Génie Logiciel

D*iagramme d 'objets*

Diagramme de classes

Objets et classes

Objet : une **entité concrète** avec une identité bien définie qui encapsule un **état** et un **comportement**. L 'état est représenté par des valeurs d'attribut et des associations, le comportement par des méthodes.

Un objet est une instance d 'une classe.

MaVoiture : Voiture
marque = Renault
Modèle = Nevada
Immatriculation = 648ADX38
AnnéeModele = 1992
Kilométrage = 285 000

Classe : Une description d'un **ensemble d'objets** qui partagent les mêmes attributs, opérations, méthodes, relations et contraintes.

Une classe peut posséder des attributs ou des méthodes « de classe ».

Voiture
marque : chaîne
Modèle : chaîne
Immatriculation : chaîne (8)
AnnéeModele : date
Age_moyen : entier
Rouler ( )
Kilometrage_annuel_moyen ( )

Diagramme d'Objets

Structure statique d'un système, en termes d'**objets** et de **liens** entre ces objets.

Ces objets et ces liens possèdent des attributs qui possèdent des valeurs.

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

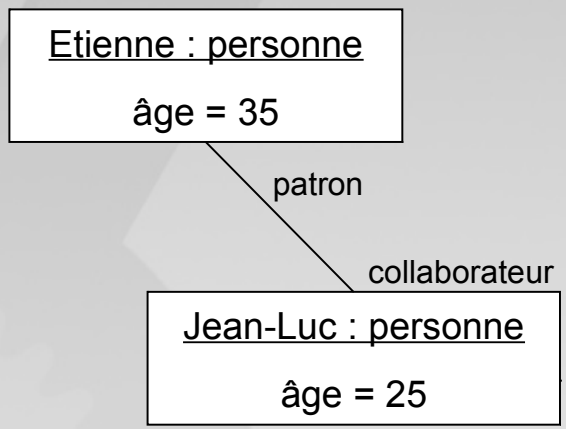
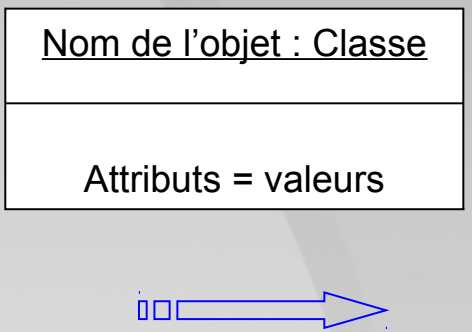


Diagramme d 'objets



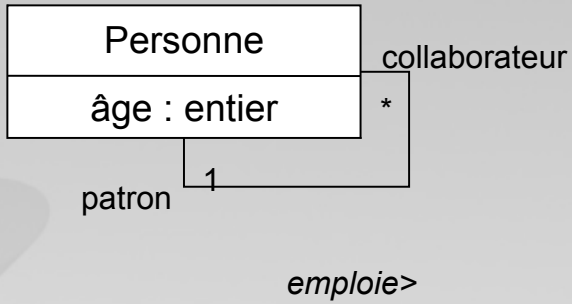


Diagramme de classes

Modèle structurel

- Un objet est instance (propre) d'une classe :
  - *il se conforme à la description que celle-ci fournit,*
  - *il admet une valeur pour chaque attribut déclaré à son attention dans la classe,*
  - *il est possible de lui appliquer toute opération définie à son attention dans la classe.*
- Tout objet admet une identité qui le distingue pleinement des autres objets :
  - *il peut être nommé et être référencé par un nom (mais son identité ne se limite pas à ça).*

Diagramme de classes

Structure statique d'un système, en termes de **classes** et de **relations** entre ces classes.

Nom de classe
Attributs
Opérations ()

exemple :

Voiture
Couleur Cylindrée Vitesse max
Démarrer () Accélérer () Freiner ()

Syntaxe:

- **nom_attribut : type_attribut = valeur initiale**
- **nom_opération (nom_argument : type_argument = valeur_par_défaut, …) : type_retourné**

Visibilité : trois niveaux de visibilité pour les attributs et les opérations:

- public (+) : élément visible à tous les clients de la classe
- protégé (#) : élément visible aux sous-classes de la classe
- privé (-) : élément visible à la classe seule
- package(~) : élément visible à l'ensemble des classes du package

Modèle structurel

Première abstraction

Une *classe* peut être vue comme

- la description en intension d'un groupe d'objets ayant
  - même structure (même ensemble d'attributs),
  - même comportement (mêmes opérations),
  - une sémantique commune.
- la « génitrice » des objets ou instances
- le « conteneur » (extension) de toutes ses instances

Classe et Attributs (propriétés)

[Visibilité] nom [[multiplicité]][:type][=valeur initiale][{propriétés}]

+ - ~ #

[0..1]

[n]

[2..*]

Nom de classe, expression

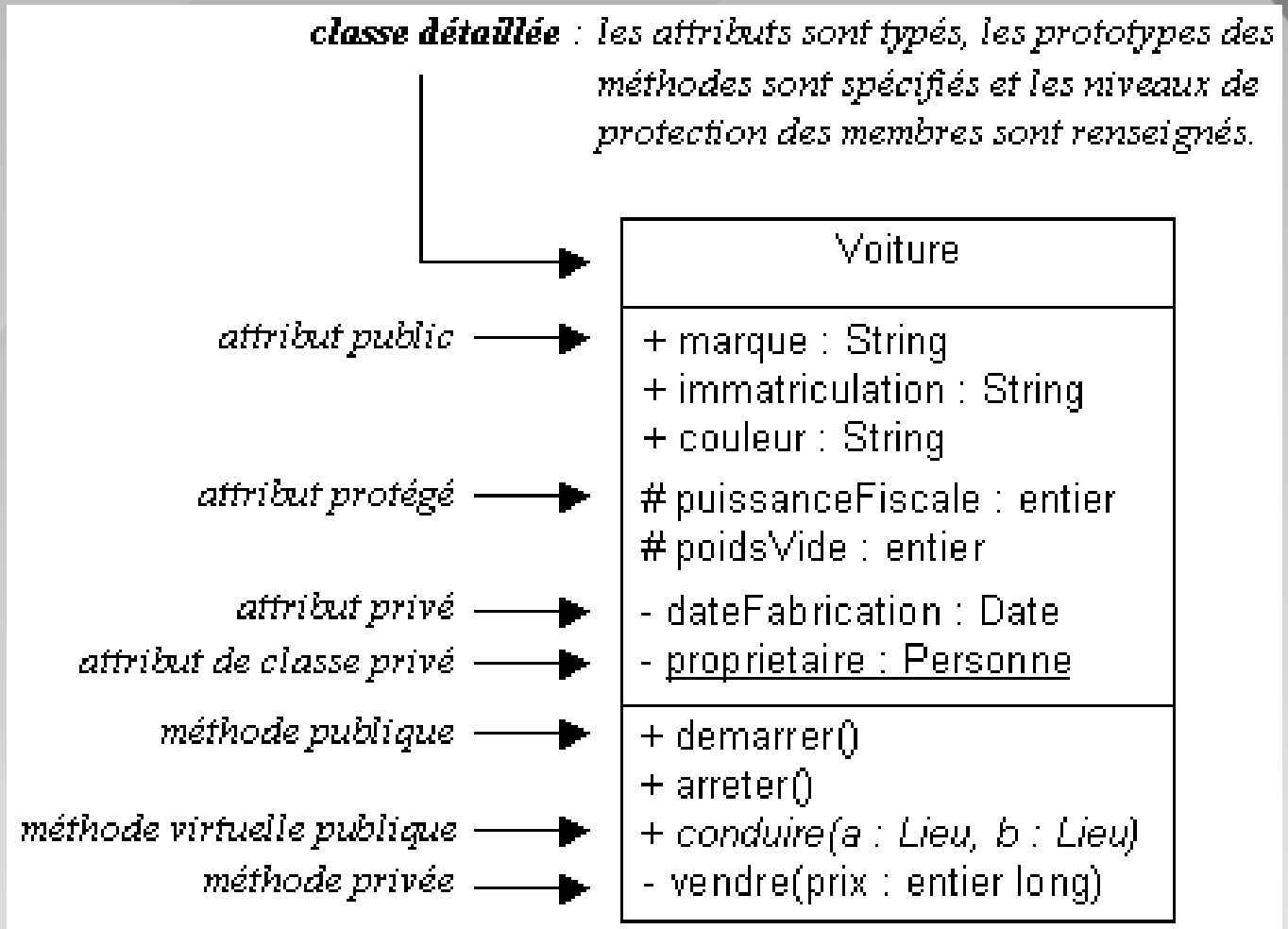
constant

addOnly

Les attributs

- Chaque instance d'une classe possède sa propre copie des attributs de la classe. Les valeurs des attributs peuvent donc différer d'un objet à un autre.
-
- Il est parfois nécessaire de définir un *attribut de classe* (*static* en Java ou en C++) qui garde une valeur unique et partagée par toutes les instances de la classe.
-
- Les instances ont accès à cet attribut mais n'en possèdent pas une copie.
-
- Un attribut de classe n'est donc pas une propriété d'une instance mais une propriété de la classe
-
- Graphiquement, un attribut de classe est souligné.

