Introduction

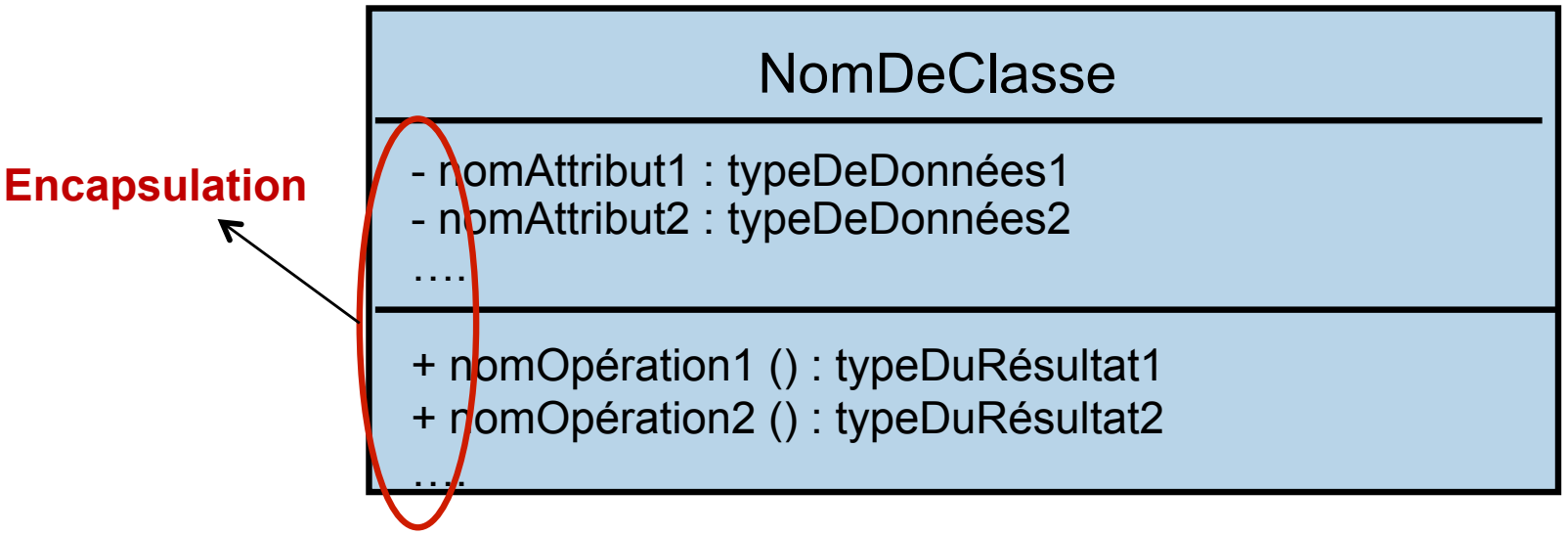
- Diagramme de Classes :

  - Description de tout ou partie du système d'une manière abstraite, par une collection d'éléments de modélisation (classes, relations, …)
  - Il sert en premier lieu à modéliser les entités du système d'information et les informations qui les caractérisent
  - Vue statique : abstraction des aspects dynamiques et temporels

La classe

- **Une classe est la description d'un groupe d'objets ayant :**
  - même structure (même ensemble d'attributs)
  - même comportement (mêmes opérations)
  - une sémantique commune
- Une classe est associée à d'autres classes
- **Exemples de classes** :
  - article, personne, société, processus, fenêtre,…
  - Les objets qui se conforment à la description d'une classe sont appelés ses **instances**

Représentation graphique



Visibilités :

public ou + : tout élément qui peut voir l'objet peut également voir l'élément indiqué

protected ou # : seul un élément situé dans l'objet ou un de ses descendants peut voir l'élément indiqué

private ou - : seul un élément situé dans l'objet peut voir l'élément

package ou ~ ou rien : seul un élément déclaré dans le même paquetage peut voir l'élément

Bonnes pratiques :

Privé (-) pour les attributs

Public (+) pour les opérations

Principe d'encapsulation

- Principe d'encapsulation (données-opérations)
  - un objet est considéré comme une boîte noire
  - un objet n'est accessible que par des opérations visibles (définies dans une interface externe)
  - interface : la vue externe d'un objet vis-à-vis des autres objets
  - il est appelé aussi masquage d'informations : séparation de la partie publique (visible) de la partie privée
- Le changement d'implémentation des opérations ne modifie pas l'interface de l'objet

  ⇒ stabilité de l'utilisation des objets facilite l'évolution d'une application
- Accès direct aux attributs des objets interdit

  ⇒ l'encapsulation garantit l'intégrité des données

Notion d'attributs

- **Un attribut**
  - propriété rattachée à la classe
  - caractérisé par un nom, un type, une visibilité et peut être initialisé
  - Le nom de l'attribut doit être unique dans la classe
  - exemple :
    - pour la classe Article : référence, désignation ...
    - Pour la classe Personne : nom, adresse, dateDeNaissance,…
- La syntaxe de la déclaration d'un attribut :

  [<visibilité>] <nom_attribut>: <Type> ['[' <multiplicité> ']' ['{' <contrainte> '}']] [= <valeur_par_défaut>]

  <Type> : nom de classe, nom d'interface ou type de donné

  <multiplicité> : nombre de valeurs que l'attribut peut contenir

  (si > 1 possibilité d'ajout d'une contrainte)

  <contrainte> : précision sur les valeurs ({ordered}, {list} …).

Notion d'identifiant de la classe

- **Identifiant de la classe = attribut particulier permettant de repérer de façon unique chaque objet, instance de la classe**

- L'identifiant doit avoir une utilité pour le gestionnaire en lui permettant de se repérer sans ambiguïté dans un ensemble d'objets analogues

- Chacun des autres attributs de la classe est relié à l'identifiant par une relation appelée **dépendance fonctionnelle**
  - *Rappel : Soit deux attributs a et b, on dit que b dépend fonctionnellement de a si connaissant la valeur de a, on peut trouver la valeur de b*
  - Exemple : Connaissant la référence d'un article, on peut trouver sa désignation, son prix unitaire, la quantité disponible en stock,…
    Connaissant le numéro du client, on peut trouver son nom, son adresse, son téléphone, ses conditions commerciales,...

Notion d'opérations (1/2)

- **Une opération**
  - Traitement que l'on peut faire sur un objet instance de classe
  - Elle a un nom et peut avoir des paramètres et un résultat.
  - définie de manière globale au niveau de la classe
  - Dans une même classe, 2 opérations ne peuvent pas avoir le même nom et les mêmes types de paramètres (unicité de l'interface)
  - On ne déclare que l'interface de l'opération
  - Appel : o.op()
  - exemple :
    - pour la classe Article : vendre, acheter, prix TTC
    - pour la classe Personne : calculerAge, changerAdresse,…

Notion d'opérations (2/2)

- Syntaxe de la déclaration d'une opération :

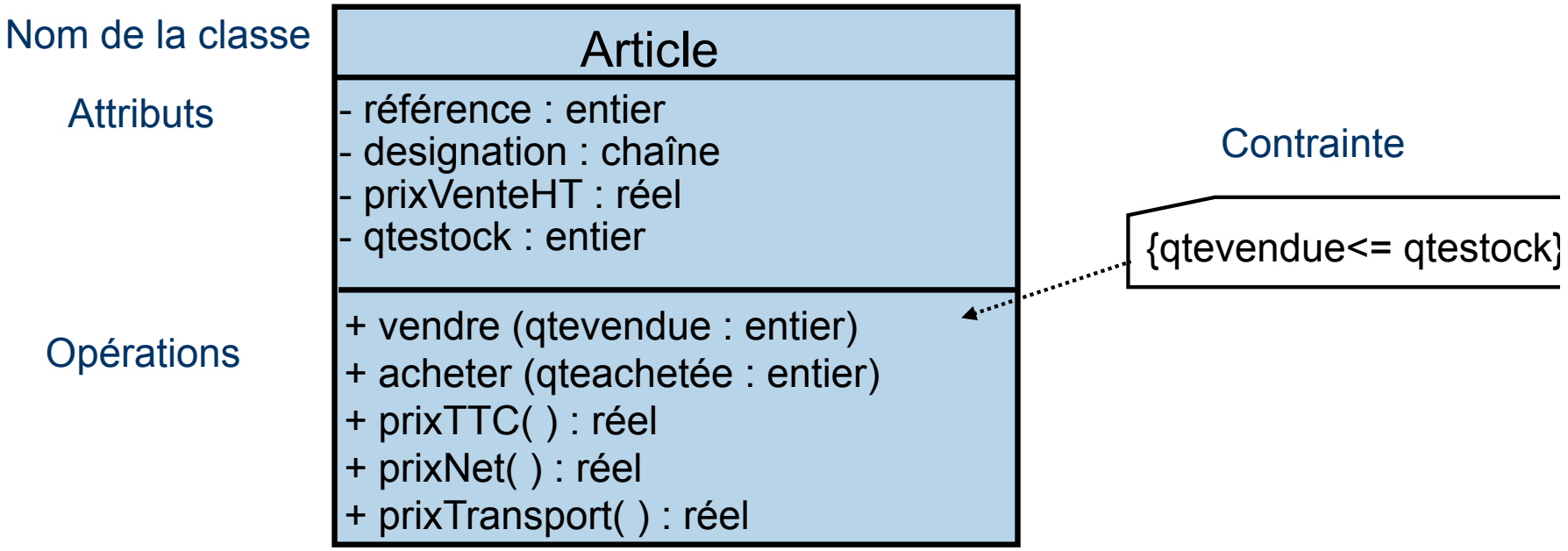
<visibilité> <nom_méthode> ([<paramètre> [, <paramètre> [, <paramètre> ...]]]) : [<valeur_renvoyée>] [{ <propriétés> }]

La syntaxe de définition d'un paramètre (<paramètre>) :

[<direction>] <nom_paramètre> : <Type> ['['<multiplicité>']'] [=<valeur_par_défaut>]

- <direction> :
  - **in** : paramètre d'entrée non modifiable (par défaut)
  - **out** : paramètre de sortie modifiable
  - **inout** : paramètre d'entré modifiable
- <Type> classe, interface ou type de donné
- <propriétés> ∫ contraintes ou informations (exceptions, les préconditions, les postconditions ou indication qu'une méthode est abstraite (mot-clef abstract), …)

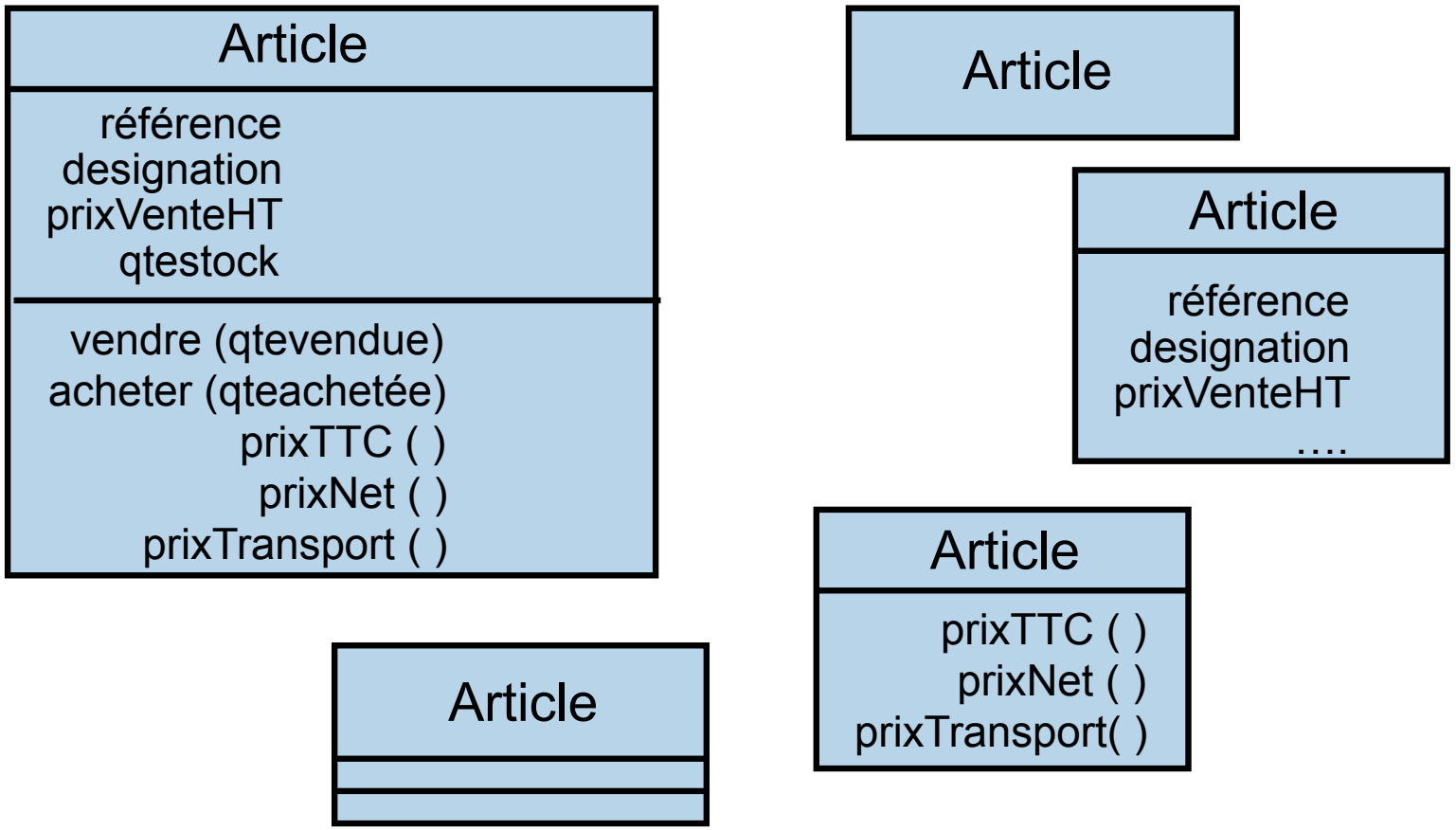
Exemple de classe



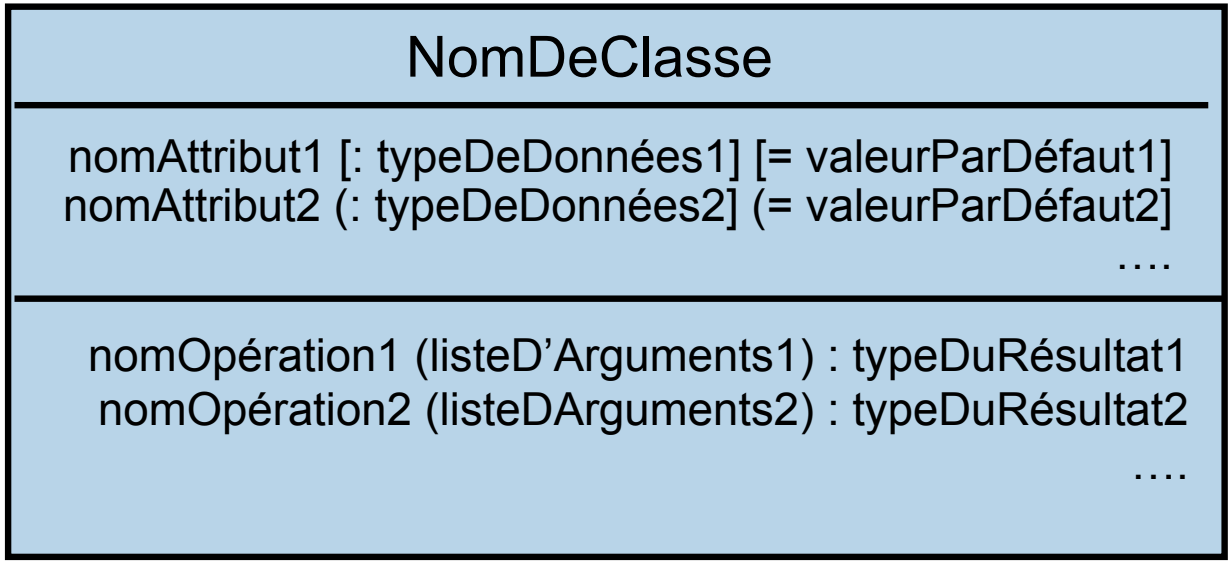
Conventions :

- Les noms de classes commencent par une majuscule
- Les noms d ’attributs et d ’opérations commencent par une minuscule
- Pas de représentation particulière pour les identifiants

Autres représentations possibles…



Représentation des classes (résumé)



Opérations d'une classe

- **Cas particuliers d'opération :**
  - **Les constructeurs :**
    - Opérations qui permettent de créer un objet instance de la classe
    - Plusieurs constructeurs peuvent exister pour la même classe.
    - Exemple pour la classe Etudiant :

      creer() : Etudiant ;

      creer (CIN, CNE, Nom, Prénom) : Etudiant ;

Surcharge du 1er

Opérations d'une classe

- **Cas particuliers d'opération :**
  - **Les accesseurs :**
    - Opérations qui permettent d'accéder aux attributs des objets soit en lecture (get) soit en écriture (set)

    - Exemple pour la classe Etudiant :

      getNom () : String

      getCNE() : String

      Accès à la valeur d'un attribut

      setAdresse(adr : String) : void

      Mise à jour de la valeur d'un attribut

Attributs et méthodes de classe

- Attribut de classe
  - Un attribut qui s'applique à une classe et non à une instance de cette classe
  - Dans un DC, les attributs de classe sont soulignés
- Méthode de classe
  - Un méthode qui s'applique à une classe et non à une instance de cette classe
  - Dans un DC, les méthodes de classe sont soulignées

L’objet

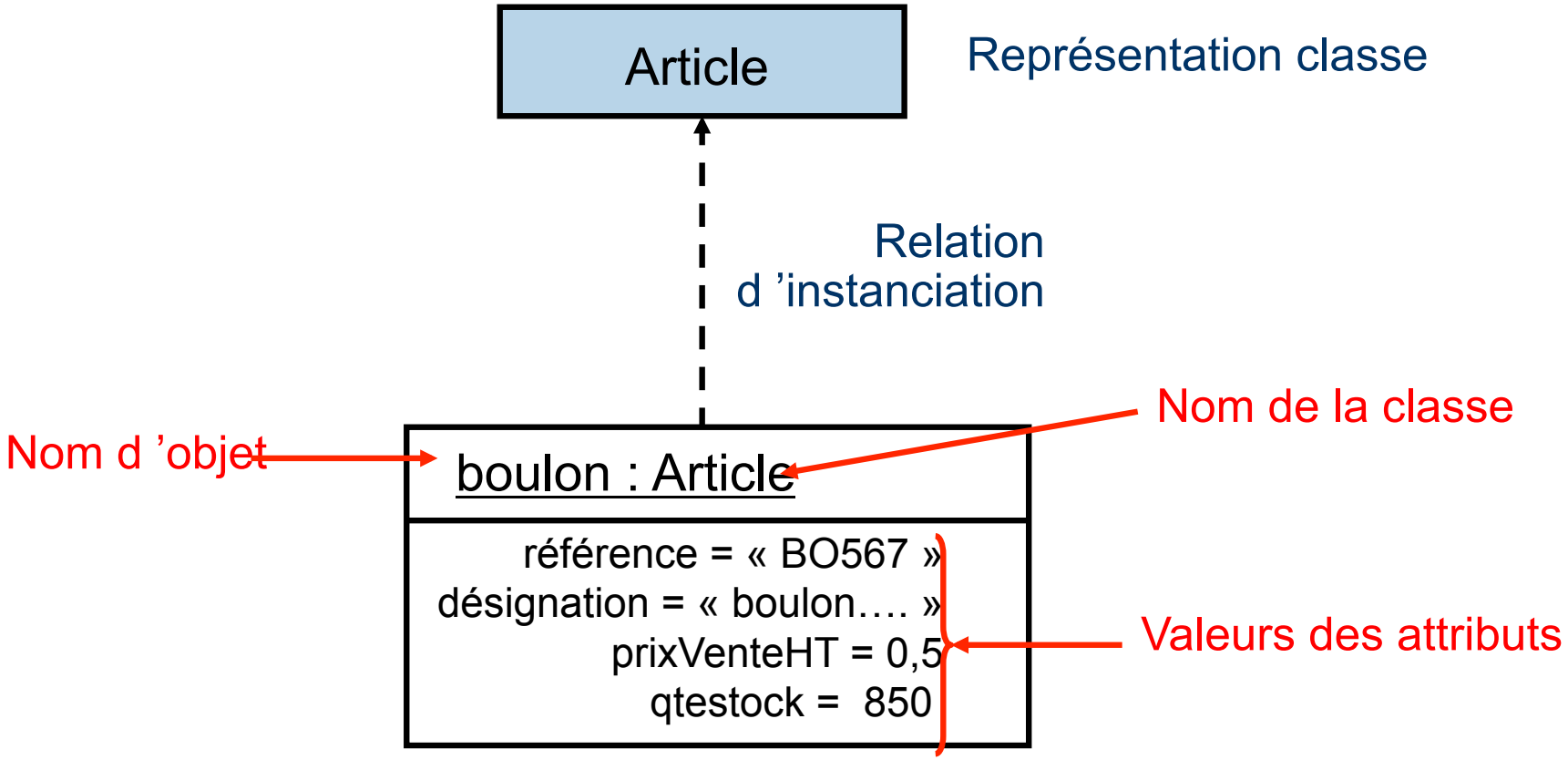
- Un objet est instance d’une (seule) classe :
  - il est conforme à la description fournie par la classe
  - il admet au plus une valeur pour chaque attribut de la classe
  - il est possible de lui appliquer toute opération définie dans la classe

- Tout objet admet une identité qui le distingue des autres objets
  - Permet de distinguer tout objet de façon non ambiguë, et cela indépendamment de son état

