TEK-UP Ecole Supérieure Privée Technologie
& Ingénierie

Modélisation UML

A.U. 2019-2020

Plan du cours

Ch. III

Analyse : Modèle conceptuel (Classe + Objet)

Sommaire

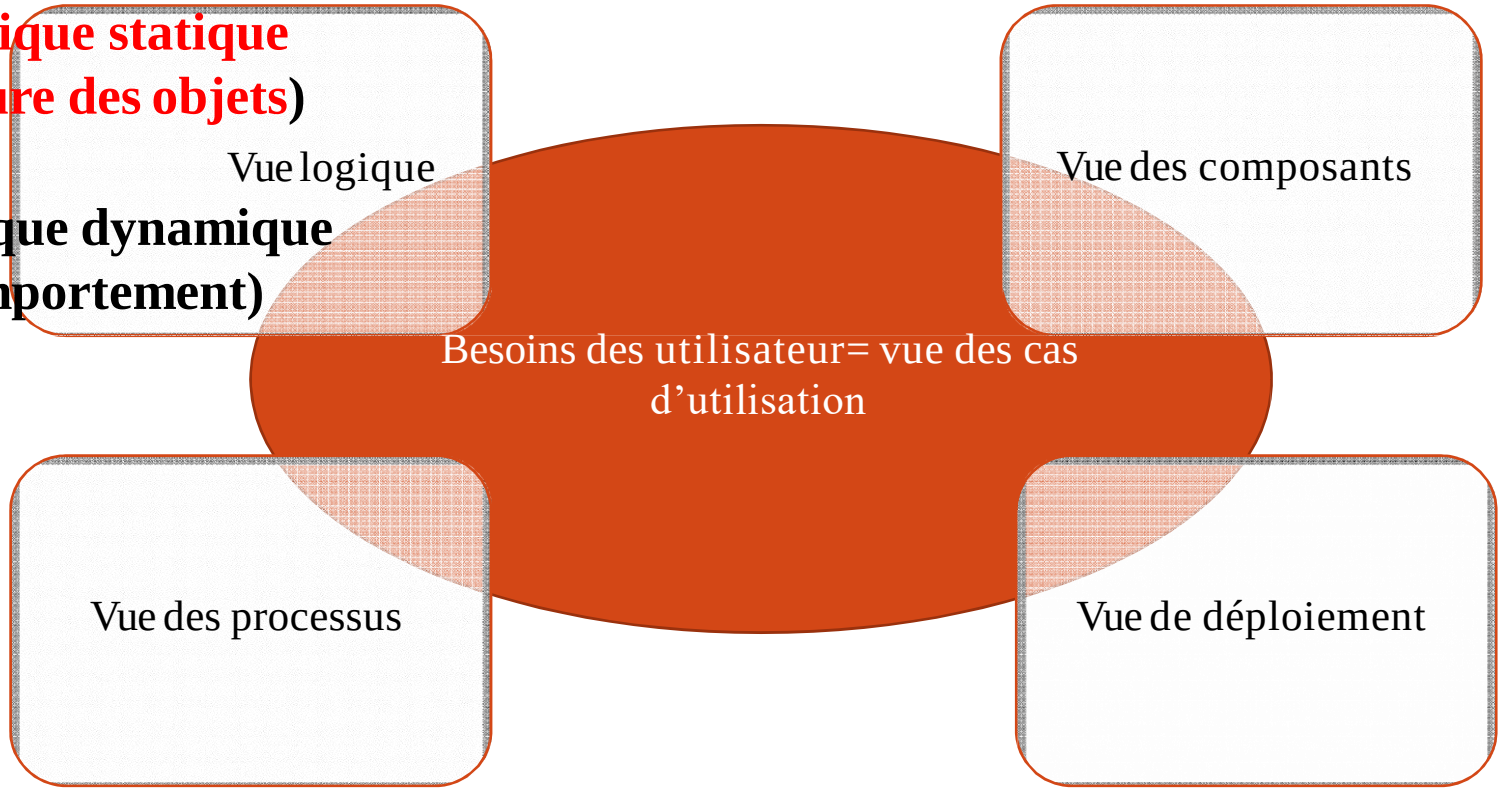
- Introduction
- Diagramme d’objets
- Diagramme de classes

Motivation

Vue logique statique (Structure des objets)

Vue logique

Vue logique dynamique (Comportement)



Modèle structurel

- Une vue d'un système qui met l'accent sur la structure des objets, avec leurs relations, leurs attributs et leurs opérations

Diagrammes structurels

- Montrent la structure statique d'un modèle
  - Les entités qui existent (e.g., classes, interfaces, composants, noeuds)
  - Leur structure interne
  - Leurs relations avec d'autres entités
- Ne montrent pas
  - Des informations temporelles ou dynamiques
- Le diagramme de classes et le diagramme d'objets sont les pièces maîtresses de la vue structurale
  - Dans UML , ils sont répertoriés comme des diagrammes montrant la structure «statique »
  - Les classes et les objets modélisent les objets matériels ou immatériels qui existent dans le système qu'on essaie de décrire.
  - Les relations entre les classes et les objets établissent les connexions entre les divers éléments de modélisation

Réification

- Définition en modélisation
  - Décision que prend le modélisateur de considérer une portion du réel comme un objet
    - Avec ce qu'implique la notion d'objet
    - Le principe de réification pragmatique formulé par Jacques FERBER
      - "Si l'on parle de quelque chose en lui attribuant des propriétés, ou si cette chose doit être manipulée, alors il faut la représenter sous forme d'objet."

Réification

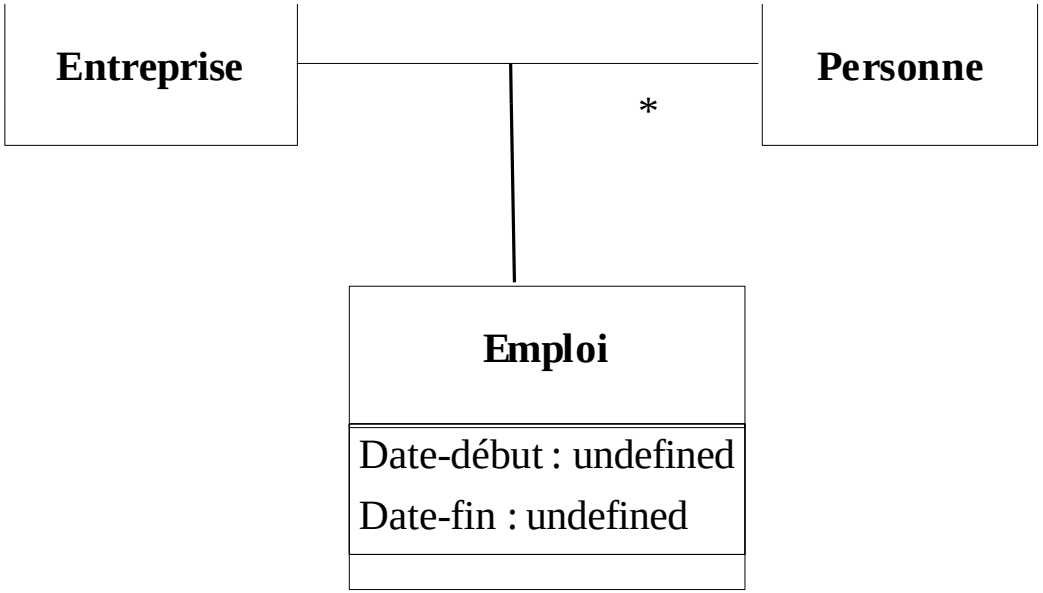
➢ En pratique

- Réification= matérialiser un concept par un objet
- Un concept abstrait peut être « réifié »
  - l'événement « à 10h45 une carte bleue à été introduite »
- Une relation entre deux objets peut être « réifiée »
  - « Ali possède la voiture immatriculée 875 TU 129 » est réifié dans le monde réel par une carte grise

- Réifier un concept permet de le manipuler concrètement

Exemple de réification

- Une personne travaille pour une entreprise
  - Cette relation est décrite par des informations



Principe d'abstraction

- Une abstraction fait ressortir les caractéristiques d'une structure qui la distinguent de tous les autres types de structures du domaine et donc procure des frontières conceptuelles rigoureusement définies par rapport au point de vue de l'observateur.

Principe d'abstraction

- Pour être véritablement intéressant, un objet doit permettre un certain degré d'abstraction.
- Le processus d'abstraction consiste à identifier pour un ensemble d'éléments :
  - Des caractéristiques communes à tous les éléments
  - des mécanismes communs à tous les éléments
- description générique de l'ensemble considéré : se focaliser sur l'essentiel, cacher les détails.

Abstraction

- L'abstraction est une ignorance **sélective**

> L'objectif de l'abstraction n'est pas d'être vague, mais de créer un nouveau niveau sémantique dans lequel il est possible d'être très précis.
> « Edsger Dijkstra »

Principe d'encapsulation

Définition :

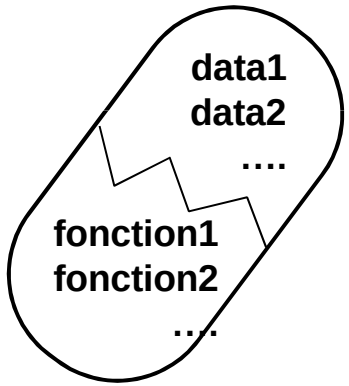
L'encapsulation est le procédé de séparation des éléments d'une abstraction qui constituent sa structure et son comportement. Elle permet de dissocier l'interface contractuelle de la mise en œuvre d'une abstraction.

Encapsulation (…)

Le principe d'encapsulation consiste à regrouper, dans un même élément informatique, les aspects statique et dynamique (c.a.d. les données et les fonctions) spécifiques à une entité.

Cet élément informatique est appelé : « **objet** »

- Les [structures de] données définies dans un objet sont appelées les attributs de l'objet;
- Les fonctions [de manipulation] définies dans un objet sont appelées les méthodes de l'objet.



On a donc la relation fondamentale:

OBJET = attributs + méthodes

Encapsulation (…)

En plus du regroupement des éléments statique et dynamique [d'une entité], l'encapsulation permet de définir deux niveaux de perception :

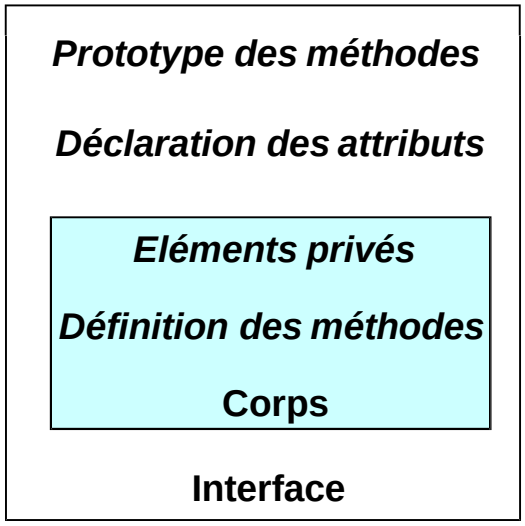
- **Le niveau externe : perception de l'objet depuis l'extérieur**
- **Le niveau interne : perception de l'objet depuis l'intérieur**

Le niveau externe, correspond à la partie visible de l'objet;

il est constitué des spécifications des éléments [de l'objet] visibles de l'extérieur (appelé: « éléments publics »), à savoir les prototypes et les déclarations de ses méthodes et attributs publics. Ce niveau représente donc l'interface de l'objet avec l'extérieur.

Le niveau interne correspond à l'implémentation de l'objet; il est constitué des éléments de l'objet visibles uniquement de l'intérieur de cet objet (appelés « éléments privés »).

Ce niveau représente donc le corps de l'objet.



Encapsulation (…)

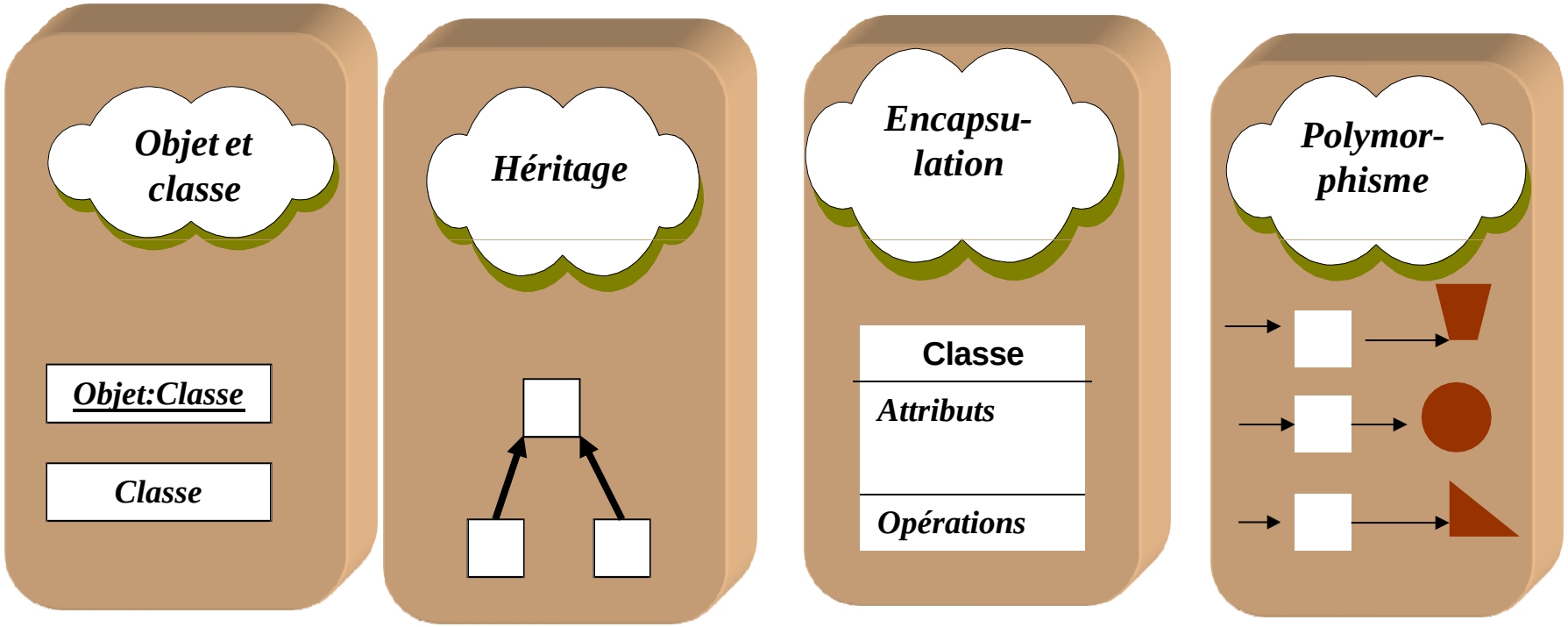
Interface

réalisation

2 rôles :

- **utilisateur : manipule les éléments de l'abstraction qui constituent l'interface**
- **implanteur :réalise ce qui est encapsulé**

Approche objet



Objets et classes

Objet : une entité concrète avec une identité bien définie qui encapsule un état et un comportement. L 'état est représenté par des valeurs d'attribut et des associations. le comportement par des méthodes.

Un objet est une instance d 'une classe.

Classe : une description d'un ensemble d'objets qui partagent les mêmes attributs, opérations, méthodes, relations et contraintes.

Une classe peut posséder des attributs ou des méthodes « de classe ».

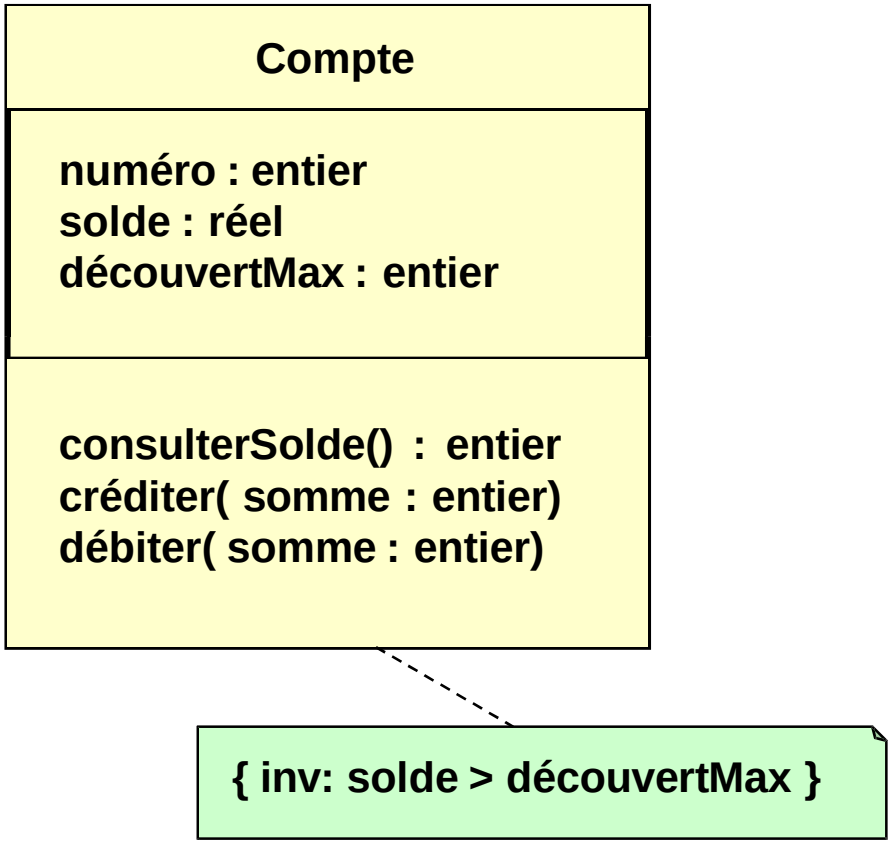
MaVoiture : Voiture
marque = Renault Modèle = Nevada Immatriculation = 648ADX38 AnnéeModele = 1992 Kilométrage = 285 000

Voiture
marque : chaîne Modèle : chaîne Immatriculation : chaîne (8) AnnéeModele : date Kilométrage : entier
Rouler ( ) Kilometrage_annuel_moyen ( )

Comment trouver les classes?

- Quels rôles ont les acteurs dans le système?
  - Ces rôles peuvent être vus comme des classes
  - Exemples: utilisateur, client,…
- A-t-on des systèmes externes?
  - Un système externe pourrait être vu comme une classe qui n'est pas dans le système étudié mais qui interagit avec lui.

Notation pour les classes



Nom de la classe

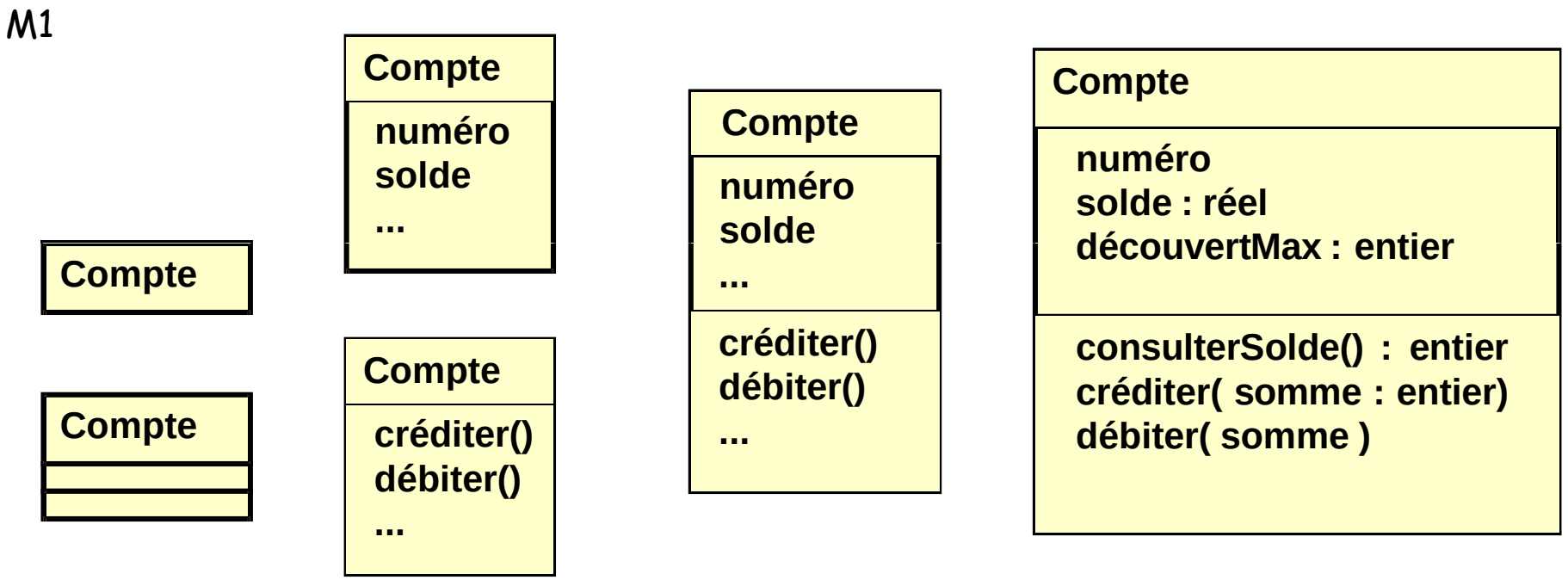
Attributs
- nom
- type

Opérations
- nom
- paramètre
- type du résultat

Contraintes

Par défaut, les attributs sont cachés et les opérations sont visibles

Notations simplifiées pour les classes

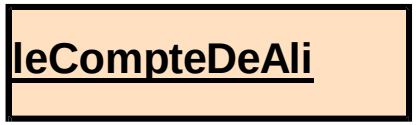


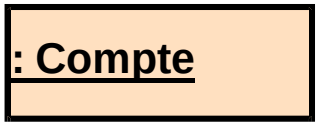
Note de style :

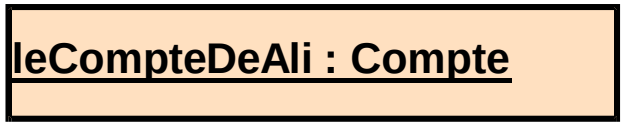
- les noms de classes commencent par une majuscule
- les noms d 'attributs et de méthodes commencent par une minuscule

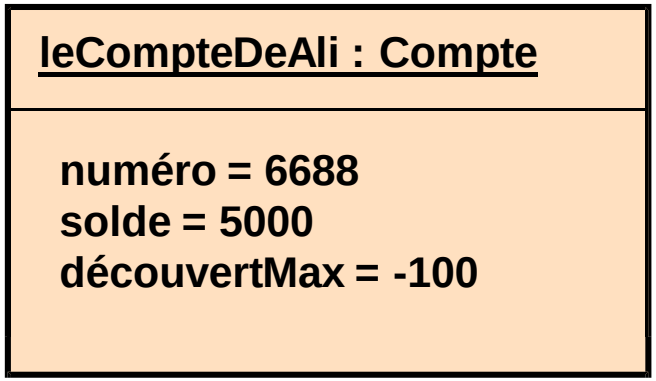
Notations pour les objets

MO

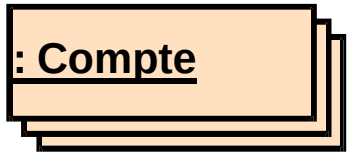








Une collection d'objets peut être représentée :



Convention :

- les noms d 'objets commencent par une minuscule et sont soulignés

Classe vs. Objets

Une classe spécifie la structure et le comportement d'un ensemble d'objets de même nature

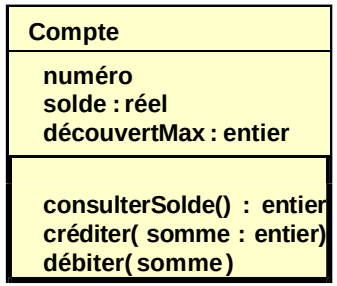


Diagramme de classes

- La structure d'une classe est constante

M1

M0

- Des objets peuvent être ajoutés ou détruits pendant l 'exécution
- La valeur des attributs des objets peut changer

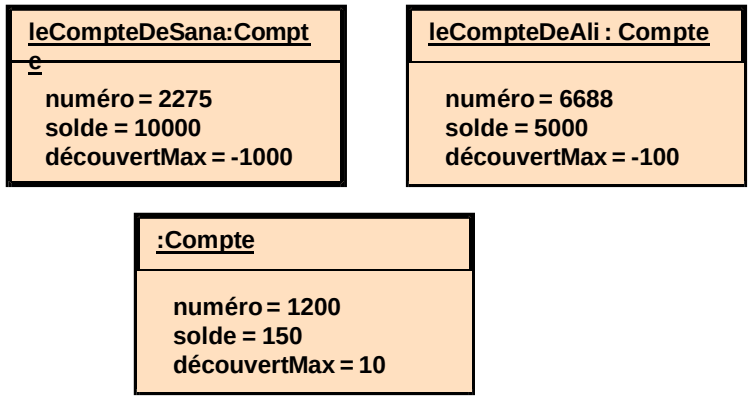


Diagramme d 'objets

Diagramme d'Objets

Structure statique d'un système, en termes d'objets et de liens entre ces objets.

