MODÉLISATION DE LA STRUCTURE

DIAGRAMMES DE CLASSES

DIAGRAMMES DE PACKAGES

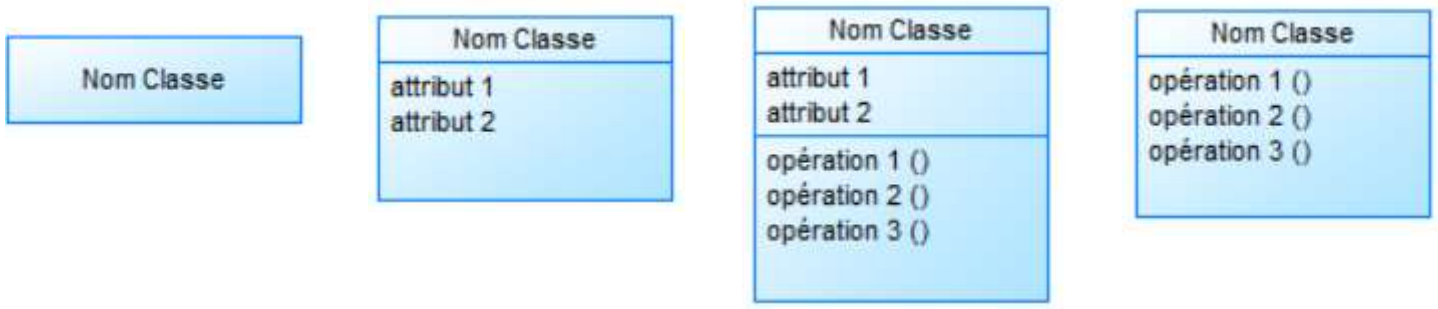
DIAGRAMMES D'OBJETS

DIAGRAMME DE CLASSES

- **Un diagramme de classes exprime la structure statique du système.**
- **un diagramme de classes est constitué d'un ensemble de classes,** d' interfaces, de packages et leurs relations, et qui fournit une vue logique de tout ou partie d'un système informatique.

LES CLASSES

- Définition: Une **classe** est une description abstraite d'un type d'objets.
- Le nom d'une classe abstraite doit être en italique.
- Représentation graphique: une classe est représentée par un rectangle contenant un ou plusieurs compartiments



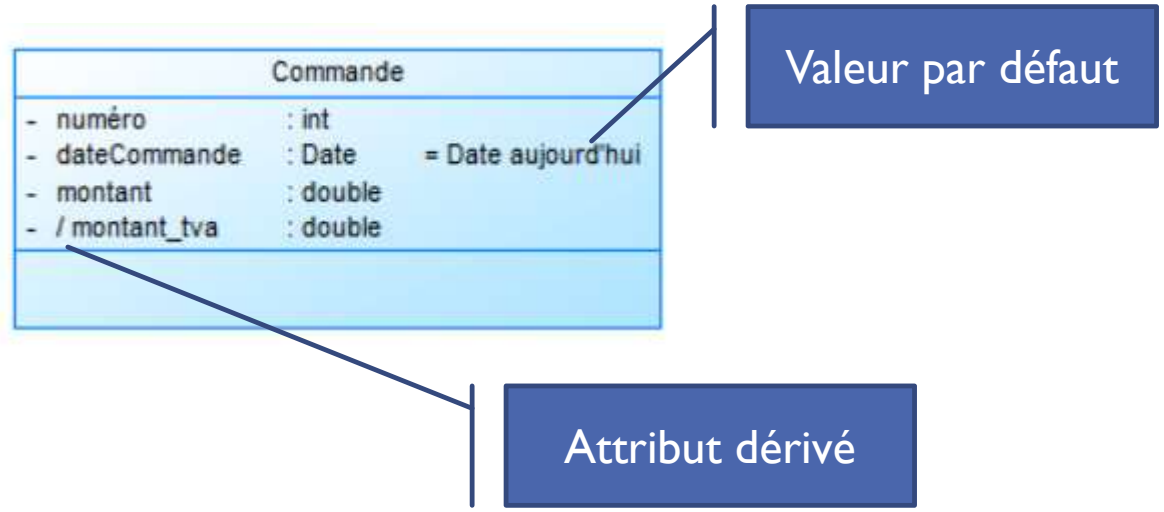
VISIBILITÉ DES MEMBRES D'UNE CLASSE

Niveau de visibilité	**Symbole**	**Définition**
Public	+	Élément visible pour tous les utilisateurs de la classe concernée
Protégé	#	Élément visible au sous élément de la classe concernée (classes dérivées)
Privé	-	Élément visible uniquement pour la classe concernée
package	~	Elément visible dans toutes les classes du package auquel appartient sa classe.

A) ATTRIBUTS

- Spécification d'un attribut:
- Visibilité attribut:type[= valeur initiale]
- Un attribut précédé du symbole / est un attribut dérivé.

