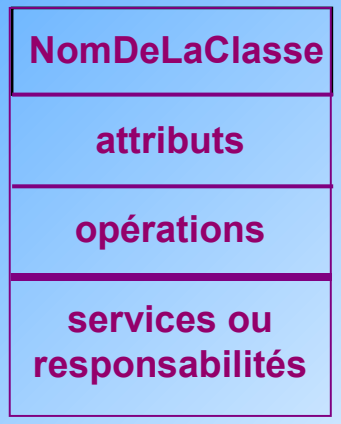
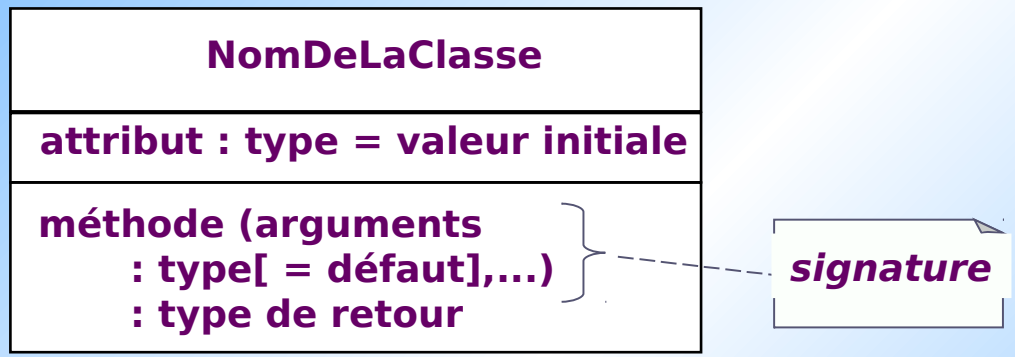
DIAGRAMME DE CLASSES

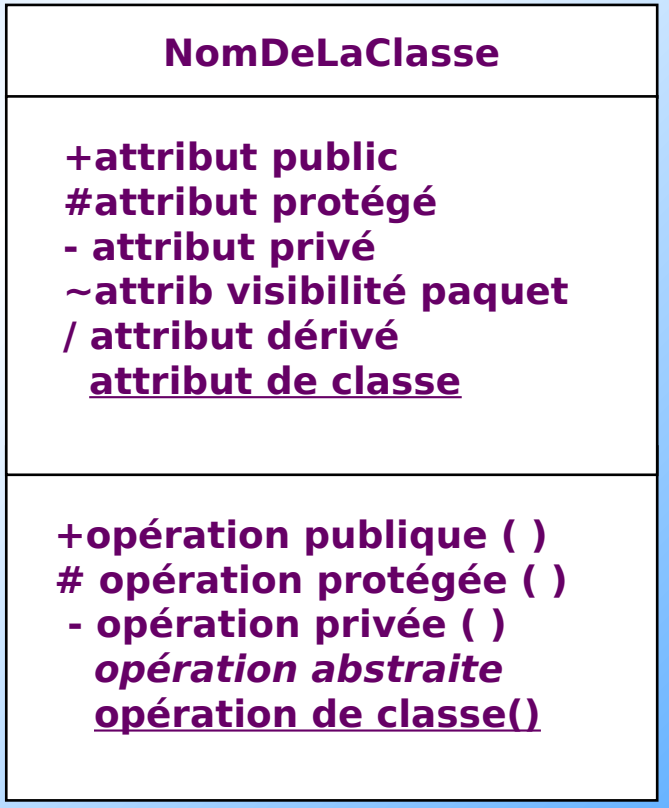
Diagramme statique ou de structure

Représentation des classes





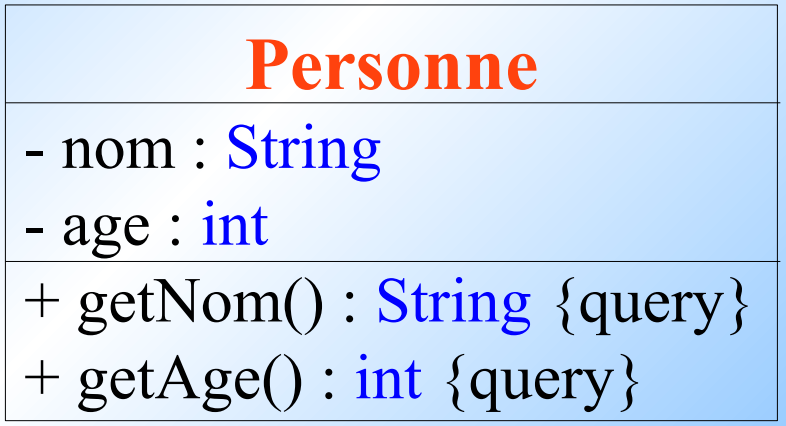
Représentation des classes



Visibilité des attributs et des opérations

Opérations de type requête

Ce sont des opérations dont le code ne modifie pas la classe donc **aucune modification des attributs de l'objet**
à l'exception d'attributs internes non significatifs pour l'utilisateur de l'objet.



CONSTRUCTEURS ET DESTRUCTEURS

Possibilité d'utiliser les stéréotypes :

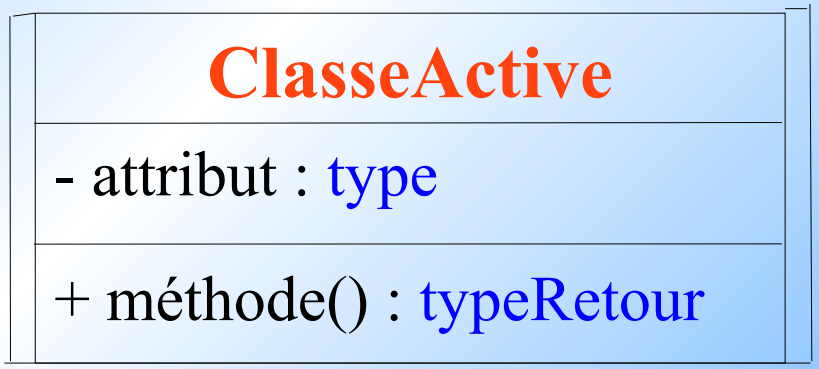
- <<create>> : constructeur
- <<destroy>> : destructeur

Personne
- nom : String - age : int
+ <<create>> Personne() + <<create>> Personne(nom : String) + <<destroy>> finalize() : void

CLASSE ACTIVE

Par défaut, une classe est passive.

- Classe **passive** : sauvegarde les données et offre des services aux autres. Elle fournit des structures de données et des comportements.
- Classe **active** : initie et contrôle le flux d'activités. Ce sont les classes qui contrôlent le système aussi appelé « main ».



ATTRIBUT DÉRIVÉ

- Un **attribut dérivé** est un attribut dont la **valeur** peut être **déduite** d'autres informations disponibles dans le modèle, par exemple d'autres attributs de la même classe, ou de classes en association.

- L'analyste garde cet attribut, pouvant être considéré comme redondant, s'il correspond à un concept important aux yeux de l'expert métier.

- **Exemple** :

L'*âge* d'une personne est un attribut dérivé de l'attribut *date de naissance*.

Notation : /âge

ATTRIBUT DE CLASSE

Par défaut, un attribut a une portée d'instance : chaque objet de la classe possède sa propre valeur pour la propriété.
Dans certains cas, l'attribut peut avoir une **portée de classe** : il existe alors une seule valeur commune de la propriété pour toutes les instances de la classe.
On parle dans ce cas d'attribut de classe, et on le souligne pour le distinguer des attributs d'instance.

attribut → *portée d'instance*

montant facture

attribut classe → *portée de classe*

TVA

ATTRIBUT DE CLASSE

- **Exemple** : la valeur de la variable PI = *3,14* définie dans la classe Math du langage Java.

- Cette valeur est accessible même s'il n'existe aucun objet de la classe Math.

- PI est un attribut de la classe Math et non un attribut de ses instances. Les instances ont accès à cet attribut, mais n'en possèdent pas une copie.

- En Java comme en C++, un attribut de classe s'accompagne du mot-clé *static*.

CONTRAINTES ATTRIBUT

Multiplicité : multiplicité pour définir des tableaux, des listes de valeurs

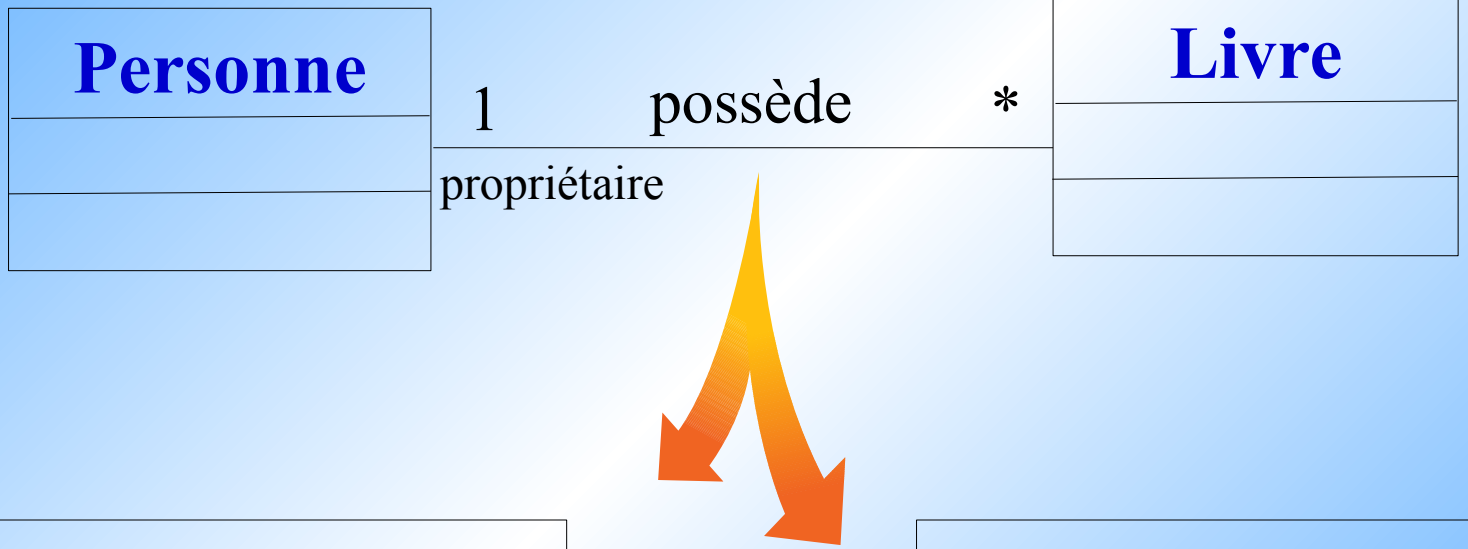
[*], [1..*], [valeur], [min .. max]

Elle est omise si la multiplicité est de 1.

Ordre : {ordered}

Annuaire
- membres : Personne[*] {ordered}

CONTRAINTES ATTRIBUT



CONTRAINTES ATTRIBUT

Unicité : les valeurs de l'attribut n'ont pas de doublons.

{unique}
{not unique}

On peut trouver cette contrainte pour les valeurs retournées par une méthode.

Personne
- noSécurité : int {unique} - nom : String - prénoms : String[1..3] - dateNaissance : Date
+getPrénoms () : String[1..3] {unique, ordered}

CONTRAINTES ATTRIBUT

Types collection :

Bag ou Sac (*pas ordre, pas unicité*)
OrderedSet ou Ensemble ordonné (*ordre, unicité*)
Set ou Ensemble (*pas ordre, unicité*)
Sequence ou Séquence (*ordre, pas unicité*)

RegistreDesVotants
- votants : Personne [*] {OrderedSet}

Chaque votant n'est autorisé qu'à voter une seule fois.
Les votants sont classés par ordre alphabétique.

CONTRAINTES ATTRIBUT

frozen : attribut non modifiable une fois initialisé

readOnly : attribut modifiable à l'intérieur de la classe
mais ne peut être changé de l'extérieur de la classe

Contraintes diverses : {attribut > 0}, {précondition}

Les gens commentent des erreurs. Si on gèle un attribut tel que la date de naissance, on ne pourra pas le modifier si une erreur de saisie a été commise.

CONTRAINTES ATTRIBUT

Employé
n°employé : int {unique}{frozen} poste : String salaire : real {readOnly}

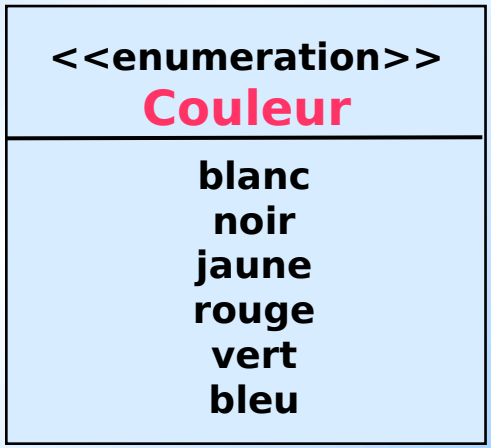
Le salaire ne peut être modifié à l'extérieur de la classe Employé.

ÉNUMÉRATION

- Une **énumération** ou **type énuméré** est un type possédant un nombre fini et arbitraire de valeurs possibles.

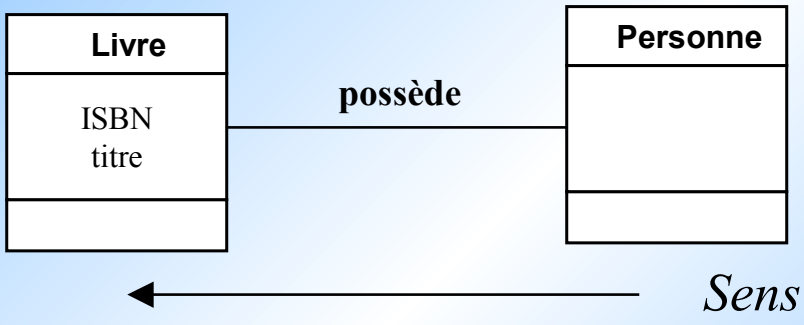
- Classeur stéréotypé <<enumeration>>

- **Exemples** : les jours de la semaine, les mois.



IDENTIFICATION DES CONCEPTS

Exemple :
Une personne peut posséder des livres. La relation *possède* est une association entre les classes Personne et Livre.



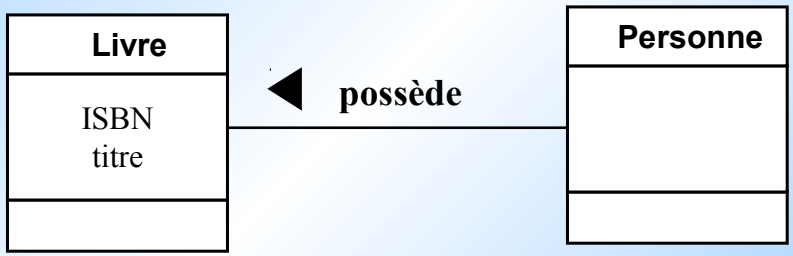
Par défaut bidirectionnelle

Sens lecture privilégié

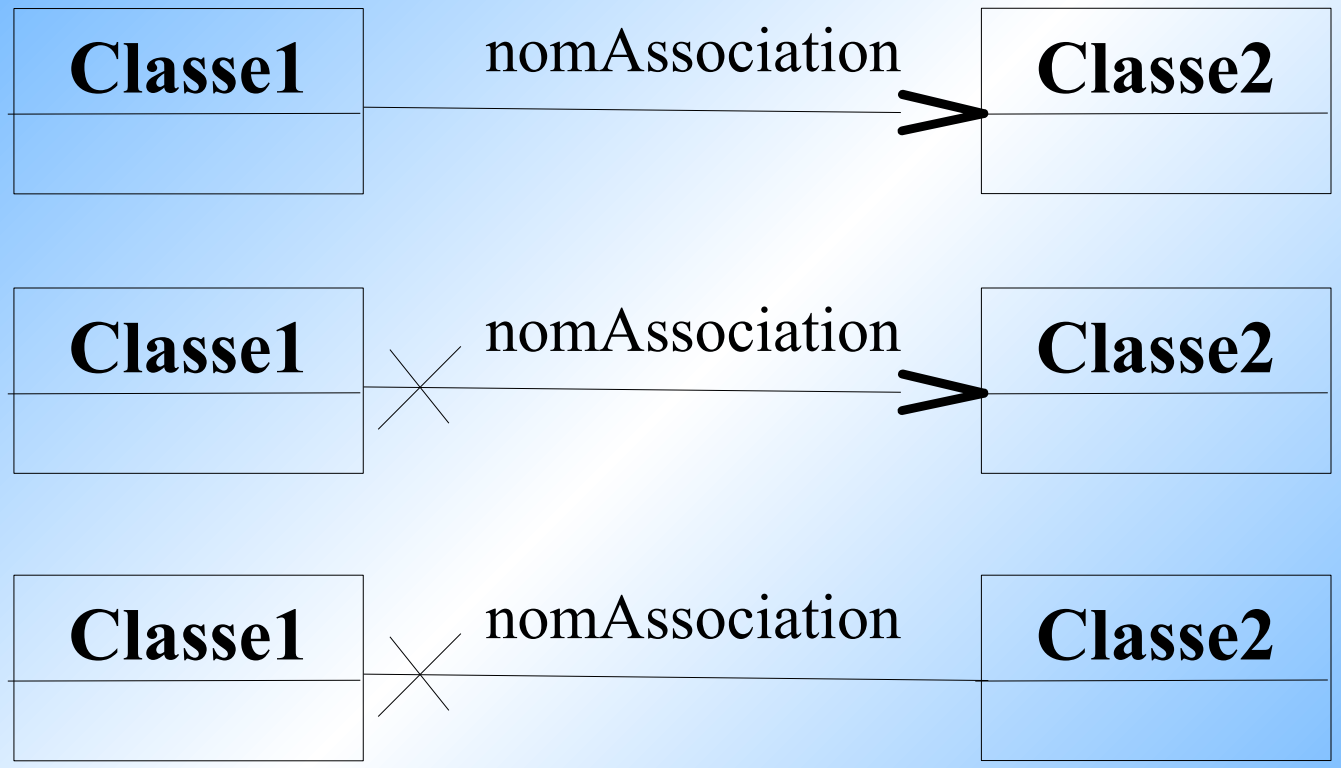
Même si le **verbe** qui nomme une association semble **privilégier un sens de lecture**, une **association** entre concepts dans un modèle du domaine est **par défaut bidirectionnelle**.
Donc, implicitement, l'exemple précédent inclut également le fait qu'un livre est possédé par une personne.

Sens de lecture des associations

- On peut faciliter la lecture des diagrammes en indiquant le sens de lecture des associations par le symbole : ▶



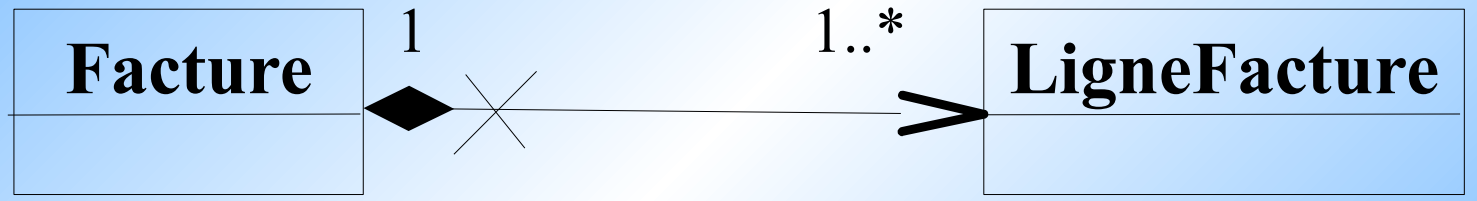
Navigabilité des associations



même sémantique

Navigabilité des associations

Exemple



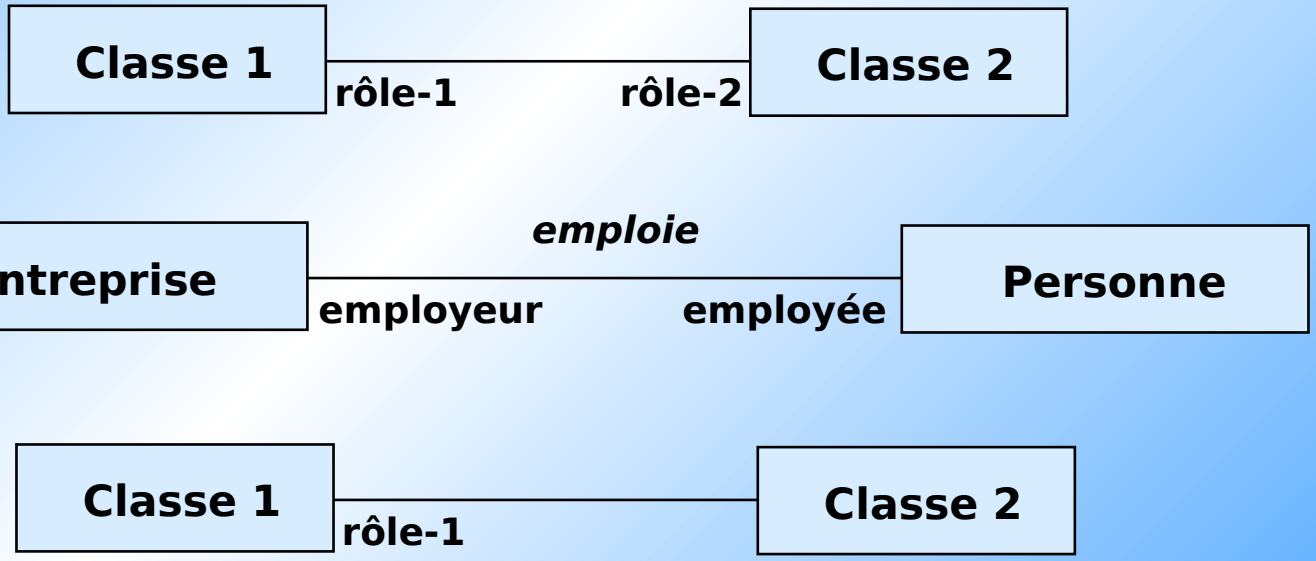
ASSOCIATION DÉRIVÉE

Les **associations dérivées** obéissent au même principe que les attributs dérivés : ce sont des associations redondantes mais considérées comme pertinentes par l'expert du domaine.

RÔLE

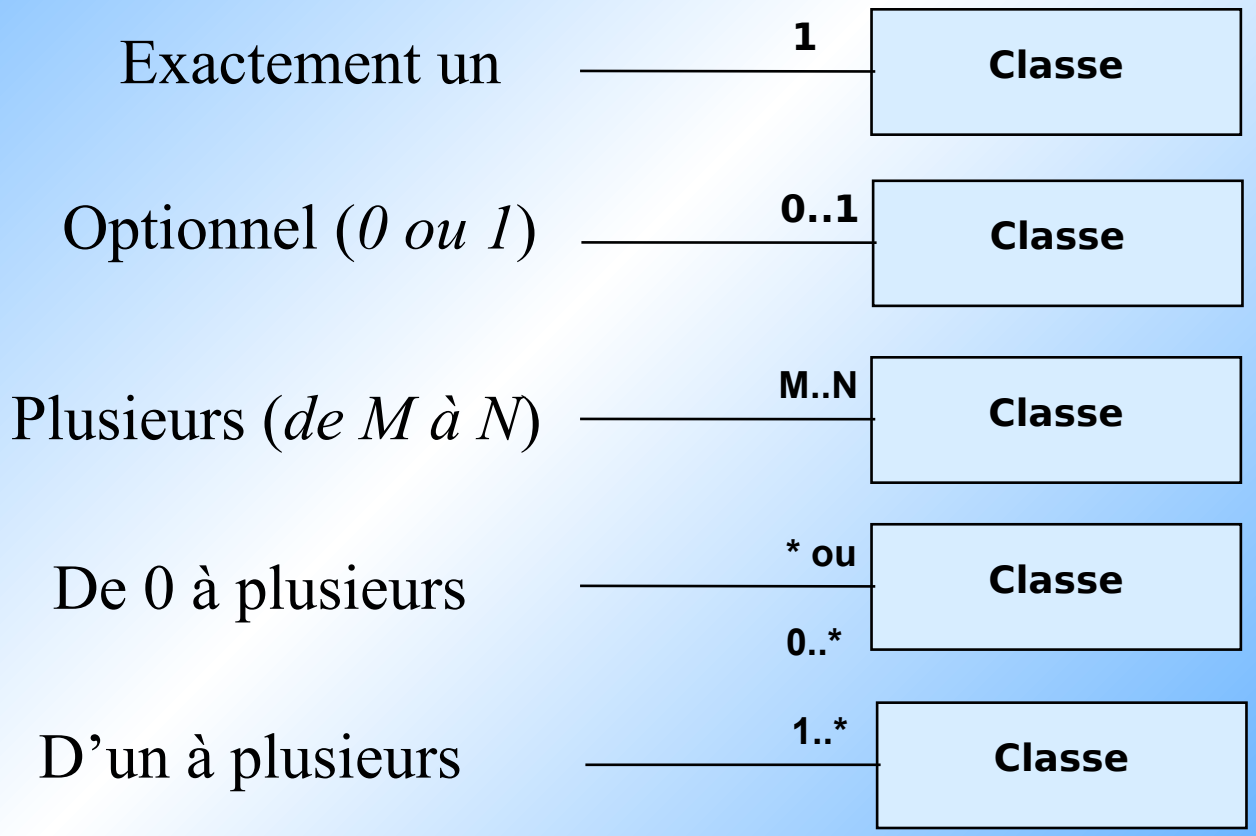
- Le **rôle** décrit comment une classe voit une autre classe au travers d'une association.

- Se place à une extrémité de l'association



CARDINALITÉS

Montrent combien d'objets d'une classe peuvent être liés à un objet de l'autre classe.



