

# UML

Baptiste Bauer

Version v0.0.3, 2022-11-21 19:24:02

# Table des matières

1. Les cardinalités d'une association .....	1
2. Implémentation des cardinalités .....	3
3. L'association réflexive .....	4
4. L'agrégation .....	5
4.1. Qu'est-ce qu'une agrégation ? .....	5
4.2. Navigabilité et agrégation .....	6
4.3. Implémentation d'une agrégation .....	6
Index .....	7

# 1. Les cardinalités d'une association

Des **cardinalités** peuvent être ajoutées afin d'exprimer une **multiplicité du lien associatif entre deux classes**. C'est utile lorsque l'on souhaite indiquer qu'une instance de classe peut être liée (sémantiquement) à plusieurs instances d'une autre classe.

Dans le cas ci-après, il est impossible de savoir si plusieurs techniciens entretiennent un même véhicule ou si un véhicule est entretenu par plusieurs techniciens.

Les règles de gestion à exprimer sur le diagramme sont les suivantes :

- Un technicien entretien zéro, un ou plusieurs véhicules.
- Un véhicule est entretenu par au moins un technicien

Voici notre diagramme à jour de ces dernières informations :

Afin de bien comprendre le sens de lecture, voici de nouvelles règles de gestion :

- Un technicien entretien au moins 1 véhicule
- Un véhicule est entretenu par 1 seul technicien

Voici quelques exemples de cardinalités :

Exemple de cardinalité	Interprétation
1	Un et un seul. On n'utilise pas la notation 1..1.
1..*	Un à plusieurs
1..5	1 à 5 (maximum)
1-5	1 à 5 (maximum)
3..7	3 à 7 (maximum)
3-7	3 à 7 (maximum)
0..1	0 ou 1 seul
1,5	1 <b>ou</b> 5
1,5,7	1 <b>ou</b> 5 <b>ou</b> 7
0..*	0, 1 ou plusieurs
*	0, 1 ou plusieurs



Si vous avez l'habitude de faire de l'analyse selon la méthode Merise, vous aurez remarqué que les cardinalités sont inversées par rapport à celles d'UML.

La cardinalité est très utile au développeur pour savoir s'il doit contrôler le nombre d'objets B qu'il est possible d'associer à un objet A.

**Q1)** Pour chaque diagramme, exprimez la relation en prenant en compte les cardinalités.

- a. )
- b. )
- c. )



Lorsqu'une association exprime un lien vers un maximum de 0 ou une instance de l'objet lié, on parle d'**association simple**.

Lorsqu'une association exprime un lien vers un maximum de plusieurs instances de l'objet lié, on parle d'**association multiple**.

**Q2)** Pour chaque diagramme, indiquer s'il s'agit d'une association simple ou d'une association multiple en fonction du sens de lecture.

- a. )
- b. )
- c. )

**Q3)**

Réalisez le diagramme de classes correspondant au domaine de gestion décrit ci-après :

Une entreprise gère des hôtels. Des clients peuvent réserver des chambres dans ces hôtels. Une réservation ne peut porter que sur une seule chambre. Des prestations supplémentaires (petit déjeuner, réveil par l'accueil, encas nocturne) peuvent compléter la mise à disposition d'une chambre. Ces prestations peuvent être prévues lors de la réservation ou ultérieurement. Une chambre est équipée ou non de différentes options (lit simple / double, micro-onde, lit enfant, baignoire de type balnéo, etc)

Les associations doivent être nommées et les cardinalités précisées.

L'implémentation des cardinalités nécessite de savoir implémenter la navigabilité. Nous reviendrons alors sur ce sujet dans la partie sur l'[implémentation des cardinalités](#).

## 2. Implémentation des cardinalités

Nous avons appris lors de l'étude de la [notion de cardinalité](#) qu'elle permet d'exprimer une contrainte

Dans la partie du cours sur les [cardinalités](#), nous avons vu que les cardinalités expriment une contrainte sur le nombre d'objets B associés à un objet A.

Le développeur doit tenir compte de celles-ci dans l'implémentation de la classe.



Pour prendre en compte la cardinalité à l'extrémité d'une association navigable, le développeur doit compter le nombre d'instances liées et s'assurer que ce nombre respecte cette cardinalité. En PHP, la fonction `count` retourne le nombre d'éléments dans un tableau (utile pour dénombrer une collection).

Les cardinalités minimale et maximale doivent être vérifiées par le développeur.

Il n'y a aucune difficulté dans le contrôle des cardinalités. Ainsi, vous pouvez attaquer les exercices qui suivent.

**Q4)** Dans le diagramme ci-dessous, y a-t-il des cardinalités à contrôler ? Si oui, indiquez pour chacune d'elle si le contrôle doit être fait dès l'instanciation de l'objet depuis lequel commence la navigabilité ou après (dans ce cas préciser depuis quelle méthode).

