Institut Supérieur des Etudes Technologiques (ISET-Tunisie)
Niveau: 2ème année Licence Appliquée
Spécialité : DSI, RSI

Diagramme de Classe

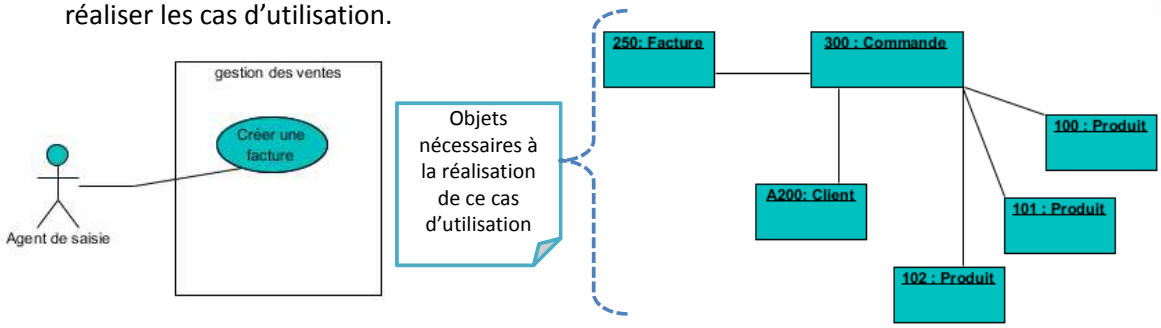
Conception Orientée Objet (UML) – Chapitre 3

Amir Souissi

Plan

1. Présentation des Classes et des Objets

- La **vue logique** a pour but d'identifier les éléments du domaine, les relations et interactions entre ces éléments. Elle les identifie selon deux aspects: dynamique et statique.
- Axe de modélisation **statique** : pas de facteur temporel
- Le système est composé **d'objets** qui interagissent entre eux et avec les acteurs pour réaliser les cas d'utilisation.



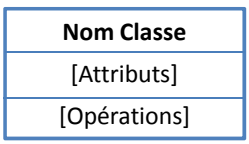
- Le diagramme de Classe permet de fournir une **représentation abstraite** de ces **objets.**
- Le diagramme de Classe montre **la structure interne** du système.
- En phase **d'analyse**, il a pour objectif de décrire la structure des entités manipulées par les utilisateurs
- En phase de **conception**, il représente la structure d'un code orienté objet, ou a un niveau de détail plus important, les modules du langage de développement.

Classe et Objet

- Une **classe** représente la description abstraite d'un ensemble d'objets possédant les mêmes caractéristiques. On parle également de **type**.
- Un **objet** est une entité possédant une **identité** et encapsulant un **état** et un **comportement**. Un objet est une instance d'une classe.
- La classe est le **modèle**, l'objet est sa **réalisation**.

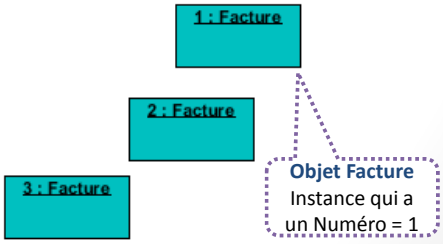
Représentation UML :

Représentation d'une classe



Représentation d'un objet

Exemple:



2. Les Eléments Constituants le Diagramme de Classe

- La **Classe** est un concept **Abstrait** qui permet de représenter toutes les entités d'un système.
- Elle est définie par son **nom**, ses **attributs** et ses **opérations** comme suit :

Commande
- Numero : int - DateCom : Date {<=DateAujourd'hui} - Montant : float = 0
+ Créer() + Calcul_Montant():float

- Un **attribut** est une propriété de classe intéressante pour l'analyste. Les attributs correspondent à des variables associées aux objets de la classe.
- Une **opération** est un service rendu par la classe. Les opérations correspondant à des fonctions associées aux objets de la classe.

Mr Amir Souissi

Classe : Attributs et Opérations

- Syntaxe de déclaration des attributs d'une Classe:

  Visibilité [/] **nomAttribut** : **Type** [= *ValeurParDefaut*]

  - **[/]** : le slash précède un attribut **dérivé**. Il s'agit d'une information valué intéressante mais redondante car sa valeur peut être déduite à partir d'autres attributs.

