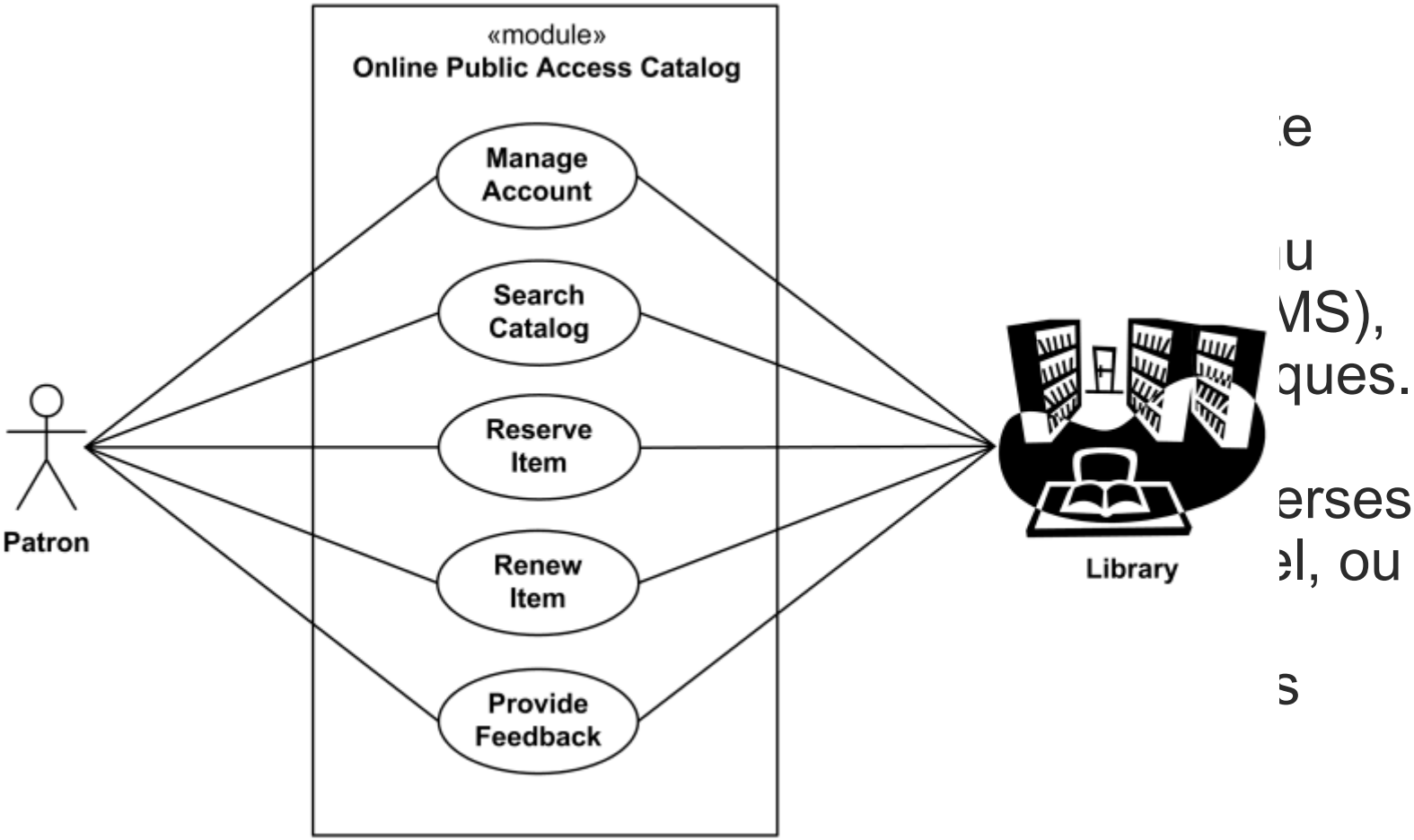
Exercices

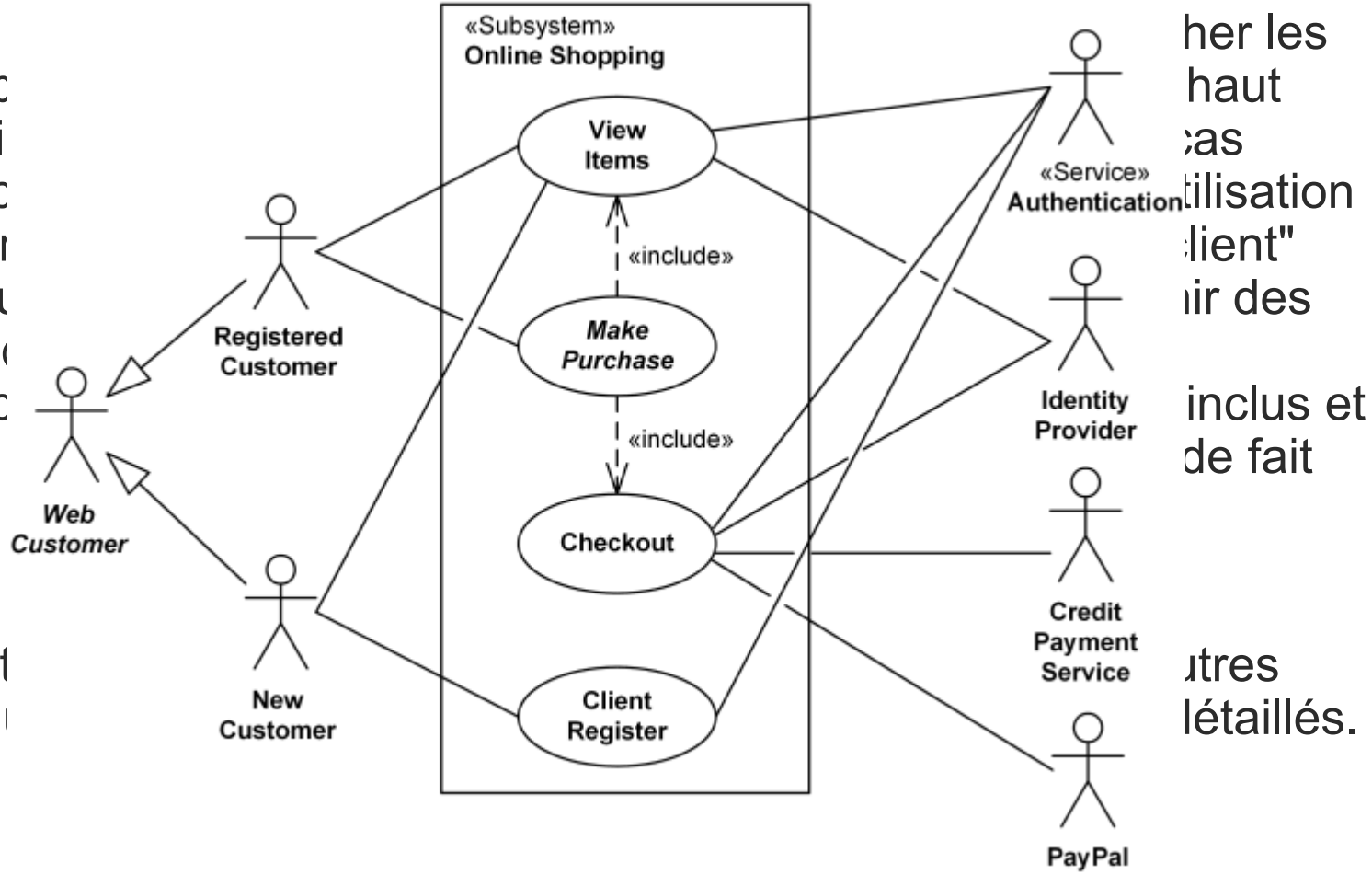
- Un
web
syst
sou
et g
Les
cata
ress
autr
peu
com

e
u
MS),
ques.
erses
el, ou
s



Online Shopping

- Le client en ligne utilise un site pour faire des achats en ligne. Les cas d'utilisation de haut niveau sont "Afficher les articles". "Effectuer un achat" et
her les
articles" p
haut
niveau s'i
cas
d'utilisatic
tilisation
"Effectuer
lient"
permet au
ir des
coupons
d'utilisatic
inclus et
n'est pas
de fait
partie de



- À l'excep
utres
acteurs q
létaillés.

