Université Sultan Moulay Slimane
Ecole Supérieure de Technologie Béni Mellal
Département ITG
DUT- IDIA 1 - S2
M122 - Ingénierie Logicielle avec Java et UML
Année universitaire: 2024-2025

Support de cours

Ingénierie logicielle avec Java et UML

Partie II: Modélisation UML

Chapitre IV: Diagrammes de classes et d'objets

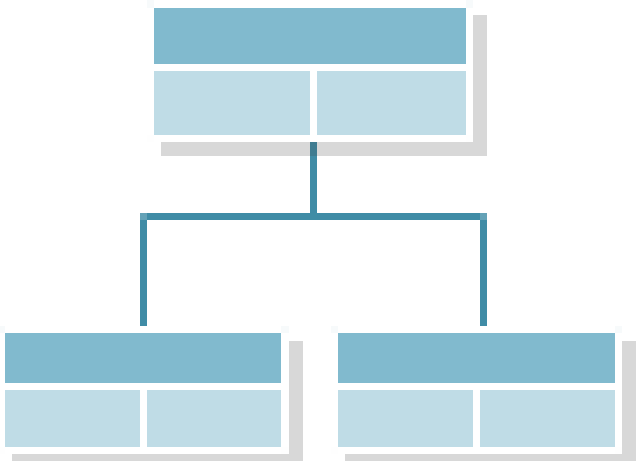
Pr. Siham BAKKOURI
Email: s.bakkouri@usms.ma

Plan

- DIAGRAMME DE CLASSES
  - DÉFINITIONS ET PRINCIPES DE BASE
  - REPRÉSENTATION D'UNE CLASSE
  - LES RELATIONS ENTRE LES CLASSES
  - LES CLASSES PARAMÉTRÉES (GÉNÉRIQUES)
- DIAGRAMME D'OBJETS
- EXERCICES

Introduction

- Les diagrammes de cas d'utilisation modélisent à **QUOI** sert le système : le système est composé d'objets qui interagissent entre eux et avec les acteurs pour réaliser ces cas d'utilisation.
- Le diagramme de classes exprime la structure statique du système en termes de classes et de relations entre ces classes
- L'intérêt du diagramme de classe est de modéliser les entités du système d'information
- **Le diagramme de classes** est sans doute le diagramme le plus important à représenter pour les méthodes d'analyse orientées objet ➔ C'est le point central de tout développement orienté objet.



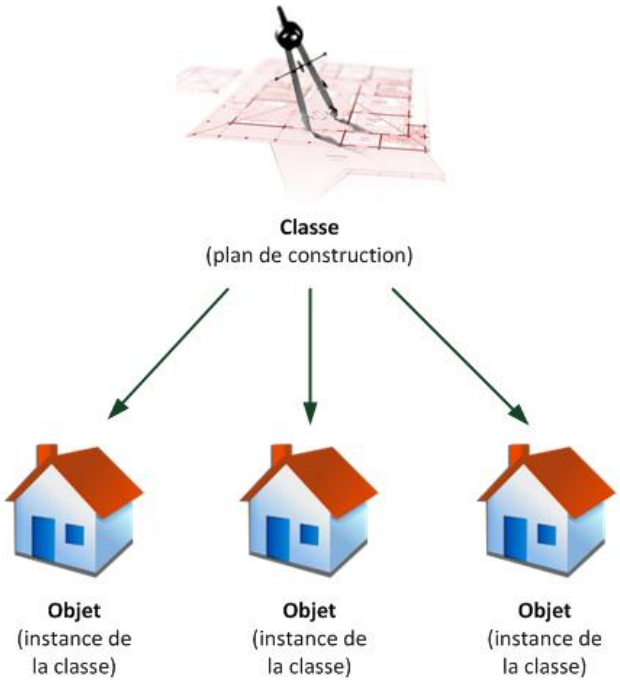
- Un **diagramme de classes** est une collection d'éléments de modélisation statique qui montre la structure d'un modèle.
- Un **diagramme de classes** fait abstraction des aspects dynamiques et temporels du Système
- **Concepts et instances**
  - Une instance est la concrétisation d'un concept abstrait.

Exemple:

- *Concept* : Stylo

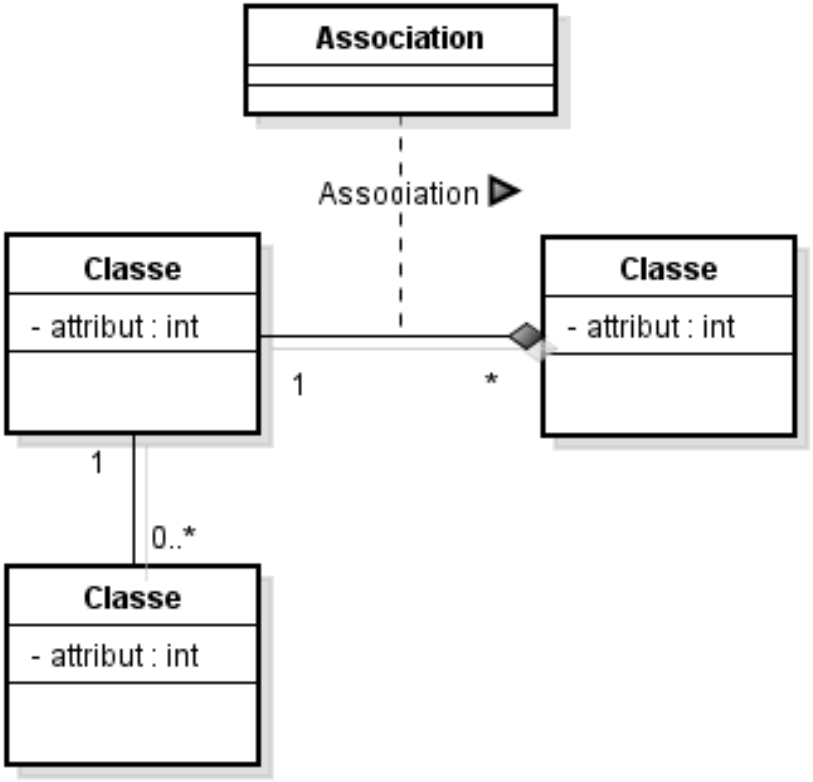
- *Instance* : le stylo que vous utilisez à ce moment précis est une instance du concept stylo: il a sa propre forme, sa propre couleur, son propre niveau d'usure, etc.

- Un **objet** est une instance d'une classe
- Une **classe** est une description abstraite d'un ensemble d'objets du domaine de l'application : elle définit leur structure, leur comportement et leurs relations
- Une **classe** représente la description d'un ensemble d'objets possédant les mêmes caractéristiques : elle spécifie la manière dont tous les objets de même type seront décrits (désignation, label, auteur, etc).
- Un **lien** est une instance **d'association**. Il représente la relation entre deux ou plusieurs entités.



Représentation d'une classe

Le diagramme de classes met en œuvre des **classes**, contenant des **attributs** et des **opérations**, et reliées par des **associations** ou des **généralisations**.

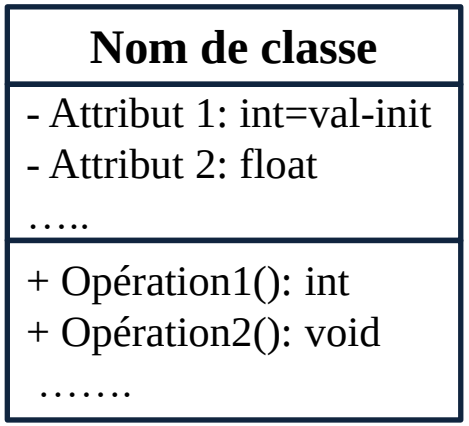


❑ Classe

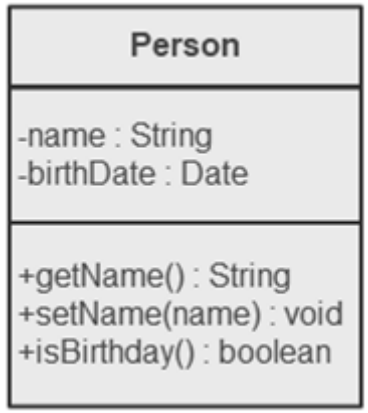
Une **classe** est représentée par un rectangle séparé en trois parties:

- La **première** partie représente le **nom** de la classe
- La **deuxième** partie représente les **attributs** de la classe
- La **troisième** partie représente les **opérations** de la classe

Formalisme



Exemple



❑ Les attributs

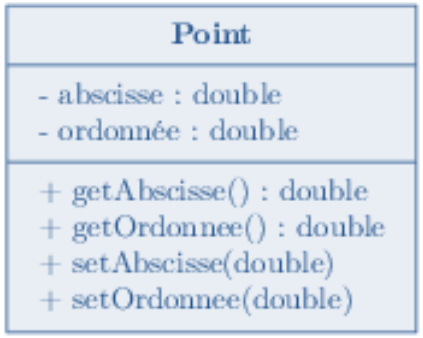
- Un attribut représente la modélisation d'une information élémentaire représentée par son nom et son format.