IMPLÉMENTATION


```
public class ClasseA {
  private Set<ClasseB> roleB ;
  ...
}
```

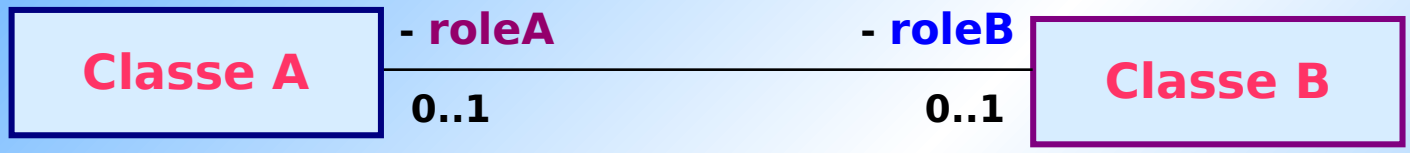
```
public class ClasseB {
  private ClasseA roleA ;
  ...
}
```

IMPLÉMENTATION


```
public class ClasseA {
  private Set<ClasseB> roleB ;
  ...
}
```

```
public class ClasseB {
  // ne connaît pas ClasseA
  ...
}
```

IMPLÉMENTATION



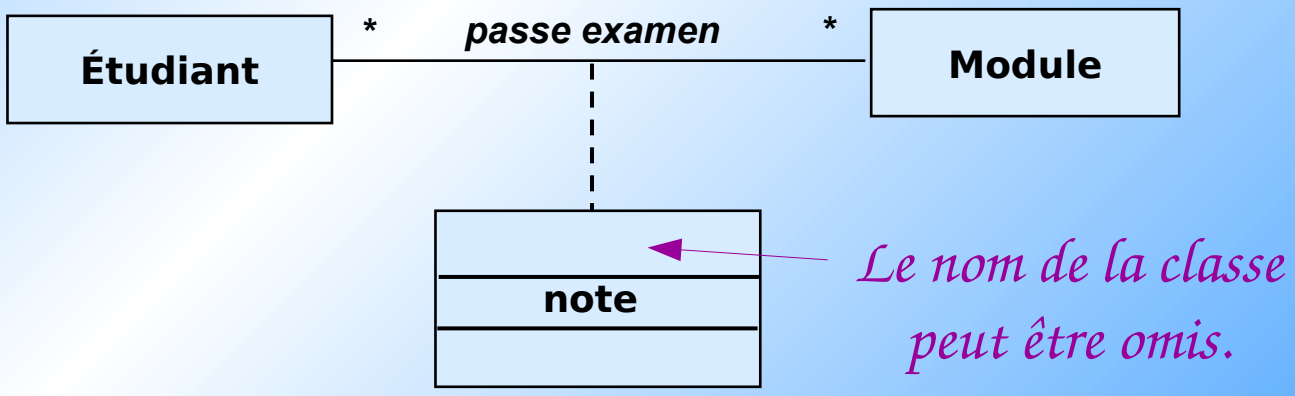

```
public class ClasseA {
  private ClasseB roleB ;
  ...
}
```

```
public class ClasseB {
  private ClasseA roleA ;
  ...
}
```

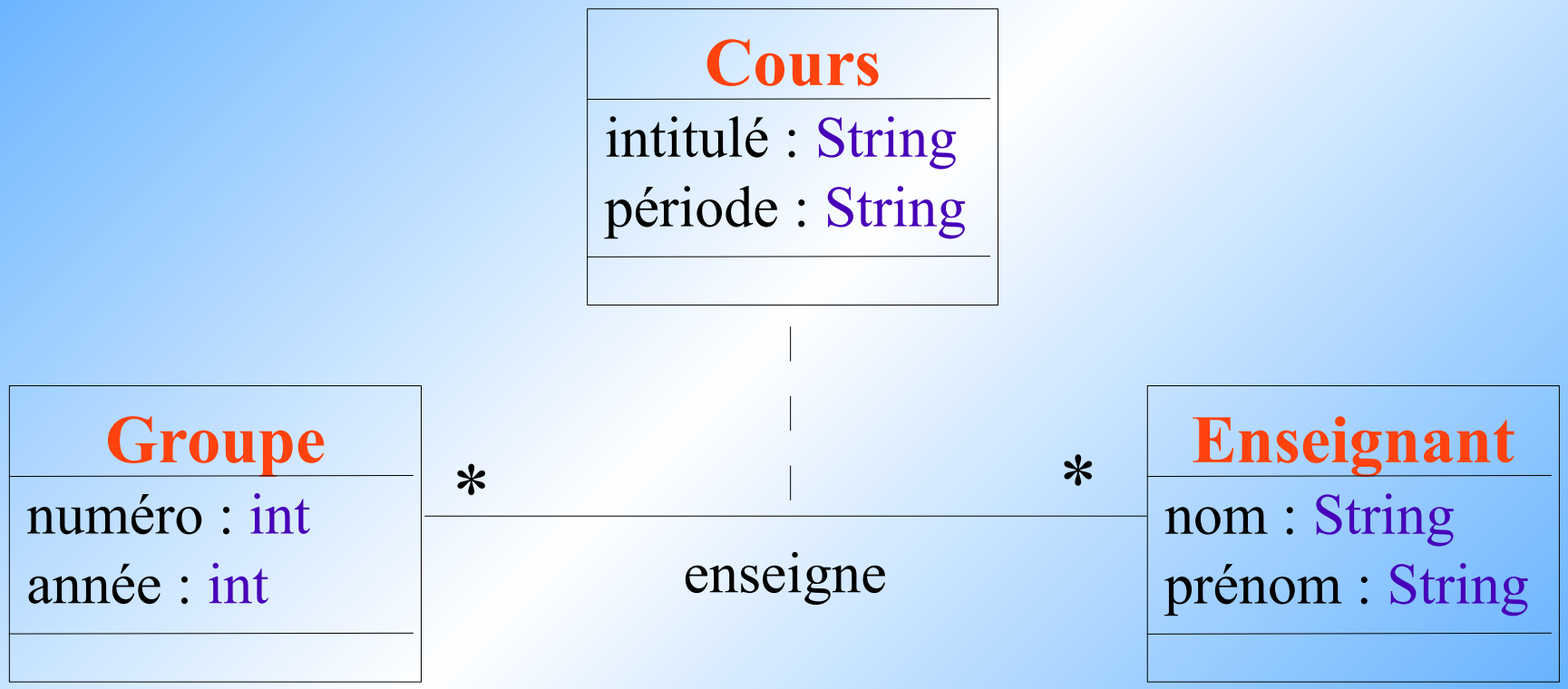
ASSOCIATION ATTRIBUÉE

- Une **association attribuée** est une association qui contient des attributs sans participer à des relations avec d'autres classes.

- Une association peut être raffinée et avoir ses propres propriétés, qui ne sont disponibles dans aucune des classes qu'elle lie.

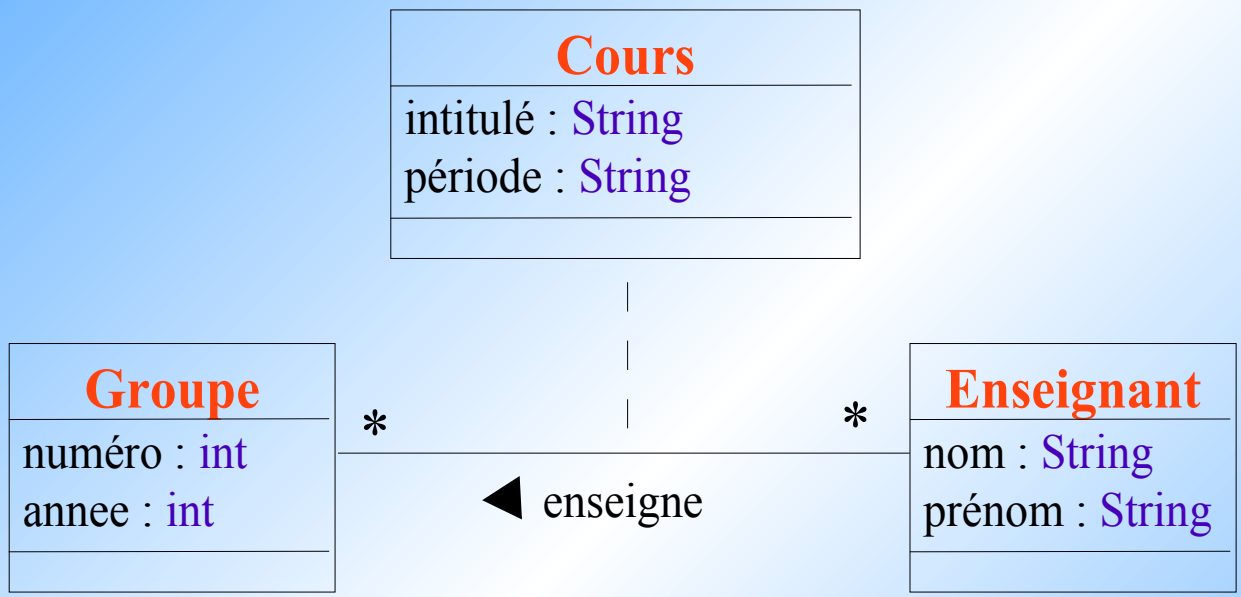


Association attribuée



Quelle est la contrainte sur Cours ?

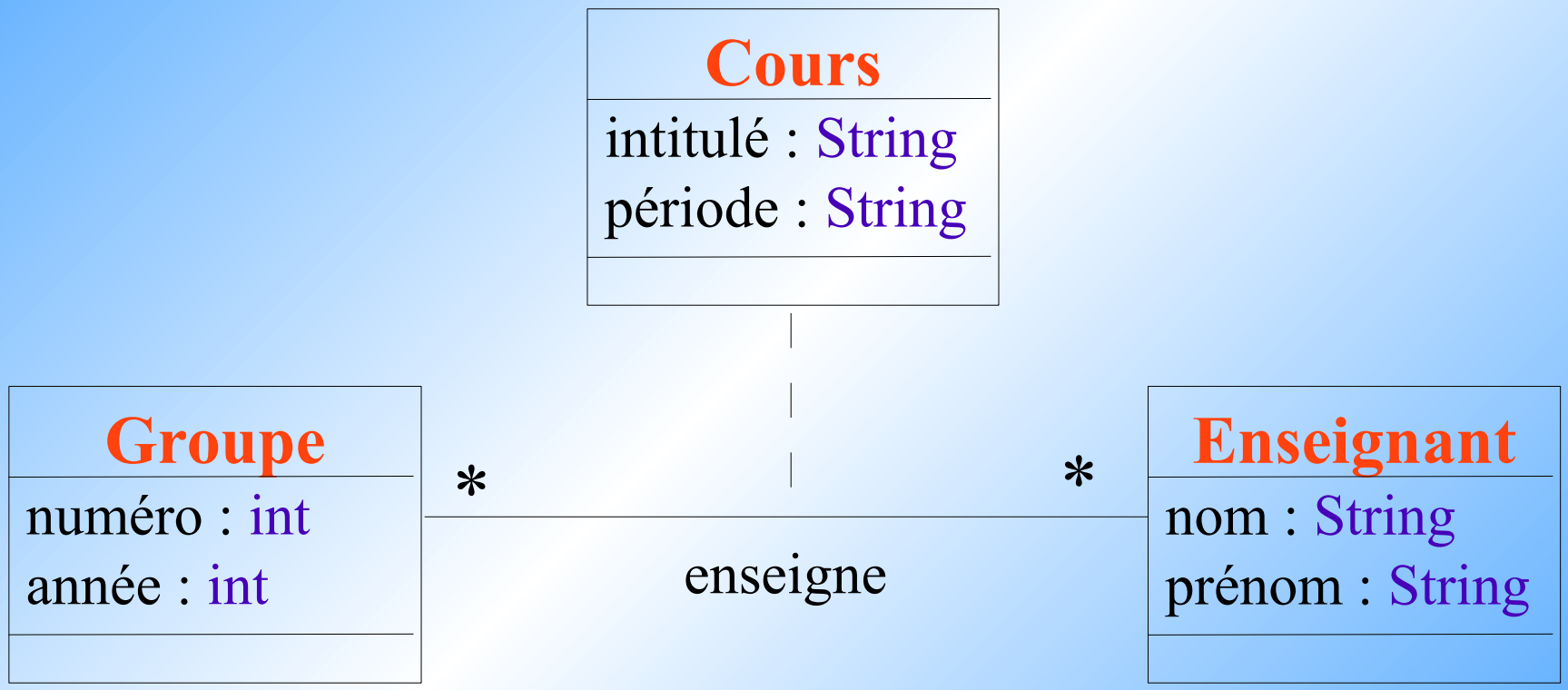
SOLUTION



Un cours ne peut exister sans un lien Groupe – Enseignant

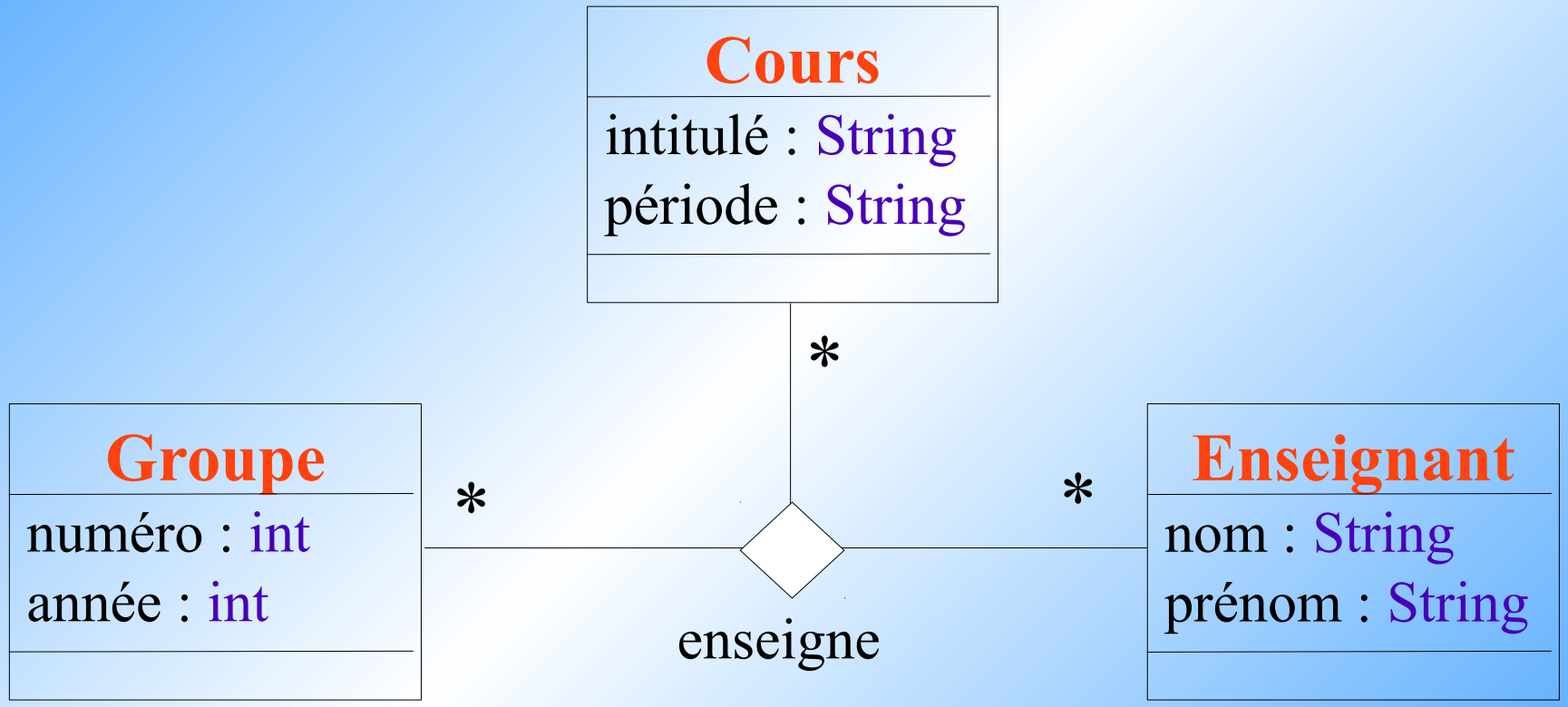
Un enseignant ne peut enseigner qu'un cours à un groupe donné

Association attribuée

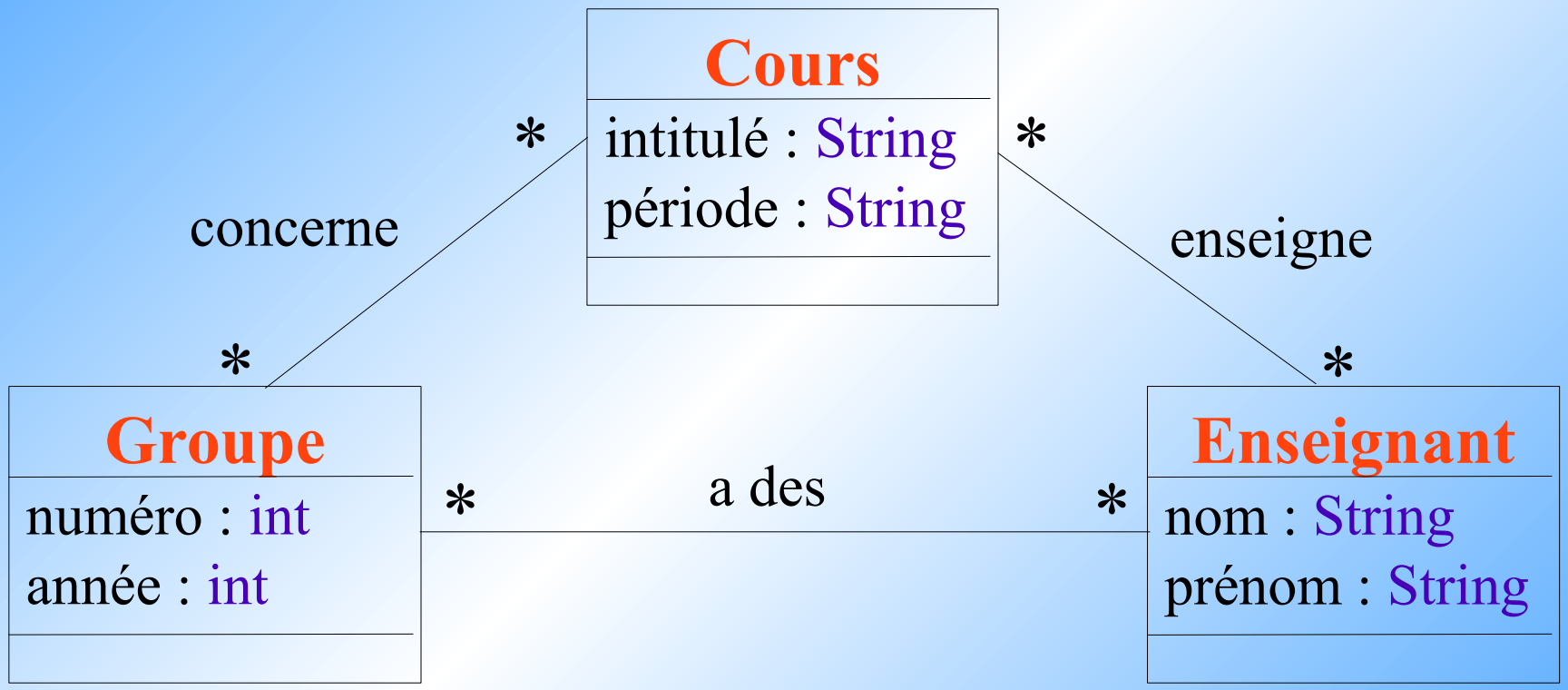


Comment y remédier ?

SOLUTION

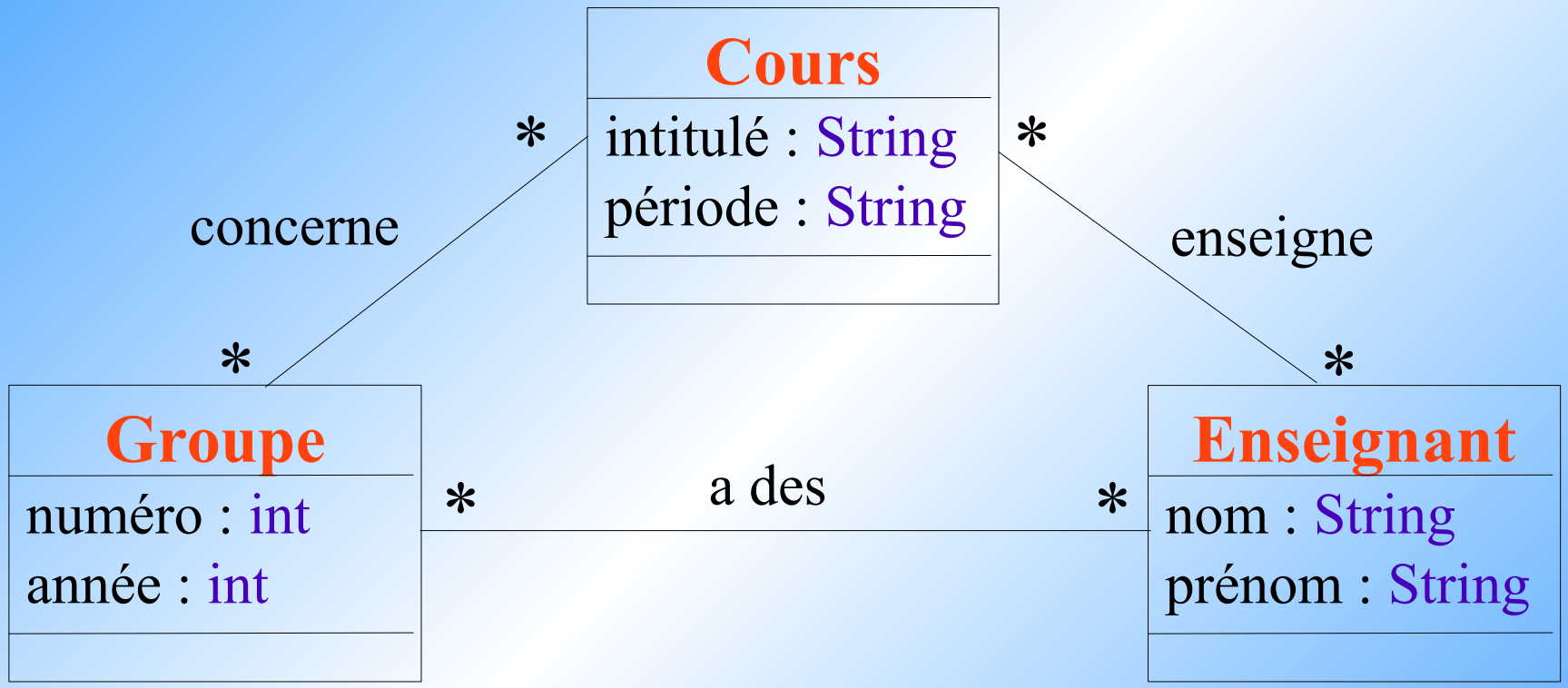

Cours
intitulé : String
période : String
*
Groupe
numéro : int
année : int
*
*
Enseignant
nom : String
prénom : String
enseigne

Association attribuée



Que pensez-vous de ce modèle ?

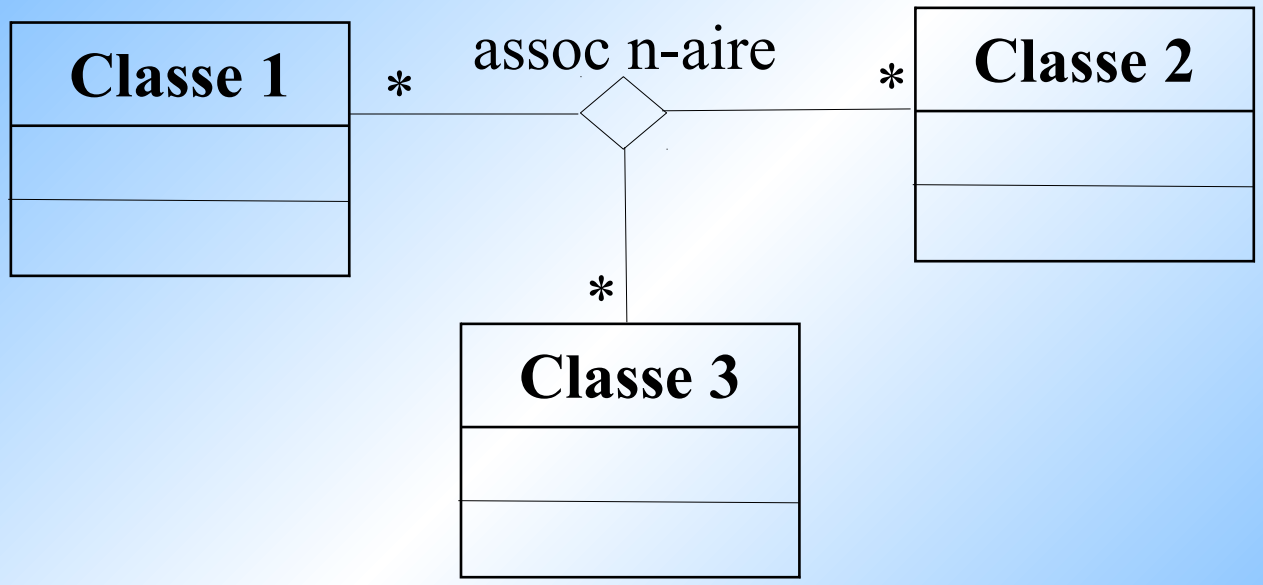
SOLUTION



On ne sait pas quels groupes a un enseignant dans un cours donnée.

ASSOCIATION N-AIRE

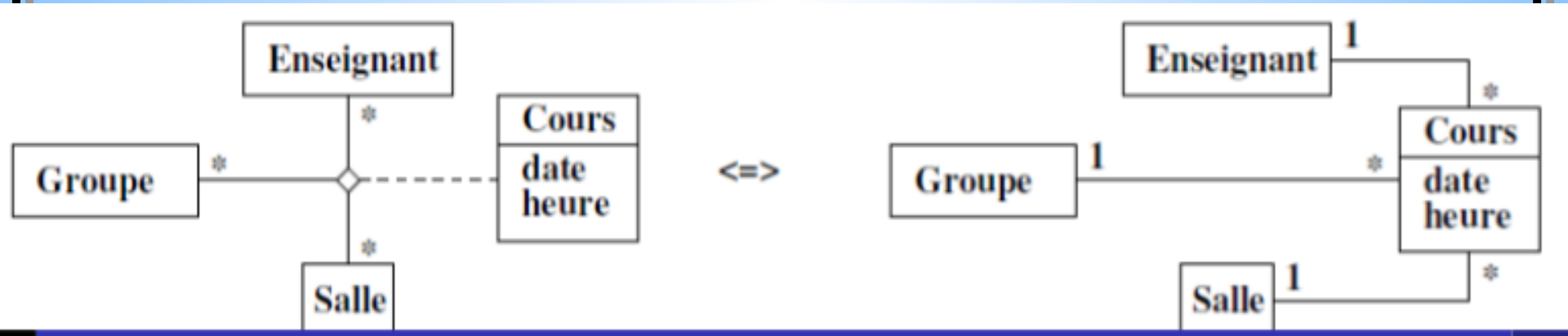
- Une **association n-aire** lie plus de *2* classes.



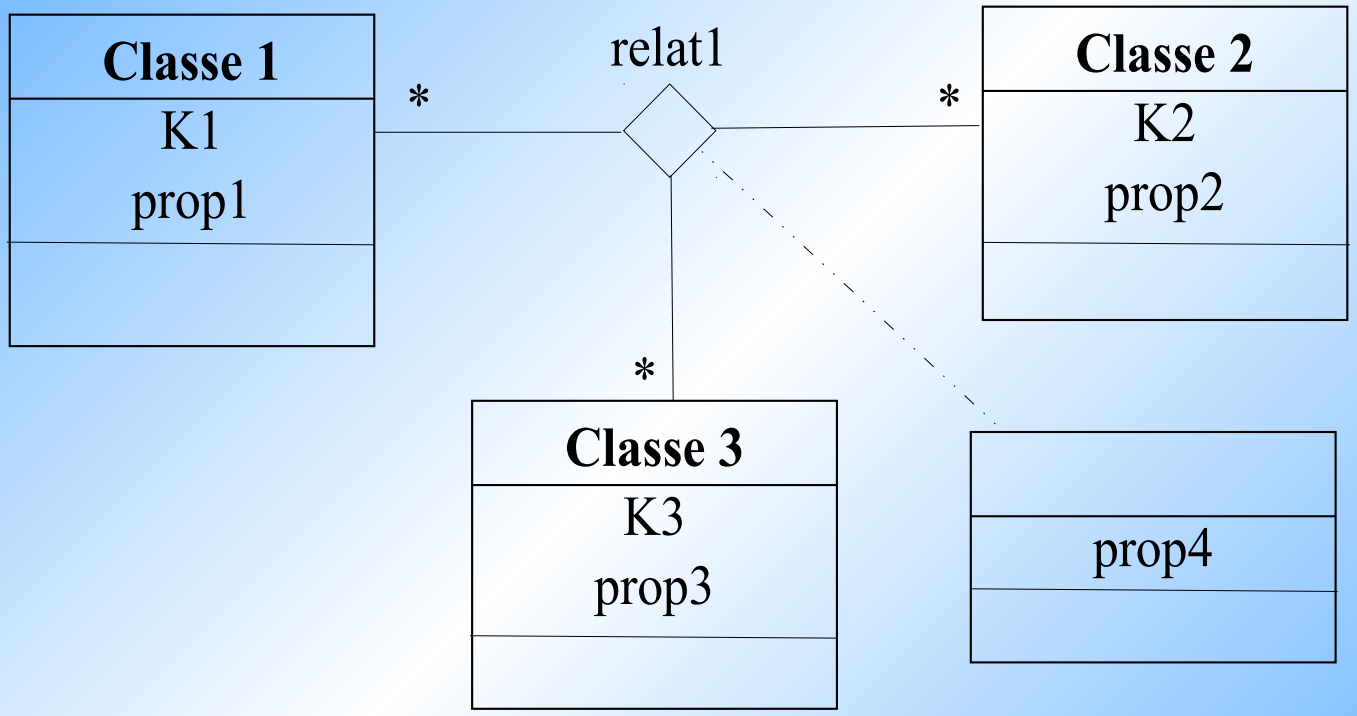
- Les multiplicités en UML sont
  - ✓ "*à l'envers*" (*par référence à Merise*) pour les associations binaires et
  - ✓ "*à l'endroit*" pour les associations n-aires (*n* > *2*).

