

# Programmation Orientée Objet

# Programmation Orientée Objet

On quitte un monde où on a :

- des données d'un côté et
- des "traitements" de l'autre.

Pour un univers focalisé sur des entités complexes, les objets qui interagissent les uns avec les autres.

Un objet a deux parties :

- ses données et
- ses comportements.



attributs ou  
propriétés

méthodes

# Programmation Orientée Objet

"Orienté objet", signifie qu'on organise le logiciel comme une collection d'objets.

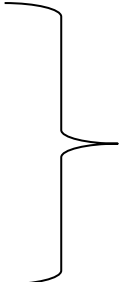
Les objets sont regroupés en catégories d'objets similaires ayant des :

- structures similaires (mêmes attributs ou propriétés) et
- des comportements (méthodes) communs.

⇒ les classes (  $\cong$  types d'objets )

# Programmation Orientée Objet

## Classe :

- Structure de données (attributs)
  - Fonctions de manipulation
    - constructeurs (initialisation des objets)
    - getters et setters (accès aux attributs)
    - autres fonctions,....  
(manipulation des objets)
- 
- Méthodes

## Généralement

- On cache la structure (les attributs)
- On met en avant les comportements (méthodes)

⇒ Encapsulation

# Programmation Orientée Objet

## Notion de *classe* (type)

La *classe* permet regrouper des objets de même nature.

Une *classe* définit des *attributs* (structure) et des *comportements* (méthodes).

Par exemple, la classe *Humain* définit

- des attributs : *nom*, *dateNais*, ...
- des fonctions (méthodes) :  
*presenteToi()*, *calculAge()*

Humain
nom dateNais nationalité
presenteToi() calculAge()

Par exemple, la classe *Point* définit

- les coordonnées du points, *x*, *y*
- des fonctions de manipulation :  
*translater()*, ...

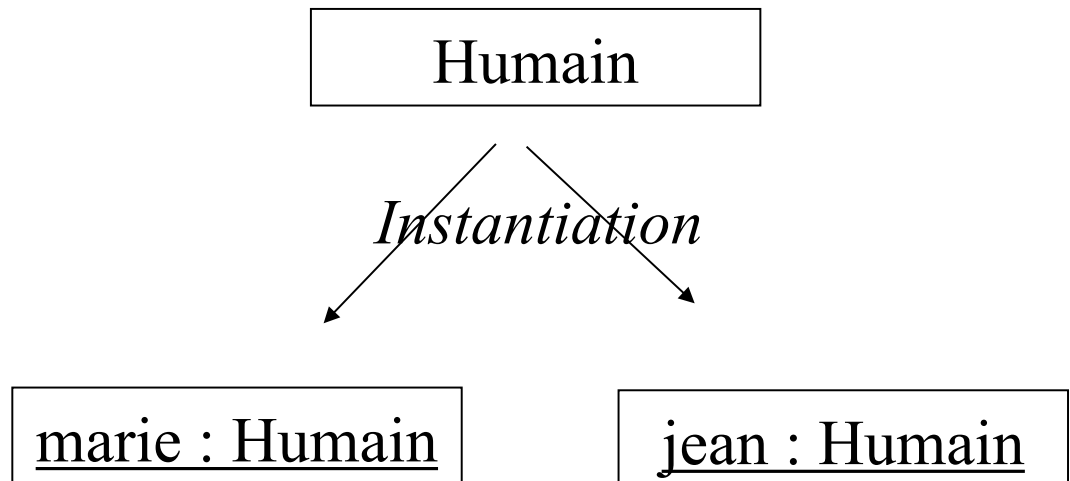
Point
x y
translater()

# Programmation Orientée Objets

## Notion d' *objet*

Un objet est une *instance* ou un *exemplaire* d'une *classe*.

Par exemple, *marie* ou *jean* sont des **instances** de la classe ***Humain***, c'est-à-dire des humains ayant leur propriétés spécifiques



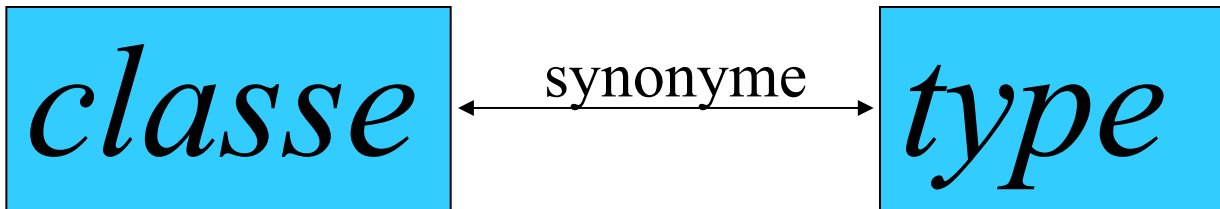
# Programmation Orientée Objet

Tout objet a un type (une classe).

