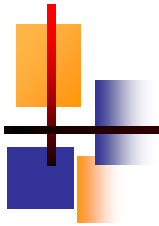


Génie Logiciel (MLI532) Relations entre classes concepts avancés

Lamine BOUGUEROUA
Lamine.bougueroua@esigetel.fr

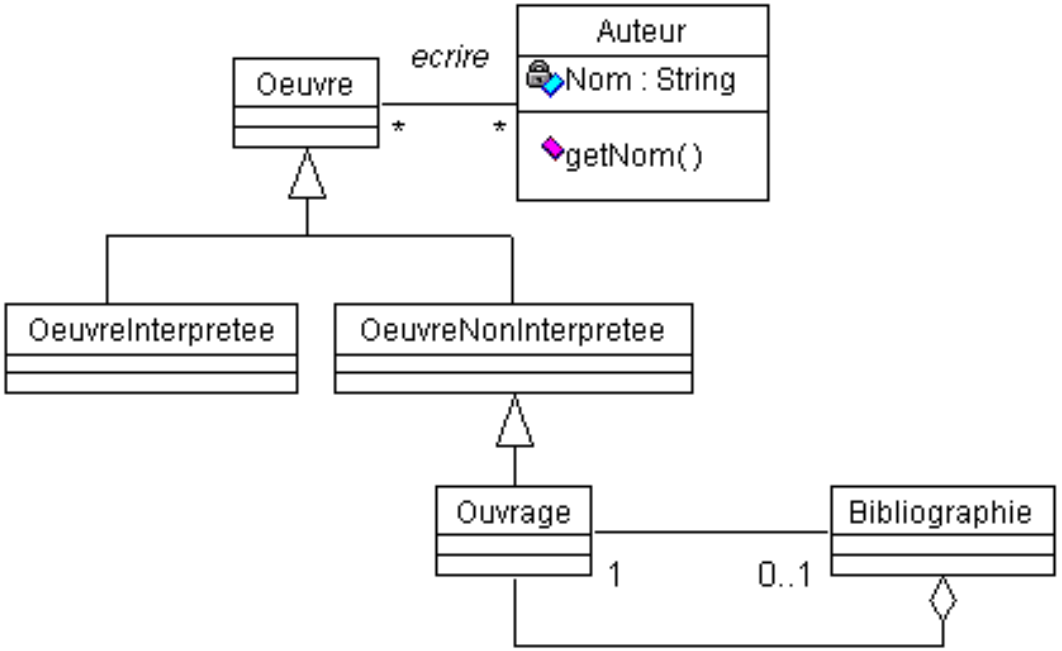
Diagramme de classes

Diagramme de classes

- Décrit le système
- Montre les classes et la façon dont elles sont associées
- une vision statique et structurelle

Dans un diagramme ont trouve des classes et des relations (des associations) entre ces classes :

- association simple,
- généralisation (héritage)
- dépendance,
- agrégation,
- composition
- Implémentation (interface).



Visibilité des propriétés

aux éléments d'un modèle

Contrôler et éviter les dépendances entre paquetages

+ public visible

protégé visible dans la classe et ses sous-classes

- privé visible dans la classe uniquement

N'a pas de sens dans un modèle abstrait

0

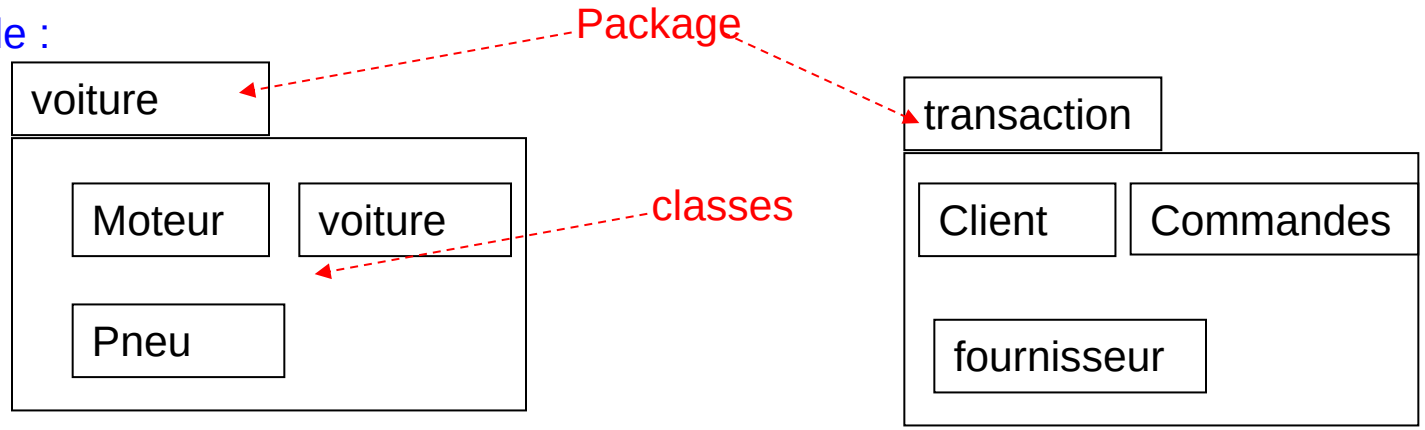
)

0

Visibilité - exemple

- Visibilités - JAVA :

Exemple :



Modificateur classe A	Même package		Autre package	
	Classe fille	Autre classe	Classe fille B	Autre classe
Public	✓	✓	✓	✓
Protected	✓	✓	✓ une instance de B invoquée dans B ou dans une sous-classe de B	
Friend	✓	✓		
private				

Propriétés & méthodes

Les propriétés :

[*visibilité*] *nom* [*card*] [**:** *type*] [**=** *valeur-initiale*] [{ *props*... }]

exemples:

age

- age : Integer = 0

age [0..1] : Integer

numsecu : Integer

motsClés [*] : String {addOnly}

[*visibilité*] *nom* [**(** *params*)] [**:** *type*] [{ *props*... }]

params := [*in* | *out*| *inout*] *nom* [**:** type] [**=**defaut] [{ *props*... }]

exemples:

getAge()

+ getAge() : Integer

- setAge(in date : Date) : Boolean

+ getAge() : Integer {isQuery}

Propriétés associées

Les propriétés suivantes peuvent être associées aux :

1. {subsets <property-name>},
2. {union},
3. {redefines <end-name>},
4. {ordered},
5. {bag}
6. {sequence} (équivalent à {seq}).

Propriétés associées

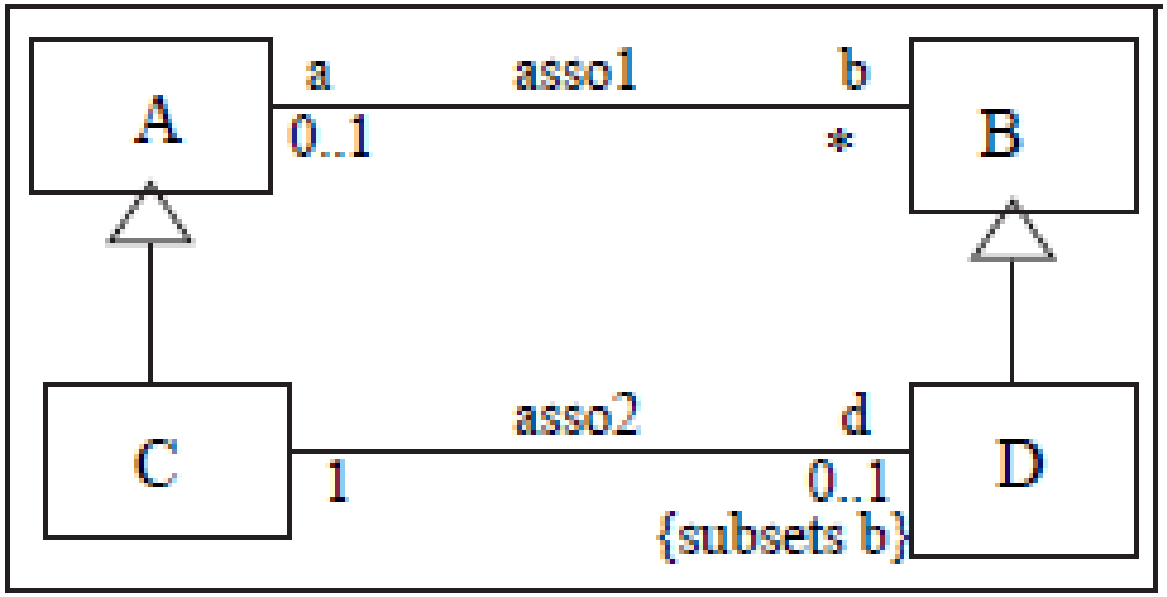
Les propriétés suivantes peuvent être associées aux :

1. {subsets <property-name>},
2. {union},
3. {redefines <end-name>},
4. {ordered},
5. {bag}
6. {sequence} (équivalent à {seq}).

Propriétés associées

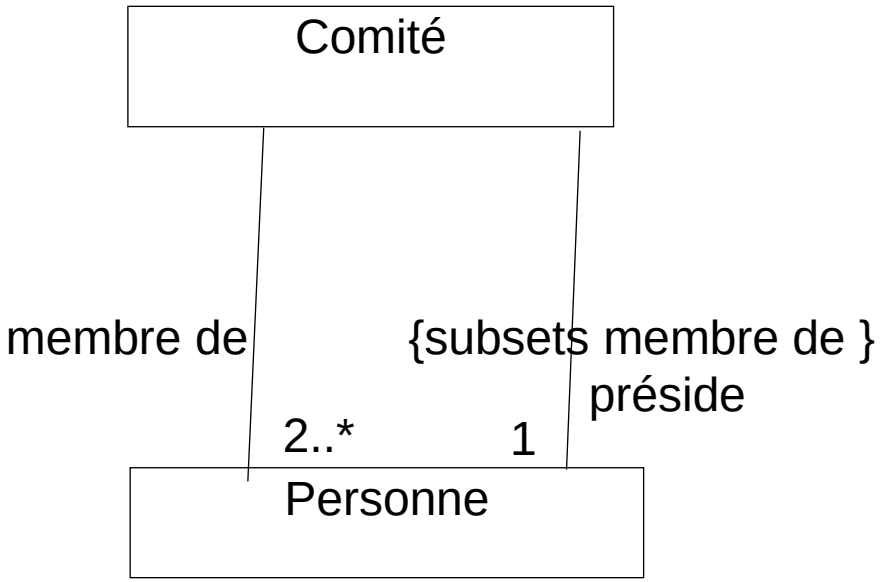
{subsets <property-name>} :

La propriété {subsets <property-name>} associée à une indique que décrit un sous-ensemble de décrit par une association plus générale.



