



Chapitre 3 : Analyse des exigences

UP IL

Partie 2 :

Analyse statique

Objectif

Diagrammes structurels

Diagramme de classes

Diagramme d'objets

Diagramme de packages



SECTION 1 : DIAGRAMME DE CLASSES D'ANALYSE

Présentation du Diagramme de Classes

- ❖ Le **diagramme de cas d'utilisation** montre le système du point de **vue acteurs et les cas réalisés** par chacun alors que **le diagramme de classes** montre **la structure interne pour réaliser ces cas d'utilisation**.
- ❖ Le diagramme de classes d'analyse est considéré **comme le plus important de la modélisation** orientée objet.
- ❖ Il s'agit **d'une vue statique**, car on ne tient pas compte du facteur temporel dans le comportement du système.
- ❖ Le diagramme de classes modélise **les concepts du domaine** d'application ainsi que **les concepts internes créés** de toutes pièces dans le cadre de l'implémentation d'une application.

Présentation du Diagramme de Classes

- ❖ Chaque langage de Programmation orienté objet donne un moyen spécifique d'implémenter le paradigme objet (pointeurs ou pas, héritage multiple ou pas, etc.), **mais le diagramme de classes permet de modéliser les classes du système et leurs relations indépendamment d'un langage de programmation particulier.**
- ❖ Il peut être utilisé en phases d'analyse et de conception :
 1. **Analyse** : description préliminaire de la structure des entités et du nom de quelques attributs et méthodes.
 2. **Conception** : description détaillée de l'ensemble des classes. Les attributs et méthodes doivent apparaître de façon détaillée avec tous les types de paramètres et les types de retour.
- ❖ Les principaux éléments de cette vue statique sont **les classes et leurs relations.**

Classe : Définition

❖ Une **classe** est une **description abstraite d'un ensemble d'objets** ayant :

- des **propriétés similaires**,
- un **comportement commun**,
- des **relations communes** avec d'autres objets,
- des **sémantiques communes**.

❖ Tout système orienté objet est organisé autour des classes.

❖ Une classe définit un jeu d'objets dotés de caractéristiques

communes (attributs) qui permettent de spécifier son état et son comportement (des opérations (méthodes)).

| |
|---------------------|
| Nom de la classe |
| Liste des attributs |
| Liste des méthodes |

Classe : Attributs et méthodes

- ❖ Un attribut de classe définit une propriété commune aux objets d'une classe :

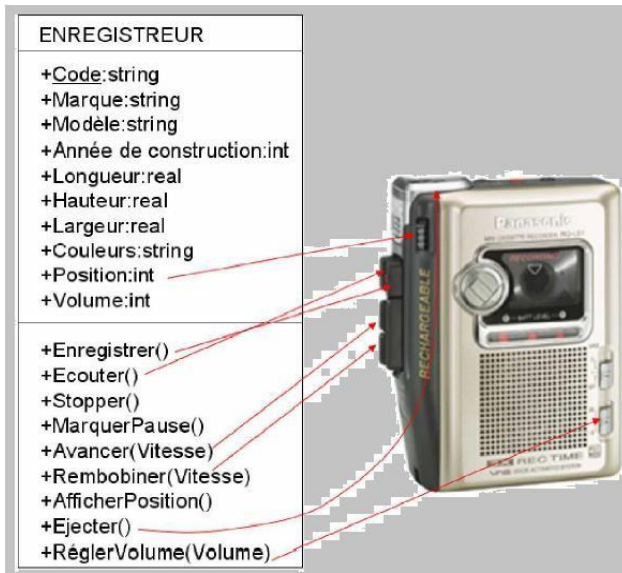
| Nom de classe |
|---|
| Visibilité(+/-/#/~) Nom d'attribut : type = valeur initiale |
| |

Dans les classes d'analyse: la visibilité, les types des attributs et l'initialisation ne sont pas obligatoires. De même pour les méthodes et leurs arguments.

- ❖ Une opération définit une fonction appliquée à des objets d'une classe :

| Nom de classe |
|---|
| |
| Visibilité/(+/-/#/~) Nom d'opération (liste d'arguments) : type de retour |

Classe : Représentation Graphique



Nom

Attributs

Méthodes

Classe : Déclarations

- ❖ Le nom de l'attribut doit être unique dans la classe. La syntaxe de la déclaration d'un attribut est la suivante :

visibilité nom : type = valeur_initiale

public +
privé –
protégé #
paquetage ~

Exemples :

