CHAP3 : Diagramme de classes

1. Introduction

Le digramme de classe est considéré comme le plus important de la modélisation orientée objet. Il est le seul obligatoire lors d'une telle modélisation. Alors que le diagramme de cas d'utilisation montre un système du point de vue des acteurs le diagramme de classe en montre la structure interne. Il s'agit d'une vue statique car on ne tient pas compte du facteur temporel dans le comportement du système. Le diagramme de classe permet de modéliser les classes du système et leur relation indépendamment d'un langage de programmation particulier.

2. Notion de classe et d'instance de classe

Une instance est une concrétisation d'un concept abstrait. Une classe est un type abstrait. Tout système orienté objet est organisé autour des classes.

Un objet est une instance d'une classe. C'est une entité discrète dotée d'une identité, d'un état et d'un comportement que l'on peut invoquer. Les objets sont des éléments individuels d'un système en cours d'exécution.

Par exemple, si l'on considère que « *Personne* » est un concept abstrait, on peut dire que Claude-EUSEBIO est une instance de Personne. Si « *Personne* » était une classe, Claude-EUSEBIO en serait une instance : un objet.

3. Diagramme de classes: sémantique

Un diagramme de classe est une collection d'éléments de modélisation statique qui montre la structure d'un modèle. Un diagramme de classe fait abstraction des aspects dynamiques et temporels.

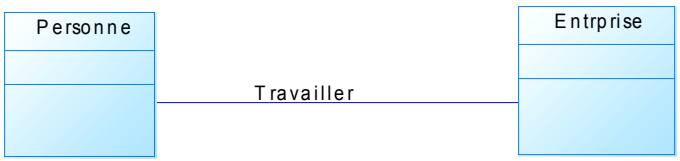
Pour un modèle complexe plusieurs diagrammes de classes complémentaires doivent être construits.
On peut par exemple se focaliser sur :

- Les classes qui participent à un cas d'utilisation ;
- Les classes associées dans les réalisations d'un scénario précis ;
- Les classes qui composent un paquetage ;
- La structure hiérarchique d'un ensemble de classes.

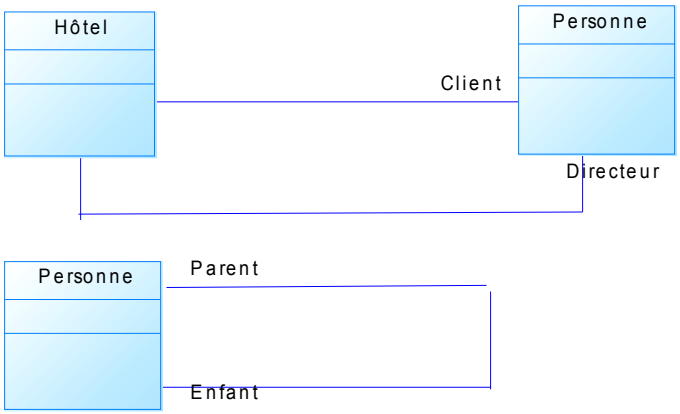
Une classe a un nom, un ou des attribut(s) et une ou des méthode(s).
Remarque: Il existe par exemple, des diagrammes de classes d'implémentation, des diagrammes de classes conceptuelles, des diagrammes de classes de spécification.

4. Associations entre classes

Une association exprime une connexion sémantique entre deux classes.



Le **rôle** spécifie la fonction d'une classe pour une association donnée (indispensable pour les associations réflexives).

