

Les diagrammes de classes

UML - LP IDSE
IUT Nice Sophia Antipolis

Site web du module :
<https://mbf-iut.i3s.unice.fr/>

Tiré du cours de Mireille Blay-Fornarino

Détermination des concepts du domaine



Détermination des concepts du domaine



Helpful
Tips

La détermination des concepts s'effectue sur la base des cas d'utilisation par simple **analyse grammaticale** de la description textuelle.

D'une manière générale,

Détermination des concepts du domaine



Helpful
Tips

La détermination des concepts s'effectue sur la base des cas d'utilisation par simple **analyse grammaticale** de la description textuelle.

D'une manière générale,
les **noms** représentent des **concepts** ou des **attributs**
tandis

Détermination des concepts du domaine



La détermination des concepts s'effectue sur la base des cas d'utilisation par simple **analyse grammaticale** de la description textuelle.

D'une manière générale,
les **noms** représentent des **concepts** ou des **attributs**
tandis
que les **verbes** représentent des **comportements**
(opérations, méthodes)

Deux règles utiles

Règle du cartographe : Le modèle du domaine se construit de la même façon qu'un cartographe dessine une carte :

- ✓ En utilisant le **vocabulaire** du domaine étudié.
 - ✓ En excluant les éléments **non pertinents**.
 - ✓ En n'incluant pas d'éléments **inexistants** dans le domaine.
- **Choix entre concept et attribut** : Si un élément du domaine étudié est autre chose qu'un nombre ou un simple texte, alors il s'agit probablement d'un **concept** et non d'un **attribut**.

Deux règles utiles



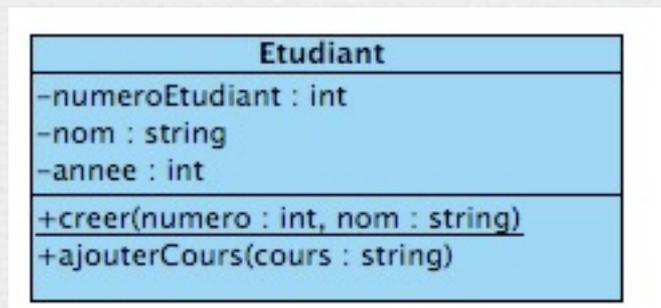
Règle du cartographe : Le modèle du domaine se construit de la même façon qu'un cartographe dessine une carte :

- ✓ En utilisant le **vocabulaire** du domaine étudié.
 - ✓ En excluant les éléments **non pertinents**.
 - ✓ En n'incluant pas d'éléments **inexistants** dans le domaine.
- **Choix entre concept et attribut** : Si un élément du domaine étudié est autre chose qu'un nombre ou un simple texte, alors il s'agit probablement d'un **concept** et non d'un **attribut**.



Classes

- Une classe est une **collection (*modèle*)** d'objets avec une structure commune, un comportement commun, des relations identiques et une sémantique identique
- On **identifie** les classes en recherchant les *concepts* du domaine et en examinant les *objets* dans les diagrammes
- La **représentation graphique** d'une classe consiste en un rectangle avec 3 compartiments
- Les **noms des classes** devraient être choisis dans le vocabulaire du domaine
 - il est bon d'établir des standards pour les nom
 - i.e., toutes les classes sont des noms communs commençant par une majuscule



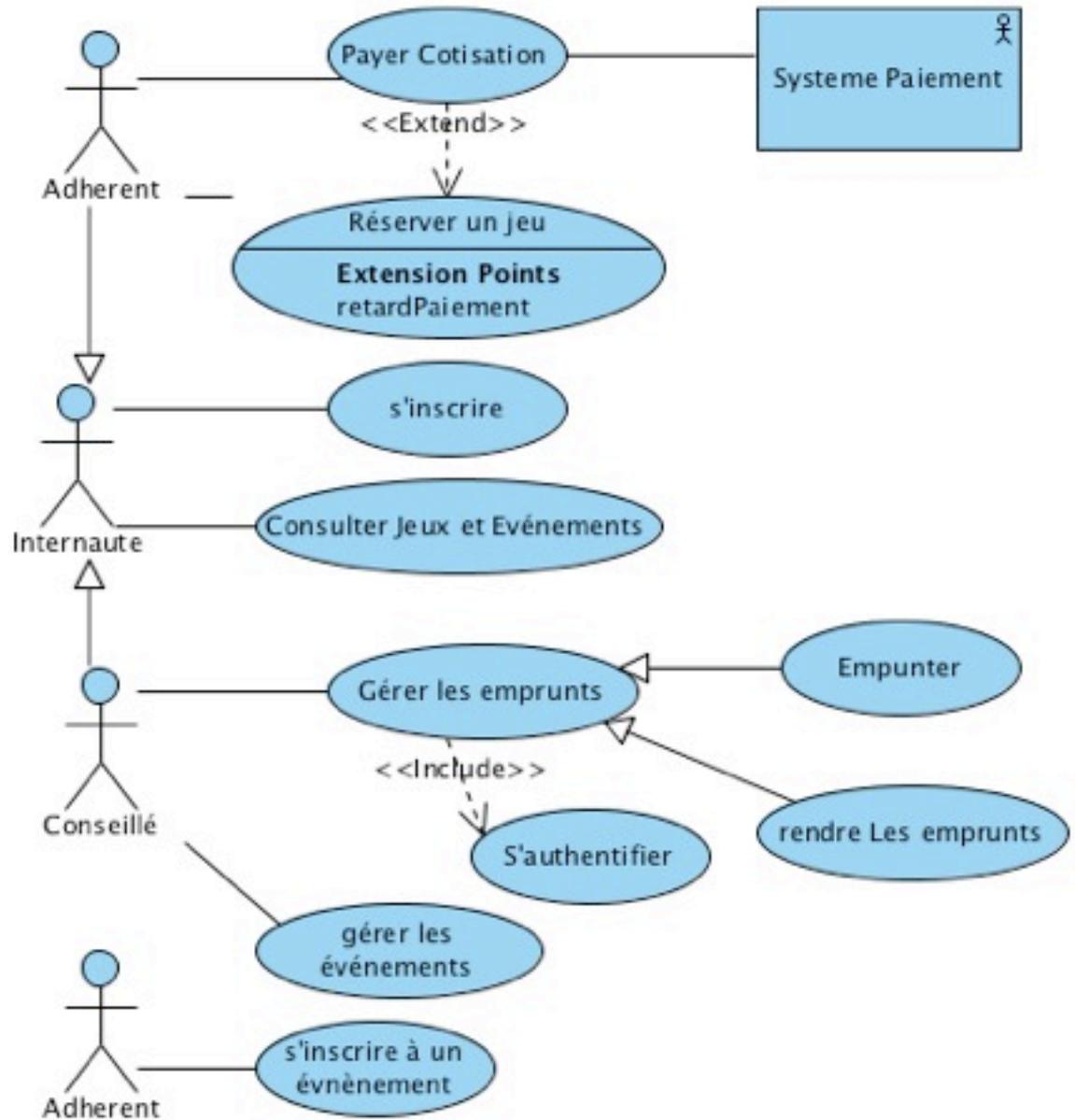
Concepts du domaine

Par l'exemple

UML au travail : Une ludothèque

- (1) Nous voulons informatiser une ludothèque pour favoriser la consultation des jeux proposés par la ludothèque.
- (2) Les adhérents peuvent emprunter des jeux en s'adressant à un conseiller qui enregistre l'emprunt.
- (3) Les jeux empruntés sont rendus à un conseiller....
- (4) Un adhérent peut réserver des jeux. Une réservation précise l'emprunteur, le jeu et la date de la demande de réservation. L'adhérent est averti quand le jeu revient en rayon.
- (5) Pour organiser un événement le conseiller spécialisé doit alors donner les informations suivantes : les jeux à tester, le nombre maximal et minimal de participants attendus, la date, et l'heure de début de l'événement.
- (6) Un adhérent peut s'inscrire pour participer à un événement à condition qu'il y ait encore de la place.
- (7) Un adhérent peut payer sa cotisation en ligne par un système de paiement externe

Ludothèque



Description d'un flot associé à un cas d'utilisation

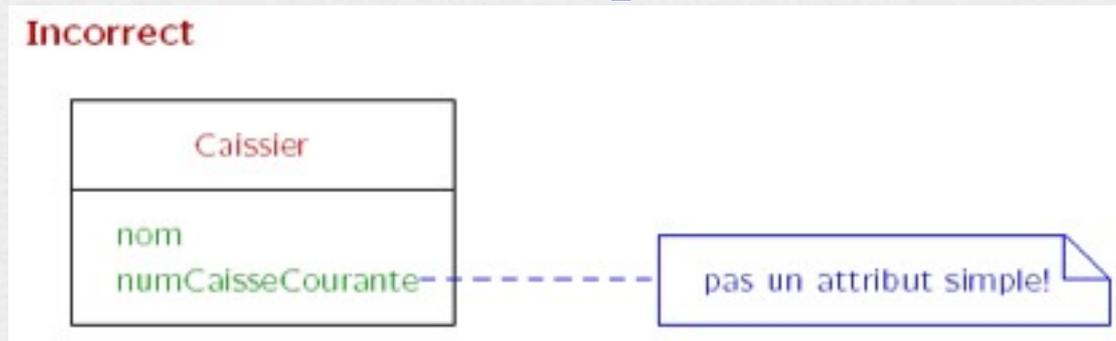
Un conseiller enregistre l'emprunt d'un jeu pour un adhérent

- 1) Le conseiller s'authentifie;
- 2) Le conseiller saisit l'identifiant du jeu et de l'adhérent
- 3) Le système vérifie la disponibilité du jeu
- 4) Le système vérifie que la cotisation est bien payée
- 5) Le système vérifie que l'adhérent n'a pas de pénalité impayée
- 6) Le système enregistre l'emprunt.
- 7) Le système signale que l'emprunt est valide.

Notes

- Un attribut ne peut représenter qu'une **valeur primitive** (entier, texte, date, identificateur, matricule, ...).
- Un attribut ne peut représenter que des données **relatives au concept auquel il est associé**.