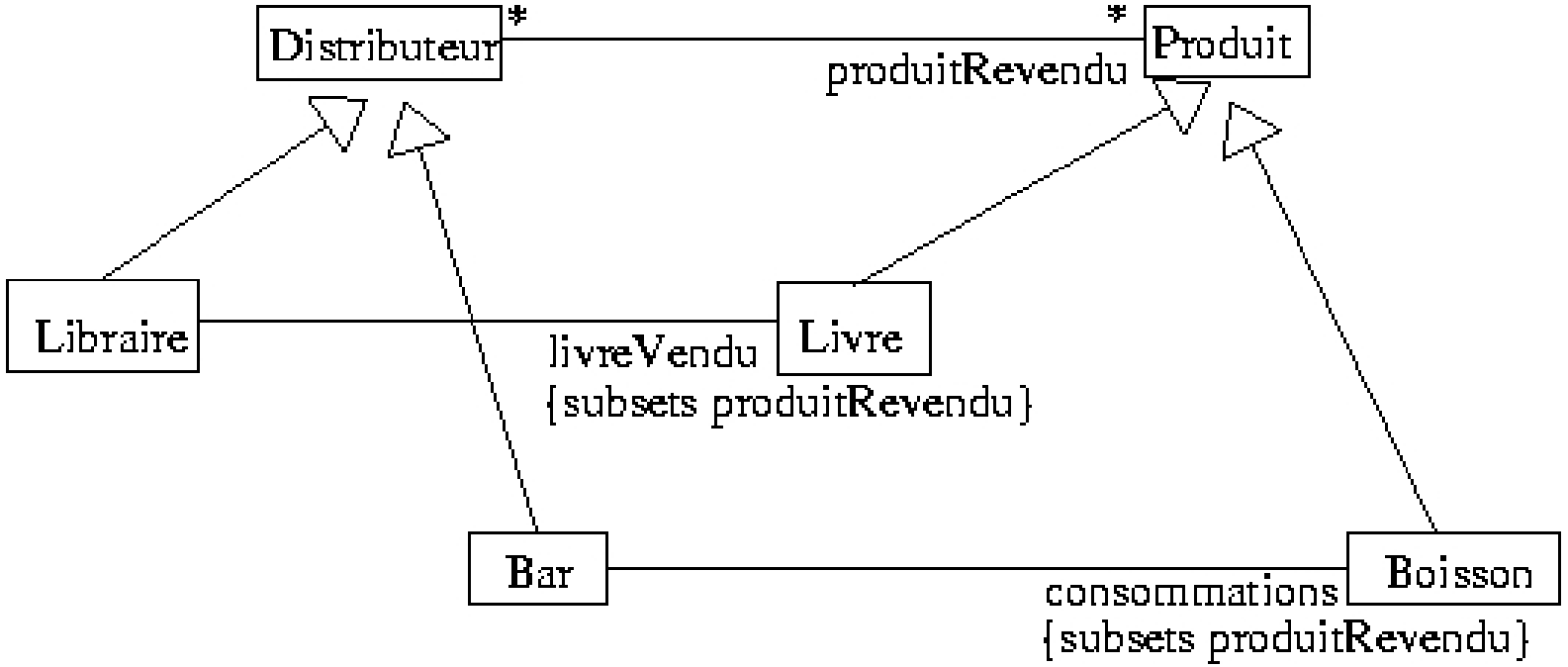
Propriétés associées

{subsets <property-name>} :



Propriétés associées

Exemple *subsets* :
un distributeur de produits qui se spécialise en Libraire (vendant des livres) et Bar (vendant des boissons).

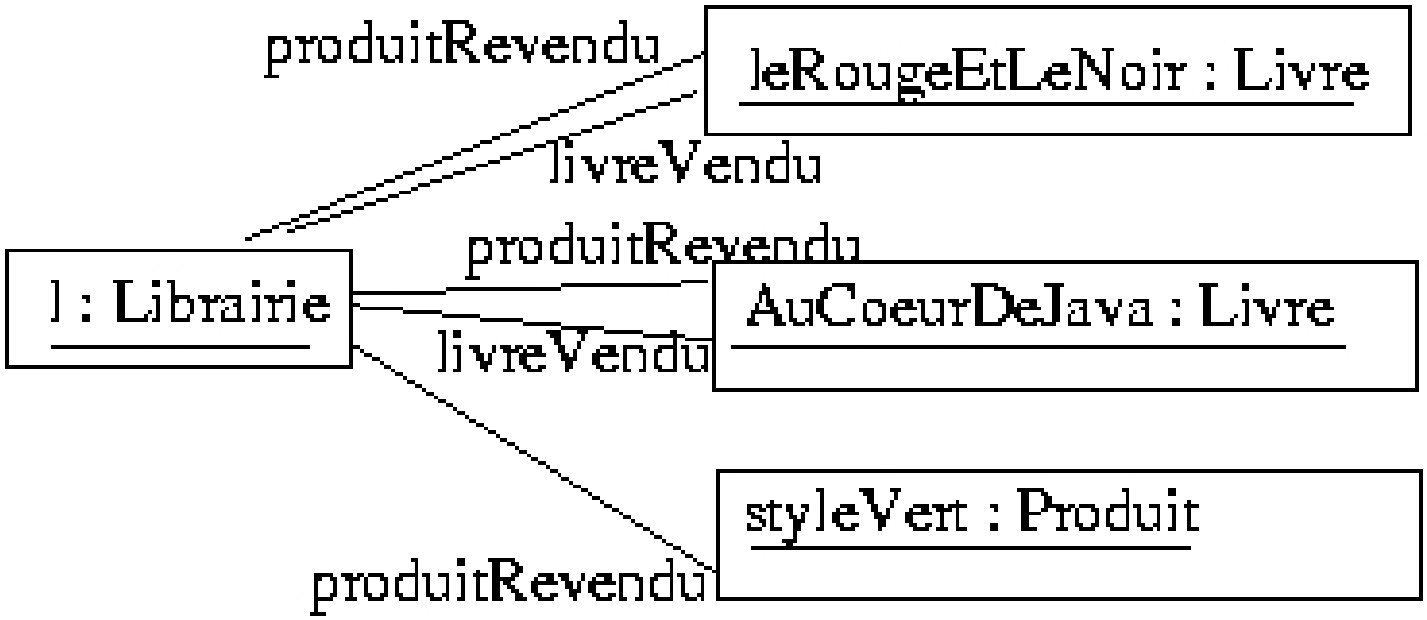


La mention subsets sur les extrémités d'associations a comme conséquence que toute présence d'un lien livreVendu entraîne la présence d'un lien produitRevendu

Propriétés associées

Exemple *subsets* :

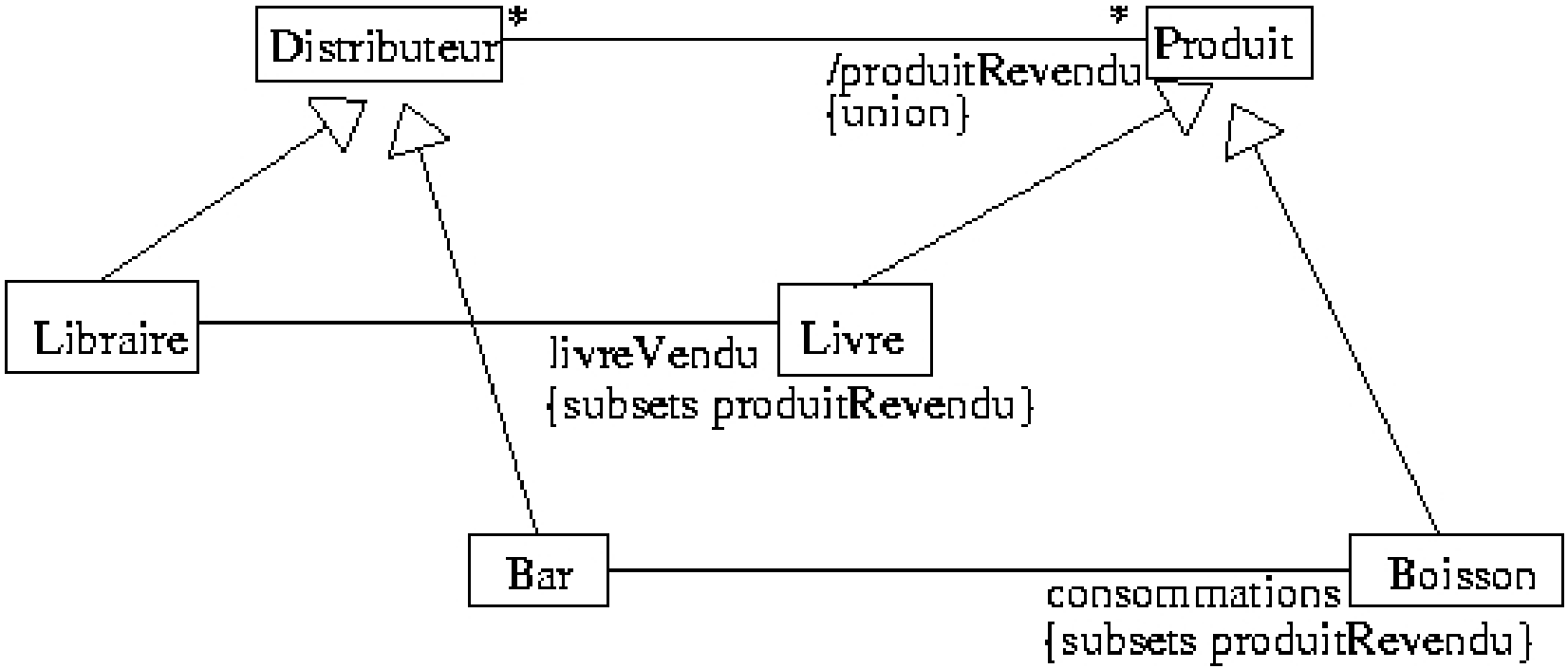
On peut trouver cependant d'autres liens produitRevendu comme vers l'instance de stylo



Propriétés associées

{union} :