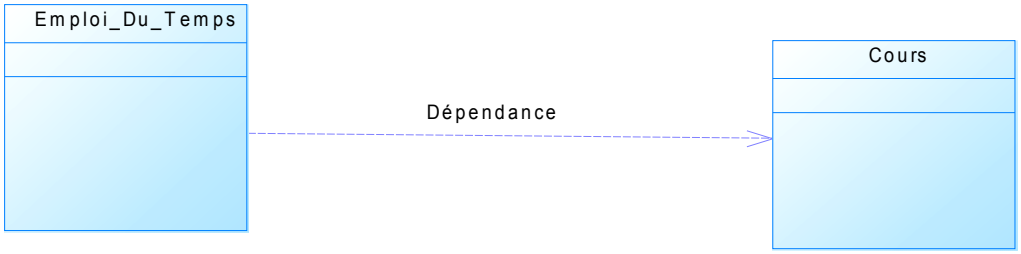


La cardinalité précise le nombre d'instance qui participe à une relation.

n : exactement « n » (n, entier naturel >0)
n..m : « n » à « m » (entier naturels, m>n)
* : Plusieurs (équivaut à « 0..n » ou « 0..*)
n..* : « n » ou plus (n,entier naturel)

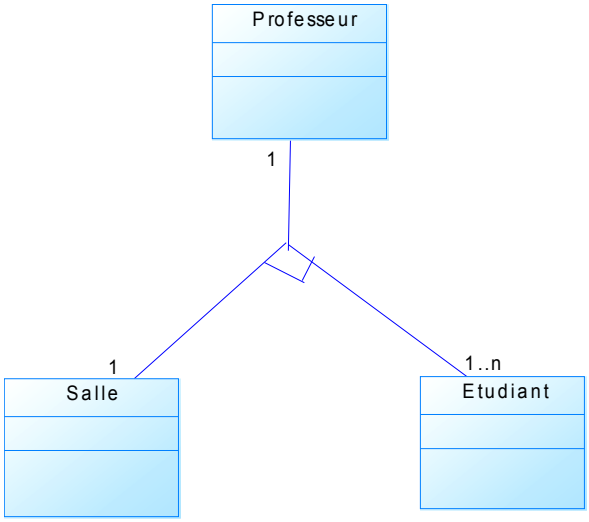
5. Relation de dépendance

C'est une relation unidirectionnelle. Une modification de l'élément dont on dépend (élément cible) peut nécessiter une mise à jour de l'élément dépendant (élément source).



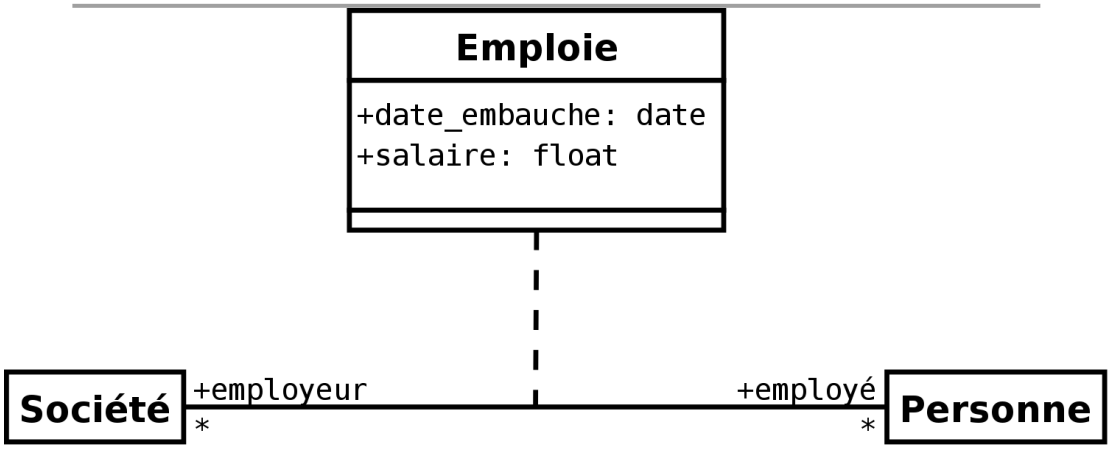
6. Association n-aire

Il s'agit d'une association qui relie plus de deux classes.



