

R3.03 – Analyse

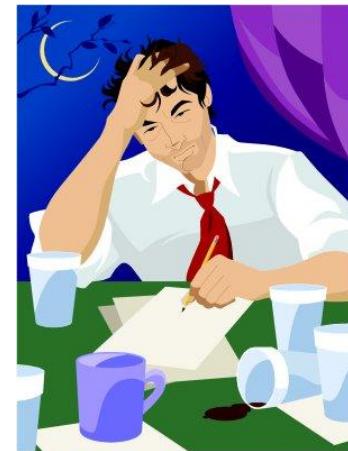
3_PHASE D'ANALYSE

Dalila TAMZALIT

IUT de Nantes – Département Informatique

Analyse

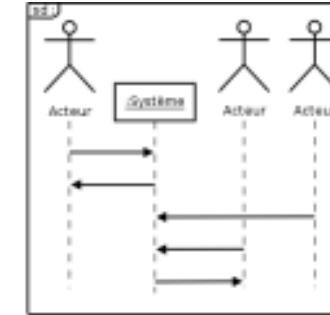
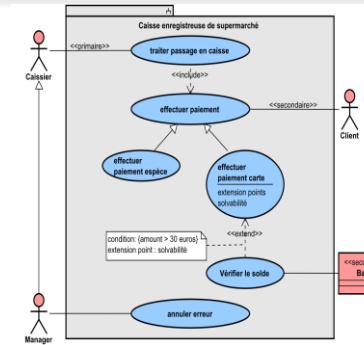
- Produire un **modèle d'analyse** du système qui soit
 - Correct, complet, cohérent et vérifiable
- But
 - Obtenir une compréhension méticuleuse des exigences
 - Décrire les exigences pour produire une conception et une implémentation **maintenables**
- Diffère du recueil des besoins
 - Structurer et formaliser les exigences
 - (Pas nécessairement compréhensible par le client)
- Force le client et les développeurs à prendre les **décisions** difficiles le plus tôt possible



Modèles d'analyse

• Modèle fonctionnel

= Fonctionnalités du système
 → Cas d'utilisations, scénarios

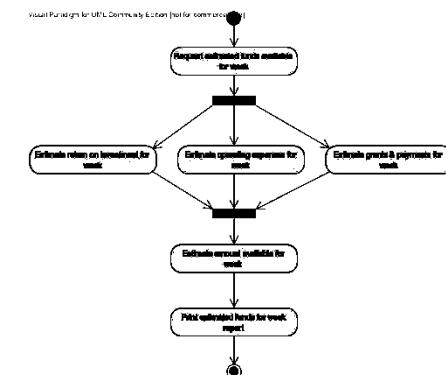
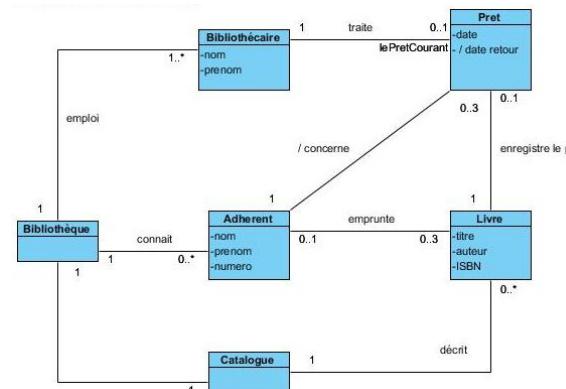


• Modèle dynamique

= Comportement du système
 → Activités, flux de données

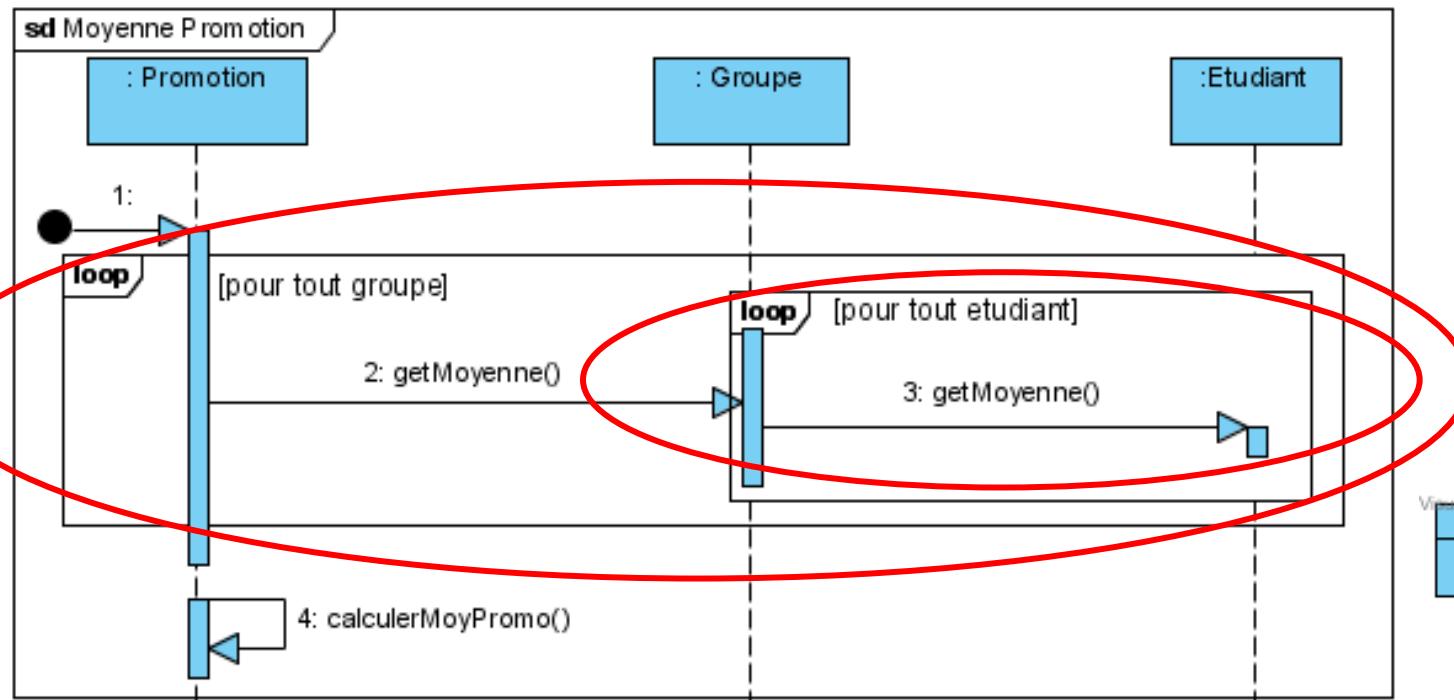
• Modèle de classes

= Concepts individuels manipulés par le système et leurs propriétés
 → Classes, composants

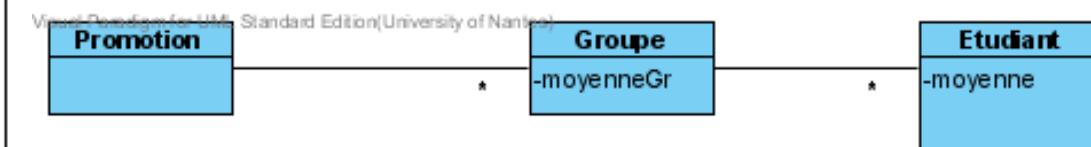


Structures de contrôle dans les DS

- L'itération d'un envoi de message : loop



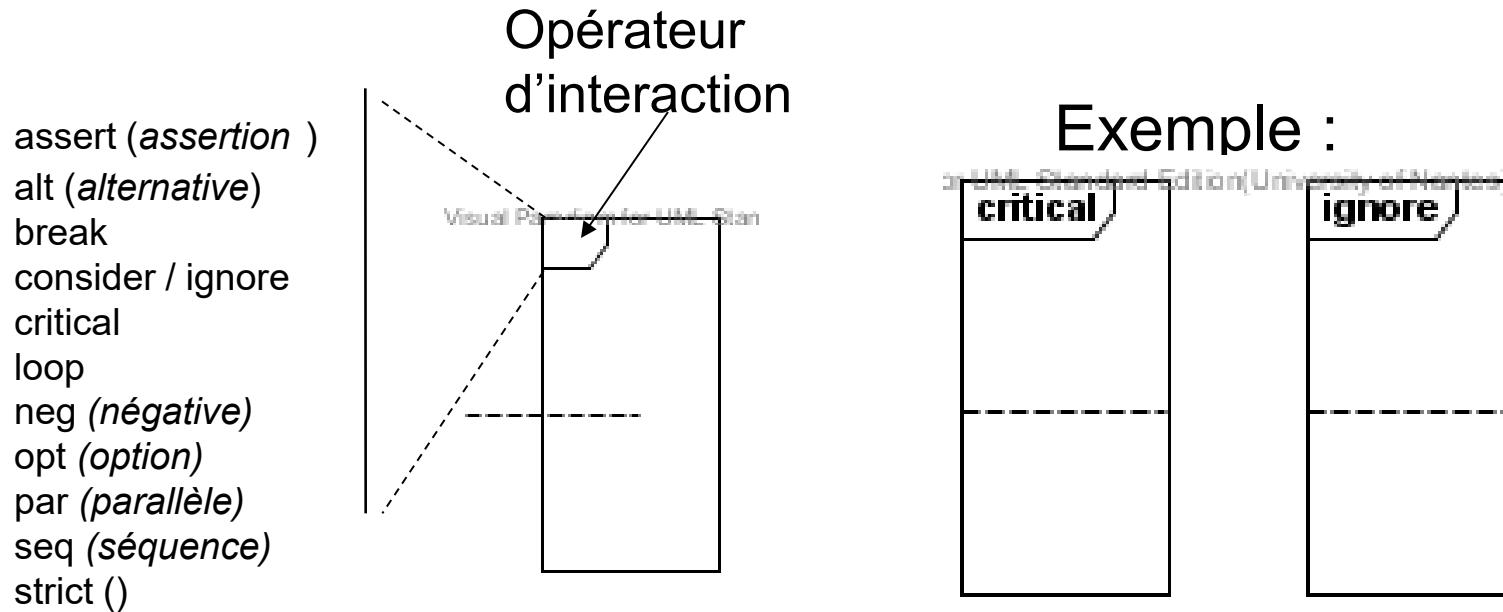
- Envoi de message s'appuie sur une association /dépendance



