



# Génie Logiciel 1

## Diagramme de Classe



Ecole d'ingénieurs et d'architectes de Fribourg  
Hochschule für Technik und Architektur Freiburg

Télécom 2ème année  
Semestre d'automne  
2018-2019

.....  
**Hes-SO** // FRIBOURG  
Haute Ecole Spécialisée  
de Suisse occidentale  
Fachhochschule Westschweiz



## Diagramme de classe



- **Le diagramme de classes est considéré comme le plus important de la modélisation OO.**
  - C'est très certainement le plus utilisé
- **Il montre la structure interne du système**
  - Alors que le UC montre le système du point de vue des acteurs
  - Pas d'acteur présent dans ce diagramme
- **Le diagramme de classe ne montre pas comment utiliser les opérations**
  - C'est une description purement **statique** du système

# Points de vue

- Lorsque vous tracez un diagramme de classe vous pouvez adopter trois points de vue
  - Conceptuel
    - Le diagramme représente des concepts du domaine que vous étudiez
      - Indépendant du langage
  - Spécification
    - Permet de spécifier les interfaces
      - Indépendant de l'implémentation
  - Implémentation
    - Permet de spécifier comment sera implémentée la classe
      - Dépendante du langage et de l'implémentation

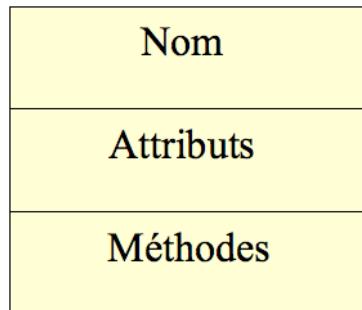
# Qu'est-ce qu'une classe

- Une classe est une description d'un groupe d'objets avec des propriétés communes, un comportement commun, des relations à d'autres objets communes et une sémantique commune.
- Une classe est un modèle pour créer des objets, chaque objet est une instance d'une certaine classe.
  - Un objet est une instance de classe !

# Diagramme de classes (syntaxe)

- Une classe est décrite par:

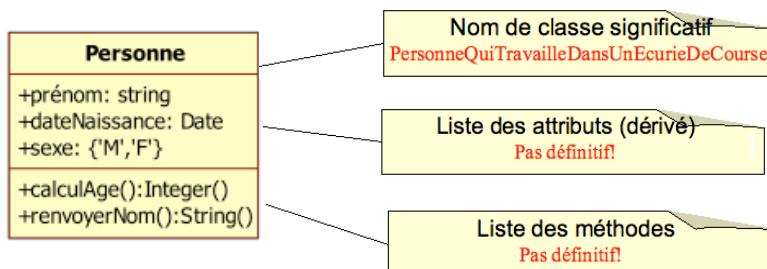
- son nom
- ses attributs (états)
- ses méthodes (actions)



## De l'objet à la classe

- Beaucoup d'objets **naturels** ou **informatiques** se ressemblent!!
  - Niveau description
  - Niveau comportement
- Regrouper objets similaires en une classe → limiter la complexité

Jean est un homme célibataire âgé de 40 ans et Anne est une femme mariée

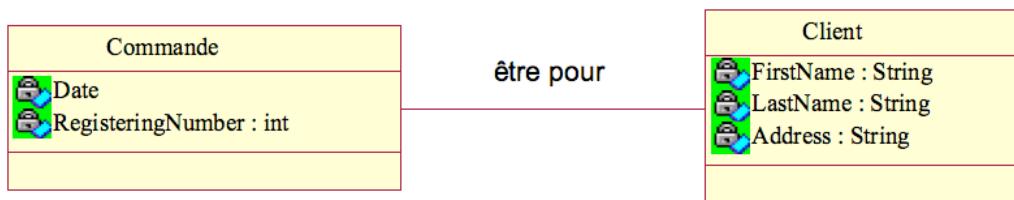


# Trouver la classe et abstraction

- La définition d'une classe contient des détails qui sont importants **selon le point de vue adopté par le système que vous modélisez.**
- Abstraction: mise à l'écart des détails non pertinents dans un contexte donné (point de vue)
- Difficile de trouver le bon niveau d'abstraction
- Exemple : Tenue d'un compte bancaire
  - Point de vue de la banque, du client, ...

