GÉNIE LOGICIEL

PARLEZ VOUS UML?

Victor Fernandes
Chef de projet informatique
Région Alsace
Victor.fernandes@region-alsace.eu

UML présentation

- I. Introduction
- I-A. Modélisation, modèle ?
- II. UML, OMG, OMA et c ie.
- III. Penser objet avec UML, pour concevoir objet.
- IV. Un langage universel et visuel.
- V. UML comme cadre d'une analyse objet.
- VI. UML : le chemin vers l'unification des processus.
- VII. Conclusion.

UML Modélisation

- 1. Qu'est-ce qu'un modèle ?
- 2. Comment modéliser avec UML ?
- 3. Les vues statiques d'UML
- 4. Exercices GAB

UML - Introduction

- UML (Unified Modeling Language)
  - fusion des trois méthodes OMT, Booch et OOSE.
  - De très nombreux acteurs industriels de renom ont adopté UML et participent à son développement.

UML - Introduction

- L'approche objet n'est pas une idée récente
  - 1967: Simula, premier langage de programmation à implémenter le concept de type abstrait à l'aide de classes
  - 1976: Smalltalk implémente les concepts fondateurs de l'approche objet : encapsulation, agrégation, héritage
  - Années 80 :
    - Les premiers compilateurs C++
    - nombreux langages orientés objets (Eiffel, Objective C, Loops...)

UML - Introduction

- L'approche objet est devenue une réalité
- Les concepts de base de l'approche objet sont stables et largement éprouvés
- L'approche objet est une solution technologique incontournable

UML - Introduction

- L'approche objet est moins intuitive que l'approche fonctionnelle
- Nécessité d'avoir une méthodologique approprié

UML - Introduction

- l'application des concepts objet nécessite une très grande rigueur
  - Le vocabulaire précis est un facteur d'échec important dans la mise en oeuvre d'une approche objet
  - Beaucoup de développeurs (même expérimentés) ne pensent souvent objet qu'à travers un langage de programmation

UML - Introduction

- Connaître C++ ou java n'est pas une fin en soi
- Il est très simple de décrire le résultat d'une analyse fonctionnelle, mais qu'en est-il d'une découpe objet ?

UML - Introduction

- Pour remédier aux inconvénients majeurs de l'approche objet il nous faut :
  - **un langage (pour s'exprimer clairement à l'aide des concepts objets)**, qui doit permettre de représenter des concepts abstraits (graphiquement par exemple), **limiter les ambiguïtés** (parler un langage commun, au vocabulaire précis, indépendant des langages orientés objet), **faciliter l'analyse** (simplifier la comparaison et l'évaluation de solutions).
  - **une démarche d'analyse et de conception objet**, pour ne pas effectuer une analyse fonctionnelle et se contenter d'une implémentation objet, **mais penser objet dès le départ**, définir les vues qui permettent de décrire tous les aspects d'un système avec des concepts objets

UML - Introduction

- La prise de conscience de l'importance d'une méthode spécifiquement objet ne date pas d'hier
- Plus de 50 méthodes objet sont apparues durant le milieu des années 90 (Booch, Classe-Relation, Fusion, HOOD, OMT, OOA, OOD, OOM, OOSE...).
- Ce n'est que récemment que les grands acteurs du monde informatique ont pris conscience de ce problème
- L'unification et la normalisation des méthodes objet dominantes (OMT, Booch et OOSE) ne datent que de 1995.
- UML est le fruit de cette fusion.
- **UML : une analyse objet passe par une modélisation objet.**

UML, OMG, OMA et c ie.

- Fin 1997, UML est devenu une norme OMG (Object Management Group).
- L'OMG est un organisme à but non lucratif, créé en 1989 à l'initiative de grandes sociétés (HP, Sun, Unisys, American Airlines, Philips...).

UML, OMG, OMA et c ie.

- L'OMG propose notamment l'architecture CORBA (Common Object Request Broker Architecture)
  - un modèle standard pour la construction d'applications à objets distribués (répartis sur un réseau).
- Au centre de l'architecture CORBA, un routeur de messages (ORB : Object Request Broker)
  - permet à des objets clients d'envoyer des requêtes et de recevoir des réponses, sans avoir à se préoccuper des détails techniques propres à l'infrastructure du réseau.

UML, OMG, OMA et c ie.

- L'approche Corba consiste à masquer les couches techniques du réseau et permet d'assurer l'interopérabilité de toute application à objets distribués.
- CORBA fait partie d'une vision globale de la construction d'applications réparties, appelée OMA (Object Management Architecture) et définie par l'OMG.
- Sans rentrer dans les détails, on peut résumer cette vision par la volonté de **favoriser l'essor industriel des technologies objet**, en offrant un ensemble de **solutions technologiques non propriétaires**, qui suppriment les clivages techniques.
- UML a été adopté (normalisé) par l'OMG et intégré à l'OMA, car il participe à cette vision et parce qu'il répond à la "philosophie" OMG.

III. Penser objet avec UML, pour concevoir objet

- Pour penser et concevoir objet il faut :
  - savoir "prendre de la hauteur",
  - jongler avec des concepts abstraits,
  - indépendants des langages d'implémentation et des contraintes purement techniques
- Pour conduire une analyse objet cohérente il faut :
  - Penser en terme d'association
  - de propriétés et de cardinalités...

IV. Un langage universel et visuel.

- UML est un support de communication performant facilitant la représentation et la compréhension de solutions objet
- Sa notation graphique permet :
  - d'**exprimer visuellement une solution objet,**
  - **limite les ambiguïtés** et les incompréhensions
- Son **indépendance** par rapport aux langages de programmation, aux domaines d'application et aux processus, en font un langage universel.

UML comme cadre d'une analyse objet

- UML permet de représenter un système selon différentes vues complémentaires :
  - Les diagrammes
- Un diagramme UML est une représentation graphique, qui s'intéresse à un aspect précis du modèle
  - Chaque type de diagramme UML possède une structure (les types des éléments de modélisation qui le composent sont prédéfinis) et véhicule une sémantique précise (il offre toujours la même vue d'un système).

UML comme cadre d'une analyse objet

- Une caractéristique importante des diagrammes UML, est qu'ils supportent l'abstraction.

- UML opte en effet pour **l'élaboration des modèles,** plutôt que pour une approche qui impose une barrière stricte entre analyse et conception. Les modèles d'analyse et de conception ne diffèrent que par leur niveau de détail, il n'y a pas de différence dans les concepts utilisés.

- UML n'introduit pas d'éléments de modélisation propres à une activité (analyse, conception...) ; **le langage reste le même à tous les niveaux d'abstraction.**

- **Cette approche simplificatrice facilite le passage entre les niveaux d'abstraction. L'élaboration encourage une approche non linéaire,** les "retours en arrière" entre niveaux d'abstraction différents sont facilités et la traçabilité entre modèles de niveaux différents est assurée par l'unicité du langage.

UML comme cadre d'une analyse objet

- UML favorise donc le prototypage, et c'est là une de ses forces.