Association n-aire

- Association n-aire quand on ne peut pas faire autrement.
- Privilégier toujours les associations binaires.



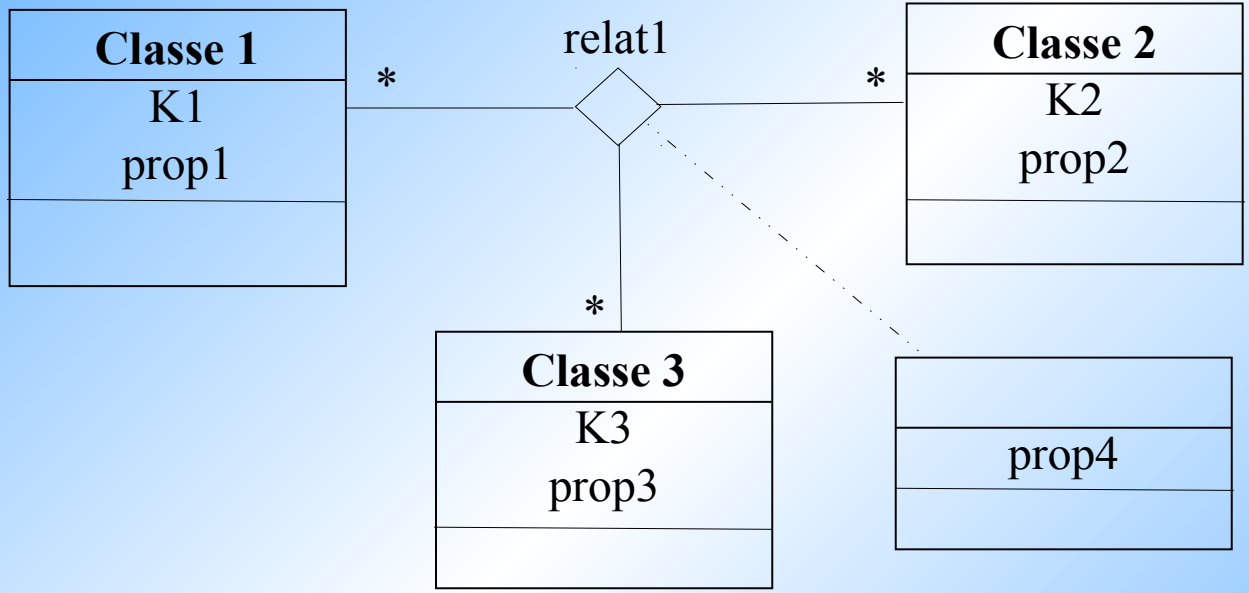
ASSOCIATION N-AIRE



Quelle contrainte a-t-on avec la propriété "prop4" ?

K1,K2,K3 → prop4

SOLUTION



Pour un triplet (K1, K2, K3), si K1, K2 et K3 sont des identifiants respectivement des classes Classe1, Classe2 et Classe3,

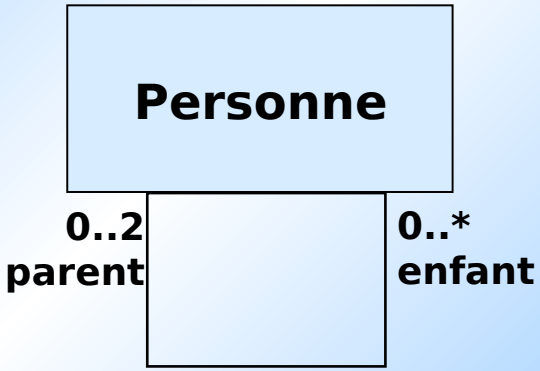
il n'y a qu'une valeur pour "prop4"

Privilégier
les associations binaires
aux associations n-aires

- plus simples à interpréter
- plus simples à implémenter
- autorisent la qualification
- autorisent la navigation

Association réflexive

Les associations peuvent relier une classe à elle-même.
Le nommage des rôles est ici très important pour distinguer les instances qui participent à la relation.

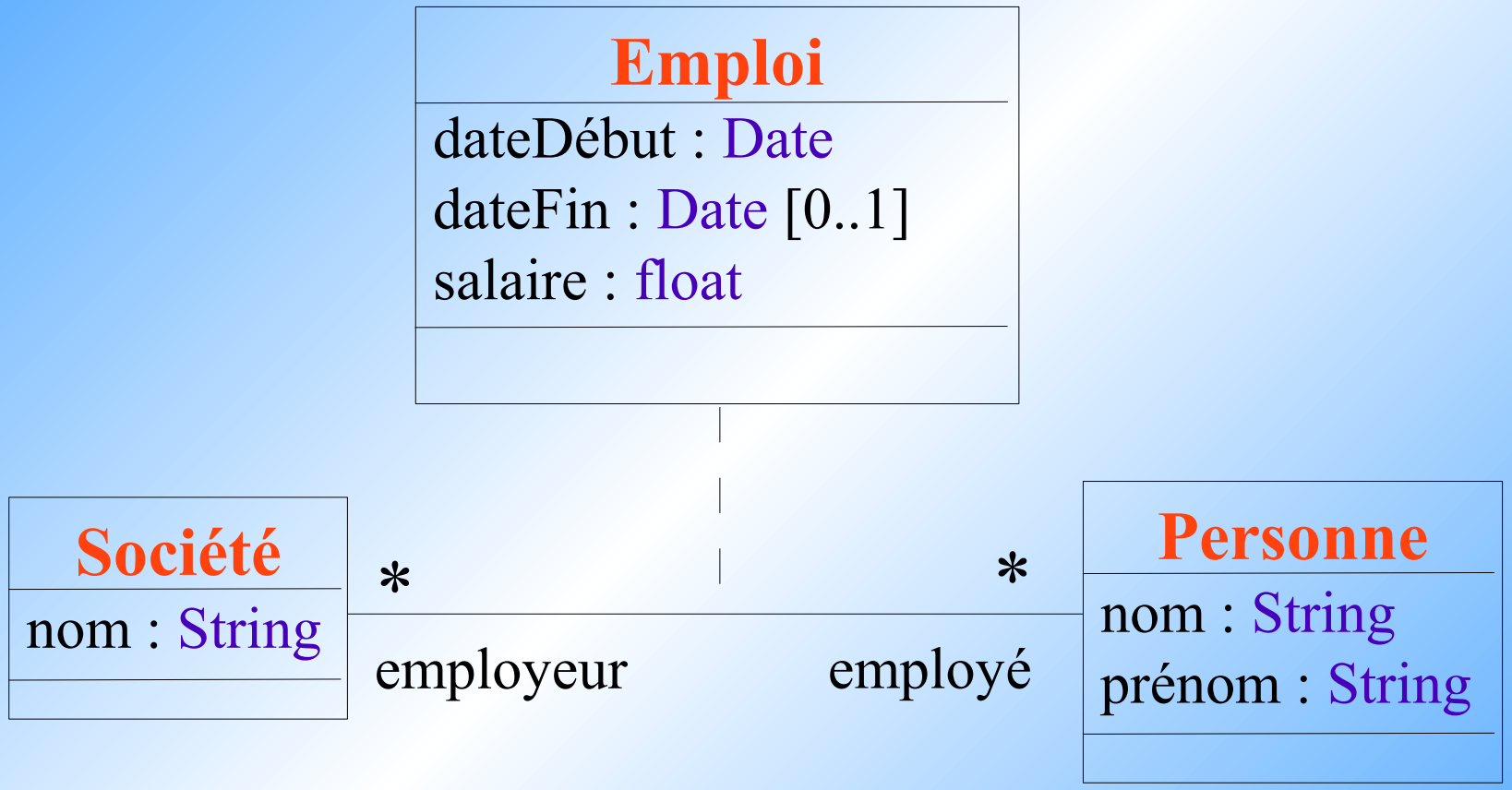


Une personne (*parent*) a combien d'enfants ? : *0* à plusieurs.
Une personne (*enfant*) a combien de parents ? : *0* à *2*

EXERCICE

Modéliser les derniers différents emplois d'un salarié dans ses diverses entreprises

SOLUTION

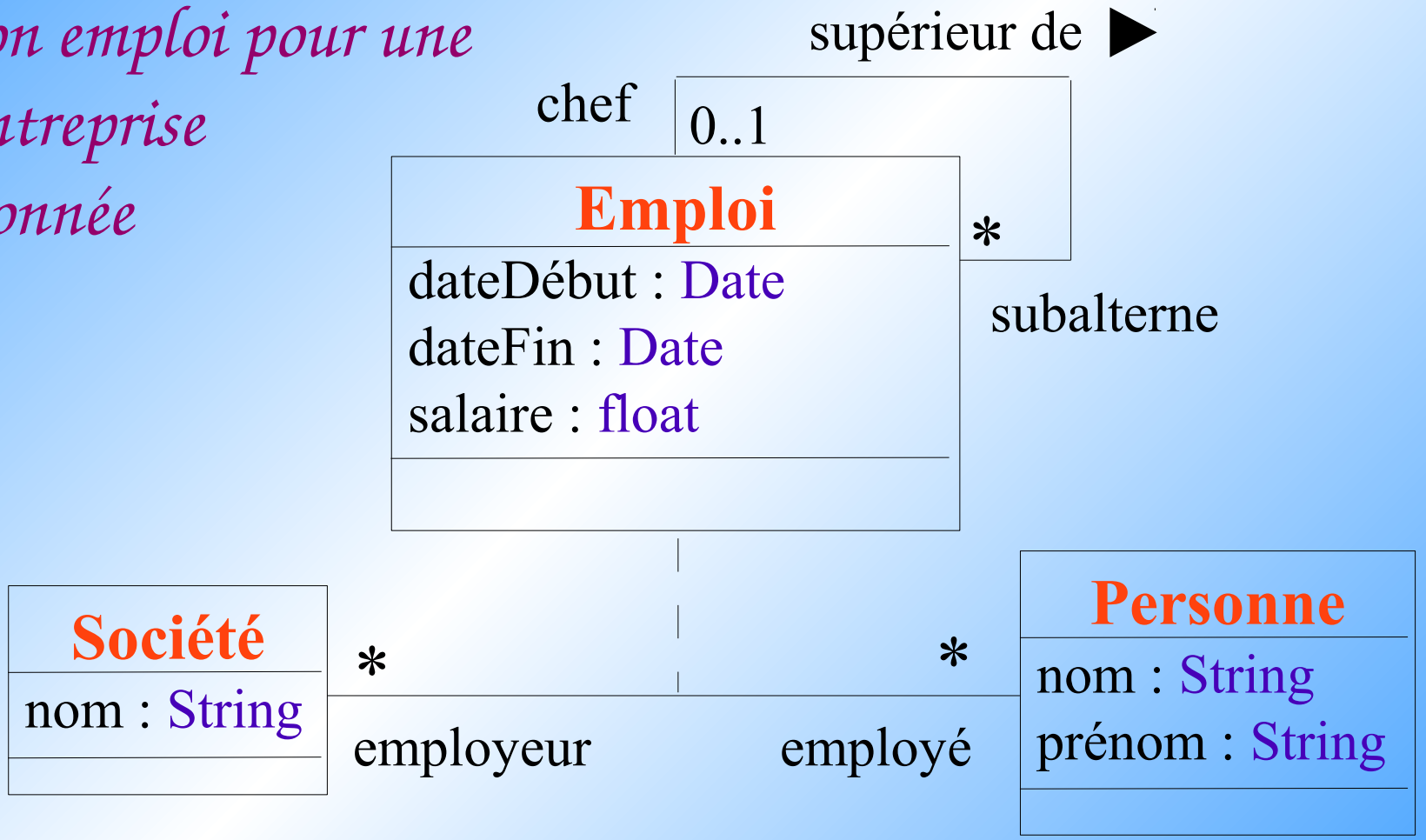

Emploi
dateDébut : Date
dateFin : Date [0..1]
salaire : float
Société
nom : String
*
employeur
*
employé
Personne
nom : String
prénom : String

EXERCICE

Rajouter le concept de supérieur hiérarchique

SOLUTION

Une personne est le supérieur d'une autre dans le cadre de son emploi pour une entreprise donnée

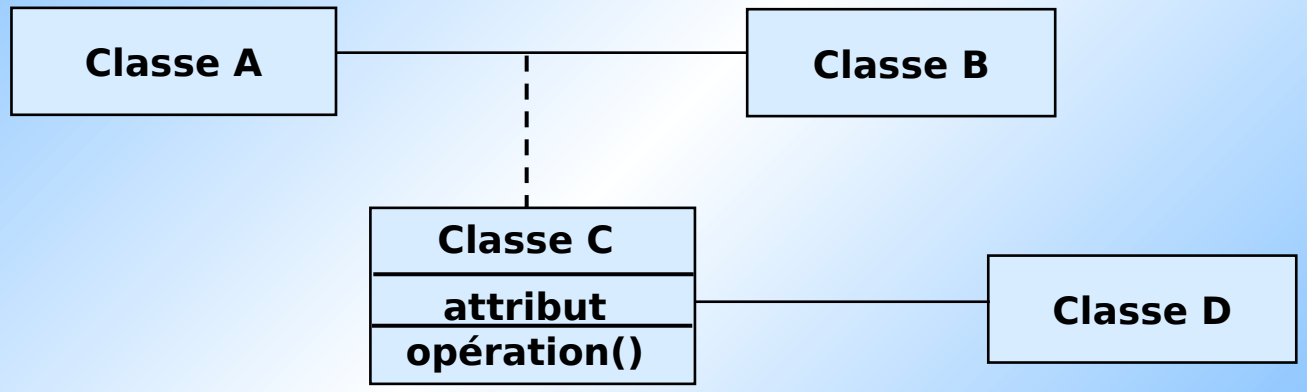


CLASSE D'ASSOCIATION

Classe d'association : association promue au rang de classe. Elle possède tout à la fois les caractéristiques d'une association et d'une classe et peut donc porter des attributs qui se valorisent pour chaque lien.

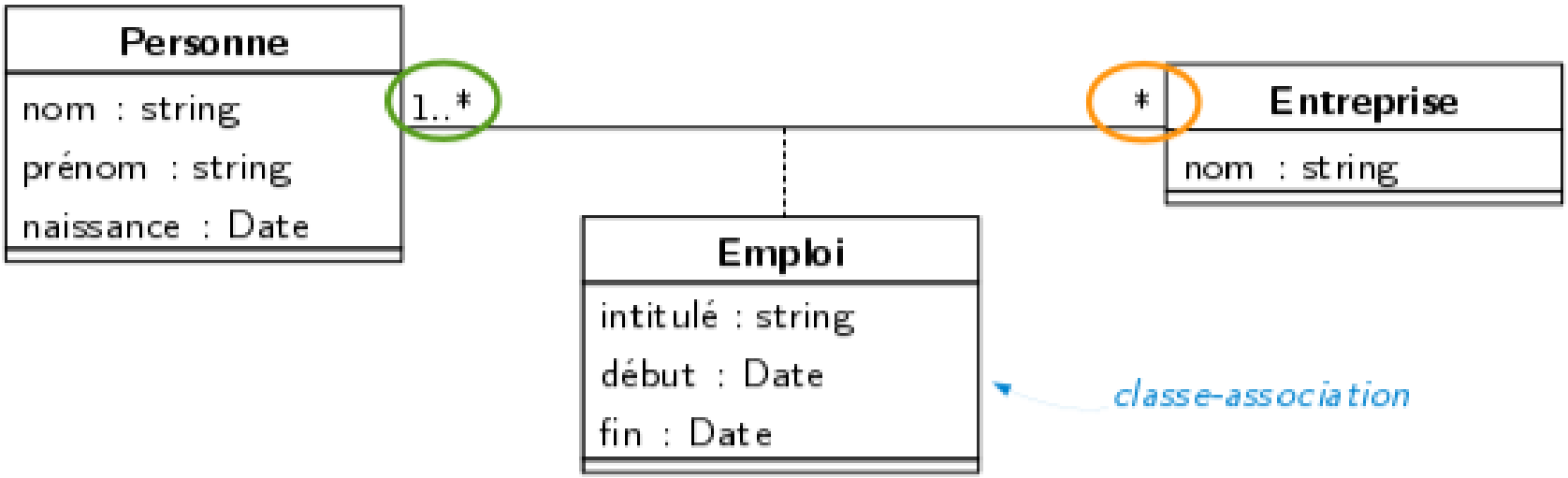
CLASSE D'ASSOCIATION

Une association peut être représentée par une classe (*classe C*) pour ajouter des attributs et des opérations dans l'association.

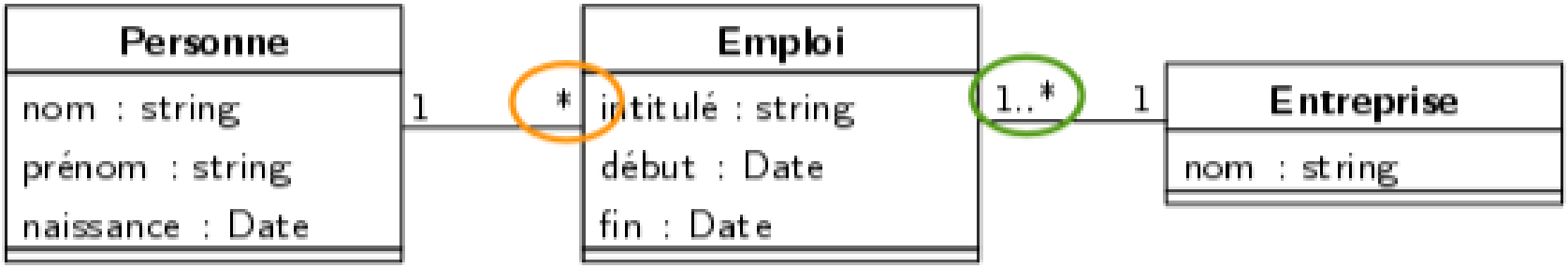


C'est une classe comme les autres et donc elle peut avoir d'autres relations (*avec la classe D*).

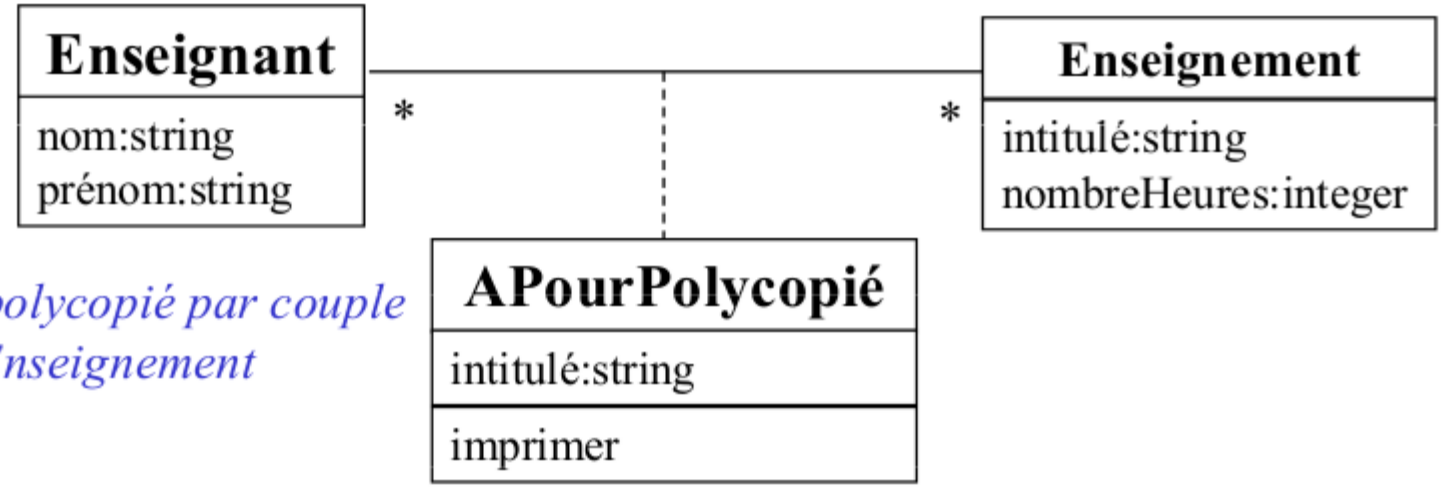
CLASSE D'ASSOCIATION



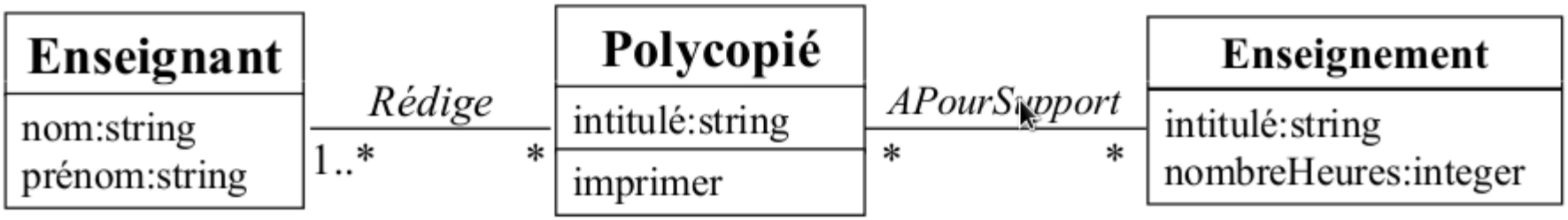
Équivalence en termes de classes et d'associations



Une classe d'association peut être remplacée par une classe intermédiaire qui sert de pivot.



Il y a un seul polycopié par couple Enseignant / Enseignement



Un nombre quelconque d'occurrences de Polycopié pour chaque Enseignant et chaque Enseignement

QUALIFICATIF

- Soit une association binaire qui fait correspondre un objet à un ensemble d'objets liés.

- Parfois, il peut être préférable de sélectionner un objet dans cet ensemble en fournissant une valeur qui permette de distinguer les objets dans l'ensemble.

- Il peut s'agir d'un attribut de la classe cible mais en général, c'est un attribut d'association dont la valeur est fournie par le créateur lorsqu'il ajoute un nouveau lien.

- Dans une association binaire, un tel attribut s'appelle un qualificatif.

QUALIFICATIF

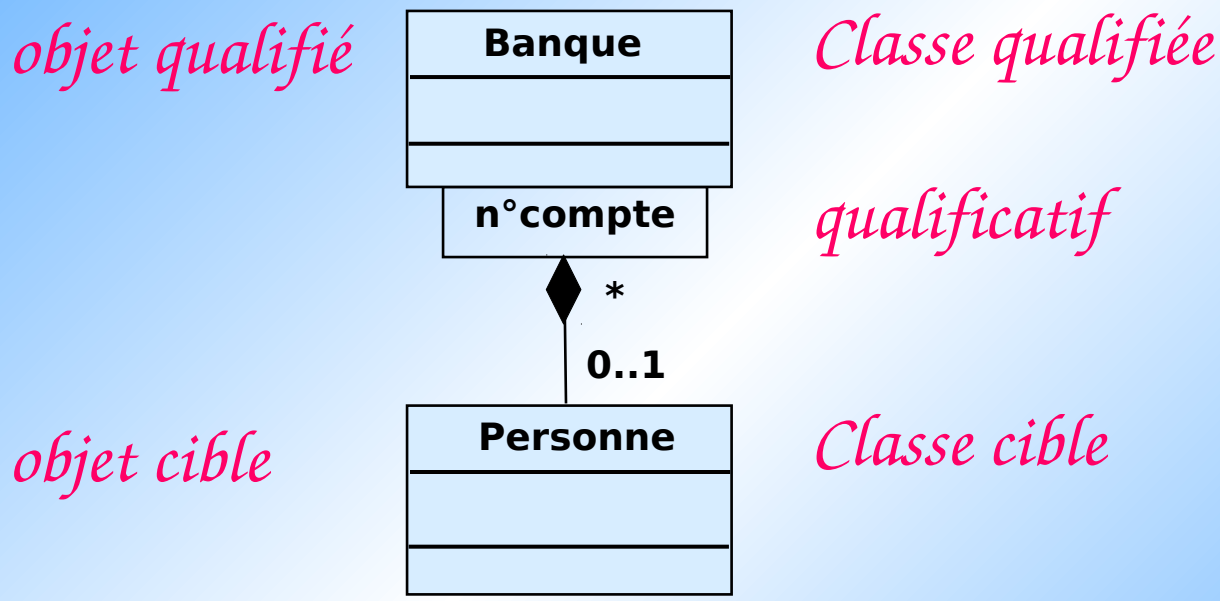
- Un objet associé à une valeur qualifiée détermine un objet lié unique ou (*moins souvent*) un sous-ensemble d'objets liés.

- Cette valeur qualifie l'association.

- Dans un contexte d'implémentation, on peut qualifier ces attributs de valeur d'index.

- Un qualificatif permet de sélectionner un ou des objet(s) dans l'ensemble des objets reliés à un objet (*appelé* ***objet qualifié***) par une association.

QUALIFICATIF



- L'objet sélectionné par la valeur du qualificatif est appelé objet cible.
- Un qualificatif agit toujours sur une association dont la multiplicité est *many* dans la direction cible.

QUALIFICATIF

- Dans le cas le plus simple, chaque valeur de qualificatif sélectionne un seul objet dans l'ensemble cible des objets liés.

- En d'autres termes, un objet qualifié et une valeur de qualificatif génèrent un objet cible lié unique.

- On peut modéliser un tableau comme une association qualifiée. Le tableau est l'objet qualifié, l'index du tableau est le qualificatif et l'élément du tableau est l'objet cible.

- Le qualificatif sert de sélecteur dans l'ensemble des objets reliés par l'association. Il scinde l'ensemble en sous-ensembles par valeur de qualificatif.

QUALIFICATIF

- Dans la plupart des cas, l'objectif d'un qualificatif est de sélectionner un objet unique dans l'ensemble des objets liés pour qu'une association qualifiée puisse se comporter comme une table de consultation.

- Un qualificatif possède un nom et un type mais pas de valeur initiale car les qualificatifs ne sont pas des objets indépendants et chaque valeur de qualificatif doit être explicite lorsqu'on ajoute un lien à l'association.

- On n'utilise pas de qualificatifs dans des associations n-aires.

QUALIFICATIF

- La classe qualifiée et le qualificatif forment à eux deux une valeur composite reliée à la classe cible.