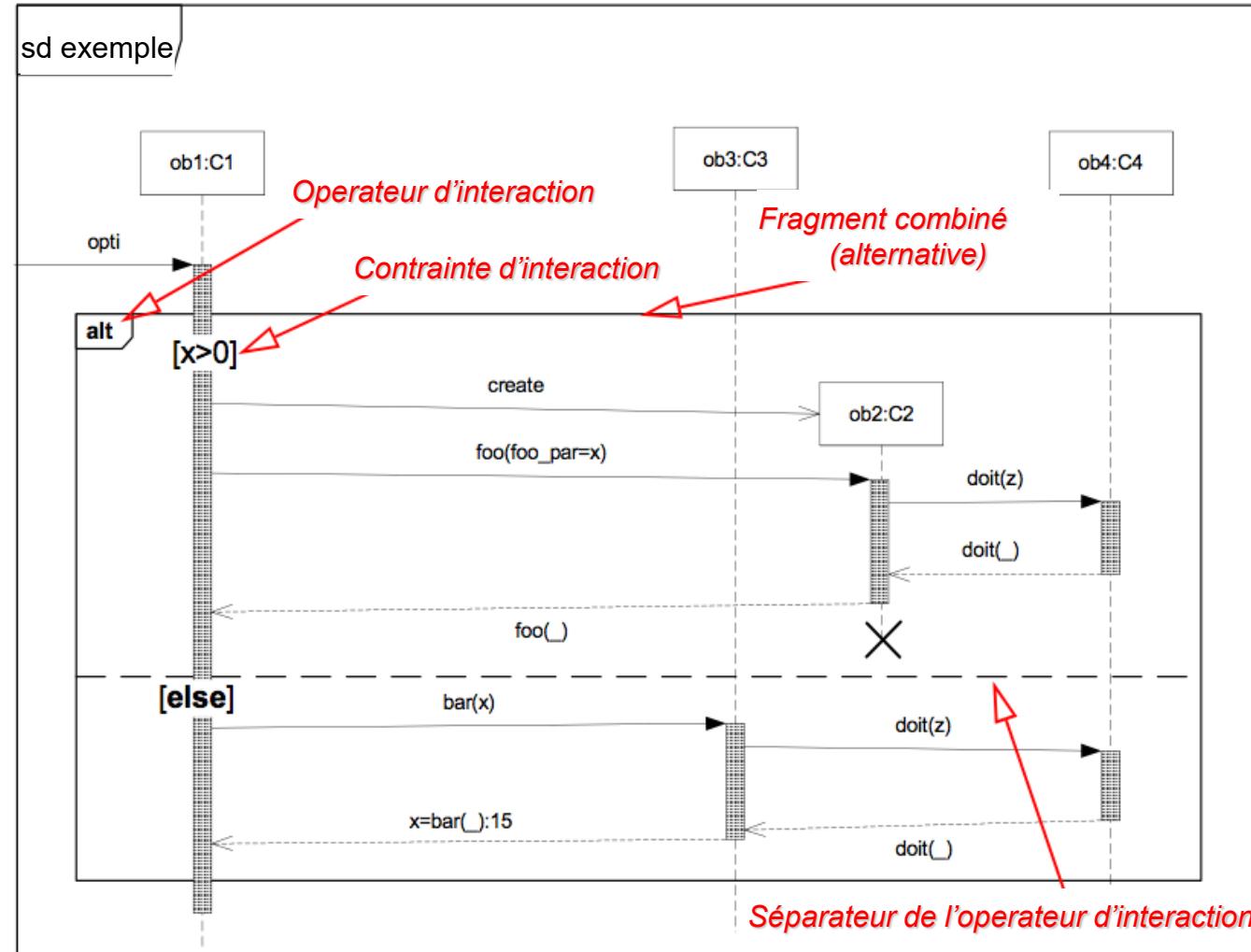
Utilisation de *fragments combinés*

Fragments combinés

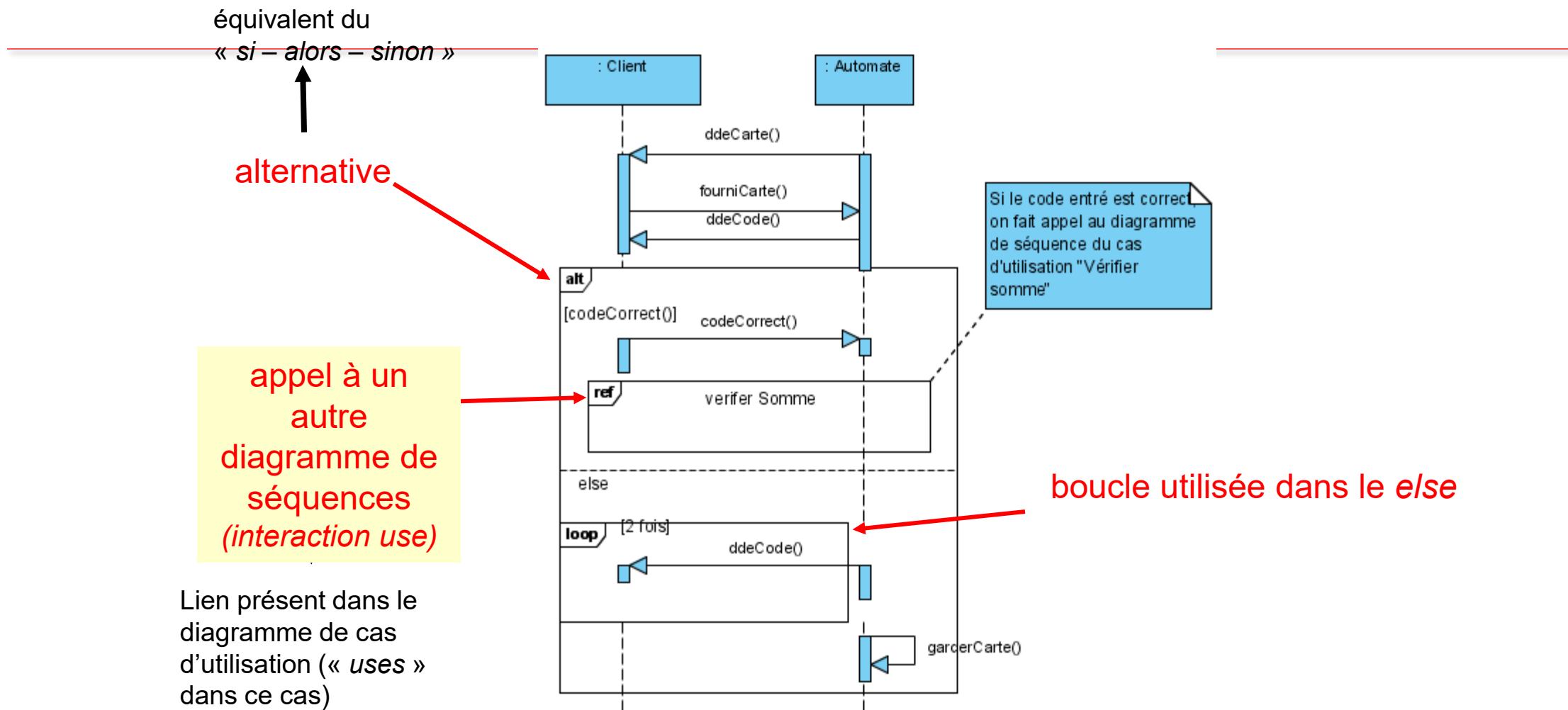
- Articulation d'interactions
- Description compacte des diagrammes de séquences
- Représentation de l'ensemble des entités participantes ou une partie



Exemple



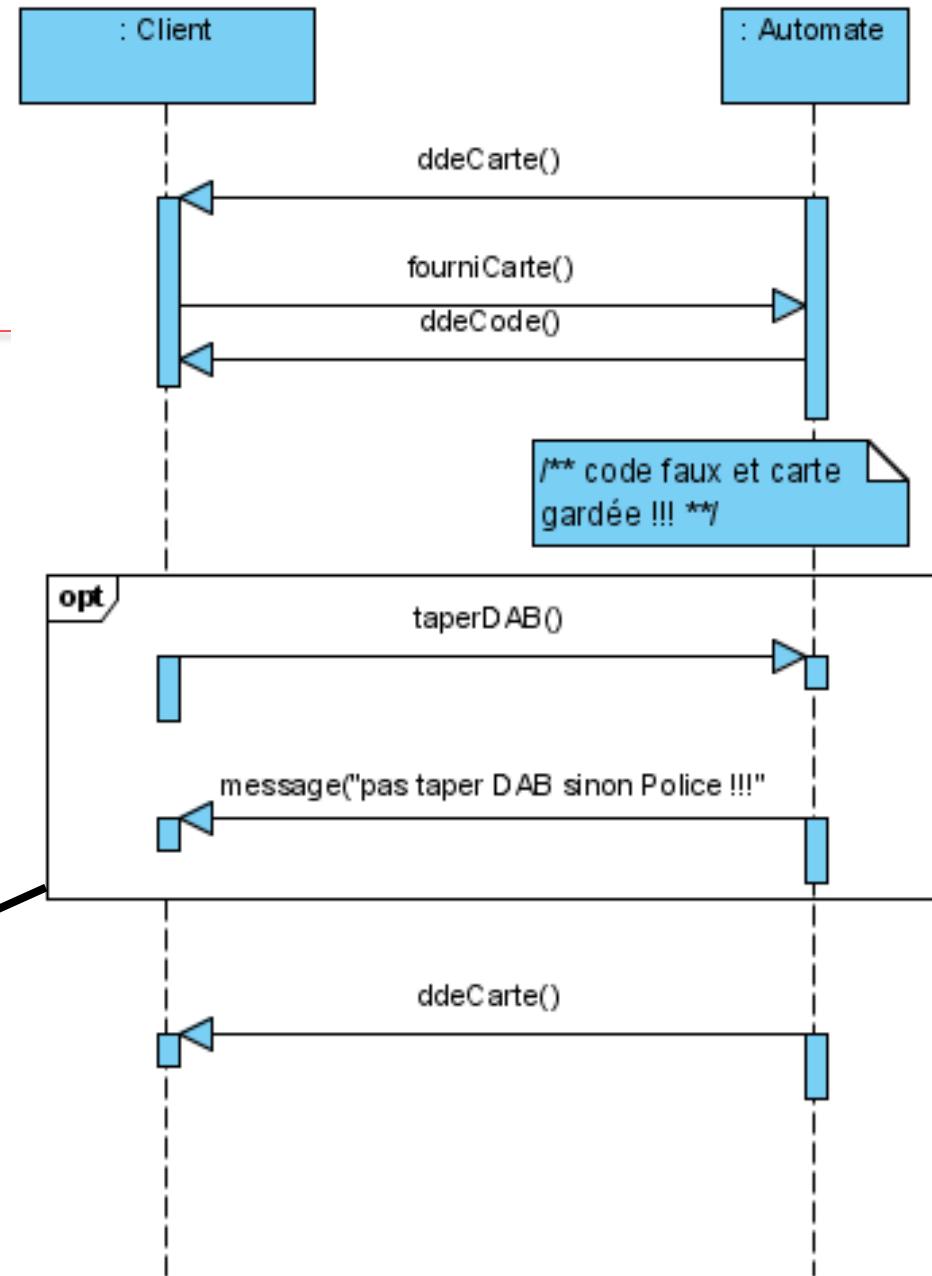
Fragment combiné alt – ref – loop



Fragments combinés opt

- un comportement optionnel :
- Rares sont les utilisateurs mécontents qui se défoulement sur le DAB ... mais il peut y en avoir !
... Que faire dans ce cas ?

équivalent du
« si – alors »

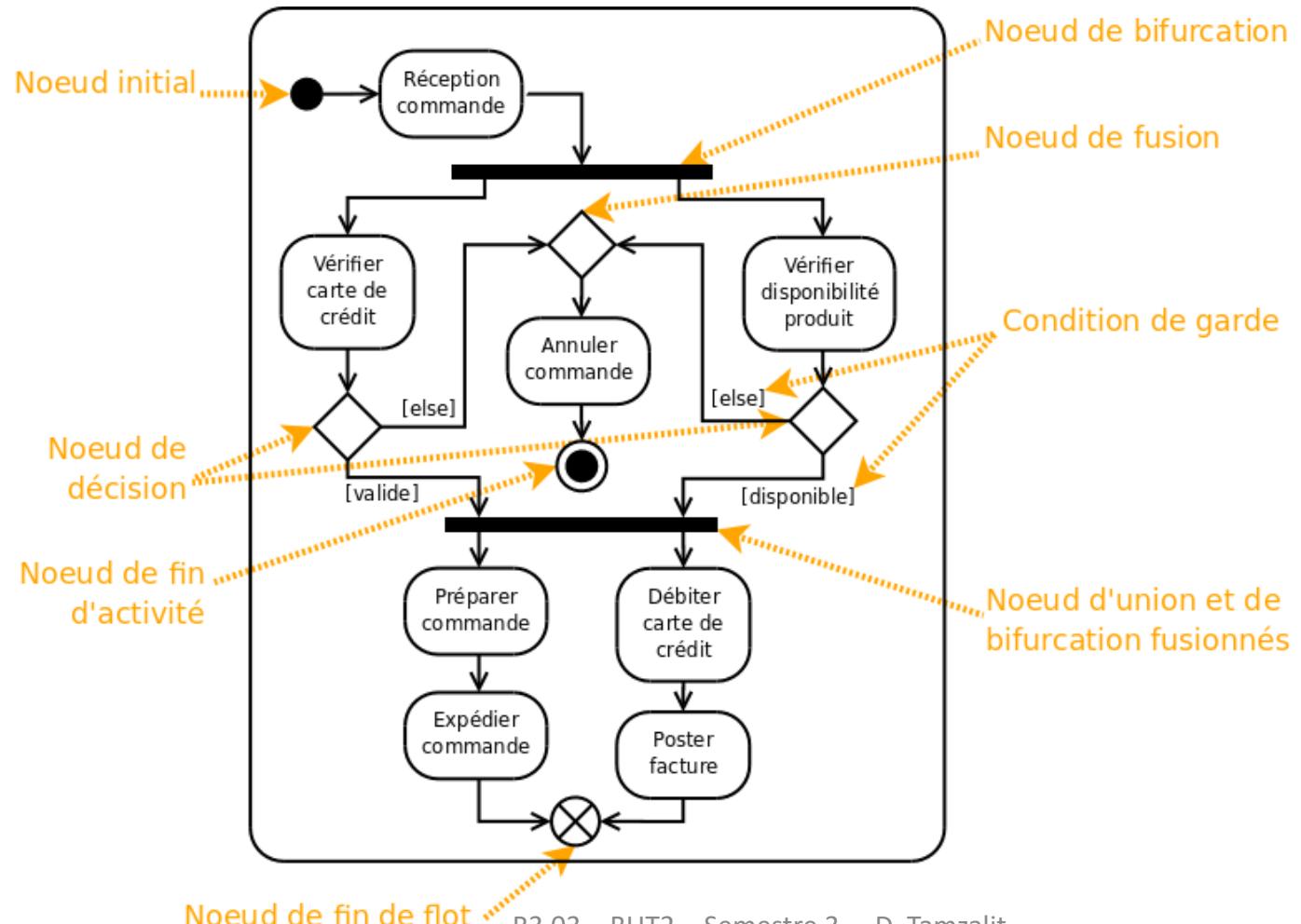


Modèle dynamique : modélisation du processus

- Identifier les activités nécessaires pour utiliser le logiciel
 - Définir les **étapes** du processus
 - Coordonner les différents **événements**
 - Identifier les responsabilités par **rôle**
- Pour cela on utilise un **diagramme d'activité UML**
 - Activités, actions
 - Transitions
 - Objets
 - Nœuds de contrôle
 - Partitions



Diagramme d'activité UML



Action

noeud d'action

- Plus petit **traitement** en UML qui a une incidence sur l'état du système
 - Affectation d'attributs
 - Accès à une propriété structurelle
 - Création d'objet
 - Calcul simple
 - Appel d'opération ou d'événement
 - Acceptation d'opération ou d'événement
 - Levée d'exception
- Activité est un comportement **complexe**
 - Terme **abstrait** représentant un séquencement d'actions

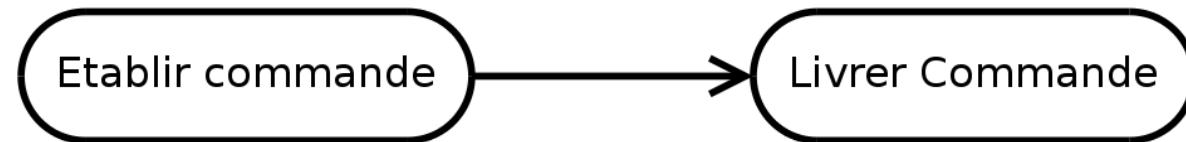
noeud d'action

noeud d'objet



Transition

- Passage d'une activité à une autre
 - Déclenchée quand l'activité source est terminée
 - Provoque le début de la prochaine activité cible
- Contrairement aux activités, les transitions sont **atomiques**
 - Les activités ont une durée donc peuvent être interrompues (pas les transitions !)



