

Méthodes pour l'informatisation - compléments

Modélisation des classes

Christine Plumejeaud

Doctorante Informatique UJF

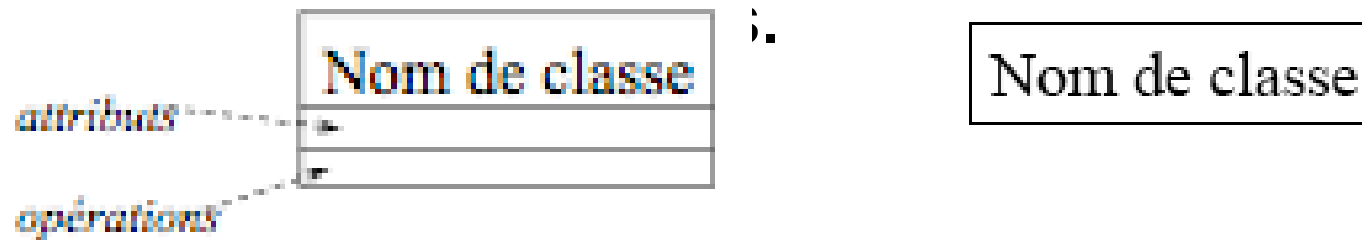
CNAM - centre de Grenoble

Plan

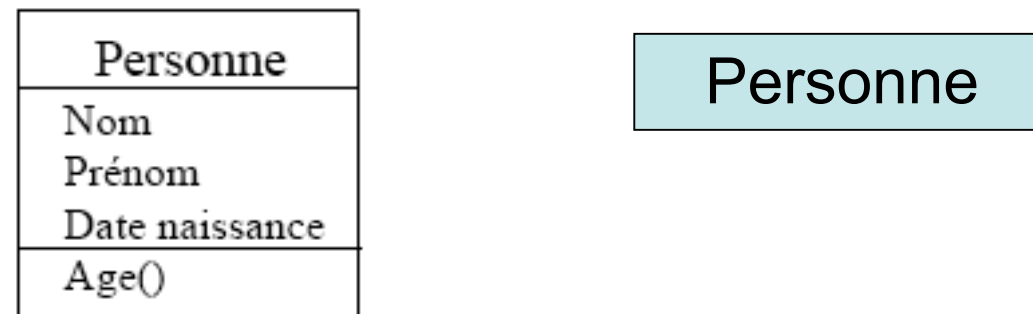
1. Objet et classe
2. Lien et association
3. Généralisation et héritage

Classe

Une **classe** est une description abstraite d'un ensemble d'objets ayant des propriétés similaires, un comportement commun, des relations communes avec d'autres objets et des



Exemples

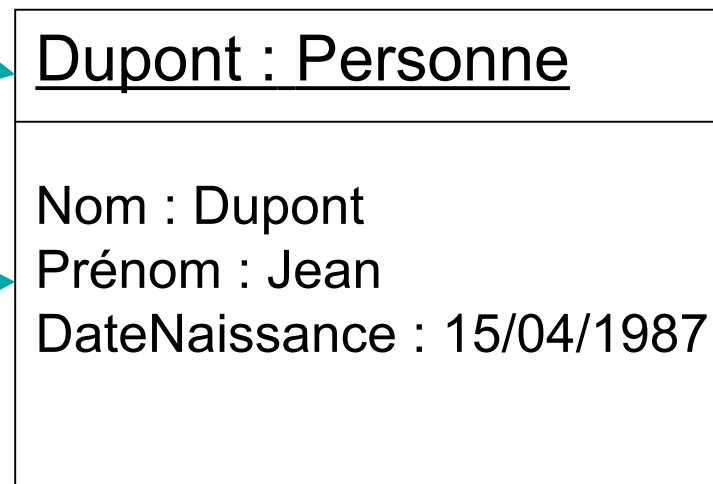


L'objet : instance d'une classe

Représentation

Nom de l'objet : nom de sa classe

Nom des attributs : valeur



Un objet n'a pas d'opération

Les attributs d'une classe

1. Définissent les propriétés des objets d'une classe
2. Chaque nom d'attribut est unique dans une classe
3. L'identification est fournie par le système, elle n'est pas à considérer dans le modèle objet
4. Chaque objet a ses propres valeurs pour chacun des attributs de sa classe.

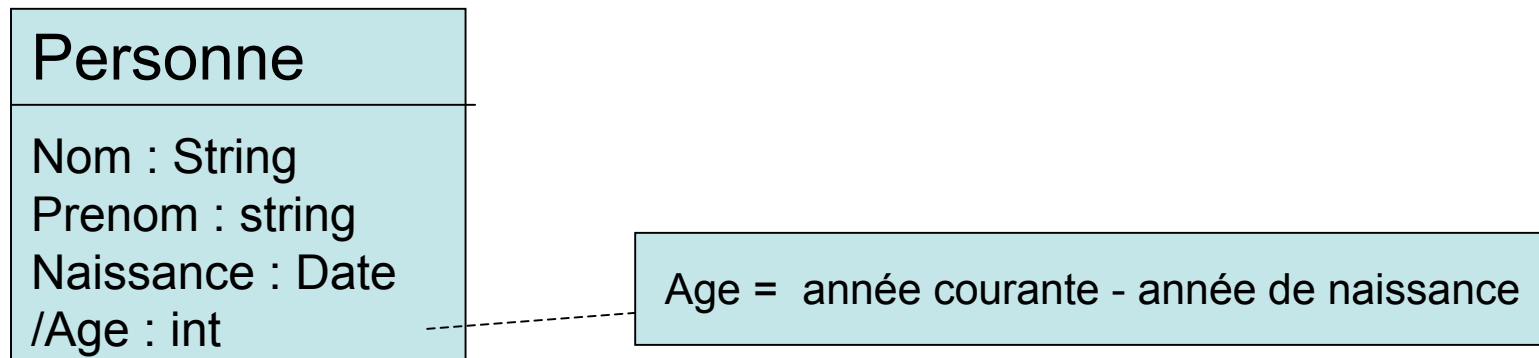
Nom de classe
Nom d'attribut : type = valeur initiale

Personne
Nom : String Prenom : string Naissance : Date

Personne
ID_pers : ID Nom : String Prenom : string Naissance : Date

Les attributs

1. Le type n'est pas obligatoire
2. Les types possibles ne sont pas spécifiés en UML: int, char, string, boolean, etc.
3. Les attributs dérivés notés */nom d'attribut* peuvent se déduire des autres attributs (mais à éviter)
4. Un *attribut de classe* est un attribut partagé par toutes les instances de la classe (*static*)



Les attributs

La multiplicité d'un attribut spécifie le nombre de valeurs possibles pour chacune des instanciations.

- Valeur obligatoire [1] (par défaut)
- Une seule valeur optionnelle [0..1]
- Plusieurs valeurs [*]

Personne
Nom : String [1] Adresse : string [1..*] Naissance : Date [1] Numero de telephone : string [*]

Une personne a une ou plusieurs adresses, zéro ou plusieurs numéros de téléphone, et une seule date de naissance

Les opérations

Opérations (spécifie un service) \neq Méthodes
(implémente l'opération)

Une opération :

- des paramètres (avec un type, un mode, une valeur par défaut). Ils sont optionnels.
- un résultat (typé)

Operation (mode₁ param₁ : Type₁ = val₁, ...,
mode_n param_n : Type_n = val_n) : TypeRésultat

Le mode peut-être :

- In : paramètre d'entrée
- Out : paramètre de sortie (modifié en sortie d'opération)
- Inout : paramètre d'entrée-sortie (lu en entrée et modifié en sortie d'opération)

Niveau de visibilité

Les membres d'une classe (attributs ou opérations) ont un niveau de visibilité

- Public (+) : visible pour tous les clients de la classe
- Protégé (*protected*) (#) : accessible uniquement pour les sous-classes
- Paquetage (*package*) (~) : accessible uniquement pour les classes du même paquetage (par défaut dans java)
- Privé (*private*) (-) : seulement la classe qui le définit peut l'utiliser

Critères pour choisir le niveau de visibilité :

- Compréhension
- Extensibilité
- Contexte

Les énumérations

Une énumération est un type de données qui a un ensemble fini de valeurs.