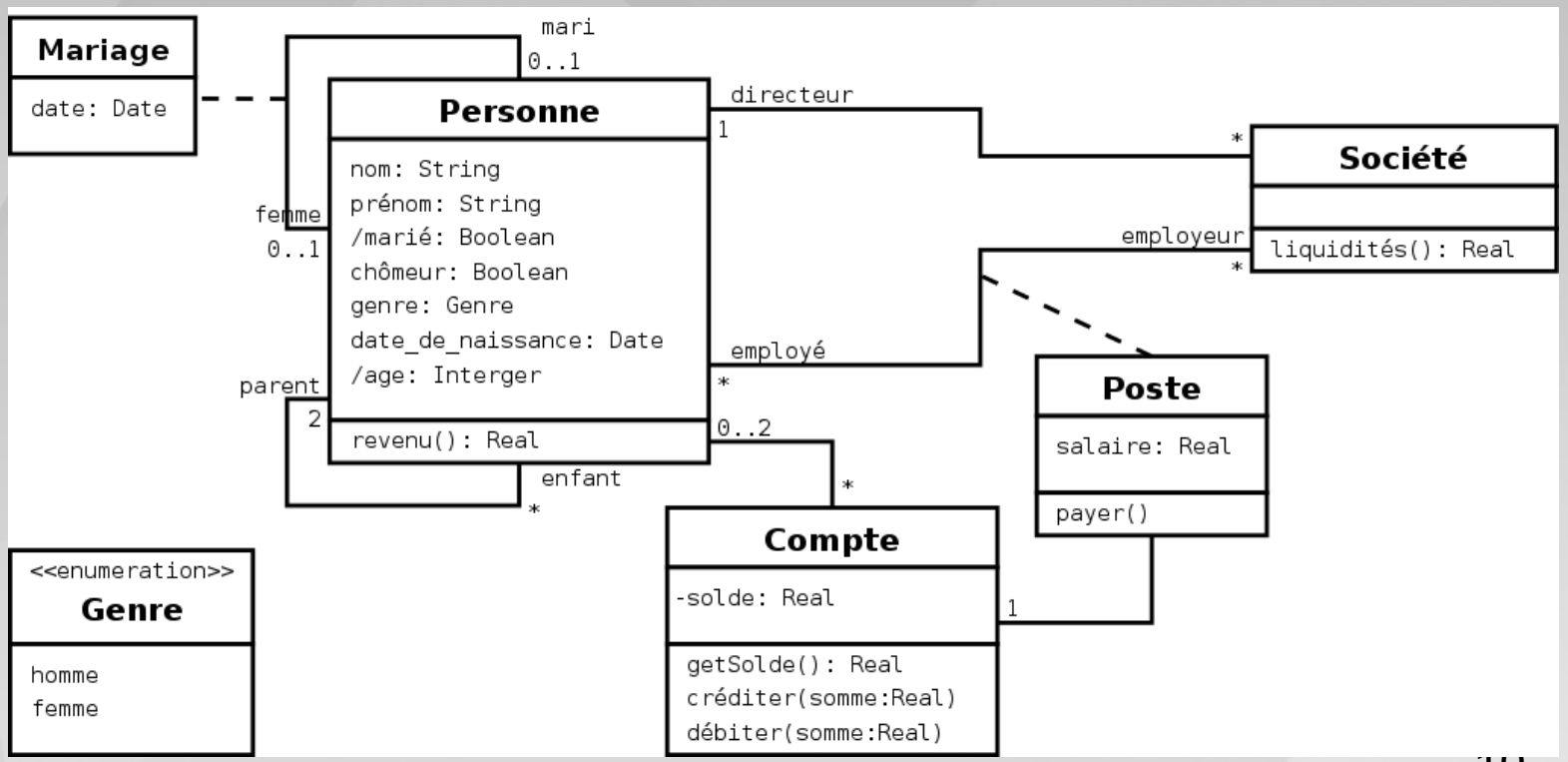
Les attributs



9

Attributs dérivés

- Les attributs dérivés peuvent être calculés à partir d'autres attributs et de formules de calcul.
- Lors de la conception, un attribut dérivé peut être utilisé comme marqueur pour déterminer les règles à lui appliquer.
- Les attributs dérivés sont symbolisés par l'ajout d'un « / » devant leur nom.



Classe, opérations, méthodes

[Visibilité] nom [(paramètres)][:type retour][{propriétés}]

+ - ~ #

[mode] param : type [=valeur défaut]

mode = in (par défaut), out, in/out

query

abstract

Les opérations

- Dans une classe, une opération (même nom et même types de paramètres) doit être unique.
- Quand le nom d'une opération apparaît plusieurs fois avec des paramètres différents, on dit que l'opération est surchargée.
- La déclaration d'une opération contient les types des paramètres et le type de la valeur de retour, sa syntaxe est la suivante :

<visibilité> <nom_opération> ([<paramètre_1>, ... , <paramètre_N>]) : [<type_renvoyé>] [{<propriétés>}]

12

Paramètre d'opération

La syntaxe de définition d'un paramètre (<paramètre>) est la suivante :

[<direction>] <nom_paramètre>:<type> ['['<multiplicité>']'] [=<valeur_par_défaut>]

La direction peut prendre l'une des valeurs suivante :

in
: Paramètre d'entrée passé par valeur. Les modifications du paramètre ne sont pas disponibles pour l'appelant. C'est le comportement par défaut.

out
: Paramètre de sortie uniquement. Il n'y a pas de valeur d'entrée et la valeur finale est disponible pour l'appelant.

inout
: Paramètre d'entrée/sortie. La valeur finale est disponible pour l'appelant.

Le type du paramètre (<type>) peut être

- un nom de classe,
- un nom d'interface,
- un type de donné prédéfini.

13

Opération de classe

- Comme pour les **attributs de classe**, il est possible de déclarer des **opérations de classe.**
- Une opération de classe ne peut manipuler que des **attributs de classe** et ses propres paramètres.
- Cette méthode n'a pas accès aux autres attributs
- Graphiquement, une opération de classe est soulignée.

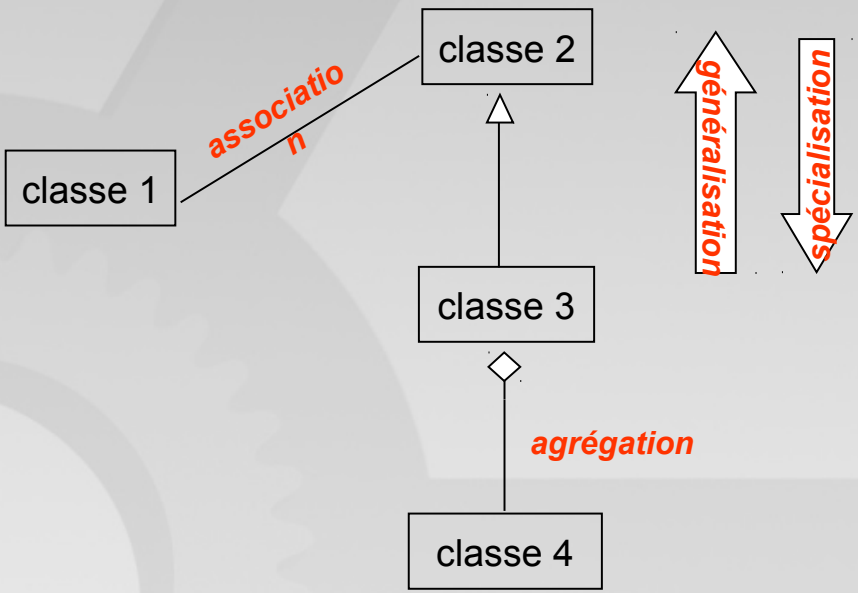
14

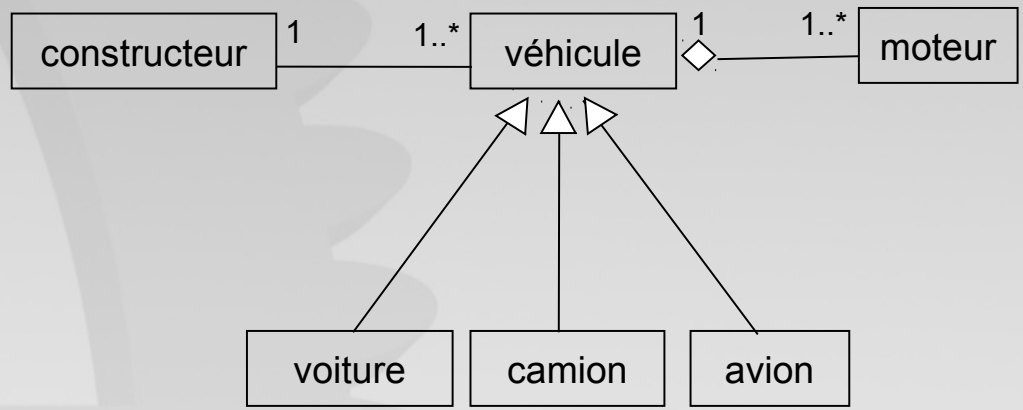
Diagramme de classes : Relations entre classes