# Associations entre « classes »

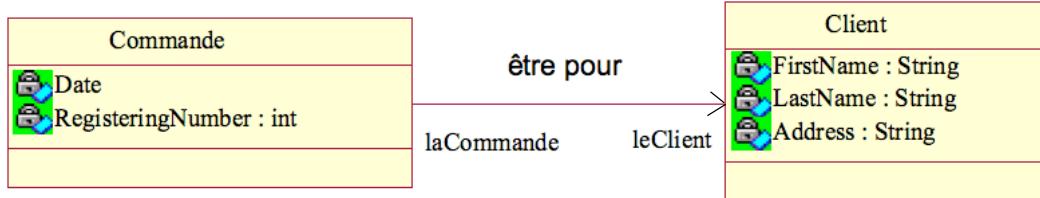
- Une association est une relation entre **instances** de classes (objets)
- Il s'agit d'un élément essentiel d'un diagramme de classes
- Représente une relation **sémantique** entre classes
- On peut lui donner un nom qui représente cette sémantique
  - Une commande «**est pour** » un client



# Caractéristique d'une association

## Navigabilité

- Permet d'indiquer le sens de l'association
- La navigabilité peut être bi-directionnelle



Une commande a la responsabilité du client auquel elle est associée mais pas l'inverse

- Une commande doit pouvoir me dire qui est son client
- Un client ne peut pas me dire (directement) quelles sont ses commandes

```
class Commande {
    Client leClient;
}

class Client { }
```

# Caractéristique d'une association

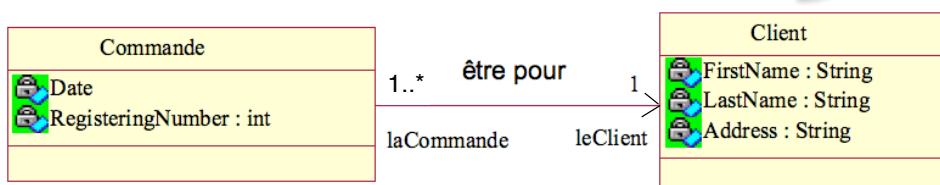
## Rôle

- Chaque extrémité a un rôle
- Il s'agit du nom donné à l'instance de la classe qui sera en relation

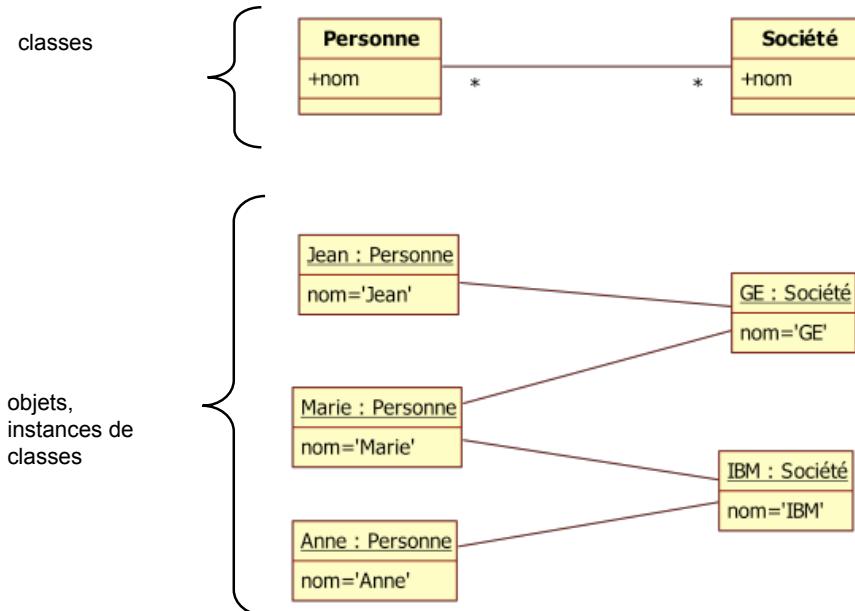
## Multiplicité

- Indique avec combien d'instances de l'autre classe une instance de la classe peut être associées

0..1	Zéro ou un
1	Exactement un
0..*	Zéro ou plus
1..*	Un ou plus
*	Plusieurs
0..n	de 0 à n
n..*	n ou plus
n..m	de n à m



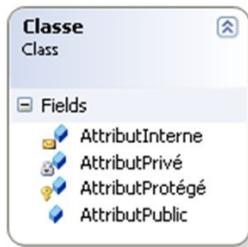
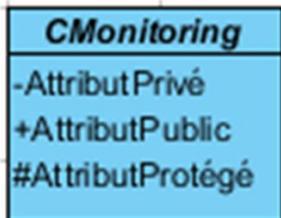
# Classes et Objets



## Exercice 1

- Modélez à l'aide d'un diagramme de classes la situation suivante :
  - Un blog est constitué de messages
  - Le blog appartient à un utilisateur
  - Un utilisateur peut avoir plusieurs blogs
  - Les messages peuvent-être commentés par d'autres utilisateurs
- Pour chaque relation, utiliser toutes notations vues précédemment (rôle, multiplicité et navigabilité)

