Relations entre classes [kr]
Support de Cours

Karine Zampieri, Stéphane Rivière, Béatrice Amerein-Soltner

Unisciel algoprog Version 20 mai 2018

Table des matières

Relations entre classes

Mots-Clés Relations entre classes, Association, Agrégation, Héritage ■
Requis Classes, Classes (suite) ■
Difficulté ● ○ ○

Objectif

Il existe trois grandes catégories de relations entre classes qui diffèrent à la fois au niveau conceptuel et au niveau du codage :

- L'association : simple relation d'utilisation
- L'agrégation : une classe entre dans la composition d'une autre classe
- L'héritage : une classe hérite des caractéristiques d'une autre classe

Pour chaque catégorie, ce module présente la notation utilisée dans le diagramme de classes d'UML (*Unified Modeling Language*, langage de modélisation objet unifié) et sa traduction en programmation.

Modularisation

Ce n'est pas une science exacte. Chaque programmeur décide des classes qu'il trouve utile pour son programme. C'est souvent l'étape la plus difficile d'un projet de programmation.

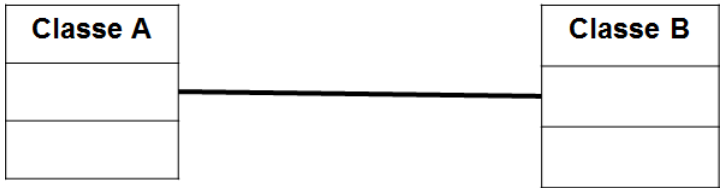
1 L'association

1.1 Définitions

Relation d'association

Deux classes sont en **association** lorsqu'elles sont unies par un lien conceptuel. Les deux classes se connaissent : chacune possède un rôle dans l'association.

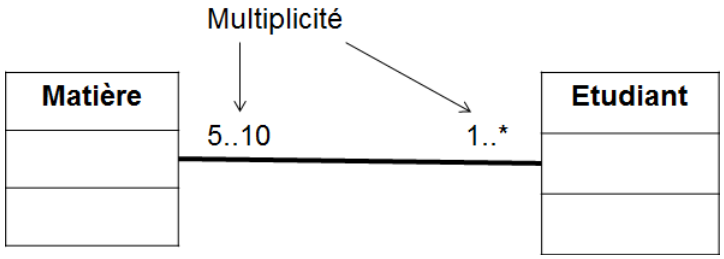
UML : Schéma d'une association



Exemple

Un exemple classique est celui des étudiants et de leurs cours. La matière sera caractérisée par son intitulé, et l'étudiant sera défini par son nom et son numéro d'étudiant. Les classes `Etudiant` et `Matière` sont associées l'une à l'autre :

- Une matière a besoin des étudiants pour être enseignée.
- L'étudiant a besoin de suivre certains cours pour apprendre.



Multiplicité

Une association possède des **cardinalités** qui représentent le nombre d'instances impliquées.

UML : Multiplicité

Sur un schéma UML, elle précise les limites inférieures et supérieures du nombre d'associations entre deux classes. L'étoile « * » signifie que la borne supérieure ne peut pas être déterminée.

Exemple

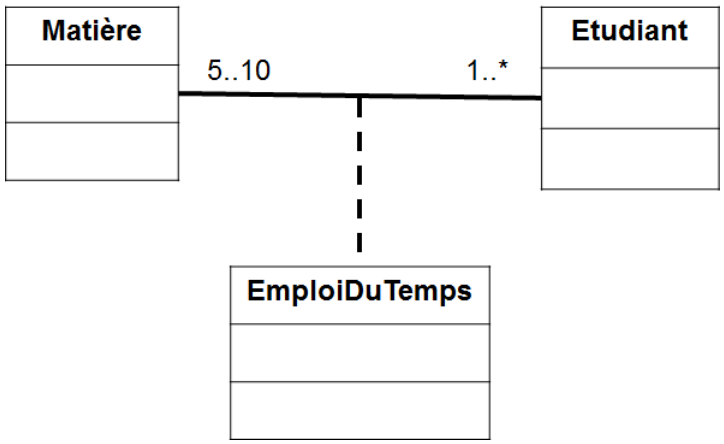
Sur la figure ci-avant, la multiplicité (5..10) signifie qu'un étudiant peut suivre entre 5 et 10 matières différentes. La multiplicité (1..*) signifie qu'une matière peut être suivie par 1 ou plusieurs étudiants.

