Fiche 7 : Référence simplifiée pour la notation UML

La maîtrise de la notation UML (*Unified Modeling Langage*) ne faisant pas partie des objectifs pédagogiques de cet enseignement, vous trouverez ici une mini-fiche de référence (très simplifiée) permettant une bonne lecture des diagrammes du sujet de TPL. Seules les notations des **diagrammes de classes** sont couvertes, pas celles des autres types de diagrammes UML.

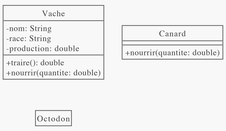
Classes

Les classes sont représentées sous leur forme la plus détaillée par un tableau en trois parties :

1. nom de la classe
2. liste des attributs (avec leur type)
3. liste des méthodes (avec leur signature).

Il est courant que tous les attributs et méthodes ne soient pas renseignés, mais seuls ceux importants dans un contexte donné. Le signe `+/-/#` est la *visibilité*, respectivement publique, privée ou protégée. Dans la forme simplifiée, on peut omettre la liste des attributs et/ou des méthodes (si les deux sont omis, il n'y a parfois qu'un rectangle avec le nom de la classe).

Syntaxe



Traduction

`Vache` est une classe, possédant (au moins) trois attributs privés `nom`, `race` et `production`, de types respectifs `String`, `String` et `double`, et deux méthodes publiques `traire`, sans paramètre et retournant un `double`, et `nourrir`, prenant en paramètre un `double` et ne retournant rien.

`Canard` est une classe dont les attributs ne sont pas détaillés, et possédant au moins une méthode `nourrir`, de visibilité non spécifiée, prenant en paramètre un `double` et ne retournant rien.

`Octodon` est une classe, dont le détail des attributs et méthodes n'est pas donné.

Associations

Une association est une relation structurelle forte entre deux classes. Comme pour les classes, différents niveaux de détails peuvent être affichés.

Associations simple

Syntaxe

Traduction

Il y a une relation entre les deux classes, mais on n'en sait pas plus. Par exemple, la classe `Vache` possède un *attribut* de type `Prairie` (ou tableau, ou une collection de `Prairie`), et vice-versa.

Syntaxe

Traduction

Il y a, dans la classe `Vache`, un attribut nommé `pâture` de type `Prairie`, dont la valeur peut être `null` (multiplicité 0..1). Réciproquement, il y a, dans la classe `Prairie`, un attribut nommé `occupants` de type tableau (ou collection) de `Vache`. Une prairie est associée à un nombre de vaches allant de 0 à n.

Association directionnelle

Syntaxe

Traduction

Il y a, dans la classe `Vache`, un attribut nommé `pâture` de type `Prairie`, dont la valeur peut être `null` (multiplicité 0..1).

Par contre, la classe `Prairie` n'a pas d'attribut de type `Vache`. Ainsi, « une vache sait dans quelle prairie elle se trouve », mais « une prairie ne sait pas quelle(s) vache(s) elle accueille »...

Agrégation et composition

Syntaxe

Traduction

Une `Prairie` connaît les vaches qu'elle accueille, elle possède un attribut nommé `occupants` de type tableau (ou collection) de `Vache`. Par contre, les vaches ne savent pas forcément dans quelle prairie elles se trouvent. Elles peuvent aussi ne pas être dans une prairie.

Dans une **agrégation** (losange **creux**), une instance de `Vache` peut exister même si elle n'est associée à aucune `Prairie`.

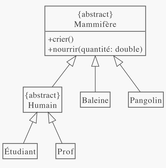
Syntaxe

Traduction

Comme pour une agrégation, une `Prairie` connaît les vaches qu'elle accueille. Par contre une **composition** (losange **plein**) signifie qu'une vache est toujours dans une et une seule prairie. La classe `Prairie` est « propriétaire » des instances de `Vache` qui la composent; si la prairie est détruite, ses vaches le sont également.

Héritage

Syntaxe



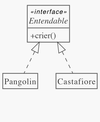
Traduction

Les classes `Baleine`, `Pangolin` et `Humain` sont des sous-classes de la classe `Mammifere`, et `Etudiant` et `Enseignant` sont des sous-classes de `Humain`.

La classe `Mammifere` est **abstraite**. Elle possède deux méthodes `crier`, qui est concrète, et `nourrir` qui est abstraite et devra être redéfinie dans les classes filles.

Interface

Syntaxe



Traduction

`Entendable` est une **interface** spécifiant une seule méthode `crier`. Les classes `Pangolin` et `Castafiore` **réalisent** cette interface, elles donc doivent (re)définir la méthode `crier`.

Commentaires

Syntaxe

Traduction

Les commentaires sont informatifs mais n'ont aucune signification formelle (comme n'importe quel commentaire dans un programme Java).