- UML permet donc non seulement de représenter et de manipuler les concepts objet, il sous-entend une démarche d'analyse qui permet de concevoir une solution objet de manière itérative, grâce aux diagrammes, qui supportent l'abstraction

VI. UML : le chemin vers l'unification des processus.

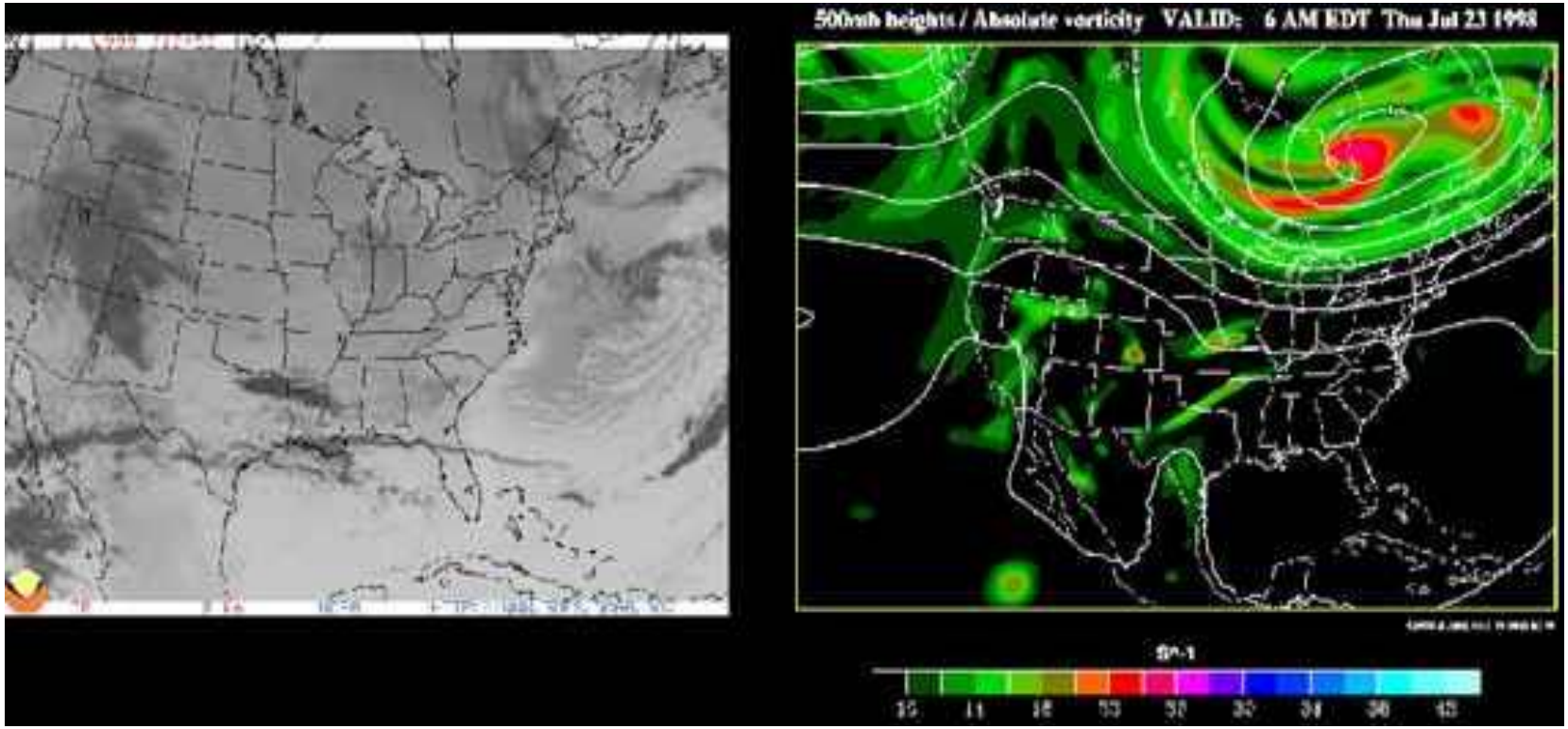
- D'après les auteurs d'UML, un processus de développement qui possède ces qualités fondamentales "devrait" favoriser la réussite d'un projet.
  - guidée par les besoins des utilisateurs du système,
  - centrée sur l'architecture logicielle,
  - itérative et incrémentale.
- Le RUP ("Rational Unified Process"),
  - processus de développement "clé en main", proposé par Rational Software, est lui aussi modélisé (documenté) avec UML. Il offre un cadre méthodologique générique qui repose sur UML et la suite d'outils Rational.
- VALtech propose le 2TUP ("**2 Tracks Unified Process**",
  - cf. "UML en action", ed. Eyrolles), un processus unifié (c'est-à-dire construit sur UML, itératif, centré sur l'architecture et conduit par les cas d'utilisation), qui apporte une réponse aux contraintes de changement continuel imposées aux systèmes d'informations des entreprises.
- Bien qu'un processus universel soit une utopie, la capitalisation des meilleures pratiques, à travers une famille de processus unifiés (tels que le RUP et le 2TUP), devient donc une réalité.

VII. Conclusion

- UML n'impose pas de méthode de travail particulière,
  - il peut être intégré à n'importe quel processus de développement I
- Intégrer UML par étapes dans un processus, de manière pragmatique, est tout à fait possible.

- Intégrer UML dans un processus ne signifie donc pas révolutionner ses méthodes de travail, mais cela devrait être l'occasion de se remettre en question, en s'inspirant des meilleures pratiques, capitalisées à travers les processus unifiés (RUP et 2TUP).

Modéliser avec UML



Modéliser avec UML - Qu'est-ce qu'un modèle ?

- **Un modèle est une abstraction de la réalité**

  L'abstraction est un des piliers de l'approche objet.

  - Il s'agit d'un processus qui consiste à identifier les caractéristiques intéressantes d'une entité, en vue d'une utilisation précise.
- L'abstraction désigne aussi le résultat de ce processus, c'est-à-dire l'ensemble des caractéristiques essentielles d'une entité, retenues par un observateur

Modéliser avec UML - Qu'est-ce qu'un modèle ?

- **Un modèle est une vue subjective mais pertinente de la réalité**
  - Un modèle définit une frontière entre la réalité et la perspective de l'observateur. Ce n'est pas "la réalité", mais une vue très subjective de la réalité.
- Bien qu'un modèle ne représente pas une réalité absolue, un modèle reflète des aspects importants de la réalité, il en donne donc une vue juste et pertinente.

Modéliser avec UML - Qu'est-ce qu'un modèle ?

- **Quelques exemples de modèles**

  - Modèle météorologique :
    à partir de données d'observation (satellite ...), permet de prévoir les conditions climatiques pour les jours à venir.
  - Modèle économique :
    peut par exemple permettre de simuler l'évolution de cours boursiers en fonction d'hypothèses macro-économiques (évolution du chômage, taux de croissance...).

- Modèle démographique :
  définit la composition d'un panel d'une population et son comportement, dans le but de fiabiliser des études statistiques, d'augmenter l'impact de démarches commerciales, etc...

Modéliser avec UML - Qu'est-ce qu'un modèle ?

- **Caractéristiques fondamentales des modèles**

  Le caractère abstrait d'un modèle doit notamment permettre :

  - de faciliter la compréhension du système étudié
    > Un modèle réduit la complexité du système étudié.
  - de simuler le système étudié
    > Un modèle représente le système étudié et reproduit ses comportements.

- Un modèle réduit (décompose) la réalité, dans le but de disposer d'éléments de travail exploitables par des moyens mathématiques ou informatiques :

  modèle / réalité $\simeq$ digital / analogique

Modéliser avec UML - Qu'est-ce qu'un modèle ?

- **Comment modéliser avec UML ?**
- **UML est un langage qui permet de représenter des modèles, mais il ne définit pas le processus d'élaboration des modèles !**

- Cependant, dans le cadre de la modélisation d'une application informatique, les auteurs d'UML préconisent d'utiliser une démarche :
  - itérative et incrémentale,
  - guidée par les besoins des utilisateurs du système,
  - centrée sur l'architecture logicielle.
- D'après les auteurs d'UML, un processus de développement qui possède ces qualités devrait favoriser la réussite d'un projet.

Modéliser avec UML - Qu'est-ce qu'un modèle ?

- **Une démarche itérative et incrémentale ?**
- L'idée est simple : pour modéliser (comprendre et représenter) un système complexe, il vaut mieux s'y prendre en plusieurs fois, en affinant son analyse par étapes.
- Cette démarche devrait aussi s'appliquer au cycle de développement dans son ensemble, en favorisant le prototypage.
- Le but est de mieux maîtriser la part d'inconnu et d'incertitudes qui caractérisent les systèmes complexes.