Ces objets et ces liens possèdent des attributs qui possèdent des valeurs.

Un objet est une *instance* de classe et un lien est une *instance* d'association.

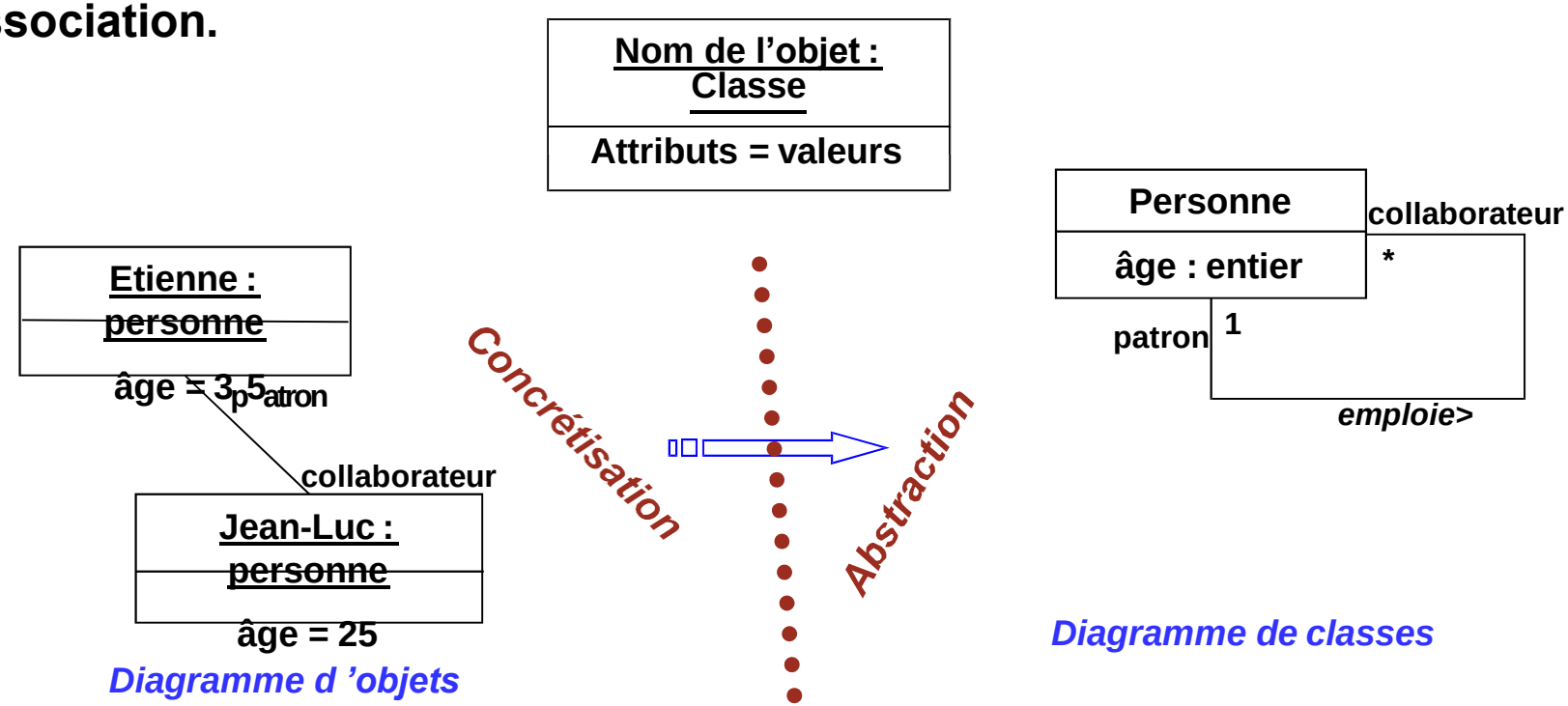
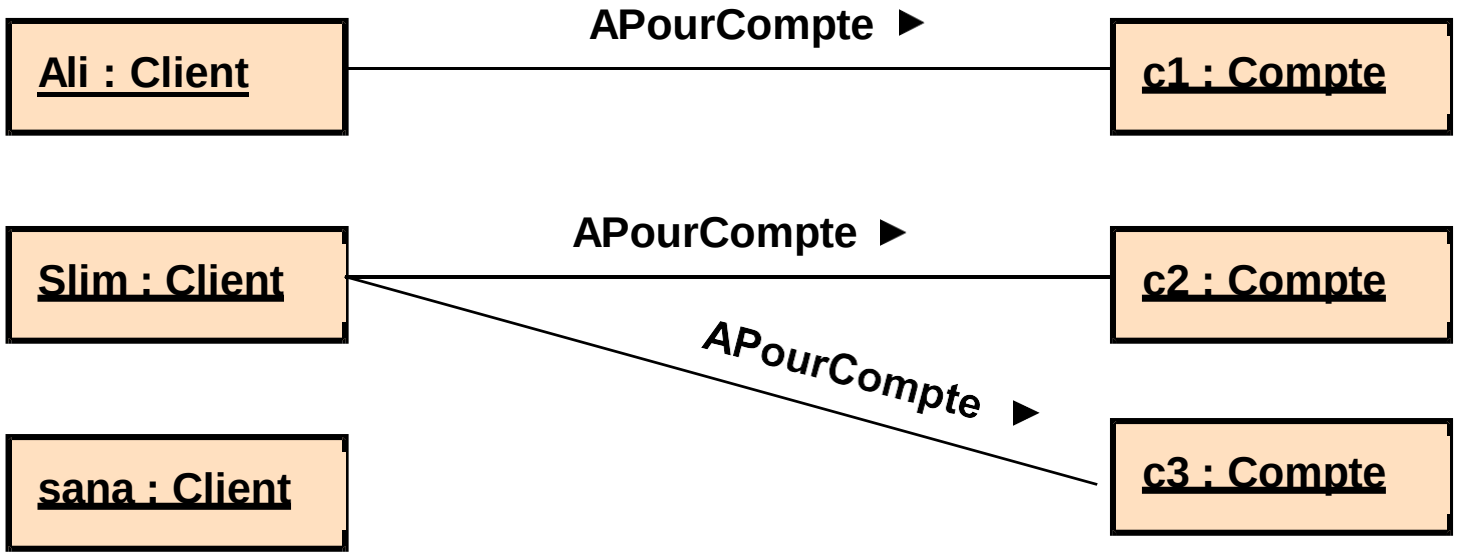


Diagramme d 'objets

Diagramme de classes

Liens (entre objets)

Un lien indique une connexion entre deux objets

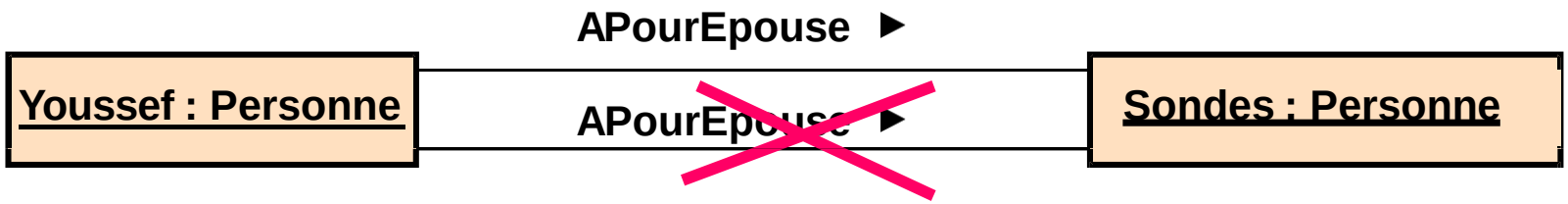


Note de style :

- les noms des liens sont des formes verbales et commencent par une majuscule
- ▶ indique le sens de la lecture (ex: « ali APourCompte c1 »)

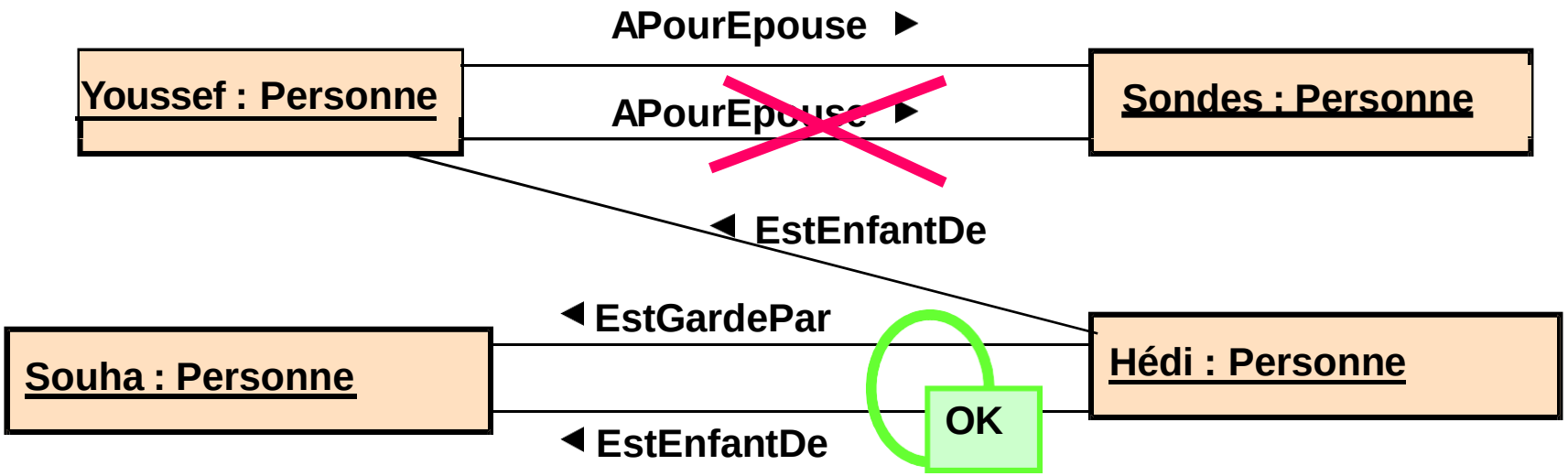
Contrainte sur les liens

- Au maximum un lien d'un type donné entre deux objets donnés*



Contrainte sur les liens

- Au maximum un lien d'un type donné entre deux objets donnés*



- Contrainte importante pour compendre les "classes associatives"
- (*) Contrainte pouvant être relachée via {nonunique} en UML 2.0
- ... voir plus loin les concepts avancés

Rôles

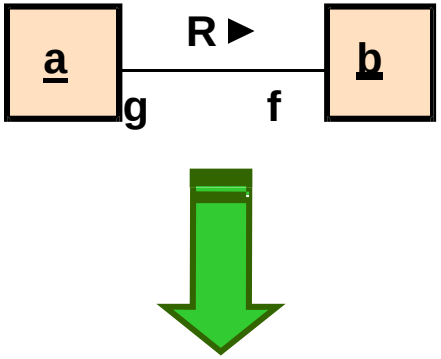
Chacun des deux objets joue un rôle différent dans le lien

Note de style :

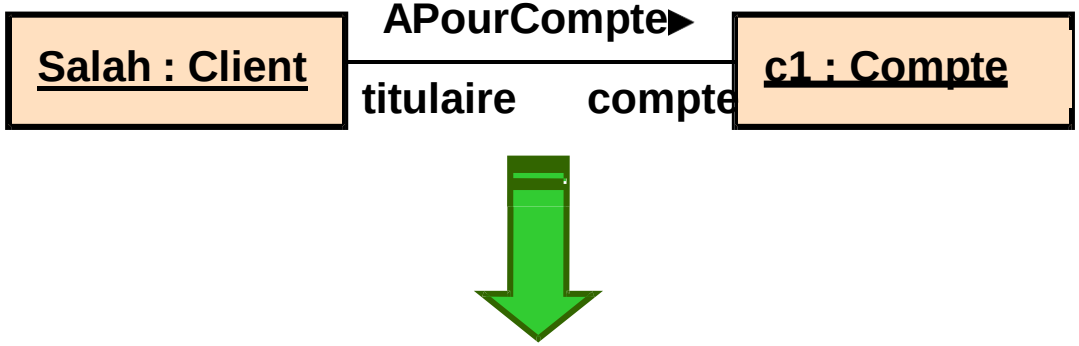
- choisir un groupe nominal pour désigner un rôle
- si un nom de rôle est omis, le nom de la classe fait office de nom

3 noms pour 1 concept

utilisations différentes selon le contexte



- a R b
- b "*joue le role de*" f "*pour*" a
- a "*joue le role de*" g "*pour*" b



- salah a pour compte c1
- c1 *joue le role de* compte *pour* salah
- salah *joue le role de* titulaire *pour* c1

Diagrammes d'objets

MO

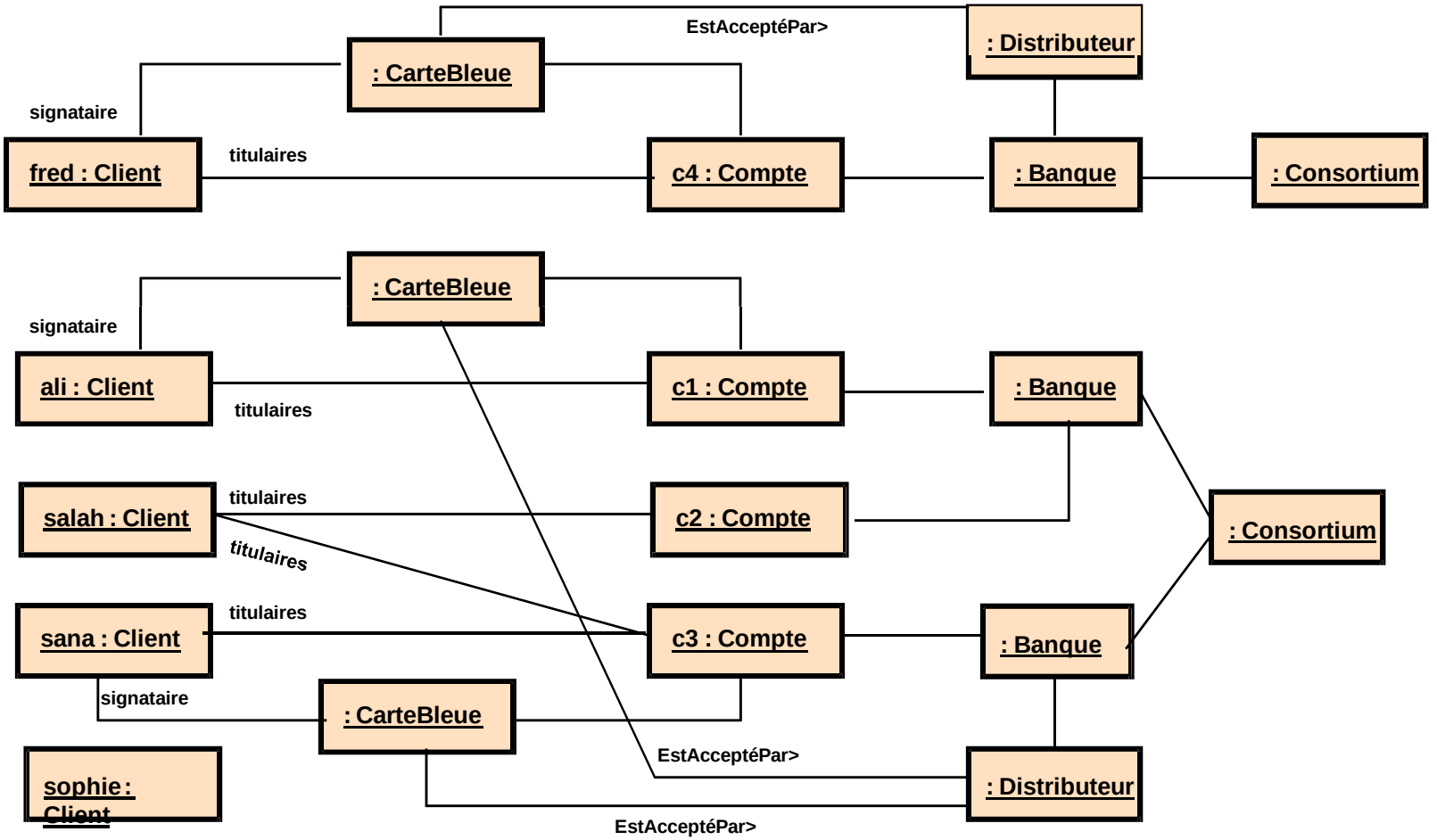
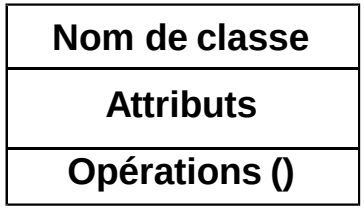
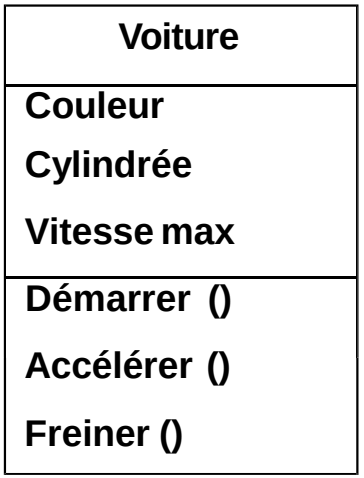


Diagramme de classes

Structure statique d'un système, en termes de classes et de relations entre ces classes.



exemple :

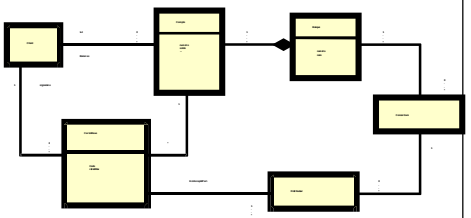


Syntaxe:

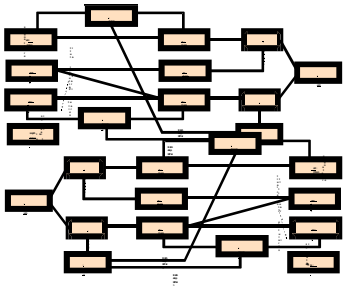
- **nom_attribut : type_attribut = valeur initiale**
- **nom_opération (nom_argument : type_argument = valeur_par_défaut, …) : type_retourné**

Diagrammes de classes vs. d'objets

- Un diagramme de classes
  - défini l'ensemble de tous les états possibles
  - les contraintes doivent toujours être vérifiées

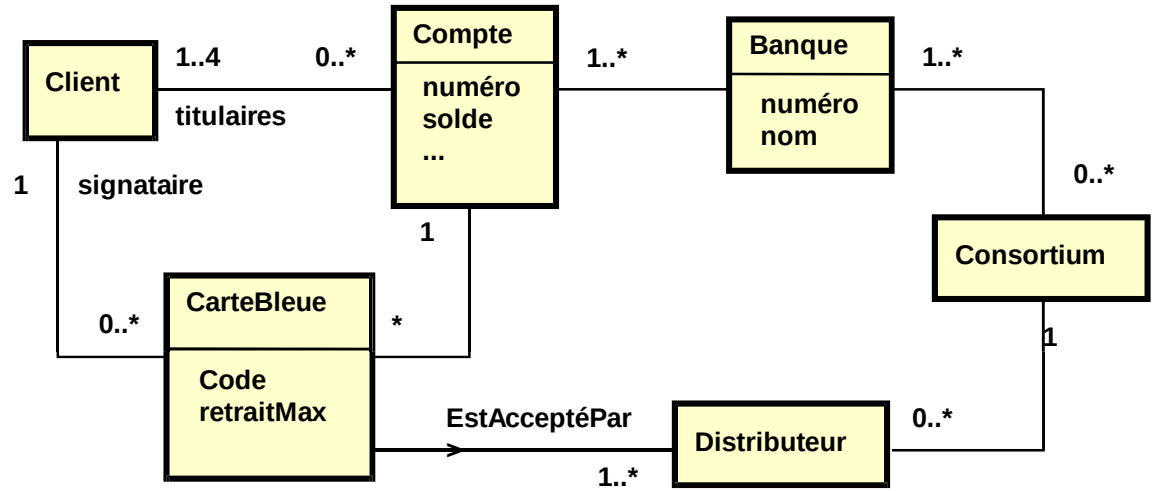


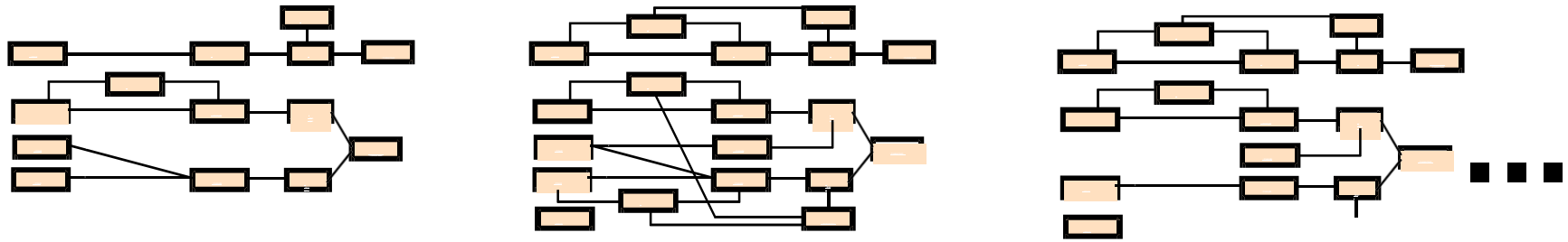
- Un diagramme d'objets
  - décrit un état possible à un instant t, un cas particulier
  - doit être conforme au modèle de classes



