Qu'est-ce que UML ?

UML : **U**nified **M**odeling **L**anguage

C'est un langage de modélisation objet.

Il ne s'agit pas d'une méthode de développement de systèmes mais d'une notation graphique qui vient supporter une méthode.

Cette notation permet de représenter les résultats des différentes étapes du processus de développement d'un système.

Développement de logiciels

Le processus de développement de logiciels comporte 6 étapes :

- Spécification des besoins
- Analyse
- Conception
- Implémentation
- Test
- Maintenance

La conception est la phase où les spécifications techniques du logiciel à développer sont écrites. Après la conception, les programmeurs peuvent commencer à implémenter (coder) le système à partir des spécifications.

Diagramme de classes

Dans le processus de développement de systèmes, UML permet de décrire un système selon différents points de vue. À peu près tous les aspects d'un système peuvent être décrits à l'aide de 9 types de diagrammes dont le diagramme de classes qui permet de représenter le système en termes de classes et des relations statiques entre ces classes.

Donne une vue statique du système.

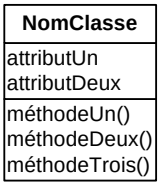
Une classe représente un élément important du système à développer :

- un objet physique
- une personne
- une machine
- un concept…

Classe

Une classe est caractérisée par

- un ensemble d'attributs qui représentent l'état des objets de cette classe
- un ensemble de méthodes qui représentent les opérations que cette classe peut faire (ou les services qu'elle offre)



Lorsqu'on s'intéresse à la classe de façon générale, on peut la représenter simplement par son nom dans un rectangle.

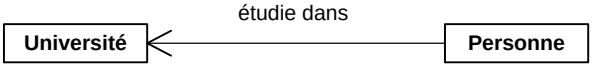
Association

L'association est le lien entre une ou plusieurs classes. Elle représente le fait qu'une instance d'une classe connaît une instance d'une autre classe et utilise ses services.

Nom de l'association

Lorsque la signification d'une association n'est pas évidente, on peut la nommer. Par convention, dans la notation UML, toute association doit être nommée par un **verbe**.

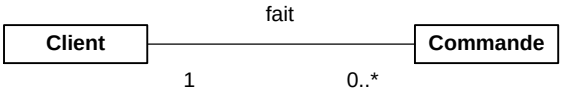
Toute association peut être **navigable**. Dans ce cas, on ajoute une flèche pour indiquer le sens de la lecture.



Multiplicité ou Cardinalité

Dans une association, la multiplicité indique le nombre d'instances de la classe associée qui peuvent être en relation à une instance d'une classe.

Exemple : Un client peut faire plusieurs commandes. Une commande est faite par un seul client.

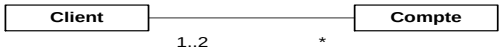


Multiplicités possibles:

1 (par défaut)
1..*
n..m
0..1
0..* ou simplement *

Exemples :

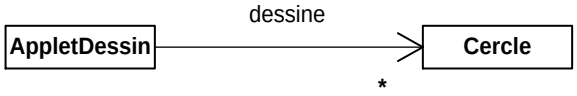
Un client peut avoir **plusieurs** comptes dans une banque. Un compte appartient à **un** client ou **deux** dans le cas d'un compte conjoint.



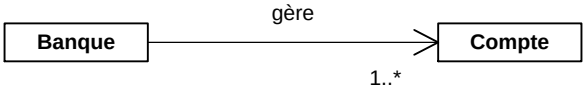
Une multiplicité de 0 indique qu'on objet peut exister même si l'objet associé n'existe pas. Autrement dit, la création d'un objet A ne requiert pas l'existence d'un objet B.

Dans l'exemple précédent, un client peut être créé sans qu'il ait un compte, mais un compte ne peut être créé pour un client inexistant.

Un applet dessine ou affiche **plusieurs** cercles.



Une banque gère **plusieurs** comptes.



Rôles

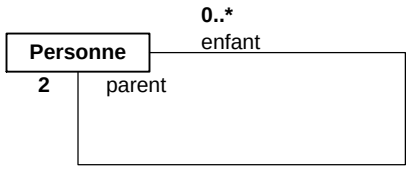
Il est utile de nommer les rôles des classes dans une association dans deux cas :

1. Association avec elle-même (réflexive)
2. Plus d'une association entre deux classes

Le rôle de chacune des classes qui sont reliées doit être placé aux extrémités de l'association.

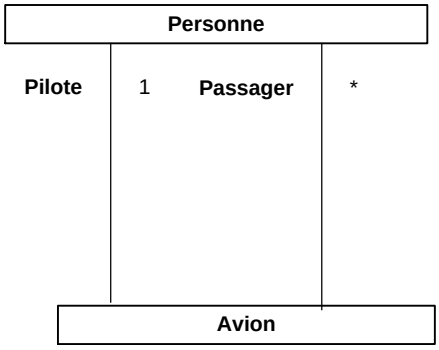
Exemple 1 :

Un parent qui est une personne peut avoir **plusieurs** enfants qui sont eux-aussi des personnes. Un enfant a **2** parents.



Exemple 2 :

Un avion est piloté par **un** pilote qui est une personne. L'avion transporte **plusieurs** passagers qui sont eux aussi des personnes.



Représentation détaillée d'une classe

Une classe est caractérisée par :

- un ensemble d'attributs qui représentent l'état des objets de cette classe
- un ensemble de méthodes qui représentent les opérations que cette classe peut faire (ou les services qu'elle offre).

Syntaxe :

nom de la classe
- attribut privé + attribut public # attribut protégé attribut de classe
- opération privée + opération publique # opération protégée opération de classe

Exemple :

Compte
-numéro : int -solde : double -nbreComptes : int = 0
+Compte() +getNuméro() : int +getSolde() : double +getNbreComptes() : int +setNuméro(entrée montant : double) +déposer(entrée montant : double) +retirer(entrée montant : double) +toString() : String

Déclaration des attributs

Les attributs d'une classe peuvent être visualisés ou pas, selon le niveau de détail souhaité.

Syntaxe : [Visibilité] NomAttribut [Multiplicité] : Type

ou [Visibilité] NomAttribut [Multiplicité] : Type [= InitVal]

Visibilité : type d'accessibilité

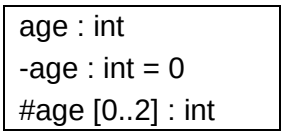
notation UML	En Java	
+	public	visible et modifiable par tout objet
-	private	seulement visible et modifiable par les opérations de l'objet auquel il appartient
#	protected	seulement accessible et modifiable par les opérations des classes descendantes

Multiplicité : intervalle ou nombre

InitVal : valeur initiale

- Dans la représentation de la classe en UML, les **attributs de classe** et les **méthodes de classe** sont identifiés par leur nom souligné.
- On rappelle qu'un attribut commun à toutes les instances d'une classe est appelé **attribut de classe** (en java, il est précédé du mot clé **static**).

Exemple :



Déclarations des opérations (méthodes)

Une opération représente un service spécifique offert par un objet

Syntaxe : [Visibilité] NomMéthode ([Liste param]) : Type

Visibilité : +, -, #

Liste param : nomParam : Type [= DefaultVal]