Introduction à UML

Diagramme de Classes

BSTORM CONSULTING

Plan

1. Introduction à la Spécification
2. Aperçu d'UML
3. Diagramme de Use Case et Description Textuelle
4. Diagramme d'Activité
5. Diagramme de Séquence
6. **Diagramme de Classes**
7. Diagramme d'États

Programmation Orientée Objet

Programmation Procédurale

La plupart des langages de programmation développés au cours des 30 dernières années sont ce qu'on appelle des langages orientés objet. Avant cela c'étaient plutôt des **langages de programmation procédurale** dans lesquels le programme est écrit comme une longue procédure.

Même s'il peut contenir des fonctions et des sous-programmes pour le rendre plus modulaire et facile à gérer, il s'agit en réalité d'un long code combinant données et logique.

Programmation Orientée Objet

Désormais, dans un langage orienté objet, ce programme serait plutôt scindé en objets autonomes, presque comme si vous disposez de plusieurs mini-programmes. Chaque objet représente une partie différente de l'application et chaque objet contient ses propres données et sa propre logique et ils communiquent entre eux.

Il est conçu pour faciliter la réflexion sur votre programme. Les objets ne représentent pas des idées abstraites mystérieuses. Ils représentent d'abord des éléments tels que les employés, les images, les comptes bancaires, les objets joueur, les objets voiture, tout ce qui existe réellement dans votre programme.

Classes et Objets

Classes

- Décrit ce que quelque chose est, mais ce n'est pas la chose elle-même. C’est un plan, une définition, une description.

Définit :

1. **Attributs** : description des caractéristiques.
    - Nom, taille, poids, âge, etc.
2. **Méthodes** (function) : description de ce qu’il peut faire.
    - Marcher, courir, sauter, parler, dormir, etc.

Objets

Les objets sont créer à partir d'une classe.

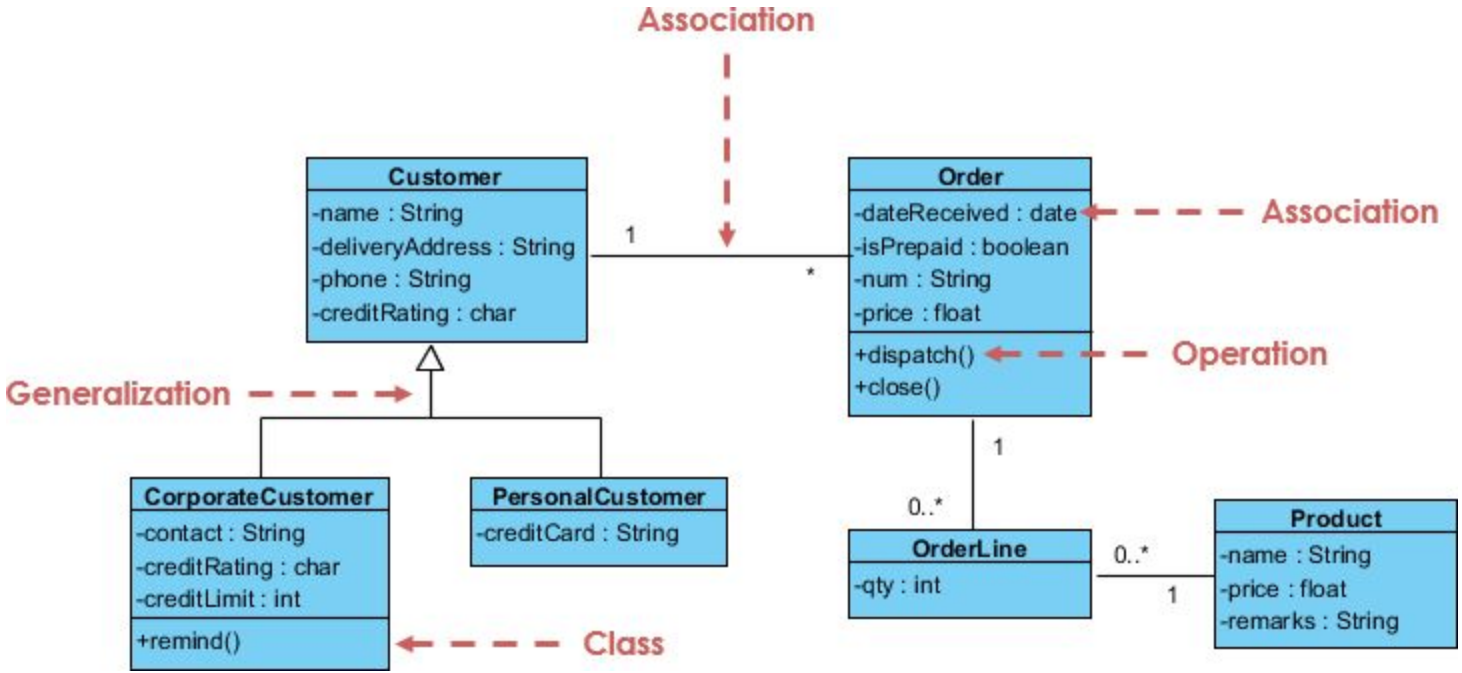
- Plusieurs objets peuvent être créer à partir d'une seule classe.
- Exemple : l'objet Bernard (créer à partir de la classe personne)
  - Nom : Bernard
  - Taille : 1,78
  - Poids : 78
  - marcher()
  - sauter()
  - etc.

Diagramme de Classes

Big Picture

Le **diagramme de classes** est l'une des diagrammes les plus utilisées, car il décrit la structure d'un système particulier en modélisant ses classes, ses attributs et les différents types de relations statiques qui existent entre eux.

Exemple


Association
Customer
-name : String
-deliveryAddress : String
-phone : String
-creditRating : char
1
*
Order
-dateReceived : date
-isPrepaid : boolean
-num : String
-price : float
+dispatch()
+close()
Association
Operation
Generalization
CorporateCustomer
-contact : String
-creditRating : char
-creditLimit : int
+remind()
PersonalCustomer
-creditCard : String
Class
1
0..*
OrderLine
-qty : int
0..*
1
Product
-name : String
-price : float
-remarks : String

Définition

Le **diagramme de classes** est un schéma utilisé pour représenter les classes du système ainsi que les différentes relations entre celles-ci.