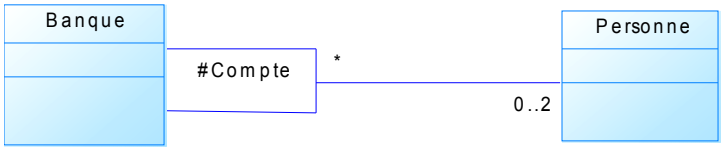
7. Classe d'association

Il s'agit d'une classe qui réalise la navigation entre d'autres instances de classes.



8. Qualification

Elle permet de sélectionner un sous ensemble d'objet parmi l'ensemble des objets qui participent à une association.

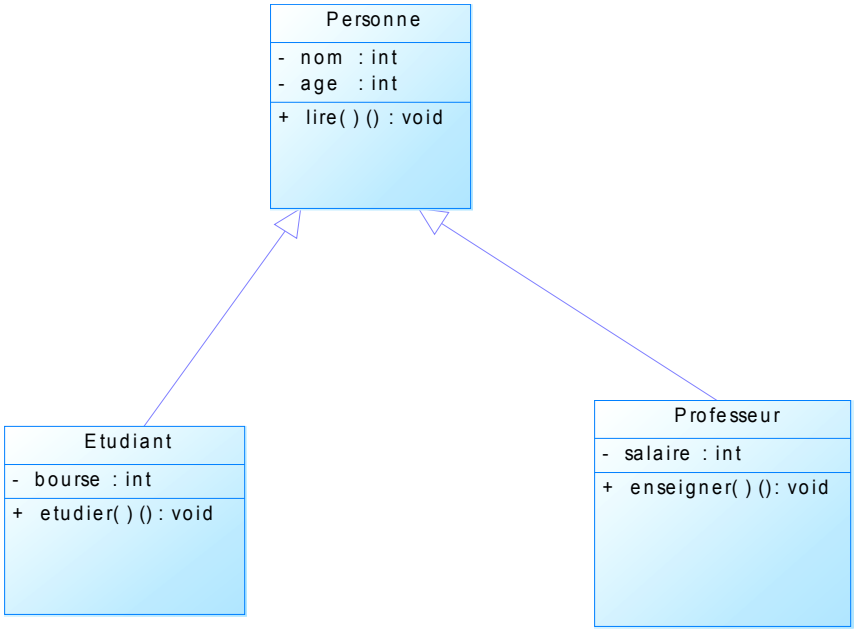


(2)

Une personne peut posséder plusieurs comptes dans plusieurs banques

Un compte dans une banque appartient à au plus deux personnes. Autrement dit une instance du compte (banque, compte) est en association au plus avec deux instances de personne.

9. Héritage



Spécialisation : Consiste à étendre les propriétés d'une classe sous forme de sous classe plus spécifique.

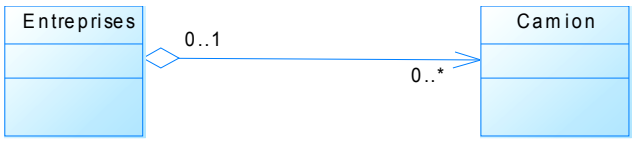
Généralisation : Consiste à factoriser les propriétés d'un ensemble de classes, sous forme de classes plus abstraite.
Il doit être possible de substituer n'importe quelle instance d'une super classe par n'importe quelle instance de sous classes sans que la sémantique d'un programme écrit dans le terme de la super classe n'en soit affectée.
Si Y hérite de X cela signifie que Y est une sorte de X.

10. Agrégation

L'agrégation est une association non symétrique qui exprime un couplage fort et une relation de subordination.

Une agrégation peut notamment exprimer :
- Qu'une classe (« un élément ») fait partie d'une autre (« l'agrégat »).
- Qu'un changement d'état d'une classe implique un changement d'état d'une autre.
- Qu'une action sur une classe entraîne une action sur une autre.

