- **Semestre 2** R2.01 Développement orienté objets : UML
- **Semestre 3** R3.03 Analyse : UML

CM N°1	Système et Modèle	UML	Cas d'utilisation	
CM N°2	Déploiement	Classes		
CM N°3	**Dépendances**	**Associations**		
CM N°4	Code et dépendances	Code et associations		
CM N°5	Héritage	Stéréotypes	Abstraites	Interfaces
CM N°6	Rappels: Cas d'utilisation, classes, associations			
CM N°7	Rappels : Héritage, abstraites, interfaces			
CM N°8	Objets	Communication	Séquences	
CM N°9	Fragments	Scénarios	Séquences et code	
CM N°10	États-Transitions	Activités	Paquets	

R2.01 Développement orienté objets

Cours N°3 : UML : Les diagrammes de classes : les associations

Sommaire

E.Porcq : R2.01 UML
Département : IUT Caen Informatique

Année universitaire : 2024-2025

Sources bibliographiques :

-Cours "M2104 Approche qualité" de P.Brutus

- Cours de Laurent AUDIBERT

http://remy-manu.no-ip.biz/

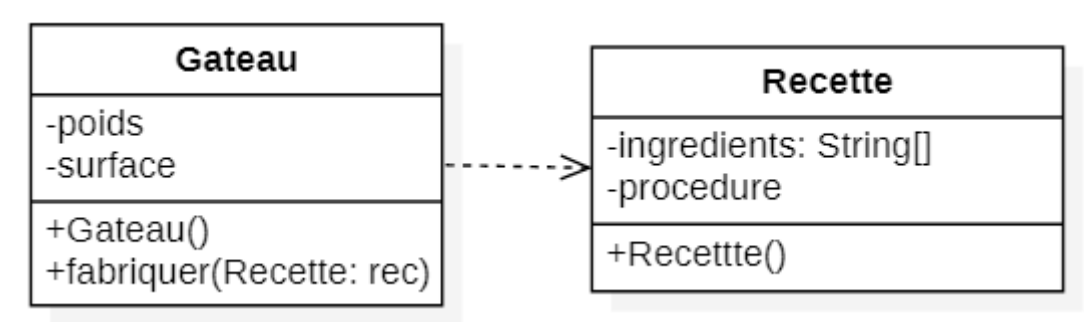
R2.01 Développement orienté objets

Cours N°3 : UML : Les diagrammes de classes : les associations

Les dépendances

1) Les dépendances

- Une classe peut en **utiliser** une autre. Elle **dépend** alors de cette autre classe.
- La dépendance se représente par une **flèche** à trait **discontinu**.
- La dépendance est la forme la plus **faible** de relation entre classes.
- Il s'agit d'une relation **transitoire**, au sens où la première interagit brièvement avec la seconde sans conserver à terme de relation avec elle (liaison **ponctuelle**).



