Modèle structurel

(Aspects orientés objets, données, vue statique)

Principes du modèle structurel

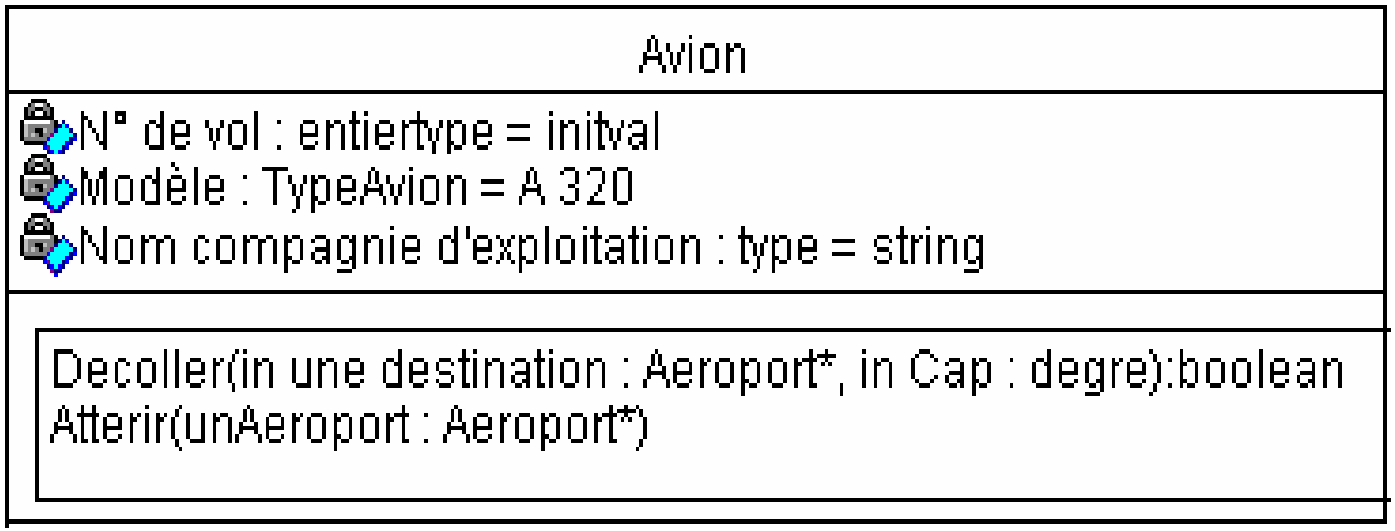
- Nous limiterons le modèle structurel aux aspects statiques centrés sur les objets. Deux types de diagrammes sont décrits dans ce modèle :
  - Le diagramme de classe
  - Le diagramme d'instance

Le diagramme de classe

- Central pour la modélisation objet
- Montre la structure statique du système
  - Les types d'objets
  - Les relations entre eux
    - Les associations
    - Les sous-types

Vues graphique d'une classe

Représentation complète



Représentation simplifiée

Avion

Compagnie :: Avion

Représentation avec nom du paquetage

Visibilité

- Trois visibilités sont disponibles pour les attributs et les opérations de la classe (pas sur les associations)
  - Public : accessible pour tout utilisateur d'une classe, y compris la classe elle-même.
  - Protégé : accessible par la classe elle-même et par ses héritiers.
  - Privé : accessible seulement par la classe elle-même.

Le modèle de classe/d'objet

Attribut :

visibilité *nom* : type = valeur initiale

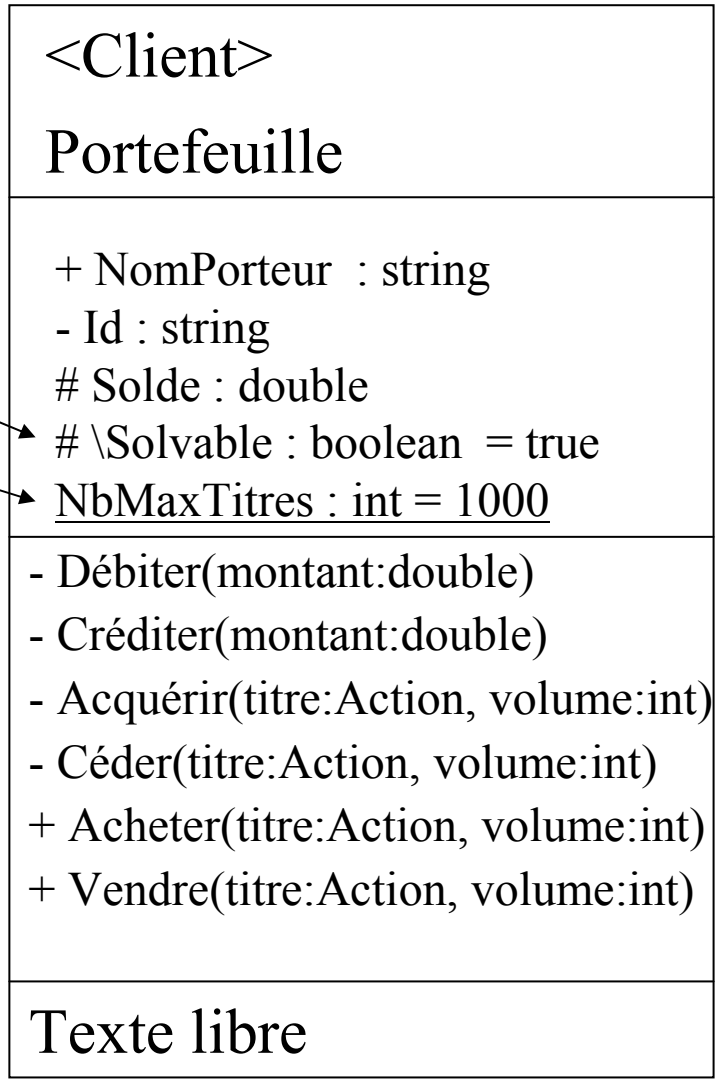
attribut dérivé (redondance)

attribut de classe (static)

Opération :

visibilité *nom* (liste params) : type-retour

en analyse, seul le *nom* est obligatoire



Règle de non redondance

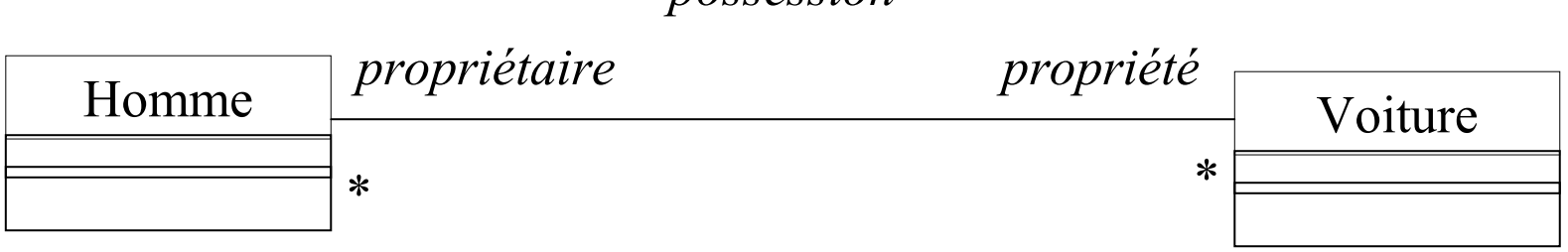
- Toutes les informations mémorisées, manipulées, partagées … pour accomplir les finalités du domaine doivent apparaître dans le diagramme de classe. Cependant chaque propriété ne doit figurer qu'une fois.