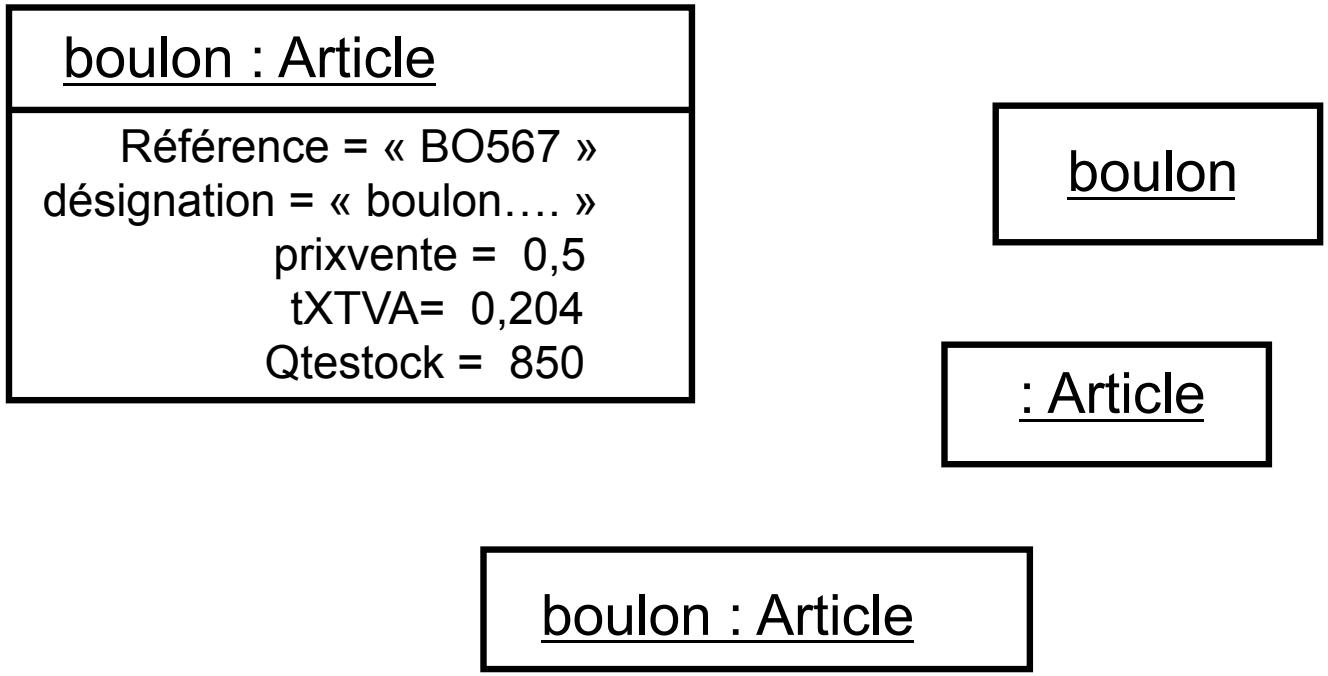
Etat et comportement d'un objet

- **État d'un objet :**
  - Ce sont les attributs et les terminaisons d'associations qui décrivent l'état d'un objet
  - Les propriétés décrites par les attributs prennent des valeurs lorsque la classe est instanciée
  - L'instance d'une association est appelée un lien
- **Comportement d'un objet :**
  - Les opérations décrivent les éléments individuels d'un comportement que l'on peut invoquer
  - Ce sont des fonctions qui peuvent prendre des valeurs en entrée et modifier les attributs ou produire des résultats
  - Une opération est la spécification (i.e. déclaration) d'une méthode
- Les attributs, les terminaisons d'association et les opérations constituent donc les propriétés d'une classe (et de ses instances)

Représentation d'un d'objet



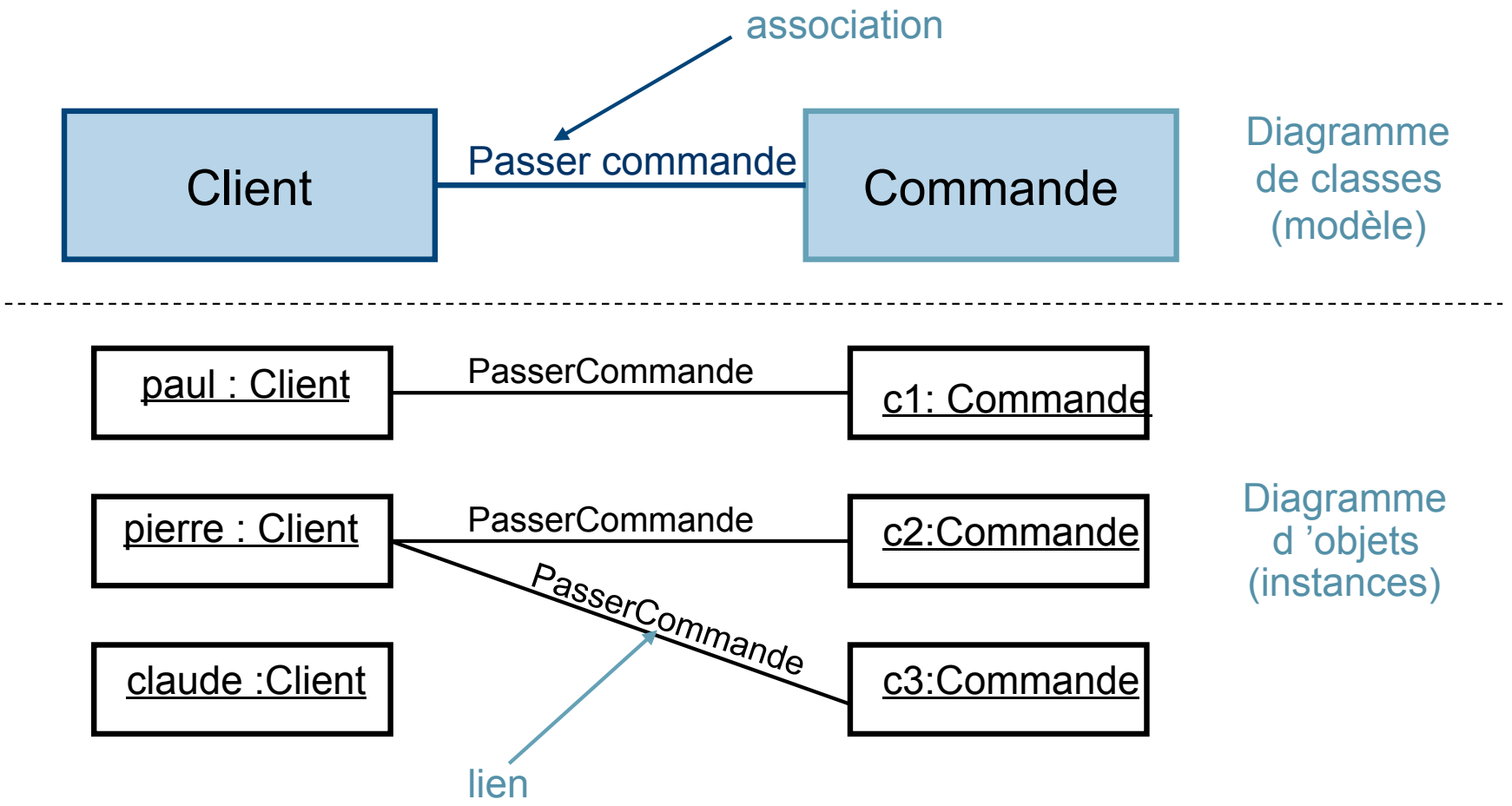
Autres représentations possibles…



Les associations

- Les associations entre classes expriment les liens structurels entre leurs instances
- Une association est identifiée par la concaténation des identifiants des classes qui participent à l'association
  - Chaque instance d'une association est identifiée de manière unique par le tuple des valeurs des identifiants de ces classes
- Les terminaisons d'associations sont des propriétés structurelles de la classe (comme les attributs)
- La plupart des associations sont des associations binaires (éventuellement réflexives)
- Il existe aussi des associations n-aires

Représentation des associations (entre classes)



Conventions :

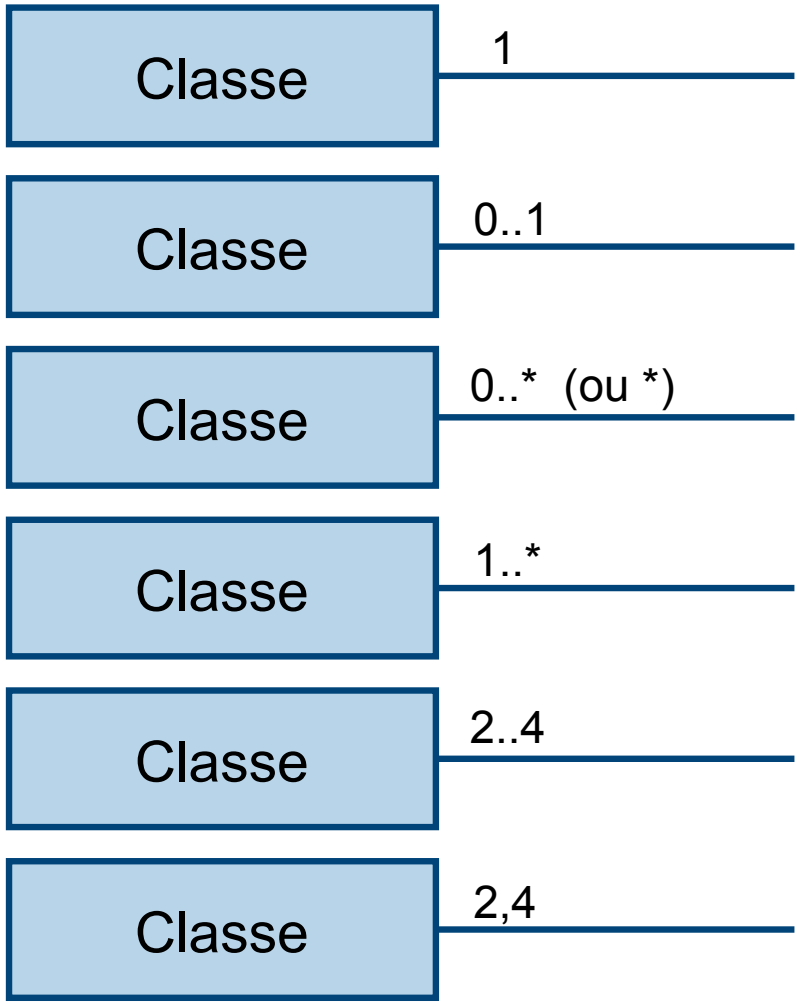
- Les noms des liens sont des formes verbales et commencent par une majuscule.

Multiplicités d'une association

- Précise combien d'objets peuvent être liés à un seul objet source
- Multiplicité minimale et maximale ($\mathbf{C_{min}}$ . . $\mathbf{C_{max}}$).

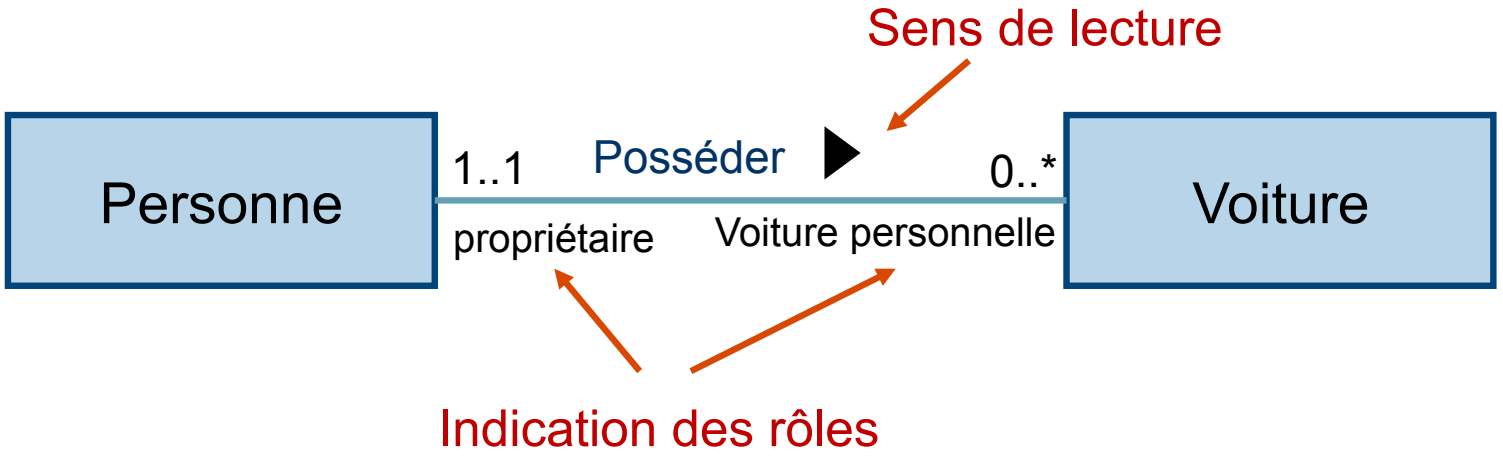
« Une personne possède 0 ou plusieurs voitures »
« Une voiture a 1 et 1 seul propriétaire »

Multiplicités d'une association

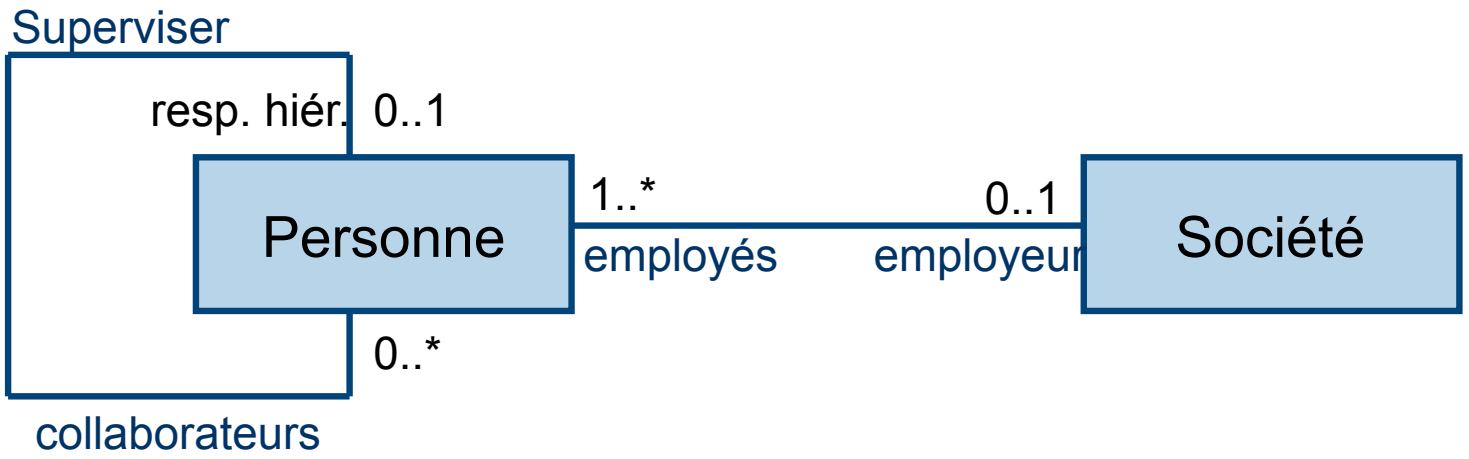


Notation des noms de rôles et sens de lecture

- Pour améliorer la lisibilité des associations, on peut indiquer :
  - des noms de rôles (en plus du nom de l'association)
  - Le sens de lecture



Exemple d'utilisation des noms de rôles

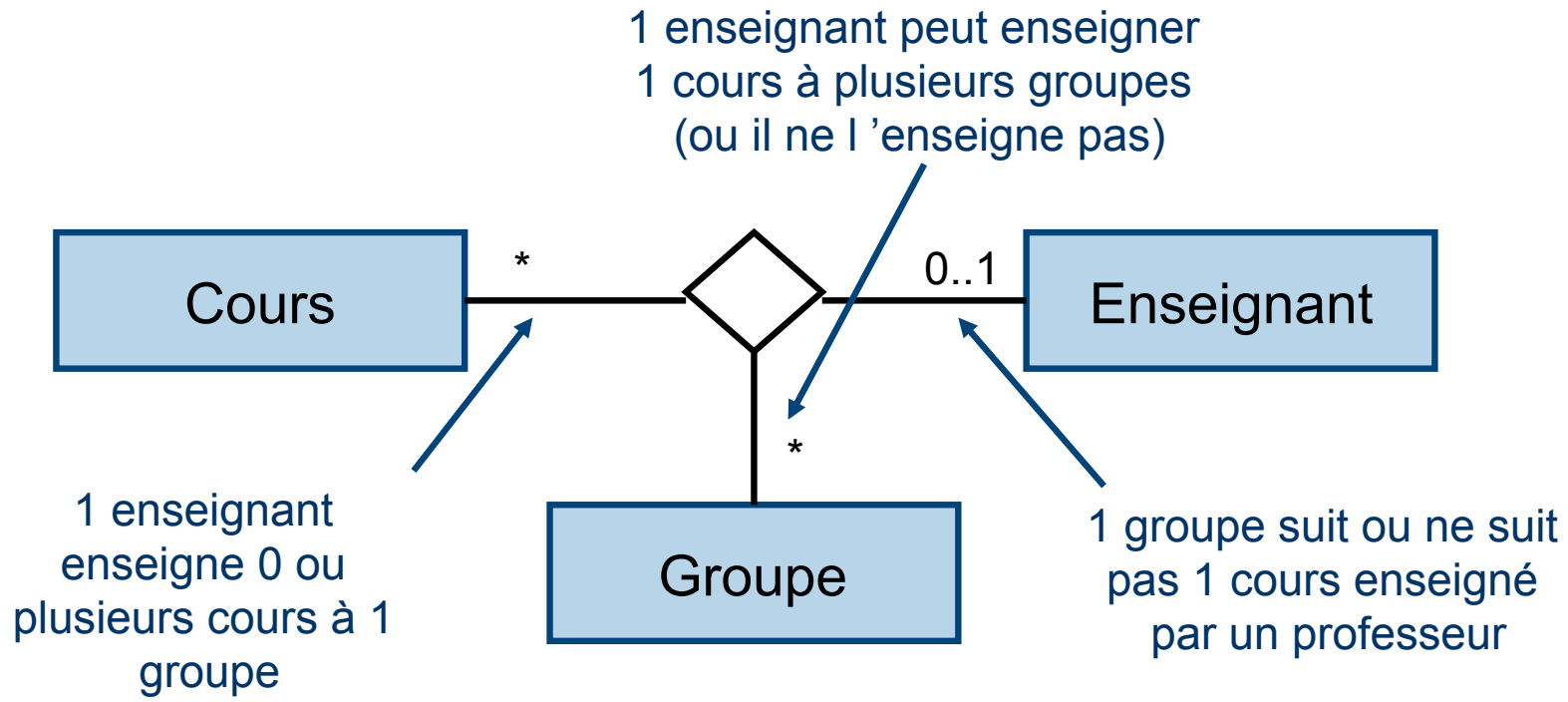


1) Dans toute société, il y a au moins une personne qui n 'a pas de responsable hiérarchique (le directeur de la société).
2) Une personne ne peut pas avoir plus d 'un responsable hiérarchique.
3) Une personne peut avoir zéro ou plusieurs collaborateurs.
4) Une personne ne peut pas être employée par plusieurs sociétés.
5) Une société emploie une ou plusieurs personnes.

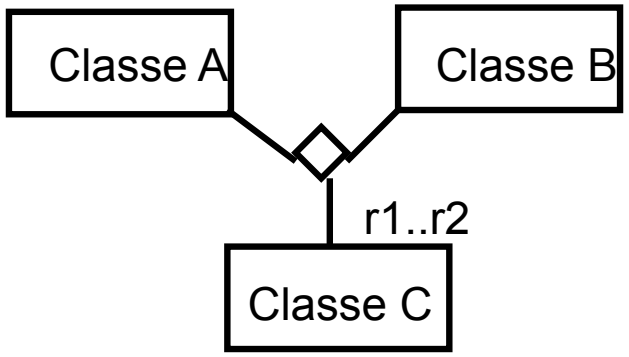
Les associations n-aires

- La plupart des associations sont dites binaires car elles relient deux classes
- Des associations d'arité supérieure peuvent néanmoins exister
- Beaucoup de langages de programmation ne permettent pas d'exprimer des associations n-aires, il faut alors les transformer en classes
  - Attention, la signification du modèle peut en être modifiée !

Notation d'une association ternaire



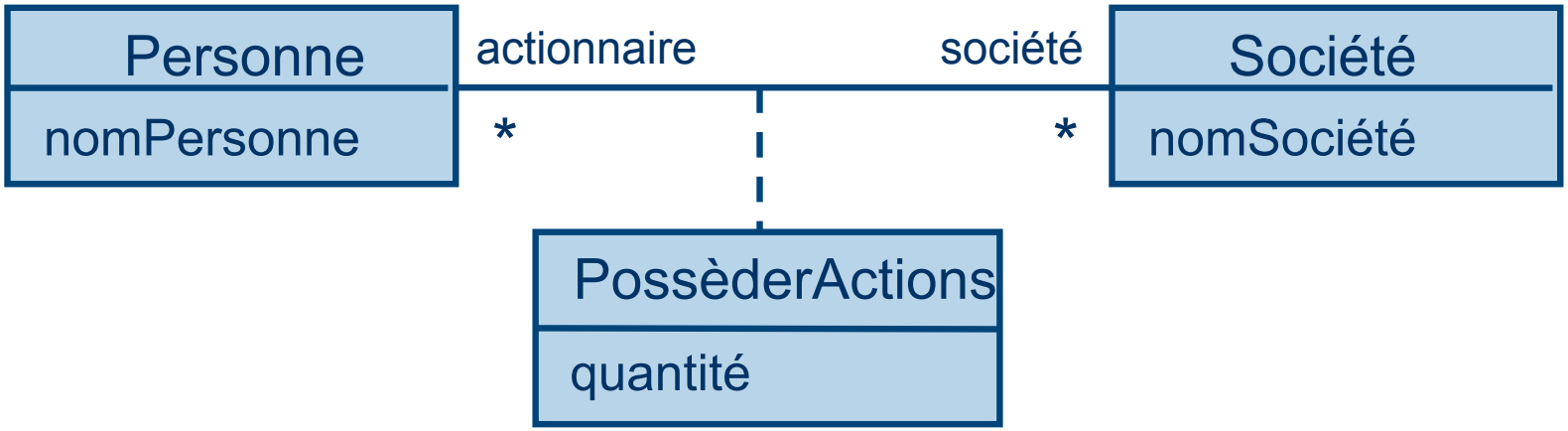
Pour un couple d 'instances de la classe A et de la classe B, il y a au minimum r1 instances de la classe C et au maximum r2 instances.



