Conception UML

Diagramme de classe

Rania OTHMAN

Sommaire

Sommaire

UML : Unified Modeling Language ? I

Langage graphique pour visualiser, spécifier, construire et documenter un **logiciel**

- Langage
  - Syntaxe et règles d'écriture
  - Notations graphiques normalisées.
- Modélisation
  - Abstraction du fonctionnement et de la structure du système
  - Spécification et conception
- Unifié
  - Fusion de plusieurs notations : Booch, OMT, OOSE
  - Standard défini par l'OMG (Object Management Group)
  - Dernière version : UML 2.5.1

UML : Unified Modeling Language ? I

1. UML est un langage universel de modélisation objet ... pas une méthode
2. Différence Langage & Méthode
   ⇒ Langage de modélisation = notations, grammaire, sémantique
   ⇒ Méthode = comment utiliser le langage de modélisation (recueil des besoins, analyse, conception, mise en oeuvre, validation, ...)

Différents modèles UML I

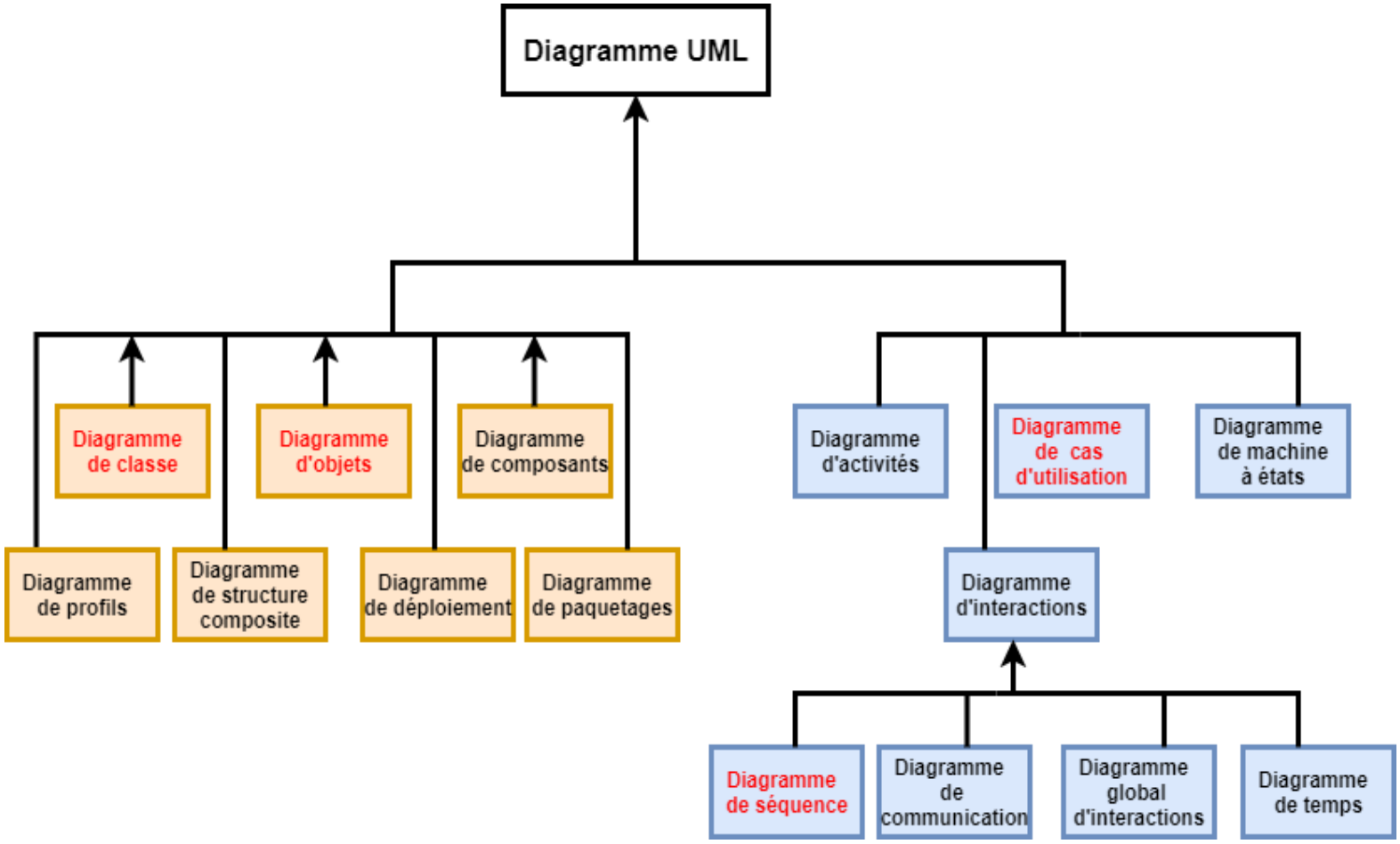
UML peut être utilisé pour définir de nombreux modèles :

1. Modèles descriptifs vs prescriptifs
2. Modèles destinés à différents acteurs
   - Pour l'utilisateur ; Décrire le quoi
   - Pour les concepteurs/développeurs ; Décrire le comment
3. Modèles statiques vs dynamiques
   - Statiques : Décrire les aspects structurels
   - Dynamiques : Décrire comportements et interactions

Les modèles sont décrits par des diagrammes (des graphes) ; Chaque diagramme donne un point de vue différent sur le système.

Diagrammes de UML I

UML 2.5 est composé de 14 diagrammes : Diagrammes statiques ou structurels et Diagrammes dynamiques ou comportementaux



Qu'est-ce qu'un objet ? I

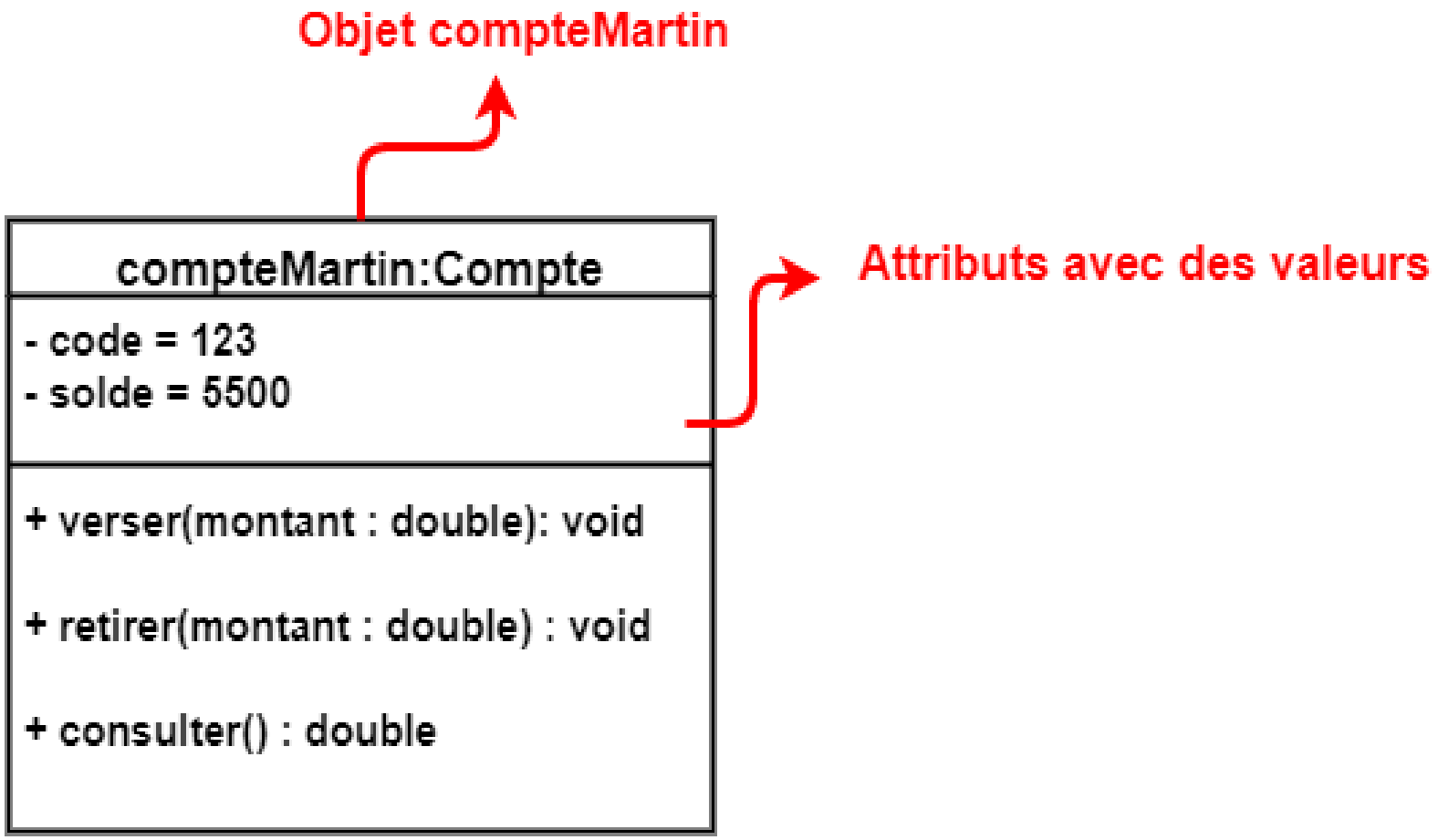
Objet = Etat + Comportement + Identité

1. Etat d'un objet : Ensemble de valeurs décrivant l'objet ;
   ⇒ Chaque valeur est associée à un attribut (propriété)
2. Comportement d'un objet : Ensemble d'opérations que l'objet peut effectuer

3. Identité d'un objet : Permet de distinguer les objets indépendamment de leur état

   ⇒ Attribuée implicitement à la création de l'objet
   ⇒ 2 objets différents peuvent avoir le même état