Règle de non redondance : une information élémentaire d'un domaine ne doit figurer qu'à un seul endroit du diagramme de classe des entités de ce domaine.

Association entre classes

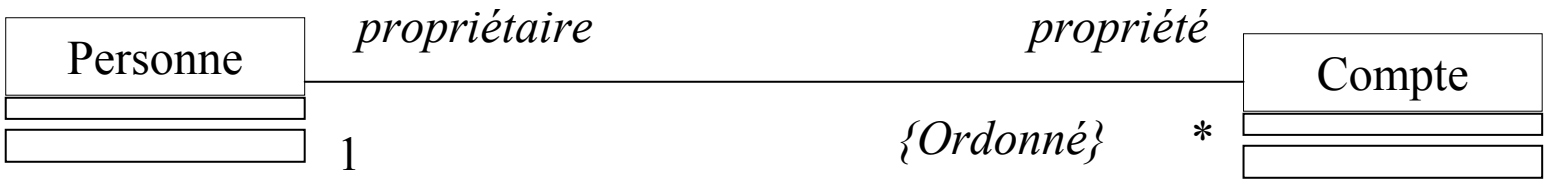
- attributs / associations
  - en analyse, un attribut ne peut être une classe
- Pour décrire les attributs d'une classe on fait souvent référence à des attributs d'autres classes. **Une association** permet cette référence tout en respectant la règle de non redondance.

- Notation (remarquer le nom et les rôles)

