



UNIVERSITÉ  
DE MONTPELLIER



# Modélisation des Objets

Ressource R3.03 — Analyse

Nadjib Lazaar ([nadjib.lazaar@umontpellier.fr](mailto:nadjib.lazaar@umontpellier.fr))

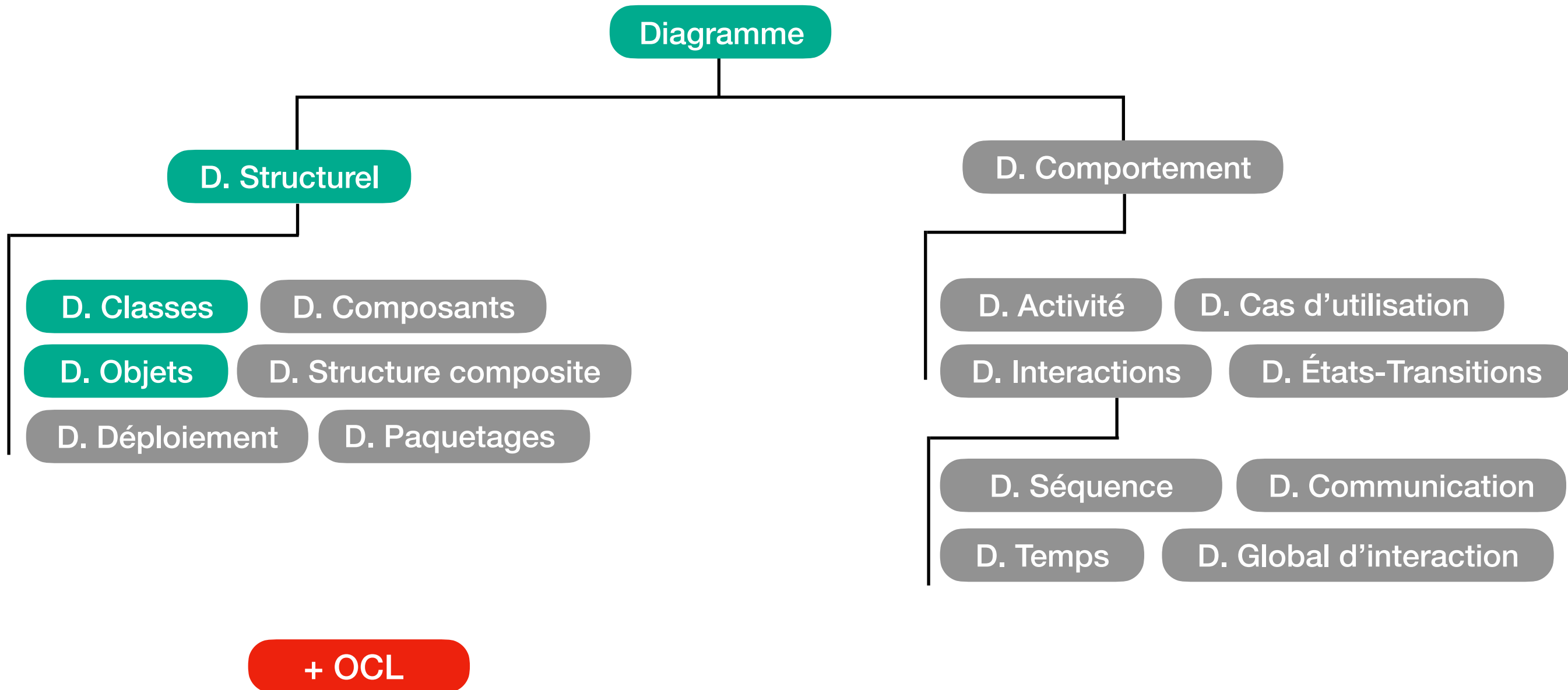
# UML

## Dans ce module

- **Specification**
  - Besoins des utilisateurs (diag. cas d'utilisations)
  - Interaction Utilisateur <-> Logiciel (diag. séquence)
- **Conception**
  - **Structure interne du logiciel (diag. classes)**
  - **État interne du logiciel à l'instant T (diag. objets)**
  - Évolution des objets (diag. états-transitions)
  - Interaction des objets (diag. séquence)

# UML

## Les diagrammes



# Diagrammes d'objets

## Définition

- Représentation des objets et leurs relations à un instant donné
- Graphe (noeuds = objets, arrêtes = relations)
- Permet de décrire des cas de test, illustrer et analyser l'état du système, expliquer les cas particuliers, ...

# Diagrammes d'objets

## Définition

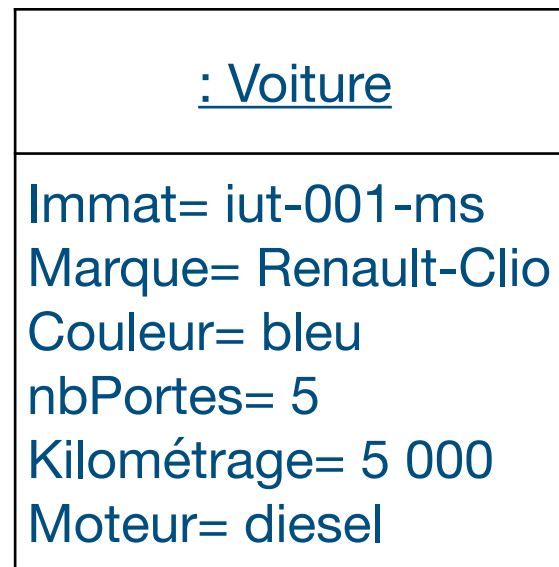
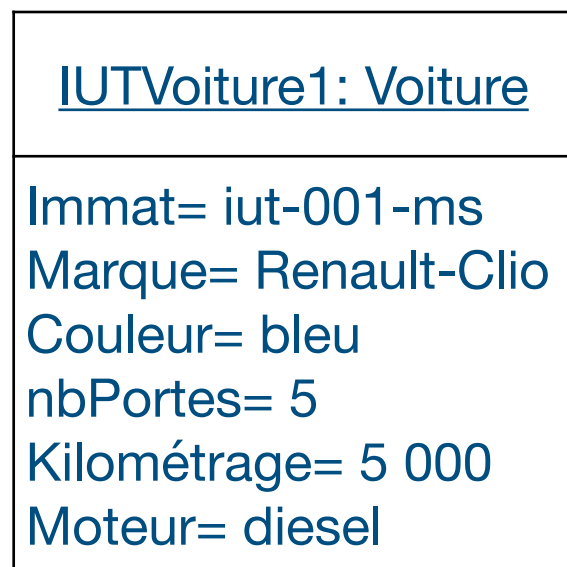
- Représentation des objets et leurs relations à un instant donné
- Graphe (noeuds = objets, arrêtes = relations)
- Permet de décrire des cas de test, illustrer et analyser l'état du système, expliquer les cas particuliers, ...

<u>IUTVoiture1: Voiture</u>
Immat= iut-001-ms Marque= Renault-Clio Couleur= bleu nbPortes= 5 Kilométrage= 5 000 Moteur= diesel

# Diagrammes d'objets

## Définition

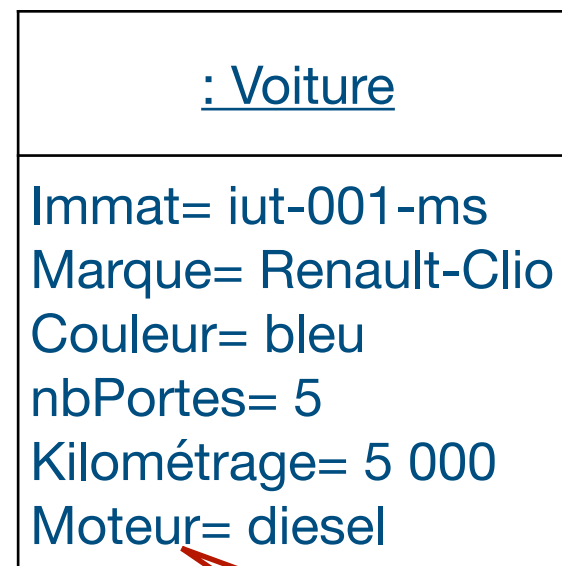
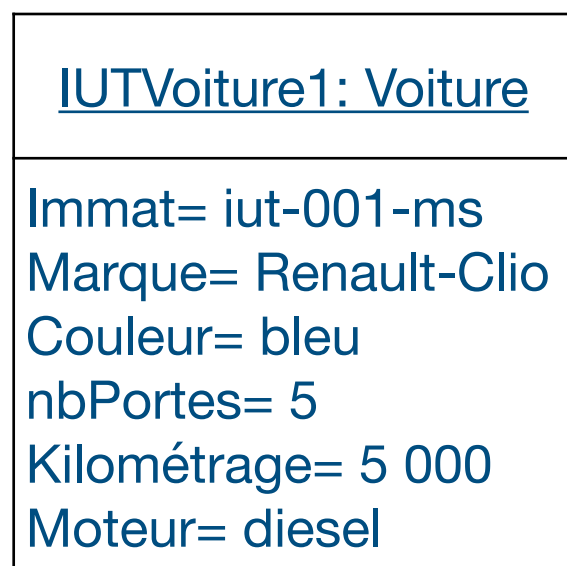
- Représentation des objets et leurs relations à un instant donné
- Graphe (noeuds = objets, arrêtes = relations)
- Permet de décrire des cas de test, illustrer et analyser l'état du système, expliquer les cas particuliers, ...



# Diagrammes d'objets

## Définition

- Représentation des objets et leurs relations à un instant donné
- Graphe (noeuds = objets, arrêtes = relations)
- Permet de décrire des cas de test, illustrer et analyser l'état du système, expliquer les cas particuliers, ...

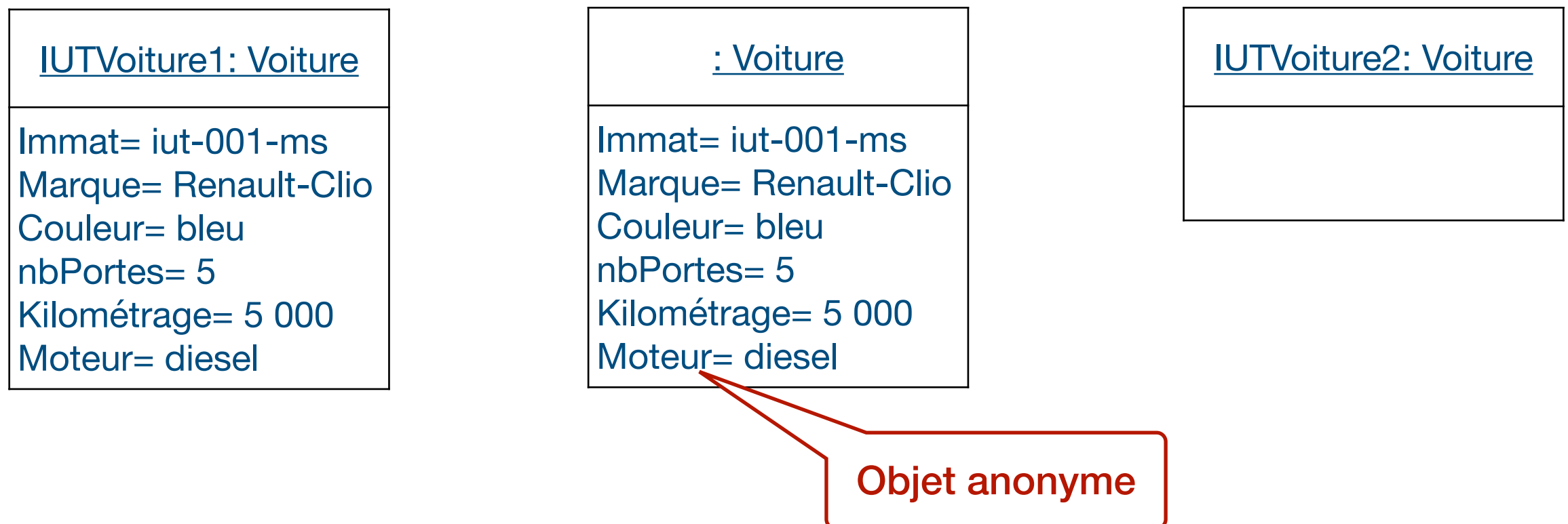


Objet anonyme

# Diagrammes d'objets

## Définition

- Représentation des objets et leurs relations à un instant donné
- Graphe (noeuds = objets, arrêtes = relations)
- Permet de décrire des cas de test, illustrer et analyser l'état du système, expliquer les cas particuliers, ...

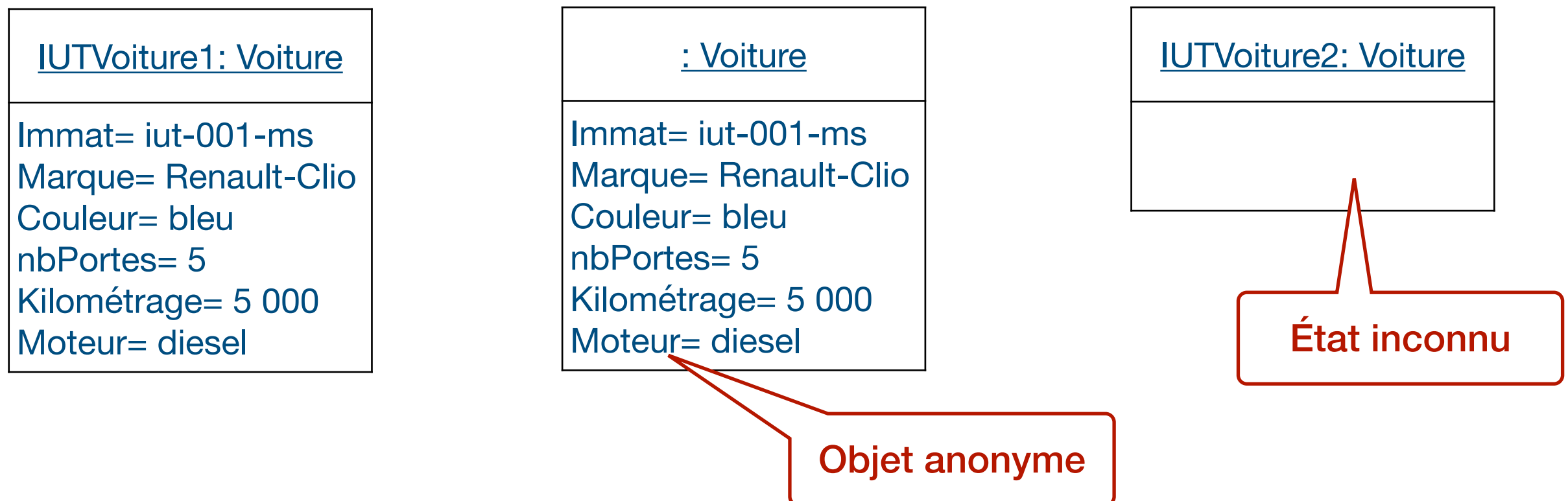




# Diagrammes d'objets

## Définition

- Représentation des objets et leurs relations à un instant donné
- Graphe (noeuds = objets, arrêtes = relations)
- Permet de décrire des cas de test, illustrer et analyser l'état du système, expliquer les cas particuliers, ...



# Diagrammes de classes

## Définition

- Abstraction de l'ensemble des diagrammes d'objets possibles
- Graphe (noeuds = classes, arrêtes = relations)
- Permet la définition des objets possibles + structure du système

# Diagrammes de classes

## Définition

- Abstraction de l'ensemble des diagrammes d'objets possibles
- Graphe (noeuds = classes, arrêtes = relations)
- Permet la définition des objets possibles + structure du système



# Diagrammes de classes

## Définition

- Abstraction de l'ensemble des diagrammes d'objets possibles
- Graphe (noeuds = classes, arrêtes = relations)
- Permet la définition des objets possibles + structure du système

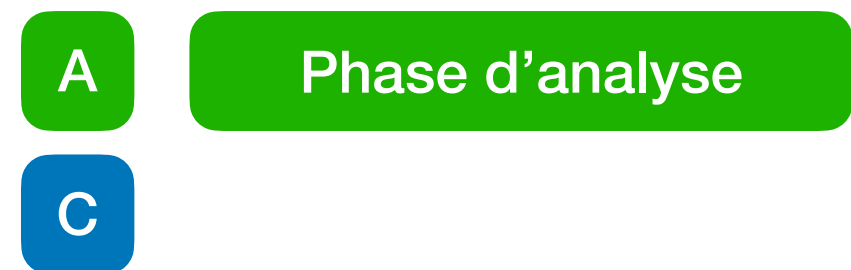
A

Phase d'analyse

# Diagrammes de classes

## Définition

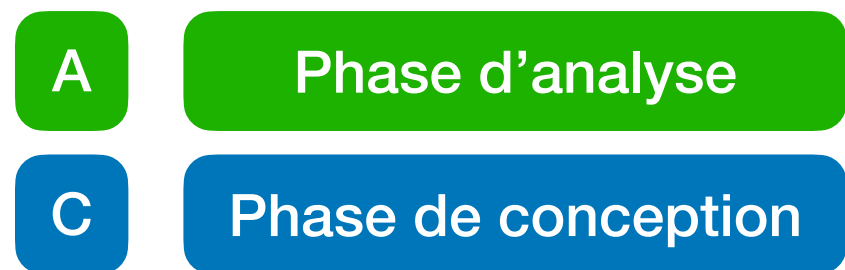
- Abstraction de l'ensemble des diagrammes d'objets possibles
- Graphe (noeuds = classes, arrêtes = relations)
- Permet la définition des objets possibles + structure du système



# Diagrammes de classes

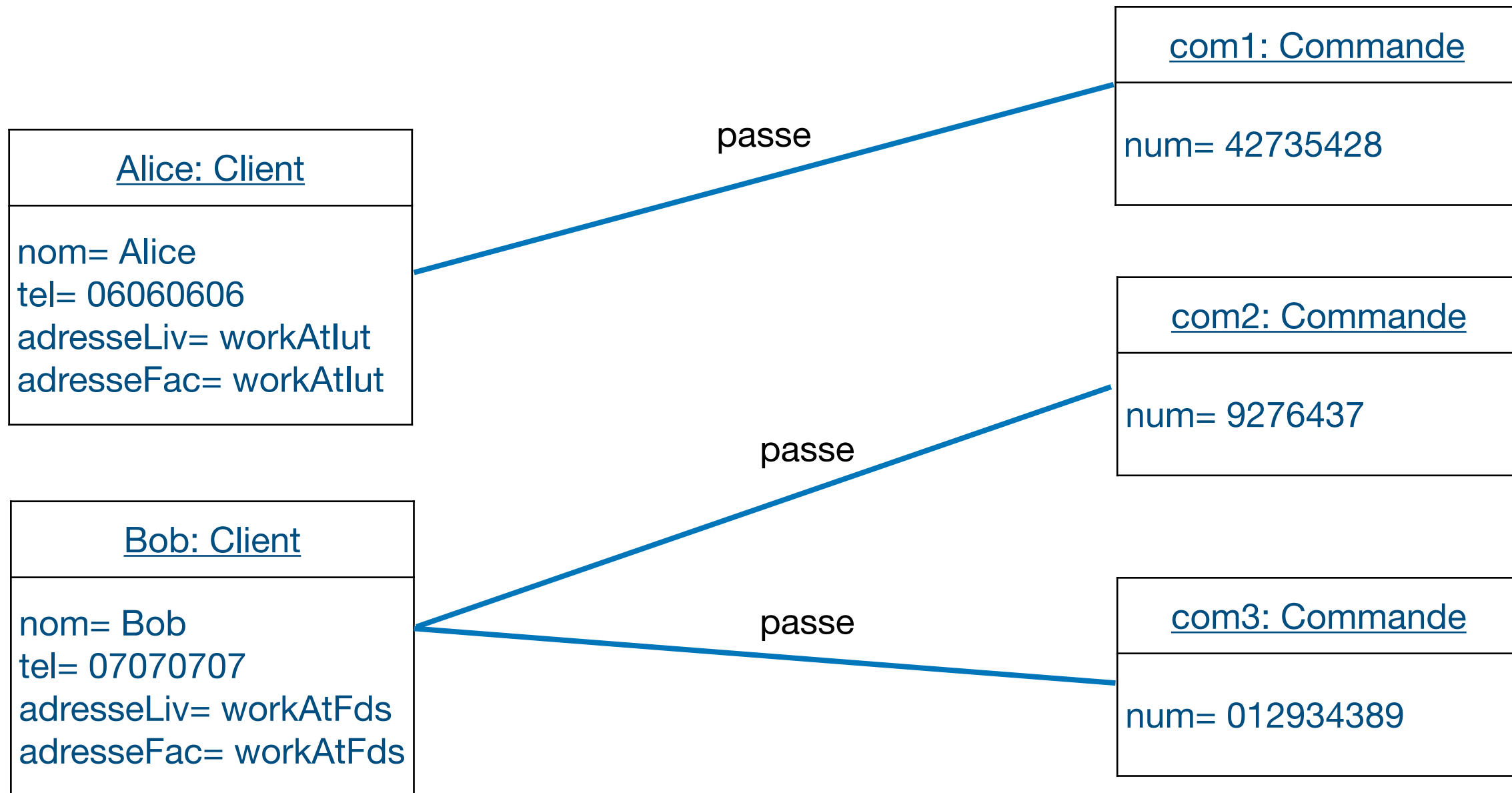
## Définition

- Abstraction de l'ensemble des diagrammes d'objets possibles
- Graphe (noeuds = classes, arrêtes = relations)
- Permet la définition des objets possibles + structure du système



