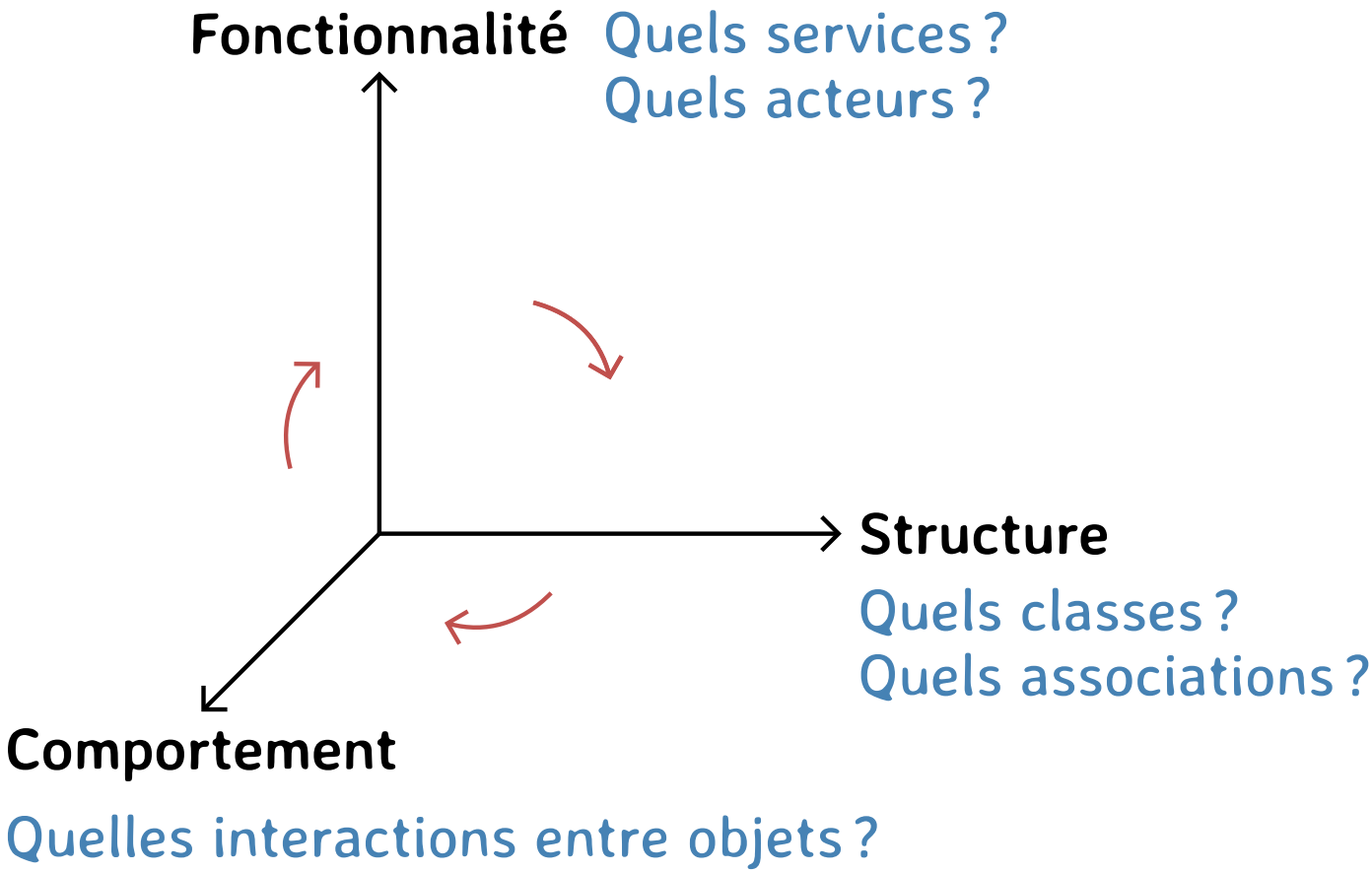
# UML - La vue statique

## Ingéniérie des systèmes d'information

Paul Brunet

# UML - La vue statique
## Plan

# L'approche UML



# Diagrammes statiques

La vue statique est décrite par quatre types de diagrammes :

- Diagramme de classes
- Diagramme d'objet
- Diagramme de composants
- Diagramme de déploiement

# Diagrammes statiques

La vue statique est décrite par quatre types de diagrammes :

- **Diagramme de classes**
- **Diagramme d'objet**
- Diagramme de composants
- Diagramme de déploiement

# UML - La vue statique
## Plan

1. Introduction

2. Diagramme de classes
   - 2.1. Classes
   - 2.2. Associations
   - 2.3. Généralisations
   - 2.4. Construire un diagramme de classe

3. Autres diagrammes statiques

# Diagramme de classes
## Objectifs

☞ Le diagramme de classes permet de décrire les classes d'objets (description structurelle et comportementale) et les relations qui interviennent, dans le cadre du problème posé.

# Diagramme de classes
## Objectifs

- Le diagramme de classes permet de décrire les classes d'objets (description structurelle et comportementale) et les relations qui interviennent, dans le cadre du problème posé.
- Cette description est indépendante de la solution technique choisie.

# Diagramme de classes
## Objectifs

- Le diagramme de classes permet de décrire les classes d'objets (description structurelle et comportementale) et les relations qui interviennent, dans le cadre du problème posé.
- Cette description est indépendante de la solution technique choisie.
- Le diagramme de classes est au centre de la modélisation UML. C'est celui qui permet la génération automatique du code et c'est celui que produit l'ingénierie inverse.

# Diagramme de classes
## Objectifs

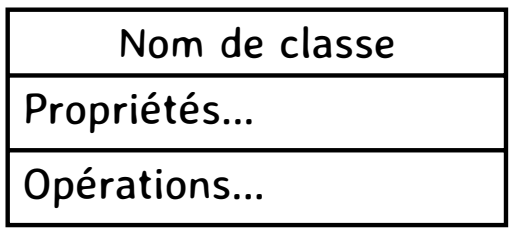
- Le diagramme de classes permet de décrire les classes d'objets (description structurelle et comportementale) et les relations qui interviennent, dans le cadre du problème posé.
- Cette description est indépendante de la solution technique choisie.
- Le diagramme de classes est au centre de la modélisation UML. C'est celui qui permet la génération automatique du code et c'est celui que produit l'ingénierie inverse.
- **Les autres diagrammes servent donc de supports pour la construction du diagramme des classes.**

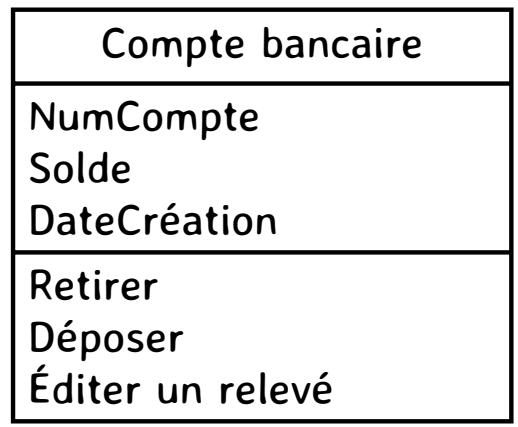
# UML - La vue statique
## Plan

# Classe
## Concept

définition

Une **classe** est une description d'un **ensemble d'objets** qui ont des **propriétés** similaires (attributs), un **comportement** commun (opérations) et des **relations** communes avec d'autres objets.




Une classe permet de construire les objets. On dit qu'un objet est une instance d'une classe. **Un objet : Etat, Identité, Comportement.**

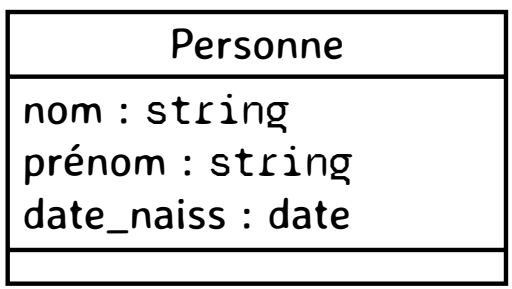
# Classe : propriétés, attributs
## Concept

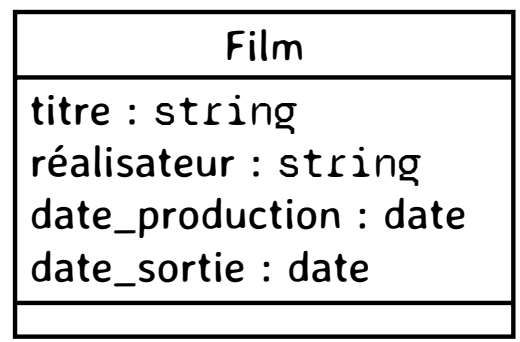
### définition

Un attribut représente une information pertinente *(pour le problème posé)* contenue dans une classe.

### syntaxe

```
visibilité (/) <nom-attribut> ('['multiplicité']') : <nom-classe> (=valeur initiale) (propriété)
```



# Déclaration d'attributs

syntaxe

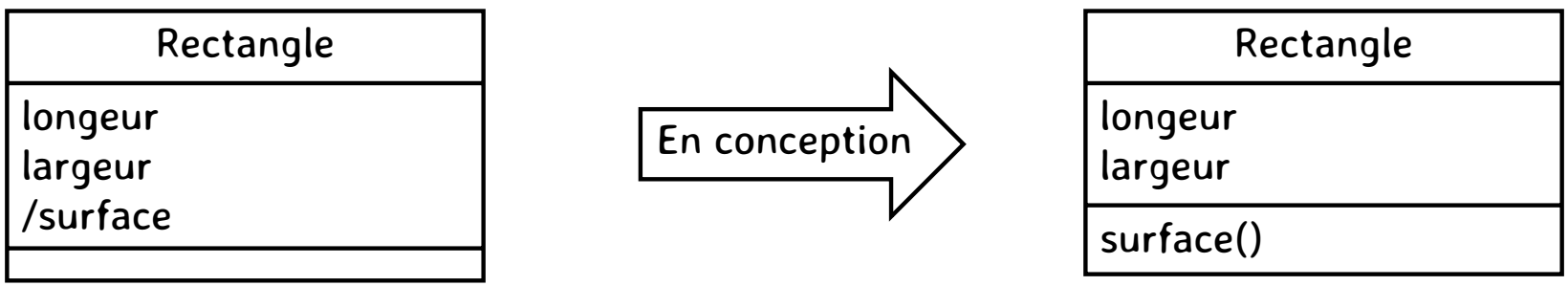
```
visibilité (/) <nom-attribut> ('['multiplicité']') : <nom-classe> (=valeur initiale) (propriété)
```

- La visibilité représente le principe d'encapsulation dans le langage UML.
- Niveaux de visibilité pour les attributs :
  - **public (+)** : l'accès est possible pour tous les objets de toutes les autres classes
  - **protégé (#)** : L'accès est possible pour les objets d'une sous-classe
  - **privé (-)** : l'accès est possible pour les objets de la classe
- La multiplicité peut indiquer les nombres minimaux/maximaux de valeurs possibles de l'attribut pour une instance de la classe. La multiplicité par défaut est 1,1.