Agrégation : quand une classe fait partie d'une autre classe (agrégat - composant)

Association : toute relation structurelle entre classes, autre que l 'agrégation et la généralisation

Généralisation : factorisation des éléments communs d'un ensemble de classes dits *sous-classes* dans une classe plus générale dite *super-classe.* Elle signifie que la sous-classe ***est un*** ou ***est une sorte de*** la super-classe. Le lien inverse est appelé spécialisation





Associations

Agrégation:

• Association transitive : si voiture *est composée de* moteur et si moteur *est composé de* courroie alors voiture *est composée de* courroie

• Association non systémique : si voiture *est composée de* moteur, moteur ne peut pas être *composé de* voiture

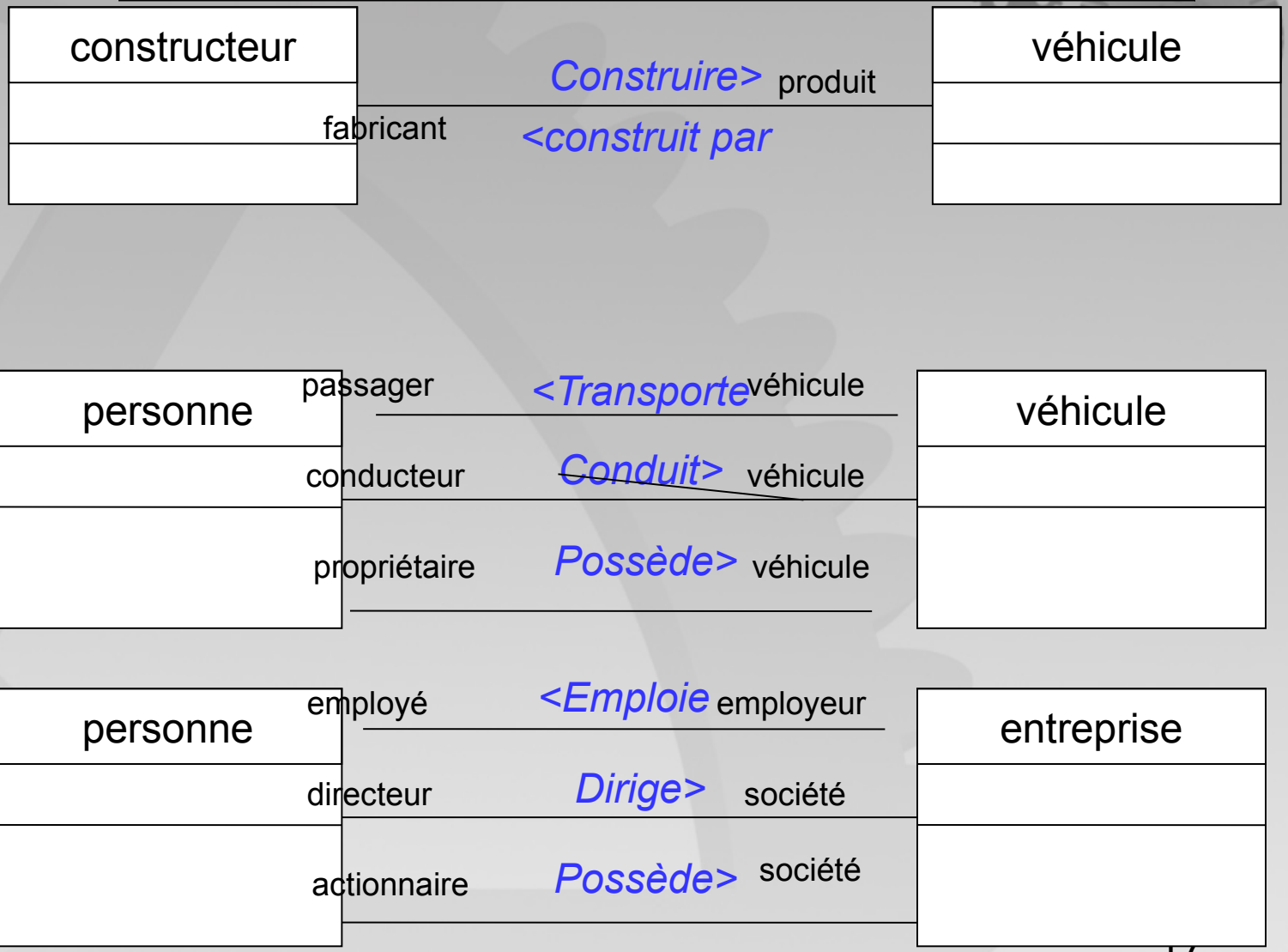
• Association qui peut être réflexive : une fonction peut être *composée* d ’autres fonctions

Rôle et multiplicité :

• Une classe a un rôle dans une association.

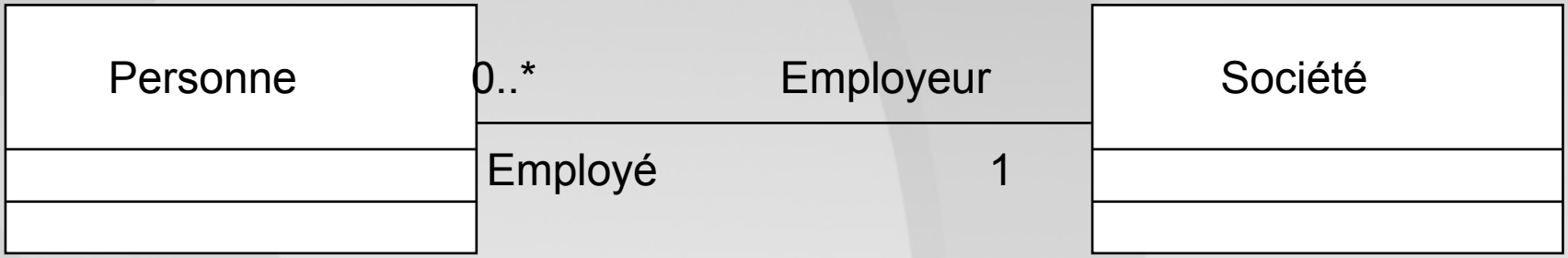
• Les rôles portent une information de multiplicité précisant le nombre d ’associations auquel une instance d ’objet peut être associée. Les multiplicités les plus courantes sont : 1 / 0..1 / m..n / * /0..* / 1..*