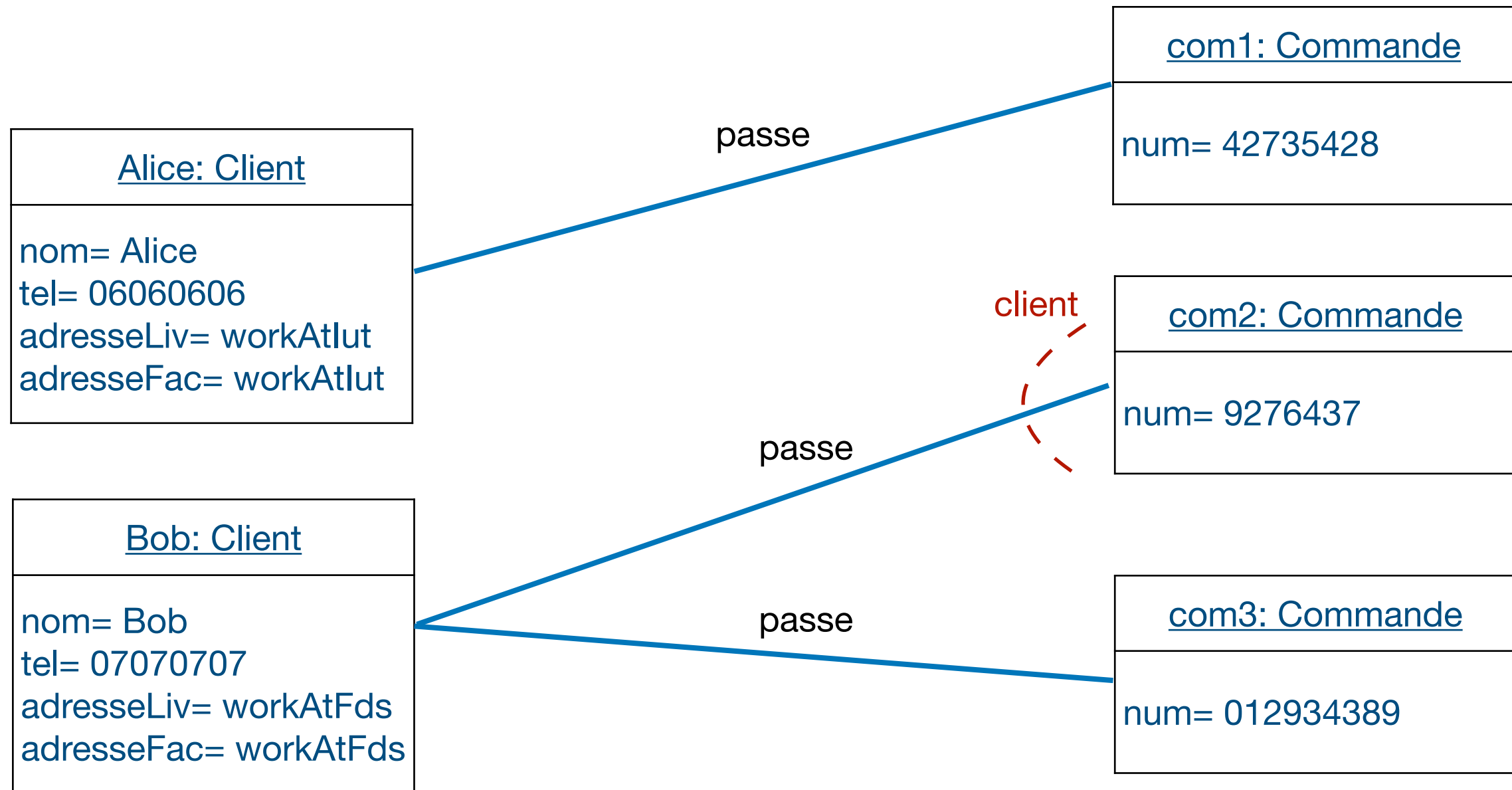
# Relation entre objets

## Association



# Relation entre objets

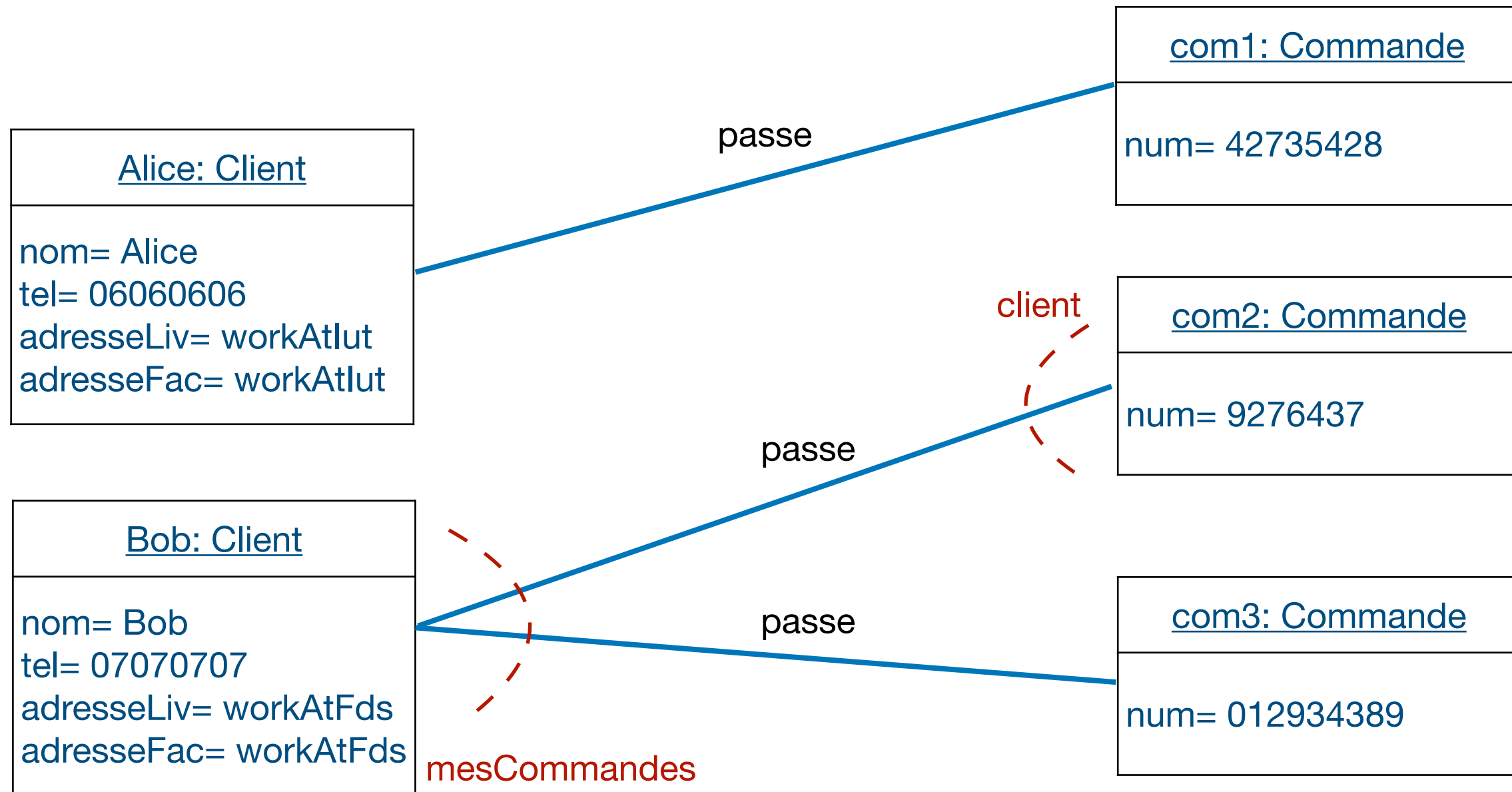
## Association





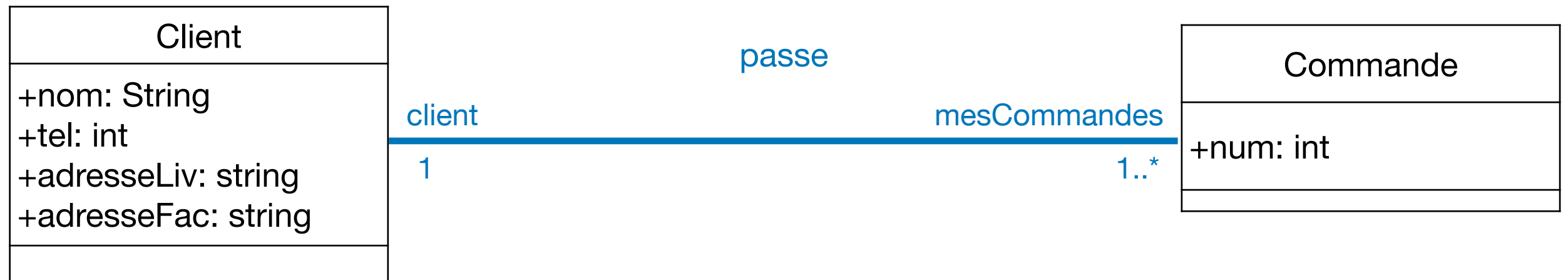
# Relation entre objets

## Association



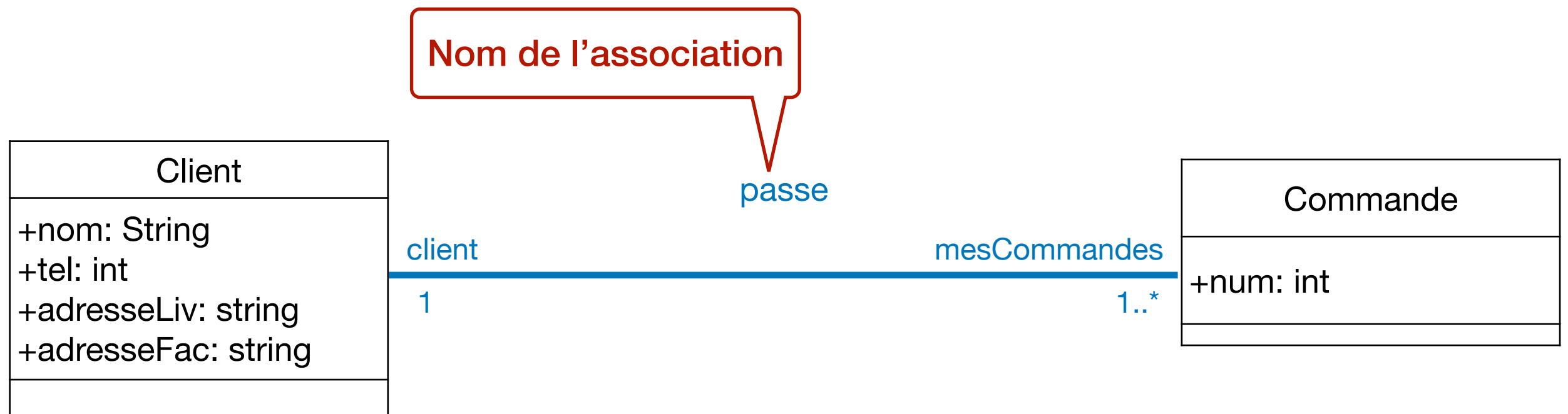
# Relation entre classes

## Association



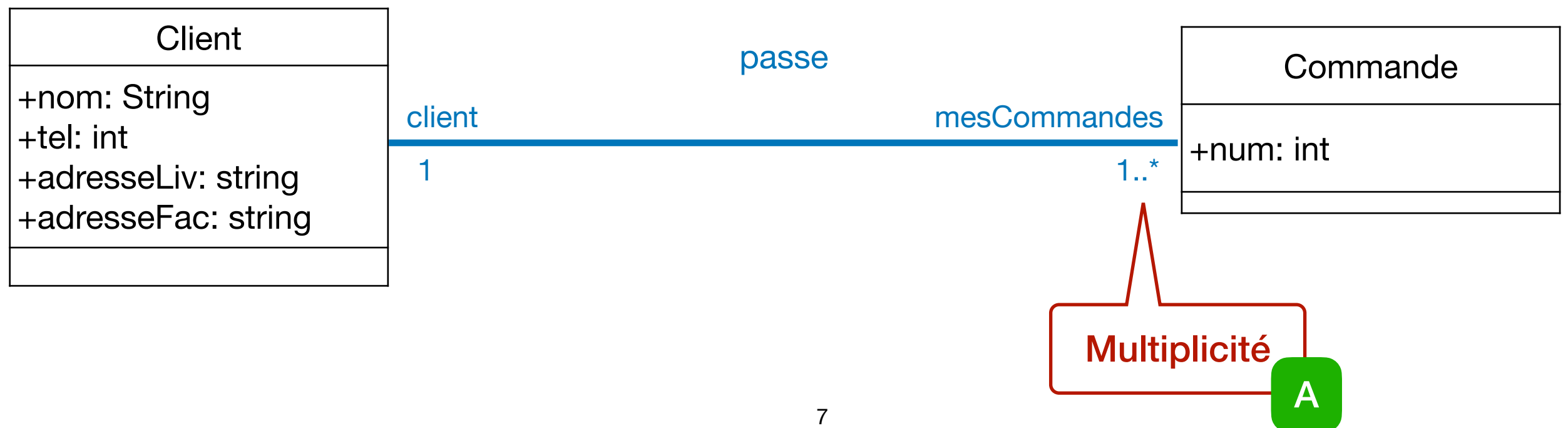
# Relation entre classes

## Association



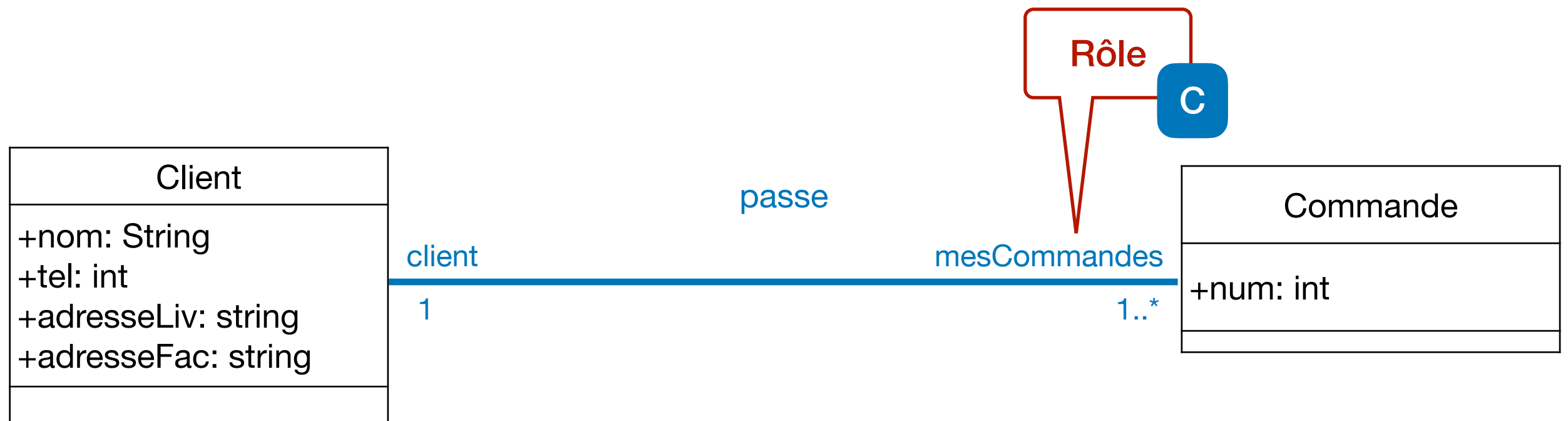
# Relation entre classes

## Association



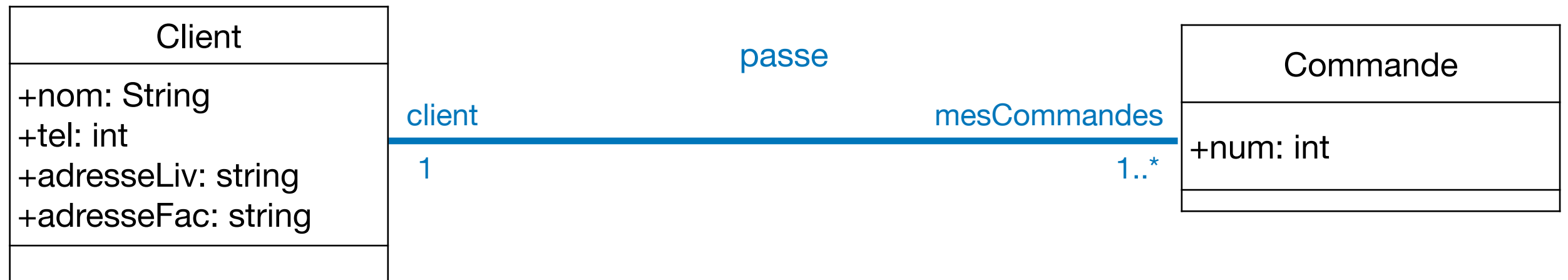
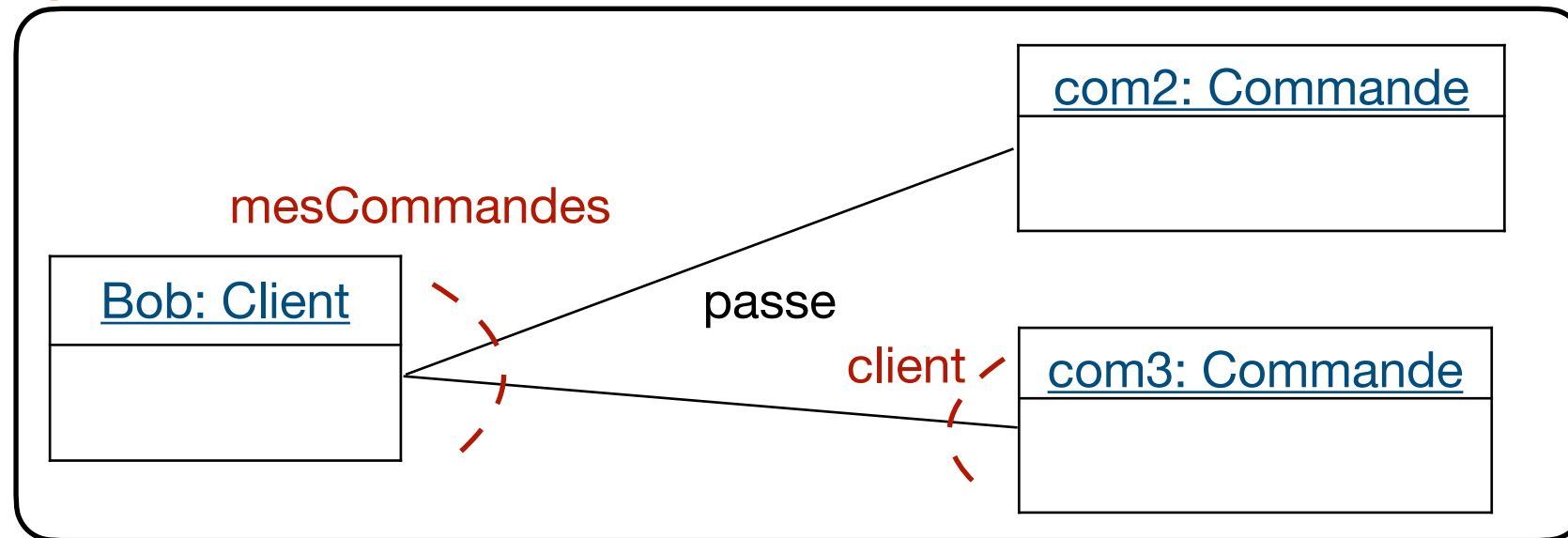
# Relation entre classes

## Association



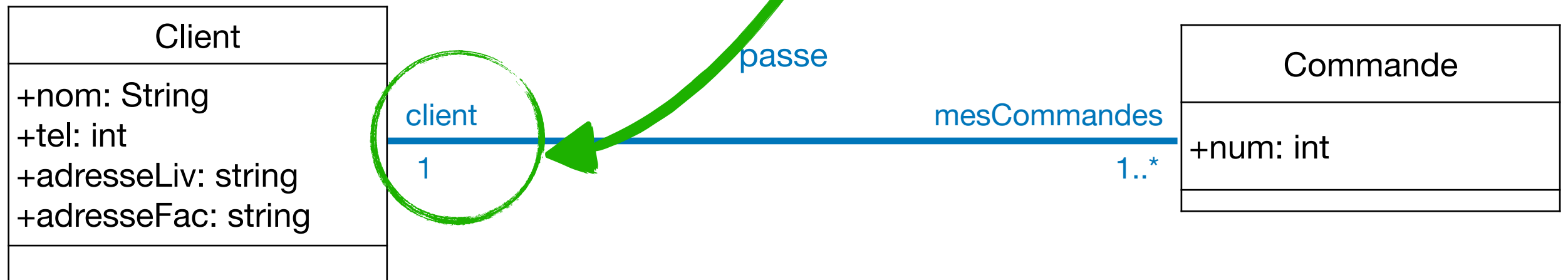
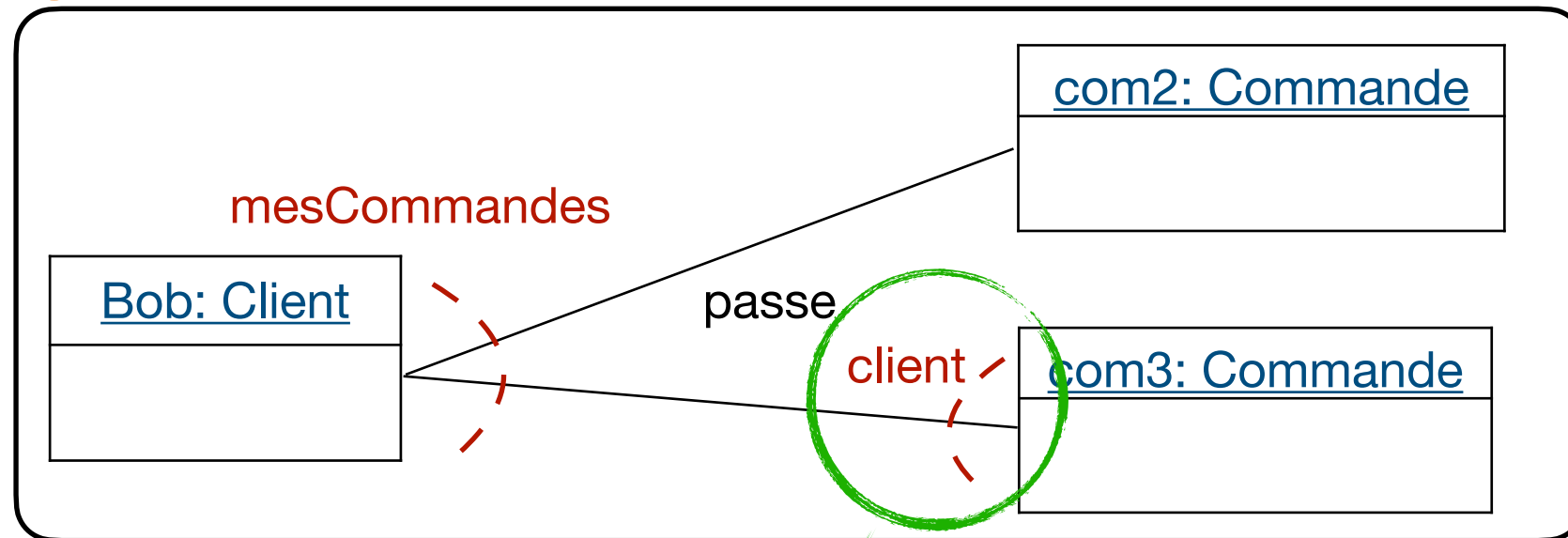
# Relation entre classes

## Association



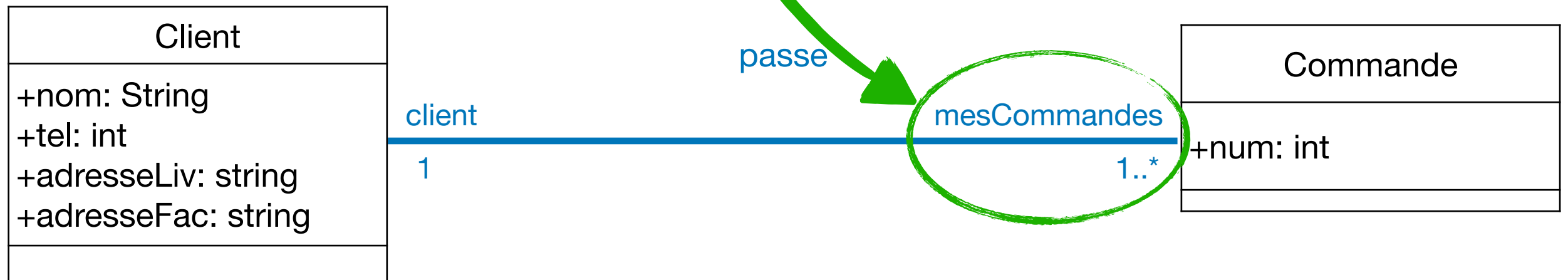
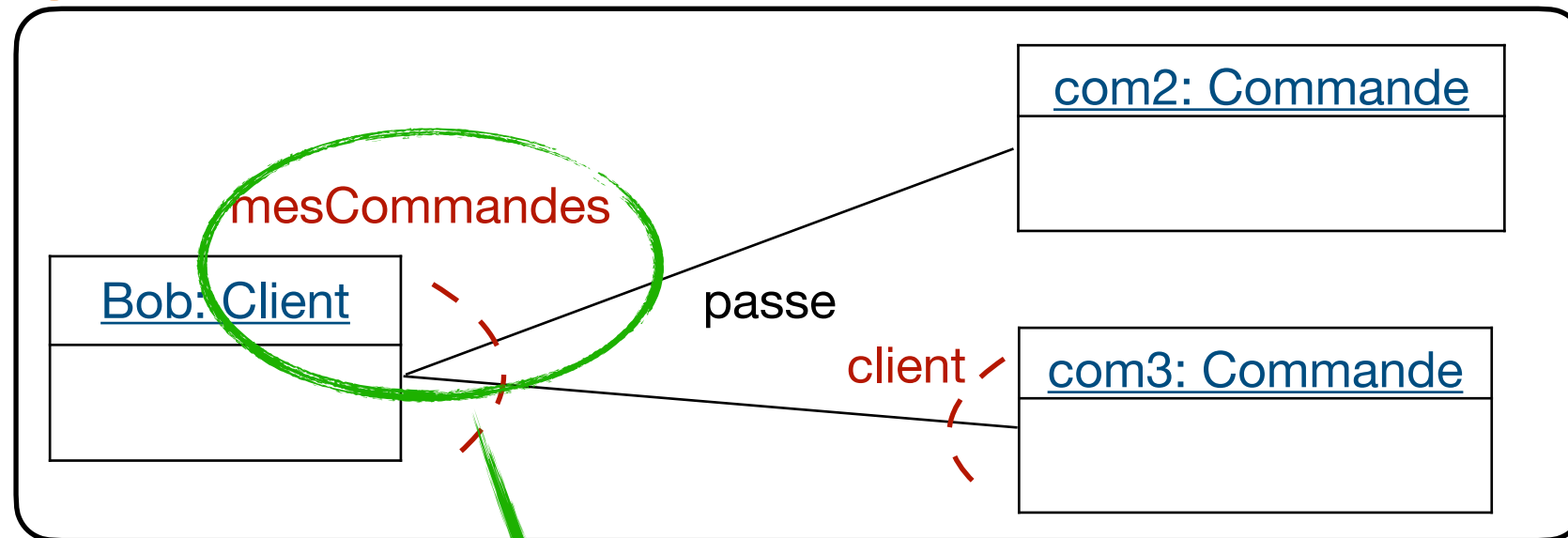
# Relation entre classes

## Association



# Relation entre classes

## Association





# Relation entre classes

## Types des attributs

- **Types des attributs simple** : primitif, prédéfini ou énuméré



# Relation entre classes

## Types des attributs

- **Types des attributs simple** : primitif, prédéfini ou énuméré

Voiture
immat: String marque: String couleur: String nbPortes: int kilométrage: int moteur: CARBURANT propriétaire: Personne



# Relation entre classes

## Types des attributs

- **Types des attributs simple** : primitif, prédéfini ou énuméré

<<enumeration>> CARBURANT
Diesel Essence GPL Electrique

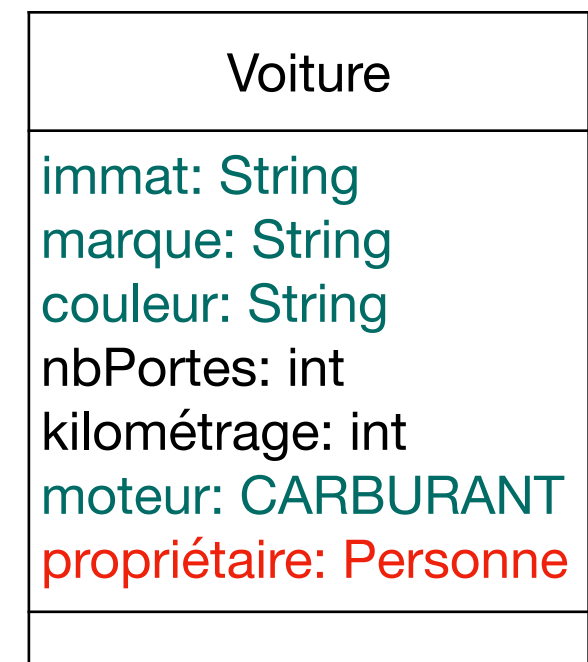
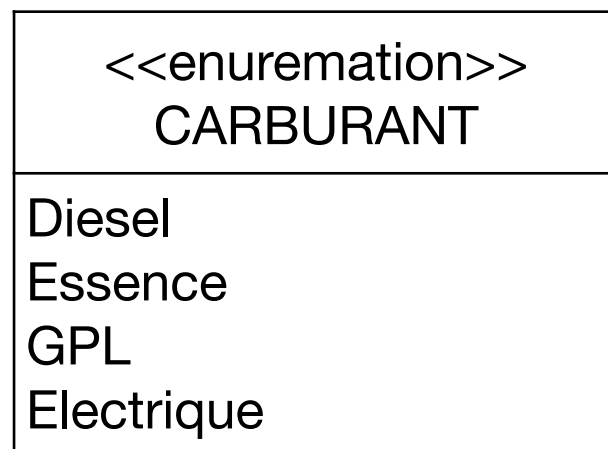
Voiture
immat: String marque: String couleur: String nbPortes: int kilométrage: int moteur: CARBURANT propriétaire: Personne



