Modélisation objet des systèmes d'information avec UML

Introduction - Diagramme de classe

Christine Verdier, Professeur UJF

Cours réalisé à partir des supports de cours de JP Giraudin, D. Rieu, A. Flory et C. Verdier

Historique

Deux courants

Traitements de
plus en plus complexes

Découpage en
sous-programmes indépendants

Programmation structurée

Données de plus en
plus volumineuses

Structuration et regroupement
en caractéristiques communes

Naissance de l'approche objet

Historique

- 1966 : langage Simula (notion de classe)
- Début de l'utilisation : en Intelligence Artificielle puis dans le domaine des BD
- Apports de la modélisation objet :
  - interfaces conviviales (fenêtres, outils de pointage, pictogrammes...)
  - applications complexes (domaine médical...)
  - Réutilisation, patrons

Notion d'objets

- Les entités conceptuelles du monde réel peuvent être représentées par des **objets**
  - Exemples d'objets :
    - Le nombre 12
    - L'étudiant Mathilde
    - Le dossier médical de Jeanne
    - Le diplôme de Lionel

Notion d'objets

- Ces objets servent à décrire le monde réel
- Les objets existent indépendamment les uns des autres
- Les objets possèdent des caractéristiques appelées **attributs**
  - Exemples : nom, prénom, âge, adresse sont des attributs de l'objet 'Etudiant Pierre'

Opérations et méthodes

- Une opération est une transformation qui est réalisée **par** ou **sur** un objet
- On appelle **méthode** l'écriture informatique de l'opération
- Exemple : L'attribut «âge » de l'objet « Patient Pierre» peut avoir la méthode :
  - ChangerAge

Encapsulation

- La description d'un objet nécessite :
  - la description de sa structure (ses attributs)
  - la description de ses comportements associés à l'objet (ses méthodes)
- Méthodes et attributs ne sont pas visibles de l'extérieur de l'objet

  On parle d'**encapsulation**

Les classes d'objet

- Les objets qui possèdent les mêmes attributs et les mêmes méthodes sont regroupés dans des **classes**
- Les objets représentent des **occurrences** de la classe (on parle aussi d'instanciation)
- Dans les approches objet, tout objet appartient nécessairement à une classe

Les classes d'objet

- Par exemple, tous les étudiants d'une université seront regroupés dans une classe ETUDIANT
- De la même manière, tous les enseignants de l'université seront regroupés au sein de la classe ENSEIGNANT

Les liens entre les classes

- Les classes ne peuvent pas être isolées dans un modèle de données. Elles sont forcément liées entre elles.
- Si l'on veut par exemple représenter qu'un étudiant suit un cours donné par un enseignant, il est nécessaire de définir des liens entre ces différentes classes.

Les liens entre les classes

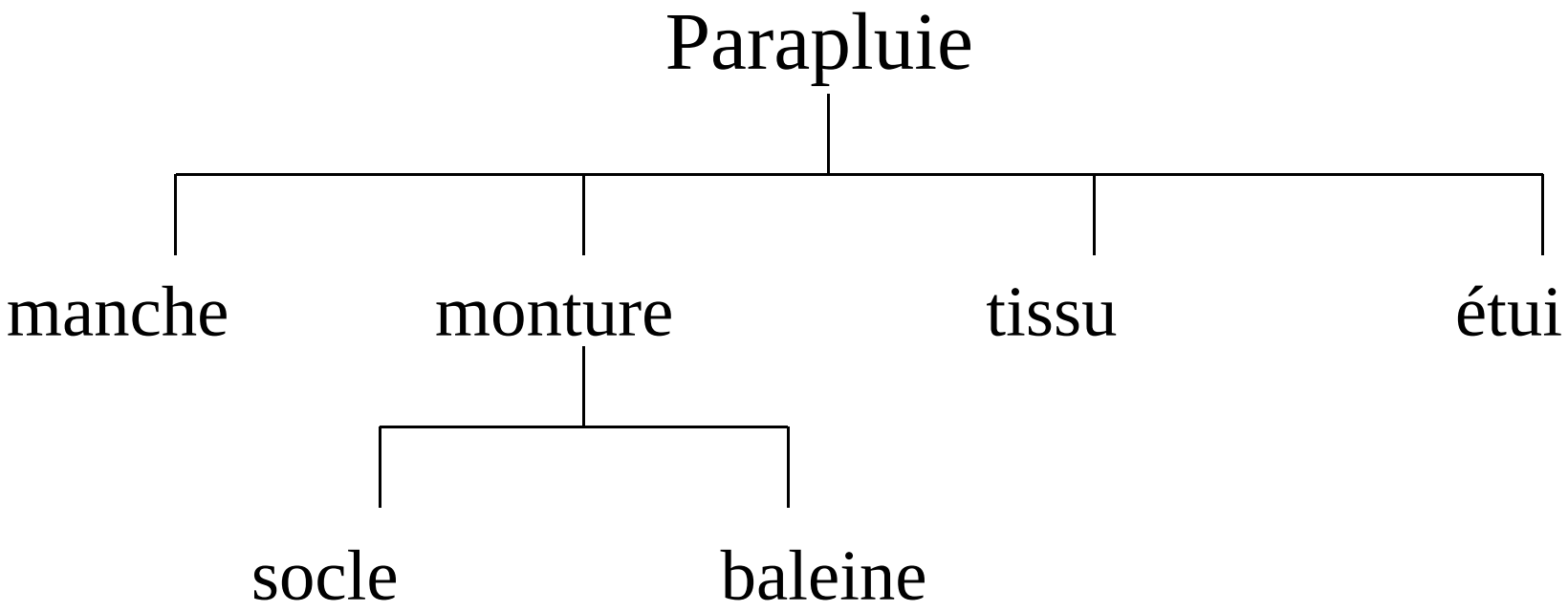
- Il existe plusieurs natures de liens qui représentent une sémantique différente.
- En objet, on trouve les liens suivants :
  - Agrégation
  - Généralisation/spécialisation → héritage
  - Association

L’agrégation

- L’agrégation permet de construire des objets complexes à partir d’autres objets appelés objets composants
- L’agrégation se matérialise par un graphe acyclique
- Le lien qui relie les objets composants signifie «composé» ou «est composé»

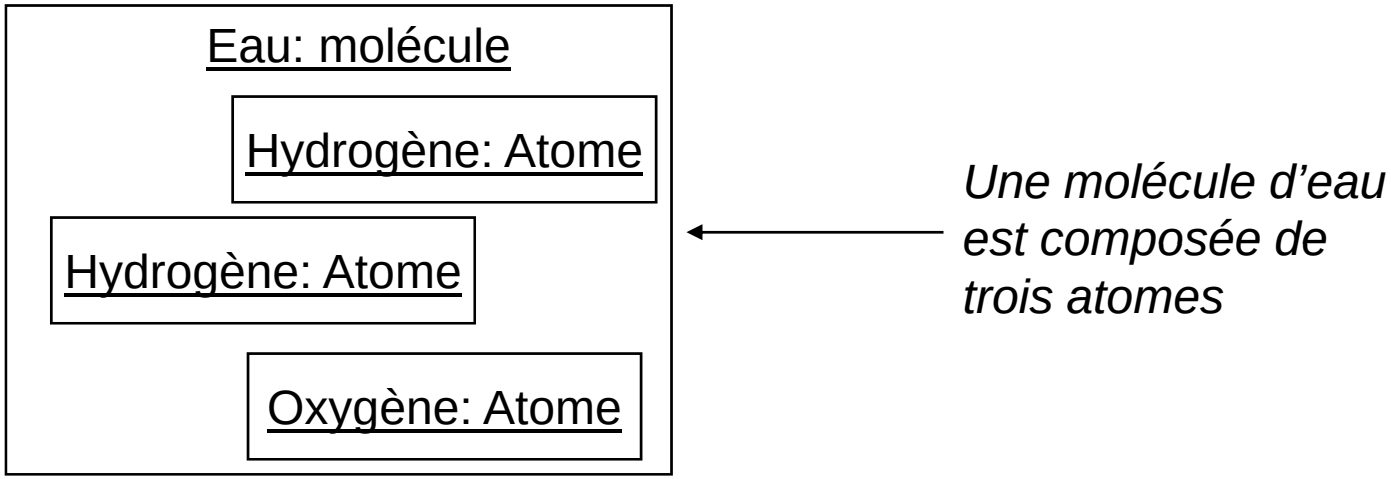
L’agrégation

- Exemple d’agrégation : la nomenclature



L'agrégation

- Les objets d'une classe sont les composants d'une autre classe
- Intérêt : partir d'objets de base pour arriver à des objets plus complexes

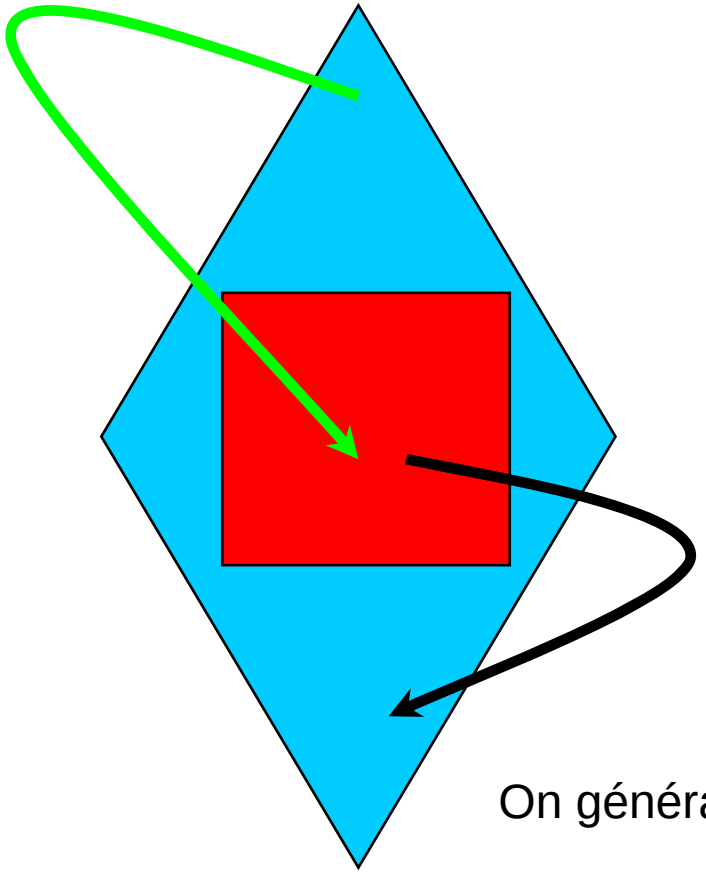


Généralisation et spécialisation

- Ces concepts permettent de modéliser l'emboîtement de classes les unes dans les autres
- Exemple :
  - Soient la classe C des carrés et la classe L des losanges
  - On dit que L généralise C et que C spécialise L
  - C est une sous-classe de L
  - L est une sur-classe de C

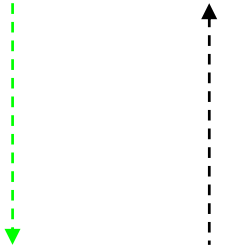
Généralisation/spécialisation

On spécialise



On généralise

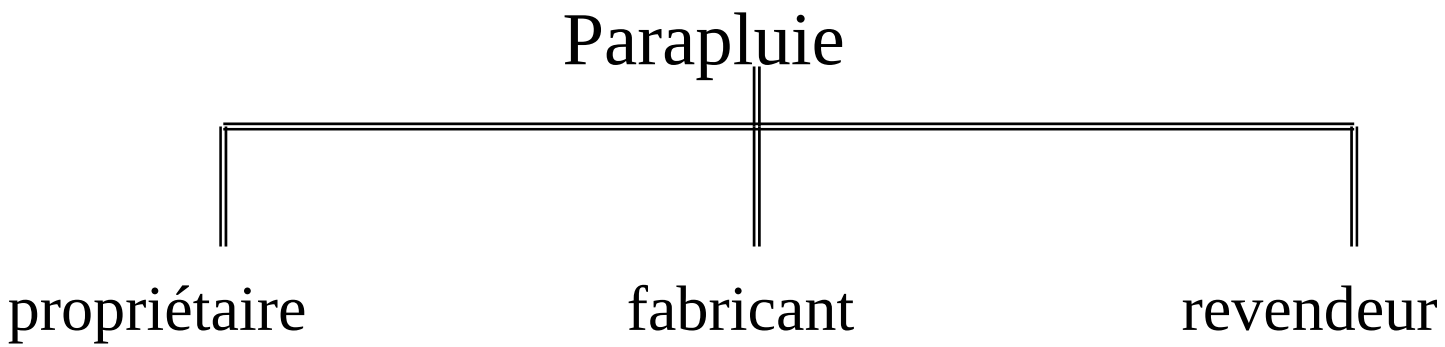
Losange = quadrilatère dont les 4 côtés sont de même longueur



Carré = losange qui a 4 angles droits

Les associations

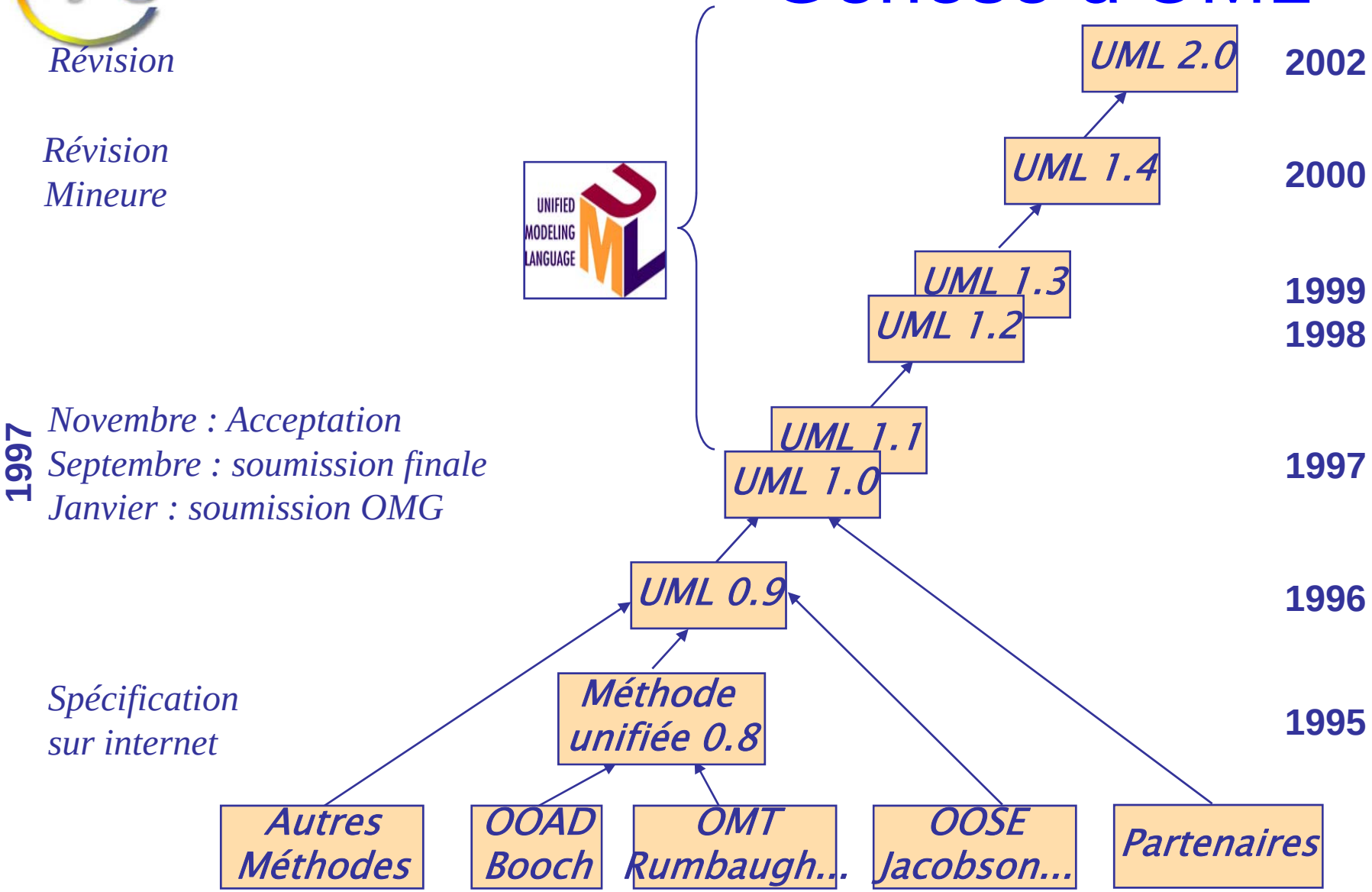
- L’association est un mécanisme qui permet de regrouper des objets qui ne sont pas reliés par une relation d’agrégation ou par une relation de généralisation/spécialisation



Le modèle UML
(**U**nified **M**odeling **L**anguage)

