TN29 : Introduction à la modélisation UML pour la conception bases de données

Stéphane Crozat

13 mars 2018

Table des matières

Classes et associations en UML

A. Notion de modèle

Objectifs

Savoir ce qu'est un modèle
Savoir ce qu'est le langage UML

1. Exercice : Centre médical

[5 min]
Soit le modèle conceptuel suivant représentant des visites dans un centre médical. Quelles sont les assertions vraies selon ce schéma ?

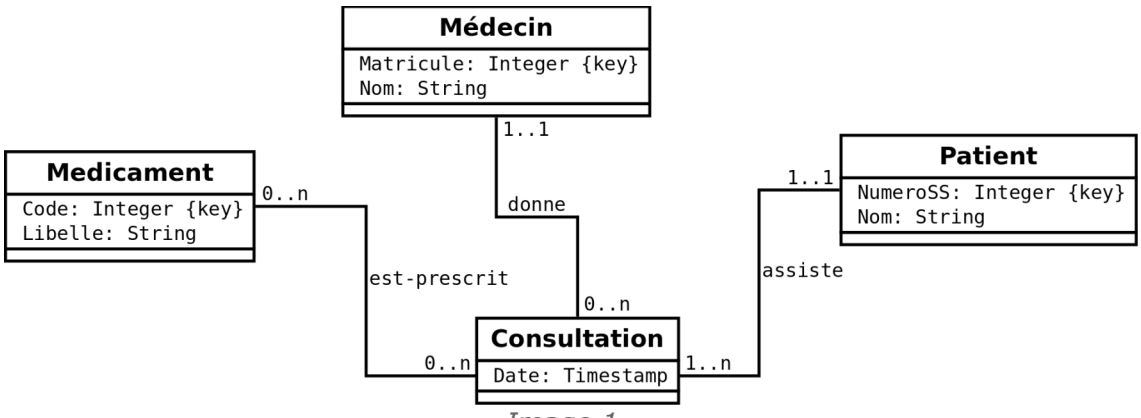


Image 1

☐	Un patient peut effectuer plusieurs visites.
☐	Tous les patients ont effectué au moins une consultation.
☐	Un médecin peut recevoir plusieurs patients pendant la même consultation.
☐	Un médecin peut prescrire plusieurs médicaments lors d'une même consultation.
☐	Deux médecins différents peuvent prescrire le même médicament.

2. Qu'est ce qu'un modèle ?

Définition : Modèle

« Modeling, in the broadest sense, is the cost-effective use of something in place of something else for some cognitive purpose. It allows us to use something that is simpler, safer or cheaper than reality instead of reality for some purpose. A model represents reality for the given purpose; the model is an abstraction of reality in the sense that it cannot represent all aspects of reality. » (Rothenberg, 1989 [Rothenberg et al., 1989], cité par Arribe, 2014 [Arribe, 2014])

« Système physique, mathématique ou logique représentant les structures essentielles d'une réalité et capable à son niveau d'en expliquer ou d'en reproduire dynamiquement le fonctionnement. » (TLFi)

Fondamental : Modèle

Un modèle est une représentation simplifiée de la réalité en vue de réaliser quelque chose.

3. Qu'est ce qu'un modèle en informatique ?

Définition : Modèle informatique

Un modèle informatique est une représentation simplifiée de la réalité en vue de réaliser un traitement avec un ordinateur.

Complément : Numérisation et abstraction : Toute information numérique a été codée selon un modèle donné

« Tout numérisation est une représentation de la réalité sous la forme d'une modélisation numérique. Cette modélisation procède d'une abstraction au sens où c'est une séparation d'avec le réel, au sens où c'est une construction destinée à la manipulation (algorithmique en l'occurrence) et au sens où c'est une simplification de la réalité. »

http://aswemay.fr/co/tropism-pres.html[1]

4. Qu'est ce qu'un bon modèle ?

Attention

Un modèle est une abstraction, une simplification de la réalité, ce n'est pas la réalité, il n'est jamais complètement fidèle par construction.

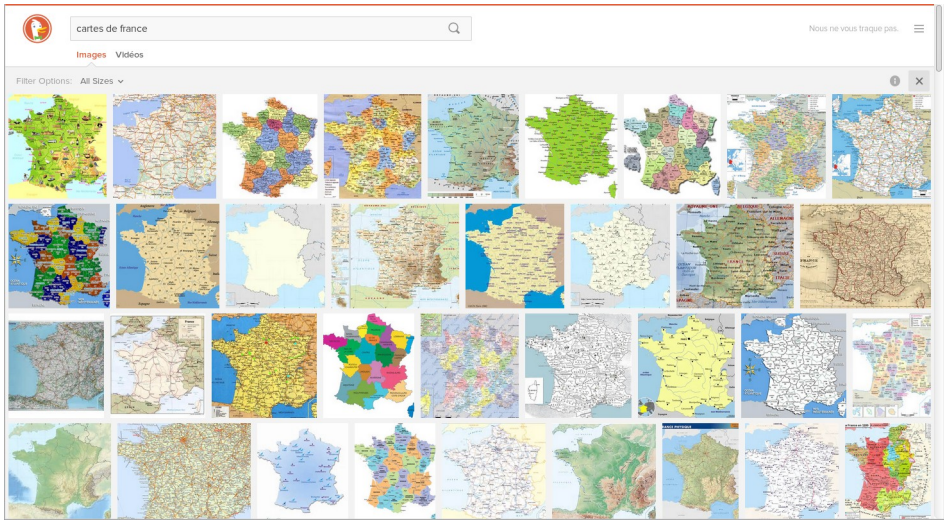
Le seul modèle complètement fidèle à la réalité est la réalité elle-même, et ce n'est donc pas un modèle.

1 - http://aswemay.fr/co/tropism-pres.html

Exemple : La carte et le territoire

Une carte est un modèle d'un territoire. Elle est une représentation simplifiée destiné à un usage particulier :

- randonner à pied, en vélo ;
- se diriger en voiture sur des grands axes, sur des axes secondaires ;
- voler en avion de tourisme, en avion de ligne ;
- naviguer sur fleuve, sur mer ;
- étudier les frontières d'une région, d'un pays, de la terre ;
- étudier la démographie, l'économie... ;
- ...



Fondamental

À partir de cet exemple on notera que :

1. **Un modèle est orienté par un usage.**
   Chacune de ces cartes est très différente selon ce que l'on veut faire.
2. **Un modèle ne cherche pas à être proche de la réalité.**
   Chacune de ces cartes est très différente de la réalité qu'elle représente.
3. **Un modèle adresse un niveau d'information qui existe mais qui n'est pas accessible dans la réalité.**
   Chacune de ces cartes permet quelque chose que ne permet pas l'accès direct à la réalité.

Méthode : Le rasoir d'Ockham : Entre deux modèles donnés le meilleur modèle est-il toujours le plus fourni ?

La méthode de raisonnement connue sous le nom de rasoir d'Ockham (du nom du philosophe éponyme) consiste à préférer les solutions les plus simples aux plus complexes, lorsqu'elles semblent permettre également de résoudre un problème donné ; entre deux théories équivalentes, toujours préférer la plus simple.

Ce principe s'applique très bien à la modélisation : étant donné un objectif et plusieurs modèles possibles, il ne faut pas choisir a priori celui qui représente le plus de choses, mais préférer le plus simple dès qu'il couvre le besoin.

C'est un principe d'économie (il coûte moins cher à produire) et d'efficacité (car les éléments inutiles du modèle plus fourni nuiront à l'efficacité de la tâche).

Exemple

Ainsi, pour naviguer en voiture, il est plus simple de ne **pas** avoir sur la carte les chemins de randonnées qui ne sont pas praticables en voiture.

B. Introduction au diagramme de classes UML : classes et associations

Objectifs

Savoir faire un modèle conceptuel.
Savoir interpréter un modèle conceptuel.

1. Lab I

Description du problème

[15 min]

Un laboratoire souhaite gérer les médicaments qu'il conçoit.

- Un médicament est décrit par un nom, qui permet de l'identifier. En effet il n'existe pas deux médicaments avec le même nom. Un médicament comporte une description courte en français, ainsi qu'une description longue en latin. On gère aussi le conditionnement du médicament, c'est à dire le nombre de pilules par boîte (qui est un nombre entier).
- À chaque médicament on associe une liste de contre-indications, généralement plusieurs, parfois aucune. Une contre-indication comporte un code unique qui l'identifie, ainsi qu'une description. Une contre-indication est toujours associée à un et un seul médicament.

Exemple de données

Afin de matérialiser notre base de données, nous obtenons les descriptions suivantes :

- Le **Chourix** a pour description courte *« Médicament contre la chute des choux »* et pour description longue *« Vivamus fermentum semper porta. Nunc diam velit, adipiscing ut tristique vitae, sagittis vel odio. Maecenas convallis ullamcorper ultricies. Curabitur ornare. »*. Il est conditionné en boîte de 13.

  Ses contre-indications sont :
  - CI1 : Ne jamais prendre après minuit.
  - CI2 : Ne jamais mettre en contact avec de l'eau.
- Le **Tropas** a pour description courte *« Médicament contre les dysfonctionnements intellectuels »* et pour description longue *« Suspendisse lectus leo, consectetur in tempor sit amet, placerat quis neque. Etiam luctus porttitor lorem, sed suscipit est rutrum non. »*. Il est conditionné en boîte de 42.