La propriété {union} spécifie décrit
calculé en -ensemble

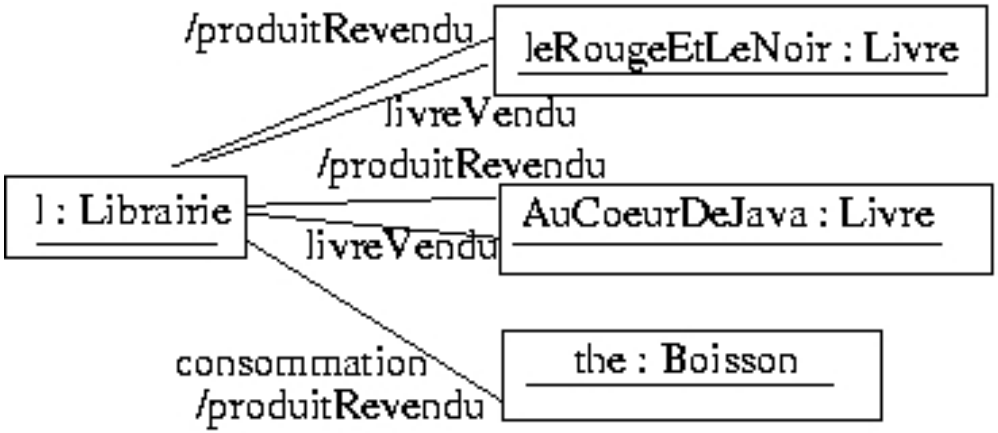


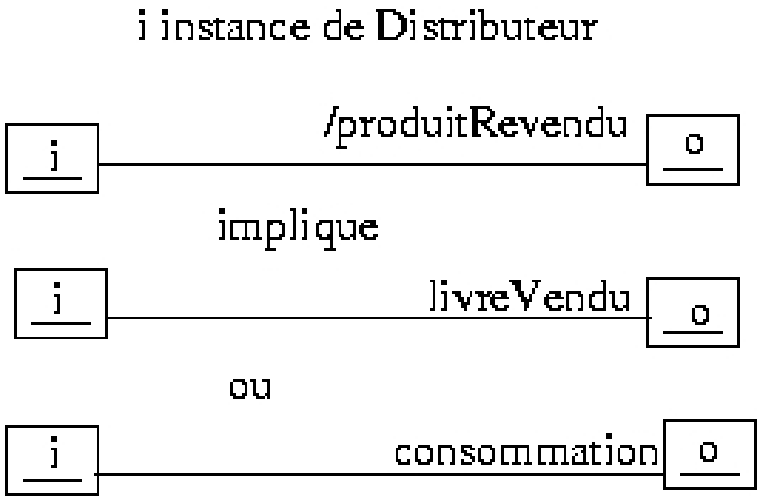
le libraire ne peut plus vendre de stylo grâce à cette restriction

Propriétés associées

Exemple *{union}* :

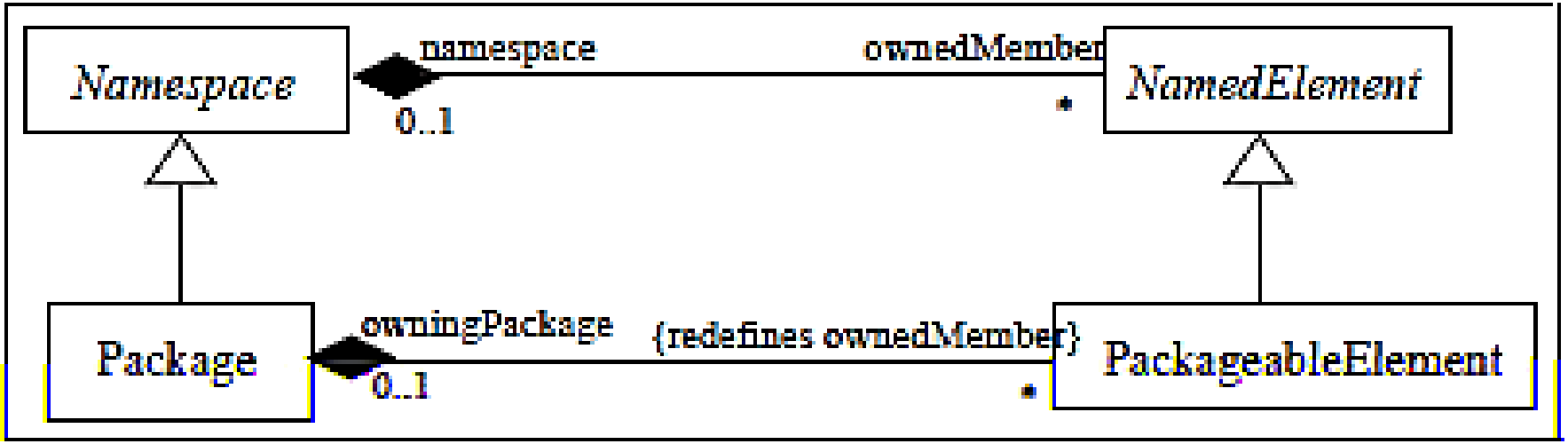
Les produits revendus se déduisent de l'union des livres et des boissons vendues Un distributeur à la fois instance de Libraire et de Bar peut vendre des livres et des boissons (mais pas autre chose).





Propriétés associées

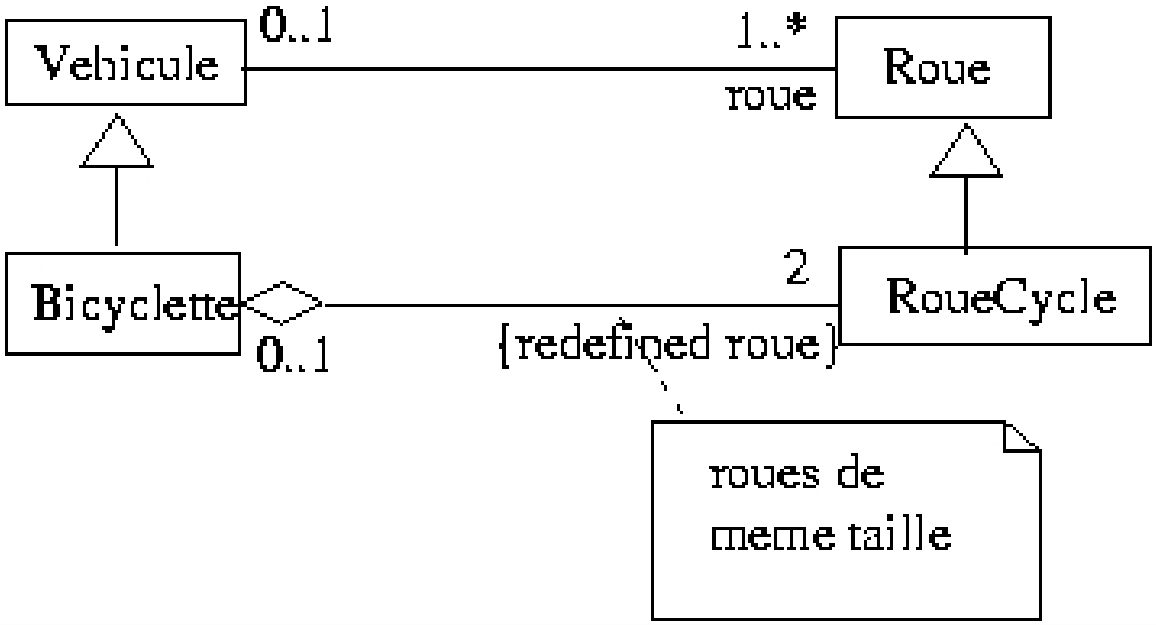
{redefines <end-name>} :
associée à une décrit plus générale.
redéfinit
des classificateurs décrit par une association



Propriétés associées

Exemple *{redefines <end-name>}* :

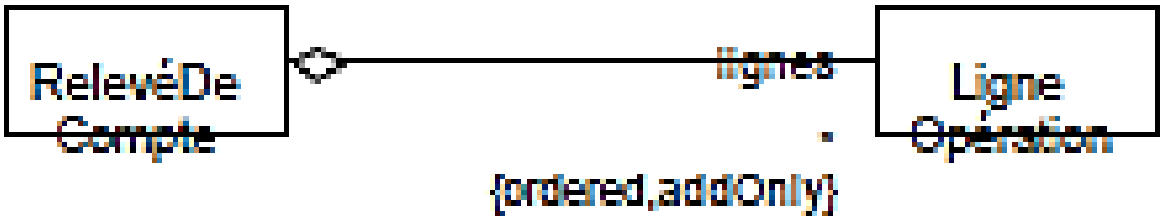
Lorsqu'on ajoute *redefined<en>* près d'une extrémité, cela signifie que l'extrémité est une spécialisation de *en*. Une redéfinition valide vérifie :
- le nouveau type est un sous-type de celui de la propriété redéfinie
- la nouvelle multiplicité est incluse dans l'ancienne
- si la propriété redéfinie est dérivée, la redéfinition l'est aussi.

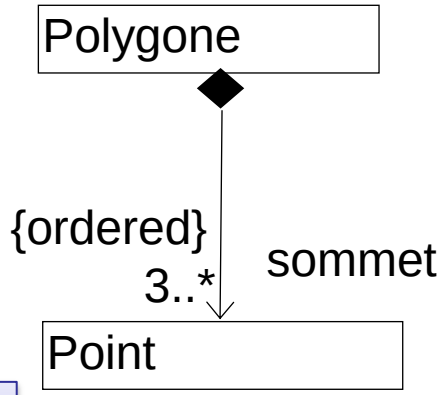


Lors d'une redéfinition, on ne change pas le nom, contrairement au cas de subset

Propriétés associées

{ordered} :
La propriété {ordered} spécifie être établi entre les instances en relation.
{ ordered } => les éléments de la collection sont ordonnés





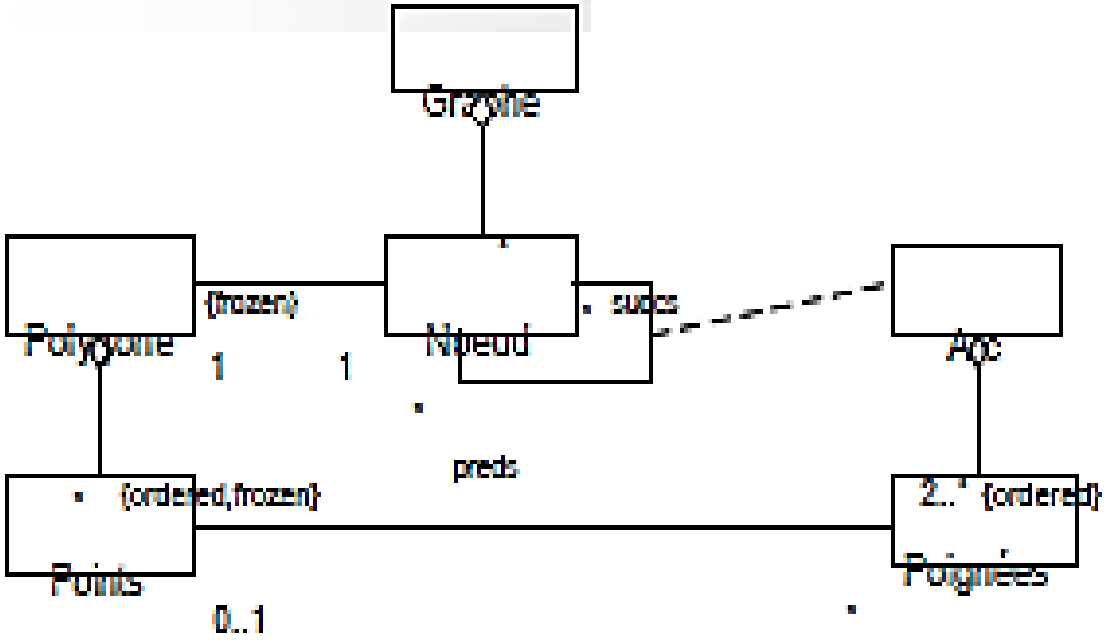
REMARQUE :
Par défaut un ensemble spécifie
ordonné et ne contient pas de doublon.