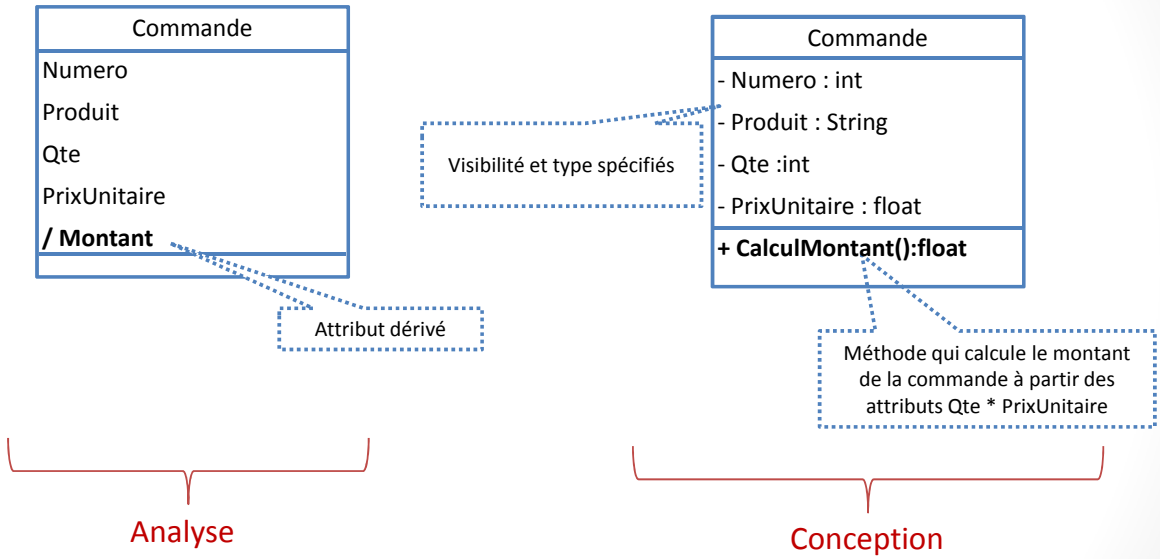
- Syntaxe de déclaration des opérations d'une Classe:

  Visibilité nomOperation ([param1, ... , paramN]**)** : [typeRetour] [{propriétés}]

  - **Visibilité** : les modificateurs d'accès permettent de définir la visibilité :
    - ✓ **-** : privée : visible seulement par la classe elle-même
    - ✓ **+** : public : visible par toutes les autres classes
    - ✓ **#** : protégé: visible seulement par la classe elle-même et ses classes filles
    - ✓ **~** : package: seul un élément déclaré dans le même paquetage peut voir l'élément

Mr Amir Souissi

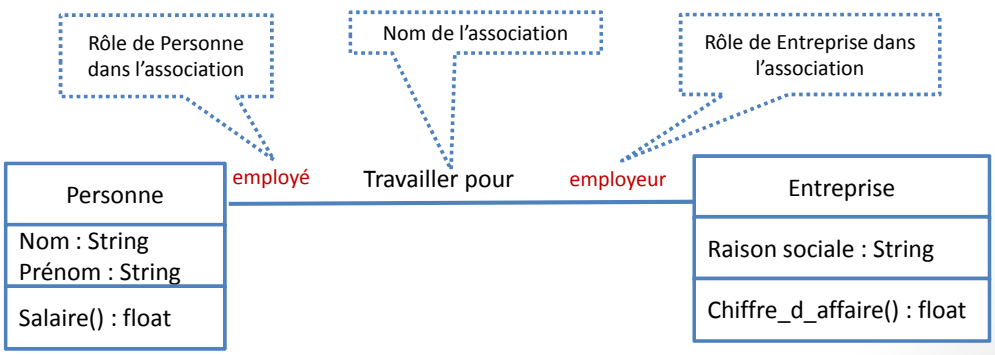
Classe : Analyse vs Conception



3. Associations

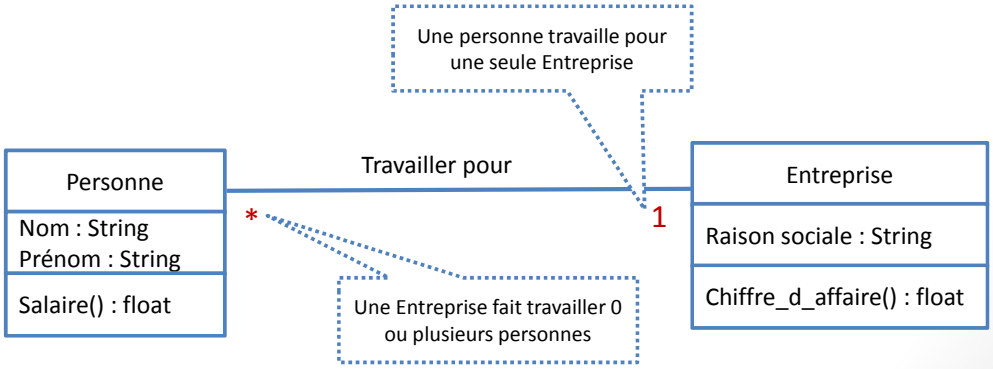
Association et Rôle

- Une **association** est une relation statique entre plusieurs classes.
- Elle représente une **relation possible entre les objets** des classes.
- On peut donner à une classe un **rôle** dans une association.
- Un rôle précise la **signification de l'entité** à proximité du rôle dans l'association.



