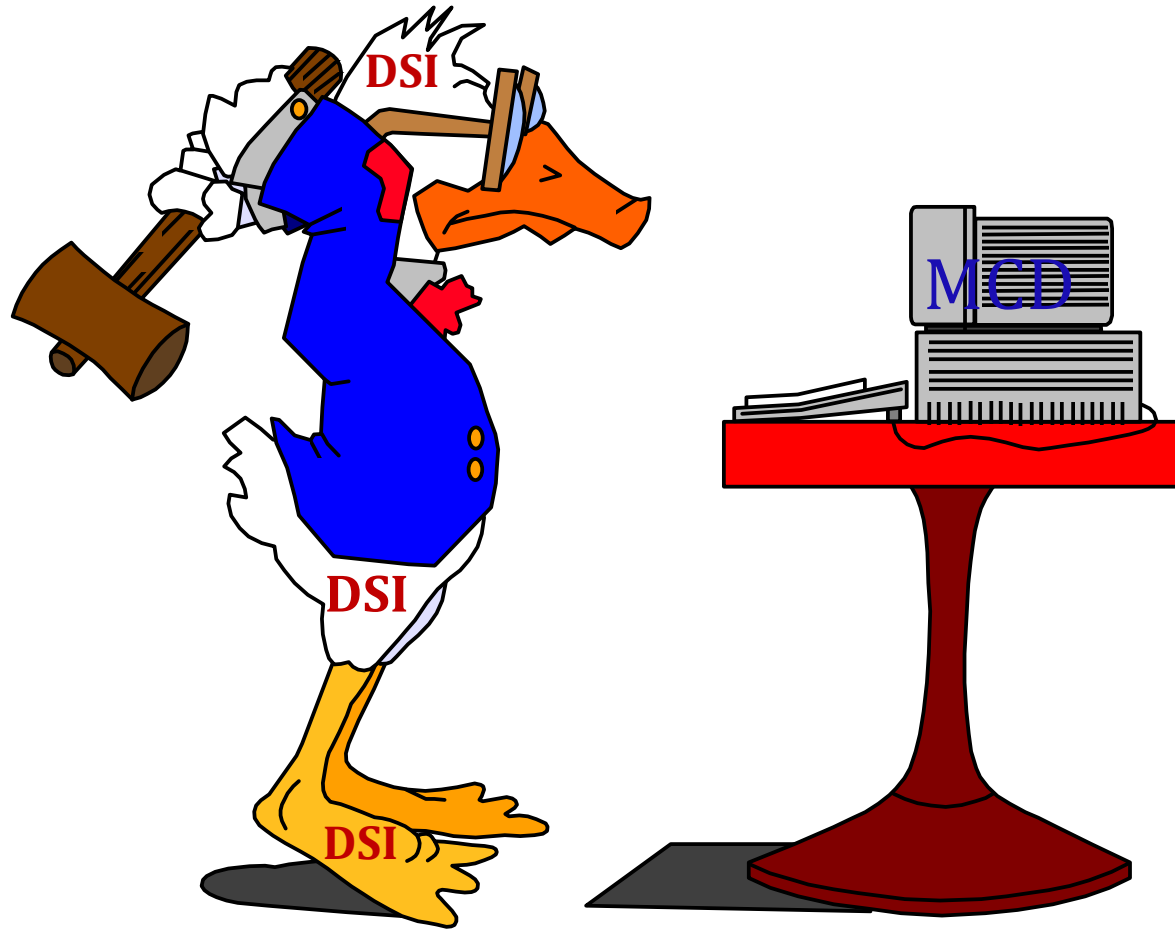


Conception Orientée Objet: UML



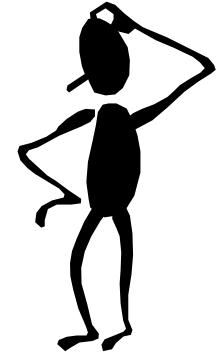
SupNum

Les classes d'objet

Définitions

- ❑ Une classe est la description d'un ensemble d'objets qui ont les mêmes attributs, les mêmes opérations, les mêmes relations et des sémantiques communes
ex : Personne, Etudiant, Compagnie,...
- ❑ Un attribut est une propriété nommée d'une classe qui décrit un ensemble de valeurs, il représente une propriété de l'élément modélisé
ex : nom, age, note, adresse,...
- ❑ Une opération est l'implantation d'un service qui peut être demandé aux objets d'une même classe dans le but de déclencher un comportement
ex : manger, marcher, virer, retirer, atterrir...

Qu 'est-ce qu 'un Objet ?



□ Définition

- c'est une abstraction du monde réel, *ce sur quoi porte notre connaissance*

□ Exemples:

- Objets matériels (table, chaise, crayon, avion...)
- Objets immatériels, concepts (compte en banque, match de boxe...)
- Objets virtuels (groupe de travail, division...)
- ...

- Tout objet possède les caractéristiques suivantes:

Objet = État + Comportement + Identité

Les Objets : *L'identité*

- ✓ Chaque objet possède une identité qui caractérise son existence propre
- ✓ L'identité permet de distinguer tout objet de façon non ambiguë, indépendamment de son état
- ✓ Permet de distinguer deux objets dont toutes les valeurs d'attributs sont identiques
 - 2 pommes de la même *couleur*, du même *poids* et de la même *taille* sont deux objets distincts.
 - 2 véhicules de la même *marque*, même *série* et ayant exactement les mêmes *options* sont aussi 2 objets distincts.

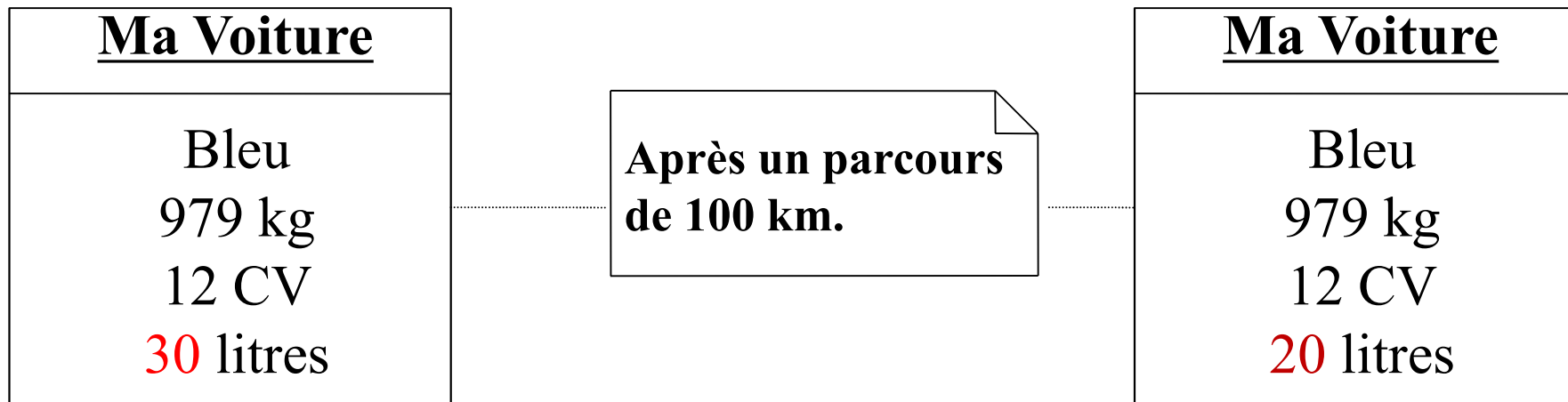
Les Objets : *L'état*

- Regroupe des valeurs instantanées de tous les attributs d'un objet.
- L'état d'un objet est décrit par l'ensemble de ses attributs et liens et est exprimé au travers des opérations.
- Exemple

<u>Ma Voiture</u>	
Bleu	← <i>Couleur</i>
979 kg	← <i>Masse</i>
12 CV	← <i>Puissance fiscale</i>
30 litres	← <i>Quantité de carburant</i>

Les Objets: L'état

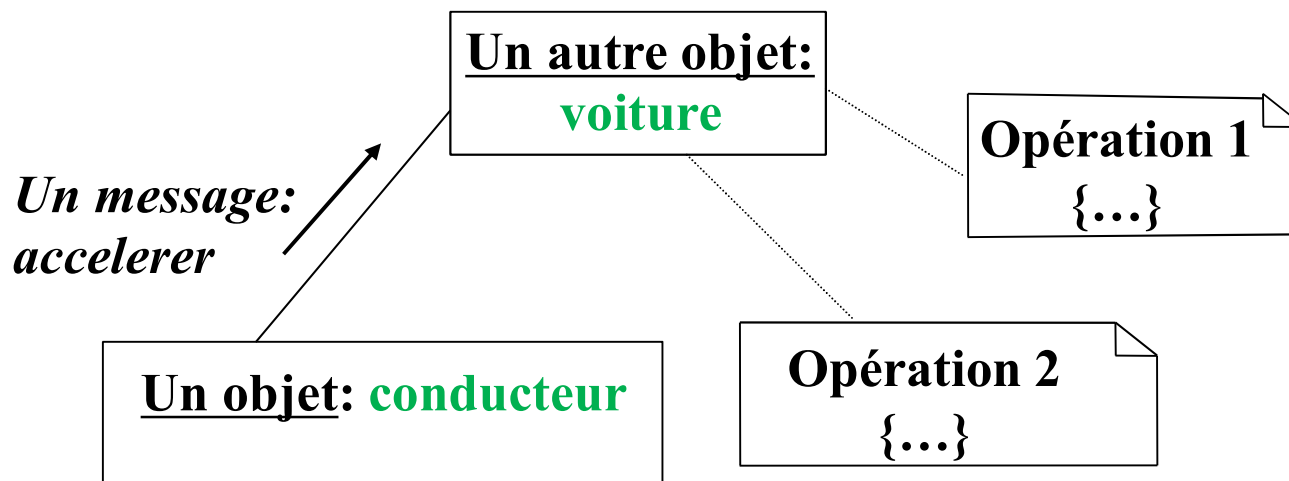
- **L'état évolue au cours du temps:**
 - Certaines valeurs d'attribut vont évoluer
 - D'autres vont être constantes
- **Exemple**



- *Une voiture roule, la quantité de carburant diminue, les pneus s'usent, etc.*
- *La marque de la voiture, pays de la construction de la voiture*

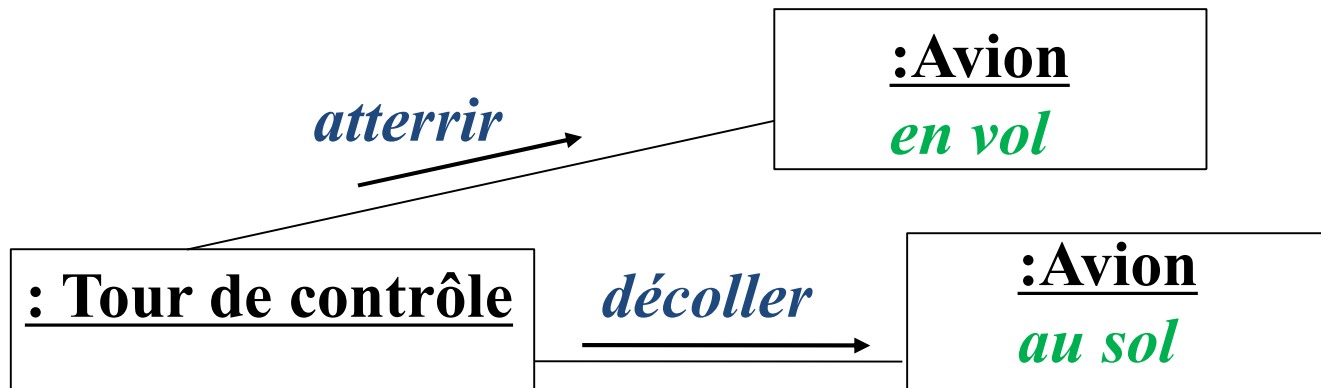
Les Objets: *Le Comportement*

- Regroupe l'ensemble des operations (ou méthodes) applicables à cet objet
- Décrit les actions et les réactions de l'objet.
 - Concept d'opérations (méthodes) ; une opération a une signature et s'applique à un objet
 - Une opération est déclenchée suite à une stimulation externe: message envoyé par un autre objet.

