La notation UML

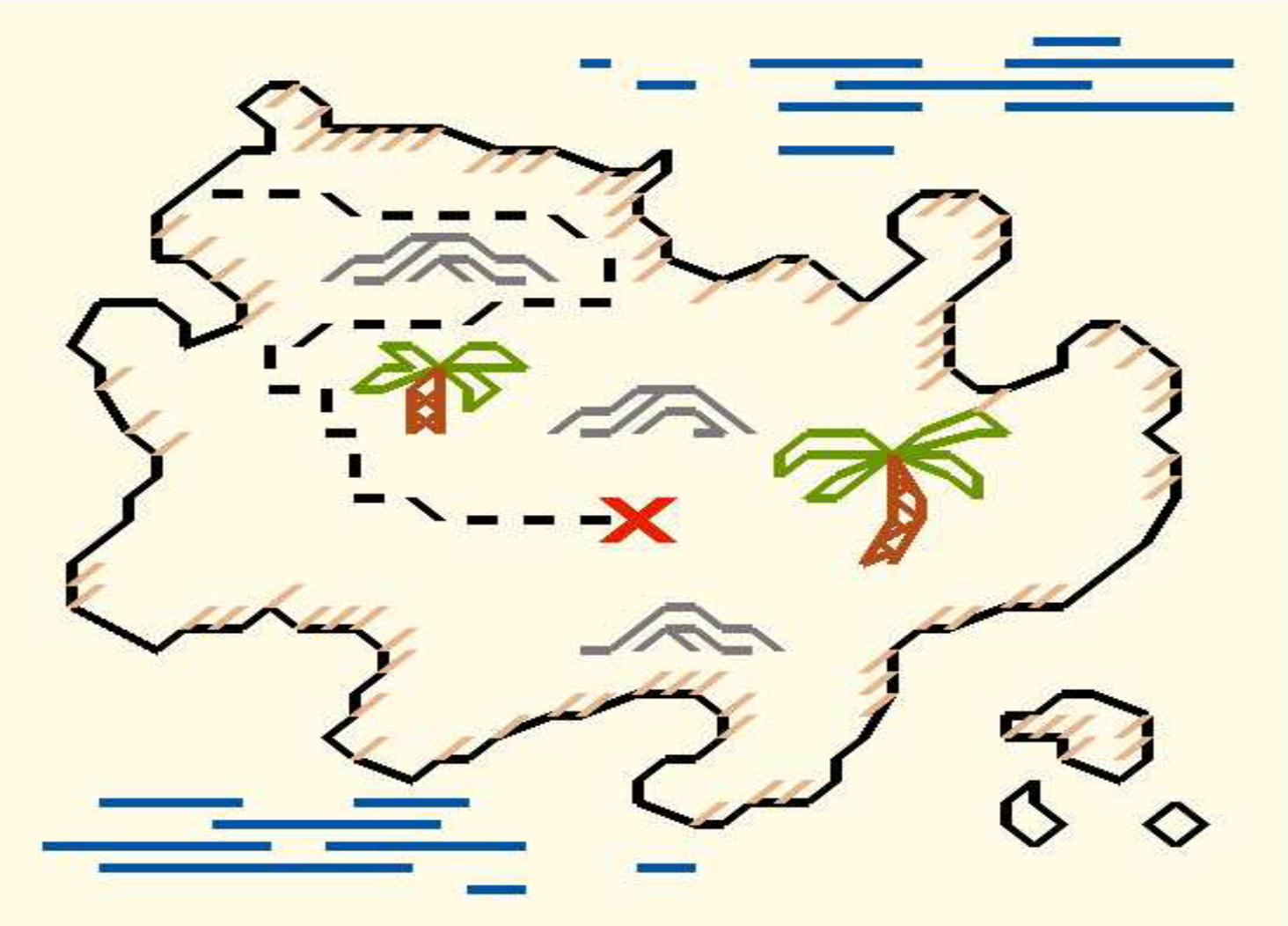
Cedric Dumoulin

Plan

- Pourquoi modéliser ?
- D'où vient UML
- Les diagrammes statiques
  - Cas d'utilisation
  - Classes
  - Objets

Modèle ?
Vous avez dit modèle ?

Exemple de modèle

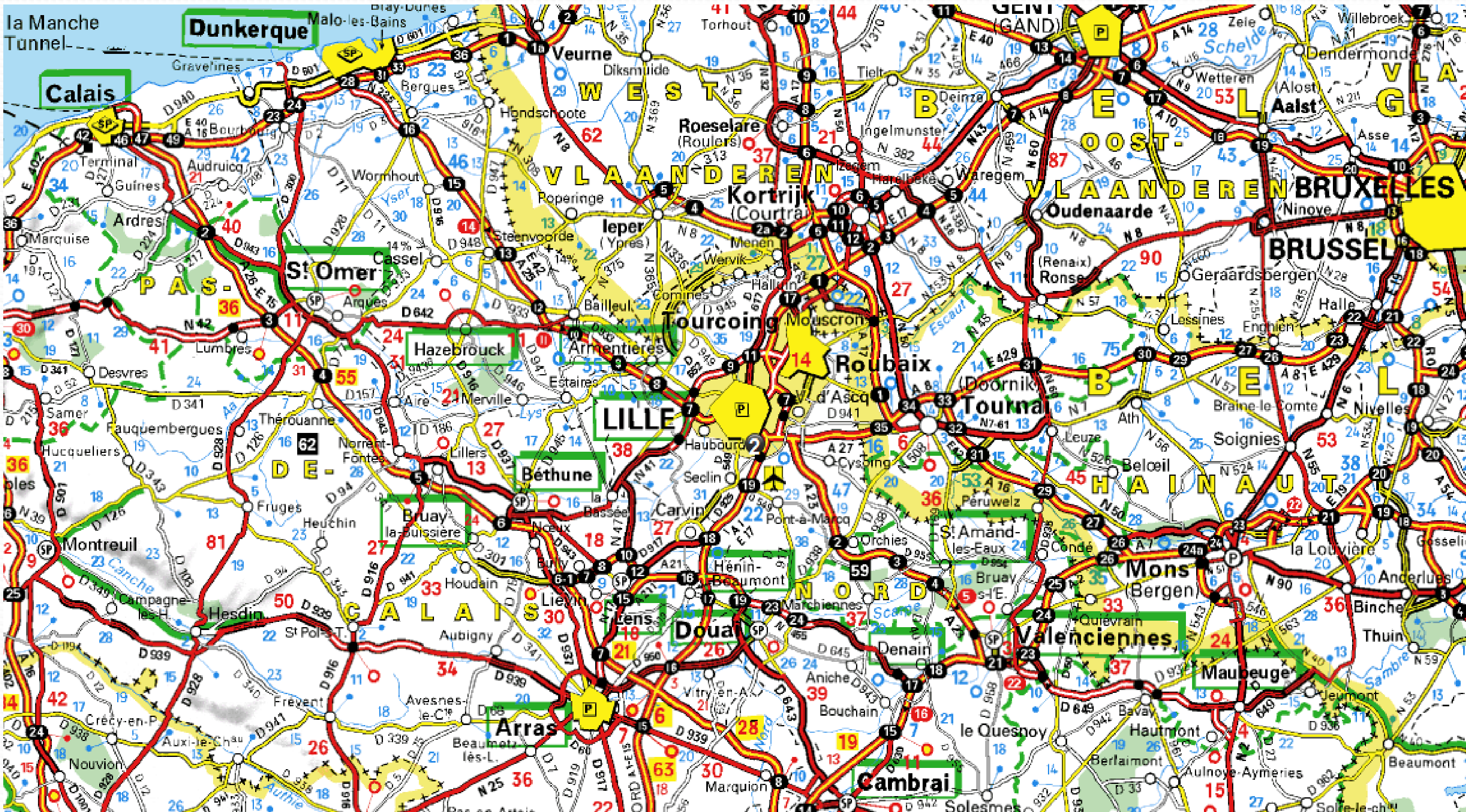


Qu'est ce qu'un modèle ?

- Définitions (Wikipedia by Google)
  - « *Un modèle mathématique est une traduction de la réalité pour pouvoir lui appliquer les outils, les techniques et les théories mathématiques* »
  - « [En économie] *Un modèle est une représentation de la réalité.* »
  - « *En informatique, un modèle a pour objectif de structurer les données, les traitements, et les flux d'informations entre entités.* »
  - C'est une abstraction de la réalité

Le modèle doit pouvoir être utilisé pour répondre à des questions sur le système modélisé

Exemple de modèle



Avantages d'un modèle

- Abstrait
  - Il fait ressortir les points importants tout en enlevant les détails non nécessaires
- Compréhensible
  - Il permet d'exprimer une chose complexe dans une forme plus facilement compréhensible par l'observateur
- Précis
  - Il représente fidèlement le système modélisé
- Prédictif
  - Il permet de faire des prévisions correcte sur le système modélisé
- Peu coûteux
  - Il est bien moins coûteux à construire et étudier que le système lui même

Les modèles et l'informatique

- pour faire un programme complexe, on a besoin de le modéliser.

- pour explorer les solutions, pour les valider, pour montrer au client ce que sera l'application.

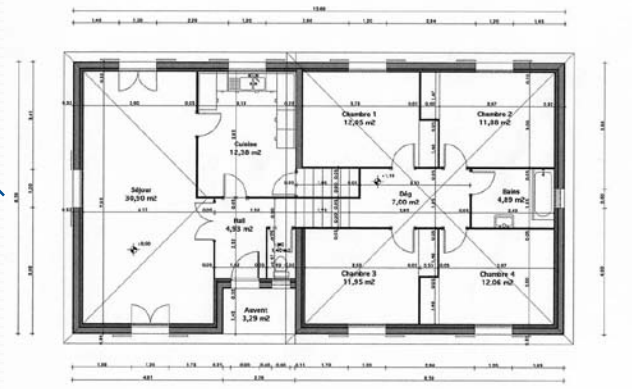
UML : Unified Modeling Langage

Un modèle, différent niveaux de granularité

LILLE

Un modèle, plusieurs points de vues

modèle



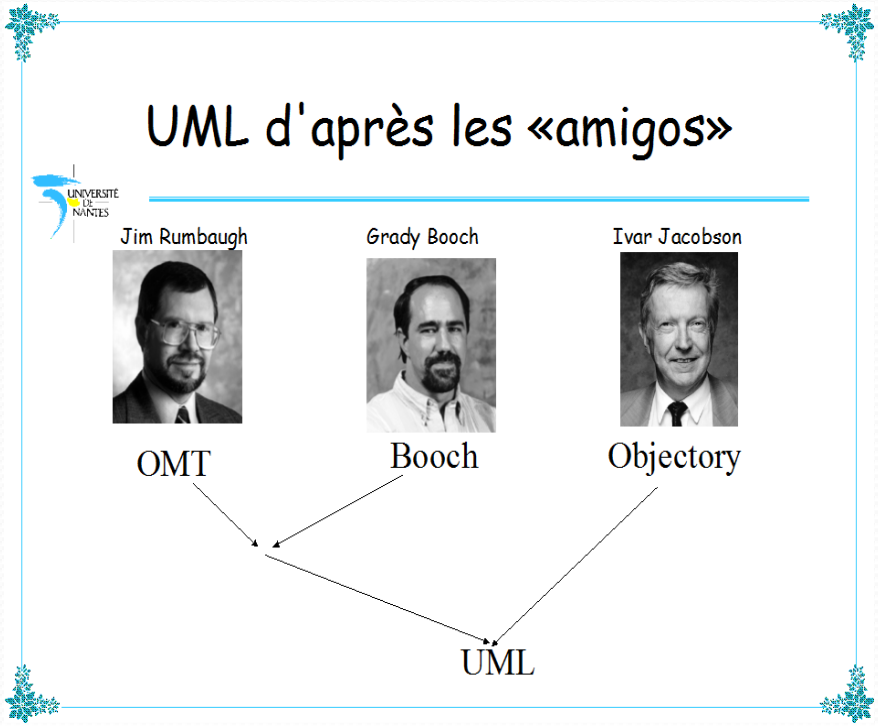
Pourquoi modéliser ?

- Pour comprendre.
  - On modélise des systèmes complexes car on ne peut pas comprendre de tels systèmes dans leur intégralité :
  - La modélisation permet de reculer les limites humaines en se focalisant sur un aspect à la fois.
- Pour borner le champ d'investigation
  - On n'a pas à comprendre tout sur tout.
- Pour communiquer
  - Nécessité d'un langage commun, précis sans ambiguïté, en minimisant au maximum les interprétations possibles. Permet de tenir compte du point de vue et de la connaissance de chacun.

Un peu d'histoire

Ou comment en est on arrivé là

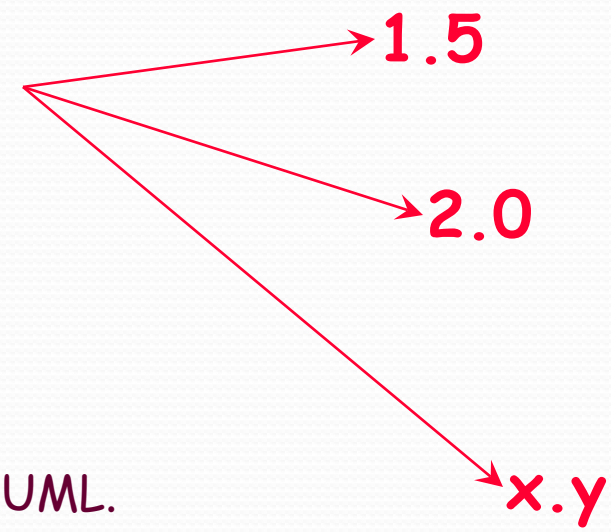
Historique : Évolution de UML



0.8 ->0.9

0.9->0.91->1.0

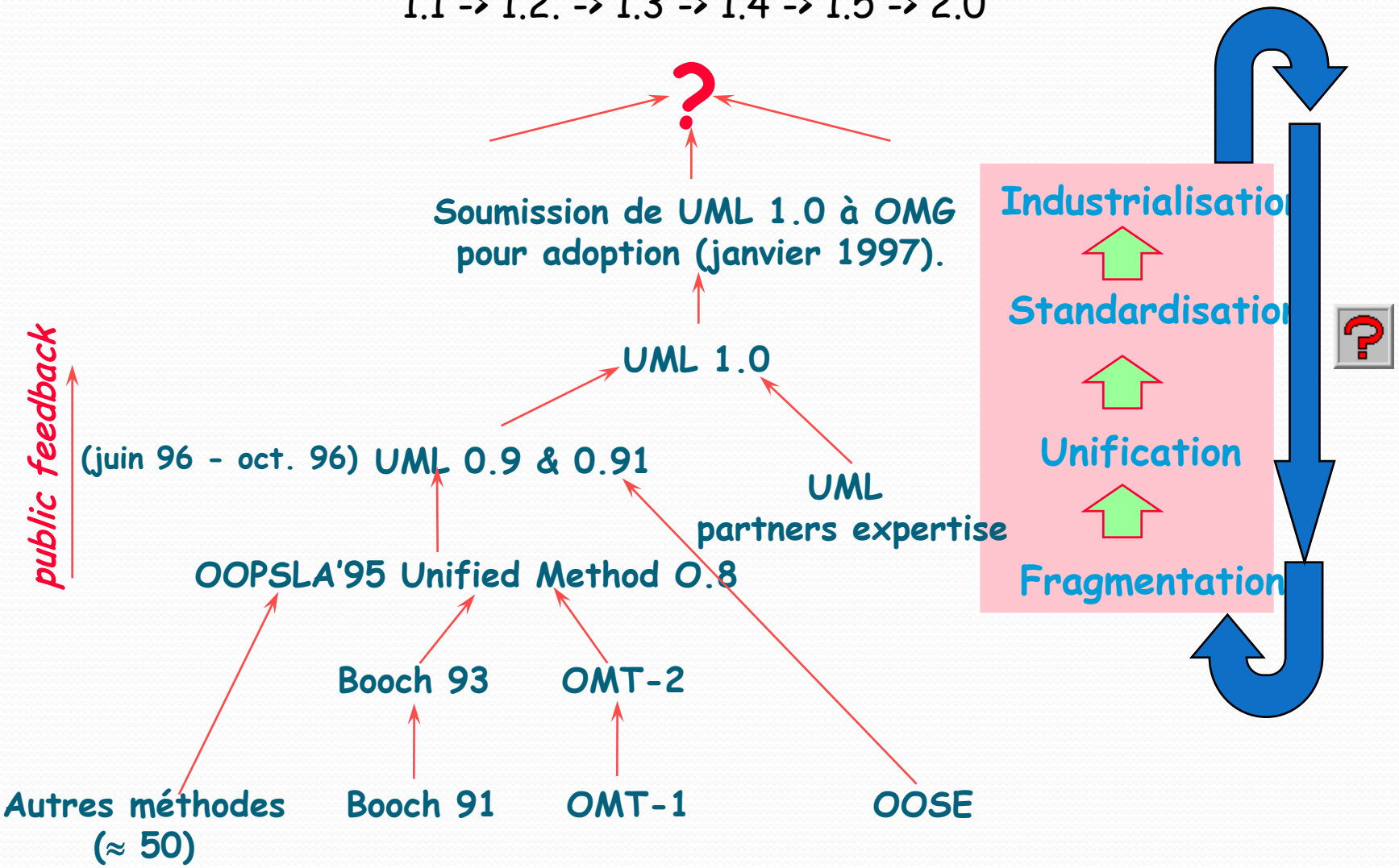
1.0->1.1->1.2->1.3->1.4



Comprendre la logique d'évolution d'UML.