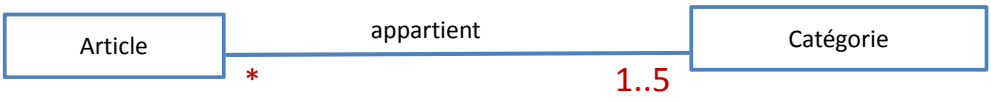
Multiplicité

- La **multiplicité** spécifie l'ensemble des cardinalités possibles (**parmi les entiers naturels**) sur un rôle.
- Les multiplicités permettent de **contraindre le nombre d'objets** intervenant dans les instanciations des associations. On en place de chaque côté des associations.
- Une multiplicité d'un côté spécifie combien d'objets de la classe du côté considéré sont associés à un objet donné de la classe de l'autre côté.



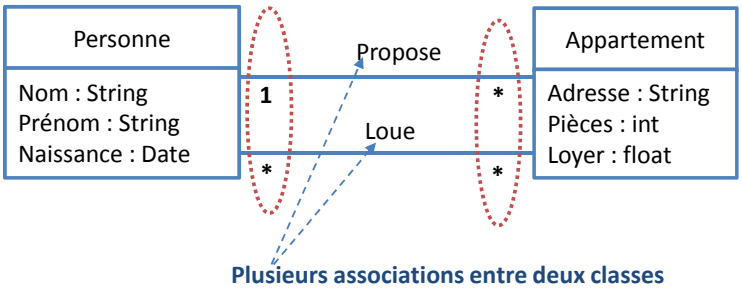
Multiplicité

- On peut préciser :
  - Un intervalle (min . . max): 2 . . 5 ➔ de 2 à 5
    0 . . 1 ➔ 0 ou 1
  - Une liste de valeurs (val1, val2, …) : 2,5,7 ➔ 2 ou 5 ou 7
  - Ne pas borner supérieurement (*) : 1. . * ➔ au moins 1
    0 ..* ou * ➔ 0 ou plusieurs

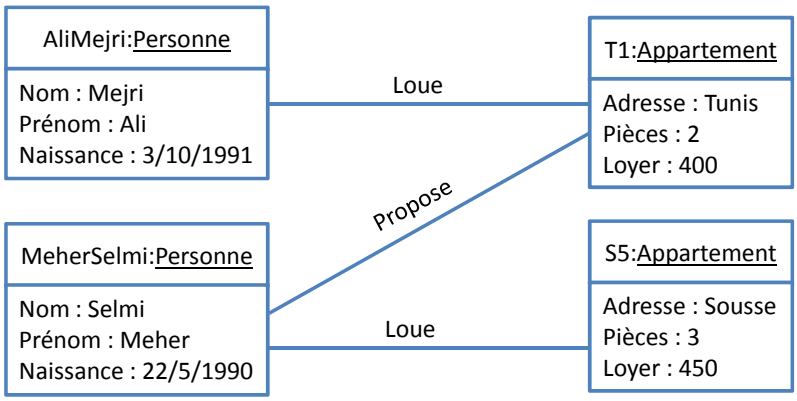


- **1 .. 5** s'interprète : « à un objet donné de la classe Article, on doit associer au minimum 1 objet de la classe Catégorie et on peut en associer au maximum 5 »,
- * s'interprète : « à un objet donné de la classe Catégorie, on peut associer 0 ou plusieurs objets de la classe Article »

Association Multiple

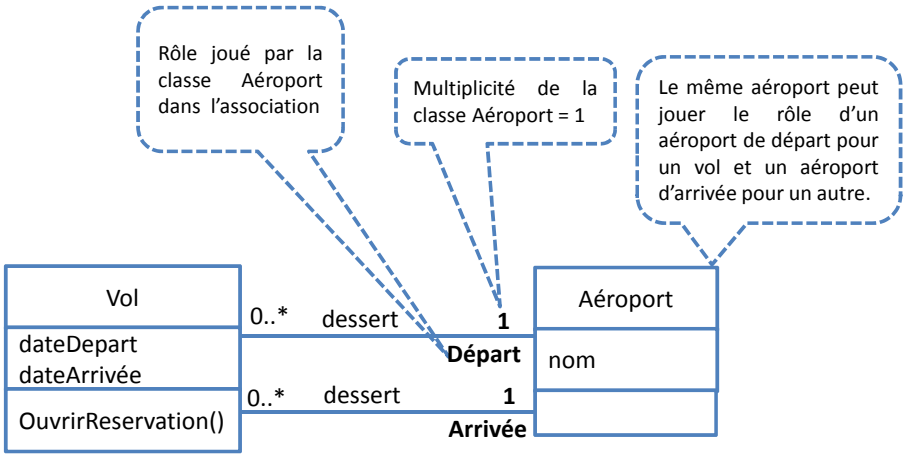


Exemple de diagramme d'objets :



Association Multiple

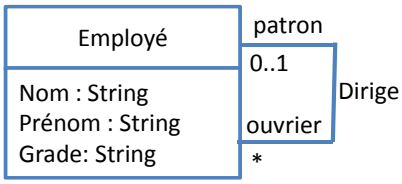
- Lorsqu'on a une classe dont les instances peuvent être redondants (c-à-d la même instance qui participe plusieurs fois à une association), alors créer différentes associations entre les classes, chacune affectée d'un rôle différent avec une multiplicité égale à 1 exactement.

