Méthodologies de conception des SI

Chapitre 4
Le diagramme de classes

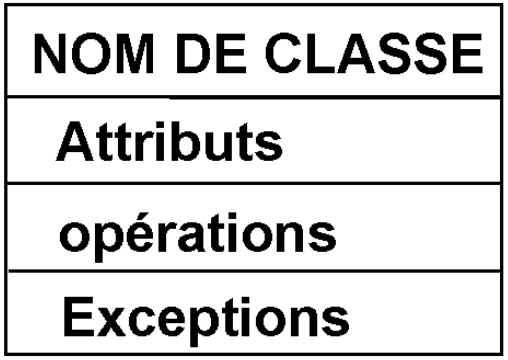
Concepts de base

Diagramme de classes

- Diagramme central du modèle du SI.
- Montre les classes et leurs relations statiques.
- Le plus riche en notations.
- Équivalent du modèle E-A.
- Les erreurs dans ce diagramme ont souvent un impact sur les autres diagrammes.

Concepts de base

- **Avec UML, une classe :**
  - ✓ **est représentée par un rectangle avec :**
    - Attributs et
    - opérations.



Classe = Attributs + Opérations + Instanciation (Constructeur)

Concepts de base

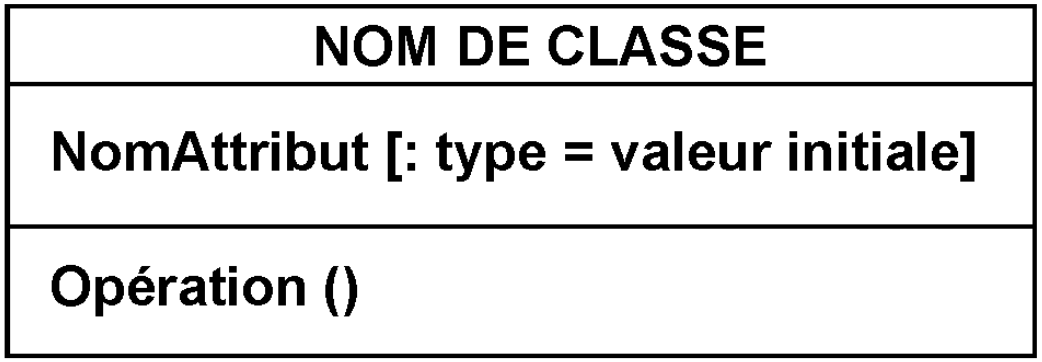
- ✓ **Remarque : Par abus de langage**
  - ◆ Attribut = propriété = donnée-membre, …
  - ◆ Opération = méthode = fonction-membre, …

Concepts de base

- **Pour une classe :**
  - ✓ Le nom de la classe, selon la norme UML est en gras,
    mais on peut se limiter à l'écrire en majuscule.
  - ✓ Une propriété d'une classe constitue un élément de l'état de ses objets,
    → participe à la caractérisation des objets.
  - ✓ Une opération représente un service spécifique offert par les objets de la classe.

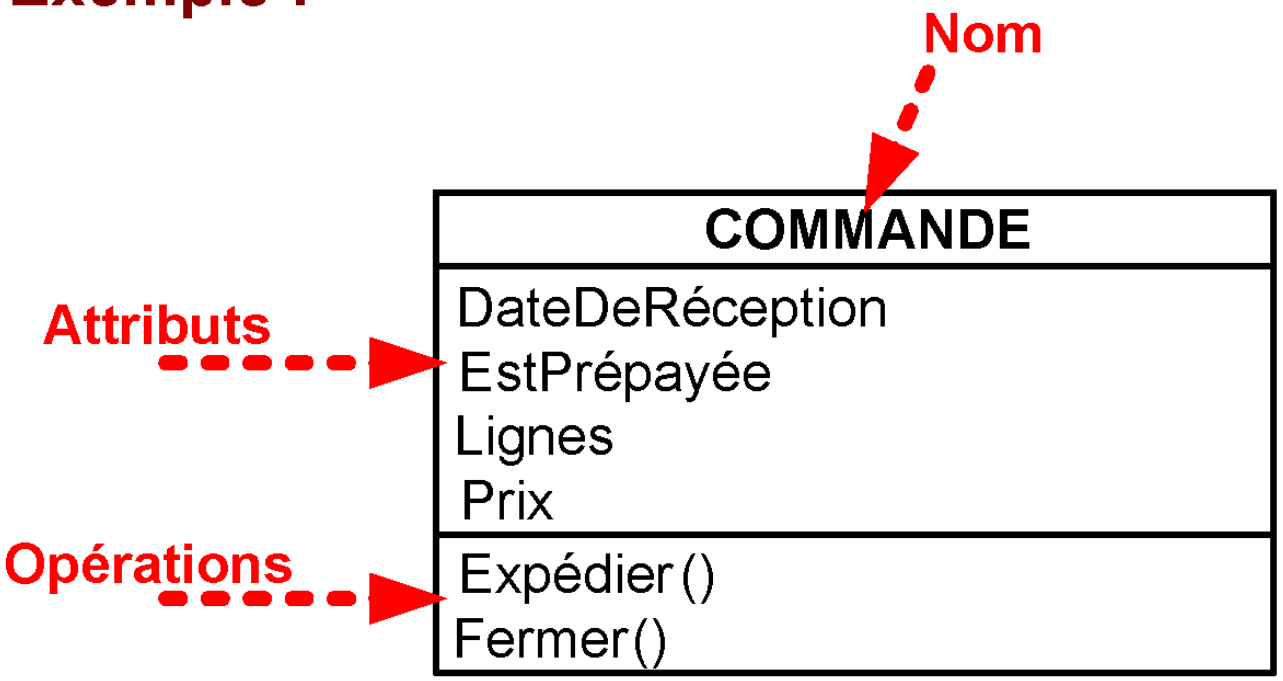
Concepts de base

- **Les attributs et les opérations :**
  - ✓ **sont décrits dans le deuxième et troisième compartiments.**



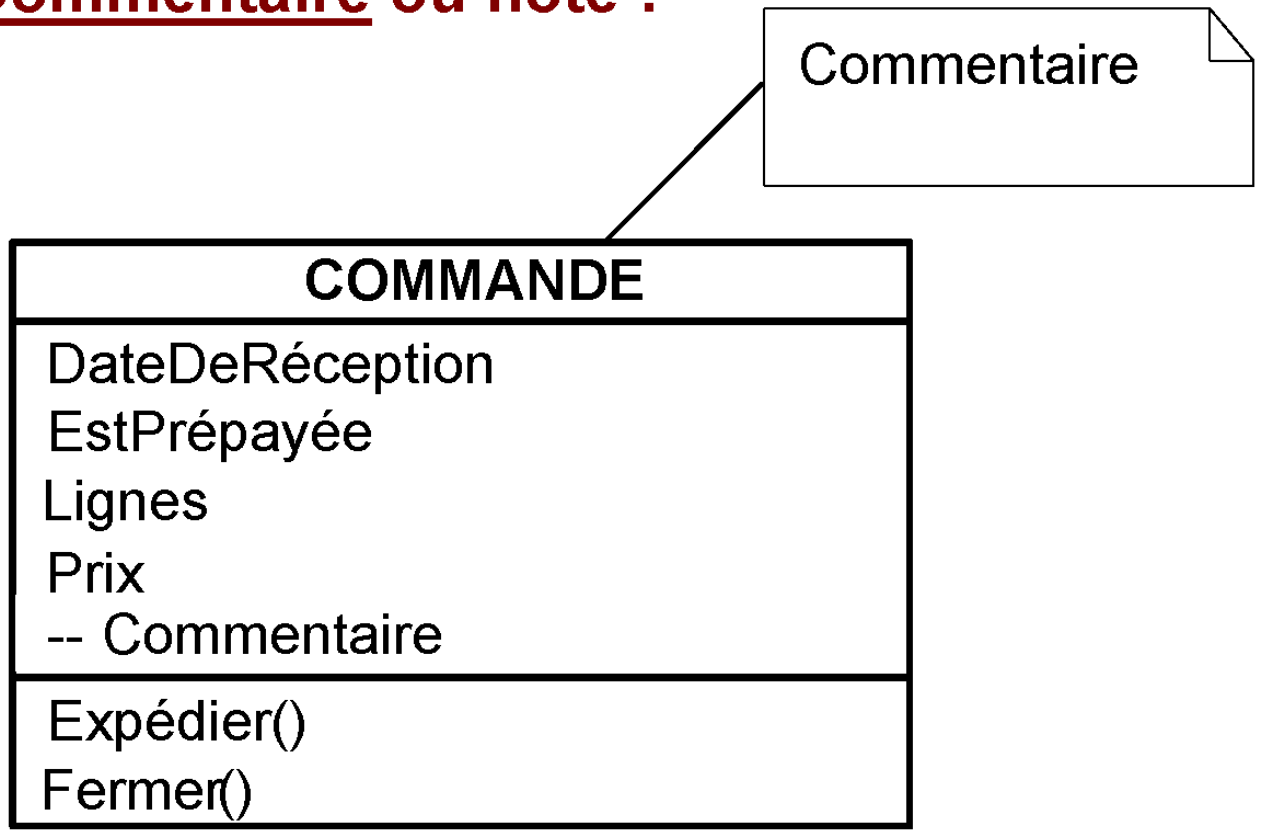
Concepts de base

- **Exemple :**



Concepts de base

- **Commentaire ou note :**



Concepts de base

- La syntaxe de description des attributs est :

  [Visibilité] NomAttribut [Multiplicité]
  [: Type [=Valeur Initiale] [{Propriété}]*]

  ✓ **Visibilité = type d'accessibilité :**

  + : public, visible et modifiable par tout objet du même paquetage.
  - : private, seulement visible et modifiable par les opérations de l'objet auquel il appartient.
  # : protected, seulement accessible et modifiable par les opérations des classes descendantes.

