

Bases de données et SGBD orientés objet



Chapitre 1: Introduction aux SGBD orientés objet

INTRODUCTION: RAPPELS

CONSTATS :

- INCAPACITE HUMAINE A MAITRISER LA COMPLEXITE CROISSANTE DES LOGICIELS
- L 'ACCROISSEMENT DE LA PUISSANCE DES ORDINATEURS, ALLIE A LA VULGARISATION DE L 'INFORMATIQUE A PERMIS LE DEVELOPPEMENT D 'APPLICATIONS GERANT DES TYPES COMPLEXES DE DONNEES (applications multimédia)
- LIMITES DES SYSTEMES RELATIONNELS

NECESSITE DE LA DEFINITION D 'UN NOUVEL ENVIRONNEMENT DE PROGRAMMATION

■ TROIS REPOSES

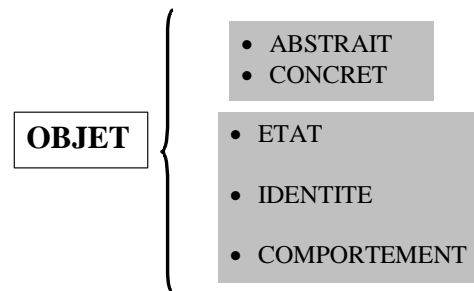
- BD ORIENTEES OBJET
- METHODES D 'ANALYSE
- SI ORIENTE OBJET

CONCEPTS FONDAMENTAUX DES MODELES OBJETS

OBJETS, TYPES, CLASSES

APPLICATIONS CLASSIQUES :
DUALITE DONNEES – TRAITEMENTS

APPROCHE OBJET :
APPLICATION = ENSEMBLE D'OBJETS
AYANT UN COMPORTEMENT ET
ECHANGEANT DES MESSAGES



- **ETAT** : ENSEMBLE DE VALEURS D 'ATTRIBUTS = CONTENU INFORMATIONNEL
- **IDENTIFIANT** : IDENTIFIE DE MANIERE UNIQUE UN OBJET ET N 'EST NI MODIFIABLE NI REUTILISABLE (INTERNE AU SYSTEME).
- **COMPORTEMENT**: DIFFERENTS MESSAGES AUXQUELS L 'OBJET PEUT REAGIR.

CONCEPTS FONDAMENTAUX DES MODELES OBJETS



OBJETS, TYPES, CLASSES

- **IDENTIFIANT D 'UN OBJET :**
 - PROPRIETES D 'INVARIANCE ET D 'UNICITE
 - NON GERE PAR LE PROGRAMMEUR

INITIALEMENT, L 'OID (OBJECT IDENTIFIER) ETAIT UN MECANISME DE DESIGNATION ET DE RESTITUTION : **OID PHYSIQUE.**

DANS LE DOMAINE DES BD, L 'OID REpond A UN MECANISME DE SELECTION D 'OBJETS REPERTORIES VIA DES PREDICATS DE SELECTION. LES MECANISMES D 'ACCES AGISSENT SUR UN RESEAU D 'IDENTIFIANTS : **OID LOGIQUE.**



CONCEPTS FONDAMENTAUX DES MODELES OBJETS

OBJETS, TYPES, CLASSES

IDENTIFIANT D 'UN OBJET

PROPRIETES D 'IDENTITE ET D'EGALITE

IDENTITE DE DEUX OBJETS : DEUX OBJETS SONT IDENTIQUES S 'ILS ONT MEME IDENTIFIANT.

EGALITE DE DEUX OBJETS : DEUX OBJETS SONT EGAUX SI LEURS ETATS SONT EGAUX.

INTERET :

- EXISTENCE DE L 'OBJET INDEPENDEMMENT DE SON ETAT. ON PEUT COMPRENDRE LA SEMANTIQUE DE MODIFICATION DE L 'OBJET COMME DES CHANGEMENTS D 'ETATS.
- DEUX OBJETS DIFFERENTS PEUVENT AVOIR LE MEME ETAT.

CONCEPTS FONDAMENTAUX DES MODELES OBJETS

OBJETS, TYPES, CLASSES

■ IDENTIFIANT D 'UN OBJET : REMARQUES

- L 'OBJET CONSERVE SON IDENTITE ET SES RELATIONS EVENTUELLES AVEC D 'AUTRES OBJETS NE SONT PAS REMISES EN CAUSE LORS DE CHANGEMENTS D 'ETATS (propriété d 'identité).
- DANS LE MODELE RELATIONNEL, IL EST NECESSAIRE DE PREVOIR UN IDENTIFIANT ARTIFICIEL AFIN QUE LES RELATIONS AVEC LES AUTRES OBJETS NE SOIENT PAS REMISES EN CAUSE.
- DEUX OBJETS DISTINCTS PEUVENT ÊTRE CARACTERISES PAR LE MEME ETAT (propriété d'égalité).

Exemple : Id1 : [type: A3, Marque : Audi , Année : 2019]

Id2 : [type : A3f, Marque : Audi, Année : 2019]

CONCEPTS FONDAMENTAUX DES MODELES OBJETS

OBJETS, TYPES, CLASSES

■ **LE COMPORTEMENT**

- IL REPRESENTE LA FACON DONT CELUI-CI REAGIT AUX MESSAGES QUI LUI SONT ENVOYES

- $\text{COMPORTEMENT} = \text{CONSTRAINTES} + \text{OPERATIONS}$
 - **CONSTRAINTES : ETATS POSSIBLES ET CHANGEMENTS D 'ETATS POSSIBLES DE L 'OBJET**

 - **OPERATIONS : MANIPULATION DES ETATS DE L 'OBJET = INTERFACE DE L 'OBJET, c-à-d L 'OBJET N 'EST ACCESSIBLE QU 'A TRAVERS SON INTERFACE.**

CONCEPTS FONDAMENTAUX DES MODELES OBJETS

OBJETS, TYPES, CLASSES

■ TYPE ABSTRAIT DE DONNEES (TAD)

UN TYPE DE DONNEES ABSTRAIT PERMET DE DEFINIR DES OBJETS EN TERMES DE STRUCTURE DE DONNEES (ATTRIBUTS) ET DE COMPORTEMENT (METHODES).

UN TAD EST ENTIEREMENT DEFINI PAR :

**INTERFACE OU
PARTIE PUBLIQUE**

**- SPECIFICATION QUI DECRIT LES OPERATIONS
VISIBLES PERMETTANT LA MANIPULATION DES
OBJETS APPARTENANT AU TYPE**

**PARTIE PRIVEE OU
CACHEE**

**- REPRESENTATION DE LA STRUCTURE DE
DONNEES

- IMPLEMENTATION QUI DECRIT LA
REALISATION DES OPERATIONS**

EXEMPLE : DESCRIPTION D'UN TYPE ABSTRAIT DE DONNEES EN LANGAGE ADA

Package TDA-PILE **IS**

Type ELEMENT **IS NATURAL RANGE** 0..100

Type PILE **IS** (Taille : Positive : 20) **IS Private**

Procedure DEPILER (P: IN PILE; ELEM : OUT ELEMENT)

Procedure EMPILER (P : IN PILE; ELEM : IN ELEMENT)

Function PILE-VIDE (P : PILE) return Boolean ;

Function PILE-PLEINE (P: PILE) return Boolean ;

VIDE, PLEINE : **exception** ;

PRIVATE

Définition de la structure interne du type PILE (tableau, liste,..)

END TDA-PILE

Package Body TDA-PILE **IS**

....

END

**Partie
spécification
publique**

**Partie
spécification
privée**

**Partie
implémentation
privée**

LIEN D 'HERITAGE

UTILISATION DES CONCEPTS DE SPECIALISATION / GENERALISATION POUR METTRE EN EVIDENCE DES SOUS-CLASSES AU SEIN D 'UNE CLASSE DITE GENERIQUE OU SUPERCLASSE.

- LES LIENS ENTRE SUPERCLASSE ET SOUS-CLASSE SONT APPELES LIENS D 'HERITAGE.
- SI UNE SOUS-CLASSE **B HERITE** D 'UNE SUPER CLASSE **A** ALORS :
 - B HERITE DE LA STRUCTURE ET DU COMPORTEMENT DE A : PROPRIETES IMPORTEES.
 - B PEUT AVOIR DES PROPRIETES SPECIFIQUES (PROPRIETES LOCALES) QUI SONT :
 - COMPLEMENTS (STRUCTURELS ET/OU COMPORTEMENTALS)
 - REDEFINITIONS (SURCHARGE)
- L 'HERITAGE PEUT ETRE :
 - SIMPLE : UNE CLASSE HERITE D 'UNE SEULE SUPERCLASSE
 - MULTIPLE : UNE CLASSE HERITE DE PLUSIEURS SUPERCLASSES
- INTERET DE L 'HERITAGE
 - SPECIFICATION PAR NIVEAUX D 'ABSTRACTION
 - CONCEPTION INCREMENTALE
 - REUTILISATION DE SPECIFICATION DE CODE.

- ◉ La suppression d'un objet généralisé implique la suppression de ses objets spécialisés
- ◉ Propriété d'inclusion des identités : l'ensemble des identités de l'objet spécialisé est inclus dans celui de l'objet généralisé
- ◉ Propriété d'inclusion des cycles de vie

Personne a: _____

Etudiant b: _____

Salarie s : _____

- Propriété d'inclusion des schémas de classe

Schéma (A) contient schéma (B)

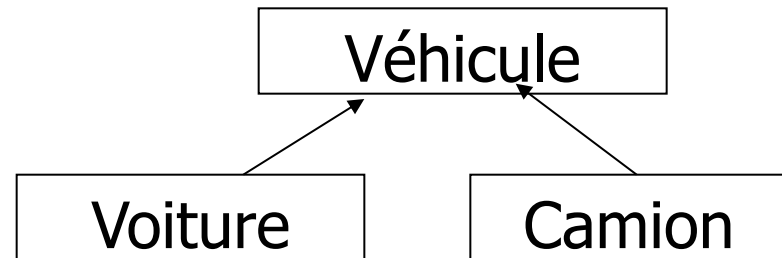
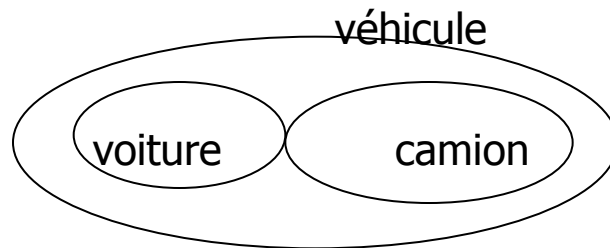
↓ ↓

Spécialisée Généralisée

LIEN DE GENERALISATION : CONTRAINTE D'HERITAGE

- Elle restreint les possibilités d'existence d'instances de classes spécialisées pour chaque instance d'une classe généralisée

- Disjonction**

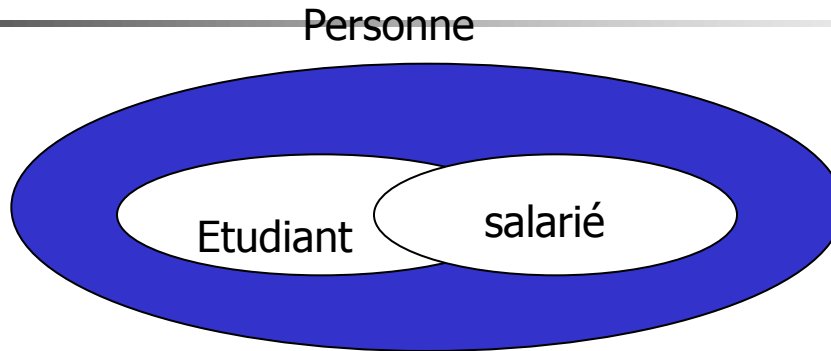


- Une contrainte de disjonction entre un ensemble de classes spécialisées exprime qu'à tout objet de la classe générale ne peut correspondre au plus qu'un seul objet spécialisée dans une de ces classes

Un véhicule est soit une voiture soit un camion soit autre chose

LIEN DE GENERALISATION : CONTRAINTE D'HERITAGE

Couverture

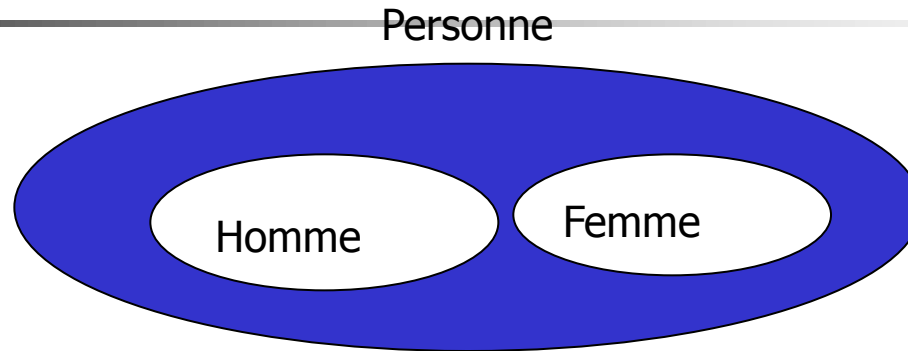


- Une contrainte de couverture entre un ensemble de classes spécialisées exprime qu'à tout objet de la classe généralisée doit nécessairement correspondre au moins un objet spécialisé appartenant à une des classes

Une personne est soit un étudiant soit un salarié soit les deux

LIEN DE GENERALISATION : CONTRAINTE D'HERITAGE

Partition



- ◉ A tout objet de la classe généralisée doit nécessairement correspondre exactement un objet spécialisée appartenant à une des classes
- ◉ C'est à la fois une disjonction et une couverture



AUTRES CONCEPTS DE L 'APPROCHE OO

- **LE POLYMORPHISME ET LA LIAISON DYNAMIQUE**

Le polymorphisme est la faculté d 'une opération à pouvoir s 'appliquer à des objets de classes différentes.

- **SURCHARGE** : redéfinition d 'une opération avec un code différent
- **LIAISON DYNAMIQUE** : mécanisme de sélection de code d 'une opération à l 'exécution, en fonction de la classe d 'appartenance de l 'objet receveur



Généralités

- Avantages qui militent en faveur de l'orienté objet:
 - **La définition en termes d'objets** facilite la **construction de composants logiciels** ressemblant plus au domaine d'application.
 - **L'encapsulation, l'utilisation de messages** encourage une **conception modulaire**; la réalisation d'un objet ne dépend plus des détails internes d'un autre objet mais seulement de la manière dont le premier objet répond aux messages qu'il reçoit.
 - **L'usage de classes et d'héritage** motive le développement **de composants modulaires réutilisables et évolutifs.**



Généralités: modèles de données et SGBD orientés objet

■ Définitions

MDOO : Un modèle de données qui capture la sémantique des objets supportée en programmation objet.

BDOO : Une collection persistante et partageable d'objets définis par un MDOO.

