Diagrammes de classes

Modéliser les vues statiques d'un système

Un diagramme de classes est une collection d'éléments de modélisation statiques (classes, paquetages...), qui montre la structure d'un modèle.

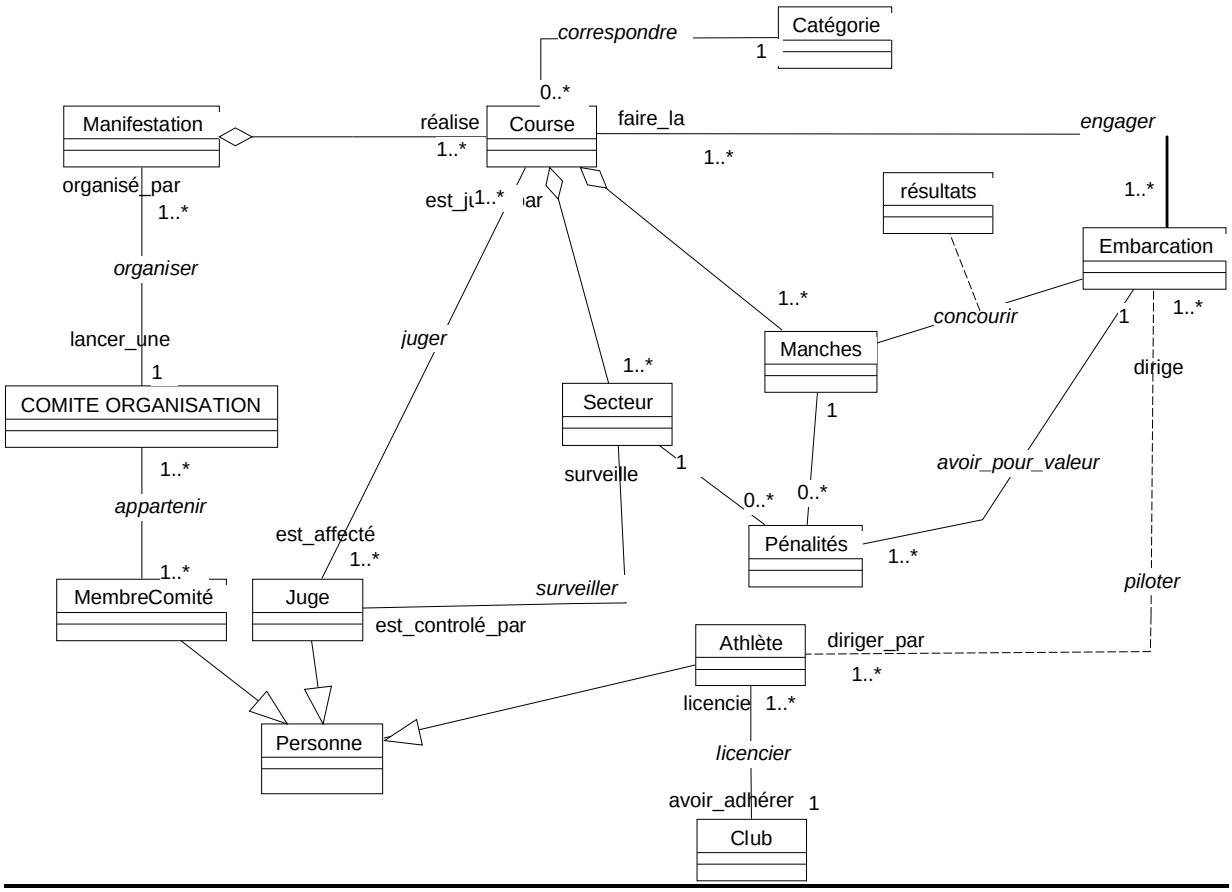
Un diagramme de classes fait abstraction des aspects dynamiques et temporels.

Pour un modèle complexe, plusieurs diagrammes de classes complémentaires doivent être construits.

On peut par exemple se focaliser sur :

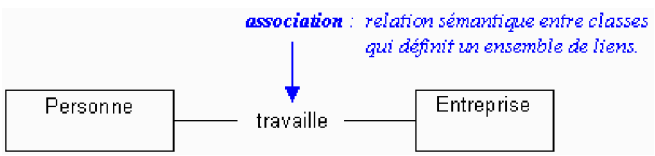
- les classes qui participent à un cas d'utilisation (voir collaboration),
- les classes associées dans la réalisation d'un scénario précis,
- les classes qui composent un paquetage,
- la structure hiérarchique d'un ensemble de classes.

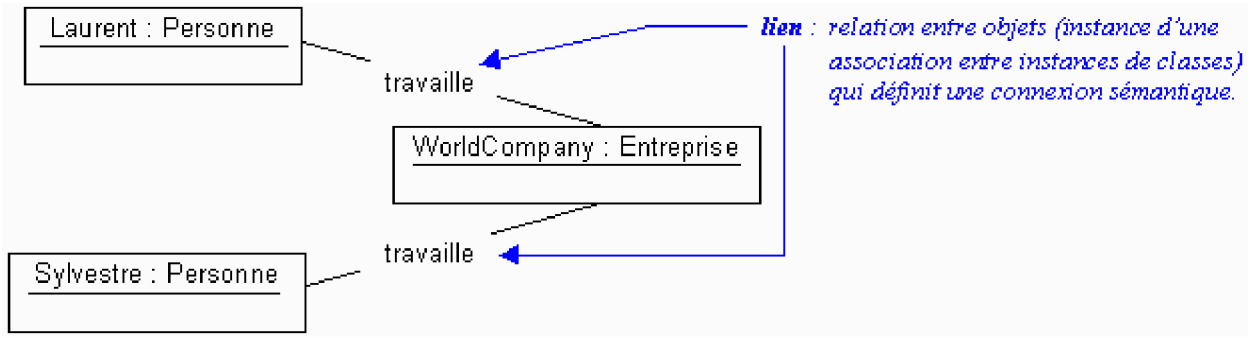
Pour représenter un contexte précis, un diagramme de classes peut être instancié en diagrammes d'objets.



Associations entre classes

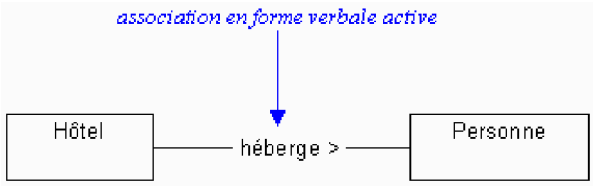
Une association exprime une connexion sémantique bidirectionnelle entre deux classes. L'association est instanciable dans un diagramme d'objets ou de collaboration, sous forme de liens entre objets issus de classes associées.



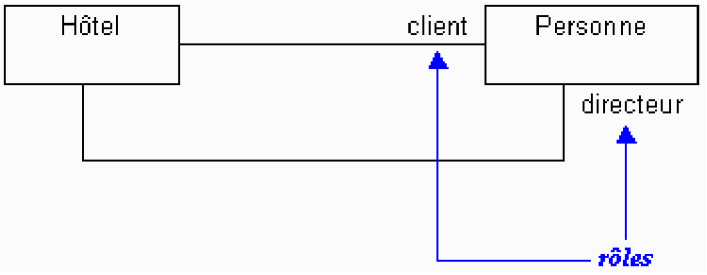


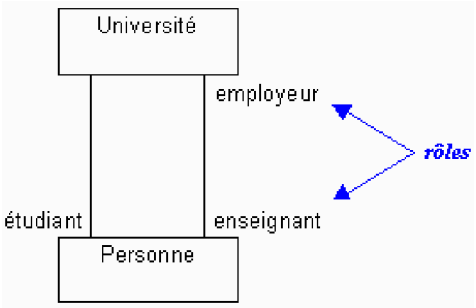
Documentation d'une association et types d'associations

Association en forme verbale active : précise le sens de lecture principal d'une association.

