ECOLE MAROCAINE DES SCIENCES DE L'INGENIEUR
Membre de HONORIS UNITED UNIVERSITIES

Conception Orientée Objet

3IIR

Chap.2-4:Diagramme de Classes, Contraintes et diagramme d'objet

Pr. Khalid SRAIDI

2024/2025

Introduction

- Le 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 :
- Les diagrammes de classes permettent de spécifier **QUI** intervient à l'intérieur du système
- Ils spécifient également quels liens peuvent entretenir les objets du système.

Diagramme de Classes: Exemple

Société XYZ,

N°Facture : 0034/2019, Le 01/03/2019

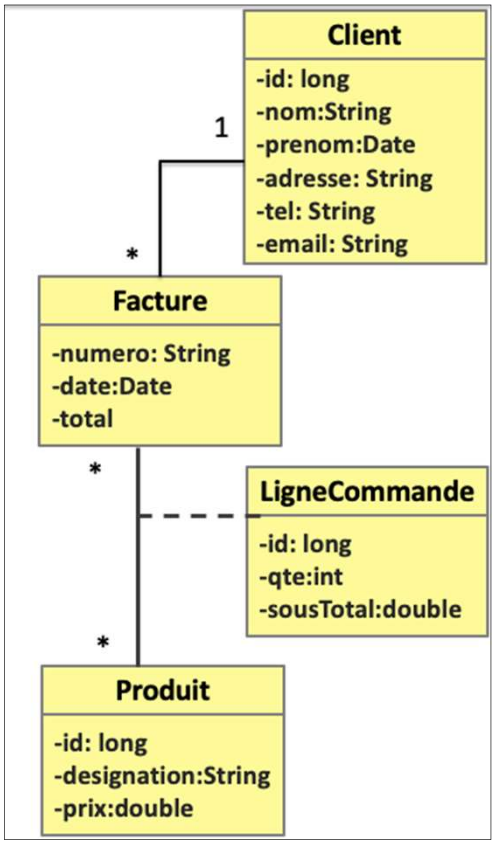
Client
Nom : Ali Baba
Adresse : Route 1, N°100 - Casablanca
Tél : 0666666
Email : uvw@m.ma

DÉSIGNATION	PU	QTE	SOUS TOTAL
GALAXY S4	1000,00	2	2000,00
GALAXY S5	1500,00	2	3000,00
PC HP 4555	4500,00	1	4500,00
IMPRIMANTE LASER 200	1300,00	1	1300,00
BUREAU PLAX-55	1200,00	1	1200,00

Total : 12000,00

Règlement Espèce

Réalité (exécution)



Modèle

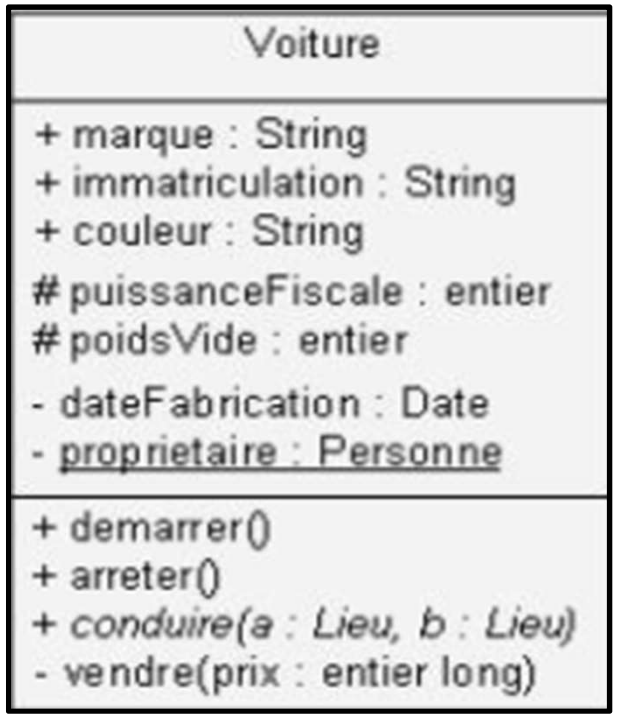
Diagramme de Classes

- La construction du diagramme de classes constitue l'objectif de tous processus de modélisation « objet ».
- Le système est composé d'objets qui interagissent entre eux et avec les acteurs pour réaliser ces cas d'utilisation.
- Les diagrammes de classes permettent de spécifier la structure et les liens entre les objets dont le système est composé.

Qu’est ce qu’une Classe ?

«Une classe est une description d’un ensemble d’objets ayant une sémantique, des attributs, des méthodes et des relations en commun. Un objet est une instance d’une classe»

Exemple d'une classe



Objet ou une instance de la classe voiture

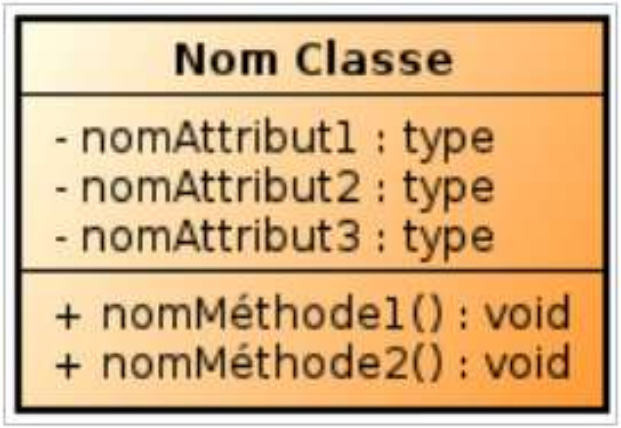
Représentation graphique d'une Classe

Une classe est représentée par un rectangle (***appelé aussi classeur***) divisé en 3 compartiments.

1°) Le premier compartiment contient le *nom de la classe qui :*
- représente le type d'objet instancié.
- débute par une lettre majuscule.

2°) Le deuxième compartiment contient les *attributs*.

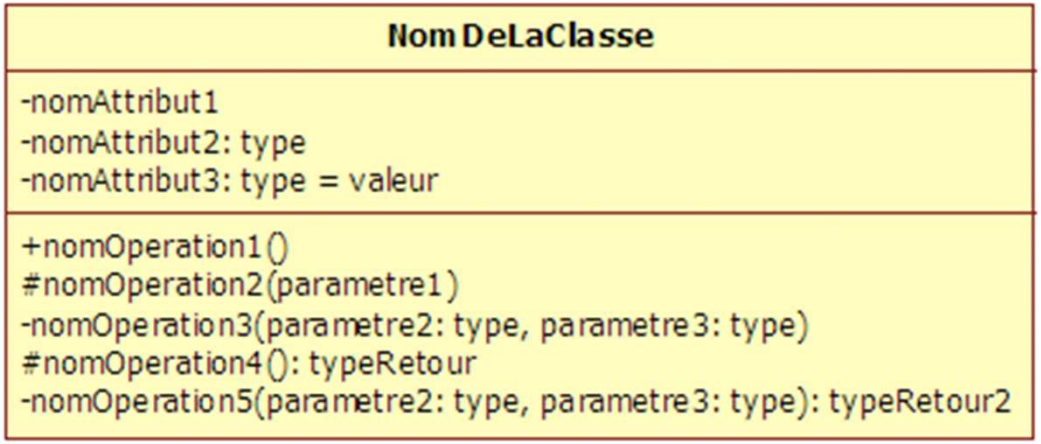
3°) Le troisième compartiment contient les *méthodes*.



Syntaxe des attributs et des méthodes

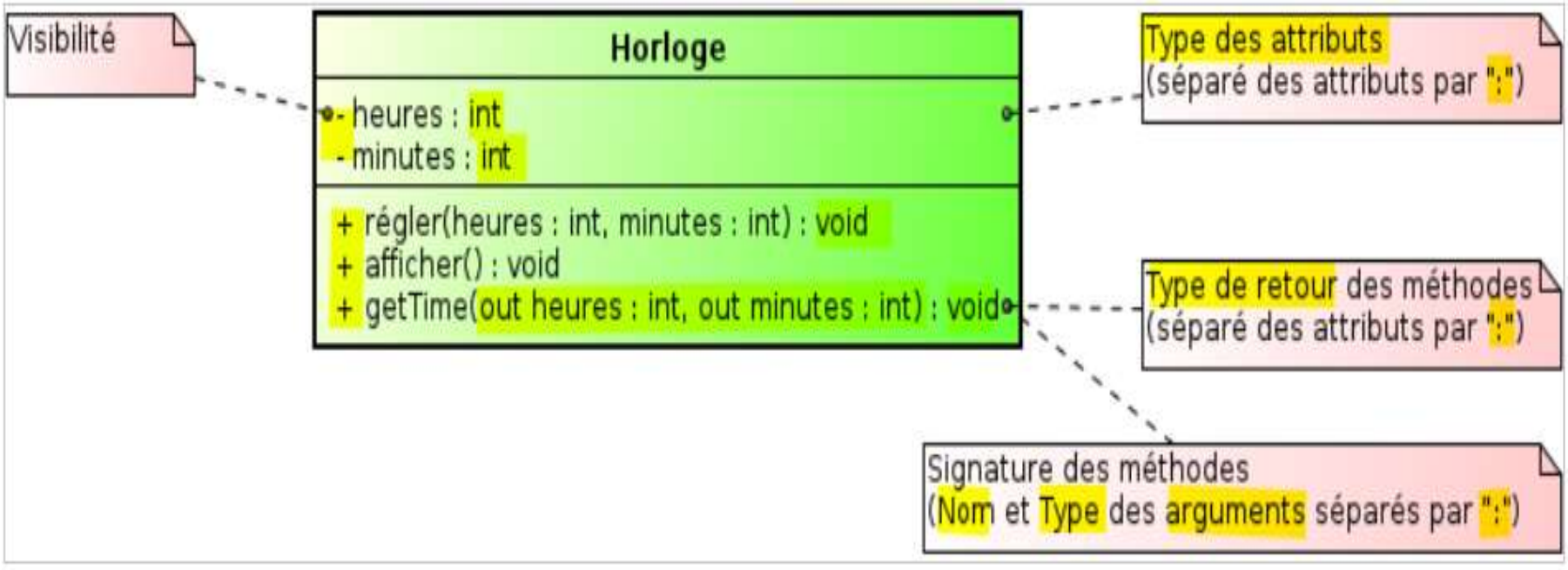
Nous pouvons détailler la classe en indiquant :

- La visibilité (encapsulation) des méthodes et des attributs.
- Le type de chaque attribut.
- La signature de chaque méthode.
- Le type de valeur retournée par chaque méthode.



Syntaxe des attributs et des méthodes
Exemple

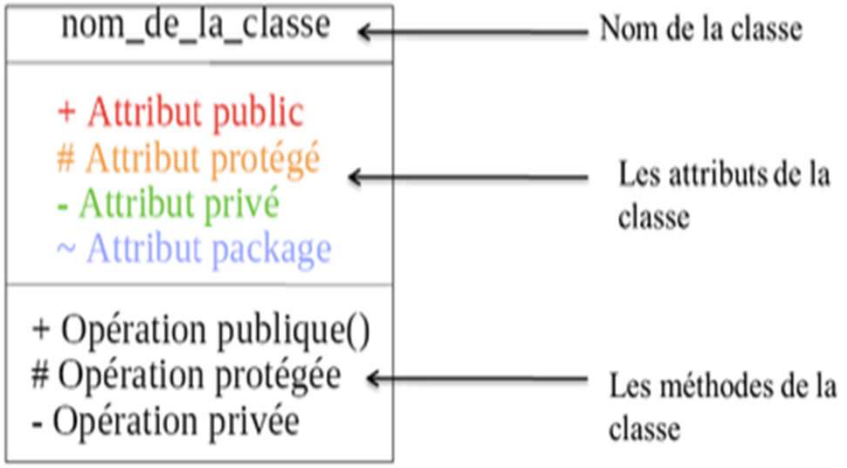
Exemple : *La classe* ***Horloge****.*



Syntaxe des attributs et des méthodes: Visibilité

- **La visibilité (encapsulation) des méthodes et des attributs**

Type de visibilité	Symbole à placer devant l'attribut ou la méthode
public : élément non encapsulé visible par tous	+
privé : élément encapsulé visible seulement dans la classe (les objets de cette classe) et non pas dans les sous-classe.	-
protégé : élément encapsulé visible dans la classe et dans les sous-classes.	#
package : élément encapsulé visible dans les classes du même paquetage.	~



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.

2. En plus des informations fournies précédemment, 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. Pour calculer les charges globales, on applique un coefficient fixe de 20% sur les salaires et de 15% sur les autres revenus.
Enrichissez la représentation précédente pour prendre en compte ces nouveaux éléments.

3. Un objet de la classe Personne peut être créé à partir du nom et de la date de naissance. il est possible de changer le prénom d'une personne. par ailleurs, le calcul des charges ne se fait pas de la même manière lorsque la personne décède.
Enrichissez encore la représentation précédente pour prendre en compte ces nouveaux éléments.