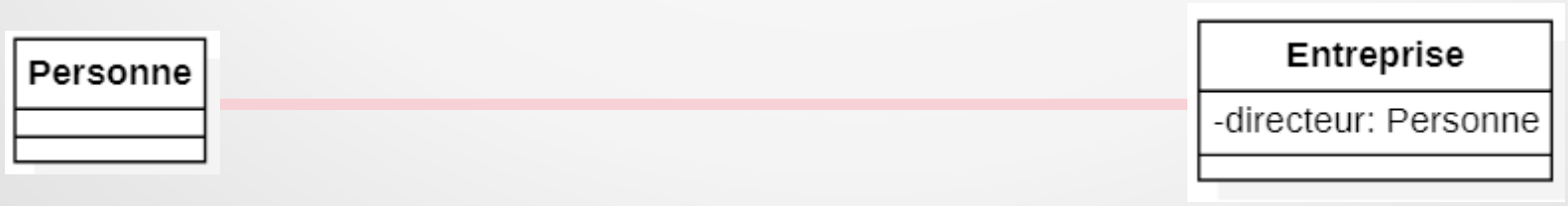
Quand fabriquer a terminé

Gateau n'a plus de lien avec Recette

Gateau ne connaît plus la Recette

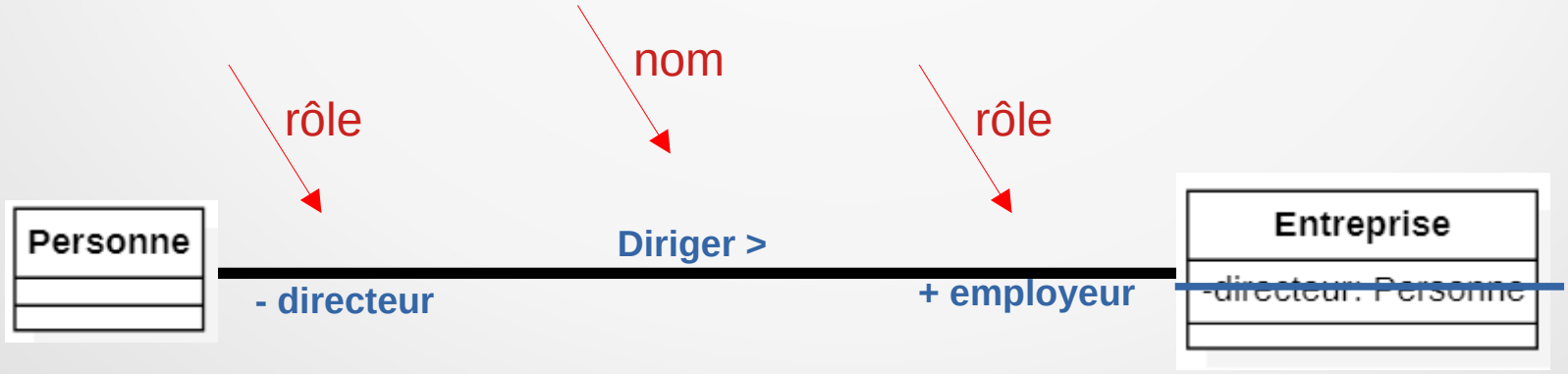
2) Les associations simples

- L'association simple (ou association) signifie qu'une classe **contiendra** une référence de l'objet de la classe associée sous la forme d'une **propriété**.
- Cette relation est plus **forte**. Elle indique qu'une classe est en **relation** avec une autre pendant un certain laps de temps. La ligne de vie des deux objets concernés ne sont cependant pas associés étroitement (un objet peut être **détruit** sans que l'autre le soit nécessairement).
- L'association se représente par une ligne à trait **continu**.



2-1) Noms et rôles des associations

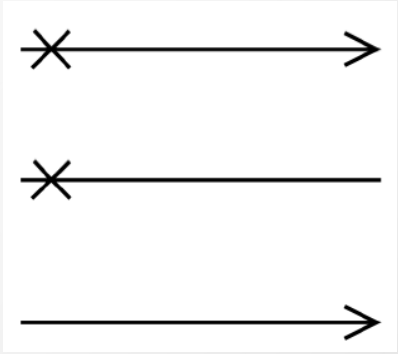
- L'association peut avoir un **nom** avec un éventuel **sens** de lecture qui permet de nous informer de **l'intérêt** de cette relation.
- Ce texte n'a aucune **signification** au niveau du **code**.
- Chaque extrémité d'une association peut être **nommée**. Ce nom est appelé **rôle** et indique la manière dont l'objet est vu de l'autre coté de l'association. Pour un rôle, on peut préciser la **visibilité**. Il a par contre une signification importante au niveau code.



L'attribut devient redondant

2-2) La navigabilité

- Les associations possèdent une navigation **bidirectionnelle** par **défaut**, c'est-à-dire qu'il est possible de déterminer les liens de l'association depuis une instance de chaque classe d'origine.
- Cela suppose que chaque classe possède un attribut qui fait référence à l'autre classe en association. Une navigation bidirectionnelle est plus complexe à réaliser au niveau codage et **rarement** nécessaire;
- Les représentations possibles et complètement équivalentes pour la navigation unidirectionnelle sont les suivantes :