Nommage des associations



Multiplicité des associations

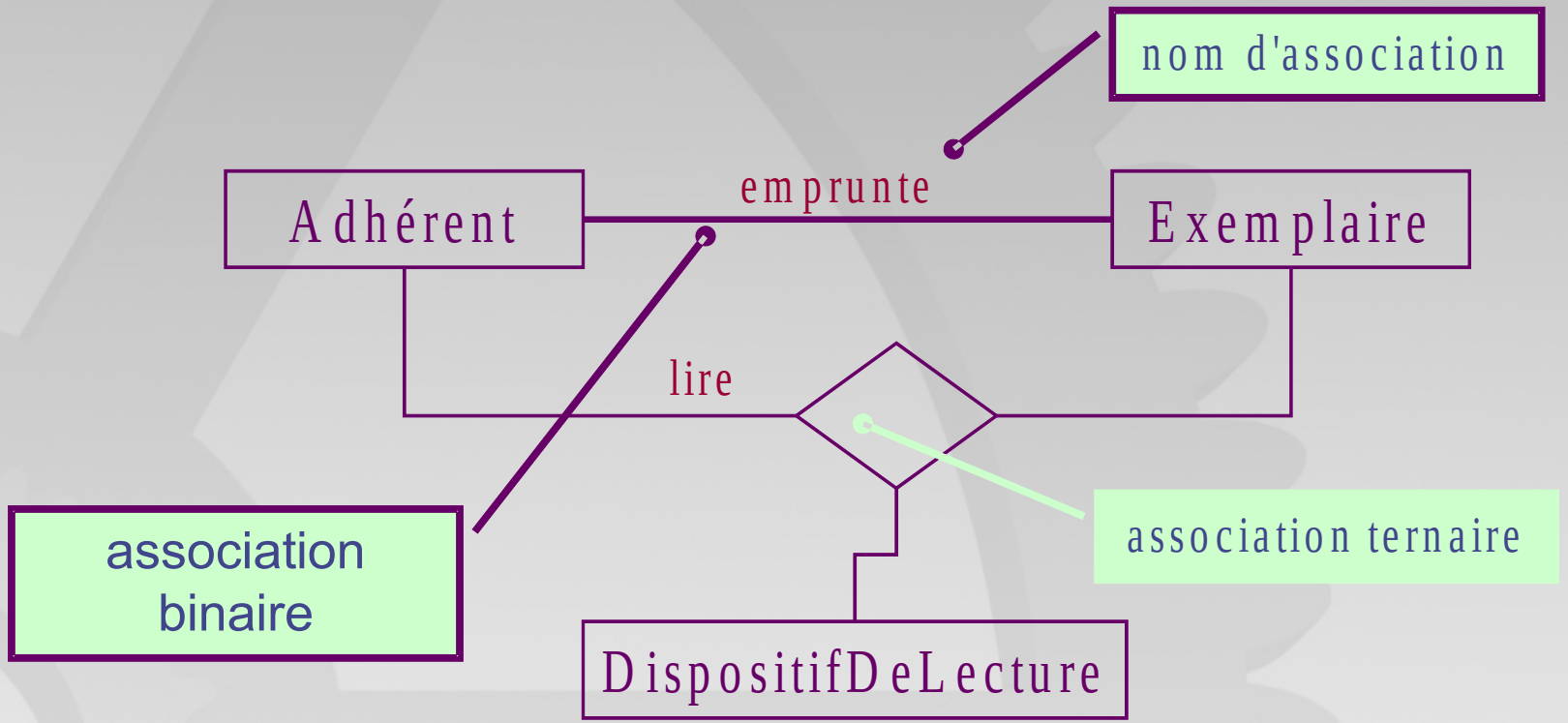
1	Un et un seul (obligatoire)
0 .. 1	Zéro ou un (optionnel)
m .. n	De m à n (entiers)
* *ou* **0 ..** *	quelconque
1 .. *	Au moins 1



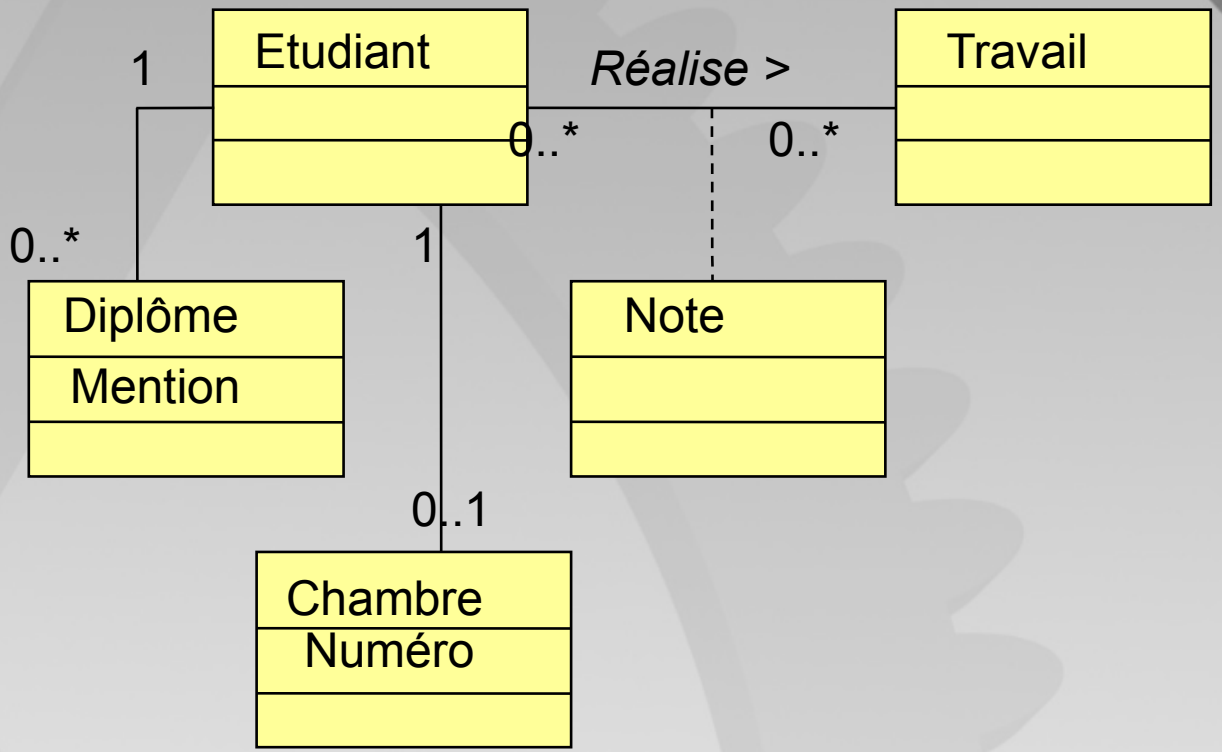
Arité des associations

Association d'arité 3

Association en général binaire (degré = 2) mais ..



Placement des attributs et des associations



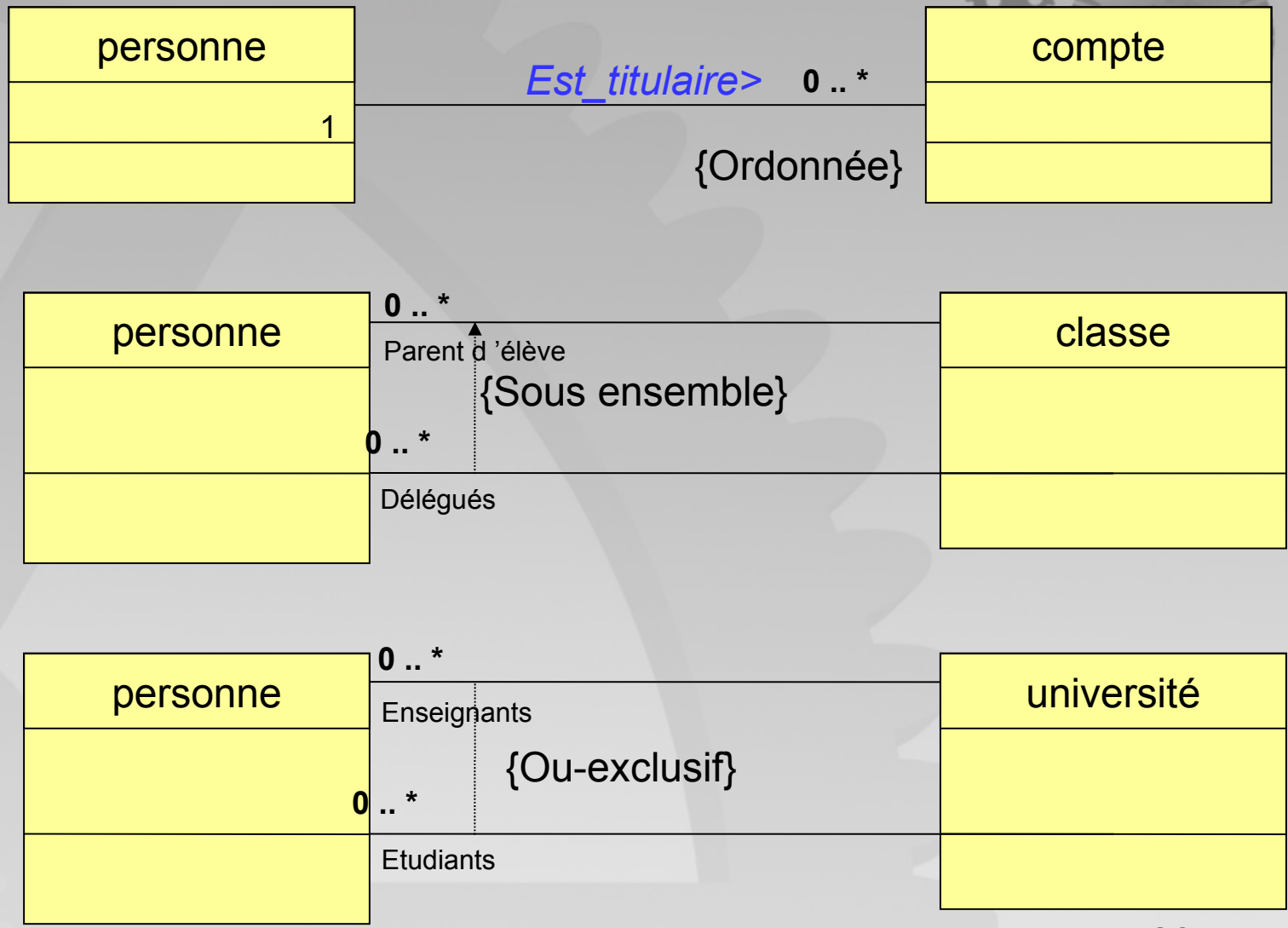
Classe d'association

- Une classe d'association permet de modéliser une association par une classe, donc de disposer d'attributs et d'opérations spécifiques.
- Les liens d'une telle association sont alors des objets instances de cette classe.
- À ce titre, ils admettent une valeur pour tout attribut déclaré dans la classe d'association ; et on peut leur appliquer toute opération définie dans celle-ci.
- En tant que classe, une classe d'association peut à son tour être associée à d'autres classes (voire à elle-même par une association réflexive).