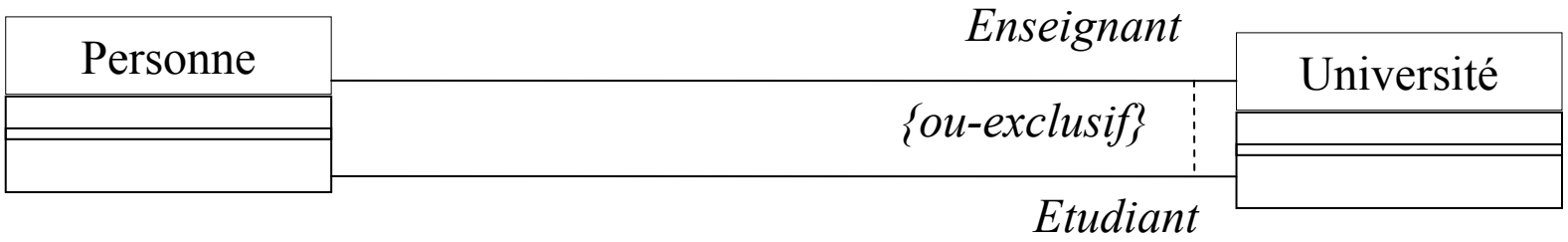


Contraintes sur les associations

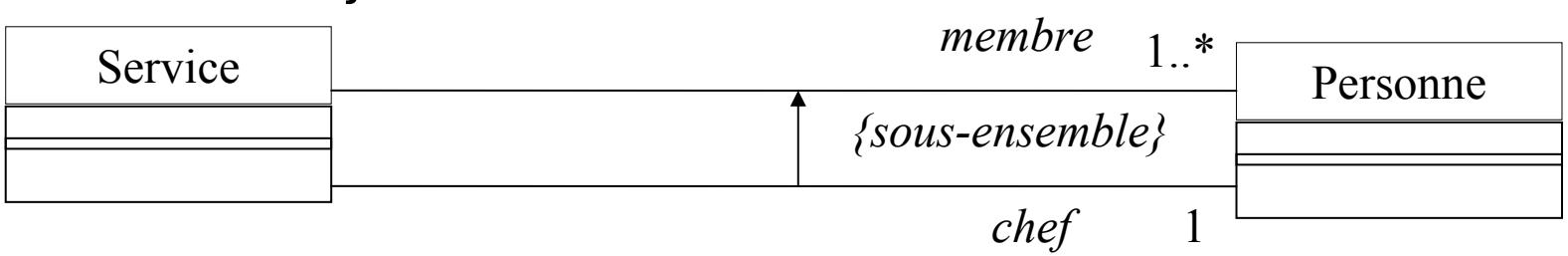
- {Ordonné}



- {Ou-exclusif}

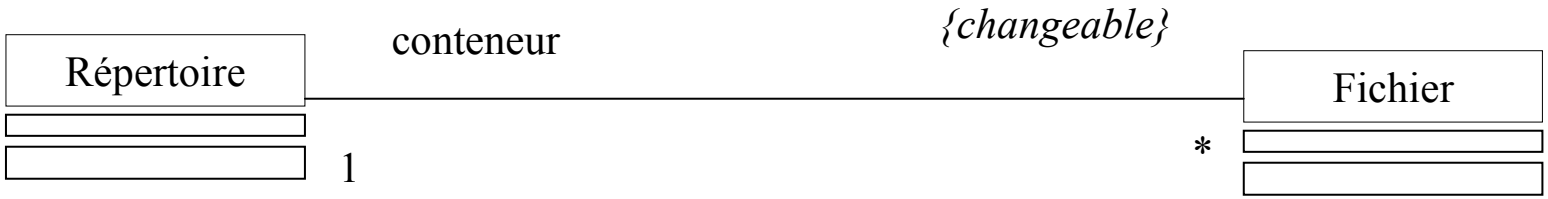


- {Sous-ensemble}

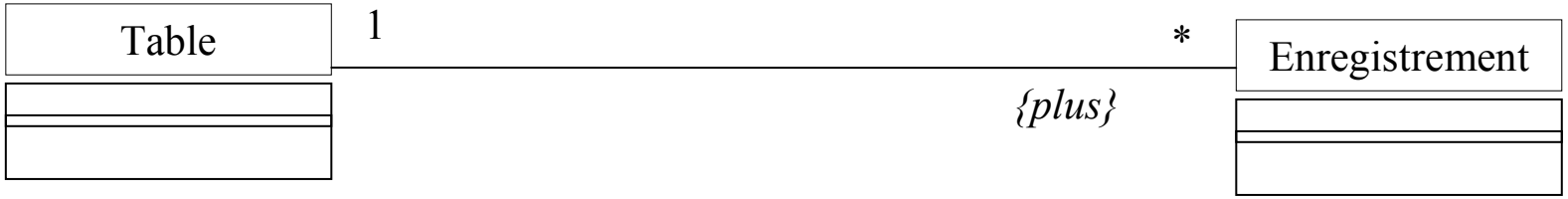


Contraintes sur les associations

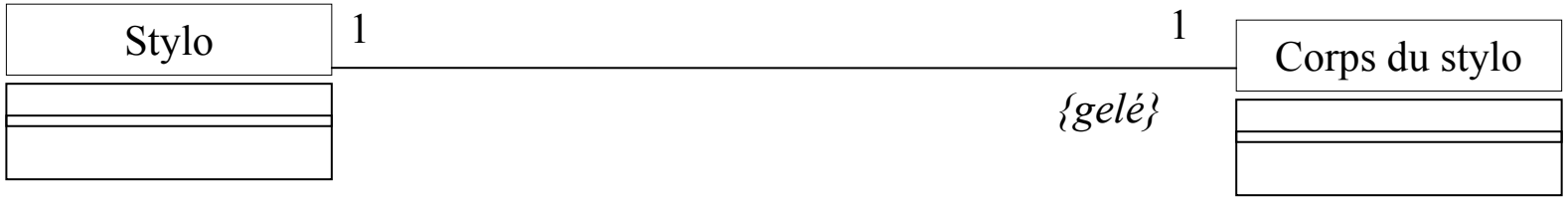
- {changeable}



- {plus}

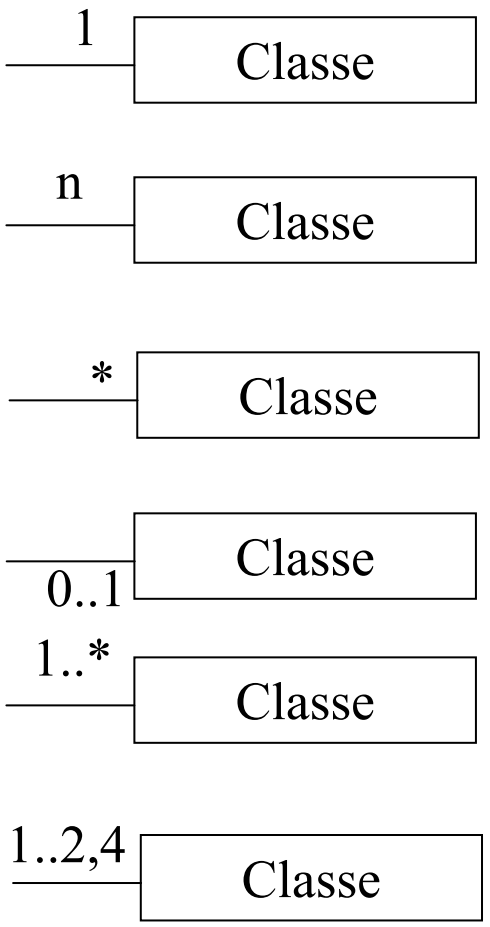


- {gelé}

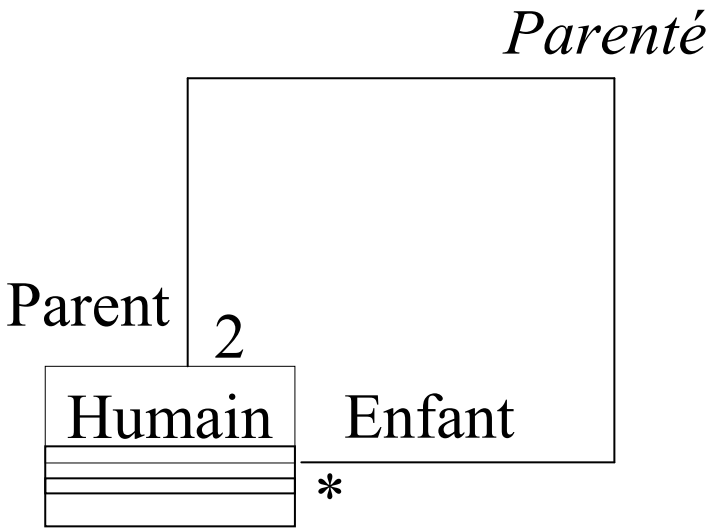


Cardinalité d'une association

- Exactement une
- Exactement n
- Plusieurs (0 ou plus)
- Optionnelle (0 ou 1)
- 1 ou plus
- Cardinalité spécifiée



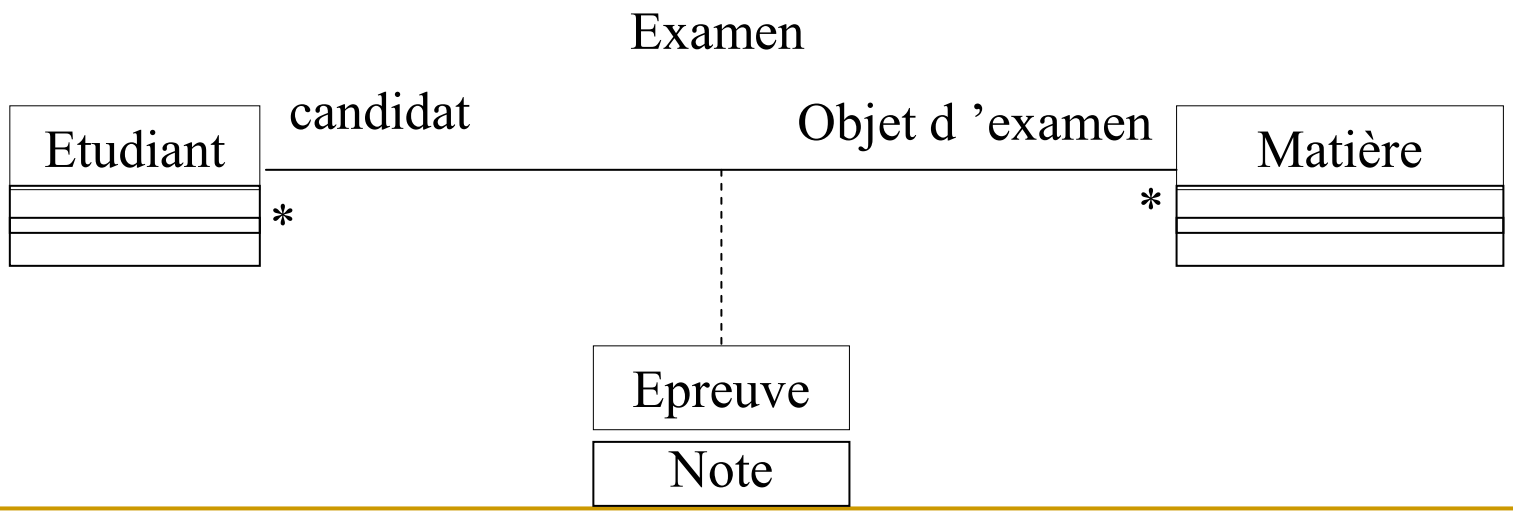
Association réflexive



- Une association réflexive lie des objets de même classe
- Dans ce cas, le nom des rôles est obligatoire