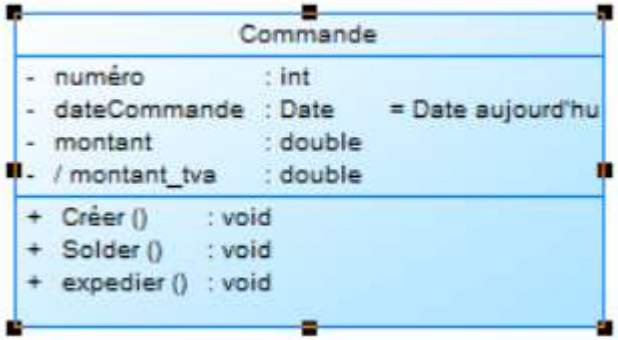
- Le type d'un attribut peut être:
  - Un type primitif: entier , réel, …
  - Un type complexe : tableau, enregistrement ou une classe



LES OPÉRATIONS

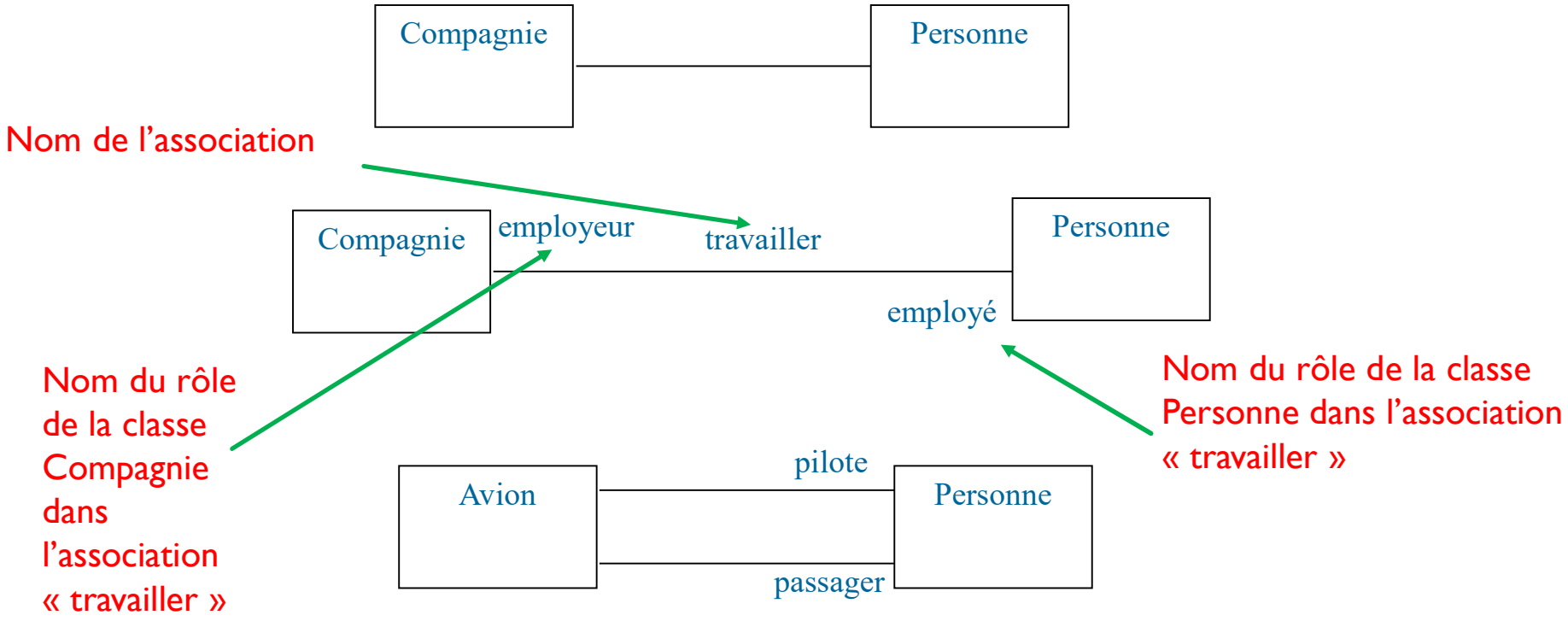
- Une opération représente les traitements effectués par les objets et les classes.
- Spécification d'une opération :

Visibilité opération([arguments:type[=valeur initiale],]):type de retour



- Le nom d'une méthode abstraite doit être en italique.
- Un membre statique est souligné (méthode ou attribut).