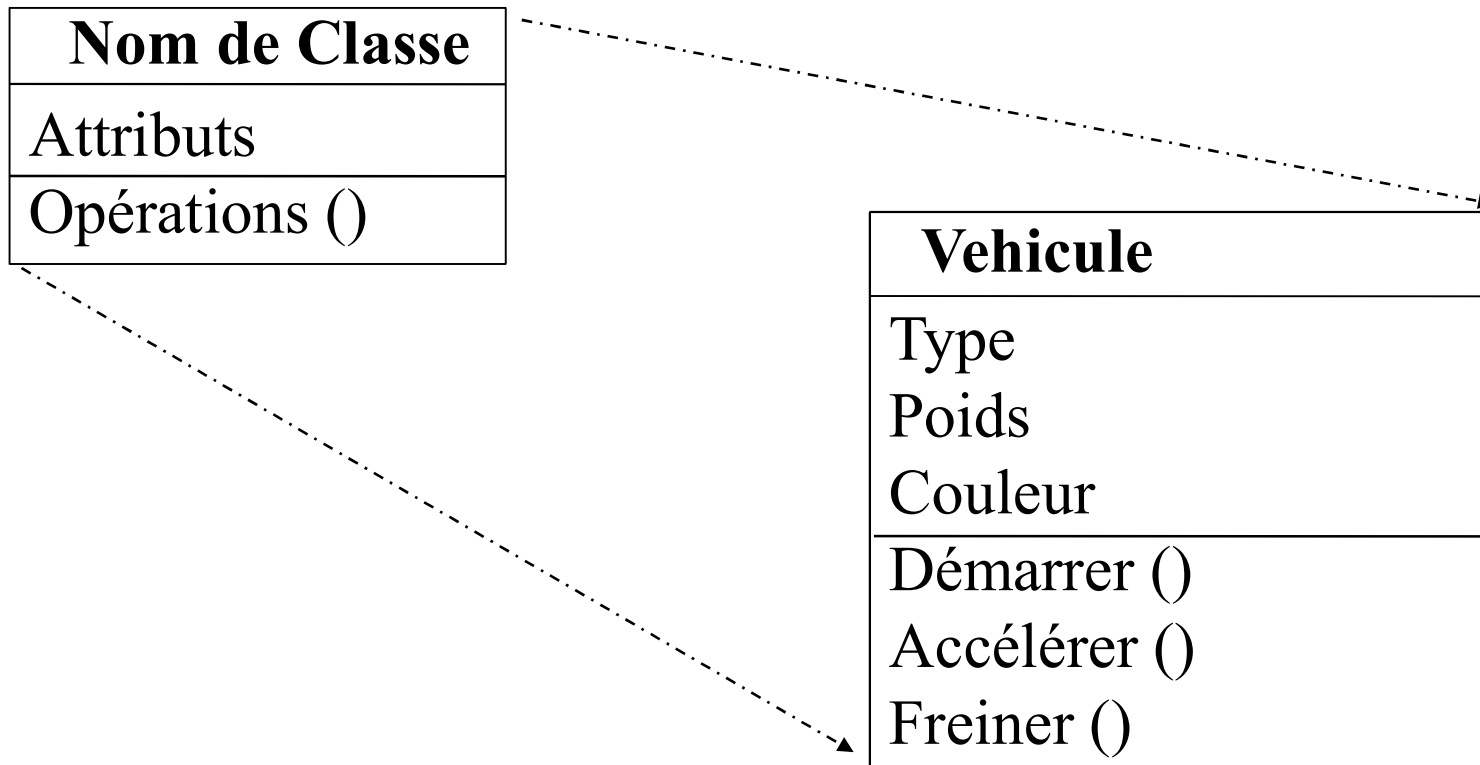


Les Objets: Caractéristiques

- **L'état et le comportement sont liés:**
 - Le comportement, à un instant donné, dépend de l'état courant.
 - L'état peut être *modifié* par le comportement.
- **Exemple: Un avion ne peut atterrir que s'il est en vol.**
 - *atterrir* représente un comportement de l'avion.
 - *en vol* correspond à un état de l'avion.

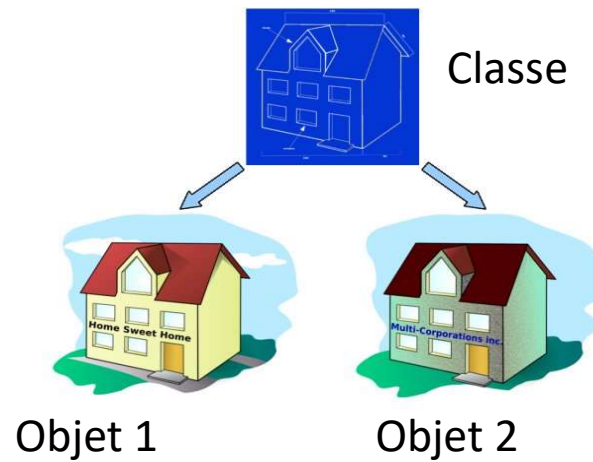


Représentation graphique des classes

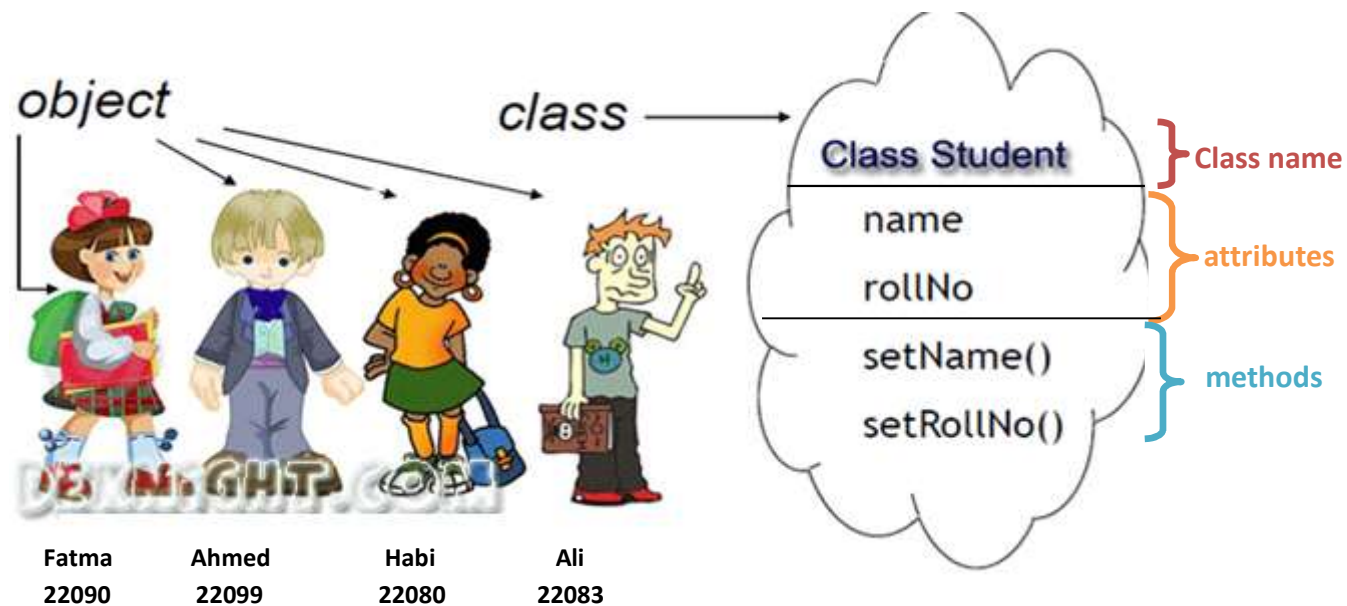


Classe *versus* Objet

Classe: un *moule* pour creer des objets.

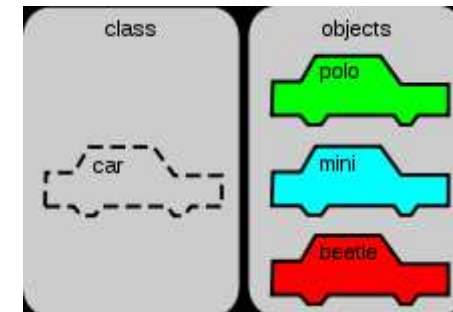


Classe *versus* Objet



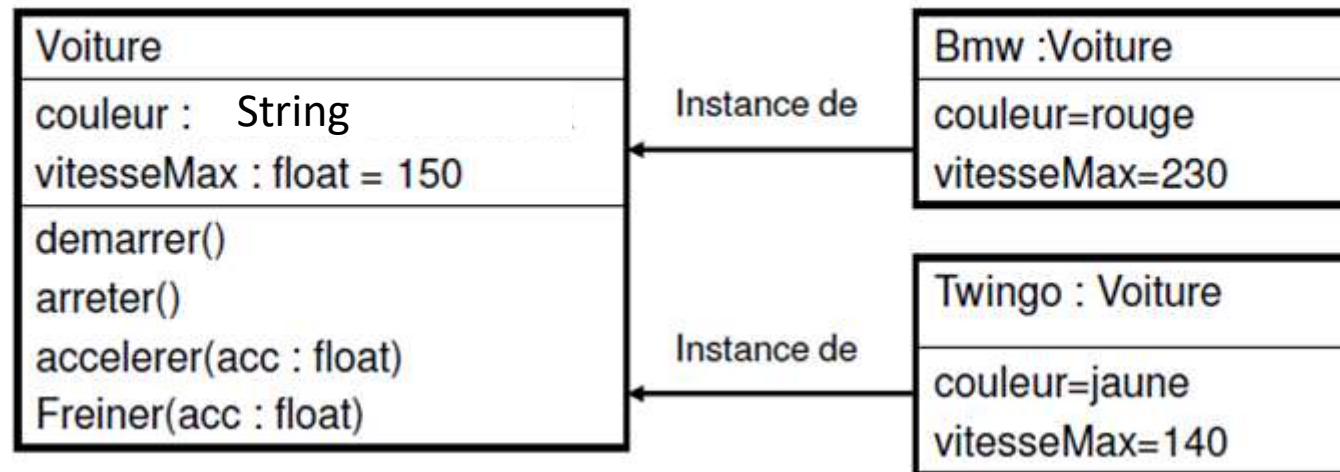
Classe *versus* Objet

Vehicle
-CurrentGear:Integer -CurrentSpeed:Integer -VehicleColor:Integer -WheelCount:Integer -DoorCount:Integer -Cylinders:Integer
+ChangeGear(GearNumber:Integer):Integer +TurnLeft():Integer +TurnRight():Integer +GoForward():Integer +GoBackward():Integer +GetSpeed():Integer +GetGear():Integer +GetColor():Integer +SetSpeed(DesiredSpeed:Integer):Integer -OKToShiftGears():Integer -IncreaseSpeed(DesiredSpeed:Integer):Integer -DecreaseSpeed(DesiredSpeed:Integer):Integer