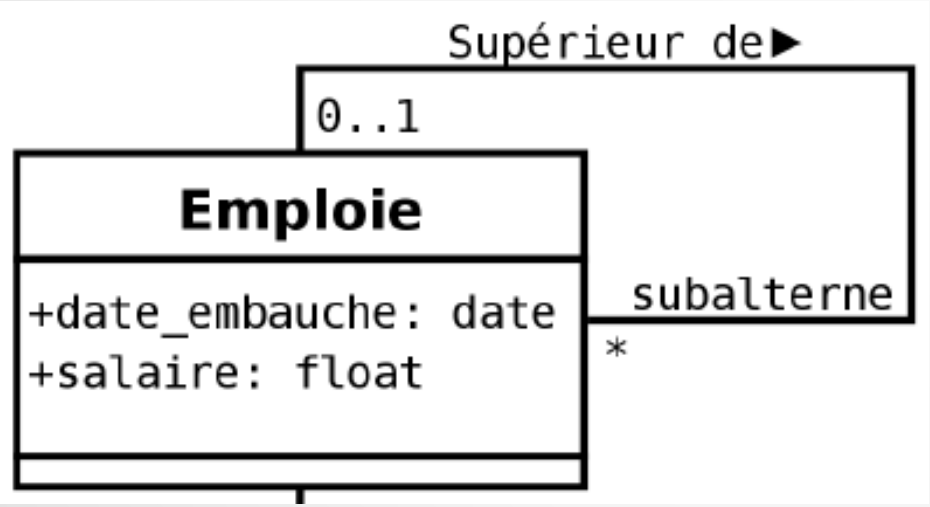


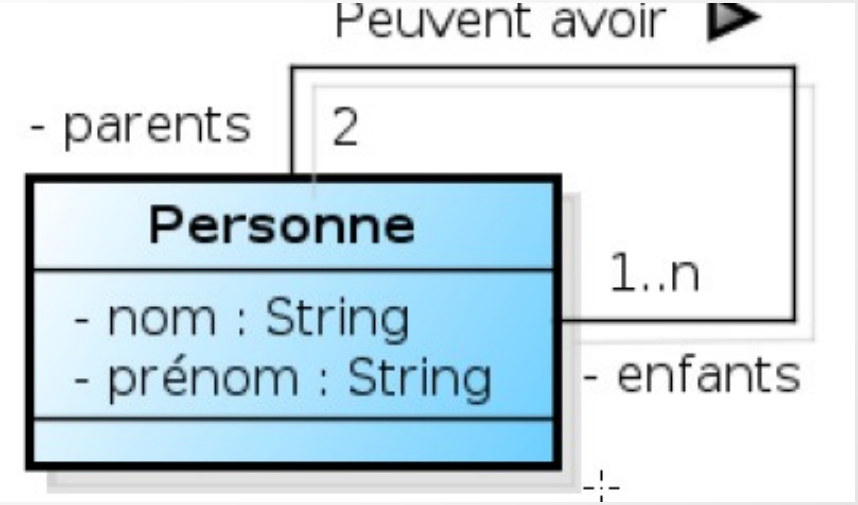
2-3) La multiplicité

- Les rôles montrent généralement les attributs réalisant l'association entre les classes. Il est possible de montrer la multiplicité (ou cardinalité) d'un rôle dans l'association.
- On trouve en général les représentations suivantes :
  - 0..1 0 à 1 instance
  - 0..* ou * 0 à plusieurs instances
  - 1 instance exactement
  - 1..* 1 à plusieurs instances
  - n..m n à m instances
  - n..* à plusieurs instances

2-4) L'association réflexive

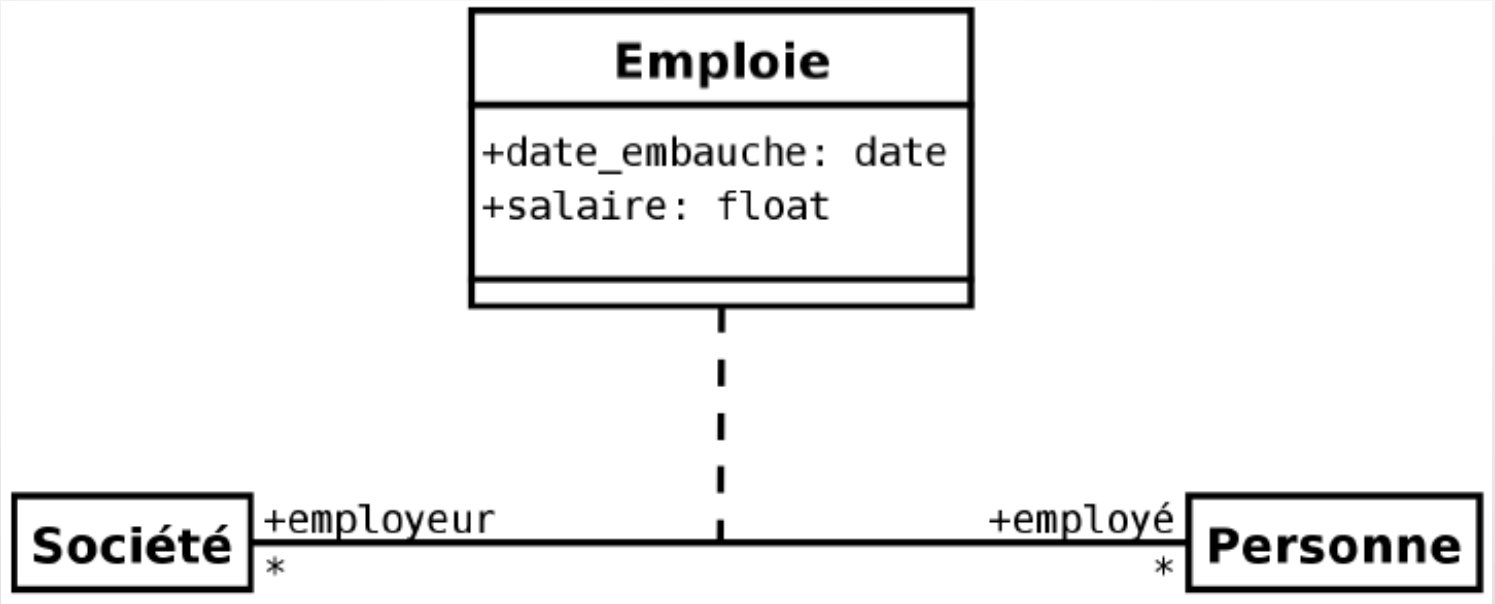
- Une association qui lie une classe avec elle-même est une association réflexive.





