Haute École Léonard de Vinci
SCIENCES ET TECHNIQUES

UML

UE Analyse et modélisation

BINV2160

B. Lehmann

Le diagramme de classes
Le diagramme d’objets

Rappel : l'orienté Objet

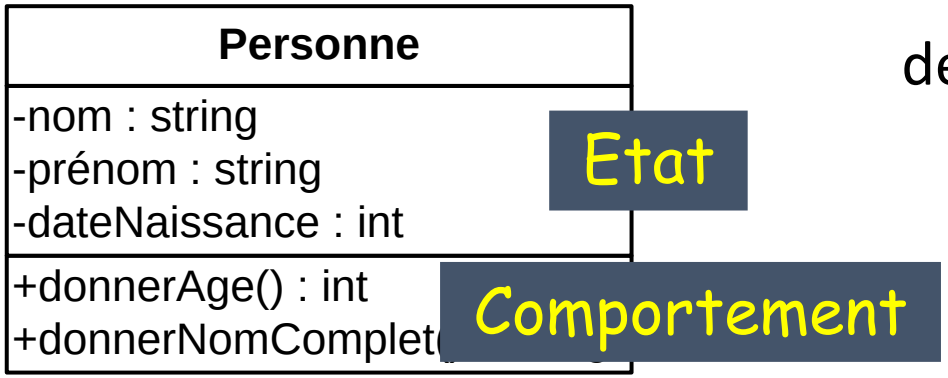
C'est l'approche de la conception de programmes qui tend à structurer les différentes parties d'un programme en objets ayant des **responsabilités** bien définies et interagissant entre eux pour les honorer.

Diagramme de classes

Notions

- Une **classe** représente un **groupe d'objets** possédant des états et un comportement communs.
- Un **objet** est une entité aux frontières bien définies, possédant une identité et encapsulant un état et un comportement.

Classe, attributs et méthodes



La **classe** `Personne`

définit 3 **attributs**:

- nom
- prénom
- dateNaissance

comporte 2 **méthodes**:

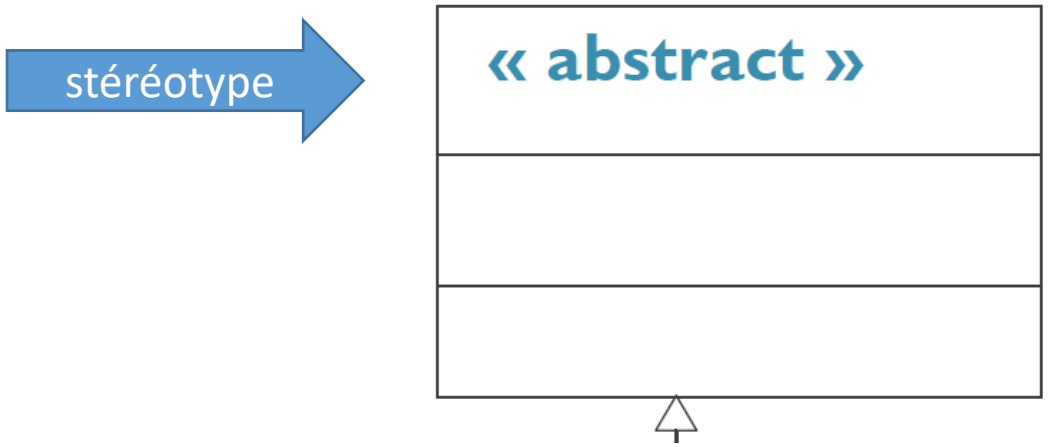
- donnerAge
- donnerNomComplet

Classe abstraite

- Une classe dont l'implémentation n'est pas complète.
- Une classe qui n'est pas destinée à être instanciée.
- Toute classe qui contient une méthode abstraite.

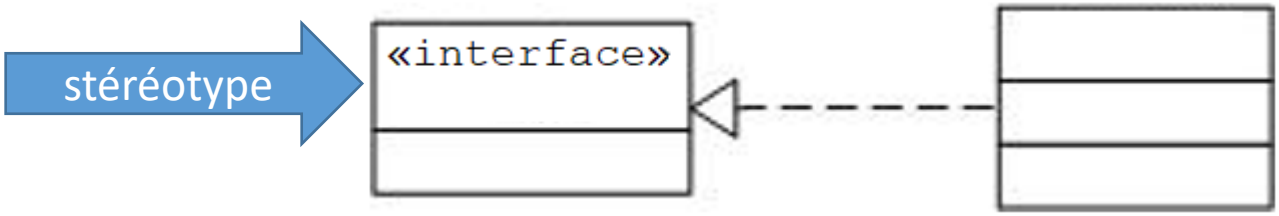
Il s'agit d'un mécanisme général d'extension du langage UML.

Les stéréotypes sont indiqués entre « ».



Interfaces

- Une interface est une classe dans laquelle **aucune méthode n'est implémentée**, et où les **champs ne sont pas indiqués**.
- L'interface et toutes ses méthodes sont abstraites.



Implémentation illustrée
par un **trait discontinu**

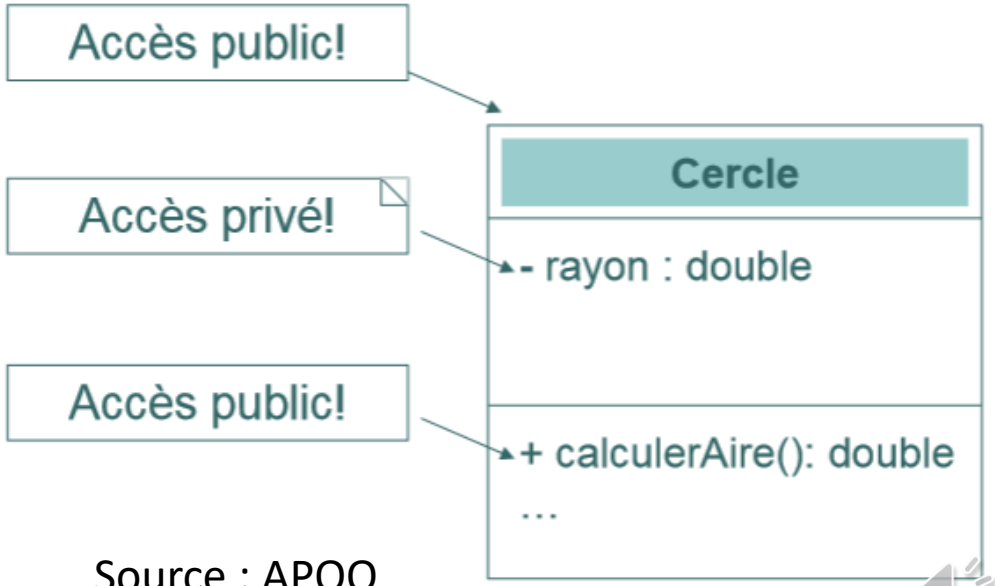
Visibilité

- Private -
- Package (default) ~
- Protected #
- Public +

Personne
-nom : string -prénom : string -dateNaissance : int
+donnerAge() : int +donnerNomComplet() : string

Rappel : encapsulation

- Une classe encapsule, c'est-à-dire regroupe et **protège** des propriétés et des comportements; les données protégées ne sont accessibles que par des méthodes qui sont rendues publiques.



