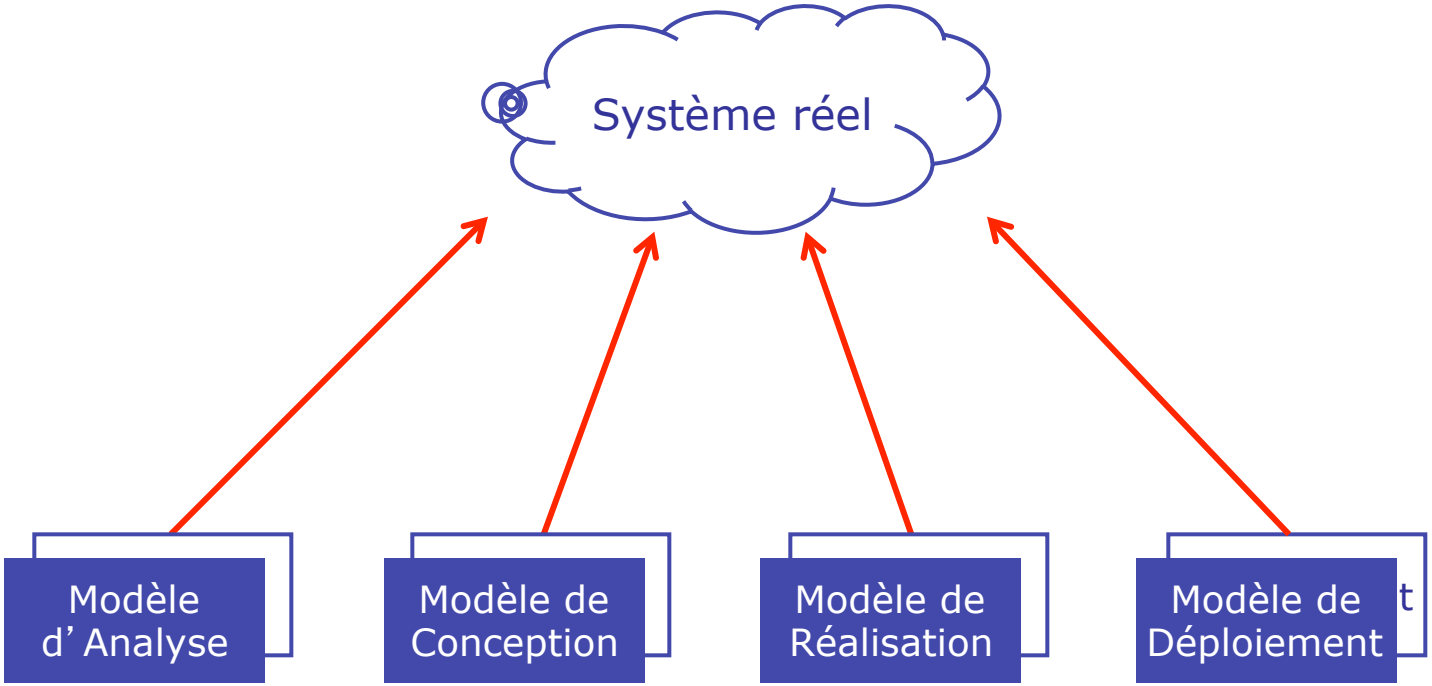
Modélisation UML

Plan

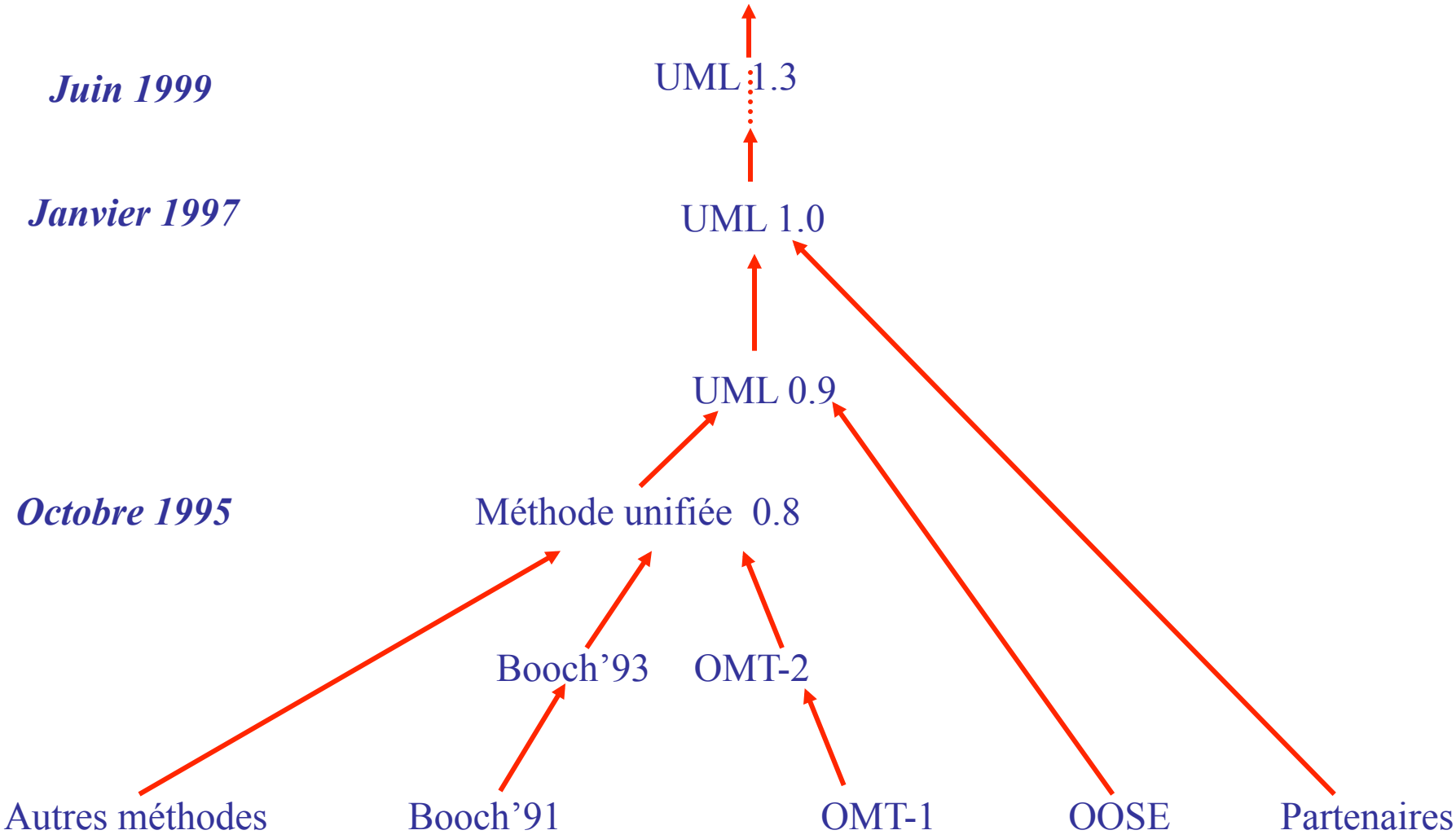
- Introduction
- Modélisation Objet
- Types de relation
  - Héritage
  - Association
  - Contenance
- Diagrammes UML
  - Diagramme d'objets
  - Cas d'utilisation
- Exercice

