Modélisation des Objets

Bases de la Conception Orientée Objet - AS

Nadjib Lazaar (nadjib.lazaar@umontpellier.fr)

UML

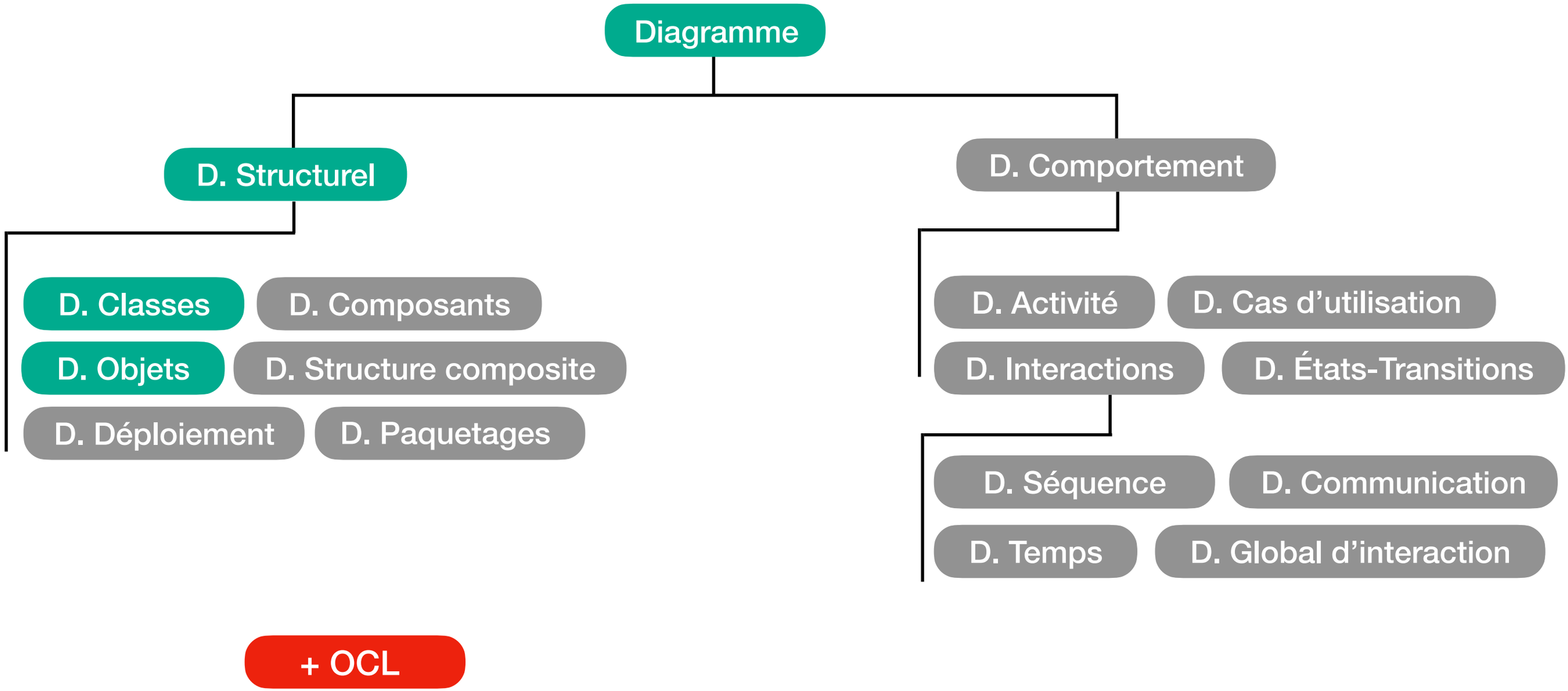
Dans ce module

- **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

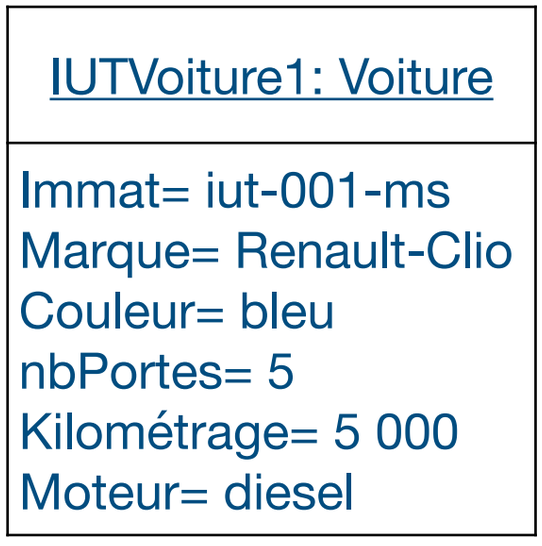
Définition

- 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

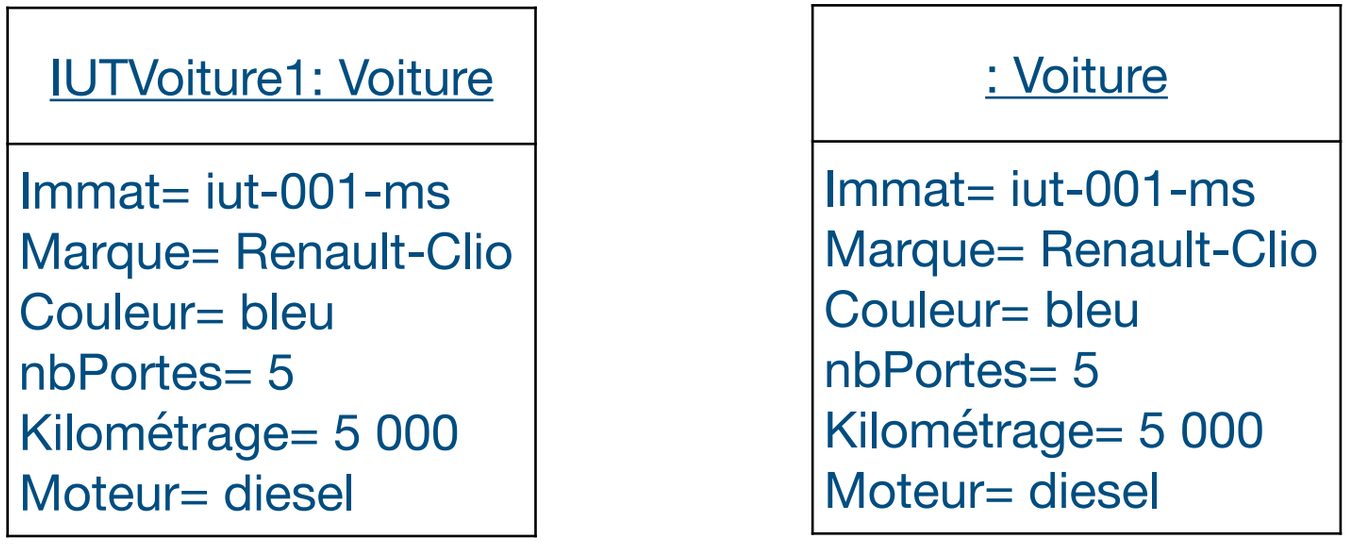
- Représentation des objets et leurs relations à un instant donné
- Graphe (noeuds = objets, arrêtes = relations)
- Permet de décrire des cas de test, illustrer et analyser l’état du système, expliquer les cas particuliers, …



Diagrammes d'objets

Définition

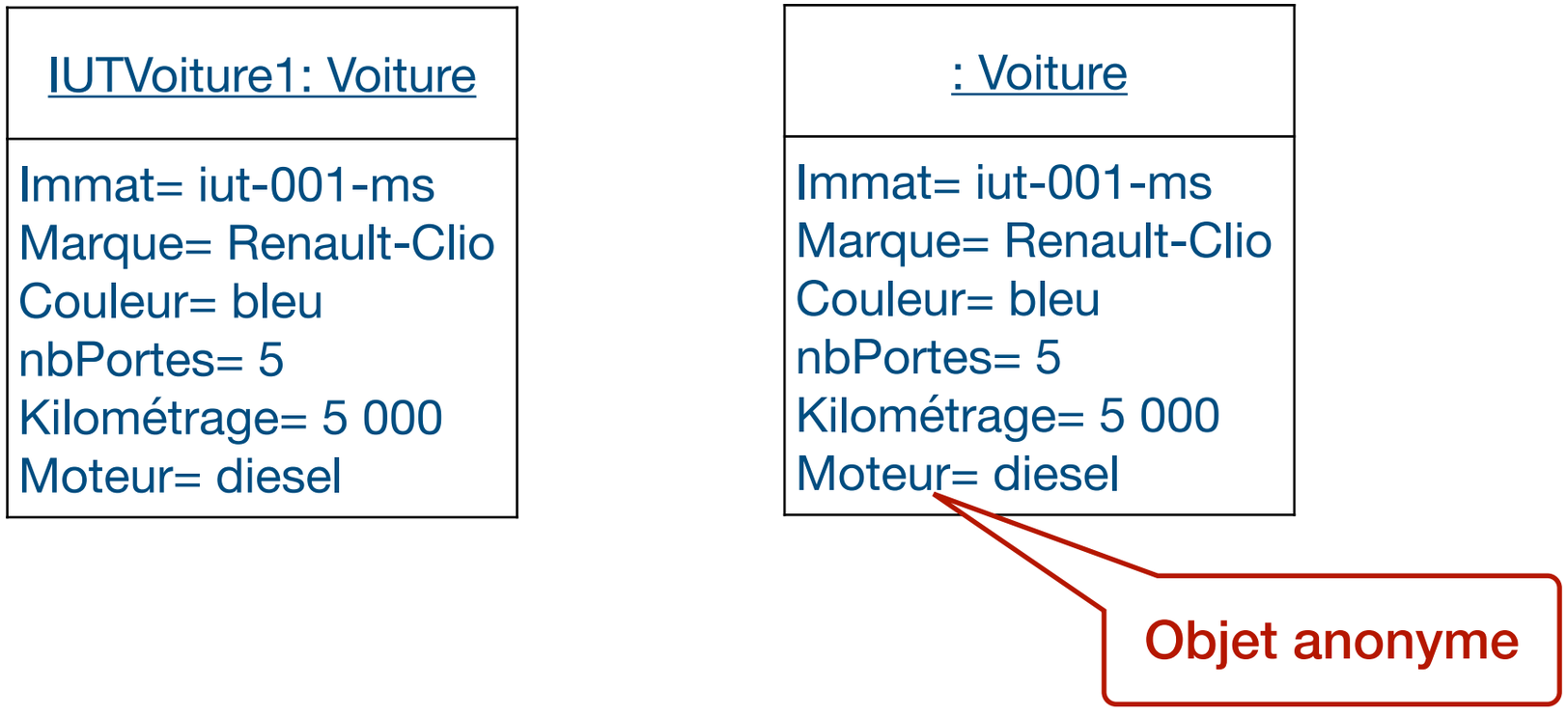
- Représentation des objets et leurs relations à un instant donné
- Graphe (noeuds = objets, arrêtes = relations)
- Permet de décrire des cas de test, illustrer et analyser l'état du système, expliquer les cas particuliers, …



Diagrammes d'objets

Définition

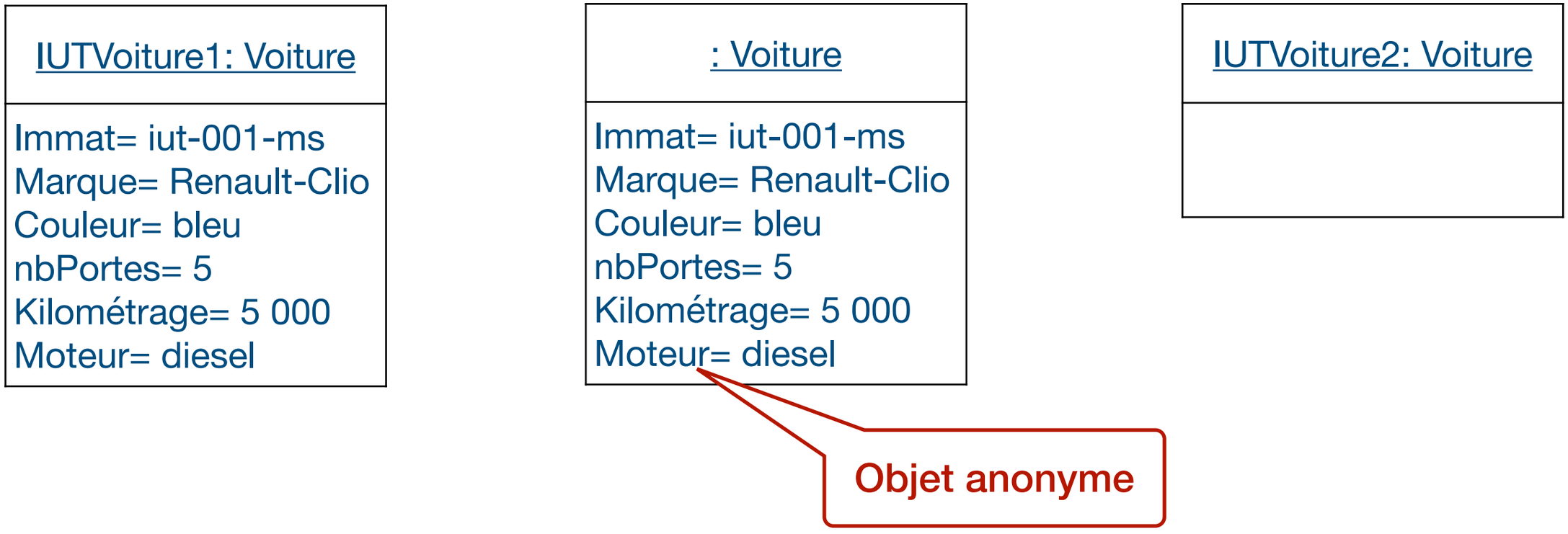
- Représentation des objets et leurs relations à un instant donné
- Graphe (noeuds = objets, arrêtes = relations)
- Permet de décrire des cas de test, illustrer et analyser l'état du système, expliquer les cas particuliers, …



Diagrammes d'objets

Définition

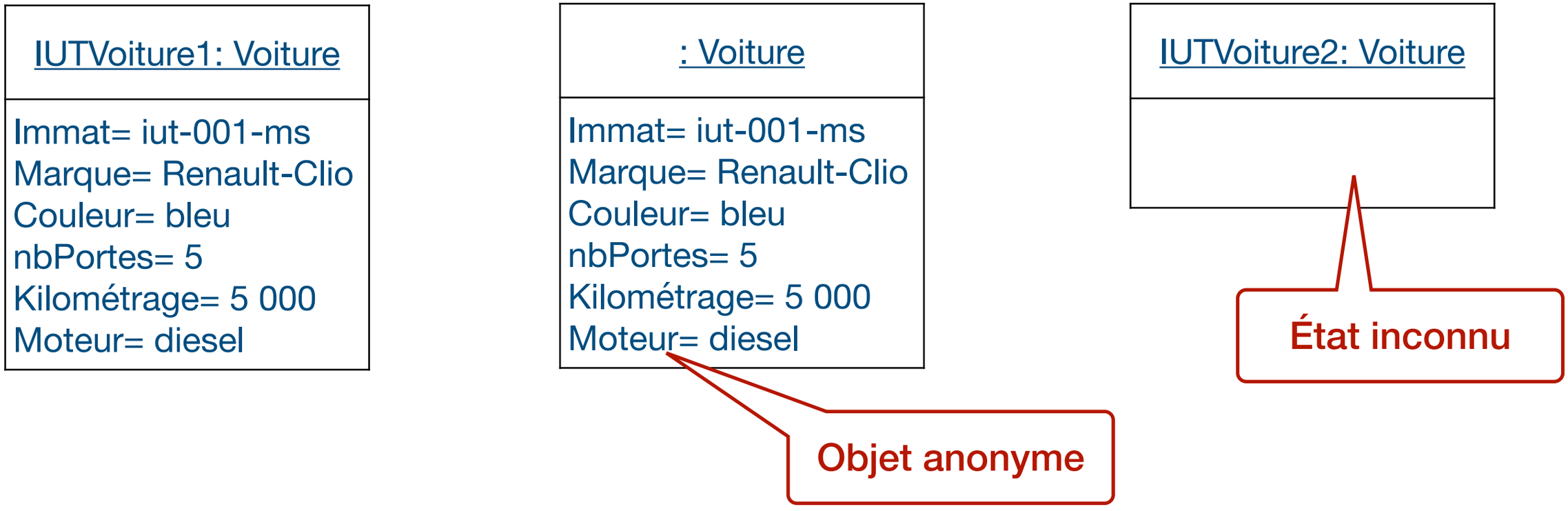
- Représentation des objets et leurs relations à un instant donné
- Graphe (noeuds = objets, arrêtes = relations)
- Permet de décrire des cas de test, illustrer et analyser l'état du système, expliquer les cas particuliers, …



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

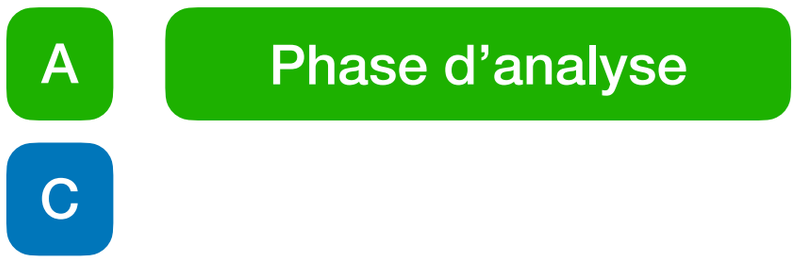
- Abstraction de l’ensemble des diagrammes d’objets possibles
- Graphe (noeuds = classes, arrêtes = relations)
- Permet la définition des objets possibles + structure du système

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

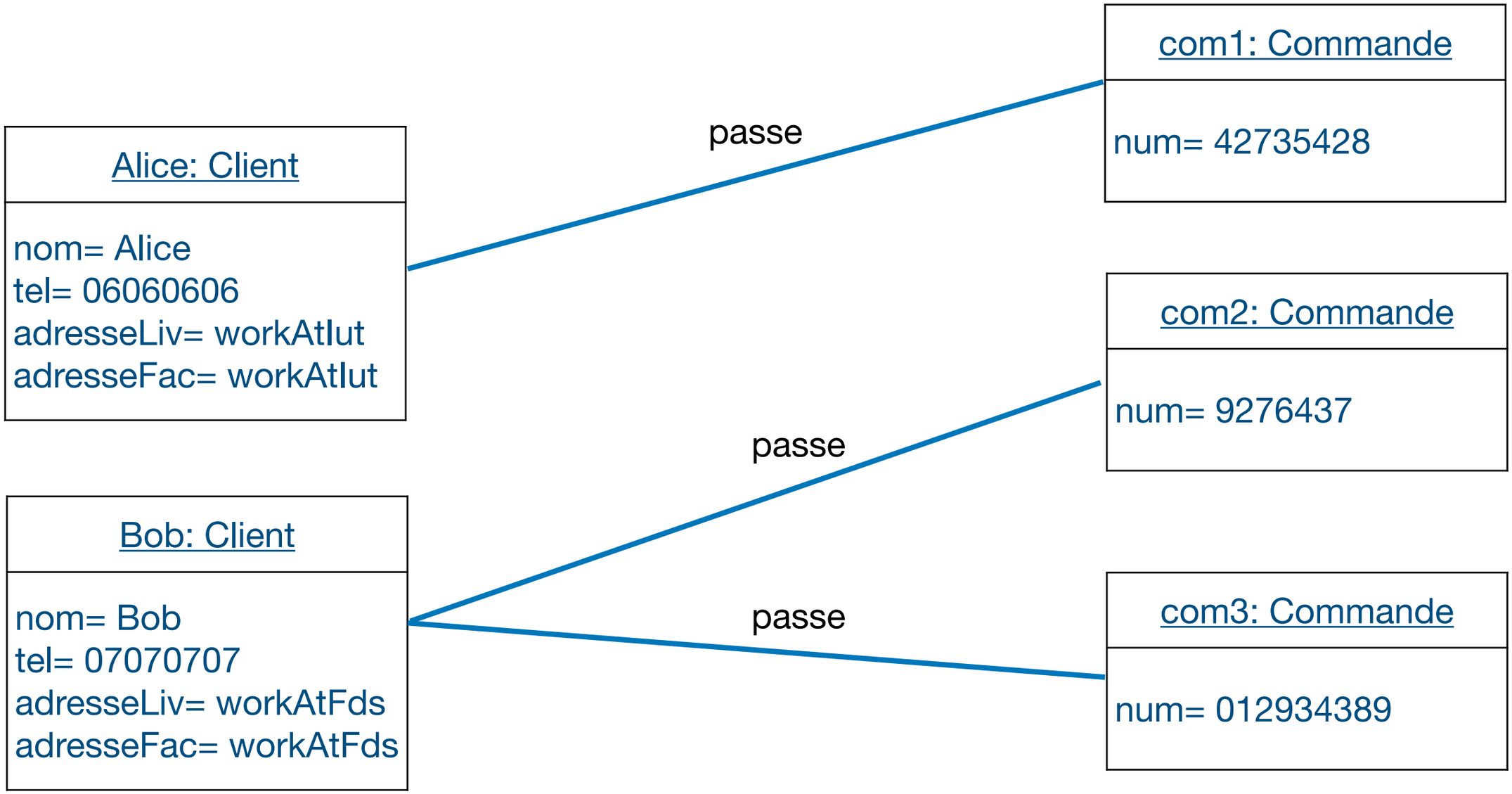
- Abstraction de l’ensemble des diagrammes d’objets possibles
- Graphe (noeuds = classes, arrêtes = relations)
- Permet la définition des objets possibles + structure du système