LES ASSOCIATIONS



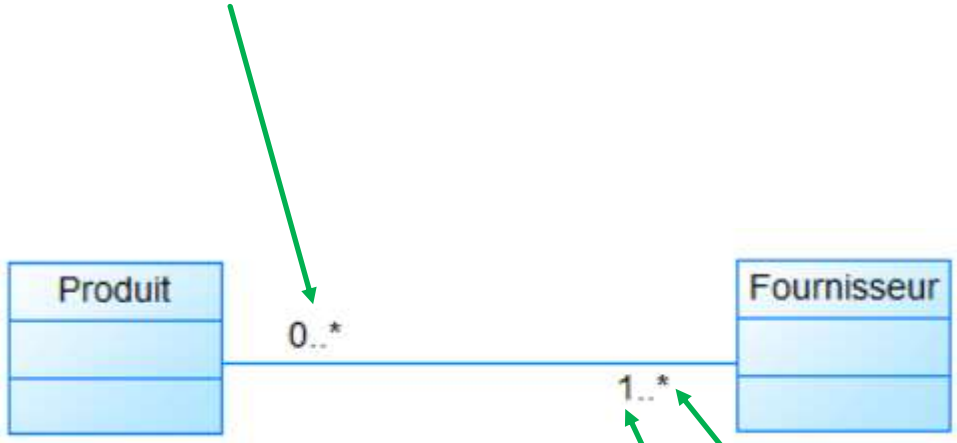
Une association décrit une relations sémantique qui peut lier deux ou plusieurs classes.
UML permet de nommer l'association ou bien le nom de rôle d'une classe dans l'association

LES ASSOCIATIONS

- Les noms de rôle sont aussi appelés terminaisons d'association.
- Une terminaison d'association peut avoir les multiplicités suivantes
  - m..n: au minimum m occurrences et au maximum n occurrences
  - 0..*: 0 ou plusieurs occurrences
  - 1..*: au moins une occurrence.
  - 1: équivalent à 1..1 (par défaut, si aucune multiplicité n'est mentionnée).
  - *: equivalent à 0..*
  - 1,5,8: une liste de valeurs d'occurrences.
- Une association n-aire est représentée par un losange avec un chemin partant vers chaque classe participante dans l'association, le nom de l'association doit être à côté du losange.

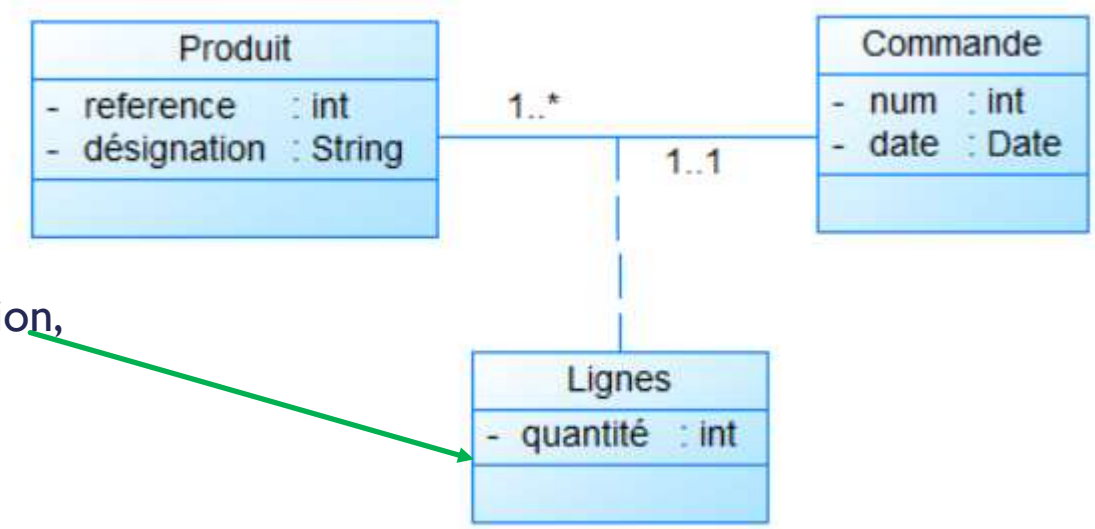
EXEMPLE

- Un fournisseur peut être associé à 0 ou plusieurs produits



- Un produit doit être associé à au moins un fournisseur
- Un produit peut être associé à plusieurs fournisseurs

ATTRIBUTS D'ASSOCIATION

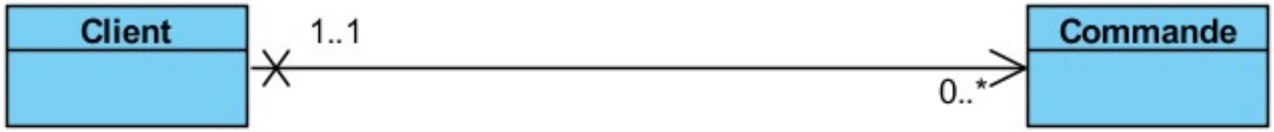


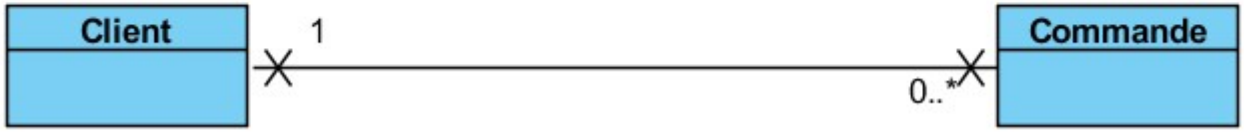
- Classe Association,

  - Dans le cas d'une association n-aire, la classe association doit être liée au losange.