Introduction



UML (Unified Modeling Language)

Introduction



Introduction

Résumé

- UML est une notation, pas une méthode
- UML est un langage de **modélisation objet**
- UML convient pour toutes les méthodes objet
- UML est dans le domaine public

Programmation Orientée Objet

modéliser informatiquement des éléments d'une partie du monde réel en un ensemble d'entités informatiques (*objets*)

Intérêt d'une méthode objet

- définir le problème à haut niveau sans rentrer dans les spécificités du langage
- définir un problème de façon graphique
- utiliser les services offertes par l'objet sans rentrer dans le détail de programmation (**Encapsulation**)
- Réutilisation du code

Modélisation objet

Notion d'*Objet*

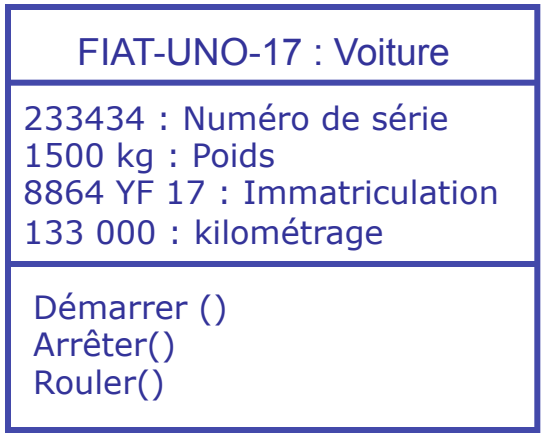
Une abstraction du monde réel c.-à-d. des données informatiques regroupant des caractéristiques du monde réel

Exemple

une personne, une voiture, une maison, ...

Caractérisation d'un objet

- **Identité**
  permet de le distinguer des autres objets
- **Attributs**
  données caractérisant l'objet
- **Méthodes**
  actions que l'objet est à même de réaliser



Modélisation objet

Notion de *Classe*

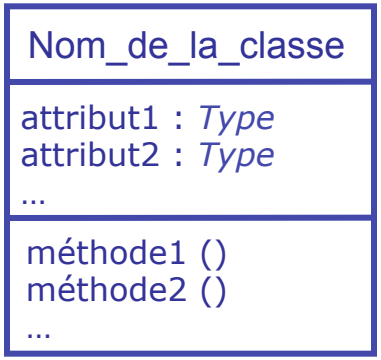
- Structure d'un objet, c.-à-d. une déclaration de l'ensemble des entités qui composeront l'objet
- Un objet est donc "issu" d'une classe, c'est le produit qui sort d'un moule

Notation

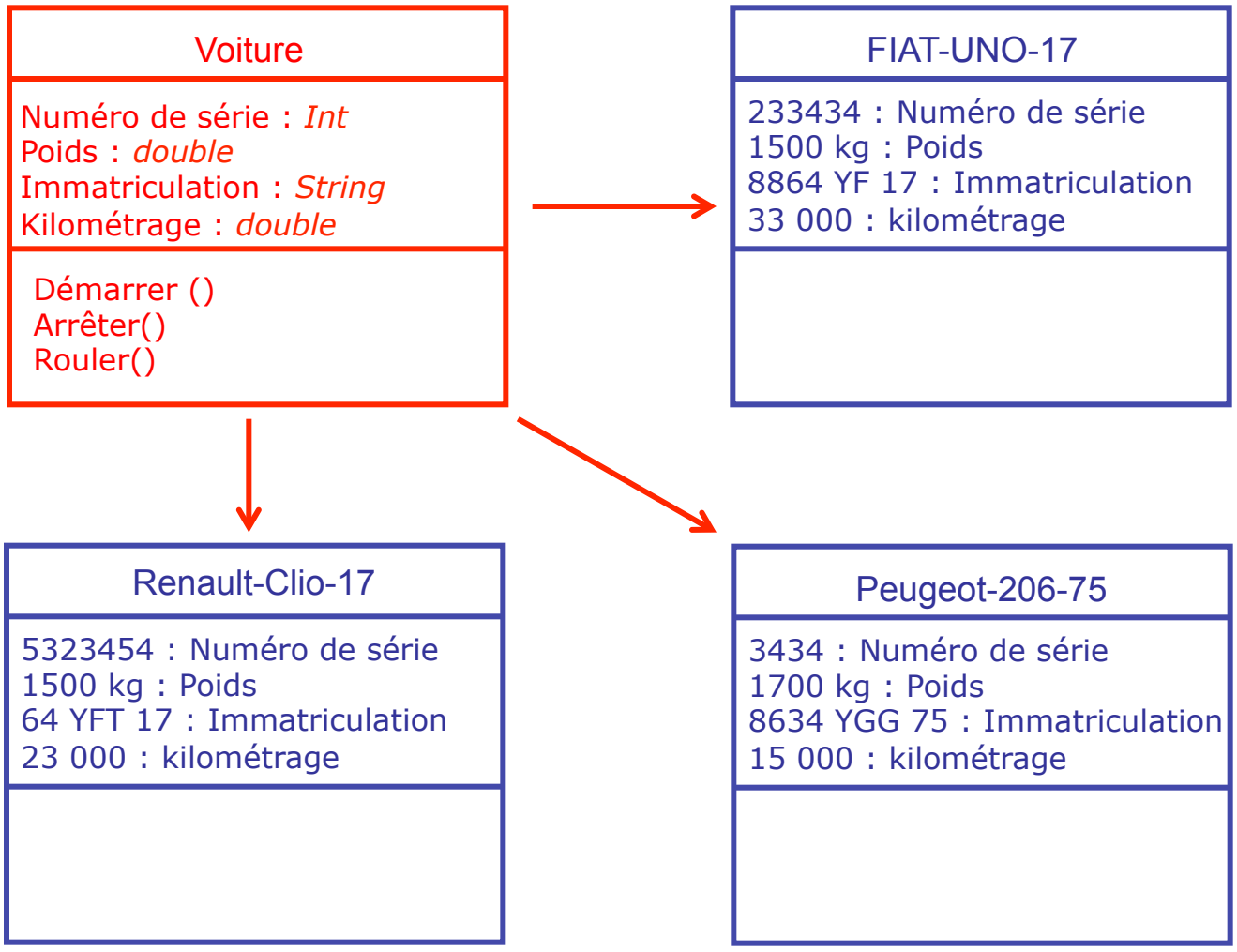
un objet est une **instanciation** (***occurrence***) d'une classe

Une classe est composée:

- **attributs**
  données dont les valeurs représentent l'état de l'objet
- **méthodes**
  opérations applicables aux objets



