# Génie Logiciel

## Chapitre 4-Partie 1

# Diagrammes de classes et d'objets UML : vue statique

Niveau: 3ième année Licence informatique

Année: 2023/2024

# Quelques outils pour la modélisation

- ArgoUML (Open source)
- PlantUML
- StarUML
- BoUML
- Power Designer (payant )
- Visual Paradigm (payant)
- Enterprise Architect (payant)
- Eclipse: MoDisco
- Eclipse: Papyrus

# Diagramme de Classes

# Diagramme de Classes

- Les *diagrammes de cas d'utilisation* modélisent à *QUOI* sert le système.
- Le système est composé d'objets qui interagissent entre eux et avec les acteurs pour réaliser ces cas d'utilisation.

- Diagramme de classes
  - ✓ Offre une représentation abstraite de l'ensemble des objets du système qui vont interagir ensemble pour l'achèvement des cas d'utilisation

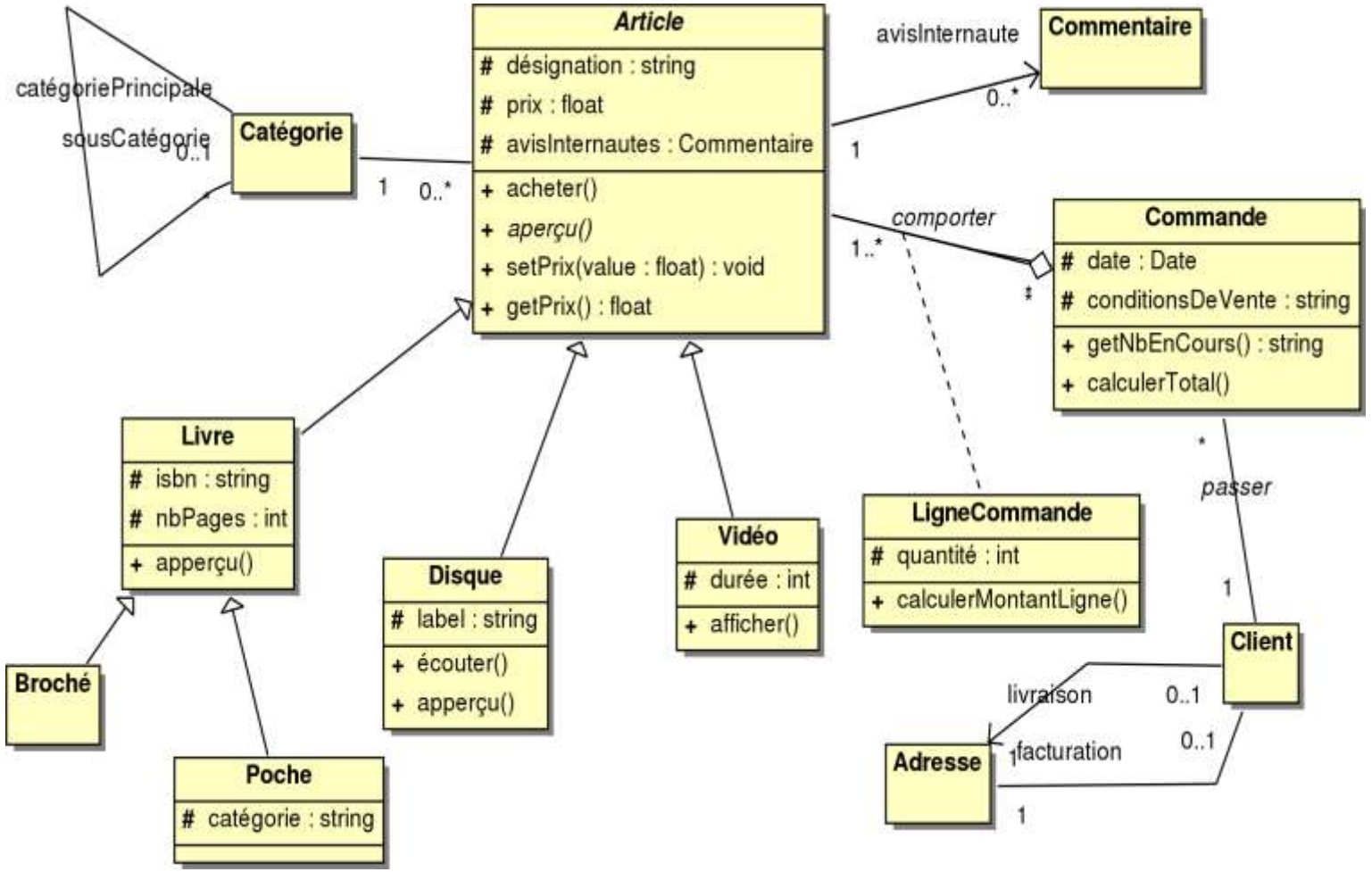
# Diagramme de Classes

- Modélise la structure statique d'un système sous forme de classes, d'interfaces et de leurs relations

- Diagramme le plus important dans la modélisation O.O.