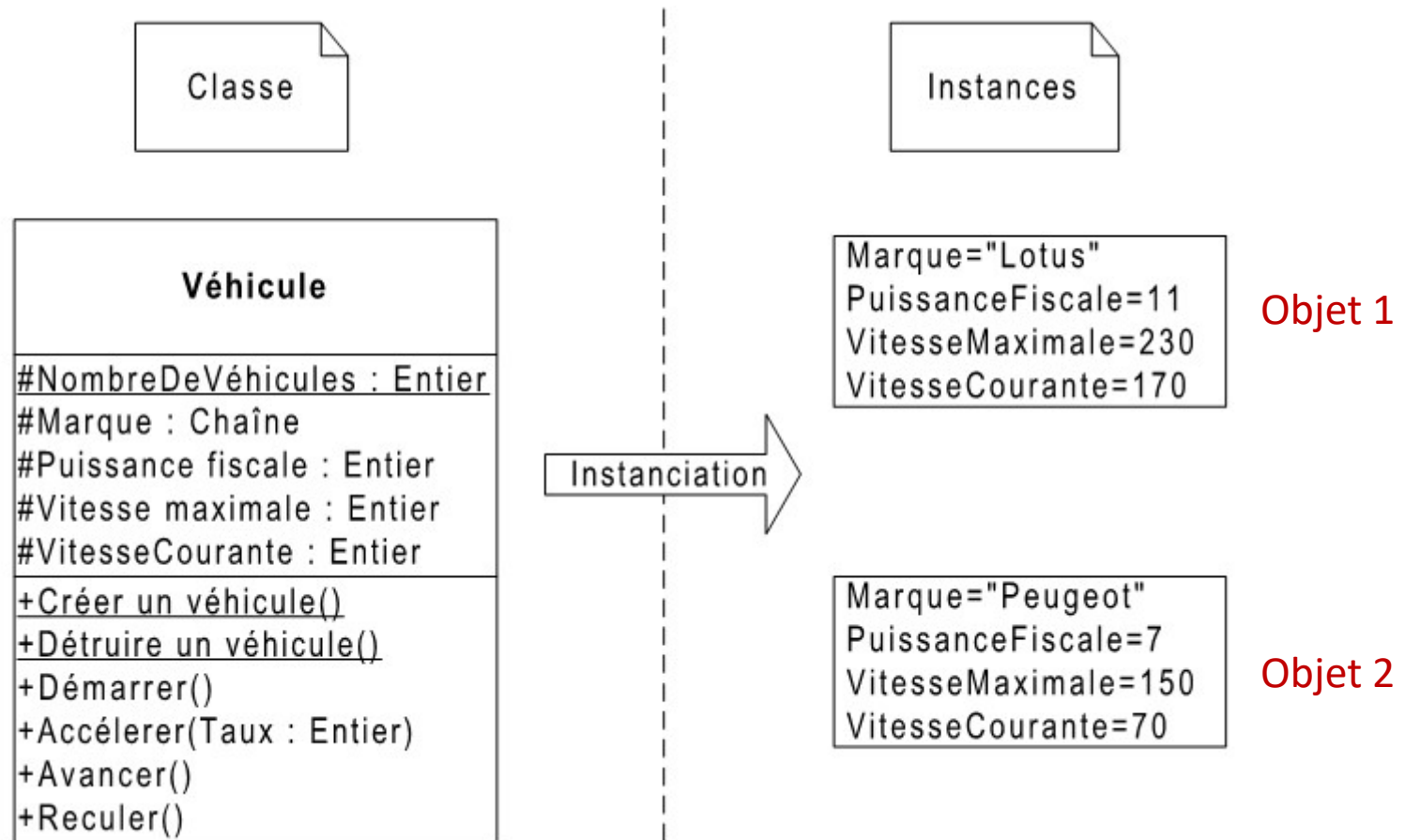
Classe et objets

- Les deux instances (objets) sont définies par une taille, un nombre et des types d'attributs et un ensemble de méthodes identiques.
- Par contre, les objets n'ont pas le même état.



Instanciation d'une Classe

Instanciation : concrétisation d'une classe en un objet « *concret* ».



Instanciation d'une classe en deux objets

Visibilité des attributs et des operations: Encapsulation

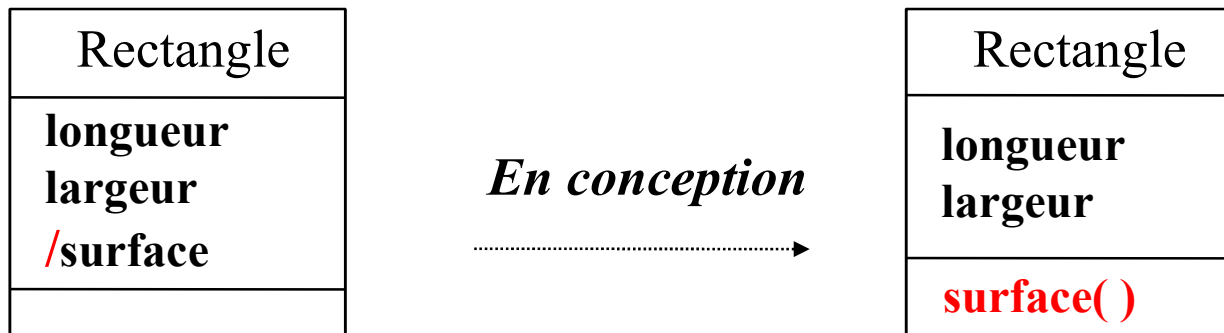
- **Access Modifier**

- determine les droits d'accès a la classe et ses membres
- Définie la ou` la classe et ses membres peuvent etre utilisés

- ✓ public (+)
- ✓ protected (#)
- ✓ private (-)
- ✓ package(~)

Attribut dérivé

- Les attributs dérivés offrent une solution pour allouer des propriétés à des classes, tout en indiquant clairement que ces propriétés sont dérivées d'autres propriétés déjà allouées.
- *Exemple:*



Les Relations entre classes

Lien et Association

□ Association

- Une relation sémantique entre classes
- Représente l'ensemble des liens entre les objets des classes qui participent à l'association

□ Degré d'une association(2,3,...)

- Le nombre de classes qui participent à l'association
- Une association peut être réflexive

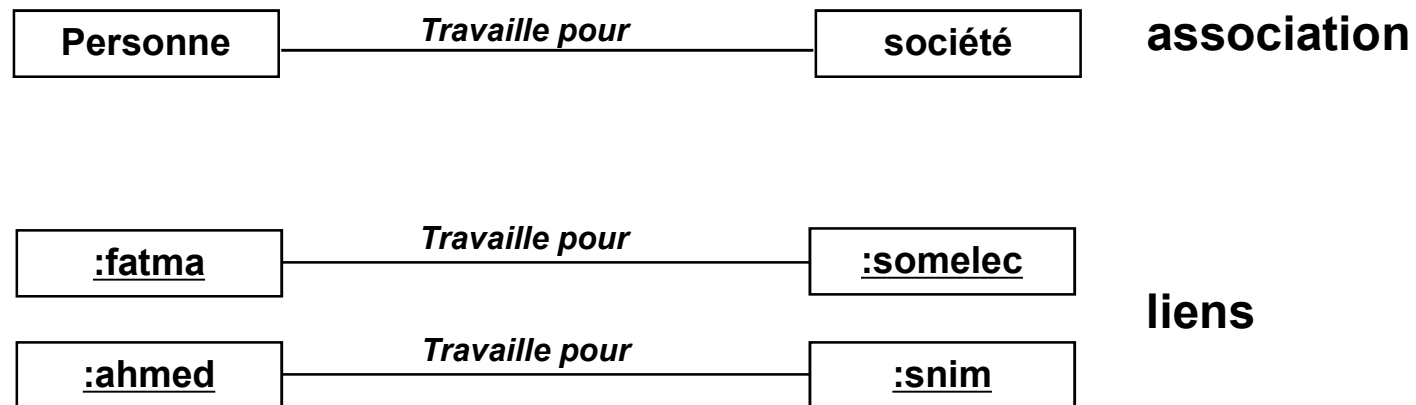
□ Lien

- Une connexion physique entre objets
- Une instance d'une association
- Similarité : classe-objet et association-lien

Les relations entre classes

Lien et Association

- Une association est une relation structurelle qui précise que les objets d'une classe sont reliés aux objets d'une autre classe

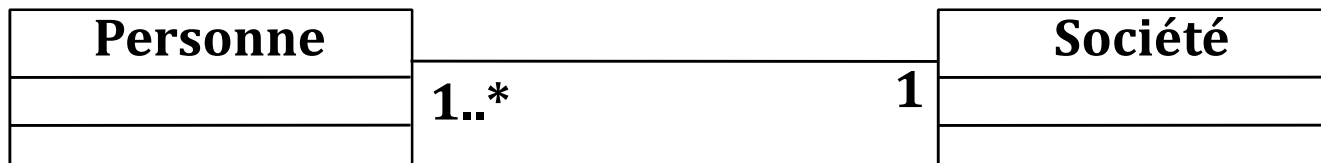


Lien et Association : multiplicité

- La multiplicité est une information qui définit combien d'objets de la classe considérée peuvent être liés à un objet de l'autre classe
- Les valeurs de multiplicité expriment des contraintes liées au domaine de l'application, valables durant l'existence des objets.



Chaque personne travaille pour une société, et chaque société emploie de une à plusieurs personnes.



Multiplicité = cardinalité

Multiplicité

Mérise	UML
Orienté BD relationnelles	BD orienté objet
Franco-français	International
Plus adapté a une approche théorique	Plus orienté vers la conception

Tableau 1. Terminologie

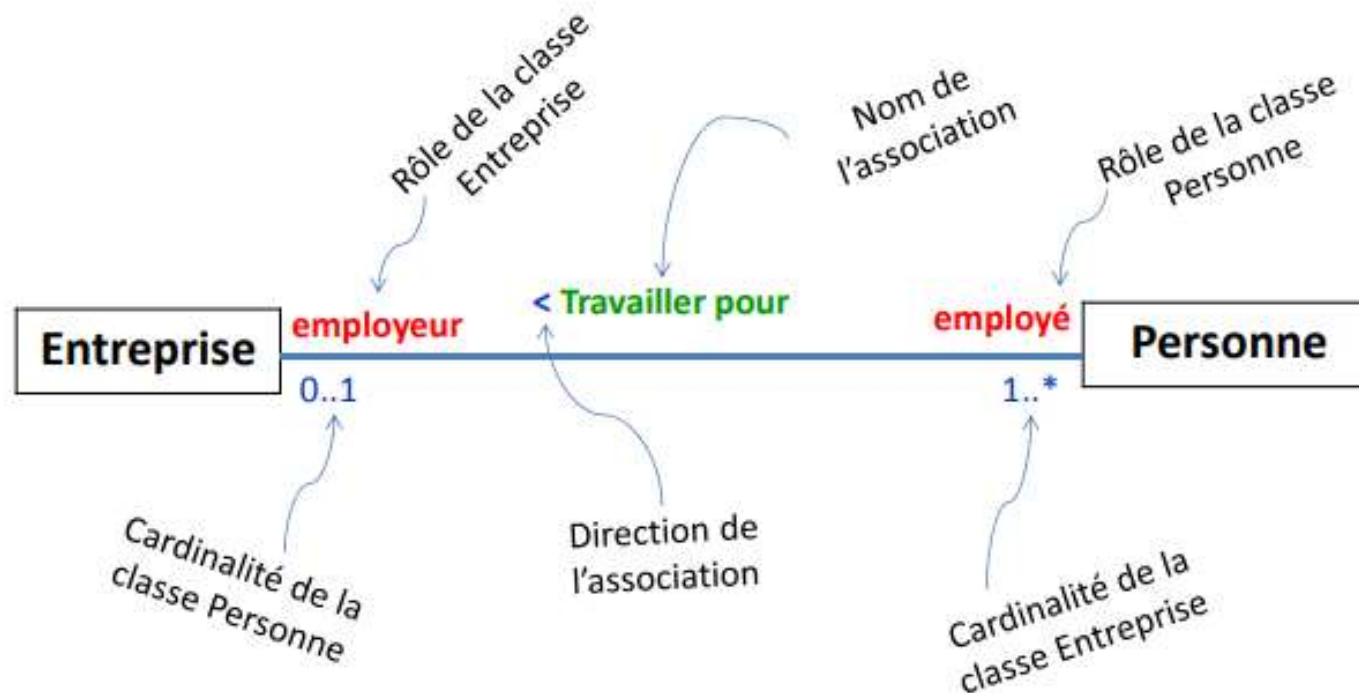
Entité-association (Merise)	UML
Entité	Classe
Association (Relation)	Association (Relation)
Occurrence	Objet
Cardinalité	Multiplicité
Modèle conceptuel de données	Diagramme de classes

Tableau 2. Cardinalités/multiplicités

Cardinalités	Multiplicités d'UML
0, 1	0..1
1, 1	1
0, N	0..* ou *
1, N	1..*
N, N	N..N

Lien et Association : rôle

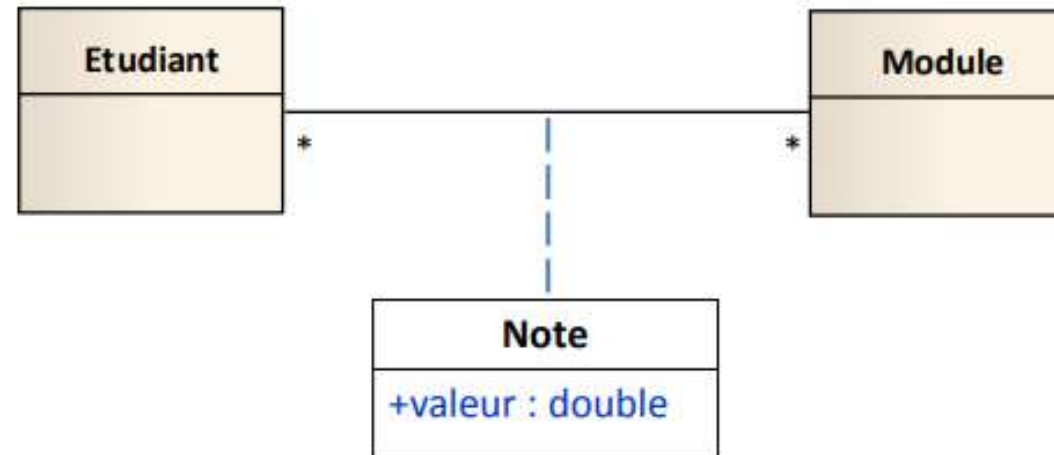
- "Rôle" des participants dans une association
 - Un "rôle" peut être spécifié pour une extrémité de l'association.
 - Il exprime le rôle d'une classe dans l'association.
 - il facilite la lecture et la compréhension du modèle objet.