Événements

- **Événements externes**
 - Survient à l'extérieur du système (par un acteur)
 - Client passe une commande
- **Événements temporels**
 - Attente dans le temps
 - À chaque semaine
- **Événements d'état**
 - Survient à l'interne et déclenche un besoin de traitement
 - Rupture de stock

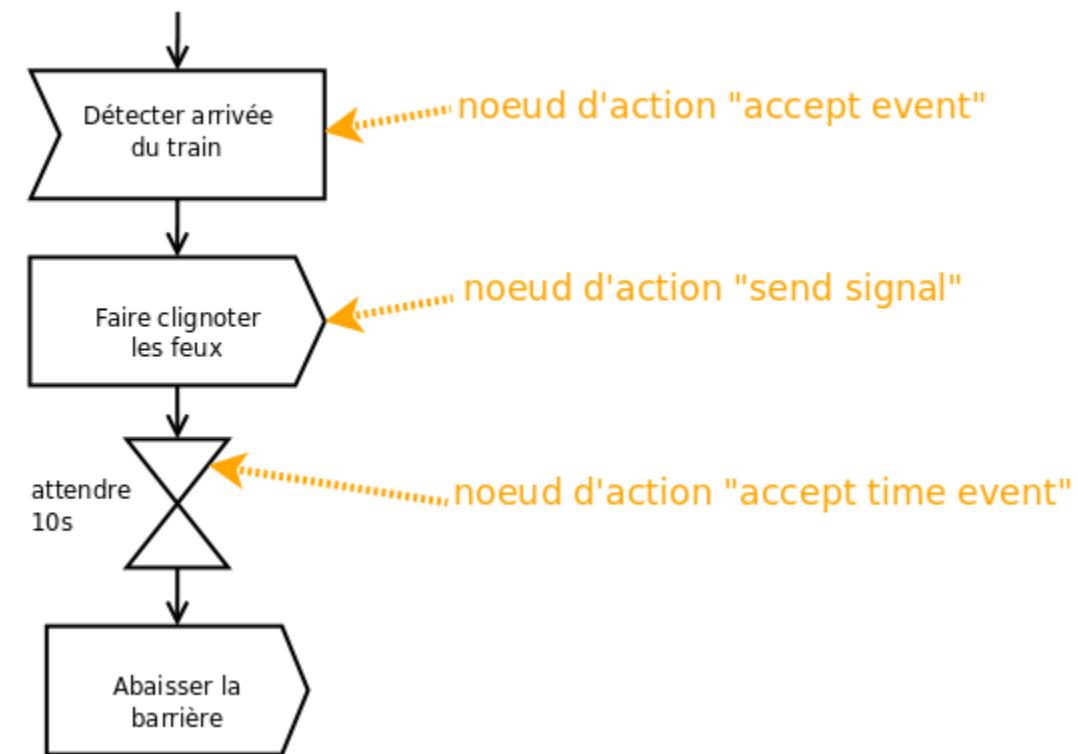
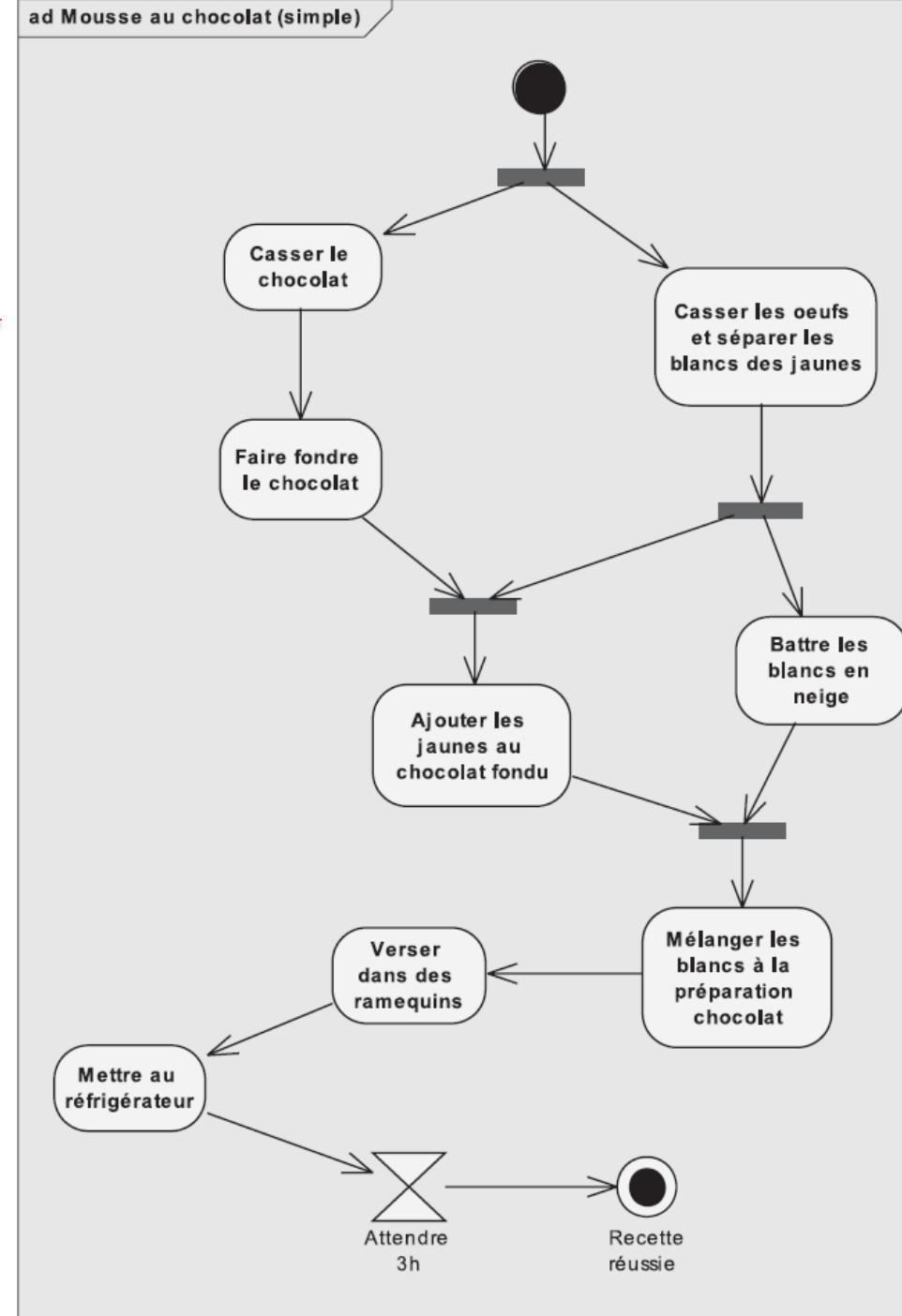
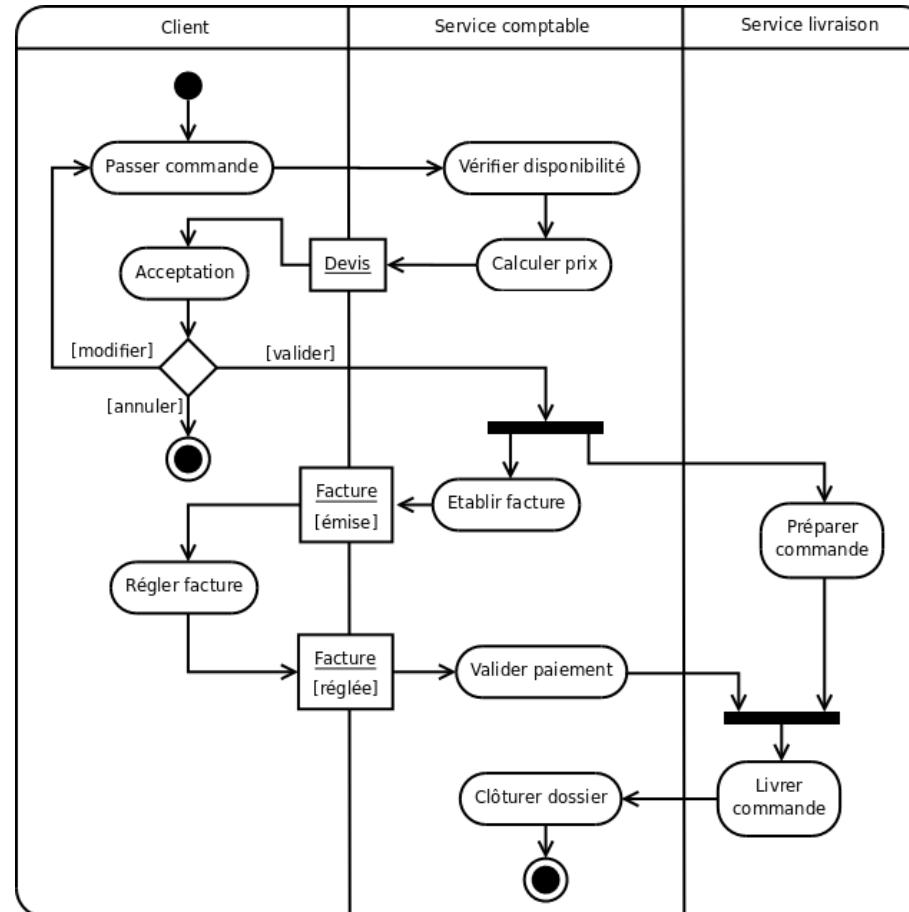


Diagramme d'activités

- Comment faire une mousse au chocolat ?
 - Recette simplifiée : commencer par casser le chocolat en morceaux, puis le faire fondre.
 - En parallèle, casser les œufs en séparant les blancs des jaunes.
 - Quand le chocolat est fondu, ajouter les jaunes d'oeuf.
 - Battre les blancs en neige jusqu'à ce qu'ils soient bien fermes.
 - Les incorporer délicatement à la préparation chocolat sans les briser.
 - Verser dans des ramequins individuels.
 - Mettre au frais au moins 3 heures au réfrigérateur avant de servir