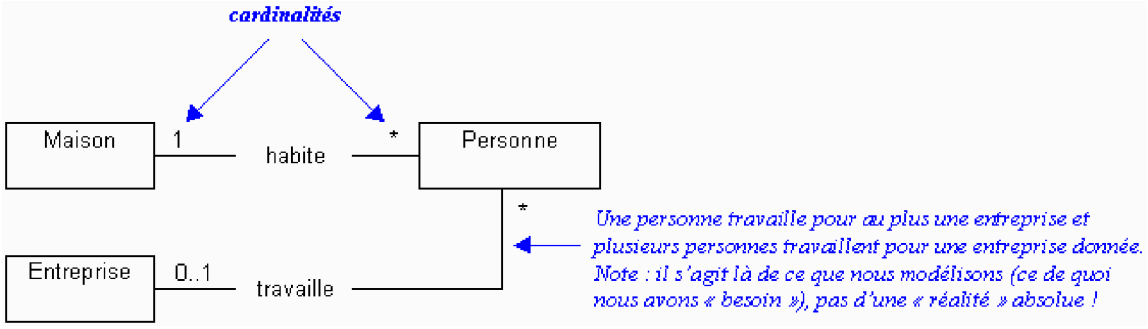


Rôles : spécifie la fonction d'une classe pour une association donnée (indispensable pour les associations réflexives).





Cardinalités : précise le nombre d'instances qui participent à une relation.



Expression des cardinalités d'une relation en UML :

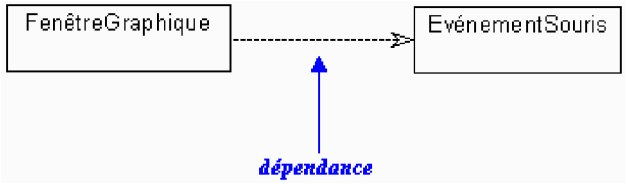
n : exactement "n" (n, entier naturel > 0)
exemples : "1", "7"

n..m : de "n" à "m" (entiers naturels ou variables, m ≥ n)
exemples : "0..1", "3..n", "1..31"

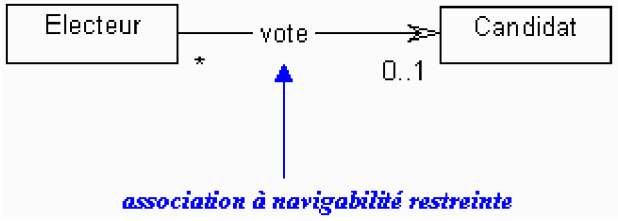
*** :** plusieurs (équivalent à "0..n" et "0..*")

n..* : "n" ou plus (n, entier naturel ou variable)
exemples : "0..*", "5..*"

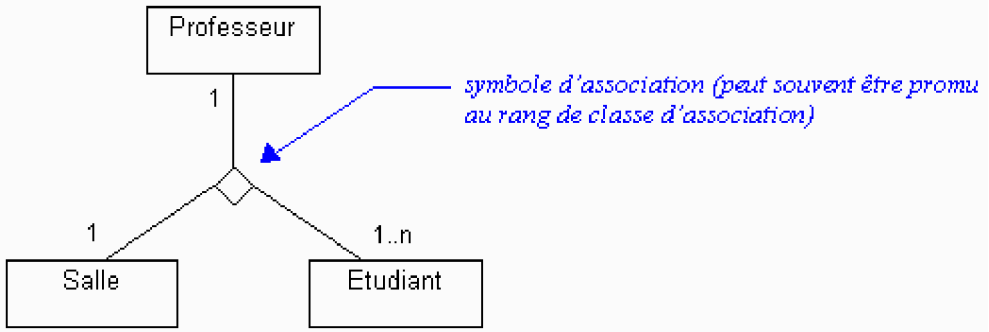
Relation de dépendance : relation d'utilisation unidirectionnelle et d'obsolescence (une modification de l'élément dont on dépend, peut nécessiter une mise à jour de l'élément dépendant).



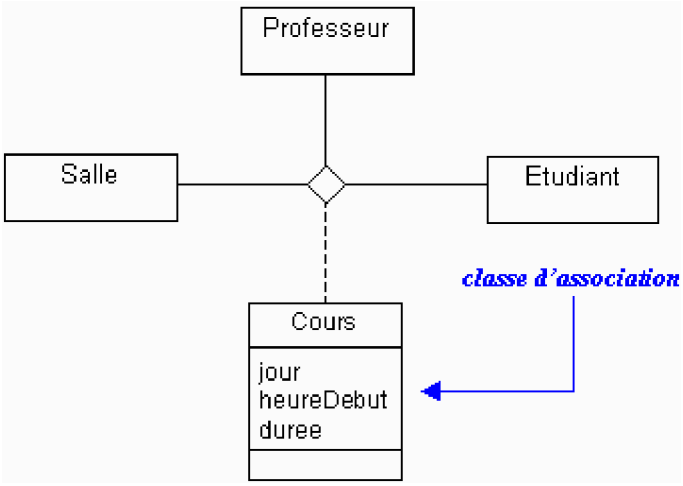
Association à navigabilité restreinte : Par défaut, une association est navigable dans les deux sens. La réduction de la portée de l'association est souvent réalisée en phase d'implémentation, mais peut aussi être exprimée dans un modèle pour indiquer que les instances d'une classe ne "connaissent" pas les instances d'une autre.

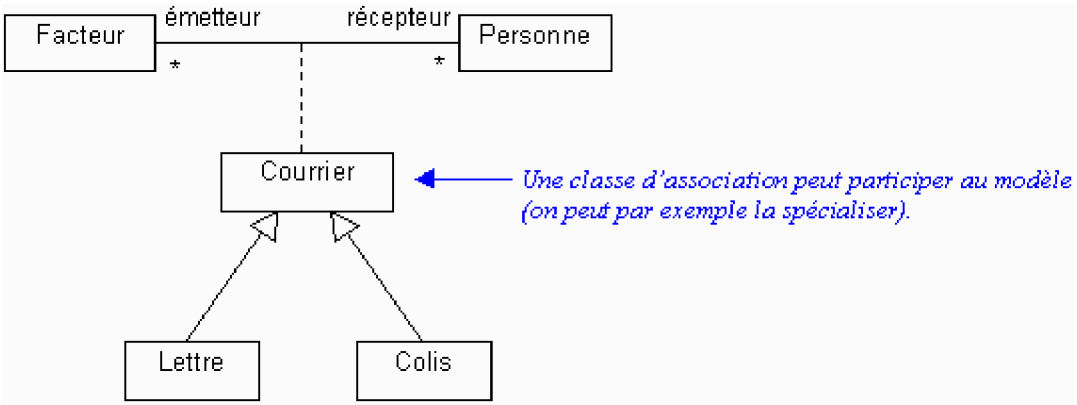


Association n-aire : il s'agit d'une association qui relie plus de deux classes...
Note : de telles associations sont difficiles à déchiffrer et peuvent induire en erreur. Il vaut mieux limiter leur utilisation, en définissant de nouvelles catégories d'associations.

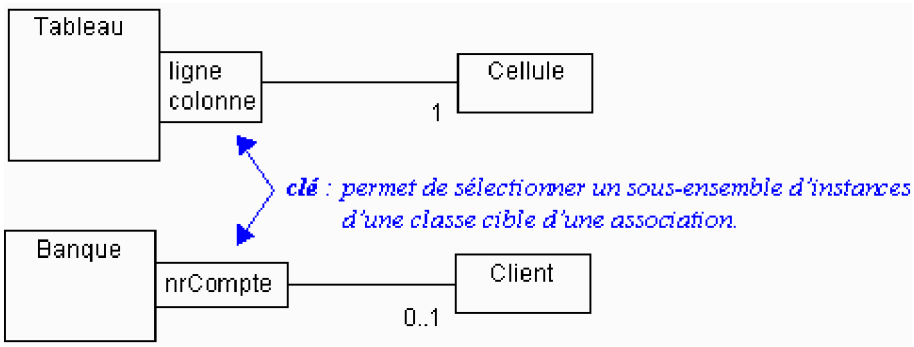


Classe d'association : il s'agit d'une classe qui réalise la navigation entre les instances d'autres classes.





Qualification : permet de sélectionner un sous-ensemble d'objets, parmi l'ensemble des objets qui participent à une association.
La restriction de l'association est définie par une clé, qui permet de sélectionner les objets ciblés.



Héritage.

Les hiérarchies de classes permettent de gérer la complexité, en ordonnant les objets au sein d'arborescences de classes, d'abstraction croissante.

Spécialisation : Démarche descendante, qui consiste à capturer les particularités d'un ensemble d'objets, non discriminés par les classes déjà identifiées.