Les classes-associations

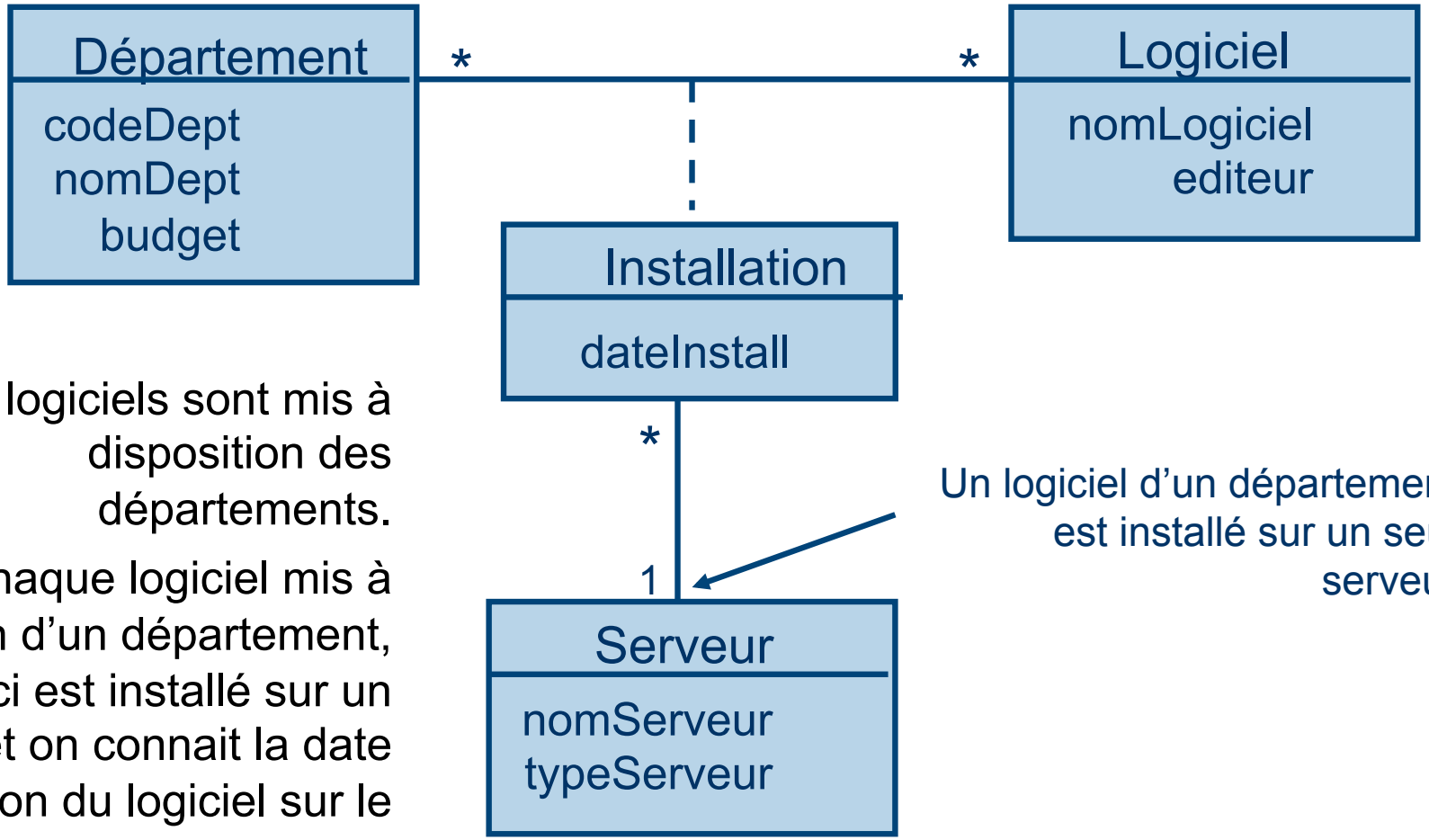
- Une association porteuse d'attributs est appelée une **classe-association**
- Une classe-association est :
  - une classe rattachée à une classe *plusieurs-à-plusieurs*
  - Possède éventuellement des attributs et des opérations
  - Est rattachée éventuellement à d'autres classes

Exemple de classe-association



- Une classe-association, en tant que classe, peut être associée à d'autres classes (voir à elle-même dans le cas d'une association réflexive).
- Une classe-association peut être transformée en classe

Exemple de classe-association participant à une association



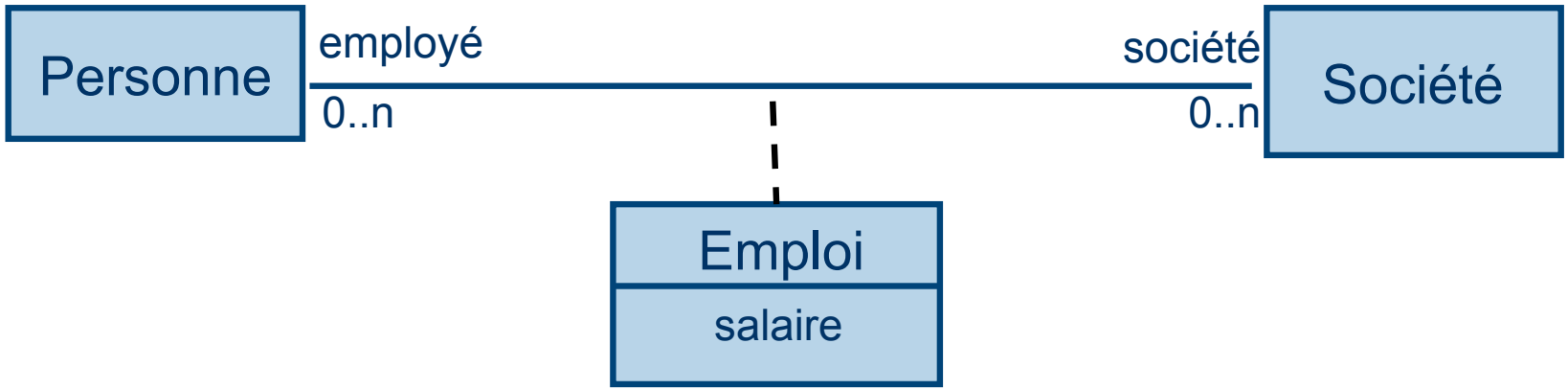
Des logiciels sont mis à disposition des départements.

Pour chaque logiciel mis à disposition d'un département, celui-ci est installé sur un serveur et on connait la date d'installation du logiciel sur le serveur.

Un logiciel d'un département est installé sur un seul serveur

ENSIAS

Transformation d'une classe-association en classe



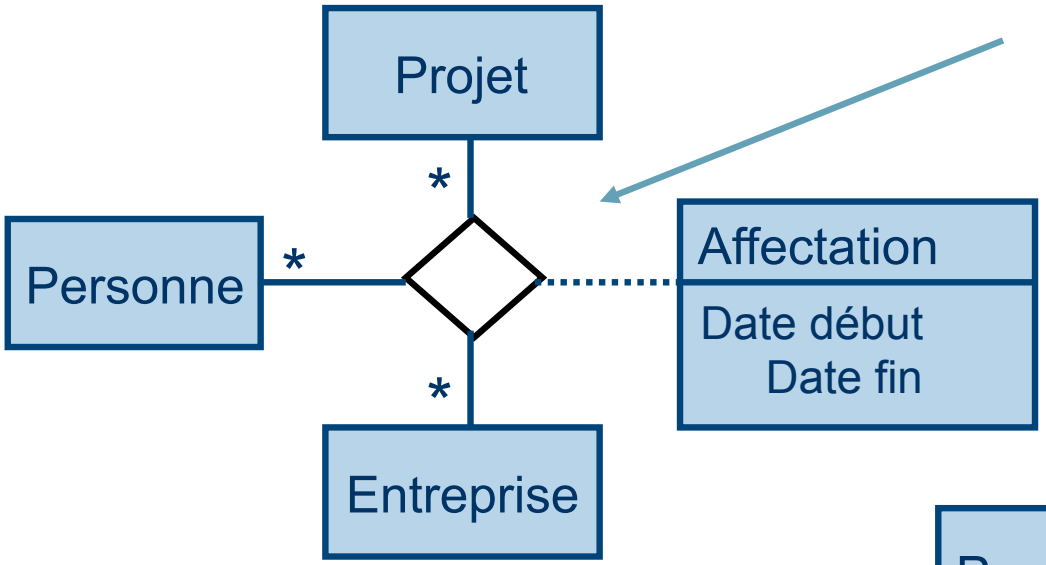
Ci-dessus, une personne ne peut pas avoir deux emplois dans une même société...

…alors que ci-dessus oui !

La transformation d'une classe-association en classe peut modifier la signification du modèle…

ENSIAS

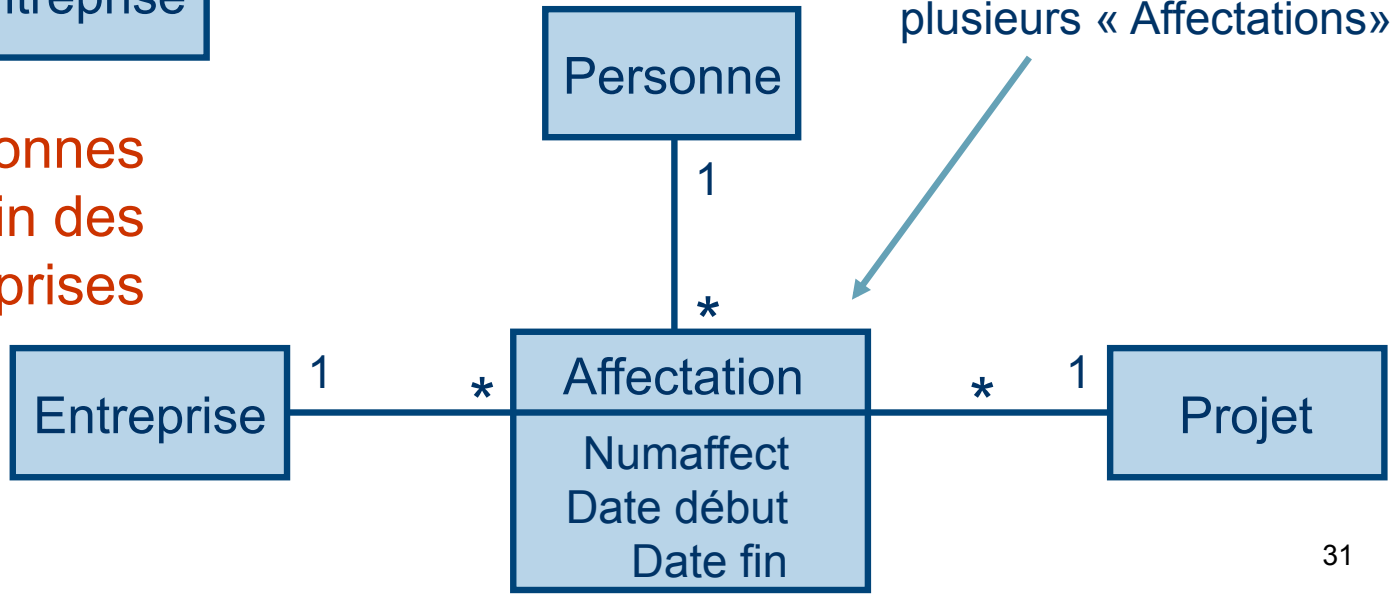
Autre exemple dans le cas d'une association ternaire



Pour une combinaison de Personne, Projet et Entreprise, une seule Affectation.

Pour chaque combinaison de Personne, Projet, Entreprise, il peut y avoir plusieurs « Affectations».

Affectation des personnes sur les projets au sein des entreprises



Navigabilité

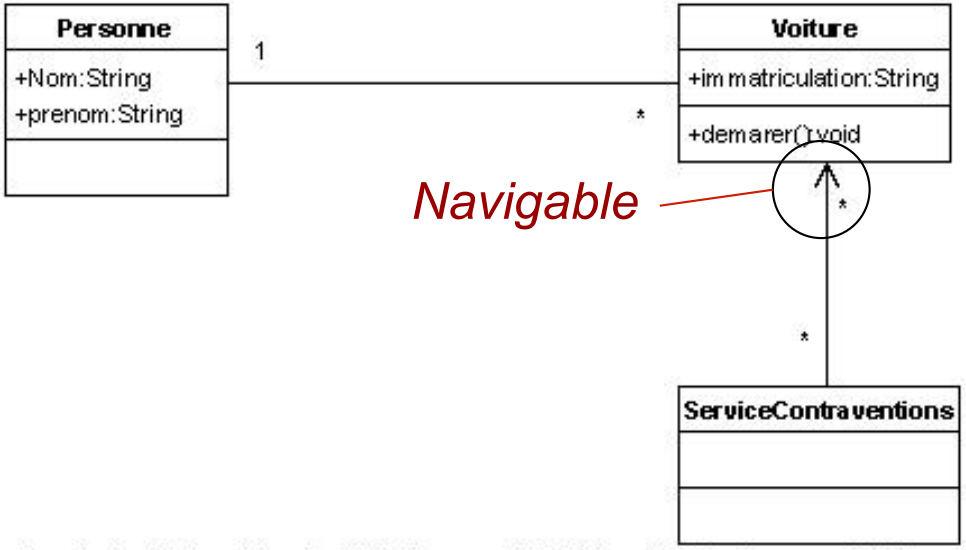
- Par défaut une association est navigable dans les deux sens

  - Chaque instance de voiture a un lien vers le propriétaire
  - Chaque instance de Personne a un ensemble de liens vers les voitures

- Restriction de la navigabilité
  - Le service de contravention

  est associé à une ou plusieurs voitures
  - La voiture ne connaît pas

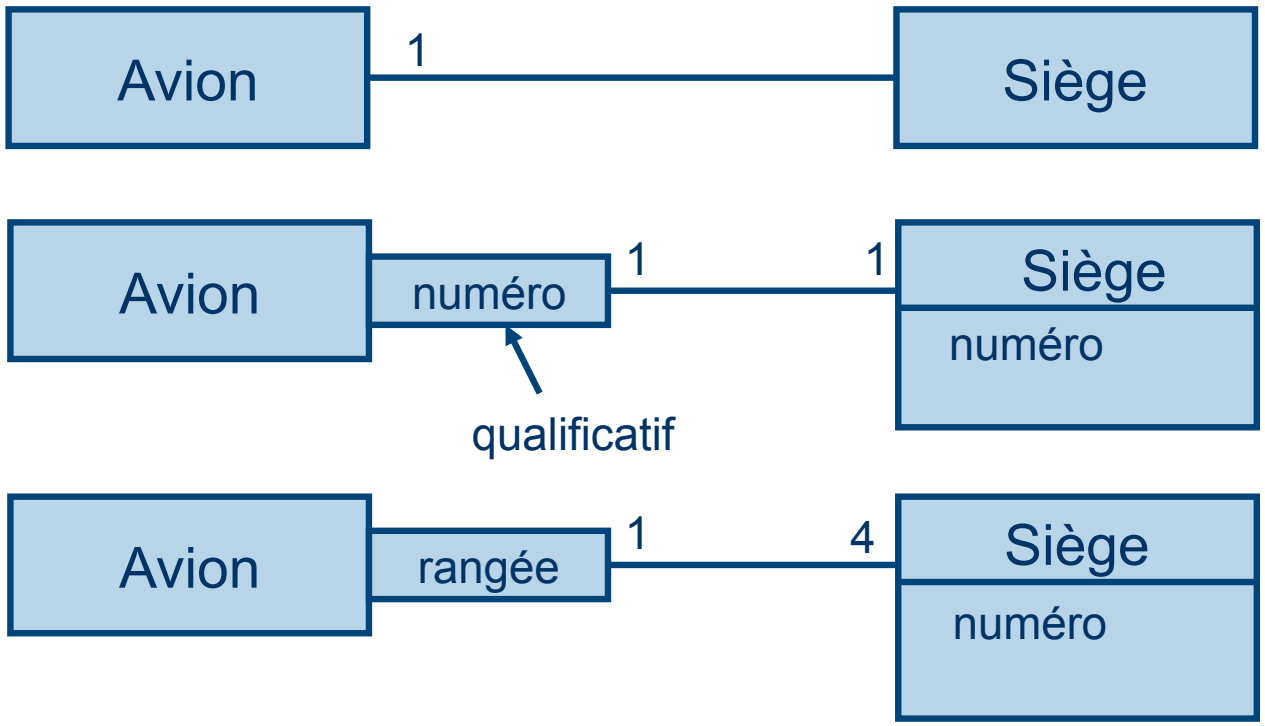
  le service de contraventions



ENSIAS

Les associations qualifiées

- Une association qualifiée est une association dans laquelle un attribut nommé qualificateur est utilisé pour lever l'ambiguïté sur les objets situés à l'extrémité de la multiplicité.



L'agrégation

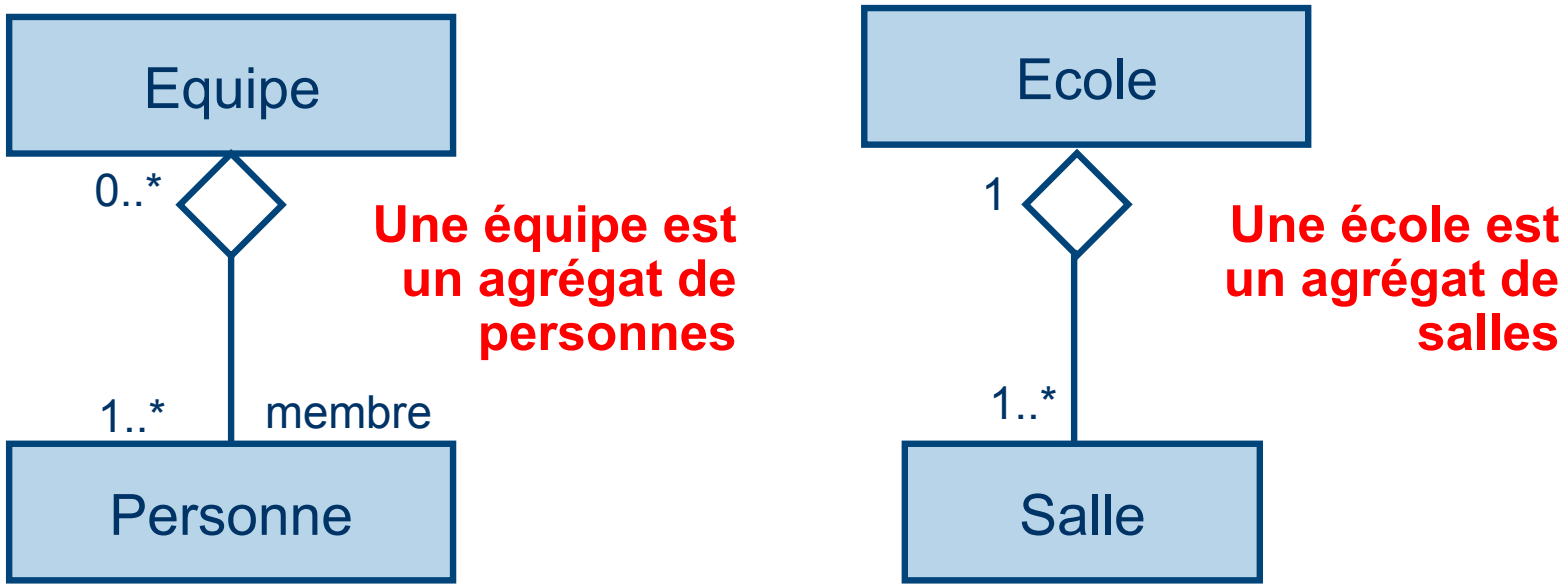
- **L'agrégation est un cas particulier d'association**

- L 'agrégation exprime une relation de contenance, de « partie à tout » entre des parties et un agrégat

- Une partie peut être partie de plusieurs agrégats

  ☞ **Le choix d'une association de type agrégation traduit la volonté de renforcer la dépendance entre les classes**

Exemples d'agrégation



✓ Une personne peut être membre de plusieurs équipes.
✓ La destruction d'une instance de la classe Equipe n'entraine pas la destruction des instances correspondantes de la classe Personne

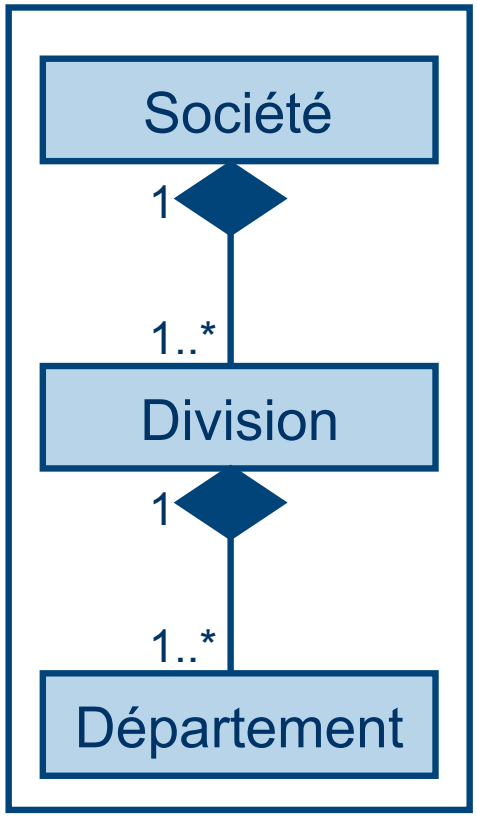
Agrégation vs association

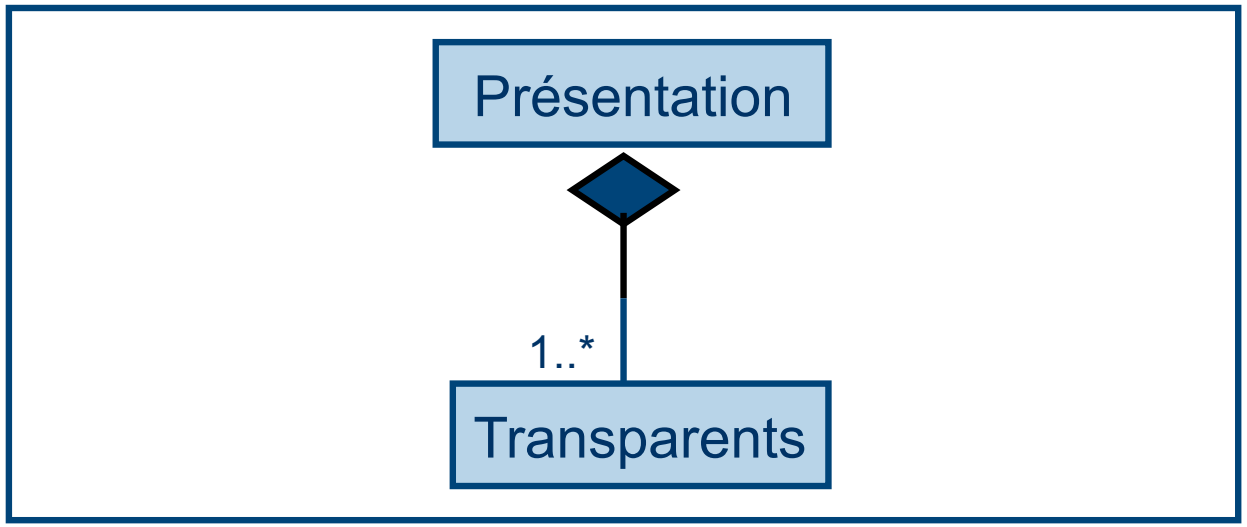
- **Propriétés de l'agrégation**
  - L'agrégation est transitive
  - L'agrégation est asymétrique
- **Agrégation vs association**
  - Si deux objets sont étroitement liés par une relation constitué-constituant, il s'agit d'une agrégation
    - Peut-on utiliser l'expression « fait partie de » ?
    - Ex : un Véhicule fait partie d'un TypeDeVéhicule
  - Si ces deux objets sont habituellement considérés comme indépendants, même s'ils sont souvent liés, il s'agit d'une association.
    - Ex : un Employé est salarié d'une Entreprise → association