*marie est de type Humain*

*marie est une instance de la classe Humain*

La classe est le modèle, le type d'un objet.



# Programmation Orientée Objet

Exemples de *Classes* et d'*objets* :

<b><i>Classe, type, modèle</i></b>	<b><i>objet, instance, exemplaire</i></b>
Animal	babar, medor
Point	A(0,0), B(5,10)
Humain	jean, marie
Université	univSavoie, cambridge
Musée	louvre, quaiBranly
Date	17/10/2007, 31/3/1985



# Programmation Orientée Objet

Programme : ensemble de Classes

Classe :

- Structure de données (attributs)
- Fonctions de manipulation
  - constructeurs (initialisation des objets)
  - getters et setters (accès aux attributs)
  - autres fonctions,....  
(manipulation des objets)

Méthodes

# Programmation Orientée Objet : en Java

```
public class Compte {  
    // attributs  
    private String numero;  
    private double solde;  
    private Client detenteur;  
  
    //méthodes  
    public void debiter(double montant ) {  
        solde = solde - montant ;  
    }  
    public void crediter(double montant ) {  
        solde = solde + montant ;  
    }  
    public double getSolde() {  
        return solde ;  
    }  
}
```

Compte
numero
solde
detenteur
crediter() debiter()

# Conception Orientée Objet

Identifier les classes (types) à modéliser :

- les attributs de ces classes,
- les manipulations de ces classes (méthodes)
- les relations entre ces classes

Notation

UML : Unified Modeling Language

# Conception Orientée Objet

## UML : Unified Modeling Language

- C'est ...
  - Une notation standard pour spécifier, construire, visualiser et documenter les composants d'un logiciel.
  - Fondé sur le concept d'objet
  - Notation graphique
- Ce n'est pas...
  - Une méthode. Elle ne décrit aucune démarche.

# Conception Orientée Objet

## UML : les différents diagrammes

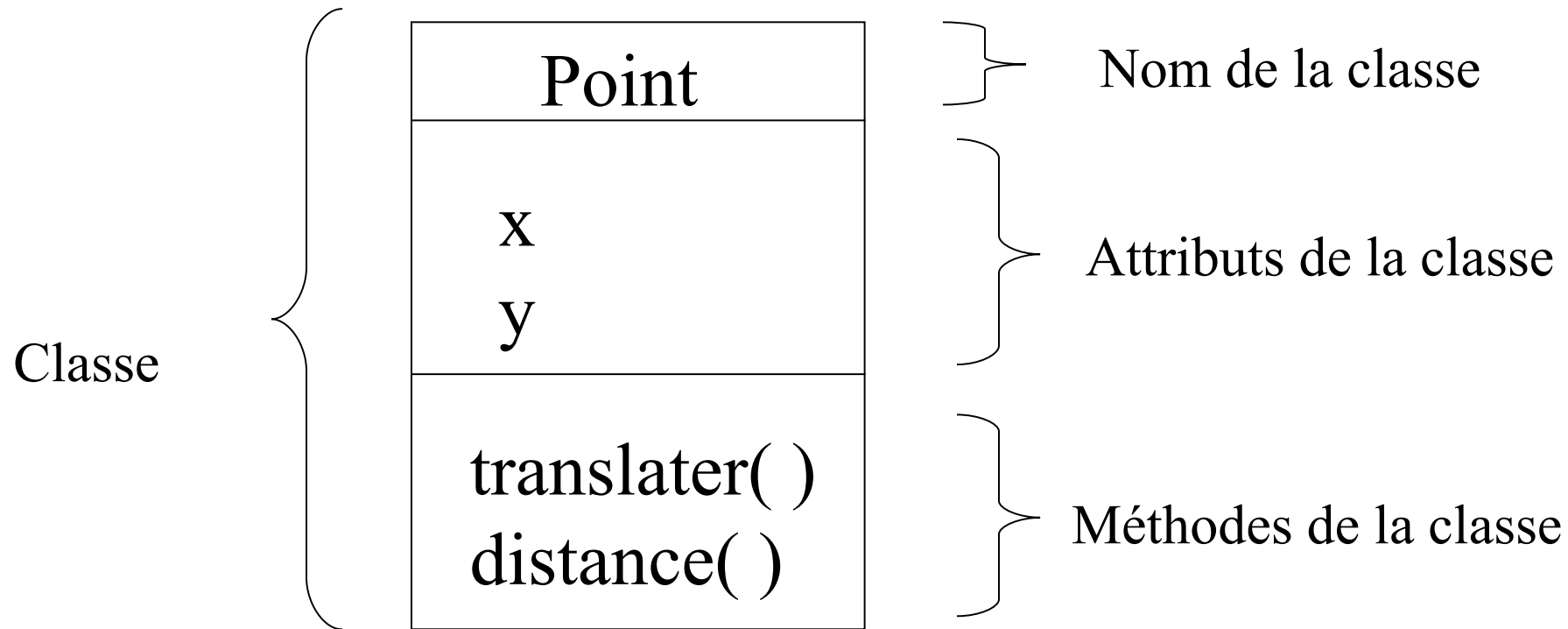
Un diagramme UML est une représentation graphique, qui s'intéresse à un aspect précis du modèle.