Contexte d'utilisation

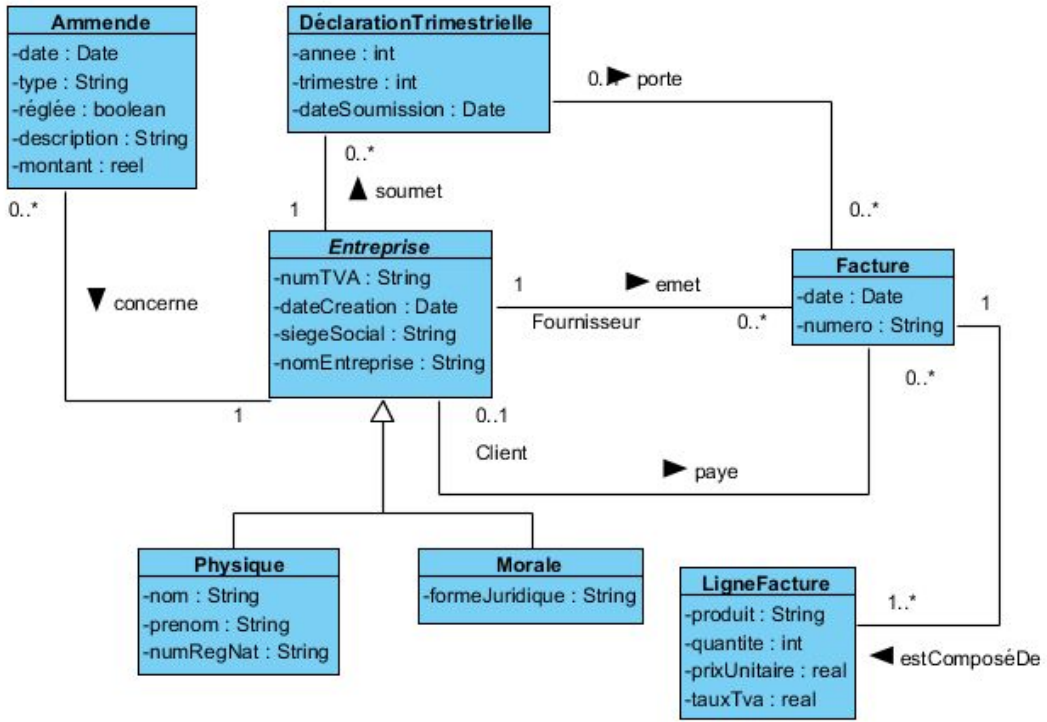
Analyse (Business)

- Décrire la structure des entités manipulées dans le domaine d'application.

Conception (Système)

- Représenter la structure d'un code orienté objet (conception).
- Représenter les classes à implémenter dans un langage de développement précis (implémentation).

Exemple


Ammende
-date : Date
-type : String
-réglée : boolean
-description : String
-montant : reel
DéclarationTrimestrielle
-annee : int
-trimestre : int
-dateSoumission : Date
0..* porte
0..*
soumet
1
0..*
0..*
concerne
Entreprise
-numTVA : String
-dateCreation : Date
-siegeSocial : String
-nomEntreprise : String
1
emet
Fournisseur
0..*
Facture
-date : Date
-numero : String
1
0..*
1
0..1
Client
paye
Physique
-nom : String
-prenom : String
-numRegNat : String
Morale
-formeJuridique : String
LigneFacture
-produit : String
-quantite : int
-prixUnitaire : real
-tauxTva : real
1..*
estComposéDe

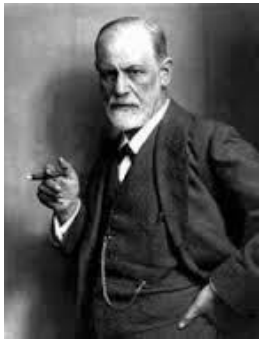
Classe

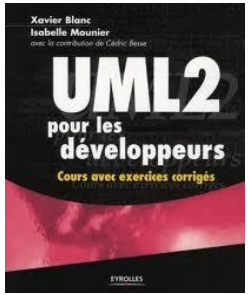
Description d'un groupe d'objets ayant tous des rôles similaires dans le système, ils ont donc des propriétés communes.

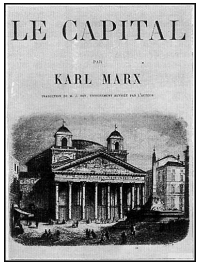
- **Attributs** : représentent l'état des objets, ce que les objets de la classe "*savent*".
- **Opérations** : représentent le comportement possible des objets, ce que les objets de la classe "*peuvent faire*".

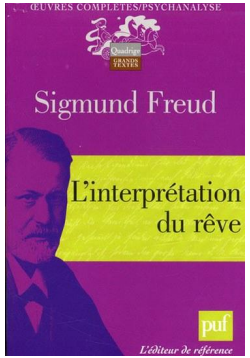
Objet / Classe

Classifier ces différents objets (entités)



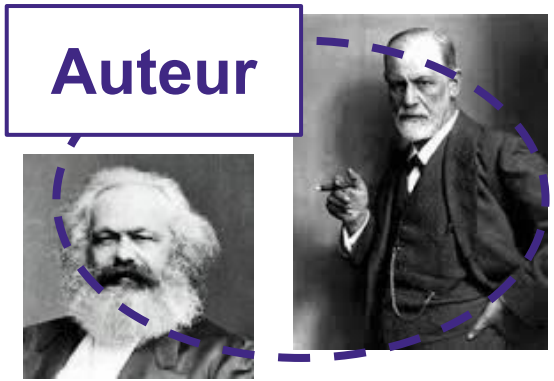






Objet / Classe

Classifier ces différents objets (entités)



Objet / Classe

objet

Instance de classe

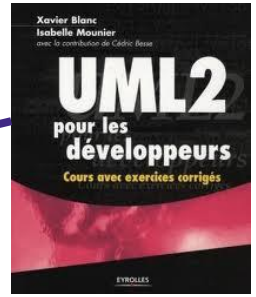
CLASSE

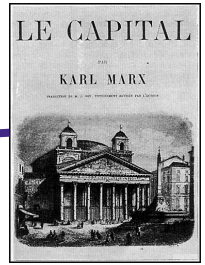
Livre

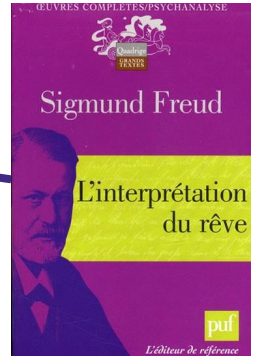
est de type

est de type

est de type



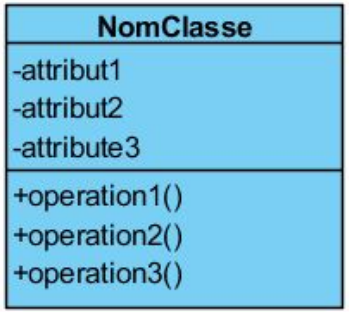




Notation

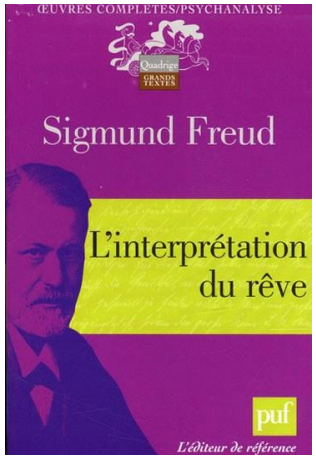
En 3 parties :