Noté <<enumeration>>, l'ensemble des valeurs possibles est listé sous le nom de l'énumération

Carte
<code>_couleur : Couleur</code> <code>_rang : Rang</code>

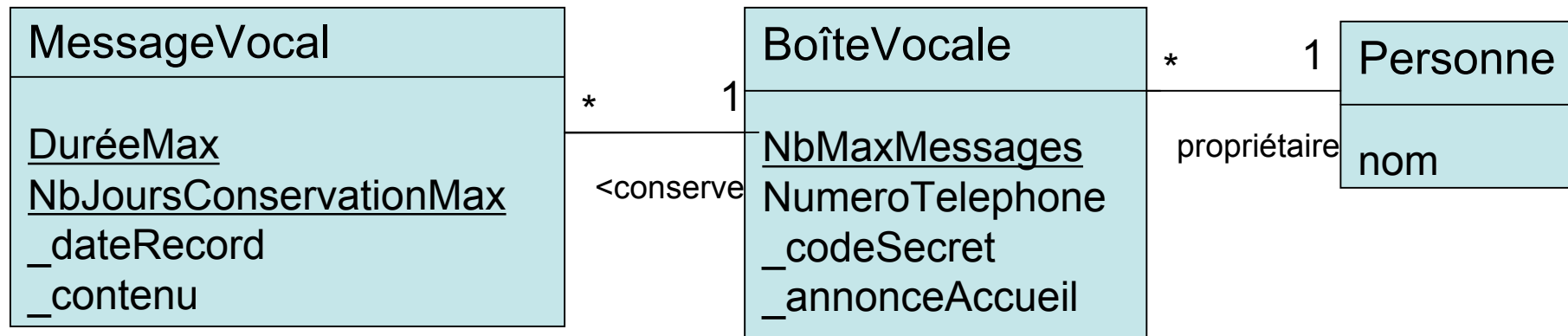
<< enumeration >> Couleur
Pique Trèfle Cœur Carreau

<< enumeration >> Rang
As Roi Dame ...

Portée des membres d'une classe

La *portée* indique si une propriété s'applique à un seul objet, ou est partagée par toutes les instances de la classe.

- Si la portée est la classe, on dit que le membre (attribut ou opération) est statique.
- Statique s'utilise pour contenir l'extension d'une classe (l'ensemble d'objets de la classe)



Classe utilitaire

Une classe utilitaire est une sorte de boîte à outils : elle ne s'instancie pas, et tous ses membres publics (attributs et méthodes) sont statiques (de portée de classe)

Permet de regrouper un ensemble de valeurs et d'opérations.

<<utilitaire>> Math
+pi : float = 3.14159
+sin(x:float):float +cos(x:float):float +sqrt(x:float):float +pow(x:float, n:int):float

Math ne peut être instanciée.

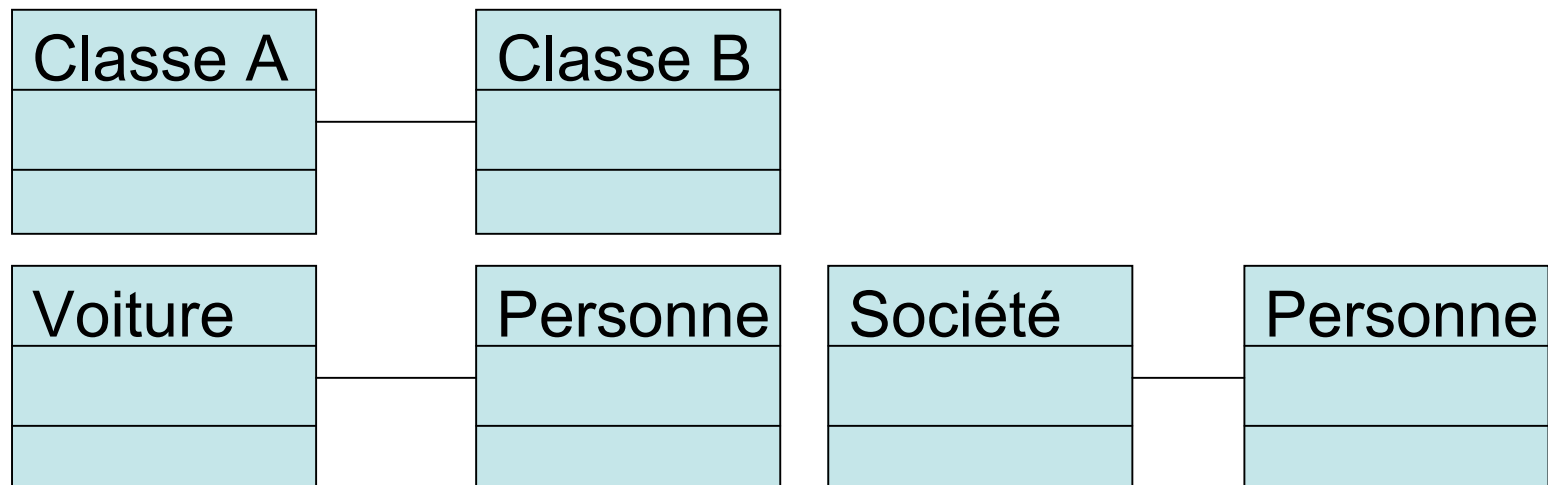
On l'utilise ainsi :

pour un disque de rayon R

Aire = Math.pi * Math.pow(R, 2)

