L'héritage dans la modélisation conceptuelle de données

Table des matières

Introduction

Prérequis :

- Savoir faire un MCD UML avec des classes et des associations simples.

Cours

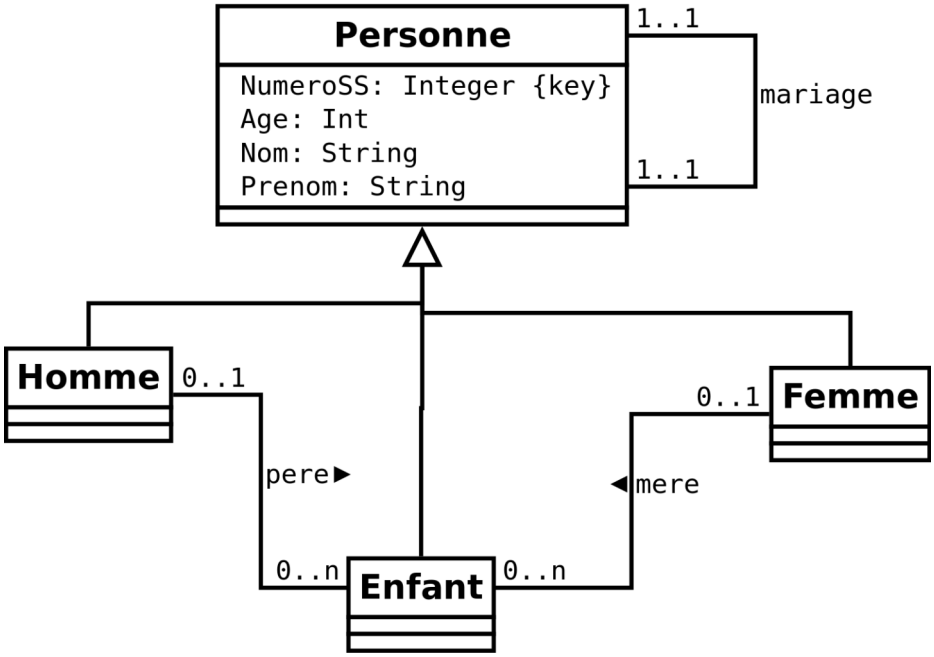
1. Introduction à l'héritage

Objectifs

Savoir utiliser l'héritage lors une modélisation

1.1. Exercice : Mariages

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



- ☐ 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.

1.2. Héritage

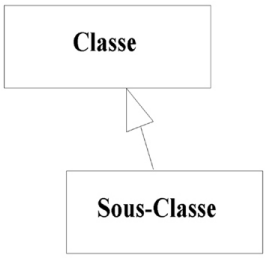
Héritage

Définition

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

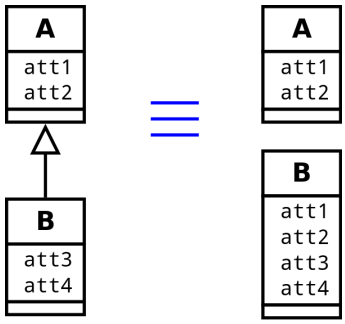


Notation de l'héritage en UML

Factorisation

Fondamental

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.