Source : APOO

Attributs

- Attribut dérivé :

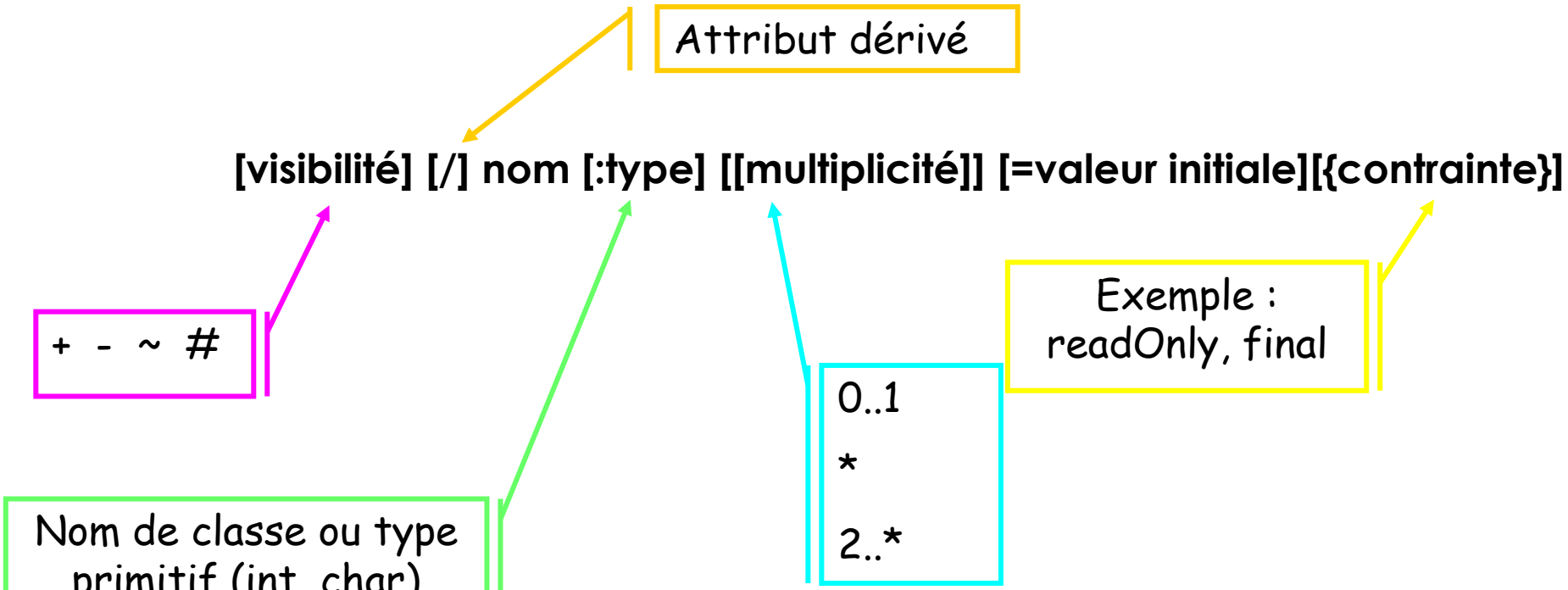
  / prixTTC (= prixHTVA*TVA)

- Attribut de classe :

  double TVA = 21;

  Valeur identique pour toutes les instances de la classe

Signature d'attribut



Exemples :

`+ tailleMoy:int`		Public
`# nom : String[0..10]`		Protected, tableau
`- prénom:String`		Private
`+ tailleMaximum:int = 2`	-	Public, static, valeur initiale

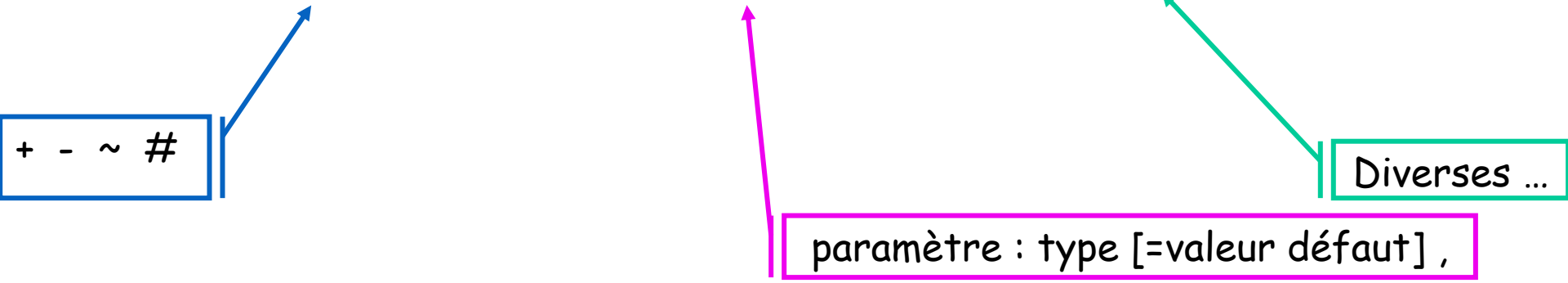
Méthode

- Méthode de classe:
  - soulignée()
- Méthode abstraite:
  - *enItalique()*

- 3 types d'opération:
  - **Getter et setter** : obtient/modifie la valeur d'un attribut
  - **Query** : renvoie une valeur sans changer **l'état observable** d'un objet.
    Autrement dit après l'exécution d'un query les valeurs renvoyées par l'ensemble des queries seront les mêmes.
  - **Modificateur** : change l'état observable d'un objet.