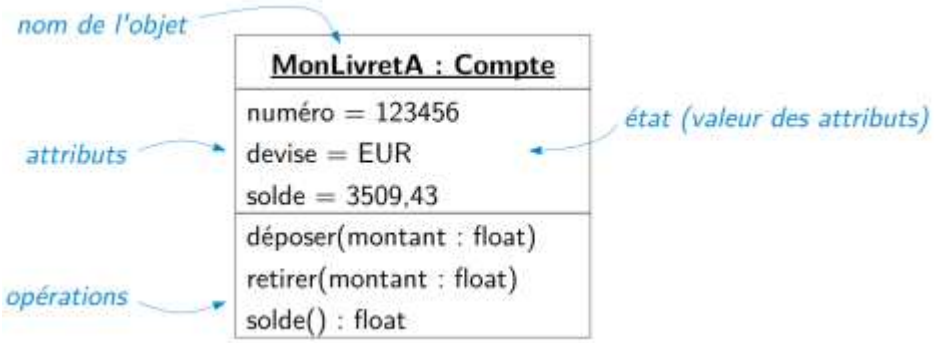
- Un graphes
  - ✓Nœuds : Classes et interfaces
  - ✓Arcs : Relations entre des classes et des interfaces

# Diagramme de Classes



# Objet

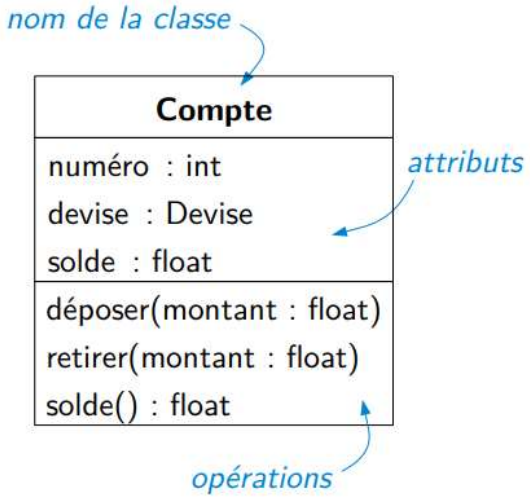
- Entité concrète ou abstraite du domaine d'application
- Décrit par :
  - ✓ identité (adresse mémoire)
  - ✓ état (attributs): ensemble de valeurs
  - ✓ comportement (opérations)




- Entité autonome
- Possède une durée de vie (création et la destruction)
- Objet = Etat + Comportement + Identité

# Classe

- Une description abstraite d'un type d'objets
- La description d'un ensemble d'objets ayant une sémantique, des attributs, des méthodes et des relations en commun.
  - Spécifie l'ensemble des caractéristiques qui composent des objets de même type.
- Une classe est composée d'un *nom*, *d'attributs* et d'*opérations*.
  - ✓ Selon l'avancement de la modélisation, ces informations ne sont pas forcement toutes connues.



# Objets et Classes

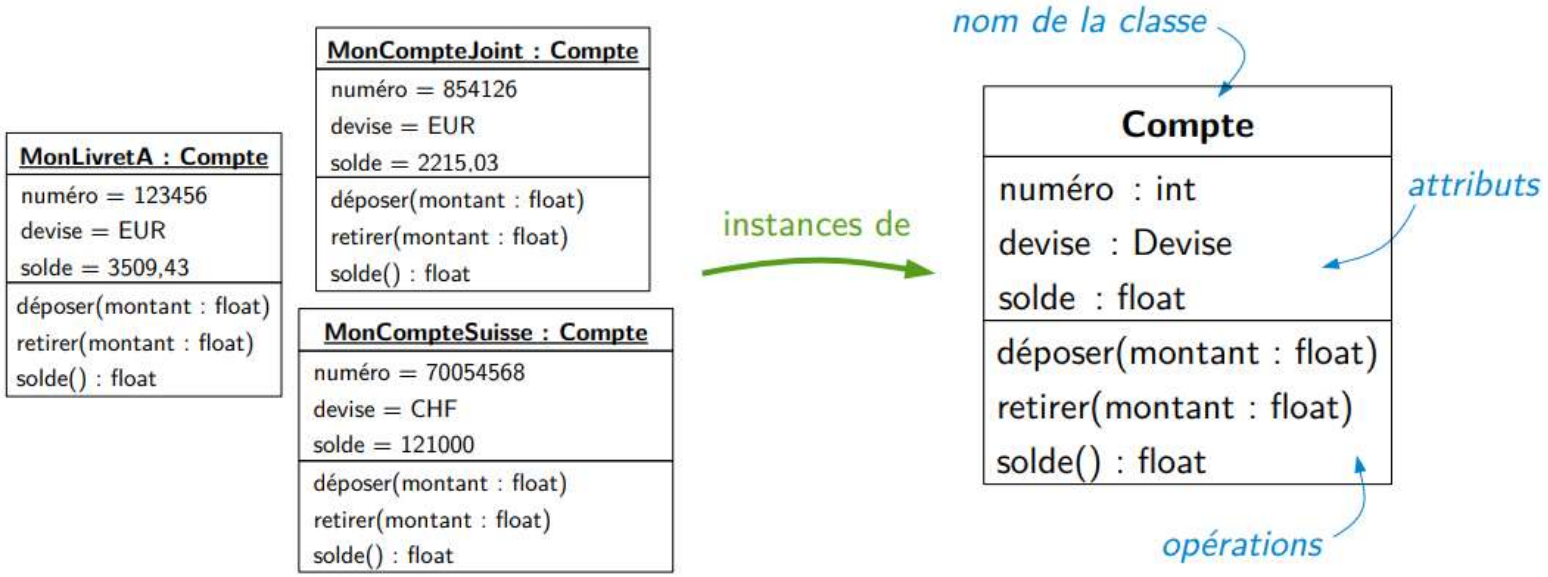
- Une *instance* est la concrétisation d'un concept abstrait
  - ✓ Concept : Stylo
  - ✓ Instance : le stylo que vous utilisez à ce moment précis: il a sa propre forme, sa propre couleur, son propre niveau d'usure, etc.