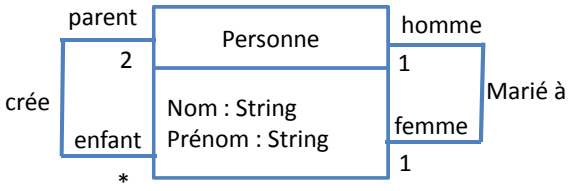


- Le rôle ici est très utile puisque la même association concerne les mêmes classes.

Association Réflexive

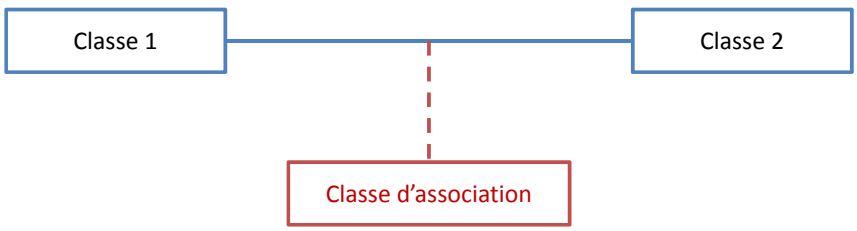
- C'est une association entre une classe donnée et elle-même.





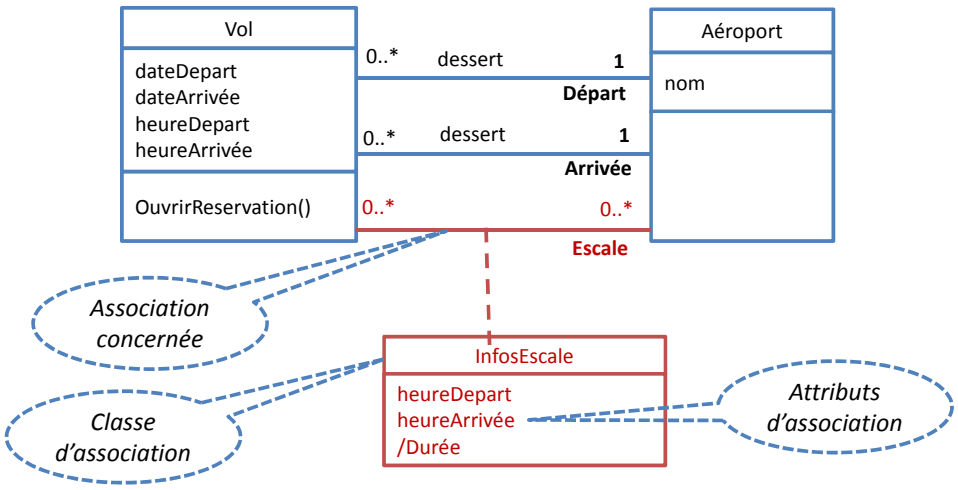
Classe d'Association

- Une association peut parfois avoir des **propriétés**. Dans ce cas, ces propriétés sont encapsulées dans une classe nommée **classe d'association**.
  - Une classe d'association est représentée par un **trait discontinu** qui relie la classe avec l'association qu'elle caractérise.



  - Généralement, une association de cardinalité **plusieurs à plusieurs** est une classe d'association.

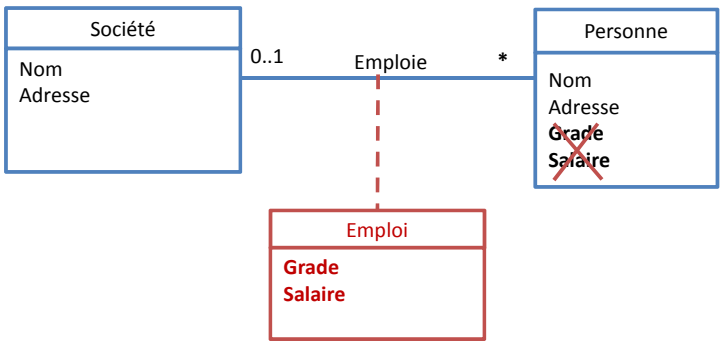
Classe d'Association



Chaque Vol peut avoir plusieurs Aéroport servant d'escales et chaque Aéroport peut servir d'escale pour plusieurs vols. Les informations sur les escales sont des attributs d'associations qui caractérisent la classe d'association.