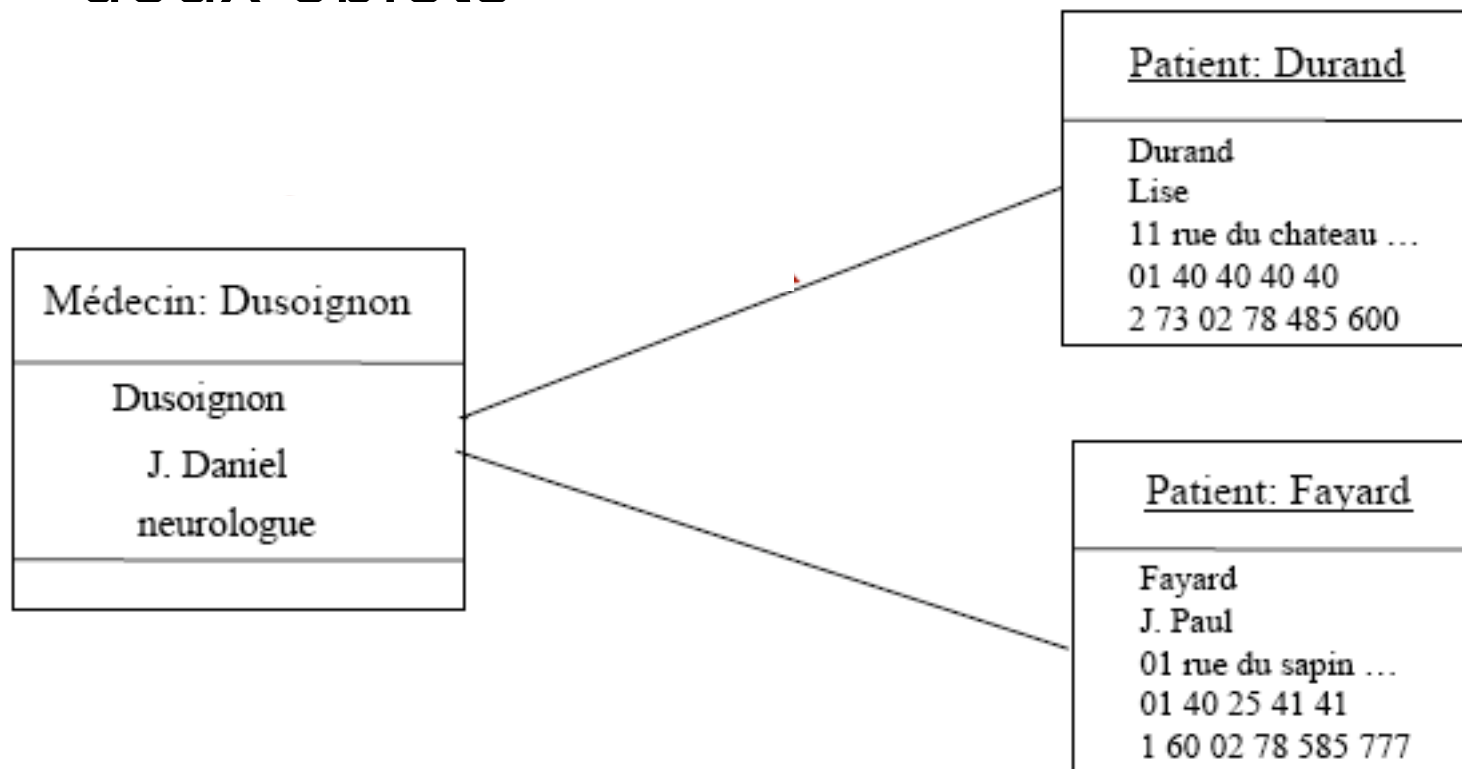
Association

- Elle représente une relation structurelle entre classes d'objet
- Elle symbolise une information dont la durée de vie n'est pas négligeable par rapport à la dynamique générale des classesinstanciées.



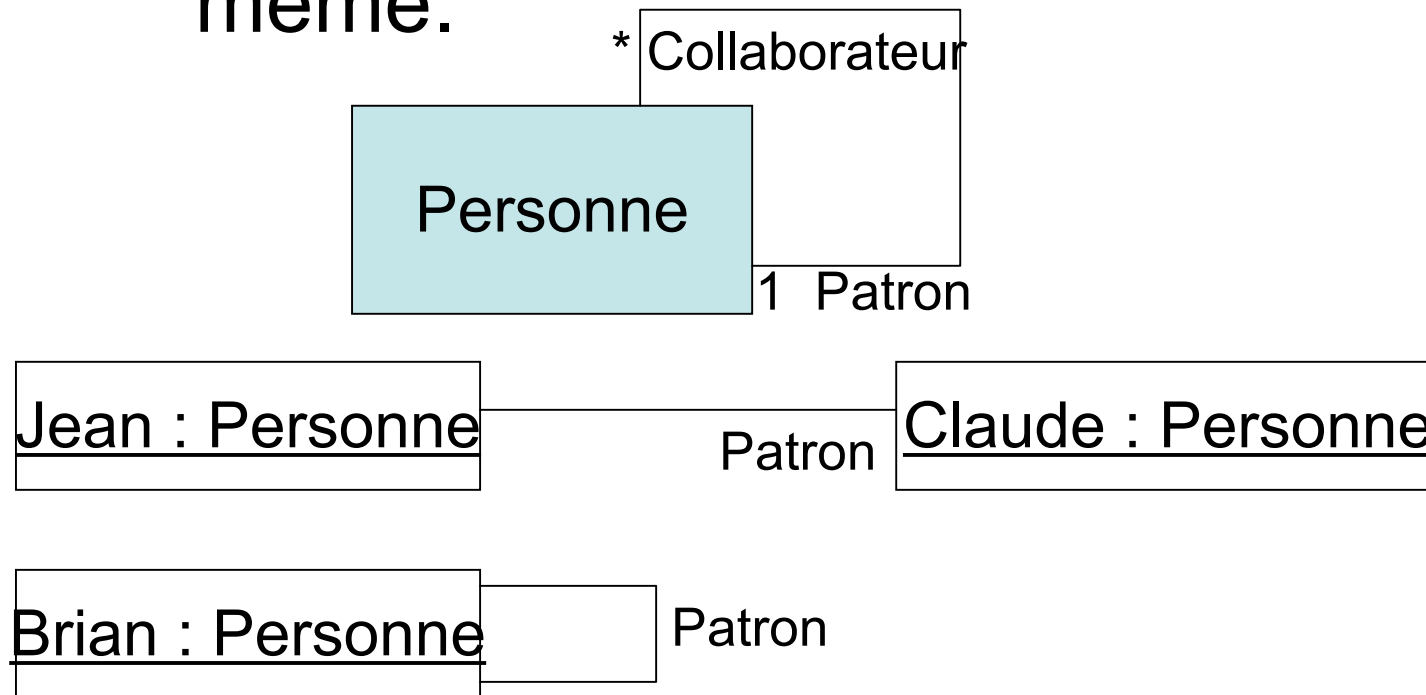
Les liens entre les objets

Un lien est une connexion logique entre deux objets



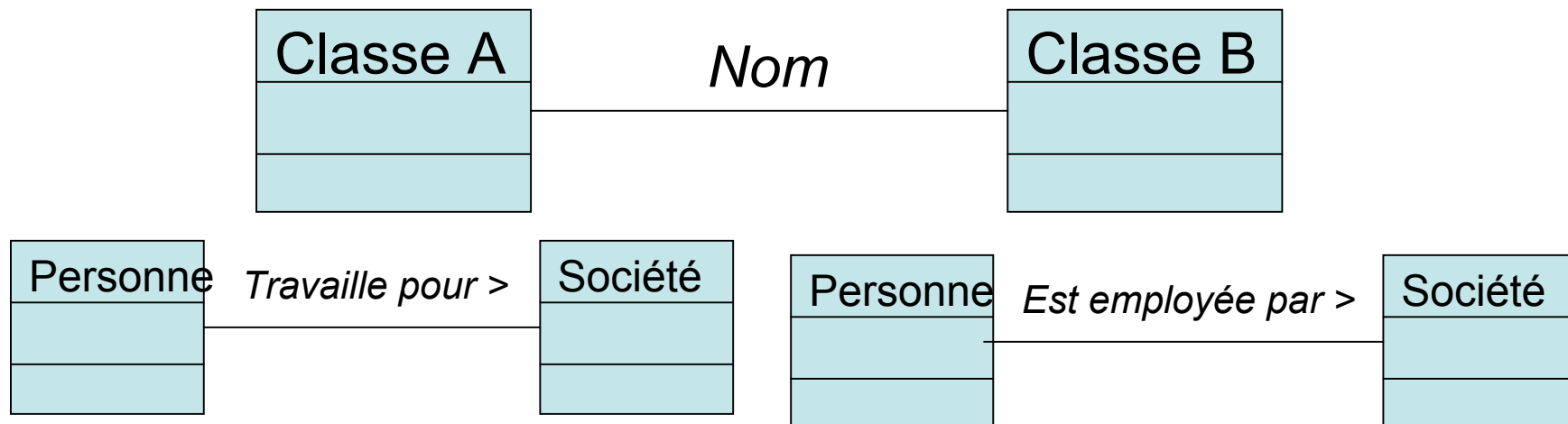
Liens entre les objets

1. Les liens instances des association réflexives peuvent lier un objet à lui-même.



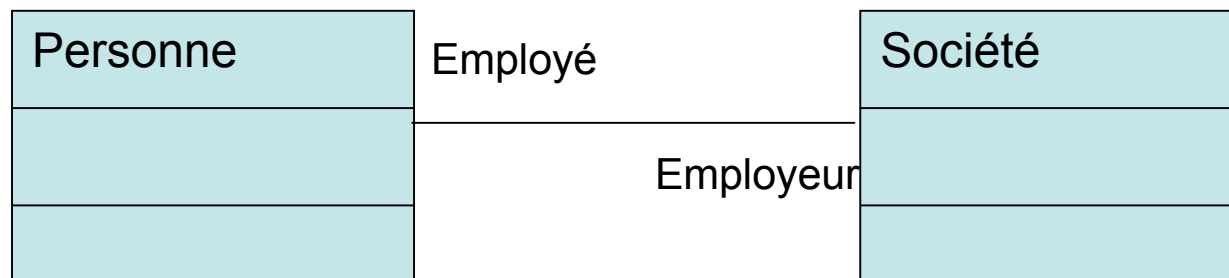
Nommage des associations

- Le nom d'une association apparaît en italique, au milieu du lien
- Nommer l'association par une forme verbale (active ou passive)