- UML définit **4 niveaux de visibilité** pour les attributs :

  - **public** (+) : l'élément est visible par tous les membres de la classe
  - **protégé** (#) : L'élément est visible dans la classe où il est déclaré et dans ses sous-classes.
  - **privé** (-) : l'élément n'est visible que par les objets de la classe dans laquelle il est déclaré
  - **package (~)** : propriété ou classe visible uniquement dans le paquetage où la classe est définie.

- **Syntaxe** de la déclaration d'un attribut :

  visibilité **nomAtt : *type* [*multiplicité*] = valeur-Init**



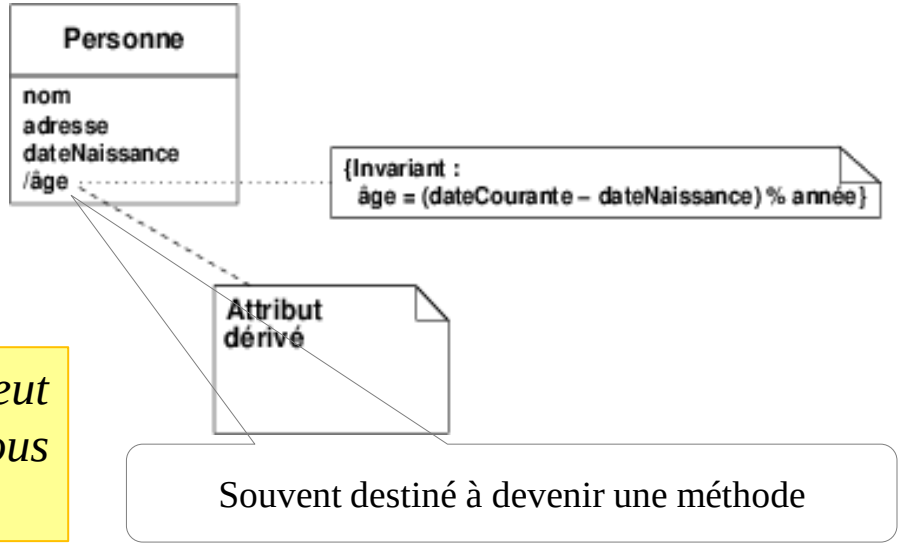
❑ Les attributs

- L'identifiant est un attribut particulier, qui permet de repérer de façon unique chaque objet, instance de la classe.

❑ Les attributs dérivés

- Les **attributs dérivés** peuvent être calculés à partir d'autres attributs et des formules de calcul.

- Les attributs dérivés sont symbolisés par l'ajout d'un / devant leur nom.
- **Attribut dérivé** : attribut dépendant d'autres attributs.



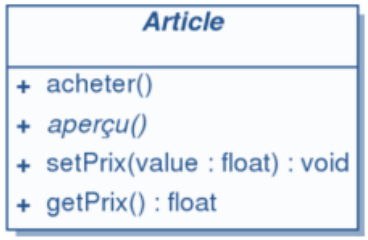
Lors de la conception, un attribut dérivé peut être utilisé comme marqueur jusqu'à ce que vous puissiez déterminer les règles à lui appliquer.

❑ Les opérations

- Une **opération** représente un élément de comportement des objets, défini de manière globale dans la classe.
- Une **opération** est une fonctionnalité assurée par une classe. La description des opérations peut préciser les paramètres d'entrée et de sortie ainsi que les actions élémentaires à exécuter.
- Comme pour les attributs, on retrouve **4 niveaux de visibilité** pour les opérations (Publique, Privée, protégée et package)

→ La **syntaxe** de la déclaration d'une **opération** est la suivante :

visibilité **nom_méthode ([paramètre_1, ..., param_N]) : type_renvoyé**



→ La **syntaxe** de la liste des **paramètres** est la suivante :

[NomClasse] **<nom_paramètre> : type = valeur_par_défaut**

❑ Méthodes et Opération

- Une **opération** est la spécification d'une **méthode** (sa signature) indépendamment de son implémentation.
- **UML 2** autorise également la définition des opérations dans n'importe quel langage donné.

Exemples de **méthodes** pour **l'opération** `fact(n:int):int` :

```
{ // implementation iterative
  int resultat =1~;
  for (int i = n~; i>0~; i--)
    resultat*=i~;
  return resultat~;
}
```

```
{ // implementation recursive
  if (n==0 || n==1)
    return 1~;
  return (n * fact(n-1))~;
}
```

❑ Attributs de classe

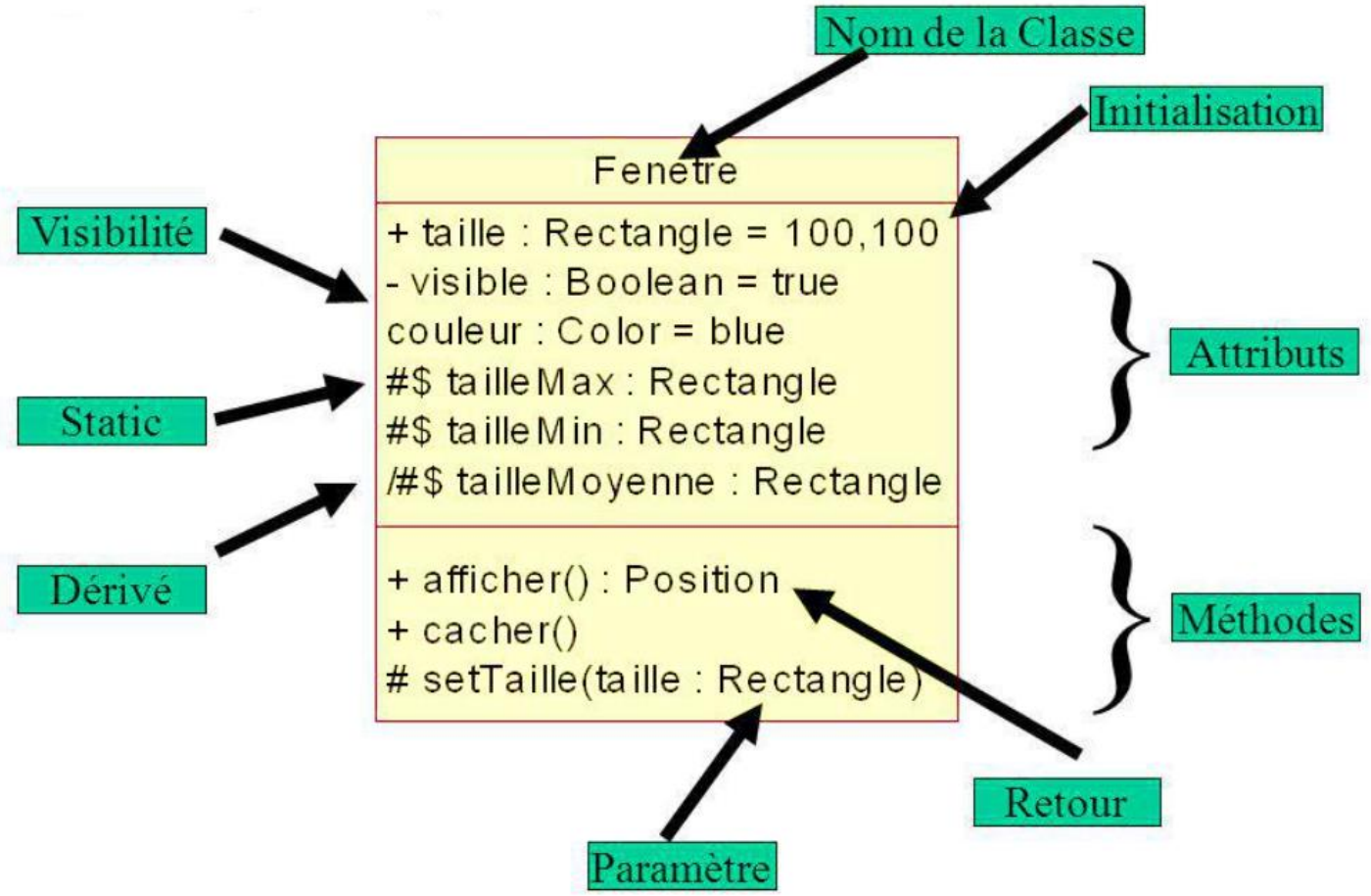
- Par défaut, les valeurs des attributs définis dans une classe diffèrent d'un objet à un autre. Parfois, il est nécessaire de définir un attribut de classe qui garde une valeur unique et partagée par toutes les instances.
- Graphiquement, un attribut de classe (***static***) est **souligné**