SGBDOO : Le gestionnaire d'une BDOO



Généralités: modèles de données et SGBD orientés objet

- Fonctionnalités d'un SGBDOO les plus générales

A partir de « **orientation objet = types de données abstraits + identité d'objet + héritage** » alors:

- Fonctionnalités d'un SGBD classique **plus,**
- Il supporte la notion d'identité des objets,
- Il assure l'encapsulation,
- Il gère les objets complexes

Même si l'héritage est très utile, il n'est pas essentiel à la définition d'un SGBDOO



Généralités : origine du modèle OO

Systèmes de BD traditionnels:

**Persistance,
partage,
Transaction,
contrôle de
concurrence**

**Sécurité,
intégrité,
requêtes**

**Modèles de
données
sémantiques:
Généralisation,
agrégation**

Programmation OO:

**Identité des
objets,
encapsulation,
héritage, types et
classes, méthodes,
objets complexes,
polymorphisme,
extensibilité.**

**Exigences
spéciales:
Gestion de
versions,
Evolution
de
schémas**



```
graph TD; A["Systèmes de BD traditionnels:  
Persistance, partage,  
Transaction, contrôle de concurrence  
Sécurité, intégrité, requêtes"] --> D["Modèle de données OO"]; B["Modèles de données sémantiques:  
Généralisation, agrégation"] --> D; C["Programmation OO:  
Identité des objets,  
encapsulation, héritage, types et classes, méthodes,  
objets complexes, polymorphisme, extensibilité."] --> D; E["Exigences spéciales:  
Gestion de versions,  
Evolution de schémas"] --> D;
```

Modèle de données OO



Modélisation Objet avec UML

Diagrammes de classes

Diagrammes d'Etat/Transition



Bibliographie UML

UML 2.0, guide de référence

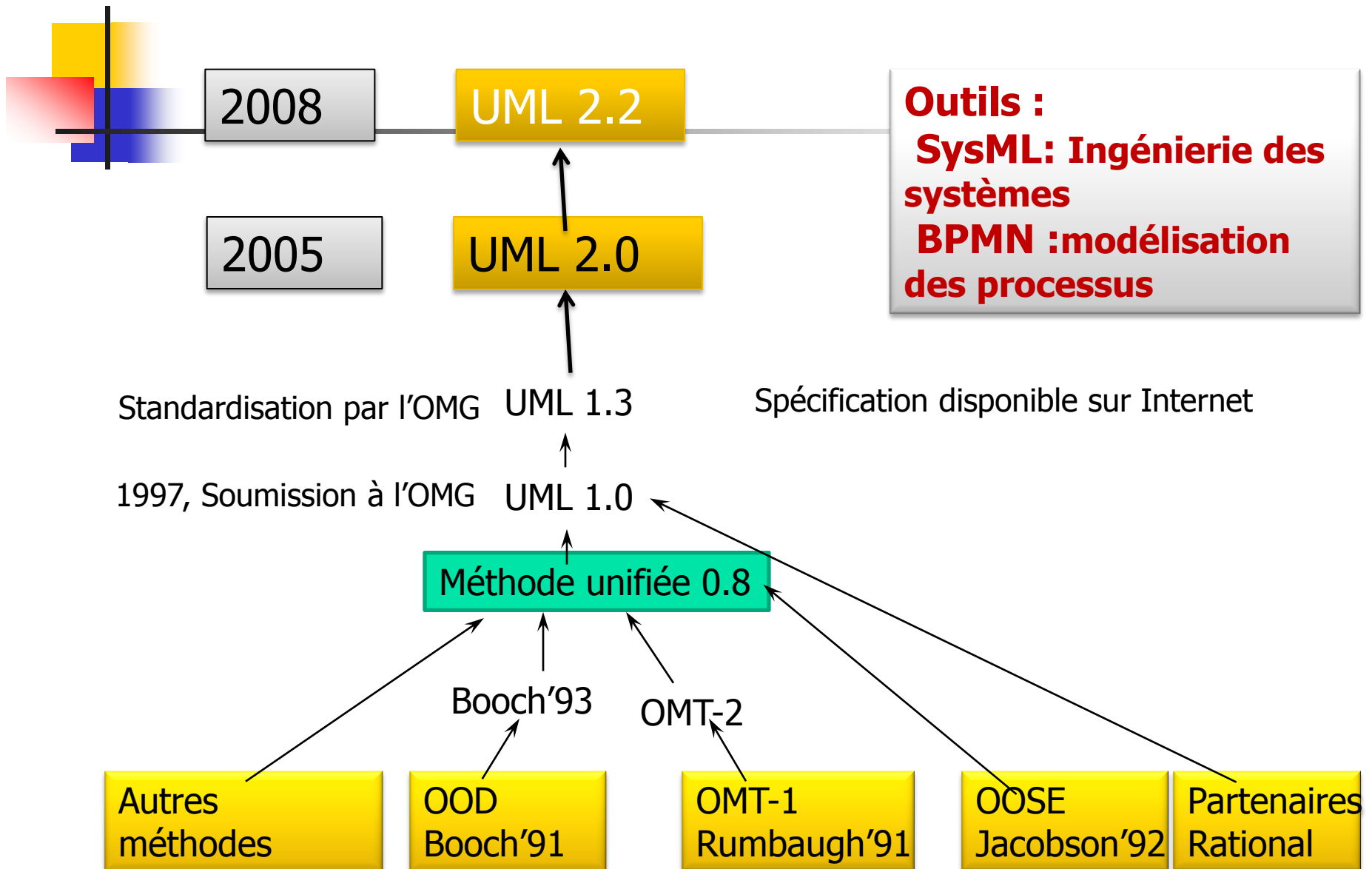
James Rumbaugh, Ivar Jacobson, Grady Booch Editions
Campus Press (2005)

UML 2.0

Benoît Charoux, Aomar Osmani, Yann Thierry-Mieg Editions
Pearson, Education France (2008)

UML 2.0 Superstructure et UML 2.0 Infrastructure
OMG (Object Management Group) www.uml.org (2004).

Principales étapes de la définition d'UML



GENERALITES



UML : Langage graphique de modélisation objet résultat des tentatives de l'unification des méthodes OOD (G. Booch), OMT (J. Rumbaugh) et OOSE (I. Jacobson)

Rapprochement d'OMT et OOD


Garder la puissance d'expression de la sémantique d'OMT à travers les concepts d'association entre objets, et l'expression de leur comportement à travers la notion de diagramme d'État- Transition
Sélectionner de OOD, les concepts de modules (sous systèmes) et la notion de flots de messages

Rapprochement avec OOSE

Utilisation des « Use case » (cas d'utilisation) comme moyen d'expression des besoins des utilisateurs dans la phase d'analyse.

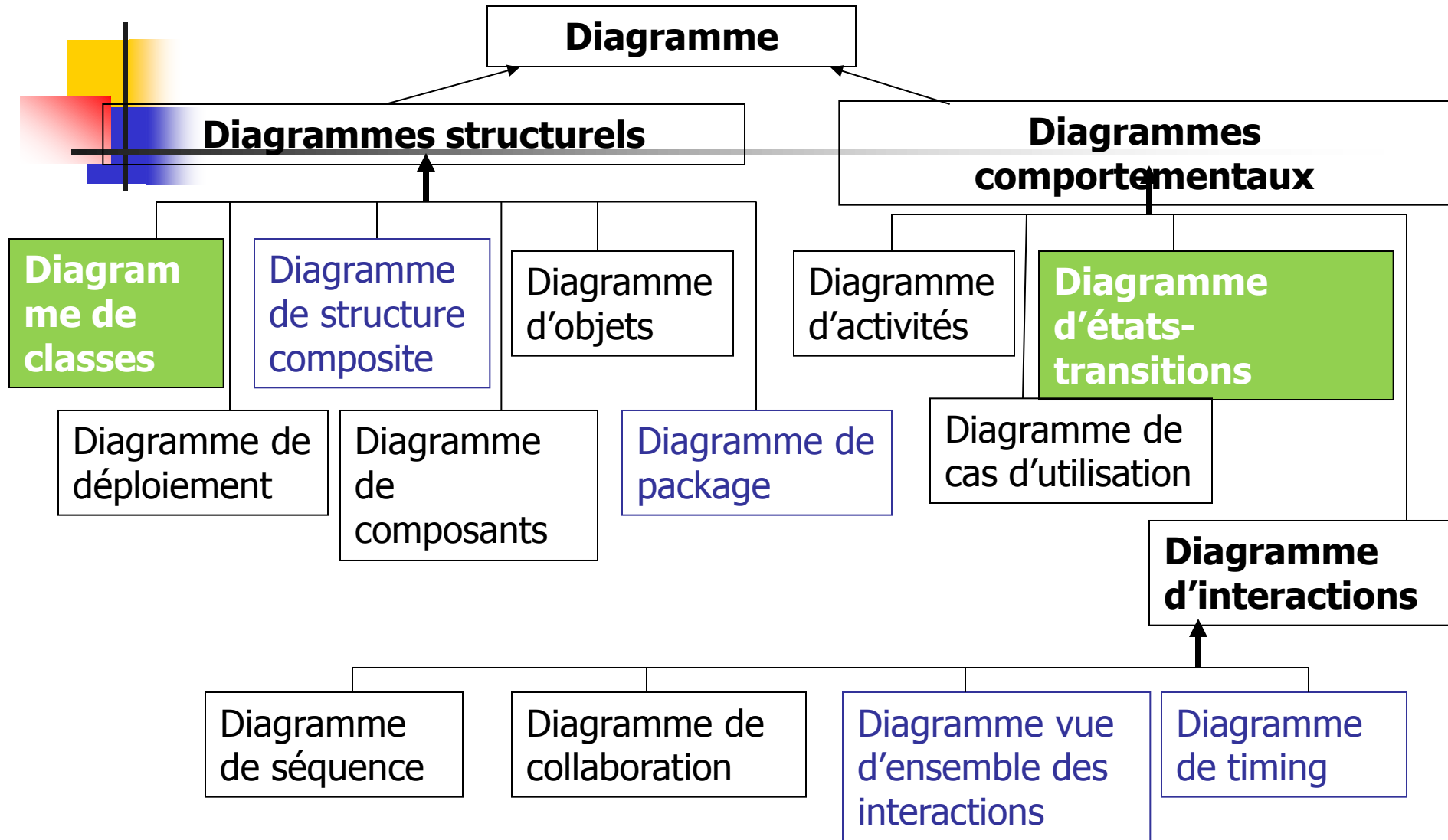
UML devient une norme de l'OMG (Object Management Group) en Novembre 1997.

Objectifs d'UML

- 
- **Proposer un langage visuel de modélisation**
 - **Utilisable par toutes les méthodes**
 - **Adapté à toutes les phases du développement**
 - **Compatible avec toutes les techniques de réalisation**
 - **Proposer des mécanismes d'extension et de spécialisation pour pouvoir étendre les concepts de base**
 - **Être indépendant des langages particuliers de programmation**
 - **Proposer une base formelle pour comprendre le langage de modélisation**
 - **Encourager l'application des outils OO**
 - **Supporter les concepts de développement de haut-niveau tels que les *frameworks*, les *patterns* et les *composants***

- 
-
- GENERALITES
 - **DIAGRAMME DE CLASSE**
 - **DIAGRAMME D'ETAT-TRANSITION**

LES DIAGRAMMES D'UML





LES DIAGRAMMES D'UML

Les diagrammes structurels permettent de représenter l'aspect statique

D'éléments d'analyse **classes, structure composite** et **objets**

D'éléments physiques **composants** et **déploiement** et **package**

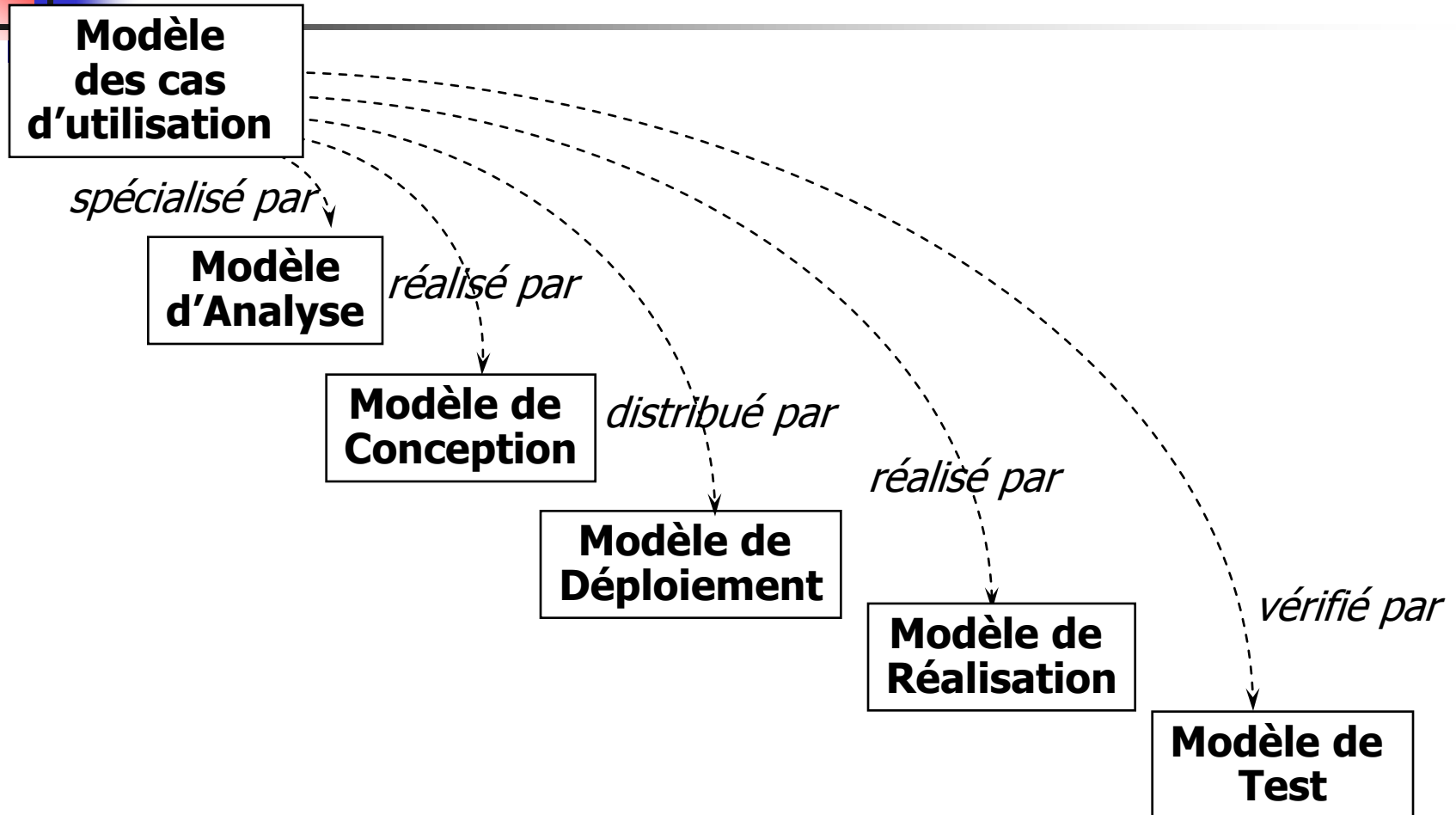
Les diagrammes comportementaux permettent de représenter dans l'étape d'analyse :

Les interactions entre les objets : **diagrammes de séquence et collaboration (communication), vue d'ensemble des interactions et timing**

Le comportement des objets : **diagrammes d'état-transition et activité**

Les diagrammes de cas d'utilisation même si on les classent dans les diagrammes comportementaux sont utilisés tout au long du processus de développement d'UML.

