UML
(Diagramme de classes)

Unified Modeling Language

Sommaire

Diagramme de classes

- Apport en grande partie de la méthode OMT (Rumbaugh)
- S'apparente à un **diagramme entité-association** (MERISE). Il présente les différents objets (classes) du système ainsi que les liens entre ces objets (associations)
- Diagramme **le plus important** dans une modélisation objet

Diagramme de classes

Objectifs

- Déterminer les **données** qui seront manipulées par le système
  Ces données sont organisées en classes

- Donner la **structure statique** de ces données
  Ce diagramme permet de décrire la structure interne de chacune des classes

- Représenter les **relations statiques** existant entre les différentes données du système
  La navigation parmi les classes est rendue possible par l'existence d'associations qui les unissent

Objectifs (suite)

- Poser les **fondements stables** régissant la totalité de l'architecture du système

  Ce modèle est le garant du respect du paradigme objet

- Faire abstraction des aspects temporels et dynamiques de la modélisation

  Seul **l'aspect statique** compte, la dynamique est prise en charge par d'autres modèles

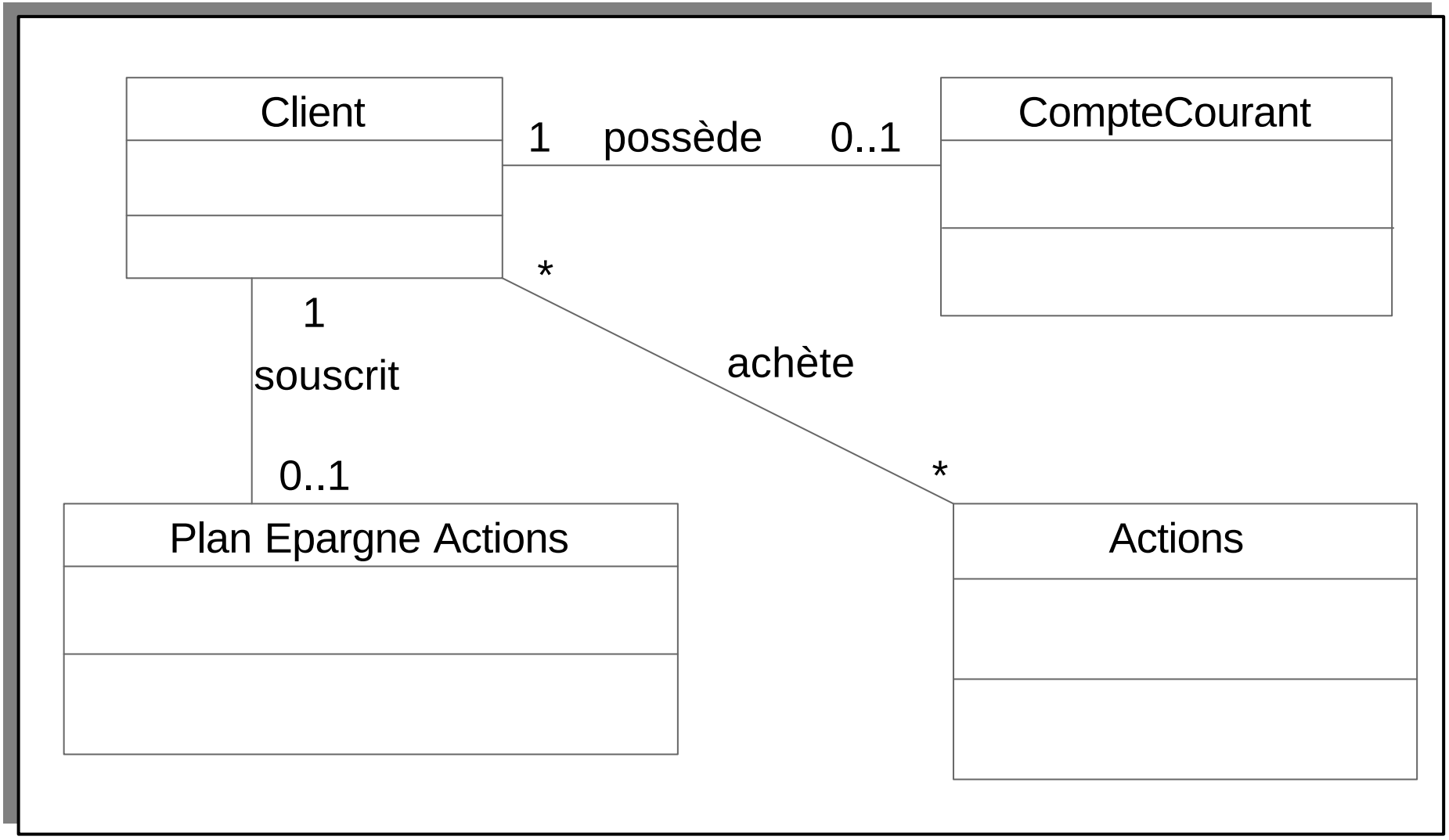
Diagramme de classes (Définition)

Le diagramme de classes est un **diagramme entités-associations** décrivant les différentes classes, leur structure et les associations **statiques** les unissant

- Le diagramme de classes est un diagramme structurel ne présentant que les classes et pas les instances de classe
- Il permet de décrire la structure interne des classes en terme **d'attributs et d'opérations**
- Il permet de représenter les **associations statiques** entre les classes, mais ne décrit pas comment les liens effectifs entre les instances sont effectués

Diagramme de classes

Exemple



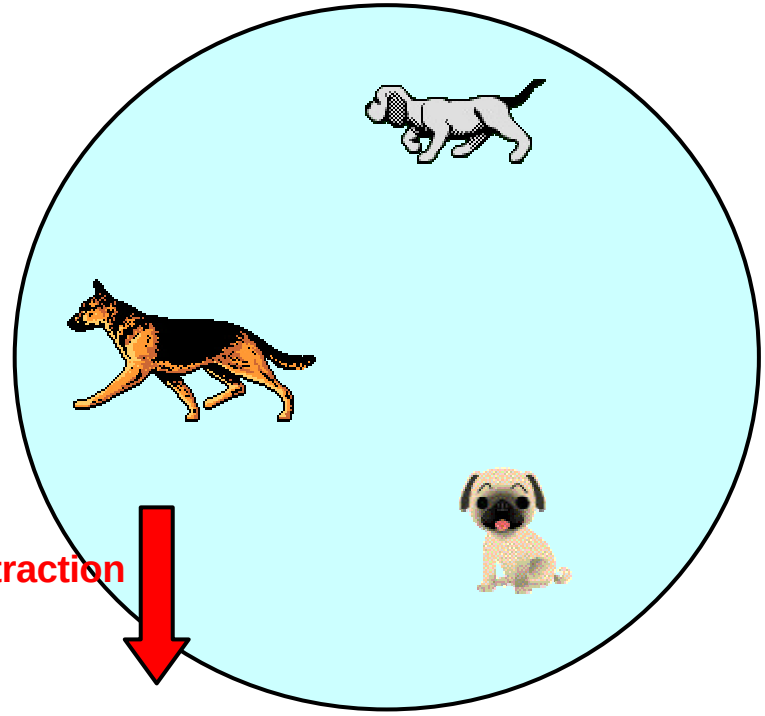
Classe
(Définition)

Une classe est une **abstraction** de choses
du monde réel possédant des **caractéristiques** et
des **comportements communs**

- La classe est la fabrique, le moule, à partir duquel on fabrique les **instances** (les objets)

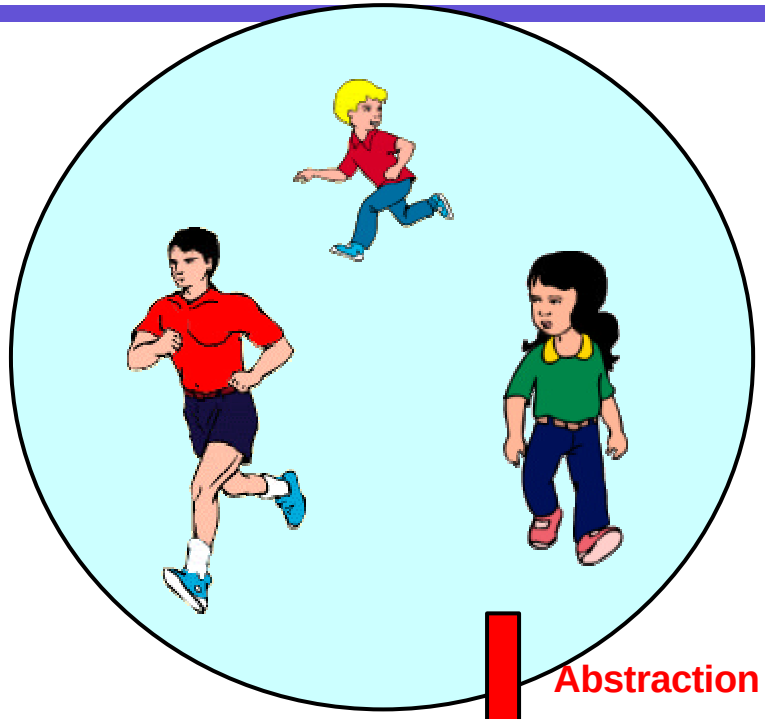
- Seules les **caractéristiques pertinentes** pour le problème étudié entrent dans la composition de la classe

Classe



Abstraction

Chien
age pedigree
courir() aboyer()



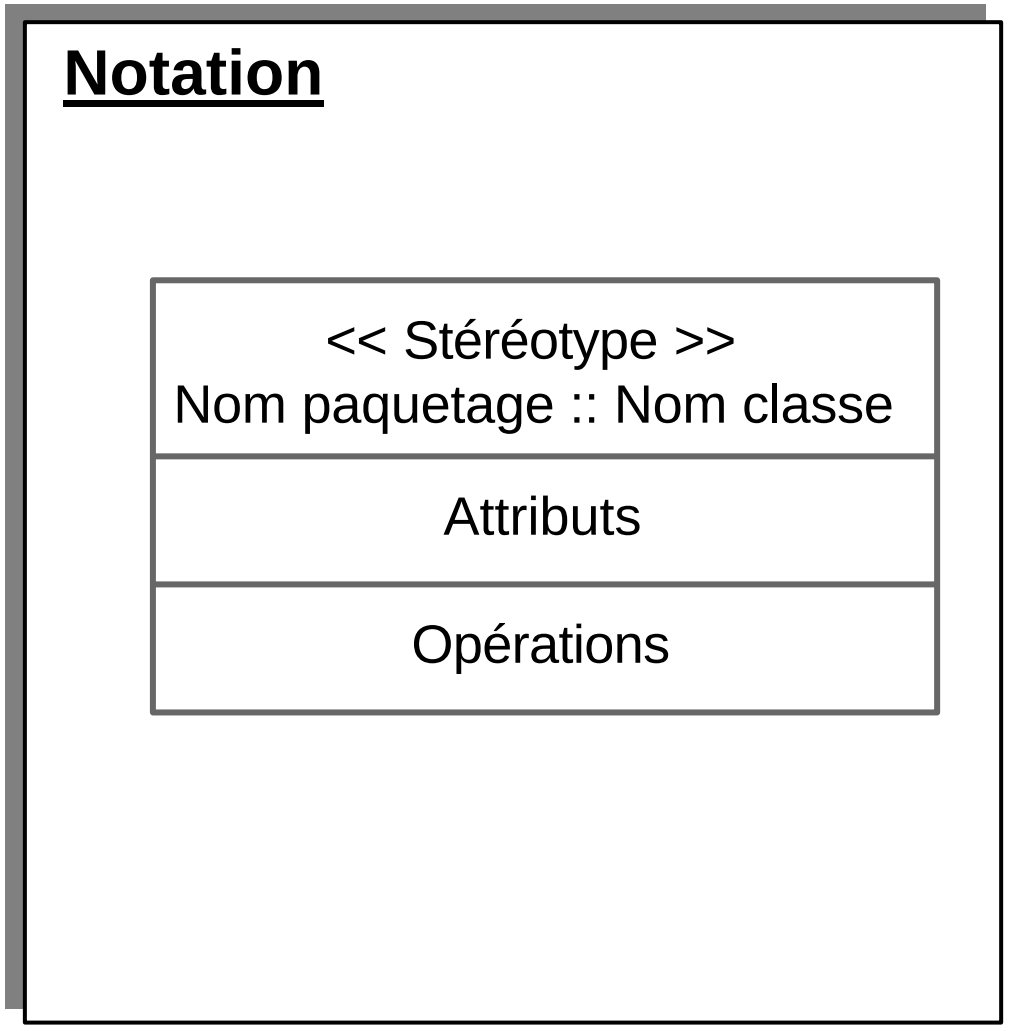
Abstraction

Personne
age nationalité
se promener() crier()

Classe
(Notation)

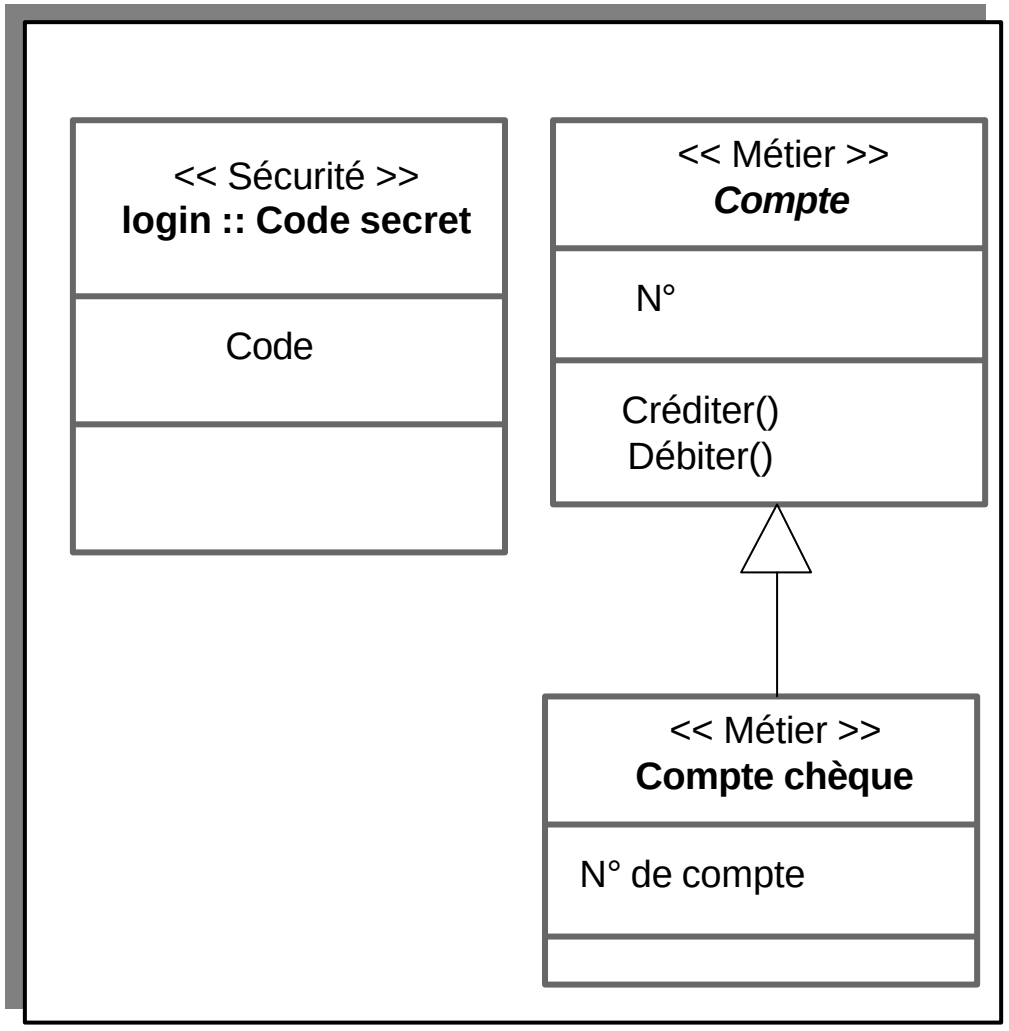
- Une classe est représentée par un rectangle découpé en 3 parties

- Sont présents :
  - le nom de la classe
  - la liste de ses attributs
  - la liste de ses opérations



Nom de la classe

- Le nom de la classe peut être précédé d'un stéréotype qualifiant le type de la classe
- Le nom de la classe est préfixé par un nom de paquetage si la classe est **externe** au paquetage courant
- Le nom d'une classe **abstraite** est donné en *Italique*

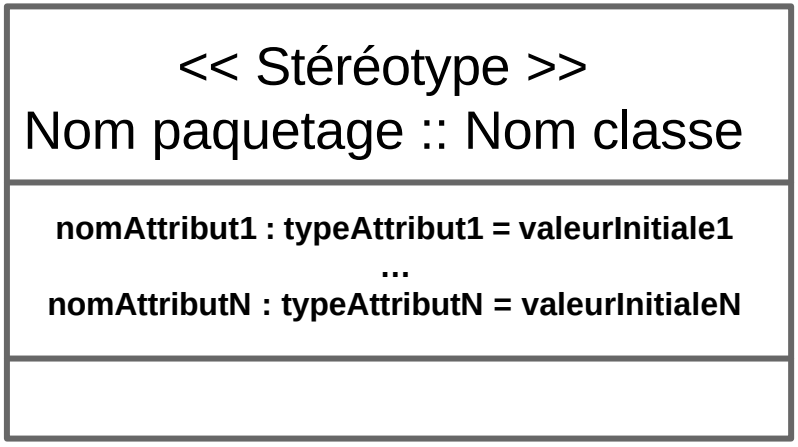


Attribut de la classe (Définition et notation)

Un attribut est une caractéristique **intrinsèque** partagée par tous les objets d'une classe

- L'attribut possède un **nom unique** dans la classe
- On peut associer à l'attribut le **type** des valeurs qu'il peut prendre
- On peut donner une **valeur initiale** à l'attribut

Notation



Attribut de la classe

Recommandations pour trouver les attributs

Les 3 règles d'or de l'attribut :

- Eliminer les attributs caractérisant une autre classe

  ex : l'attribut « nom client » dans la classe « compteBancaire »

- Se méfier des **attributs multi-valués**, ils cachent souvent eux-mêmes une classe

  ex : l'attribut « enfants » dans la classe « Personne »

- Se méfier des **attributs structurés**, ils cachent souvent eux-mêmes une classe

  ex : l'attribut « Adresse » dans une classe « Personne »

Attribut de la classe

Recommandations pour trouver les attributs

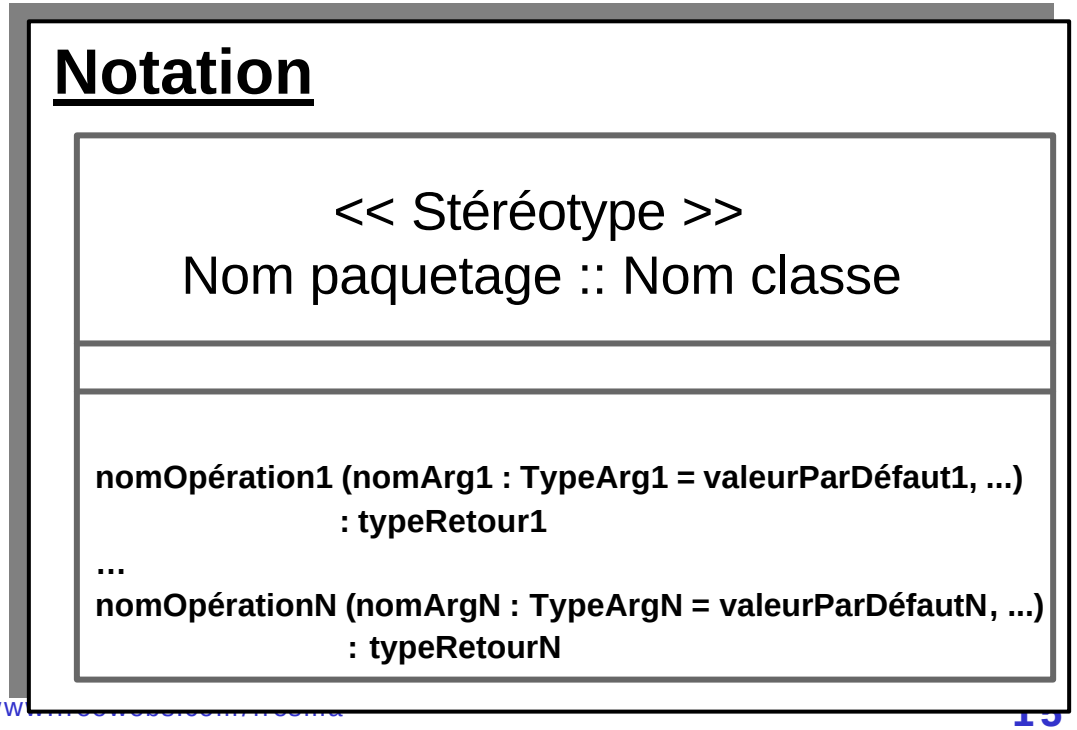
- Ne donner les types et les valeurs initiales des attributs qu'en phase de **Conception**

- Lorsqu'une classe possède de très nombreux attributs se poser la question du **découpage** de la classe

Opération de la classe (Définition et notation)

Une opération est un **service** que propose une classe sur son **interface**

- L'opération possède un nom **pas forcément unique** dans la classe
- On peut associer à l'opération ses **arguments**
- On peut associer à l'opération son **type de retour**



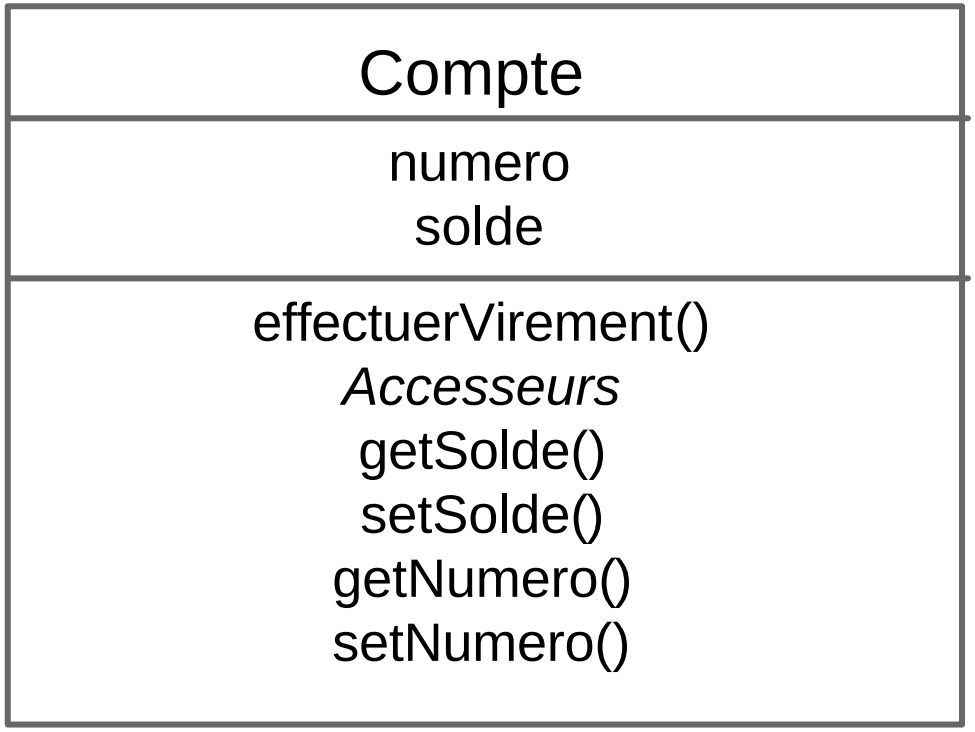
Opération de la classe

Recommandations pour trouver les opérations

- Ne donner les informations sur les arguments et le type de retour qu'en phase de **Conception**

- Lorsqu'une classe possède de très nombreuses opérations se poser la question du **découpage** de la classe

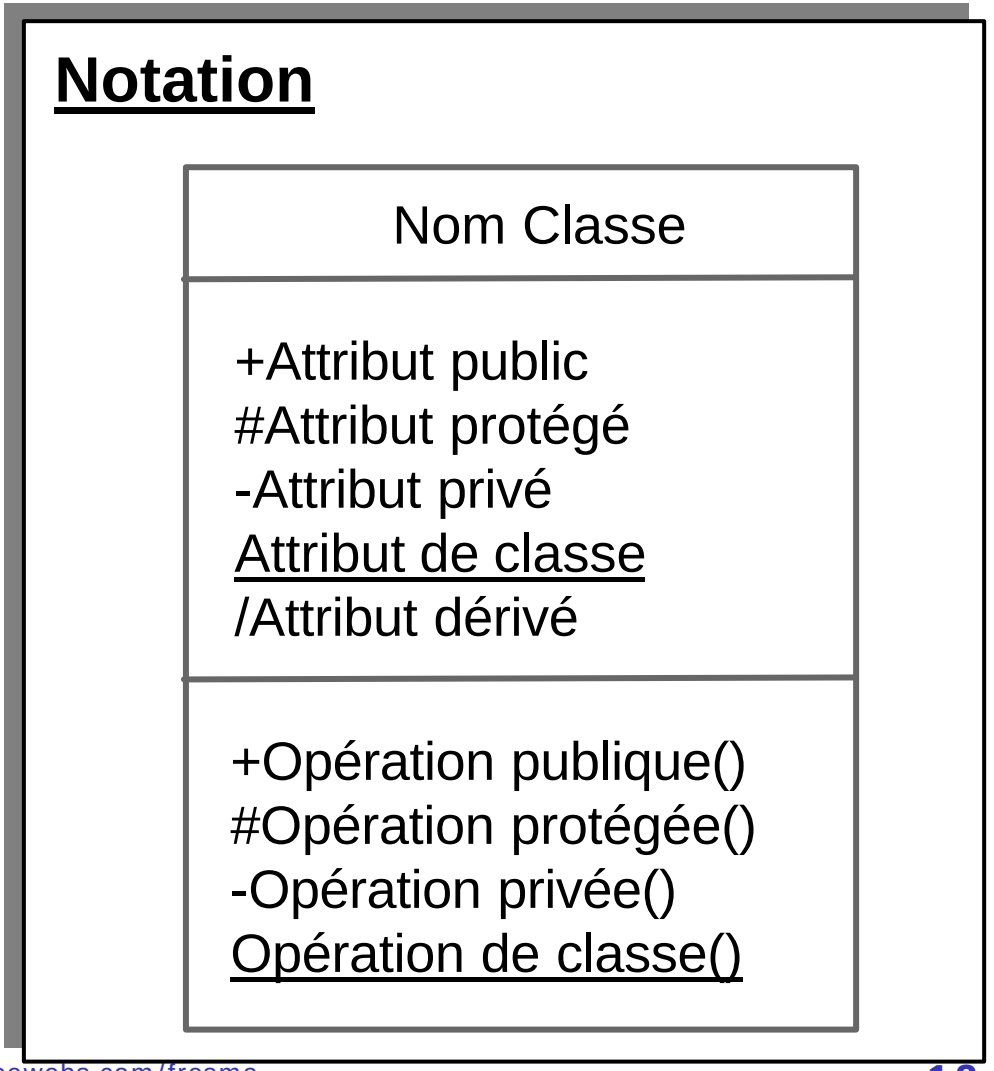
Exemple de classe



Visibilité et Portée des constituants de la classe

- La **visibilité** précise la manière dont un nom peut être vu et utilisé par les autres (public, protégé, privé)
- La **portée** précise dans quel contexte un nom prend sa signification (instance ou classe)
- Par défaut, la visibilité est publique et la portée est d'instance

Notation

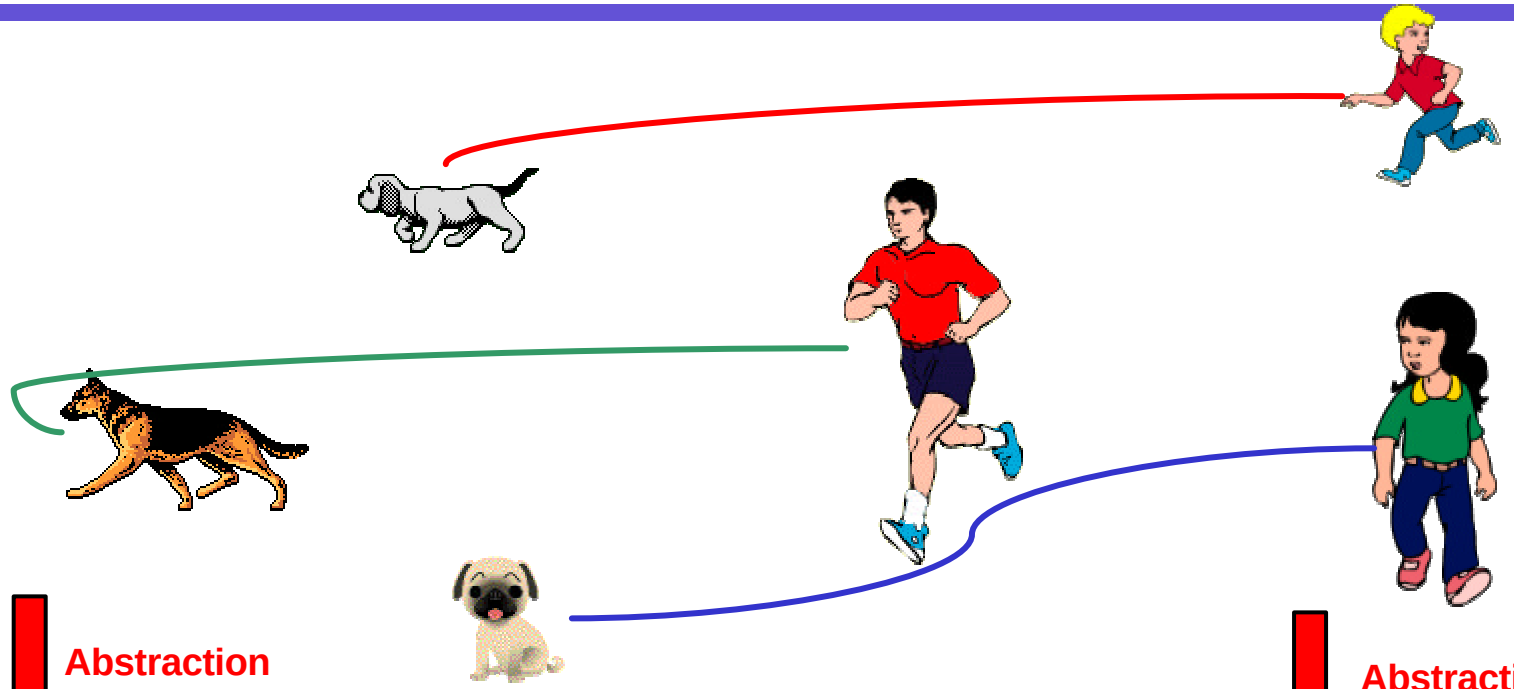


Association (Définition)

Une association est une **abstraction de liens** qui peuvent exister entre les instances de plusieurs classes

- Dans le monde réel, les objets sont **liés** physiquement ou fonctionnellement les uns avec les autres
- Ces liens entre objets se traduisent au niveau des classes par des **associations**
- Une association traduit donc une relation structurelle **statique** entre deux ou plusieurs classes

Association



Abstraction

Abstraction

Abstraction

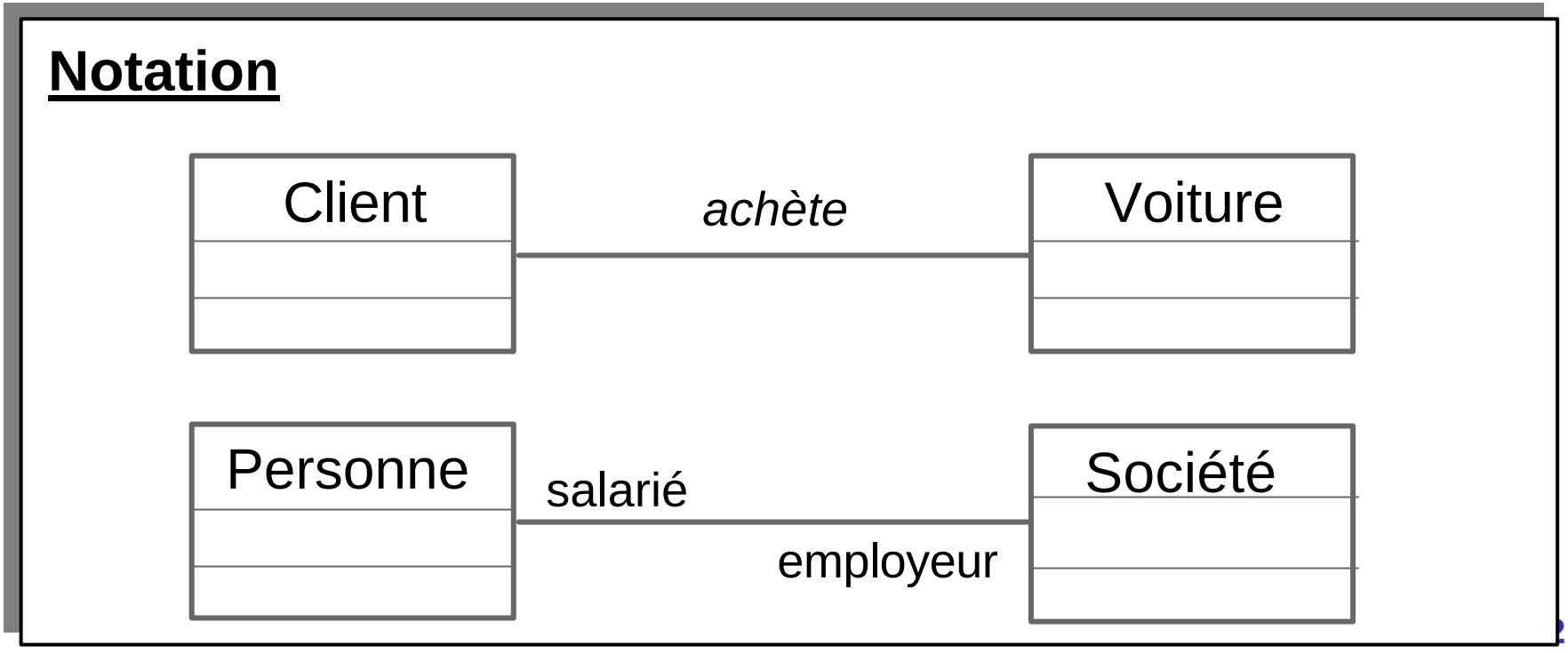
Chien
âge pedigree
courir() aboyer()

Personne
âge nationalité
se promener() crier()

Association (Notation)

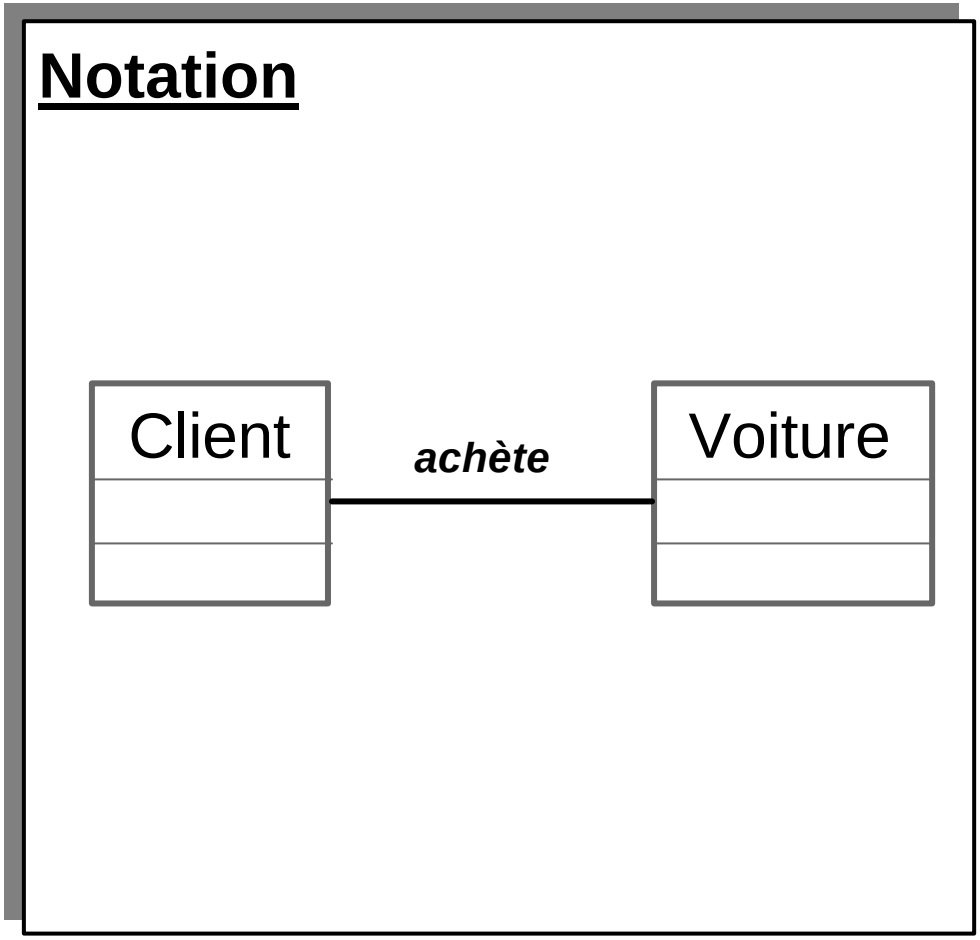
- Une association est représentée au moyen d'un trait orienté reliant chacune des classes concernées

- Il est possible de **nommer** l'association et de préciser les **rôles** tenus par chaque classe



Nom de l’association

- Le nom de l’association est en général une **forme verbale** active ou passive qui décrit globalement le lien
- Le nom de l’association est **facultatif**
- Le nom doit apparaître sur l’association, mais ne doit pas être rattaché à l’une des extrémité



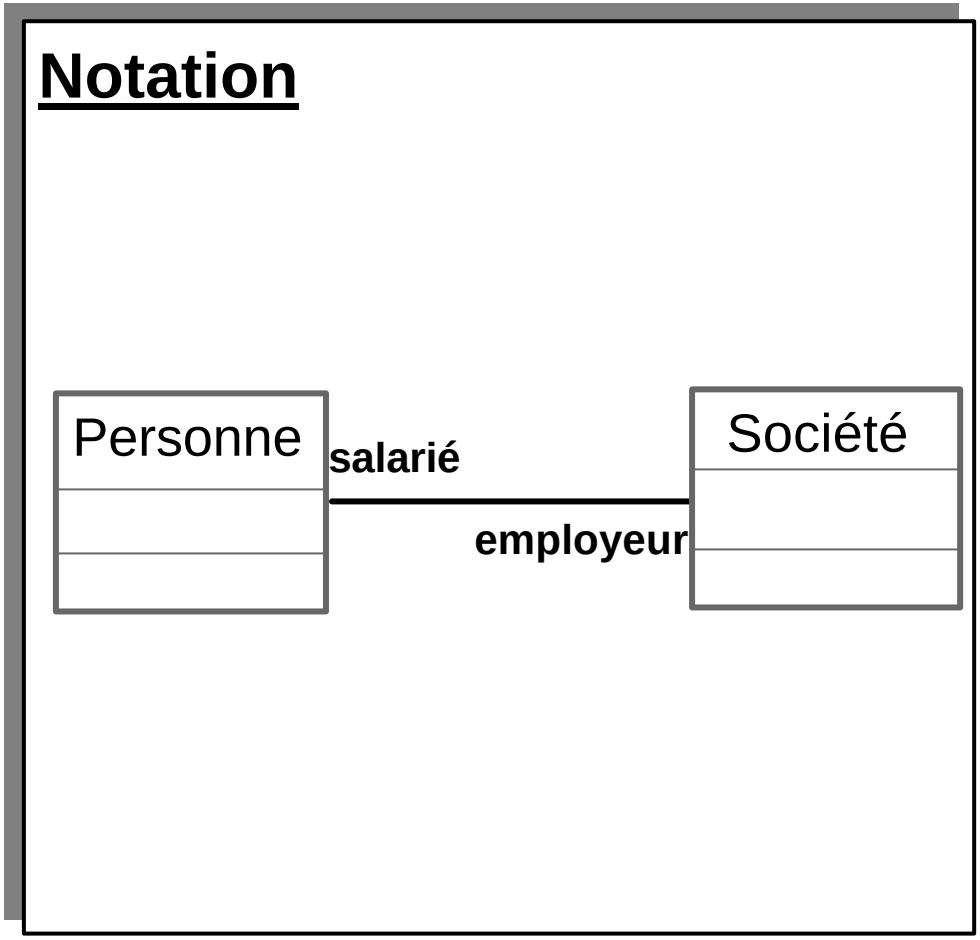
Rôles de l’association

- Le rôle permet de décrire, à l’aide d’un nom, comment une classe perçoit une autre classe au travers de l’association

- Un rôle doit figurer à **l’extrémité** de l’association qu’il qualifie

- Les rôles sont **facultatifs**

  L’association peut faire figurer les deux, un seul ou aucun des rôles



Nom et rôles de la classe (Recommandations)

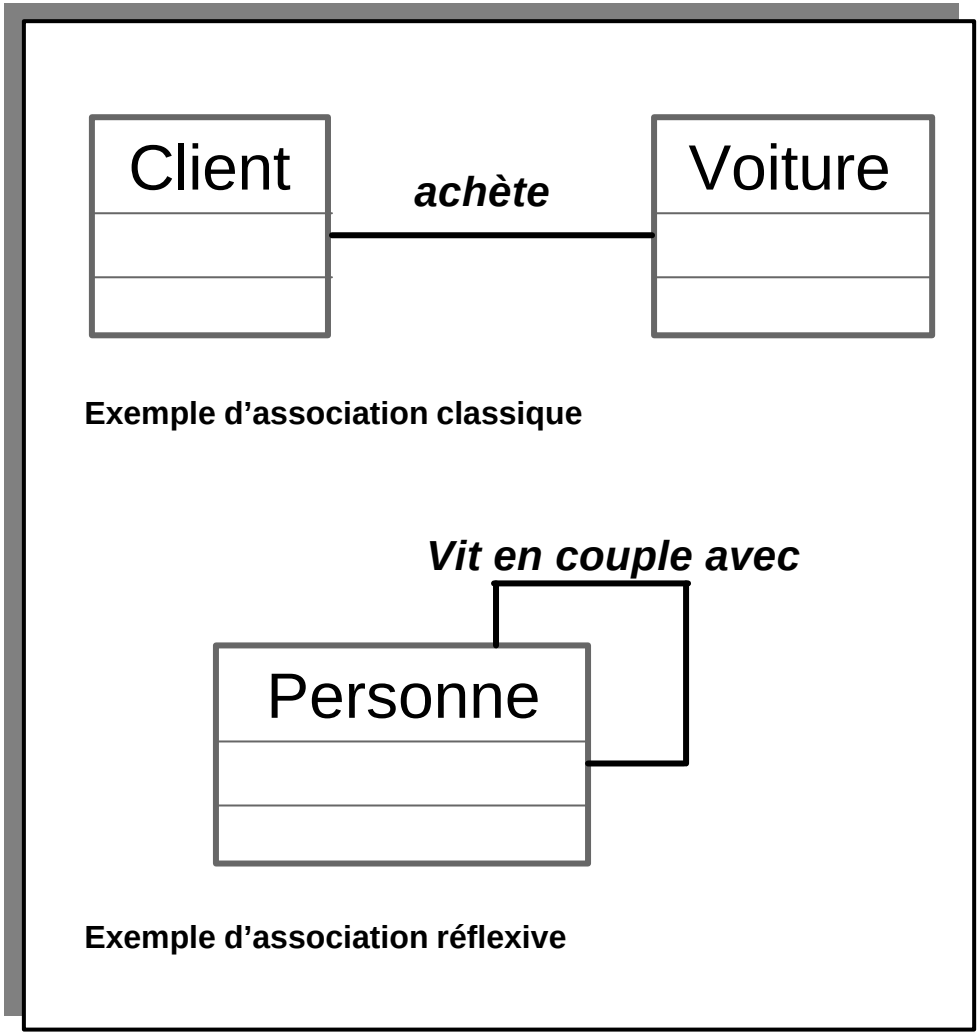
- L'utilisation du nom et des rôles d'une association n'est **pas exclusif**
- Cependant, les deux notations sont rarement utilisées conjointement
- L'usage du nom et des rôles n'est **pas obligatoire**, mais il s'avère indispensable si deux objets sont reliés par plusieurs associations ou si une association est réflexive

Association réflexive

- Une association peut mettre en jeux deux classes distinctes

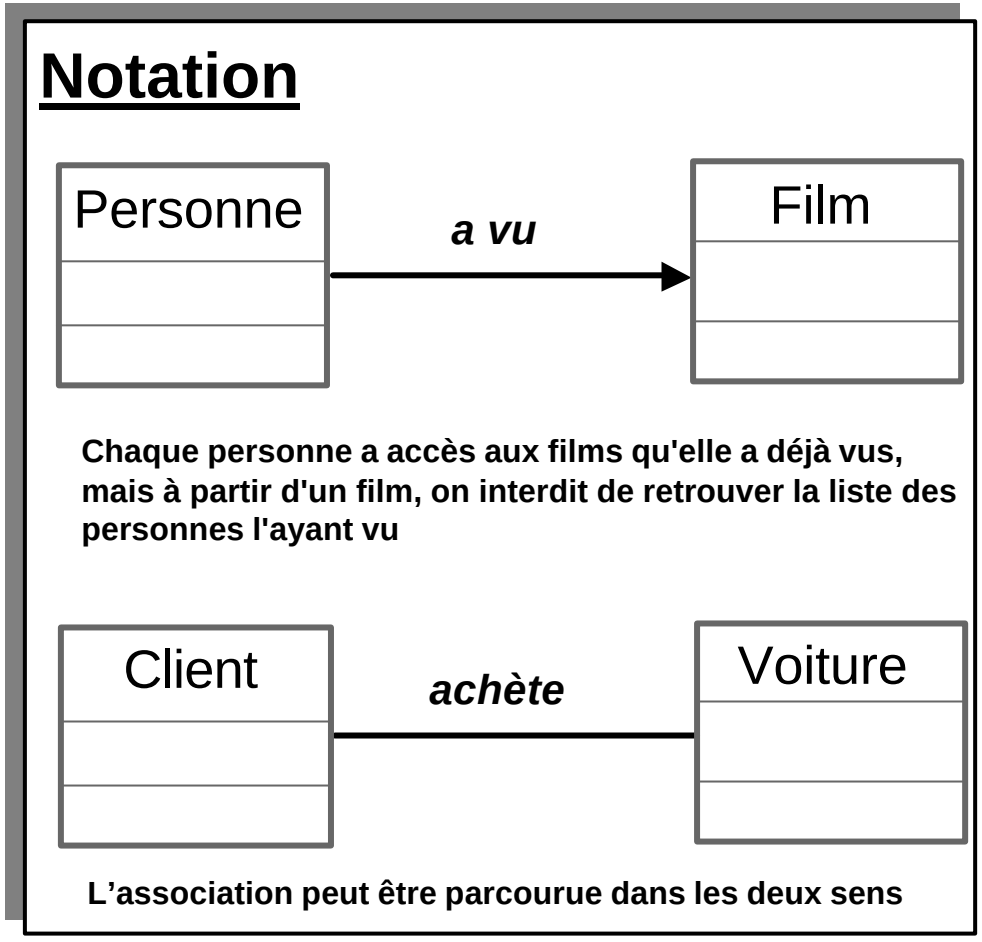
- Mais, elle peut aussi apparaître sur une seule et même classe

  Dans ce cas précis, l'association est dite **réflexive**



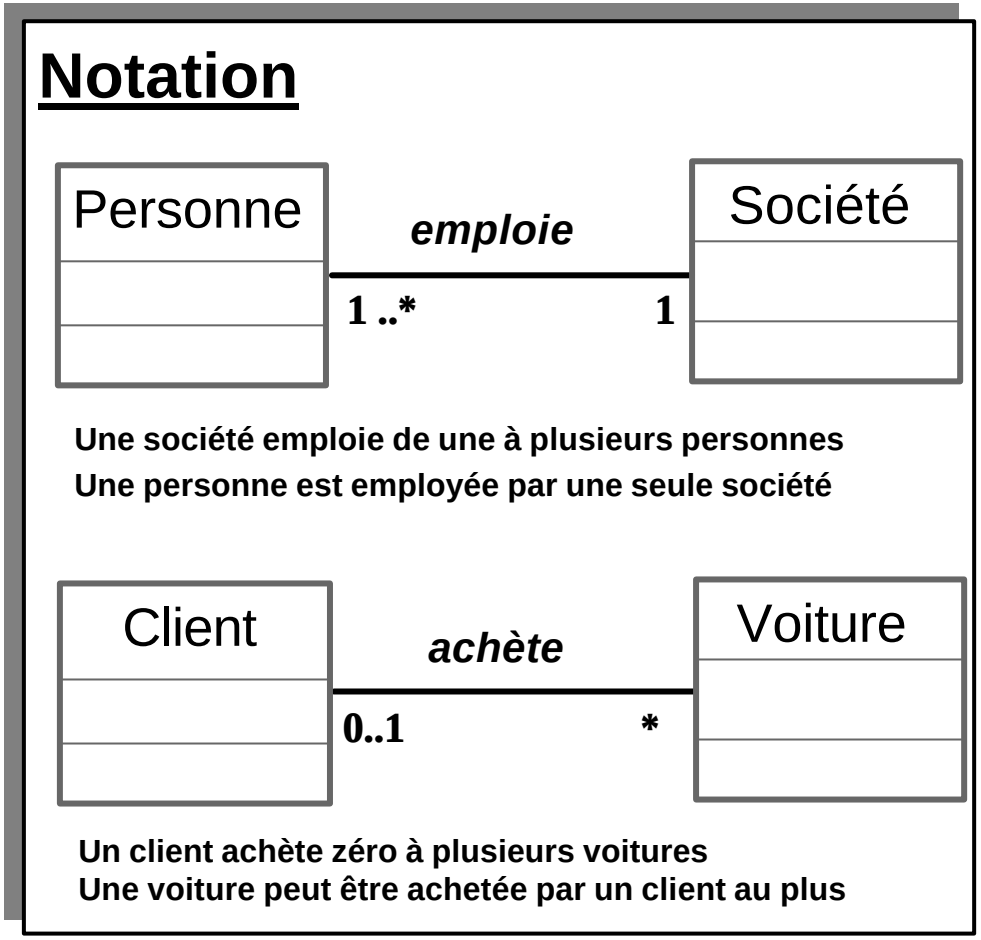
Navigabilité de l'association

- La **navigabilité** d'une association permet de définir dans quel sens l'association peut être parcourue
- La navigabilité d'une association est modélisée par une flèche sur l'extrémité pouvant être atteinte par navigation
- La navigabilité peut être bi-directionnelle
  L'absence de flèche sur les deux extrémités signifie que l'association est bi-directionnelle



Multiplicités de l'association

- La **cardinalité** d'un ensemble est le nombre d'éléments de cet ensemble
- La **multiplicité** est la spécification des valeurs de cardinalité admissibles pour un ensemble
- La **multiplicité** est associée à une extrémité de l'association et indique combien d'instances de la classe considérée peuvent être liées à une instance de l'autre classe



Multiplicités de l'association

La multiplicité est une spécification respectant les conventions suivantes :

1	: un et un seul (notation facultative)
0..1	: zéro ou un
N	: exactement N (N: entier naturel)
M..N	: de M à N (deux entiers naturels)
*****	: de zéro à plusieurs
0..*	: de zéro à plusieurs
1..*	: de un à plusieurs
N..*	: N ou plus (N: entier naturel)

Contraintes sur association

D’autres types de contraintes existent sur une association :

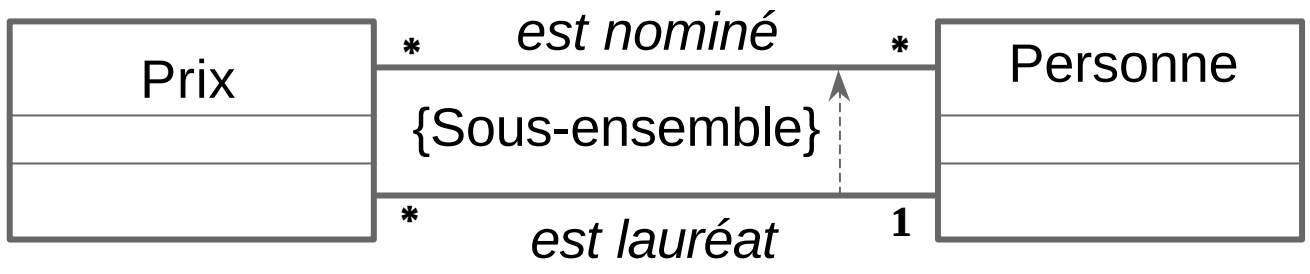
- Des contraintes prédéfinies :
  - Les contraintes **ensemblistes** : {Sous-ensemble}
  - Les contraintes **d’ordonnancement** : {Ordonné}
  - Les contraintes **d’exclusion** : {Ou - exclusif}

- Des contraintes spécifiques au moyen du langage **OCL** (Object Constraint Language)

Contraintes ensemblistes sur association

- Ce type de contrainte permet de modéliser le cas où un ensemble d'associations est inclus dans un autre ensemble d'association

Notation

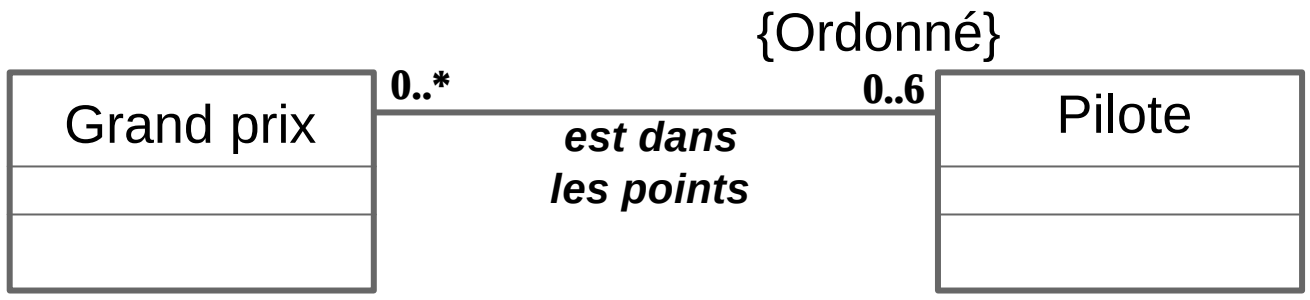


Les personnes lauréates d'un prix sont obligatoirement choisies parmi celles qui sont nominées pour ce prix

Contraintes d'ordonnancement sur association

- Ce type de contrainte permet de modéliser le cas où pour une instance donnée, l'ensemble des instances avec lesquelles elle est en relation doit être ordonné

Notation



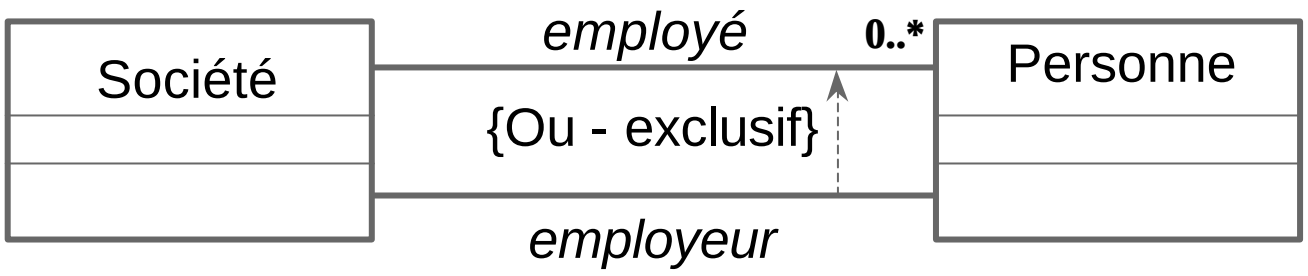
Lors d'un grand prix au maximum 6 pilotes peuvent être dans les points

Ces 6 pilotes sont classés à l'arrivée

Contraintes d'exclusion sur association

- Ce type de contrainte permet de modéliser le cas où pour une instance donnée d'une classe, une seule association prise parmi plusieurs possibles, peut être valide à un instant donné

Notation



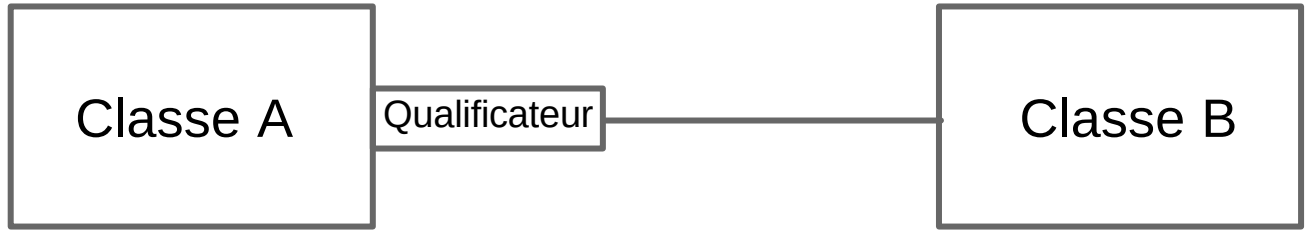
Une personne peut être soit employée par une société, soit employeur d'une société

Mais une personne ne peut pas être à la fois employeur et employé

Qualificateur d’une association

- Le qualificateur est un attribut ou un ensemble d’attributs permettant de partitionner l’ensemble des instances d’une classe qui sont en relation avec une instance donnée
- Le qualificateur permet de restreindre la multiplicité de l’association

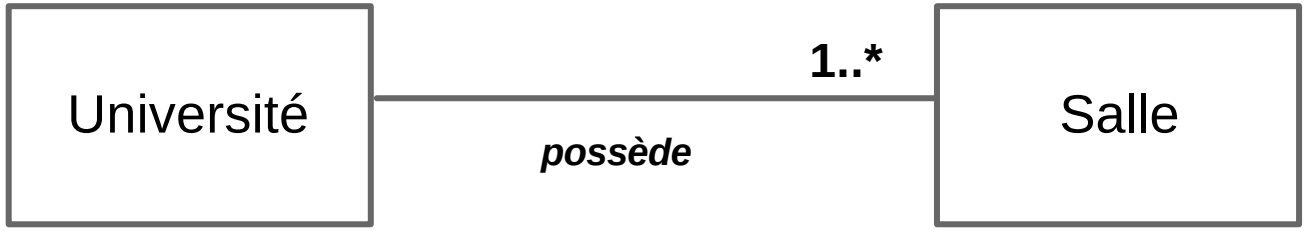
Notation



Qualificateur = attribut1, …, attributN

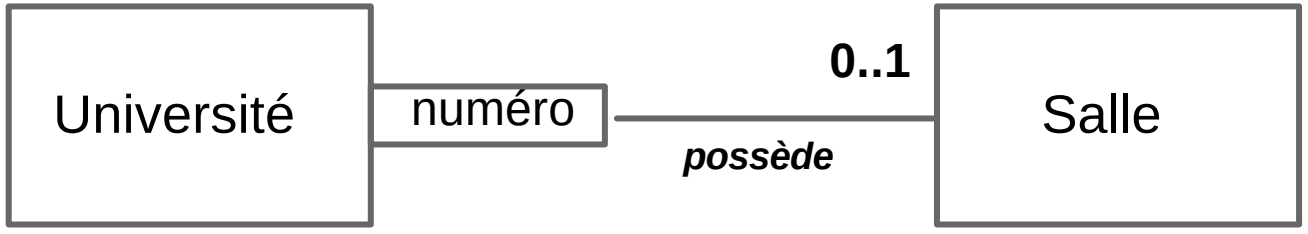
Qualificateur d'association

Exemples



Une université possède de une à plusieurs salle(s)

Une salle appartient à une et une seule université

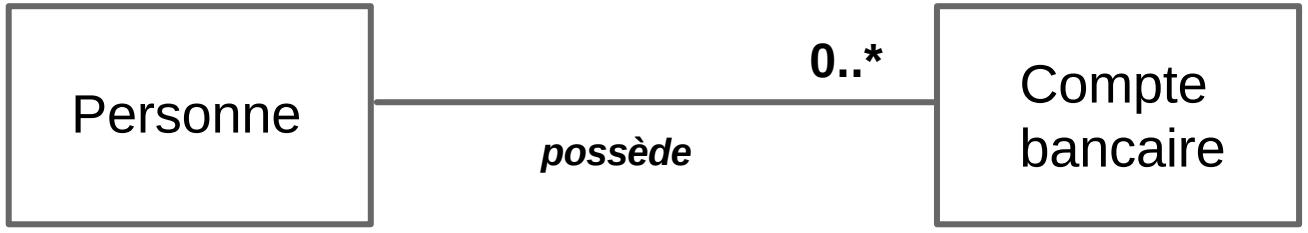


Une université possède au plus une salle ayant ce numéro

Une salle appartient à une et une seule université

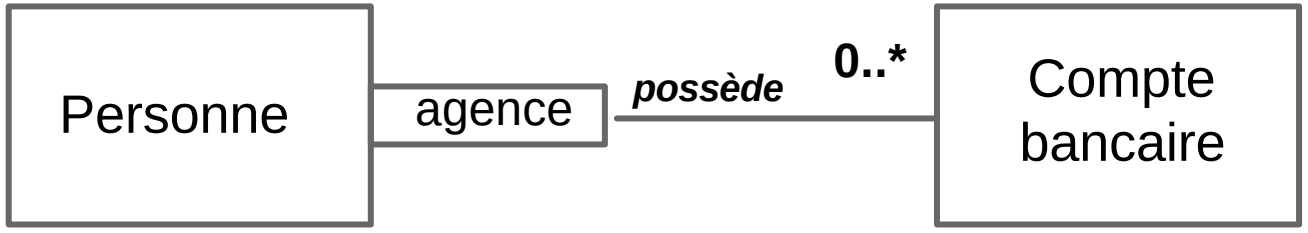
Qualificateur d'association

Exemples



Une personne possède de zéro à plusieurs compte(s) bancaire(s)

Un compte bancaire appartient à une et une seule personne

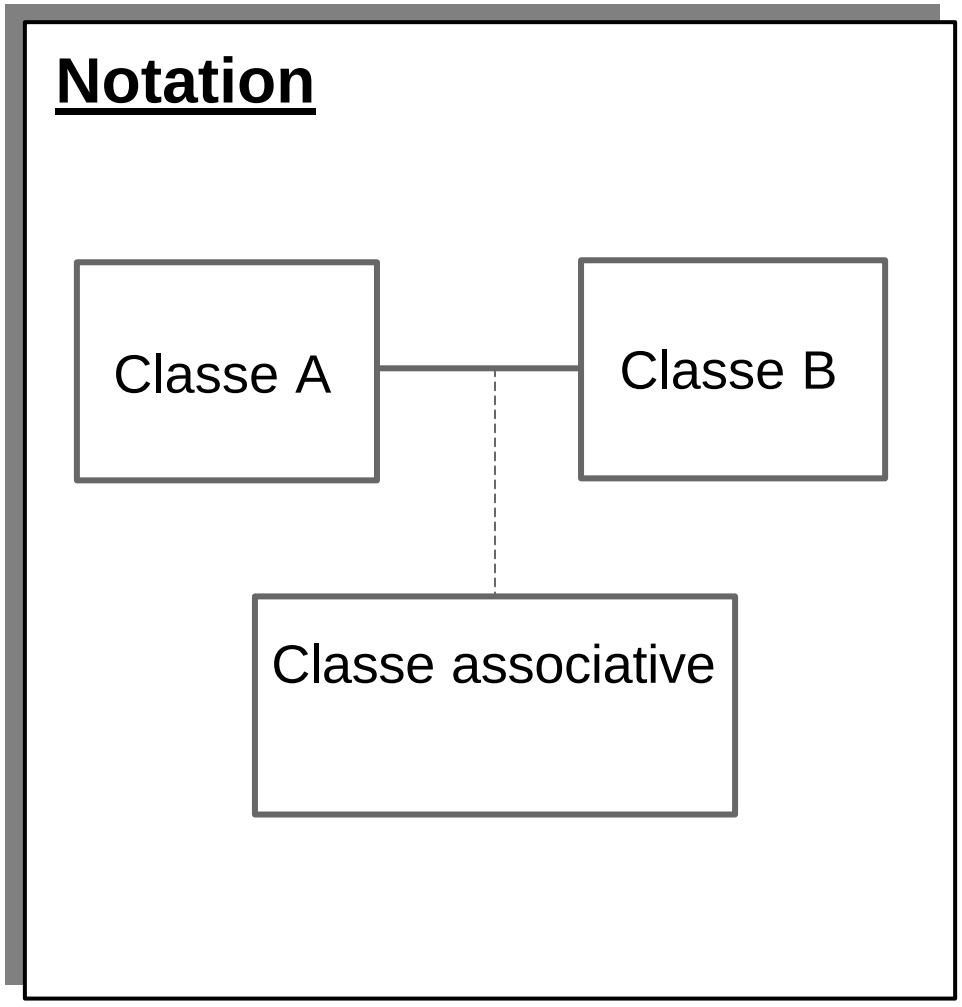


Une personne possède de zéro à plusieurs compte(s) bancaire(s) par agence

Un compte bancaire appartient à une et une seule personne

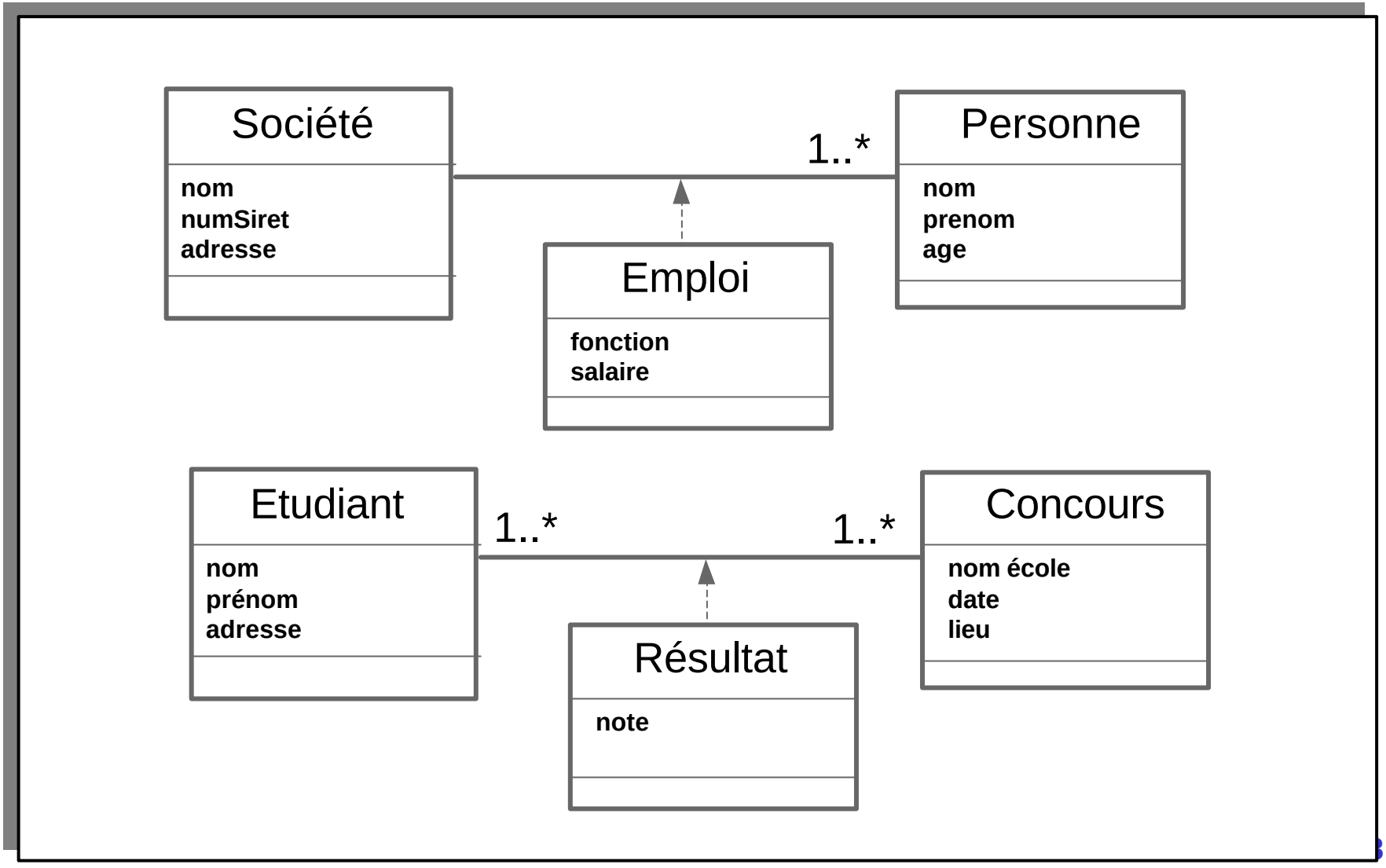
Classe associative

- Une association peut être matérialisée par une classe dans une des circonstances suivantes :
  - si l'association est porteuse d'attributs
  - si l'association se matérialise par un objet concret dans le monde réel
  - si l'association est de multiplicité M .. N
- Une classe associative est une classe à part entière
- Elle est modélisée par un lien en pointillé allant de la classe vers l'association concernée



Classe associative

Exemples



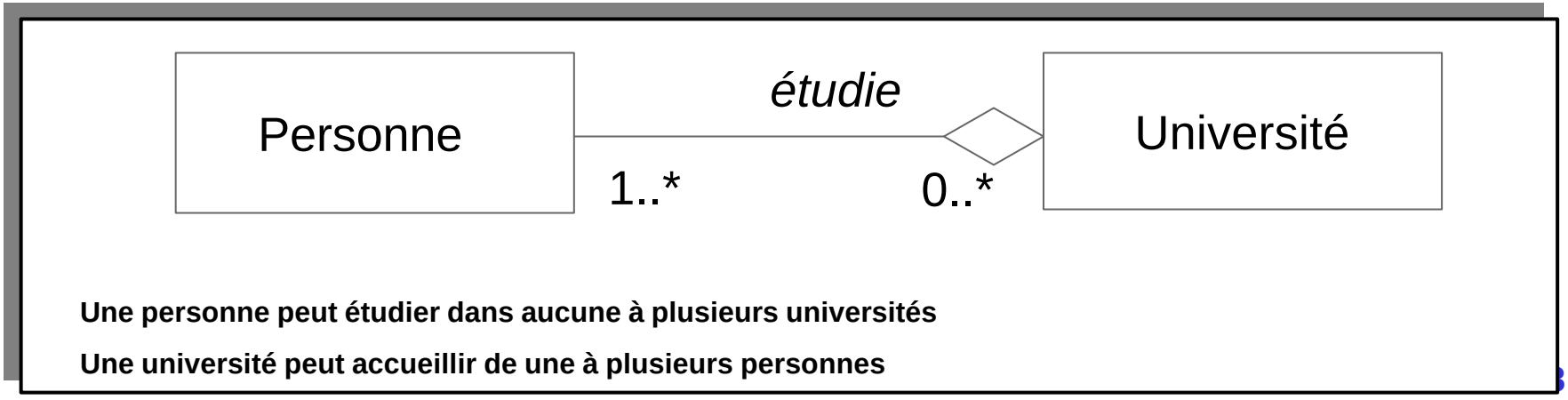
Les différents types d'association

Il existe plusieurs types d'association

- L'**agrégation**
  Forme spéciale d'association entre un tout et une partie
- La **composition**
  Forme spéciale d'agrégation où le cycle de vie de la partie est régi par celui du tout
- L'**héritage**
  Forme spéciale d'association permettant de factoriser les caractéristiques et comportement communs à un ensemble de classes
- L'**association simple**
  Ce sont les associations qui ne se réclament d'aucune des catégories précédemment citées

Agrégation

- Une agrégation est une association **non symétrique** dans laquelle l'une des deux classes joue un rôle prépondérant
- Une agrégation est une **relation tout-partie** entre un agrégat (le tout) et un composant (la partie)
- L'agrégation est représentée par un losange blanc du côté de l'agrégat
- Le composant **peut appartenir simultanément** à plusieurs agrégats
- Le cycle de vie des composants **n'est pas tributaire** de celui de l'agrégat

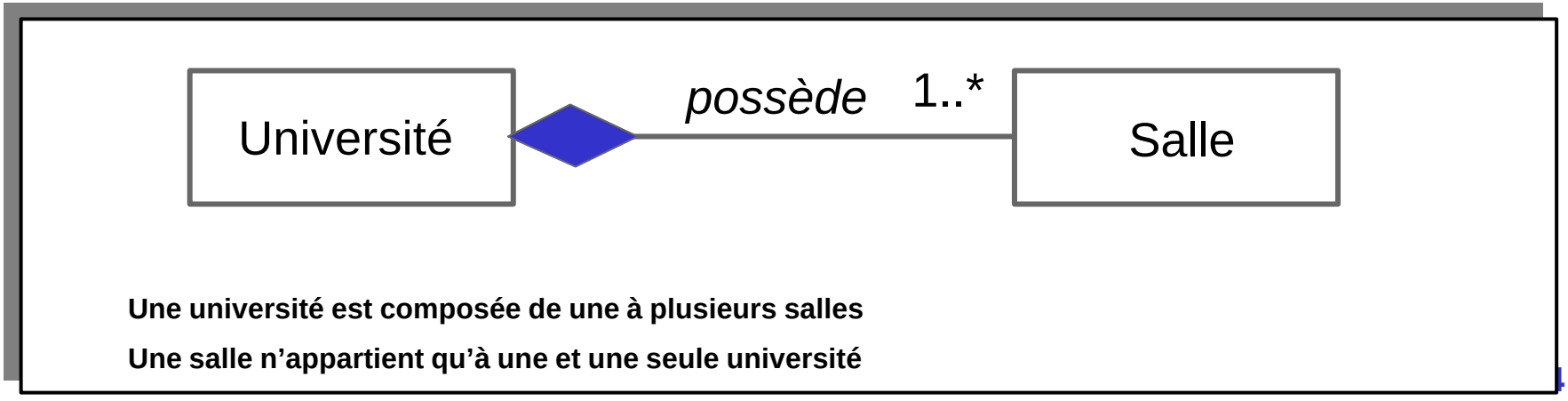


Une personne peut étudier dans aucune à plusieurs universités

Une université peut accueillir de une à plusieurs personnes

Composition

- Une composition est une **agrégation** à part entière
- La composition est représentée par un losange noir du côté de l'agrégat
- Le composant **ne peut pas appartenir simultanément** à plusieurs agrégats (multiplicité 1 ou 0..1 côté agrégat)
- Le cycle de vie des composants **est tributaire** de celui de l'agrégat
- Si la multiplicité est 0..1 côté agrégat, le composant peut ne pas être associé à l'agrégat immédiatement, mais une fois l'association effectué le composant vit et meurt avec l'agrégat

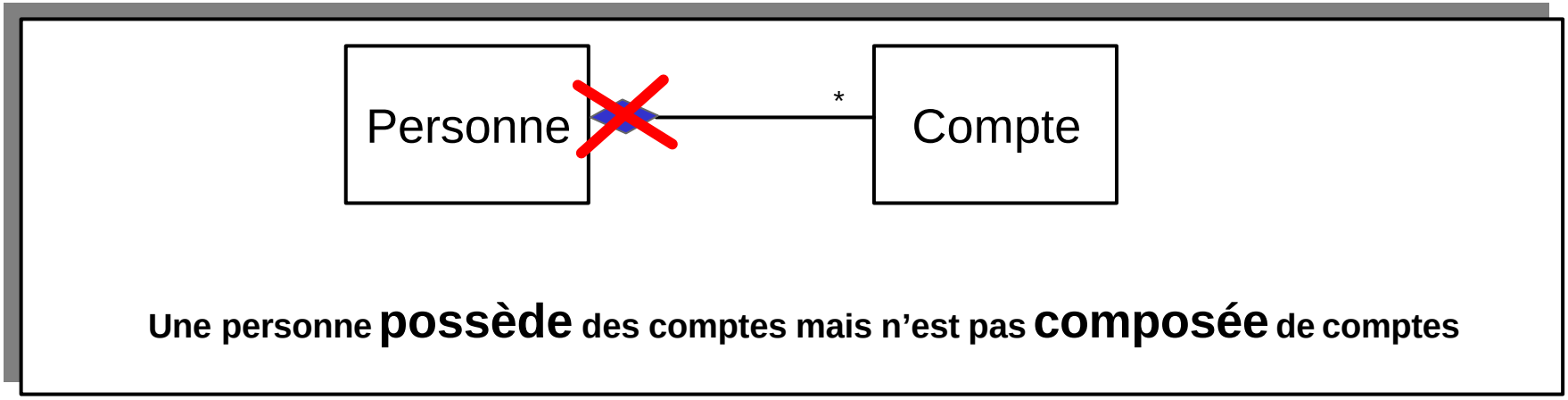


Une université est composée de une à plusieurs salles

Une salle n'appartient qu'à une et une seule université

Composition : abus de langage !

- Attention : on parle très souvent de composition pour désigner une simple association entre deux classes. Cela ce traduit par l'existence d'un attribut qui référence une (des) instance(s) d'une autre classe.

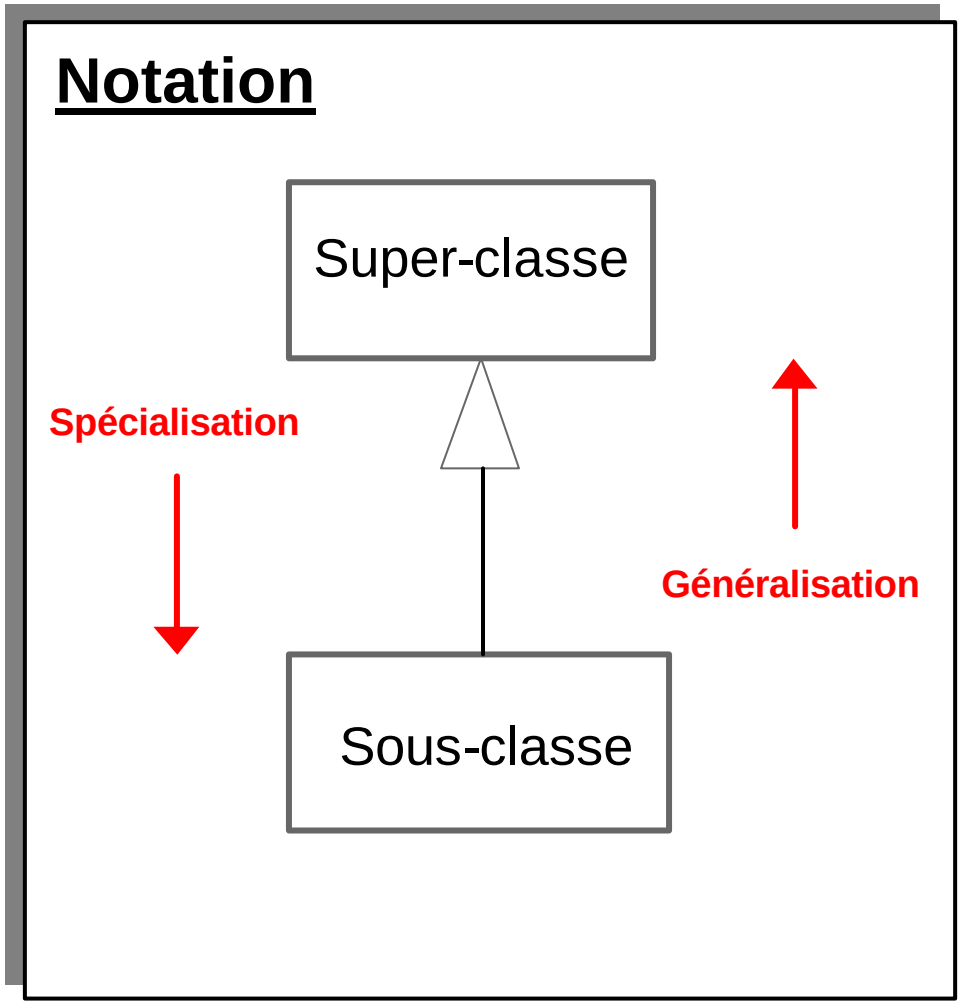


Généralisation / Spécialisation

- La généralisation / spécialisation est une relation de **classification** entre une classe plus générale et une classe plus spécialisée

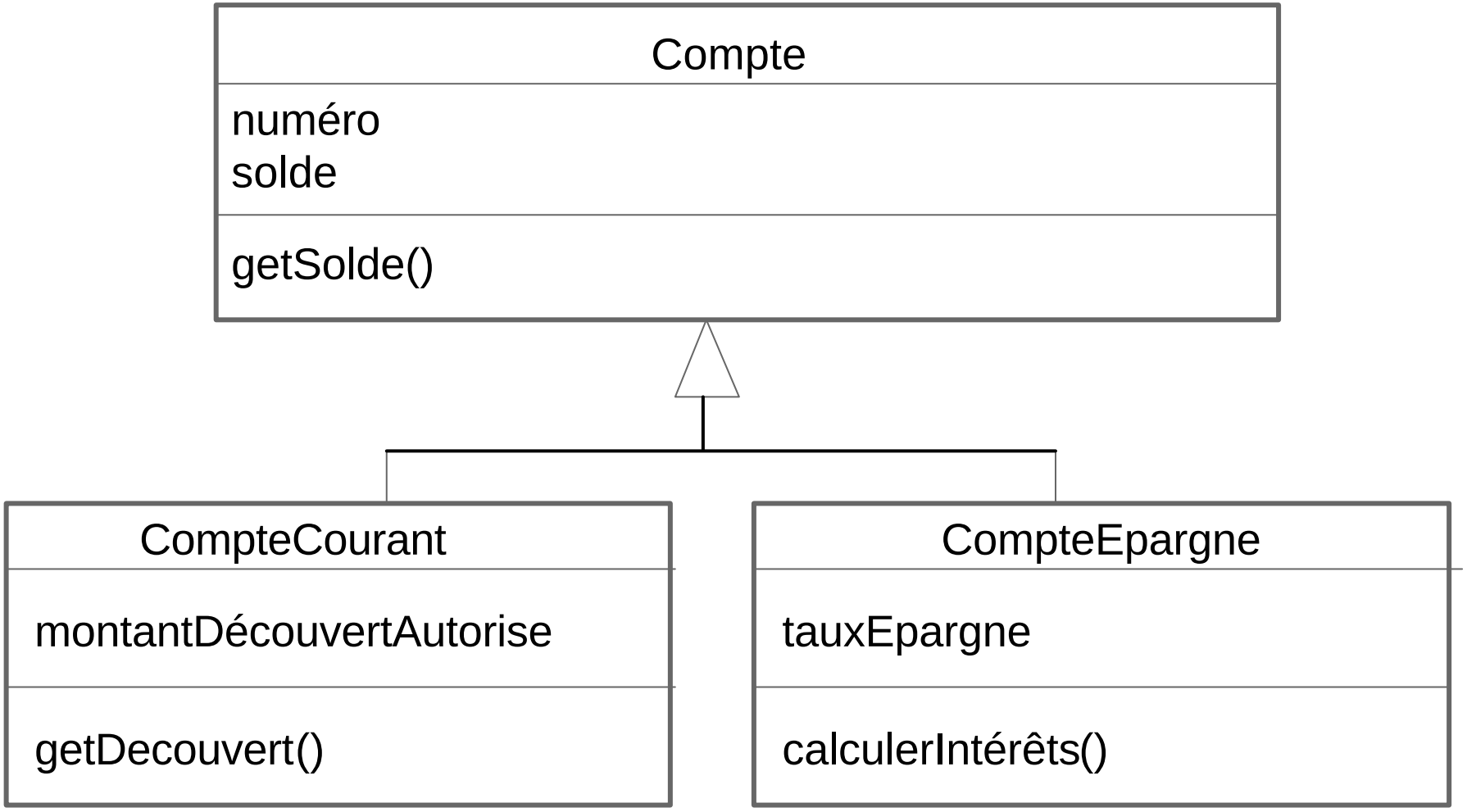
- On l'appelle aussi **relation d'héritage** ou **relation « est-une-sorte-de »**

- La généralisation est représentée au moyen d'une flèche pointant de la classe la plus spécialisée vers la classe la plus générale



Généralisation / Spécialisation

Exemple



Classe et opération abstraites

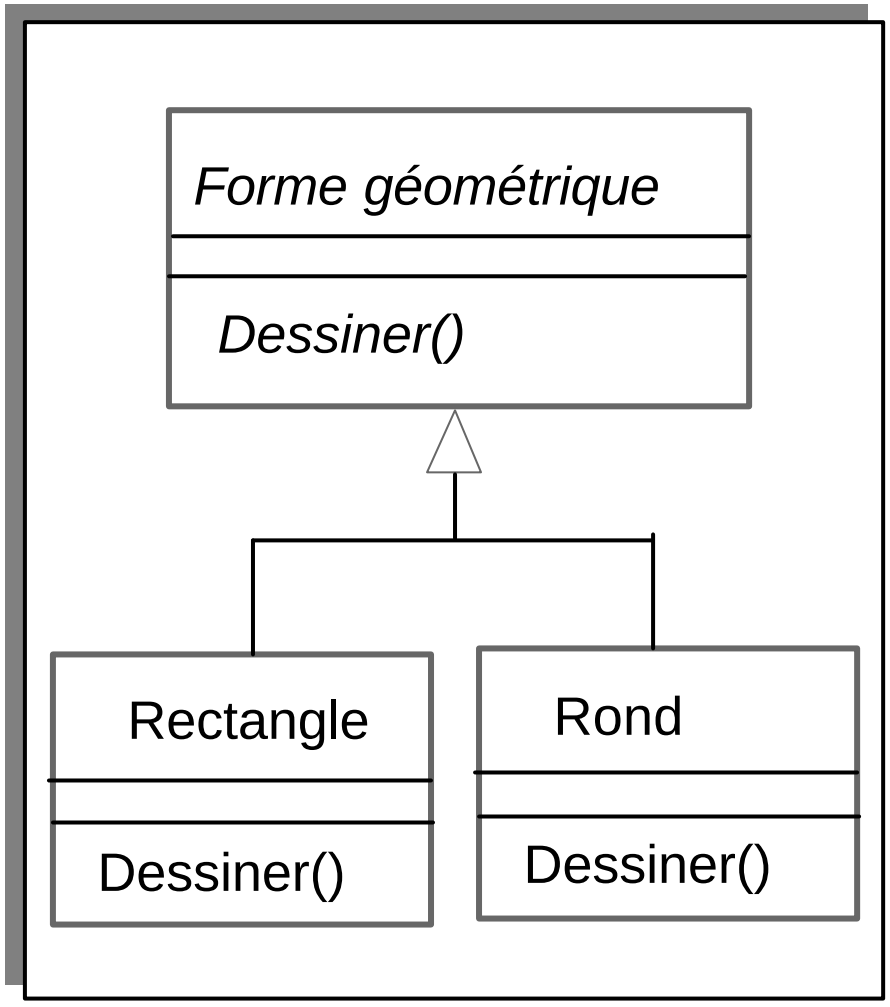
- Une **classe abstraite** est une classe pour laquelle il n'est pas possible de créer d'instances directement

  Son nom est écrit en *italique*

- Une **opération abstraite** d'une classe A est une opération ne possédant pas d'implémentation dans A mais qui doit obligatoirement être implémentée dans les sous-classes de A

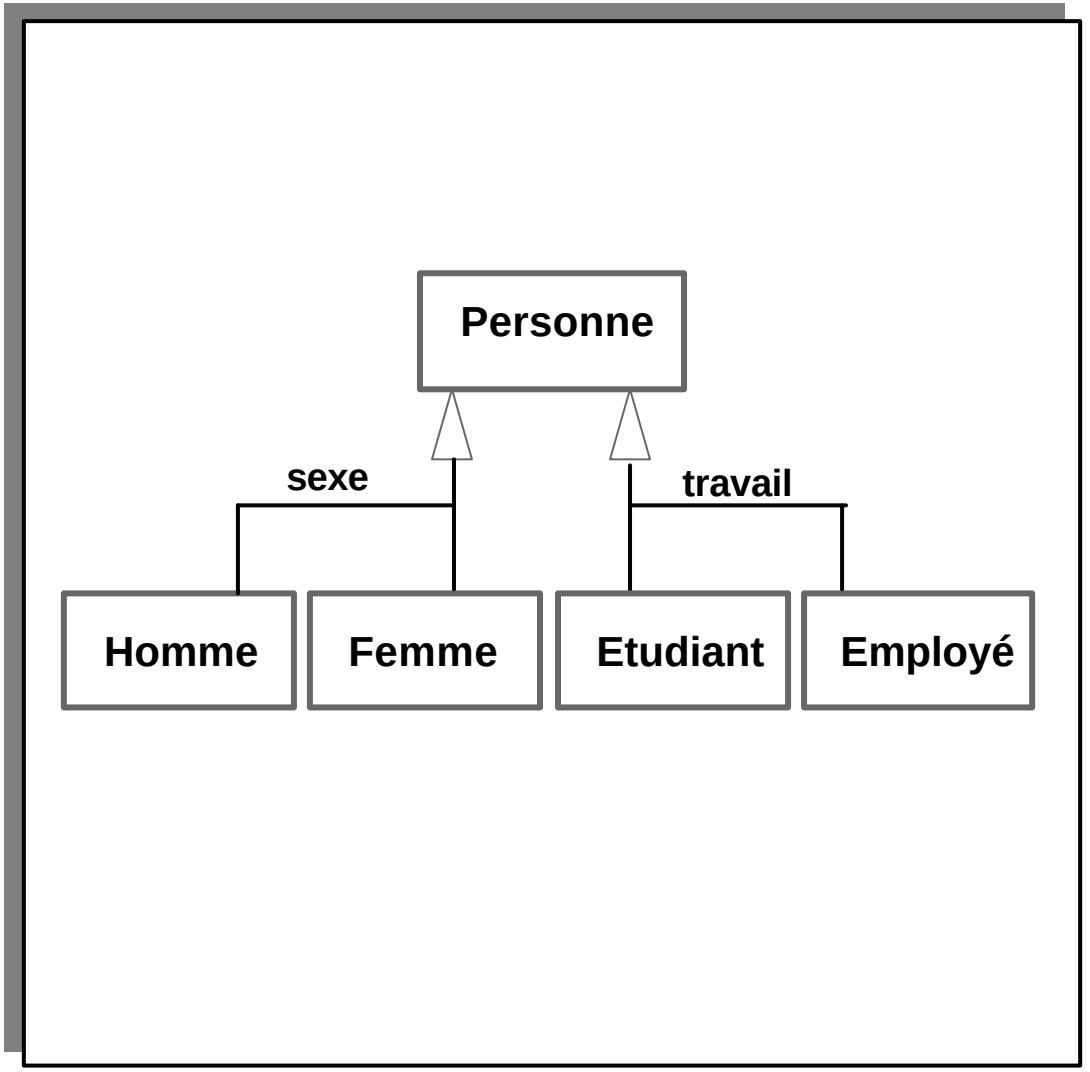
  Sa signature est écrite en *italique*

- Toute classe contenant au moins une opération abstraite est abstraite



Discriminant sur relation d'héritage

- La spécialisation d'une super-classe peut avoir lieu selon différents **critères** simultanés
- Chacun de ces critères est représenté par une chaîne de caractères et s'appelle un **discriminant**
- Le discriminant est positionné à côté de la sous-arborescence qu'il qualifie



Contraintes sur généralisation / spécialisation

Il est possible d'exprimer deux types de contraintes prédéfinies sur les sous-classes d'une généralisation:

- La contrainte de **complétude**

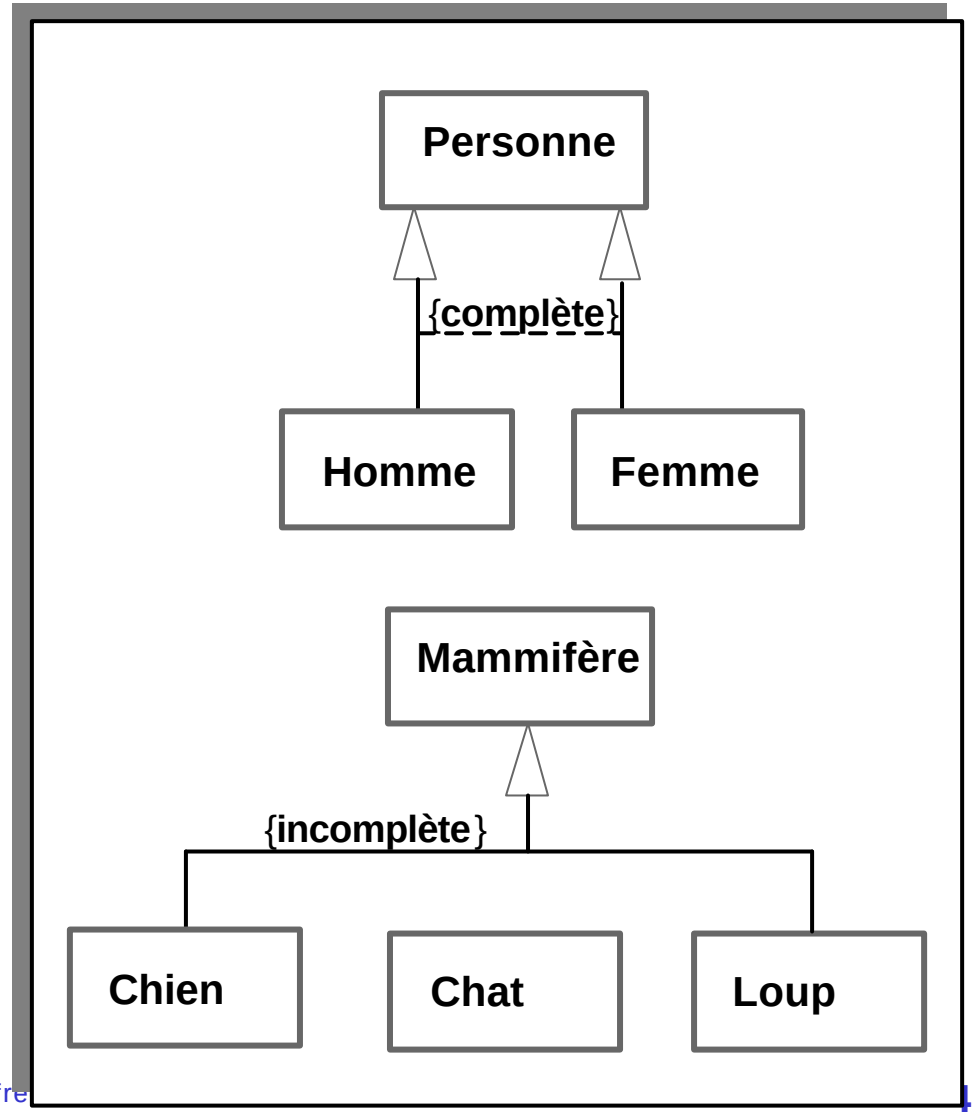
  Précise l'état d'avancement de la classification proposée par la généralisation / spécialisation

- La contrainte de **chevauchement**

  Contrainte ensembliste sur les instances de la sous-classe vis-à-vis des instances de la super-classe

Contrainte de complétude sur relation d'héritage

- La contrainte de complétude permet d'indiquer si la généralisation peut être étendue ou non
- **{complète}** indique que l'on ne peut plus ajouter de classe à l'arborescence
- **{incomplète}** indique que l'arborescence peut être complétée ultérieurement



Contrainte de chevauchement sur relation d'héritage

- La contrainte de chevauchement apporte des précisions sur la nature des instances de la super-classe
- **{disjoint}** indique que l'ensemble des instances des sous-classes forment une **partition** de la super-classe
- **{chevauchement}** indique qu'il peut exister des instances qui soient à la fois instance de deux sous-classes

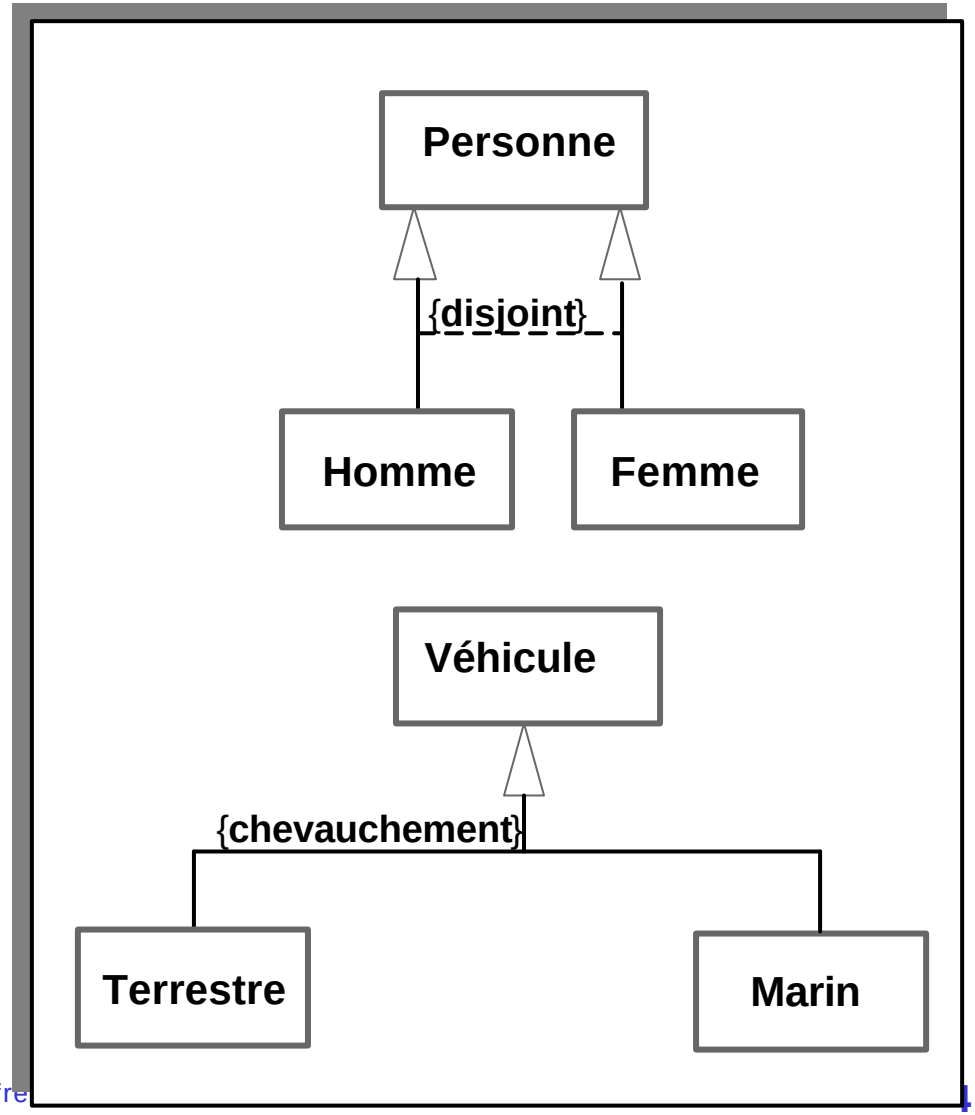


Diagramme de classes (Recommandations)

- Toujours garder à l'esprit qu'un diagramme de classe propose une vision **statique** des données du problème
- Les associations d'un diagramme de classes sont statiques, mais la création des liens entre objets est **dynamique**
- Ne jamais hésiter à donner les multiplicités

Diagramme de classes (Recommandations)

- Dans la mesure du possible, éviter les **discriminants** dans les associations de type généralisation / spécialisation
- Eviter l'utilisation des **contraintes de chevauchement** dans les associations de type généralisation / spécialisation
- Privilégier la **délégation** à l'héritage