❑ Opérations de classe

- Similaires aux attributs de classe
- Les opérations de classe sont des propriétés de la classe, et non de ses instances.
- Contrairement aux objets de la classe, elles n'ont pas accès aux attributs des instances de la classe.
- Les opérations de classe représentent généralement les comportements ou les actions que les objets de la classe peuvent effectuer.

Représentation d'une classe

❑ Notation complète



Les relations entre les classes

Il existe plusieurs types de relations entre classes :

- Une **association** représente une relation sémantique entre les objets d'une classe.
- Une relation **d'agrégation** décrit une relation de contenance ou de **composition**.
- Une relation **d'héritage** est une relation de généralisation/spécialisation permettant l'abstraction.

→ Les associations entre classes

- Une **association** est une relation entre deux classes 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.
- Elle est représentée par un trait entre classes.

Exemple:

→ Les associations entre classes

Les propriétés d'une association entre deux classes comprennent plusieurs éléments paramétrables :

- **Nom de l'association :** Une association peut être nommée. Le nom est situé près de la terminaison de l'association. Cependant, ce nom est facultatif.
- **Nom de rôle:** indication sur la participation de la classe à l'association
- **Multiplicité:** définit le nombre d'instances de l'association pour une instance de la classe
- **Navigabilité:** La navigabilité précise la direction dans laquelle la navigation est possible, indiquant si un objet peut naviguer de l'un à l'autre ou dans les deux sens.

Ces propriétés sont essentielles pour définir correctement l'association entre les classes dans un modèle UML.

→ Les associations entre classes

a) Multiplicités des associations

- La notion de **multiplicité** permet le contrôle du nombre d'objets intervenant dans chaque instance d'une association.

Exemple : un article n'appartient qu'à une seule catégorie (1) ; une catégorie concerne plus de 0 articles, sans maximum (*).

- La **syntaxe** est :

MultMin .. MultMax

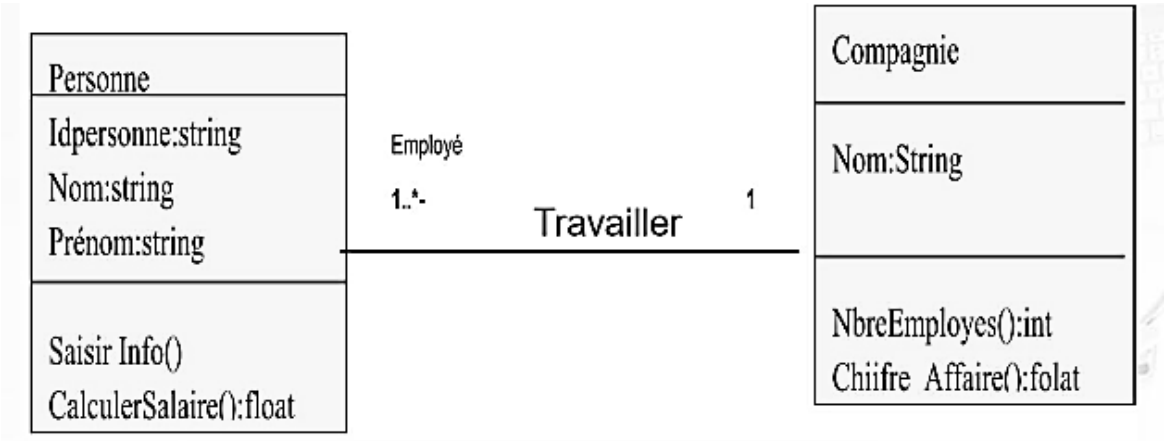
- ***** à la place de ***MultMax*** signie **plusieurs** sans préciser de nombre
- **n..n** se note aussi **n** , et **0..*** se note *

→ Les associations entre classes

a) Multiplicités des associations

Exemple : Dans ce diagramme, le nom de l'association est **Travailler**, le nom du rôle de la classe **Personne** pour l'association est: **Employé**. Le diagramme se lit comme suit :

- ✓ Une personne travail pour une seule entreprise.
- ✓ Dans une entreprise il y a une à plusieurs personnes



1	un et un seul
0..1	Zéro ou un
m..n	De m à n (entier)
*	Plusieurs
0..*	De zéro à plusieurs
1..*	d'un à plusieurs

→ Les associations entre classes

b) Navigabilité d'une association

- La **navigabilité** permet de spécifier dans quel(s) **sens** il est possible de traverser l'association à l'exécution.
- On restreint la navigabilité d'une association à un seul sens à l'aide d'une flèche.

Exemple :Connaissant un article on connaît les commentaires, mais pas l'inverse.

- On peut aussi représenter les associations navigables dans un seul sens par des attributs.

→ Les types d'associations

❑ Association Binaire

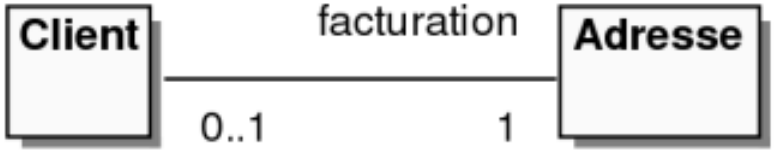
- Dans une **association binaire** la multiplicité sur la terminaison cible contraint le nombre d'objets de la classe cible pouvant être associés à un **seul objet** donné de la classe source (la classe de l'autre terminaison de l'association).
- Une **association binaire** est matérialisée par un trait plein entre les classes associées.
- Elle peut être ornée d'un nom, avec éventuellement une précision du sens de lecture (▸ ou ◂).
- Quand les deux extrémités de l'association pointent vers la même classe, l'association est dite **réflexive**.

→ Les types d'associations

❑ Association Binaire

i. Association unidirectionnelle de 1 vers 1

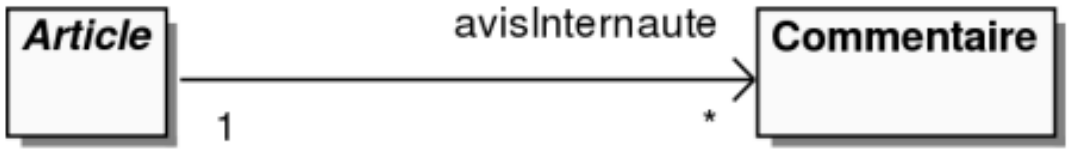
ii. Association bidirectionnelle de 1 vers 1