+nom : string

- salaireBase : float = 2000,000

- ❖ La déclaration d'une opération contient les types des paramètres et le type de la valeur de retour. Sa syntaxe est la suivante :

visibilité nom (liste paramètres) : type-retour {propriétés}

Exemple : -calculerSalaire (ancienneté) : float

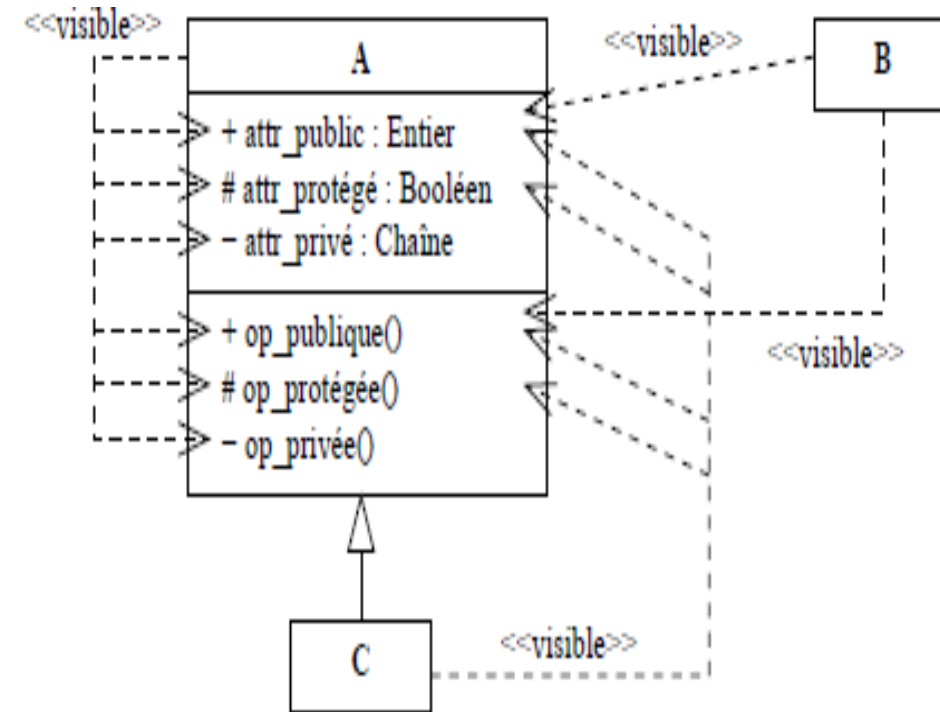
Exp : {abstract}

Classe : VISIBILITE

❖ Dans une classe, le marqueur de visibilité se situe au niveau de chacune de ses caractéristiques (attributs, terminaisons d'association et opération). Il permet d'indiquer si une autre classe peut y accéder.

❖ La visibilité concerne les attributs et les méthodes:

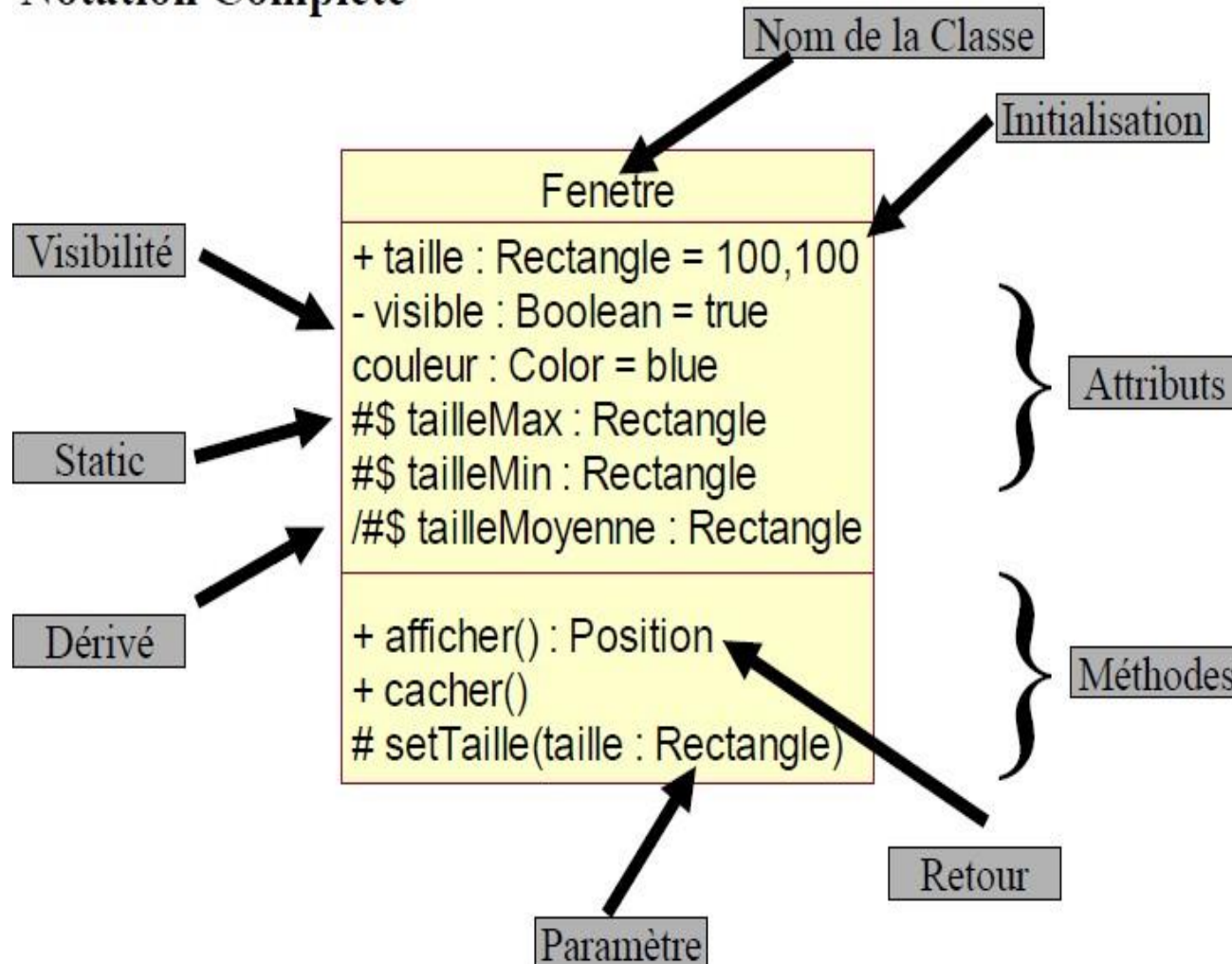
- Public (+) : visible pour toutes les classes.
- Privé (-) : n'est visible que pour la classe elle-même.
- Protégé (#) : visible pour la classe et toutes ses sous-classes.
- Package (~ ou rien) : visible seulement par les classes déclarées dans le même paquetage.