A Phase d’analyse

C Phase de conception

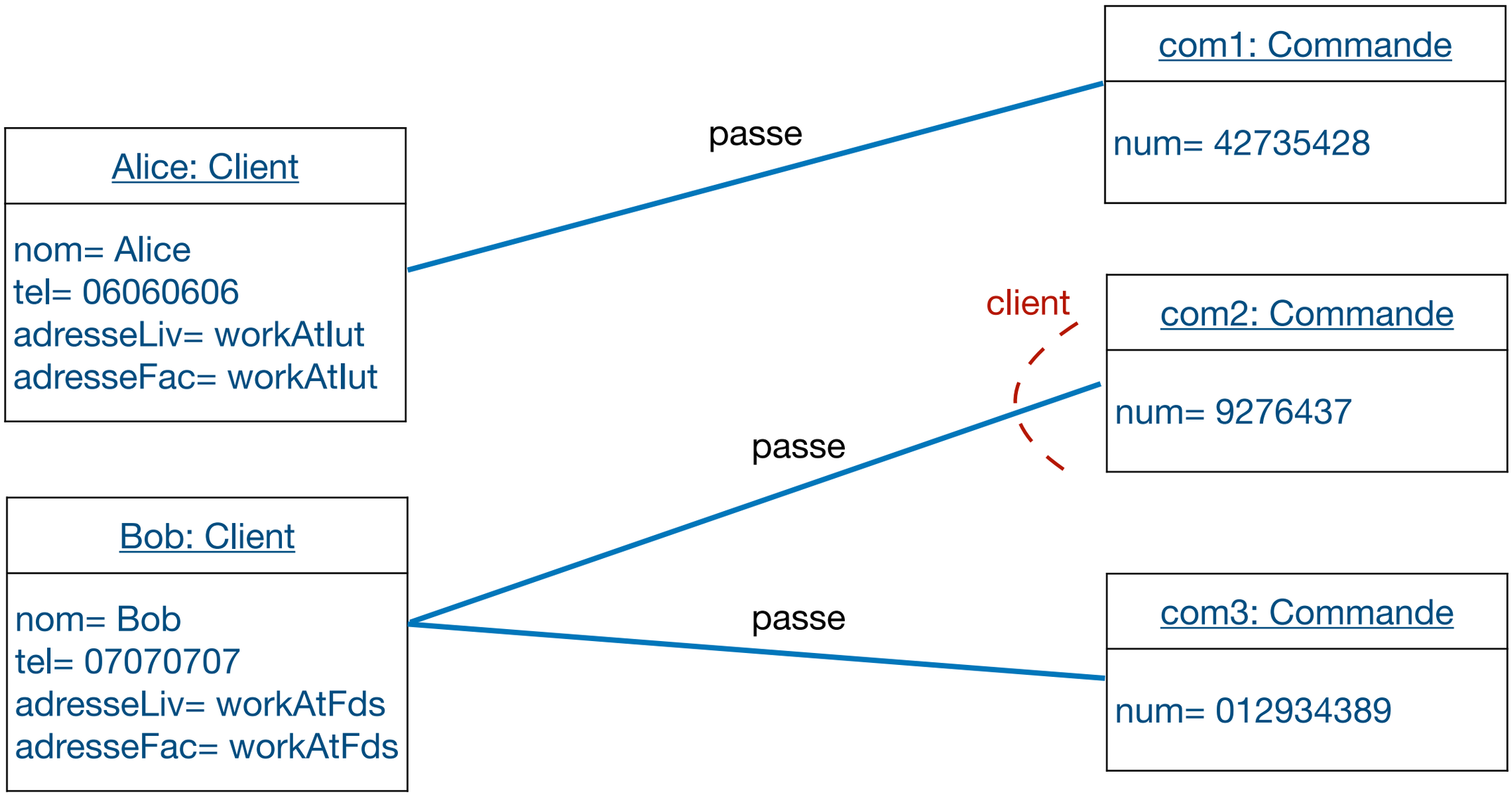
Relation entre objets

Association



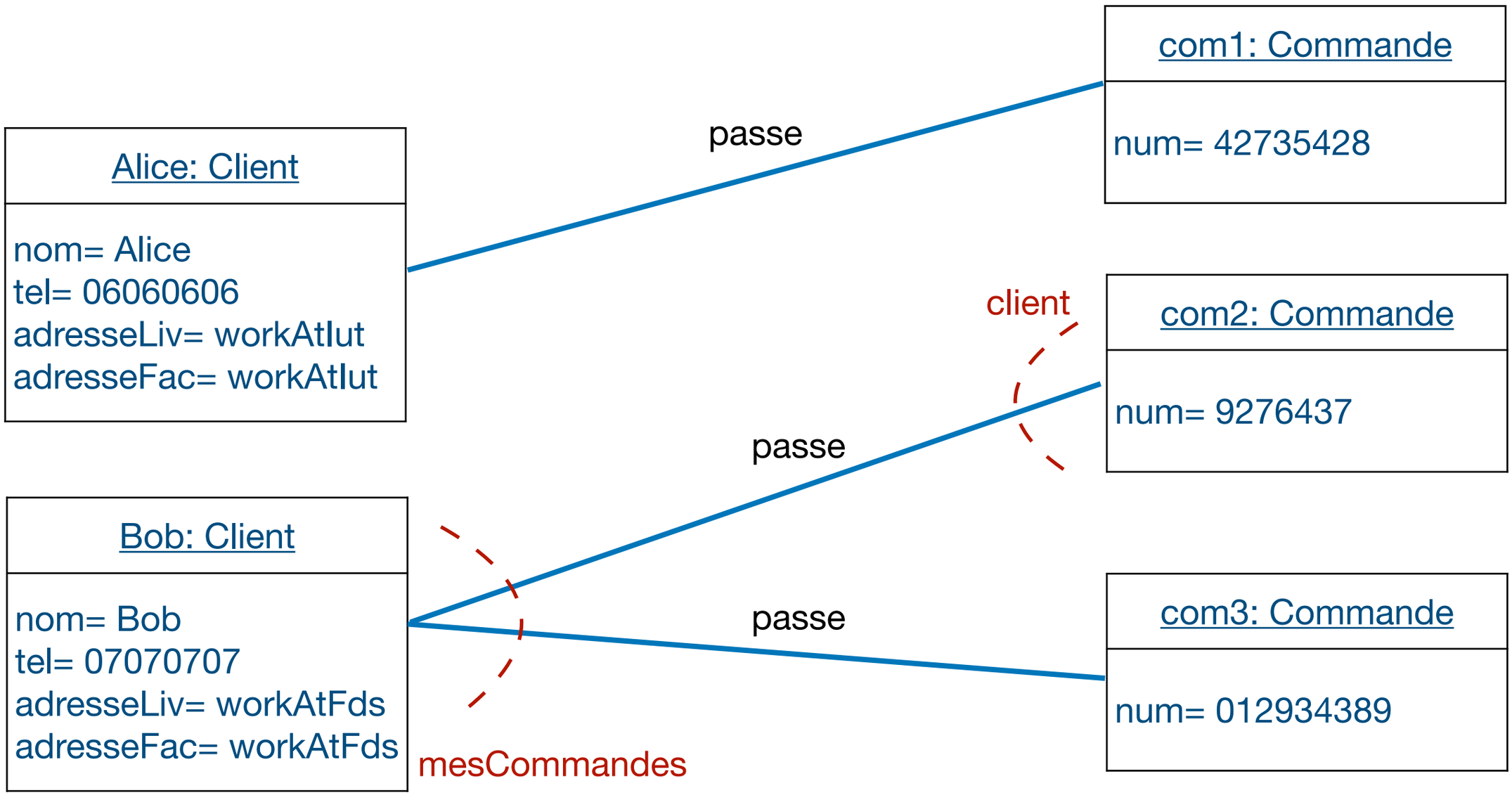
Relation entre objets

Association



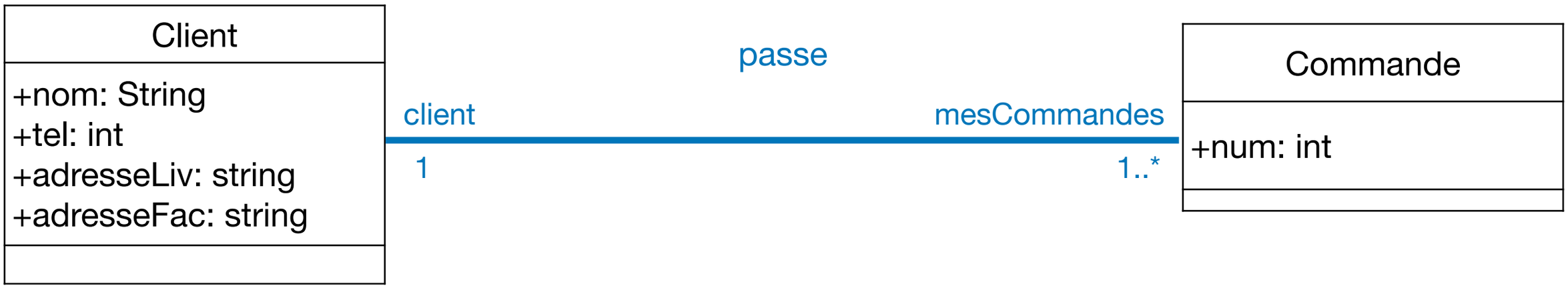
Relation entre objets

Association



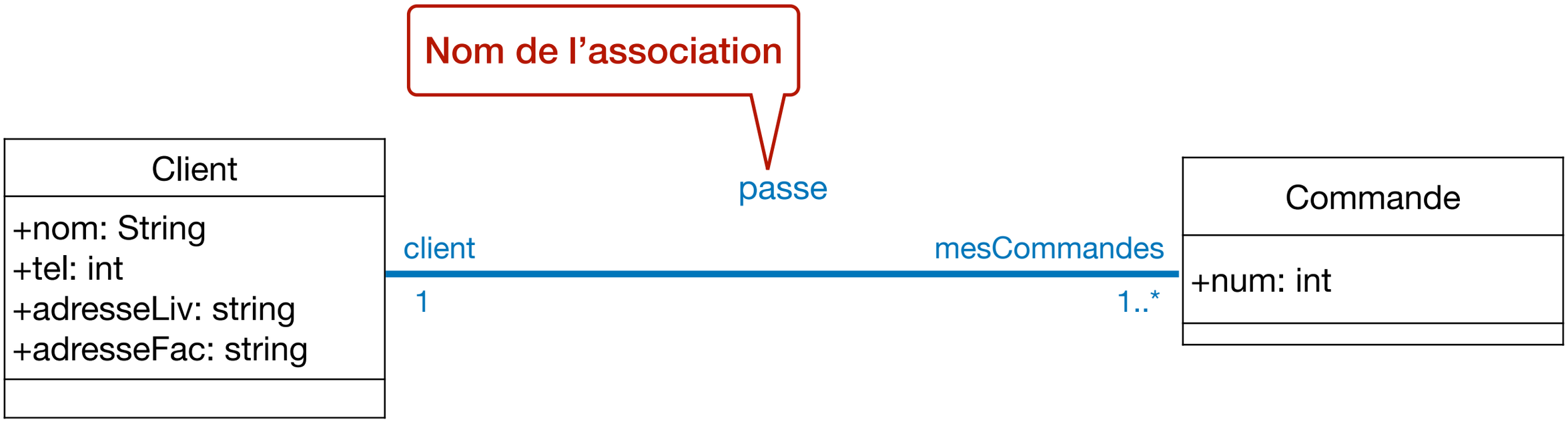
Relation entre classes

Association



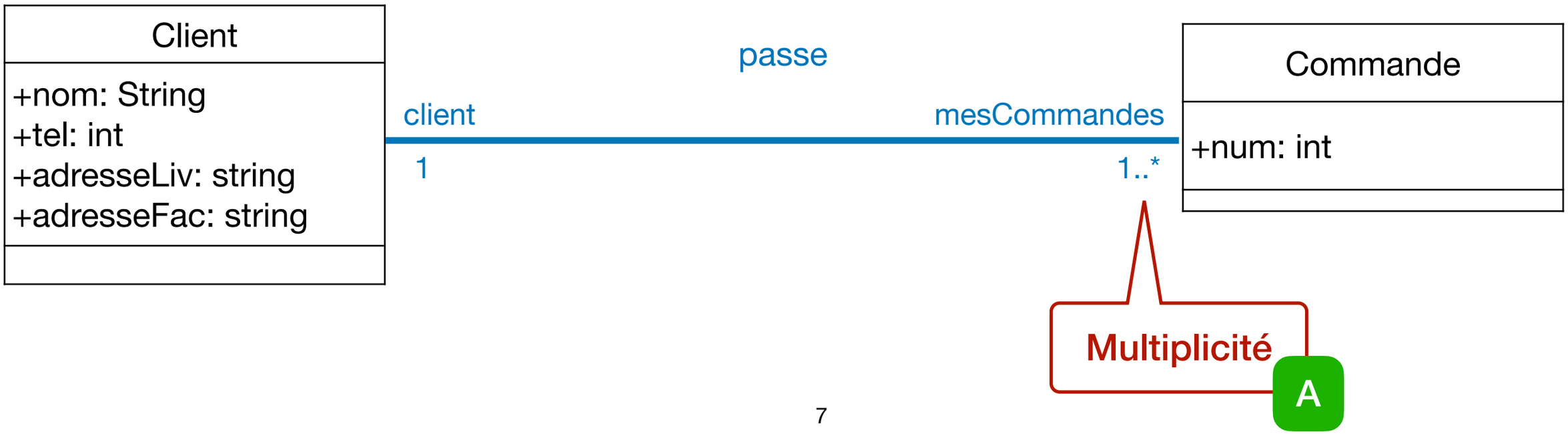
Relation entre classes

Association



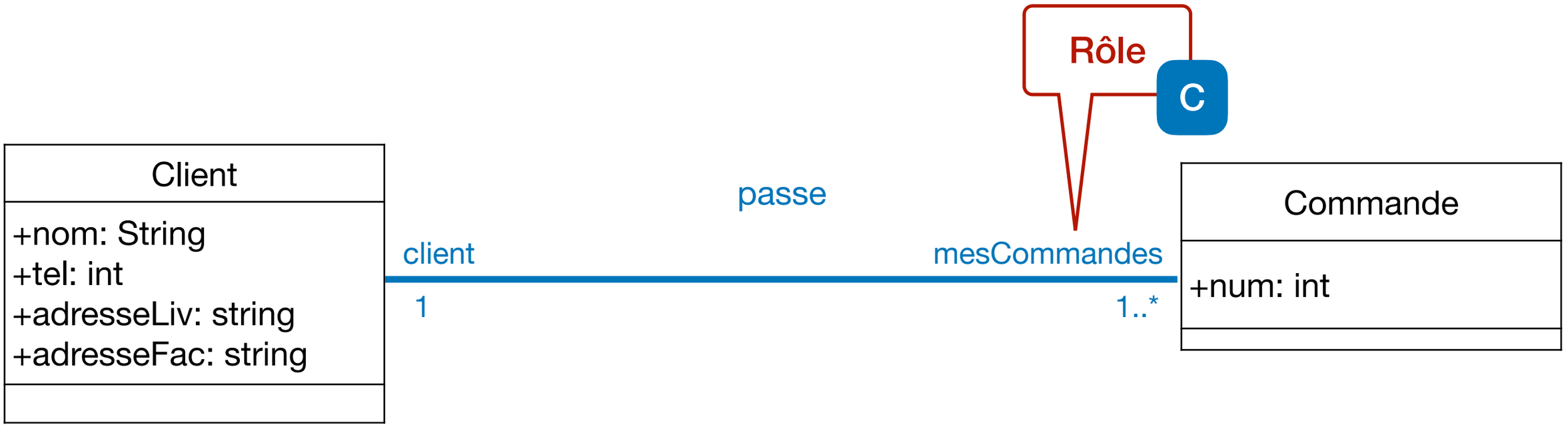
Relation entre classes

Association



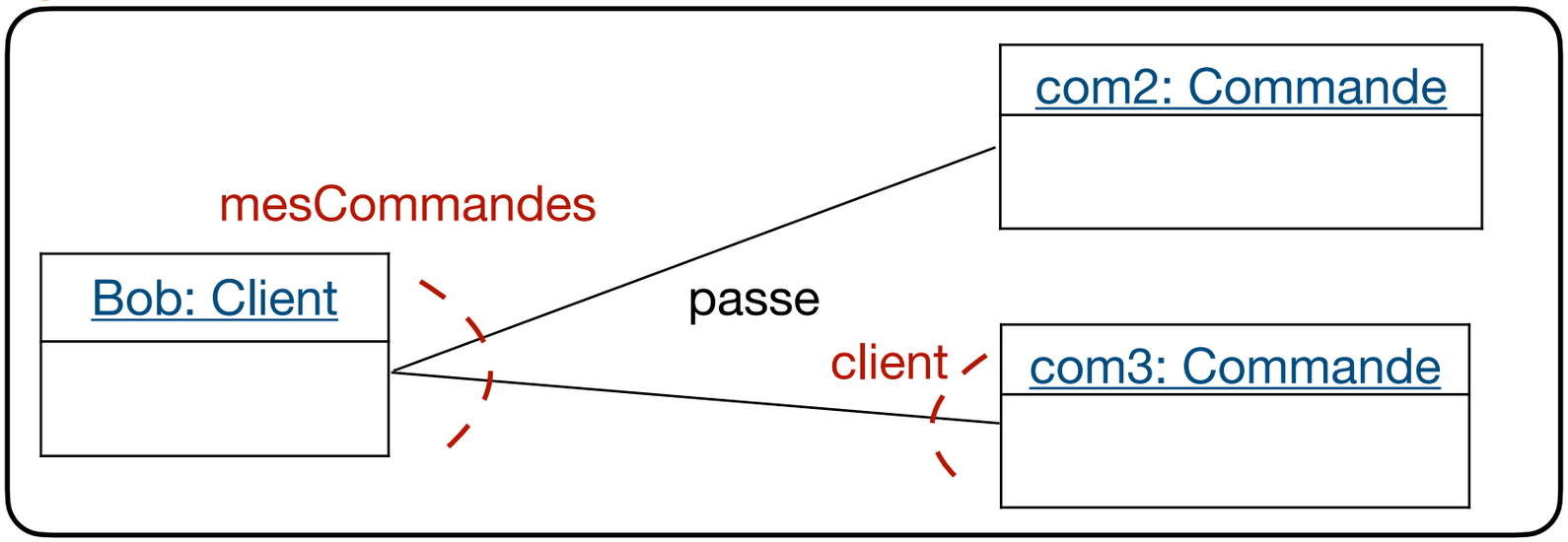
Relation entre classes

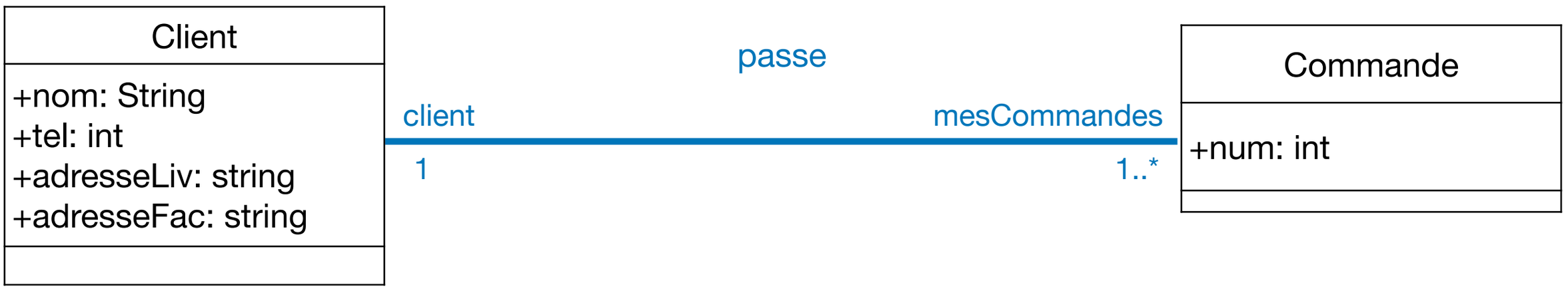
Association



Relation entre classes

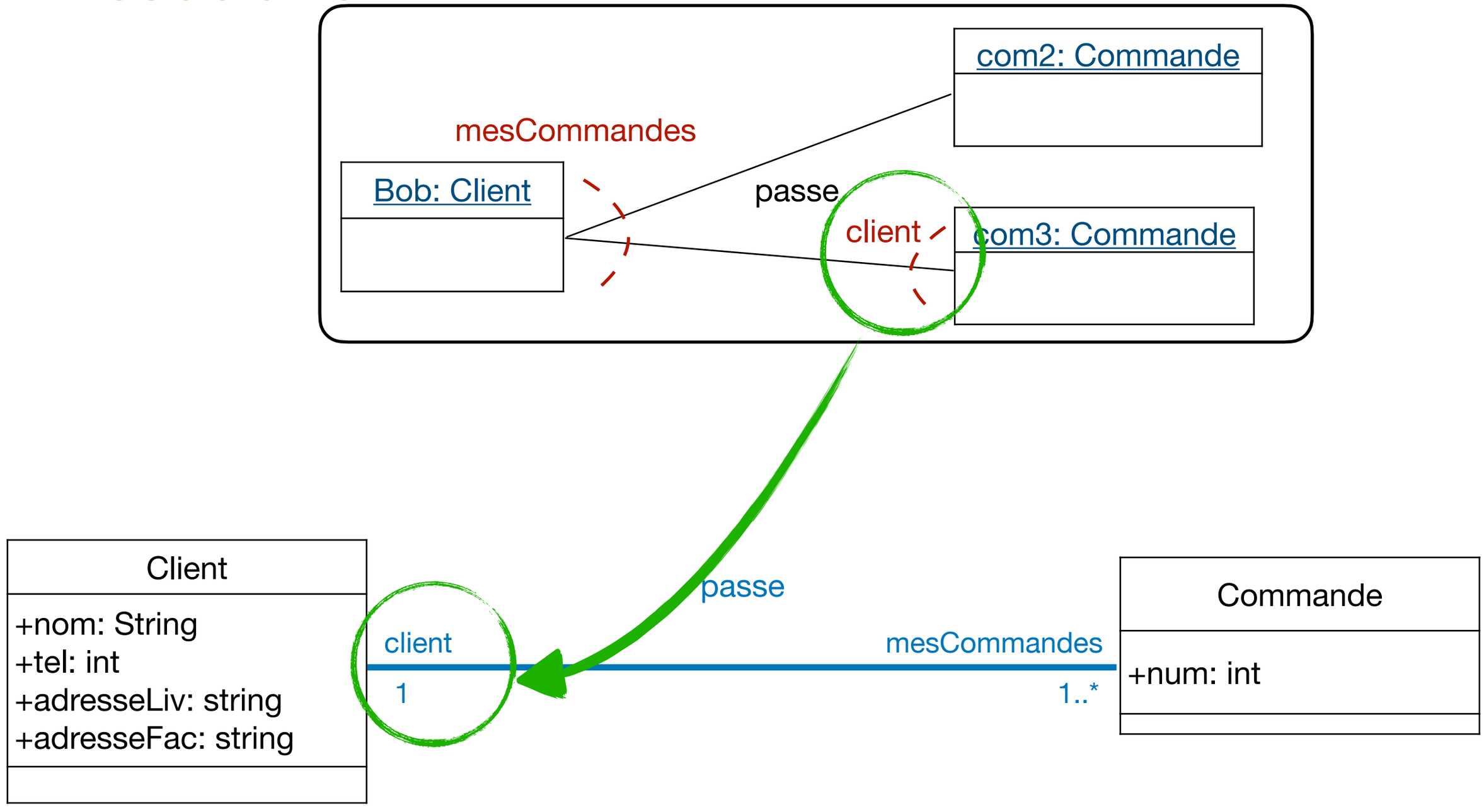
Association





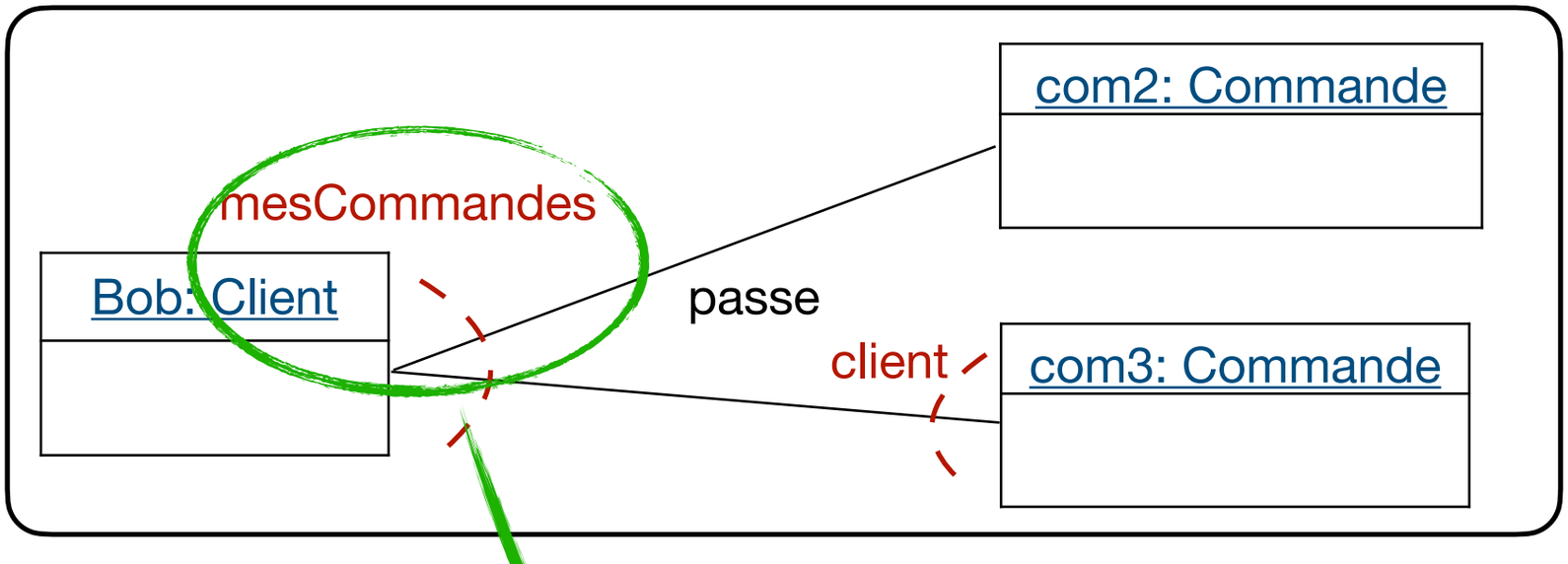
Relation entre classes

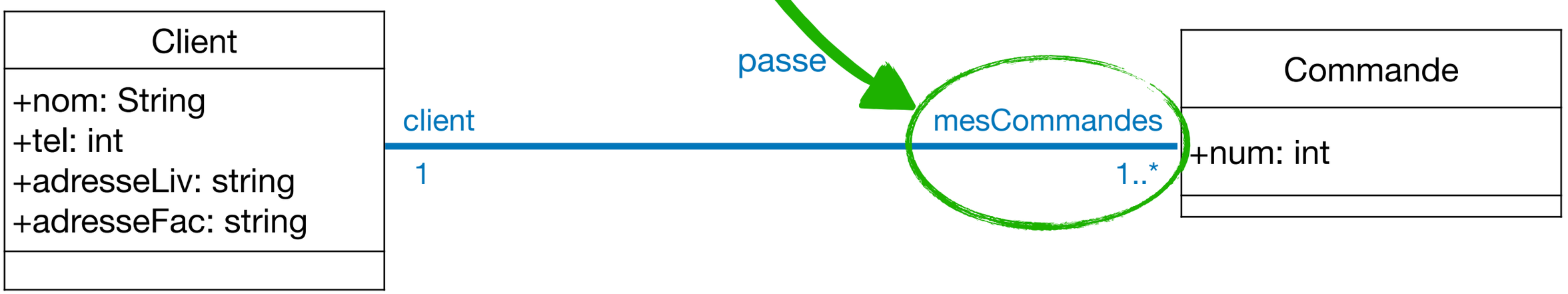
Association



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

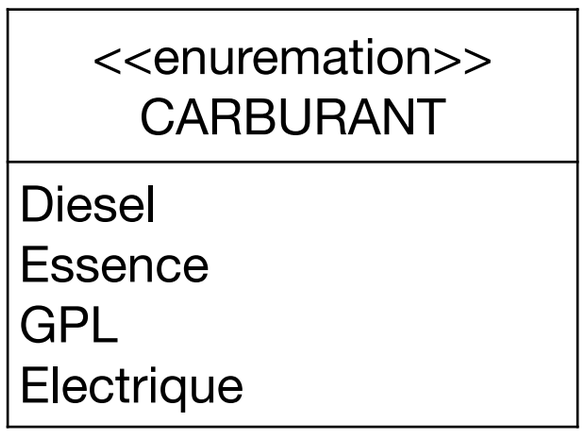
Voiture
immat: String marque: String couleur: String nbPortes: int kilométrage: int moteur: CARBURANT propriétaire: Personne