→ Les types d'associations

❑ Association Binaire

iii. Association unidirectionnelle de 1 vers *

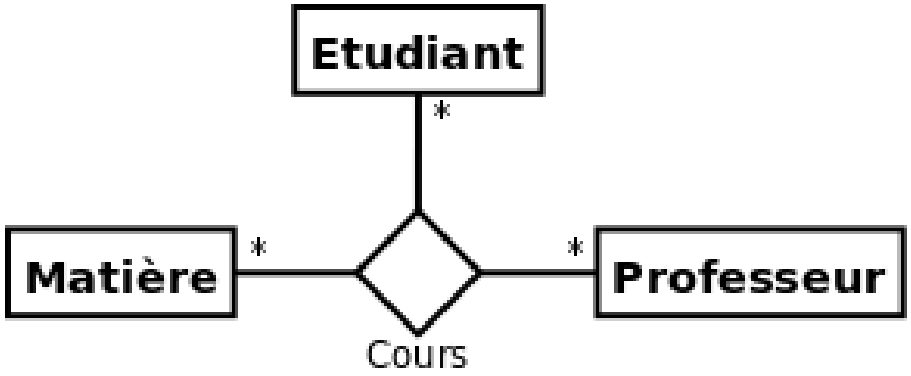


iii. Association bidirectionnelle de * vers *

→ Les types d'associations

❑ Association n-aire

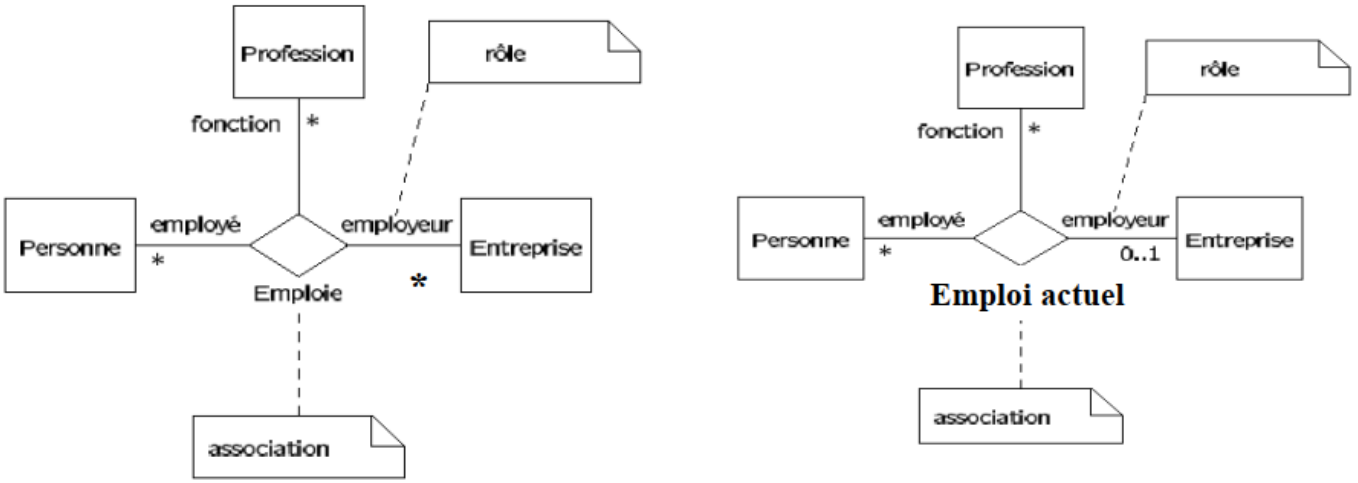
- Dans **une association n-aire**, la multiplicité apparaissant sur le lien de chaque classe s'applique sur une instance de chacune des classes, à l'exclusion de la classe-association et de la classe considérée.
- Une association n-aire lie plus de deux classes.
- On représente une ***association n-aire*** par un grand losange avec un chemin partant vers chaque classe participante. Le nom de l'association, le cas échéant, apparaît à proximité du losange.



→ Les types d'associations

❑ Association n-aire

Exemple



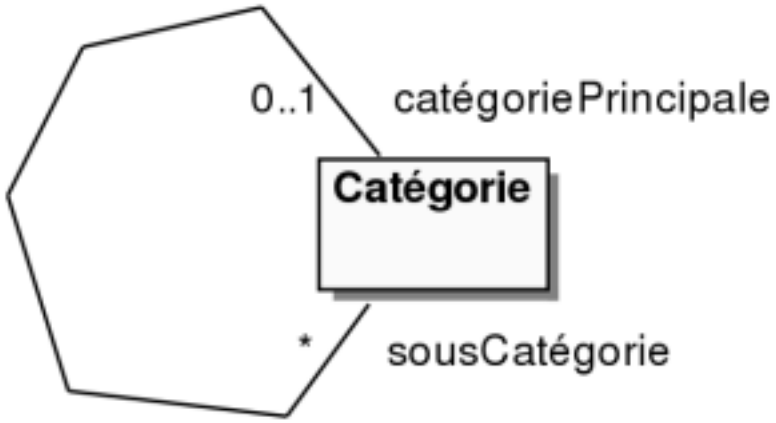
Si (A,B,C) représente un triplet d'instances de type (***Personne***, ***Profession***, ***Entreprise***) :

- Pour obtenir la multiplicité côté ***C*** , on fixe un couple d'instances (***A,B***) et on examine le nombre d'instances ***C_min*** et ***C_max*** qui lui sont reliées à travers l'association .
- Même raisonnement pour trouver les multiplicités côté ***A*** ou côté ***B*** .
- Par exemple : Une personne ayant une profession (couple (***A,B***) fixé) peut être employée dans 0 à plusieurs entreprises selon l'association « Emploie » alors qu'elle peut posséder 0 ou 1 employeur (entreprise) selon l'association « Emploi actuel » .

→ Les types d'associations

❑ Association Réflexive

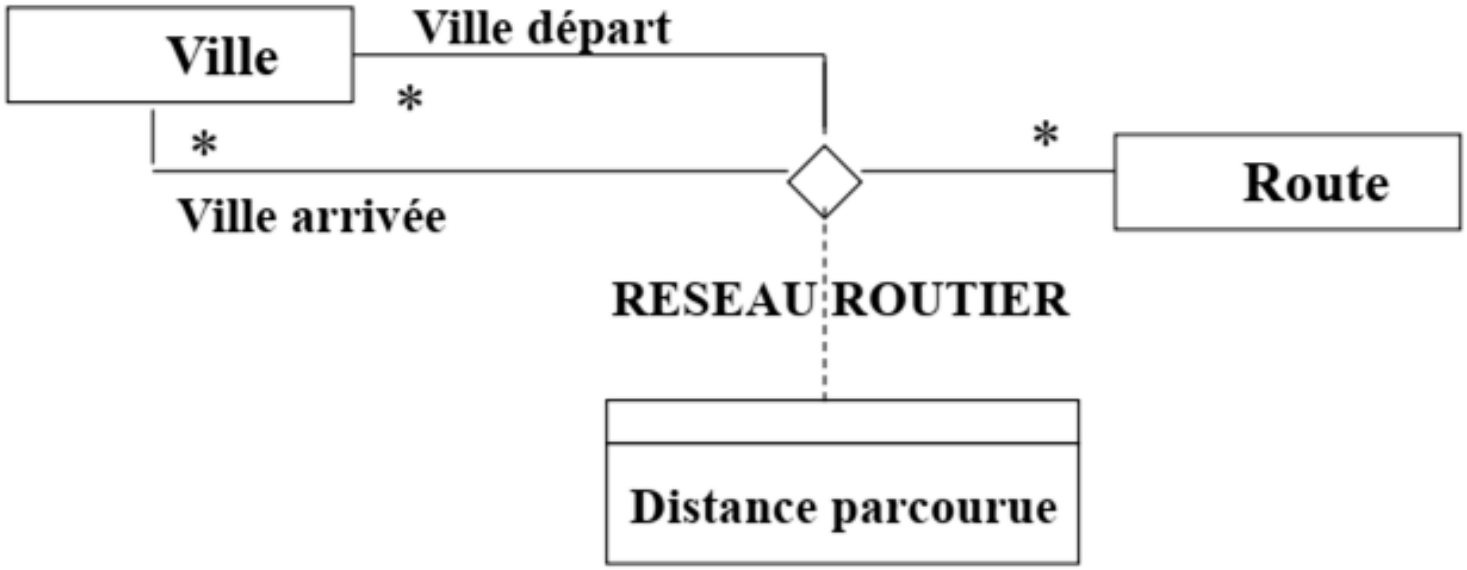
- Dans ce type d'association, les deux extrémités de l'association pointent vers la même classe. Cela signifie qu'une classe est associée à elle-même.
- L'association la plus utilisée est l'association binaire (reliant deux classes).
- Dans les associations réflexives, les **rôles** des classes associées sont souvent **obligatoires**. Cela signifie qu'une instance de la classe doit être associée à au moins une autre instance de la même classe.



