Prérequis sur la modélisation UML — Diagramme de classes

Denis Conan

avec Chantal Taconet et Christian Bac

CSC4102

Télécom SudParis Janvier 2025

Table des matières

Grille d'auto-évaluation des prérequis sur le diagramme de classes UML

Voici la grille d'auto-évaluation de compétences *a priori* acquises dans les modules CSC3601 et PRO3600 sur le diagramme de classes UML. Si certaines notions de la grille ne vous semblent pas acquises, prenez connaissance des pages qui suivent. En outre, deux exercices sont proposés en dernière page.

Notions / concept/ élément de modélisation	**0**	**1**	**2**	**3**	**4**	**5**
Classe	X	X	X	X		
Attribut	X	X	X	X		
Attribut dérivé	X	X	X	X		
Opération	X	X	X	X		
Association	X	X	X	X		
Multiplicité	X	X	X	X		
Classe d'association	X	X	X			
Agrégation	X	X	X			
Généralisation spécialisation	X	X	X	X		
Visibilité	X	X	X			

Table 1 : Grille d'auto-évaluation des prérequis — diagramme de classes UML

2

Sommaire

Les diapositives qui suivent sont très importantes pour comprendre les concepts de base de l'orientation objet. Les principes de base présentés dans cette section sont complétés en cours par des concepts un peu plus avancés.

Le diagramme de classes décrit les classes et les relations entre les classes. En bref, les classes possèdent des attributs et des opérations. Les relations entre classes peuvent être de différents types : l'association (binaire), la généralisation spécialisation ou héritage, l'agrégation, etc. Une association possède un sens de lecture du verbe la nommant, une navigabilité indiquant dans quel sens l'association peut être parcourue. Chaque extrémité de l'association possède un rôle spécifiant le rôle de la classe dans l'association par rapport à l'autre extrémité, ainsi qu'une multiplicité.