Signature de méthode

[visibilité] nom [(paramètres)][:type retour][{contrainte}]



Exemples :

```
+ getTaille():int;
- setNom(nom:String);
+ estValide(nom:String):boolean;
+ getTailleMaximum():int
```

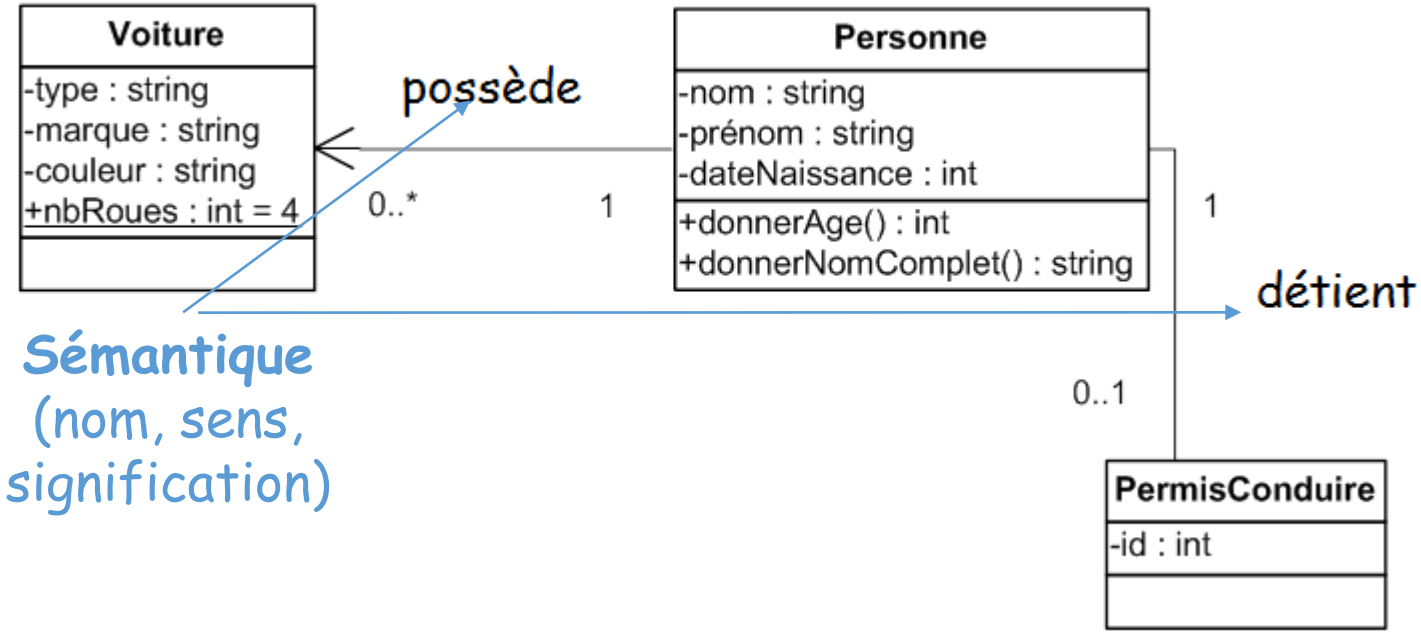
Association entre classes

Une association a des caractéristiques :

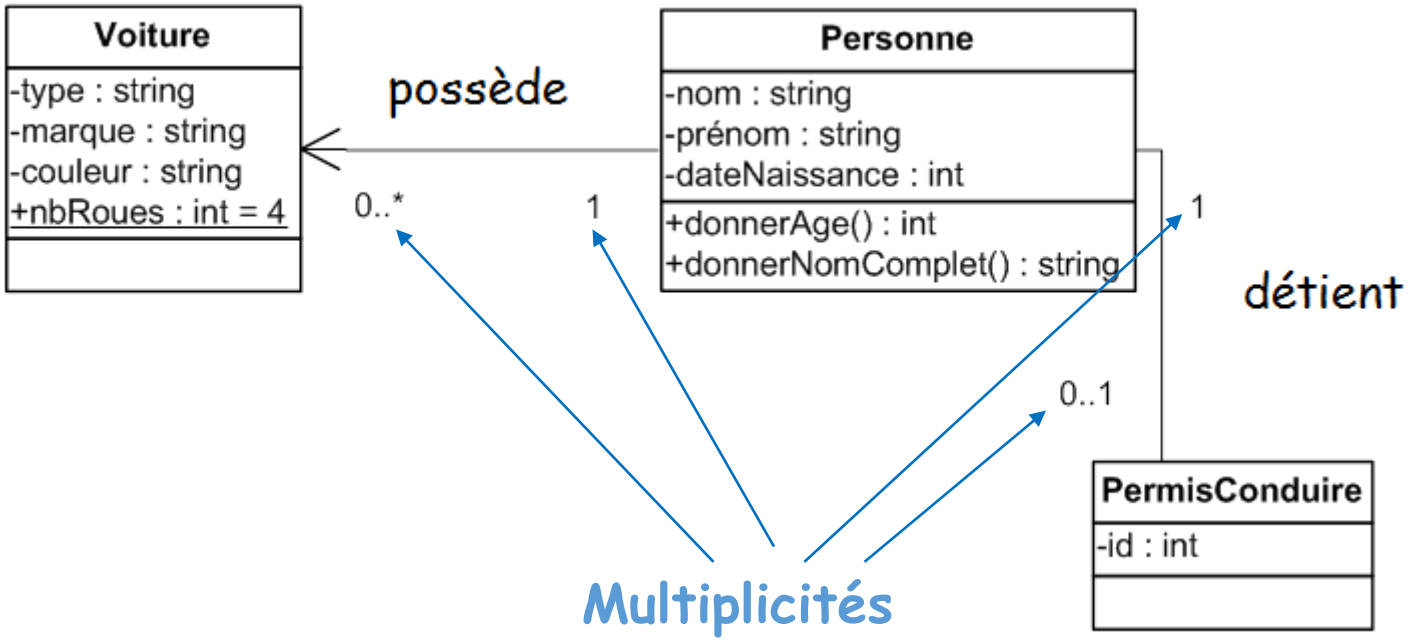
- un **nom** décrivant la **sémantique**** de l'association;
- des **rôles** décrivant la fonction jouée par chaque classe dans l'association;
- des **multiplicités** décrivant le nombre de fois qu'une instance peut participer à une relation;
- un sens de **navigation** permettant de limiter les responsabilités.
    - **Bidirectionnelle**
    - **Unidirectionnelle**