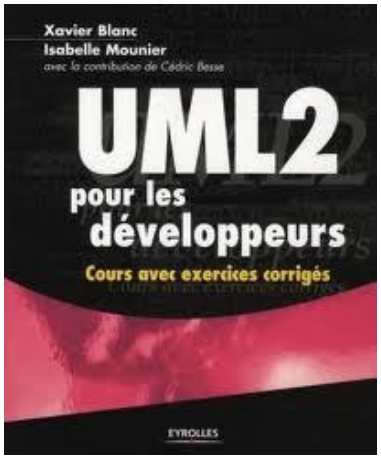
1. **Le nom de la classe**
2. **Les attributs**
3. **Les opérations** (méthodes)

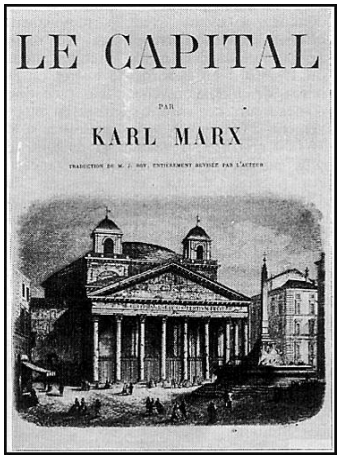


Attributs

Identifier les attributs de cette classe ?

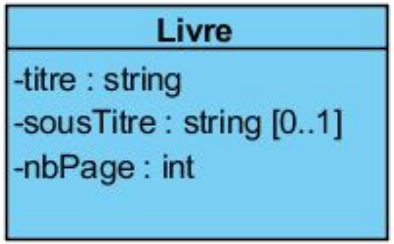






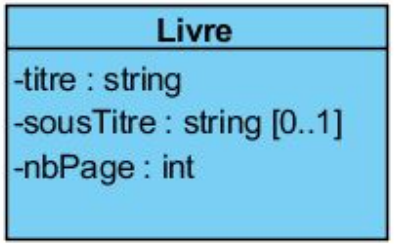
Attributs

1. Propriété structurelle partagée par tous (?) les objets d'une classe.
2. À chaque attribut est associé un type qui définit un espace de valeur possible.
3. Chaque objet peut avoir une valeur différente pour un attribut particulier.



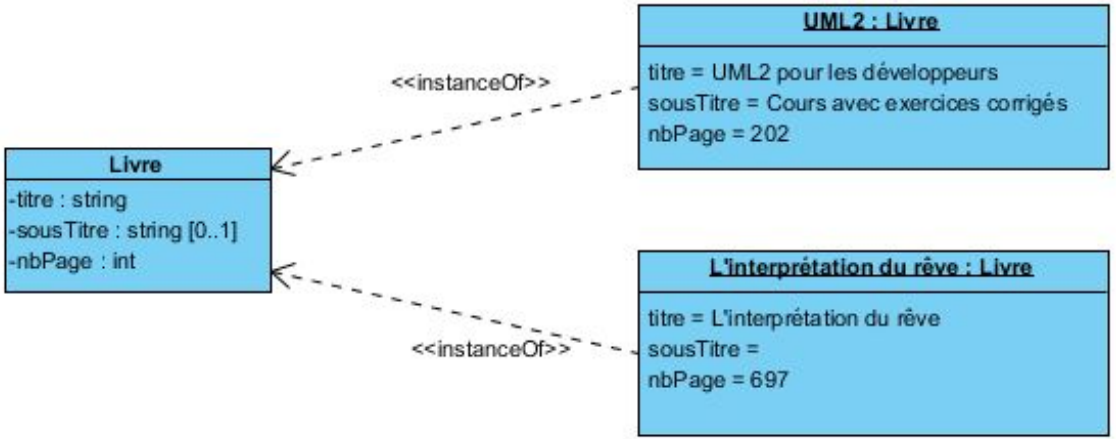
Attributs

1. Propriété structurelle partagée par tous (?) les objets d'une classe.
2. À chaque attribut est associé un type qui définit un espace de valeur possible.
3. Chaque objet peut avoir une valeur différente pour un attribut particulier.



Attributs

1. Propriété structurelle partagée par tous (?) les objets d'une classe.
2. À chaque attribut est associé un type qui définit un espace de valeur possible.
3. Chaque objet peut avoir une valeur différente pour un attribut particulier.



Attributs de classe et d’instance

Attribut d’instance (par défaut)

- Chaque instance peut avoir une valeur différente pour cet attribut.
- Exemple
  - Chaque client possède son propre prénom.

Attribut de classe

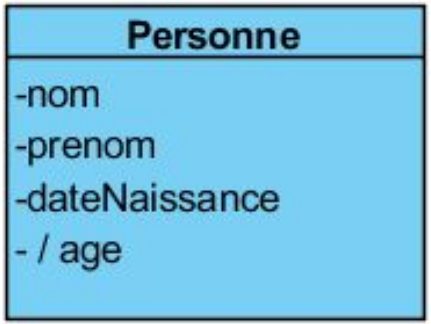
- Toutes les instances partagent la même valeur pour cet attribut.
- Exemple
  - Tous les prêts ont une durée maximale de 5 jours.

Attributs dérivés

Un **attribut dérivé** est une propriété intéressante pour l'analyse, mais redondante, car sa valeur peut être déduite d'autres informations disponibles dans le modèle.

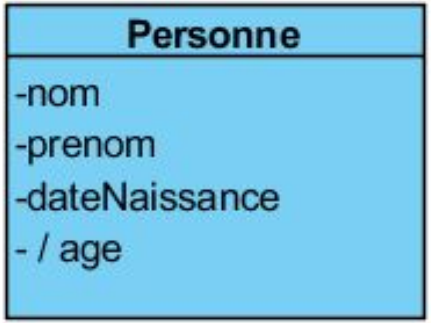
Attributs dérivés

Identifiez l'attribut dérivé dans cette classe.



Attributs dérivés

Âge : puisqu'elle peut être déduite à partir de la date de naissance et aujourd'hui.



Attributs dérivés

Durant l'analyse

- Permet de ne pas faire de choix de prématuré.

Durant la conception / implémentation, 2 solutions:

1. Garder un attribut en mémoire et mettre sa valeur à jour adéquatement si calcul complexe et/ou mise à jour rare.
2. Ne pas stocker pas de valeur redondante, mais la calculer à la demande au moyen d'une opération si calcul simple et/ou mise à jour fréquente.