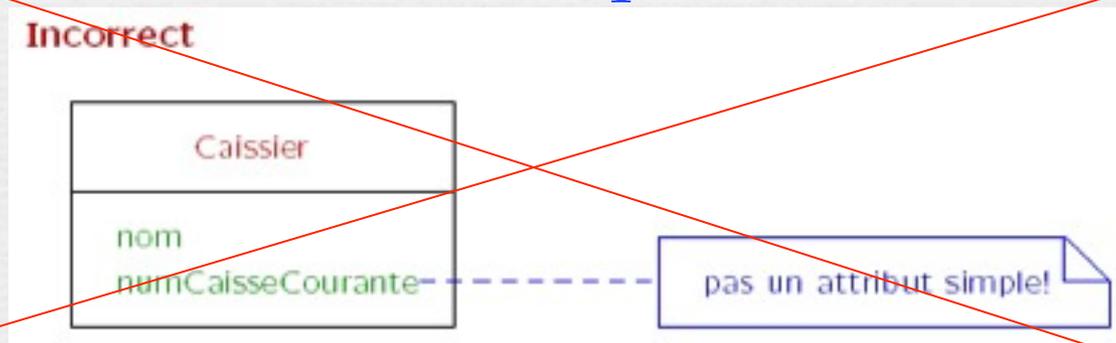
Exemple :



Notes

- Un attribut ne peut représenter qu'une **valeur primitive** (entier, texte, date, identificateur, matricule, ...).
- Un attribut ne peut représenter que des données **relatives au concept auquel il est associé**.

Exemple :

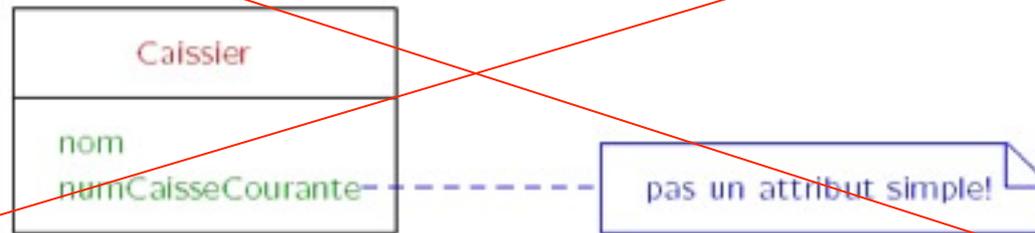


Notes

- Un attribut ne peut représenter qu'une **valeur primitive** (entier, texte, date, identificateur, matricule, ...).
- Un attribut ne peut représenter que des données **relatives au concept auquel il est associé**.

Exemple :

Incorrect



Correct

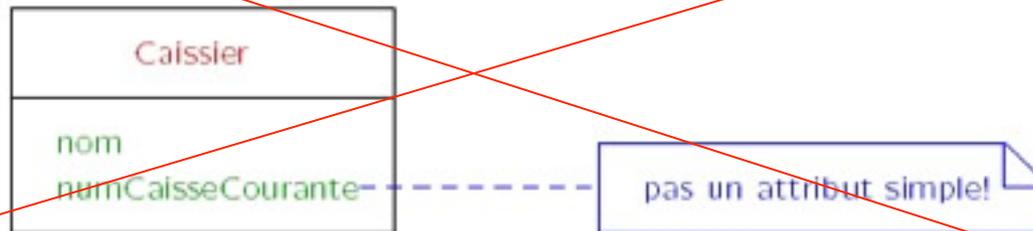


Notes

- Un attribut ne peut représenter qu'une **valeur primitive** (entier, texte, date, identificateur, matricule, ...).
- Un attribut ne peut représenter que des données **relatives au concept auquel il est associé**.

Exemple :

Incorrect



Correct

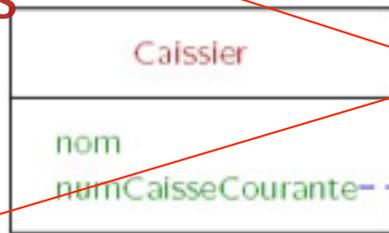


Notes

- Un attribut ne peut représenter qu'une **valeur primitive** (entier, texte, date, identificateur, matricule, ...).
- Un attribut ne peut représenter que des données **relatives au concept auquel il est associé**.

Exemple :

Incorrect



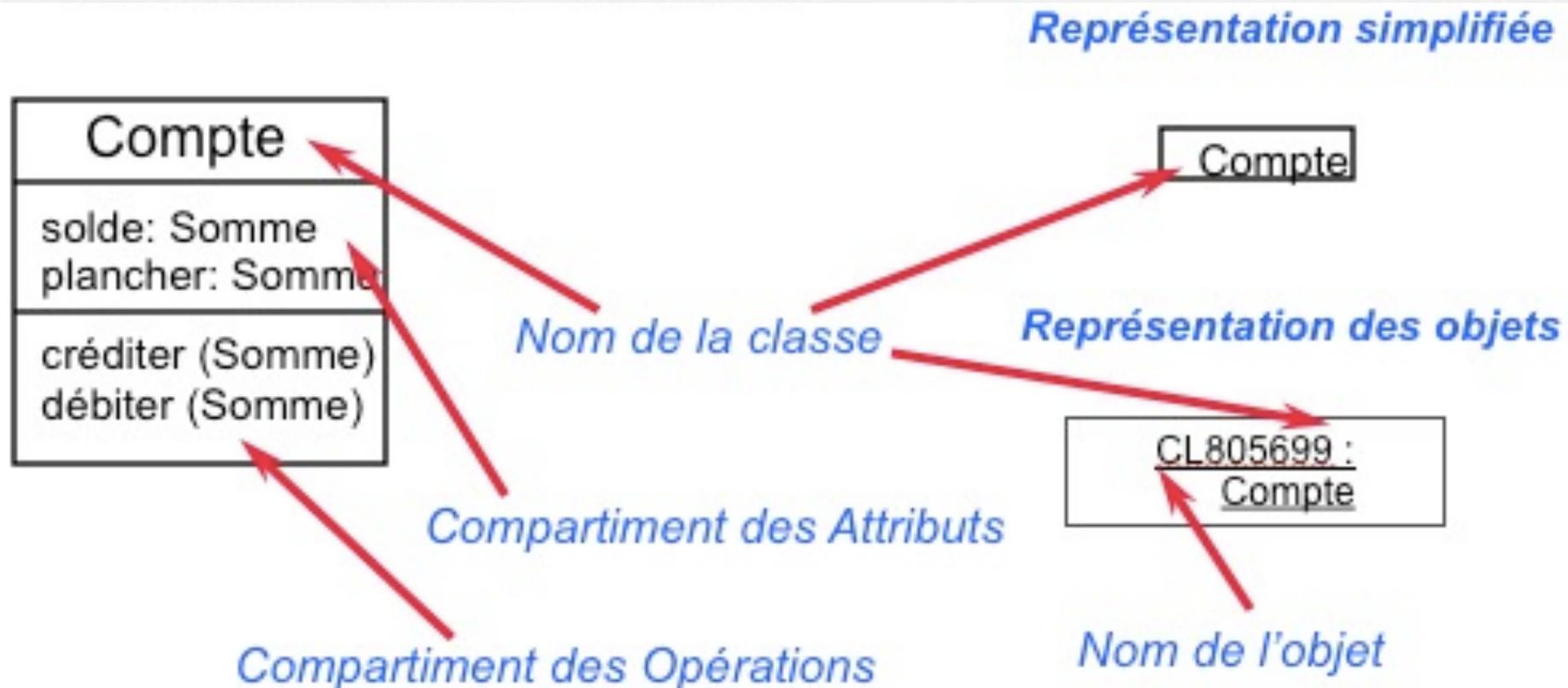
pas un attribut simple!

Correct



Pas d'attributs complexes!!

Notations UML pour classes et objets



Relations entre classes

Explicitation des notations

«Relations»,
associations,
agrégation,
généralisation,
compléments

Relations

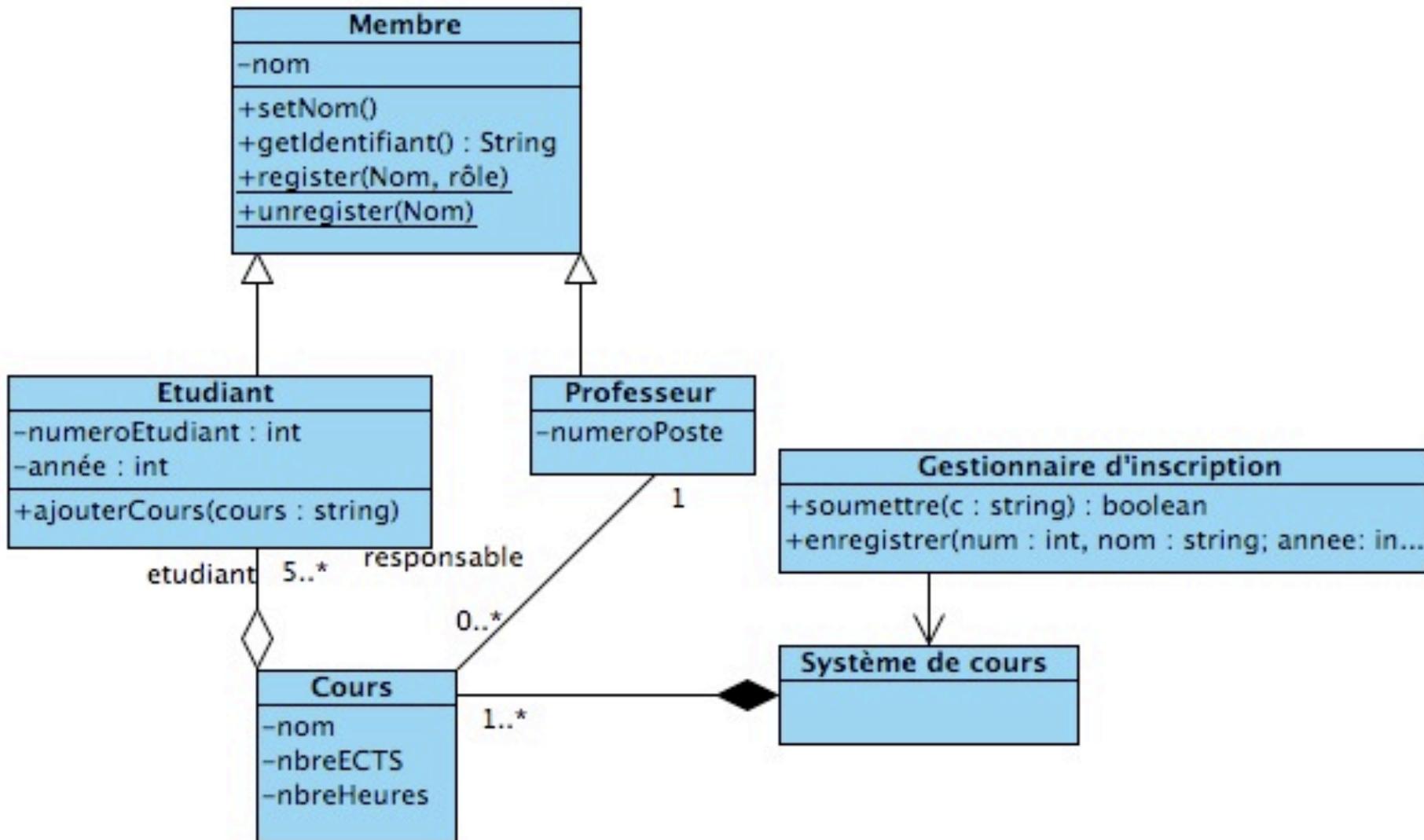
Les relations fournissent un chemin de communication entre objets - si deux objets ont besoin de se parler, il doit exister un lien entre eux

Trois types de relations :

- Association
- Agrégation
- Dépendance*

Si vous hésitez, utiliser une association !