Allons plus loin

- Le
  plu
  re
  ar
  pa
  d'u
  fou
  de
- Le
  da
  re
  de
  ter

t

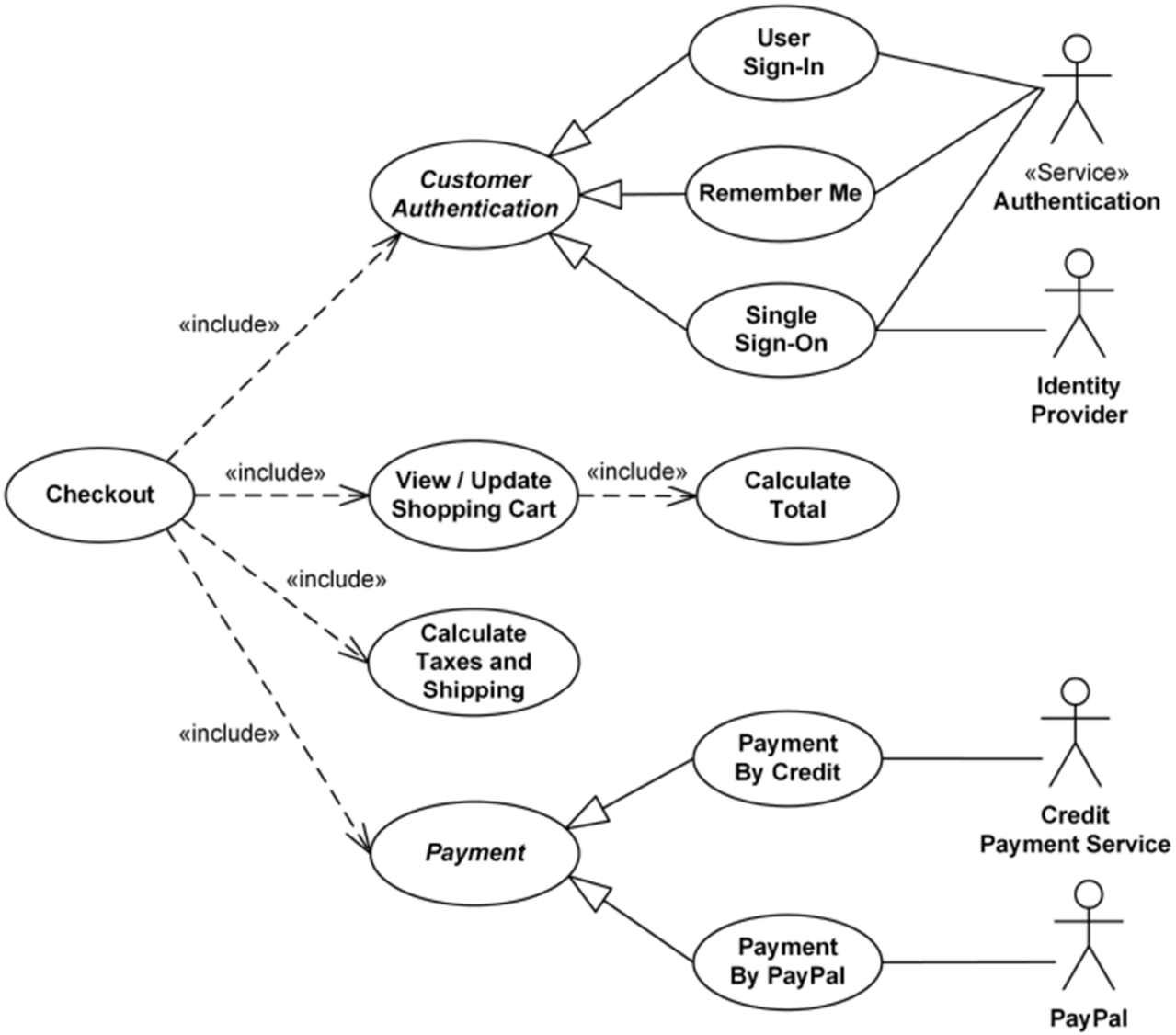
ns

authentification de l'utilisateur.

Encore plus loin ?