Modélisation objet



Modélisation objet

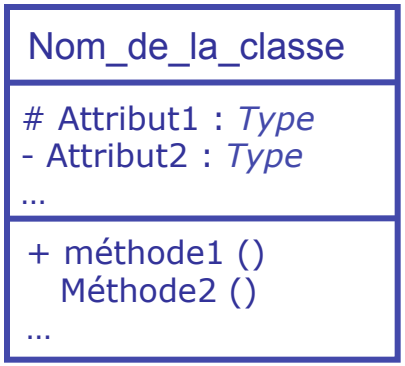
Visibilité des attributs

définissent les droits d'accès aux données (pour la classe elle-même, d'une classe héritière, ou bien d'une classe quelconque)

- **Publique (+)**
  les classes peuvent accéder aux données et méthodes d'une classe définie avec le niveau de visibilité *public*

- **Protégée (#)**: l'accès aux données est réservé aux fonctions des classes héritières

- **Privée (-)**: l'accès aux données est limité aux méthodes de la classe elle-même



Types de relation entre classes

Héritage

Association

Contenance

Types de relation : Héritage

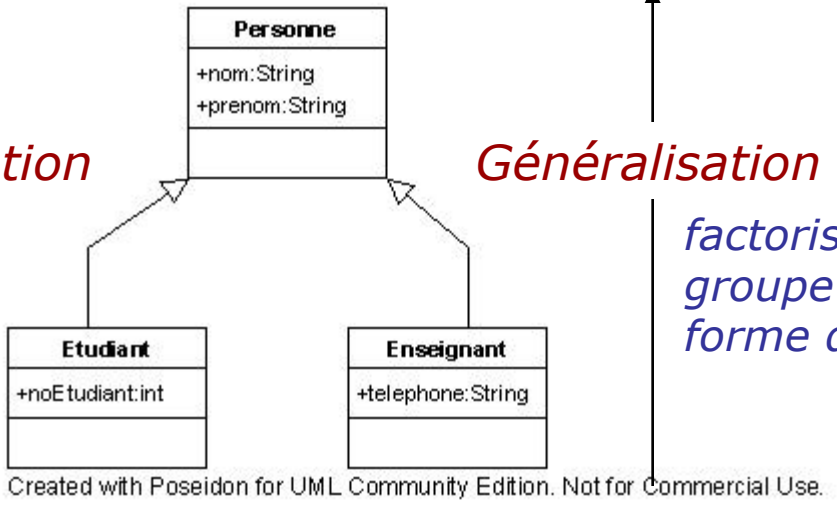
permet de créer une nouvelle classe à partir d'une classe existante

Principe

classe dérivée contient les attributs et les méthodes de sa superclasse

Spécialisation

étendre les propriétés d'une classe, sous forme de sous-classes



Généralisation

factoriser les propriétés groupe de classes sous forme de super-classe

Chaque personne de l'université est identifiée par son nom, prénom
Les étudiants ont plus un noEtudiant
Les enseignants ont un numéro de téléphone interne

Types de relation : Association

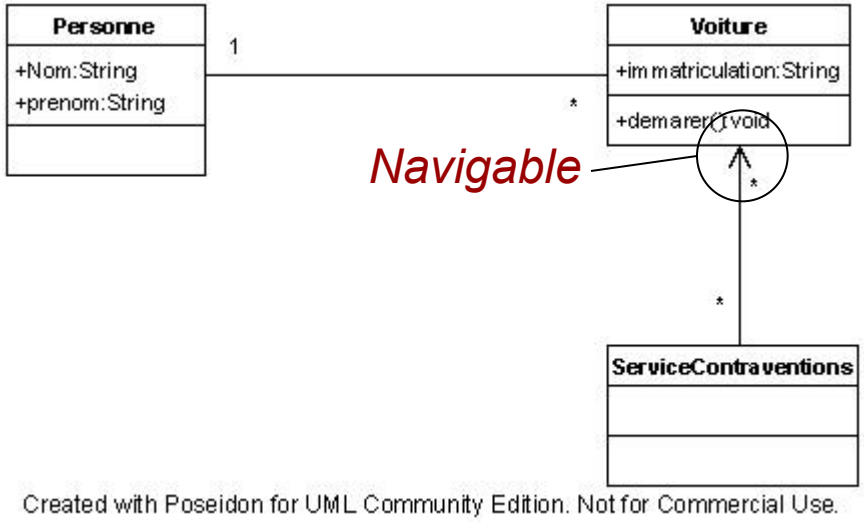
Connexion sémantique entre deux classes

Navigabilité

- Par défaut une association est navigable dans les deux sens

  - Chaque instance de voiture a un lien vers le propriétaire
  - Chaque instance de Personne a un ensemble de lien vers les voitures

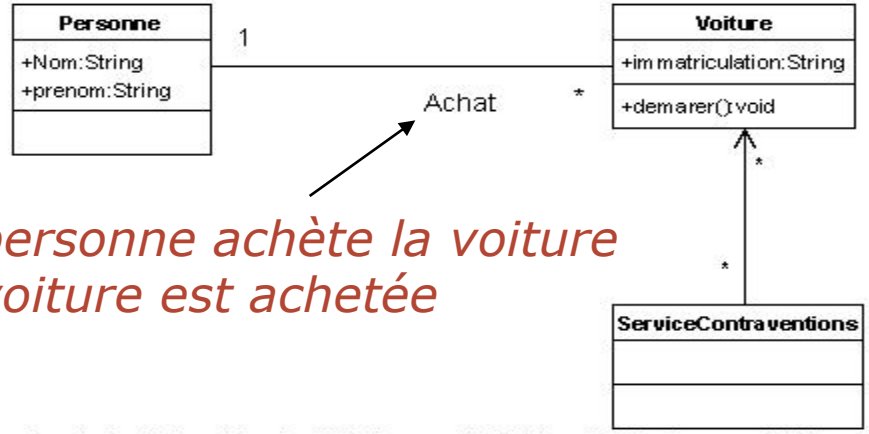
- Restriction de la navigabilité
  - Le service de contravention est associé à une ou plusieurs voiture(s)
  - La voiture ne connaît pas service de contravention