→ **Les types d'associations**

- **Association Réflexive**

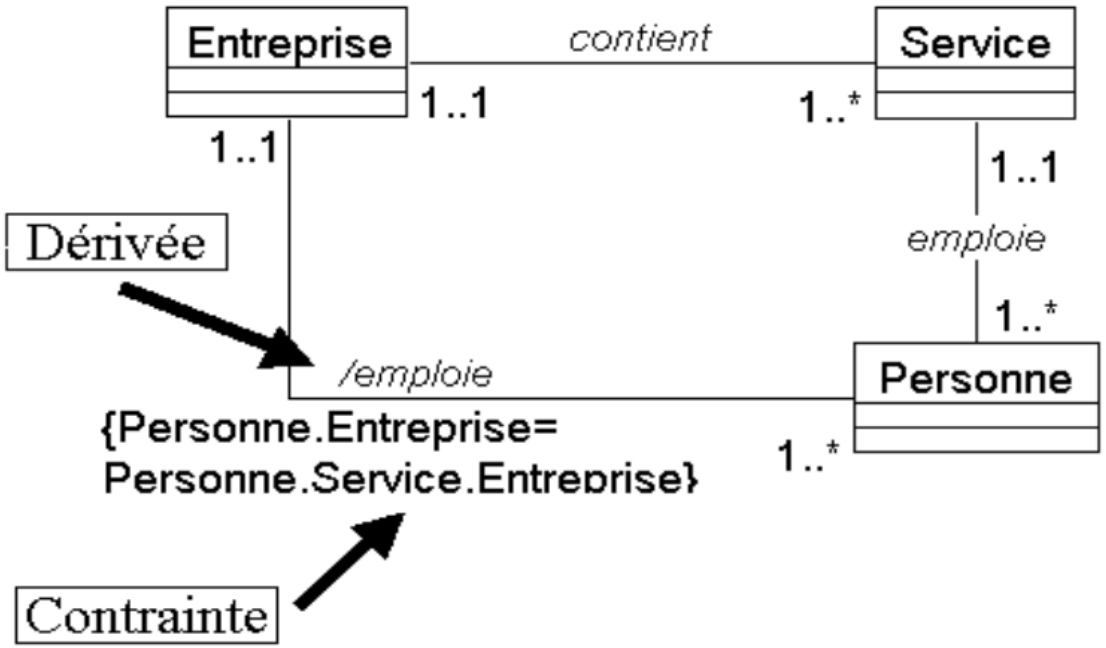
Exemple: Association réflexive n-aire



→ Les types d'associations

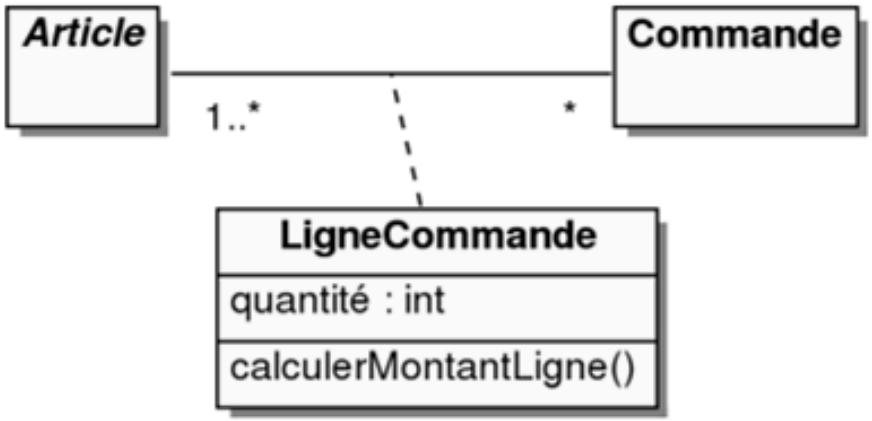
❑ Association dérivée

- Une association dérivée est conditionnée ou peut être déduite à partir d'autres autres associations. On utilise également le symbole « / ».



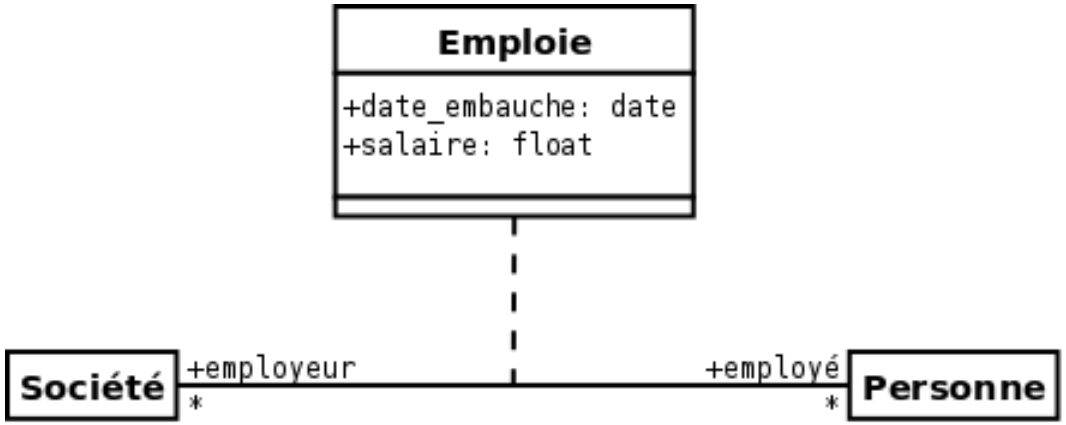
→ Classe- Association

- Une association peut être raffinée et avoir ses propres attributs, qui ne sont disponibles dans aucune des classes qu'elle lie.
- Comme, dans le modèle objet, seules les classes peuvent avoir des attributs, cette association devient alors une classe appelée **classe-association**
- Une **classe-association** possède les caractéristiques des **associations** et des **classes** : elle se connecte à deux ou plusieurs classes et possède également des **attributs** et des **opérations**.



→ Classe- Association

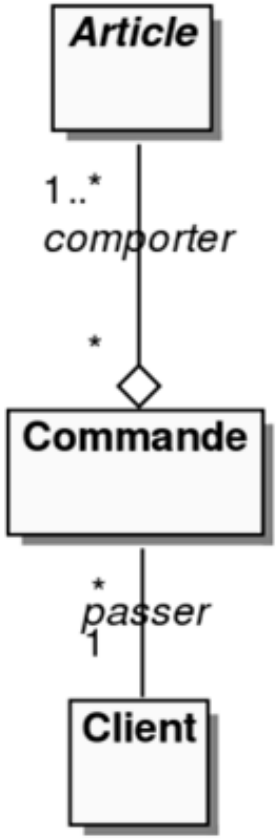
Exemple



- l'association Emploie entre une société et une personne possède comme propriétés le salaire et la date d'embauche.
- En effet, ces deux propriétés n'appartiennent **ni à la société**, qui peut employer plusieurs personnes**, ni aux personnes**, qui peuvent avoir plusieurs emplois.
- Il s'agit donc bien de propriétés de l'association Emploie. Les associations ne pouvant posséder de propriété, il faut introduire un nouveau concept pour modéliser cette situation : celui de **classe-association**.

→ L'agrégation

- Une **agrégation** est une forme particulière d'association. Elle représente la relation d'inclusion d'un élément dans un ensemble.
- On représente l'agrégation par l'ajout d'un **losange vide** du côté de **l'agrégat**.
- Une agrégation dénote une relation d'un ensemble à ses parties. L'ensemble est l'agrégat et la partie l'agrégé. → 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**.