La composition

- **La composition : cas particulier d'agrégation**
- La composition exprime une relation plus forte que l'agrégation : les parties (alors appelées *les composants)* ne peuvent être liées qu'à un seul agrégat (alors appelé *composite*)
- La vie des composants est liée à celle de l'agrégat
  - la suppression d'un objet agrégat entraîne la suppression des objets agrégés
    - ☞ La composition fait souvent référence à une contenance physique

Exemples de composition

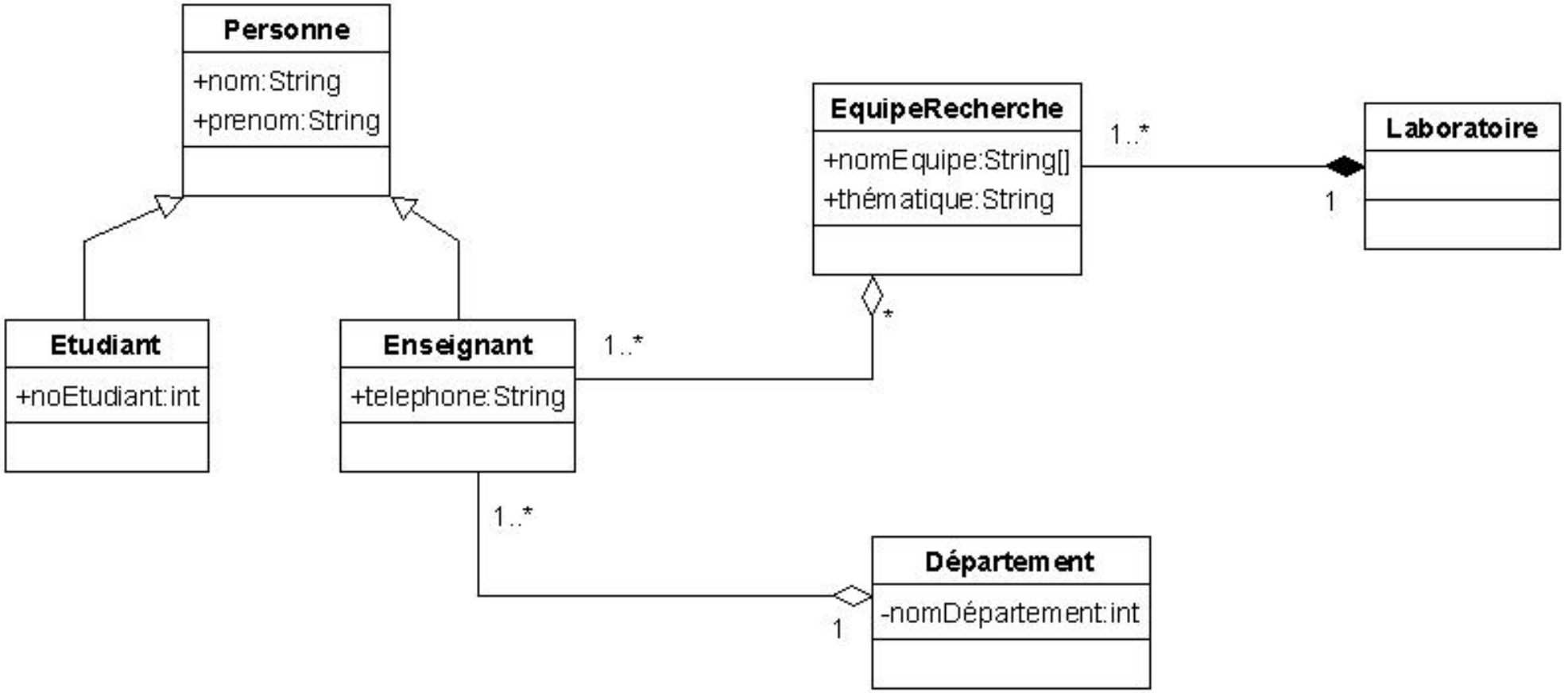




- ✓ La suppression de la société entraine la disparition des divisions qui la composent
- ✓ La suppression d'une Présentation entraîne la disparition des transparents

ENSIAS

Exemple : Agrégation/Composition



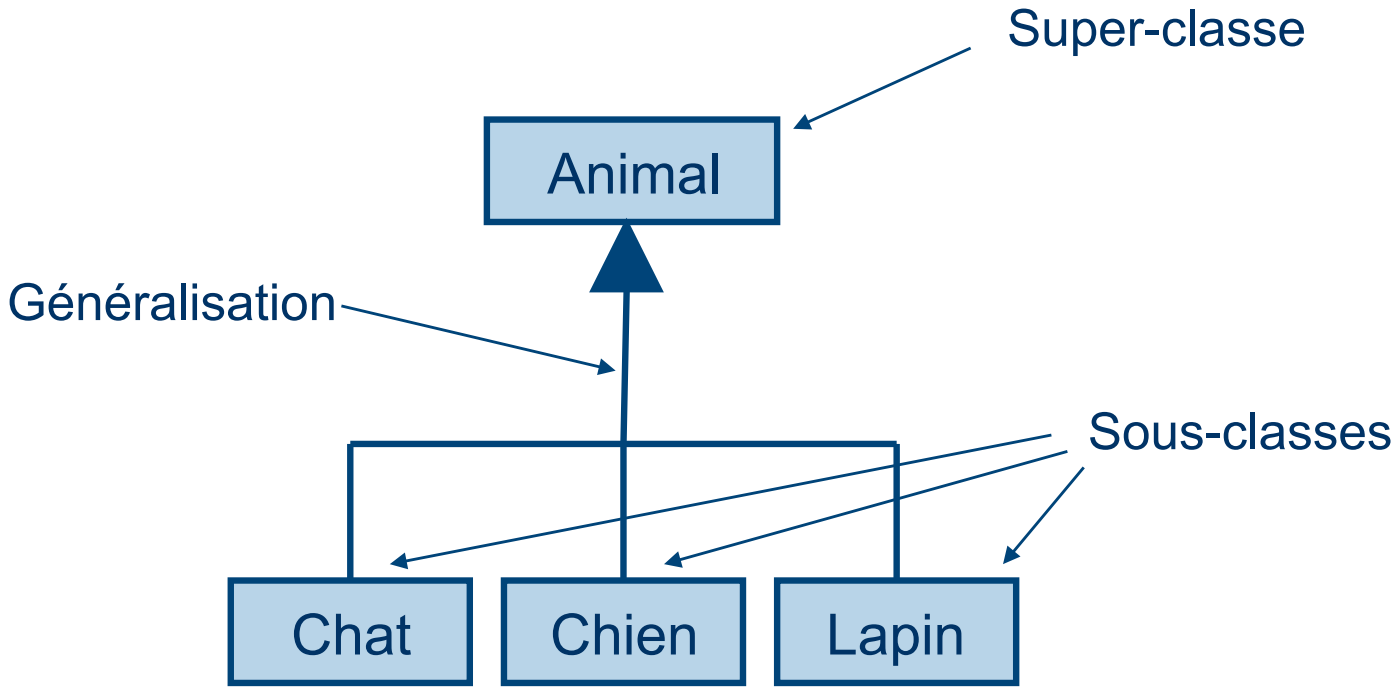
La généralisation/spécialisation

- **La généralisation/spécialisation : relation définie entre deux classes, mais ce n'est pas une association**
- La généralisation (simple) exprime une relation de « généralisation/spécialisation » entre une *superclasse* et une ou plusieurs *sous-classes* plus spécialisées.
- La généralisation exprime une relation d'inclusion entre une classe et sa super-classe.
  - Chaque instance de la classe est également instance de la super-classe.

Principes de l'héritage

- **Une sous-classe hérite de sa super-classe** (ou d'une hiérarchie de super-classes) la description des instances:
  - attributs, opérations, associations, contraintes définies sur la super-classe…
- **Une sous-classe peut redéfinir de façon plus spécialisée une partie ou la totalité de la description héritée** (voir ci-après le concept de redéfinition), mais il n 'y a pas d'exception à l'héritage

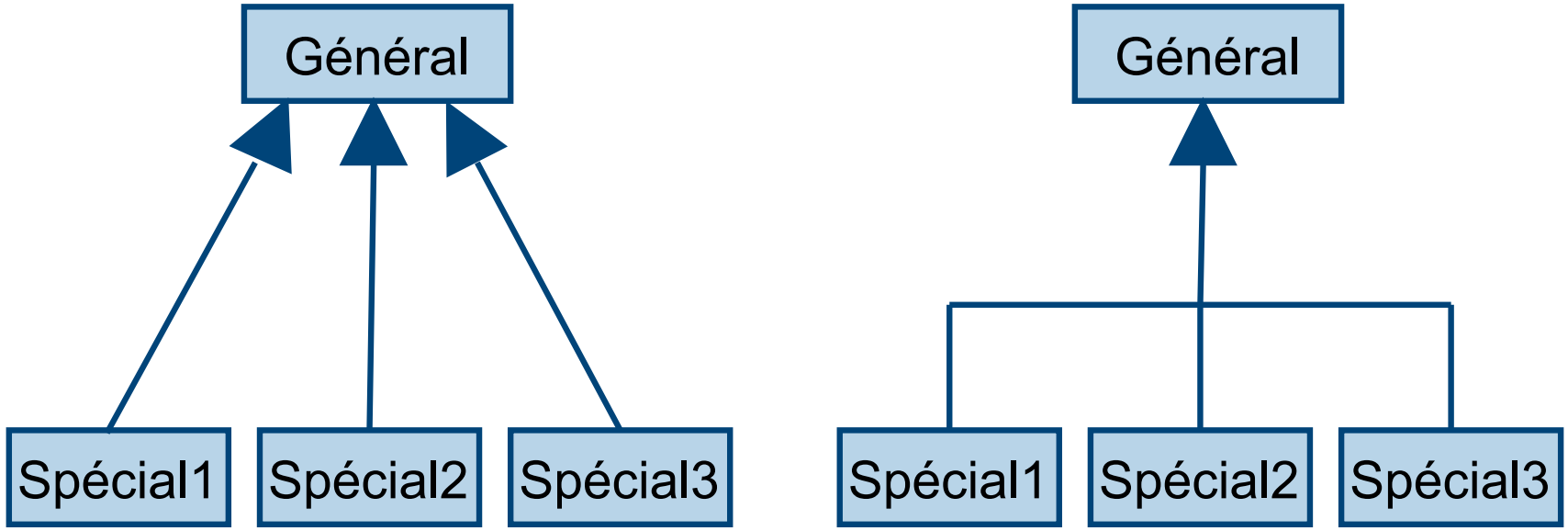
Illustration



Animal est une généralisation de Chat, Chien et Lapin.
Chat, Chien et Lapin sont des animaux.

Notations

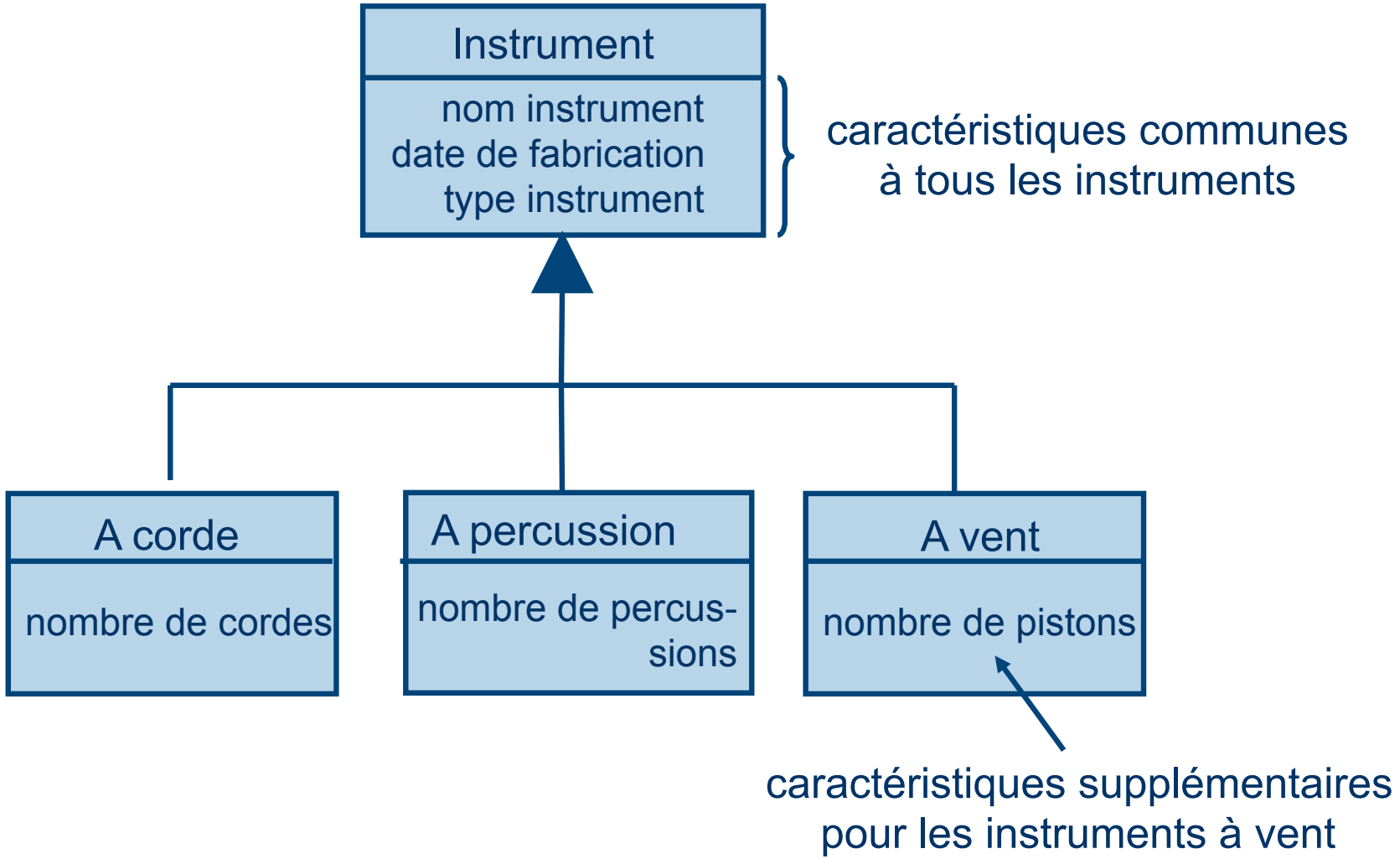
- Deux styles de notation :



Utilité

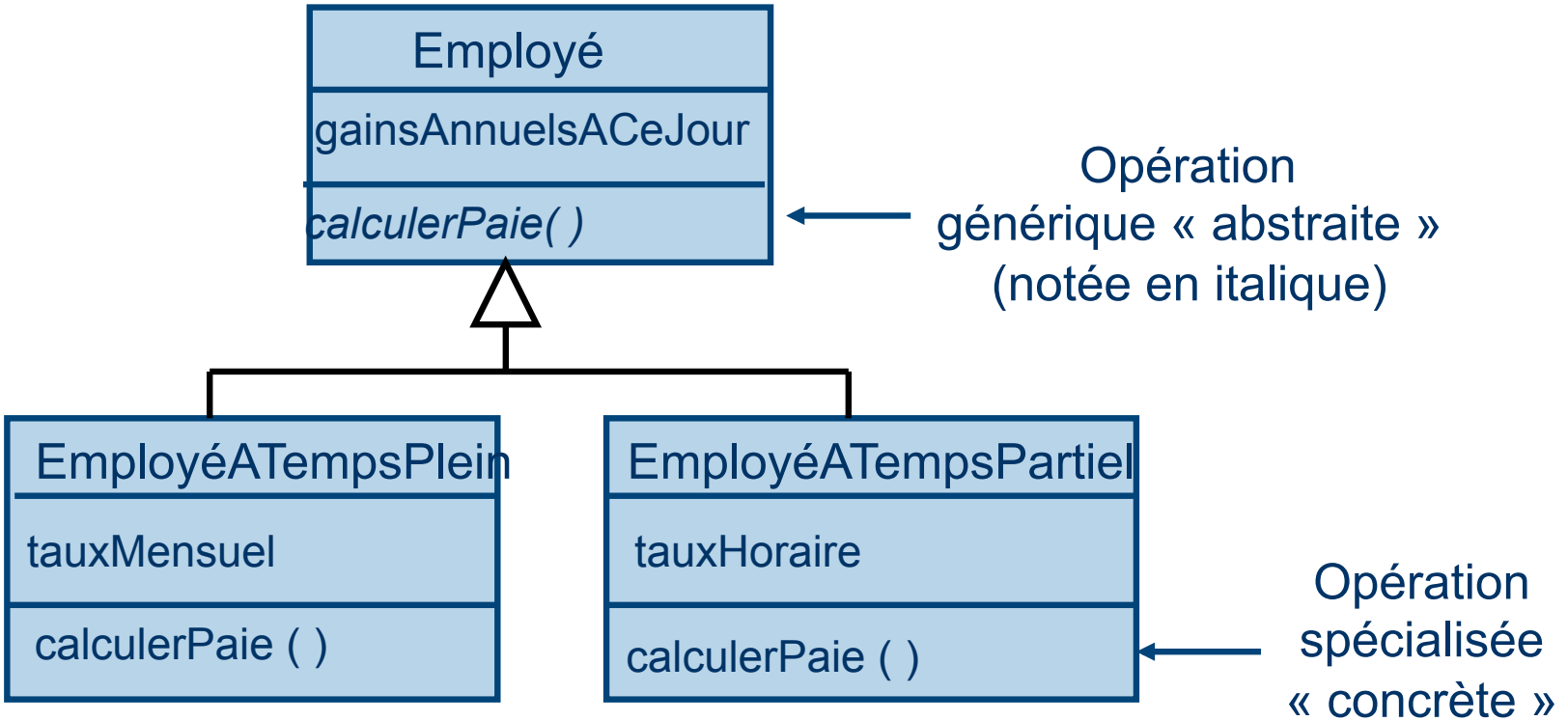
- **Dans le sens « généralisation :**
  - Permet d'abstraire en factorisant les propriétés communes aux sous-classes
- **Dans le sens spécialisation :**
  - Permet de **réutiliser** par modification incrémentielle les descriptions existantes.
  - Deux formes de spécialisation :
    - spécialisation par **enrichissement** (ajout de propriétés)
    - spécialisation par **redéfinition** (modification de propriétés)

Exemple de spécialisation par enrichissement



Exemple de spécialisation par redéfinition

- Une classe spécialisée peut redéfinir une propriété, à condition toutefois de rester compatible avec la définition originale (polymorphisme).



