2. DIAGRAMME DE CLASSES - NOTATIONS

DÉFINITION

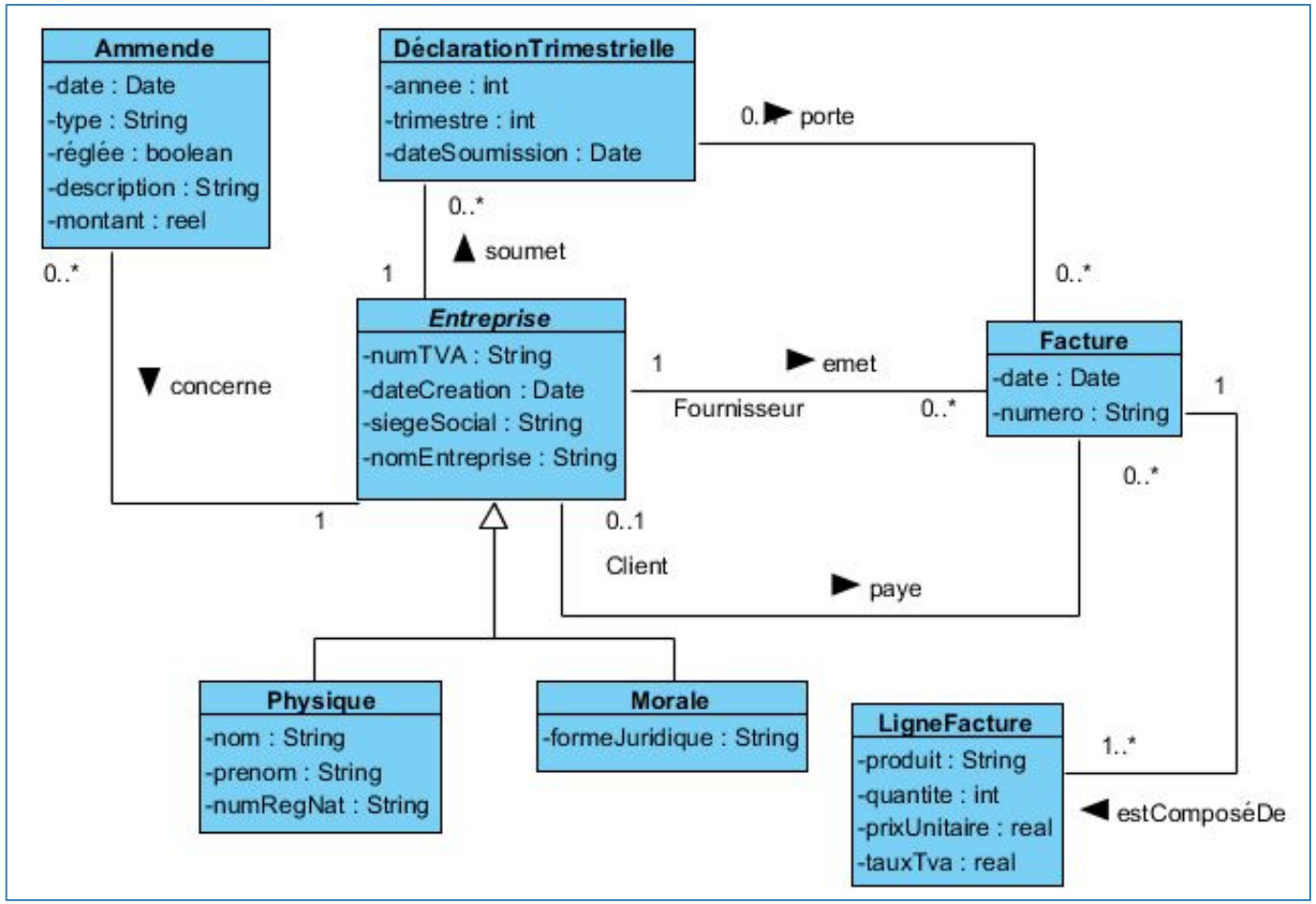
Le **diagramme de classes** est un schéma utilisé pour représenter les classes du système ainsi que les différentes relations entre celles-ci

CONTEXTE D'UTILISATION

- **Analyse (Business)**
  - décrire la structure des entités manipulées dans le domaine d'application

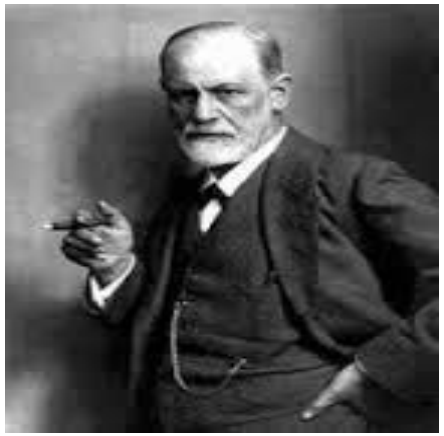
- **Conception (Système)**
  - représenter la structure d'un code orienté objet (conception)
  - représenter les classes à implémenter dans un langage de développement précis (implémentation)

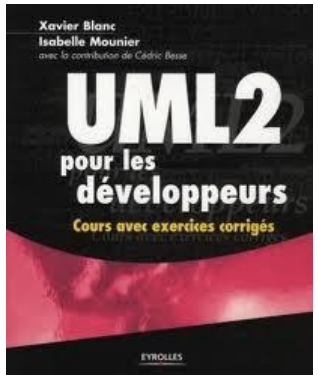
EXEMPLE

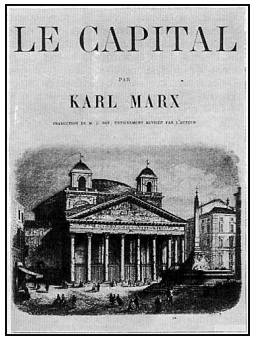

Ammende
-date : Date
-type : String
-réglée : boolean
-description : String
-montant : reel
DéclarationTrimestrielle
-annee : int
-trimestre : int
-dateSoumission : Date
0..* porte
0..*
soumet
1
0..*
0..*
concerne
Entreprise
-numTVA : String
-dateCreation : Date
-siegeSocial : String
-nomEntreprise : 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