- Normalisation des types de modélisation objet (OMT, Booch, OOSE)
- UML est un langage objet graphique
- UML n’est pas une méthode de conception
- UML est soumis à l’OMG (Object Management Group)

Genèse d'UML


UML 2.0
2002
UML 1.4
2000
UML 1.3
1999
UML 1.2
1998
UML 1.1
UML 1.0
1997
UML 0.9
1996
Méthode unifiée 0.8
1995
Autres Méthodes
OOAD Booch
OMT Rumbaugh...
OOSE Jacobson...
Partenaires
UNIFIED MODELING LANGUAGE
Révision
Révision Mineure
1997
Novembre : Acceptation
Septembre : soumission finale
Janvier : soumission OMG
Spécification sur internet

Un Modèle = Un point de vue sur le système

- Modèle de Classes qui capture la structure statique
- Modèle des Cas d'Utilisation – *Use Case*, UC – qui décrit les besoins, les fonctions
- Modèle d'Interaction qui représente les scénarios et les flots de messages
- Modèle des États qui exprime le comportement dynamique des objets, classes...
- Modèle de Réalisation qui montre des unités de travail
- Modèle de Déploiement qui précise la répartition des processus
- Descriptions abstraites pour capturer la sémantique d'un système

Statique (ce que le système EST)

- diagramme de classes
- diagramme d'objets
- diagramme de composants
- diagramme de déploiement

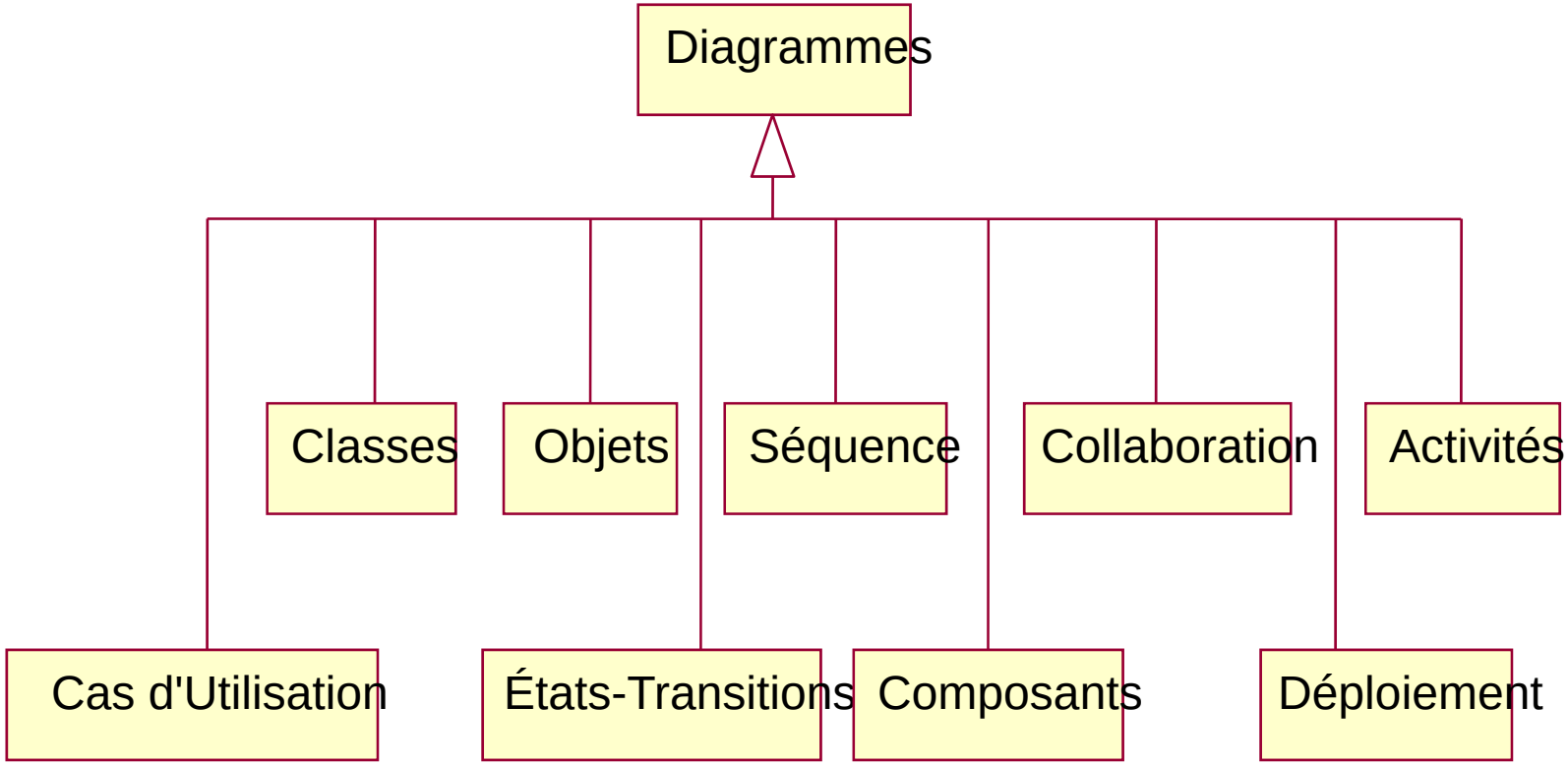
Dynamique
(comment le système EVOLUE)

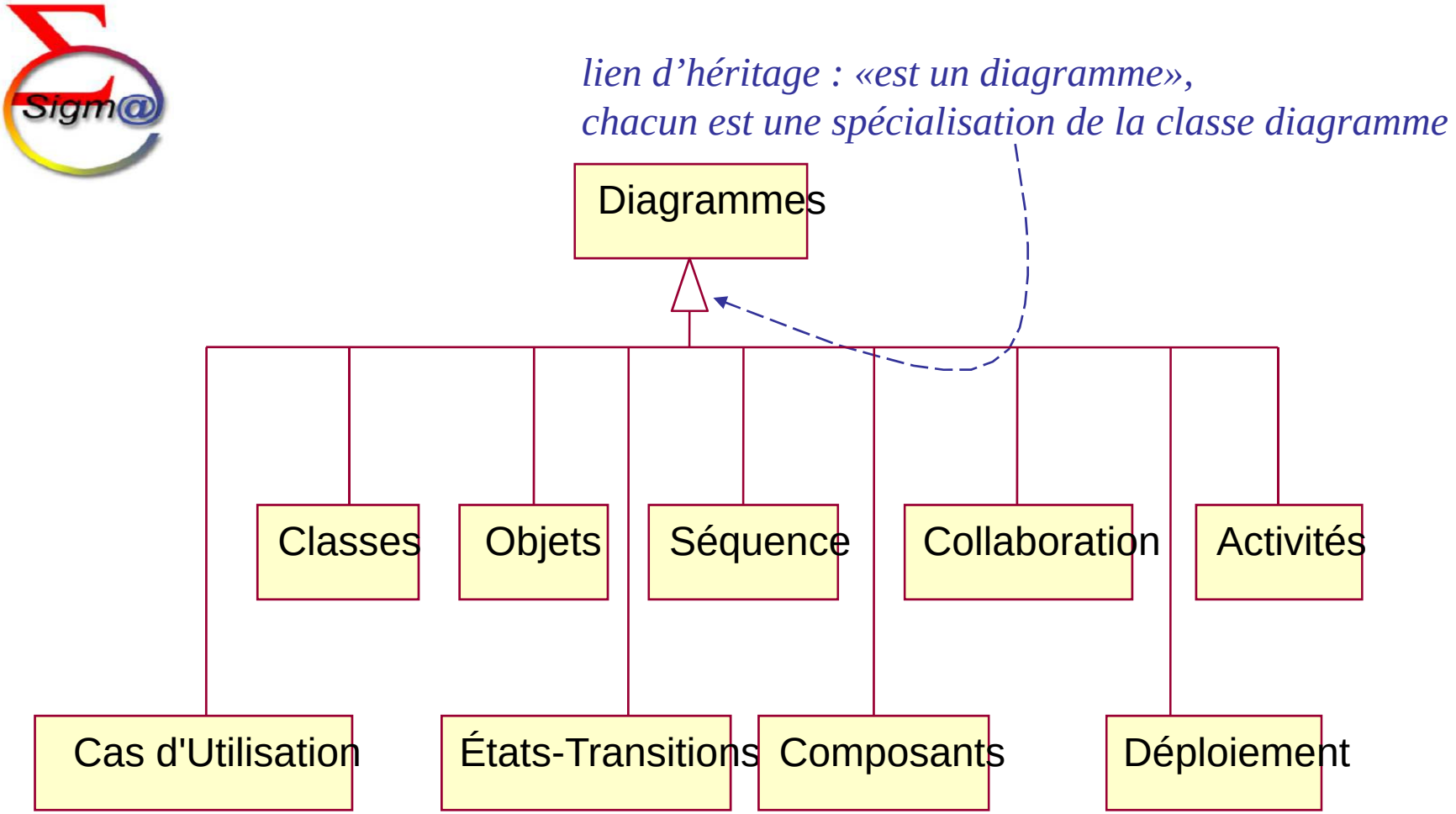
- diagramme de séquence
- *diagramme de collaboration*
- diagramme d'états-transitions
- diagramme d'activités

Fonctionnel
(ce que le système FAIT)

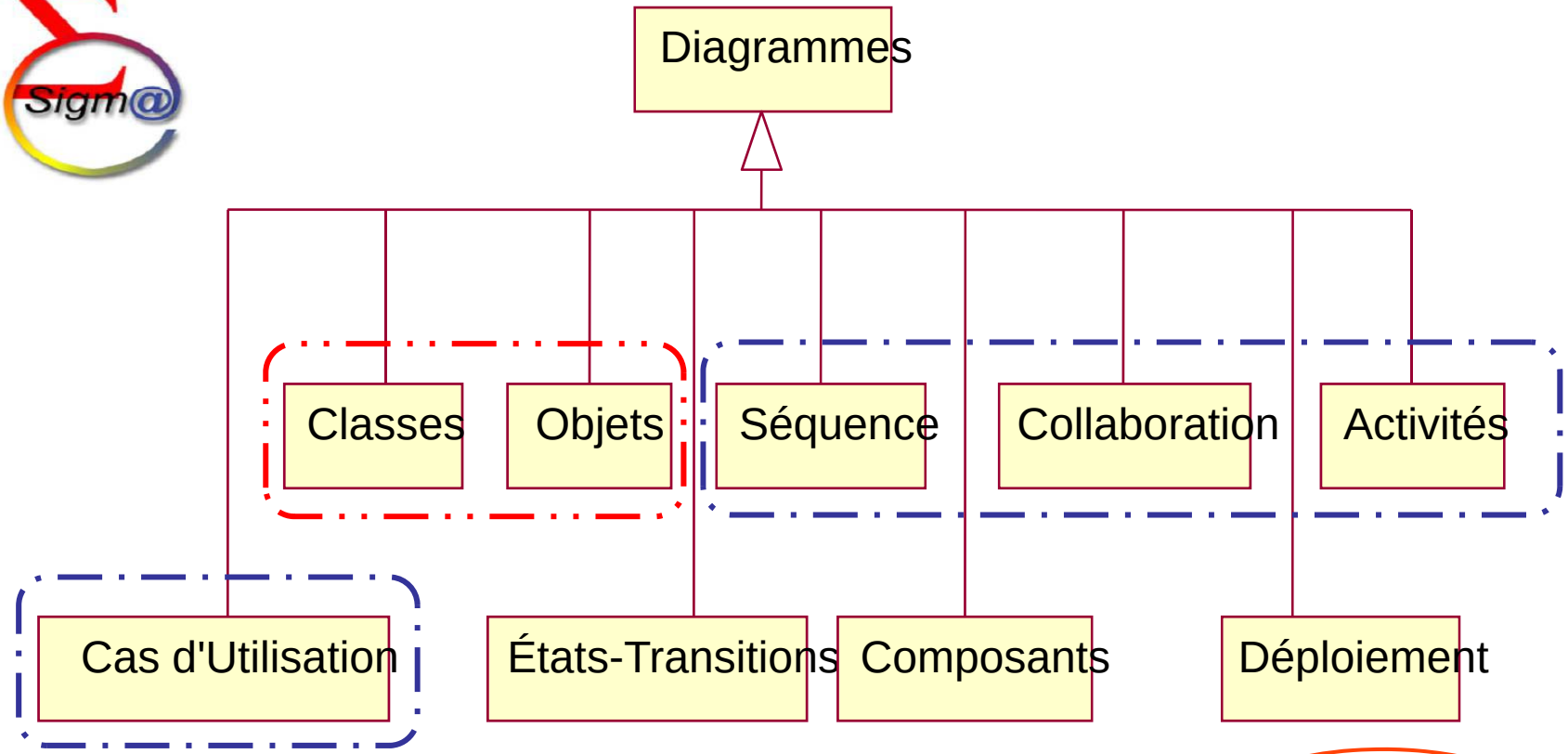
- diagramme de cas d'utilisation
- *diagramme de collaboration*

Diagrammes


Diagrammes
Classes
Objets
Séquence
Collaboration
Activités
Cas d'Utilisation
États-Transitions
Composants
Déploiement



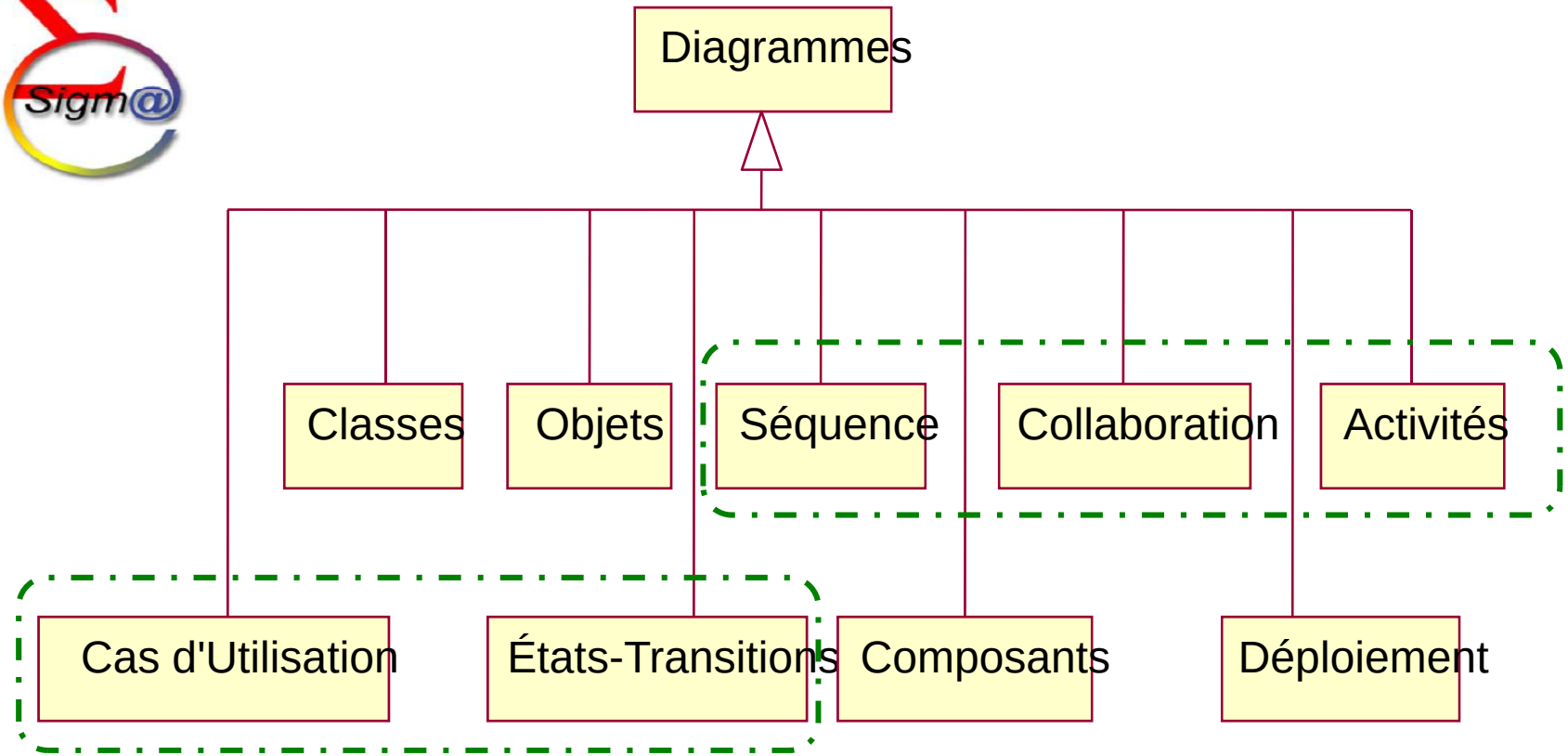
Un diagramme est un « langage graphique » de phrases traduisant entre les concepts « éléments du vocabulaire du diagramme », des relations matérialisées par des arcs du graphe, écrits avec un forme particulière.
L'ensemble donne la syntaxe graphique du langage associé