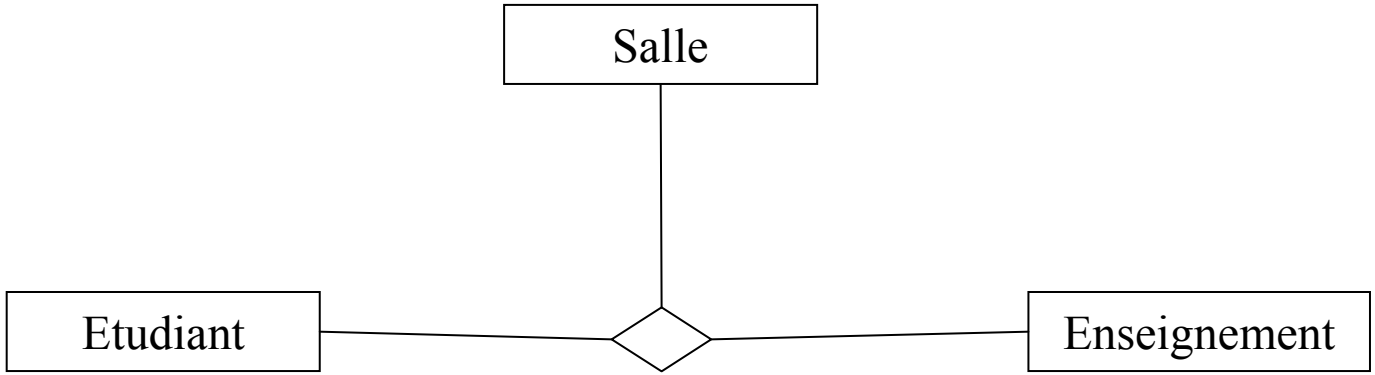
Les classes-associations

- Un attribut peut dépendre fonctionnellement de deux identifiants de deux classes, nous pouvons alors faire porter l'attribut par l'association entre ces classes.



Arité des associations

- Les associations sont souvent binaires car elles relient deux classes. On représente des arités supérieures par un losange

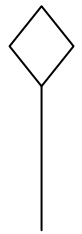


Exemple de relation ternaire qui matérialise un cours

Multiplicité : pour un couple d'instance de la classe A et de la classe B, il y a au minimum r1 instances de la classe C et au maximum r2

Agrégation/Composition

- Agrégation : Chaque élément est en relation partie-tout avec l'agrégat
  - Exemple : les wagons d'un train

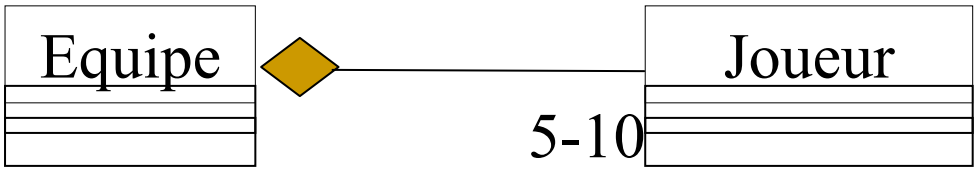


- Composition : Tous les éléments sont indispensables par nature à l'existence du composé
  - Un même composant ne peut appartenir qu'à un composé
  - Un composé ne change pas de composant
    - Exemple : les membres du corps pour le corps

En général, c'est la conception qui permet de trancher

L ’agrégation

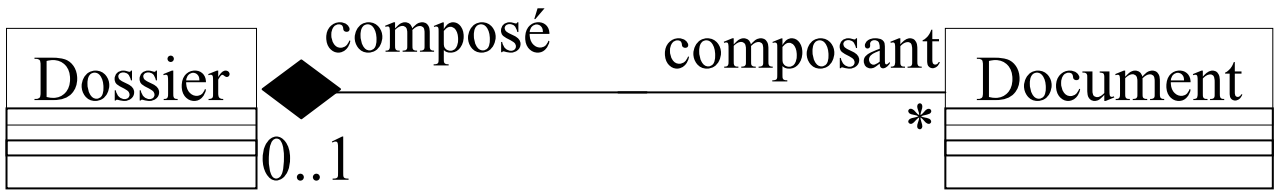
Une association de type agrégation traduit la volonté de renforcer la dépendance entre les classes. L’agrégation correspondant à une classification, la granularité de l’agrégé étant plus importante que celle de l’agrégat.



L ’agrégation se représente en ajoutant un petit losange du coté de l ’agrégat

Composition

- La composition est un cas particulier d ’agrégation. Elle implique une coïncidence des durées de vie du composant et du composite.