Avec partitions et rôles



Utilisation des diagrammes d'activité

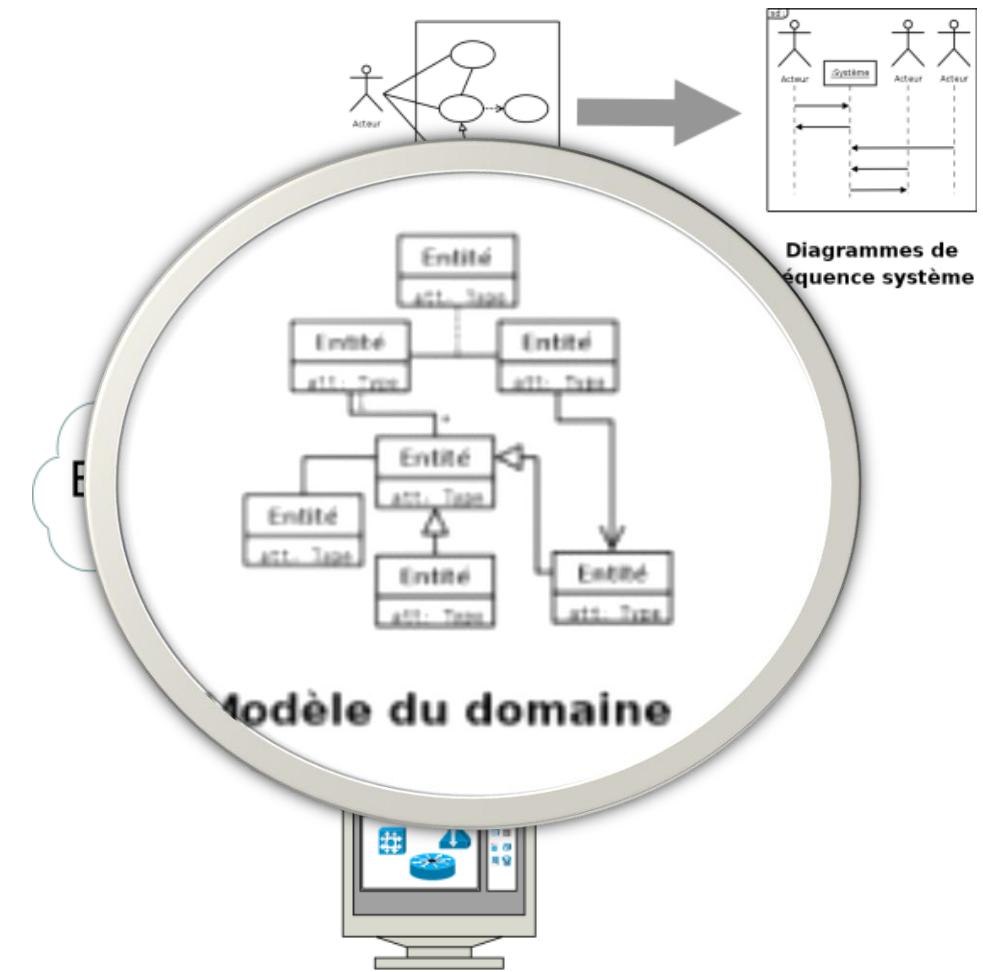
- Met l'accent sur les traitements
 - Flots de **contrôle** et de **données**
- Illustre et consolide description textuelle des CU
 - Modélisation du **workflow** de chaque scenario
 - Concentre sur les activités **vues par les acteurs**

Modèle des classes du domaine

Identifier les objets (et classes) est la tâche la plus difficile de la **conception orientée objet**

Modèles en phase d'analyse

- La modélisation des besoins par des cas d'utilisation revient à une analyse fonctionnelle classique.
- La phase d'analyse du domaine permet d'élaborer la première version du diagramme de classes.
- L'élaboration du modèle des classes du domaine permet d'opérer une transition vers une véritable modélisation objet.

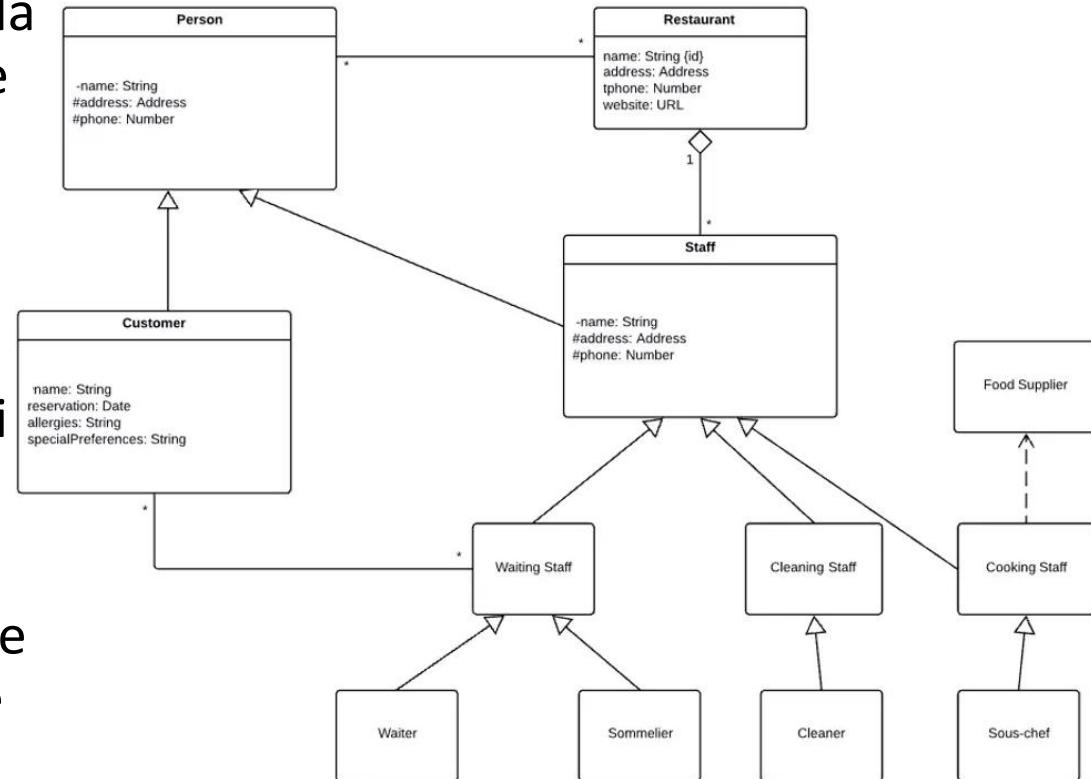


Approches pour l'identification

- Analyse basée sur les **scénarios**
 - Identifier les objets, leurs attributs et méthodes par scénario
- Approche **grammaticale**
 - Substantifs et verbes
- Baser l'identification sur les choses **tangibles** du domaine
 - Analyse du domaine
 - Structures de données qui leur sont appropriées
- Approche **comportementale**
 - Identifier les objets selon ce qui participe à chaque comportement du système

Modèle des classes du domaine

- La phase d'analyse du domaine permet d'élaborer la première version du diagramme de classes appelée *modèle du domaine* ou *diagramme de classes d'analyse* ou *diagramme de classes de conception générale*.
- Il définit les classes qui modélisent les entités ou concepts présents dans le domaine (on utilise aussi le terme de métier) de l'application.
- Ces entités ou concepts peuvent être identifiés directement à partir de la connaissance du domaine ou par des entretiens avec des experts du domaine (confère l'ingénierie des exigences).

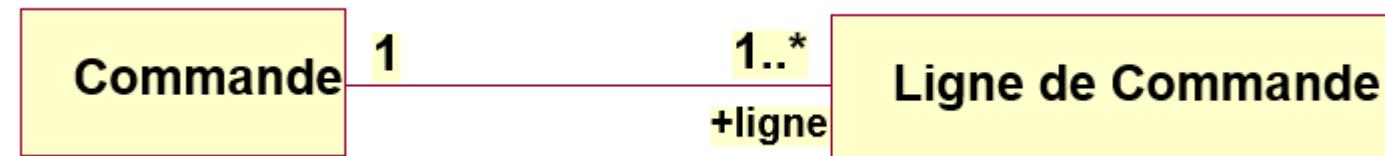


Modèle des classes du domaine

- Il faut absolument utiliser le vocabulaire du métier pour nommer les classes et leurs attributs.
- Les classes du modèle du domaine ne doivent pas contenir d'opération, mais seulement des attributs.
- Les étapes à suivre pour établir ce diagramme sont :
 - identifier les entités ou concepts du domaine ;
 - identifier et ajouter les associations et les attributs ;
 - organiser et simplifier le modèle en éliminant les classes redondantes et en utilisant l'héritage ;
 - le cas échéant, structurer les classes en paquetage selon les principes de cohérence et d'indépendance.

Concept de rôle

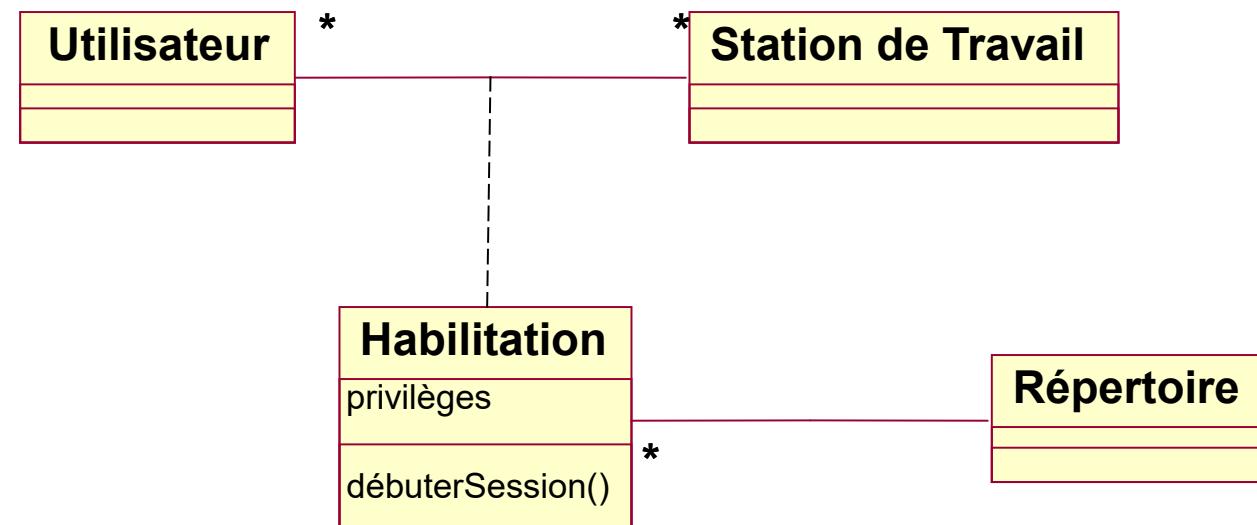
- Rôles : rôle que joue une classe dans une association.
- Aux deux extrémités d'une association.
- Nom de rôle : le rôle est nommé explicitement.
- Exemple : ligne de l'extrémité Ligne de Commande associée à Commande.



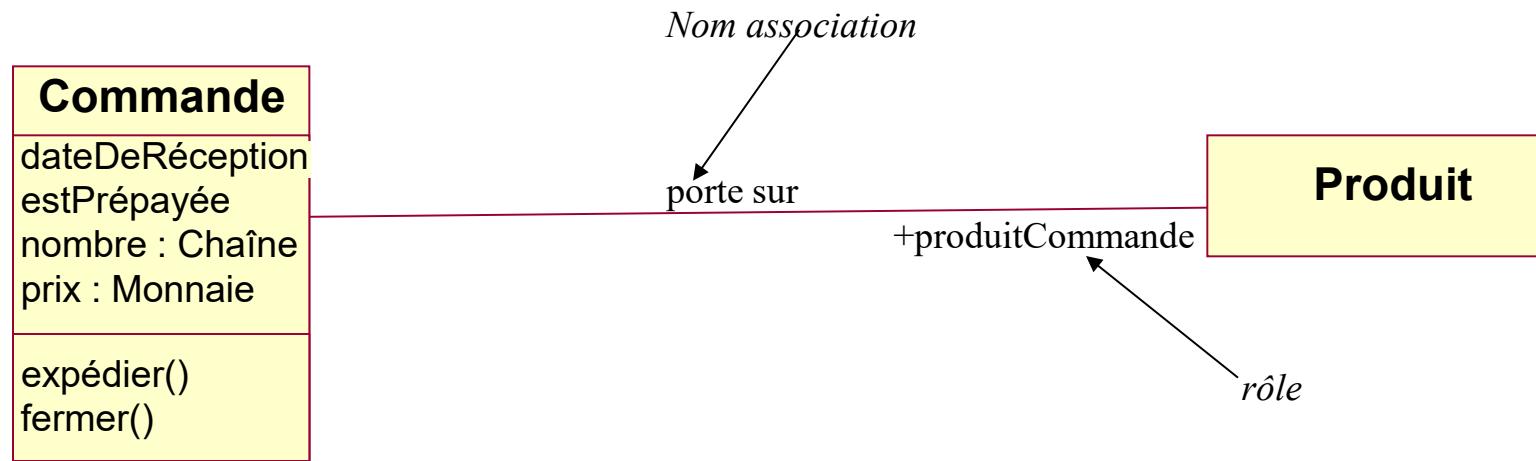
- En l'absence d'étiquette, le nom de rôle est le nom de la classe cible.

Classe-association

- Une association peut être représentée par une classe si elle est porteuse d'attributs/opérations.
- C'est une classe comme les autres. Elle peut avoir des relations.