Modéliser avec UML - Qu'est-ce qu'un modèle ?

- **Une démarche pilotée par les besoins des utilisateurs ?**
- **Avec UML, ce sont les utilisateurs qui guident la définition des modèles :**
  - Le périmètre du système à modéliser est défini par les besoins des utilisateurs (les utilisateurs définissent ce que doit être le système).
  - Le but du système à modéliser est de répondre aux besoins de ses utilisateurs (les utilisateurs sont les clients du système).
- **Les besoins des utilisateurs servent aussi de fil rouge, tout au long du cycle de développement (itératif et incrémental) :**
  - A chaque itération de la phase d'analyse, on clarifie, affine et valide les besoins des utilisateurs.
  - A chaque itération de la phase de conception et de réalisation, on veille à la prise en compte des besoins des utilisateurs.
  - A chaque itération de la phase de test, on vérifie que les besoins des utilisateurs sont satisfaits.

Modéliser avec UML - Qu'est-ce qu'un modèle ?

- **Une démarche centrée sur l'architecture ?**

- **Une architecture adaptée est la clé de voûte du succès d'un développement.**

  Elle décrit des choix stratégiques qui déterminent en grande partie les qualités du logiciel (adaptabilité, performances, fiabilité...).

Les vues statiques d'UML - Uses Cases

La conceptualisation :

- Le but de la conceptualisation est de comprendre et structurer les besoins du client.
- Il ne faut pas chercher l'exhaustivité, mais clarifier, filtrer et organiser les besoins !
- Une fois identifiés et structurés, ces besoins :
  - définissent le contour du système à modéliser (ils précisent le but à atteindre),
  - permettent d'identifier les fonctionnalités principales (critiques) du système.
- Le modèle conceptuel doit permettre une meilleure compréhension du système.
- Le modèle conceptuel doit servir d'interface entre tous les acteurs du projet.
- Les besoins des clients sont des éléments de traçabilité dans un processus intégrant UML. Le modèle conceptuel joue un rôle central, il est capital de bien le définir !

Les vues statiques d'UML - Uses Cases

Cas d'utilisation (use cases)

- Il s'agit de la solution UML pour représenter le modèle conceptuel.
- Les use cases permettent de structurer les besoins des utilisateurs et les objectifs correspondants d'un système.
- Ils centrent l'expression des exigences du système sur ses utilisateurs : ils partent du principe que les objectifs du système sont tous motivés.
- Ils se limitent aux préoccupations "réelles" des utilisateurs ; ils ne présentent pas de solutions d'implémentation et ne forment pas un inventaire fonctionnel du système.
- Ils identifient les utilisateurs du système (acteurs) et leur interaction avec le système.
- Ils permettent de classer les acteurs et structurer les objectifs du système.
- Ils servent de base à la traçabilité des exigences d'un système dans un processus de développement intégrant UML.

Il était une fois….

Il était une fois...

Le modèle conceptuel est le type de diagramme UML qui possède la notation la plus simple ; mais paradoxalement c'est aussi celui qui est le plus mal compris !

Au début des années 90, Ivar Jacobson (inventeur de OOSE, une des méthodes fondatrices d'UML) a été nommé chef d'un énorme projet informatique chez Ericsson. Le hic, c'est que ce projet était rapidement devenu ingérable, les ingénieurs d'Ericsson avaient accouché d'un monstre. Personne ne savait vraiment quelles étaient les fonctionnalités du produit, ni comment elles étaient assurées, ni comment les faire évoluer...

Il était une fois….

Classique lorsque les commerciaux promettent monts et merveilles à tous les clients qu'ils démarchent, sans se soucier des contraintes techniques, que les clients ne savent pas exprimer leurs besoins et que les ingénieurs n'ont pas les ressources pour développer le mouton à cinq pattes qui résulte de ce chaos.

Pour éviter de foncer droit dans un mur et mener à bien ce projet critique pour Ericsson, Jacobson a eu une idée. Plutôt que de continuer à construire une tour de Babel, pourquoi ne pas remettre à plat les objectifs réels du projet ? En d'autres termes : quels sont les besoins réels des clients, ceux qui conditionneront la réussite du projet ? Ces besoins critiques, une fois identifiés et structurés, permettront enfin de cerner "ce qui est important pour la réussite du projet".

Il était une fois….

Le bénéfice de cette démarche simplificatrice est double. D'une part, tous les acteurs du projet ont une meilleure compréhension du système à développer, d'autre part, les besoins des utilisateurs, une fois clarifiés, serviront de fil rouge, tout au long du cycle de développement.

A chaque itération de la phase d'analyse, on clarifie, affine et valide les besoins des utilisateurs ; à chaque itération de la phase de conception et de réalisation, on veille à la prise en compte des besoins des utilisateurs et à chaque itération de la phase de test, on vérifie que les besoins des utilisateurs sont satisfaits.

Il était une fois….

Simple mais génial. Pour la petite histoire, sachez que grâce à cette démarche initiée par Jacobson, Ericsson a réussi à mener à bien son projet et a gagné une notoriété internationale dans le marché de la commutation.

Morale de cette histoire :

La détermination et la compréhension des besoins sont souvent difficiles car les intervenants sont noyés sous de trop grandes quantités d'informations. Or, comment mener à bien un projet si l'on ne sait pas où l'on va ?

Il était une fois….

Conclusion : il faut clarifier et organiser les besoins des clients (les modéliser).

Jacobson identifie les caractéristiques suivantes pour les modèles :

Un modèle est une simplification de la réalité.

Il permet de mieux comprendre le système qu'on doit développer.

Les meilleurs modèles sont proches de la réalité.

Les use cases, permettent de modéliser les besoins des clients d'un système et doivent aussi posséder ces caractéristiques.

Ils ne doivent pas chercher l'exhaustivité, mais clarifier, filtrer et organiser les besoins !

Il était une fois….

Une fois identifiés et structurés, ces besoins :

définissent le contour du système à modéliser (ils précisent le but à atteindre),

permettent d'identifier les fonctionnalités principales (critiques) du système.

Les use cases ne doivent donc en aucun cas décrire des solutions d'implémentation. Leur but est justement d'éviter de tomber dans la dérive d'une approche fonctionnelle, où l'on liste une litanie de fonctions que le système doit réaliser.

Il était une fois….

Bien entendu, rien n'interdit de gérer à l'aide d'outils (Doors, Requisite Pro, etc...) les exigences systèmes à un niveau plus fin et d'en assurer la traçabilité, bien au contraire.

Mais un modèle conceptuel qui identifie les besoins avec un plus grand niveau d'abstraction reste indispensable. Avec des systèmes complexes, filtrer l'information, la simplifier et mieux l'organiser, c'est rendre l'information exploitable. Produisez de l'information éphémère, complexe et confuse, vous obtiendrez un joyeux "désordre" (pour rester poli).

Il était une fois….

Dernière remarque :

Utilisez les use cases tels qu'ils ont été pensé par leurs créateurs !

UML est issu du terrain.

Si vous utilisez les use cases sans avoir en tête la démarche sous-jacente, vous n'en tirerez aucun bénéfice.

Eléments de base des cas d'utlisation

Acteur : entité externe qui agit sur le système (opérateur, autre système...).

L'acteur peut consulter ou modifier l'état du système.

En réponse à l'action d'un acteur, le système fournit un service qui correspond à son besoin.

Les acteurs peuvent être classés (hiérarchisés).