Syntaxe et sémantique

[visibility][/]name[:type][“[”multiplicity“]”][=initial value][{property string}]

[visibility]

- private (visible uniquement par les opération de la classe)
+ public (visible partout à l’intérieur et à l’extérieur de la classe)
~ package (visible au sein du package uniquement)
# protected (visible au sein de la classe et par les descendants de cette classe)

[/]

/ attribut dont la valeur est dérivée (d’un ou plusieurs autres attributs)

Syntaxe et sémantique

[visibility][/]name**[:type]**[multiplicity][=initial value][{property string}]

[:type]

Spécification du domaine de valeur possible de l'attribut

PrimitiveType	ex: boolean, string, int...
DataType	ex: Date...
Enumeration	

Syntaxe et sémantique

[visibility][/]name[:type]**[multiplicity][=initial value]**[{property string}]

[multiplicity]

[min.. max] : eg, [0.. 1], [3.. 3]= 3, [0.. *] = *,…
Par défaut: attribut monovalué obligatoire, soit [1].
Si min = 0 : attribut facultatif.
Si max > 1 : attribut multivalué.

[=initial value]

Spécification de la valeur initiale de l'attribut.

Syntaxe et sémantique

[visibility][/]name[:type][multiplicity][=initial value]**[{property string}]**

Modifier	Description
id	Property is part of the identifier for the class which owns the property.
readOnly	Property is read only (isReadOnly = true).
ordered	Property is ordered (isOrdered = true).
unique	Multi-valued property has no duplicate values (isUnique = true).
nonunique	Multi-valued property may have duplicate values (isUnique = false).
sequence (or seq)	Property is an ordered bag (isUnique = false and isOrdered = true).
union	Property is a derived union of its subsets.
redefines *property-name*	Property redefines an inherited property named ***property-name***.
subsets *property-name*	Property is a subset of the property named ***property-name***.
property-constraint	A constraint that applies to the property

Les relations entre classes

Les relations entre classes

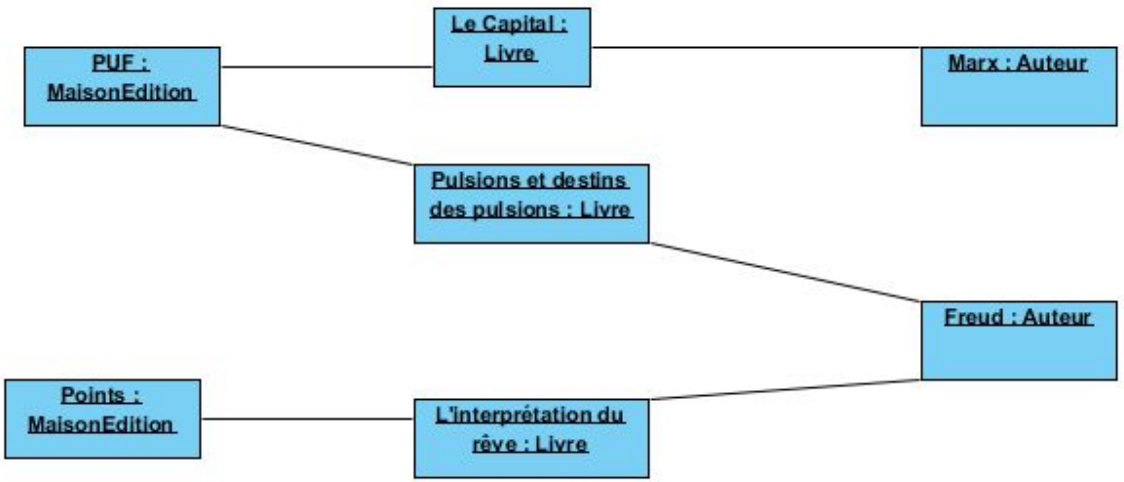
Une classe peut être impliquée dans une ou plusieurs relations avec d'autres classes.

Relation d’association

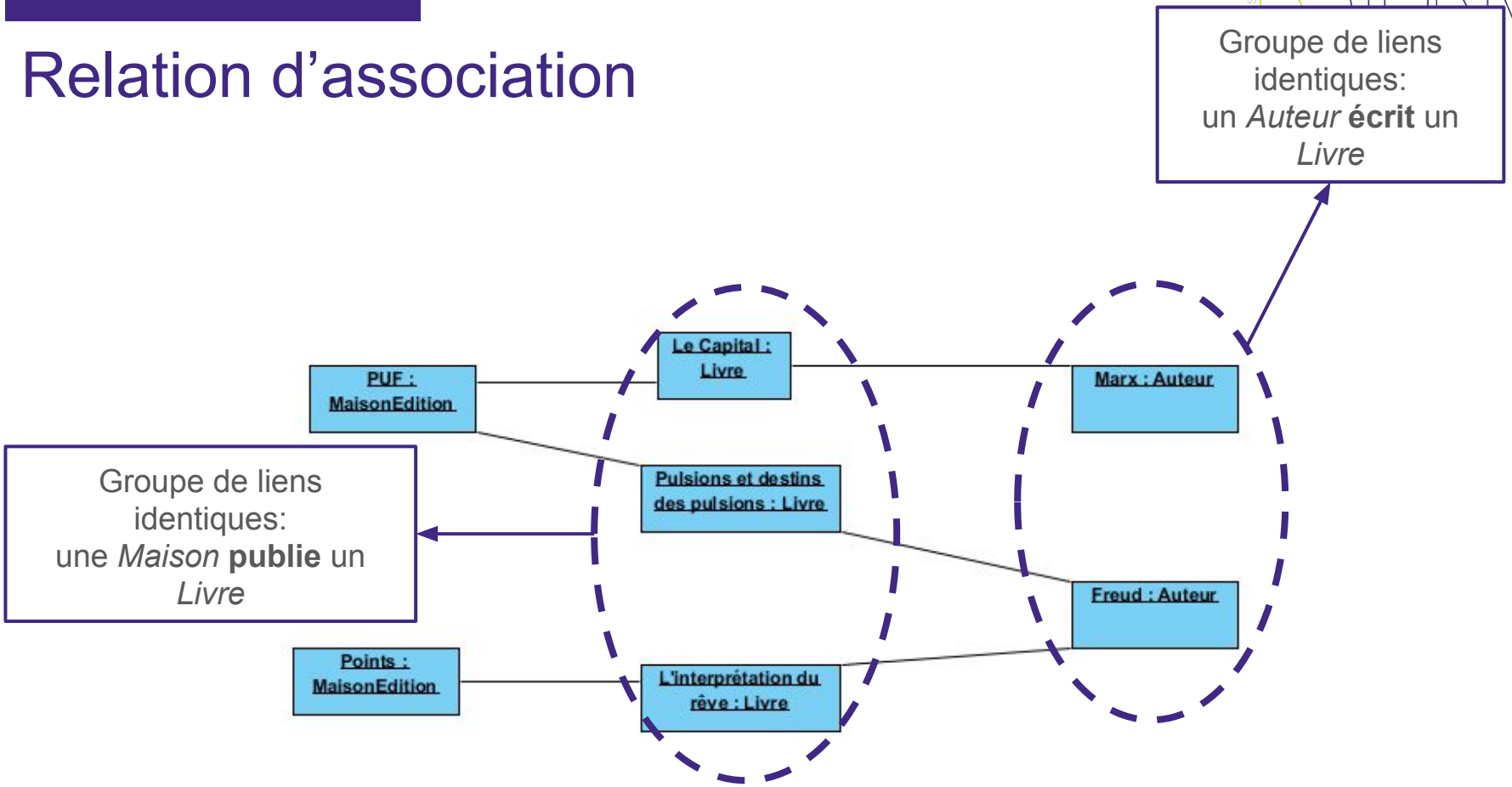
Une **association** décrit un groupe de liens ayant une même structure et une même sémantique.

