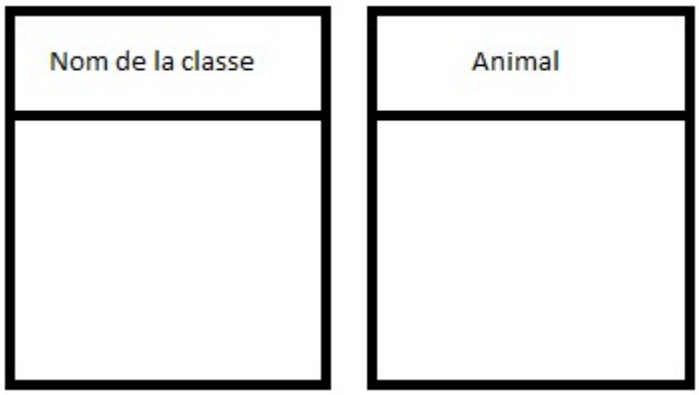
Diagramme de classe UML

Le diagramme de classes est un modèle statique. Il nous informe sur les relations structurales des différentes classes d'objets composant un système. On y décrit les structures et relations logiques des classes, mais pas leurs comportements.

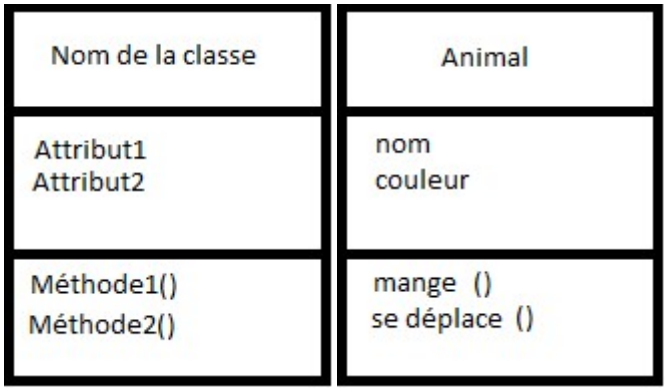
Que contient un diagramme de classe? Des classes et des associations.

Classes

On peut exprimer une classe en UML par un spectre de modèles allant du plus simple au plus complet. Voici la forme la plus simple: un rectangle avec un compartiment nommant la classe:

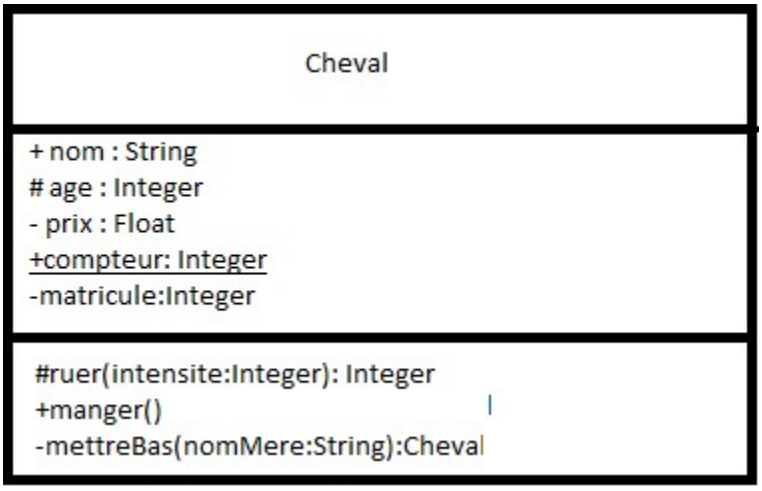


On peut ajouter les deux compartiments d'attributs et de méthodes:



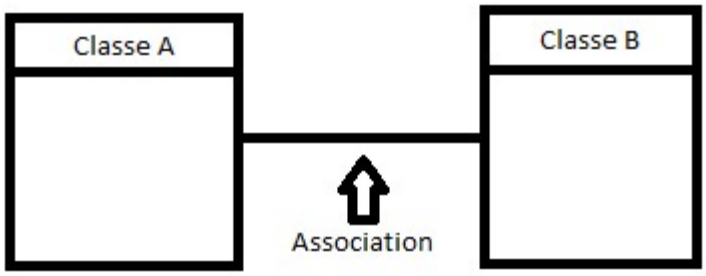
La forme complète décrit:

- Le niveau d'encapsulation (+ public # protected –private) des membres de la classe
- les membres statiques y sont soulignés
- les types sont indiqués à la droite des attributs et des méthodes (Integer, Float, String, Bool)

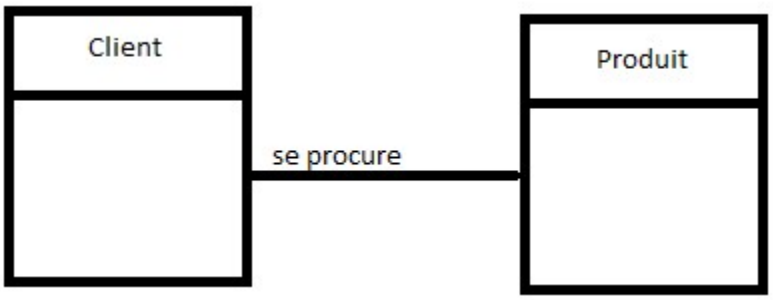


Associations

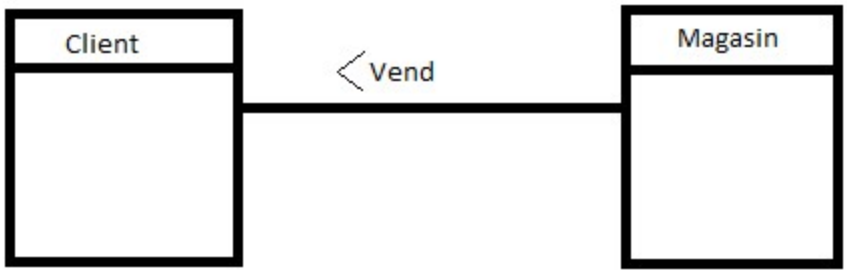
Dans le monde réel, des liens existent entre les objets. Ces liens en UML sont décrits par des **associations**. Un lien est une occurrence d'une association et cette occurrence relie entre elles des instances de ces classes.



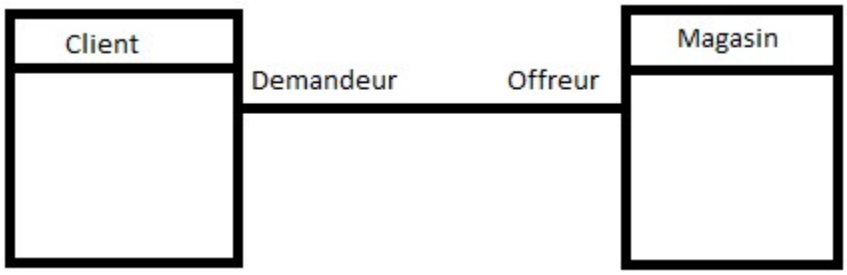
Une association porte un nom significatif:



On peut préciser le sens d'une association avec un > (ou un <) à la droite (ou à la gauche) du nom de l'association qui précise le sens de la lecture de l'association:

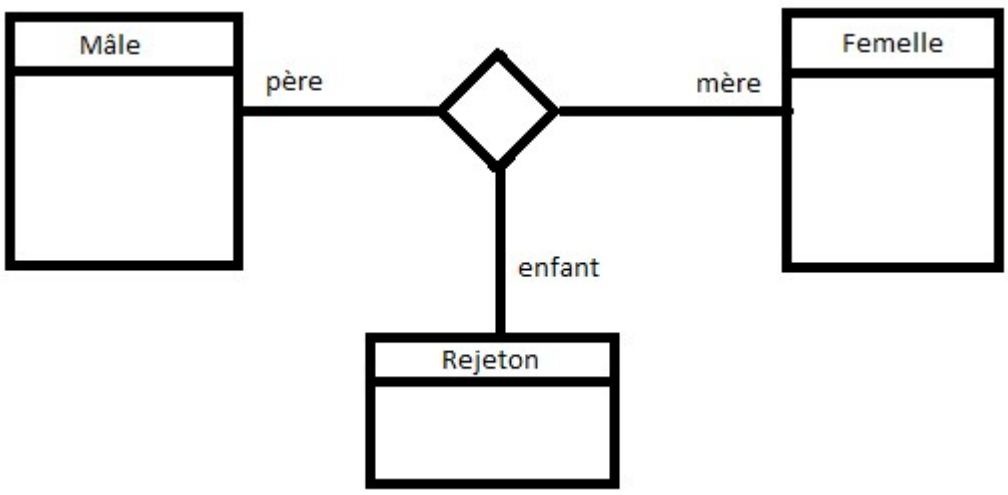


Les extrémités d'une association peuvent également être nommées. Le nom représente alors le rôle que joue l'instance de la classe dans l'association. Il devient parfois inutile de nommer l'association si chaque extrémité est nommée:



Association binaire : lien entre deux classes (très grande majorité)

Association Ternaire : lien entre trois classes (plus rare)



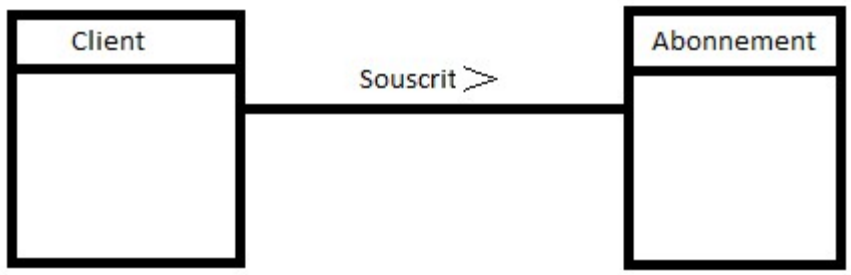
Association n-aire : lien entre n-classes (très rare)

Cardinalités des associations

Chaque extrémité d'une relation possède une cardinalité. Pour une association binaire il y a deux cardinalités : une par classe.

Pour une de ses instances données, la cardinalité de la classe exprime, le nombre d'instances de l'autre classe auxquelles elle sera liée. Pour la déterminer, il suffit de se placer dans la perspective d'une instance à l'autre extrémité de l'association.

Soit l'association suivante:

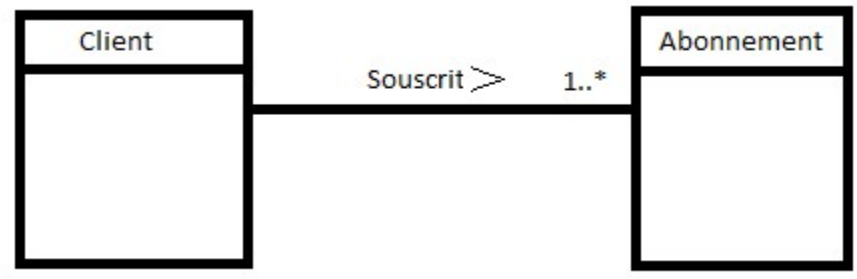


Ainsi, pour déterminer la cardinalité d'Abonnement posons la question suivante:

1- Quels sont le minimum et le maximum d'abonnements auxquels UN client peut souscrire?