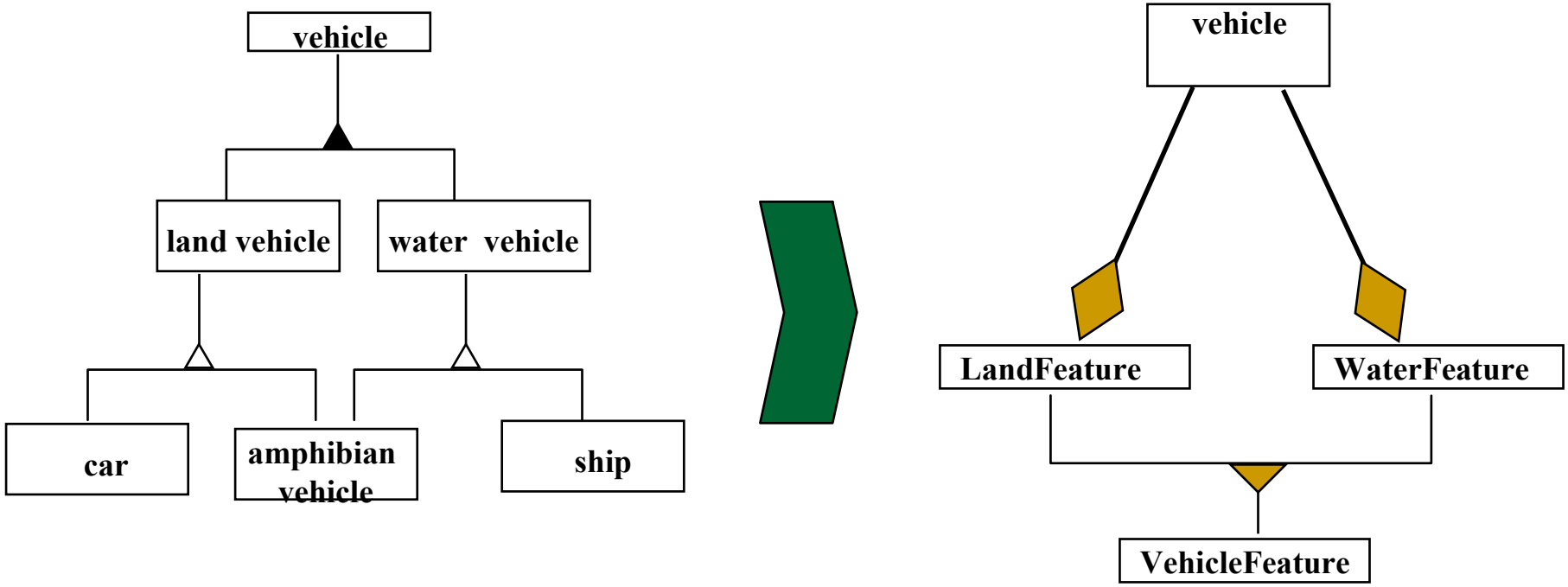


  - La destruction d’une instance de dossier entraîne la destruction des instances des document associées
  - La cardinalité du composé est 0 ou 1

Utilisation de l'agrégation

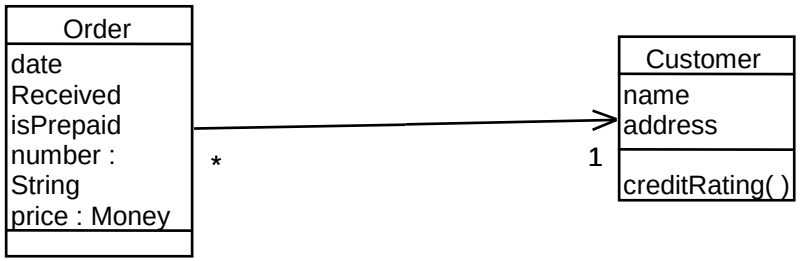
- Une agrégation EST une association
- Un agrégat est par définition la somme de ses partie
- La suite de liens d'agrégation NE PEUT PAS former de cycle
- La composition s'applique quand chaque partie appartient à un objet. Les parties n'ont pas d'existence indépendante.

Délégation et agrégation



Navigation dans le diagramme

- La flêches indiquent la “navigabilité”



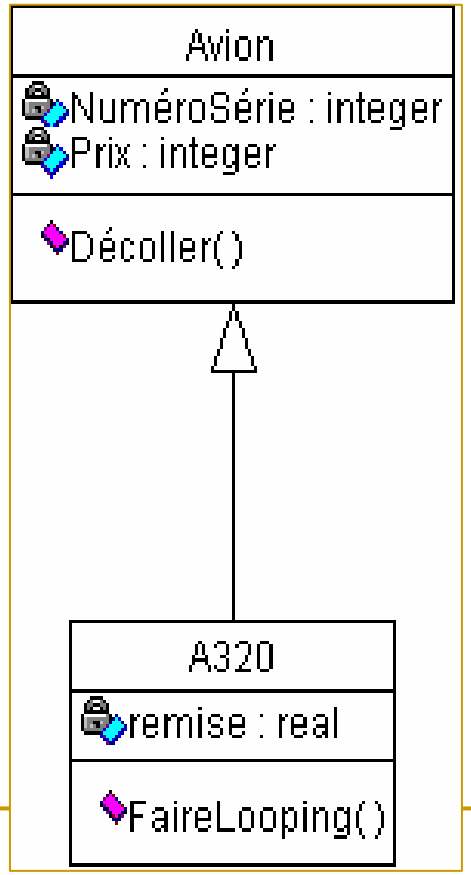
- La “facture” doit indiquer le “client”
- Le “client” ne connait pas les “factures”
- Une association peut-être bi-directionnelle (rôle et rôle-inverse)

Généralisation

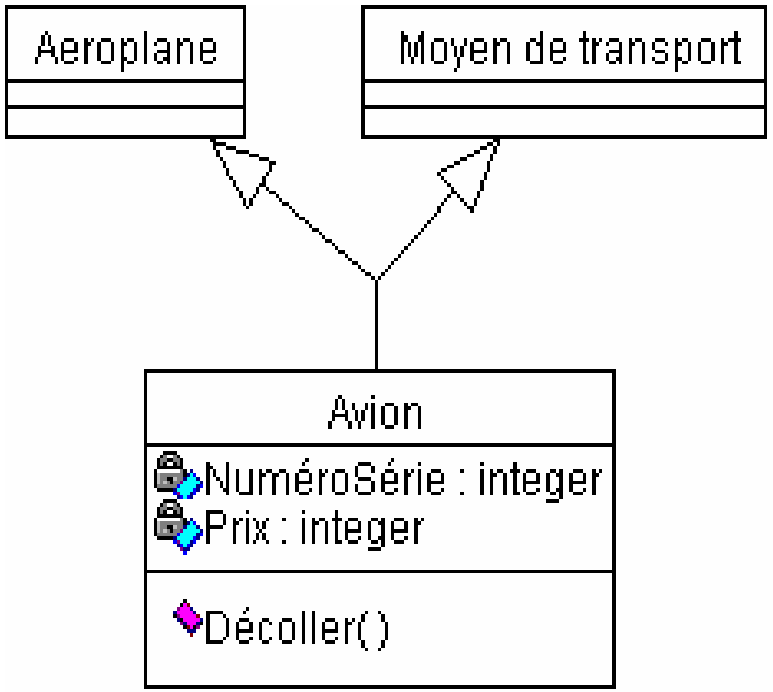
- Définition et propriétés
  - Une classe héritière est une spécialisation de sa classe mère
  - En terme de spécialisation comme de codage, une classe héritière est un sous-ensemble de la super-classe
  - Toutes les propriétés d'une super-classe sont vrai pour la classe héritière (attributs, opérations, associations)
  - L'invariant d'une sous-classe comprend celui de sa super-classe