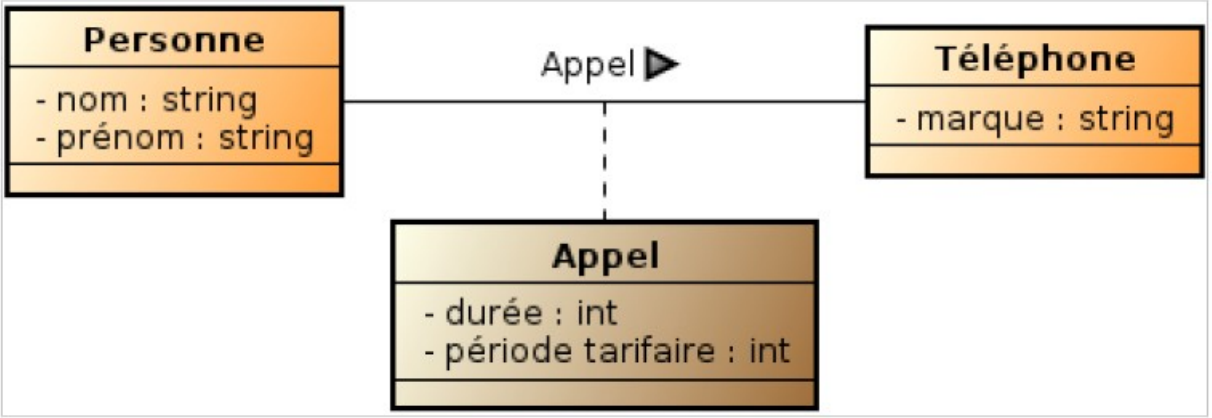
2-5) Classe d'association

- Parfois, une association doit posséder des propriétés. Par exemple, l'association Emploie entre une société et une personne possède comme propriétés le salaire et la date d'embauche. En effet, ces deux propriétés n'appartiennent ni à la société, ni aux personnes.
- Les associations ne pouvant posséder de propriété, il faut introduire un nouveau concept pour modéliser cette situation : celui de classe-association.



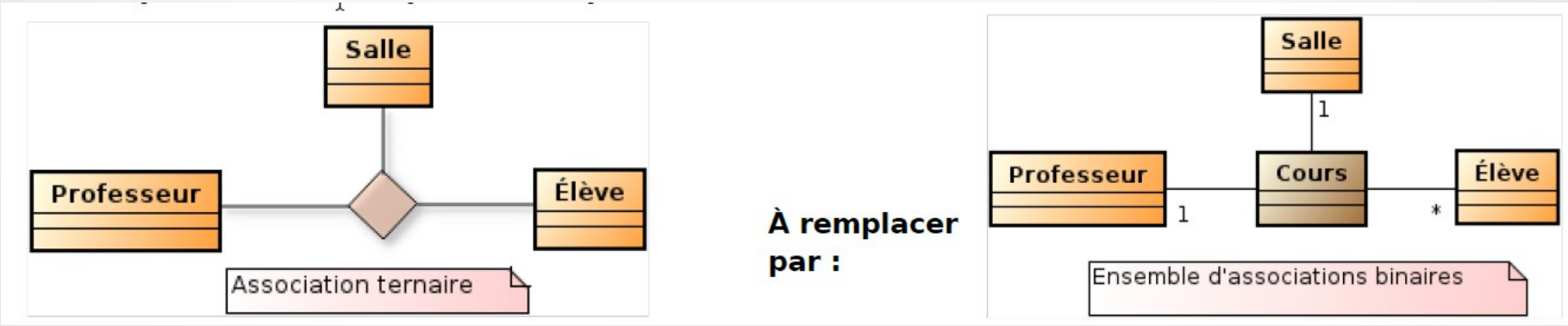
2-5) Classe d'association

- Parfois, l'information véhiculée par une classe-association peut être véhiculée sans perte d'une autre manière.
- Dans ce cas, la classe d'association est remplacée par 2 associations binaires.