- Un *objet* est une instance d'une classe
  - ✓ Classe : Vidéo
  - ✓ Objets : *Apocalypto*, Pink Floyd, Odyssée de l'Espace etc.

- L'instanciation : création d'un objet d'une classe (via le constructeur)

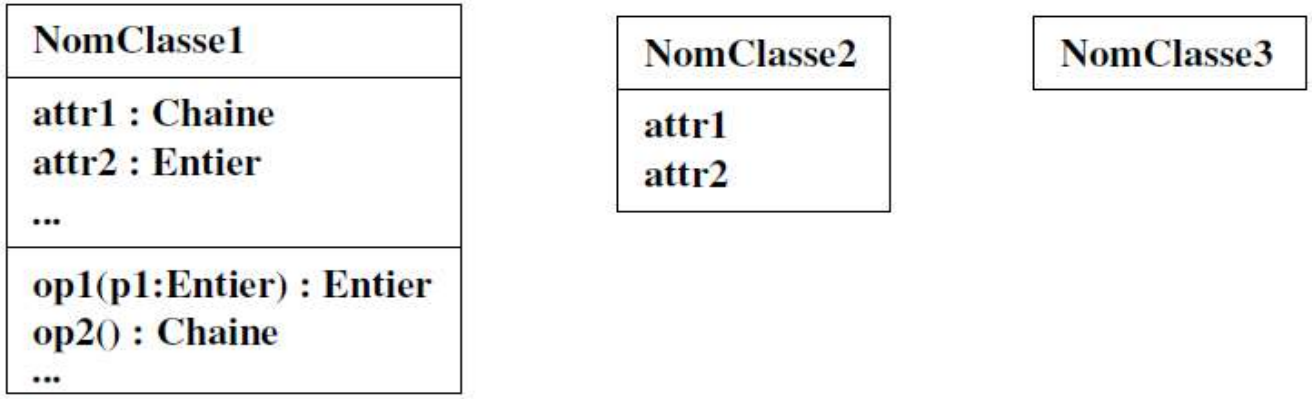
# Objets et Classes (2)

- Un objet = instance d'une classe
- Une classe correspond à un plan, un moule, une usine...



# Classes- Notation UML

- Une classe est représentée par un rectangle (classeur) contenant trois parties (compartiments)
- Différents niveaux de détail:
  - ✓ Possibilité d'omettre attributs et/ou opérations
  - ✓ D'autres compartiments peuvent être ajoutés : responsabilités, exceptions, etc.



# Propriétés : attributs et opérations

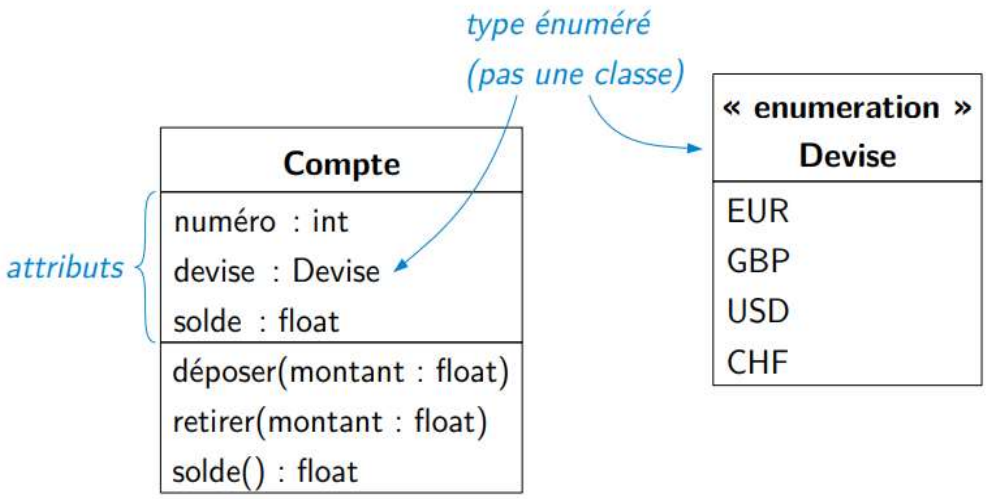
- Les attributs et les opérations sont les propriétés d'une classe.
  - ✓ Leur nom commence par une minuscule.

# Attribut

- Les attributs
  - ✓ Caractéristique partagée par tous les objets de la classe
  - ✓ Associent à chaque objet une valeur
  - ✓ Prennent des valeurs lorsque la classe est instanciée

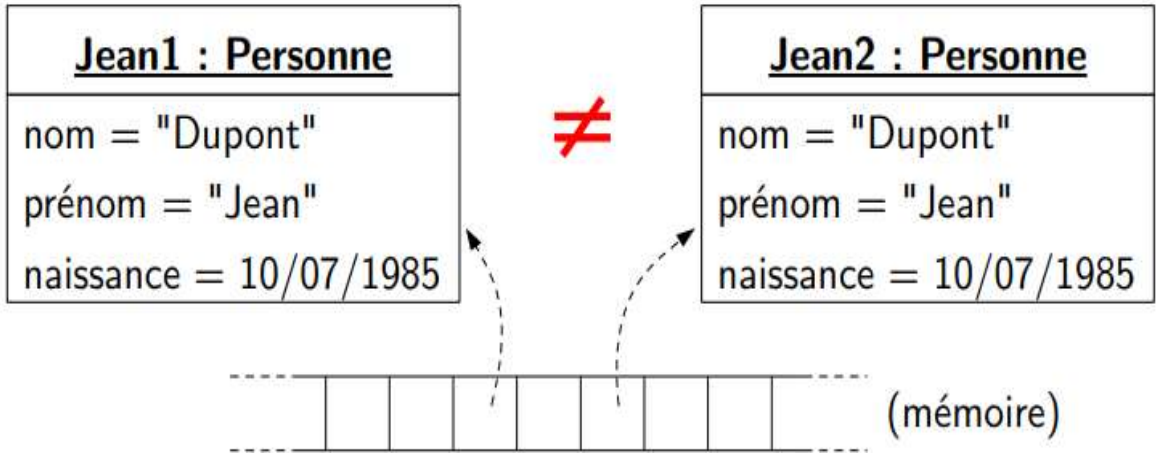
- Les types des attributs et leurs initialisations ainsi que les modificateurs d'accès peuvent être précisés dans le modèle