### **Diagramme de classes**



Description de la structure statique du système :

- les classes
  - attributs
  - méthodes
- leur organisation
  - hiérarchie
  - relations entre classes

# Conception Orientée Objet



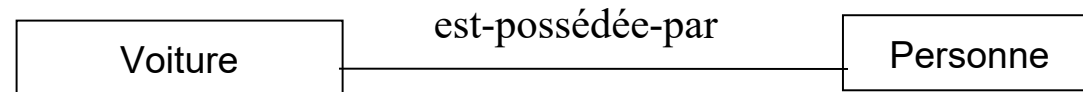
Relation entre classes :

- Relation : \_\_\_\_\_
- Agrégation :  \_\_\_\_\_
- Héritage :  \_\_\_\_\_

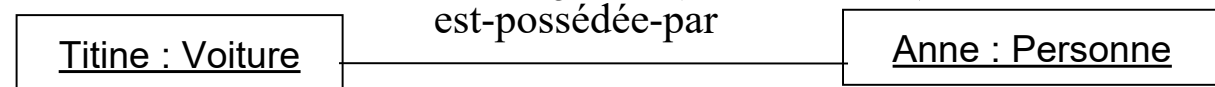
# Conception Orientée Objet

## Les relations entre classes

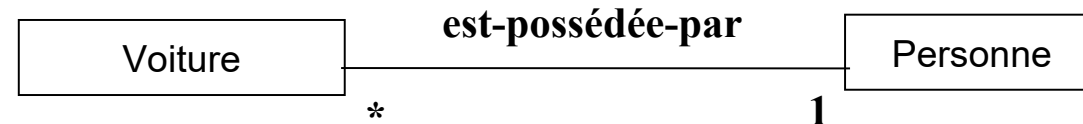
- Association :
  - Relation structurelle entre classes



- est réalisée au niveau des objets (instances)



- Cardinalité :

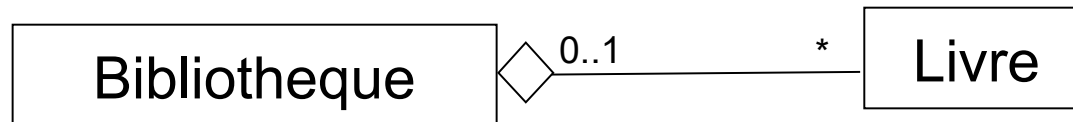


Une Personne peut posséder plusieurs Voitures

# Conception Orientée Objet

## Les relations entre classes

- Agrégation :
    - une association particulière,
    - relation de type "ensemble / élément".
- « est composé de »