LA NAVIGABILITÉ

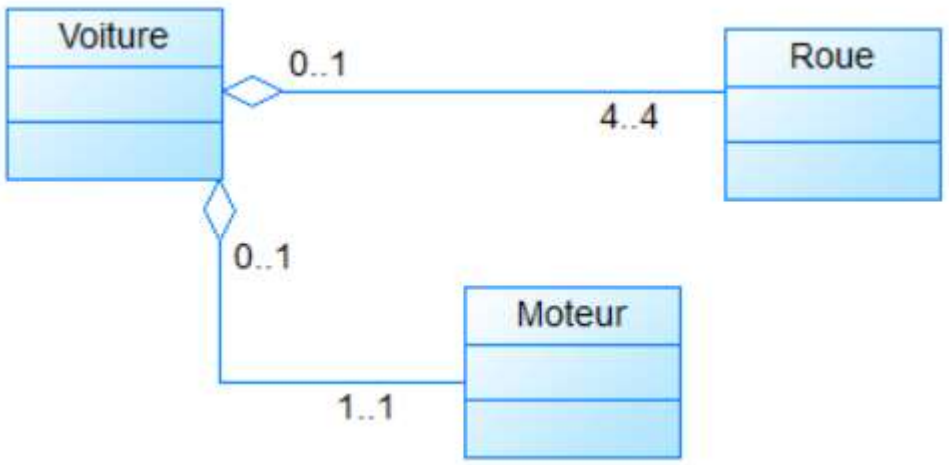
- **navigabilité** indique si l'association fonctionne de manière unidirectionnelle ou bidirectionnelle, elle est matérialisée par deux extrémité fléchées. La non navigabilité se représente par un « X ».





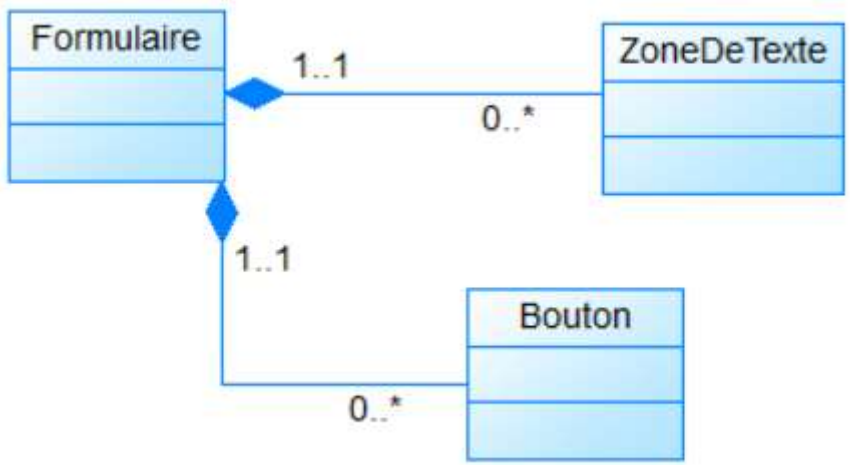
AGRÉGATION

- Agrégation: une forme d'association qui définit une relation tout-partie entre une classe composant et une classe agrégat ou composée, par exemple :
  - Une voiture a un moteur et 4 roues (agrégat: Voiture, Composants: Moteur, Roue)
  - Familles et enfants