De Rational à l'OMG

1.1 -> 1.2. -> 1.3 -> 1.4 -> 1.5 -> 2.0

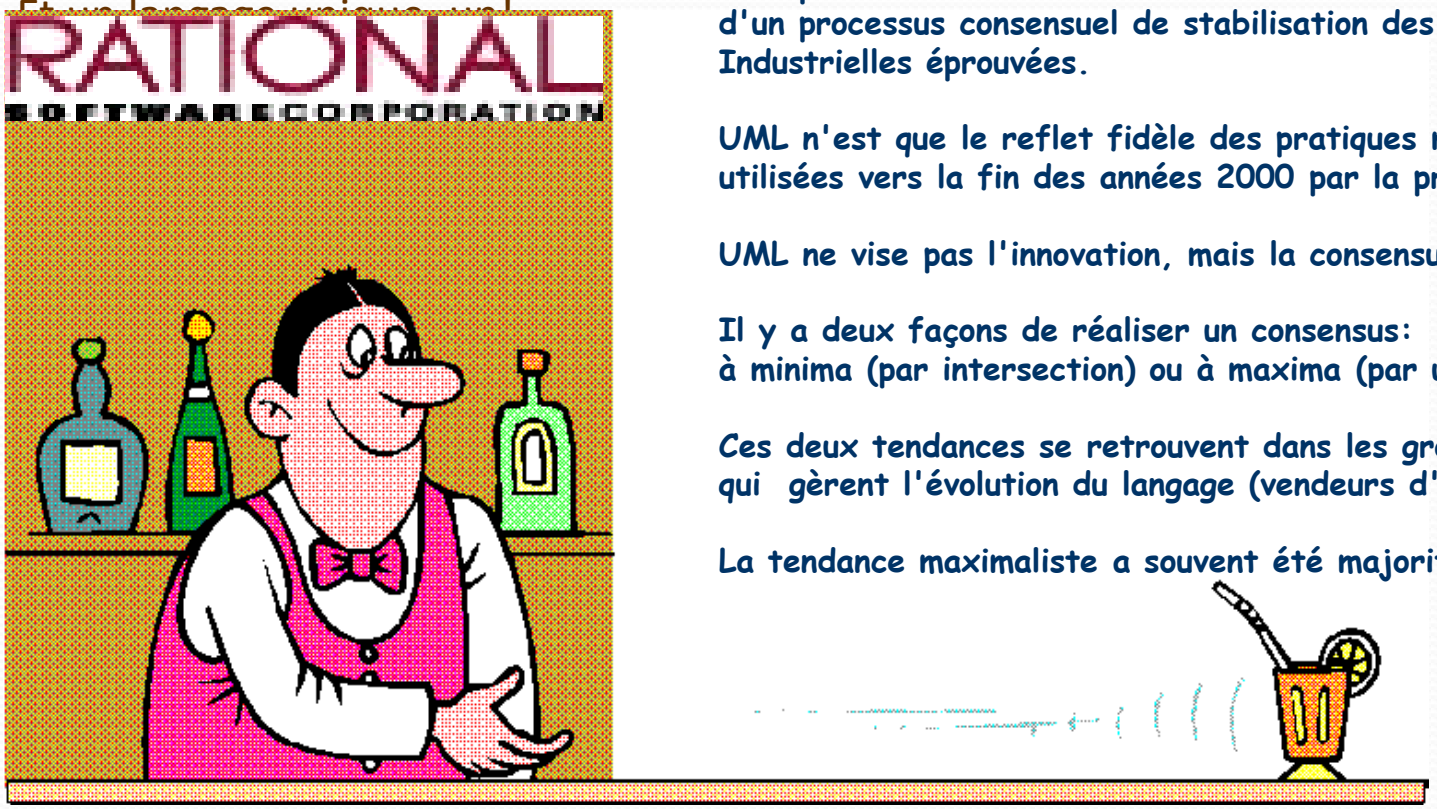


Devant et derrière,
Avant et après ...

Méthode = Langage(s) + Démarche + Outils

procédés
industriels
de production
de logiciels et
de systèmes.

OMT
SA/RT
SADT
ERD
Merise
DFD
UFD
etc.
JSD



La définition du formalisme UML ne résulte pas
d'un processus innovant de recherche mais
d'un processus consensuel de stabilisation des pratiques
Industrielles éprouvées.

UML n'est que le reflet fidèle des pratiques majoritaires
utilisées vers la fin des années 2000 par la profession.

UML ne vise pas l'innovation, mais la consensualité.

Il y a deux façons de réaliser un consensus:
à minima (par intersection) ou à maxima (par union).

Ces deux tendances se retrouvent dans les groupes d'influence
qui gèrent l'évolution du langage (vendeurs d'AGL, etc.)

La tendance maximaliste a souvent été majoritaire (!)

UNIFIED MODELING LANGUAGE

OMG
OBJECT MANAGEMENT GROUP

UML n'est pas un projet de recherche

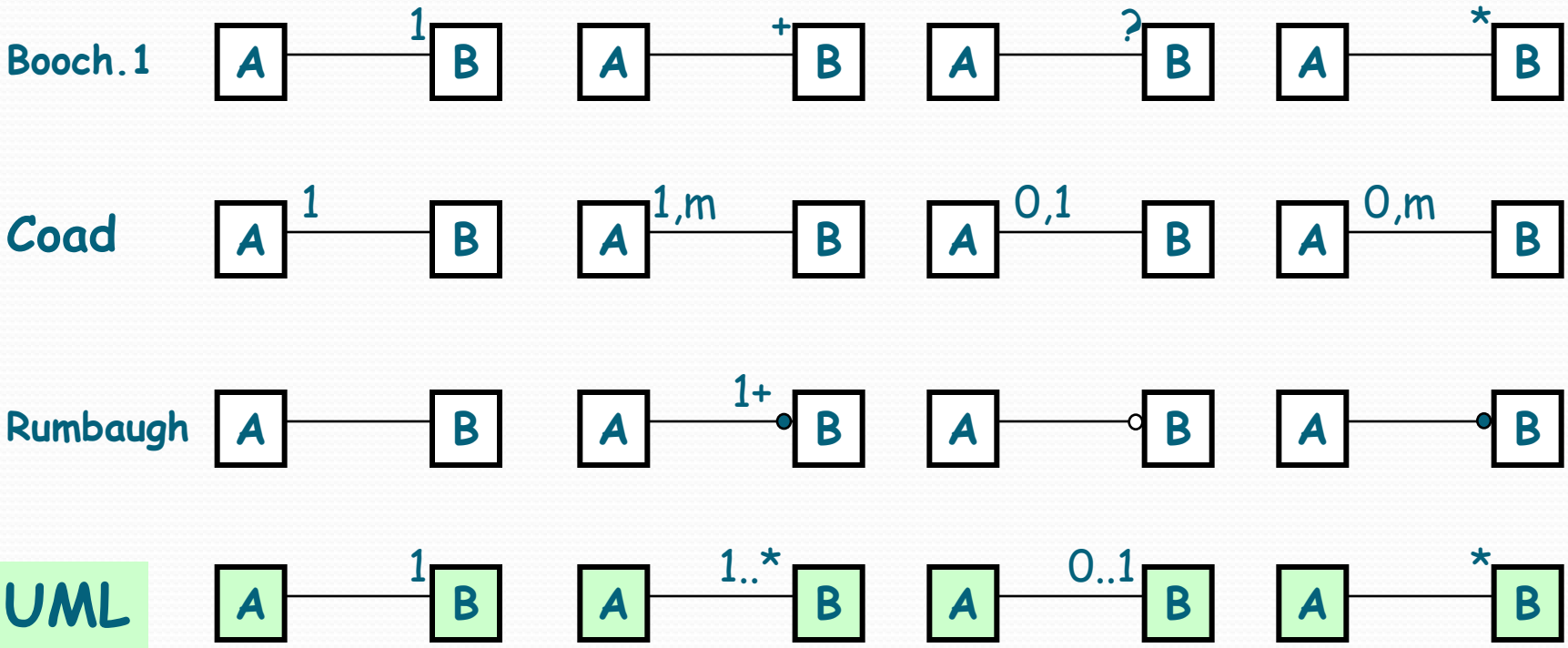
"In short: the time for experimentation is past; the time for stability and use is now."

Grady Booch
Chief Scientist
Rational Software Corporation

Evolution de la terminologie

Booch	Classe	Utilise	Hérite	Contient
Coad	Classe/Objet	Connexion d'instance	Gen/Spec	Tout/Partie
Jacobson	Objet	Accointance Association	Hérite	ConsisteEn
Odell	Objet/Type	Relation	Sous-type	Composition
Rumbaugh	Classe	Association	Généralisation	Aggrégation
Shlaer/Mellor	Objet	Relation	Sous-type	N/D
UML	Classe	Association	Généralisation	Aggrégation

Quelques notations de cardinalité



un A toujours associé avec un B.

un A toujours associé avec un ou plusieurs B.

un A associé avec zéro ou un B.

un A associé avec zéro, un ou plusieurs B.

Aujourd'hui

- UML 2 - http://www.omg.org/spec/UML/

OMG Formally Released Versions of UML		
2.4.1	August 2011	http://www.omg.org/spec/UML/2.4
2.3	May 2010	http://www.omg.org/spec/UML/2.3
2.2	February 2009	http://www.omg.org/spec/UML/2.2
2.1.2	November 2007	http://www.omg.org/spec/UML/2.1.2
2.1.1	August 2007	http://www.omg.org/spec/UML/2.1.1

UML 2

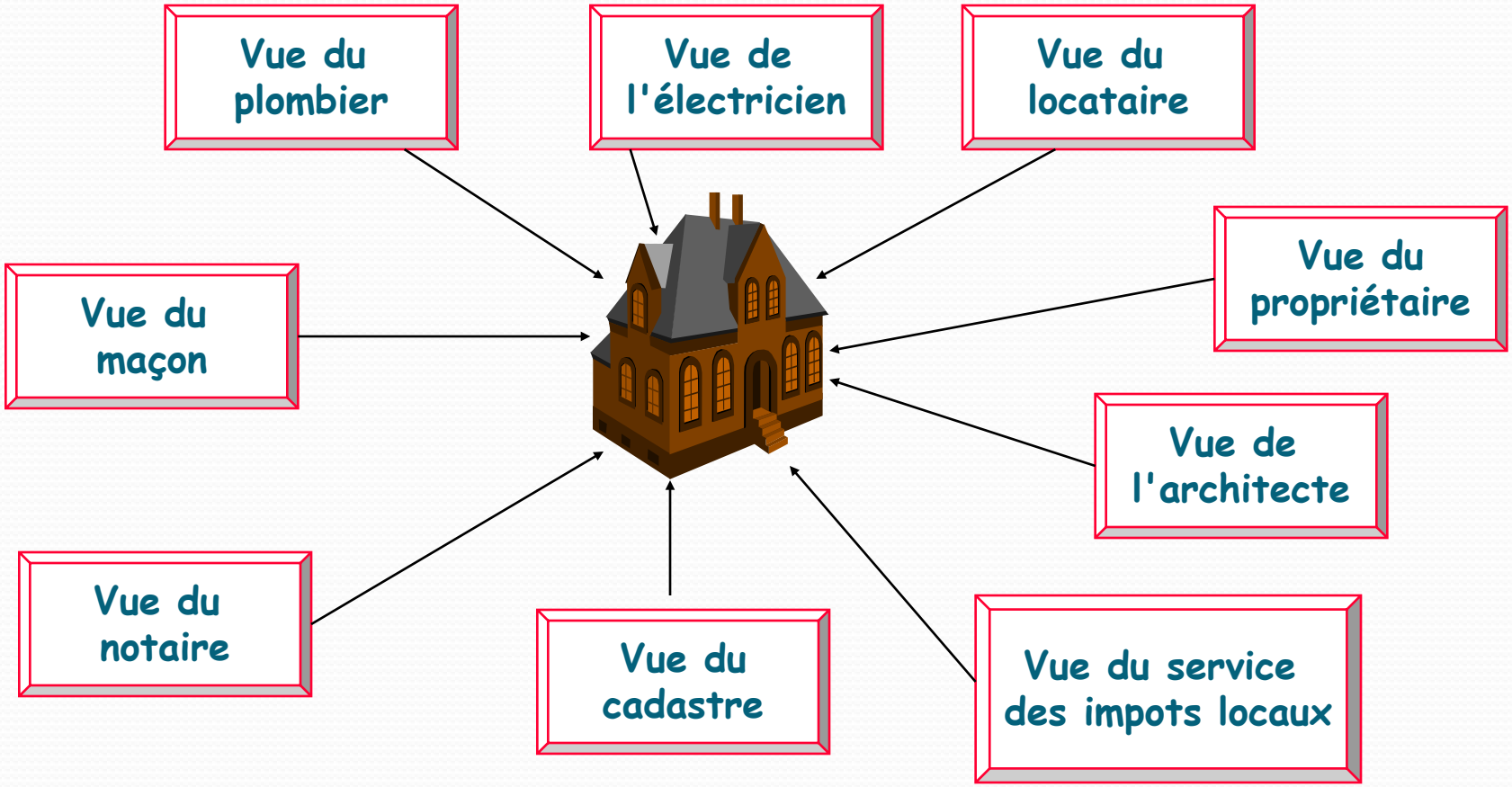
Objectifs d' UML

- Montrer les limites d'un système et ses fonctions principales (du point de vue des utilisateurs)
  - à l'aide des cas d'utilisation et des acteurs
- Illustrer les réalisations de CU
  - à l'aide de diagrammes d'interaction
- Modéliser la structure statique d'un système
  - à l'aide de diagrammes de classes, associations, contraintes
- Modéliser la dynamique, le comportement des objets
  - à l'aide de diagrammes de machines d'états
- Révéler l'implantation physique de l'architecture
  - avec des diagrammes de composants et de déploiement
- Possibilité d'étendre les fonctionnalités du langage
  - avec des stéréotypes
- Un langage utilisable par l'homme et la machine
  - permettre la génération automatique de code, et la rétro-ingénierie

Modèle, vues et diagrammes

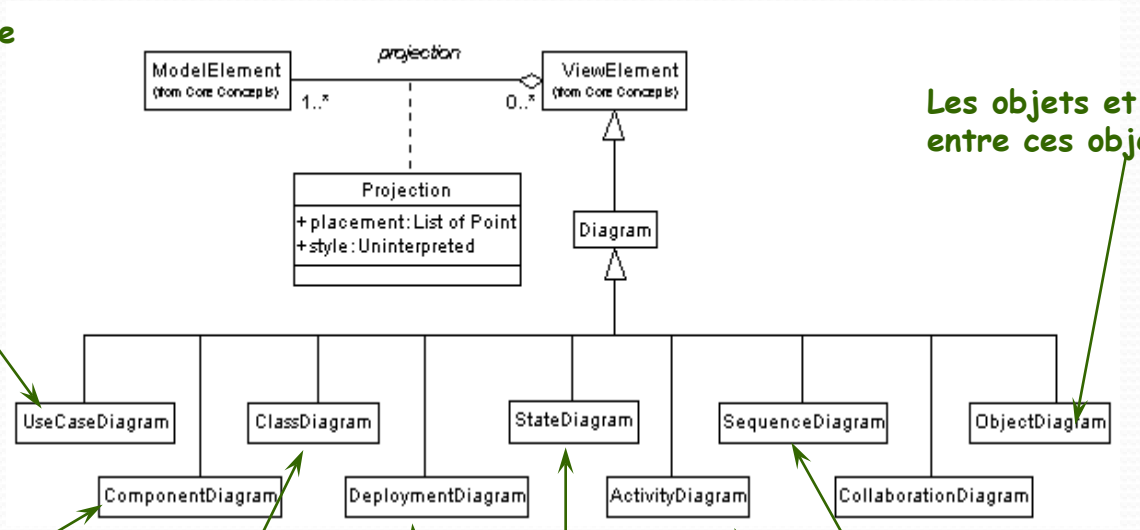
- Modèle
  - abstraction d'un système composée d'un ensemble d'éléments de modèle
  - ce qui est construit par et ce qui est perçu au travers des diagrammes (par le concepteur, le lecteur)
  - conforme au métamodèle UML
- Vue
  - projection d'un modèle suivant une perspective
  - omet les éléments non pertinents pour cette perspective.
  - Elle se manifeste dans des diagrammes
  - ex. : vue statique, vue fonctionnelle…
- Diagramme
  - représentation visuelle simplifiée et structurée des concepts du modèle.
  - Syntaxe concrète des concepts
  - ex. : diagramme de classes, de séquences…

Vues multiples (aspects d'un système logiciel)



Vues multiples (aspects d'un système logiciel)

Fonctions du système du point de vue de l'utilisateur.