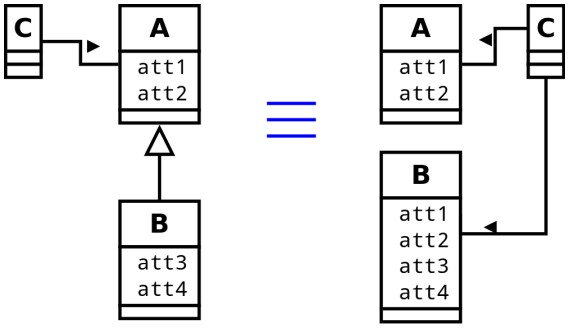
Héritage et factorisation

Is-a

Fondamental

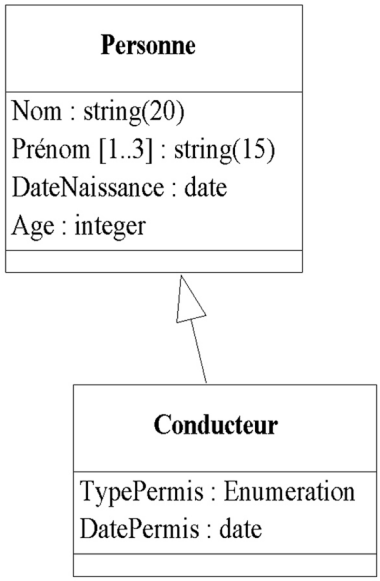
L'héritage permet de représenter la relation "est-un" entre deux objets (*is-a* en anglais).

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"

La classe Conducteur *Exemple*

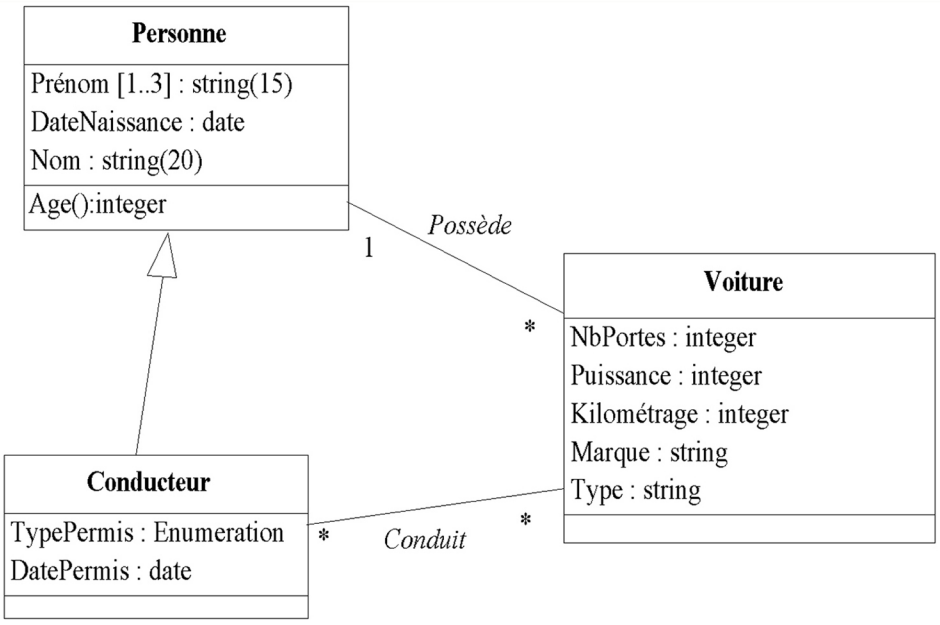


Représentation d'héritage en UML

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.

1.3. Exemple : Des voitures et des conducteurs

Exemple



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.

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.

1.4. Classe abstraite

Classe abstraite — **Définition**

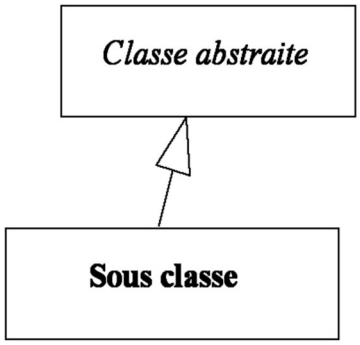
Une classe abstraite est une classe non instanciable. Elle exprime donc une généralisation abstraite, qui ne correspond à aucun objet existant du monde.

Classe abstraite et héritage — **Attention**

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.

Italique — **Syntaxe**

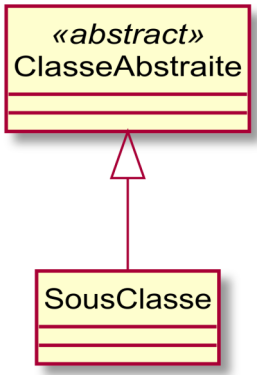
On note les classes abstraites en italique.



