UML

UE15
Informatique appliquée

Frédéric Pluquet
pluquetf@helha.be

v1

2018-2019

Unified Modeling Language (UML)

- Ce **langage** unifie les notations et les concepts orientés objets
- Depuis 2005, l'Object Management Group a adopté la norme UML 2.0
- Actuellement, on est à la version 2.5 (mars 2015)

Il permet de construire les **modèles** nécessaires à la réalisation et la documentation d'un système informatique.

Qu’est-ce qu’un modèle ?

- Une représentation de la réalité
  - Simplifiée
    - Ne schématise qu’une partie de la réalité
  - Conventionnelle
    - Respecte des conventions pour communiquer simplement
  - Pertinente
    - Inclut un maximum d’informations nécessaires

Unified Modeling Language (UML)

- UML 2.x : 14 diagrammes classés en 3 catégories
  - Diagrammes de structure (statiques)
  - Diagrammes de comportement (dynamiques)
  - Diagrammes d’interaction (dynamiques)

Au programme de l’UE15

- Diagrammes de structure (statiques)
  - Diagramme d’objets
  - Diagramme de classes
  - Diagramme de packages
- Diagramme de comportement
  - Diagramme des cas d’utilisation

Où en sommes-nous ?

1. Diagramme de objets
2. Diagramme de classes
3. Diagramme de packages
4. Diagramme de cas d’utilisation

Objet

- Objet = élément du monde réel à propos duquel on souhaite conserver des informations. Ces informations seront soumises à des traitements.
- Les objets peuvent être abstraits ou concrets (par exemple, un compte en banque et une voiture).
- Ils ont
  - un état
  - un comportement

Caractéristiques d’un objet

- Etat
  - Caractéristiques de l’objet à un moment donné, représentées par des valeurs d’attributs.
  - Exemples : taille d’une personne, nombre de kilomètres parcourus par une voiture…

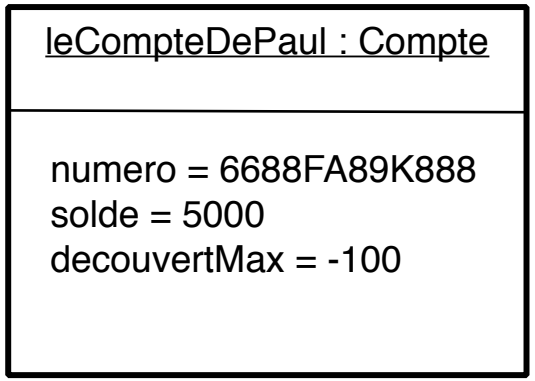
Caractéristiques d’un objet

- Comportement
  - Ensemble des opérations (méthodes) qu’un objet peut exécuter ou subir.
  - Exemples :
    - Être créé (Create)
    - Transmettre ses caractéristiques (Read)
    - Être modifié (Update)
    - Être effacé (Delete)

Diagramme d’objets en UML - un objet

leCompteDePaul

: Compte

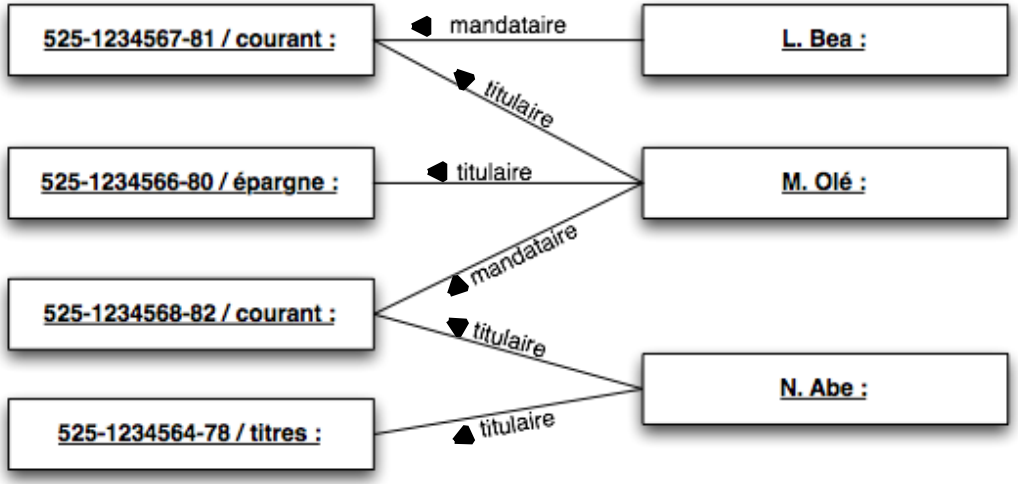


Nom explicite souligné : [Nom de la classe]

Noms et valeurs des caractéristiques à un moment donné

leCompteDePaul : Compte

Diagramme d’objets en UML - liens entre objets



Les objets sont liés entre eux par des lignes pleines dans un diagramme.

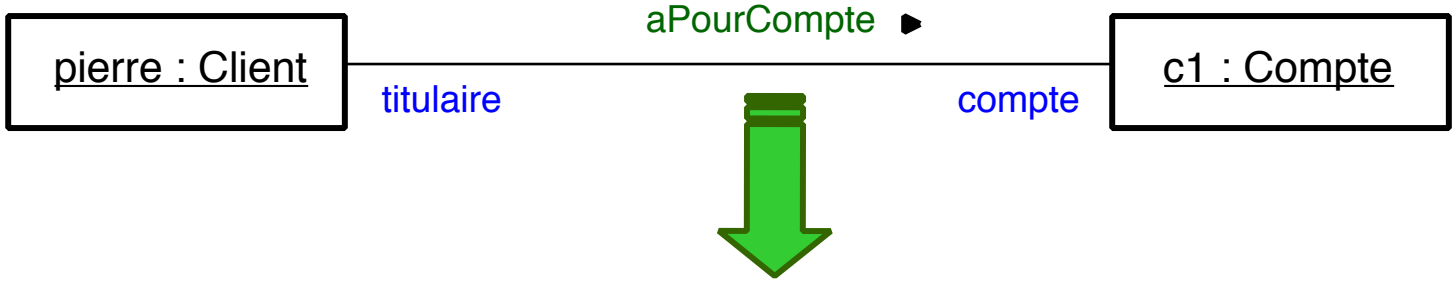
Les noms des liens sont des **formes verbales ou nominales** et commencent par une minuscule.

▶ indiquent le sens de la lecture (ex: « paul aPourCompte c1 »)

Ex: ici les objets du domaine *Clients et Comptes* d’une banque.

Rôles sur les liens

Chacun des deux objets joue un **rôle** différent dans le lien



(1) pierre a pour compte c1
(2) c1 **joue le role de** compte **pour** pierre
(3) pierre **joue le role de** titulaire **pour** c1

(2.b) **le [un des]** compte[s] **de** pierre **est** c1
(3.b) **le [un des]** titulaire[s] **de** c1 **est** pierre

Note de style :

- choisir un groupe nominal pour désigner un rôle
- si un nom de rôle est omis, le nom de la classe fait office de nom

Où en sommes-nous ?

1. Diagramme de objets
2. Diagramme de classes
3. Diagramme de packages
4. Diagramme de cas d’utilisation

De l’objet à la classe

- Impossible de représenter tous les objets qui peuvent être manipulés. D’où la notion de classe.
- Une classe représente des objets de même structure et de même comportement.
- Un objet est une occurrence ou instance d’une classe.

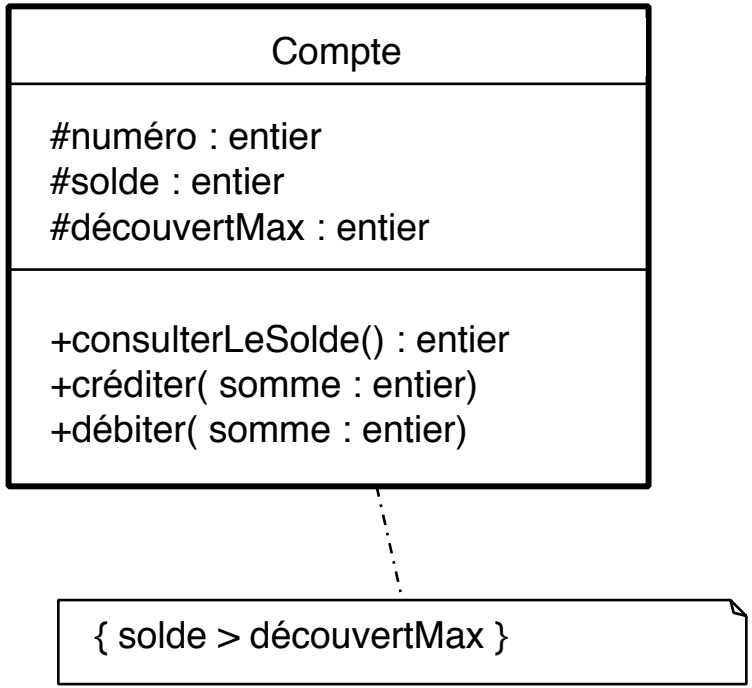
Caractéristiques d’une classe

- Déclaration de l’état
  - Ensemble des attributs communs à toutes les instances de la classe.
  - Exemple : tous les comptes en banque ont un numéro, un solde…
    ...mais pas la même valeur !
  - Pour chaque attribut, on donne un nom et un type

Caractéristiques d’une classe

- Définition du comportement
    - Ensemble des méthodes communes à toutes les instances de la classe.