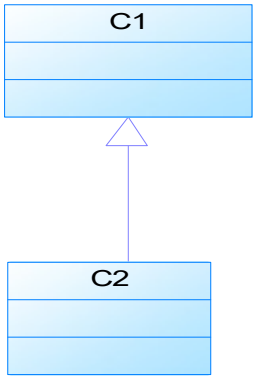


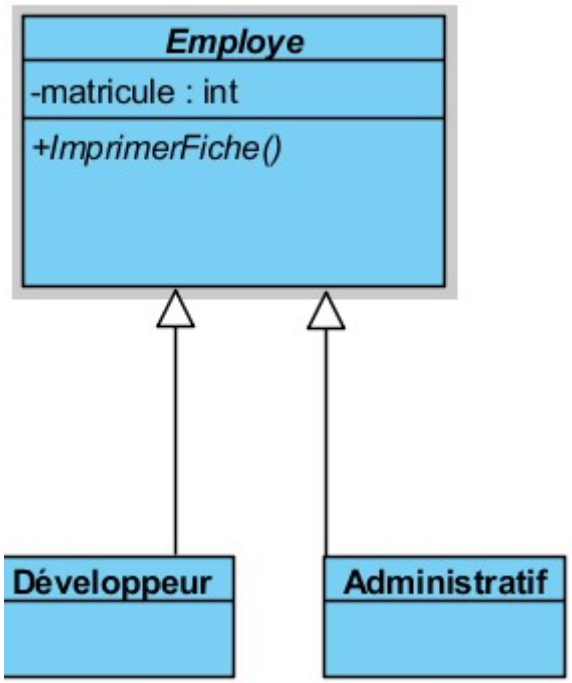
COMPOSITION

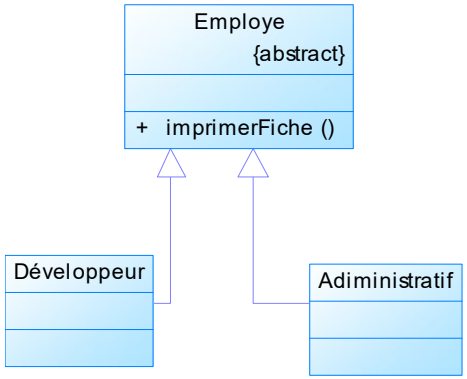
- Un type particulier d'agrégation dans lequel les parties sont fortement liées au tout
  - Un composant ne peut être partie que d'un seul agrégat.
  - L'agrégat gère le cycle de vie (création et destruction) de ces parties.



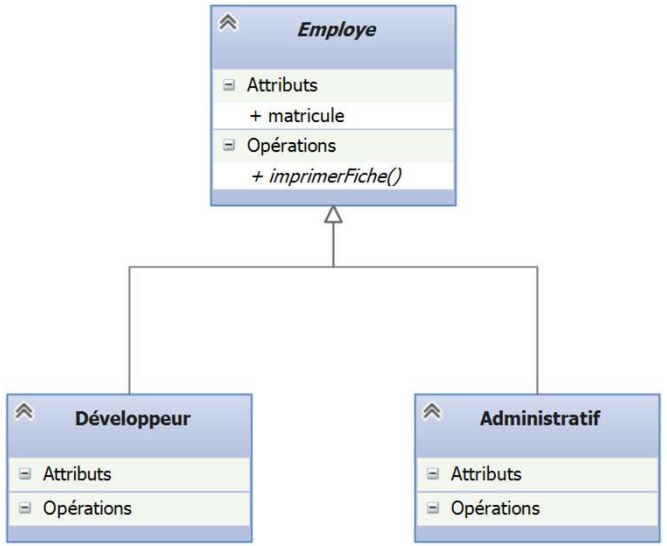
GÉNÉRALISATION / SPÉCIALISATION







- La classe C2 hérite de la classe C1



DÉPENDANCE

- Une dépendance est une relation unidirectionnelle
- Une classe B est en dépendance de la classe A si des éléments de la classe A sont nécessaires pour construire la classe B.
- Elle indique qu'une modification de la classe cible peut impliquer des modifications dans la classe source. La dépendance est souvent stéréotypée pour mieux expliciter le lien sémantique entre les éléments du modèle.
- Stéréotypes de dépendance possibles
  - « use »: la source utilise la partie publique de l'objet cible
  - « friend »: la classe source possède une visibilité spéciale dans la cible.
  - « derive »: la source est calculée à partir de la cible
  - « call » : appel d'une opération de la cible dans la source.
  - « instantiate » : une opération de la source crée une instance de la cible
- On utilise souvent une dépendance quand une classe en utilise une autre comme argument dans la signature d'une opération.