  Ses contre-indications sont :
  - CI3 : Garder à l'abri de la lumière du soleil

Question 1

Réaliser le modèle conceptuel de données en UML du problème.

Question 2

Étendre le modèle conceptuel UML afin d'ajouter la gestion des composants. Un composant est identifié par un code unique et possède un intitulé. Tout médicament possède au moins un

composant, souvent plusieurs. Tout composant peut intervenir dans la fabrication de plusieurs médicaments. Il existe des composants qui ne sont pas utilisés pour fabriquer des médicaments et que l'on veut quand même gérer.

2. Présentation d'UML

UML✯ est un langage de représentation destiné en particulier à la modélisation objet. UML est devenu une norme OMG✯ en 1997.

UML propose un formalisme qui impose de "penser objet" et permet de rester indépendant d'un langage de programmation donné. Pour ce faire, UML normalise les concepts de l'objet (énumération et définition exhaustive des concepts) ainsi que leur notation graphique. Il peut donc être utilisé comme un moyen de communication entre les étapes de spécification conceptuelle et les étapes de spécifications techniques.

Fondamental : Diagramme de classe

Le diagramme de classes est un sous ensemble d'UML qui s'attache à la description statique d'un modèle de données représentées par des classes d'objets.

Remarque

Dans le domaine des bases de données, UML peut être utilisé à la place du modèle E-A✯ pour modéliser le domaine. De la même façon, un schéma conceptuel UML peut alors être traduit en schéma logique (relationnel ou relationnel-objet typiquement).

3. Classes

Définition : Classe

Une classe est un type abstrait caractérisé par des propriétés (attributs et méthodes) communes à un ensemble d'objets et permettant de créer des instances de ces objets, ayant ces propriétés.

Syntaxe

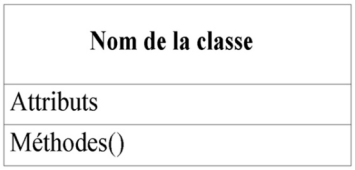


Image 2 Représentation UML d'une classe

Exemple : La classe Voiture

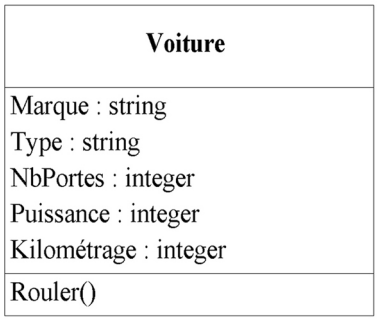


Image 3 Exemple de classe représentée en UML

Exemple : Une instance de la classe Voiture

L'objet V1 est une instance de la classe Voiture.

V1 : Voiture

- Marque : 'Citroën'
- Type : 'ZX'
- Portes : 5
- Puissance : 6
- Kilométrage : 300000

Complément

La modélisation sous forme de diagramme de classes est une modélisation statique, qui met en exergue la structure d'un modèle, mais ne rend pas compte de son évolution temporelle. UML propose d'autres types de diagrammes pour traiter, notamment, de ces aspects.

4. Attributs

Définition : Attribut

Un attribut est une information élémentaire qui caractérise une classe et dont la valeur dépend de l'objet instancié.

Un attribut est typé : Le domaine des valeurs que peut prendre l'attribut est fixé a priori.

- **Un attribut peut être multivalué** : Il peut prendre plusieurs valeurs distinctes dans son domaine.
- **Un attribut peut être dérivé** : Sa valeur alors est une fonction sur d'autres attributs de la classe
- **Un attribut peut être composé** (ou composite) : Il joue alors le rôle d'un groupe d'attributs (par exemple une adresse peut être un attribut composé des attributs numéro, type de voie, nom de la voie). Cette notion renvoie à la notion de variable de type `Record` dans les langages de programmation classiques.

Attention : On utilise peu les attributs dérivés et composés en UML

- **En UML on préfère l'usage de méthodes aux attributs dérivés. On utilisera toujours des méthodes dès que la valeur de l'attribut dérivé dépend d'autres attributs extérieurs à sa classe.**
- **En UML on préfère l'usage de compositions aux attributs composés. On utilisera toujours des compositions pour les attributs composés et multivalués.**

Syntaxe

```
attribut:type
attribut_multivalué[nbMinValeurs..nbMaxValeurs]:type
/attribut_dérivé:type
attribut_composé
    - sous-attribut1:type
    - sous-attribut2:type
    - ...
```

Exemple : La classe Personne

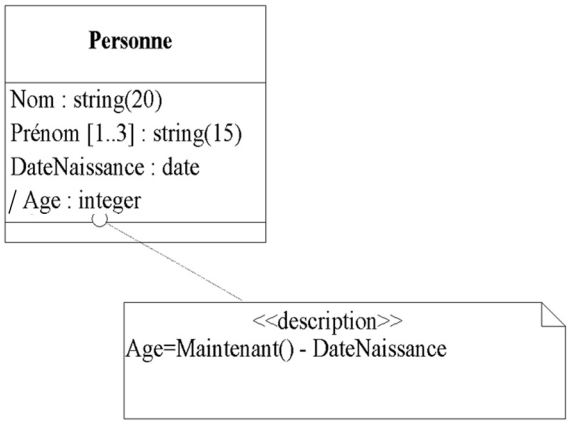


Image 4 Représentation d'attributs en UML

Dans cet exemple, les attributs Nom, Prénom sont de type *string*, l'un de 20 caractères et l'autre de 10, tandis que DateNaissance est de type *date* et Age de type *integer*. Prénom est un attribut multivalué, ici une personne peut avoir de 1 à 3 prénoms. Age est un attribut dérivé, il est calculé par une fonction sur DateNaissance.

Complément : Voir aussi

Méthodes

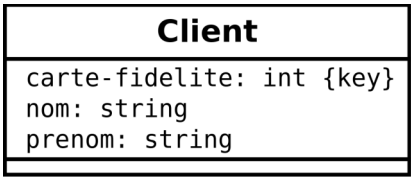
Composition

5. Repérage des clés

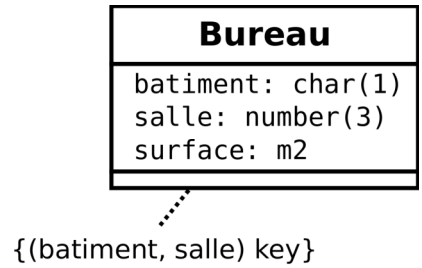
Un attribut ou un groupe d'attributs peut être annoté comme étant **clé** s'il permet d'identifier de façon unique un objet de la classe.

On ajoute le symbole `{key}` à côté du ou des attributs concernés.

Exemple



Clé en UML



Clé composée de deux attributs

Méthode

Le repérage des clés n'est pas systématique en UML (la définition des clés se fera essentiellement au niveau logique). On cherchera néanmoins à repérer les clés rendues évidentes par la phase de clarification.

Attention

On n'ajoutera jamais de clé artificielle au niveau du MCD. Si aucune clé n'est évidente, on laisse la classe sans clé.

Attention : Attribut souligné et

On trouvera dans ce cours des exemples d'attributs soulignés ou précédés de # pour exprimer l'unicité. Ce n'est pas une pratique standard et la notation `{key}` devrait lui être substituée.

- **Un attribut souligné est normalement un attribut de classe, ou *static*, en UML,**
- **Un attribut précédé de # est normalement un attribut protégé en UML.**

Mais les concepts d'attribut de classe et d'attribut protégé ne sont pas utilisés dans le cadre des bases de données.

6. Méthodes

Définition : Méthode

Une méthode (ou opération) est une fonction associée à une classe d'objet qui permet d'agir sur les objets de la classe ou qui permet à ces objets de renvoyer des valeurs (calculées en fonction de paramètres).

Syntaxe

```
1 methode(paramètres):type
```

Remarque : Méthodes et modélisation de BD

Pour la modélisation des bases de données, les méthodes sont surtout utilisées pour représenter des données calculées (à l'instar des attributs dérivées) ou pour mettre en exergue des fonctions importantes du système cible. Seules les méthodes les plus importantes sont représentées, l'approche est moins systématique qu'en modélisation objet par exemple.

Remarque : Méthodes, relationnel, relationnel-objet

Lors de la transformation du modèle conceptuel UML en modèle logique relationnel, les méthodes **ne seront généralement pas implémentées**. Leur repérage au niveau conceptuel sert donc surtout d'aide-mémoire pour l'implémentation au niveau applicatif.

Au contraire, un modèle logique relationnel-objet permettra l'implémentation de méthodes directement associées à des tables. Leur repérage au niveau conceptuel est donc encore plus important.

Complément

Transformation des méthodes par des vues - p.40

7. Associations

Définition : Association

Une association est une relation logique entre deux classes (association binaire) ou plus (association n-aire) qui définit un ensemble de liens entre les objets de ces classes.

Une association est **nommée**, généralement par un verbe. Une association peut avoir des propriétés (à l'instar d'une classe). Une association définit le nombre minimum et maximum d'instances autorisée dans la relation (on parle de cardinalité).

Syntaxe

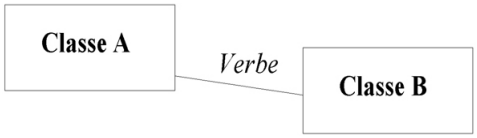


Image 5 Notation de l'association en UML

Attention

Le nom de l'association (verbe qui la décrit) est obligatoire, au même titre que le nom d'une classe ou d'un attribut.