Associations et nom de rôle



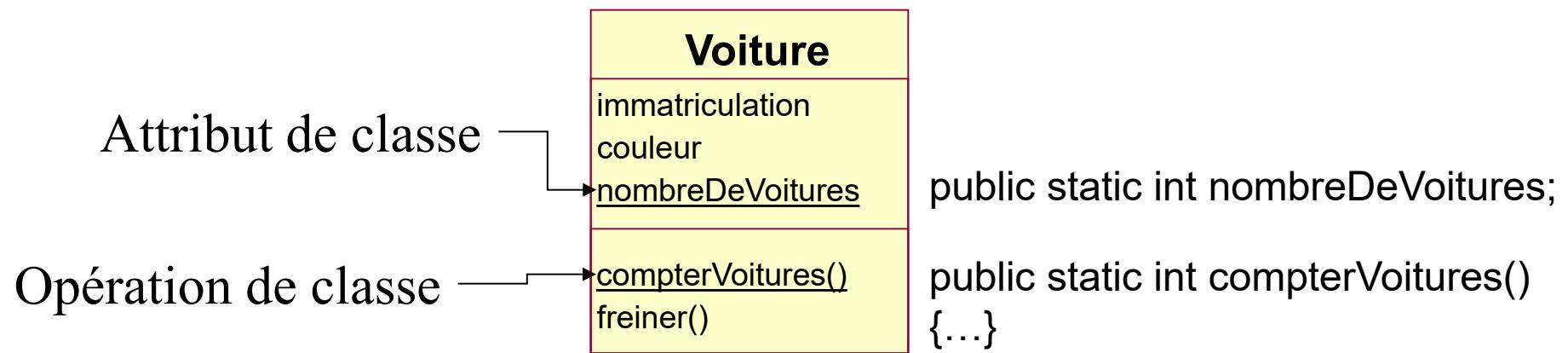
Associations

- Navigabilité :



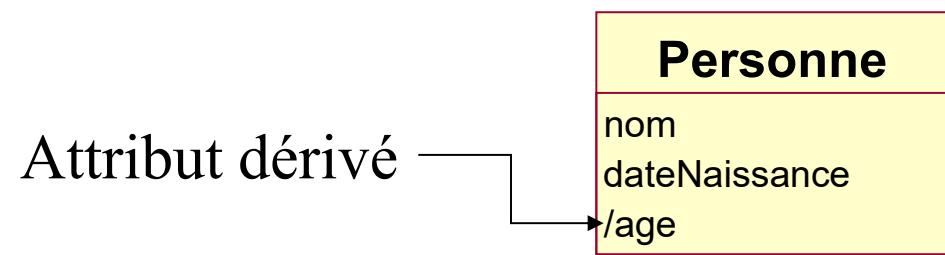
- Association unidirectionnelle.
 - Une Commande peut informer de ses Clients ; mais le client ne peut pas indiquer ses commandes.
 - Importance pour l'implémentation et non pour l'aspect purement conceptuel.
 - Commande aura un pointeur sur Client mais Client n'aura pas de pointeurs sur Commande.
- Association sans navigabilité :
 - Bidirectionnelle ou Inconnue.
 - Les navigations sont l'inverse l'une de l'autre.

Attributs/opérations de classe



Attributs dérivés

- Attribut dérivé: attribut qui se calcule

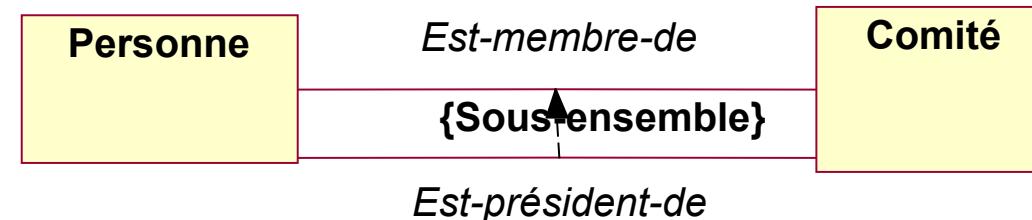


Contraintes

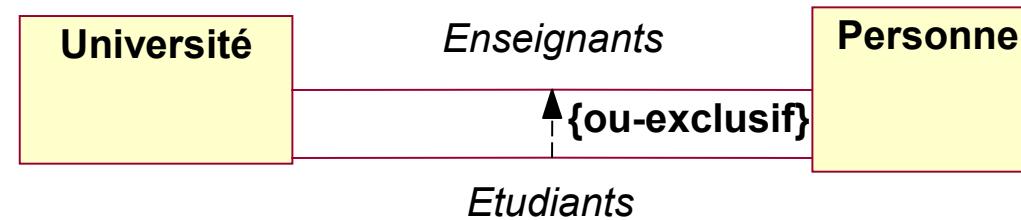
- Associations, attributs et généralisation :
 - Expriment les contraintes importantes
 - Ne peuvent pas toutes les indiquer. Il faut cependant les capturer. UML fait le choix de les exprimer au niveau du diagramme de classes.
- Contraintes :
 - Exprimées entre accolades.
 - En langage naturel ou en OCL (Object Constraint Language).
 - Généralement implémentées sous forme d'assertions dans le langage de programmation.

Quelques contraintes

- Contrainte de sous-ensemble : {sous-ensemble}
 - Une collection est incluse dans une autre collection.



- Contrainte du ou-exclusif : {ou-exclusif}
 - Pour un objet, une seule association valide parmi un groupe d'associations.



- Évite l'introduction de sous-classes artificielles pour représenter l'exclusivité.

Généralisation

- Généralisation :
 - Relation non-symétrique.
 - Relation entre une classe et une ou plusieurs de ses versions plus raffinées ou plus spécialisées.
 - Relation de classification entre un élément plus général et un élément plus spécifique.
- Héritage :
 - Mécanisme objet qui permet aux classes de partager des attributs et des opérations en se basant sur une relation, habituellement la généralisation.

Super-classe et Sous-classe

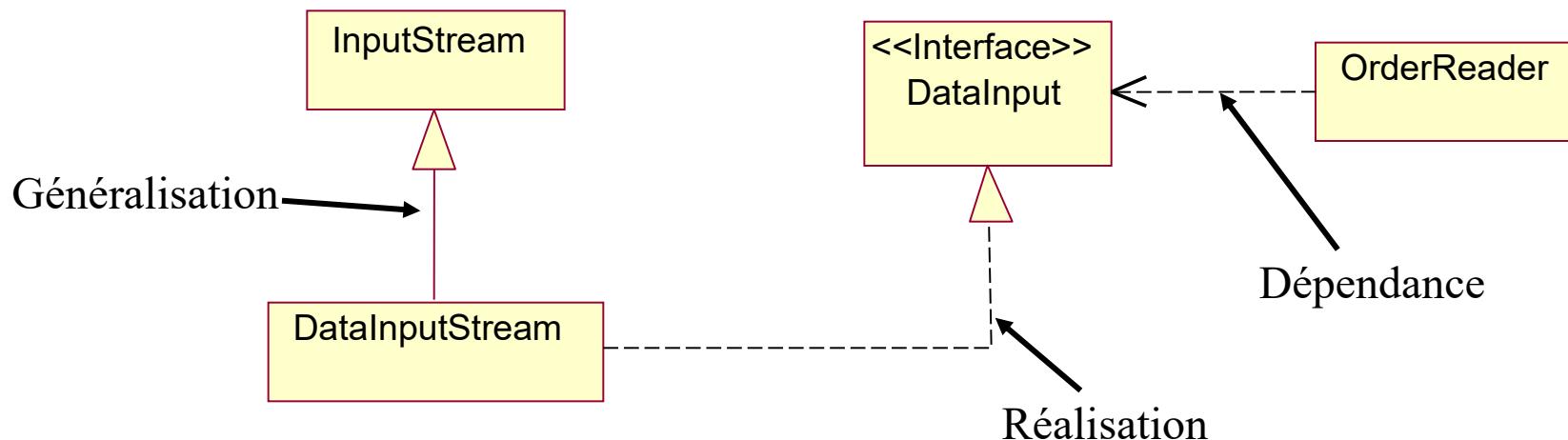
- Super-classe :
 - Contient les attributs et les opérations communs.
- Sous-classe :
 - Raffine ou restreint les attributs et opérations hérités.
 - Ajoute ses propres attributs et opérations.
- Une sous-classe hérite de sa super-classe :
 - Attributs
 - Opérations
 - Associations
- Héritage multiple :
 - Une classe hérite de ses super-classes.
 - Ses attributs et opérations sont ceux de ses parents.
 - UML ne fournit pas de règles de résolution de conflits.

Interfaces /Classes abstraites

- Classe abstraite : nom en italique ou la contrainte {abstract}
- Interface :
 - Utilise un type pour décrire le comportement visible d'une classe, sans implémentation.
 - Est un stéréotype.
- Une classe peut :
 - Spécialiser une classe abstraite.
 - Réaliser une interface.

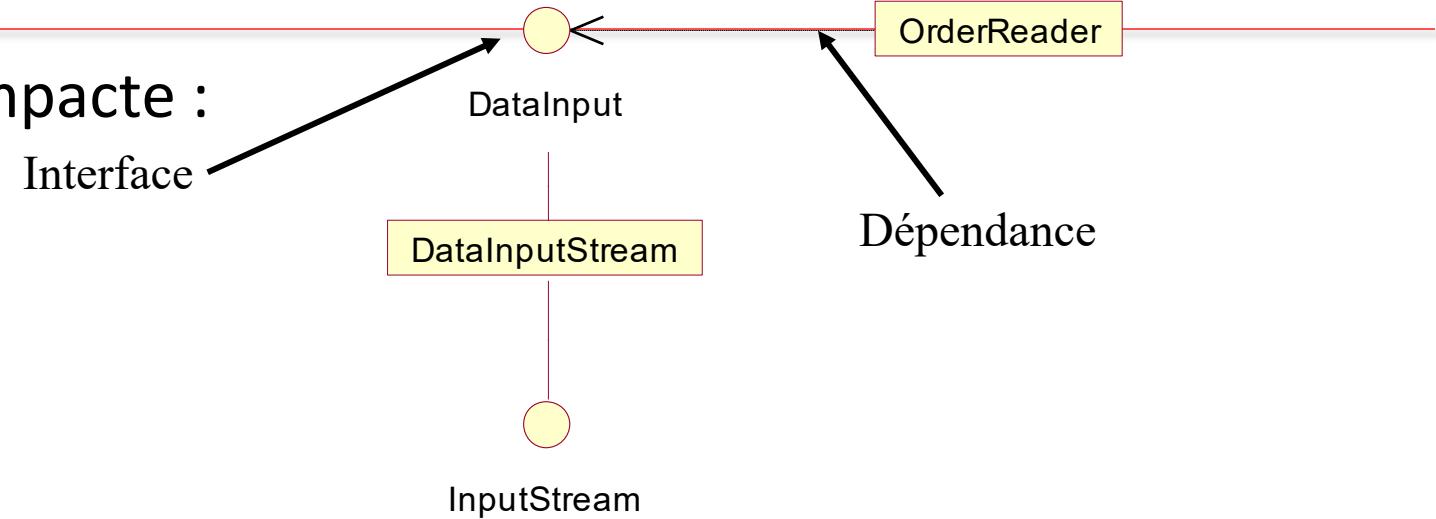
Interfaces - Classes abstraites

- En UML, le symbole de la réalisation est délibérément semblable à la généralisation.
- Réalisation : une classe implémente le comportement spécifié par une autre. Une classe d'implémentation peut en réaliser une autre : elle se conforme à son interface sans en hériter forcément.



Interfaces - Classes abstraites

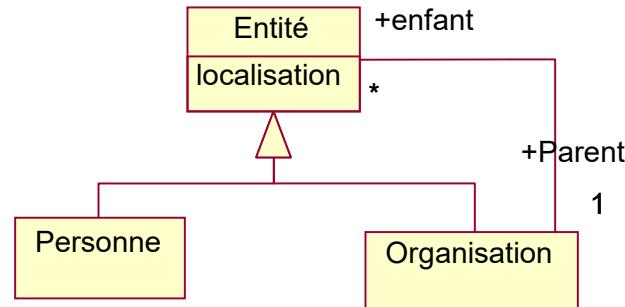
- Autre notation plus compacte :



- la réalisation des interfaces est représentée par de petits cercles, rattachés à la classe qui réalise l'interface.
- Ne distingue pas entre réalisation d'interface et sous-classement de classe abstraite.
- Mais ne permet pas de représenter les opérations des interfaces ni les relations de généralisation entre elles.