Processus d'ingénierie sous-jacent : dirigé par les modèles



Relations entre diagrammes et étapes du processus

Découverte des besoins :

- **Diagramme de cas d'utilisation :** décrit les fonctions du système selon le point de vue de ses utilisateurs futurs
- **Diagramme de séquence :** représentation des interactions temporelles entre objets dans la réalisation d'une interaction homme-système

Analyse :

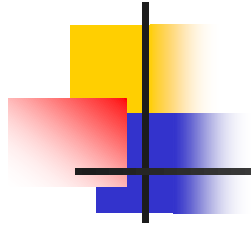
- **Diagramme de classes :** structure des données du système définie comme un ensemble de relations entre classes
- **Diagramme d'objets :** illustration des objets et de leurs relations
- **Diagramme de collaboration :** représentation des interactions entre objets
- **Diagramme d'états-transitions :** représentation du comportement des objets d'une classe en termes d'états et de transitions d'états
- **Diagramme d'activités :** structure d'une opération en actions

Relations entre diagrammes et étapes du processus



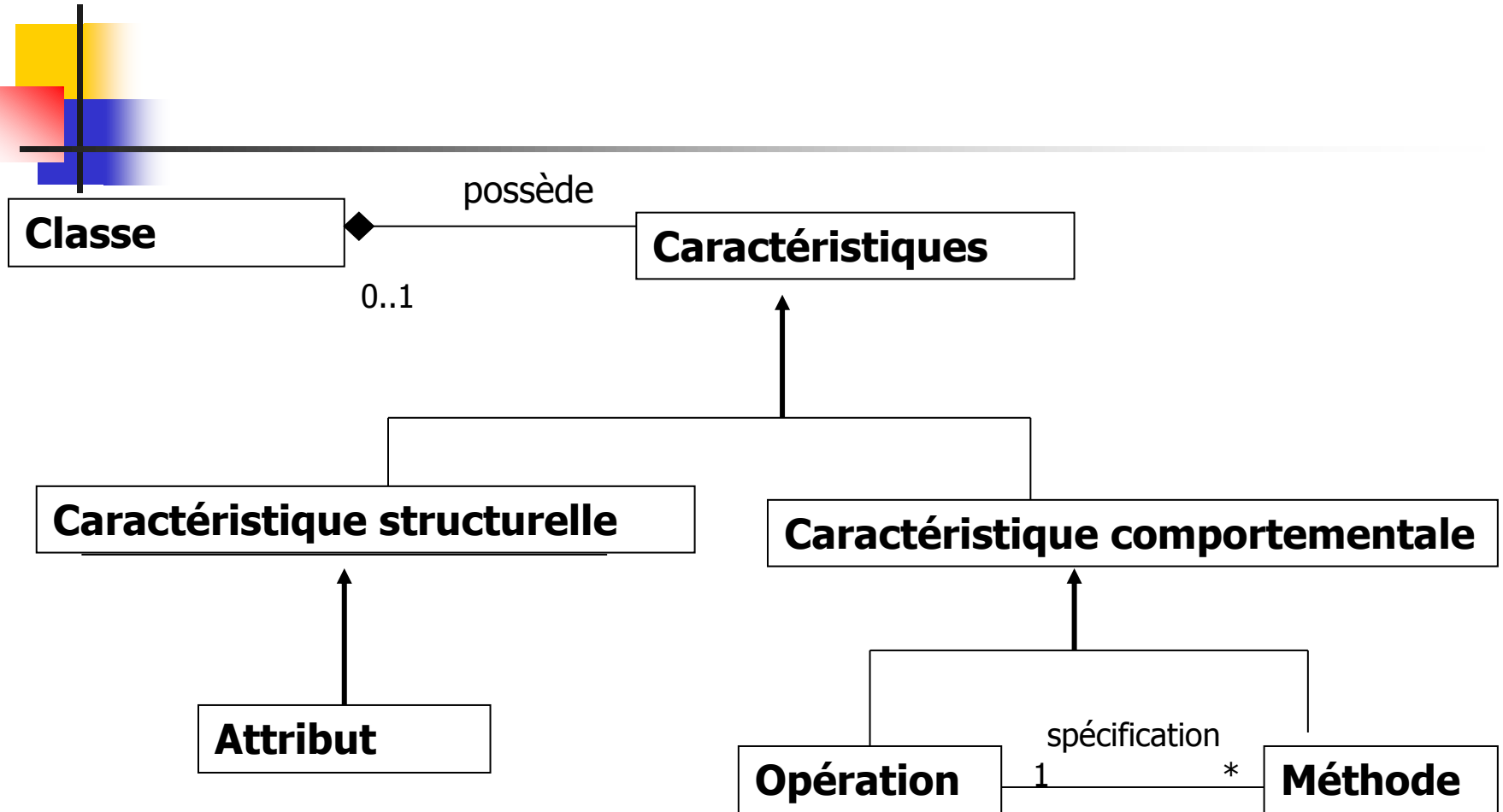
Conception/déploiement/réalisation :

- ***Diagramme de séquence***: représentation des interactions temporelles entre objets dans la réalisation d'une opération
- ***Diagramme de déploiement*** : description du déploiement des composants sur les dispositifs matériels
- ***Diagrammes de composants*** : architecture des composants physiques d'une application



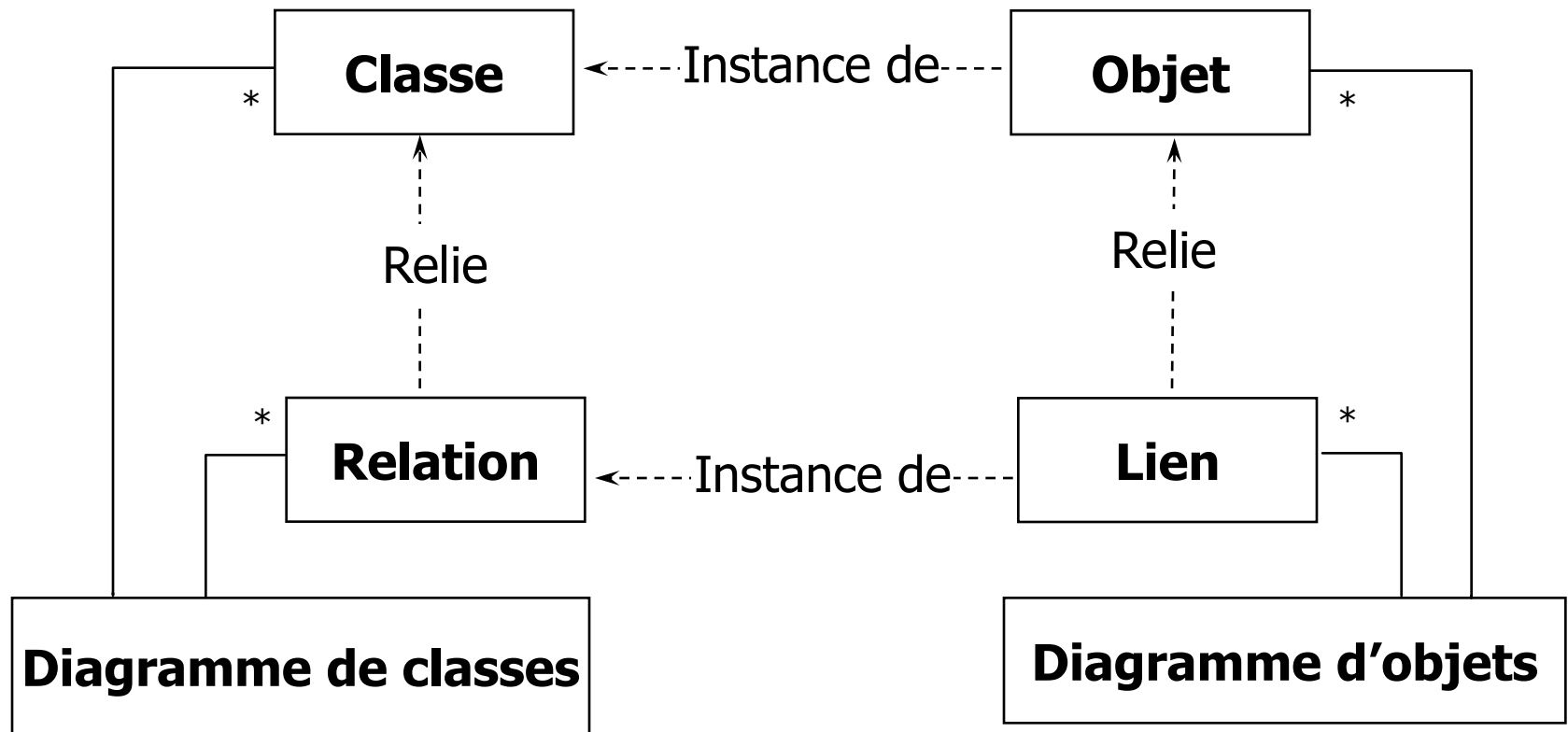
Diagrammes de classes et diagrammes d'objets

LES DIAGRAMMES DE CLASSES : le méta modèle



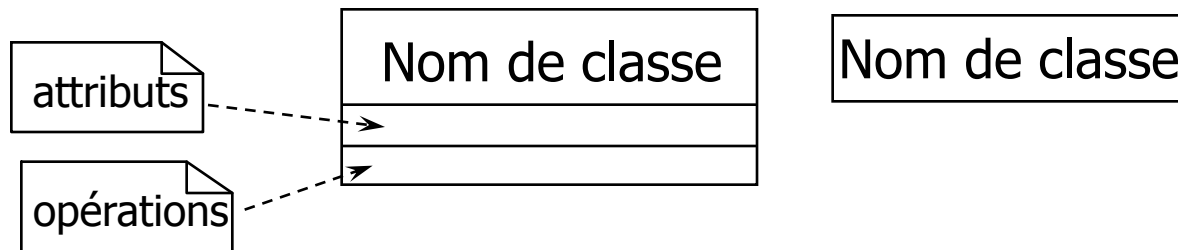
Diagrammes de classes et d'objets

Un diagramme de classes représente la structure du système en termes de classes et de relations entre ces classes; Un diagramme d'objets illustre les objets et leurs relations

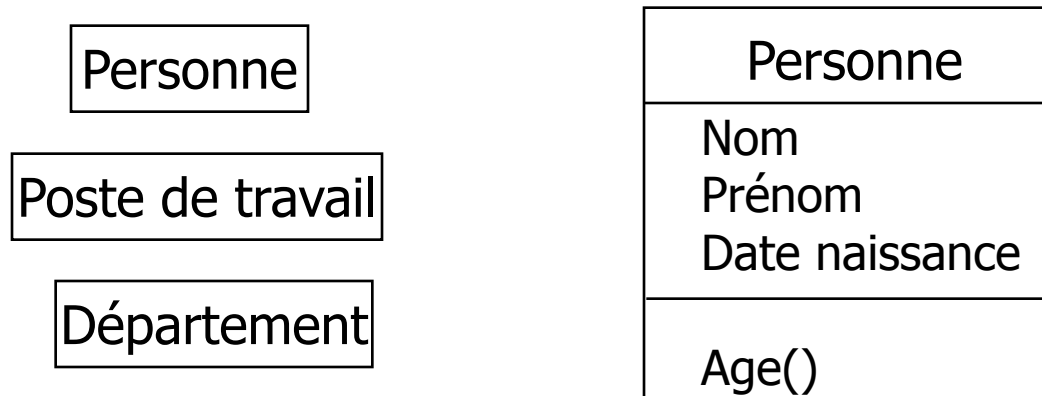


Classe

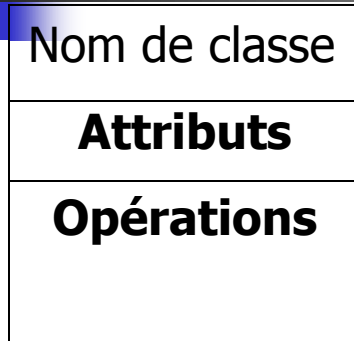
Une classe est une description abstraite d'un ensemble d'objets ayant des propriétés similaires, un comportement commun, des relations communes avec d'autres objets et des sémantiques communes



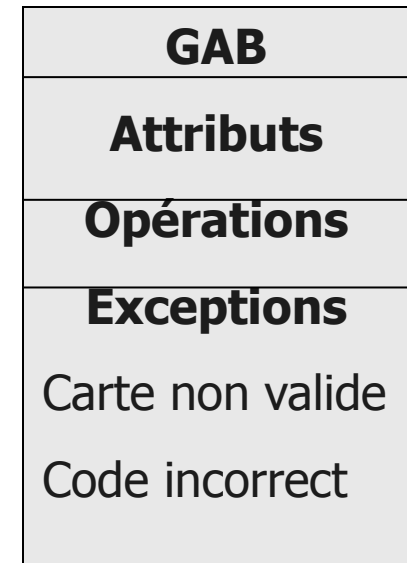
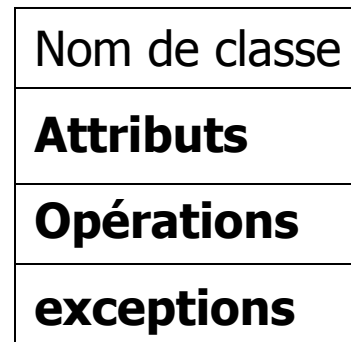
Exemples:



LES DIAGRAMMES DE CLASSES

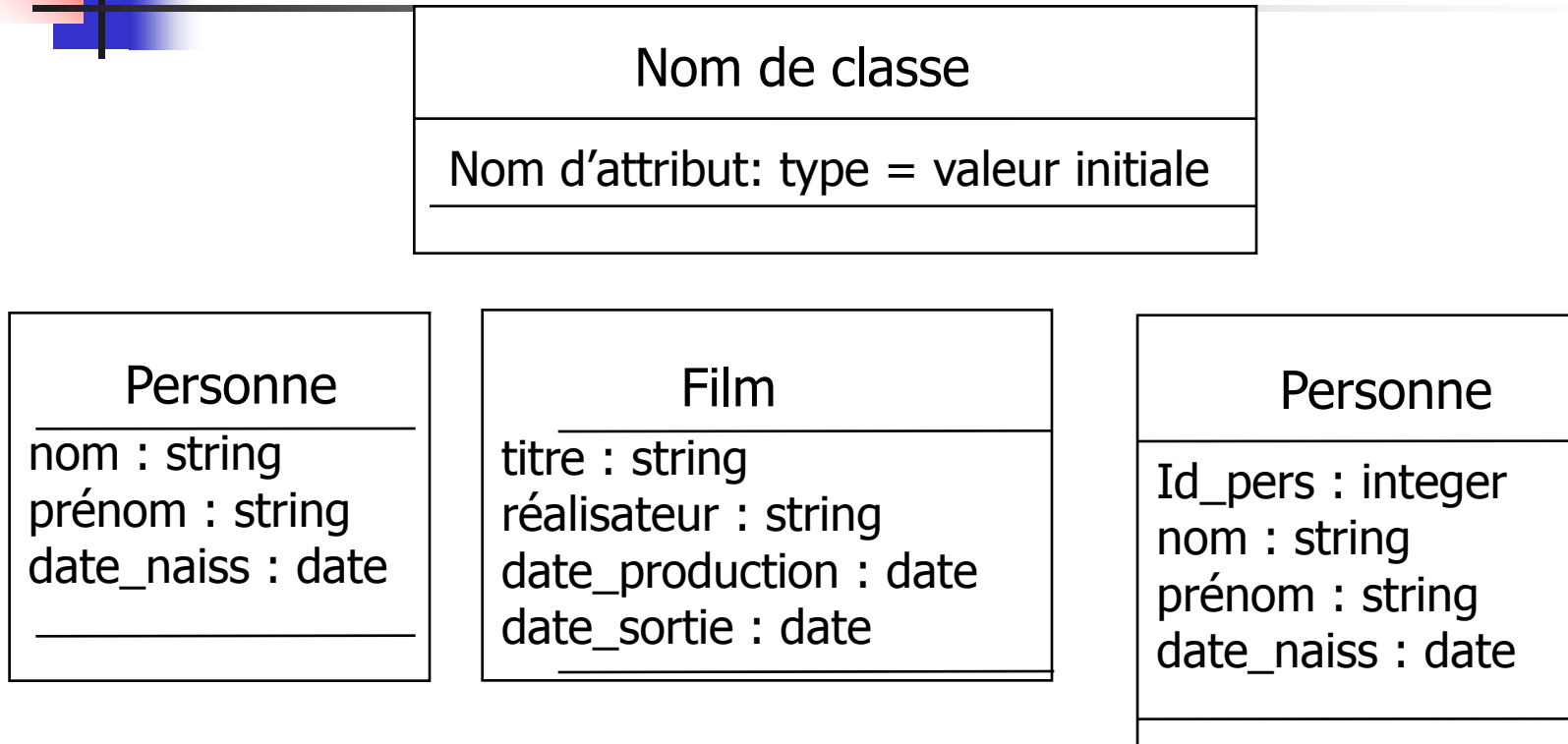


D'autres
compartiments
peuvent être ajoutés



Attributs de classe

Un attribut définit une propriété commune aux objets d'une classe



- Les noms d'attributs d'une classe sont uniques
- Chaque objet, instance d'une classe a sa propre identité indépendante des valeurs de ses attributs.

Attributs de classe



La syntaxe de description d'un attribut est la suivante :

Visibilité nom : type multiplicité = valeurParDéfaut {propriété}

- Seul le nom est indispensable
- Visibilité : public (+), privé (-)
- Propriété : propriétés supplémentaires comme {readOnly}

• *Exemple :*

- *nom: String [1] = « une valeur » {readOnly}*

Opérations de classe

Une opération définit une fonction appliquée à des objets d'une classe

Nom de classe
Nom d'opération(liste_arguments) : type_retour

Personne
nom : string adresse : string date_naiss : date
age () change_adresse()

Objet Géométrique
couleur : string position : integer
déplacer (deltat:Vecteur) sélectionner (p:Point):Booléen

Opérations de classe

La syntaxe de description d'une opération est la suivante:

**Visibilité Nom_opération (liste_de_paramètres) :
Type_Retourné {propriété}**

- **la visibilité** est (+) publique ou (-) privée
- Le **Nom_opération** est une chaîne de caractères
- La **liste_de_paramètres** est la liste de paramètres de l'opération: forme : **direction nom: type = valeurpar défaut**

Où **direction** peut être : **(in)** entrée (par défaut), **(out)** sortie, **(inout)**

- **Type_retourné** est le type de la valeur retournée si elle existe.
- **Propriété** indique les valeurs de propriétés qui s'appliquent à l'opération

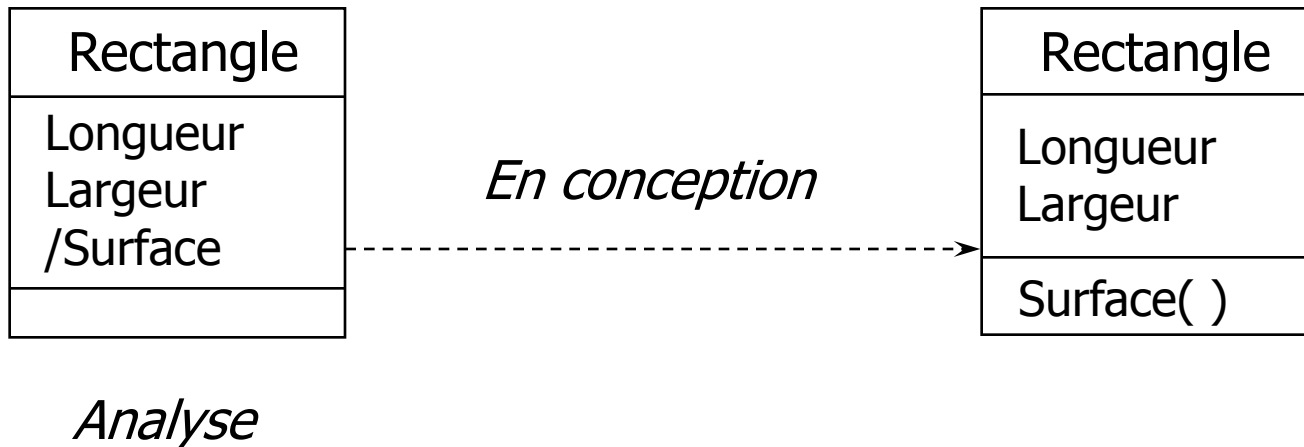
• **Exemple :**

+ soldeAu (date: Date): Monnaie

Attributs dérivés

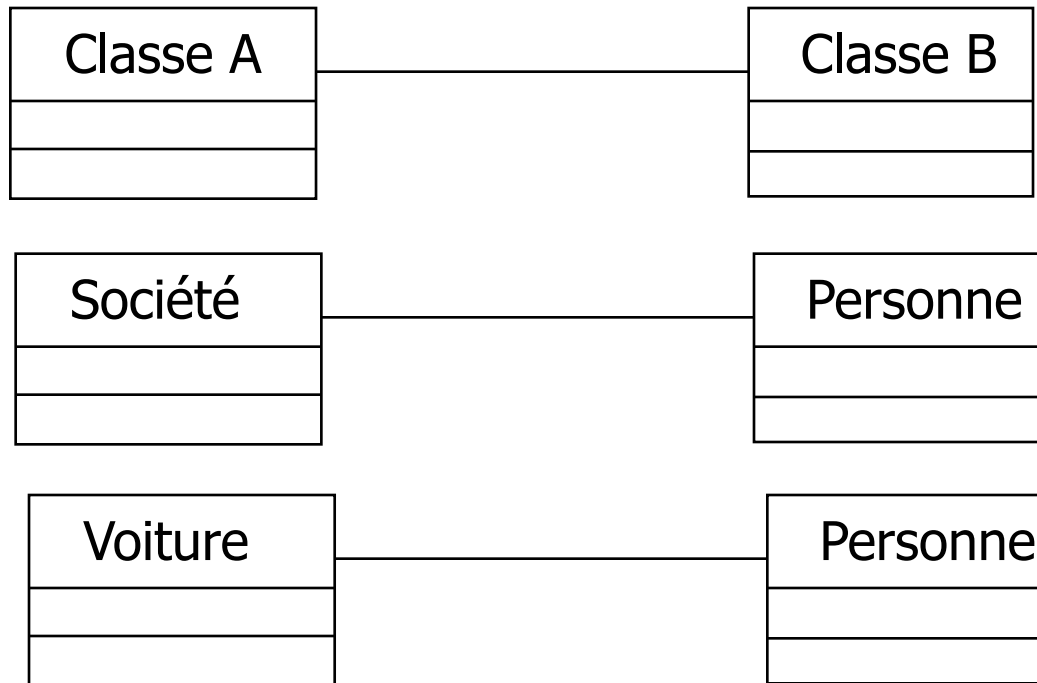
Les attributs dérivés ont des valeurs calculées à partir de celles d'autres propriétés

Exemple:



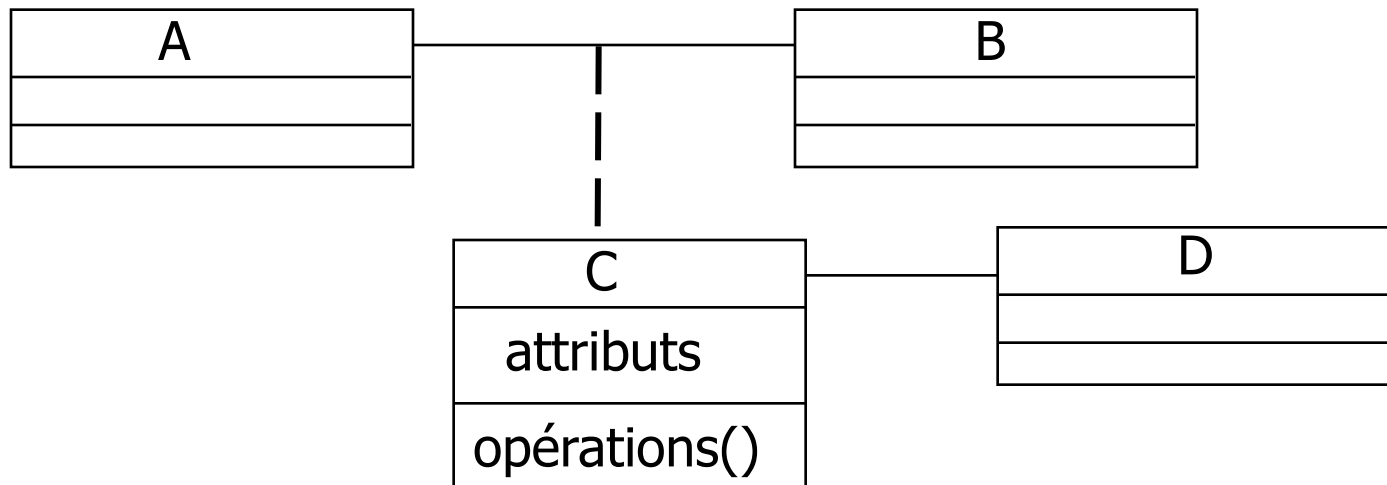
Association

Une association représente une classe de relations structurelles entre classes d'objets

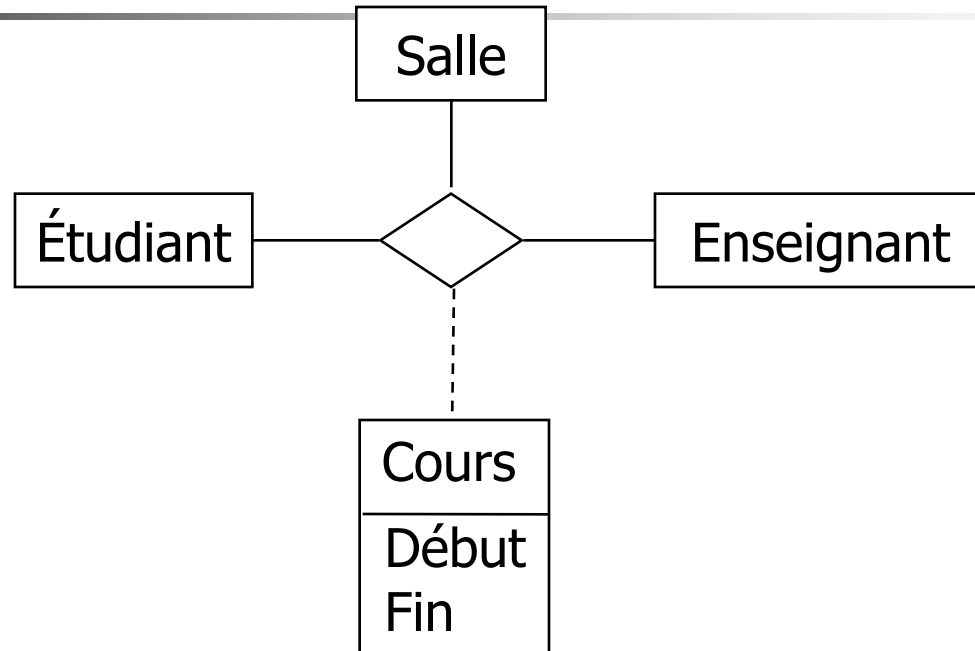


Classe association

- Une association peut être représentée (réifiée) par une classe appelée ***classe associative*** ou ***classe-association***.
- Utile par exemple, lorsque l'association a des attributs ou bien qu'on souhaite lui attacher des opérations.