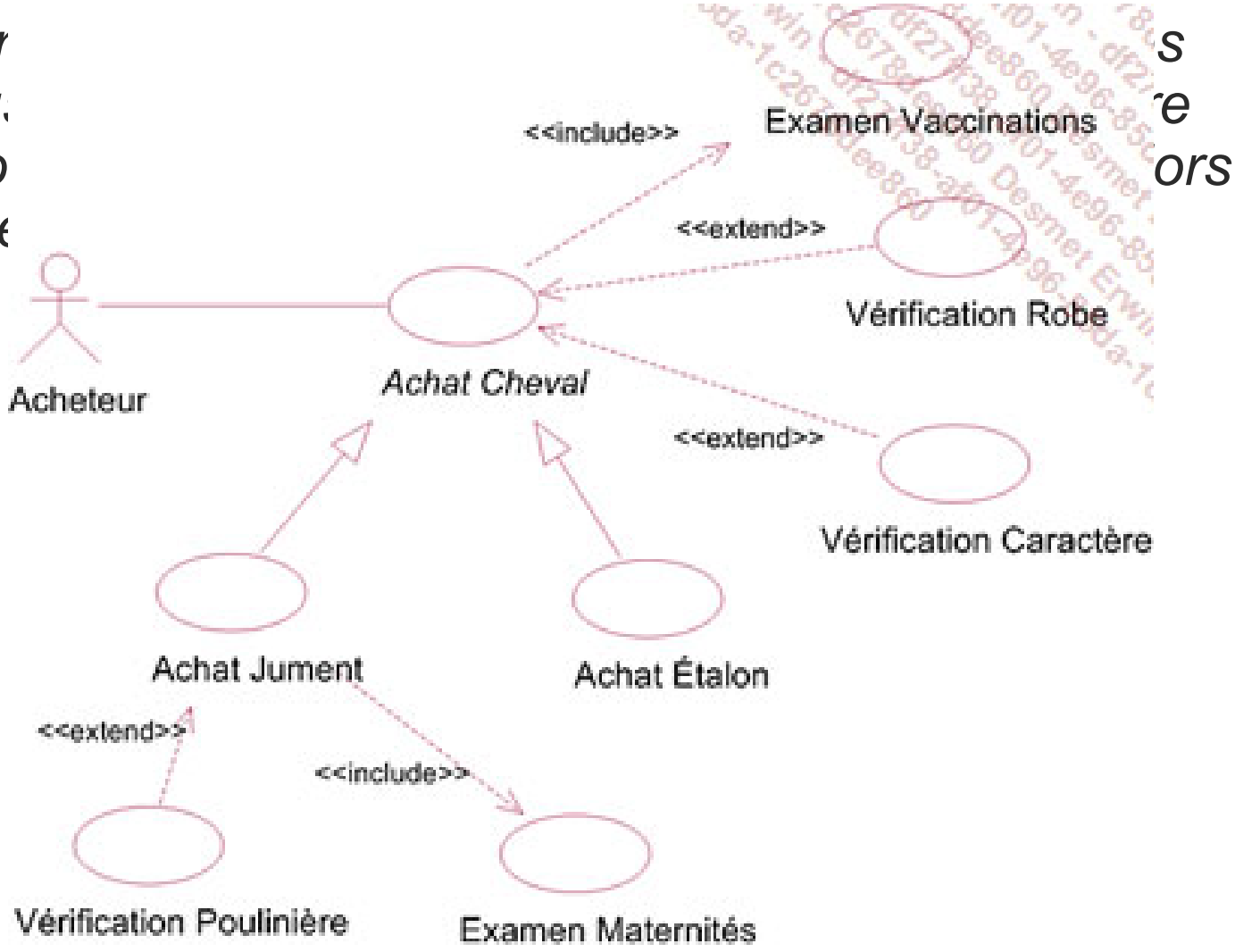
- Le
  pl
  Ce
  ur
  m
  d'
  d'
  pa

- Le
  éc
  ef
  pa



Modélisation des exigences

- *Les r… s*
  inclu… re
  facto… ors
  hérite…

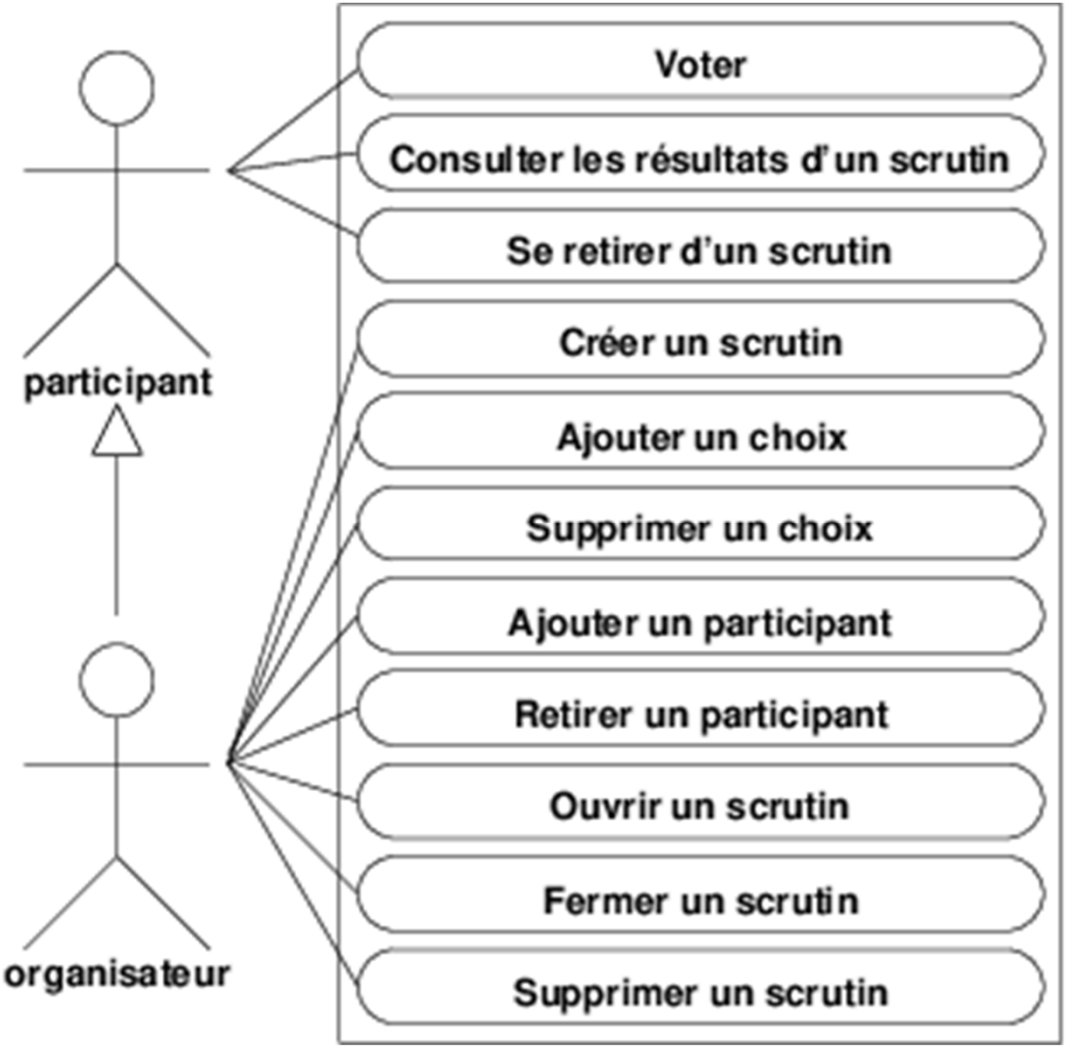


Conclusion modélisation des exigences

- Les cas d'utilisation servent à :
  - Exprimer les exigences fonctionnelles conférées au système par les utilisateurs lors de la rédaction du cahier des charges.
  - Vérifier que le système répond à ces exigences lors de la livraison.
  - Déterminer les frontières du système.
  - Écrire la documentation du système.
  - Construire les jeux de test.

- Reste proche du langage naturel → bon pour aider le client
- Ajout d'un lexique (tableau est un plus)

- https://www.youtube.com/watch?v=GC5BdRve38A&ab_channel=DelphineLonguet


Voter
Consulter les résultats d'un scrutin
Se retirer d'un scrutin
Créer un scrutin
Ajouter un choix
Supprimer un choix
Ajouter un participant
Retirer un participant
Ouvrir un scrutin
Fermer un scrutin
Supprimer un scrutin
participant
organisateur

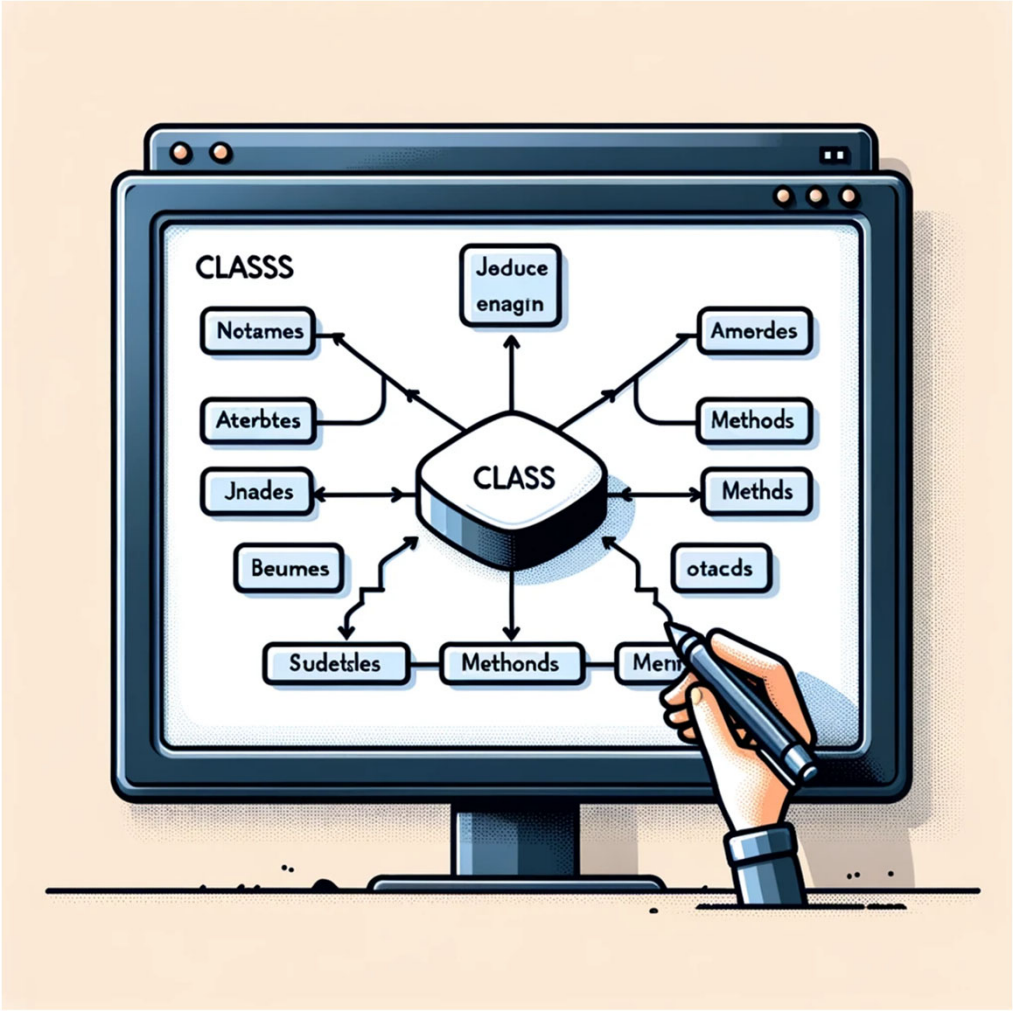
Quizz

- Use case c'est sympa même avec le tableau additionnel mais ça ne suffit pas pour développer.

- Suite de la vidéo d'aide → diagramme de seq plus tard:

  https://www.youtube.com/watch?v=1G0omjzh1OQ&ab_channel=DelphineLonguet

Les Diagrammes de Classe



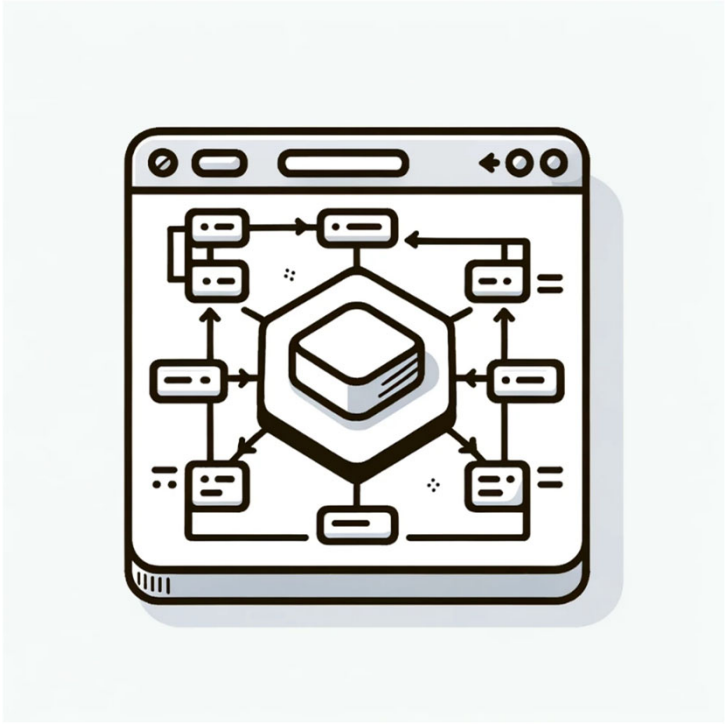
Introduction

- Importance des Diagrammes de Classe
  - Rôle dans le Développement Logiciel
  - Visualisation des Concepts de POO

- Utilité des Diagrammes de Classe:
  - Communication entre Développeurs et Parties Prenantes
  - Planification et Conception:

Le diagramme de Classe

- Définition simple
- Importance pour la modélisation
- Objectif et Utilisation
  - Modélisation de la structure
  - Illustration de la POO
- Eléments importants
  - Classes
  - Relations



Un objet c'est quoi ?

- Entité identifiable du monde réel
  - Existence physique (cheval, livre) ou pas (un texte de loi)
  - Identifiable = désignable
- En UML :
  - Attributs
  - Méthodes
- En UML : objet toujours perçu dynamique
- Objets interagissent entre eux

Abstraction

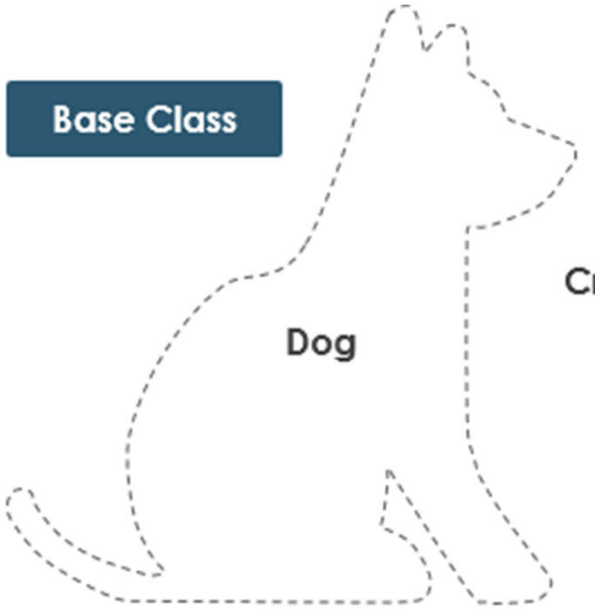
- Principe IMPORTANT
  - Retient juste les propriétés pertinentes