** **Sémantique ➔ signification, sens.**

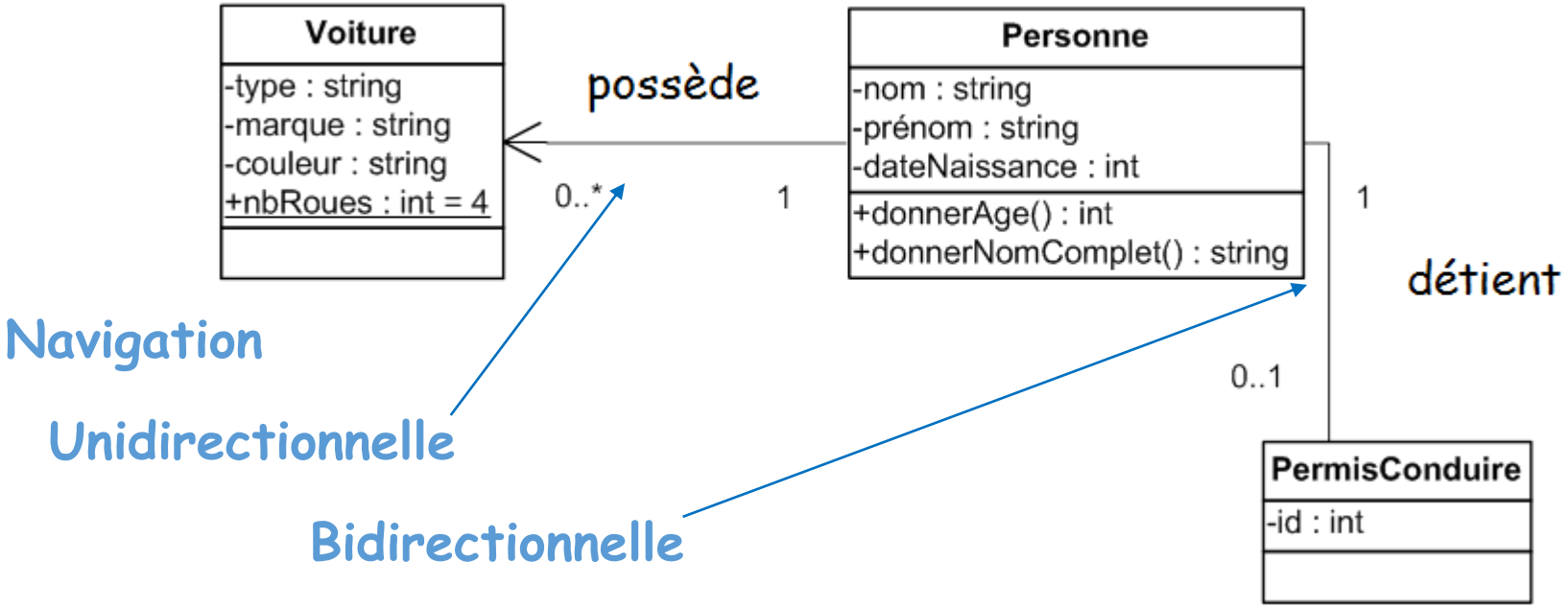
Association : sémantique



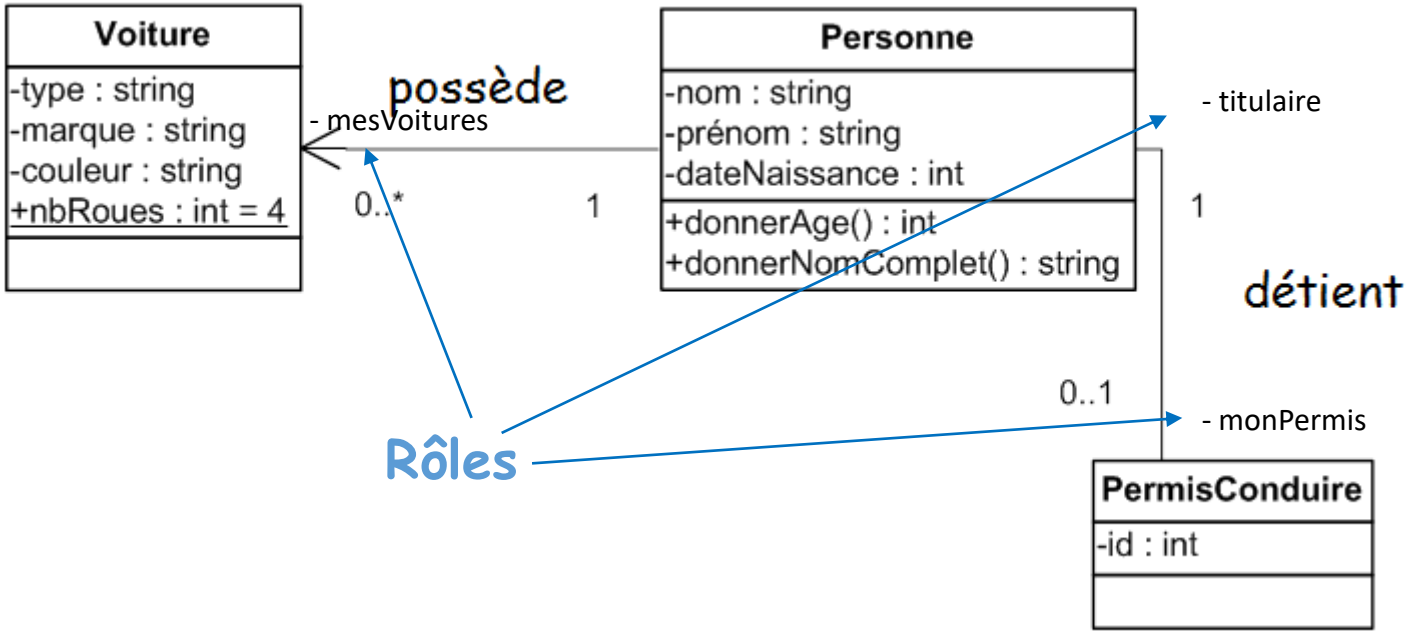
Association : multiplicités



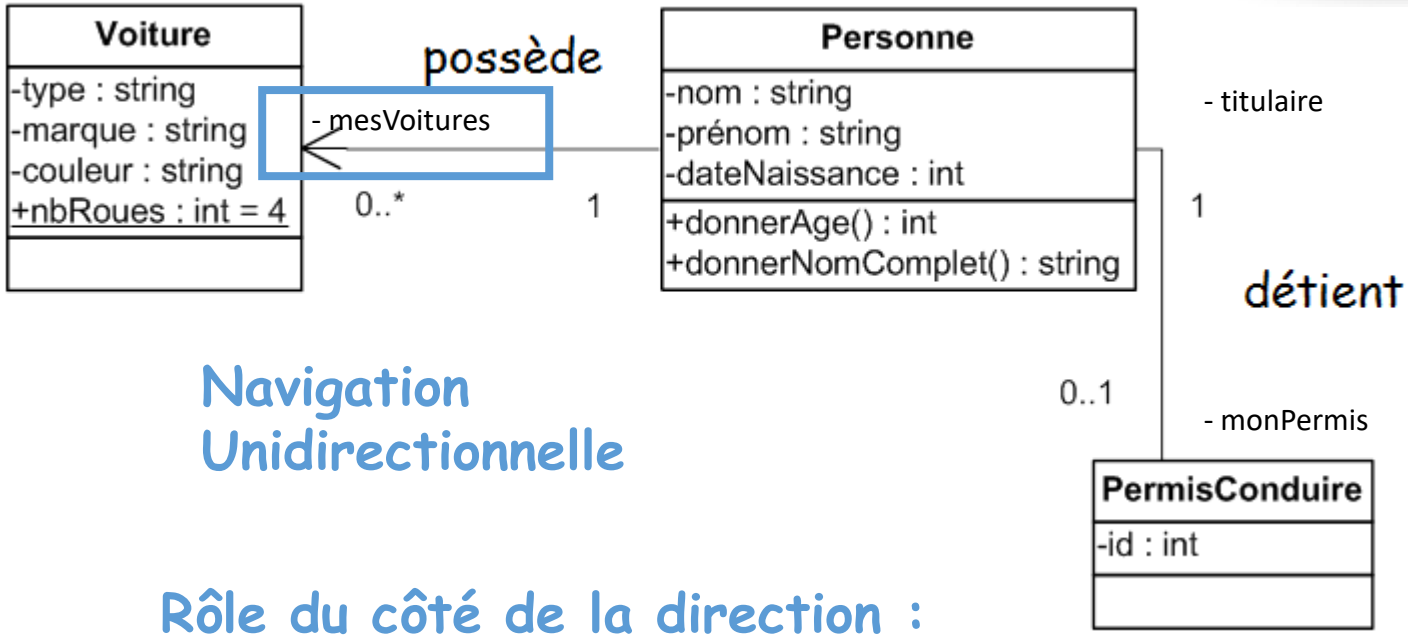
Association : navigation



Association : rôles



Association : cohérence

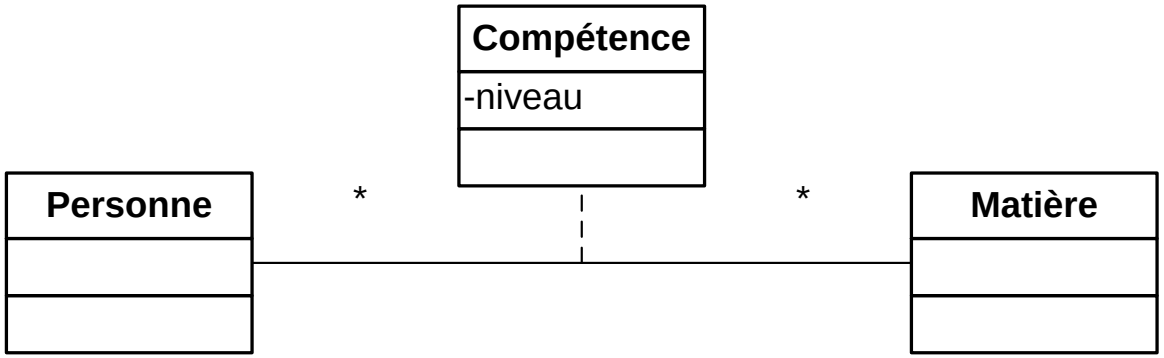


Navigation Unidirectionnelle