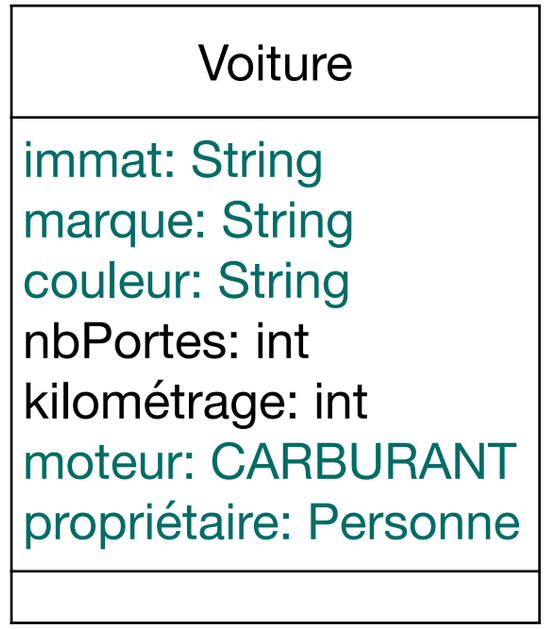
C

Relation entre classes

Types des attributs

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



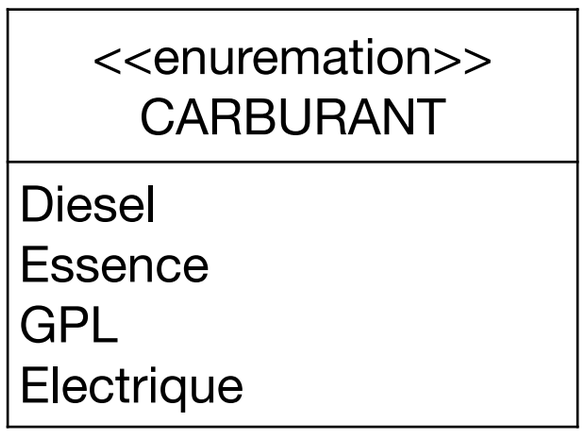


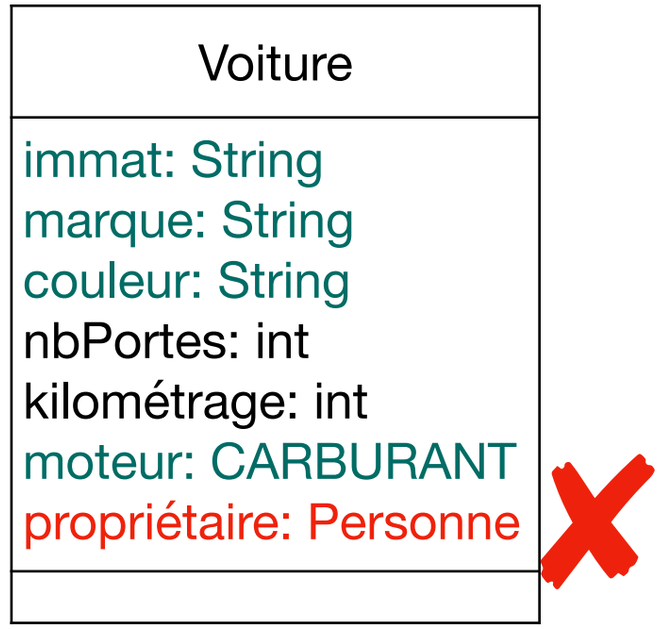
C

Relation entre classes

Types des attributs

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



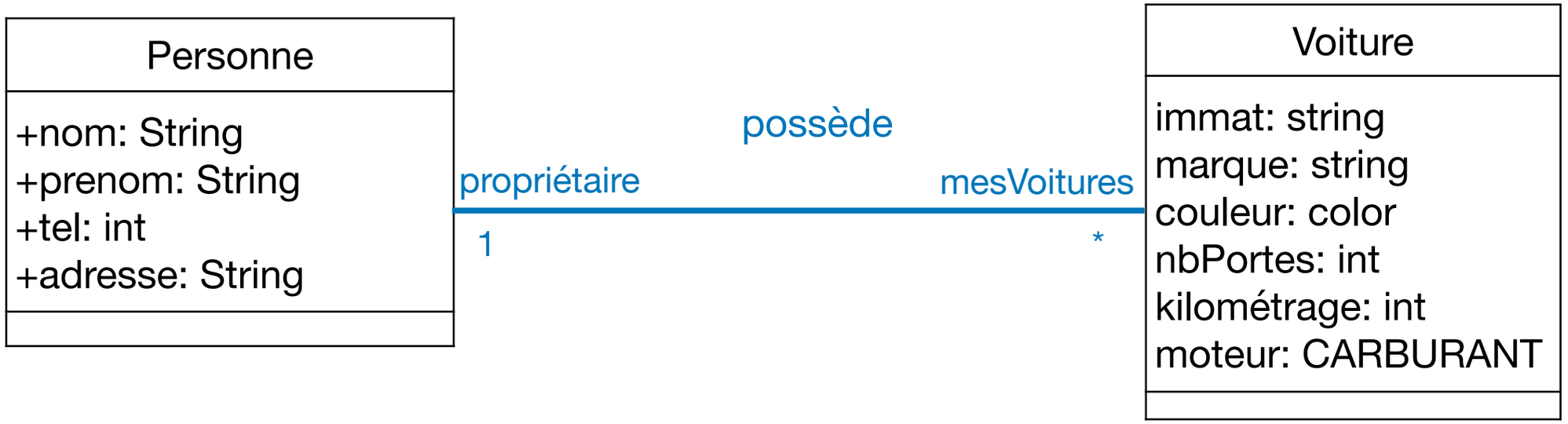


C

Relation entre classes

Types des attributs

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



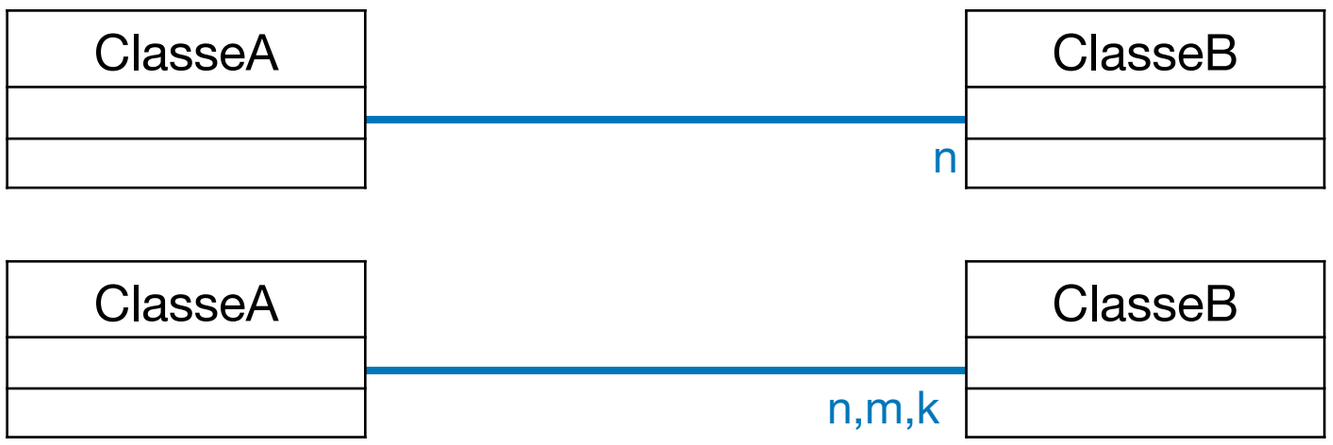
Associations

Multiplicités

A

Associations

Multiplicités



A

Associations

Multiplicités

Exactement n

Exactement n, m ou k

Entre n et m

A

Associations

Multiplicités

Exactement n

Exactement n, m ou k

Entre n et m

Au moins n

A

Associations

Multiplicités

Exactement n

Exactement n, m ou k

Entre n et m

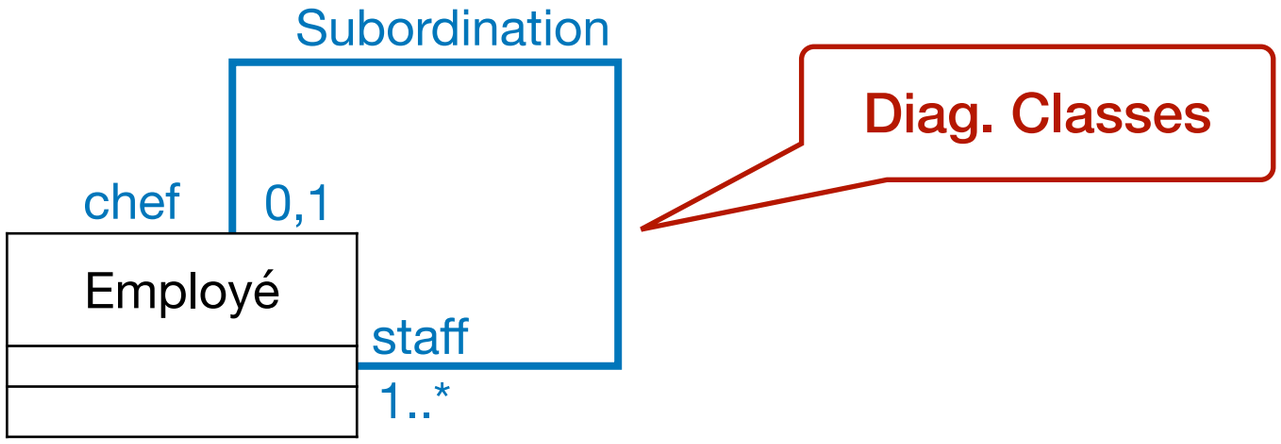
Au moins n

Plusieurs

A

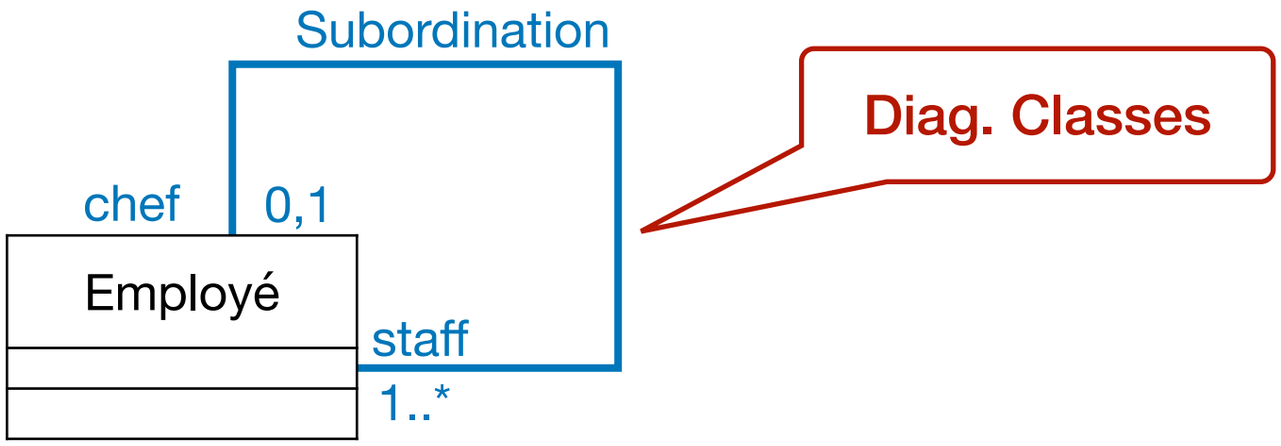
Associations

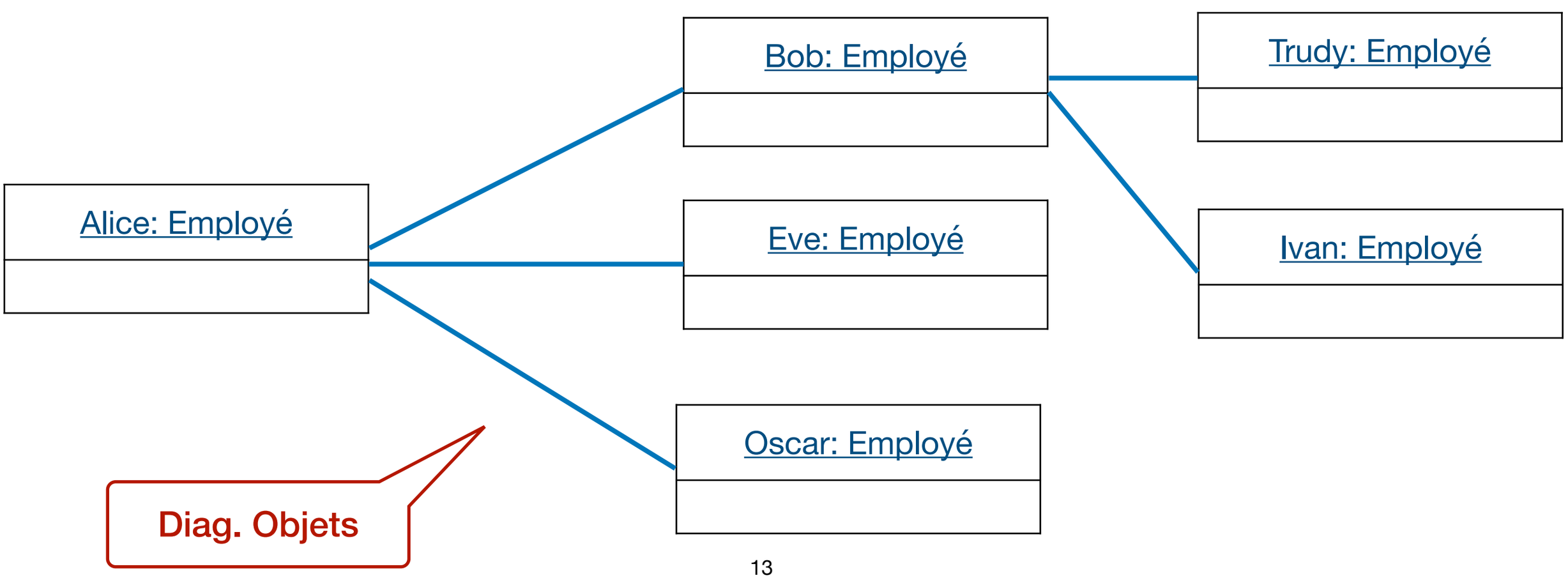
Réflexives



Associations

Réflexives



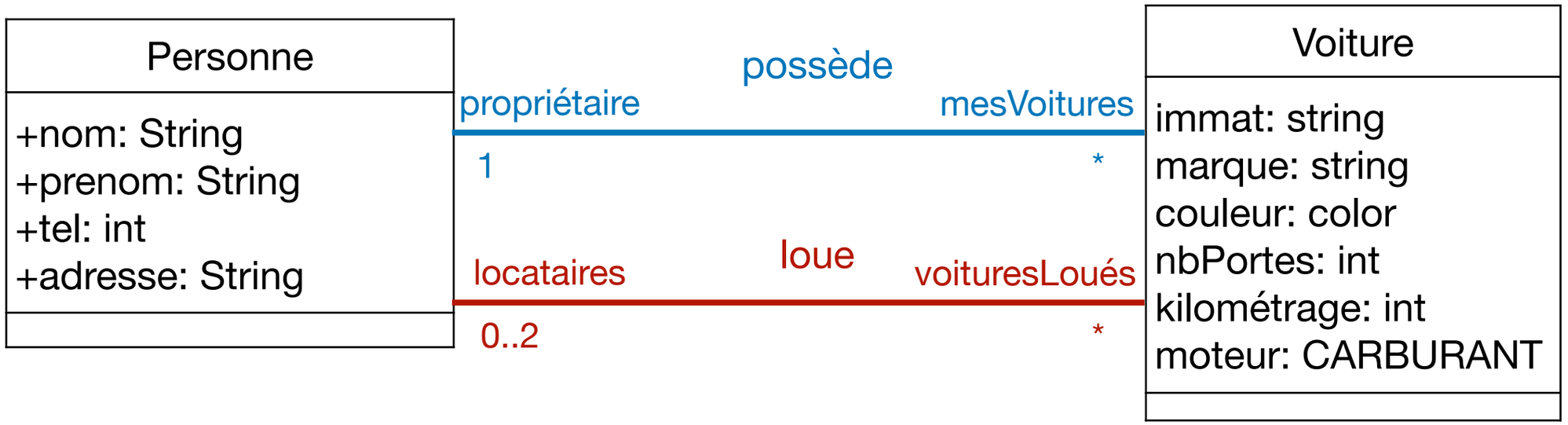


Associations

Association Multiple

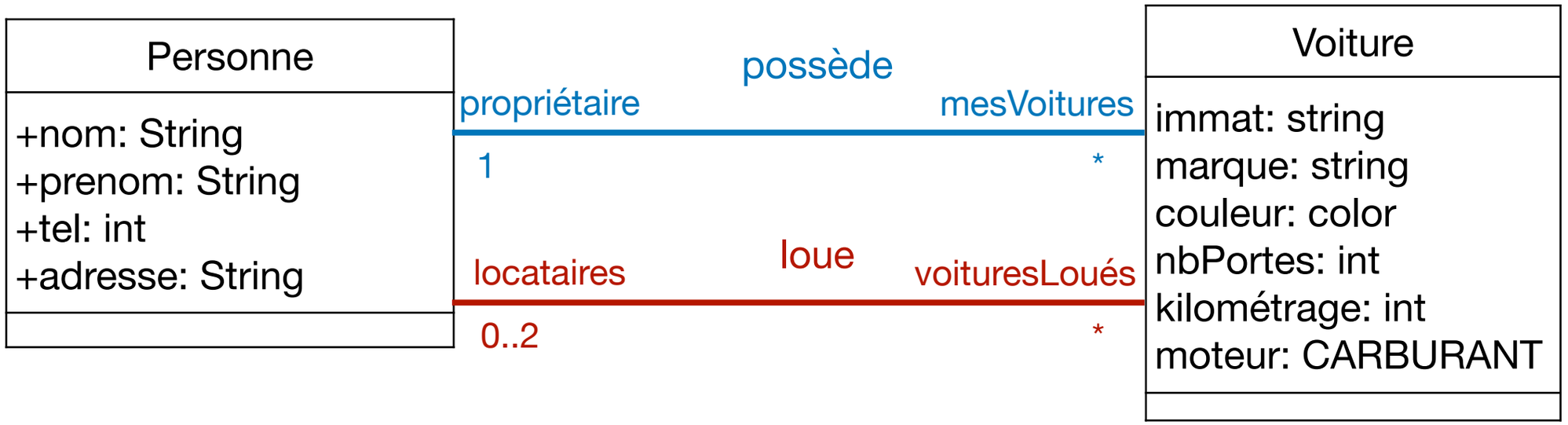
Associations

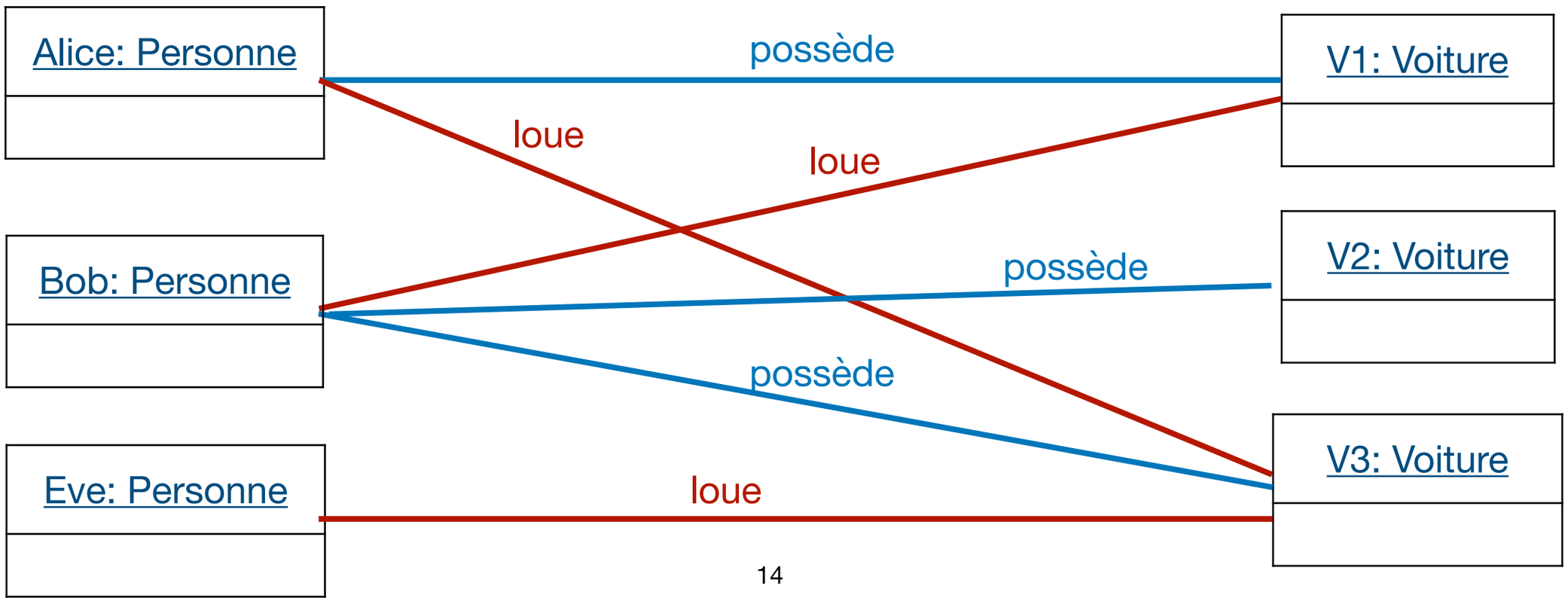
Association Multiple



Associations

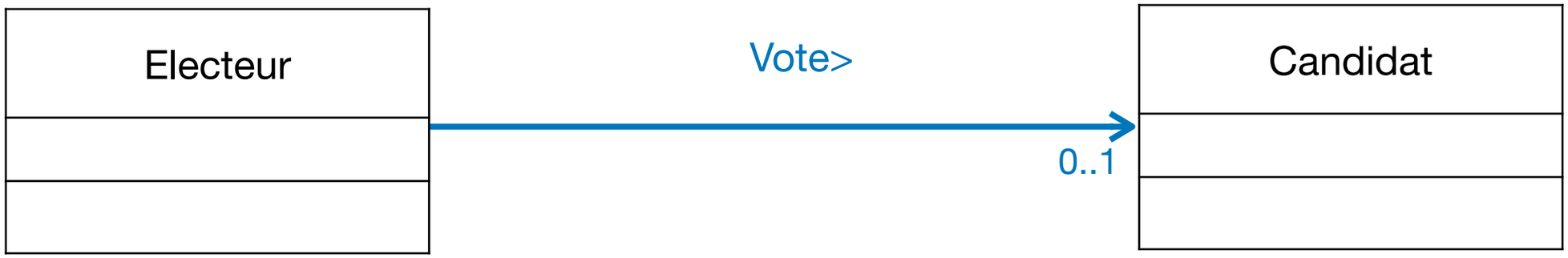
Association Multiple





Associations

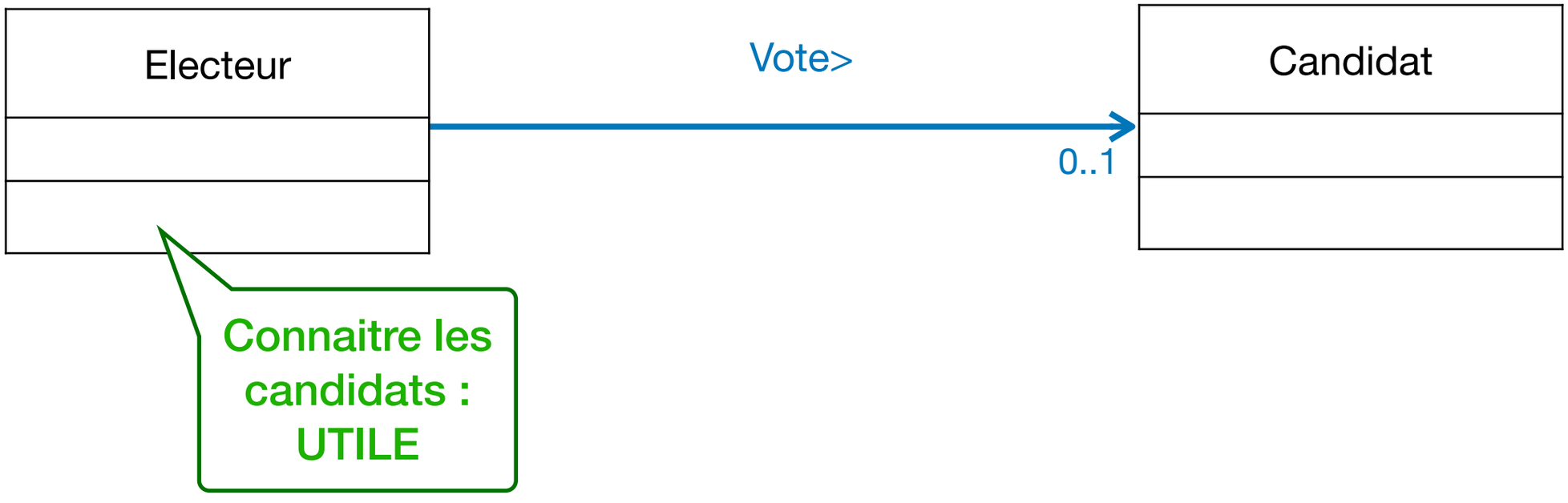
La navigation



C

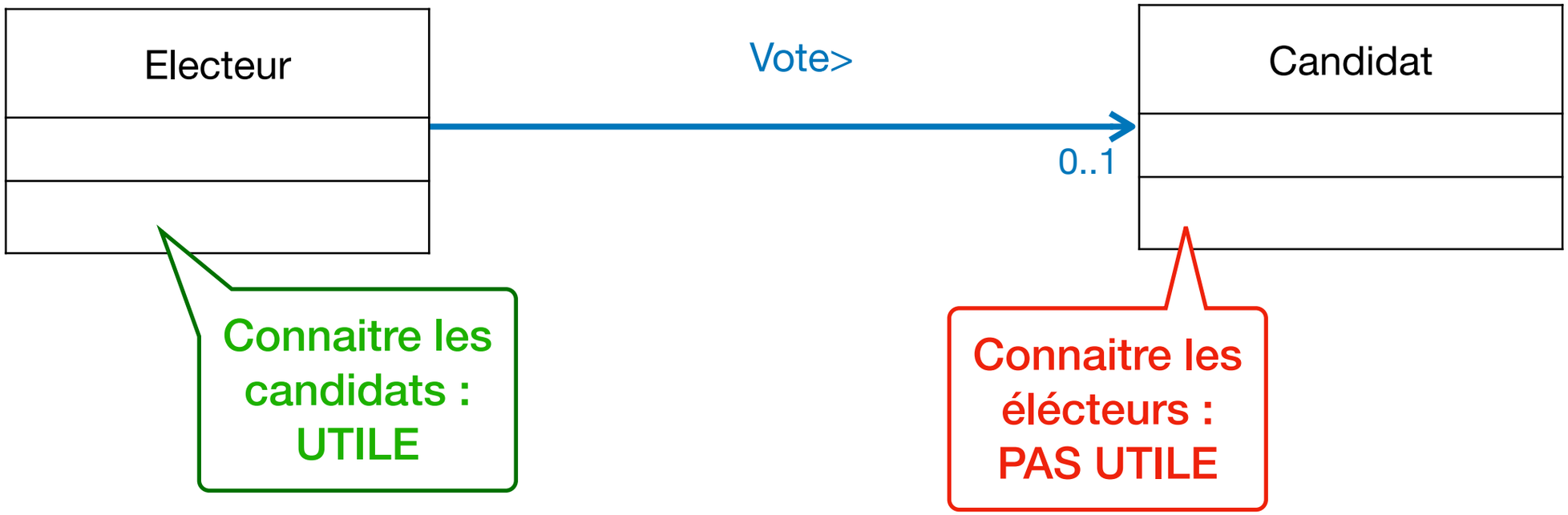
Associations

La navigation



Associations

La navigation



Classe-Association

Diagramme de classes

Classe-Association

Diagramme de classes

Date de location ?

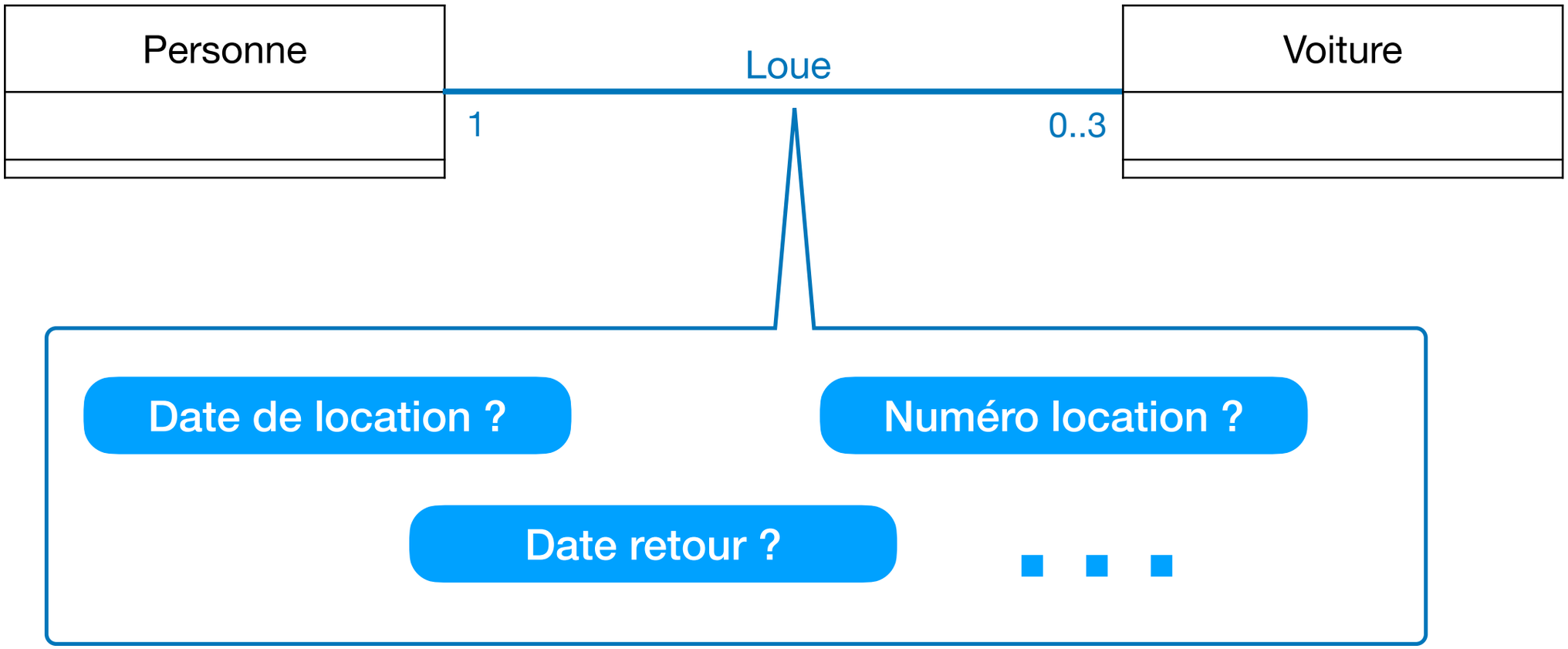
Numéro location ?

Date retour ?

. . .