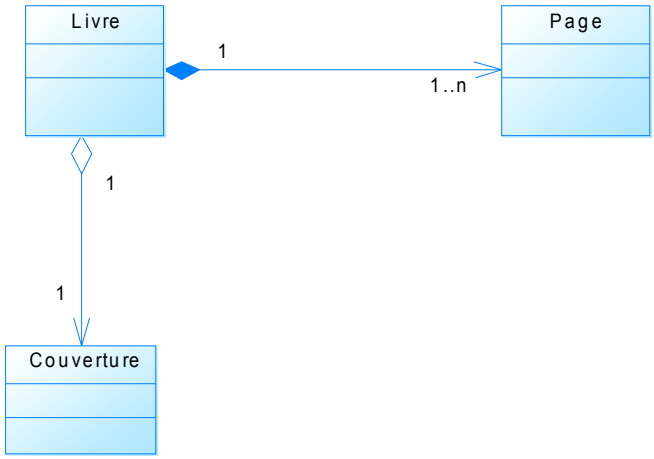
Une instance d'élément agrégé peut exister sans agrégat (et inversement).



On représente l'agrégation par un losange vide du côté de l'agrégat.

Composition: La composition est une agrégation forte.

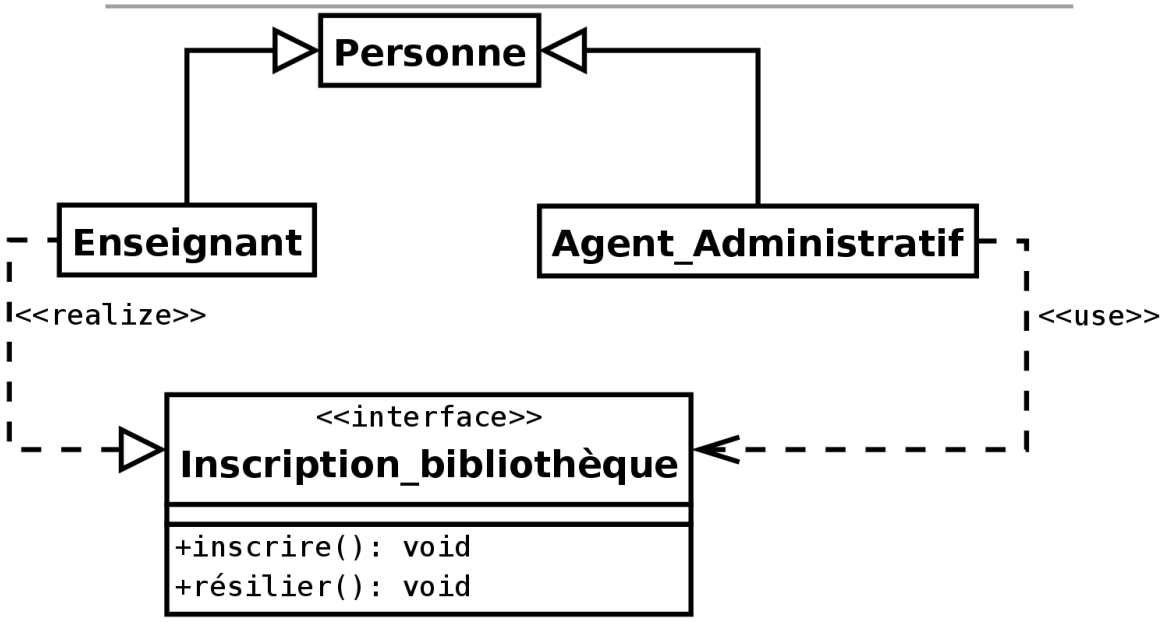
Les cycles de vie des éléments (les composants) et de l'agrégat sont liés : si l'agrégat est détruit (ou copié), ses composants le sont aussi.
A un même moment, une instance de composant ne peut être liée à un seul agrégat.