- Les diagrammes d'objets peuvent être utilisés pour
  - expliquer un diagramme de classes (donner un exemple)
  - valider un diagramme de classes (le "tester")

Diagrammes de classes vs. d'objets



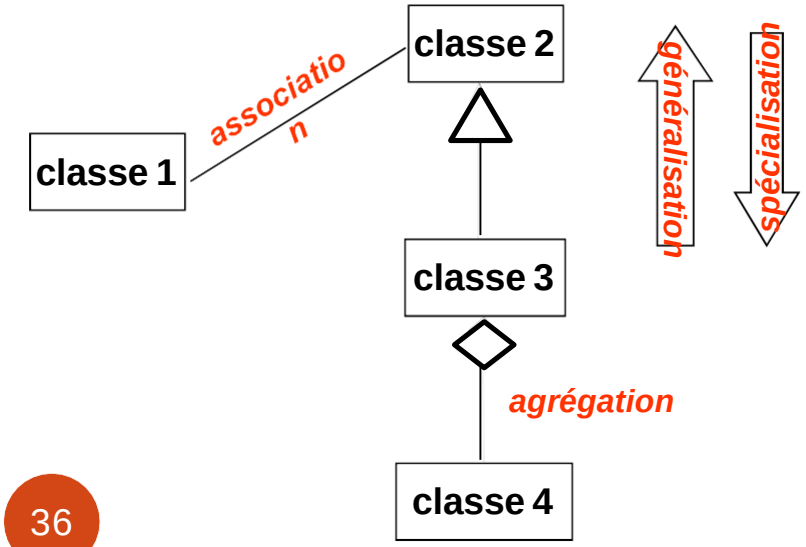


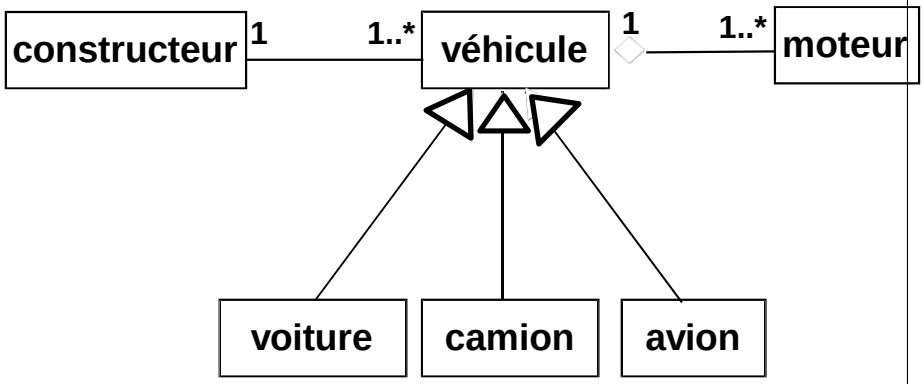
t_1 t_2 t_3

Diagramme de classes : Relations entre classes

Association : relation structurelle entre classes

Généralisation : factorisation des éléments communs d'un ensemble de classes dits *sous-classes* dans une classe plus générale dite *super-classe.* Elle signifie que la sous-classe *est un* ou *est une sorte de* la super-classe. Le lien inverse est appelé spécialisation





Associations (entre classes)

Une association décrit un ensemble de liens de même "sémantique"

Diagramme de classes (modélisation)

M1

M0

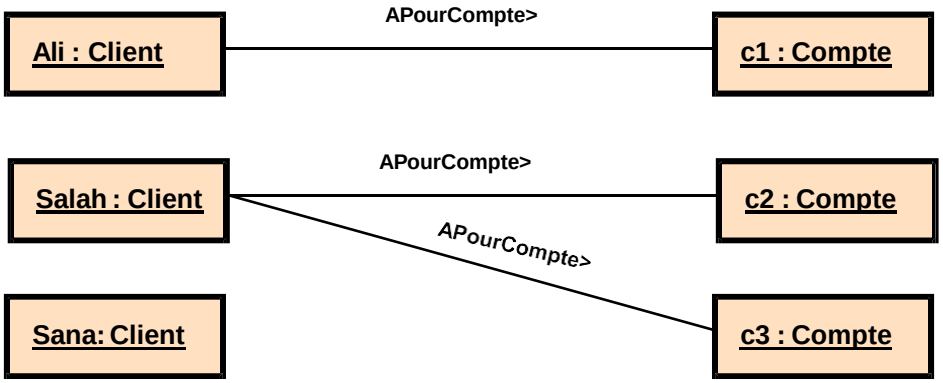
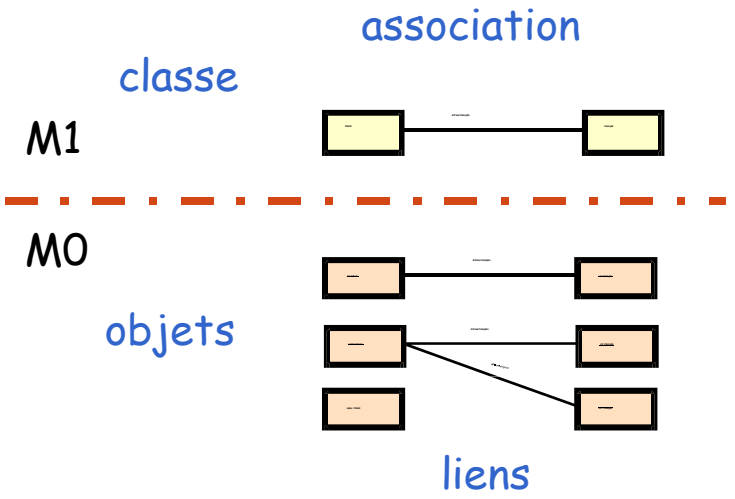


Diagramme d 'objets (exemplaires)

Association vs. Liens

- Un lien lie deux objets
- Une association lie deux classes

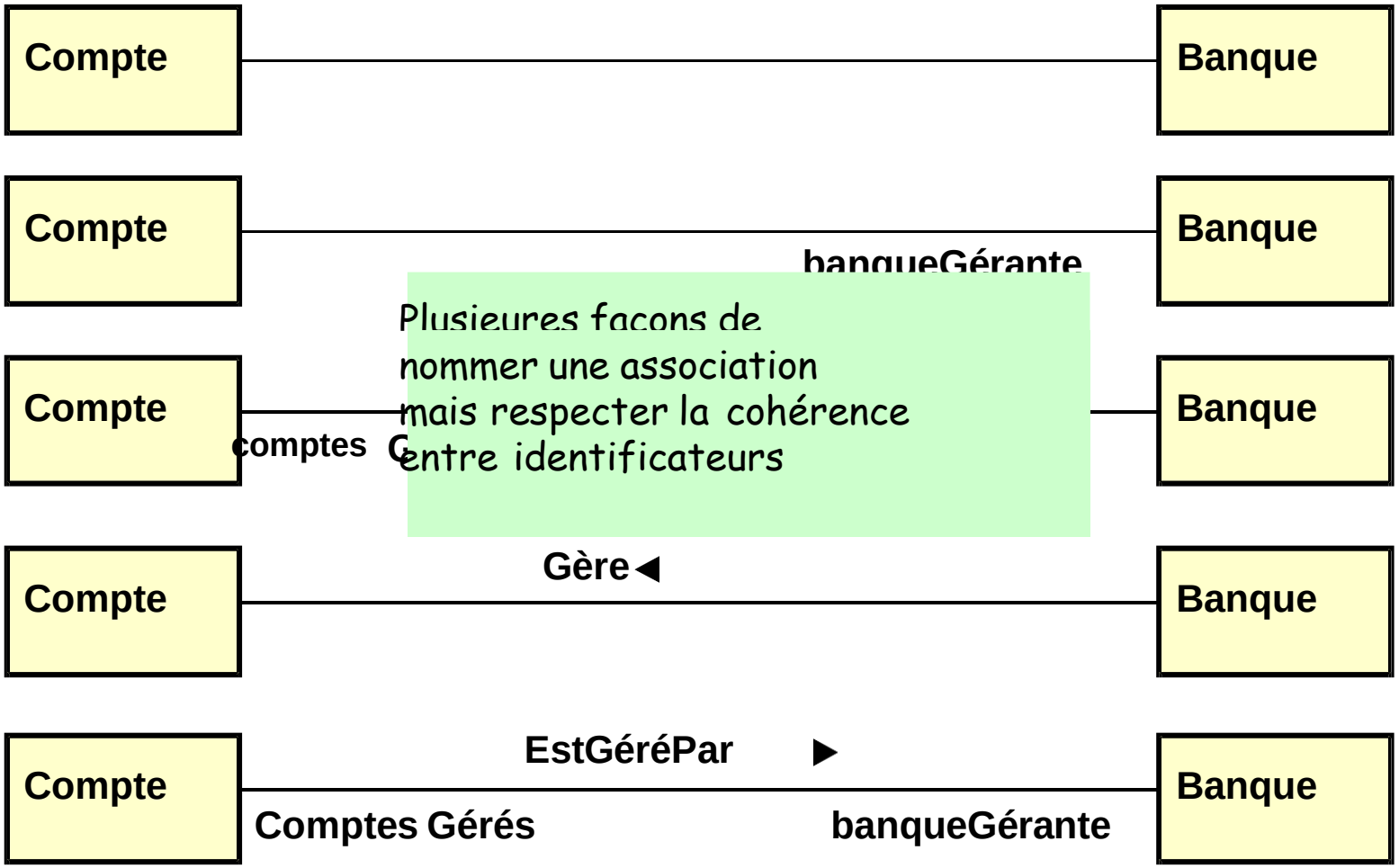


- Un lien est une instance d'association
- Une association décrit un ensemble de liens

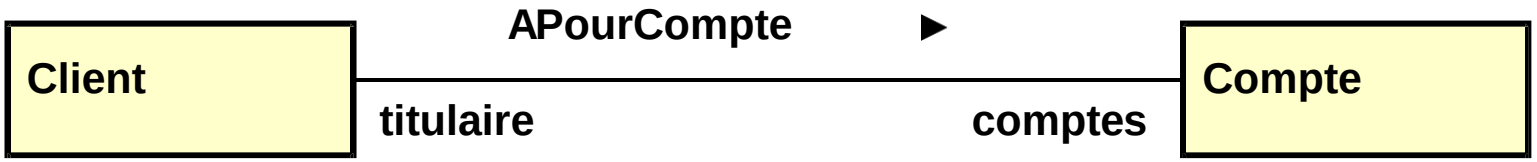
- Des liens peuvent être ajoutés ou détruits pendant l'exécution, (ce n'est pas le cas des associations)

Le terme "relation" ne fait pas partie du vocabulaire UML

Nommer les associations



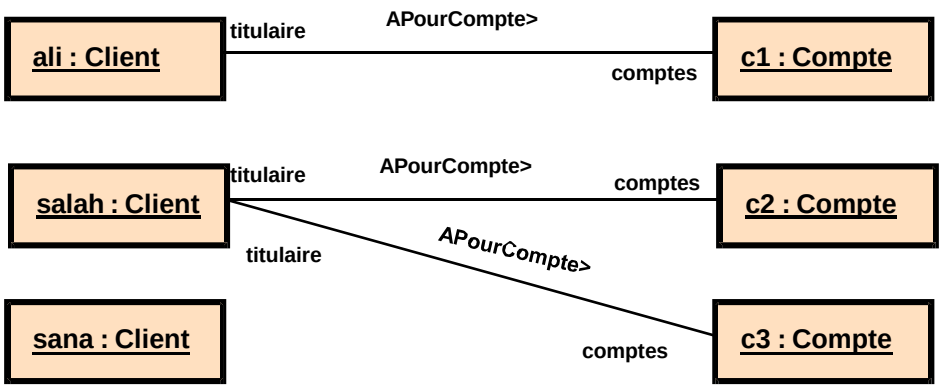
Utiliser les rôles pour «naviguer»



M1

M0

ali.comptes = {c1}
salah.comptes = {c2,c3}
sana.comptes = { }
c1.titulaire = ali
c2.titulaire = salah
c3.titulaire = salah

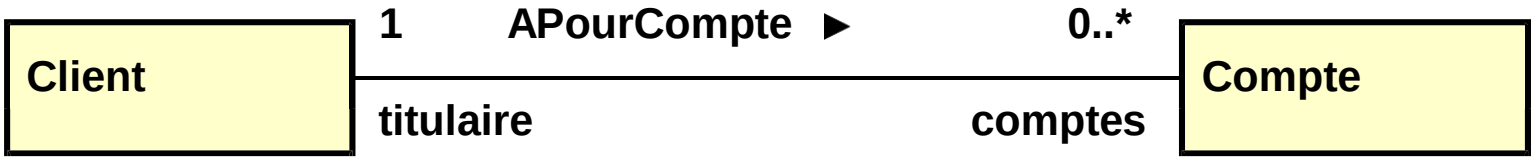


Nommer en priorité les rôles

Cardinalités d'une association

1..1 noté 1 : Un et un seul
0..1 : Zéro ou un
0..* noté * : De Zéro à n
1..* : De un à n
n..m : De n à m

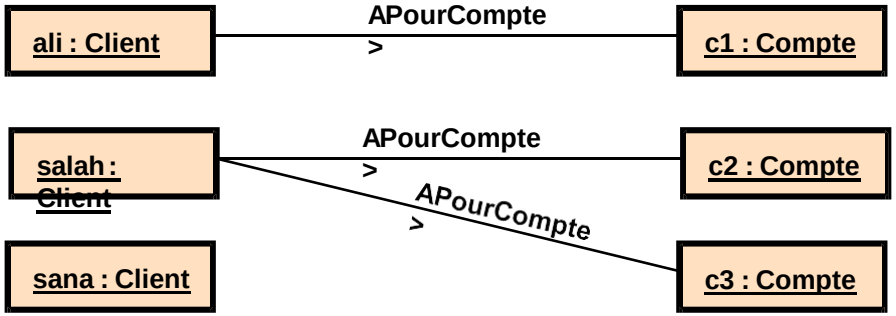
- Précise combien d'objets peuvent être liés àun seul objet source
- Cardinalité minimale et cardinalité maximale ($C_{min}..C_{max}$)



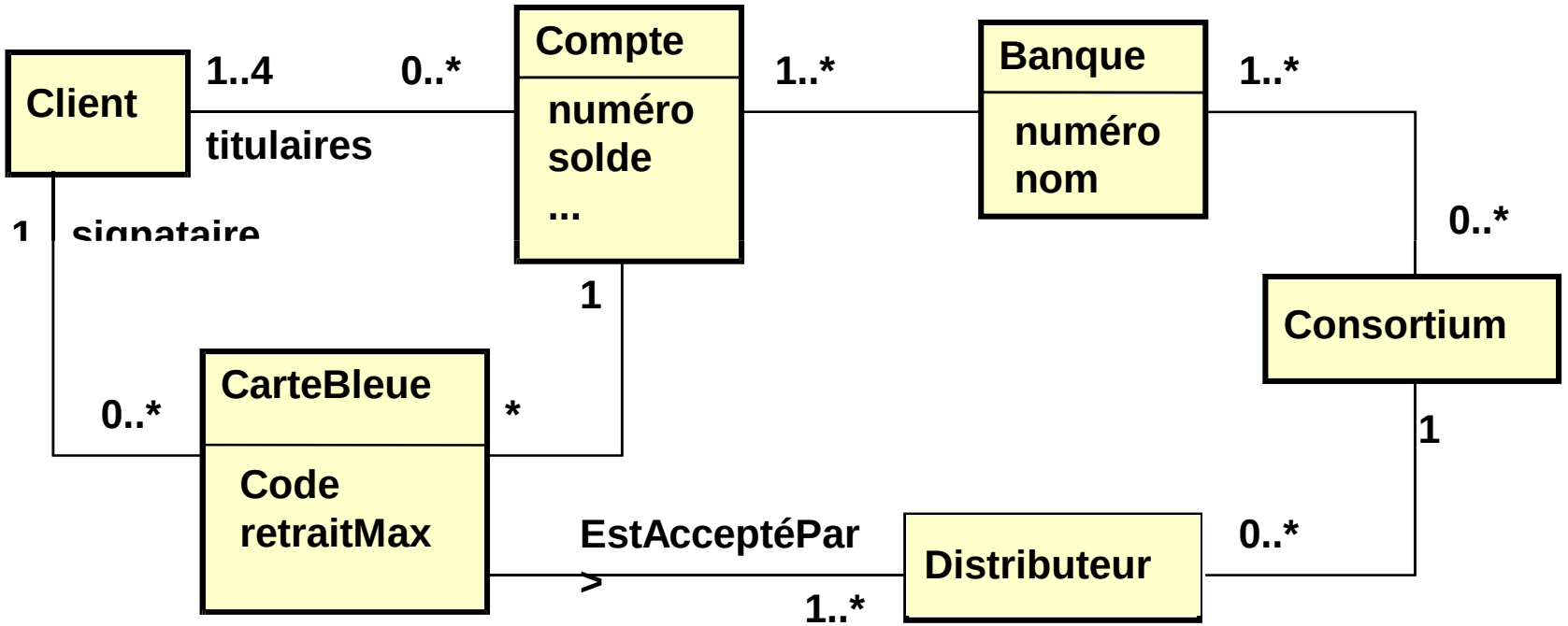
« Un client a 0 ou plusieurs comptes »
« Un compte a toujours 1 et 1 seul titulaire »

M1

M0



Exercice de lecture d'un diagramme de classes

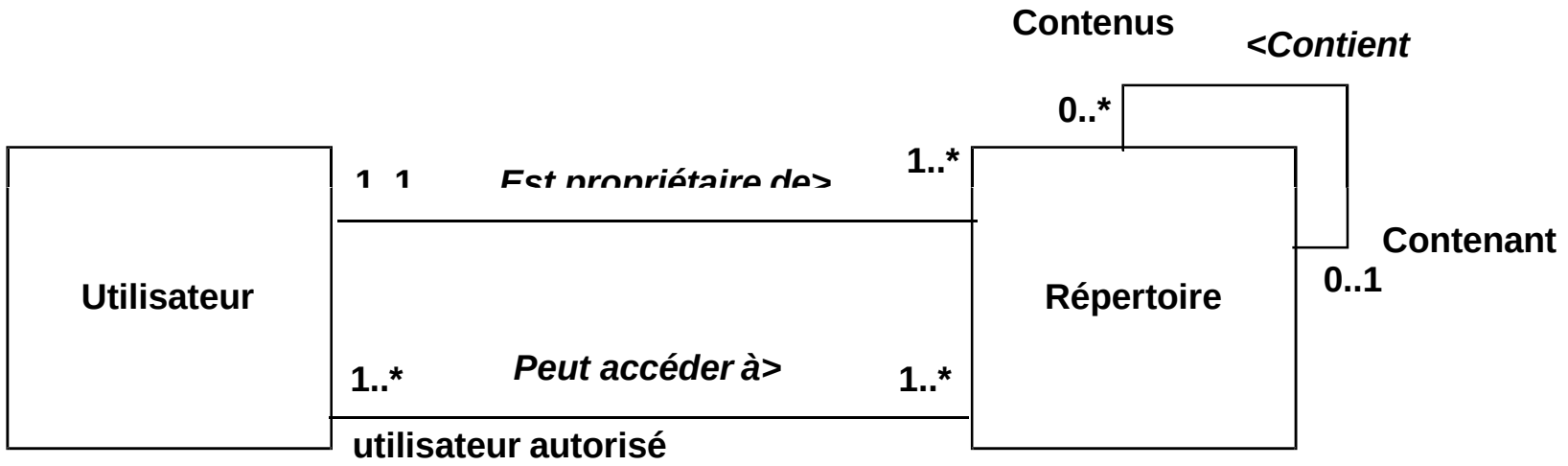


Exercice n°2

Description d'un système de fichiers

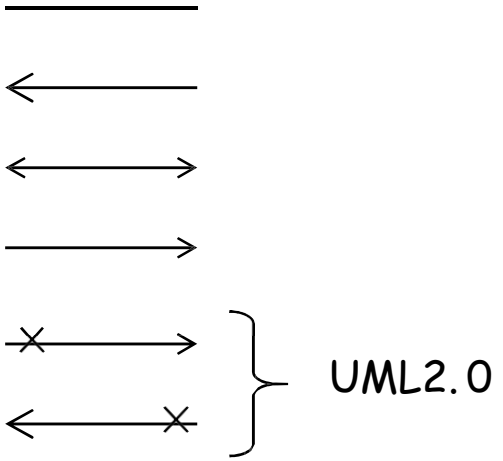
- Un utilisateur possède au moins un répertoire
- Un répertoire appartient à un et un seul utilisateur
- Un répertoire peut contenir d 'autres répertoires
- Un utilisateur peut accéder à au moins un répertoire
- Un répertoire peut être accédé par au moins un utilisateur

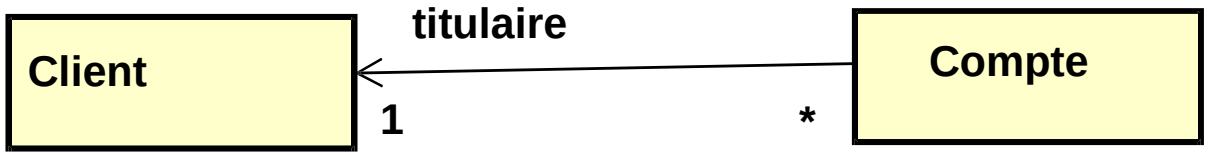
Exercice n°2 (Solution)



Navigation

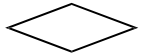
Association unidirectionnelle
On ne peut naviguer que dans un sens





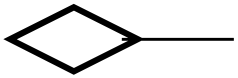
En cas de doute, ne pas mettre de flèche !!!
Son utilisation est surtout pour les modèles logiques et physiques

Cas particulier d'associations
Agrégation/composition

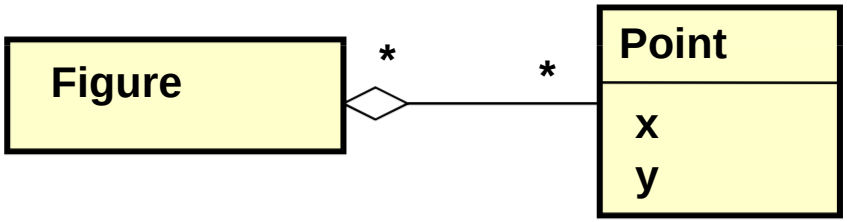


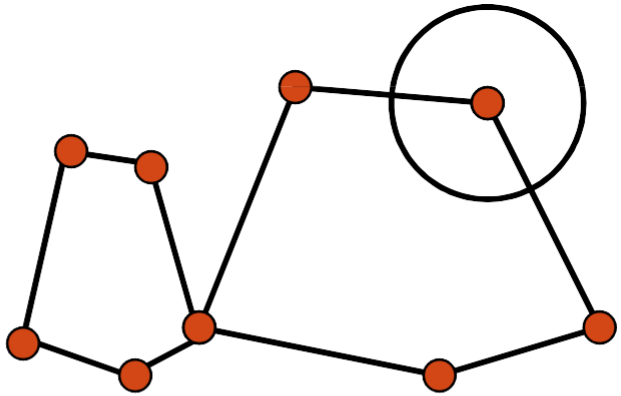

- Relation asymétrique, transitive (*Relation de subordination*)
- Agrégat / agrégé : durée de vie des agrégés indépendante de l'agrégat
- Composite / composant : durée de vie des composants dépendante du composite / conteneur - on parle d'embarquement

Agrégation



Agrégation =
cas particulier d'association
+ contraintes décrivant la notion d'appartenance...?