Les objets et les relations de base entre ces objets.

Composants physiques d'une application.

Représentation du comportement en termes d'états.

Représentation des objets, des liens mutuels et des interactions potentielles.

Schémas de l'installation des composants sur les dispositifs matériels.

Représentation des objets et de leurs interactions temporelles.

Structure statique des classes et des relations entre ces classes.

Représentation du comportement des opérations en termes d'actions.

Diagrammes en UML 2.x.

- **Structure - Structural Modeling Diagrams** (6)
  - Package diagrams - are used to divide the model into logical containers, or 'packages', and describe the interactions them a high level.
  - Class or Structural diagrams define the basic building blocks of a model: the types, classes and general materials used to construct a full model.
  - Object diagrams show how instances of structural elements are related and used at run-time.
  - Composite Structure diagrams provide a means of layering an element's structure and focusing on inner detail, construction and relationships.
  - Component diagrams are used to model higher level or more complex structures, usually built up from one or more classes, and providing a well defined interface.
  - Deployment diagrams show the physical disposition of significant artifacts within a real-world setting.
- **Comportement - Behavioral Modeling Diagrams** (7)
  - Use Case diagrams are used to model user/system interactions. They define behavior, requirements and constraints in the form of scripts or scenarios.
  - Activity diagrams have a wide number of uses, from defining basic program flow, to capturing the decision points and actions within any generalized process.
  - State Machine diagrams are essential to understanding the instant to instant condition, or "run state" of a model when it executes.
  - Communication diagrams show the network, and sequence, of messages or communications between objects at run-time, during a collaboration instance.
  - Sequence diagrams are closely related to communication diagrams and show the sequence of messages passed between objects using a vertical timeline.
  - Timing diagrams fuse sequence and state diagrams to provide a view of an object's state over time, and messages which modify that state.
  - Interaction Overview diagrams fuse activity and sequence diagrams to allow interaction fragments to be easily combined with decision points and flows.

Diagramme

- Plusieurs diagrammes (vues) par modèle
- Ce concentre sur un idée, un concept
- Ne montre que les détails pertinents pour cette idée
- N'importe quelle information peut être supprimée
  - pas d'inférence due à l'absence d'un élément

- Le diagramme est une vue (partielle) sur le modèle !!

Qualités d'UML

- S'appuie sur un métamodèle décrit en UML
  - validation et précision de la notation
- Permet la mise en place du paradigme Objet
- Langage graphique
  - relativement compréhensible par tous
- Manipulable à l'aide d'outils (modeleurs UML)
- Universalité
  - utilisable dans tous les domaines (télécom, santé, banques, transport, …)
  - et pour tout type de système (du SI aux Intranets en passant par les systèmes embarqués temps réel. mais également des systèmes non informatiques comme par exemple l'organisation du système qualité d'un hôpital).
- UML est du domaine public,
  - tout le monde peut l'utiliser.

Défauts d'UML

- C'est un langage, pas une méthode
  - Sémantique faible
  - Pas de méthodologie associée
  - En particulier, on ne nous dit pas comment utiliser UML, l'utilisation des différents outils à travers le cycle de vie d'un développement n'est pas définie. Une méthode s'impose!
- Manque de précision
  - ➔ OCL
- Difficilement manipulable sans outil.

- Son utilisation nécessite une très grande rigueur qui peut être occultée par le coté visuel de la notation

Conception et UML

- Différentes façons de penser, différentes méthodologies …
  - ➔ différentes façons d'utiliser UML
  - ➔ essayer de comprendre le point de vue de l'auteur pour chaque modèle d'une application
- UML n'est pas une méthode…
  - … mais des principes de conception orientée objet sont sous jacents
    - aux diagrammes
    - aux façons de présenter les diagrammes
- donc
  - difficile de présenter uniquement les diagrammes
  - on parlera aussi de méthode, de bonnes pratiques

Diagrammes de cas d’utilisations

Cas d'utilisation

- Technique pour capturer les exigences fonctionnelles d'un système
  - déterminer ses limites
  - déterminer ce qu'il devra faire
    - mais pas comment il devra le faire
    - point de vue de l'utilisateur
- Pour cela
  - déterminer les rôles qui interagissent avec lui
    - Acteurs
  - déterminer les grandes catégories d'utilisation
    - cas d'utilisation
  - décrire textuellement des interactions
    - scénarios

Le diagramme de cas d'utilisation

- Composé de
  - Cas d'utilisation
  - Acteurs
  - de relations de dépendance, de généralisation, d'association
  - paquetage
  - notes
  - contraintes

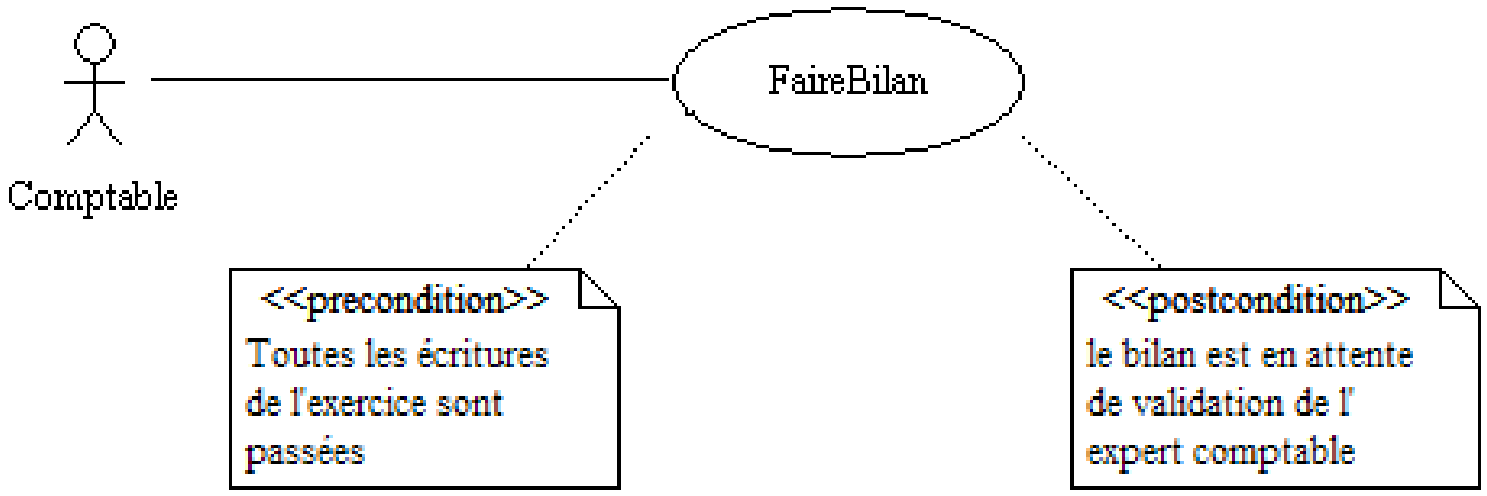
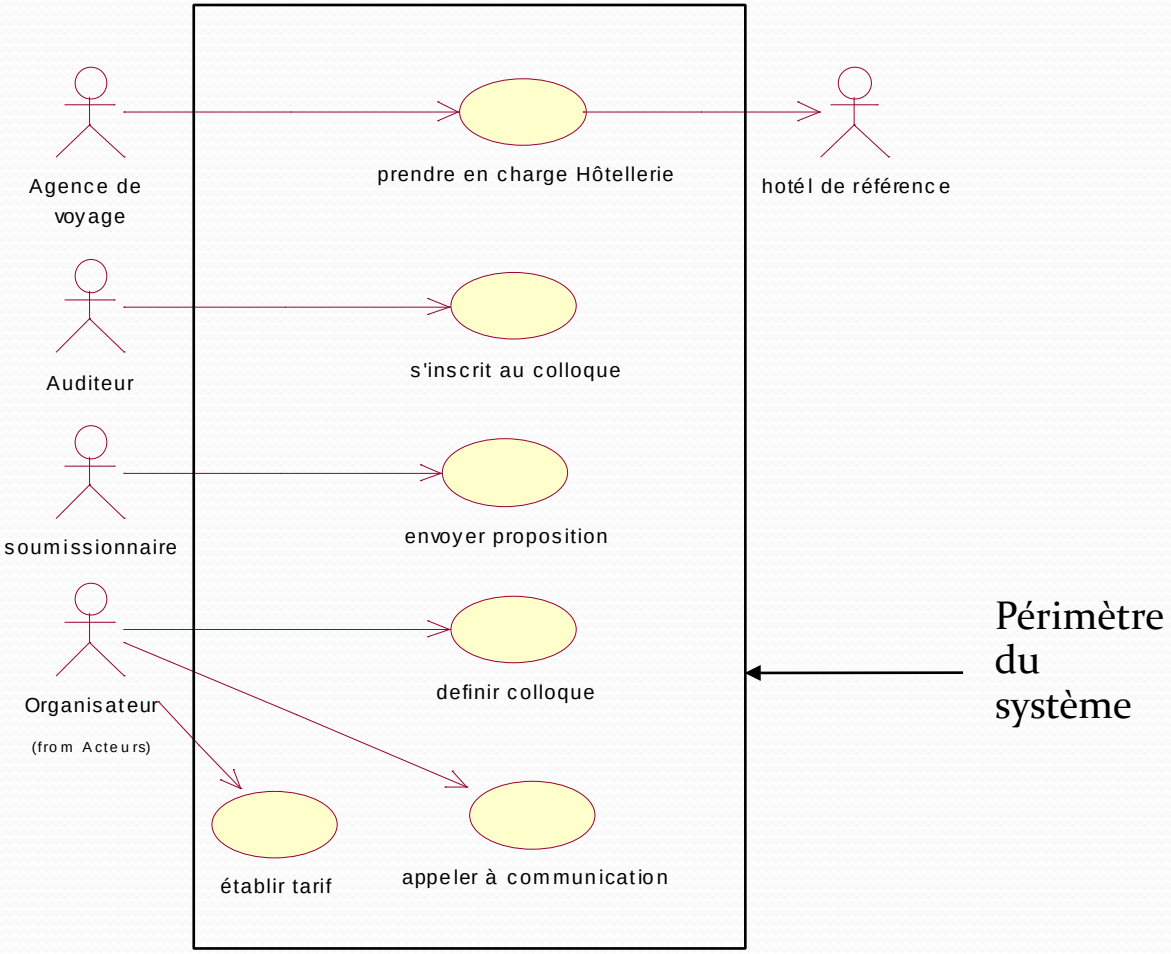


Diagramme de cas d'utilisation

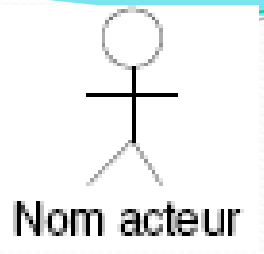


Utilisation

- Passer du flou du cahier des charges à des fonctionnalités exprimées dans le langage du domaine
  - dialogue entre concepteurs et utilisateurs
- Pour l'expression complète des besoins, tout au long d'un processus de conception de système d'information
- Attention
  - ce ne sont pas les diagrammes de CU qui sont importants, mais les descriptions textuelles des scénarios

- Un diagramme de cas d'utilisation
  - ne représente jamais le déroulement d'un système,
  - il dit ce qu'il fait (ou fera),
  - Il ne dit jamais comment il le fait (fera).

Acteur



- Entité (humain ou machine) située hors du système
  - permettant d'en déterminer les limites
  - jouant un rôle par rapport à lui
  - déclenchant un stimulus initial entraînant une réaction du système
  - ou au contraire étant sollicité par le système au cours d'un Scénario
- Un acteur est décrit précisément en quelques lignes
- 4 grandes catégories
  - acteurs principaux (fonctions principales du système)
  - acteurs secondaires (administration / maintenance)
  - matériel externe
  - autres systèmes

Cas d'utilisation

- Ensemble de séquences d'action réalisées par le système, produisant un résultat observable pour un acteur particulier
  - ex. identification, retrait de liquide
- Un CU
  - définit un ensemble de scénarios d'exécution incluant les cas d'erreurs

- Un CU recense les informations échangées et les étapes dans la manière d'utiliser le système, les différentes points d'extension et cas d'erreur

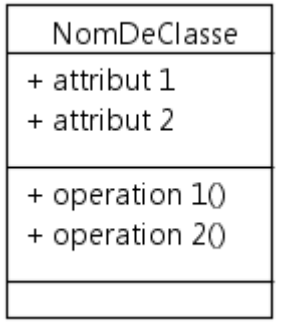
Diagrammes de classes

Diagrammes de classes : présentation générale

- Diagrammes fondamentaux
  - les plus connus, les plus utilisés
- Présentent la vue statique du système
  - représentation de la structure et des déclarations comportementales
  - classes, relations, contraintes, commentaires…
- Permettent de modéliser plusieurs niveaux
  - conceptuel (domaine, analyse)
  - implémentation (code)

Classes

- Descripteurs de jeux d'objets
  - structure / comportement / relations / sémantique communs
- Représentation
  - rectangle à trois compartiments (ou plus)
    - Nom
    - Attributs
    - Opérations
  - plus ou moins de détails suivant les besoins
- Nom : singulier, majuscule (en général)
  - ex. : Fichier, Client, Compte, Chat