Classe Association

Association et attributs

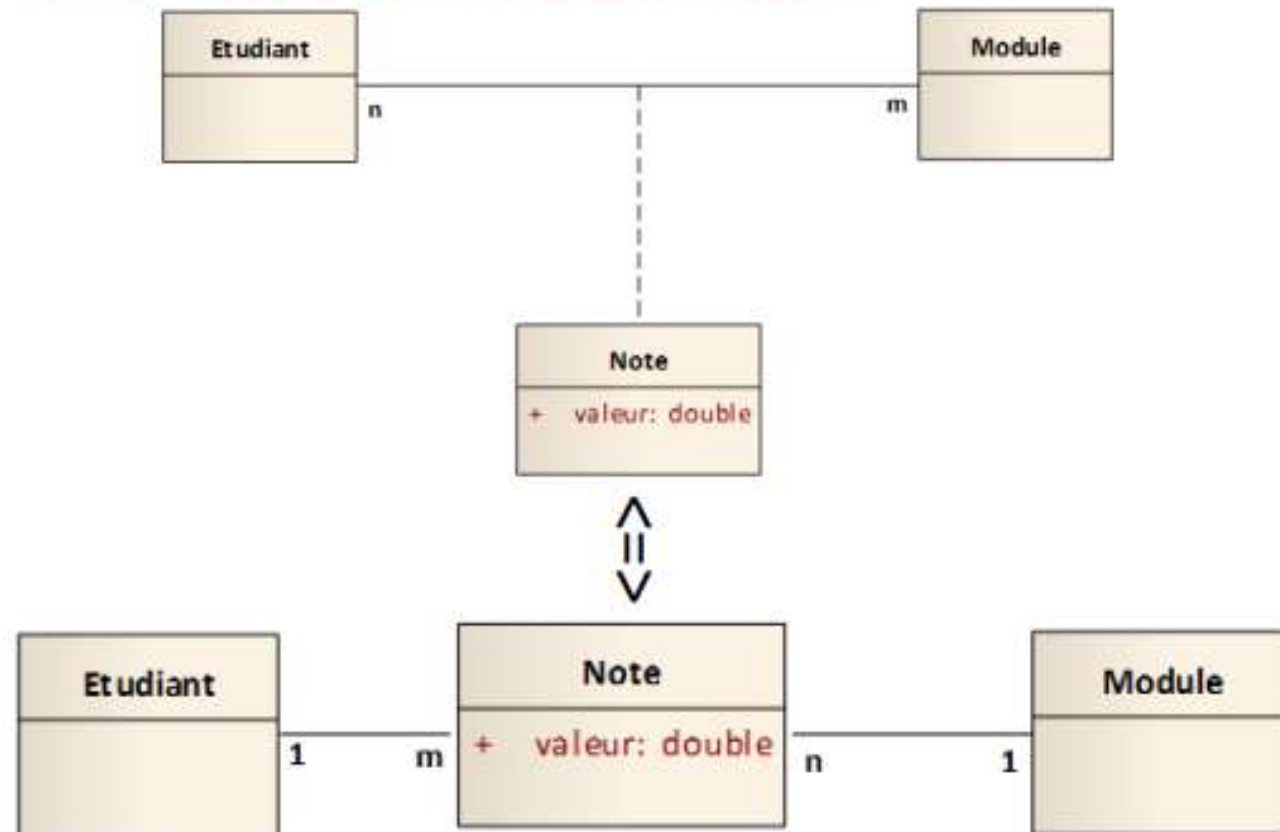
- Une association peut être raffinée et avoir ses propres attributs, qui ne sont disponibles dans aucune des classes qu'elle lie.
 - Car , dans le modèle objet, seules les classes peuvent avoir des attributs
- Cette association est représentée par une **classe d'association**
- Exemple ci-dessous : où mettre la note?



Classe Association

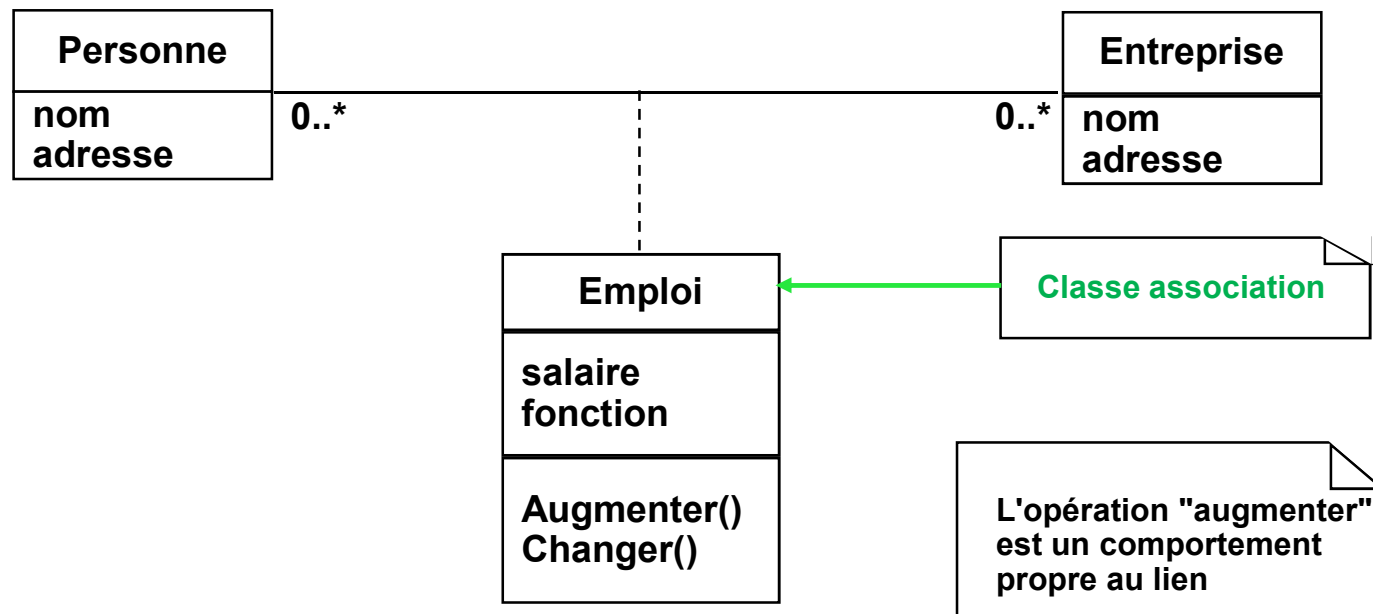
Classe d'association

– Signification d'une classe d'association



Classe Association

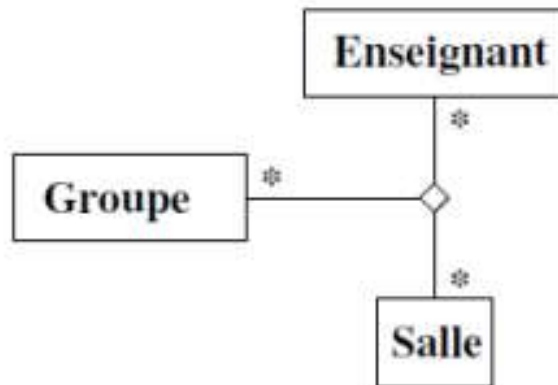
- Chaque lien est une instance de la classe qui modélise l'association
- A utiliser quand les liens ont des opérations



Association n-aires

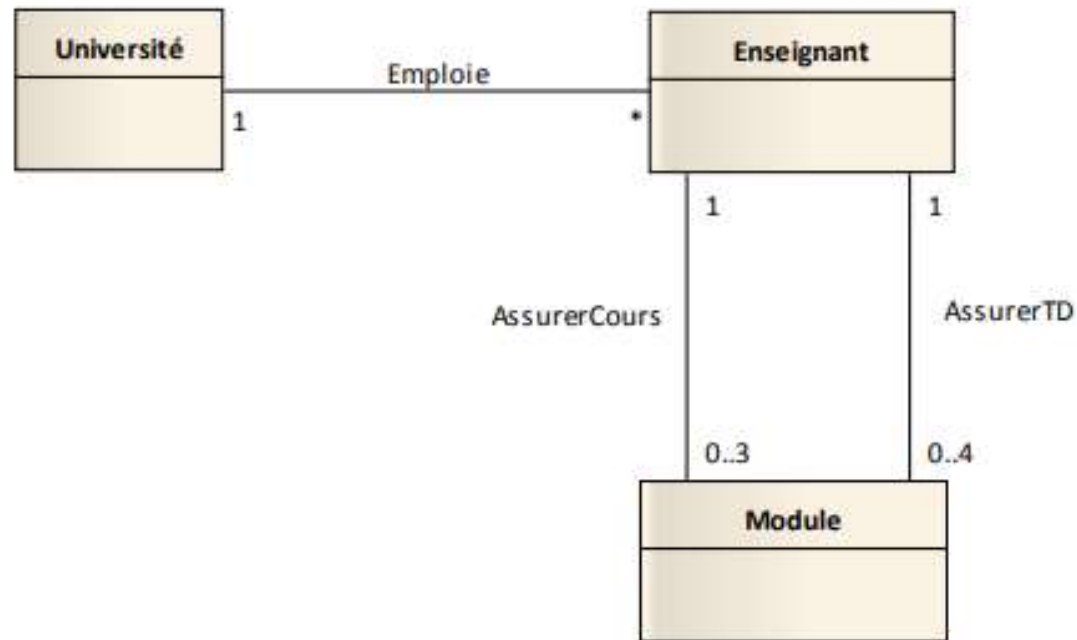
Association n-aires

- Une **association n-aire** lie plus de deux classes.
 - Notation avec un **losange central** pouvant éventuellement accueillir une classe-association.
 - La multiplicité de chaque classe s'applique à une instance du losange.