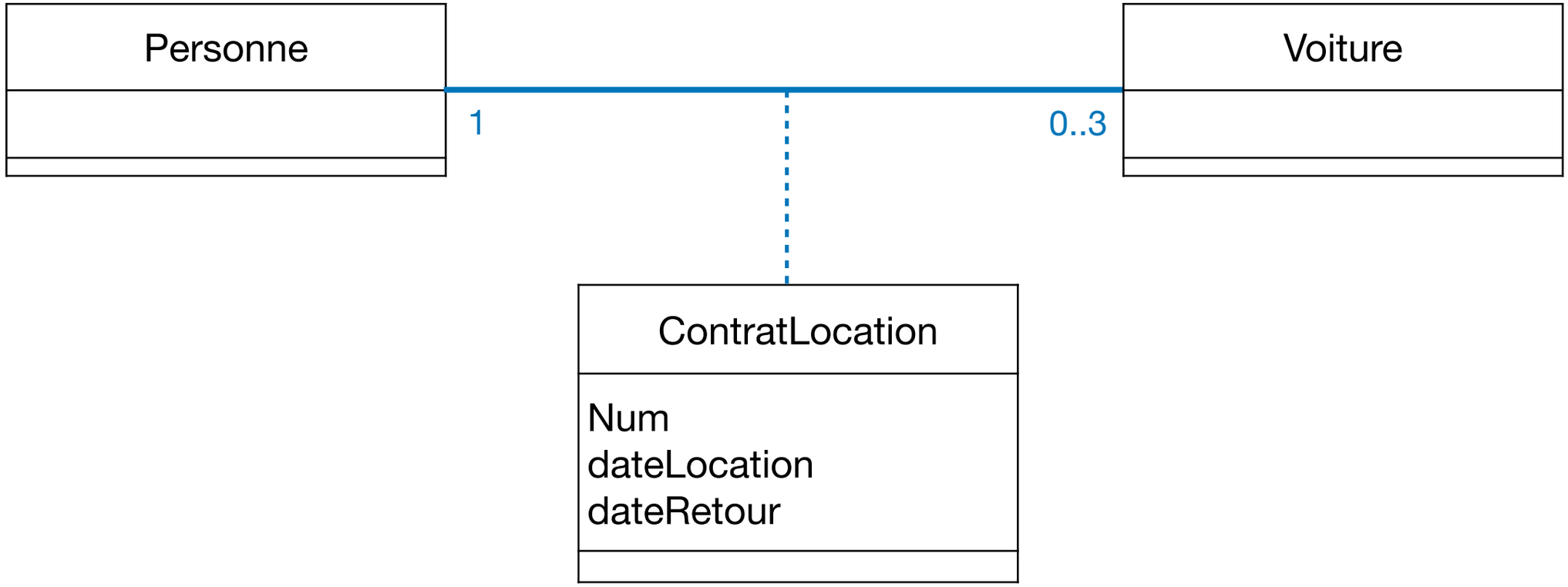
Classe-Association

Diagramme de classes



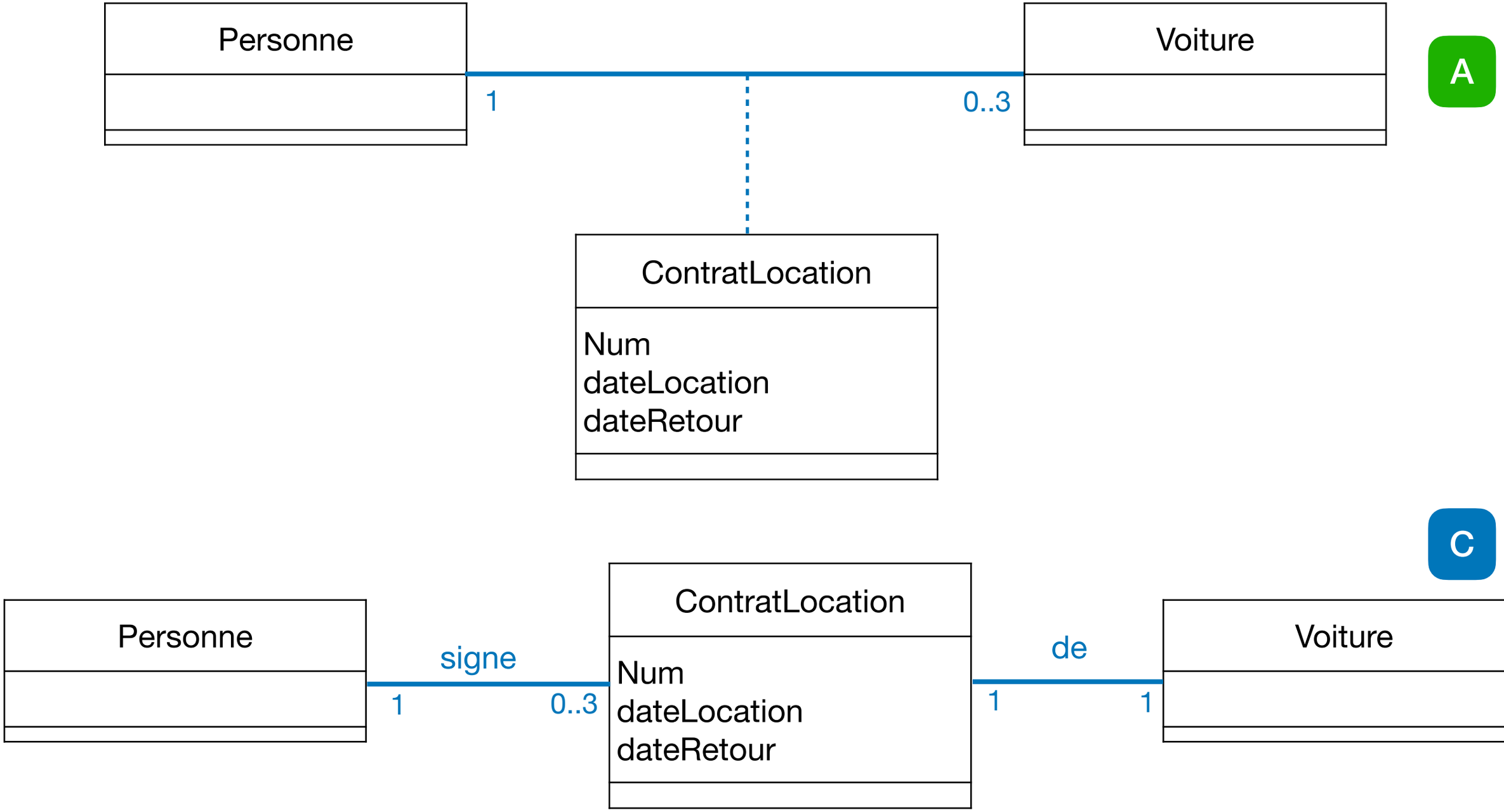
Classe-Association

Diagramme de classes



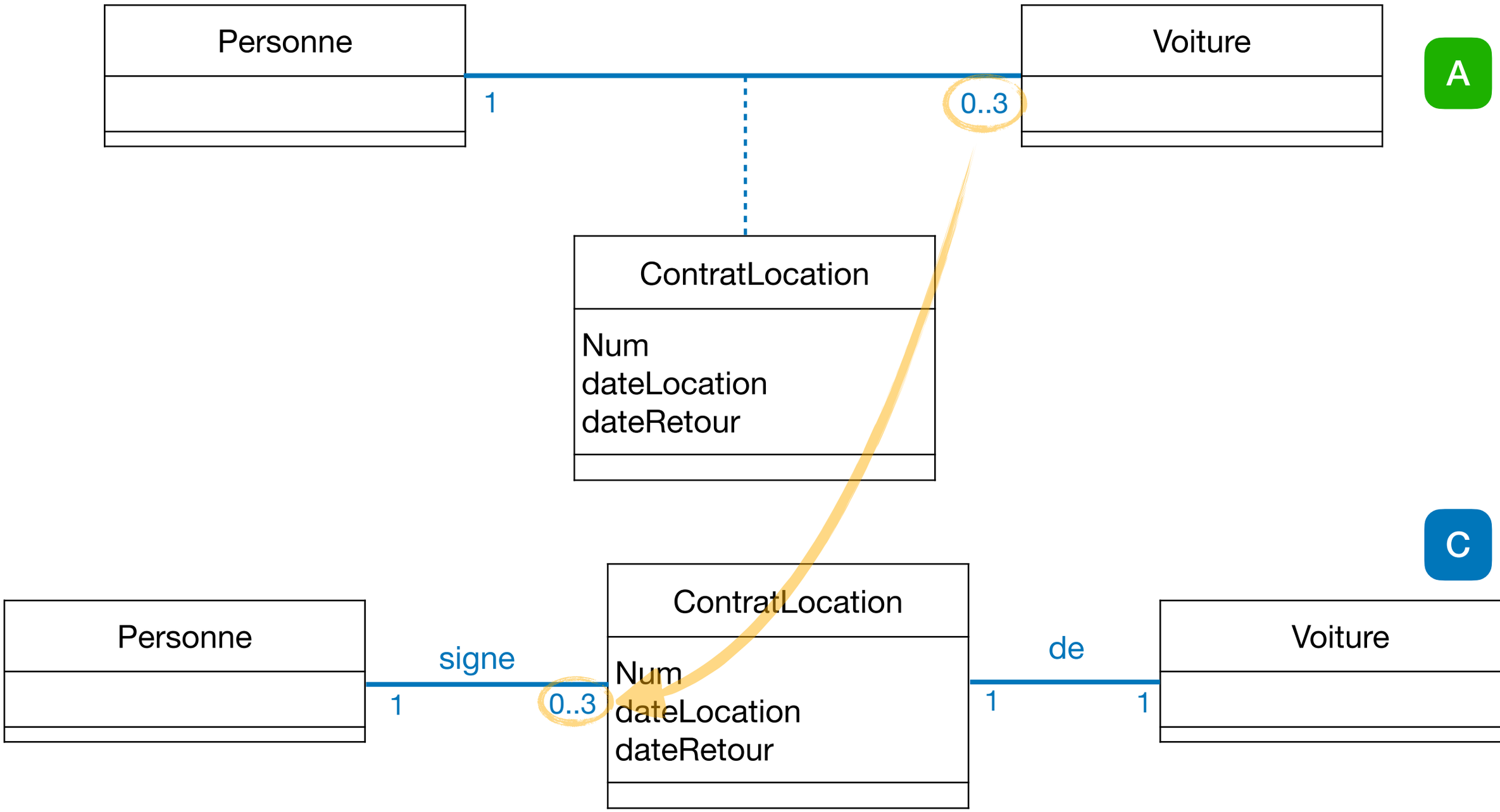
Classe-Association

Diagramme de classes



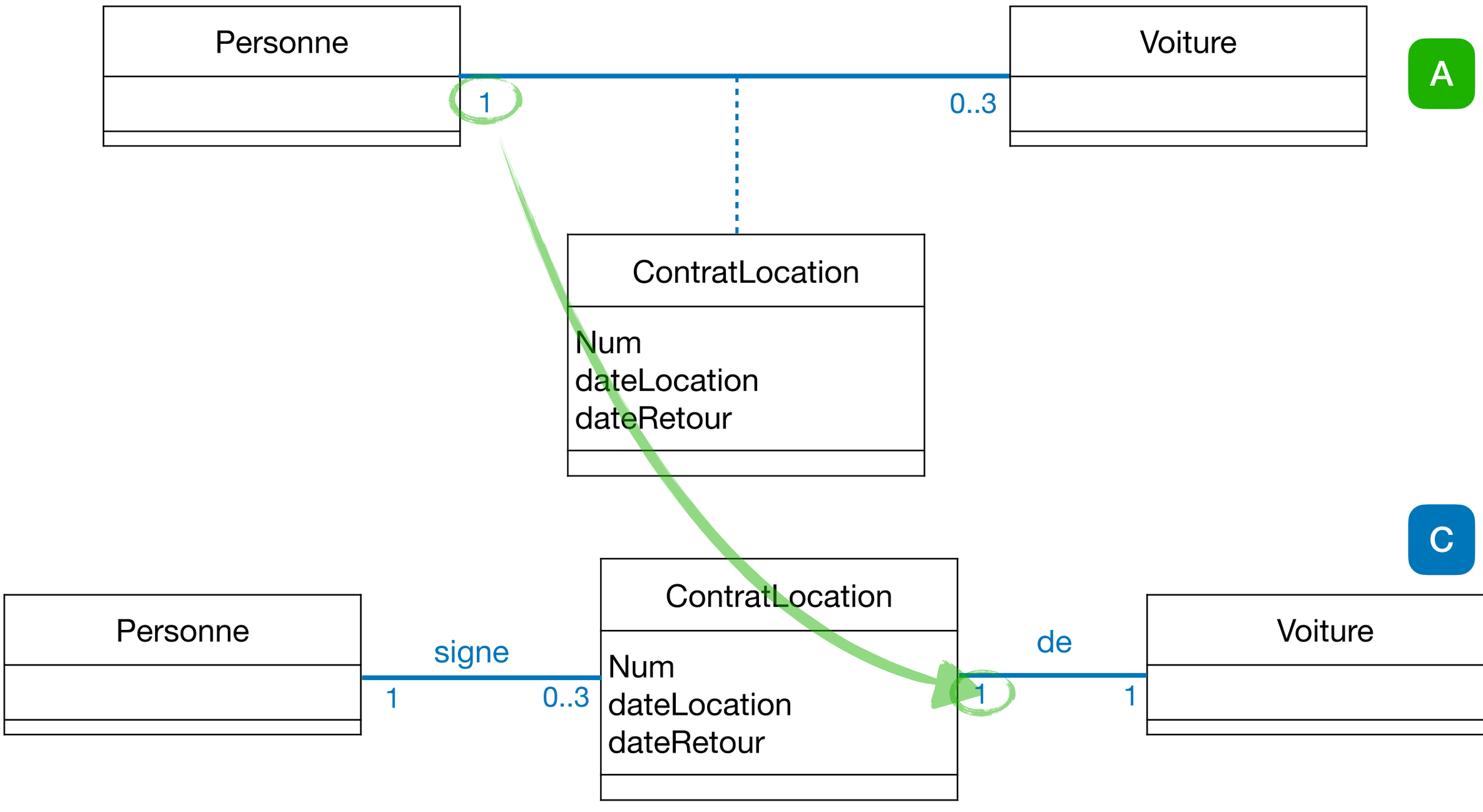
Classe-Association

Diagramme de classes



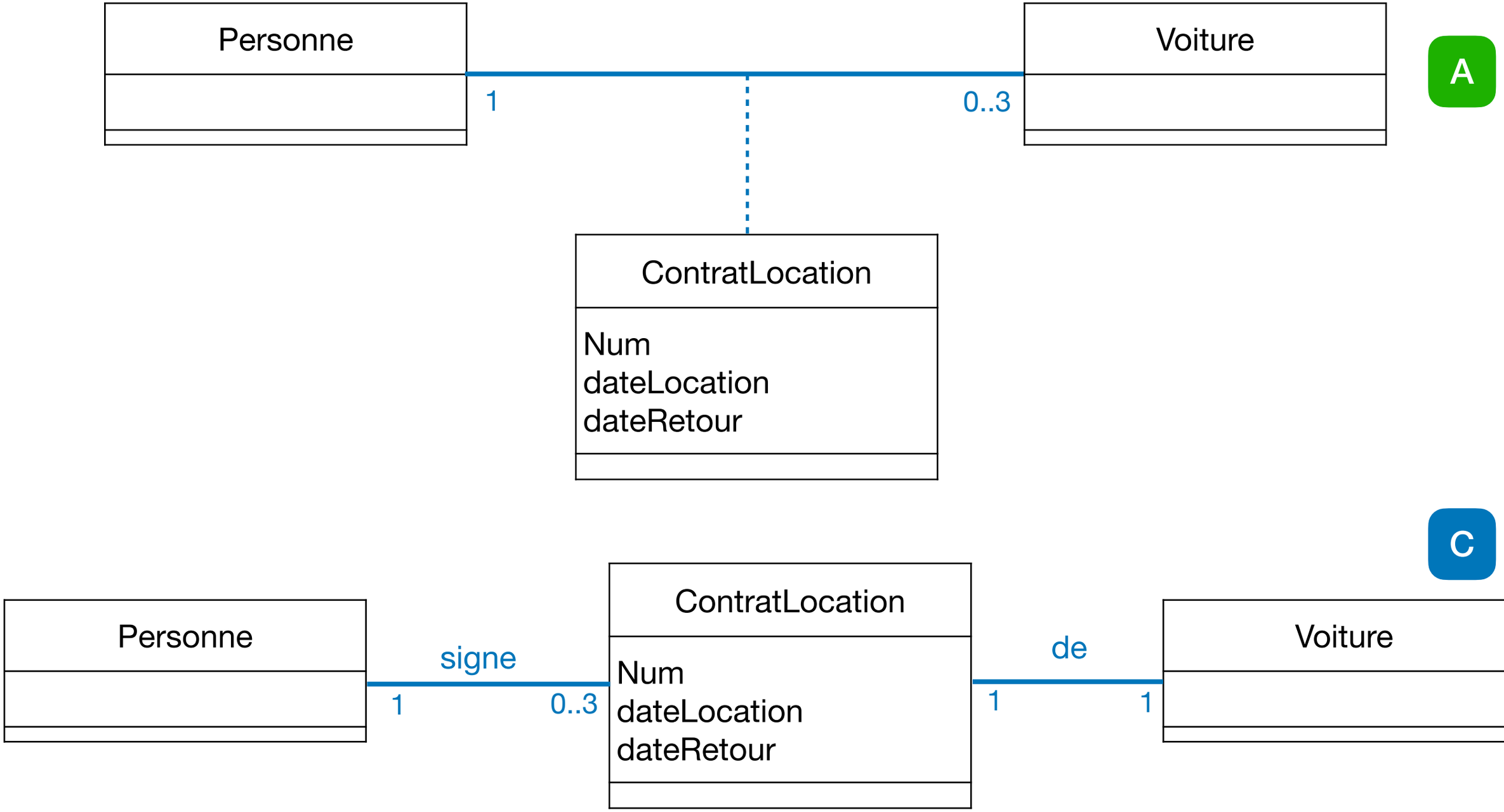
Classe-Association

Diagramme de classes



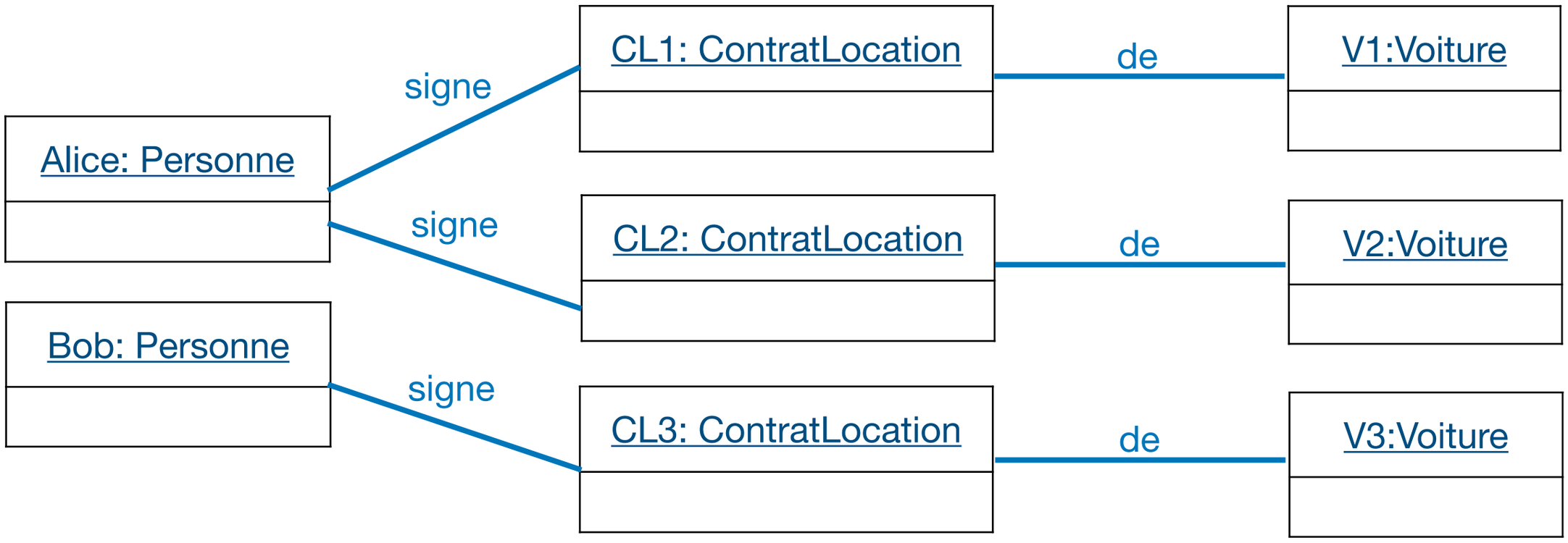
Classe-Association

Diagramme de classes



Classe-Association

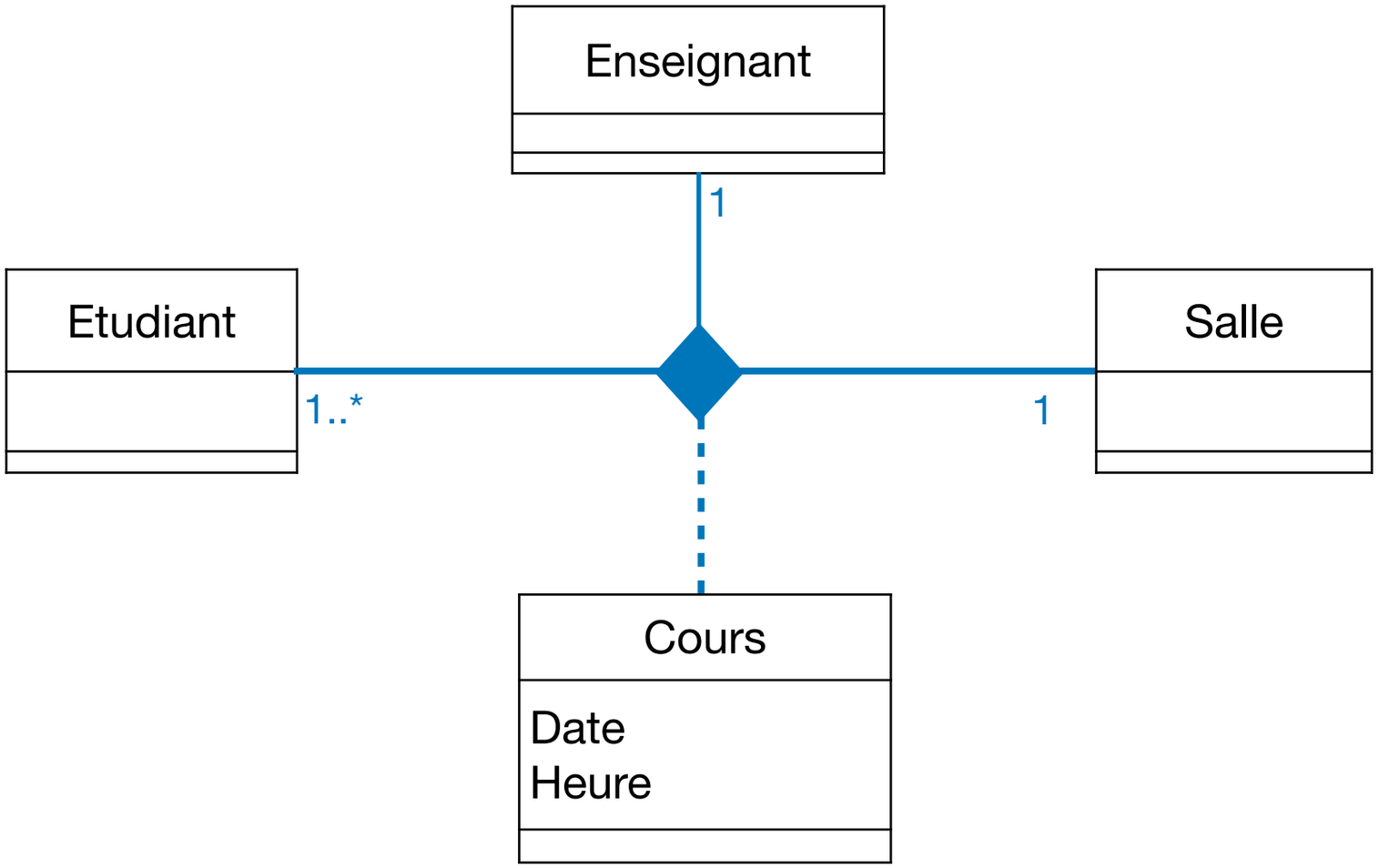
Diagramme d'objets



Association n-aire

Diagramme de classes

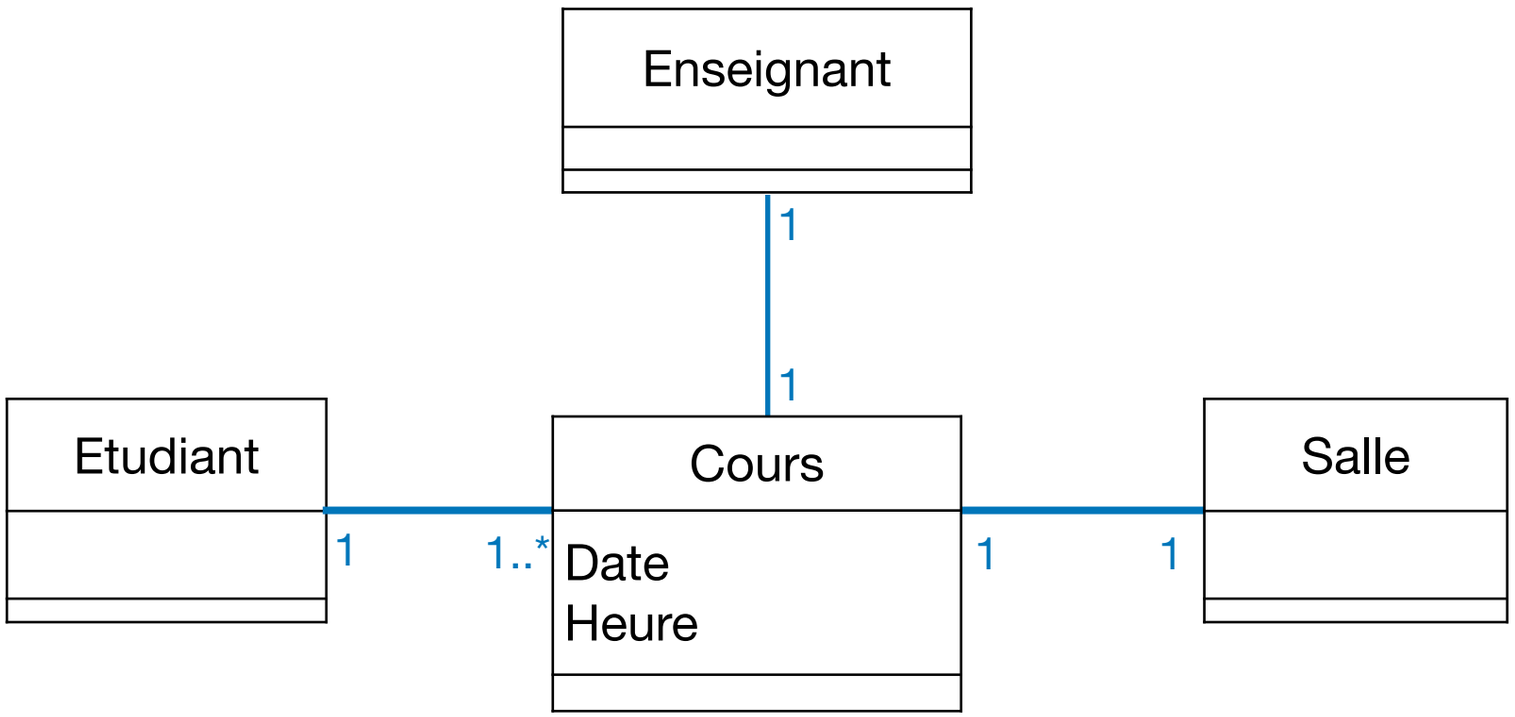
A



Association n-aire

Diagramme de classes

C



Mais aussi…

voir concepts OO (cours2)

- **La spécialisation et la généralisation (concept n°5)**

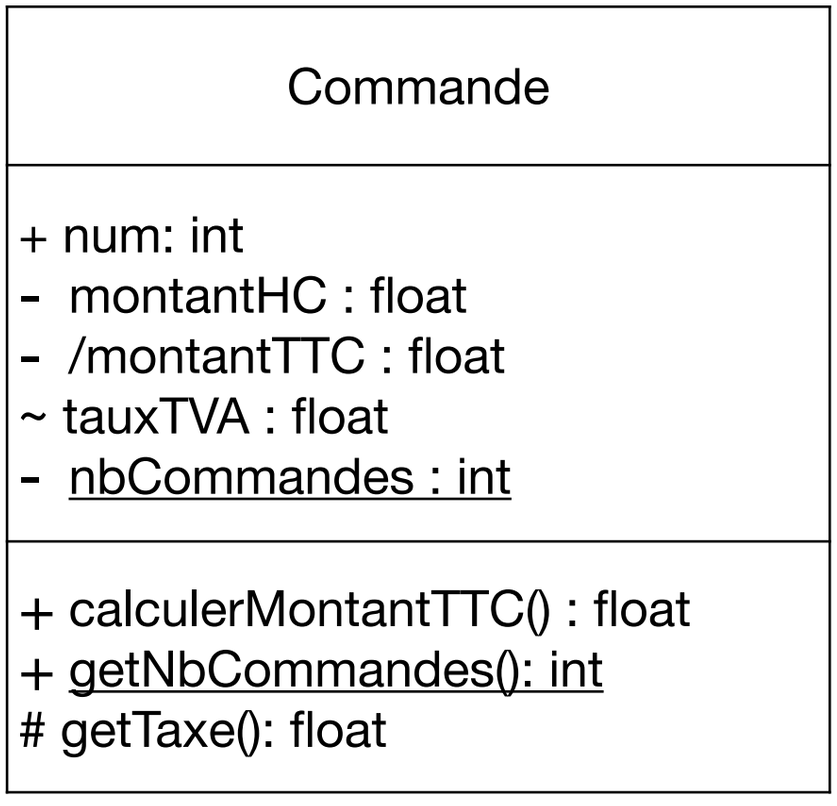
- **Les classes abstraites et concrètes (concept n°7)**