- Les valeurs de multiplicité habituelles sont :
  - ☛ 0..1 : on peut sélectionner un objet unique mais toute valeur de qualificatif associée à l'objet qualifié ne sélectionne pas nécessairement un objet
  - ☛ 1 chaque valeur de qualificatif associée à l'objet qualifié sélectionne un objet cible unique. **Le domaine des valeurs de qualificatif doit être fini**.
  - ☛ * la valeur de qualificatif est un index qui partitionne les objets cibles en sous-ensembles.

QUALIFICATIF

- Dans la plupart des cas, l'objectif d'un qualificatif est de sélectionner un objet unique dans l'ensemble des objets liés pour qu'une association qualifiée puisse se comporter comme une table de consultation.

- Un qualificatif possède un nom et un type mais pas de valeur initiale car les qualificatifs ne sont pas des objets indépendants et chaque valeur de qualificatif doit être explicite lorsqu'on ajoute un lien à l'association.

- On n'utilise pas de qualificatifs dans des associations n-aires.

QUALIFICATIF

- Dans l'autre direction de l'association qualifiée (*cible vers objet qualifié*), la multiplicité indique le nombre de paires (*objet qualifié, qualificatif*) qu'on peut relier à l'objet cible, et non pas le nombre d'objets qualifiés.

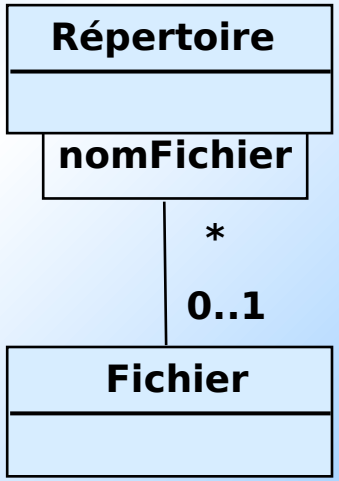
- Sur une association qualifiée, les multiplicités sont traitées comme si l'objet qualifié et le qualificatif étaient une seule entité, une clé composite.

- La valeur du qualificatif est une propriété de lien et non pas de l'objet cible.

QUALIFICATIF

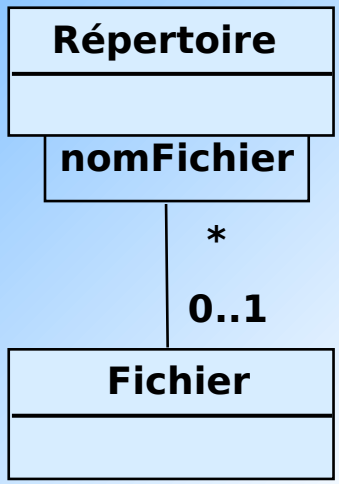
- Soit un système de fichiers UNIX dans lequel chaque répertoire est une liste d'entrées avec des noms. On peut utiliser les mêmes noms dans d'autres répertoires. Chaque entrée pointe vers un fichier qui peut être un fichier de données ou un autre répertoire. Plusieurs entrées peuvent pointer vers le même fichier (*fichier possédant plusieurs liens symboliques*).

Le répertoire qualifié par le nom du fichier génère un fichier.



Le nom du fichier ne fait pas partie du fichier. Il appartient à la relation entre un répertoire et un fichier. Un fichier ne possède pas de nom unique.

QUALIFICATIF



(répertoire, nomFichier) → 0 ou 1 fichier
répertoire → plusieurs fichiers
fichier → plusieurs (répertoire, nomFichier)
fichier → plusieurs répertoires
fichier → plusieurs nomFichiers

QUALIFICATIF

- Une association qualifiée est une table de consultation.

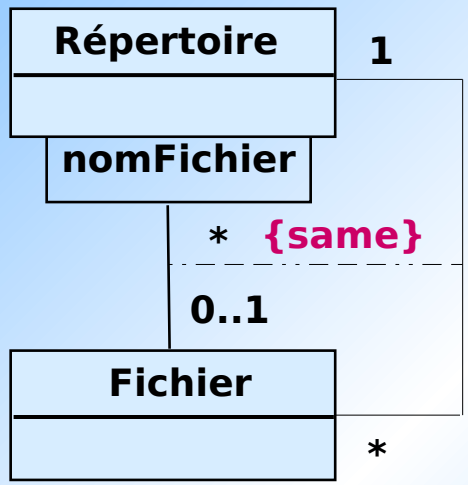
- On implémente les tables de consultation comme des tables de hachage, des arborescence-B et des listes triées.

- Une association qualifiée désigne une structure de données indexée optimisée pour la consultation fondée sur la valeur qualifiée.

- Un qualificatif d'attribut ne doit généralement pas être inclus comme attribut de la classe cible car sa présence dans l'association suffit.

QUALIFICATIF

- Supposons que chaque fichier doive se trouver dans un seul répertoire et qu'il est possible de nommer le même fichier de plusieurs manières.



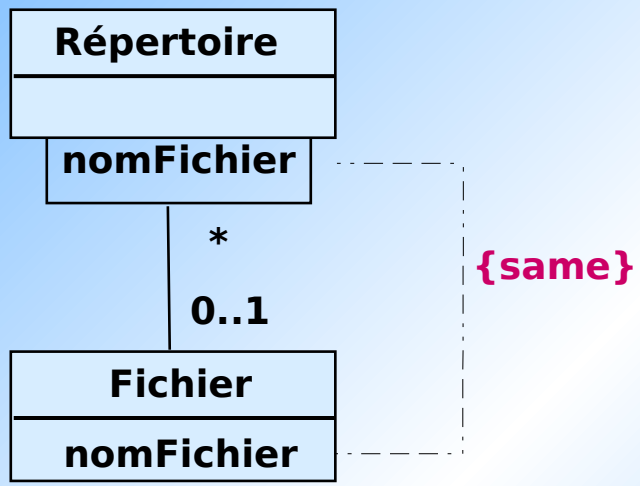
(répertoire, nomFichier) → 0 ou 1 fichier
répertoire → plusieurs fichiers
fichier → plusieurs (répertoire, nomFichier)
fichier → 1 répertoire
fichier → plusieurs nomFichiers

Fichier avec plusieurs noms dans un répertoire

- Seule l'association qualifiée sera implémentée.

QUALIFICATIF

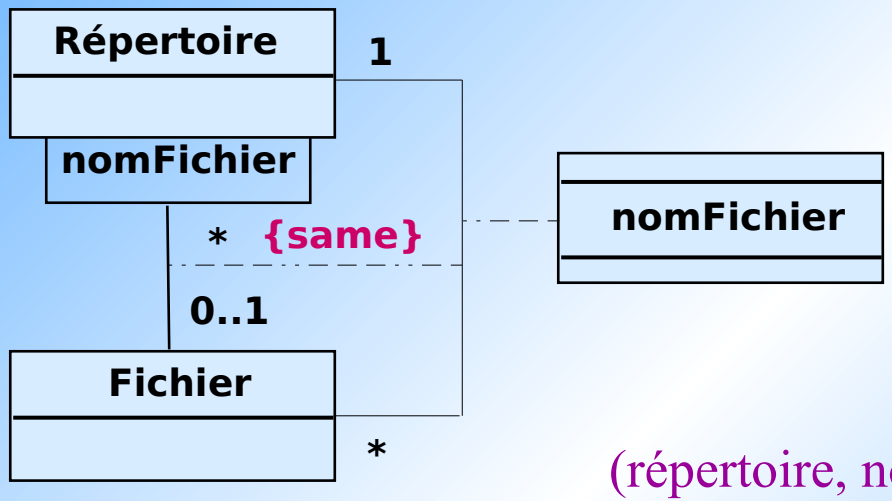
Fichier avec le même nom dans tous les répertoires



(répertoire, nomFichier) → 0 ou 1 fichier
répertoire → plusieurs fichiers
fichier → plusieurs (répertoire, nomFichier)
fichier → plusieurs répertoires
fichier → 1 nomFichier

QUALIFICATIF

Fichier avec au plus un nom dans un répertoire



(répertoire, nomFichier) → 0 ou 1 fichier
répertoire → plusieurs fichiers
fichier → plusieurs (répertoire, nomFichier)
fichier → plusieurs répertoires
fichier → plusieurs nomFichiers
(fichier, répertoire) → 0 ou 1 nomFichier

QUALIFICATIF

- Un **qualificatif** est un **attribut d'association** dont les **valeurs partitionnent la liste des objets** reliés par le biais d'une association.

- La connaissance d'un objet et d'une valeur de qualificatif permet de retrouver un ou plusieurs objets liés à l'autre bout de l'association concernée.

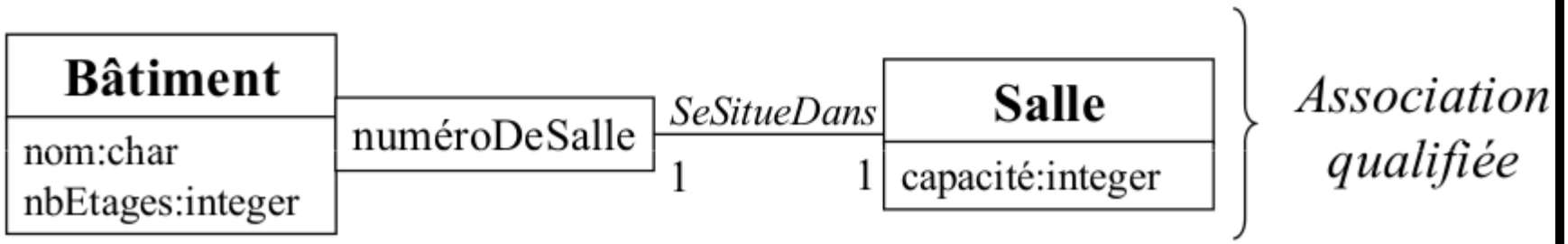
- Le qualificatif affine donc l'accès à une instance par navigation sur une association. Il ne s'applique qu'à une multiplicité supérieure à *1* puisqu'on ne peut partitionner une liste composée d'un seul élément.

Association qualifiée

- **Qualificateur** :
  - Attribut permettant de distinguer les objets situés à l'extrémité de multiplicité « plusieurs » d'une association
  - Attribut réduisant la multiplicité « plusieurs » à « un »
- **Association qualifiée** : association contenant un ou plusieurs attributs qualificateurs

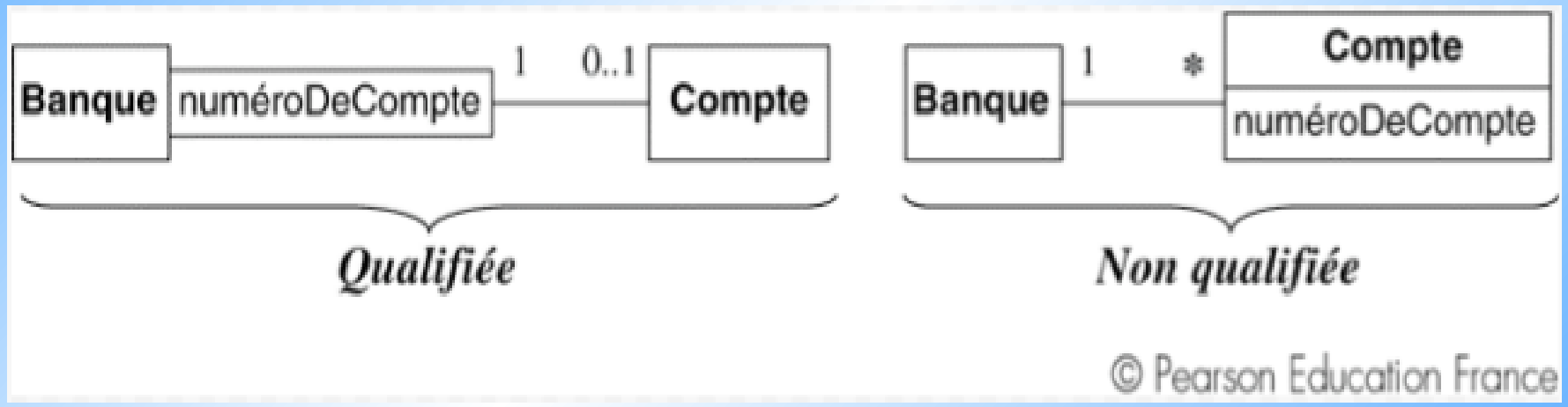
Un *numéro* de *Salle* permet d'identifier une salle unique dans un *Bâtiment* donné

Un *numéro* de *Salle* est relatif à un *Bâtiment*



Association qualifiée

- Une **association qualifiée** est l'équivalent UML d'un concept de programmation : tableau associatif ou map ou encore dictionnaire.



Méthode de Banque :
Compte getCompte(numéroDeCompte integer)

IMPLÉMENTATION

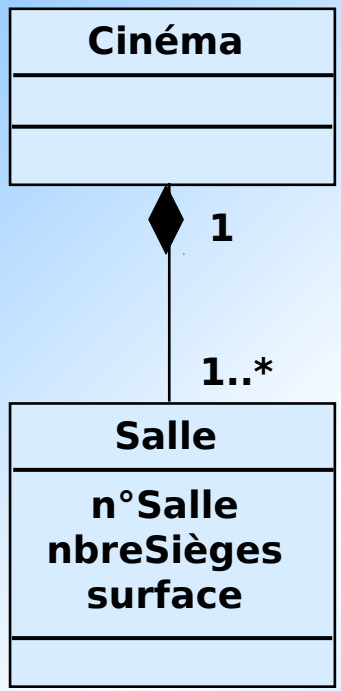

```
public class ClasseA {
  private Map<Q,ClasseB> roleB
      = new HashMap<Q,ClasseB>() ;
  ...
}
```

```
public class ClasseB {
  // ne connaît pas
  // la classe ClasseA
  ...
}
```

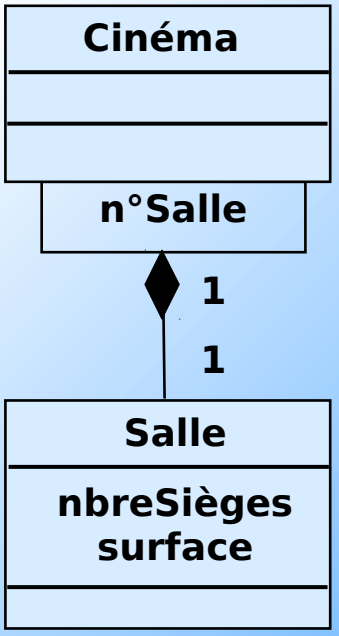
ASSOCIATION QUALIFIÉE

Exemple

Association non qualifiée

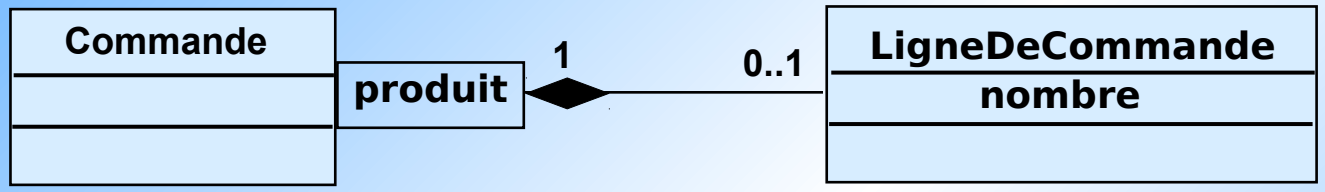


Association qualifiée



Cinéma + n°Salle : identifiant de Salle.

ASSOCIATION QUALIFIÉE



- Une commande ne peut pas contenir deux lignes de commande pour le même produit.

- Tout accès à une ligne donnée nécessite un produit comme argument.

- Une multiplicité de *1* indiquerait qu'il doit y avoir une ligne pour chaque produit, et une multiplicité de * que l'on peut avoir plusieurs lignes de commande par produit, mais que l'accès aux lignes est toujours indexé par produit.

ASSOCIATION QUALIFIÉE

- On utilise des qualificatifs pour montrer des contraintes du type *"une seule ligne de commande par produit dans une commande"*.

```
class Commande {
    public LigneDeCommande getLigneDeCommande
                                        (Produit produit);
    public void addLigneDeCommande
                                (int nombre, Produit pourProduit);
}
```

ASSOCIATION QUALIFIÉE

- Un objet + une valeur de qualificatif identifient un objet unique au travers de l'association. Ensemble ils forment une clé composite.

- Un qualificatif peut lui-même être composé (*exemple d'une ligne et d'une colonne d'une case d'un échiquier*).

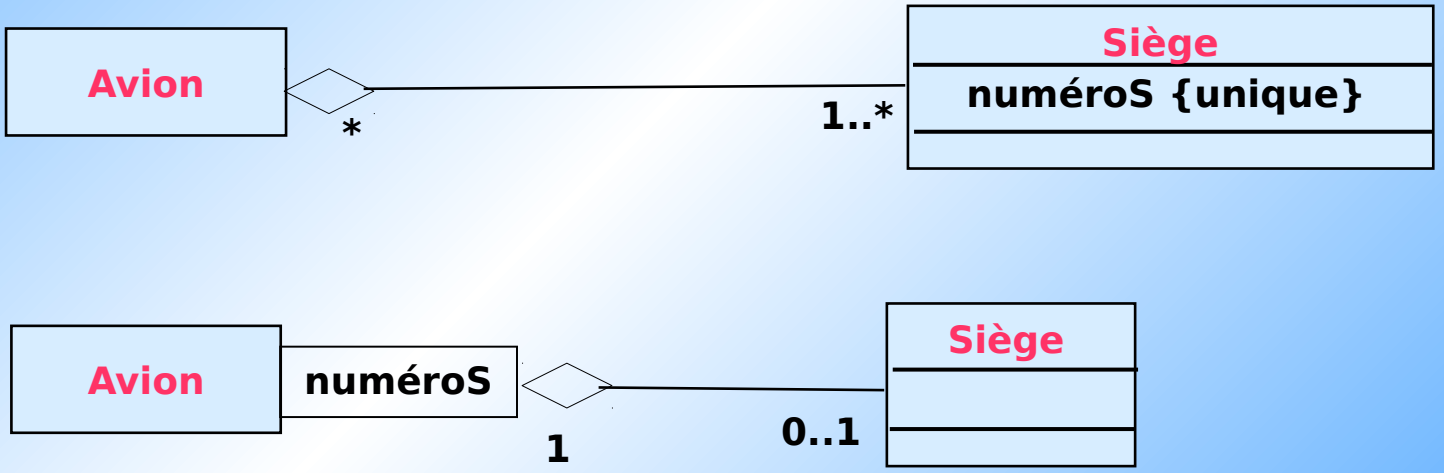
ASSOCIATION QUALIFIÉE

- Mise en place d'un mécanisme d'indexation.

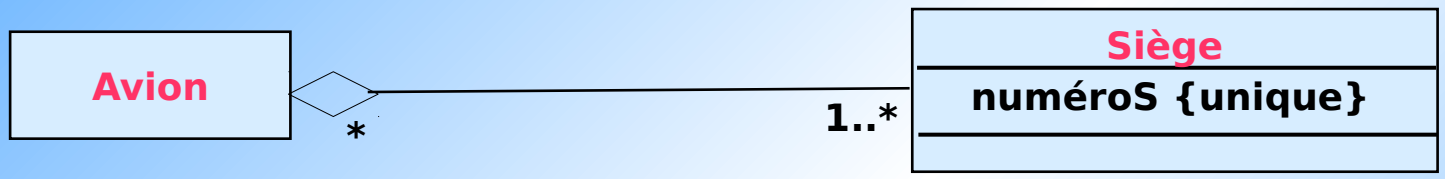
- Un qualificatif est typiquement un attribut de l'élément cible, bien que ceci ne soit pas une nécessité.

EXERCICE

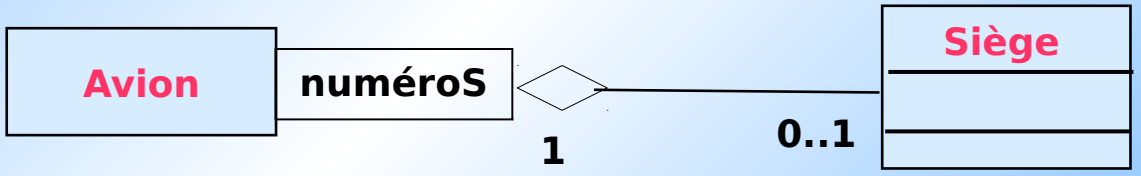
Quelle est la différence entre ces deux modèles ?



SOLUTION



- Un avion contient plusieurs sièges qui ont chacun un numéro différent. L'identifiant de Siège est : numéroS.

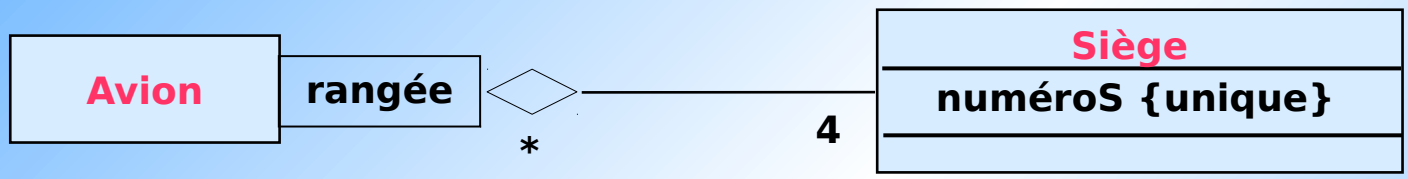


- Chaque siège a un numéro qui est unique pour chaque avion. L'identifiant de Siège est : identifiantAvion + numéroS.

EXERCICE

Quelle est la sémantique du modèle suivant ?

SOLUTION

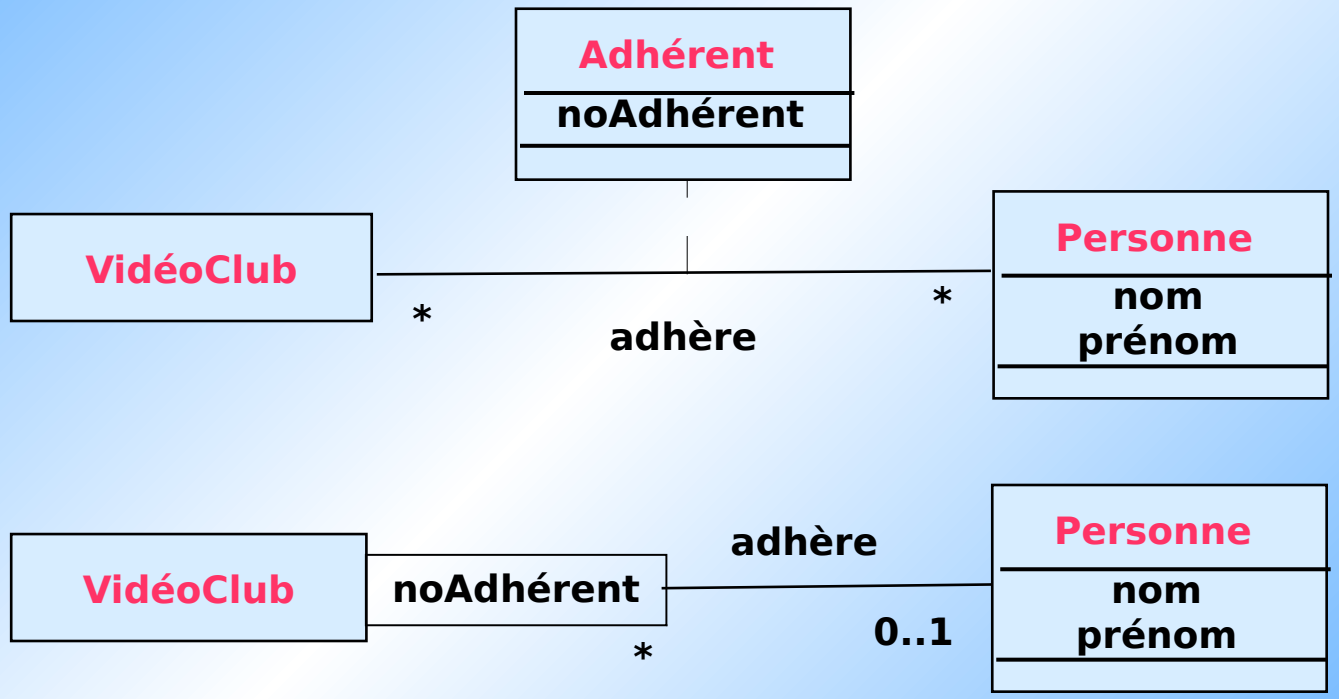


- Dans chaque avion, pour une rangée donnée, il y a quatre sièges.

- Le qualificateur permet de partitionner l'ensemble des instances de *Siège* suivant un critère indiqué par le qualificateur.

EXERCICE

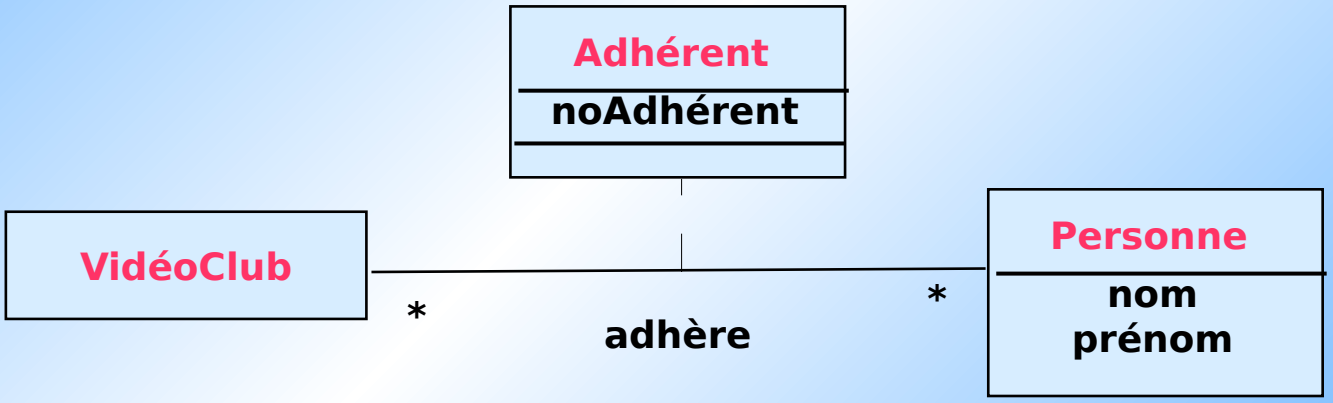
Quelle est la différence entre ces deux modèles ?