- Type associé:
  - ✓ simple (int, bool...),
  - ✓ primitif (Date)
  - ✓ ou énuméré



# Attribut (2)

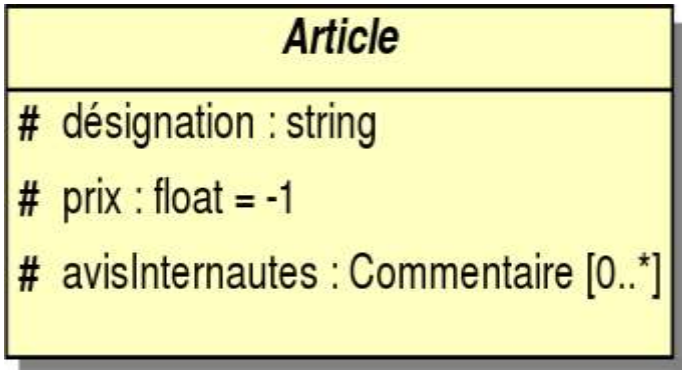
- Valeur des attributs : État de l'objet
- Objets différents (identités différentes) peuvent avoir mêmes attributs



# Attribut (3)

- Un attribut peut être initialisé et sa visibilité est définie lors de sa déclaration
- Syntaxe :

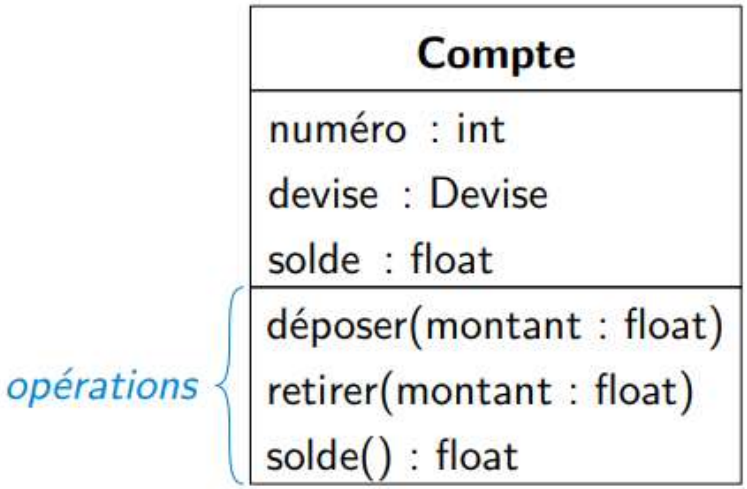
modifAcces nomAtt : nomClasse [ multi ]= valeurInit




- modifAcces: Le symbole de visibilité : + (public), - (private), # (protected), ~ (package)
- multi : multiplicité (optionnel) Indique combien de valeurs peuvent être contenues dans l'attribut (utile pour les collections): [1], [0..*], [1..5]

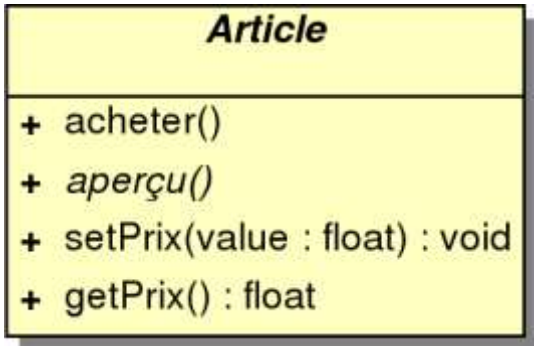
# Opération

- Un service offert par tout objet de la classe (un traitement effectué).
- Comportement commun à tous les objets de la classe



# Opération (2)

- Une opération est définie par
  - ✓ son non
  - ✓ les types de ses paramètres
  - ✓ le type de sa valeur de retour




- Syntaxe de la déclaration :

```
modifAcces nomOperation ( parametres ): typeRetour
```

- Syntaxe de la liste des paramètres :

```
nomParam1 : typeParam1, ... , nomParamN : typeParamN
```

# Opération (3)

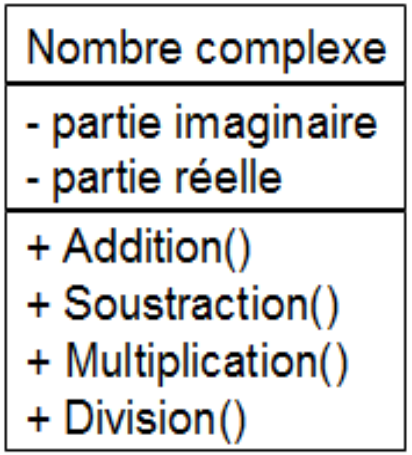
- Ne pas confondre avec une méthode
  - ✓ Une opération est la spécification d'une méthode (sa signature) indépendamment de son implantation
  - ✓ méthode = implantation de l'opération

- UML 2 autorise également la définition des opérations dans n'importe quel langage donné

# Visibilité

- Niveaux de visibilité
  - ✓ (+) public : visible par tous les autres objets
  - ✓ (-) private : visible seulement depuis l'intérieur de l'objet
  - ✓ (#) protected : visible par certains objets (les descendants de la classe)
  - ✓ (package ou ~ ou rien) : visibilité à l'intérieur du package

- Sil les attributs sont accessibles de l'extérieur, l'accès doit se faire par l'intermédiaire d'opérations et non directement.