Relations



Associations

Nommage
Rôle
Multiplicité
Navigation

- Les associations peuvent avoir des étiquettes:
 - Il s'agit du **nom de l'association**.
- Les associations peuvent avoir des noms de **rôle**:
 - un nom de rôle identifie le rôle ou la responsabilité de l'objet dans l'association.
- Les associations peuvent indiquer la **navigation** avec une pointe de flèche ouverte:
 - Pas de flèche => bidirectionnelle
 - La plupart des associations sont unidirectionnelles en fin de conception.
- Les associations peuvent indiquer une **multiplicité**.

Relations

- Une **association** est une connexion entre classes
 - Une association est représentée par une ligne connectant les classes
- Une **agrégation** est une relation plus forte et s'établit entre un tout et ses parties
 - Une agrégation est représentée par une ligne connectant les classes avec un losange du côté de la classe représentant le tout
- Une **dépendance** est une relation faible établie entre un client et un fournisseur et où le client n'a pas de connaissance sémantique sur le fournisseur
 - Une dépendance est représentée par une flèche en pointillés allant du client au fournisseur

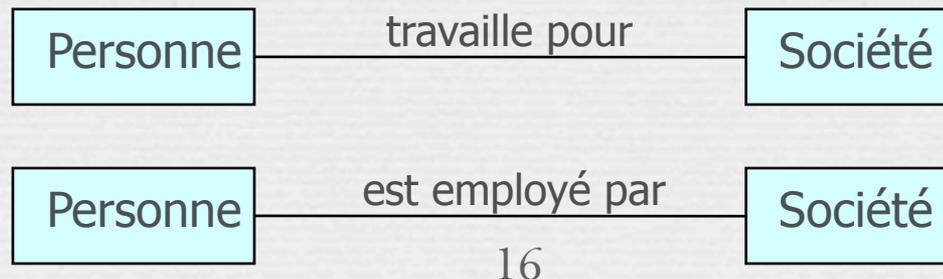
Nommage des associations

- Une association peut être nommée afin de faciliter la compréhension des modèles. Dans ce cas le nom est indiqué au milieu du lien symbolisant l'association



Nommage
Rôle
Multiplicité
Navigation

- L'usage recommande de choisir comme nom d'une association une forme verbale active (exemple : travaille pour) ou passive (exemple : est employé par)

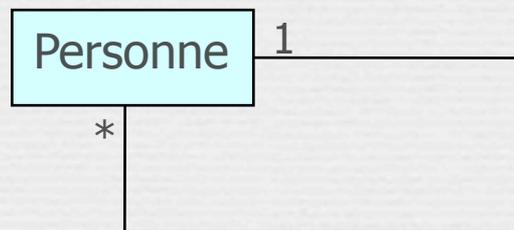


Nommage des associations...

Par défaut le sens de lecture du nom d'une association est de gauche à droite

Dans le cas où la lecture du nom est ambiguë, on peut ajouter l'un des signes < ou > pour indiquer le sens de lecture

- Exemples**

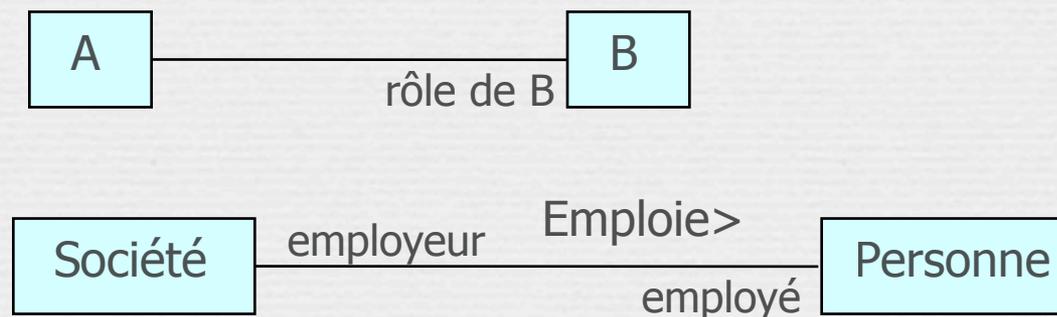


< est père de

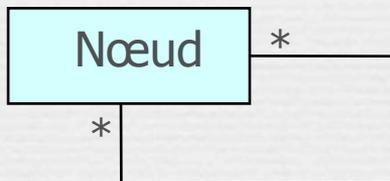
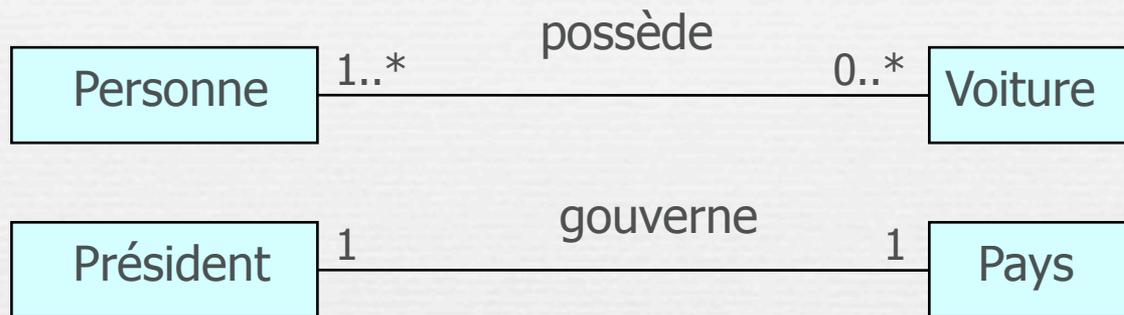
Rôles des extrémités d'association

- On peut attribuer à une extrémité d'une association un nom appelé rôle qui décrit comment une classe source voit une classe destination au travers de l'association
- Le rôle est placé près de la fin de l'association et à côté de la classe à laquelle il est appliqué
- L'utilisation des rôles est optionnelle
- Représentation et exemple

Nommage
Rôle
Multiplicité
Navigation



Multiplicité : exemple



Association réflexive

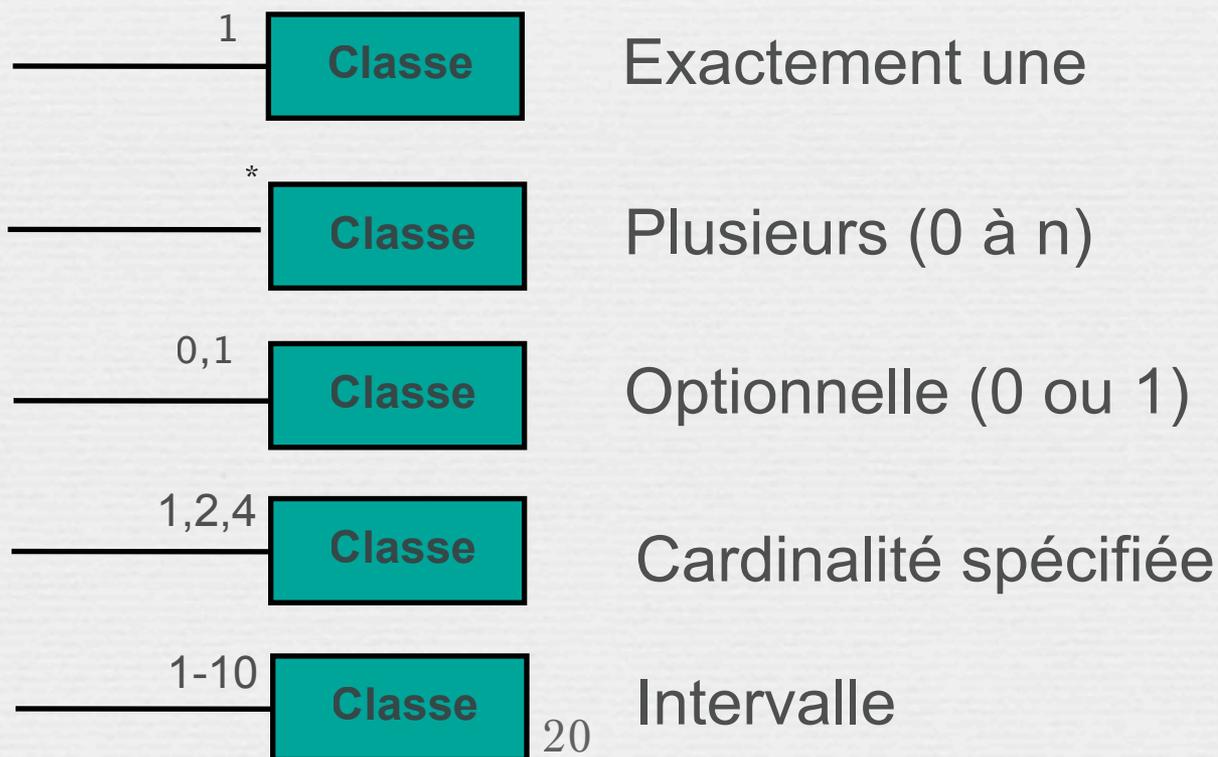
Un réseau informatique est composé de nœuds inter-connectés

Nommage
Rôle
Multiplicité
Navigation

Multiplicité

La multiplicité est définie par le nombre d'objets qui participent à une relation

- La multiplicité est le nombre d'instances d'une classe reliées à UNE instance d'une autre classe
- Pour chaque association et agrégation, il y a deux multiplicités : une à chaque bout de la relation

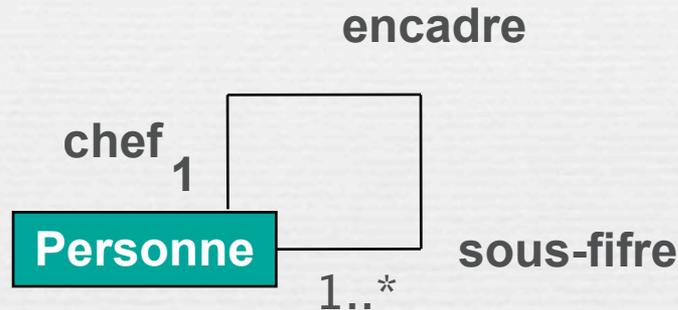


20

Cas particuliers de relations

■ Relations réflexives

Aie aie...
une
personne a
toujours un
chef... elle
est le chef
d'elle-
même?