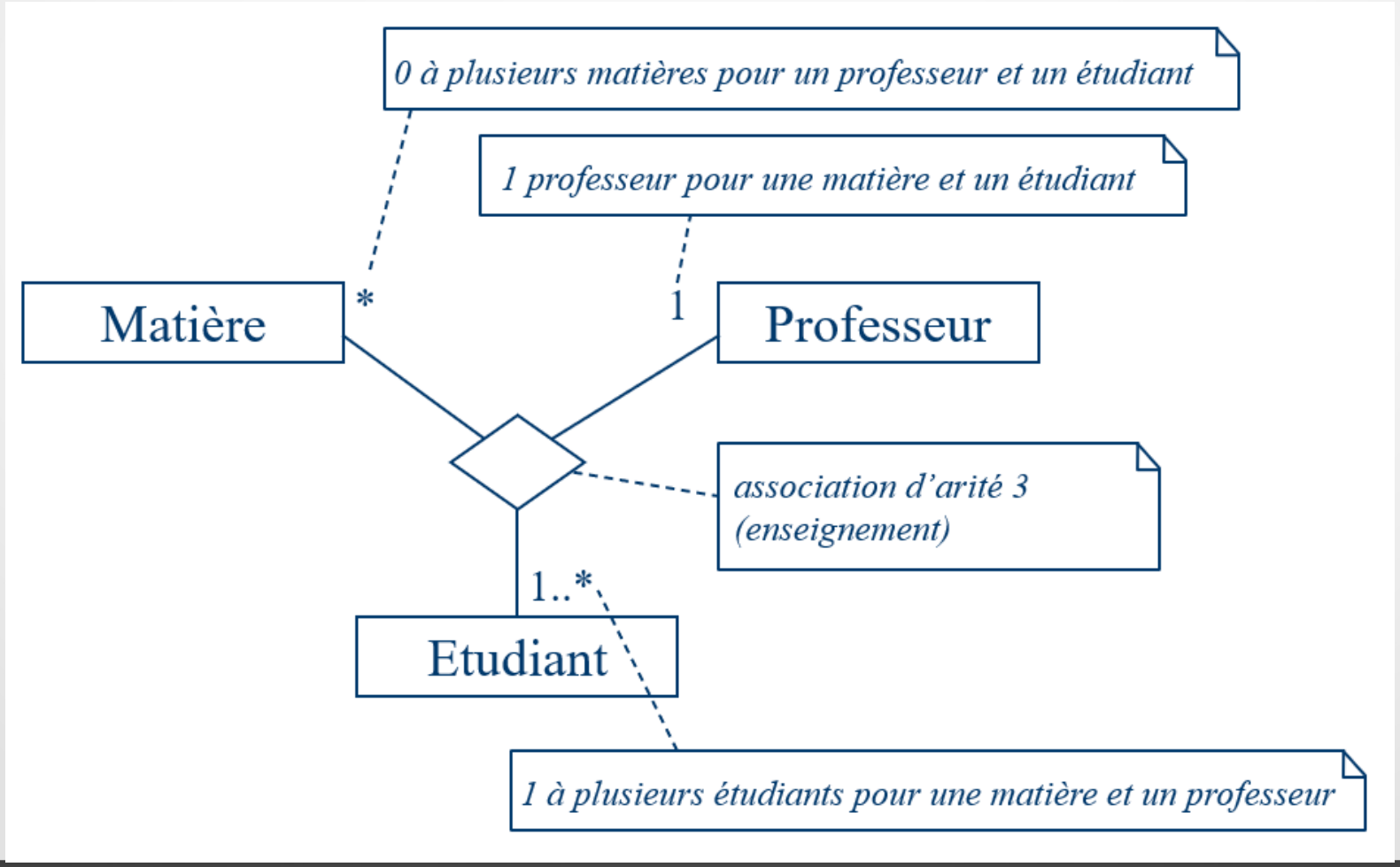
2-6) Association n-aire

- Une association qui lie plus de **2** classes entre elles, est une association n-aire.
- L'association n-aire se représente par un losange d'où part un trait allant à chaque classe.
- L'association n-aire est **imprécise**, difficile à interpréter et souvent source d'erreur, elle est donc très peu utilisée.
- La plupart du temps nous nous en servons que pour esquisser la modélisation au début du projet, puis elle est vite remplacée par un ensemble d'**associations binaires** afin de lever toute ambiguïté.