- Aspects structurels statiques
  - diagramme de classes : classes et relations statiques
  - diagramme des objets : objets et liens
- Aspects fonctionnels et *dynamiques*
  - diagramme de cas d'utilisation : acteurs et utilisation du système

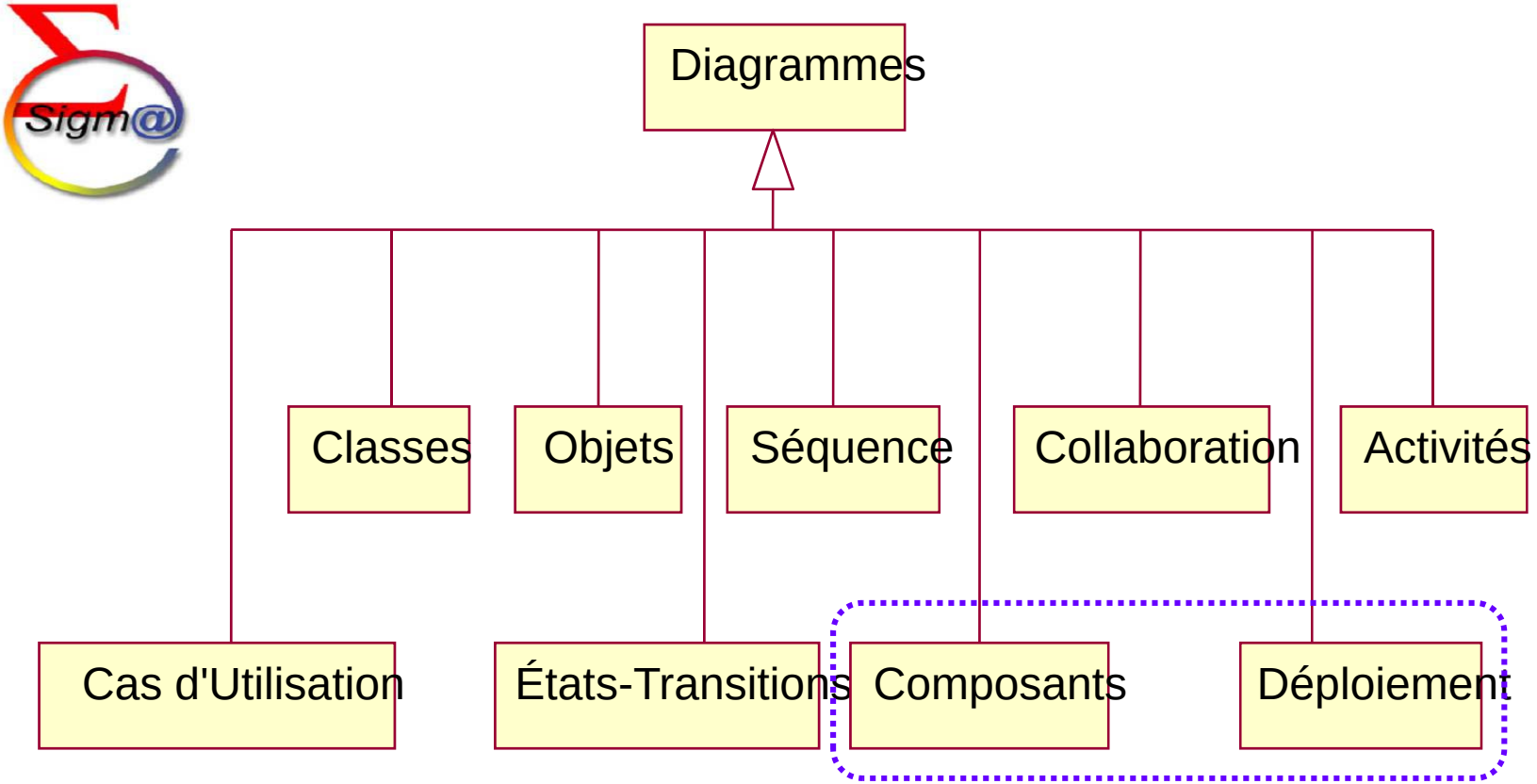
Ce que le système EST

Ce que le système FAIT



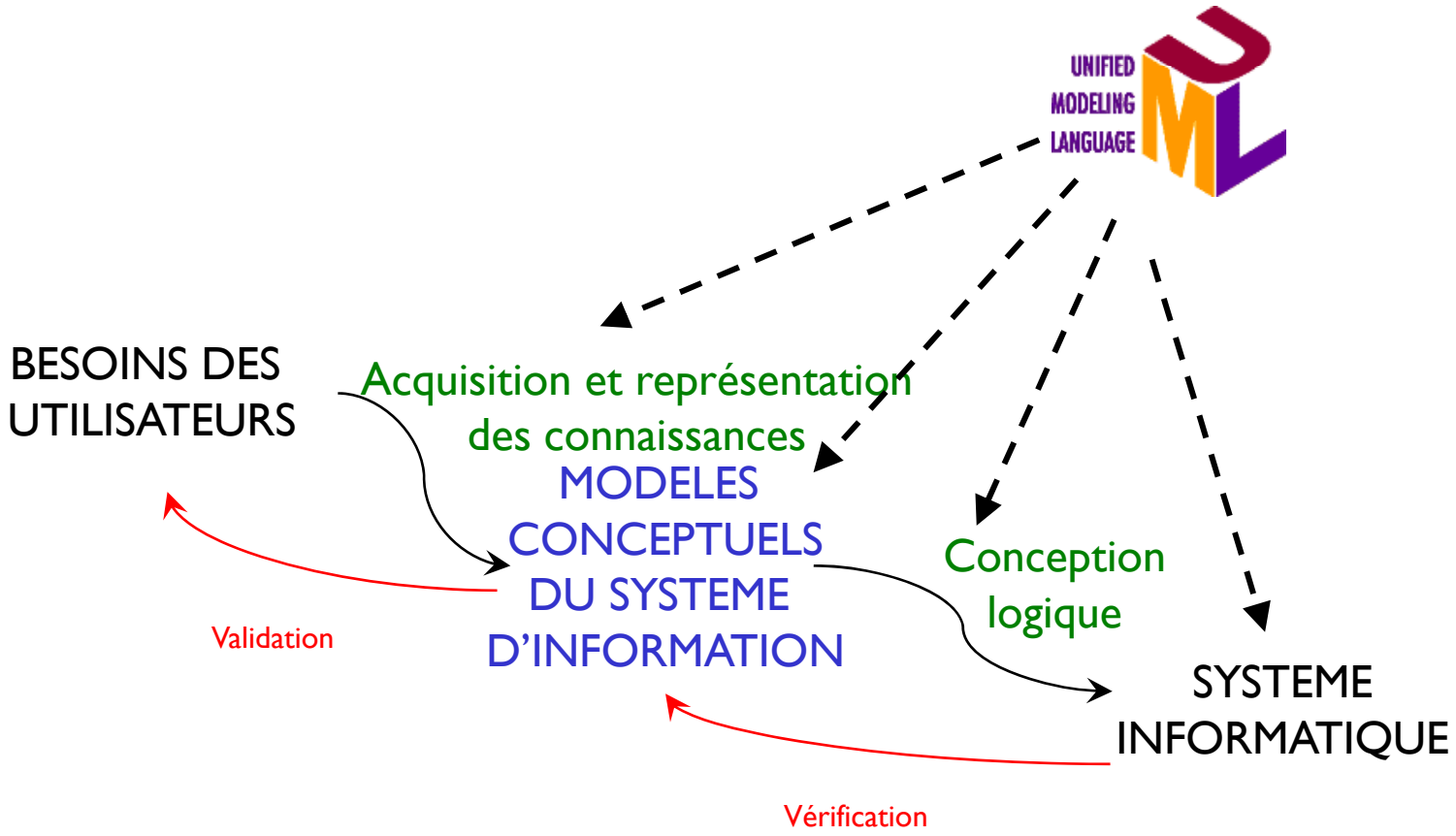
- Aspects dynamiques
  - diagramme de séquence : vision temporelle des interactions
  - diagramme de collaboration : vision spatiale des interactions
  - diagramme d'états-transitions : comportement des objets
  - diagramme d'activités : flot de contrôle interne aux opérations

comment le système EVOLUE



- Aspects implantation
  - diagramme de composants : codage
  - diagramme de déploiement : implantation, distribution

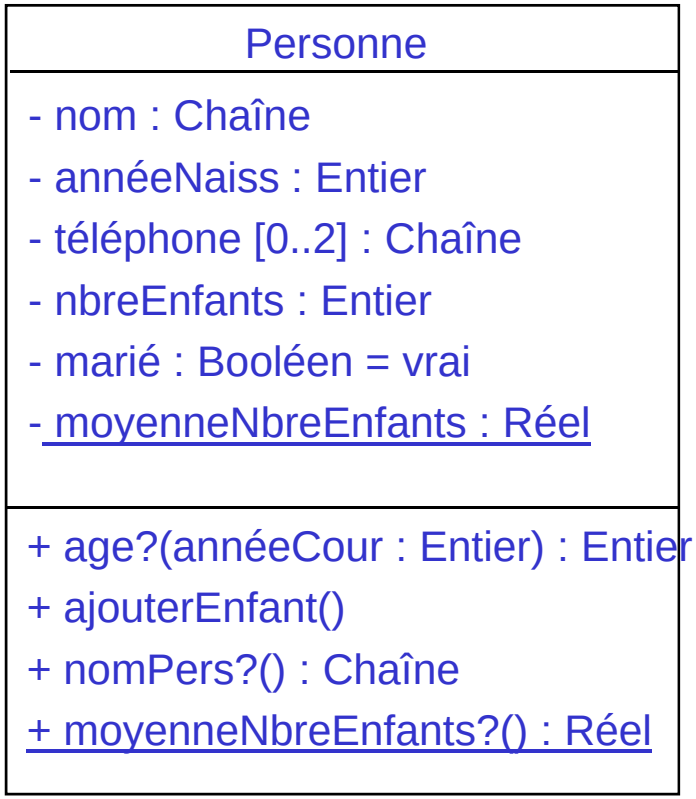
Ingénierie des SI et des BD



Classe

Une classe est un ensemble d'objets ayant mêmes attributs, mêmes opérations, mêmes relations, et même sémantique.

Classe avec ses attributs et ses opérations



nom de la classe (*commence par une majuscule*)

attribut

- visibilité
- nom (*commence par une minuscule*)
- multiplicité (*1 par défaut*)
- type de la valeur
- valeur par défaut ou valeur initiale

attribut de classe (*souligné*)

opération

- visibilité
- nom (*commence par une minuscule*)
- paramètres typés
- type du résultat

opération de classe (*souligné*)

Notation visibilité

+ public
# protégé
- privé

La valeur par défaut est affectée à l'attribut à la création des instances de la classe à moins qu'une autre valeur ne soit spécifiée.

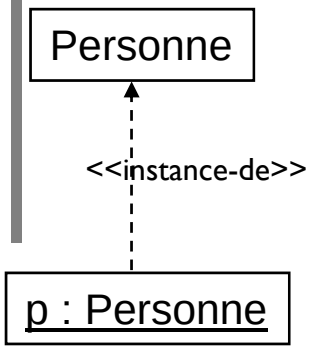
La multiplicité indique le nombre de valeurs possibles pour l'attribut

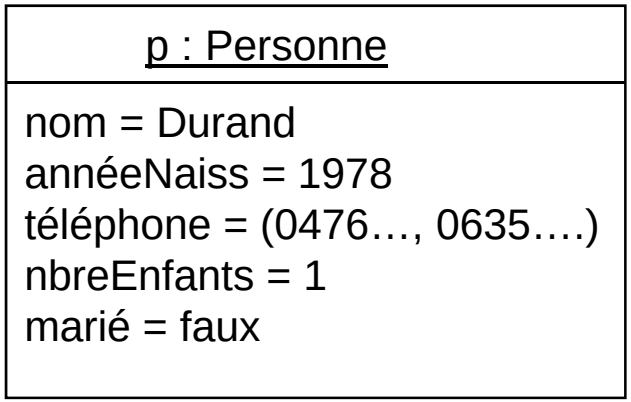
Objet

Un objet est une instance (une occurrence) d'une classe.

Une classe est un modèle caractérisé par des propriétés(attributs et méthodes) communes à des objets et permettant de créer des objets possédant ces propriétés.

Un système informatique en exploitation est composé d'un ensemble d'objets de différentes classes qui collaborent.





nom de l'objet et
nom de la classe (*soulignés*)

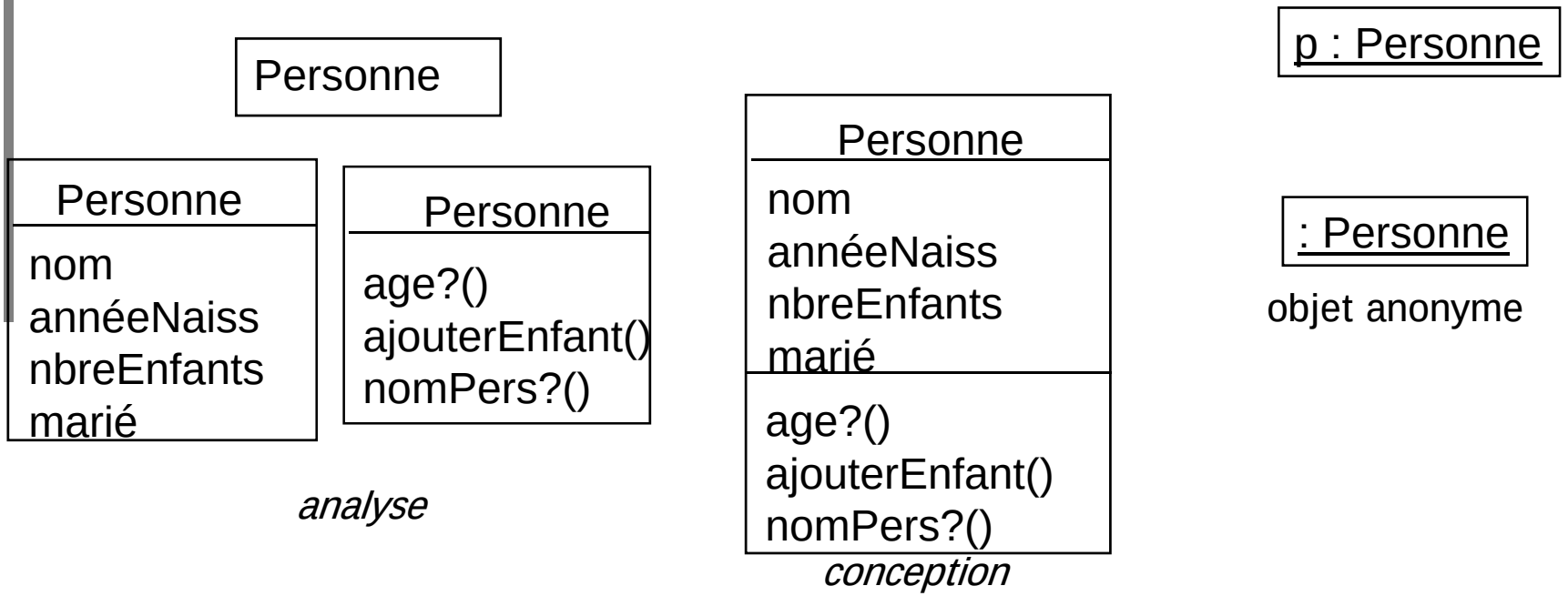
noms des attributs et valeurs

Note : *la valeur d'un attribut de classe est associée à la classe et non pas à chaque objet*

Notations

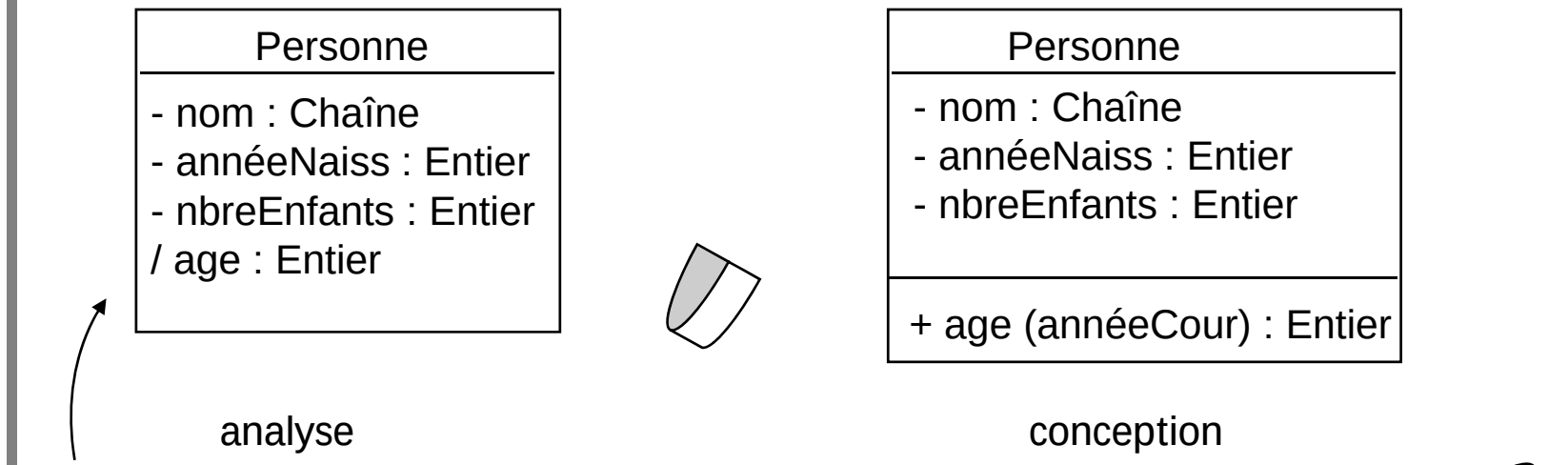
Le niveau de détail du diagramme est adapté au niveau d'abstraction correspondant à une phase du cycle de vie du logiciel et d'autre part au type de communication.