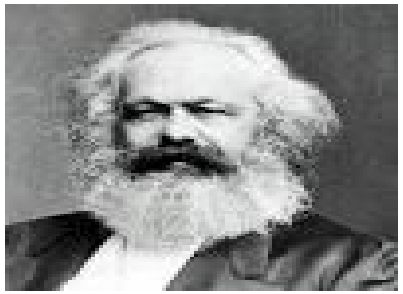
NOTION CLASSE/OBJET

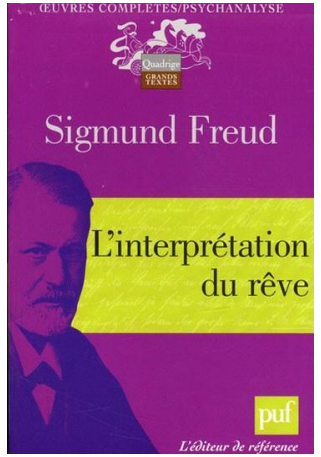
Classifier ces différents objets (entités)







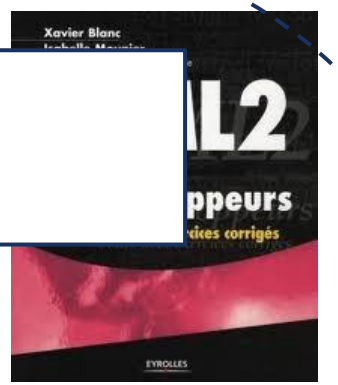


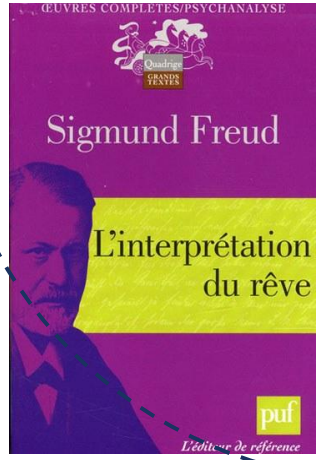


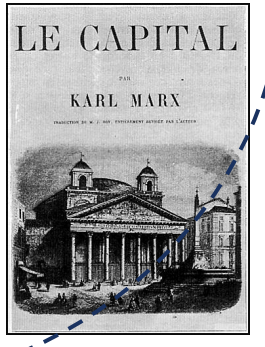
NOTION CLASSE/OBJET

Classifier ces différents objets

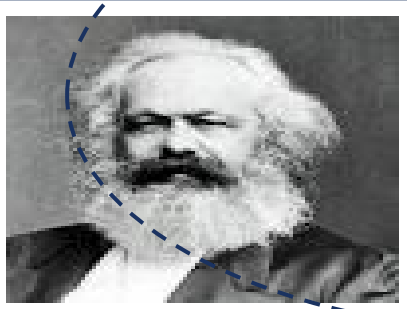
Livre

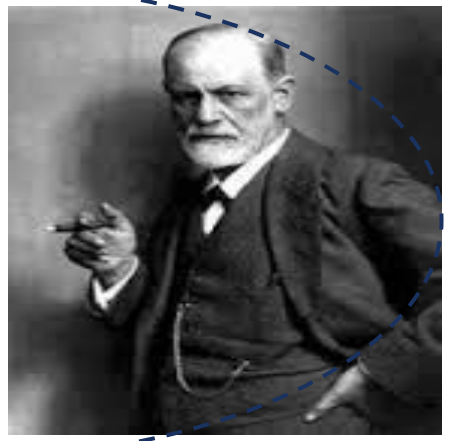






Auteur





NOTION CLASSE/OBJET

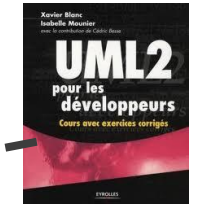
CLASSE

INSTANCE DE CLASSE

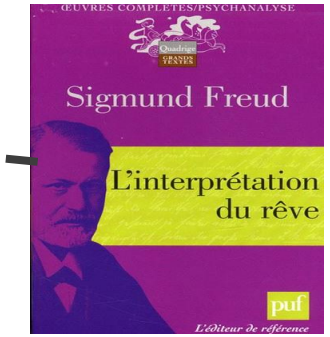
OBJET

Livre

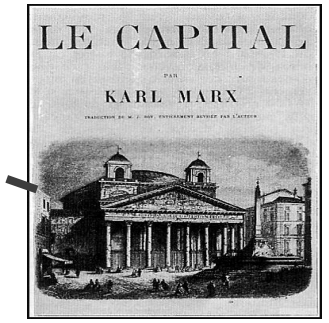
est de type



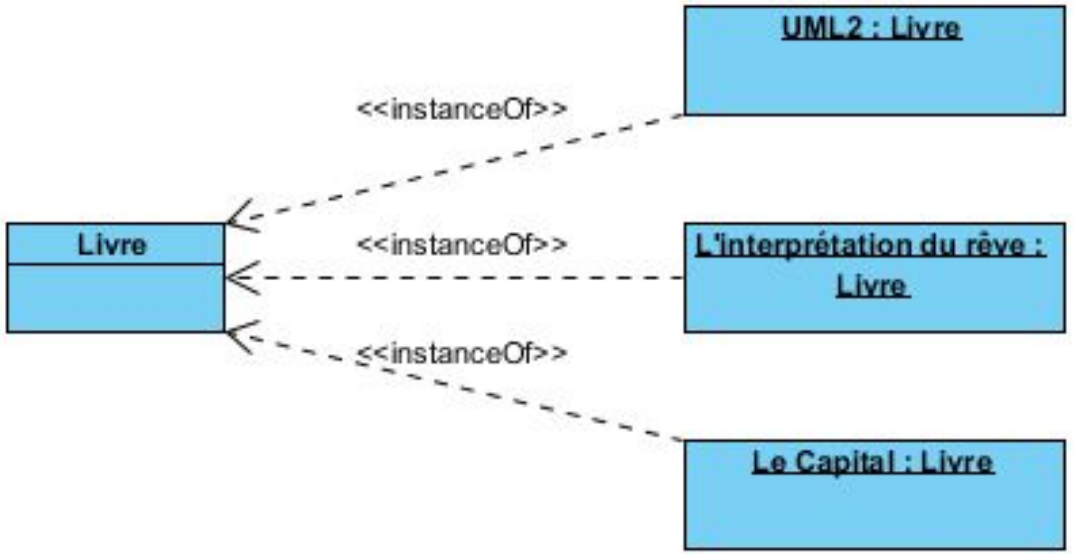
est de type



est de type



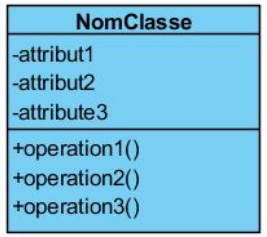
NOTION CLASSE/OBJET



CLASSE

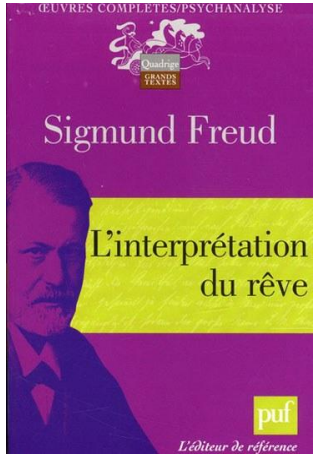
Une classe déclare des propriétés communes à un ensemble d'objets