Autre exemple…

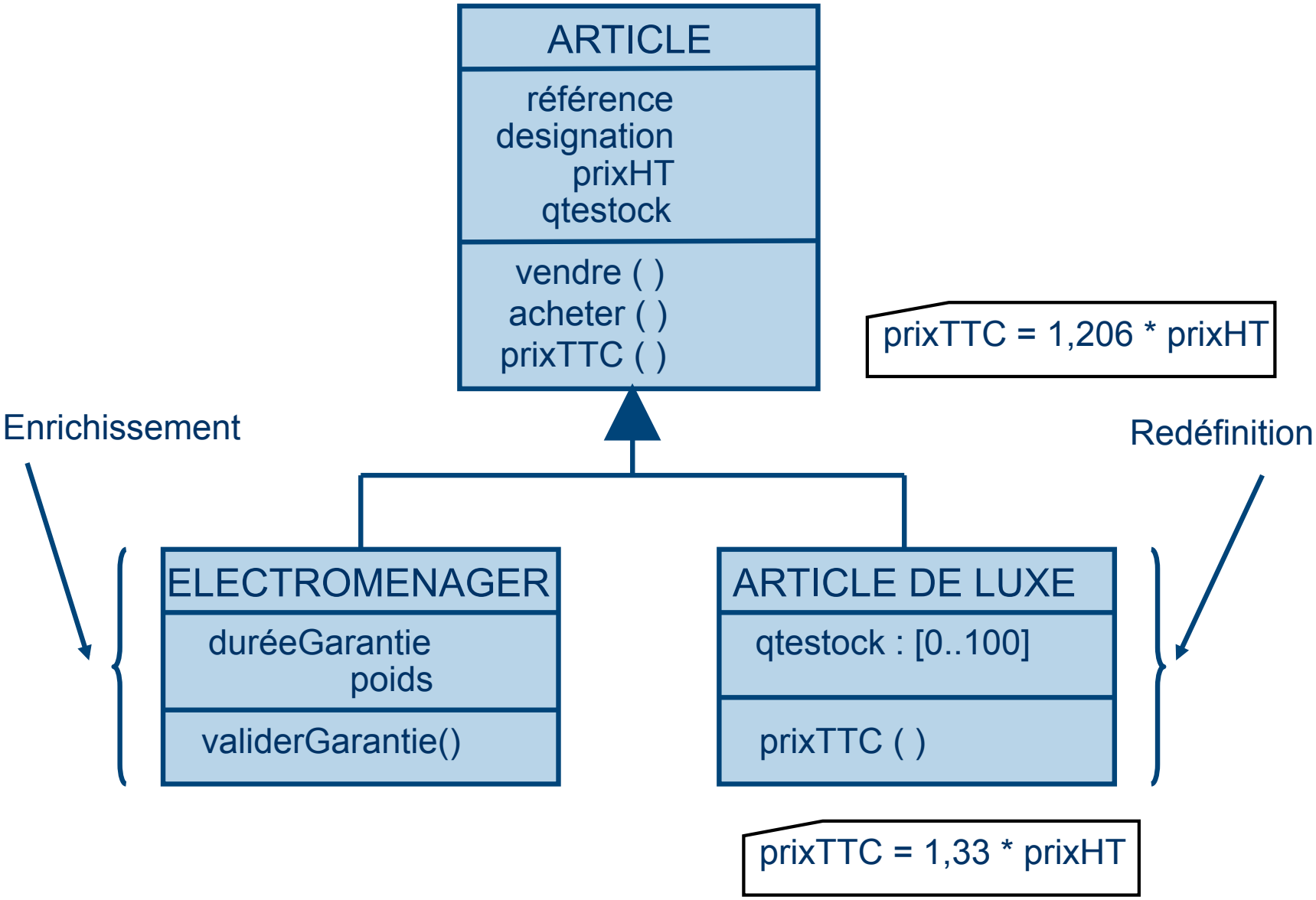


Diagramme de classes : exemple

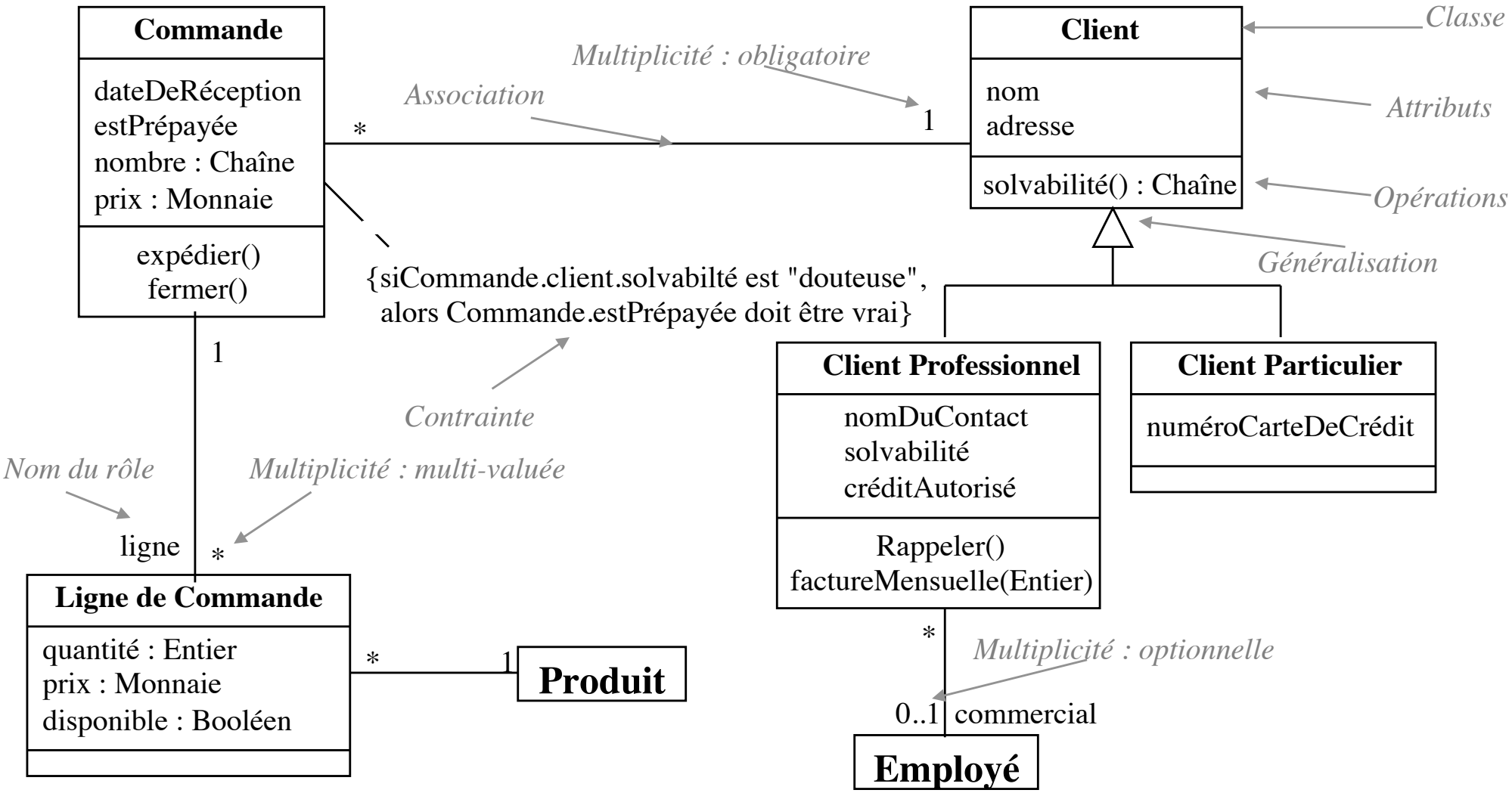


Diagramme de classes et diagrammes d'objets

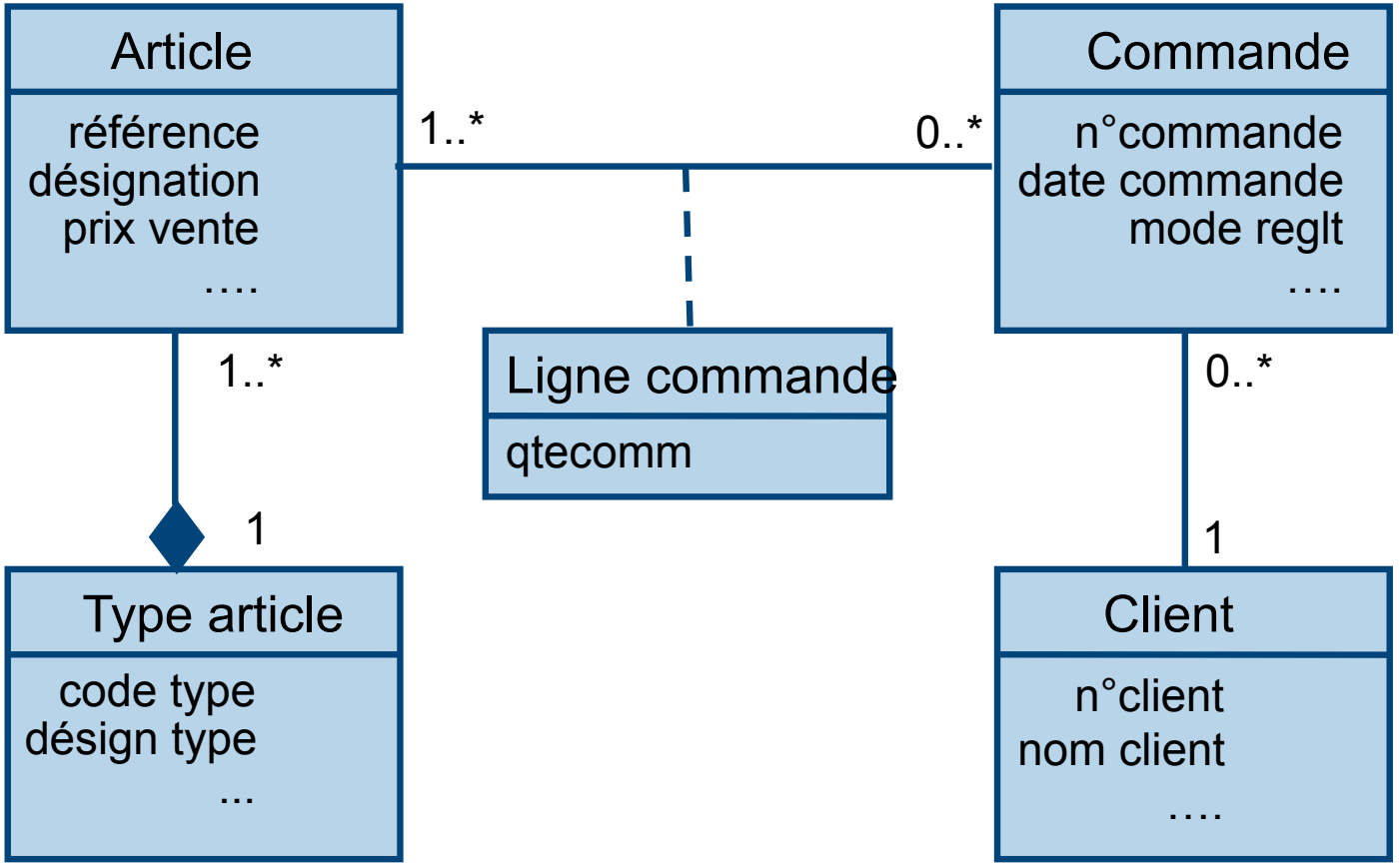


Diagramme de classes

Diagramme de classes et diagrammes d'objets

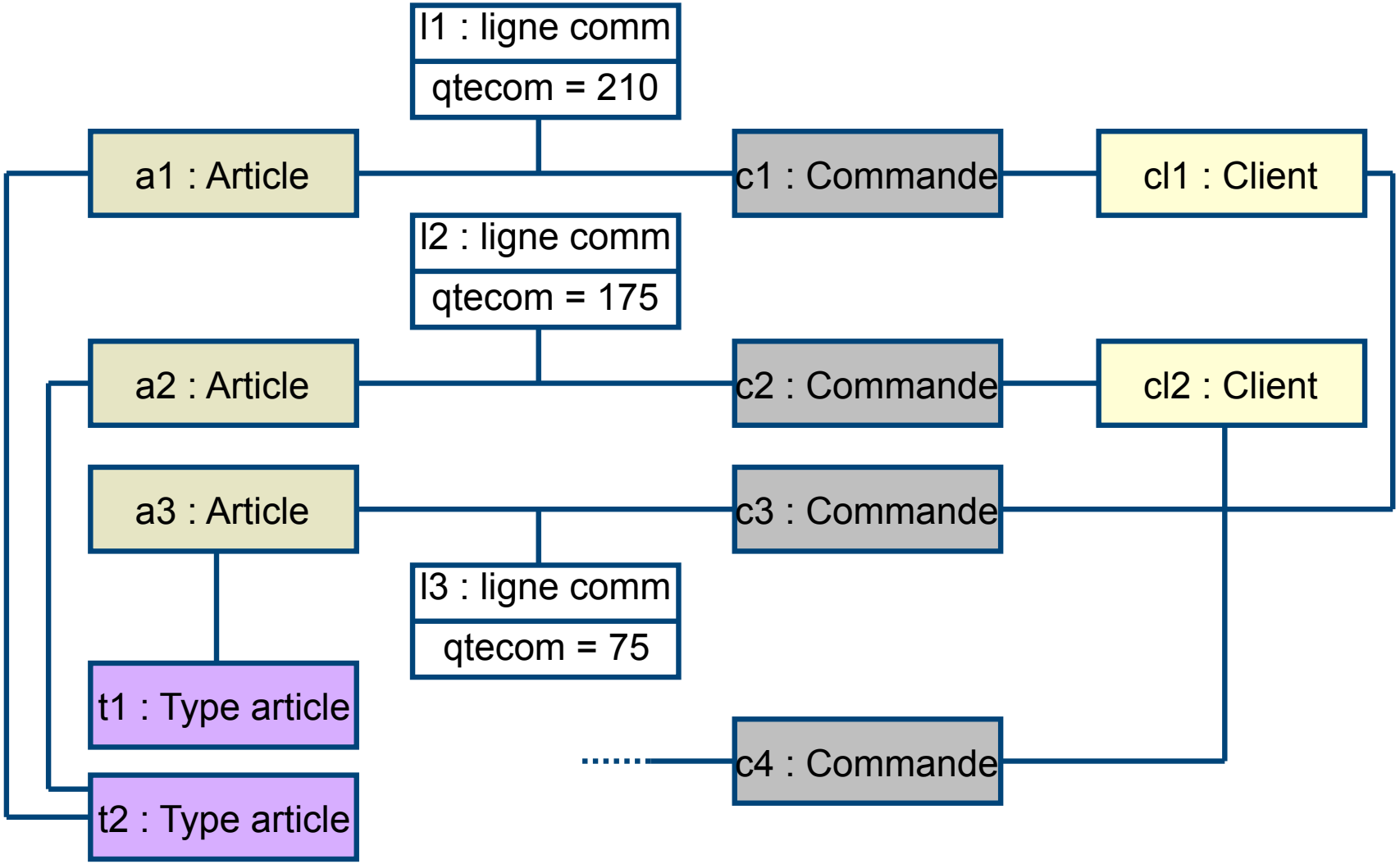


Diagramme d 'objets

Diagramme de classes et diagrammes d ’objets

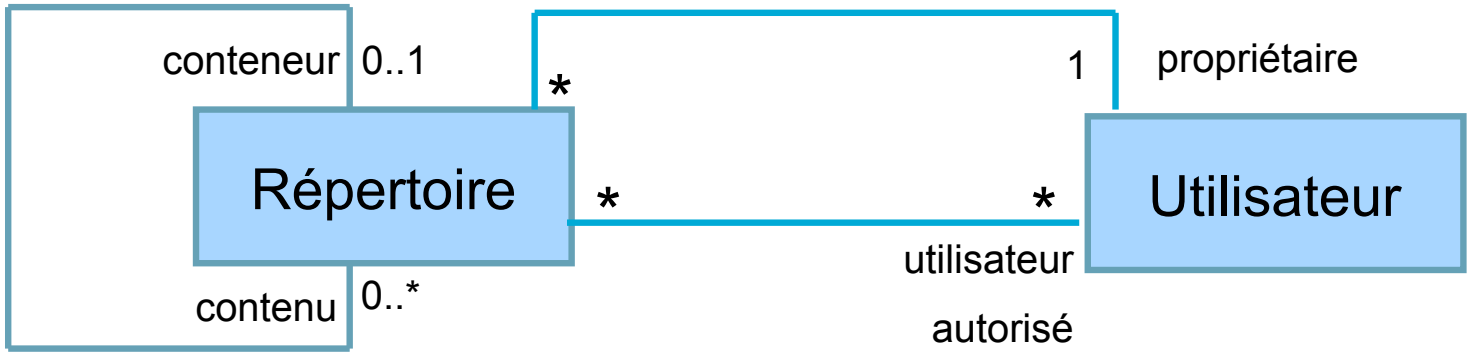
- **Un diagramme de classes**
  - définit l’ensemble de tous les états possibles
  - définit les contraintes qui doivent toujours être vérifiées
- **Un diagramme d’objets**
  - décrit un état possible à un instant t (un état particulier)
  - doit être conforme au modèle de classes
- **Les diagrammes d’objets peuvent être utilisés pour :**
  - expliquer un diagramme de classes (donner un exemple)
  - valider un diagramme de classes

Mise en application

- **Déterminer la relation statique (généralisation, composition, agrégation ou association) dans chacune des phrases suivantes :**
  - Un répertoire contient des fichiers
  - Tout écrivain a écrit au moins une oeuvre
  - Les modems et les claviers sont des périphériques d'E/S
  - Une transaction boursière est un achat ou une vente
  - Un compte bancaire peut appartenir à une personne physique ou morale
  - Les personnes peuvent être associées à des universités en tant qu'étudiants aussi bien qu'en tant que professeurs.
  - Les cinémas sont composés de plusieurs salles. Les films sont projetés dans des salles. Les projections correspondantes ont lieu à chacune à une heure déterminée

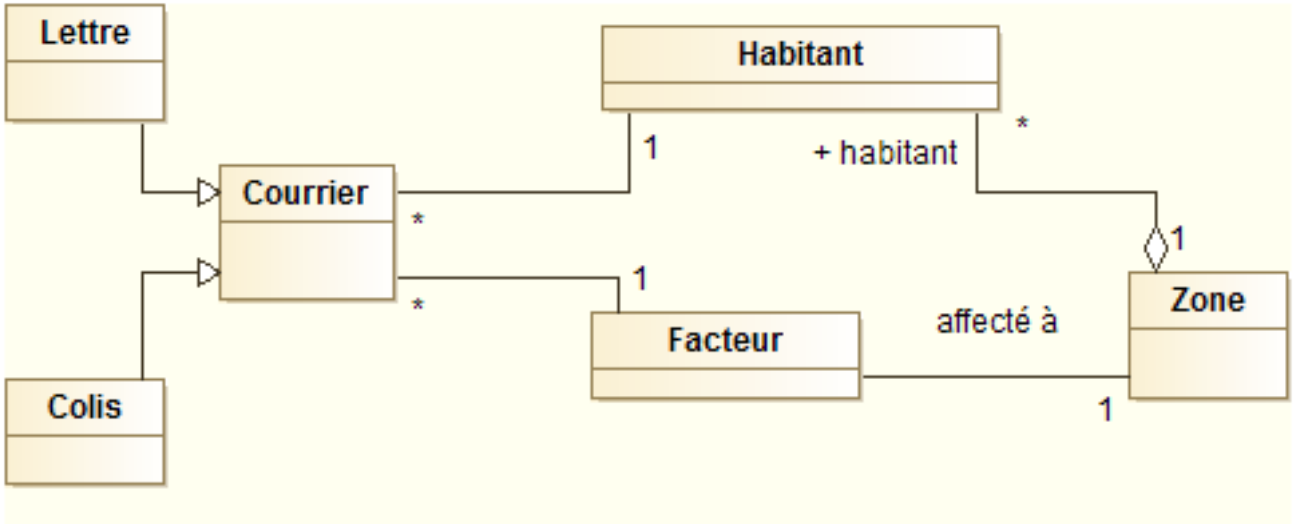
Mise en application

- Représenter sous la forme d'un modèle de classes la situation suivante :
  - Une répertoire peut contenir plusieurs sous répertoires et être éventuellement lui-même être contenu dans un autre répertoire.
  - Chaque répertoire a exactement un utilisateur qui est son propriétaire et plusieurs utilisateurs qui sont autorisés



Mise en application

- Représenter sous la forme d'un modèle de classes la situation suivante :
- Tous les jours, le facteur distribue du courrier aux habitants de sa zone d'affectation. Les habitants sont aussi associés à une zone géographique. Les courriers sont de deux sortes : lettres ou colis.



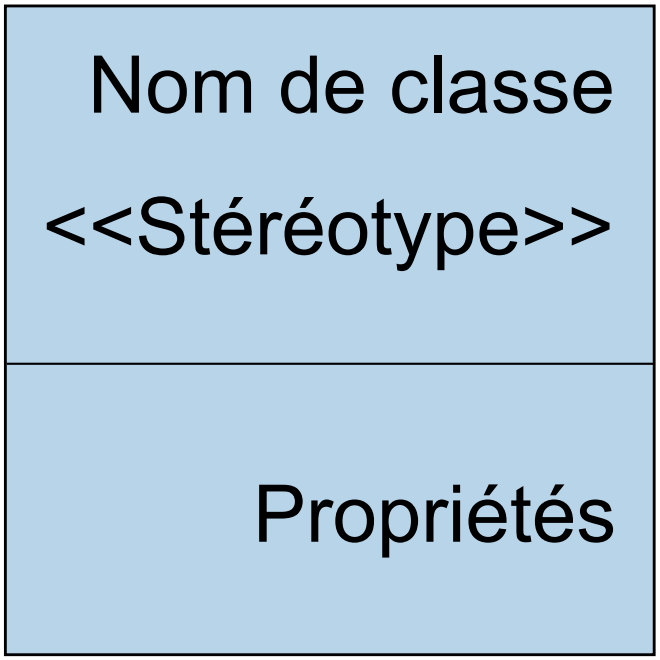
Concepts avancés du diagramme de classes

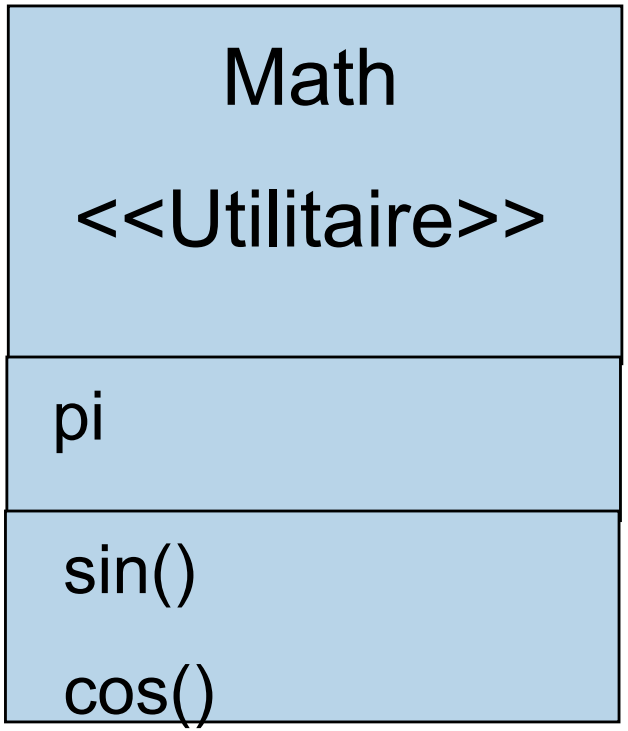
- Stéréotypes
- Attributs dérivés
- Notion de classe abstraite et d'opération abstraite
- Spécialisation multiple et héritage multiple
- Expression des contraintes
- Paquetages

Concepts avancés : les stéréotypes

- Le rectangle de classe dans UML : peut contenir un stéréotype et des propriétés
- Les stéréotypes :
  - permettent d'étendre la sémantique des éléments de modélisation : mécanisme d'extensibilité d'UML
  - permettent de définir de nouvelles classes d'éléments de modélisation, en plus du noyau prédéfini par UML.
  - UML définit différents stéréotypes de classes suivants : <<interface>>, <<métaclasse>> , <<utilitaire>>, …