Exemple d'Objet I



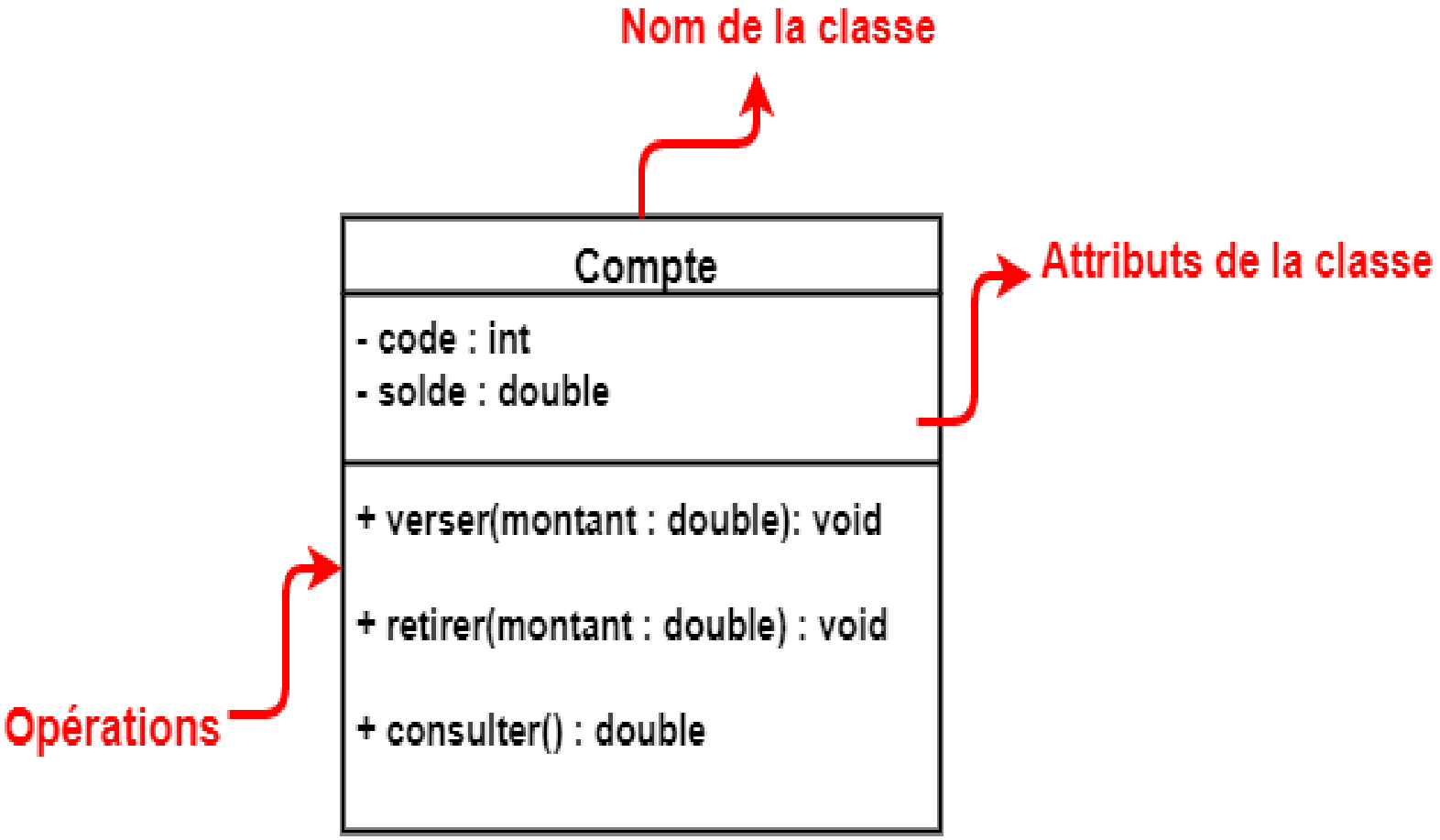
Q'est ce qu'une classe ? I

Une classe est un :

1. Regroupement d'objets ayant le même comportement
   ⇒ même attributs
   ⇒ même opérations
2. Abstraction décrivant les propriétés communes des objets qui en sont des instances.

1. Une classe peut décrire une infinité d'instances
2. Un objet sait toujours à quelle classe il appartient

Exemple de classe I



Classe vs Objet I

1. La structure d'une classe est constante
2. La valeur des attributs des objets peut changer pendant l'exécution
3. Des objets peuvent être ajoutés ou détruits pendant l'exécution
4. Deux objets dont les attributs ont les mêmes valeurs sont distincts.

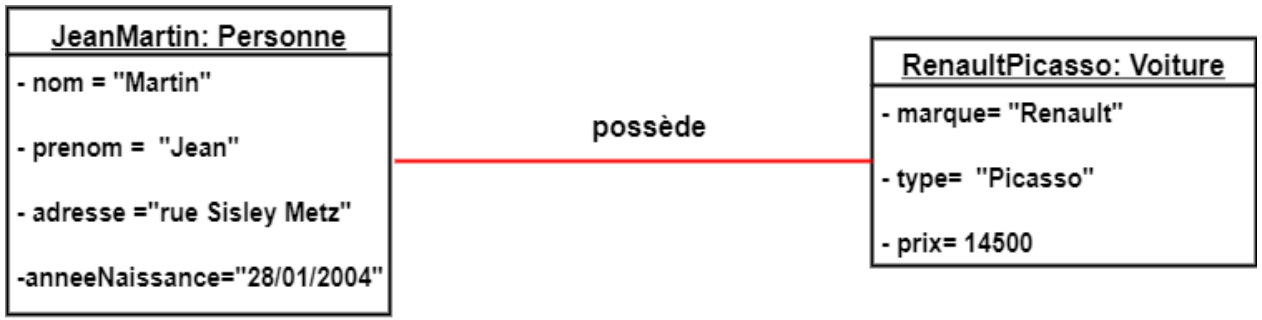
Diagrammes de classes et d'objets I

1. Diagramme de classes
   ⇒ Représentation de la structure interne du logiciel
   ⇒ Utilisé surtout en conception mais peut être utilisé en analyse
2. 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

Relations entre objets I

Lien entre objets

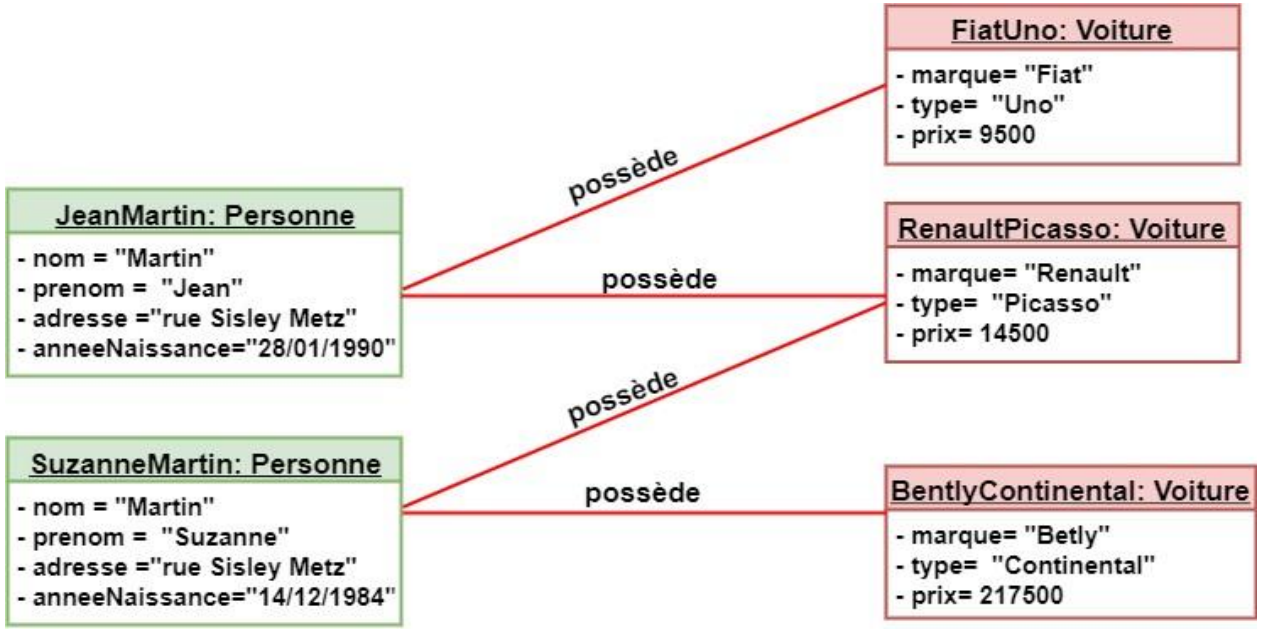
- relation binaire (en général)
- au plus un lien entre deux objets (pour une association)