NB : Dans un paquetage, le marqueur de visibilité se situe sur des éléments contenus directement dans le paquetage, comme les classes, les paquetages imbriqués, etc. Il indique si un autre paquetage susceptible d'accéder au premier paquetage peut voir les éléments.

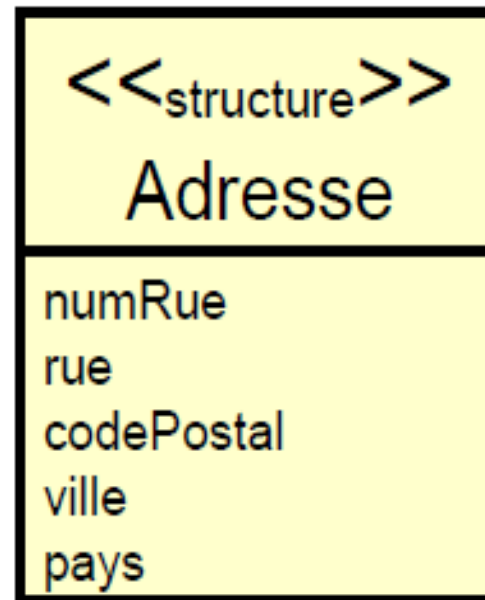
Classe : Notation complète

Notation Complète



Classe : Type « Structure »

- **Définition** : Ensemble d'attributs pouvant être regroupés.



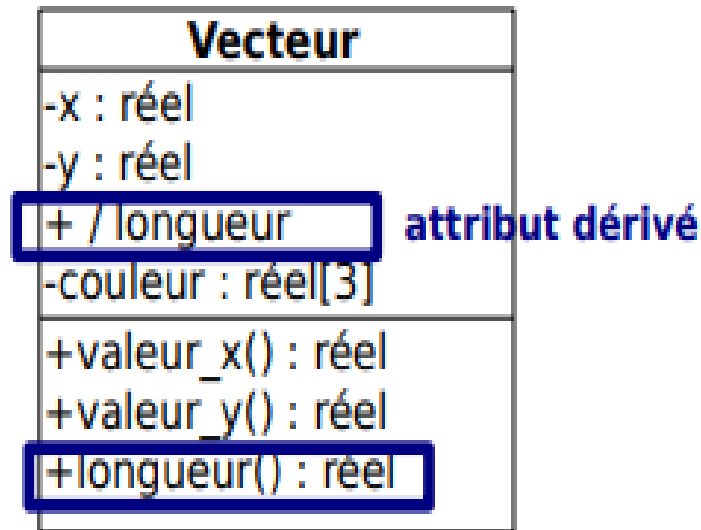
Classe : Type « Enumération »

- **Définition** : Ensemble de littéraux d'énumération, ordonné.