- L’association est un **type** de relation.
- Une instance d’association est appelé un **lien**.
- Relié par une ligne.

Relation d’association



Relation d'association



Relation d'association

Les association sont caractérisées par :

1. Un **nom**.
2. Une **multiplicité**.

Multiplicité

Combien d'objets de chaque classe prennent part aux relations

- Zéro ou un - 0..1
- Exactement un - 1
- Beaucoup - 0 .. * ou *
- Un ou plusieurs - 1 .. *
- Nombre exact - par exemple 3..4 ou 6
- Ou une relation complexe - par exemple 0..1, 3..4, 6. * signifierait n'importe quel nombre d'objets autres que 2 ou 5

Multiplicité

Chaque **multiplicité** peut être caractérisé par les propriétés **isUnique** et **isOrdered**.

isOrdered	isUnique	Collection type
false	*true*	*Set*
true	*true*	*OrderedSet*
false	*false*	*Bag*
true	*false*	*Sequence*

Objet ou attribut ?

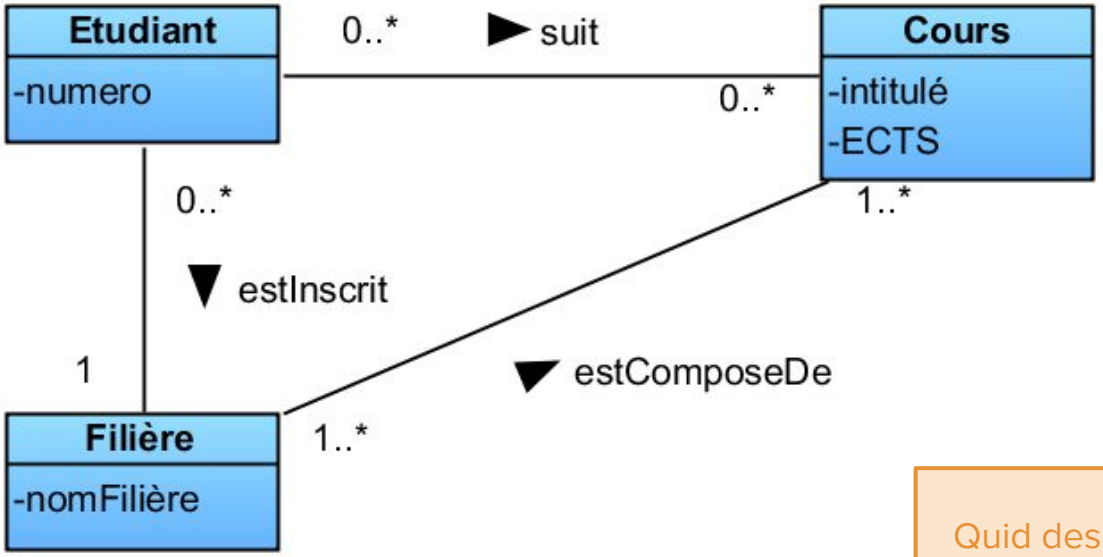
1. Un objet est un élément plus « *important* » qu'un attribut.
2. Questions :
   a. Si l'on ne peut demander à un élément que sa valeur : **attribut**.
   b. Si l'on peut lui poser plusieurs questions : **objet** qui possède à son tour plusieurs attributs, ainsi que des liens avec d'autres objets.

Etude de cas - Bibliothèque

Identifier les différentes classes du domaine d'application de la bibliothèque de l'UNamur ainsi que leurs attributs respectifs

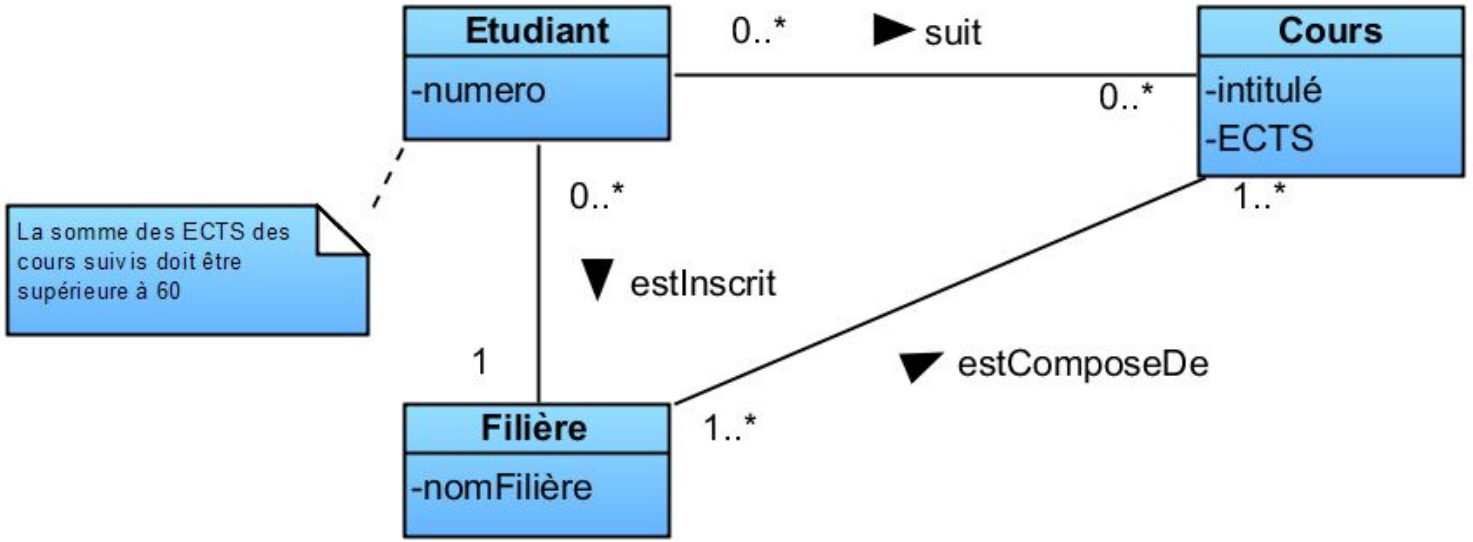
1. Les étudiants de l'université de Namur doivent s'inscrire à une seule filière qui propose un ensemble de cours.
2. Chaque cours est caractérisé par un nombre de crédits ECTS (représentant la charge de travail).
3. Au sein de ces cours, un étudiant doit en sélectionner un sous-ensemble pour l'équivalent de 60 crédits.
4. Un cours peut se retrouver dans plusieurs filières.
5. Les étudiants connus par un numéro ne peuvent suivre que des cours de leur filière (pas d'élève libre).