Relations entre objets I

Lien entre objets

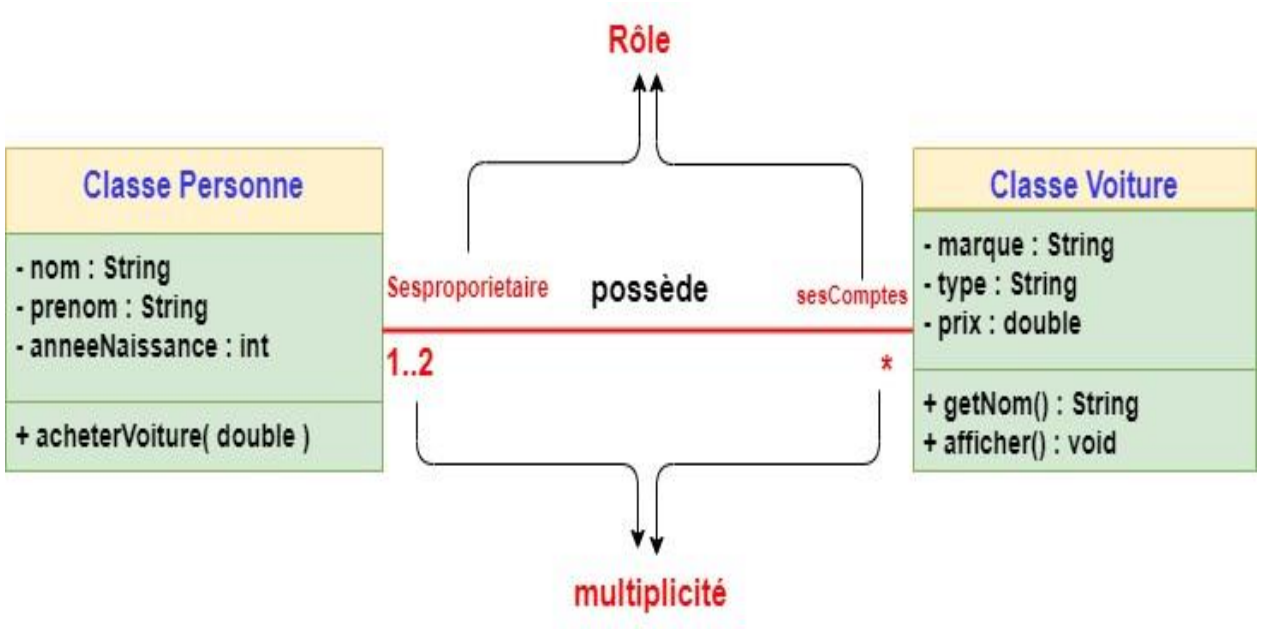
- relation binaire (en général)
- au plus un lien entre deux objets (pour une association)



Relations entre classe I

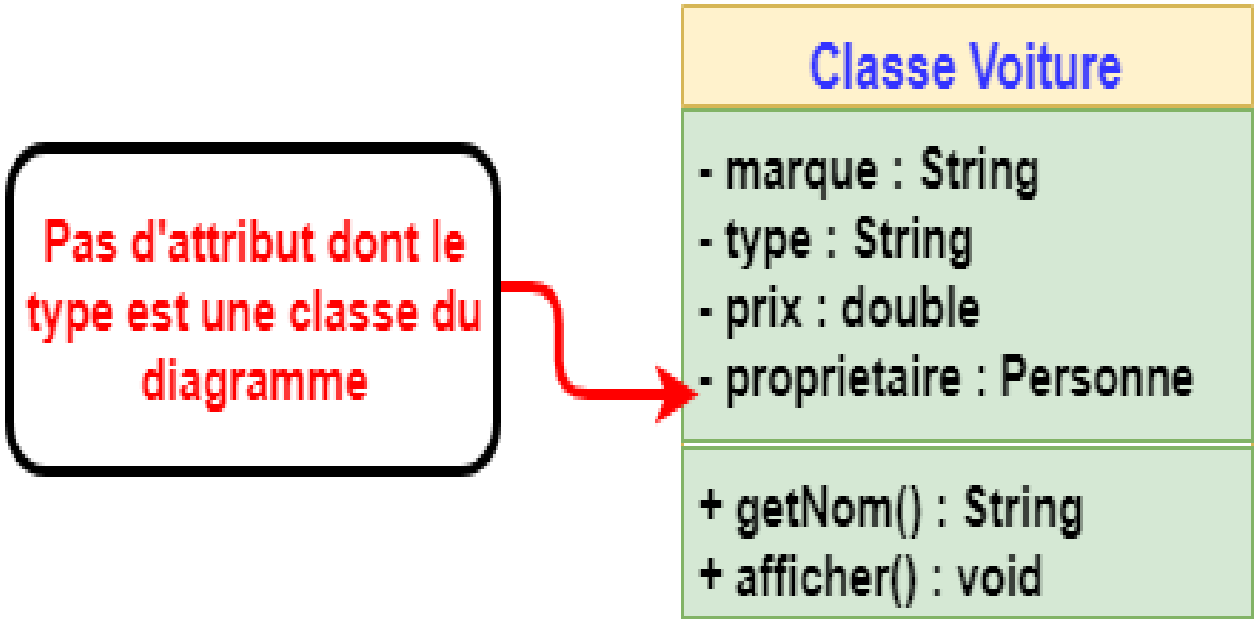
Association entre classes : Relation binaire (en général)

- Rôle : Nomme l'extrémité d'une association, permet d'accéder aux objets liés par l'association à un objet donné.
- Multiplicité : Contraint le nombre d'objets liés par l'association



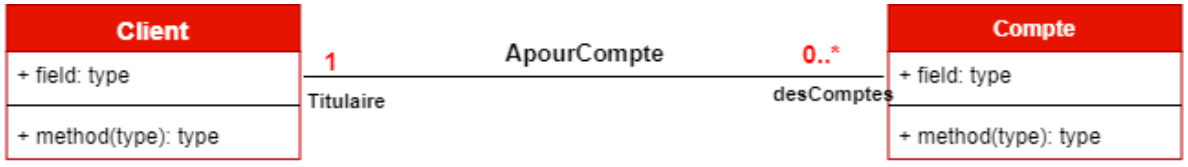
Attribut et association I

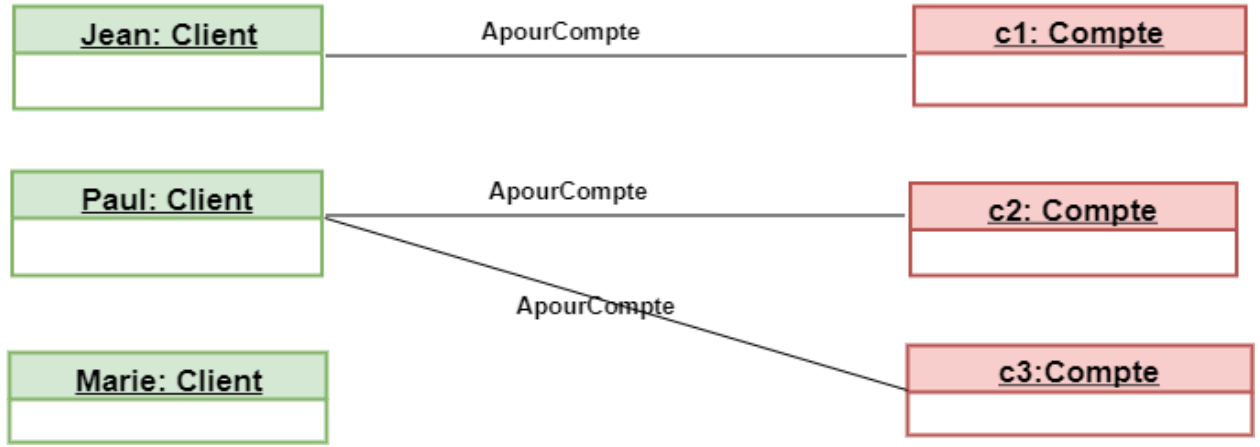
- Types des attributs simple, primitif ou énuméré
- il n y a pas d'attribut dont le type est une classe du diagramme



Cardinalités d'une association I

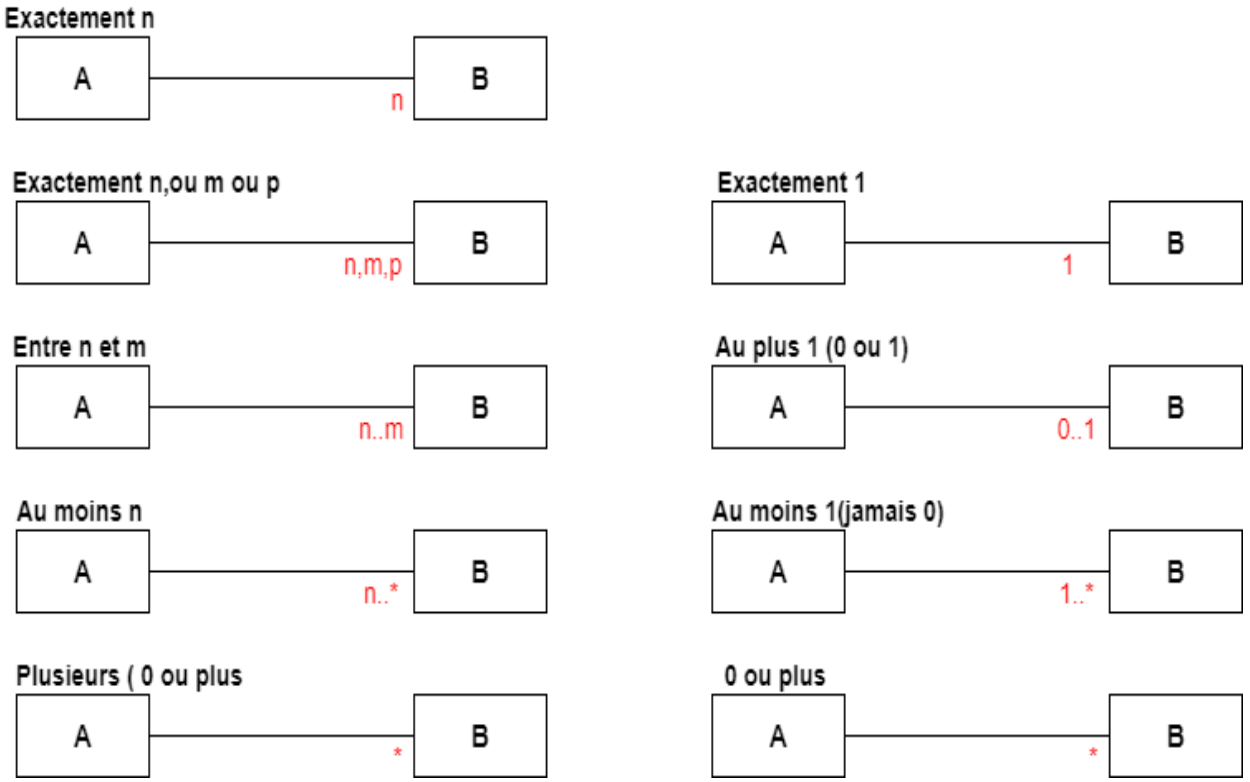
- Précise combien d'objets peuvent être liés à un seul objet source.
- Cardinalité minimale et cardinalité maximale (min..max)
- Doivent être des constantes.