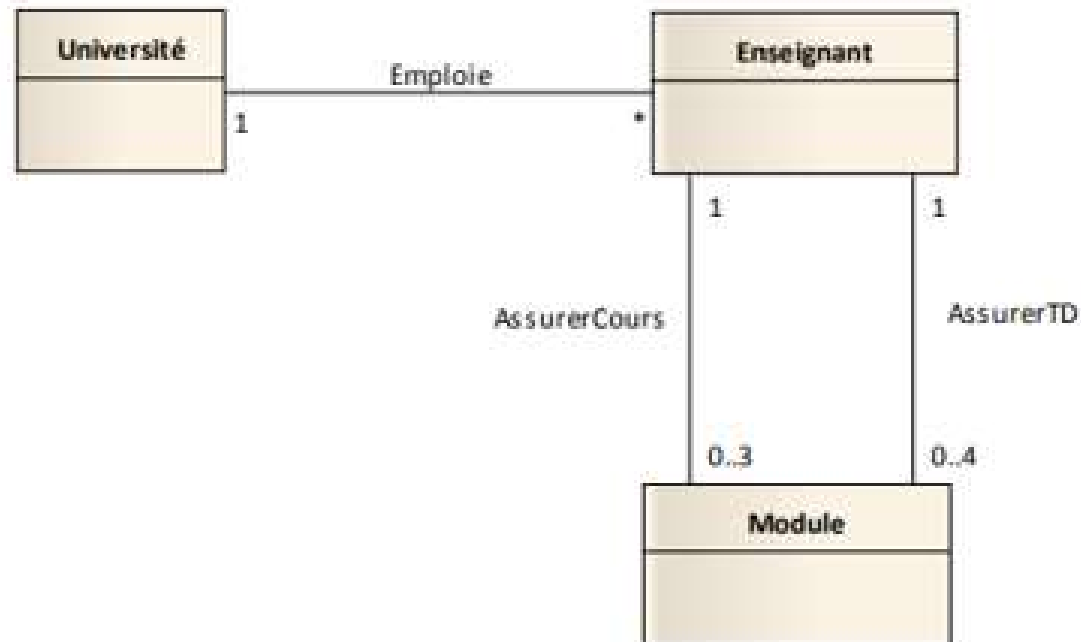
Association multiple

- Entre deux classes peut avoir plusieurs associations
- Dans cet exemple la classe *Enseignant* est reliée par deux associations avec la classe *Module*



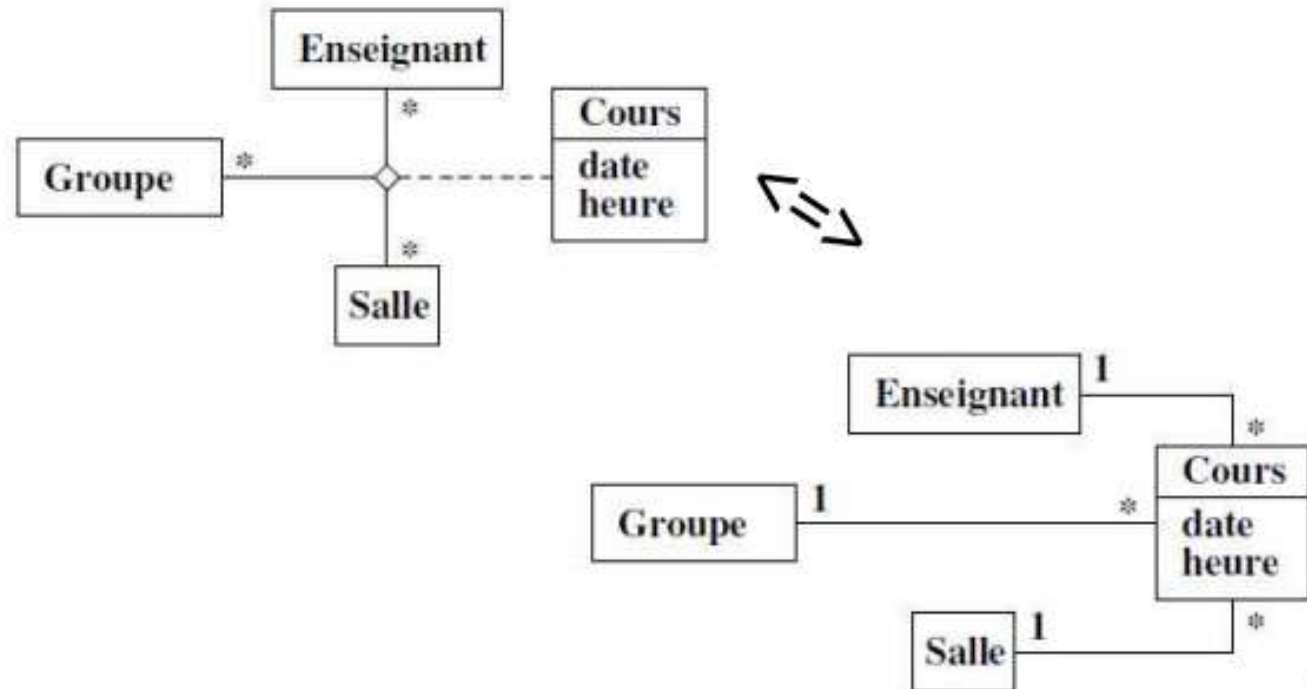
Association multiple

- Entre deux classes peut avoir plusieurs associations
- Dans cet exemple la classe *Enseignant* est reliée par deux associations avec la classe *Module*



Association n-aires

- Les associations **n-aires** sont peu fréquentes et concernent surtout les cas où les **multiplicités sont toutes ***. Dans la plupart des cas, on utilisera plus avantageusement des **classes-association** ou plusieurs relations binaires.



Association reflexive



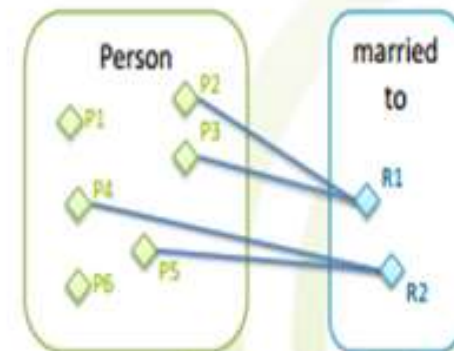
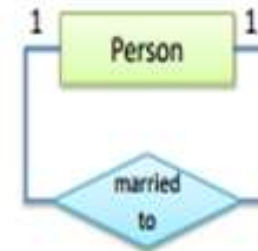
- Example:

- “Each person can only be married to one other person”



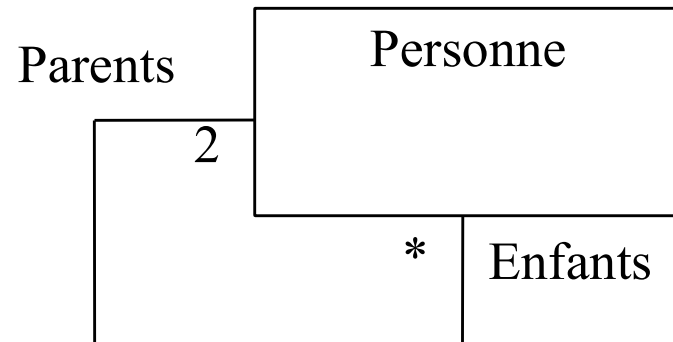
- Each entity can only appear in one instance of the “married to” entity set

- Note: following this model a person actually can marry him/herself



Association reflexive

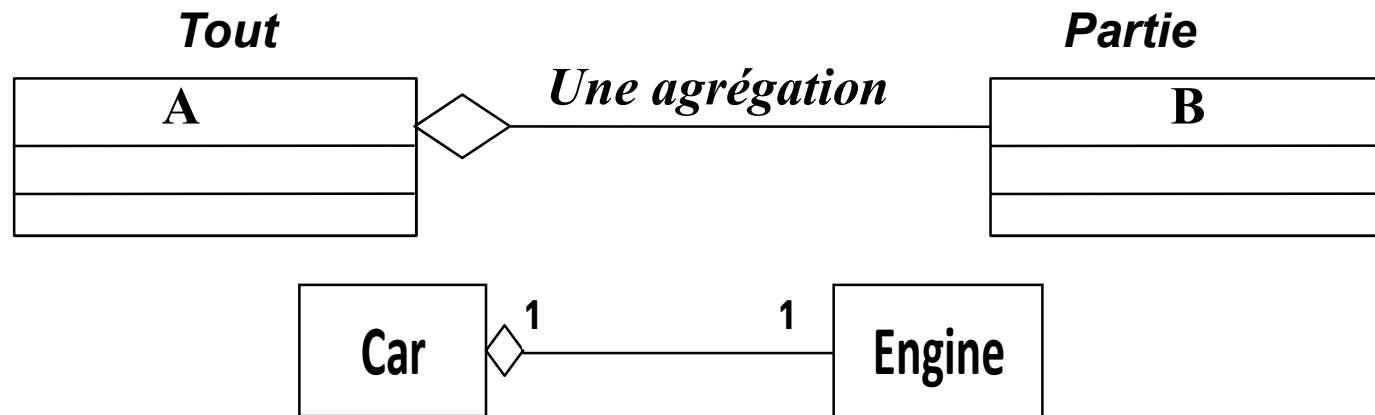
- *Association reflexive*



Le nommage des rôles est essentiel à la clarté du diagramme.

Relations entre classes : Les agrégations

- Représente une association *non symétrique* dans laquelle une classe joue un rôle prédominant par rapport à l'autre classe.
- Les critères suivants impliquent une agrégation:
 - une classe fait partie d'une autre classe;
 - les objets d'une classe sont subordonnés aux objets d'une autre classe;



La relation d'agrégation représente la relation '**Possède ou est parti de**'

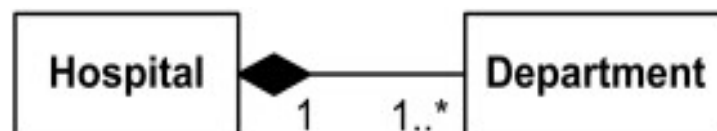
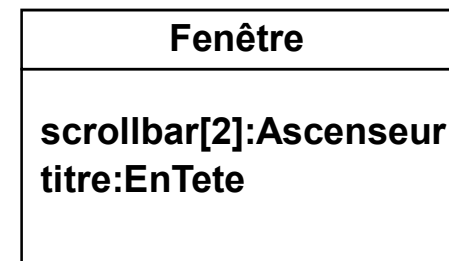
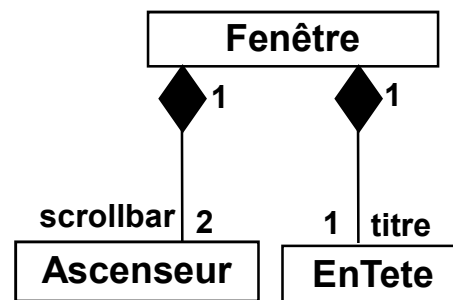
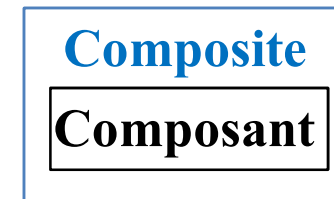
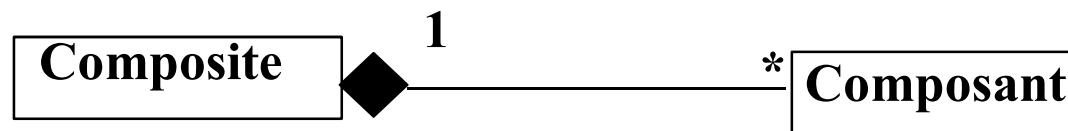
Agrégat : caractéristiques

- Transitivité
- Antisymétrie
- Propagation d'opérations
 - Certaines opérations du "tout" se propagent (avec modification) aux "parties"
 - Exemples : corriger document, corriger paragraphe, corriger mot
- Quand utiliser l'agrégat
 - Une "partie" ne peut pas exister indépendamment du "tout"



Les relations entre classes : La composition

- La composition est un cas particulier de l'agrégation
- Agrégation réalisée par valeur sur des attributs: composition.
- Ils sont physiquement contenus dans l'agrégat.



*Each Department belongs to exactly one Hospital.
If Hospital is closed, so are all of its Departments.*

L'association de composition

- La **composition** décrit une contenance structurelle entre instances. On utilise un **losange plein** côté de la **classe composite** (conteneur).
 - La **composition** est aussi dite **agrégation forte**.
 - La **destruction** ou la **copie** de la **classe composite** (l'ensemble) impliquent respectivement la **destruction** ou la **copie** de ses **composants** (les parties).
 - Une instance de la partie n'appartient jamais à plus d'une instance de l'élément composite.