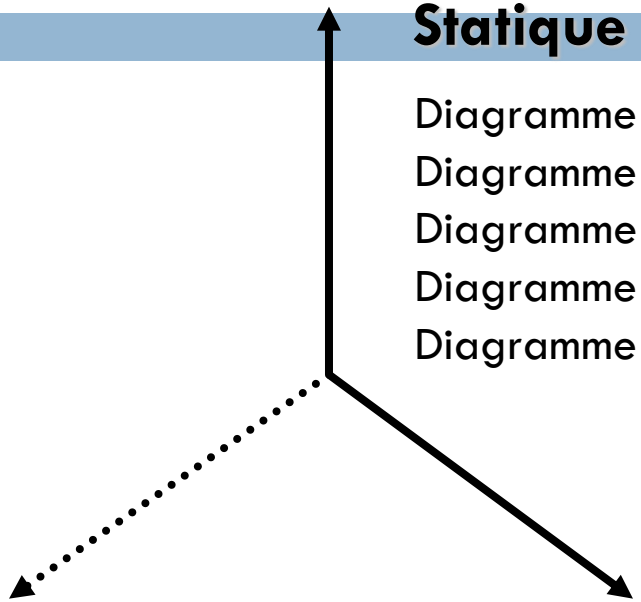
Use case : ensemble d'actions réalisées par le système, en réponse à une action d'un acteur.

Les uses cases peuvent être structurés.

Les uses cases peuvent être organisés en paquetages (packages).

L'ensemble des use cases décrit les objectifs (le but) du système.

Axe de Modélisation



Statique

Diagramme de Classes
Diagramme d'Objets
Diagramme de Composants
Diagramme de Déploiement
Diagramme de Use Case

Fonctionnel

Diagramme de Use Case

Dynamique

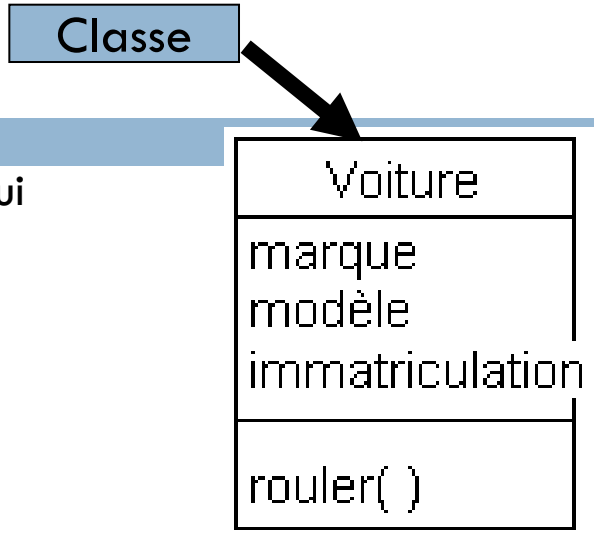
Diagramme d'Etats-Transitions
Diagramme d'Activité
Diagramme de Séquence

L ’AXE STATIQUE

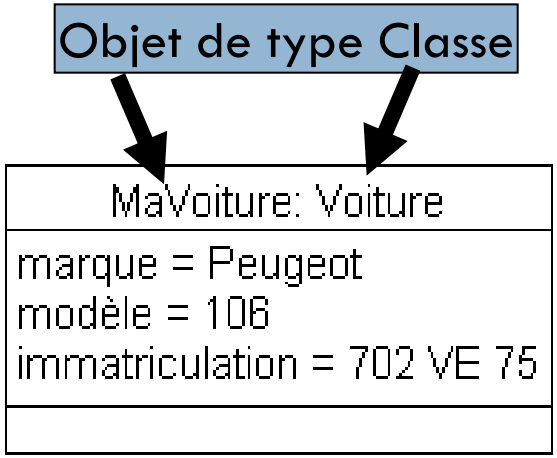
NOTATION UML

Notation de base

- **Classe**
  - Une description d'un ensemble d'objets qui partage les mêmes attributs, opérations, méthodes, relations et contraintes



- **Objet**
  - Une entité avec une limite et une identité bien définies qui encapsule de l'état et du comportement. L'état est représenté par des attributs et des relations, le comportement est représenté par des opérations et des méthodes. Un objet est une instance d'une classe.



NOTATION UML

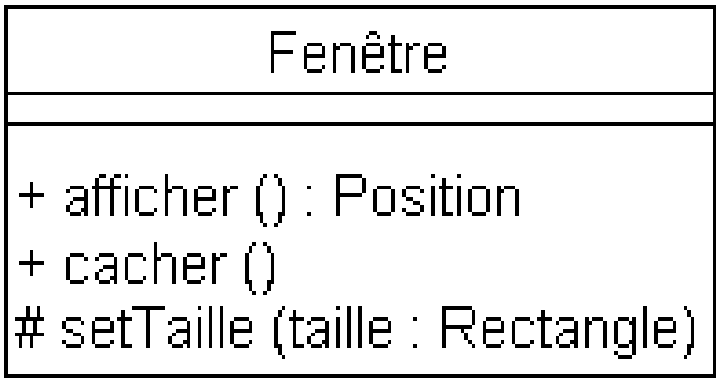
Attribut

- **Attribut** = **propriété** nommée d 'une classe
- **Syntaxe**
  - *visibilité* ***nom*** *: type = valeur initiale*
- **Visibilité**
  - + public
  - # protégé
  - - privé
  - package
- **Attribut de classe**
  - la **portée standard** d'un attribut est limité à un **objet**
  - quand cette **portée** s'applique à la **classe** elle même, on parle d'**attribut de classe** (représenté par le symbole **$** ou **souligné**)
- **Attribut dérivé**
  - attribut qui peut être **déduit** d'un ou plusieurs **autres attributs** (représenté par le symbole /)

Fenêtre
+ taille : Rectangle = (100,100) - visible : boolean = TRUE - xptr : XWindow #$ tailleMax : Rectangle #$ tailleMin : Rectangle /#$ tailleMoyenne : Rectangle