Multiplicités I

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



Association vs. Liens I

Une association

- lie deux classes
- décrit un ensemble de liens

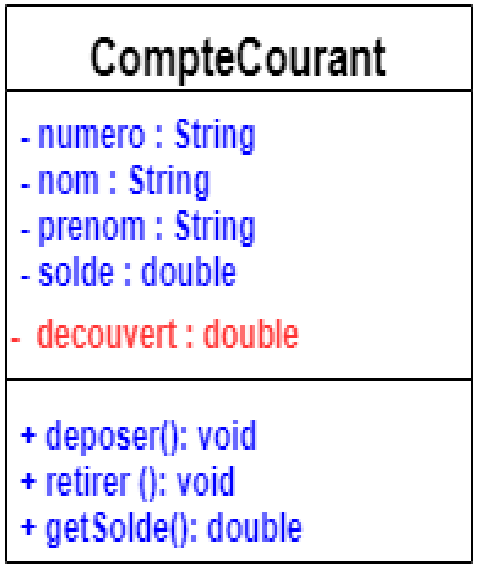
Un lien

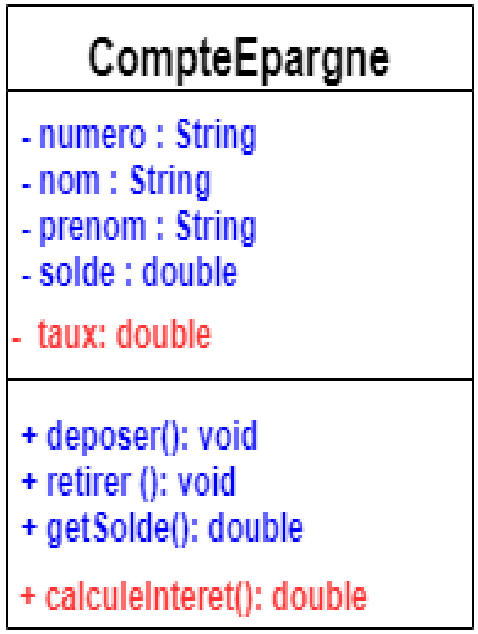
- lie deux objets
- Un lien est une instance d'association
- des liens peuvent être ajoutés ou détruits pendant l'exécution, (ce n'est pas le cas des associations)

Hiérarchie de classes I

Principe : Regrouper les classes partageant des attributs et des opérations et les organiser en arborescence

- Spécialisation : raffinement d'une classe en une sous-classe
- Généralisation : abstraction d'un ensemble de classes en super-classe



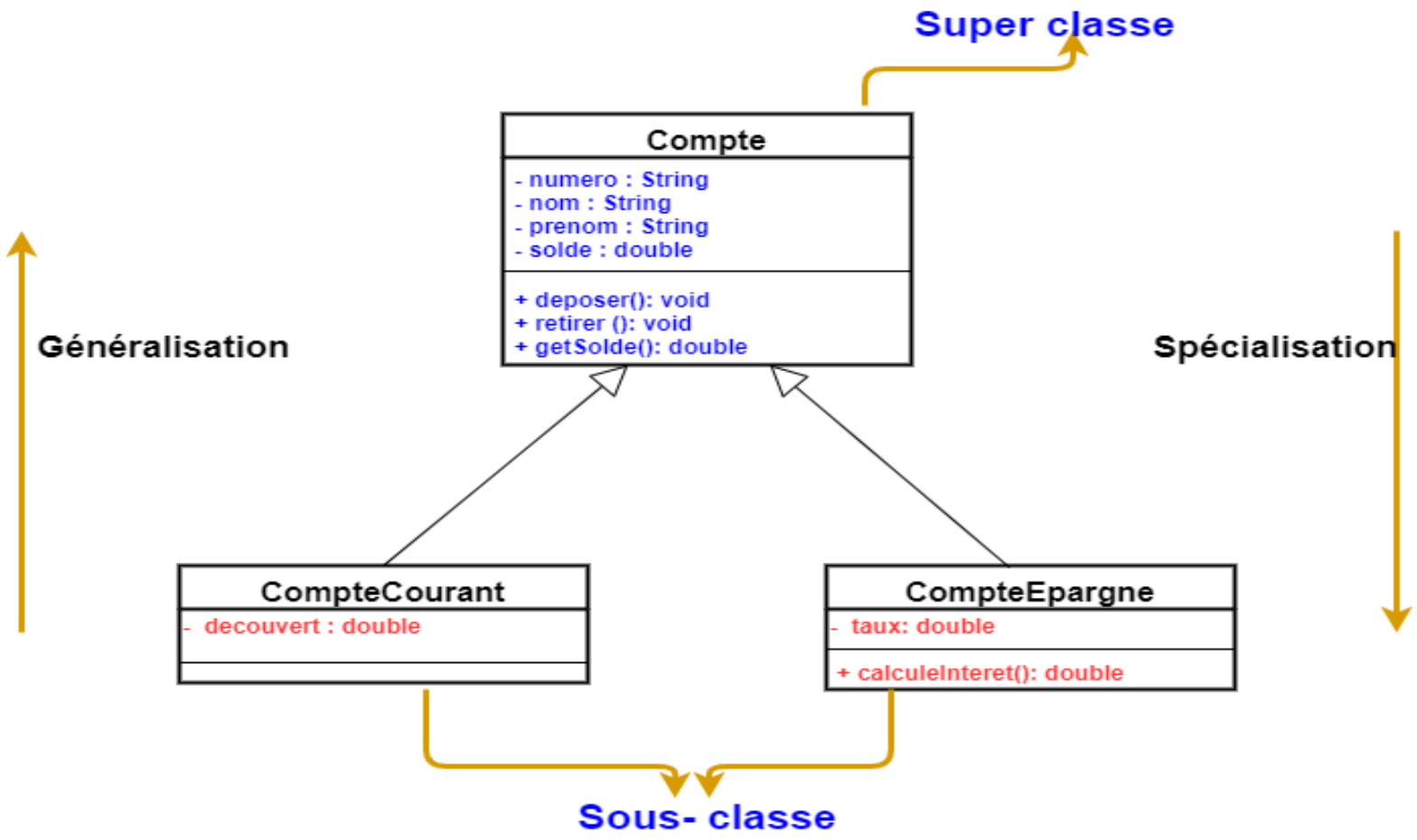


Hiérarchie de classes I

Principe : Regrouper les classes partageant des attributs et des opérations et les organiser en arborescence

- la classe Compte : est la super-classe (classe mère)
- la classe CompteCourant hérite de la classe Compte
- la classe CompteEpargne hérite de la classe Compte