Diagramme de classes en UML - une classe



Nom de la classe (non souligné)

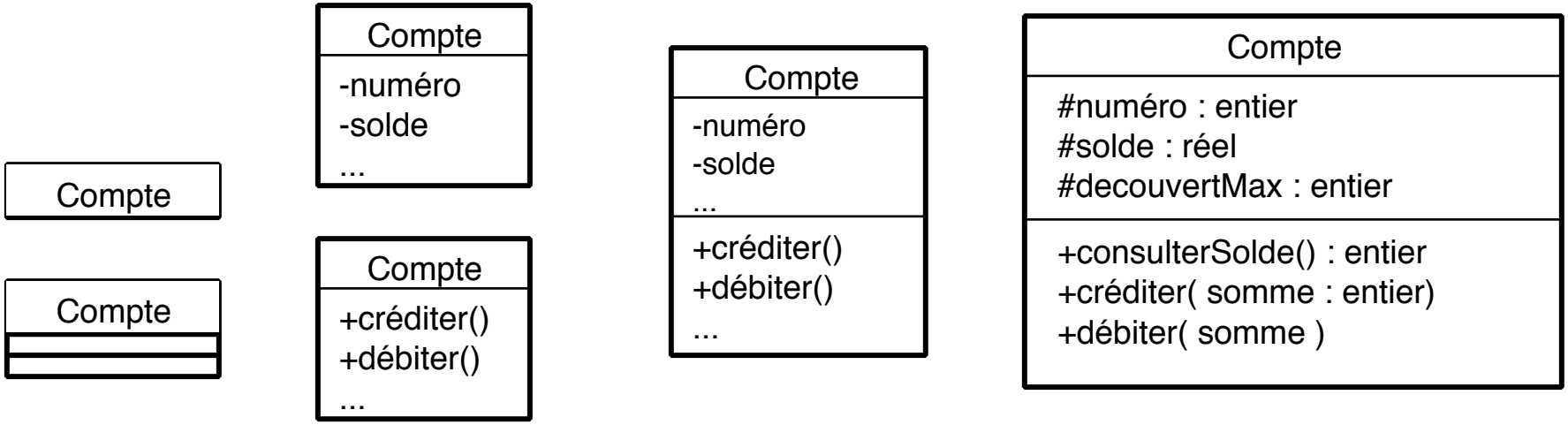
Attributs
- visibilité
- nom
- type

+ : public
# : protégé
- : privé
~ : package

Opérations
- visibilité
- nom
- paramètre
- type du résultat

Contraintes

Diagramme de classes en UML - notations



Notes de style :

- les noms de classes commencent par une majuscule
- les noms d'attributs et de méthodes commencent par une minuscule

Caractéristiques d’une classe

Une **classe** spécifie la structure et le comportement d'un ensemble d'objets de même nature.

La structure d'une classe est constante

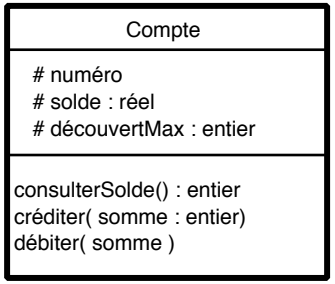


Diagramme de classes

Des **objets** peuvent être ajoutés ou détruits pendant l ’exécution

La valeur des attributs des objets peut changer

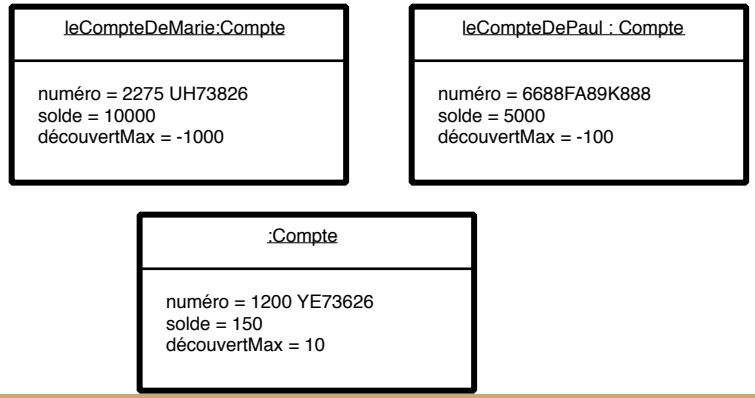


Diagramme d ’objets

Classes «types de données »

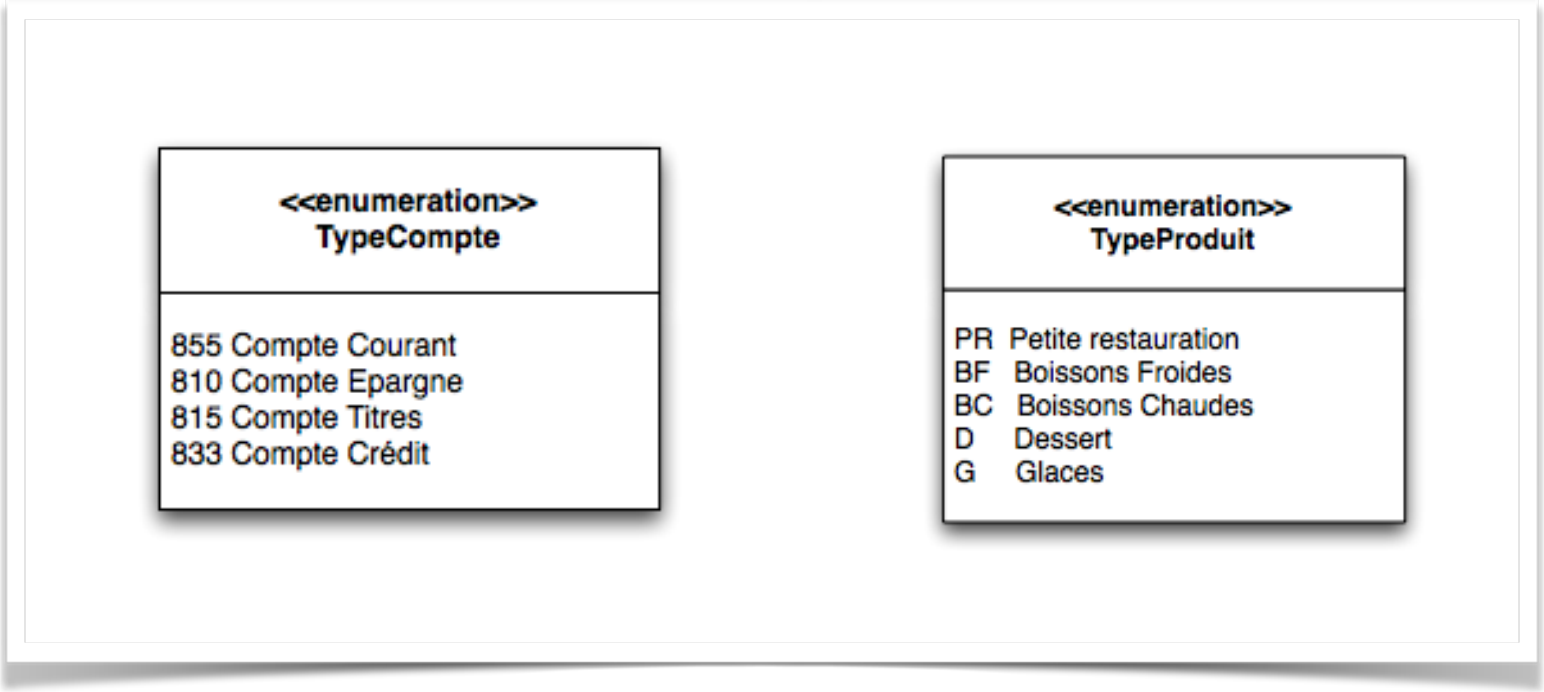
UML propose les types « standard » de données pour les attributs: entiers, booléens, date, ...