Relations entre classes/ liens entre objets

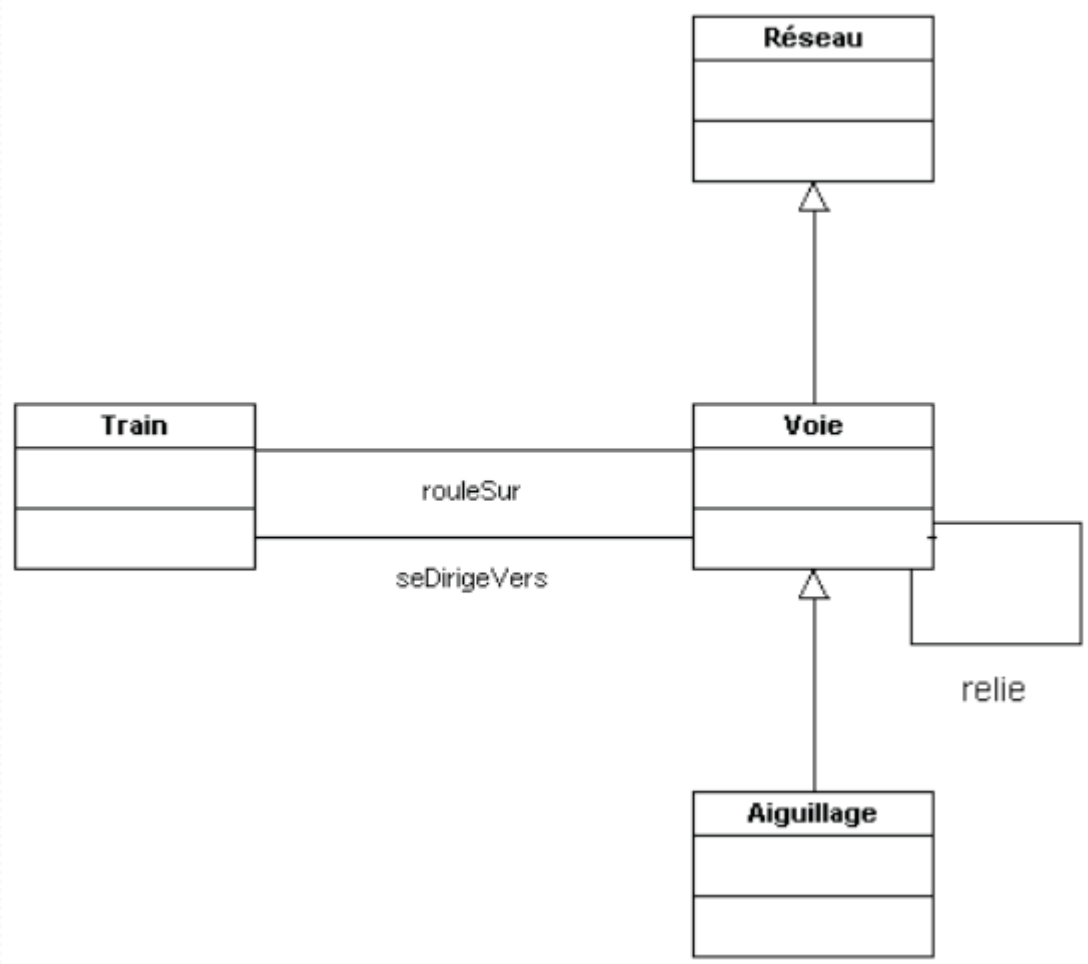
- Association
  - les instances des classes sont liées
  - possibilité de communication entre objets
  - relation forte : composition

- Généralisation/spécialisation
  - les instances de la sous-classe sont des instances de la super-classe (niveau conceptuel)
  - héritage (niveau implémentation)

- Dépendance
  - la modification d'une classe peut avoir des consequences sur une autre

- Réalisation
  - une classe réalise une interface

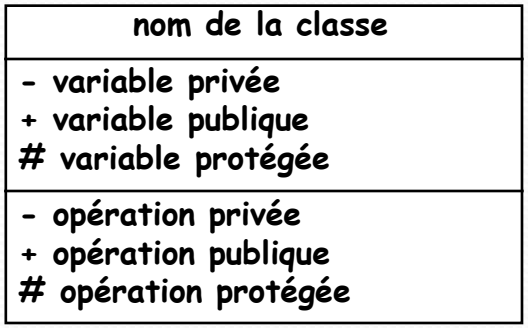
Exemple

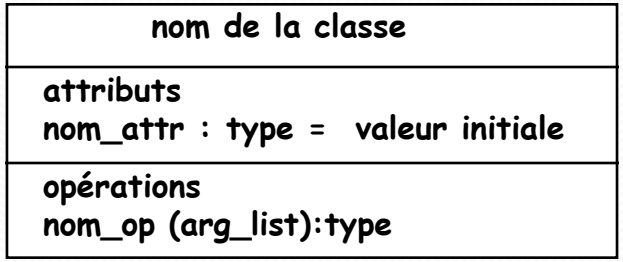

Réseau
Train
Voie
rouleSur
seDirigeVers
relie
Aiguillage

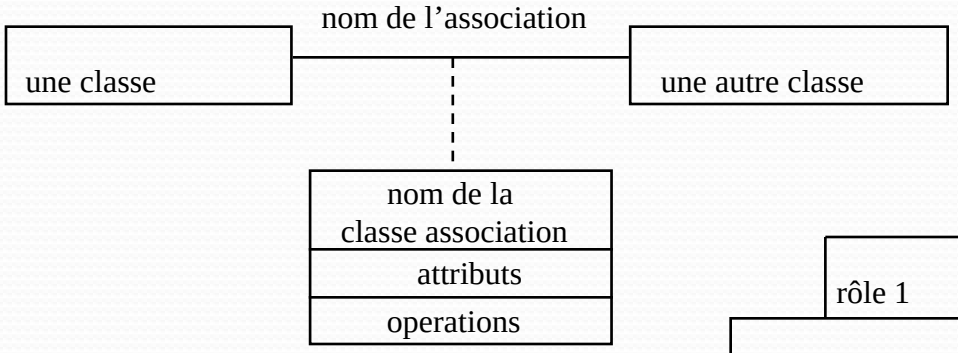
Utilisation des diagrammes de classes

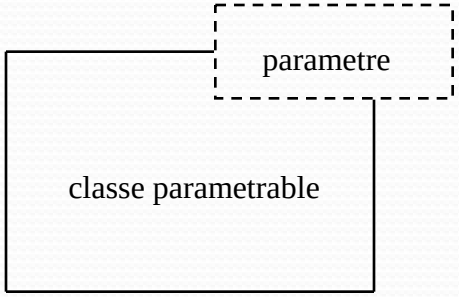
- Expression des besoins
  - modélisation du domaine
- Conception
  - spécification : gros grain
- Construction
  - implémentation : précis
  - rétro-ingénierie

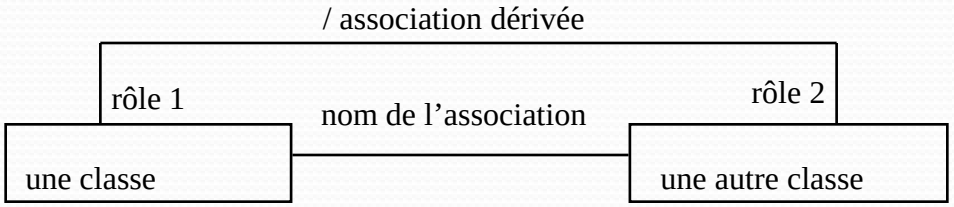
Diagrammes de classe.











Visibilité des propriétés

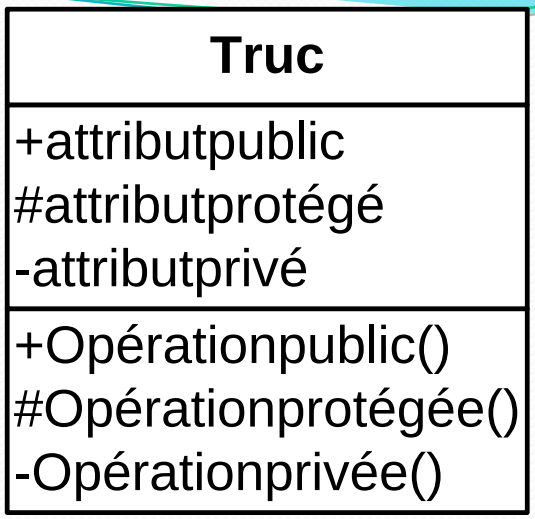
- Public +

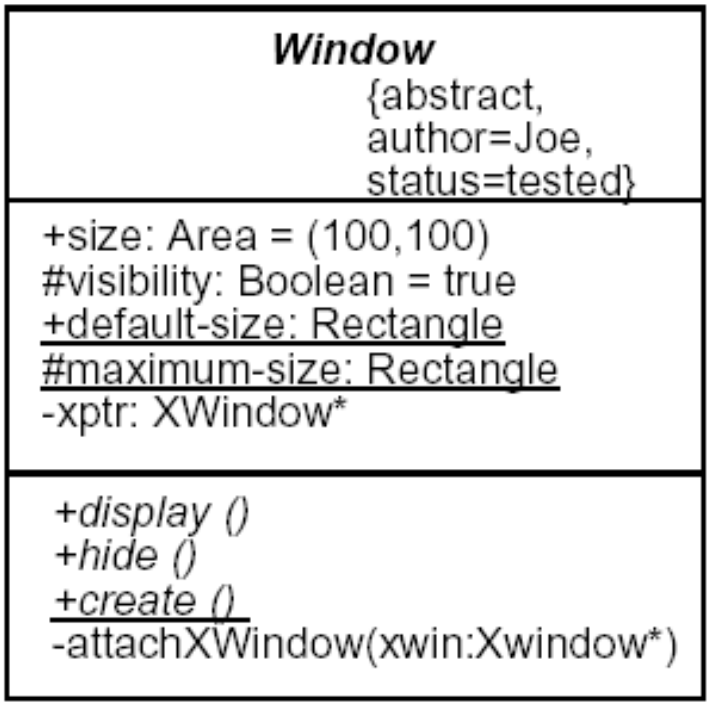
  Visible à l'extérieur de la classe
- Protégé #

  Visible seulement par les descendants
- Private -

  Visible à l'intérieur des méthodes
- Souligné

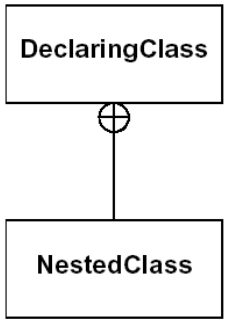
  Variable/opération de classe





Variantes de classes

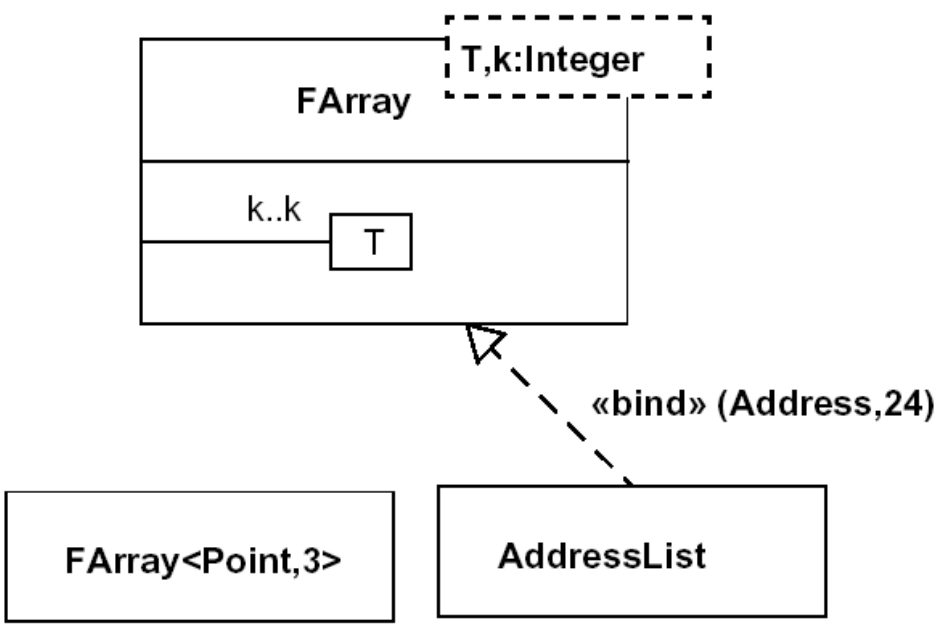
- Classe emboîtée
  - Déclarée dans la portée lexicale d'une autre classe.



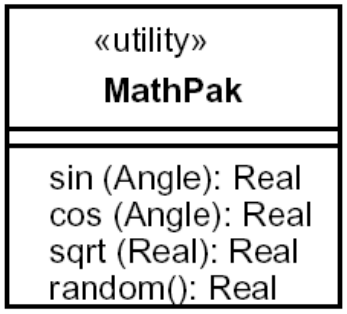

```
public class DeclaringClass {

  public class NestedClass {

  }
}
```

Variantes de classes

- Classe paramétrée
  - Modèle de classe
  - Paramètres Formels génériques (types, opérations…)

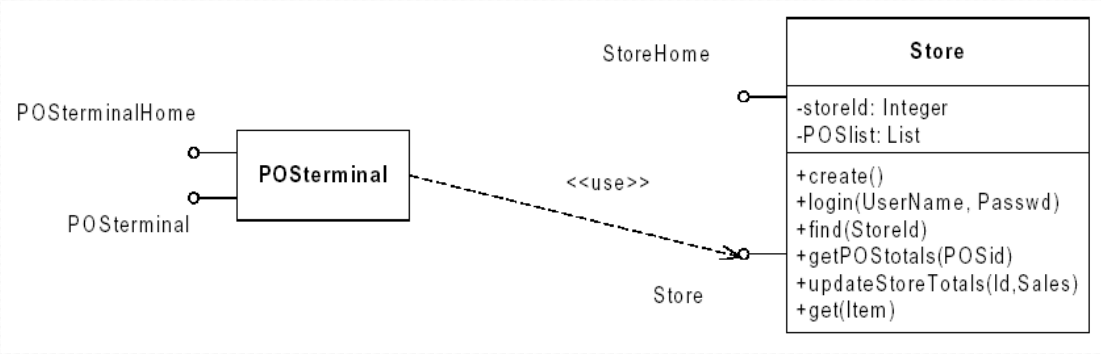


- Classe utilitaire
  - Espace de nommage, groupement de variables + opérations



Variantes de classes

- Interfaces
- Spécification des opérations visibles
  - Classes, paquetage, composants
- Contrat sans implémentation
  - Pas d'attributs, de méthodes, de relations
  - Peut être cible d'une association monodirectionnelle
- Deux représentations graphique
  - Stéréotype, lollipop



Variantes de classes

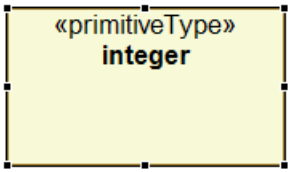
- Metaclasse

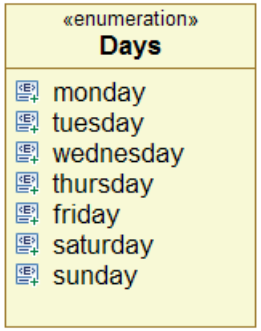
  La classe d'une classe.
  Pour la métamodèlisation

- Datatype et PrimitiveType

- Énumération

  Ensemble de littéraux d'énumération, ordonné.