Notation d'une classe abstraite en UML

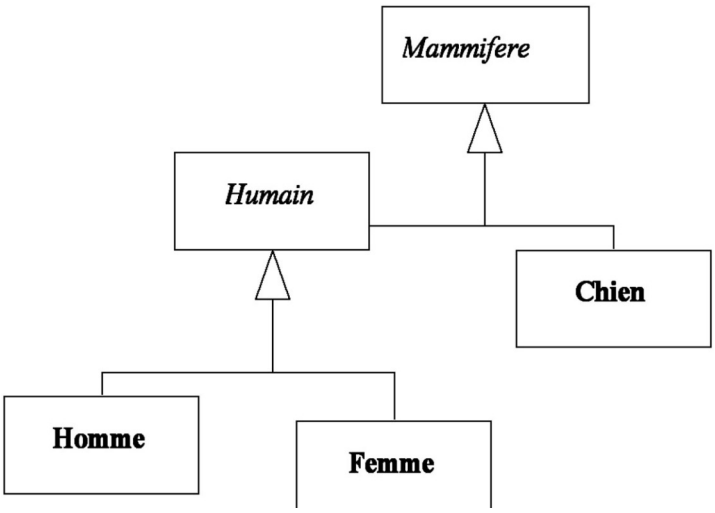
<<abstract>> — **Syntaxe**

On peut utiliser le stéréotype `<<abstract>>` pour noter les classes abstraites (cette représentation est plus évidente à lire).



Notation <<abstract>>

Exemple de classes abstraites *Exemple*



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.

Complément

Stéréotype *(cf. p.15)*

2. Notions avancées pour l'usage de l'héritage en modélisation des BD

Objectifs

Savoir utiliser l'héritage lors une modélisation

2.1. Exercice : 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**.

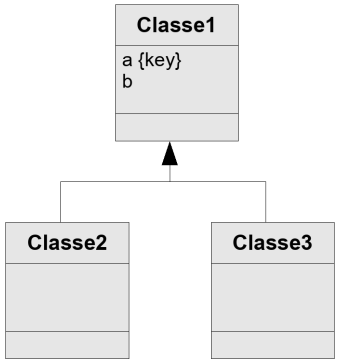
Question

Réaliser le modèle conceptuel de données en UML du problème.

2.2. Héritage complet

Héritage complet et presque complet — ***Définition***

Un héritage est complet si ses classes filles n'ont aucune caractéristiques (attributs, méthodes, associations) propres.



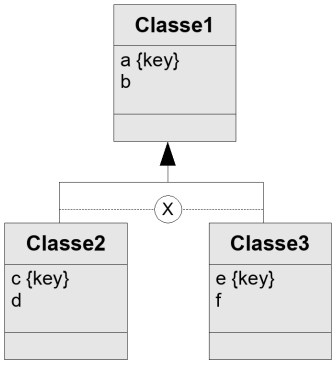
Héritage complet

Un héritage est presque complet si les classes filles ont des méthodes propres, quelques attributs propres, et **aucune association propre**.

2.3. Héritage exclusif

Héritage exclusif *Définition*

Un héritage est exclusif si les objets d'une classe fille ne peuvent appartenir aussi à une autre classe fille. On peut le noter avec la contrainte `{X}` sur le diagramme UML (`{XT}` si la classe mère est abstraite).



Héritage exclusif

Héritage non exclusif *Définition*

L'héritage non exclusif est une forme d'héritage qui exprime qu'un objet peut appartenir à deux classes filles en même temps.

Notation des héritages exclusifs *Syntaxe*

Normalement la notation UML par défaut désigne un héritage non exclusif.

Mais l'héritage exclusif étant plus commun, on conviendra plutôt qu'un schéma UML qui ne comporte que des héritages non contraints exprime par défaut des héritages exclusifs.

Une note peut être ajoutée précisant que tous les héritages sont exclusifs pour lever toute ambiguïté.

En revanche un schéma qui comporte explicitement des contraintes sur certains héritages est pas sur d'autres permet effectivement d'exprimer de l'héritage non exclusif.

Une dernière possibilité, moins standard, est de marquer d'une contrainte de type {NON X} un héritage non exclusif.