11. Encapsulation visibilité interface

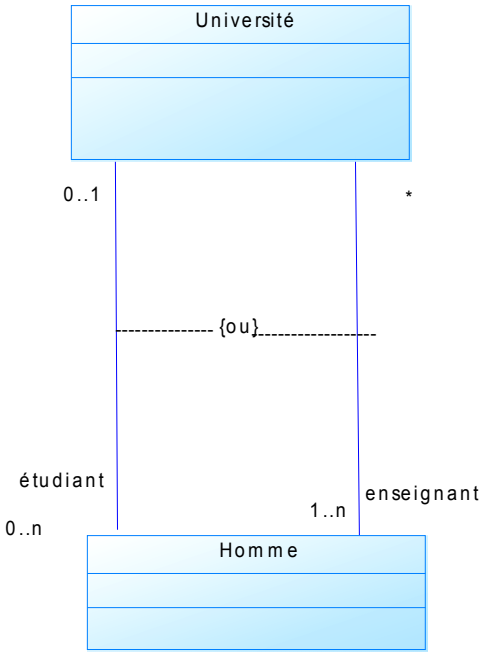
Public ou +
Private ou –
Protected ou #
Package ou ~

Les classes permettent de définir en même tant un objet et son interface. Une interface fournit une vue totale ou partielle d'un ensemble de services offert par une classe, un paquetage ou un composant.



12. Contraintes sur une association

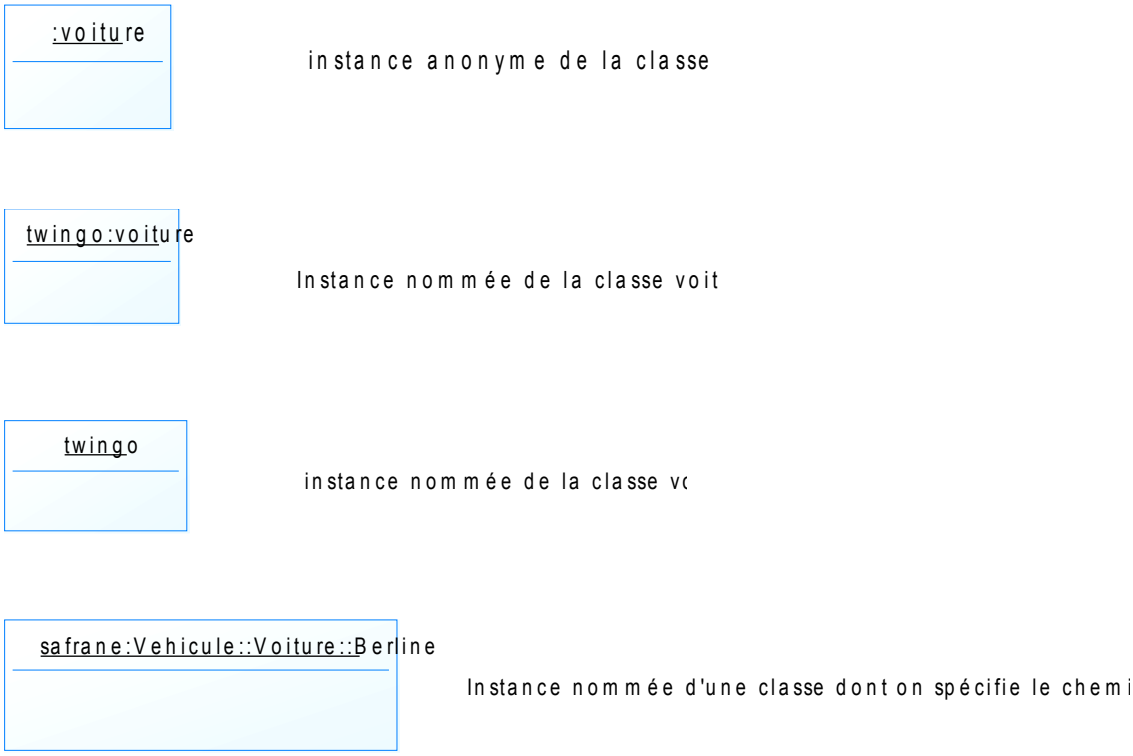
Les contraintes sont des expressions qui précisent le rôle ou la portée d'un élément de modélisation (elles permettent d'étendre ou de préciser sa sémantique).