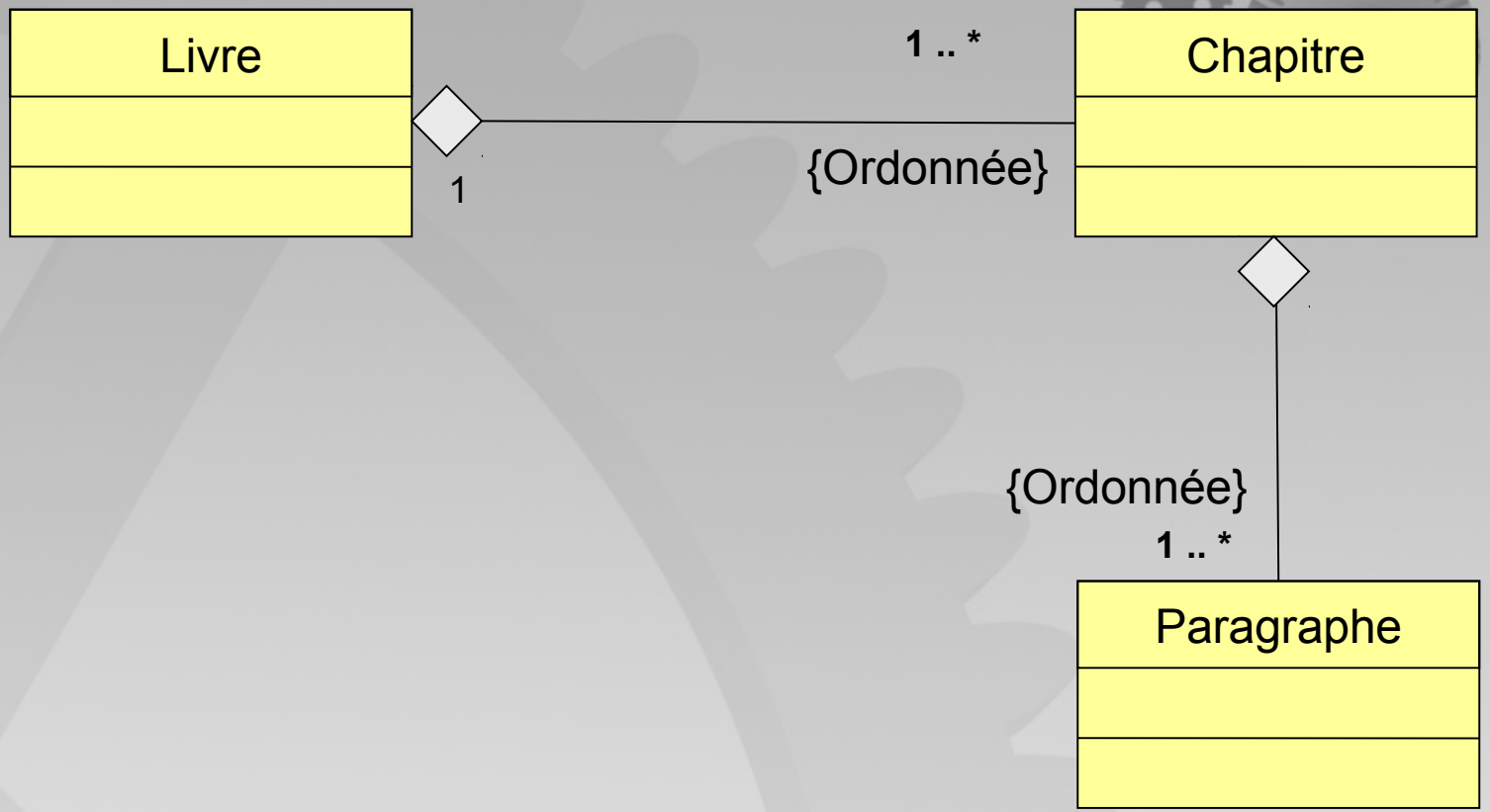
Les contraintes

- Les *contraintes* sont des prédicats, pouvant porter sur plusieurs éléments du modèle statique, qui doivent être vérifiés à tout instant.
- Les contraintes permettent de rendre compte de détails à un niveau de granularité très fin dans un diagramme de classe. Elles peuvent exprimer des conditions ou des restrictions.
- En UML, les contraintes sont exprimées sous forme textuelle, entre accolades et de préférence en OCL (Object Constraint Language).
- Les contraintes sont héritées.

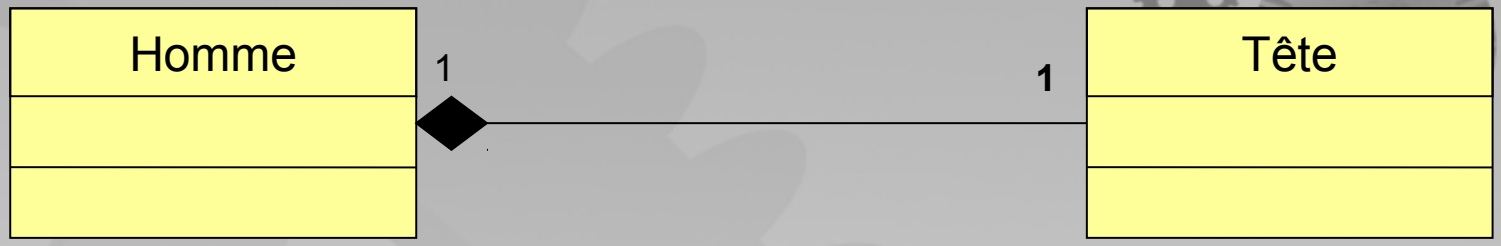
Contraintes



Agrégation

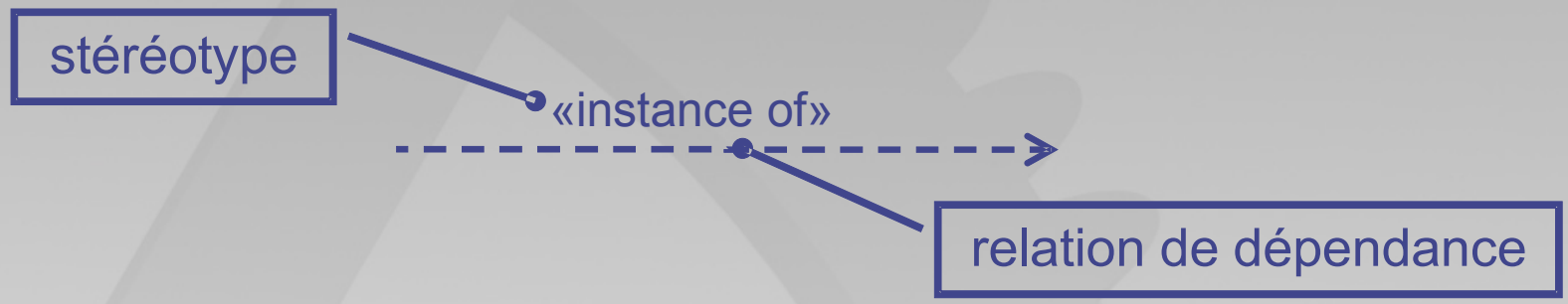


Composition



La composition traduit une dépendance existentielle forte.

Quelques compléments de notation



- Un stéréotype est un label qui permet d'apporter une précision supplémentaire à un élément de notation (classe, relation, ...)