- On peut créer des données de type particulier qui seront reprises dans des classes.
- Notation: au-dessus du nom de la classe « particulière », on place le type de données entre << >> (on appelle cela un stéréotype)

Classes «types de données »

- Classe <<énumération>>
    - deux attributs: un code et un libellé (nom).
    - listes connues des utilisateurs.
    - un objet = une valeur de la liste.
- Classe <<structure>>
    - modélisation d’un attribut décomposable (ou composé)
- Ces classes ne peuvent pas avoir d’associations

Classes <<enumeration>>: exemples



Classes <<structure>>: exemples

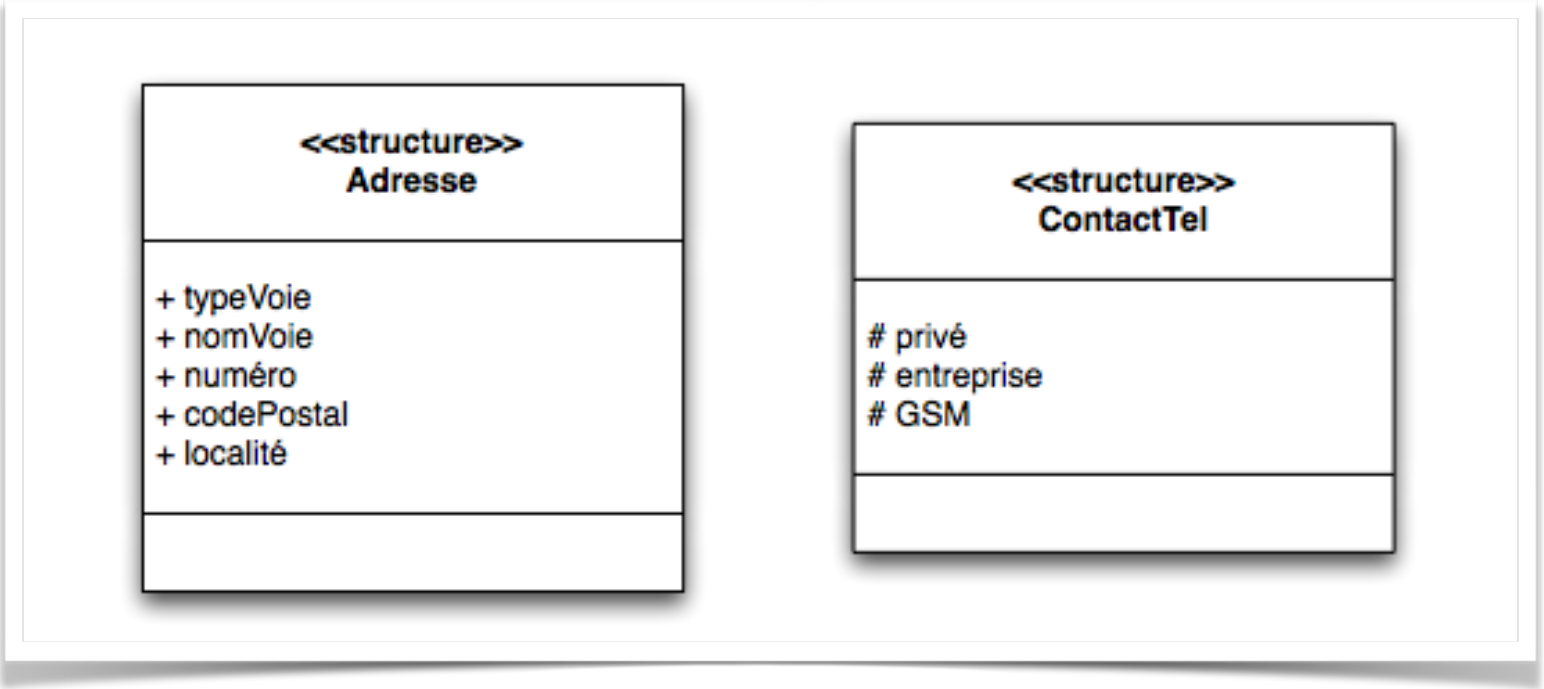


Diagramme de classes en UML - Associations

Une association peut être définie

- soit par un verbe (accompagné d'une flèche qui indique le sens de lecture)
- soit par le rôle d'au moins une classe dans l'association.

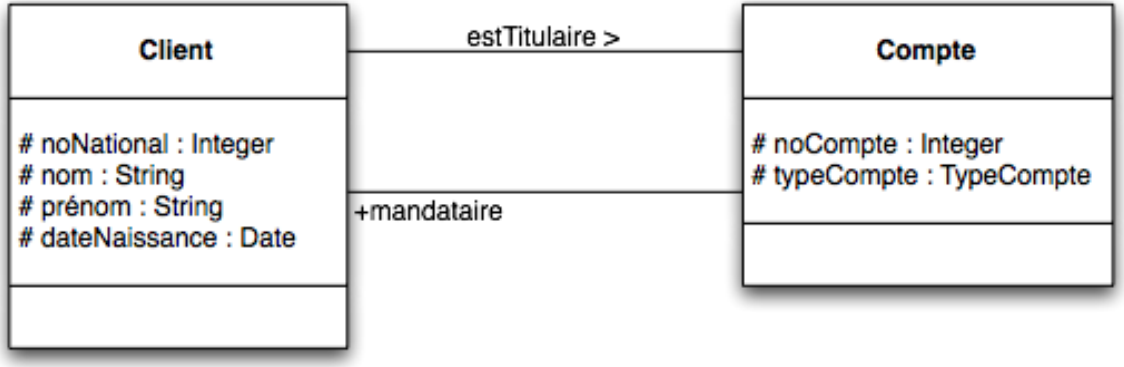


Diagramme de classes en UML - Associations

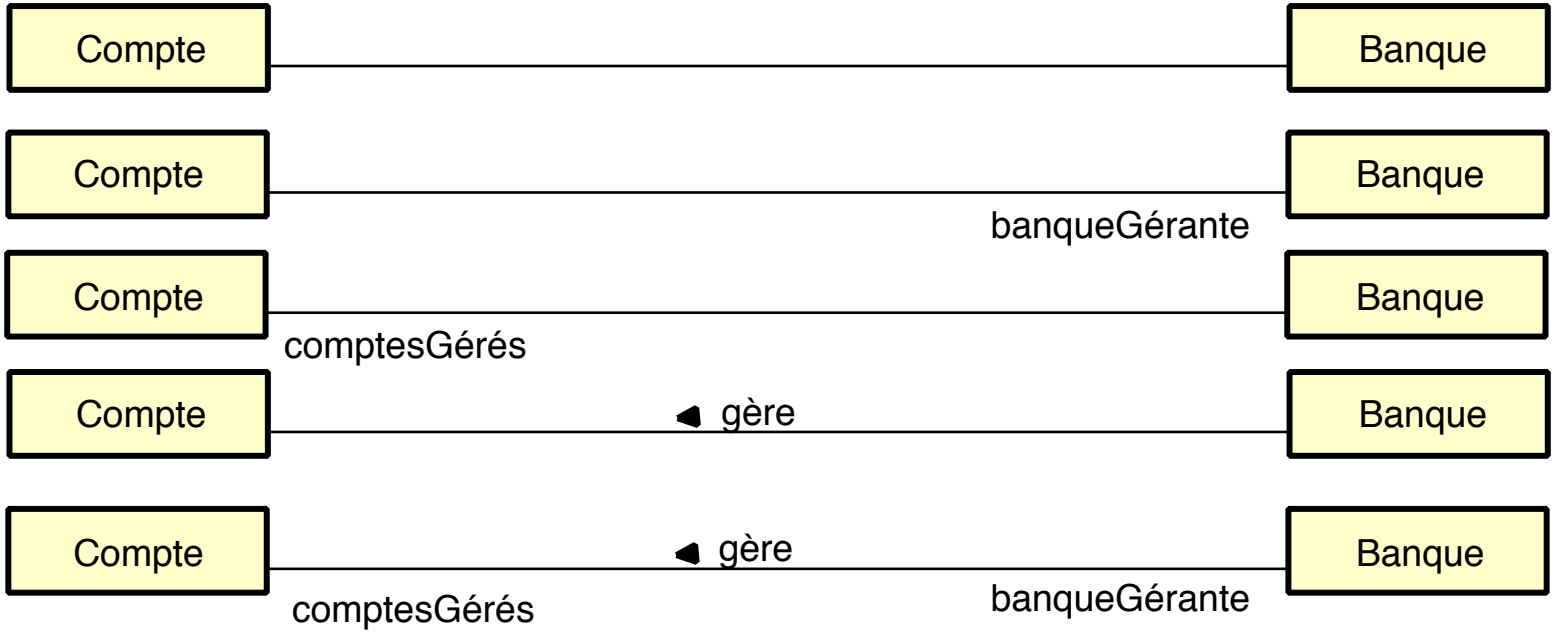
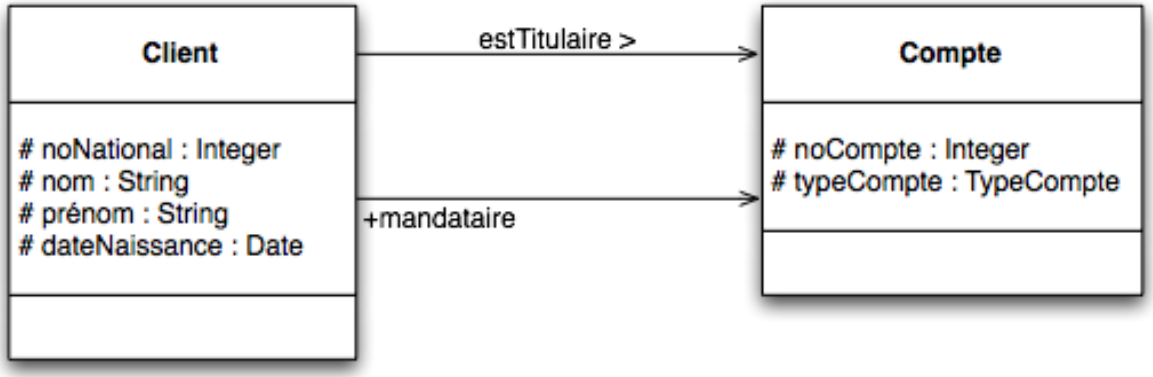