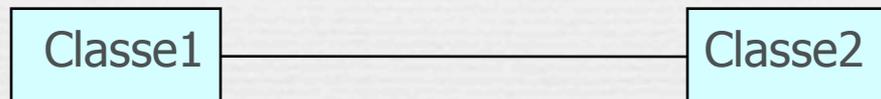


Une relation réflexive lie
des objets de même classe

Navigation

Bien que les associations soit bi-directionnelles par défaut, il peut être bon de limiter la navigation à un seul sens

- Les objets de Classe2 sont accessibles à partir de ceux de Classe1 et vice-versa



Si la navigation est restreinte, une flèche indique le sens de navigation

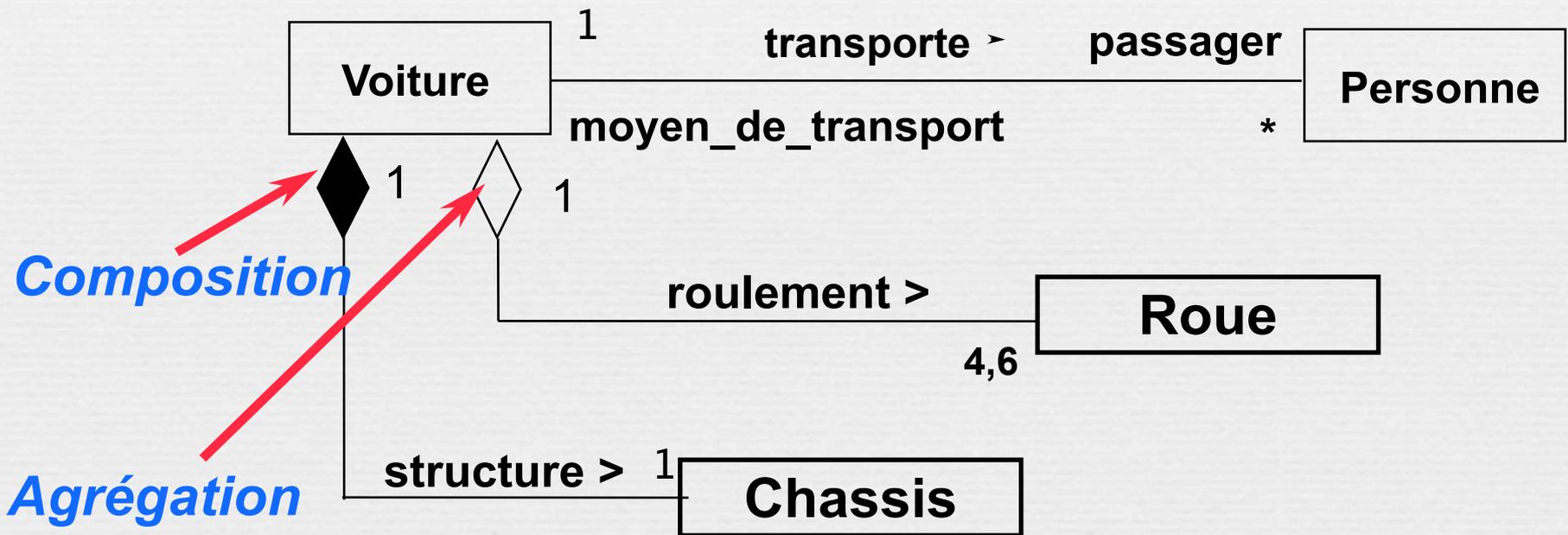
- Les objets de la Classe 1 sont accessibles à la classe 2



Nommage
Rôle
Multiplicité
Navigation

Composition et Agrégation

- Cas particuliers de relations :
 - Notion de *tout* et *parties*



Agrégation

L'agrégation représente une association de type ensemble/élément

L'agrégation ne concerne qu'un seul rôle d'une association

Représentation

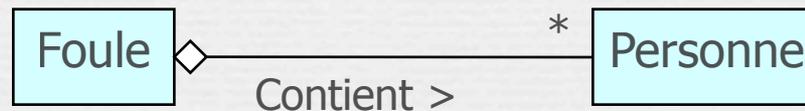


L'agrégation permet de modéliser une contrainte d'intégrité et de désigner l'agrégat comme gérant de cette contrainte

Agrégation...

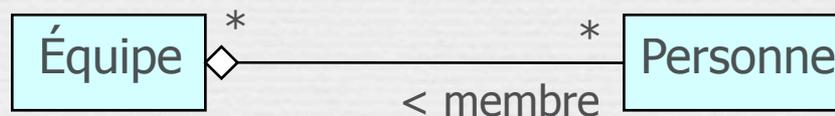
Exemple 1

- Une personne est dans une foule
- Une foule contient plusieurs personnes



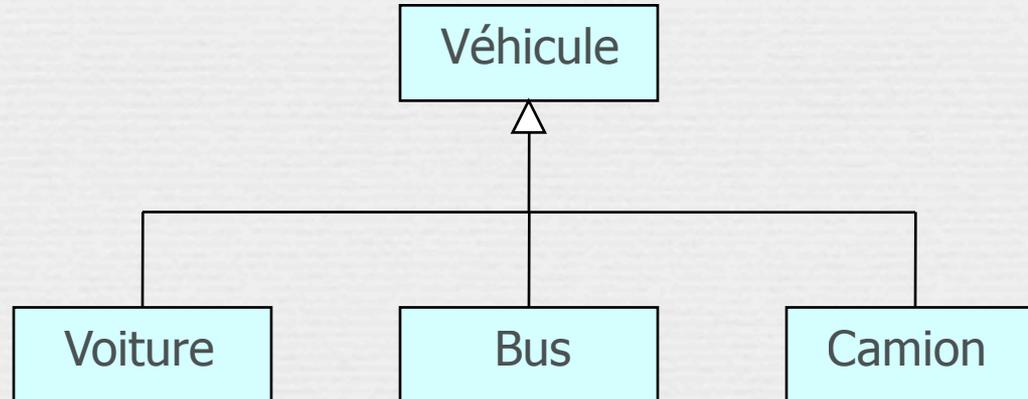
Exemple 2 (Agrégation partagée)

- Une personne fait partie de plusieurs équipes
- Une équipe contient plusieurs personnes



Généralisation

- C'est une relation de classification entre un élément général et un élément plus spécifique
- L'élément le plus spécifique est cohérent avec l'élément le plus général et contient plus d'informations
- Exemple

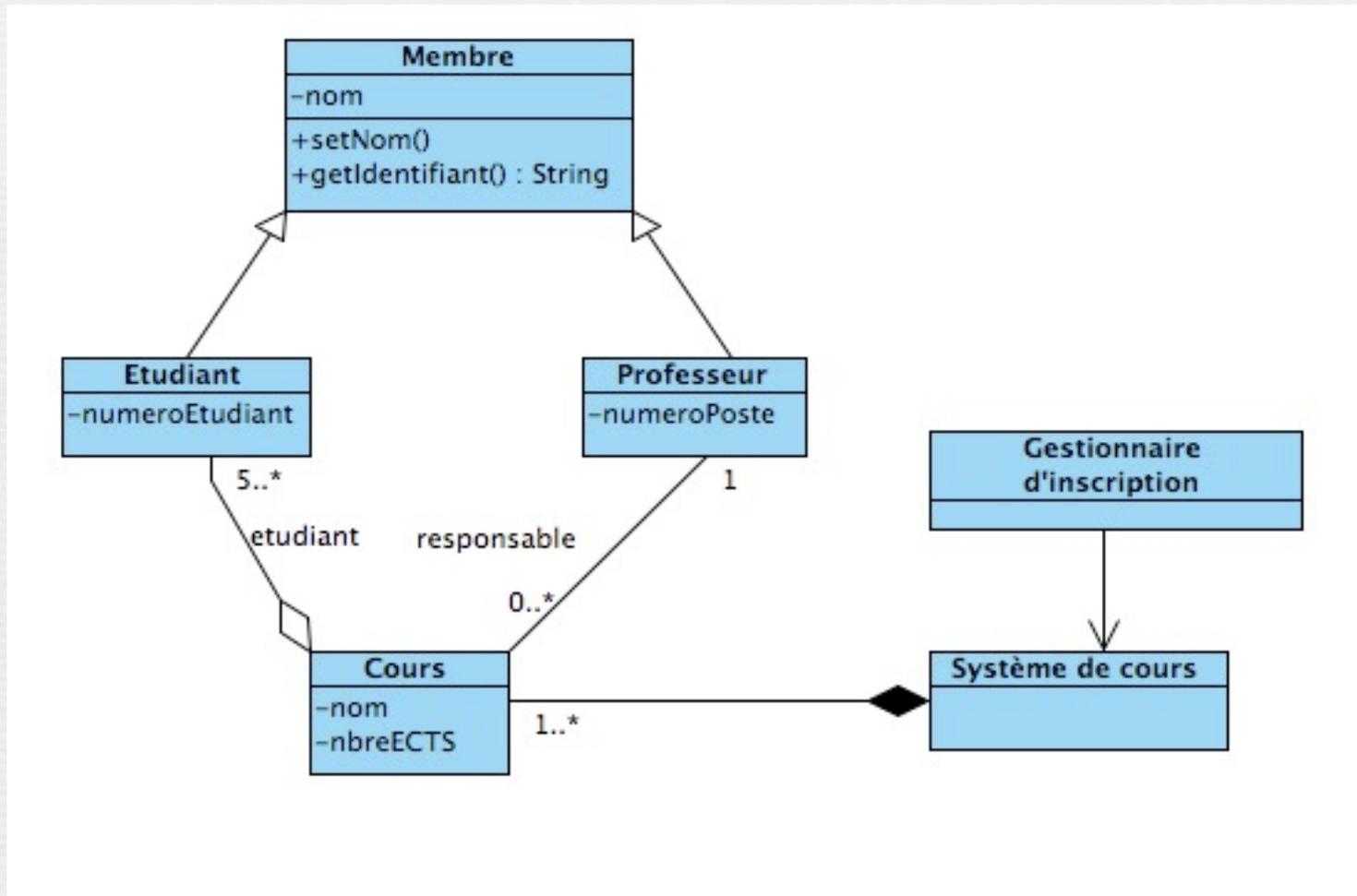


Héritage

- L'héritage est une relation entre une super-classe (classe de base) et ses sous-classes (classes dérivées)
- Deux manières d'identifier une relation d'héritage :
 - Généralisation
 - Spécialisation
- Les éléments communs (attributs, comportements, relations) sont reportés au niveau le plus haut de la hiérarchie

