Centre de Réadaptation de Mulhouse

Modélisation UML

Utiliser la notation UML pour représenter les diagrammes de classes et d'objets

Plan

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

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: l'objet ou l'instance

Notion d'*Objet*

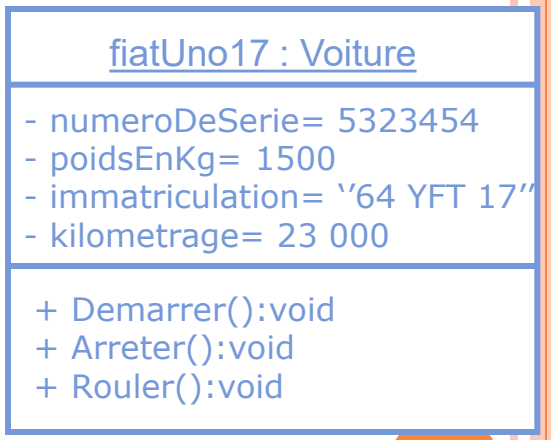
Une abstraction du monde réel c.-à-d. des données informatiques regroupant des caractéristiques du monde réel

Exemple

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

Caractérisation d'un objet

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



Modélisation objet: l'objet ou l'instance

- C'est une entité atomique qui possède :
  - une identité qui le caractérise de façon non ambiguë,
  - un état représenté par le contenu de ses attributs et les liens qu'il a avec les autres objets,
  - un comportement qui regroupe les compétences d'un objet et décrit ses actions et ses réactions (messages reçus , messages envoyés).
- Les objets communiquent entre eux par des messages.
- La persistance des objets est la capacité pour un objet de sauvegarder son état dans un système de stockage de l'information.

Modélisation objet

Un objet a une vie :

- il naît,
- il vit,
- il meurt.

Il est symbolisé par :

ou

ou

Le stéréotype de la classe peut surmonter le nom

Modélisation objet: la classe

Notion de *Classe*

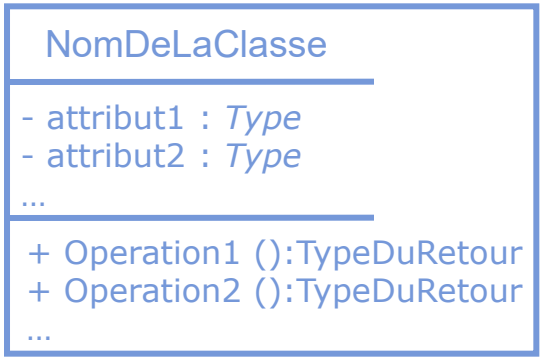
- Structure d'un objet, c.-à-d. une déclaration de l'ensemble des entités qui composeront l'objet
- Un objet est donc "issu" d'une classe, c'est le produit qui sort d'un moule

Notation

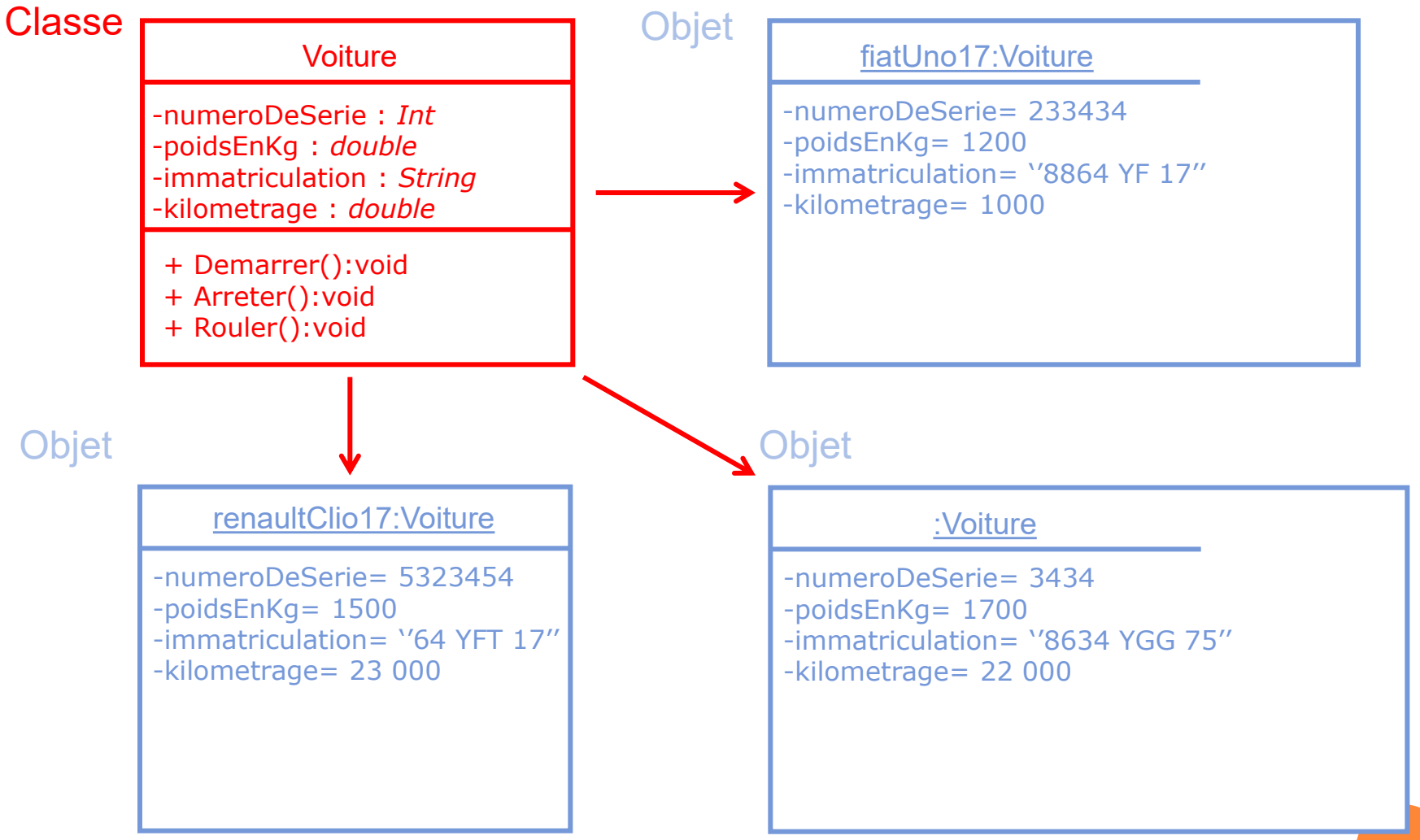
un objet est une **instance** (***occurrence***) issu de l'instanciation d'une classe