Types de relation : Association

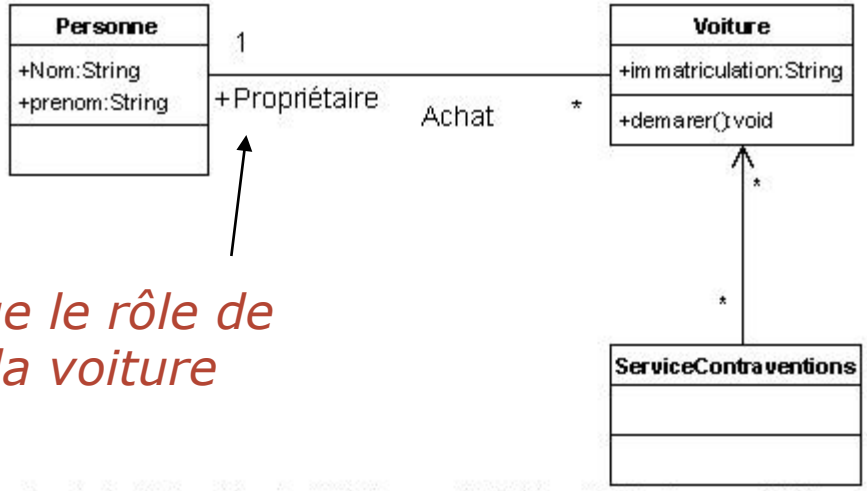
Documentation d'une association

- **Nom de l'association**
  lien sémantique entre les classes



La personne achète la voiture
La voiture est achetée

- **Rôle d'une association**
  Spécification du rôle de la classe

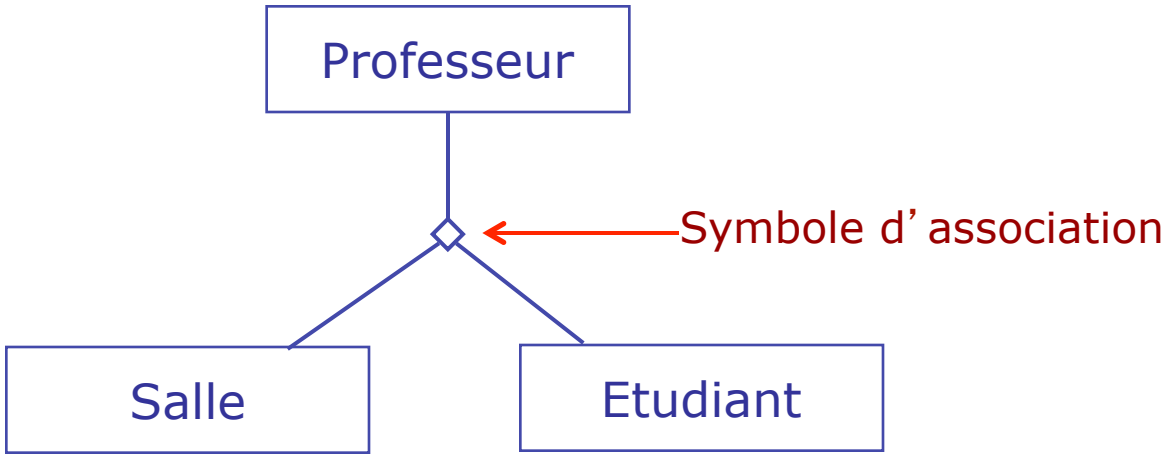


La personne joue le rôle de propriétaire de la voiture

Types de relation : Association

Relation n-aire

Type particulier d'association qui relie plus de deux classes



Attention

difficiles à déchiffrer

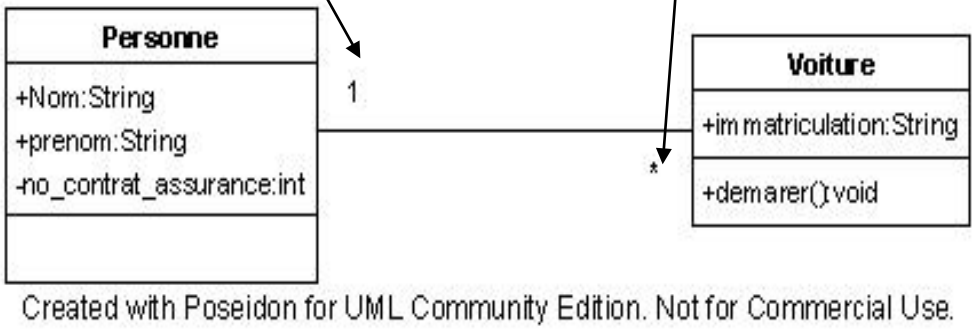
Types de relation : Association

Multiplicités

1 : la classe est en relation avec un et un seul objet de l'autre classe
1..* : la classe est en relation avec au moins un objet de l'autre classe
0..* : la classe est en relation avec 0 ou n objets de l'autre classe
0..1 : la classe est en relation avec au plus un objet de l'autre classe

Une voiture est achetée par une et une seule personne

Une personne peut acheter 0 ou n voitures



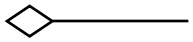
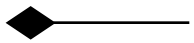
Types de relation : Contenance

Cas particulier d'association exprimant une relation de contenance

Exemples:

- Une voiture a 4 roues
- Un dessin contient un ensemble de figures géométriques
- Une présentation PowerPoint est composé de transparents
- Une équipe de recherche est composée d'un ensemble de personnes

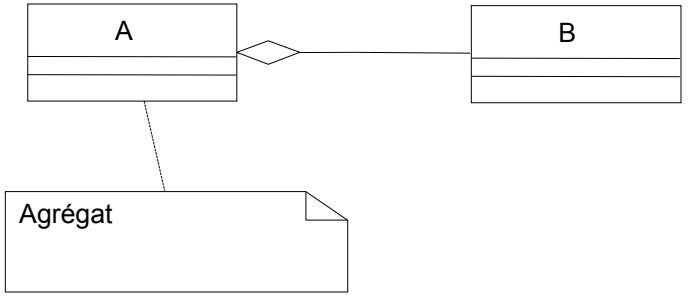
Deux types de relations de contenance en UML

- Agrégation 
- Composition (Agrégation forte) 

Types de relation : Agrégation

Type de relations

– A « contient » des instances de B,

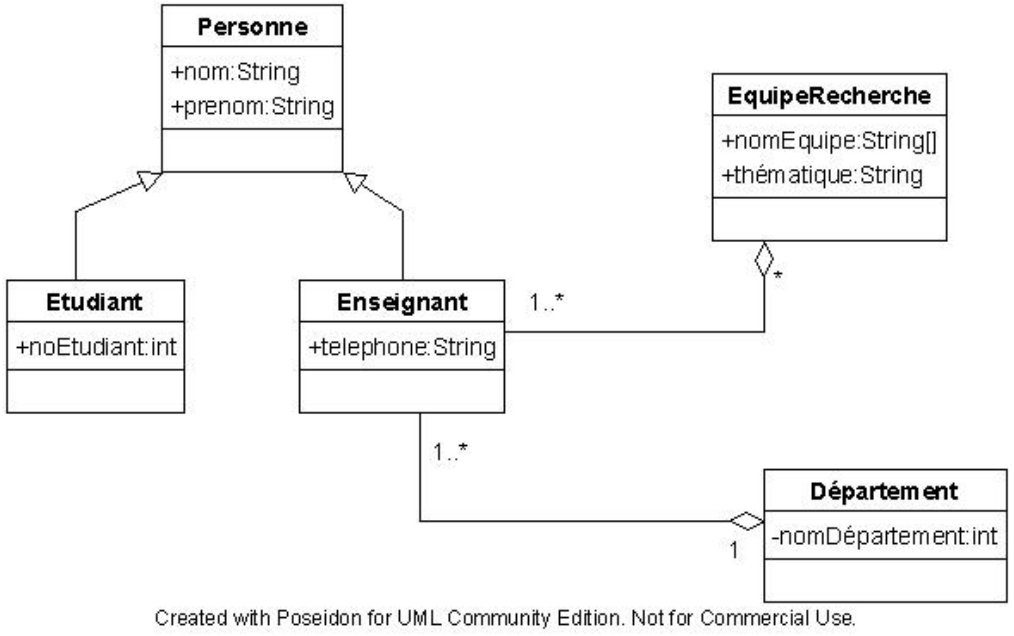


Propriétés de l'agrégation

- La suppression de A n'implique pas la suppression de B
- L'élément agrégé peut être partagé