SOLUTION

Pour une personne donnée dans un vidéo club, il n'y a qu'un seul numéro d'adhérent.

nom, prénom, n°VidéoClub → n°Adhérent

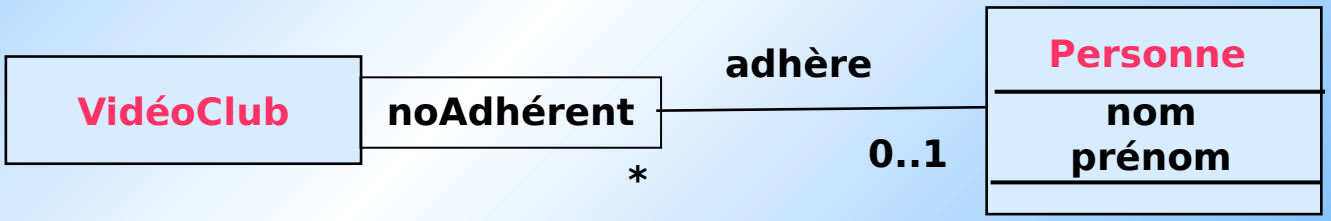


modélisation maladroite (*classe Adhérent ?*)

SOLUTION

Pour un vidéo club donné et un numéro d'adhérent correspond au plus une personne.

n°VidéoClub, n°Adhérent → nom, prénom

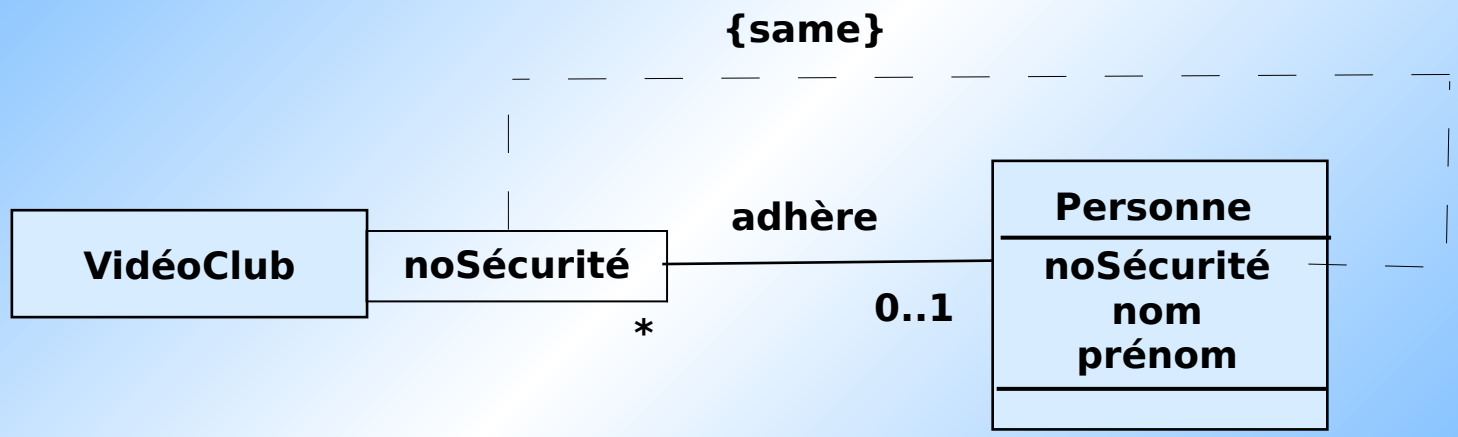


EXERCICE

La vidéothèque souhaite indexer ses adhérents par le numéro de sécurité sociale

SOLUTION

La vidéothèque souhaite indexer ses adhérents par le numéro de sécurité sociale

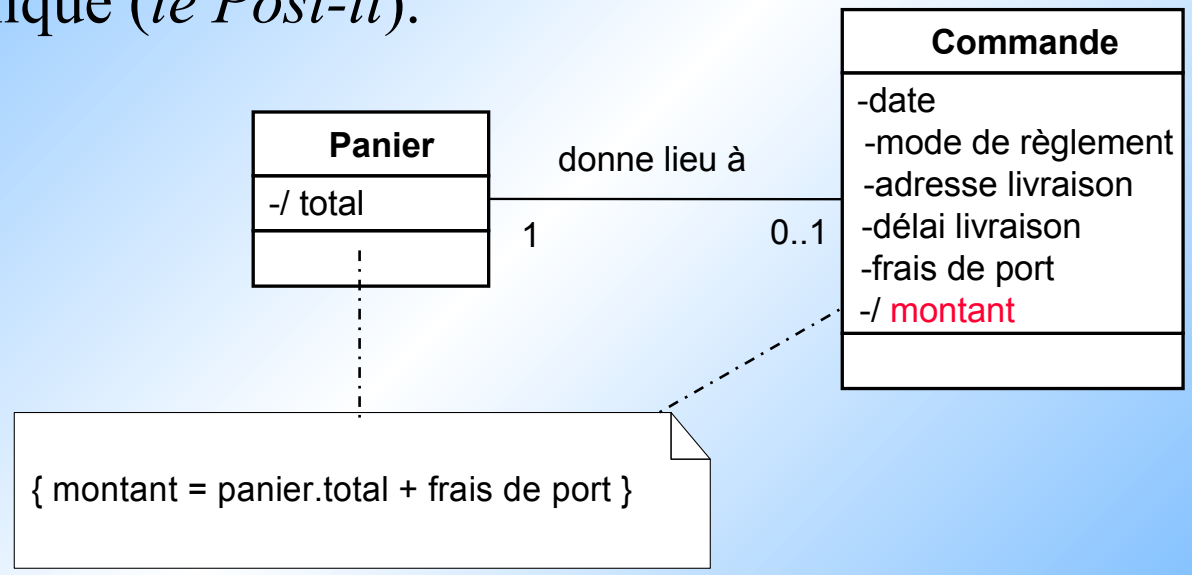


noSécurité est une propriété de Personne et non une propriété de l'association adhère.

CONTRAINTE

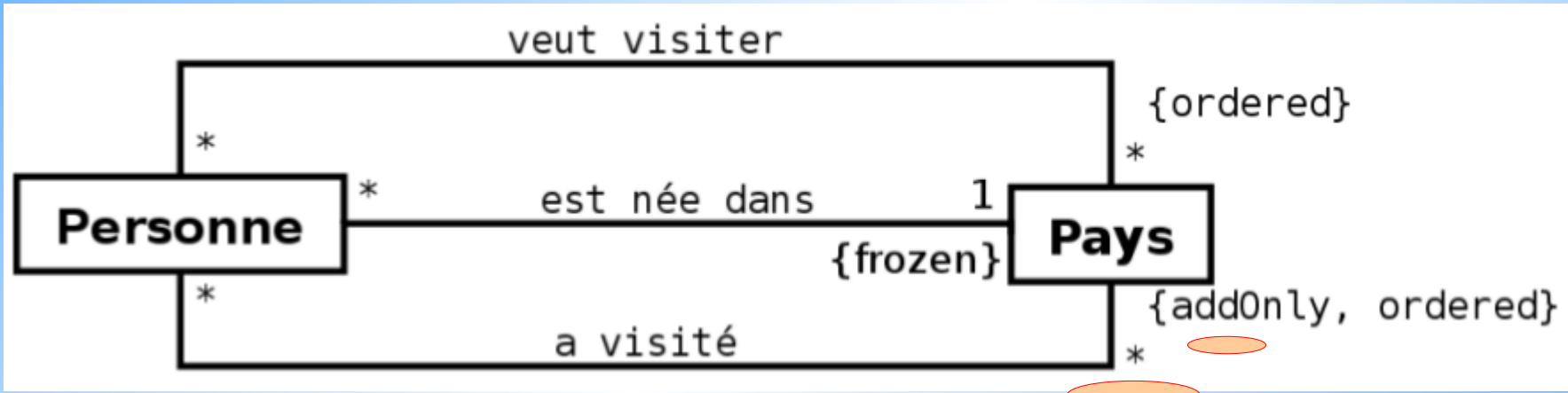
Une **contrainte** est une condition entre éléments du modèle, qui doit être vérifiée par les éléments concernés.

Elle est définie entre accolades { }, et est insérée dans une note graphique (*le Post-it*).



CONTRAINTE

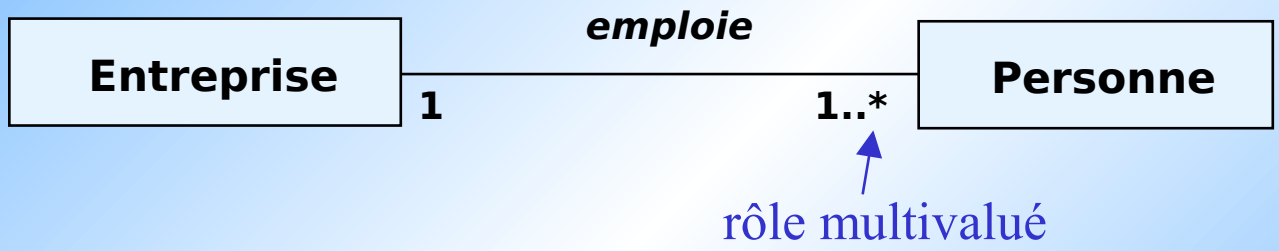
Exemple



On peut ajouter de nouveaux liens
mais pas en supprimer

Rôle multivalué

- Un **rôle multivalué** est un rôle dont la multiplicité a une valeur maximale supérieure strictement à *1* (* *par exemple*).



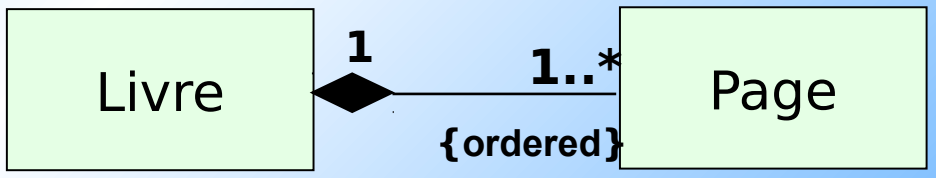
- La convention usuelle est de considérer les rôles multivalués comme des ensembles. Les objets ne sont pas ordonnés, et aucun objet n'apparaît plus d'une fois dans l'ensemble.

- On peut modifier cette convention en attachant une contrainte au rôle.

CONTRAINTE D'ORDRE

- La contrainte {ordered} peut être placée sur le rôle pour spécifier qu'une relation d'ordre décrit les objets placés dans la collection.

- On ne spécifie pas comment les éléments sont ordonnés (*numéro, ordre alphabétique, etc*) car c'est un choix de conception, mais seulement que l'ordre doit être maintenu durant l'ajout et/ou la suppression des objets.



CONTRAINTE D'IMMUABILITÉ

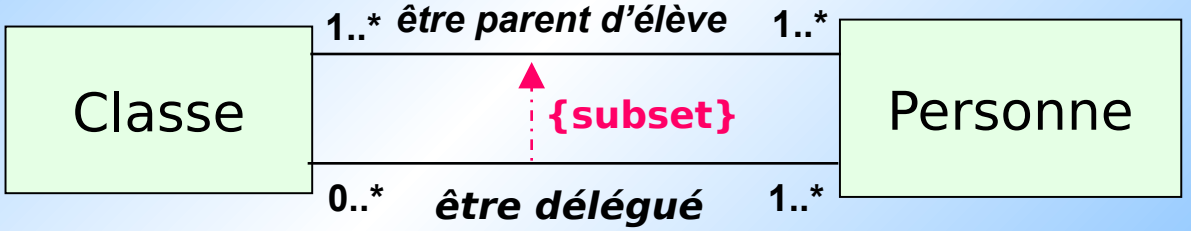
- La contrainte {frozen} (*gelé*) peut être placée sur un attribut, une extrémité d'association ou une classe.

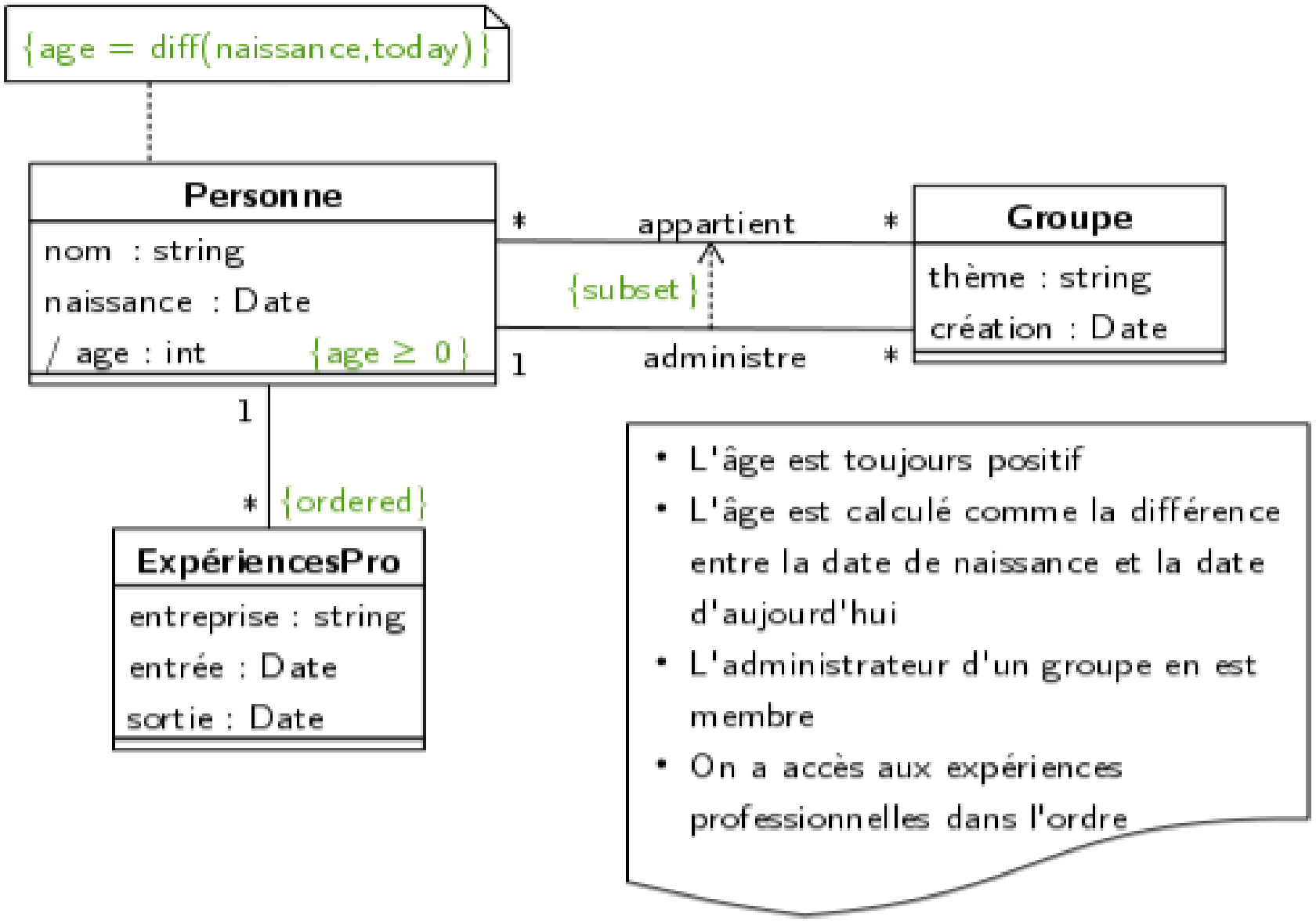
- Sur un attribut ou une extrémité d'association, la contrainte d'immuabilité indique que la valeur de cet attribut ou de cette association est immuable pendant toute la durée de vie de l'objet. Elle doit être définie à la création de l'objet et ne peut jamais être modifiée.

- Appliquée à une classe, la contrainte d'immuabilité indique que toutes les terminaisons d'association et tous les attributs de la classe sont gelés.

CONTRAINTE DE SOUS-ENSEMBLE

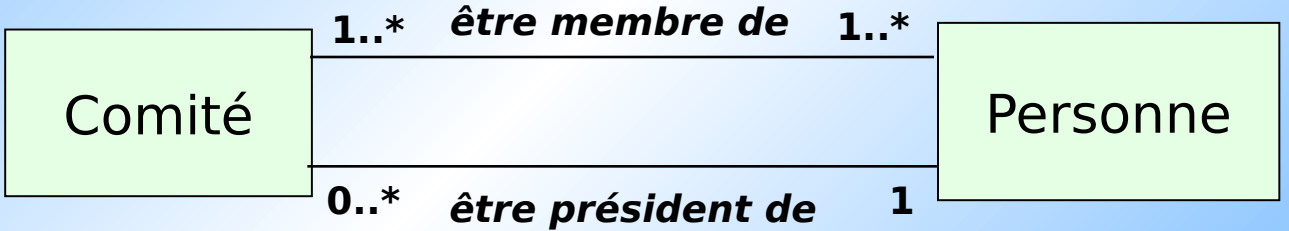
- La contrainte de *sous-ensemble* {subset} indique qu'une collection est incluse dans une autre collection.




{age = diff(naissance,today)}
Personne
nom : string
naissance : Date
/ age : int
{age ≥ 0}
*
appartient
*
Groupe
thème : string
création : Date
{subset}
1
administre
*
1
*
{ordered}
ExpériencesPro
entreprise : string
entrée : Date
sortie : Date
• L'âge est toujours positif
• L'âge est calculé comme la différence entre la date de naissance et la date d'aujourd'hui
• L'administrateur d'un groupe en est membre
• On a accès aux expériences professionnelles dans l'ordre

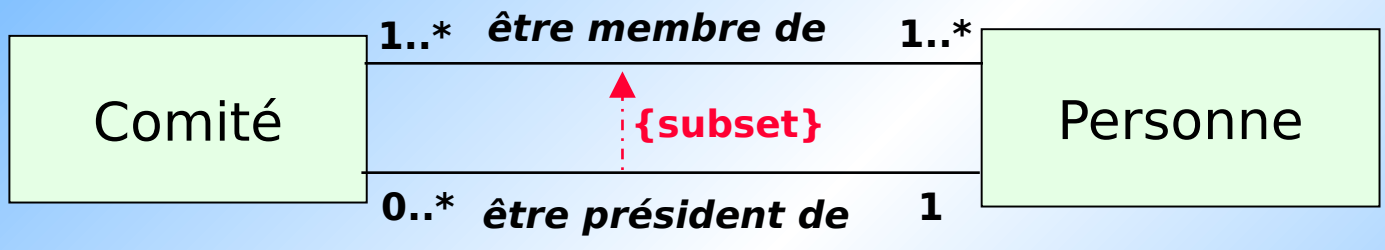
EXERCICE

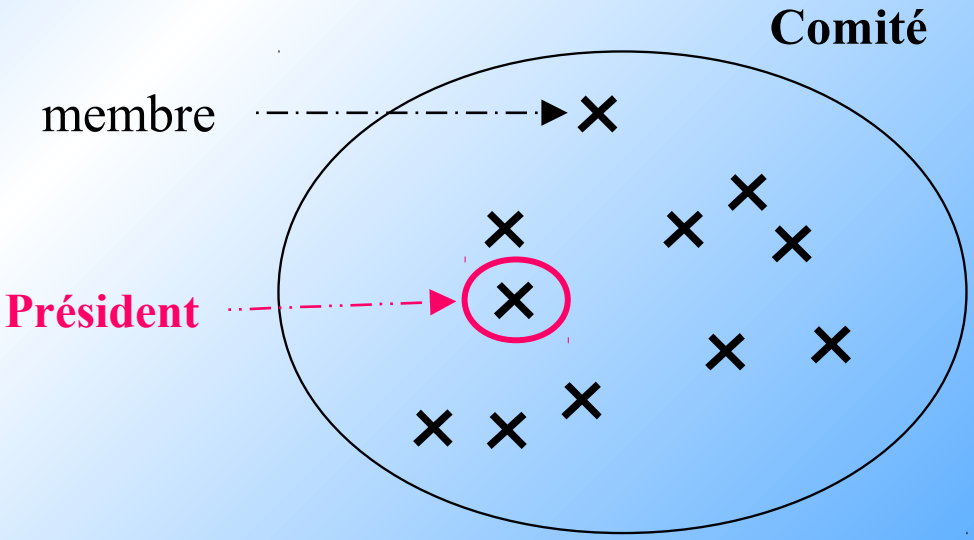
Compléter le modèle



Un président doit faire partie du comité

SOLUTION

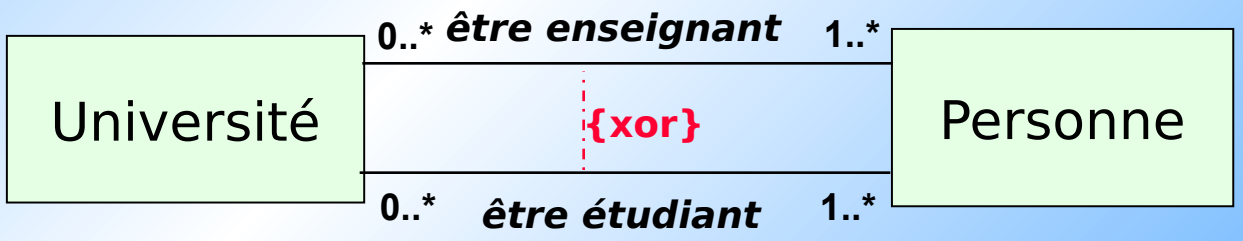

Comité
1..*
être membre de
1..*
{subset}
Personne
0..*
être président de
1


Comité
membre
Président

CONTRAINTE DU OU EXCLUSIF

- La contrainte du *ou exclusif* {xor} précise que, pour un objet donné, une seule association parmi un groupe d'associations est valide.

- Cette contrainte évite l'introduction de sous-classes artificielles pour matérialiser l'exclusivité.



AGRÉGATION

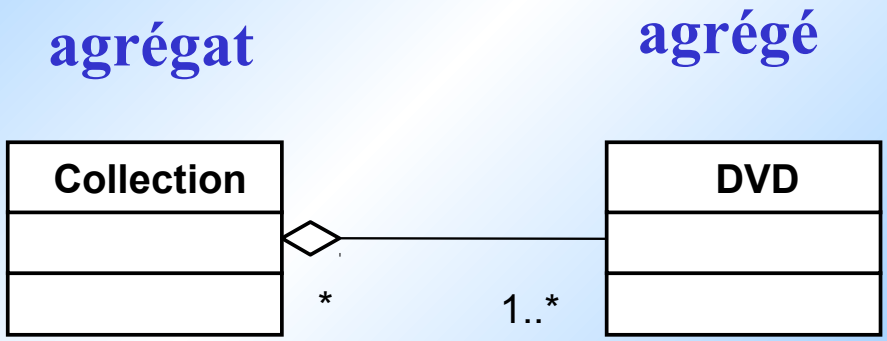
- Si l'une des classes joue le rôle d'**ensemble composé d'instances** de l'autre classe, on utilise l'**agrégation**.

- Une agrégation n'est plus sémantiquement symétrique puisqu'elle privilégie l'une des deux classes en l'élevant au rang de conteneur.

- Les agrégations n'ont pas besoin d'être nommées : implicitement, elles signifient *"contient"*, *"est composé de"*.

AGRÉGATION

- Forme particulière d'association entre un tout et ses parties, dans laquelle le tout est composé de parties.
- Association non symétrique, une extrémité joue un rôle prédominant par rapport à l'autre.



AGRÉGATION

- L'**agrégation** garde cependant les **propriétés d'une association** et n'influe :

  - ✓ ni sur l'expression des multiplicités,
  - ✓ ni sur la navigabilité,
  - ✓ ni sur le cycle de vie des instances reliées.

- Par conséquent, il est possible de partager l'agrégation : une partie peut appartenir simultanément à plusieurs agrégats.

COMPOSITION

La **composition** est une **agrégation particulière** où l'objet ne peut appartenir qu'à un seul composite **et** le cycle de vie est très imbriqué.

(*la fermeture de l'entreprise entraîne la suppression des services*).

COMPOSITION

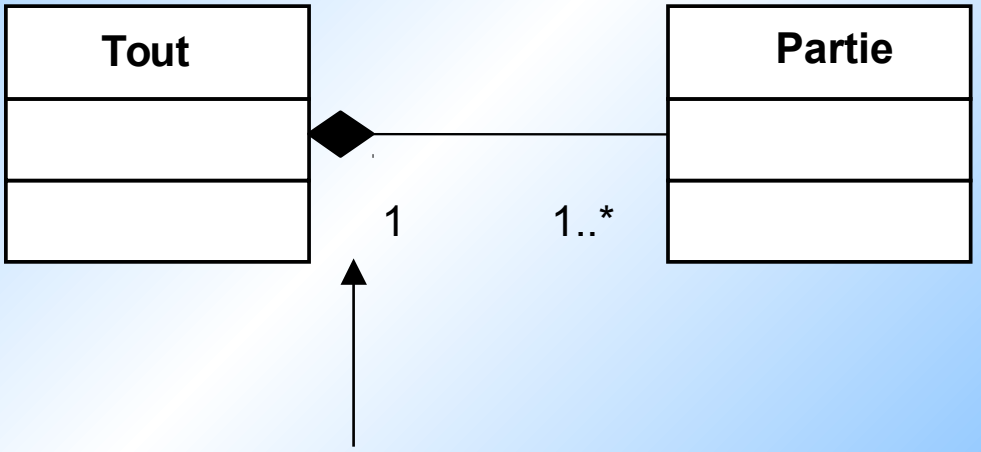
Avec une composition, on introduit les deux caractéristiques suivantes :

1 - la composition n'est pas partageable : un objet ne peut appartenir qu'à un seul composite à la fois.

2 - le cycle de vie des parties est forcément lié à celui du composite : la destruction du composite entraîne en particulier la destruction de ses parties.

COMPOSITION

La composition implique une contrainte sur la valeur de la multiplicité du côté du composite : elle ne peut prendre que les valeurs 0 ou 1.



Cardinalités : 0..1 ou 1 seulement

0 en cardinalité minimale : attribut non renseigné

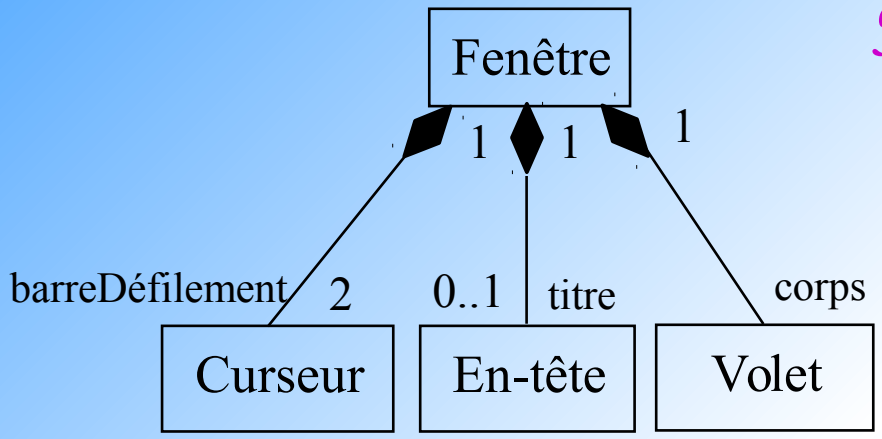
COMPOSITION

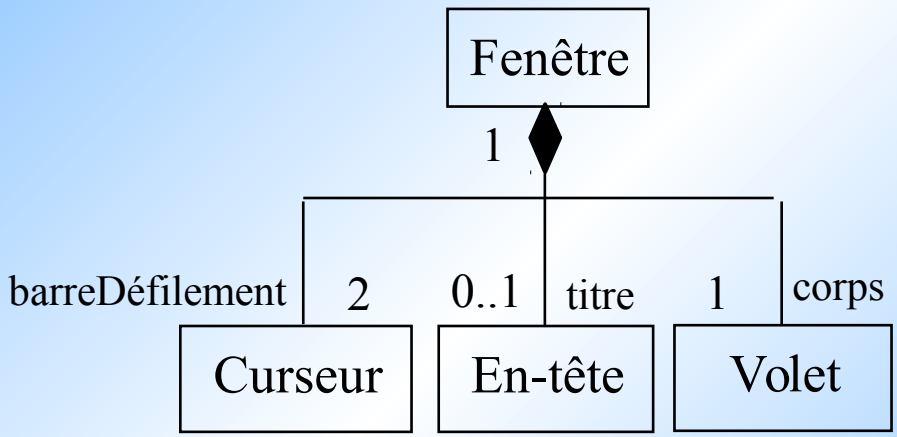
- La composition et les attributs sont sémantiquement équivalents, leurs représentations graphiques sont interchangeables.