Classe d'Association

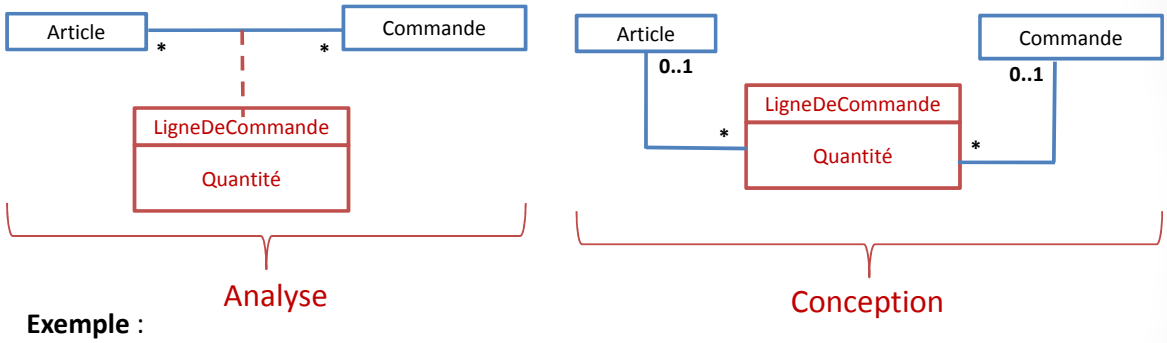
- On veut caractériser chaque personne travaillant dans une société par ses grades et ses salaires.



- La société emploie plusieurs personnes avec des grades et des salaires différents.
- Une personne peut avoir plusieurs grades et salaires dans la société.
- ➔ les attributs grade et salaire n'appartiennent ni à la classe Société ni à la classe Personne.

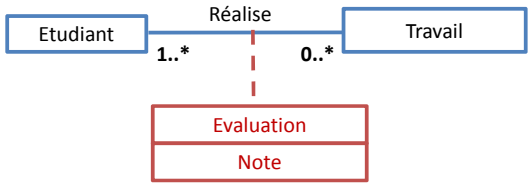
Classe d'Association

- Lors de la conception, une classe-association peut être remplacée par une classe intermédiaire.



Exemple :

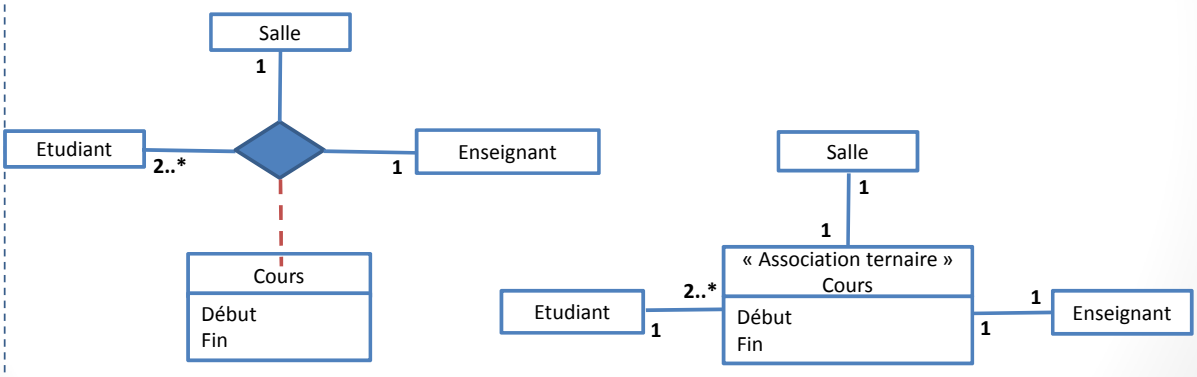
- Un étudiant peut réaliser plusieurs travaux
- Un travail est réalisé par un ou plusieurs étudiants
- Pour chaque travail réalisé pour chaque étudiant, on lui attribue une note



Association n-aire

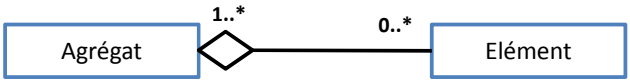
- Les associations ont souvent une **arité binaire**: deux classes en relation.
- Association **n-aire** : relie **plus de deux classes**.
- Représentée par un losange.

➔Il vaut mieux limiter leurs utilisations.



Relation d'Agrégation

- C'est un cas particulier d'une association qui exprime la **contenance**
- • N'a pas besoin d'être nommée (elle signifie « **contient** », « **est composé de** »)
- • Un élément peut appartenir à plusieurs agrégats (agrégation partagée)
- • La destruction de l'agrégat **n'entraine pas** **la destruction de tous ses éléments**.

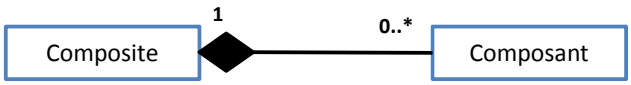


Représentation de l'agrégation en UML

- Une agrégation peut exprimer :
  - Qu'une classe « **élément** » fait partie d'une autre « **agrégat** »
  - Un changement **d'état** d'une classe entraine un changement d'état d'une autre.
  - Une **action** sur une classe entraine une action sur une autre.