# Visibilité des attributs et méthodes

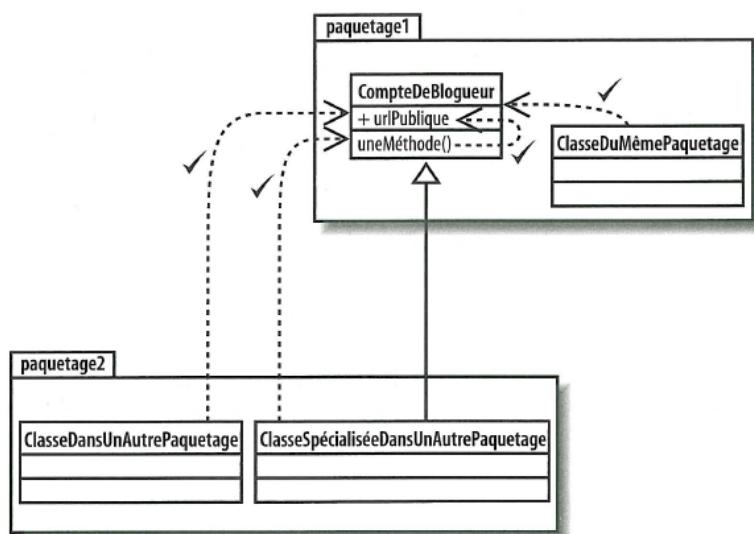


- La syntaxe UML utilise les marques de visibilité suivantes:
  - + pour un attribut **public**
  - pour une attribut **privé**
  - # pour un attribut **protégé**

Notation	Signification
privé	La primitive n'est visible que dans la classe
public	La primitive est visible par toutes les autres classes
protégé	La primitive est visible par la classe et ses sous classes

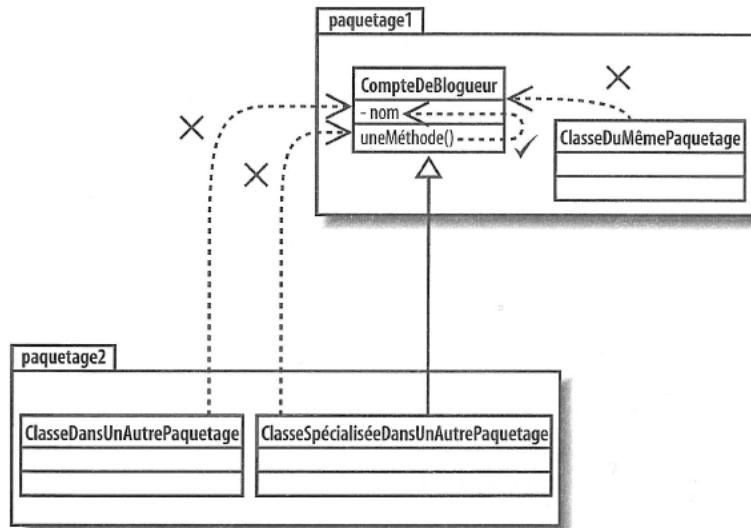
# Visibilité des attributs et méthodes

## • Publique



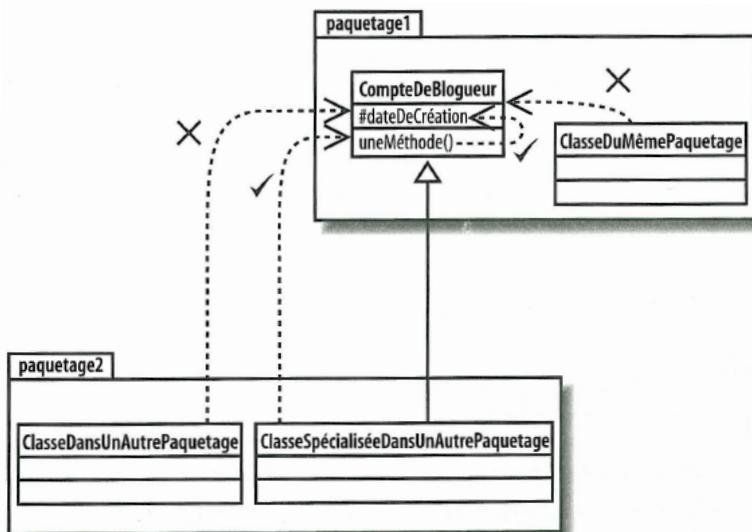
# Visibilité des attributs et méthodes

- Privée



# Visibilité des attributs et méthodes

- Protected



# Construction d'un diagramme de classes

- Trouver les classes

- Faire une liste de candidates (s'aider des diagrammes d'interaction)
- Eliminer les classes redondantes, superflues
- Donner de bons noms! -> "**PersonneQuiTravailleDansUnEcurieDeCourse**"

- Trouver les associations entre elles

- Verbes qui mettent en relation plusieurs classes (par exemple : est composé de)
- A nouveau s'appuyer sur les diagrammes d'interaction