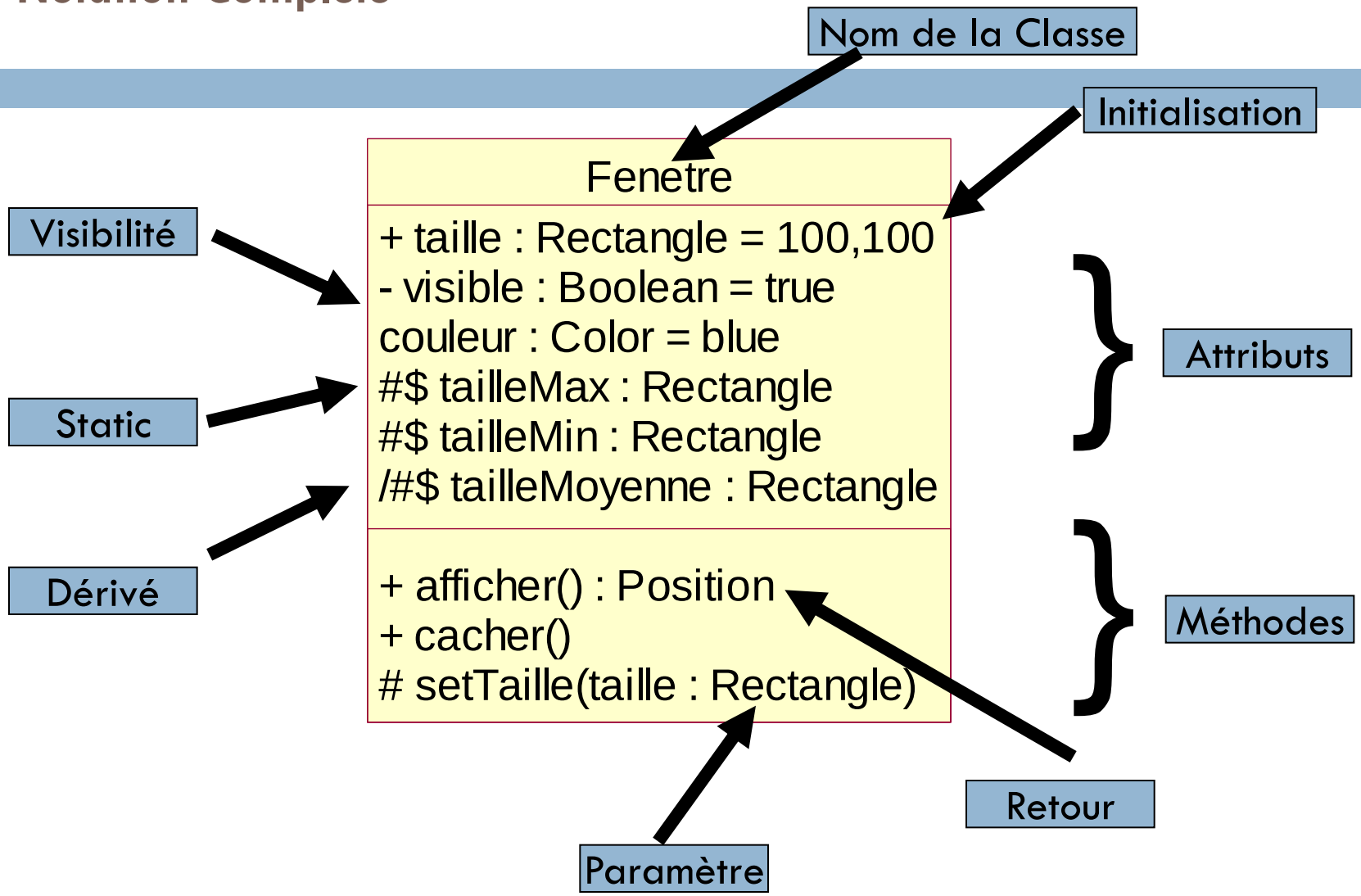
Méthode

- **Méthode** = **service** que l 'on peut demander à un objet pour réaliser un comportement
- **Syntaxe**
  - *visibilité* ***nom*** *(paramètres) : type retour*
- Mêmes notions que l'attribut
  - **visibilité**
  - **méthode de classe**