Remarque

Une association est généralement bidirectionnelle (c'est à dire qu'elle peut se lire dans les deux sens). Les associations qui ne respectent pas cette propriété sont dites unidirectionnelles ou à navigation restreinte.

Exemple : L'association Conduit

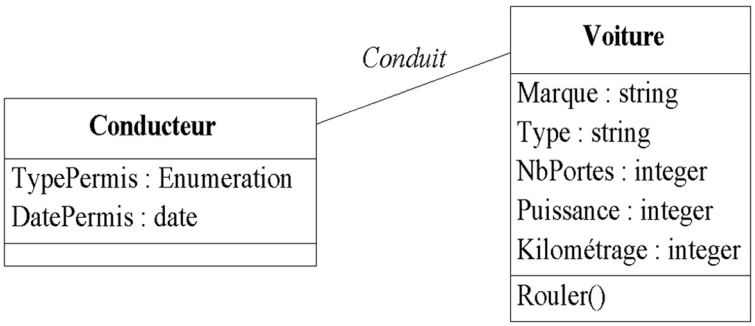


Image 6 Représentation d'association en UML

L'association Conduit entre les classes Conducteur et Voiture exprime que les conducteurs conduisent des voitures.

Complément : Voir aussi

- *Cardinalité*
- *Explicitation des associations*
- *Associations ternaires*
- *Contraintes sur les associations* - p.40

8. Cardinalité

Définition : Cardinalité d'une association

La cardinalité d'une association permet de représenter le nombre minimum et maximum d'instances qui sont autorisées à participer à la relation. La cardinalité est définie pour les deux sens de la relation.

Syntaxe

Si min_a (resp. max_a) est le nombre minimum (resp. maximum) d'instances de la classe A autorisées à participer à l'association, on note sur la relation, à côté de la classe A : $min_a..max_a$.

Si le nombre maximum est indéterminé, on note n ou *.

Attention

La notation de la cardinalité en UML est opposée à celle adoptée en E-A. En UML on note à gauche (resp. à droite) le nombre d'instances de la classe de gauche (resp. de droite) autorisées dans l'association. En E-A, on note à gauche (resp. à droite) le nombre d'instances de la classe de droite (resp. de gauche) autorisées dans l'association.

Remarque

Les cardinalités les plus courantes sont :

- 0..1 (optionnel)
- 1..1 ou 1 (un)
- 0..n ou 0..* ou * (plusieurs)
- 1..n ou 1..* (obligatoire)

Exemple : La cardinalité de l'association Possède

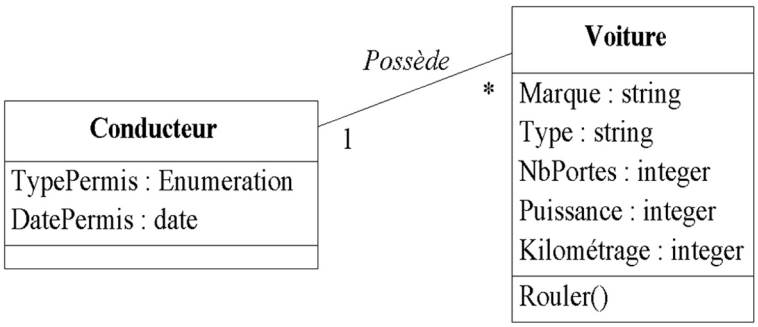


Image 7 Représentation de cardinalité en UML

Ici un conducteur peut posséder plusieurs voitures (y compris aucune) et une voiture n'est possédée que par un seul conducteur.

Fondamental : Terminologie

- **On appelle association 1:1 les associations de type :**
  - **0..1:0..1**
  - **0..1:1..1**
  - **1..1:0..1**
  - **1..1:1..1**
- **On appelle association 1:N les associations de type :**
  - **0..1:0..N**
  - **0..1:1..N**
  - **1..1:0..N**
  - **1..1:1..N**
- **On appelle association N:M (ou M:N) les associations de type :**
  - **0..N:0..N**
  - **0..N:1..N**
  - **1..N:0..N**
  - **1..N:1..N**

9. Classe d'association

Définition : Classe d'association

On utilise la notation des classes d'association lorsque l'on souhaite ajouter des propriétés à une association.

Syntaxe : Notation d'une classe d'association en UML

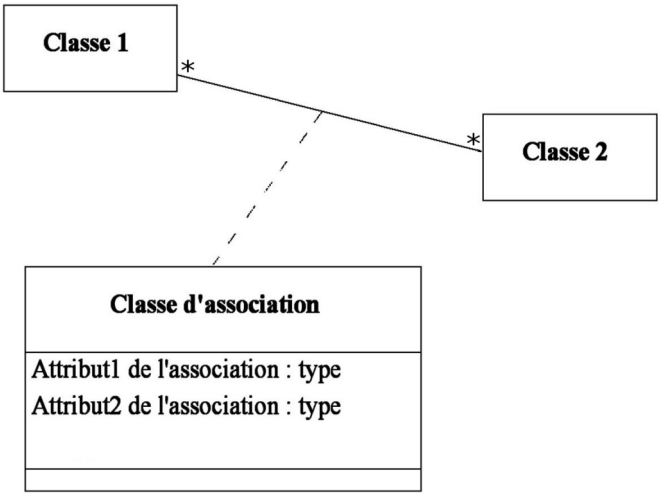


Image 8 Notation d'une classe d'association en UML

Méthode

On réserve en général les classes d'association aux associations N:M.

Il est toujours possible de réduire une classe d'association sur une association 1:N en migrant ses attributs sur la classe côté N, et c'est en général plus lisible ainsi.

Exemple : Exemple de classe d'association

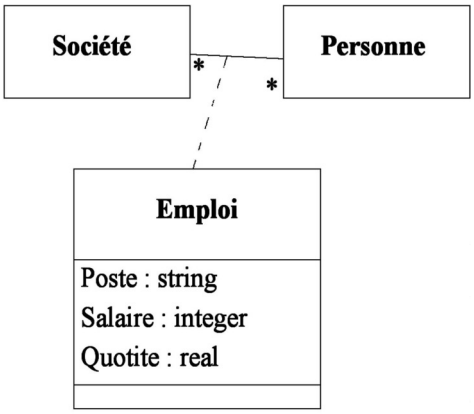


Image 9 Emplois

Conseil

Selon le standard UML une classe d'association est une classe et à ce titre elle peut être mobilisée dans d'autres associations ou dans des héritages. Nous déconseillons néanmoins ces notations qui ont tendance à complexifier la lecture et la transformation du diagramme.

Nous conseillons donc de ne jamais associer une classe d'association.

Exercices d'initiation

II

A. Exercice : Lire l'UML

[15 min]

Tennis

Le schéma suivant représente les rencontres lors d'un tournoi de tennis. Quelles sont les assertions vraies selon ce schéma ?

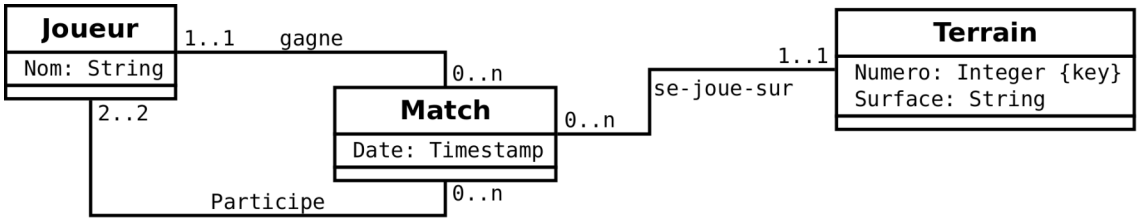


Image 10

- ☐ On peut jouer des matchs de double.
- ☐ Un joueur peut gagner un match sans y avoir participé.
- ☐ Il peut y avoir deux matchs sur le même terrain à la même heure.
- ☐ Connaissant un joueur, on peut savoir sur quels terrains il a joué.

Journal

Voici le schéma conceptuel du système d'information (très simplifié) d'un quotidien. Quelles sont les assertions vraies selon ce schéma ?