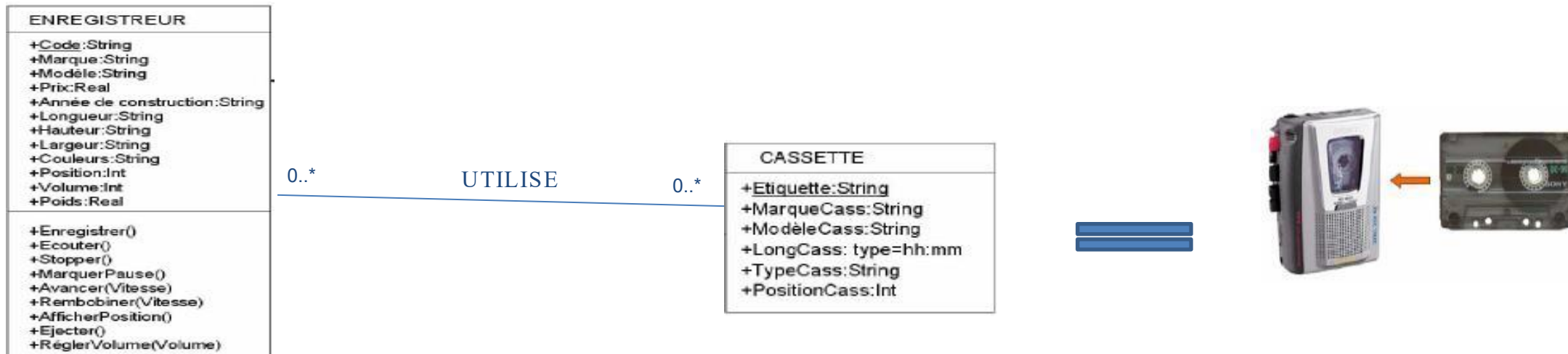
Classe : Attribut dérivé

- ❖ Les attributs dérivés peuvent être calculés à partir d'autres attributs et de formules de calcul..
- ❖ Les attributs dérivés sont symbolisés par l'ajout d'un « / » devant leur nom.



Association : Définition

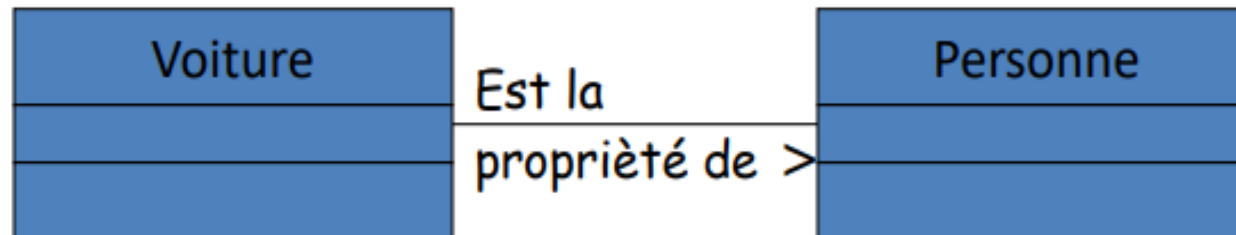
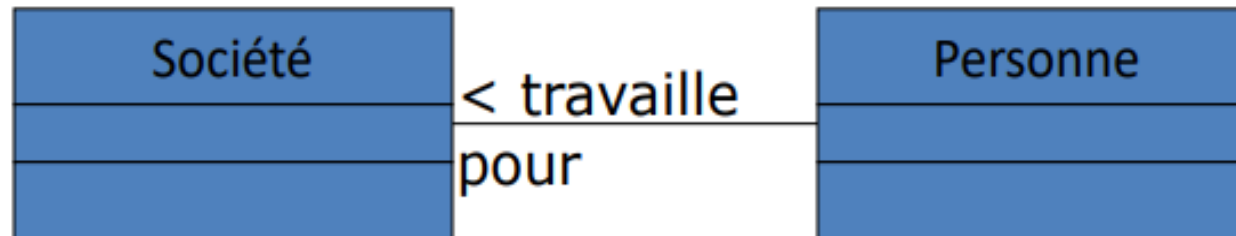
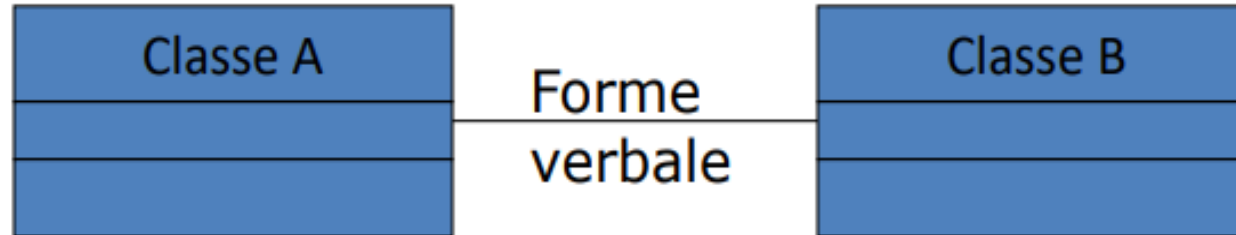
- ❖ Une association est une relation entre deux classes (association binaire) ou plus (association n-aire), qui décrit les connexions structurelles entre leurs instances.
- ❖ Une association indique donc qu'il peut y avoir des liens entre des instances des classes associées.