Exemple :

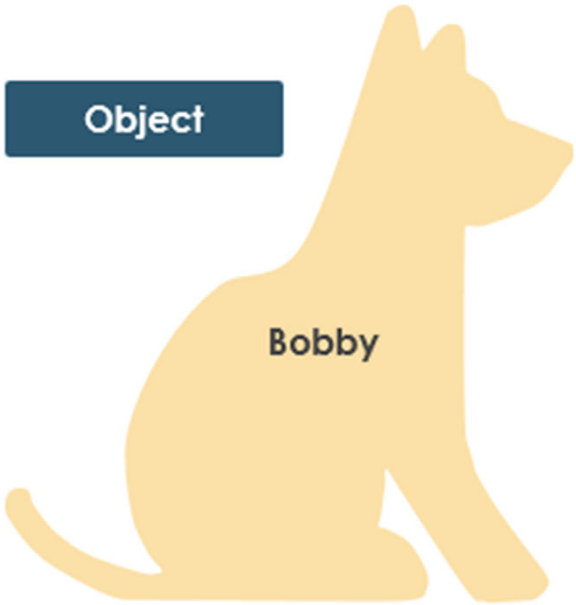
- *On s'intéresse aux chevaux pour l'activité de course. Les propriétés d'aptitude de vitesse, d'âge et d'équilibre mental ainsi que l'élevage d'origine sont pertinentes pour cette activité et sont retenues.*
- *On s'intéresse aux chevaux pour l'activité de trait. Les propriétés d'âge, de taille, de force et de corpulence sont pertinentes pour cette activité et sont retenues.*

Un classe : rappel

- **Définition:**
  - "Dans un diagramme de classe, une classe est représentée par un rectangle divisé en trois parties : le nom, les attributs, et les méthodes."
- **Importance:**
  - "Chaque classe dans le diagramme représente une entité ou un concept dans le système, avec ses propres responsabilités et caractéristiques."



Create instance



Properties	Methods
Color	Sit
Eye Color	Lay Down
Height	Shake
Length	Come
Weight	

Property Values	Methods
Color: Yellow	Sit
Eye Color: Brown	Lay Down
Height: 17 in	Shake
Length: 35 in	Come
Weight: 24 pounds	

Class without signature

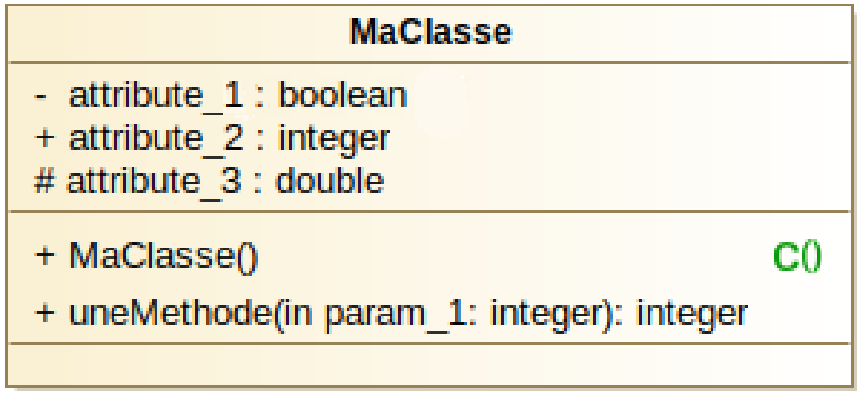
Class with signature

Avant d'aller plus loin !

- Rappel :
  - Le nom d'une classe est au singulier. Il est constitué d'un nom commun précédé ou suivi d'un ou plusieurs adjectifs qualifiant le nom. Ce nom est significatif de l'ensemble des objets constituant la classe. Il représente la nature des instances d'une classe.
  - Le nombre d'attributs et de méthodes est variable selon chaque classe. Toutefois, un nombre élevé d'attributs et/ou de méthodes est déconseillé. Il ne reflète pas, en général, une bonne conception de la classe.

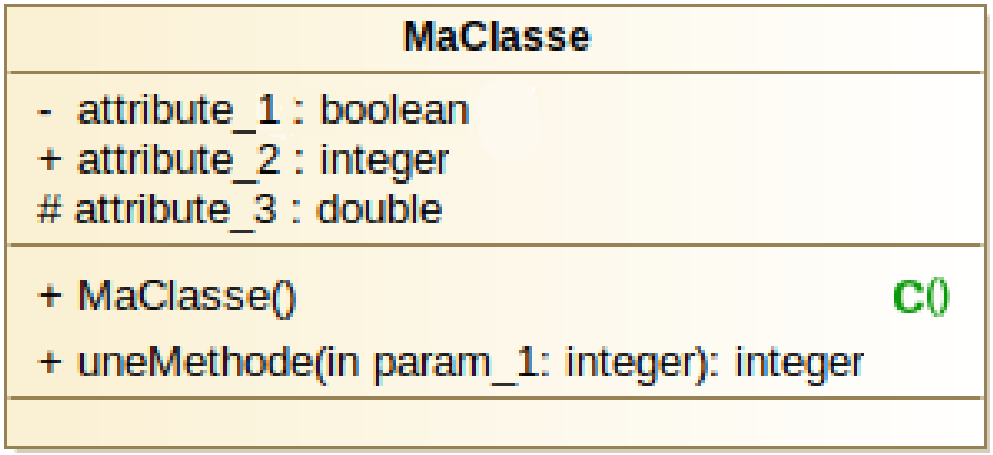
Les attributs

- **Explication:**
  - "Les attributs, listés dans la deuxième partie du rectangle, sont les propriétés ou les caractéristiques de la classe. Ils peuvent inclure des données comme l'âge, le nom, ou la taille."
- **Rôle:**
  - "Les attributs aident à définir l'état et les caractéristiques uniques de chaque objet qui sera créé à partir de la classe."

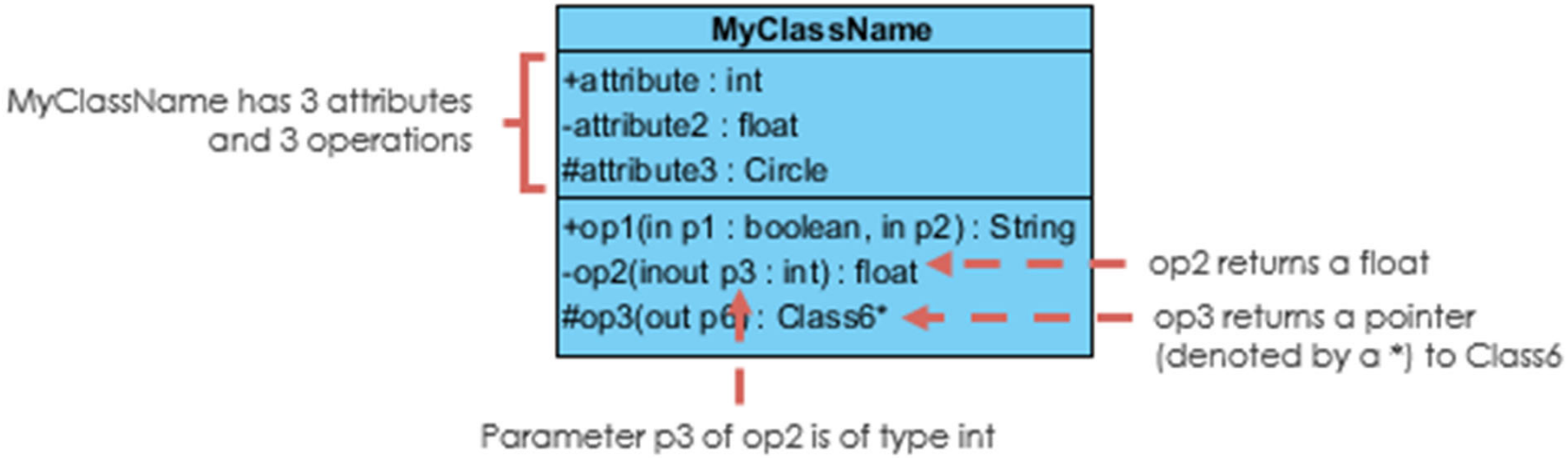


Les méthodes

- **Définition:**
  - "Les méthodes, situées dans la troisième partie, sont les actions ou les fonctions que la classe peut exécuter. Elles définissent le comportement de la classe."
- **Exemples:**
  - "Par exemple, une classe 'Voiture' pourrait avoir des méthodes comme 'démarrer', 'arrêter', et 'accélérer'."