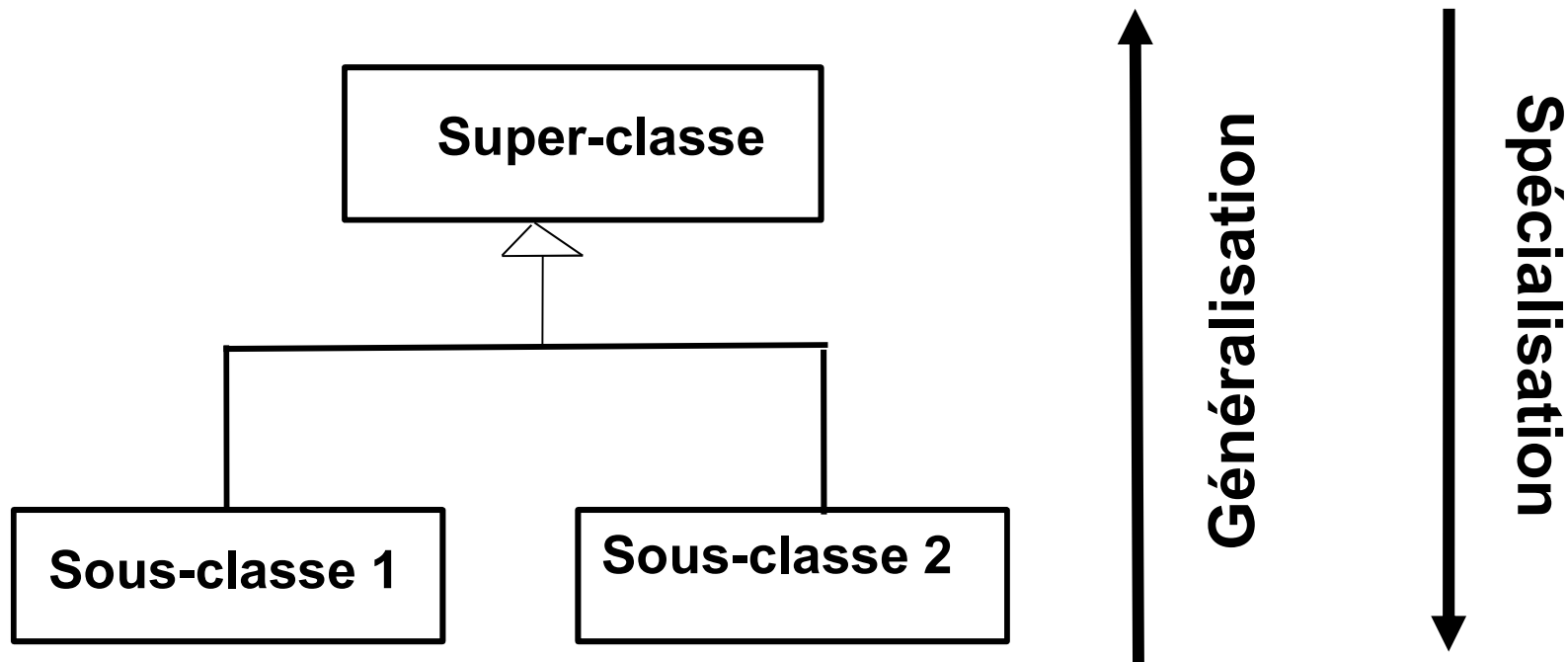


Les relations entre classes : La composition

- Forme d'agrégat avec appartenance forte et vies coïncidentes des parties au tout:
 - La destruction du composite implique la destruction de tous ses composants,
 - La création, la modification et la destruction des composants sont de la responsabilité du composite
- La multiplicité du côté du tout ne peut dépasser « 1 »
 - Un composant est non partagée

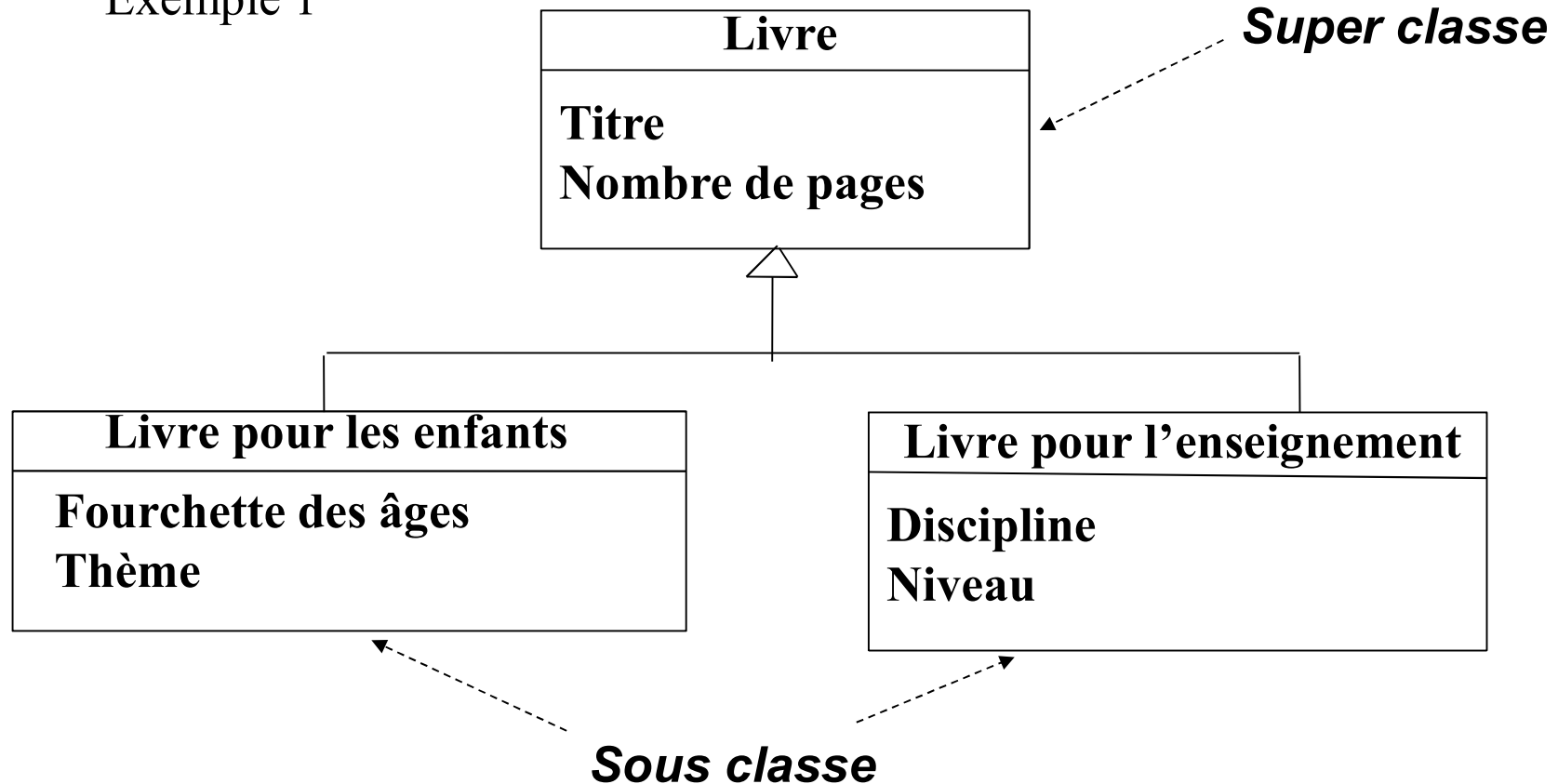
Les relations entre classes : généralisation/Spécialisation

- La généralisation est une relation entre classes qui exprime le lien sémantique “est-un”



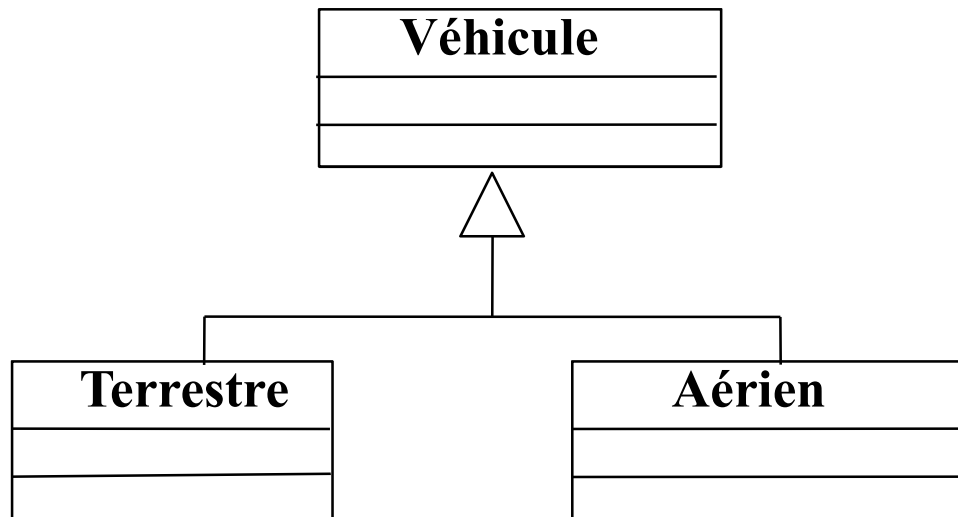
Les relations entre classes : généralisation/Spécialisation

Exemple 1



Les relations entre classes : généralisation/Spécialisation

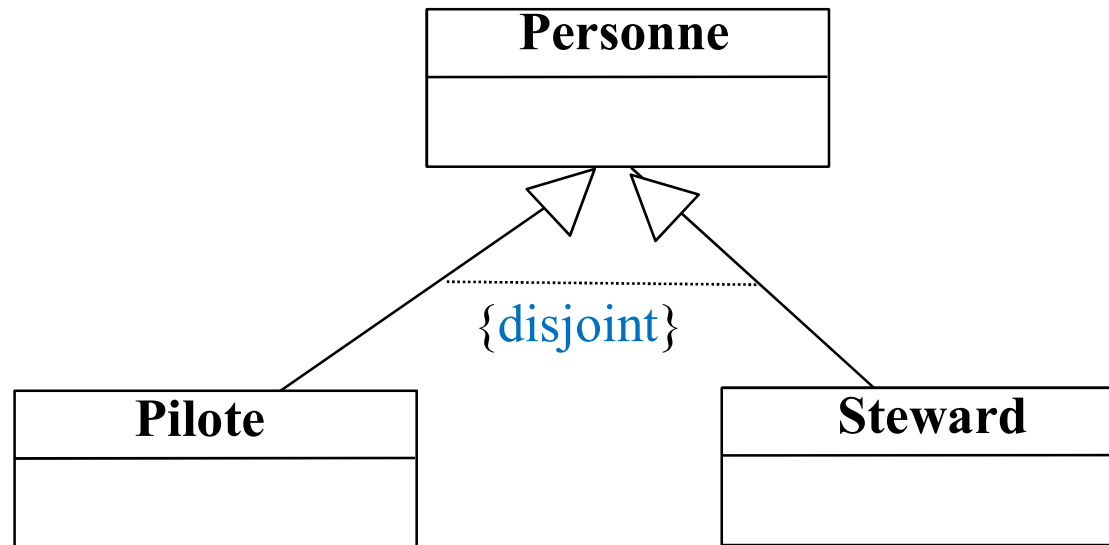
Les classes sont ordonnées selon une hiérarchie : une superclasse est une abstraction de ses sous-classes



- La généralisation est une relation non réflexive.
- La généralisation est une relation non symétrique.
- La généralisation est une relation transitive.