# Relation entre classes

## Types des attributs

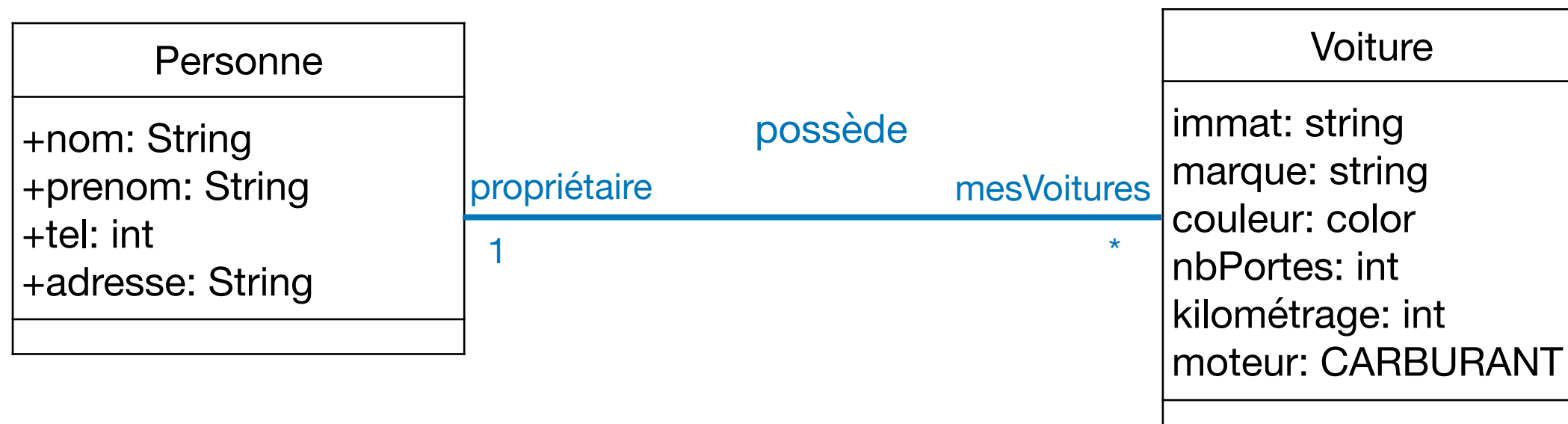
- **Types des attributs simple** : primitif, prédéfini ou énuméré



# Relation entre classes

## Types des attributs

- **Types des attributs simple** : primitif, prédéfini ou énuméré



# Associations

## Multiplicités



# Associations

## Multiplicités



Exactement n



Exactement n, m ou k

# Associations

## Multiplicités



Exactement n



Exactement n, m ou k



Entre n et m



# Associations

## Multiplicités



Exactement n



Exactement n, m ou k



Entre n et m



Au moins n

# Associations

## Multiplicités



Exactement n



Exactement n, m ou k



Entre n et m



Au moins n

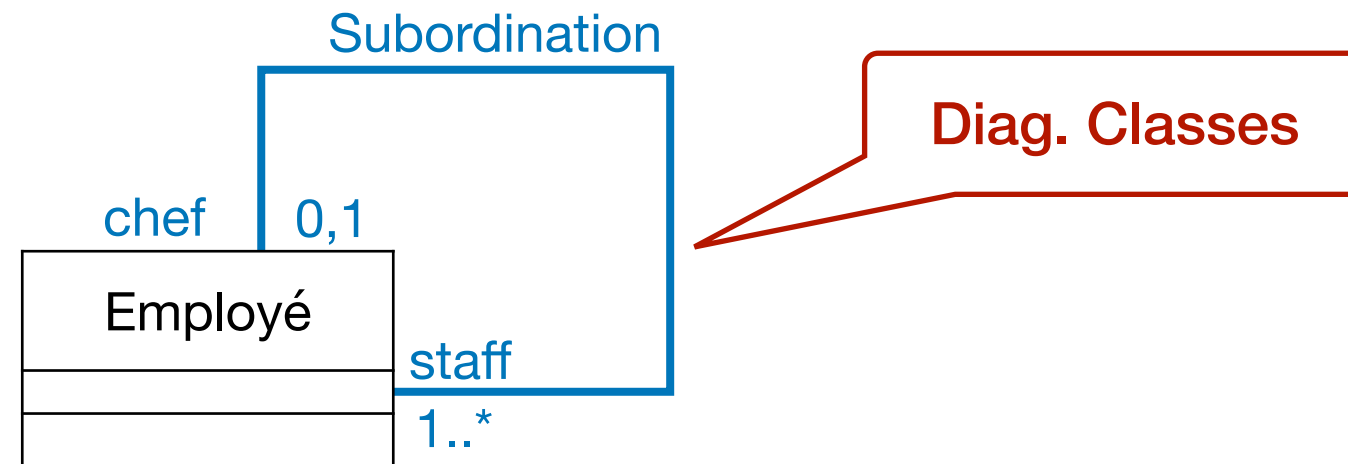


Plusieurs

A

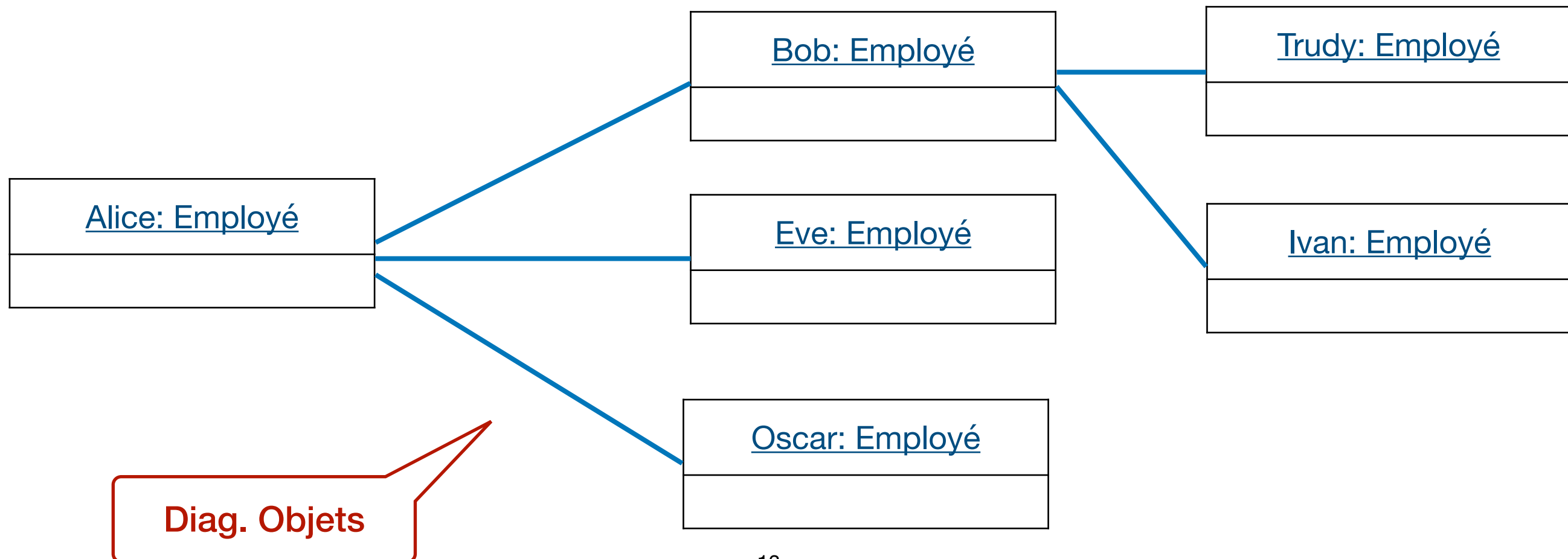
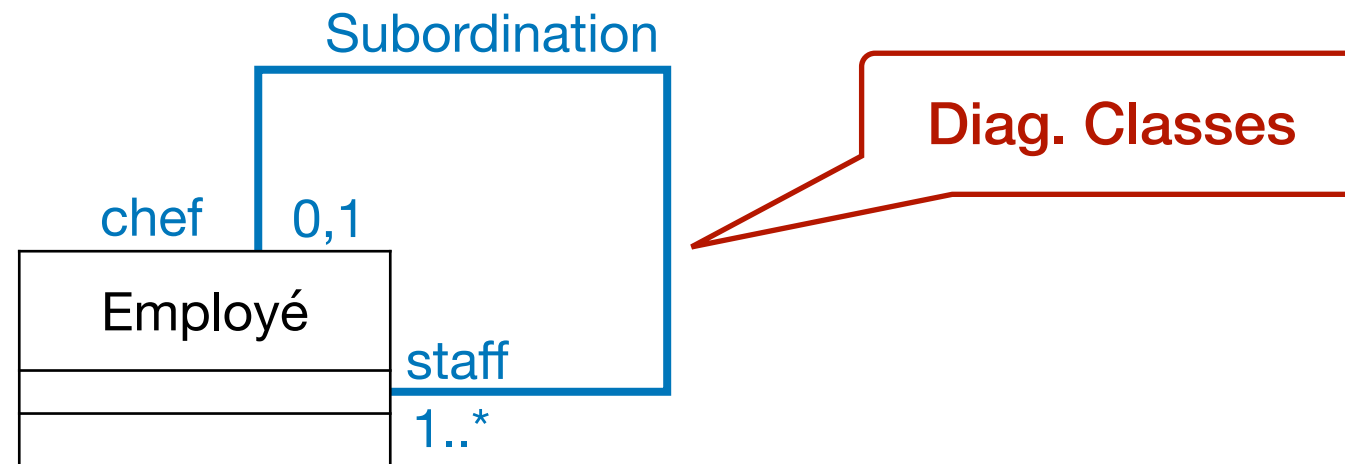
# Associations

## Réflexives



# Associations

## Réflexives



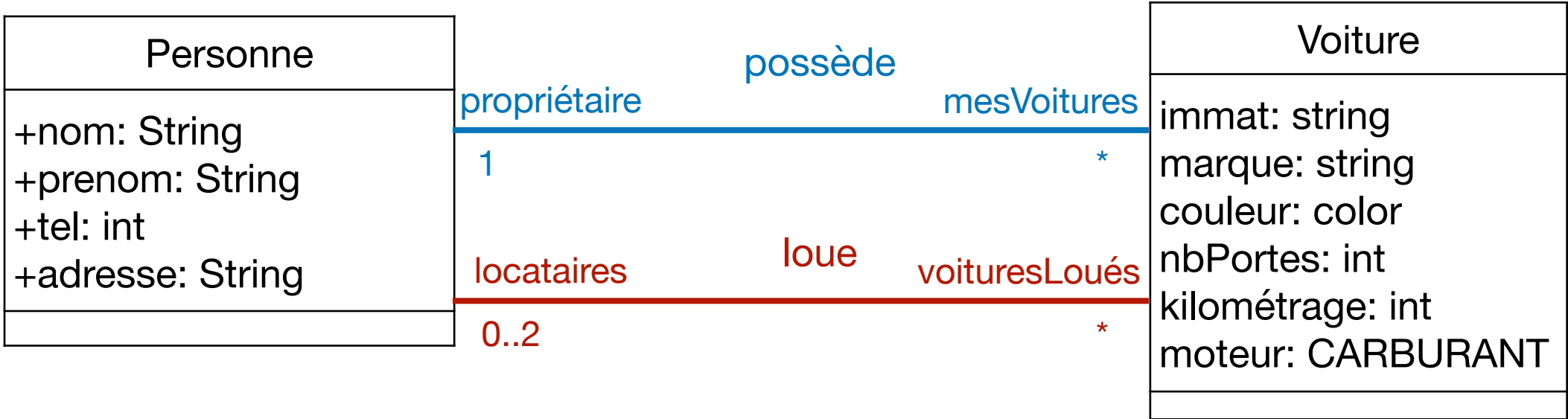
# Associations

## Association Multiple



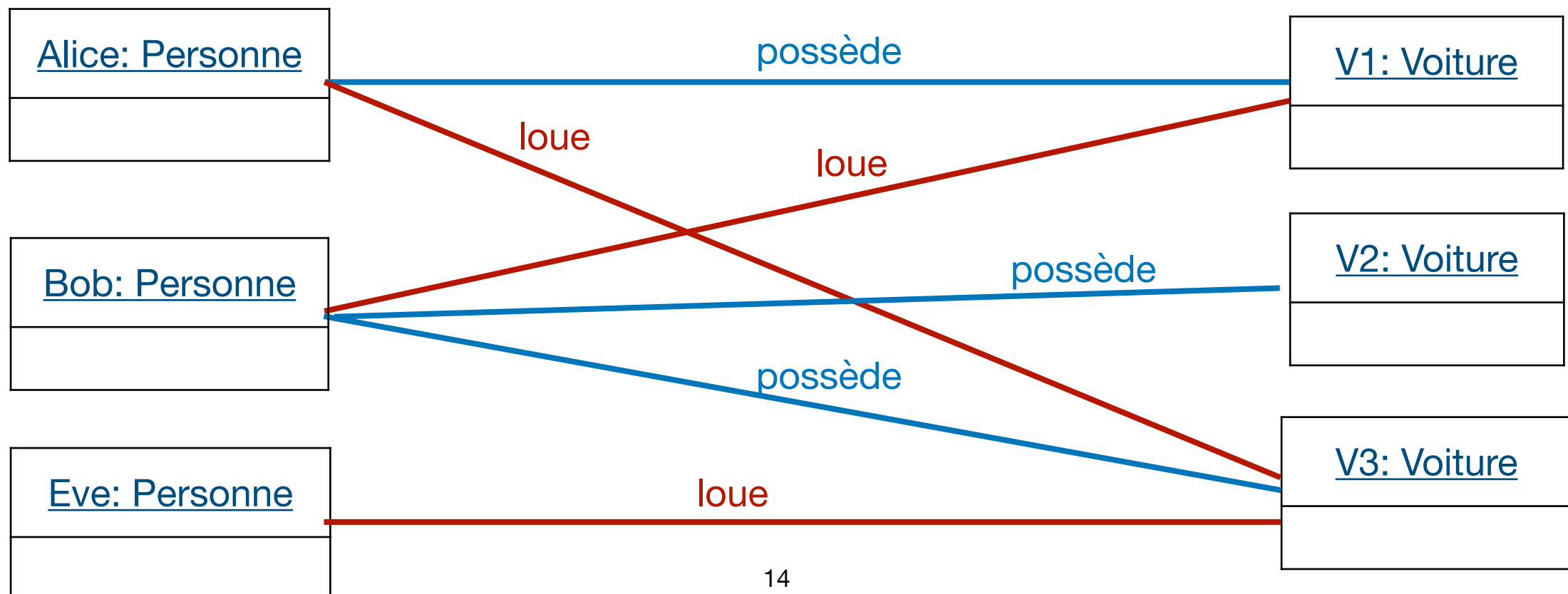
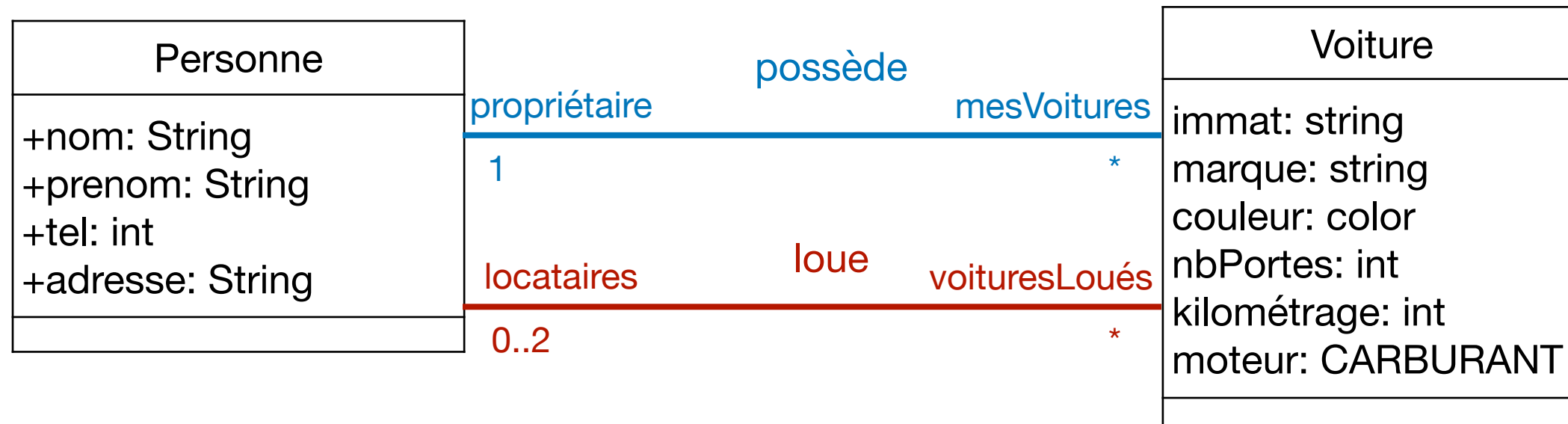
# Associations

## Association Multiple



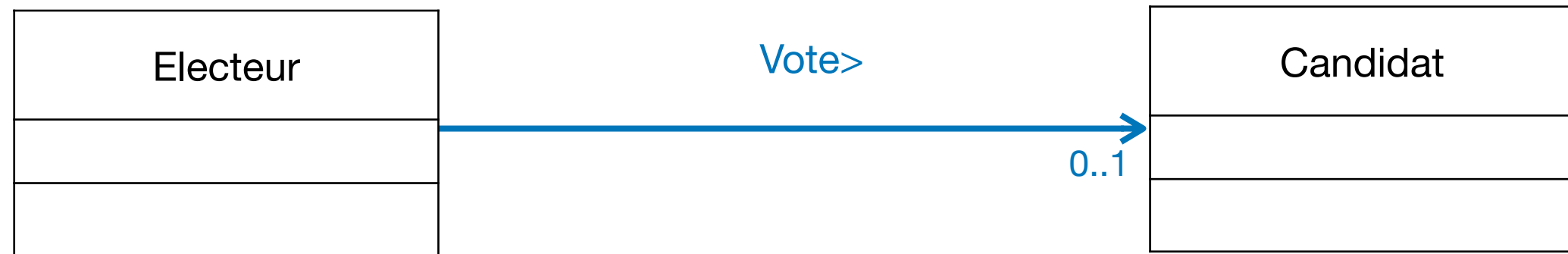
# Associations

## Association Multiple



# Associations

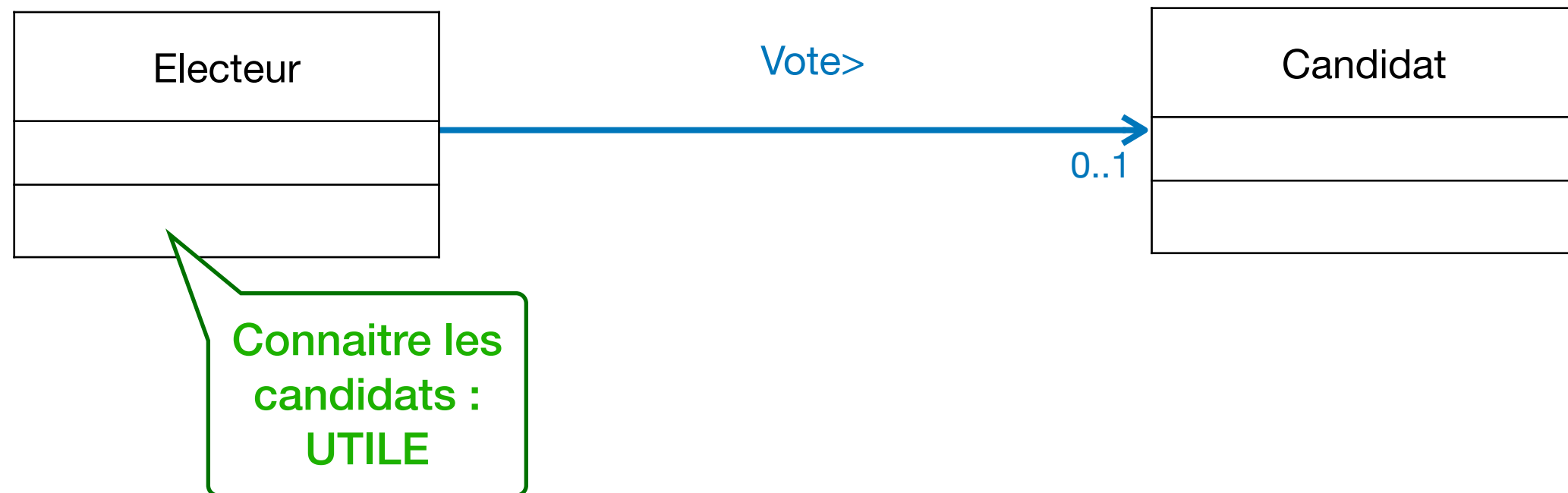
## La navigation





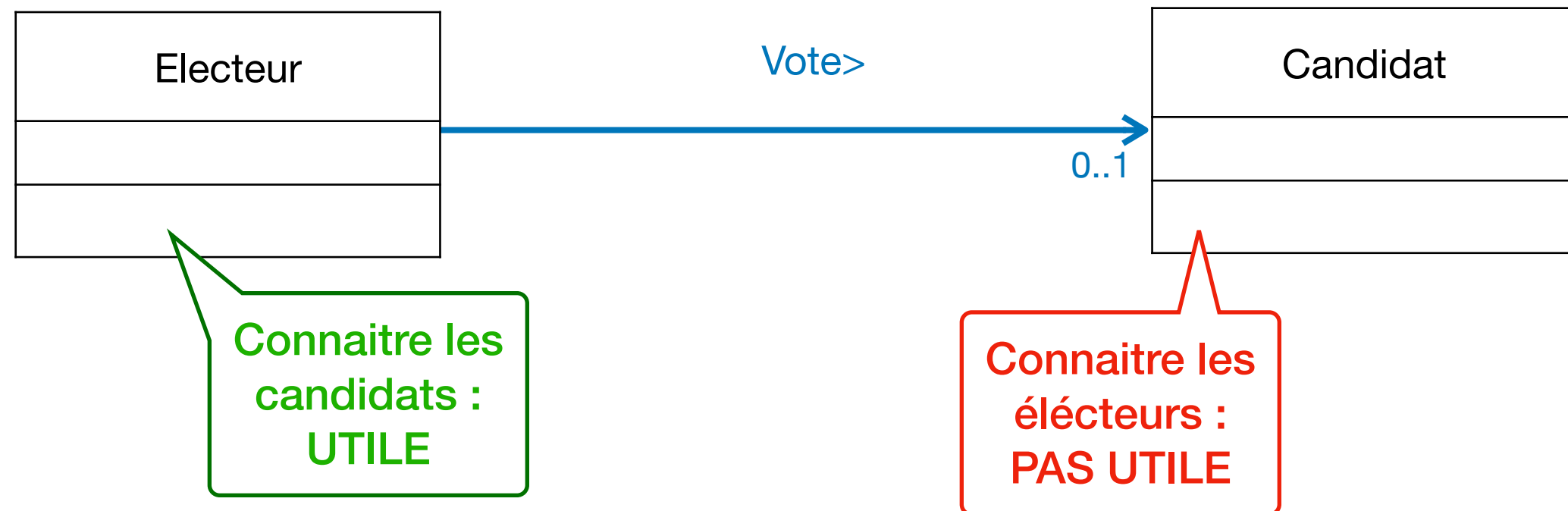
# Associations

## La navigation



# Associations

## La navigation



# Classe-Association

## Diagramme de classes



# Classe-Association

## Diagramme de classes



Date de location ?

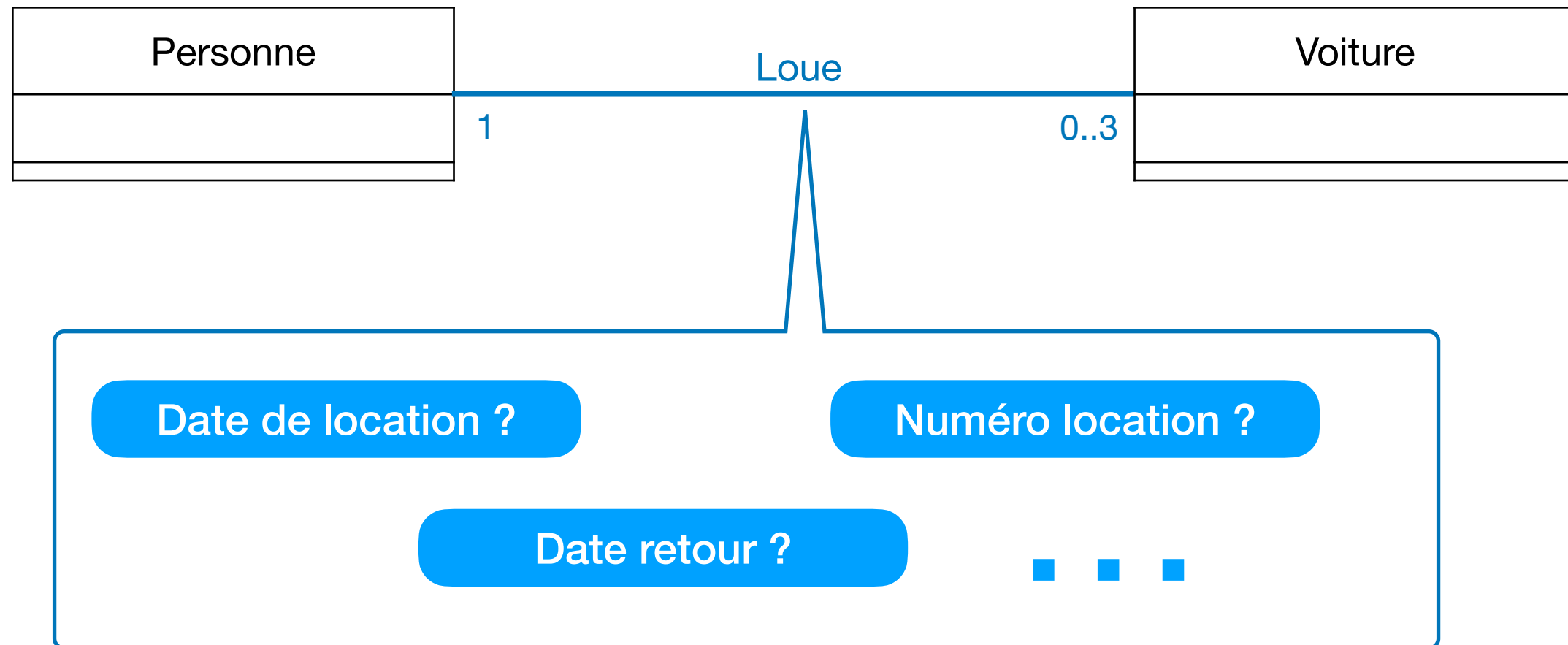
Numéro location ?