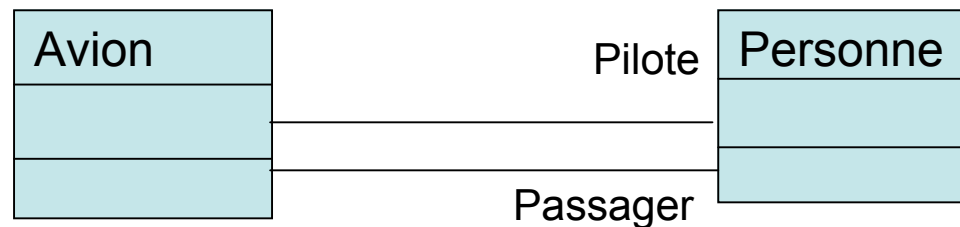
Nommage des rôles

- Chaque association binaire possède 2 rôles
- Le rôle décrit comment une classe voit une autre classe au travers d'une association
- Le nommage des associations et des rôles ne sont pas exclusifs l'un de l'autre

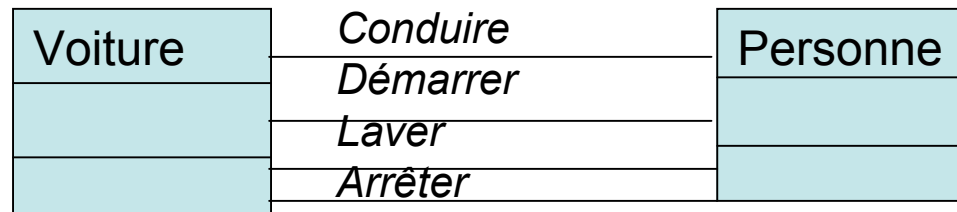


Nommage des rôles

- Le nommage des rôles prend tout son intérêt lorsque plusieurs associations relient 2 classes. Chaque association exprime un concept distinct



- La présence d'un grand nombre d'associations entre 2 classes est suspecte : signe d'une mauvaise décomposition



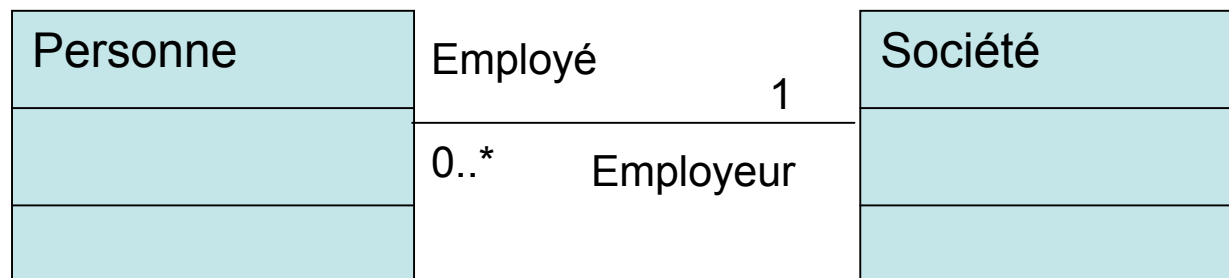
Multiplicité des associations

La *multiplicité* est une information portée par le rôle, sous la forme d'une expression entière bornée.

1	Un et un seul
0..1	Zéro ou un
M..N	De M à N, entiers positifs
*	De zéro à plusieurs
0..*	
1..*	De un à plusieurs

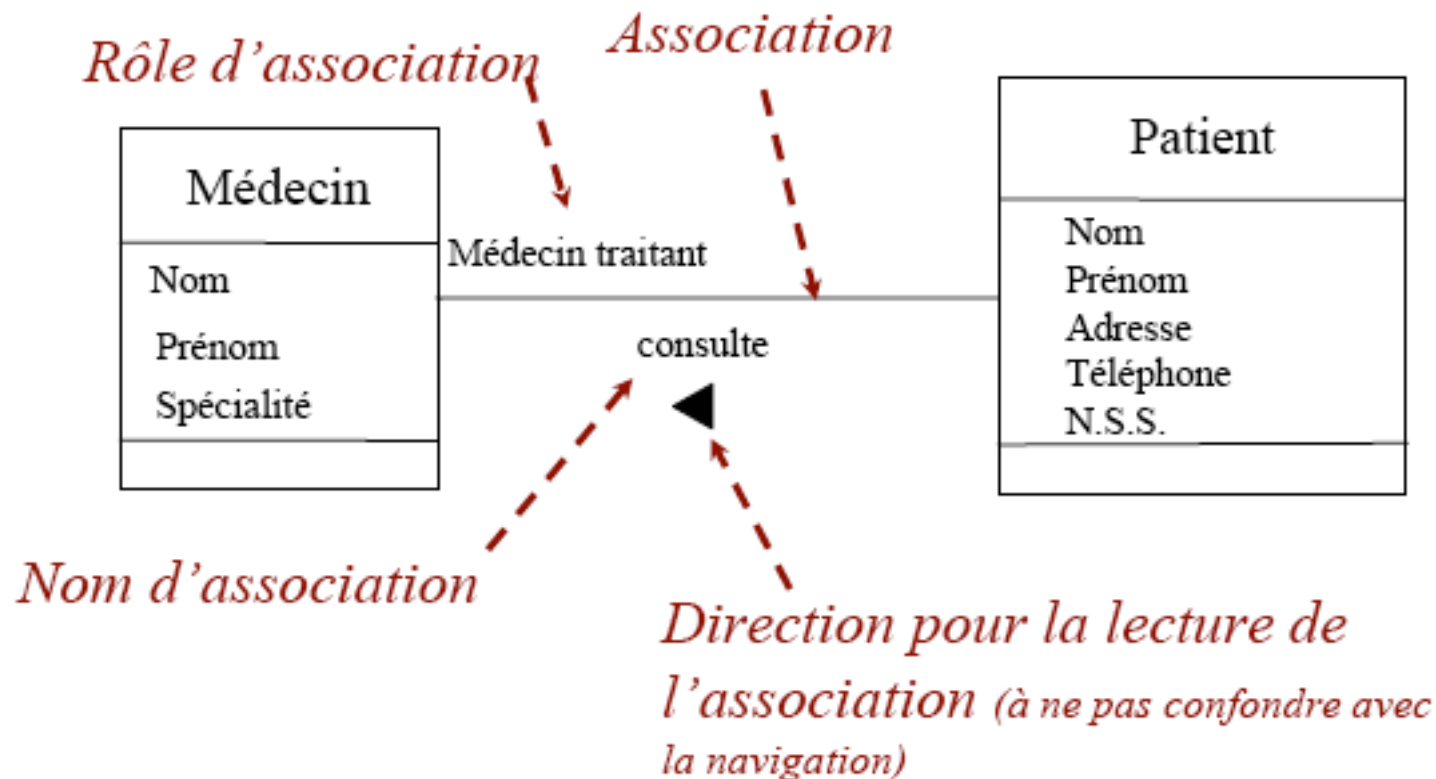
Multiplicité des associations

- Une valeur de multiplicité > 1 implique une collection d'objets.
- Les valeurs de multiplicité sont des contraintes liées au domaine de l'application, valables durant l'existence des objets. A ignorer durant les régimes transitoires