Une classe est composée:

- **attributs**
  données dont les valeurs représentent l'état de l'objet
- **opérations (équivalent à une méthode)**
  opérations applicables aux objets



Modélisation objet



Modélisation objet

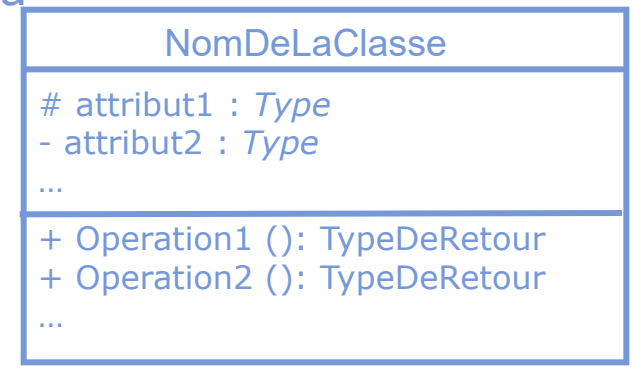
Visibilité des attributs

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



Attribut de classe et attribut d'instance

- Un attribut d'instance a une valeur différence d'un objet à l'autre.
- Pour atteindre un attribut d'instance :

  nomObjet.nomAttribut

- Un attribut de classe a une valeur pour la classe; il est précédé par le mot clé **« static »**.
- Pour atteindre un attribut de classe :

  NomClasse.nomAttribut

Méthode de classe et méthode d'instance

- Une méthode d'instance représente un comportement propre à chaque instance.
- Une méthode de classe représente un comportement commun et ne dépendant donc pas de l'état de l'objet.
- Pour atteindre une méthode d'instance :
  nomObjet.Methode();
- Une méthode de classe est précédé par le mot clé **« static »**.
- Pour atteindre une méthode de classe :
  NomClasse.Methode();

Classes et objets: Les catégories de méthode/opération

- *Méthodes Constructeur*

Elles permettent de construire un nouvel objet, une nouvelle instance de la classe

- *Méthodes de classe*

caractérisées en C# par le mot clé static

- *Méthodes d'instance :*

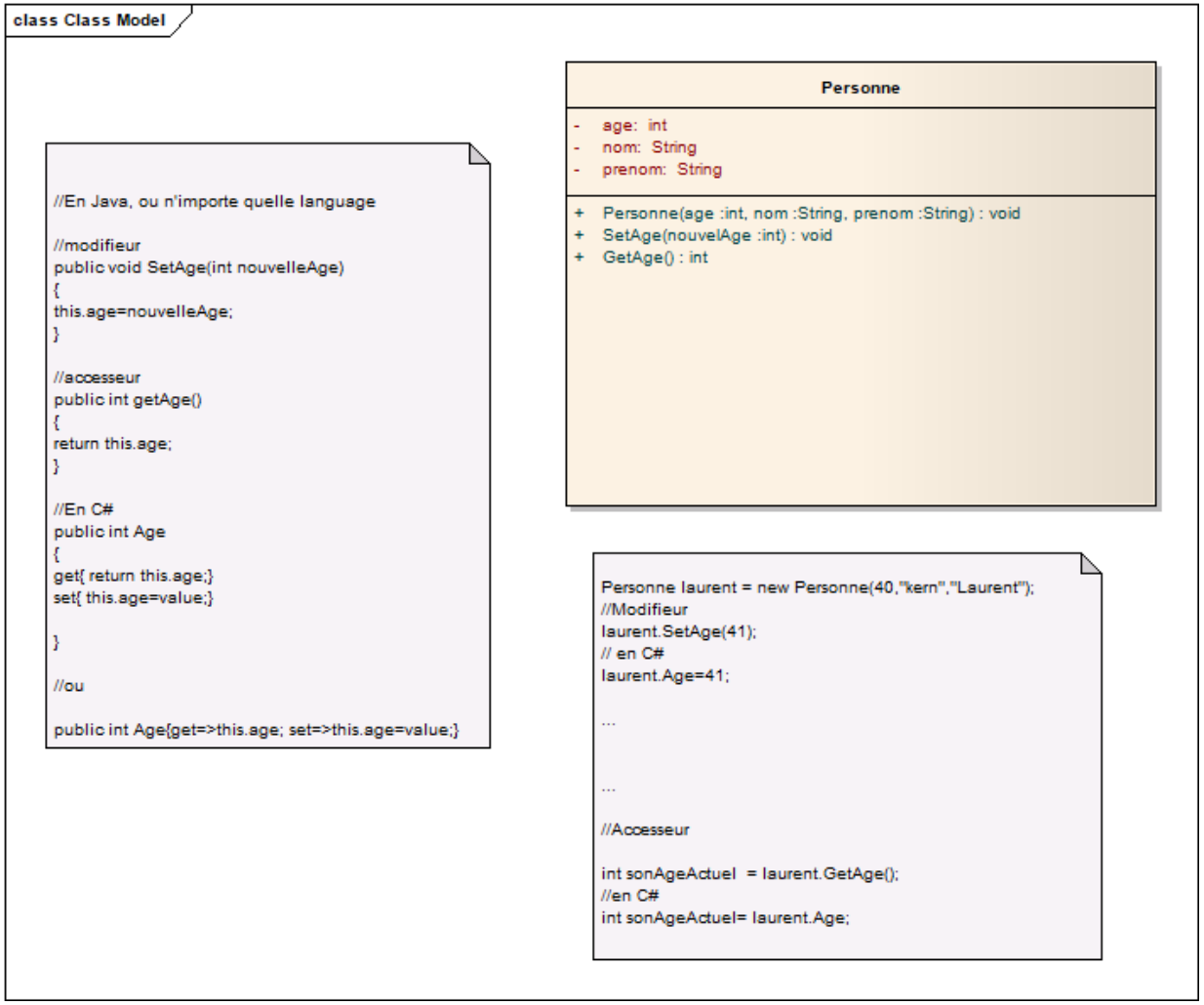
Elle s'adressent aux objets et non à la classe

  - *Modifieur : méthode qui met à jour une donnée privée (« set»)*
  - *Accesseur : méthode retournant la valeur d'une donnée privée (« get »)*
  - *Autres méthodes*

- *Méthodes Destructeur*

Elles permettent de détruire les objets.