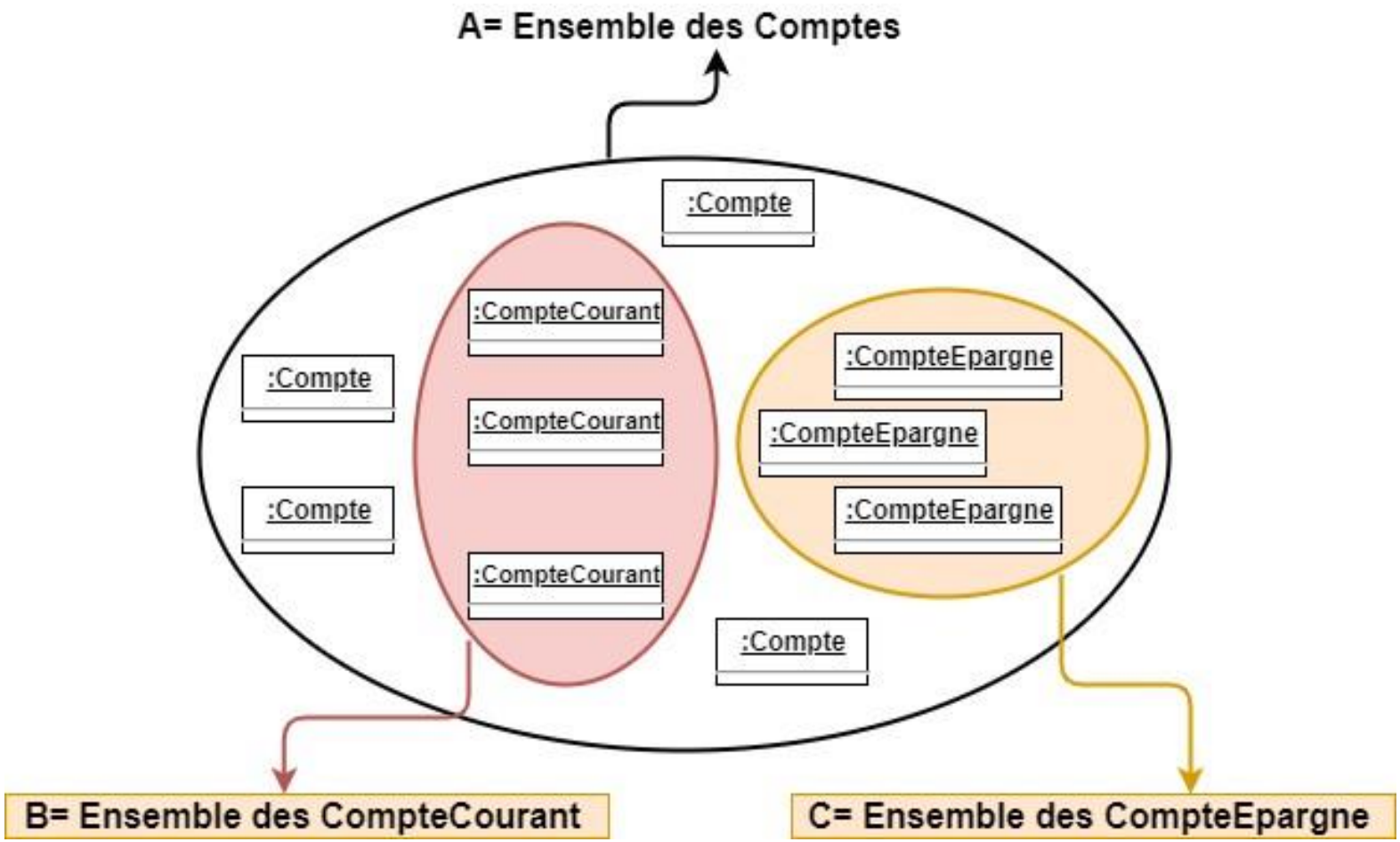
Hiérarchie de classes II



Objet et Héritage I

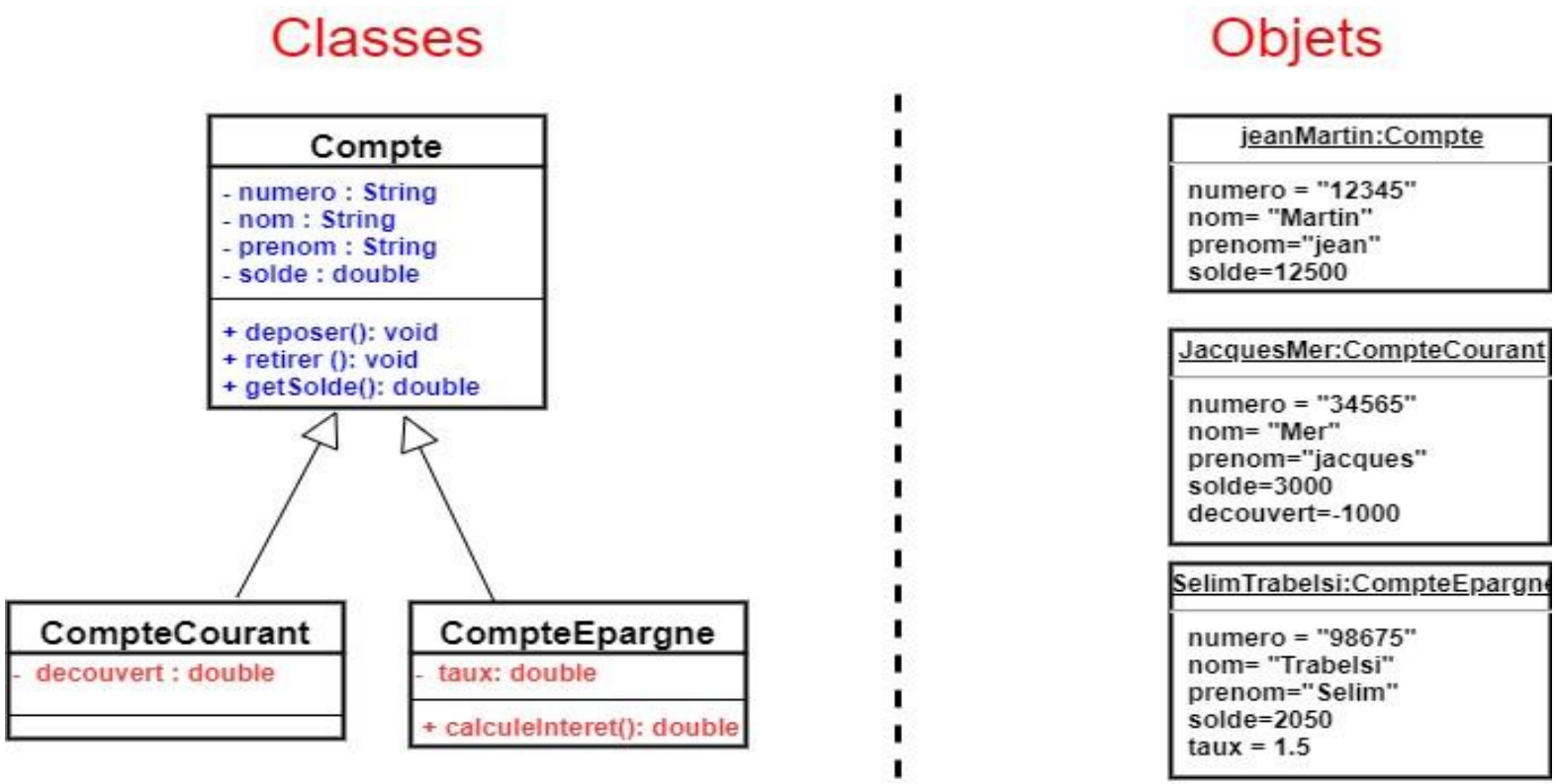
- L'ensemble des objets CompteCourant est inclus dans l'ensemble des objets de la classe Compte
- L'ensemble des objets CompteEpargne est inclus dans l'ensemble des objets de la classe Compte