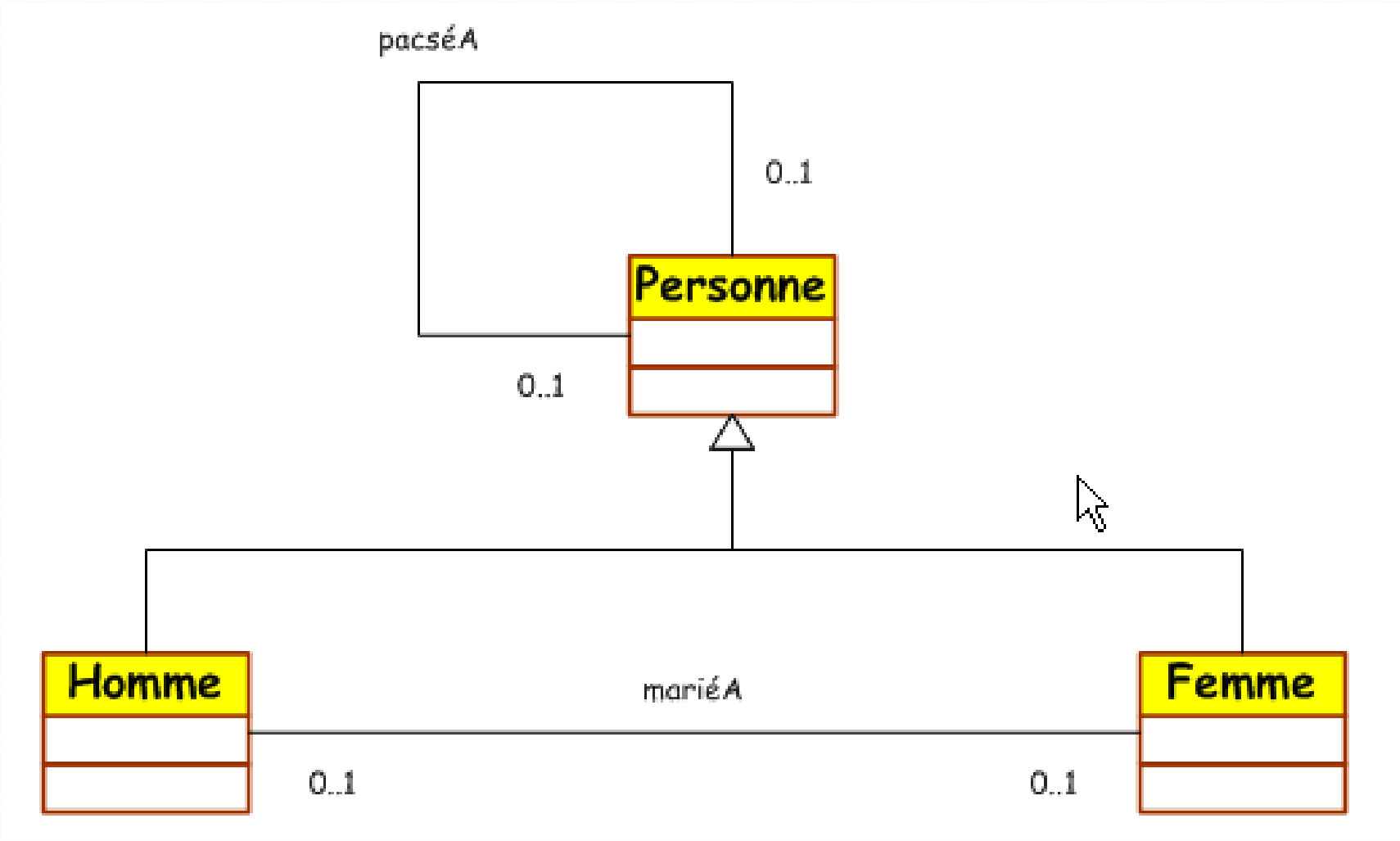
Les relations entre classes

- L'association
- L'agrégation
- La composition

- La généralisation
- La dépendance

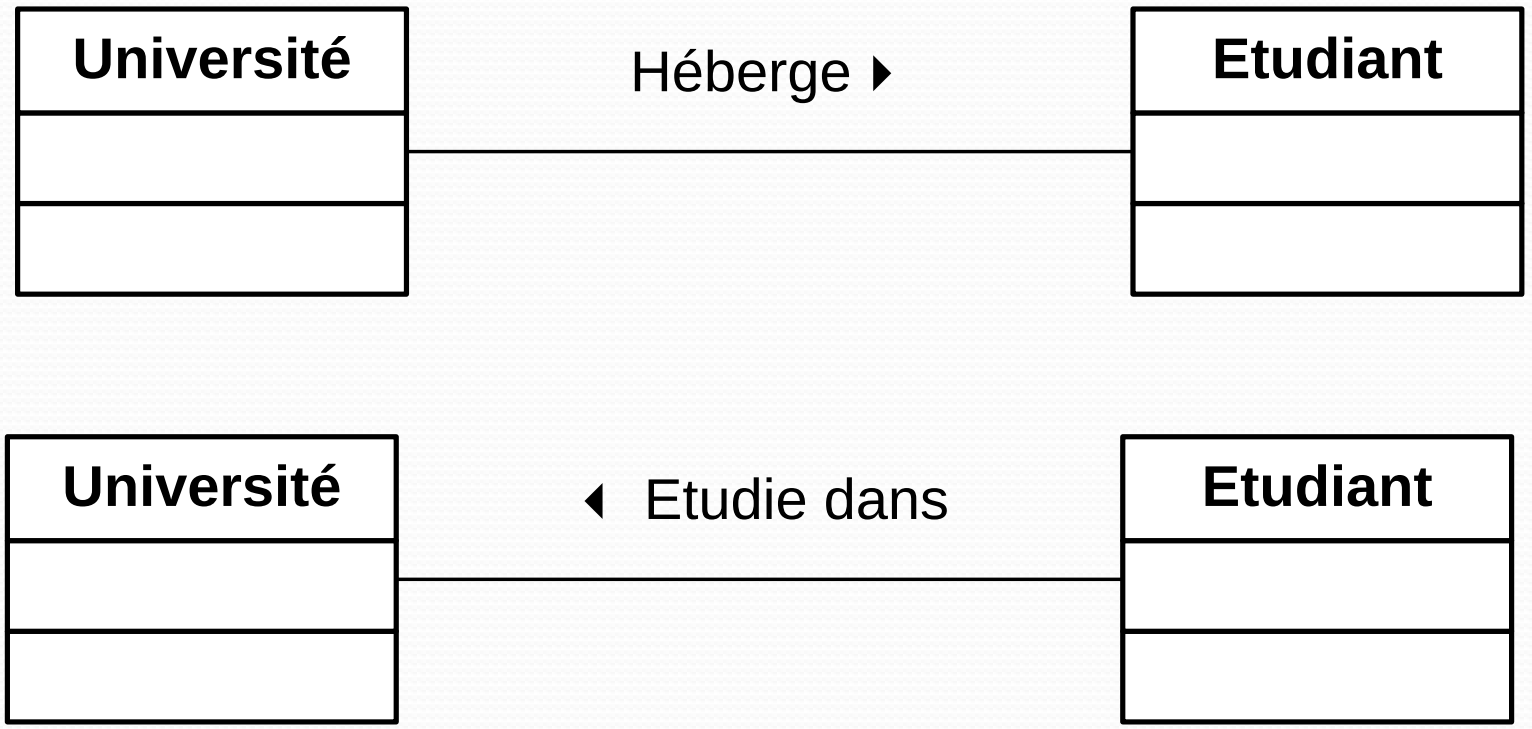
- L'association exprime une connexion sémantique bidirectionnelle entre classes
- Une association est une abstraction des liens qui existent entre les objets instances des classes associées

Exemple



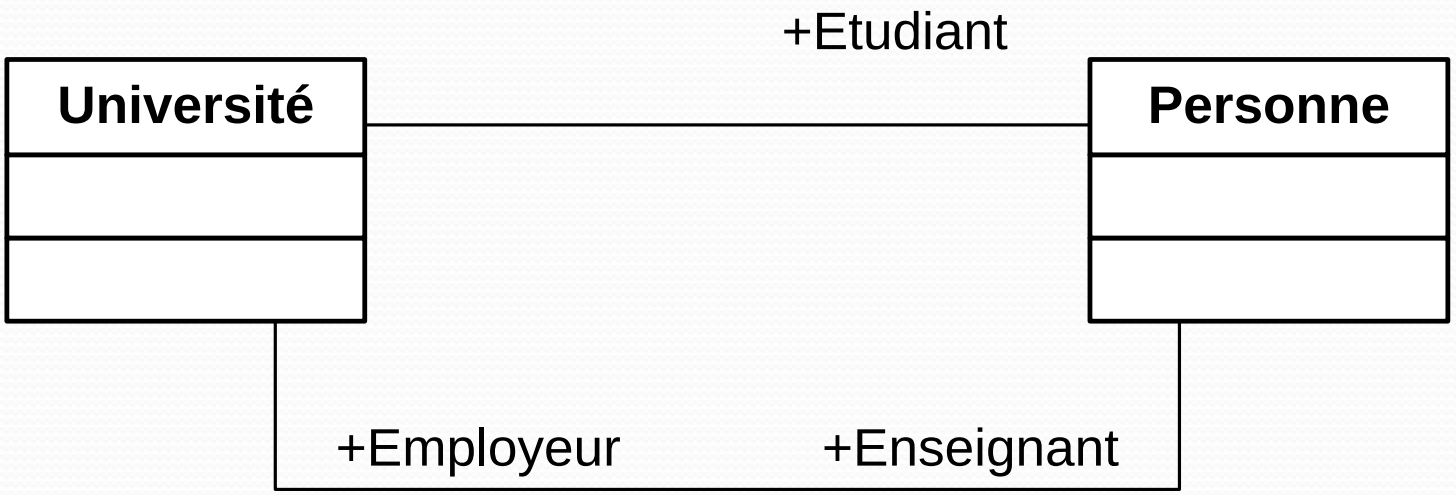
Nommage des associations

- Indication du sens de lecture

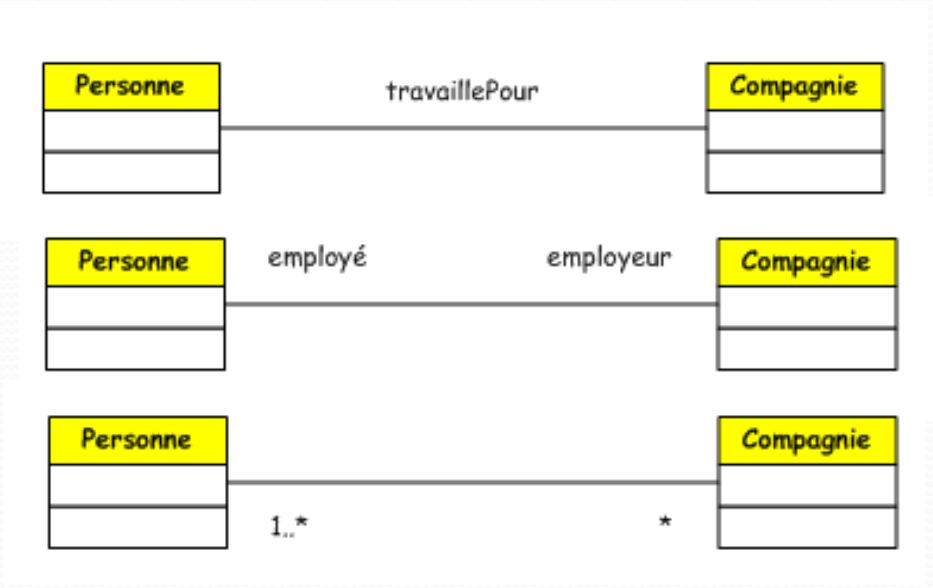


Nommage des rôles

- Le rôle décrit une extrémité d'une association



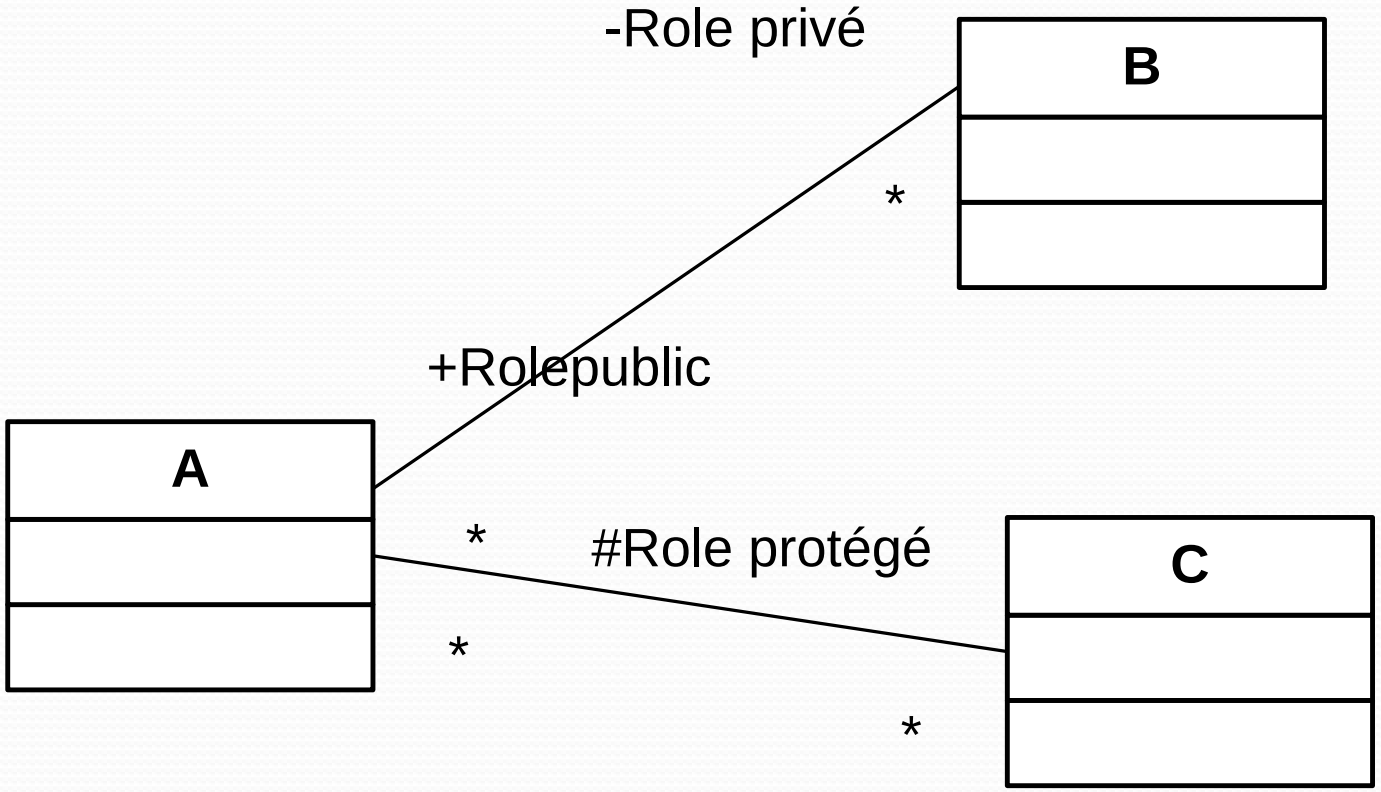
Multiplicité des rôles



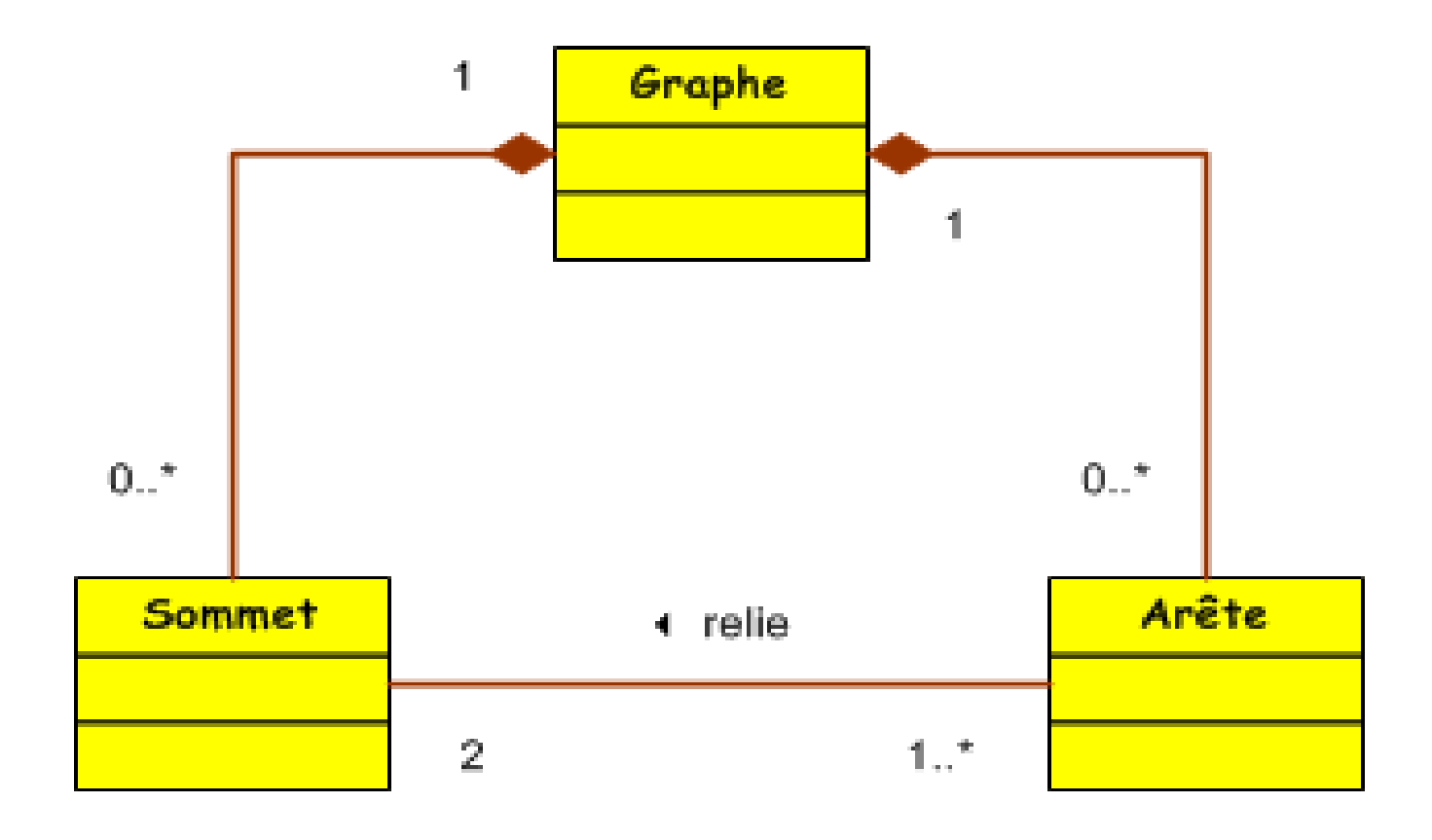
1	Un et un seul
0..1	Zéro ou un
M .. N	De M à N (entiers naturels)
*	Plusieurs
0 .. *	De zéro à plusieurs
1 .. *	D'un à plusieurs