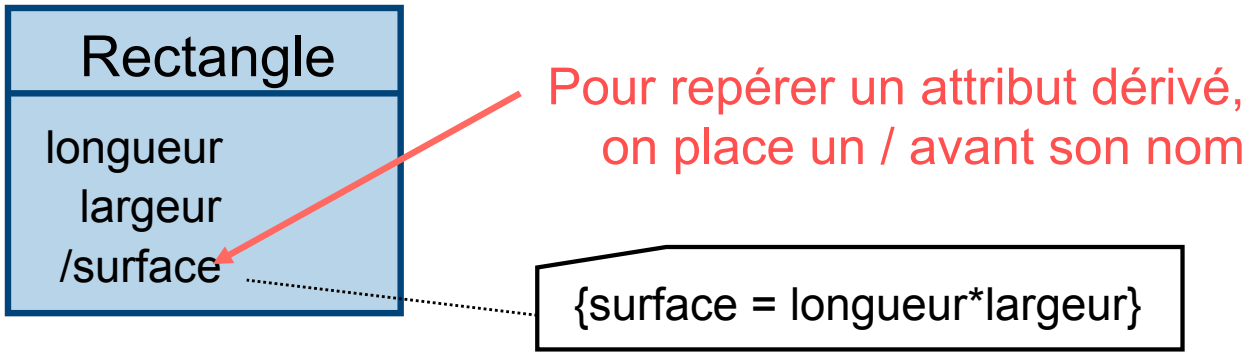
Exemples de classes stéréotypées





Concepts avancés : les attributs dérivés

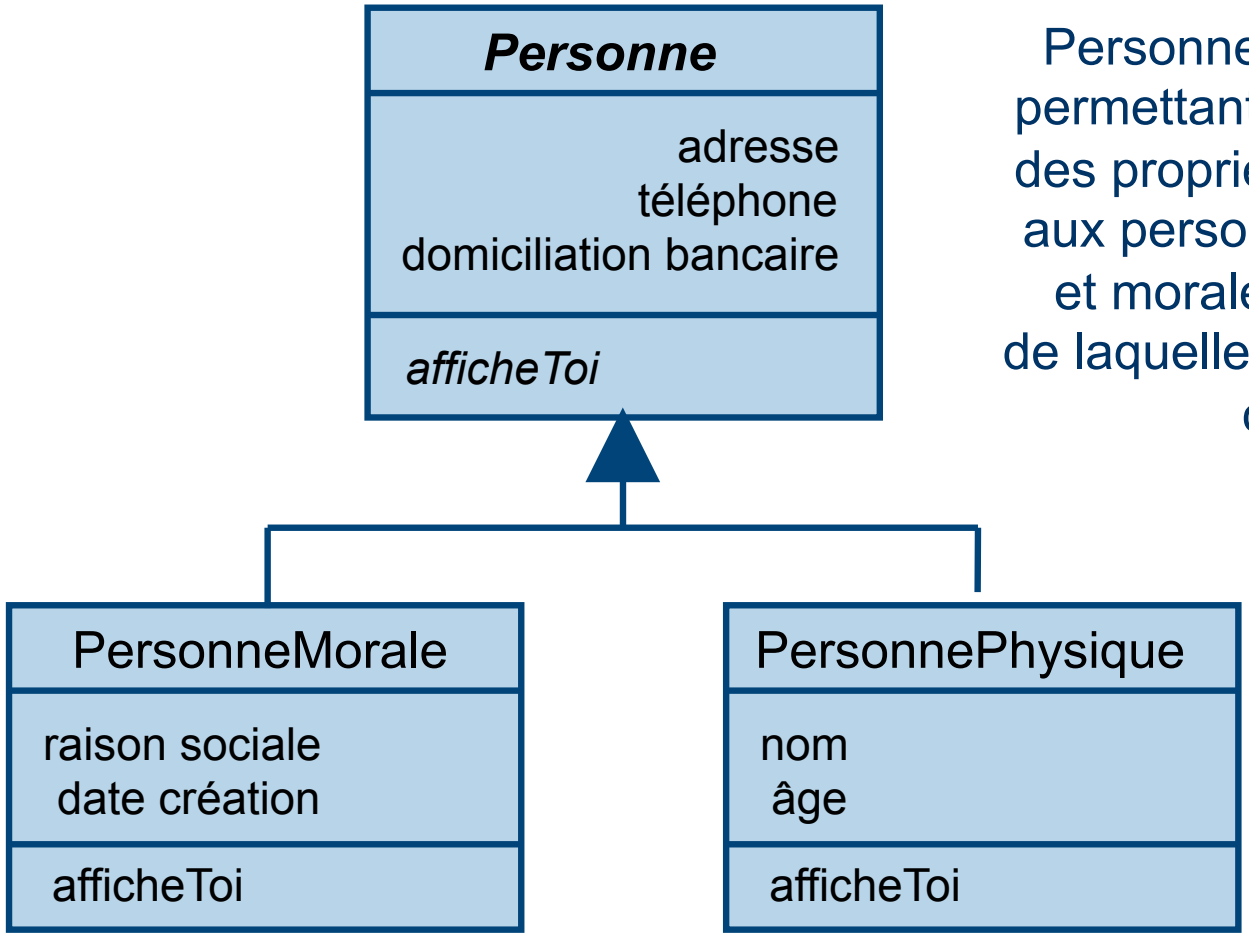
- En général, un attribut d'une classe est une information primaire pour cette dernière
- Parfois, par commodité de gestion, on choisit de conserver dans un attribut le résultat d'un calcul effectué à partir d'autres attributs de la classe, on parle alors d'attribut dérivé



Concepts avancés : la classe abstraite

- **Une classe abstraite est une classe non instanciable, càd qu'elle n'admet pas d'instances directes**
- Une classe abstraite est une description d'objets destinée à être héritée par des classes plus spécialisées
- Pour être utile, une classe abstraite doit admettre des sous-classes concrètes
- La factorisation optimale des propriétés communes à plusieurs classes par généralisation nécessite le plus souvent l'utilisation de classes abstraites

Classe abstraite : Personne

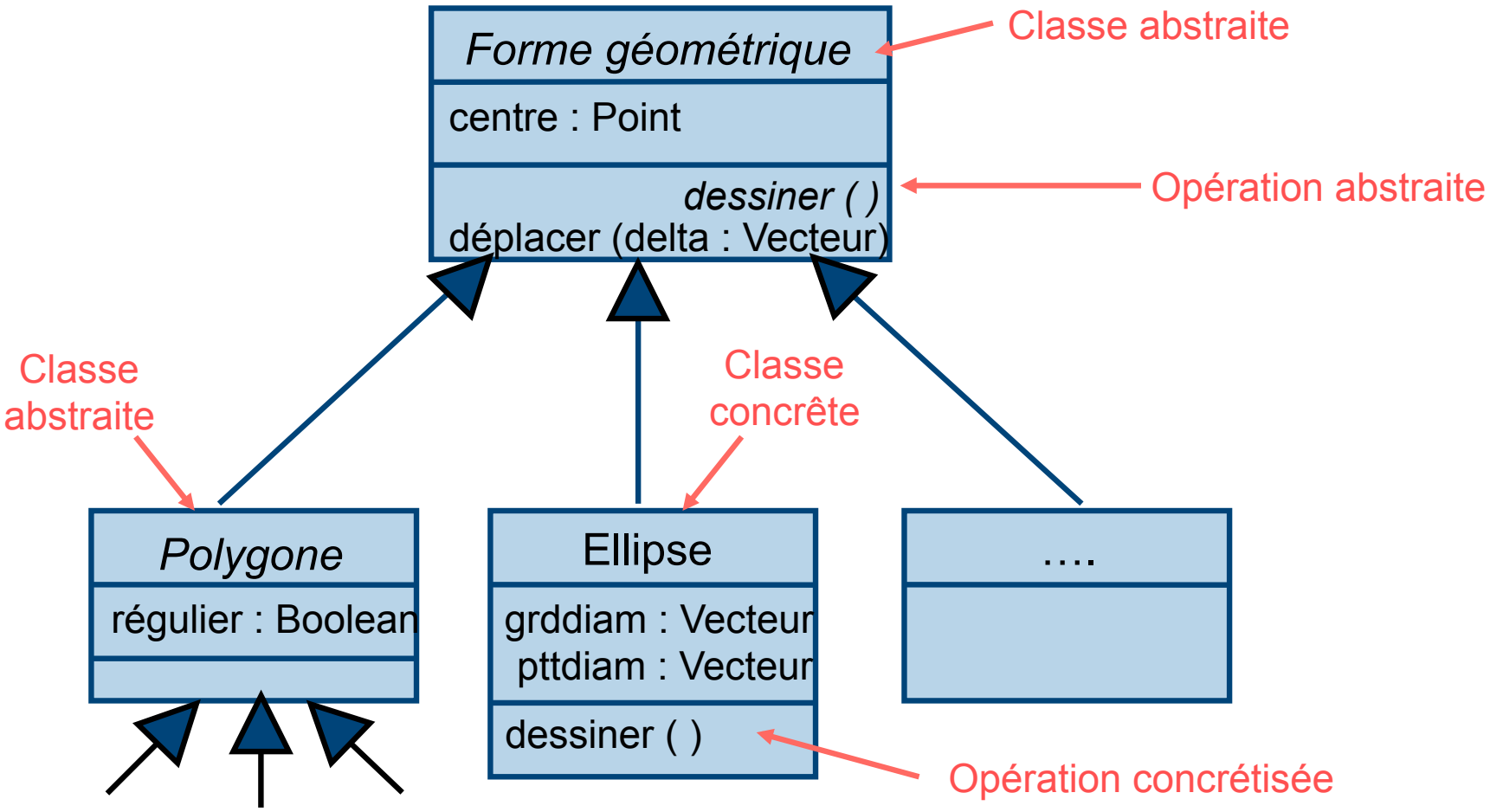


Personne est une classe permettant la factorisation des propriétés communes aux personnes physiques et morales, mais à partir de laquelle on ne peut pas créer d'instance

Opération abstraite

- **Une opération abstraite est une opération n'admettant pas d'implémentation :**
  - au niveau de la classe dans laquelle elle est déclarée, on ne peut pas dire comment la réaliser
- Les opérations abstraites sont particulièrement utiles pour mettre en œuvre le polymorphisme (utilisation du principe de redéfinition)
- Une classe pour laquelle au moins une opération abstraite est déclarée est une classe abstraite (l'inverse n'est pas vrai).
- Toute classe concrète sous-classe d'une classe abstraite doit « concrétiser » toutes les opérations abstraites de cette dernière
- Interface
  - Classe abstraite pure qui ne comporte que des méthodes abstraites

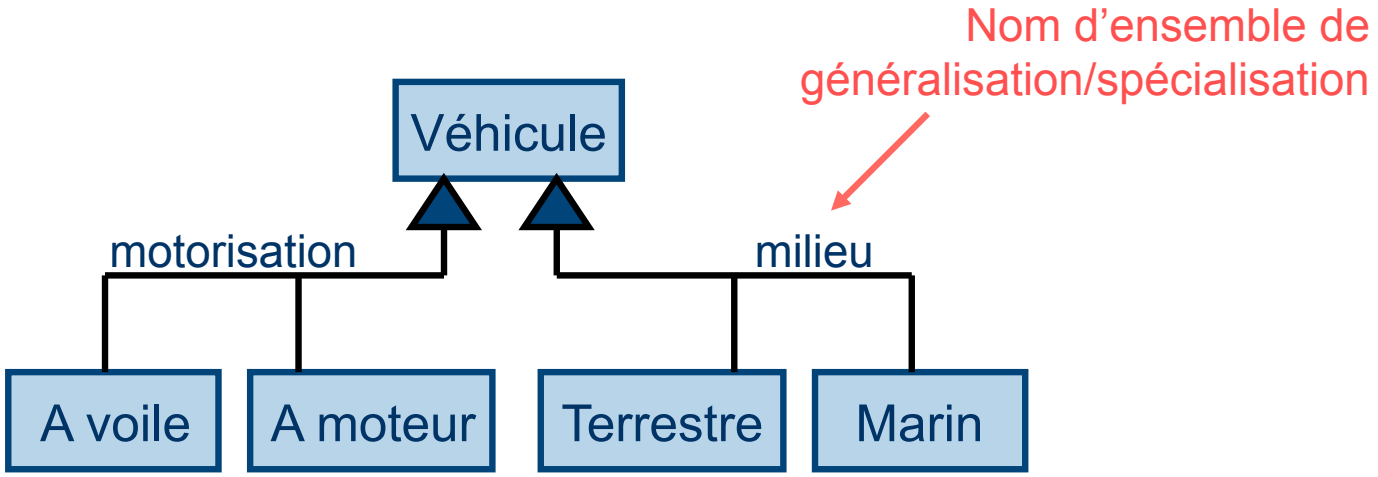
Les classes et opérations abstraites permettent de mieux structurer...



La classe abstraite « Polygone »
n'est utile que si elle est spécialisée !

Concepts avancés : la spécialisation multiple

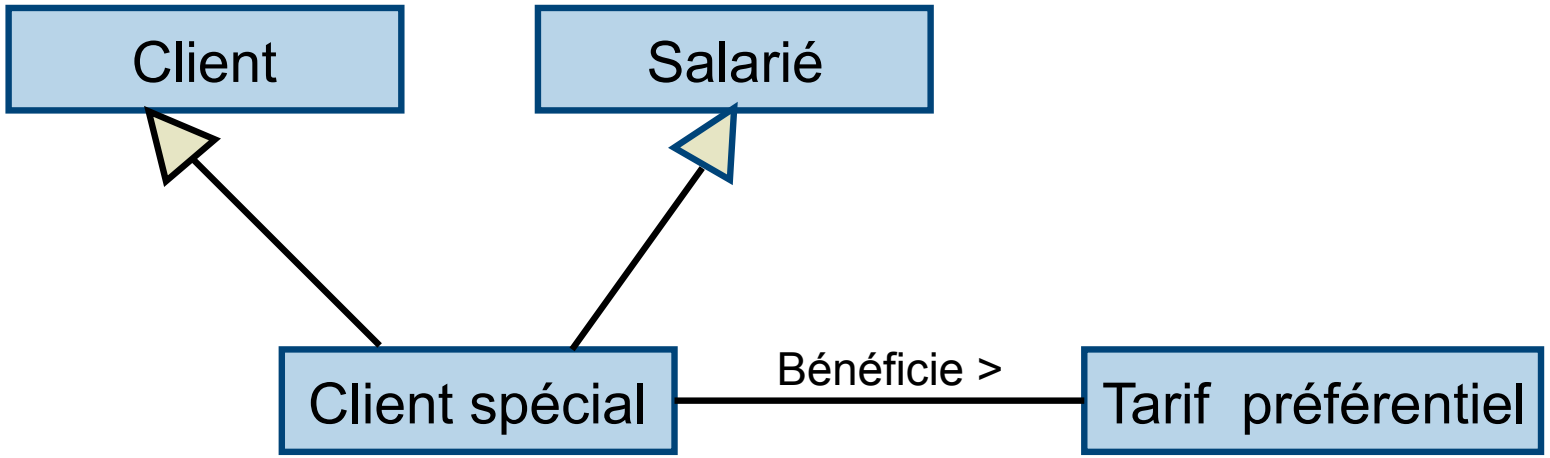
- Toute spécialisation ne doit concerner qu'un seul aspect
- On utilise la spécialisation multiple si une classe peut être spécialisée selon plusieurs aspects distincts et indépendants



L’héritage multiple

- **Dans le cas de l’héritage multiple, une classe peut être sous-classe de plusieurs superclasses (et hériter des propriétés de tous ses parents)**
- L’héritage multiple pose certains problèmes spécifiques tels que des conflits d’héritage
- Un mécanisme d’héritage simple est intégré dans tous les langages de programmation par objets, seuls quelques-uns intègrent un mécanisme d’héritage multiple

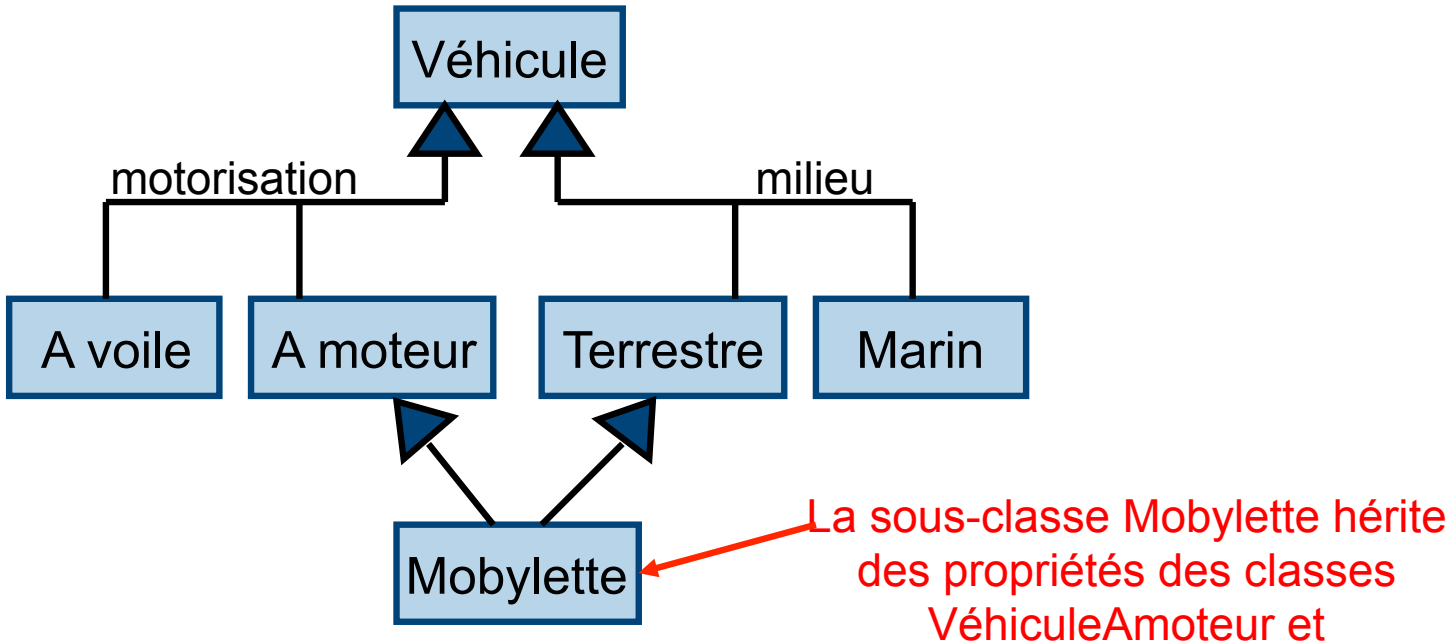
Exemple d'héritage multiple



La classe « Client spécial » est une spécialisation de « Client » et de « Salarié ».

☞ **Le modèle ci-dessus permet d'indiquer que l'on accorde des tarifs préférentiels aux salariés**

Autre exemple…

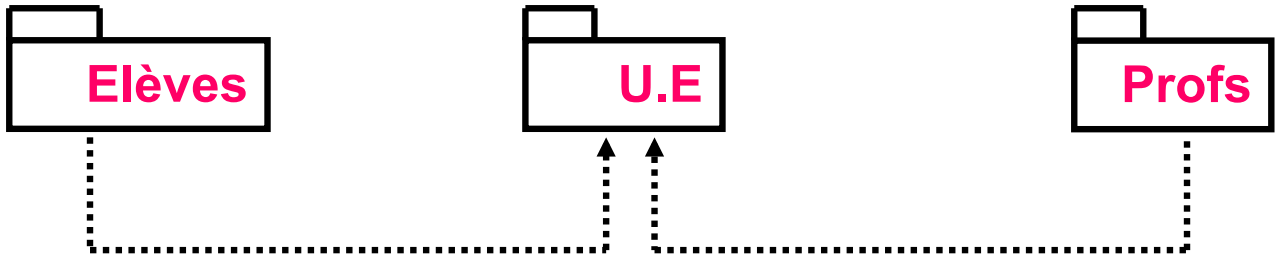


Remarque

Une sous-classe n'hérite qu'une seule fois d'une propriété d'une classe ancêtre se trouvant sur plusieurs chemins menant à cette sous-classe

Concepts avancés : les paquetages

- Paquetage (package)
  - Mécanisme général de regroupement d'éléments en UML
    - Ensemble logique d'éléments du modèle
  - Espace de noms
  - Relations entre paquetages



Concepts avancés : les contraintes

- **Une contrainte :**
  - **information sémantique associée à un élément du modèle qui spécifie les conditions que le modèle doit satisfaire pour être correct**
- L'expression des contraintes permet de compléter le modèle (les cardinalités ne permettant pas de tout décrire)
- Expression des contraintes
  - Écriture entre { }
  - sous forme textuelle
  - ou dans un langage d 'expression des contraintes OCL (Object Constraint Language).

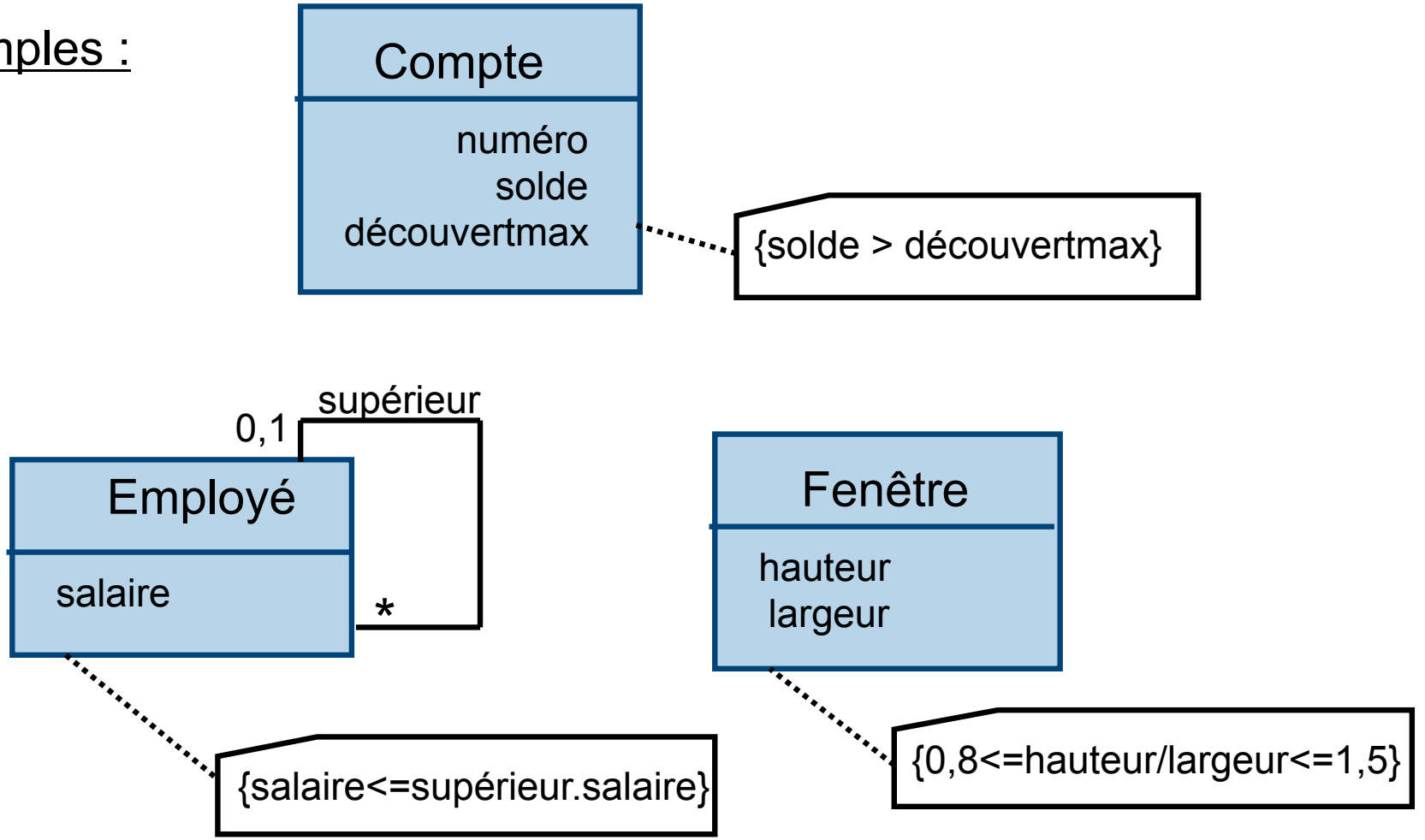
☞ **Les contraintes sont héritées**

Exemples de contraintes dans un diagramme de classes

- Contraintes sur attributs
- Contraintes entre classes (ou contraintes de spécialisation)
- Contraintes entre associations
- Contraintes sur l'extrémité d'une association