Diagramme d'objets

- Un diagramme d'objets ou d'instances est un instantané des objets présents à un moment donné.
- Diagramme de collaborations dépourvu de messages.
- Utilité : pour représenter une configuration particulière d'objets, surtout quand les liens possibles entre objets sont complexes :



Structure d'une entité

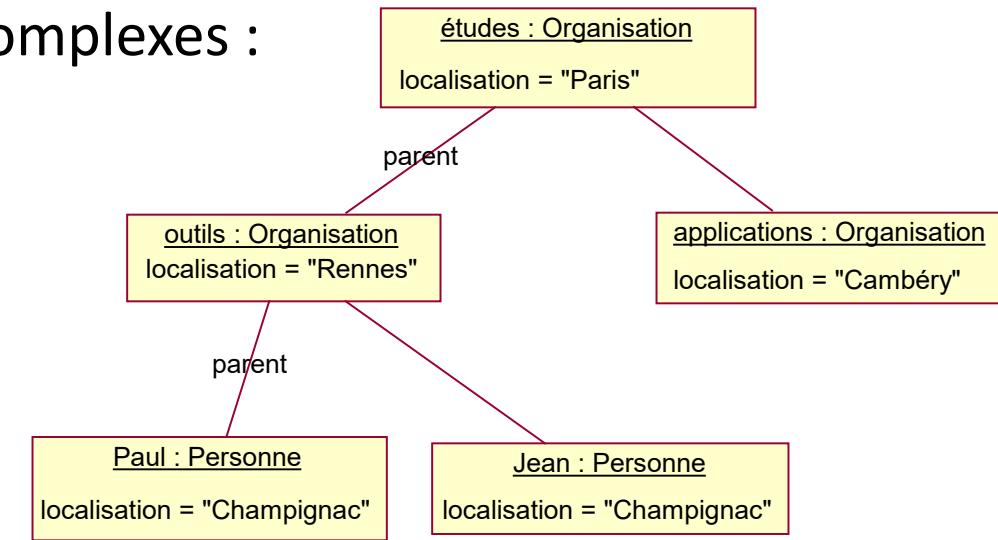
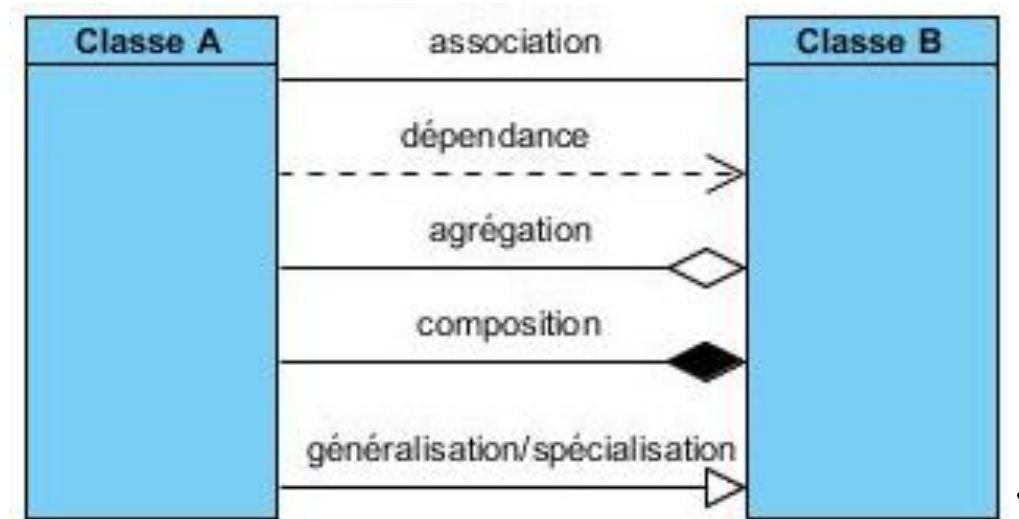


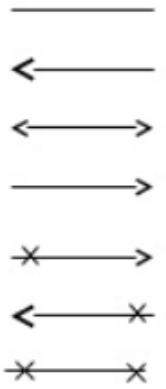
Diagramme d'objets : exemples d'instances
d'une entité

Associations & Relations

- Au cœur des phases d'analyse et de conception
- **Différents types de relations :**
 - **entre classes :**



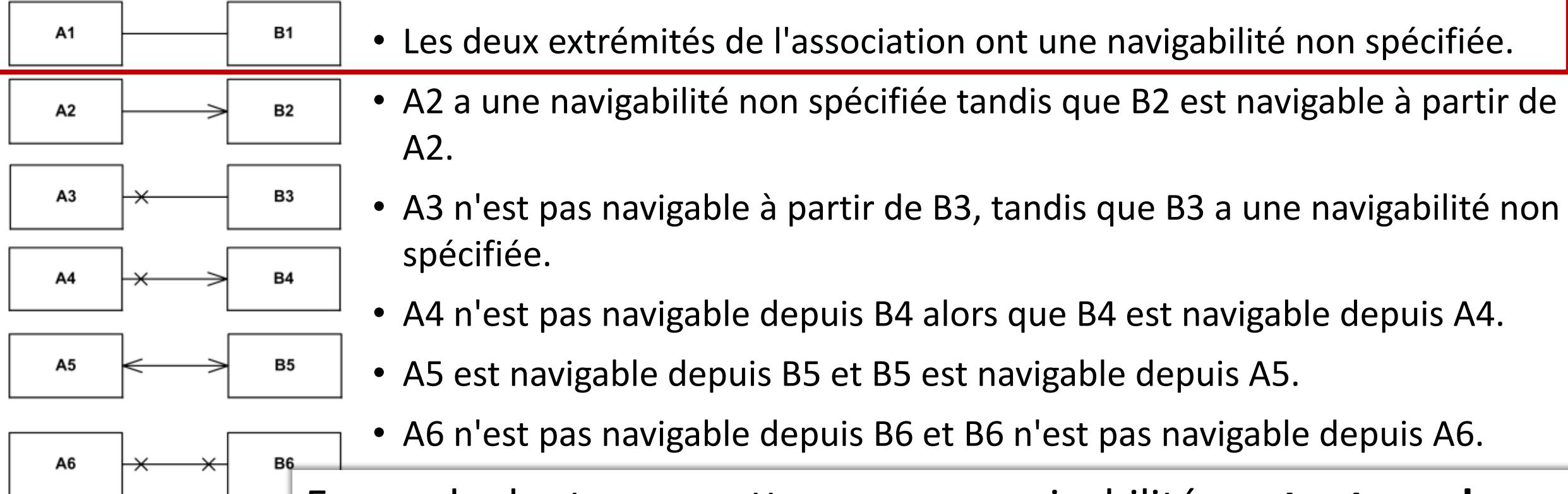
Association :



- **entre classe et interface :** implémentation & association
- **entre interfaces :** généralisation

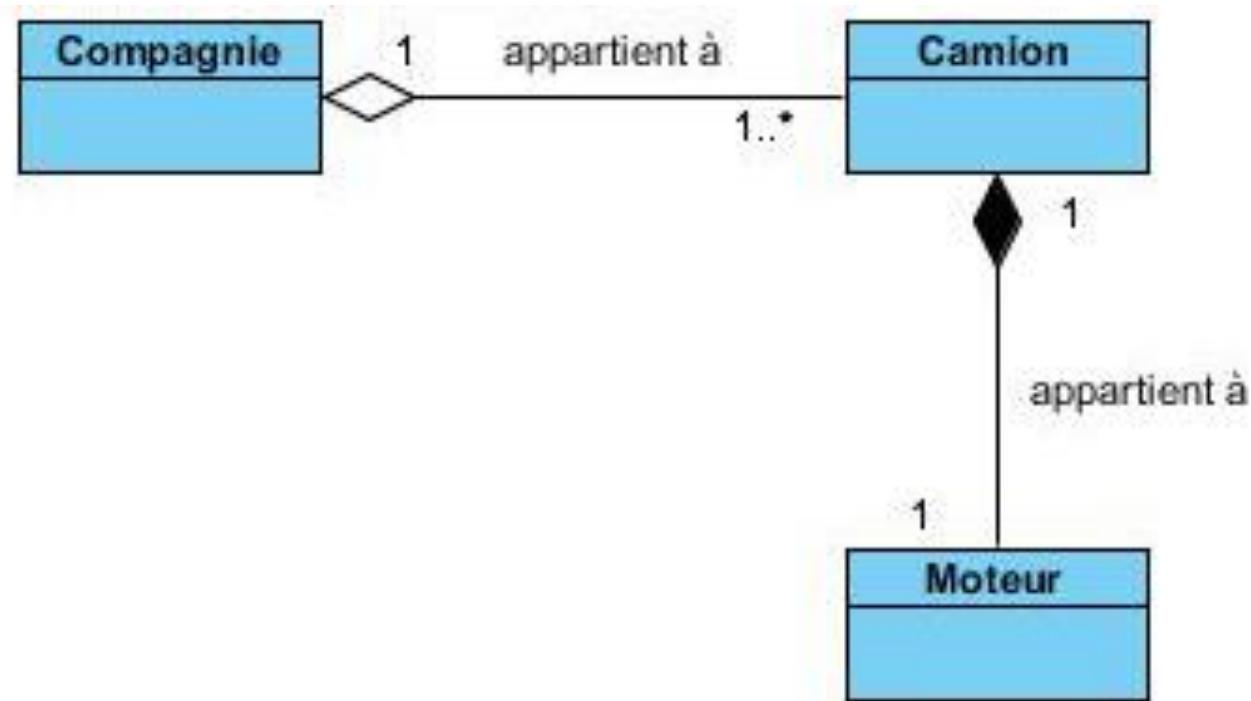
Association et navigation

Souvent en phase d'analyse



En cas de doute, ne mettre aucune navigabilité, **surtout en phase d'analyse**. Généralement spécifiée en fin de phase de conception.

Agrégation et Composition



Agrégation

- Une association exprimant un couplage fort lié à une relation de subordination : fort couplage logique
- Bidirectionnelle et asymétrique.
- **Attention :**
 - Un élément agrégé peut être lié à d'autres classes
 - La suppression de l'ensemble n'entraîne pas celle de l'élément
- **Exemple :** un ouvrage (*agrégat*) et ses exemplaires (*composant*).

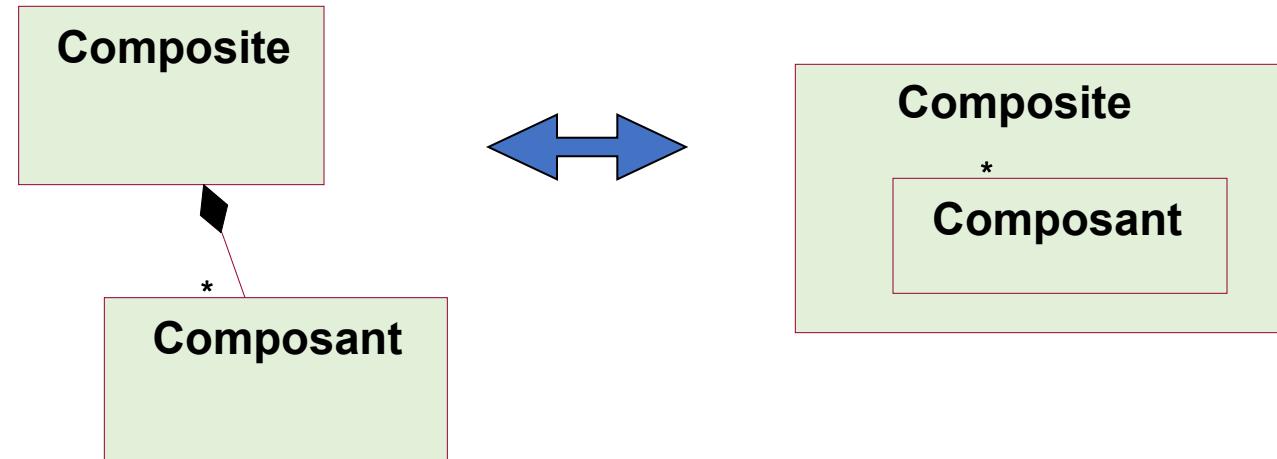


Composition

- La composition est
 - une agrégation forte
 - lie les cycles de vie entre le composé ou composite (ensemble) et les composants (éléments)
- Les règles supplémentaires obligatoires pour la composition sont les suivantes :
 - la suppression du composé entraîne la suppression des composants
 - les composants sont créés par le composé
 - Un composant est lié à un composé unique