Réponse: ça dépend toujours du contexte de modélisation, mais on peut poser l'hypothèse qu'un client doit souscrire à au moins un abonnement s'il veut se mériter le titre de client. Pour le maximum, il n'y a pas de limites à ses souscriptions. En terme UML on dirait: "un ou plusieurs" et ça se représente par: 1..*

Ainsi, UN client souscrit à un ou plusieurs abonnements:

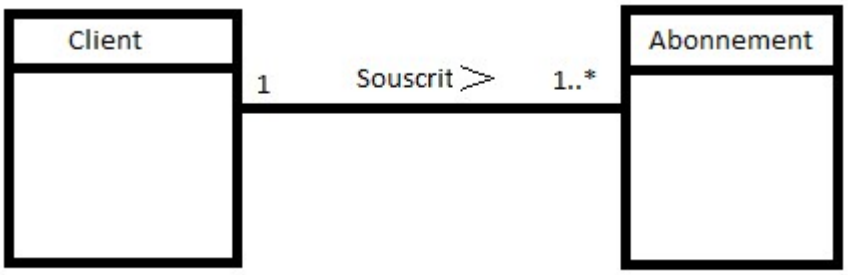


Pour l'autre extrémité, on procède à l'inverse:

2- Quels sont le minimum et le maximum de clients pouvant souscrire à UN abonnement?

Réponse: ça dépend de nos hypothèses. Ici on dira qu'un abonnement est souscrit par un seul client, jamais plus ni moins.

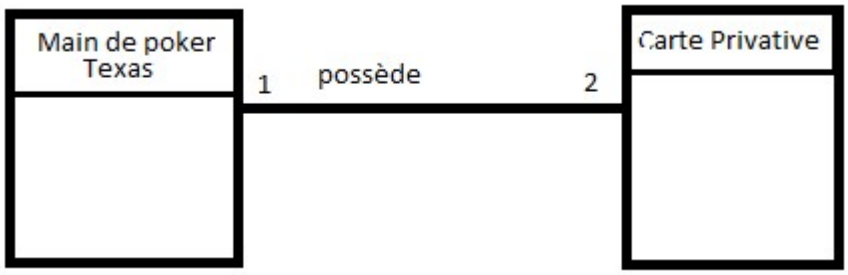
UN abonnement est souscrit par un client:

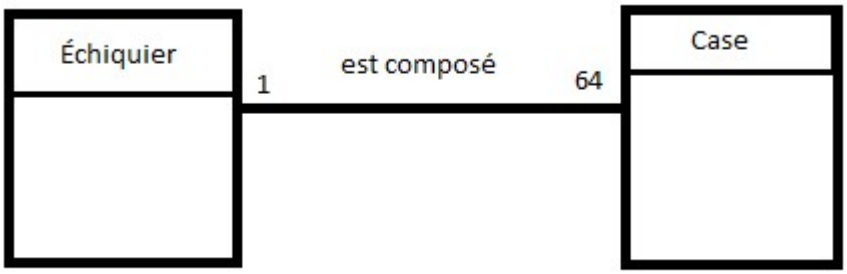


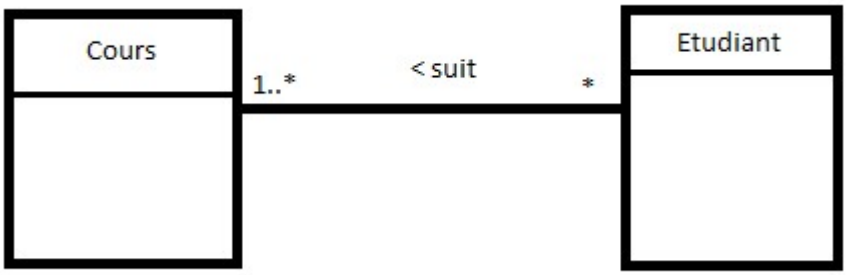
Voici un tableau des différentes combinaisons de cardinalité:

Spécification	Cardinalités
0..1	Zéro ou un
1	Un et seulement 1
*	De zéro à plusieurs
1..*	De 1 à plusieurs
M..N	Minimum M maximum N
N	N fois

D'autres exemples de cardinalités:

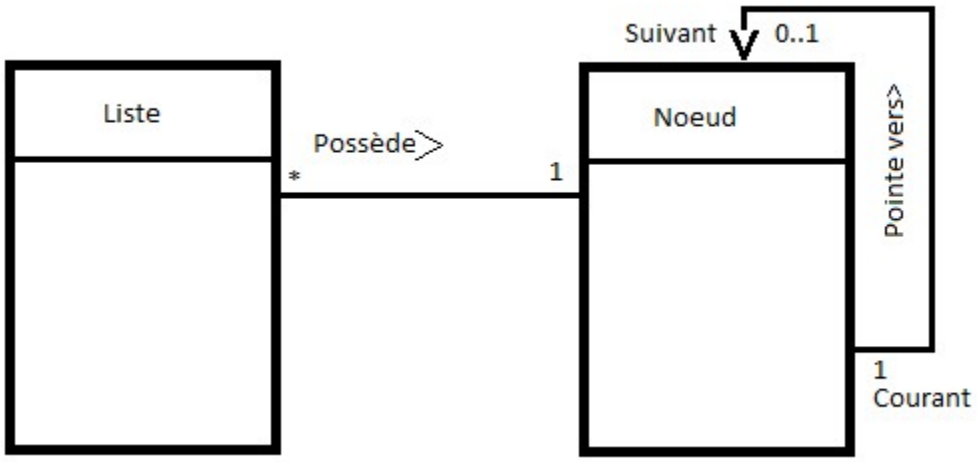






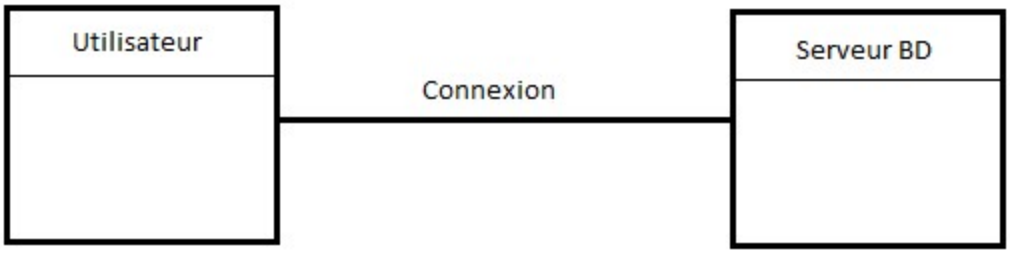
Association réflexive

Une classe peut avoir une association avec elle-même. On parle alors d'association réflexive. Dans ce cas il est souhaitable de nommer le rôle joué par la classe à chaque extrémité de l'association. L'exemple suivant montre la classe Noeud qui a une relation réflexive.

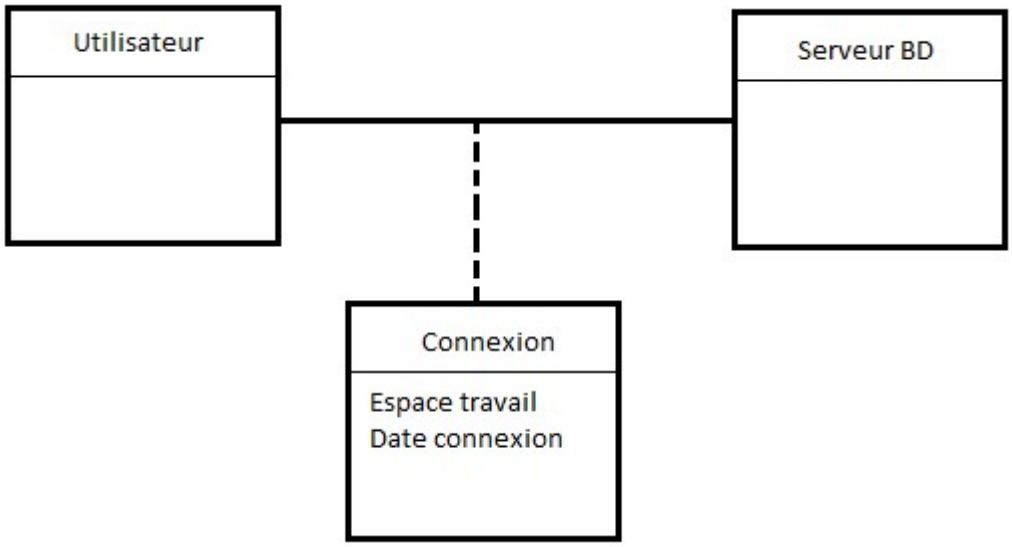


Classe Association

Quand un lien entre deux instances porte de l'information, cette association peut devenir une classe. Ainsi:



pourrait être représenté par



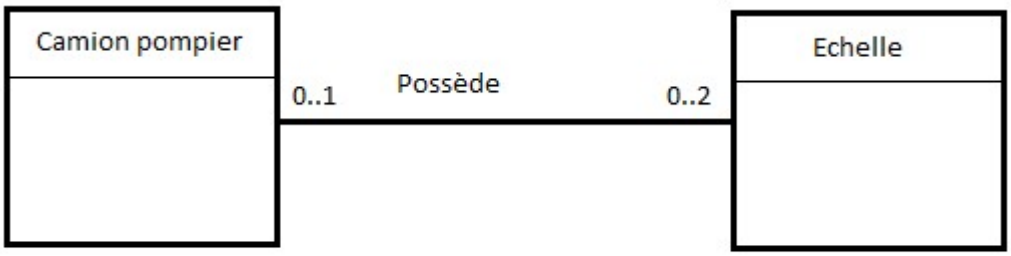
L'association connexion a ses propres attributs, elle devient une classe-association et elle est reliée à l'association par un pointillé.

Objet composé

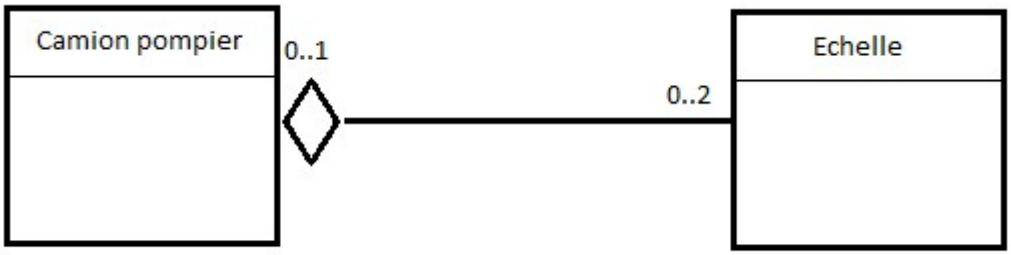
Les objets du monde réel se composent presque toujours d'autres objets. Un ordinateur est composé d'une foule de sous-composants, un escalier de marches, contremarches, rampes, etc. On appelle **composition** ce type d'association entre instances de classes. Si votre association s'appelle:

- possède
- a
- est composé

il s'agit fort probablement d'une association de composition.



On exprime la composition par un losange du côté de la classe composée.



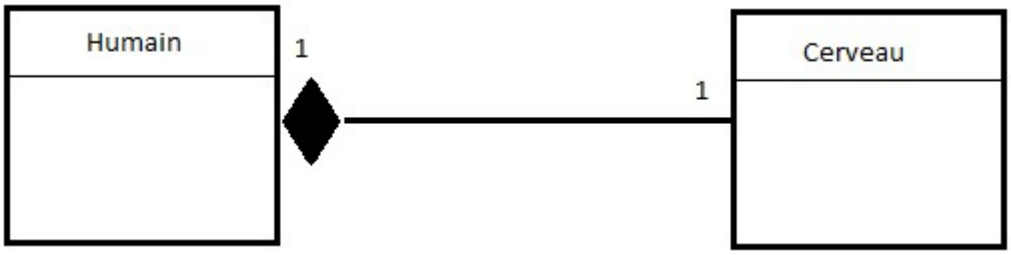
I

un camion de pompier possède 0 ou deux échelles.

Il existe deux types de composition: composition forte et composition faible.