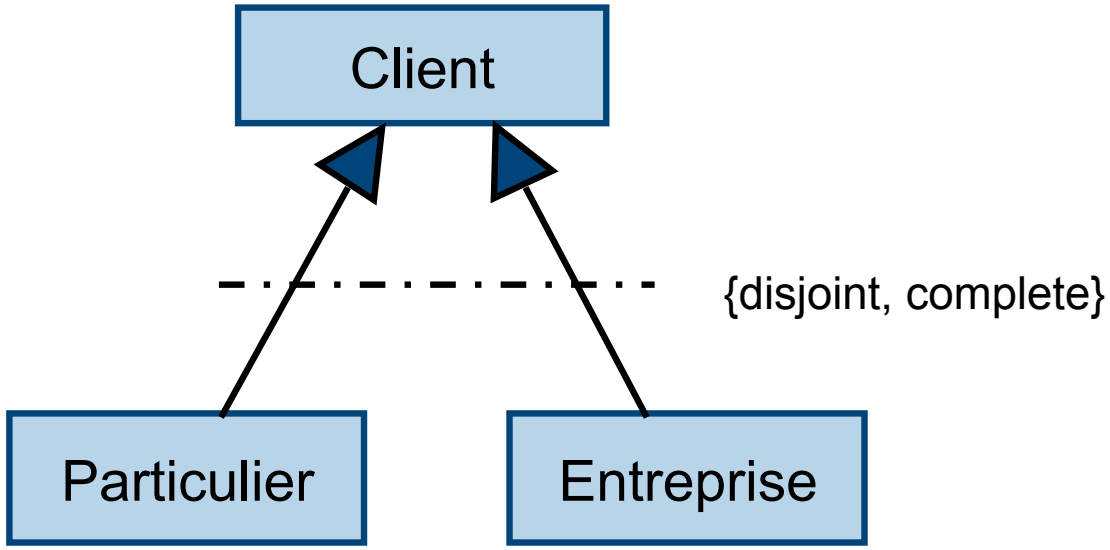
Contraintes sur attributs

Exemples :



ENSIAS

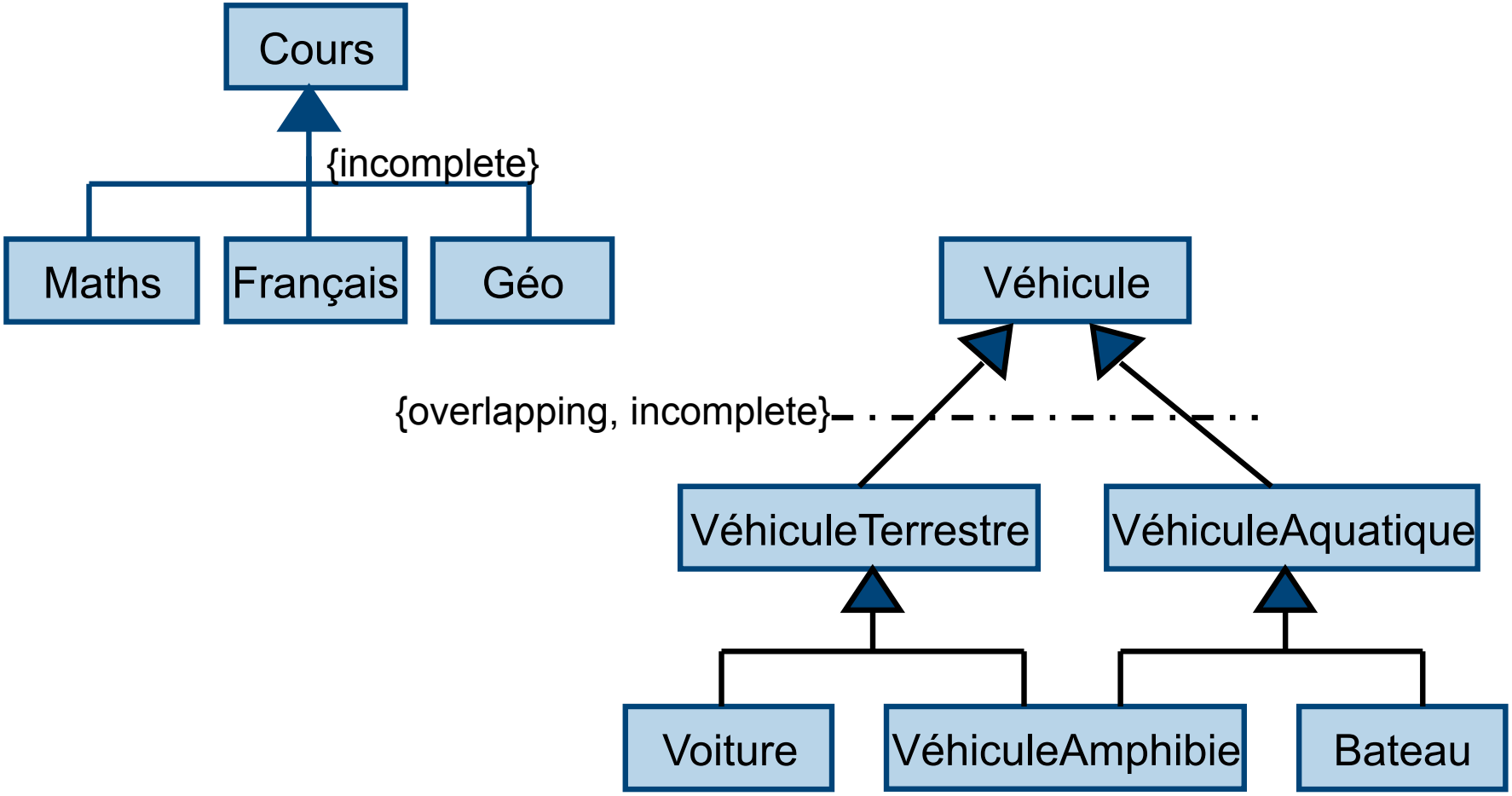
Contraintes entre classes (contraintes de spécialisation)



Différents types de contraintes :

- ✓ Disjoint — les sous-classes sont mutuellement exclusives.
- ✓ Overlapping — (recouvrement) les sous-classes qui se recouvrent peuvent partager des objets.
- ✓ Complete — la spécialisation énumère toutes les classes possibles.
- ✓ Incomplete — la spécialisation est extensible.

Exemples de contraintes de spécialisation

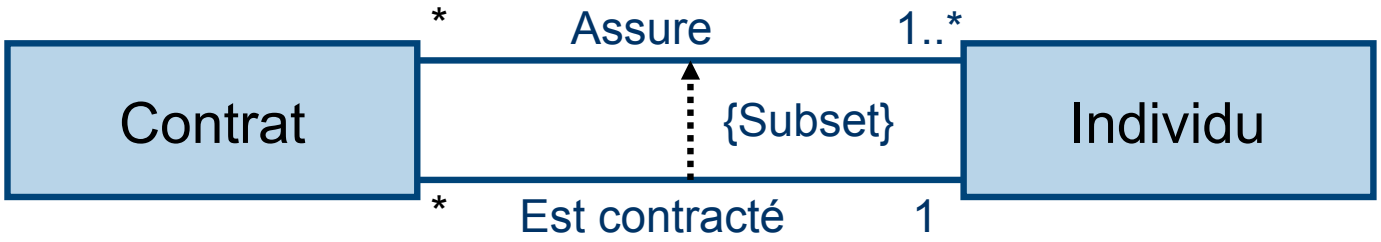


Contraintes entre associations

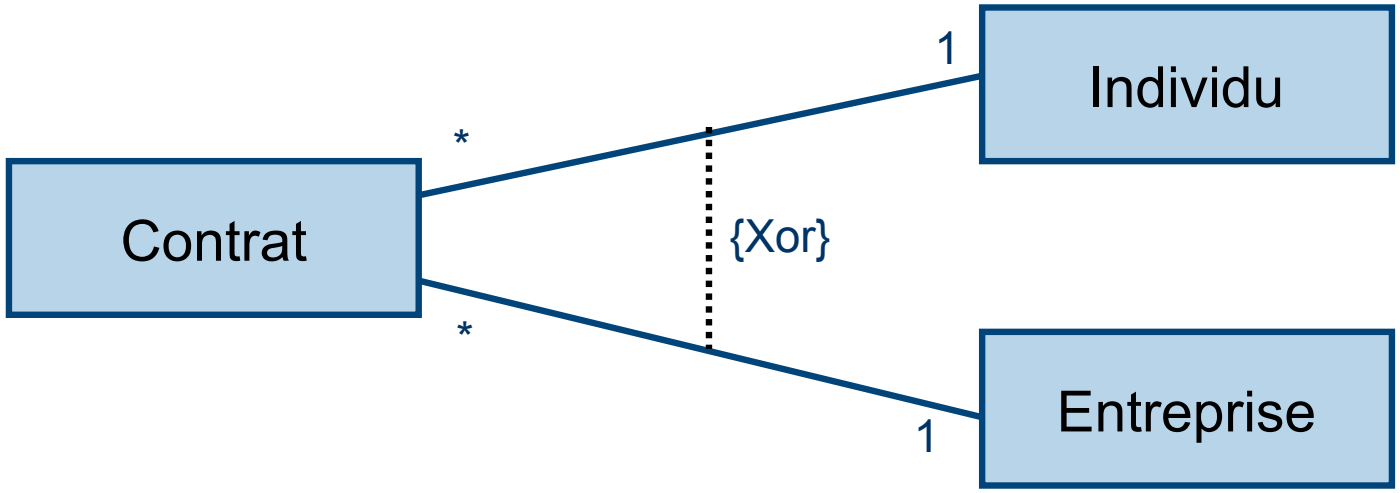
- Contrainte sur associations reliant deux même classes
- Contrainte sur associations reliant des classes différentes

- **Contraintes d'inclusion (subset)**
- **Contrainte d'exclusion (xor)**

Contraintes entre associations

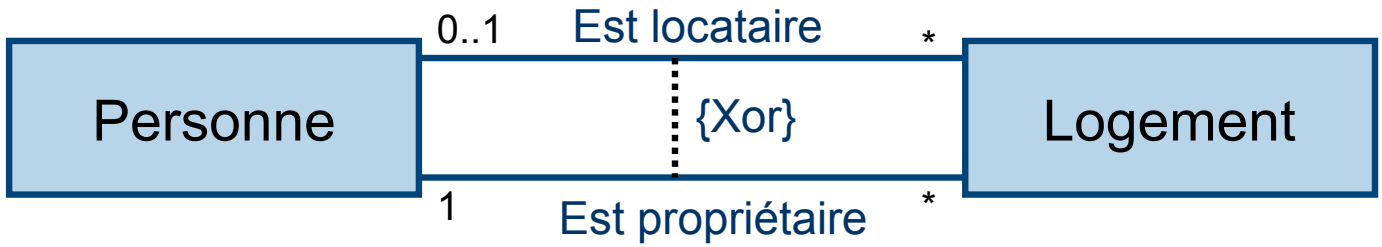


Le contractant d ’un contrat d’assurance fait obligatoirement partie des assurés

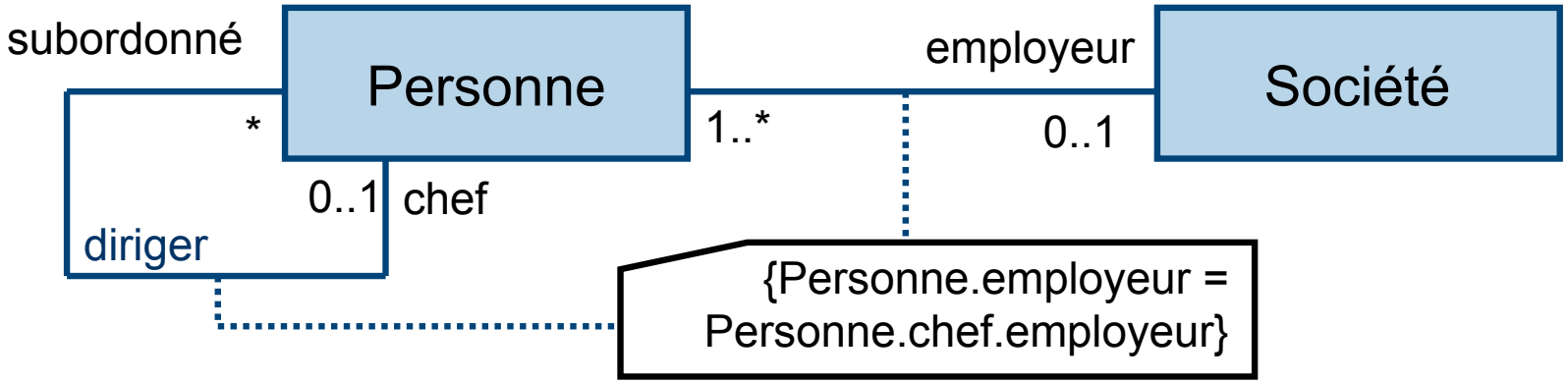


Un contrat assure un individu ou une entreprise mais pas les deux

Contraintes entre associations

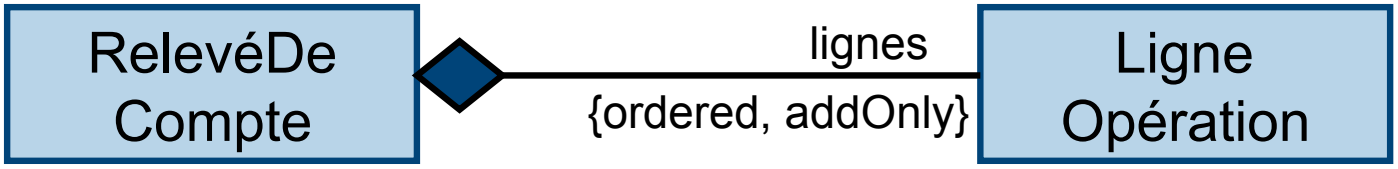


Une même personne est soit locataire, soit propriétaire du logement dans lequel elle habite (exclusion).



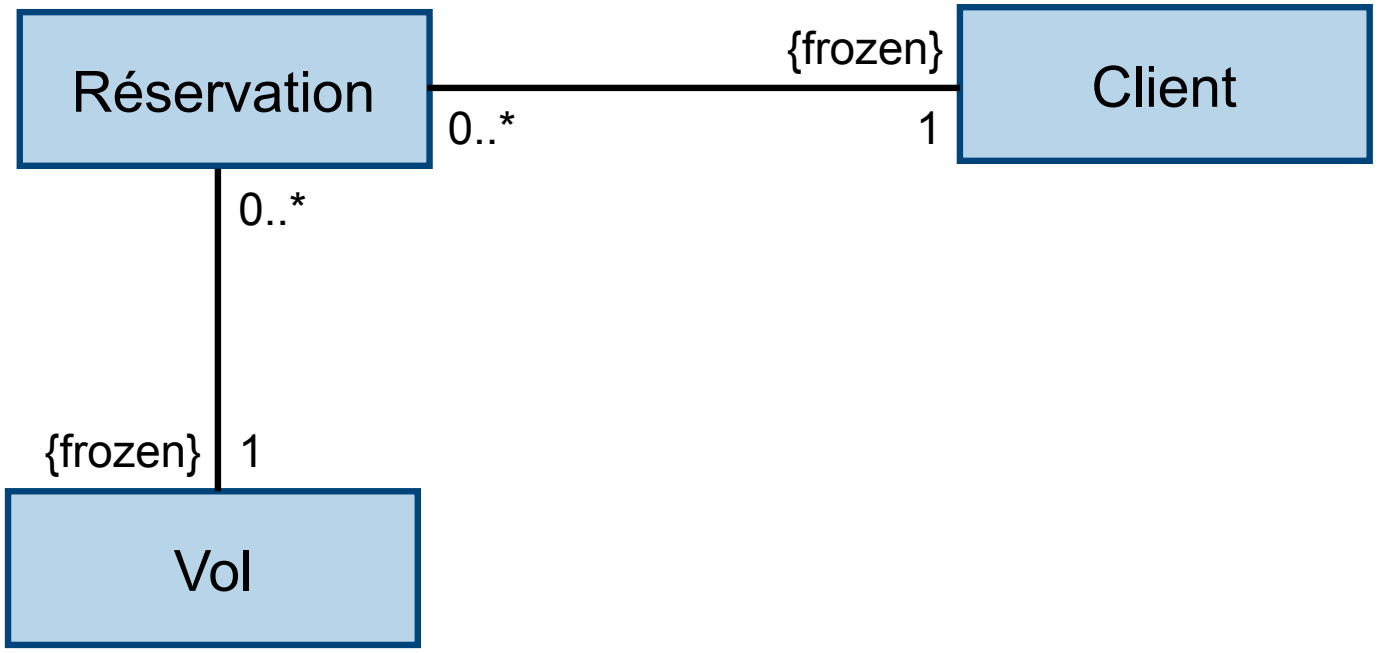
Le supérieur hiérarchique d'une personne a nécessairement le même employeur que cette personne.

Contraintes sur l'extrémité d'une association



{ordered} : les éléments de la collection sont ordonnés
{addOnly} : il est impossible de supprimer un élément

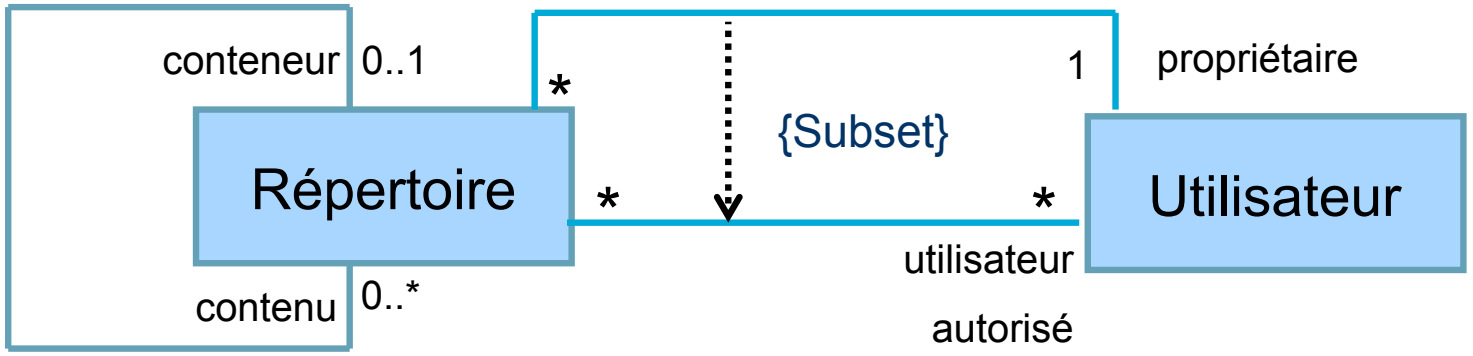
Contraintes sur l'extrémité d'une association



{frozen} : l'instance de la classe à cette extrémité d'association est non modifiable

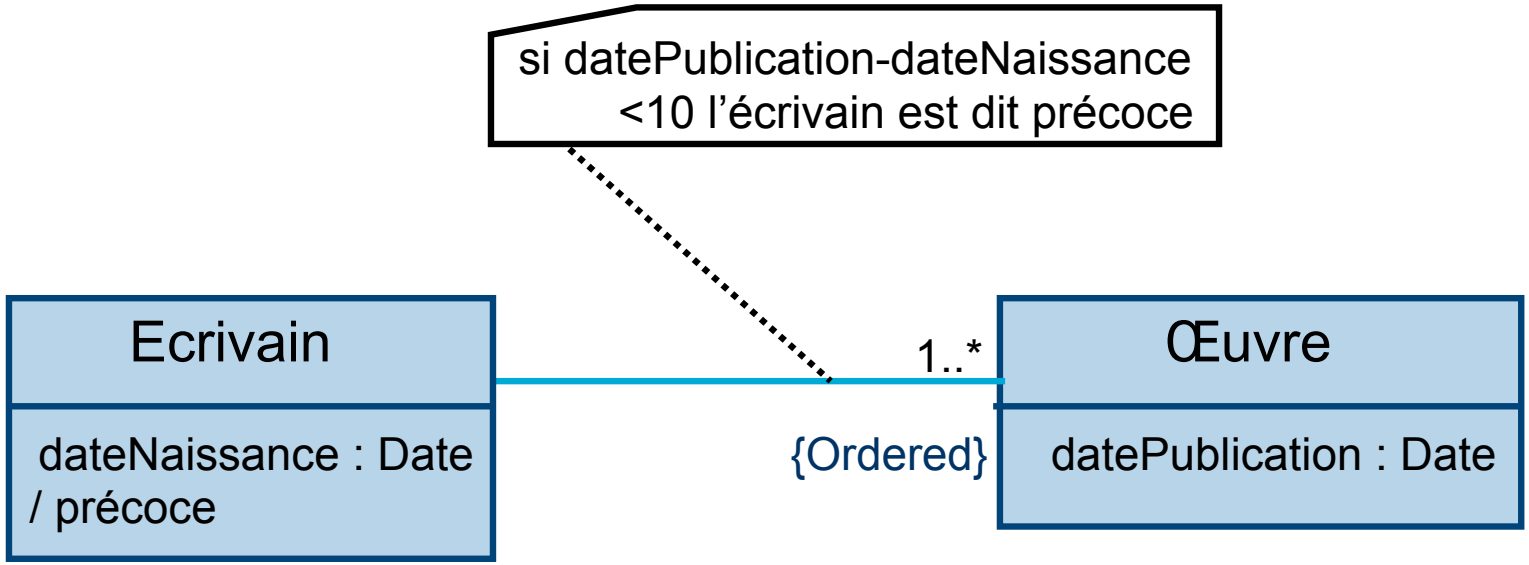
Mise en application

- Représenter sous la forme d'un modèle de classes :
  - Une répertoire peut contenir plusieurs sous répertoires et être éventuellement lui-même être contenu dans un autre répertoire.
  - Chaque répertoire a exactement un utilisateur qui est son propriétaire et plusieurs utilisateurs qui sont autorisés
  - Le propriétaire fait partie de l'ensemble des utilisateurs autorisés



Mise en application

- Représenter sous la forme d'un modèle de classes :
  - Un écrivain possède au moins une œuvre. Ses œuvres sont ordonnées selon l'année de publication. Si la première publication est faite avant l'âge de 10 ans, l'écrivain est dit « précoce »



Mise en application

- **Un groupe d'instituts de formation professionnelle souhaite réaliser un sous- ensemble de son SI**
- Cela concerne le suivi des formateurs des instituts implantés dans les régions. Chaque région est pilotée par une direction régionale qui est composée d'un certain nombre d'instituts. Une direction régionale est caractérisée par un code et un intitulé.
- Chaque institut est caractérisé par un code, un intitulé, une date de création.
- À un institut, sont rattachées une à plusieurs formateurs. Chaque formateur est caractérisé par les données : numéro, qualité (M., Mme, Mlle), nom, prénom, date de naissance, diplôme, spécialité.