Solution

Les Relations

Relations entre les classes

En COO avec UML, il existe plusieurs types de relations entre les classes dont on cite:

- Dépendance
- Association
- Héritage
- Agrégation
- Composition

La Relation de Dépendance

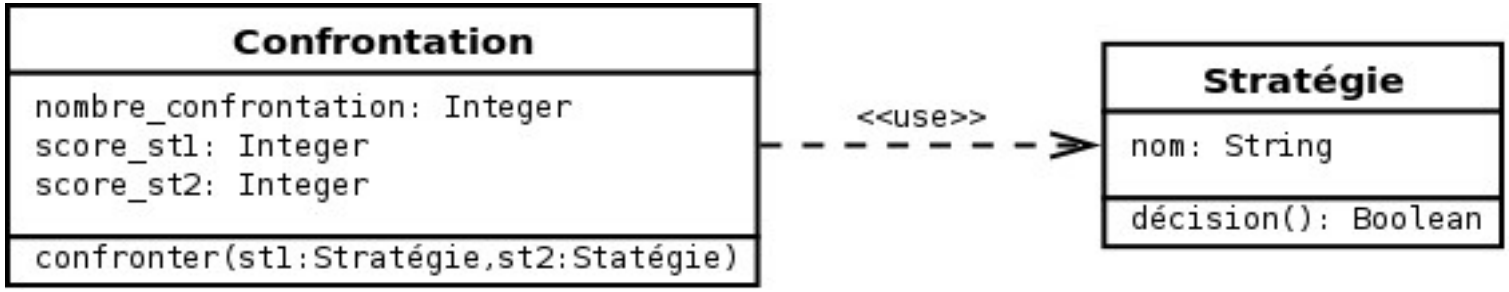
Une dépendance peut s’interpréter comme une relation de type «utilise un ». Elle est habituellement utilisée lorsqu'une classe utilise un objet d'une autre classe comme argument dans la signature d’une méthode ou alors lorsque l'objet de l'autre classe est créé à l'intérieur de la méthode. Dans les deux cas la durée de vie de l'objet est très courte, elle correspond à la durée d'exécution de la méthode.

La Relation de Dépendance

❑ Notation:

Elle est représentée par un trait discontinu orienté, reliant les deux classes. La dépendance est souvent stéréotypée **«use»** pour mieux expliciter le lien sémantique entre les éléments du modèle.

Exemple:



La Relation d'Association

- Une association est une relation structurelle entre objets. Une association est souvent utilisée pour représenter les liens possibles entre objets de classes données.
- Elle est représentée par un trait entre classes
- Il y a deux type d'associations:

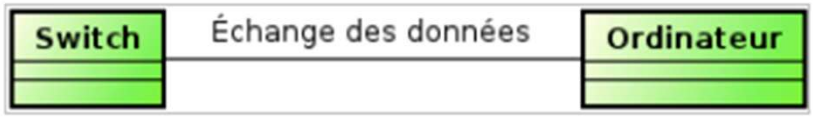
  - Binaire: est une associations qui lie deux classes

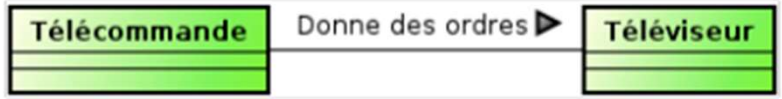
  - n-aire: est une association qui lie plusieurs classes

La Relation d'Association

Nous pouvons détailler l'association en indiquant :

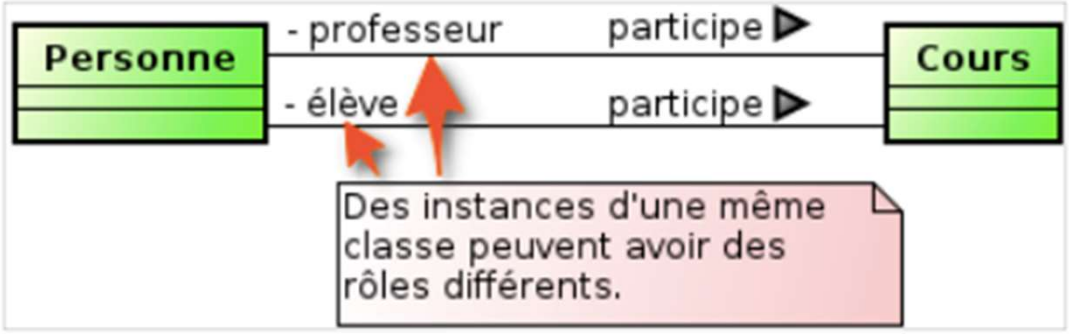
- **Le nom de l'association** : L'association peut être orné d'un texte, avec un éventuel sens de lecture, qui permet de nous informer de l'intérêt de cette relation. Nous rajoutons une phrase courte permettant de préciser le contexte de cette association.





La Relation d'Association

- **Le rôle :** Chaque extrémité d'une association peut être nommée. Ce nom est appelé rôle et indique la manière dont l'objet est vu de l'autre coté de l'association. Lorsqu'un objet A est lié à un autre objet B par une association, cela se traduit souvent par un attribut supplémentaire dans A qui portera le nom du rôle B. (et inversement).



La Relation d'Association

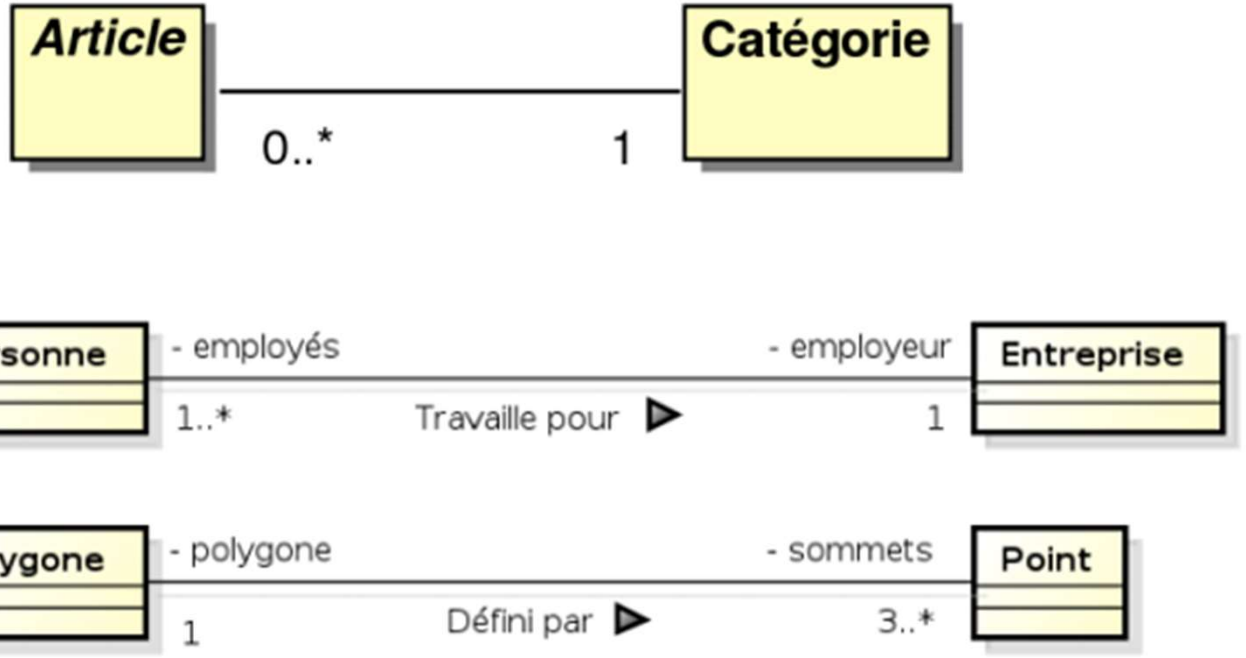
- **La cardinalité (ou multiplicité) :** La notion de multiplicité permet le contrôler du nombre d'objets intervenant dans chaque instance d'une association.

Cardinalité	Signification
0..1	Zéro ou une fois
1..1 (ou 1)	Une et une seule fois
0..* (ou *)	De zéro à plusieurs fois
1..*	De une à plusieurs fois
m..n	Entre **m** et **n** fois
n..n (ou n)	**n** fois

La Relation d'Association

- **Cardinalité**

Exemple:



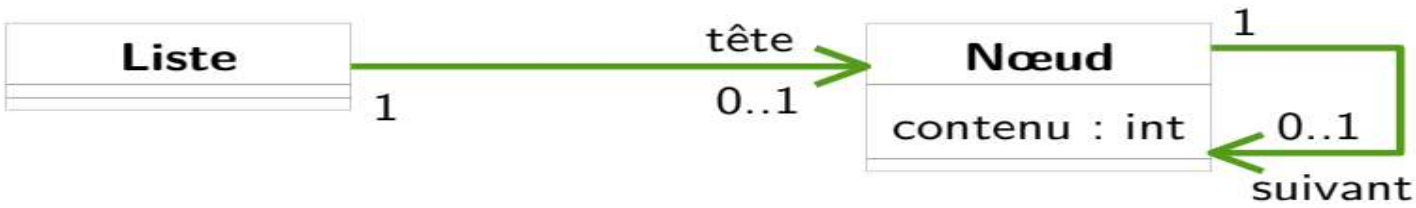
La relation de navigabilité

La **navigabilité** permet de spécifier dans quel(s) sens il est possible de traverser l'association à l'exécution.

Orientation d'une association:

- Restreint l'accessibilité des objets
- Depuis un A, on a accès aux objets de B qui lui sont associés, mais pas l'inverse

Exemple (listes chaînées)

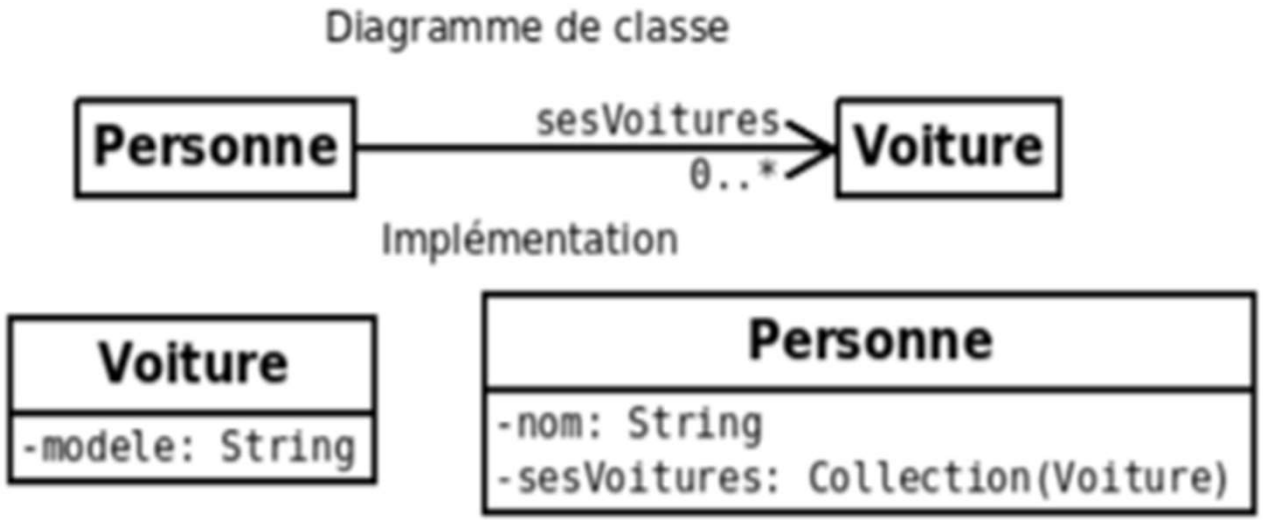


La Relation de navigabilité

Navigabilité: Selon la navigabilité nous spécifions deux types d'associations:

1. Association Unidirectionnelle :On restreint la navigabilité d'une association à un seul sens à l'aide d'une flèche.

Exemple: On peut accéder à ses voitures à partir d'une personne ; pas à ses propriétaires à partir d'une voiture



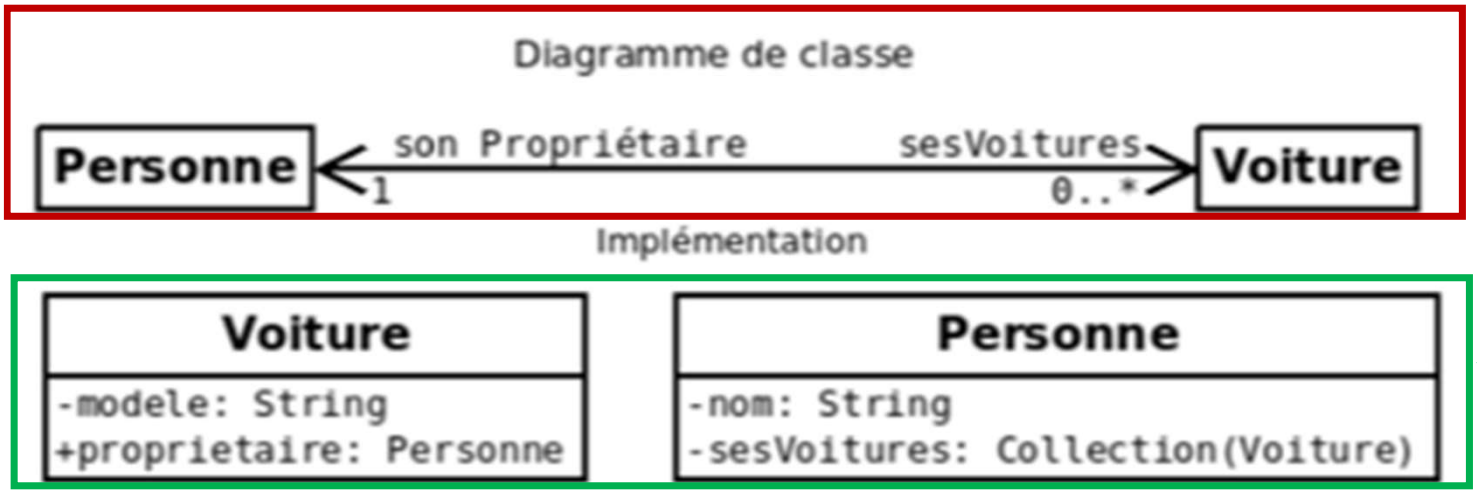
La Relation de navigabilité

Navigabilité:

2. **Association bidirectionnelle** : la navigabilité est faite dans les deux sens.