Objet et Héritage II



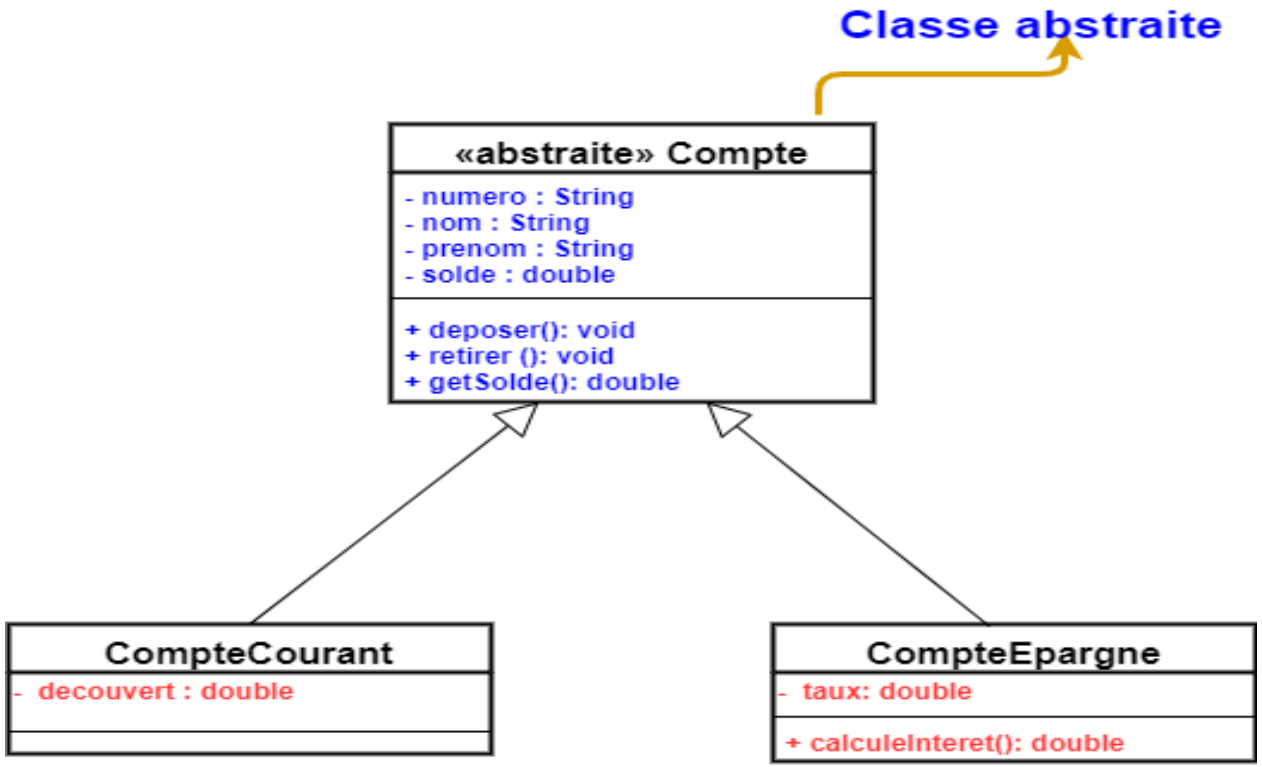
Objet et Héritage I

On peut instancier des objets de Compte, CompteCourant et CompteEpargne



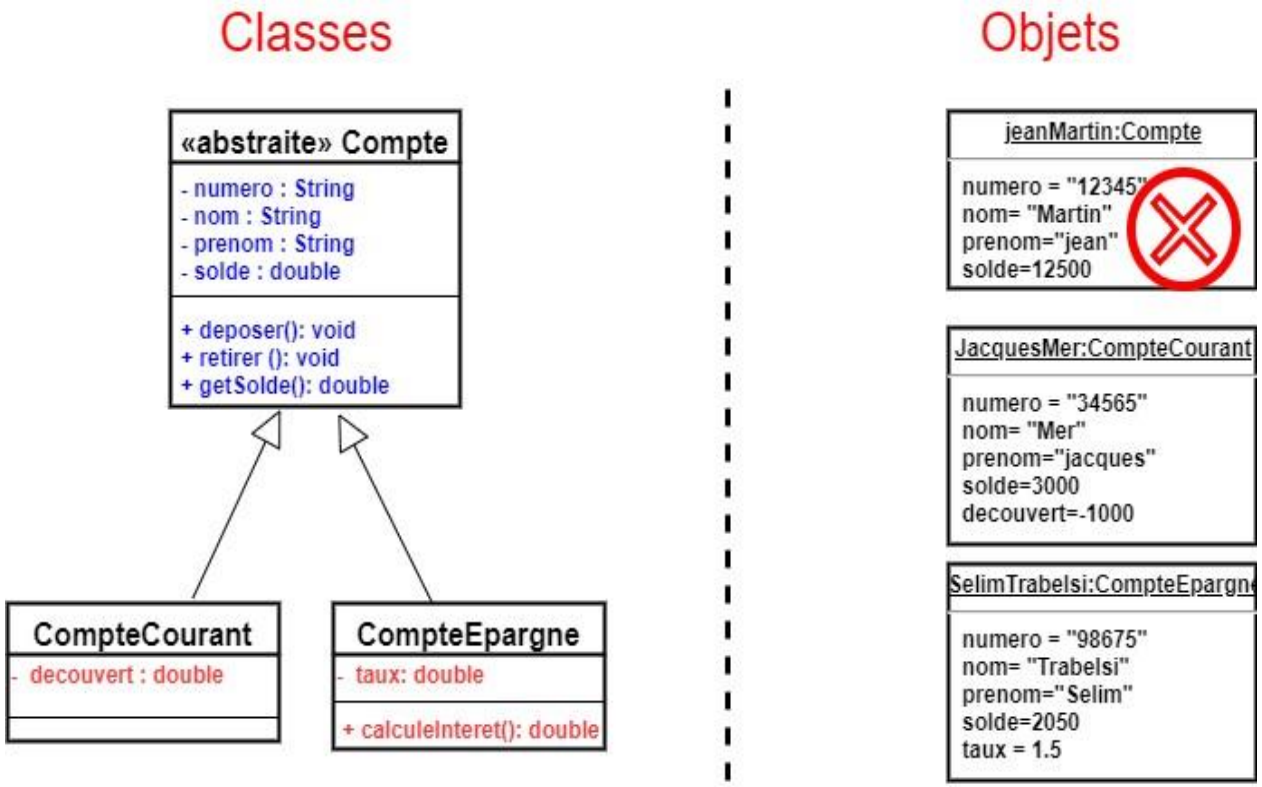
Classe abstraite I

- Elle sert de classe de base pour l'héritage

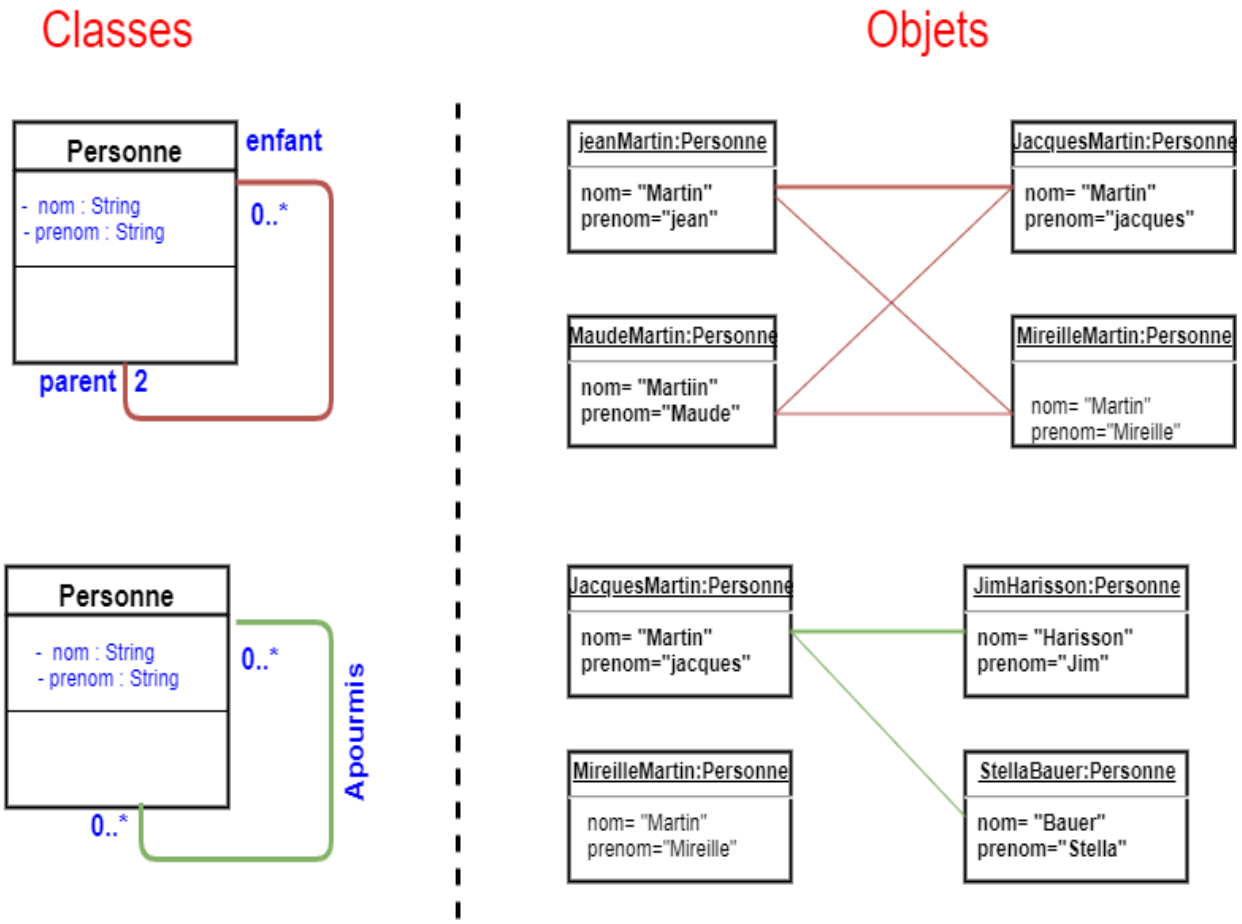


Classe abstraite I

- On ne peut instancier un objet de la classe abstraite Compte.