- ❖ Une association peut être nommée (facultatif).

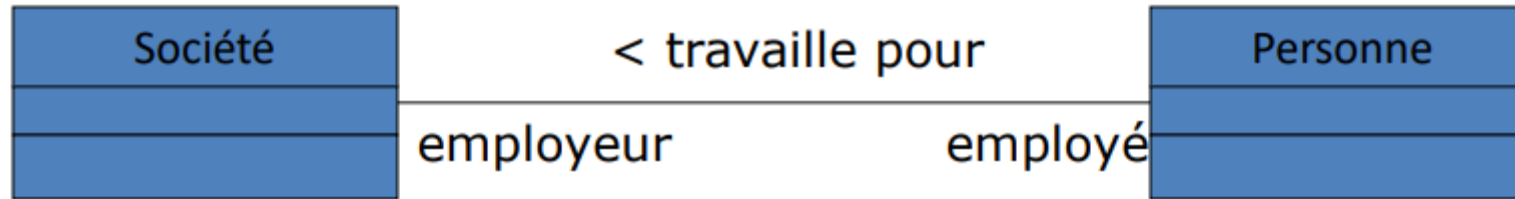
Association : Nommage

Généralement, on note les associations par une forme verbale, soit active, soit passive.

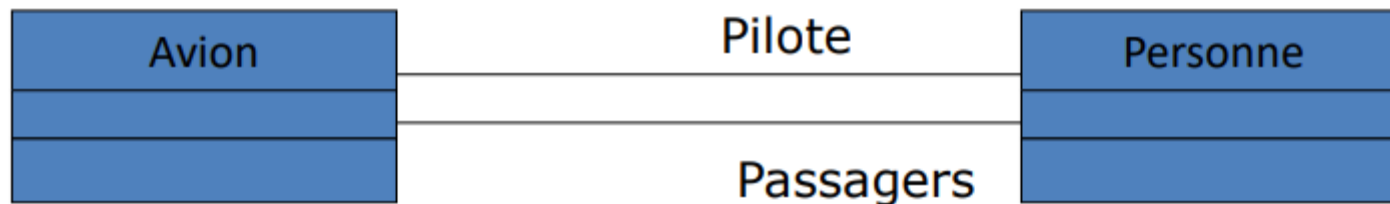


Association : Rôle

- ❖ **Rôle** : spécifie la fonction d'une classe pour une association donnée (indispensable pour les associations réflexives)
- ❖ Le rôle définit la manière dont une classe intervient dans une relation



- ❖ Intérêt des rôles dans le cas où plusieurs associations lient deux classes : distinction des concepts attachés aux associations



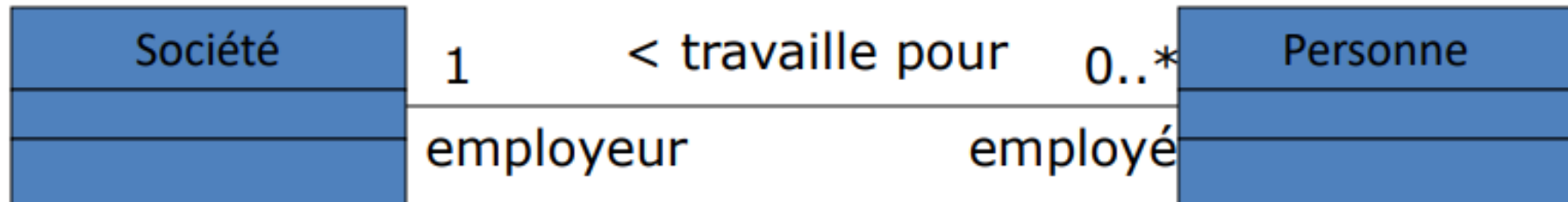
Association : Multiplicité

- ❖ Aux deux extrémités d'une association, on doit faire figurer une indication de multiplicité: (q1..q2)
- ❖ Elle spécifie -sous la forme d'un intervalle d'entiers positifs ou nuls - le nombre d'objets qui peuvent participer à une relation avec un objet de l'autre classe, dans le cadre d'une association.