Rôle du côté de la direction : ce rôle deviendra un attribut de la classe "Personne", responsable de l'association.

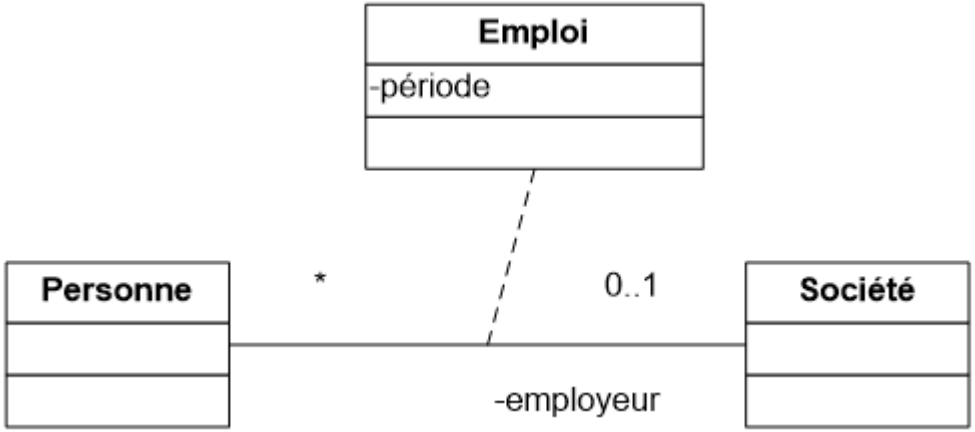
Classe-association - 1

En utilisant une classe pour représenter une association, on peut y ajouter des attributs, des opérations, ...



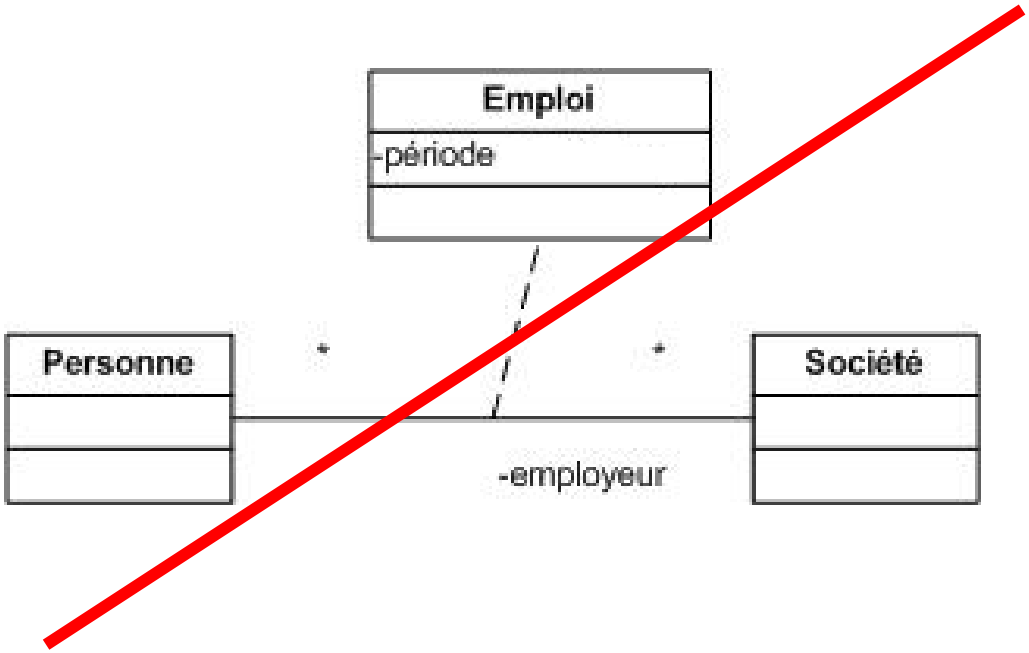
Classe-association - 2

Une personne travaille pour un employeur à la fois. On désire préciser la période pendant laquelle il est employé.