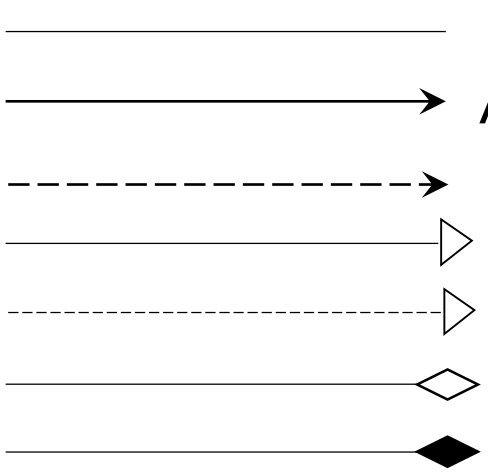


# Les associations

# Relations entre classes

- Héritage
  - ✓ une relation de généralisation/spécialisation permettant l'abstraction.
- Association
  - ✓ Représente une relation sémantique entre les objets d'une classe.
- Dépendance
  - ✓ une relation unidirectionnelle exprimant une dépendance sémantique entre les éléments du modèle.
- Agrégation
  - ✓ décrit une relation de contenance ou de composition.

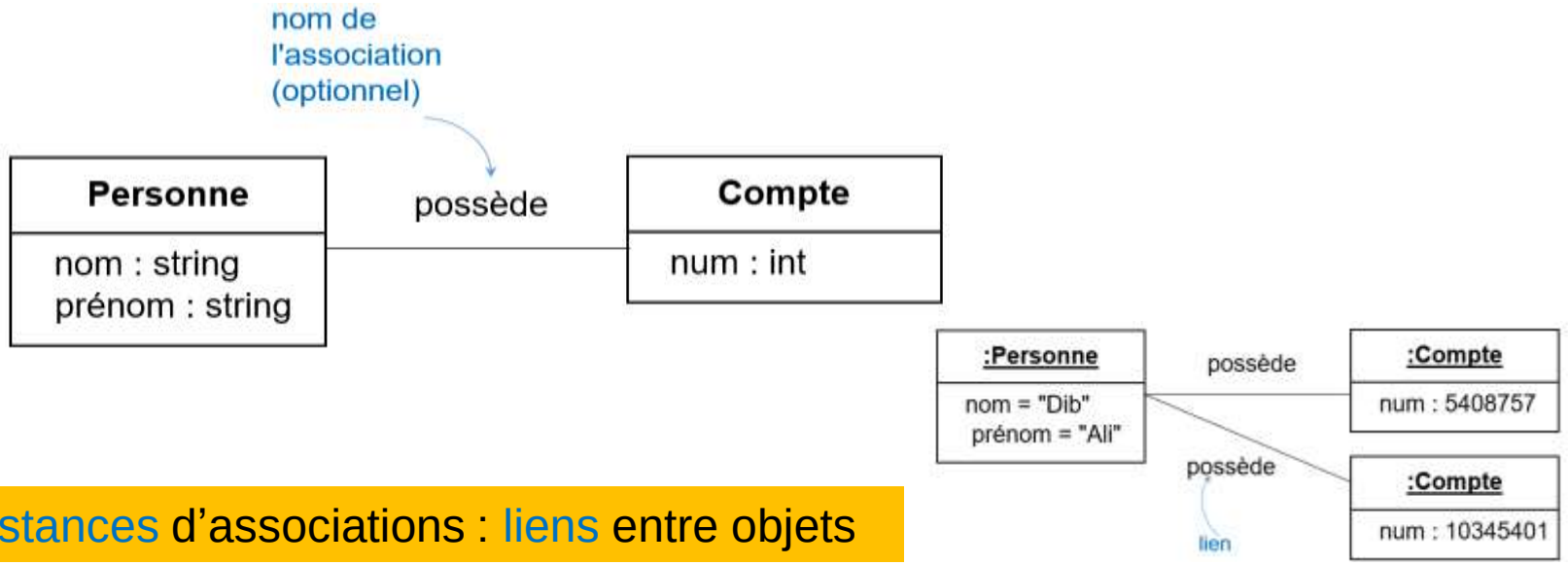
# Les flèches en UML



Association bidirectionnelle
Association unidirectionnelle
Dépendance
Héritage
Implémentation
Agrégation
Composition