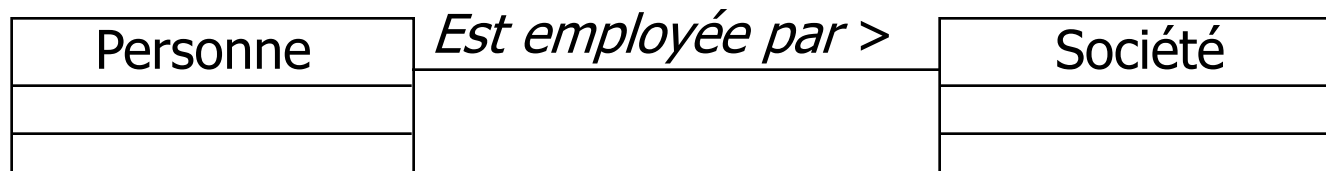
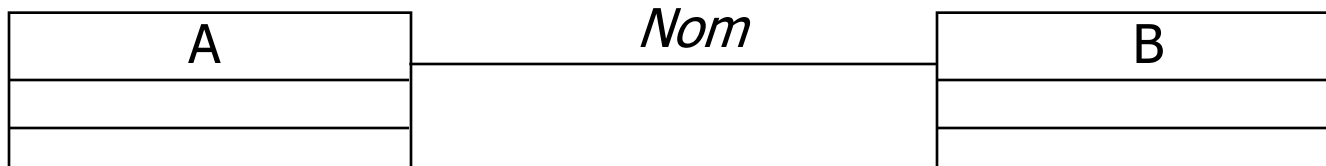
Association n-aire



L' association ternaire entre salle, étudiant et enseignant est réifiée comme une classe cours ayant deux attributs, Début et Fin

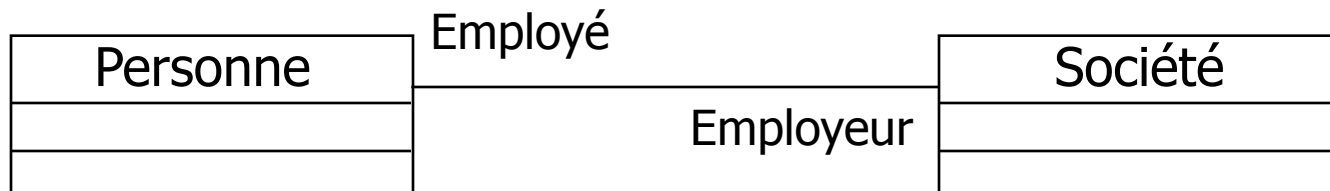
Nommage des associations

- Le nom d'une association apparaît en *Italique* au milieu de la ligne qui symbolise l'association
- L'usage recommande de nommer les associations par une forme verbale, soit active, soit passive



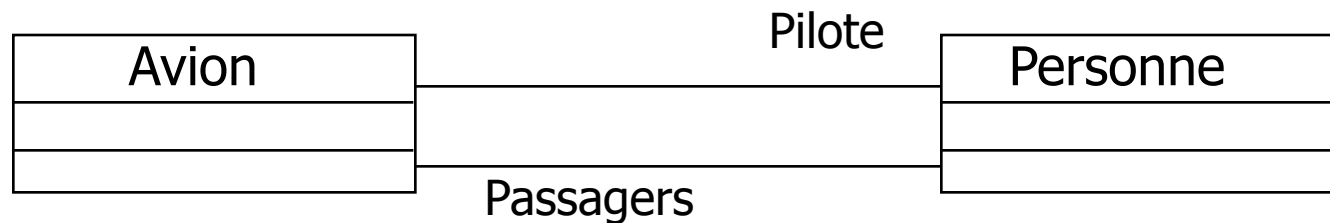
Nommage des rôles

- Chaque association binaire possède 2 rôles
- Le rôle décrit comment une classe intervient dans une association
- Le nommage des associations et le nommage des rôles ne sont pas exclusifs l'un de l'autre



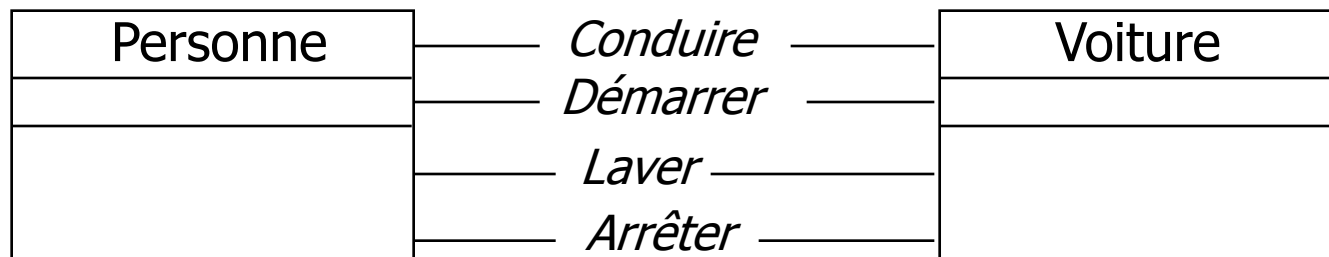
Nommage des rôles

Le nommage des rôles prend tout son intérêt lorsque il y a plusieurs associations entre les deux mêmes classes.



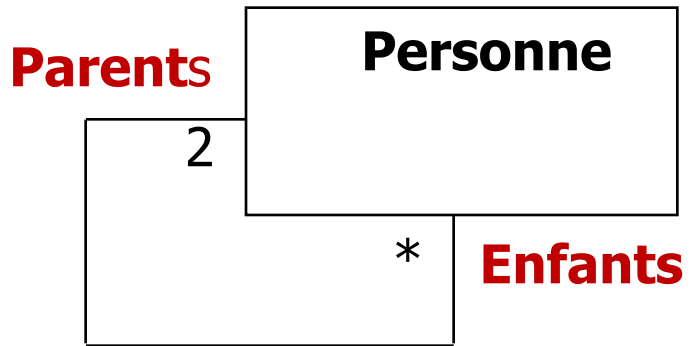
- ❑ La présence d'un grand nombre d'associations entre 2 classes est suspecte. Il est souvent le signe d'une mauvaise modélisation

Exemple :



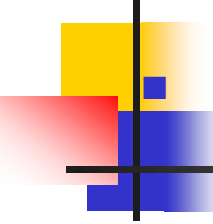
*

Association réflexive



*Le nommage des rôles est indispensable
à la clarté du diagramme*

Multiplicité des associations

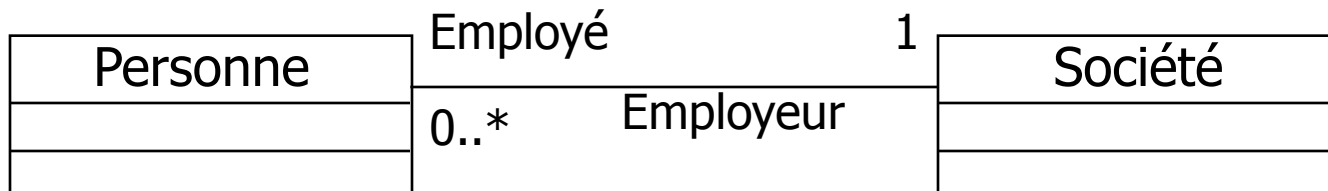


■ La multiplicité est une information portée par le rôle, qui quantifie le nombre de fois où un objet participe à une instance de relation

1	Un et un seul
0 .. 1	Zéro ou un
M .. N	De M à N (entiers naturels)
*	De zéro à plusieurs
0 .. *	De zéro à plusieurs
1 .. *	De un à plusieurs

Multiplicité des associations

Les valeurs de multiplicité expriment des contraintes à un instant donné



*Chaque personne travaille pour une seule société,
chaque société emploie zéro ou plusieurs personnes*

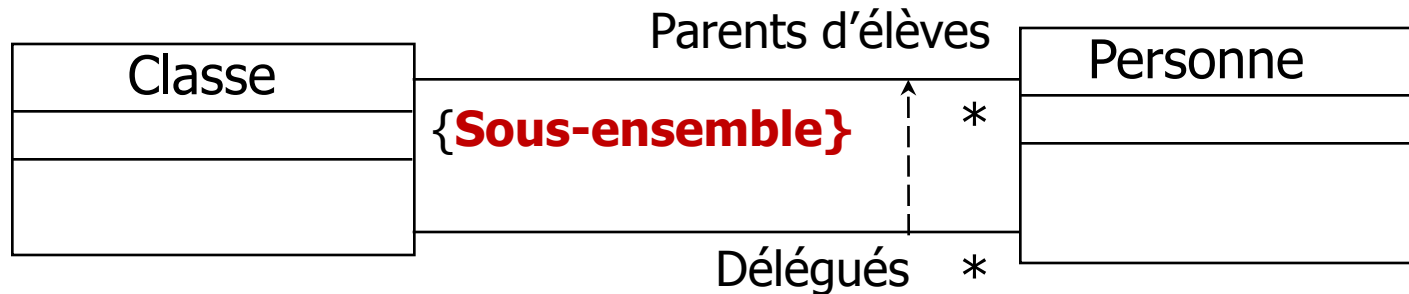
Contraintes sur les associations

- Les contraintes sont représentées sur les diagrammes par des expressions placées entre accolades
- La contrainte **{ordonnée}** placée sur le rôle définit une relation d'ordre sur les objets de la collection

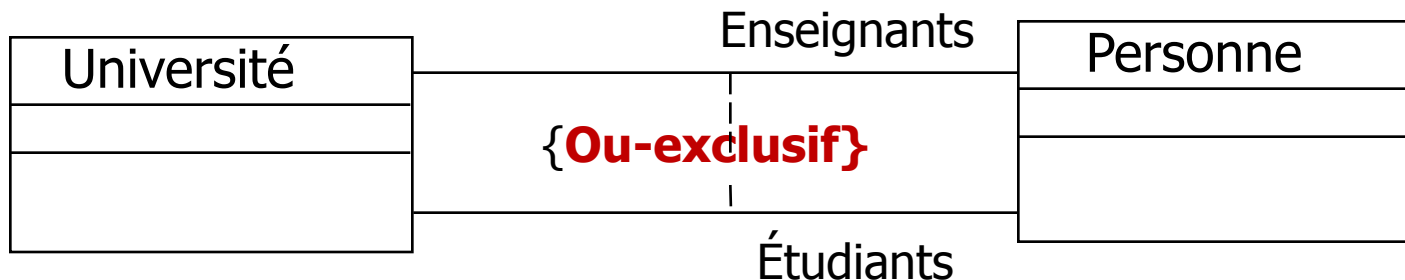


Contraintes sur les associations

- La contrainte **{Sous-ensemble}** indique qu'une collection est incluse dans une autre collection

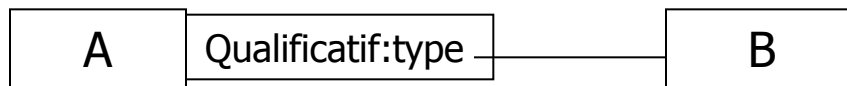


- La contrainte **{Ou-exclusif}** précise que, pour un objet donné, une seule association parmi un groupe d'associations est valide



Qualification des associations

- La qualification (*restriction*) d'une association consiste à sélectionner un sous-ensemble d'objets parmi l'ensemble des objets qui participent à une association

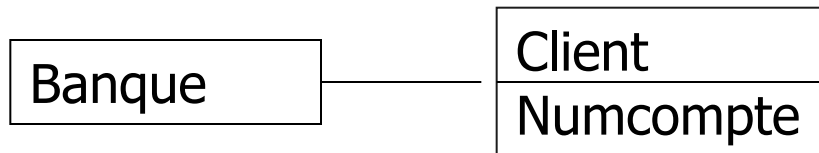
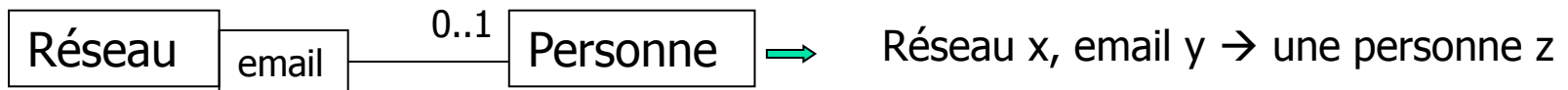
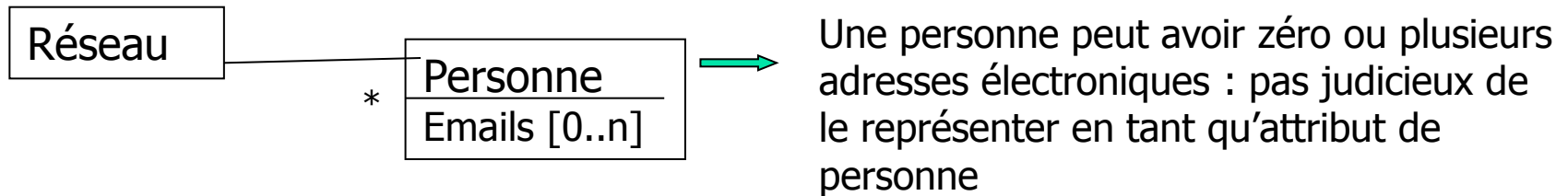


(Instance A, valeur du qualificatif)
identifie un sous-ensemble des instances
de B

Qualification des associations

- Une restriction réduit le nombre d'instances qui participent à une association

Exemples

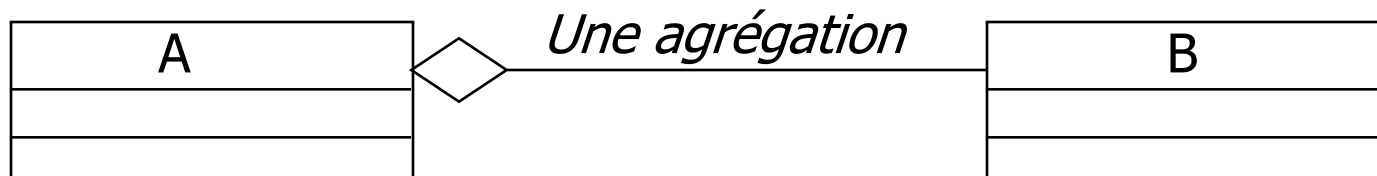


Banque, numcompte → Un client

Agrégation

Une agrégation représente une association non symétrique dans laquelle une des extrémités joue un rôle prédominant par rapport à l'autre

- Les propriétés suivantes suggèrent une agrégation:
 - une classe B 'fait partie' d'une classe A
 - les valeurs d'attributs de la classe B se propagent dans les valeurs d'attributs de la classe A
 - une action sur la classe A implique une action sur une la classe B;
 - les objets de la classe B sont subordonnés aux objets de la classe A



Agrégation

- L'agrégation peut être multiple, comme l'association



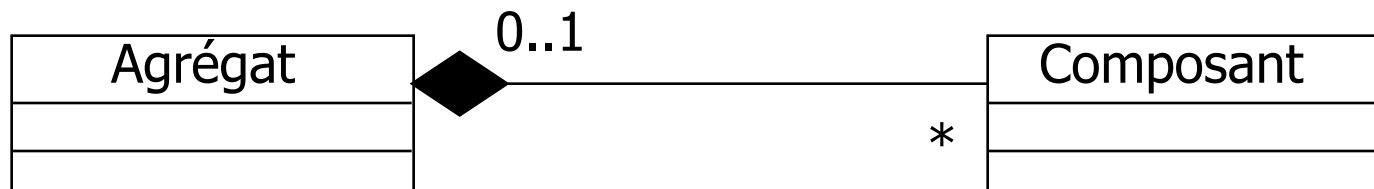
**En tant que 'propriétaire' une personne est un agrégat d'immeubles!
Les immeubles dont elle est propriétaire font 'partie de' la description
d'une personne**