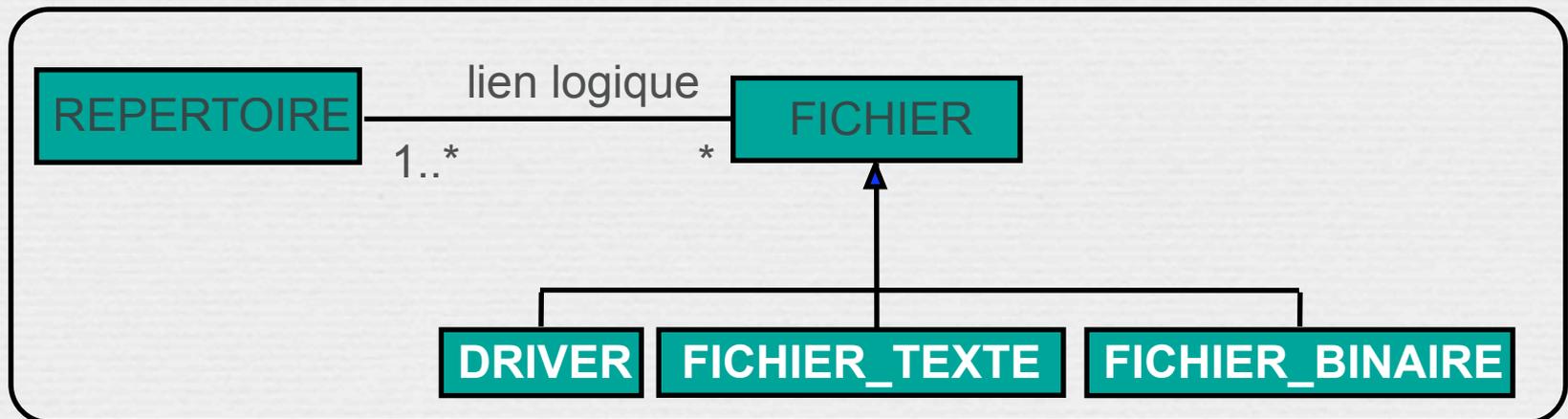
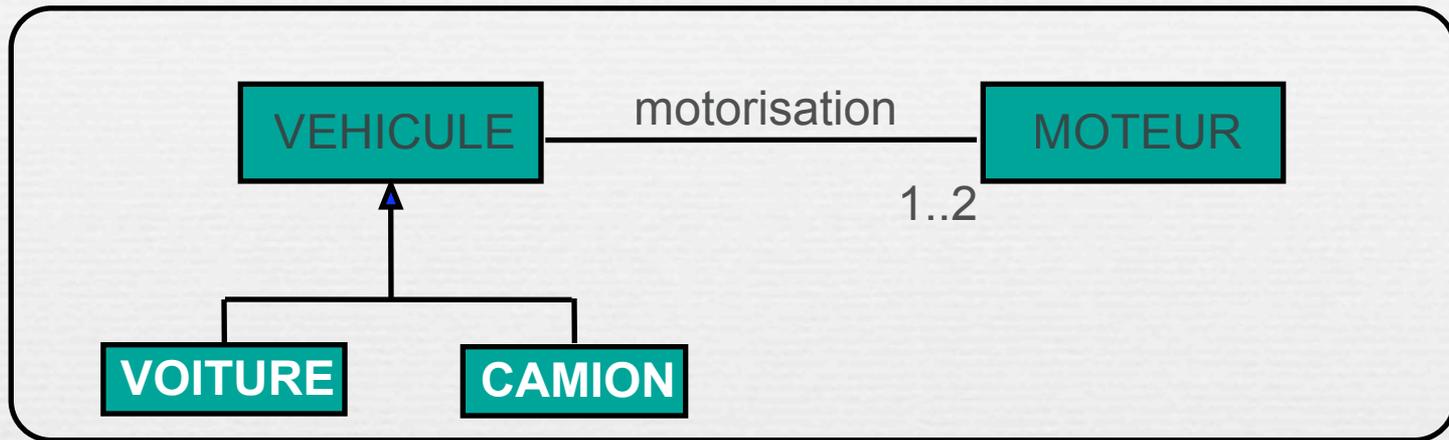
Généralisation

- Extension par l'ajout d'attributs et d'opérations



Héritage des relations

- Les relations sont héritées par les sous classes :



Généralisation signifie :

- Héritage
 - L'enfant acquiert les propriétés du ou des parents (s) : les attributs, les relations et les opérations
- Substituabilité
 - Il est possible de substituer une instance d'une sous-classe à une instance de la classe.

- UML Par la pratique (surtout dans sa dernière édition)
- Méthodologie en Ingénierie du logiciel, Modélisation Orientée objet, M.Grimaldi – janvier 2010

Application

Systeme de réservation de vol



Approfondissement Par l'exemple

Interview des experts métier

1. Des compagnies aériennes proposent différents vols.
2. Un vol est ouvert à la réservation et refermé sur ordre de la compagnie.
3. Un client peut réserver un ou plusieurs vols, pour des passagers différents.
4. Une réservation concerne un seul vol et un seul passager.
5. Une réservation peut être annulée ou confirmée.
6. Un vol a un aéroport de départ et un aéroport d'arrivée.
7. Un vol a un jour et une heure de départ, et un jour et une heure d'arrivée.
8. Un vol peut comporter des escales dans des aéroports.
9. Une escale a une heure d'arrivée et une heure de départ.
10. Chaque aéroport dessert une ou plusieurs villes.

Interview des experts métier

1. Des **compagnies aériennes** proposent différents **vols**.
2. Un vol est **ouvert** à la réservation et **refermé** sur ordre de la compagnie.
3. Un **client** peut réserver un ou plusieurs **vols**, pour des **passagers** différents.
4. Une **réservation** concerne un seul **vol** et un seul **passager**.
5. Une **réservation** peut être **annulée** ou **confirmée**.
6. Un **vol** a un **aéroport** de départ et un **aéroport** d'arrivée.
7. Un **vol** a un **jour et une heure de départ**, et un **jour et une heure d'arrivée**.
8. Un **vol** peut comporter des **escales** dans des **aéroports**.
9. Une **escale** a une **heure d'arrivée** et une **heure de départ**.
10. Chaque **aéroport** dessert une ou plusieurs **villes**.

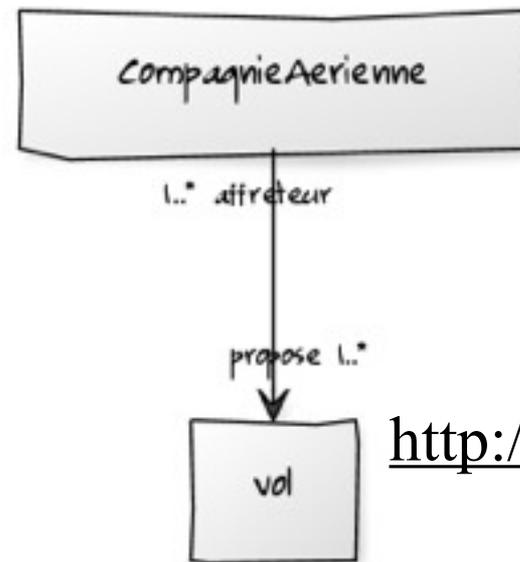
Nous partirons du principe qu'un vol est proposé le plus souvent par une seule compagnie aérienne, mais qu'il peut également être partagé entre plusieurs affréteurs.

<http://yuml.me>

2. Un vol **est ouvert** à la réservation et **refermé** sur ordre de la compagnie.

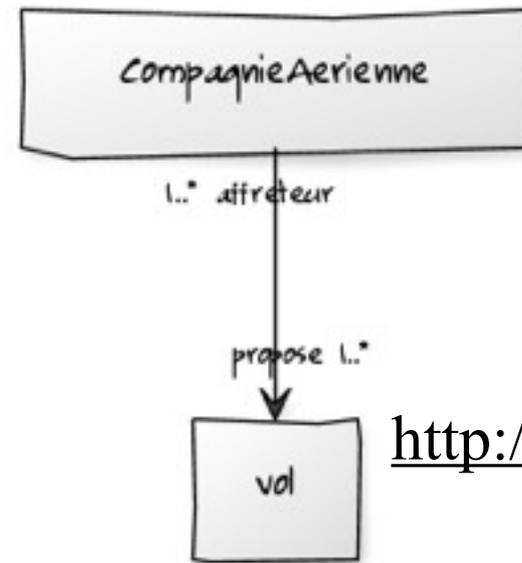
Nous partirons du principe qu'un vol est proposé le plus souvent par une seule compagnie aérienne, mais qu'il peut également être partagé entre plusieurs affréteurs.

2. Un vol **est ouvert** à la réservation et **refermé** sur ordre de la compagnie.



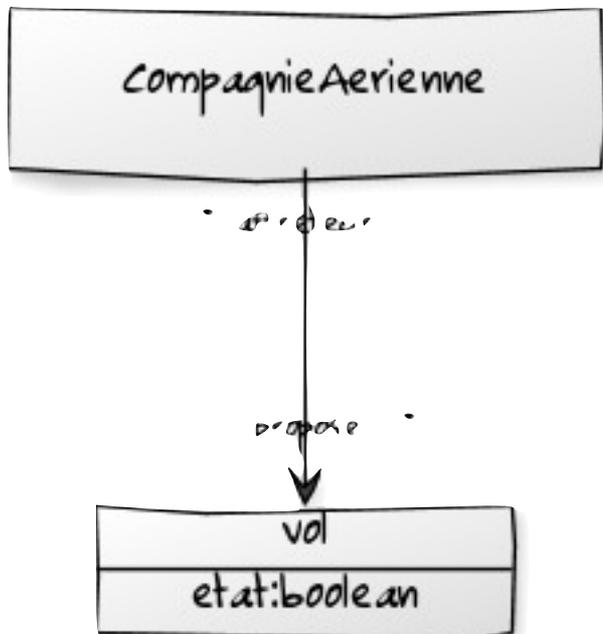
<http://yuml.me>

Nous partirons du principe qu'un vol est proposé le plus souvent par une seule compagnie aérienne, mais qu'il peut également être partagé entre plusieurs affréteurs.