Lire une Classe



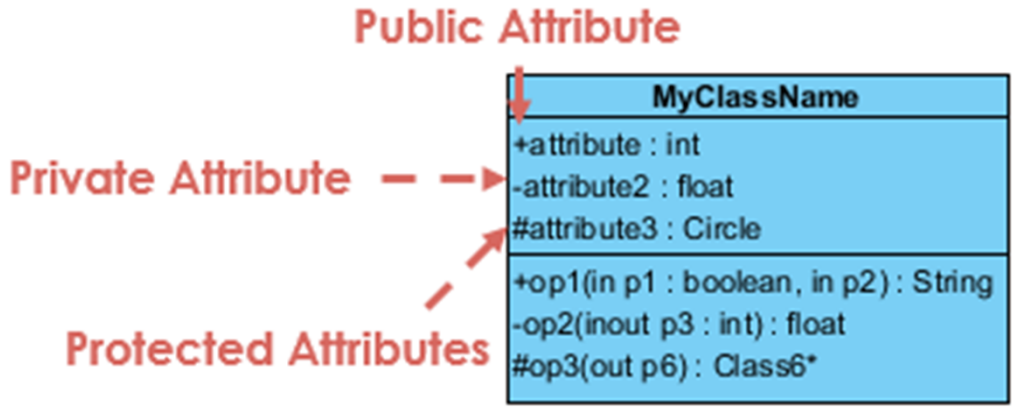
Les types

- Variable : attribut, paramètres, valeur de retour

  → Tout élément pouvant prendre une valeur
- Le type est une contrainte appliqué à une variable
- Type spécifique comme une classe
- Type standard comme un entier (Integer), un réel(Float /Real), une chaine de caractères(String), un booléen(Boolean)

La visibilité

- public (**+**) : l'élément est visible pour tous les clients de la classe ;
- protégé (**#**) : l'élément est visible pour les sous-classes de la classe ;
- privé (**-**) : l'élément n'est visible que par les objets de la classe dans laquelle il est déclaré.



L'encapsulation

- Certains attributs et méthodes visible que de l'intérieur

  On parle de méthodes/attributs privés de l'objet

  - L'attribut privé ou la méthode privée : la propriété n'est pas exposée en dehors de la classe, y compris au sein de ses sous-classes.
  - L'attribut protégé ou la méthode protégée : la propriété n'est exposée qu'aux instances de la classe et de ses sous-classes.
  - L'encapsulation de paquetage : la propriété n'est exposée qu'aux instances des classes de même paquetage. La notion de paquetage sera abordée plus tard

Petit complément

- Privé est rarement utilisée en réel
  - Instaure une différence entre les instances de classe et sous-classes
- Encapsulation de paquetage → Très liée à JAVA
- Conseil Protect → ouvrage parle souvent de privée (mais car ils mixte souvent privée-protect-paquetage)

Propriétés des variables

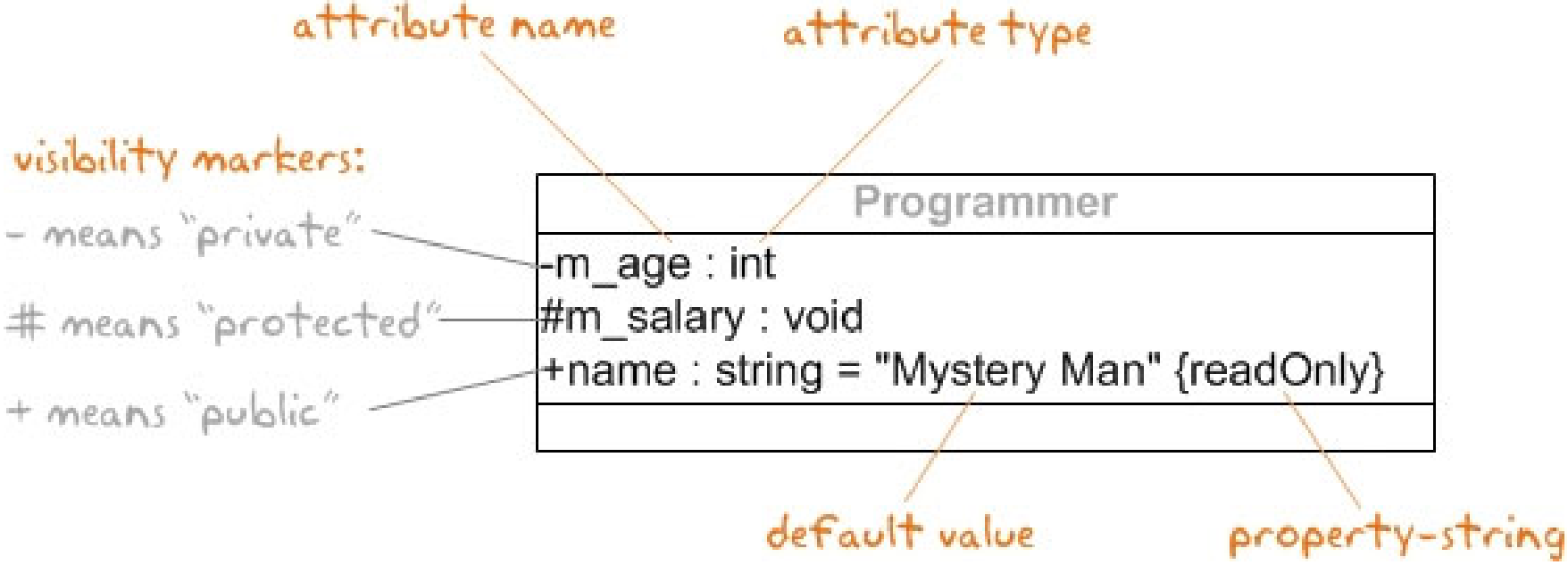
- Une propriété s'indique en accolades
  - {readOnly} : cette propriété indique que la valeur de la variable ne peut pas être modifiée. Elle doit être initialisée avec une valeur par défaut.
  - {redefines nomAttribut} : cette propriété n'est applicable qu'à un attribut. Elle spécifie la redéfinition de l'attribut de nom nomAttribut de l'une des surclasses. La redéfinition peut notamment porter sur le nom de l'attribut ou sur son type. En cas de changement de type, le nouveau type doit être compatible avec l'ancien, c'est-à-dire que son ensemble de valeurs doit être inclus dans l'ensemble des valeurs de l'ancien type.

- {ordered} : lorsqu'une variable peut contenir plusieurs valeurs (cardinalité supérieure à 1), les valeurs doivent être ordonnées.

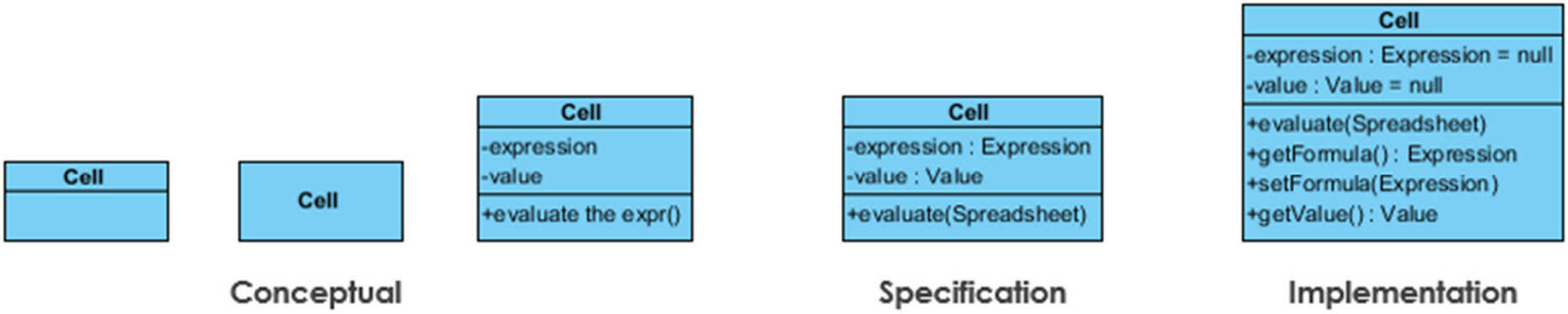
- {unique} : lorsqu'une variable peut contenir plusieurs valeurs (cardinalité supérieure à 1), chaque valeur doit être unique (interdiction d'avoir des doublons). Cette propriété s'applique par défaut.

- {nonunique} : lorsqu'une variable peut contenir plusieurs valeurs (cardinalité supérieure à 1), la présence de doublons est possible.

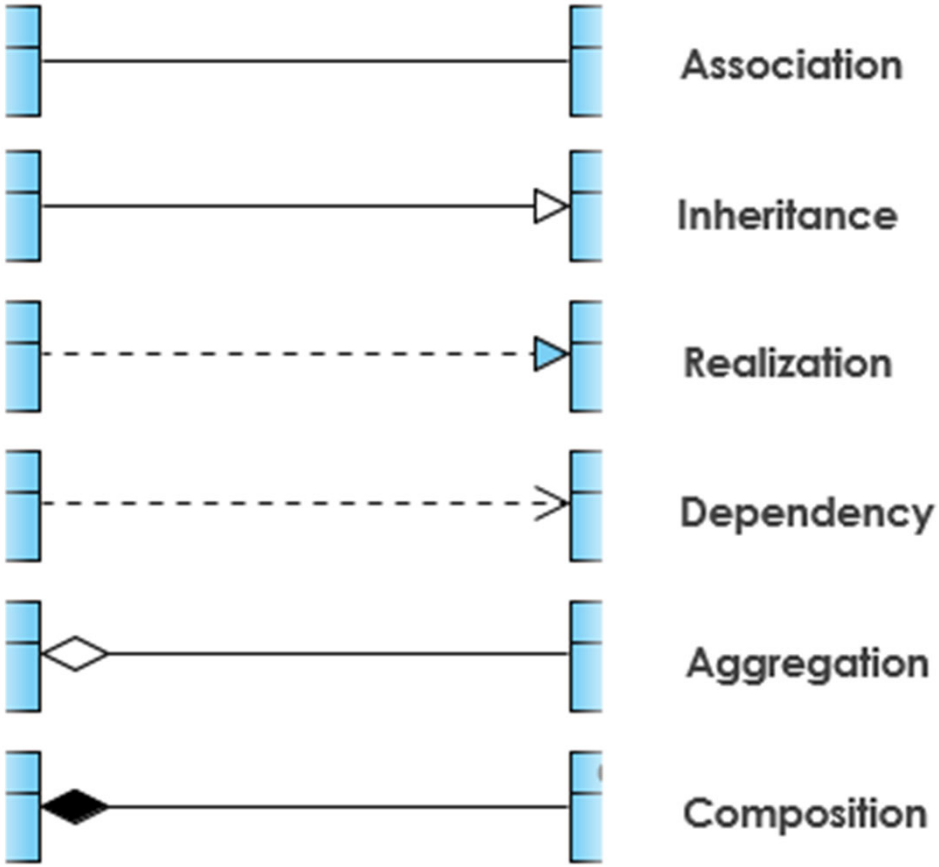
+ nom : String[1..3]{unique, ordered}


attribute name
attribute type
visibility markers:
- means "private"
means "protected"
+ means "public"
Programmer
-m_age : int
#m_salary : void
+name : string = "Mystery Man" {readOnly}
default value
property-string