Diagramme de classes en UML - Associations

Attention: le sens de lecture ne se met pas sur la flèche.

La signification d'une association fléchée (unidirectionnelle) est différente de celle d'une association bidirectionnelle: elle indique que l'accès est unidirectionnel.

Par exemple ici, un client peut accéder à ses comptes mais à partir d'un compte, on ne peut savoir qui sont les clients titulaire ou mandataires.

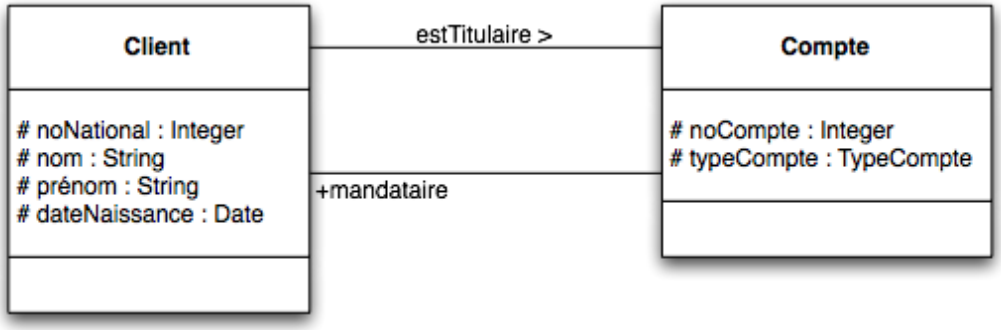


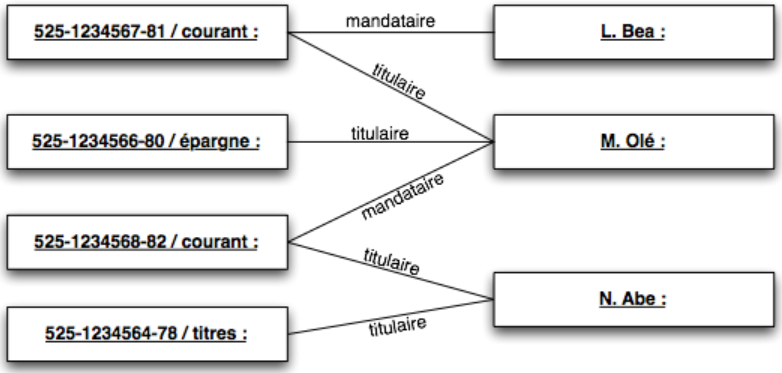
Différences entre les diagrammes de classes et d'objets

Diagramme d'objets	Diagramme de classes
Instances de classes	*Classes*
Liens entre les instances	*Associations* entre les classes

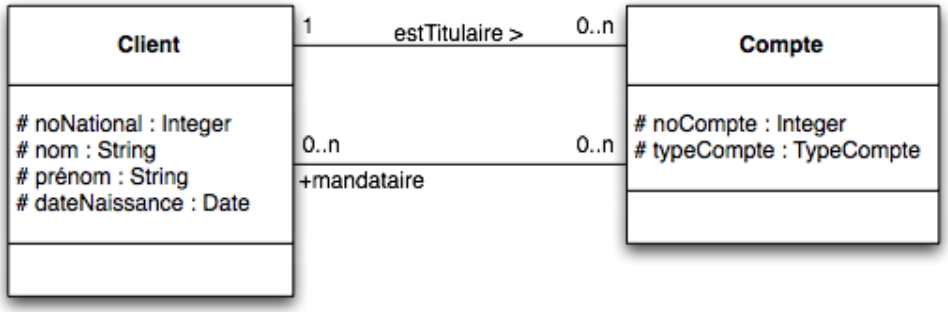
Un objet est une instance d'une classe
et un lien est une instance d'une association.

Relations entre les diagrammes de classes et d'objets





Multiplicités



1. Un compte a un et un seul titulaire.

1. Un client peut n'être titulaire d'aucun compte (0), de 1 ou de plusieurs (n ou *) comptes.

1. Un compte peut n'avoir aucun (0) mandataire ou en avoir 1 ou plusieurs (n ou *).

1. Un client peut n'être mandataire d'aucun compte (0), de 1 ou de plusieurs (n) comptes.

Multiplicités

- Permettent de représenter toutes les possibilités de nombres de liens d'un objet d'une classe avec un objet de la même classe ou d'une autre classe.
    - Pour un objet d'une classe, à combien d'objets sera-t-il lié au minimum et au maximum ?
        - Attention: on ne lit qu'une seule multiplicité à la fois !

- Syntaxe : {Minimum} .. {Maximum}

Multiplicités

- Les plus courantes:
  - 0..1
  - 1..1 ou 1
  - 0..n ou n ou 0..* ou *
  - 1..n ou 1..*

On peut avoir aussi:

3..5

0..18

15..*

Remarques

* équivalent à 0..*

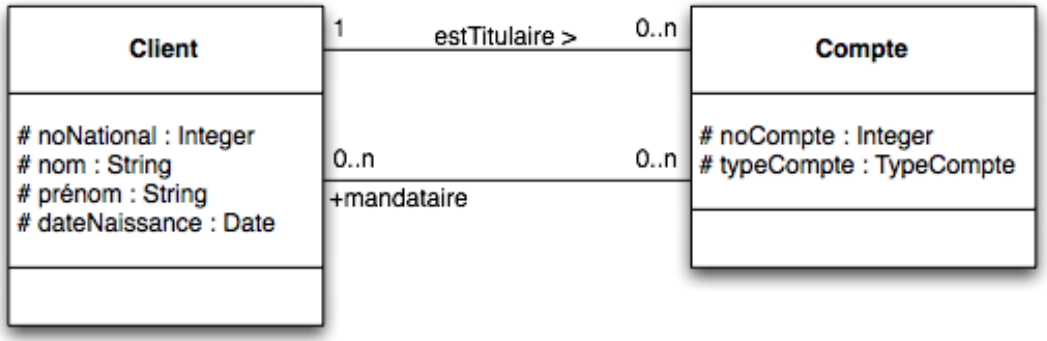
3 équivalent à 3..3

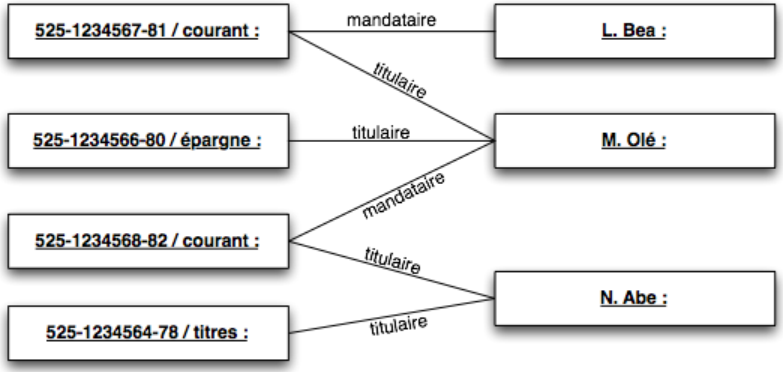
Multiplicités

Attention:

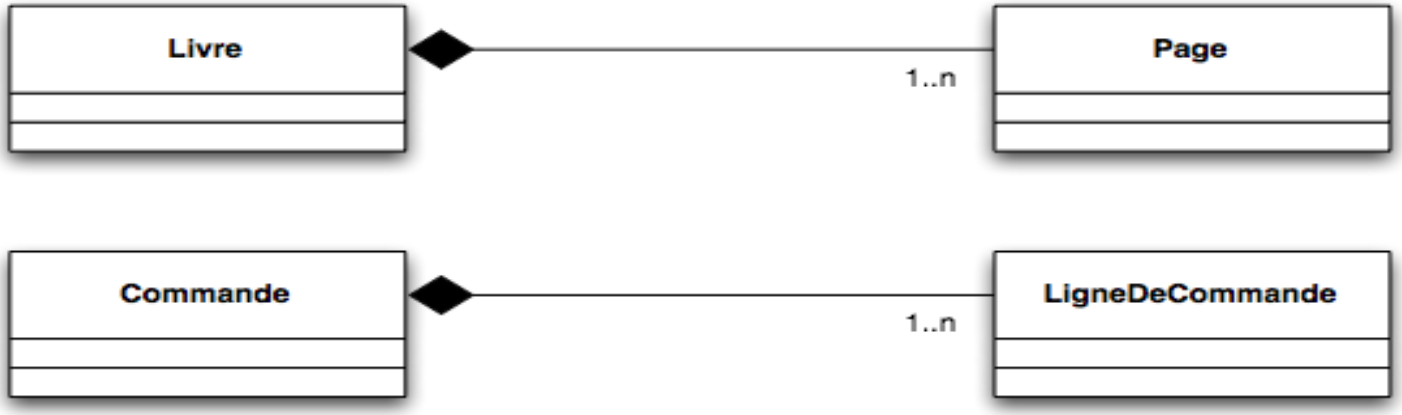
- Ce sont des choix de gestion.
- Ce sont des contraintes d'intégrité.

Relations entre les multiplicités du diagramme de classes et du diagramme d'objets





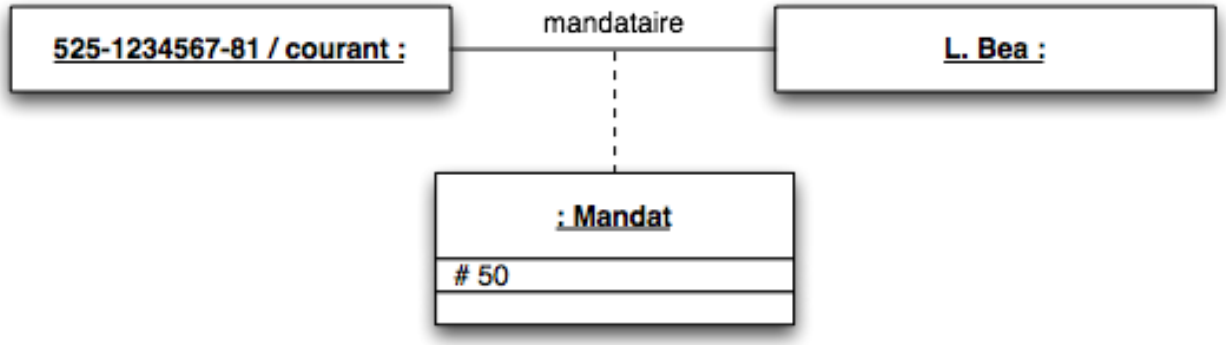
Composition



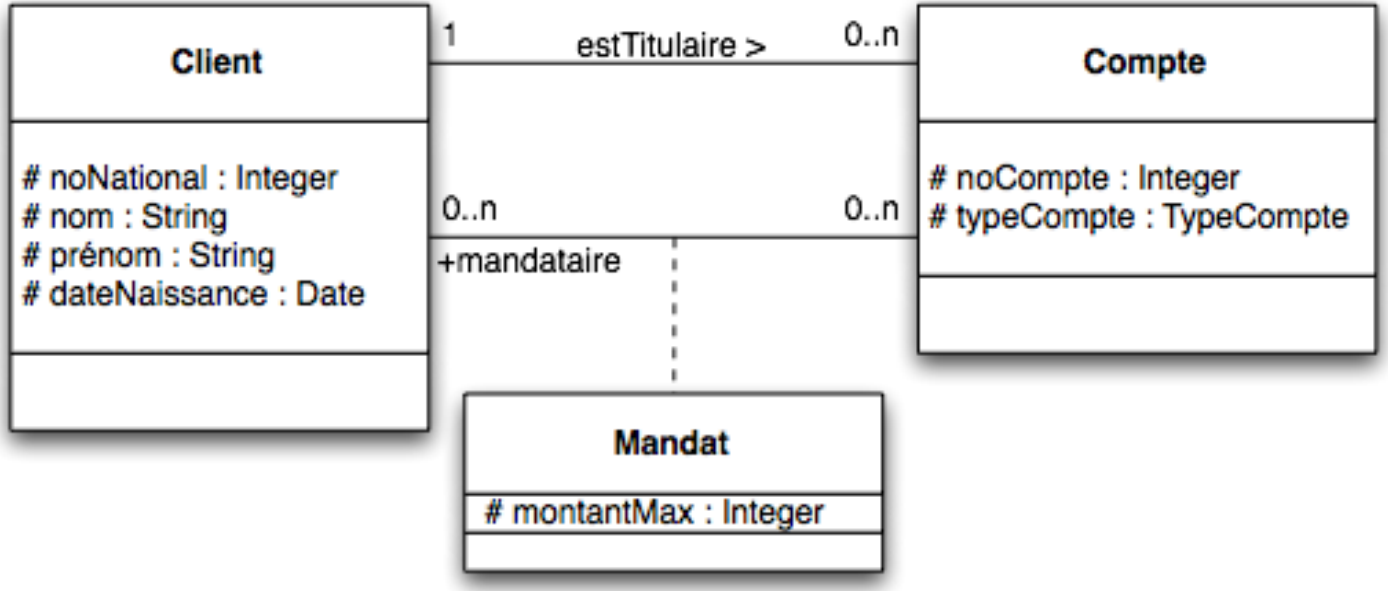
- Ajoute une sémantique : “est composé de” / “fait partie de”
- Contraintes liées à la composition :
  - Un objet *composant* ne peut être que dans 1 seul objet *composite*
  - Un objet *composant* n’existe pas sans son objet *composite*
  - Si un objet composite est détruit, ses composants aussi

Classes d’association

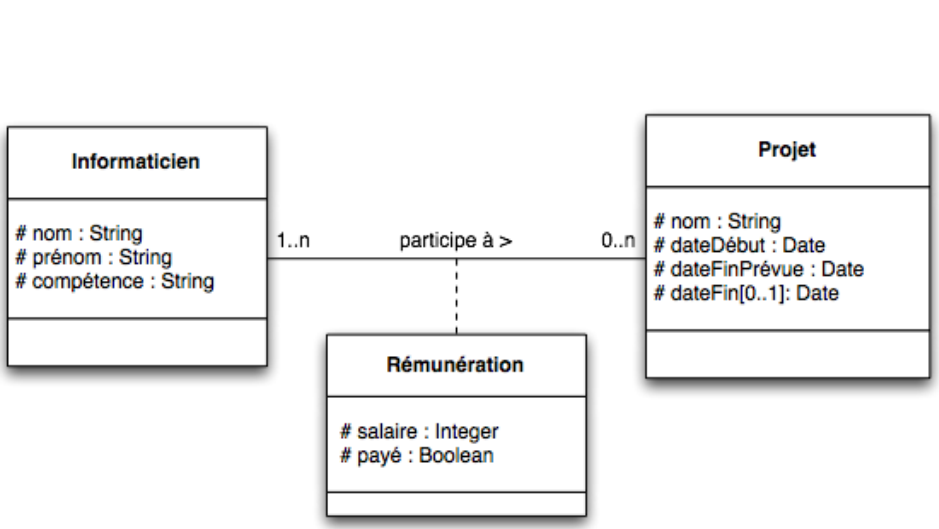
- On constate parfois qu’on veut conserver des données au sujet d’un lien entre objets.
- Ca donnera lieu à la création d’une classe d’association.