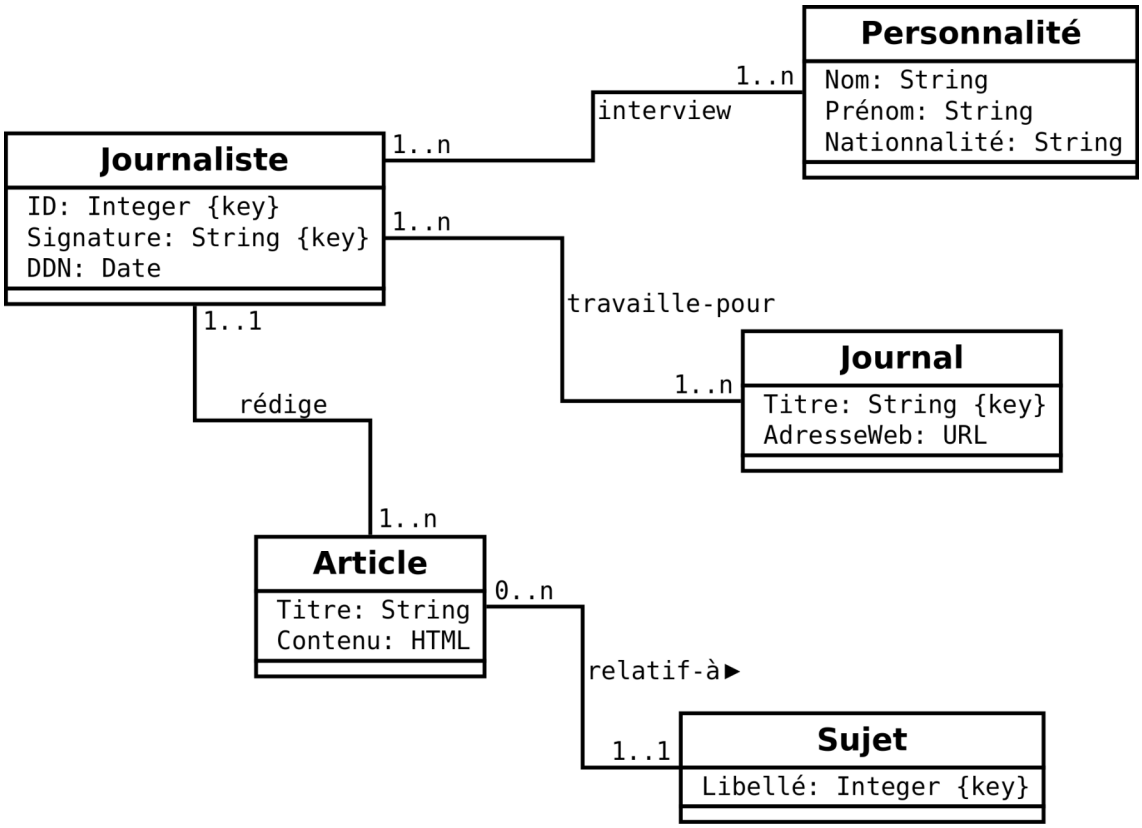


Image 11

☐	Un article peut être rédigé par plusieurs journalistes.
☐	Un article peut être publié plusieurs fois dans le même journal.
☐	Un article peut être publié dans un journal par un journaliste qui ne travaille pas pour ce journal.
☐	Il peut y avoir plusieurs articles sur le même sujet.
☐	Un journaliste peut interviewer une personnalité sans faire d'article à ce propos.

Logistique

Une société de transport routier veut installer un système d'information pour rendre plus efficace sa logistique. Embauché au service informatique de cette compagnie, vous êtes donc chargé de reprendre le travail déjà effectué (c'est à dire le schéma suivant).

Quelles sont les assertions vraies selon ce schéma ?

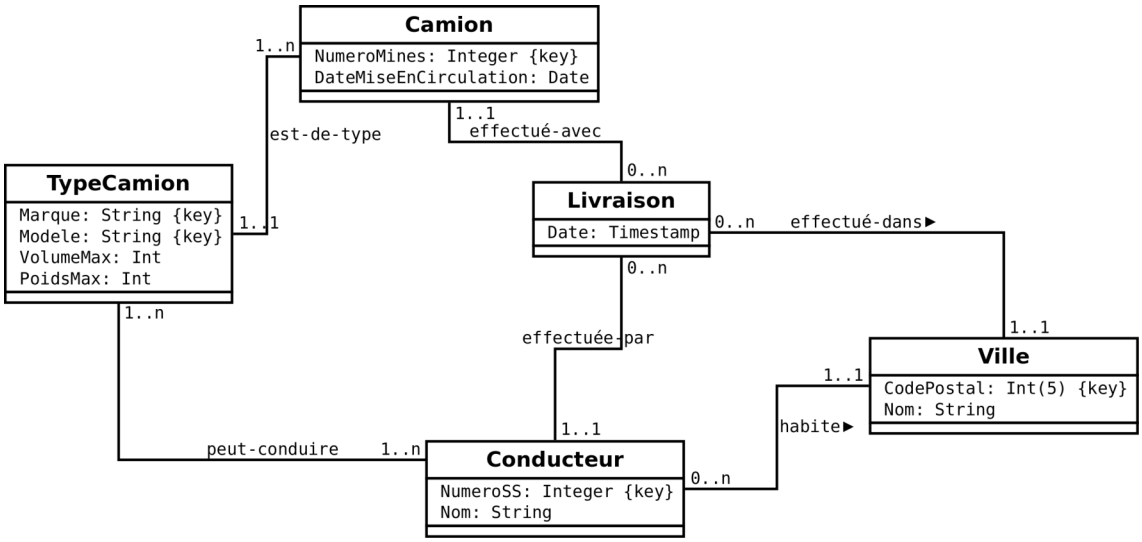


Image 12

- [ ] Un conducteur peut conduire plusieurs camions.
- [ ] Un conducteur peut conduire un camion sans y être autorisé.
- [ ] Il peut y avoir plusieurs conducteurs pour le même camion.
- [ ] Un conducteur peut livrer sa propre ville

B. Cours et intervenants

[20 min]

On souhaite réaliser une base de données pour gérer les cours dispensés dans une école d'ingénieur, ainsi que les personnes qui interviennent dans ces cours.

Chaque cours est identifié par une année et un numéro. Chaque cours a donc un numéro unique localement à chaque année. Un cours possède un titre et un type ('C' pour Cours, 'TD' ou 'TP'). Un cours possède également une date de début, et une date de fin, qui est toujours de 5 jours après la date de début.

Chaque intervenant est identifié par son nom (deux intervenants ne peuvent pas avoir le même nom). Il a un prénom, un bureau, un ou plusieurs numéros de téléphones (jusqu'à trois numéros) et des spécialités. Un bureau est défini par un centre ('R' pour Royallieu, 'BF' pour Benjamin Franklin et 'PG' pour Pierre Guillaumat), un bâtiment (une lettre de A à Z), et un numéro (inférieur à 1000). Les spécialités sont des couples de chaînes de caractères désignant un domaine (par exemple 'BD') et une spécialité (par exemple 'SGBDRO').

Chaque cours est donné par un unique intervenant.

Voici un exemple : Le cours 'Machines universelles', n°21 de l'année 2014 est donné par Alan Turing entre le 05/01/2014 et le 10/01/2014. C'est un cours de type 'C'. Alan Turing a le bureau 666 au bâtiment X de PG. Il a les numéros de téléphone 0666666666 et 0766666666. Il possède les spécialités suivantes :

- Domaine : Mathématique ; Spécialité : Cryptographie
- Domaine : Informatique ; Spécialité : Algorithmie
- Domaine : Informatique ; Spécialité : Intelligence Artificielle

Question

Réaliser le modèle UML de la base de données. Préciser les clés et les types des attributs.

Introduction à l'héritage

III

A. Introduction à l'héritage

Objectifs

Savoir utiliser l'héritage lors une modélisation

1. Exercice : Mariages

Étant donné le schéma UML suivant, quelles sont les assertions vraies ?

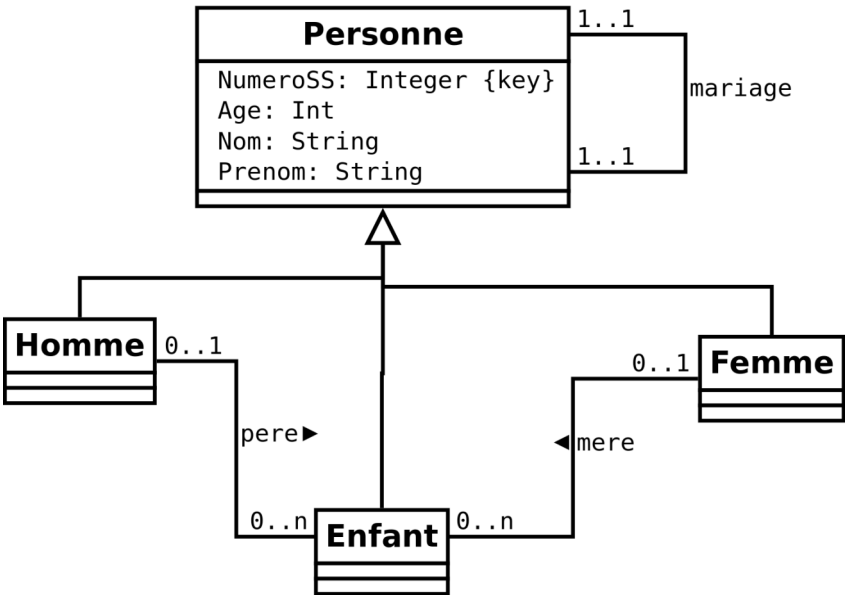


Image 13

☐	Les mariages homosexuels sont possibles.
☐	La polygamie est possible.
☐	Une femme peut ne pas être mariée.
☐	Les enfants peuvent être plus âges que leurs parents.
☐	Deux hommes peuvent avoir un enfant ensemble.
☐	Une femme peut avoir plusieurs enfants.
☐	Un enfant a obligatoirement deux parents.
☐	Les enfants peuvent se marier.
☐	Tous les enfants sont mariés.
☐	Les personnes mariées ont toujours le même nom.

2. Héritage

Définition : Héritage

L'héritage est l'association entre deux classes permettant d'exprimer que l'une est plus générale que l'autre. L'héritage implique une transmission automatique des propriétés (attributs et méthodes) d'une classe A à une classe A'.

Dire que A' hérite de A équivaut à dire que A' est une sous-classe de A. On peut également dire que A est une généralisation de A' et que A' est une spécialisation de A.