Les perspectives d'une classe



Les liaisons entre les classes



- **Association** : Une ligne solide avec une flèche indique une association, qui est une connexion entre deux classes. Elle montre que les instances de ces classes peuvent être liées entre elles, mais sans spécifier la nature précise de la relation.
- **Héritage** : Représentée par une ligne solide avec une flèche en forme de triangle vide, cette relation indique qu'une classe (souvent appelée classe enfant ou sous-classe) hérite des propriétés et des méthodes d'une autre classe (classe parent ou superclasse). Cela signifie que la sous-classe est une spécialisation de la superclasse.

- **Réalisation** : Une ligne en pointillés avec une flèche en forme de triangle vide symbolise une relation de réalisation. Cela se produit généralement entre une interface et une classe qui implémente les méthodes définies dans l'interface.
- **Dépendance** : Représentée par une ligne en pointillés avec une flèche standard, cette relation signifie qu'un changement dans la définition de l'une des classes pourrait affecter l'autre classe. C'est souvent une relation d'utilisation où une classe utilise une autre temporairement ou dans un contexte particulier.

- **Agrégation** : Une ligne solide avec un losange vide indique une relation d'agrégation, qui est une forme spéciale d'association. Elle représente une relation "tout/partie" où la partie peut exister indépendamment du tout.
- **Composition** : Une ligne solide avec un losange plein représente une composition, qui est aussi une forme spéciale d'association. C'est une relation "tout/partie" forte où la partie ne peut pas exister sans le tout, signifiant que si le tout est détruit, les parties le sont également.

L’association

- **Définition:**

  "L'association est une relation qui définit une liaison entre deux classes, indiquant que des instances de l'une sont connectées à des instances de l'autre."

- **Exemple:**

  "Par exemple, dans une relation enseignant-étudiant, un enseignant peut avoir plusieurs étudiants et un étudiant peut avoir plusieurs enseignants."

- Lien : Connexion physique ou conceptuelle entre objets
  Ex : ErwinDesmet **ENSEIGNE** l'UML

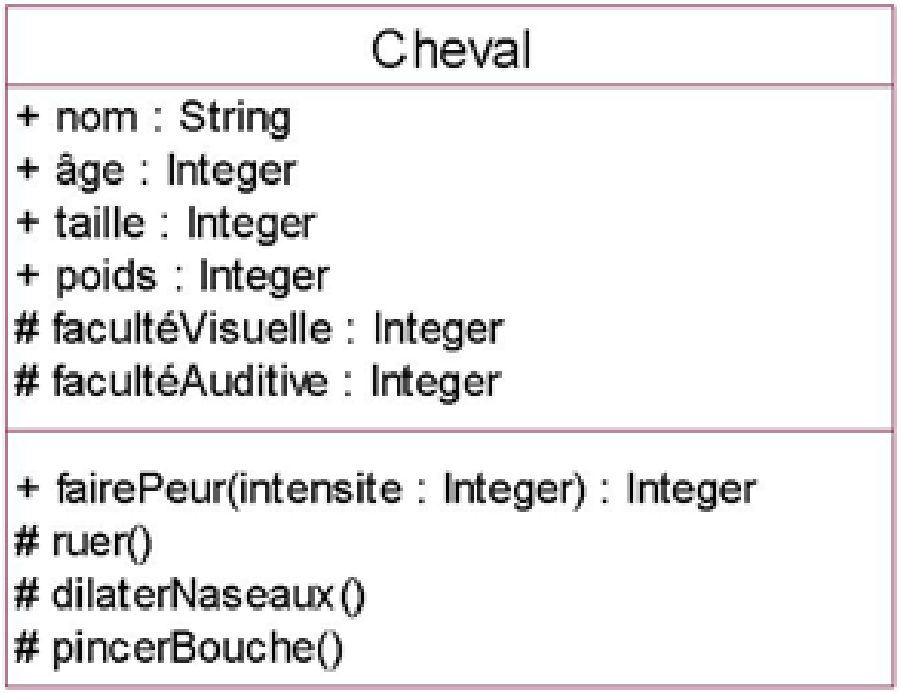
- Association : Description d'un groupe de liens qui partagent une structure et une sémantique commune
  Ex : un enseignant **ENSEIGNE** un cours

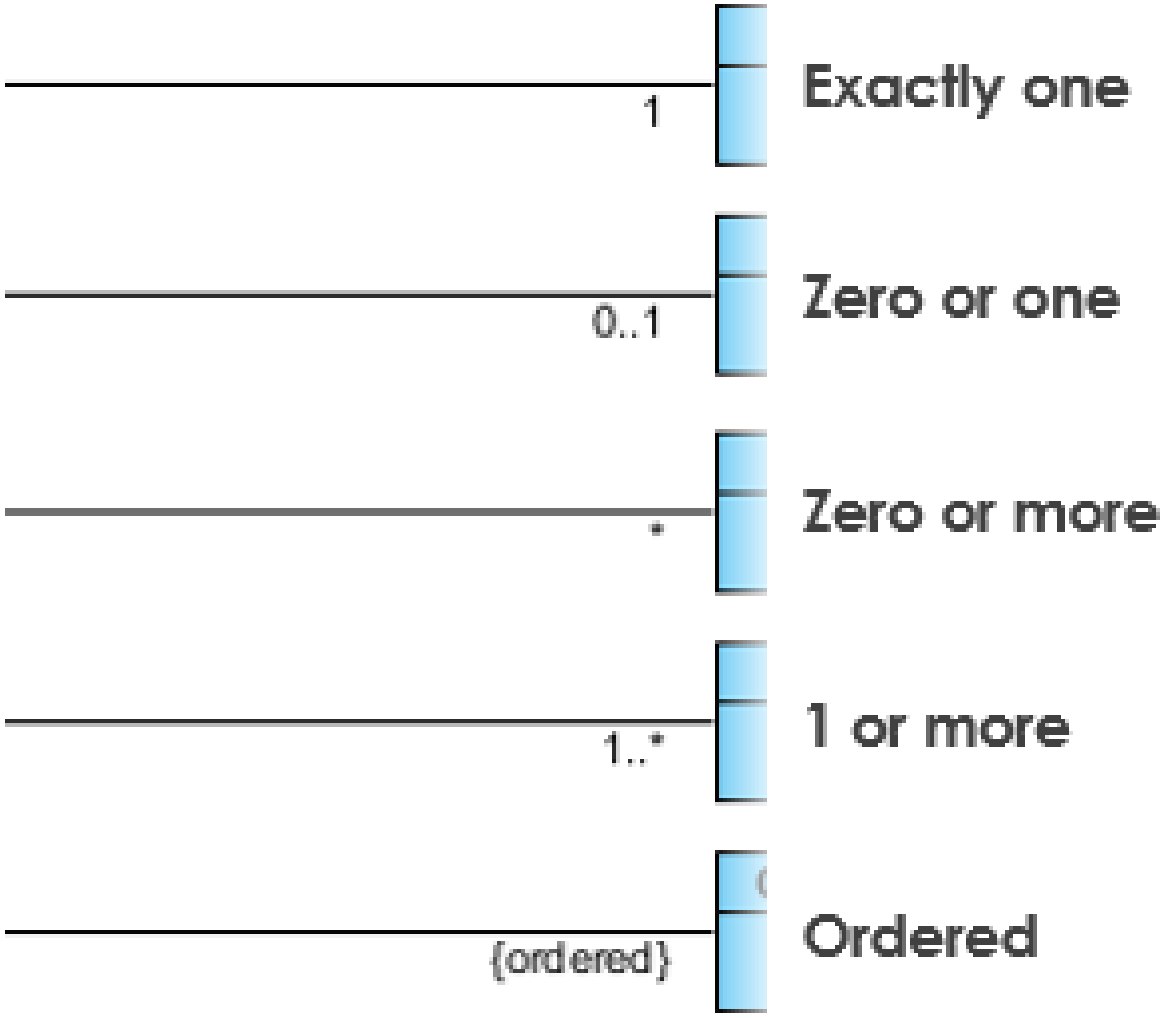
La multiplicité

Multiplicité	Définition
1	Un et un seul
0..1	Zéro ou un
n ou *	n (entier naturel)
m..n	De m à n (entiers naturels)
0..*	De zéro à plusieurs
1..*	D'un à plusieurs