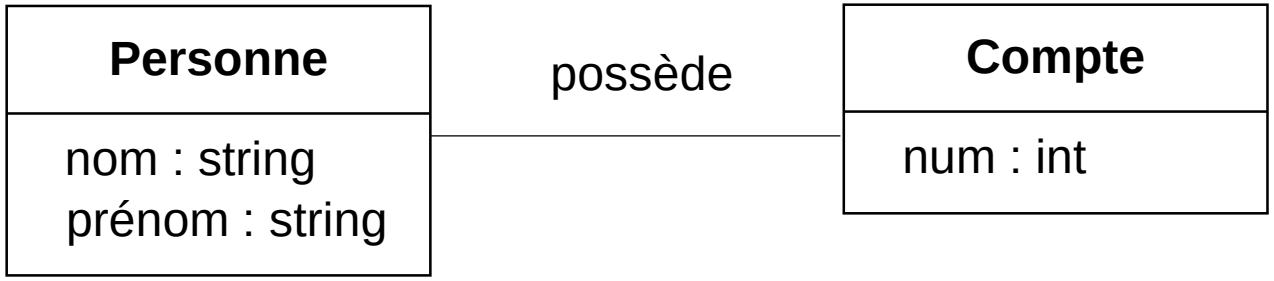
# Association

- Association
  - ✓ Une relation entre plusieurs classes (binaire en général)
  - ✓ Correspond à l'abstraction des liens qui existent entre les objets dans le monde réel
- Un lien
  - ✓ Une connexion physique ou conceptuelle entre des instances de classes



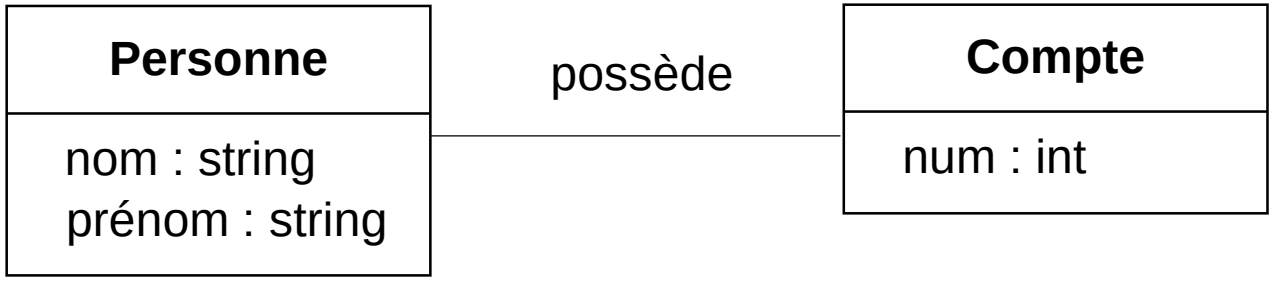

Instances d'associations : liens entre objets

# Questions



# Questions

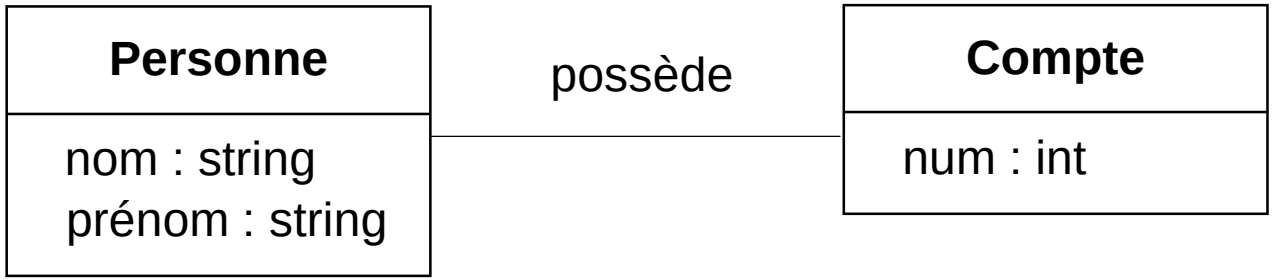
1) Combien d'instances d'une classe peuvent participer à une même association ?
2) Comment représenter "*une personne peut avoir plusieurs comptes, mais un compte n'appartient qu'à une seule personne*" ?
3) Comment implémenter cette association dans un langage comme Java ?
4) Comment une association UML se traduit-elle dans une base de données relationnelle ?
5) …



# Questions-Réponses

1) Combien d'instances d'une classe peuvent participer à une même association ?
2) Comment représenter "*une personne peut avoir plusieurs comptes, mais un compte n'appartient qu'à une seule personne*" ?

- **Réponse:**
  - ✓ Multiplicité (Rappel: Cardinalité)



# Questions-Réponses

3) Comment implémenter cette association dans un langage comme Java ?

- **Réponse:**
  - ✓ Une association est souvent implémentée par un attribut (référence à l'autre classe) ou une collection.

# Implémentation

- Association bidirectionnelle
- Un attribut dans chaque classes: comptes et proprietaire

```
class Personne {
    private String nom;
    private String prenom;

    private List<Compte> comptes = new ArrayList<>();

    public void ajouterCompte(Compte c) {
        comptes.add(c);
        c.setProprietaire(this);
    }
}
class Compte {
    private String numero;

    private Personne proprietaire;

    public void setProprietaire(Personne p) {
        this.proprietaire = p;
    }
}
```

D'où viennent les noms comptes et proprietaire ?

# Questions-Réponses

4) Comment une association UML se traduit-elle dans une base de données relationnelle ?