Concepts de base

- ✓ **Multiplicité : intervalle ou nombre**

  Multiplicité := (Intervalle|nombre)

- ✓ **Le type des attributs peut être :**
  - ➢ **Un type primitif (supporté par les LP): Entier, chaîne, …**
  - ➢ **Une classe (type utilisateur) : BOUTON, RECTANGLE, …**
  - ➢ **Expression : chaîne de caractères dont la syntaxe est en dehors de la portée d'UML.**

Concepts de base

✓ **Propriété :**

Mutabilité (gelé, variable, ajout Uniquement, …)

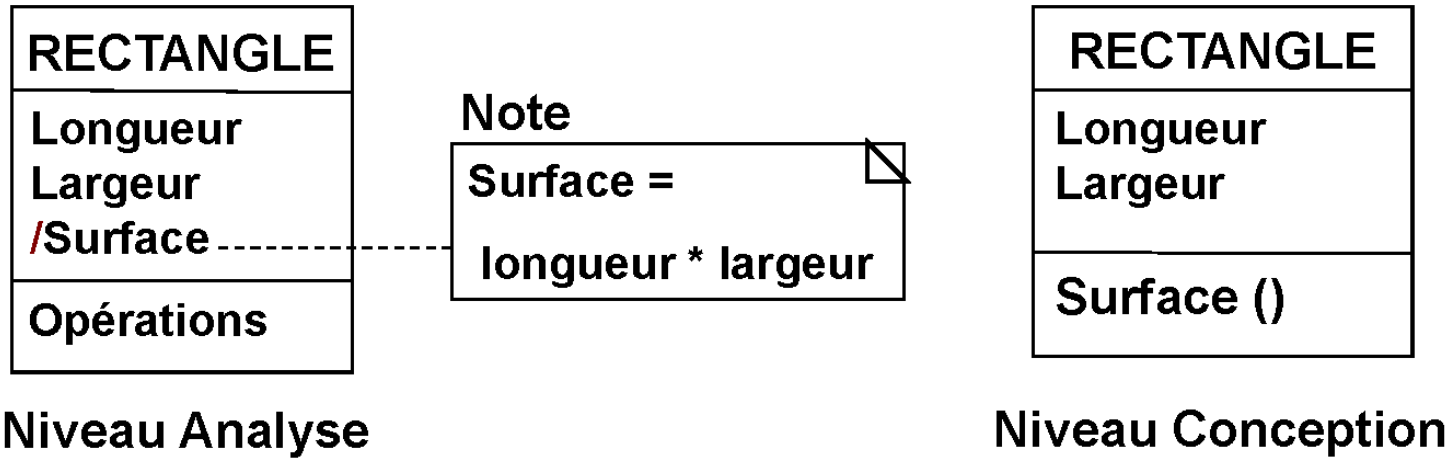
- **Gelé : attribut non modifiable (const de C++).**
- **Variable : attribut modifiable (propriété par défaut).**
- **Ajout Uniquement : seul l'ajout est possible (multiplicité >1).**

Contrainte :

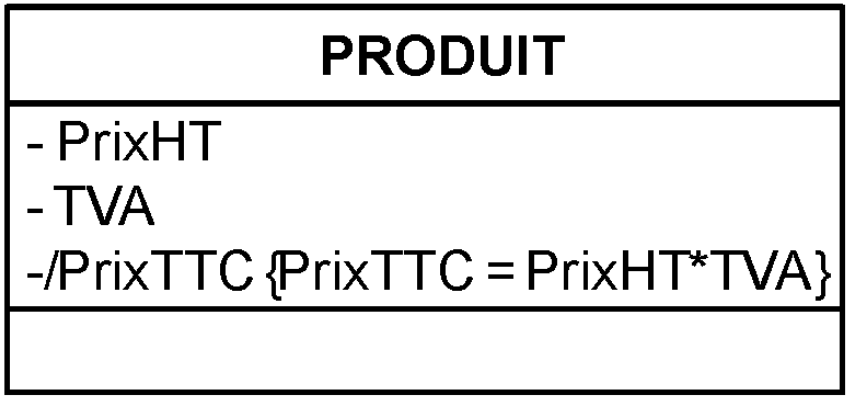
Exemple : - num_sec_soc:string[10] = " " {readOnly}

Concepts de base

- **Un attribut peut être dérivé (/Attribut) :**
  - ✓ peut être déduit par application d'une formule sur d'autres attributs.
  - ✓ qui conduit en implémentation à une opération.



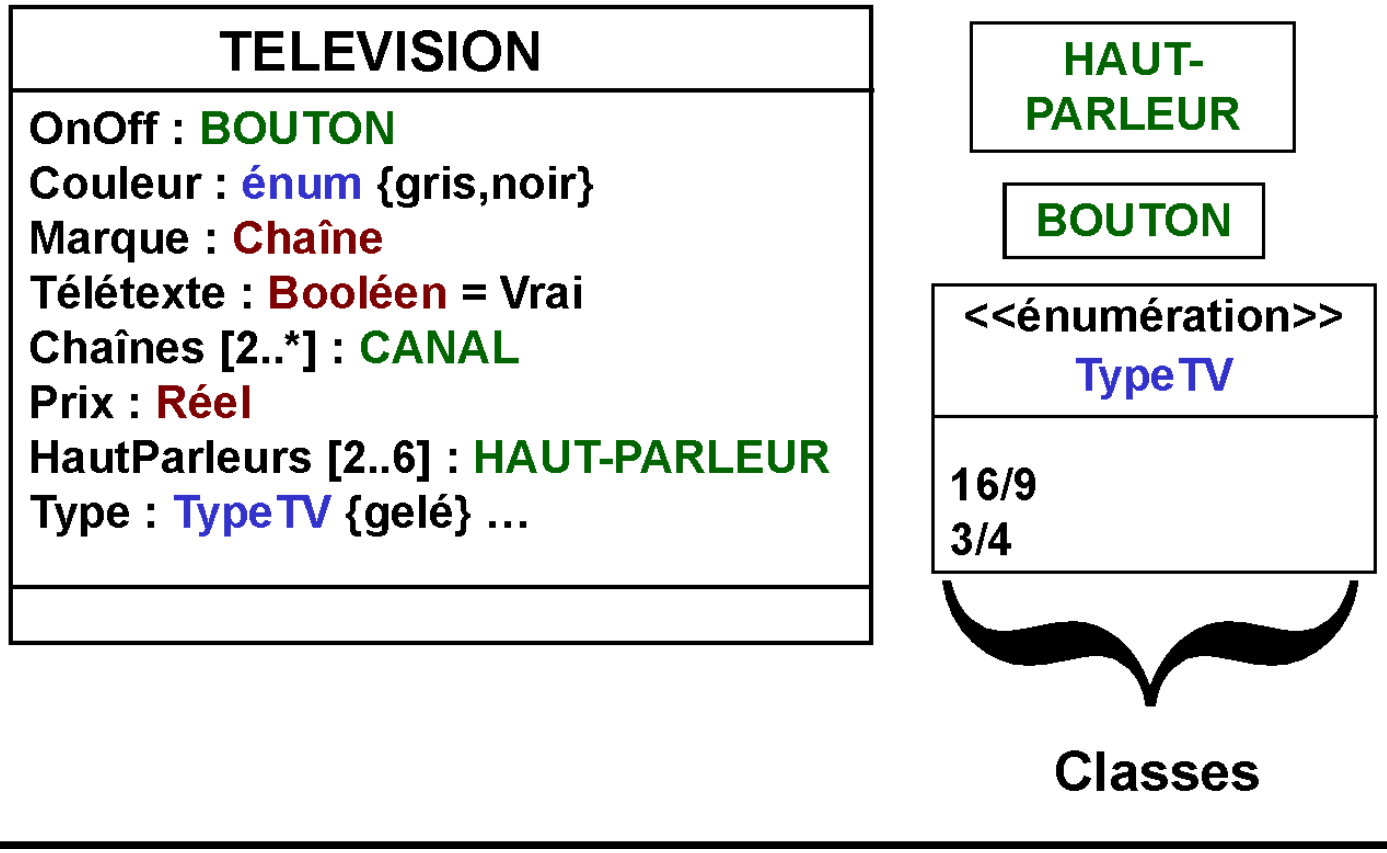
Concepts de base



- **Remarque :**
  - ✓ **Une opération : un service qu'une instance de la classe peut réaliser.**
  - ✓ **Une méthode est l'implémentation d'une opération.**
    - ◆ **Abus de langage : opération = méthode**

Concepts de base

- **Exemple :**



Concepts de base

- **Syntaxe de description des opérations :**

  [Visibilité] NomOpération [[Arguments] :
  TypeRetourné [{Propriété}]]

  Exemple : + fact(n:int) : int {récursive}

- ✓ **Visibilité : +, -, #**
- ✓ **Arguments :**

  Direction NomArgument : TypeArgument
  [= ValeurDefaut]