Syntaxe

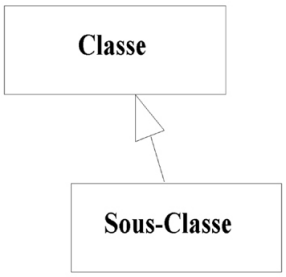
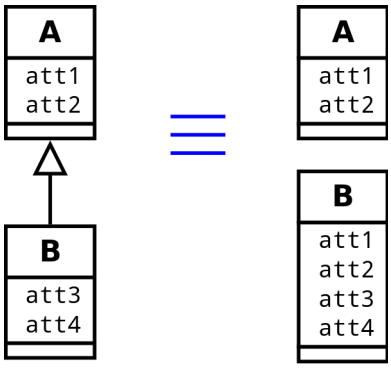


Image 14 Notation de l'héritage en UML

Fondamental : Factorisation

Outre qu'il permet de représenter une relation courante dans le monde réel, l'héritage a un avantage pratique, celui de factoriser la définition de propriétés identiques pour des classes proches.

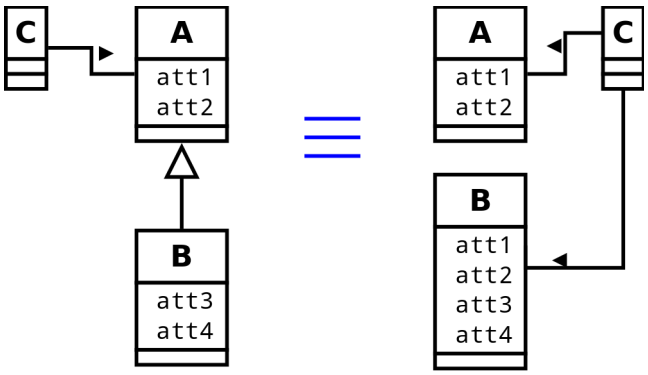


Héritage et factorisation

Fondamental : Is-a

L'héritage permet de représenter la relation "est-un" entre deux objets (*is-a* en anglais).

Donc tout ce qui est vrai pour la classe mère est vrai pour ses classes filles. En particulier si une classe C exprime une association avec une classe A dont hérite B, cela signifie que C peut être associée à B.



Héritage et propriété "is-a"

Exemple : La classe Conducteur

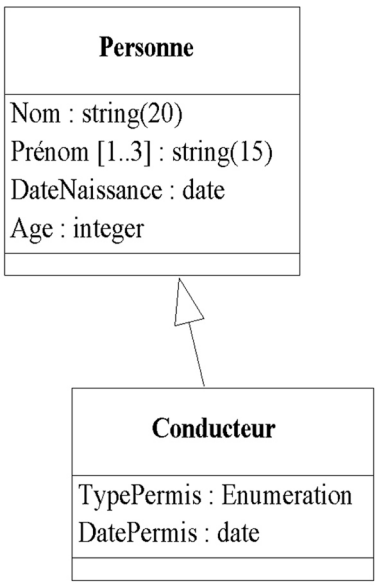


Image 15 Représentation d'héritage en UML

Dans cet exemple la classe Conducteur hérite de la classe Personne, ce qui signifie qu'un objet de la classe conducteur aura les attributs de la classe Conducteur (TypePermis et DatePermis) mais aussi ceux de la classe Personne (Nom, Prénom, DateNaissance et Age). Si la classe Personne avait des méthodes, des associations..., la classe Conducteur en hériterait de la même façon.

3. Exemple : Des voitures et des conducteurs

Exemple

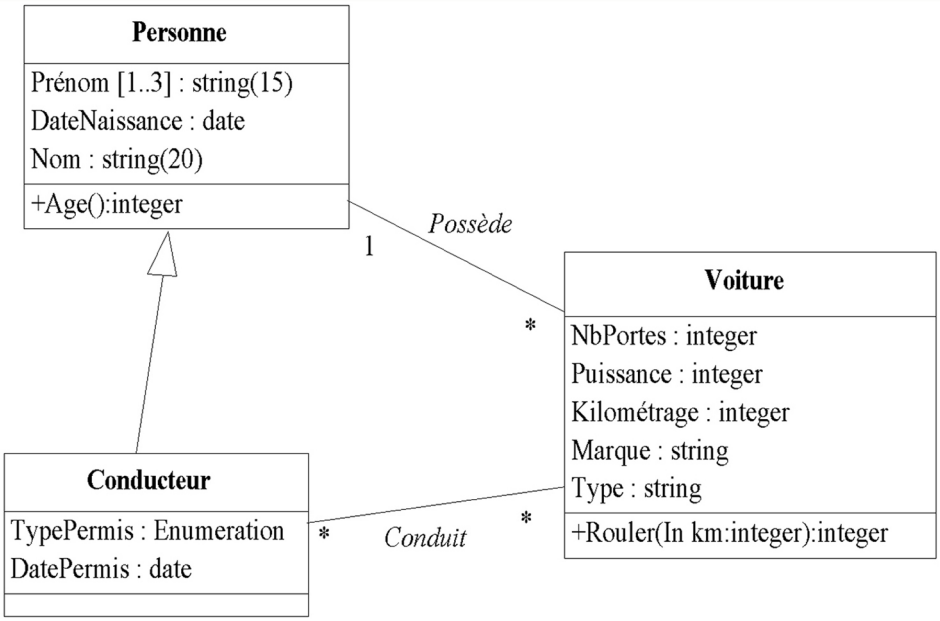


Image 16 Exemple très simple de diagramme de classes

Les relations de ce diagramme expriment que les conducteurs sont des personnes qui ont un permis ; que toute voiture est possédée par une unique personne (qui peut en posséder plusieurs) ; que les voitures peuvent être conduites par des conducteurs et que les conducteurs peuvent conduire plusieurs voitures.

Remarque

Les mots clés in, out et in/out devant un paramètre de méthode permettent de spécifier si le paramètre est une donnée d'entrée, de sortie, ou bien les deux.

Attention

Le but d'une modélisation UML n'est pas de représenter la réalité dans l'absolu, mais plutôt de proposer une vision d'une situation réduite aux éléments nécessaires pour répondre au problème posé. Donc une modélisation s'inscrit toujours dans un contexte, et en cela l'exemple précédent reste limité car son contexte d'application est indéfini.

4. Classe abstraite

Définition : Classe abstraite

Une classe abstraite est une classe non instanciable. Elle exprime donc une généralisation abstraite, qui ne correspond à aucun objet existant du monde.

Attention : Classe abstraite et héritage

Une classe abstraite est toujours héritée. En effet sa fonction étant de généraliser, elle n'a de sens que si des classes en héritent. Une classe abstraite peut être héritée par d'autres classes abstraites, mais en fin de chaîne des classes non abstraites doivent être présentes pour que la généralisation ait un sens.

Syntaxe : Notation d'une classe abstraite en UML

On note les classes abstraites en italique.

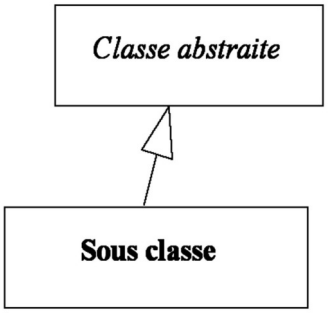


Image 17 Notation d'une classe abstraite en UML

Exemple : Exemple de classes abstraites

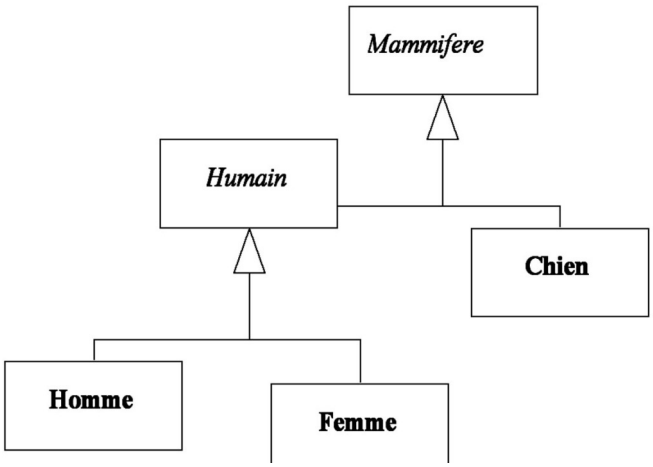


Image 18 Des chiens et des hommes

Dans la représentation précédente on a posé que les hommes, les femmes et les chiens étaient des objets instanciables, généralisés respectivement par les classes mammifère et humain, et mammifère.

Selon cette représentation, il ne peut donc exister de mammifères qui ne soient ni des hommes, ni des femmes ni des chiens, ni d'humains qui ne soient ni des hommes ni des femmes.

B. Lab II

[15 min]

Description du problème

Un laboratoire souhaite gérer les médicaments qu'il conçoit.

- Un médicament est décrit par un nom, qui permet de l'identifier. En effet il n'existe pas deux médicaments avec le même nom. Un médicament comporte une description courte en français, ainsi qu'une description longue en latin. On gère aussi le conditionnement du médicament, c'est à dire le nombre de pilules par boîte (qui est un nombre entier).
- À chaque médicament on associe une liste de contre-indications, généralement plusieurs, parfois aucune. Une contre-indication comporte un code unique qui l'identifie, ainsi qu'une description. Une contre-indication est toujours associée à un et un seul médicament.
- Tout médicament possède au moins un composant, souvent plusieurs. Un composant est identifié par un code unique et possède un intitulé. Tout composant peut intervenir dans la fabrication de plusieurs médicaments. Il existe des composants qui ne sont pas utilisés pour fabriquer des médicaments et que l'on veut quand même gérer.
- Il existe des composants naturels et des composants artificiels. Pour les composants naturels, on gère l'espèce végétale qui porte le composant. Pour les composants artificiels, on gère le nom de la société qui le fabrique.