2-6) Association n-aire

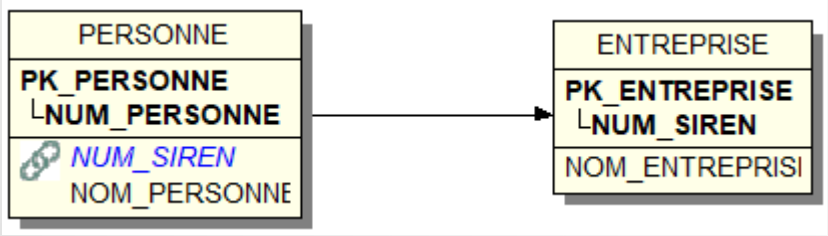
- Il n'est pas du tout aisé d'interpréter les multiplicités pour les associations n_aire.



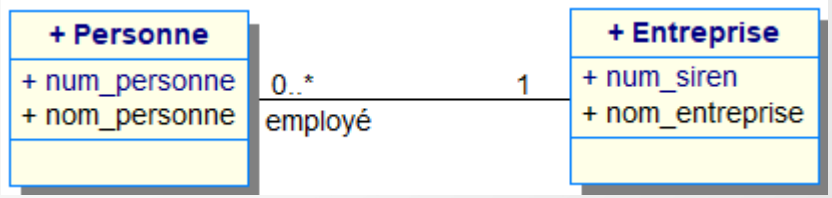
3-1) De Merise à UML

- Ces associations réalisées avec la notation UML ont elles un rapport avec la notation Merise étudiée en S1 ? Tout à fait
- Exemple d'une association sans propriété de type (*,n)-(1,1)

MCD



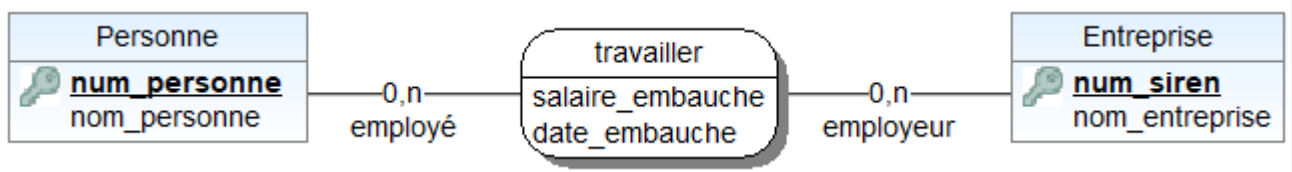
MLD



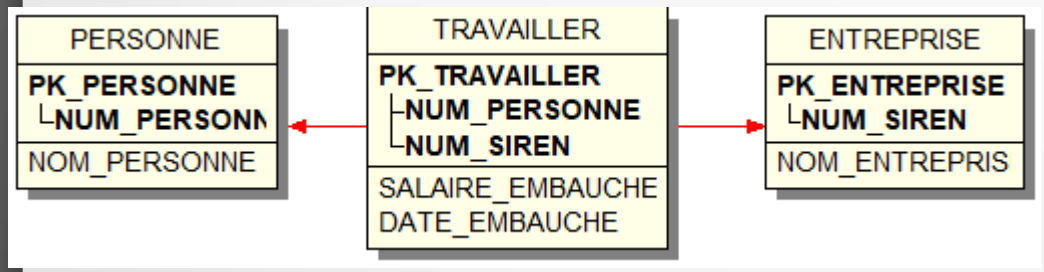
UML

3-1) De Merise à UML

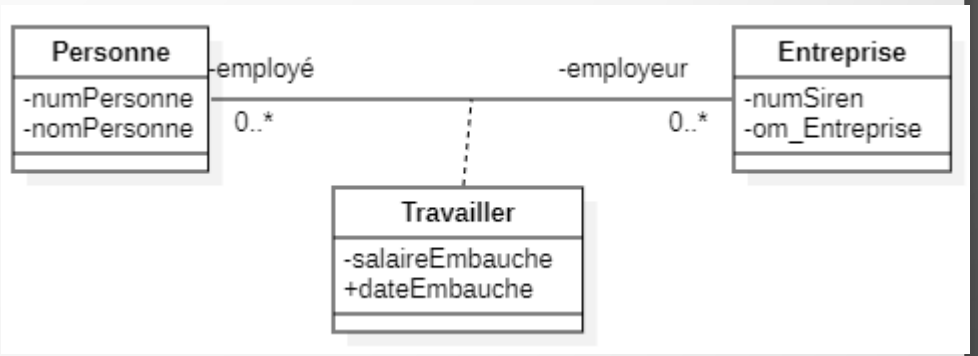
- Exemple d'une association avec propriété de type (*,n)-(*,n)