- **La composition et l’agrégation (concept n°9)**

Classes

Notations avancées

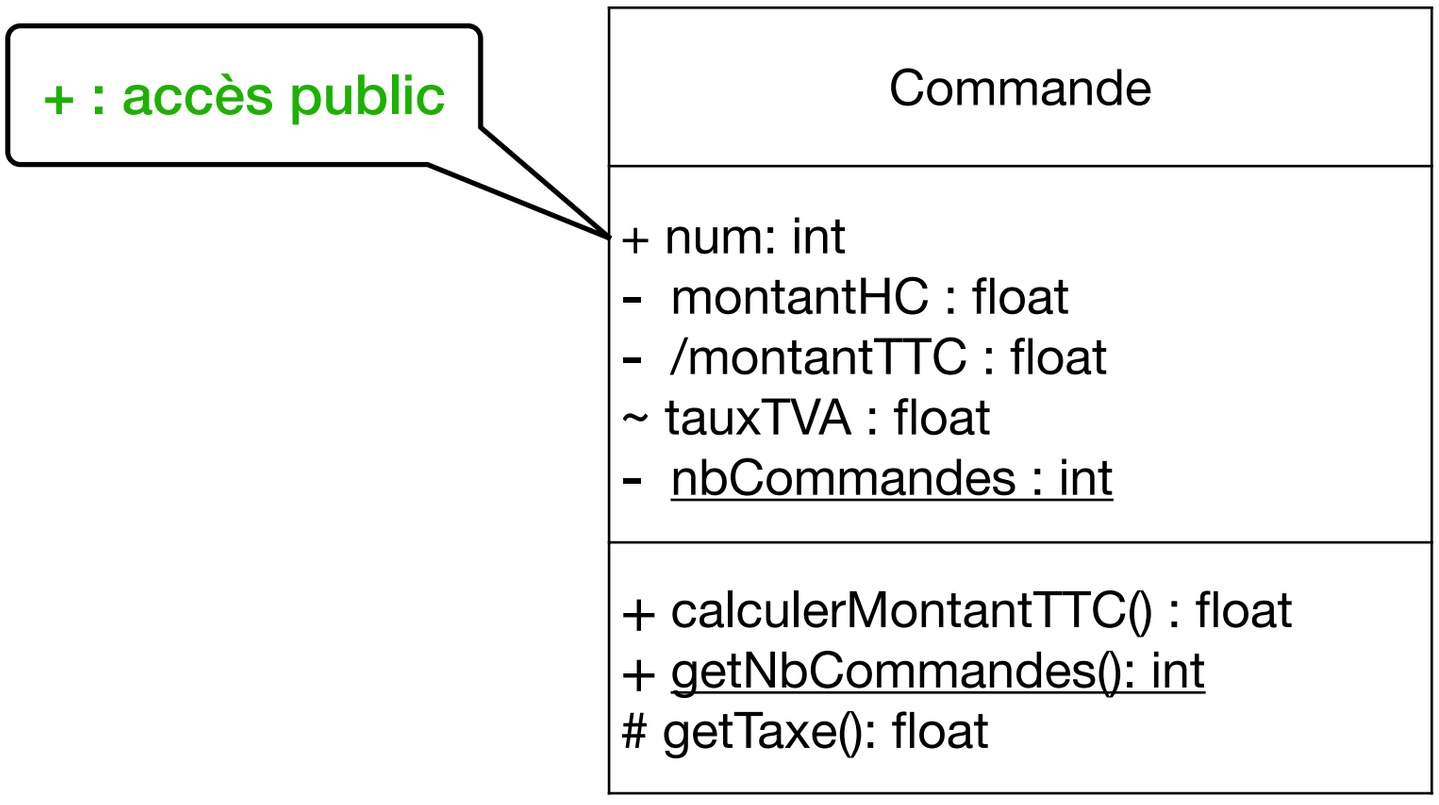
C



Classes

Notations avancées

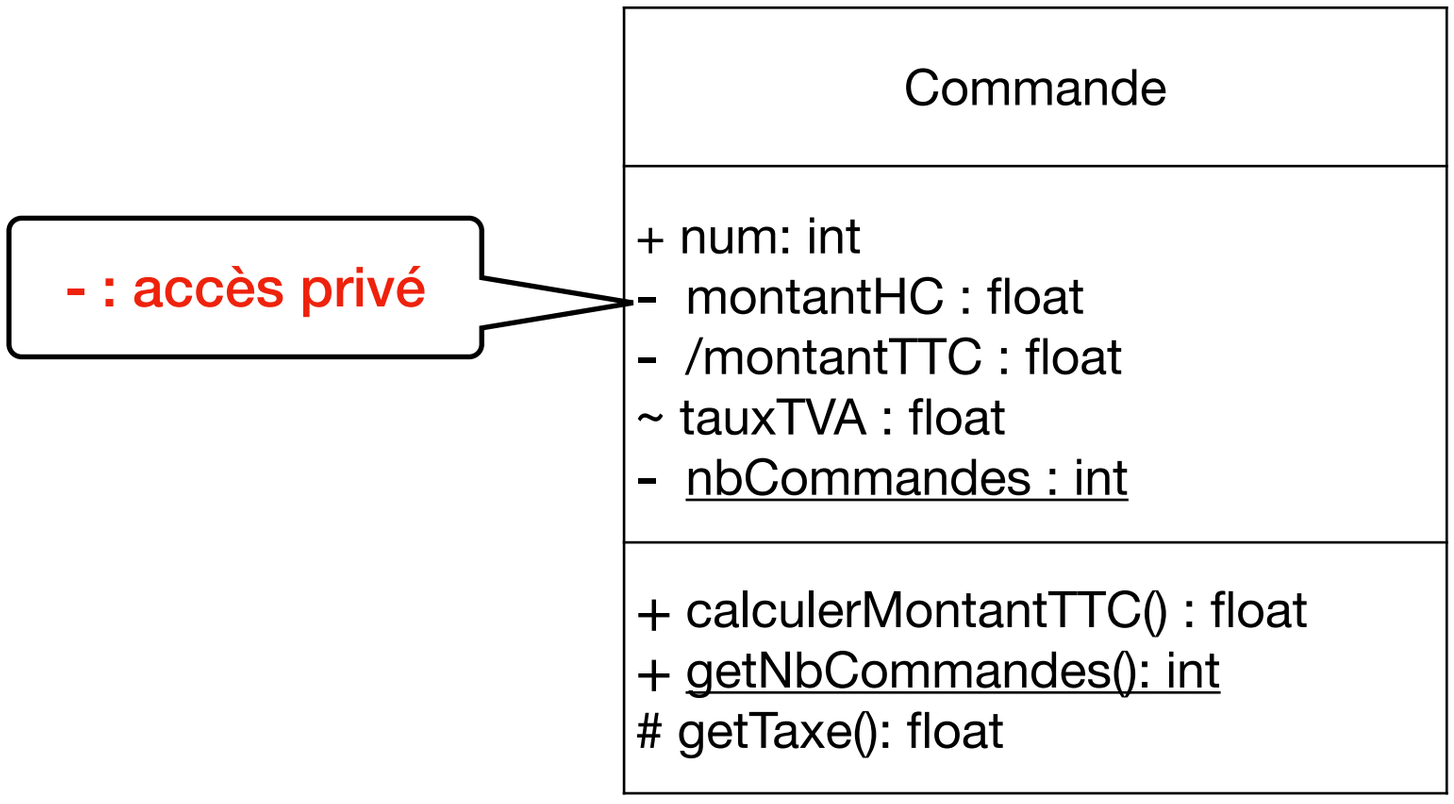
C



Classes

Notations avancées

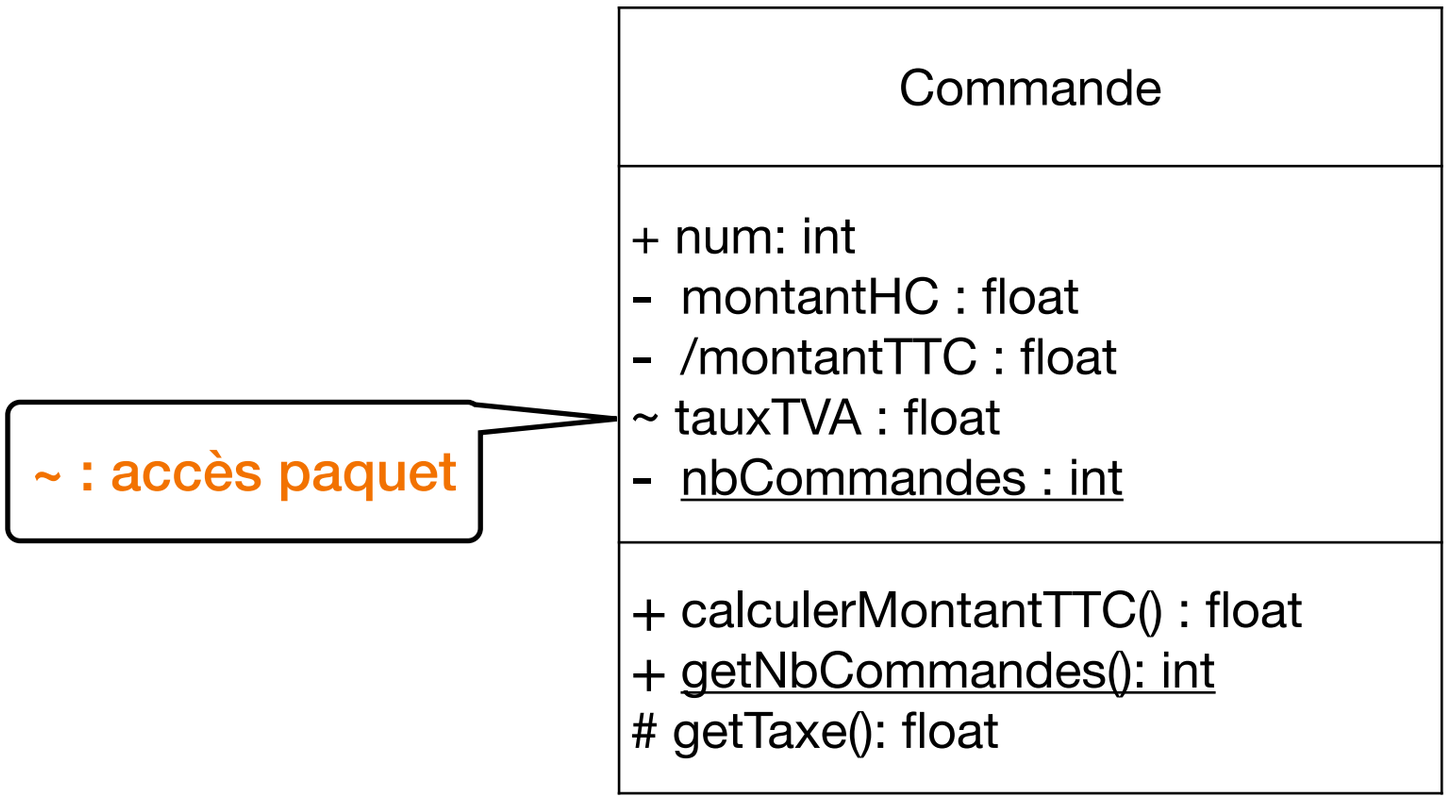
C



Classes

Notations avancées

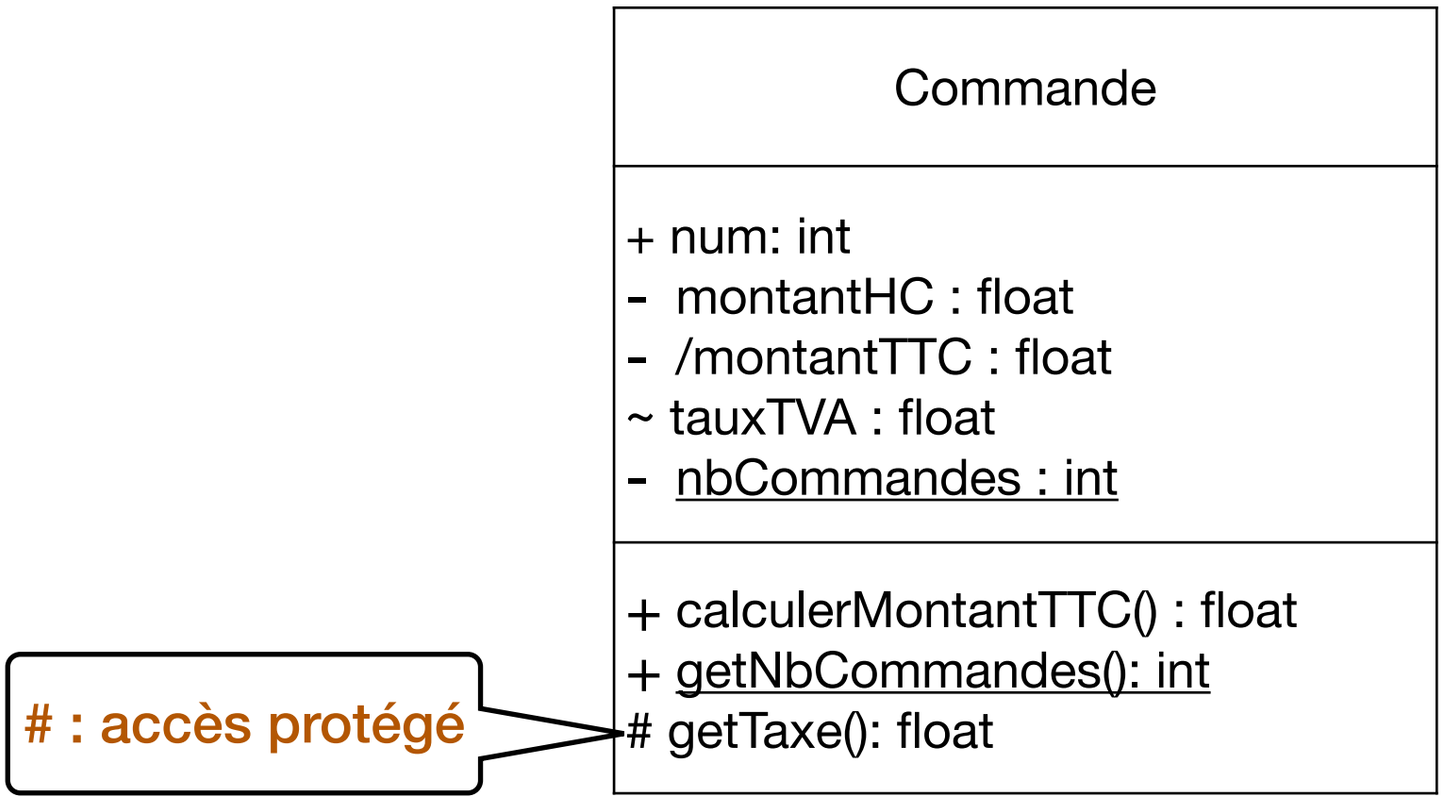
C



Classes

Notations avancées

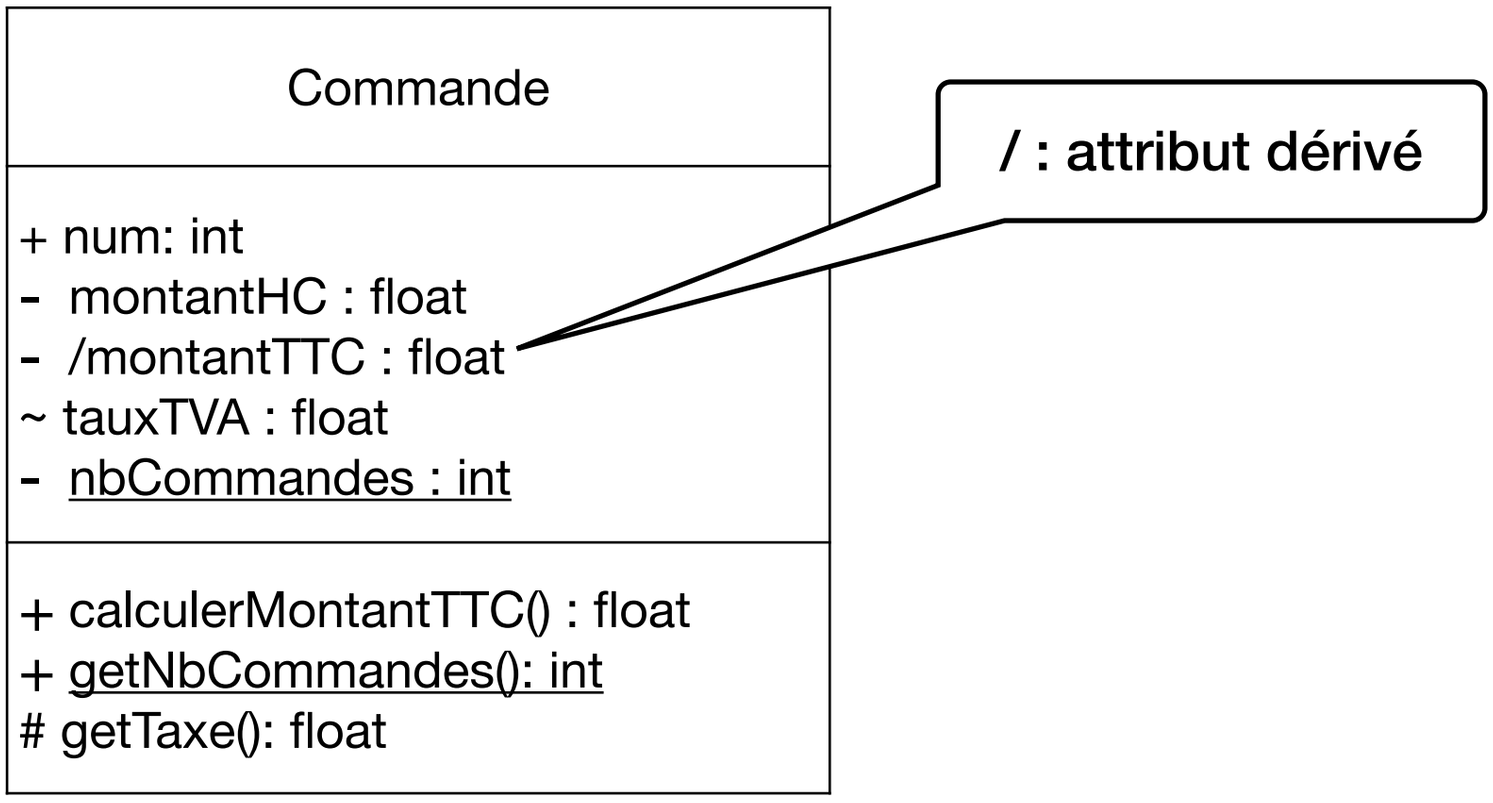
C



Classes

Notations avancées

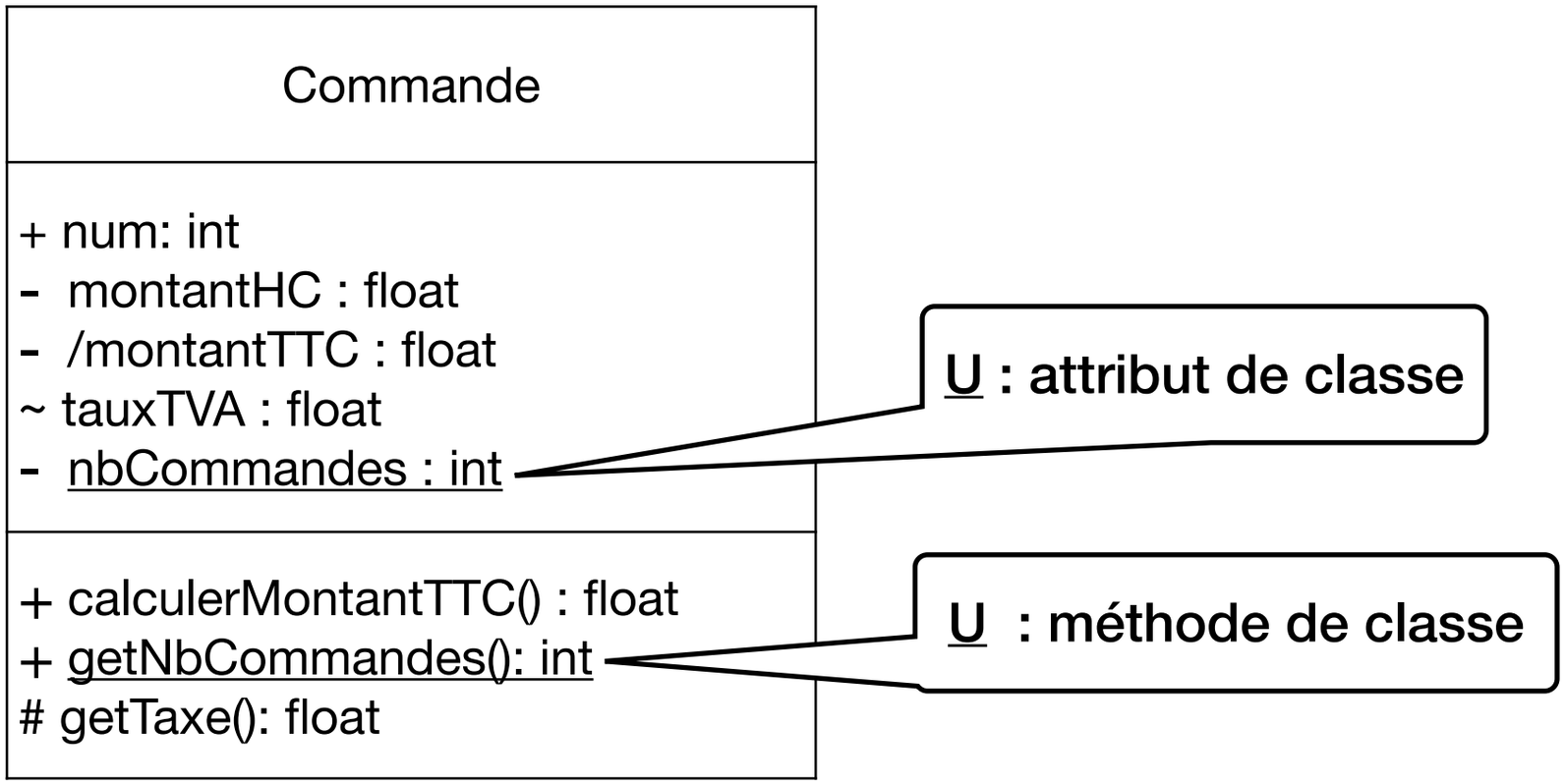
C



Classes

Notations avancées

C



Relation de dépendance

Différents stéréotypes

A

C

Relation de dépendance

Différents stéréotypes

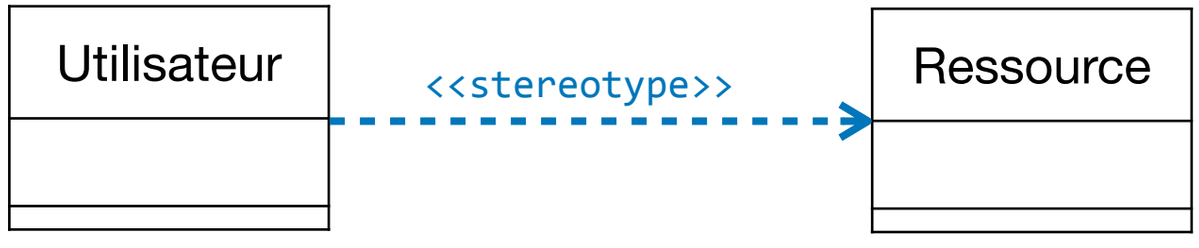
A

C

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

Relation de dépendance

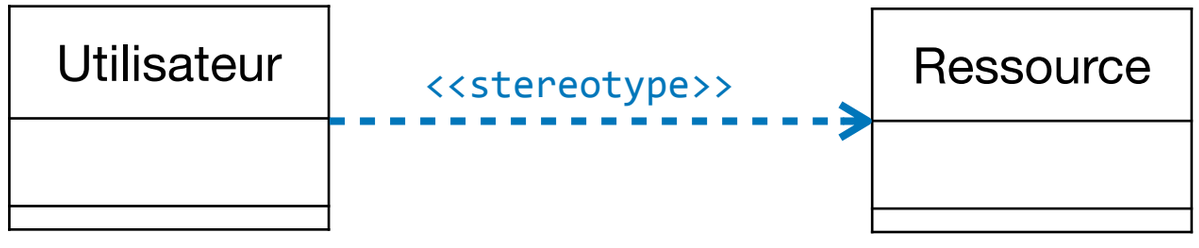
Différents stéréotypes



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

Relation de dépendance

Différents stéréotypes



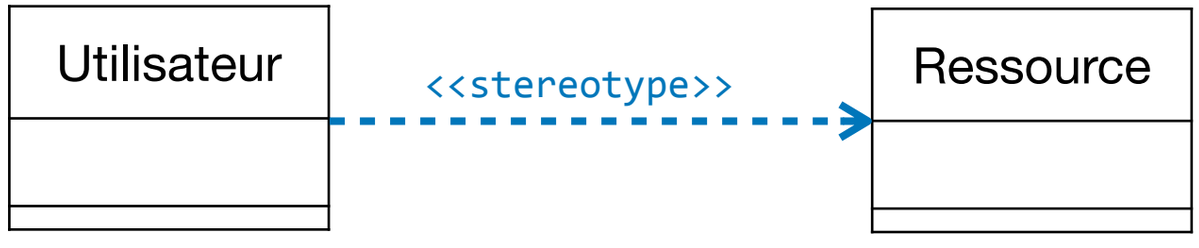
A

C

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

Relation de dépendance

Différents stéréotypes



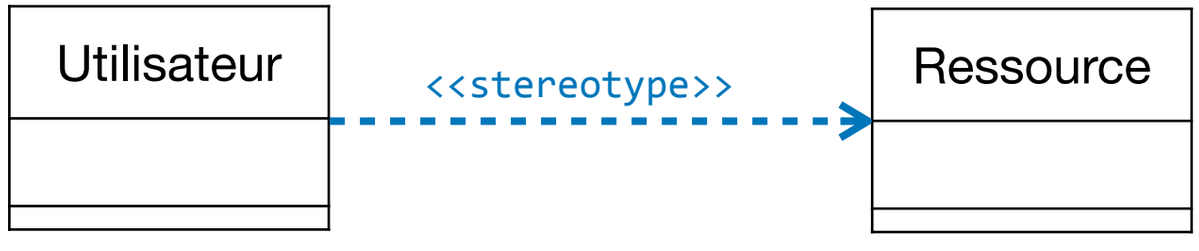
A

C

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

Relation de dépendance

Différents stéréotypes



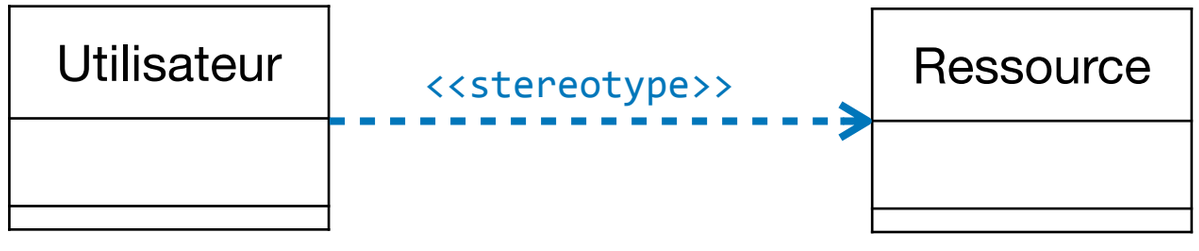
A

C

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



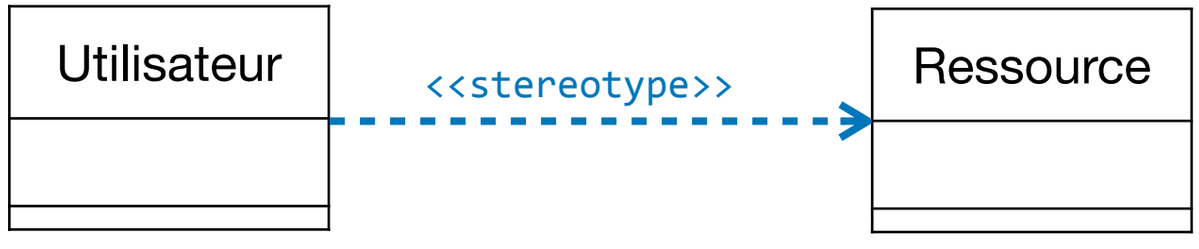
A

C

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

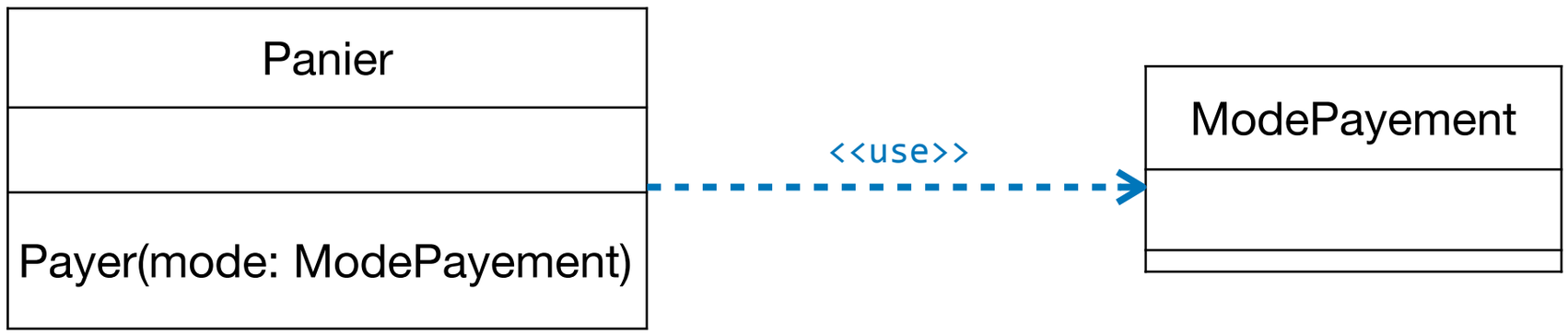
Différents stéréotypes



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

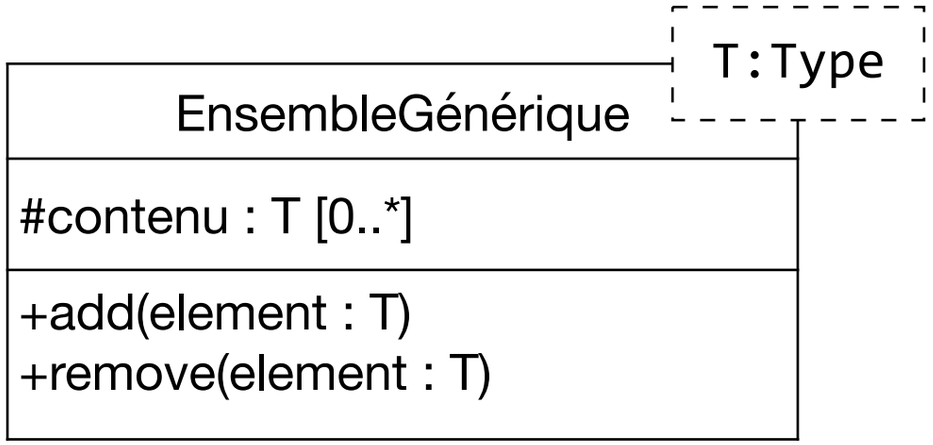
Exemple



Classes génériques

Classes template

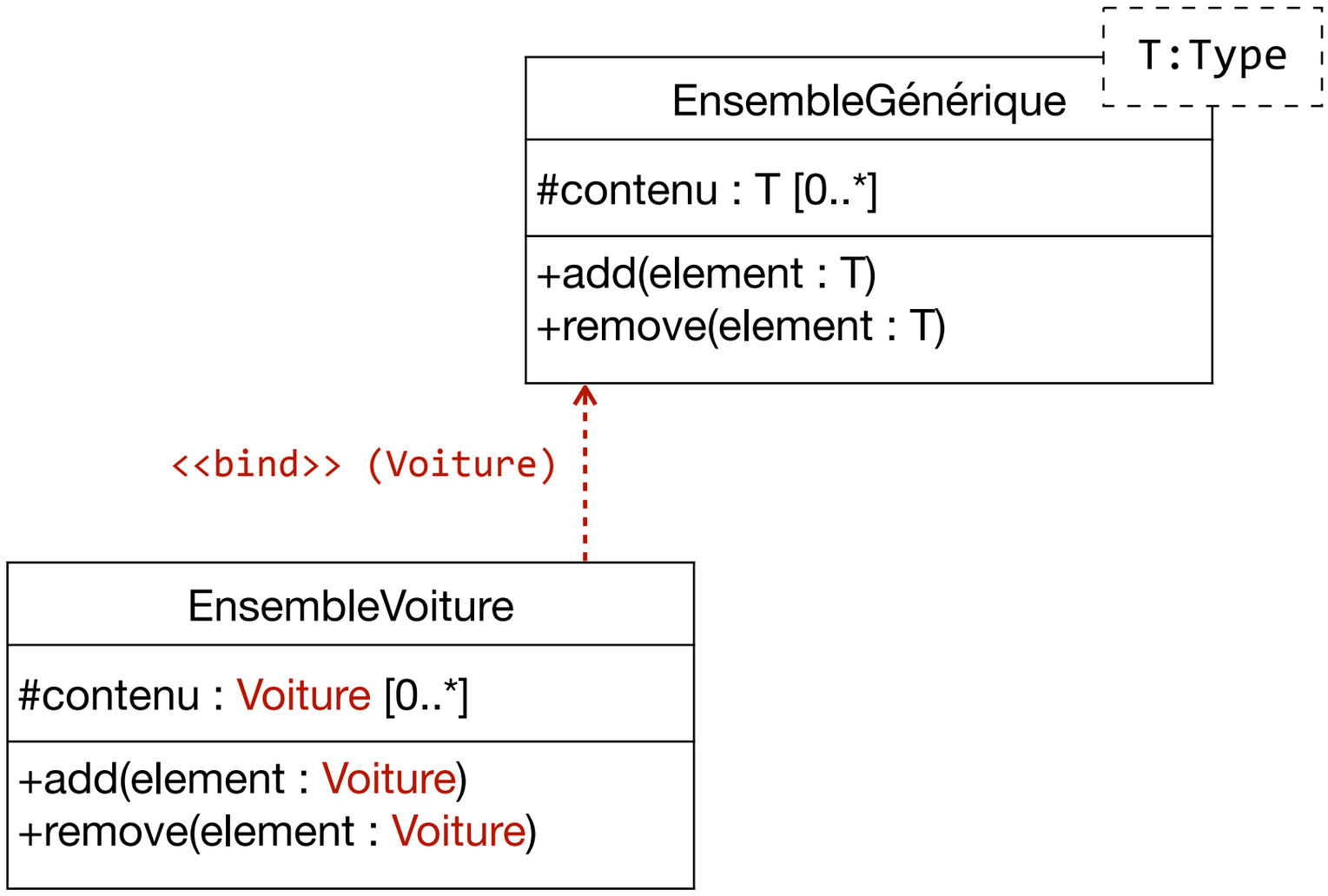
- Une classe template est un modèle de classes
- Des classes génériques avec des paramètres



Classes génériques