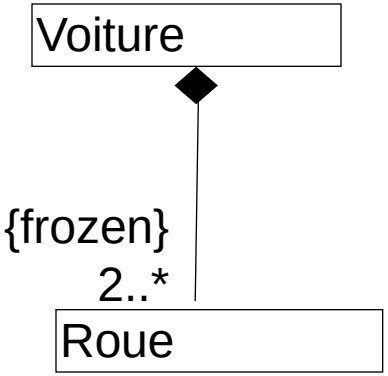
Propriétés associées

Autres contraintes :

frozen } : fixé lors de la création de , ne peut pas changer

{ addOnly} : impossible de supprimer un élément





Propriétés associées

{bag} :

spécifie décrit une collection qui permet a un ordonnée.

non ordonnée qui peut contenir deux fois le même élément

Propriétés associées

{seq} :

Les propriétés {sequence} et {seq} spécifient décrit par la fin une . -à-dire un ensemble ordonné où les doublons peuvent apparaître.

Equivalences

« modélisations équivalentes »

Equivalences

Il existe parfois plusieurs façons de modéliser une relation entre classes

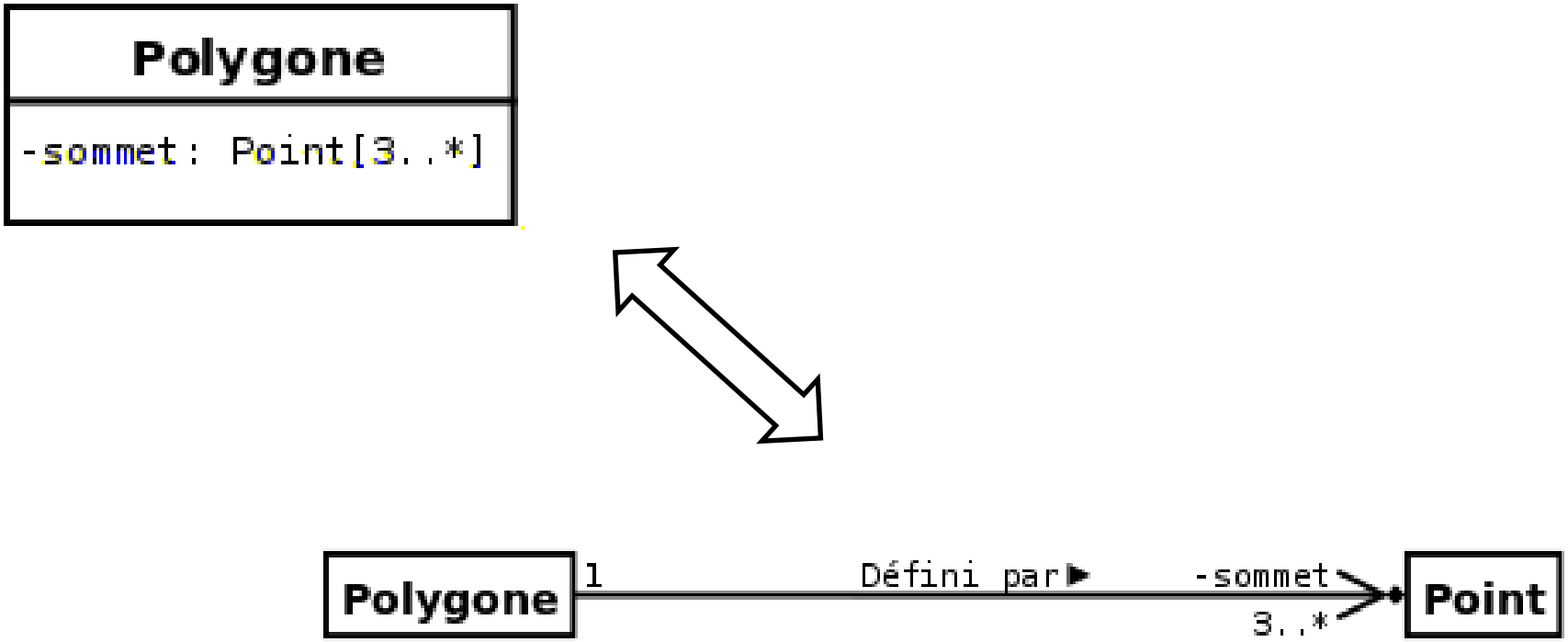


Diagramme de classes

Association qualifiée :

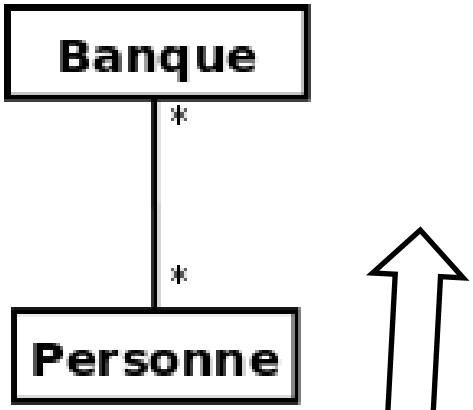
association

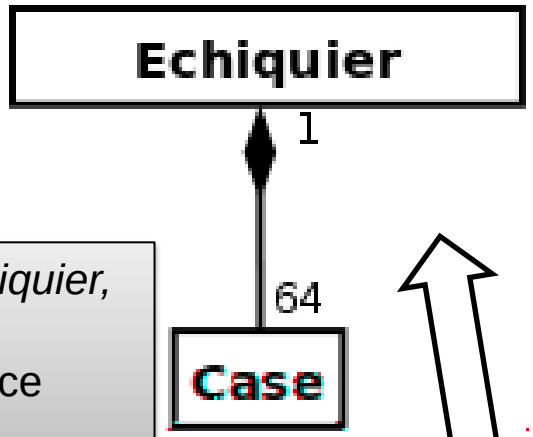
Un ***qualificateur*** est un attribut ou un ensemble d'attributs dont la valeur sert à déterminer l'ensemble des instances associées à une instance via une association

Les qualificateurs sont des attributs de

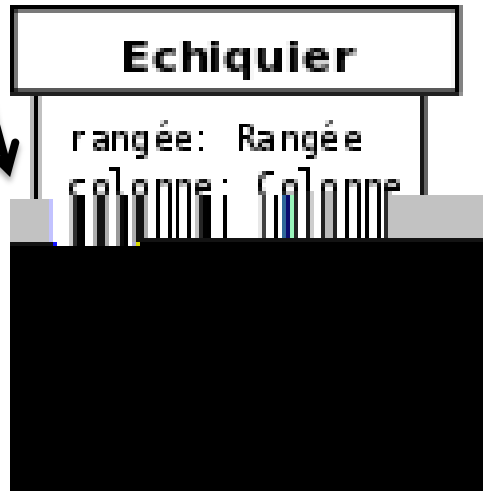
Diagramme de classes

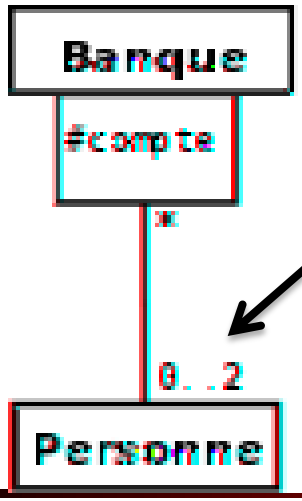
- Association qualifiée :





Une instance du triplet {*Echiquier, rangée, colonne*} est en association avec une instance unique de la classe *Case*





Un compte dans une banque appartient à au plus deux personnes

Diagramme de classes

Classe association :

Une classe-association possède les propriétés des associations et des classes : elle se connecte à deux ou plusieurs classes et possède également des attributs et des opérations