Visibilité des rôles

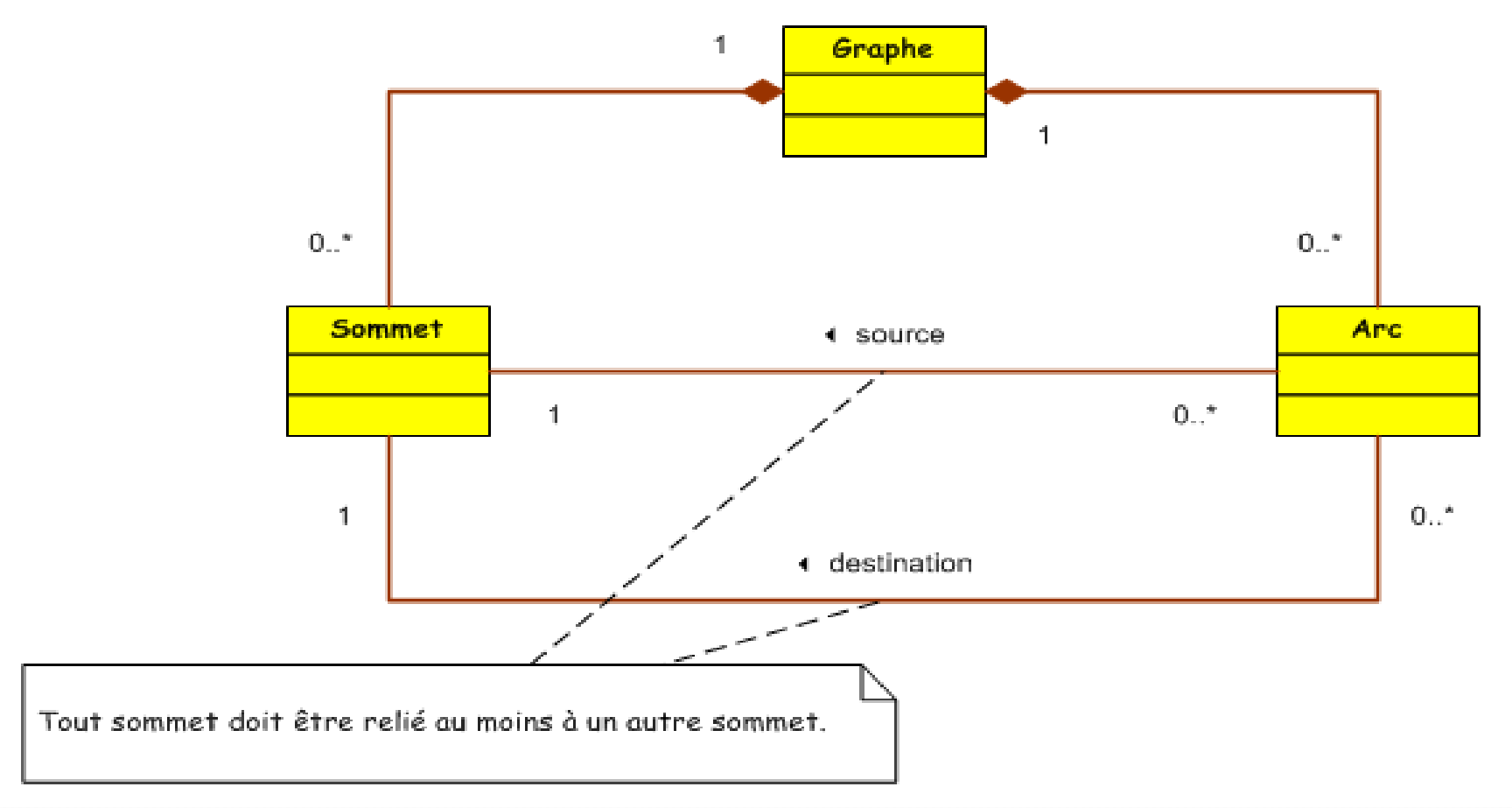
- Public
- Protégé
- Private



Exemple : graphe non orienté



Exemple : graphe orienté

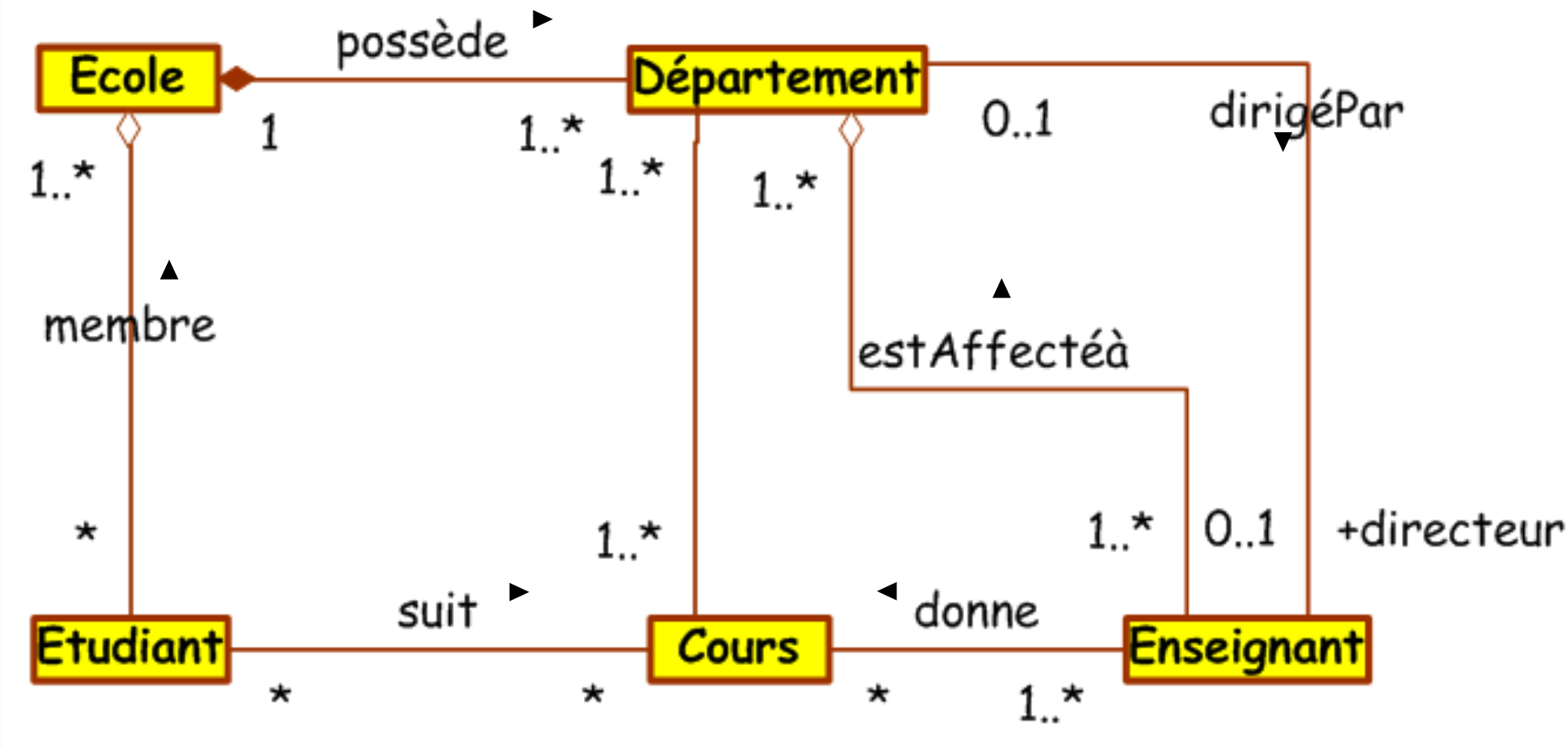


Navigabilité

- étant donnée une association non décorée entre deux classes, on peut naviguer d'un type d'objet vers un autre type d'objet.
- par défaut une association est navigable dans les deux sens.
- une indication de navigabilité suggère en général qu'à partir d'un objet à une extrémité on peut directement et facilement atteindre l'un des objets à l'autre extrémité.
- lors d'une mise en œuvre programmée, ceci peut suggérer par exemple qu'un objet source mémorise une référence directe aux objets cible.

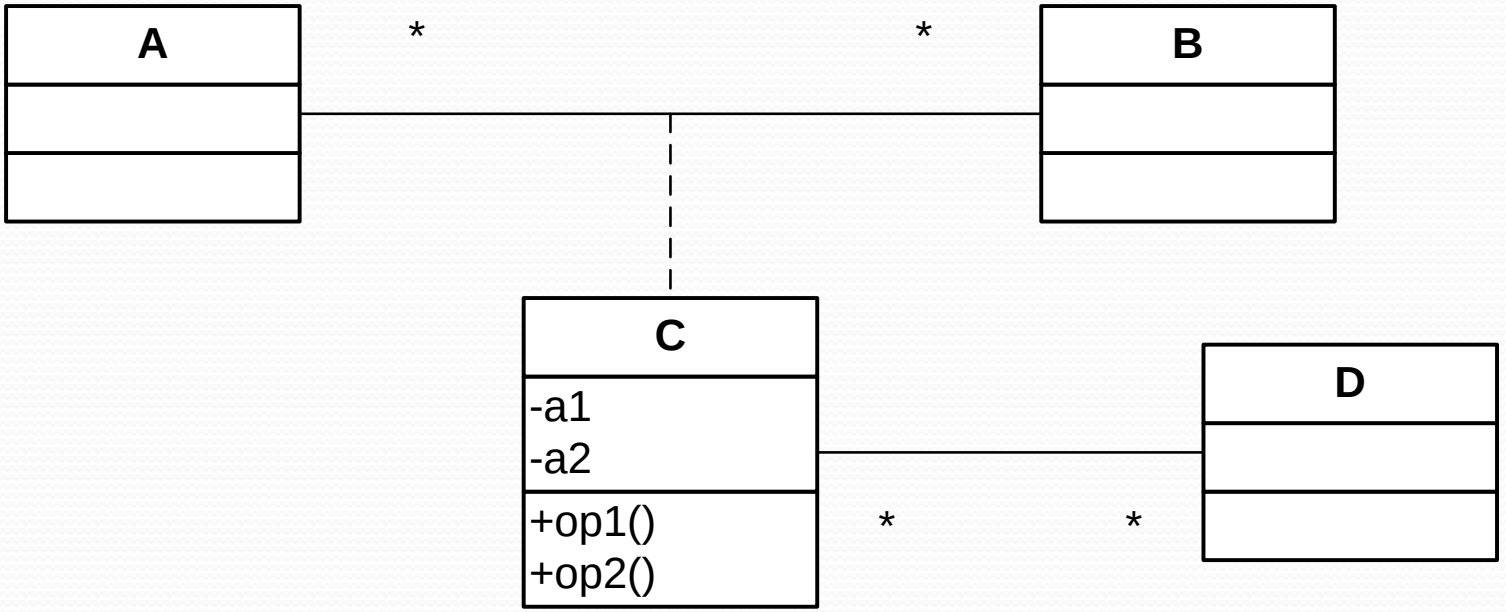
- étant donné un utilisateur, on désire pouvoir accéder à ses mots de passe
- étant donné un mot de passe, on ne souhaite pas pouvoir accéder à l'utilisateur correspondant

Exemple


Ecole
possède
Département
1
1..*
1..*
1..*
0..1
dirigéPar
membre
estAffectéà
*
1..*
1..*
0..1
+directeur
suit
donne
Etudiant
Cours
Enseignant
*
*
*
1..*

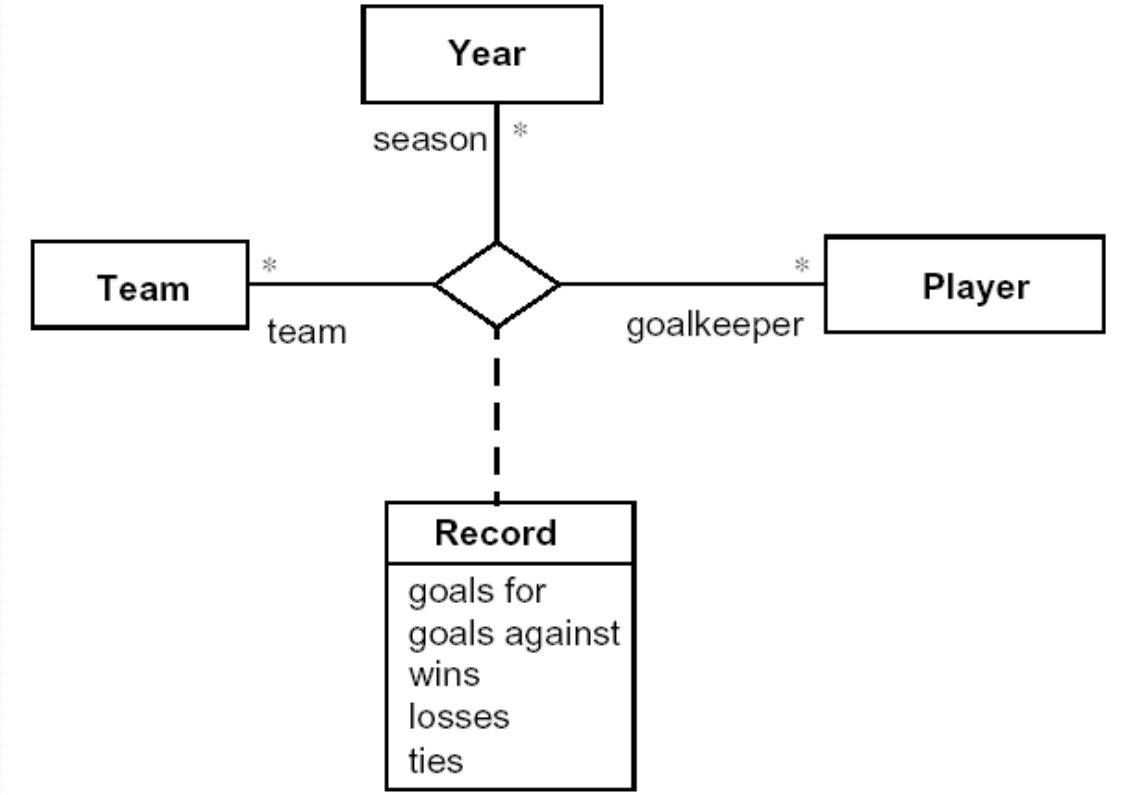
Les classes-associations

- Ajout d'attributs ou d'opérations dans la relation
  - Pour chaque instance de l'association (A,B), il y a une instance de C



Associations ternaires (et plus)