Classe-association - 3

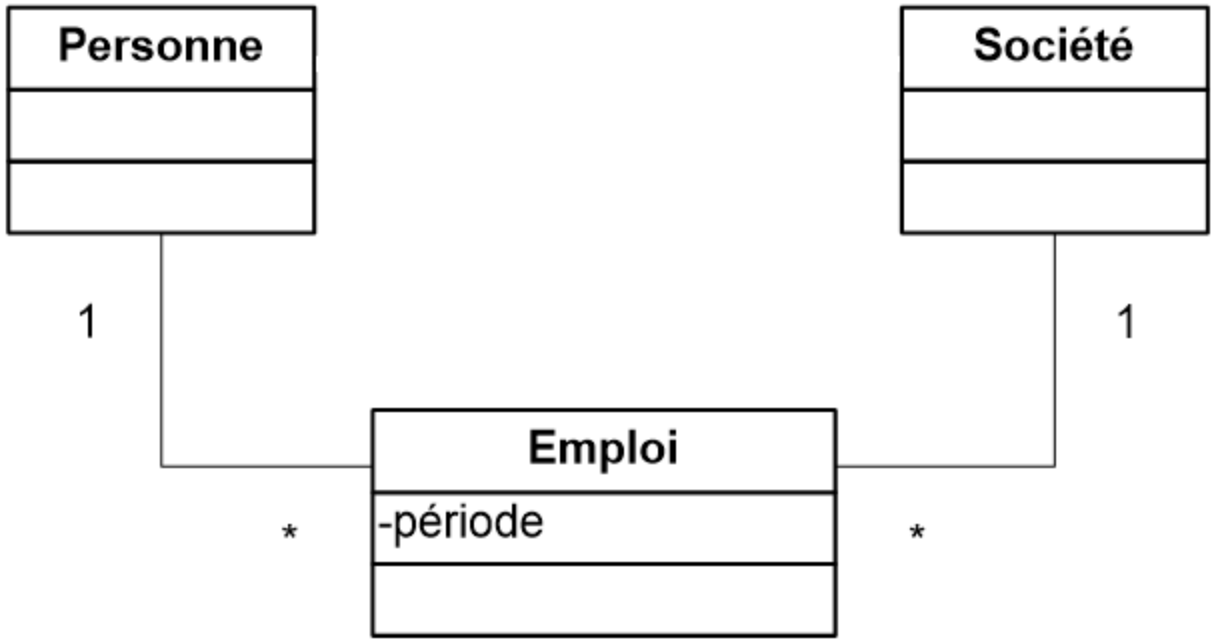
Mais si on considère divers emplois d'une personne tout au long de sa vie …



L'emploi d'une classe - association restreint une personne à ne pouvoir travailler **qu'une seule fois pour une société**, ce qui est irréaliste.