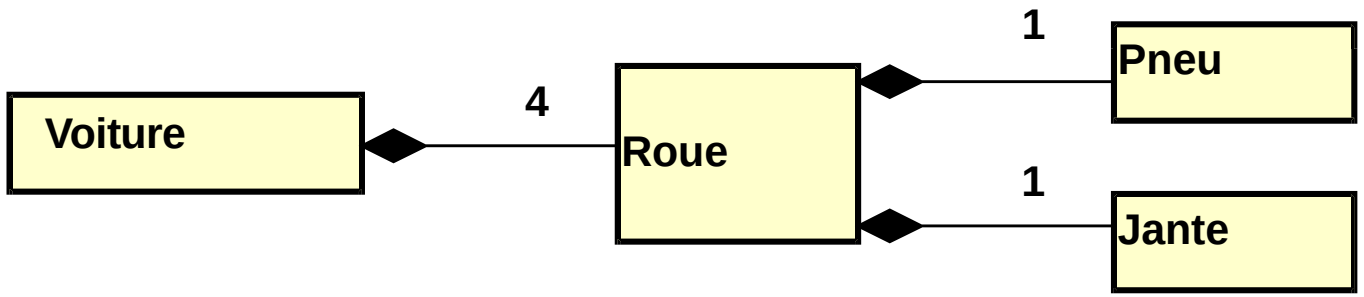


Appartenance faible =
➢ Partage possible du composant avec d'autres Agrégat/Eléments Agrégés
➢ Une instance agrégée peut exister sans son agrégat et inversement

Utiliser avec précautions pendant l'analyse (ou ne pas utiliser...)

Composition

Notion intuitive de "composants" et de "composites" ("Conteneur")



composition =

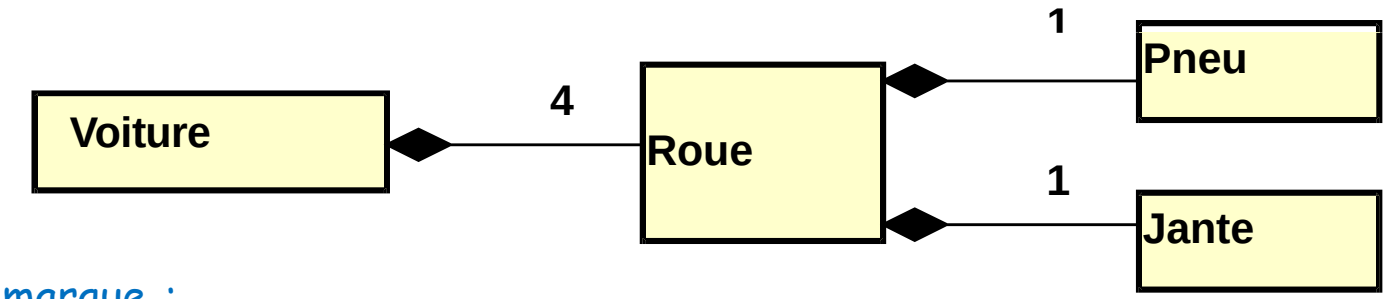
cas particulier d'association

\+ contraintes décrivant la notion de "composant"...

Composition

Contraintes liées à la composition :

1. Un objet *composant* ne peut être que dans 1 seul objet *composite*
2. Un objet *composant* n'existe pas sans son objet *composite*
3. Si un objet composite est détruit (copié), ses composants aussi



Remarque :

i. Agrégation avec relation d'appartenance forte et coïncidence des durées de vie
ii. Les composants peuvent être créés après le composite, mais après création ils ont la même durée de vie
iii. Les composants peuvent être enlevés avant la mort du composite

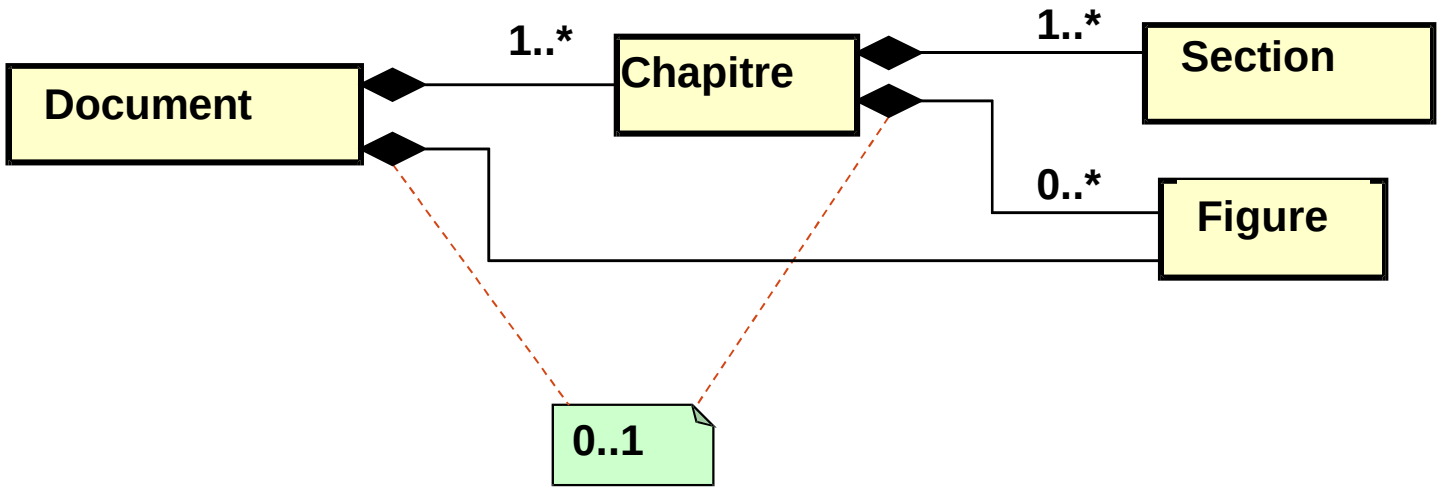
Dépend de la situation modélisée !

(Ex: vente de voitures vs. casse)

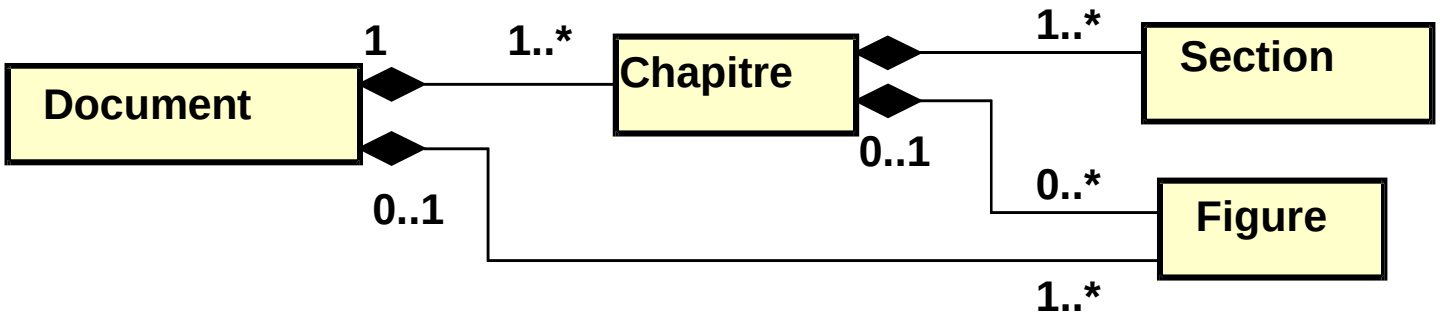
Composition

Contraintes liées à la composition :

1. Un objet *composant* ne peut être que dans 1 seul objet *composite*
2. Un objet *composant* n'existe pas sans son objet *composite*
3. Si un objet composite est détruit, ses composants aussi

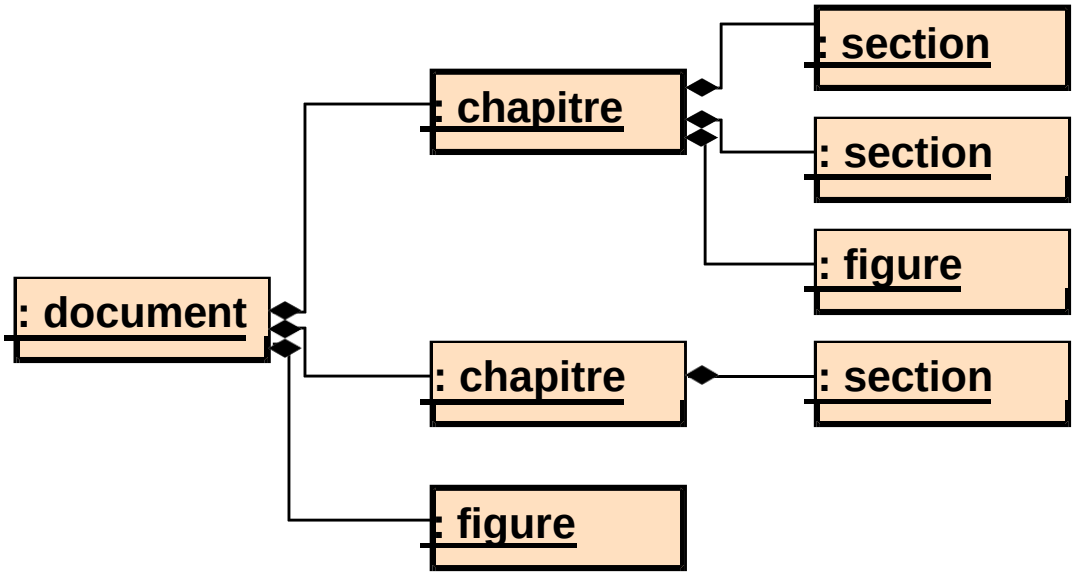


Composition



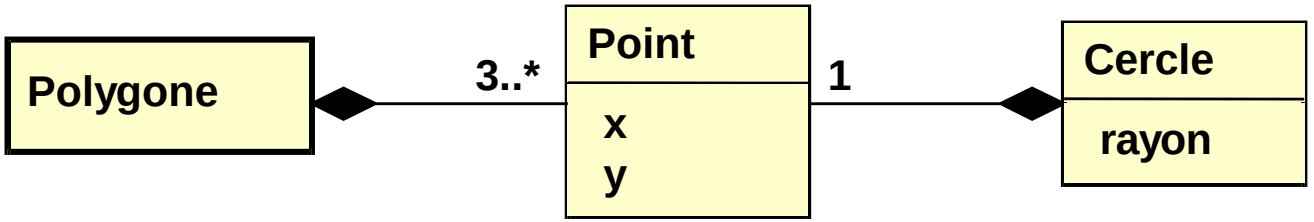
M1

M0



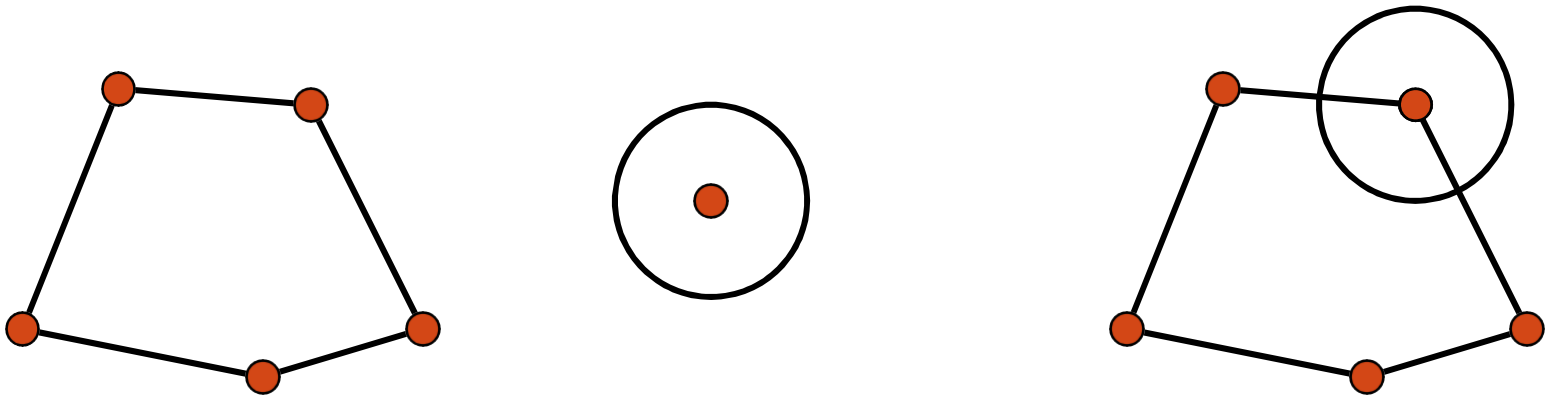
Contrainte :
le graphe d'objets forme un arbre (ou une forêt)

Composition

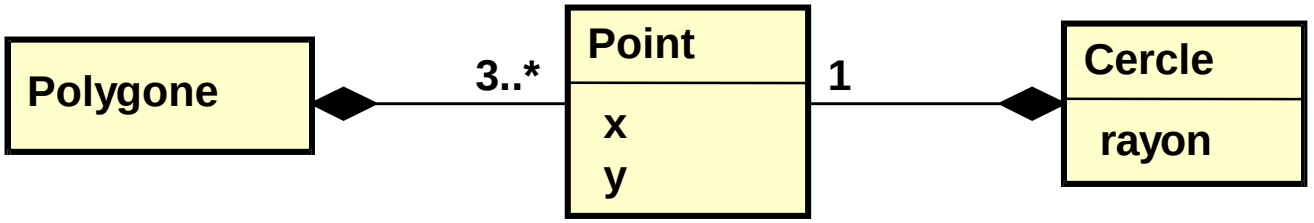


M1

M0

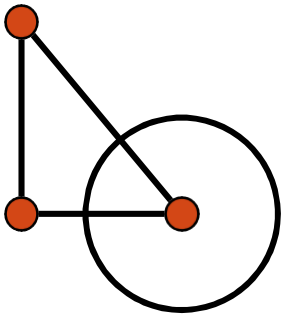


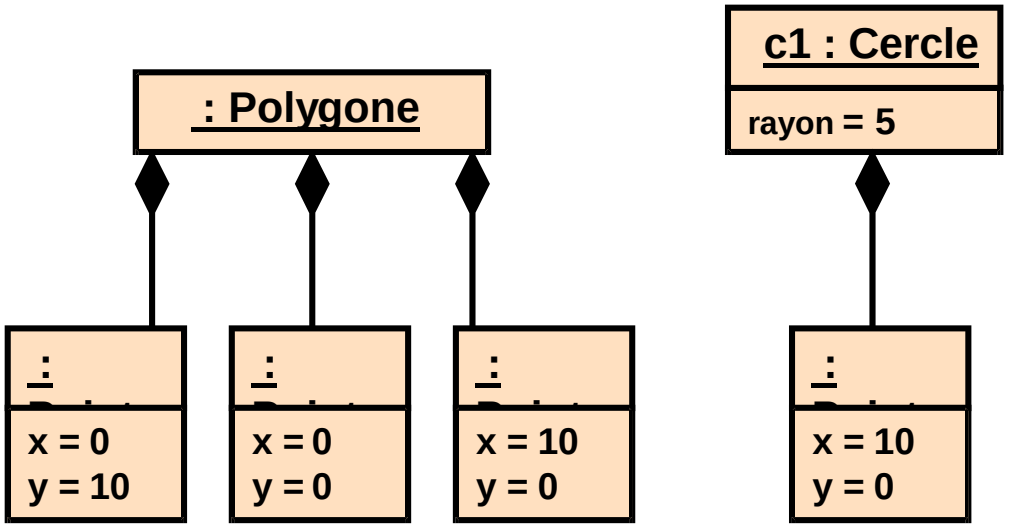
Composition



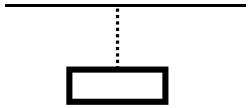
M1

M0



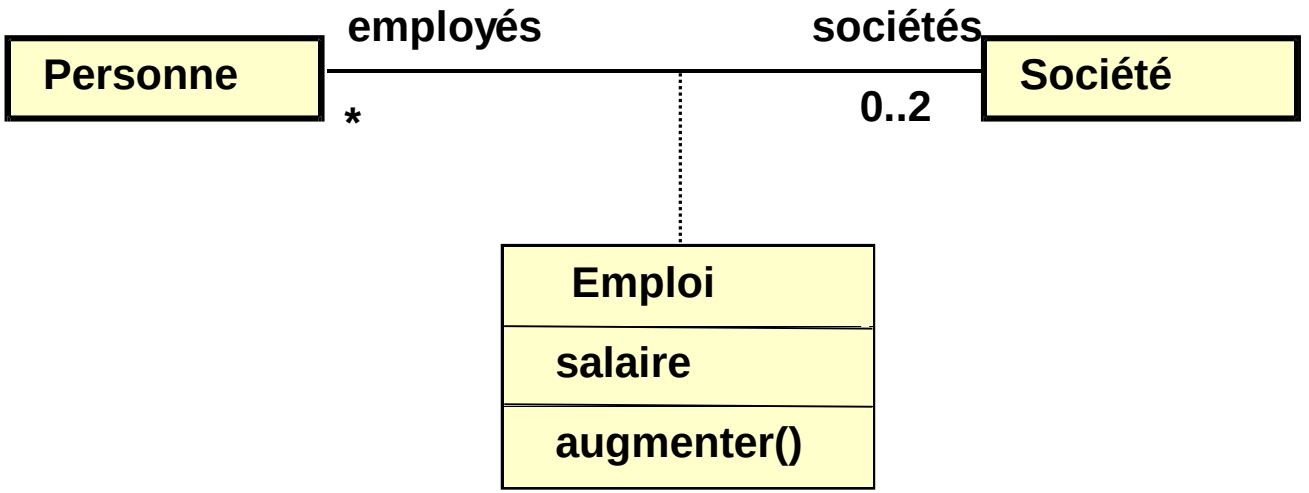


Classes associatives



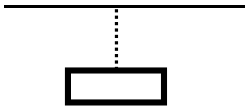
Pour associer des attributs et/ou des méthodes aux associations

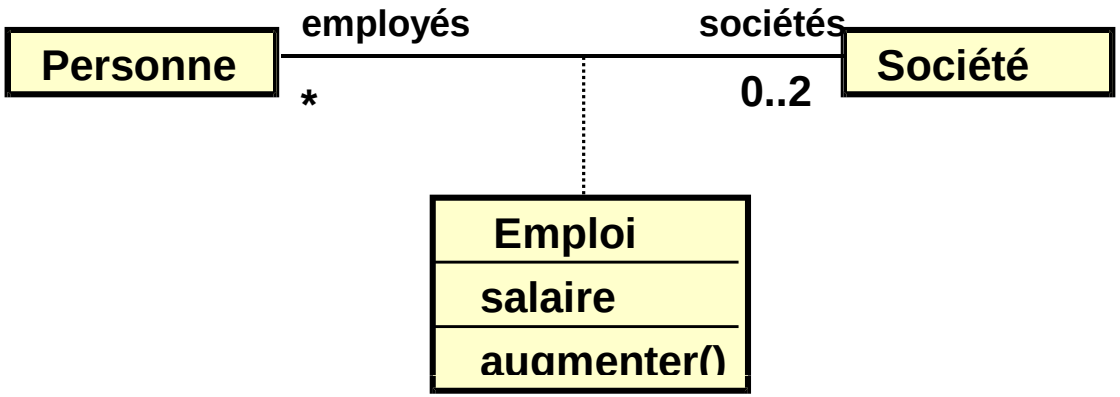
=> classes associatives

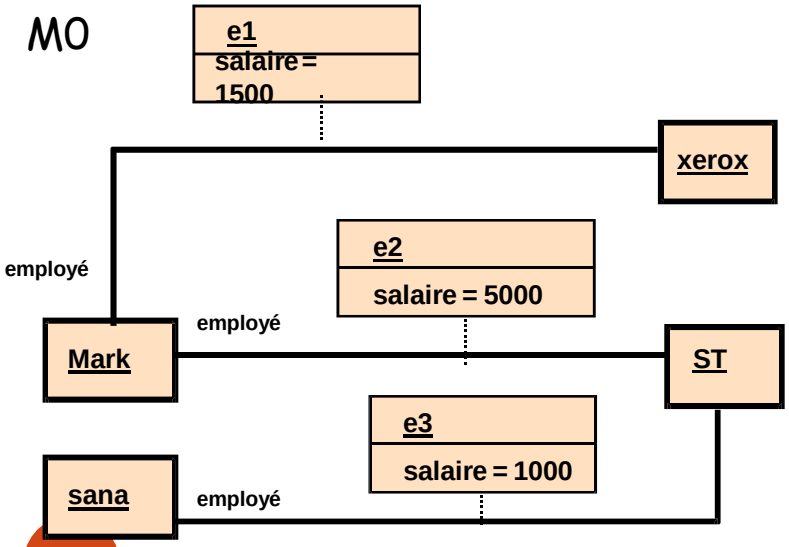


Le nom de la classe correspond au nom de l'association (problème: il faut choisir entre forme nominale et forme verbale)

Classes associatives



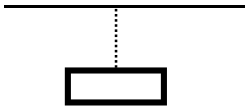




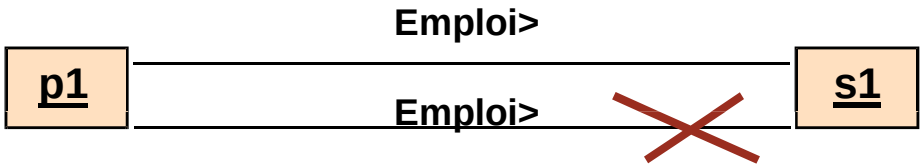
Le salaire est une information correspondant
- ni à une personne,
- ni à une société,

mais à un emploi (un couple personne-société).

