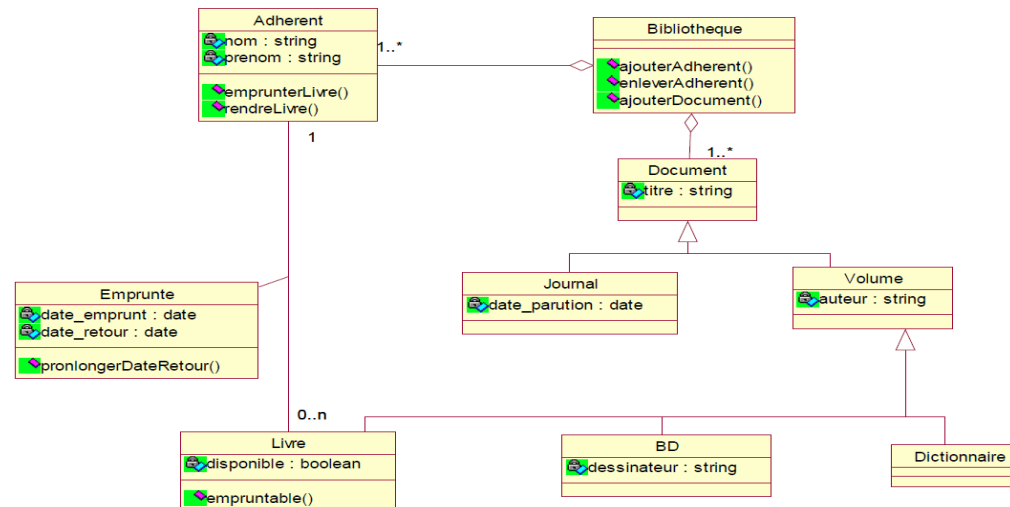


Modélisation UML

Chapitre 3

Le Diagramme de Classe



Plan

- ▶ Introduction
- ▶ Concepts de base:
 - Classe
 - Attribut
 - Méthode
 - Association
 - Classe d'association
 - Multiplicité
 - Agrégation / composition
- ▶ Concepts de l'Orienté Objet
 - Classe, objet
 - Abstraction
 - Généralisation / Spécification
 - Encapsulation
 - polymorphisme

Introduction

Introduction

- ▶ Le diagramme de classe constitue la base de la modélisation avec UML.
- ▶ Ce diagramme permet de donner la représentation statique du système à développer.
- ▶ Cette représentation est centrée sur les concepts de classes et d'association.
- ▶ Chaque classe est décrite par les données et par les traitements dont elle est responsable pour elle-même et vis-à-vis des autres classes.
- ▶ Les traitements sont matérialisés par des opérations (méthodes).

Concepts de base

Concepts de base

Classe:

- Est un ensemble d'objets ayant les mêmes propriétés ou attributs, un même comportement (opération) et une sémantique commune.

Nom Classe
attributs
méthodes

Exp:

Personne
CIN: Entier adresse: chaine
ajouter () modifier ()

Concepts de base

Attribut:

- ▶ Un attribut est une propriété élémentaire d'une classe, pour chaque objet d'une classe l'attribut prend une valeur.
- ▶ Un attribut dont la valeur peut être calculée à partir d'autres attributs de la classe est un attribut dérivé qui est noté par '*\nomattribut*'.

Nom Classe
nomAttribut: type [valeur initiale]

Exp:

Voiture
numImmatriculation: Entier marque: chaîne année: entier <i>\ancienneté: entier</i>



Concepts de base

Opération ou méthode:

- ▶ Une opération est une fonction ou un traitement appliqué aux objets d'une classe.
- ▶ Elle permet de décrire le comportement d'un objet.

Nom Classe
nomAttribut : type [valeur initiale]
nomOp([param]) [:type valeur retour]

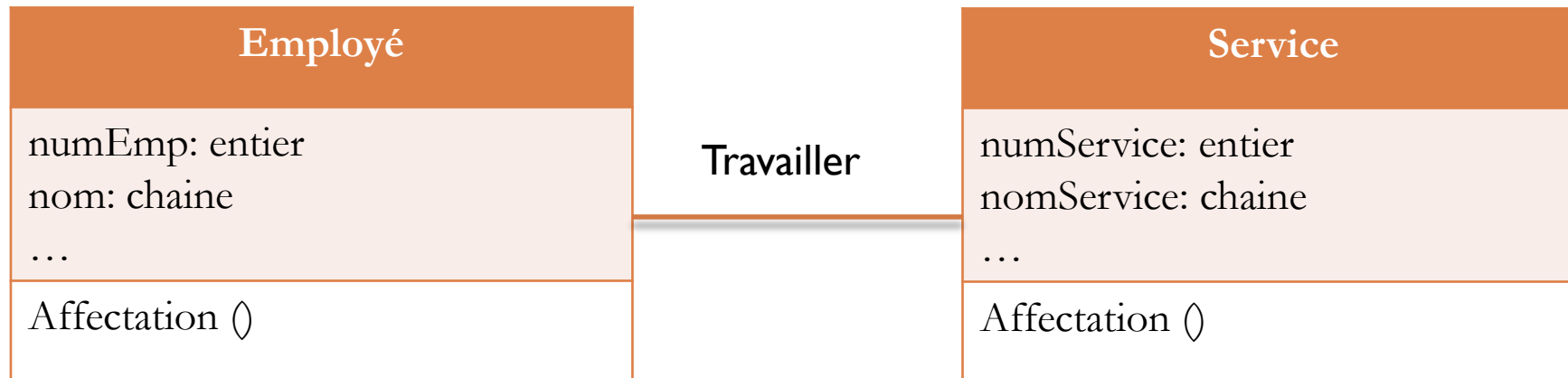
Exp:

Employé
numEmp: entier nom: chaîne ...
affectation ()

Concepts de base

Lien et association:

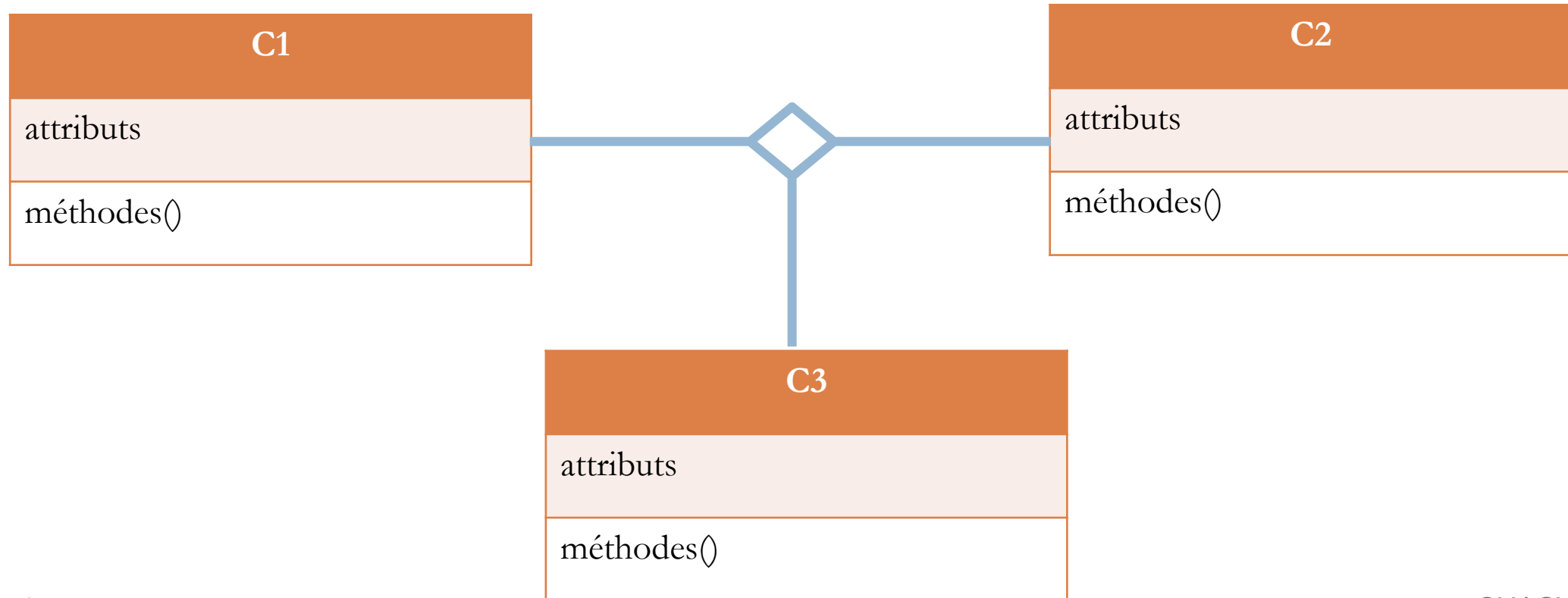
- ▶ Un lien permet d'établir une relation entre objets.
- ▶ Une association permet d'établir une relation entre classes.



Concepts de base

Lien et association:

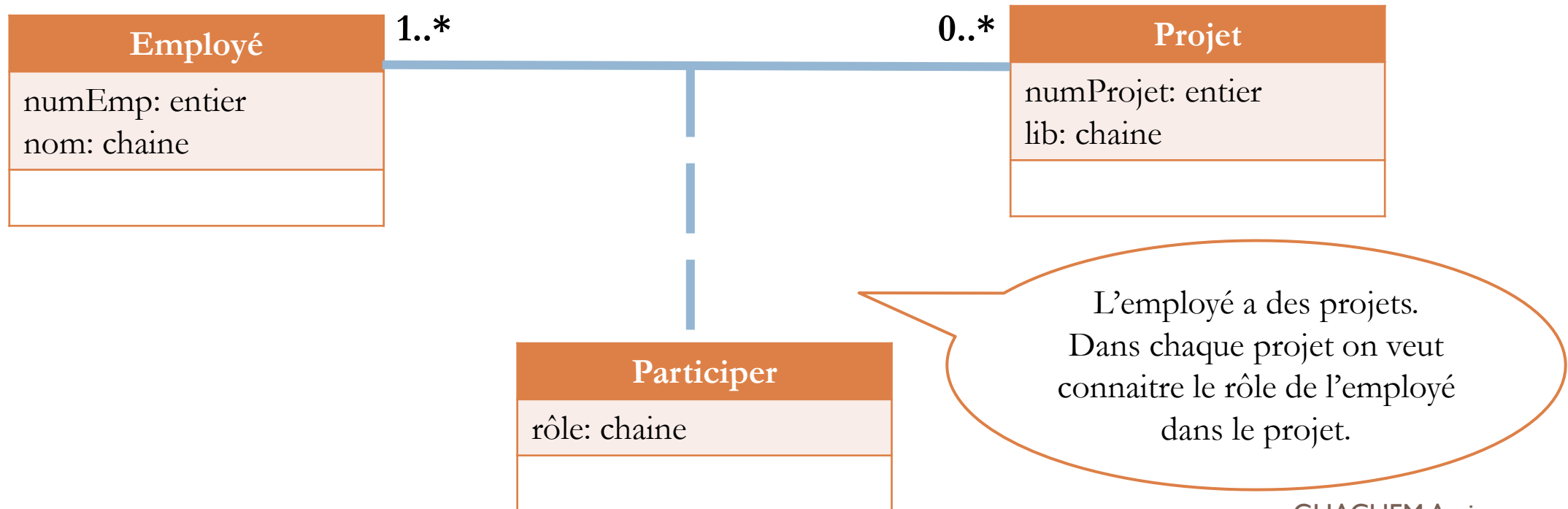
- ▶ Une association peut relier plusieurs classes.
- ▶ Dans ce cas, elle sera modélisée par le formalisme suivant:



Concepts de base

Classe d'association:

- ▶ Elle relie deux ou plusieurs classes.
- ▶ Elle est porteuse de données.



Concepts de base

Multiplicité:

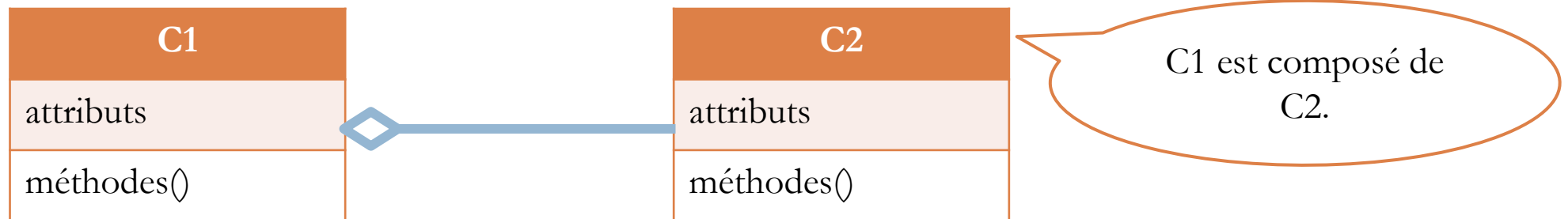
- ▶ Elle indique un domaine de valeurs pour les cardinalités entre classes et associations.
- ▶ Elle exprime le nombre minimum et maximum d'objets d'une classe qui peuvent être reliées à des objets d'une autre classe.

Cardinalité	Multiplicité
0,1	0..1
1,1	1
0,N	0..* ou *
1,N	1..*

Concepts de base

Agrégation:

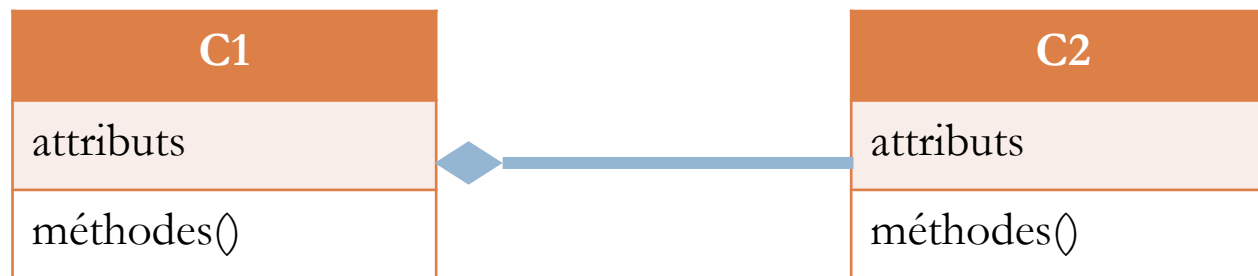
- ▶ C'est un cas particulier qui exprime une relation de contenance.
- ▶ Les agrégations n'ont pas besoin d'être nommées implicitement.
- ▶ Elle signifie « *contient* » ou « *composé de* ».



Concepts de base

Composition:

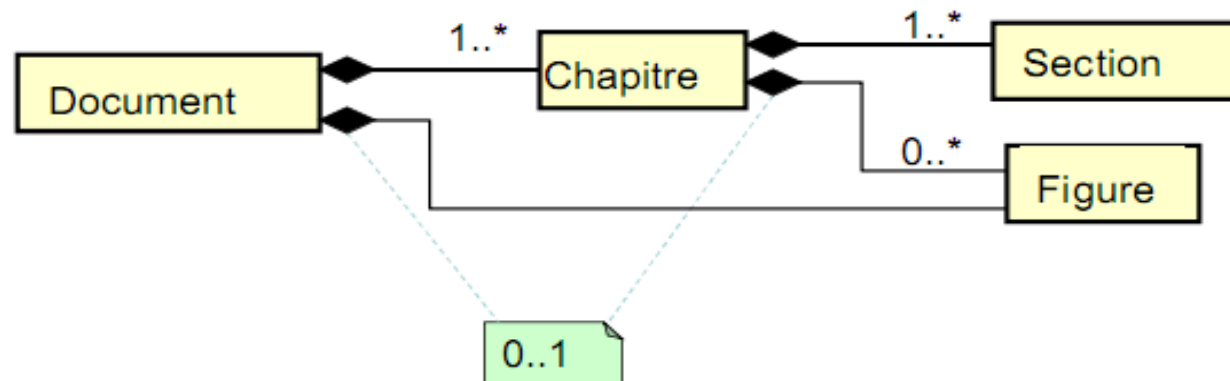
- ▶ Est une relation d'agrégation dont laquelle il existe une contrainte de durée de vie entre la classe composant et la ou les classes composées.
- ▶ Autrement dit, la suppression de la classe mère entraine la suppression de la classe fille.



Concepts de base

Composition (Remarque):

- Pour qu'une agrégation soit une composition, il faut vérifier les deux critères suivants:
 - La multiplicité du côté mère ne doit pas être supérieur à 1.
 - La destruction de l'agrégat composite (la classe mère) entraine la destruction de tous ses éléments.



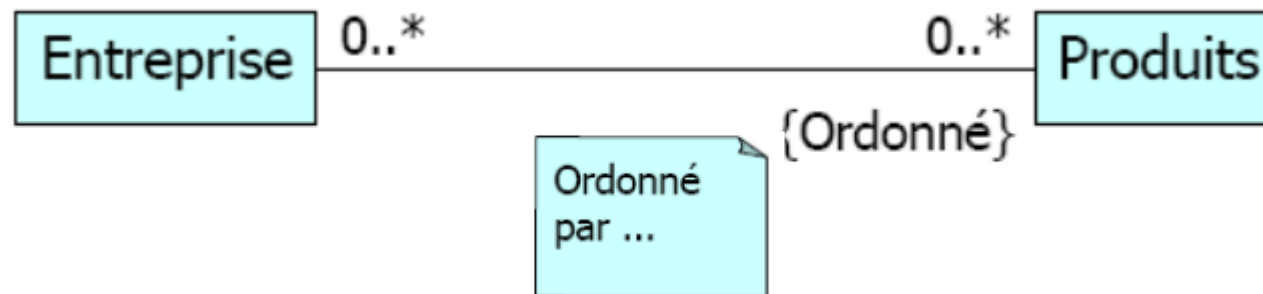
Les différentes contraintes sur les associations

Les différentes contraintes sur les associations

- ▶ Les contraintes (prédéfinies) souvent utilisées :
 - ▶ {sous ensemble}
 - ▶ {xor}
 - ▶ {or}
 - ▶ {ordonné}
 - ▶ {addOnly}
 - ▶ {frozen}

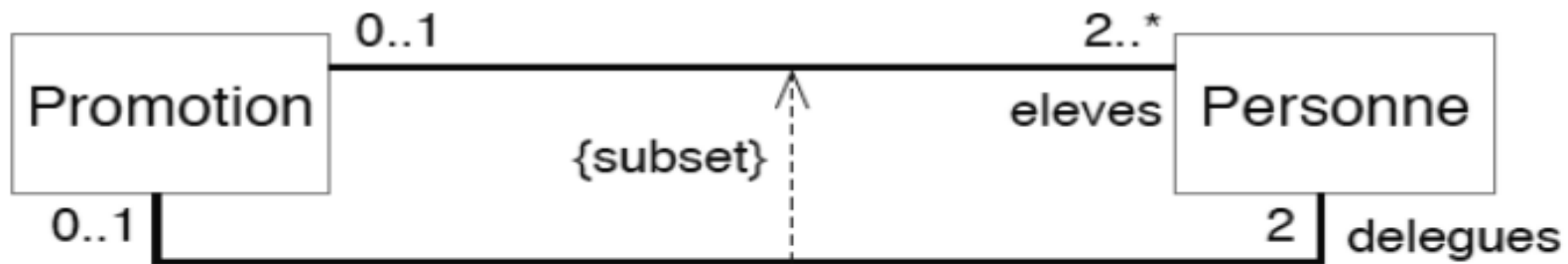
La contrainte {ordonné}

- ▶ Indique que les objets seront ordonnés à chaque opération de:
 - ▶ création,
 - ▶ modification,
 - ▶ suppression, ...
- ▶ Le modèle ne spécifie pas comment les objets sont ordonnés => On utilise un commentaire.



La contrainte {sous-ensemble} ou {subset}

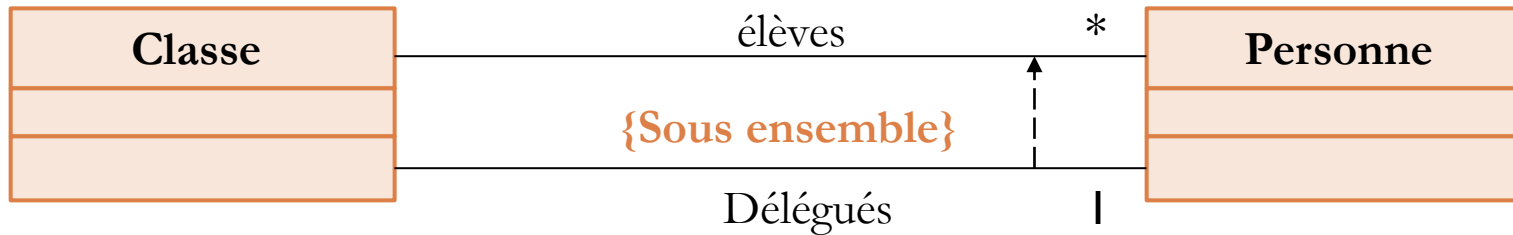
- ▶ Indique qu'une collection est incluse dans une autre.
- ▶ Nécessite la présence d'au moins deux relations.



Les personnes qui jouent le rôle de délégué font partie des personnes qui jouent le rôle d'élèves

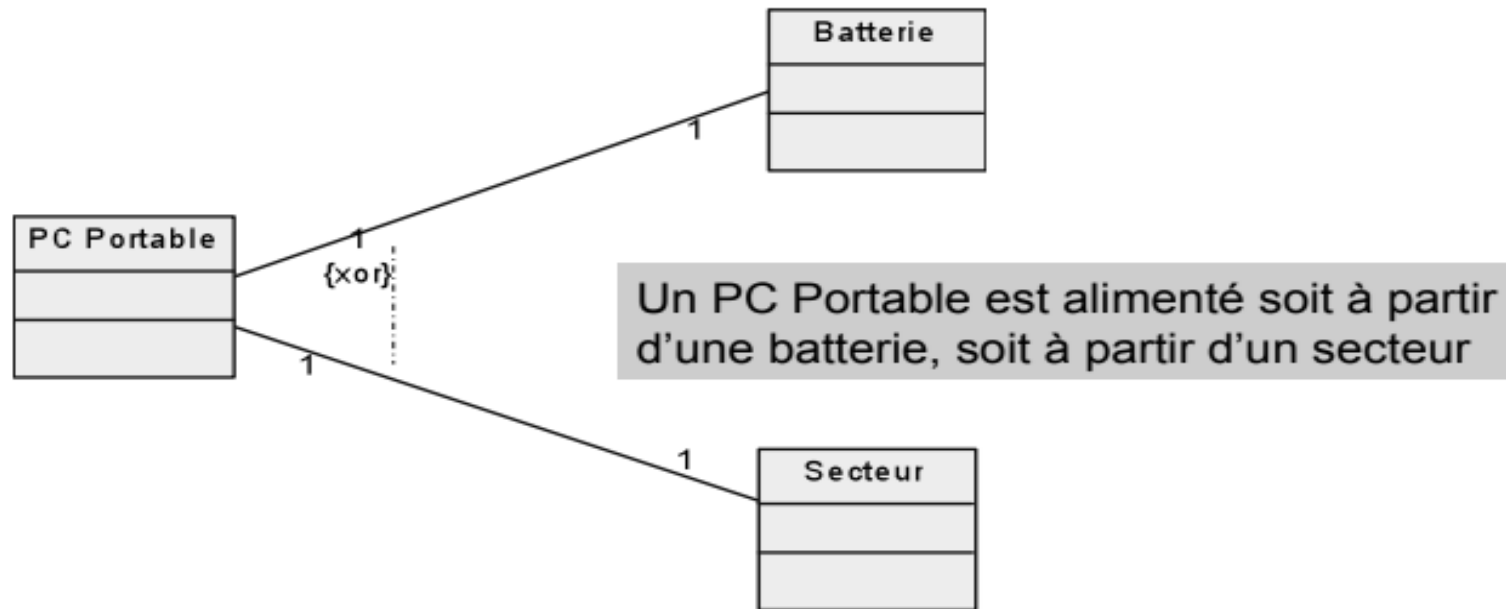
La contrainte {sous-ensemble} ou {subset}

- ▶ Elle indique une collection est incluse dans un autre ensemble.



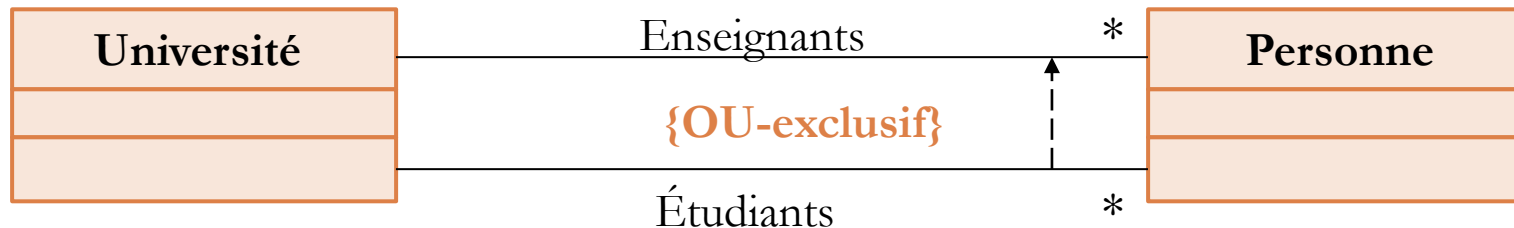
La contrainte {OU-exclusif} ou {xor}

- ▶ Indique que parmi un groupe d'associations, une seule est valide à la fois → pas de simultanéité.



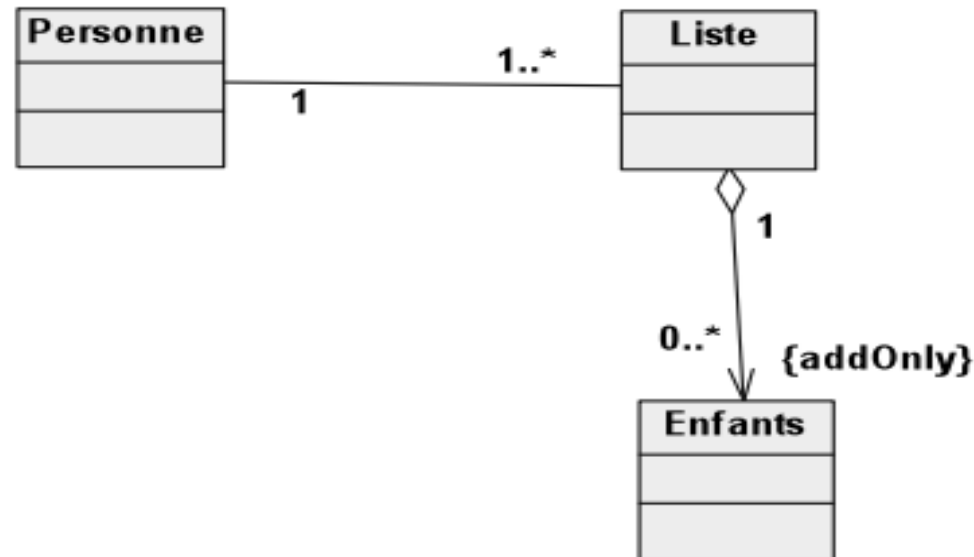
La contrainte {OU-exclusif} ou {xor}

- Cette contrainte précise que pour un objet donné, une seule association parmi un groupe d'association est valide.



La contrainte {addOnly}

- ▶ La contrainte prédéfinie {addOnly} autorise l'ajout de nouveaux objets, mais pas leur suppression ni leur mise à jour.

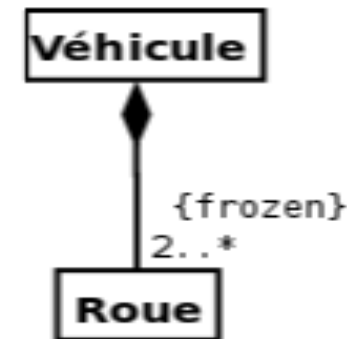


La contrainte {frozen}

► La contrainte prédéfinie {frozen} interdit:

- l'ajout,
- la suppression
- la mise à jour

des liens d'un objet vers les objets de la classe associée, après l'initialisation du premier.



Concepts d'Orienté Objet

Objet

- ▶ Les objets sont une représentation de l'univers du discours du domaine d'application.
- ▶ Ils sont identifiables et entretiennent des relations entre eux.
- ▶ Un objet est un concept, une abstraction ou une chose qui est significative pour le domaine concerné.

Objet = Identité + Comportement + État

Objet

- ▶ **L'État** regroupe les valeurs des différents attributs qui caractérisent un objet peut évoluer ou être constant
- ▶ **Comportement** regroupe toutes les compétences d'un objet et décrit les actions et les réactions de cet objet.
- ▶ Ce comportement est appelé opération (méthode).

Concepts d'Objet, Classe

Objet, classe

- ▶ Une instance est une concrétisation d'un concept abstrait.
- ▶ Par exemple :
 - ▶ la Ferrari Enzo qui se trouve dans un garage est une instance du concept abstrait Automobile;
 - ▶ l'amitié qui lie Jean et Marie est une instance du concept abstrait Amitié;
- ▶ Une classe est un concept abstrait représentant des éléments variés comme :
 - ▶ des éléments concrets (ex : des avions),
 - ▶ des éléments abstraits (ex : des commandes),
 - ▶ des composants d'une application (ex : les boutons des boîtes de dialogue),
 - ▶ des structures informatiques (ex : des tables de hachage),
 - ▶ des éléments comportementaux (ex : des tâches), etc.

Objet, classe

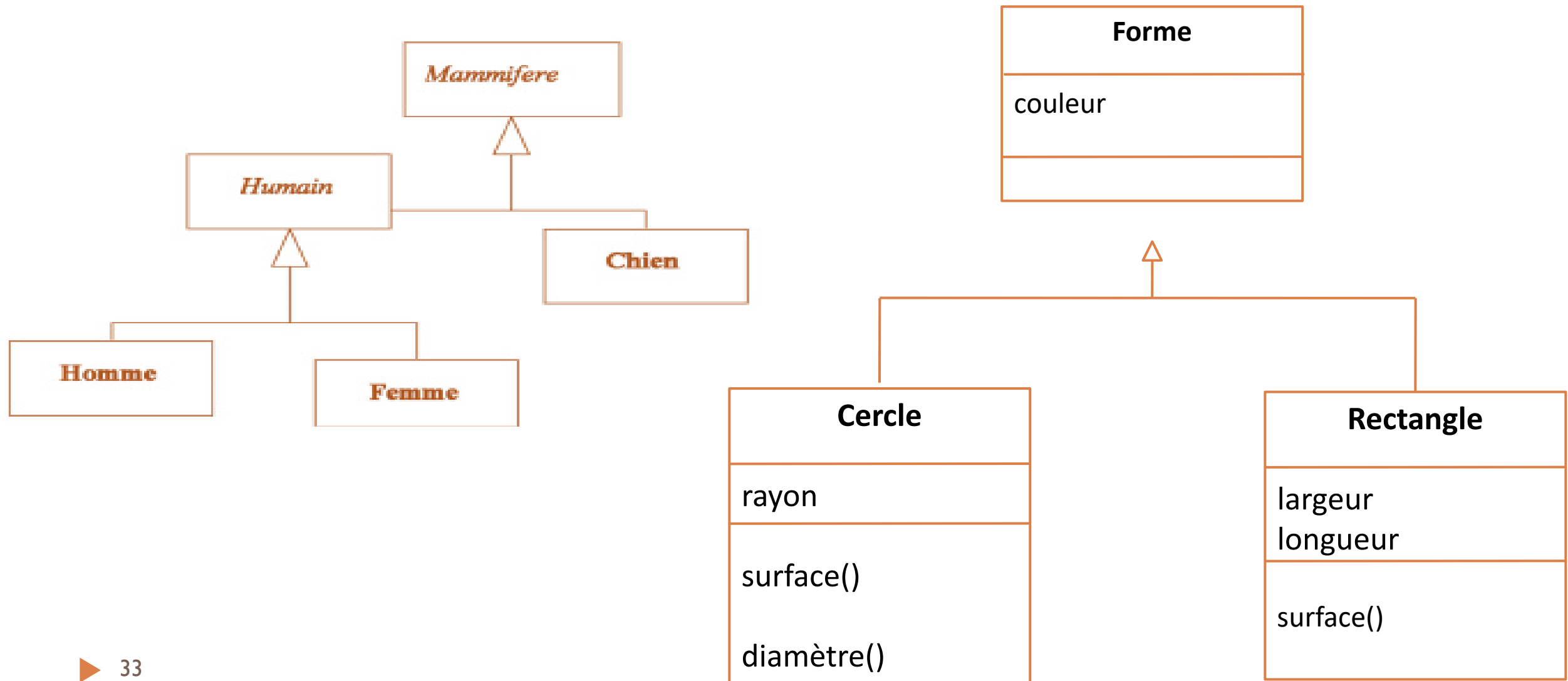
- ▶ Tout système orienté objet est organisé autour des classes.
- ▶ Une classe est la description formelle d'un ensemble d'objets ayant une sémantique et des propriétés communes.
- ▶ Un objet est une instance d'une classe.
- ▶ C'est une entité discrète dotée d'une identité, d'un état et d'un comportement que l'on peut invoquer.
- ▶ Les objets sont des éléments individuels d'un système en cours d'exécution.
 - ▶ Par exemple, si l'on considère que Homme (au sens être humain) est un concept abstrait, on peut dire que la personne *François* est une instance de Homme.
 - ▶ Si Homme était une classe, *François* en serait une instance → un objet.
- ▶ Une **classe** est la description d'une collection d'objets ayant la même structure, le même comportement, les mêmes relations ainsi que la même sémantique
- ▶ **L'instanciation** est la relation entre un objet et sa classe

Concept d'abstraction

Abstraction

- ▶ L'abstraction est un principe très important en modélisation.
- ▶ Elle consiste à retenir uniquement les propriétés pertinentes d'un objet pour un problème précis.
- ▶ Les objets utilisés en UML sont des abstractions d'objets du monde réel.
- ▶ Permet de s'attacher aux aspects essentiels sans entrer dans les détails, autrement dit de se concentrer sur ce que représente l'objet et sur son comportement avant de décider de la façon de l'implémenter.

Exemples



Concepts de généralisation/spécification

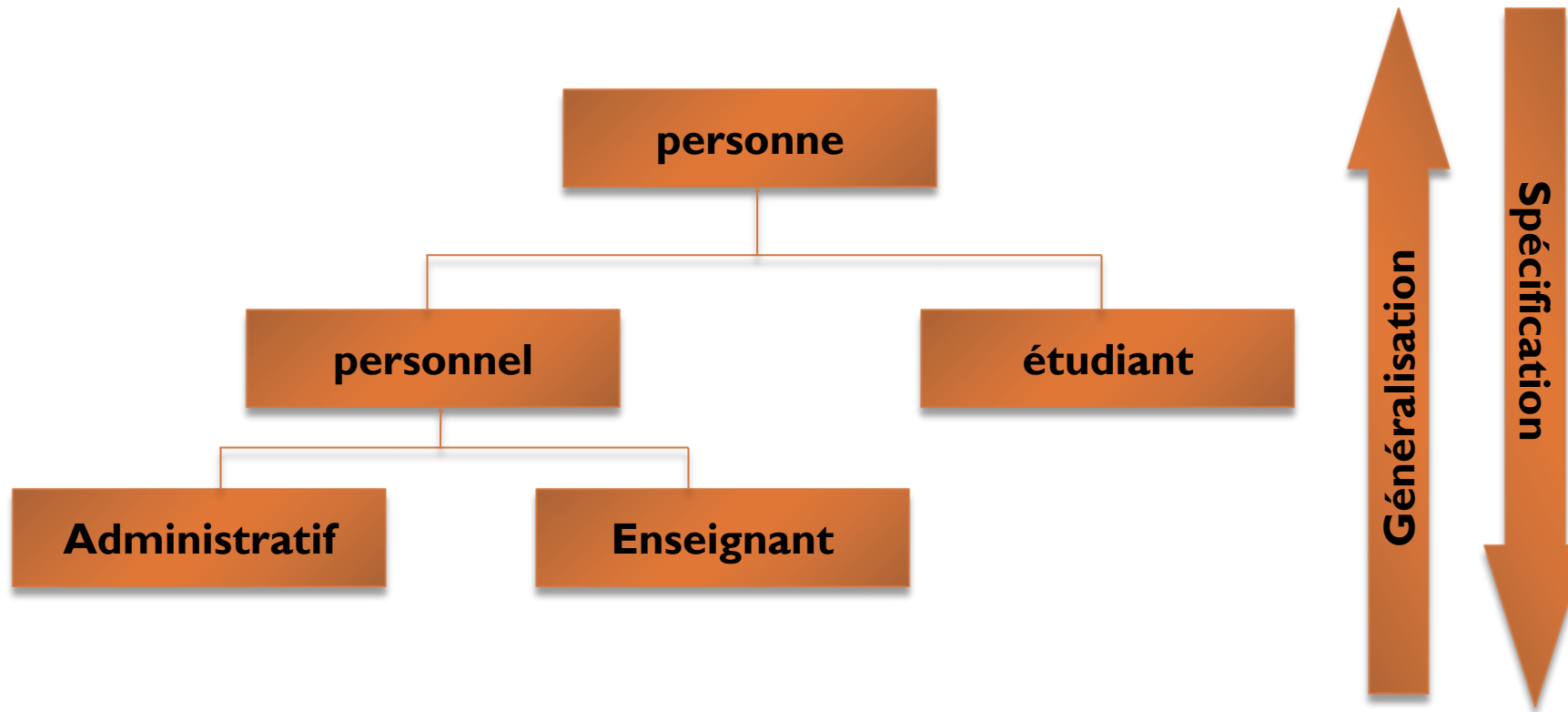
Concepts de généralisation/spécification

- ▶ Le concept de généralisation traduit la notion de relation entre une classe et deux ou plusieurs autres classes qui partagent un sous ensemble commun d'attributs et/ou d'opérations.
- ▶ La classe générique est appelée *superclasse*.
- ▶ Les classes spécialisées s'appellent *sous classe*.

Concepts de généralisation/spécification

- ▶ L'opération qui consiste à créer une superclasse à partir des classes spécialisées est appelée ***généralisation***; dans le cas contraire ***spécialisation***.
- ▶ Les attributs et les opérations d'une superclasse sont transmis aux sous classes par le lien d'héritage.

Exemple

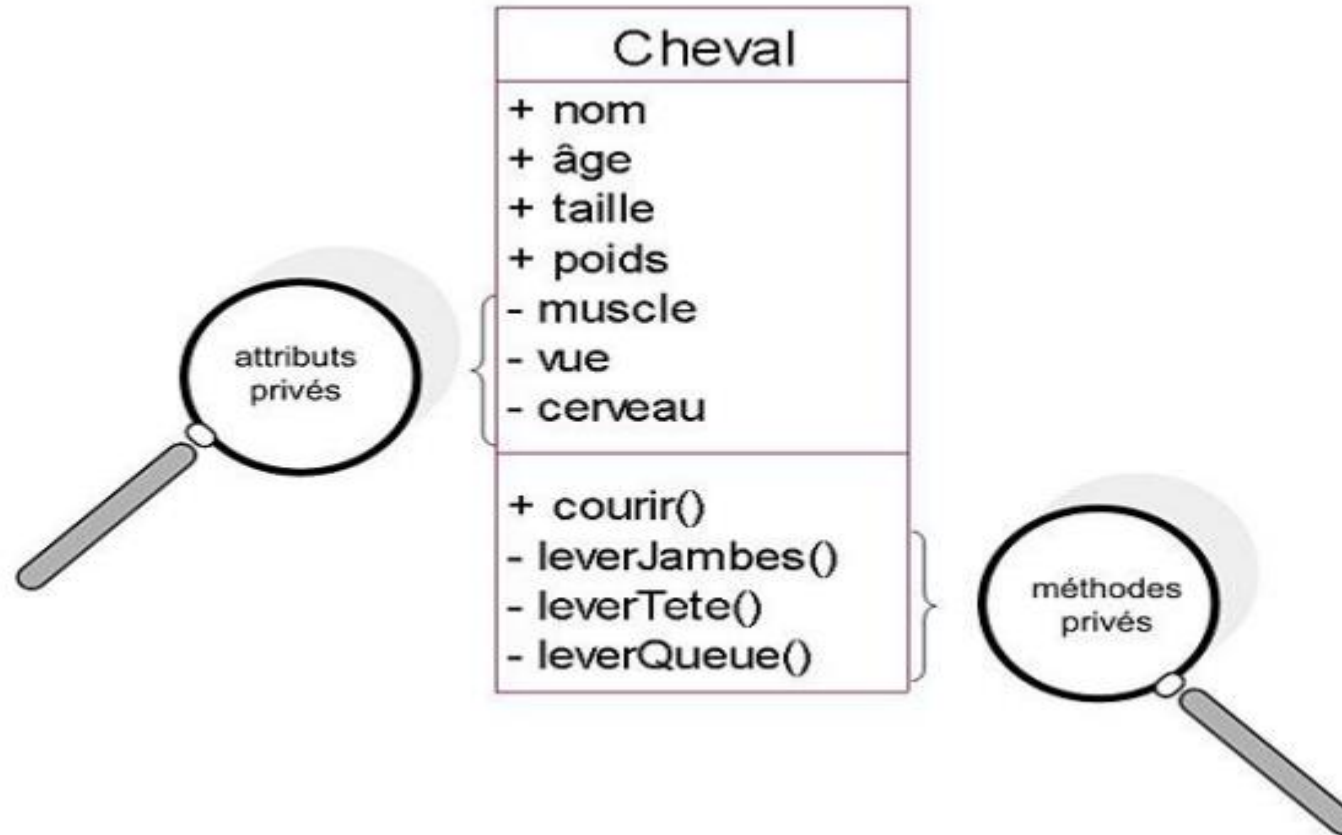


Concept d'Encapsulation

Encapsulation

- ▶ Sépare les aspects externes d'un objet, accessibles aux autres objets, des détails d'implémentation internes, qui leur sont cachés.
- ▶ L'encapsulation consiste à masquer des attributs et des méthodes de l'objet vis-à-vis des autres objets.
- ▶ En effet, certains attributs et méthodes ont pour seul objectif des traitements internes à l'objet et ne doivent pas être exposés aux objets extérieurs.
- ▶ Encapsulés, ils sont appelés les attributs et méthodes privés de l'objet. L'encapsulation est une abstraction puisque l'on simplifie la représentation de l'objet vis-à-vis des objets extérieurs.
- ▶ Cette représentation simplifiée est constituée des attributs et méthodes publiques de l'objet.
- ▶ La définition de l'encapsulation se fait au niveau de la classe.
- ▶ Les objets extérieurs à un objet sont donc les instances des autres classes.

Exemple



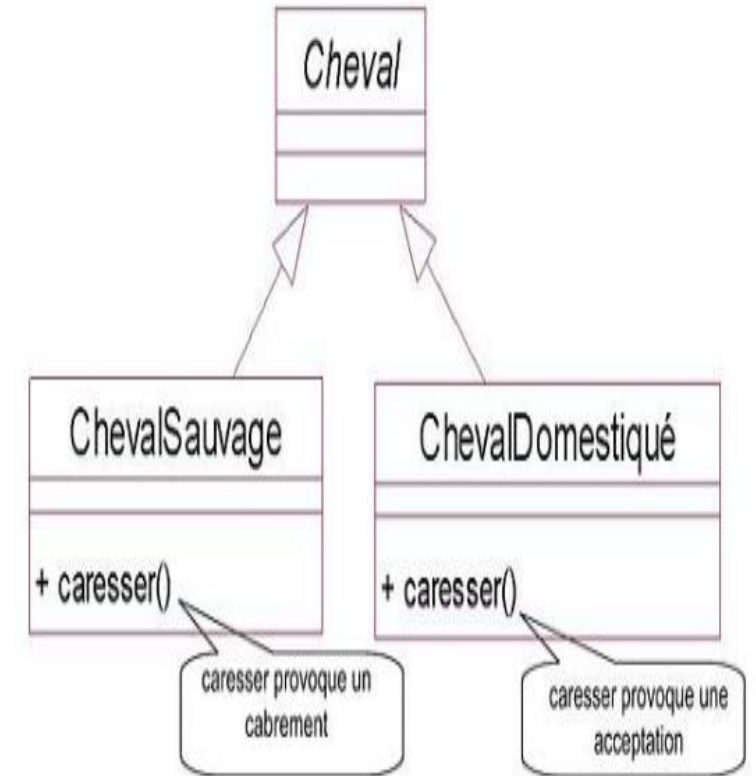
Concept de Polymorphisme

Polymorphisme

- ▶ Le polymorphisme signifie qu'une classe représente un ensemble constitué d'objets différents car ils sont instances de sous-classes distinctes.
- ▶ Lors de l'appel d'une méthode de même nom, cette différence se traduit par des comportements différents (sauf dans le cas où la méthode est commune et héritée de la superclasse dans les sous-classes).

Exemple

- ▶ La méthode caresser a un comportement différent selon que le cheval est instance de *ChevalSauvage* ou de *ChevalDomestiqué*.
- ▶ Dans le premier cas, le comportement sera un refus (se traduisant par un cabrement) alors que dans le second, le comportement sera une acceptation.
- ▶ Si l'on considère la classe Cheval dans son intégralité, on a donc un ensemble de chevaux qui ne réagissent pas de la même façon lors de l'activation de la méthode caresser.



Notion de paquetage

Les paquetages

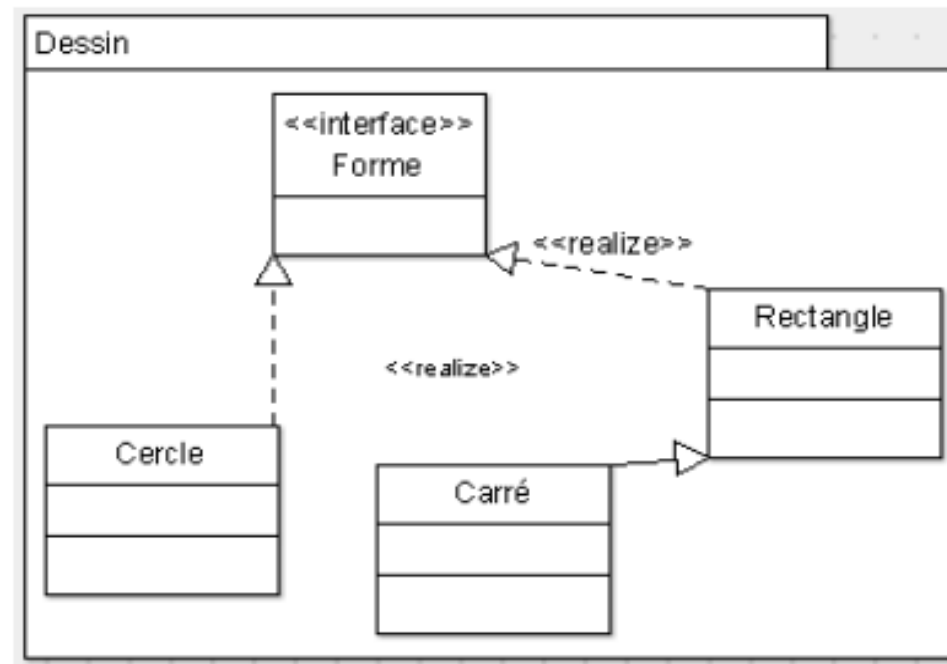
- ▶ Les paquetages sont des éléments d'organisation des modèles.
- ▶ Ils regroupent des éléments de modélisation, selon des critères purement logiques.
- ▶ Ils représentent le bon niveau de granularité pour la réutilisation.

Les paquetages

- ▶ Il permettent de:
 - ▶ Organiser:
 - ▶ Introduire des niveaux d'abstraction
 - ▶ Structurer en couches
 - ▶ Découper un problème en sous-problèmes
 - ▶ Découper en sous-systèmes
 - ▶ Structurer pour réutiliser et pour comprendre
 - ▶ Unités logiques
 - ▶ Unités homogènes

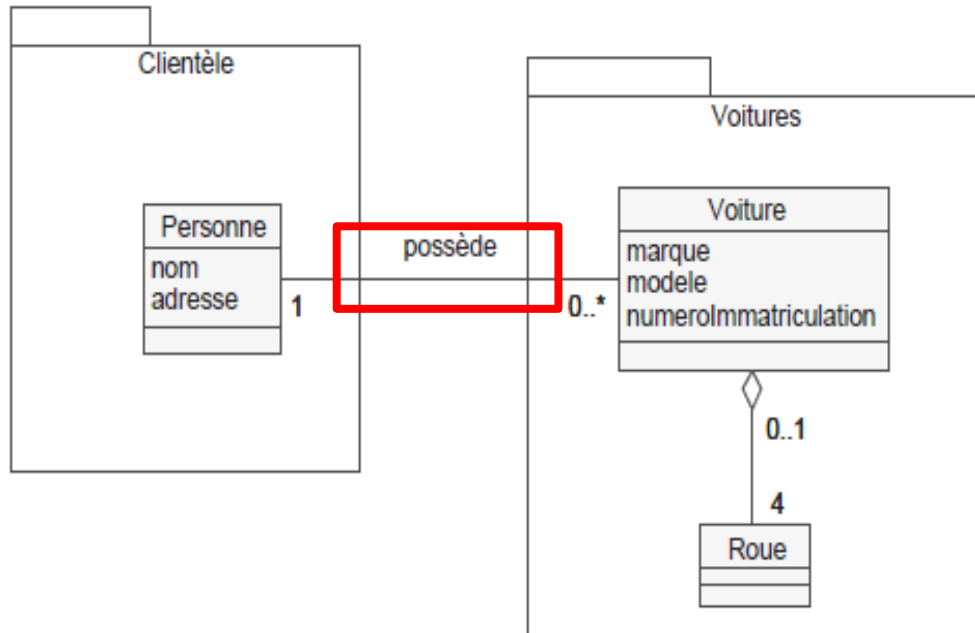
Les paquetages

- ▶ Un package est un regroupement de concepts.
- ▶ Un package est représenté par un rectangle possédant un onglet dans lequel est inscrit le nom du package.

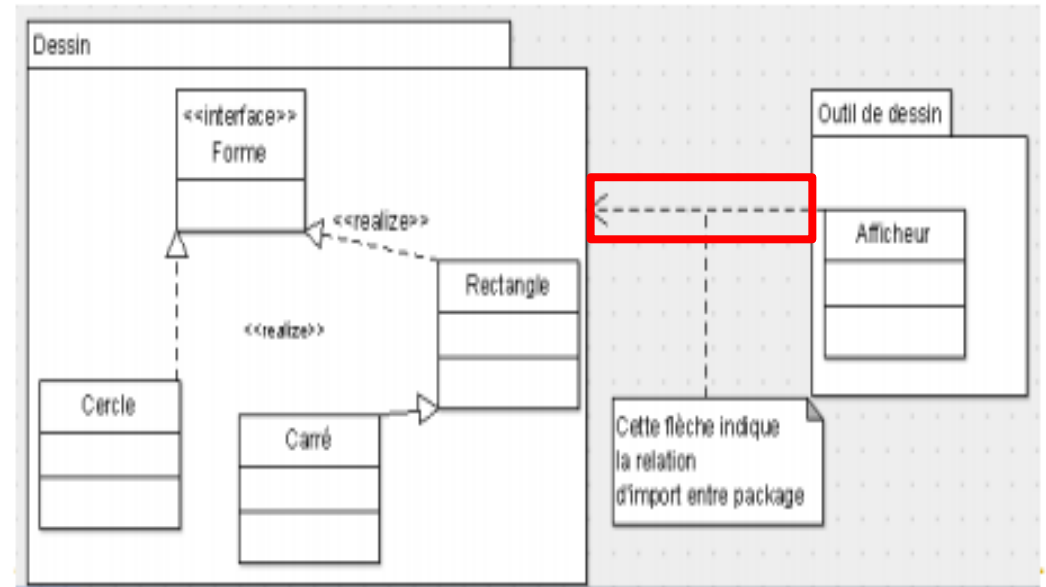


Exemples de packages contenant des classes

Exemple 1:



Exemple 2:



Les interfaces


Rôle d'une interface

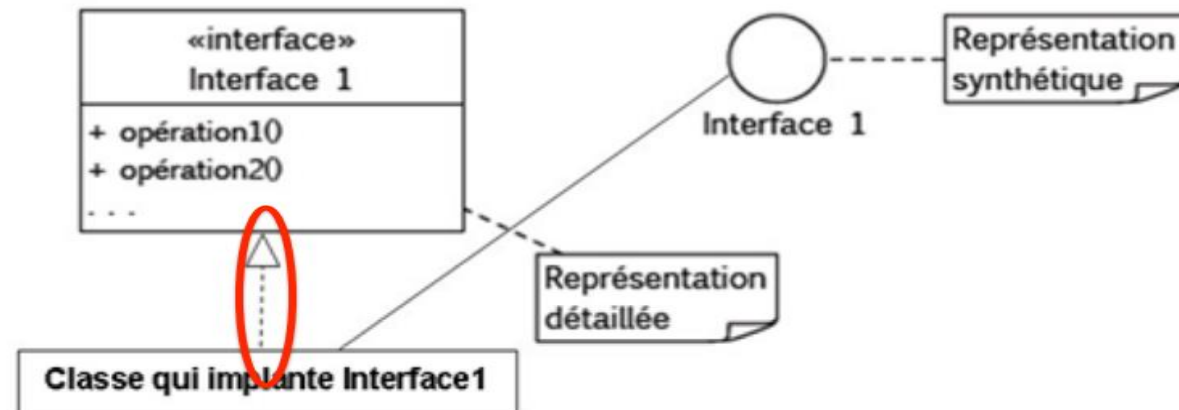
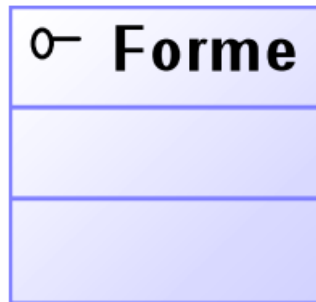
- ▶ Elles sont utiles pour des équipes qui travaillent sur un même projet, chacune doit pouvoir connaître les méthodes qui seront implémentées par les autres et leur spécification pour pouvoir inclure les appels dans leurs propres méthodes.

Les interfaces

- ▶ La classe d'interface est une spécification et non une classe réelle.
- ▶ Seules des signatures de méthodes sont présentes dans les interfaces.
- ▶ Le terme d'héritage n'est pas utilisé entre classes et interfaces :
 - ▶ on dit qu'une classe implémente ou réalise une interface,
- ▶ On parle de relation de réalisation entre une interface et une classe qui l'implémente.
- ▶ Une classe qui réalise une interface doit définir le corps de toutes ses méthodes abstraites.

Schématisation d'une interface

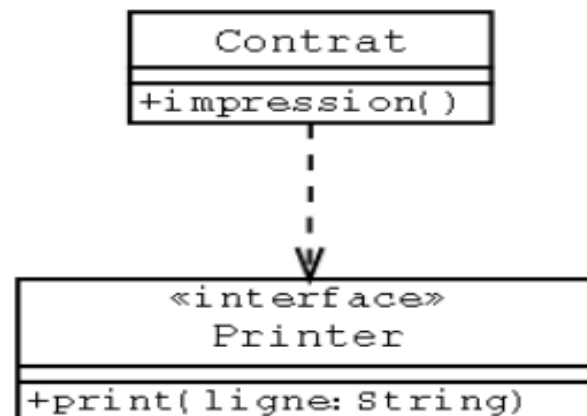
- ▶ Une interface est une classe spéciale dont toutes les méthodes sont abstraites.
- ▶ Une interface se note en UML avec le stéréotype `<<interface>>` ou le symbole  .



Dépendance entre les classes

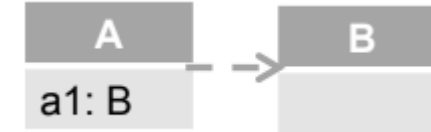
Dépendances entre les classes

- ▶ Une classe A dépend d'une classe B
 - ▶ noté $A - - - - - > B$
- ▶ Exemple: Un contrat dispose d'un service d'impression (méthode impression), qui utilise une méthode (print), dont la spécification est déclarée par l'interface Printer.

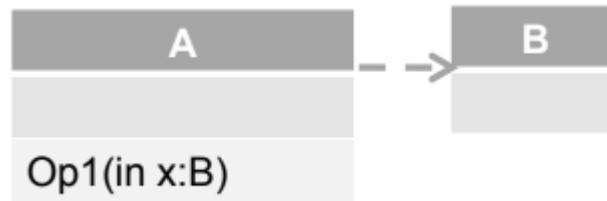


Dépendances entre les classes

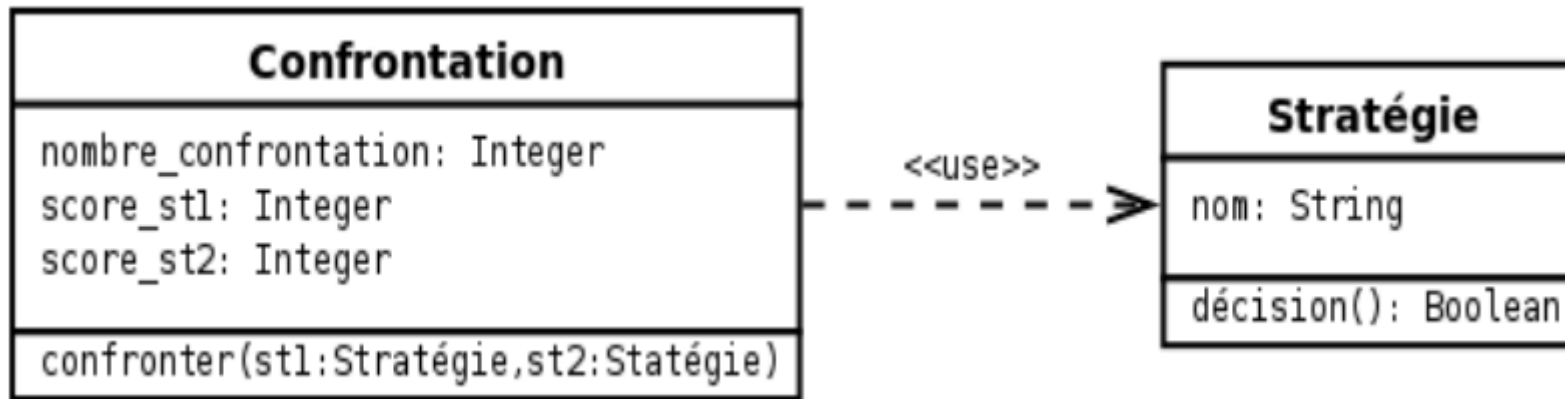
- ▶ A possède un attribut dont le type est B.



- ▶ A possède une opération dont le type de l'un de ses paramètres est B.



Dépendances entre les classes



- ▶ la classe **Confrontation** utilise la classe **Stratégie** car la classe **Confrontation** possède l'opération `confronter` dont ses deux paramètres sont du type **Stratégie**.

Exercices d'application

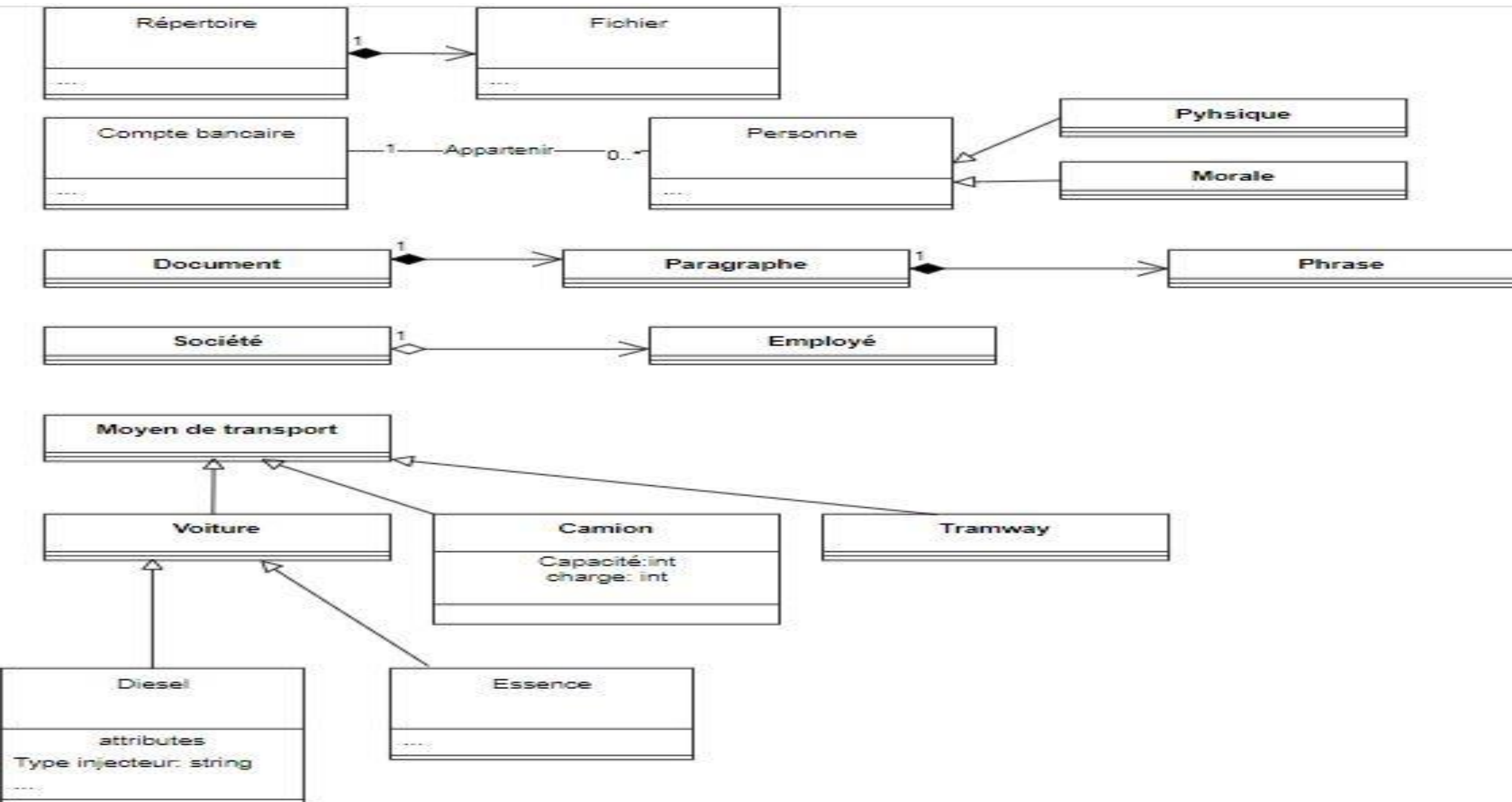
Exercice 1

Considérons les phrases suivantes:

1. Un répertoire contient des fichiers.
2. Un compte bancaire peut appartenir à une personne physique ou morale.
3. Un document est composé de paragraphes. Un paragraphe est composé de phrases.
4. Une société contient des employés.
5. Voitures, camion et tramway sont des moyens de transport. Pour les camions on veut connaître aussi leur capacité et leur charge. Une voiture peut être diesel ou essence. Pour les voitures diesel on veut connaître en plus le type d'injecteur.

Déterminer la relation appropriée (généralisation, spécification, composition, agrégation ou association) dans chacune de ces phrases en dessinant le diagramme de classe correspondant à chaque cas.

Correction



Exercice 2

Il s'agit d'établir le schéma conceptuel d'une base de données pour la gestion des formations d'un institut privé :

Un cours est caractérisé par un numéro de cours (NOCOURS), un libellé (LIBELLE), une durée en heures (DUREE) et un type (TYPE). Un cours peut faire l'objet dans l'année de plusieurs sessions identiques. Une session est caractérisée par un numéro (NOSES), une date de début (DATE) et un prix (PRIX). Une session est le plus souvent assurée par plusieurs animateurs et est placée sous la responsabilité d'un animateur principal. Un animateur peut intervenir dans plusieurs sessions au cours de l'année. On désire mémoriser le nombre d'heures (NBH) effectué par un animateur pour chaque session. Un animateur est caractérisé par un numéro (NOANI), un nom (NOMA) et une adresse (ADRA).

Chaque session est suivie par un certain nombre de participants. Un participant est une personne indépendante ou un employé d'une entreprise cliente. Un participant est caractérisé par un numéro (NOPAR), un nom (NOMP) et une adresse (ADRP). Dans le cas d'un employé, on enregistre le nom (NOMEN) et l'adresse de l'entreprise (ADREN).

On désire pouvoir gérer d'une manière séparée (pour la facturation notamment) les personnes indépendantes d'une part, et les employés d'autre part.