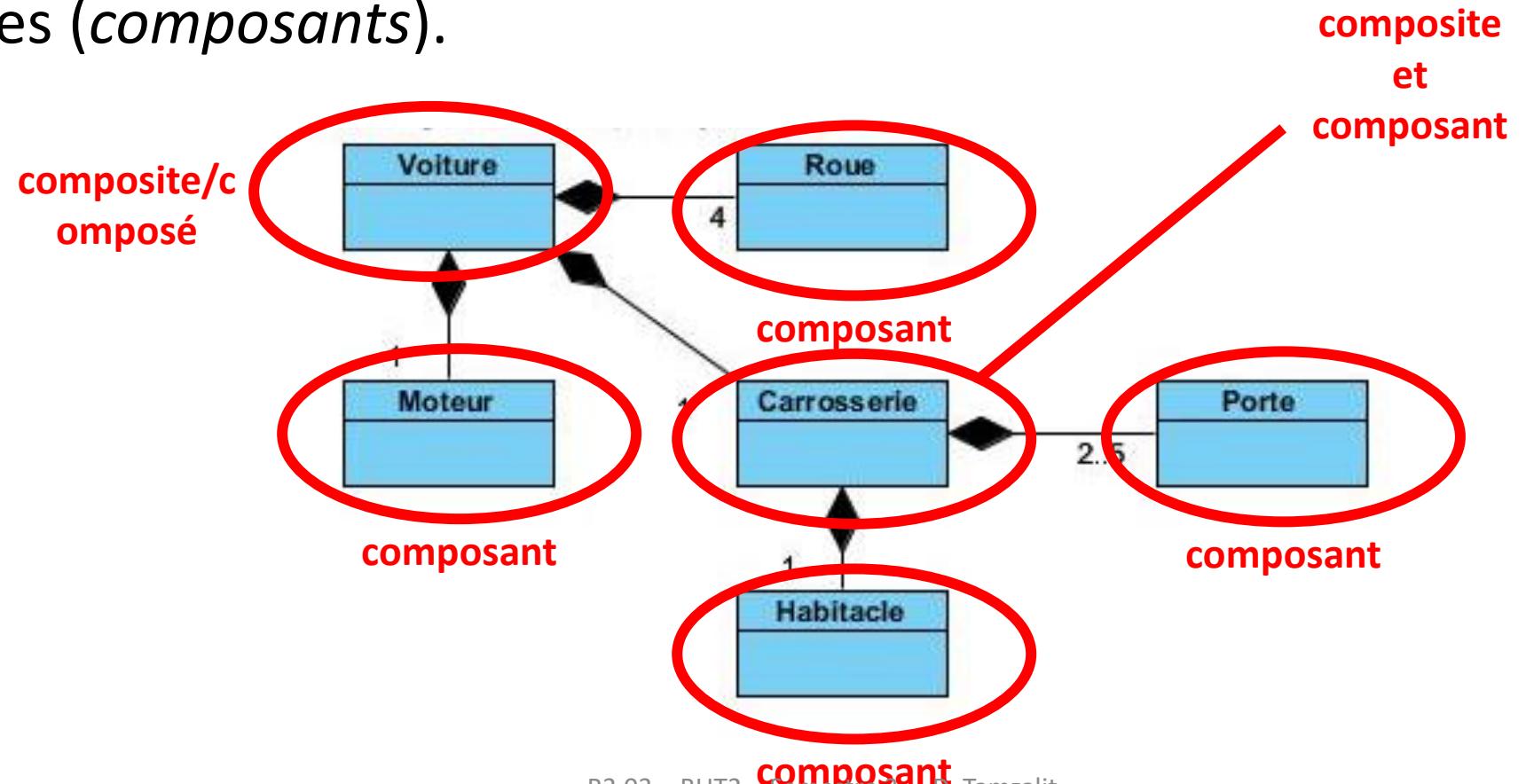
Composition

- Exprime souvent une composition physique.
- UML offre deux possibilités de représentation :

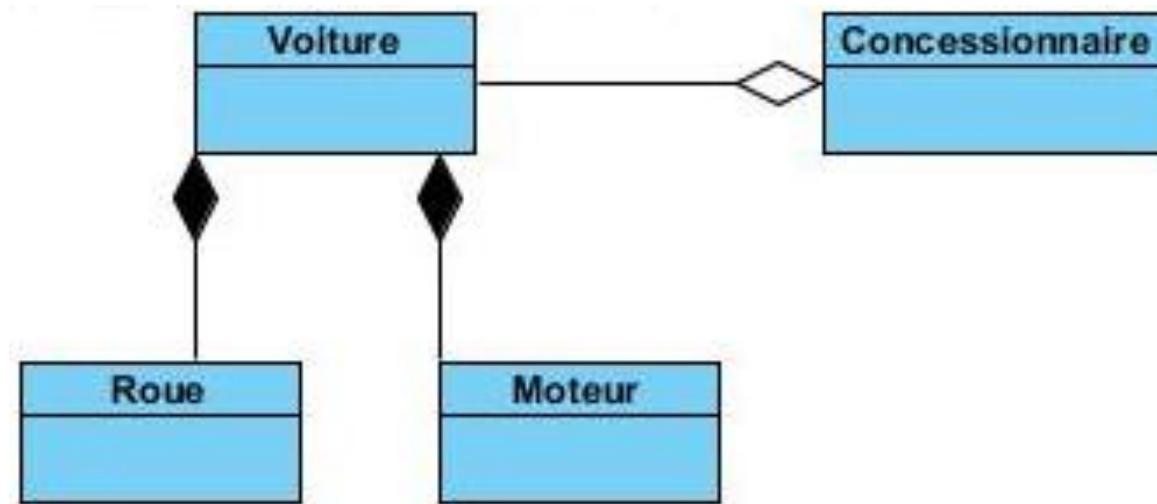


Composition

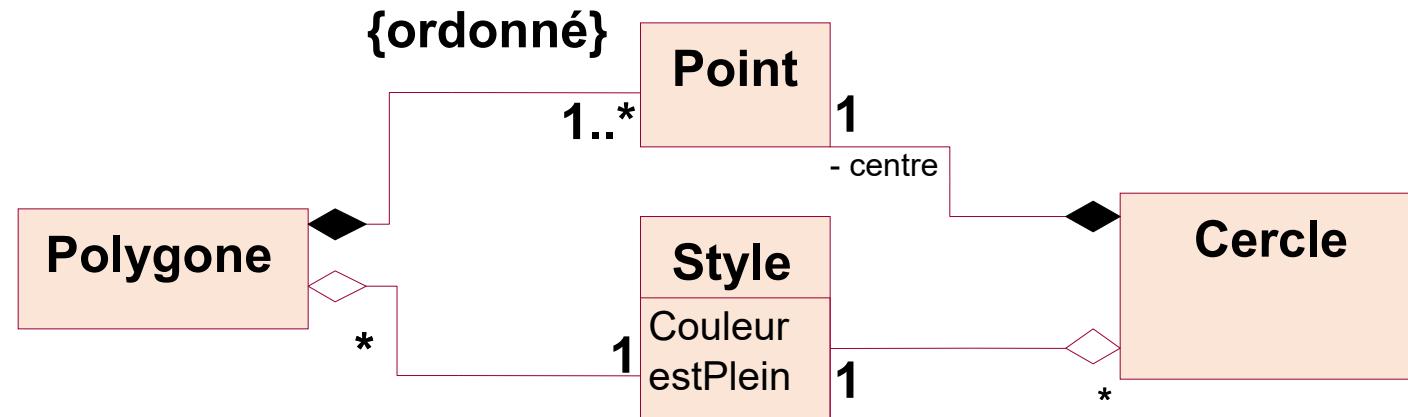
- Exemple : une voiture (*composite*) et son moteur, sa carrosserie et ses roues (*composants*).



Agrégation et composition

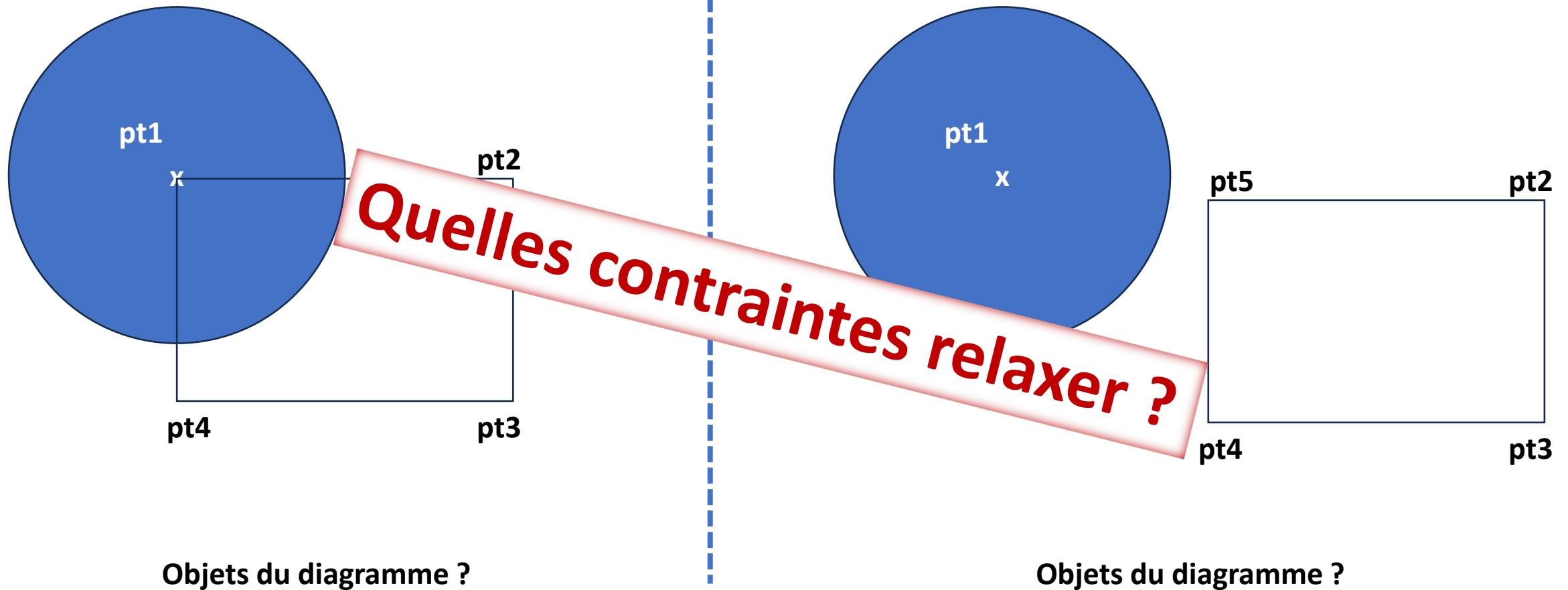
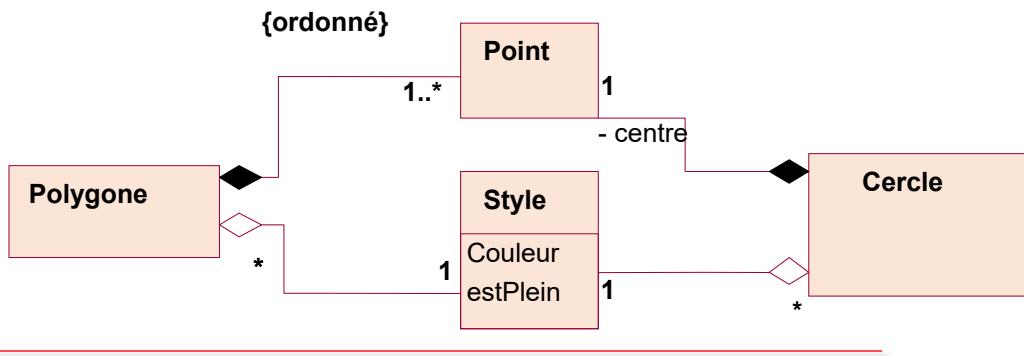


Agrégation et Composition



- Un Polygone ainsi qu'un Cercle sont composés de points :
 - La destruction du composite implique la destruction de la composition.
 - La destruction du composant implique celle du composite.
- Un Style peut concerner plusieurs Polygones et/ou Cercles :
 - Style est indépendant de l'existence d'agrégats.
 - Sa modification implique forcément celle de ses agrégats.

Agrégation et Composition

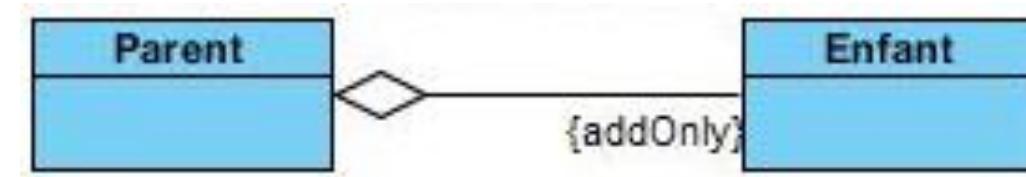


Agrégation & composition vers le Code

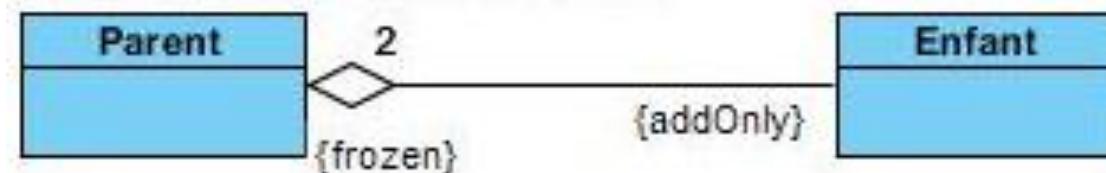
- Agrégation = Association
 - Règles de passage au code identiques,
- Composition = association + dépendance forte de la vie des composants vis-à-vis de leur composite
 - Règles de passage au code identique à une association
 - **+ prévoir un destructeur !**

D'autres contraintes entre associations

- *{addOnly}* : autorise l'ajout de nouveaux objets mais pas leur suppression ni leur mise à jour.