Exemple:

Diagramme de classe



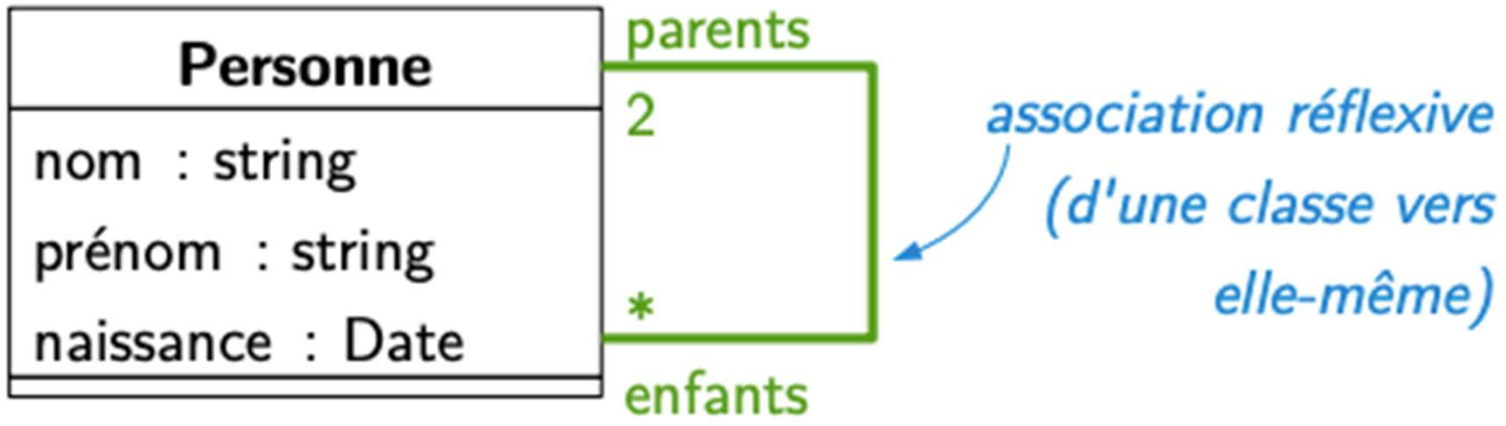
L'implémentation de diagramme de classes et l'interprétation des associations en passant au codage

La Relation d'Association

Association réflexive:

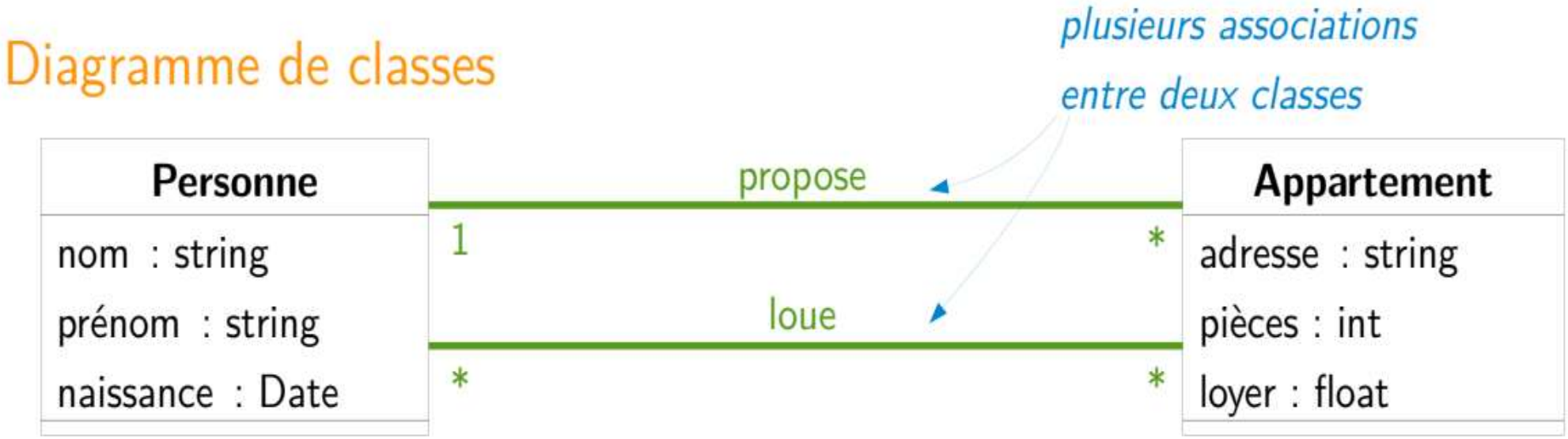
Une classe peut être associée à elle-même. Une classe peut contenir une référence à des objets de même classe.

- **Exemple**



La Relation d'Association

Association multiple: les classes ont plusieurs relations distinctes entre elles.

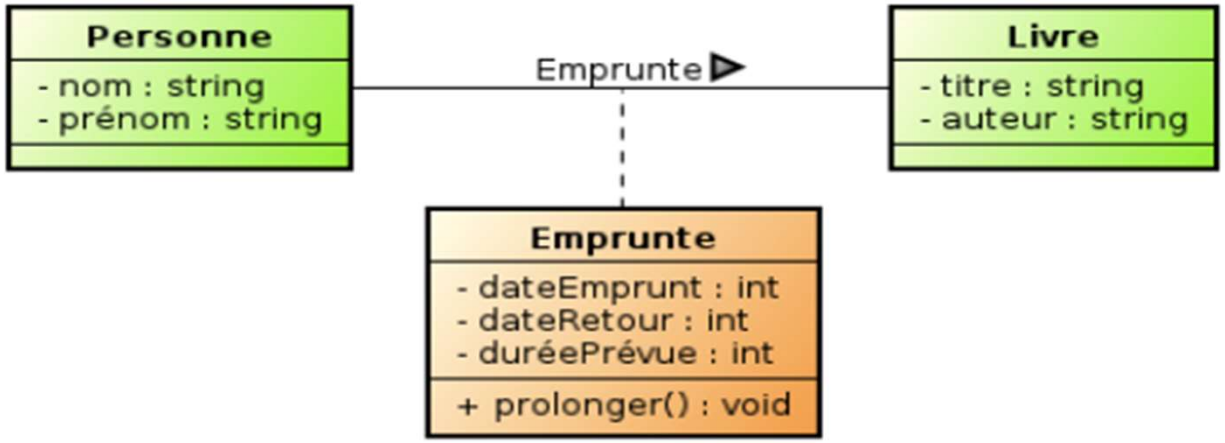


La Relation d'Association

Classes- Association:

Une association peut apporter de nouvelles informations (***attributs*** et ***méthodes***) qui n'appartiennent à aucune des deux classes qu'elle relie et qui sont spécifiques à l'association. Ces nouvelles informations peuvent être représentées par une nouvelle classe attachée à l'association via un trait en pointillés.

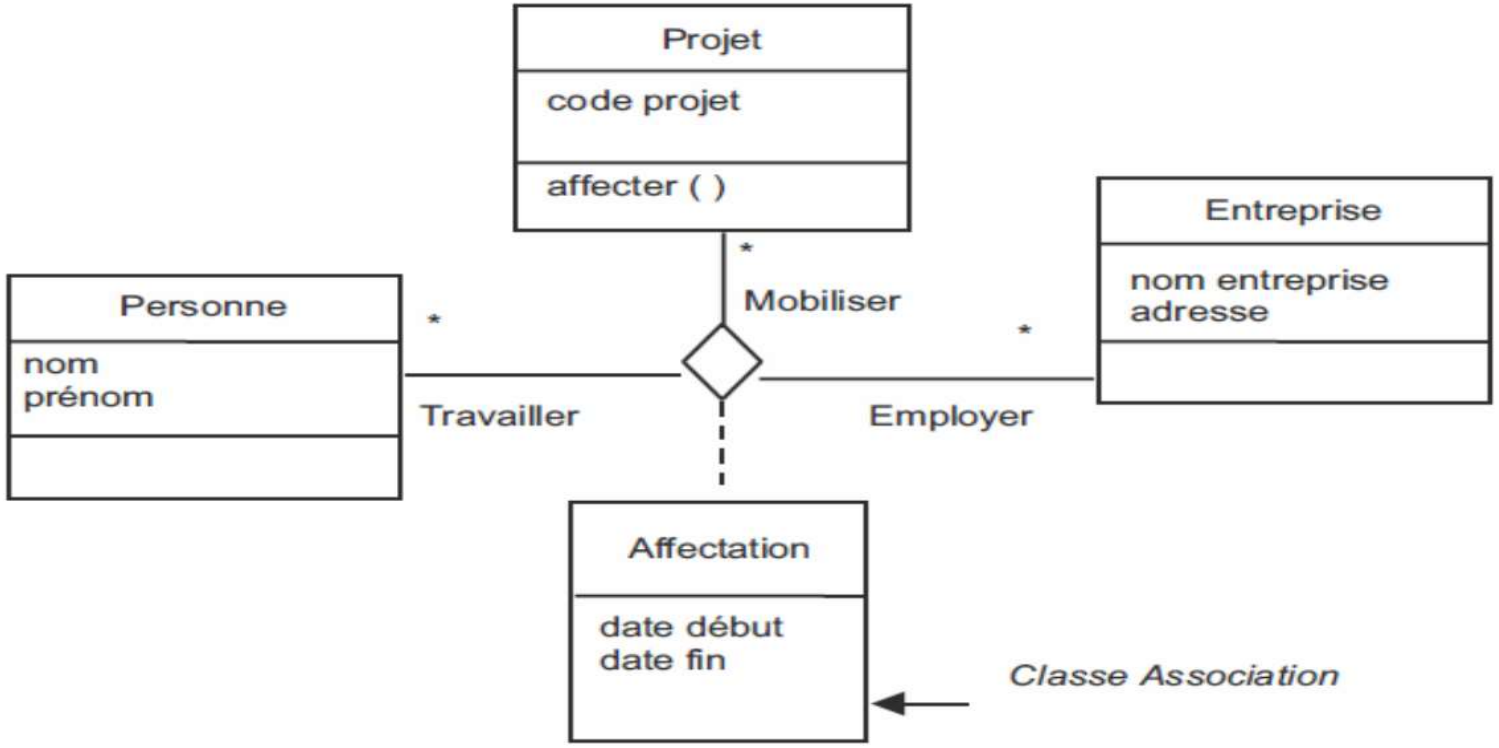
- **Exemple**



La Relation d'Association

ASSOCIATION N-AIRE ET CLASSE-ASSOCIATION

Exemple:

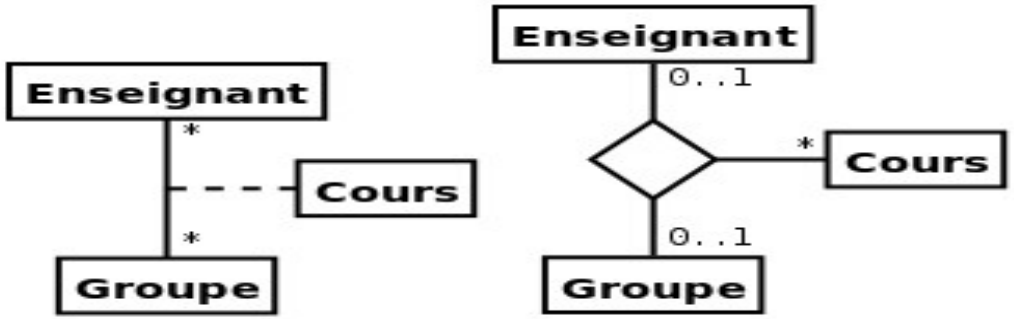


La Relation d'Association

Association n-aire :

Une association n-aire est une association entre 3 classes ou plus.

- **Exemple**



Dans le diagramme de gauche de la figure, un cours ne peut exister que s'il existe un lien entre un objet Enseignant et un objet Groupe. Quand le lien est rompu (effacé), le cours l'est également. Si un cours doit pouvoir exister indépendamment de l'existence d'un lien il faut opter pour une association ternaire (modèle de droite dans figure).