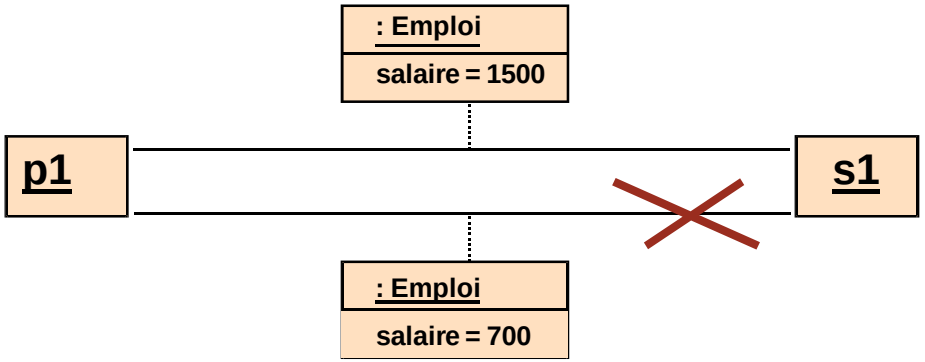
Classes associatives



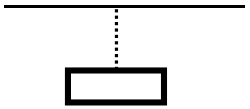
RAPPEL: Pour une association donnée, un couple d'objets ne peut être connectés que par un seul lien correspondant à cette association.
(sauf si l'association est décorée par {nonunique} en UML2.0)

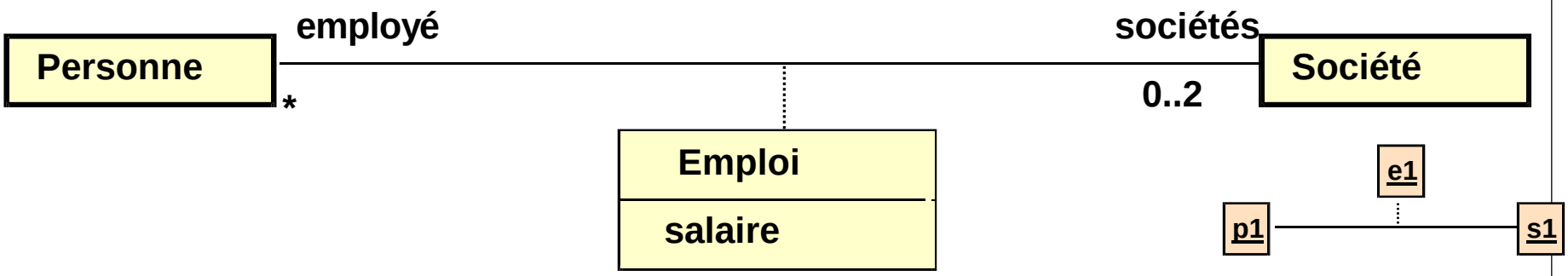


Cette contrainte reste vraie dans le cas où l'association est décrite à partir d'une classe associative.

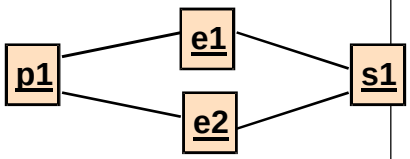


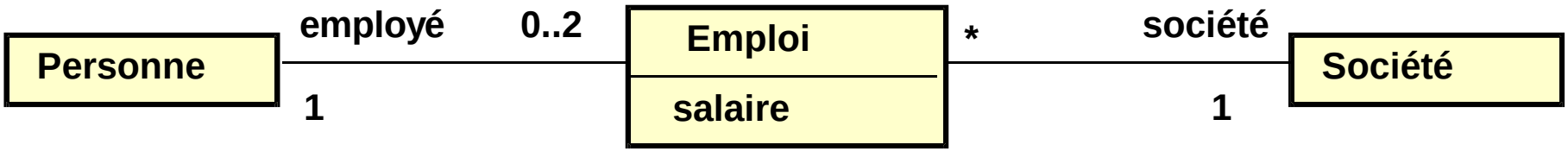
Classes associatives





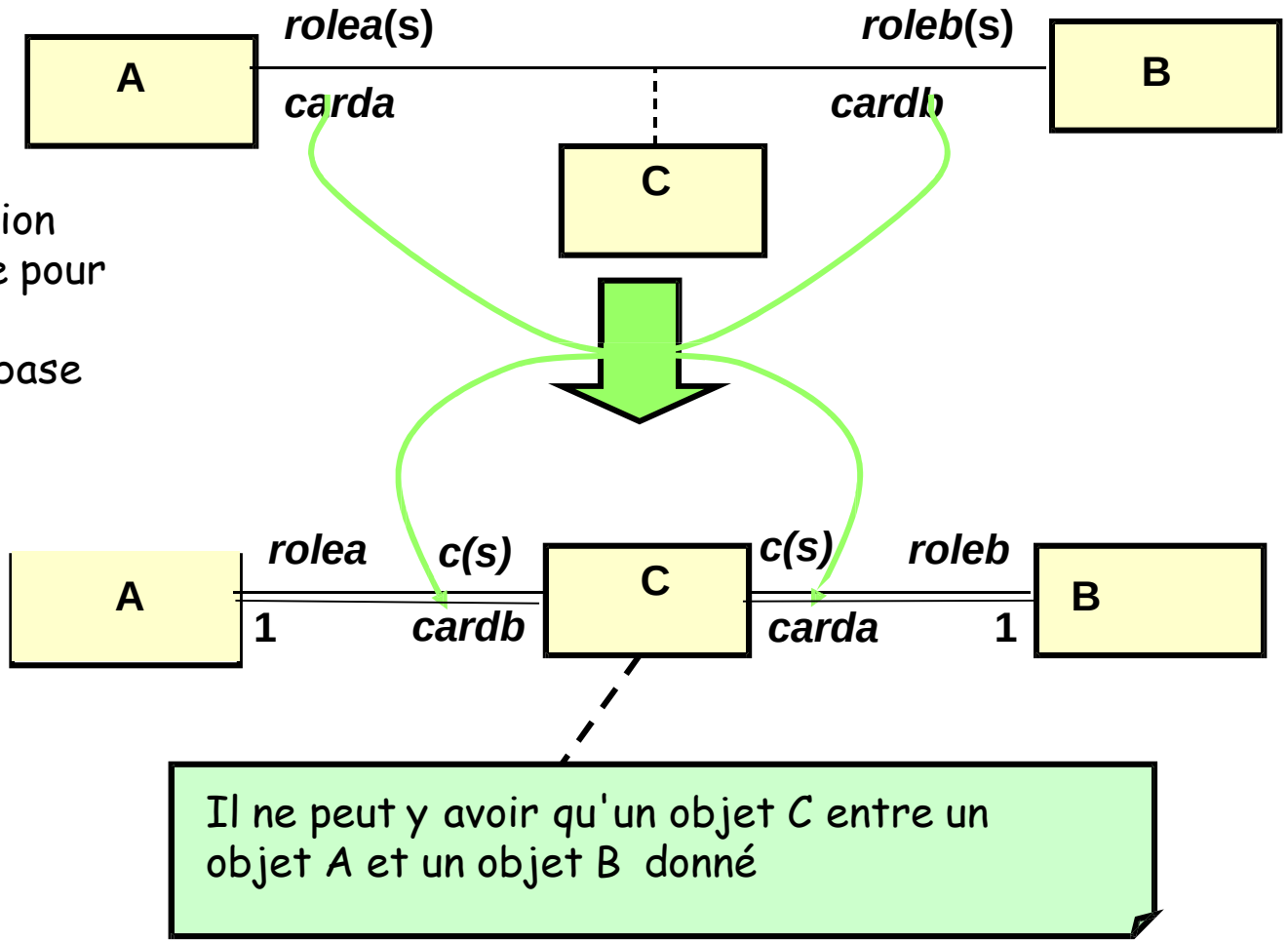
Ci-dessus, une personne peut avoir deux emplois, mais pas dans la même société



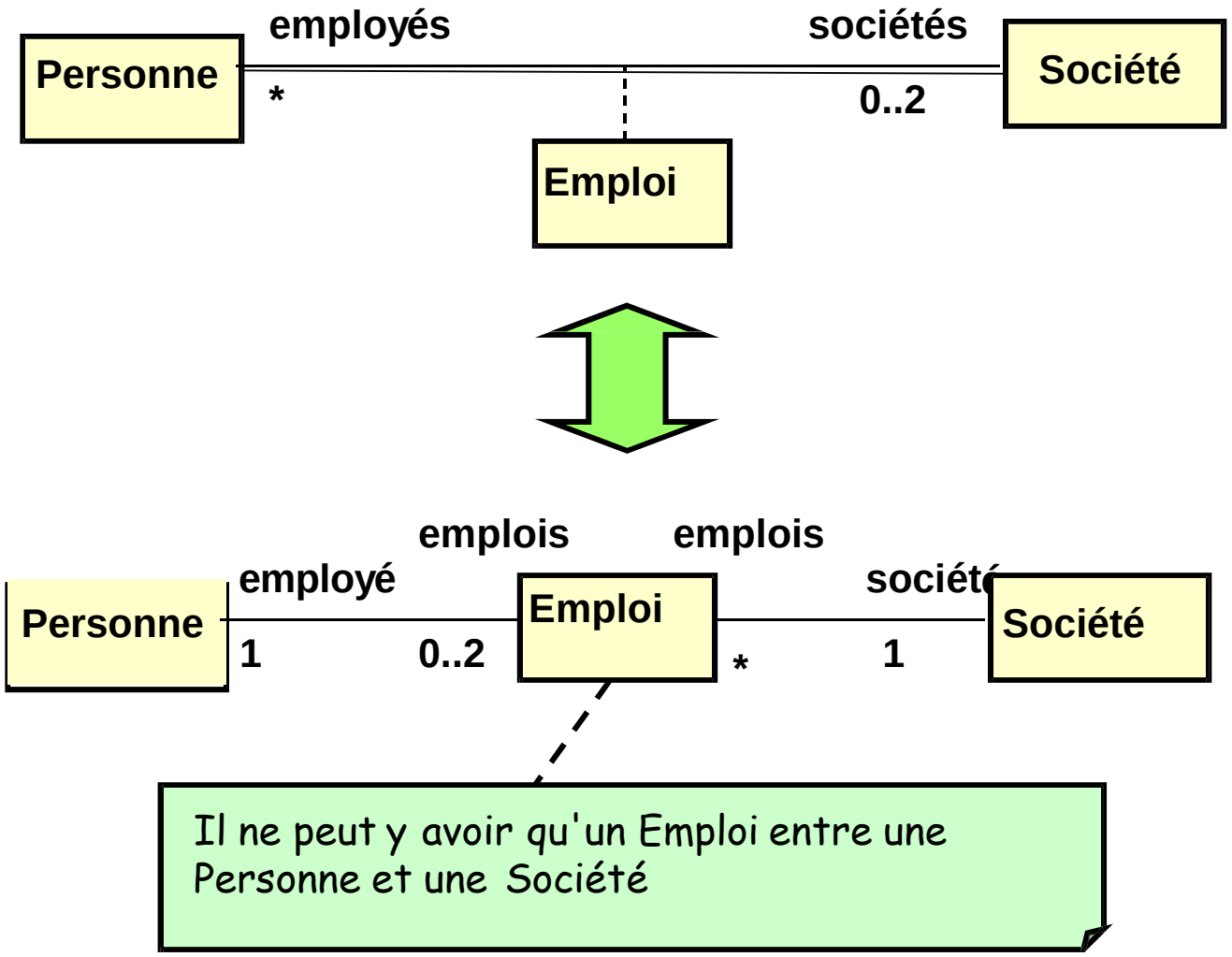


Ci-dessus, une personne peut avoir deux emplois dans la même société

Classes associatives : traduction

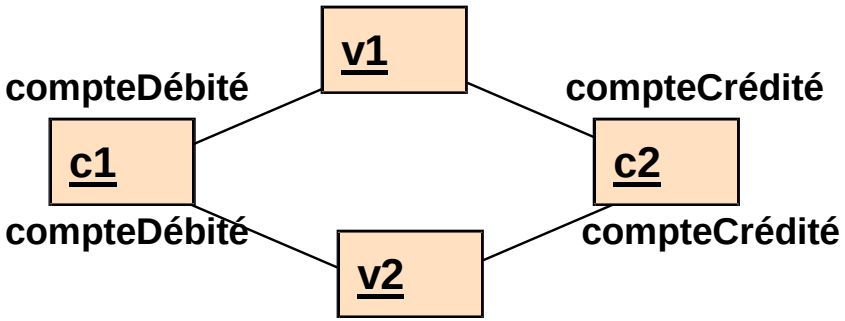


Classes associatives : traduction



Exemple 1

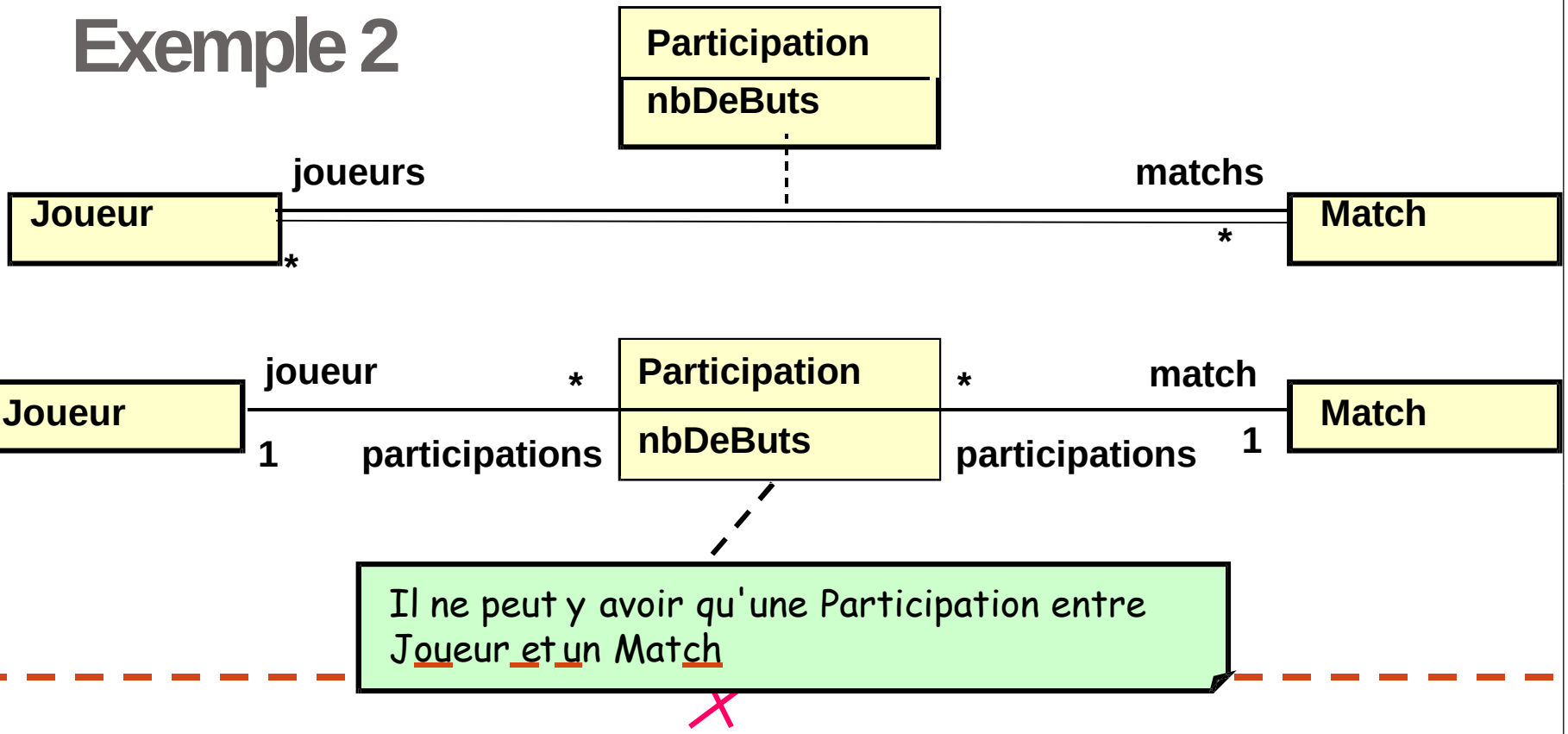
M0



Situation possible ?

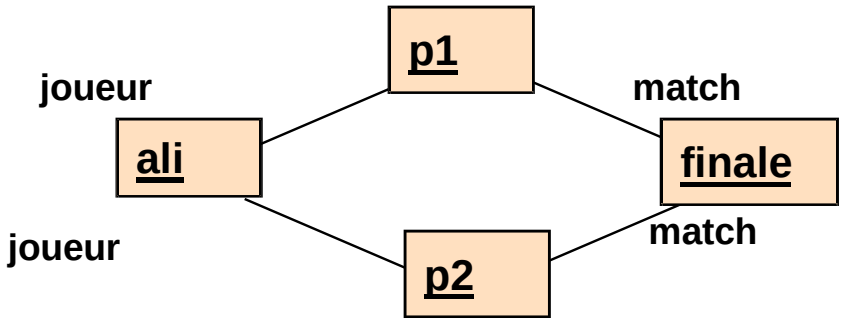
Peut-on avoir plusieurs virements entre deux comptes ?

Exemple 2



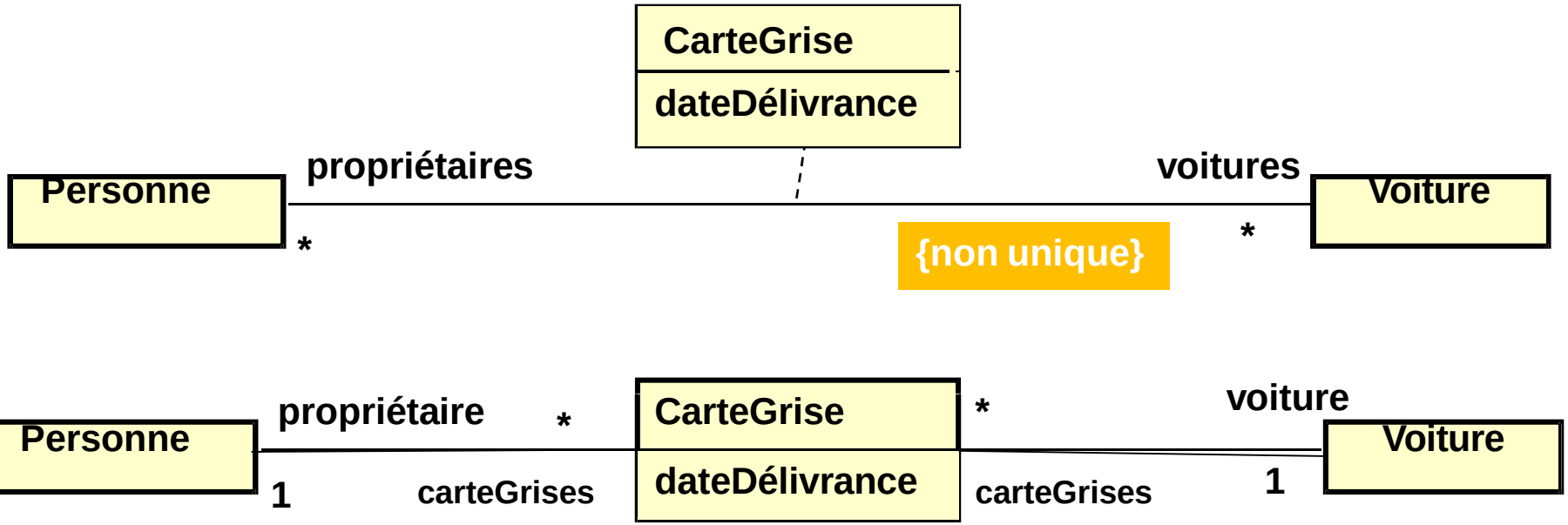
MO

Situation possible ?



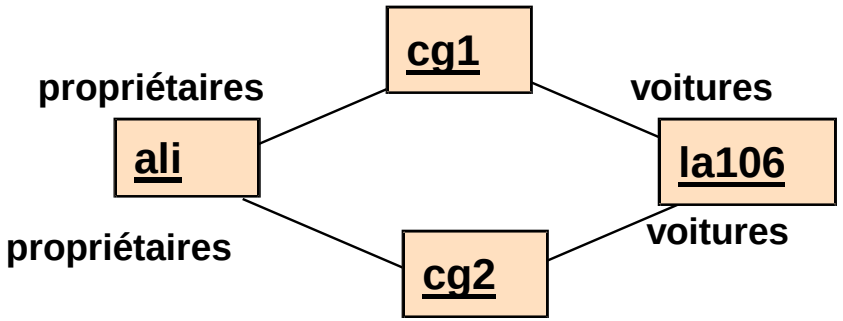
Un joueur peut il avoir plusieurs participations à un match donné ?

Exemple 3

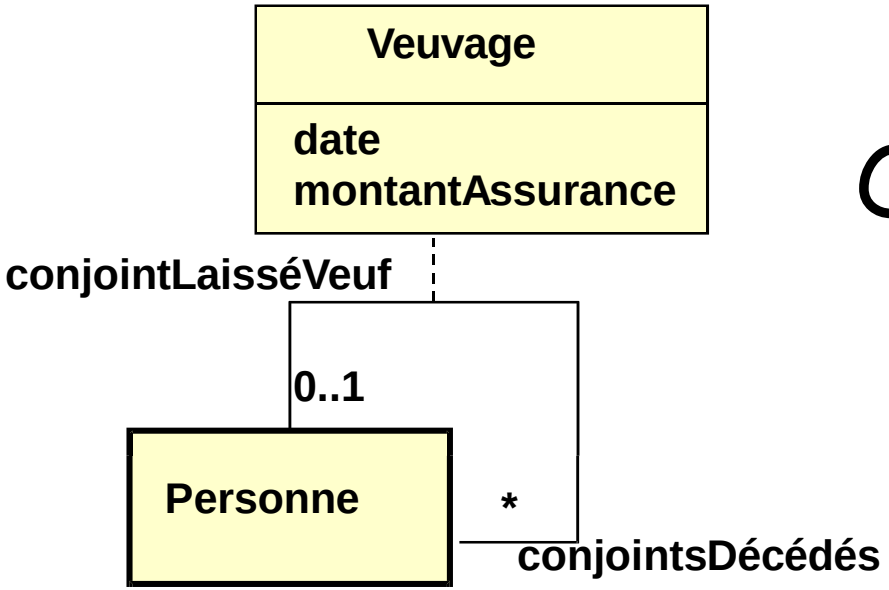


M0

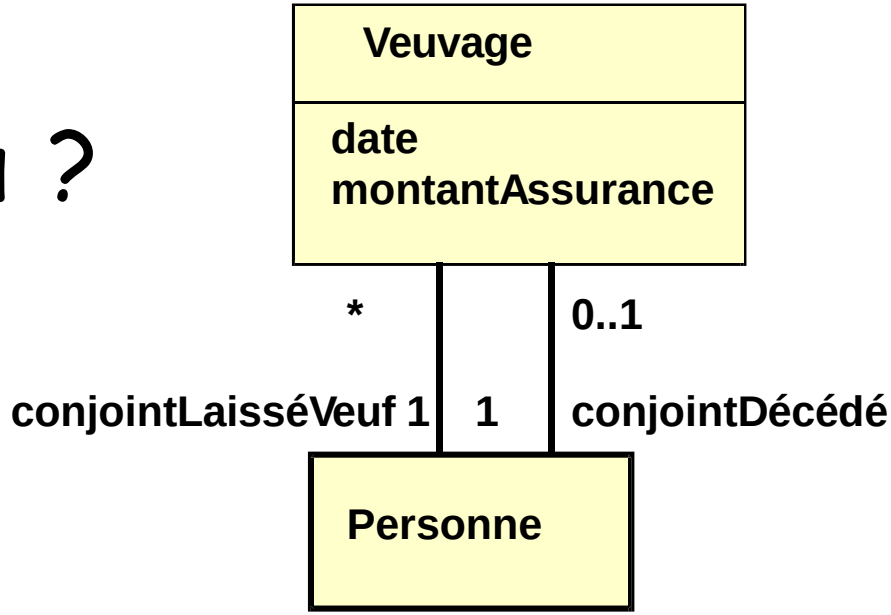
Situation possible ?