NOTATION UML

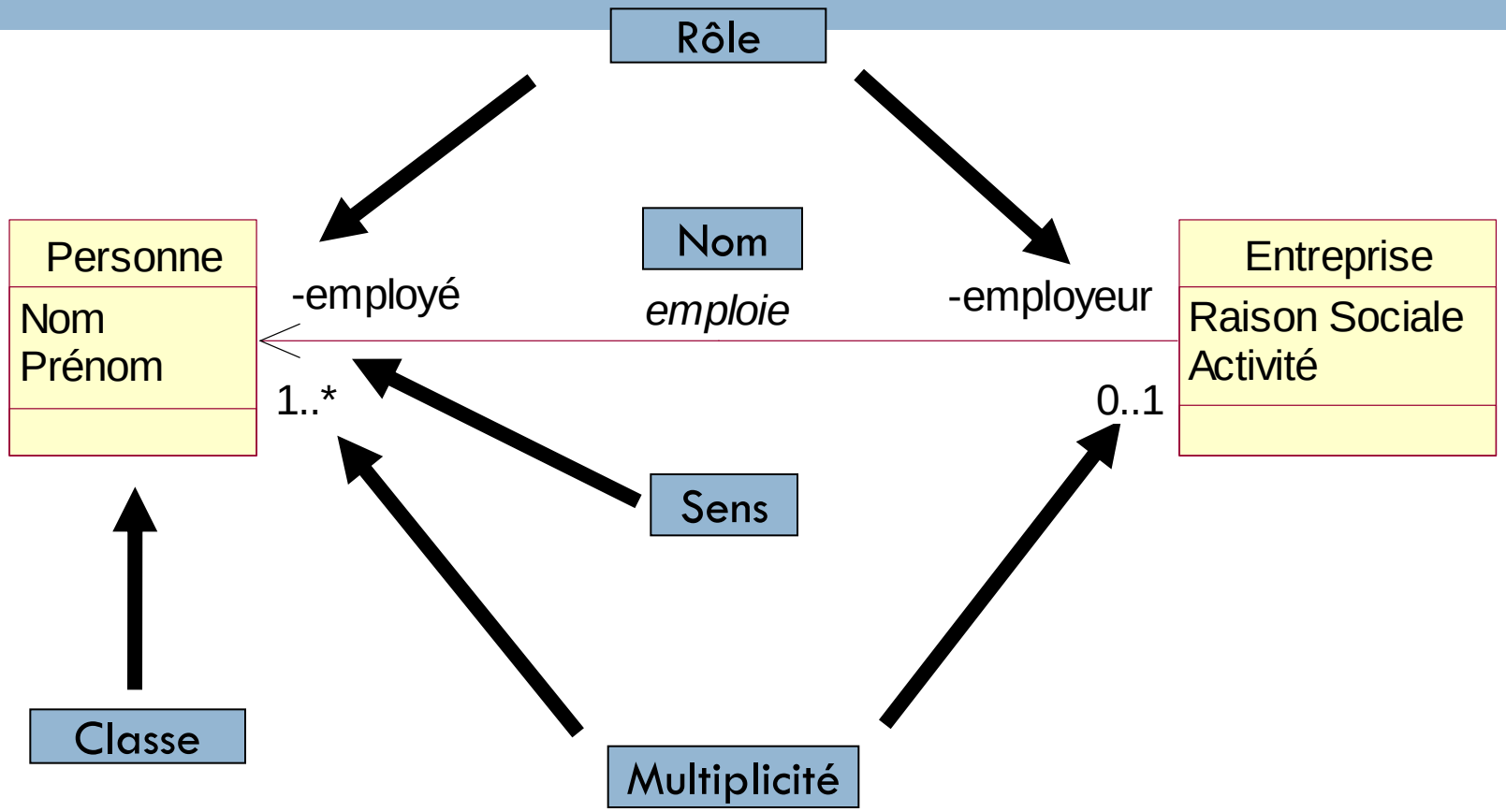
Notation Complète



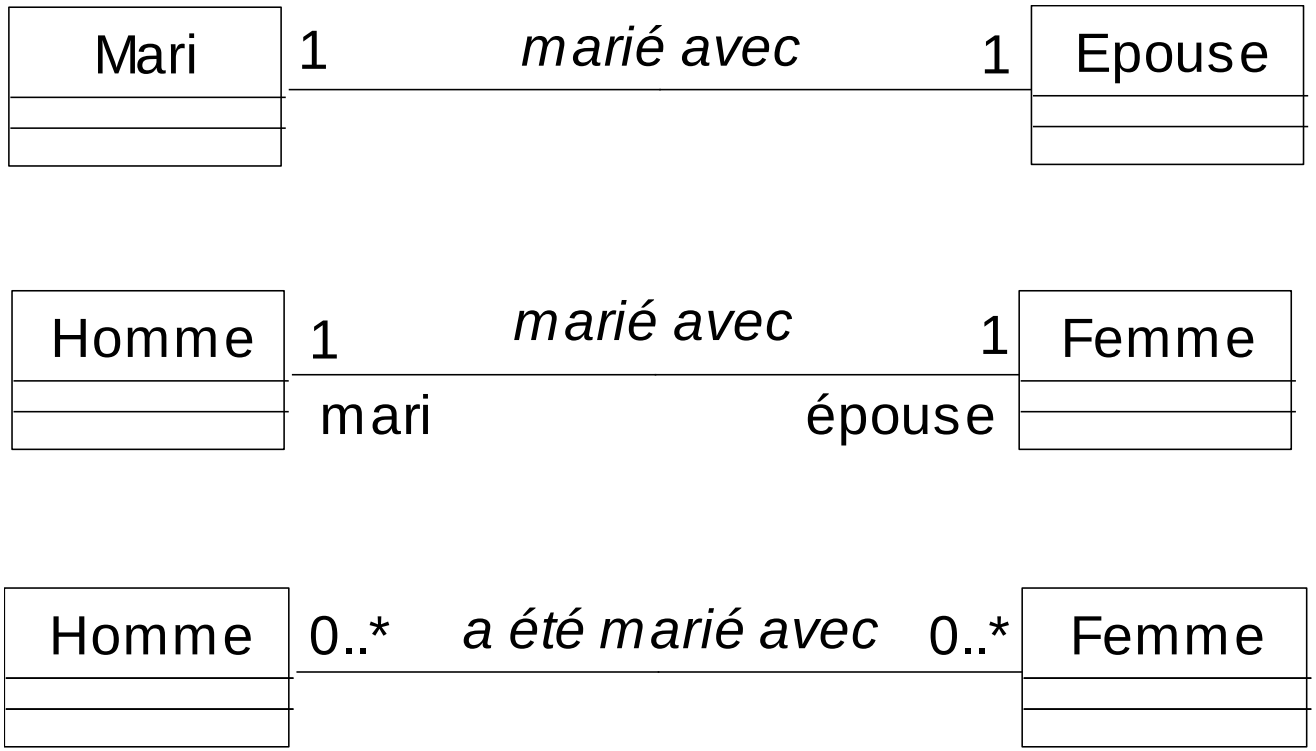
Définition

- **Association**
  - Exprime une **connexion** sémantique bi-directionnelle entre classes
  - Abstraction des liens qui existent entre objets
  - Le **sens** d 'une association peut-être précisé par une **flêche**
- **Association binaire** = Association entre 2 classes. Cas particulier d 'association **n-aire**
- **Rôle** = rôle joué par une classe dans une association
- **Multiplicité** = indique le **nombre** d'instances d 'une classe qui peut être mise en relation avec une seul instance de la classe associée
  - 1 : obligatoire
  - 0..1 : optionnel
  - 0..* ou * : quelconque
  - 1..* : au moins 1
  - 1..5, 10 : entre 1 et 5, ou 10

Exemple

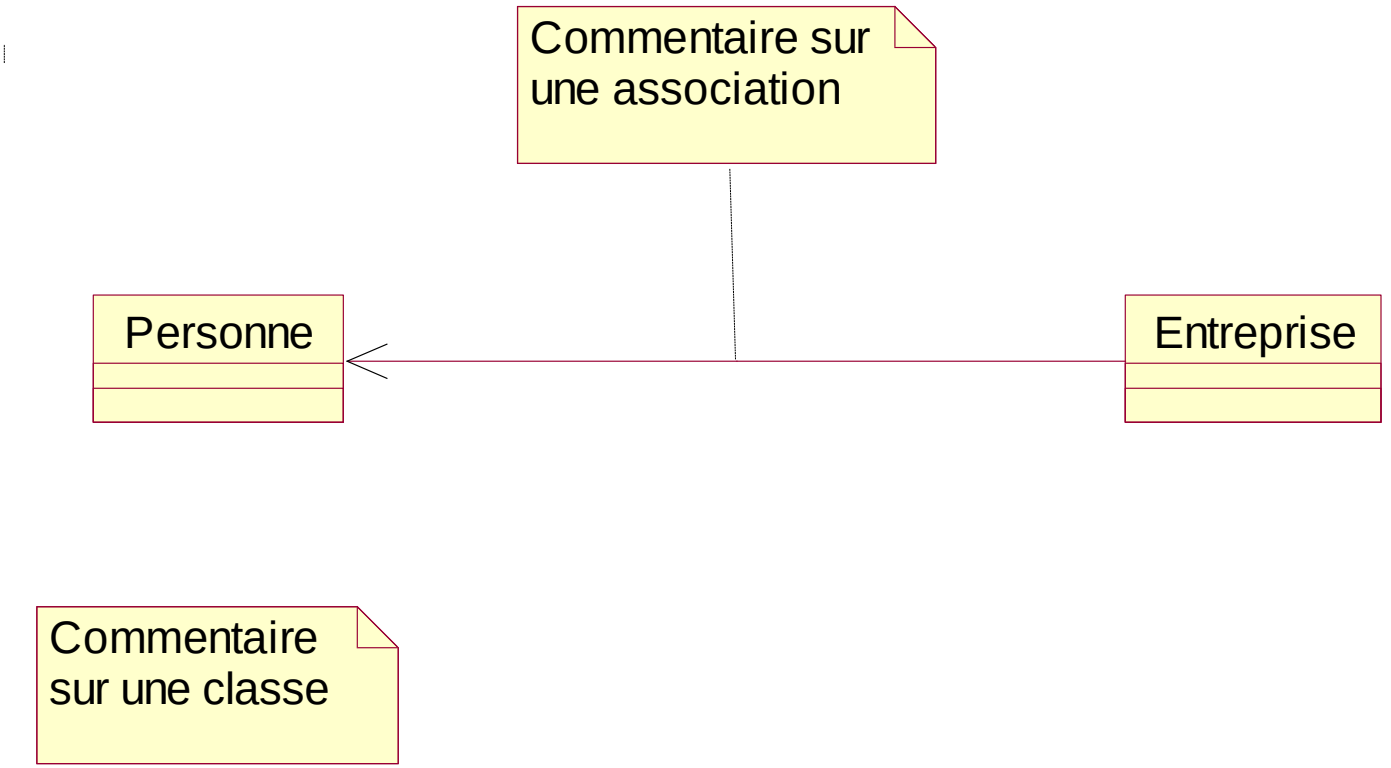

Rôle
Personne
Nom
Prénom
-employé
Nom
emploie
-employeur
Entreprise
Raison Sociale
Activité
1..*
0..1
Sens
Classe
Multiplicité

Sémantique


Mari
1
marié avec
1
Epouse
Homme
1
marié avec
1
Femme
mari
épouse
Homme
0..*
a été marié avec
0..*
Femme