Classe d'association

La conception de deux classes en relation d'association peut être complétée par l'existence d'une troisième classe, appelée **classe d'association** (ou *classe associative*), qui permet de spécifier les fonctionnalités de la liaison.

Exemple

La classe associative `EmploiDuTemps` permet de structurer les classes associées `Etudiant` et `Matière`.



1.2 Traduction en programmation

Les associations entre classes sont représentées par des références (ou des pointeurs). Les classes associées possèdent en attribut une ou plusieurs références vers l'autre classe. Le nombre de référence dépend de la cardinalité. Lorsque la cardinalité est 1 ou 0..1, il n'y a qu'une seule référence de l'autre classe.

Exemple
Un compte a une personne comme propriétaire mais une personne n'a pas forcément de compte (si c'est un prospect par exemple). Par conséquent un compte bancaire n'est pas initialisé dans le constructeur car une personne peut ne pas avoir de compte.

2 L'agrégation (ou composition)

2.1 Agrégation

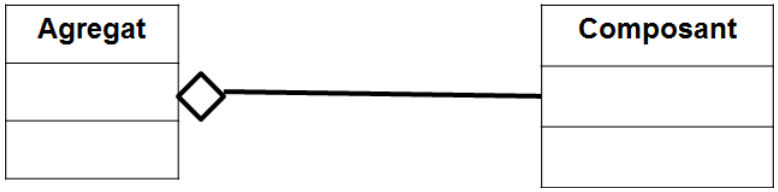
Relation d'agrégation
C'est une association particulière où une classe permet de créer (c.à.d. fait partie de) une autre classe.

Relation dissymétrique
Les deux classes ne sont pas au même niveau : une classe **contient** l'autre.

Objet agrégat
Appelé aussi *objet composite*, c'est celui qui contient comme attribut un autre objet.

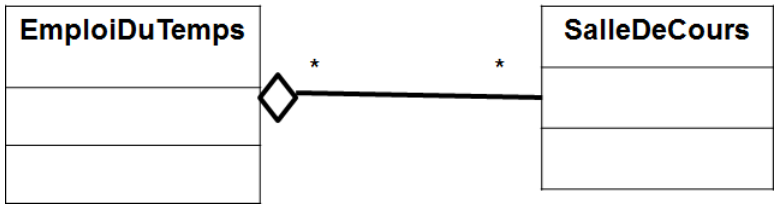
Objet agrégé
Appelé aussi *objet composant*, c'est celui qui est contenu dans l'objet agrégat.

UML : Schéma de l'agrégation



Lien contenant-contenu
Dans une agrégation, l'objet composant est une caractéristique de l'objet agrégat. Dit autrement : « l'agrégation permet de justifier un lien contenant-contenu, mais l'existence de l'instance contenue est indépendante de celle du contenant. »

Exemple
Supposons que les emplois du temps soient définis chaque semestre et que chacun contienne, entre autres, l'ensemble des salles de cours. Il s'agit là d'une relation contenant (l'emploi du temps) et contenu (les salles). Tous les semestres, quand l'emploi du temps sera remplacé et détruit, il ne faudra pas détruire les instances des salles qui sont toujours utilisées par d'autres emplois du temps. Il s'agit là d'une relation d'agrégation.



Exemple
Une automobile contient un moteur.

Navigabilité restreinte
Il arrive qu'une classe `A` ait comme attribut une ou plusieurs références à une autre classe `B`, mais que la classe B ne possède aucune référence à la classe `A`.

On dit dans ce cas que la navigabilité est restreinte de `A` vers `B`. Cela veut dire qu'à partir d'un objet `B`, on ne peut pas connaître le ou les objets `A` qui lui sont liés. `A` connaît `B` mais `B` ne connaît pas `A`.

2.2 Composition

Relation de composition

C'est une relation d'agrégation du type un contenant dans un contenu. La composition de deux objets implique l'existence des deux : l'existence de l'un n'aurait pas de sens sans l'autre.