DIAGRAMMES DE PACKAGES

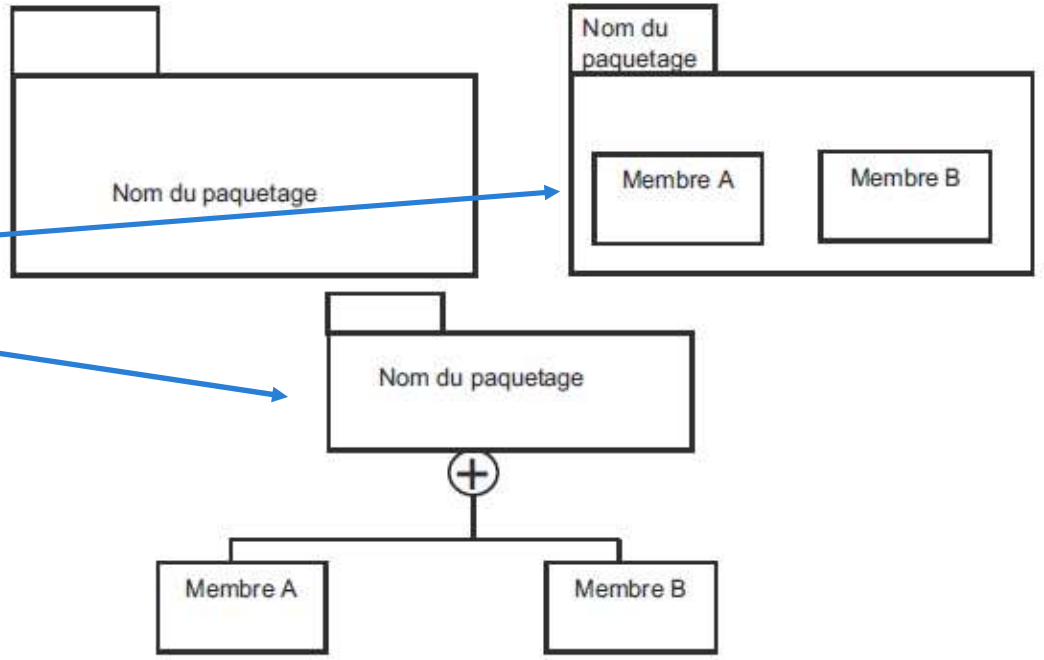
- Un package permet d'organiser les éléments de modélisation en groupes.
- Un package définit un espace de noms, les noms des éléments dans un packages doivent être uniques.
- Un élément E1 dans un package P1, peut être utilisé à partir d'autres packages en utilisant le nom complétement qualifié P1::E1
- Un diagramme de packages est un diagramme UML qui décrit la structure des packages de l'application et il offre une vue de haut niveau du système. Il existe deux types de liens entre packages:
  - Liens de généralisation
  - Liens de dépendance.

3 types de représentation:

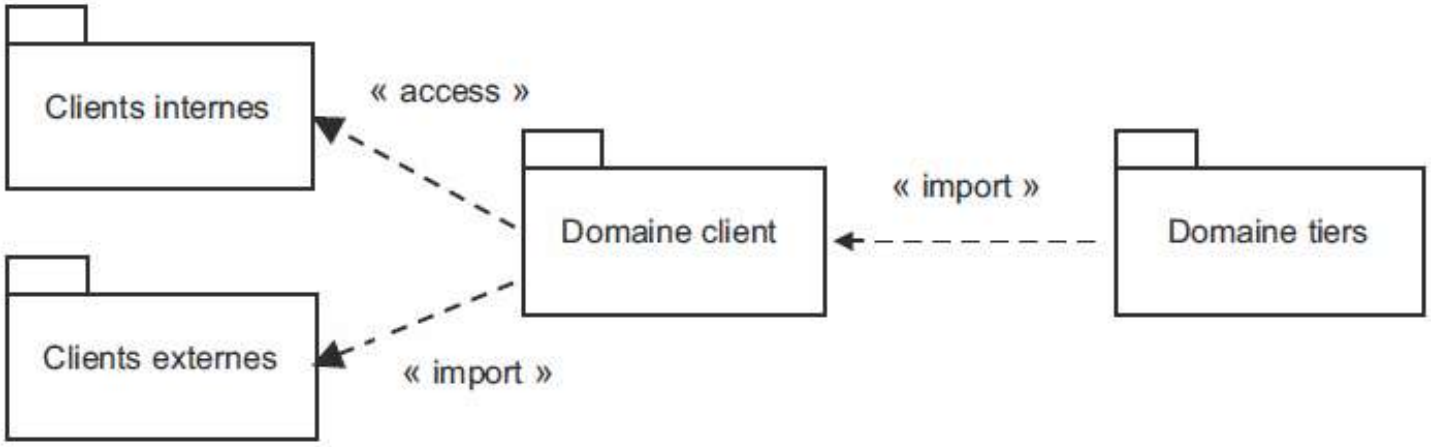
Représentation globale

Représentation détaillée

Représentation éclatée



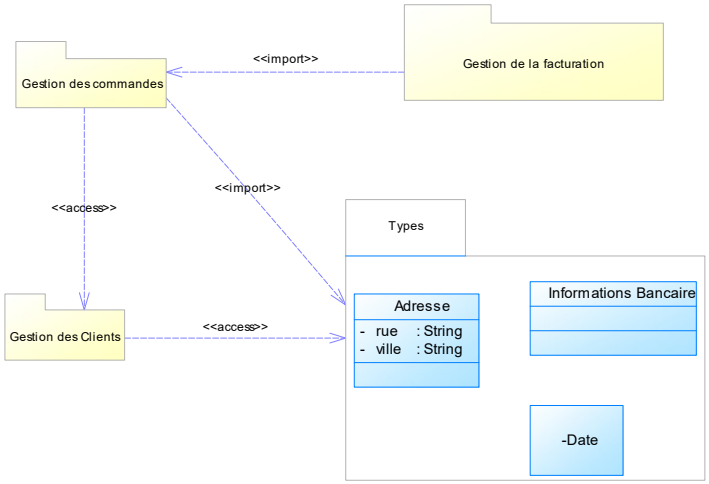
- Dépendances entre packages:
  - « import »:l'import est publique, donc « clients externes » est visible dans « Domaines tiers ».
    « import » est une relation transitive, si A importe B et B importe C, alors A importe indirectement C.
  - « access »: import privé, « clients internes » est visible dans « Domaine client » mais pas dans « Domaine tiers ».
  - « merge »: fusion de deux packages.



EXEMPLE

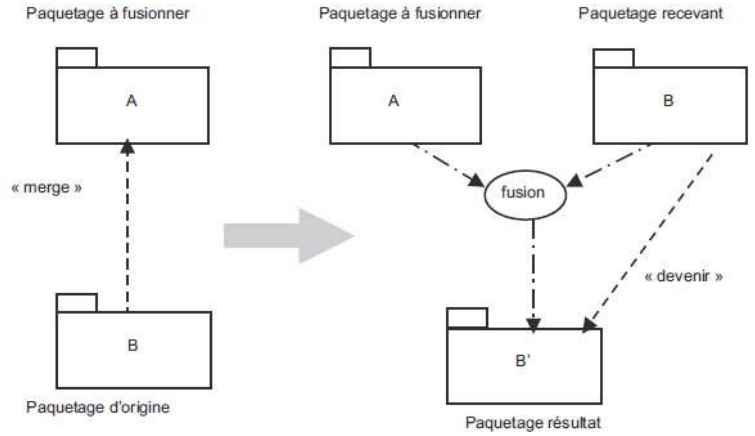
Dans l'exemple suivant, le package « Gestion de la facturation » importe le package « Gestion des commandes » qui, à

son tour, a accès au package « Gestion des clients ». Le package Types est montré en vue détaillée:

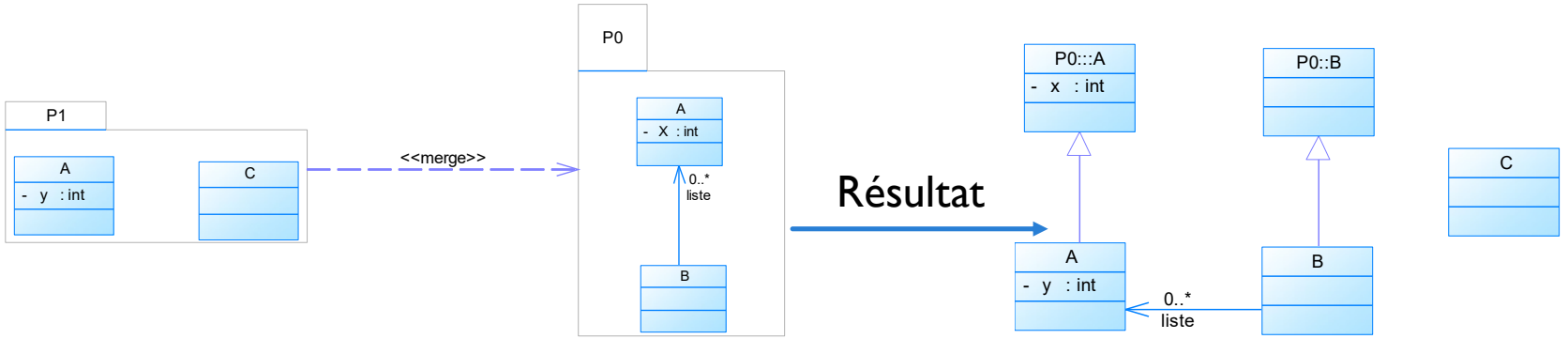


- « Date » est un élément privé dans le package « Types »
- « Adresse » et « Informations bancaires » sont visibles dans « Gestion des commandes »
- « Date » n'est pas visible dans « Gestion des commandes ».
- « import » est transitive, donc « Adresse » et « Informations bancaires » sont aussi visibles dans « Gestion de la facturation ».
- Les éléments du package « Gestion des clients » sont visibles dans « Gestion des commandes », mais pas dans le package « Gestion de la facturation ».

« MERGE »



- Les éléments du package destination sont fusionnés dans le package source.
- Une relation de généralisation est établie entre les éléments du package source et le package destination.
- Les éléments privés ne sont pas inclus dans l'opération de fusion.