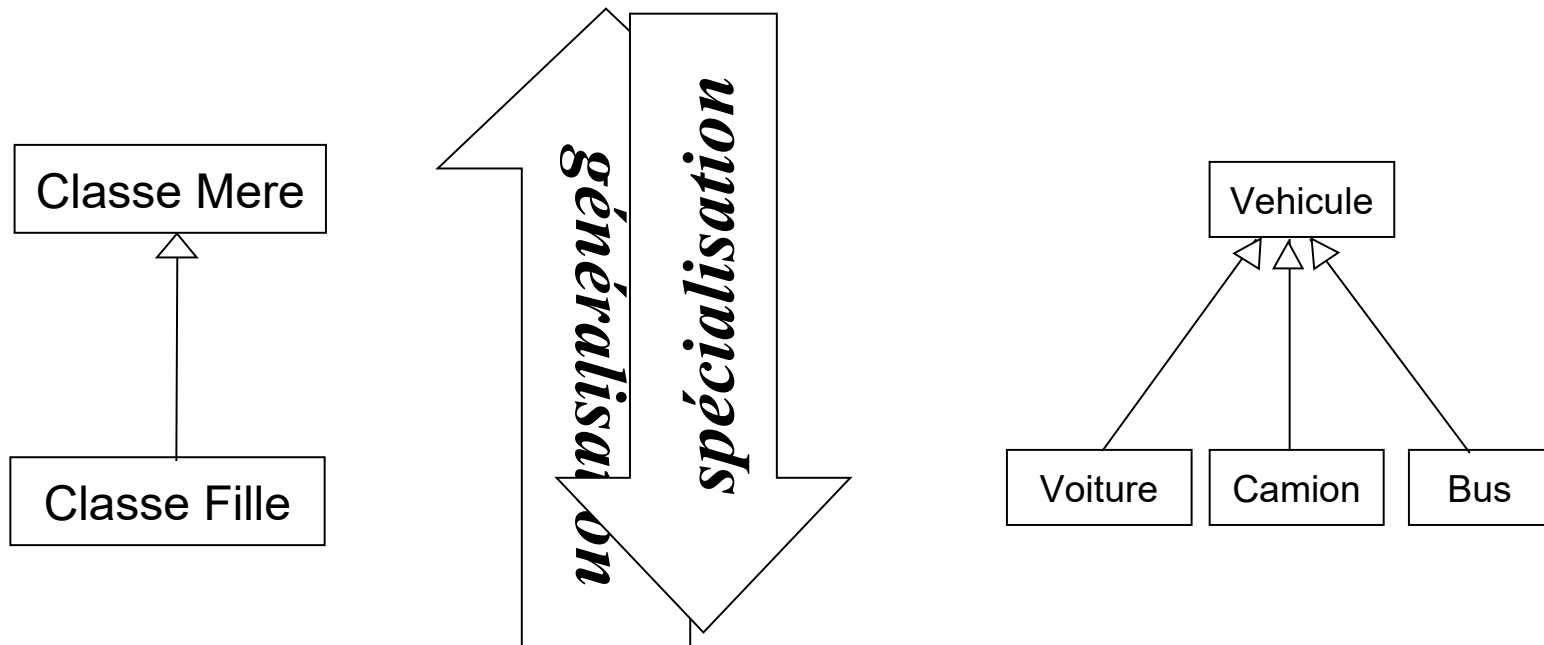




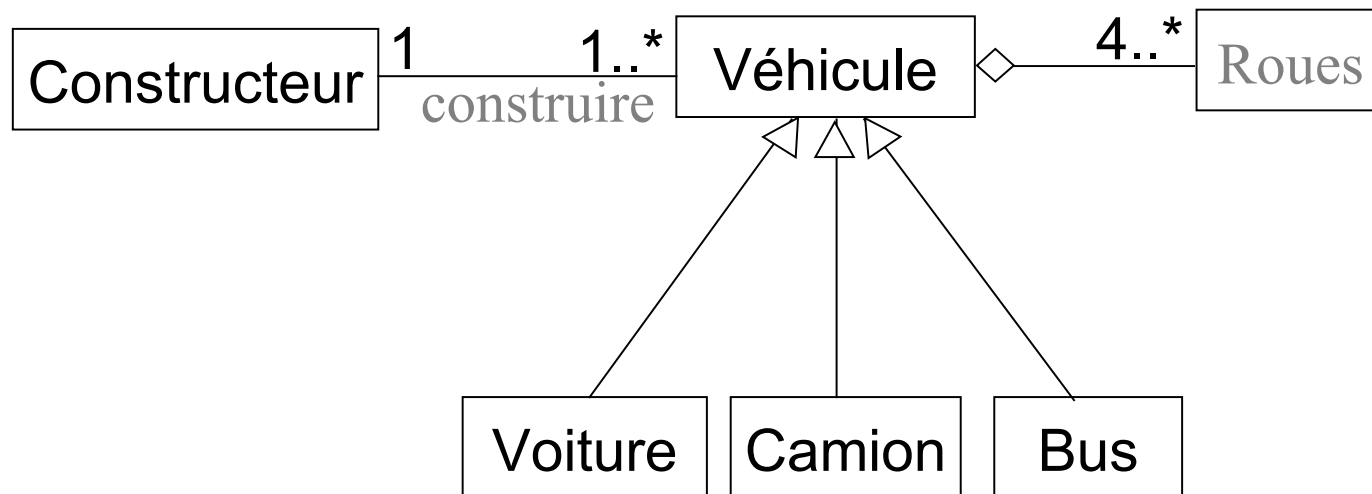
# Conception Orientée Objet

## Les relations entre classes

- Héritage (sous-classe) :
  - relation de Généralisation/Spécialisation
  - « **est un sorte de** »