Etude de cas - Bibliothèque

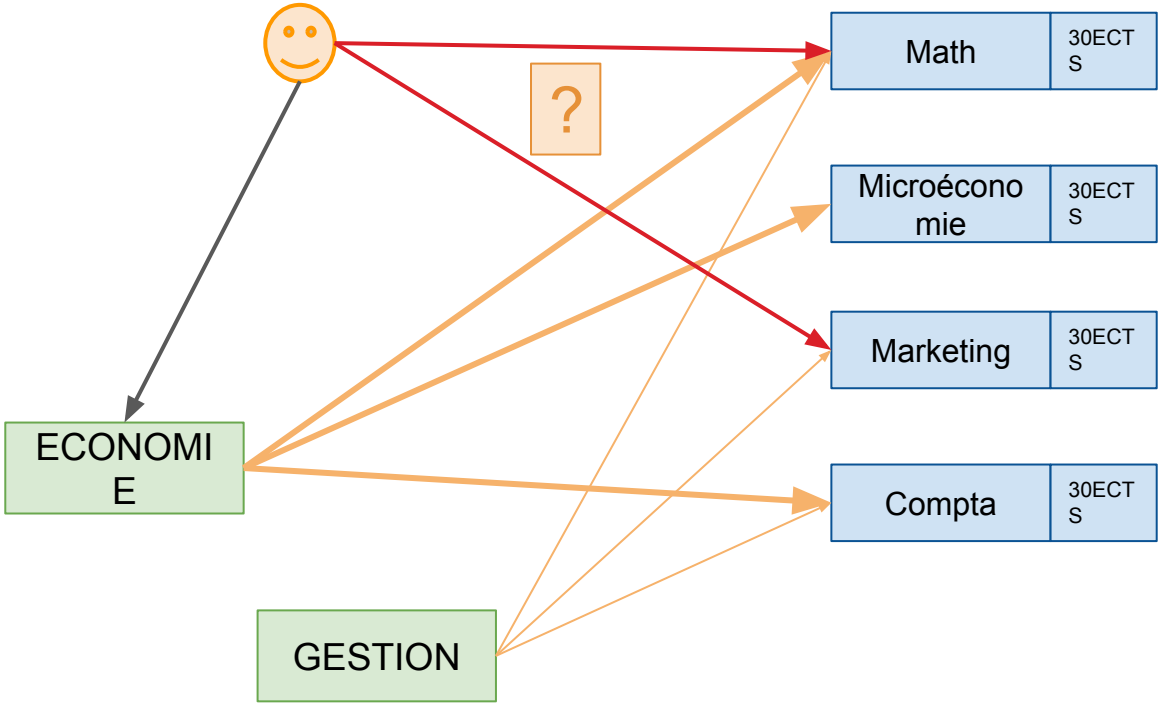


Quid des 60 ECTS minimum ?