On évitera l'héritage non exclusif pour la conception de BD *Méthode*

Bien que cette forme d'héritage soit autorisée en modélisation conceptuelle de bases de données et en UML, on évitera de la mobiliser, sauf en cas d'apport vraiment important d'expressivité, car elle a tendance à complexifier la modalisation, que ce soit au niveau de son interprétation humaine ou de son implémentation en machine.

Il est toujours possible d'exprimer différemment un héritage non exclusif :

- en multipliant les classes filles pour les singulariser,
- et/ou en utilisant la composition pour factoriser les parties communes.

2.4. Héritage multiple

Héritage multiple — *Définition*

L'héritage multiple est une forme d'héritage qui exprime qu'une classe fille hérite de plusieurs classes mères.

Notation des héritages multiples — *Syntaxe*

On conseillera d'ajouter une note lors de l'utilisation d'un héritage multiple, expliquant le choix de modélisation et actant qu'il ne s'agit pas d'une erreur.

On évitera l'héritage multiple pour la conception de BD — *Méthode*

On évitera d'utiliser l'héritage multiple en modélisation de bases de données, sauf en cas d'apport vraiment important d'expressivité.

Il est toujours possible de remplacer un héritage multiple par plusieurs héritages simples.

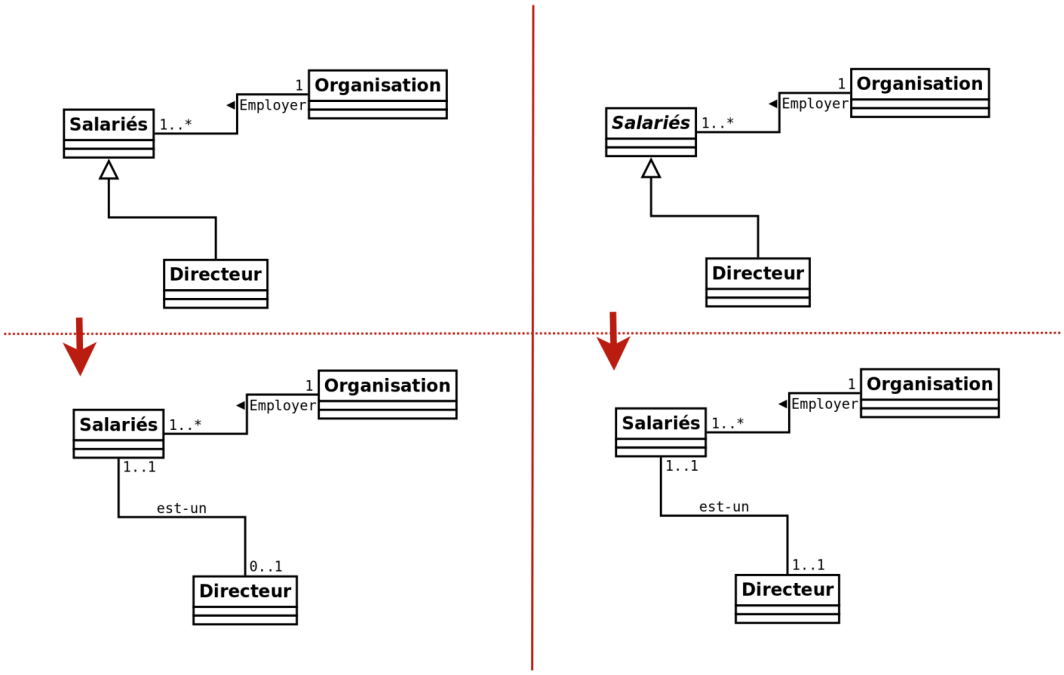
2.5. Équivalence entre association d'héritage et association 1:1

Remarque

Une association d'héritage est un cas particulier d'association 1:1.

Elle ajoute une sémantique de type "est-un".