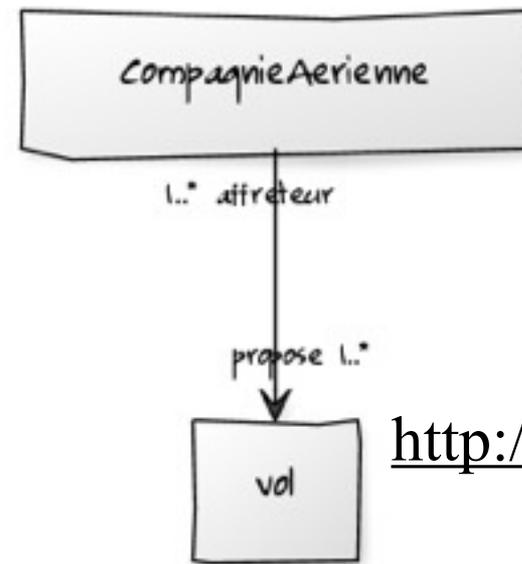
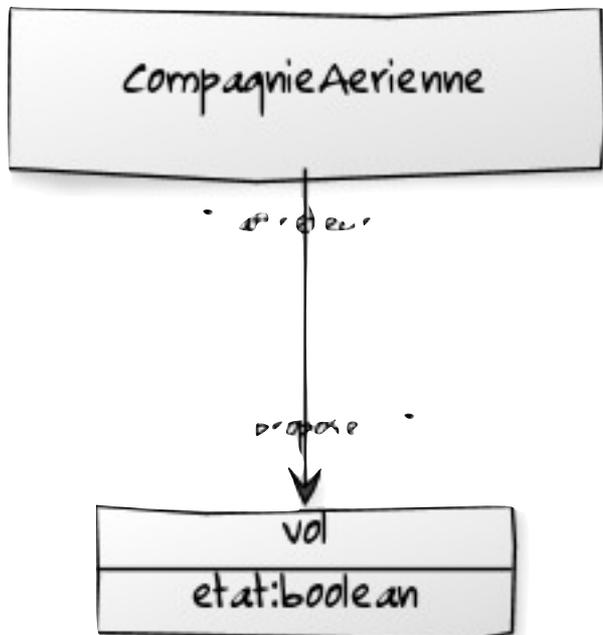
<http://yuml.me>

2. Un vol **est ouvert** à la réservation et **refermé** sur ordre de la compagnie.

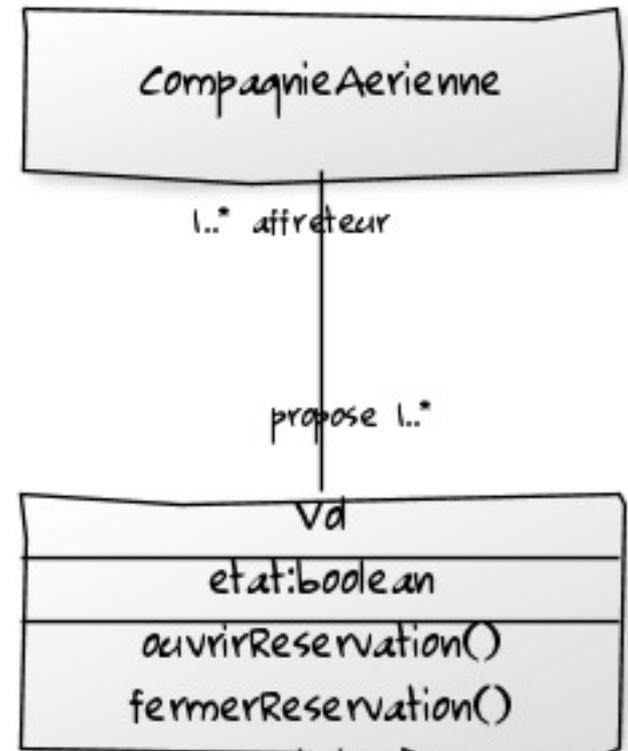


Nous partirons du principe qu'un vol est proposé le plus souvent par une seule compagnie aérienne, mais qu'il peut également être partagé entre plusieurs affréteurs.

2. Un vol **est ouvert** à la réservation et **refermé** sur ordre de la compagnie.



<http://yuml.me>

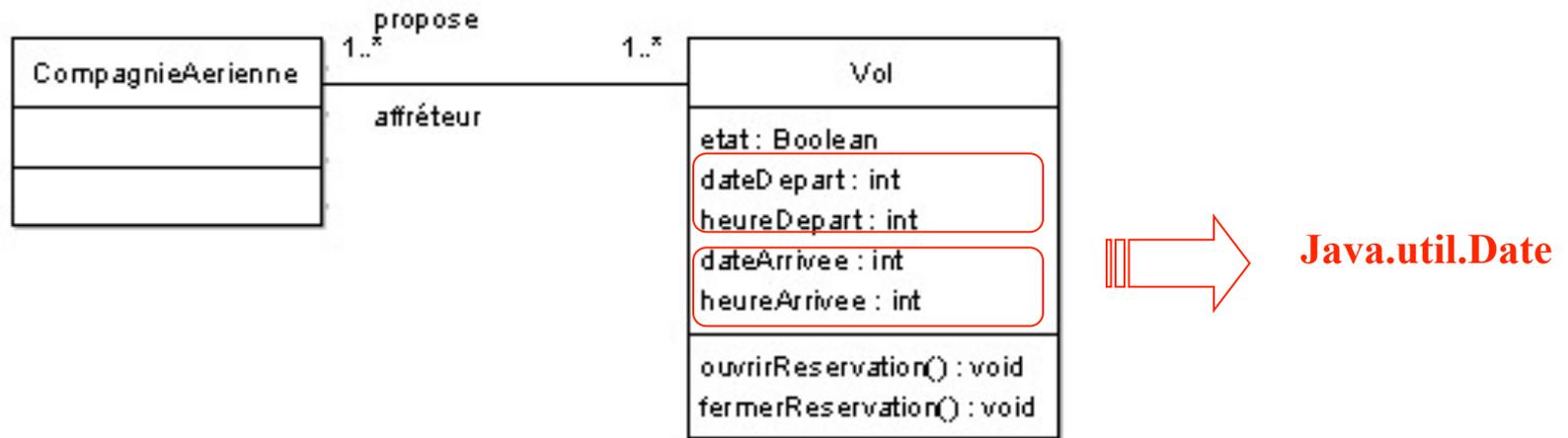


Étape 2 - Modélisation des phrase 6, 7 et 8

7. Un vol a **un jour** et une **heure** de **départ**, et un **jour** et une **heure** d'arrivée.

Étape 2 - Modélisation des phrase 6, 7 et 8

7. Un vol a **un jour** et une **heure** de **départ**, et un **jour** et une **heure** d'arrivée.



Objet ou attribut?

Un objet est un élément plus « important » qu'un attribut.

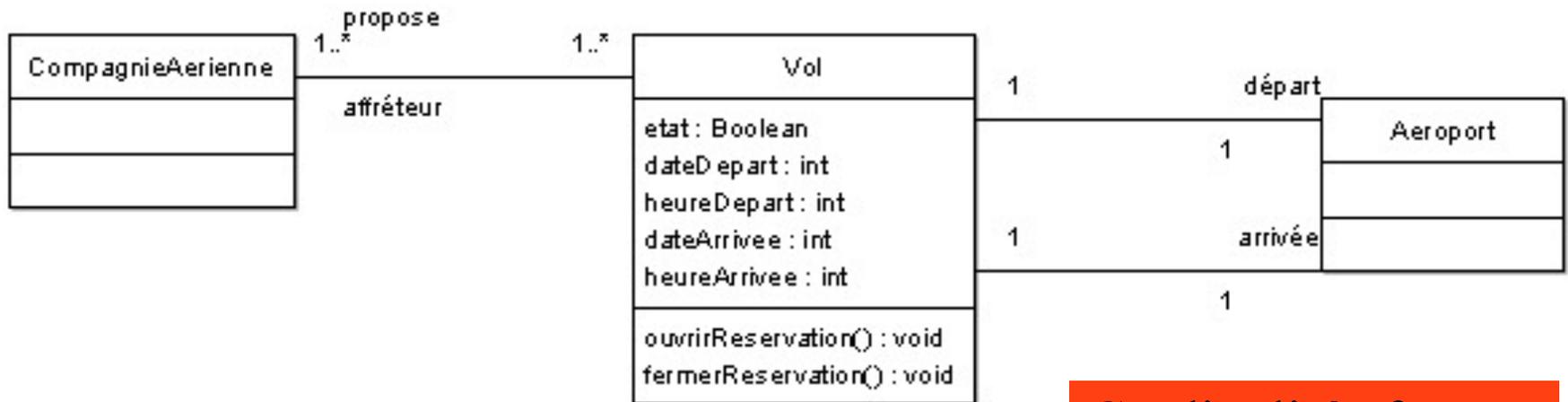
- si l'on ne peut demander à un élément que sa valeur - **attribut**
- si plusieurs questions s'y appliquent - **objet** (qui possède lui-même plusieurs attributs, ainsi que des liens avec d'autres objets.)

6. Un vol a un **aéroport de départ** et un **aéroport d'arrivée**.

Cardinalités fausses

6. Un vol a un **aéroport de départ** et un **aéroport d'arrivée**.

Contrairement aux notions d'heure et de date qui sont des types «*simples* », la notion **d'aéroport** est complexe; elle fait partie du «**métier**». Un aéroport ne possède pas seulement un nom, il a aussi une capacité, dessert des villes, ...etc... A ce stade nous connaissons mal l'usage qui en sera fait. C'est pour ces raisons que nous préférons créer une classe *Aéroport* plutôt que de simples attributs *aéroportDepart* et *aéroportArrivee* dans la classe *Vol*.



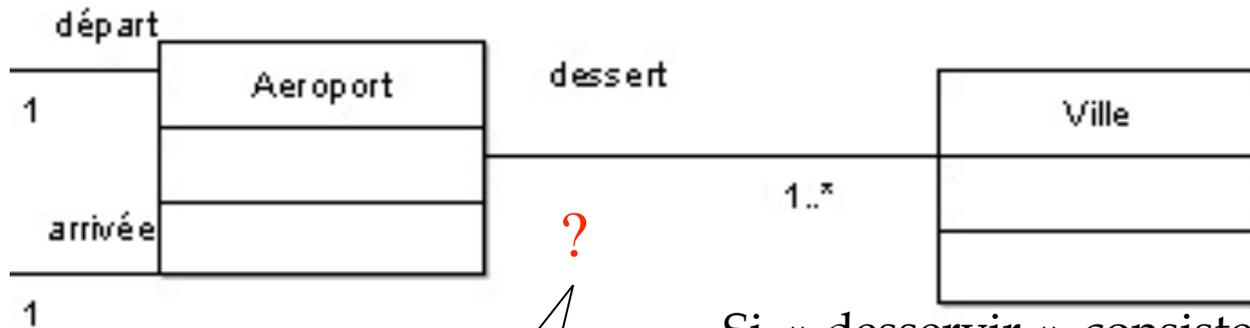
Cardinalités fausses

Modélisation de la phrase 10.

10. Chaque aéroport dessert une ou plusieurs villes.

Modélisation de la phrase 10.

10. Chaque aéroport **dessert** une ou plusieurs **villes**.



1

0..*

Si « desservir » consiste simplement à désigner le moyen de transport par les airs le plus proche, toute ville est toujours desservie par **un et un seul** aéroport.