Exercice

- Exemple : *la classe **Cheval** dont la méthode **fairePeur** a été munie d'un paramètre, à savoir **l'intensité** avec laquelle le cavalier fait peur, et d'un retour qui est l'intensité de la peur que le cheval ressent. Ces deux valeurs sont des entiers. Quant aux autres méthodes, elles ne prennent pas de paramètres et ne renvoient pas de résultat.*

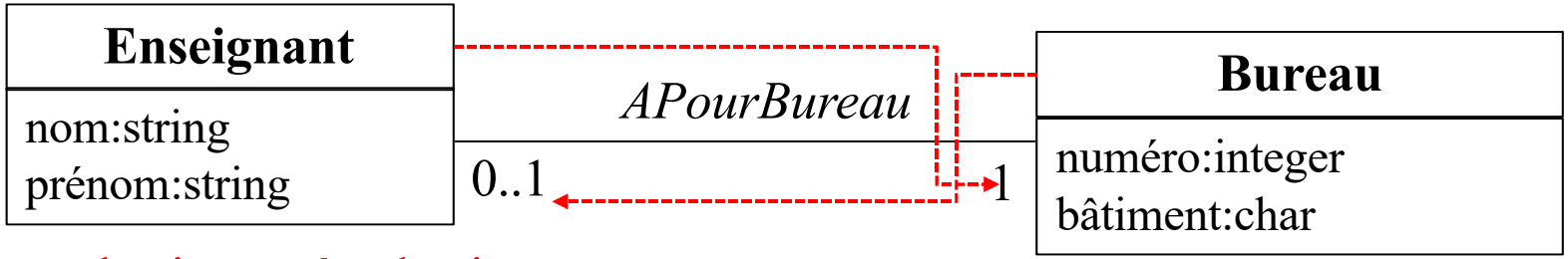



1
Exactly one
0..1
Zero or one
*
Zero or more
1..*
1 or more
{ordered}
Ordered

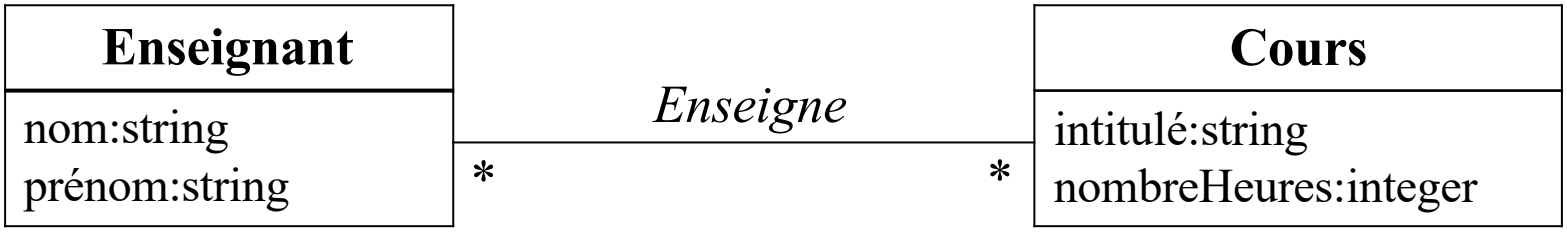
Nombre d'instances d'une classe pouvant être liées à une instance d'une autre classe / contrainte sur le nombre d'objets liés

- « un-à-un »

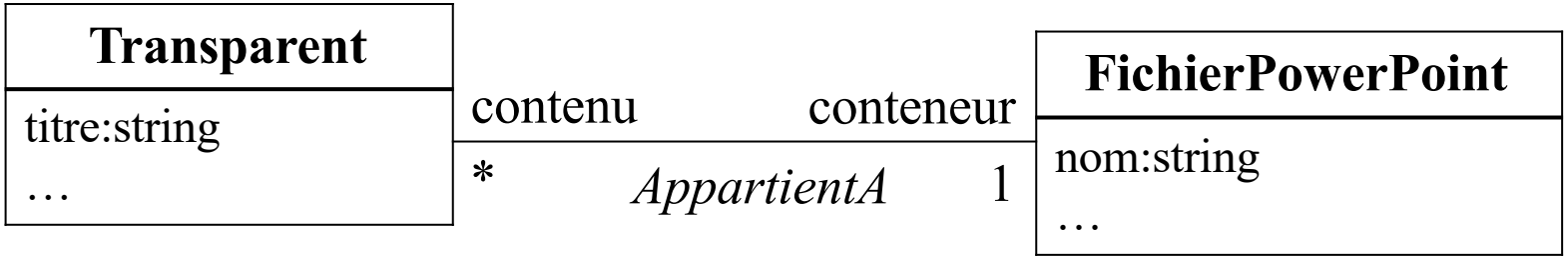
- « zéro-à-un »



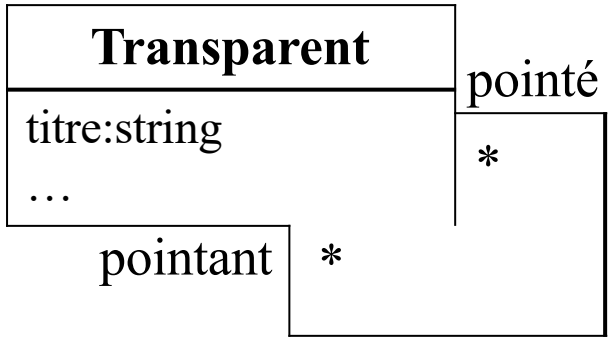
- « plusieurs-à-plusieurs »



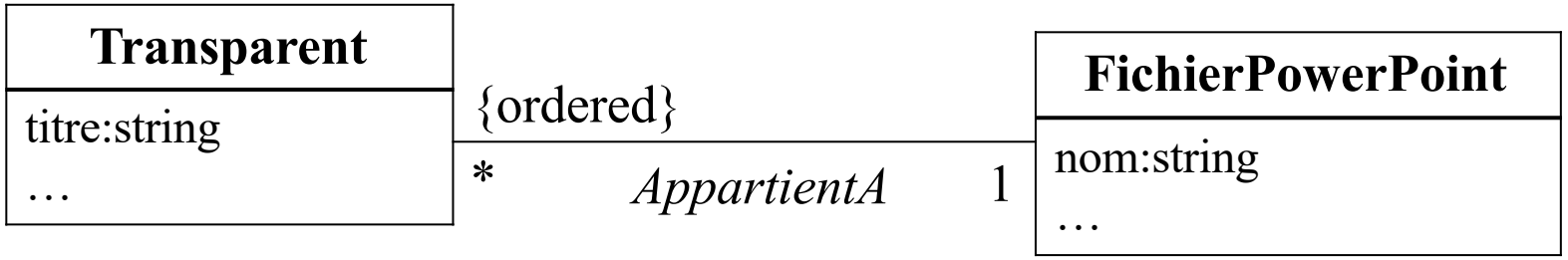
- **Possibilité de nommer les extrémités d'association**



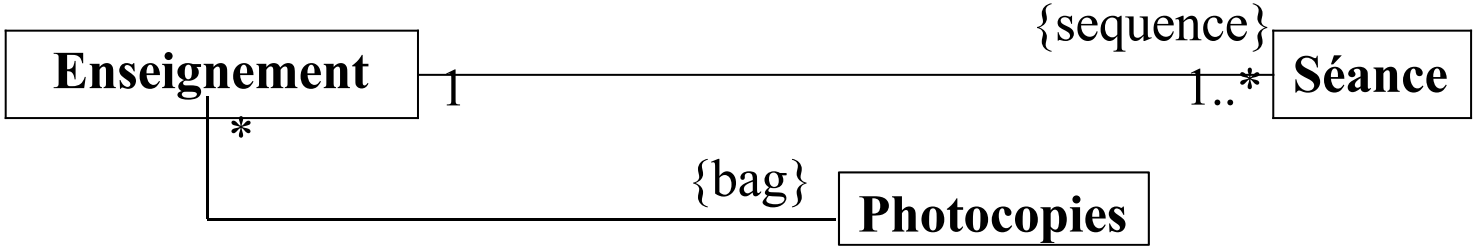
- **Indispensable pour les associations entre objets de même classe**



- **Ordonnancement des objets situés à l'extrémité d'une association « plusieurs »**

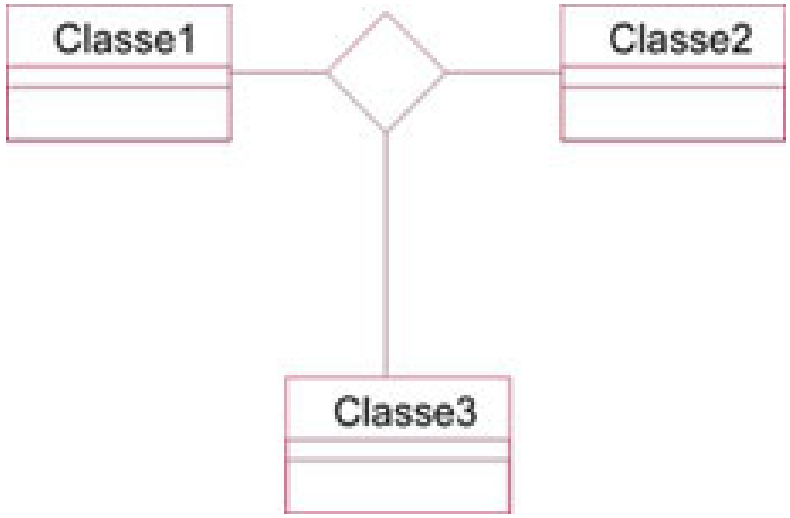


- ***Bag*** **(sac) : collection non ordonnée avec autorisation de doublons**
- **Séquence : collection ordonnée avec autorisation de doublons**