Exemple



2.6. Exercice : 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.

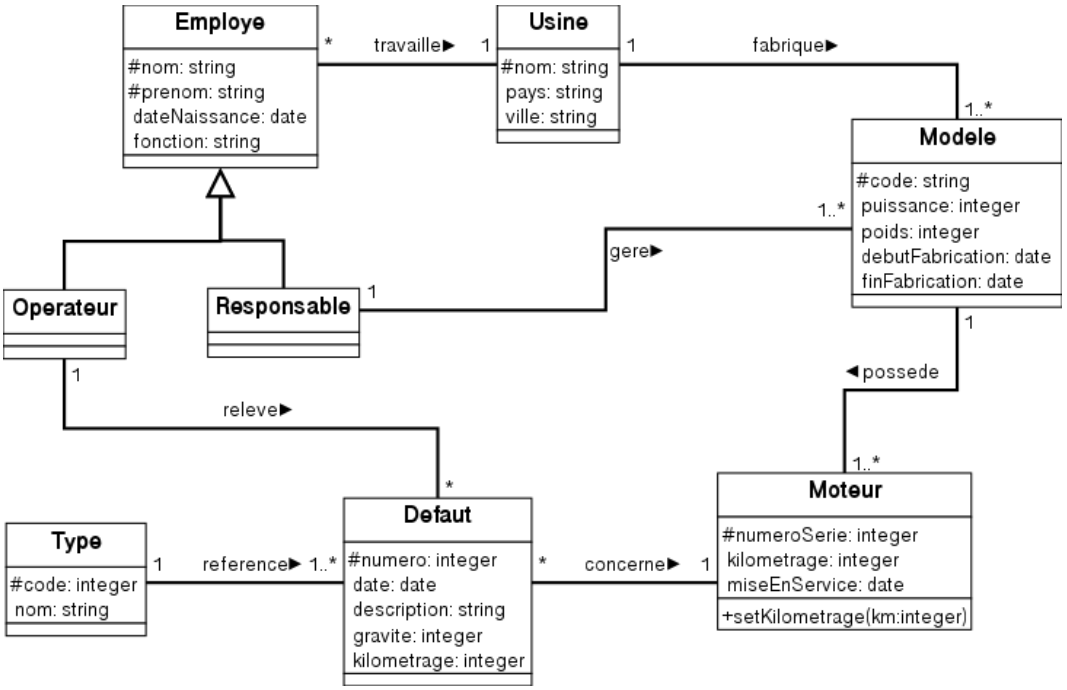


Diagramme de classes de gestion des défauts

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 ?

Exercices

1. Exercice : Armoires secrètes

[30 minutes]

Dans le cadre de la réalisation d'une base de données pour les services secrets français, vous disposez de l'analyse de besoins suivant :

- les agents secrets sont identifiés par un code sur 3 chiffres (comme *007*) et possède un nom et un prénom ;
- les agents secrets produisent des rapports, parfois seul, parfois à plusieurs. Tous les agents secrets ont produit au moins un rapport ;
- les agents secrets sont communément appelés par leurs initiales et leur code : ainsi James Bond 007 est en général appelé *JB007* ;
- un rapport est identifié par un titre (il n'existe pas deux rapports avec le même titre) et il possède une description ainsi que des mots-clés (au moins 2, au plus 10) ;
- le rangement des rapports est organisé comme suit : les rapports sont situés dans des dossiers, qui sont classés dans des casiers, qui sont rangés sur des étagères, dans des armoires ;
- les dossiers, casiers, étagères et armoires sont des rangements, qui sont identifiés par une lettre et un nombre (inférieur à 100). Chaque rangement a une capacité qui détermine le nombre de rangements qu'il peut contenir ;
- en dehors des armoires, il n'existe pas de rapport ou de rangement qui ne serait rangé nulle part.

Question

Proposez un *MCD*∗ en *UML*∗ de ce problème. L'on cherchera le modèle le plus expressif possible.

On fera apparaître les types des attributs, en étant le plus précis possible avec les informations dont nous disposons, ainsi que les clés.

2. Exercice : Capitaine Krik

[45 min]

Le Capitaine Krik a pour tâche de développer une base de données sur les vaisseaux spatiaux et les équipages de la TarFleet.