Composition

La composition est une forme particulière d'agrégation

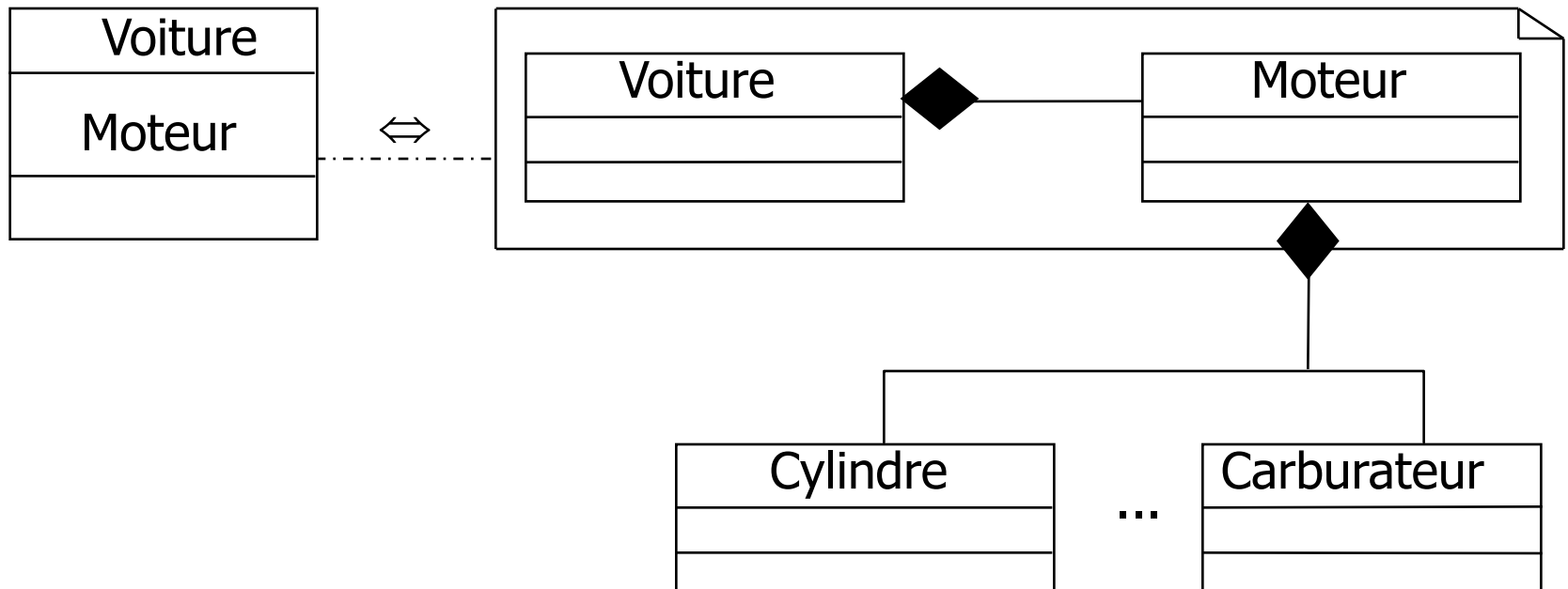
Le composant est physiquement contenu dans l'agrégat

- La composition implique une contrainte sur la valeur de la multiplicité du côté de l'agrégat: elle ne peut prendre que les valeurs 0 ou 1
- La valeur 0 du côté du composant correspond à un attribut non renseigné



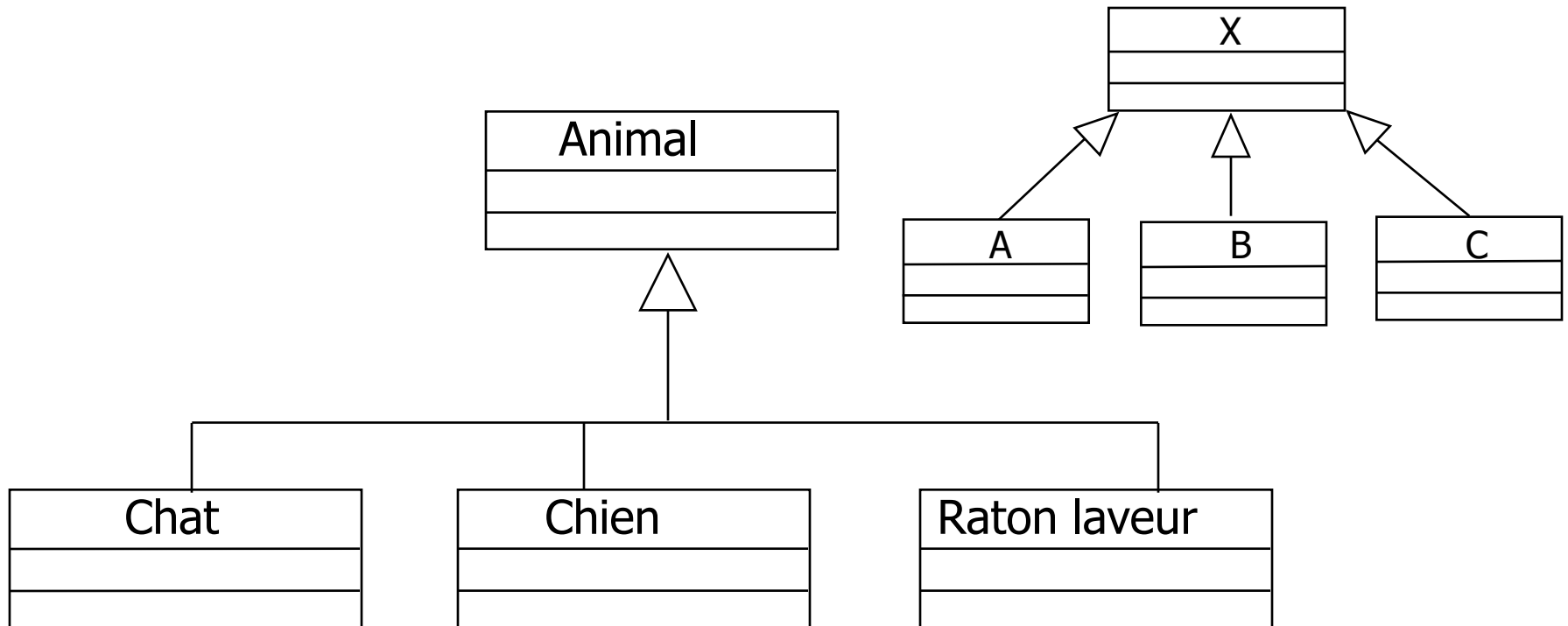
Composition

- La composition peut être modélisée au moyen d'attributs
- La notation par composition doit être retenue lorsqu'un attribut participe à d'autres relations dans le modèle

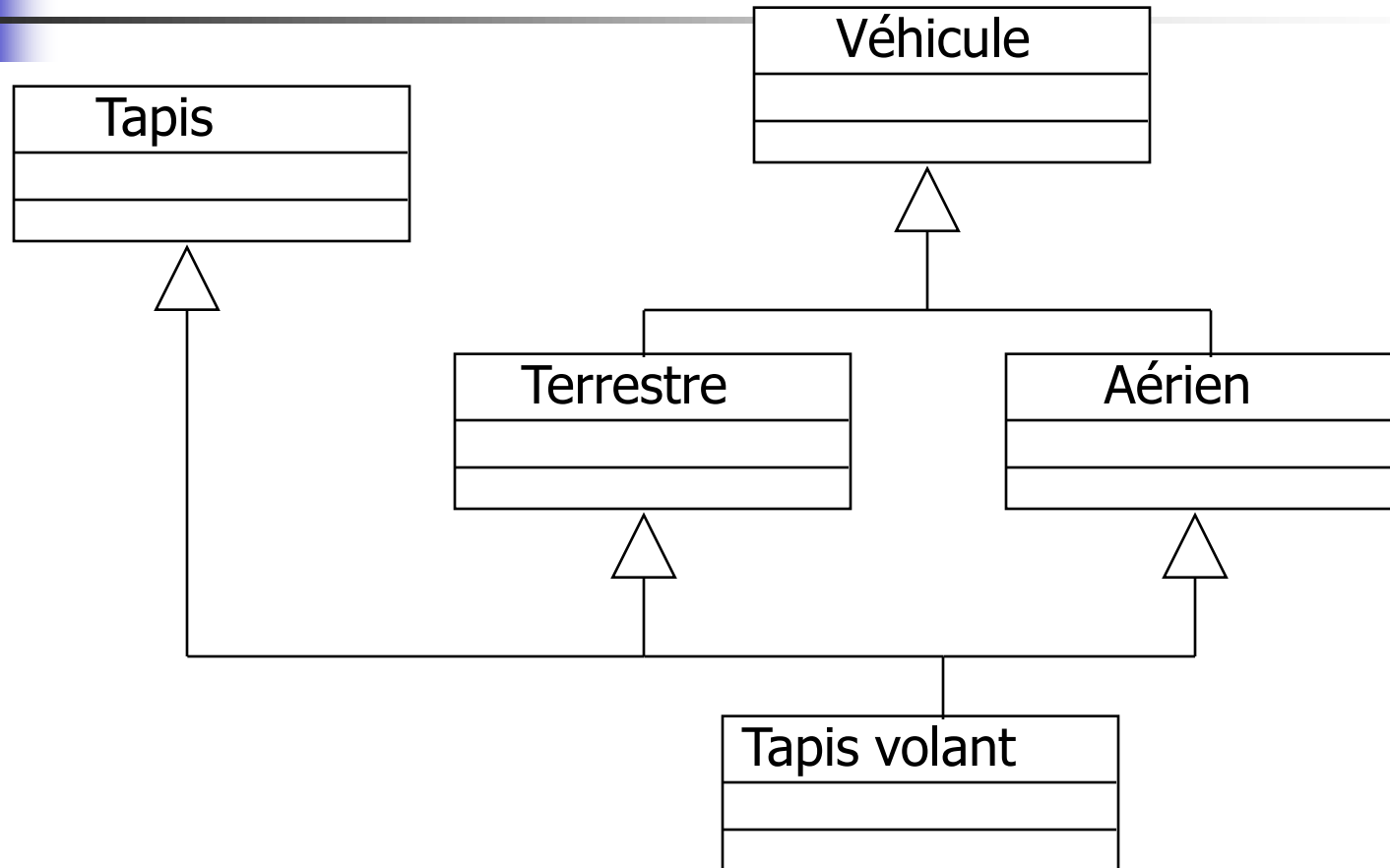


Généralisation

Dans le cas des classes, la relation de généralisation signifie ***est un*** ou ***est une sorte de***

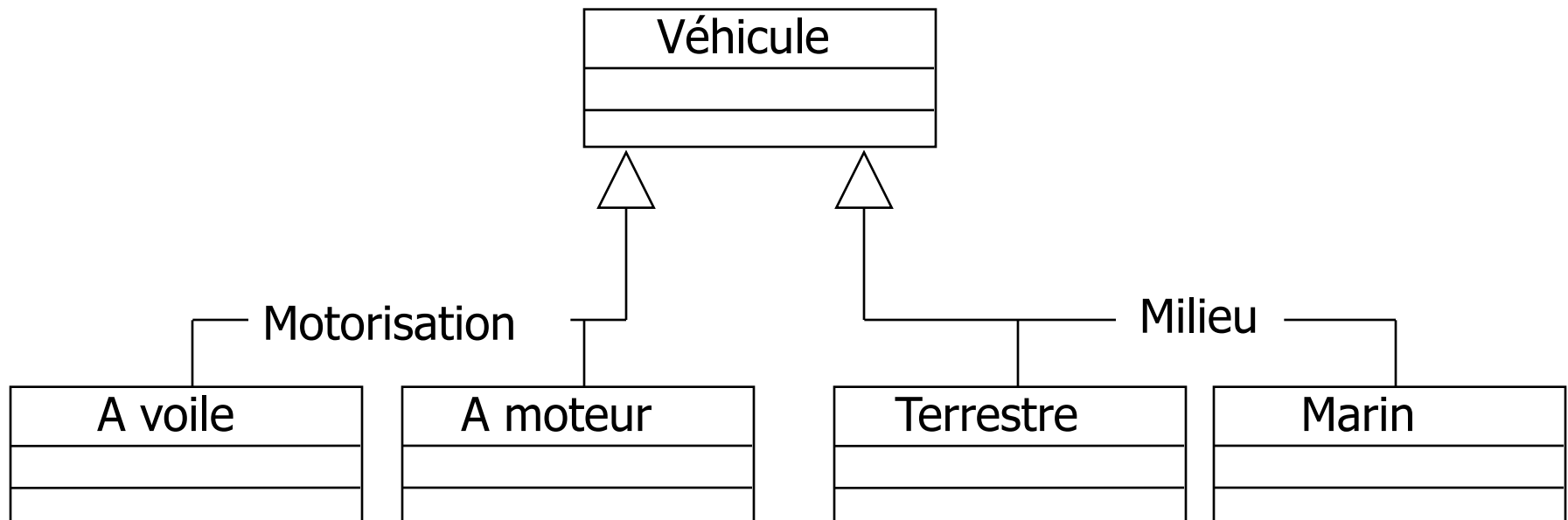


Généralisation multiple



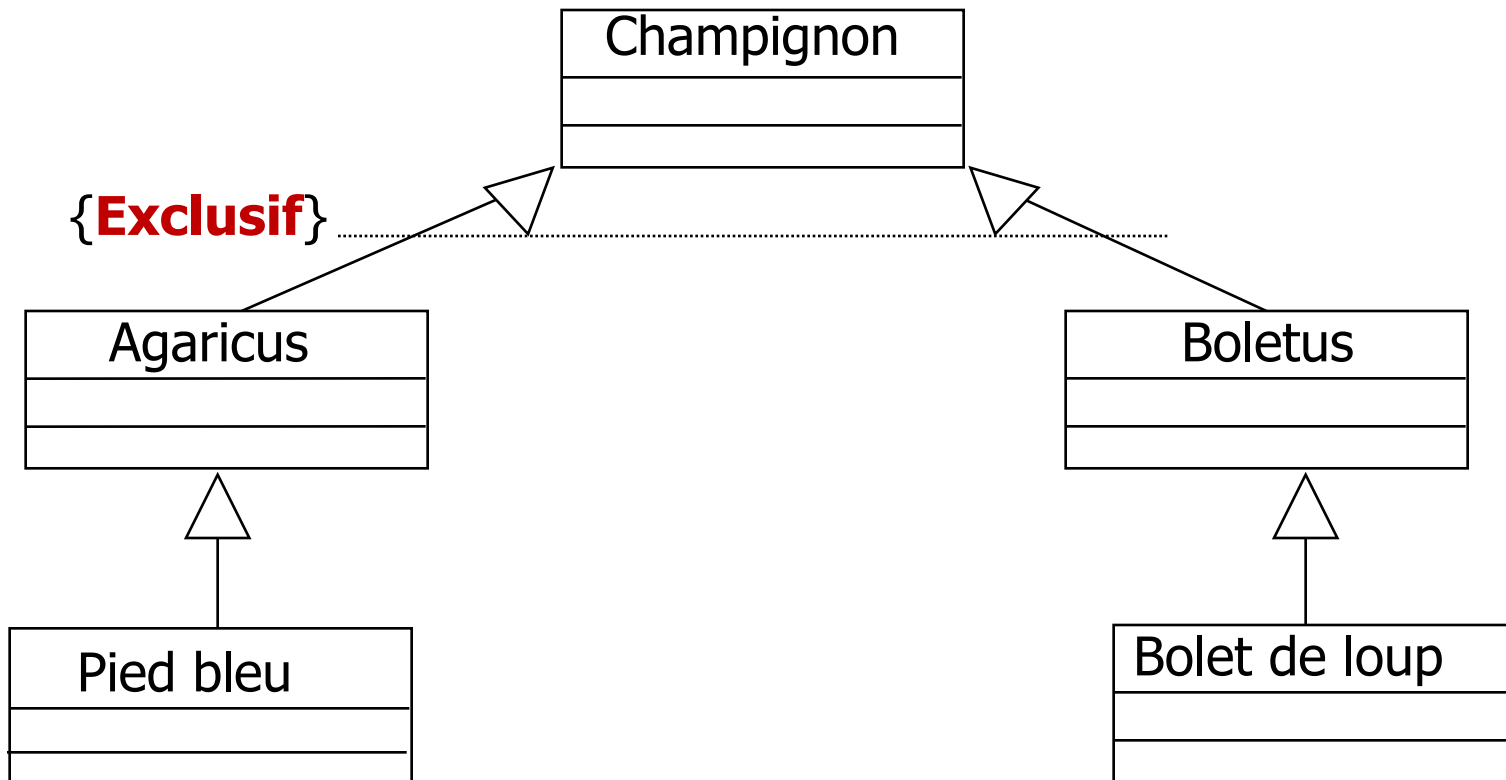
Généralisation

- Une classe peut être spécialisée selon plusieurs critères
- Différentes contraintes peuvent être appliquées aux relations de généralisation
- Par défaut, la généralisation symbolise une décomposition exclusive