Questions d'application

Dessiner les diagrammes de classe correspondant aux situations suivantes :

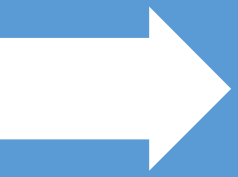
1. Tout écrivain a écrit au moins une œuvre ;

2. Les personnes peuvent être associées à des universités en tant qu'étudiants aussi bien qu'en tant que professeurs ;

Solution

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 ;



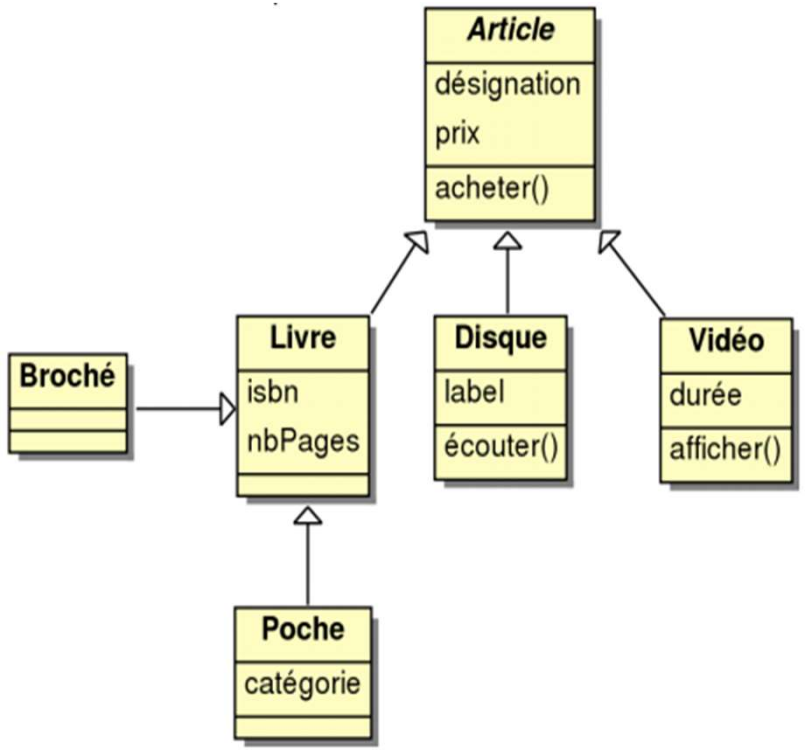
Héritage

Héritage

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

- **Exemple:**

Par héritage d'Article, un livre a d'office un prix, une désignation et une opération acheter(), sans qu'il soit nécessaire de le préciser.

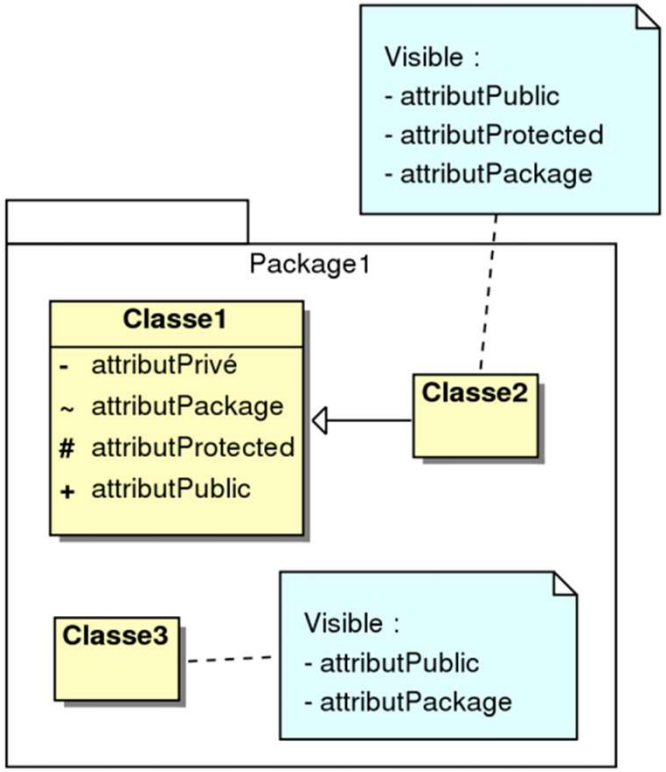


Héritage

La classe enfant possède toutes les propriétés de ses classes parents (attributs et opérations):

- La **classe enfant** est la **classe spécialisée**
- La **classe parent** est la **classe générale**

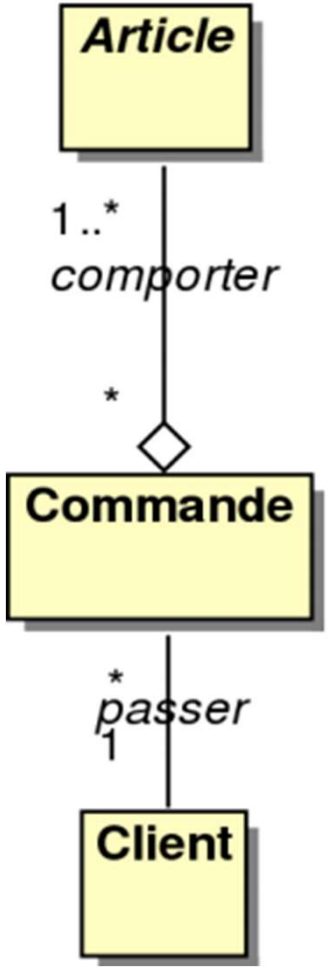
Toutefois, elle n'a pas accès aux propriétés privées.



Agrégation et Composition

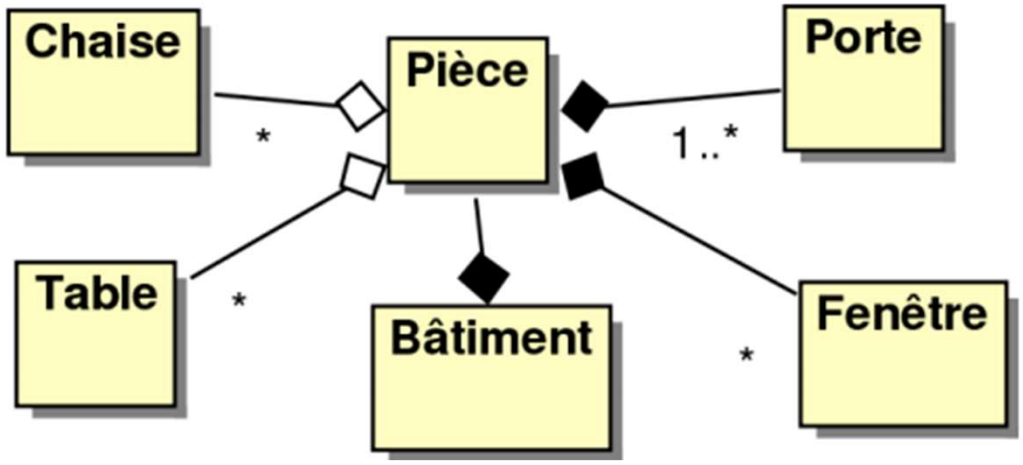
Agrégation

- 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.



Composition

- La relation de composition décrit une contenance structurelle entre instances. On utilise 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.



Agrégation ou Composition ?

Pour décider de mettre une composition plutôt qu'une agrégation, on doit se poser les questions suivantes :

1. Est-ce que la destruction de l'objet composite (du tout) implique nécessairement la destruction des objets composants (les parties) ? C'est le cas si les composants n'ont pas d'autonomie vis-à-vis des composites.

2. Lorsque l'on copie le composite, doit-on aussi copier les composants, ou est-ce qu'on peut les réutiliser , auquel cas un composant peut faire partie de plusieurs composites ?

=> Si on répond par l'affirmative à ces deux questions, on doit utiliser une composition

Questions d'application

Dessiner les diagrammes de classe correspondant aux situations suivantes :

3. Les cinémas sont composés de plusieurs salles qui projettent des films à une heure déterminée ;

4. Tous les jours, le facteur distribue des recommandés dans uns zone géographique qui lui est affectée. Les habitants sont aussi associés à une zone géographique. Les recommandés sont de deux sortes : lettres ou colis. Comme plusieurs facteurs peuvent intervenir sur la même zone, on souhaite, pour chaque recommandé, le facteur qui l'a distribué, en plus du destinataire.

Solution

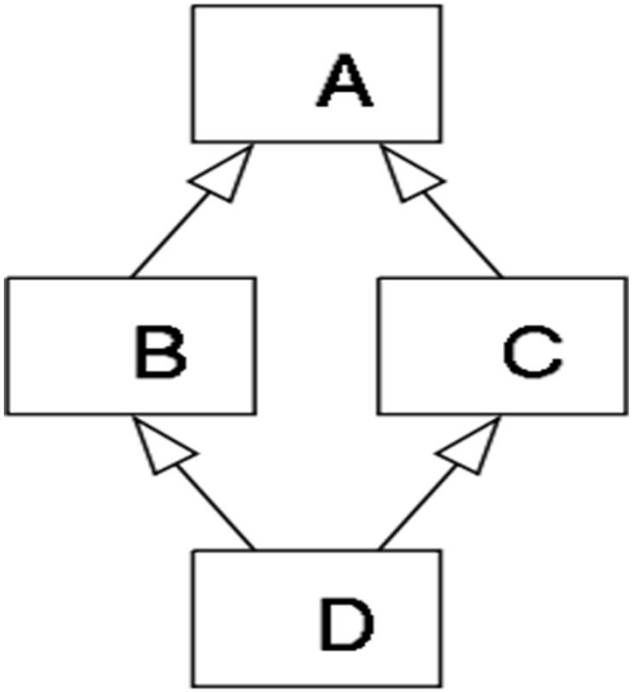
3. Les cinémas sont composés de plusieurs salles qui projettent des films à une heure déterminée ;

4. Tous les jours, le facteur distribue des recommandés dans une zone géographique qui lui est affectée. Les habitants sont aussi associés à une zone géographique. Les recommandés sont de deux sortes : lettres ou colis. Comme plusieurs facteurs peuvent intervenir sur la même zone, on souhaite, pour chaque recommandé, le facteur qui l'a distribué, en plus du destinataire.

Héritage multiple

Une classe peut avoir plusieurs classes parents. On parle alors d'héritage multiple.

- Le langage C++ est un des langages objet permettant son implantation effective.
- Java ne le permet pas.



Polymorphisme et Héritage

Définition :

- *Poly* : plusieurs
- *Morphisme* : Forme

Faculté d'une méthode à pouvoir s'appliquer à des objets de classes différentes

Capacité d'une classe à redéfinir une méthode héritée à partir d'une classe mère

- Surcharge

Avantages

- Lisibilité du code
- Généricité du code

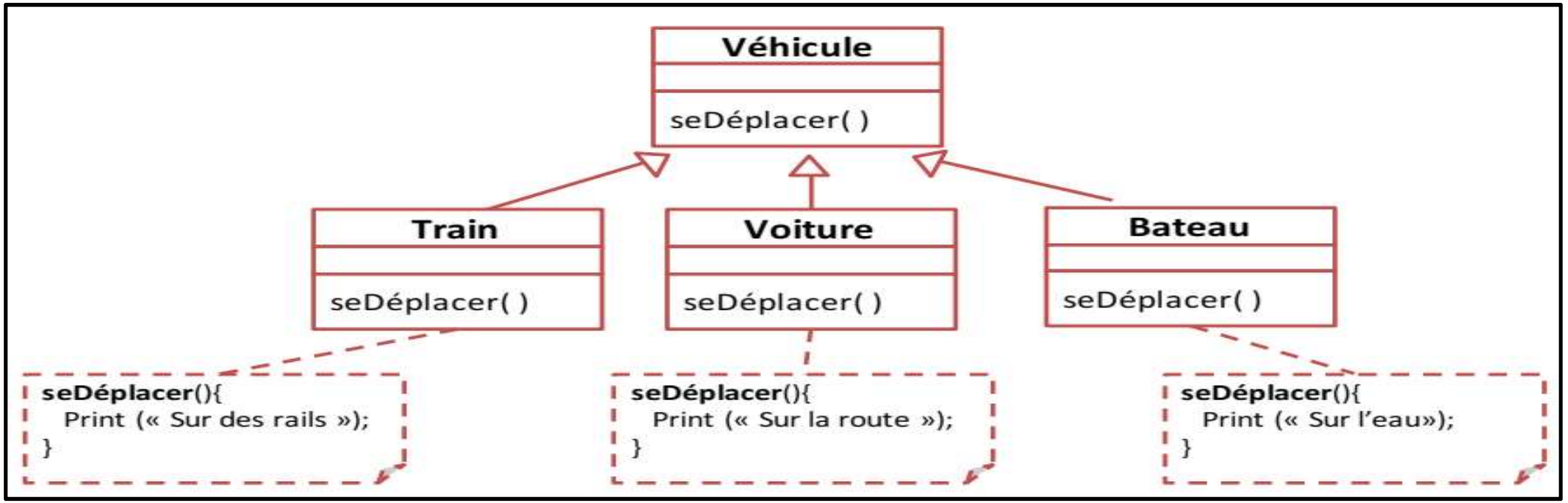
Polymorphisme et Héritage

Définition:

Poly: Plusieurs

Morphisme: Forme

Le polymorphisme est la faculté de désigner un objet comme étant une instance de plusieurs classes différentes et donc d'effectuer un traitement différent pour l'appel d'un même symbole.