Exemple de données

Afin de matérialiser notre base de données, nous obtenons les descriptions suivantes :

- Le **Chourix** a pour description courte *« Médicament contre la chute des choux »* et pour description longue *« Vivamus fermentum semper porta. Nunc diam velit, adipiscing ut tristique vitae, sagittis vel odio. Maecenas convallis ullamcorper ultricies. Curabitur ornare. »*. Il est conditionné en boîte de 13.

  Ses contre-indications sont :
  - CI1 : Ne jamais prendre après minuit.
  - CI2 : Ne jamais mettre en contact avec de l'eau.

  Ses composants sont le **HG79** et le **SN50**.
- Le **Tropas** a pour description courte *« Médicament contre les dysfonctionnements intellectuels »* et pour description longue *« Suspendisse lectus leo, consectetur in tempor sit amet, placerat quis neque. Etiam luctus porttitor lorem, sed suscipit est rutrum non. »*. Il est conditionné en boîte de 42.

  Ses contre-indications sont :
  - CI3 : Garder à l'abri de la lumière du soleil

  Son unique composant est le **HG79**.
- Les composants existants sont :
  - **HG79** : "Vif-argent allégé" ; il s'agit d'un composant naturel extrait de l'**edelweiss**.
  - **HG81** : "Vif-argent alourdi" ; il s'agit aussi d'un composant naturel extrait de l'**edelweiss**.
  - **SN50** : "Pur étain" ; il s'agit d'un composant artificiel fabriqué par **Lavoisier et fils SA**.

Question

Réaliser le modèle conceptuel de données en UML du problème.

C. Gestion des défauts

[20 minutes]

Un groupe industriel construisant des moteurs cherche à organiser la gestion des défauts observés sur des moteurs confrontés à des tests en situation réelle. Pour cela un de ses ingénieurs modélise le processus de gestion des défauts, tel qu'il existe actuellement, par le diagramme de classes suivant.

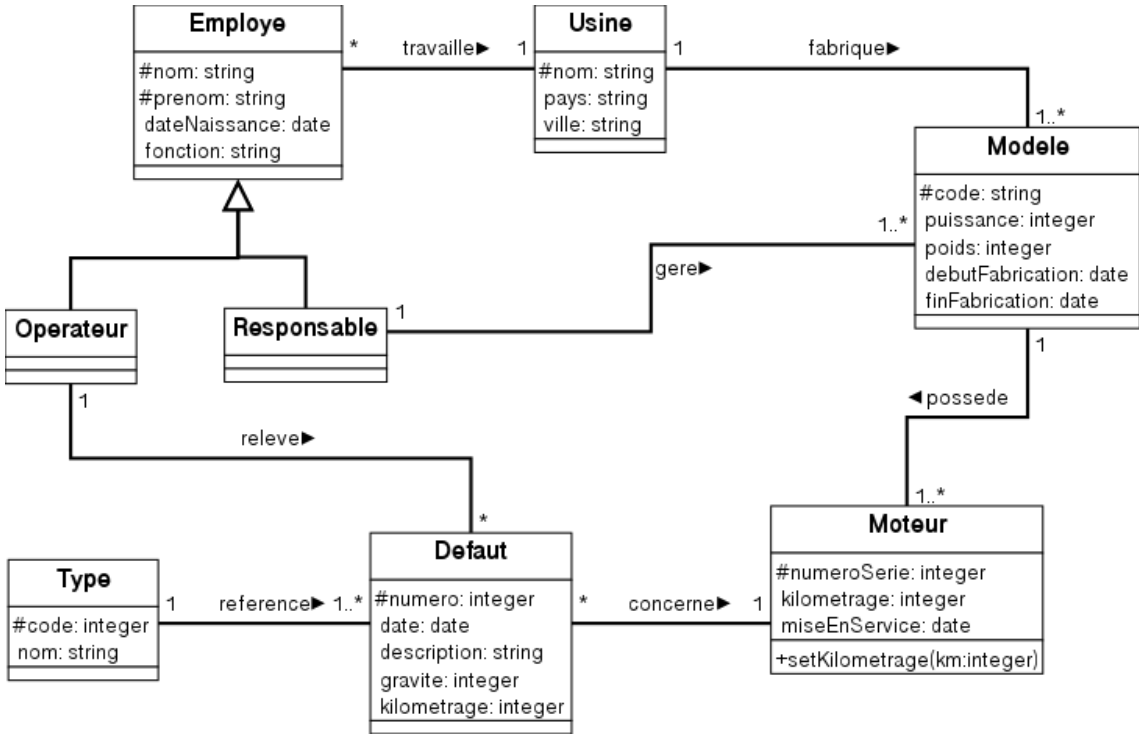


Image 19 Diagramme de classes de gestion des défauts

Question 1

Décrivez en français ce que représente ce diagramme.

Question 2

Étant donné ce modèle, est-il possible de savoir dans quelle usine a été fabriqué un moteur et qui est responsable de sa production ?

Question 3

La responsabilité d'un modèle est elle toujours assumée par un employé travaillant dans l'usine dans laquelle ce modèle est produit ?

Question 4

Pourquoi avoir fait le choix d'une classe **Type** pour codifier les défauts, plutôt qu'un attribut de type énuméré directement dans la classe **Defaut** ?

Question 5

Pourquoi l'attribut `kilométrage` apparaît-il à la fois dans les classes **Defaut** et **Moteur** et pourquoi avoir fait apparaître la méthode `SetKilometrage` ?

Question 6

Ce diagramme permet-il de répondre à la question : Quel est le nombre moyen de défauts rencontrés par les moteurs de chaque modèle ayant été mis en service avant 2000 ? Quelles sont les classes et attributs utiles ?

Question 7

Peut-on également répondre à la question : Quel est le kilométrage moyen des moteurs ayant été concernés par au moins deux défauts d'une gravité supérieure à 5 ?

Modélisation avancée des associations

IV

A. Modélisation avancée des associations en UML

Objectifs

Maîtriser le diagramme de classe UML dans le cas de la conception de BD.

1. Exercice : Entreprise

En analysant le schéma UML ci-après, sélectionner toutes les assertions vraies.

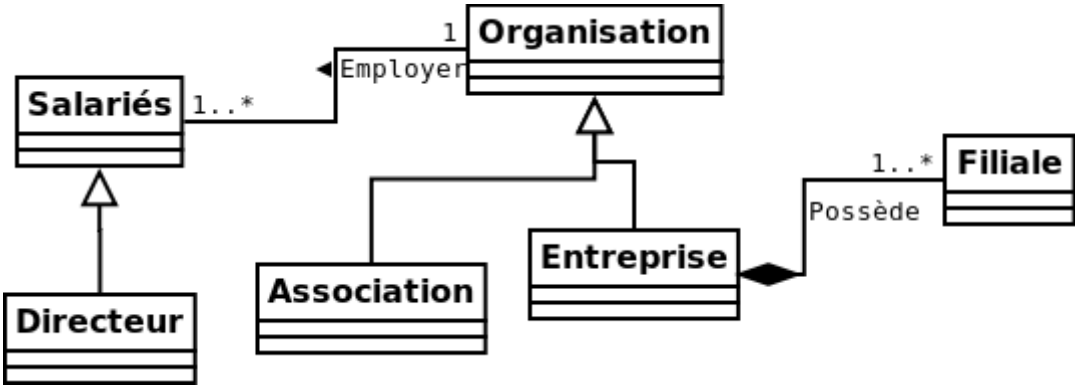


Image 20 MCD UML

☐	Une association peut employer un directeur.
☐	Une association peut employer plusieurs directeurs.
☐	Une association peut ne pas employer de directeur.
☐	Une filiale peut appartenir à plusieurs entreprises.
☐	Il existe des organisations qui ne sont ni des entreprises ni des associations.

2. Composition

Définition : Association de composition

On appelle composition une association particulière qui possède les propriétés suivantes :

- La composition associe une classe composite et des classes parties, tel que tout objet partie appartient à un et un seul objet composite. C'est donc une association 1:N (voire 1:1).
- La composition n'est pas partageable, donc un objet partie ne peut appartenir qu'à un seul objet composite à la fois.
- Le cycle de vie des objets parties est lié à celui de l'objet composite, donc un objet partie disparaît quand l'objet composite auquel il est associé disparaît.

Remarque

- La composition est une association particulière (binaire de cardinalité contrainte).
- La composition n'est pas symétrique, une classe joue le rôle de conteneur pour les classes liées, elle prend donc un rôle particulier a priori.
- La composition est une agrégation avec des contraintes supplémentaires (non partageabilité et cycle de vie lié).

Syntaxe : Notation d'une composition en UML

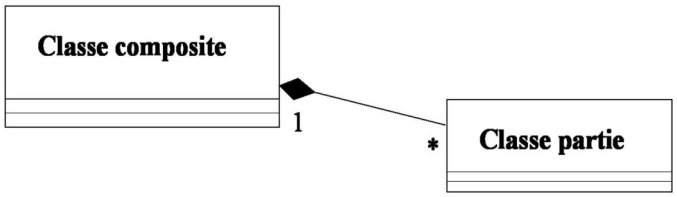


Image 21 Notation de la composition en UML

Attention : Composition et cardinalité

La cardinalité côté composite est toujours de exactement 1.

Côté partie la cardinalité est libre, elle peut être 0..1, 1, * ou bien 1..*.

