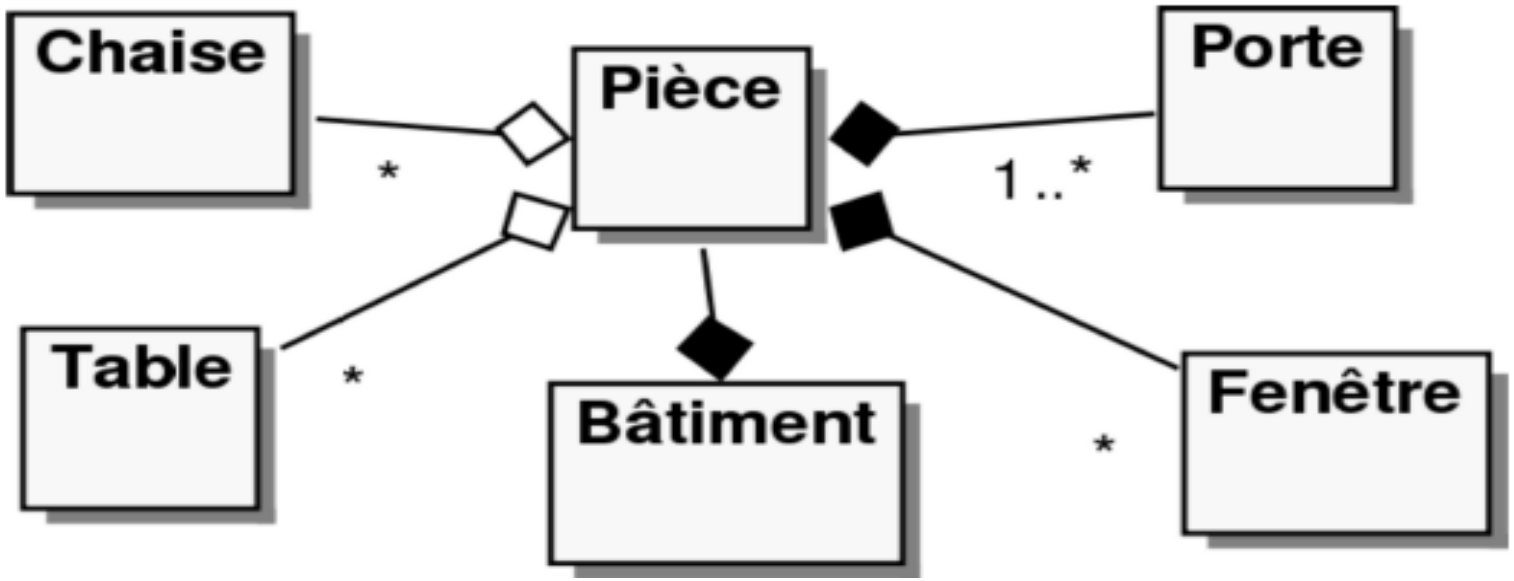
Une agrégation dénote une relation d'un ensemble à ses parties. L'ensemble est l'**agrégat** et la partie l'**agrégé.**



Composition

- La relation de **composition** décrit une **contenance** structurelle entre instances. On utilise un losange plein.