| | |
|---------------|--------------------------------------|
| 1 | un et un seul |
| 0 .. 1 | zéro ou un |
| M .. N | de M à N (entiers naturels) |
| * | de zéro à plusieurs |
| 0 .. * | de zéro à plusieurs |
| 1 .. * | de un à plusieurs |
| N | exactement N (entier naturel) |

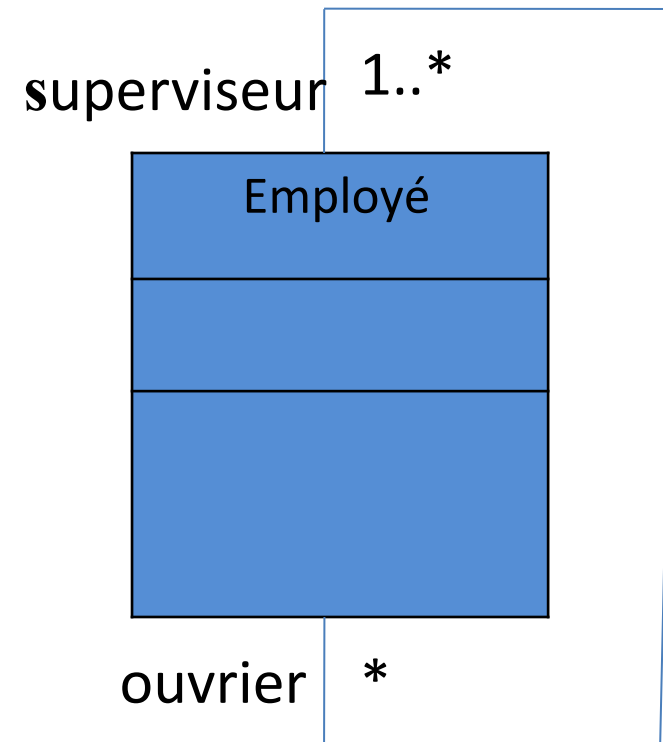
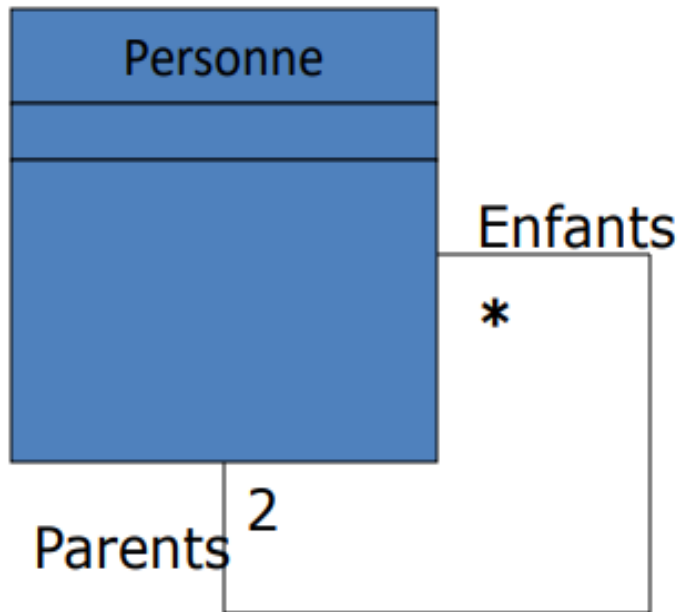
Association : Multiplicité

- 1 : Une personne travaille pour une et une seule société.
- 0 .. * : Une société emploie de zéro à plusieurs personnes.