Exemple : Exemple de composition

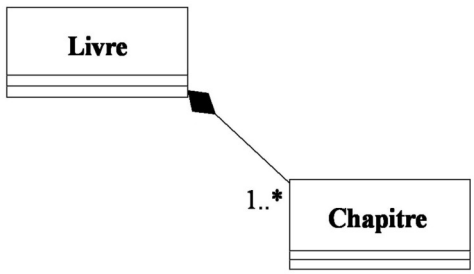


Image 22 Un livre

On voit bien ici qu'un chapitre n'a de sens que faisant partie d'un livre, qu'il ne peut exister dans deux livres différents et que si le livre n'existe plus, les chapitres le composant non plus.

Remarque : Composition et entités faibles

La composition permet d'exprimer une association analogue à celle qui relie une entité faible à une entité identifiante en modélisation E-A✯. L'entité de type faible correspond à un objet partie et l'entité identifiante à un objet composite.

Conseil : Composition et attribut multivalué

- Une composition avec une classe partie dotée d'un seul attribut peut s'écrire avec un attribut multivalué.
- Un attribut composé et multivalué peut s'écrire avec une composition.

Rappel : Voir aussi

- *Attributs*
- *Agrégation*

3. Agrégation

Définition : Association d'agrégation

L'agrégation est une association particulière utilisée pour préciser une relation tout/partie (ou ensemble/élément), on parle d'association **méréologique**.

Syntaxe

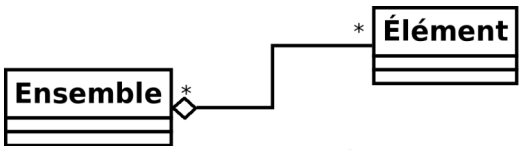


Image 23 Notation de l'agrégation en UML

La cardinalité, peut être exprimée librement, en particulier les instances de la classe Élément peuvent être associées à plusieurs instances de la classe Ensemble, et même de plusieurs classes.

Attention

L'agrégation garde toutes les propriétés d'une association classique (cardinalité, cycle de vie, etc.), elle ajoute simplement une terminologie un plus précise via la notion de tout/partie.

4. Explicitation des associations (sens de lecture et rôle)

Syntaxe : Sens de lecture

Il est possible d'ajouter le sens de lecture du verbe caractérisant l'association sur un diagramme de classe UML, afin d'en faciliter la lecture. On ajoute pour cela un signe < ou > (ou un triangle noir) à côté du nom de l'association

Syntaxe : Rôle

Il est possible de préciser le rôle joué par une ou plusieurs des classes composant une association afin d'en faciliter la compréhension. On ajoute pour cela ce rôle à côté de la classe concernée (parfois dans un petit encadré collé au trait de l'association.

Exemple

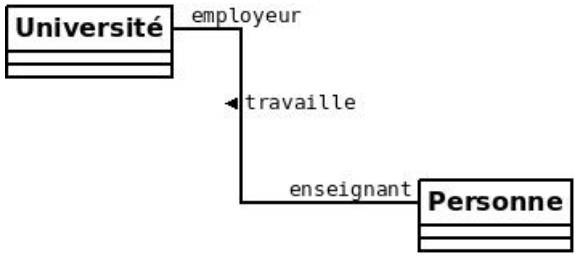


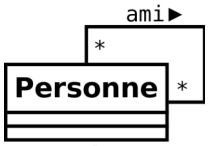
Image 24 Rôle et sens de lecture sur une association

5. Associations réflexives

Définition : Association réflexive

Une association réflexive est une association qui associe une classe avec elle-même.

Exemple



Association réflexive « ami »

Méthode

L'explicitation des associations est souvent utile dans le cas des associations réflexives non symétrique (ou chaque objet ne joue pas le même rôle).

Attention : Auto-association dans les associations réflexives

Une instance peut être associée avec elle-même dans le cas de d'une association réflexive.

Si l'on souhaite exprimer le contraire (une instance peut être associée avec d'autres instances de la même classe, mais pas avec elle-même) :

- **on ajoute une contrainte en UML (par exemple {les personnes ne se marient pas avec elles-mêmes}) ;**
- **que l'on traduira en relationnel par une contrainte du type `AVEC pk ≠ fk` ;**
- **que l'on traduira en SQL par une clause du type `CHECK pk != fk`.**

6. Notion de clé locale dans les compositions et les associations N:M

Le concept de clé locale appartient au niveau conceptuel, il est hérité de l'entité faible du modèle conceptuel Entité-Association (équivalent de la composition en UML). Dans une entité faible ou une composition, une clé de la classe composant est dite locale, car elle ne permet d'identifier l'objet que si l'on connaît la classe composite.

Définition

Dans certaines constructions en UML (association N:M et composition) la clé peut être locale, c'est à dire qu'au lieu d'identifier pleinement un objet (comme une clé classique), elle identifie un objet étant donné un contexte (les membres de l'association N:M ou l'objet composite).

Attention

Une clé locale n'est donc pas une clé au sens relationnel, elle ne permet pas d'identifier un enregistrement, mais elle deviendra une partie d'une clé lors du passage au relationnel.

Attention

Dans une associations N:M on peut avoir des (vraies) clés ou des clés locales. En revanche dans une composition on n'a que des clés locales : en effet si le composant est identifiable indépendamment de son composite, c'est en général qu'il a une vie propre et donc que l'on est pas en présence d'une composition.

7. Classe d'association avec clé locale

Rappel

Classe d'association

Contrainte inhérente à la relation N:M

Dans l'exemple suivant, chaque personne peut avoir un emploi dans plusieurs sociétés, mais elle ne peut pas avoir plusieurs emplois dans une même société.

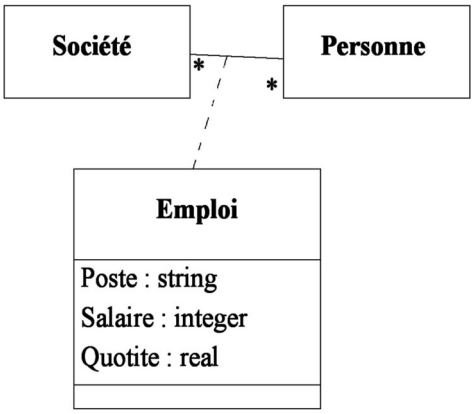


Image 25 Emplois

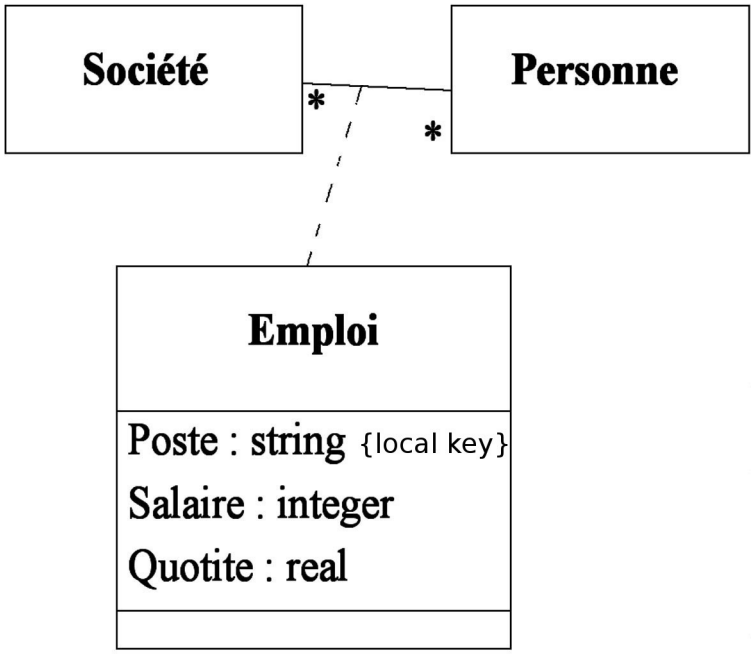
La transformation en relationnelle est cohérente avec cette contrainte.

```
1  Société(...)
2  Personne(...)
3  Emploi(#personne=>Personne, #societe=>Societe, poste:string, salaire:integer,
   quotite:numeric(1,2))
```

#personne	#societe	poste	salaire	quotite
~~Al~~	~~Canonical~~	Directeur	100000	0,5
~~Al~~	~~Canonical~~	Développeur	50000	0,5

Méthode : Intérêt de la clé locale

La spécification d'une clé locale dans emploi, par exemple ici le poste, permet de lever cette contrainte, lorsqu'on le souhaite, en permettant d'identifier chaque instance de l'association, ici l'emploi d'une personne par sa société.



La transformation en relationnelle permettra de maintenir la modélisation, en ajoutant la clé locale à la clé initiale

```
1  Société(...)
2  Personne(...)
3  Emploi(#personne=>Personne, #societe=>Societe, #poste:string, salaire:integer,
   quotite:numeric(7,2))
```

#personne	#societe	#poste	salaire	quotite
Al	Canonical	Directeur	100000	0,5
Al	Canonical	Développeur	50000	0,5

8. Associations ternaires

Syntaxe

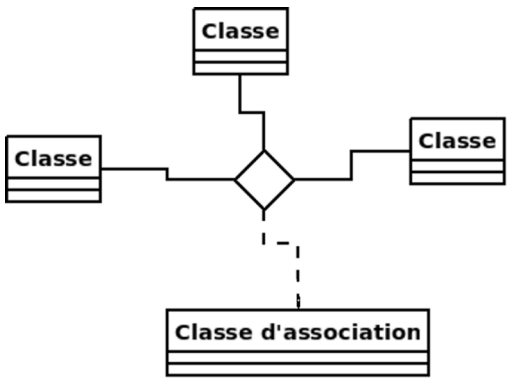


Image 26 Notation d'une association ternaire

Conseil : Ne pas abuser des associations ternaires

Il est toujours possible de réécrire une association ternaire avec trois associations binaires, en transformant l'association en classe.