- **Réponse:**
  - ✓ Une association UML correspond à une relation entre tables (via clé étrangère(s) ou table d'association).

```
CREATE TABLE Personne (
    id INT PRIMARY KEY,
    nom VARCHAR(50),
    prenom VARCHAR(50)
);
```

```
CREATE TABLE Compte (
    numero INT PRIMARY KEY,
    personne_id INT,
    FOREIGN KEY (personne_id)
            REFERENCES   Personne(id)
);
```

**Table Compte**

| numéro | Solde (€) | personne_id |
|---|---|---|
| 2001 | 5700.00 | 501 |
| 2002 | 13600.00 | 501 |
| 2003 | 500.00 | 502 |

**Table Personne**

| id | nom | prenom |
|---|---|---|
| 501 | Dupont | Alice |
| 5O2 | Martin | Karim |

# Multiplicité

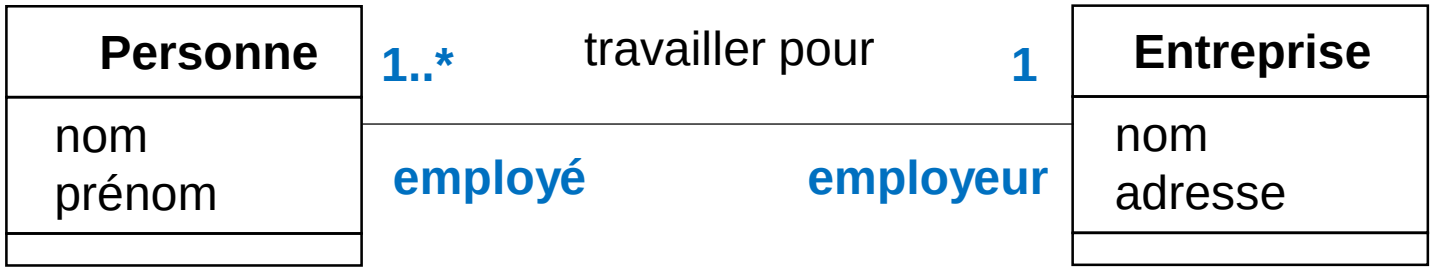
- Précise le nombre possible d'instances reliées à l'autre instance dans la relation

| | | |
|---|---|---|
| n | Exactement n | n: entier naturel , n> 0 |
| n..m | De n à "m" | n, m: entiers naturels ou variables, m > n |
| * | Plusieurs (0 ou plus) | équivalent à 0..* |
| n..* | Au moins n | n, entier naturel ou variable |


- ✓ Un Employé est affecté à un d'un seul Service
- ✓ A un Service, on peut affecter d'un ou plusieurs Employé (s)

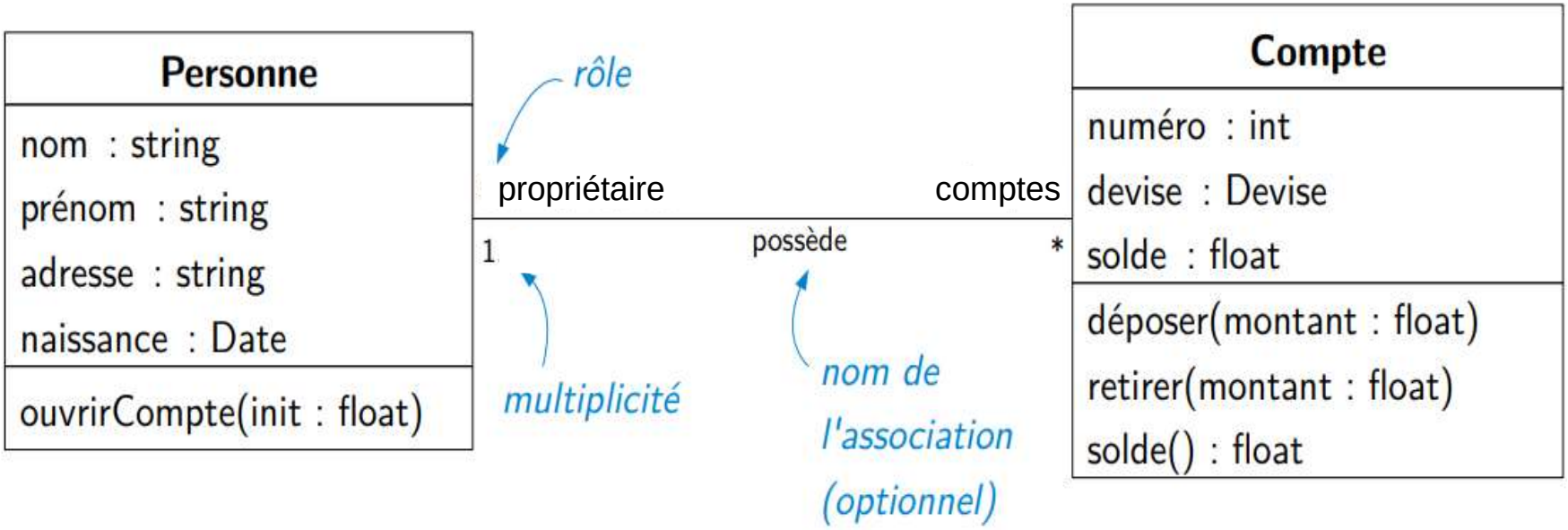
# Rôle

- Il est possible de préciser le rôle d'une classe au sein d'une association
- Décrit comment une classe voit une autre classe au travers d'une association
- Prend tout son intérêt lorsque plusieurs associations existent entre 2 classes



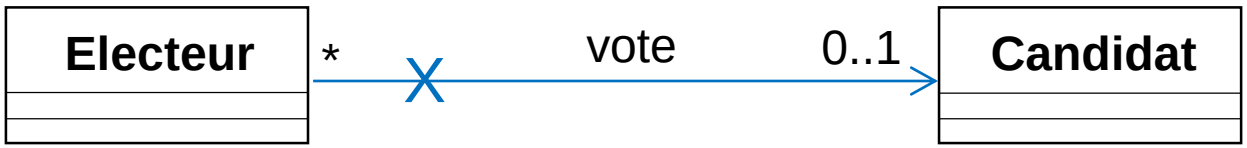

✓ L'entreprise est l'employeur des personnes qui travaillent pour elle
✓ Une personne a un statut d'employé dans l'entreprise