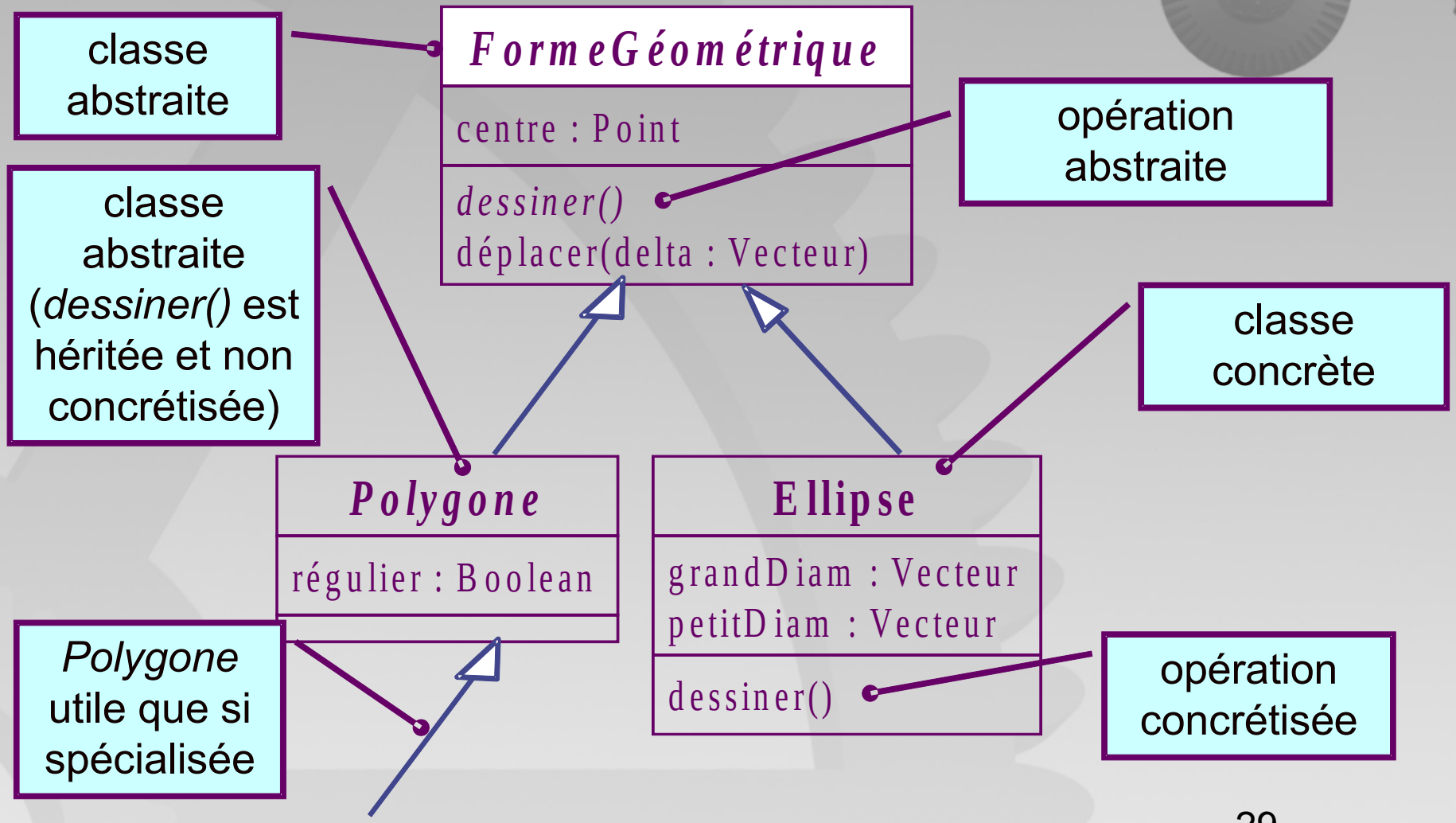
Classes abstraites

- Une *classe abstraite* est une classe non instanciable, c'est à dire qu'elle n'admet pas d'instances directes.
- Une classe abstraite est une description d'objets destinée à être « héritée » par des classes plus spécialisées.
- Pour être utile, une classe abstraite doit admettre des classes descendantes *concrètes*.
- La factorisation optimale des propriétés communes à plusieurs classes par généralisation nécessite le plus souvent l'utilisation de classes abstraites.

Opérations abstraites

- Une *opération abstraite* est une opération n'admettant pas d'implémentation : au niveau de la classe dans laquelle est déclarée, on ne peut pas dire comment la réaliser.
- Les opérations abstraites sont particulièrement utiles pour mettre en œuvre le polymorphisme.
- Toute classe concrète sous-classe d'une classe abstraite doit “concrétiser” toutes les opérations abstraites de cette dernière.