Représentation de la généralisation

- Généralisation simple

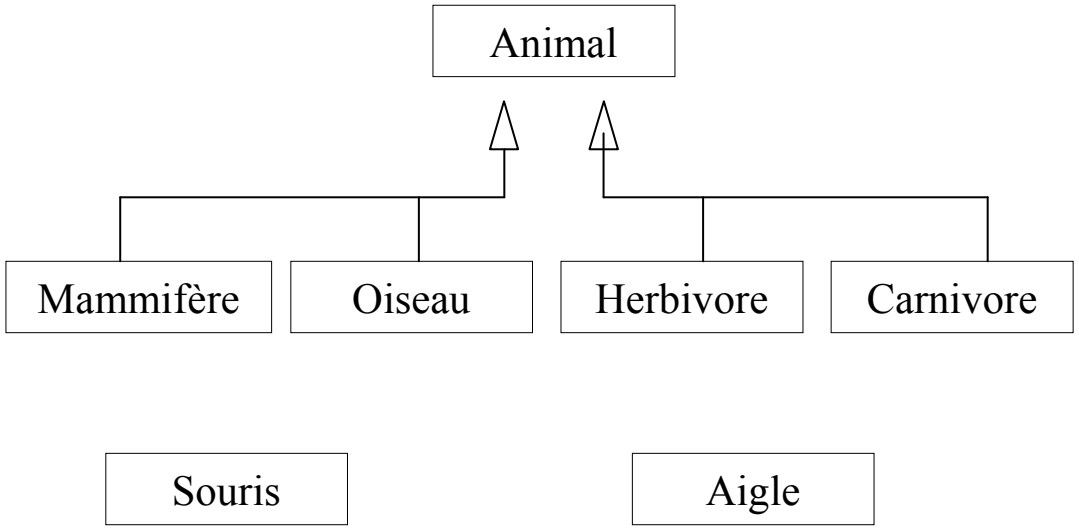


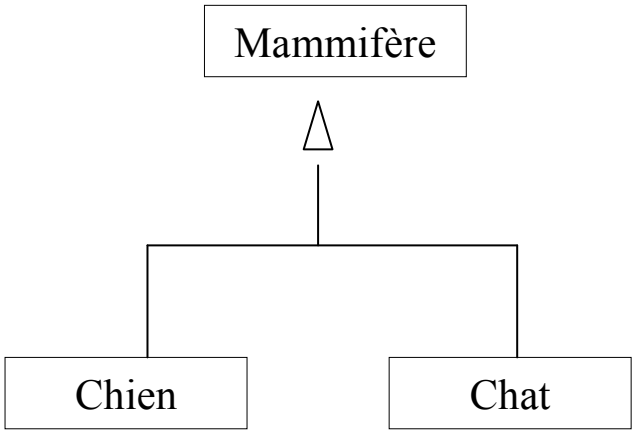
- Généralisation multiple



Contraintes sur la généralisation

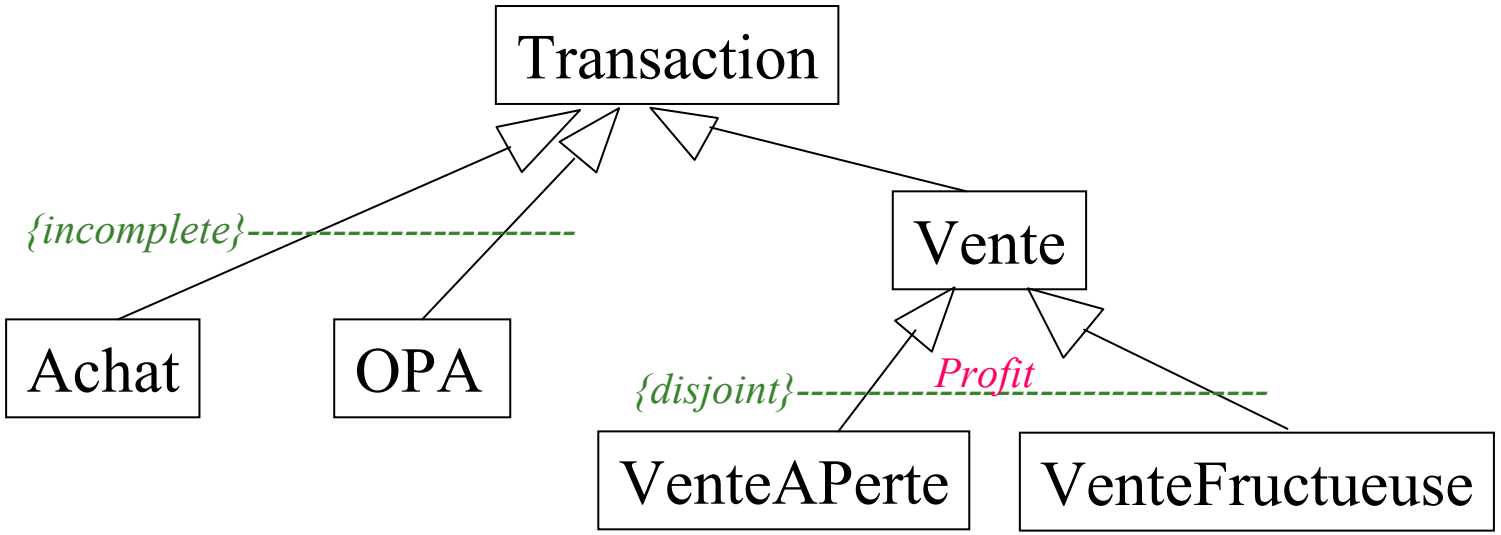
- {overlapping}:une classe dérivée peut hériter de plus d ’une mère
- {disjointe} : une classe dérivée ne peut hériter que d ’une mère
- {complète} : toutes les classes filles sont spécifiées
- {incomplète} : la liste des classes filles est incomplète





Généralisation

- Les éléments sont plus spécifiques que la classe générale
- On peut indiquer des *discriminants*