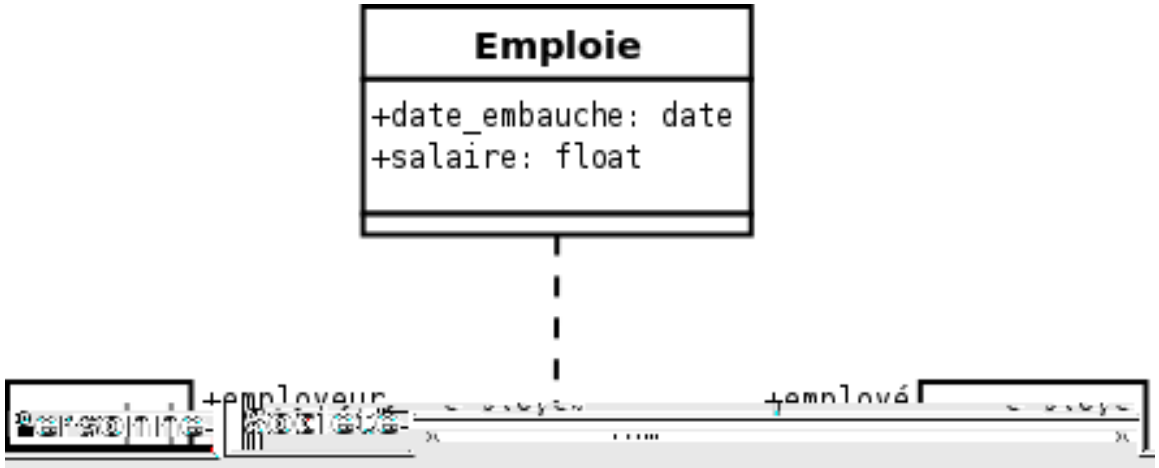
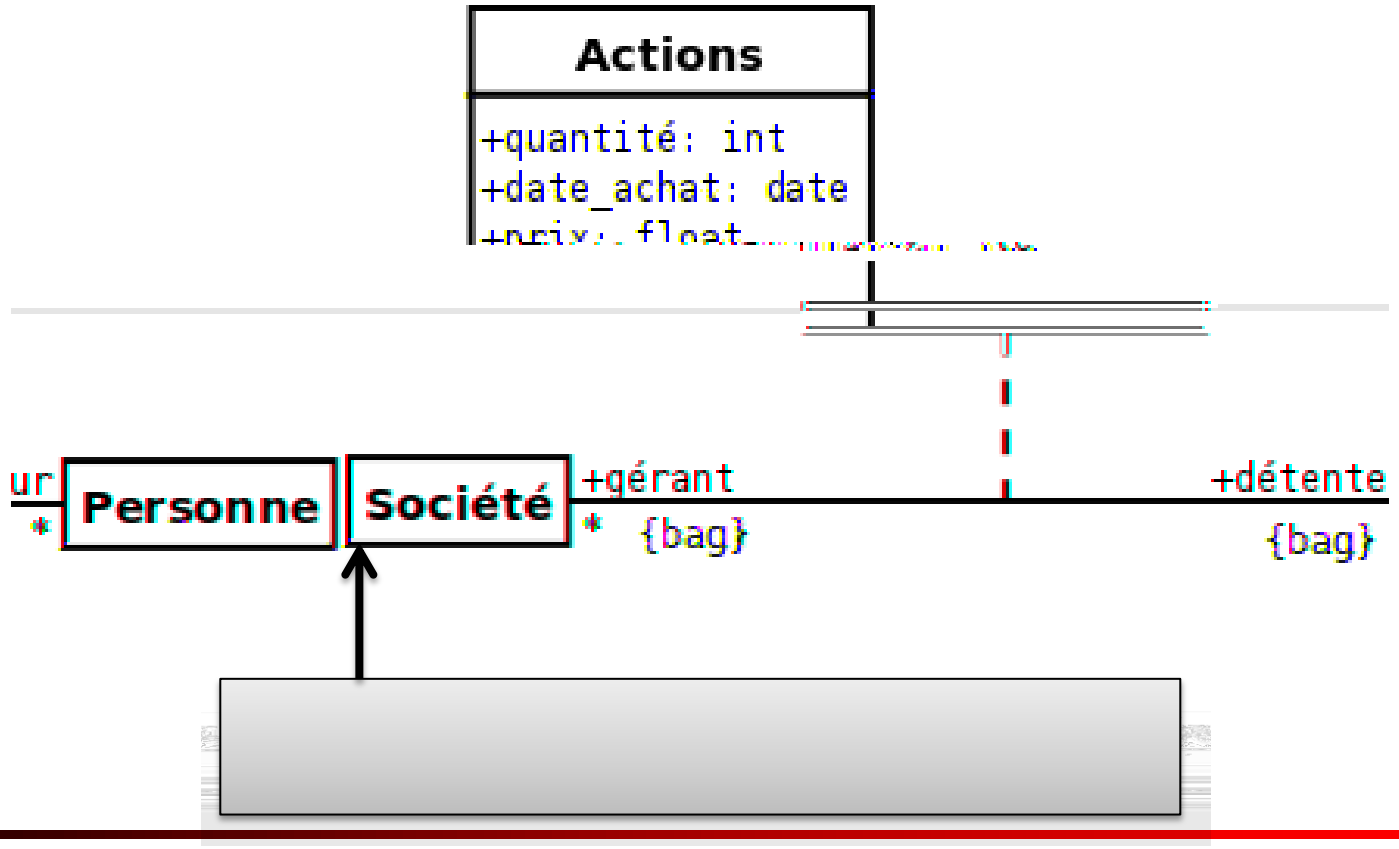


Diagramme de classes

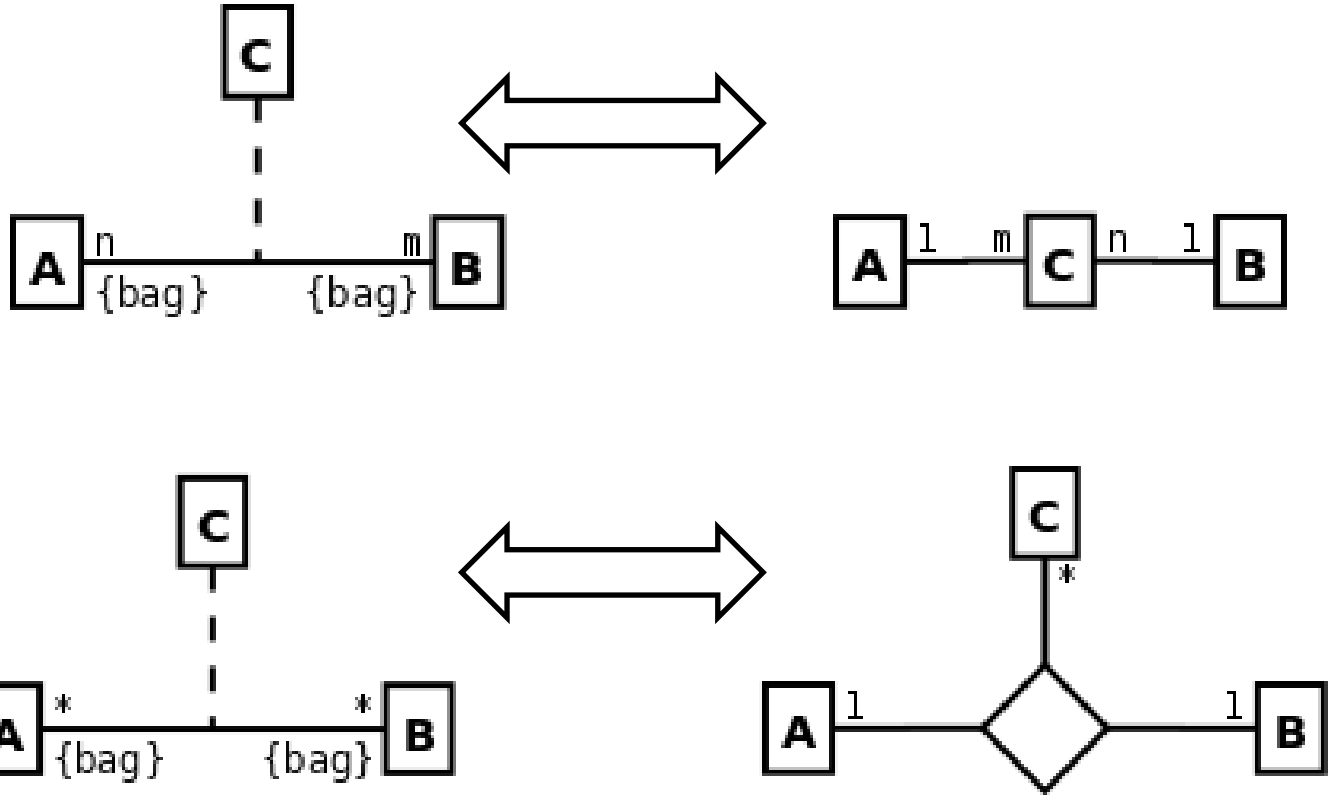
Classe association - exemple:

Une même personne peut acheter à des moments différents des actions



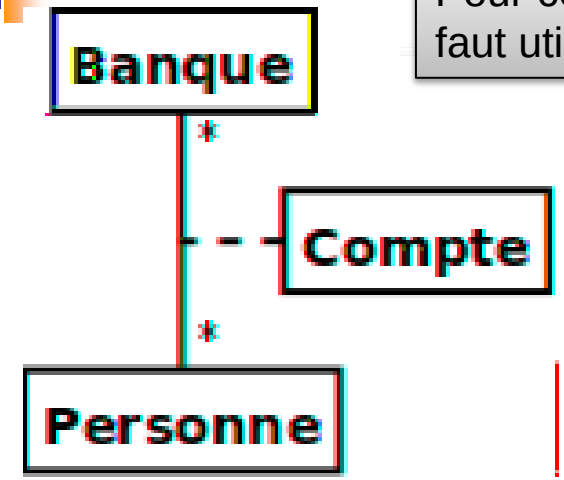
Equivalences

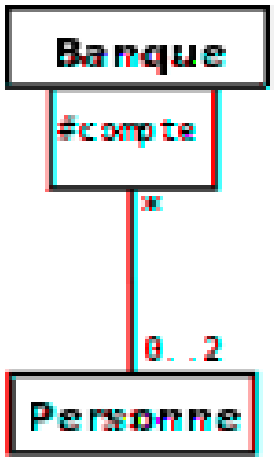
Il existe parfois plusieurs façons de modéliser une relation entre classes