Classes template

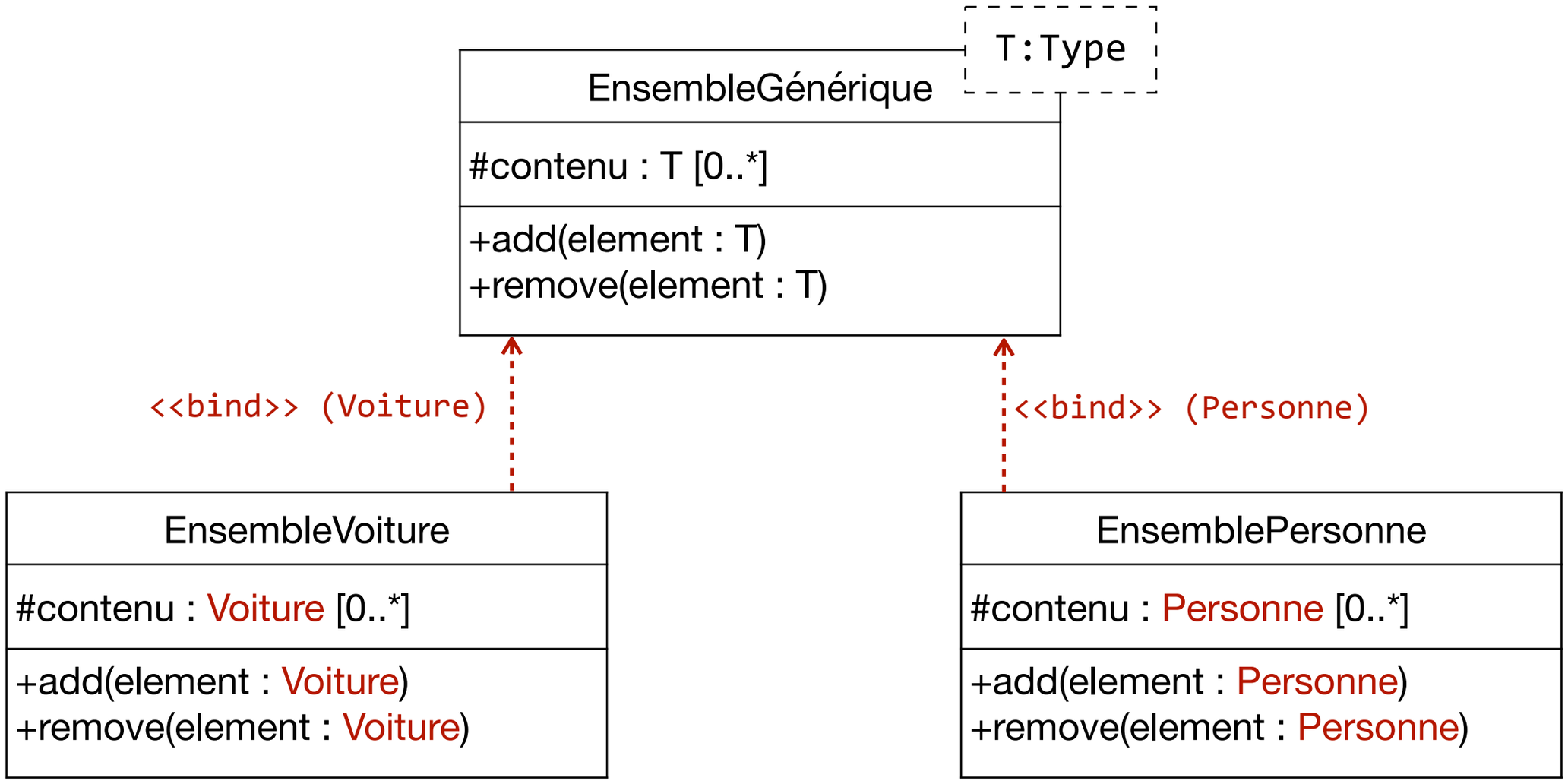
- Une classe template est un modèle de classes
- Des classes génériques avec des paramètres



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

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

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

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

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

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

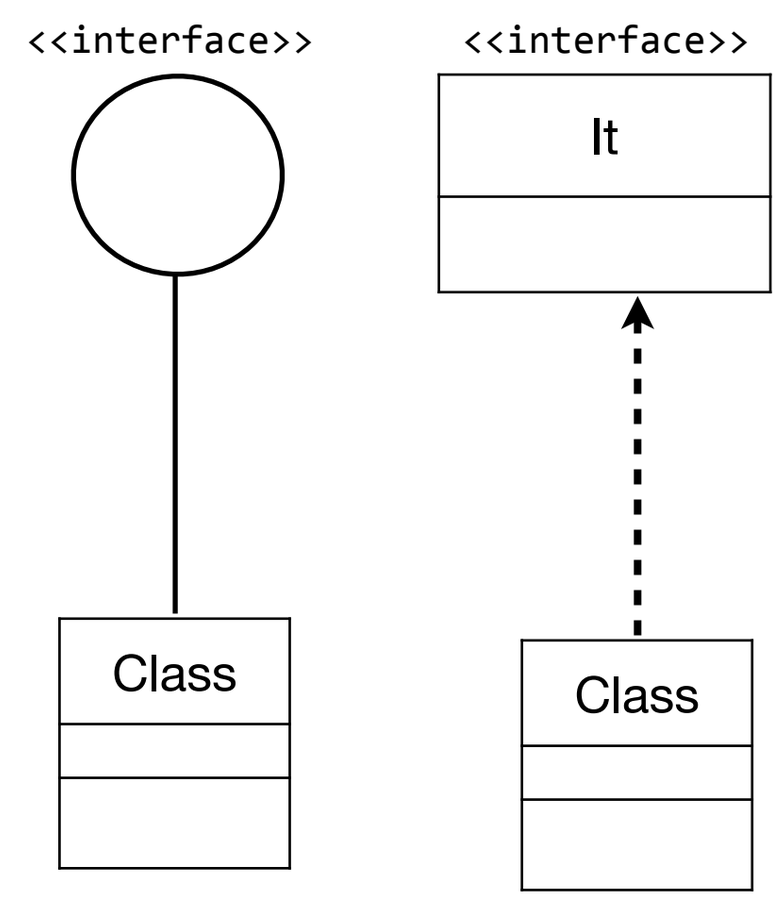
Relation de réalisation

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

A

C

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

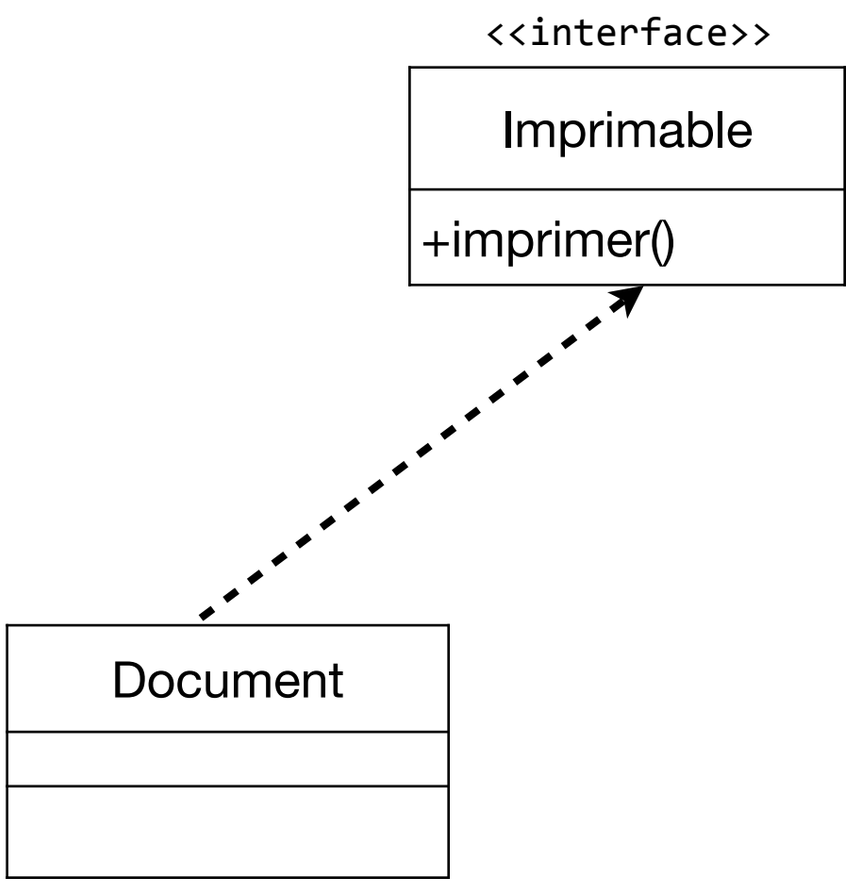


Interface

Exemple (1/2)

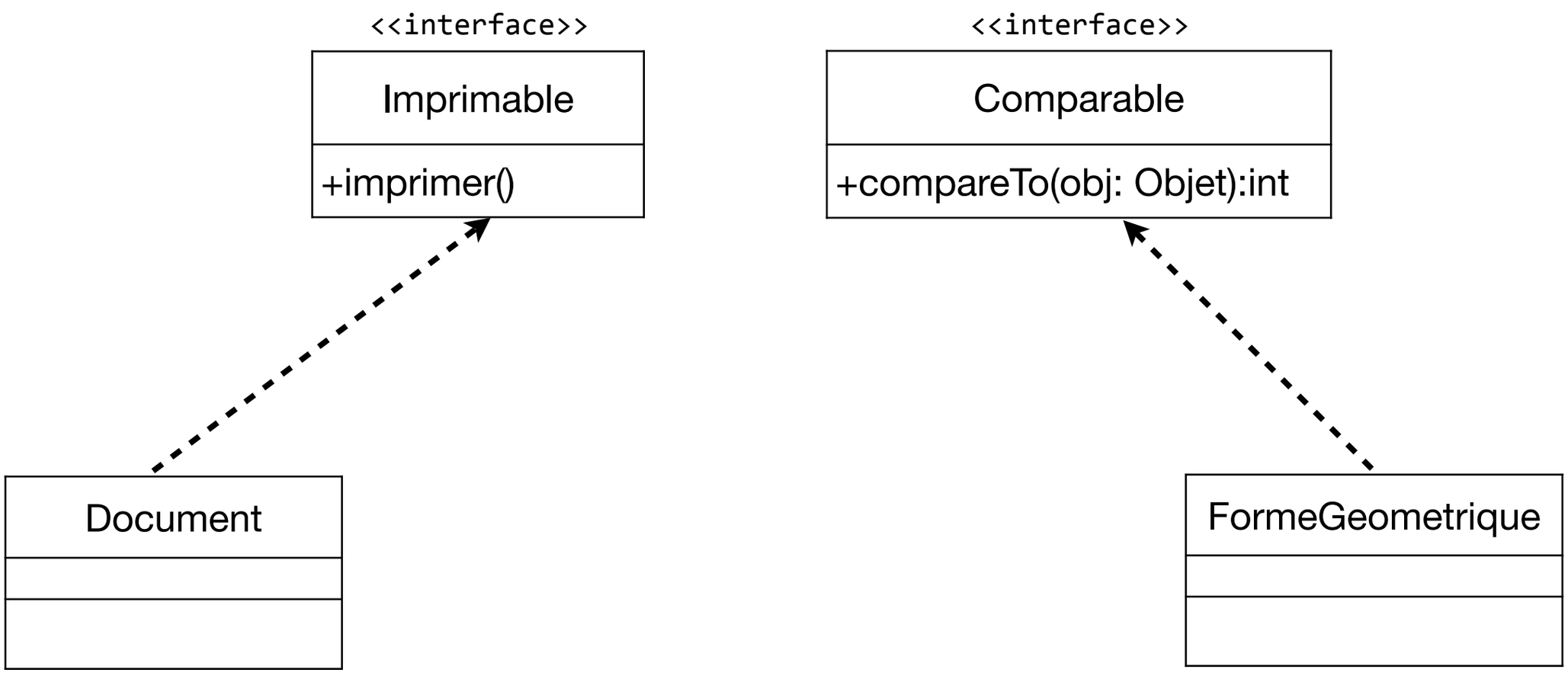
Interface

Exemple (1/2)



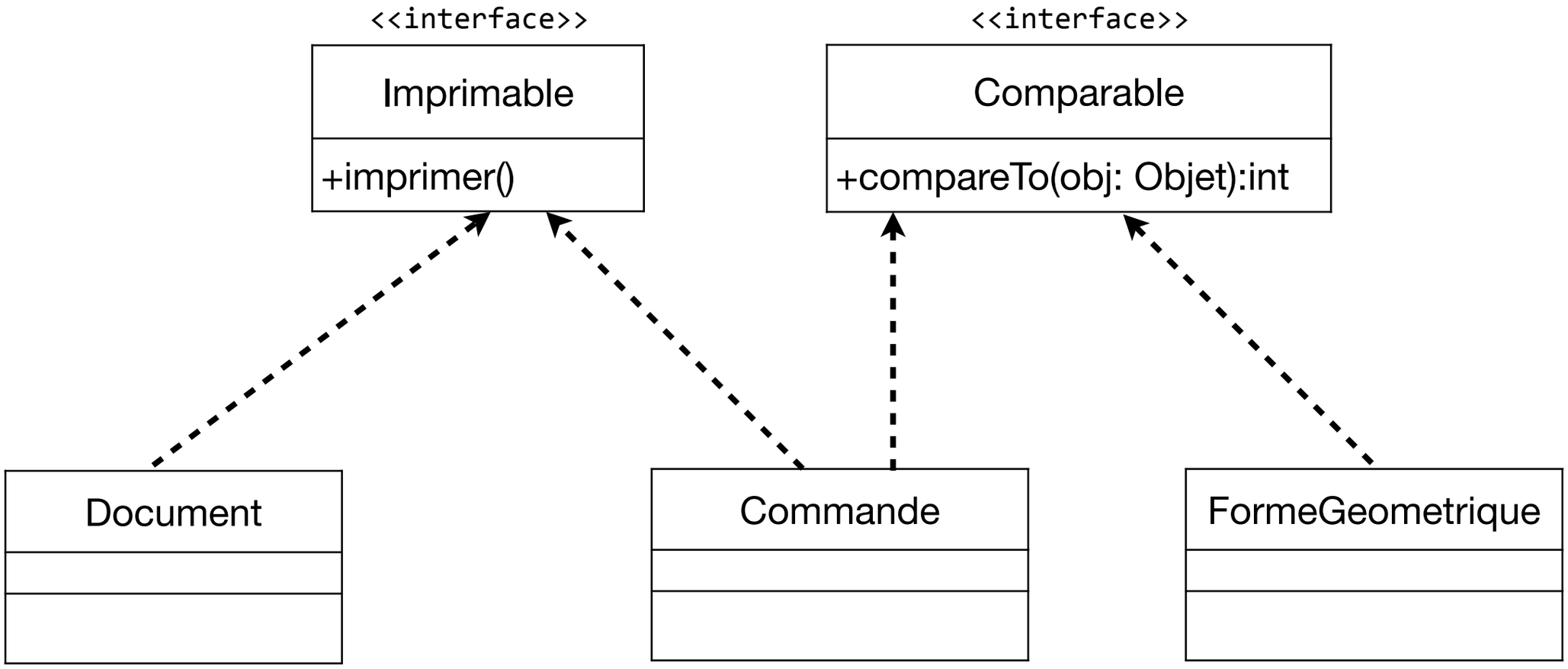
Interface

Exemple (1/2)



Interface

Exemple (1/2)



Interface

Exemple (2/2)

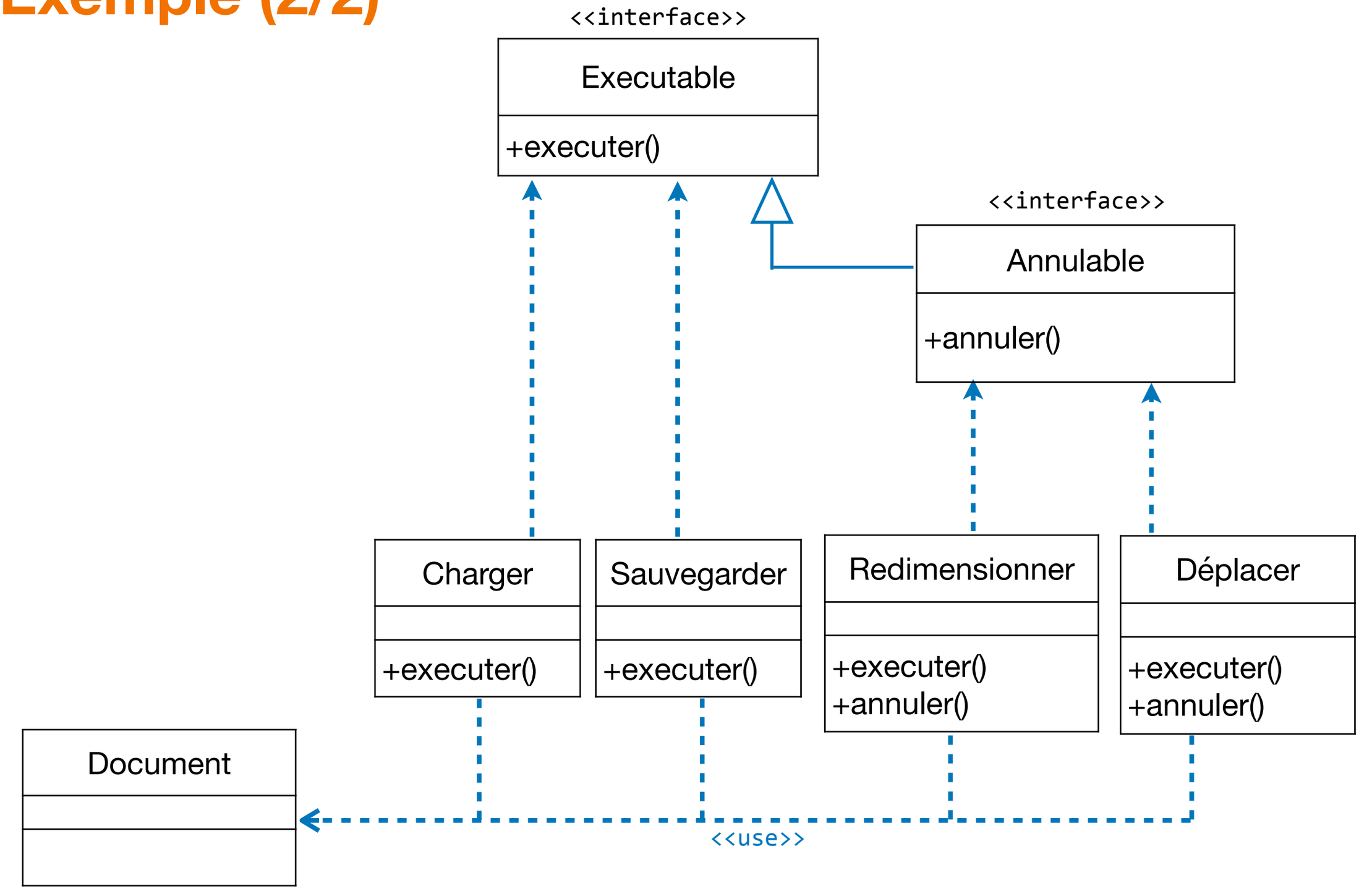


Diagramme de classes

Limitations et contraintes

C

Diagramme de classes

Limitations et contraintes

C

- **Limitations :**

Diagramme de classes

Limitations et contraintes

C

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

Diagramme de classes

Limitations et contraintes

C

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

Diagramme de classes

Limitations et contraintes

C

- **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

C

- **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

C

- **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

C

- **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

C

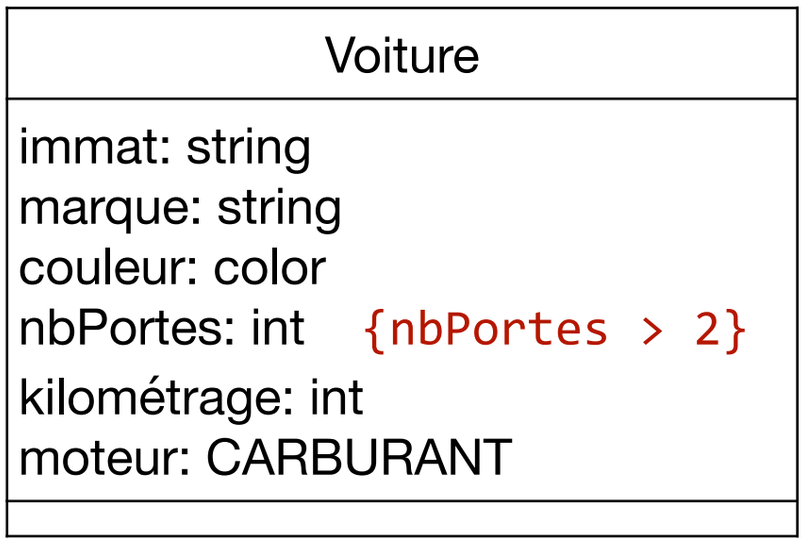
- **Limitations :**
  - Les contraintes sur les attributs
  - Les contraintes sur les associations
  - Les contraintes entre attributs et/ou entre associations