- Trouver les attributs

- Correspondent généralement à des substantifs, ex. la masse d'une voiture
- Les adjectifs représentent souvent des valeurs d'attribut, ex. rouge

- Organiser et simplifier le diagramme en utilisant la généralisation/  
spécialisation

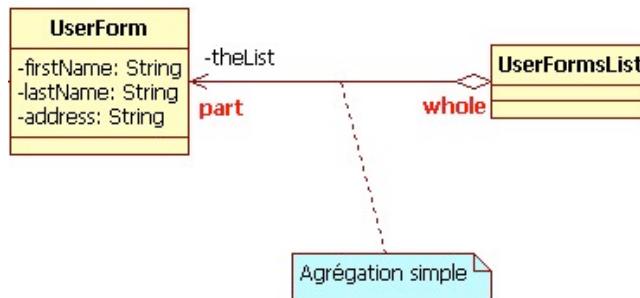
- véhicule – voiture – etc.
- ....à itérer et raffiner pendant la phase de conception!!

## Raffinement des associations (1)

- A ce stade les associations définissent un lien sémantique entre classes (le quoi)
- Elles n'indique pas précisément comment cette association est faite
  - Quelles sont les conséquences sur les classes impliqués
- On peut raffiner les associations en précisant de quelle genre d'association il s'agit
  - Agrégation
  - Composition

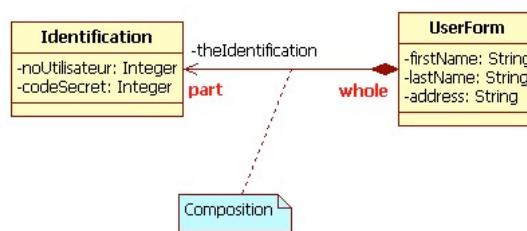
## Raffinement des associations (2)

- L'agrégation modélise une relation du type “**has a**” (possède-un)
- Les parties ont une vie indépendante du tout



## Raffinement des associations (3)

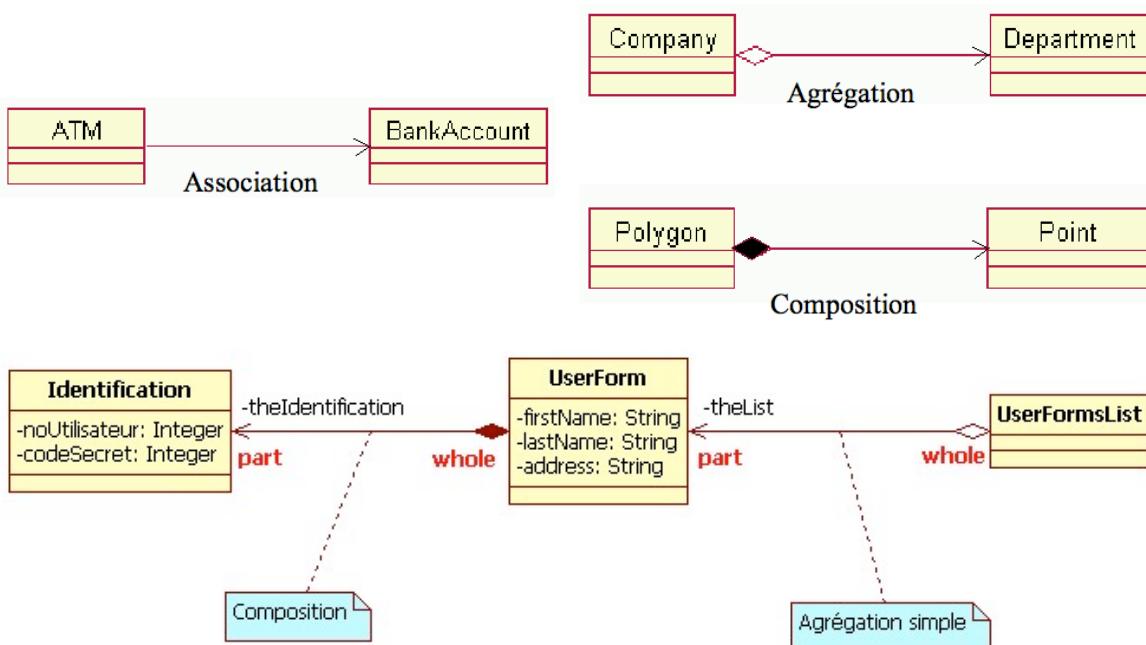
- Composition plus restrictive
- Composition = agrégation + 2 contraintes supplémentaires
  - 1<sup>ère</sup> contrainte = une partie constituante ne peut pas appartenir à plus d'un assemblage
  - 2<sup>ème</sup> contrainte = une fois une partie constituante assignée à un assemblage, sa durée de vie coïncide avec celle de l'assemblage
    - La destruction d'un objet déclenche celle de tous les objets qui le constituent



