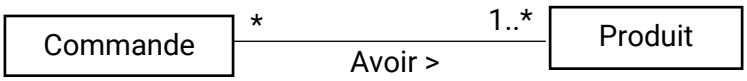
Diagramme de classes

Associations particulières

Association

La navigabilité

Soit les deux classes suivantes Commande et Produit reliée par l'association Avoir :



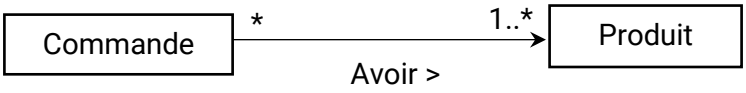
Lors de l'implémentation, c'est-à-dire, lors du développement de l'application, et selon cette association **Avoir**, nous devons trouver un attribut de type Produit dans la classe Commande et dans la classe Produit, nous devons trouver un attribut de type Commande. Dans la classe Commande nous avons l'attribut RefProfduit de type Produit. Dans la classe Produit nous avons l'attribut RefCommande de type Commande.

Ainsi, à partir d'une instance de la classe Commande on peut accéder aux instances de la classe Produit qui lui sont associés. Et inversement, à partir d'une instance de la classe Produit on peut accéder aux instances de la classe Commande qui lui sont associés.

Supposons que nous souhaitons connaitre **que** les produits d'une commande. C'est-à-dire, pour une commande définie, nous voulons savoir ses produits commandés. Nous nous n'intéressons pas aux commandes d'un produit. Dans ce cas-là, et dans la phase d'implémentation, il faut ajouter **juste** l'attribut, par exemple, RefProfduit de type Produit dans la classe Commande. On n'ajoute pas l'attribut RefCommande de type Commande dans la classe Produit.

On aura dans l'implémentation les deux classes suivantes :

Pour représenter graphiquement cela dans la phase de modélisation, nous utilisons un lien fléché de la classe Commande vers la classe Produit comme indique dans l'association suivante :

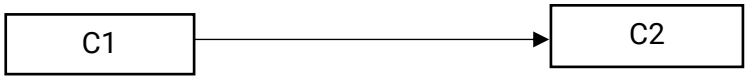


Cette façon de modélisation d'association s'appelle **Navigabilité d'une association** ou **Navigation d'une association**.

Qu'est-ce que la navigabilité d'une association entre une classe C1 et une classe C2 ?

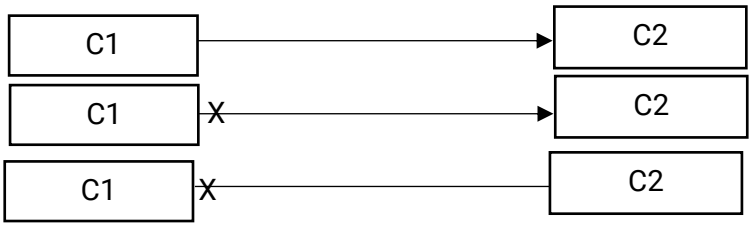
La navigabilité, est exprimée par une flèche dans un seul sens, et qui traduit la capacité d'une instance de la classe C1 à accéder **aux** instances de la classe C2.

Cette navigabilité entre C1 et C2 est représentée par un lien fléché vers C2 :



Une autre façon de représenter cette navigabilité entre C1 et C2 est d'ajouter une croix à côté de la classe C1. La croix indique la non navigabilité de C2 vers C1.

Ainsi ces trois navigabilités représentent la même chose :

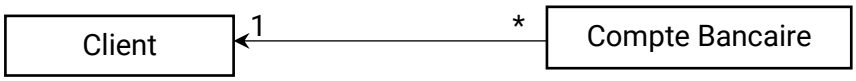


N.B : Par défaut, les associations sont navigables dans les deux sens.

Attention, ne pas confondre le sens de l'association avec la navigabilité de l'association.

Exemple contenant qu'une association avec un sens. La flèche pointe sur Client pour traduire el sens de lecture de l'association, qu'un compte appartient à un client.

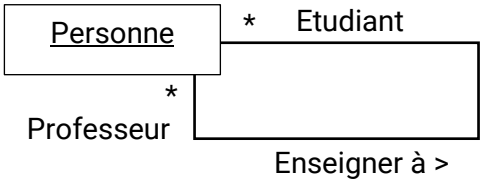
Exemple suivant, contenant la navigabilité, exprimée par la flèche pointe sur la classe Client traduisant qu'à partir d'une instance de la classe Compte on accède à son titulaire, instance de la classe Client.



Associer une classe à elle-même ou association réflexive

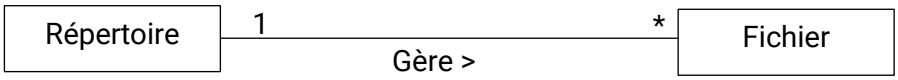
Une association est lien entre deux classes. Cependant, dans le cas d'un lien avec la classe elle-même, on parle d'une association réflexive. Dans ce genre d'association, il est préférable de préciser les rôles à chaque extrémité de l'association.

Dans cet exemple, une personne dont le rôle Enseignant enseigne à plusieurs personnes dont le rôle est étudiant.



Qualification d'association ou association qualifiée

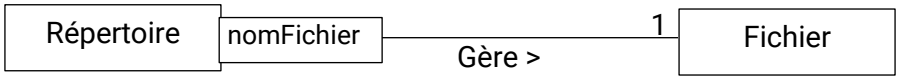
Soit l'exemple suivant :



nomFichier est un attribut de la classe Fichier.

Grâce à l'attribut *nomFichier, de la classe Fichier*, une instance de la classe Répertoire accède à une instance de la classe Fichier.