Relation de Composition

- C'est une relation qui exprime une **agrégation plus forte**.
  - Un élément ne peut appartenir qu'à **un seul agrégat** composite (agrégation non partagée)
  - La destruction de l'agrégat composite **entraine** la **destruction de tous ses éléments**. (le composite est responsable du cycle de vie des composants)

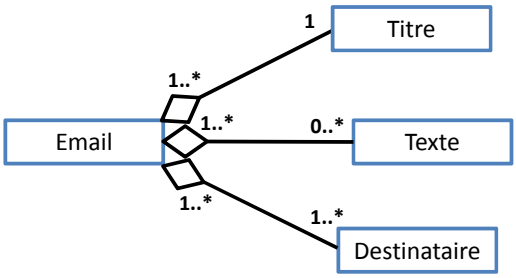


Représentation de la composition en UML

- Les objets composants sont des instances de la classe composite.

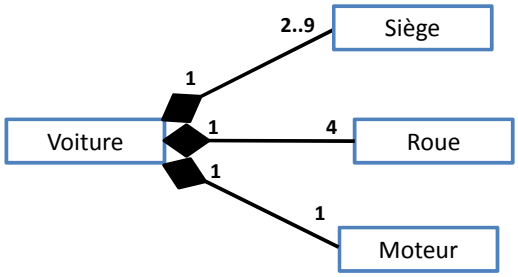
Agrégation vs Composition

Agrégation



- Un email peut contenir un titre, du texte et des destinataire
- Titre, Texte et Destinataire sont des classes qui peuvent constituer un email.
- Titre, Texte et Destinataire peuvent être partagés entre plusieurs emails
- La destruction de l'email n'entraine pas la destruction de tous ses éléments

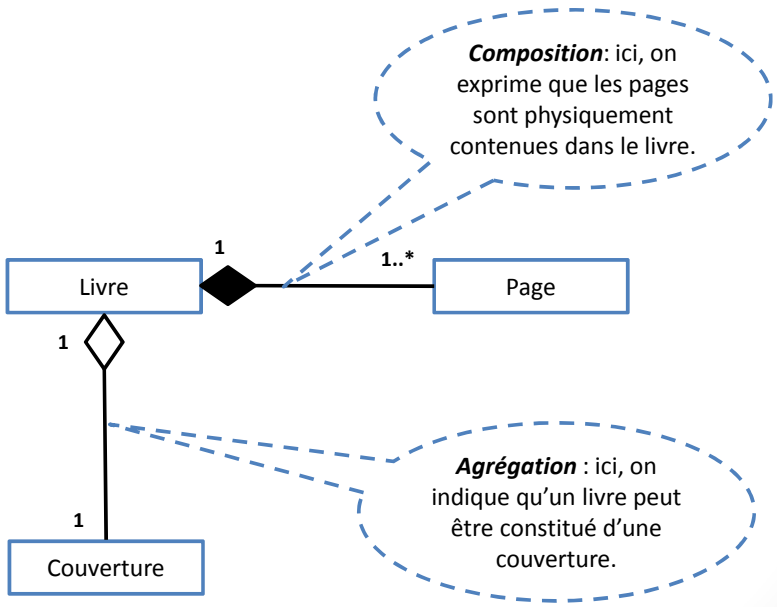
Composition



- Une voiture doit contenir des sièges,4 roues et 1 moteur.
- Siège, Roue et Moteur sont des classes qui constituent une voiture.
- Siège, Roue et Moteur ne sont pas partagés entre plusieurs voitures
- La destruction de la voiture entraine la destruction de tous ses éléments

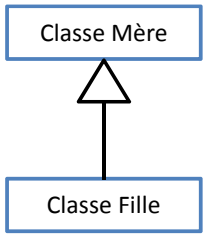
Agrégation vs Composition

Un livre est composé d'une ou plusieurs pages et d'une couverture.



Relation d'Héritage

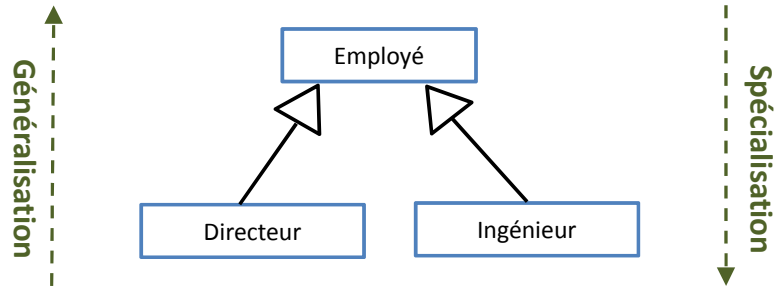
- L'association de **généralisation** définit une relation de **classification** entre une classe plus générale et une classe plus spécifique.
- La classe spécifique contient par **héritage** tous les **attributs**, les **opérations** et les **relations** de la classe générale et peut en contenir d'autres.