Date retour ?



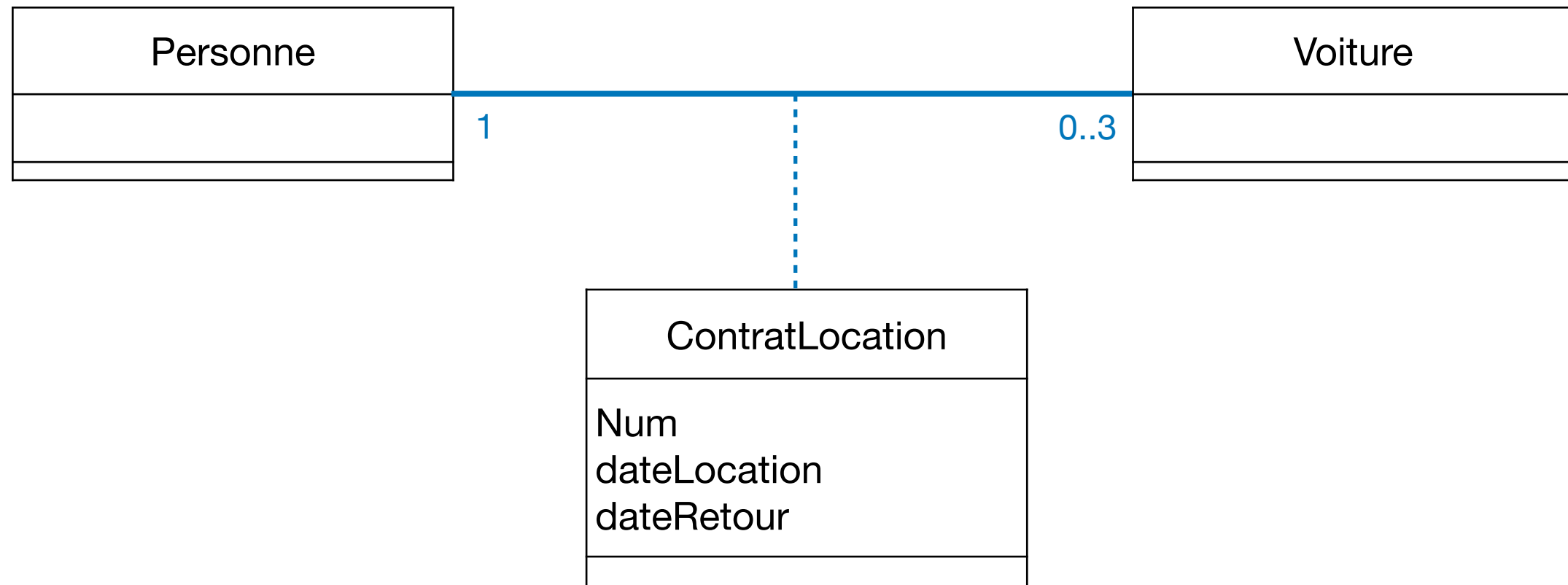
# Classe-Association

## Diagramme de classes



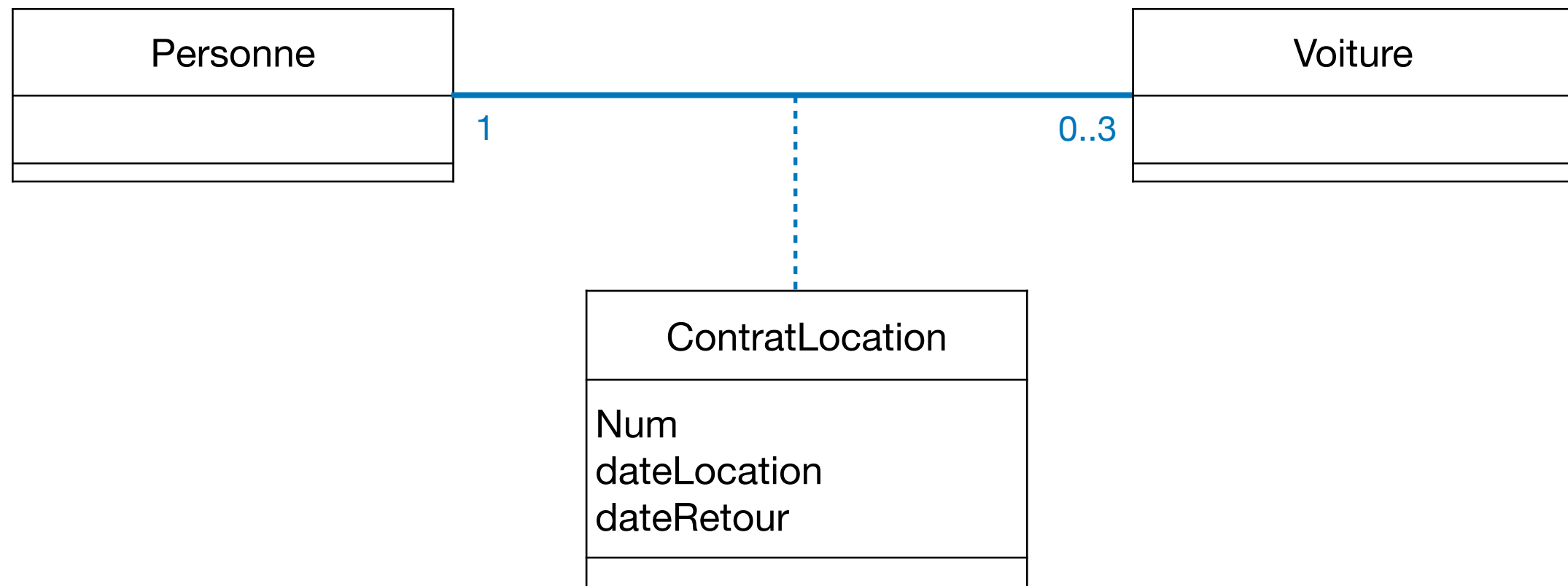
# Classe-Association

## Diagramme de classes

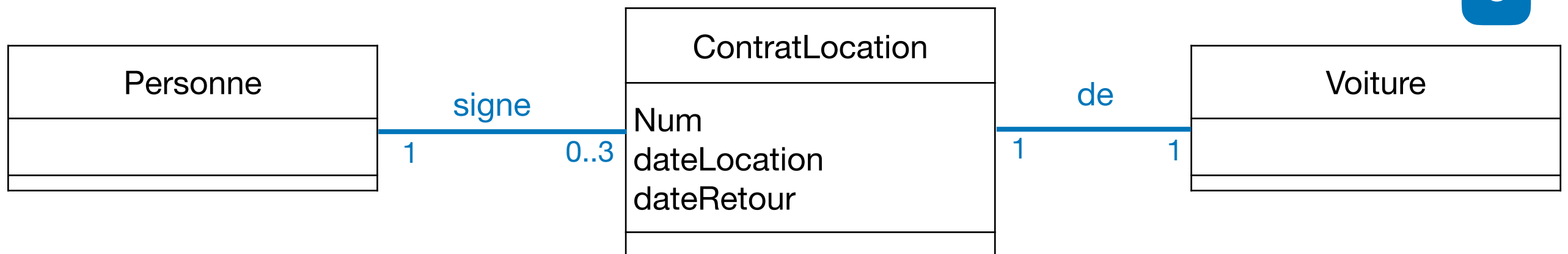


# Classe-Association

## Diagramme de classes



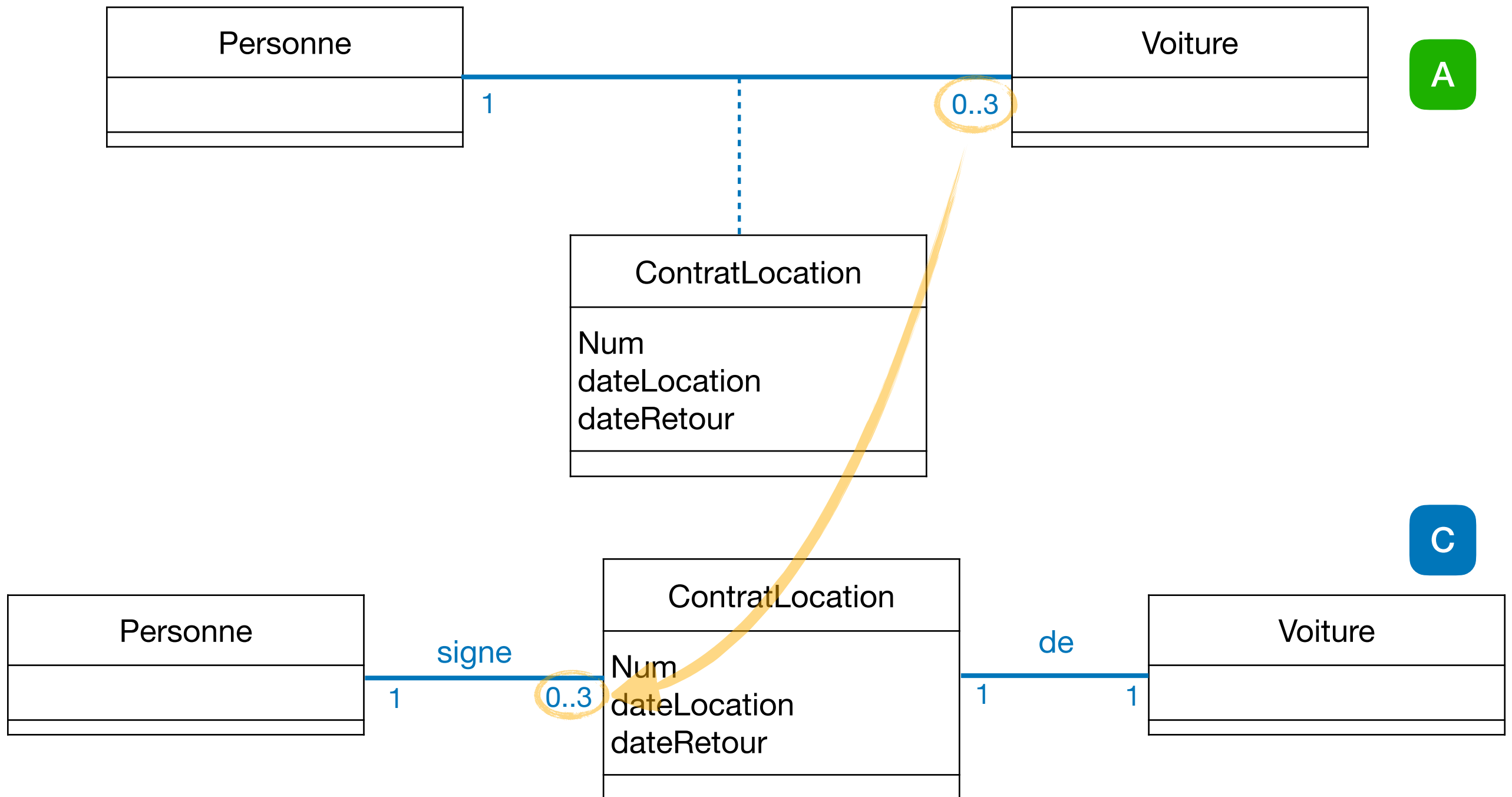
A



C

# Classe-Association

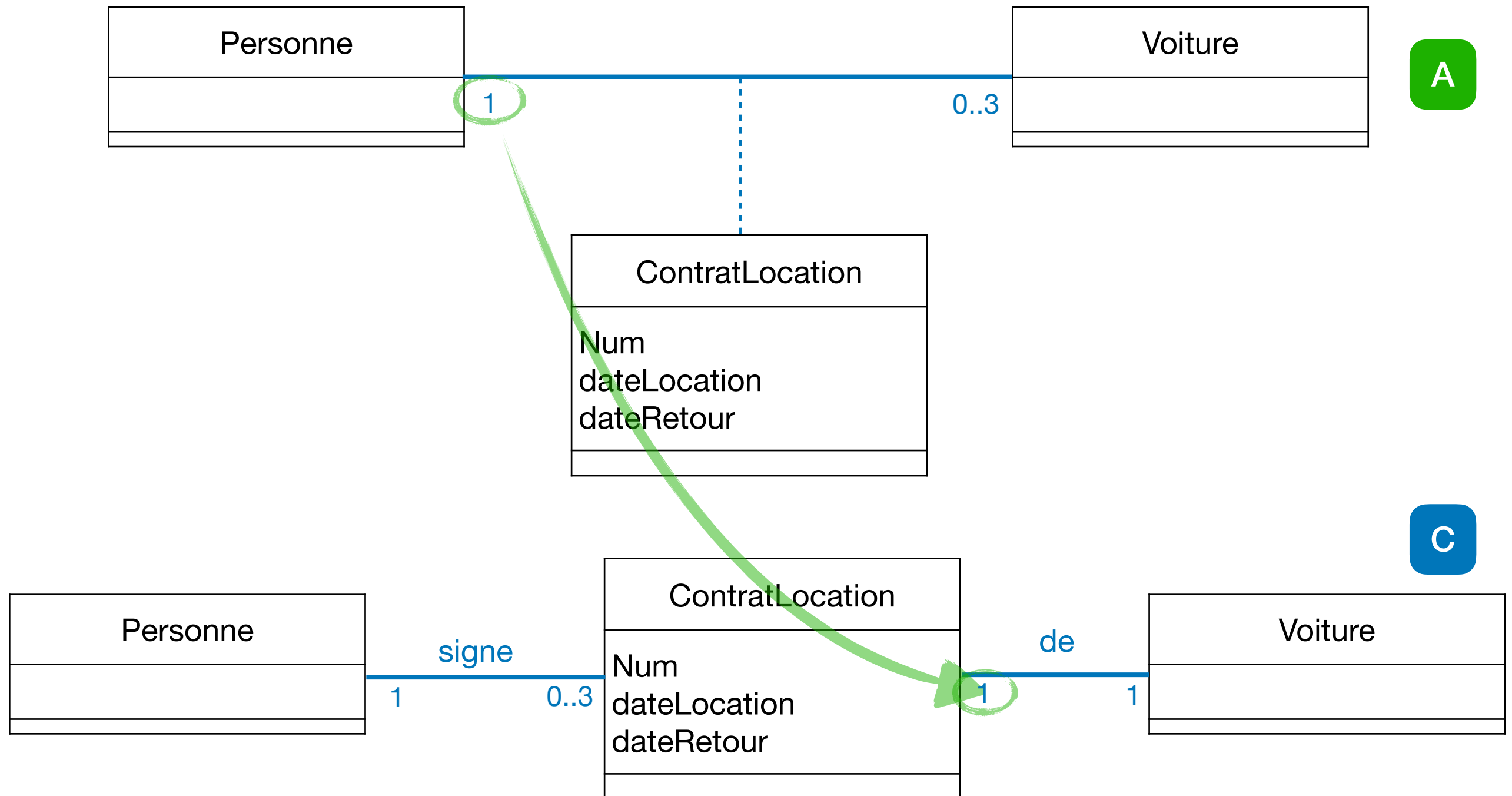
## Diagramme de classes





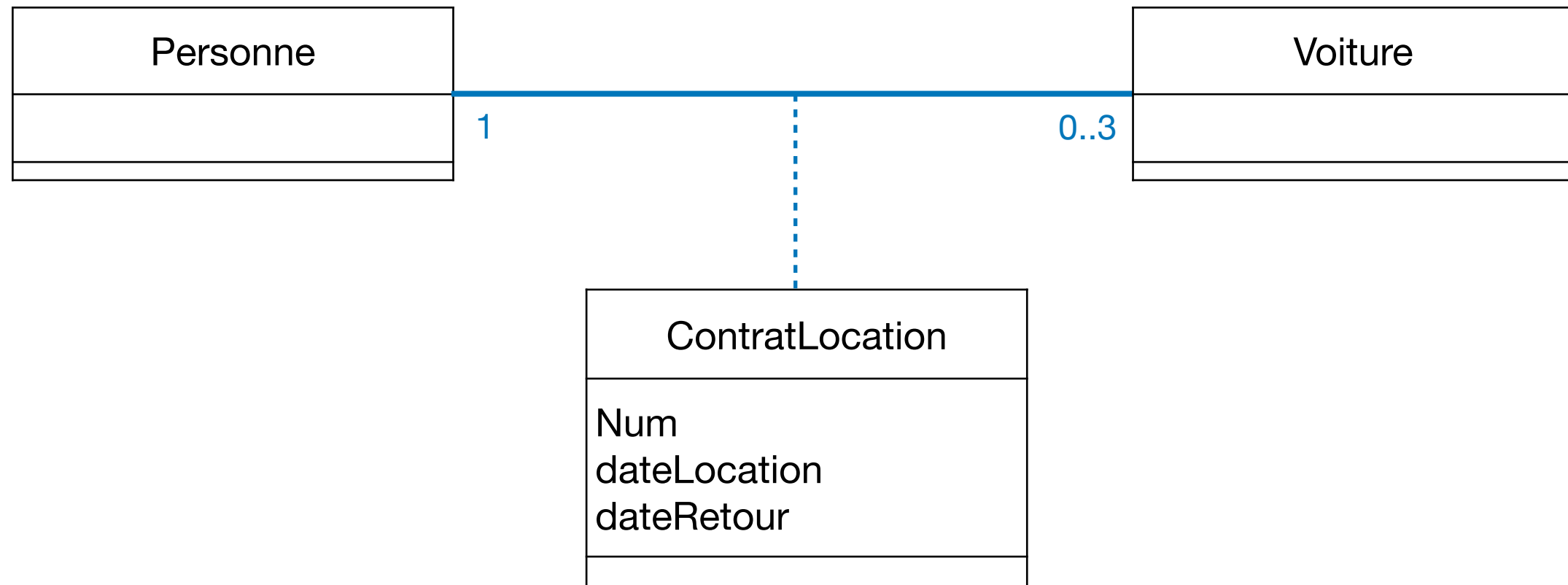
# Classe-Association

## Diagramme de classes

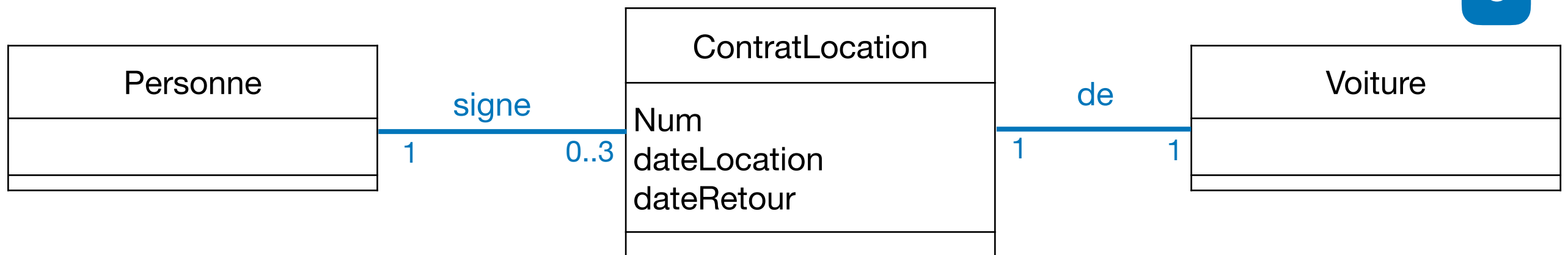


# Classe-Association

## Diagramme de classes



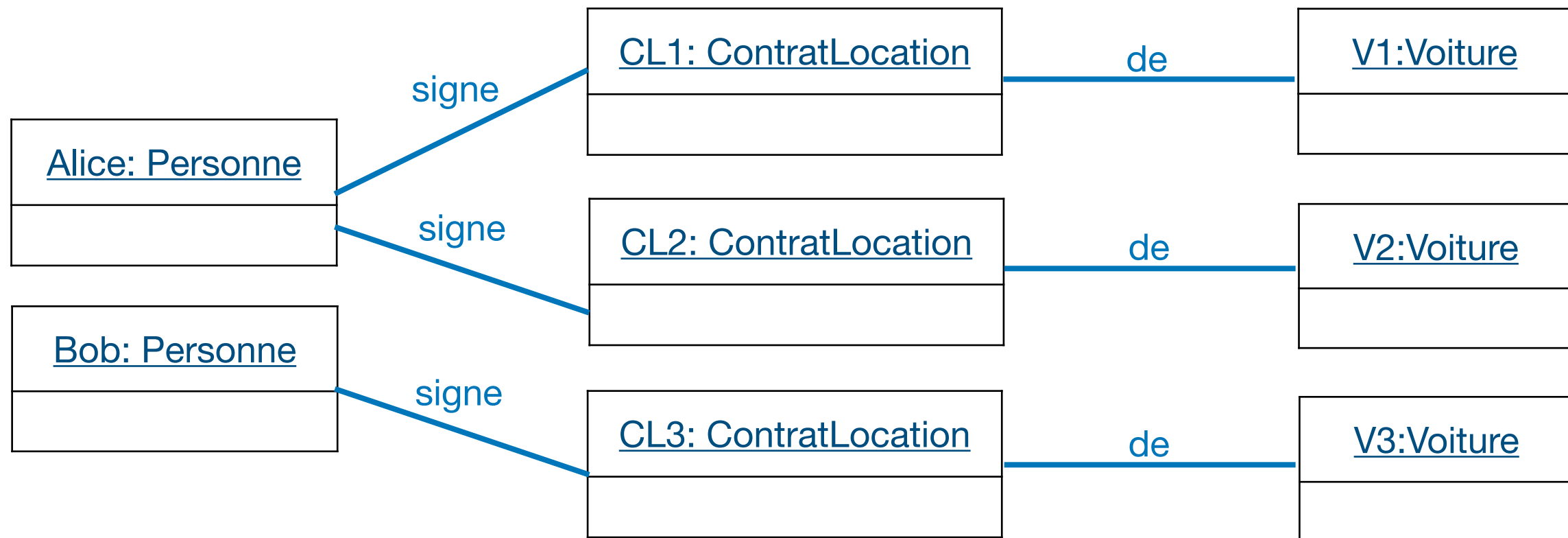
A



C

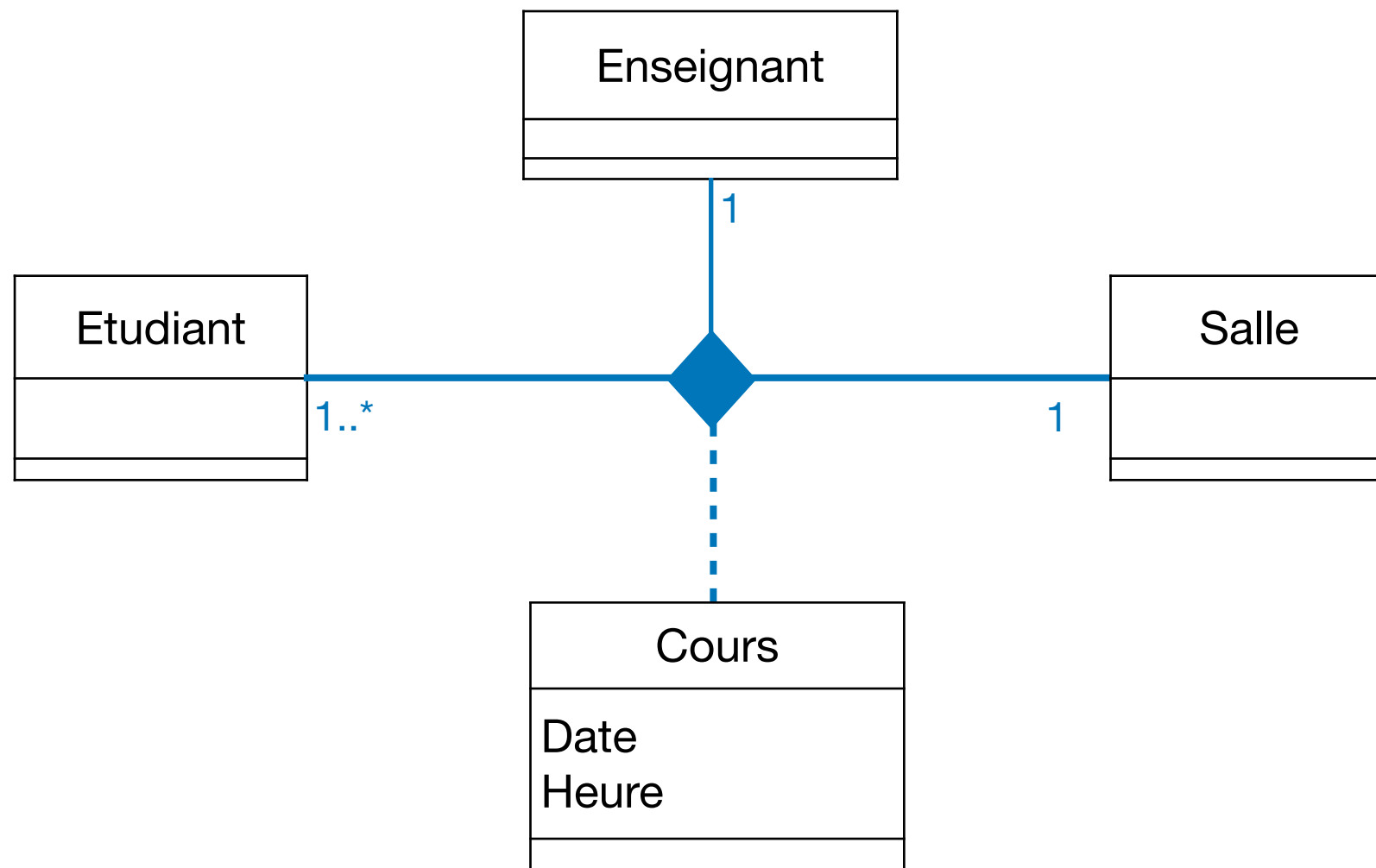
# Classe-Association

## Diagramme d'objets



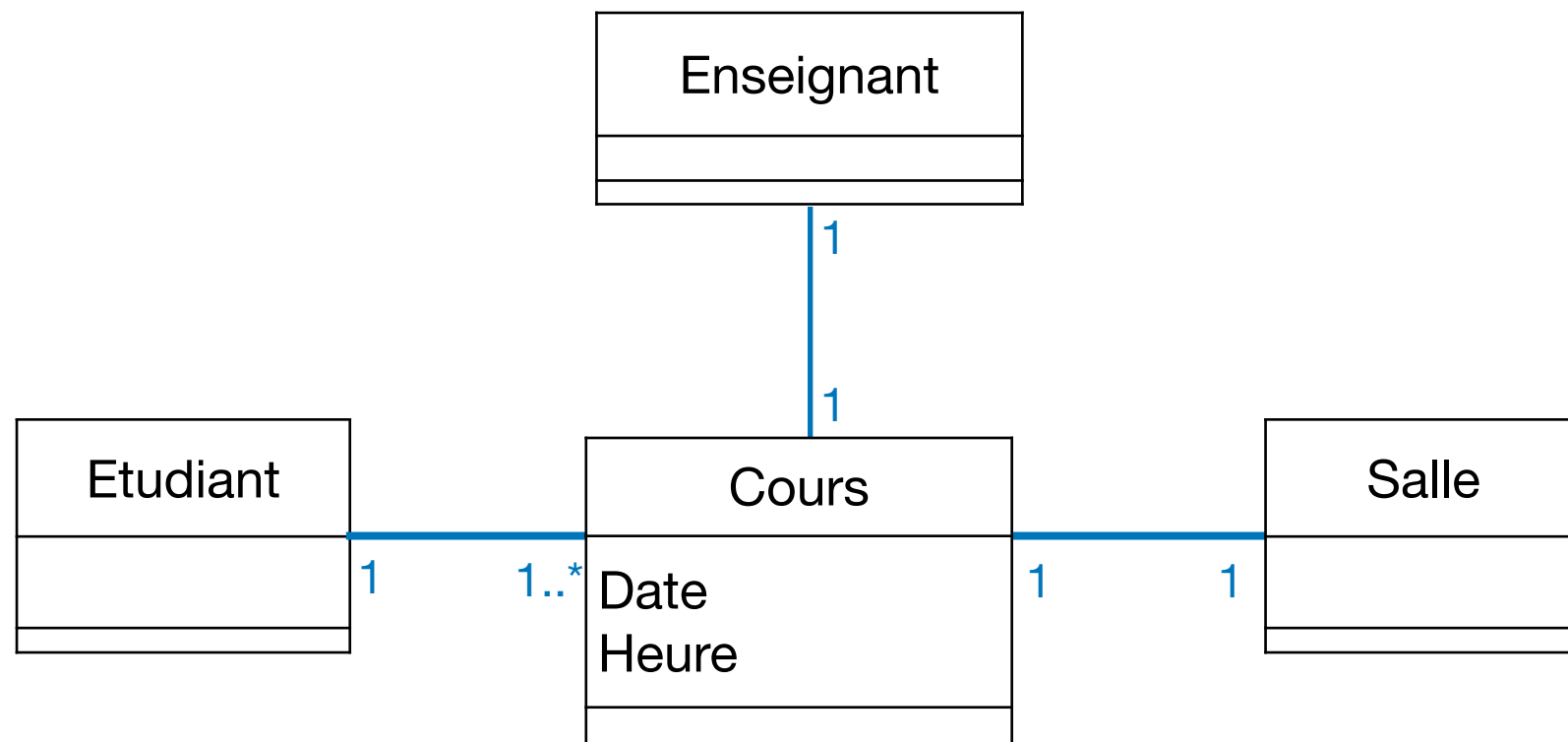
# Association n-aire

## Diagramme de classes



# Association n-aire

## Diagramme de classes



# **Mais aussi...**

**voir concepts OO (cours2)**

- **La spécialisation et la généralisation (concept n°5)**
- **Les classes abstraites et concrètes (concept n°7)**
- **La composition et l'agrégation (concept n°9)**

# Classes

## Notations avancées



Commande
+ num: int - montantHC : float - /montantTTC : float ~ tauxTVA : float - <u>nbCommandes</u> : int
+ calculerMontantTTC() : float + <u>getNbCommandes()</u> : int # getTaxe(): float

# Classes

## Notations avancées



+ : accès public

Commande
+ num: int - montantHC : float - /montantTTC : float ~ tauxTVA : float - <u>nbCommandes</u> : int
+ calculerMontantTTC() : float + <u>getNbCommandes()</u> : int # getTaxe(): float



# Classes

## Notations avancées



- : accès privé

Commande
+ num: int - montantHC : float - /montantTTC : float ~ tauxTVA : float - <u>nbCommandes</u> : int
+ calculerMontantTTC() : float + <u>getNbCommandes()</u> : int # getTaxe(): float

# Classes

## Notations avancées



~ : accès paquet

Commande
+ num: int - montantHC : float - /montantTTC : float ~ tauxTVA : float - <u>nbCommandes</u> : int
+ calculerMontantTTC() : float + <u>getNbCommandes()</u> : int # getTaxe(): float

# Classes

## Notations avancées

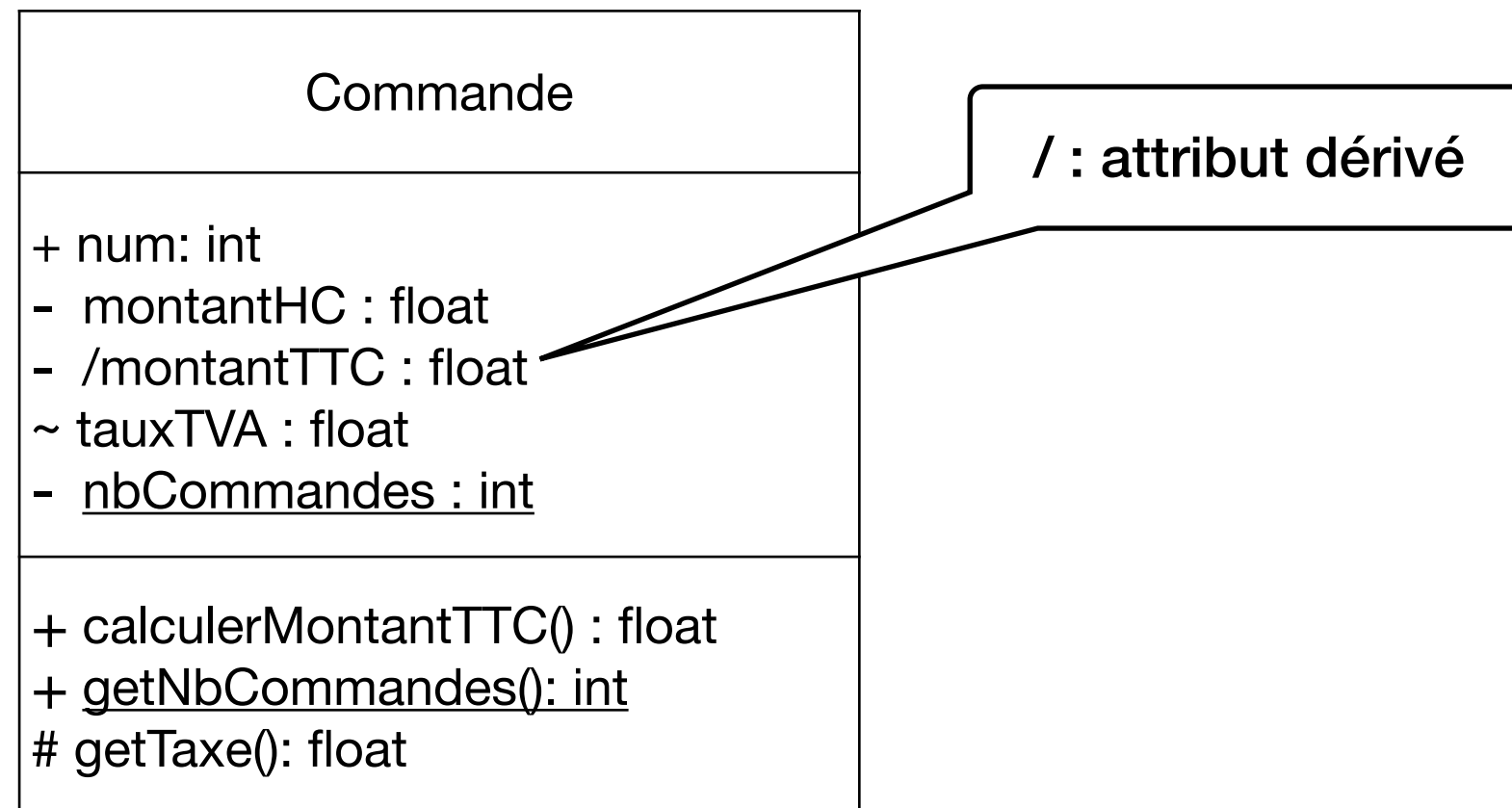


Commande	
<div>+ num: int</div> <div>- montantHC : float</div> <div>- /montantTTC : float</div> <div>~ tauxTVA : float</div> <div>- <u>nbCommandes</u> : int</div>	
<div>+ calculerMontantTTC() : float</div> <div>+ <u>getNbCommandes()</u>: int</div> <div># getTaxe(): float</div>	

# : accès protégé

# Classes

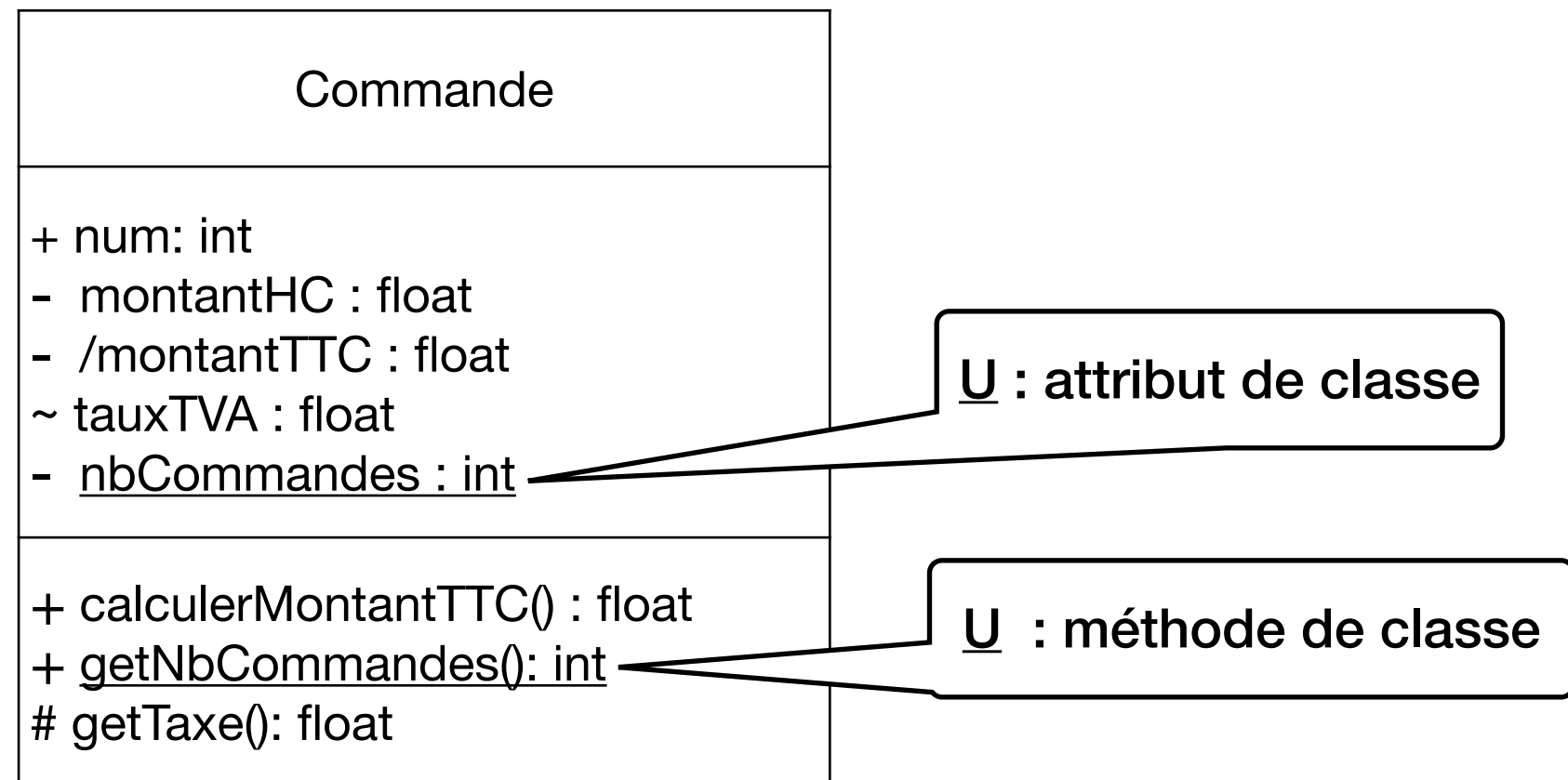
## Notations avancées



# Classes

## Notations avancées

C



# Relation de dépendance

## Différents stéréotypes



# Relation de dépendance

## Différents stéréotypes



- <<use>> : **Utilisateur** utilise la **Ressource** (stéréotype général)



# Relation de dépendance

## Différents stéréotypes



- <<use>> : **Utilisateur** utilise la **Ressource** (stéréotype général)
- <<call>> : **Utilisateur** invoque une méthode de la **Ressource**