Concepts de base

Direction (idem PL/SQL) : in, out, inout

in est la valeur par défaut

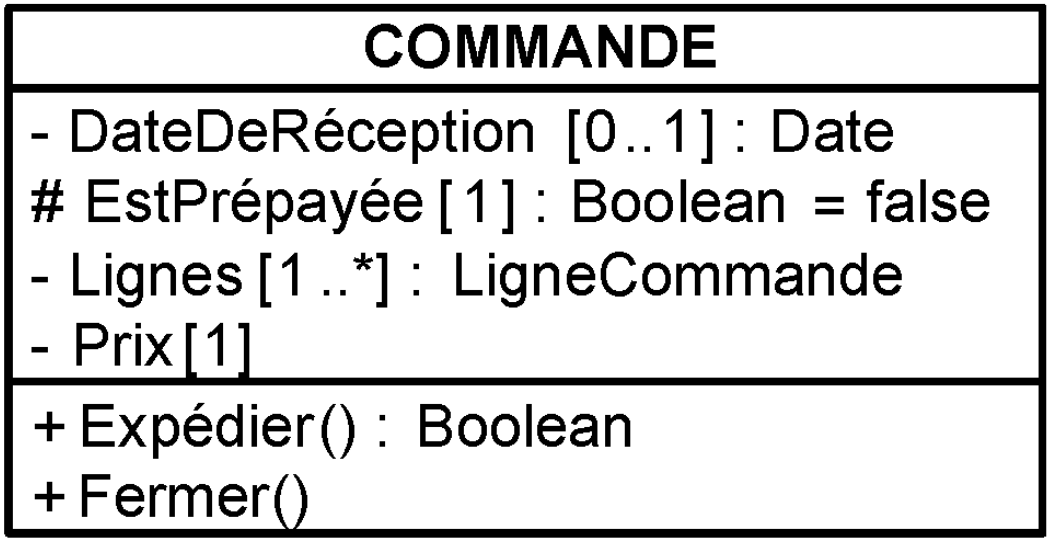
- **In : argument est un paramètre en entrée seule ; non modifié par l'exécution de cette opération.**
- **Out : argument est un paramètre en sortie seule ; l'appelant peut récupérer sa valeur.**
- **inOut : argument est un paramètre en entrée-sortie ; passé à l'opération, modifiable et récupérable.**

Concepts de base

- ✓ **Propriété :**
  - ◆ **requête : l'opération ne modifie pas les attributs ;**
  - ◆ **abstrait : l'opération n'est pas implémentée dans la classe ;**
  - ◆ **estFeuille : l'opération ne peut pas être redéfinie ;**
  - ◆ **estRacine : l'opération est définie pour la première fois dans la hiérarchie ;**
  - ◆ **récursive : l'opération est récursive ;**
  - ◆ **…**

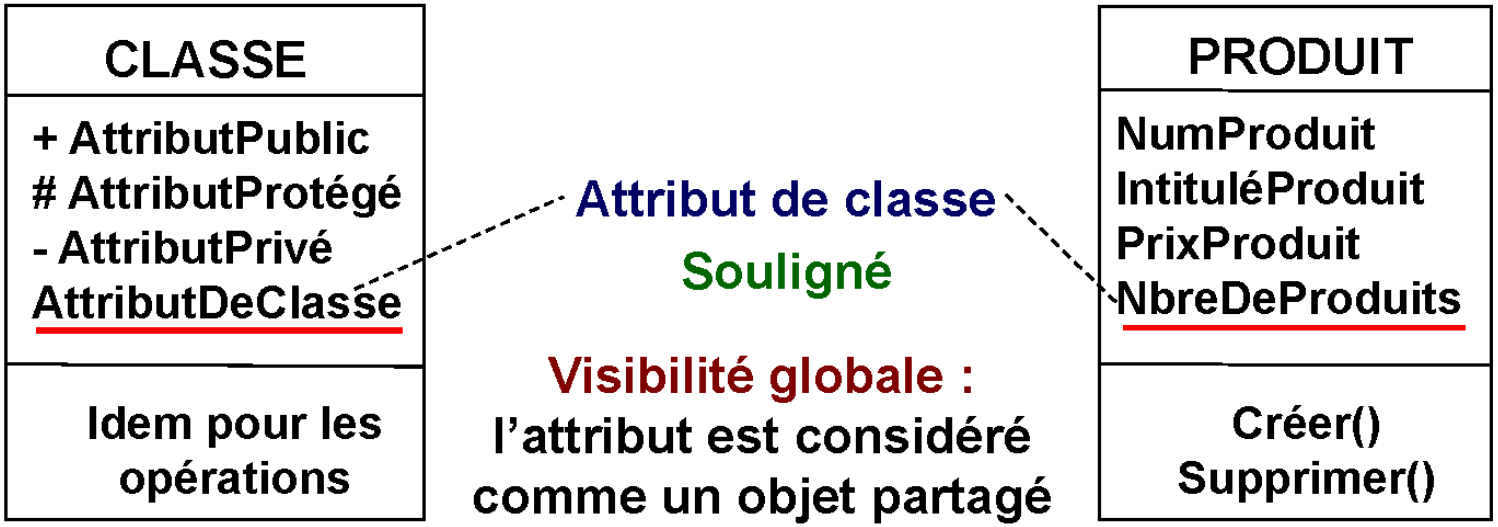
Concepts de base

- ✓ **Représentation détaillée (conception)**



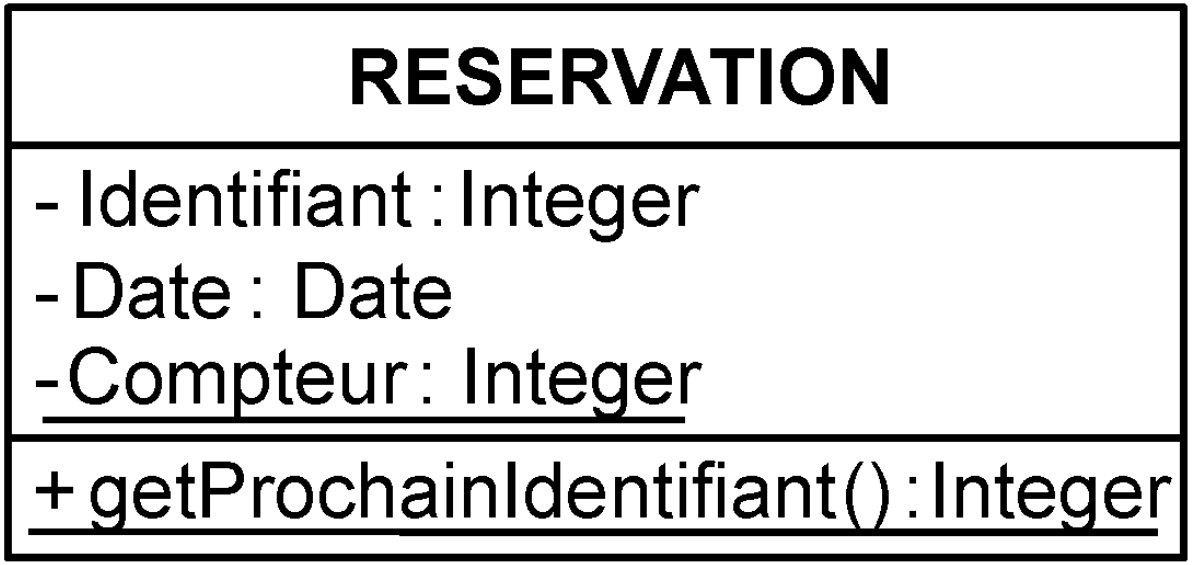
Concepts de base

- **Visibilité et portée des attributs et des opérations statiques :**



Concepts de base

- **Correspondent aux membres static en C++ ou Java**



Concepts de base

- **Les relations entre classes :**
  - ✓ **Association.**
  - ✓ **Agrégation.**
  - ✓ **Composition.**
  - ✓ **Héritage.**

Remarque : par rapport au modèle E/A de base, les Représentations Conceptuelles UML :

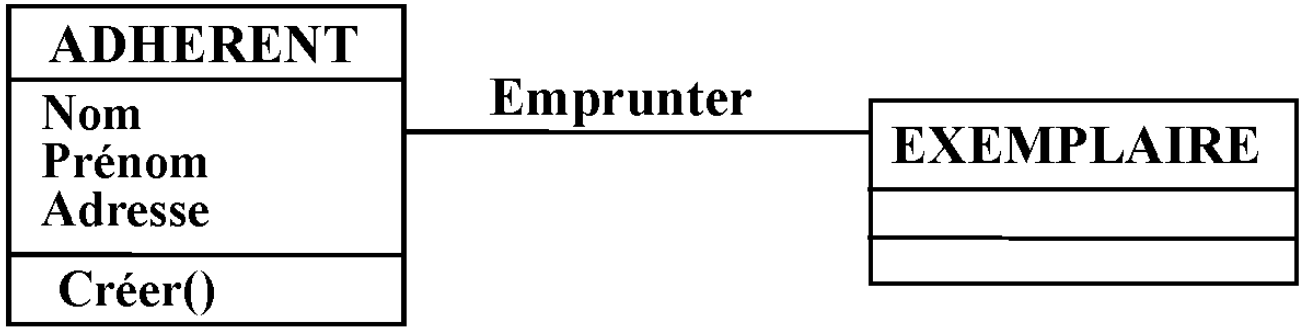
+ riches sémantiquement et

+ proches de la réalité.