- **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

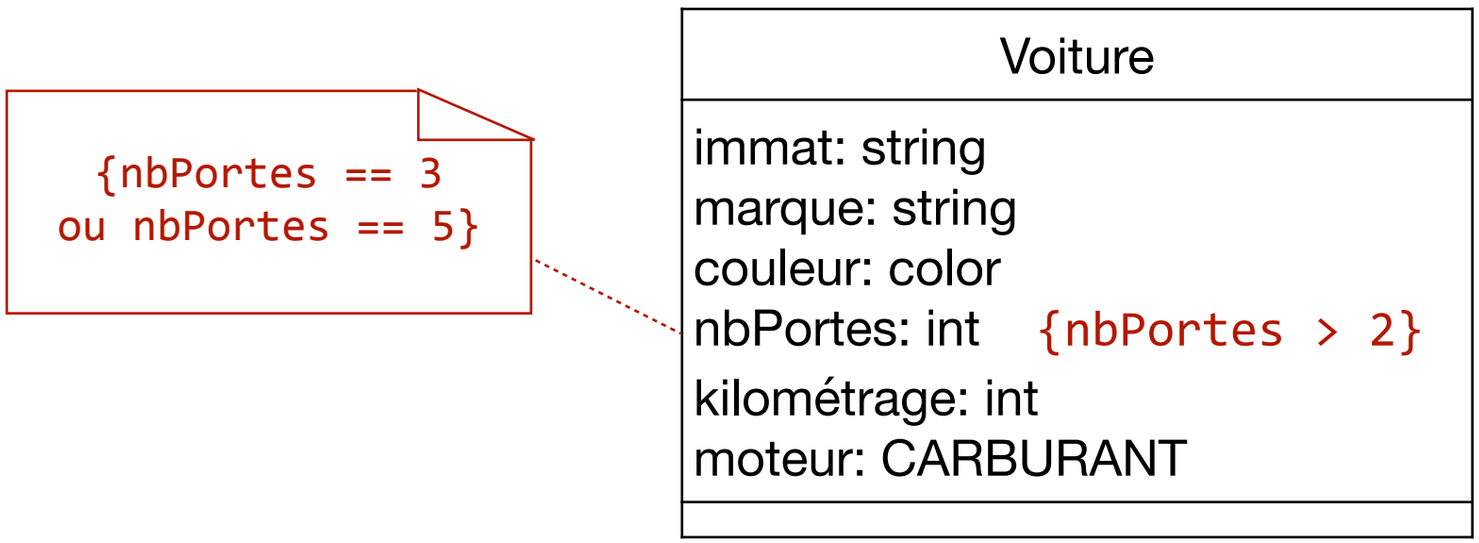
C



Contraintes

Contraintes sur les attributs

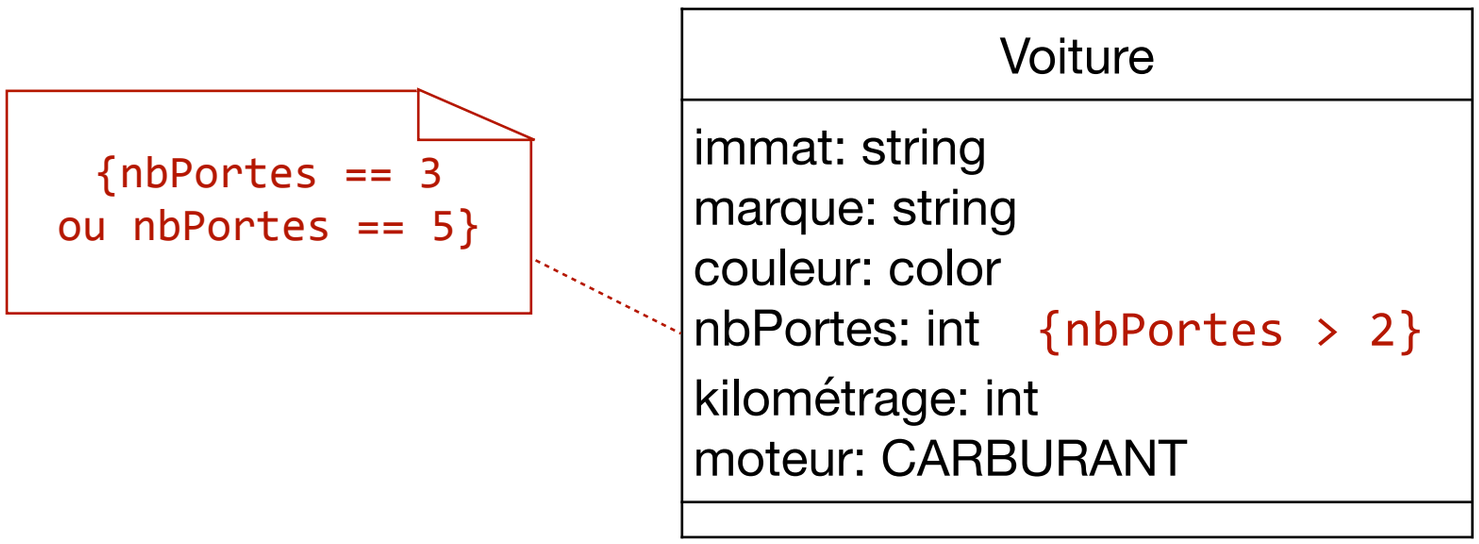
C



Contraintes

Contraintes sur les attributs

C



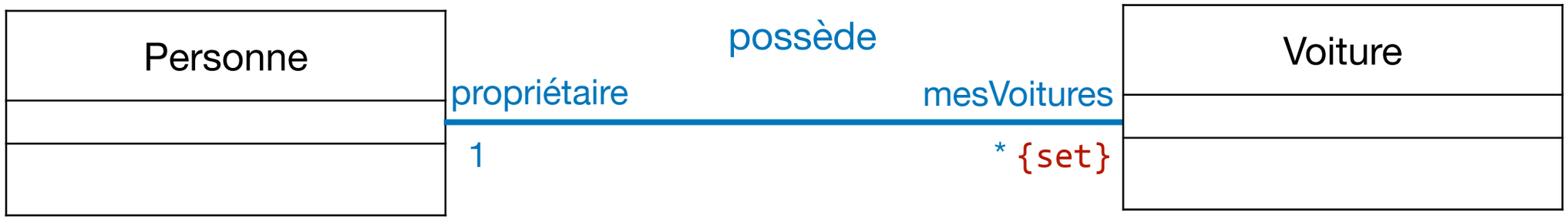
+

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

Contraintes

Contraintes sur les multiplicités

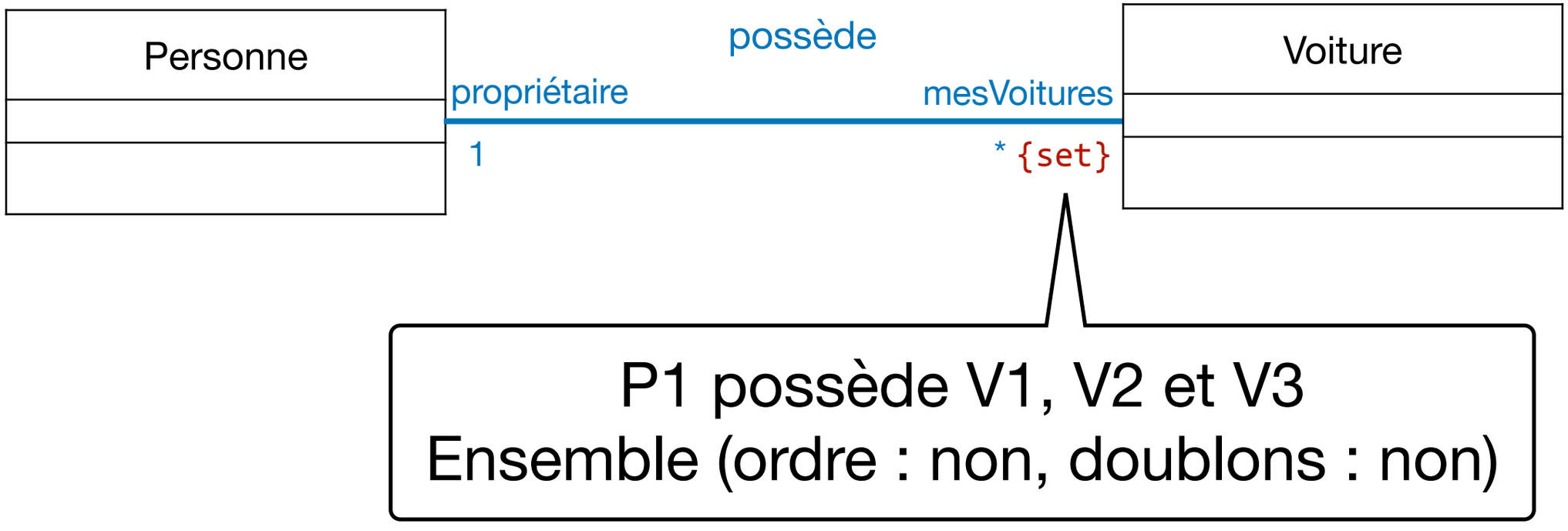
C



Contraintes

Contraintes sur les multiplicités

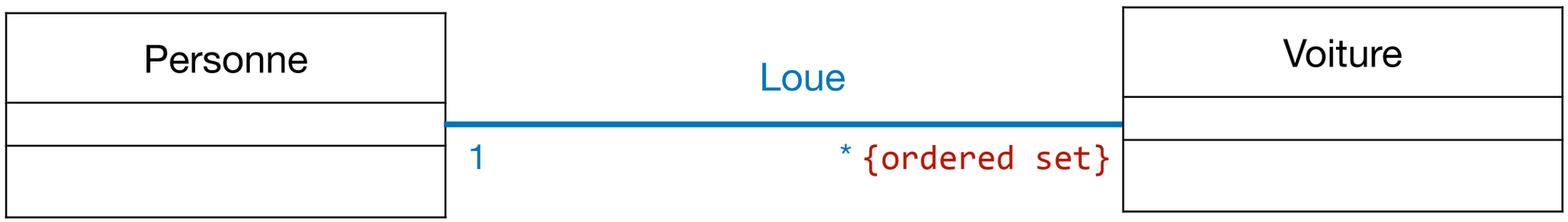
C



Contraintes

Contraintes sur les multiplicités

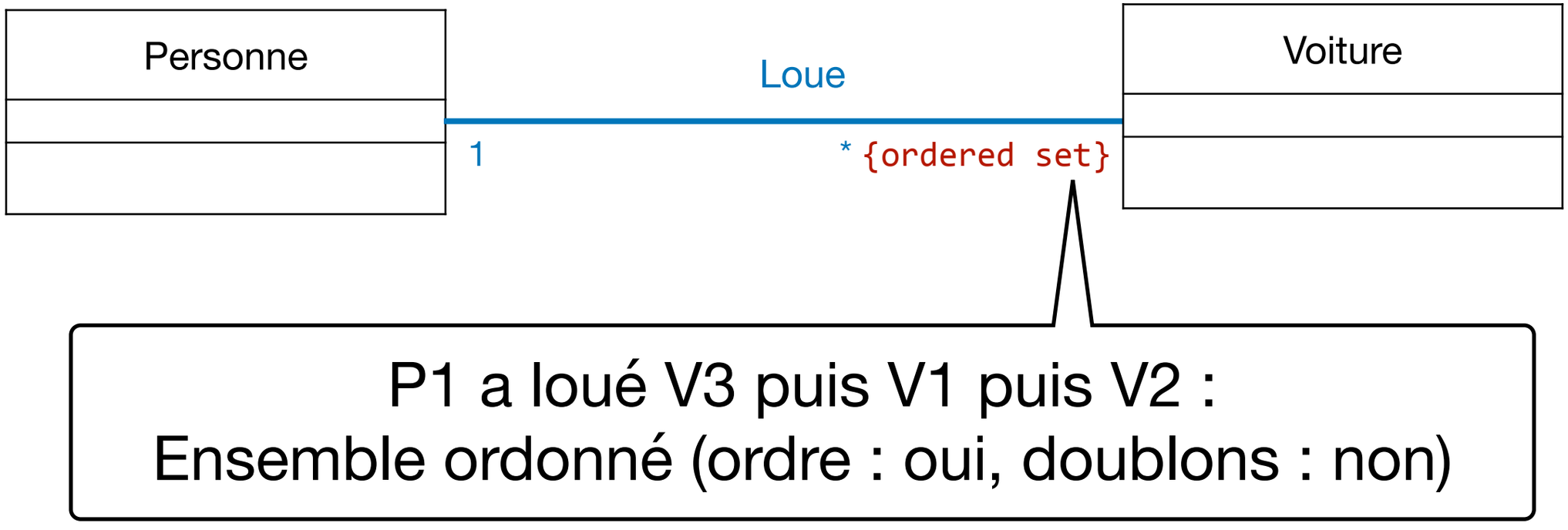
C



Contraintes

Contraintes sur les multiplicités

C



Contraintes

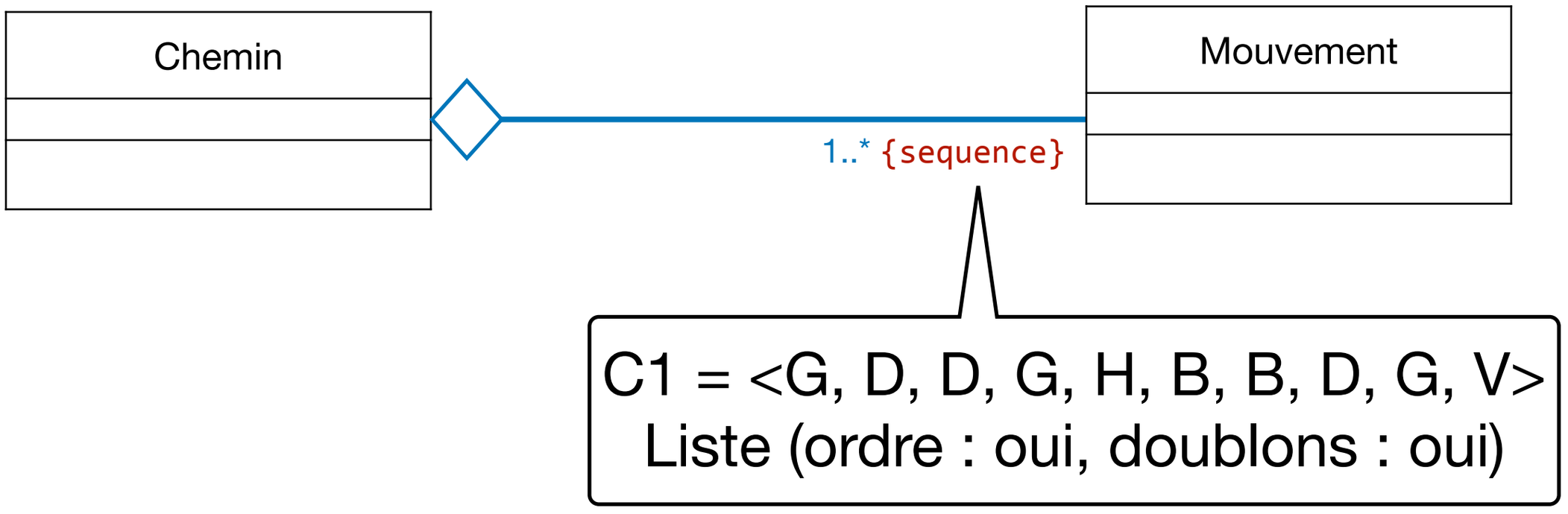
Contraintes sur les multiplicités

C

Contraintes

Contraintes sur les multiplicités

C



Contraintes

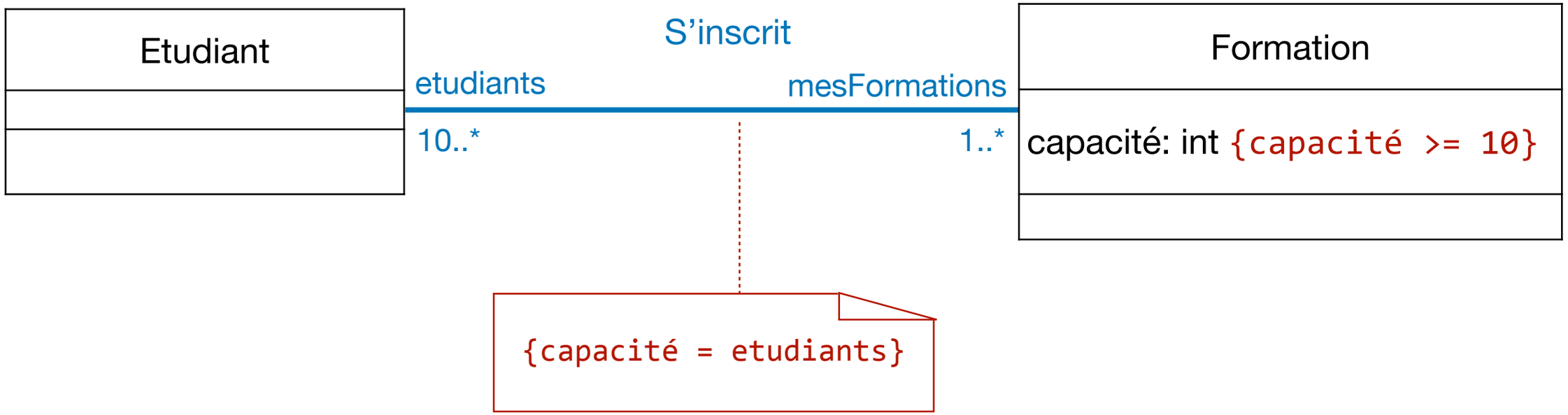
Contraintes sur les associations

C

Contraintes

Contraintes sur les associations

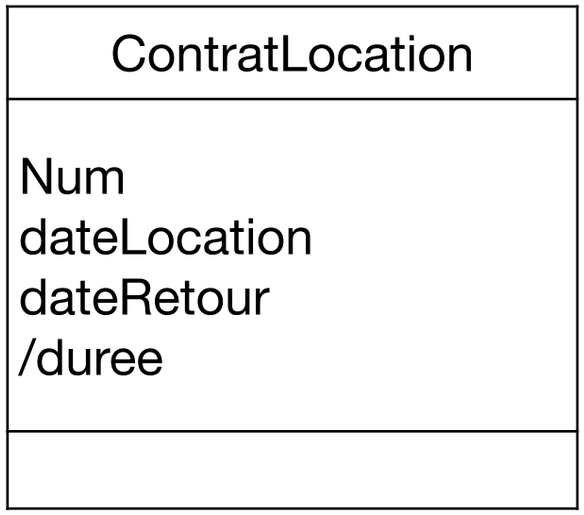
C



Contraintes

Contraintes entre attributs

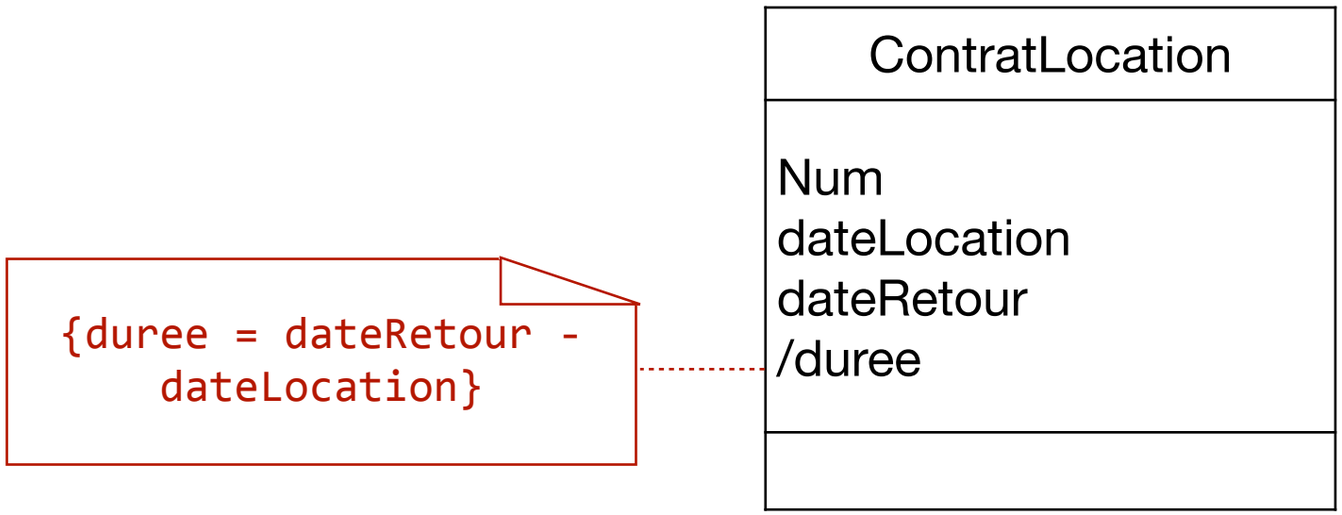
C



Contraintes

Contraintes entre attributs

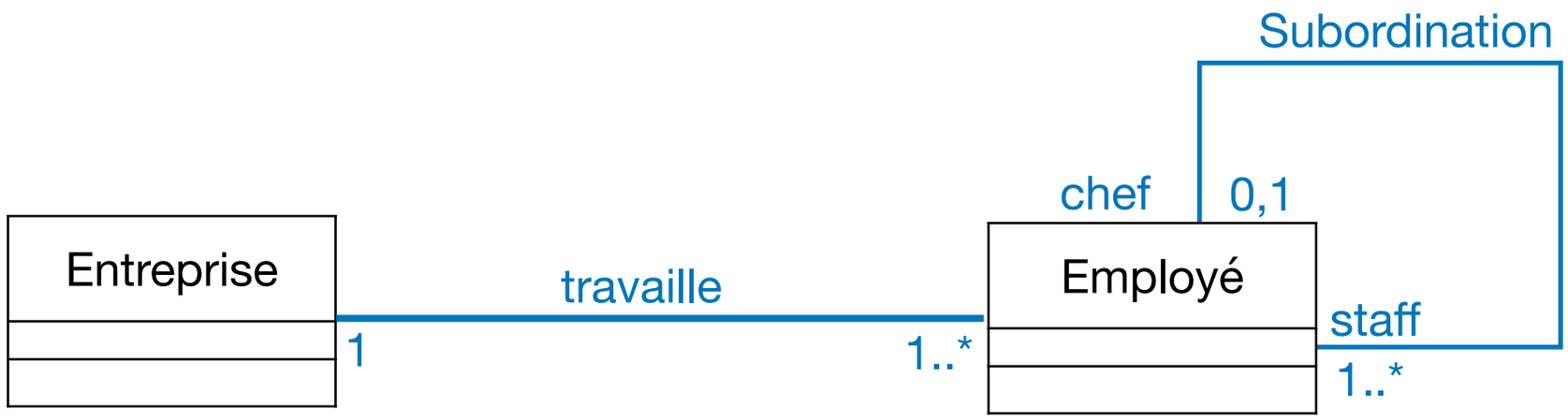
C



Contraintes

Contraintes entre associations

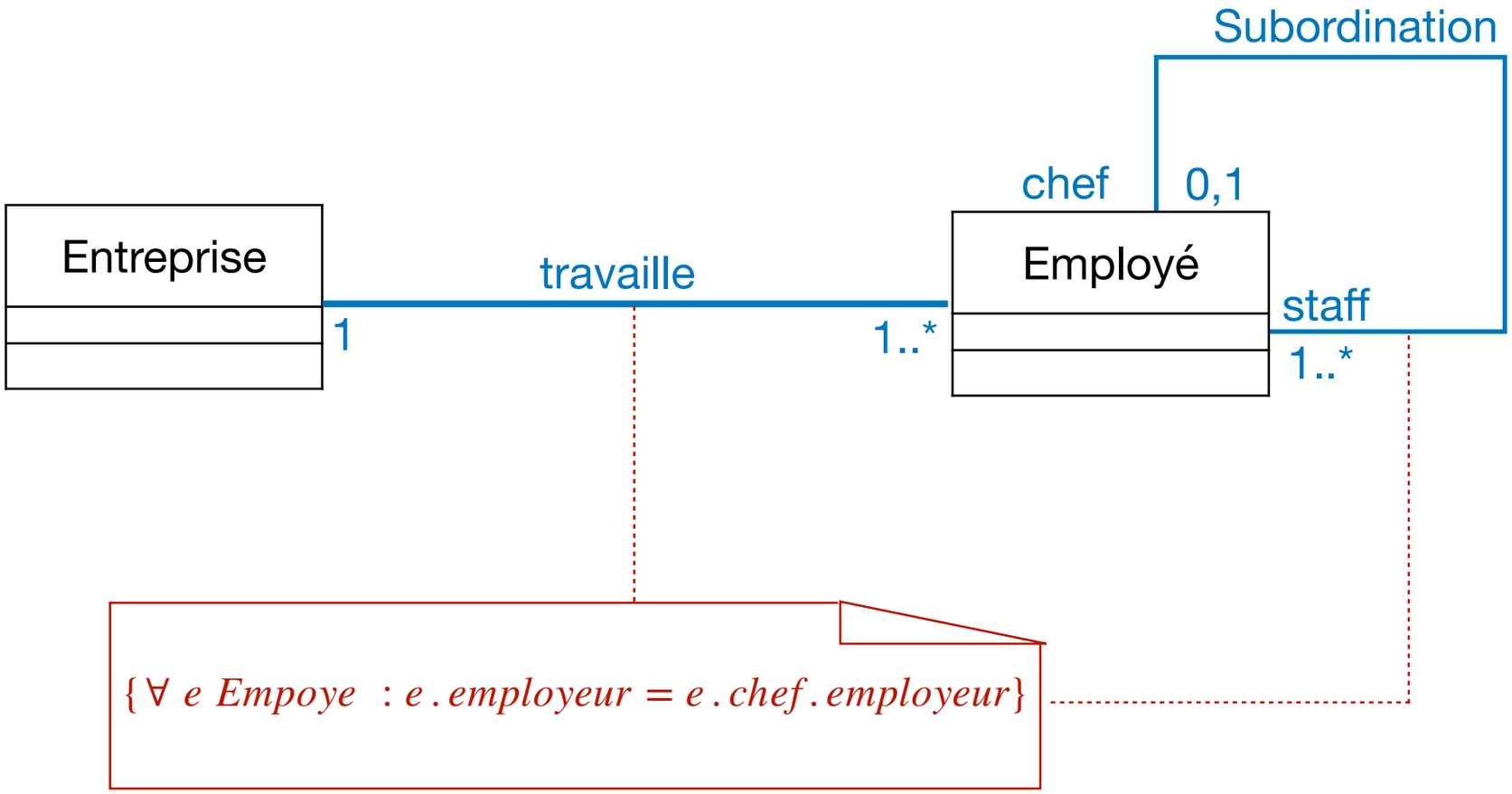
C



Contraintes

Contraintes entre associations

C



Contraintes