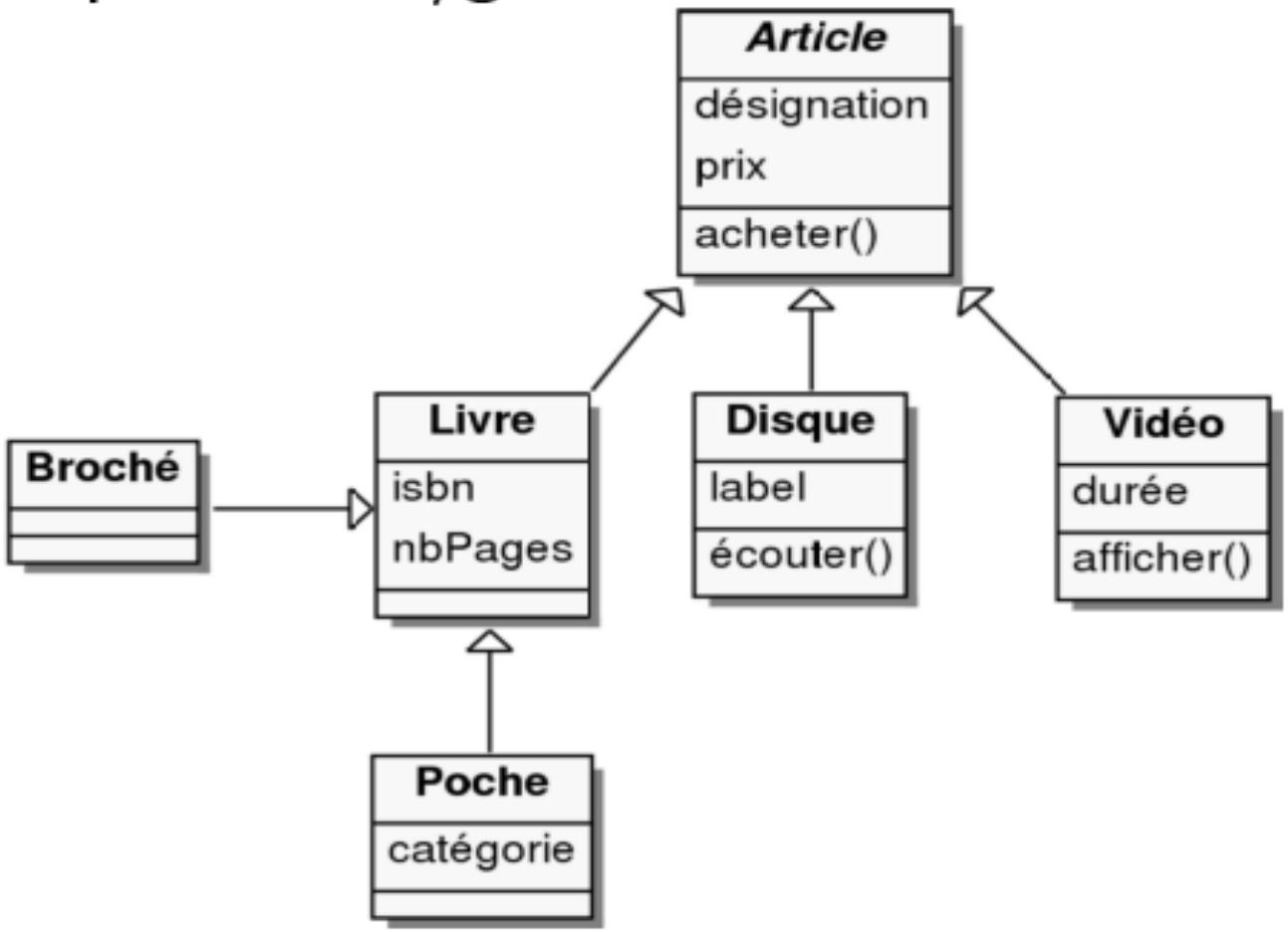
- La **destruction** et la **copie** de l'objet composite (l'ensemble) impliquent respectivement la destruction ou la copie de ses composants (les parties).
- Une instance de la partie n'appartient jamais à plus d'une instance de l'élément composite.



Héritage

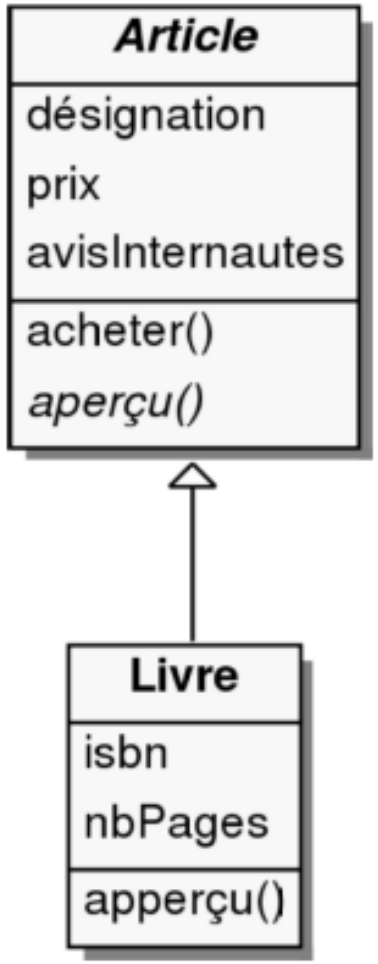
- **L'héritage** une relation de spécialisation/**généralisation.**

- Les éléments spécialisés héritent de la structure et du comportement des éléments plus généraux (attributs et opérations)
- **Exemple :** Par héritage d'Article, un livre a d'office un prix, une désignation et une opération acheter(), sans qu'il soit nécessaire de le préciser