Classe Objet



Attribut dérivé

Un attribut dérivé est un attribut dont la valeur est calculée à partir de celles d'autres attributs.



réalisation

```
age?(annéeCours : Entier) : Entier
    retourner (annéeCours-annéeNaiss );
```

Classe = attributs + opérations + responsabilités

Vol
numvol hdep har
Définir-périodes()
définit la desserte d'une ligne (ville de départ, ville d 'arrivée) selon un horaire donné

Avion
no-immat : integer type : string rayon : integer hrvol : hrcumul = 000 •••
ajouter-avion(no-immat, type, rayon) enreg-vol (durée : hrcumul) : hrcumul affecter-instance(numvol, jr, sem) mettre-révision :date •••
assure le transport du fret et des passagers selon les affectations correspondant à la politique commerciale

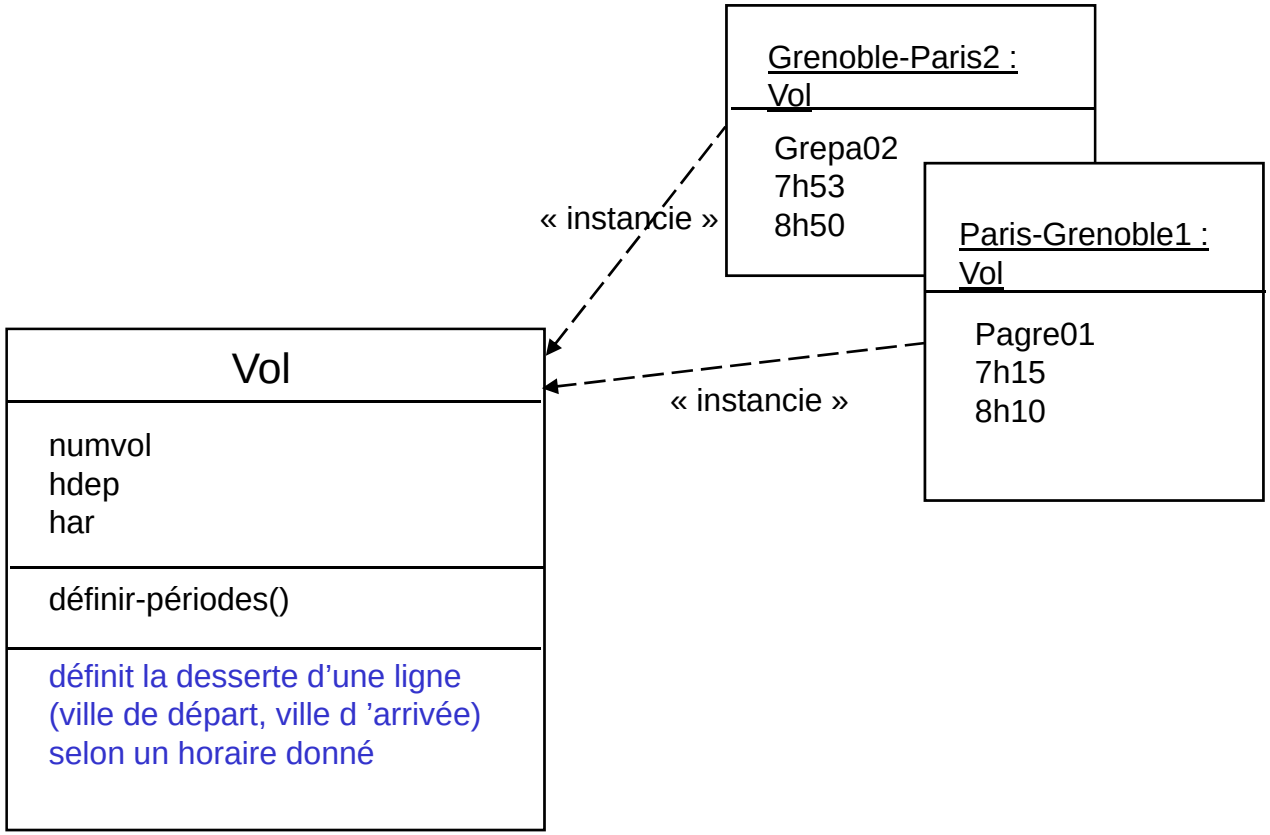
Les attributs sont typés ou non, simples ou multiples, avec valeur initiale éventuelle.

Les opérations sont avec ou sans leurs paramètres.

Les responsabilités sont très utiles en phase d'analyse de besoins pour décrire le rôle des objets de la classe par rapport à l'environnement, pour se concentrer sur le « pourquoi » et pas uniquement sur les structures (attributs) ou les comportements (opérations).

Classe & objets

Une classe correspond à un type (attributs, opérations, propriétés, …) et c'est un conteneur d'objets conformes à ce type.



Attributs

Une syntaxe de base et des propriétés

[visibilité] [/] nom [multiplicité] [: type][= valeur initiale] [{propriétés et contraintes}]

4 valeurs de visibilité :

- **+ : public, visible de toute classe**
- **# : protégé, visible dans la classe et ses *sous-classes***
- **- : privé, visible dans la classe uniquement**
- **~ : visible dans le paquetage uniquement**

des propriétés :

- **ordered, ordonné pour un attribut à valeurs multiples**
- **unique, unicité pour un attribut à valeurs multiples**
- **…**

Attributs

Exemples

…

-soldeCourant : float = 0

-/estADecouvert : boolean {estADecouvert = = true si soldeCourant < 0 }

…

- largeur : int {largeur > 0}

-…

-villes_Voyage [1..*] : Ville

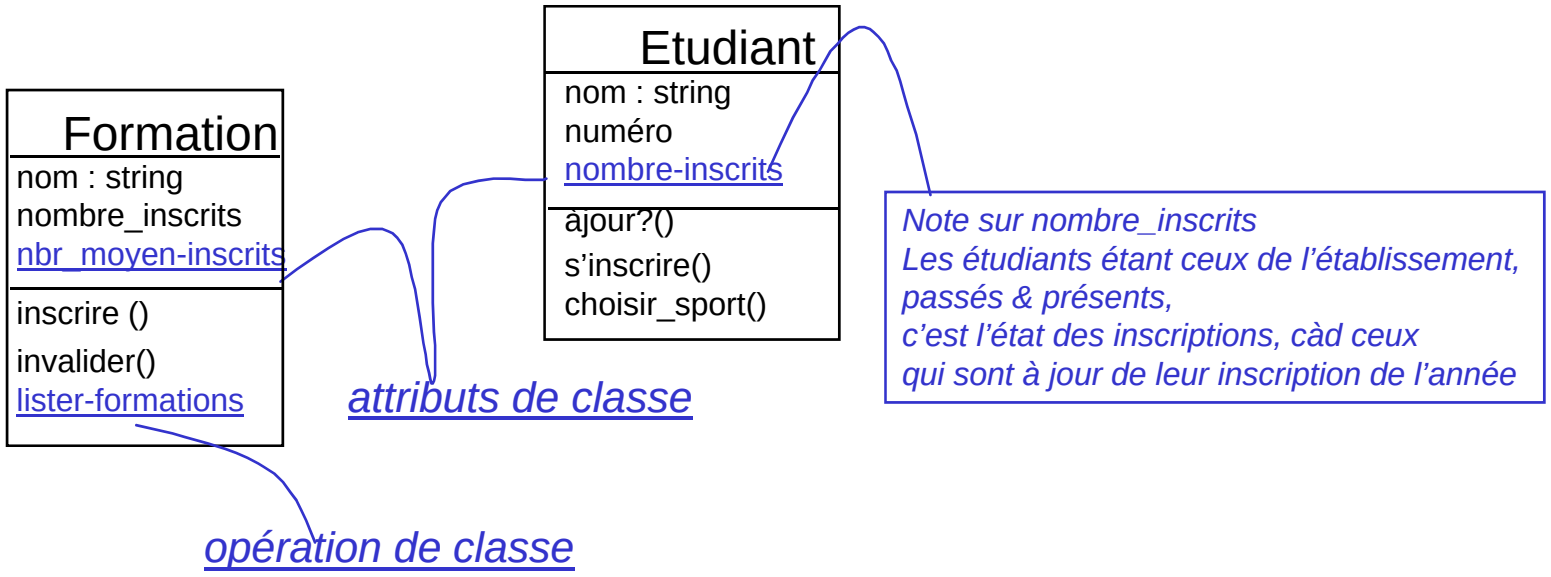
-villes_Itinéraire [1..*] : Ville {not unique, ordered}

…

Attributs et Opérations de classes

Les attributs et les opérations concernent tous les objets de la classe et s'appliquent sur chaque objet de la classe.

On peut définir des attributs et des opérations qui s'appliquent sur la classe elle-même, c'est-à-dire sur la collection des objets de la classe ou un sous-ensemble de ceux-ci.



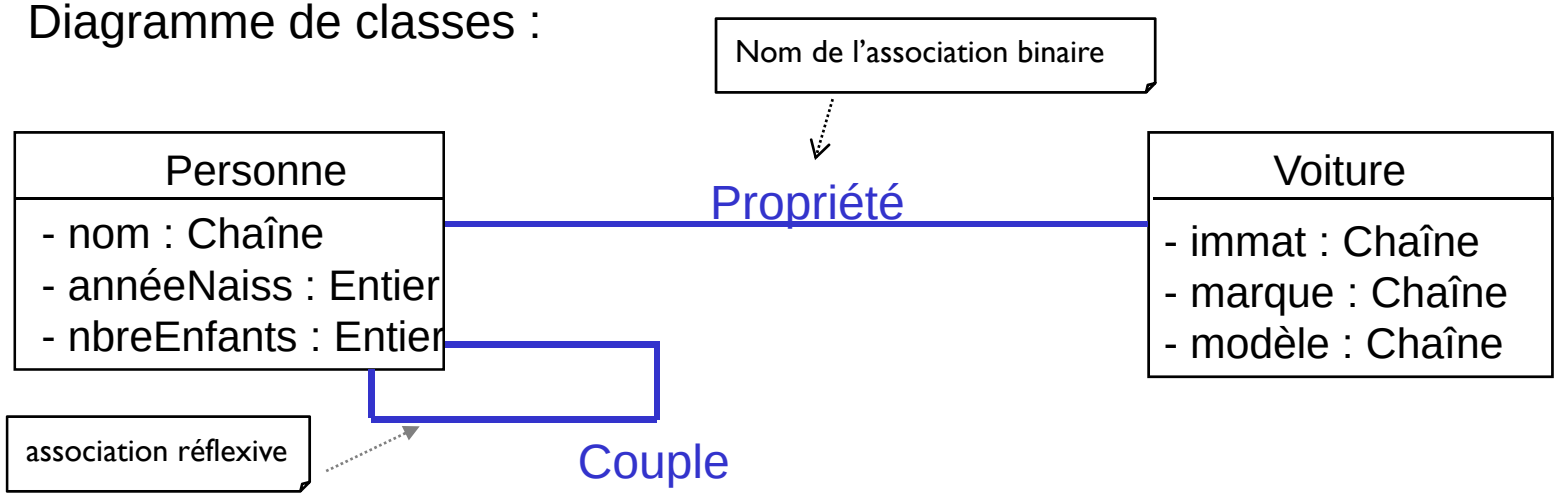
Association

Le concept d'association ne fait pas partie des concepts élémentaires du paradigme objet.

Une association entre deux classes est une relation ou ensemble de liens entre des objets de ces classes.

Une association binaire peut être vue comme un sous-ensemble d'un produit cartésien entre les ensembles des objets des classes associées.

Diagramme de classes :

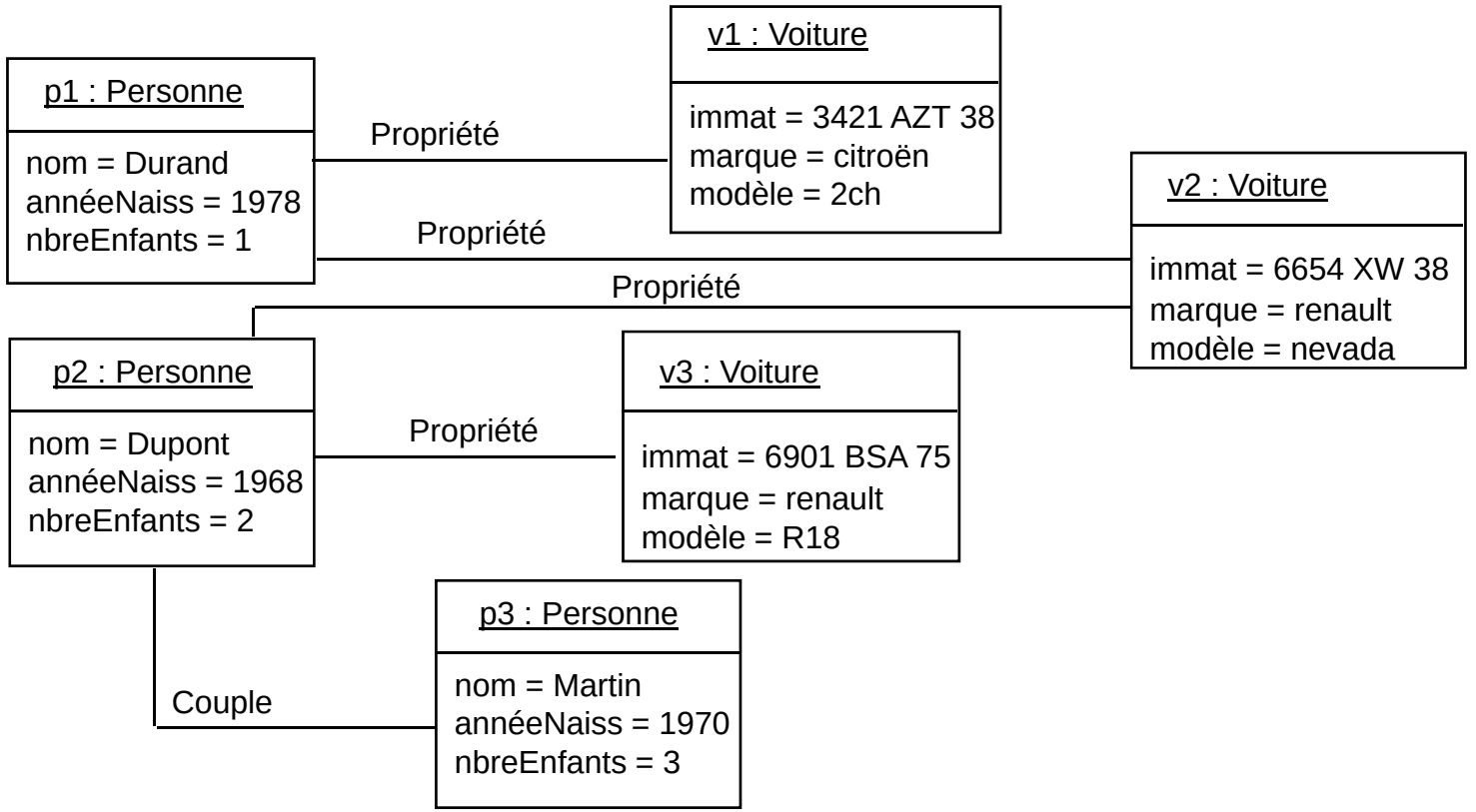


Liens

Un lien indique une connexion entre des objets.

Un lien est une instance d'association.