La flotte ne possède que trois types de vaisseaux : les croiseurs, les frégates et les chasseurs. Chaque membre de la TarFleet a un nom, un prénom, une date de naissance, une planète d'origine et un numéro d'identifiant unique. Certains membres sont aussi soit pilotes, soit capitaines : les pilotes ont un nombre de chasseurs ennemis abattus, tandis que les capitaines ont un certain nombre d'étoiles (entre zéro et cinq).

Krik veut savoir dans la base de données quelles personnes sont affectées à quel vaisseau. Dans la flotte, un chasseur a pour équipage un unique pilote, une frégate a deux pilotes et cinq autres membres d'équipage, tandis qu'un croiseur a un capitaine et de nombreux autres personnels. Les croiseurs peuvent aussi transporter dans leur hangar plusieurs chasseurs (leurs pilotes ne sont alors pas comptés dans l'équipage du croiseur).

Pour chaque vaisseau, on veut connaître son nom et son identifiant, sachant que celui-ci est généré automatiquement à partir du nom et du type de vaisseau (par exemple "EnterpriseCruser"). Pour les frégates et les croiseurs, on veut également connaître la puissance de leur bouclier (les chasseurs n'en sont pas équipés), et, pour un croiseur, le nombre de membres d'équipage maximal qu'il peut accueillir.

Krik veut aussi des informations sur les réacteurs de chaque vaisseau. Les réacteurs sont soit à fission nucléaire, soit à trou noir miniature. Chacun a un numéro d'emplacement (qui indique où le moteur est monté sur le vaisseau), un poids et une poussée, mais les réacteurs à fission ont également une quantité maximale de carburant, tandis que les réacteurs à trou noir ont une puissance critique.

Question

Proposer un schéma UML permettant de modéliser une telle base de données.

Contenus annexes

1. Stéréotype

Stéréotype UML *Définition*

Un stéréotype UML est une syntaxe permettant d'ajouter de la sémantique à la modélisation des classes. Il permet de définir des **types de classe**, afin de regrouper conceptuellement un ensemble de classes (à l'instar d'une classe qui permet de regrouper conceptuellement un ensemble d'objets).

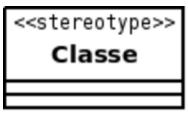
C'est une mécanique de méta-modélisation : elle permet d'étendre le méta-modèle UML, c'est à dire le modèle conceptuel du modèle conceptuel.

Méta-modèle *Définition*

Un méta-modèle est le modèle d'un modèle. Par exemple le méta-modèle UML comprend les concepts de classe, attribut, association, cardinalité, composition, agrégation, contraintes, annotations, ... On mobilise ces concepts (on les instancie) pour exprimer un modèle particulier suivant le formalisme UML.

Les stéréotypes permettent donc d'ajouter au méta-modèle UML standard, celui que tout le monde utilise, des concepts locaux pour enrichir le langage de modélisation que l'on utilise pour réaliser des modèles.

Syntaxe



Notation d'un stéréotype en UML

Stéréotypes spécifiques et stéréotypes standard *Conseil*

Un stéréotype spécifique enrichit le méta-modèle UML, mais selon une sémantique qui est propre à celui qui l'a posé, non standard donc. La conséquence est que pour un tiers, l'interprétation du stéréotype n'est plus normalisée, et sera potentiellement plus facilement erronée. Il convient donc de ne pas abuser de cette mécanique.

Deux ou trois stéréotypes spécifiques, correctement définis, sont faciles à transmettre, plusieurs dizaines représenteraient un nouveau langage complet à apprendre pour le lecteur du modèle.

Il existe des stéréotypes fournit en standard par UML, ou communément utilisés par les modélisateurs. L'avantage est qu'il seront compris plus largement, au même titre que le reste du méta-modèle (ils ont une valeur de standard).

Abréviations

MCD : Modèle Conceptuel de Données
UML : Unified Modeling Language

Index

Crédits des ressources

Diagramme de classes de gestion des défauts p. 12
http://creativecommons.org/licenses/by-sa/4.0/fr/, Stéphane Crozat