Classes Concrètes et Classes Abstraites

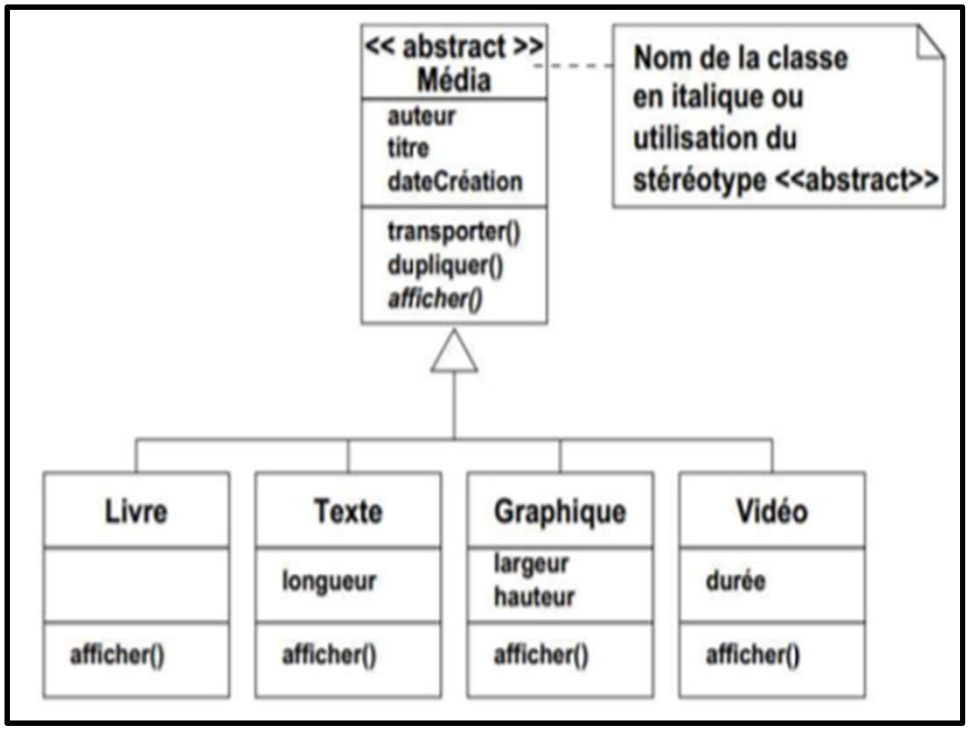
- Une classe **concrète** possède des instances. Elle constitue un modèle complet d'objet: tous les attributs et méthodes sont complètement décrits.
- Une classe **abstraite** ne peut pas posséder d'instance directe car elle ne fournit pas une description complète. Elle a pour vocation de posséder des sous classes concrètes et sert à factoriser des attributs et méthodes communs à ses sous classes.
- En UML, une classe ou une méthode abstraite sont représentées avec une mise en ***italique*** du nom de la classe ou de la méthode ou par le stéréotype **<<Abstract>>.**

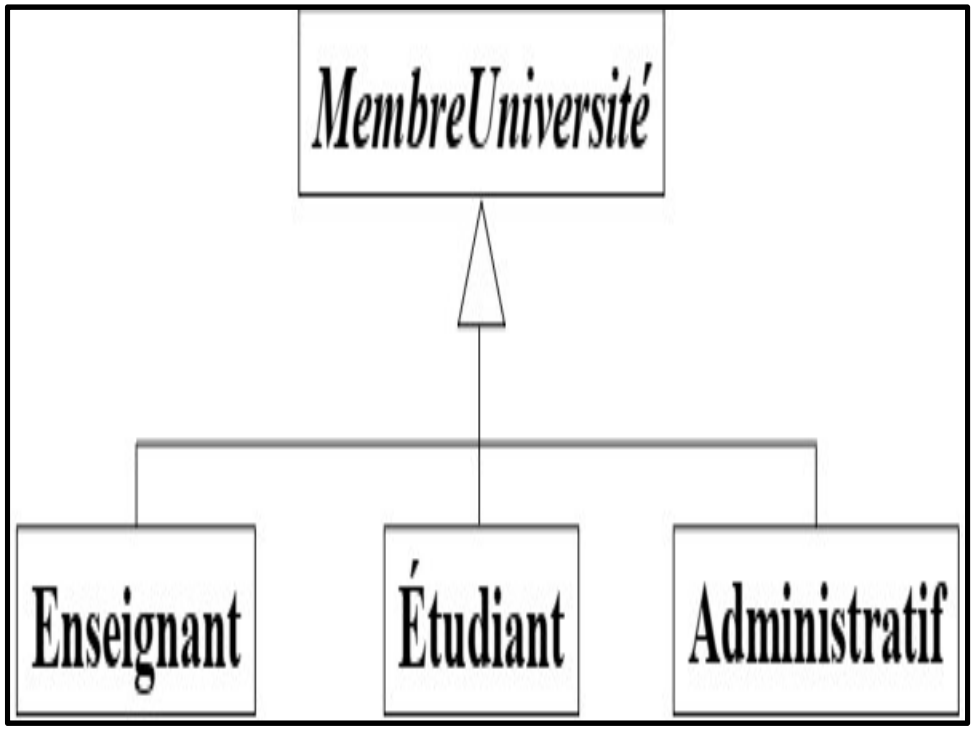
Classes Concrètes et Classes Abstraites

Il n'existe pas d'instance de la classe Média.

Un Média n'existe qu'en tant que Livre, Texte, Graphique ou Vidéo.

Un membre d'université ne peut être instancié que en tant que Enseignant, Etudiant ou Administratif

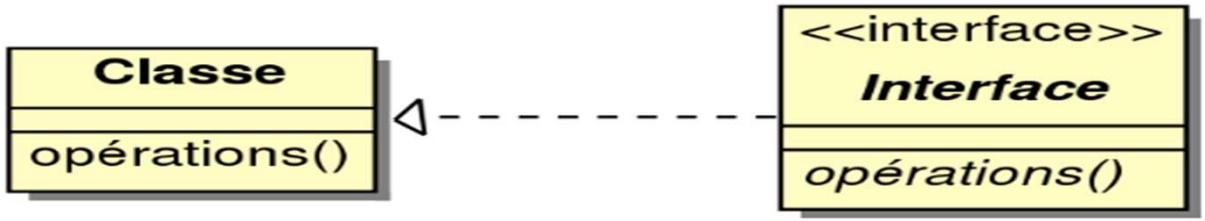




Classe Interface

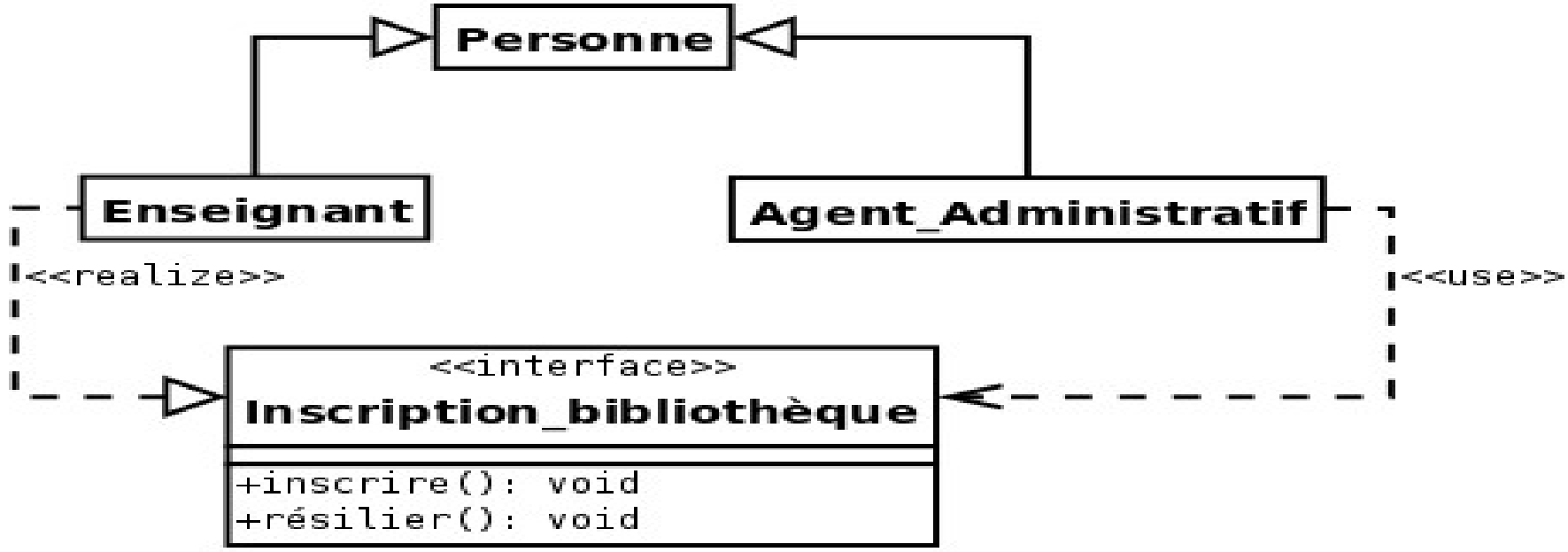
- Le rôle d'une **interface** est de regrouper un ensemble ***d'opérations (méthodes)*** assurant un service cohérent.
- L'utilisation des **interfaces** permet de contourner la **limite de l'héritage unique** et d'organiser le code de manière plus flexible en appliquant le **principe de polymorphisme**.
- Une interface est définie 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 l'implémente.



Classe Interface

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

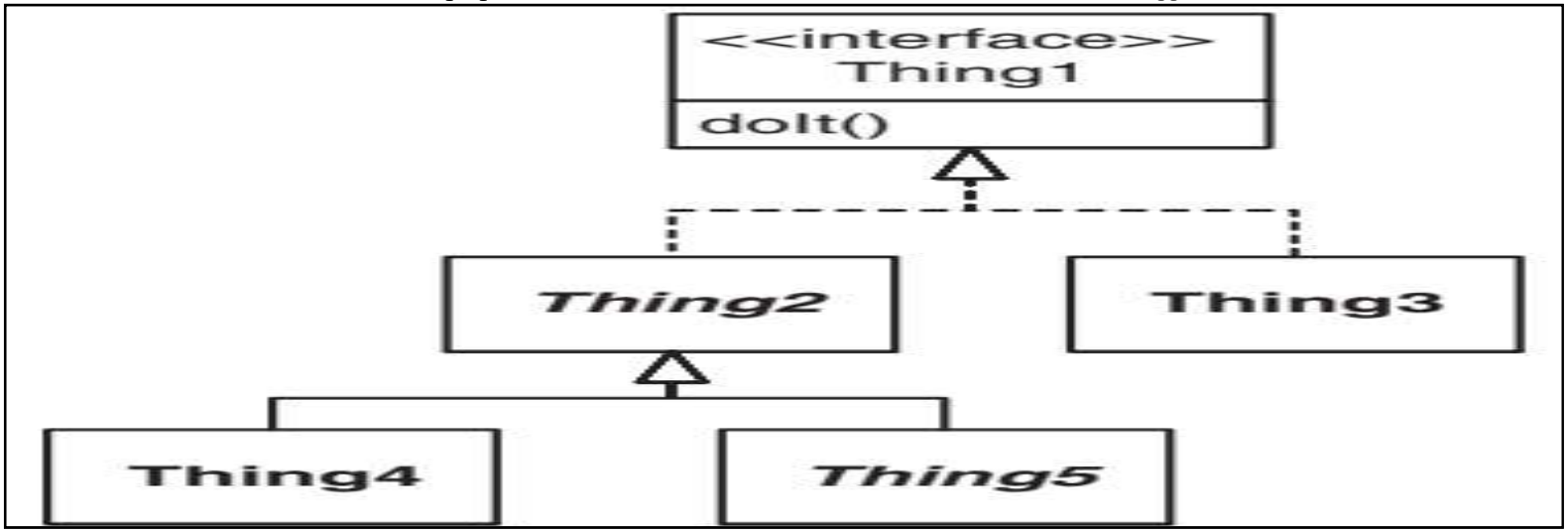


Classe Abstraite et Interface

- Les interfaces ressemblent aux classes abstraites sur un seul point : elles contiennent des membres **expliquant certains comportements sans les implémenter**.
- Les classes abstraites et les interfaces se différencient principalement par le fait qu'**une classe peut implémenter un nombre quelconque d'interfaces**, alors qu'une classe abstraite ne peut hériter que d'**une seule classe** abstraite ou non.