Concepts de base

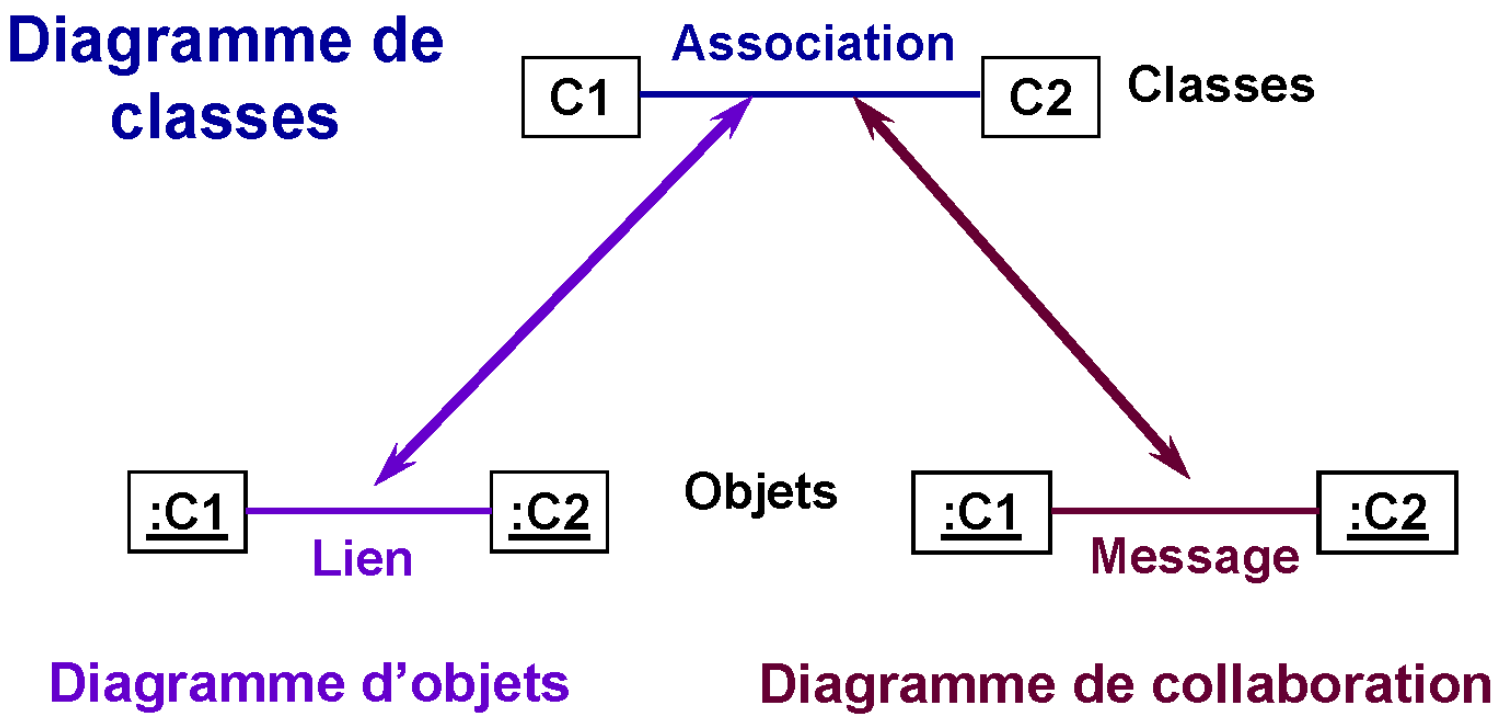
❑ Les associations :

- ✓ Une association exprime une connexion sémantique bidirectionnelle entre n classes (n>=1).



- ✓ Une association est instanciable dans un diagramme d'objets ou de collaboration sous forme de liens ou messages entre objets issus des classes associées.

Concepts de base



Concepts de base

- **Les associations :**
  - ✓ **Par rapport au modèle Entité/Association :**

Diagramme Entité / Association

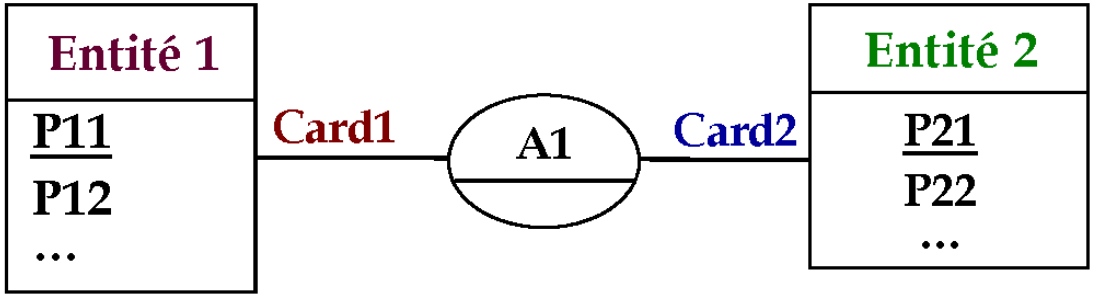
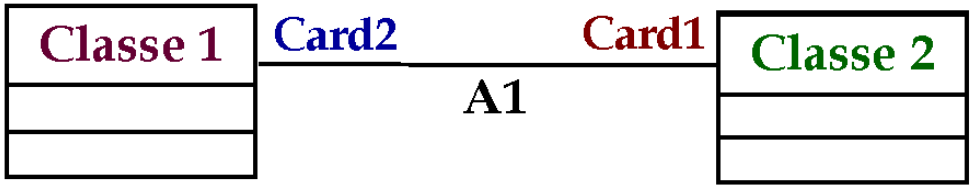
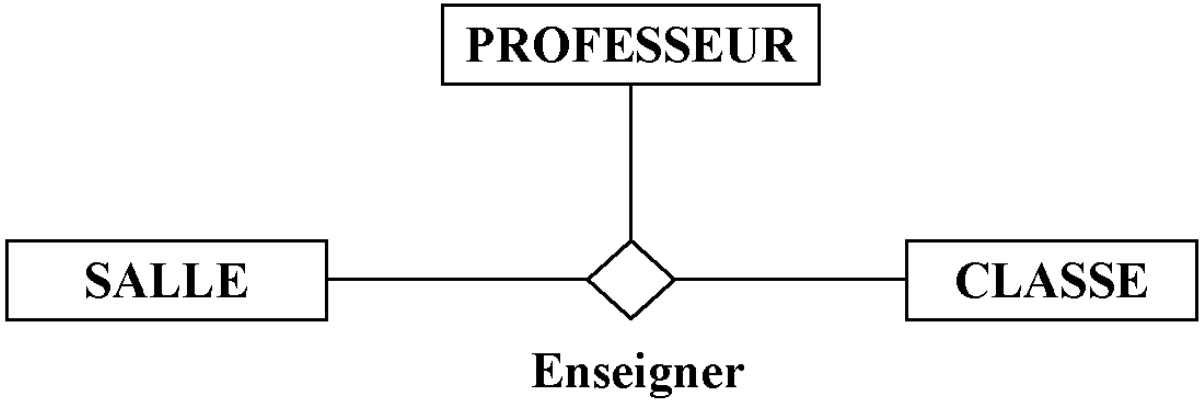


Diagramme de classes



Les associations

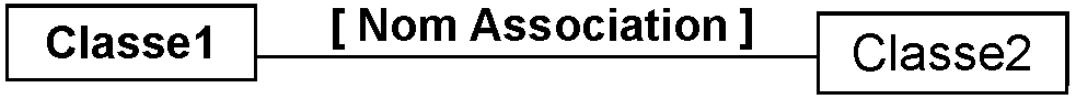
- **Arités des associations :**
  - ✓ Une association peut être binaire ou n-aire (à éviter).
  - ✓ Exemple : on désire représenter le fait suivant : Un professeur enseigne dans une salle des étudiants d'une classe.



Les associations

❑ Nommage des associations

- ✓ Une association peut être nommée : c'est optionnel.



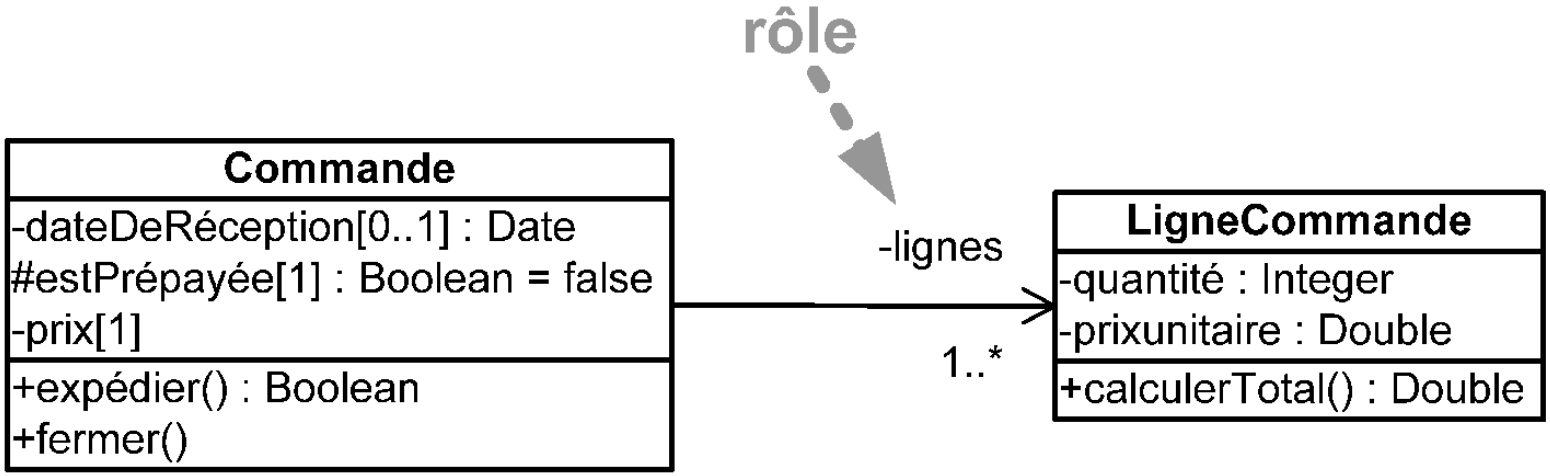
- ✓ Les noms peuvent être en forme verbale active ou passive
- ✓ Le sens de lecture d'une association peut être précisé lorsqu'il est ambigu :

Les associations

- **Association à navigabilité restreinte :**
  - ✓ **Par défaut, une association est navigable dans les deux sens.**
  - ✓ **On peut la limiter à un seul sens dans un modèle**

    ➔ **indique que les instances d'une classe « ne voient pas » les instances de l'autre.**