# Relation de dépendance

## Différents stéréotypes



- <<use>> : **Utilisateur** utilise la **Ressource** (stéréotype général)
- <<call>> : **Utilisateur** invoque une méthode de la **Ressource**
- <<create>> : **Utilisateur** crée une instance de la **Ressource**

A

C

# Relation de dépendance

## Différents stéréotypes



- `<<use>>` : **Utilisateur** utilise la **Ressource** (stéréotype général)
- `<<call>>` : **Utilisateur** invoque une méthode de la **Ressource**
- `<<create>>` : **Utilisateur** crée une instance de la **Ressource**
- `<<instantiate>>` : **Utilisateur** est une fabrique de la **Ressource**

# Relation de dépendance

## Différents stéréotypes



- `<<use>>` : **Utilisateur** utilise la **Ressource** (stéréotype général)
- `<<call>>` : **Utilisateur** invoque une méthode de la **Ressource**
- `<<create>>` : **Utilisateur** crée une instance de la **Ressource**
- `<<instantiate>>` : **Utilisateur** est une fabrique de la **Ressource**
- `<<permit>>` : **Utilisateur** peut accéder à une partie ou à la totalité des éléments privés de la **Ressource**

# Relation de dépendance

## Différents stéréotypes



- <<use>> : **Utilisateur** utilise la **Ressource** (stéréotype général)
- <<call>> : **Utilisateur** invoque une méthode de la **Ressource**
- <<create>> : **Utilisateur** crée une instance de la **Ressource**
- <<instantiate>> : **Utilisateur** est une fabrique de la **Ressource**
- <<permit>> : **Utilisateur** peut accéder à une partie ou à la totalité des éléments privés de la **Ressource**
- <<derive>> : **Utilisateur** est un objet dérivé de la **Ressource**

# Relation de dépendance

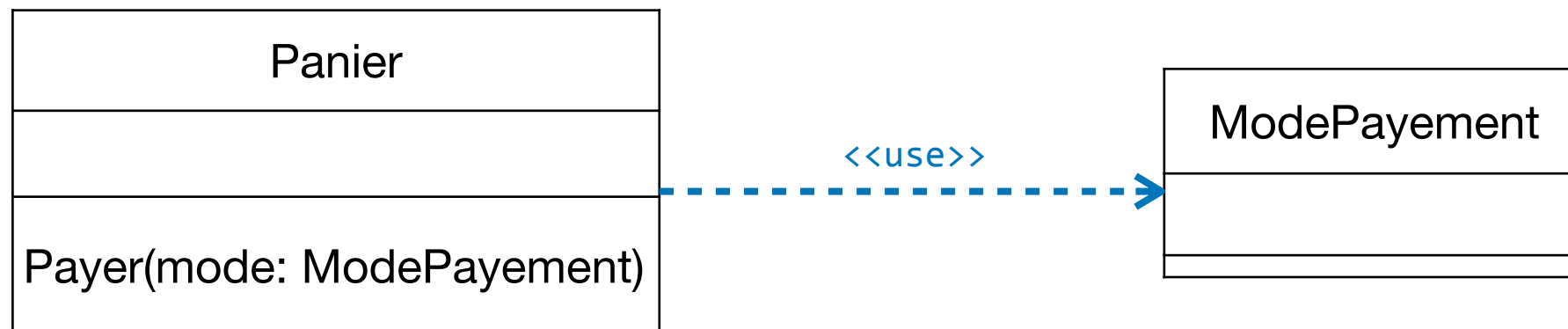
## Différents stéréotypes



- <<use>> : **Utilisateur** utilise la **Ressource** (stéréotype général)
- <<call>> : **Utilisateur** invoque une méthode de la **Ressource**
- <<create>> : **Utilisateur** crée une instance de la **Ressource**
- <<instantiate>> : **Utilisateur** est une fabrique de la **Ressource**
- <<permit>> : **Utilisateur** peut accéder à une partie ou à la totalité des éléments privés de la **Ressource**
- <<derive>> : **Utilisateur** est un objet dérivé de la **Ressource**
- <<refine>> : **Utilisateur** est une spécialisation de **Ressource** (rapidement remplacée par une spécialisation dans le diagramme)

# Relation de dépendance

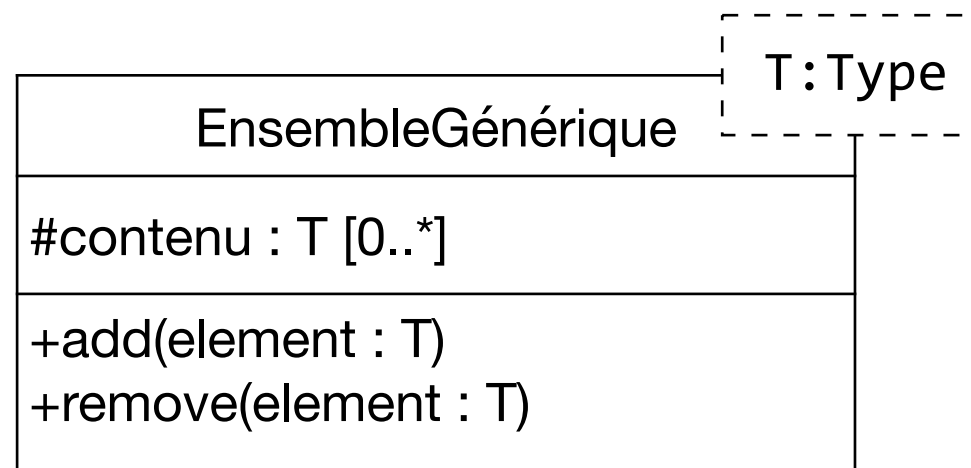
## Exemple



# Classes génériques

## Classes template

- Une classe template est un modèle de classes
- Des classes génériques avec des paramètres



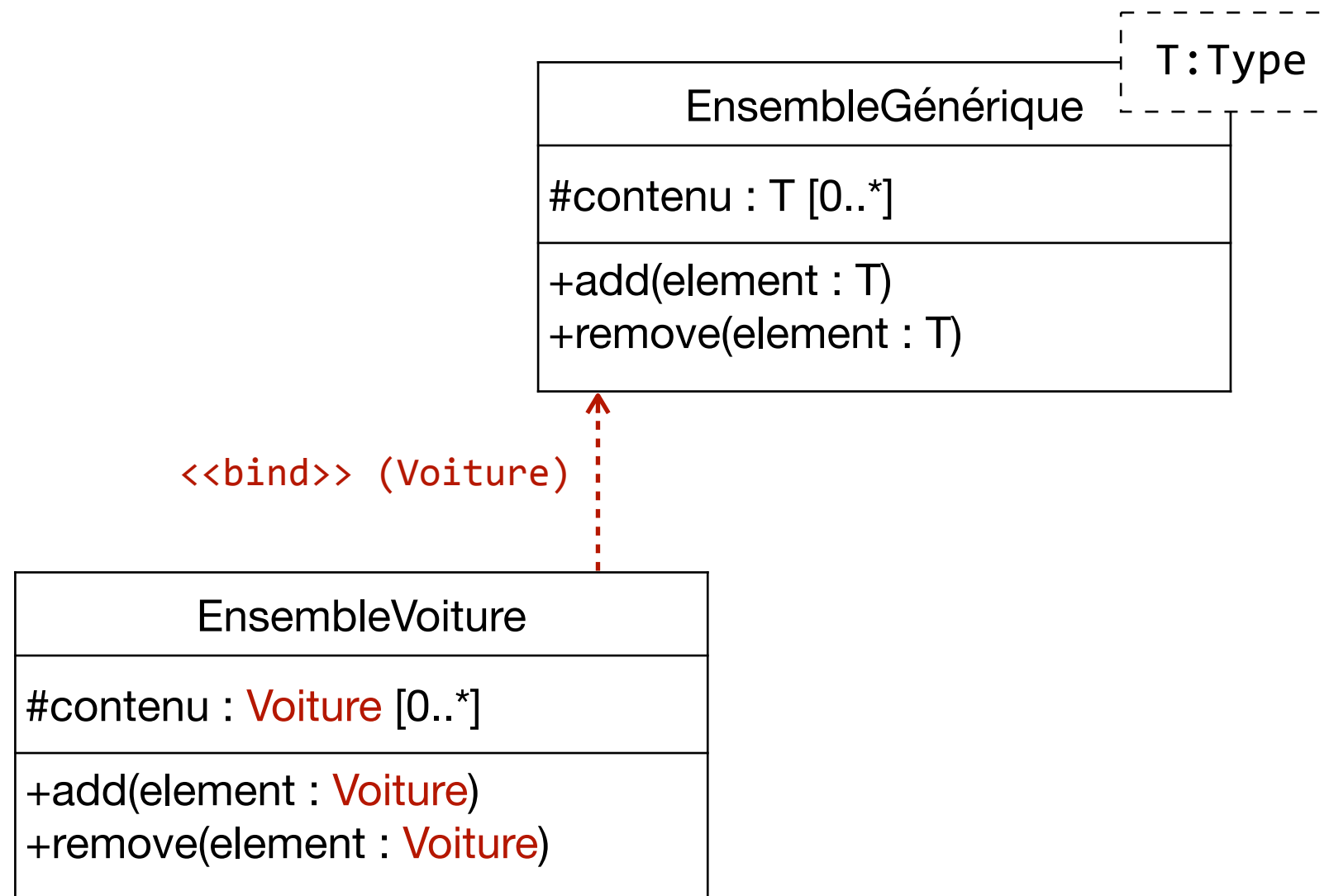
A

C

# Classes génériques

## Classes template

- Une classe template est un modèle de classes
- Des classes génériques avec des paramètres



A

C



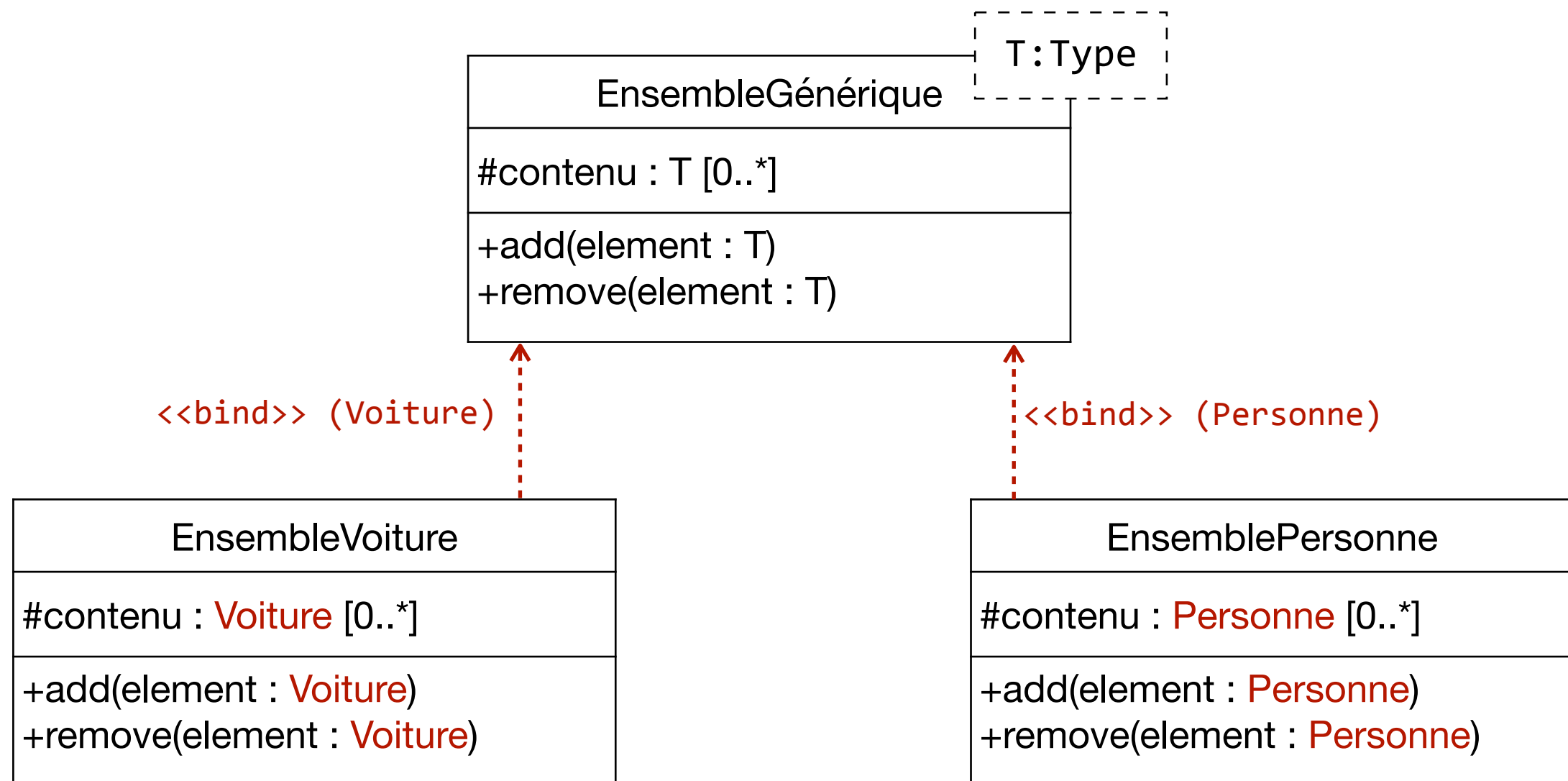
# Classes génériques

## Classes template

- Une classe template est un modèle de classes
- Des classes génériques avec des paramètres