Consiste à étendre les propriétés d'une classe, sous forme de sous-classes, plus spécifiques (permet l'extension du modèle par réutilisation).

Généralisation : Démarche ascendante, qui consiste à capturer les particularités communes d'un ensemble d'objets, issus de classes différentes.

Consiste à factoriser les propriétés d'un ensemble de classes, sous forme d'une super-classe, plus abstraite (permet de gagner en généricité).

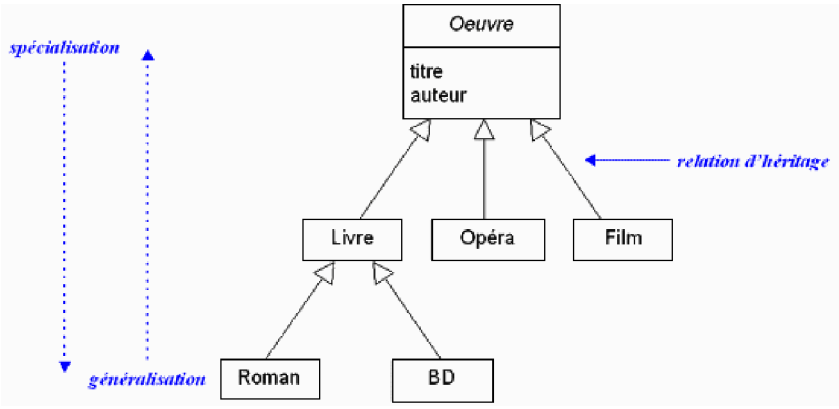
Classification : L'héritage (spécialisation et généralisation) permet la classification des objets.

Une bonne classification est stable et extensible : ne classifiez pas les objets selon des critères instables (selon ce qui caractérise leur état) ou trop vagues (car cela génère trop de sous-classes).

Les critères de classification sont **subjectifs**.

Le principe de substitution (Liksow, 1987) permet de déterminer si une relation d'héritage est bien employée pour la classification :
"Il doit être possible de substituer n'importe quel instance d'une super-classe, par n'importe quel instance d'une de ses sous-classes, sans que la sémantique d'un programme écrit dans les termes de la super-classe n'en soit affectée."

Si Y hérite de X, cela signifie que "Y est une sorte de X" (analogies entre classification et théorie des ensembles).



Agrégation

L'agrégation est une association non symétrique, qui exprime un couplage fort et une relation de subordination. Elle représente une relation de type "ensemble / élément".

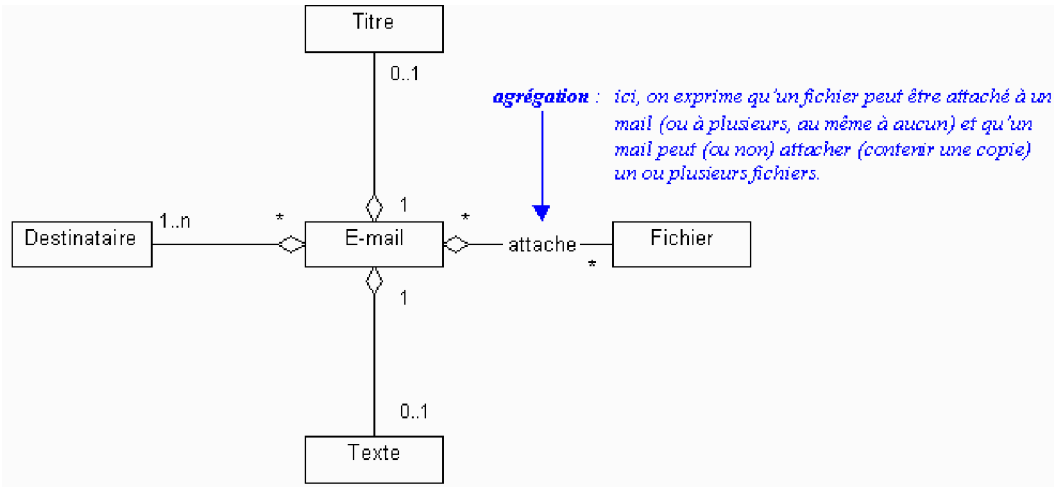
UML ne définit pas ce qu'est une relation de type "ensemble / élément", mais il permet cependant d'exprimer cette vue **subjective** de manière explicite.

Une agrégation peut notamment (mais pas nécessairement) exprimer :

- qu'une classe (un "élément") fait partie d'une autre ("l'agrégat"),
- qu'un changement d'état d'une classe, entraîne un changement d'état d'une autre,
- qu'une action sur une classe, entraîne une action sur une autre.

A un même moment, une instance d'élément agrégé peut être liée à plusieurs instances d'autres classes (l'élément agrégé peut être partagé).

Une instance d'élément agrégé peut exister sans agrégat (et inversement) : les cycles de vies de l'agrégat et de ses éléments agrégés peuvent être indépendants.



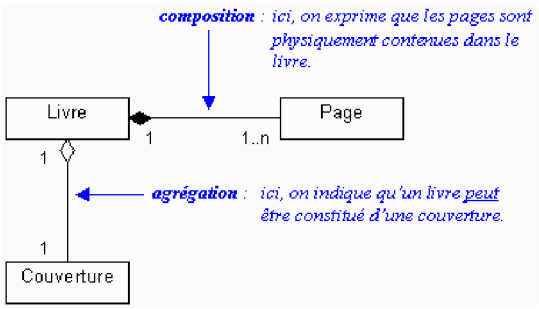
Composition

La composition est une agrégation forte (agrégation par valeur).

Les cycles de vies des éléments (les "composants") et de l'agrégat sont liés : si l'agrégat est détruit (ou copié), ses composants le sont aussi.

A un même moment, une instance de composant ne peut être liée qu'à un seul agrégat.

Les "objets composites" sont des instances de classes composées.



Agrégation et composition : rappel

L'agrégation et la composition sont des vues subjectives.

Lorsqu'on représente (avec UML) qu'une molécule est "composée" d'atomes, on sous-entend que la destruction d'une instance de la classe "Molécule", implique la destruction de ses composants, instances de la classe "Atome" (cf. propriétés de la composition).

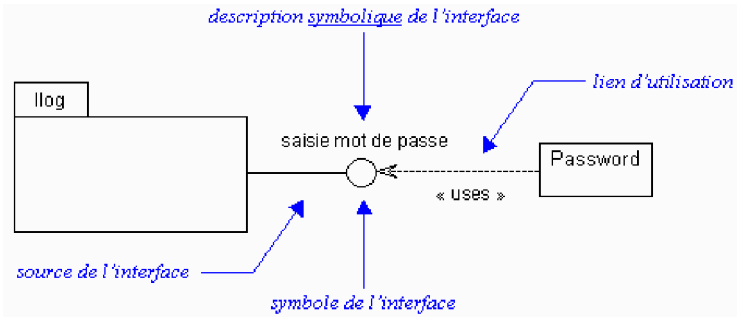
Bien qu'elle ne reflète pas la réalité ("rien ne se perd, rien ne se crée, tout se transforme"), cette abstraction de la réalité nous satisfait si l'objet principal de notre modélisation est la molécule...

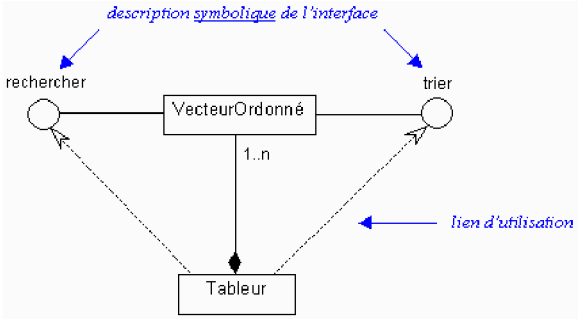
En conclusion, servez vous de l'agrégation et de la composition pour ajouter de la sémantique à vos modèles lorsque cela est pertinent, même si dans la "réalité" de tels liens n'existent pas !