Exemple Accesseur/Modifieur



Types de relation entre classes

Héritage

Association

Contenance

Types de relation : Héritage

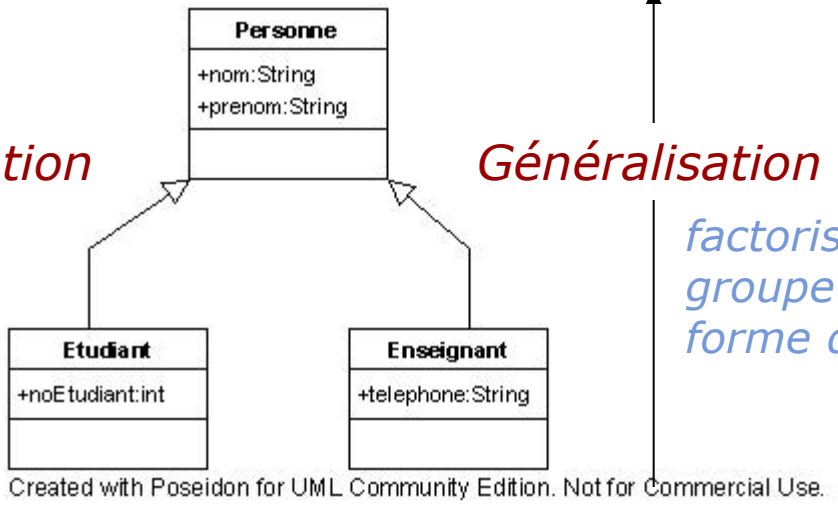
permet de créer une nouvelle classe à partir d'une classe existante

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

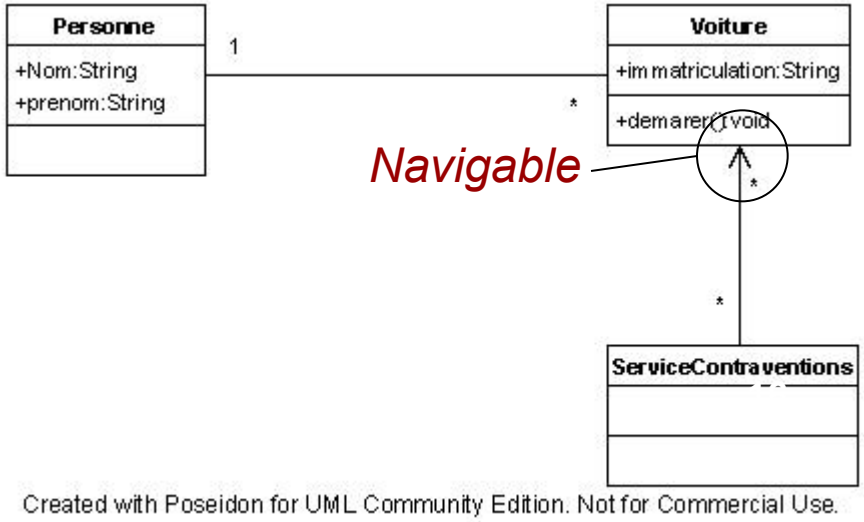
Connexion sémantique entre deux classes

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

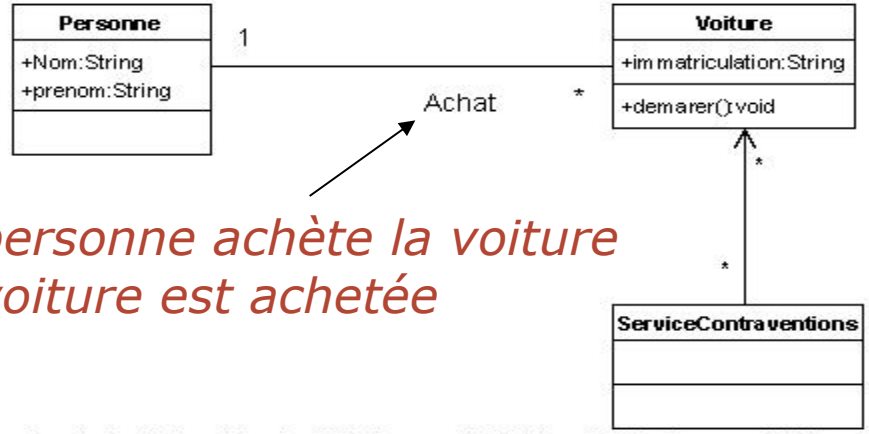


Types de relation : Association

Documentation d'une association

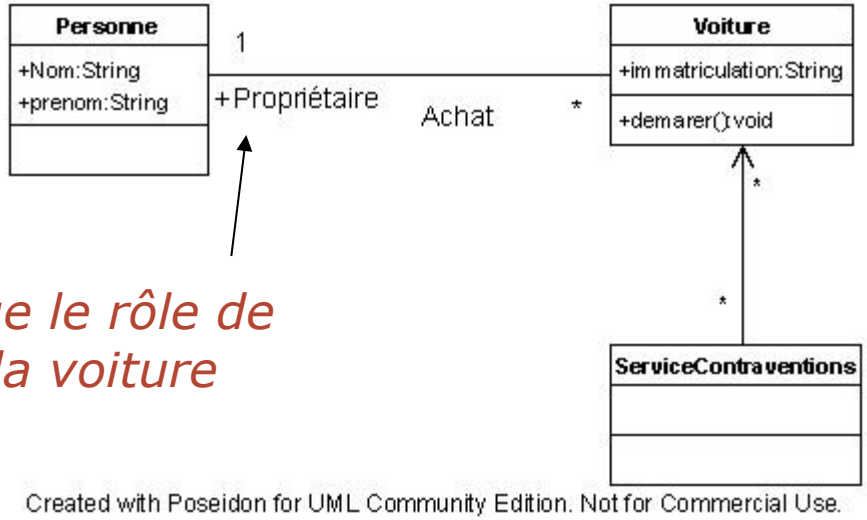
- **Nom de l'association**

  lien sémantique entre les classes



- **Rôle d'une association**

  Spécification du rôle de la classe



Types de relation : Association

Multiplicités

1 : la classe est en relation avec un et un seul objet de l'autre classe

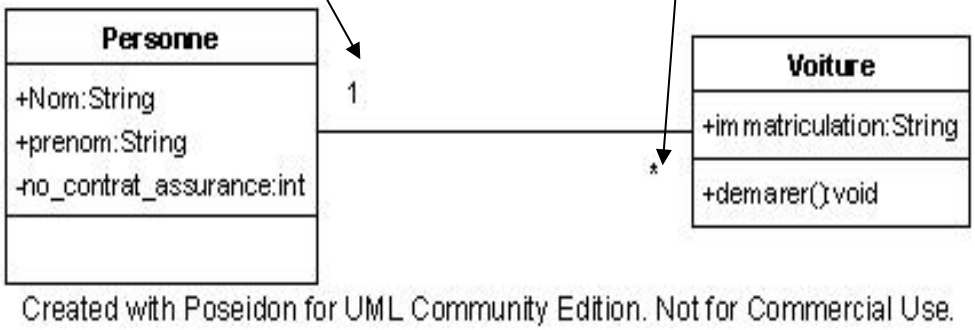
1..* : la classe est en relation avec au moins un objet de l'autre classe

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



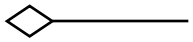
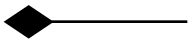
Types de relation : Contenance

Cas particulier d'association exprimant une relation de contenance

Exemples:

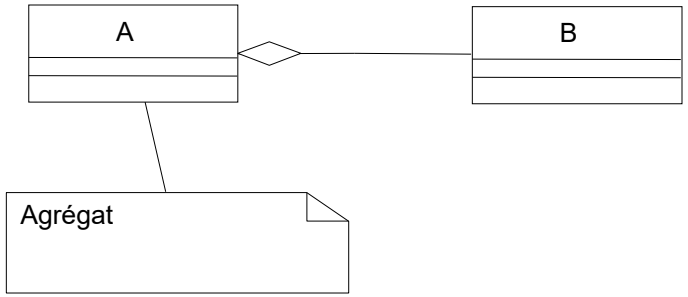
- Une voiture a 4 roues
- Un dessin contient un ensemble de figures géométriques
- Une présentation PowerPoint est composé de transparents
- Une équipe de recherche est composée d'un ensemble de personnes

Deux types de relations de contenance en UML

- Agrégation 
- Composition (Agrégation forte) 

Types de relation : Agrégation

- L'agrégation est une forme particulière d'association non symétrique qui exprime l'idée qu'un objet « fait partie » d'un autre objet. Sa sémantique détermine généralement un lien fort entre les classes concernées. L'une des classes est composée d'instances issues de l'autre classe et joue donc un rôle de conteneur.



Type de relations

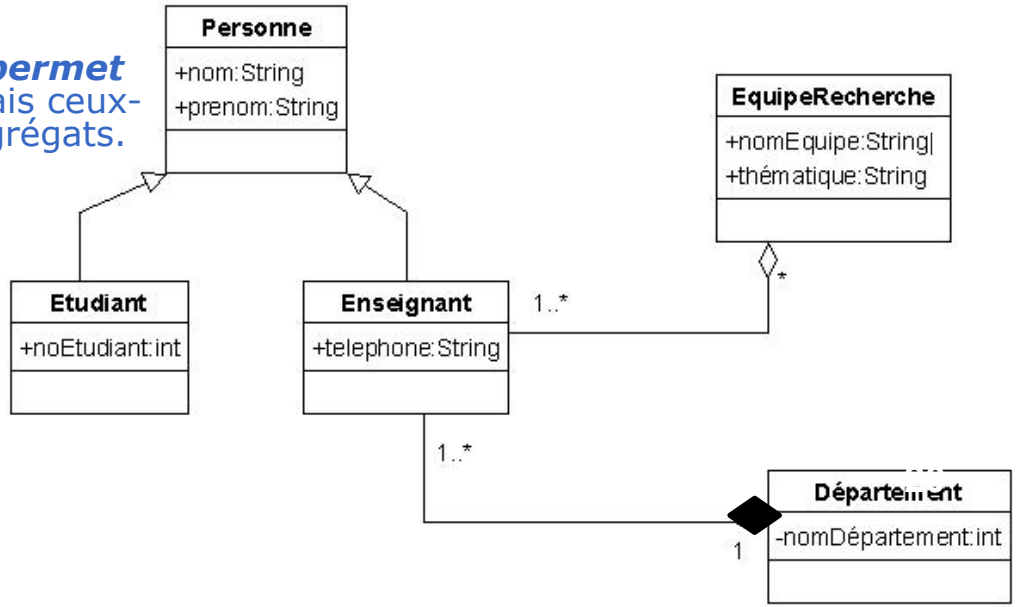
- A « contient » des instances de B

Propriétés de l'agrégation

- La suppression de A n'implique pas la suppression de B
- L'élément agrégé peut être partagé: ***: elle permet de regrouper*** des éléments indépendants mais ceux-ci peuvent appartenir également à d'autres agrégats.

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
- *A possède toujours un B, qui peut être remplacé*

***Exemple**:*

« Une présentation PowerPoint est composé de transparents »

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

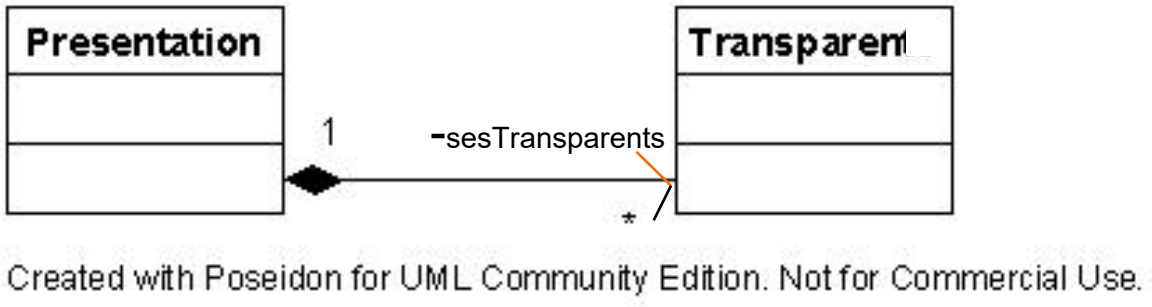
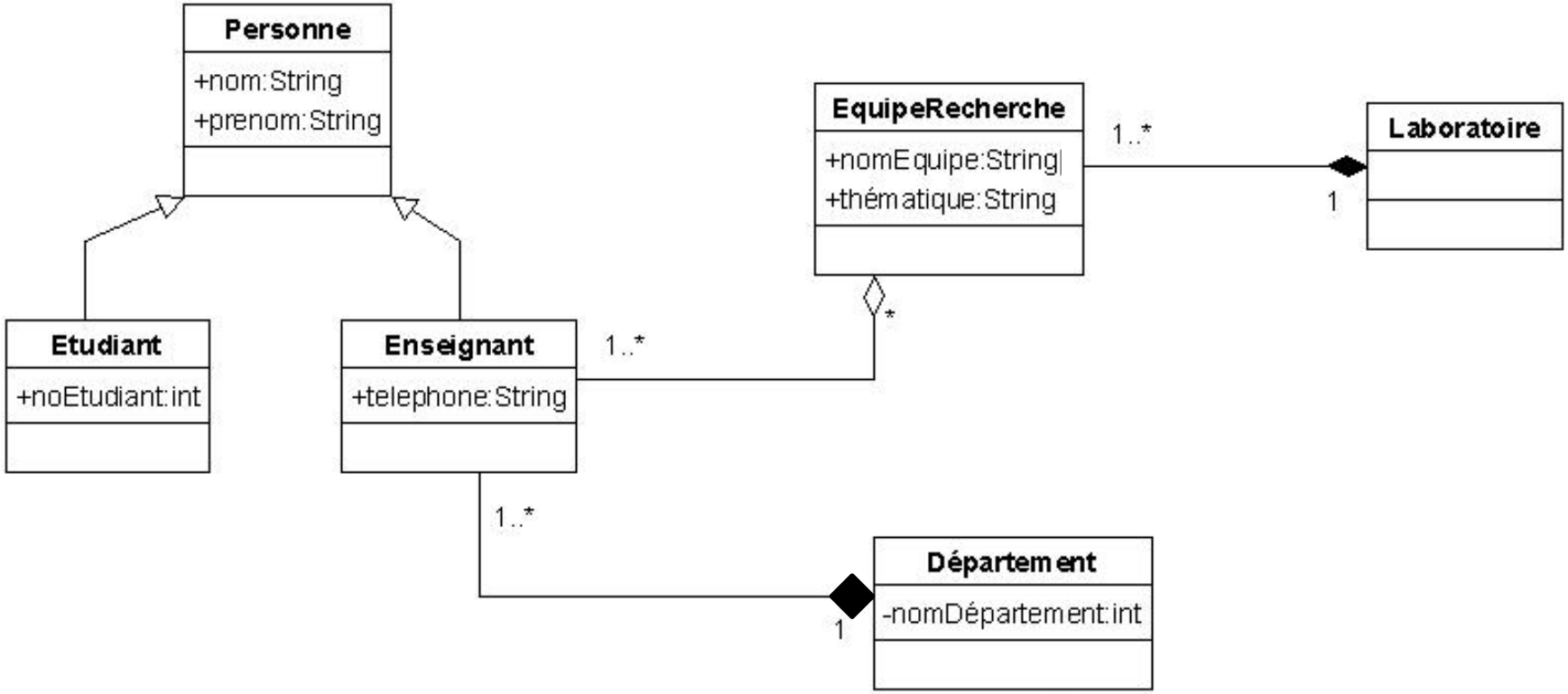
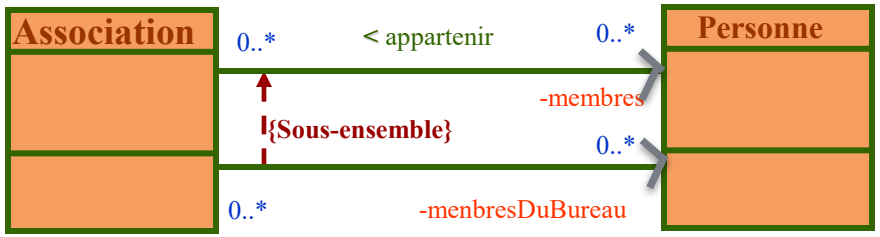


Diagramme de classes



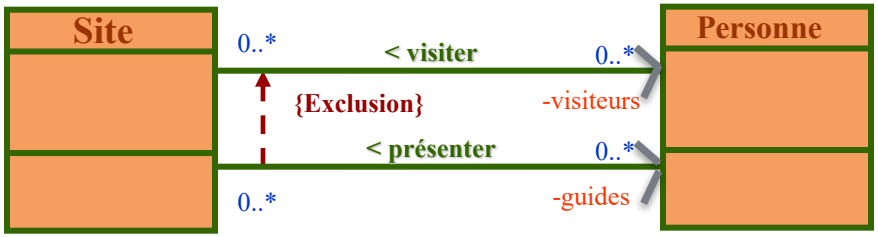
Contraintes sur les relations

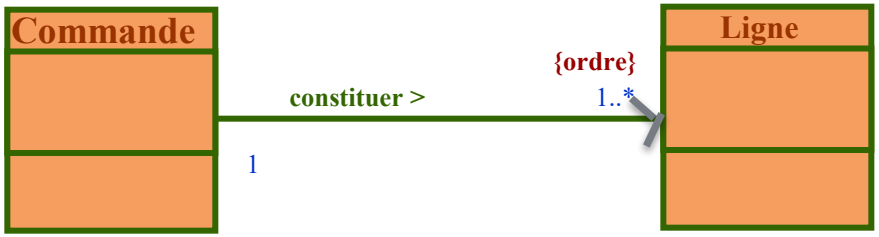
Les contraintes sont représentées par des expressions placées entre accolades.



Contrainte « sous-ensemble »

Contrainte «d'exclusion »



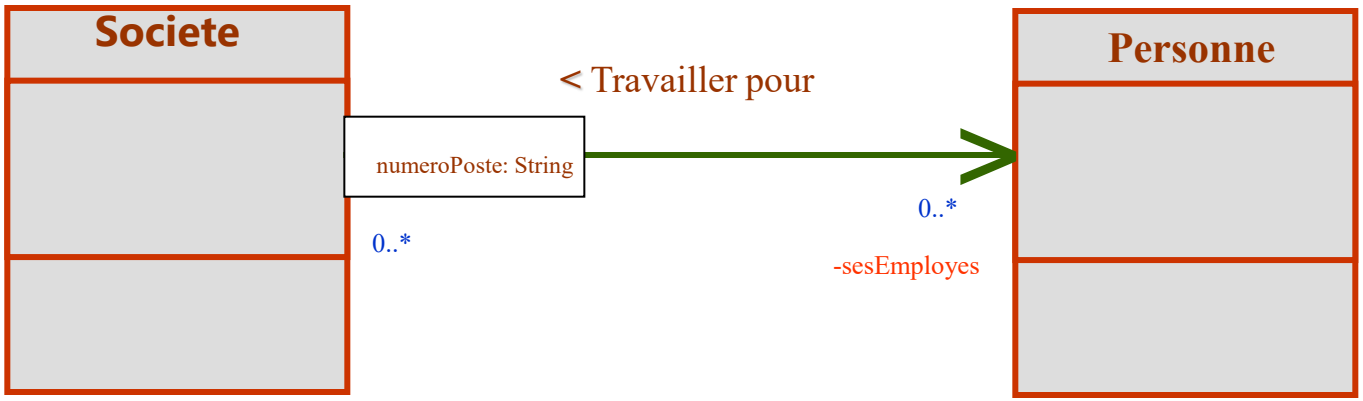


Contrainte «d'ordre »

Qualification

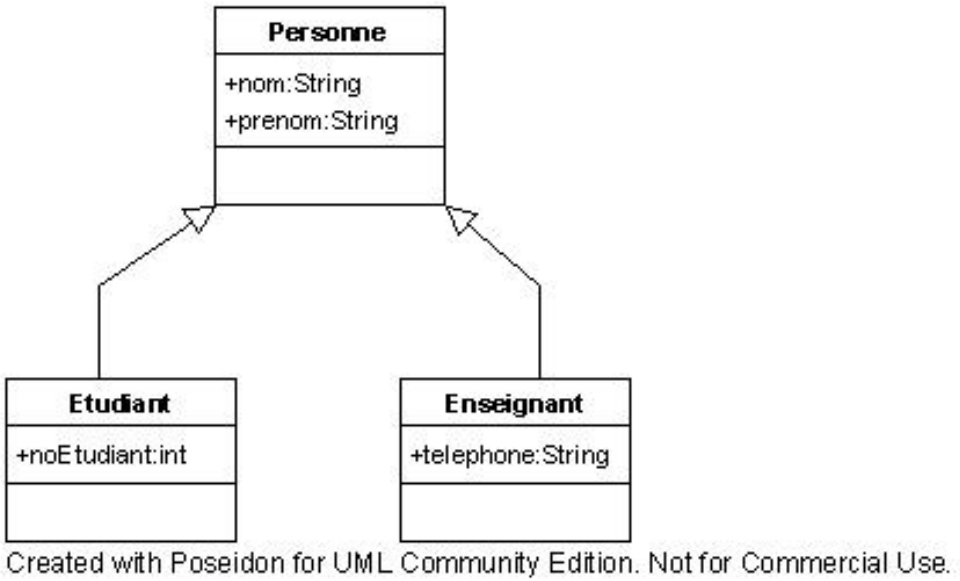
Elle permet de sélectionner un sous-ensemble d'objet parmi les instances d'objets qui participent à une association.

Un qualificatif est un attribut d'association dont les valeurs partitionnent la liste des objets mis en relation par une association



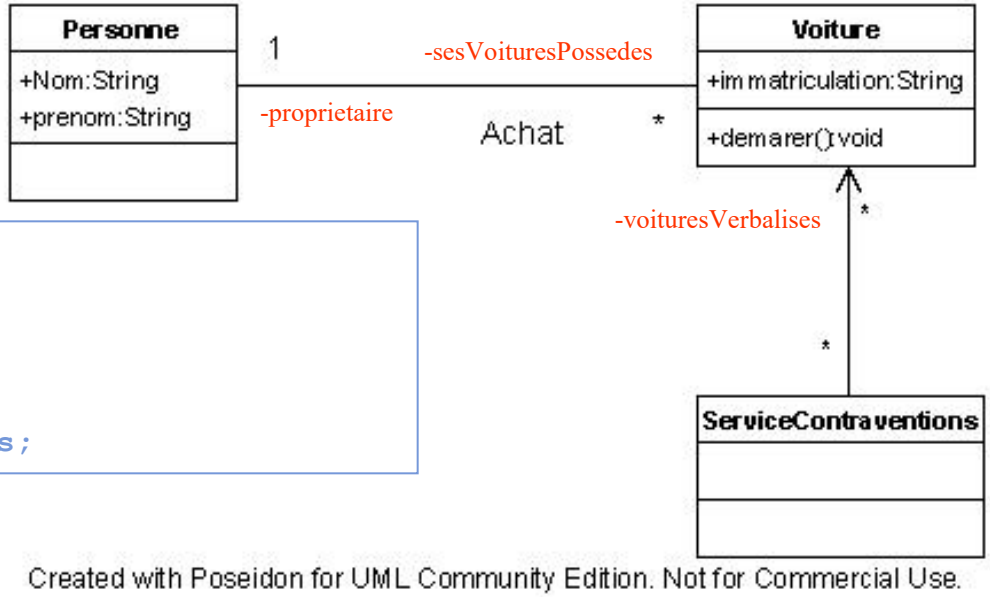
Implémentation : Héritage

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




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

Implémentation : Associations




```
public class Personne
{

   public string nom;
   public string prenom;
   public List<Voiture> sesVoituresPossedes;
```

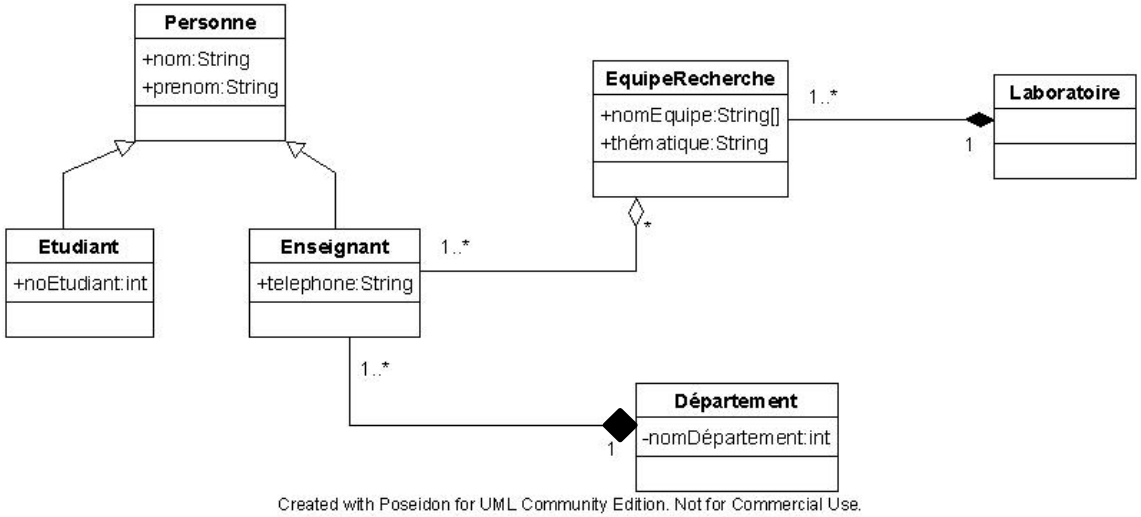
```
public class Voiture
{

        public string immatriculation;
        public Personne propriétaire;
        public void Demarrer() {  }
}
```

```
public class ServiceContraventions
{

public List<Voiture> voituresVerbalises;
}
```

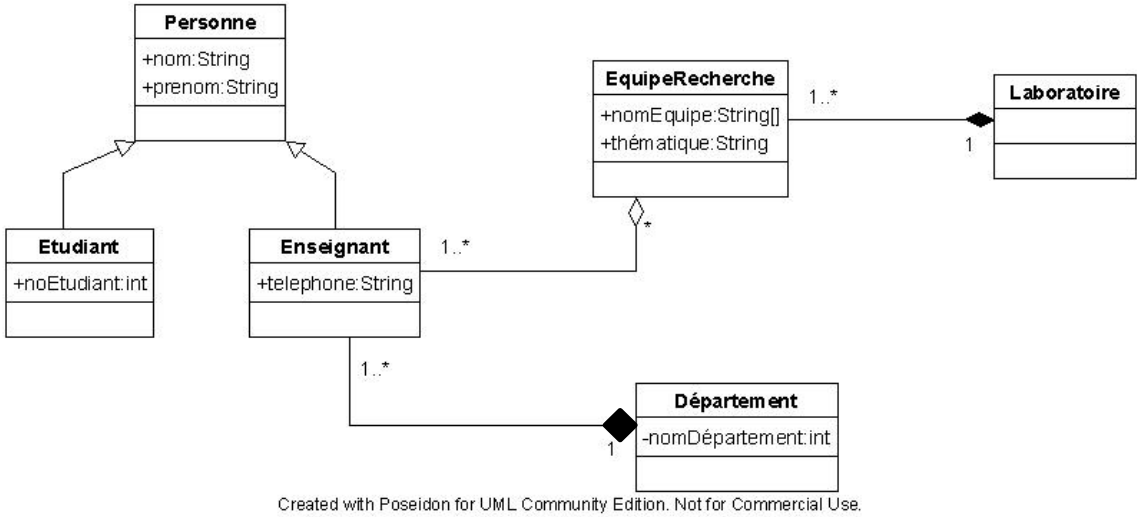
Implémentation : Agrégation




```
public class Enseignant:Personne
{
  public string telephone;
  public List<EquipeRecherche> equipeRecherches;
  public Departement departement;
 }
```

```
public class Département
{
   private int nomDépartement;
   private int codetheme;
   public List<Enseignant> enseignants;
}
```

Implémentation : Composition




```
public class EquipeRecherche
{
   public String[] nomEquipe;
   public String thématique;
   public List<Enseignant> enseignants;
   public Laboratoire laboratoire;
}
```

```
public class Laboratoire
{
    public List<EquipeRecherche> equipeRecherches;
}
```

Différence agrégation et composition

- Une composition est une agrégation forte. Elle est composée à tout moment d'élément. Ces éléments peuvent changer avec le temps.

```
public class Laboratoire
{
        public List<EquipeRecherche> equipeRecherches;

public Laboratoire()
{
        equipeRecherches = new List<EquipeRecherche>();
        equipeRecherches.Add(new EquipeRecherche());//1 au minimum
}
```

EXEMPLES: DIAGRAMME DE CLASSE ET DIAGRAMME D'OBJET

Diagramme d'objets

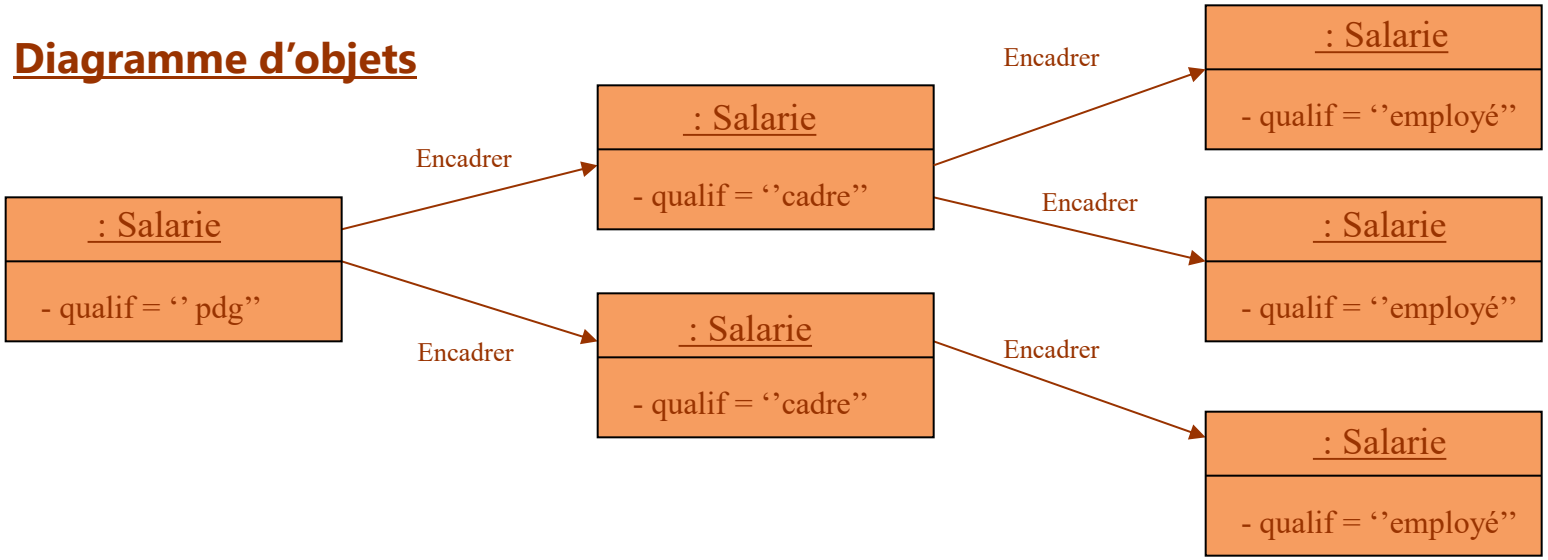
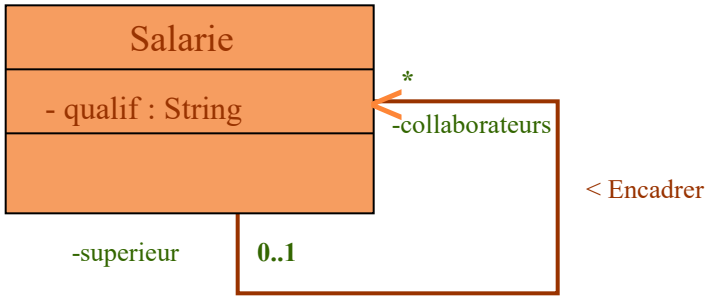
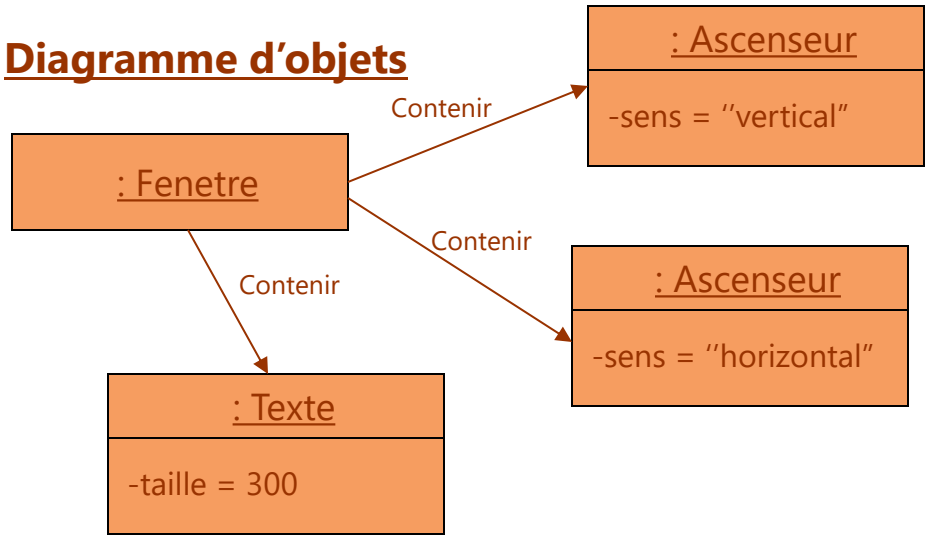


Diagramme de classes



EXEMPLES: DIAGRAMME DE CLASSE ET DIAGRAMME D'OBJET

Diagramme d'objets



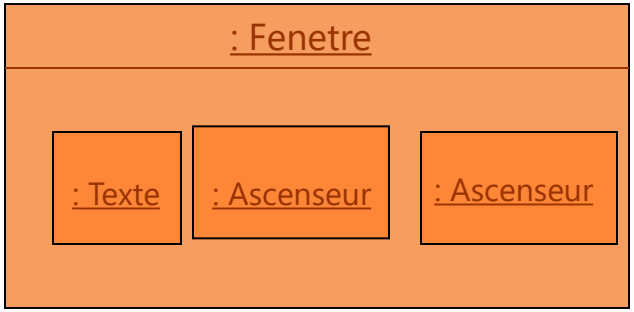


Diagramme de classes