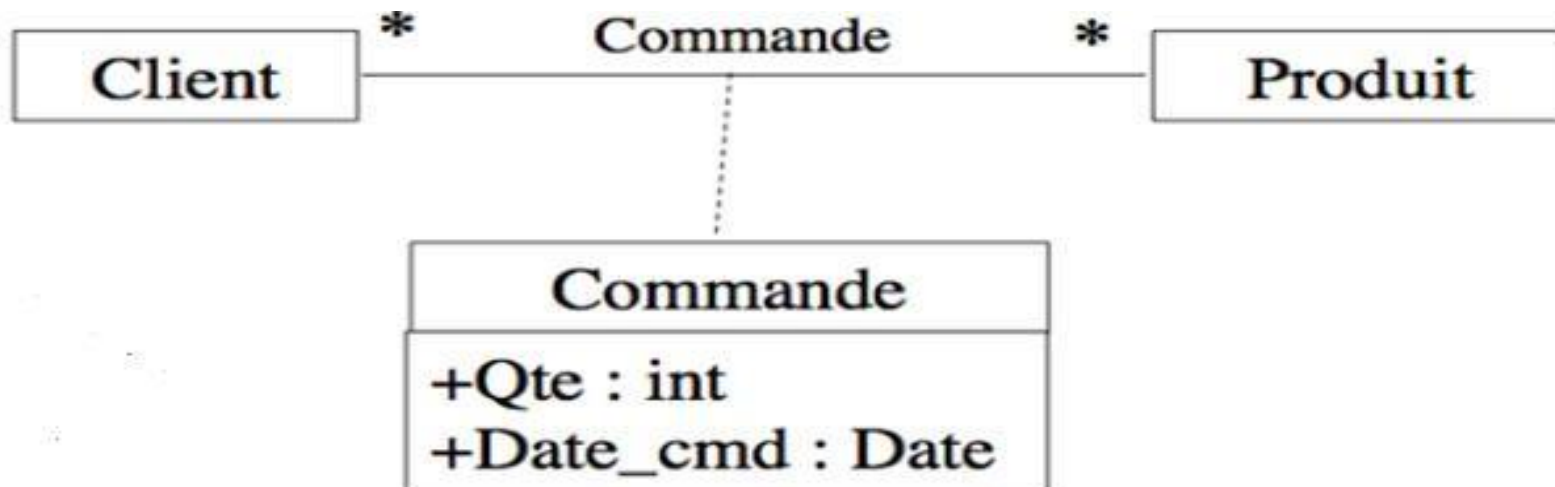
Association : Association réflexive

- Une association est dite réflexive si elle s'applique à des objets d'une même classe.

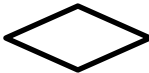


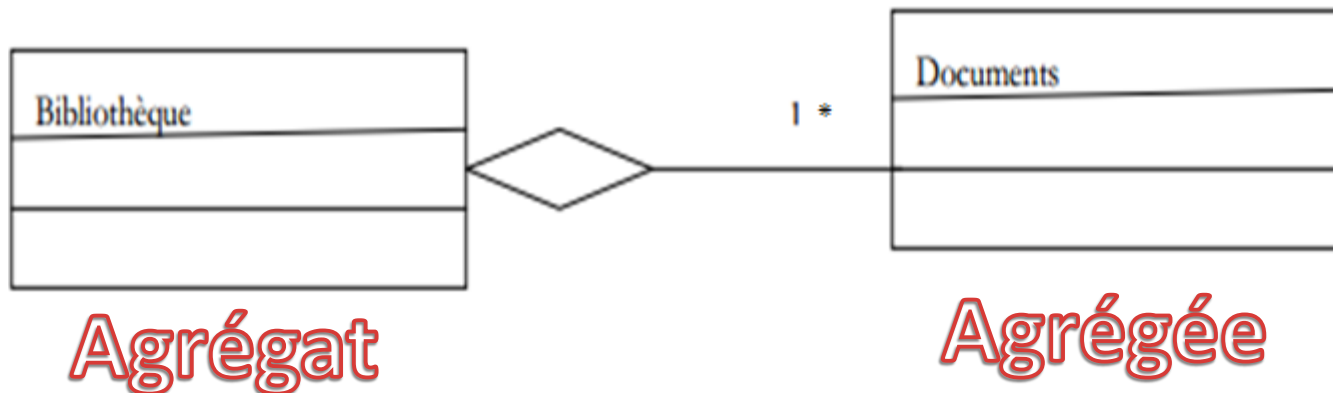
Association : Classe d'association

- ❖ Une association qui contient des attributs et qui ne participe pas à des relations avec d'autres classes est appelée **classe d'association**.
- ❖ Il s'agit d'une classe qui réalise la navigation entre les instances d'autres classes. Elle sert à connecter les classes entre elles.




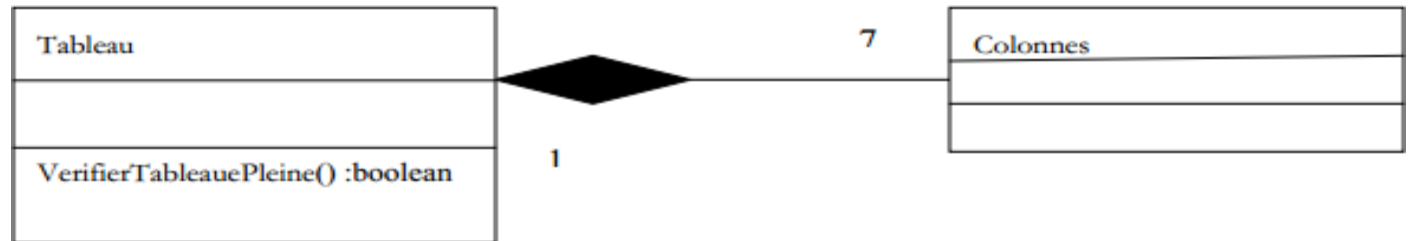
Association : Agrégation

- ❖ Lorsque l'on souhaite modéliser une relation *tout/partie* où une classe constitue un élément plus grand (*tout*) composé d'éléments plus petits (*partie*), il faut utiliser une agrégation. 
- ❖ Une agrégation est une association qui représente une relation d'inclusion structurelle (contenance) ou comportementale d'un élément dans un ensemble. Graphiquement, on ajoute un losange vide du côté de l'agregat.
- ❖ Elle définit la relation «partie de »



Association : Composition

- Une composition est une agrégation forte, elle est représentée par le symbole 
- Egalement, appelée agrégation composite, décrit une contenance structurelle entre instances. Ainsi, la destruction de l'objet composite implique la destruction de ses composants.