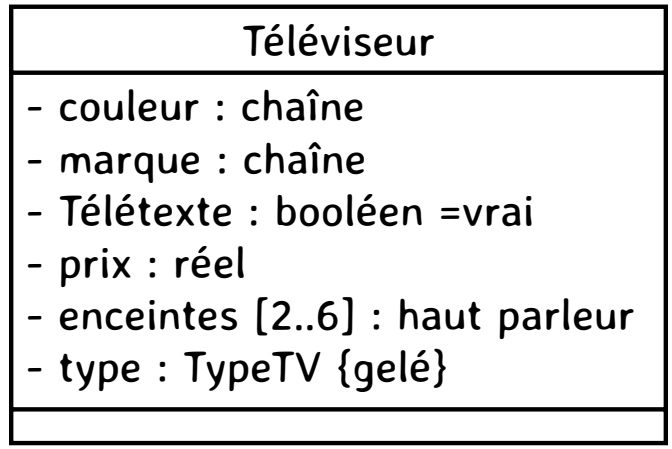
# Classe : attribut dérivé
## Concept

définition

Les attributs dérivés sont des attributs dont les valeurs sont déduites d'autres attributs de la classe.



# Classe
## Exemple



# Classe : opération
## Concept

### définition

Une opération représente un élément de comportement (un service) contenu dans une classe.

### syntaxe

```
visibilité <nom-opération>( [arguments]): [<type-valeur-renvoyée>]
```

**Exemples**

- display()
- nbrElement() : entier
- +setSolde(montant : euros)
- getSolde() : euros

# Déclaration d'opération

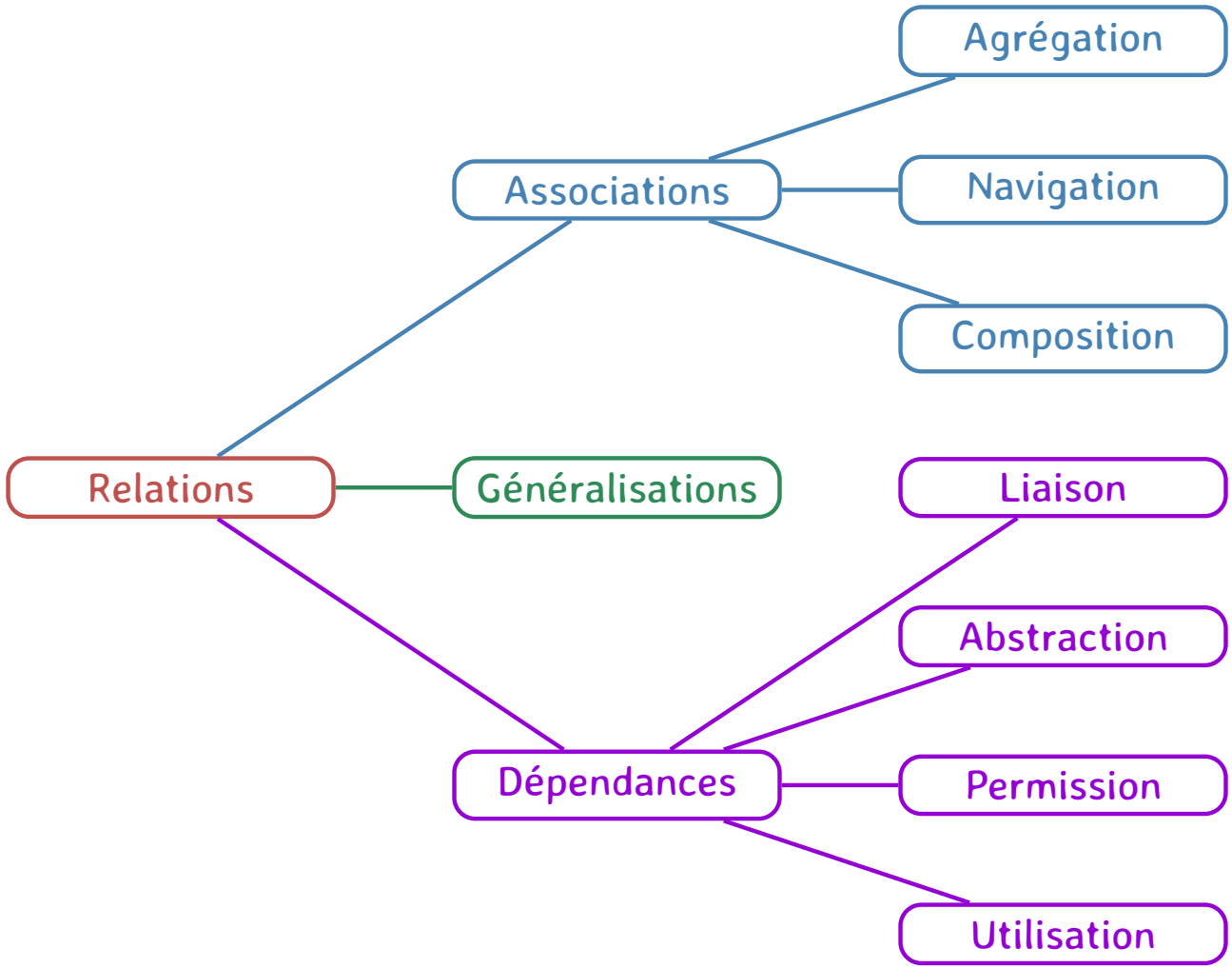
syntaxe

```
visibilité <nom-opération>( [arguments]): [<type-valeur-renvoyée>]
```

- L'opération correspond à la déclaration de la signature ou d'interface.
- La méthode correspond à l'implantation (le corps). UML autorise d'utiliser n'importe quel langage pour écrire les méthodes (C++, Java, C sharp, ...).
- Il peut y avoir plusieurs méthodes pour une même opération.
- La notion de visibilité s'applique de la même manière que pour le concept d'attribut.

# UML - La vue statique
## Plan

# Relations entre classes
## Concept



# Association
## Concept

définition

Une association représente une relation sémantique entre les objets des différentes classes.


Il est recommandé de nommer les associations par une forme verbale, soit active, soit passive.

# Association
## Concept

définition

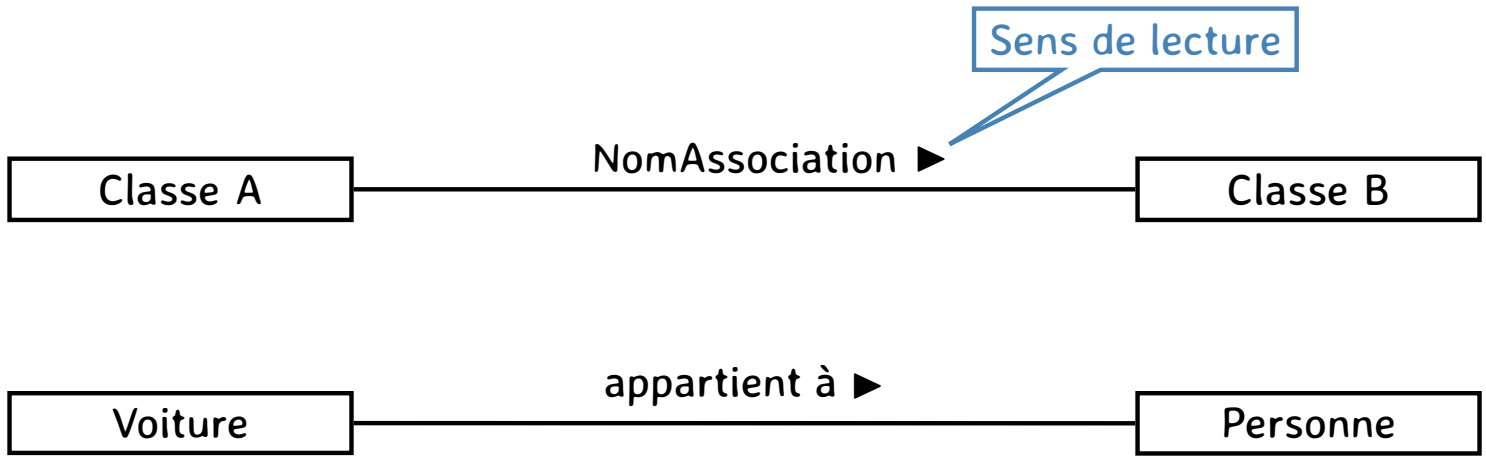
Une association représente une relation sémantique entre les objets des différentes classes.


Il est recommandé de nommer les associations par une forme verbale, soit active, soit passive.

# Association
## Concept

définition

Une association représente une relation sémantique entre les objets des différentes classes.

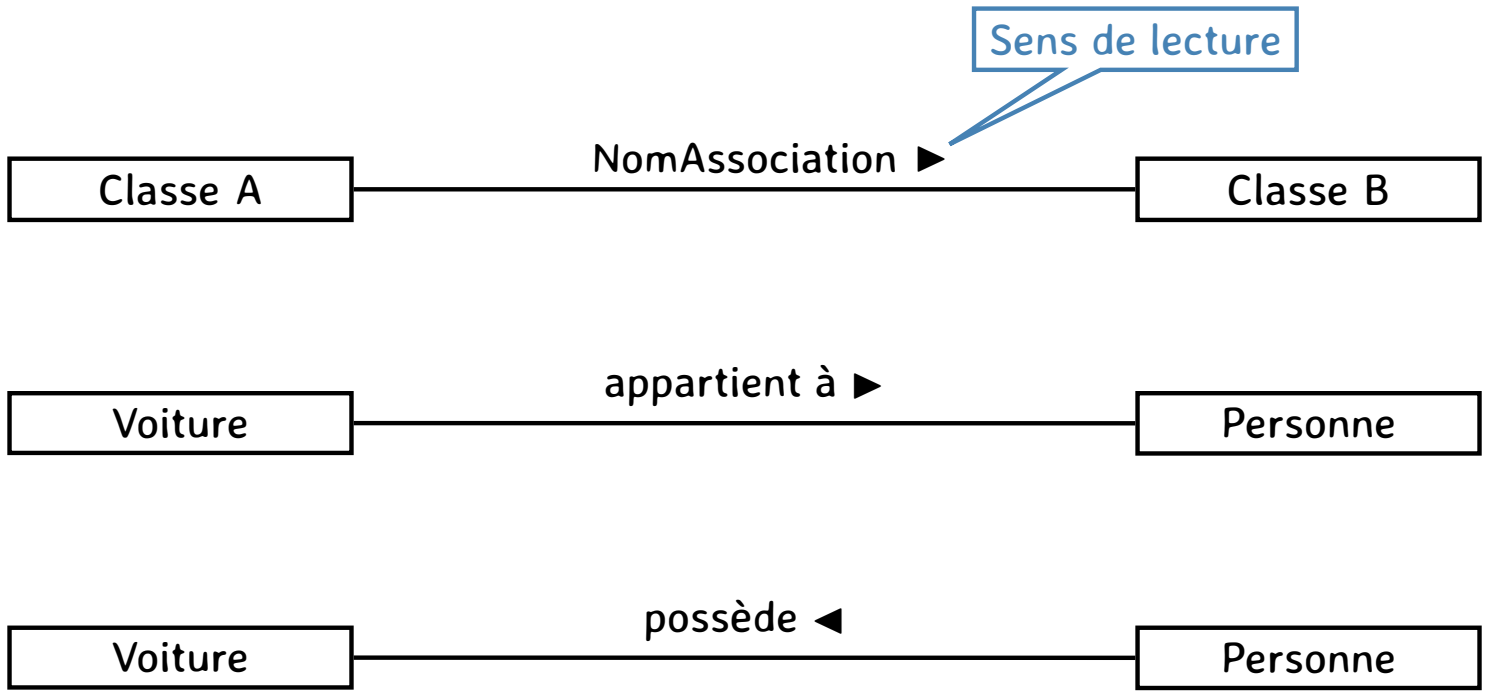



Il est recommandé de nommer les associations par une forme verbale, soit active, soit passive.