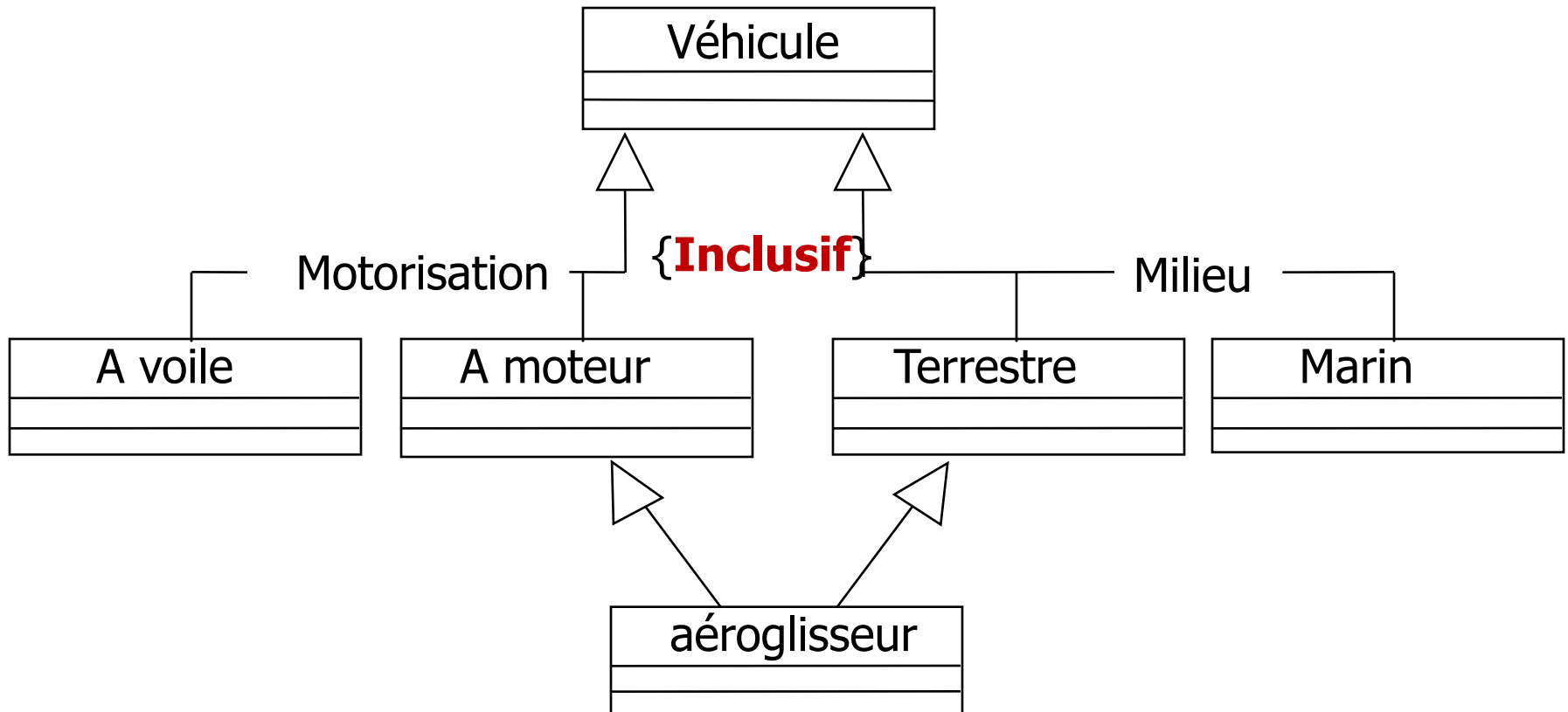
Contraintes de généralisation

- La contrainte **{Disjoint}** ou **{Exclusif}** indique la participation exclusive d'un objet à l'une des collections spécialisées



Contraintes de généralisation

- La contrainte **{Chevauchement}** ou **{Inclusif}** indique qu'un objet peut appartenir à plusieurs collections spécialisées .



Contraintes de généralisation

- La contrainte **{Complète}** 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

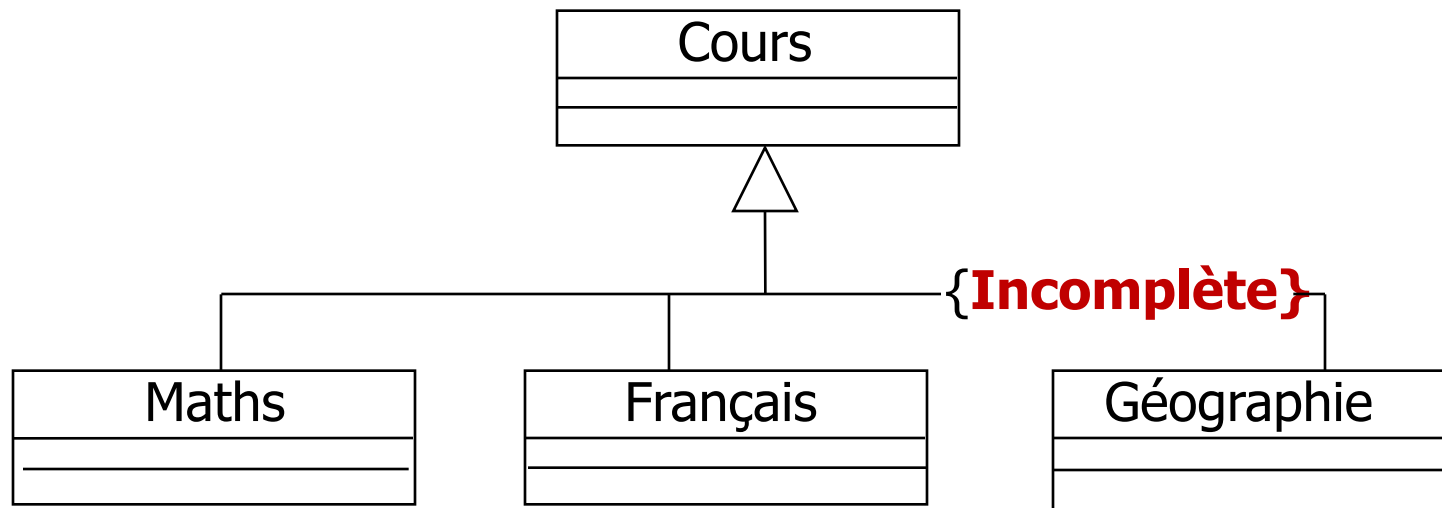


Diagramme d'objets



- Permet de visualiser la structure du système au niveau des instances
- Facilite la compréhension des structures de données complexes
- Trois possibilités de représentation :

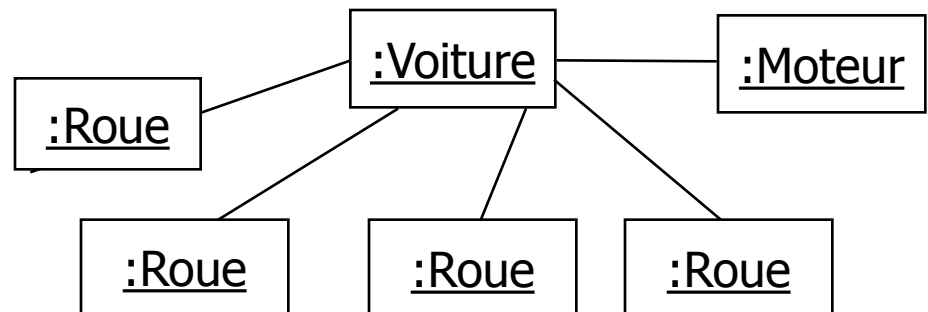
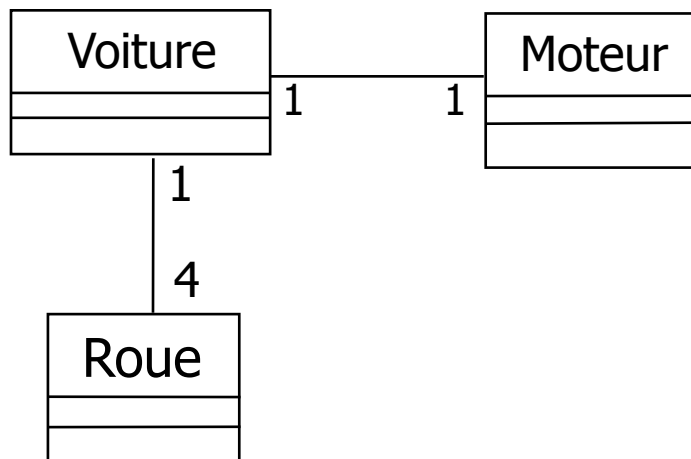
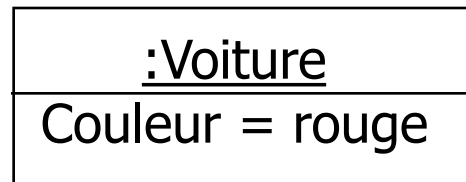
Nom de l'objet

Nom de l'objet : Classe

: Classe

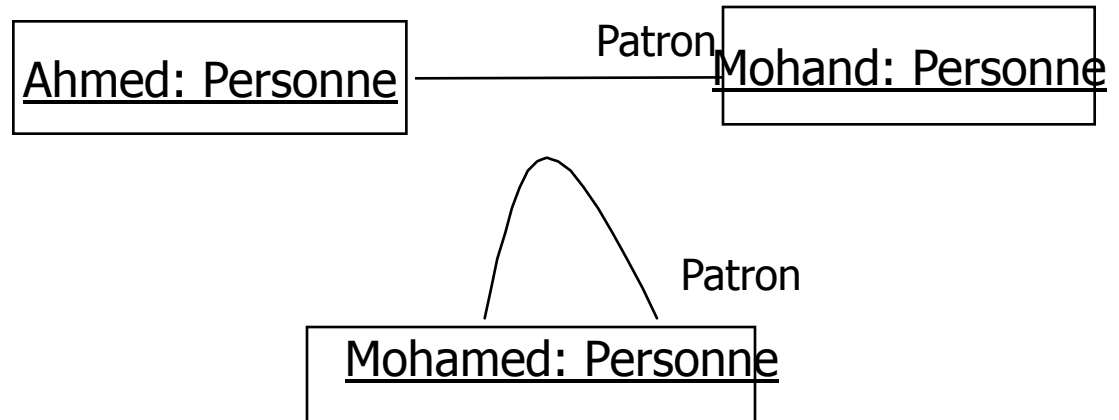
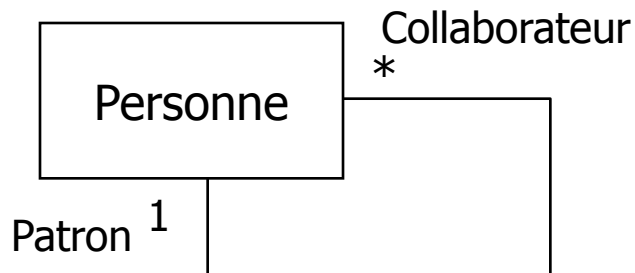
Diagramme d'objets

- On peut faire apparaître des valeurs d'attributs dans un objet
- ainsi que les liens entre objets



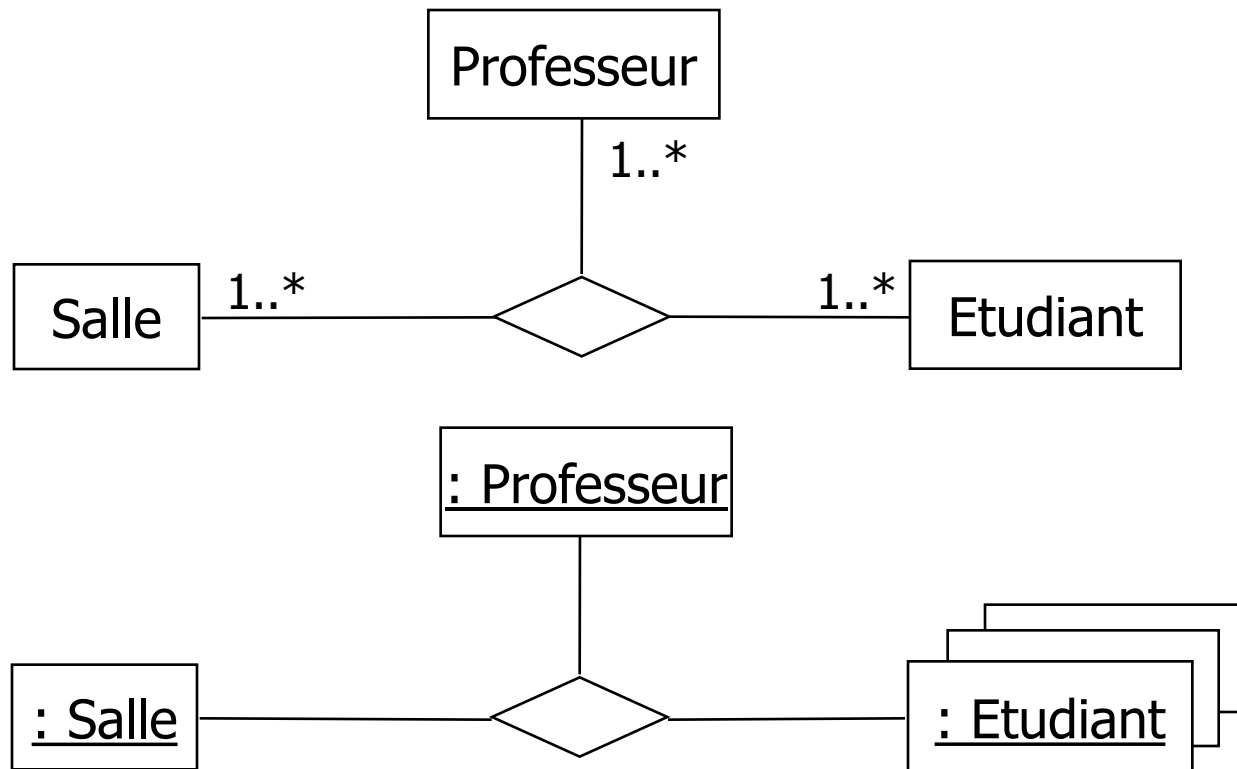
Liens entre les objets

- Les liens instances des associations réflexives peuvent relier un objet à lui-même