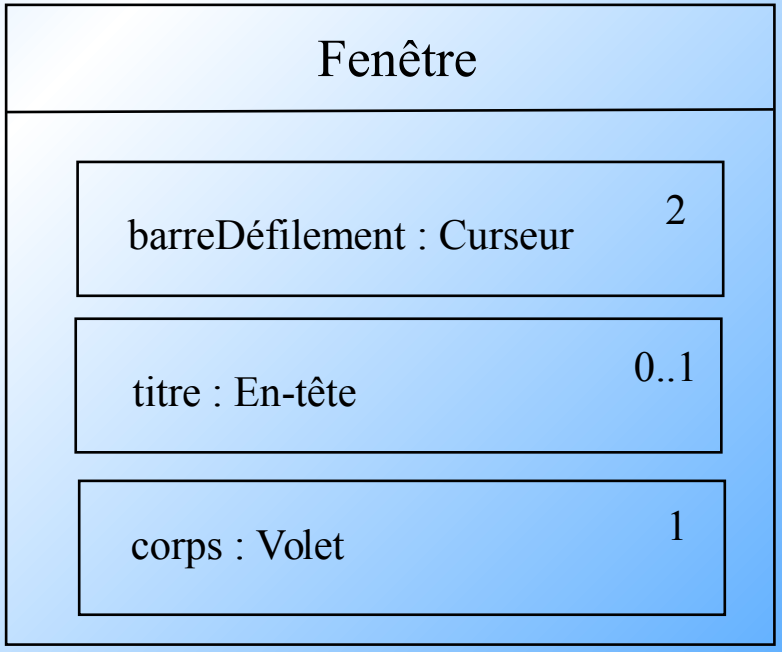
- La notation par composition s'emploie dans un diagramme de classes lorsqu'un attribut participe à d'autres relations dans le modèle.

COMPOSITION

Notations







HÉRITAGE

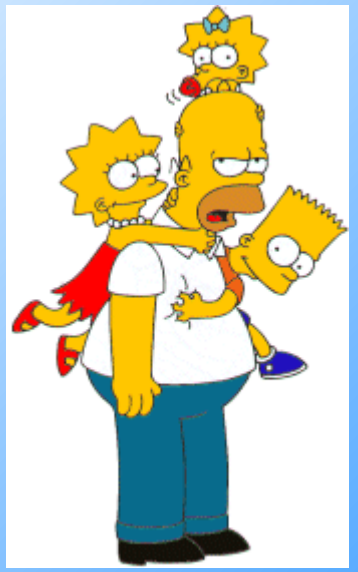
- C'est le procédé de partage ou de factorisation de l'information dans les technologies à objets

ou

- Mécanisme de dérivation d'une nouvelle classe ou d'un nouveau objet à partir d'une classe existante.

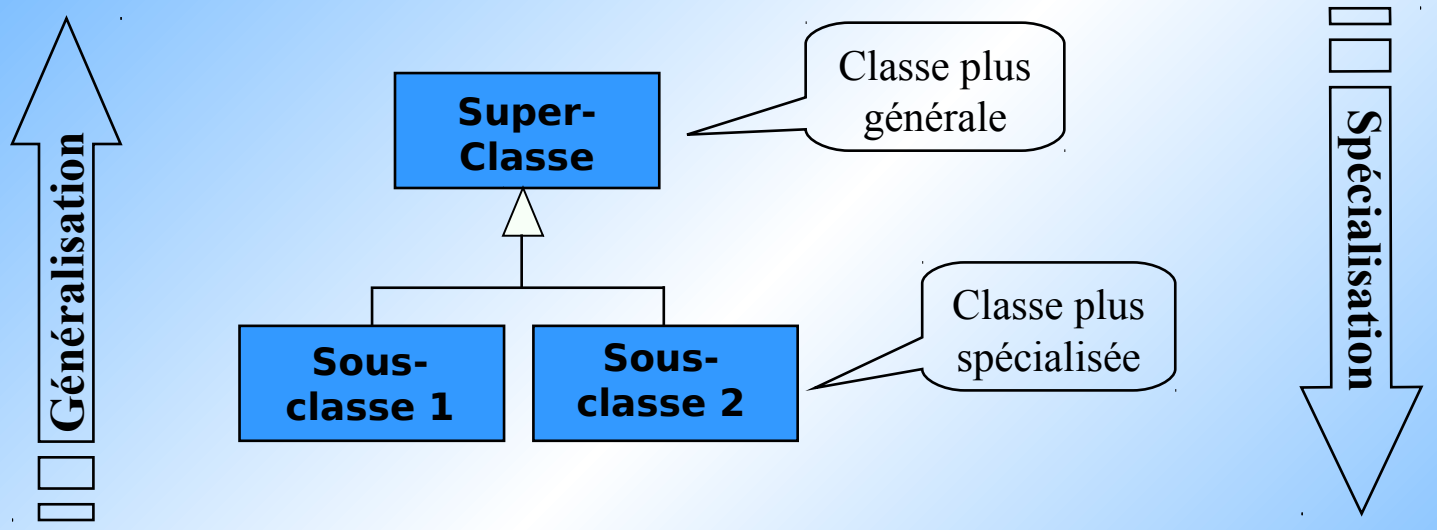
- L'idée provient de l'existence d'une hiérarchie de classes.

Comportement
Apparence
Attitude



HÉRITAGE

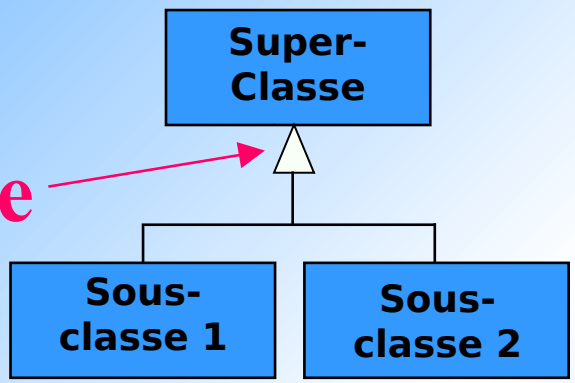
- Hiérarchie de classes : super-classe et sous-classe(*s*).



- **Généralisation** : factorisation des attributs et des opérations communs à plusieurs classes dans une classe plus générale.

- **Spécialisation** : raffinement d'une classe par création d'une classe plus spécialisée.

HÉRITAGE



Pas de flèche mettre un triangle

super-classe
classe de base
classe mère

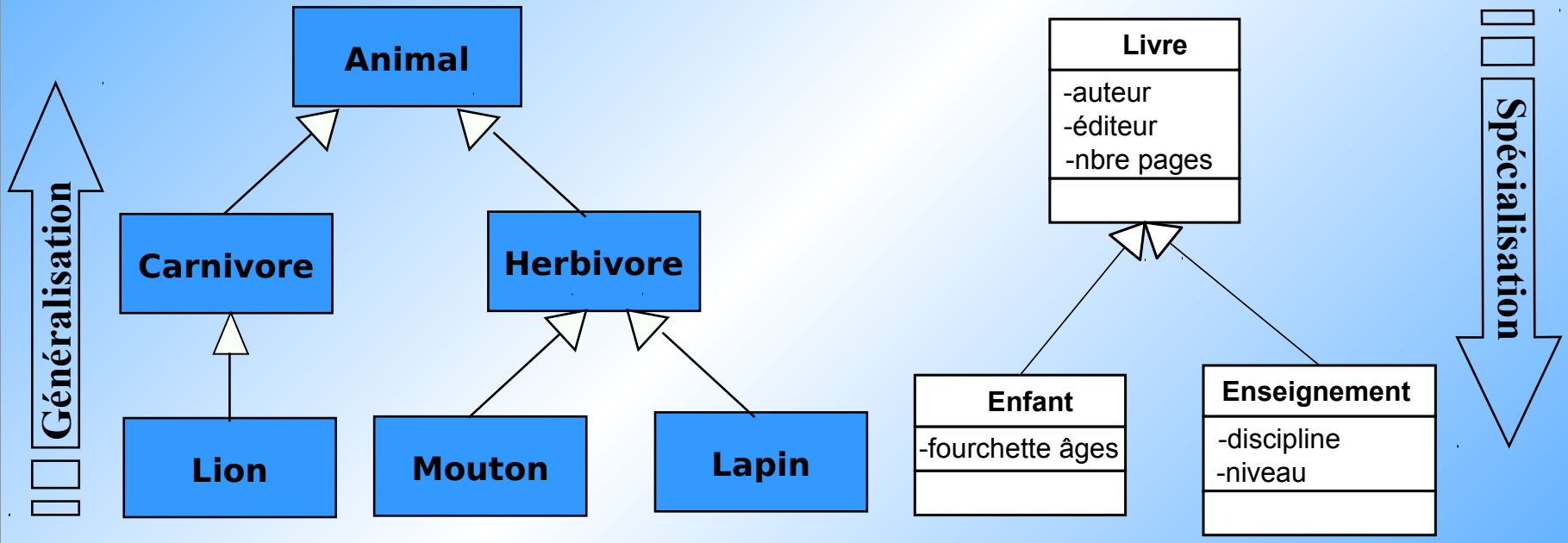
sous-classe
classe dérivée
classe fille

- La classe dérivée **est une** version spécialisée de sa classe de base.

- Relation "**est-un**" pour traduire le principe de généralisation / spécialisation.

HÉRITAGE

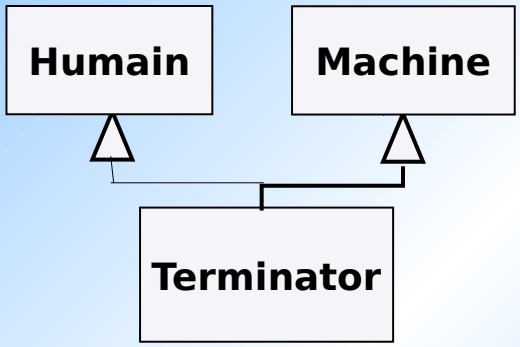
Exemples



Chaque instance d'une sous-classe est aussi une instance de toutes les super-classes.

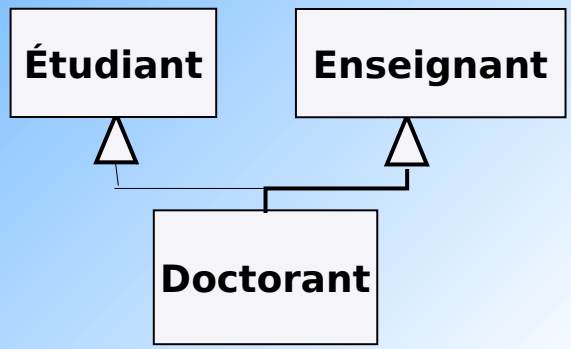
HÉRITAGE

Héritage multiple



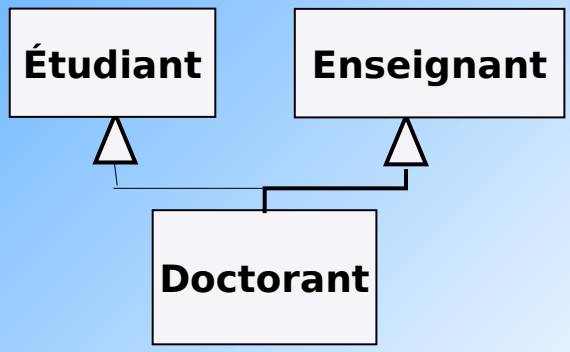
HÉRITAGE

Héritage multiple



HÉRITAGE

Héritage multiple



Résilience : phénomène psychologique qui consiste, pour un individu affecté par un traumatisme, à prendre acte de l'événement traumatique pour ne plus vivre dans la dépression et se reconstruire.

HÉRITAGE

- Un objet hérite les attributs et les méthodes de la classe dont il est issu

  ou

  la classe spécialisée hérite des propriétés qu'elle n'a pas substituées.

- Une sous-classe peut ajouter des attributs et/ou des méthodes.

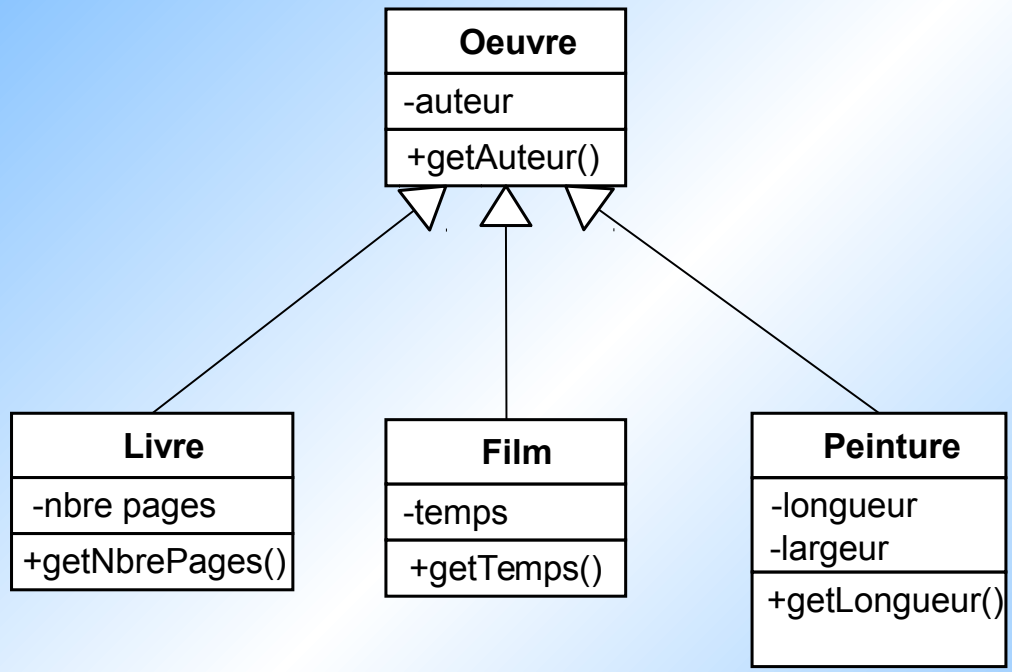
- Une sous-classe peut redéfinir une méthode par surcharge.

Comportement
Apparence
Attitude

HÉRITAGE

Exemple

- Une œuvre est la composition d'un artiste. Parmi les œuvres, on peut citer les livres qui ont un certain nombre de pages, les films qui durent un certain temps, les peintures d'une certaine taille …
- Lorsqu'on dit tout livre est une œuvre, et qu'une œuvre a un auteur, on veut se passer de redire qu'un livre a un auteur.



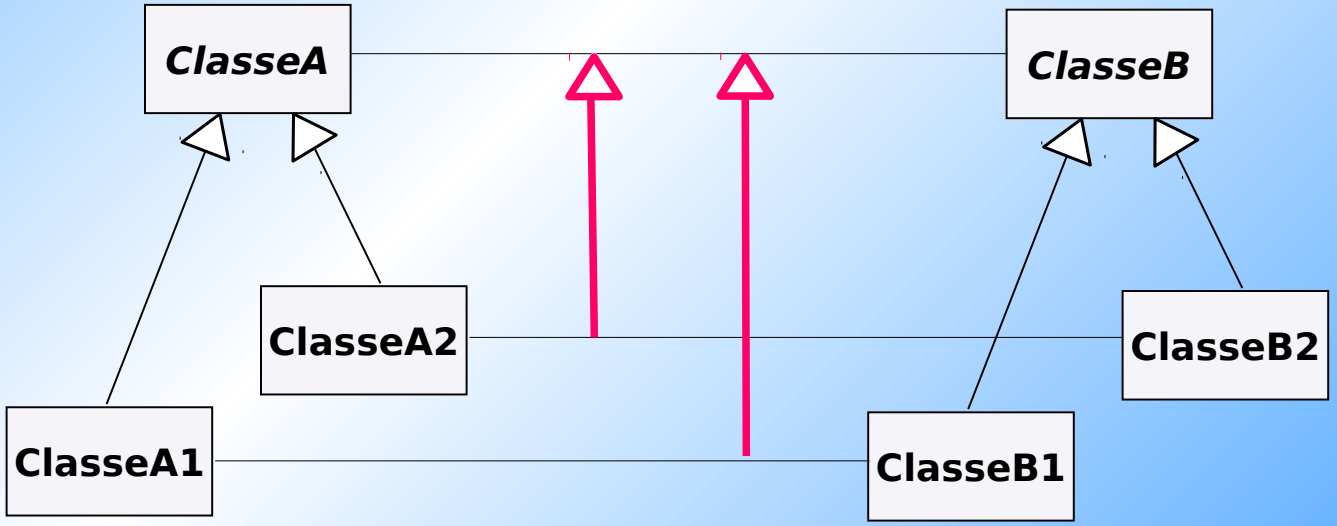
- Le but est de ne pas écrire deux fois la même chose : réutiliser le code existant.
- Éviter de faire des bêtises identiques à plusieurs endroits.

SPÉCIALISATION DE L'ASSOCIATION

- Généralisation peu courante.

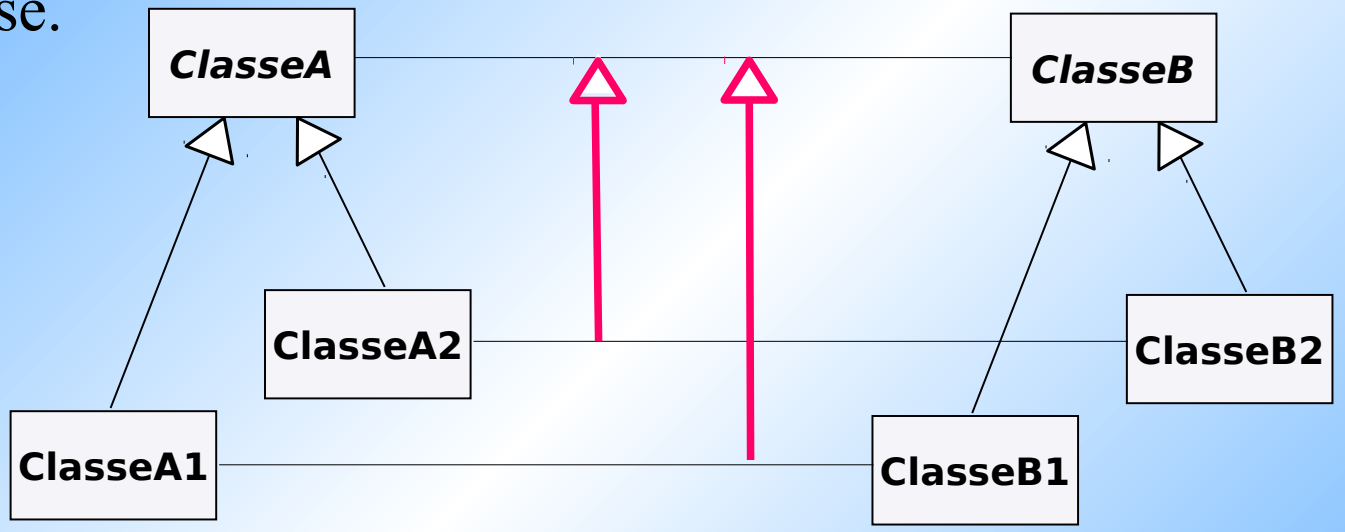
- L'élément enfant doit apporter des contraintes supplémentaires au parent et constituer un sous-ensemble de l'extension du parent.

- Une association enfant est plus contrainte que son parent.



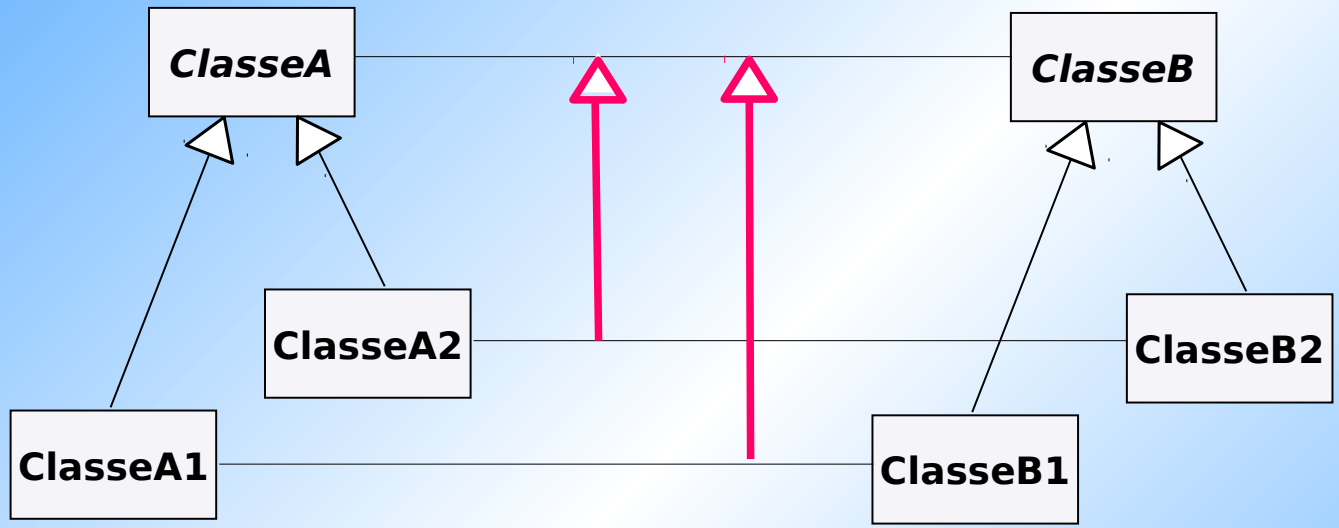
SPÉCIALISATION DE L'ASSOCIATION

- Définir des sous-ensembles de l'étendue signifie que chaque lien de l'association enfant est un lien de l'association parent, mais pas l'inverse.



- Tout lien connectant ClasseA1 et ClasseB1 connectera également *ClasseA* et *ClasseB*, mais tous les liens connectant *ClasseA* et *ClasseB* ne se connecteront pas à ClasseA1 et ClasseB1.

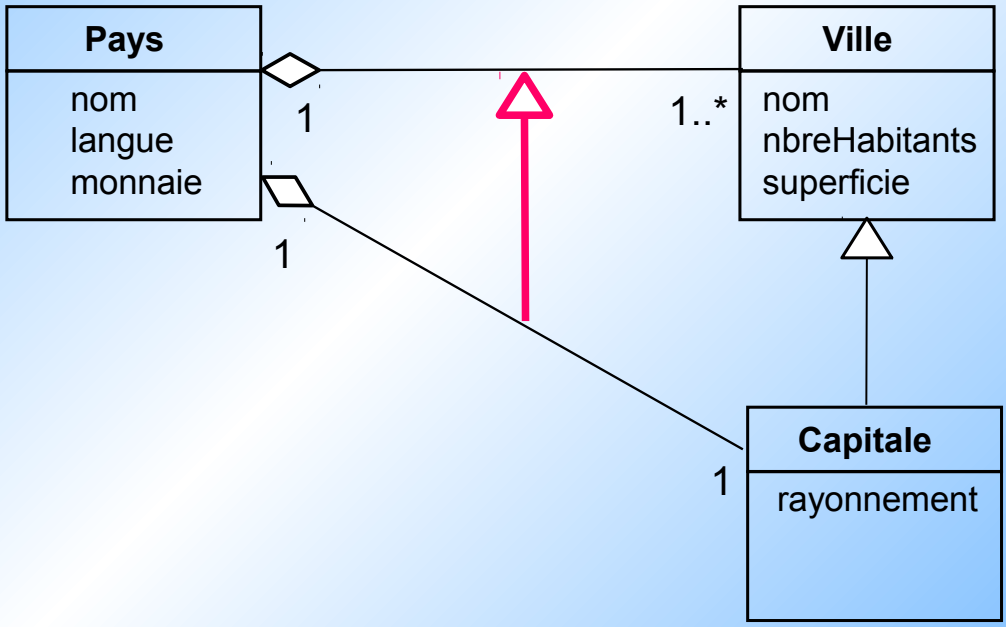
SPÉCIALISATION DE L'ASSOCIATION



- L'association générale peut être considérée comme une **association abstraite** tandis que les deux **associations** enfant sont **concrètes**.
- **Pattern** de hiérarchies de classes par paires connectées par des associations est **relativement courante** dans la généralisation d'associations.

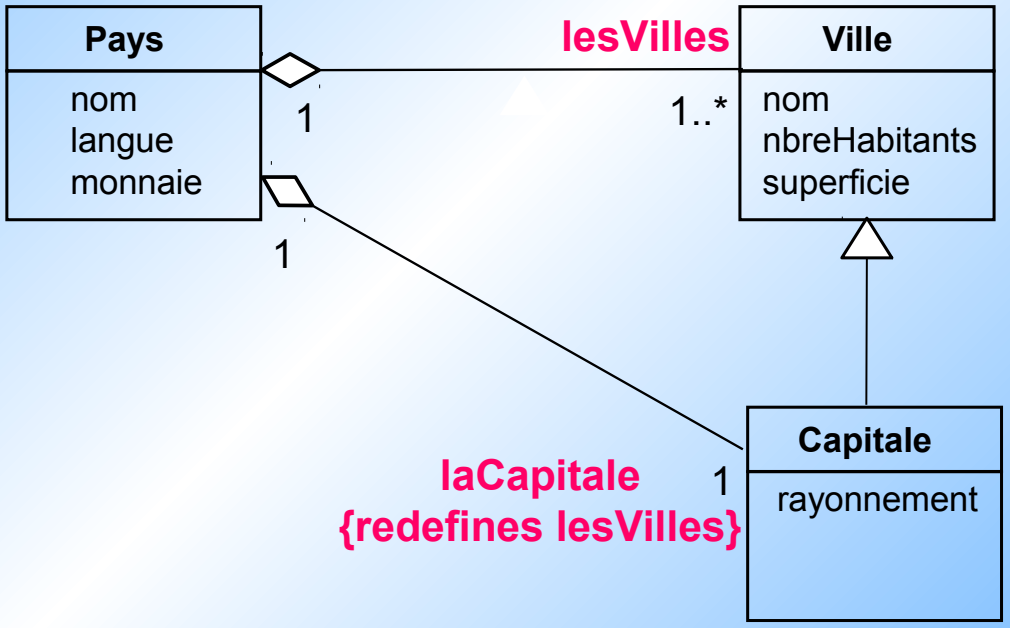
SPÉCIALISATION DE L'ASSOCIATION

- La distinction entre la définition de sous-ensembles et la spécialisation d'une association n'est pas clairement décrite dans la spécification UML2.



SPÉCIALISATION DE L'ASSOCIATION

- Peu implémenté par les outils du marché, d'où la notion de redéfinition de propriété. Le mot-clé *redefines* remplace le symbole de spécialisation entre associations



CONTRAINTE ENTRE SOUS-CLASSES

- {disjoint} ou {exclusif} : une classe descendante d'une classe A ne peut être descendante que d'une seule sous-classe de A (*défaut*).

- {chevauchement} ou {inclusif} : une classe descendante d'une classe A appartient au produit cartésien des sous-classes de la classe A. Un objet concret est alors construit à partir d'une classe obtenue par mélange de plusieurs super-classes.

- {incomplète} indique une généralisation extensible.

- {complète} indique qu'une instance est forcément d'une des sous classes (*la super classe est alors abstraite*).

CONTRAINTE ENTRE SOUS-CLASSES

Il est possible de représenter un certain nombre de contraintes d'intégrité entre sous-classes d'objets.