# Association
## Concept

définition

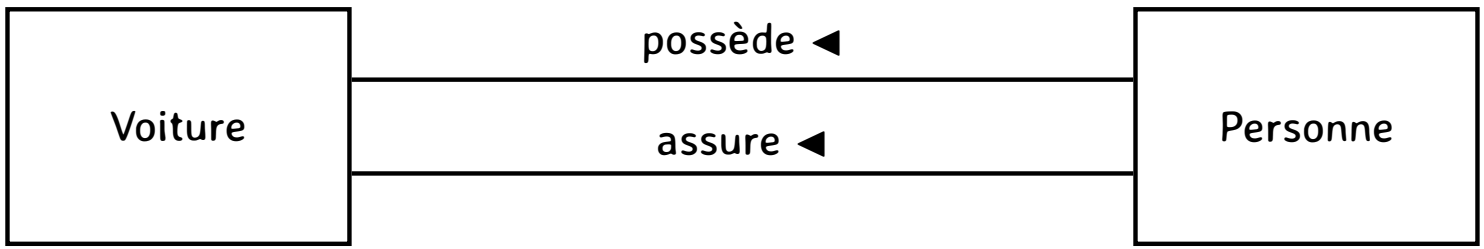
Une association représente une relation sémantique entre les objets des différentes classes.




Il est recommandé de nommer les associations par une forme verbale, soit active, soit passive.

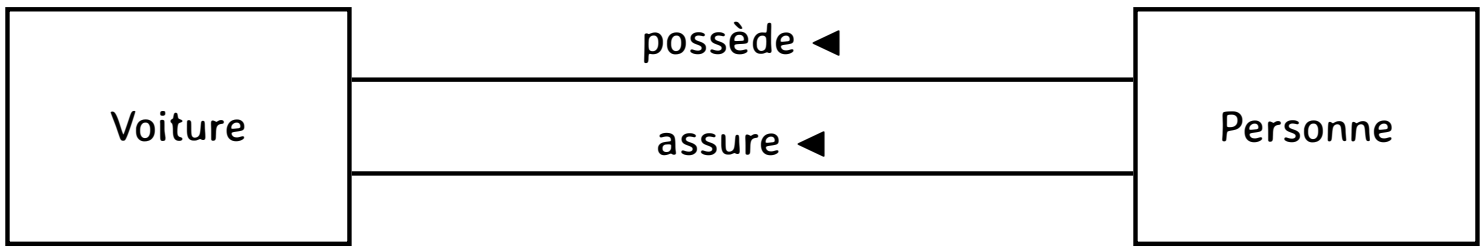
# Association
## Concept

☞ Il peut y avoir deux ou plusieurs associations de nature différente entre les deux mêmes classes.

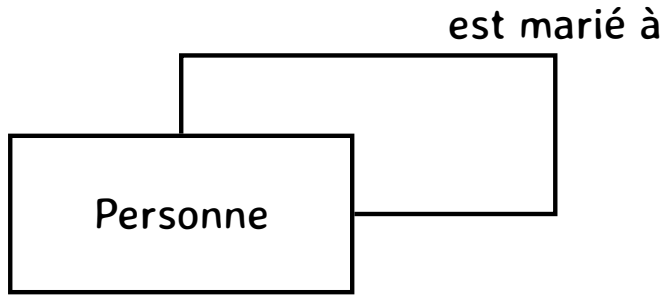


# Association
## Concept

☞ Il peut y avoir deux ou plusieurs associations de nature différente entre les deux mêmes classes.

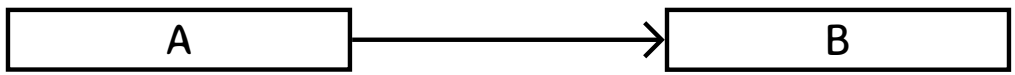



☞ Une association peut affecter une seule classe, c'est une association dite réflexive.



# Association
## Concept

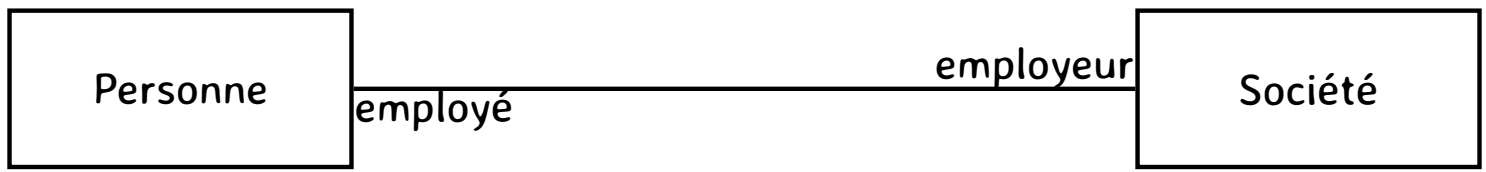
- Par défaut, les associations sont bidirectionnelles.
- Si seule une direction est utile, ceci se représente par une flèche portée par le rôle vers lequel la navigation est possible.




- Distinction souvent faite au moment de la conception

# Rôle
## Concept

- Le rôle décrit comment une classe « voit » une autre classe via une association.
- Les noms des associations et les noms des rôles ne sont pas exclusifs l'un de l'autre.



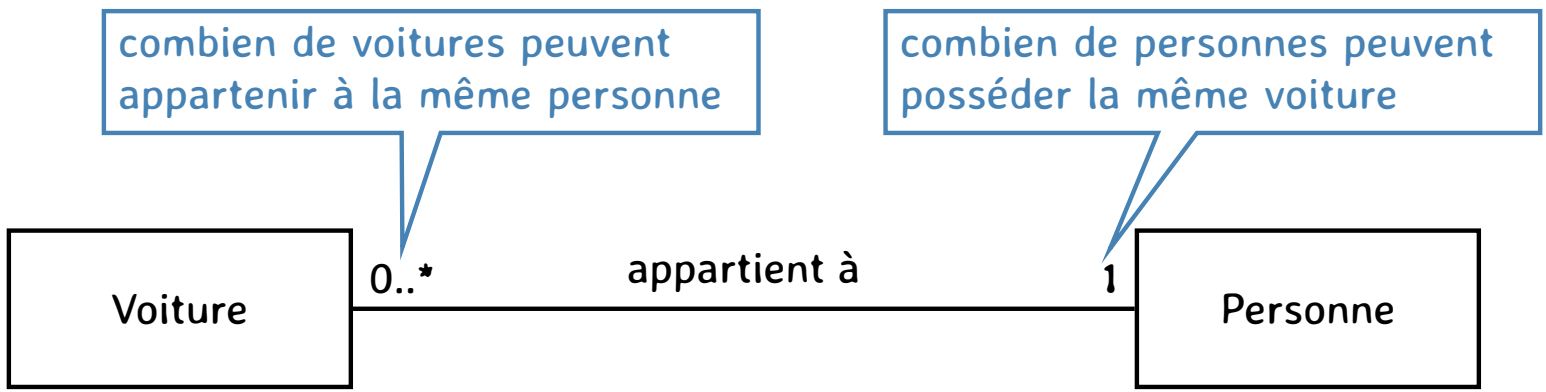
# Contraintes
## Concept

- De manière générale, une contrainte UML est une règle de gestion attachée à un élément du modèle. Elle est exprimée entre accolades, en langage naturel ou formel (ex : en Object Constraint Language).
- Une contrainte peut porter sur une association ou sur un groupe d'associations.
- On distingue principalement trois types de contraintes sur les groupes d'associations :
  - Multiplicités
  - Contrainte d'inclusion
  - Contrainte d'exclusion

# Multiplicité
## Concept

définition

La multiplicité d'une association indique le nombre minimum et maximum d'objets de la classe qui peuvent être liés à un objet de l'autre classe.



# Multiplicités
## Notations

La multiplicité est une information traduisant une règle de gestion, elle est exprimée sous la forme d'une expression entière bornée.

syntaxe

- $n$ : exactement $n$.
- $n..m$ : entre $n$ et $m$ inclus.
- $n_1, n_2, n_3$ : soit $n_1$, soit $n_2$, soit $n_3$.
- * : n'importe quel nombre.

42 ..... Exactement quarante-deux
2..5 ..... Entre deux et cinq inclus
2,5 ..... Soit deux, soit cinq
3..* ..... Au moins trois
1..* ..... Au moins un

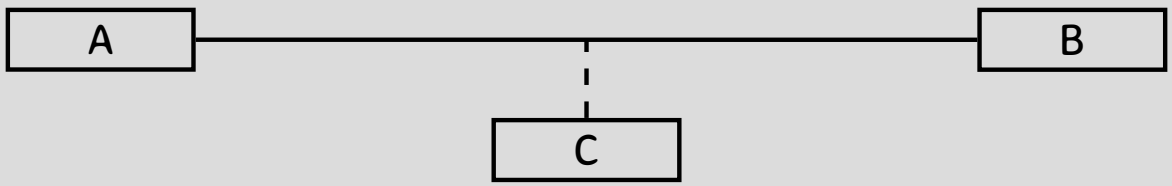
**NB :** différences de convention avec Merise dans le placement des multiplicités.