Les associations



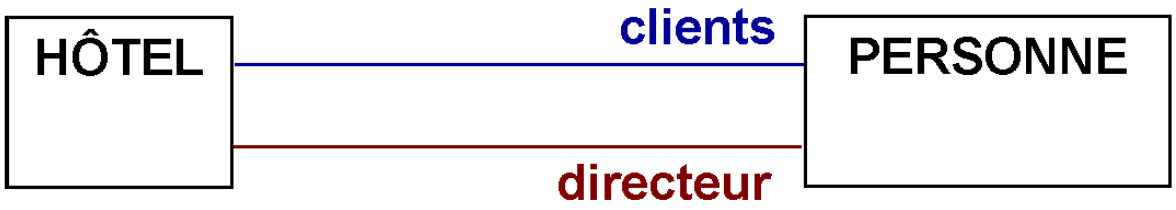
Les associations

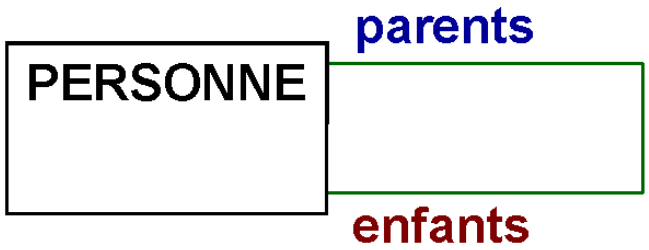
- **La notion de rôle :**
  - ✓ **L'extrémité d'une association peut avoir un nom, appelé rôle, qui décrit comment une classe source voit une classe destination au travers de l'association.**

  - ✓ **Rôle 1 : le rôle joué par Classe 1 dans l'association**
  - ✓ **Rôle 2 : le rôle joué par Classe 2 dans l'association**

Les associations

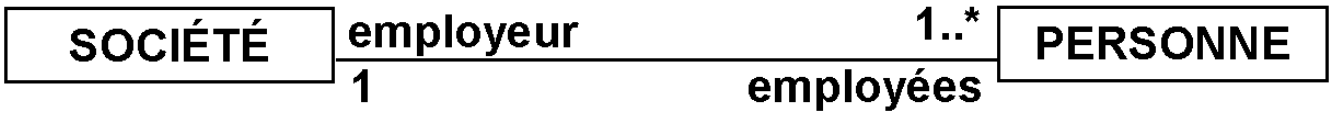
- ✓ **L'indication des rôles est nécessaire pour les associations ambiguës.**





Les associations

- **Les multiplicités (cardinalités)**
  - ✓ précisent le nombre d'objets d'une classe qui peuvent être liés à un objet de l'autre.



Valeurs de cardinalité conventionnelles

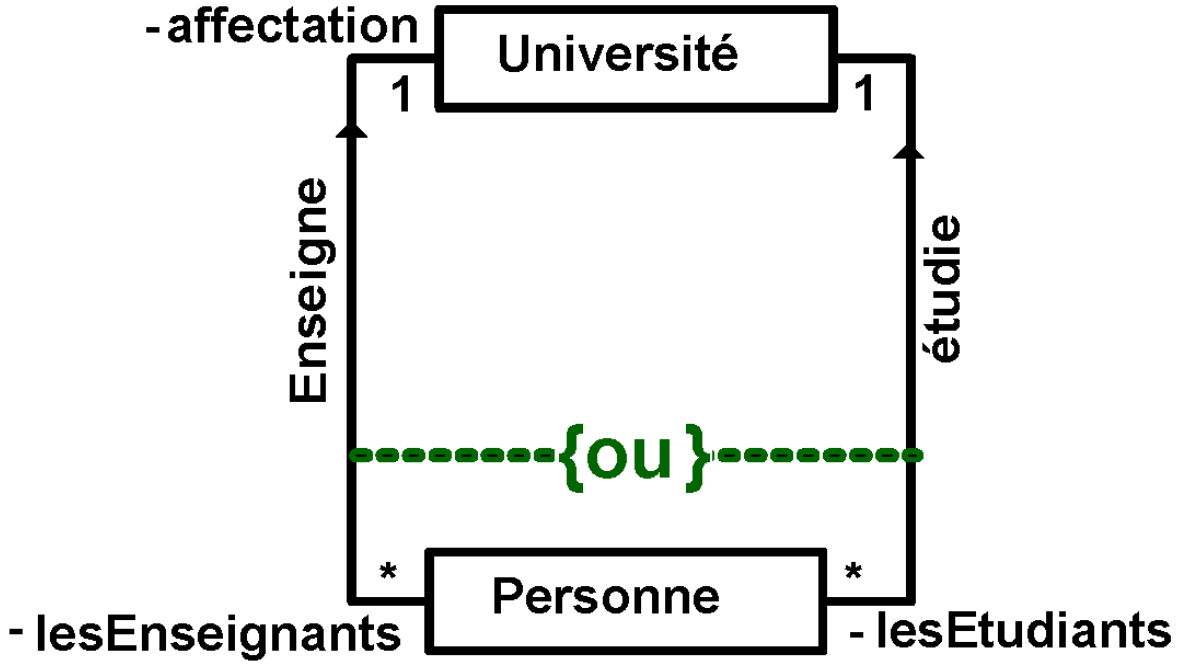
1	Un et un seul
0 .. 1	Zéro ou un
N	N (entier naturel)
M .. N (3..7)	De M à N (entiers naturels)
*	De 0 à plusieurs
0 .. *	De 0 à plusieurs
1 .. *	De 1 à plusieurs

Les associations

- ✓ **Les types de contraintes exprimables sur les associations :**
  - ➢ **Ordonné ;**
  - ➢ **Sous-ensemble ;**
  - ➢ **Ou ;**
  - ➢ **Partition (Ou-exclusif) ;**
  - ➢ **…**
- ✓ **Elles sont placées entre accolades.**

Les associations

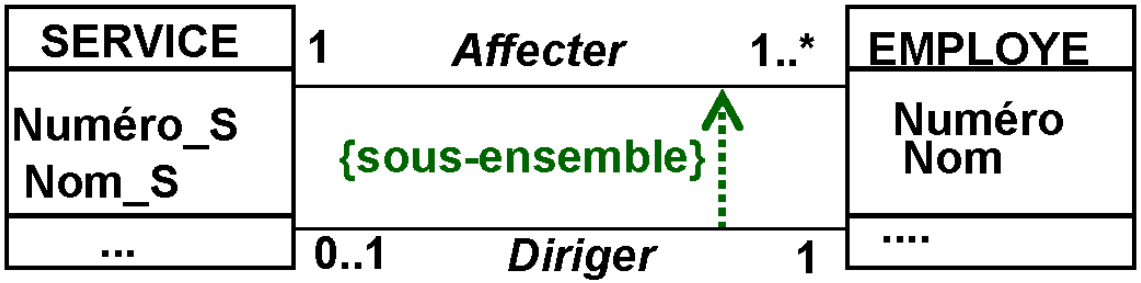
✓ **Exemples :**



Les associations

On désire représenter les règles de gestion suivantes :

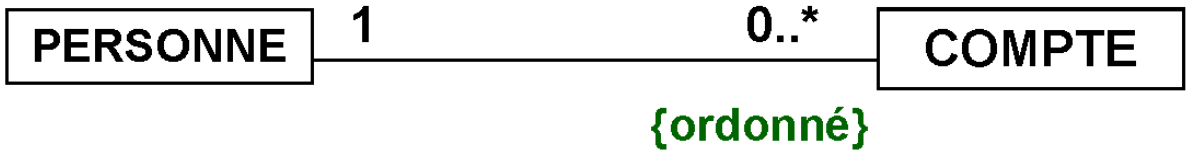
1. Un employé est affecté à un seul service.
2. Plusieurs employés sont affectés à un service.
3. Un service est dirigé par un seul employé.
4. Le directeur d'un service est obligatoirement l'un des employés affectés à ce service.



Toute instance de *Diriger* est aussi instance de *Affecter*

Les associations

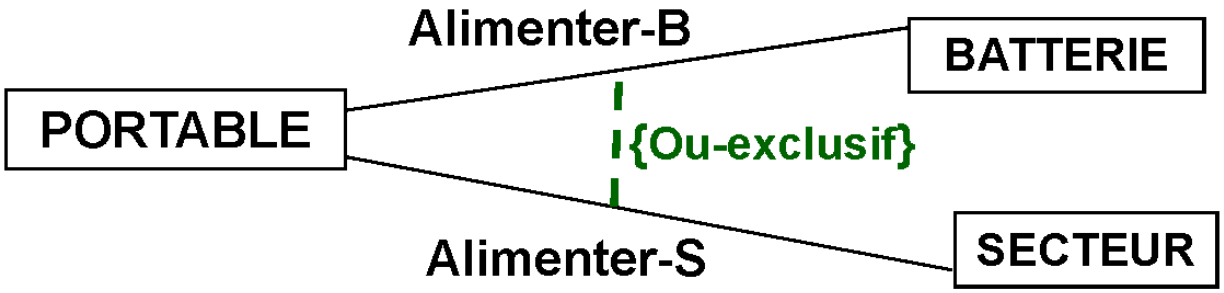
Ordonné :



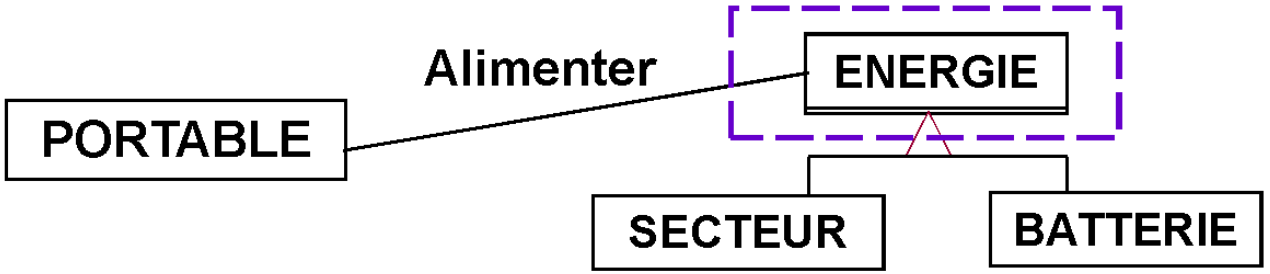
➔ La collection des comptes d'une personne est triée

Les associations

Ou-exclusif : Indique que pour un objet donné, une seule association est valide.