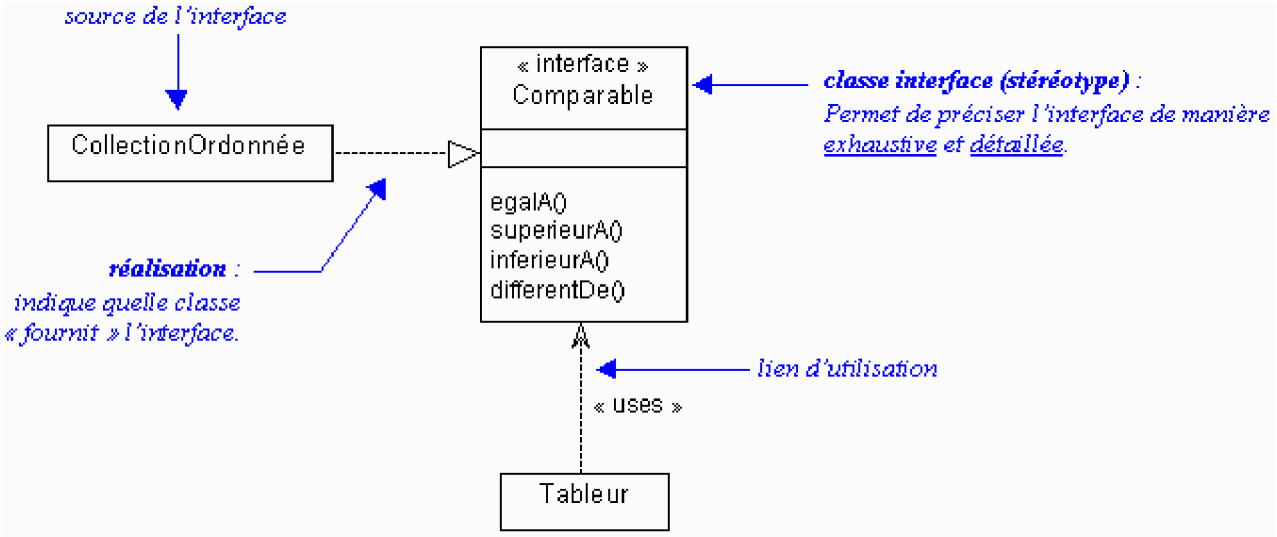
Interfaces

Une interface fournit une vue totale ou partielle d'un ensemble de services offerts par une classe, un paquetage ou un composant. Les éléments qui utilisent l'interface peuvent exploiter tout ou partie de l'interface.

Dans un modèle UML, le symbole "interface" sert à identifier de manière explicite et symbolique les services offerts par un élément et l'utilisation qui en est faite par les autres éléments.





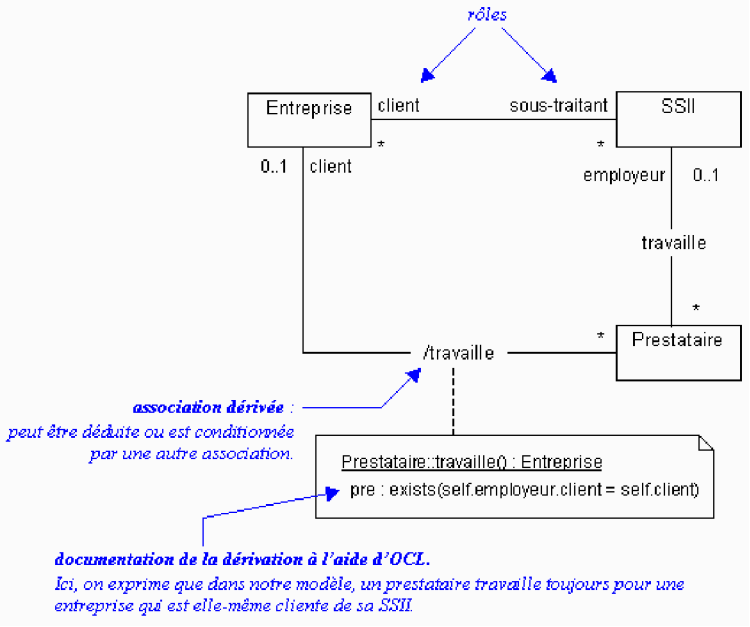


Association dérivée

Les associations dérivées sont des associations redondantes, qu'on peut déduire d'une autre association ou d'un ensemble d'autres associations.

Elles permettent d'indiquer des chemins de navigation "calculés", sur un diagramme de classes.

Elles servent beaucoup à la compréhension de la navigation (comment joindre telles instances d'une classe à partir d'une autre).

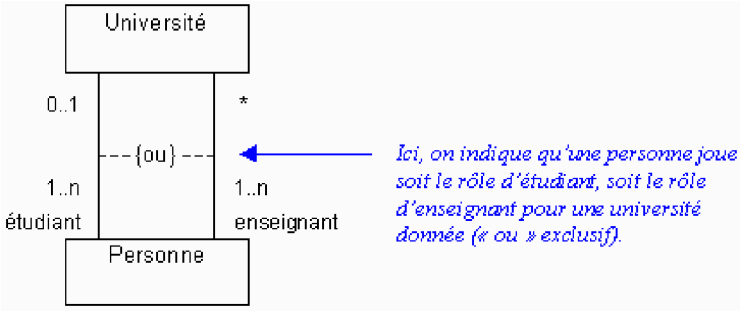


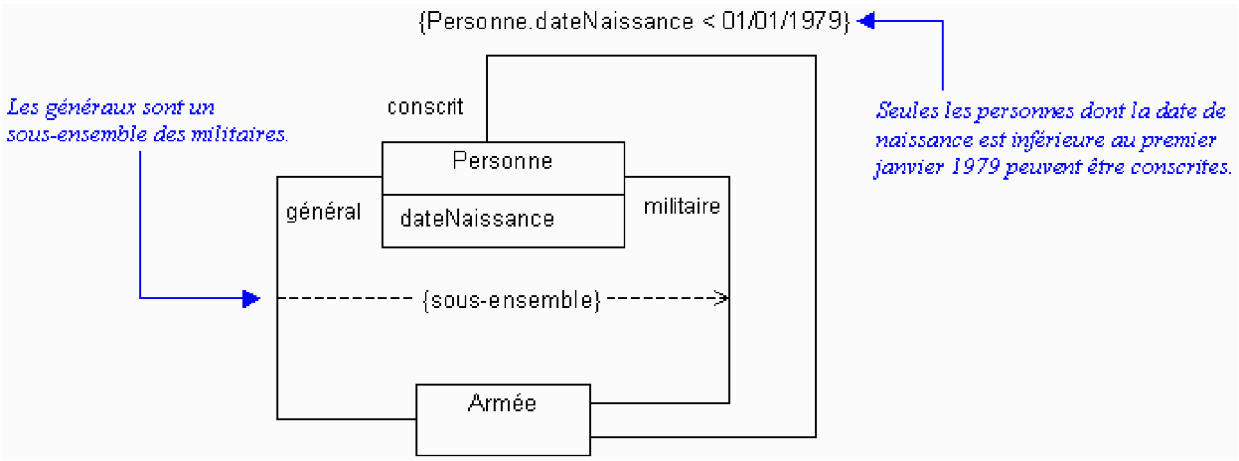
Contrainte sur une association

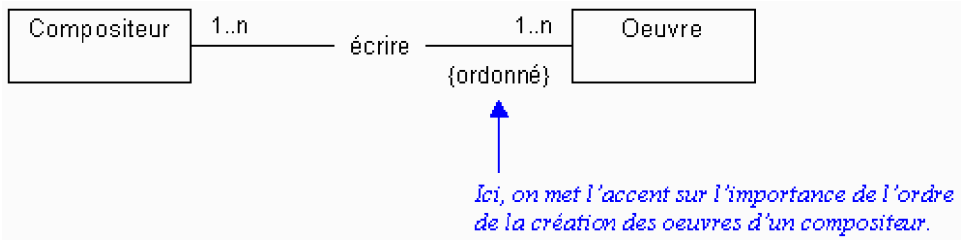
Les contraintes sont des expressions qui précisent le rôle ou la portée d'un élément de modélisation (elles permettent d'étendre ou préciser sa sémantique).

Sur une association, elles peuvent par exemple restreindre le nombre d'instances visées (ce sont alors des "expressions de navigation").

Les contraintes peuvent s'exprimer en langage naturel. Graphiquement, il s'agit d'un texte encadré d'accolades.