Exemples :

- L'enseignant est un composant d'une (ou plusieurs) équipe de recherche d'un seul département
- La disparition d'une équipe de recherche n'entraine pas la disparition d'un enseignant



Types de relation : Composition

- La suppression de A entraine la suppression de B

***Exemple**:*

« Une présentation PowerPoint est composé de transparents »

La suppression de la présentation entraine la disparition des transparents qui la compose

Diagramme de classes

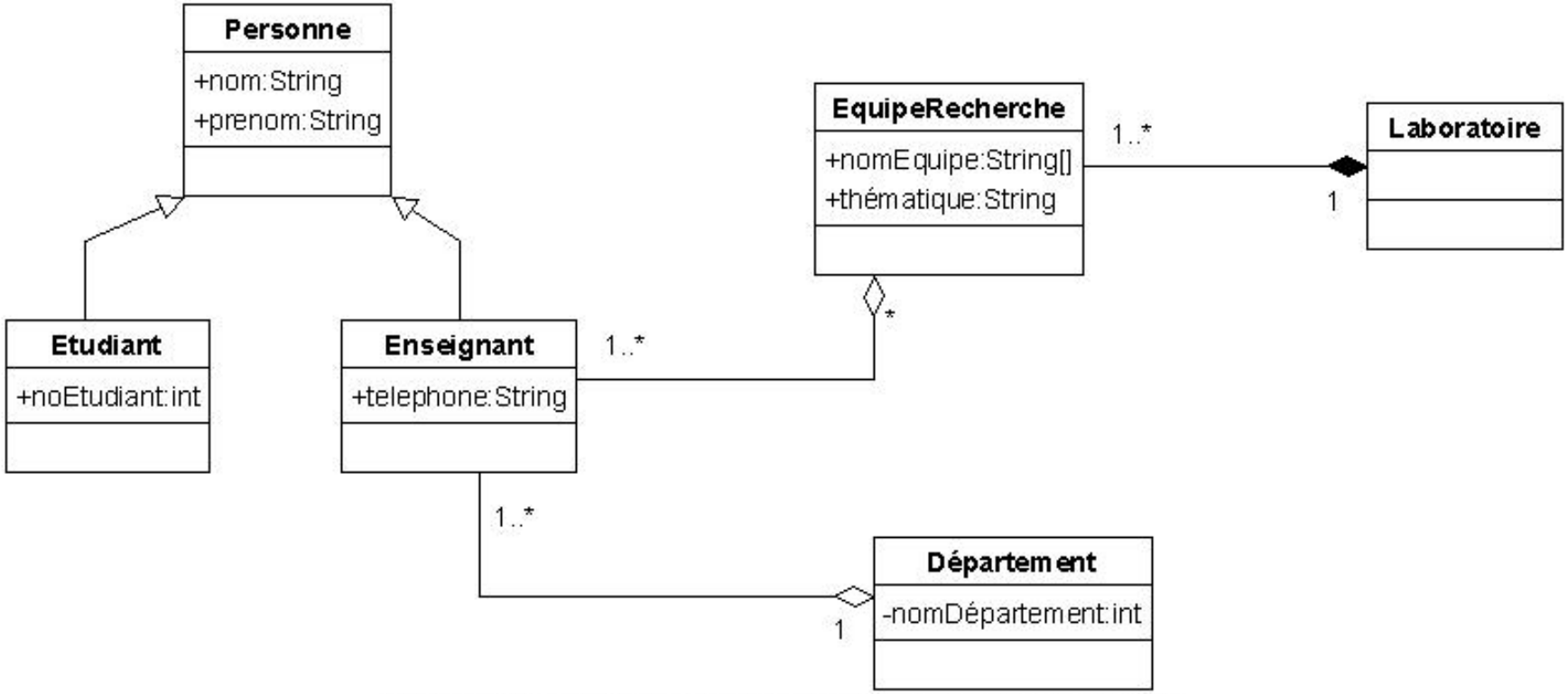
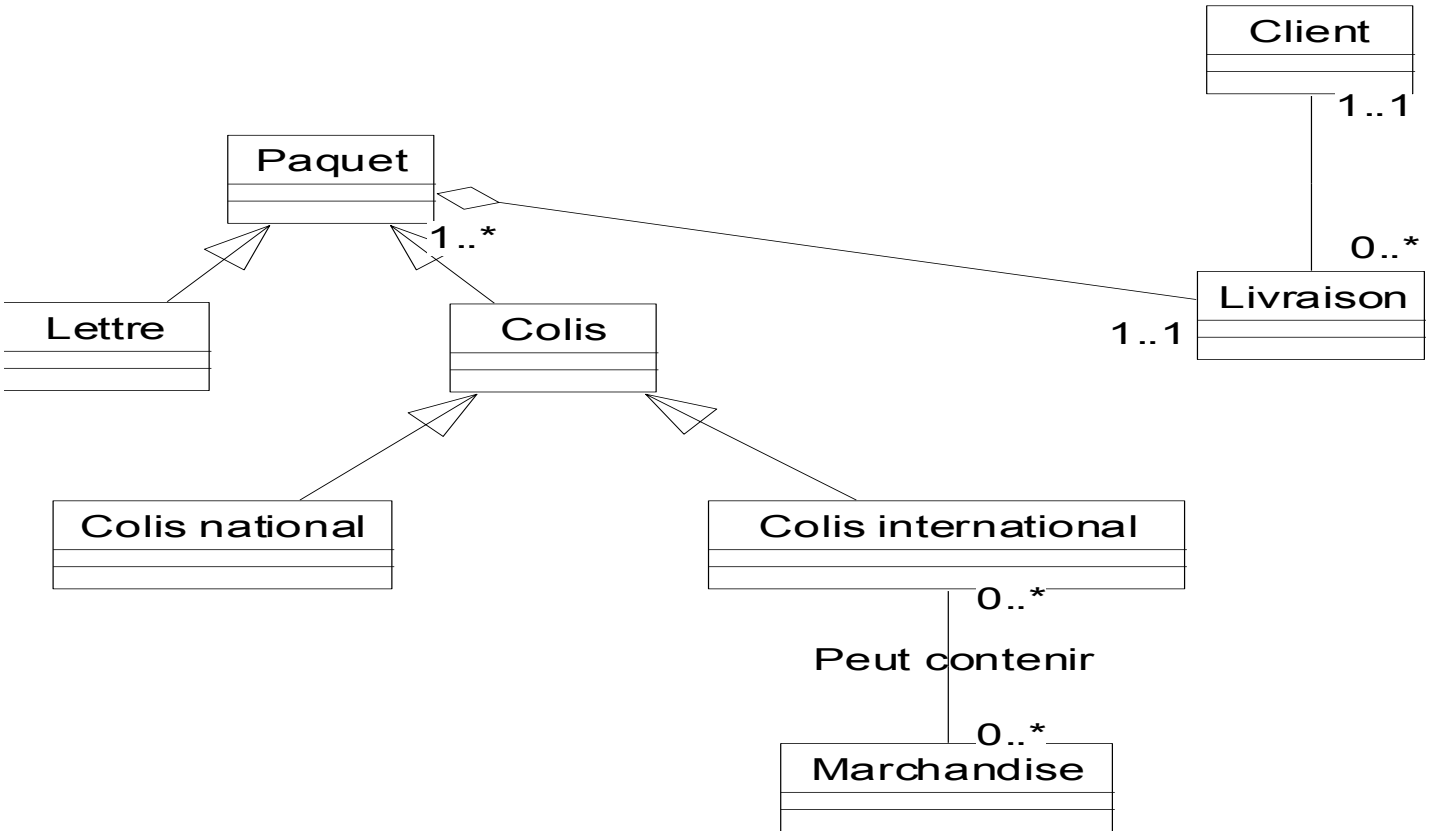