Diagramme d'objets :

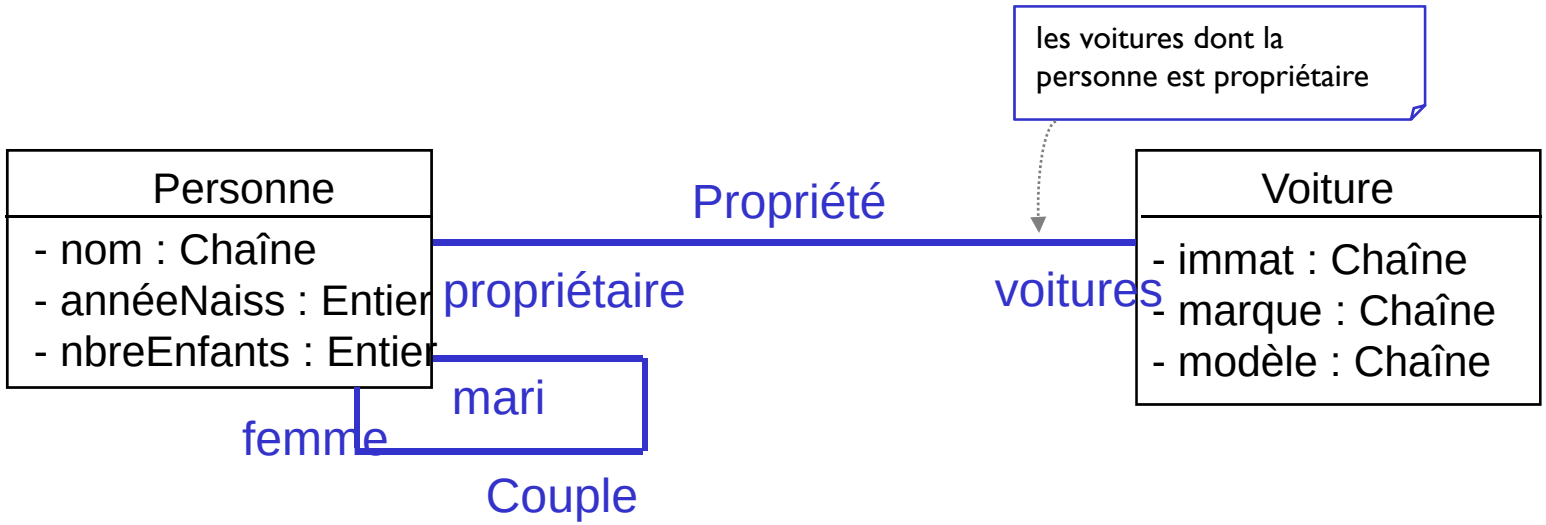


Rôles d'une association

Une association possède 2 rôles inverses l'un de l'autre.

Un rôle indique comment une classe source voit la classe destination.

Le nom du rôle est écrit du côté de la classe qui joue ce rôle.

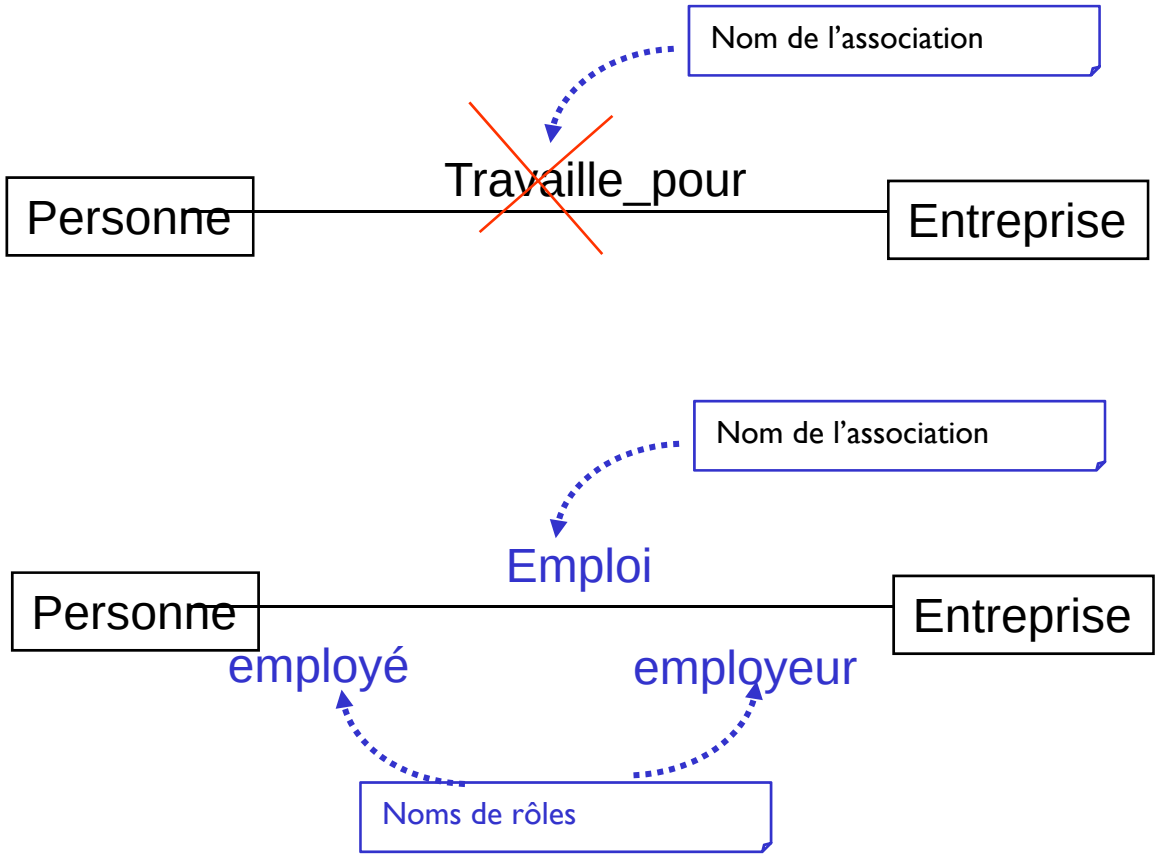


Le nom de l'association ne doit pas être confondu avec un nom de rôle.

Il est souvent plus facile de nommer les rôles d'une association plutôt que l'association elle-même. Dans ce cas, le nom de l'association peut être omis.

Nom d'association

Il ne faut pas confondre le nom de l'association avec le nom d'un rôle.



! Il est recommandé de donner des noms d'associations qui ne nécessitent pas d'indiquer de sens de lecture, c'est-à-dire des noms d'associations qui sont des noms des « couples » liés, des noms de produits cartésiens.

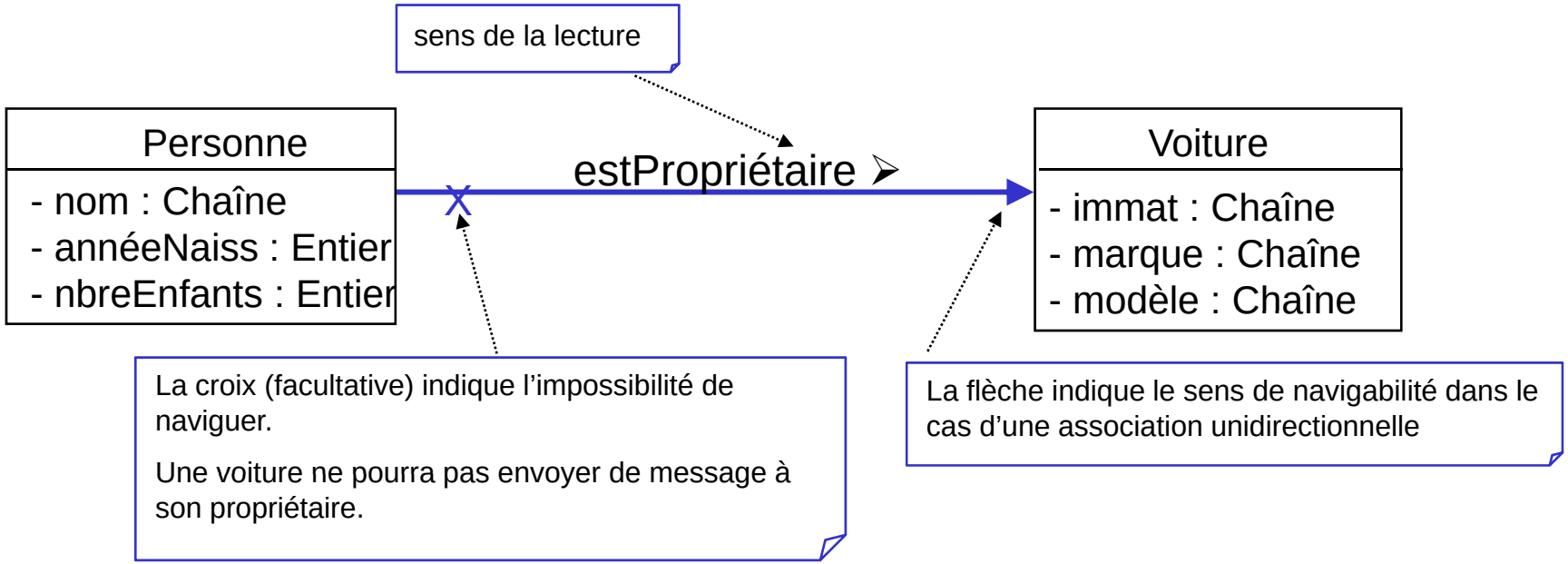
Sens de lecture et navigation

Le sens de lecture est une « décoration » qui indique comment interpréter le nom de l'association.

L'indication de navigabilité d'un rôle exprime l'obligation pour un objet source d'identifier le ou les objets cibles.

Une association est bidirectionnelle quand il y a navigation dans les 2 sens (par défaut).

Une association unidirectionnelle n'est navigable que dans un seul sens.



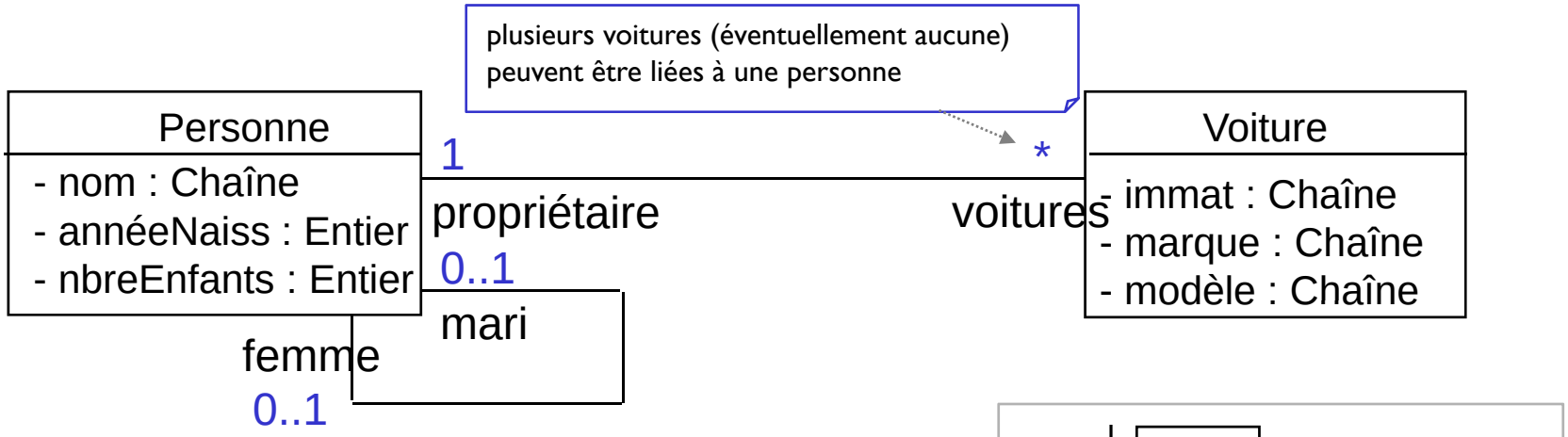
!

Il est recommandé de « différer » l'introduction de sens de lecture et de navigabilités à la conception.

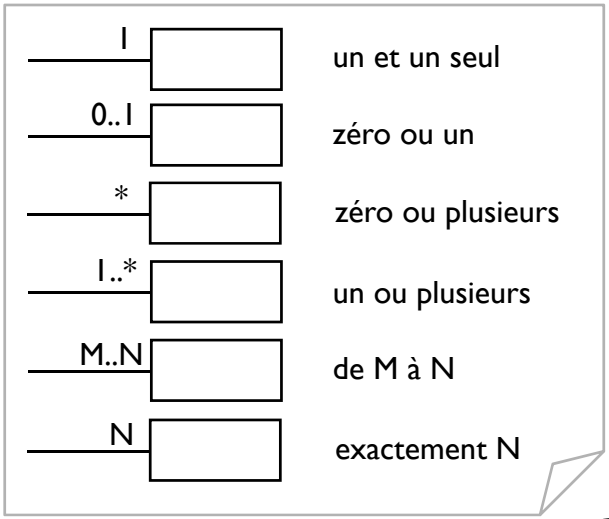
Multiplicité d'un rôle

La multiplicité est portée par le rôle. Elle précise combien d'objets de la classe destination peuvent être liés à un objet de la classe source. On parle aussi de cardinalité du rôle.

La multiplicité est notée du côté de la classe destination.



Notations de la multiplicité



Traduction des rôles

Lors de la réalisation, un rôle navigable est traduit par un attribut.


```
Classe Personne
        Attributs
                nom : Chaine;
                ...
                voitures : Ensemble [Voiture];
        Méthodes
                ajouterVoiture (v:Voiture);
                        voitures.inserer(v);
                        v.affecterProprio (self);
                fin ajouterVoiture;
fin Personne;
```

```
Classe Voiture
        Attributs
                marque : Chaine;
                ...
                propriétaire : Personne;
        Méthodes
                affecterProprio (p:Personne);
                        Propriétaire := p;
                fin affecterProprio;
fin Voiture;
```


```
Classe Personne
        Attributs
                nom : Chaine;
                ...
                voitures : Ensemble [Voiture];
        Méthodes
                ajouterVoiture (v:Voiture);
                        voitures.inserer(v);
                fin ajouterVoiture;
fin Personne;
```

```
Classe Voiture
        Attributs
                marque : Chaine;
                ...
        Méthodes
        ....

fin Voiture;
```

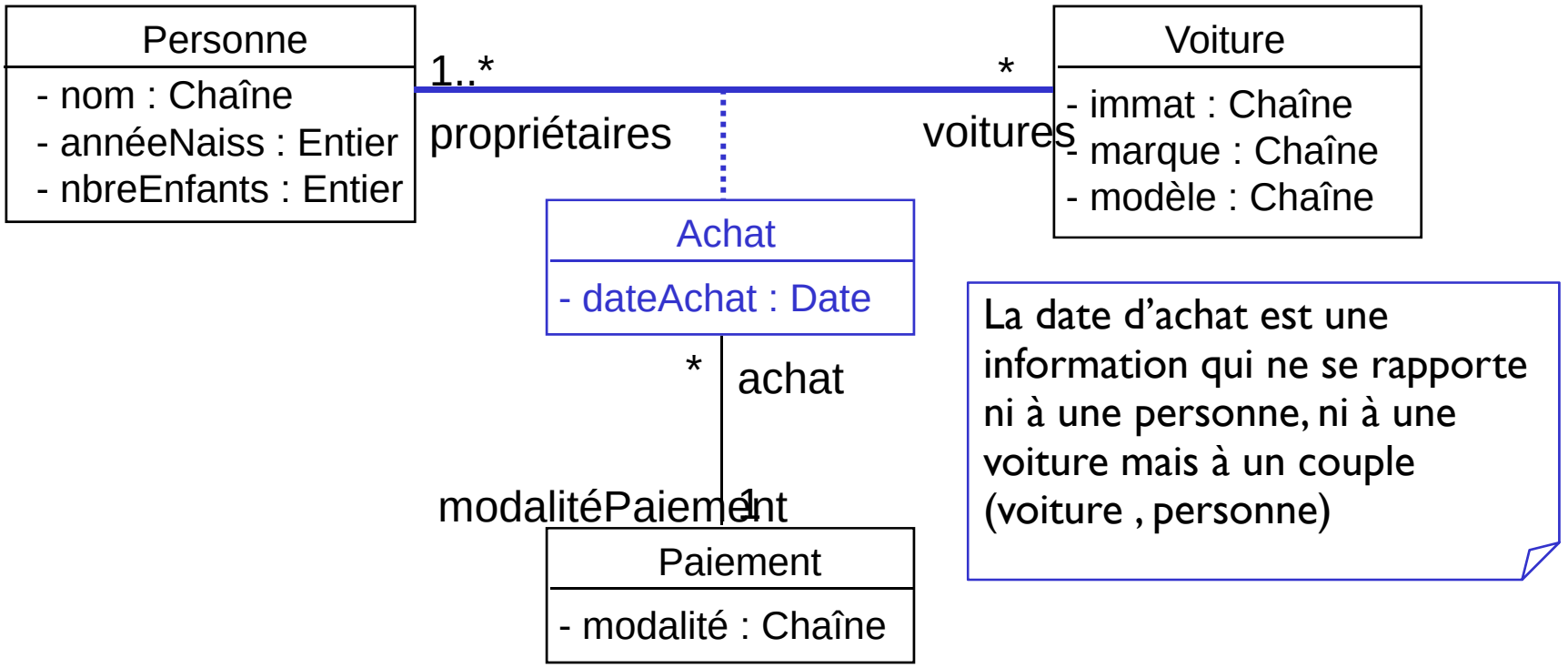
Le rôle inverse n'est pas traduit.