Conseil : Pas de degré supérieur à 3

En pratique on n'utilise jamais en UML d'association de degré supérieur à 3.

B. Objets Numériques Libres

[30 min]

L'association ONL (Objets Numériques Libres) est une association de promotion des logiciels libres. Elle souhaite exposer sur un site Internet une liste de logiciels libres. Ce site sera adossé à une base de données relationnelle ou relationnel-objet. La première étape de sa démarche est de réaliser un modèle conceptuel représentant ce qu'elle souhaite faire.

- La base de données permet de gérer des applications.

  Les applications sont identifiées par leur nom (LibreOffice, Gimp...) et leur version (1.0, 2.1), et comportent une description courte et une URL. Tous les attributs sont obligatoires. Chaque application a une URL unique.
- La base de données permet de gérer des librairies.

  Les librairies sont des logiciels mais pas des applications. Elles ont les mêmes attributs que les applications (nom, version, description courte, URL), mais les URL ne sont pas nécessairement uniques. Les applications peuvent dépendre de librairies ou d'autres applications, et les librairies peuvent dépendre d'autres librairies (mais pas d'une application).
- La base de données permet de gérer des composants.

  Les composants sont intégrés à une application ou librairie. Les composants ont un code interne à l'application ou la librairie qu'il servent, une version et une description courte, et une URL. Le code et le numéro de version permettent d'identifier localement le composant au sein de la librairie ou de l'application, la description courte et l'URL sont optionnelles.
- La base de données permet de gérer des licences.

  Les applications, librairies et composants sont attachés à une ou plusieurs licences identifiées par leur nom (GPL, MPL...), leur version et leur langue, et comportant le texte intégral de la licence. Les versions des logiciels et licences sont de type "numéro de licence majeur point numéro de licence mineur", comme "1.0" ou "2.2".
- La base de données permet de gérer des catégories.

  Chaque logiciel est rangé dans une catégorie principale, et plusieurs catégories secondaires. Exemple de catégories : bureautique, dessin, multimédia, physique...

Exemple (factice) de données

- Applications :
  - Scenari 4.1, une chaîne éditoriale XML, http://scenari.org, dépend de Libreoffice 4.3 et de ImageMagick 6.8
  - Libreoffice 4.3, une suite bureautique WYSIWYG, http://libreoffice.org
- Librairie :
  - ImageMagick 6.8, permet de créer, de convertir, de modifier et d'afficher des images, http:// imagemagick.org
- Composant :
  - impng 0.2 est un composant de ImageMagick 6.8, permet de compresser une image au format PNG.
- Toutes ces applications, librairies et composants sont disponibles sous une licence LGPL 3.0 et GPL 3.0 françaises.
- Toutes ces applications et librairies sont rangées dans la catégorie principale "document". Scenari est rangé dans la catégorie secondaire "Édition WYSIWYM",

Libreoffice dans la catégorie secondaire "Bureautique", ImageMagick dans la catégorie secondaire "Multimédia".

Question

Réaliser un MCD en UML.

C. Arbre de scène 3D

[30 minutes]

On souhaite pouvoir gérer les différents éléments composant des scènes 3D dans une base de données. Une scène contient des objets qui eux-mêmes peuvent appartenir à plusieurs scènes (au moins une), mais il ne peut y avoir plusieurs fois le même objet dans une même scène. Dans chaque scène, les objets peuvent être visibles ou invisibles. Les scènes et les objets sont identifiés de manière unique par un nom. Une scène peut être active ou inactive. Un objet a comme propriété une position dans l'espace, représentée par un vecteur de réels à trois composantes (x, y, z). Les objets sont organisés de manière hiérarchique : un objet peut être parent de plusieurs objets et chaque objet peut avoir au plus un parent. Des scripts peuvent être associés à un objet ou à une scène (à l'un ou à l'autre mais pas aux deux). Un script est identifié de manière unique par un nom et possède également un attribut permettant de connaître son état (actif ou inactif). Un personnage est un objet particulier qui possède des animations. Une animation est relative à un personnage et est identifiée de manière locale par un nom. À un objet est associé un maillage et celui-ci peut être utilisé par plusieurs objets. Un maillage est identifié de manière unique par un nom et est composé de plusieurs éléments. Chaque élément est relatif à un maillage et est identifié de manière locale par un numéro. Il existe exactement trois types d'élément : des nœuds, des arrêtes et des faces. Une face est délimitée par trois arrêtes et chaque arrête est délimitée par deux nœuds. Plusieurs arrêtes peuvent partager un même nœud et plusieurs faces peuvent partager une même arrête. Afin d'évaluer la complexité d'une scène, on souhaite pouvoir calculer le nombre de faces affichées pour une scène donnée (c'est-à-dire la somme du nombre de faces du maillage associé aux objets visibles de la scène). Un maillage possède plusieurs matériaux identifiés de manière unique par un nom. Un matériau peut être associé à plusieurs maillages. Un matériau est caractérisé par une couleur définie par un vecteur d'entiers à quatre composantes : rouge, vert, bleu, alpha. Un matériau peut posséder une texture et celle-ci peut être utilisée par plusieurs matériaux. Une texture est identifiée de manière unique par un nom et possède comme attribut une image.

Question 1

Proposez une clarification de ce problème. On pourra reclasser les informations par grande catégorie : scène, objets, scripts...

Question 2

Établissez un modèle conceptuel en UML de ce problème.

Indice :

On mobilisera le stéréotype dataType pour représenter les couleurs et les positions.

Questions de synthèse

Quels sont les principaux éléments du diagramme de classes UML ?

Quelles sont les différences et points communs entre la diagramme de classe UML et le modèle E-A étendu ?

A quoi servent les classes abstraites ?

Quand doit-on expliciter des contraintes sur les associations ?

Glossaire

Clé artificielle (surrogate key)

En relationnel une clé artificielle est un attribut artificiel (qui ne représente aucune donnée) ajouté à la relation pour servir de clé primaire. On mobilise cette technique lorsque l'on n'a pas pu ou voulu choisir une clé naturelle pour clé primaire.

Signification des abréviations

- E-A Entité-Association
- OMG Object Management Group
- UML Unified Modeling Language

Bibliographie

[Arribe, 2014] Arribe Thibaut. 2014. *Conception des chaînes éditoriales : documentariser l'activité et structurer le graphe documentaire pour améliorer la maîtrise de la rééditorialisation*. Université de Technologie de Compiègne, Mémoire de Doctorat. *http://ics.utc.fr/~tha* .

[Muller98] Muller P.A., *Modélisation objet avec UML*, Eyrolles, 1998.

[Roques04] Roques Pascal, Vallée Franck. *UML 2 en action : De l'analyse des besoins à la conception J2EE*. ISBN 2-212-11462-1 (3ème édition). Paris : Eyrolles, 2004. 385 p. architecte logiciel.

[Rothenberg et al., 1989] Rothenberg Jeff, Widman Lawrence E, Loparo Kenneth A, Nielsen Norman R. 1989. *The nature of modeling*. Rand. vol.3027.

[Soutou02] Soutou Christian. *De UML à SQL : Conception de bases de données*. Eyrolles, 2002.

Webographie

[w_journaldunet.com(1)] MORLON JÉRÔME, *UML en 5 étapes*, *http://developpeur.journaldunet.com/dossiers/alg_uml.shtml* , 2004.

[w_journaldunet.com(2)] BORDERIE XAVIER, *Cinq petits conseils pour un schéma UML efficace*, *http://developpeur.journaldunet.com/tutoriel/cpt/031013cpt_uml5conseils.shtml* , 2004.

[w_uml.free.fr] *UML en Français*, *http://uml.free.fr* , consulté en 2002.

Index

Contenus annexes

- Transformation des méthodes par des vues

Méthode

Lorsqu'une méthode est spécifiée sur un diagramme UML pour une classe C, si cette méthode est une fonction relationnelle (elle renvoie une unique valeur et elle peut être résolue par une requête SQL), alors on crée une vue qui reprend les attributs de la classe C et ajoute des colonnes calculées pour les méthodes.

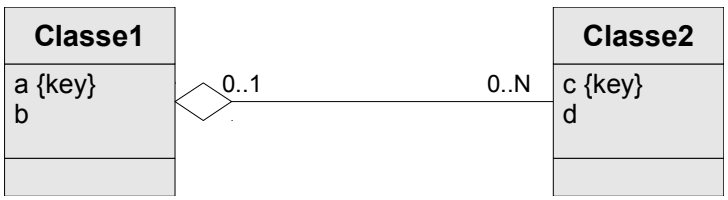
Remarque : Attributs dérivés

Les attributs dérivés étant apparentés à des méthodes, ils peuvent également être gérés par des vues.

- Transformation des agrégations

Rappel : Agrégation

Les associations de type agrégation se traitent de la même façon que les associations classiques.



Graphique 1 Agrégation 1:N

```
Classe1(#a,b)
Classe2(#c,d,a=>Classe1)
```

Graphique 2 Agrégation N:M

```
Classe1(#a,b)
Classe2(#c,d)
Assoc(#a=>Classe1,#c=>Classe2)
```