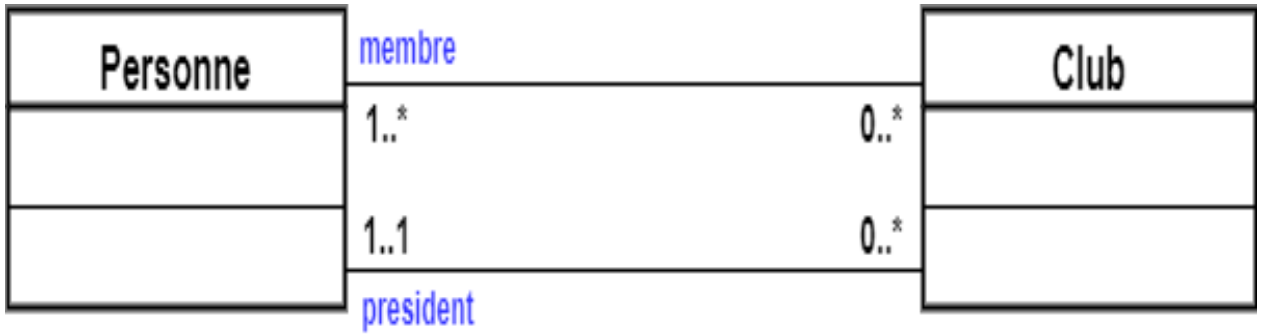


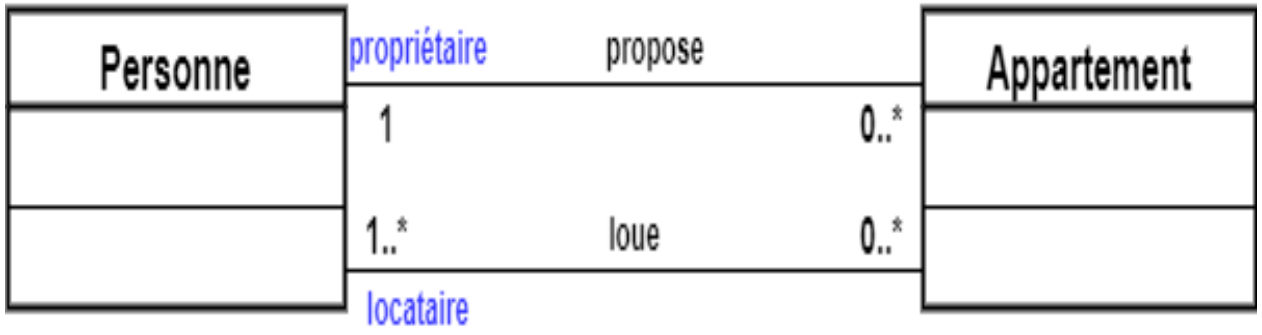
Association réflexive I



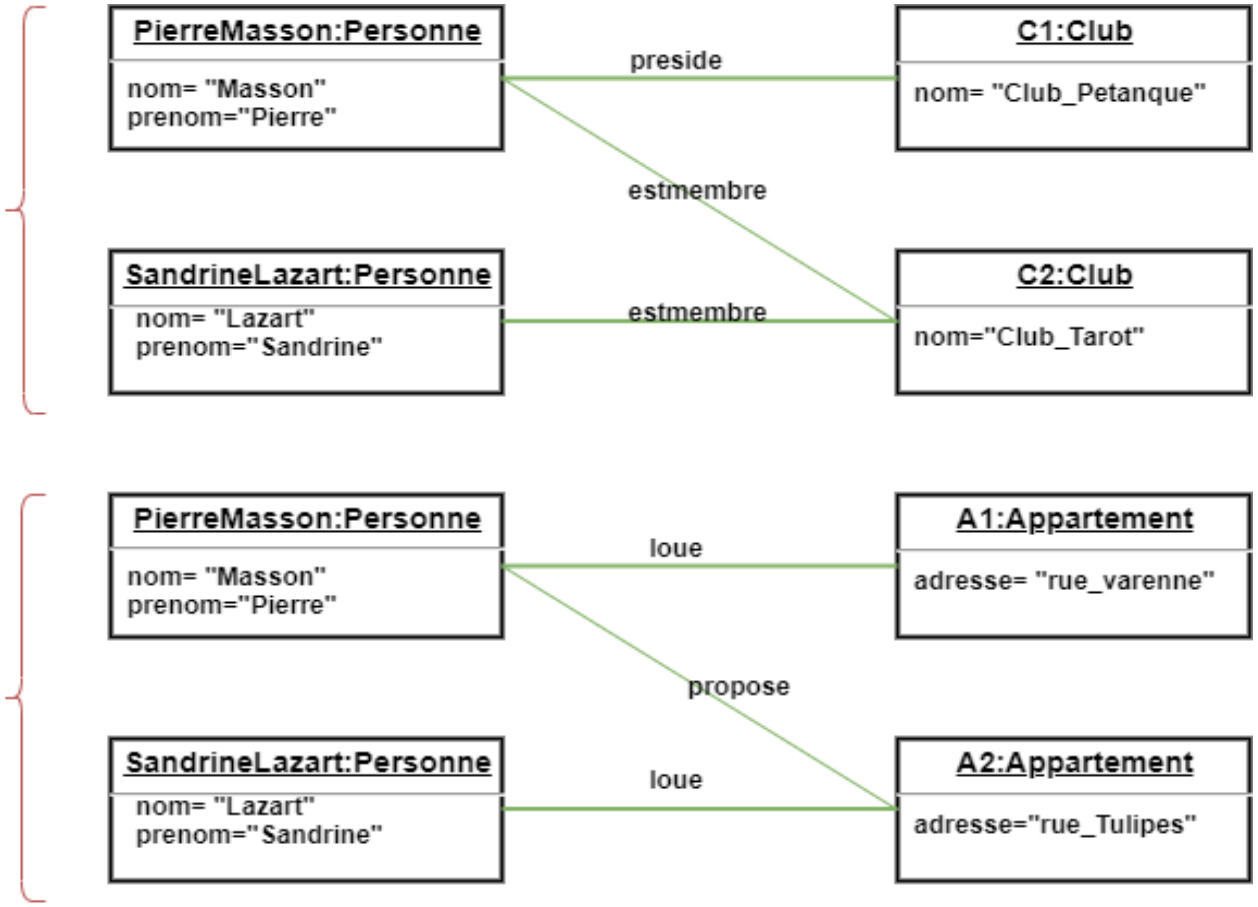
Association multiple I

- On peut avoir plusieurs associations entre deux classes.



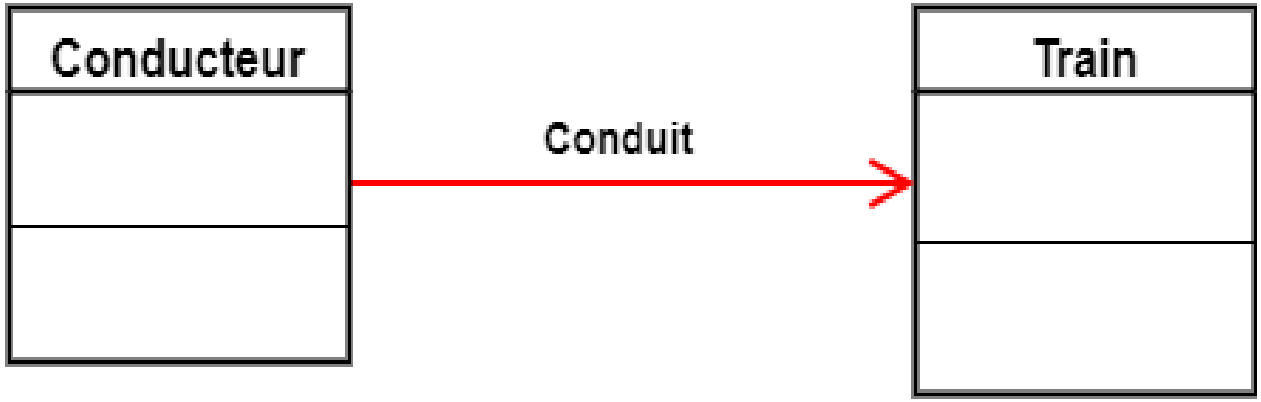


Association multiple et diagramme des objets I



Navigabilité I

- Par défaut, une association est navigable dans les deux sens.
- Cependant, on peut restreindre L'accèssibilité aux objets d'une classe.



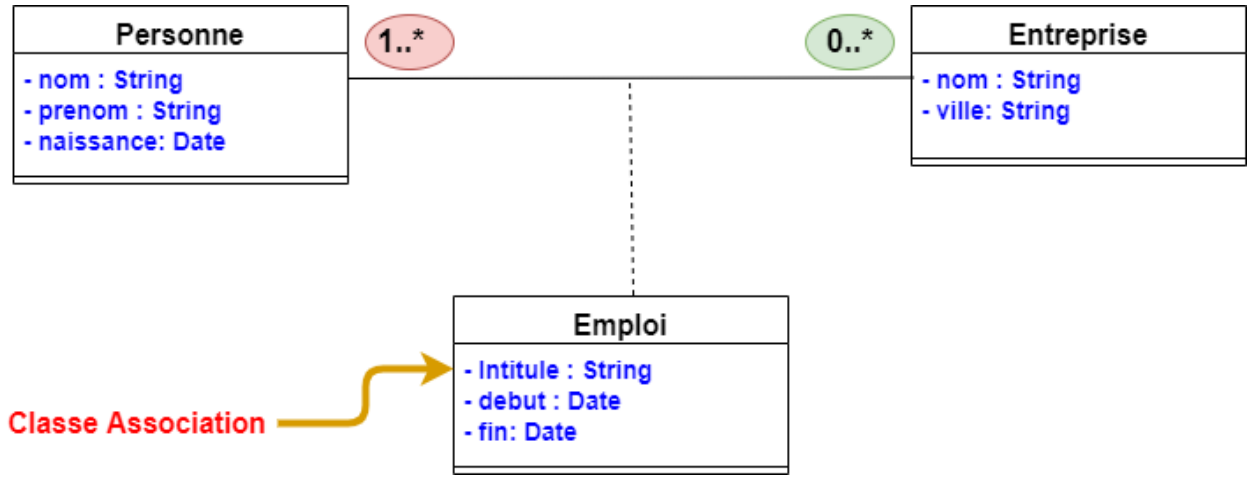
- On peut considérer que le train n'a pas besoin de "connaître" son conducteur

Visibilité des attributs I

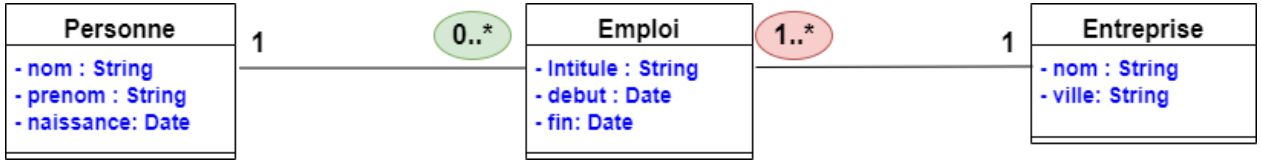
1. visibilité
   - — privé : accessible par les opérations de la classe
   - + public : accessible en dehors de la classe
   - # protected : accessible par les classes qui héritent.
2. Attribut de classe
   - La portée d'un attribut est réduite à l'objet.
   - Un attribut de classe est un attribut commun à tous les objets de cette classe (notation : souligné ou $)

Classe Association I

- Objectif : paramétrer une association entre deux classes par une classe



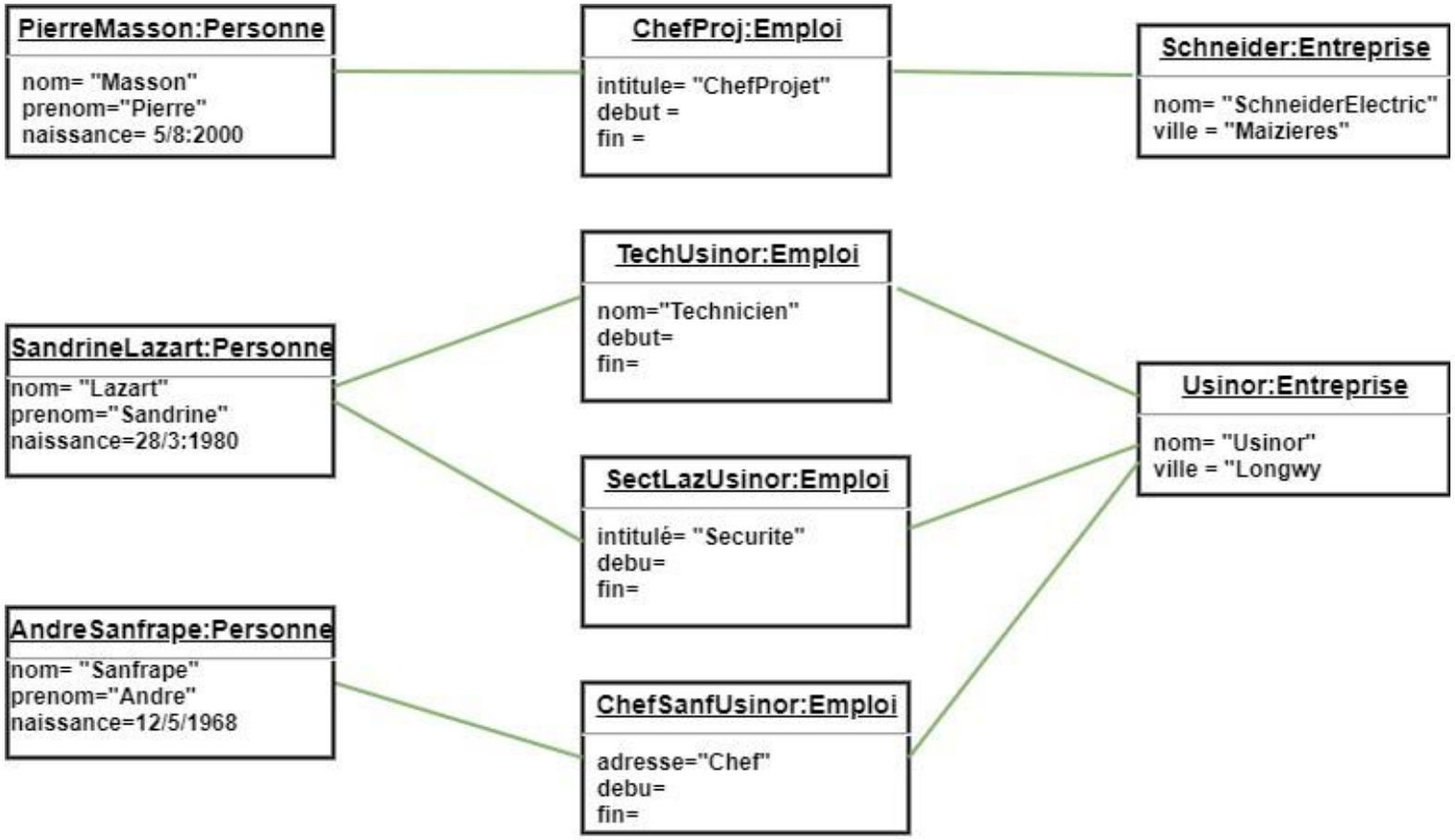
- Equivalence en termes de classes et d'associations :