Si « desservir » vaut par exemple pour tout moyen de transport aérien se trouvant à moins de trente kilomètres, alors une ville peut être desservie par **0 ou plusieurs** aéroports.

Étape 3 - Modélisation des phrases 8 et 9

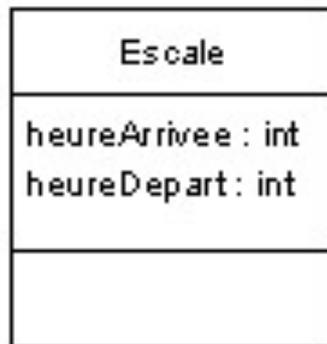
8. Un **vol** peut comporter des **escales** dans des **aéroports**.
9. Une **escale** a une **heure d'arrivée** et une **heure de départ**.

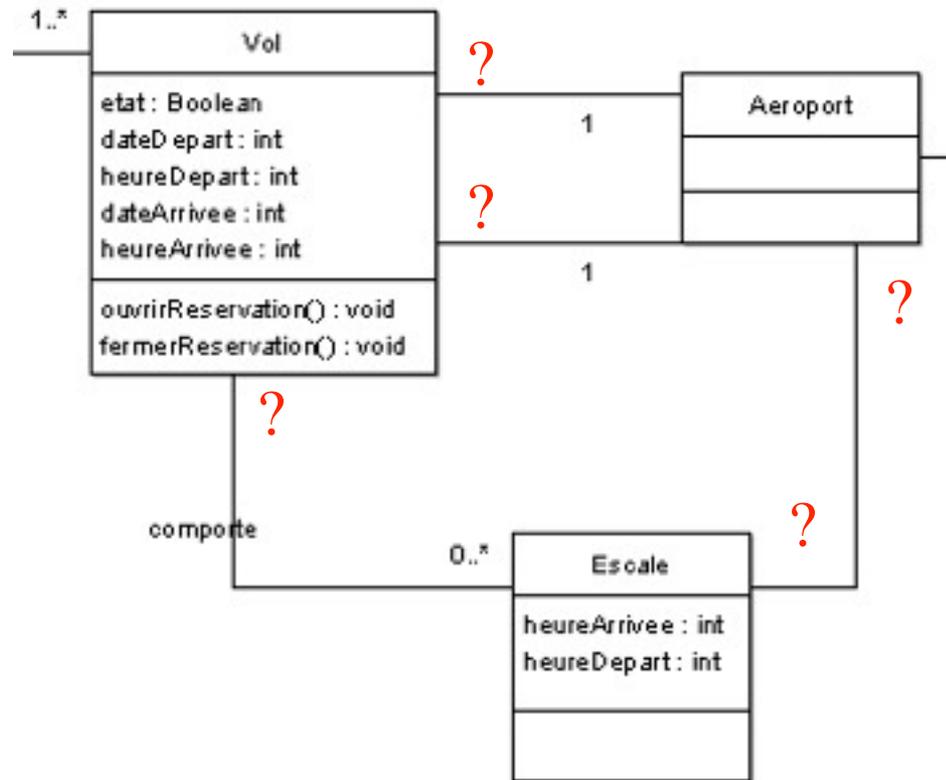
Étape 3 - Modélisation des phrases 8 et 9

8. Un **vol** peut comporter des **escales** dans des **aéroports**.
9. Une **escale** a une **heure d'arrivée** et une **heure de départ**.

- Chaque escale a deux **propriétés** : **heure d'arrivée** et **heure de départ**.
- Elle est en relation avec des **vols** et des **aéroports**, qui sont eux-mêmes des **objets** (phrase 8).

Il est donc naturel d'en faire une classe à son tour.



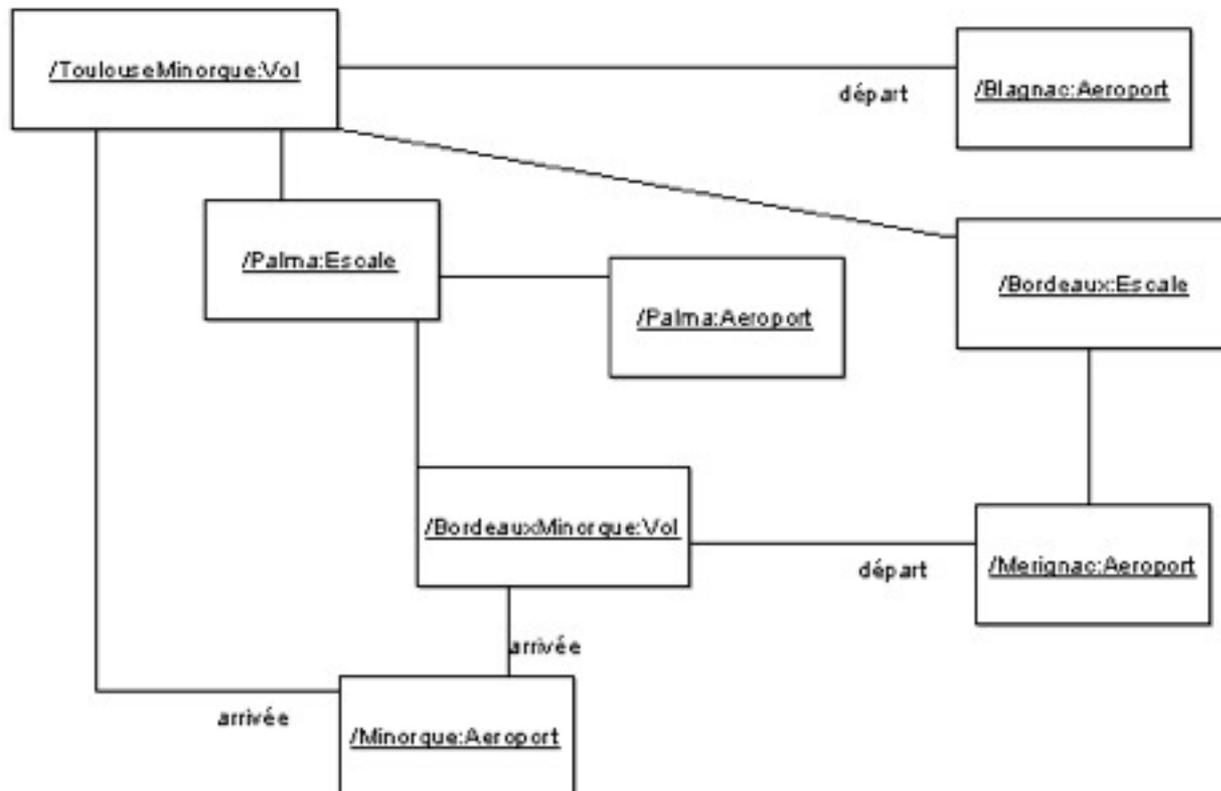


La phrase 8 est imprécise : une escale peut-elle appartenir à plusieurs vols, et quelles sont les multiplicités entre *Escale* et *Aéroport* ?

De plus, le schéma n'indique toujours pas les multiplicités du côté *Vol* avec *Aéroport*

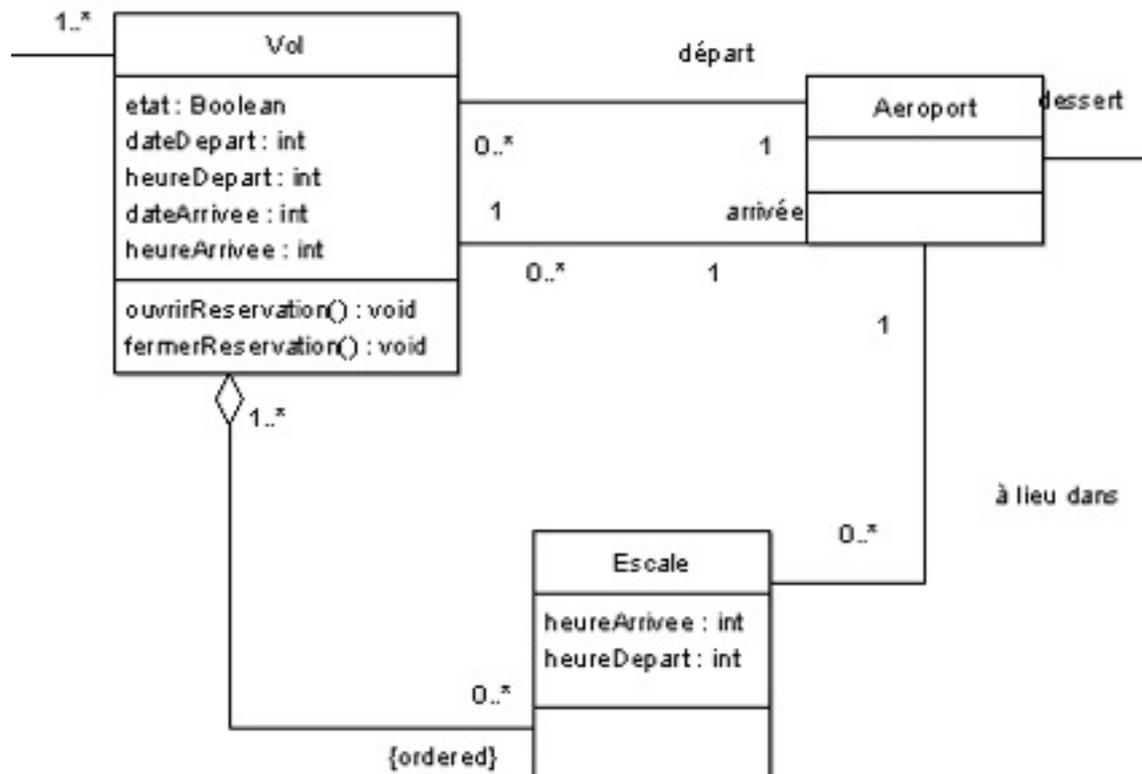
On peut ajouter les multiplicités entre *Escale* et *Aéroport*: une escale a lieu dans un et un seul aéroport, et un aéroport peut servir à plusieurs escales. De même, un aéroport peut servir de départ ou d'arrivée à plusieurs vols.

Après consultation de l'expert métier, un contre-exemple nous est donné, sous forme du diagramme d'objets (collaboration)



Pour finaliser le diagramme des phrases 8 et 9, il nous suffit d'ajouter deux précisions :

- l'association entre **Vol** et **Escale** est une **agrégation** (pas une composition, puisqu'elle est partageable) ;
- les escales sont **ordonnées** par rapport au vol.



Étape 4 - Modélisation des phrases 3, 4 et 5

3. Un **client** peut réserver un ou plusieurs **vols**, pour des **passagers** différents.
4. Une **réservation** concerne un seul **vol** et un seul **passager**.
5. Une **réservation** peut être annulée ou confirmée.