Interface QCM 1

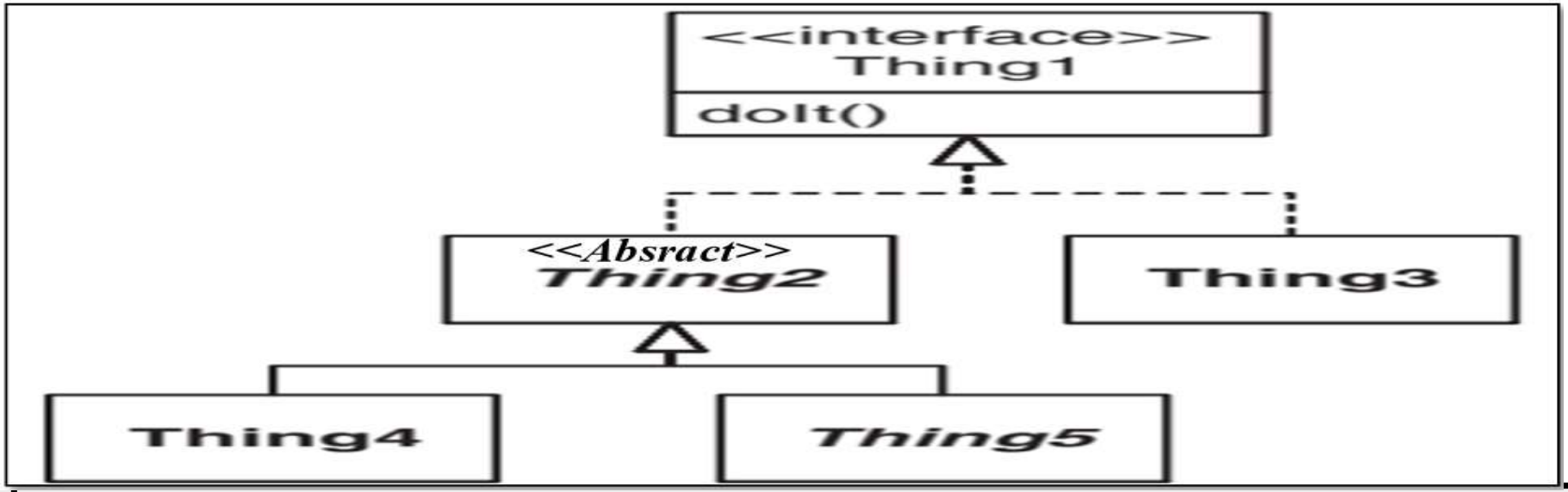
Quelle affirmation en rapport avec la méthode doIt() est vraie?



1	La méthode doIt() doit être implémentée par Thing3 et peut-être aussi par Thing4.	
2	La méthode doIt() doit seulement être implémentée par Thing5.	
3	La méthode doIt() doit être implémentée par Thing2, Thing3, Thing4 et Thing5.	
4	Aucune classe ne doit implémenter doIt() parce qu'elle est déjà implémentée par Thing1	
5	Autre réponse	

Interface QCM 2

Quelle affirmation en rapport avec la méthode dolt() est vraie?



1	La méthode dolt() doit être implémentée par Thing3 et peut-être aussi par Thing2.	
2	La méthode dolt() doit seulement être implémentée par Thing5 et Thing4.	
3	La méthode dolt() doit être implémentée par Thing2, Thing3, Thing4 et Thing5.	
4	La méthode dolt() doit être implémentée par Thing3, Thing4 et Thing5.	
5	Autre réponse	

Chap.3: Contraintes d'association

Une contrainte constitue une condition ou une restriction sémantique exprimée sous forme d'instruction dans un langage textuel qui peut être naturel ou formel.

En effet, UML permet d'associer une contrainte à un, ou plusieurs, élément(s) de modèle de différentes façons :

• En plaçant directement la contrainte à côté d'une propriété ou d'une opération dans une classe.

• En ajoutant une note associée à l'élément à contraindre ;

• En plaçant la contrainte à proximité de l'élément à contraindre, comme une extrémité d'association par exemple.

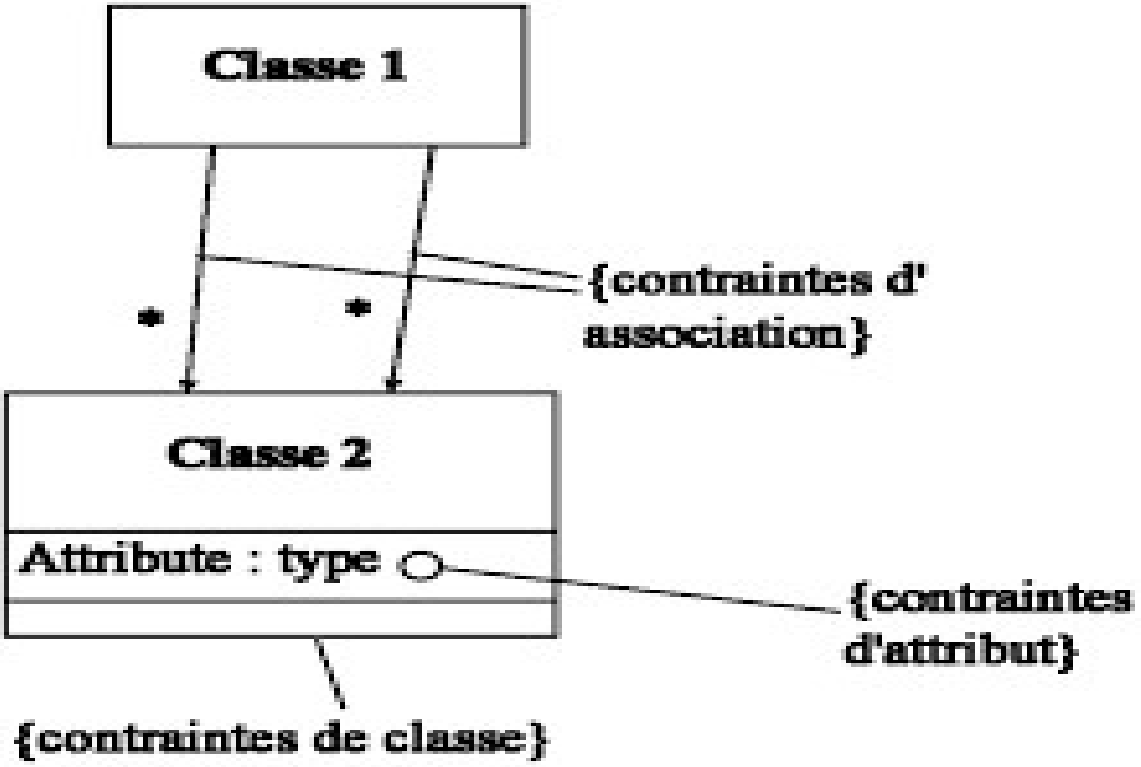
• En plaçant la contrainte sur **une flèche** en pointillés joignant les deux éléments de modèle à contraindre ensemble, la direction de la flèche constituant une information pertinente au sein de la contrainte ;

• En plaçant la contrainte sur **un trait** en pointillés joignant les deux éléments de modèle à contraindre ensemble dans le cas où la contrainte est **bijective**.

• En utilisant une note reliée, par des traits en pointillés, à chacun des éléments de modèle, subissant la contrainte commune, quand cette contrainte s'applique sur plus de deux éléments de modèle.

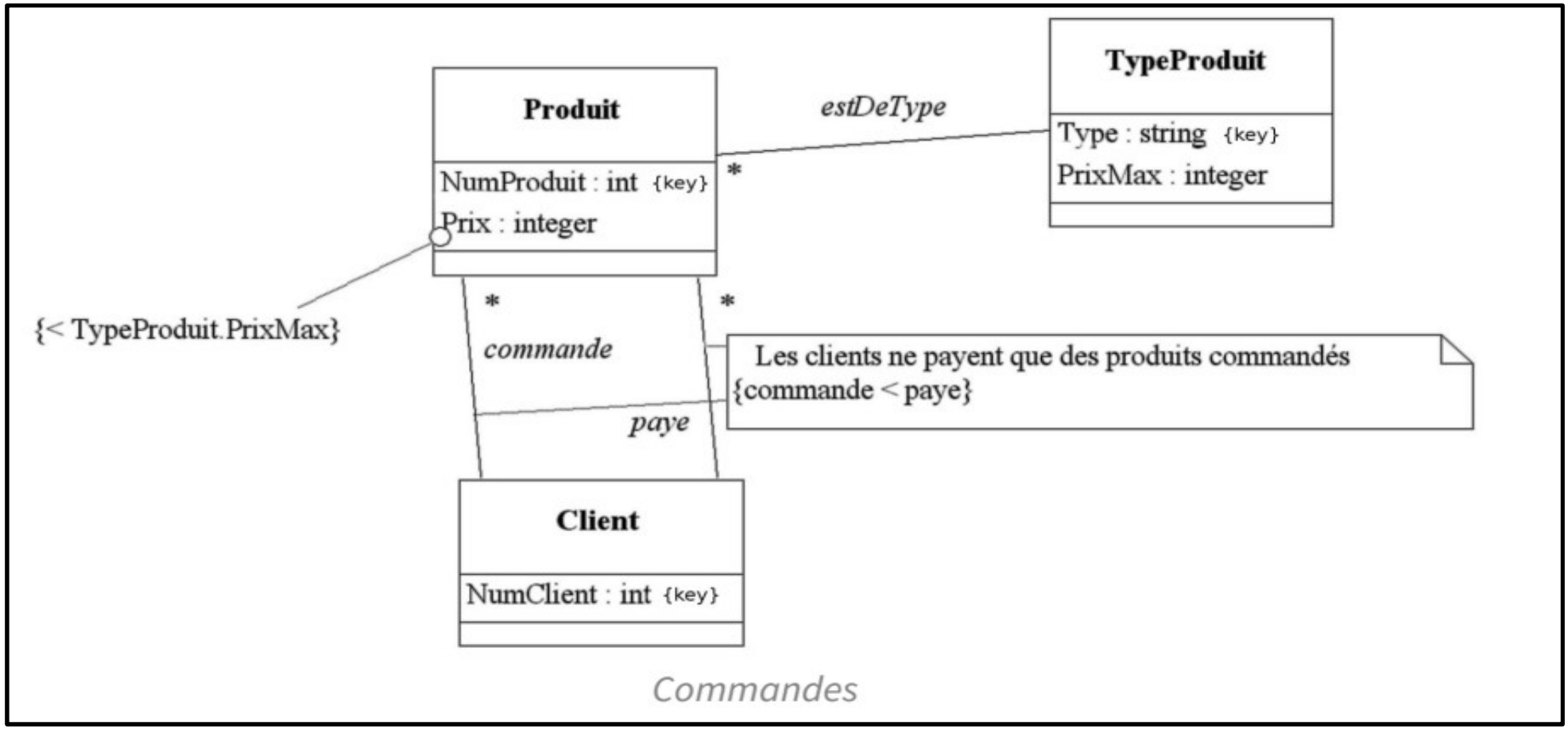
Rappel : toutes les contraintes en UML s'expriment **entre accolades {}**.

Expression des contraintes



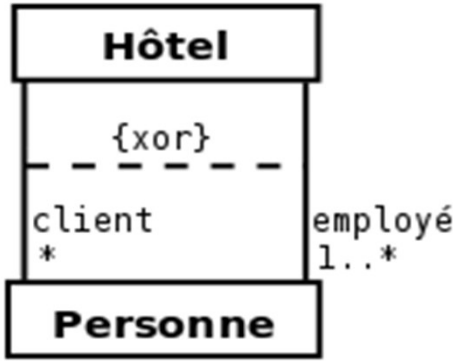
Expression des contraintes

- **Exemple:**

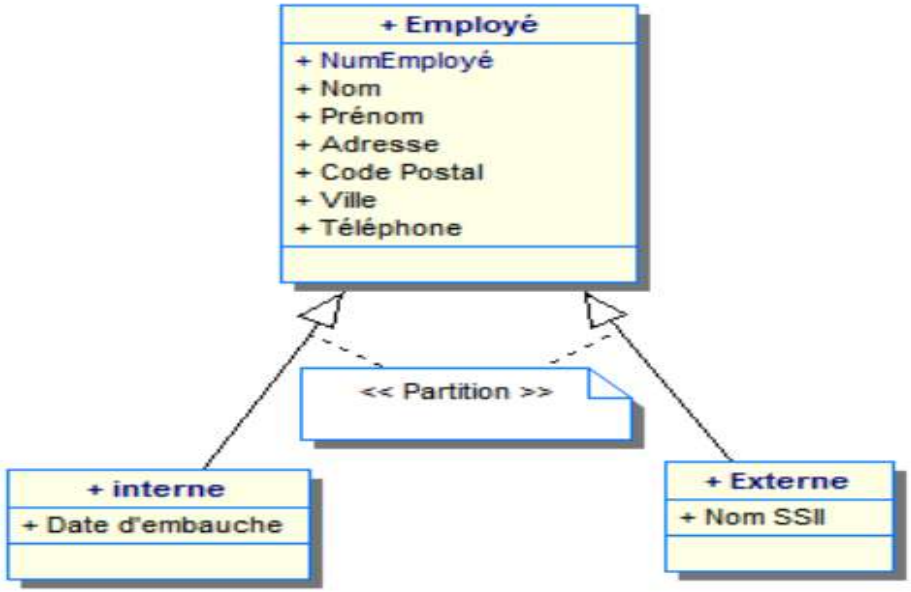


Chap.3: Contraintes d'association

Partition: {XT},{+}, {xor} ou {ou-exclusif}, {P} : indique qu'une instance ne peut appartenir en même temps à la réalisation de deux associations, Exactement une des deux associations doit être instanciée.