Représentation de l'héritage en UML

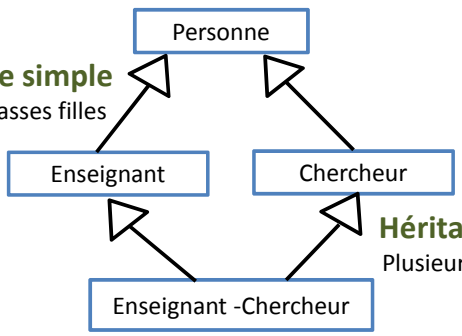
- Une hiérarchie d'héritage **ne doit pas contenir de cycle**.
- **Non réflexive** : une classe ne peut pas dériver d'elle-même.
- **Non symétrique** : si B dérive de A alors A ne peut pas dériver de B.

Héritage : généralisation / spécialisation simple / multiple



Héritage simple

Une classe mère et plusieurs classes filles

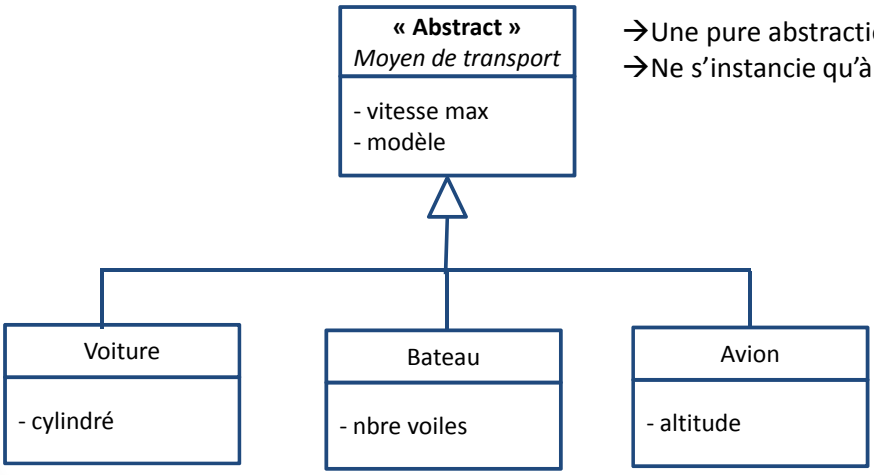


Héritage multiple

Plusieurs classes mères et une classe fille

Héritage et Classes Abstraites

- Une **classe abstraite** est une classe qui ne peut pas être directement instanciée.
- Son objectif est de **factoriser** des propriétés communes à plusieurs sous-classes.
- Lorsqu'une classe possède une seule spécialisation, alors elle ne doit pas être abstraite.



→Une pure abstraction
→Ne s'instancie qu'à travers ses filles

4. Contraintes

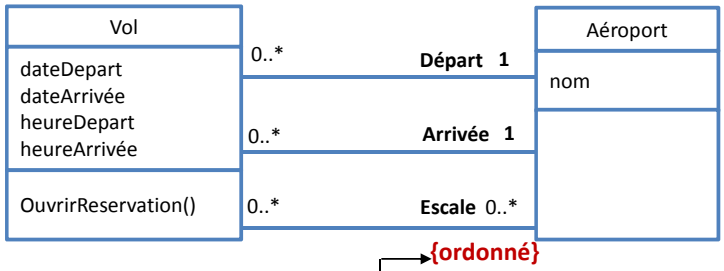
- Ce sont des relations sémantiques définies sur une relation ou sur un groupe de relations.
- Elles permettent de restreindre le nombre d'instances visées.

- Elles peuvent s'exprimer:
  - graphiquement par **{ texte }**
  - en langage naturel
  - en langage OCL (Object Constraint Language)

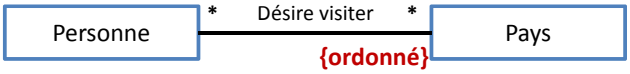
- Elles sont de deux types:
  - Prédéfinie : un standard d'UML
  - Non prédéfinie

Contrainte {ordonné}

- Elle est définie sur un rôle et spécifie qu'une relation **d'ordre** décrit les objets de la collection.

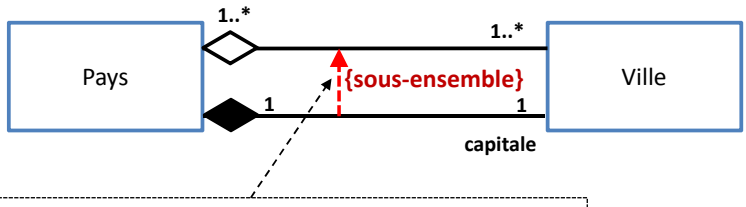


On met l'accent sur l'importance de l'ordre des aéroports qui jouent le rôle d'escale dans un vol

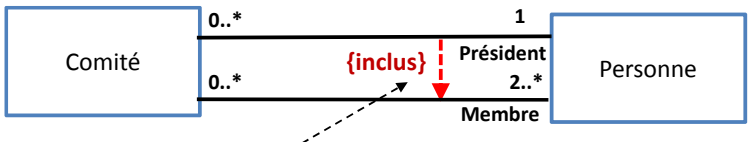


Contrainte {sous-ensemble} ou {inclusion}

- Elle indique qu'une collection est un **sous ensemble** (**est inclus**) d'une autre.