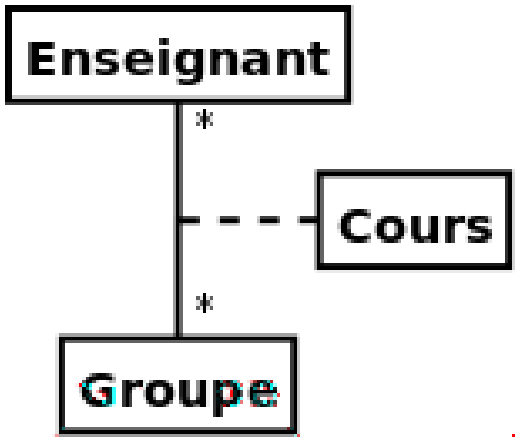


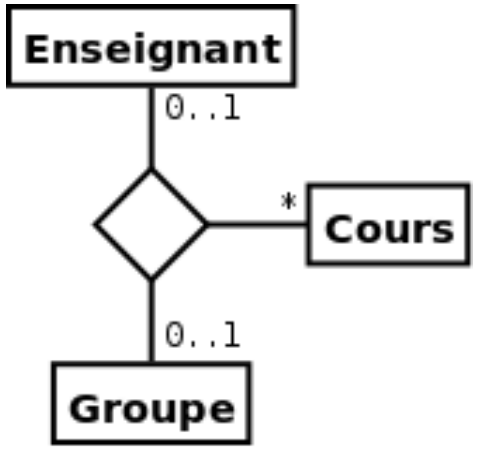
Equivalences

Pour couvrir le cas des comptes joints, il faut utiliser le modèle de droite





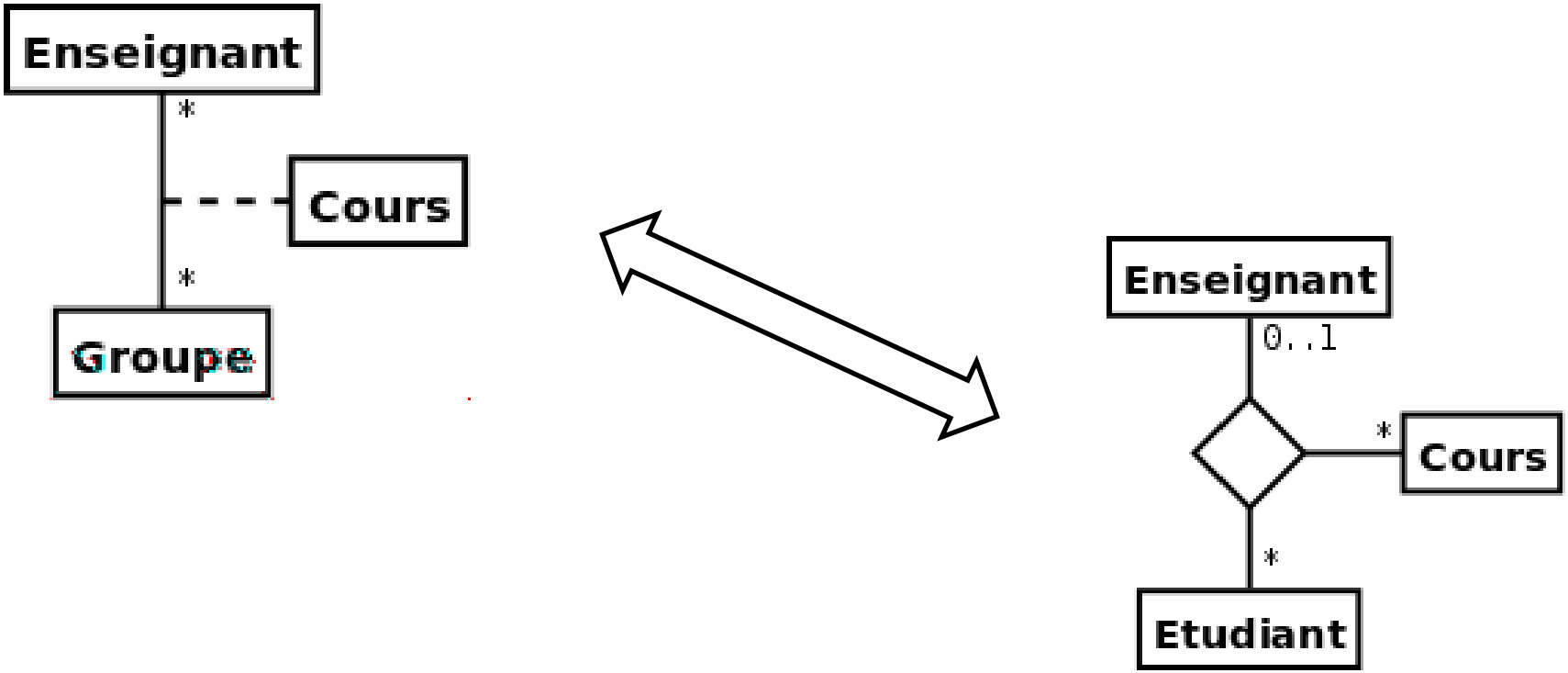




enseignant et un groupe, il faut utiliser le modèle de droite.

Equivalences

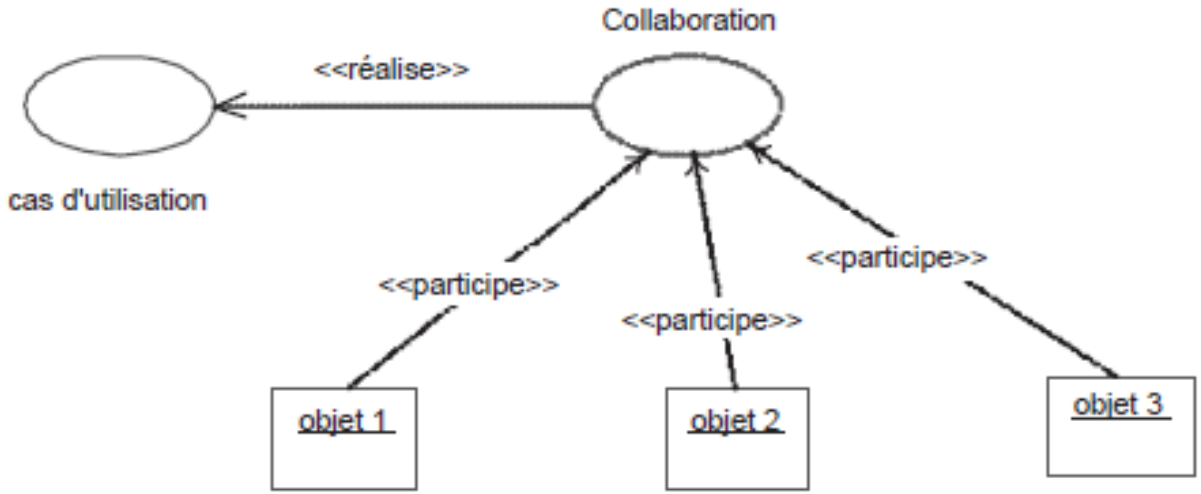
Si un même cours doit concerner plusieurs couples *Enseignant*/*Etudiant*, il ne faut pas utiliser une classe-association, mais une association ternaire comme sur le modèle de droite



Diagrammes de collaborations

Diagramme d'objet avec messages

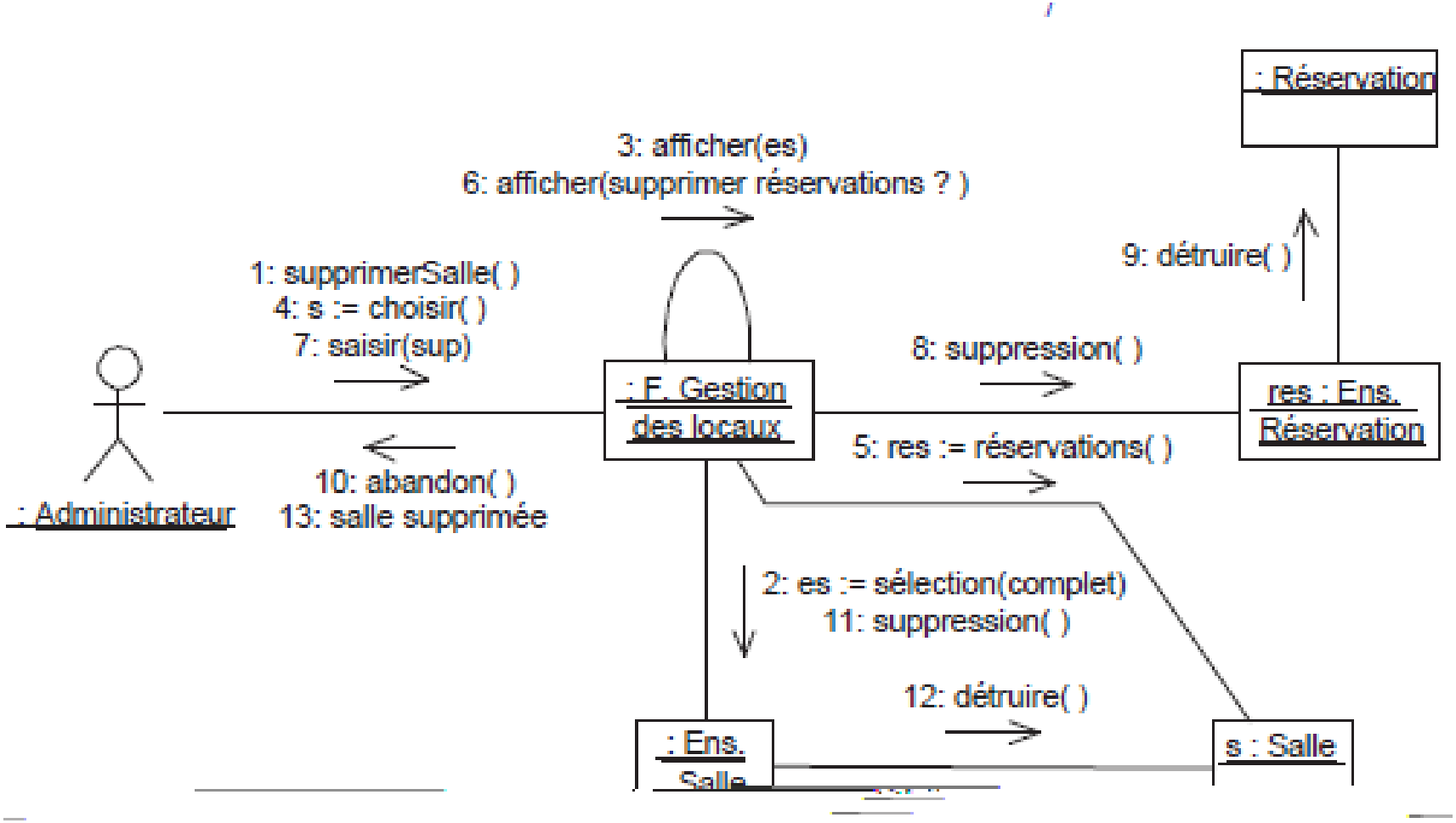
- objets et liens, multi-objets, objet actif, instance
- envoi de message (numérotation hiérarchique, paramètres...)
- structures de contrôle, synchronisation



Les diagrammes de collaboration montrent des interactions entre objets (instances de classes et acteurs).