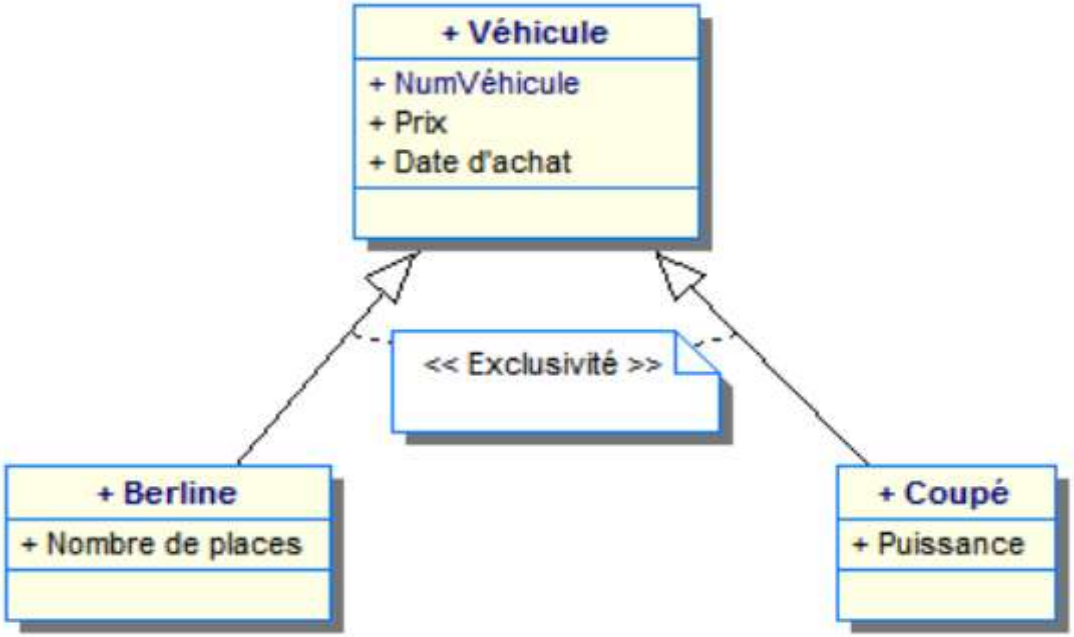
Une personne est soit un employé ou client d'un hôtel et non pas les deux à la fois, il n'existe pas d'autre type d'association.



Un employé est soit un employé interne, soit un employé externe, mais pas les deux, il n'existe pas d'autre type d'employé.

Chap.3: Contraintes d'association

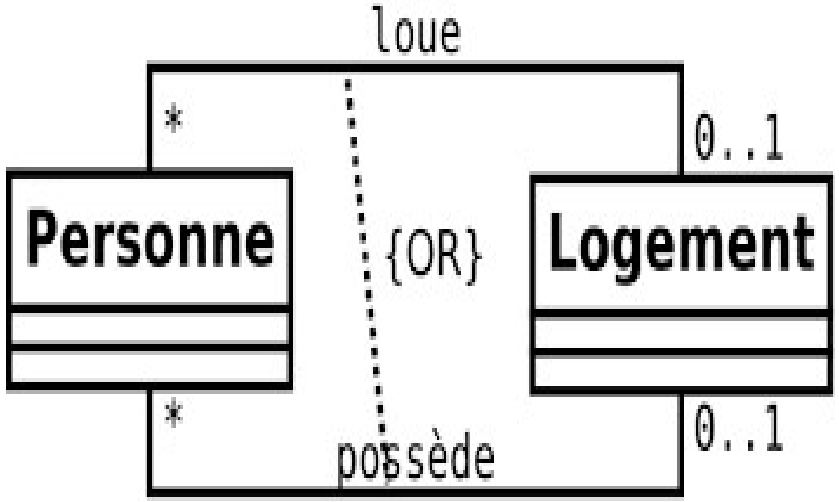
Exclusion {X} Les deux associations ne peuvent être instanciées en même temps.



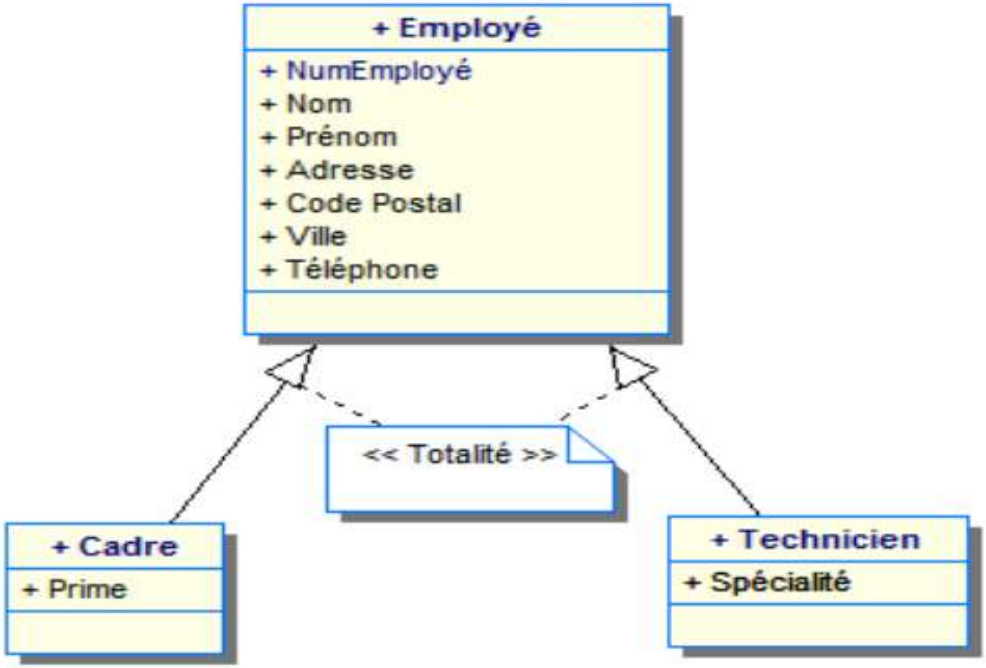
Un véhicule est soit une berline, soit un coupé mais pas les deux et il peut exister d'autres véhicules.

Chap.3: Contraintes d'association

Totalité {T}, également notée **{OR}** : Au moins une des deux associations doit être instanciée.



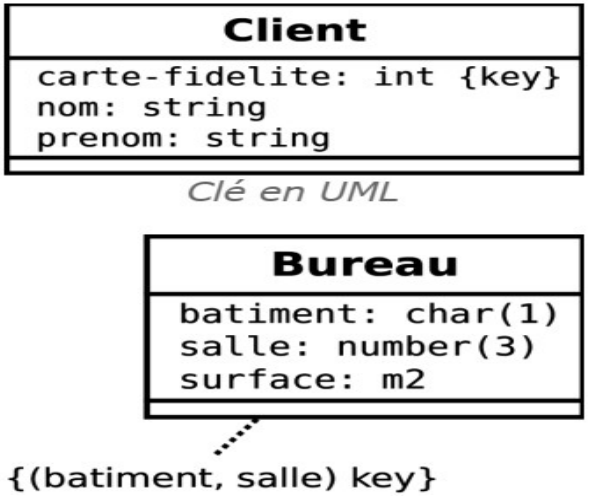
Une personne peut à la fois louer et posséder un logement, mais au moins l'un ou l'autre .

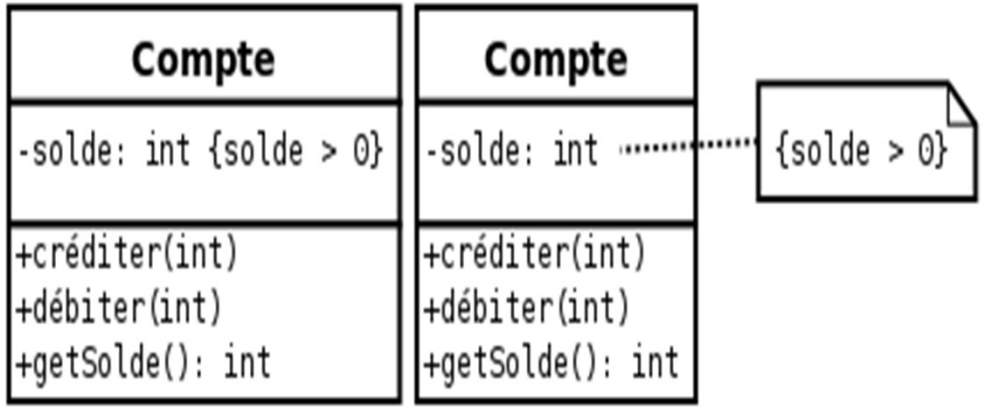


Un employé peut être à la fois cadre et technicien. Il est au moins l'un ou l'autre.

Chap.3: Contraintes d'association

- ***{unique}:*** La valeur de l'attribut est unique pour la classe.
- ***{frozen}:*** Une fois instancié l'attribut ne peut plus changer.
- ***{key}:*** Bien que non standardisé en UML, en base de données, il est courant d'utiliser {key}.

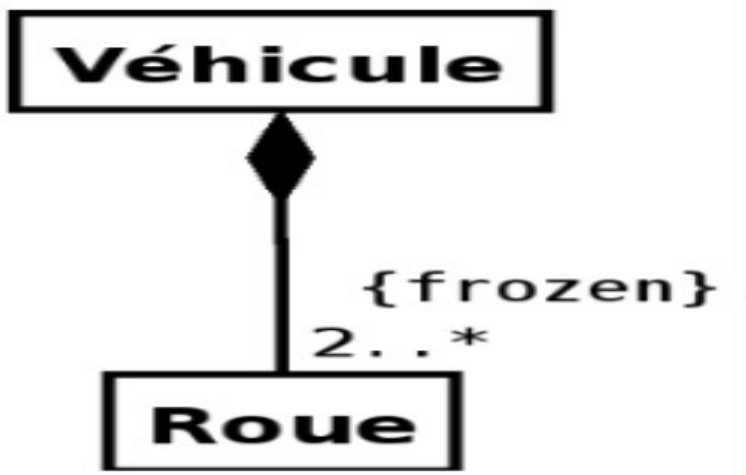




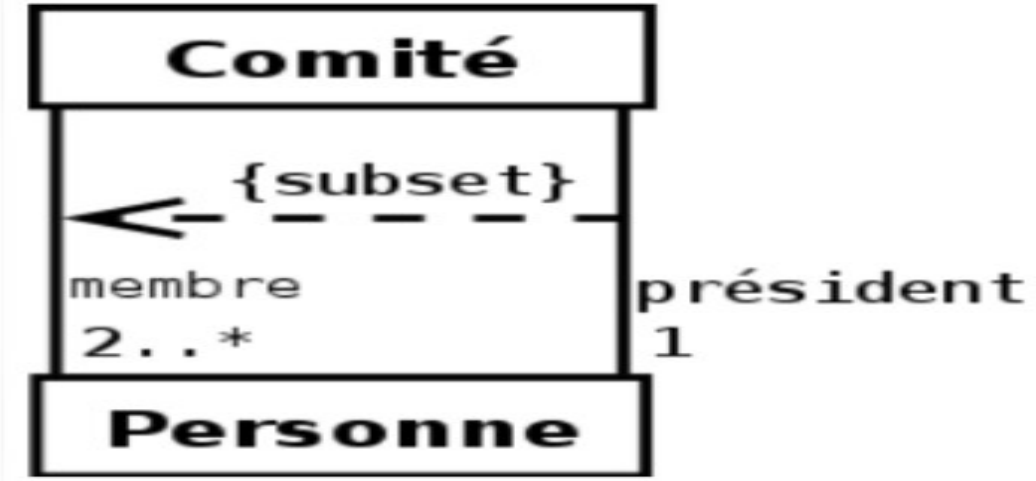
la contrainte porte sur un attribut qui doit être positif.

Chap.3: Contraintes d'association

- **Inclusion: {subset} /{IN}/{I}:** indiquant qu'une occurrence fait partie obligatoirement partie des occurrences d'une autre association ;
- **{frozen}/{immuable} :** Un lien ne peut plus être modifié ni détruit après sa création ;



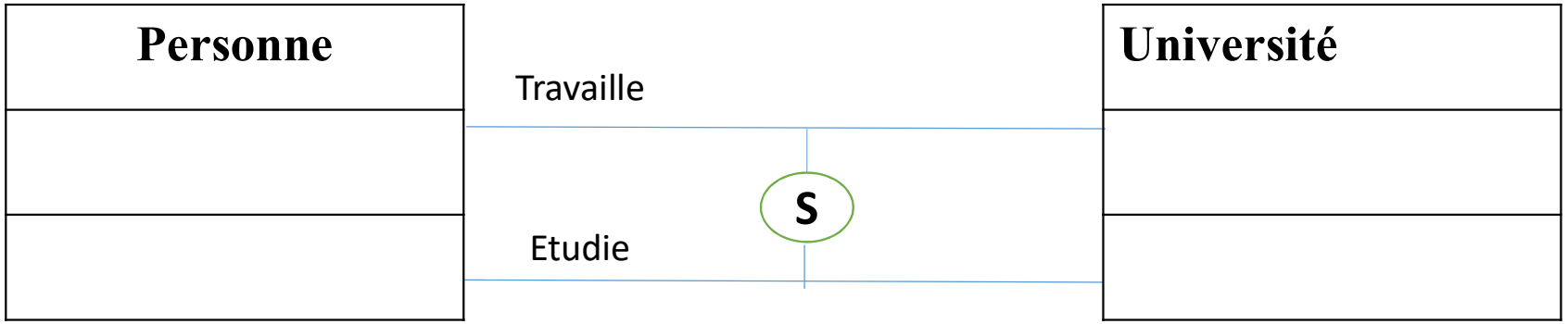
Une fois un véhicule est fabriqué, le nombre des roues est fixé, on ne peut pas ni le changer ni les détruire



un Comité possède au moins 2 personnes (membres), et un président qui fait nécessairement partie du Comité.

Chap.3: Contraintes d'association

- **Simultanéité {S}, également notée {=} ou {AND}** :Si une association est instanciée, l'autre doit l'être aussi. La simultanéité est équivalente à une double inclusion.