**Q5)** Dans le diagramme ci-dessous, y a-t-il des cardinalités à contrôler ?

**Q6)** Dans le diagramme ci-dessous, y a-t-il des cardinalités à contrôler ?

**Q7)** Dans le diagramme ci-dessous, y a-t-il des cardinalités à contrôler ?

**Q8)** Implémentez le diagramme suivant :

**Q9)** Implémentez le diagramme suivant :

**Q10)** Implémentez le diagramme suivant :

## 3. L'association réflexive

**Une association réflexive est un lien entre deux objets de même type.**

Imaginons un technicien qui peut être le supérieur hiérarchique d'autres techniciens. Le diagramme suivant illustre cette relation :

Comme les deux classes mobilisées sont identiques, il ne faut en utiliser qu'une seule et donc faire un lien qui point sur elle-même :

C'est équivalent à cette représentation :

Voici la même modélisation mais avec une bidirectionnalité :

Ce qui est équivalent à :

### **Q11)** Travail à faire

- a. Implémentez le diagramme suivant (il n'y a rien de nouveau, cela reste une association bidirectionnelle comme nous savons les implémenter) :
- b. Vous veillerez à ce qu'un technicien ne puisse pas être son propre subordonné ou supérieur.

## 4. L'agrégation

### 4.1. Qu'est-ce qu'une agrégation ?



Rappel : l'association traduit un lien entre deux objets.

Le terme d'agrégation signifie l'action d'agréger, d'unir en un tout.

L'**agrégation** exprime la construction d'un objet à partir d'autres objets. Cela se distingue de la notion d'association que nous avons abordée jusqu'à maintenant. Effectivement, l'association traduit un lien entre deux objets alors que l'agrégation traduit le "regroupement" ou "l'assemblage" de plusieurs objets. Le lien exprimé est donc plus fort que pour une association.

Imaginez des pièces de Légo que vous utilisez pour construire une maison. Chaque pièce est un objet qui une fois agrégée avec les autres pièces permettent d'obtenir un autre objet (la maison). Il est tout à fait possible d'utiliser chaque pièce pour faire une autre construction. Détruire la maison ne détruit pas les pièces.

Implicitement, l'agrégation signifie « contient », « est composé de ». C'est pour cela qu'on ne la nomme pas sur le diagramme UML.

Autrement dit, une agrégation est le regroupement d'un ou plusieurs objets afin de construire un objet « complet » nommé **agrégat**.

Représentons le lien entre une équipe et les joueurs qui composent celle-ci (ici une équipe de volley) :

Le losange vide est le symbole qui caractérise une agrégation. Il est placé du côté de l'agrégat (l'objet qui est composé / assemblé).

La classe **Team** est l'**agrégat** (le composé de) alors que la classe **Player** est le **composant**.

Ce diagramme objet représente les joueurs qui composent une équipe de volley

Une agrégation présente les caractéristiques suivantes :



- L'agrégation est composée d'"éléments".
- Ce type d'association est non symétrique. Il n'est pas possible de dire "Une équipe est composée de joueurs et un Joueur est composé d'une équipe"
- les composants de l'agrégation sont **partageables**
- l'agrégat et les composants ont leur **propre cycle de vie**

**Les composants sont partageables :**

La particularité d'une agrégation est que **le composant peut être partagé**.

Par exemple, un joueur d'une équipe peut jouer (se partager) dans d'autres équipes :

Voici un diagramme objet pour illustrer ce partage :

Les instances de **Player** "aurel" et "clem" sont **partagées** dans deux équipes. Celle représentant "sofian" n'est pas partagée mais peut l'être à un moment donné.

Voici un autre exemple :

L'entreprise est la réunion en un tout de personnes et de locaux. Les personnes qui composent une entreprise peuvent travailler dans d'autres et un local peut servir à plusieurs entreprises. Nous retrouvons la notion de partage des composants.



Comme les composants sont partageables, la multiplicité du côté de l'agrégat peut être supérieur à 1.

[cardinalite multiple agregation] | ../images/cardinalite\_multiple\_agregation.png

### L'agrégat et les composants ont leur propre cycle de vie



Le cycle de vie d'un objet désigne sa création, ses changements d'état jusqu'à sa destruction.

- Un agrégat **peut** exister sans ses composants. (en programmation, il doit être possible d'instancier un agrégat sans ses composants)  
Une équipe peut exister même s'il elle n'a aucun joueur.
- Un composant **peut** exister sans être utilisé par l'agrégat. (en programmation, il doit être possible d'instancier un composant sans que l'agrégat l'utilise ou existe)
- la destruction de l'agrégat ne détruit pas ses composants (et vice versa) ce qui va dans le sens des deux points précédents

## 4.2. Navigabilité et agrégation

Tout ce que nous avons vu dans la [partie sur la navigabilité](#) s'applique dans le cadre d'une agrégation.

Il y a navigabilité bidirectionnelle entre **Person** et **Enterprise** et navigabilité unidirectionnelle de **Enterprise** vers **Local**.

## 4.3. Implémentation d'une agrégation

L'implémentation d'une agrégation est exactement la même qu'une association classique.

L'agrégation permet seulement d'exprimer conceptuellement le fait que les instances d'une classe sont des "assemblages" d'autres instances de classe. Une conceptualisation est une représentation de la réalité. Cela peut aider à mieux cerner la logique métier de l'application.



# Index

## A

agrégat, [5](#)

agrégation, [5](#)

association multiple, [2](#)

association simple, [2](#)

## C

Caractéristiques d'une agrégation, [5](#)

cardinalités, [1](#)

composant, [5](#)

## M

multiplicité du lien associatif entre deux classes,  
[1](#)