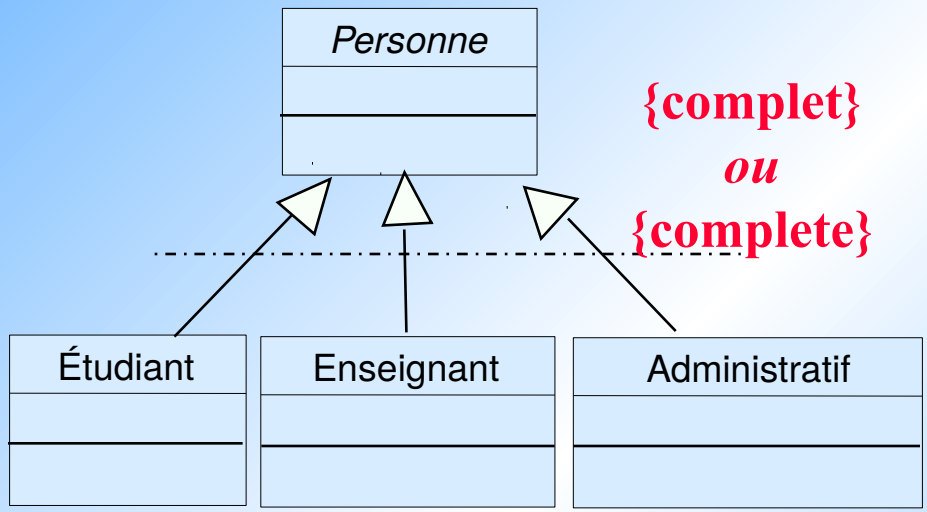
Ces contraintes correspondent aux différentes combinaisons possibles autour des contraintes de couverture et de disjonction.

Contrainte de couverture
Contrainte de disjonction

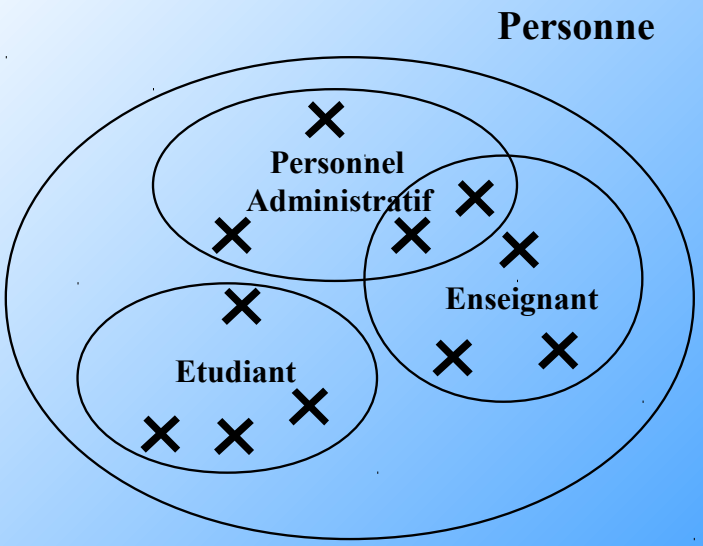
CONTRAINTE DE COUVERTURE

Contrainte de couverture : tout objet de la super-classe doit appartenir à au moins l'une des sous-classes.
(*liste exhaustive de classes*)

Par défaut : incomplet



{complet}
ou
{complete}

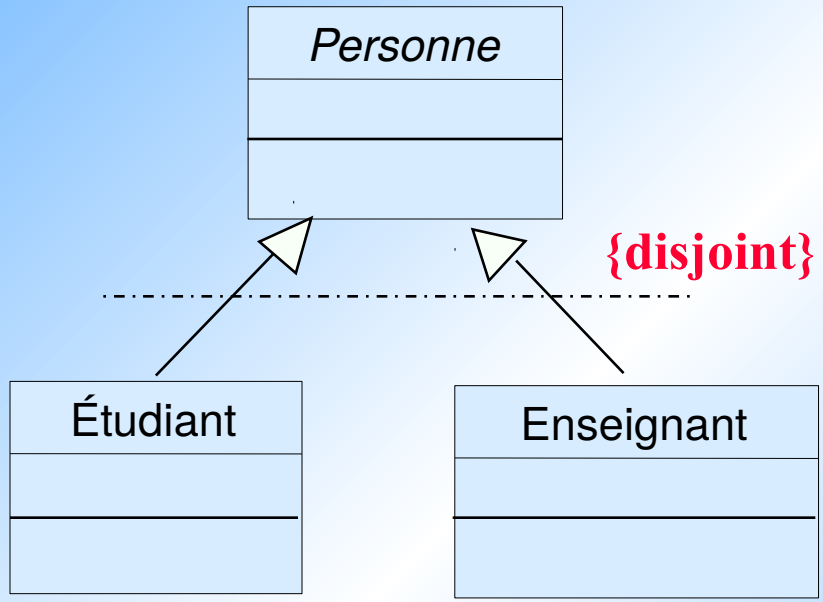


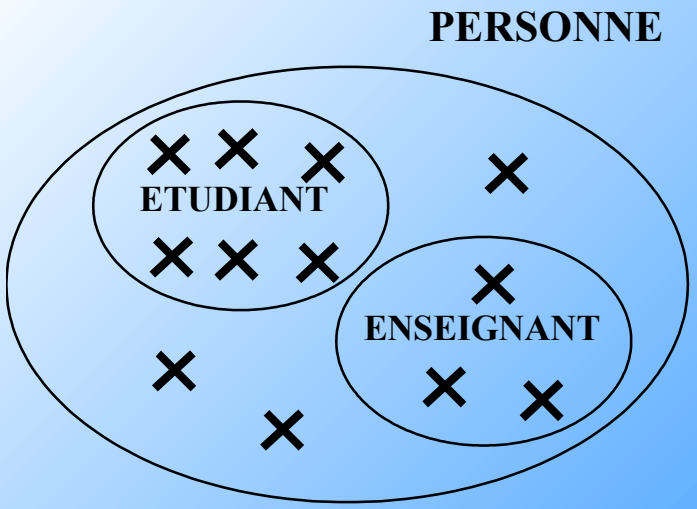
non couverture : {incomplete}

CONTRAINTE DE DISJONCTION

Contrainte de disjonction : tout objet de la super-classe doit appartenir à une seule sous-classe : les sous-classes sont mutuellement exclusives.

Par défaut : disjoint

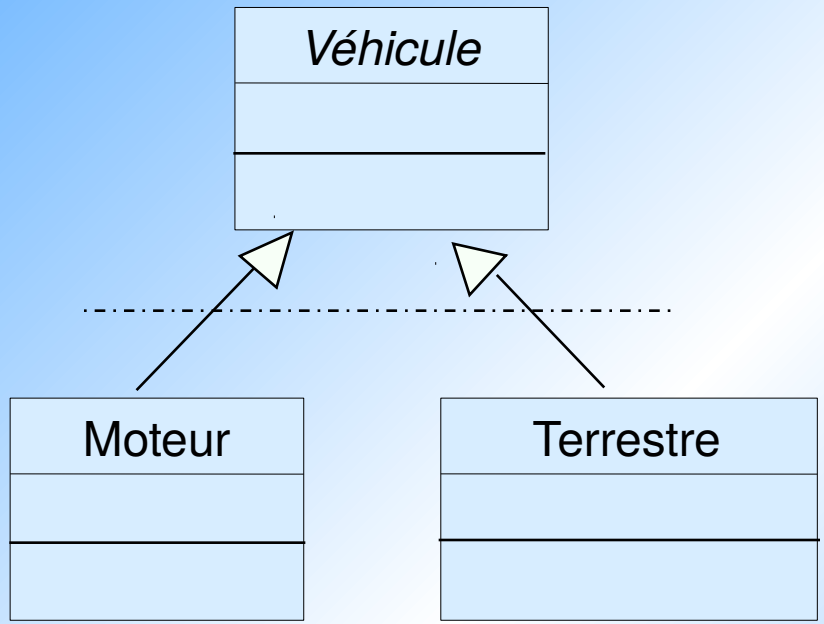


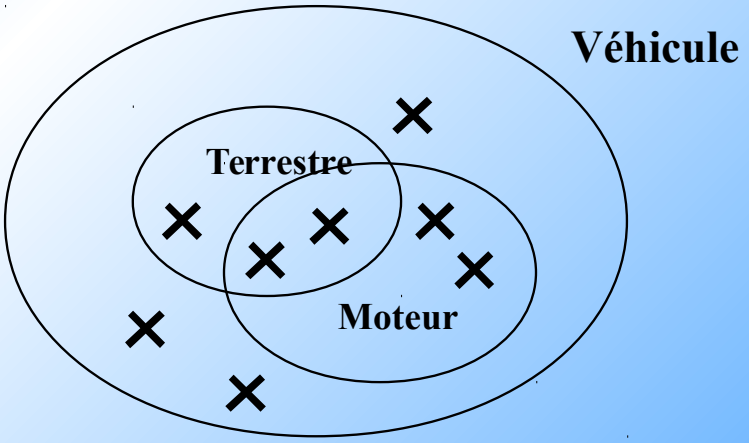


non disjonction : {overlapping}

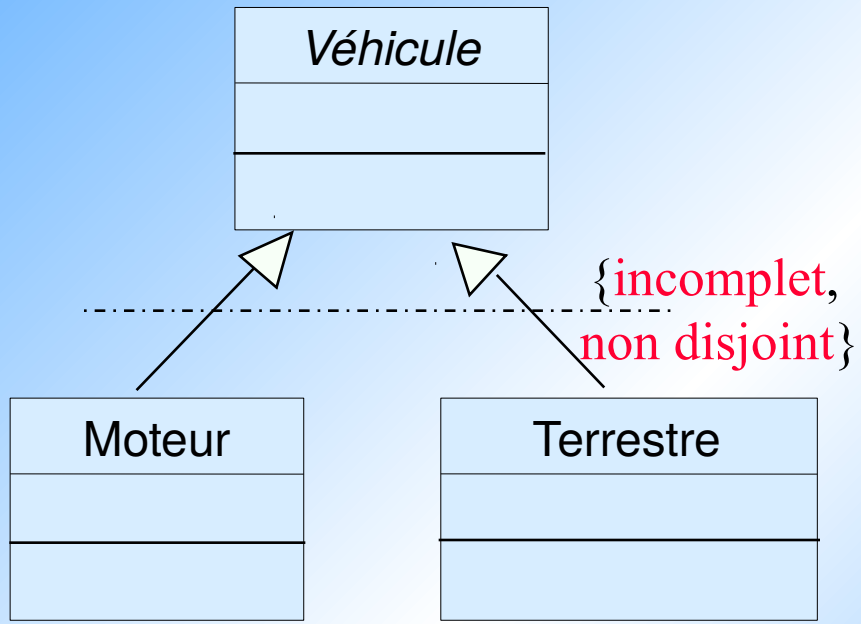
EXERCICE

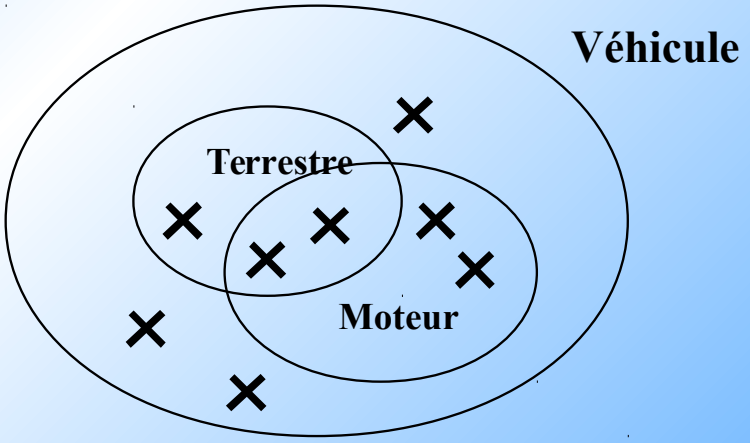
Quelles sont les contraintes vérifiées ?





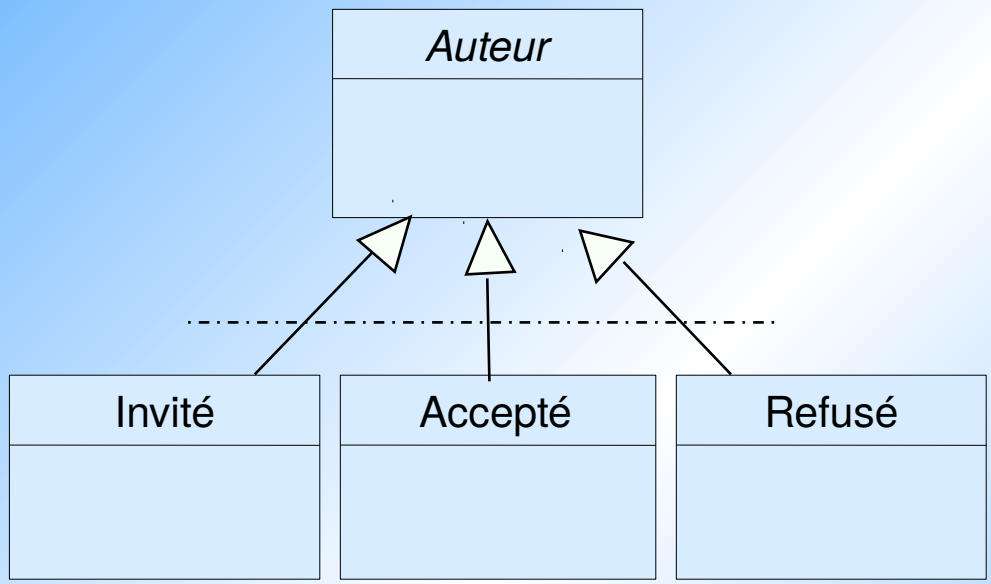
SOLUTION

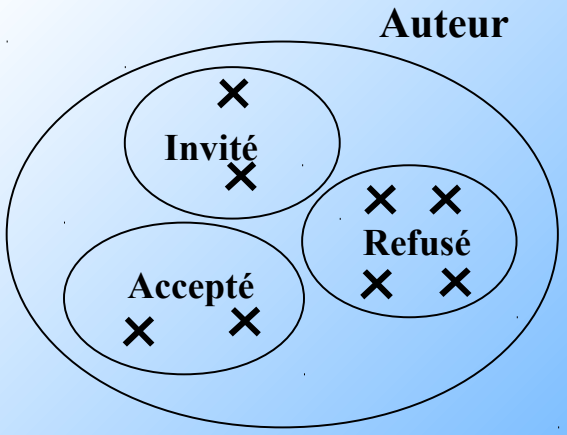

Véhicule
{incomplet,
non disjoint}
Moteur
Terrestre


Véhicule
Terrestre
Moteur

EXERCICE

Quelles sont les contraintes vérifiées ?

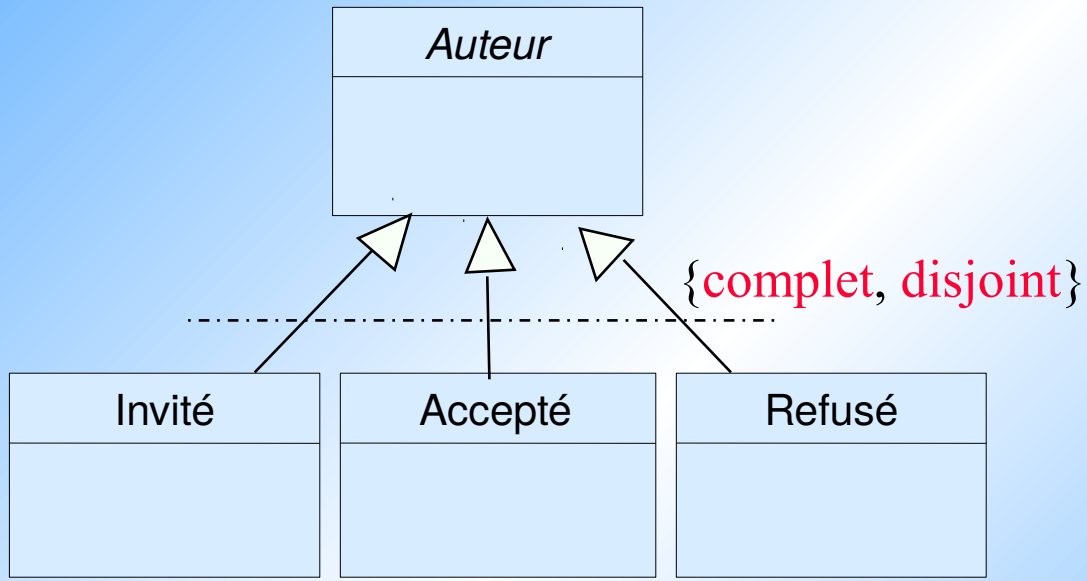


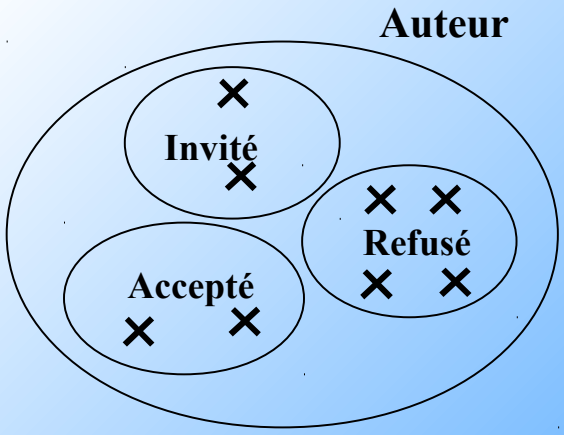


Un auteur est soit invité, soit accepté, soit refusé.

SOLUTION

{complete, disjoint}

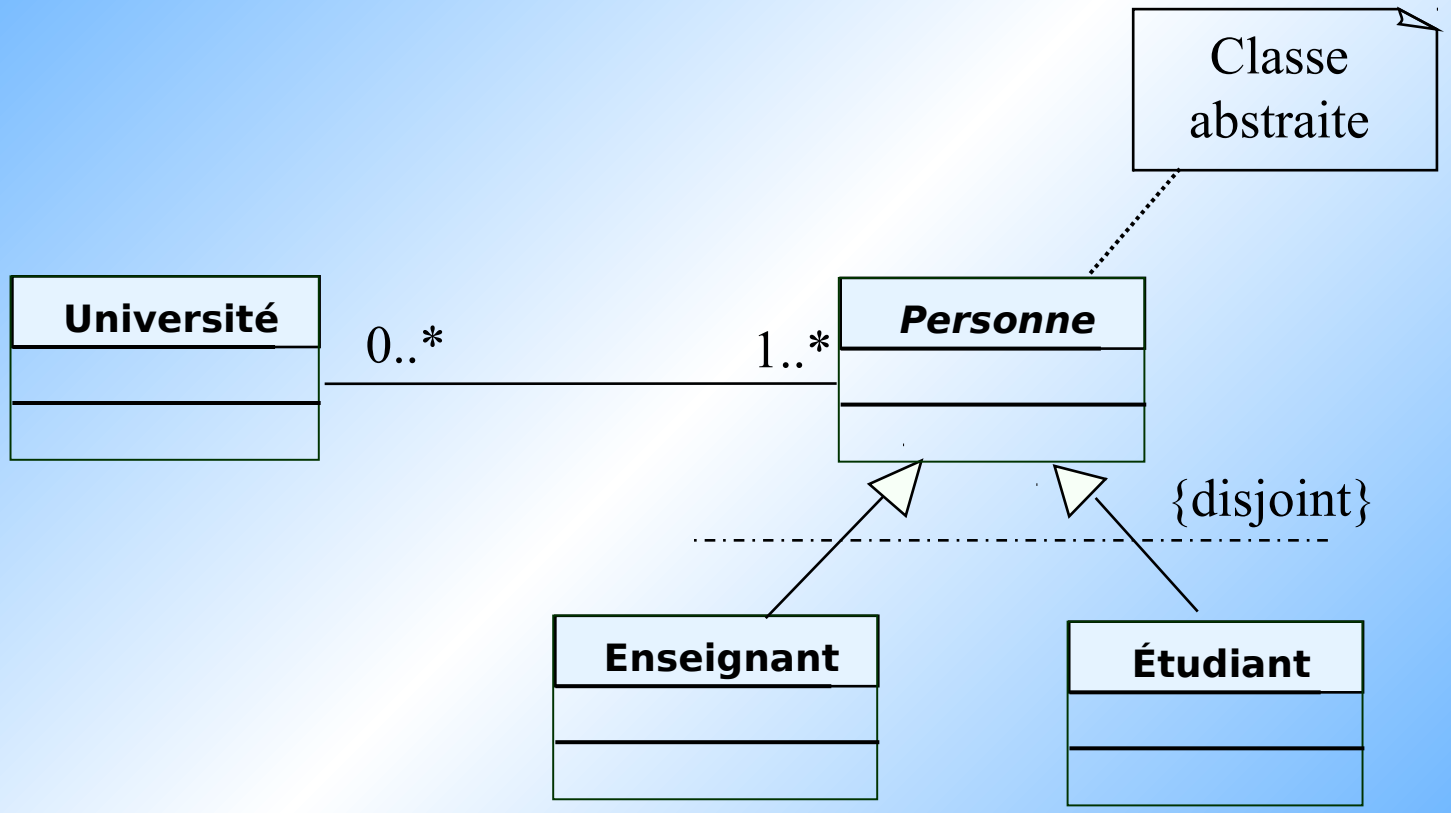




EXERCICE

Trouver la hiérarchie de classes qui reflète la contrainte du *"ou exclusif "*

SOLUTION


Classe abstraite
Université
0..*
1..*
Personne
{disjoint}
Enseignant
Étudiant

STRUCTURATION EN PAQUETAGES

Paquetage = package en anglais

La structuration d'un modèle est une activité délicate. Elle doit s'appuyer sur deux principes fondamentaux :

1- cohérence

2- ***indépendance***.

STRUCTURATION EN PAQUETAGES

- Le **premier principe** (*cohérence*) consiste à regrouper les classes qui sont proches d'un point de vue sémantique.
Un critère intéressant consiste à évaluer les durées de vie des instances de concept et à rechercher l'homogénéité.

- Le **deuxième principe** (*indépendance*) s'efforce de minimiser les relations entre paquetages, c'est-à-dire plus concrètement les relations entre classes de paquetages différents.

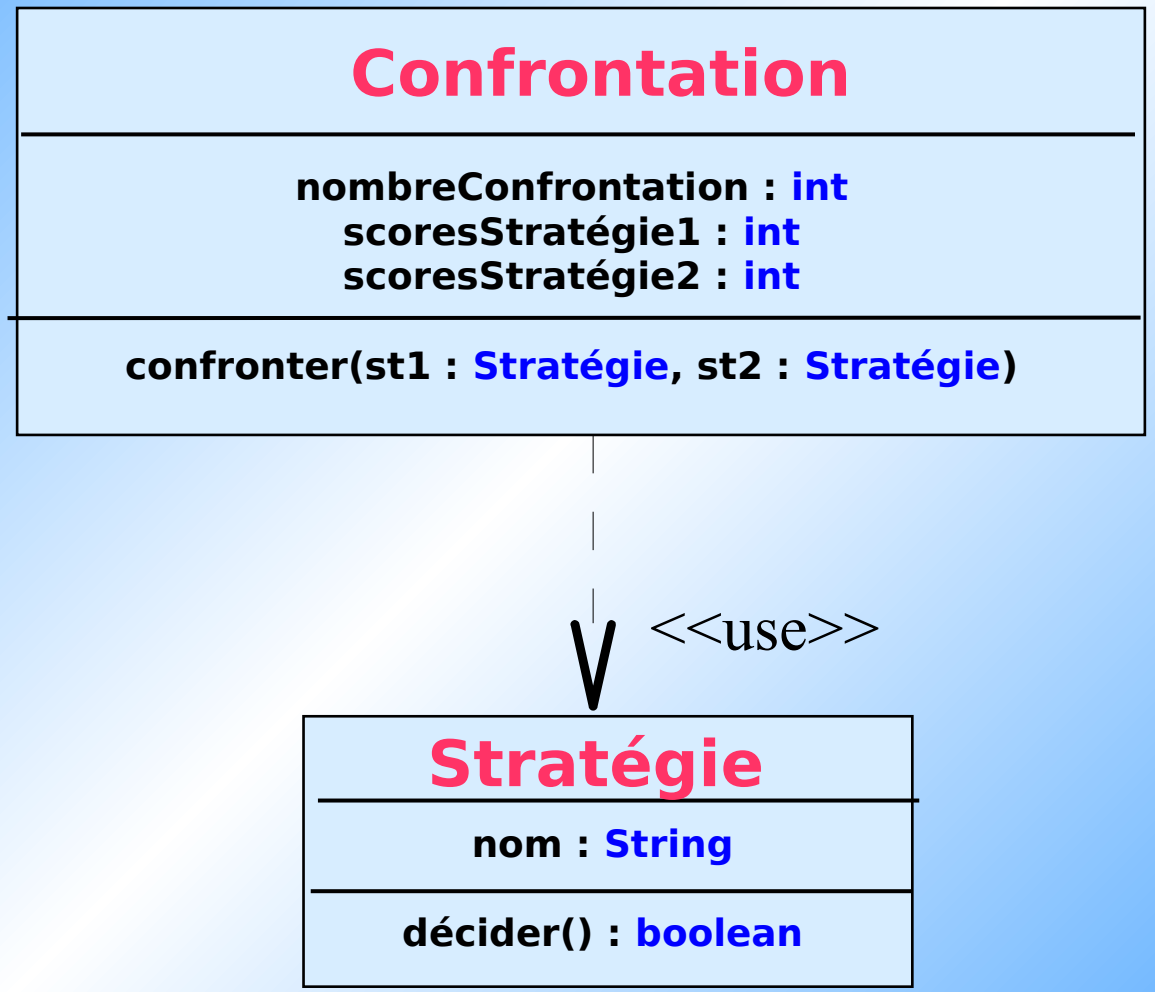
RELATIONS DE DÉPENDANCE

- Une dépendance est une relation unidirectionnelle exprimant une dépendance sémantique entre des éléments du modèle.

- Une dépendance est habituellement utilisée quand une classe en utilise une autre comme argument dans la signature d'une opération.

- Elle indique que la modification de la cible peut impliquer une modification de la source.