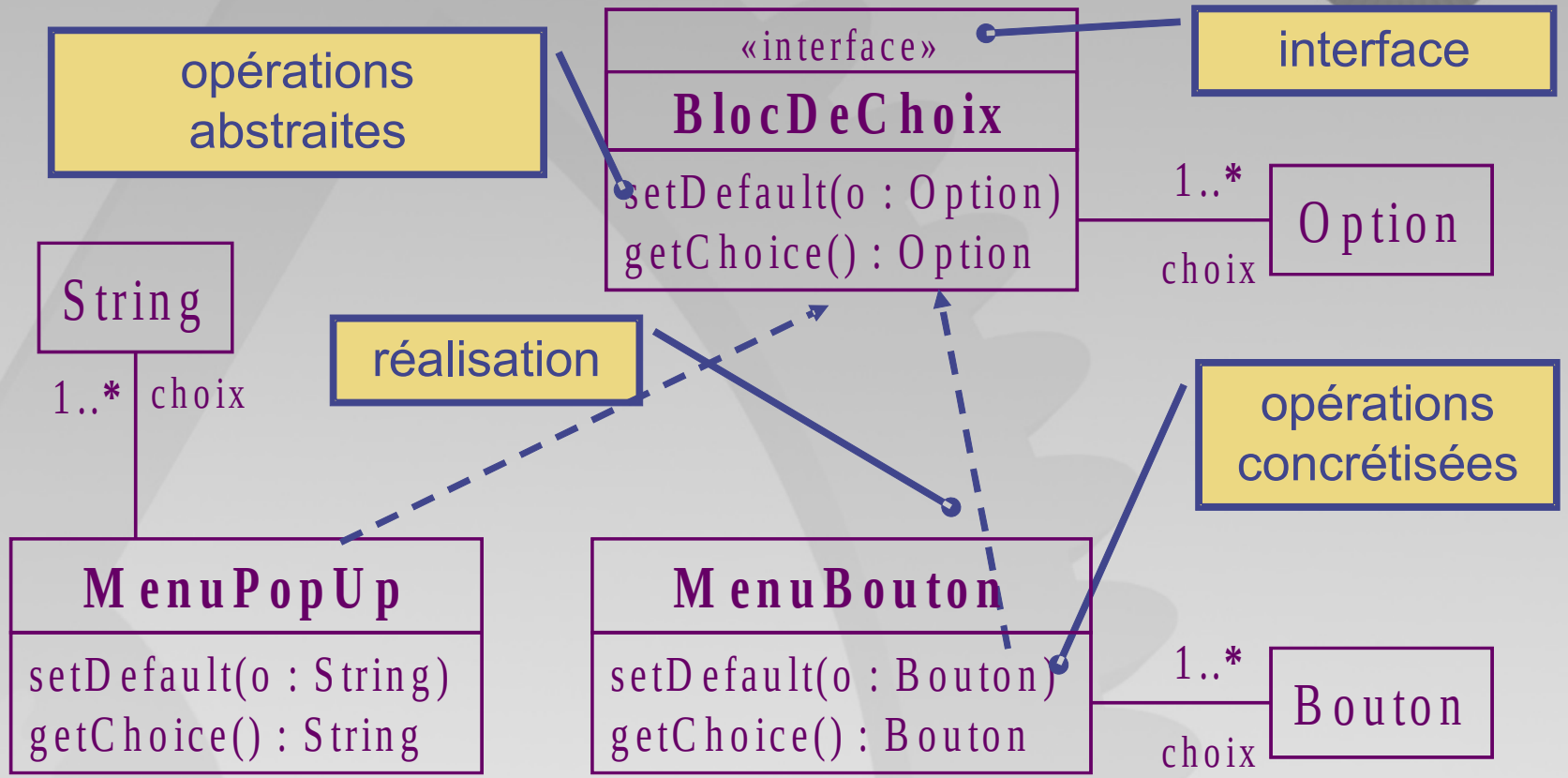
Classes abstraites



Interfaces

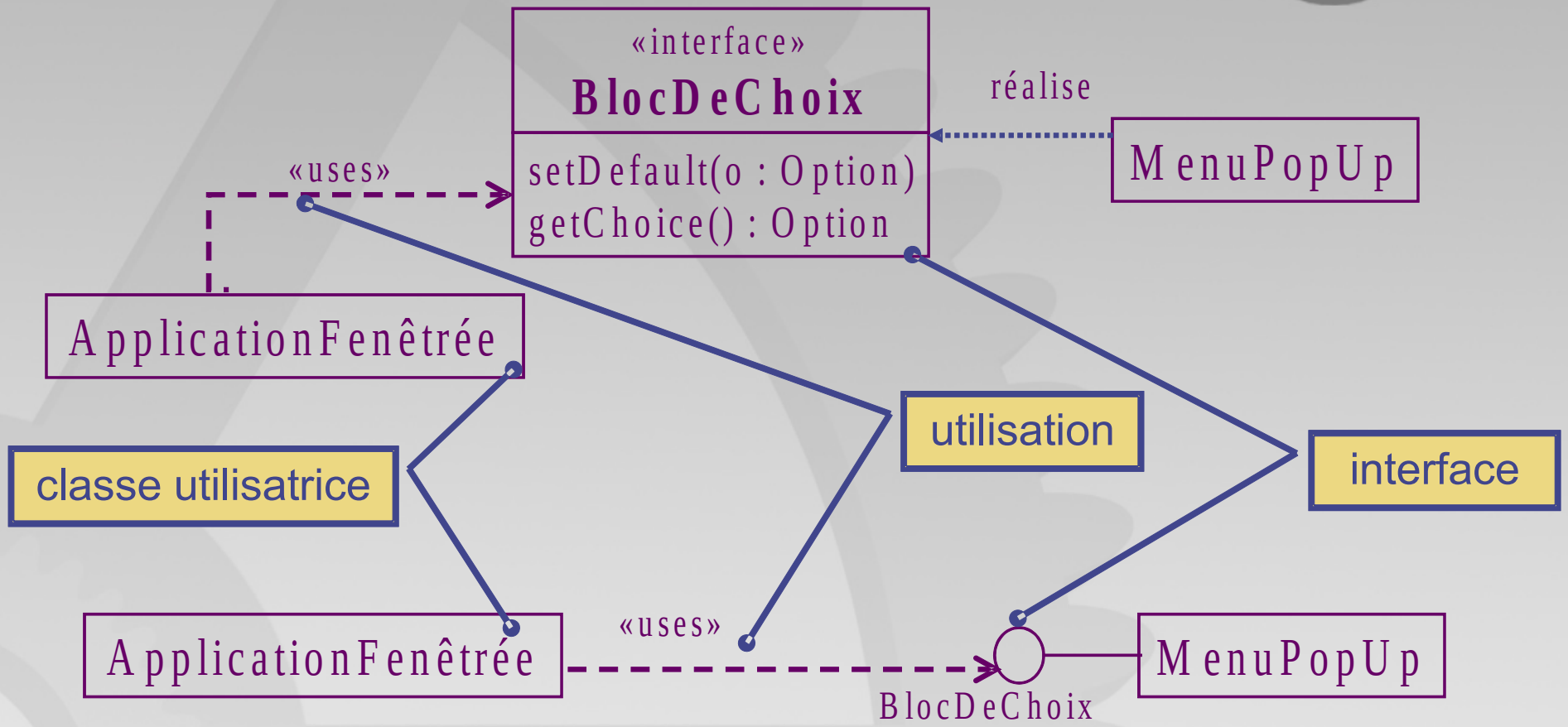
- Une *interface* est une collection d'opérations utilisée pour spécifier un service offert par une classe.
- Une interface être vue comme une classe sans attributs et dont toutes les opérations sont abstraites.
- Une interface est destinée à être "*réalisée*" par une classe (celle-ci en hérite toutes les descriptions et concrétise les opérations abstraites).
- Une interface peut en spécialiser une autre, et intervenir dans des associations avec d'autres interfaces et d'autres classes.

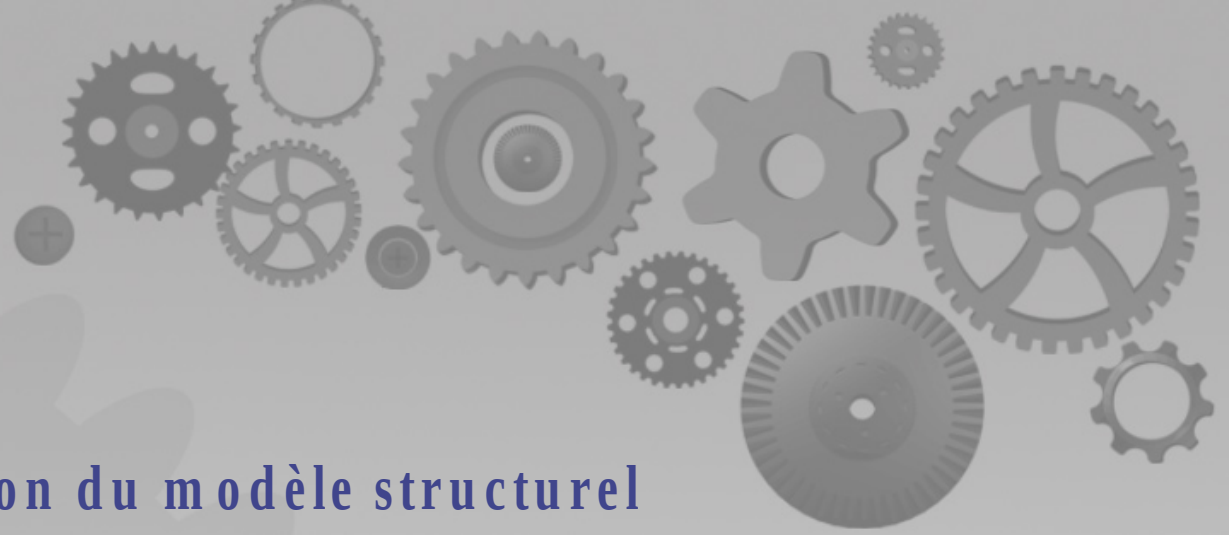
Interfaces


opérations abstraites
«interface»
BlocDeChoix
setDefault(o : Option)
getChoice() : Option
interface
1..*
choix
Option
String
1..*
choix
réalisation
opérations concrétisées
MenuPopUp
setDefault(o : String)
getChoice() : String
MenuBouton
setDefault(o : Bouton)
getChoice() : Bouton
1..*
choix
Bouton