Exemple 4

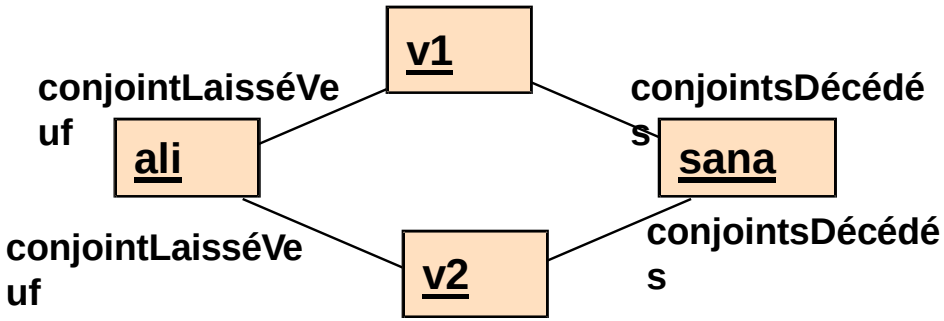


Ou ?



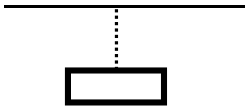
MO

Situation possible ?

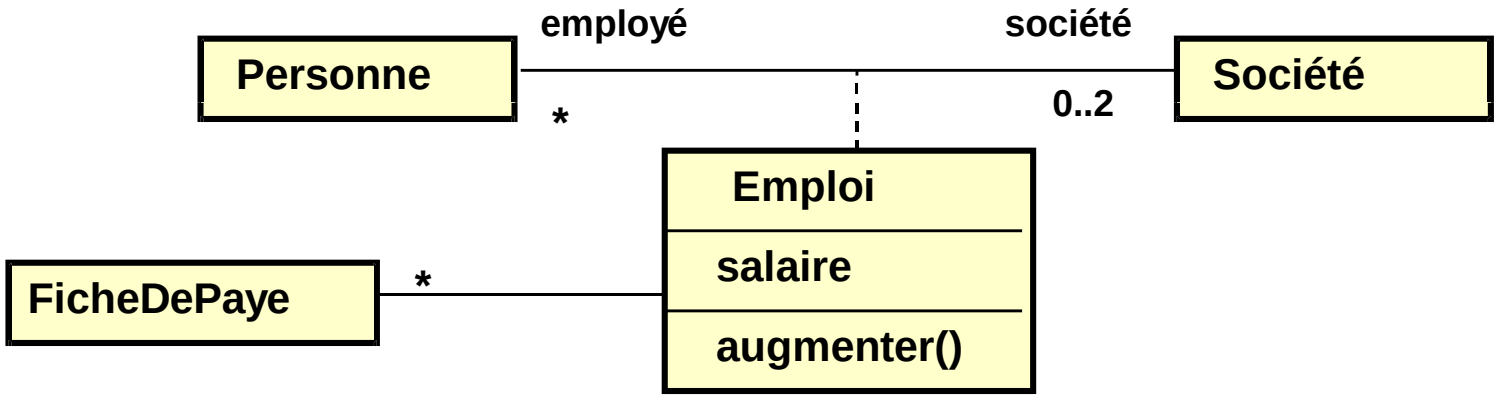


Peut on toucher deux fois l'assurance ?

Classes associatives

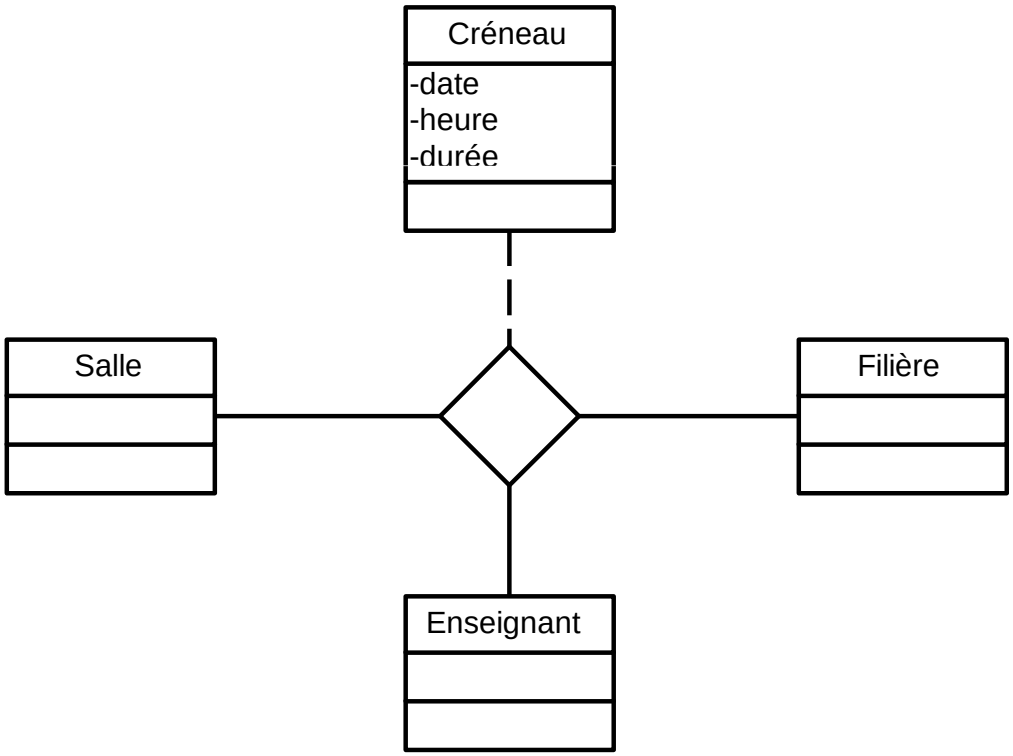


Les classes associatives sont des associations mais aussi des classes.
Elles ont donc les mêmes propriétés et peuvent par exemple être liées par des associations.



Associations n-aires

- Généralisation des classes associatives binaires : Une association peut relier une, deux ou plusieurs classes



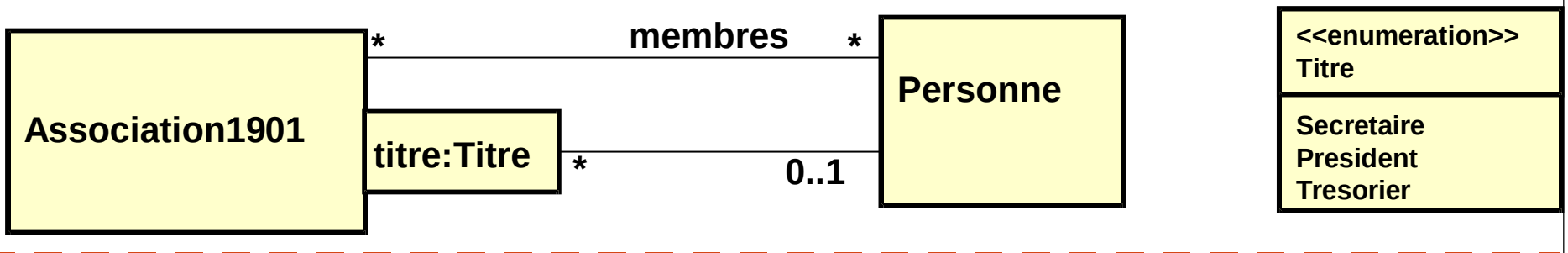
Associations qualifiées

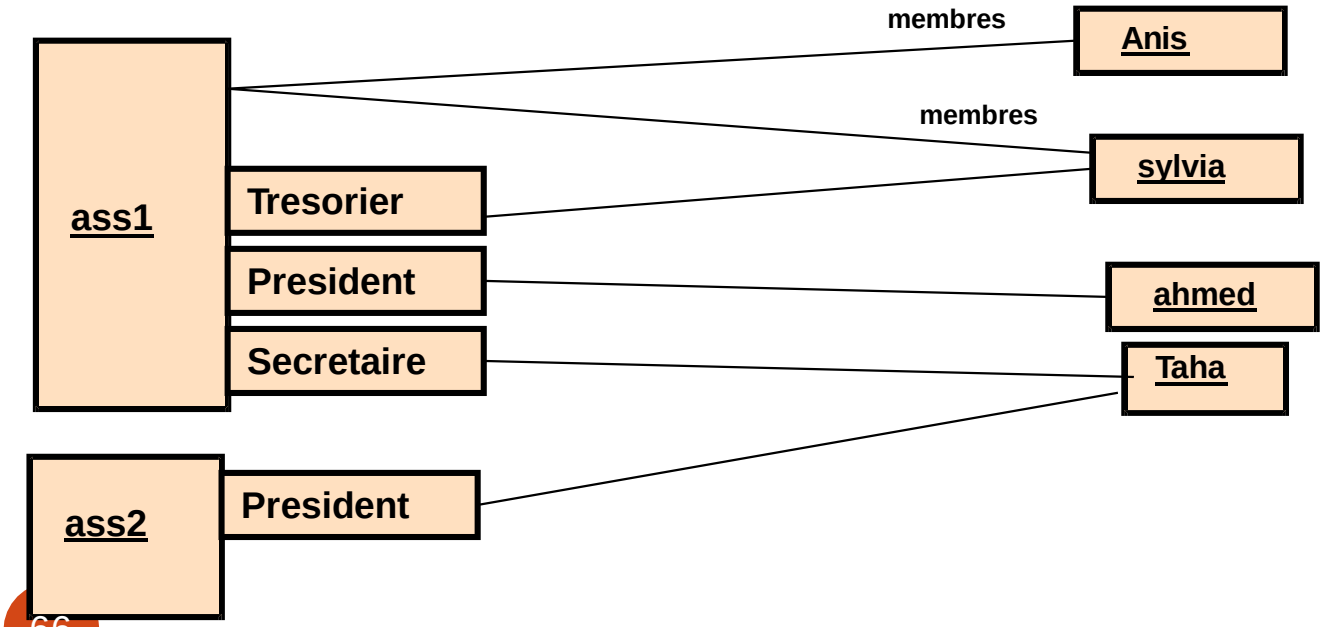
Un *qualifieur* est un attribut (ou un ensemble d'attributs) dont la valeur sert à déterminer l'ensemble des instances associées à une instance via une association.

"*Pour un répertoire, à un nom donné on associe qu'un fichier (ou 0 s'il existe aucun fichier de ce nom dans ce répertoire).*"

Correspond à la notion intuitive d'index absente ci-dessous

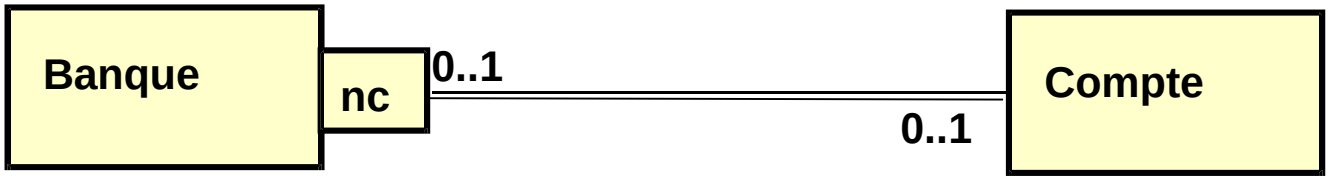
Exemple


Association1901
*
membres
*
Personne
titre:Titre
*
0..1
<<enumeration>>
Titre
Secretaire
President
Tresorier

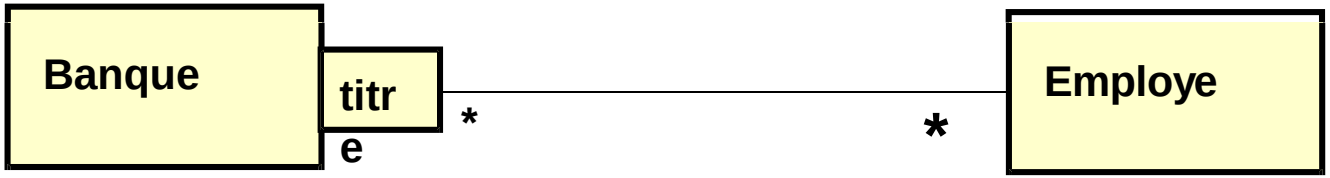

membres
Anis
membres
sylvia
ass1
Tresorier
President
Secretaire
ahmed
Taha
ass2
President

Cardinalité des Associations Qualifiées

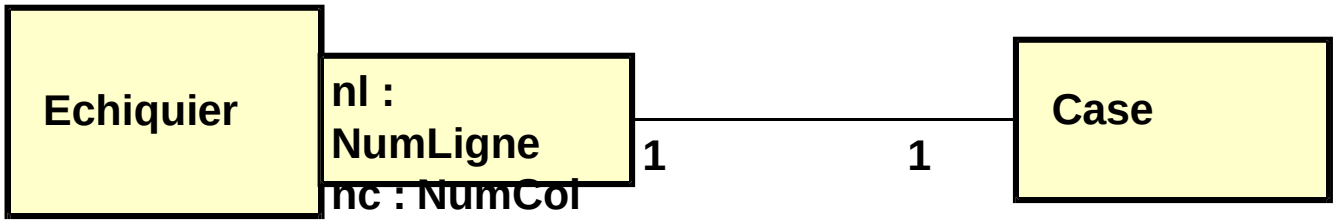
Cas classique: cardinalité 0..1



Cas plus rare: cardinalité * (pas de contrainte particulière exprimée)



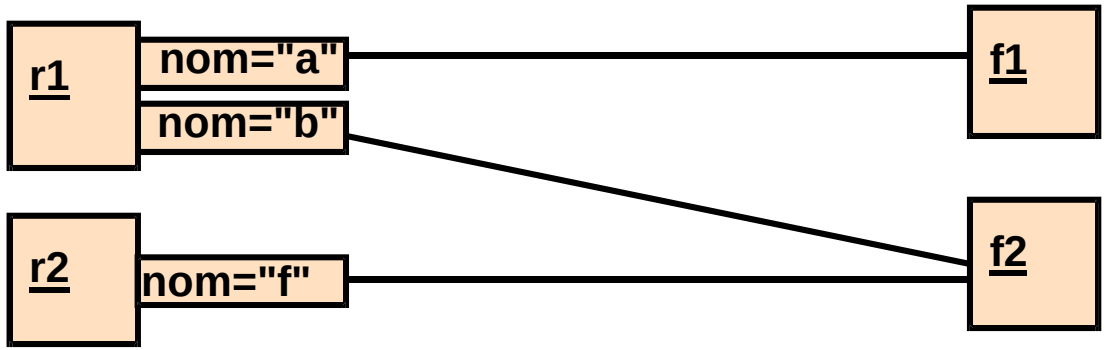
Cas plus rare: cardinalité 1 (généralement c'est une erreur)



0 comme cardinalité minimale, sauf si le domaine de l'attribut qualifieur est fini et toutes les valeurs ont une image.

Attributs de l'association

Les attributs qualifieurs sont des attributs de l'association, pas de la classe

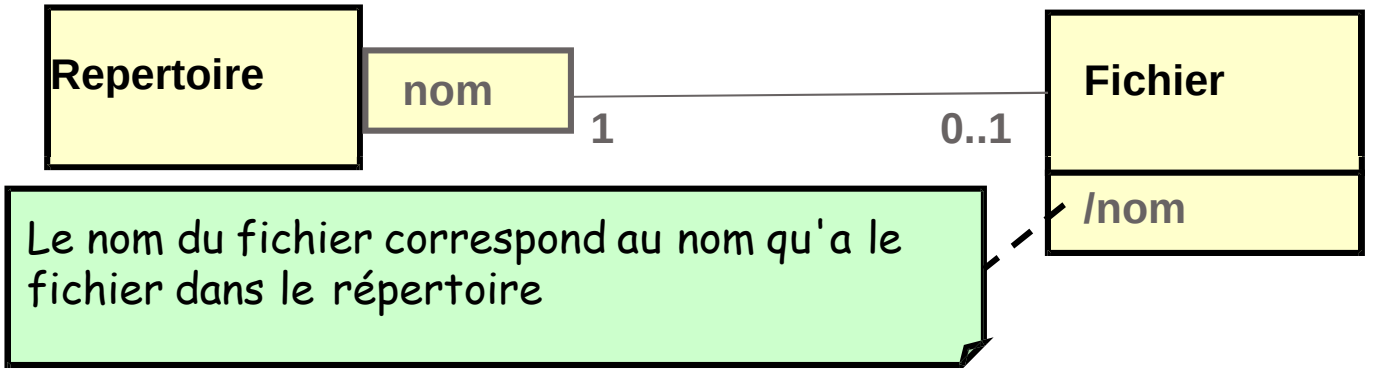


Exemple liens "hard" en Unix: un fichier peut correspondre à des noms différents dans des répertoires différents

Problème classique

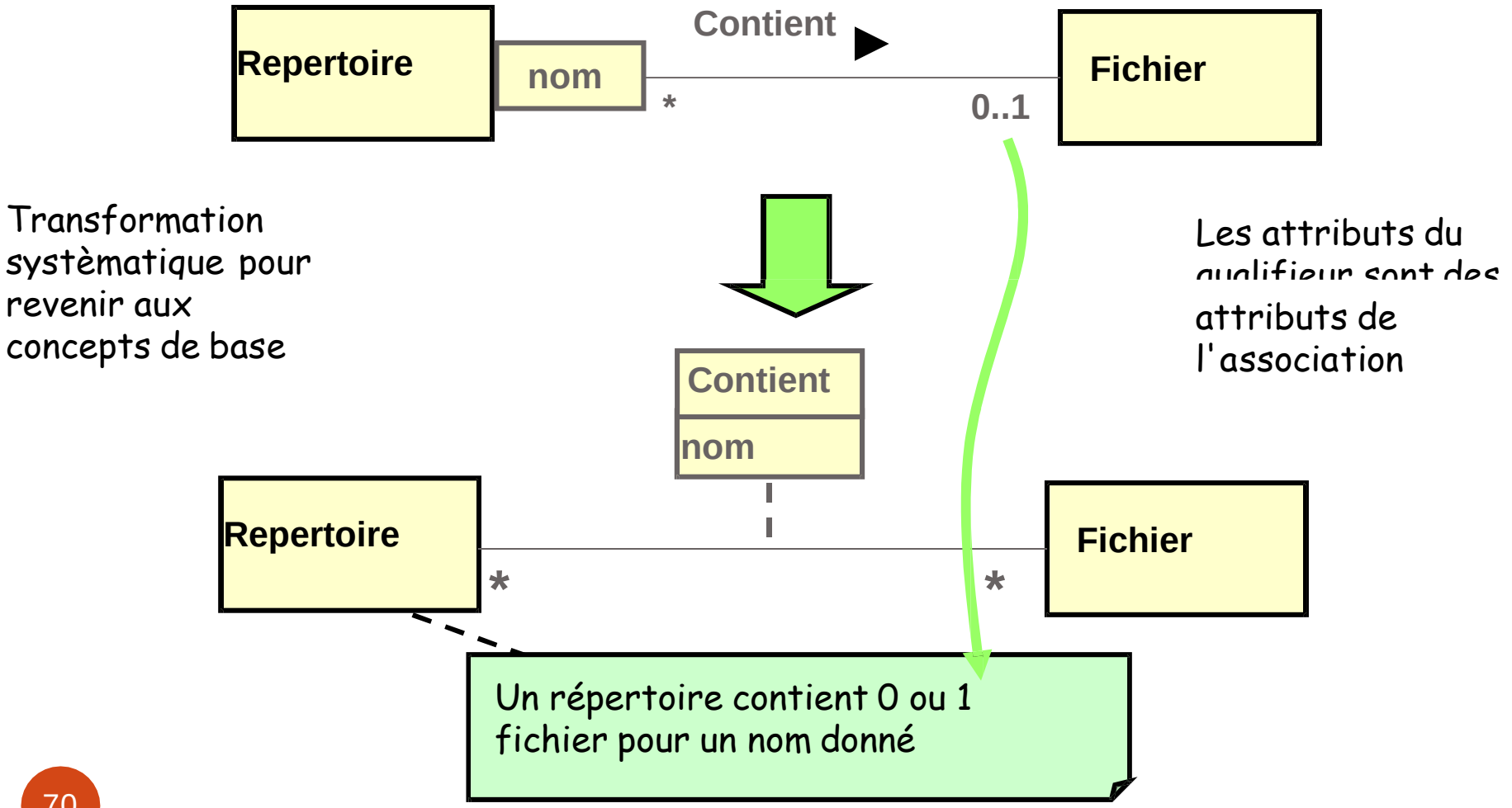
Souvent l'index est également un attribut de classe indexée

Solution correcte:

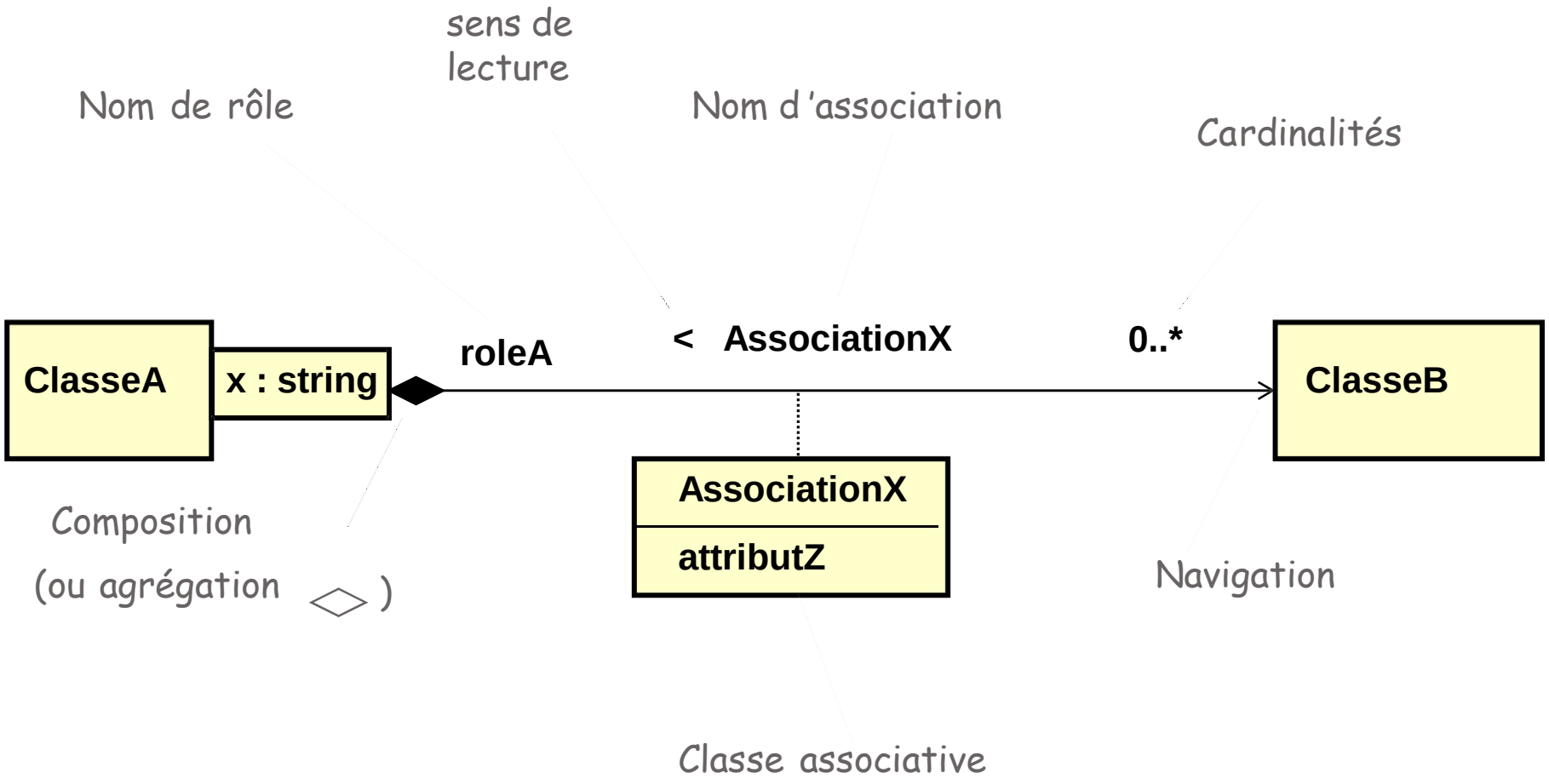


2 erreurs communes:

Classes associatives : traduction

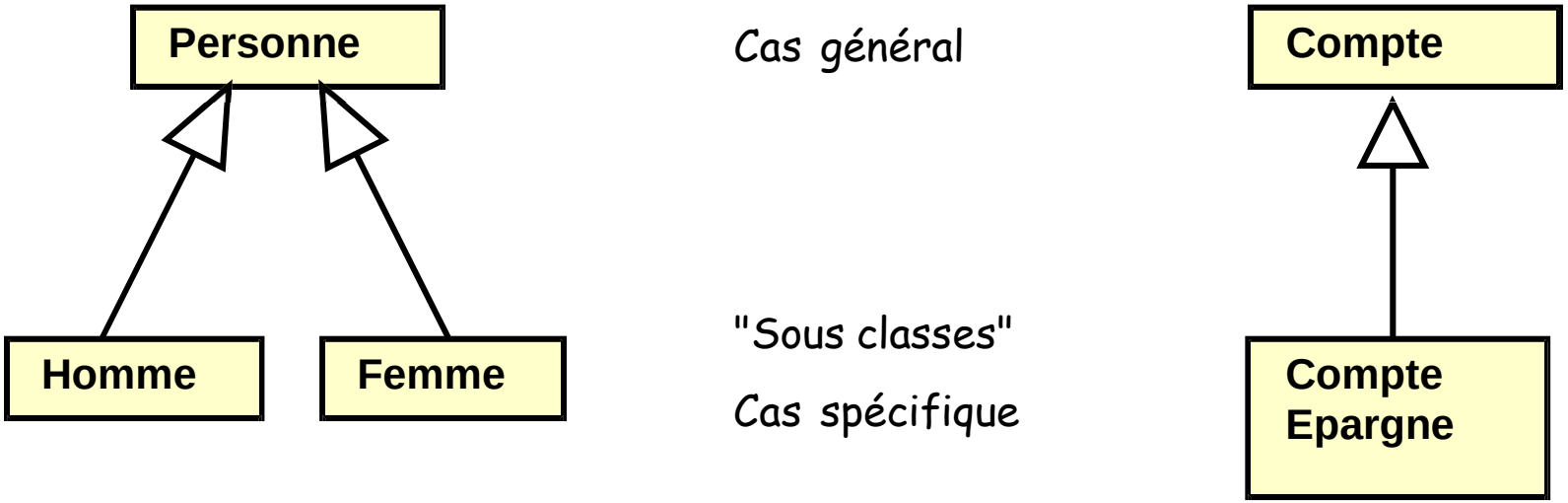


Synthèse sur les associations



Généralisation /Spécialisation

Une classe peut être la généralisation d'une ou plusieurs autres classes.
Ces classes sont alors des spécialisations de cette classe.



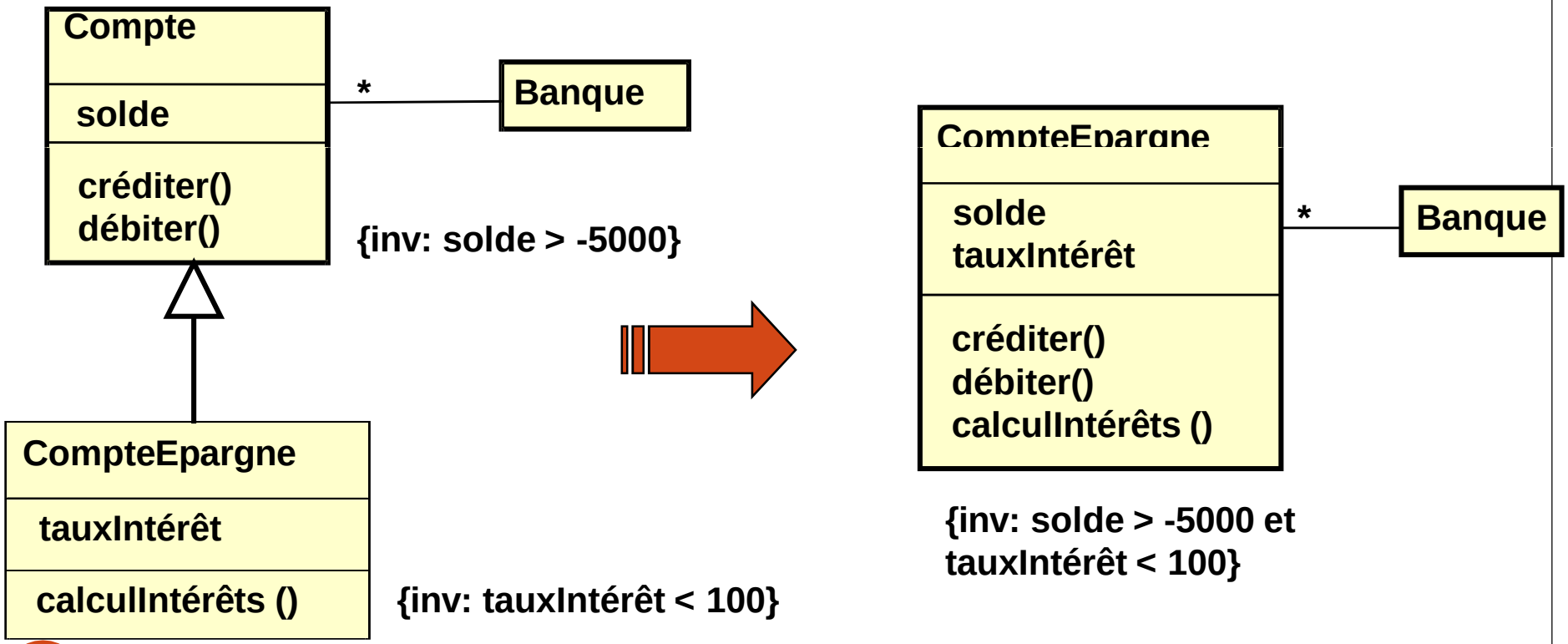
Deux points de vue lié (en UML) :
- relation d'héritage
- relation de sous-typage

Règles de généralisation (…)

- *La généralisation ne porte ni nom particulier ni valeur de multiplicité.*
- *La généralisation est une relation non réflexive: une classe ne peut pas dériver d'elle même*
- *La généralisation est une relation non symétrique: si une classe B dérive d'une classe A, alors la classe A ne peut pas dériver de la classe B.*
- *La généralisation est par contre une relation transitive: si C dérive d'une classe B qui dérive elle-même d'une classe A, alors C dérive également de A.*

Relation d'héritage

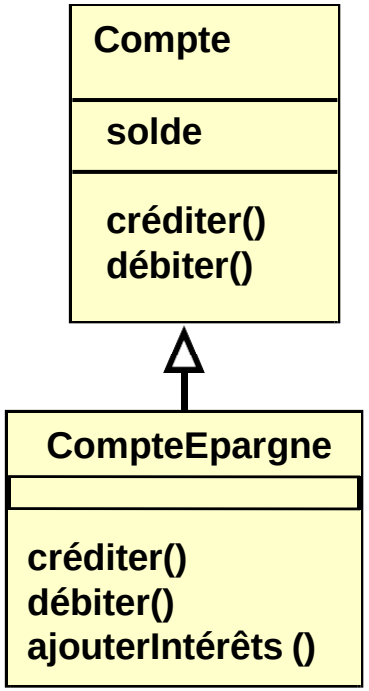
Les sous-classes « *héritent* » des propriétés des super-classes (attributs, méthodes, associations, contraintes)



Relation d'héritage et redéfinitions

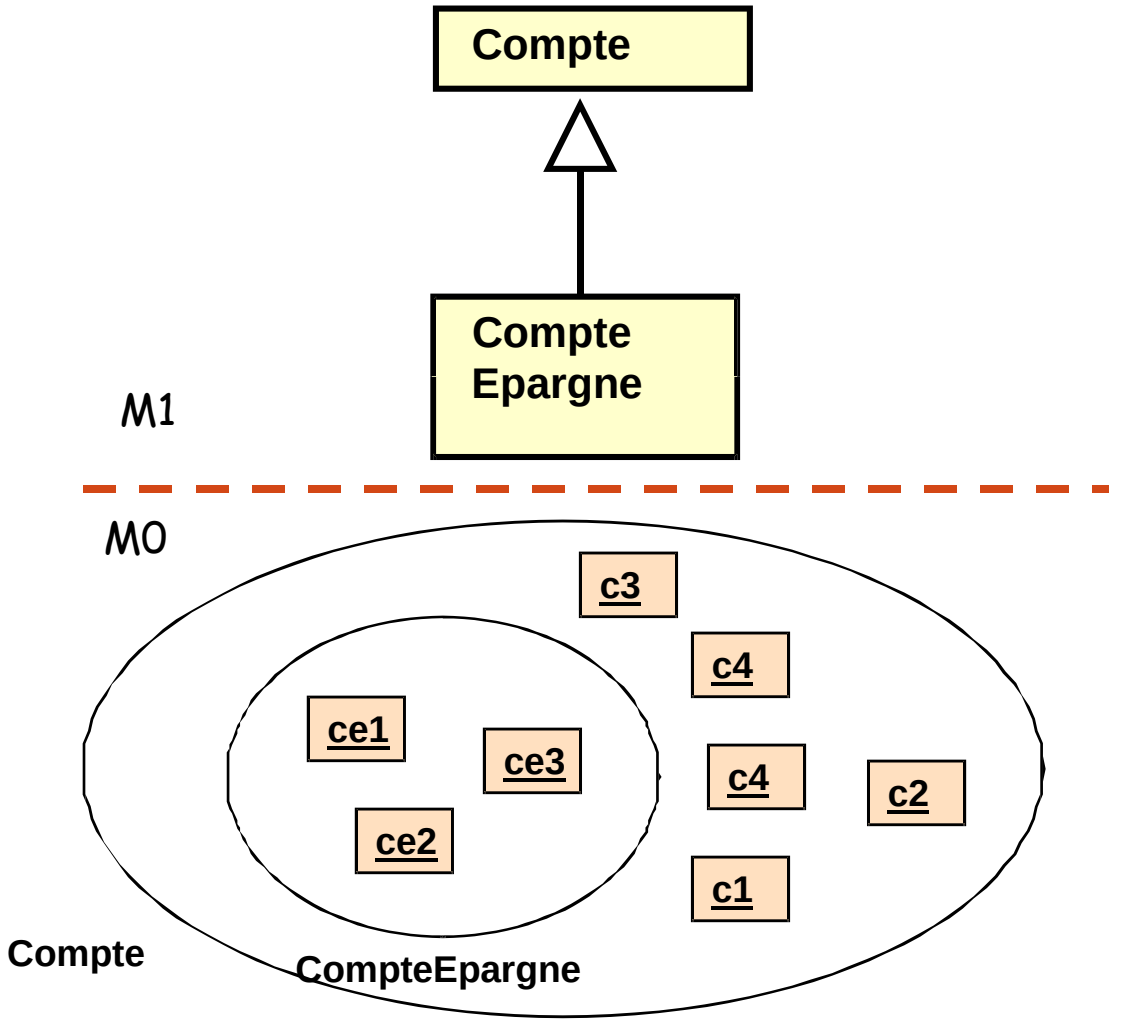
Une opération peut être "redéfinie" dans les sous-classes

Permet d'associer des méthodes spécifiques à chaque pour réaliser une même opération



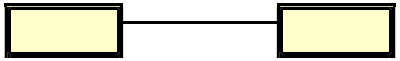
Relation de sous-typage, Vision ensembliste

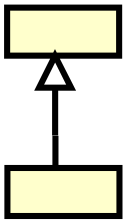
tout objet d'une sous-classe appartient également à la super-classe



Synthèse des concepts de base

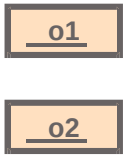
- Classe
  - attribut
  - opération
- Association
  - rôle
  - cardinalité
  - Agrég./Comp./Associative/Qualifiée
- Héritage

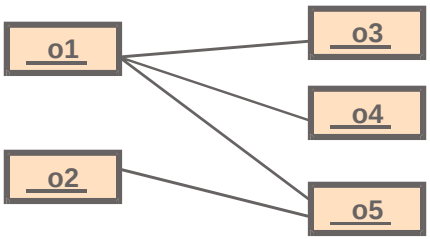


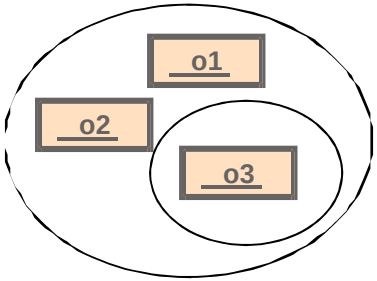


M1

M0



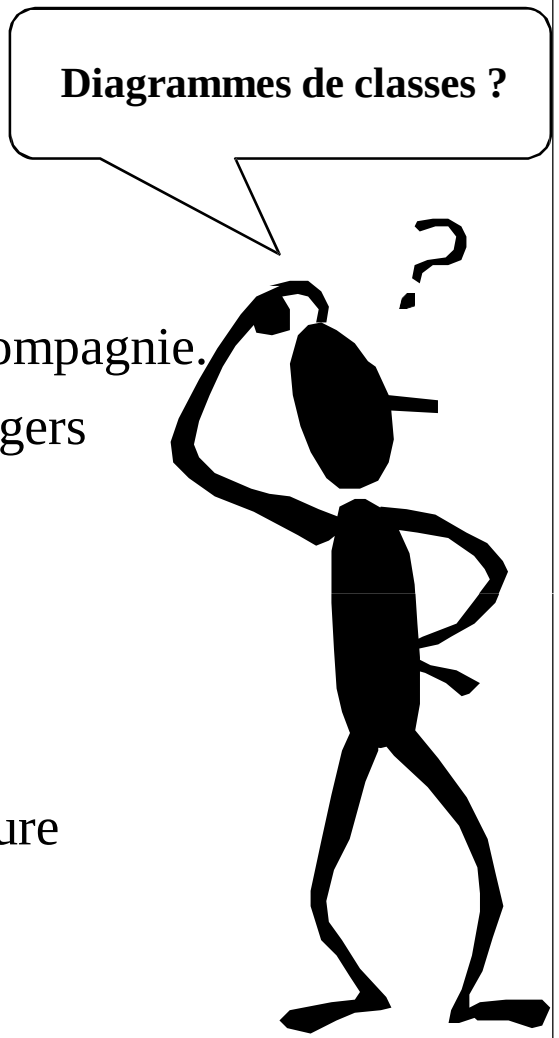




- Objet
- Lien
- Inclusion ensembliste

A vous de jouer

''Réservation de vols dans une agence de voyage''



1° Des compagnies aériennes proposent différents vols.

2° Un vol est ouvert à la réservation et fermé sur ordre de la compagnie.

3° Un client peut réserver un ou plusieurs vols, pour des passagers différents.

4° Une réservation concerne un seul vol, et un seul passager.

5° Une réservation peut être annulée ou confirmée.

6° Un vol a un aéroport de départ et un aéroport d'arrivée.

7° Un vol a un jour et une heure de départ et un jour et une heure d'arrivée.

8° Un vol peut comporter des escales dans des aéroports

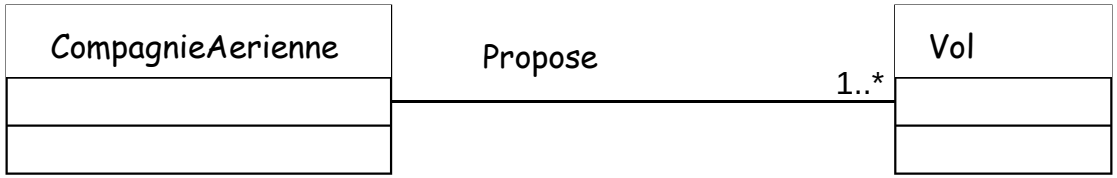
9° Une escale a une heure d'arrivée et une heure de départ.

10° Chaque aéroport dessert une ou plusieurs villes

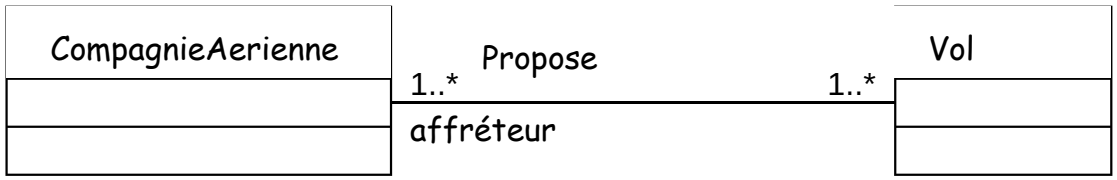
➢ Modélisation de la phrase :

1° Des compagnies aériennes proposent différents vols.

CompagnieAerienne et *Vols* sont 2 objets métiers : 2 classes

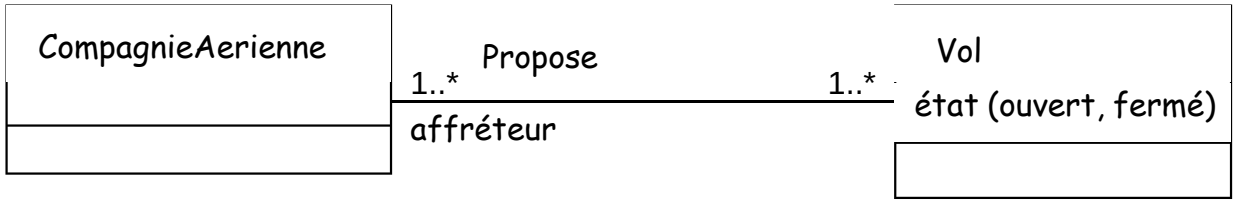


- Un vol est réalisé par une seule compagnie mais partagé par plusieurs affréteurs



➢ Modélisation de la phrase :

2° Un vol est ouvert à la réservation et fermé sur ordre de la compagnie.

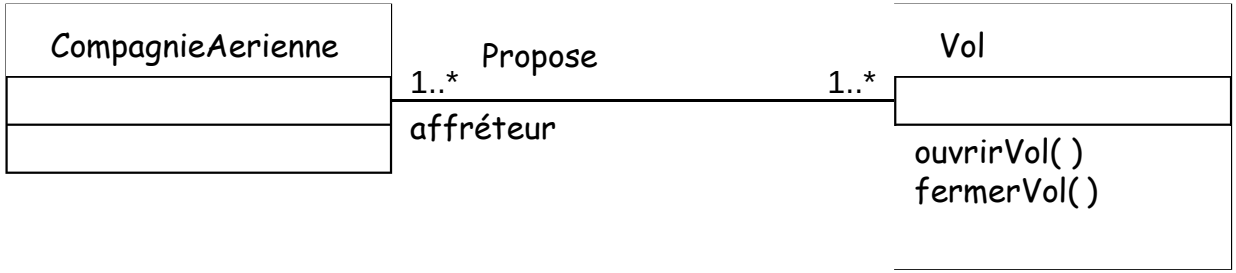


➢ Tout objet peut avoir un état (diagramme d'états).

➢ Dans un diagramme de classes tout concept dynamique est modélisé en opération.

➢ Il faut représenter la 2° phrase par 2 opérations : *ouvrirReservation( )* et *fermerReservation( )*

➢ Dans quelle classe ? Responsabilité d'une classe



➢ Les opérations sont déclarées dans l'objet dans lequel elles doivent s'exécuter

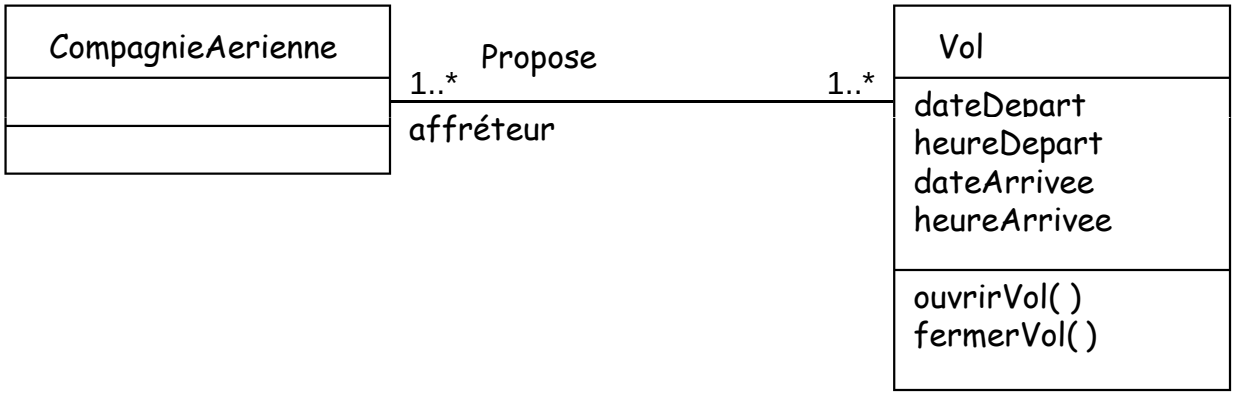
➢ Les autres pourront déclencher ces opérations par envoi de messages

➢ La classe CompagnieAerienne a une association avec la classe vol.

➢ Modélisation des phrases :

7° Un vol a un jour et une heure de départ et un jour et une heure d'arrivée.

➢ Les dates et les heures de départ et d'arrivée ne représentent que des valeurs : attributs.



➢ Pour savoir si un élément doit être représenté en attribut ou en objet :

➢ S'il n' y a que sa valeur qui est intéressante : c'est plutôt un attribut.

➢ Si plusieurs questions peuvent concerner l'élément, alors il faut le représenter en objet.