- Instance unique de la classe-association pour chaque lien entre objets

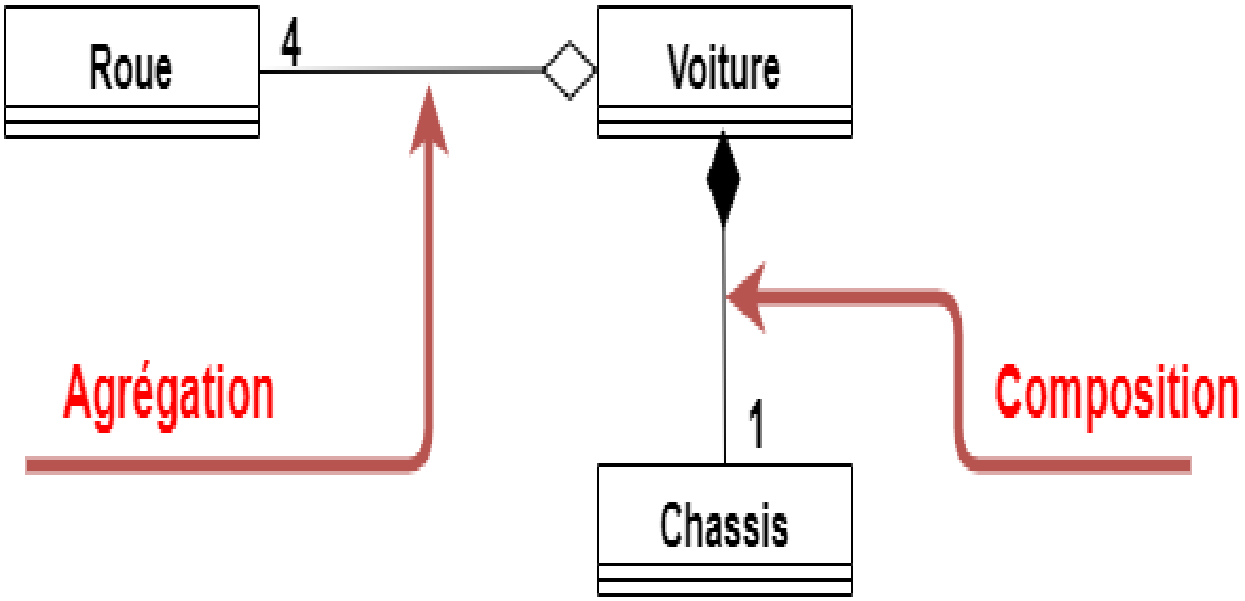
Classe Association I

Instance unique de la classe-association pour chaque lien entre objets



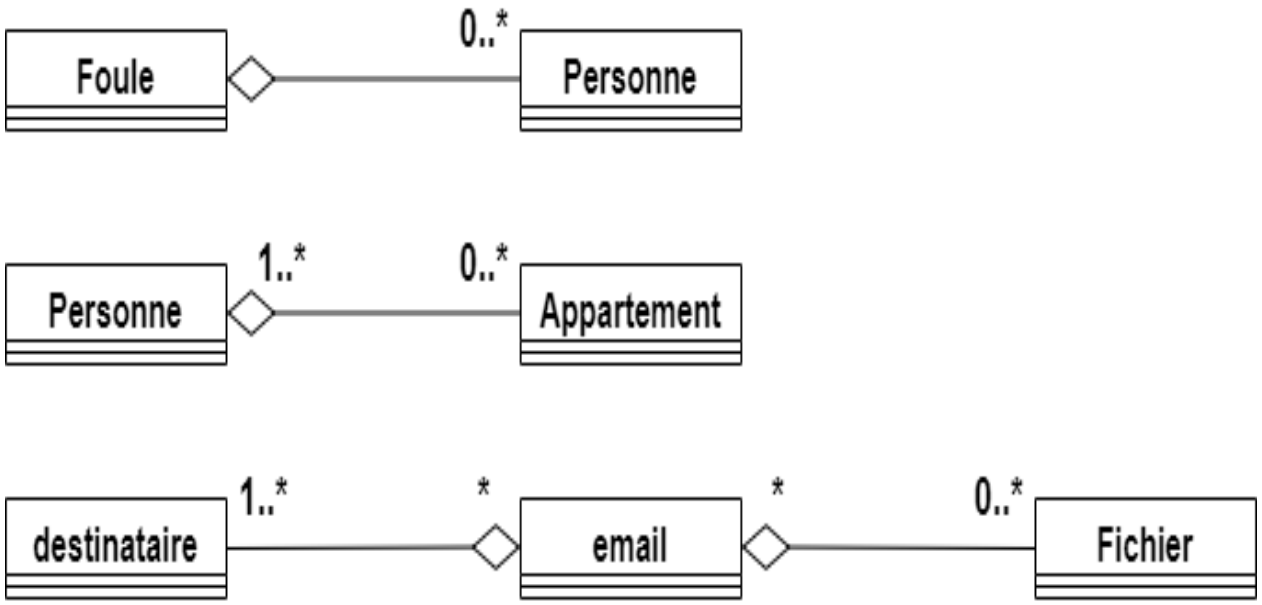
Agrégation I

- Association qui lie 2 classes, une classe prédomine l'autre
- Une relation de type composant-composite
- Il existe deux types d'agrégations : agrégation faible ou agrégation forte (composition).



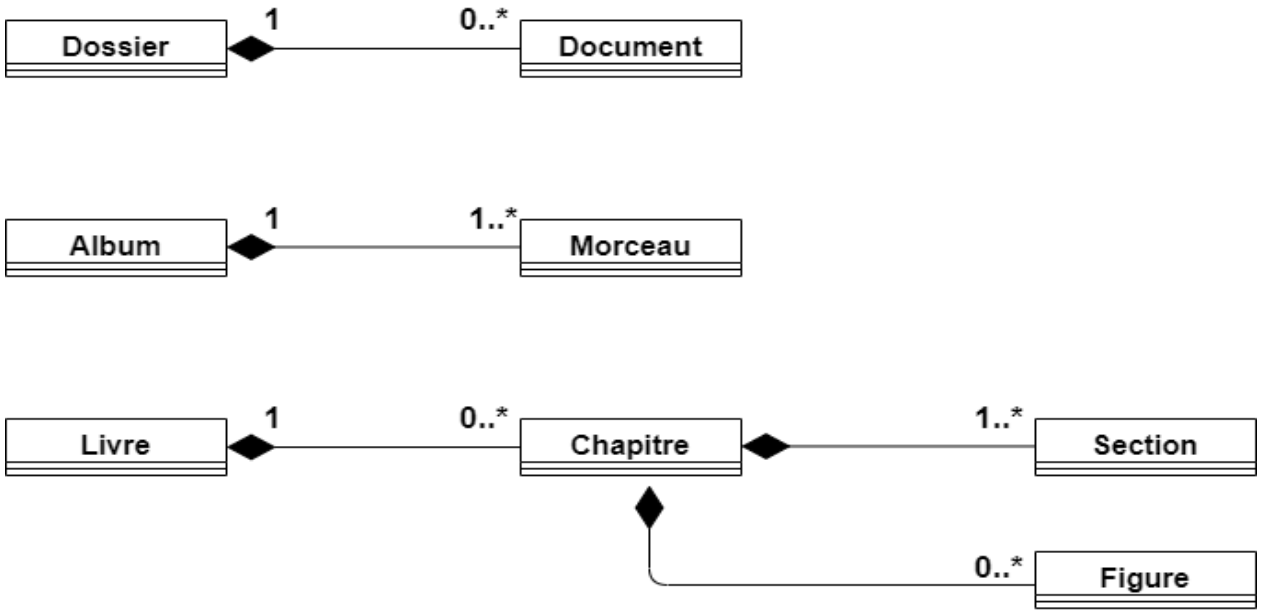
Agrégation faible I

1. Le composite fait référence à ses composants
2. La création ou destruction du composite est indépendante de la création ou destruction de ses composants
3. Un objet peut faire partie de plusieurs composites à la fois



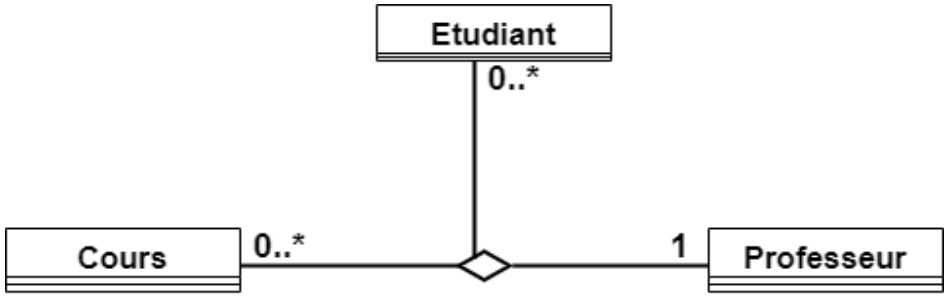
Composition I

1. Le composite contient ses composants
2. La création ou destruction du composite entraîne la création ou destruction de ses composants
3. Un objet ne fait partie que d'un composite à la fois



Association n-aire I

1. Association entre au moins trois classes
2. Chaque instance de l'association est un tuple de valeurs provenant chacune de leurs classes respectives



- Pour une paire (cours, étudiant), il n'existe qu'un professeur.
- Pour une paire (étudiant, professeur), il existe plusieurs cours.
- Pour une paire (cours, professeur), il existe plusieurs étudiants.