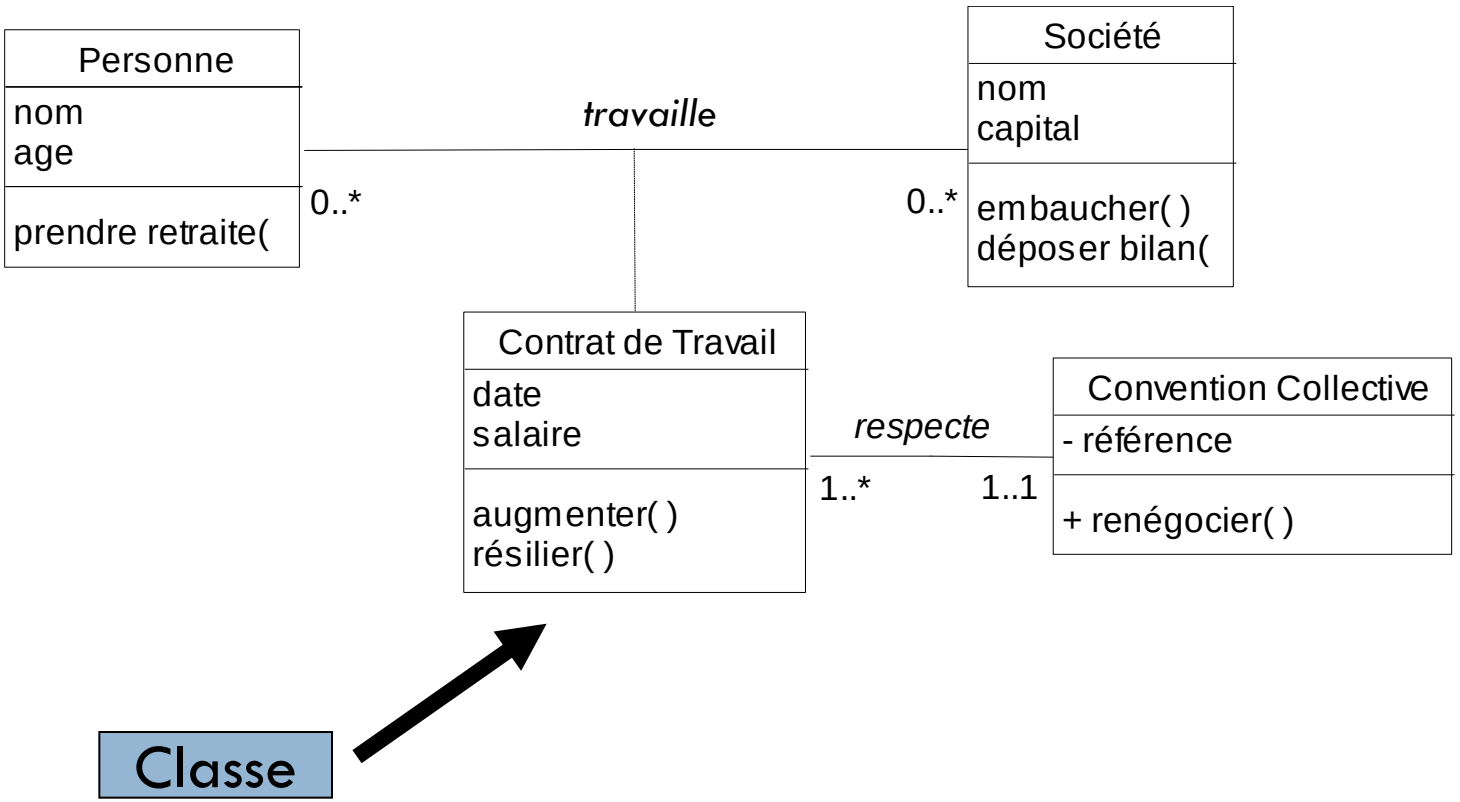
Note

- **Note** = Commentaire placé sur un diagramme



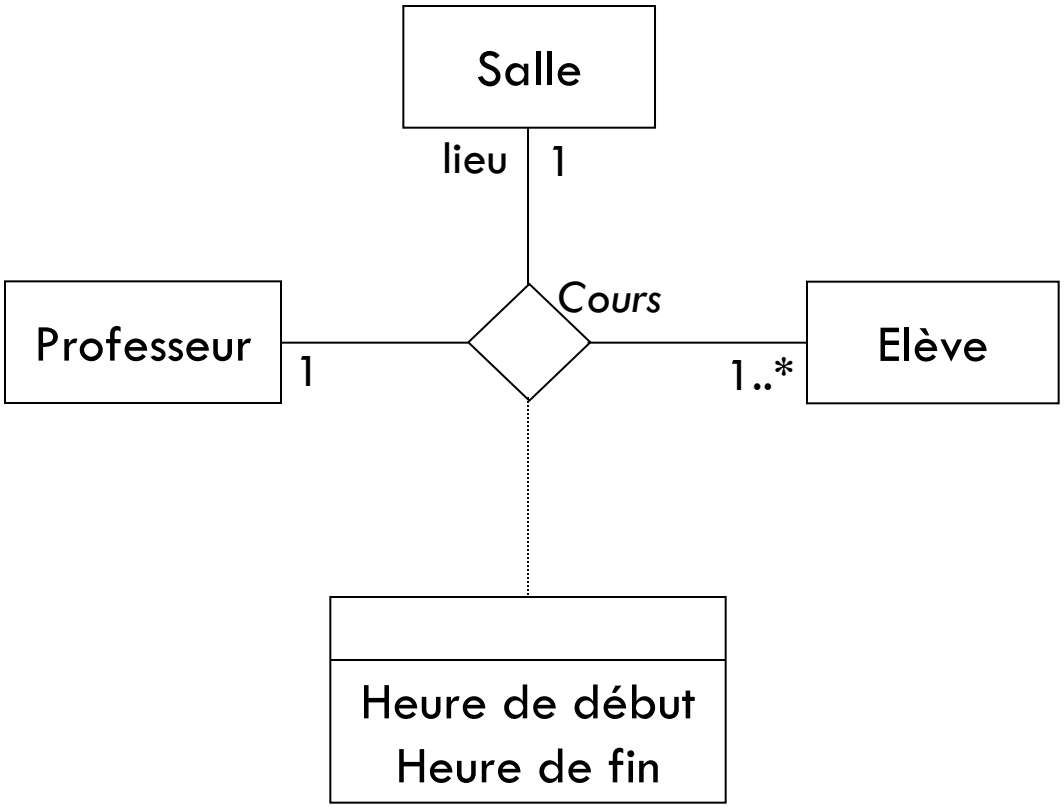
Classe d'Association

- **Classe d'association** = Elément ayant à la fois les propriétés d 'une classe et d 'une association



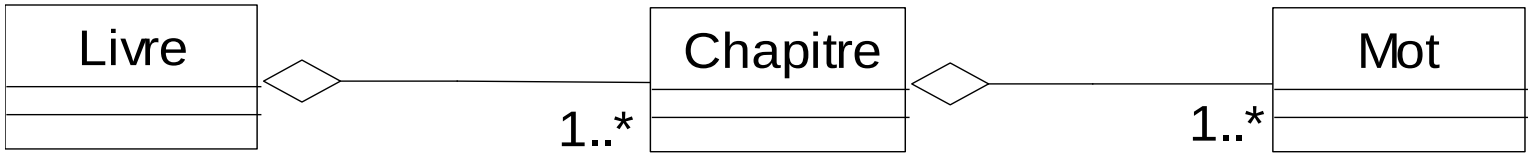
Association n-aire

- **Association n-aire** = Une association parmi 3 classes ou plus. Chaque instance de l'association est un n-tuple de valeurs des classes respectives.