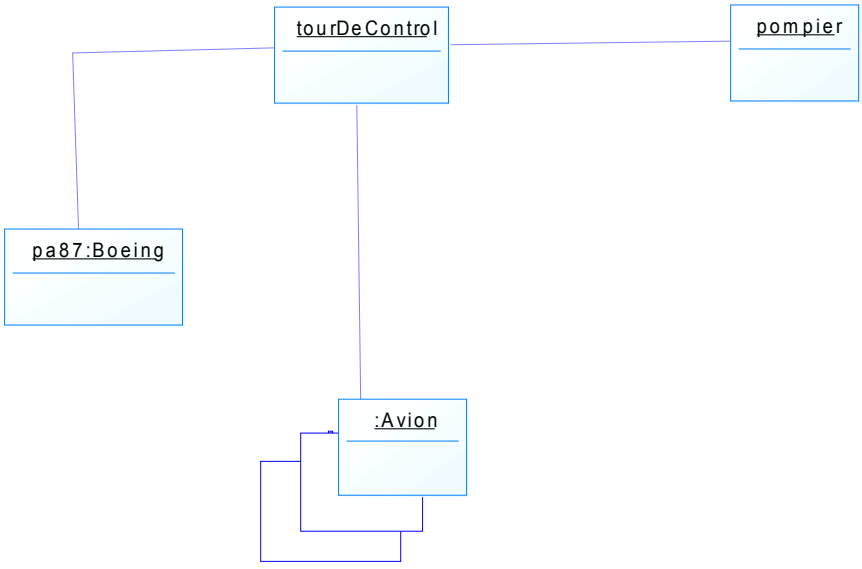
13. Les Objets

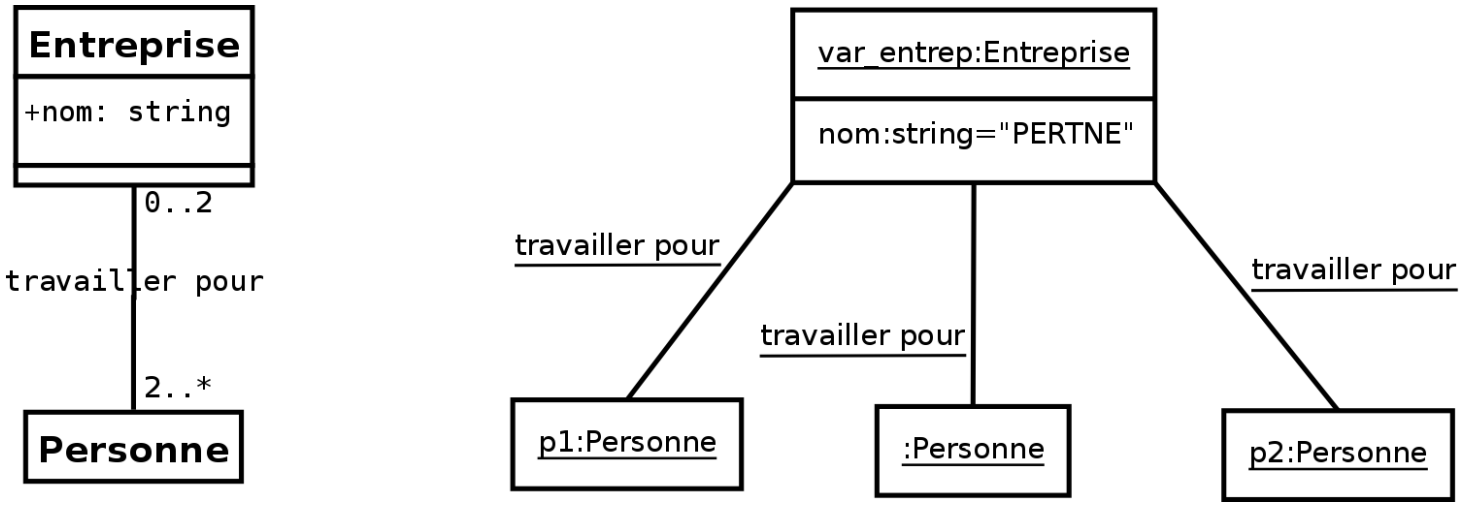
Ce type de diagramme montre des objets (instances de classe dans un état particulier) et des liens (relations sémantiques) entre ces objets.