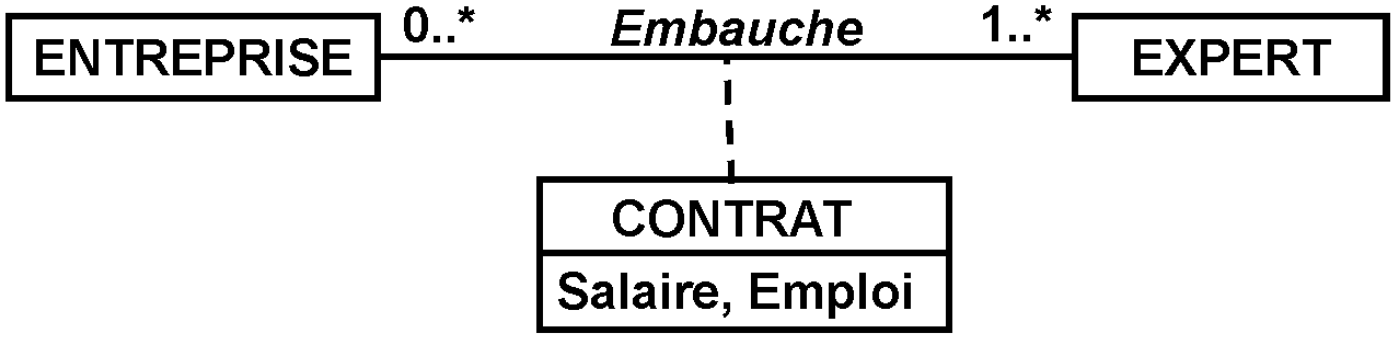


Cette contrainte permet d'éviter l'introduction de classes artificielles



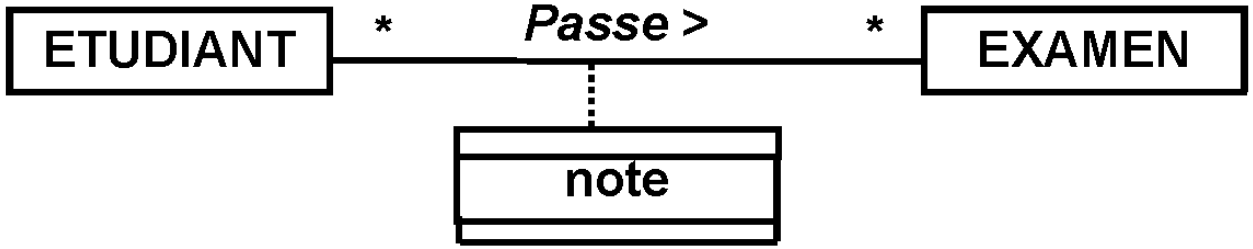
Les classes d'association et Attributs de lien

- **Une classe d'association :**
  - ✓ Permet de représenter une association par une classe pour définir des attributs et/ou des opérations dans l'association.
  - ✓ Possède les caractéristiques d'une classe et d'une association.



Les classes d'association et Attributs de lien

- **Un attribut de lien :**

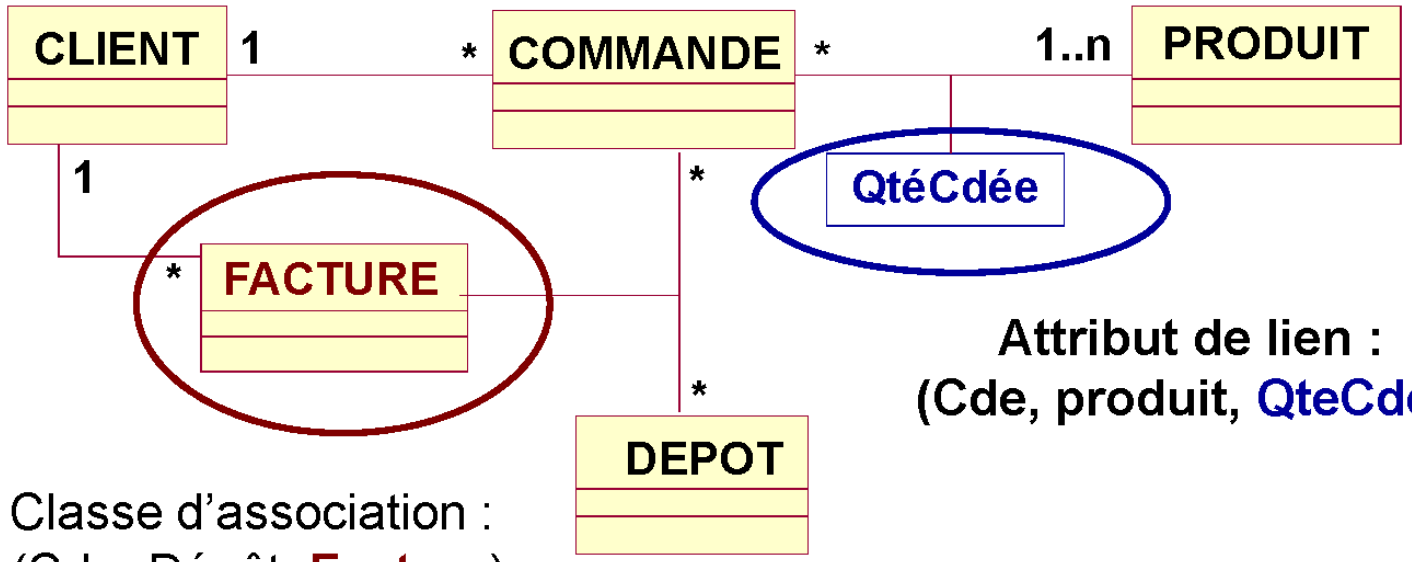


Les classes d'association

Considérons les règles de gestion suivantes :

- ✓ **Un client passe une ou plusieurs commandes.**
- ✓ **Une commande est passée par un seul client et concerne un ou plusieurs produits.**
- ✓ **Un produit peut être commandé par plusieurs commandes.**
- ✓ **Chaque commande est envoyée à un ou plusieurs dépôts pour être satisfaite.**
- ✓ **Chaque dépôt ayant satisfait une partie d'une commande, génère une facture correspondant à la partie satisfaite.**
- ✓ **Toute facture générée sera envoyée au client correspondant.**

Les classes d'association



Attribut de lien :
(Cde, produit, QteCdée)

Classe d'association :
(Cde, Dépôt, **Facture**)

QteCdée est un simple attribut

Facture est une classe à part entière :
elle a ses propres attributs, opérations et liens

L'agrégation

- **L'agrégation est une association non symétrique :**
  - ✓ **exprime un couplage fort et une relation de subordination.**
  - ✓ **représente une relation de type « ensemble/élément »**

Exemple:

L'agrégation

Une instance d'élément agrégé peut :

- ✓ **être liée à plusieurs instances d'autres classes :**
  - ◆ l'élément agrégé peut être partagé ;
- ✓ **exister sans agrégat (et inversement) :**
  - ◆ les cycles de vie de l'agrégat et de ses éléments agrégés peuvent être indépendants :
    - ➔ La création (ou la suppression) de l'un n'implique pas celle de l'autre.

La composition

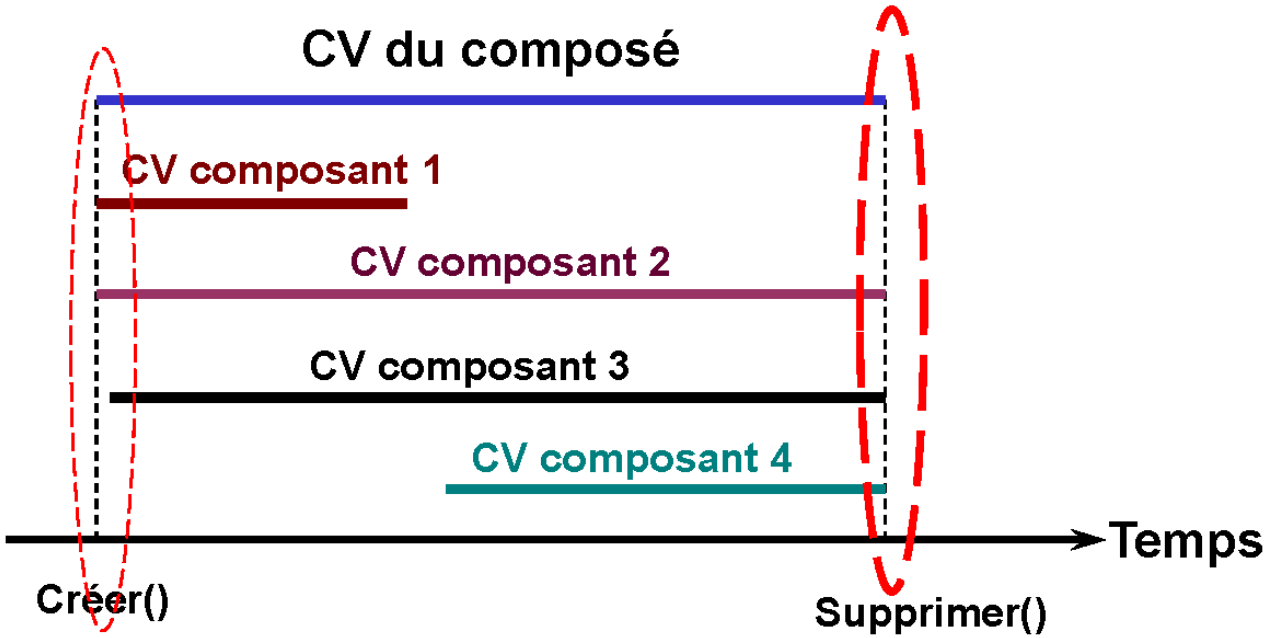
- La composition est une agrégation forte qui exprime « une partie de ».