Composition forte (composition)

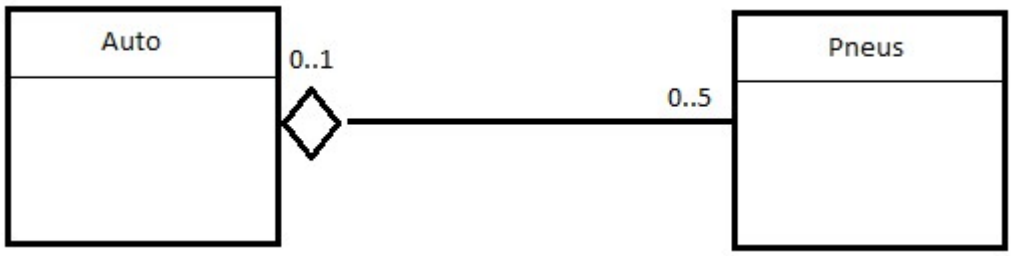
Les composants sont une partie de l'objet composé. Lorsque l'objet composé est détruit, les objets composants le sont également. La cardinalité maximale, au niveau de l'objet composé, est donc obligatoirement de 1.



Quand l'humain cesse son existence, le cerveau cesse la sienne également.

Composition faible (agrégation)

Si les composants peuvent être réutilisés dans d'autres instances, il s'agit d'agrégation, ou composition faible. Leurs vies ne se terminent pas lorsque la vie de l'objet composé se termine.



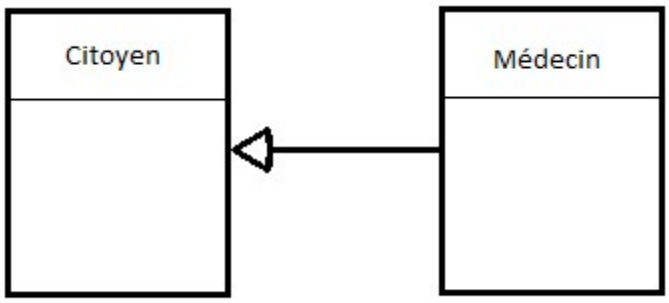
L'agrégation est plus fréquente que la composition.

	Agrégation	Composition
Représentation	Losange blanc	Losange noir
Partage des composants par plusieurs associations	oui	Non
Destruction des composants lors de la destruction du composé	Non	Oui
Cardinalité au niveau du composé	Quelconque	0..1 ou 1

On peut au départ considérer toutes compositions comme agrégation, car l'agrégation a moins de contraintes. Ajouter des contraintes signifie raffiner notre modèle, l'enrichir.

Généralisation / Spécialisation

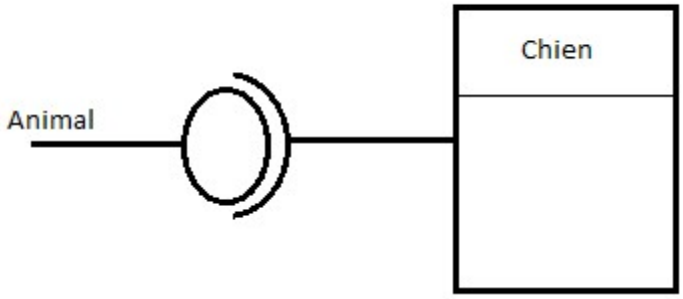
Quand une classe en spécialise une autre, on parle d'héritage. Cette association de spécialisation entre deux classes se symbolise par une flèche fermée:



Un médecin est une forme spécialisée de citoyen, mais il est aussi un citoyen.

Les interfaces

Une interface est une classe totalement abstraite : elle n'a pas d'attribut et ses méthodes sont toutes publiques et abstraites. L'implémentation des méthodes est réalisée par les classes concrètes, sous-classes de l'interface. La relation d'héritage dans ce contexte est appelée « relation de réalisation » et on la représente par le "lollipop". On dit de la classe qui hérite d'une interface qu'elle réalise l'interface.



Dans l'exemple précédent, la classe Chien hérite de la classe Animal. Animal est une classe abstraite, une interface. La classe Chien réalise l'interface Animal.