- Pas d'agrégation, pas de qualifier
- Multiplicité plus difficile à lire

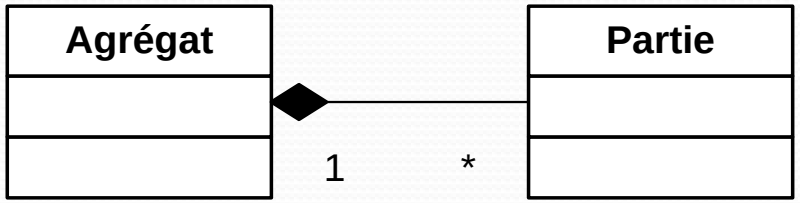


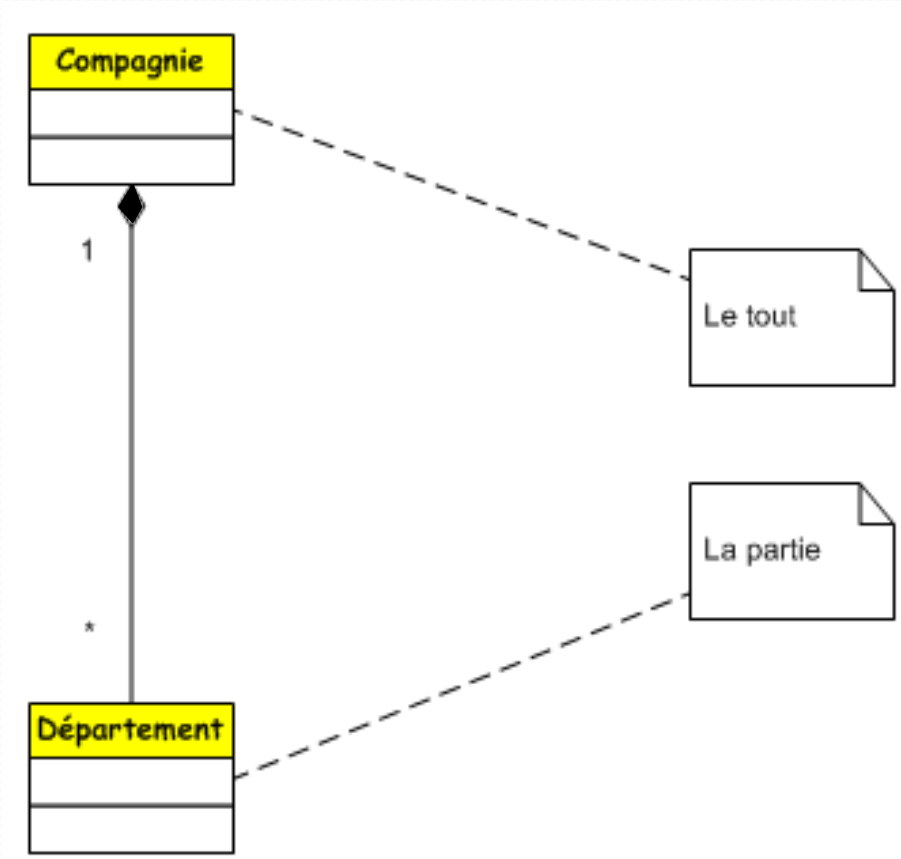
L'agrégation

- (losange blanc)
- Forme d'association qui exprime un couplage plus fort entre classes
- Représentation des relations
  - maître et esclaves
  - tout et parties
  - composé et composant.

La composition

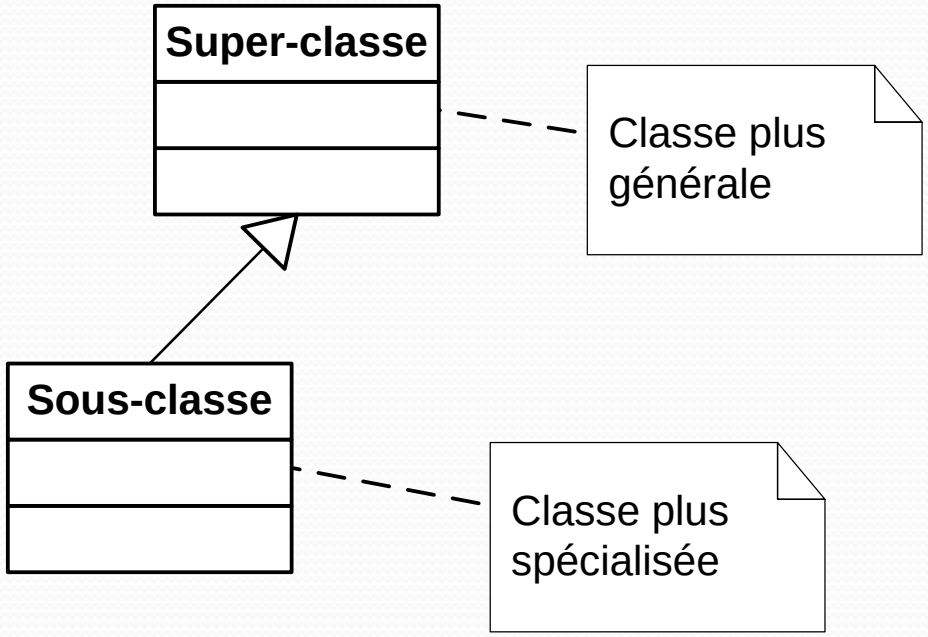
- (losange noir)
- Modélisation de la composition physique
  - Multiplicité au max de 1 du coté de l'agrégat
  - Propagation automatique de la destruction





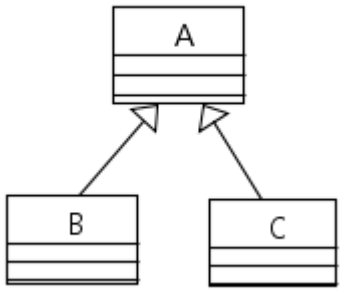
Hiérarchies de classes

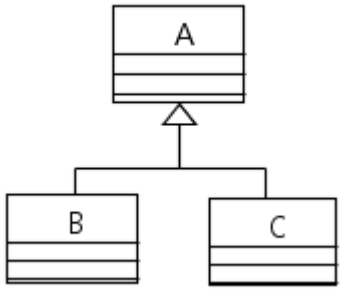
- Gérer la complexité
  - Arborescences de classes d'abstraction croissante



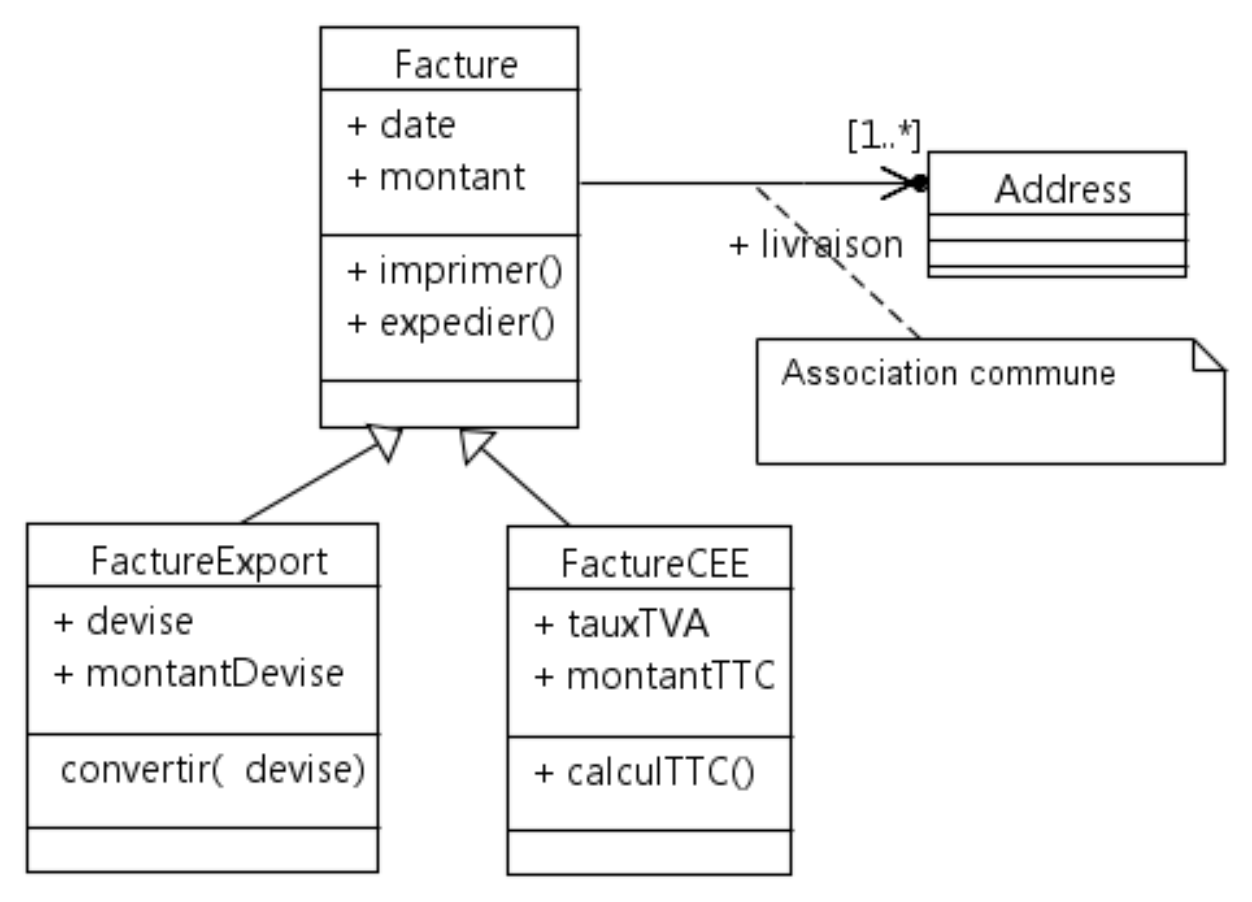
Généralisation / spécialisation

- Deux interprétations
  - niveau conceptuel
    - organisation : un concept est plus général qu'un autre
  - Implémentation
    - héritage des attributs et méthodes
- Pour une bonne classification conceptuelle
  - principe de substitution / conformité à la définition
    - toutes les propriétés de la classe parent doivent être valables pour les classes enfant
  - « A est une sorte de B » (mieux que « A est un B »)
    - toutes les instances de la sous-classe sont des instances de la super-classe (définition ensembliste)
- Spécialisation
  - relation inverse de la généralisation





Hiérarchie de classes

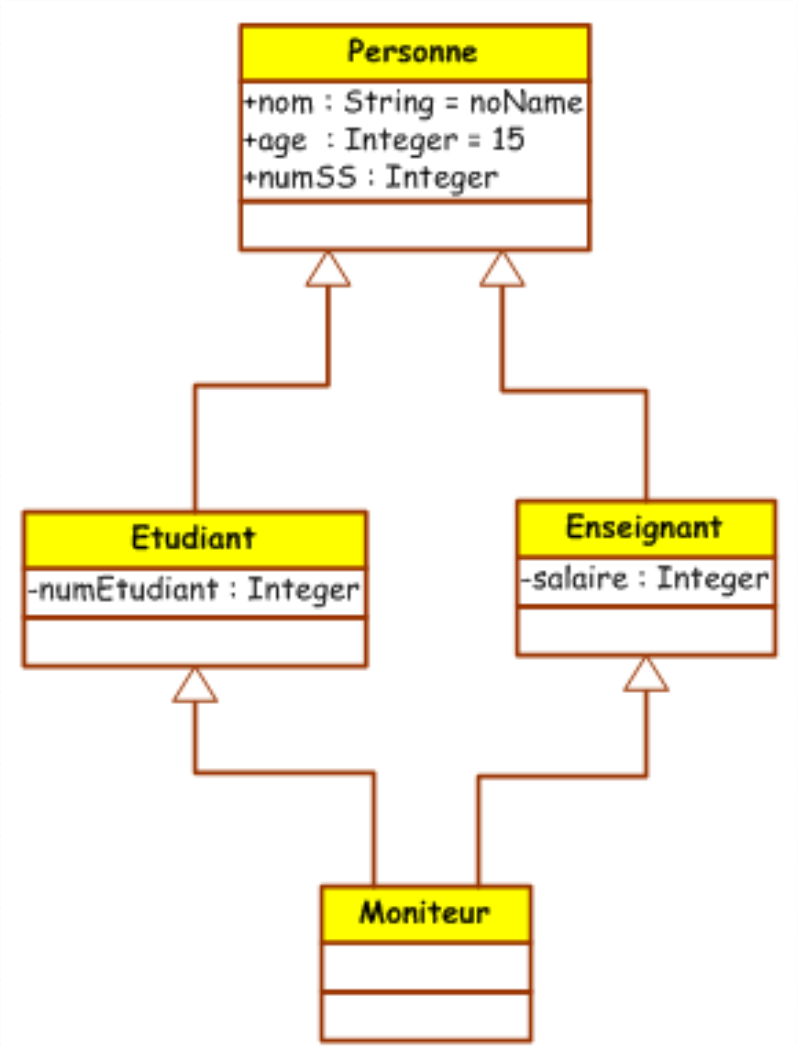


Conseils pour la classification conceptuelle

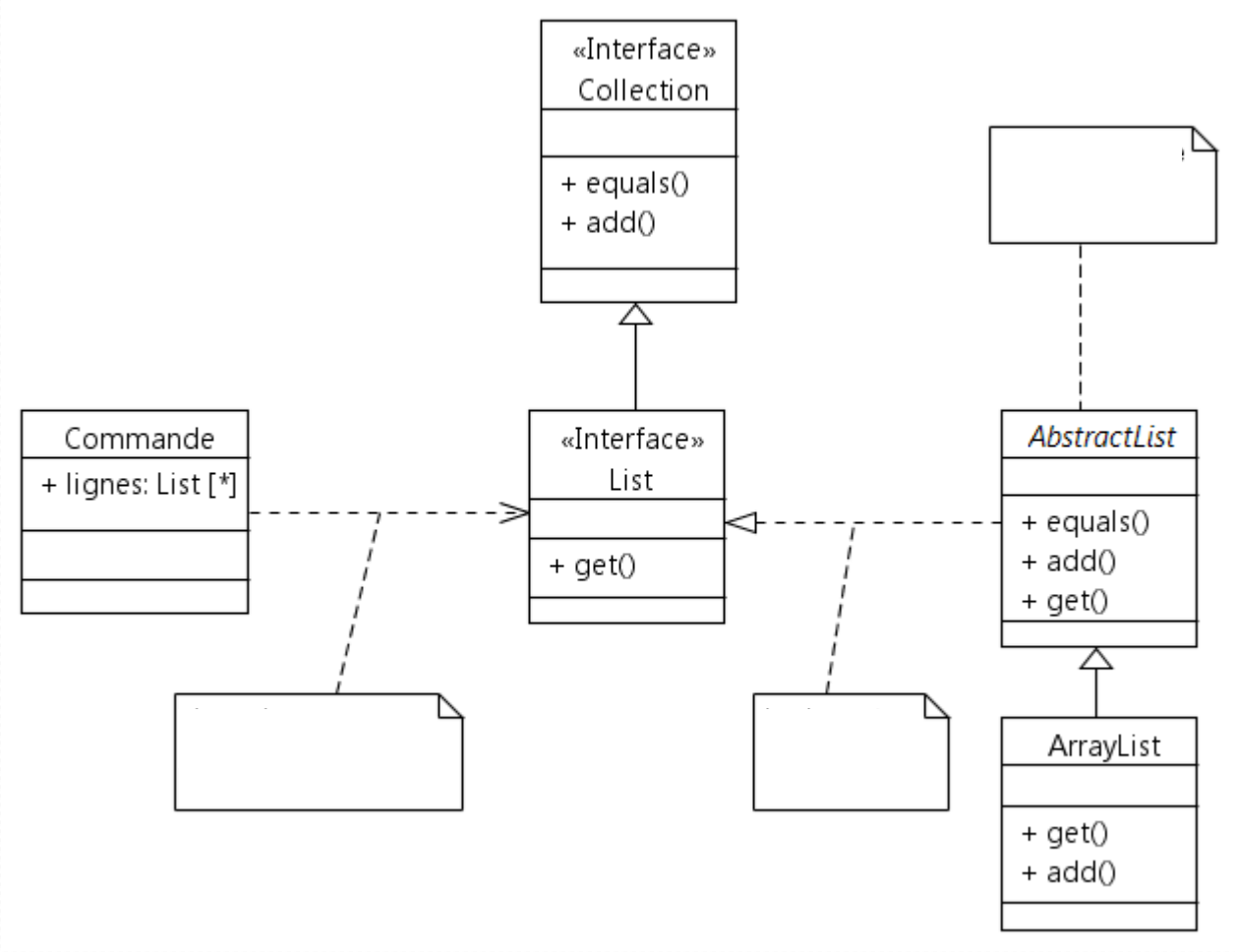
- Partitionner une classe en sous-classes
  - la sous-classe a des attributs et/ou des associations supplémentaires pertinents
  - par rapport à la superclasse ou à d'autres sous-classes, la sous-classe doit être gérée, manipulée, on doit agir sur elle ou elle doit réagir différemment, et cette distinction est pertinente
  - le concept de la sous-classe représente une entité animée (humain, animal, robot) qui a un comportement différent de celui de la superclasse, et cette distinction est pertinente
- Définir une super-classe
  - les sous-classes sont conformes aux principes de substitution et « sorte-de »
  - toutes les sous-classes ont au moins
    - un même attribut
    - et/ou une même association qui peut être extrait et factorisé dans la superclasse