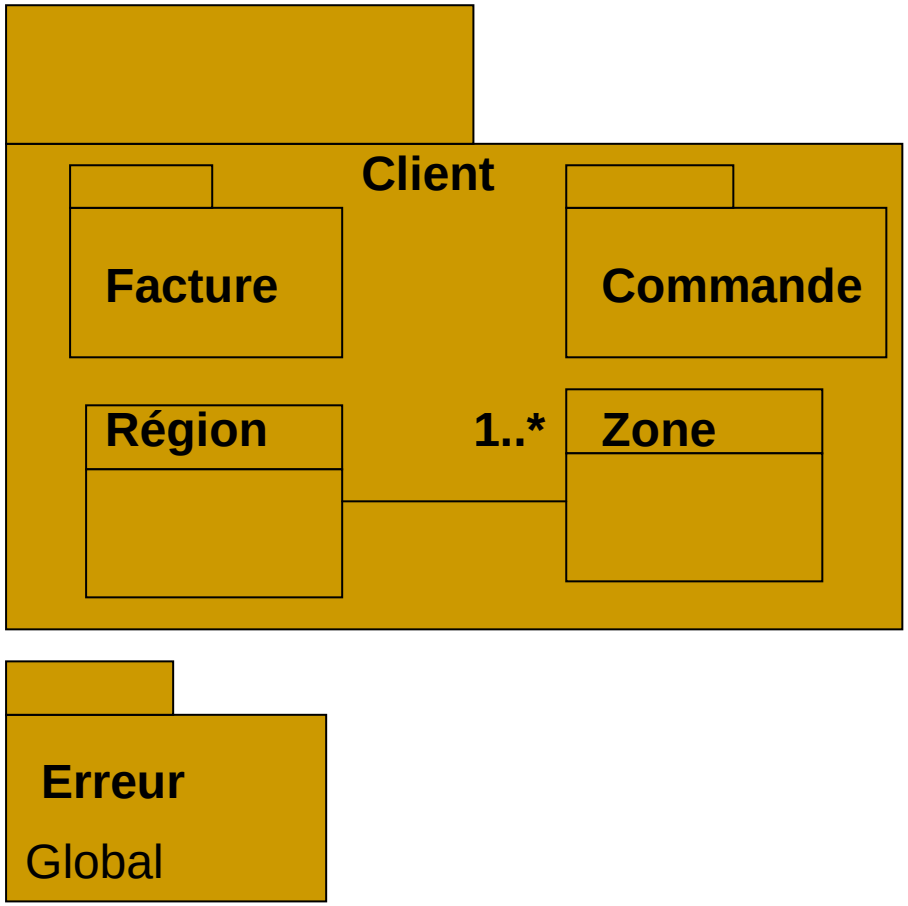
Les stéréotypes

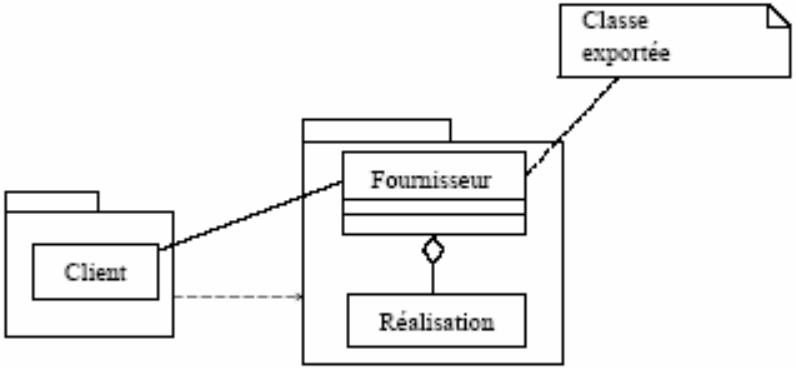
- Les stéréotypes permettent d'étendre la sémantique des éléments de modélisation : il s'agit d'un mécanisme d'extensibilité du métamodèle d'UML.
- Les stéréotypes permettent de définir de nouvelles classes d'éléments de modélisation, en plus du noyau prédéfini par UML.
- Utilisez les stéréotypes avec modération et de manière concertée (notez aussi qu'UML propose de nombreux stéréotypes standards)
- Notation << … >>>
- Exemple : indiquer les interfaces, les classes d'implantation, les utilitaires, …

Les paquetages

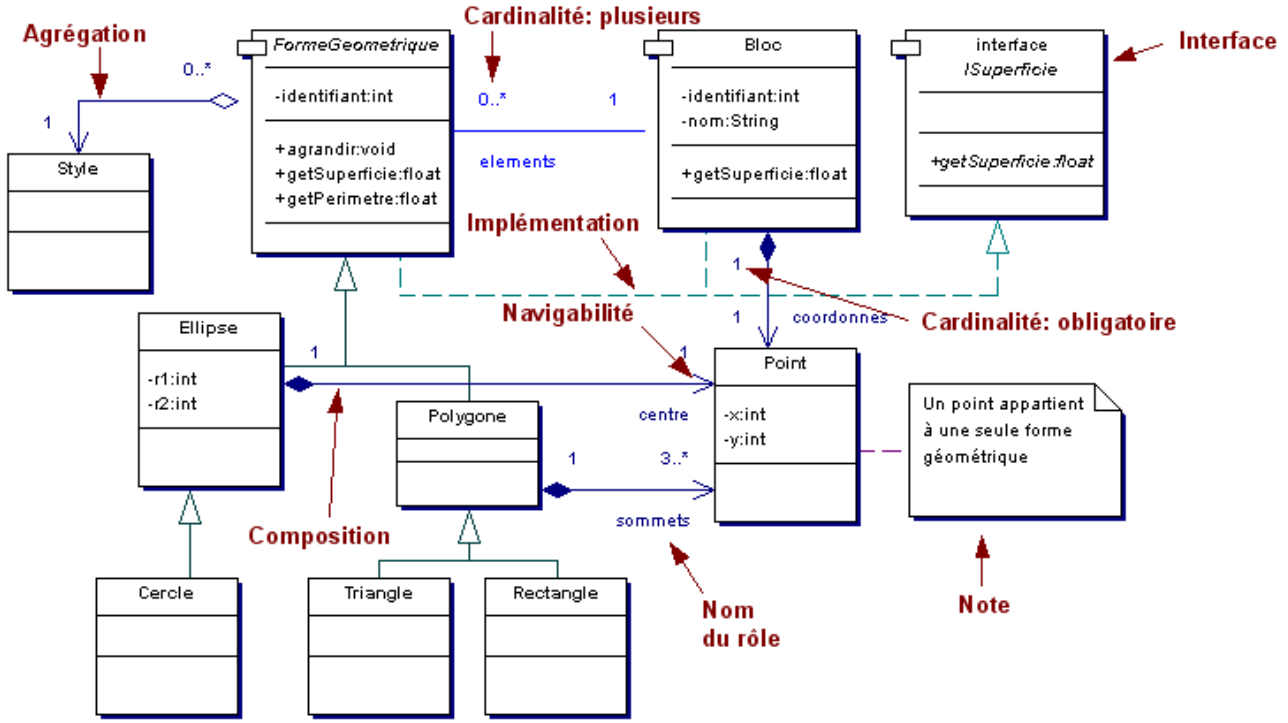
- Les paquetages sont des éléments d'organisation des modèles.
  - Ils Regroupent des éléments de modélisation, selon des critères purement logiques.
  - Ils Permettent d'encapsuler des éléments de modélisation (ils possèdent une interface).
- Ils permettent de structurer un système en catégories (vue logique) et sous-systèmes (vue des composants).
- Ils représentent le bon niveau de granularité pour la réutilisation.
- Les paquetages sont aussi des espaces de noms.

Exemple de paquetages





Exemple de diagramme de classe



- ISuperficie est une interface (FormeGeometrique et Bloc l'implémentent)
- agrégation : FormeGeometrique est associée à un style qui lui peut être associé à plusieurs instances de FormeGeometrique
- composition : Ellipse réfère à un Point qui a le rôle *centre* par une relation de composition
- Ellipse connait son centre mais l'inverse n'est pas vrai (navigabilité).