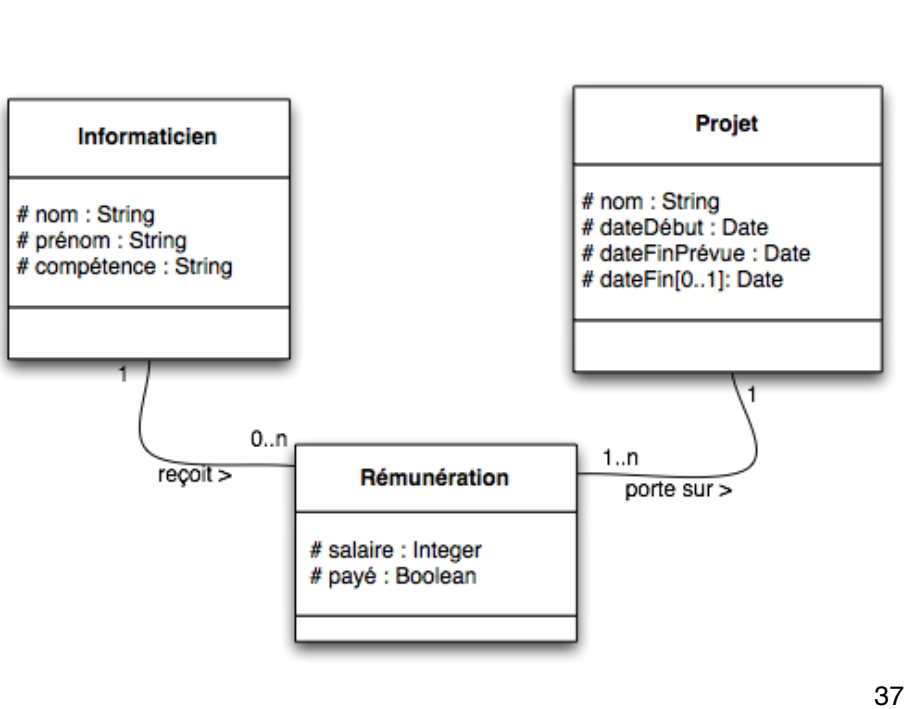


Classes d’association - exemple



Classes d’association - Jeu des différences :)



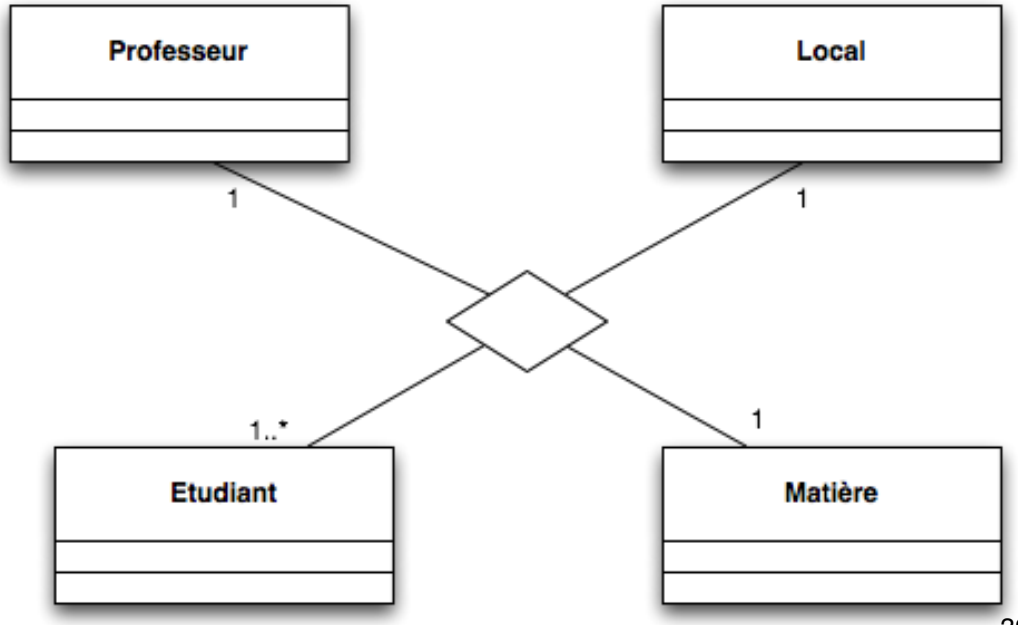


Associations n-aire

- Parfois plus de 2 classes peuvent être liées.
- Une association ternaire, quaternaire, … doit alors être créée.
- On appelle cela des associations n-aires.

Associations n-aire - exemple

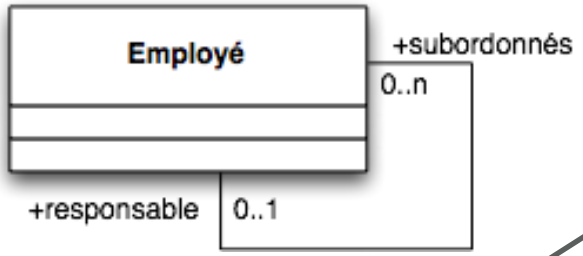
Cette association représente le lien entre un professeur qui donne un cours dans un local à un groupe d'au moins un étudiant.



Association réflexive

- Association qui part et arrive sur une même classe.
- Elle signifie donc qu'il existe des liens entre objets d'une classe.

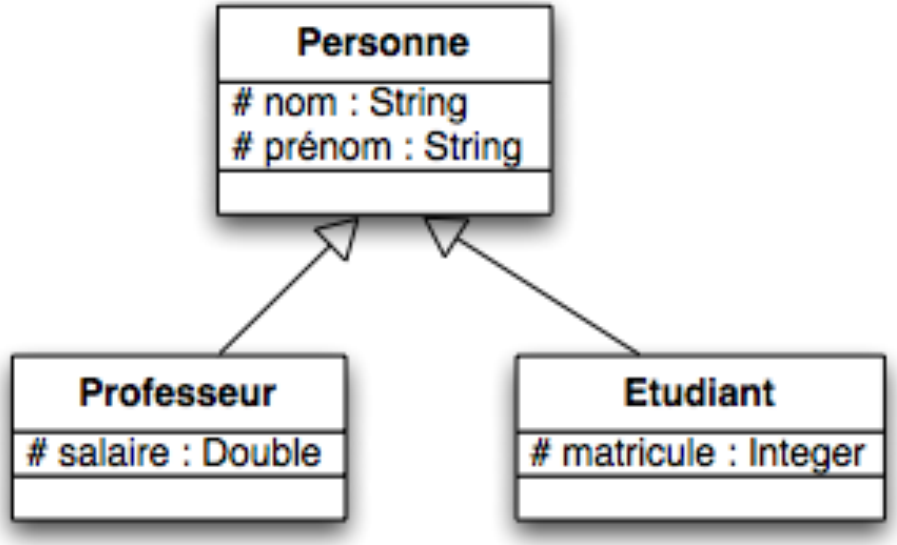
Association réflexive - exemple



Héritage

- L’héritage permet de classifier les classes dans une hiérarchie.
- On peut ainsi définir dans une « super-classe » les attributs et les méthodes communs à plusieurs « sous-classes ». Cela permet donc la réutilisation.
- Toutes les classifications (botaniques, zoologie…) répondent a ce type de hiérarchie.
- On a une hiérarchie de classes quand on peut dire: « est un(e) »
    - Exemple: Un professeur (sous-classe) est une personne (super-classe).