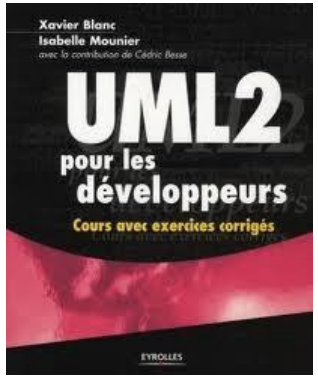
- Les attributs représentent l'état des objets
- Les opérations représentent le comportement possible des objets
- Notation:

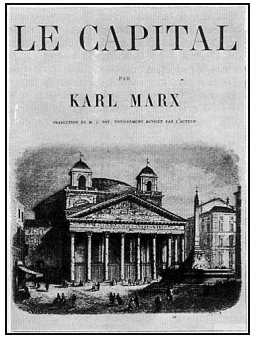


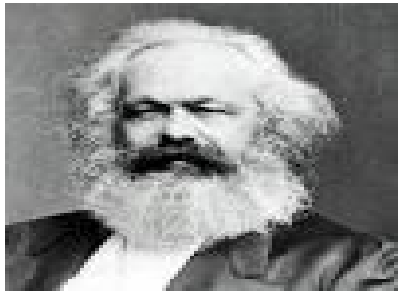
ATTRIBUTS

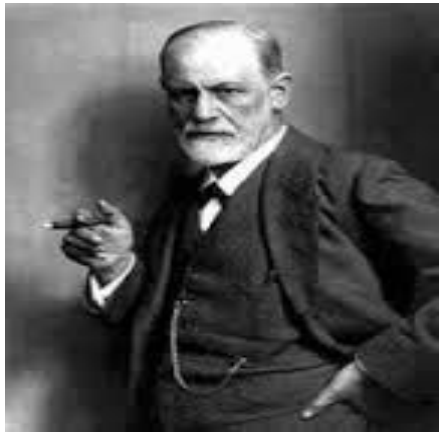
Identifier les attributs de ces 2 classes