- *{Frozen}* : interdit l'ajout, la suppression ou la mise à jour des liens d'un objet vers les objets de la classe associée.



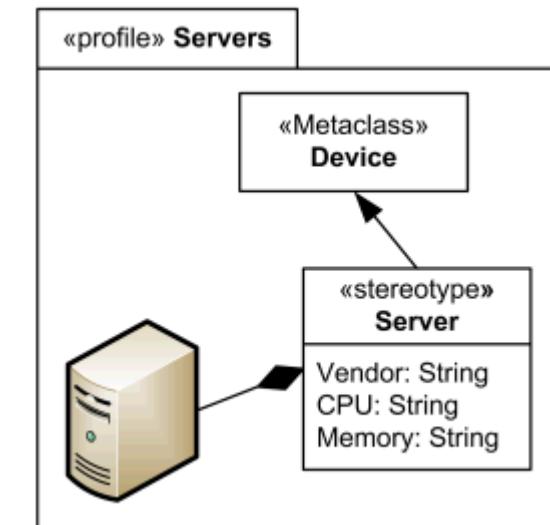
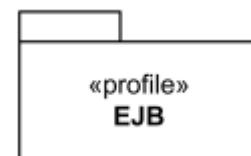
Stéréotype

- Principal mécanisme d'extension d'UML.
- Permet de construire de nouveaux types, par exemple des objets affichables.
- Généralement représenté par un texte entre guillemets
 - Exemple <<Objets Dessinables>> et une icône associée.
- Stéréotypes de classe, d'association ou de généralisation.



<<Objets Dessinables>>
Class4

Exemples stéréotypes



Qualification

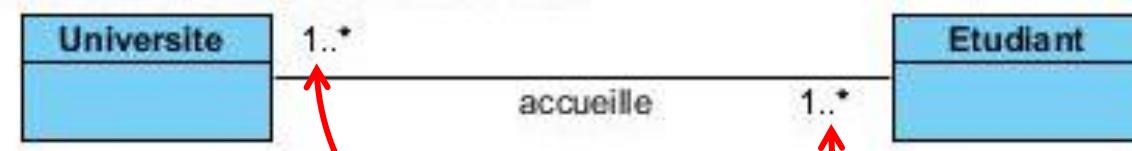
- Une association peut être qualifiée.
- Restriction qui consiste à sélectionner un sous-ensemble d'objets parmi l'ensemble des objets qui participent à une association
 - Restriction de la multiplicité



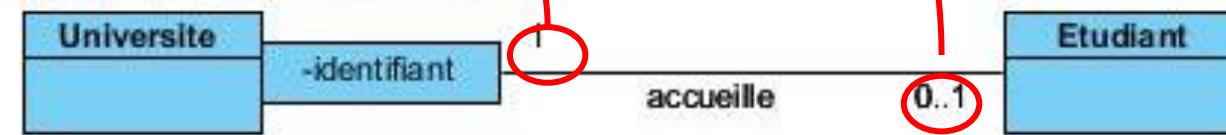
- La paire (instance de la classe A, valeur de la qualification) identifie un sous-ensemble des instances de la classe B.
- La clé est représentée sur le rôle de la classe de départ, dans un rectangle.

Exemples qualification

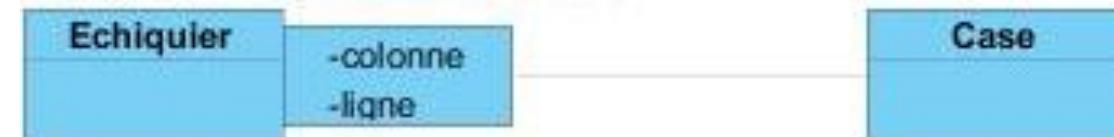
- Une université peut accueillir plusieurs étudiants et un étudiant peut être inscrit à plusieurs universités.



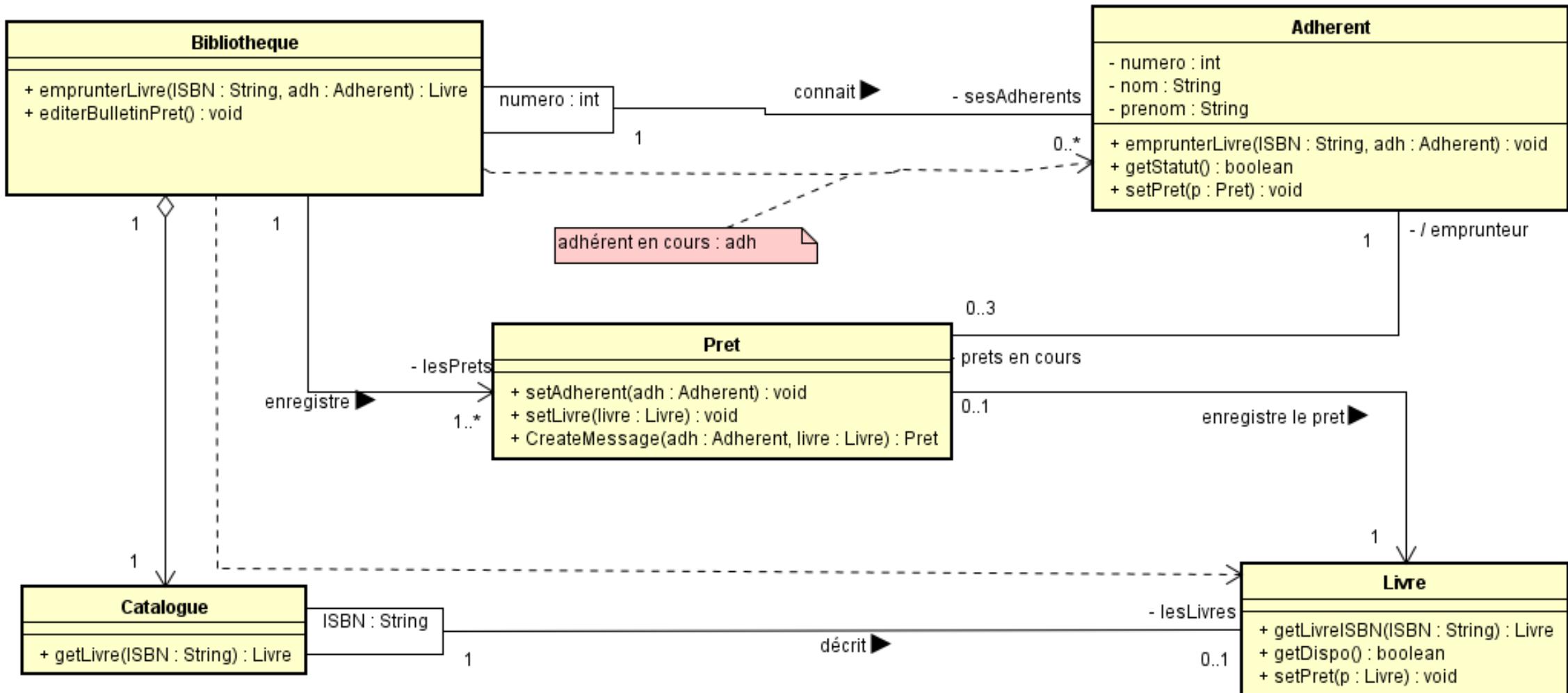
- Un étudiant a un identifiant unique dans une université qui l'accueille



- Clés multiples



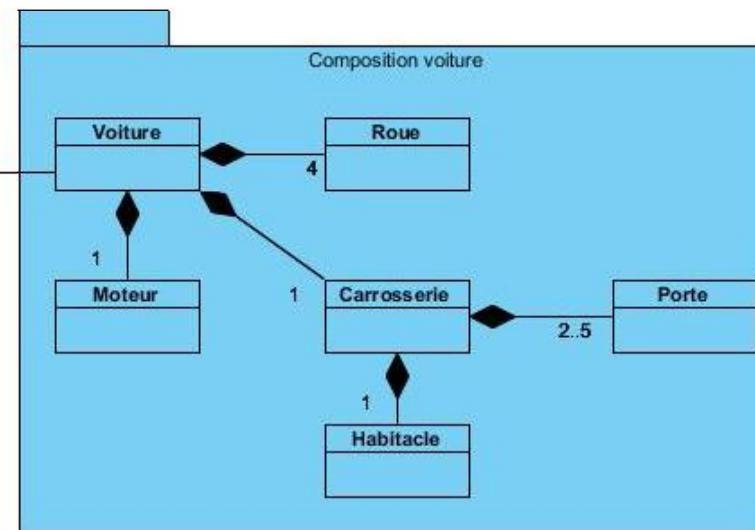
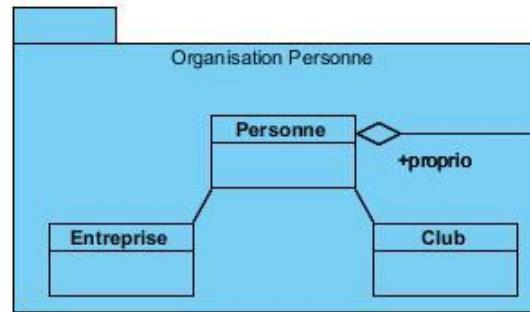
Autres exemples qualification



Paquetages

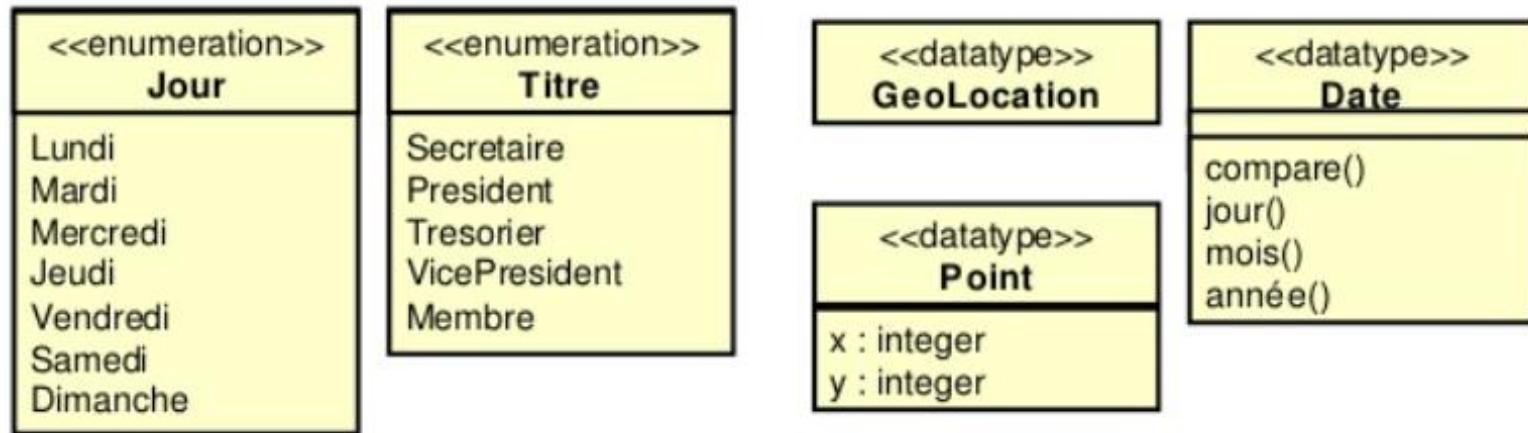
- Un paquetage (*package*) peut regrouper des classes, des interfaces et des paquetages.

Visual Paradigm Standard Edition (University of Nantes)



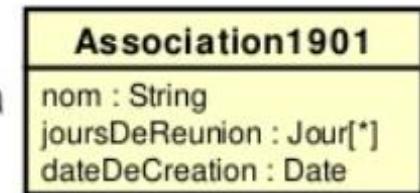
- Relations entre paquetages :
 - Relations classiques entre classes
 - Relations d'import

Enumération et types de données



- Utilisable comme type d'attributs
- Valeurs (pas d'identité)
- Type de données typiquement définis dans des « librairies »

Exemple
d'utilisation



Ressources

- Précédents cours DUT
- UML2 par la pratique, Pascal Roques, Editions Eyrolles,
- UML2, de l'apprentissage à la pratique, Laurent Audibert, Edition Ellipses, <https://laurent-audibert.developpez.com/Cours-UML/?page=mise-en-oeuvre-uml> .
- Support cours E. Syriani, Université de Montréal.