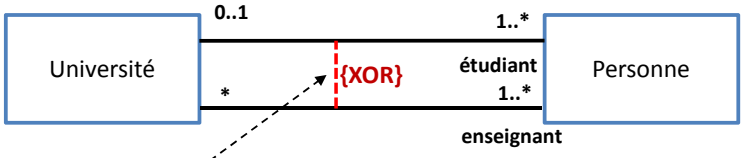
La capitale d'un pays est forcément l'une de ses villes.



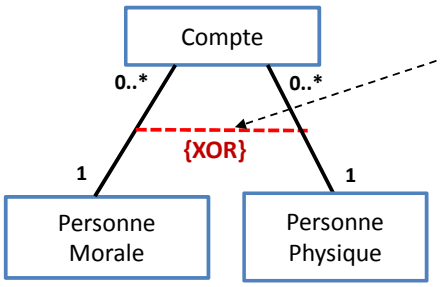
Le président est également un membre du comité.

Contrainte {XOR}

- Elle indique que tous les objets d'une classe peuvent participer **à l'une des deux associations** mais pas aux deux à la fois.



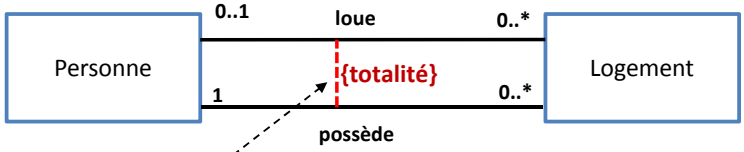
Les personnes qui jouent le rôle d'étudiant ne peuvent pas jouer le rôle d'enseignant dans la même université, et inversement.
➔une personne est soit enseignant soit étudiant au sein de la même université



Un compte bancaire appartient
soit à une personne physique
soit à une personne morale, mais pas les deux.

Contrainte {totalité}

- Elle indique que **toutes** les instances d'une classe participent **au moins** à une association.



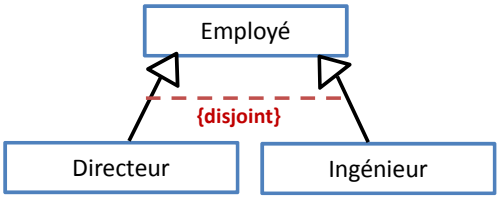
Toutes les instances de *Personne* et de *Logement* participent
Toutes au moins à l'une des associations « loue » ou « possède ».

Remarques :

- Il existe plusieurs autres contraintes prédéfinies, tels que :
  - {unique} : l'attribut a une valeur unique dont tous les objets ne doivent pas la partager avec lui, une clé primaire est obligatoirement {unique}.
  - {addOnly} : le nombre d'objet ne peut qu'augmenter.
  - Certaines ne sont plus un standard UML tel que {frozen}.
- Le langage OCL permet de décrire les contraintes de manière plus précise => un langage de contraintes.

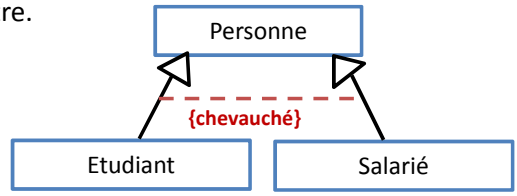
Contraintes d'héritage {disjoint}, {chevauché}, {complet}

❑ **{Disjoint}** : tout objet est au plus une instance d'une sous-classe.



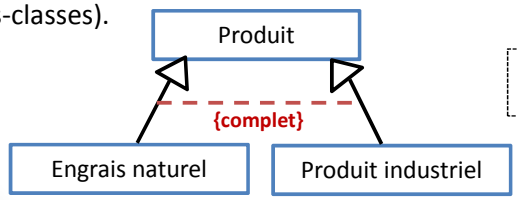
Directeur ∩ Ingénieur = ∅

❑ **{Chevauché}** : Une instance de l'une des spécialisations peut être simultanément une instance de l'autre.



Etudiant ∩ Salarié ≠ ∅

❑ **{Complet}** : indique que la généralisation est terminée (il n'est pas possible d'ajouter d'autres sous-classes).



Engrais naturel ∪ Produit industriel = Produit

Contraintes non prédéfinies

❑ UML permet de spécifier explicitement des contraintes particulières sur des éléments du modèle.

❑ Certaines contraintes sont propres au système à modéliser.