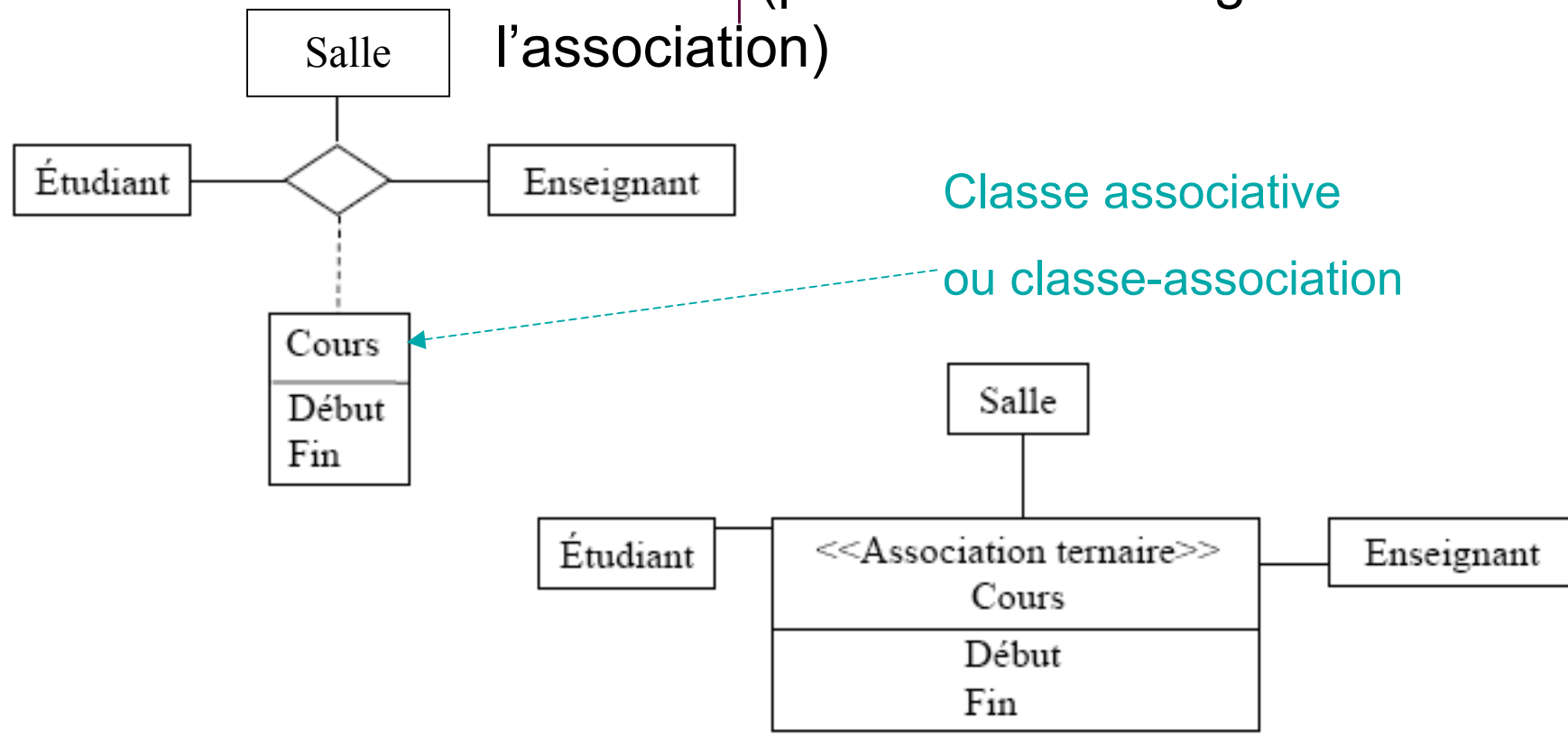
Les associations dans un diagramme de classe

Une association est une relation n-aire entre n classes d'objets. Les plus utilisées sont les relations binaires (entre 2 classes)



Association n-aire

Exemple : une association ternaire avec réification (promotion au rang de classe de l'association)

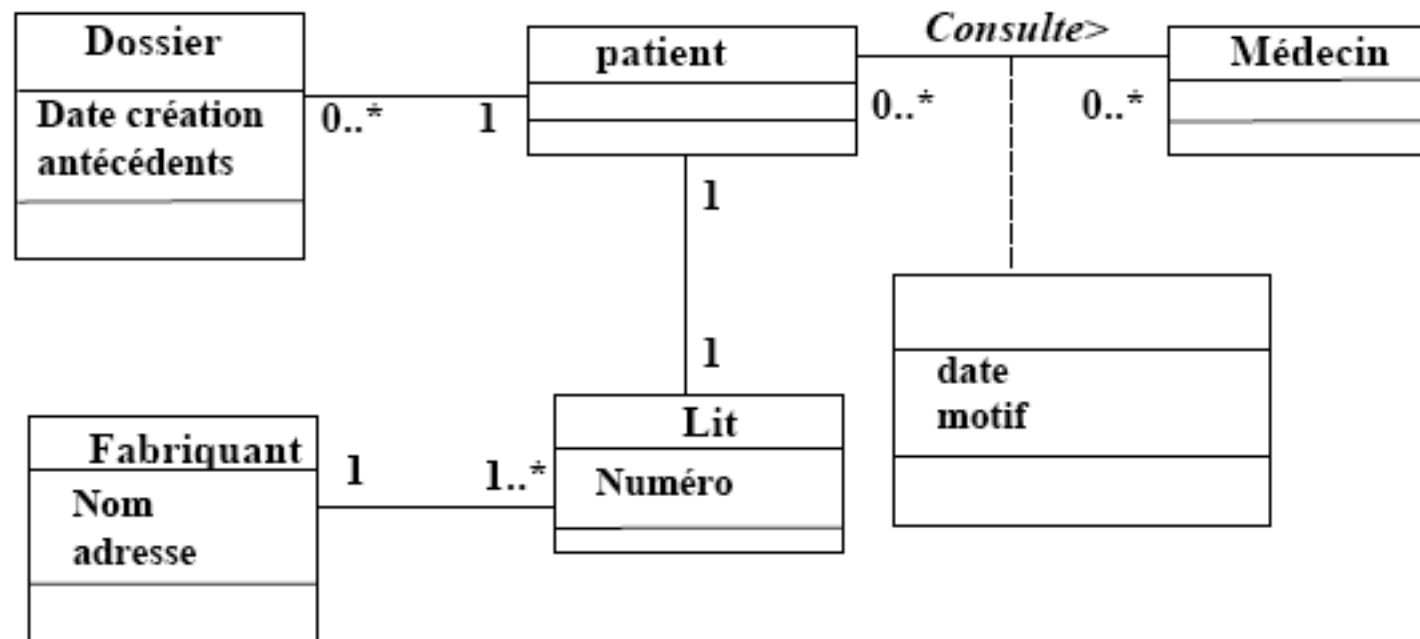


Placements des attributs en fonction des valeurs de multiplicité

1. La réification des associations prend tout son sens pour les associations **N vers N**
2. Pour les associations **1 vers 1**, les attributs de l'association peuvent toujours être déplacés dans une des classes qui participent à l'association.
3. Pour les associations **1 vers N**, le déplacement est généralement possible vers la classe côté N; il est fréquent de promouvoir l'association au rang de classe pour augmenter la lisibilité ou en raison de la présence d'associations vers d'autres classes.

Placements des attributs en fonction des valeurs de multiplicité

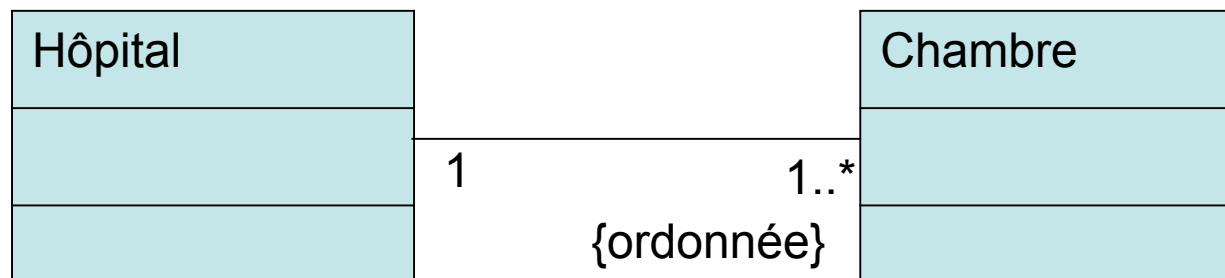
Exemple



Contraintes sur les associations

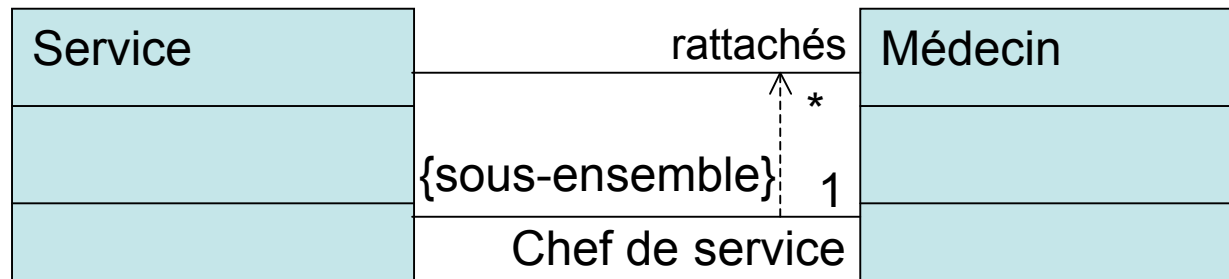
Toutes sortes de contraintes peuvent être définies sur une relation ou un groupe de relations. La multiplicité est une sorte de contrainte.