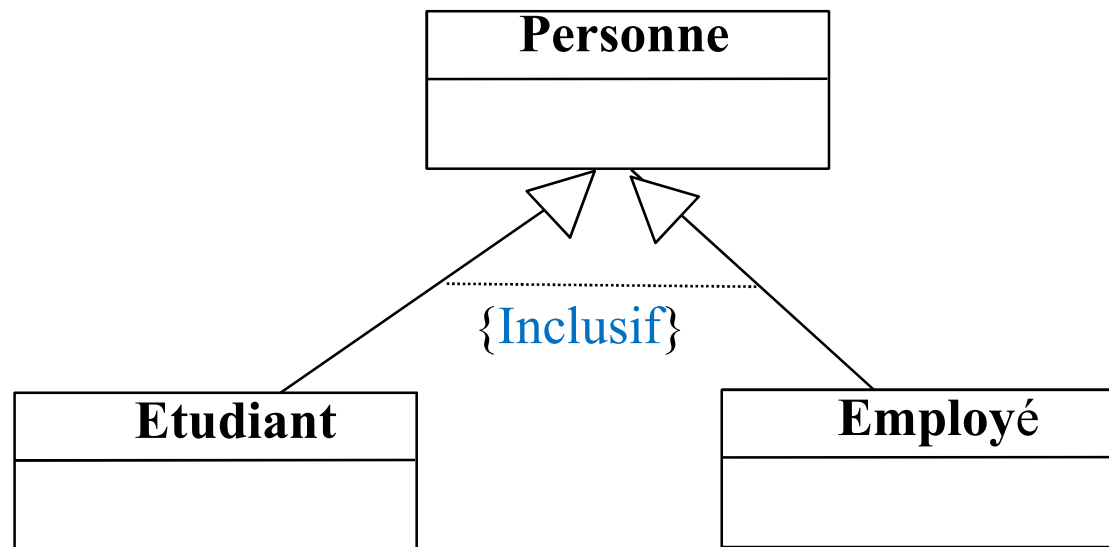
La généralisation: *Contraintes*

- La contrainte {Disjoint} ou {Exclusif} indique que les instances d'une sous-classe ne peuvent pas être incluses dans une autre sous classe de la même classe



La généralisation: *Contraintes*

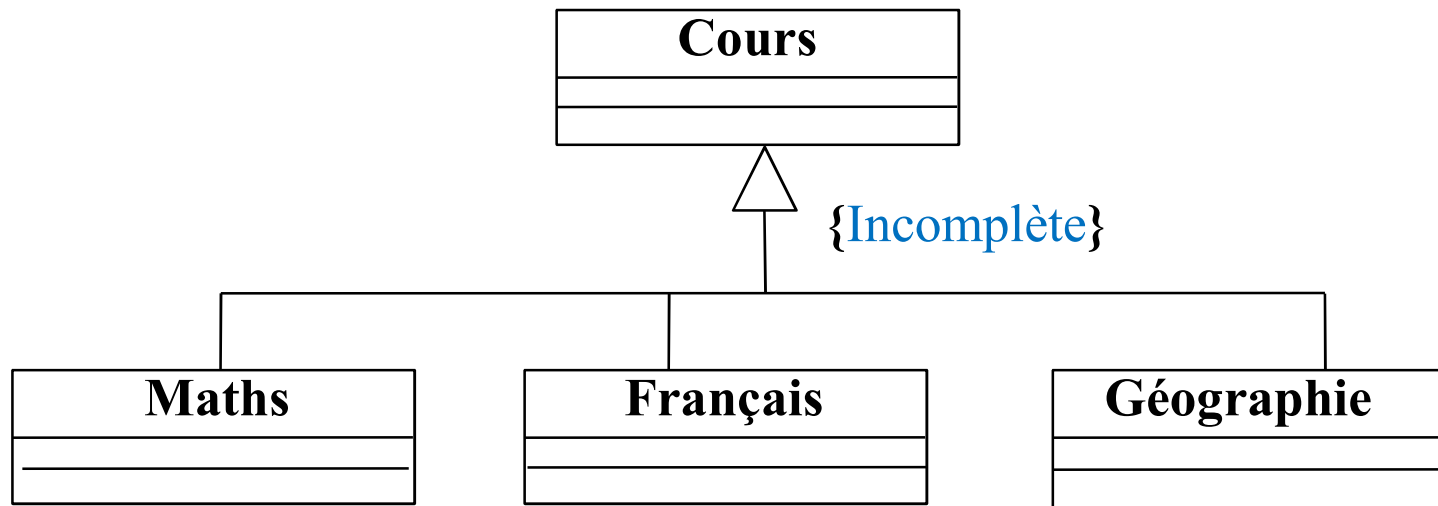
- La contrainte {Chevauchement} ou {Inclusif} indique que deux sous classes peuvent avoir des instances communes



inclusif ≠ exclusif

La généralisation: *Contraintes*

- La contrainte {Compleète} indique que la généralisation est terminée et qu'il n'est pas possible de rajouter des sous-classes. Inversement, la contrainte {Incomplète} désigne une généralisation extensible.



Exemple

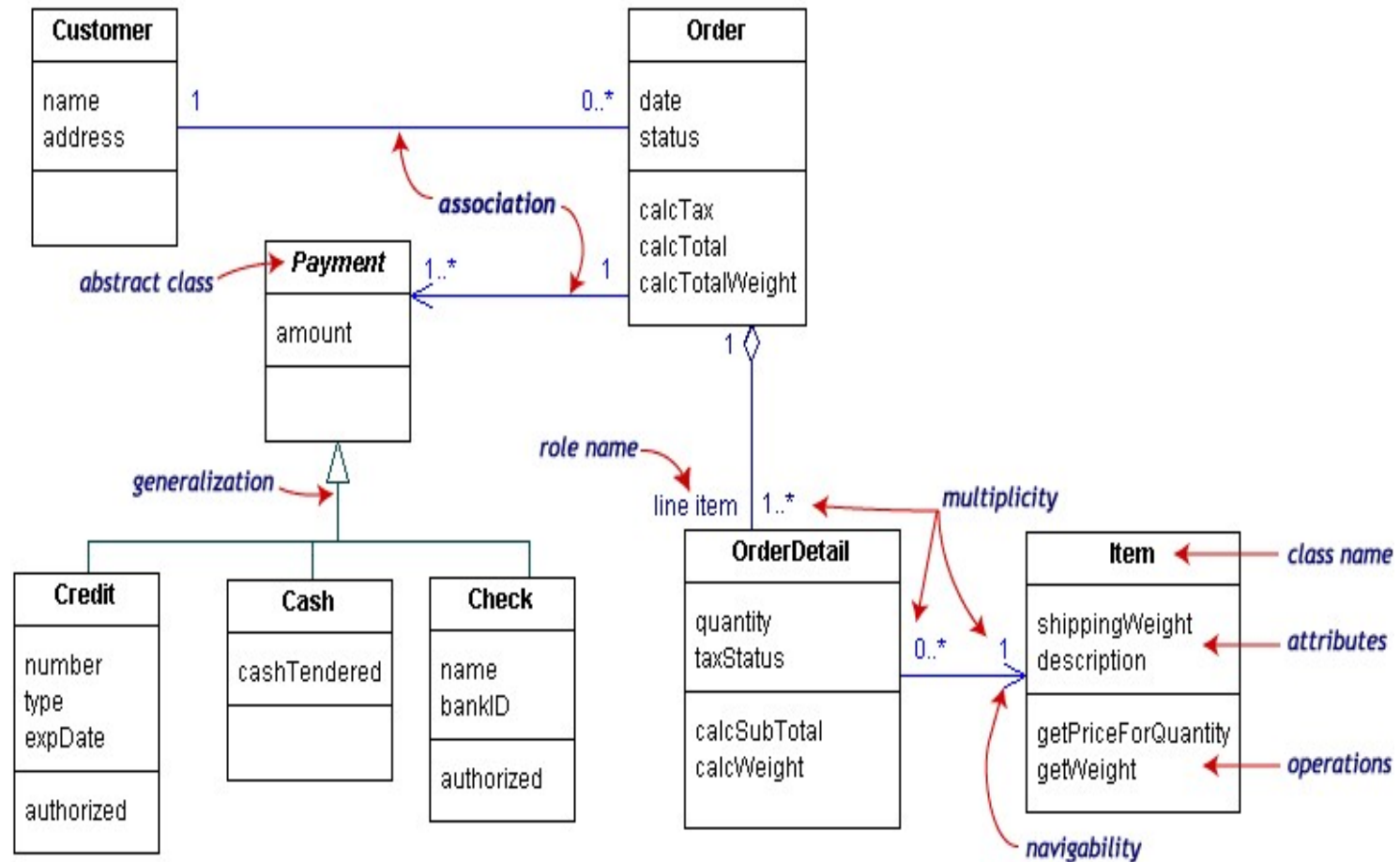


Diagramme d'objets

- Un diagramme sert à montrer un contexte. Ils facilitent la compréhension des structures de données complexes.

Nom de l'objet

Nom de l'objet : Classe

: Classe

Exemple

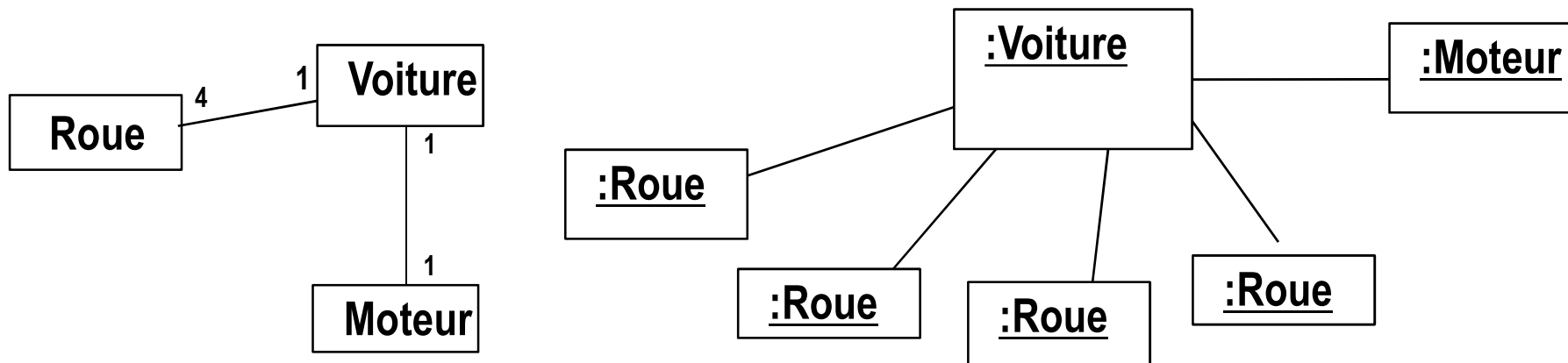


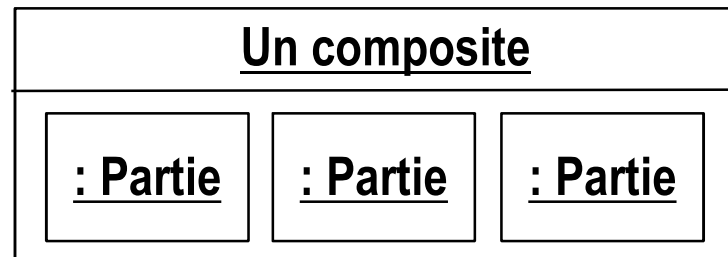
Diagramme de classes

Diagramme d'objets

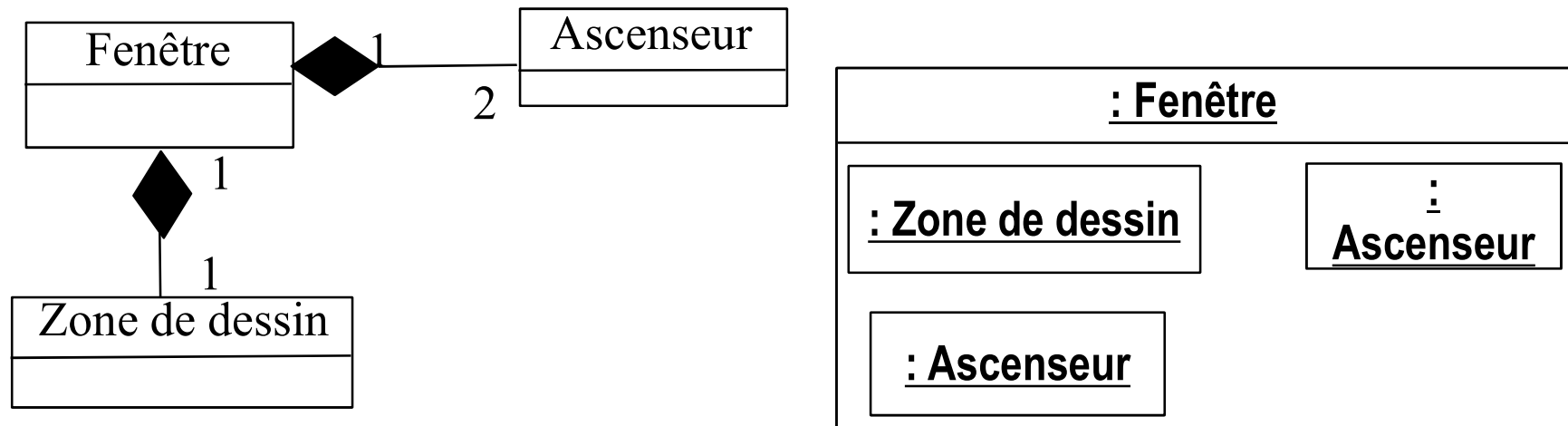
Une voiture a 4 roues et 1 moteur

Objet composite

- Les objets composés de sous-objets peuvent être représentés au moyen d'un objet composite.