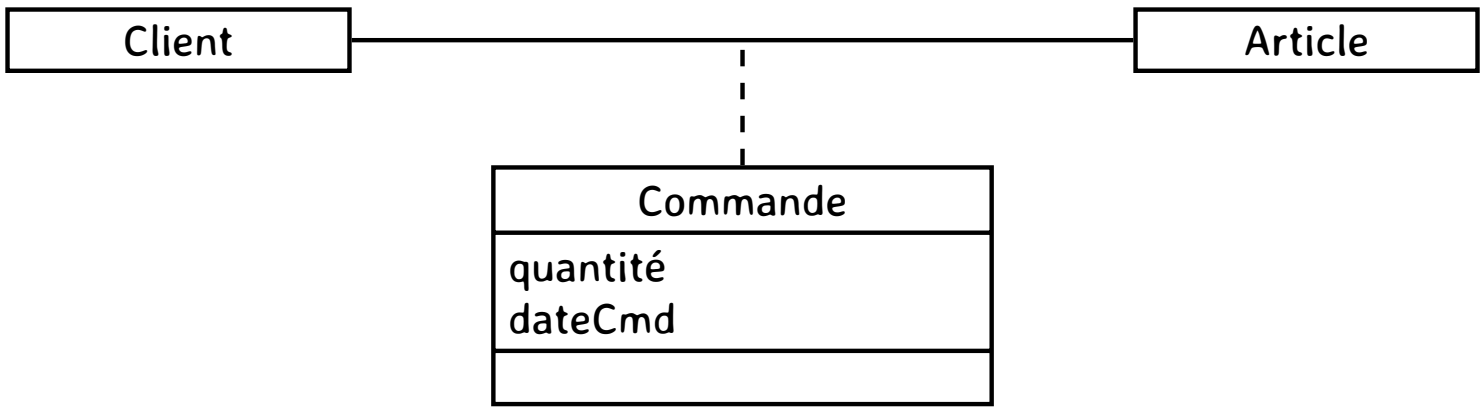
# Classe d'association
## Concept

définition

Une classe d'association contient des informations (attributs ou opérations) concernant une association.

syntaxe

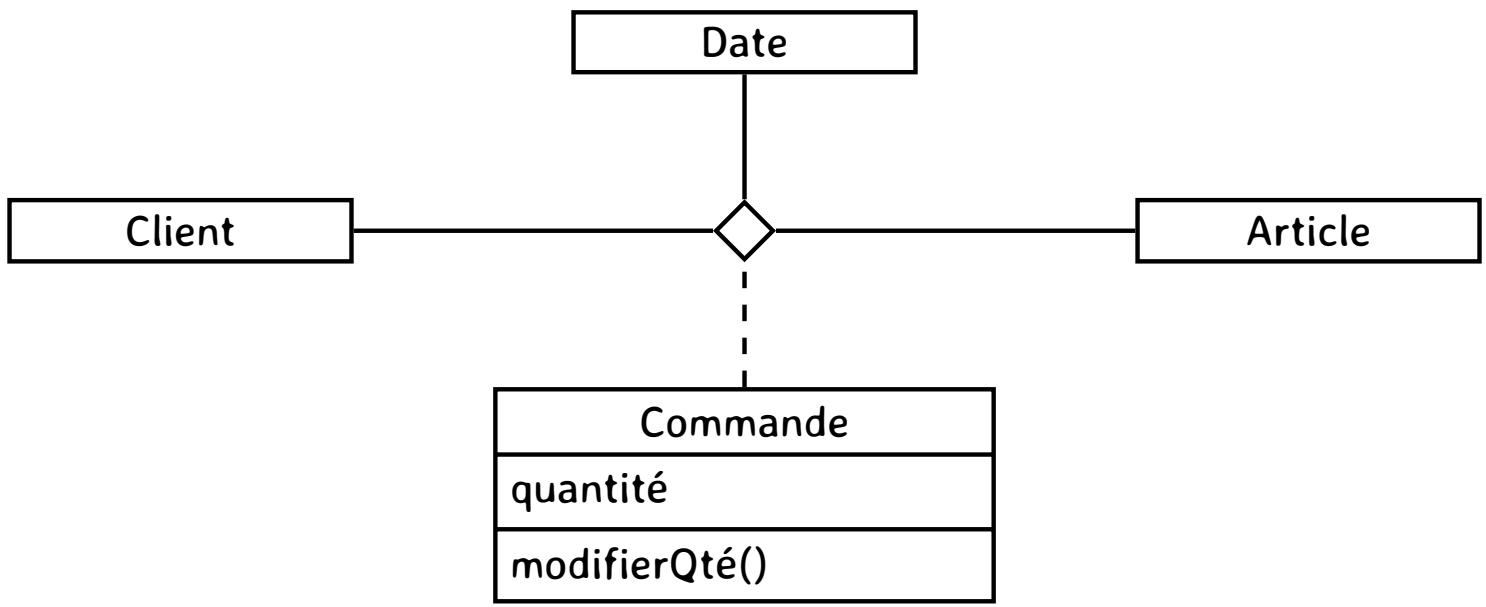


# Association ternaire
## Concept

définition

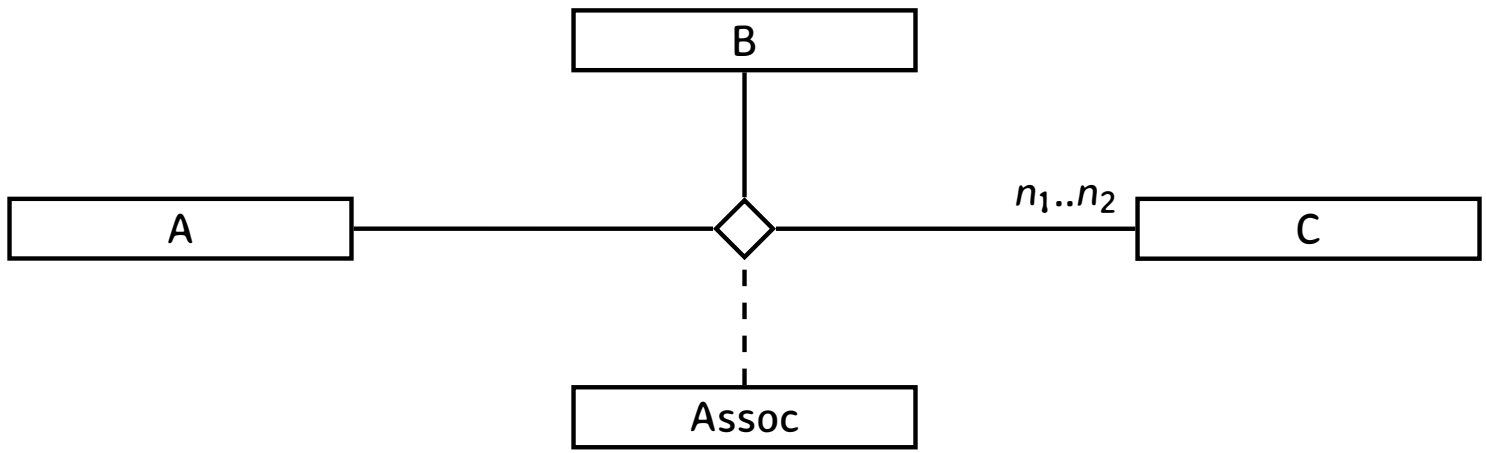
Une association ternaire fait intervenir trois classes. Une association ternaire est généralement porteuse d'attributs.



# Association ternaire
## Multiplicité

Pour une association ternaire, les multiplicités se lisent de la façon suivante :

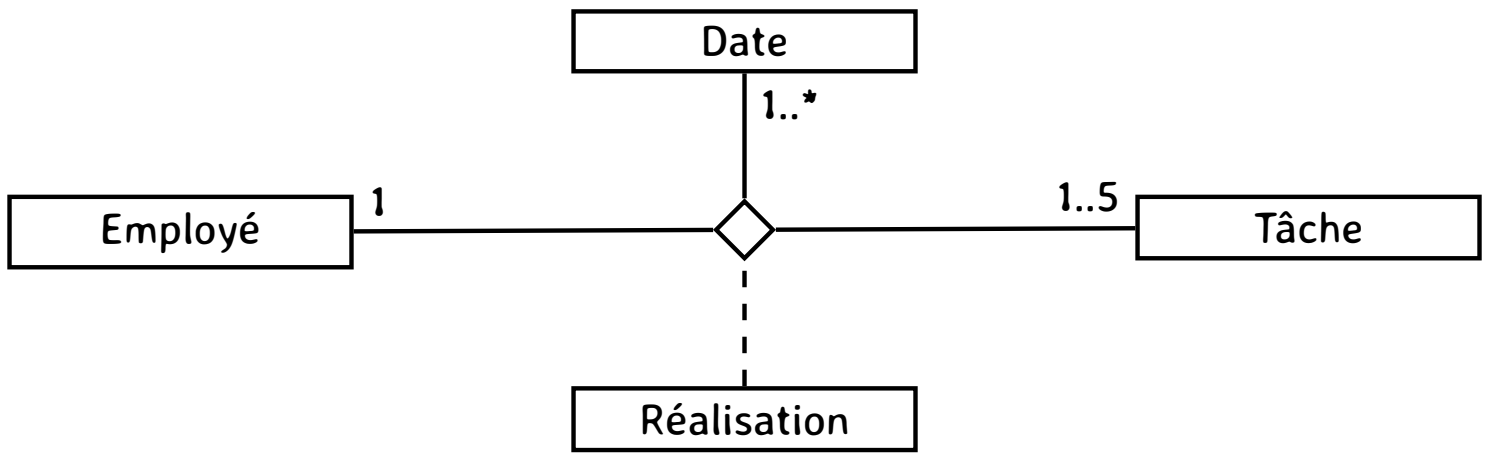
pour chaque paire $(a, b)$ d'une instance de la classe A et d'une de la classe B, il y a entre $n_1$ et $n_2$ instances $c$ de la classe C telles que le triplet $(a, b, c)$ appartienne à la relation.



# Association ternaire
## Multiplicité

Pour une association ternaire, les multiplicités se lisent de la façon suivante :