Les cycles de vies (CV) des composants et du composé ne sont pas indépendants :

- ✓ Si le composé est détruit (ou copié), ses composants le sont aussi.
- ✓ Une instance de composant ne peut être créée qu'avec ou après la création du composé. Elle ne peut être liée qu'à un seul composé.
- ✓ Les "objets composites" sont des instances de classes composées.

La composition

✓ **Relations entre les CV des objets :**



Remarque : toutes les conventions relatives aux cardinalités restent valables pour les agrégations et les compositions.

Association, agrégation et composition

- **Exemple récapitulatif :**
  - ✓ Une personne possède un immeuble.
  - ✓ Dans un immeuble, on trouve un ascenseur.
  - ✓ Un immeuble est composé d'étages.
  - ✓ Une personne possède un compte et une adresse.

Explications :

- ✓ **Une personne possède un immeuble :**
  - ◆ Un lien conceptuel : les objets ont des CV indépendants.
  - ◆ Ce lien exprime une relation temporaire.

Association

Association, agrégation et composition

- ✓ **Dans un immeuble, on trouve un ascenseur :**
  - **Un lien : ensemble/élément, les CV des objets non disjoints.**
  - **la suppression de l'immeuble n'entraîne pas obligatoirement celle de l'ascenseur.**
  - **Un ascenseur ne peut être utilisé (au même temps) par plus qu'un immeuble. Mais, dans le temps, le même ascenseur peut être utilisé par différents immeubles.**

Agrégation

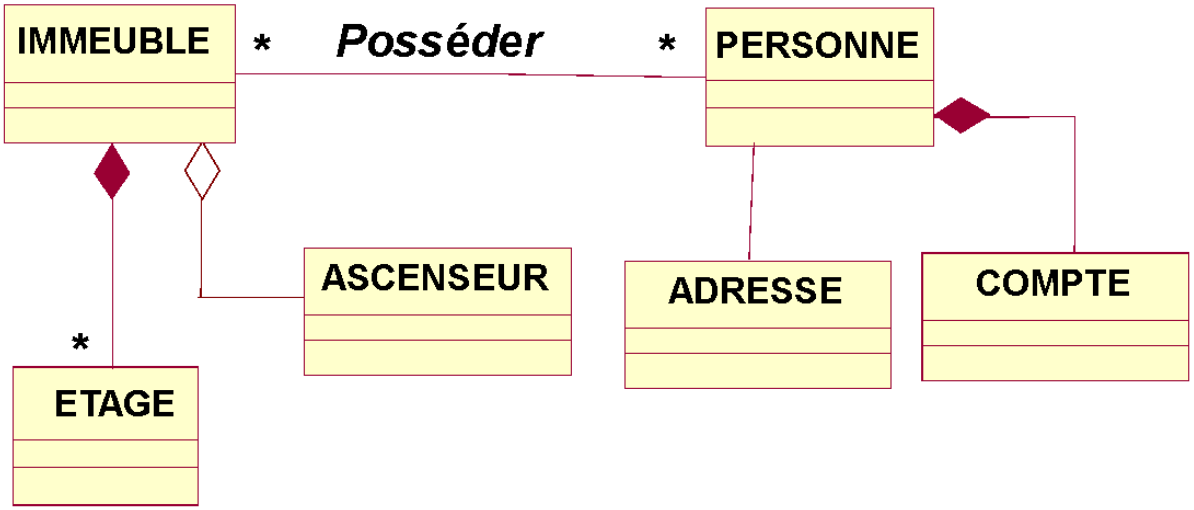
Association, agrégation et composition

- ✓ **Un immeuble est composé d'étages :**
  - Un lien : composé/composants : les CV des objets coïncident.
  - La création de l'immeuble → la création de ses étages.
  - La suppression de l'immeuble → la suppression de ses étages.
  - Un étage ne peut pas être partagé par différents immeubles.

Composition

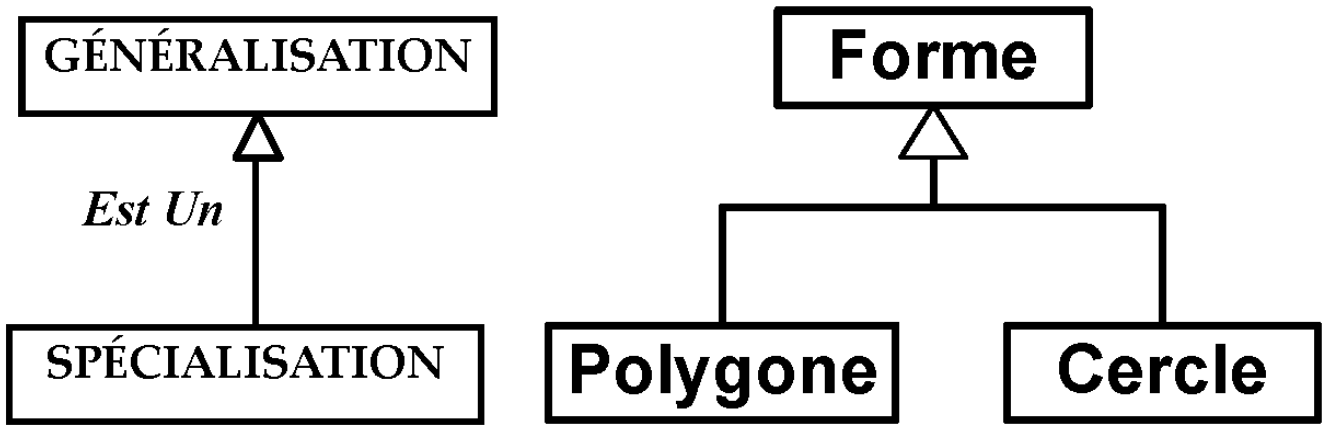
Association, agrégation et composition

Diagramme de Classes :



La généralisation / spécialisation

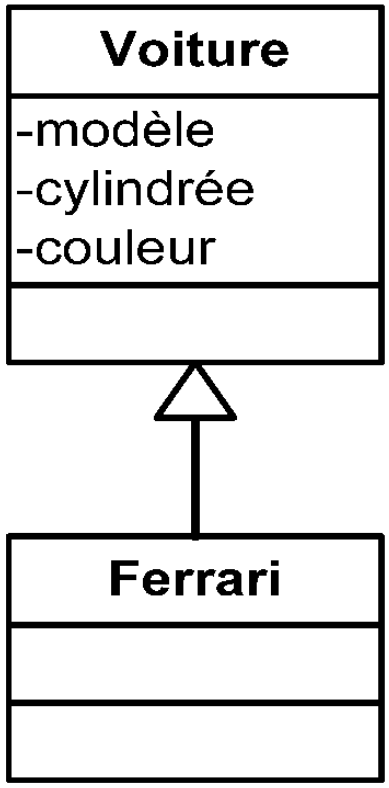
- L'héritage, avec UML, est désigné par Généralisation
  - ✓ La généralisation peut être :
    - Simple



    - Multiple : La spécialisation a plus d'une généralisation

La généralisation / spécialisation

Attention : à ne pas confondre héritage et instanciation



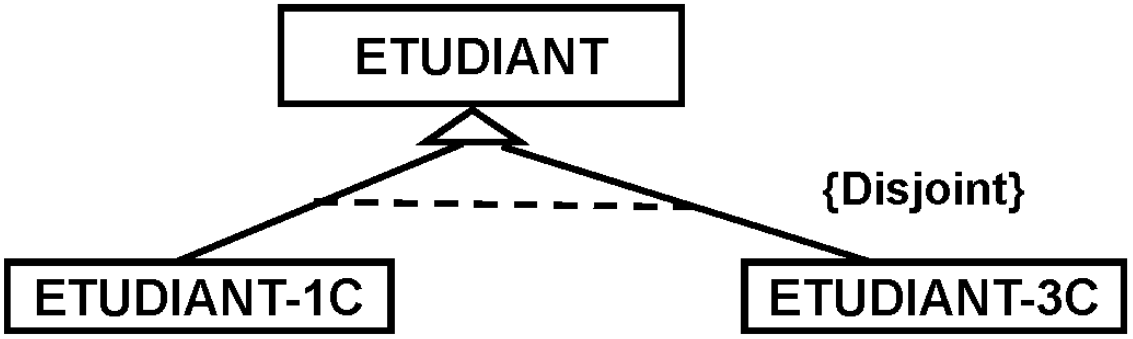
NON !