→ La composition

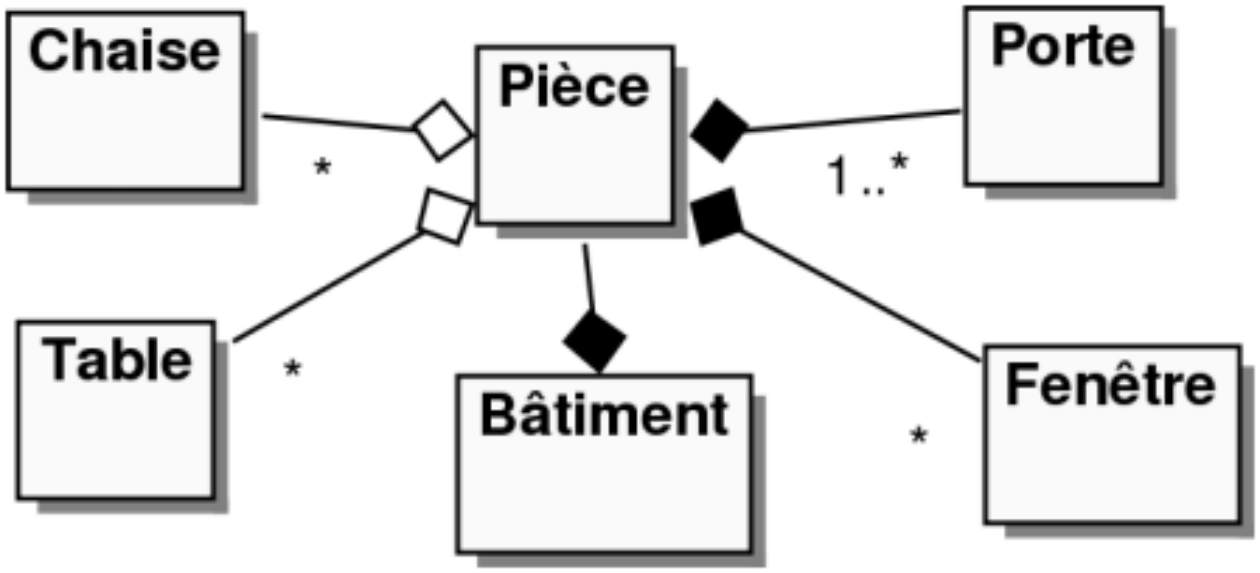
- La relation de **composition** décrit une **contenance** structurelle entre instances.
- On représente la composition par un **losange plein**.
- La destruction et la copie de l'objet composite (l'ensemble) impliquent respectivement la destruction ou la copie de ses composants (les parties).
- Une instance de la partie n'appartient jamais à plus d'une instance de l'élément composite

→ Composition et agrégation

- Lorsqu'il existe une relation du tout à ses parties, nous parlons d'agrégation ou de composition.
- La composition également appelée agrégation composite ou agrégation forte.
- Pour décider entre l'usage de la composition et de l'agrégation, il est essentiel de se poser les questions suivantes :
  - Est-ce que la destruction de l'objet composite (le tout) entraîne nécessairement la destruction des objets composants (les parties) ?
  - Lorsque le composite est copié, doit-on également copier les composants ou peut-on les réutiliser ?
- Si les réponses à ces deux questions sont affirmatives, alors l'utilisation de **la composition** est appropriée.

→ Composition et agrégation

Exemple

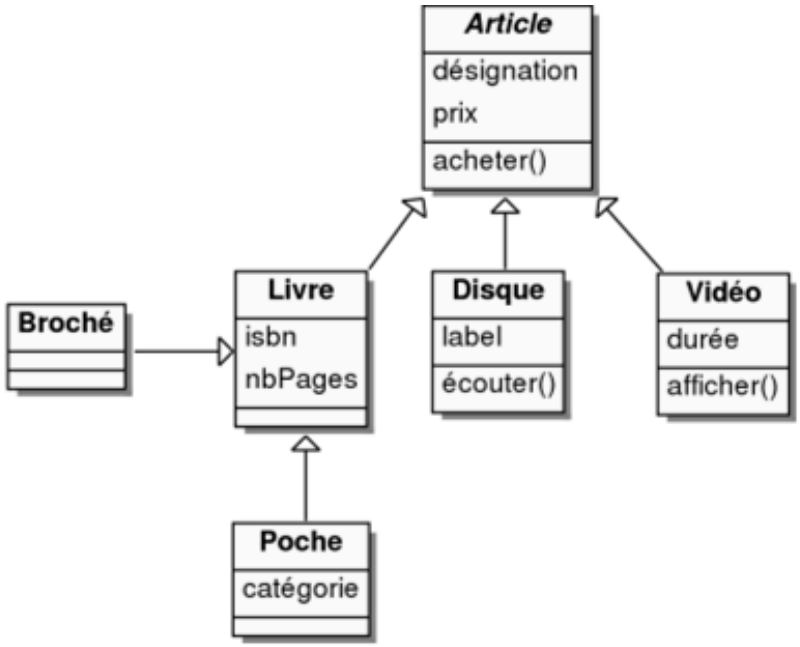


→ L'héritage

- L'héritage une relation de **spécialisation/généralisation**.
- Les éléments spécialisés héritent de la structure et du comportement des éléments plus généraux (attributs et opérations)

Exemple

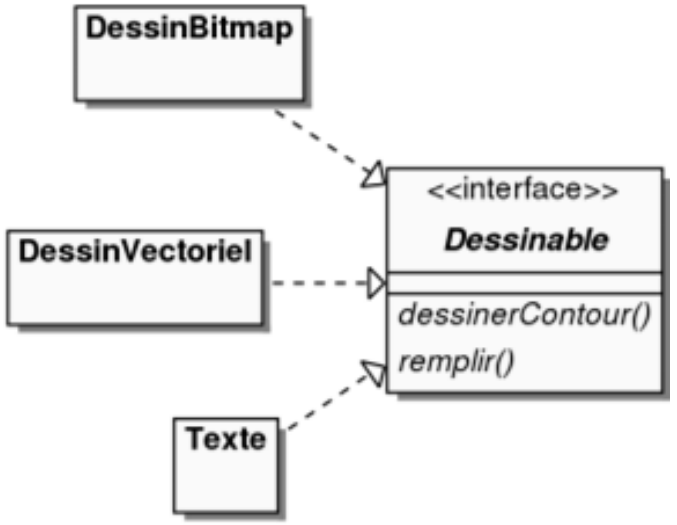
- La classe ***enfant*** possède toutes les propriétés de ses classes parents (attributs et opérations)
- La classe ***enfant*** est la classe spécialisée (ici *Livre*)
- La classe ***parent*** est la classe générale (ici *Article*)
- Toutefois, elle n'a pas accès aux propriétés privées.



→ La réalisation (les interfaces)

- Le rôle d'une interface est de regrouper un ensemble d'opérations assurant un service cohérent offert par un classeur et une classe en particulier.
- Une interface est dénie comme une classe, avec les mêmes compartiments. On ajoute le stéréotype « ***interface*** » avant le nom de l'interface.
- On utilise une relation de type réalisation entre une interface et une classe qui implémente.

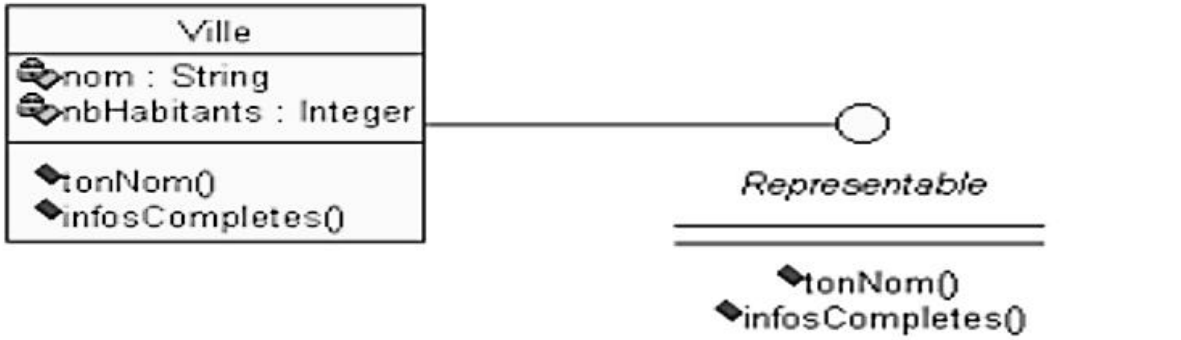
- Les classes implémentant une interface doivent implémenter toutes les opérations décrites dans l'interface



→ La réalisation (les interfaces)