Diagrammes de collaborations

Diagramme de collaborations - Exemple



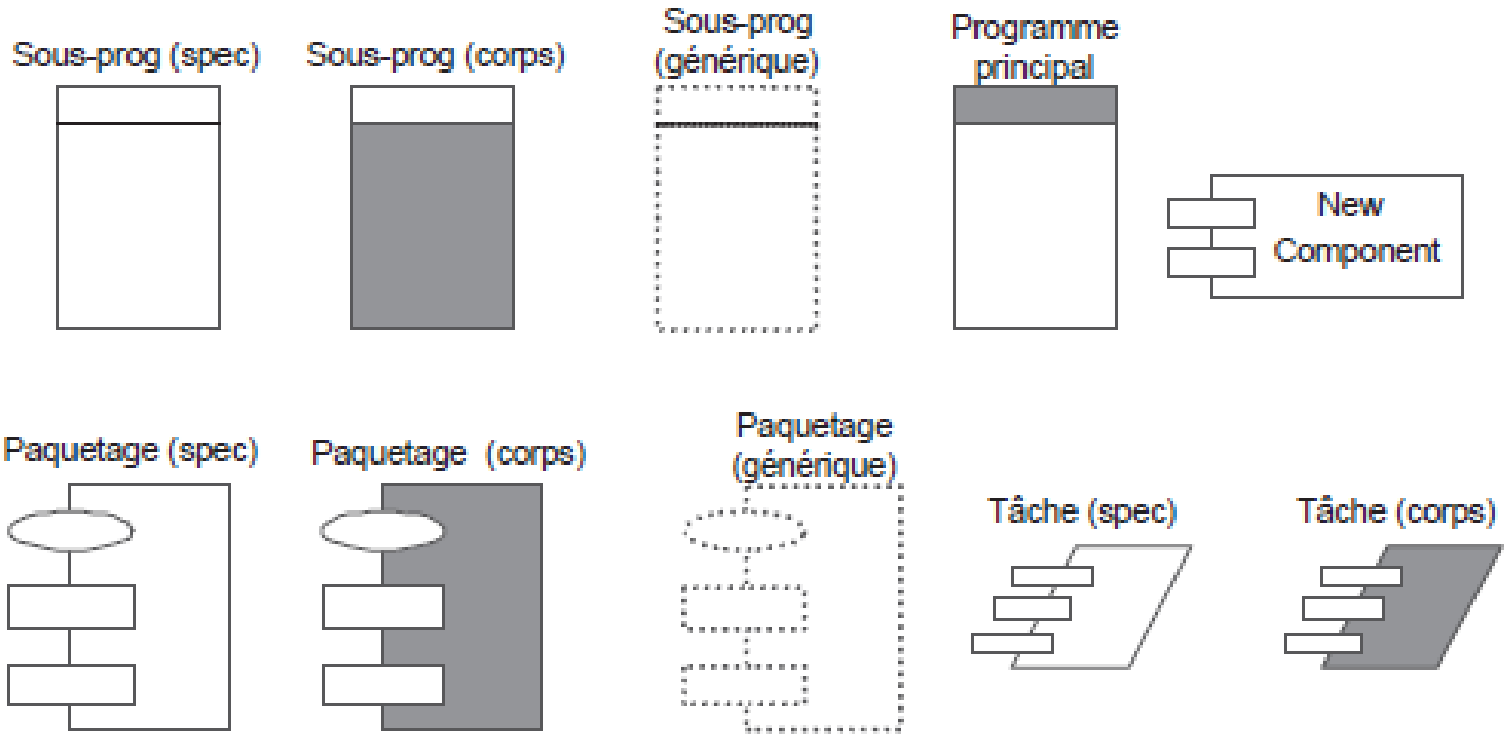
Diagrammes de composants

Diagramme de composants

permettent de décrire l'architecture physique et statique d'une application en terme de modules :

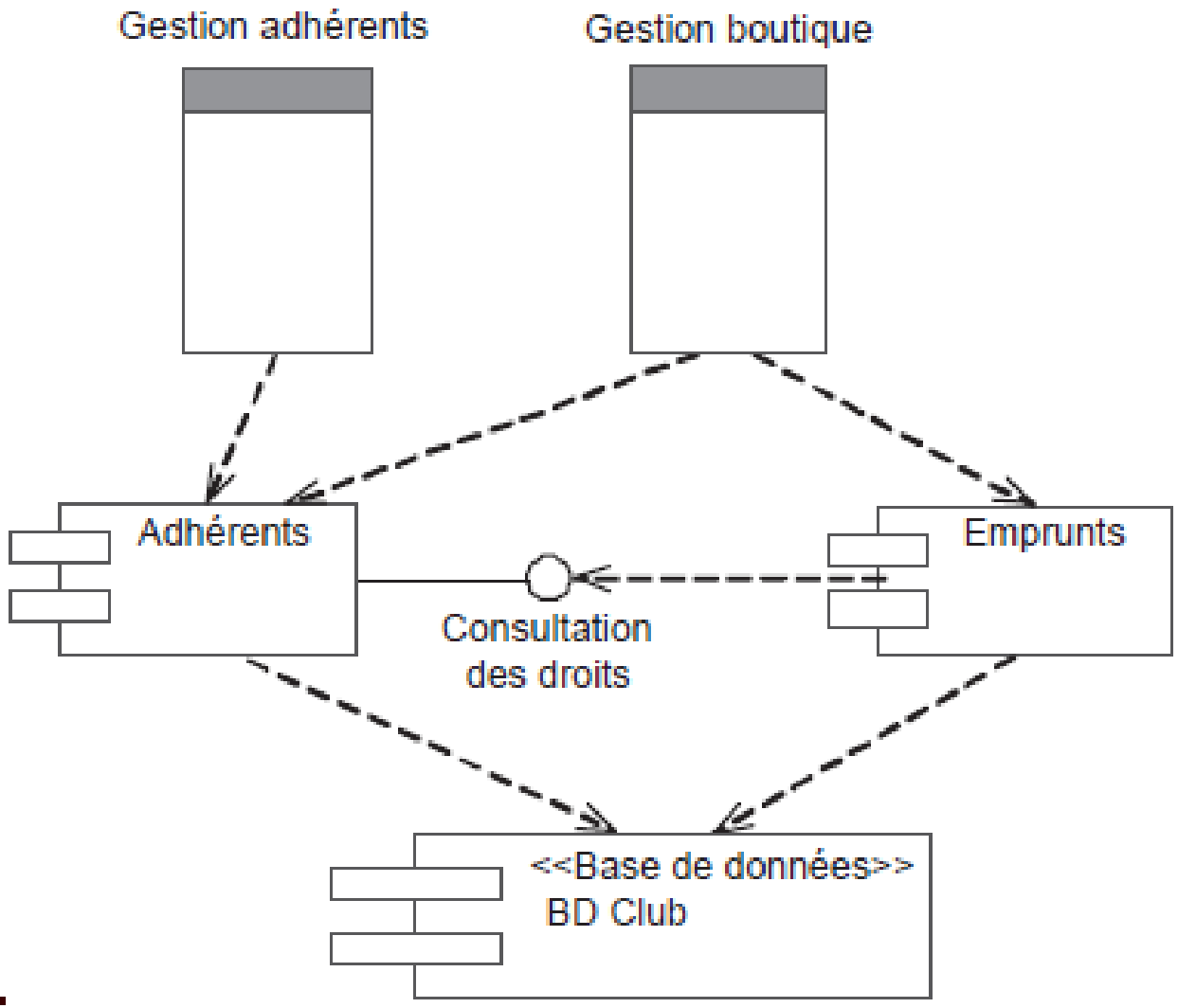
fichiers sources, librairies, exécutables, etc.

Ils montrent la mise en physique des modèles avec l'environnement de développement.