- Elles se représentent par des expressions placées entre accolades
- La contrainte **{ordonnée}** peut être placée sur le rôle pour spécifier qu'une relation d'ordre décrit les objets placés dans la collection.

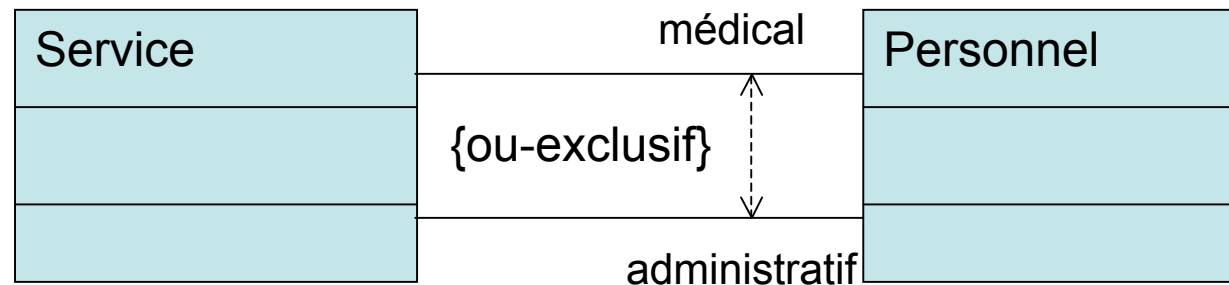


Contraintes sur les associations

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



- La contrainte **{ou-exclusif}** précise que pour un objet donné, une seule association est valide

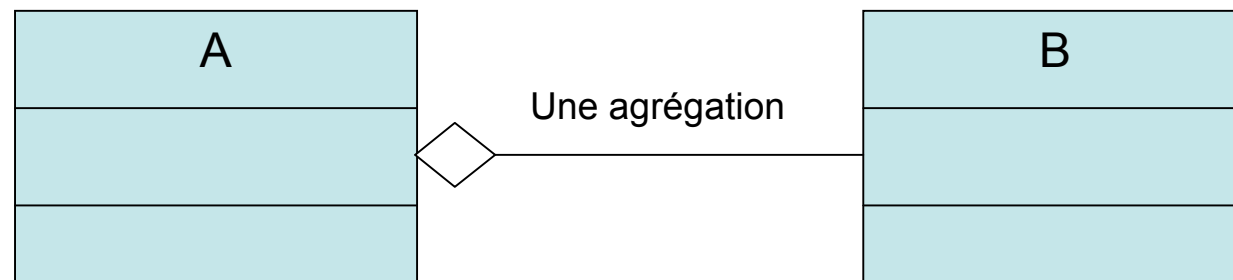


Agrégation

Représente une association **non symétrique** dans laquelle une des extrémités joue un rôle prédominant par rapport à l'autre extrémité. Quelle que soit l'arité, l'agrégation ne concerne qu'un seul rôle de l'association.

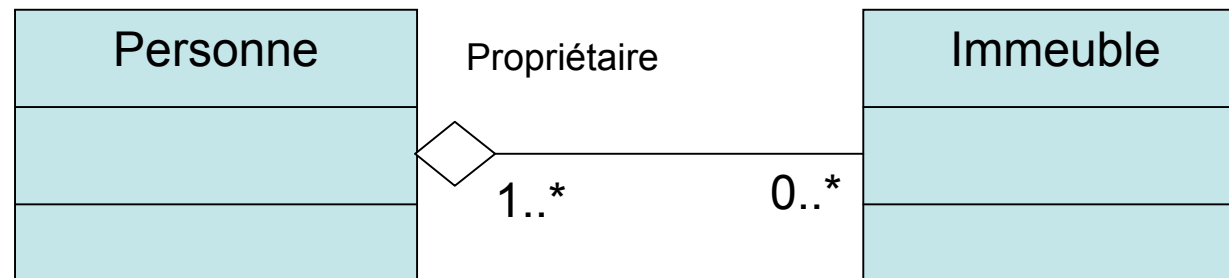
Les critères suivant impliquent une agrégation :

- Une classe fait partie d'une autre classe
- Les valeurs d'attribut d'une classe se propagent dans les valeurs d'attribut d'une autre classe
- Une action sur une classe implique une action sur une autre classe
- Les objets d'une classe sont subordonnés aux objets d'une autre classe



Agrégation

L'agrégation peut-être multiple, comme l'association



Composition

La composition est une agrégation réalisée par valeur sur les attributs.

Les attributs sont physiquement contenus dans l'agrégat.

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

La valeur 0 du côté du composant correspond à un attribut non renseigné.



Agrégation versus Composition

L'agrégation représente une relation de **partie-de**. La sémantique peut être imprécise

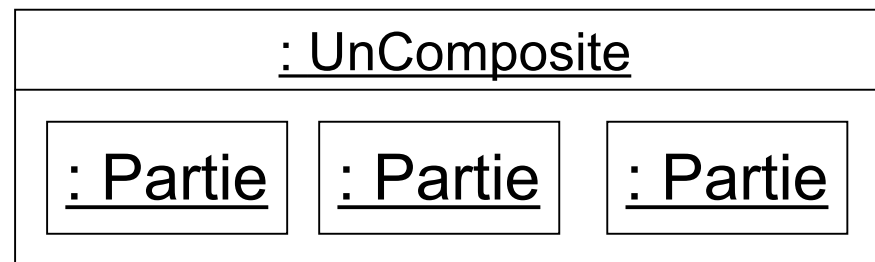
Composition est une relation plus « forte » :

- Chaque partie peut appartenir à une et un seul tout à un instant donné
- Lorsque le tout est détruit (n'a plus d'existence) il en est de même pour toutes les parties qui le composent.

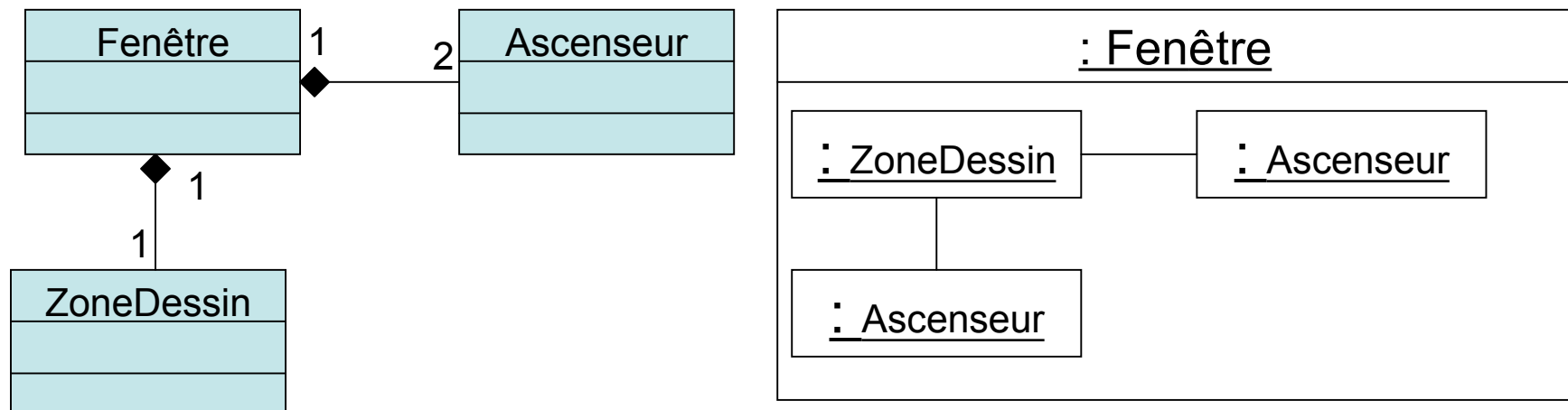
Exemple : une voiture est composée avec 4 roues. Une roue ne peut pas être partagée par plusieurs voiture. Et lorsque la voiture brûle, ses roues aussi.