Classe-association ou classe associative

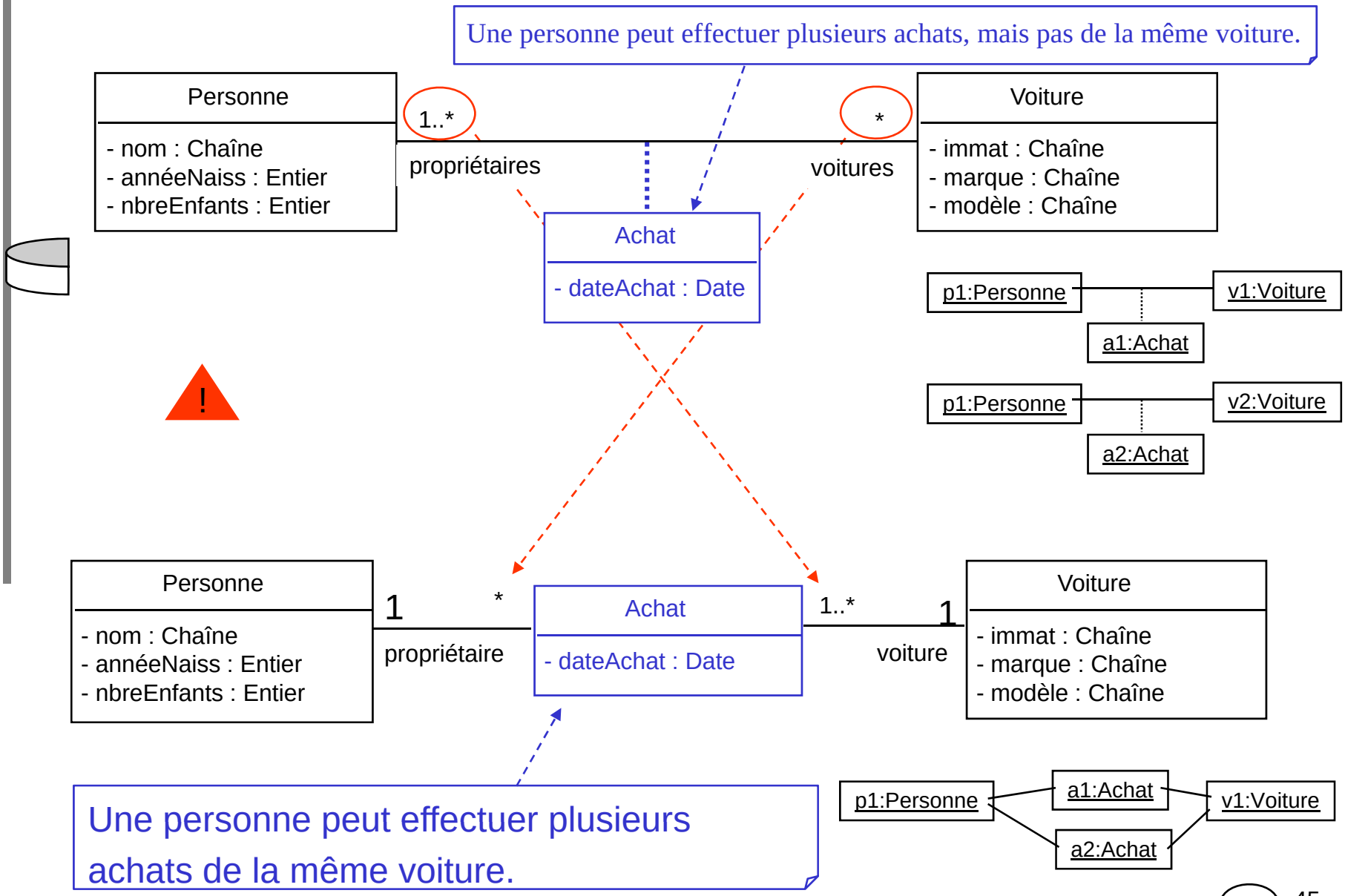
Une association peut posséder des attributs et des opérations.

Elle est alors représentée à l'aide d'une classe-association.

Une classe-association est une classe à part entière et peut donc participer à d'autres associations.

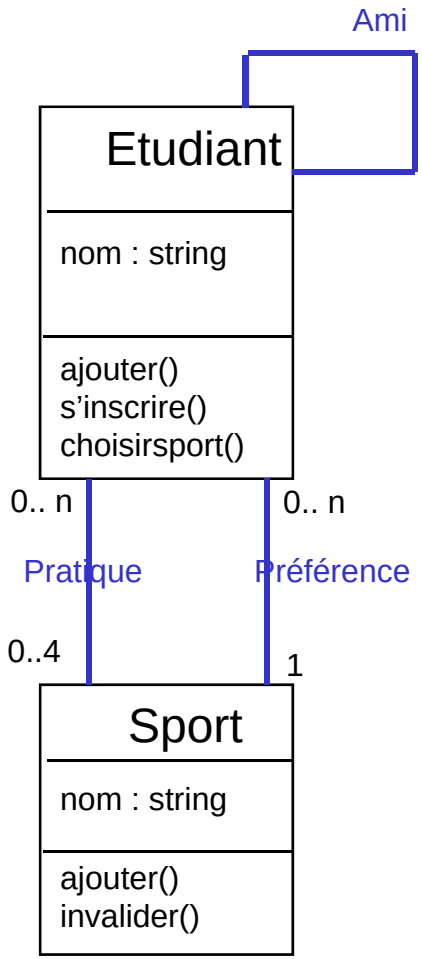


Transformation d'une classe-association



Associations multiples

Plusieurs associations peuvent être définies entre deux classes.
Une association peut être définie sur la même classe.



Agrégation

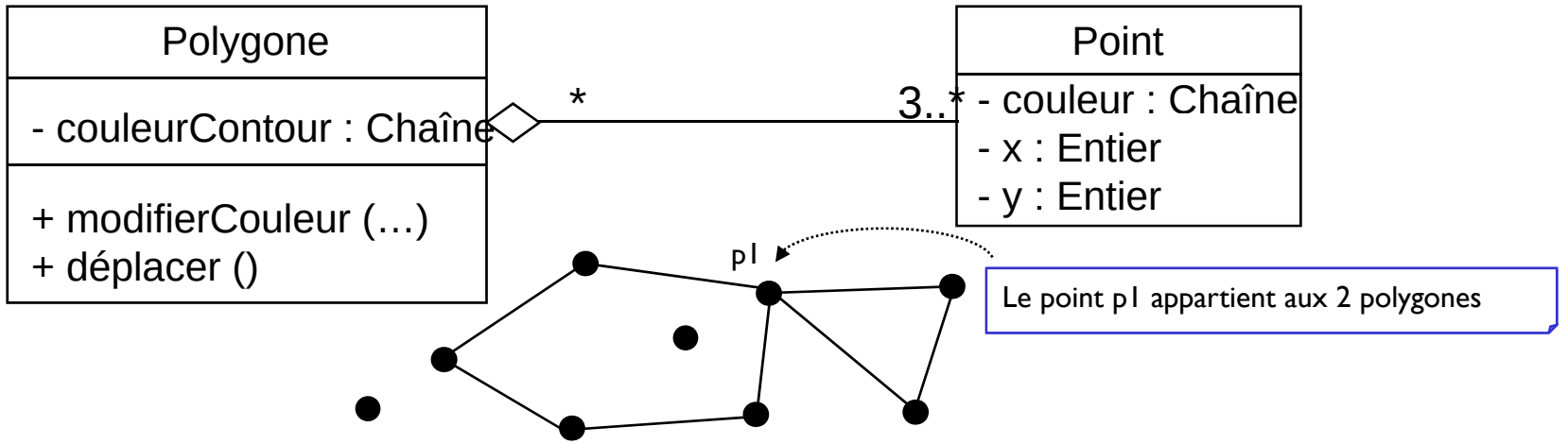
Forme particulière d'association non symétrique dans laquelle l'une des extrémités joue un rôle prédominant par rapport à l'autre.

Les 2 objets liés par une agrégation sont distingués : l'un fait "partie de" l'autre, considéré comme l'objet global (l'agrégat).

Un objet "partie" peut être partagé (pas d'exclusivité).

Reconnaître une agrégation :

- **Des valeurs d'attributs sont-elles propagées de l'agrégat vers les parties ?**
- **Des opérations appliquées à l'agrégat sont-elles automatiquement appliquées aux parties ?**



Composition

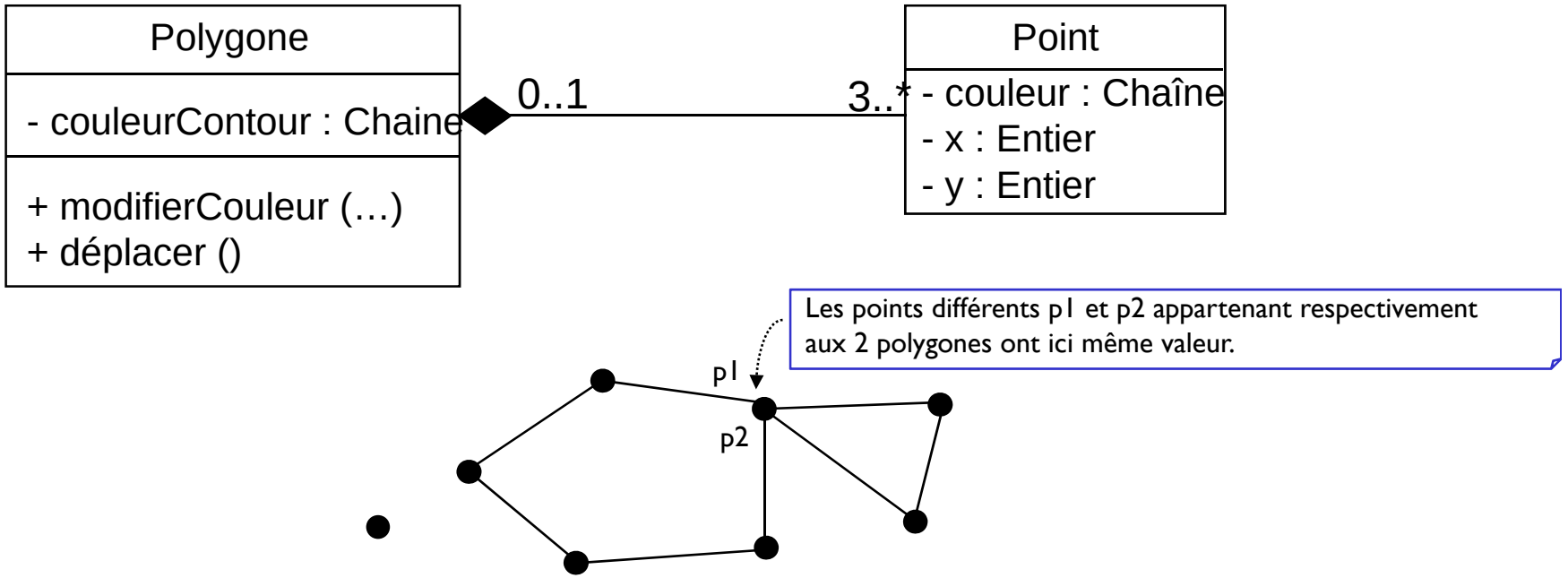
Forme particulière d'agrégation avec une dépendance forte entre composé et composant.

Un objet composant n'appartient qu'à un seul composé (pas de partage).

Le composant est détruit lors de la destruction du composé.

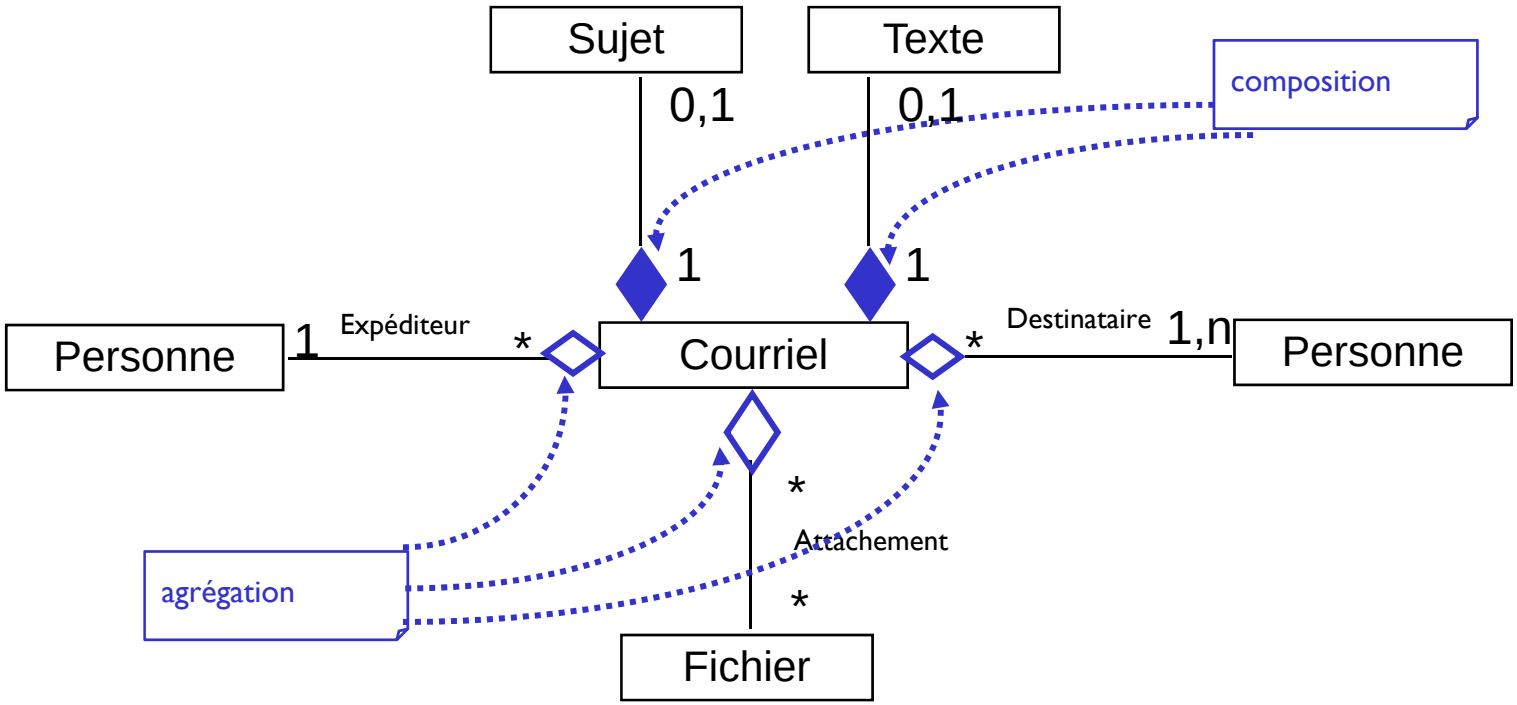
Le composant peut éventuellement être créé hors de son composé.

Le composant n'intervient pas dans d'autres associations de composition ni d'agrégation.

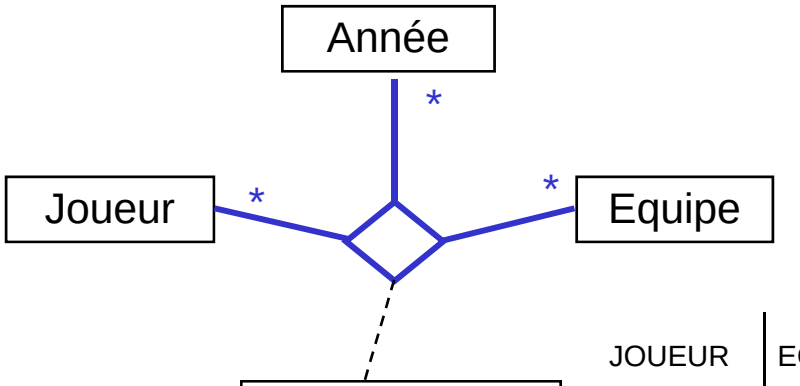


Agrégation & Composition

2 associations « tout-partie » à sémantique différente.