# Raffinement des associations (4)

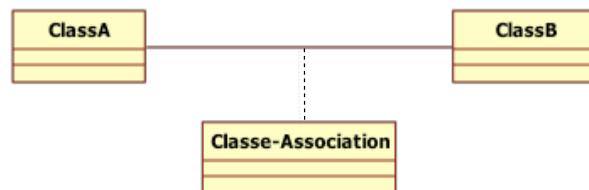
- **Agrégation:** si deux objets sont étroitement liés par une relation constitué-constituant
- **Association:** si deux objets sont indépendants même s'ils sont souvent liés
- **S'agit-il d'une agrégation ?**
  - Peut-on utiliser l'expression « fait partie de » ?
  - Les opérations appliquées à l'objets s'appliquent-elles automatiquement à ses constituants ?
  - L'association présente-t-elle une asymétrie intrinsèque, dans laquelle une classe est subordonnée à une autre ?

# Raffinement des associations (5)



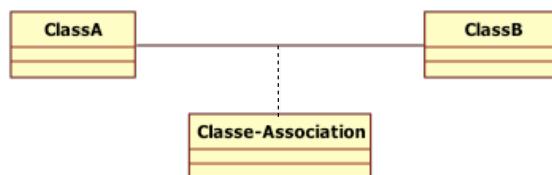
# Les classes-association (1)

- Dans certains cas on aimerait raffiner une association
  - Indiquer des propriétés propres à l'association
  - Dans UML1 seules les classes peuvent avoir des propriétés
- Pour résoudre ce problème UML 2 définit
  - Les classes-association
- Classe-association
  - C'est une classe reliée à une association entre classes



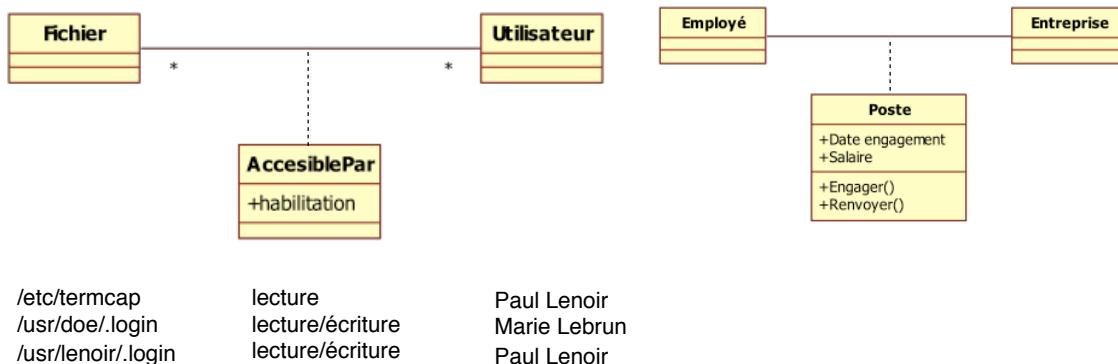
# Classe-association (2)

- Remarques:
  - On ne donne pas de nom à une association qui est relié à une classe-association
    - C'est le nom de la classe association



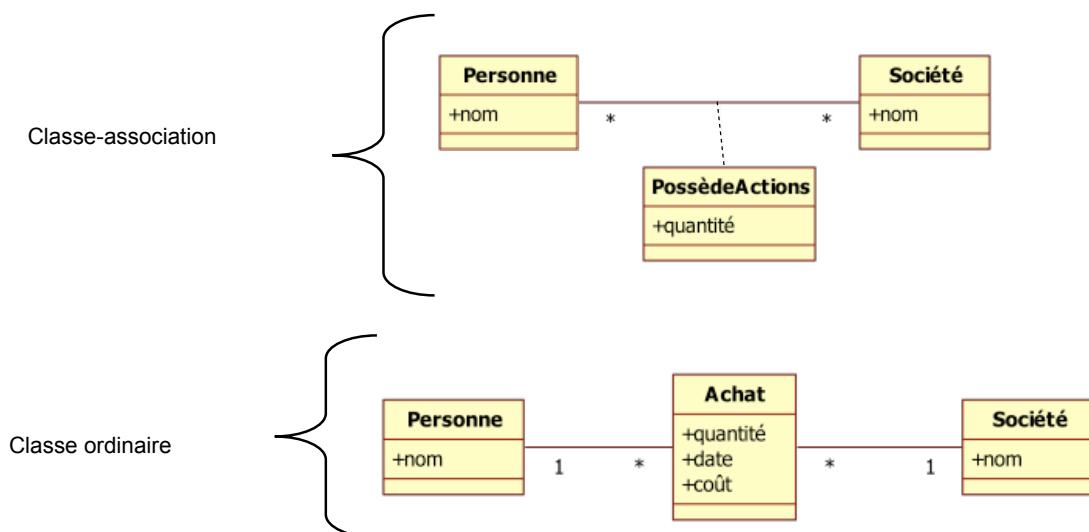
## Classe-association (3)