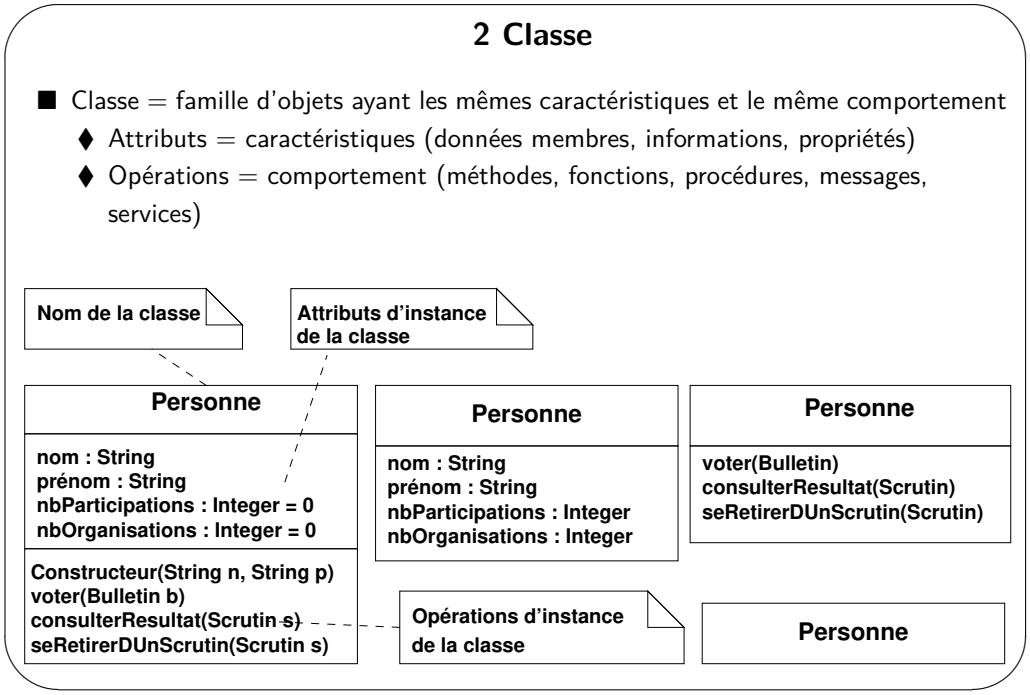
3

1 Modéliser la structure logique du système dans un diagramme de classes

- Diagramme au cœur de la conception orientées objet
- Abstraction
  - Abstraire = ignorer / cacher des caractéristiques non significatives
  - Ne garder que les caractéristiques d'une classe importantes pour le lecteur
- Encapsulation comme mécanisme d'abstraction
  - Cacher des détails en les rendant « privés » (non visibles)
    - P. ex., en conception préliminaire, modéliser ce qui est « public » (visible) Puis, en conception détaillée, ajoutés les éléments « privé »
  - Protéger la conception préliminaire des changements effectués lors de la conception détaillée dans la partie « privée » en ne remettant pas en cause ce qui est exposé « publiquement » au niveau métier
    - Par exemple, la conception préliminaire indique que a est un ensemble alors que la conception détaillée précise que c'est une liste doublement chaînée

Les éléments de base de la structure logique du système sont les classes. La définition d'une classe contient les détails d'un élément de cette structure qui sont importants pour l'intervenant et le système modélisé. Retirer des détails non significatifs à une étape donnée du développement s'appelle l'abstraction. Les diagrammes de classes de la conception préliminaire sont plus abstraits que les mêmes diagrammes repris et raffinés lors de la conception détaillée.

Très liée au concept d'abstraction, l'encapsulation est une particularité de l'orientation objet. L'encapsulation permet à chaque élément structurel de base, c'est-à-dire à chaque classe, de cacher des détails, soit des données contenues dans l'élément soit des actions possibles sur l'élément. Ainsi, une classe peut n'exposer que certaines de ses caractéristiques. L'encapsulation est très importante car elle permet de passer de manière gracieuse de la conception préliminaire à la conception détaillée en exposant tout d'abord les caractéristiques pertinentes pour le métier de l'application et ensuite en montrant aux développeurs des caractéristiques particulières donnant les indications sur les aspects techniques. Cela protège la conception préliminaire des changements effectués lors de la conception détaillée dans la partie « privée » en ne remettant pas en cause ce qui est exposé « publiquement » au niveau métier.



Les classes décrivent les différents types d'objets[1] que le système possède. Une classe est un type de quelque chose. Vous pouvez penser à une classe comme à un modèle (au sens patron) à partir duquel les instances ou objets[2] conformes au type défini par la classe sont créés.

La description de la classe inclut deux catégories d'informations : l'état définissant les informations que les objets de la classe contiennent et le comportement que les objets de la classe autorisent (au sens « services » qu'ils rendent). La conjonction de l'état et du comportement caractérise l'orientation objet par rapport aux autres approches en génie logiciel, par exemple qui séparent les données des traitements. Ainsi, en orienté objet, le comportement d'un objet d'une classe définit les transformations sur l'état de l'objet de la classe. Une classe peut être vue comme le couplage d'une structure de données et de procédures utilisant la structure. L'état d'un objet d'une classe est défini par un ensemble d'attributs et le comportement de l'objet par un ensemble d'opérations[3] sur ses attributs. Un attribut est une information portée par un objet. Par convention, le nom d'un attribut commence par une lettre minuscule. Une opération spécifie une transformation de l'état de l'objet. Elle possède un nom (par convention, commençant par une lettre minuscule), une éventuelle liste de paramètres et un éventuel type de retour. Par conséquent, une classe rassemble la spécification d'objets possédant les mêmes attributs et les mêmes opérations.

Grâce aux mécanismes d'encapsulation, lors de l'analyse, les premiers attributs et les premières opérations sont spécifiés. D'autres attributs et d'autres opérations peuvent être ajoutés par la suite lors de la conception (pour prendre en compte une préoccupation technique comme la persistance), voire lors de l'implantation (pour par exemple optimiser un comportement [par exemple, la recherche dans un ensemble de très grande taille]).