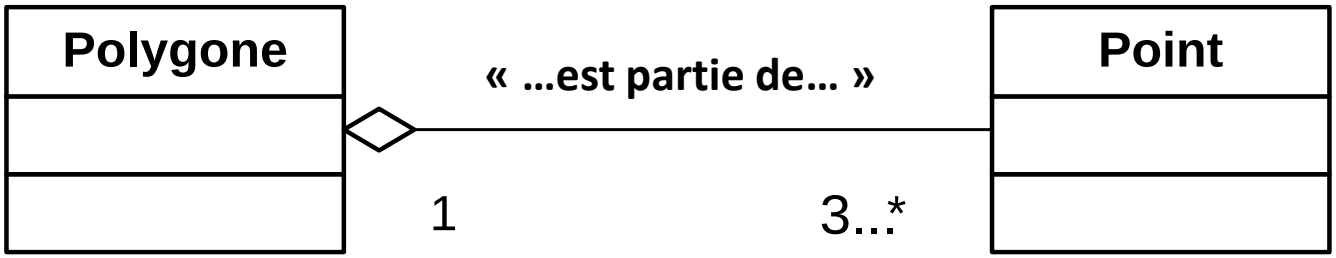
(pas de) Classe-association - 4

Une solution…

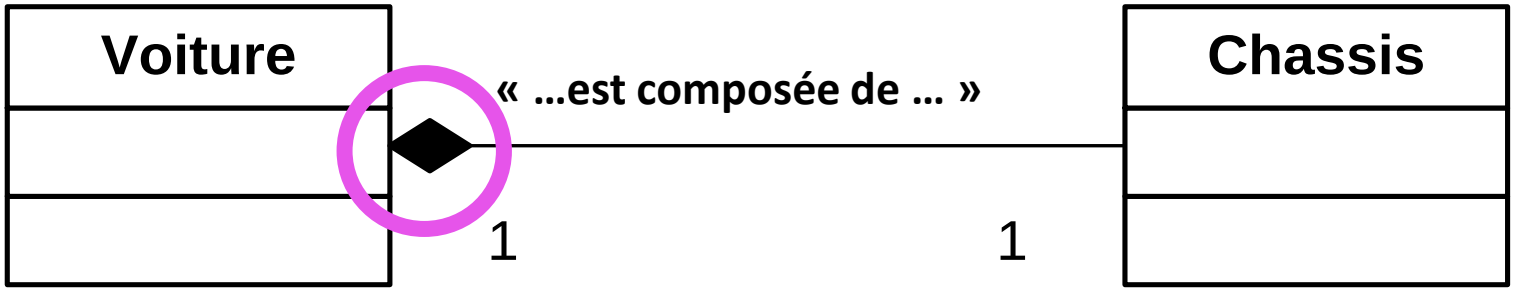


Agrégation / composition

Agrégation inclusion d'un élément dans un ensemble

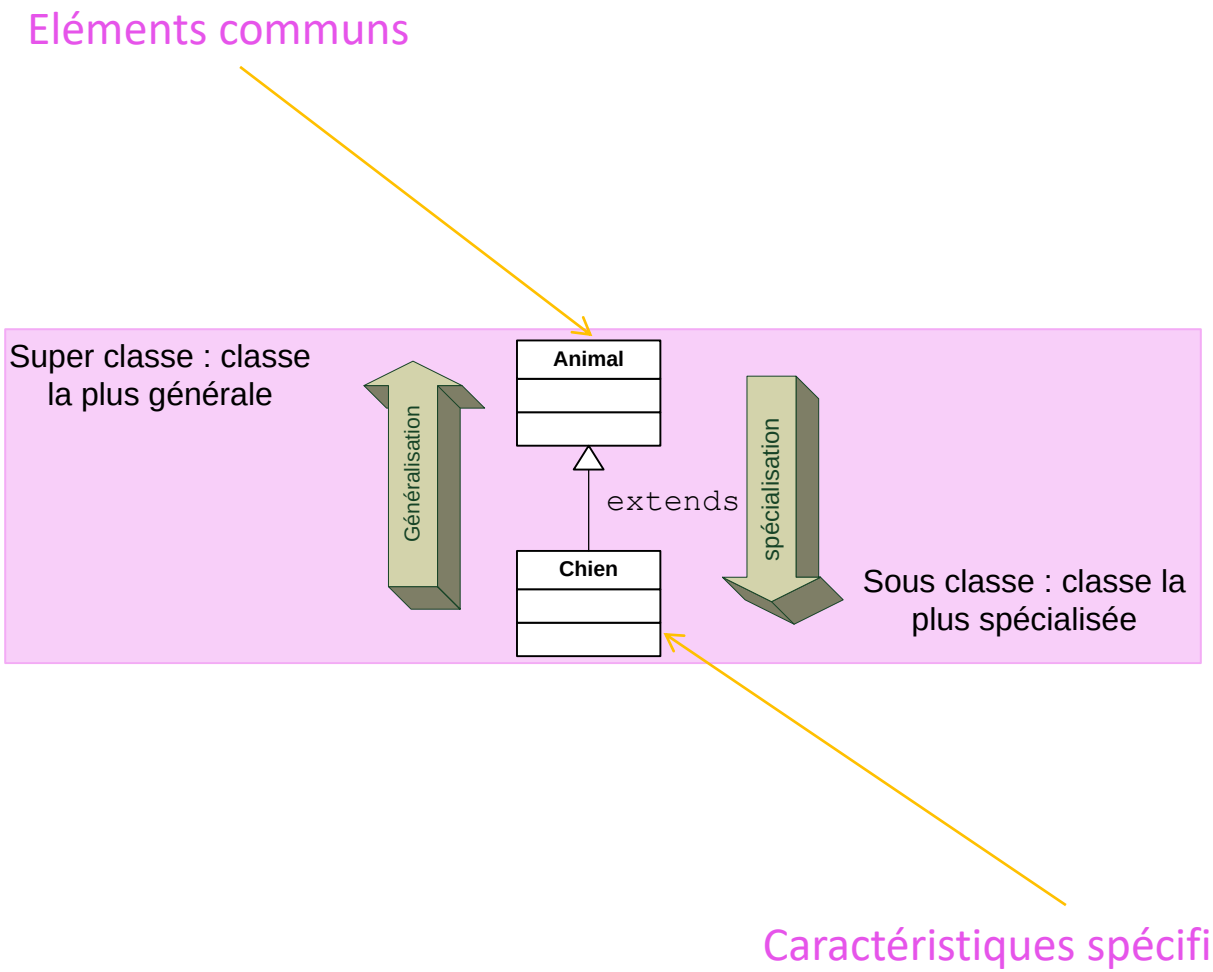


Composition forme forte d'agrégation



Les parties vivent et meurent avec le tout

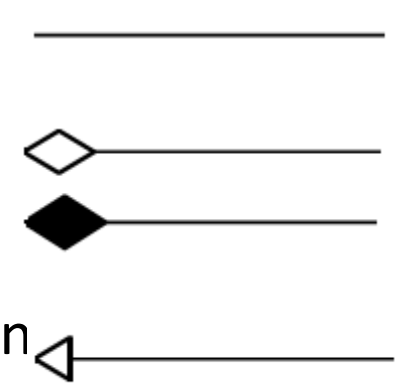
Généralisation / spécialisation



Résumé des relations entre classes

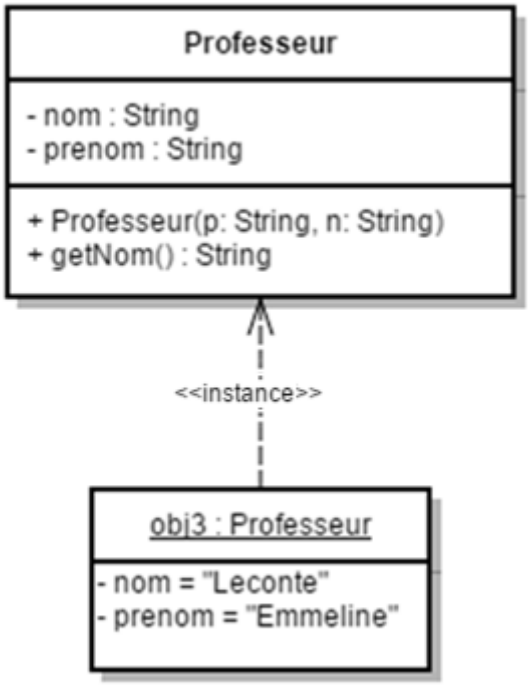
3 types de relations:

- Association.
- Agrégation / composition.
- Généralisation / spécialisation



Classification / instanciation

- Un **objet** est une **instance** d'une **classe**.
- L'**instanciation** d'une classe A correspond à la création d'objets de type A.
- La **classification** réfère à la relation entre un objet et son type. Un objet possède un type d'une certaine classe.

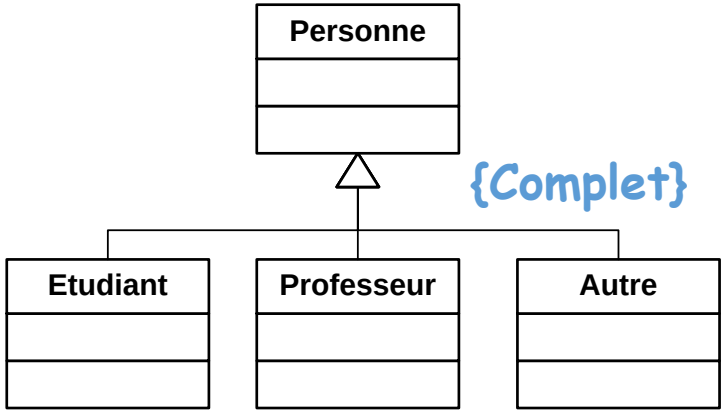


OCL

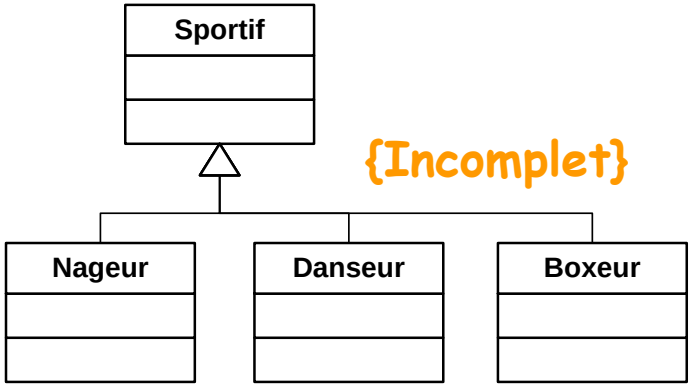
- **Object Constraint Language**
- Ce langage formel est volontairement simple d'accès et possède une grammaire élémentaire.
- Les contraintes seront mises entre accolades { }.
- Pas de syntaxe particulière pour les écrire. On pourra utiliser du français courant, ou un langage semi-formel voire un fragment de code.

OCL exemple
Spécialisation complète/incomplète

- **Complète :** l'ensemble des classes spécialisant la classe mère est complet. Il n'existe pas d'autres classes la spécialisant.