Objet composite

Les objets composés de sous-objets peuvent être représentés au moyen d'un objet composite.



Les objets composites sont instances de classes composites.

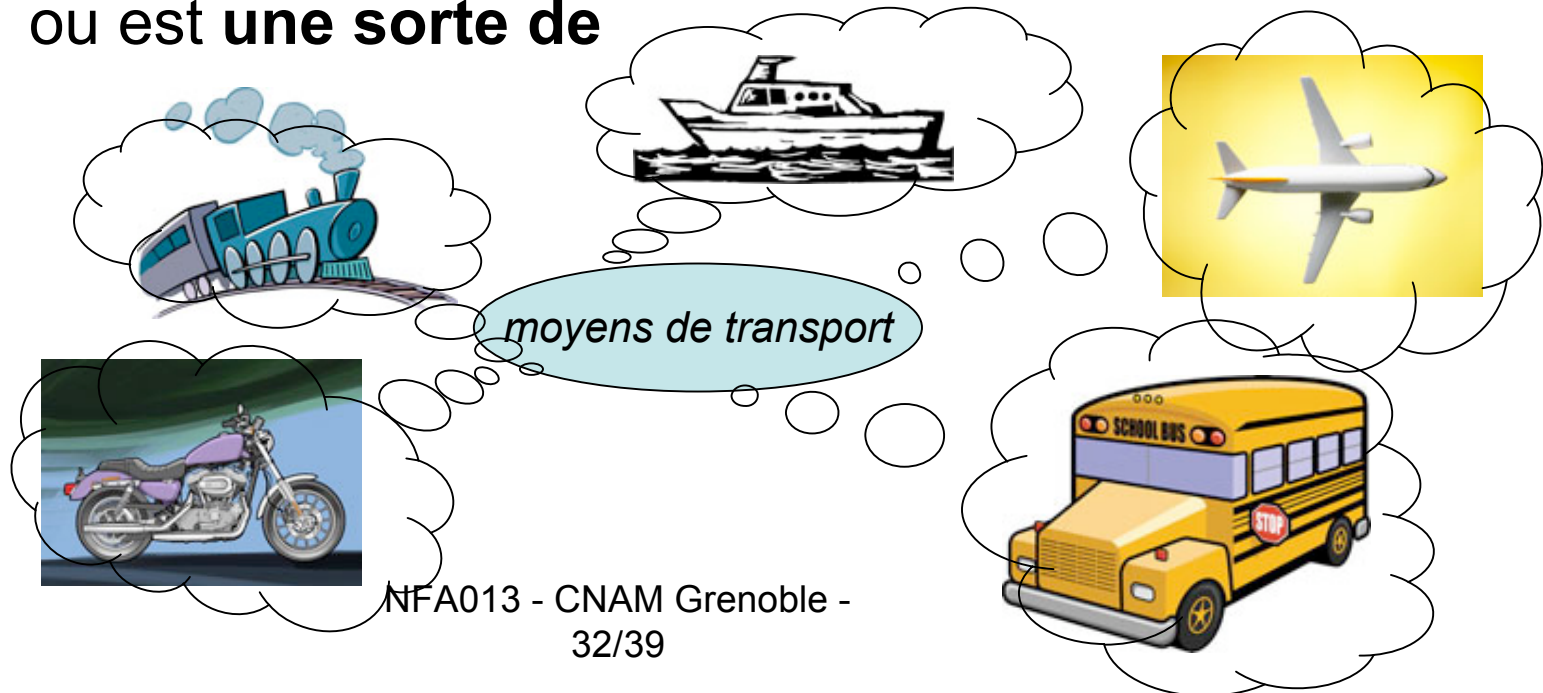


Généralisation

La généralisation s'applique

- aux classes
- aux paquetages
- aux cas d'utilisation

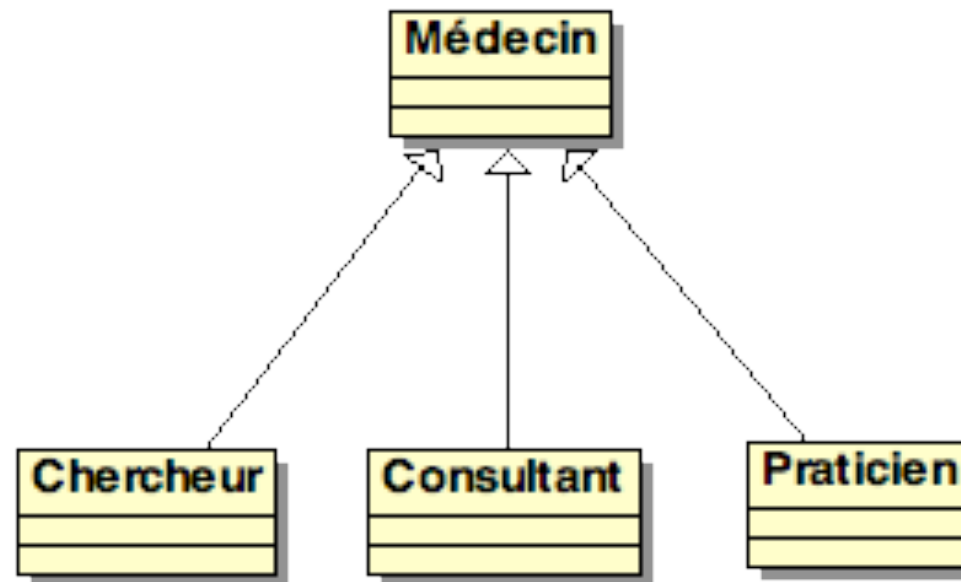
Dans le cas des classes, cette relation signifie **est un** ou est **une sorte de**