Différences entre une agrégation et une composition :

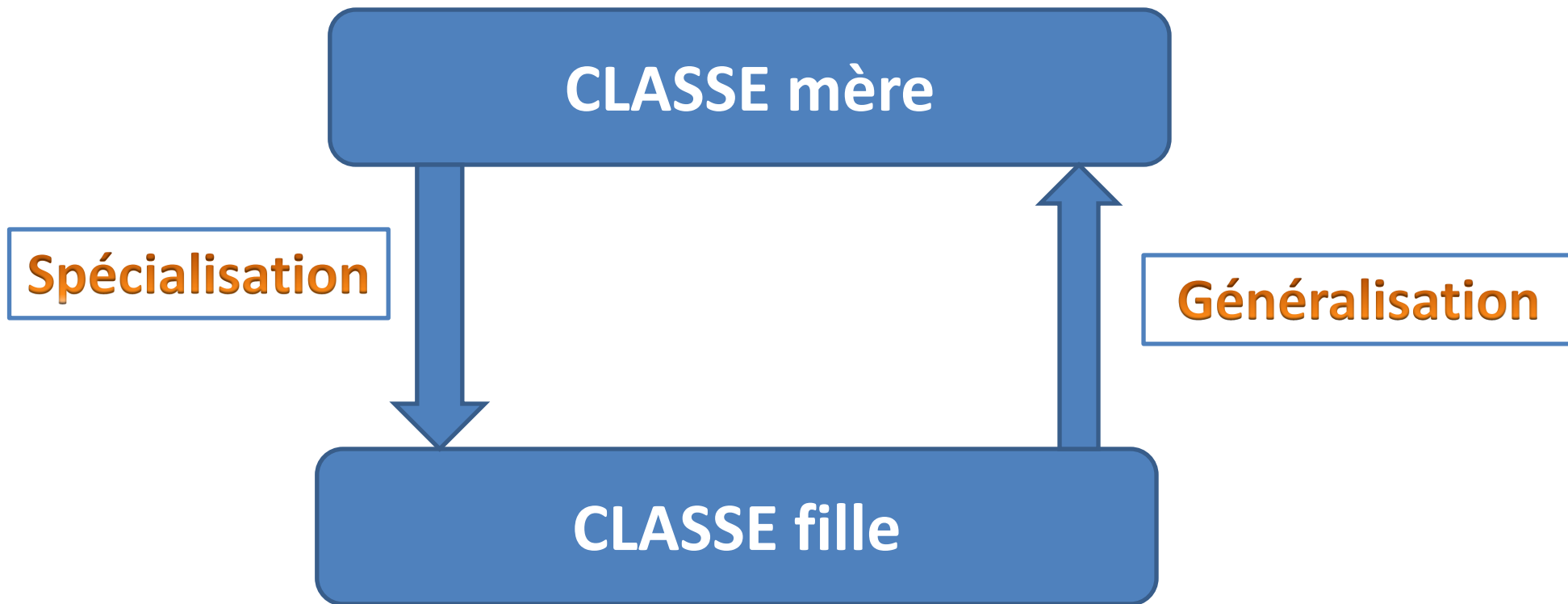
Pour la relation de composition :

- La destruction de l'objet composite implique la destruction de ses composants.
- La multiplicité du côté composite ne doit pas être supérieure à 1 (i.e. 1 ou 0..1).
- La composition est une agrégation non partagée.

Association : Héritage

- Jusqu'à maintenant chaque classe d'objets est introduite séparément des autres classes.
- Dans le monde réel, il existe des objets qui appartiennent à plusieurs classes.
- **Exemple:**
 - Mon chat appartient à la classe des chats , à la classe des mammifères , à la classe des animaux , à la classe des êtres vivants .
 - La classe des êtres vivants est plus générale que celles des animaux , elle-même plus générale que celle des mammifères , etc.

Association : Héritage



Association : Héritage

L'héritage : mécanisme basé sur la généralisation et qui permet aux sous-classes d'hériter, c'est à dire d'avoir les mêmes attributs, opérations et associations que la super-classe.