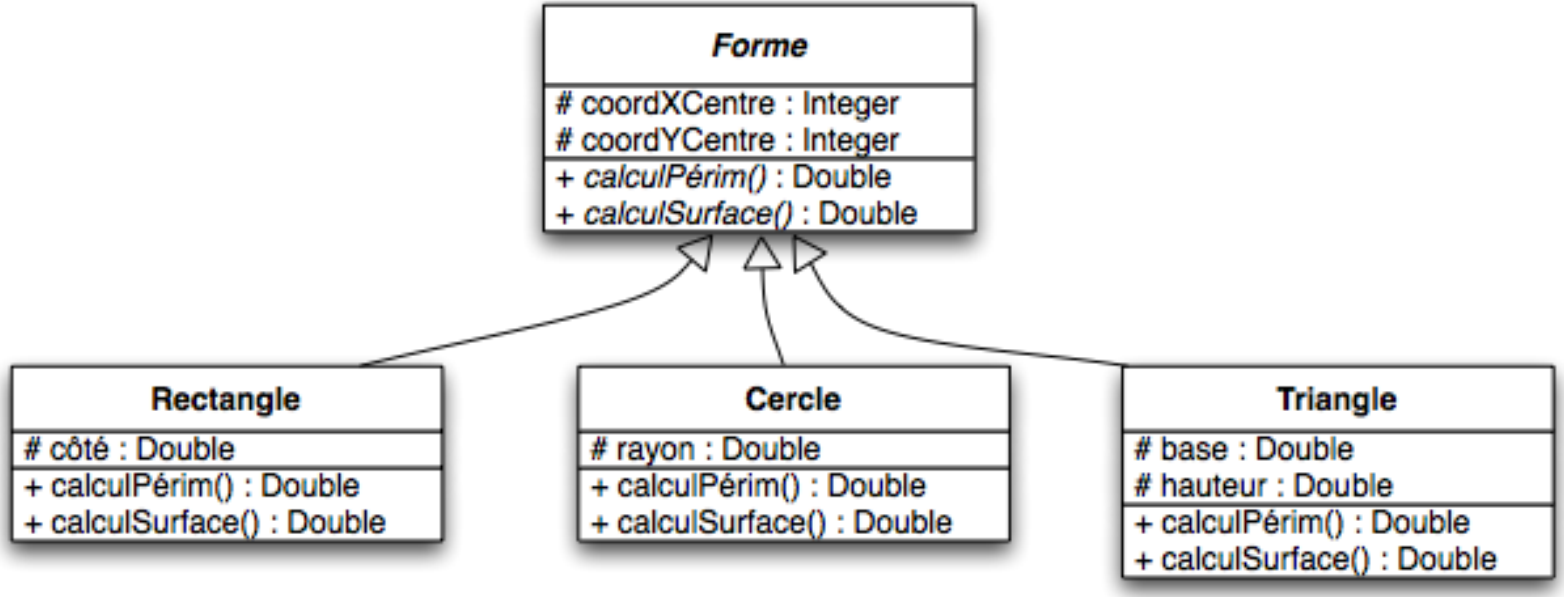
Héritage


Personne
nom : String
prénom : String
Professeur
salaire : Double
Etudiant
matricule : Integer

Héritage et polymorphisme

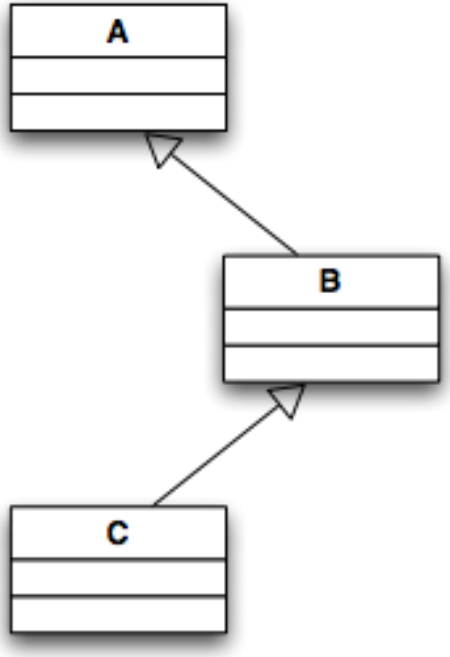
- Certaines opérations décrites au niveau d'une super-classe peuvent être redéfinies au niveau des sous-classes.
- On a donc plusieurs formes (poly-morphe) d'une même entité (ici une méthode).
- Il existe du polymorphisme de méthodes, de variables…

Héritage et polymorphisme



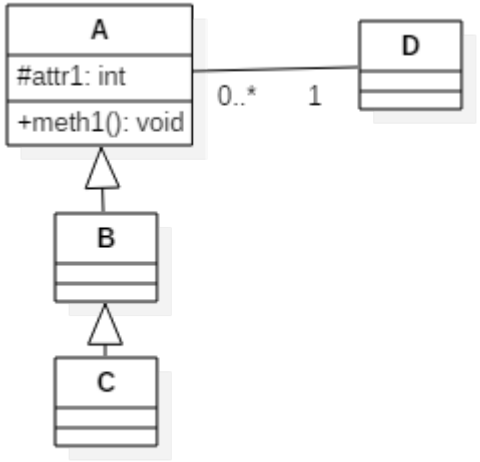
Héritage: propriétés

- Relation transitive
  - Si C hérite de B
  - et que B hérite de A
  - Alors C hérite de A.
- C possède alors toutes les caractéristiques de B et de A.



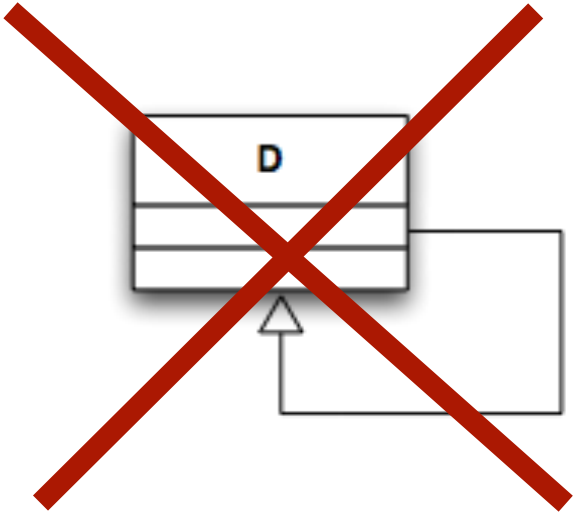
Héritage: propriétés

- Relation transitive
  - pour les attributs
  - pour les méthodes
  - pour les associations
- A, B et C ont:
  - un attribut entier protégé *attr1*
  - une méthode *meth1*
  - une association vers D



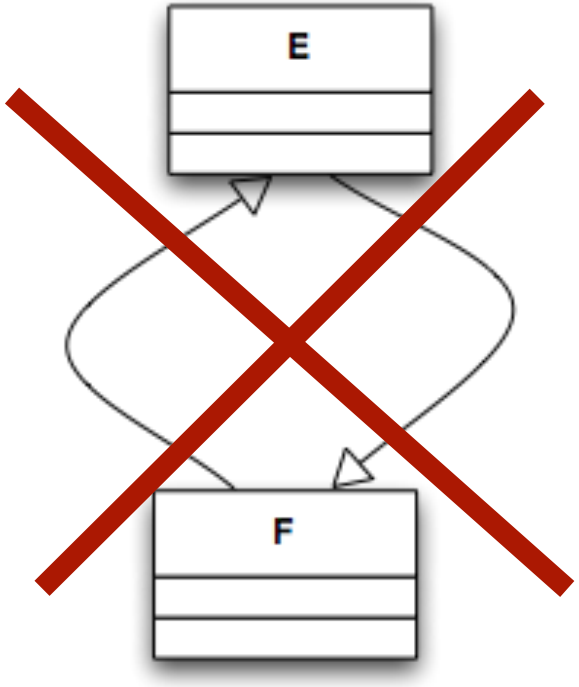
Héritage: propriétés

- Relation non réflexive
  - Une classe ne peut pas hériter d'elle-même.



Héritage: propriétés

- Relation non symétrique
  - Si F hérite de E
  - Alors E ne peut pas hériter de F.
- La hiérarchie ne peut pas former de cycle.