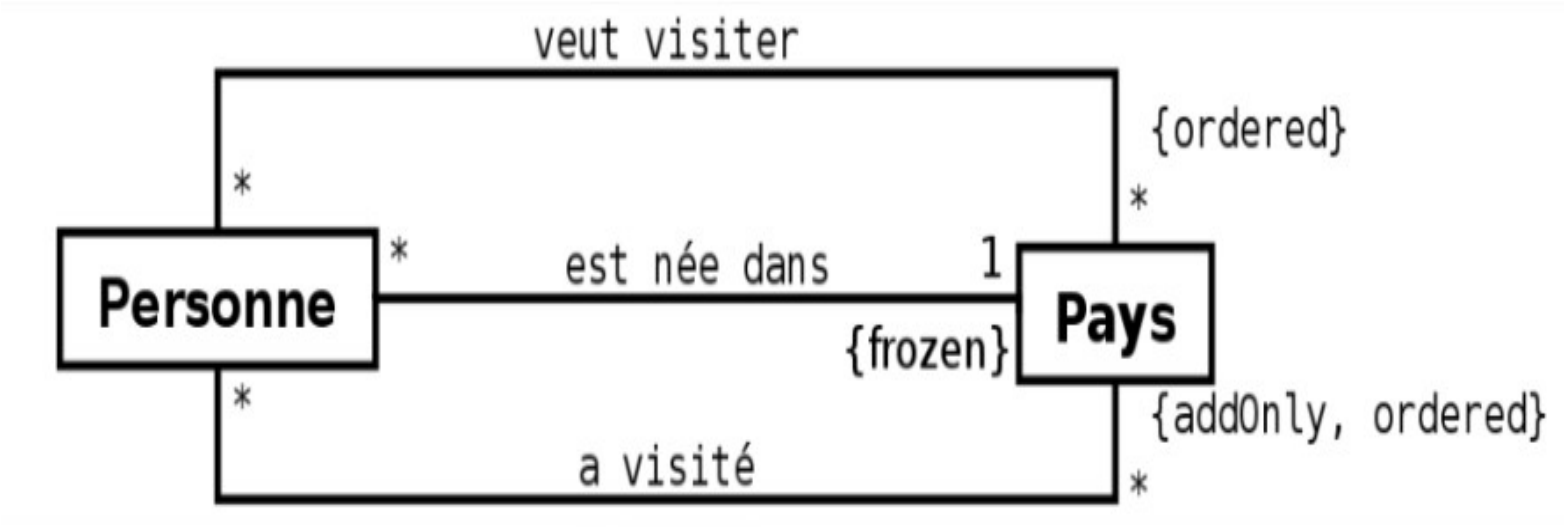
Une personne qui travaille dans une université est obligatoirement un étudiant et vice versa

Chap.3: Contraintes d'association

- **{ordered} ou {ordonné} :** Les objets doivent être ordonnés.

Rq : on ne sait pas *comment* cela sera ordonné (c'est un choix de conception) ;

- **{addOnly} :** On ne peut qu'ajouter un objet, pas le détruire ;



Une Personne veut visiter ou pas des pays (ordre traduisant sa préférence) ; il est né dans un pays (qui ne changera jamais) ; il a visité un certain nombre de pays (peut-etre aucun), on ne peut qu'en ajouter, et ces pays visités sont ordonnés (selon sans doute leur date de visite).

Exercices: Dessiner diagramme de classes de chaque situation ci-dessous

1°) Un médicament a un nom unique, et peut être soit proscrit ou conseillé pour une pathologie, s'il est conseille, i faut indiquer la posologie.

2°) Une personne a un nom, prénom, lieu et date de naissance (date), ces deux derniers attributs sont invariables.

3°) Dans une entreprise de transport, les chauffeurs (caractérisés par un N° de permis, un nom et un prénom) se classent en deux catégories : les chauffeurs « Transport International Routier (TIR) et les chauffeurs « transport national » (TN), un chauffeur TIR peut assurer le transport national.

4°) Un client peut être soit un prospect soit un client en portefeuille s'il a déjà passé au moins une commande.

5°) Une personne peut être un membre de jury d'une soutenance ou un président, ce dernier est obligatoirement fait partie des membres de jury.

6°) Les types de produit (CodeTypeProduit, LibTypeProduit) sont commercialisés sur un ou plusieurs secteurs (CodeSecteur, NomRégion). Les représentants (NumRep, NomRep) sont responsables d'un ou plusieurs types de produit sur un ou plusieurs secteurs. Un représentant ne peut être responsable que si le produit est commercialisé sur son secteur.

7°) Même question que 6, sauf qu'on considère cette fois qu'un produit commercialisé sur un secteur a obligatoirement un responsable.

Chap. 4: Diagramme d'objet

- Un diagramme d'objets représente des objets et leurs liens pour donner une vue de l'état du système à un instant donné de l'exécution.

- **Exemple**

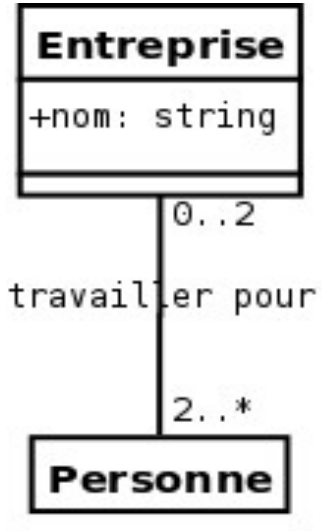


Diagramme de classes

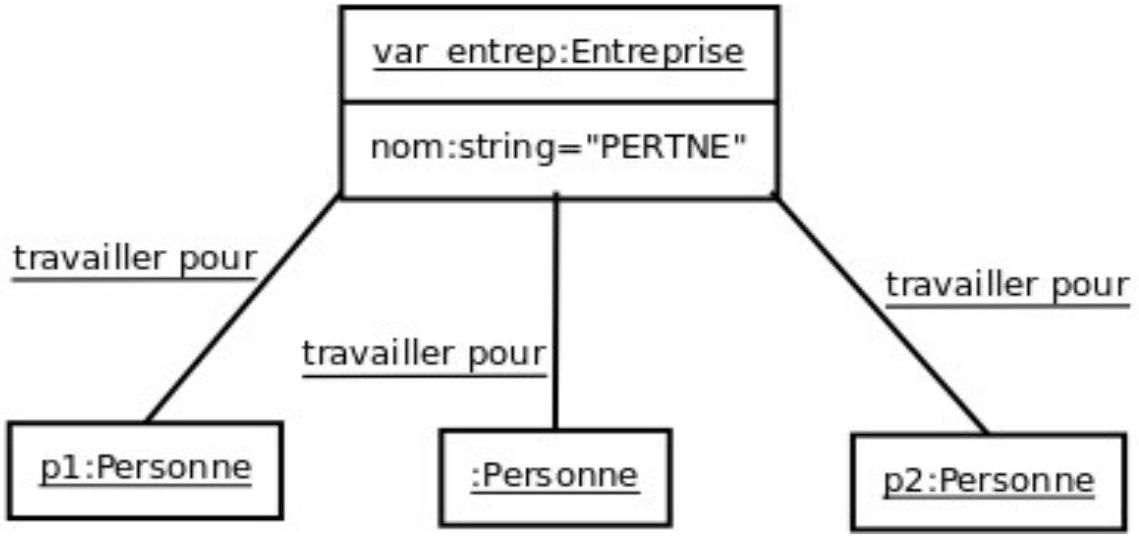


Diagramme d'objets

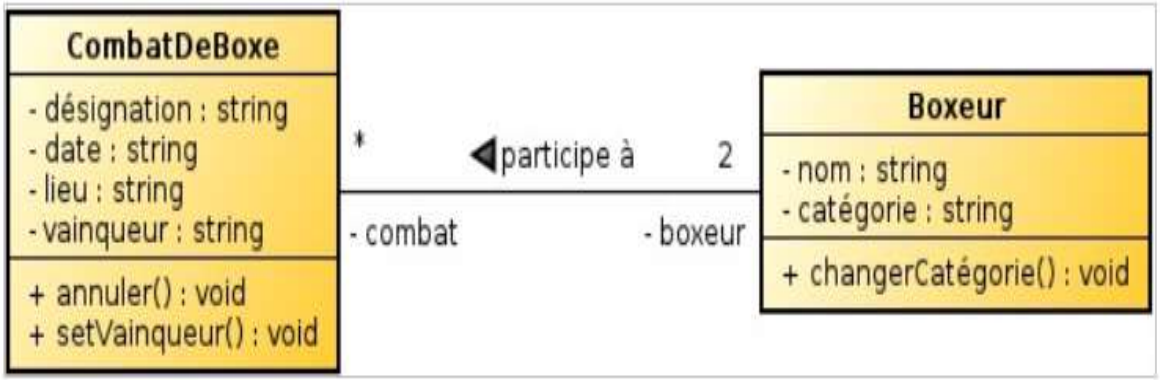
Chap. 4: Diagramme d'objet

- Les **attributs** **prennent** des valeurs lorsque la classe est **instanciée** : ils sont en quelque sorte des « variables » attachées aux objets.
- Une **méthode** est un service offert par la classe (un traitement que les objets correspondant peuvent effectuer).

Chap. 4: Diagramme d'objet

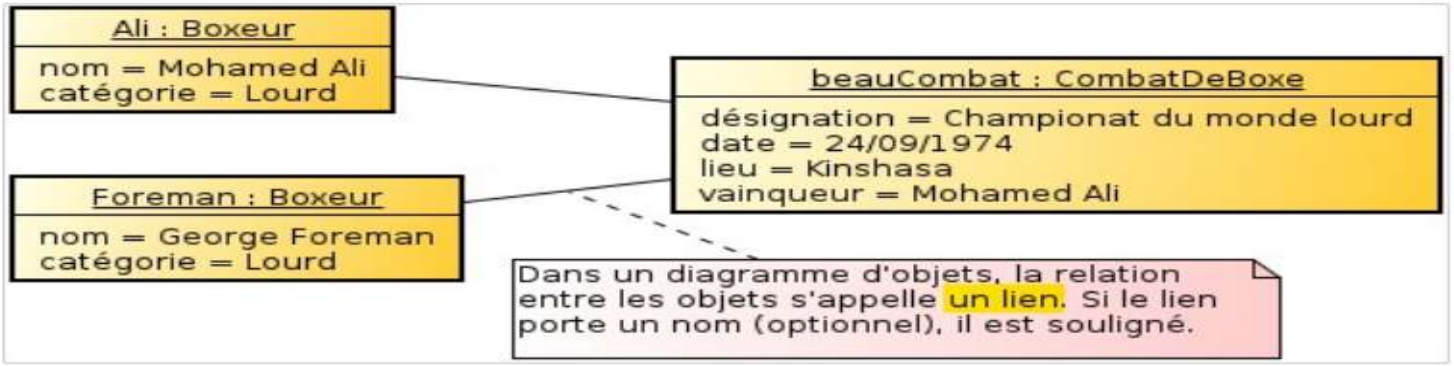
- **Exemple:**

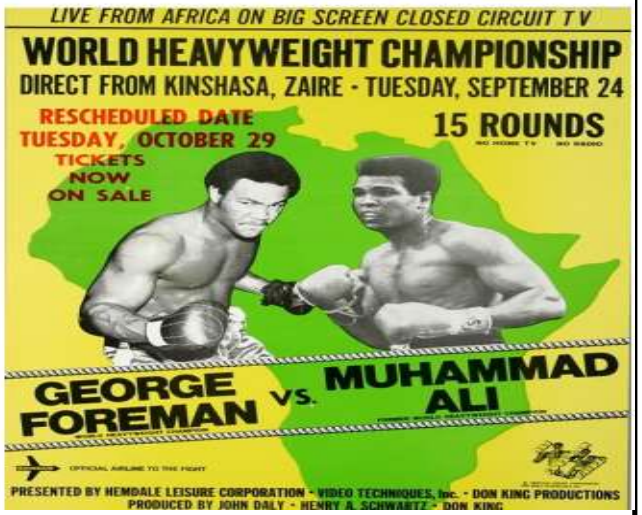
Soit le diagramme de classe suivant :



- **Diagramme d'objet correspondant :**

Avec le diagramme objet ci-dessous on définit une instance particulière du diagramme de classe (qui est le combat entre ***Mohamed Ali*** et ***George Foreman*** qui a eu lieu le 24 septembre 1974 à Kinshasa).





Chap.4: Exercice d'application

Une start-up de développeurs travaille pour une entreprise dans un cadre de sous-traitance. Cette start-up possède un identifiant et un nom. Chaque développeur est caractérisé par son numéro de carte d'identité, son nom, son prénom, et son adresse email. Il utilise un ordinateur portable qui lui est personnel. Un développeur peut être un programmeur spécialisé dans un langage de programmation, ou un concepteur expert dans une méthode ou un langage de modélisation. les concepteurs créent les modèles de conception et les programmeurs écrivent le code.

1. Créer le diagramme de classe correspondant à la description faite en dessus.
2. Créer le diagramme d'objets correspondant au texte suivant : Ali et Mostapha sont des programmeurs spécialisés respectivement dans les langages Ada et Java. Mohamed est un concepteur spécialisé dans le langage de modélisation UML . Ils font tous partie du start up « DevSoft » qui travaille pour le compte de l'entreprise «AtlasTechno »

Solution Ex 2:

Solution Ex 2: