Diagramme de classes

Diagramme de classes

Permet de donner une vue statique du système en terme de :

- Classes d'objets
- Relations entre classes
  - Associations
  - agrégation/composition
  - héritage

La description du diagramme de classes est centrée sur trois concepts :

- Le concept d'objets
- Le concept de classes d'objets comprenant des attributs et des opérations
- Les différents types de relations entre classes.

Concept d'objet

Objet = un concept, abstraction ou une chose autonome qui a un sens dans le contexte du système à modéliser

- une **personne** : le client « Jean Pierre »
- un **objet concret** : le livre intitulé « Initiation à… »
- un **objet abstrait** : le compte bancaire n° 1915233…

Concept d'attribut

Un attribut est une propriété, caractéristique d'un objet. Par exemple :

- un client a un **nom**, un **prénom**, une **adresse**, un **code client**, …
- un compte bancaire a un **numéro**, un **solde**, …

Un attribut doit (généralement) avoir une **valeur atomique**

Concept d'attribut

La description d'un attribut comporte :

Visibilité attribut:type[= valeur initiale]

Où :

Visibilité :

- + (publique, public) : visible par tous
- - (privée, private) : visible seulement dans la classe
- # (protégée, protected) : visible seulement dans la classe et dans les sous-classes de la classe.

Nom d'attribut

Type de l'attribut

Valeur initiale (facultative)

Concept d'attribut

Le type d'un attribut peut être :

- Un type de base : entier, réel, …
- Une expression complexe : tableaux, enregistrements, …
- Une classe

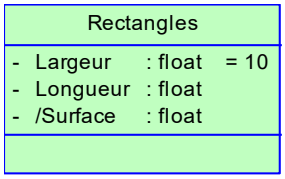
Exemples d'attributs :

- *- couleur : enum{Rouge, Vert, Bleu}*
- *# b : boolean = vrai*
- *- Client : Personne*

Concept d'attribut

Lorsqu'un attribut peut être dérivé ou calculé à partir d'autres attributs, il est précédé d'un **/**.
Par exemple, une classe « Rectangle » peut contenir les attributs suivants :

- longueur : réel,
- largeur : réel,
- /surface : réel.



Concept d'attribut

On distingue deux types d'attributs :

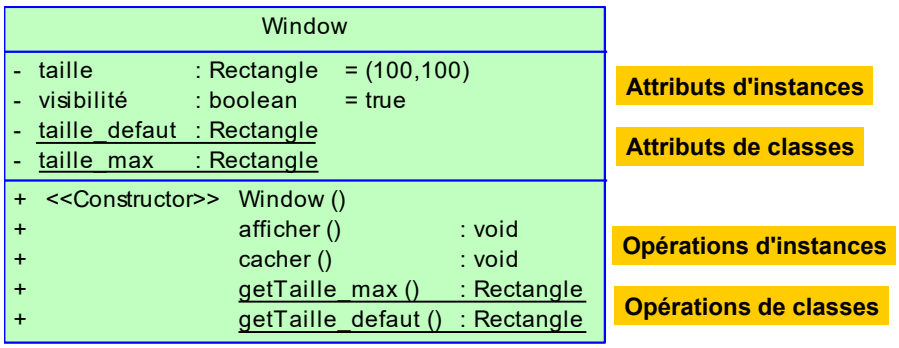
Attribut d'instance :

- Chaque instance de la classe possède une valeur particulière pour cet attribut
- Notation : **Visibilité attribut:type[= valeur initiale]**

Attribut de classe

- Toutes les instances de la classe possède la même valeur pour cet attribut
- Notation : **Visibilité attribut:type[= valeur initiale]**
- Équivalent en C++, Java : static

Concept d'attribut



Concept d'opération et méthode

Une opération est :

- un service offert par la classe
- une fonction ou une transformation qui peut être appliquée aux objets d'une classe.
- permet de décrire le comportement d'un objet. Par exemple, « Embaucher », « Licencier » et « Payer » sont des opérations de la classe « Société ».

Concept d'opération et méthode

Une méthode est

- l'implémentation d'un service offert par la classe (opération).
- de différents types :
  - accesseurs (get...): renvoie une information sur l'état d'un objet (fonction)
  - modifieurs (set...): modifie l'état de l'objet (procédure)
  - constructeurs: initialise une nouvelle instance

Concept d'opération et méthode

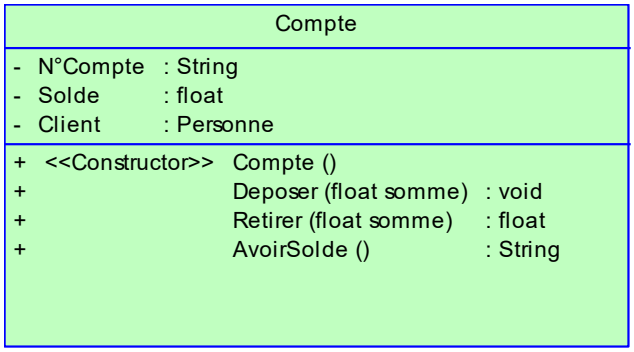
La description d'une opération comporte :

Visibilité opération([arguments:type[=valeur initiale]]):type de résultat

- Visibilité de l'opération (-, +, #)
- Nom de l'opération
- Liste des arguments avec leurs types et éventuellement leurs valeurs par défaut
- Le type du résultat retourné

Concept d'opération et méthode

Exemples d'opérations :



Concept de classes d'objets

Classe = ensemble d'objets ayant les mêmes propriétés (attributs) et le même comportement (opérations)

- tous les clients sont décrits par un **nom**, un **prénom**, … et peuvent **marcher**, **parler**,**courir**, …
- tous les comptes bancaires ont un **numéro**, un **solde**, … et sur lesquels on peut **déposer** ou **retirer** l'argent, ou les **consulter**
- …

Un objet est **instance** d'une classe, et le fait de créer un objet d'une classe est dite instanciation.

Concept de classes d'objets

Classe représentée par un rectangle à trois parties :

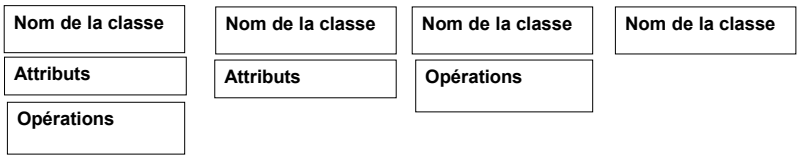
- Partie 1 : Nom de la classe
- Partie 2 : Attributs (propriétés, champs)
- Partie 3 : Méthodes (fonctions, opérations)

Concept de classes d'objets

Compte			
-	N°Compte	: String	
-	Solde	: float	= 100
#	Client	: Personne	
+	<<Constructor>>	Compte ()	
+		Deposer (float somme)	: void
+		Retirer (float somme)	: float
+		AvoirSolde ()	: String

Concept de classe d'objets

On peut ne pas visualiser les attributs et/ou les opérations, afin de ne pas alourdir inutilement le schéma.



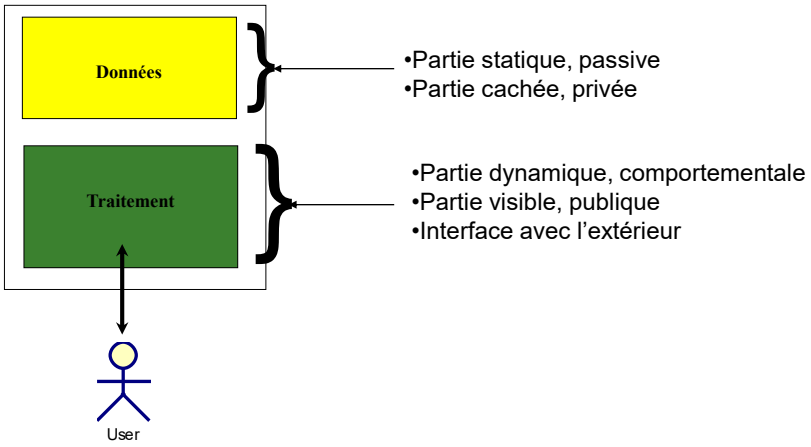
Encapsulation, visibilité et interface

Encapsulation est le mécanisme de regrouper les attributs et les opérations au sein d'une même entité (classe)

Ce regroupant permet d'empêcher d'accéder directement aux données par un autre moyen que les services proposés (opérations)

Ces services offerts aux utilisateurs définissent ce que l'on appelle l'interface de la classe.

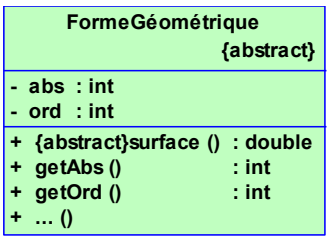
Encapsulation, visibilité et interface



Méthodes et classes abstraites

Une méthode est dite abstraite si on connaît son entête, mais pas la manière dont elle peut être réalisée.

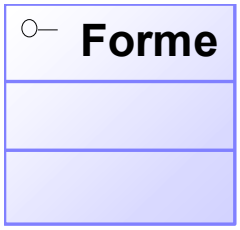
Une classe est dite abstraite lorsqu'elle définit au moins une méthode abstraite.

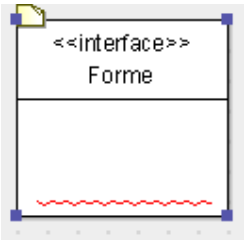


Classe « Interface »

Une interface est une classe spéciale dont toutes les méthodes sont abstraites

Une interface se note en UML avec le stéréotype <<interface>> ou symbole





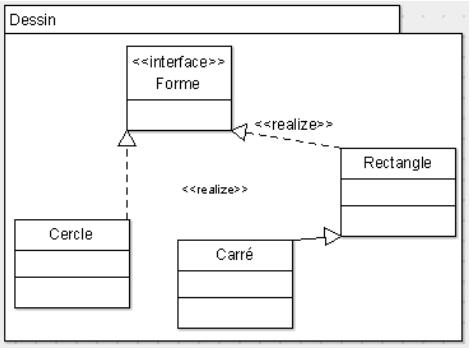
Package

Un package permet de regrouper des classes, des interfaces et des packages.

Les classes, les interfaces et les packages ne peuvent appartenir à un seul package dans lequel ils sont regroupés.

Package

Un package est représenté par un rectangle possédant un onglet dans lequel est inscrit le nom du package.



Import des packages

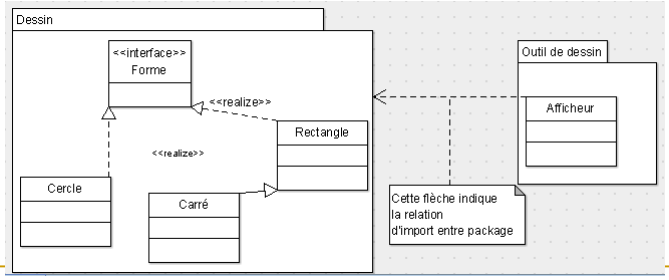
La relation d'import permet à une classe d'un package d'utiliser les classes d'un autre package.

La relation est monodirectionnelle : elle comporte un package source et un package cible.

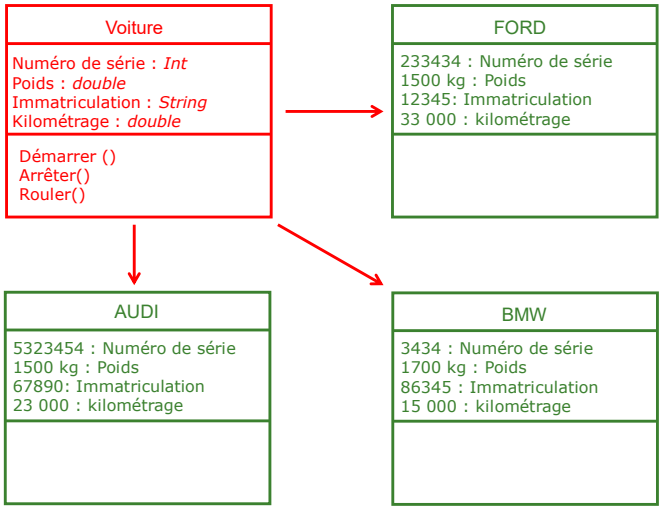
Import de packages

La relation d'import s'exprime avec une flèche en pointillé

Dans l'exemple, la classe 'Afficheur' a besoin des classes du package 'Dessin'



Modélisation objet



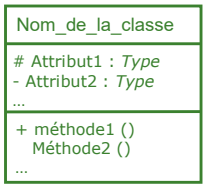
Modélisation objet

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

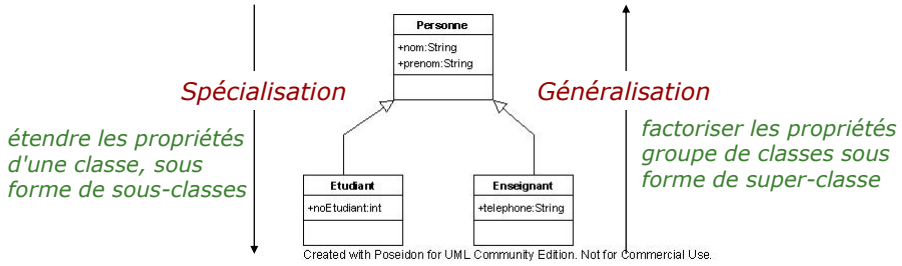
➢**Privée (-)**: l'accès aux données est limité aux méthodes de la classe elle-même



Types de relation : 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



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

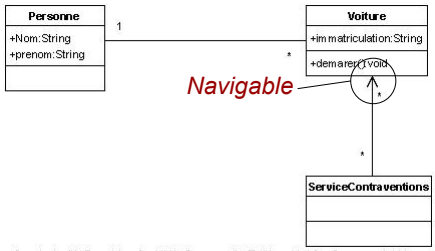
Navigabilité

- Par défaut une association est navigable dans les deux sens

Created with Poseidon for UML Community Edition. Not for Commercial Use.

  - 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

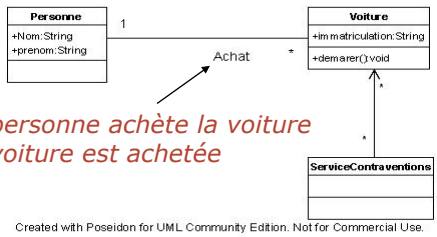


Created with Poseidon for UML Community Edition. Not for Commercial Use.

Types de relation : Association

Documentation d'une association

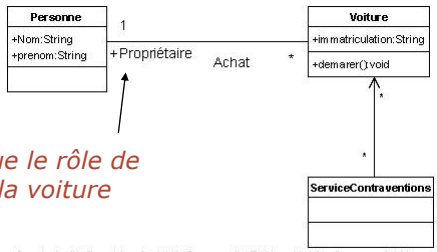
- **Nom de l'association**
  lien sémantique entre les classes



La personne achète la voiture
La voiture est achetée

Created with Poseidon for UML Community Edition. Not for Commercial Use.

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



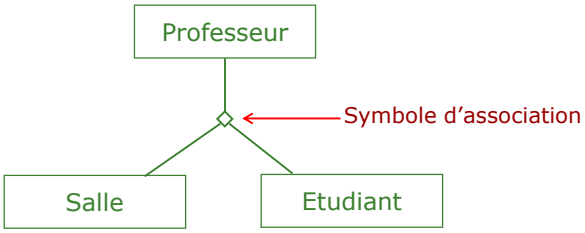
La personne joue le rôle de propriétaire de la voiture

Created with Poseidon for UML Community Edition. Not for Commercial Use.

Types de relation : Association

Relation n-aire

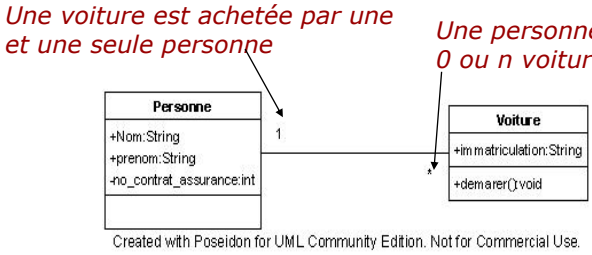
Type particulier d'association qui relie plus de deux classes



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



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

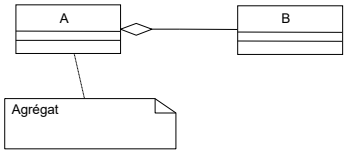
Deux types de relations de contenance en UML