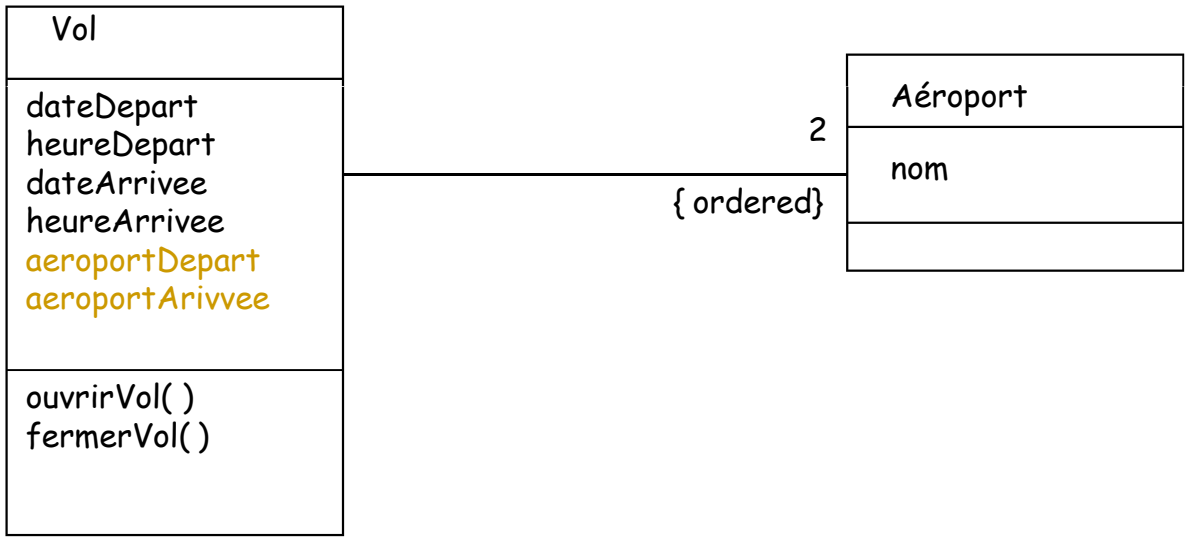
➢ Modélisation des phrases :

6° Un vol a un aéroport de départ et un aéroport d'arrivée.

➢Par quoi peut-on représenter l'élément "Aéroport" ?

3 réponses sont envisageables :

1. Soit avec une classe et une association de multiplicité 2

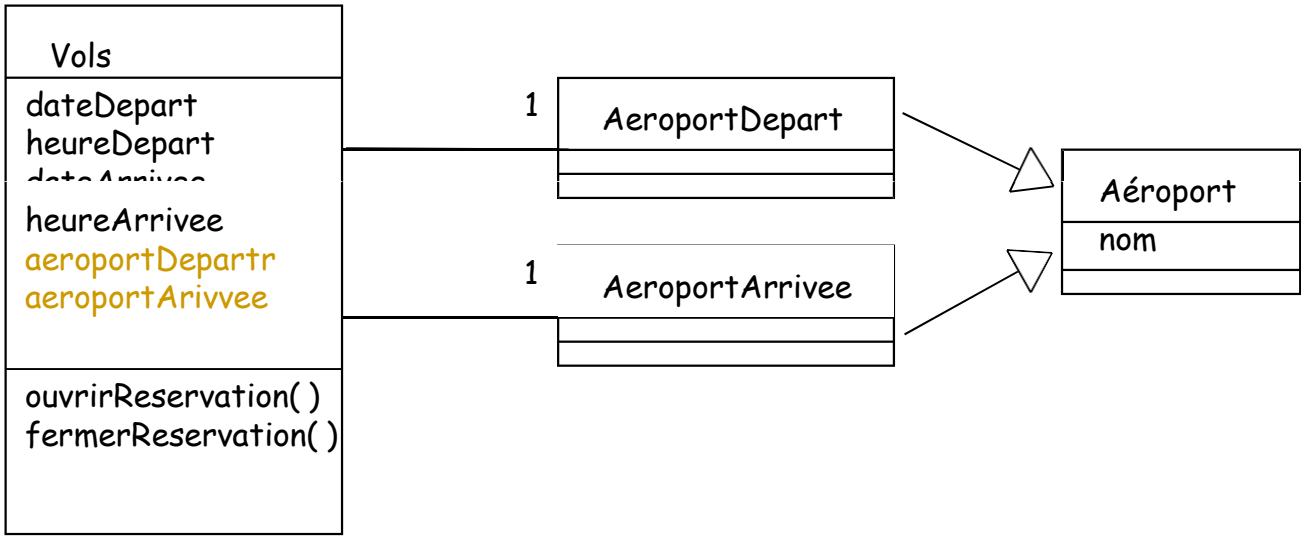


Modélisation peu parlante.

➢ Modélisation des phrases :

6° Un vol a un aéroport de départ et un aéroport d'arrivée.

2. Soit avec 2 classes

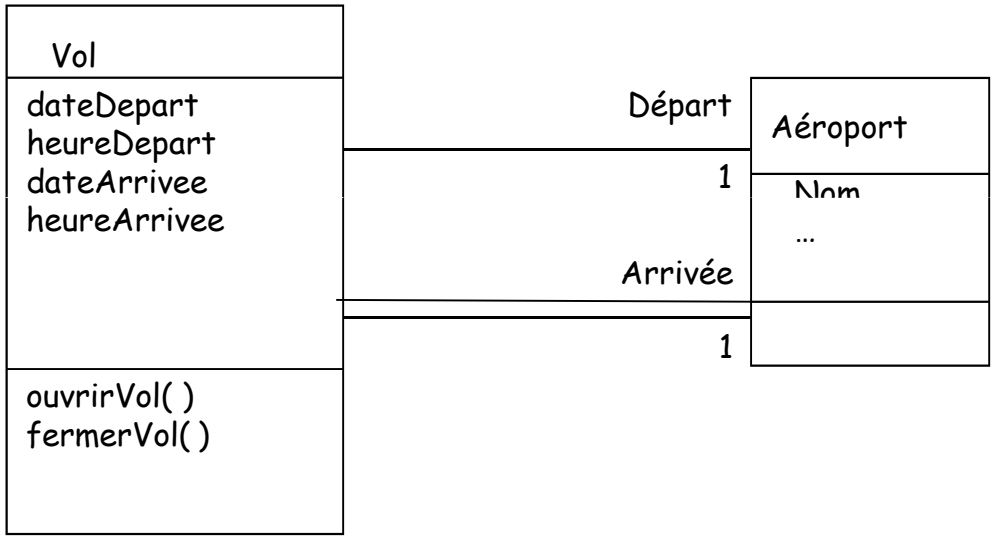


Modélisation non correcte. Tout aéroport peut être de départ et d'arrivée.

➢ Modélisation des phrases :

6° Un vol a un aéroport de départ et un aéroport d'arrivée.

2. Soit avec 2 associations

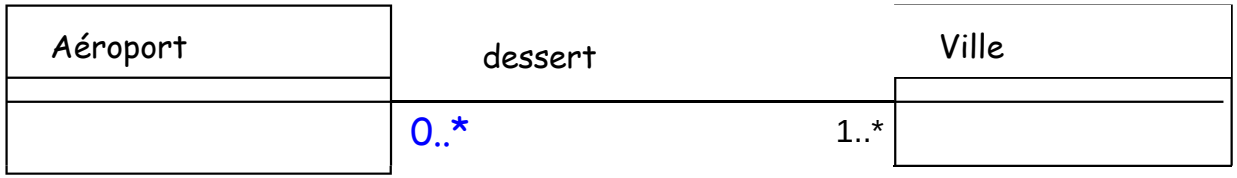


Le rôle de chaque association précise son sens.

➢ Modélisation des phrases :

10° Chaque aéroport dessert une ou plusieurs villes

➢ On ne peut pas savoir la multiplicité de ''*Aéroport*"



➢ Si on considère que desservir une ville signifie l'aéroport le plus proche, il n' en y a qu'un:
la multiplicité est de 1

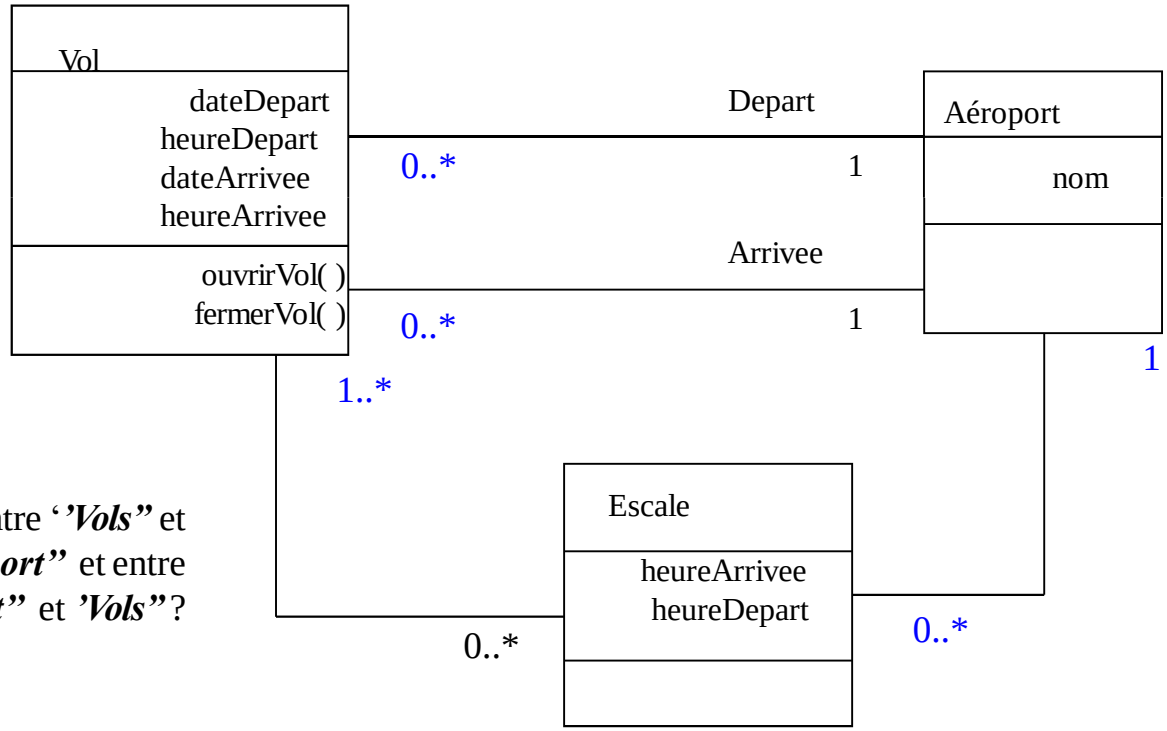
➢ Si on considère que desservir une ville signifie les aéroports dans un rayon de 35 km:
la multiplicité est de 0..*

➢Modélisation des phrases :

8° Un vol peut comporter des escales dans des aéroports

9° Une escale a une heure d'arrivée et une heure de départ.

➢Une escale a les propriétés heure d'arrivée et heure de départ, c'est donc un objet.



➢Quelles sont alors les multiplicités entre ***''Vols''*** et ***''Escale''***, entre ***''Escale''*** et ***''Aeroport''*** et entre ***''Aeroport''*** et ***''Vols''***?

➢ Modélisation des phrases :

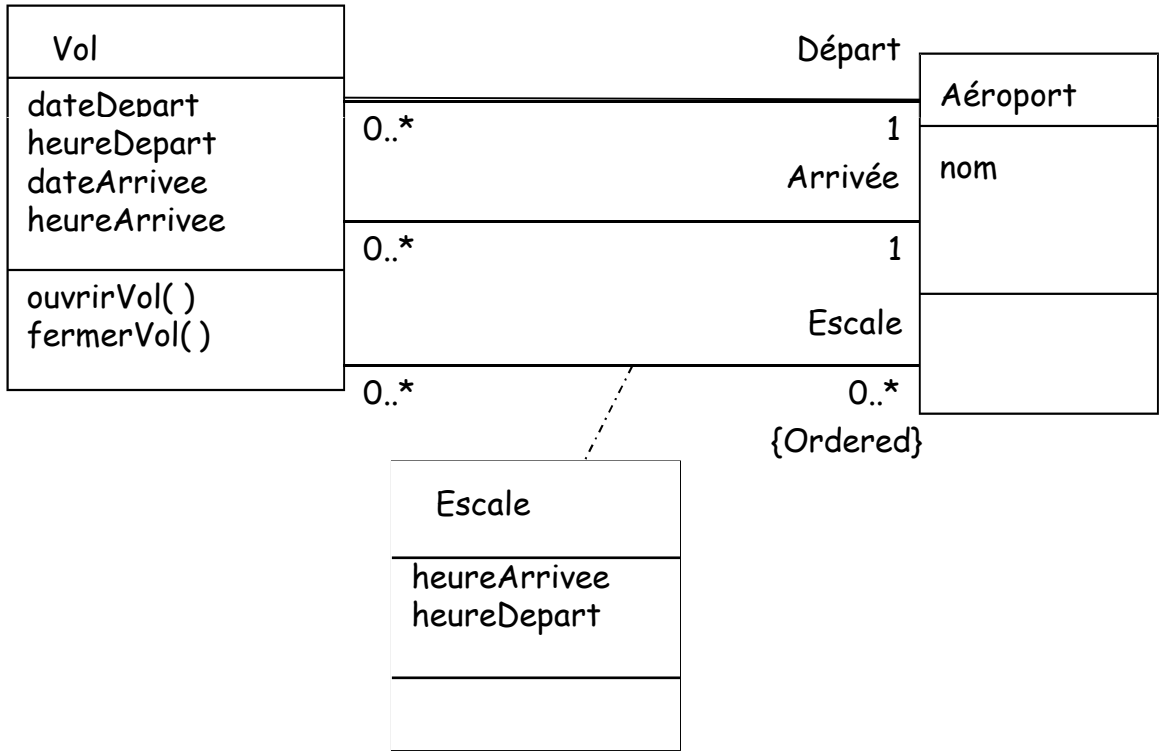
8° Un vol peut comporter des escales dans des aéroports

9° Une escale a une heure d'arrivée et une heure de départ.

➢ ''*Escale*'' a peu d'informations propres. Elle n'est qu'une partie de *"Vol"*.

➢ On peut la représenter comme une spécialisation de *"Aéroport"* . Mais elle n'est pas totalement un aéroport

➢ La meilleure solution serait de la modéliser comme une classe d'association entre et *'Vols"* et *''Aéroport"*.



➢ Modélisation des phrases :

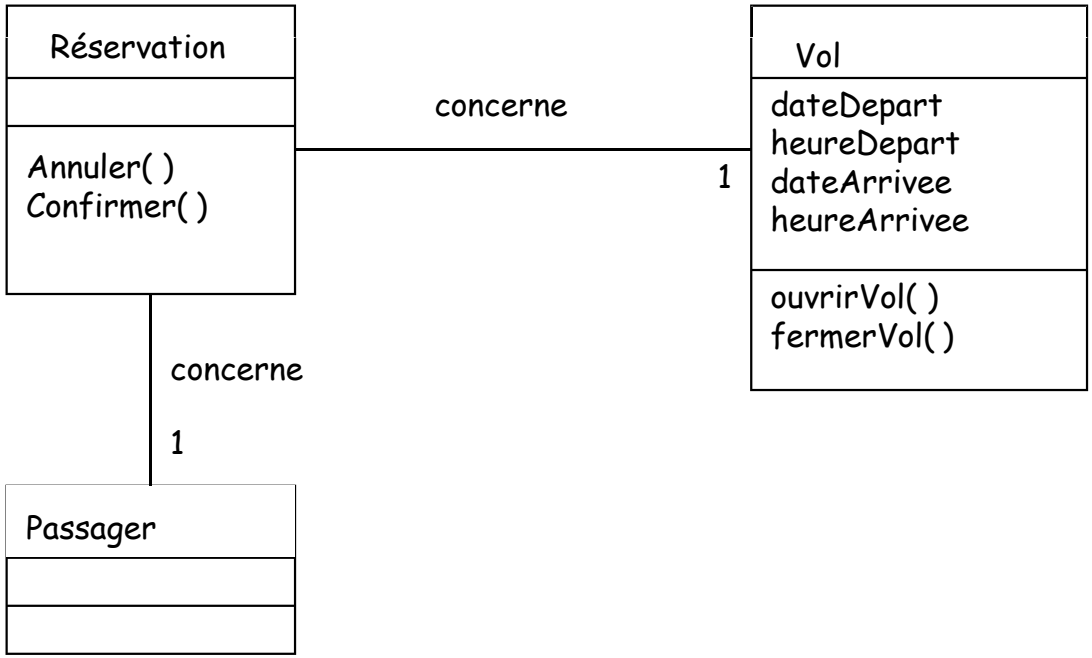
4° Une réservation concerne un seul vol, et un seul passager.

5° Une réservation peut être annulée ou confirmée.

➢ La réservation et le passager sont 2 concepts métier : 2 classes d'objets

➢ Un réservation concerne un seul vol et un seul passager: donc 2 associations entre "*Vol*" et "*Réservation*" et entre "*Réservation*" et "*Passager*".

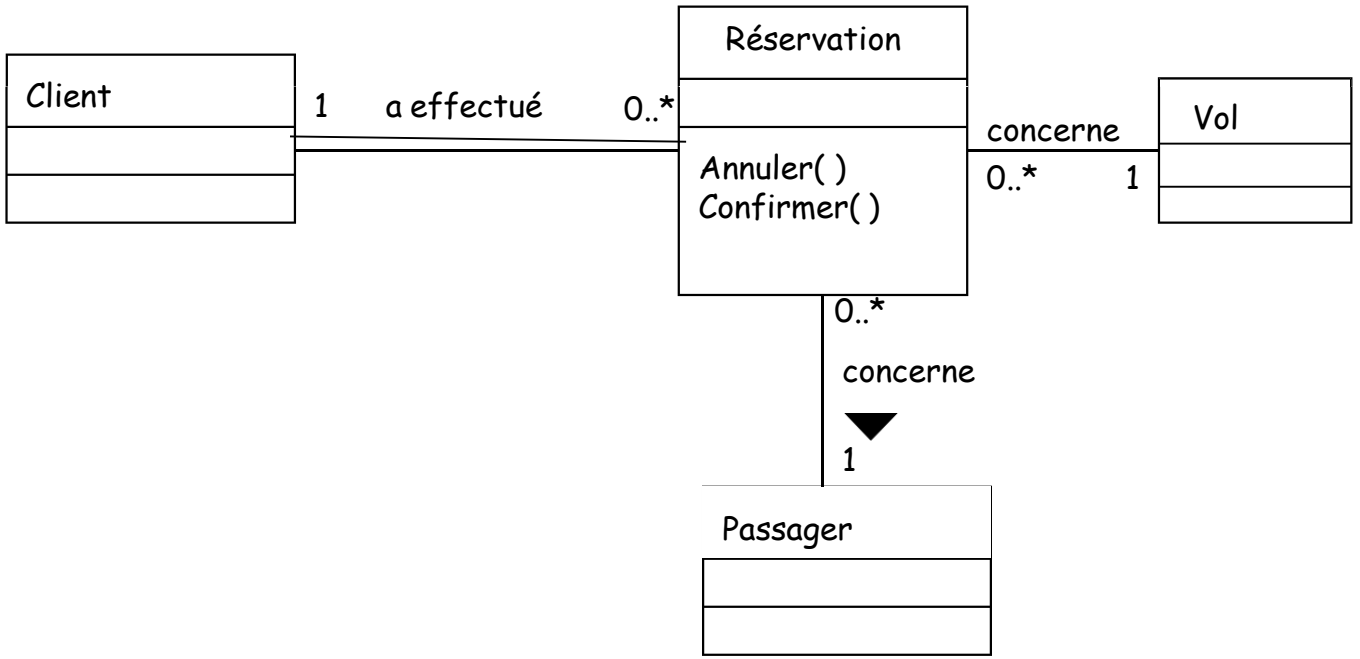
➢ La 5° phrase se traduit par l'ajout de 2 opérations *annuler( )* et *confirmer( )* dans "*Reservation*".



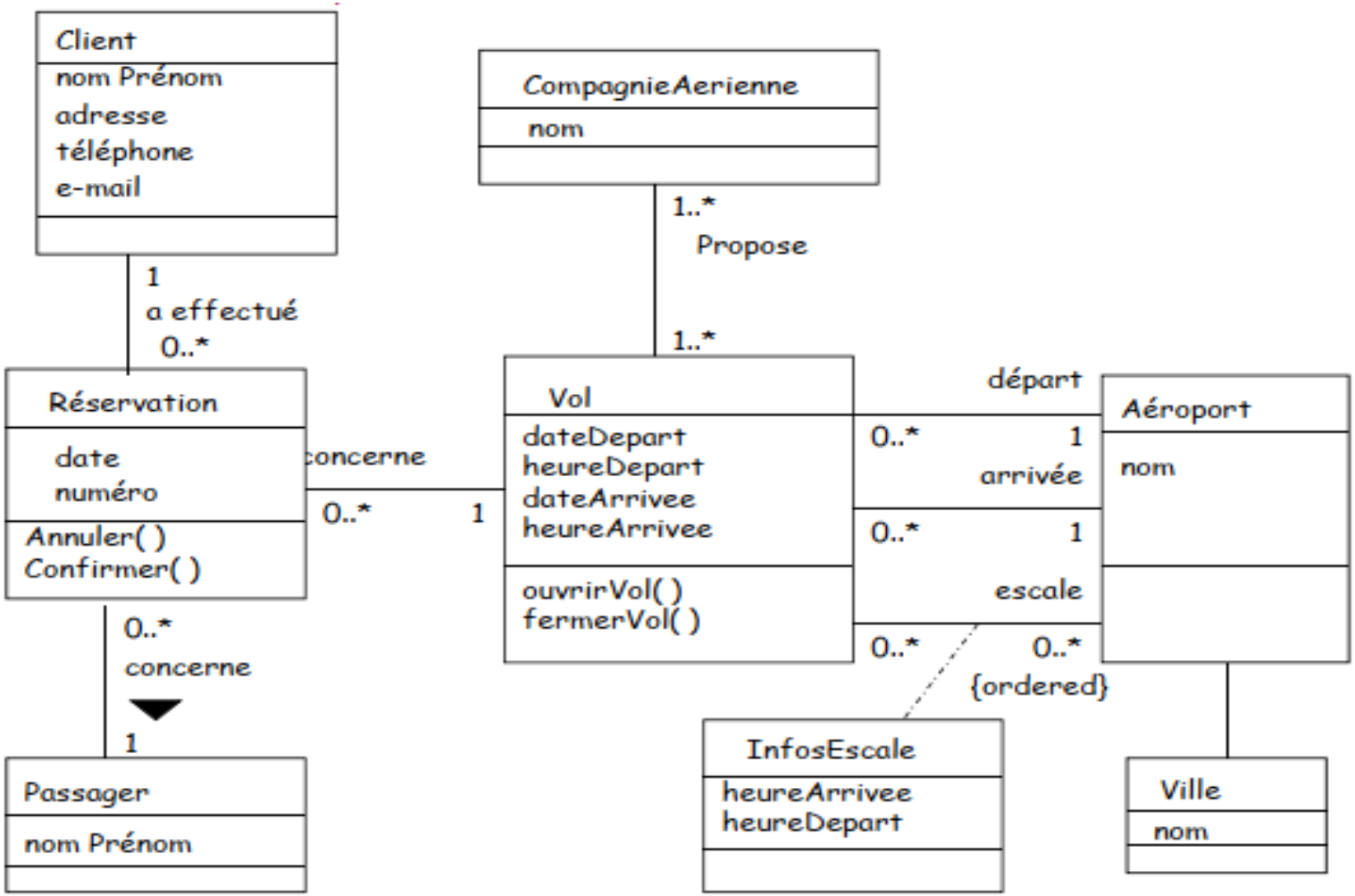
➢ Modélisation des phrases :

3° Un client peut réserver un ou plusieurs vols, pour des passagers différents.

➢ Il faut discerner un client d'un passager



➢ Le diagramme des classes complet est :



➢Diagramme des classes complet et annoté