# Exemple


Personne
nom : string
prénom : string
adresse : string
naissance : Date
ouvrirCompte(init : float)
rôle
propriétaire
comptes
1
possède
*
multiplicité
nom de l'association (optionnel)
Compte
numéro : int
devise : Devise
solde : float
déposer(montant : float)
retirer(montant : float)
solde() : float

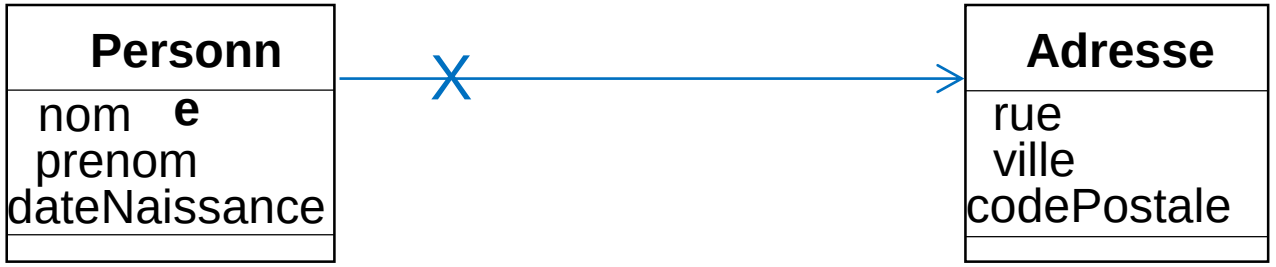
# Navigabilité

- Indique si l'association fonctionne de manière unidirectionnelle ou bidirectionnelle
- Une association est navigable dans un sens signifie que les instances de la classe cible peuvent être atteints par les instances de la classe source
  - ✓ Cela suppose que source possède un attribut qui fait référence à la classe cible




- Par défaut les associations possèdent une navigabilité bidirectionnelle

# Navigabilité – autre exemple



# Question

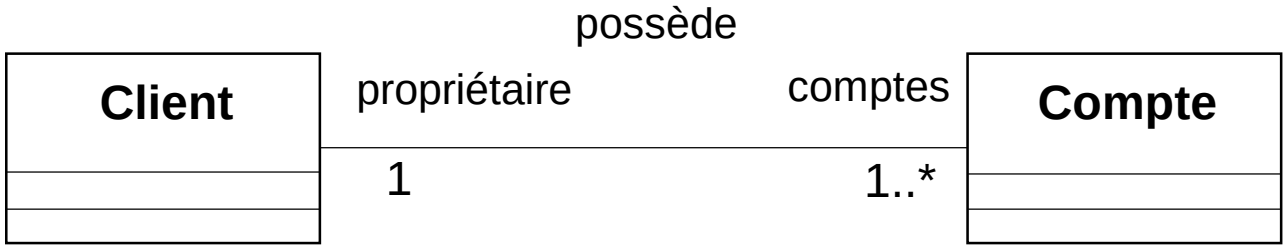
Modéliser les descriptions suivantes :

Dans une librairie en ligne, un internaute crée un compte pour devenir client. Il peut créer plusieurs comptes.

# Réponse

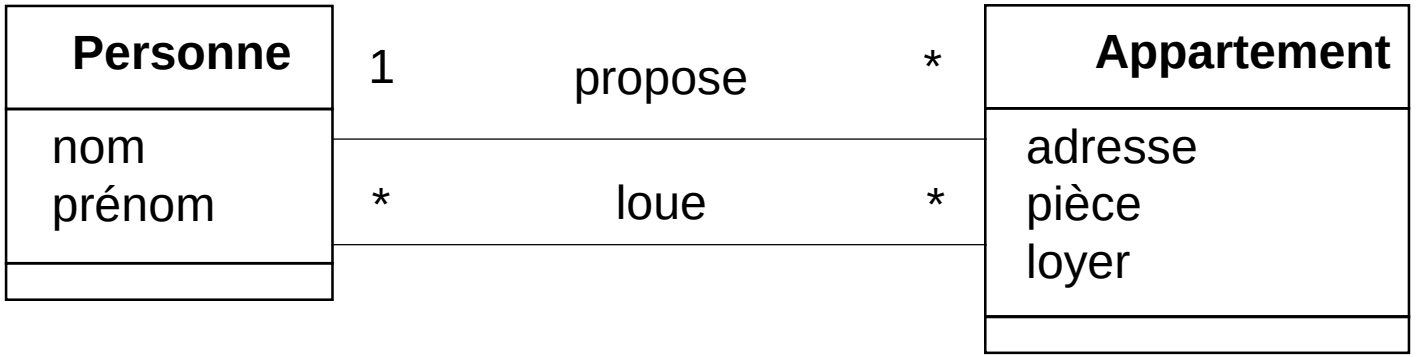
Modéliser les descriptions suivantes :

Dans une librairie en ligne, un internaute crée un compte pour devenir client. Il peut créer plusieurs comptes.



# Remarques

- Les associations sont spécifiquement destinées à représenter des relations durables entre des classes
- Il sont souvent utilisées pour représenter les attributs de la classes
- Plusieurs associations peuvent exister entre 2 classes



# Questions