La notation UML autorise la représentation d'une classe uniquement avec son nom, ou avec son nom et ses attributs, ou avec son nom et ses opérations, ou encore avec les trois caractéristiques. Lorsque des attributs ou des opérations ne sont pas présents, cela signifie que celui qui a construit le diagramme visualisé estime que les caractéristiques non représentées ne sont pas assez pertinentes dans la vue qu'il a choisi. (Il est bien sûr possible de discuter et contester les choix de l'auteur du diagramme ; et cela signifie que le modèle sous-jacent contient peut-être d'autres attributs ou opérations, peut-être visualisés dans d'autres diagrammes.)

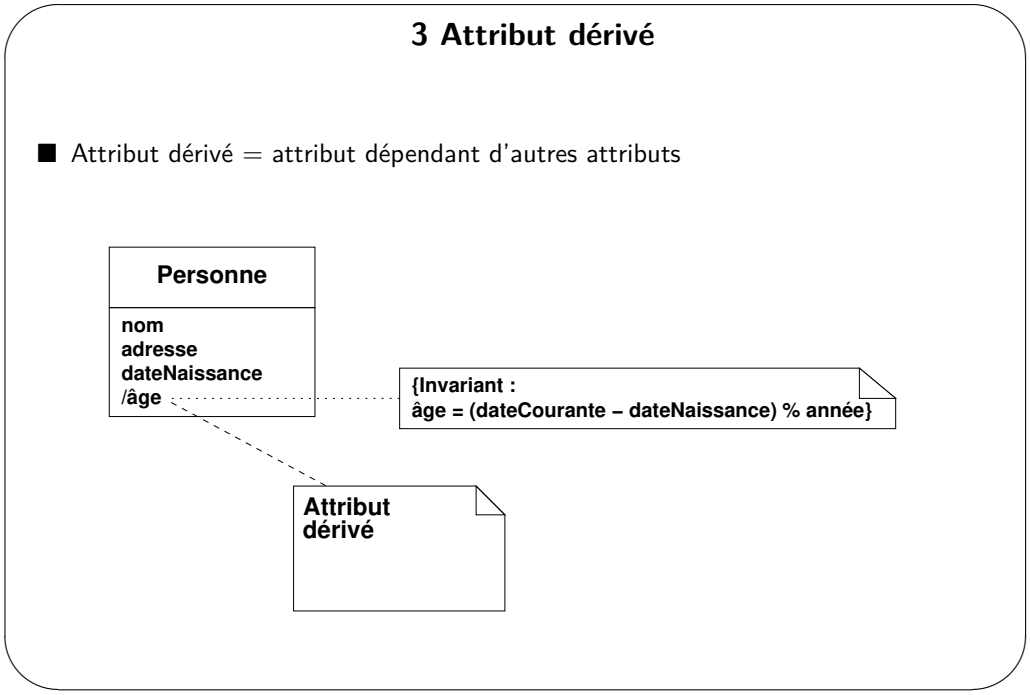
Un objet est une instance d'une classe. Le mécanisme qui crée des objets à partir (de la spécification) d'une classe s'appelle l'instanciation. Autrement dit, un objet est une variable conforme à un type, qui est une classe. Les opérations de l'objet utilisent les valeurs des attributs de l'objet. En outre, un objet possède

1. Le terme « objet » est ici pris dans son acception générale.
2. Cette fois-ci, au sens orienté objet du terme.
3. Aussi appelées des méthodes dans les langages orientés objet.

une identité qui permet de le distinguer des autres objets. On accède à un objet via une référence qui pointe sur l'objet. Ainsi, dans l'instruction « Personne j ; », j est déclarée comme étant une référence sur un objet de type Personne.

Cf. le glossaire pour la définition des termes « classe », « attribut », et « opération », et « objet ».