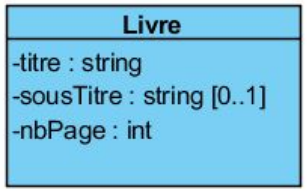






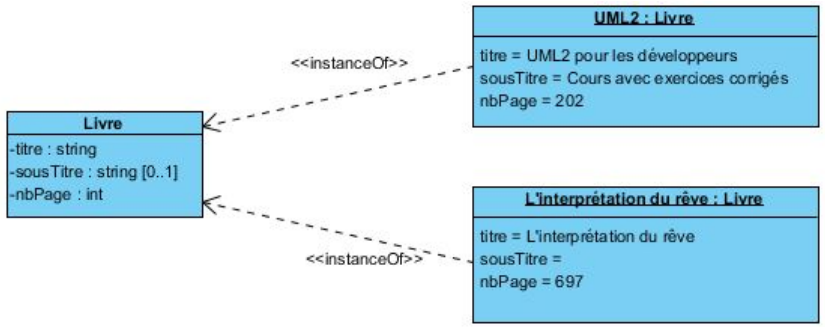
ATTRIBUTS

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



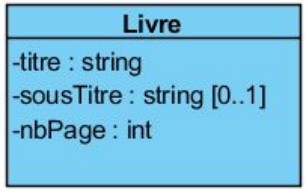
ATTRIBUTS

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



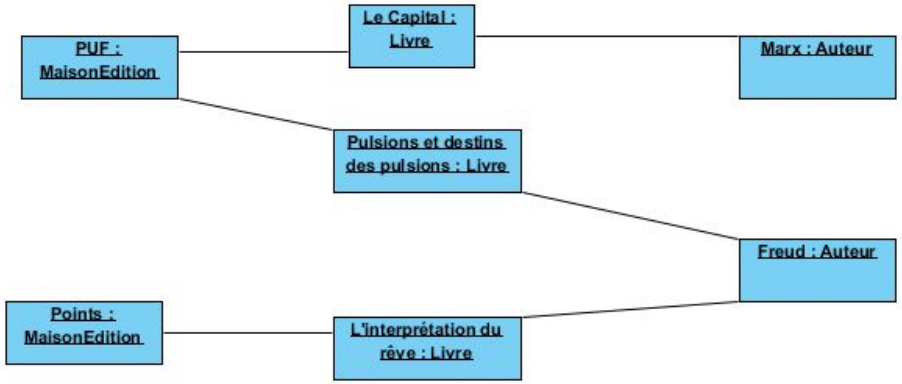
ATTRIBUTS

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



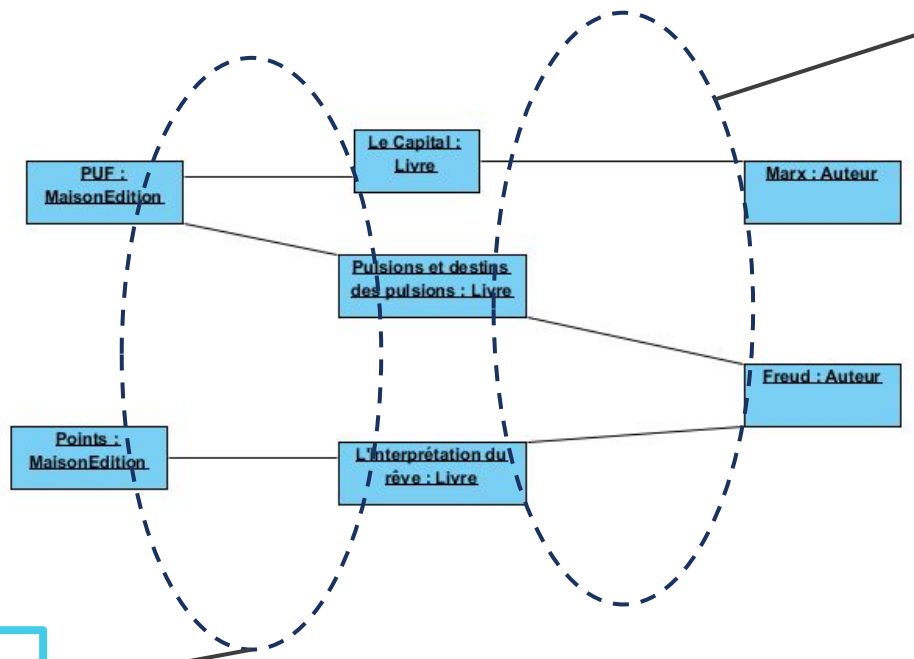
LA RELATION D'ASSOCIATION

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



LA RELATION D'ASSOCIATION

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

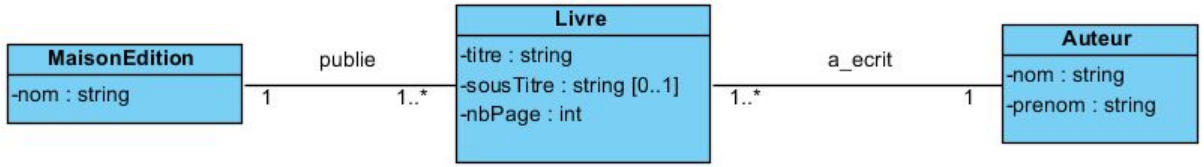


Groupe de liens identiques:
un *Auteur* **écrit** un *Livre*

Groupe de liens identiques:
une *Maison* **publie** un *Livre*

LA RELATION D'ASSOCIATION

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



LA RELATION D'ASSOCIATION

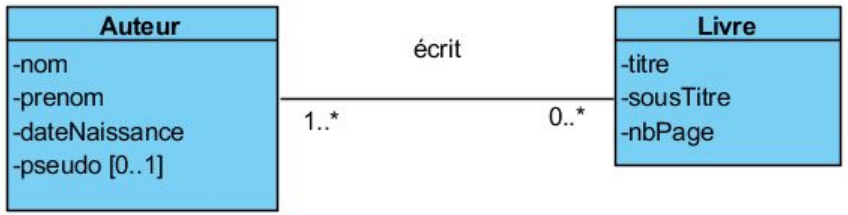
Terminologie:

- L'association est un type de **relation** (il en existe d'autres, ex.: la composition, l'agrégation)
- Une instance d'association est appelé un **lien**

LA RELATION D'ASSOCIATION

Les association sont caractérisées par:

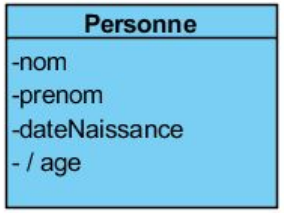
- Un nom
- Une multiplicité



ATTRIBUT DÉRIVÉ

- 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

- Identifiez l'attribut dérivé dans cette classe:



ATTRIBUT DÉRIVÉ

- **Durant l'analyse**, permet de ne pas faire de choix de prématuré
- **Durant la conception/implémentation**, 2 solutions:
  - Garder un attribut en mémoire et mettre sa valeur à jour adéquatement
    - ✓ Si calcul complexe et/ou mise à jour rare
  - 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

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
- Notation: souligné

ATTRIBUTS

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)

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

[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

ATTRIBUTS

Syntaxe et sémantique:

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

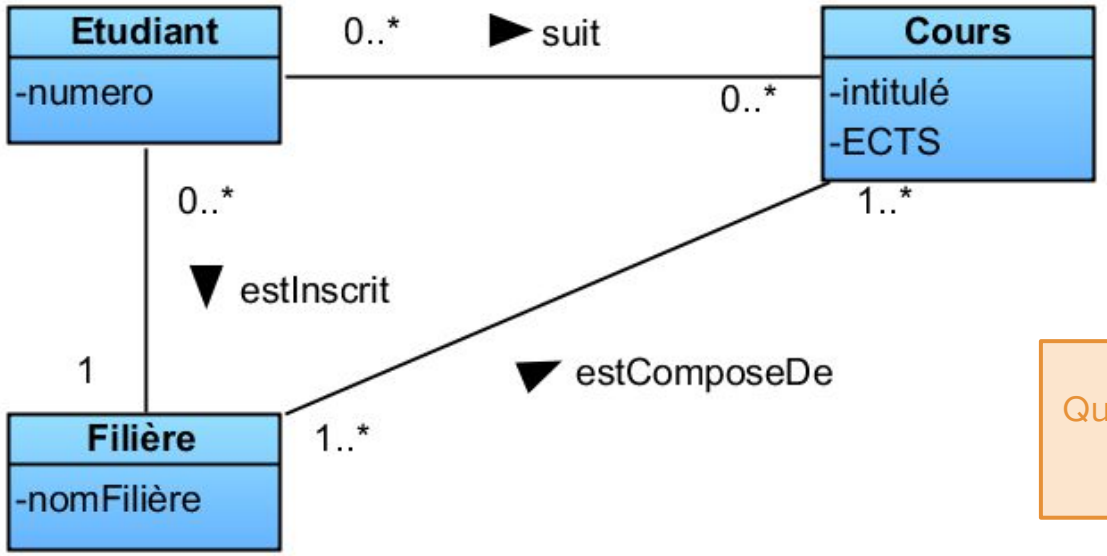
Modifier	Description
id	Property is part of the identifier for the class which owns the property.
readOnly	Property is read only (isReadOnly = true).
ordered	Property is ordered (isOrdered = true).
unique	Multi-valued property has no duplicate values (isUnique = true).
nonunique	Multi-valued property 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

LA RELATION D'ASSOCIATION

Modélisez la situation suivante:

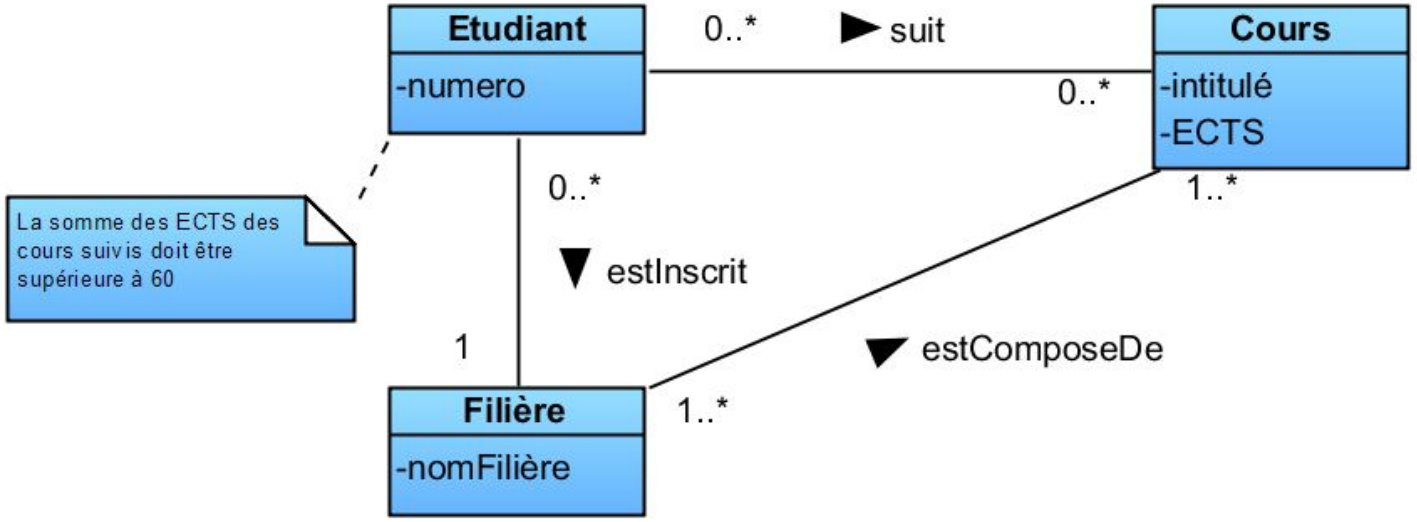
> Les étudiants de l'université de Namur doivent s'inscrire à une seule filière qui propose un ensemble de cours. Chaque cours est caractérisé par un nombre de crédits ECTS (représentant la charge de travail). Au sein de ces cours, un étudiant doit en sélectionner un sous-ensemble pour l'équivalent de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

CONTRAINTES D'INTÉGRITÉS SUR LES ASSOCIATIONS

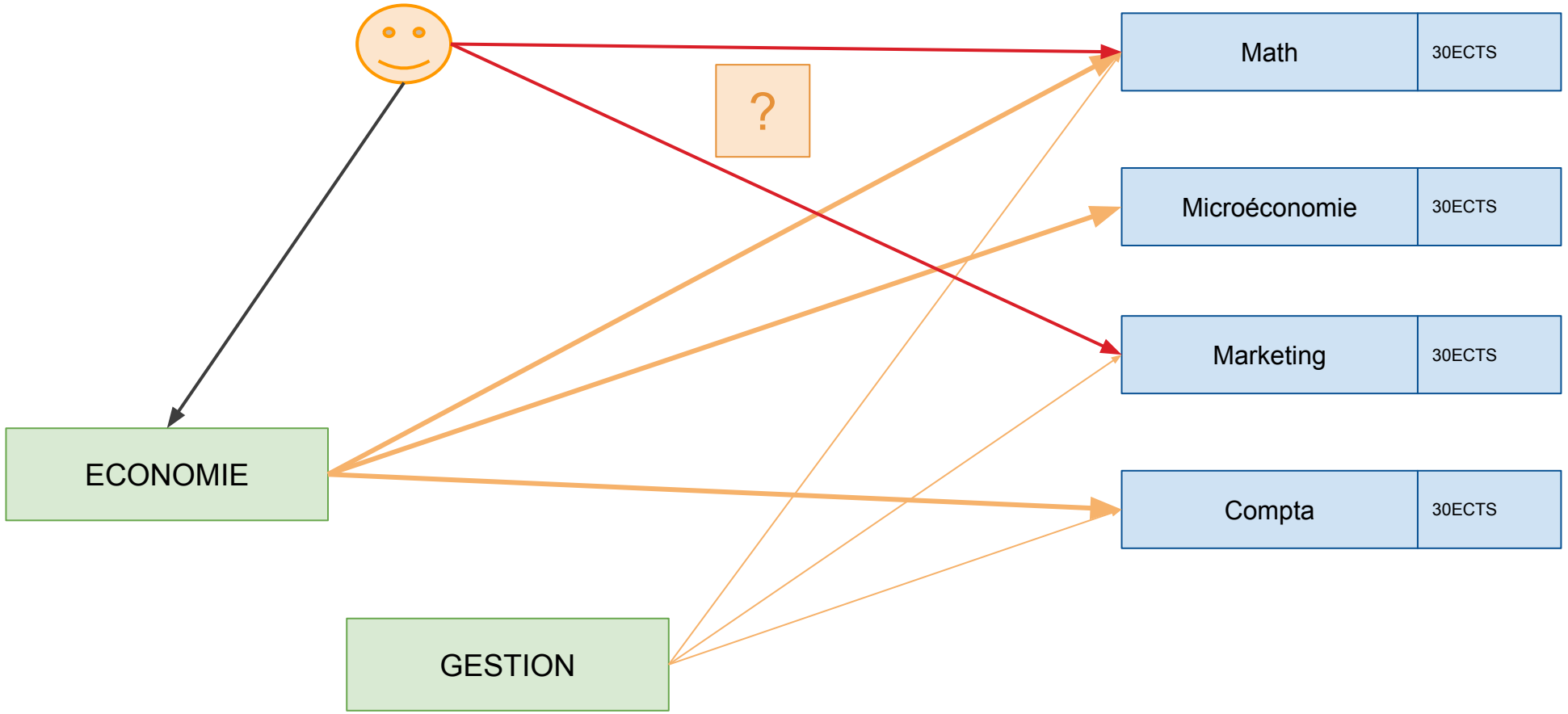


Quid des 60 ECTS minimum ?