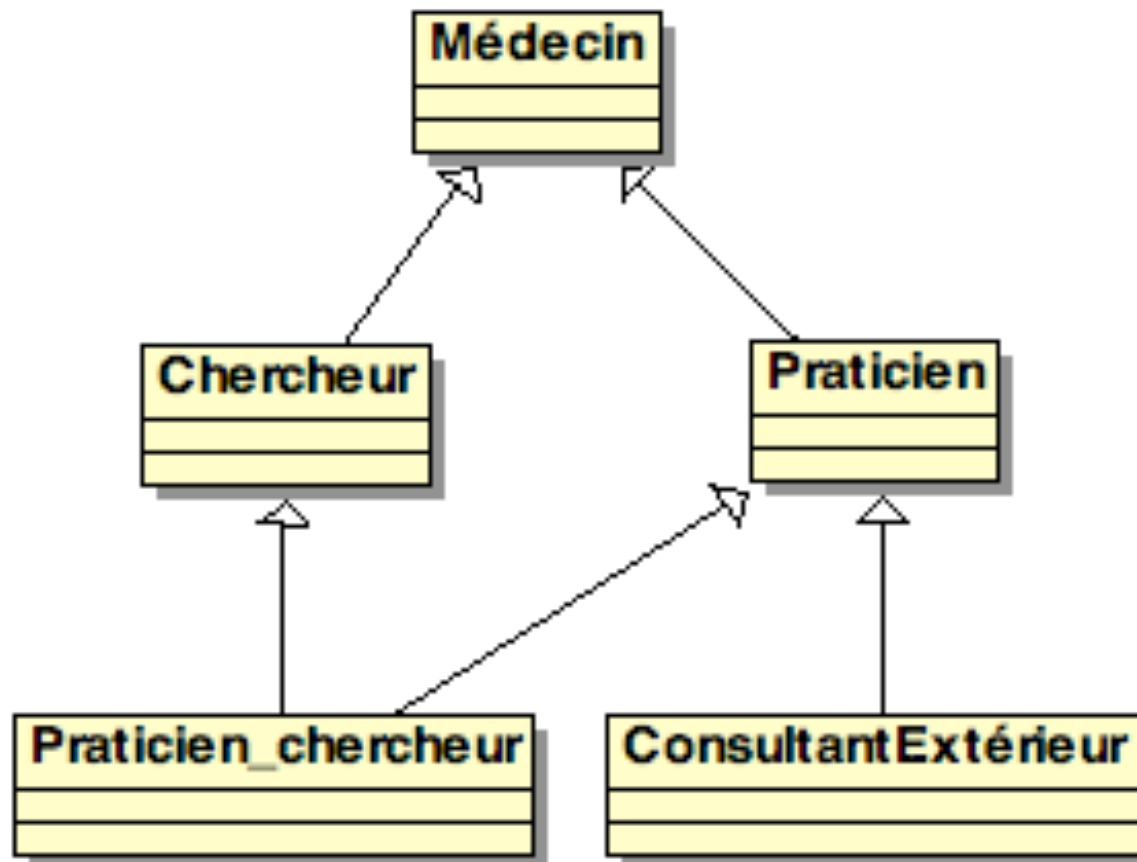
Généralisation

En programmation, la relation de généralisation est très souvent réalisée en utilisant la relation d'héritage entre classes

En UML, l'héritage est une manière de classifier mais ce n'est pas la seule



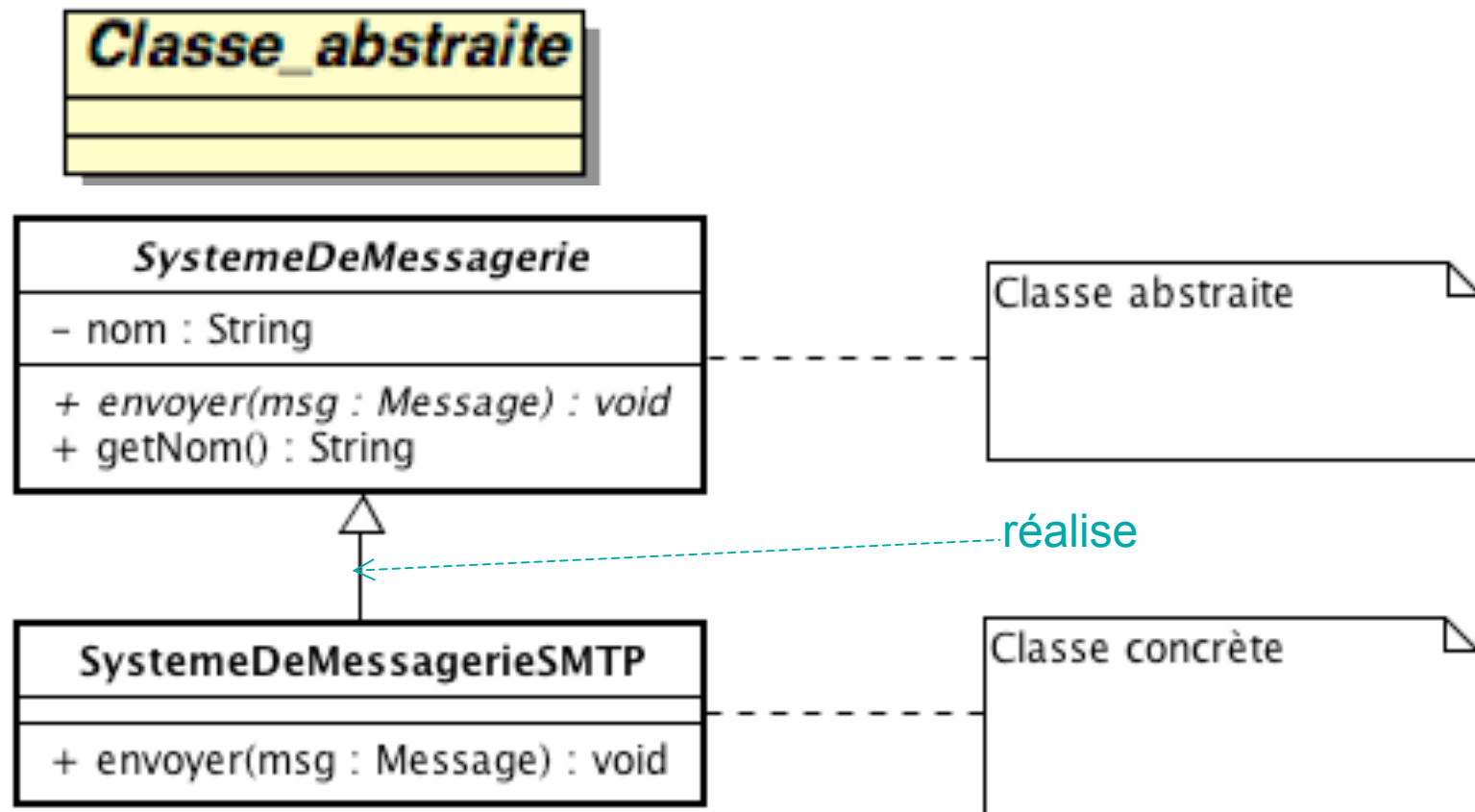
Généralisation multiple



Classe abstraite

- Une classe abstraite n'est pas instanciable directement
- Spécification plus abstraite pour des objets instances de ses sous-classes
- Elles forment une base pour des logiciels extensibles
- Une classe est désignée comme abstraite au moyen de la propriété booléenne **Abstraite** définie pour tous les éléments généralisables : les types, les paquetages, les stéréotypes.
- La propriété abstraite peut également être appliquée à une opération afin d'indiquer que le corps de l'opération doit être défini explicitement dans les sous-classes

Classe abstraite : notation



Interface

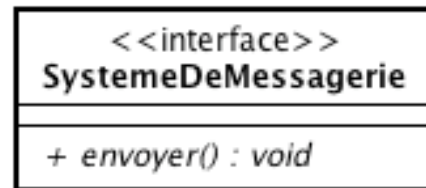
- Déclarer des méthodes que des classes concrètes doivent implémenter
- Eviter l'héritage

⇒ solution = l'interface

Mieux que l'héritage de classe abstraite : en Java, une classe peut implémenter (*implements*) n interfaces, $n \geq 0$, mais n'hérite (*extends*) que d'une classe directement

Une interface peut contenir des attributs : ils sont statiques, le plus souvent des constantes

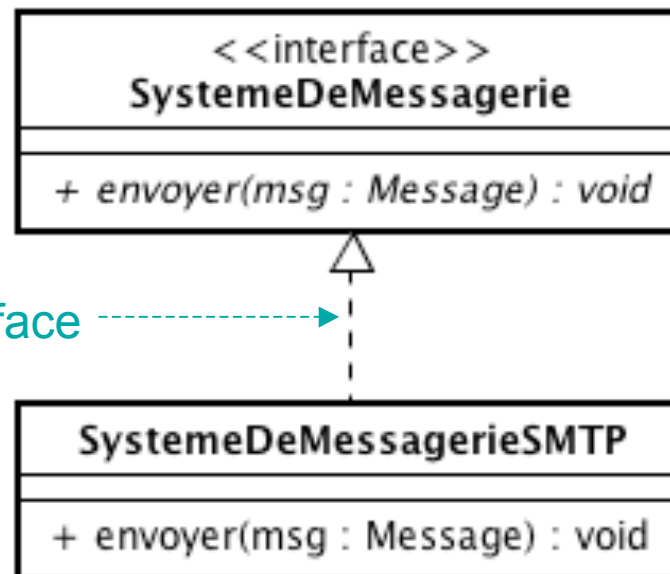
Interface : notation



Notation de stéréotype



Notation « à rotules »



Réalisation de l'interface →

Exemple de diagramme de classe