- Les attributs d'une classe-association appartiennent au lien lui-même et ne peuvent pas être attribués à aucun objet
- Ex: attribut habilitation



## Classe-association (4)

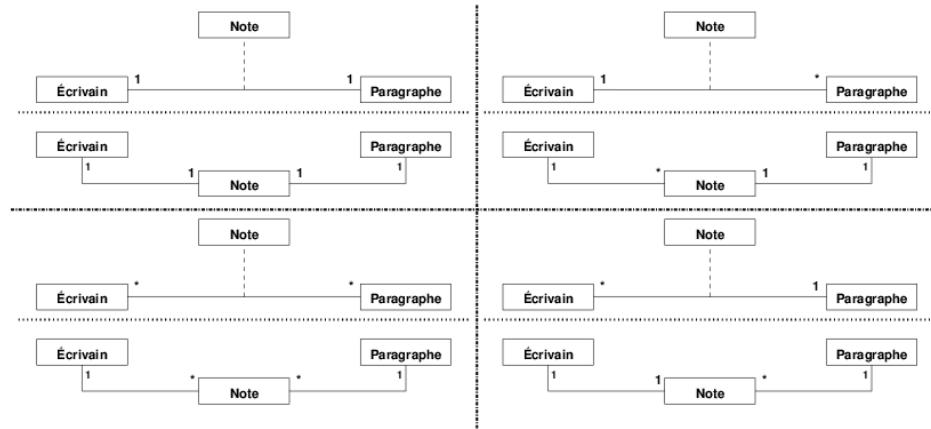
- Traduction des classes-association



# Traduction des classes d'association

La conception des classes d'association peut conduire à plusieurs solutions. La plus courante est de transformer la classe d'association en classe intermédiaire. L'association initiale est découpée en deux : une de la première classe vers la classe intermédiaire et une autre de la classe intermédiaire vers la seconde classe. Attention aux multiplicités : si l'association initiale est de type « 1 – \* », la première association est de type « 1 – 1 » et la seconde est de type « \* – 1 ».

La figure qui suit donne les quatre possibilités de configuration des multiplicités avec les traductions correspondantes.

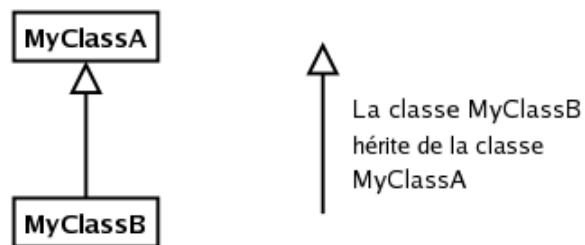


<http://www-inf.int-evry.fr/cours/CSC4002/EnLigne/Cours/CoursUML/7.40.html>

## Relations d'héritage (1)

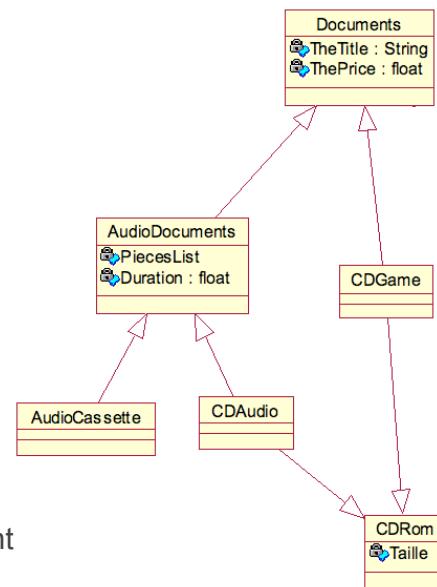
### • Propriétés de l'héritage

- Classe enfant possède toutes les propriétés de la classe parent
  - Il ne peut pas accéder au propriété privées directement mais les possèdeClasse enfant peut redéfinir les méthodes non-privées de la classe parent
- Toutes les associations de la classe parent s'appliquent à la classe enfant
- Une instance d'une classe peut être utilisé partout où une instance de sa classe parent est attendue (principe de substitution)
  - l'inverse n'est en général pas vrai !