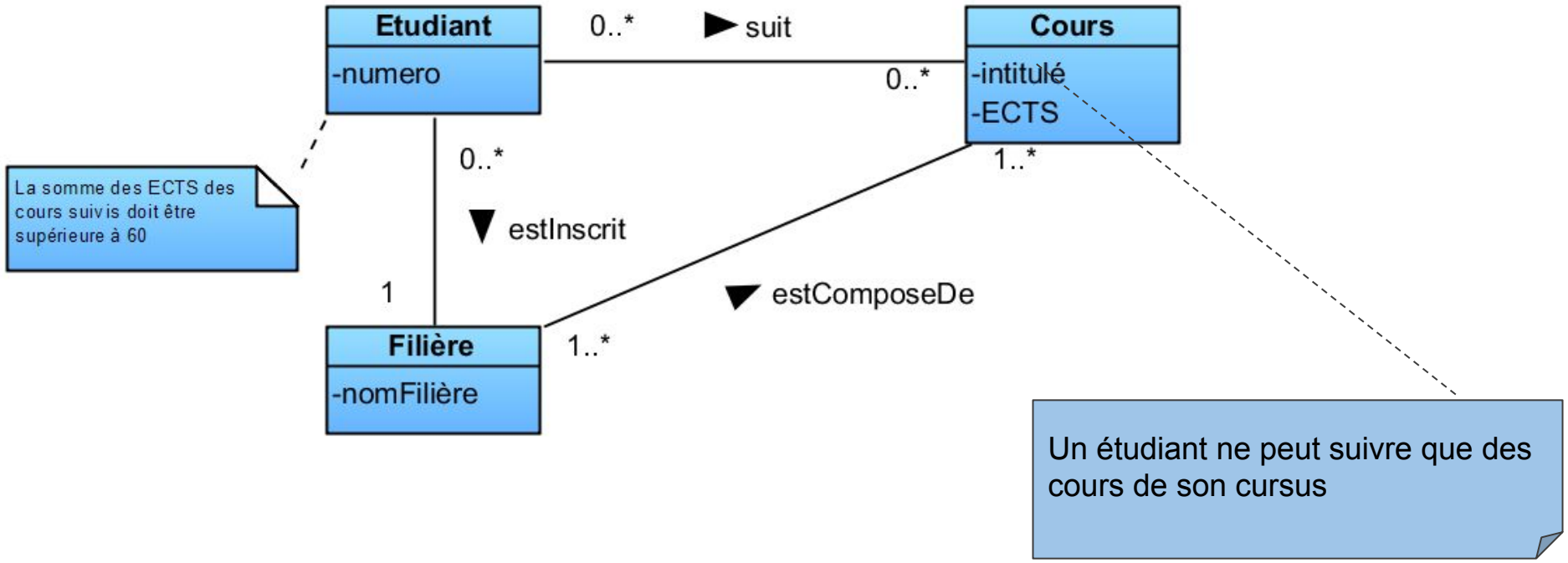
CONTRAINTES D'INTÉGRITÉS SUR LES ASSOCIATIONS



CONTRAINTES D'INTÉGRITÉS SUR LES ASSOCIATIONS



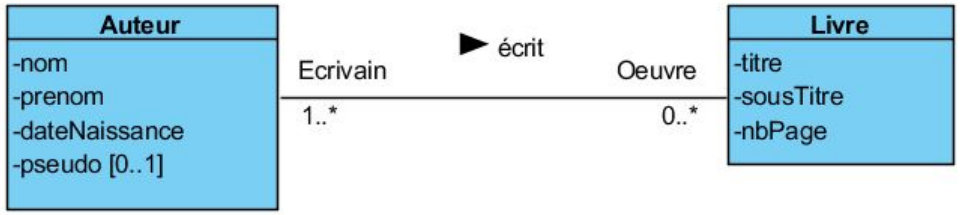
CONTRAINTES D'INTÉGRITÉS SUR LES ASSOCIATIONS



LA RELATION D'ASSOCIATION

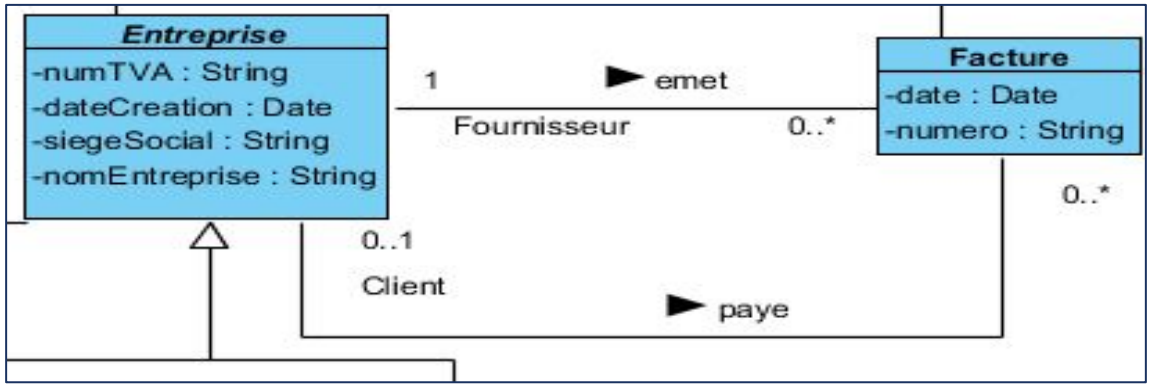
Les association *peuvent être* caractérisées par:

- Un sens de lecture
- Un rôle



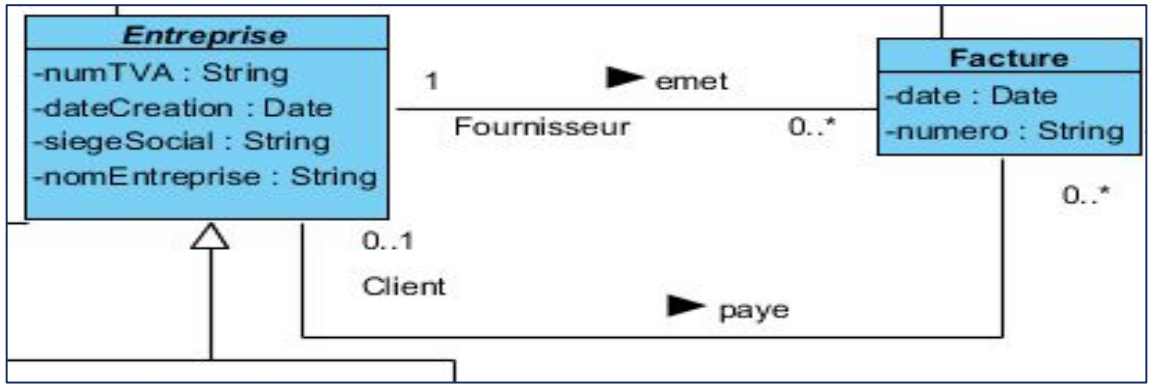
LA RELATION D'ASSOCIATION

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



LA RELATION D'ASSOCIATION

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

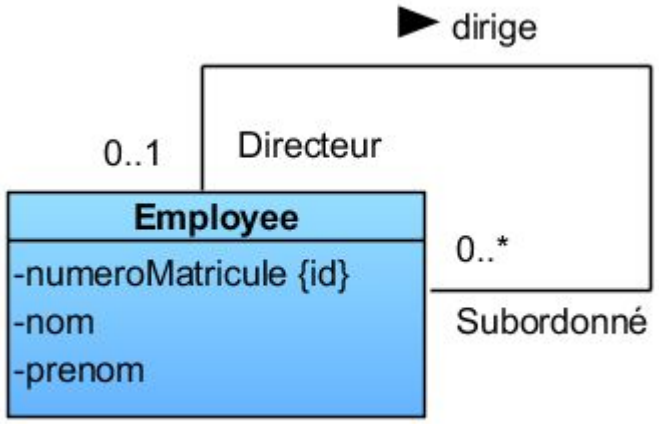



```
class abstract Entreprise {

    private String numTVA;
    private Date dateCreation;
    private String siegeSocial;
    private String nomEntreprise;

    /*
    ...

    */

}

class Facture {

    private String numero;
    private Date date;
    private Entreprise fournisseur;
    private Entreprise client;

    /*
    ...

    */

}
```