Le diagramme de classe est répertorié dans UML comme faisant partie du modèle statique. Le diagramme de classes peut contenir les abstractions suivantes :

- Classes
- Interfaces (partie visible d'un "contrat" "réalisable" par une instance. L'interface n'est pas instanciable mais la classe qui supporte l'interface l'est)
- Paquetages (packages)
- Relations
- Objets et les liens (nouveauté)

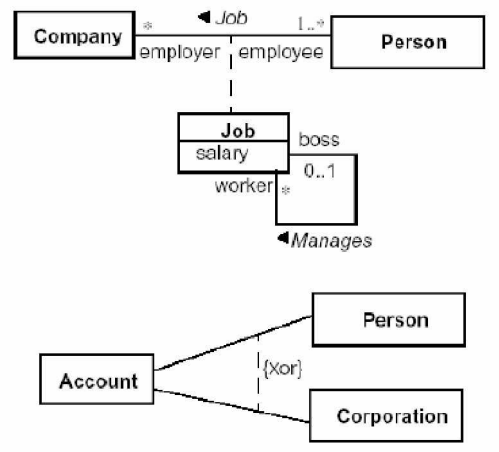
Associations entre les classes

Une association est une connexion entre les classes (navigable lorsqu'elle est orientée).

Quand deux classes sont impliquées dans une association, l'association est binaire, trois classes, ternaire, n classes, n-aire.

Une association binaire est tirée avec un trait plein. Les deux bouts de l'association sont des terminaisons (ends). Un trait pointillé sur un trait plein conduit à une classe d'association.

Chaque association a un nom qui précise la sémantique de l'association. Un triangle noir optionnel précise le sens de la lecture en présence d'une ambiguïté d'interprétation possible. Une contrainte peut s'ajouter avec sa paire d'accolades (exemple de OMG).

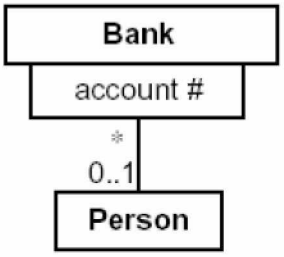


Chaque terminaison peut recevoir des spécifications de multiplicité et de rôles à la manière des diagrammes d'entités/relations que vous aviez vue dans le cours prérequis.

Qualificateurs (qualifiers)

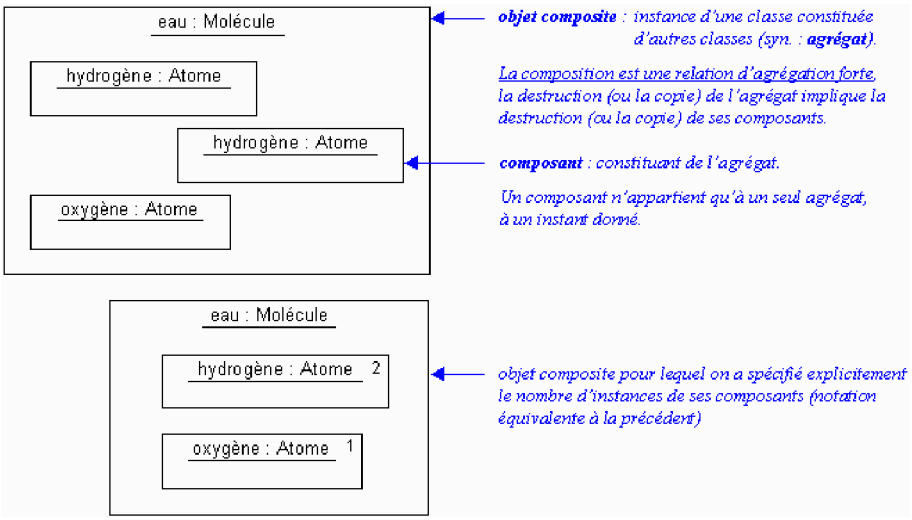
Un qualificateur est un attribut ou un groupe d'attributs qui servent à filtrer une population instanciable.

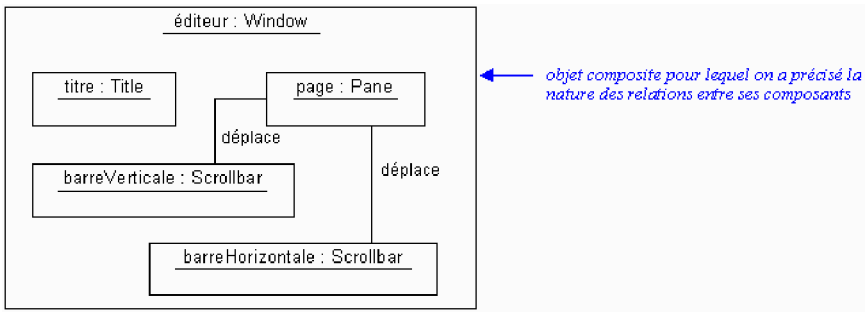
Un qualificateur caractérise l'association. Il est représenté par un petit rectangle sur la terminaison de l'association. (Exemple tiré de OMG-UML 1.4)



UML définit quelques associations particulières :

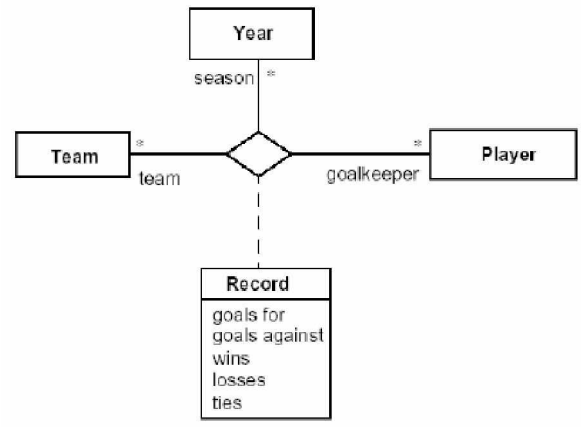
- Agrégation (petite décoration en forme de losange « non coloré » attaché à l'objet conteneur)
- Composition (petite décoration en forme de losange « coloré » attaché à l'objet conteneur). La composition est une forme d'agrégation « forte » de l'agrégation.





(Les exemples sont empruntés à OMG 1.4)

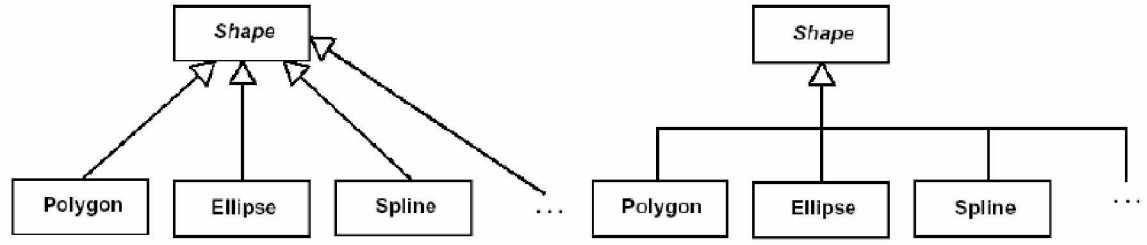
Représentation d'une relation n-aire (exemple dans OMG-UML 1.4)



Évitez dans la mesure du possible des relations ternaires ou n-aires. Décomposez si possible ces relations en relations binaires.

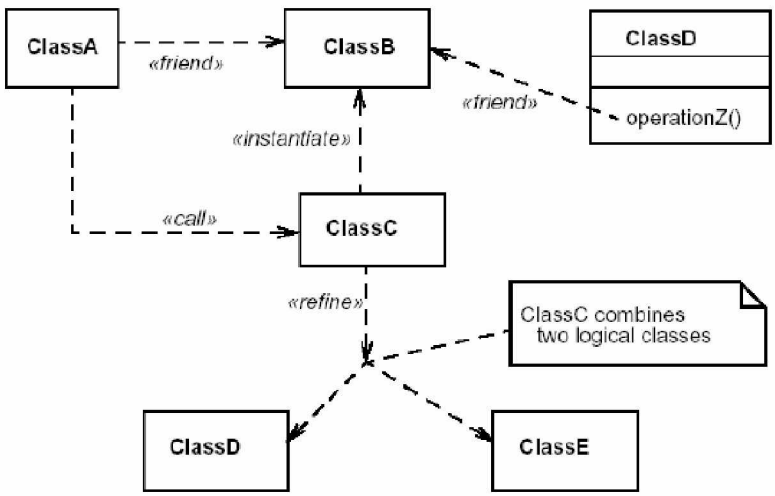
Représentation d'une relation généralisation/spécification/ et héritage/dérivation

Le triangle creux est du côté du parent.