Interfaces

Deux notations pour l'utilisation d'une interface

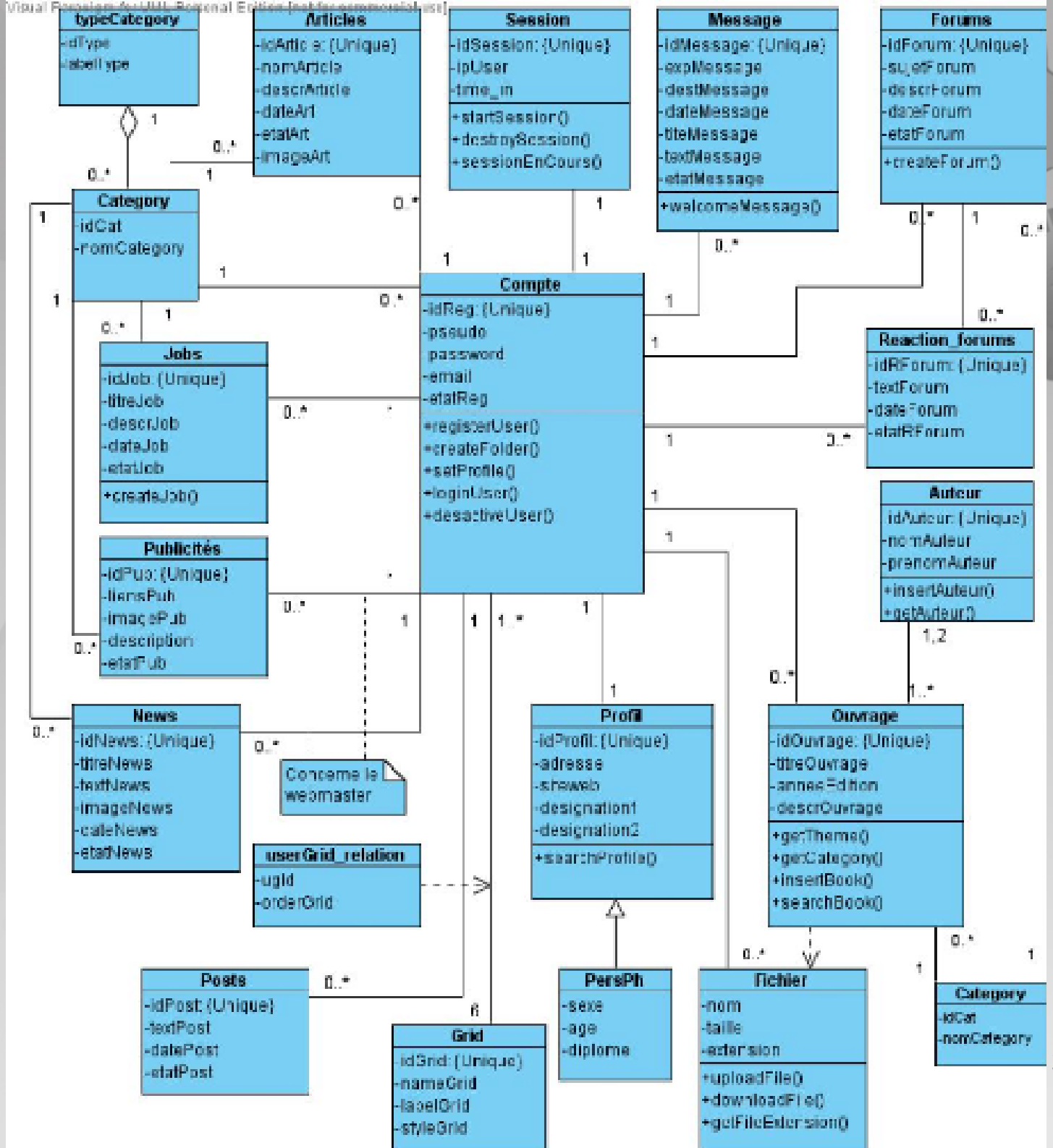


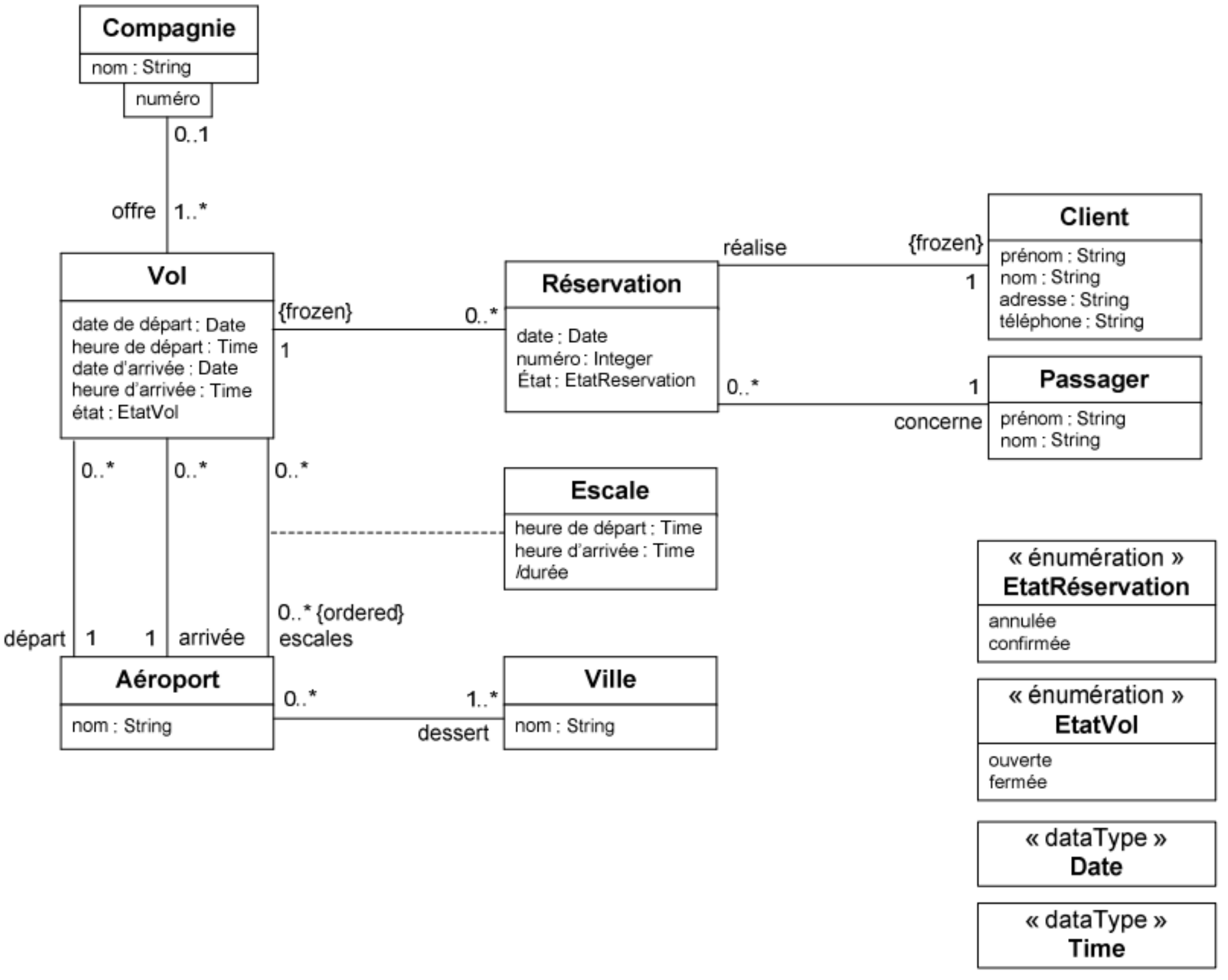


Heuristiques d'élaboration du modèle structurel

- Bien comprendre le problème
- Faire simple
- Bien choisir les noms
- Bien expliciter les associations
- Ne pas trop "généraliser"
- Relire
- Documenter

De nombreuses révisions sont nécessaires !


typeCategory
-idType
-libelleType
Articles
-idArticle: {Unique}
-nomArticle
-descrArticle
-dateArt
-etatArt
-imageArt
Session
-idSession: {Unique}
-ipUser
-time_in
+startSession()
+destroySession()
+sessionEnCours()
Message
-idMessage: {Unique}
-expMessage
-destMessage
-dateMessage
-titeMessage
-textMessage
-etatMessage
+welcomeMessage()
Forums
-idForum: {Unique}
-sujetForum
-descrForum
-dateForum
-etatForum
+createForum()
Category
-idCat
-nomCategory
Compte
-idReg: {Unique}
-pseudo
password
-email
-etatReg
+registerUser()
+createFolder()
+setProfile()
+loginUser()
+desactiveUser()
Jobs
-idJob: {Unique}
-titreJob
-descrJob
-dateJob
-etatJob
+createJob()
Publicités
-idPub: {Unique}
-lienPub
-imagePub
-description
-etatPub
Reaction_forums
-idRForum: {Unique}
-textForum
-dateForum
-etatRForum
Auteur
-idAuteur: {Unique}
-nomAuteur
-prenomAuteur
+insertAuteur()
+getAuteur()
News
-idNews: {Unique}
-titreNews
-textNews
-imageNews
-dateNews
-etatNews
Concerne le webmaster
userGrid_relation
-ugId
-orderGrid
Profil
-idProfil: {Unique}
-adresse
-siteweb
-designation1
-designation2
+searchProfile()
Ouvrage
-idOuvrage: {Unique}
-titreOuvrage
-anneeEdition
-descrOuvrage
+getTheme()
+getCategory()
+insertBook()
+searchBook()
Posts
-idPost: {Unique}
-textPost
-datePost
-etatPost
Grid
-idGrid: {Unique}
-nameGrid
-labelGrid
-styleGrid
PersPh
-sexe
-age
-diplome
Fichier
-nom
-taille
-extension
+uploadFile()
+downloadFile()
+getFileExtension()
Category
-idCat
-nomCategory


Compagnie
nom : String
numéro
0..1
offre
1..*
Vol
date de départ : Date
heure de départ : Time
date d'arrivée : Date
heure d'arrivée : Time
état : EtatVol
{frozen}
1
0..*
Réservation
date : Date
numéro : Integer
État : EtatReservation
réalise
{frozen}
1
Client
prénom : String
nom : String
adresse : String
téléphone : String
0..*
1
concerne
Passager
prénom : String
nom : String
0..*
0..*
0..*
Escale
heure de départ : Time
heure d'arrivée : Time
/durée
0..* {ordered}
escales
départ
1
1
arrivée
Aéroport
nom : String
0..*
1..*
dessert
Ville
nom : String
« énumération »
EtatRéservation
annulée
confirmée
« énumération »
EtatVol
ouverte
fermée
« dataType »
Date
« dataType »
Time