A

C



# Interface

## Relation de réalisation



# Interface

## Relation de réalisation

- Classes sans instance :
  - **Classe abstraite** : certaines méthodes sont abstraites
  - **Interface** : pas d'attributs et toutes les méthodes sont abstraites

A

C

# Interface

## Relation de réalisation

- Classes sans instance :
  - **Classe abstraite** : certaines méthodes sont abstraites
  - **Interface** : pas d'attributs et toutes les méthodes sont abstraites
- Interface :
  - Pas d'attributs

A

C

# Interface

## Relation de réalisation

- Classes sans instance :
  - **Classe abstraite** : certaines méthodes sont abstraites
  - **Interface** : pas d'attributs et toutes les méthodes sont abstraites
- Interface :
  - Pas d'attributs
  - Méthodes abstraites

A

C

# Interface

## Relation de réalisation

- Classes sans instance :
  - **Classe abstraite** : certaines méthodes sont abstraites
  - **Interface** : pas d'attributs et toutes les méthodes sont abstraites
- Interface :
  - Pas d'attributs
  - Méthodes abstraites
  - Doit être réalisée par des classes concrètes

A

C

# Interface

## Relation de réalisation

- Classes sans instance :
  - **Classe abstraite** : certaines méthodes sont abstraites
  - **Interface** : pas d'attributs et toutes les méthodes sont abstraites
- Interface :
  - Pas d'attributs
  - Méthodes abstraites
  - Doit être réalisée par des classes concrètes
  - spécialisation/généralisation des interfaces

A

C

# Interface

## Relation de réalisation

- Classes sans instance :
  - **Classe abstraite** : certaines méthodes sont abstraites
  - **Interface** : pas d'attributs et toutes les méthodes sont abstraites
- Interface :
  - Pas d'attributs
  - Méthodes abstraites
  - Doit être réalisée par des classes concrètes
  - spécialisation/généralisation des interfaces
  - Une classe peut réaliser plusieurs interfaces

A

C



# Interface

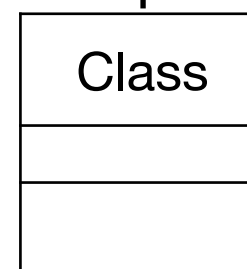
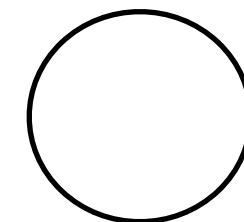
## Relation de réalisation

- Classes sans instance :
  - **Classe abstraite** : certaines méthodes sont abstraites
  - **Interface** : pas d'attributs et toutes les méthodes sont abstraites

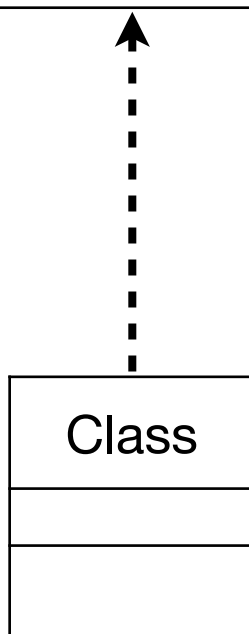
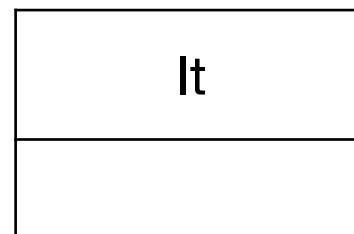


- Interface :
  - Pas d'attributs
  - Méthodes abstraites
  - Doit être réalisée par des classes concrètes
  - spécialisation/généralisation des interfaces
  - Une classe peut réaliser plusieurs interfaces

<<interface>>



<<interface>>

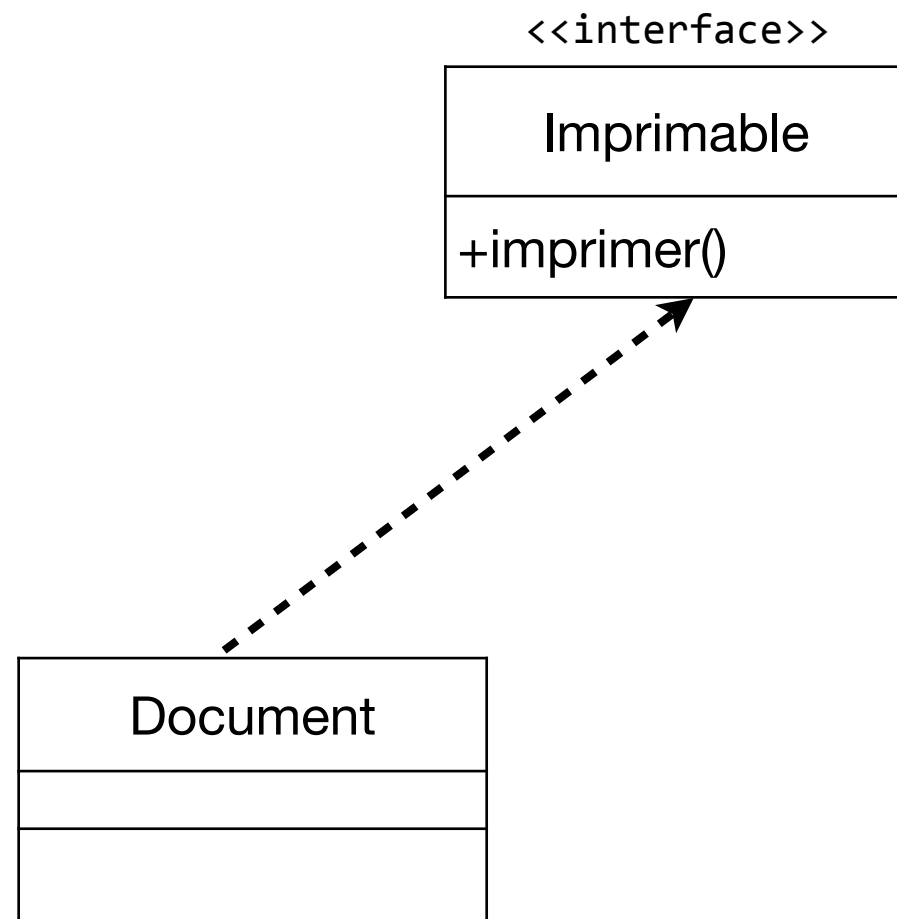


# Interface

## Exemple (1/2)

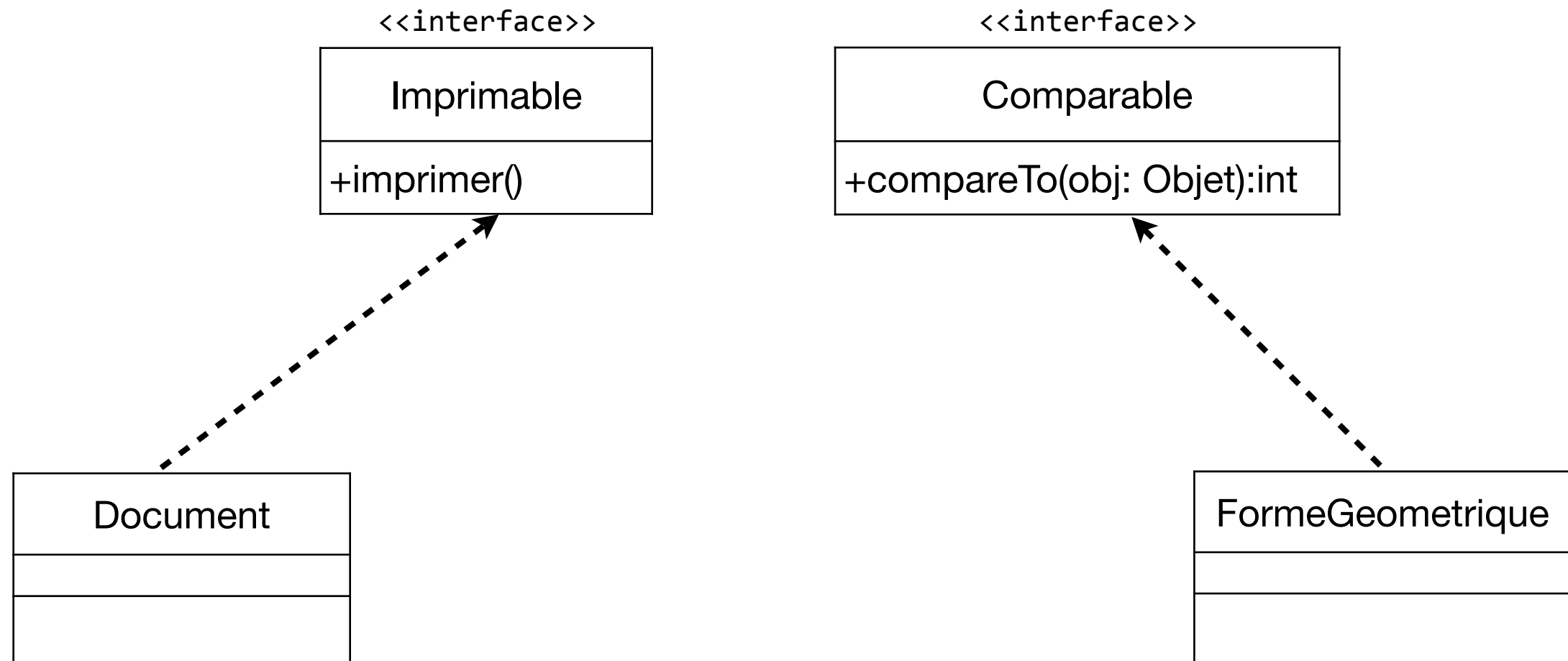
# Interface

## Example (1/2)



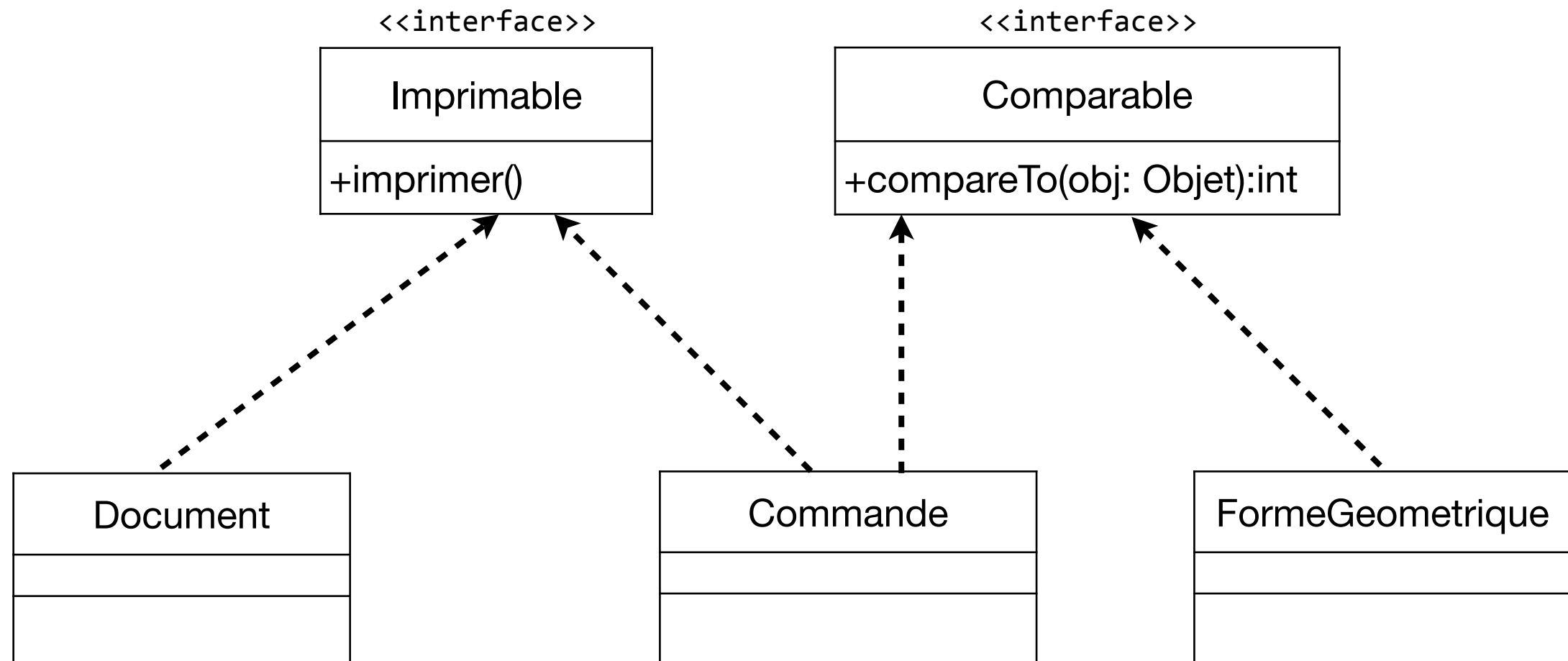
# Interface

## Example (1/2)



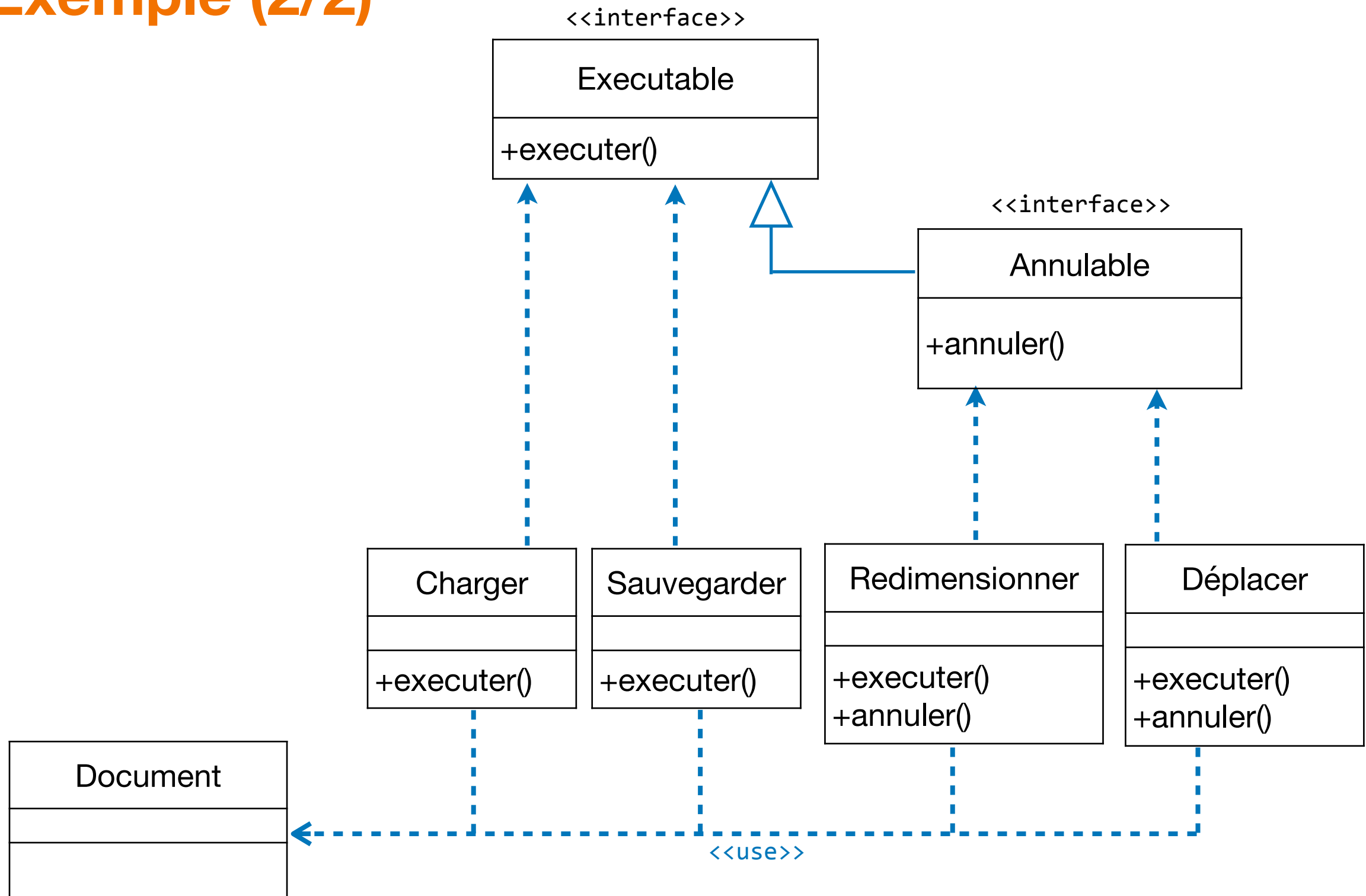
# Interface

## Example (1/2)



# Interface

## Exemple (2/2)



# Diagramme de classes

## Limitations et contraintes



# Diagramme de classes

## Limitations et contraintes



- **Limitations :**



# Diagramme de classes

## Limitations et contraintes



- **Limitations :**
  - Les contraintes sur les attributs

# Diagramme de classes

## Limitations et contraintes



- **Limitations :**
  - Les contraintes sur les attributs
  - Les contraintes sur les associations

# Diagramme de classes

## Limitations et contraintes



- **Limitations :**
  - Les contraintes sur les attributs
  - Les contraintes sur les associations
  - Les contraintes entre attributs et/ou entre associations

# Diagramme de classes

## Limitations et contraintes



- **Limitations :**
  - Les contraintes sur les attributs
  - Les contraintes sur les associations
  - Les contraintes entre attributs et/ou entre associations

# Diagramme de classes

## Limitations et contraintes



- **Limitations :**
  - Les contraintes sur les attributs
  - Les contraintes sur les associations
  - Les contraintes entre attributs et/ou entre associations
- **Contraintes :**

# Diagramme de classes

## Limitations et contraintes



- **Limitations :**
  - Les contraintes sur les attributs
  - Les contraintes sur les associations
  - Les contraintes entre attributs et/ou entre associations
- **Contraintes :**
  - Sous la forme d'un texte descriptif et/ou d'une note

# Diagramme de classes

## Limitations et contraintes



- **Limitations :**
  - Les contraintes sur les attributs
  - Les contraintes sur les associations
  - Les contraintes entre attributs et/ou entre associations
- **Contraintes :**
  - Sous la forme d'un texte descriptif et/ou d'une note
  - Utilisation de langage formel associé à UML : OCL (Object Constraint Language)

# Contraintes

## Contraintes sur les attributs

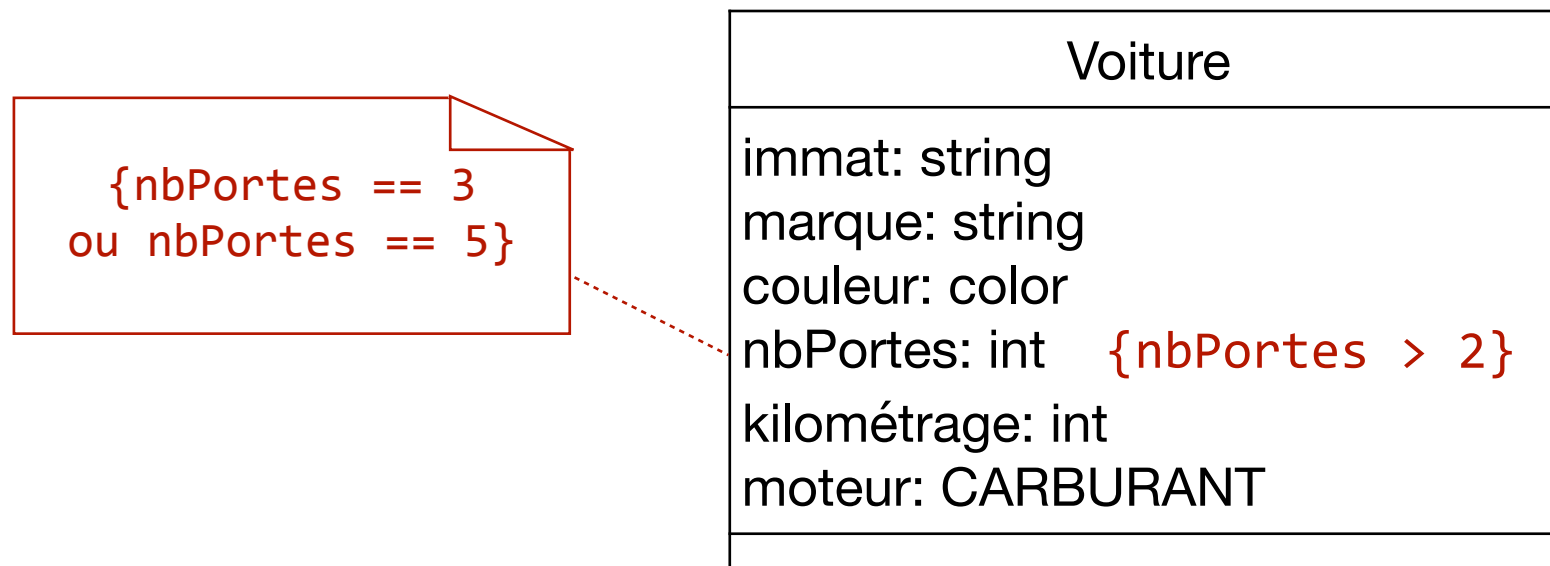


Voiture
immat: string marque: string couleur: color nbPortes: int {nbPortes > 2} kilométrage: int moteur: CARBURANT



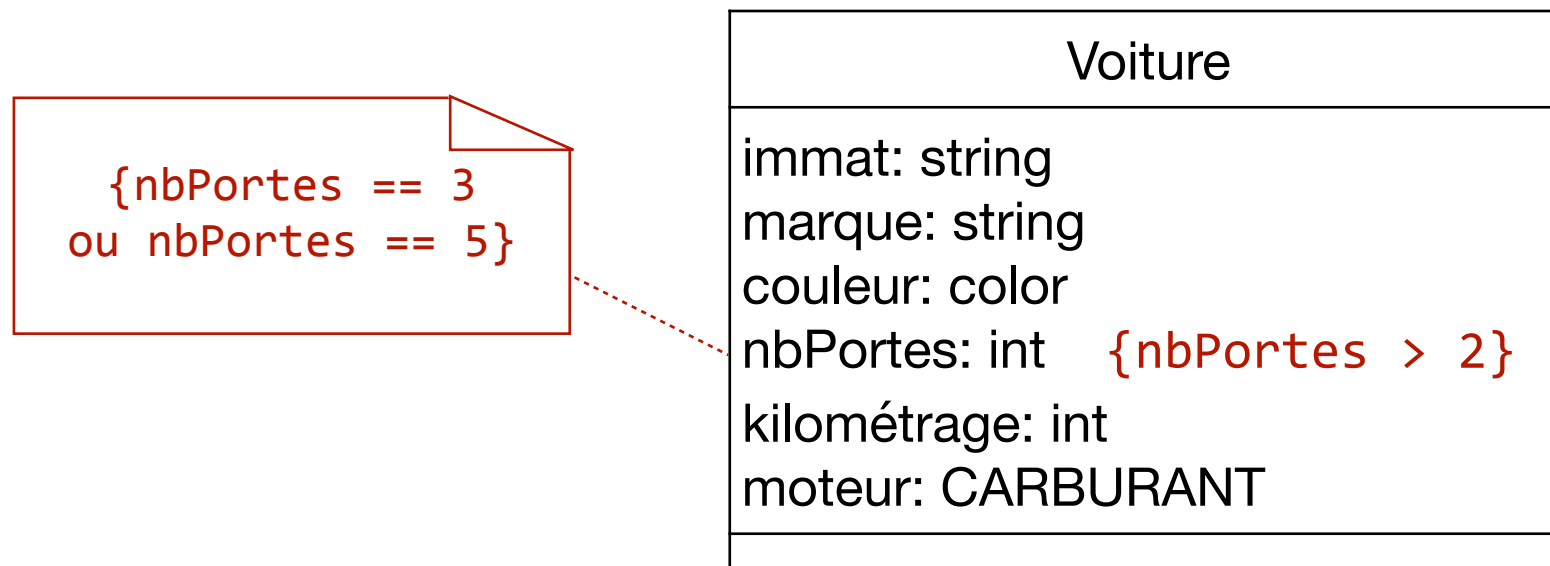
# Contraintes

## Contraintes sur les attributs



# Contraintes

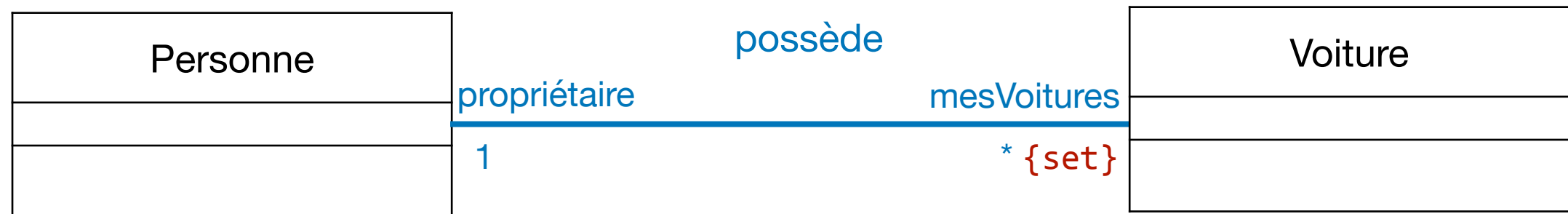
## Contraintes sur les attributs



Les voitures peuvent, en fonction des modèles comporter 3 ou 5 portes

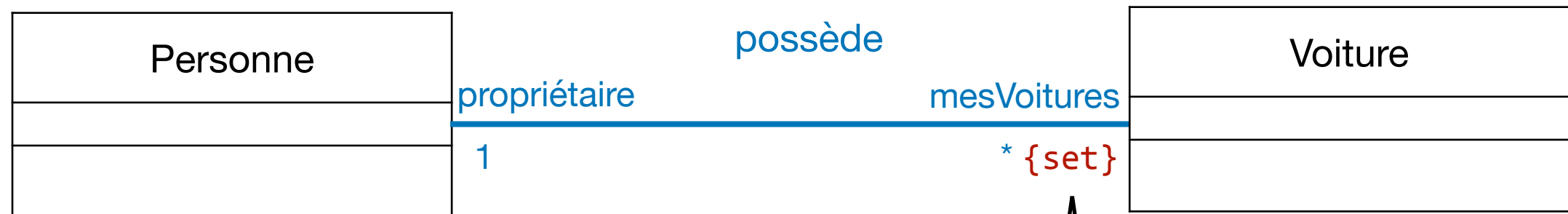
# Contraintes

## Contraintes sur les multiplicités



# Contraintes

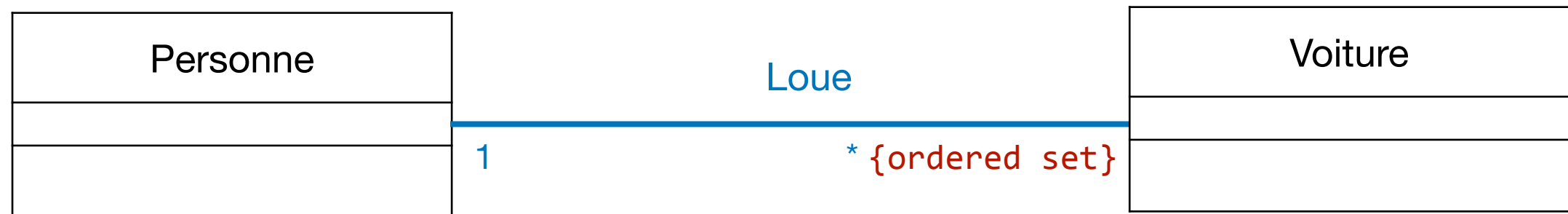
## Contraintes sur les multiplicités



P1 possède V1, V2 et V3  
Ensemble (ordre : non, doublons : non)