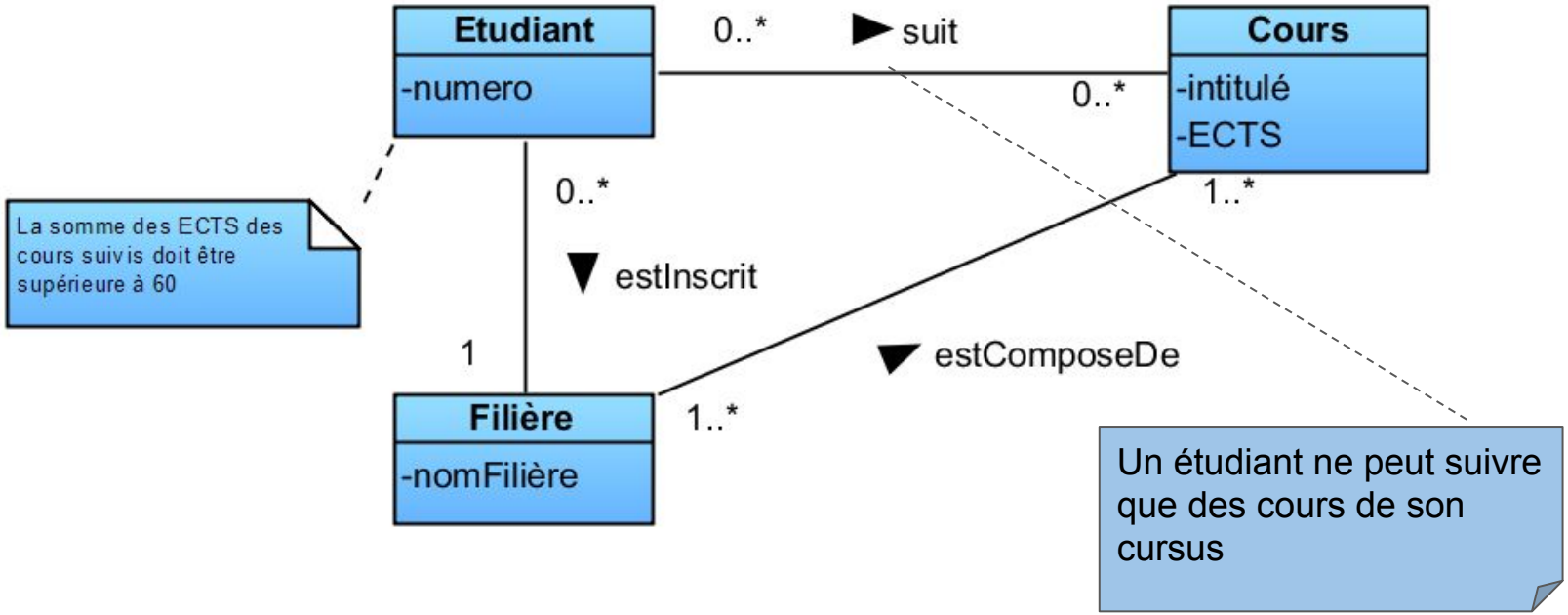
Etude de cas - Bibliothèque



Etude de cas - Bibliothèque



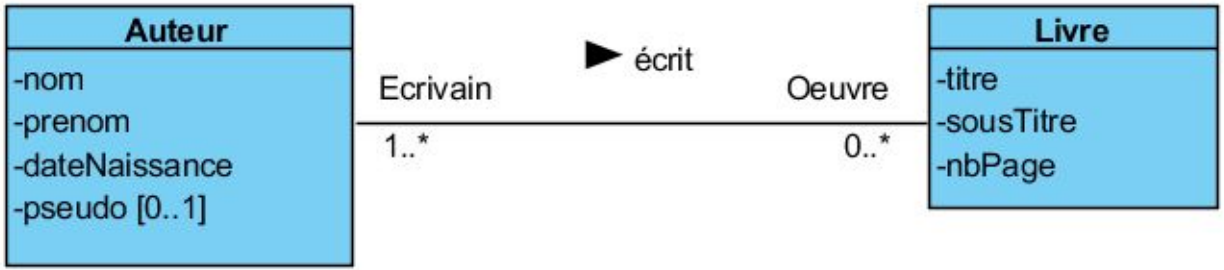
Etude de cas - Bibliothèque



Caractéristiques

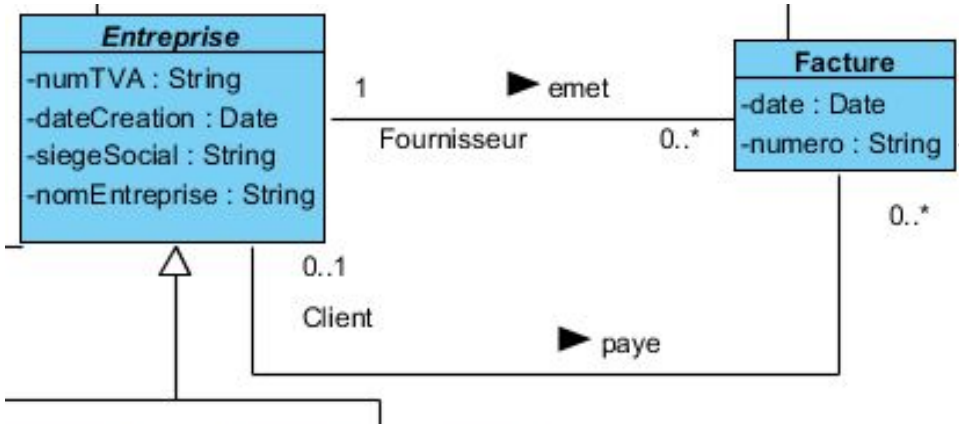
Les association peuvent être caractérisées par:

1. Un **sens de lecture**.
2. Un **rôle**.



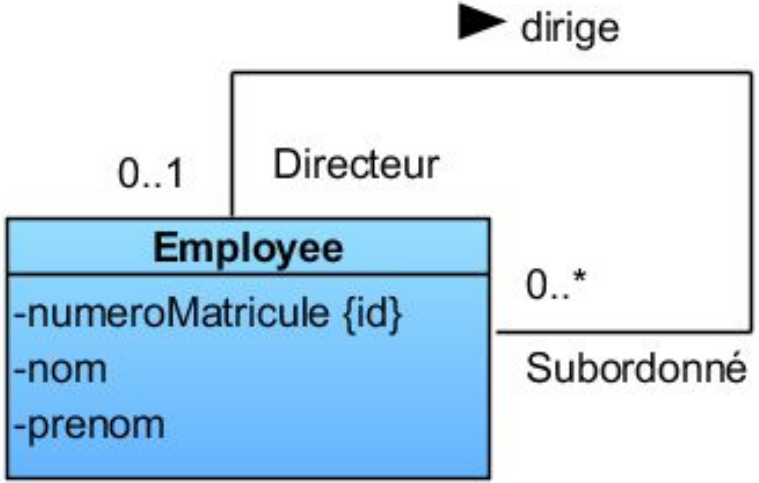
Rôle

Il est nécessaire de mettre un rôle lorsqu'une classe est lié plusieurs fois à une autre même classe.



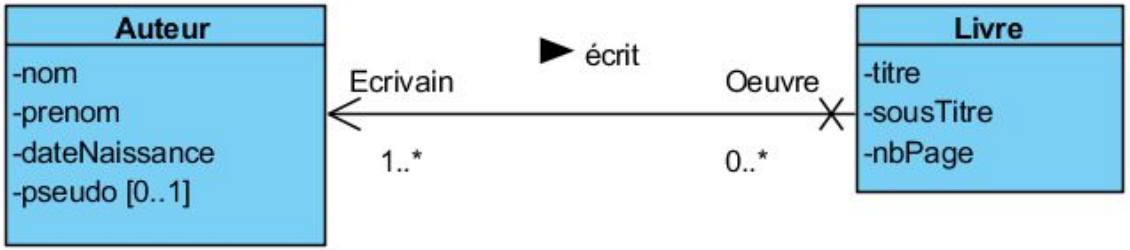
Association réflexive

- Association dont la cible est identique à la source.
- Les rôles sont obligatoires.



Sens de navigation

Les association peuvent être caractérisées par un sens de navigation qui comment naviguer entre les objets.



Sens de navigation

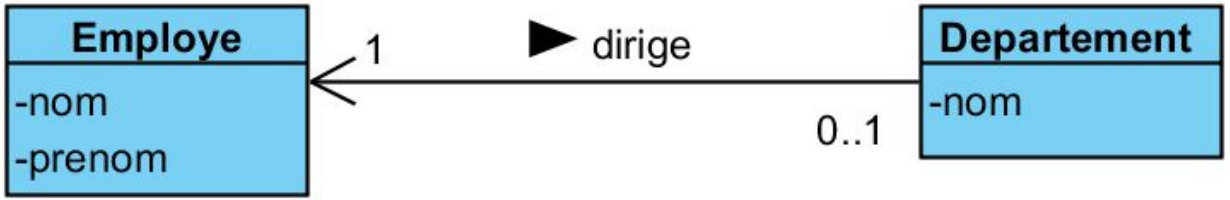
Les sens de navigation peuvent être :

1. **Non-spécifié** (choix laissé au développeur)
2. Navigable
3. Non-navigable

Sens de navigation

Les sens de navigation peuvent être :

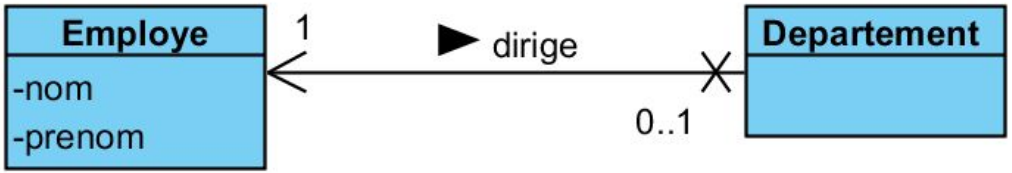
1. Non-spécifié (choix laissé au développeur)
2. **Navigable**
3. Non-navigable



Sens de navigation

Les sens de navigation peuvent être :

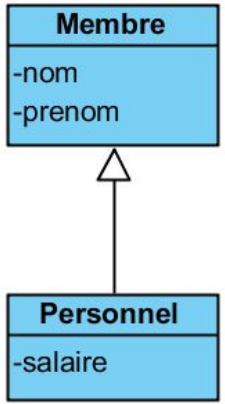
1. Non-spécifié (choix laissé au développeur)
2. Navigable
3. **Non-navigable**



Généralisation / Spécialisation

La relation de **généralisation / spécialisation** est une relation de hiérarchisation des classes d'entités.