MCD



MLD



UML

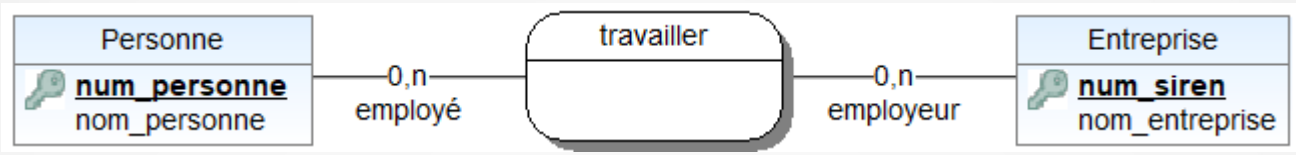
R2.01 Développement orienté objets

Cours N°3 : UML : Les diagrammes de classes : les associations

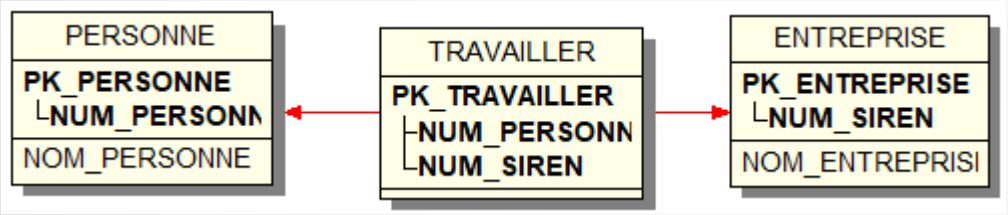
Classe d'association

3-1) De Merise à UML

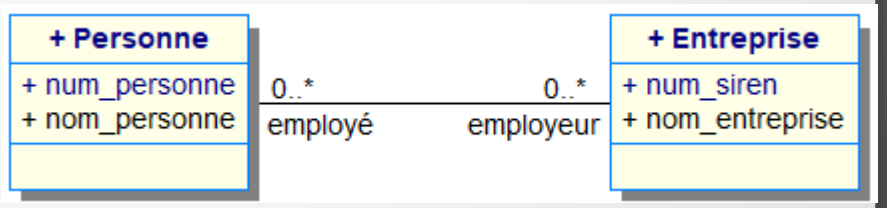
- Exemple d'une association avec propriété de type (*,n)-(*,n)



MCD



MLD



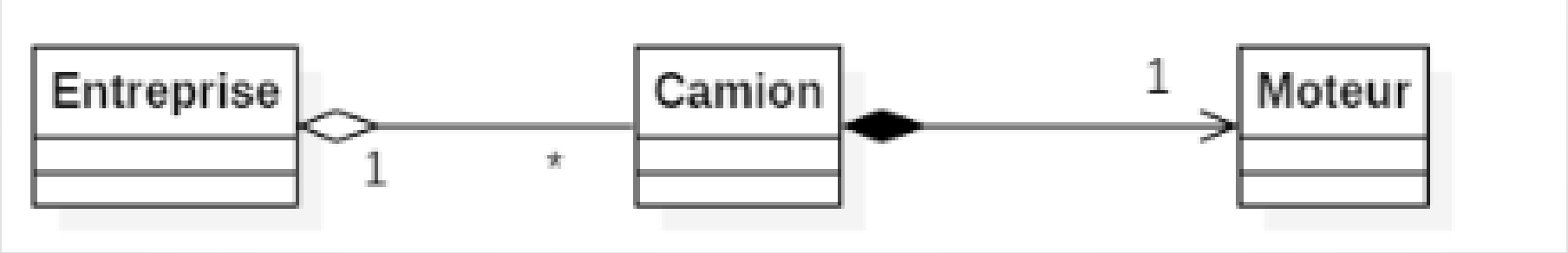
UML

4) L'agrégation

- Il n'est pas si évident de distinguer une association simple d'une agrégation.
- Une association simple entre deux classes représente une relation structurelle entre deux classes où aucune des deux **n'est plus importante** que l'autre.
- Lorsqu'un un élément est **contient** des éléments plus petit (partie), il faut utiliser une agrégation. Ce n'est qu'un point de vue **conceptuel**.
- On dit dans ce cas qu'un ou plusieurs objets A **contiennent** un ou plusieurs objets B.
- La **création** de l'objet A n'entraîne **pas obligatoirement la création** de l'objet B.
- La **suppression** de l'objet A n'entraîne pas **obligatoirement la suppression** de l'objet B