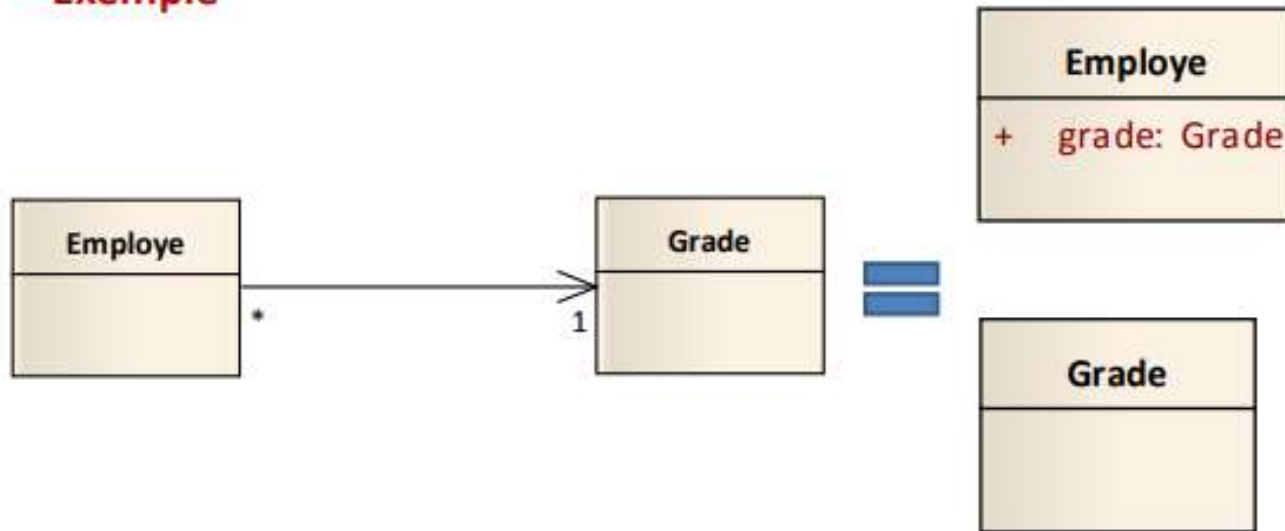


- Les objets composites sont instances de classes composites:



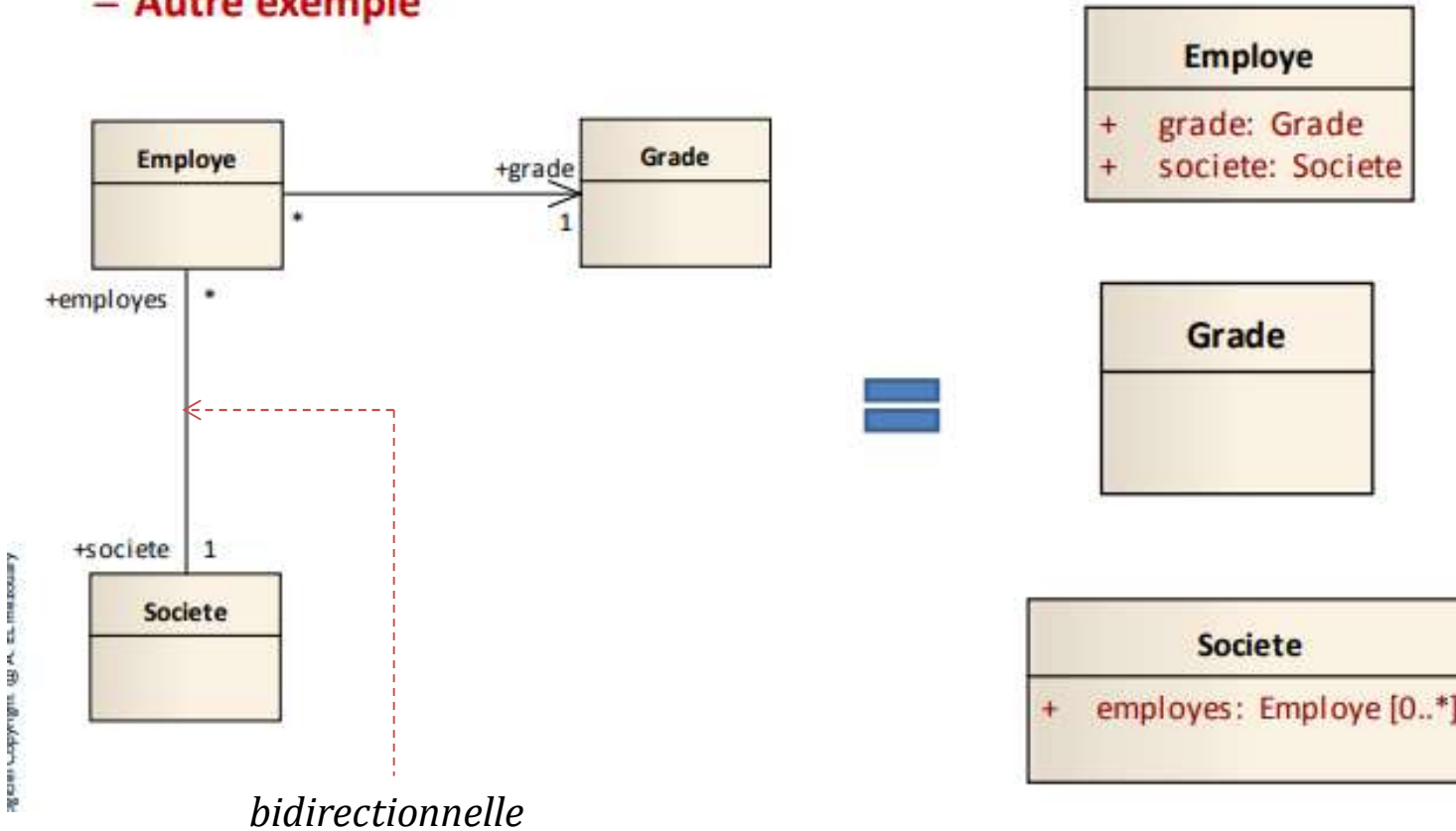
Association et attributs

- Les associations à multiplicité multiple peuvent être représentées par des tableaux ou des collections
- Lors de la génération de code, les associations sont générées en tant qu'attributs
- **Exemple**



Association et attributs

– Autre exemple



Implementation en Java

Classe avec attributs et opérations

A
+a1: String a2: String #a3: String -a4: String
+op1() +op2()

```
public class A {  
    public String a1;  
    package String a2;  
    protected String a3;  
    private String a4;  
    public void op1() {  
        ...  
    }  
    public void op2() {  
        ...  
    }  
}
```

Implementation en Java

Classe abstraite

A {abstract}

```
public abstract class A {  
    ...  
}
```

Interface

<<interface>>

A

```
public interface A {  
    ...  
}
```

Héritage simple



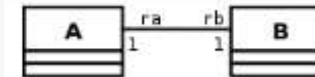
```
public class A {  
    ...  
}  
  
public class B extends A {  
    ...  
}
```

Implementation en Java

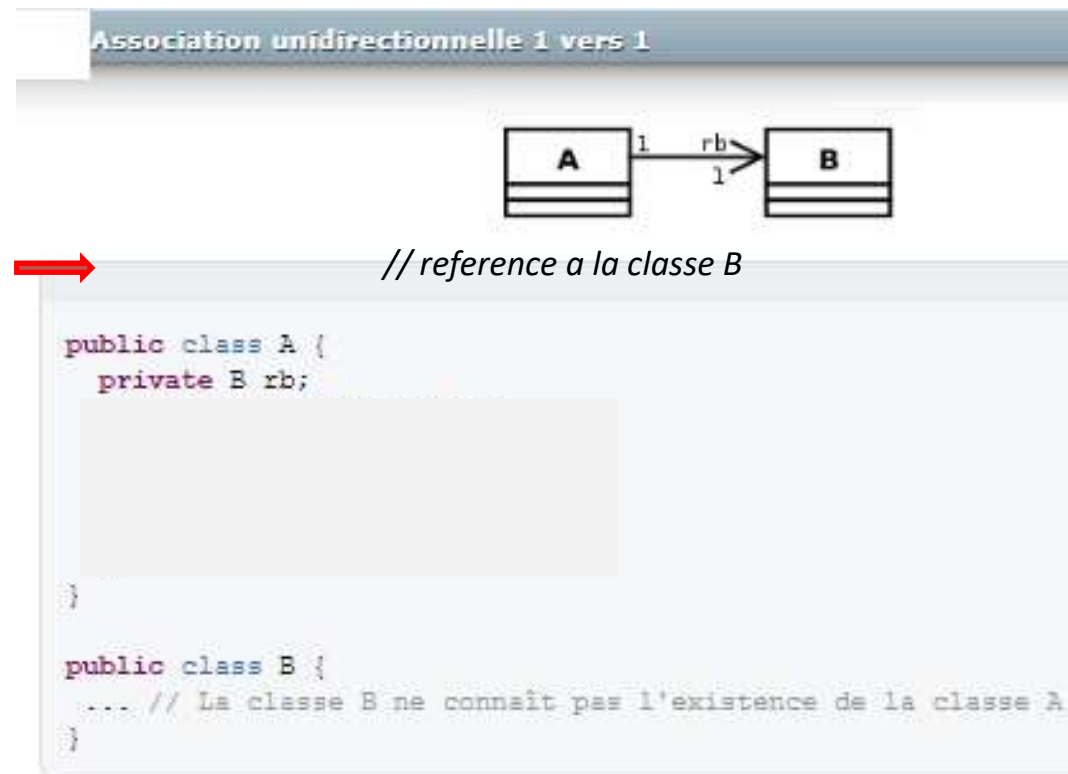
Association bidirectionnelle 1 vers 1

```
public class A {  
    private B rb; // reference a la classe B
```

```
public class B {  
    private A ra; // reference a la classe A
```

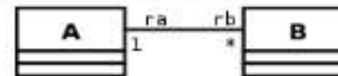


Implementation en Java



Implementation en Java

Association bidirectionnelle 1 vers N

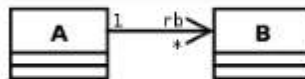


```
public class A {  
    private ArrayList <B> rb;  
}
```

```
public class B {  
    private A ra;  
}
```

Implementation en Java

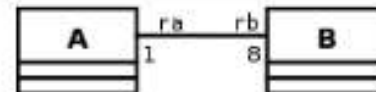
Association unidirectionnelle 1 vers plusieurs



```
public class A {
    private ArrayList <B> rb;
}

public class B {
    ... // B ne connaît pas l'existence de A
}
```

Association 1 vers N



Dans ce cas, il faut utiliser un tableau plutôt qu'un vecteur. La dimension du tableau étant donnée par la cardinalité de la terminaison d'association.