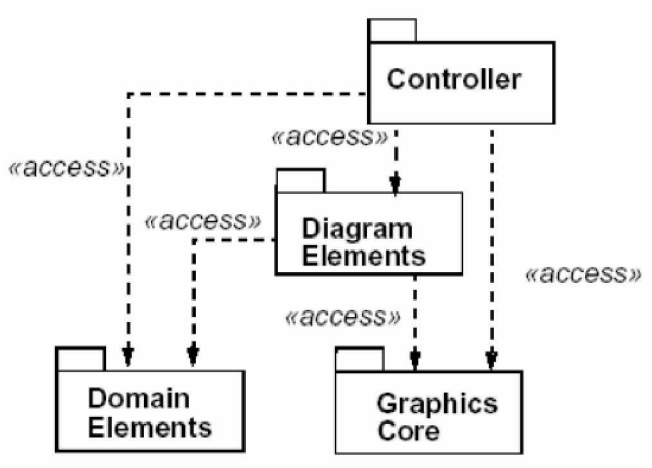
Représentation d'une dépendance (dependency)

Cette relation lie deux ou plusieurs classes ensemble. Une flèche en pointillé lie la classe « client » (queue de la flèche) à son « fournisseur » (pointe de la flèche).



Une modification du fournisseur affecte en principe tous les clients (exemple tiré de OMGUML1.4)

Dépendance entre paquetages



Les relations entre les classes.

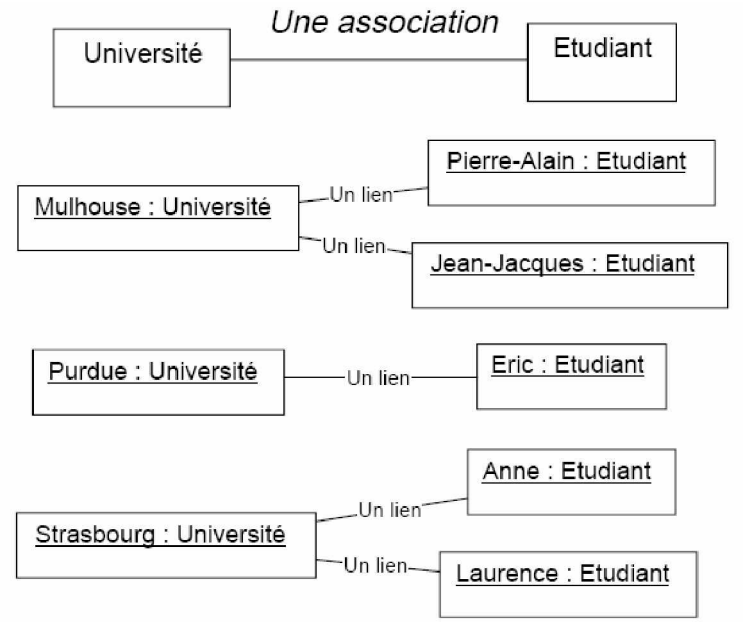
- L'association
- L'agrégation
- La généralisation

L'association

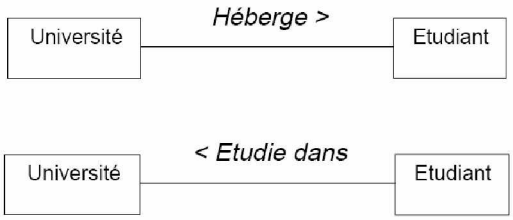
L'association exprime une connexion sémantique bidirectionnelle entre classes. C'est une abstraction des liens qui existent entre les objets instances des classes associées.

Les associations se représentent de la même manière que les liens mais la distinction opérée en fonction du contexte.

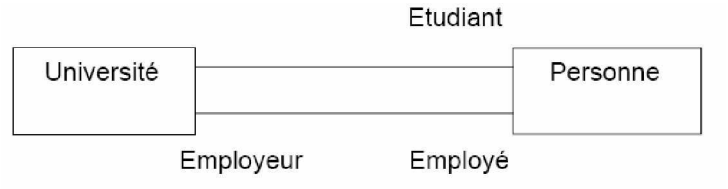
Exemple



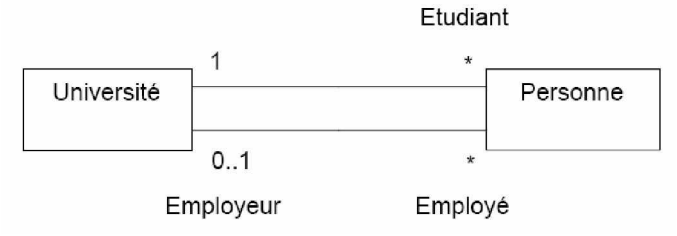
On nomme les associations par indication du sens de lecture.



Le rôle décrit une extrémité d'une association.



Les rôles peuvent être multiples.



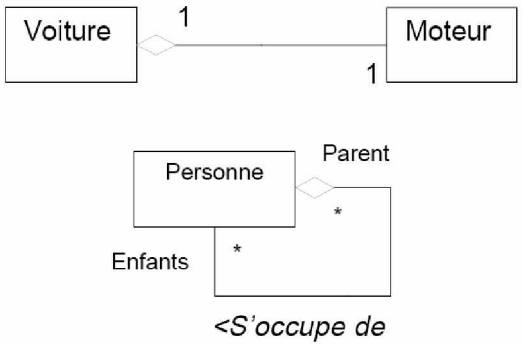
1 Un et un seul
0..1 Zéro ou un
M .. N De M à N (entiers naturels)
* Plusieurs
0 .. * De zéro à plusieurs
1 .. * D'un à plusieurs

L'aggrégation

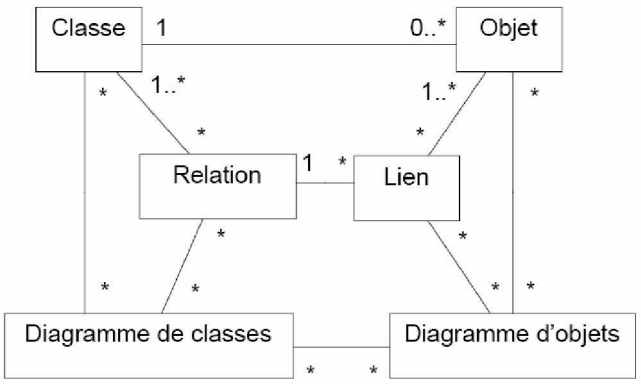
L'aggrégation est représentée par des connexions bidirectionnelles dissymétriques ou une forme d'association qui exprime un couplage plus fort entre classes. Les relations sont de types :

- maître et esclaves
- tout et parties
- composé et composant.

Exemples



Correspondances entre diagrammes

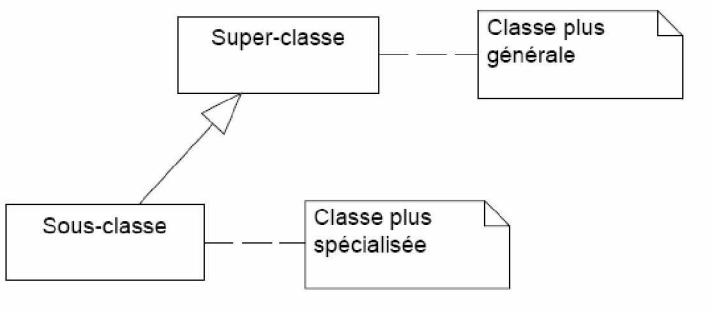


- Chaque objet est instance d'une classe
- Chaque lien est instance d'une relation

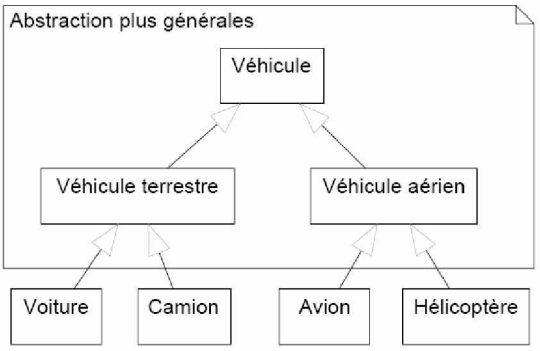
- Les liens relient les objets, les relations relient les classes
- Un lien entre deux objets implique une relation entre les classes des deux objets