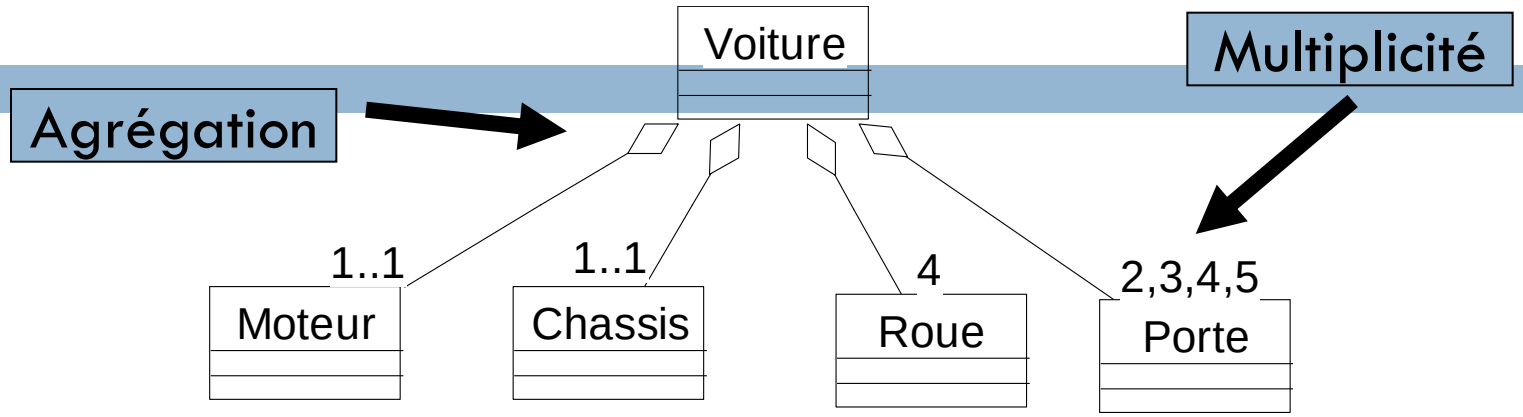
Définitions

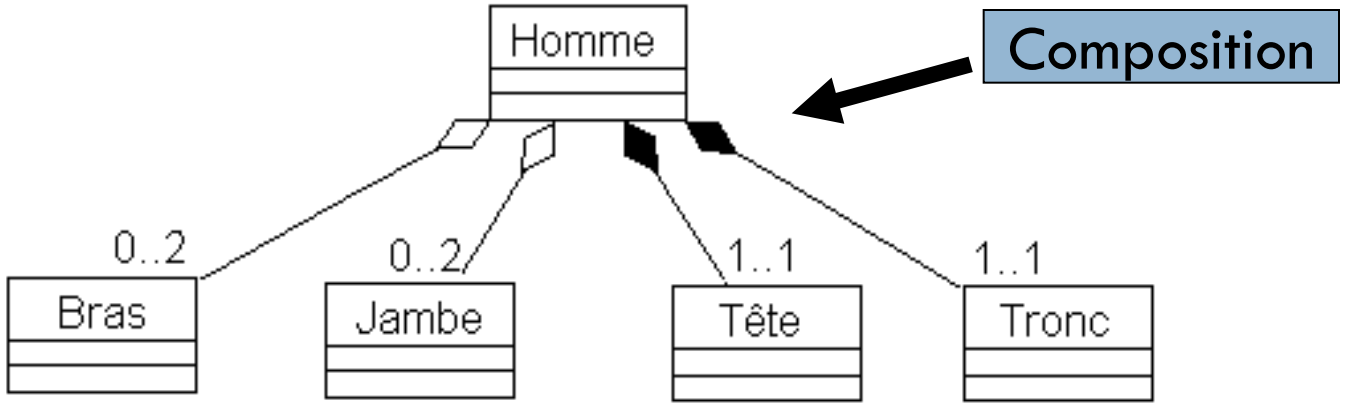
- **Agrégation** = **association** particulière spécifiant une relation 'tout - partie' entre l'agrégat et un composant
  - **Inclusion**
  - **Propagation**



- **Composition** = forme **forte d'agrégation** avec un **cycle de vie** des parties lié à celui du composite

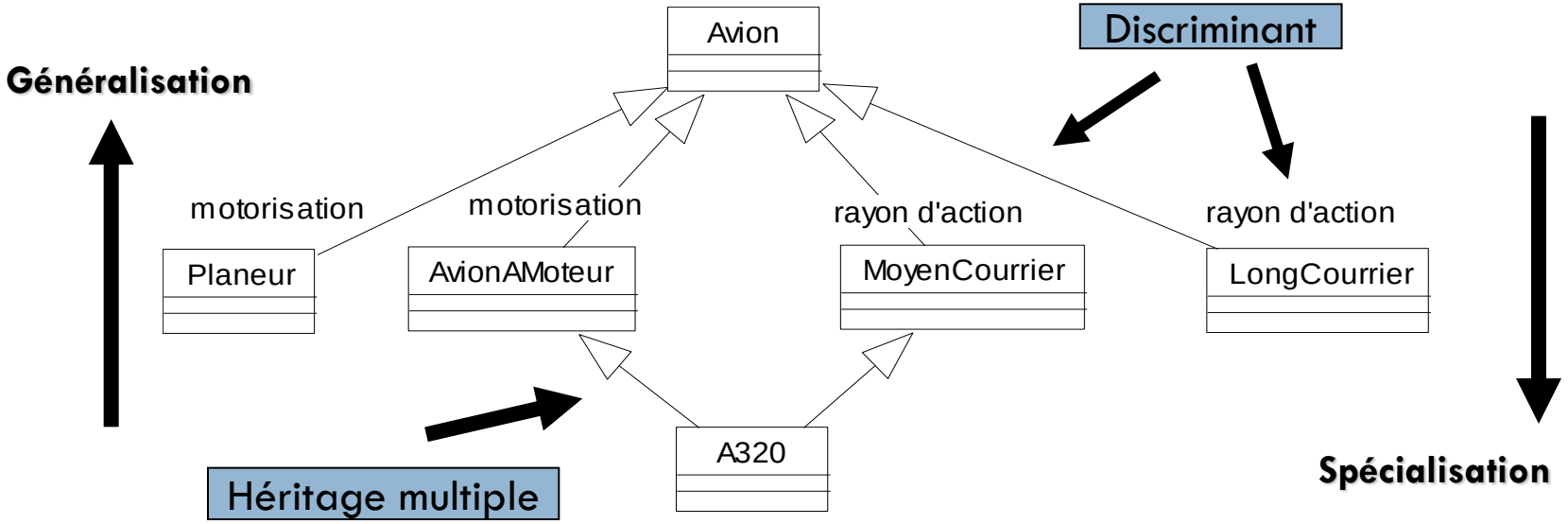
Exemples


Voiture
Agrégation
Multiplicité
1..1
Moteur
1..1
Chassis
4
Roue
2,3,4,5
Porte


Homme
Composition
0..2
Bras
0..2
Jambe
1..1
Tête
1..1
Tronc

Définitions

- **Généralisation** = relation ente un élément plus général et un élément plus spécifique qui est entièrement conforme avec le premier élément, et qui ajoute de l 'information supplémentaire
- **Spécialisation** = mécanisme par lequel des éléments plus spécifiques incorporent la structure et le comportement d'éléments plus généraux (notion **d'héritage**).



Exercices : modéliser

Un pays possède une capitale

Exercices : modéliser

Une personne dîne avec une fourchette

Exercices : modéliser

Un chemin peut représenter un fichier ou un répertoire

Exercices : modéliser

Un chemin est un répertoire avec éventuellement un nom de fichier

Exercices : modéliser

Un fichier contient des enregistrements

Exercices : modéliser

Un fichier est accessible par un utilisateur selon des droits d'accès

Exercices : modéliser

Un dessin est soit du texte, soit une forme géométrique, soit un groupe de dessins.

Exercices : modéliser

Des personnes utilisent un langage pour un projet

Exercices : modéliser

Une personne joue dans une équipe pour une certaine durée

Exercices : modéliser

Une équipe est composée de plusieurs personnes.

Exercices : modéliser

Une route connecte deux villes

Exo : AU COEUR D'UN ORDINATEUR

Un ordinateur est composé d'un ou plusieurs moniteurs, d'un boîtier, d'une souris optionnelle et d'un clavier. Un boîtier a un châssis métallique, une carte mère, plusieurs barrettes de mémoire (RAM, ROM et cache), un ventilateur optionnel, des supports de stockage (disquette, disque-dur, CD-ROM, DVD-ROM...), et des cartes périphériques (son, réseau, graphique...). Un ordinateur possède toujours au moins un lecteur de disquette ou un disque-dur.

Exo : L'université

L'université comporte des personnels administratifs et techniques, des enseignants, des étudiants et des chercheurs (qui sont tous des personnes). Certains étudiants peuvent être des chercheurs (les doctorants) ou des enseignants (les assistants enseignants). Certaines personnes (étudiants ou non) peuvent être à la fois chercheurs et enseignants.

Exo : L'éditeur graphique

Un éditeur de documents graphiques supporte le groupement d'objets graphiques. Un document se compose de plusieurs feuilles, chacune contenant des objets graphiques (texte, forme géométrique et groupe d'objets). Un groupe est un ensemble d'objets pouvant contenir d'autres groupes. Un groupe doit contenir au moins deux éléments. Les formes géométriques comprennent les cercles, les ellipses, les rectangles, les carrés, les lignes...

Exo : Achat de voiture

Une personne physique peut avoir jusqu'à trois sociétés (personnes morales) qui l'emploient. Chaque personne physique possède un numéro de sécurité sociale qui l'identifie. Une voiture a un numéro d'immatriculation. Une voiture est la propriété d'une personne (physique ou morale). Un emprunt dans une banque peut être demandé pour l'achat d'une voiture.