Notions avancées

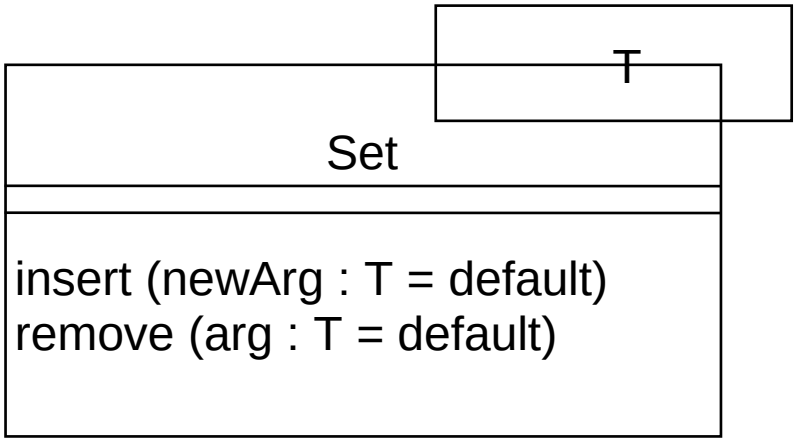
- Classe abstraite : classe générique qui n'est pas instanciée dans l'application.
  - Correspond à une classe << mère >> dans une relation d'héritage
  - Equivalent de la notion en C++, Java
- Classe paramétrée :
  - La classe est paramétrée par un type d'objet
  - Equivalent des templates C++
  - Exemple : liste
- Interface : décrit un comportement générique de classe
  - Notion Java plutôt que C++ (partie externe)

Interface

- Une *interface* est une collection d'opérations utilisée pour spécifier un service offert par une classe.
- Une interface être vue comme une classe sans attributs et dont toutes les opérations sont abstraites.
- Une interface est destinée à être *“réalisée”* par une classe (celle-ci en hérite toutes les descriptions et concrétise les opérations abstraites).
- Une interface peut en spécialiser une autre, et intervenir dans des associations avec d'autres interfaces et d'autres classes.

Les classes paramétrées

- Classe paramétrée = template
- Souvent utilisé pour représenter les collections
- INUTILE pour la modélisation conceptuelle
  - Les collections sont “cachées” dans la multiplicité



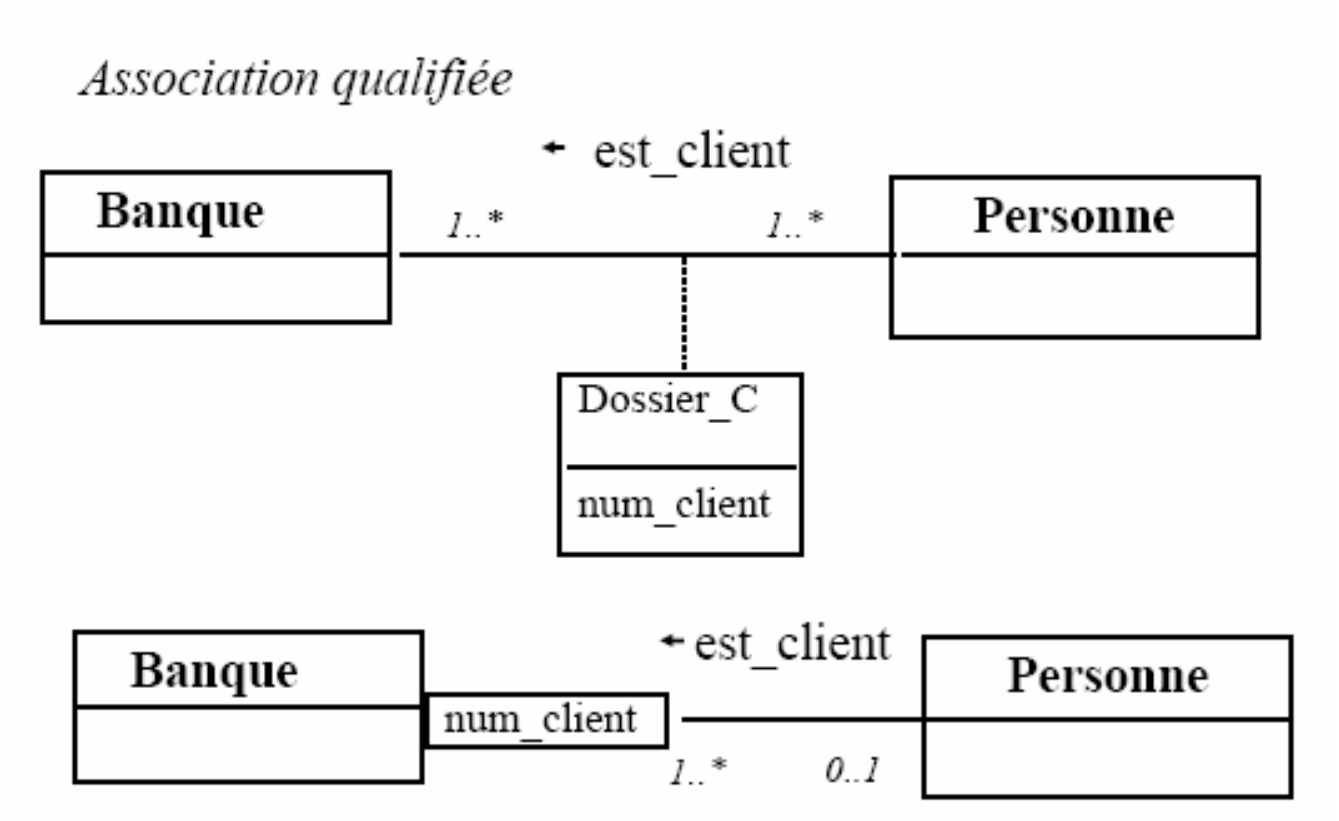
Convention de nommage en U.M.L.

- Les noms et les attributs commencent toujours par une minuscule (contrairement au nom des classes qui commencent systématiquement par une majuscule) et peuvent contenir ensuite plusieurs mots concaténés, commençant par une majuscule.

- Il est préférable de ne pas utiliser d ’accent ni de caractères spéciaux.

- Les mêmes conventions s ’appliques au nommage des rôles des associations ainsi qu’aux opérations.

Associations qualifiées



Pour résumer

- Nous avons définie le diagramme de classe ainsi que les objets qui le composent : les classes (attributs, opération), les associations (arité, multiplicité, contraintes), et les liens de généralisation/spécialisation (avec les contraintes associées).

- Nous avons donné les règles de nomenclature (de nommage) des objets manipulés dans le diagramme de classe.

Diagramme de classes/d'objets