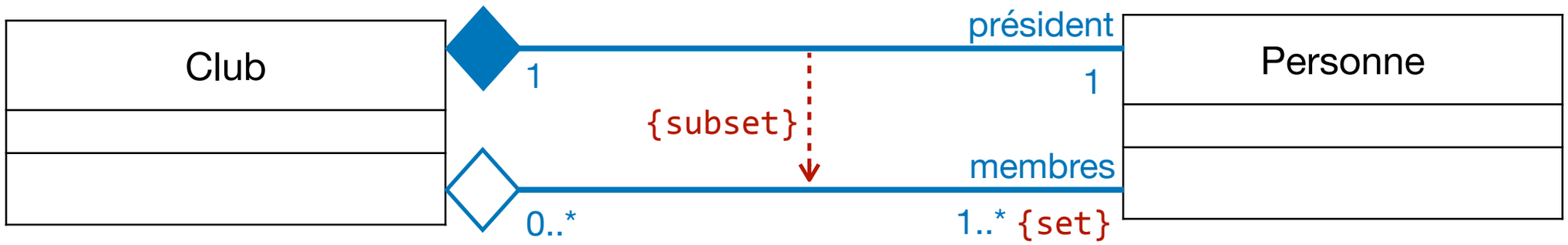
Contrainte {subset}

C

Contraintes

Contrainte {subset}

C



Contraintes

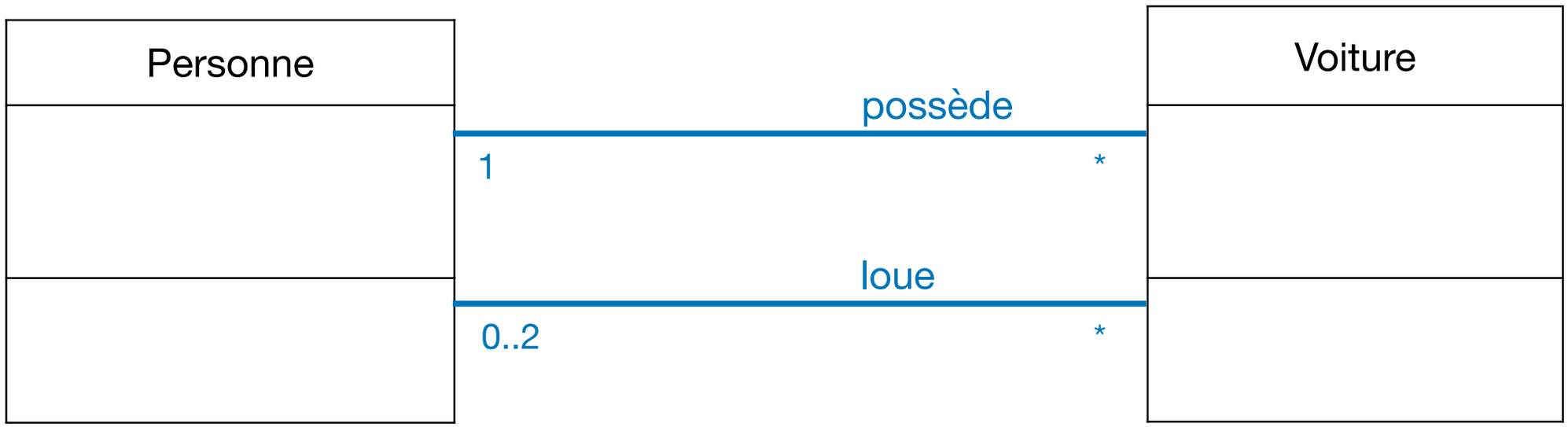
Contrainte {XOR}

C

Contraintes

Contrainte {XOR}

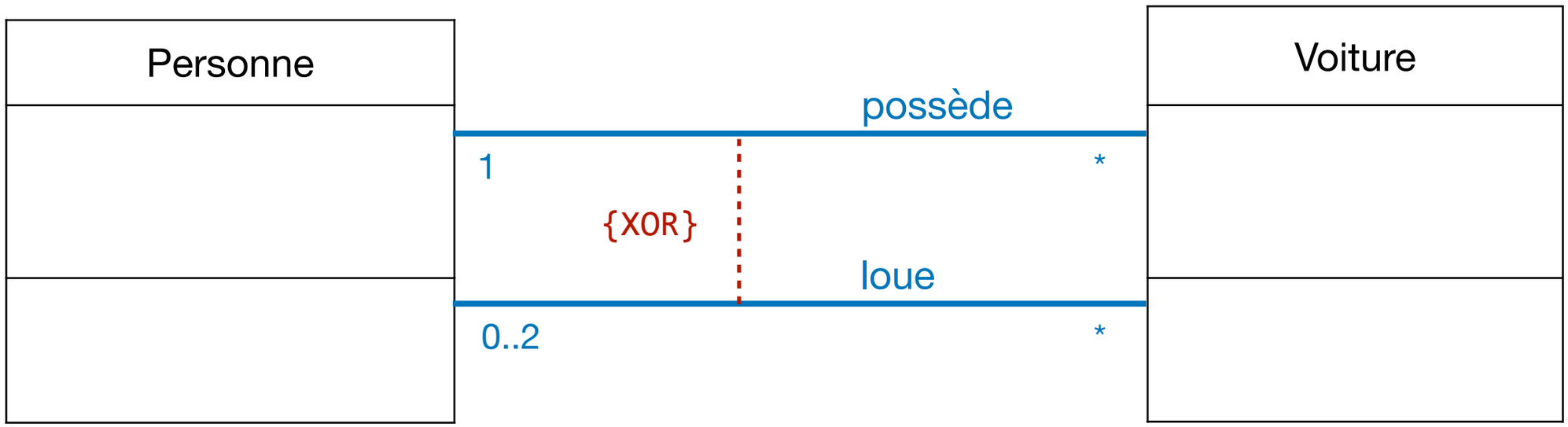
C



Contraintes

Contrainte {XOR}

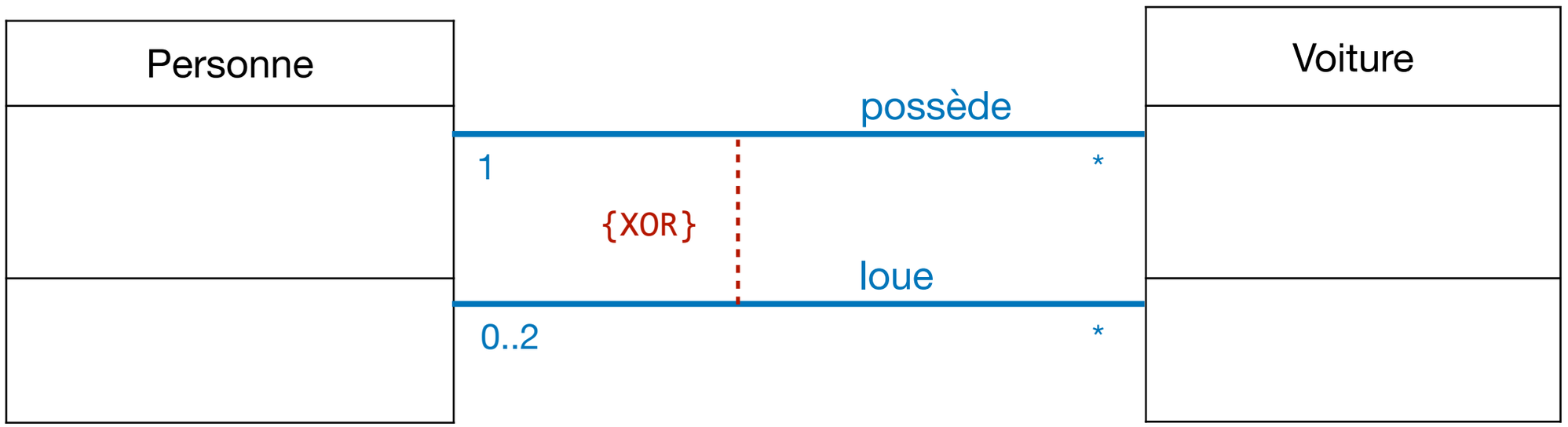
C

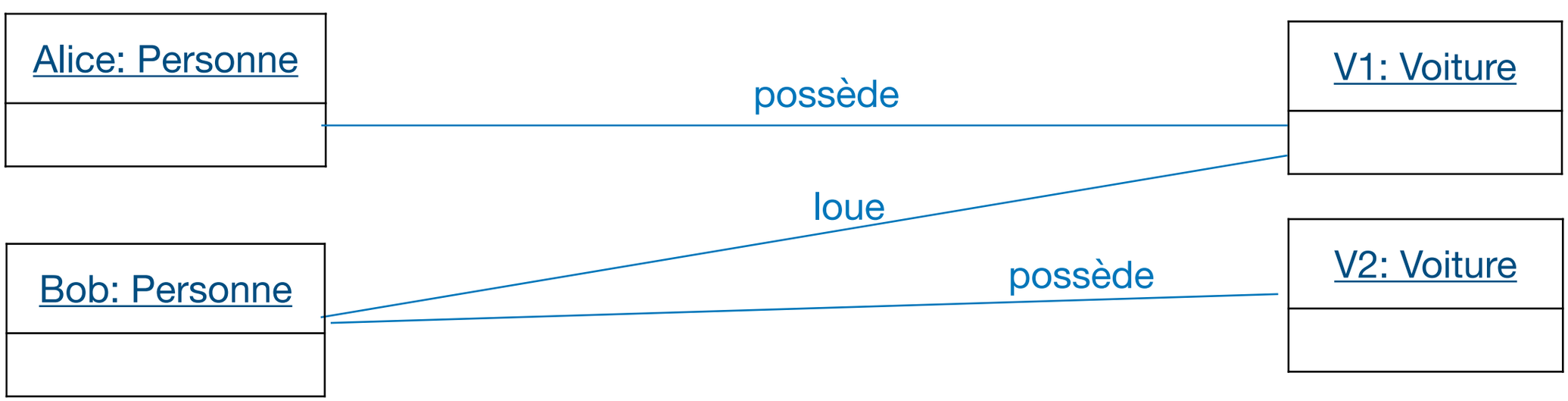


Contraintes

Contrainte {XOR}

C

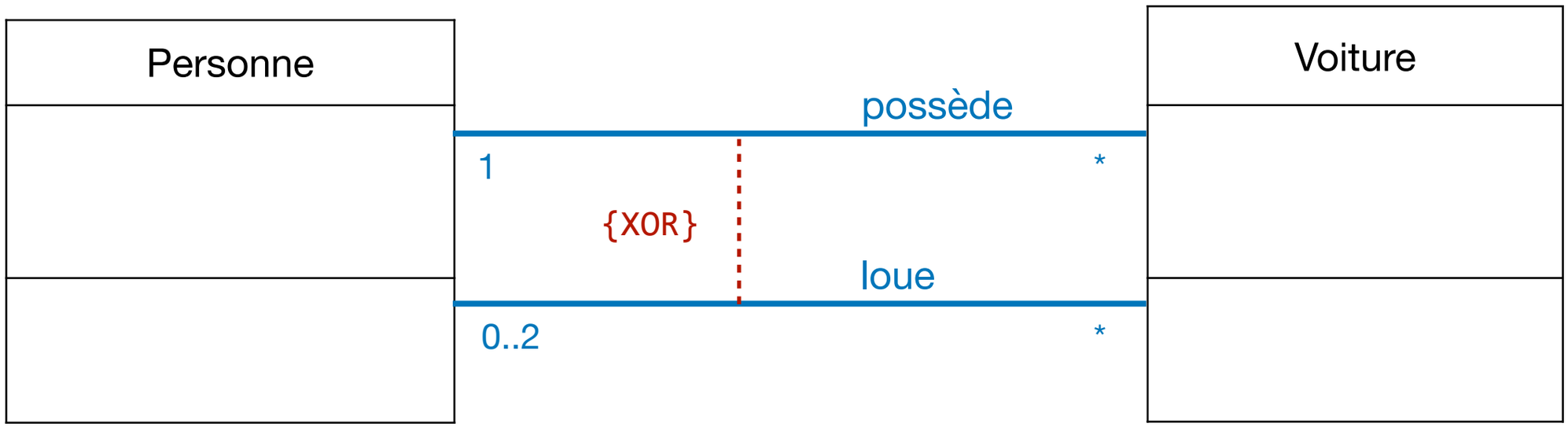


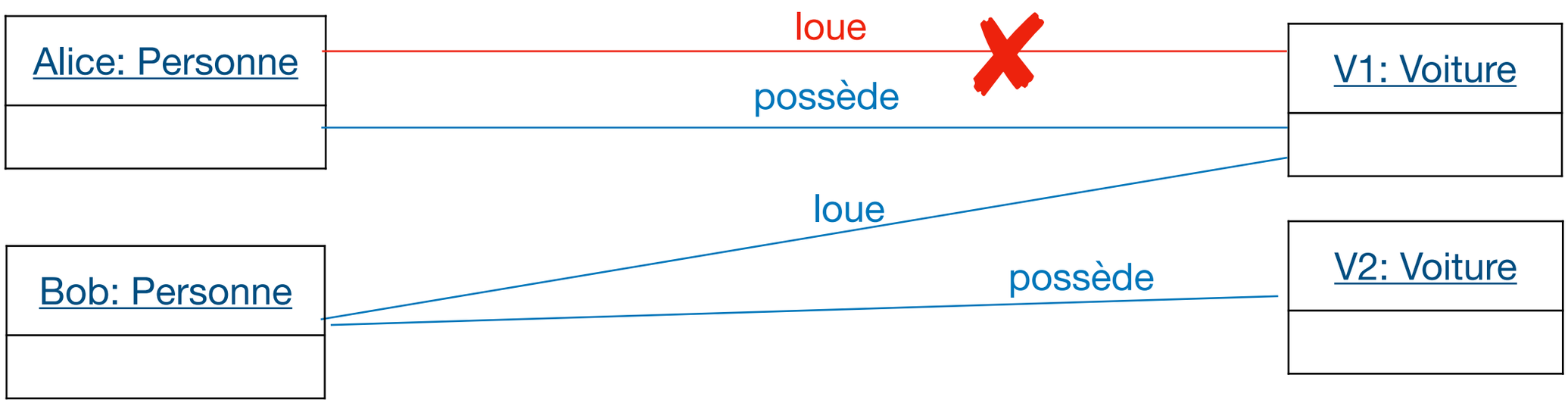


Contraintes

Contrainte {XOR}

C





Contraintes

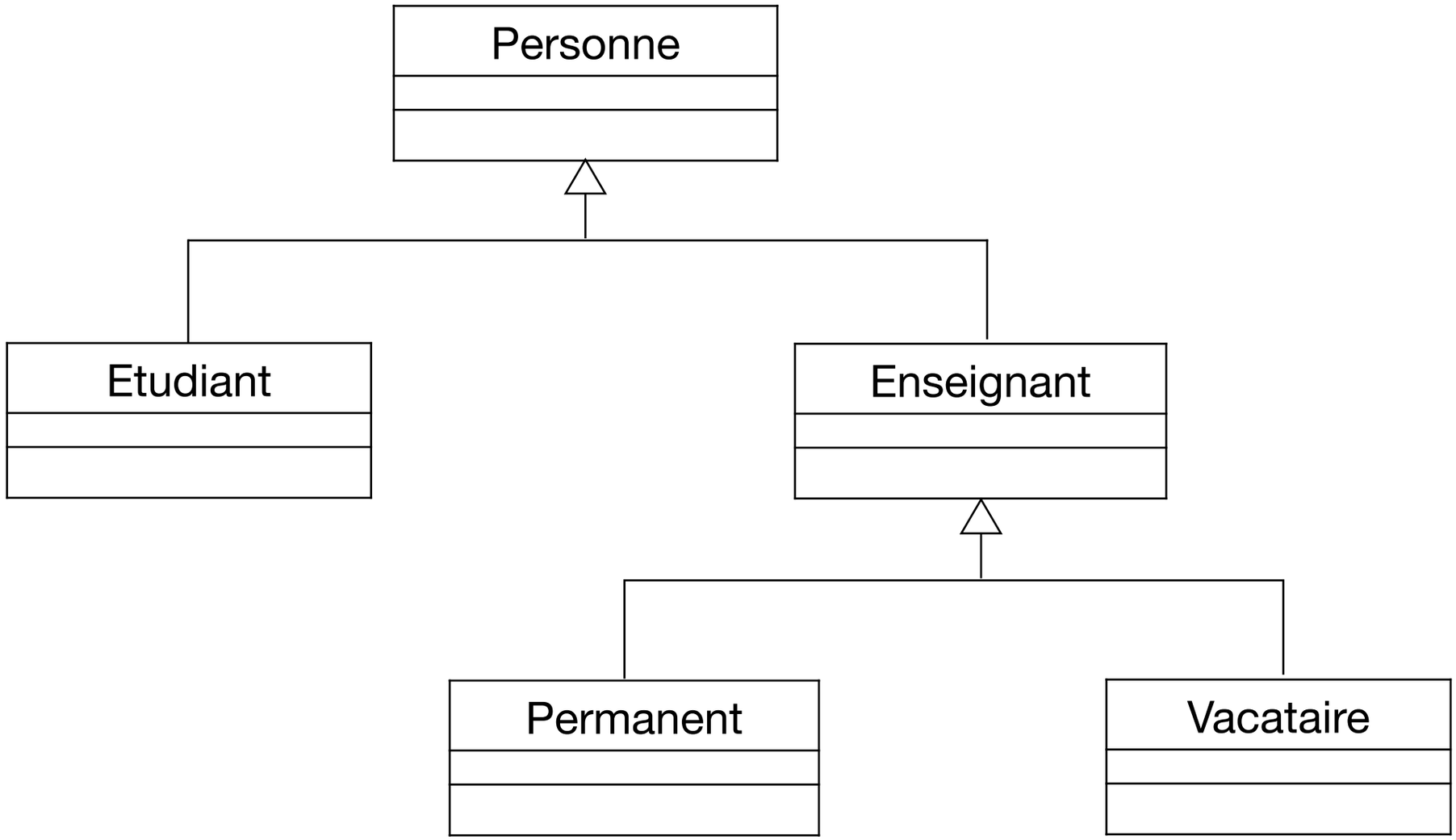
Contraintes sur la généralisation/spécialisation

C

Contraintes

Contraintes sur la généralisation/spécialisation

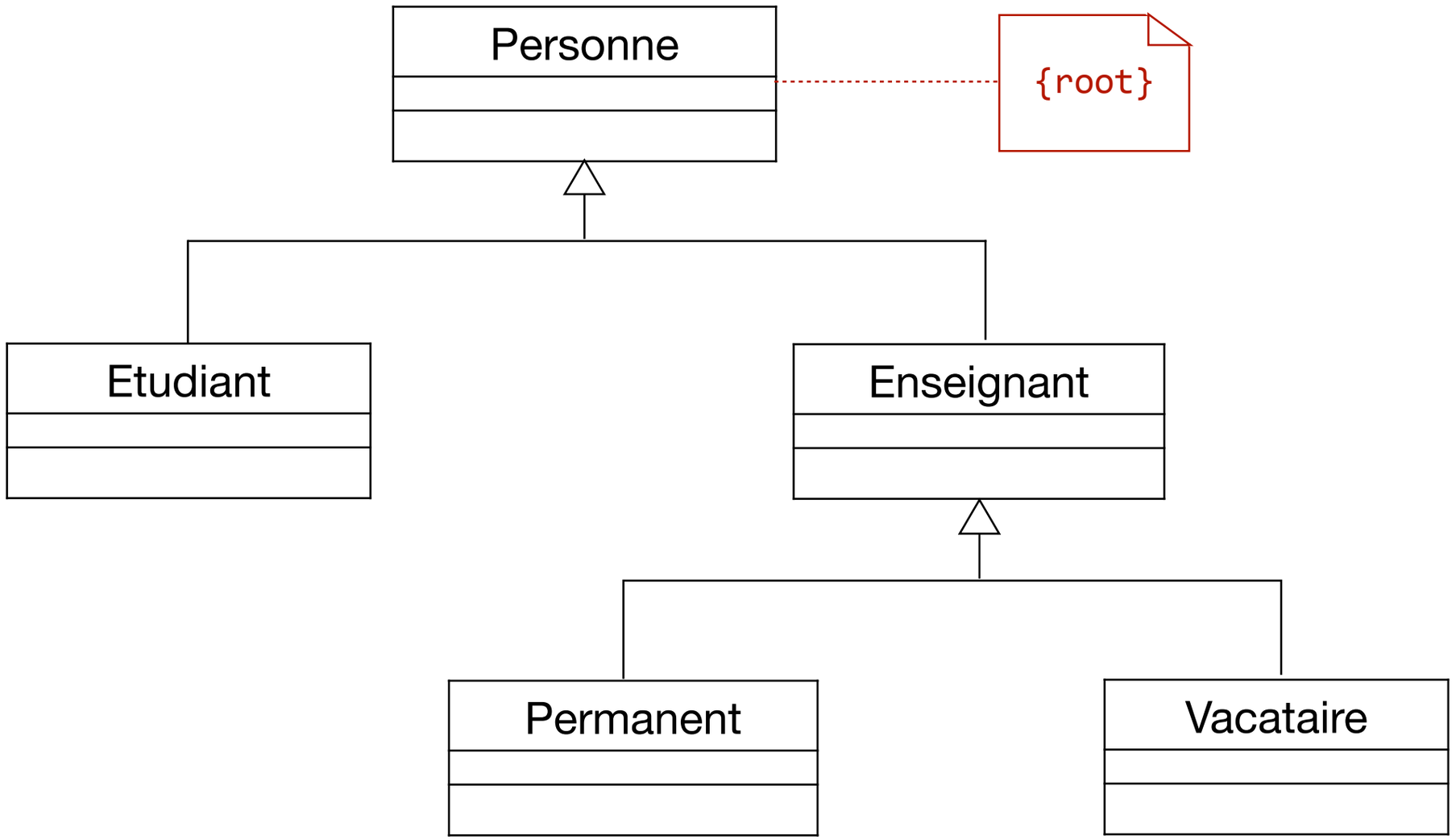
C



Contraintes

Contraintes sur la généralisation/spécialisation

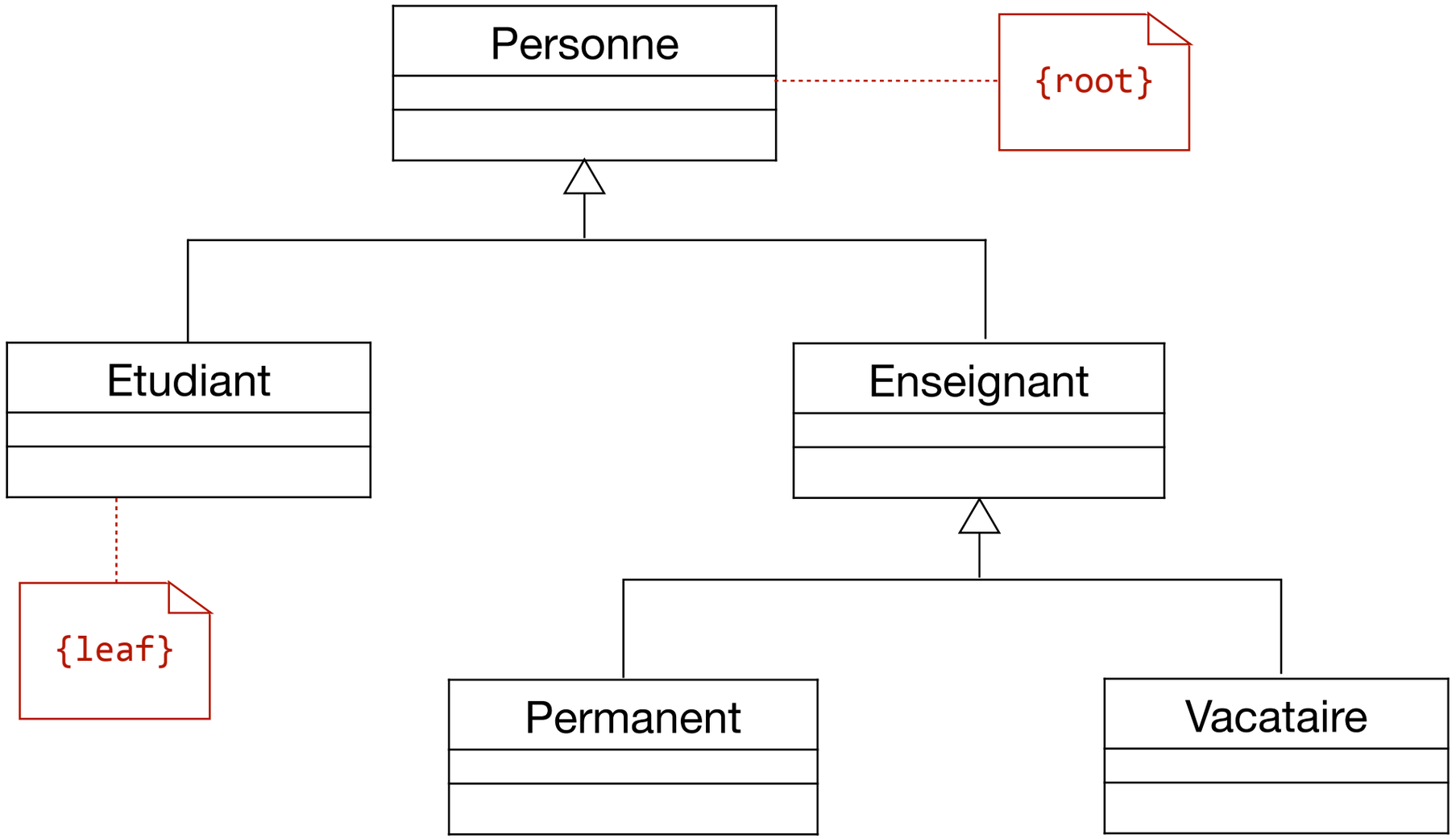
C



Contraintes

Contraintes sur la généralisation/spécialisation

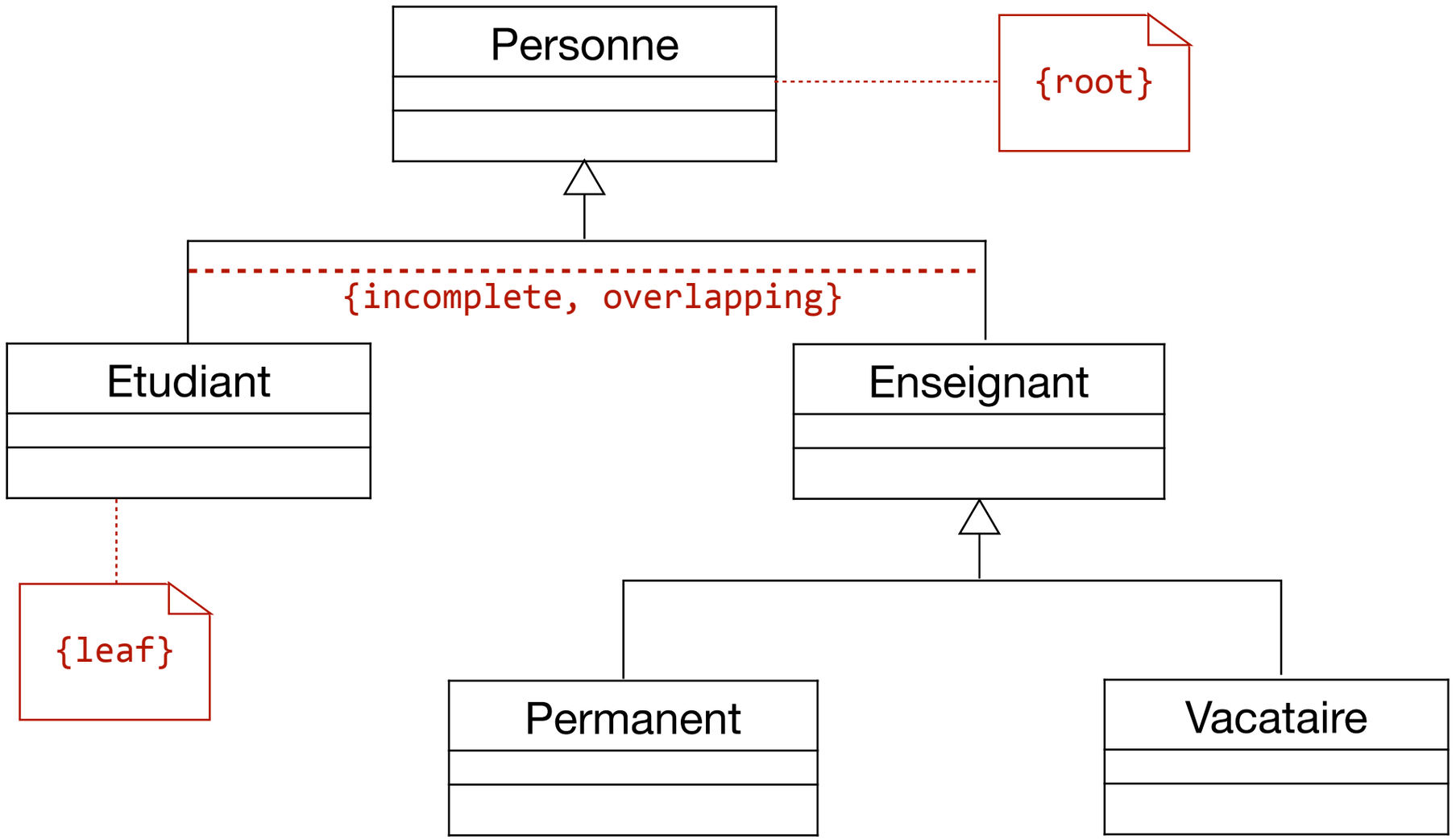
C



Contraintes

Contraintes sur la généralisation/spécialisation

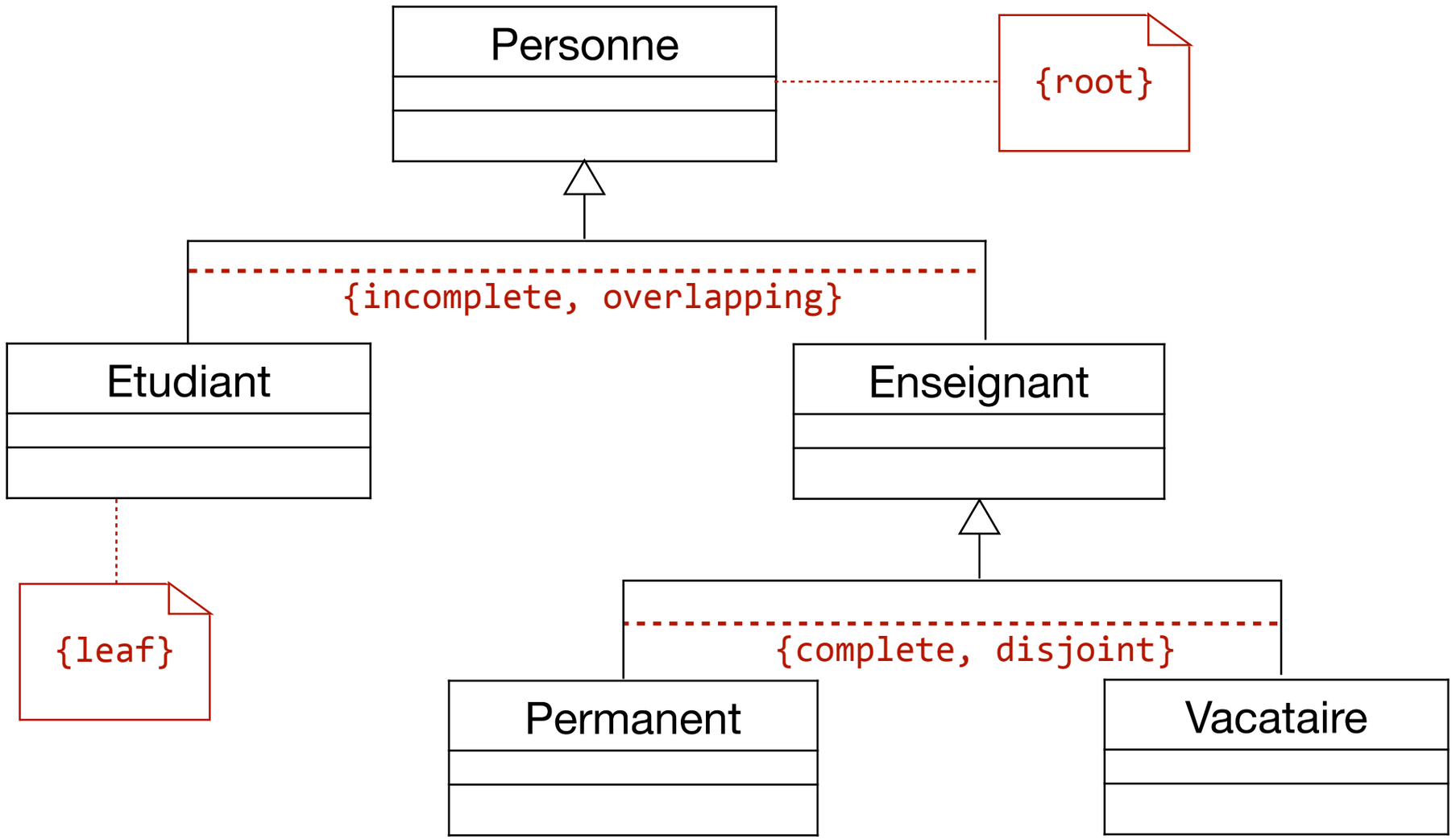
C



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

- Arnaud Gotlieb, SIMULA Research Lab., Oslo, Norway
- Christine Solnon, CITI, INSA Lyon
- Delphine Longuet, LRI, Paris-Sud (youtube channel)
- Keunhyuk Yeom, Pusan Univ
- Pierre Gérard, Paris 13