Hiérarchies de classes

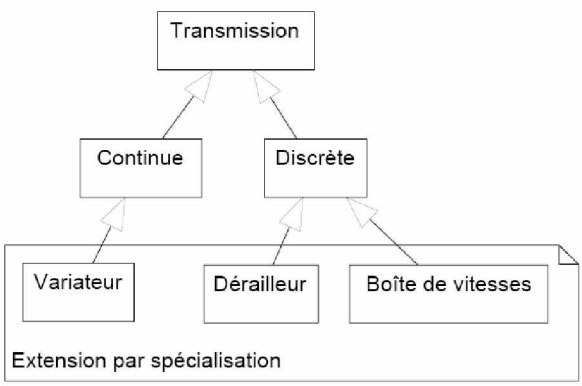
Pour gérer la complexité, on utilise une arborescence de classes d'abstraction croissante



Généralisation : Pour cela, on va factoriser les éléments communs : attributs, opérations et contraintes

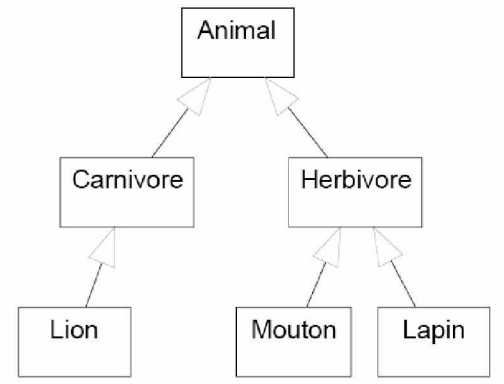


Spécialisation : Extension cohérente d'un ensemble de classes

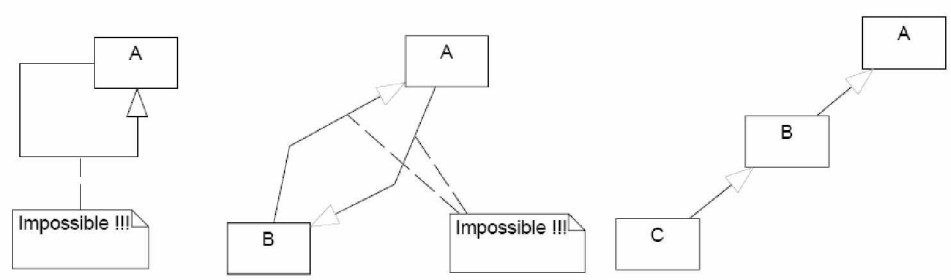


Propriétés de la généralisation

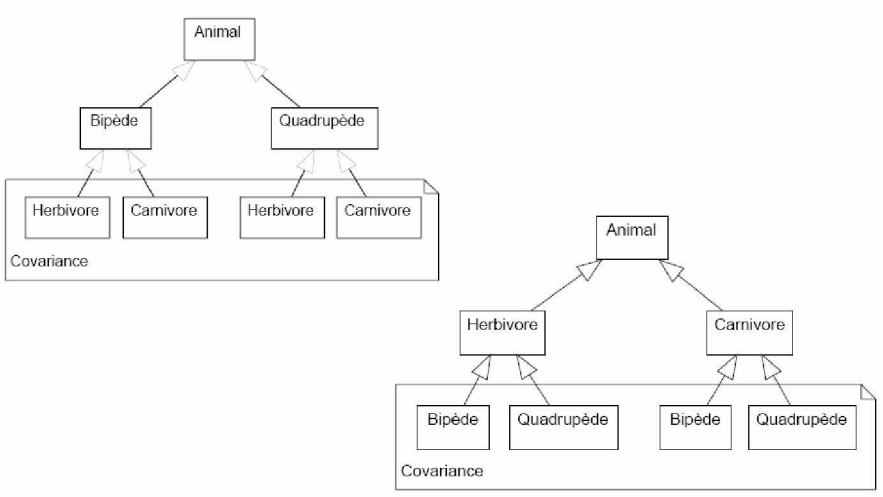
Signifie toujours : est un ou est une sorte de



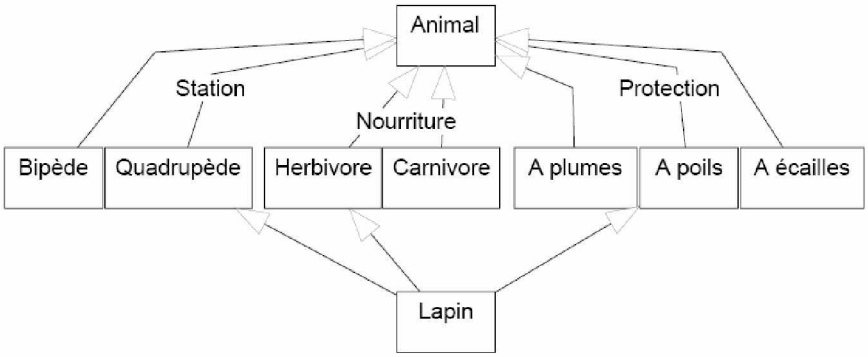
Non-réflexive, non-symétrique, transitive



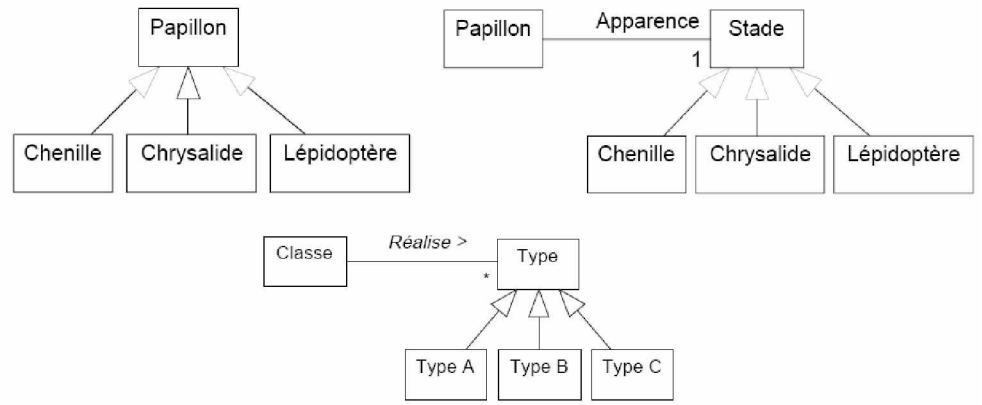
Covariance



Généralisation multiple



Mutation : La classe d'un objet est figée à la création

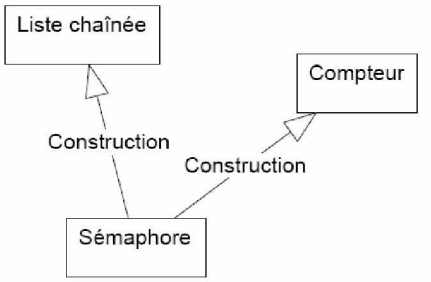


L'héritage

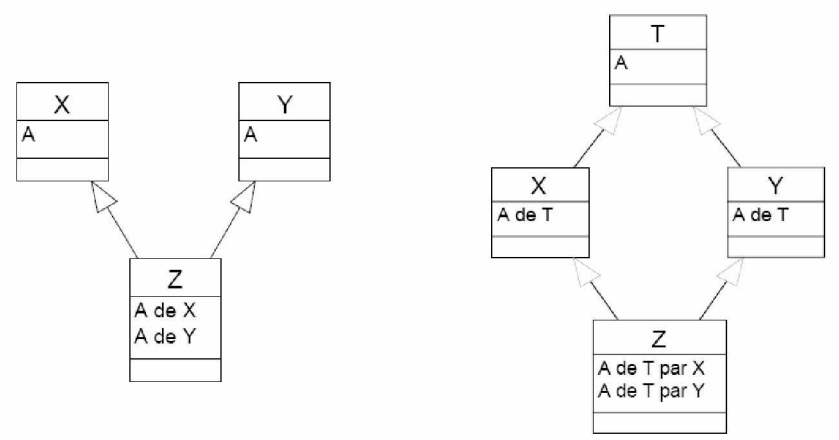
• Technique la plus utilisée pour réaliser la généralisation

• Construire une classe à partir d'une ou plusieurs autres classes, en partageant des attributs, des opérations et parfois des contraintes, au sein d'une hiérarchie de classes.