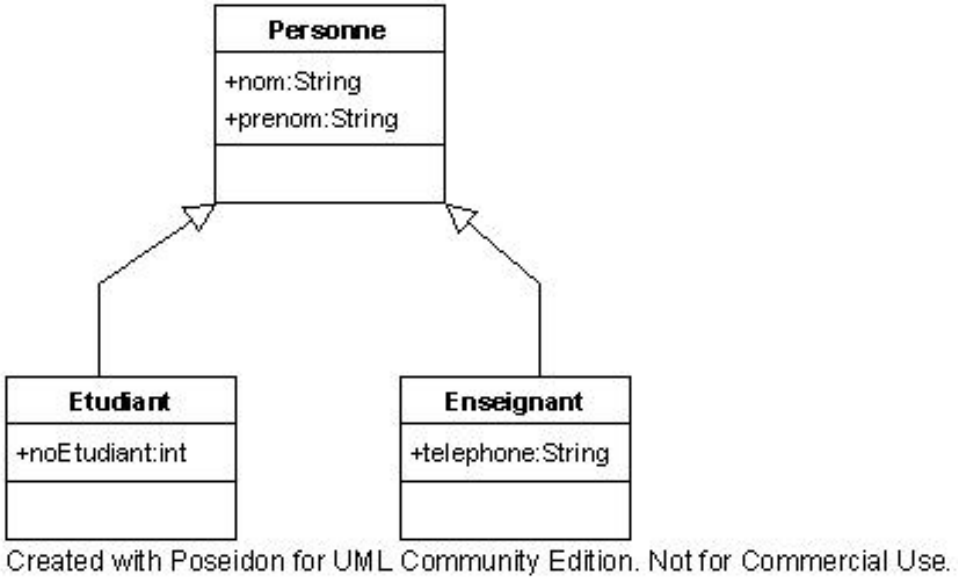
Diagramme de classes



Interpréter le diagramme de classes suivant afin de donner une spécification en langage naturel.

Implémentation : Héritage

```
public class Personne {
        public String nom;
        public String prenom;
}
```

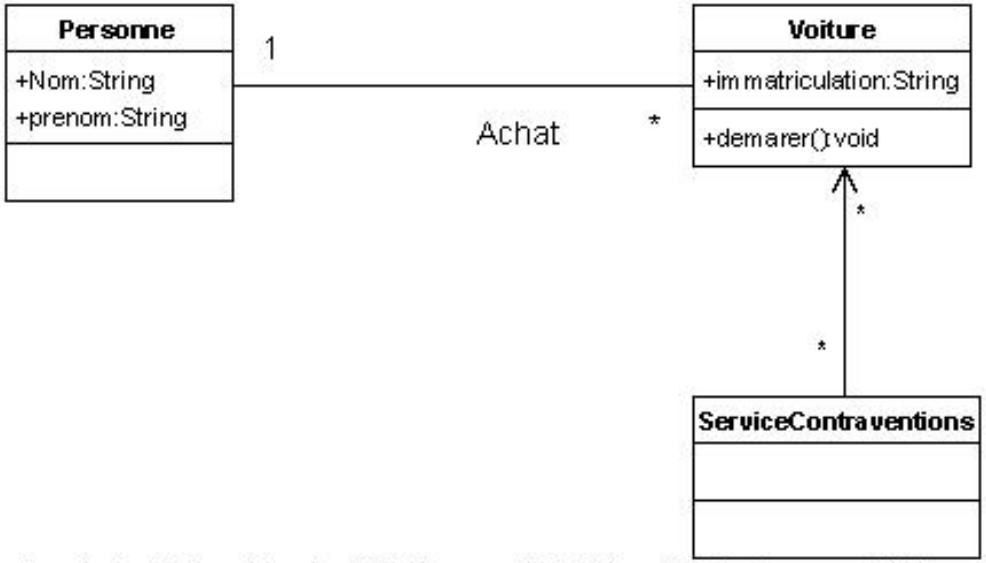



```
public class Etudiant extends Personne {
       public int noEtudiant;
}
```

Implémentation : Associations

```
public class Personne {

   public String Nom;
   public String prenom;
   public java.util.Collection voiture =
                new java.util.TreeSet();
}
```