Association ternaire



JOUEUR	EQUIPE	ANNEE	nbreVictoires	nbreDéfaites
dupont	louvetau x	96	6	2
durand	louvetaux	96	3	4
dupont	totoche	96	4	5
dupont	totoche	97	5	3

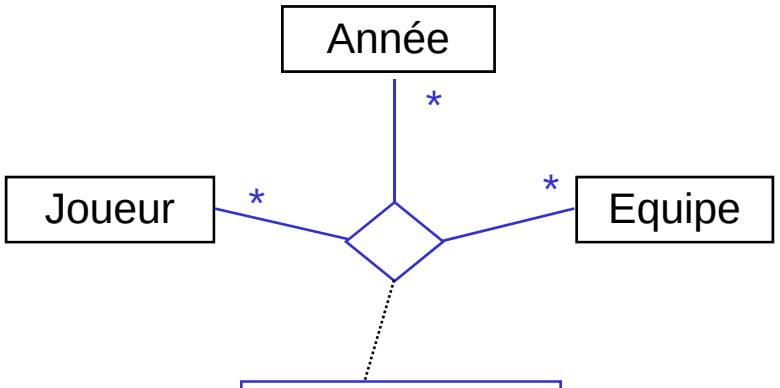
Multiplicités :

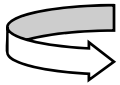
Un joueur, une année donnée, peut jouer dans plusieurs équipes

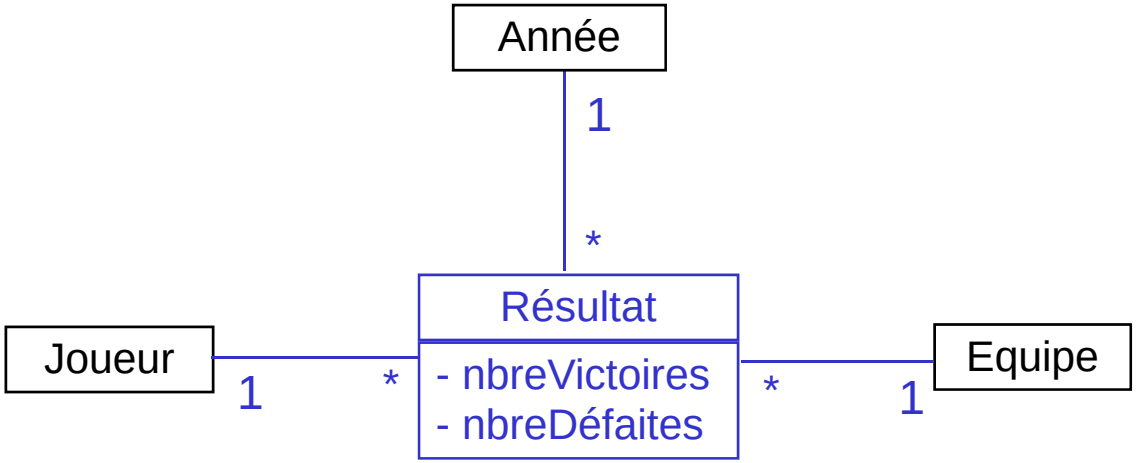
Un joueur peut jouer plusieurs années dans une même équipe

Une équipe, une année donnée, est composée de plusieurs joueurs

Transformation d'une association ternaire



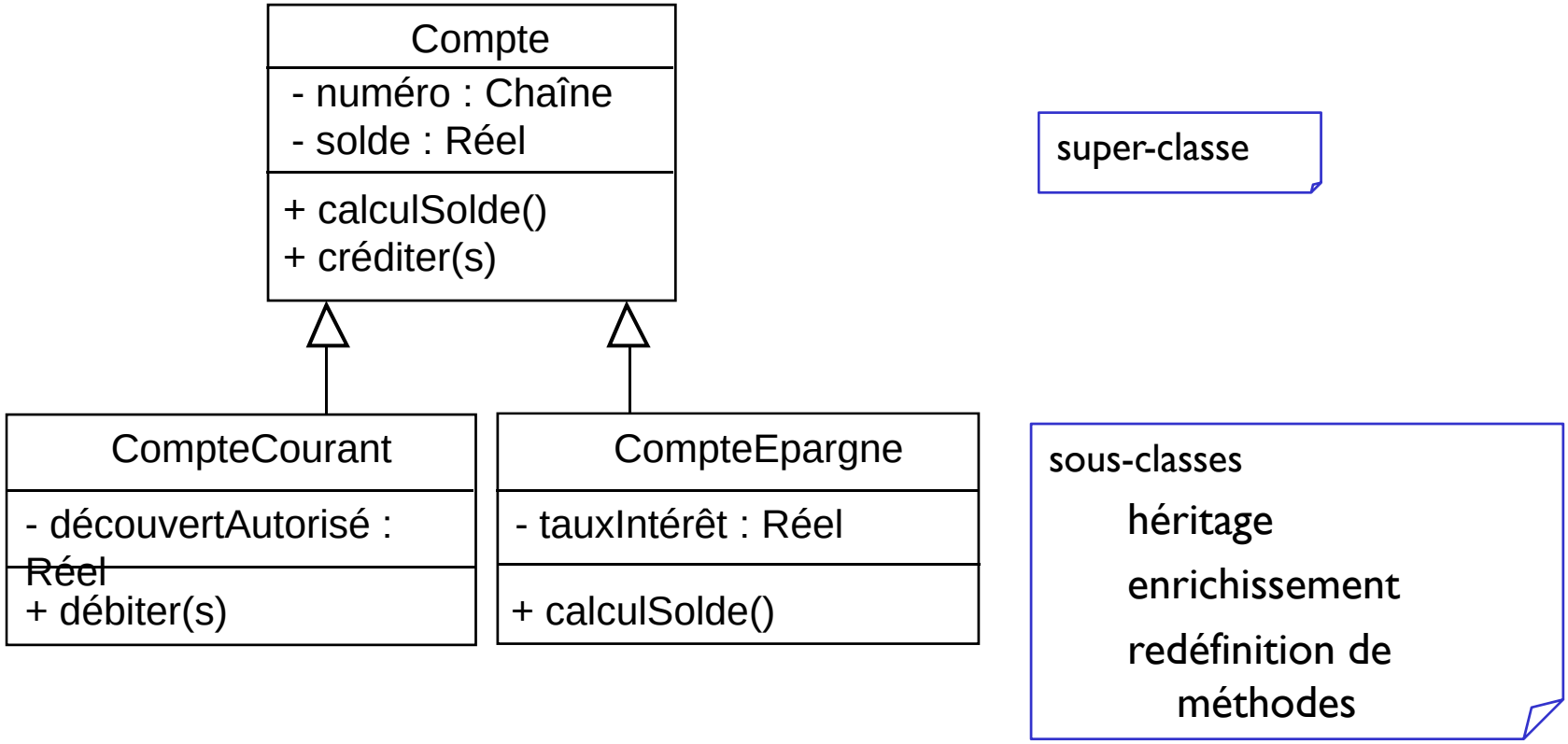




Mais dans le diagramme ci-dessus, un joueur peut avoir plusieurs résultats la même année avec la même équipe.

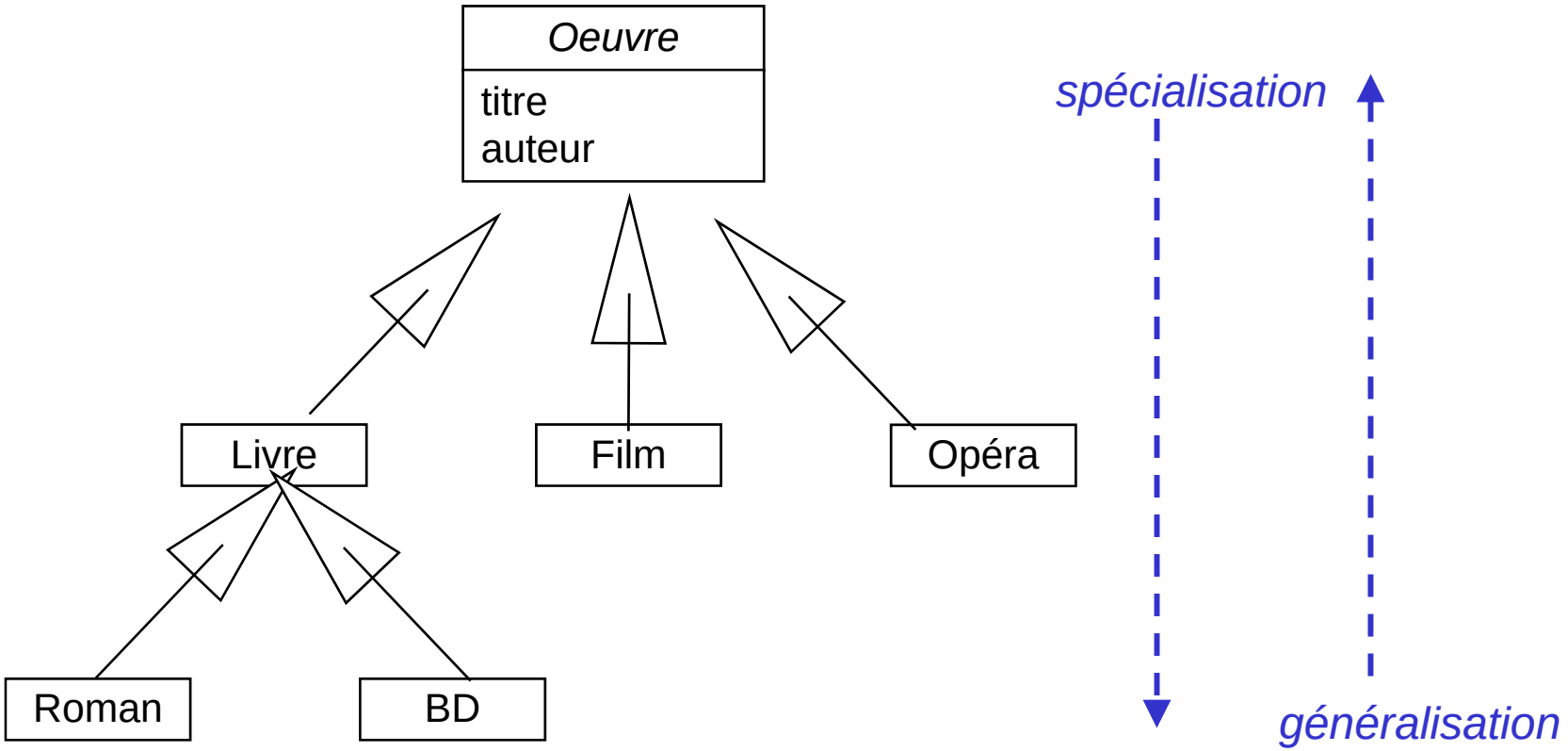
Généralisation-spécialisation

- Une classe peut être spécialisée en d'autres classes, afin d'y ajouter des caractéristiques spécifiques ou d'en adapter certaines.
- Plusieurs classes peuvent être généralisées en une classe qui les factorise, afin de regrouper les caractéristiques communes d'un ensemble de classes.



Généralisation-spécialisation

- La spécialisation et la généralisation permettent de construire des hiérarchies de classes.
- La généralisation et la spécialisation évitent la duplication et encouragent la réutilisation.

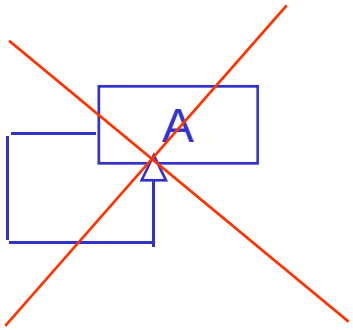


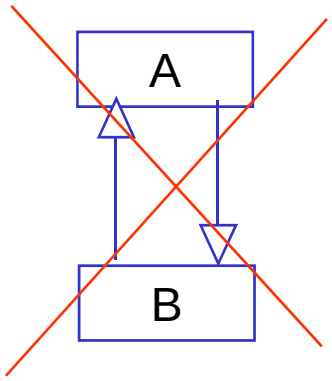
Généralisation-spécialisation

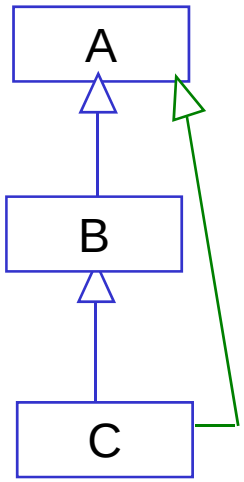
Sens : «est un» , «est une sorte de» , «est de la famille des»

Propriétés :

- non réflexive : A n'hérite pas de lui-même ; il EST lui-même
- non-symétrique : B sous-classe de A, interdit A sous-classe de B (pas de cycle)
- transitive : C sous classe de B, B sous-classe de A==> C sous-classe de A

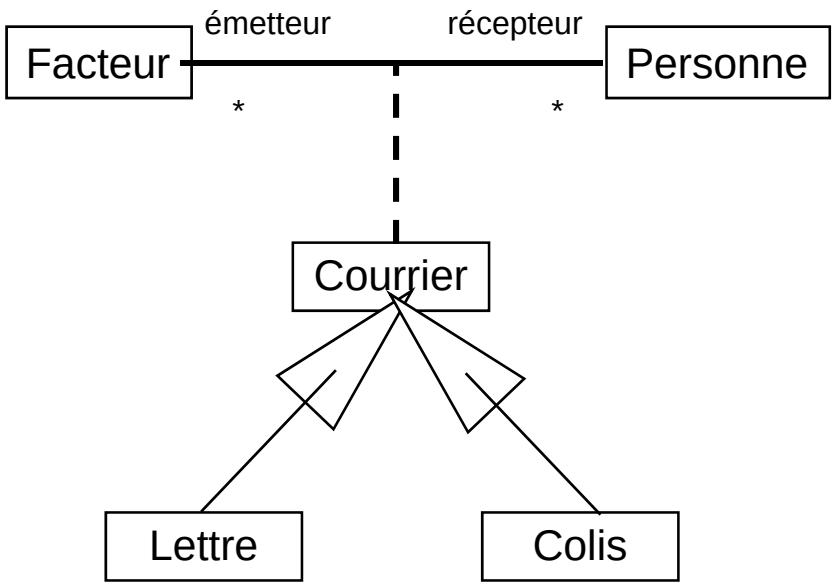






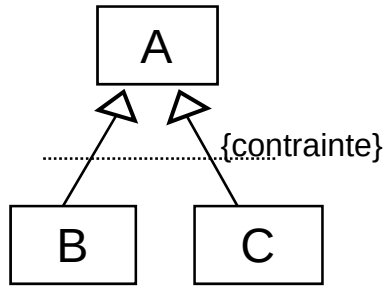
Généralisation-spécialisation

Une classe d'association peut être spécialisée

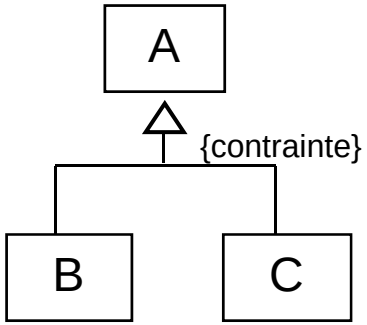


Contraintes sur la généralisation-spécialisation

B spécialise A : * les objets de B sont implicitement des objets de A
* les propriétés de A s'appliquent aux objets de B



Autre notation (en « ratelier »)



{disjoint} ou {exclusif} :

un objet de A est instance d'au plus une sous-classe

conséquence : B et C n'ont aucune sous-classe commune

{chevauchement} ou {inclusif} :

un objet de A peut être instance de plusieurs sous-classes

conséquence : B et C peuvent avoir une sous-classe commune

{complète} :

un objet de A est instance d'au moins une sous-classe

conséquence : la classification de A est terminée

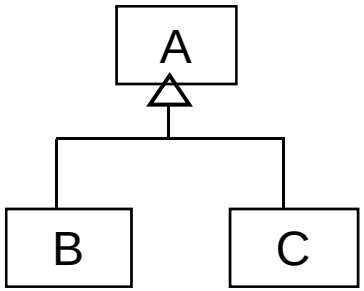
{incomplète} :

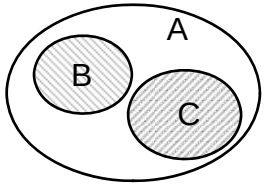
un objet de A peut n'être instance d'aucune sous-classe

conséquence : la classification de A est extensible

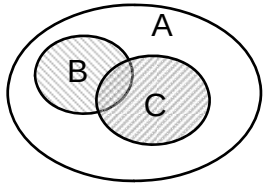
Contraintes sur la généralisation-spécialisation