- **Incomplète :** inverse de complète.



Perspectives du diagramme

- **Conceptuelle** :
  - On y représente les concepts du domaine étudié. A ce niveau, le diagramme est indépendant du langage d'implémentation.
- **De spécification** :
  - On précise ici les interfaces pour le langage de programmation qui sera utilisé.
- **D'implémentation** :
  - On y indique comment les interfaces seront implémentés.

Niveau conceptuel

- Il n’y a pas de navigation ni de type.
- Il y a peu de méthode et peu de détails.

On ne représente pas les associations dans l’état !

Niveau spécification

- Les types sont connus.
- Les méthodes sont précisées.
- Les navigations sont présentes.

- Les détails d'implémentation ne sont pas présents.

Niveau implémentation

- Le diagramme de classes est le reflet de l'implémentation.
- Tous les détails d'implémentation sont présents;
  - Les classes utilisées.
  - Les méthodes privées.
  - …

Conclusion

- Durant l'analyse, ne dessinez que des diagrammes **conceptuels.**

- Avant d'implémenter et pendant, dessinez des diagrammes de **spécification.**

- Ne dessinez pas de diagramme d'implémentation si cela n'est pas nécessaire.

Conclusion

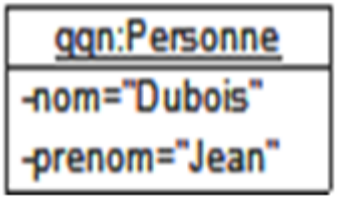
- Les diagrammes de Classes sont **très riches.**
- N'utilisez dans un premier temps que les **concepts principaux** : classes, associations, attributs, opérations, contraintes et généralisations.
- N'introduisez les **autres notions** que quand vous en avez vraiment besoin (10% des cas).
    - Nous n'avons pas abordé pour le cours d'UML :
        - Association qualifiée.
        - Classes paramétrées.
        - Associations dérivées.

Le diagramme d’objets

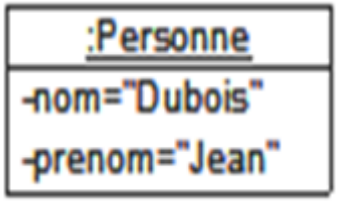
Notions & notation

- Les **diagrammes d'objet** constituent des **instanciations** de diagrammes de classes.
- On y indique l'état des objets, càd l'ensemble des valeurs de leurs attributs.
- On indique le nom de l'objet, souligné et en minuscules. Le nom de la classe est précédé de « : ».

Objets

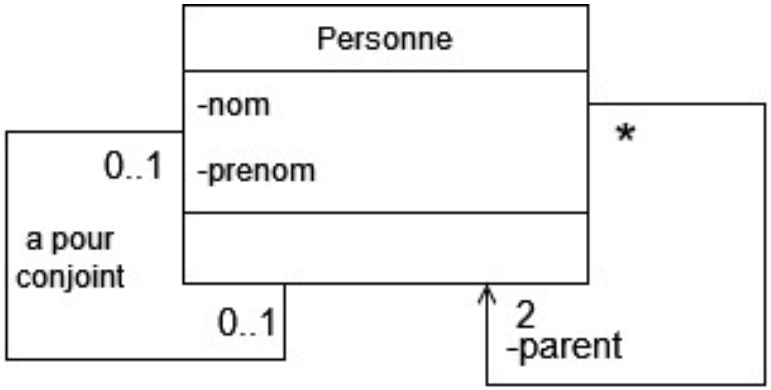


L'**objet** `qqn` est de type `Personne`.



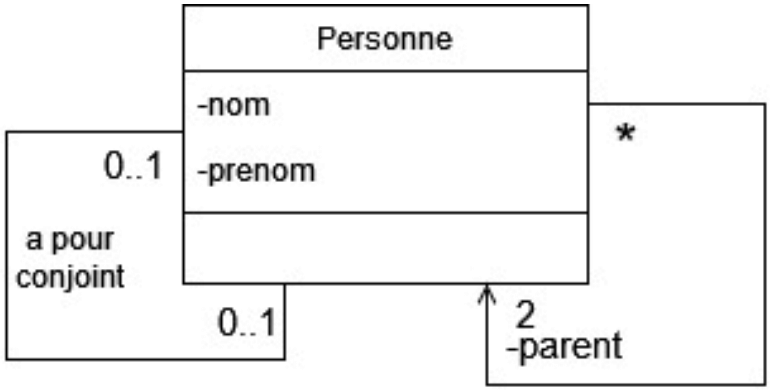
Il y a un objet **anonyme** de type `Personne`.

Liens entre objets

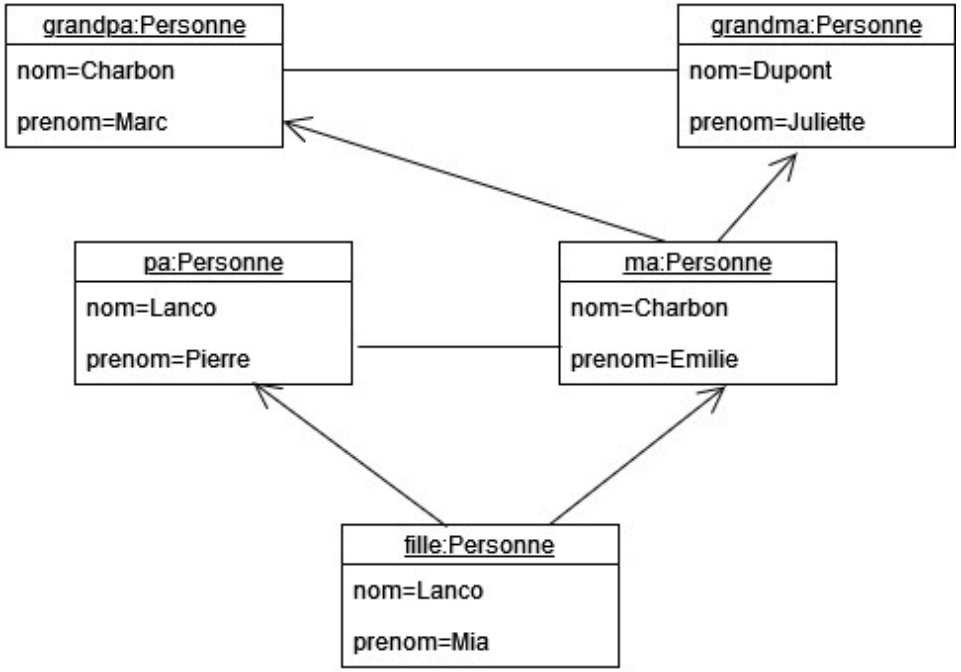


Les **liens** (instances d'une association) sont les relations entre objets.

Liens entre objets



Les **liens** (instances d'une association) sont les relations entre objets.



Questions ?