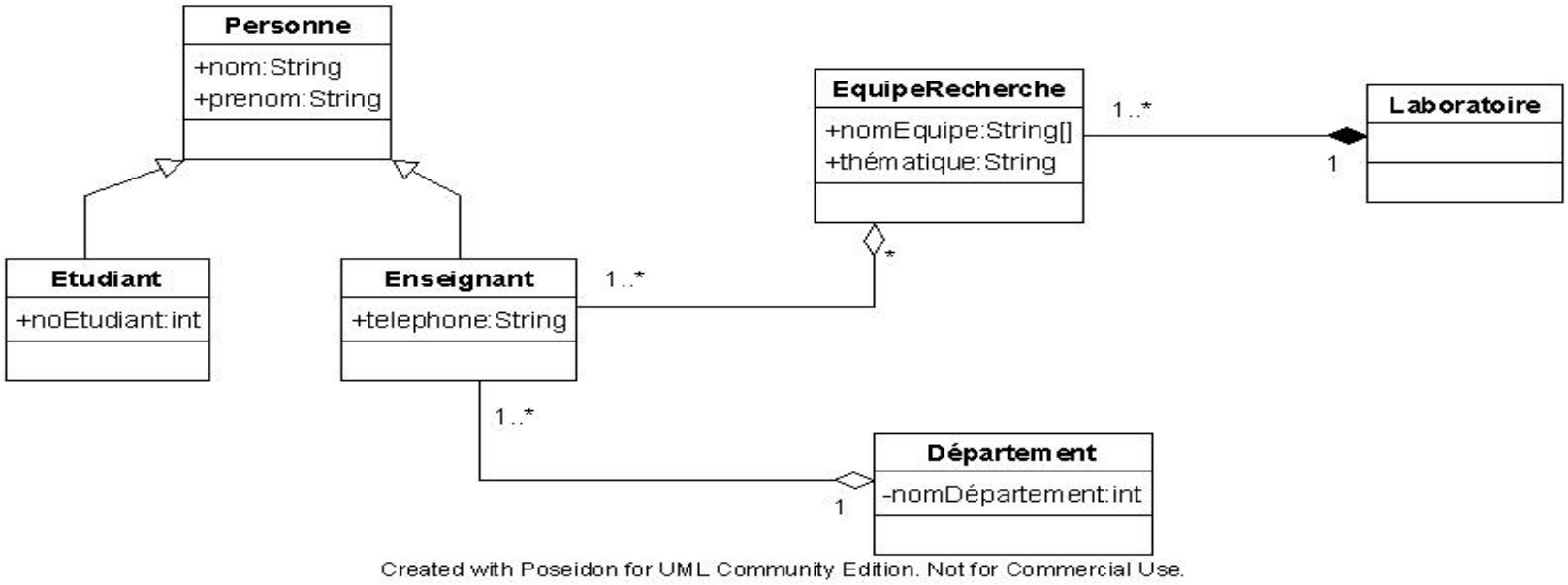

```
public class Voiture {

          public String immatriculation;
          public Personne Propriétaire;
          public void demarer() {  }
}
```

```
public class ServiceContraventions {

public java.util.Collection Voiture = new java.util.TreeSet();
}
```

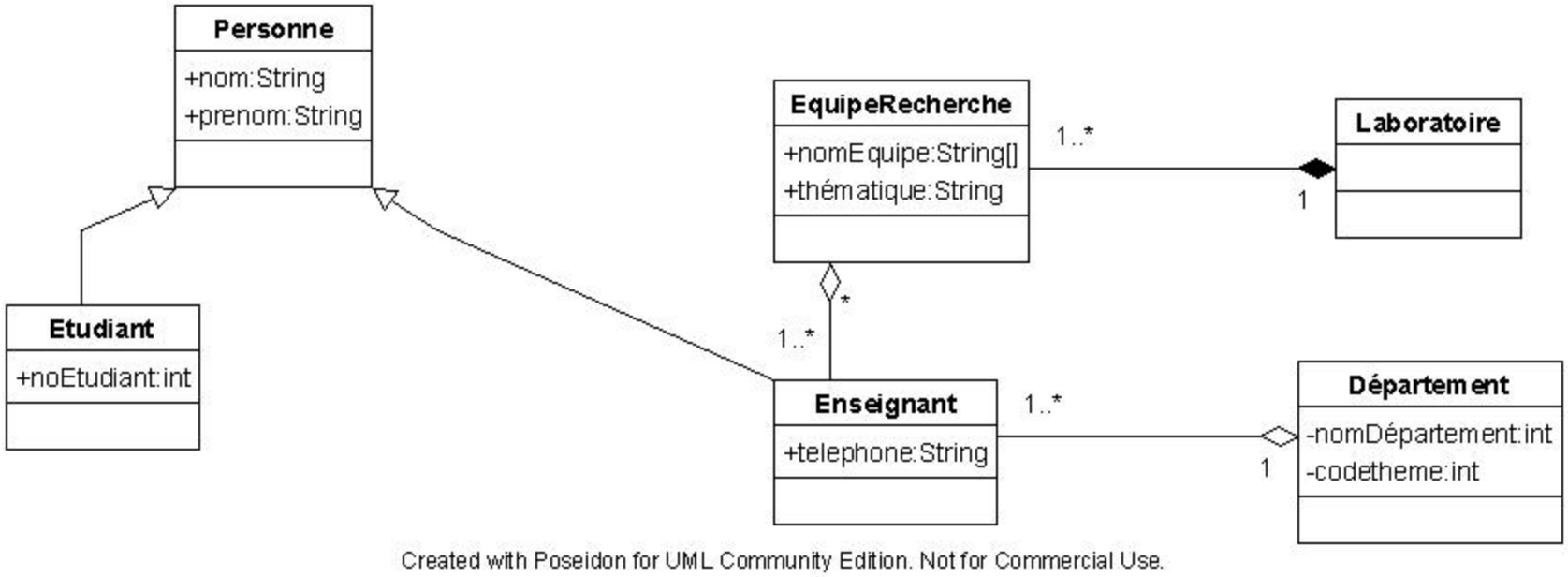
Implémentation : Agrégation




```
public class Enseignant extends Personne {
  public String telephone;
  public java.util.Collection equipeRecherche = new java.util.TreeSet();
  public Departement departement;
 }
```

```
public class Département {
private int nomDépartement;
private int codetheme;
public java.util.Collection enseignant = new java.util.TreeSet();
}
```

Implémentation : Composition




```
public class EquipeRecherche {
public String[] nomEquipe;
public String thématique;
public java.util.Collection enseignant = new java.util.TreeSet();
public Laboratoire laboratoire;
}
```

```
public class Laboratoire {
public java.util.Collection equipeRecherche = new java.util.TreeSet();
}
```

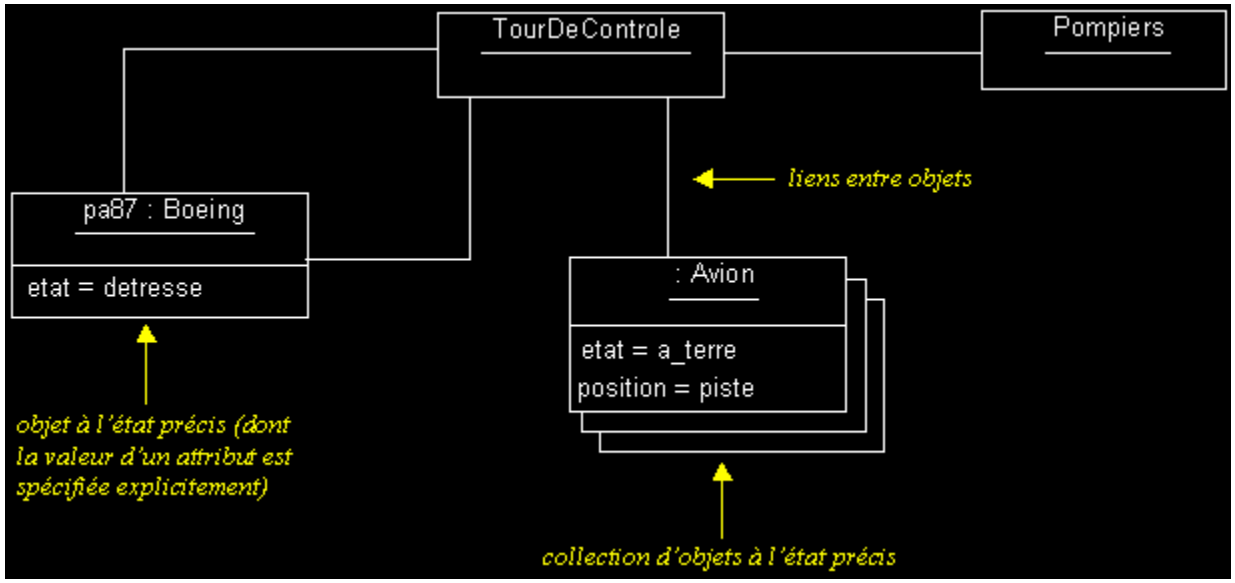
Les diagrammes UML

- **Vues statiques**
  - Les diagrammes de classes
  - Les diagrammes d'objets
  - Les diagrammes de cas d'utilisation
  - Les diagrammes de composants
  - Les diagrammes de déploiement