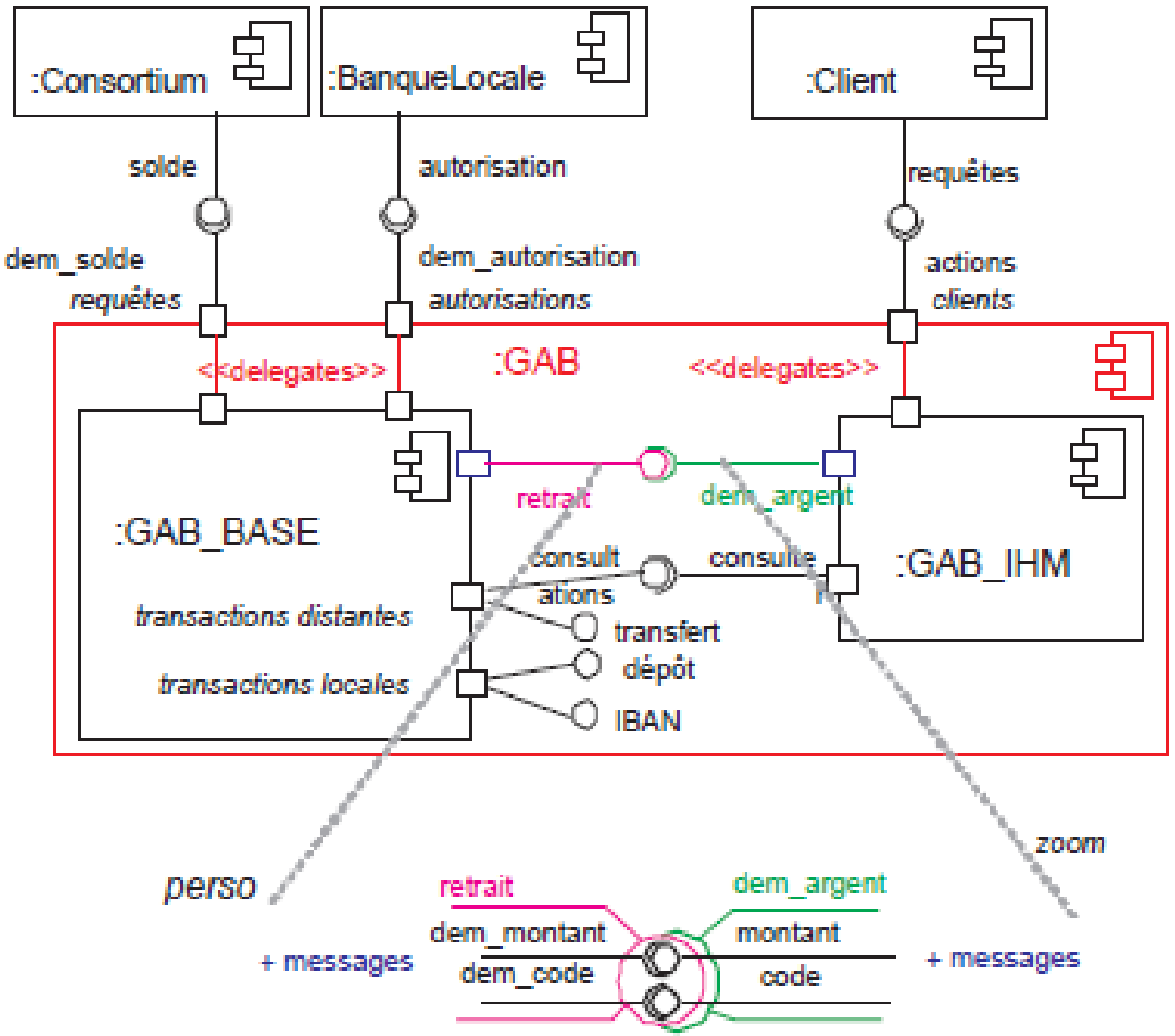
Diagrammes de composants

Diagramme de composants Exemple club vidéo



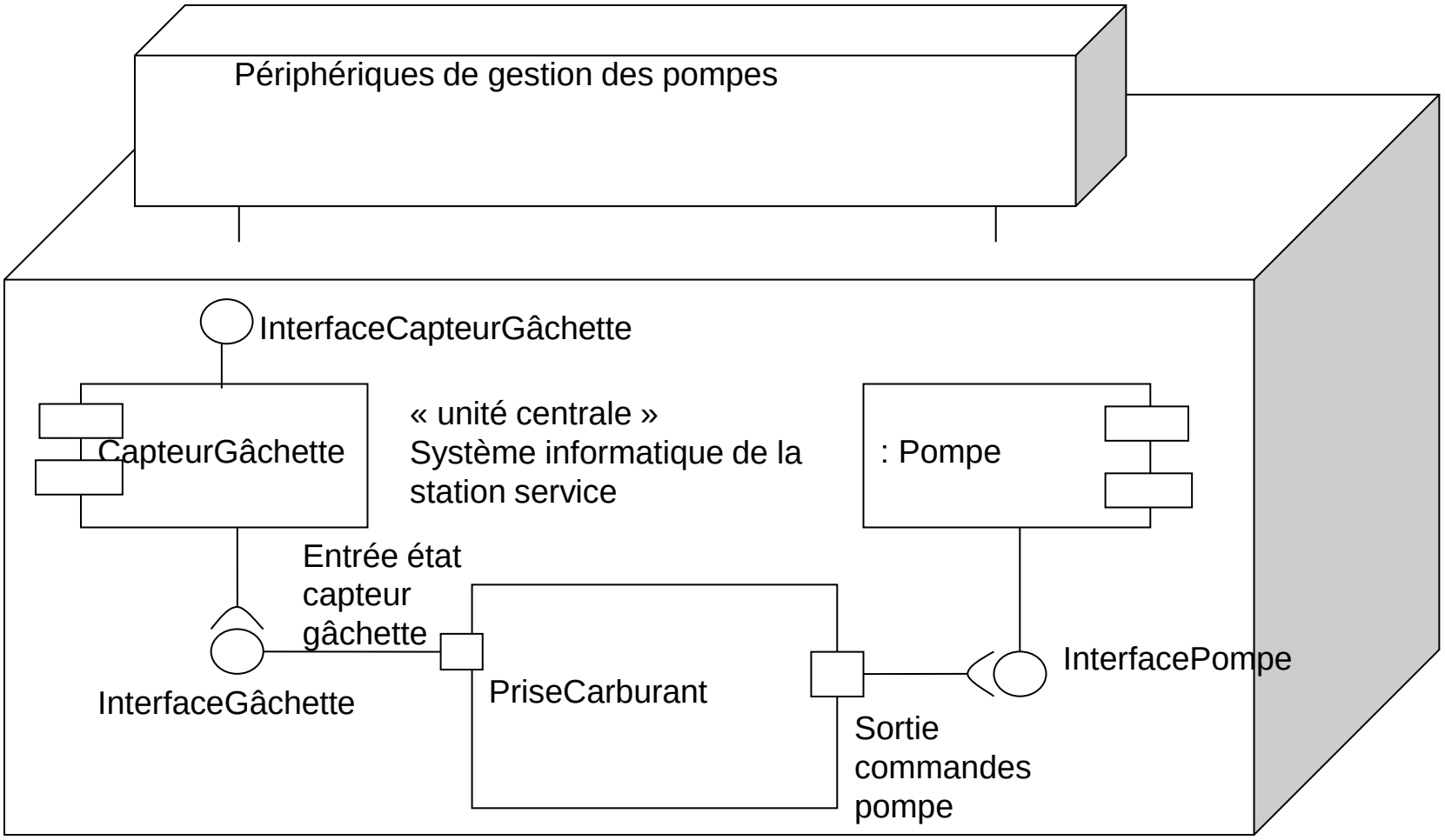
Diagrammes de composants

Diagramme de composants gestionnaire automatique de billets



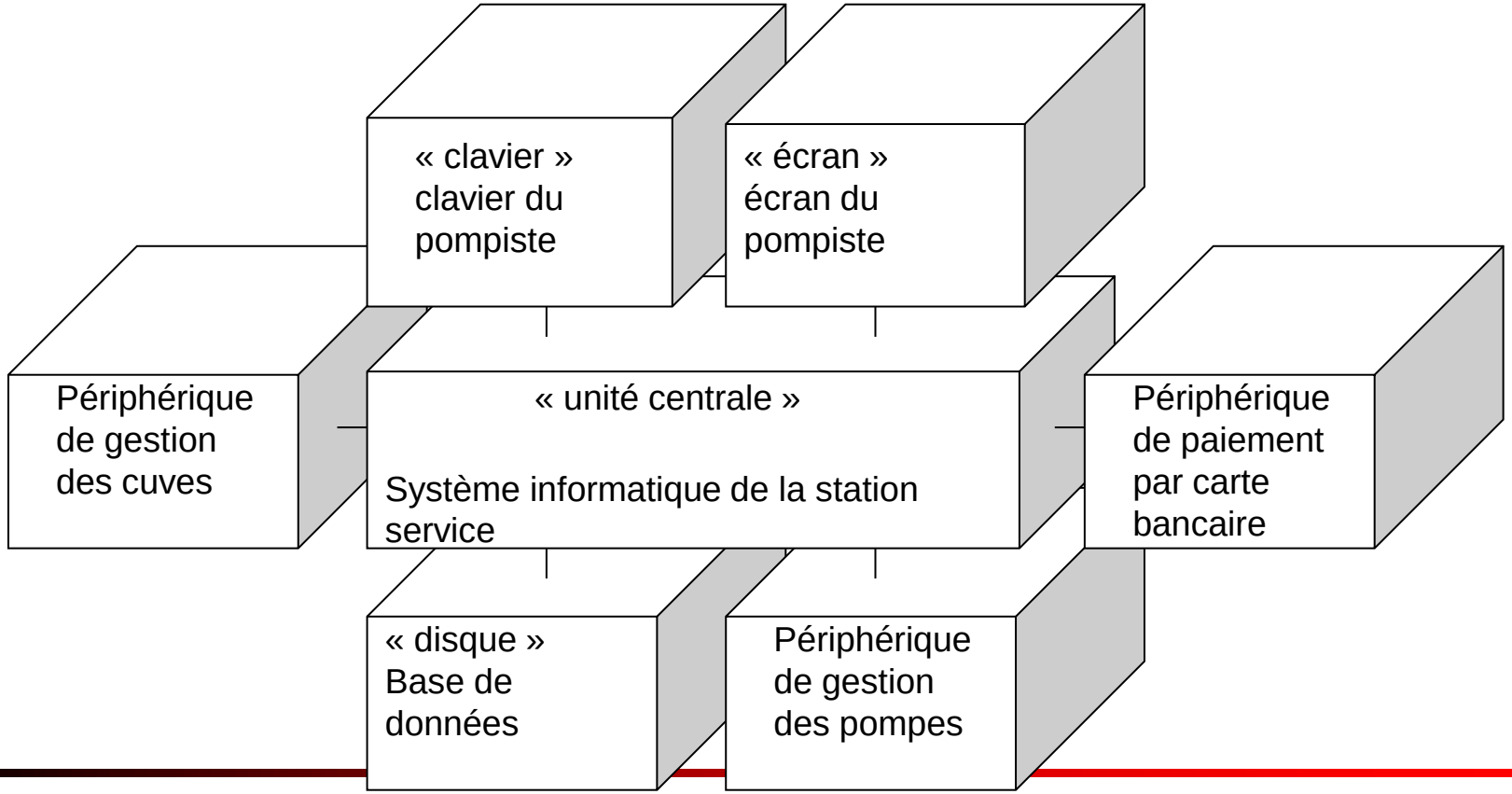
Diagrammes de composants

Diagramme de composants gestionnaire de pompe



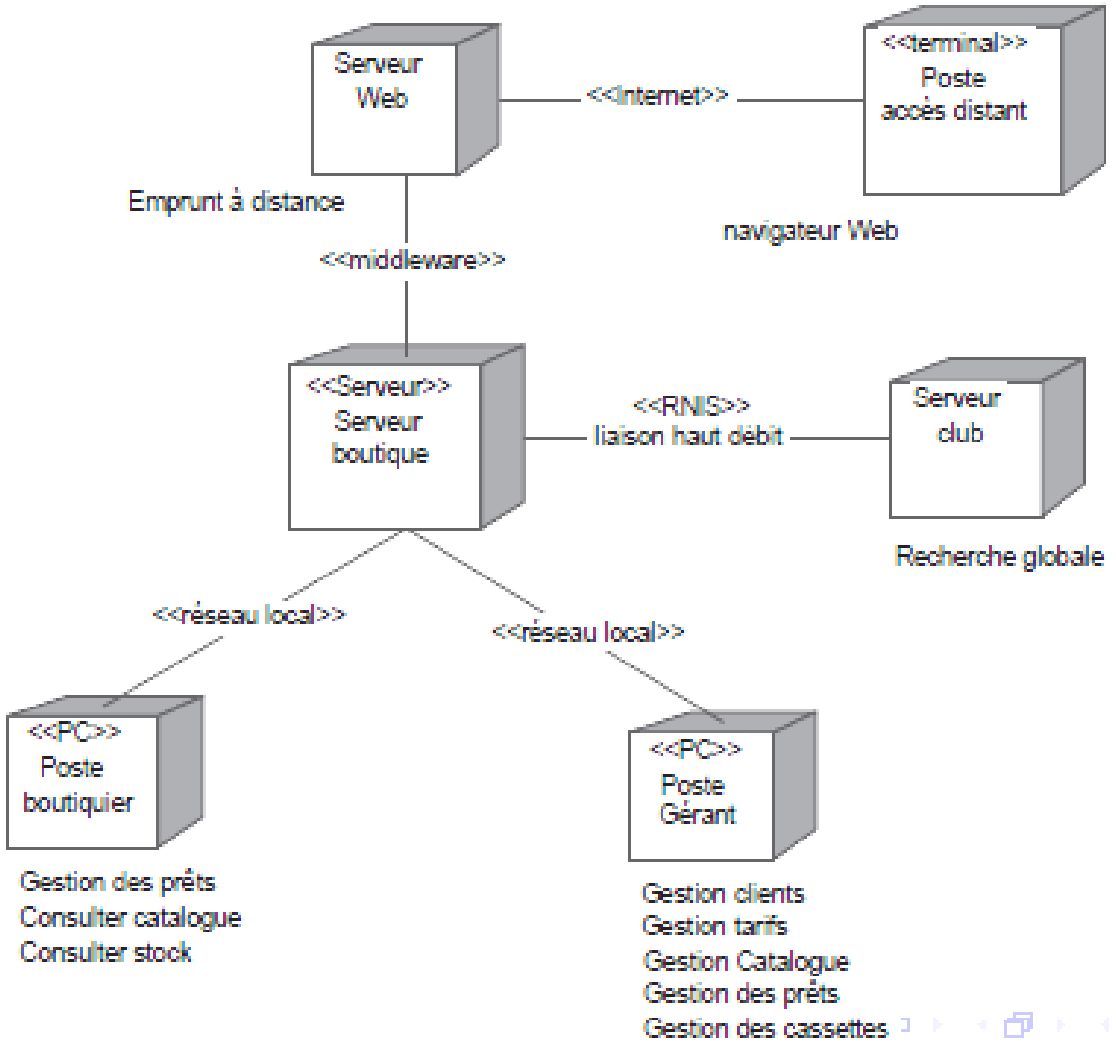
Diagrammes de déploiement

Diagramme de déploiement =
architecture physique+ répartition des composants sur les
*
)



Diagrammes de déploiement

Diagramme de déploiement - exemple



Modélisation

Chaîne complète de la démarche de modélisation du