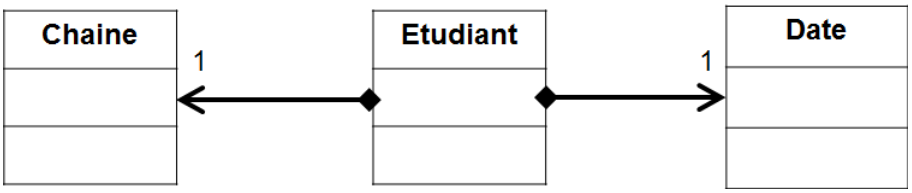
UML : Schéma de la composition

Différence entre l'agrégation et la composition

Elle dépend des propriétés de la liaison entre les deux objets. Lors de la composition, l'objet agrégat contient un nombre déterminé d'objets agrégés (en général, un seul). L'objet agrégé ne doit son existence que pour assurer celle de l'agrégat. Réciproquement, l'existence de l'agrégat n'a pas de sens sans l'objet composant.

Exemple

La classe `Etudiant` utilise les classes `Date` et `Chaine` pour définir ses attributs. La navigabilité (indiquée par la flèche) précise que la chaîne sera accessible grâce à l'`Etudiant`.



2.3 Traduction en programmation

L'agrégation est traduite au niveau du code comme une association, par une ou plusieurs références dans les deux classes concernées. La différence se situe au niveau du cycle de vie du ou des composants.

Dans une relation d'agrégation :

- Le ou les objets agrégés sont construits en même temps que l'objet agrégat. Alors que dans le cadre d'une association simple, les objets en relations sont instanciés indépendamment.
- Dans le cas d'une composition, si l'objet agrégat est détruit, le ou les objets agrégés le sont aussi.

Exemple
Il semble judicieux de placer dans la classe `EmploiDuTemps` un attribut de type tableau de salle de cours : c'est une solution d'implémentation de la relation d'agrégation.

Exemple
Un `Damier` est composé de 100 cases (objets de classe `Case`). Les cases ne peuvent pas exister sans le damier, donc elles sont instanciées à la construction du damier.

3 L'héritage

3.1 Définitions

Relation d'héritage
C'est une relation entre classes de type « est une sorte de » ou même « est un(e) ».

Classe mère – Classe fille
Une classe de base, appelée **classe mère** ou **super-classe**, transfert toutes ses caractéristiques (attributs et méthodes) à une autre classe, appelée **classe fille** ou **sous-classe**.

Terminologie
On dit que :

- La classe fille **hérite** de la classe mère, ou que
- La classe fille **dérive** de la classe mère, ou encore que
- La classe fille **étend** la classe mère.

Est-un
Un objet de la classe fille **EST UN** objet de la classe mère avec des caractéristiques supplémentaires. A chaque fois qu'on crée un objet dérivé, on crée aussi un objet de base.

Conséquences
Un objet de la classe fille :

- Peut utiliser tous les attributs et les méthodes de sa classe mère (et de la mère de sa mère, etc.)
- Possède en plus des attributs et méthodes propres.

Relation entre classes

- Un objet peut appartenir à plusieurs classes à la fois (sa classe d'instanciation mais aussi sa classe mère, et la mère de sa classe mère, etc.).
- L'héritage est une relation entre classes mais pas une relation entre objets. Seules les classes ont des relations d'héritage, pas les objets.

3.2 Spécialisation et Généralisation

Deux raisons peuvent nous amener à constituer des classes sur la base d'autres classes. Ce sont la **spécialisation** et la **généralisation**.

Exemple fondé sur la spécialisation
Reprenons le thème des comptes bancaires. Supposons que la banque propose des comptes d'épargne en plus des comptes traditionnels. Un compte d'épargne est une sorte de compte bancaire qui possède en plus un certain taux d'intérêt. Pour créer une classe pour les comptes d'épargne, nous allons réutiliser la classe `CBancaire` en utilisant l'héritage. La classe `CEpargne` possédera automatiquement toutes les caractéristiques de la classe `CBancaire` plus celles que nous allons lui ajouter.

FIGURE 1 – fig Exemple1, (a)

Pour trouver la classe `CEpargne`, nous avons utilisé la démarche de spécialisation : à partir d'une classe de base, nous avons trouvé une classe plus « spécialisée » que nous avons fait dériver de cette classe de base.

Exemple fondé sur la généralisation
Considérons qu'après une première analyse, nous ayons défini les deux classes suivantes :

FIGURE 2 – fig Exemple2

Nous constatons que `BouteilleDeLait` et `BouteilleDeVin` ont des points communs. Nous pouvons « factoriser » ces caractéristiques communes en créant une super-classe commune. Cela évite de définir plusieurs fois ces caractéristiques.

FIGURE 3 – fig Exemple2, (b)