- **Vues dynamiques**
  - Les diagrammes de séquence
  - Les diagrammes de collaboration
  - Les diagrammes d'états-transition
  - Les diagrammes d'activités

Diagramme d'objets

montre des objets (instances de classes dans un état particulier) et des liens (relations sémantiques) entre ces objets.

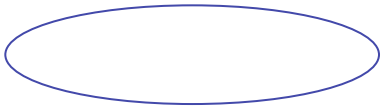


Cas d'utilisation

- structurer les besoins des utilisateurs et les objectifs correspondants du système.

- Préoccuper des cas "réels" des utilisateurs ; ils ne présentent pas de solutions d'implémentation et ne forment pas un inventaire fonctionnel du système.

Notation



Cas d'utilisation

Objectif du système, motivé par un besoin d'un (ou plusieurs) acteur(s)

Acteur

Personne ou composant d'origine d'une interaction avec le système

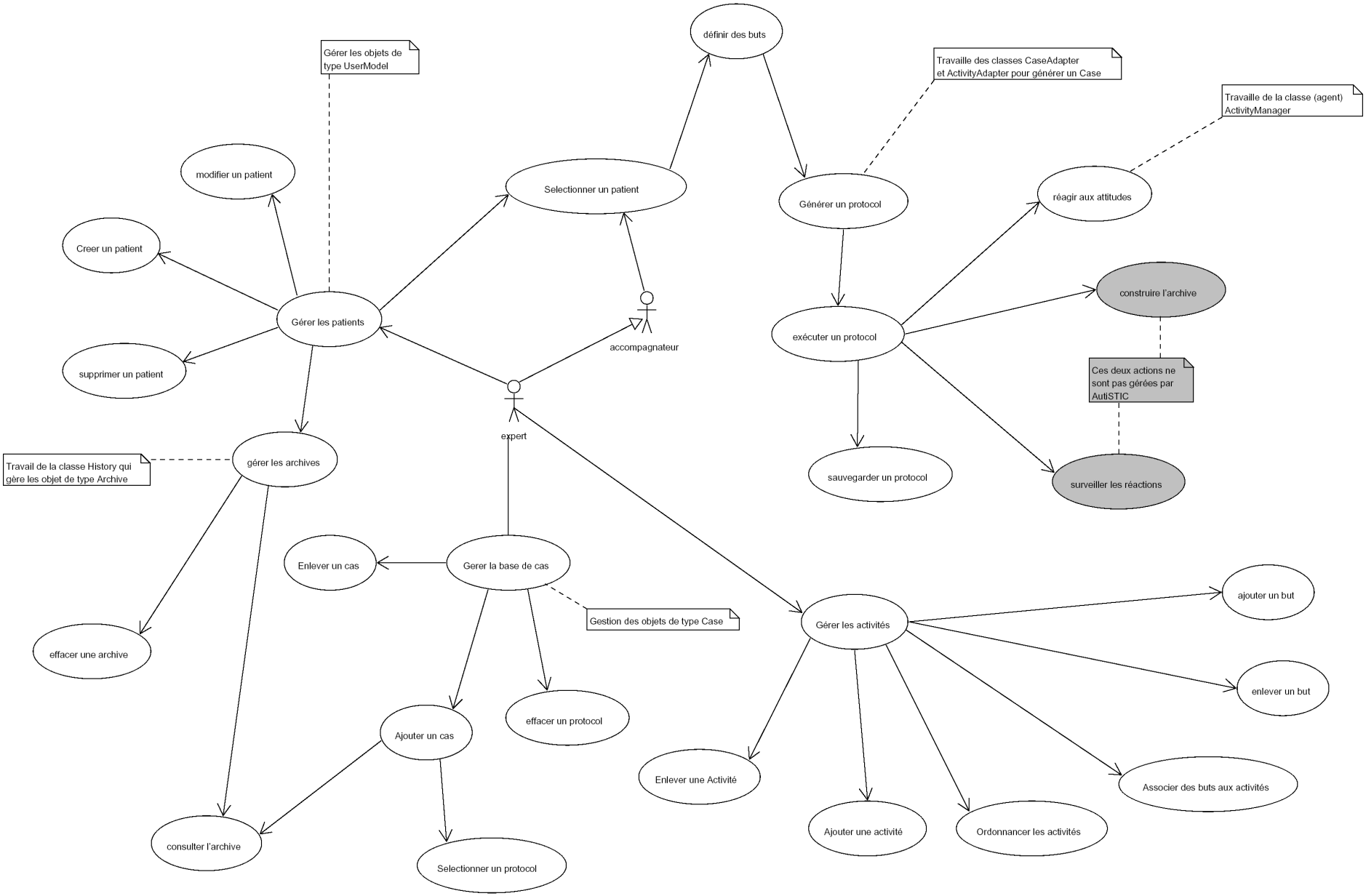
Note

Documente un élément du modèle

Relation d'utilisation

Le cas source contient aussi le comportement décrit dans la cas destination

Cas d'utilisation



Exercice

Concevoir le **diagramme de classe** d'une application de gestion d'hôtel. Voici ce que vous devez modéliser :

Un hôtel est constitué d'un certain nombre de chambres. Un responsable de l'hôtel gère la location des chambres. Chaque chambre se loue à un prix donné.

L'accès aux salles de bain est compris dans le prix de la location d'une chambre. Certaines chambres comportent une salle de bain, mais pas toutes. Les hôtes de chambres sans salle de bain peuvent utiliser une salle de bain sur le palier. Ces dernières peuvent être utilisées par plusieurs hôtes.

Les pièces de l'hôtel qui ne sont ni des chambres, ni des salles de bain (hall d'accueil, cuisine...) ne font pas partie de l'étude (hors sujet).

Des personnes peuvent louer une ou plusieurs chambres de l'hôtel, afin d'y résider. En d'autre termes : l'hôtel héberge un certain nombre de personnes, ses hôtes (il s'agit des personnes qui louent au moins une chambre de l'hôtel...).