La généralisation

Contraintes et propriétés de la généralisation :

- ✓ La contrainte exprimée par le mot-clé {Disjoint}
  - ◆ Tout objet est au plus instance d'une seule sous-classe.
  - ◆ C'est une décomposition exclusive : {Exclusif}

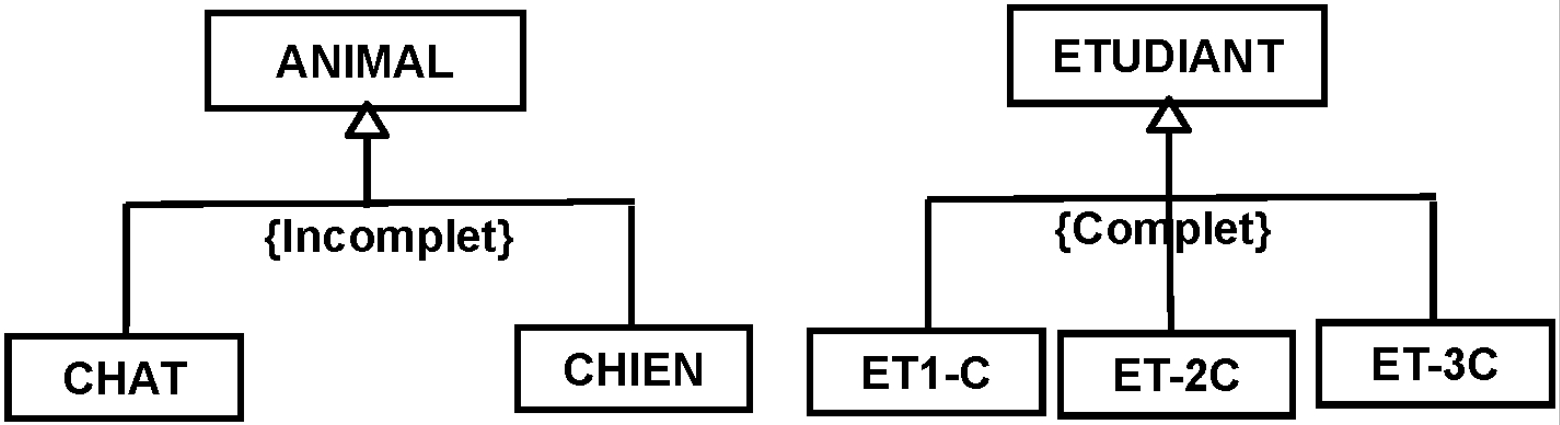
  C'est l'option par défaut.



Inter (ETUDIANT-1C, ETUDIANT-3C) = Vide

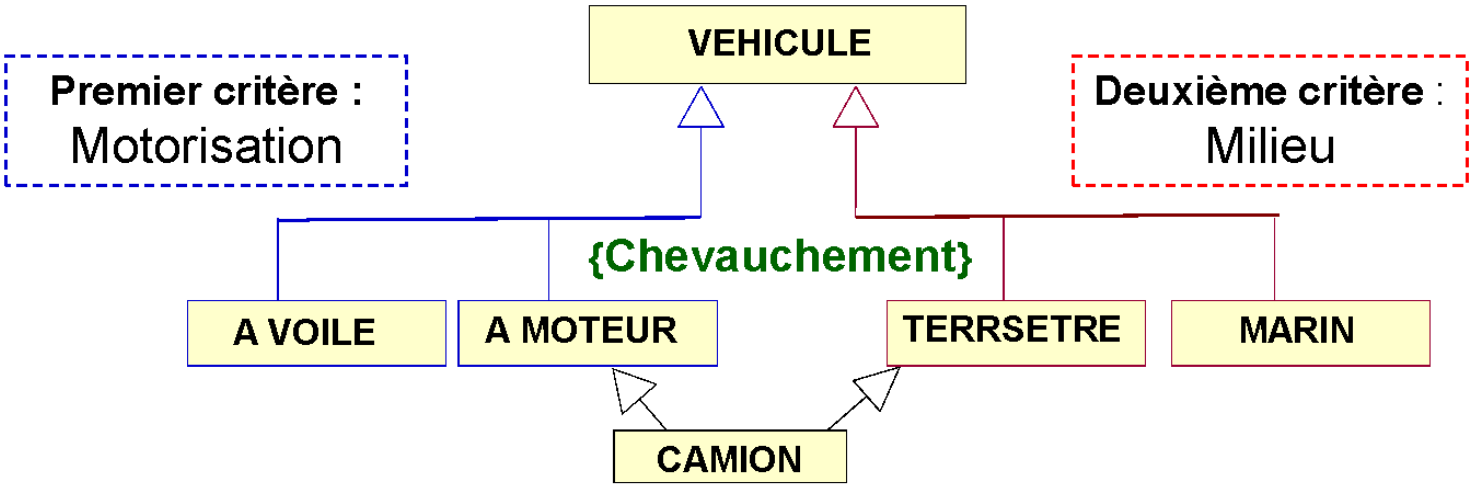
La généralisation

- ✓ **La contrainte exprimée par le mot-clé {Complet}**
  - ◆ Indique que la spécialisation est terminée (couverture) :
    - ➔ Il n'est pas possible d'ajouter d'autres sous-classes.
- ✓ **La contrainte exprimée par le mot-clé {Incomplet}**
  - ◆ Indique que la spécialisation est extensible : elle peut avoir d'autres sous classes.



La généralisation

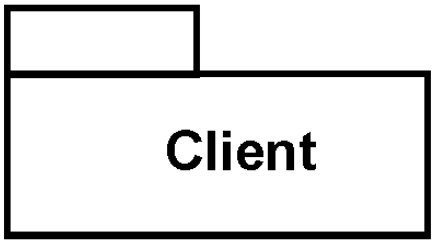
- ✓ **Une classe générique peut être spécialisée selon différents critères.**
- ✓ **La contrainte exprimée par le mot-clé {inclusif} ou {chevauchement} ou {overlapping}**
  - ◆ **Une instance de l'une des spécialisations peut être simultanément une instance d'une autre.**



Les paquetages

Un paquetage (ou package) regroupe des éléments de modélisation (EM) : classes, associations et packages.

Notation :



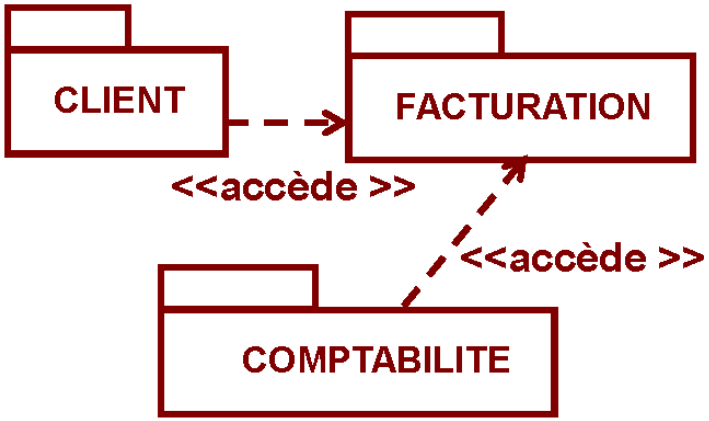
- ✓ **Un paquetage contient un sous ensemble d'un modèle.**
- ✓ **La décomposition de modèles en paquetages se fait selon un critère purement logique.**
- ✓ **L'objectif de la décomposition en paquetages est d'avoir une cohérence forte entre éléments d'un même paquetage et un couplage faible entre paquetages.**

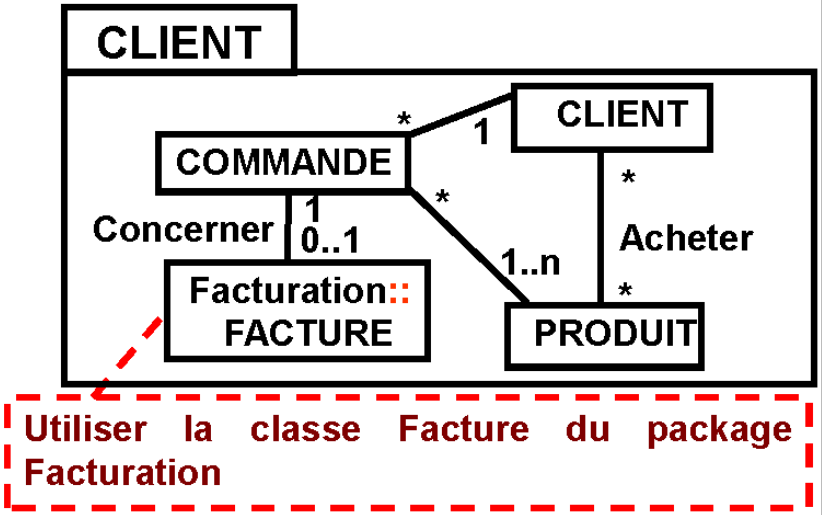
Les paquetages

- ✓ La hiérarchie des paquetages et les relations entre eux décrivent l'architecture du système.
- ✓ Deux éléments de modélisation contenus dans deux packages différents peuvent porter le même nom.
- ✓ Deux éléments de modélisation ayant le même nom, contenus dans des packages différents ne sont pas identiques.
- ✓ Les éléments contenus dans un package possèdent chacun un nom unique.

Les paquetages

- **Les relations de dépendance entre les packages sont de deux types :**
  - ✓ **<<importe>>: ajoute les éléments du package destination au package source.**
  - ✓ **<<accède >>: permet de référencer des éléments du package destination.**





Exercice 1

Elaborer le diagramme de classes correspondant à chacune de ces phrases:

- Un répertoire contient des fichiers
- Une pièce contient des murs
- Les modems et claviers sont des périphériques d'entrée / sortie
- Une transaction boursière est un achat ou une vente
- Un compte bancaire peut appartenir à une personne physique ou morale

Exercice 2

Enoncé:
On se propose d'élaborer le diagramme de classes relatif à la présentation des exercices d'un livre.
Chaque exercice comporte un numéro, un type (Construction MCD, construction MCT, déduction MLD, Optimisation, ...), un niveau de difficulté, le (ou les) nom(s) et prénom(s) de son (ses) auteur(s), le numéro de page de début et le nombre de pages.
En plus de son code, chaque type est décrit par un libellé complet.
Par ailleurs, une estimation de la durée de résolution est donnée par type d'exercice et par niveau de difficulté.

Travail demandé:
Construire ce diagramme de classes