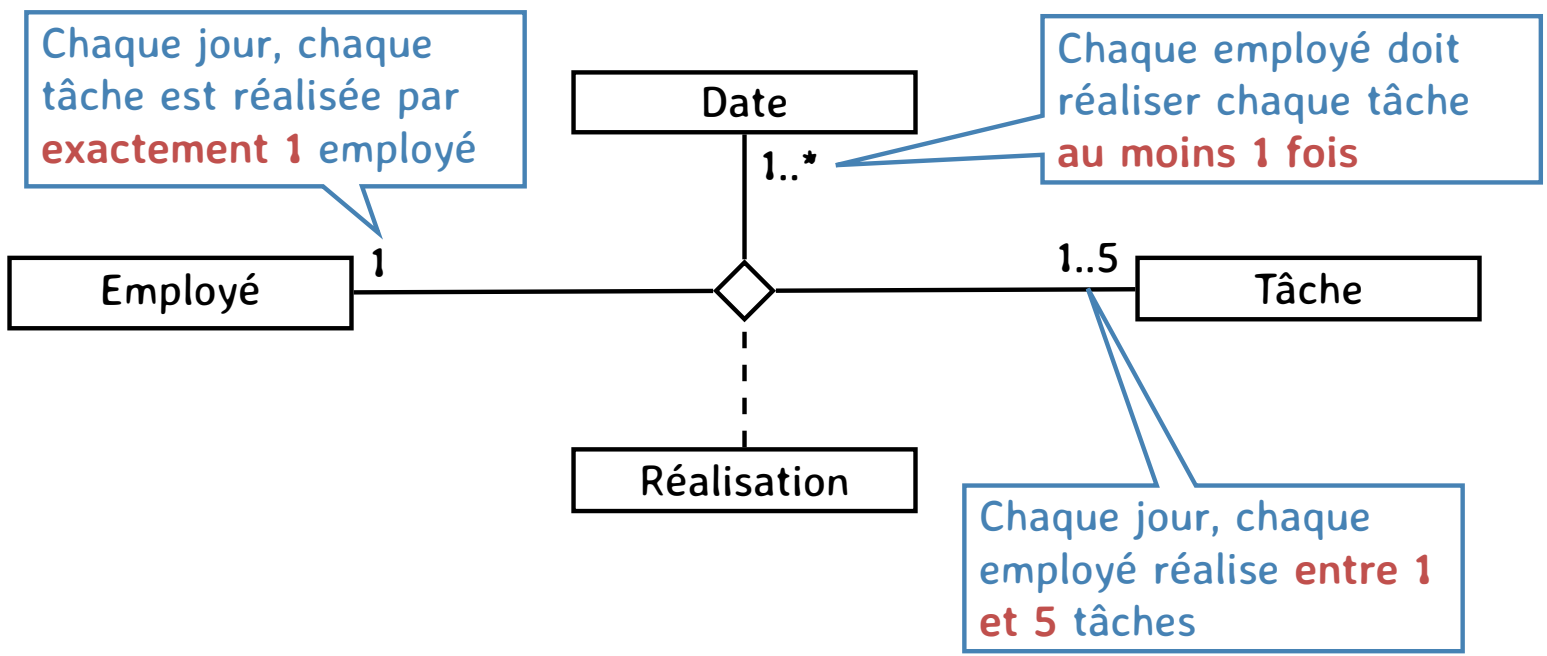
pour chaque paire $(a, b)$ d'une instance de la classe A et d'une de la classe B, il y a entre $n_1$ et $n_2$ instances $c$ de la classe C telles que le triplet $(a, b, c)$ appartienne à la relation.



# Association ternaire
## Multiplicité

Pour une association ternaire, les multiplicités se lisent de la façon suivante :

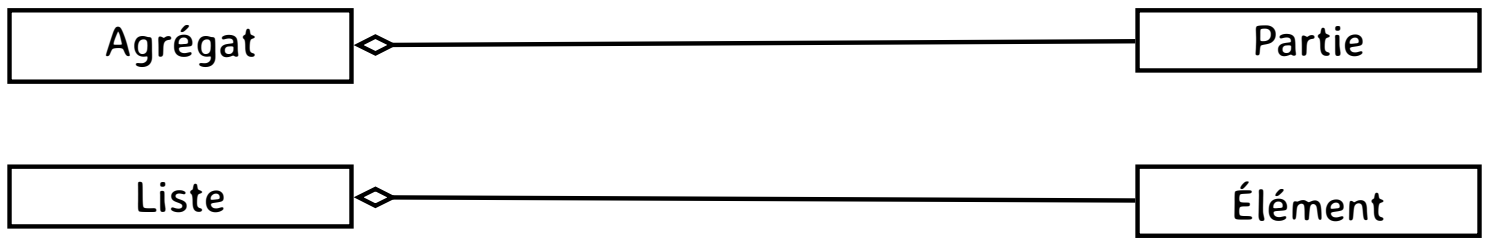
> pour chaque paire $(a, b)$ d'une instance de la classe A et d'une de la classe B, il y a entre $n_1$ et $n_2$ instances $c$ de la classe C telles que le triplet $(a, b, c)$ appartienne à la relation.



# Agrégation
## Concept

définition

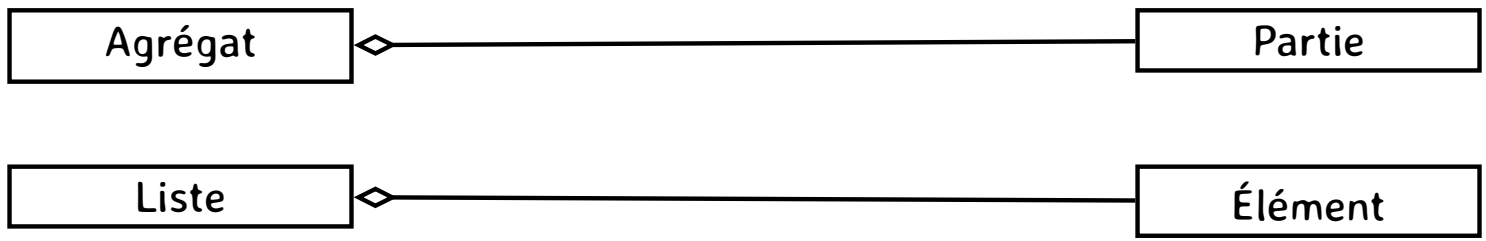
L'agrégation est une forme particulière d'association non symétrique qui décrit une relation d'inclusion entre une partie et un tout (l'agrégat).



# Agrégation
## Concept

définition

L'agrégation est une forme particulière d'association non symétrique qui décrit une relation d'inclusion entre une partie et un tout (l'agrégat).

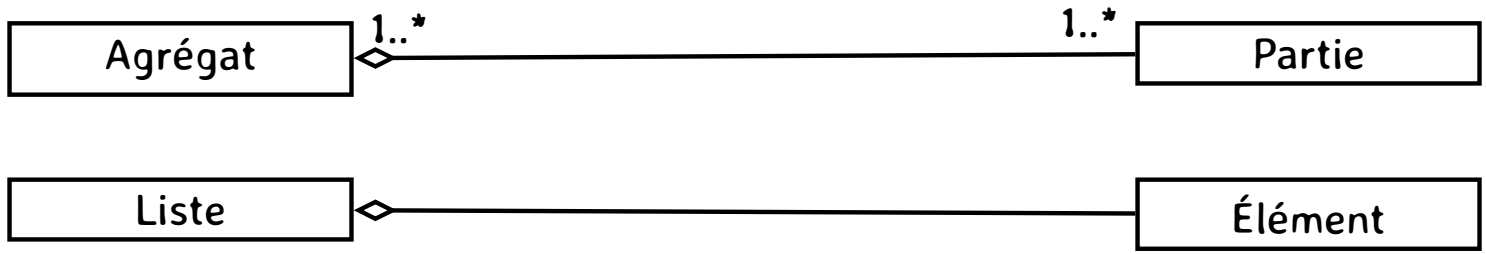



☞ Les agrégations n'ont pas besoin d'être nommées : implicitement elles signifient « contient », « est composé de ».

# Agrégation
## Concept

définition

L'agrégation est une forme particulière d'association non symétrique qui décrit une relation d'inclusion entre une partie et un tout (l'agrégat).

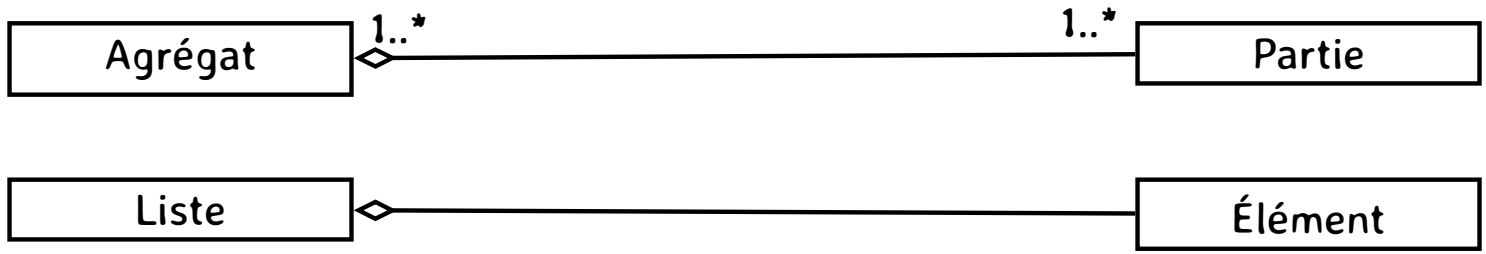



- Les agrégations n'ont pas besoin d'être nommées : implicitement elles signifient « contient », « est composé de ».
- L'agrégation peut être multiple, comme l'association.

# Agrégation
## Concept

définition

L'agrégation est une forme particulière d'association non symétrique qui décrit une relation d'inclusion entre une partie et un tout (l'agrégat).

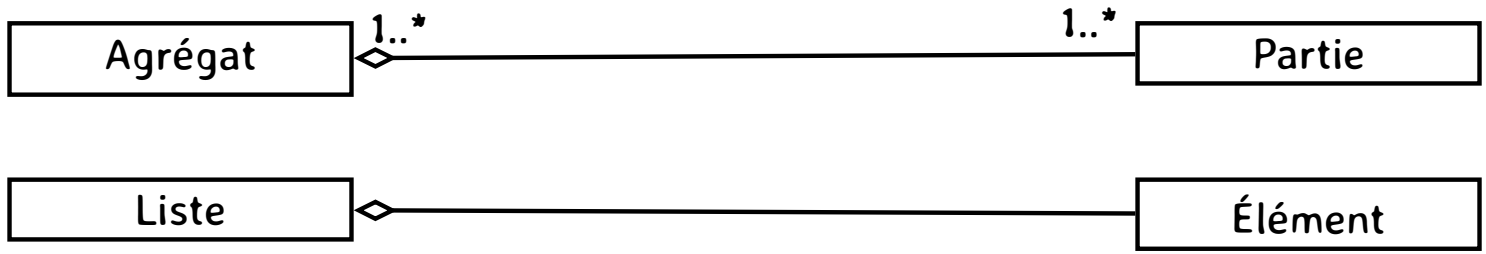



- Les agrégations n'ont pas besoin d'être nommées : implicitement elles signifient « contient », « est composé de ».
- L'agrégation peut être multiple, comme l'association.
- La durée de vie de l'agrégat est indépendante des éléments qui le constituent.

# Agrégation
## Concept

définition

L'agrégation est une forme particulière d'association non symétrique qui décrit une relation d'inclusion entre une partie et un tout (l'agrégat).

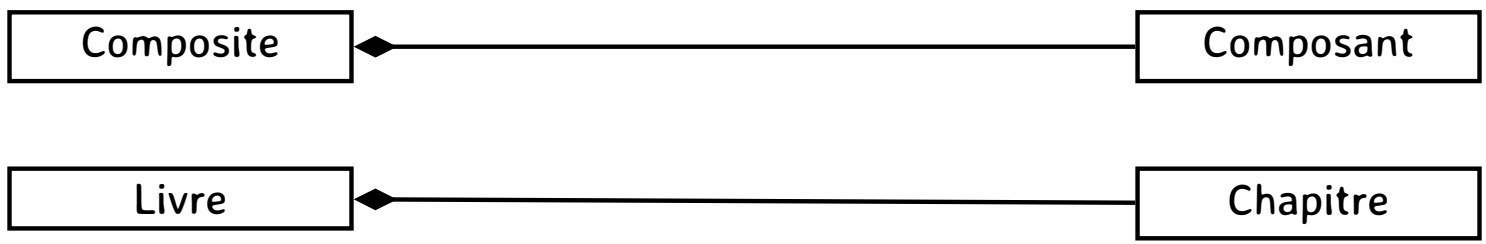



- Les agrégations n'ont pas besoin d'être nommées : implicitement elles signifient « contient », « est composé de ».
- L'agrégation peut être multiple, comme l'association.
- La durée de vie de l'agrégat est indépendante des éléments qui le constituent.
- Un objet constituant peut être partagé par plusieurs agrégats.
  *exemple : L'objet 2 de la classe Entier peut appartenir à plusieurs listes.*

# Composition
## Concept

définition

La composition est une forme particulière de l'agrégation où un élément (composant) ne peut appartenir qu'à un seul agrégat (composite).