Exemple

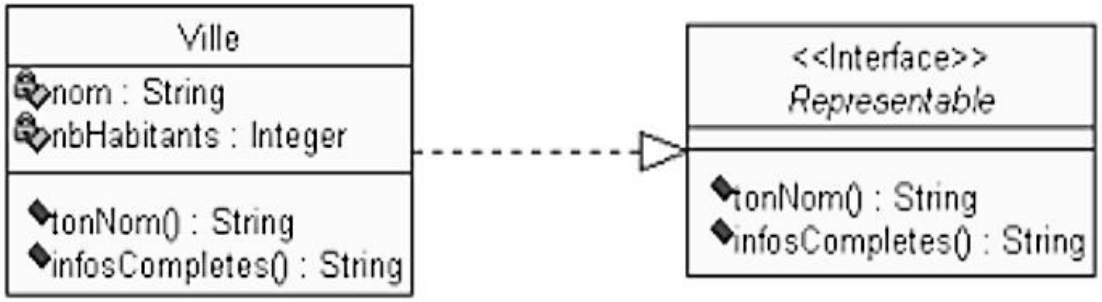
1er notation



Classe avec implémentation des méthodes

Classe générique sans implémentation des méthodes

2ème notation

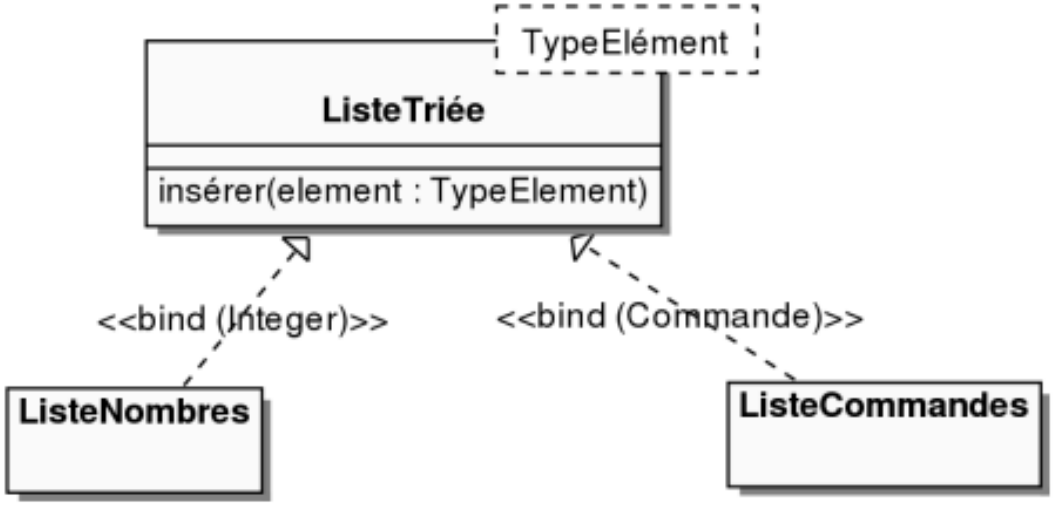


Les classes paramétrées

- Pour définir une classe générique et paramétrable en fonction de valeurs et/ou de types :

  Définition d'une classe paramétrée par des éléments spécifiés dans un rectangle en pointillés ;

- Utilisation d'une dépendance stéréotypée «***bind*** » pour définir des classes en fonction de la classe paramétrée.



- Java5 : généricité
- C++ : templates

→ Définitions

- Le **diagramme d'objets** représente les objets d'un système à un instant donné.

  Il permet de :

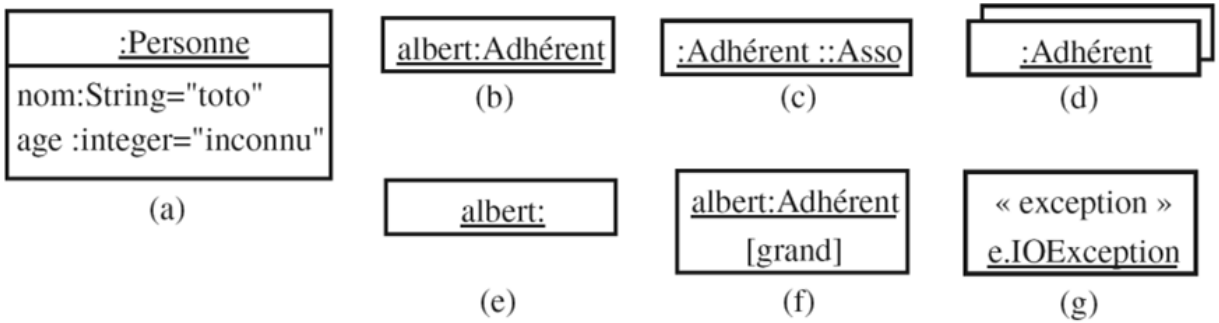
  - Illustrer le modèle de classes (en montrant un exemple qui explique le modèle)
  - Préciser certains aspects du système (en mettant en évidence des détails imperceptibles dans le diagramme de classes)
  - Exprimer une exception (en modélisant des cas particuliers, des connaissances non généralisables. . .)

Le diagramme de classes modélise des règles et le diagramme d'objets modélise des faits

→ Représentation des objets

- Comme les classes, on utilise des cadres compartimentés.
- En revanche, les noms des objets sont soulignés et on peut rajouter son identifiant devant le nom de sa classe.

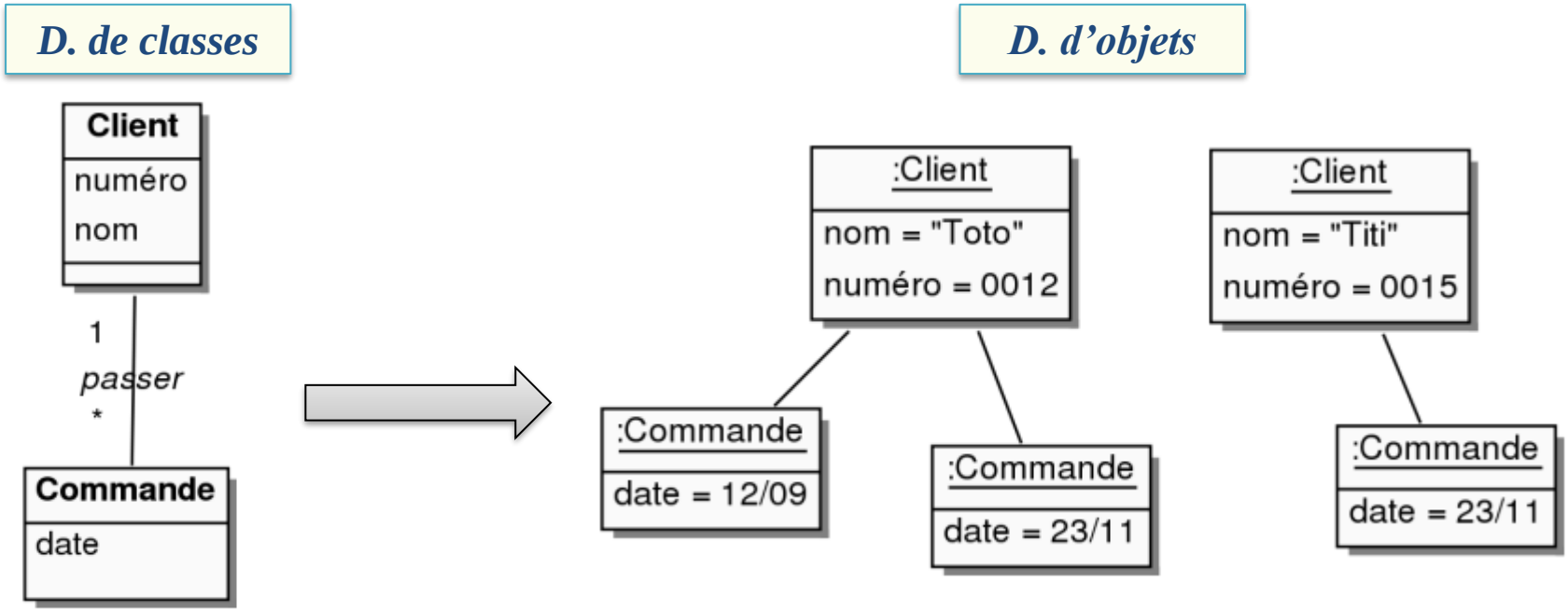
Exemple:



- Les valeurs **(a)** ou l'état **(f)** d'un objet peuvent être spécifiées.
- Les instances peuvent être anonymes (**a, c, d**), nommées (**b, f**), orphelines (**e**), multiples (**d**) ou stéréotypées (**g**).