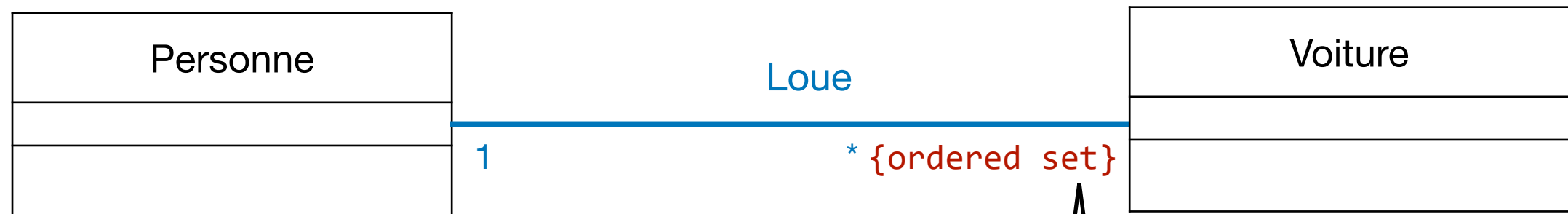
# Contraintes

## Contraintes sur les multiplicités



# Contraintes

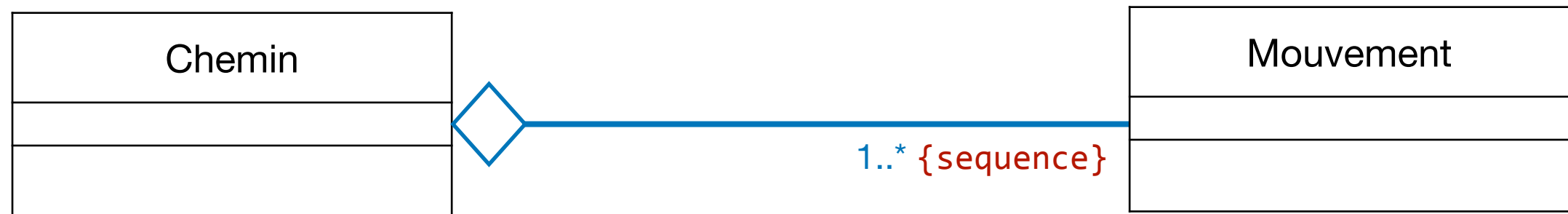
## Contraintes sur les multiplicités



P1 a loué V3 puis V1 puis V2 :  
Ensemble ordonné (ordre : oui, doublons : non)

# Contraintes

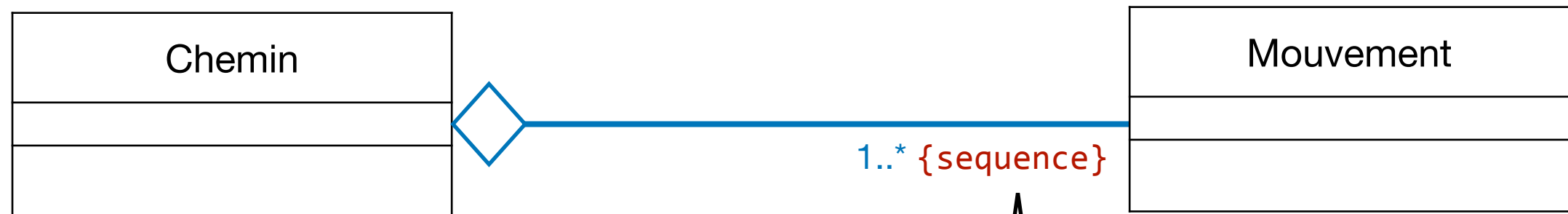
## Contraintes sur les multiplicités



# Contraintes

## Contraintes sur les multiplicités

C



C1 = <G, D, D, G, H, B, B, D, G, V>  
Liste (ordre : oui, doublons : oui)



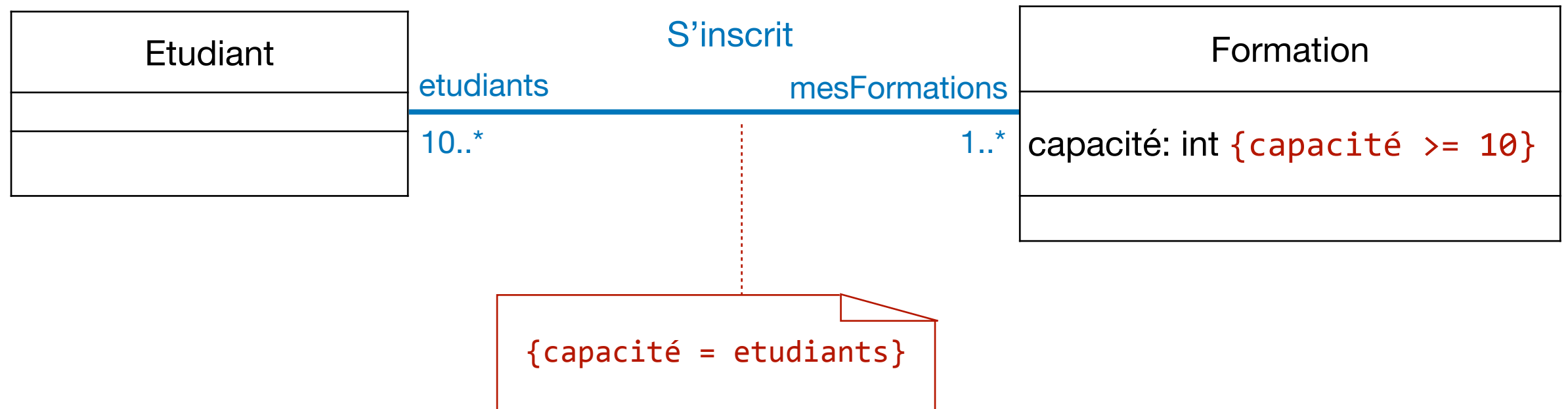
# Contraintes

## Contraintes sur les associations



# Contraintes

## Contraintes sur les associations



# Contraintes

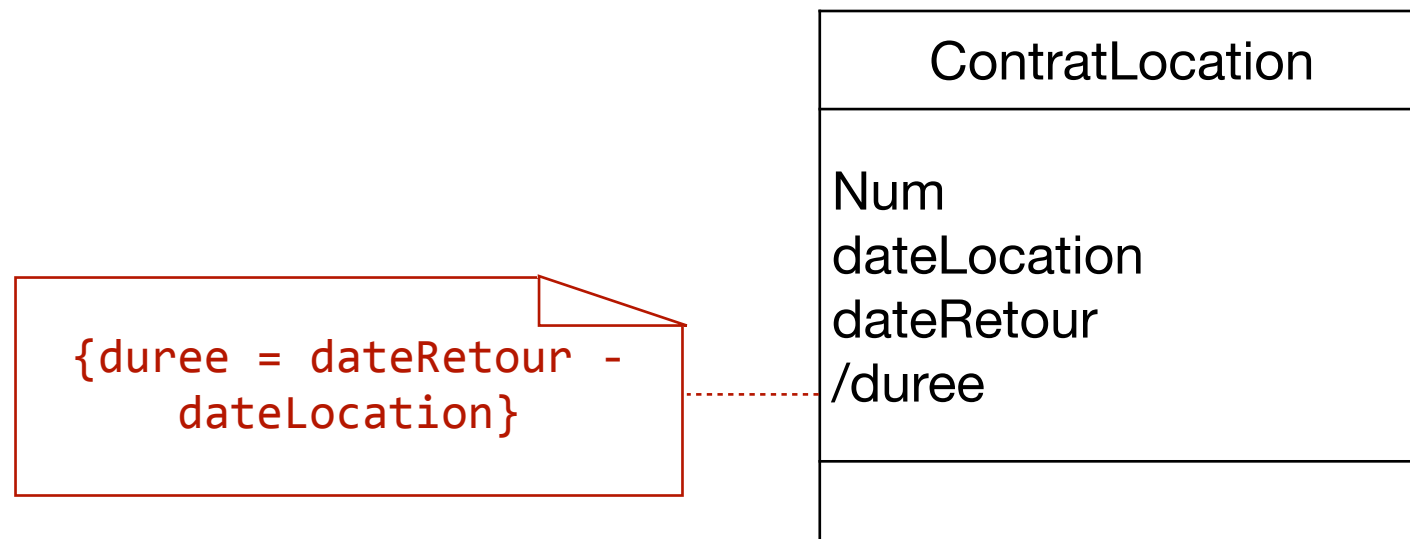
## Contraintes entre attributs



ContratLocation
Num dateLocation dateRetour /duree

# Contraintes

## Contraintes entre attributs



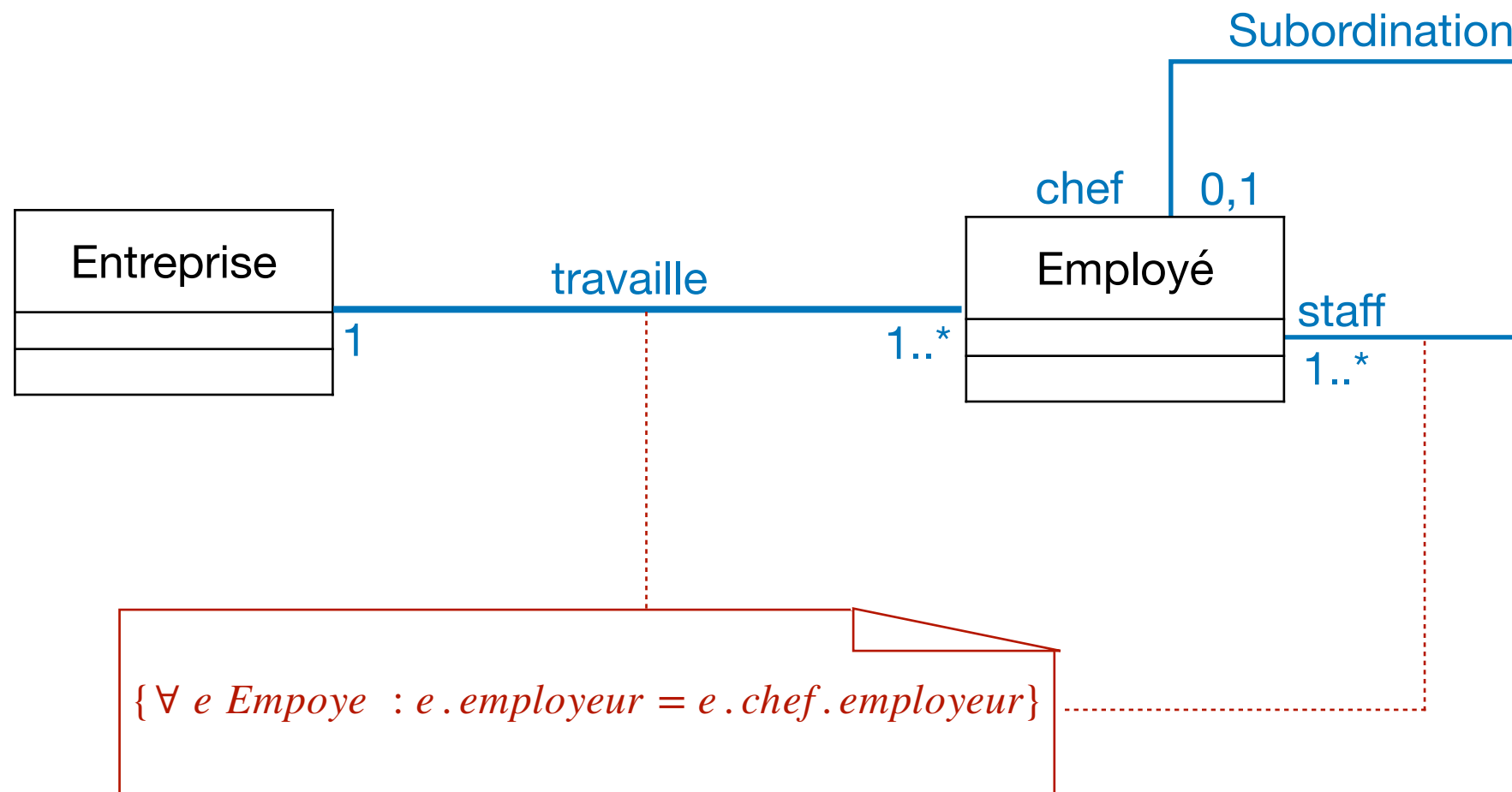
# Contraintes

## Contraintes entre associations



# Contraintes

## Contraintes entre associations



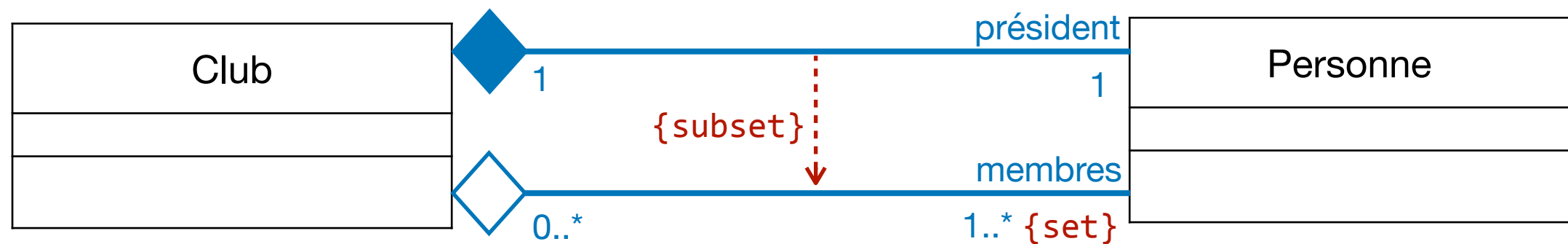
# Contraintes

## Contrainte {subset}



# Contraintes

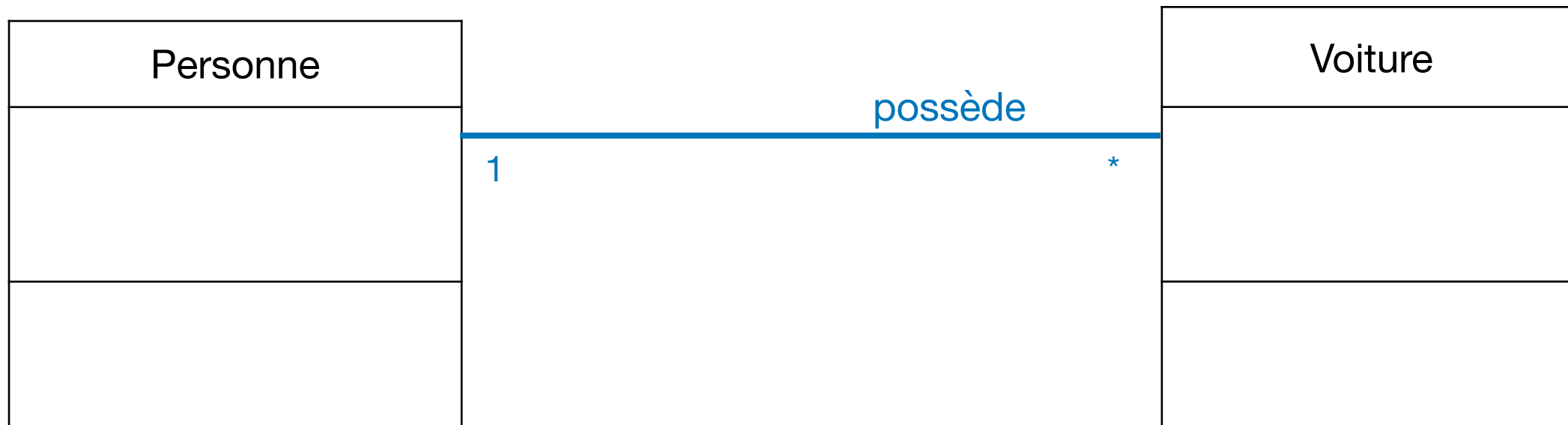
## Contrainte {subset}





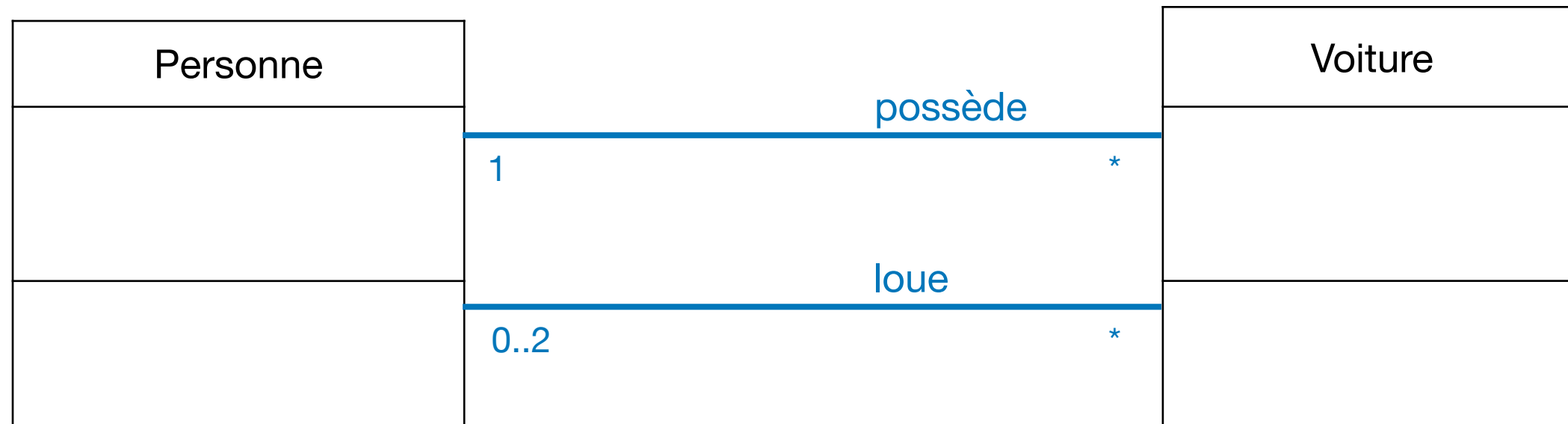
# Contraintes

## Contrainte {XOR}



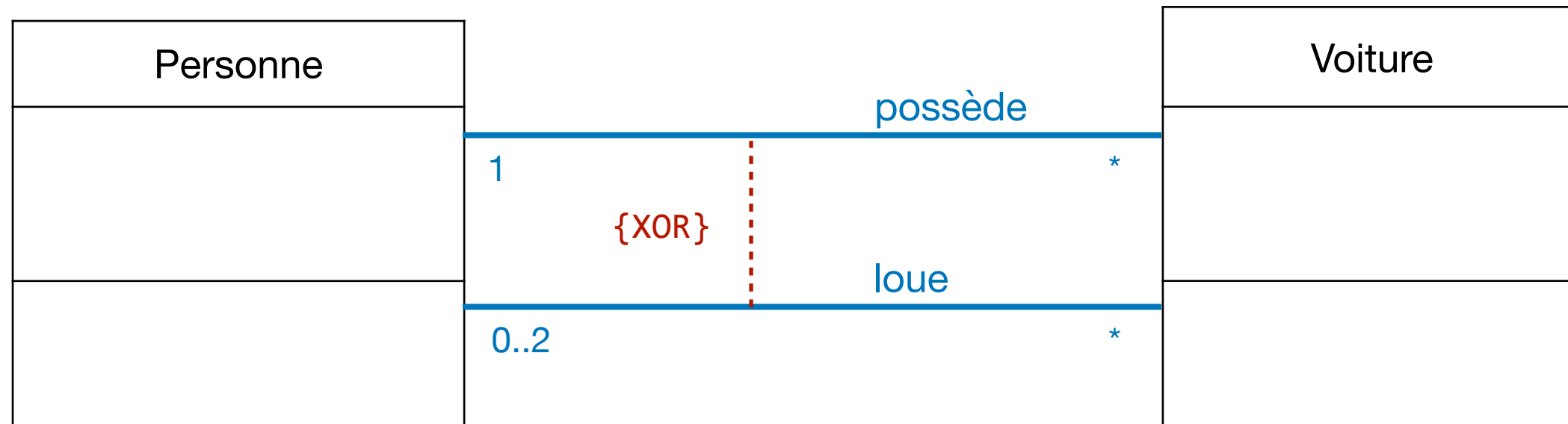
# Contraintes

## Contrainte {XOR}



# Contraintes

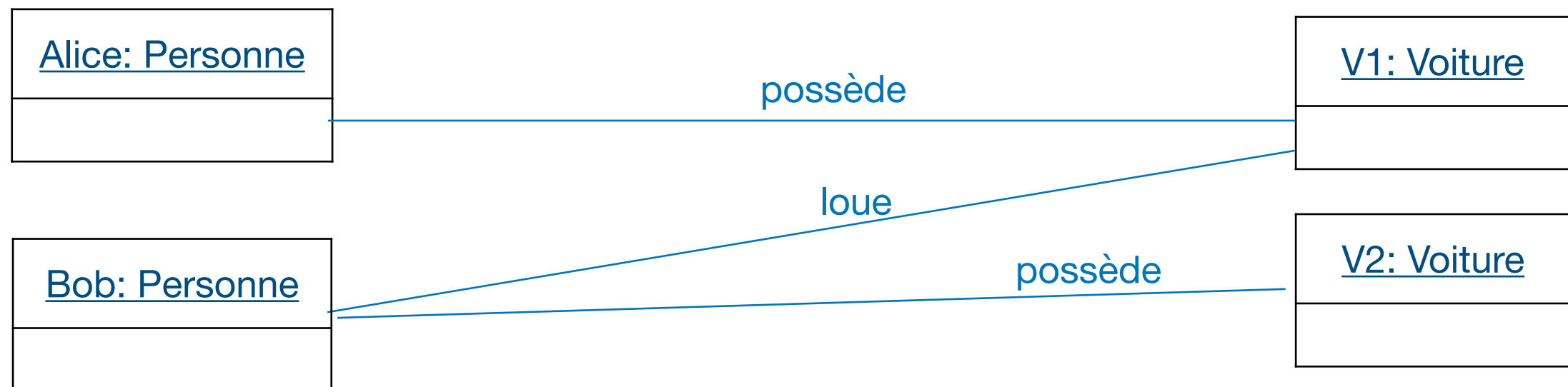
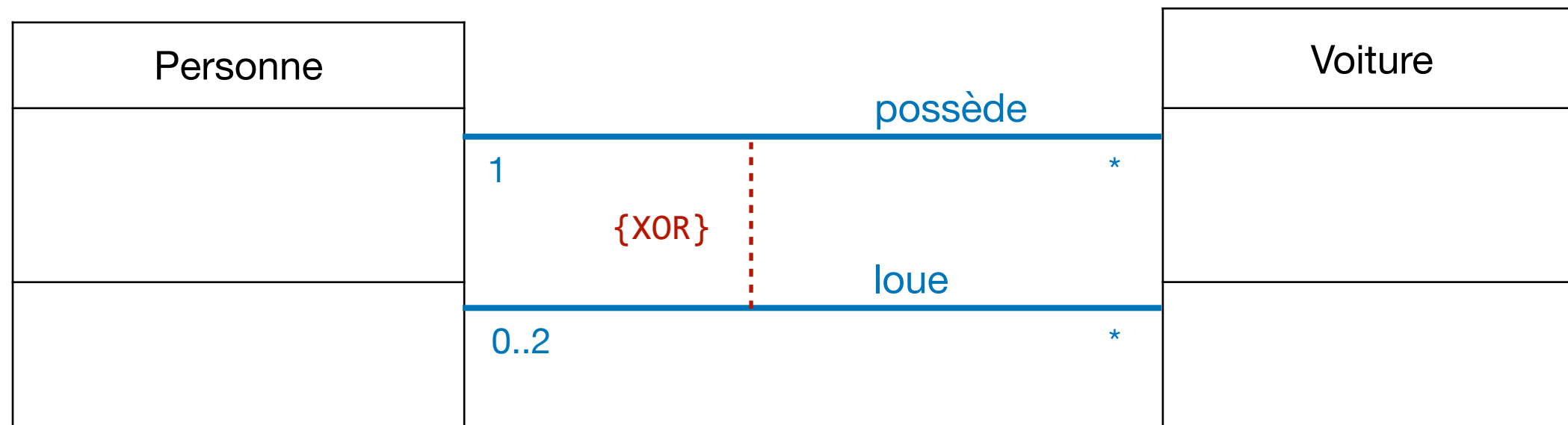
## Contrainte {XOR}