Contraintes d’intégrité

- Définition
  - Règles qui permettent de garantir la cohérence des données lors de leur mise à jour par rapport au monde réel.
- Ces règles sont soit
  - indiquées dans l’UML (technique à utiliser lorsque c’est possible)
  - écrites en français dans une note UML ou dans un texte accompagnant le diagramme

Contraintes d’intégrité

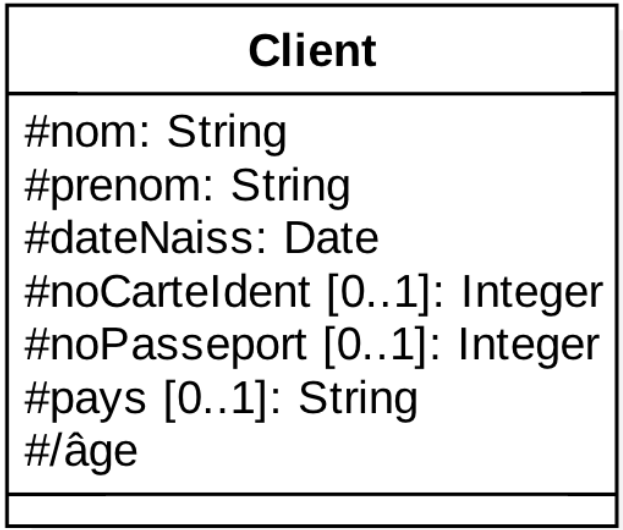
- Sur les attributs
  - Types de données
  - Domaines de valeurs
  - Obligatoires ou facultatifs
  - Attributs liés
- Sur les associations
  - Multiplicités, inclusion et exclusion

Contraintes d’intégrité sur les attributs

- Domaine de valeurs d’un attribut:
  - Types de données: entier, réel, booléen, date (une classe connue), chaîne de caractères, …
  - Limitation des ces valeurs:
    - Exemples :
      - « La date de naissance doit être antérieure à la date du jour. »
      - « Le numéro de client est un entier positif sur 5 positions en DCB. »

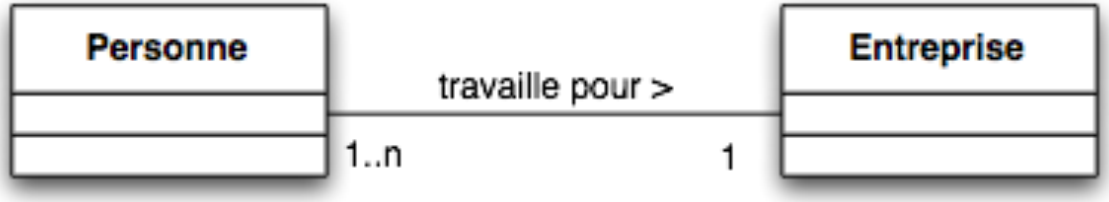
Contraintes d’intégrité sur les attributs

- Existence d’un attribut:
  - Obligatoire
  - Facultatif
    - Notation UML : [0..1]
- Lien entre attributs:
    - « noCarteIdent ou noPasseport »
    - « pays n’est rempli que si on a un numéro de passeport. »

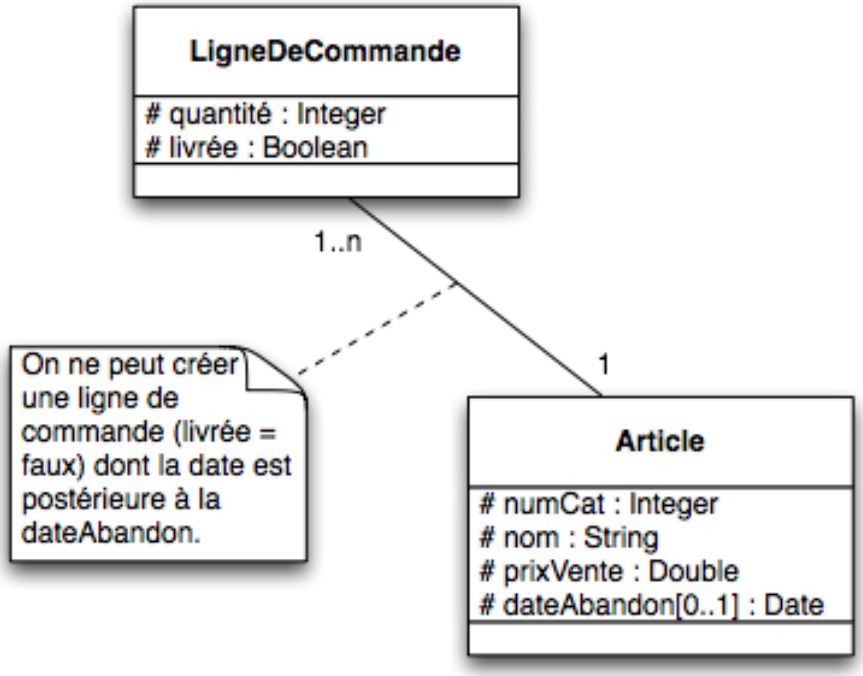


Contraintes d’intégrité sur les associations

- Données la plupart du temps par les multiplicités:
  - Contrainte d’existence: multiplicité minimale = 1



Contraintes d’intégrité sur les associations



- Une ligne de commande ne peut exister que pour un article qui existe à la date de la commande.
    - date de LigneCommande doit être antérieure à dateAbandon.

Contraintes d’intégrité sur les associations

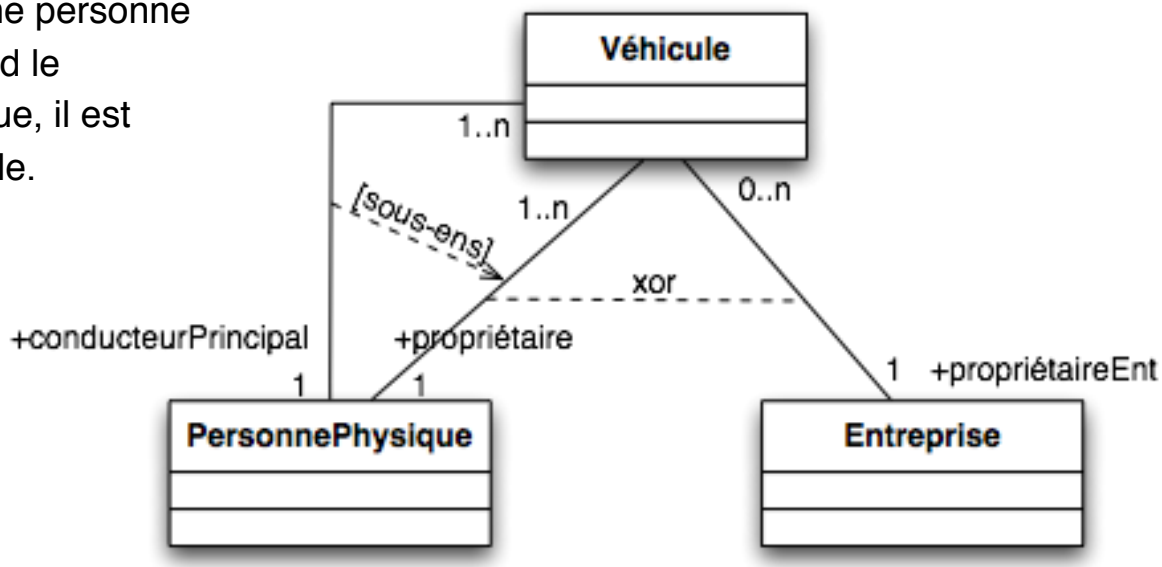
- Contrainte d’inclusion
  - Une association est incluse dans une autre quand un lien qui appartient à l’une appartient aussi à l’autre
    - Ex: La personne directeur d’un département y est aussi employé.

Contraintes d’intégrité sur les associations

- Contrainte d’exclusion
  - Deux associations sont exclusives si, quand un lien appartient à l’une, il ne peut pas appartenir à l’autre.
    - Ex: Un enseignant dans une école ne peut pas y être étudiant.

Contraintes d’intégrité sur les associations

Un véhicule n’a qu’un propriétaire: une personne ou une entreprise (ou exclusif). Quand le propriétaire est une personne physique, il est aussi le chauffeur principal du véhicule.



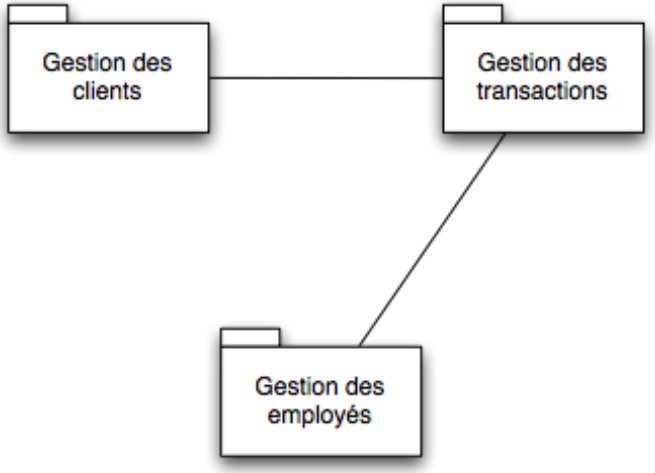
Où en sommes-nous ?

1. Diagramme d’objets
2. Diagramme de classes
3. Diagramme de packages
4. Diagramme de cas d’utilisation

Diagramme de packages

- Si les diagrammes de classes deviennent trop larges (trop de classes), nous pouvons en faire plusieurs.
- Chaque diagramme de classes sera placé dans un package.
- Un diagramme de package permettra d'avoir une vue globale.

Diagramme de packages en UML - Exemple



- Un lien indique une “dépendance” : une modification d’un package peut entraîner une modification dans l’autre package.