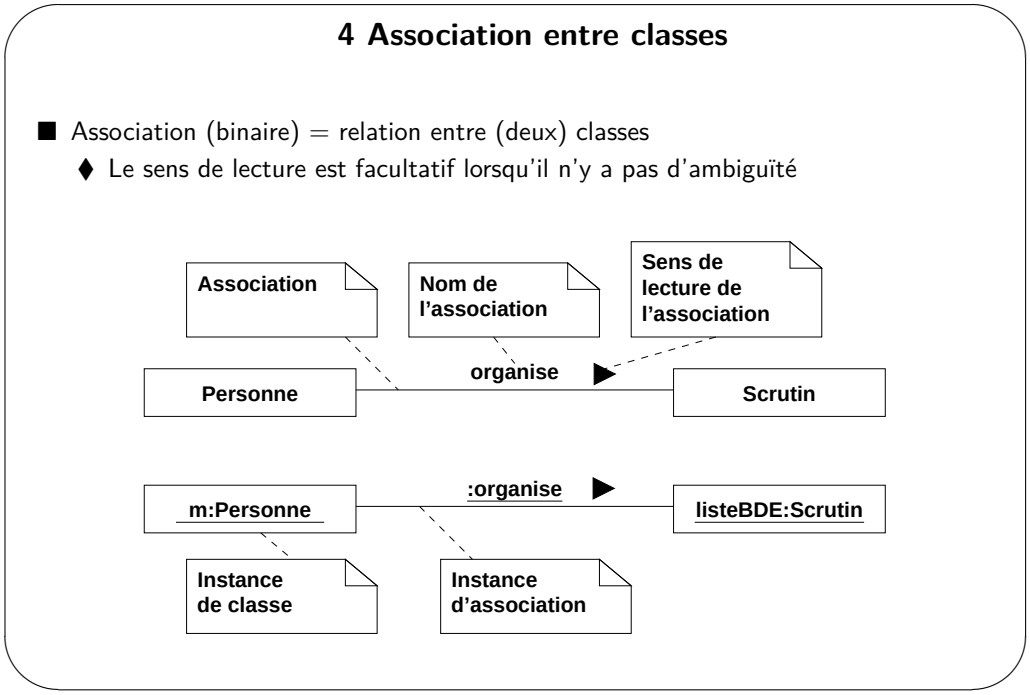
5



Dans les diagrammes de classes, il est important que les informations ne soient pas redondantes. Une forme de redondance est un attribut dérivé dont la valeur se calcule à partir des valeurs d'un ou plusieurs autres attributs. Dans ce cas, l'attribut dérivé doit être signalé en préfixant le nom de l'attribut avec une barre oblique « / » afin d'indiquer au lecteur que cet attribut est redondant. C'est une indication importante pour ceux qui programment l'application car les attributs dérivés demandent un traitement spécial : leur valeur change à chaque fois que la valeur des attributs à partir desquels ils sont calculés change. Dans notre exemple, le traitement est même un peu plus complexe car la valeur de l'attribut « âge » évolue aussi en fonction d'un paramètre indépendant de l'application : la date du jour.

Cf. le glossaire pour la définition du terme « élément dérivé ».

6



Une association est une relation entre classes qui exprime les relations entre leurs instances. Une association permet de naviguer entre classes. Une association possède un nom constitué classiquement d'un verbe conjugué (la plupart du temps au présent). Le sens de lecture permet de lire l'association en prenant le nom de la classe d'origine comme sujet, le nom de l'association comme verbe, et le nom de la classe destinatrice comme complément d'objet direct. Dans notre exemple, cela donne « une `Personne` `organise` un `Scrutin` ». Dans l'autre sens, il suffit souvent de mettre le verbe au passif. Dans notre cas, nous pouvons dire quelque chose comme « un `Scrutin` est organisé par une `Personne` ».