# Contraintes

## Contrainte {XOR}

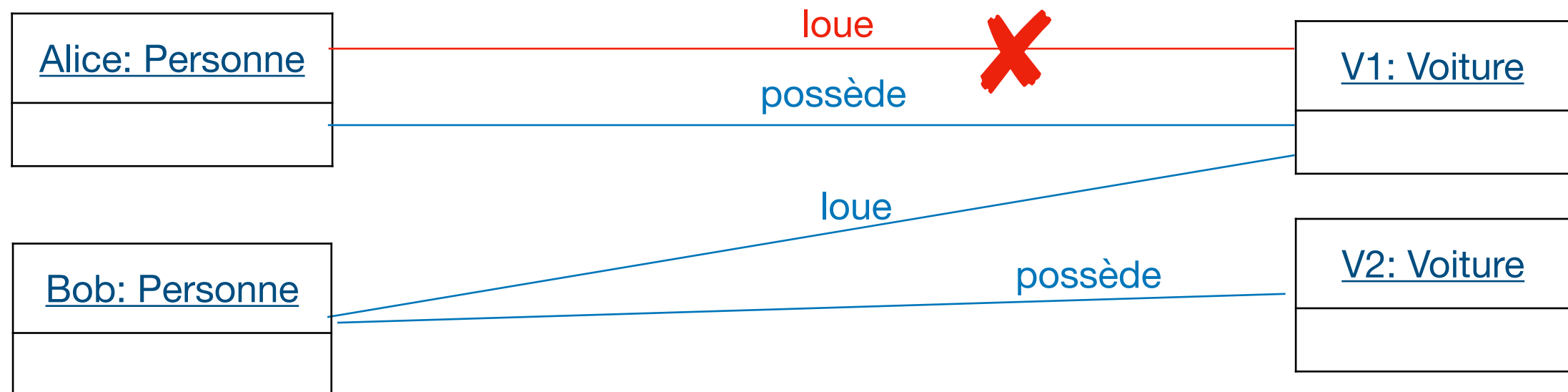
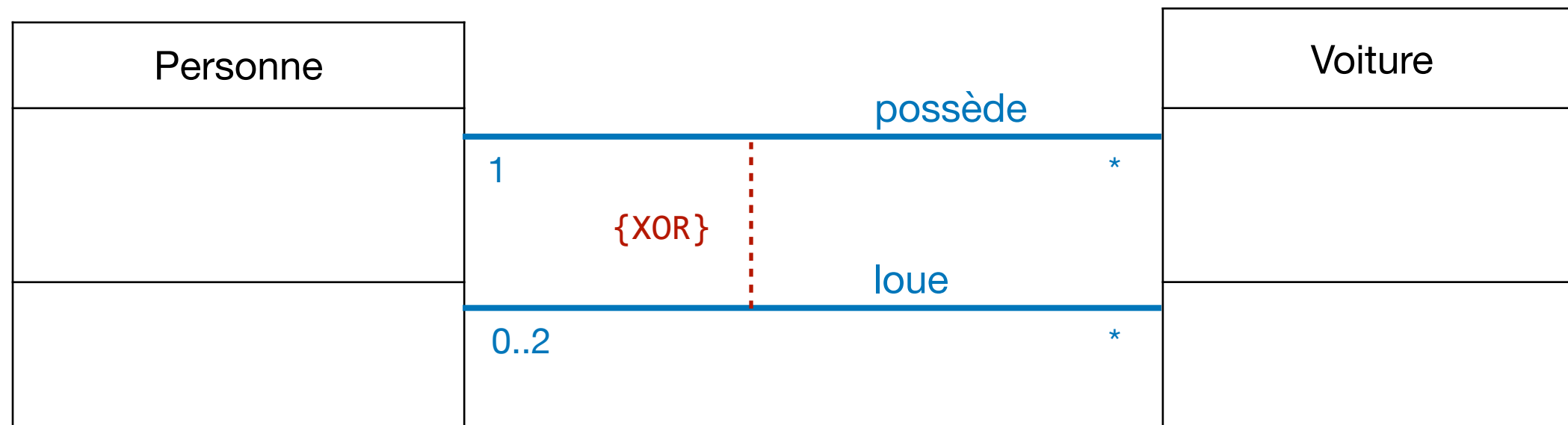
C



# Contraintes

## Contrainte {XOR}

C



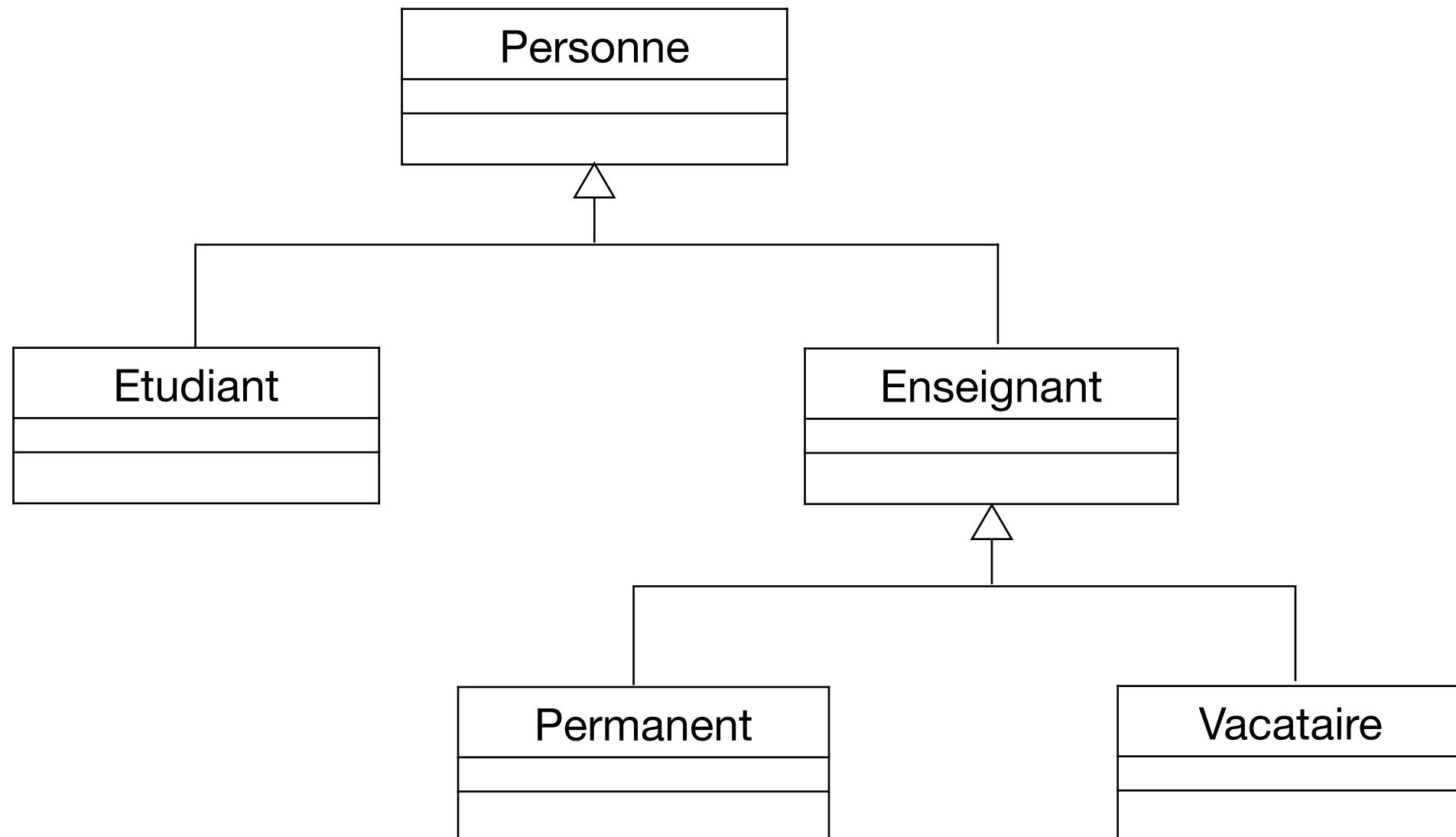
# Contraintes

Contraintes sur la généralisation/spécialisation



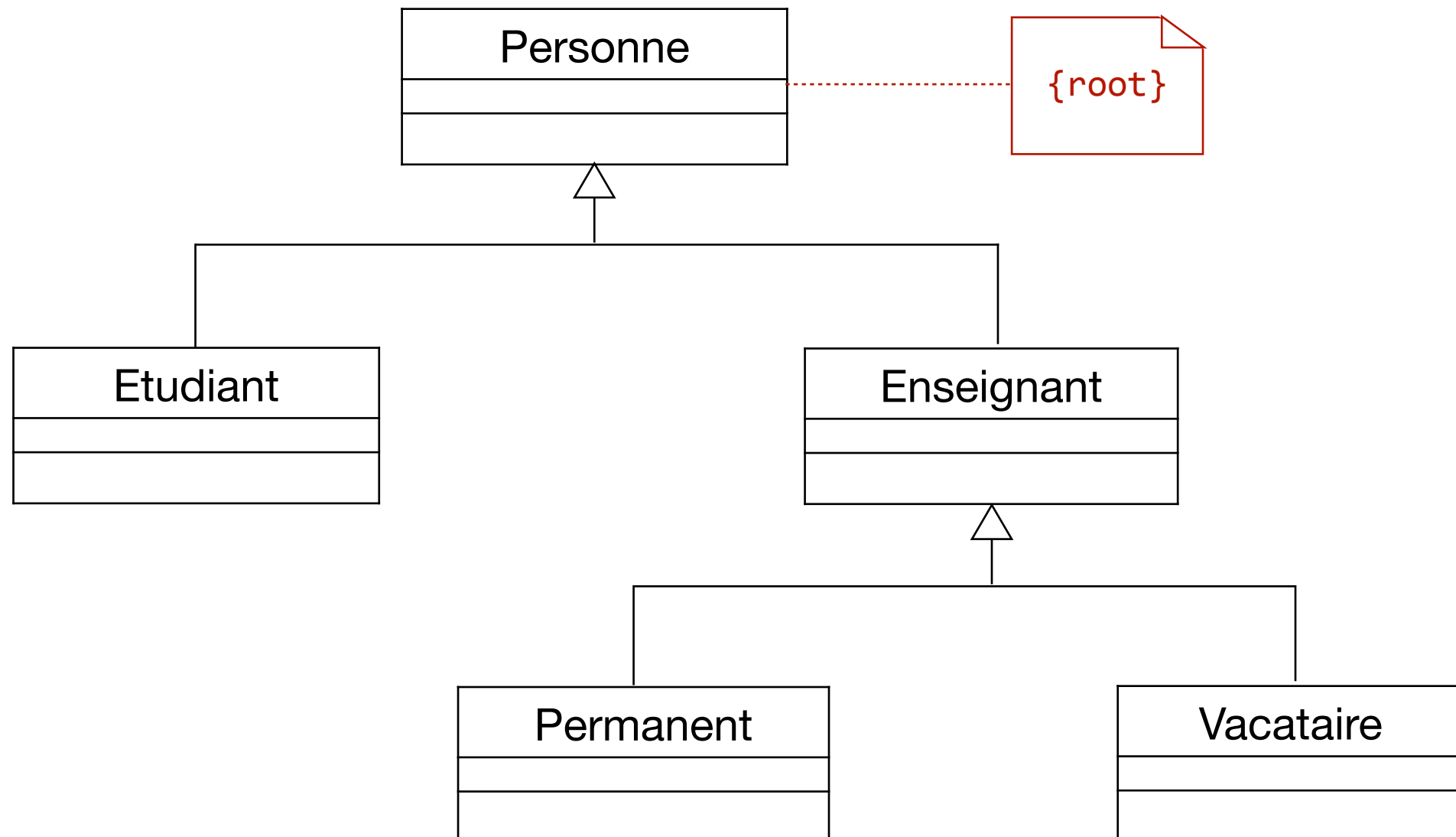
# Contraintes

## Contraintes sur la généralisation/spécialisation



# Contraintes

## Contraintes sur la généralisation/spécialisation

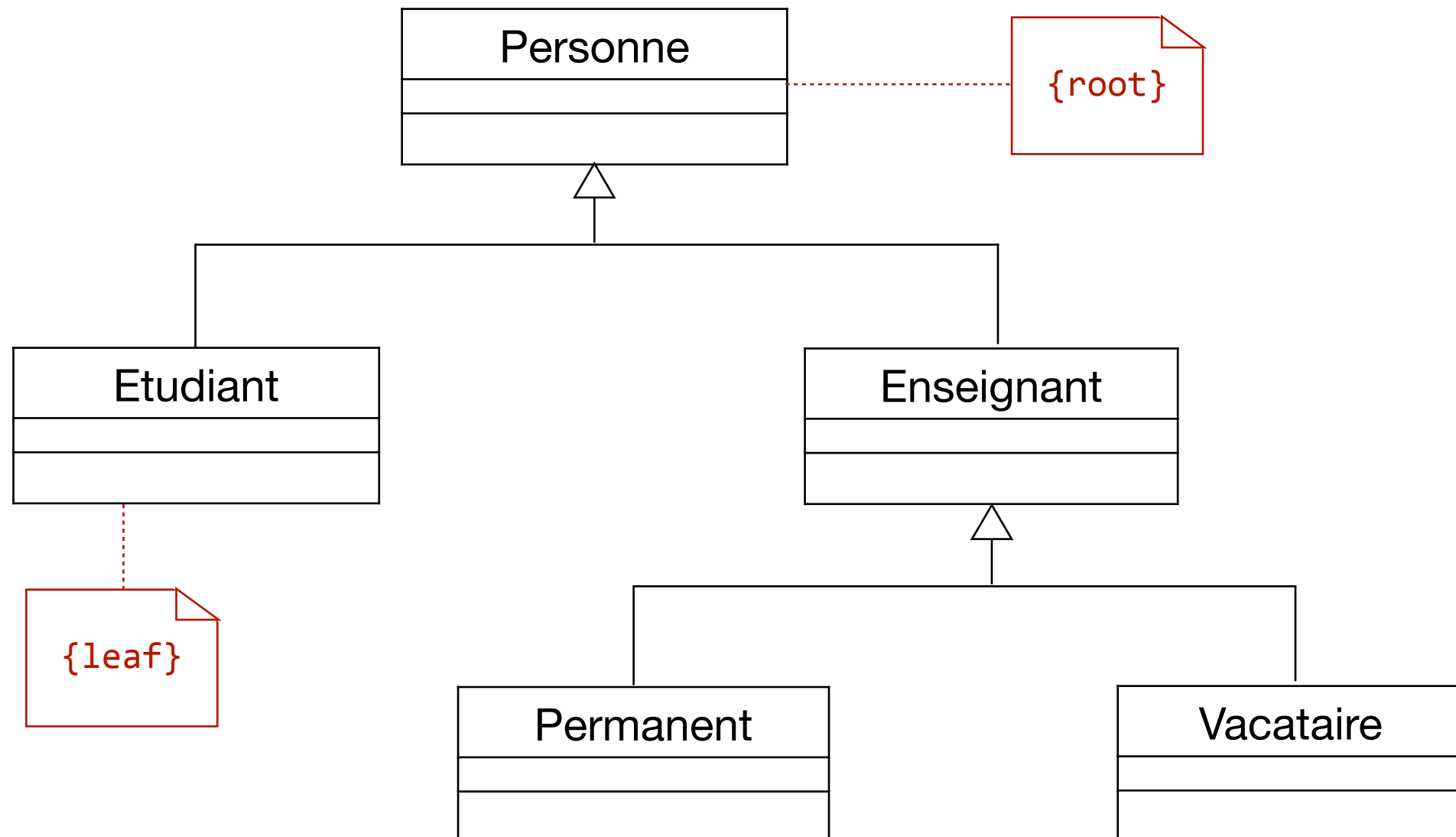




# Contraintes

## Contraintes sur la généralisation/spécialisation

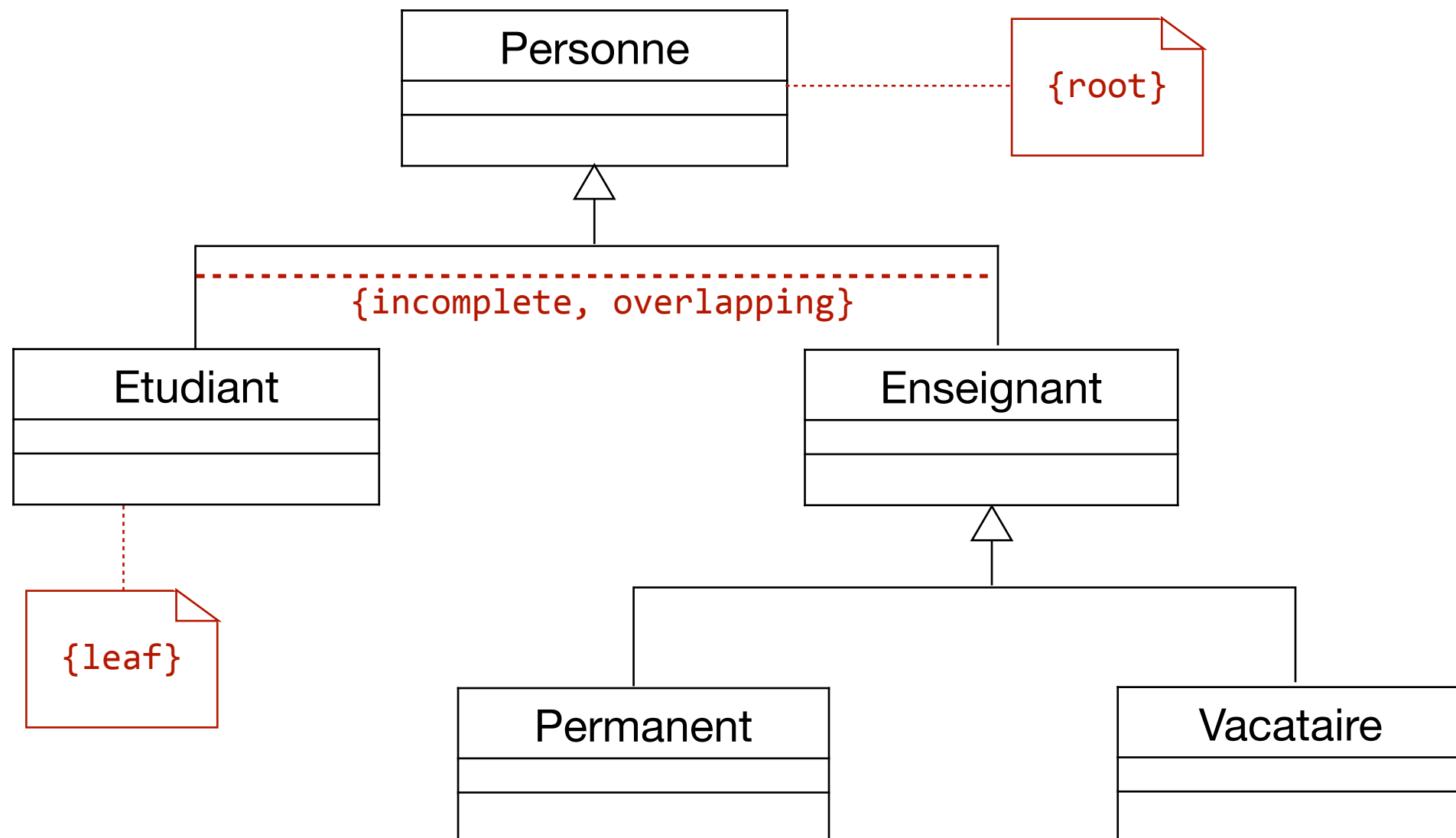
C



# Contraintes

## Contraintes sur la généralisation/spécialisation

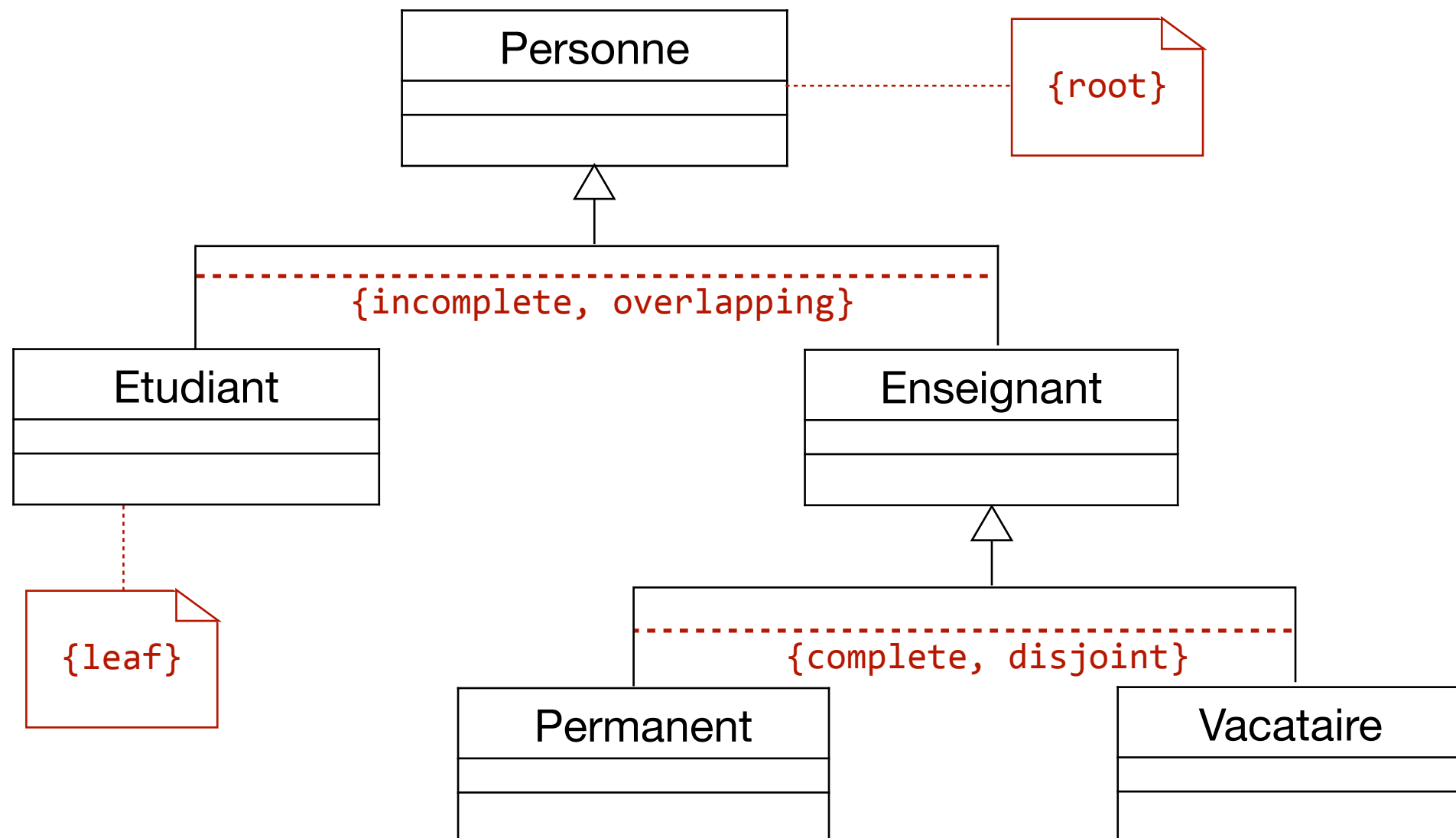
C



# Contraintes

## Contraintes sur la généralisation/spécialisation

C



# Références

## Books

- **UML Distilled (Third Edition): A Brief Guide to the Standard Object Modeling Language.** M Fowler 2004.
- **Object-Oriented Software Engineering (Second Edition): Practical Software Development Using UML and Java.** T. Lethbridge and R. Laganière 2005.
- **UML in Practice: The Art of Modeling Software Systems Demonstrated through Worked P.** Rogues 2004.
- **Requirements Engineering: From System Goals to UML Models to Software Specifications.** A. Lamsweerde 2009.
- **Software Engineering with UML.** B. Unhelkar 2018.

# Many

## Thanks to

Noureddine Aribi II Sébastien Bardin II Nassim Belmecheri II  
Kevin Chapuis II Pierre Gérard II Arnaud Gotlieb II Marianne Huchard II  
Laurent Jean-Pierre II Alain Joubert II Yahia Lebbah II  
Delphine Longuet II Samir Loudni II Sophie Nabitz II  
Clémentine Nebut II Matthieu Rosenfeld II Alain Sabatier II  
Helmut Simonis II Djamel Seriali II Christine Solnon II  
Julie Vachon II Petru Valicov II Keunhyuk Yeom II Raphael Yende