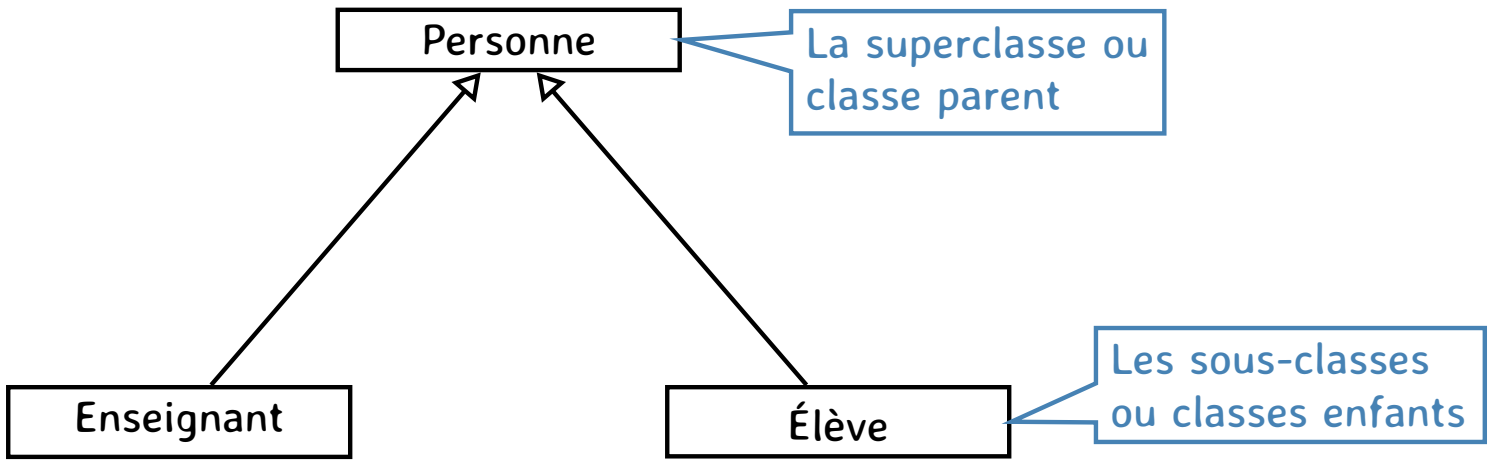

- La relation de composition exprime une contenance structurelle entre des objets
- La composition implique une contrainte sur la valeur de la multiplicité du côté de l'agrégat : elle ne peut prendre que les valeurs 1.
  *(un composant ne peut être partageable)*
- La destruction du composite entraîne la destruction des composants

# UML - La vue statique
## Plan

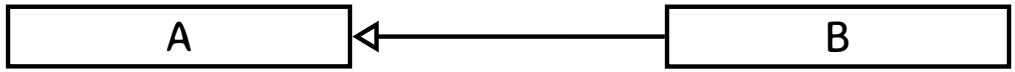
# Généralisation
## Concept

définition

La généralisation/spécialisation permet de définir une relation entre une classe plus générale (appelée super-classe) et une ou plusieurs autres classes plus spécialisées (sous-classes).

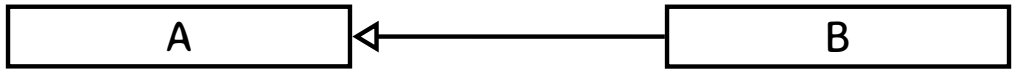


# Généralisation
## Remarques



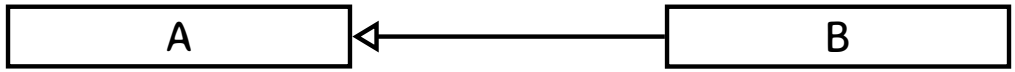

☞ Les sous classes héritent des propriétés, des opérations, des relations et des contraintes définies dans leur super-classe.

# Généralisation
## Remarques



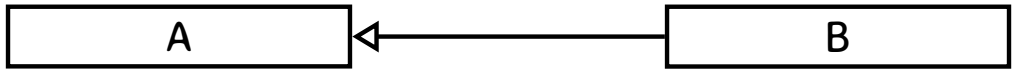

- Les sous classes héritent des propriétés, des opérations, des relations et des contraintes définies dans leur super-classe.
- Il y a la possibilité de redéfinir certains attributs ou certaines opérations.

# Généralisation
## Remarques



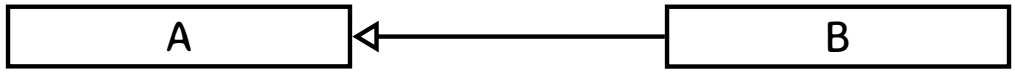

- Les sous classes héritent des propriétés, des opérations, des relations et des contraintes définies dans leur super-classe.
- Il y a la possibilité de redéfinir certains attributs ou certaines opérations.
- La relation de généralisation signifie « est un », « est une sorte de ».

# Généralisation
## Remarques



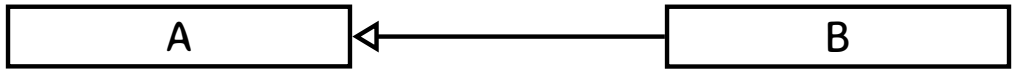

- Les sous classes héritent des propriétés, des opérations, des relations et des contraintes définies dans leur super-classe.
- Il y a la possibilité de redéfinir certains attributs ou certaines opérations.
- La relation de généralisation signifie « est un », « est une sorte de ».
- Cette relation est considérée comme une inclusion entre la superclasse et les sous classes : les objets instances des sous-classes sont objets instances de la superclasse.

# Généralisation
## Remarques

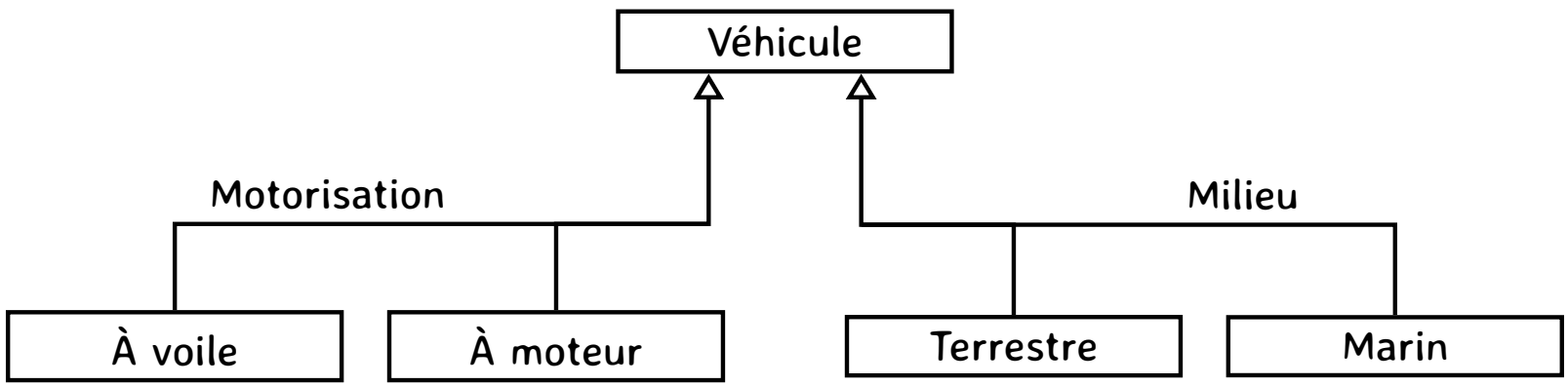



- Les sous classes héritent des propriétés, des opérations, des relations et des contraintes définies dans leur super-classe.
- Il y a la possibilité de redéfinir certains attributs ou certaines opérations.
- La relation de généralisation signifie « est un », « est une sorte de ».
- Cette relation est considérée comme une inclusion entre la superclasse et les sous classes : les objets instances des sous-classes sont objets instances de la superclasse.
- En programmation, la relation de généralisation est très souvent réalisée en utilisant la relation d'héritage entre classes, proposée par les langages objet.

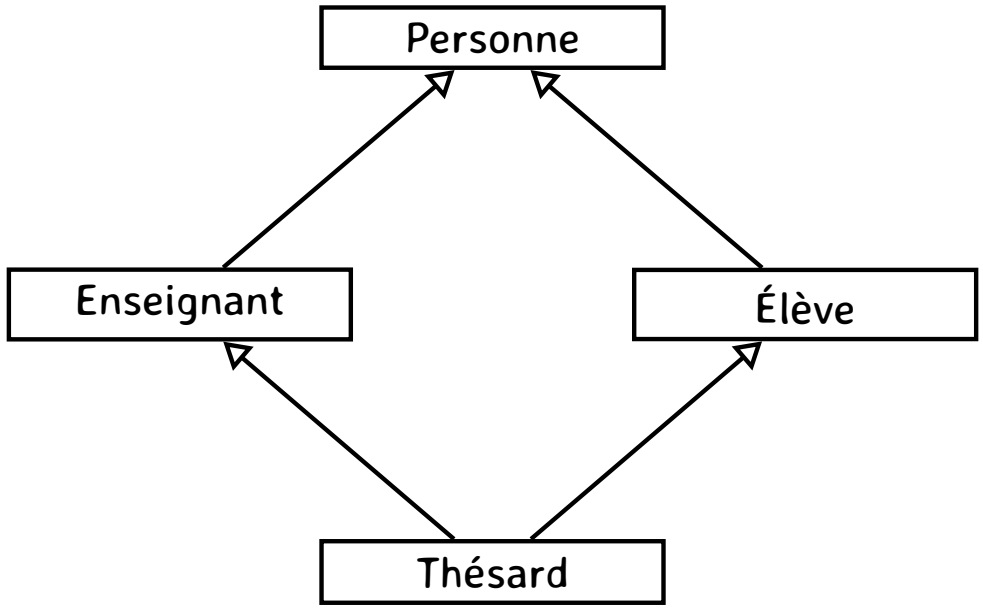
# Généralisation
## Remarques




- Les sous classes héritent des propriétés, des opérations, des relations et des contraintes définies dans leur super-classe.
- Il y a la possibilité de redéfinir certains attributs ou certaines opérations.
- La relation de généralisation signifie « est un », « est une sorte de ».
- Cette relation est considérée comme une inclusion entre la superclasse et les sous classes : les objets instances des sous-classes sont objets instances de la superclasse.
- En programmation, la relation de généralisation est très souvent réalisée en utilisant la relation d'héritage entre classes, proposée par les langages objet.
- Une classe peut être spécialisée selon plusieurs critères.



# Généralisation multiple
## Concept




L'héritage multiple met en évidence des classes spécialisées héritant de plusieurs super-classes.

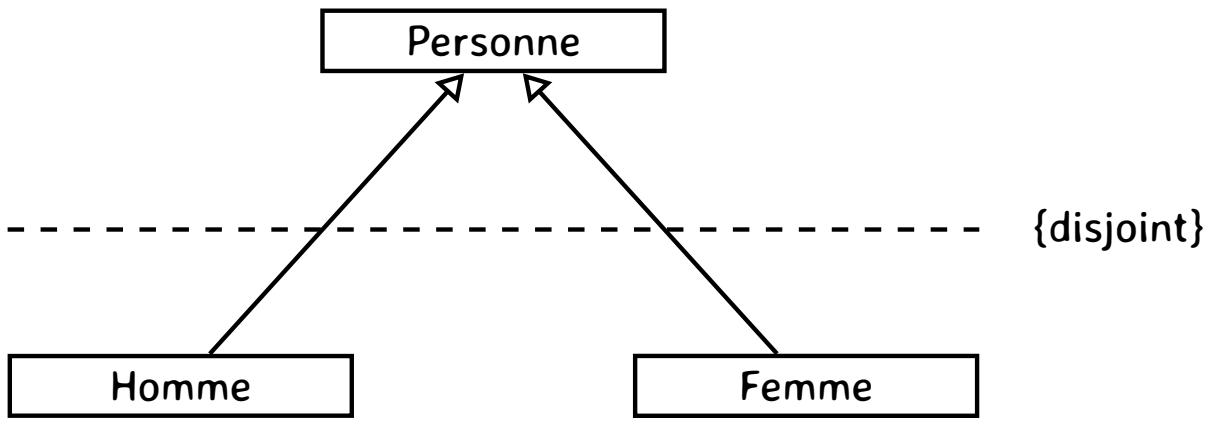
Attention !

Pas possible en Java ☕.

# Généralisations disjointes
## Contrainte

définition

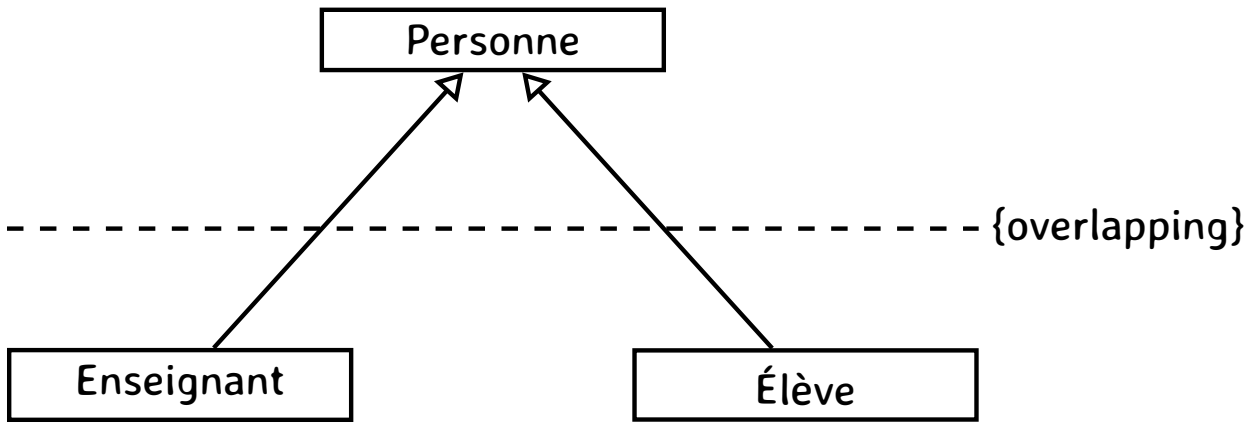
La contrainte de disjonction permet d'exprimer qu'un objet de la superclasse est au plus instance d'une seule des sous-classes.



# Généralisations couvrantes
## Contrainte

définition

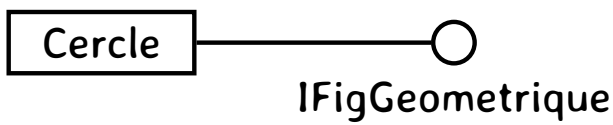
La contrainte de couverture signifie qu'un objet de la superclasse est instance de deux ou plusieurs sous-classes.

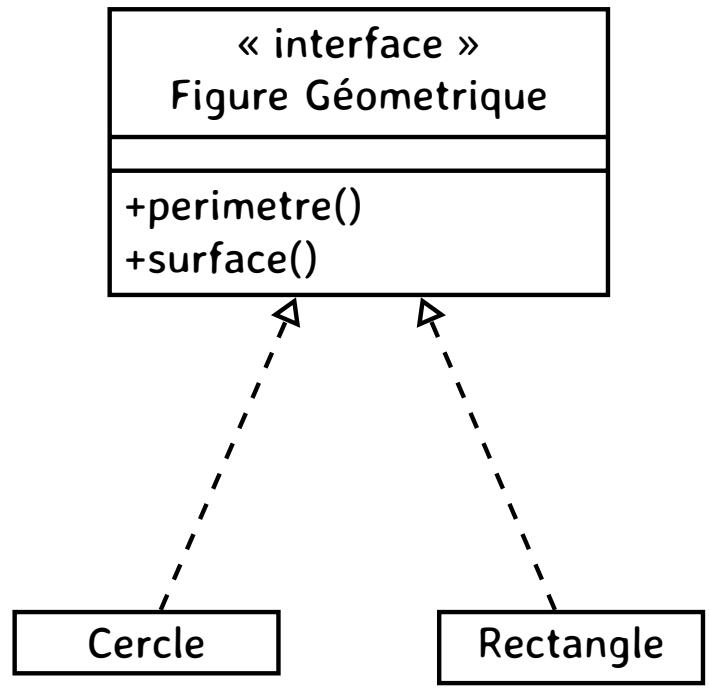


# Interface
## Concept

- Une interface décrit le comportement visible d'une classe.
- Une interface est définie par une liste d'opérations publiques (avec la visibilité +).

**Deux notations :**

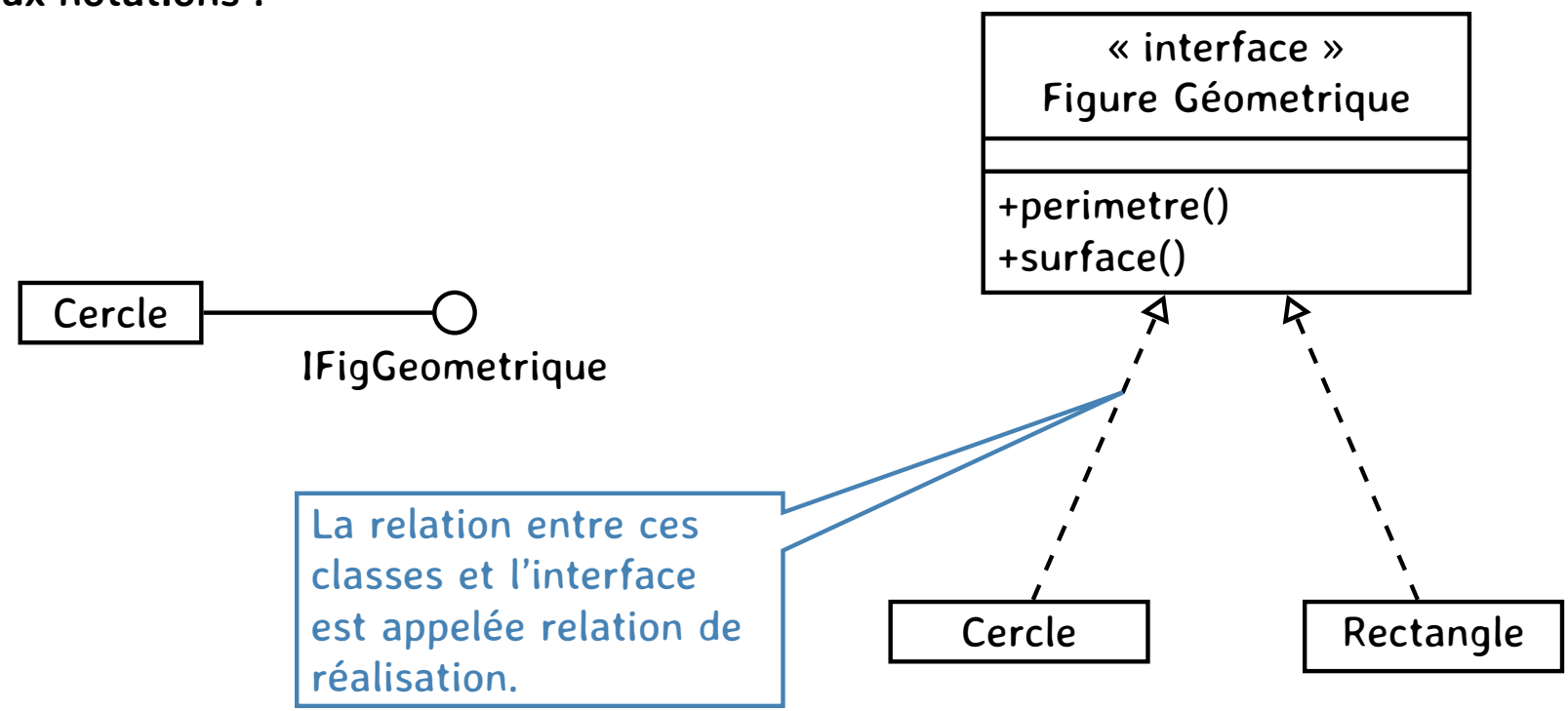


# Interface
## Concept

- Une interface décrit le comportement visible d'une classe.
- Une interface est définie par une liste d'opérations publiques (avec la visibilité +).

**Deux notations :**



# UML - La vue statique
## Plan

# Construction d'un diagramme de classes

- Centré sur les classes métier (les classes du domaine)
  - Met l'accent sur les interfaces de classes plutôt que sur leur contenu
  - Utilise les relations entre classes
- Les classes du domaine ne sont pas les tables de la base de données
- Il se construit de manière empirique
  - Exploiter son expérience dans le domaine de la construction des MCD
  - Exploiter les scénarios des cas d'utilisation, plus particulièrement les interactions entre les acteurs et le système

# Affiner le diagramme de classes

☞ **éliminer des classes :**

- redondantes (qui représentent le même concept) ;
- vagues (ne correspondant pas à des concepts que l'on peut exprimer par des classes) ;
- exprimables par
  - des attributs (elles expriment des concepts quantifiables) ;
  - un rôle (dans une association particulière) ;
- représentant
  - des acteurs à propos desquels le système n'a pas besoin de gérer des informations ;
  - des groupes d'objets (implicites dans la multiplicité des associations) ;
- de conception n'ayant pas de sens dans le métier.

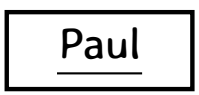
☞ **subdiviser une classe ayant trop de responsabilités**

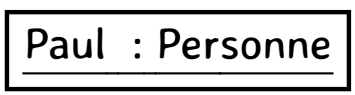
# UML - La vue statique
## Plan

# Diagramme d'objets

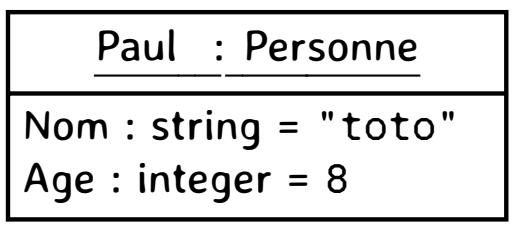
- Ils s'utilisent pour montrer un contexte.
- Ils facilitent la compréhension des structures de données complexes.
- Pratique pour raisonner sur les multiplicités, ou bien sur le partage de données.

syntaxe

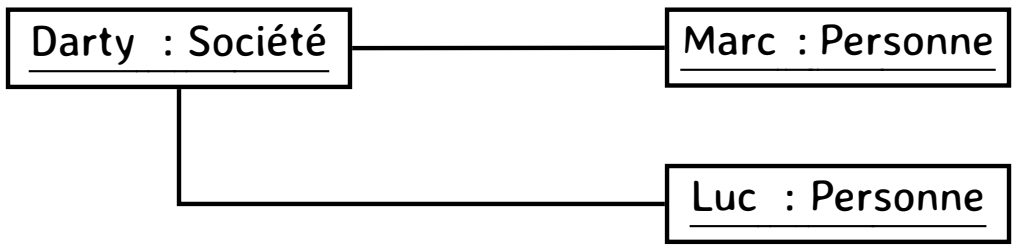




# Diagramme d'objets

définition

Le diagramme d'objets est une instance d'un diagramme de classes. Il montre l'état du système modélisé à un instant donné. Il représente la structure statique.

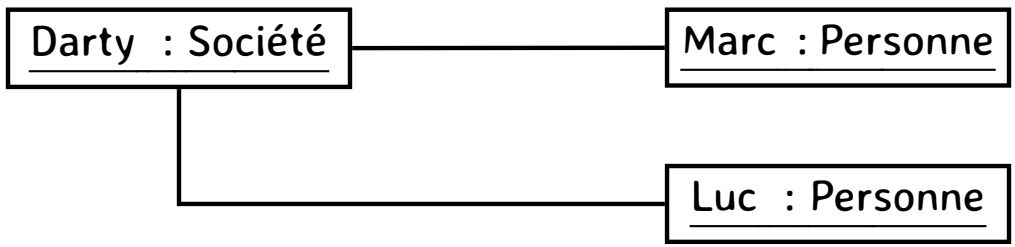



Il est utilisé en analyse pour faciliter la compréhension et vérifier l'adéquation d'un diagramme de classes.

# Diagramme d'objets

définition

Le diagramme d'objets est une instance d'un diagramme de classes. Il montre l'état du système modélisé à un instant donné. Il représente la structure statique.

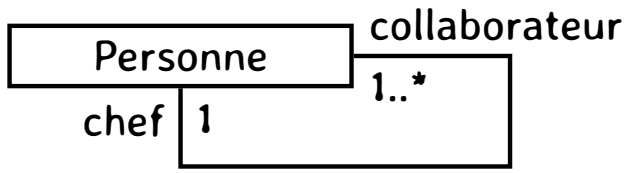


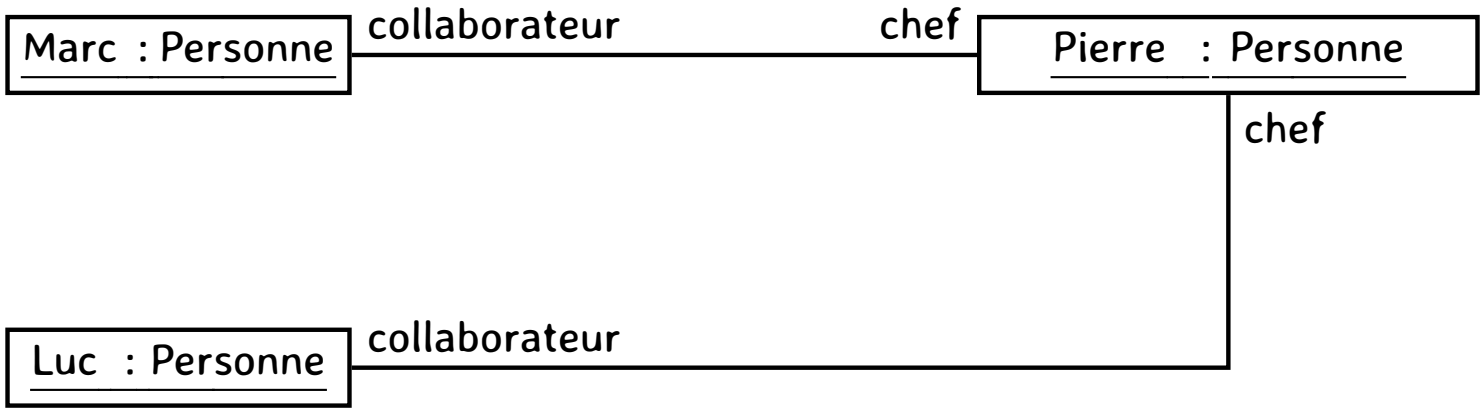

**Quel est le diagramme de classes correspondant ?**

Il est utilisé en analyse pour faciliter la compréhension et vérifier l'adéquation d'un diagramme de classes.

# Diagramme d'objets

Illustrer certains aspects du diagrammes de classes.



# Diagramme de paquetage

- Les systèmes impliquent souvent la manipulation d'un nombre élevé de classes et autres éléments du modèle.
- En UML, un paquetage (package) est un mécanisme générique pour organiser des éléments de modélisation en groupes.

concepts

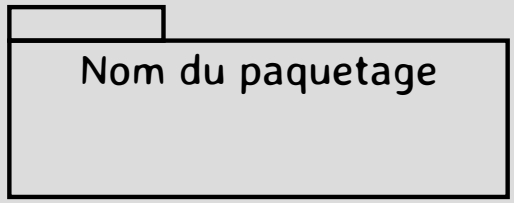
- Paquetage
- Relations entre paquetages :
  - contenance
  - dépendance

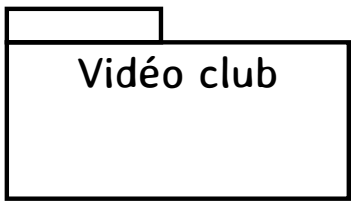
# Paquetage
## Concept

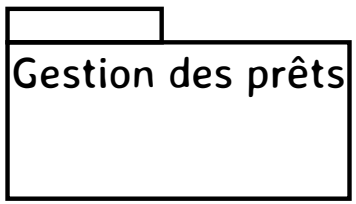
définition

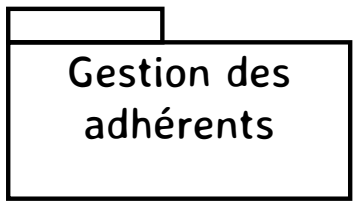
Un paquetage est défini comme un conteneur d'éléments de modélisation reliés entre eux. Il peut contenir des classes et des associations entre classes.

syntaxe



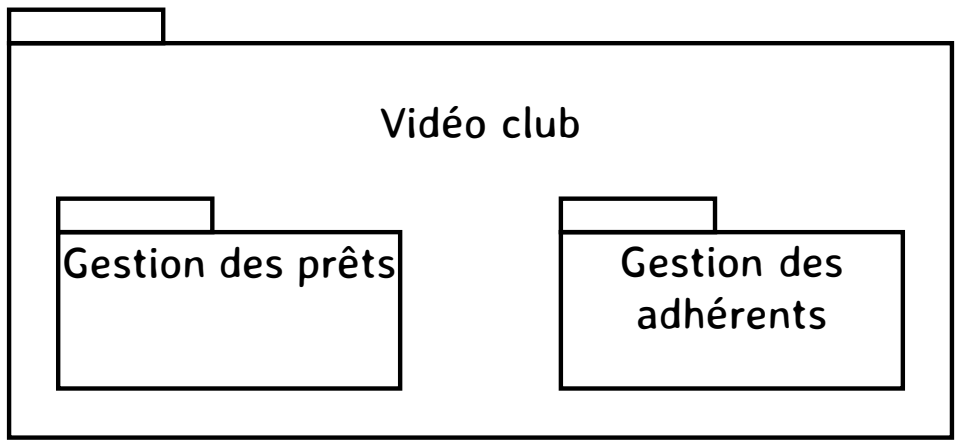



# Contenance
## Concept

définition

La relation de contenance entre paquetage indique qu'un paquetage peut contenir deux ou plusieurs paquetages.

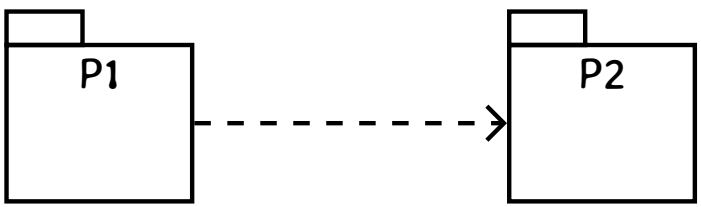


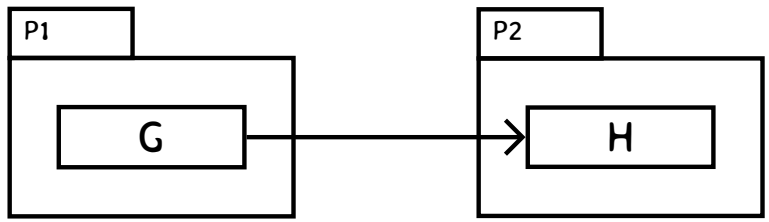

- Chaque élément du modèle appartient à un paquetage.
- Le paquetage de plus haut niveau est le paquetage racine de l'ensemble d'un modèle.
- Le système complet est une hiérarchie de paquetages.

# Dépendance
## Concept

définition

La relation de dépendance entre paquetages indique que les éléments du paquetage P1 dépendent des éléments du paquetage P2.




- La classe G nécessite la classe H pour sa mise en œuvre ou son fonctionnement.
- Les classes G & H sont liées par une association, il est nécessaire d'indiquer la navigabilité de cette association.
- La navigabilité indique le sens privilégié de parcours de l'association.
- La nature de l'utilisation peut être :
  - l'invocation d'une opération
  - la création d'un objet.