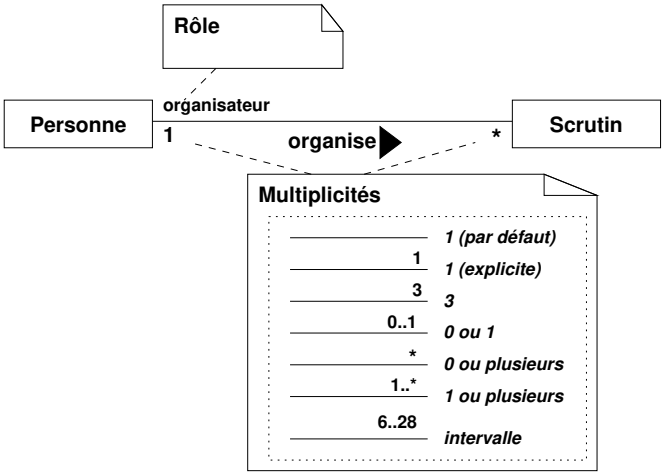
Notez que la partie haute de la figure est un extrait de diagramme de classes (avec deux classes et une association les reliant) alors que la partie basse est un extrait d'un diagramme d'objets (avec deux objets ou instances de deux classes différentes et une instance d'association). Remarquez que les instances sont soulignées et peuvent posséder un nom (p.ex. `m` et `n` dans l'exemple), qui est inséré en tant que préfixe (avant le caractère « : ») au nom de la classe ou de l'association.

Cf. le glossaire pour la définition du terme « association ».

7

5 Nom de rôle et multiplicité

- Nom de rôle = indication sur la participation de la classe à l'association
- Multiplicité = contrainte sur le nombre d'objets associés



Dans les diagrammes de classes, il est possible de nommer les extrémités des associations, ce sont des noms de rôle, lorsqu'il existe un risque de confusion quant à la participation d'une classe à une association. Les noms de rôle sont donc indiqués par exemple lorsqu'ils ne sont pas évidents ou lorsque la classe participe à plusieurs associations, tout spécialement avec la même classe. Les noms de rôle sont donc une information marquant visuellement une information importante : il ne faut pas en abuser.

Les associations relient des classes. Chaque extrémité d'une association possède une multiplicité ou cardinalité. La multiplicité est placée de l'autre côté de la classe utilisée comme sujet, du côté du complément d'objet direct. Dans notre exemple, une personne organise 0 ou plusieurs scrutins, et un scrutin est organisé par 1 personne. La diapositive donne les multiplicités possibles. La notation est la suivante : min..max, où

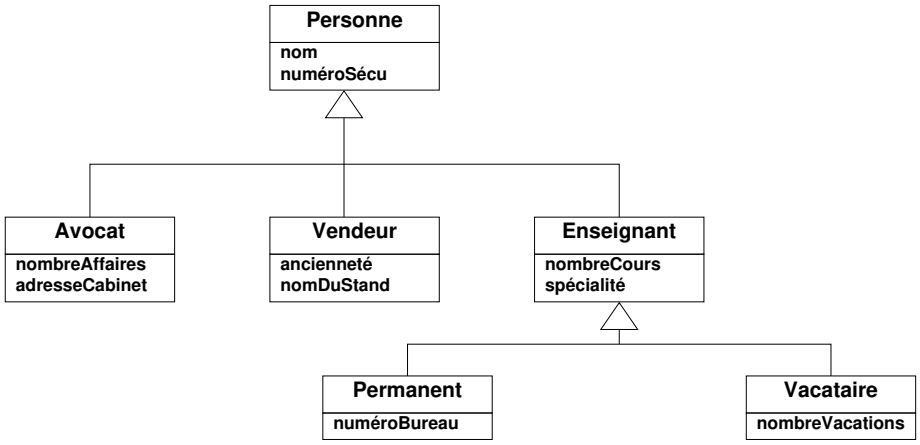
- max peut être omis, auquel cas la multiplicité est exactement min ;
- max peut prendre la valeur « * » qui signifie une limite non bornée (par exemple « 2..* » signifie 2 ou plus) ;
- la notation « 0..* » peut être abrégée en « * » ;
- la multiplicité 1 est généralement omise pour des raisons de lisibilité.

Cf. le glossaire pour la définition des termes « association binaire », « rôle », « multiplicité », « cardinalité », et « association n-aire ».

8

6 Généralisation spécialisation ou héritage

- Factorisation de caractéristiques (attributs et opérations)
  - EST-UN, EST-UNE-SORTE-DE



Le but de la généralisation est la factorisation des caractéristiques de plusieurs classes appelées classes enfants[1] dans une classe appelée classe parente[2]. Cette mise en commun des attributs (structures de données) et des opérations (services) est le mécanisme de réutilisation le plus important de l'orientation objet. La factorisation intervient aussi bien en définissant une partie d'état commune (attributs) qu'en extrayant les comportements communs (opérations). La classe parente rassemble donc les attributs et les opérations communes des classes initiales qui deviennent des classes enfants. La généralisation spécialisation s'appelle aussi un héritage et exprime intuitivement le fait que les classes enfants « héritent » des caractéristiques de leurs classes parentes. La généralisation spécialisation exprime une relation de type « est-un » ou « est-une-sorte-de ». En outre, le terme « généralisation » exprimant la factorisation des caractéristiques communes dans la classe parente, le terme « spécialisation » exprime quant à lui le fait que les classes enfants possèdent les caractéristiques de leurs classes parentes avec la possibilité de les spécialiser soit par des ajouts soit par des redéfinitions. En d'autres termes, une classe enfant possède les capacités de sa classe parente plus quelques autres.

La figure présente un exemple d'arbre de généralisation spécialisation et sous-typage :

- un objet `Personne` peut désigner une instance des classes `Personne`, `Avocat`, `Vendeur`, `Enseignant`, `Vacataire` et `Permanent` ;
- un objet `Enseignant` peut désigner une instance des classes `Enseignant`, `Vacataire` et `Permanent` ;
  - il ne peut pas désigner une instance des classes `Personne`, `Avocat` et `Vendeur`.

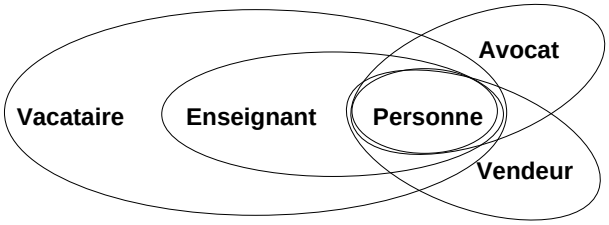
Cf. le glossaire pour la définition des termes « généralisation spécialisation », « héritage ».

1. Ou sous-classes, ou encore classes dérivées.
2. Ou super-classe.

6.1 Généralisation spécialisation : vision enregistrement

9

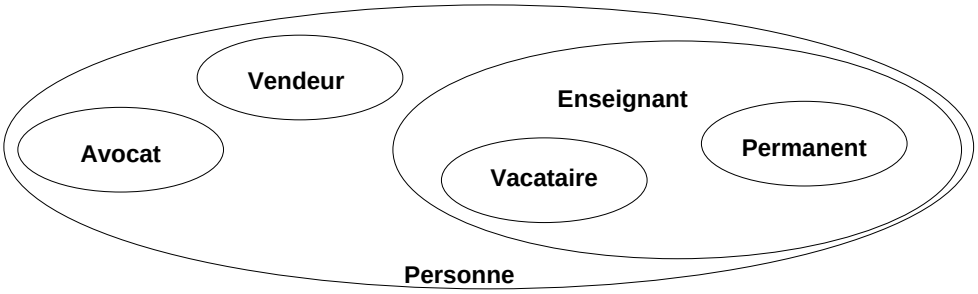
- Objet $\triangleq$ enregistrement (en anglais, *record*)
  - un objet de type `Vacataire` possède toutes les caractéristiques (attributs et opérations) du type `Enseignant`, qui contient lui-même toutes les caractéristiques (attributs et opérations) du type `Personne`.



6.2 Généralisation spécialisation : vision modèle

10

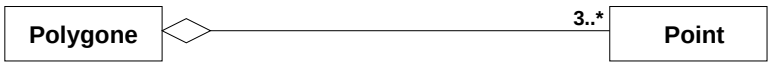
- Type $\triangleq$ x est du type $T \equiv x \in T$
  - L'ensemble des `Personne` contient les ensembles des `Avocat`, des `Vendeur`, des `Enseignant`, etc.



11

7 Agrégation

- Agrégation = une association exprimant un couplage fort lié à une relation de subordination
  - A-UN, EST-UNE-PARTIE-DE
- Elle est asymétrique du type « ensemble / élément » ou « contenant / contenu »
- Règles permettant de choisir une agrégation :
  - Est-ce une partie de ?
  - Les opérations appliquées à l'ensemble sont-elles appliquées à l'élément ?
  - Les changements d'états sont-ils liés ?
- Attention :
  - Un élément agrégé peut être lié à d'autres classes
  - La suppression de l'ensemble n'entraîne pas celle de l'élément



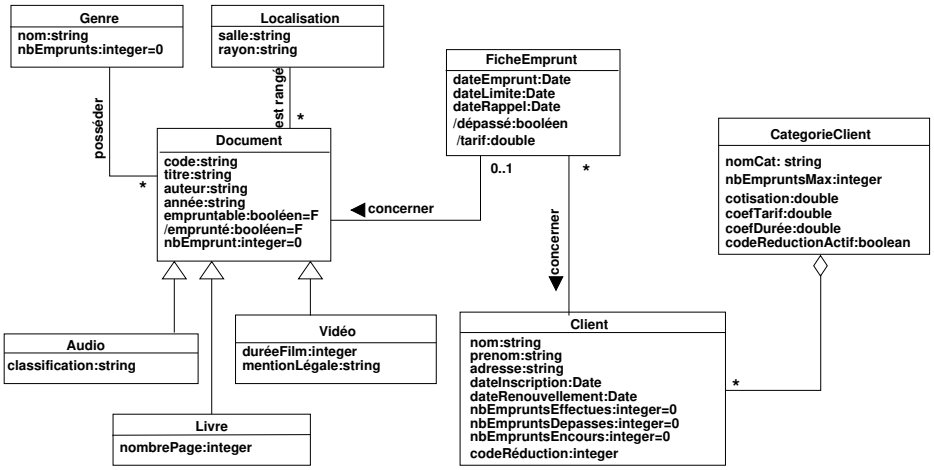
Dans les diagrammes de classes, nous sommes souvent appelés à utiliser des associations que nous nommerions `contient` ou `appartient à`. En fait, cet usage est tellement fréquent que UML a défini une notation spécifique pour ce type d'association, l'agrégation. Une agrégation exprime une relation « contenant / contenu » ou « ensemble / élément », ou encore « agrégé / agrégat ». Par analogie avec la généralisation spécialisation qui se repère par la relation « est un », l'agrégation se repère par la relation « est une partie de » ou « a un » et le couplage fort entre l'ensemble et les éléments. Par exemple, déplacer un `Polygone` revient à déplacer ses `Points`. Il est important de noter que l'agrégation autorise qu'un élément de l'ensemble appartienne à un autre ensemble. Nous verrons plus tard que ce n'est pas le cas de la composition, une relation exprimant une relation de couplage encore plus fort : un élément appartient exclusivement à un ensemble.

Cf. le glossaire pour la définition du terme « agrégation ».

8 Exercices

12

■ Exercez-vous à reconnaître tous les éléments de modélisation du diagramme de classes qui suit.



13

■ Exercez-vous à proposer un diagramme de classes correspondant à la description suivante

- ♦ Un système de gestion de fichiers est représenté par un graphe acyclique correspondant à un ensemble de nœuds. Les nœuds intermédiaires sont appelés répertoires et les feuilles fichiers. Les fichiers considérés contiennent des données de type texte.
- ♦ Un utilisateur nomme les nœuds avec des mots simples (pas de nom composé), il ne peut ainsi nommer que des nœuds du répertoire courant.
- ♦ Chaque répertoire contient au moins deux entrées : une entrée « `.` » pour son propre répertoire et une entrée « `..` » pour le répertoire parent.
- ♦ Chaque nœud possède en outre une date de création, une date d'accès et une date de modification.