# Relations d'héritage (2)

- L'héritage modélise une relation du type « **is a** » (« est-un » ou « sorte de »)
- On parle alors de
  - Spécialisation (cas particulier)
    - Créer des sous-classes qui raffinent les super-classes
    - Une sous-classe ne devrait jamais restreindre le comportement ou la structure d'une super-classe
  - Généralisation (un modèle)
    - Créer des super-classes qui encapsulent la structure et le comportement communs à plusieurs classes
    - Très habituel au début de l'analyse car les classes existantes sont celles qui modèlent le monde réel



## Exercice 2

- Modélez à l'aide d'un diagramme de classes la situation suivante:
  - Une personne est caractérisée par son nom, son prénom, son sexe et sa date de naissance. Pour une personne on désire pouvoir calculer son âge, le revenu annuel et le paiement des charges sociales.
  - Le revenu annuel est le salaire annuel net de la personne auquel s'ajoute une prime d'ancienneté égale à + 1% par année d'ancienneté. Les charges sociales sont calculées avec un coefficient fixe de 15% sur le salaire brut.
  - Une personne a un poste dans une entreprise qui est caractérisé par un salaire mensuel.

## Exercice 3

- Donnez un diagramme de classes correspondant au code source suivant

```
public class Cercle extends Figure {  
    private float rayon;  
    private Point centre;  
    public Cercle ( Point centre, float rayon) { ... }  
    public void dessiner () { ... }  
    public void effacer () { ... }  
}  
public class Rectangle extends Figure {  
    protected Point sommets[] = new Point[2];  
    public Rectangle ( Point p1, Point p2) { ... }  
    public void dessiner () { ... }  
    public void effacer () { ... }  
}  
public class Losange extends Figure {  
    protected Point sommets[] = new Point[2];  
    public Losange ( Point p1, Point p2) { ... }  
    public void dessiner () { ... }  
    public void effacer () { ... }  
}  
  
public interface Dessinable {  
    public void dessiner ();  
    public void effacer ();  
}  
abstract public class Figure implements Dessinable {  
    protected String couleur;  
    protected String getCouleur () { return couleur; }  
    protected void setCouleur ( String c ) { couleur = c; }  
}  
public class Point {  
    private float x;  
    private float y;  
    public float getX () { return x; }  
    public float getY () { return y; }  
    public void Point ( float x, float y) { ... }  
}
```

## Relations d'héritage (3)

- Modèle abstrait d'un CD-audio
  - Un CD-audio possède
    - Un titre
    - Type (petit, normal, double)
    - Une durée
    - Une liste de morceaux
    - Un prix
- Généralisation CD-audio -> document-audio
  - Un document-audio possède
    - Un titre
    - Une durée
    - Une liste de morceaux
    - Un prix

## Exercice 4

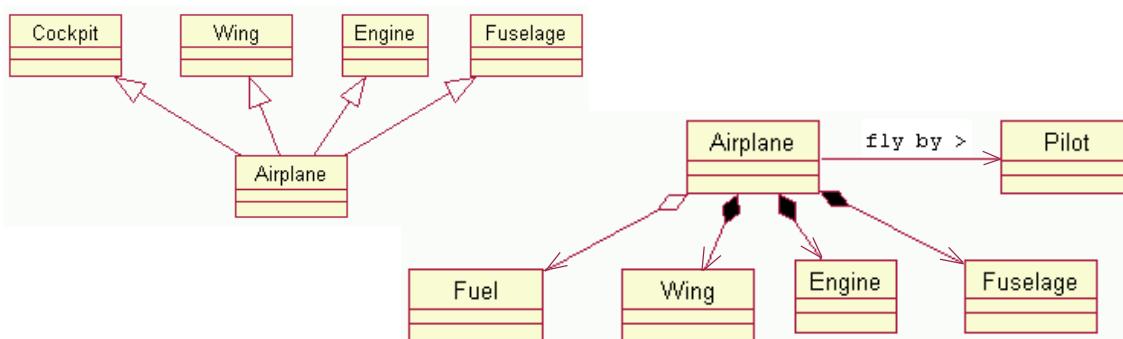
- Analysez ces classes et utilisez la généralisation pour factoriser au mieux la description des propriétés.