→ Diagramme de classes et diagramme d'objets

- Le diagramme de classes ***contraint*** la structure et les liens entre les objets.
- Le diagramme d'objets ***cohérent*** avec le diagramme de classes :

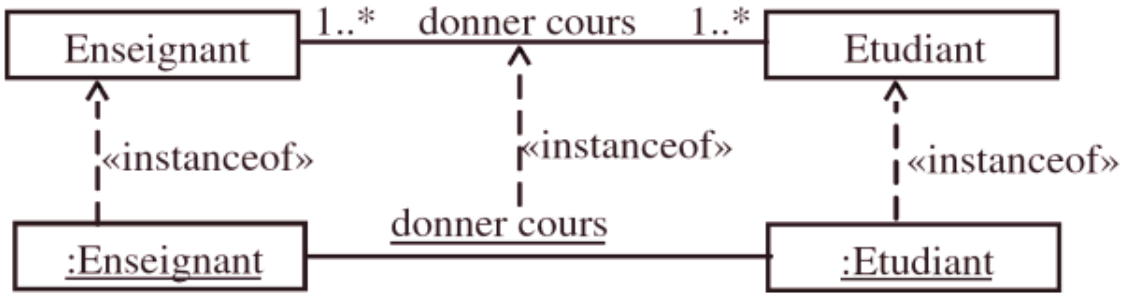


→ Les liens

- Un **lien** est une **instance d'une association**.
- Un lien se représente comme une association mais s'il a un nom, il est souligné.

→ Relation de dépendance d'instanciation

- La relation de dépendance d'instanciation (stéréotypée) décrit la relation entre un classeur et ses instances.
- Elle relie, en particulier, les associations aux liens et les classes aux objets.



Attention

On ne représente pas les multiplicités qui n'ont aucun sens au niveau des objets.

Exercice 1

- Une personne est caractérisée par son **nom**, son **prénom**, son **sexe** et son **âge**.
- Les objets de classe **Personne** doivent pouvoir calculer leurs revenus et leurs charges.
- Les attributs de la classe sont privés, le nom, le prénom ainsi que l'âge de la personne doivent être accessibles par des opérations publiques.

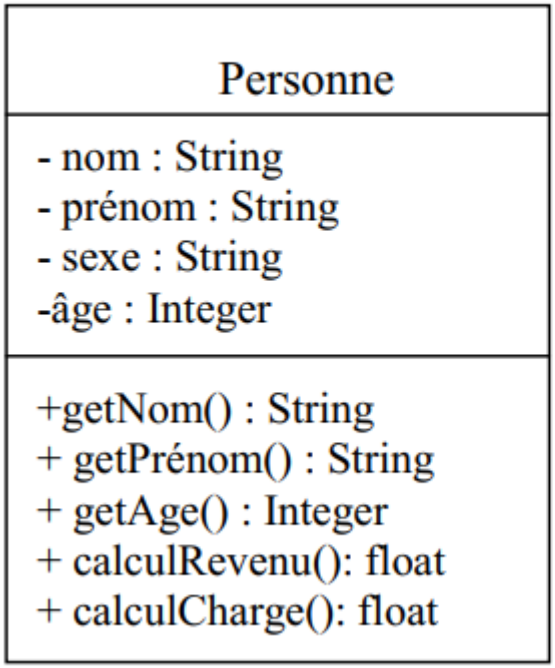
1. Donnez une représentation UML de la classe **Personne**, en remplissant tous les compartiments adéquats.

- Deux types de revenus sont envisagés : d'une part le **salaire** et d'autre part toutes les autres **sources de revenus**. Les deux revenus sont représentés par des nombres réels (float).
- Pour calculer les charges globales, on applique un coefficient fixe de **20%** sur les salaires et un coefficient de 15% sur les autres revenus.

2. Enrichissez la représentation précédente pour prendre en compte ces nouveaux éléments.

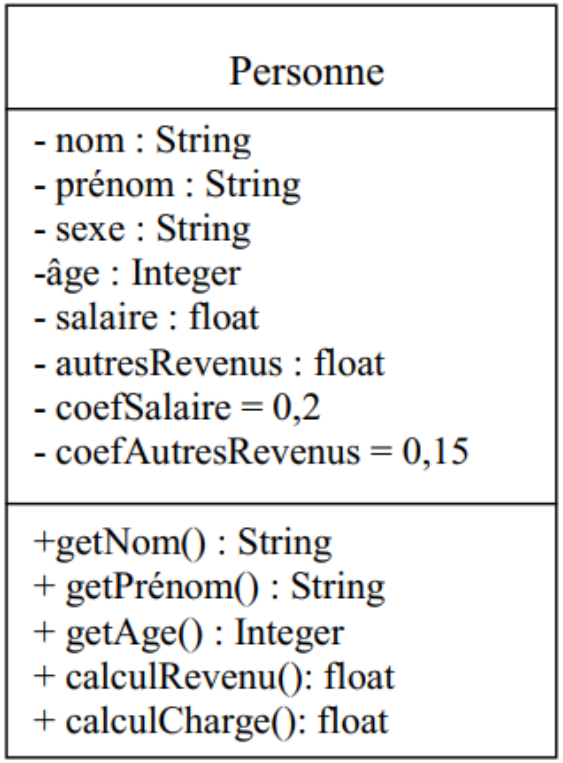
Exercice 1 – Correction

1) La classe Personne contient 4 attributs et deux méthodes (opérations) :



Exercice 1 – Correction

2) Après enrichissement, on ajoute 3 attributs : salaire, autresRevenus coefSalaire et coefAutresRevenus:



Exercice 2

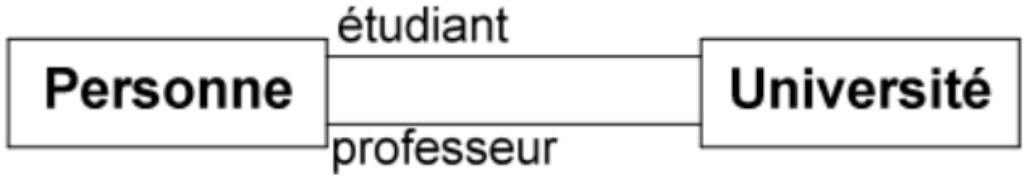
Pour chacun des énoncés suivants, donnez un diagramme de classes :

1. Tout écrivain a écrit au moins une œuvre
2. Les personnes peuvent être associées à des universités en tant qu'étudiants aussi bien qu’en tant que professeurs.
3. Un rectangle a deux sommets qui sont des points. On construit un rectangle à partir des coordonnées de deux points. Il est possible de calculer sa surface et son périmètre, ou encore de le translater.
4. Les cinémas sont composés de plusieurs salles. Les films sont projetés dans des salles. Les projections correspondantes ont lieu à chacune à une heure déterminée.

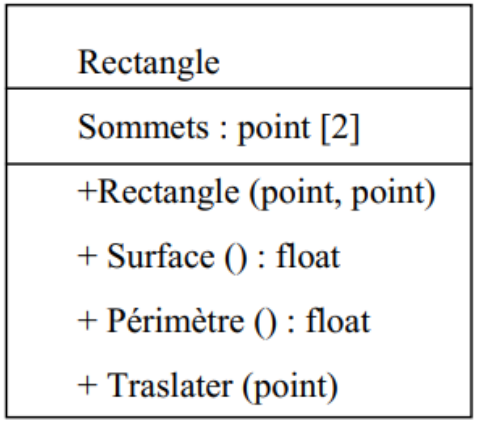
Exercice 2 – Correction

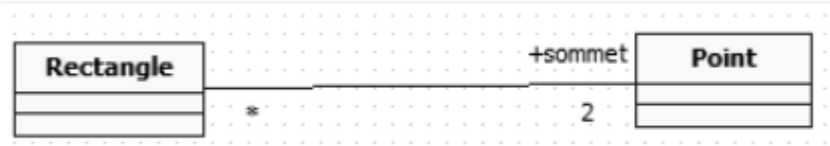
1)

2)



3)





Exercice 2 – Correction

4)