Mise en application

- **Un organisme de formation assure différents cours auprès de stagiaires salariés en formation continue**
- Inscriptions
  - En début d'année, chaque salarié remplit une fiche sur laquelle il indique son nom (nom), son email (mail), le code de son entreprise (codent), et la liste des formations qu'il envisage de suivre dans l'année (6 max parmi les 400 présentes au catalogue). Un code lui est ensuite auto. attribué (idsta)
  - Chaque formation est définie par un code (codef), un titre (titref) et une durée (durée), et placée sous la responsabilité d'un consultant caractérisé par un code, un nom et un email (idemp, nomemp et mailemp).
  - Tout consultant doit être disponible un jour par semaine (jsem) et durant une plage horaire définie (hrens) afin de renseigner le public à propos des formations dont il est le responsable

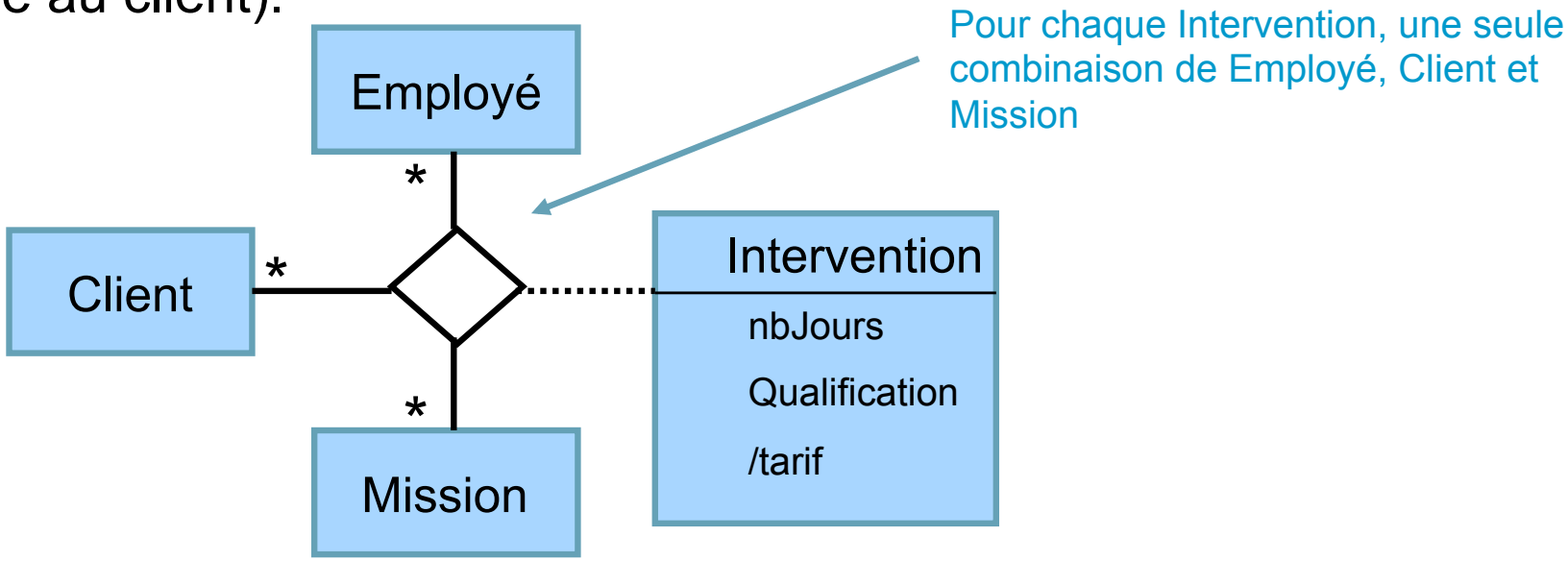
Mise en application

- **Un organisme souhaite gérer la visite de ses représentants au niveau national**

  - Dans chaque région, les représentants visitent des dépôts.
  - On souhaite mémoriser les dates des visites
  - On souhaite connaître quel véhicule de la société le représentant a utilisé
  - Rmq : décomposez toute association n-aire que vous seriez tenté de définir

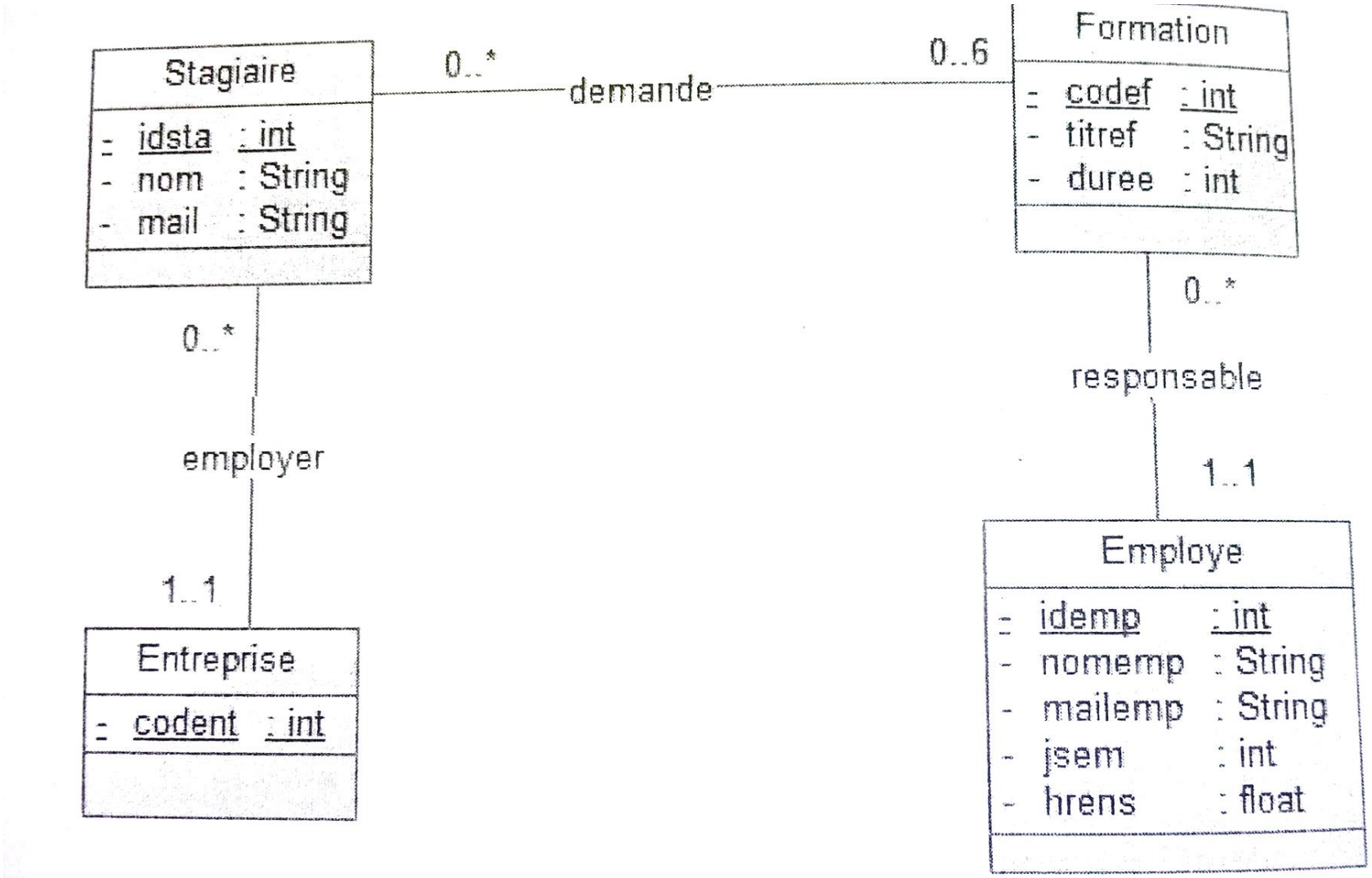
Mise en application

- Représenter sous la forme d'un modèle de classes les situations suivantes :
- Dans une société de service, les employés interviennent sur différentes missions clients.
  - Chaque intervention d'un employé sur une mission est caractérisée par un nombre de jours d'intervention et une qualification d'intervention.
  - La qualification d'intervention détermine le tarif d'intervention (tarif / jour facturé au client).

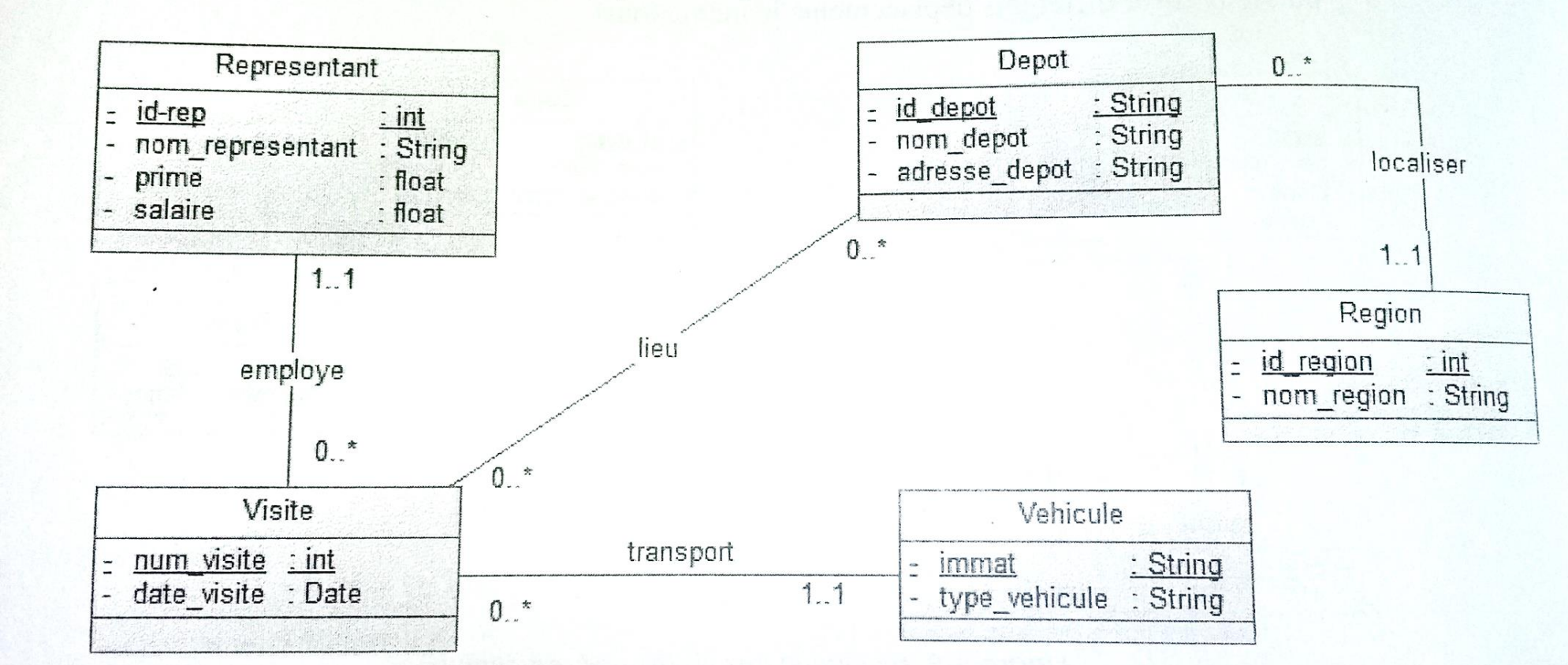


- Annexes

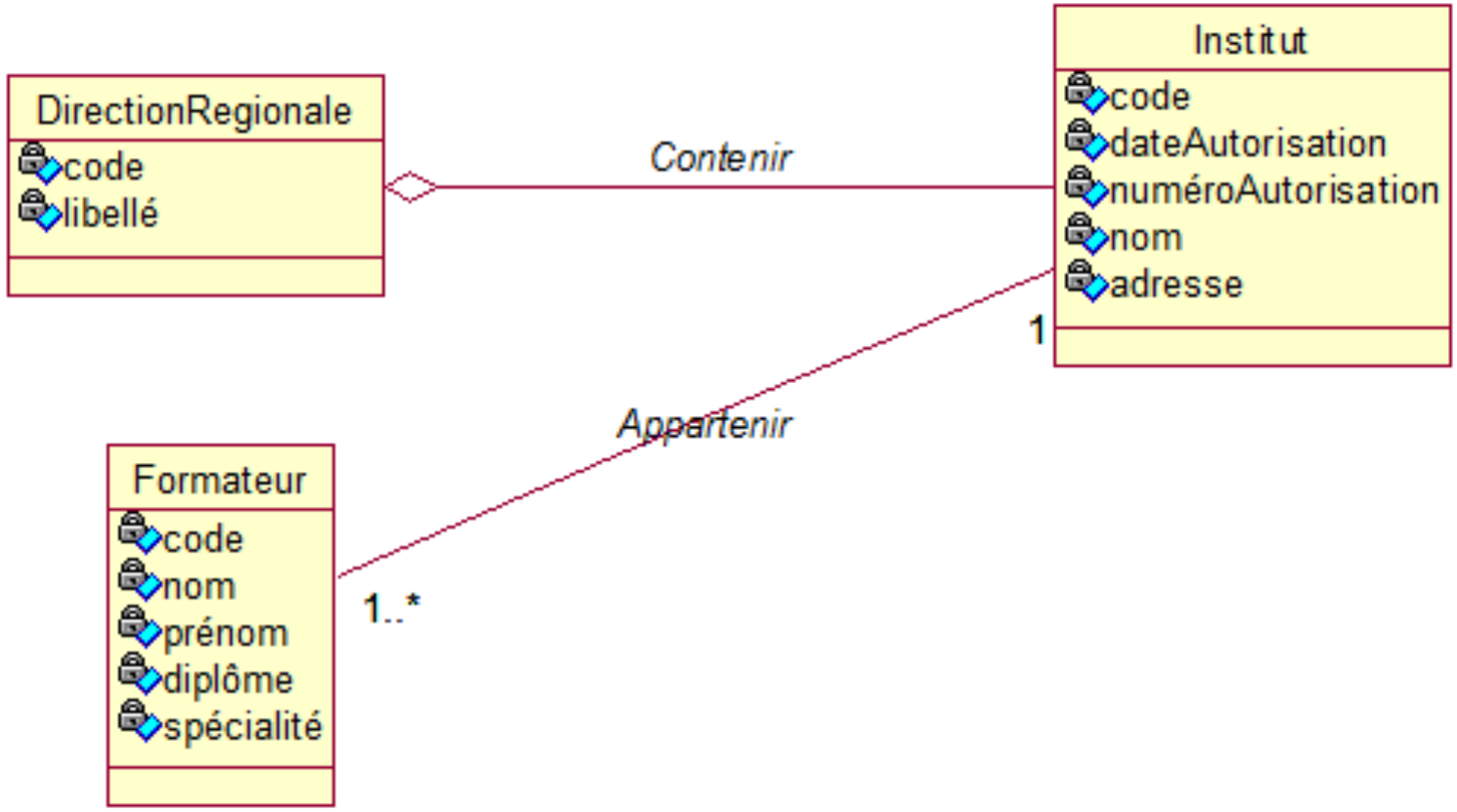
Gestion formation



Visite des représentants

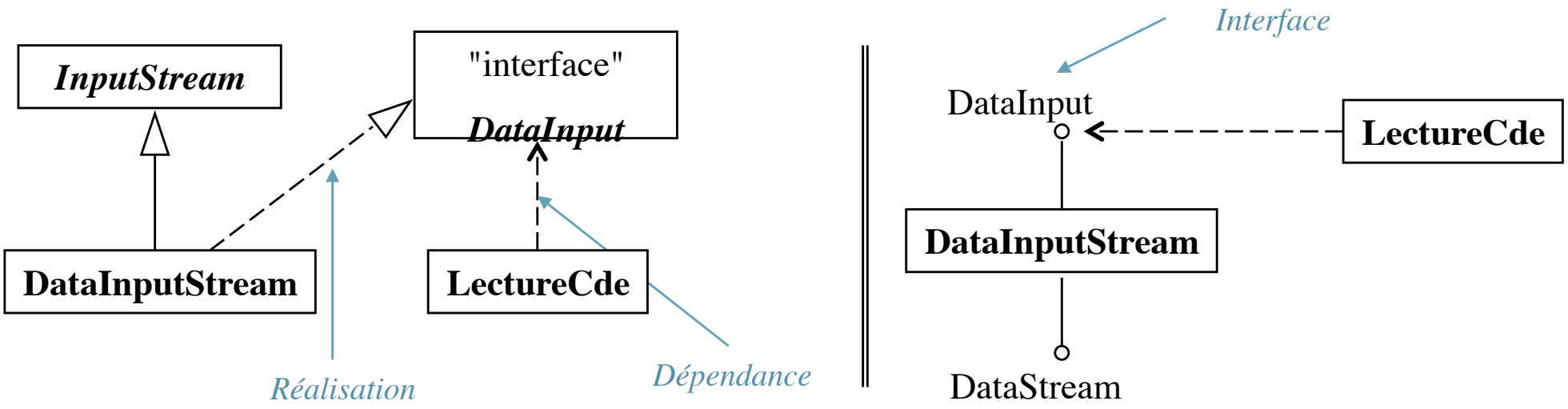


Gestion formation professionnelle



Concepts avancés : Interfaces

- Interfaces et classes abstraites
  - Une des propriétés importantes du modèle objet : faire varier les implémentations des classes indépendamment des interfaces
  - L'interface comme construction indépendante est rarement utilisée (la classe abstraite est souvent utilisée dans ce cas)
  - En Java : une interface est une classe sans implémentation qui ne contient que les déclarations des opérations



Diagrammes de classe : concepts avancés

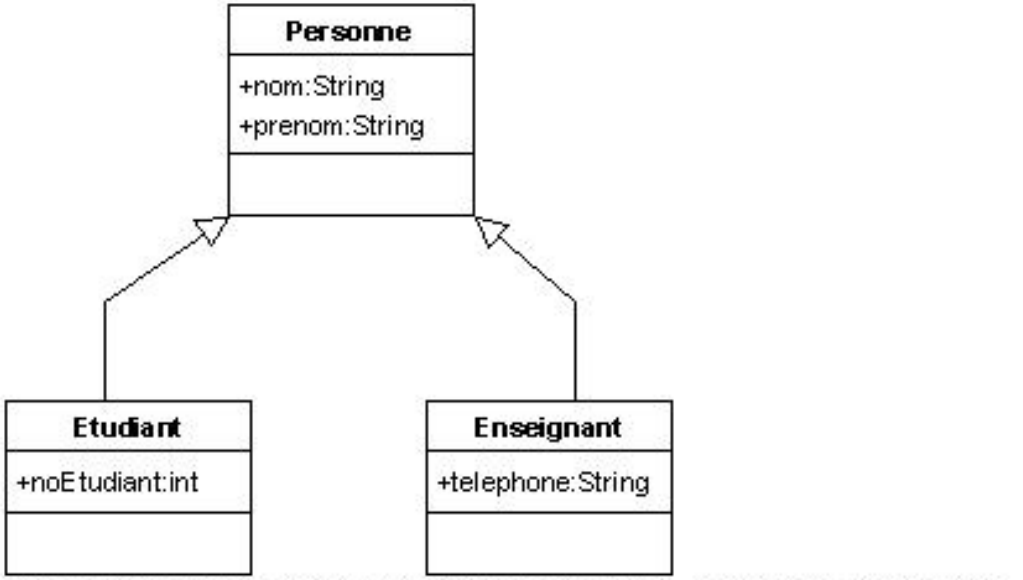
- Objets valeur et objets référence
  - La notion d'identité est plus importante pour les objets référence que pour les objets valeur
  - Objets référence :
    - Ex. Client → un seul objet logiciel représente un client du monde réel
    - Des objets mémorisant l'identité d'un objet Client le feront *via* une référence (le même objet)
    - Les changement seront disponibles pour tous les utilisateurs de l'objet Client
    - Synchronisation dans le cas de copies (rares)

Diagrammes de classe : concepts avancés

- Objets valeur et objets référence
  - Objets valeur
    - Ex. Date → plusieurs objets représentent le même objet du monde réel
    - Comparaison des valeurs et non des identités
    - Objets immuables ou gelés (frozen)
  - UML : utilisation des attributs pour les objets valeur et des associations pour les objets référence

Implémentation : Héritage

```
public class Personne {
        public String nom;
        public String prenom;
}
```

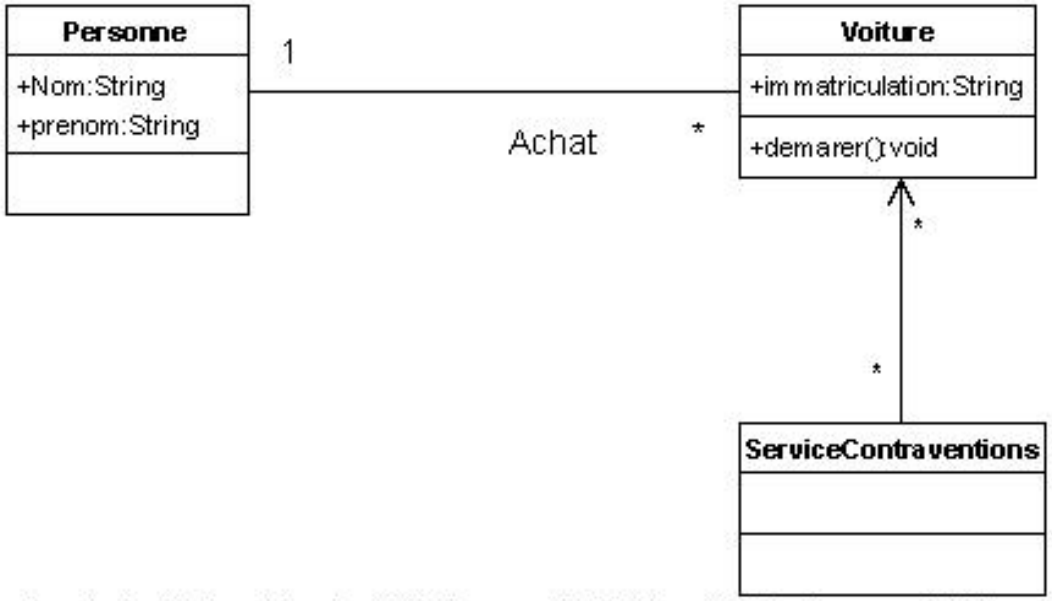



```
public class Etudiant extends Personne {
        public int noEtudiant;
}
```

Implémentation : Association

```
public class Personne {

   public String Nom;
   public String prenom;
   public java.util.Collection voiture =
                new java.util.TreeSet();
}
```




```
public class Voiture {

          public String immatriculation;
          public Personne Propriétaire;
          public void demarer() {   }
}
```

```
public class ServiceContraventions {

public java.util.Collection Voiture = new java.util.TreeSet();
 }
```