5) La composition

- La composition, également appelée agrégation composite ou agrégation forte, décrit une contenance structurelle entre instances.
- On dit dans ce cas qu'un **seul** objet A **est composé** d'un ou plusieurs objets B.
- La construction de l'objet composite **implique** la construction de ses composants .
- La **destruction** de l'objet composite implique la **destruction** de ses composants.
- La multiplicité du côté composite ne doit pas être supérieure à 1. Elle n'a pas besoin d'être représentée.

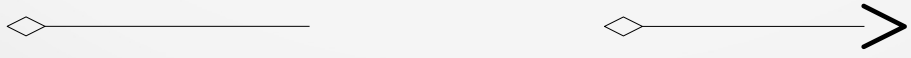


5) Rappels sur les différents liens

- La dépendance indique qu'une classe peut en **utiliser** une autre. Au niveau programmation, si l'objet A dépend de l'objet B, une instance de B peut être un **paramètre** ou une **variable locale** dans une méthode de A.
- L'association simple est plus forte. Elle est en général moins brève. Elle signifie qu'une classe A **contiendra** une référence de l'objet de la classe B associée sous la forme d'un **attribut**.
- L'agrégation est plus forte. Elle indique la notion **d'appartenance** entre A et B. Au niveau programmation, il est **possible** que cela soit A qui **fabrique ou détruise** des instances de B.
- La composition est plus forte. Les durées de vie des objets de la classe A et B sont les **mêmes**. Au niveau programmation, une ou plusieurs instances de B sont **créées** dans les attributs de A ou dans ses **constructeurs**.

5) Rappels sur les différents liens

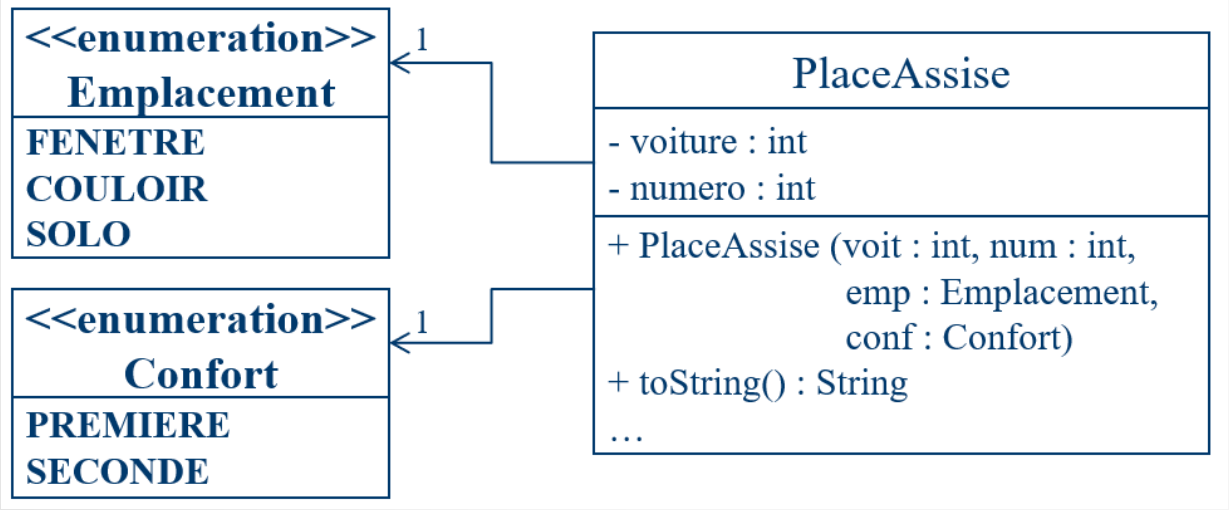
- 1) une association peut être une dépendance, une association simple, une agrégation ou une composition
- 2) une association peut être uni ou bidirectionnelle

- La dépendance 
- L'association 
- L'agrégation 
- La composition 

R2.01 Développement orienté objets
Cours N°3 : UML : Les diagrammes de classes : les associations
Les énumérations

7) Les énumérations

- Une énumération est un type pour des données qui ne peuvent prendre qu'un petit nombre de valeurs.
- Les valeurs possibles peuvent être codées par des entiers mais on préfère leur donner un nom.




```
public enum CouleurPrimaire {
  ROUGE, BLANC, ROSE
}
```