LA RELATION D'ASSOCIATION

Association réflexive

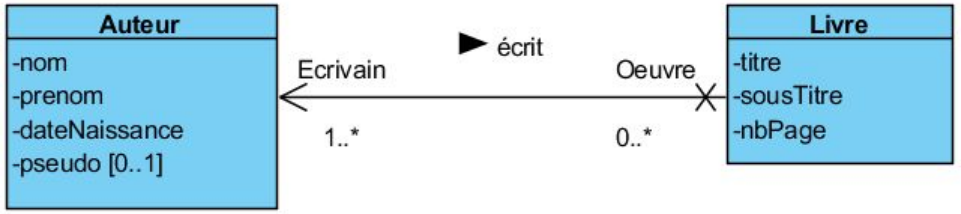
- Association dont la cible est identique à la source
- Les rôles sont obligatoires



LA RELATION D'ASSOCIATION

Les association *peuvent être* caractérisées par:

- Un sens de navigation



LA RELATION D'ASSOCIATION

Sens de navigation : Comment naviguer entre les objets


```
class Employe{


    private String nom;
    private String prenom;

    // Methods
    // ...

}
```

```
class Departement{


    private String nom;

    // Methods
    // ...

}
```

LA RELATION D'ASSOCIATION

Sens de navigation : Non-specifié (Choix laissé au développeur)


```java
class Employe{


    private String nom;
    private String prenom;
    //private Departement dirige;

    // Methods
    // ...

}
```

```java
class Departement{


    private String nom;
    //private Employe dirige;

    // Methods
    // ...

}
```

LA RELATION D'ASSOCIATION

Sens de navigation : Navigable


```
class Employe{


    private String nom;
    private String prenom;
    //private Departement dirige;

    // Methods
    // ...

}
```

```
class Departement{

    private String nom;
    private Employe dirige;

    // Methods
    // ...

}
```

LA RELATION D'ASSOCIATION

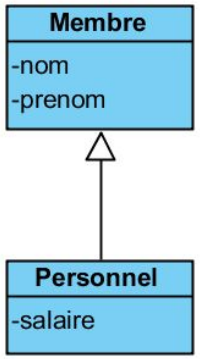
Sens de navigation : Non-navigable


```
class Employe{

    private String nom;
    private String prenom;


    // Methods
    // ...


}
```

```
class Departement{

    private String nom;
    private Employe dirige;

    // Methods
    // ...

}
```

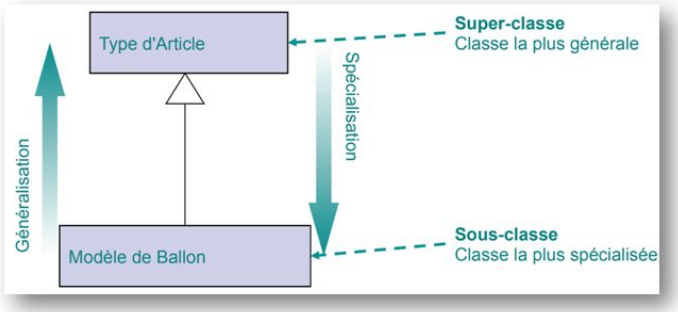
GÉNÉRALISATION/SPÉCIALISATION

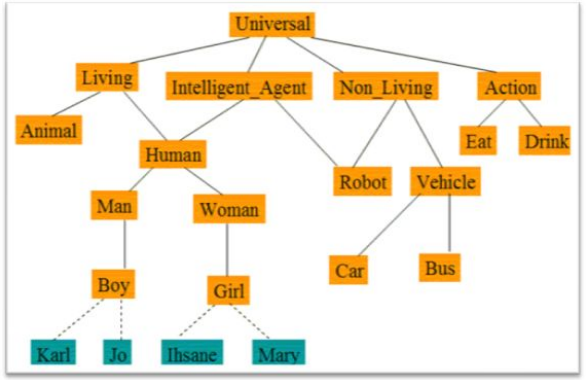
- La relation de généralisation/spécialisation est une relation de hiérarchisation des classes d'entités
- Classification des objets dans différentes classes positionnées dans différents niveaux hiérarchiques
- Représentation:



GÉNÉRALISATION/SPÉCIALISATION

- Spécialisation: ajout de caractéristiques spécifiques aux sous-classes
- Généralisation: Factorisation de caractéristiques communes (abstraction des détails)





ENSEMBLE DE GÉNÉRALISATION

- {complete, disjoint}
  - Indicates the generalization set is covering and its specific Classifiers have no common instances.
- {incomplete, disjoint}
  - Indicates the generalization set is not covering and its specific Classifiers have no common instances ***(default)***.
- {complete, overlapping}
  - Indicates the generalization set is covering and its specific Classifiers do share common instances.
- {incomplete, overlapping}
  - Indicates the generalization set is not covering and its specific Classifiers do share common instances.