Les diagrammes d'objets s'utilisent pour montrer un contexte. Ce type de digramme sert essentiellement en phase exploratoire.



Exemples de diagramme d'objets





14. Diagramme de collaboration

Une collaboration montre des instances qui collaborent dans un contexte donné pour mettre en œuvre une fonctionnalité d'un système.

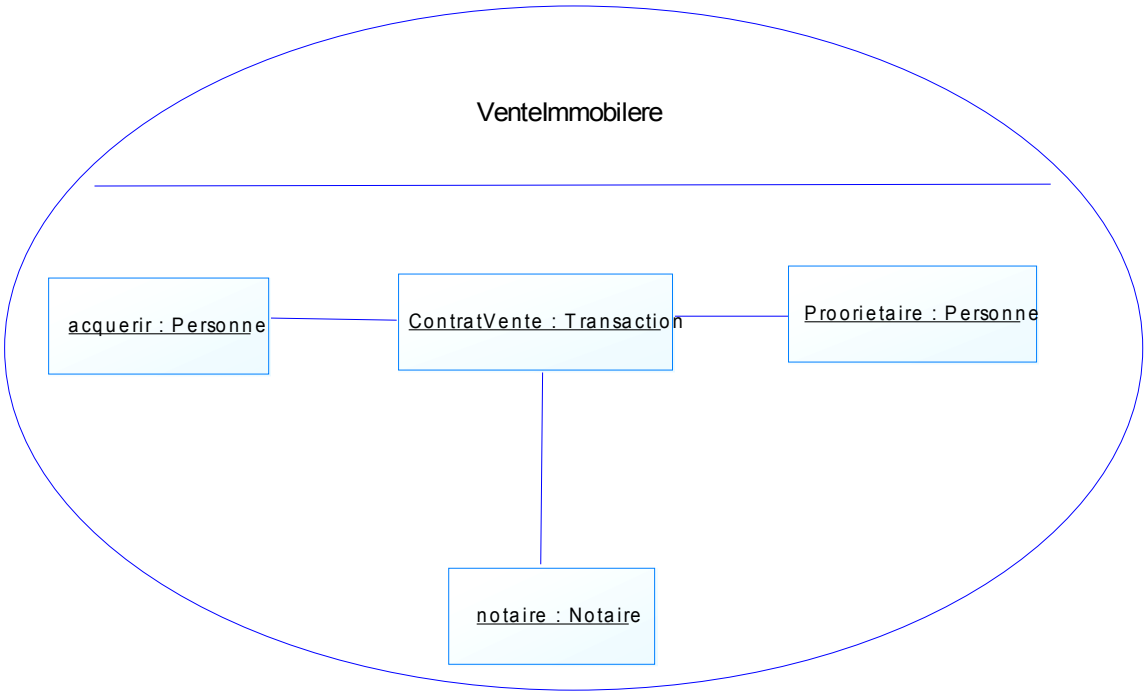


Diagramme de collaboration d'une transaction immobilière

15. Paquetages

Les packages sont des éléments d'organisation d'un modèle. Ils regroupent des éléments de modélisation selon des critères purement logiques. Ils possèdent une interface. Ils servent de brique de base dans la constriction d'une architecture. Ils représentent le bon niveau de granularité pour la réutilisation. Les packages peuvent aussi être des espaces de nom.