Étape 4 - Modélisation des phrases 3, 4 et 5

3. Un **client** peut réserver un ou plusieurs **vols**, pour des **passagers** différents.
4. Une **réservation** concerne un seul **vol** et un seul **passager**.
5. Une **réservation** peut être annulée ou confirmée.

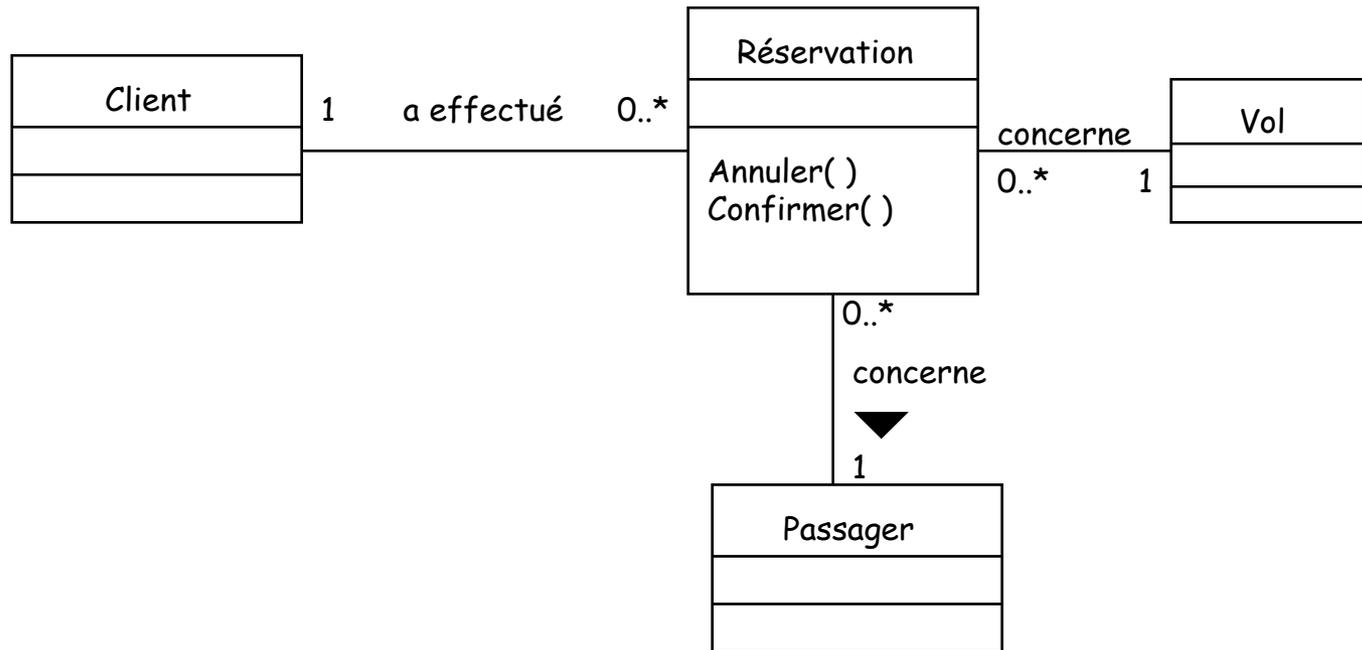
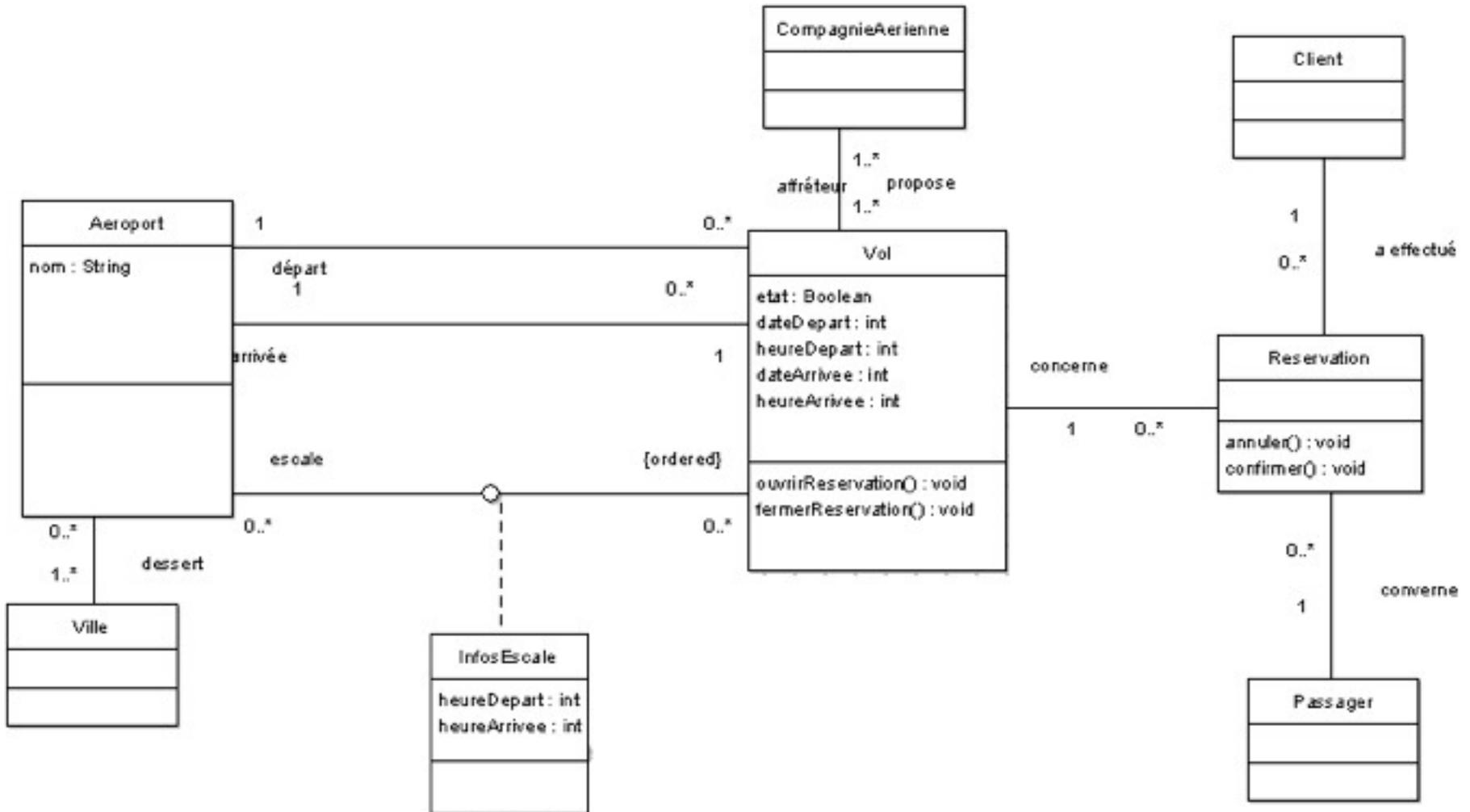


Diagramme de classes



Conseils pratiques

- Réfléchir au problème avant de commencer
 - Soigner le nommage, insister sur le nommage des relations et des rôles
 - Les noms des attributs débutent par une minuscule
 - Les noms des classes débutent par une majuscule et peuvent contenir plusieurs mots concaténés commençant par une majuscule

Conseils pratiques

- Réfléchir au problème avant de commencer
 - Soigner le nommage, insister sur le nommage des relations et des rôles
- Faire simple!
 - «*Things must be as simple as possible, but no simpler*». A. Einstein
 - éviter toute complication nuisible
 - se dégager de l'implémentation : raisonner objets, classes, messages, relations, attributs, opérations
 - ne pas s'inquiéter si les possibilités de la notation ne sont pas toutes exploitées

Conseils pratiques (suite)

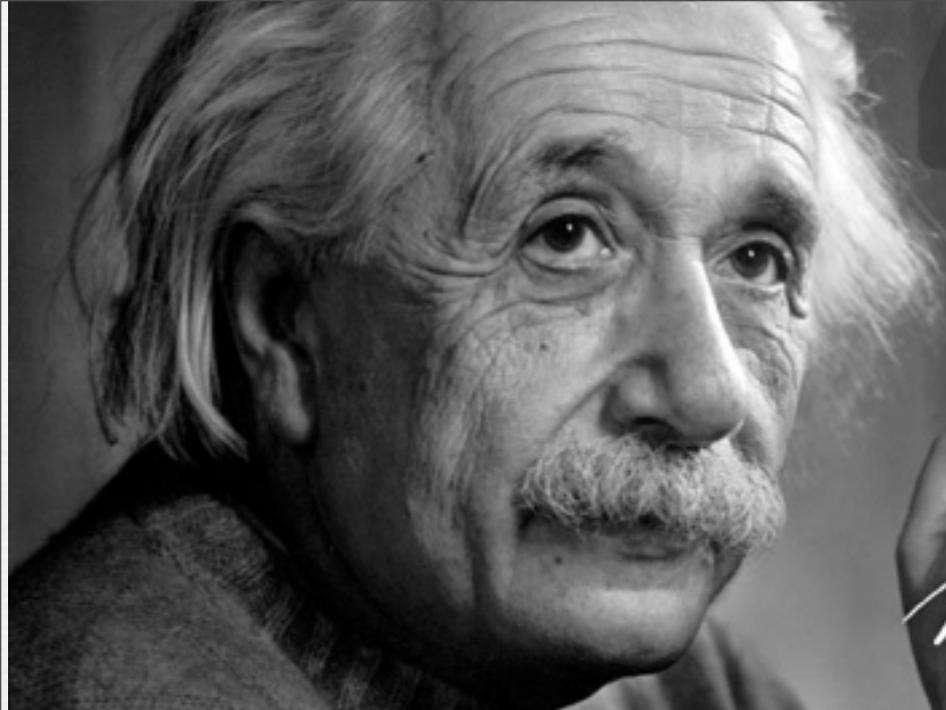
■ Approche incrémentale

- Itérer
- Savoir s'arrêter avant d'atteindre la perfection...
 - prise en compte qualité (niveau de précision), coûts, délais...
 - asservissement au processus de développement

■ Faire simple (encore)

- *Un bon modèle n'est pas un modèle où l'on ne peut plus rien ajouter, mais un modèle où on ne peut plus rien enlever.*

(d'après A. de St-Exupéry)



“
EVERYTHING SHOULD BE MADE
AS SIMPLE AS POSSIBLE
BUT NOT
SIMPLER
”

Albert Einstein

“
Il semble que la perfection soit atteinte
non quand il n'y a plus rien à ajouter,
mais quand il n'y a plus rien à retrancher. ”

ANTOINE DE SAINT-EXUPERY