La modélisation de ce que nous venons de dire peut-être représentée dans le diagramme de classe par :



On parle ainsi d'une qualification d'une association. L'attribut *nomFichier* à côté de la classe Répertoire s'appelle un qualificatif.

La qualification donne par exemple une indication pour l'élection des clefs lors de la construction du schéma de la base de données.

Vous remarquez que la cardinalité ou la multiplicité côté la classe est devenu 1 et non * (plusieurs). Cela explique, qu'une instance de la classe Répertoire accède à une seule instance de la classe Fichier grâce à son nom.

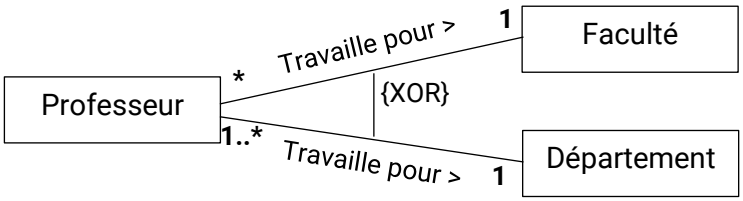
Contraintes

Il existe en UML quelques contraintes définies sur les associations.

Les contraintes sur les associations sont des informations qui s'ajoutent à votre diagramme de classes. Les contraintes sont mises entre accolades {} et elles sont exprimées soit dans le langage OCL (***Object Constraint Language***) ou dans un langage naturel.

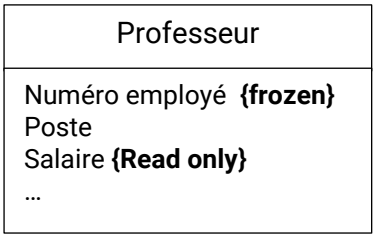
Voici des exemples de contraintes: {XOR}, {frozen} {ordered} etc.

La contrainte {XOR} qui est un ou exclusif signifie qu'une classe utilise une association ou l'autre mais pas les deux. L'exemple suivant montre qu'un Professeur peut travailler pour une Faculté ou un Département mais pas pour les deux.



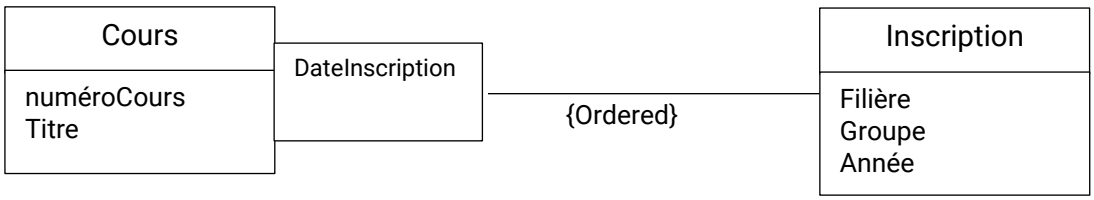
Il y a aussi le {frozen} qui signifie que l'association, la classe ou l'attribut, une fois créé ne changera jamais. Le {Read only} qui signifie que la valeur peut changer à l'intérieur de la classe mais ne peut être changée de l'extérieur de la classe, ce qui est différent d'une constante qui ne pourrait jamais changer.

Dans l'exemple suivant, on trouve les 2 contrainte. Le numéro d'un employé, une fois créé, ne peut changer et le salaire, ne peut être modifié à l'extérieur de la classe Employé.



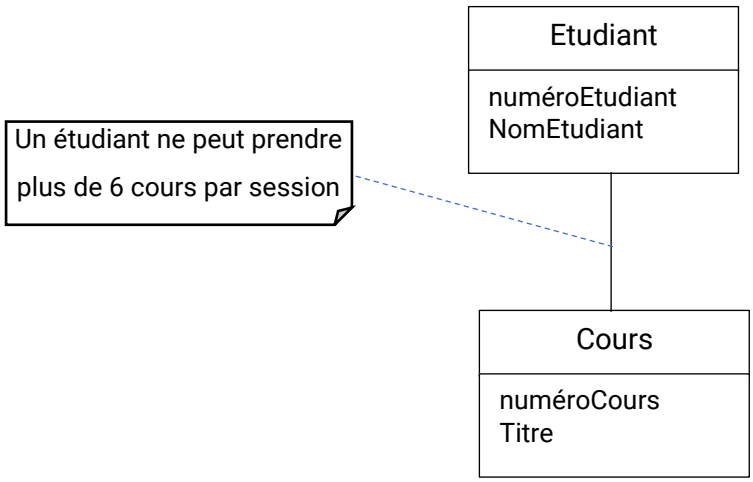
Pour les associations, on retrouve les contraintes {bag}, {hierarchy} et {ordered} qui signifient respectivement un ensemble où la classe peut se retrouver plusieurs fois, que les classes sont en hiérarchie dans la relation et en ordre dans la relation.

L'exemple suivant représente un cas où la relation entre Cours et inscription est ordonnée. C'est-à-dire où l'ordre dans lequel les inscriptions à un cours sont créées est important. La priorité est donnée aux inscriptions qui arrivent en premier.



D'autres contraintes peuvent être définies par les utilisateurs. Une contrainte est représentée par un texte entre crochets ({}). Il n'existe pas en UML d'obligation quant à l'utilisation d'un langage pour exprimer les contraintes, même si OCL est associé à la norme.

L'exemple suivant montre une contrainte exprimée en langue naturelle dans une note.



La qualification d'association est une forme de contrainte sur les associations.

Relation de dépendance

Dans la modélisation UML, une relation de dépendance est une relation unidirectionnelle dans laquelle une classe dépend d'une autre classe. La dépendance est représentée par un trait discontinu orienté.

La dépendance indique qu'un changement apporté à la classe source, de départ, pourrait nécessiter un changement dans la classe cible.