Danger avec l'héritage : Amalgame entre la notion de classification impliquée par l'héritage et la composition virtuelle qui en résulte

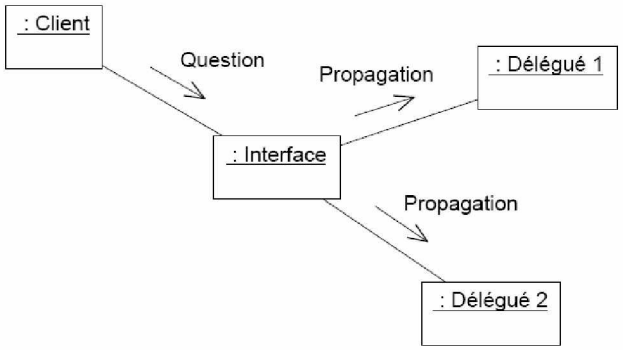


Collision de noms

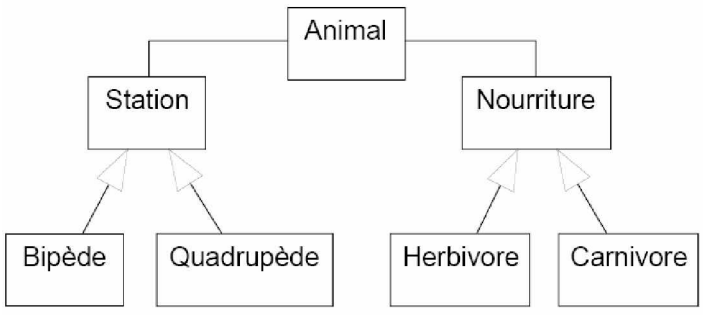


La délégation

Remplacement de l'héritage

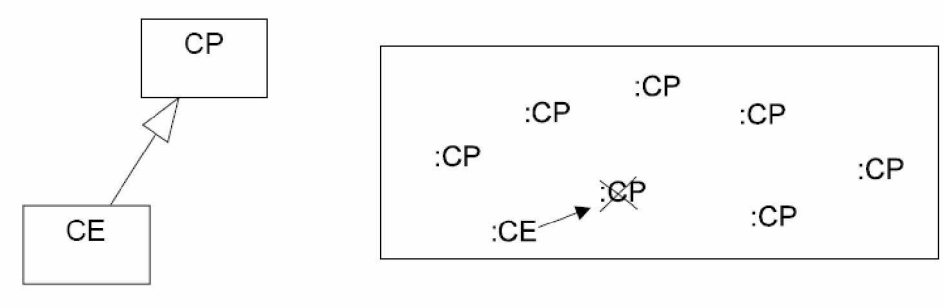


• Elimination de la covariance



Le principe de substitution

Classification ou non ? Il doit être possible de substituer n'importe quel objet instance d'une sous-classe à n'importe quel objet instance d'une super-classe sans, que la sémantique du programme écrit dans les termes de la super-classe ne soit affectée.

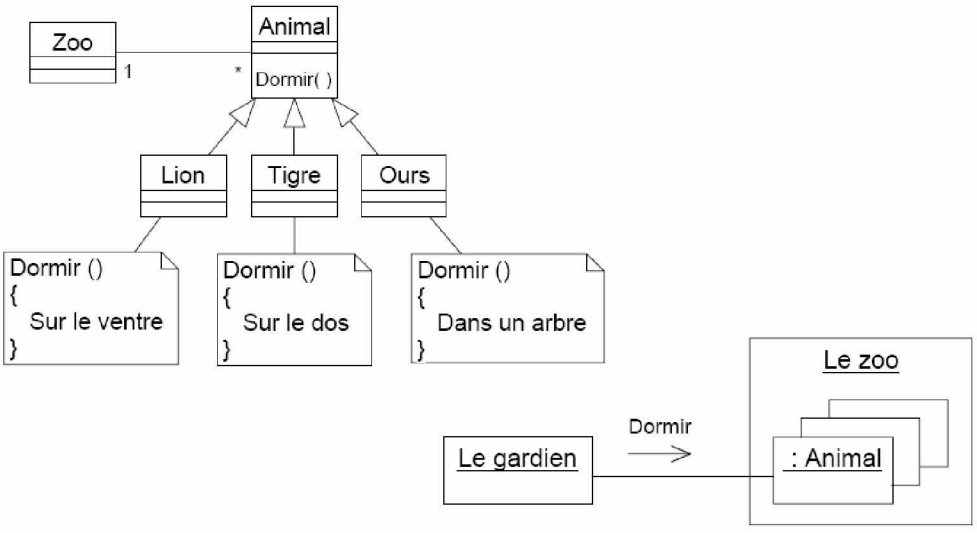


Polymorphisme

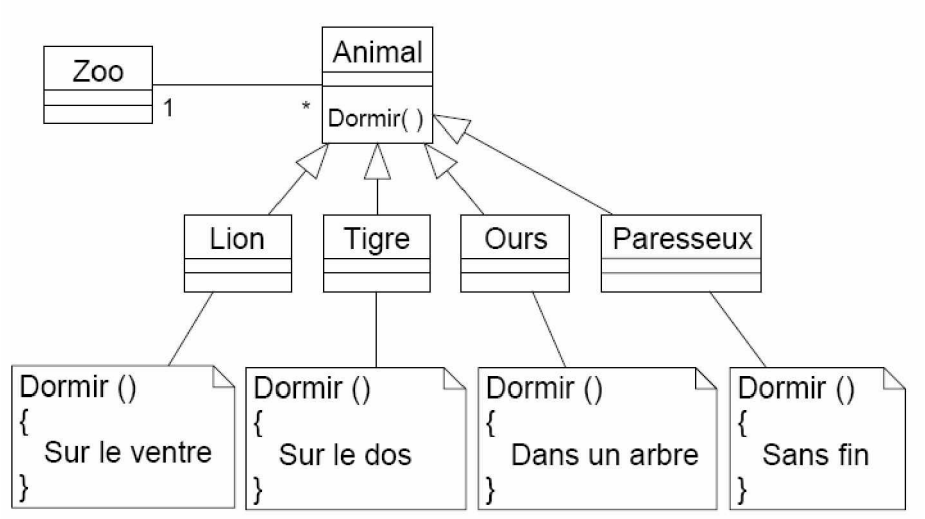
• Concept de la théorie des types, selon lequel un nom d'objet peut désigner des instances de classes différentes issues d'une même arborescence.

• Etroitement associé à l'interaction entre l'héritage et la liaison dynamique

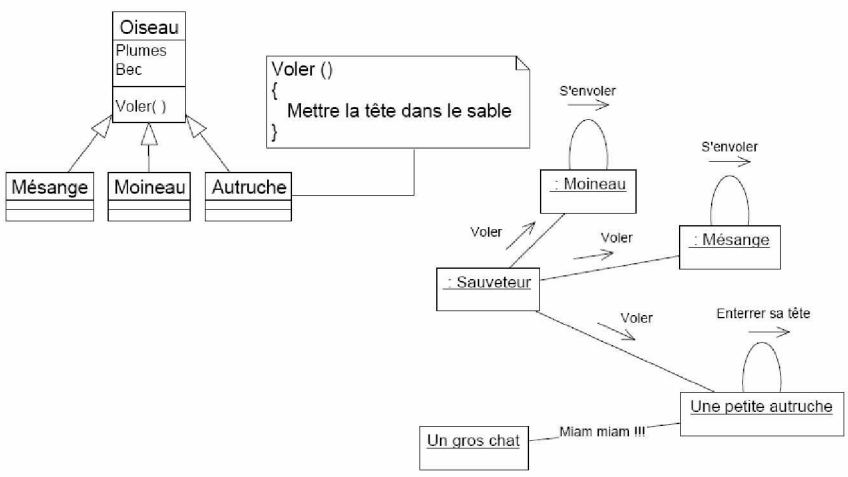
Application du polymorphisme



Extension du zoo



Non respect du principe de substitution



Conclusion

• Les classes sont connectées par des relations

• L'association exprime une connexion sémantique

• L'agrégation est une forme d'association plus forte

• La généralisation permet d'ordonner les objets au sein de hiérarchies de classes