# Synthèse des relations entre classes



# Exemple

Banque
nom : string adresse : string
creerAgence() fermerAgence()

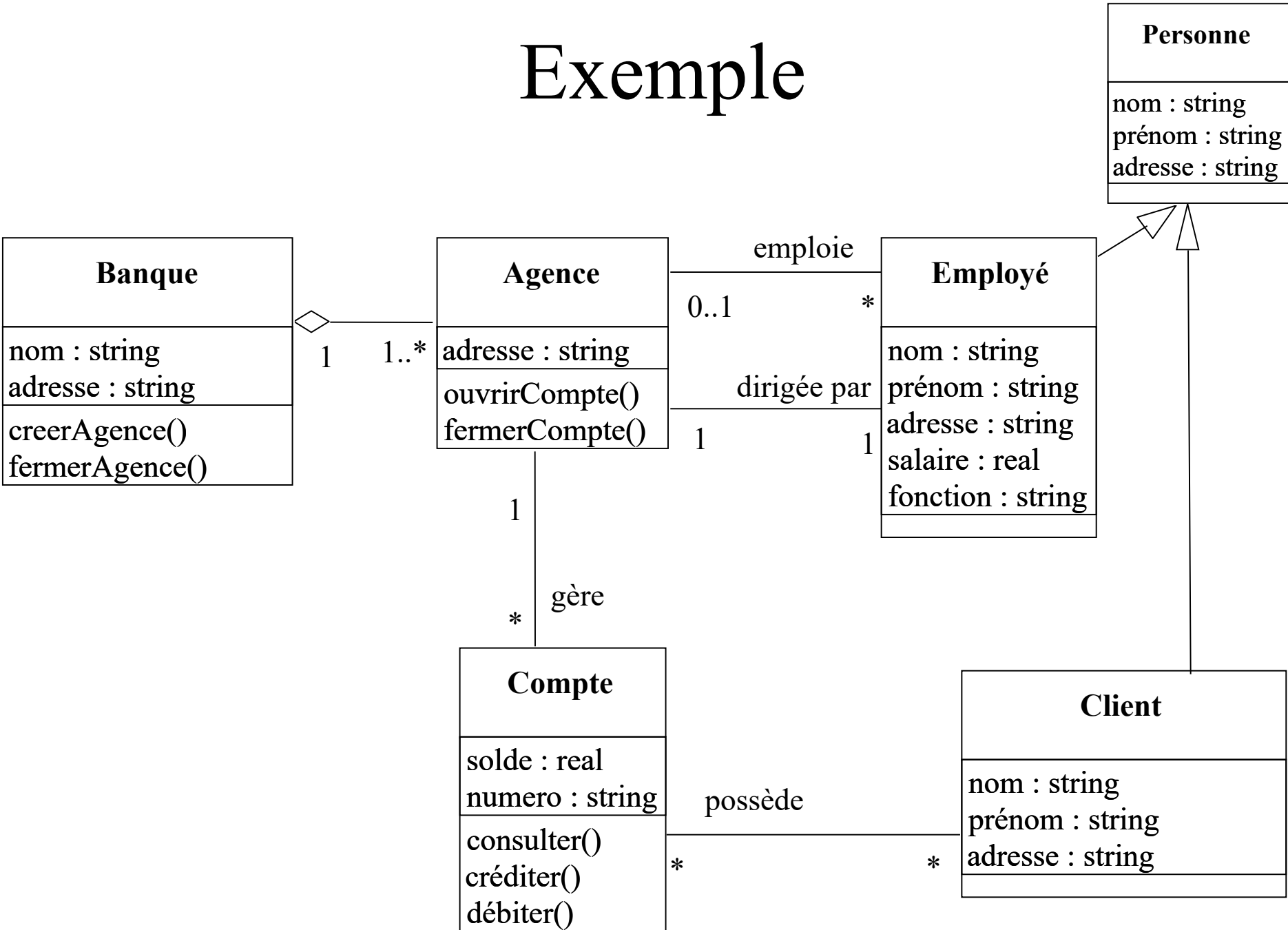
Agence
adresse : string
ouvrirCompte() fermerCompte()

Employé
nom : string prénom : string adresse : string salaire : real fonction : string

Compte
solde : real numero : string
consulter() créditer() débiter()

Client
nom : string prénom : string adresse : string

# Exemple



# Exemple