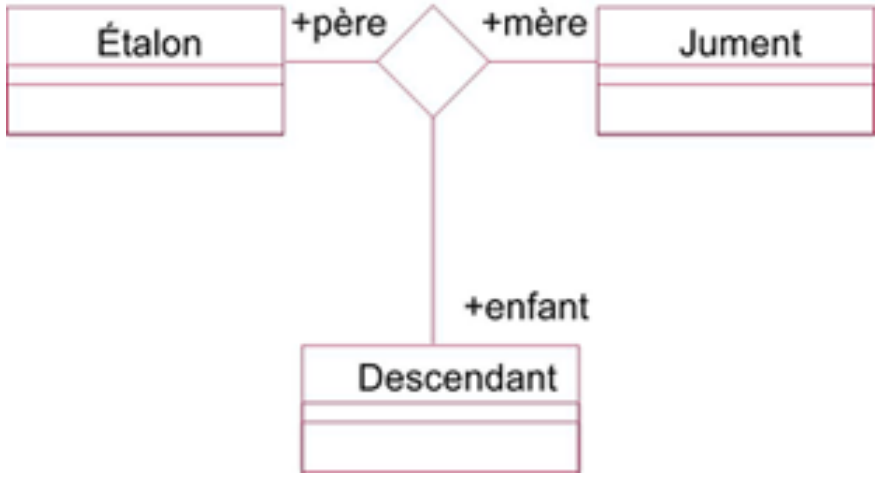


Association ternaire

- Représenté par un losange



- Exemple : association famille



Les classes-associations

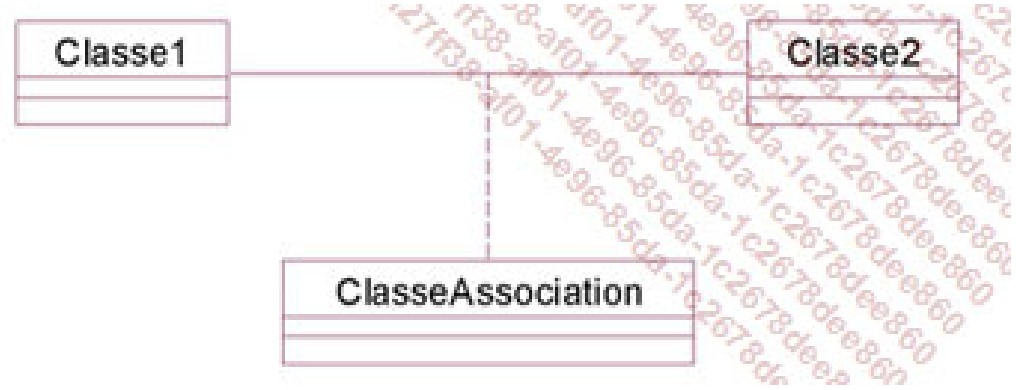
- Les liens entre les instances peuvent porter des infos
- Elles sont spécifiques au lien

→L'association reçoit alors le statut de classe

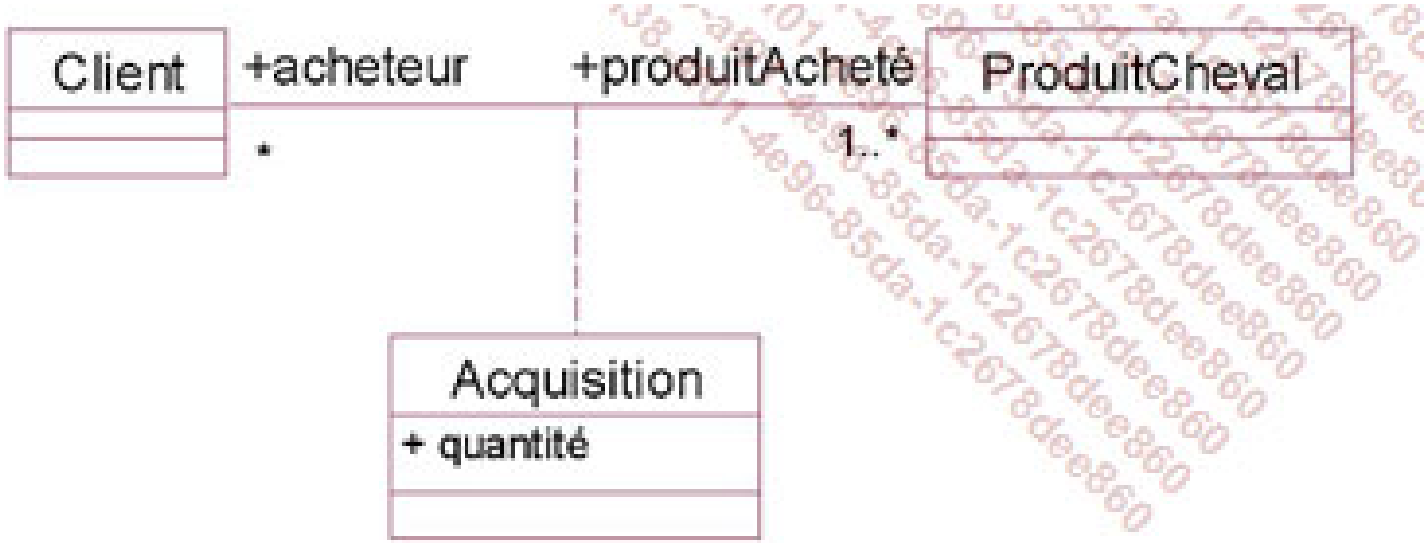
→Ces instances sont donc des occurrences de l'association

→Peut avoir des attributs, opérations et des associations

- On représente avec un trait pointillé

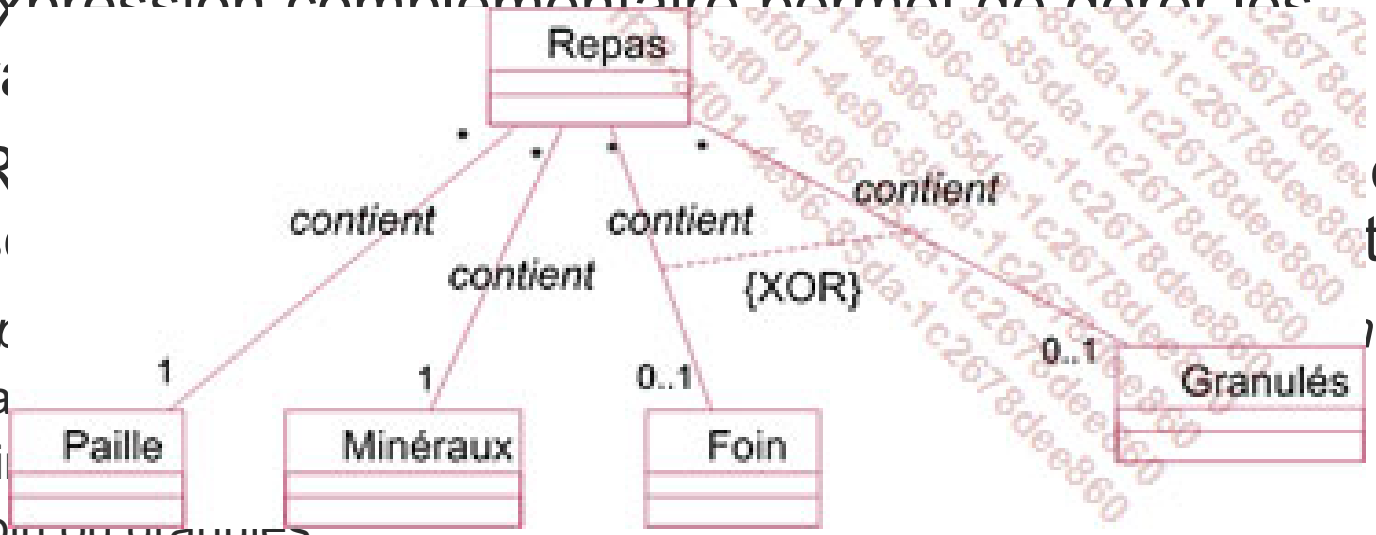


- Exemple : Un client achète un produit pour cheval (produits d'entretien, etc.), il convient de spécifier la quantité de produits acquis par une classe association, ici la classe Acquisition



Contraintes sur les associations

- Un ex… complémentaire permet de gérer les contra…
- {XOR… e la class… tions
- *Exemp… ts :*
  - Pa…
  - Mi…
  - Foin ou granulés



La figure illustre le repas et ses différents constituants. Qu'il contienne soit du foin, soit des granulés est une contrainte exprimée par l'opérateur XOR. Celui-ci exprime la contrainte du ou exclusif entre les deux associations.

Est-ce suffisant ?

- Et bien non …

- Uml propose de s'appuyer sur OCL (ou langage naturel)
  - Langage sous forme de contrainte logique
  - On ajoute des notes aux diagrammes
  - Contraintes OCL toujours vrai ou fausse
  - Utilisation des opérateurs applicables sur des attributs par apport à leur type
  - On peut utiliser les méthodes des objets dans les conditions
  - Pour les ensembles OCL propose : **union, intersection, difference**
  - Pour les collections d'objets : **collect, includes, includesAll, asSet, exists, forAll**