- Les **enfants** ou **sous-classes** adoptent la fonctionnalité d'un **parent** ou **super-classe**.
- Relié avec une flèche fermée orientée vers la super-classe.



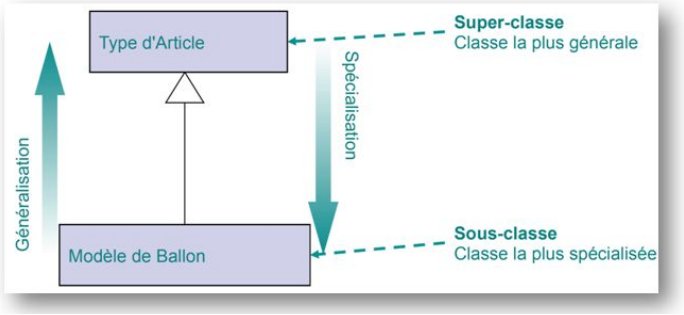
Généralisation / Spécialisation

Spécialisation

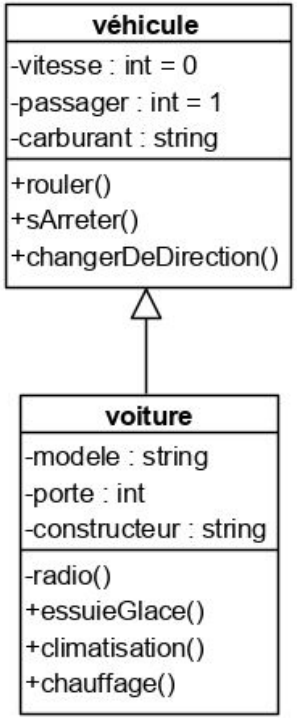
- Ajout de caractéristiques spécifiques aux sous-classes.

Généralisation

- Factorisation de caractéristiques communes (abstraction des détails).



Exemple



La classe « *Voiture* » hérite de tous les attributs et méthodes de la classe parent « *Véhicule* »

- vitesse, nombre de passagers, carburant,
- rouler(), s'arrêter(), changerDeDirection()).

En plus de ses attributs spécifiques

- type de modèle, nombre de portes, constructeur,
- radio(), essuie-glace(), climatisation/chauffage())

Classe Abstraite / Concrète

Classe Abstraite

- Toutes les entités de la super classe sont obligatoirement spécialisées.
- La classe ne peut être instanciée.
- Nom de classe en *italique*.

Classe Concrète

- La classe peut être instanciée.

Principe de Substitution de Liskov

Bonne pratique

N’employez la relation de généralisation que lorsque la sous-classe est conforme à 100 % aux spécifications de sa super-classe.

Q: *un carré est-il rectangle ?*

Principe de Substitution de Liskov

Mathématiquement parlant oui, du moins un cas particulier.

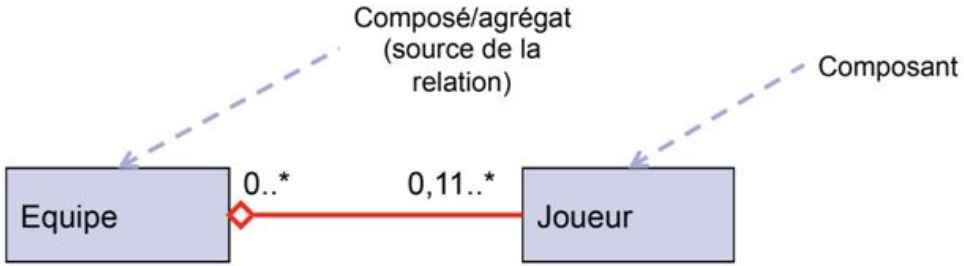
Mais faire hériter une classe Square d'une classe Rectangle n’est pas un bonne idée !

- La classe Square a un invariant particulier, à savoir sa hauteur et sa largeur doivent être égales à tout moment. La notion de hauteur ou de largeur n'a sémantiquement pas de sens dans le cas d'un carré.
- De fait, le comportement d'une classe Square va à l'encontre de celui de notre classe Rectangle.
- Le Principe de Substitution de Liskov n'est alors pas respecté.

Relation d'Agrégation/Composition

Agrégation

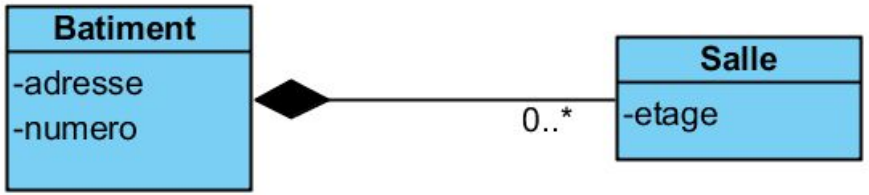
- Indique que les instances d'une classe sont les agrégats d'instance de l'autre classe.



Relation d'Agrégation/Composition

Composition

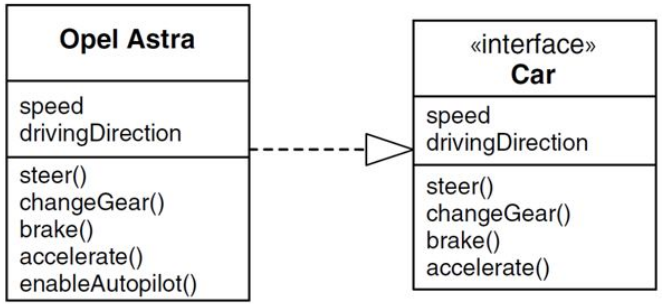
- Forme forte d'agrégation.
- L'existence du composant dépend strictement de l'existence du composé/agrégat. Il ne peut pas exister tout seul.
- La multiplicité du côté du composé vaut 1.
- Le composant ne peut être impliqué que dans une seule composition.



Interface

Ensemble des attributs et des méthodes publiques que des classes peuvent s'engager à fournir ou à exiger vis-à-vis de l'extérieur.

- Contrat qu'une classe s'engage à respecter.
- Semblables à des classes, sauf qu'une classe peut avoir une instance de son type et qu'une interface doit compter au moins une classe pour la mettre en œuvre.



Conventions de nommage

Les noms des classes (**MaClasse**) qui

- commencent par une majuscule
- peuvent contenir ensuite plusieurs mots concaténés commençant par une majuscule

Les noms des attributs, des rôles, des associations et des opérations (**monAttribut**)

- commencent toujours par une minuscule

Il est préférable de ne pas utiliser d'accents ni de caractères spéciaux

Exercices

Réaliser les exercices des diagrammes de classe.