Banque	Directeur	Employé	Client
- <i>nomDirecteur</i> : String - <i>capital</i> : int - <i>adresseSiege</i> : String  + <i>getNomDirecteur()</i> : String + <i>setNomDirecteur</i> (String n) + <i>getCapital()</i> :int + <i>setCapital</i> (int capital) + <i>getAdresseSiege()</i> :String + <i>setAdresseSiege</i> (String s) <i>Banque</i> (String Adresse)	- <i>nom</i> : String - <i>prenom</i> : String - <i>revenu</i> : float  + <i>getNom()</i> : String + <i>setNom</i> (String n) + <i>getPrenom</i> ():String + <i>setPrenom</i> (String p) + <i>getRevenu</i> ():float + <i>setRevenu</i> (float s)	- <i>nom</i> : String - <i>prenom</i> : String - <i>dateEmbauche</i> : Date  + <i>getNom</i> : String + <i>setNom</i> (String n) + <i>getPrenom</i> ():String + <i>setPrenom</i> (String s) + <i>getRevenu</i> (): Date + <i>setDate</i> (Date s)  <i>mutation</i> (Agence g): boolean	- <i>nom</i> : String - <i>prenom</i> : String - <i>adresse</i> : String - <i>conseiller</i> : Employeur - <i>agence</i> : Agence - <i>comptes</i> : [1..N] Compte  + <i>getNom</i> : String + <i>setNom</i> (String n) + <i>getPrenom</i> ():String + <i>setPrenom</i> (String s) + <i>getDate</i> (): Date + <i>setDate</i> (Dates)  <i>mutation</i> (Agence g):boolean
<i>CompteNonRémunéré</i>	<i>Agence</i>	<i>CompteRémunéré</i>	
- <i>solde</i> : float - <i>numero</i> : int  ...	- <i>nomAgence</i> :String - <i>adresseAgence</i> : String  + <i>getNomAgence</i> () : String + <i>setNomAgence</i> (String n)	- <i>solde</i> : float - <i>numero</i> : int - <i>taux</i> : float  ... <i>verserInteret()</i> : void	

## Questions

Est-ce que ces deux diagrammes sont sémantiquement correcte ?

Est-ce que ces deux diagrammes modélisent le même comportement ?

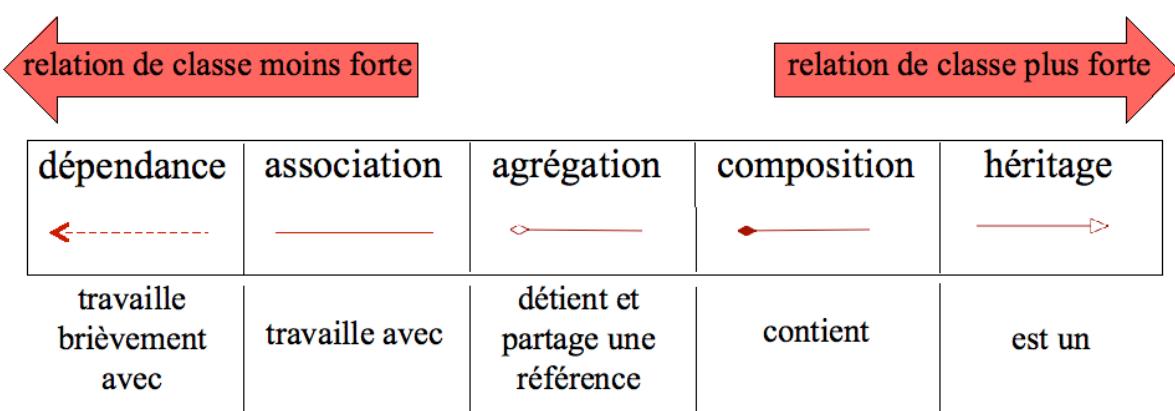


## Exercice 5

- Donnez un diagramme de classes correspondant à la donnée suivante :
  - Les étudiants et les enseignants sont deux sortes de personnes. Proposez un modèle de classes correspondant.
  - Un doctorant est un étudiant qui assure des enseignements. Complétez le modèle de classes précédent.
  - Les doctorants et les étudiants doivent s'inscrire au début de l'année et éventuellement modifier leur inscription. On connaît le nom et le prénom de toutes les personnes. On doit pouvoir calculer le salaire des doctorants aussi bien que celui des enseignants. Ajoutez ces éléments au modèle précédent.

## Relations entre classes

- (Association), agrégation, ou composition: si une classe **a** un élément qui est un objet d'une autre classe
- Généralisation: si une classe **est** du même genre qu'une autre classe



# Exercice 6

- **Modélez à l'aide de diagramme de classes chacunes des situations suivantes**
  - Tout écrivain a écrit au moins une oeuvre
  - Les personnes peuvent être associées à des universités en tant qu'étudiants aussi bien qu'en tant que professeurs.
  - Un rectangle a deux sommets qui sont des points. On construit un rectangle à partir des coordonnées de deux points. Il est possible de calculer sa surface et son périmètre, ou encore de le translater.
  - Les cinémas sont composés de plusieurs salles. Les films sont projetés dans des salles. Les projections correspondantes ont lieu à chacune à une heure déterminée.
  - Tous les jours, le facteur distribue des recommandés dans une zone géographique qui lui est affectée. Les habitants sont aussi associés à une zone géographique. Les recommandés sont de deux sortes : lettres ou colis. Comme plusieurs facteurs peuvent intervenir sur la même zone, on souhaite, pour chaque recommandé, le facteur qui l'a distribué, en plus du destinataire.