Cas possibles

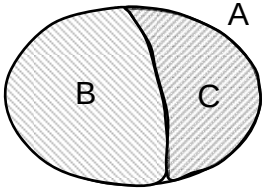




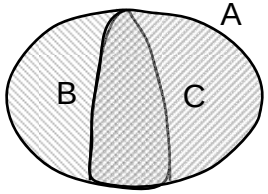
{disjoint, incomplète}



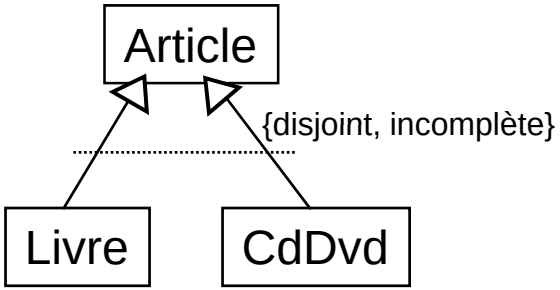
{chevauchement, incomplète}

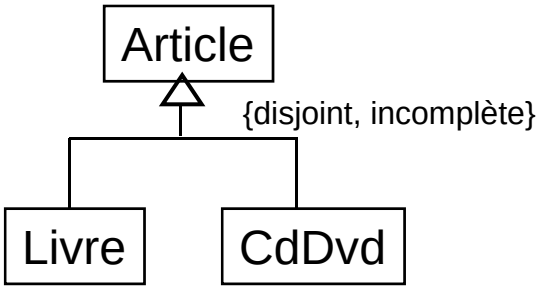


{disjoint, complète}



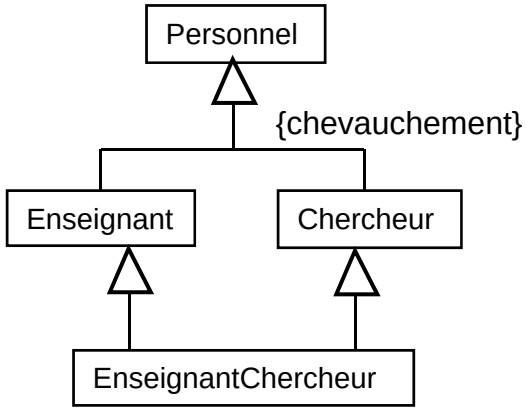
{chevauchement, complète}



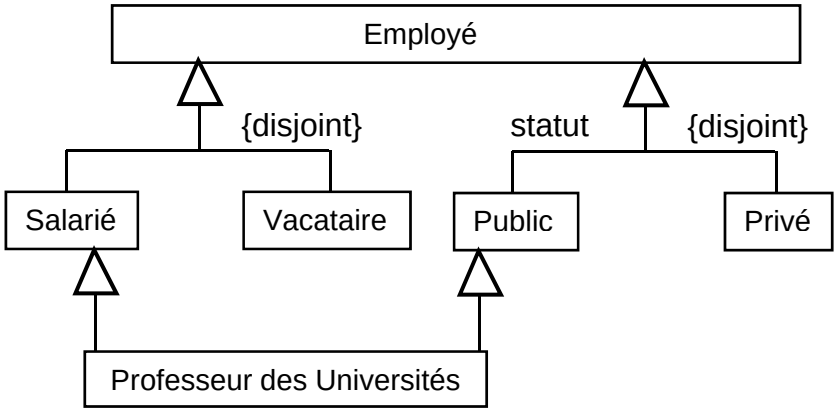


Spécialisation multiple

- **Dans une relation de spécialisation avec chevauchement**



- Dans plusieurs branches de spécialisation selon des critères différents



Contraintes sur les classes

- **Contrainte sur une classe**

  {abstraite} : classe ne pouvant pas être instanciée *(concrète par défaut)*

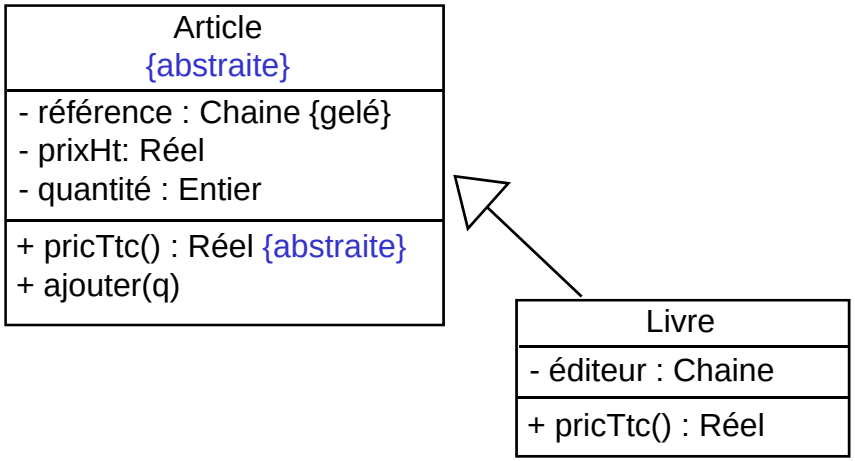
- **Contraintes sur un attribut**

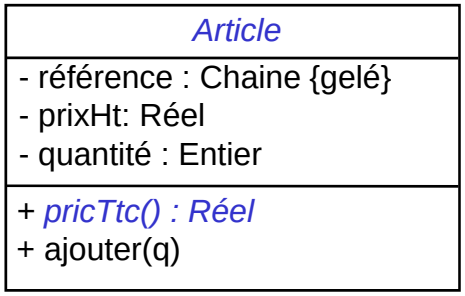
  {gelé} : mise à jour interdite

- **Contraintes sur une opération**

  {abstraite} : opération non implémentée. L'implémentation devra être réalisée par une sous-classe concrète.

  {est-feuille} : redéfinition interdite





autre notation pour classe et opération abstraites (en italique)

Contraintes sur les classes

- **Contrainte sur une classe**

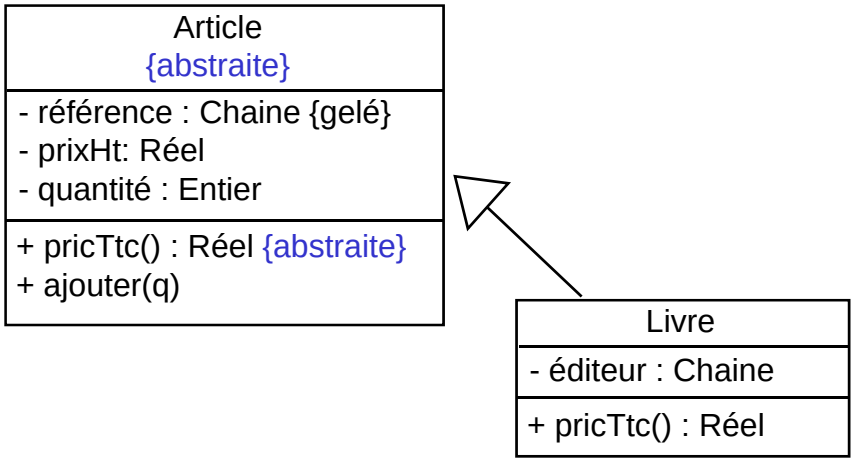
  {abstraite} : classe ne pouvant pas être instanciée *(concrète par défaut)*

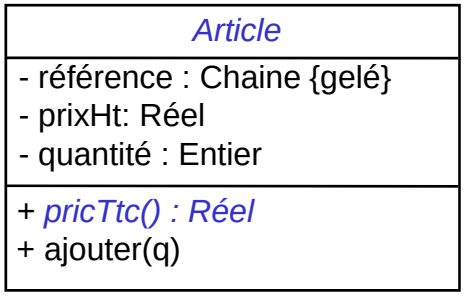
- **Contraintes sur un attribut**

  {gelé} : mise à jour interdite

- **Contraintes sur une opération**

  {abstraite} : opération non implémentée. L'implémentation devra être réalisée par une sous-classe concrète.

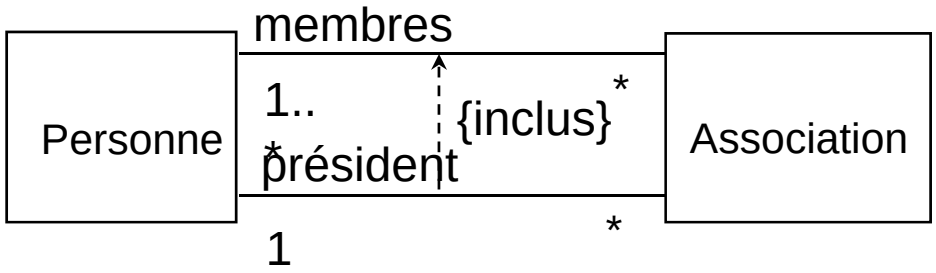




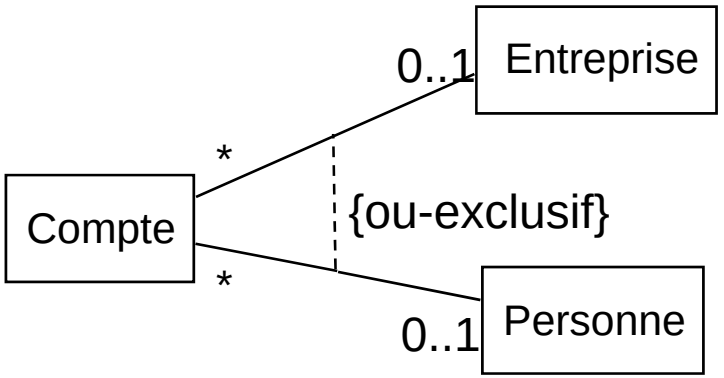
autre notation pour classe et opération abstraites (en italique)

Contraintes sur les associations

- **Contraintes inter-liens : inclusion, exclusion, totalité**



Le président d'une association fait partie des membres de cette association.



Un compte est soit un compte entreprise soit un compte individuel.

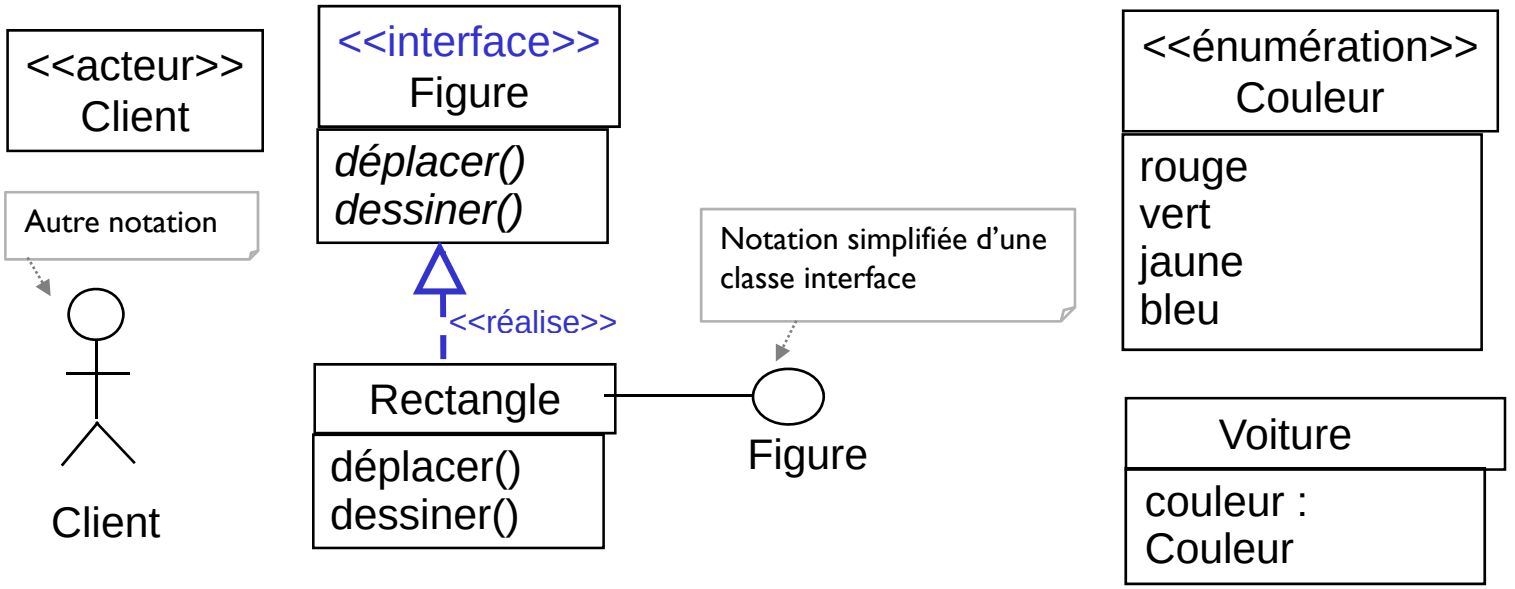
Stéréotype

- **Un stéréotype permet d'apposer une sémantique particulière aux éléments de modélisation**
- **Quelques stéréotypes de classe prédéfinis :**

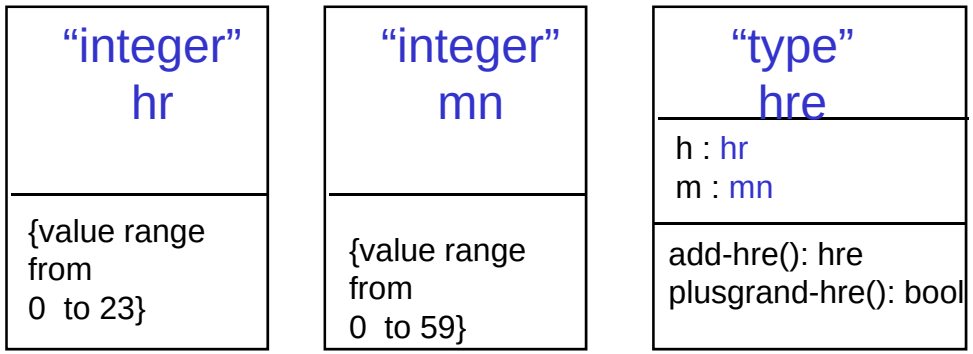
  <<acteur>> : classe modélisant un ensemble de rôles joués par un acteur

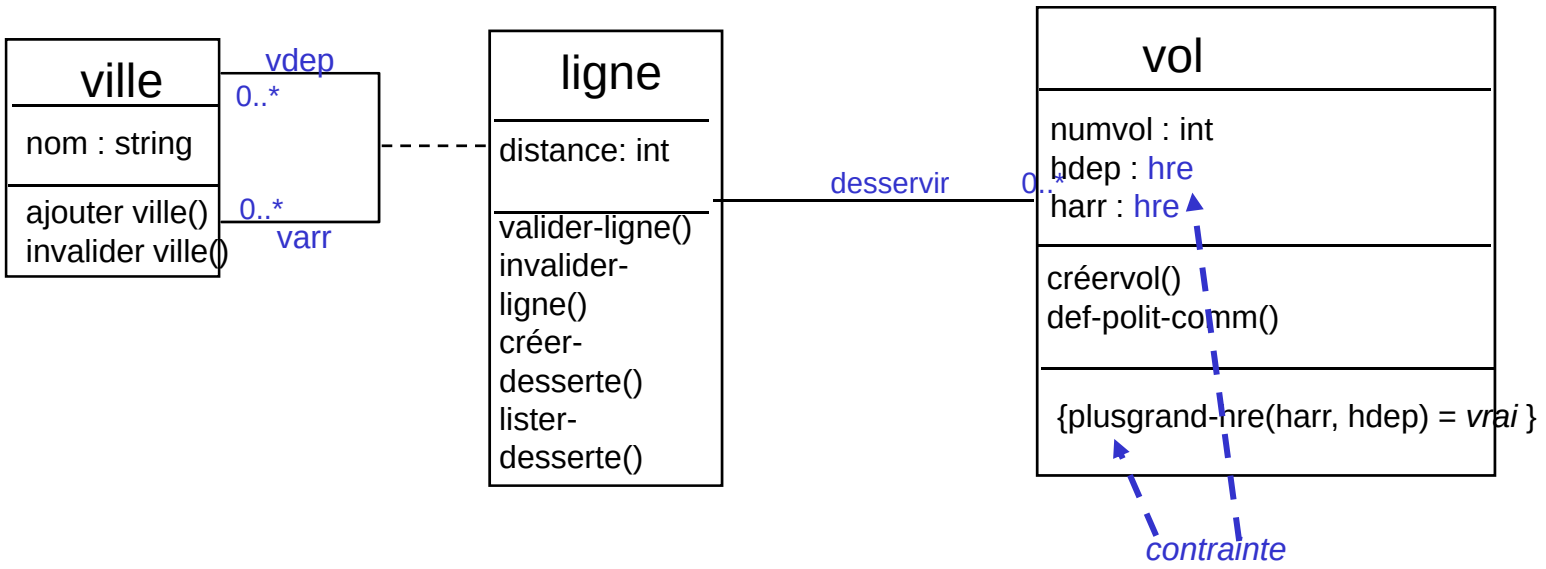
  <<interface>> : classe contenant uniquement les opérations publiques abstraites qu'une classe concrète doit implémenter si elle veut disposer de cette interface

  <<énumération>> : classe définissant en extension l'ensemble des valeurs d'un type



Stéréotypes et contraintes





Diagrammes de classes

Classe, objet, association, lien, rôle, multiplicité, navigabilité, visibilité, agrégation, composition, spécialisation, généralisation, contrainte, stéréotype