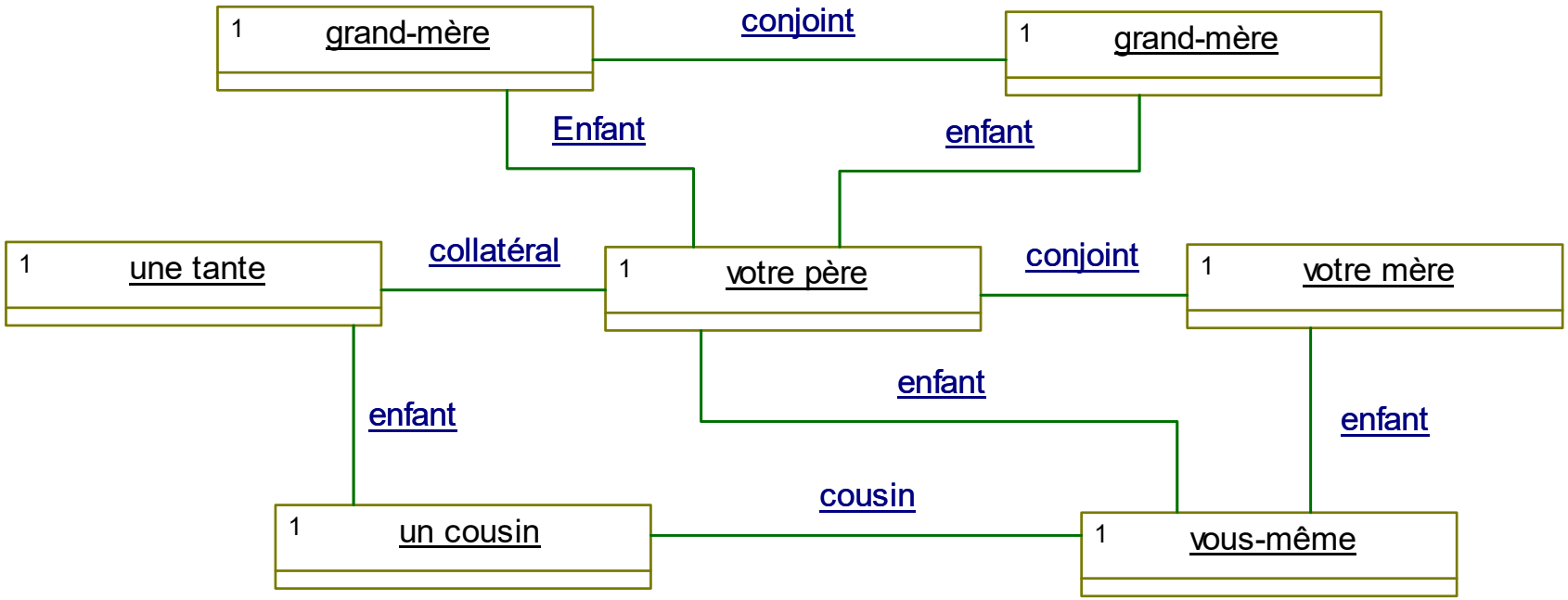


DIAGRAMMES D'OBJETS

- Le diagramme d'objets sert à modéliser une situation avant ou après une interaction ou à faciliter le choix des multiplicités, mais également à faciliter la compréhension des structures de données complexes.
- Les **objets** (instances de classes) sont reliés par des **liens** (instances d'associations) sans multiplicité.
- Le **nom d'un objet** est toujours souligné :
  - nom_objet :le nom seul correspond à une modélisation incomplète dans laquelle la classe de l'objet n'a pas encore été précisée.
  - nom_objet : nom_classe
  - :nom_classe (désigne un objet quelconque de la classe) : La classe seule évite l'introduction de noms inutiles dans les diagrammes
- Un objet peut être persistant ou transitoire (Persistent/Transient)

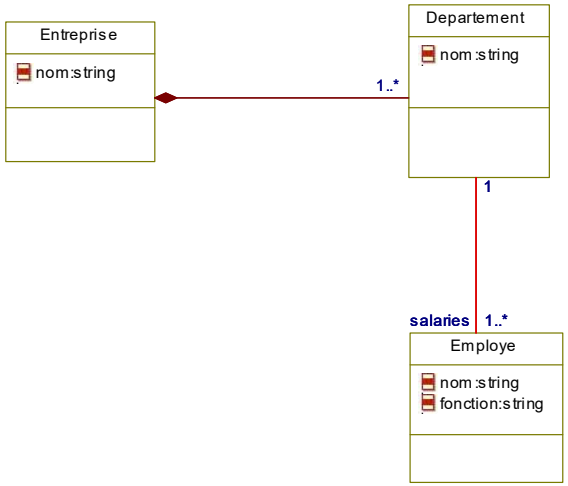
EXEMPLE 1

- Tous les objets sont de type Personne



EXEMPLE 2

- Diagramme de classes



- Diagramme d'objets