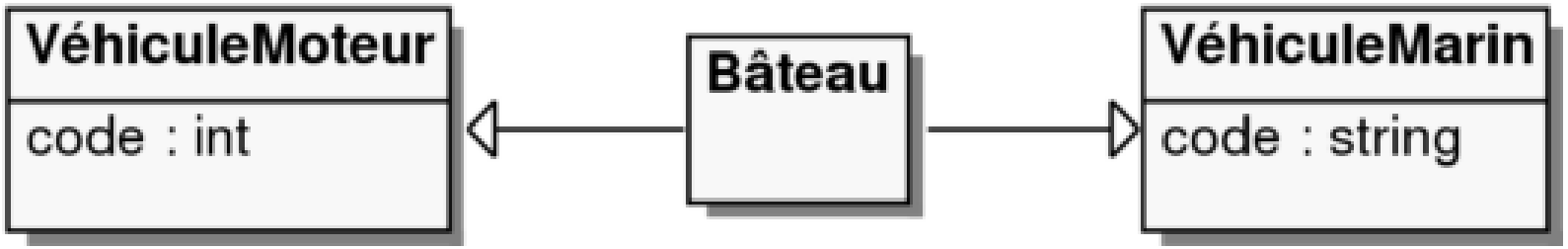
Héritage

- **La classe enfant possède toutes les propriétés de ses classes parents** (attributs et opérations)
  - La classe **enfant** est la classe spécialisée (ici Livre)
  - La classe **parent** est la classe générale (ici Article)
- Toutefois, elle n'a pas accès aux propriétés privées.



Héritage multiple

- Une classe peut avoir plusieurs classes parents. On parle alors d'**héritage multiple.**
  - Le langage C++ est un des langages objet permettant son implantation effective.
  - Java ne le permet pas.

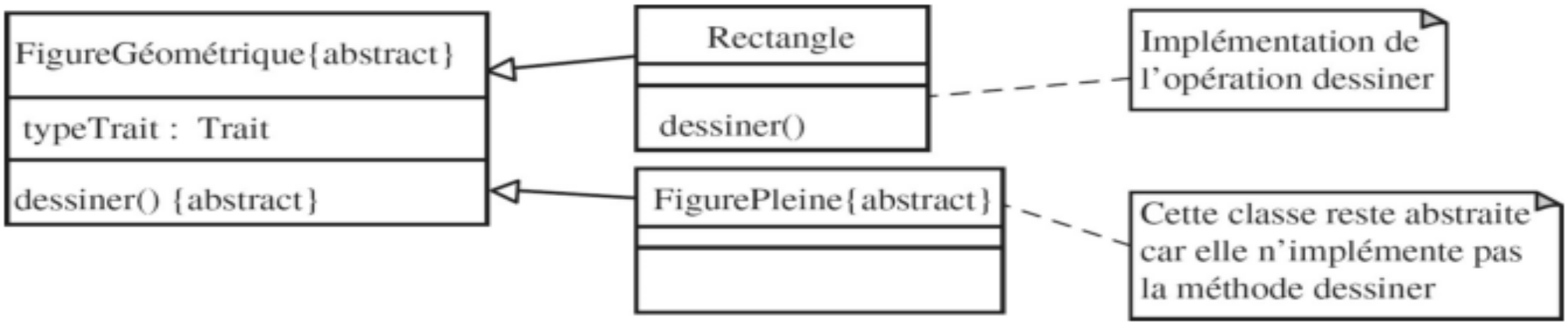


Classe Abstraite

- Une méthode est dite **abstraite** lorsqu'on connaît son entête mais pas la manière dont elle peut être réalisée.

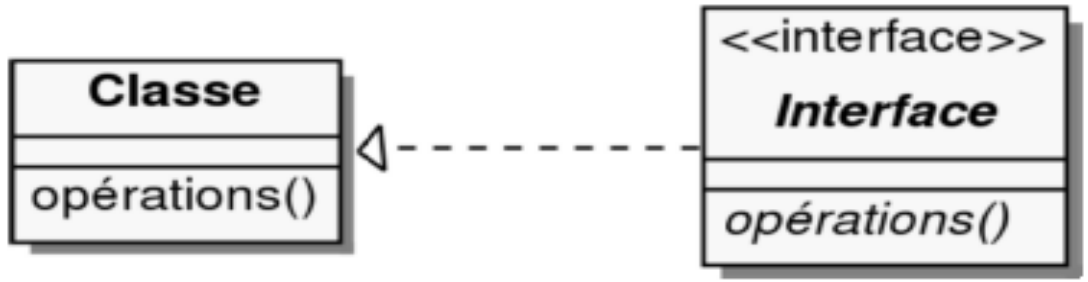
Il appartient aux classes enfant de définir les methodes abstraites.

- Une classe est dite **abstraite** lorsqu'elle définit au moins une méthode abstraite ou lorsqu'une classe parent contient une méthode abstraite non encore réalisée.



Interface

- Le rôle d'une **interface** est de regrouper un ensemble d'opérations assurant un service cohérent offert par un classeur et une classe en particulier.
- Une interface est définie comme une classe, avec les mêmes compartiments. On ajoute le stéréotype « interface » avant le nom de l'interface.
- On utilise une relation de type **réalisation** entre une interface et une classe qui l'implémente.



- Les classes implémentant une interface doivent implémenter toutes les opérations décrites dans l'interface

Références

- Cours de delphine Longuet. Université Paris Sud.
- Autre… a compléter