- Agrégation ◇——
- Composition (Agrégation forte) ◆——

Types de relation : Agrégation

Type de relations

- A « contient » des instances de B,



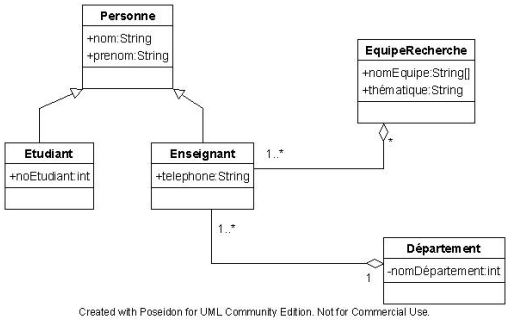
Propriétés de l'agrégation

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

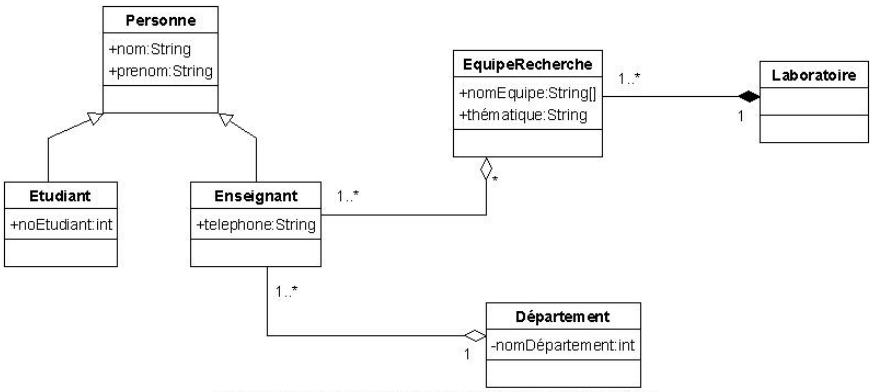
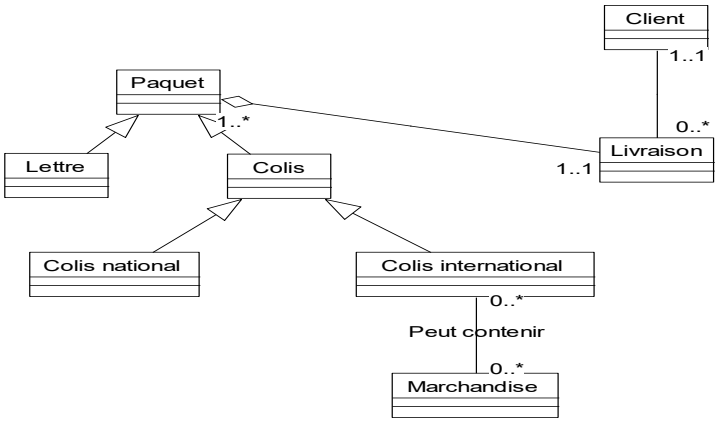


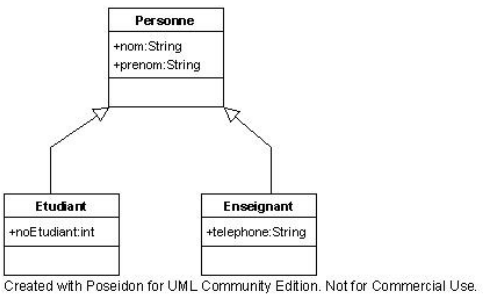
Diagramme de classes



Exercice : 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;
}
```

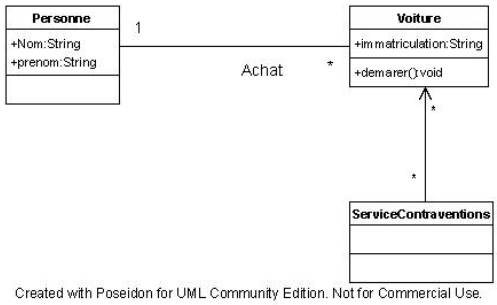



```
public class Etudiant extends Personne {
       public int noEtudiant;
}
```

Implémentation : Associations

```
public class Personne {

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



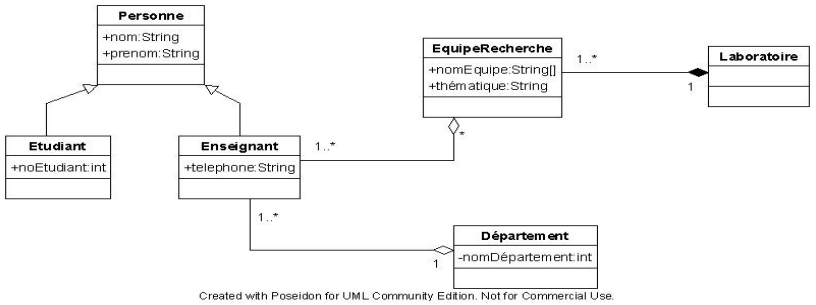

```
public class Voiture {

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

```
public class ServiceContraventions {

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

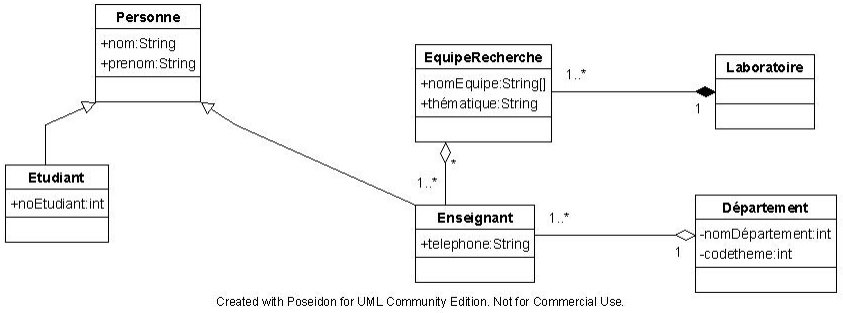
Implémentation : Agrégation




```
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();
}
```

À faire

1. Les exercices dans LEA
2. Le TP3 dont énoncé est dans LEA