Relations de dépendance



INTERFACES

- Liste d'opérations constituant un contrat à respecter par les classes implémentant l'interface

- Déclarations de propriétés et de méthodes **publiques** mais aucune implémentation de celles-ci (*méthodes abstraites*)

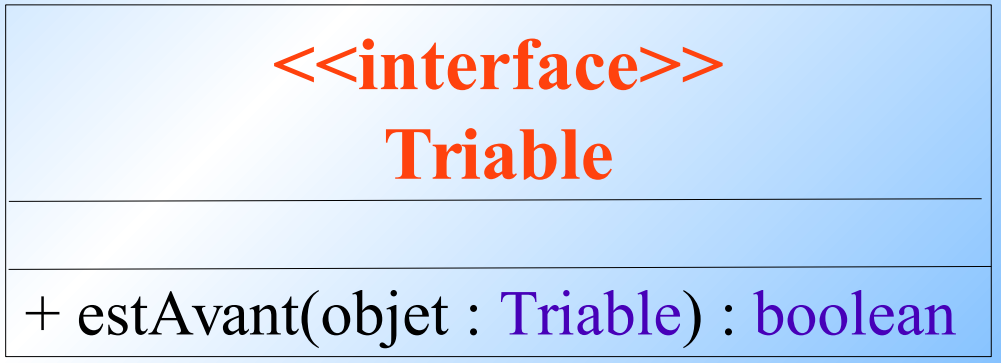
- Prendre en compte le langage d'implémentation du modèle
  (*C++ pas de concept d'interface,*
  Java pas de propriété, constantes uniquement)

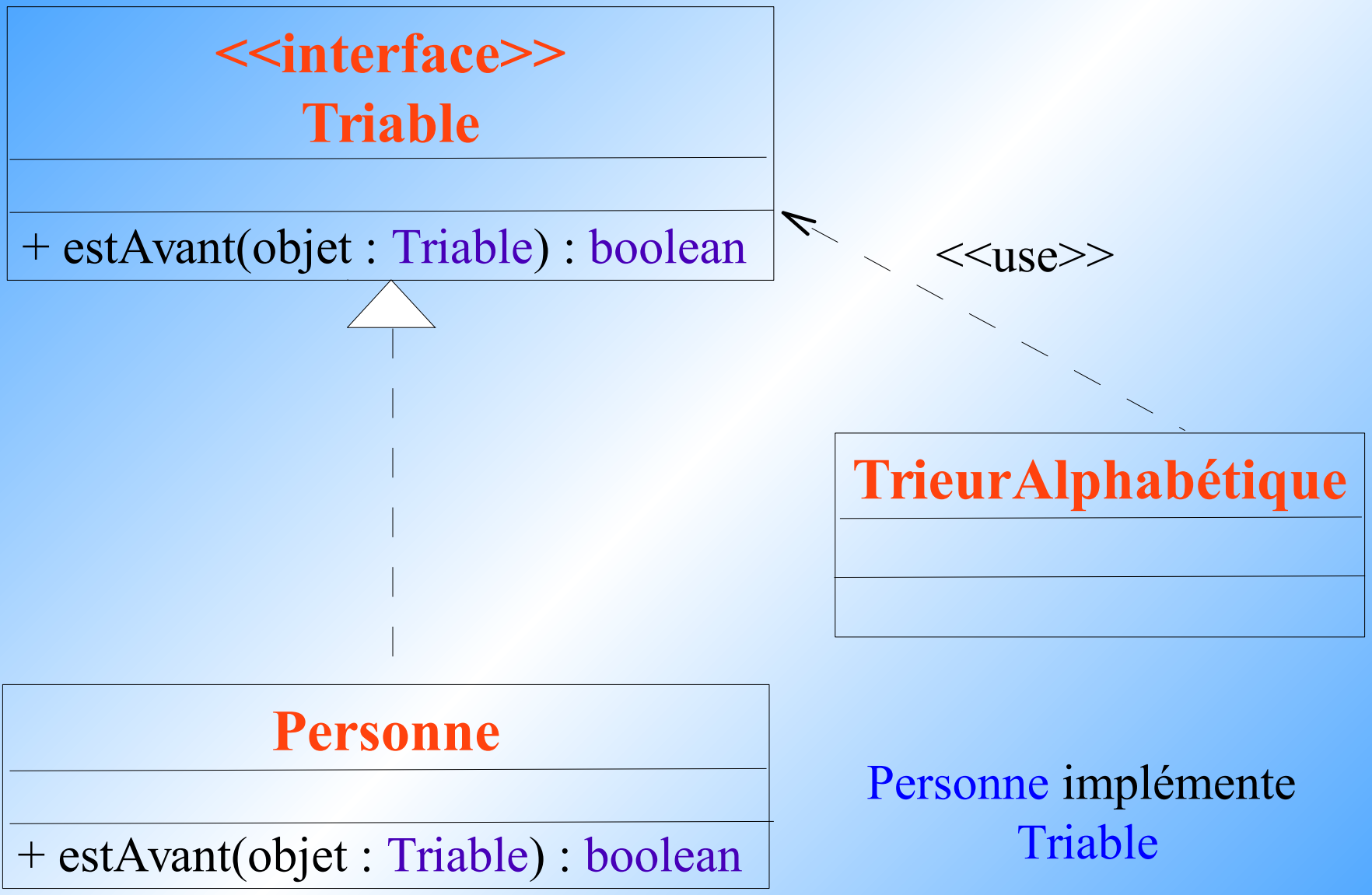
- Une propriété d'interface indique que toute classe implémentant celle-ci doit stocker l'information et fournir les moyens de la manipuler.

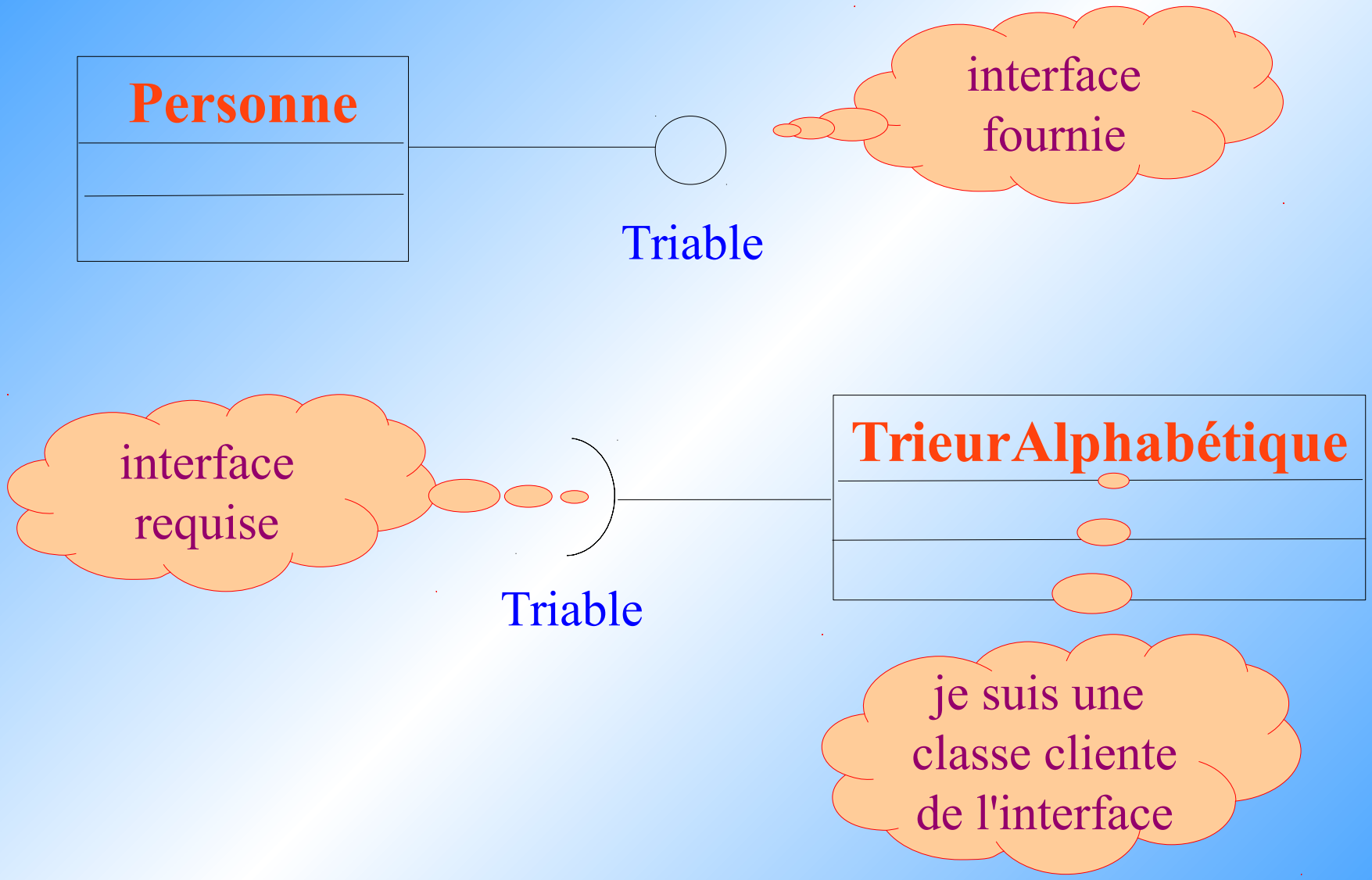
INTERFACES

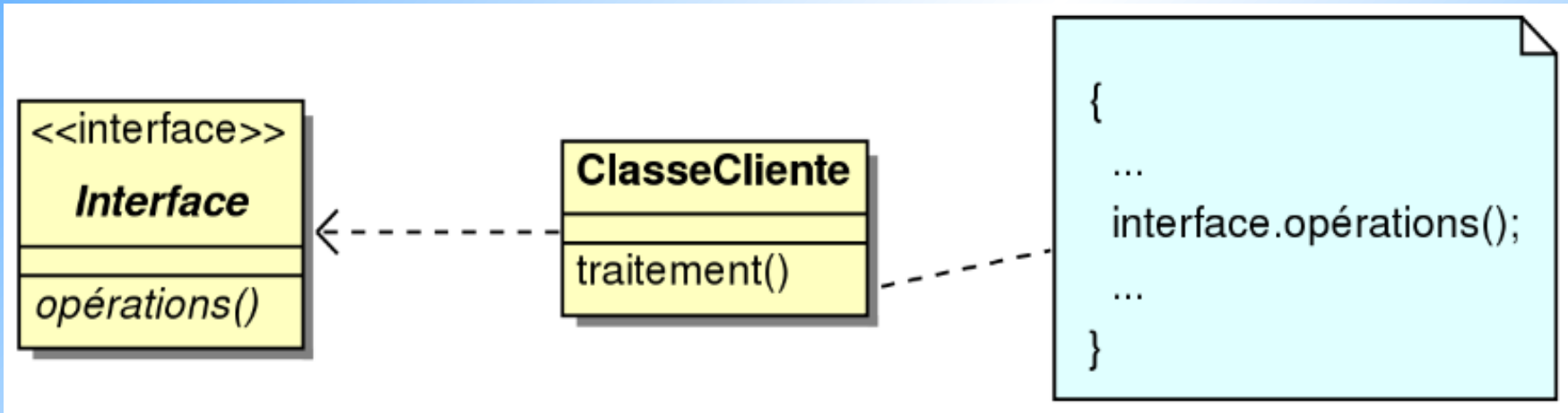
- 2 types de représentations
  (*fonction de ce que l'on doit mettre en lumière*) :
  - Classe avec stéréotype interface
  - Notation à rotule

Classe avec stéréotype

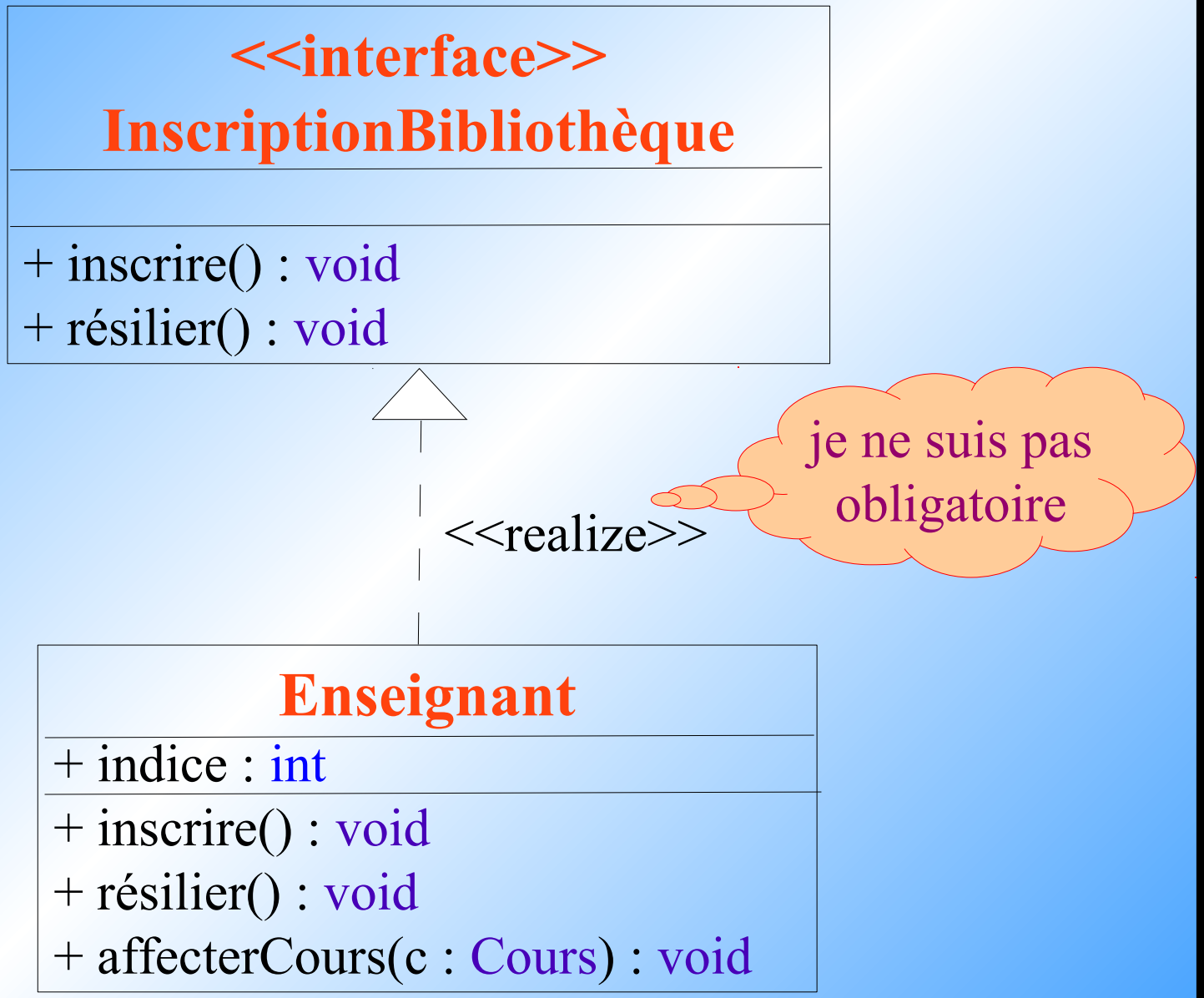


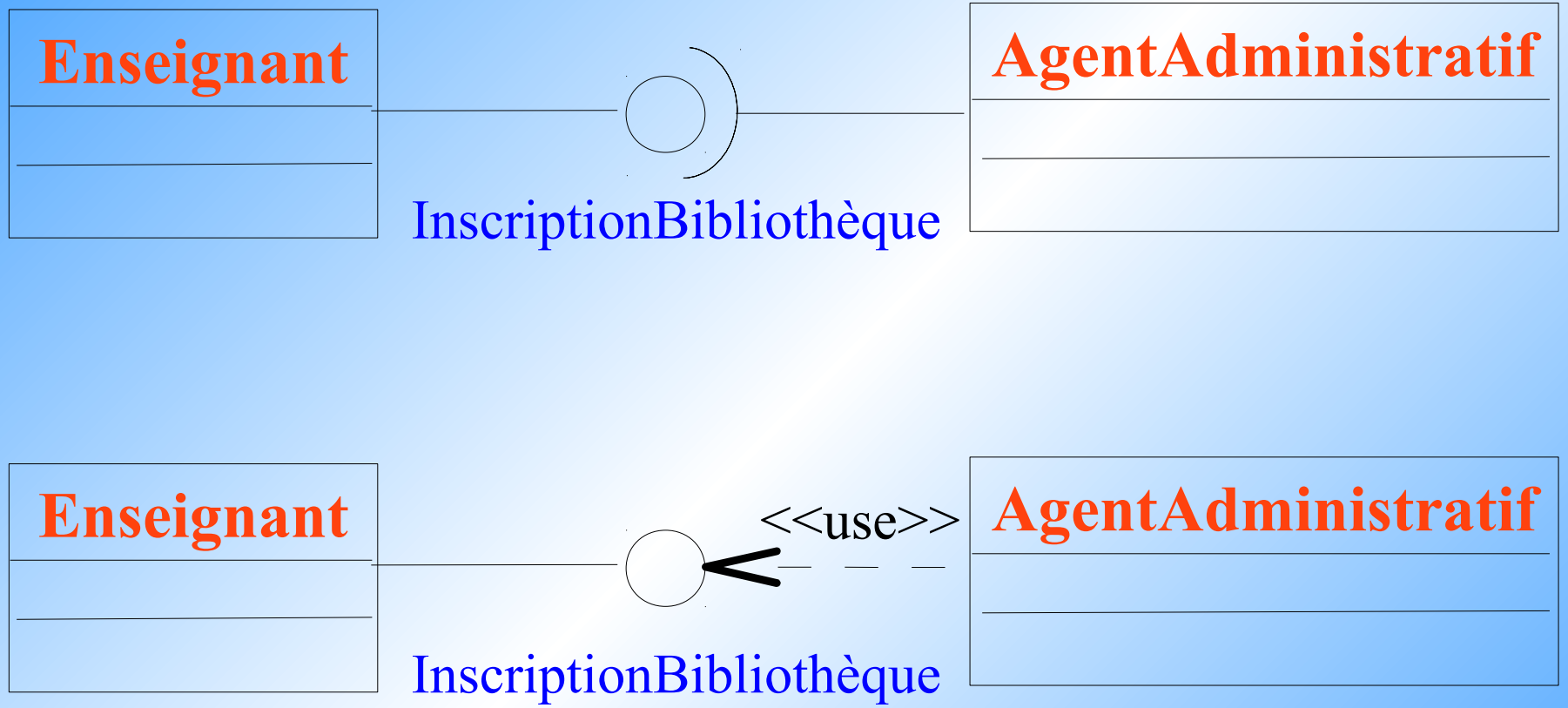

<<interface>>
Triable
+ estAvant(objet : Triable) : boolean
<<use>>
TrieurAlphabétique
Personne
+ estAvant(objet : Triable) : boolean
Personne implémente
Triable


Personne
interface
fournie
Triable
interface
requise
Triable
TrieurAlphabétique
je suis une
classe cliente
de l'interface



(Transparent de Pierre Gérard – IUT de Villetaneuse)


<<interface>>
InscriptionBibliothèque
+ inscrire() : void
+ résilier() : void
<<realize>>
je ne suis pas obligatoire
Enseignant
+ indice : int
+ inscrire() : void
+ résilier() : void
+ affecterCours(c : Cours) : void

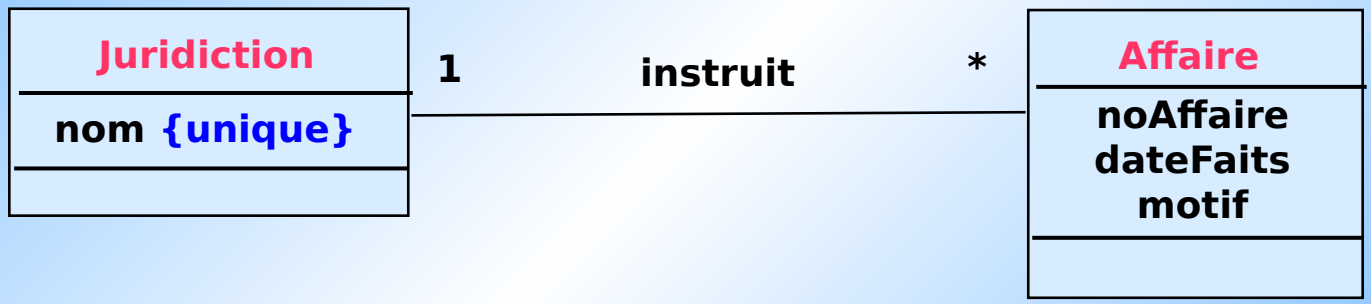

Enseignant
AgentAdministratif
InscriptionBibliothèque
Enseignant
<<use>>
AgentAdministratif
InscriptionBibliothèque

EXERCICE

Différentes juridictions peuvent instruire des affaires. Une juridiction est caractérisée par un nom unique. Une affaire est caractérisée par un numéro, une date et un motif. Le numéro d'une affaire est unique pour une juridiction donnée, mais pas dans l'absolu. En effet, deux affaires disctinctes peuvent se voir attribuer le même numéro si elles sont instruites par deux juridictions distinctes.

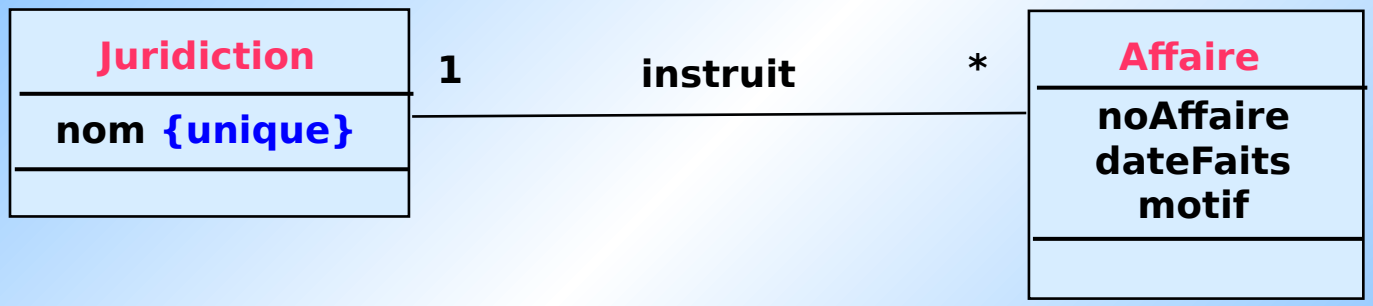
SOLUTION

Que pensez-vous de ce modèle ?



SOLUTION

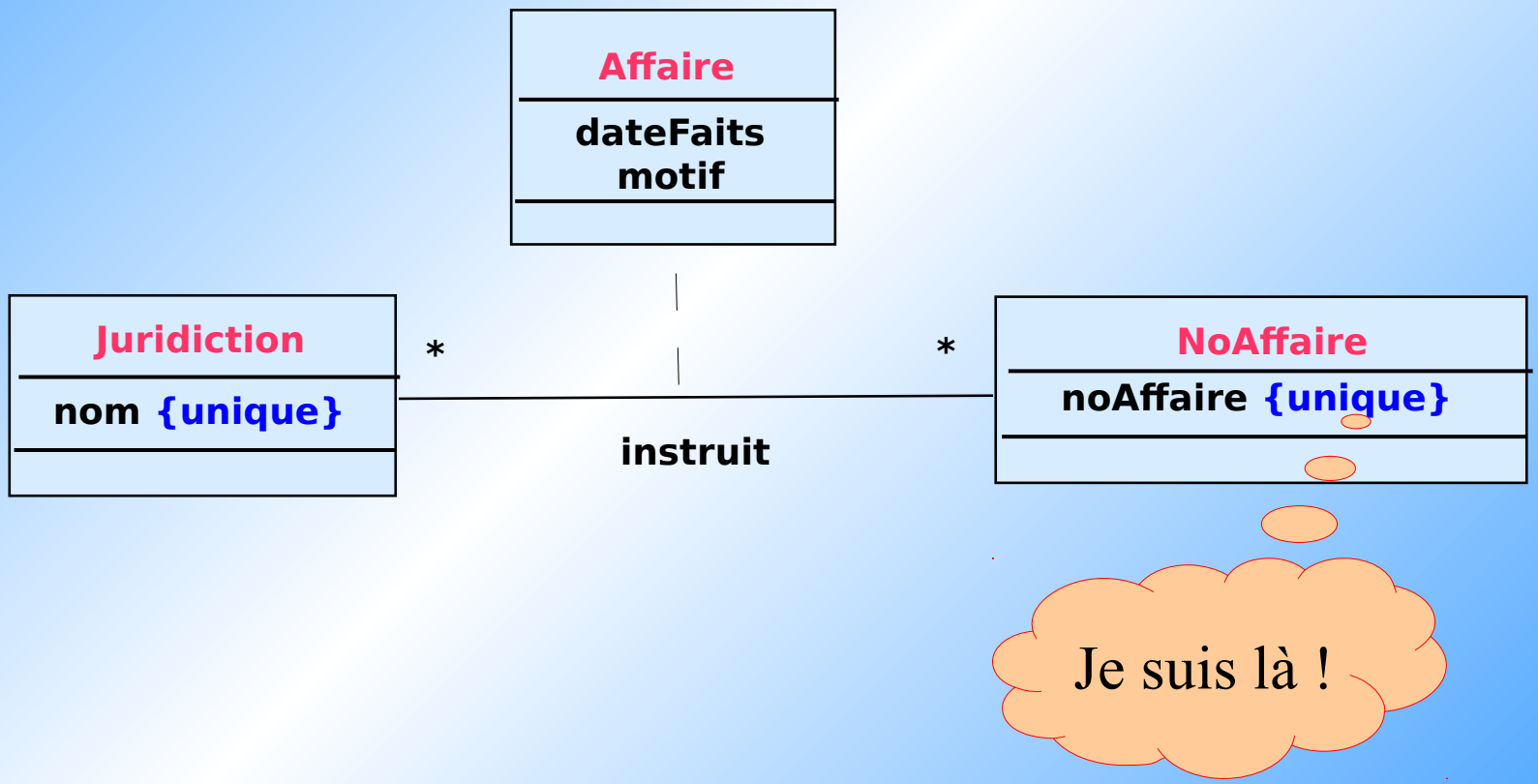
Que pensez-vous de ce modèle ?



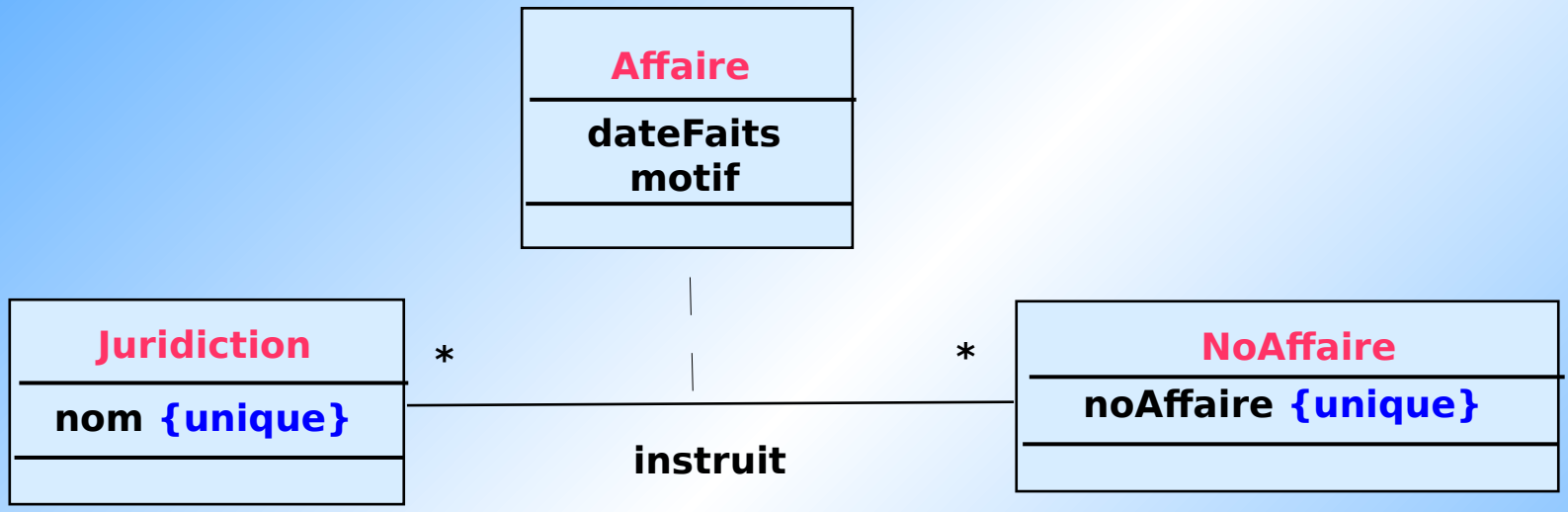
Il ne rend pas compte de l'unicité de noAffaire pour une juridiction donnée

SOLUTION

Que pensez-vous de ce modèle ?



SOLUTION



Prend en compte l'unicité de noAffaire pour une juridiction donnée

Introduction d'une classe très artificielle : NoAffaire

SOLUTION

Que pensez-vous de ce modèle ?