GÉNÉRALISATION/SPÉCIALISATION

Classe Abstraite:

Toutes les entités de la super classe sont obligatoirement spécialisées (la classe ne peut être instanciée)

Classe Concrète:

La classe peut être instanciée

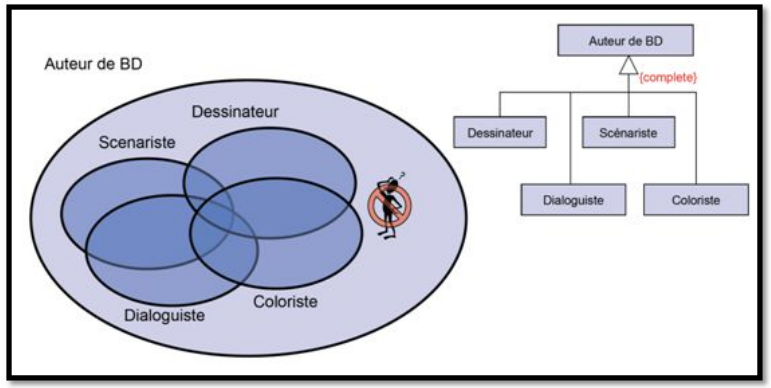
GÉNÉRALISATION/SPÉCIALISATION

Classe Abstraite:

Toutes les entités de la super classe sont obligatoirement spécialisées (la classe ne peut être instanciée)

Classe Concrète:

La classe peut être instanciée



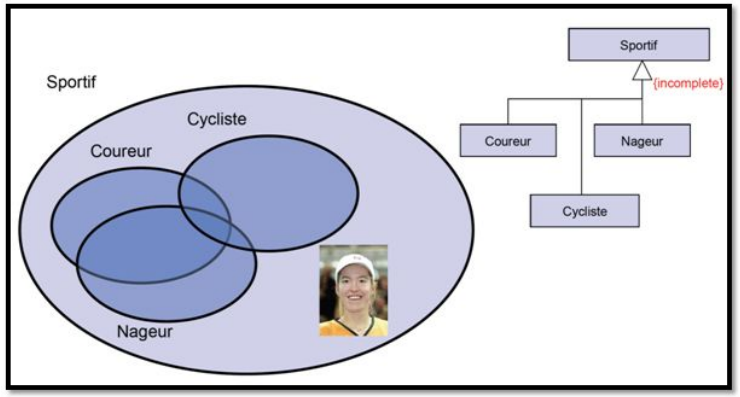
GÉNÉRALISATION/SPÉCIALISATION

Classe Abstraite:

Toutes les entités de la super classe sont obligatoirement spécialisées (la classe ne peut être instanciée)

Classe Concrète:

La classe peut être instanciée



GÉNÉRALISATION/SPÉCIALISATION

Représentation classe abstraite :

Nom de classe en italique

Bonne pratique:

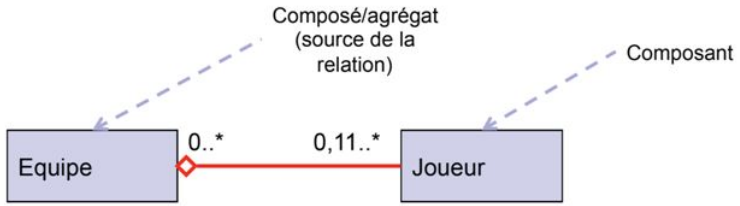
N'employez la relation de généralisation que lorsque la sous-classe est conforme à 100 % aux spécifications de sa super-classe.

RELATION D'AGRÉGATION/COMPOSITION

Agrégation

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

Représentation:

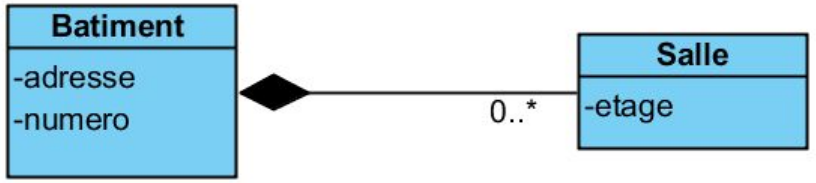


RELATION D'AGRÉGATION/COMPOSITION

Composition

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

Représentation:



OPÉRATIONS

Représente un service/une fonction permis par les instances de la classe

Définition des responsabilités:

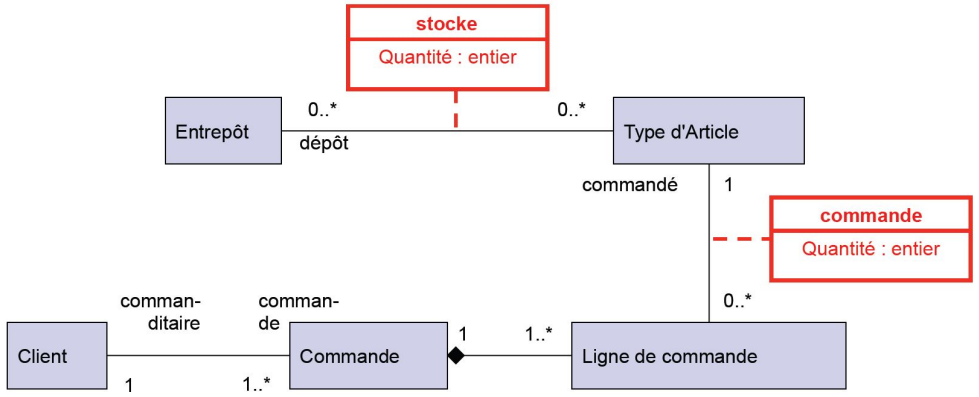
> En orienté objet, on considère que l'objet sur lequel on pourra **réaliser un traitement** doit le déclarer en tant qu'opération. Les autres objets qui posséderont une référence dessus pourront alors lui envoyer un message qui invoque cette opération.

Syntaxe:

[visibility] name (parameter list) [: type] [{property string}]

CLASSE D'ASSOCIATION

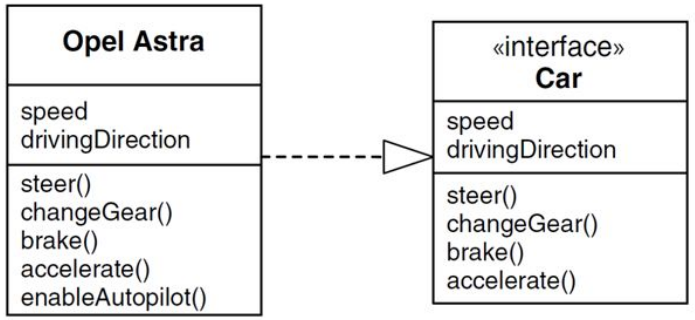
Permet de spécifier des propriétés d'une association



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



CONVENTIONS DE NOMMAGE

- Les noms des classes qui commencent par une majuscule et peuvent contenir ensuite plusieurs mots concaténés, commençant par une majuscule (**MaClasse**)
- Les noms des attributs, des rôles, des associations et des opérations commencent toujours par une minuscule (**monAttribut**)
- Il est préférable de ne pas utiliser d'accents ni de caractères spéciaux

EXERCICES

Exercices sur la prise en main du diagramme de classe