Généralisation multiple

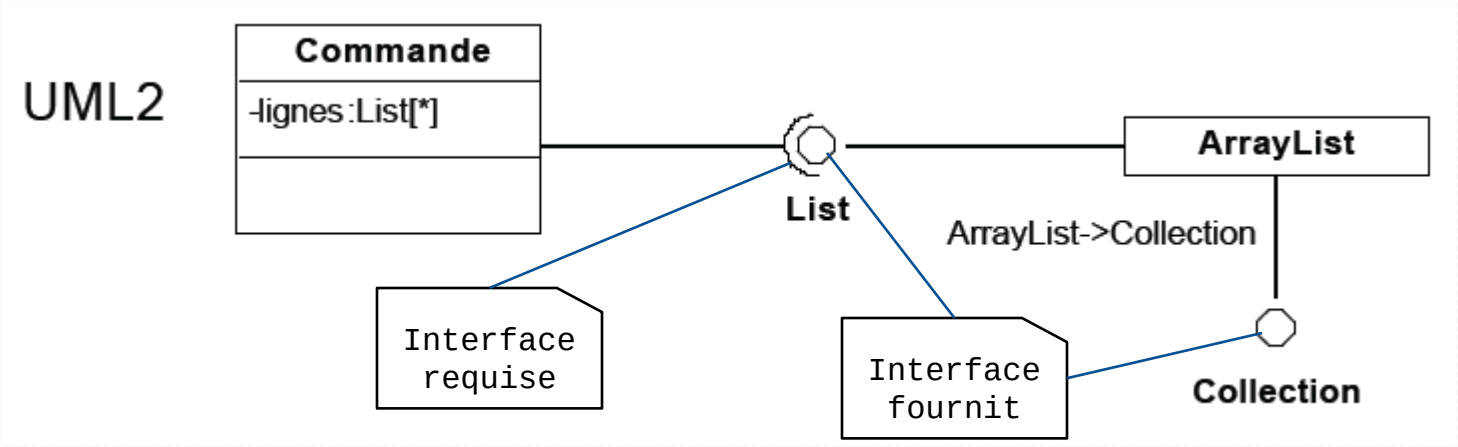
- Autorisée en UML
- Attention aux conflits : il faut les résoudre
- Possibilité d'utiliser aussi délégations ou interfaces

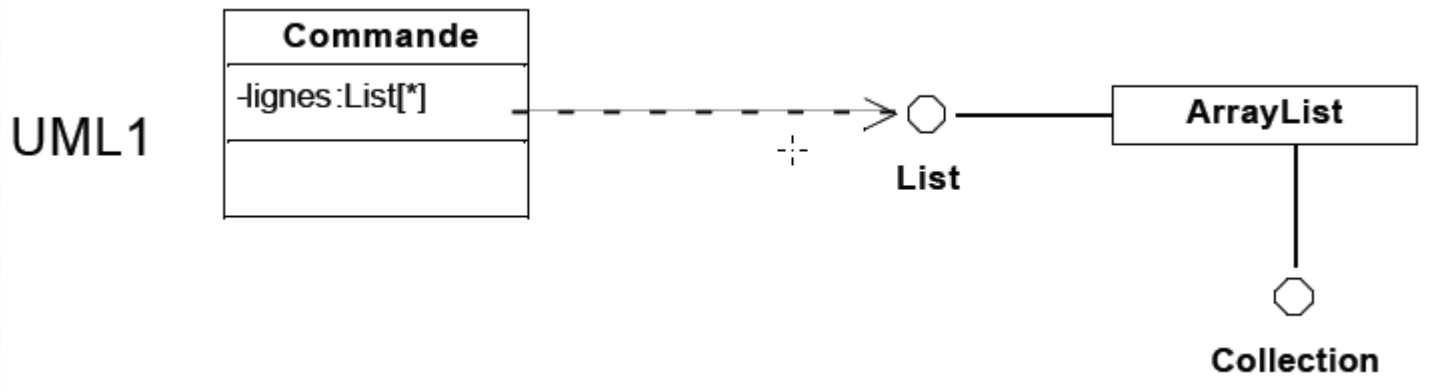


Interfaces et classes abstraites



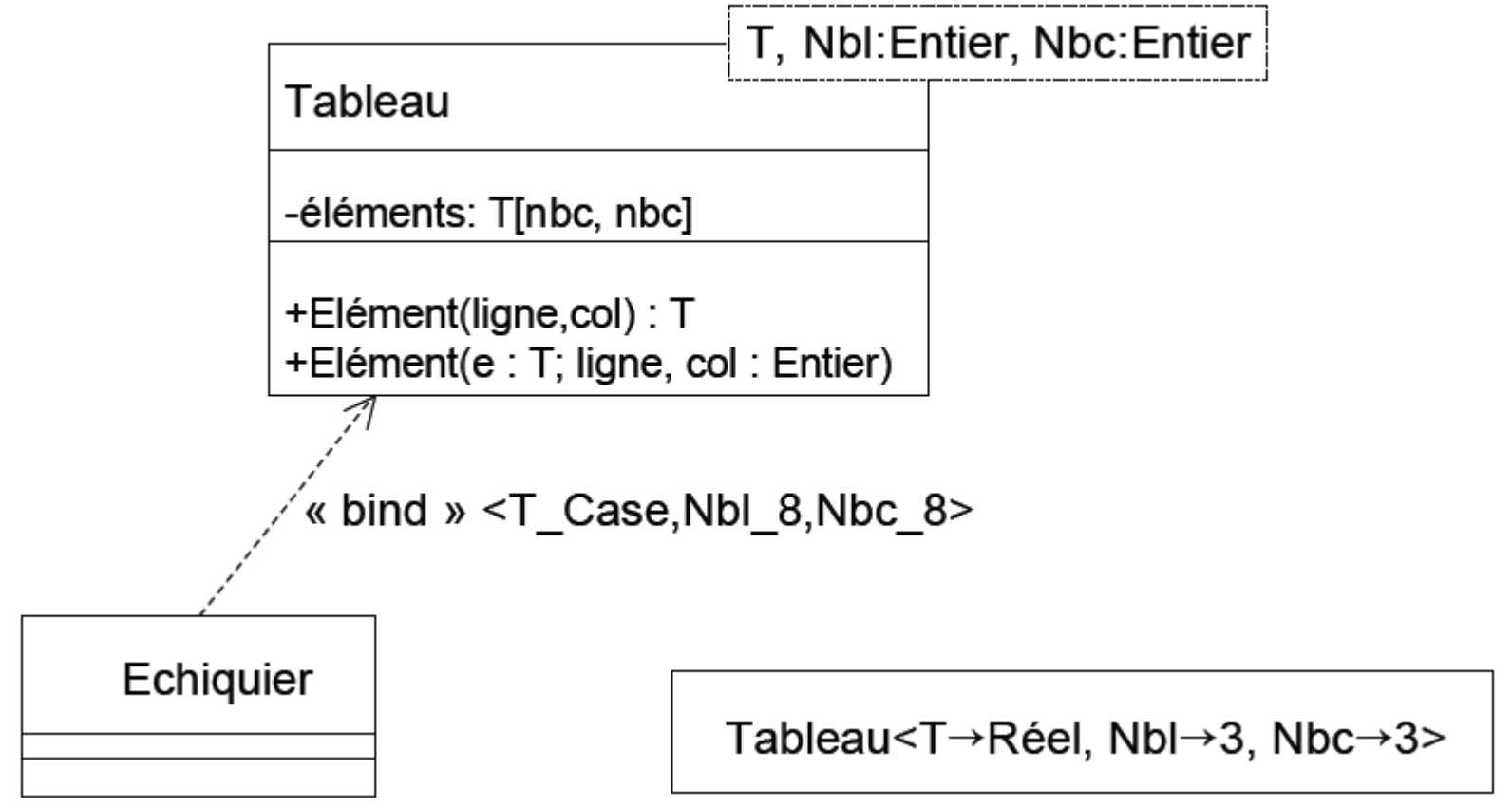
Interface et utilisation : notation





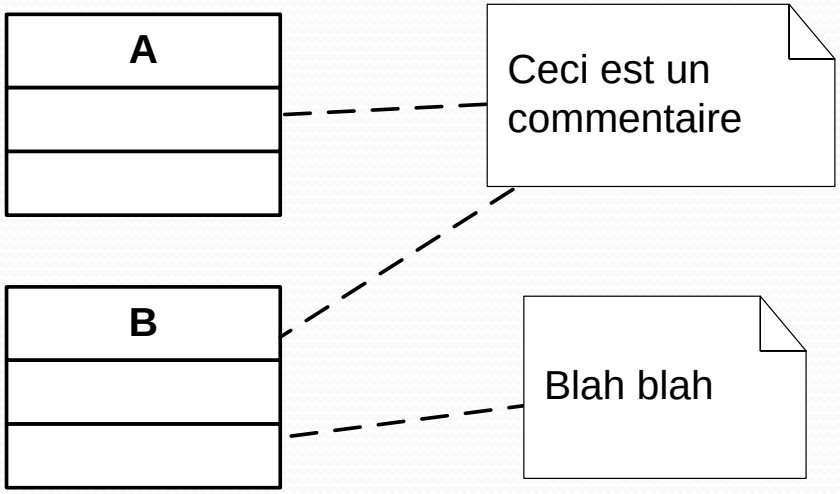
Classes paramétrables (templates)

- Deux notations possibles pour la substitution



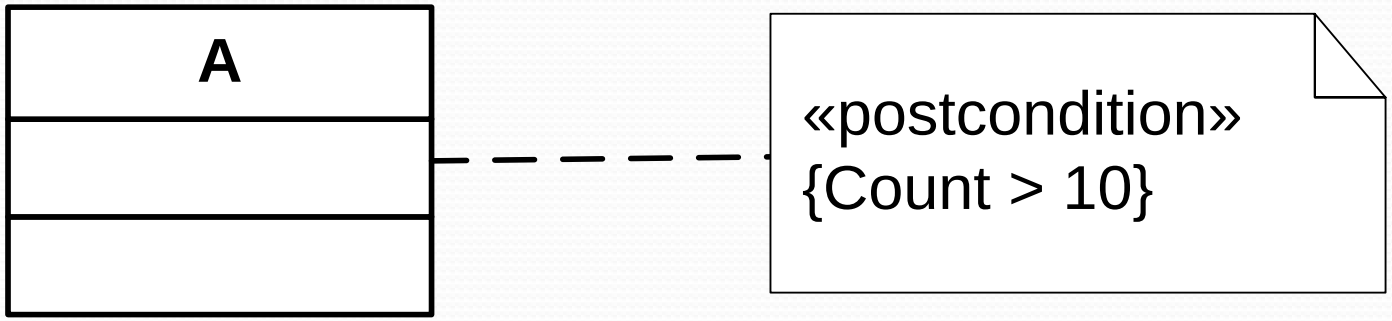
Les notes

- Commentaire attaché à un ou plusieurs éléments de modélisation
  - Appartient à la vue, pas au modèle
  - Peut être stéréotypée en contrainte

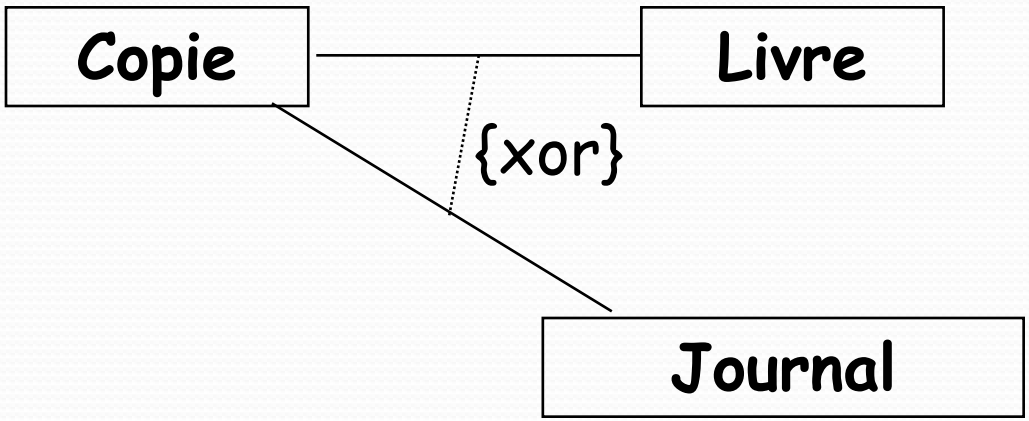


Les contraintes

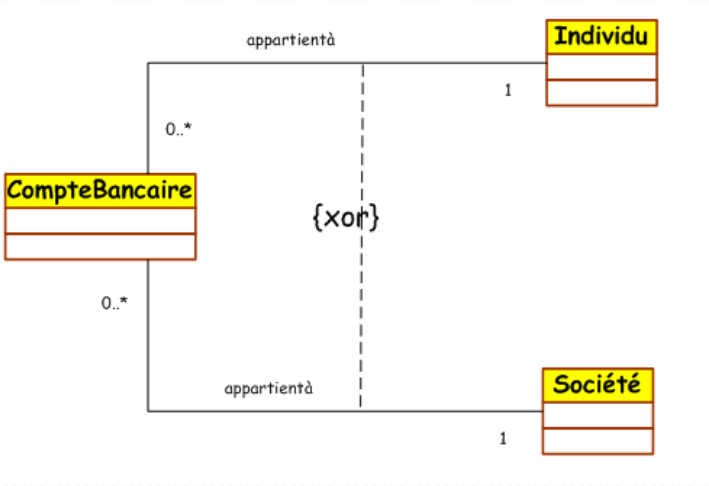
- Relation sémantique quelconque entre éléments de modélisation
- Exprimée en OCL (Object Constraint Language) ou en langage naturel
  - {contrainte}, inv, pre-, post-condition



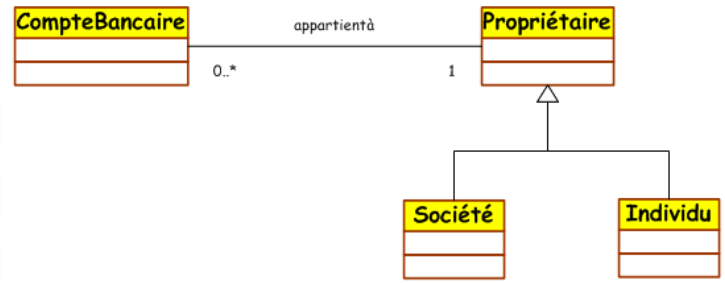
Exemples de contraintes



{self.roues->size <= 4}



Transaction
quantité:Euro {quantité est un multiple de €5}



Conclusion partielle

Ce que vous avez appris

- Ce qu'est un modèle
- Pourquoi fait-on des modèles
- Histoire d'UML
- Diagramme de Cas d'utilisation
- Diagramme de Classes

Pour la prochaine séance

- Lecture obligatoire :
  - UML 2: Initiation, exemples et exercices corrigés
    - Chap. 7 « La structure des éléments de modélisation »
      - A, B, C, D
    - Chap. 10 La modélisation de l'architecture du système
      - A, B, C, D

The End