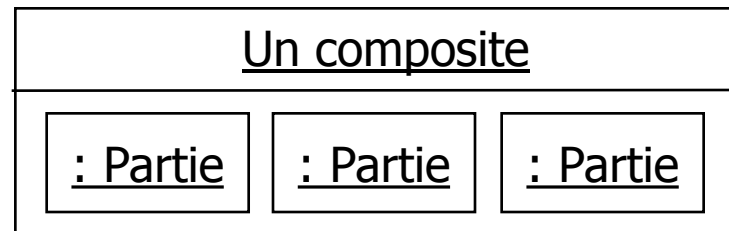
Liens entre les objets

On peut illustrer les liens d'arité supérieure à 2 et la multiplicité *



Objet composite

Les objets composés de sous-objets peuvent être visualisés



❑ Les objets composites sont instances de classes composites:

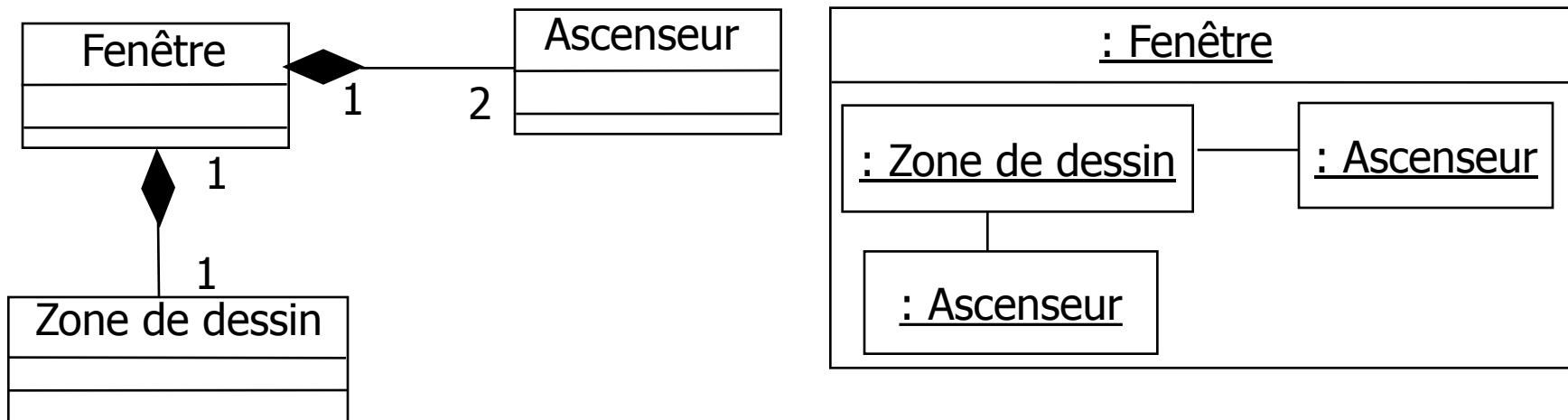


Illustration de structures complexes

Les diagrammes d'objets facilitent la compréhension des diagrammes de classes

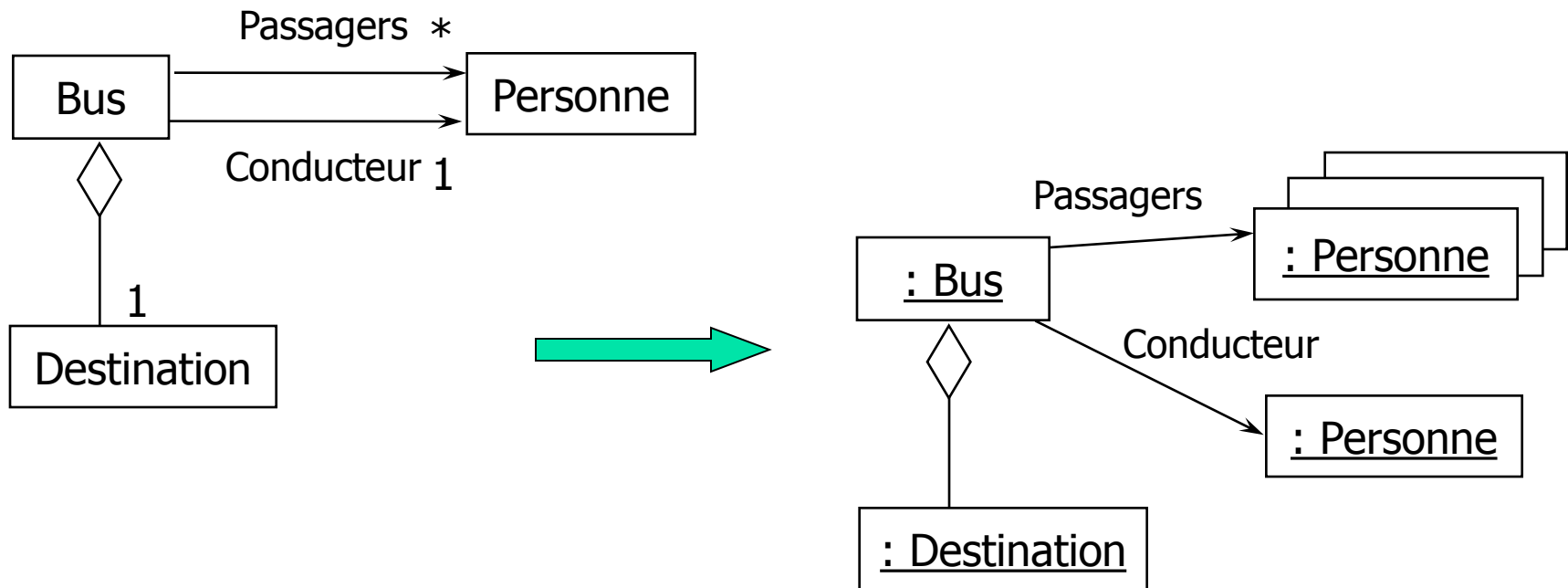
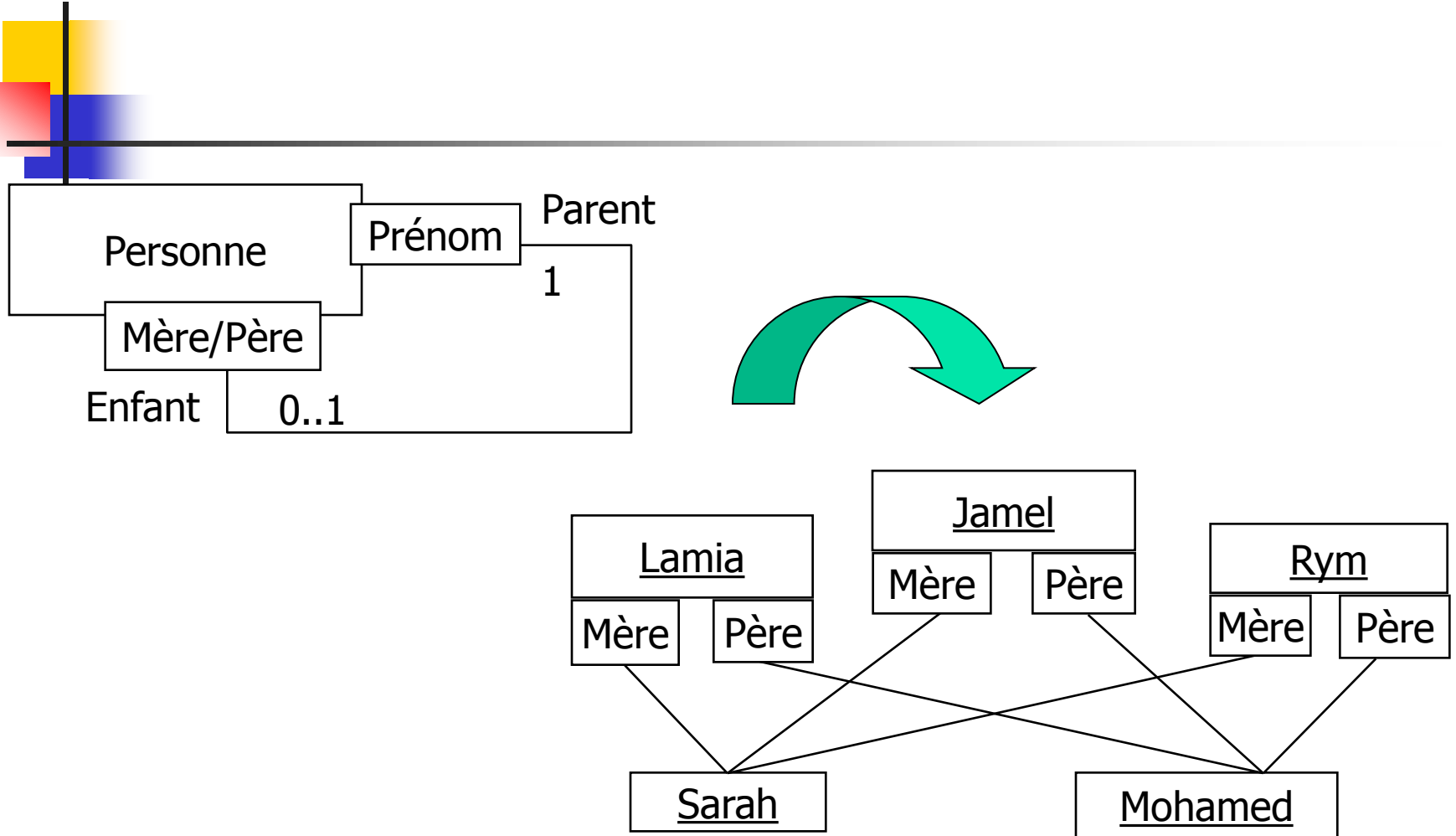
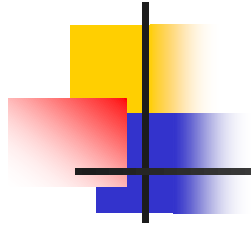


Illustration de structures complexes

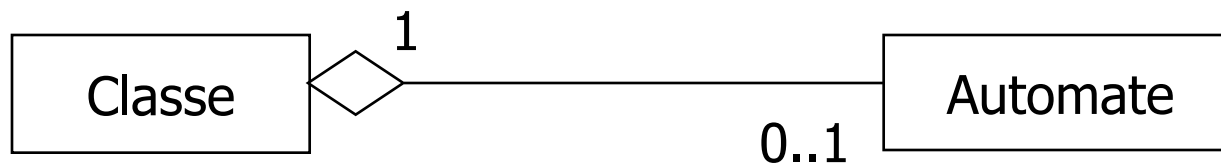




Diagrammes d'états-transitions

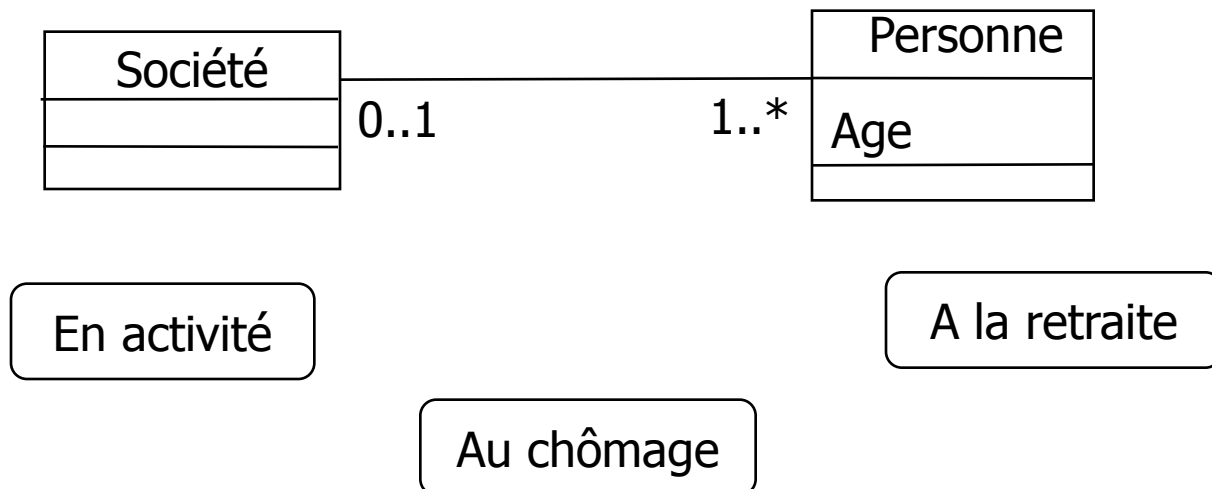
Diagramme d'états-transitions

- Décrit le comportement des objets d'une classe au moyen d'un automate d'états associé à la classe
- Le comportement est modélisé dans un graphe dont les nœuds sont les états possibles des objets de la classe et les arcs sont les transitions d'état à état. Chaque transition résulte de l'exécution d'une action et représente la réaction des objets aux événements qui surviennent
- Les objets qui n'ont pas un comportement réactif très important peuvent être considérés comme des objets qui restent toujours dans le même état: leurs classes ne possèdent alors pas d'automate



La notion d'état

- Un état est une étape dans le cycle de vie d'un objet durant lequel l'objet satisfait certaines conditions, réalise certaines actions ou attend certains événements
- Chaque objet est à un moment donné, dans un état particulier
- Chaque état possède un nom qui l'identifie
- Un état est stable et durable

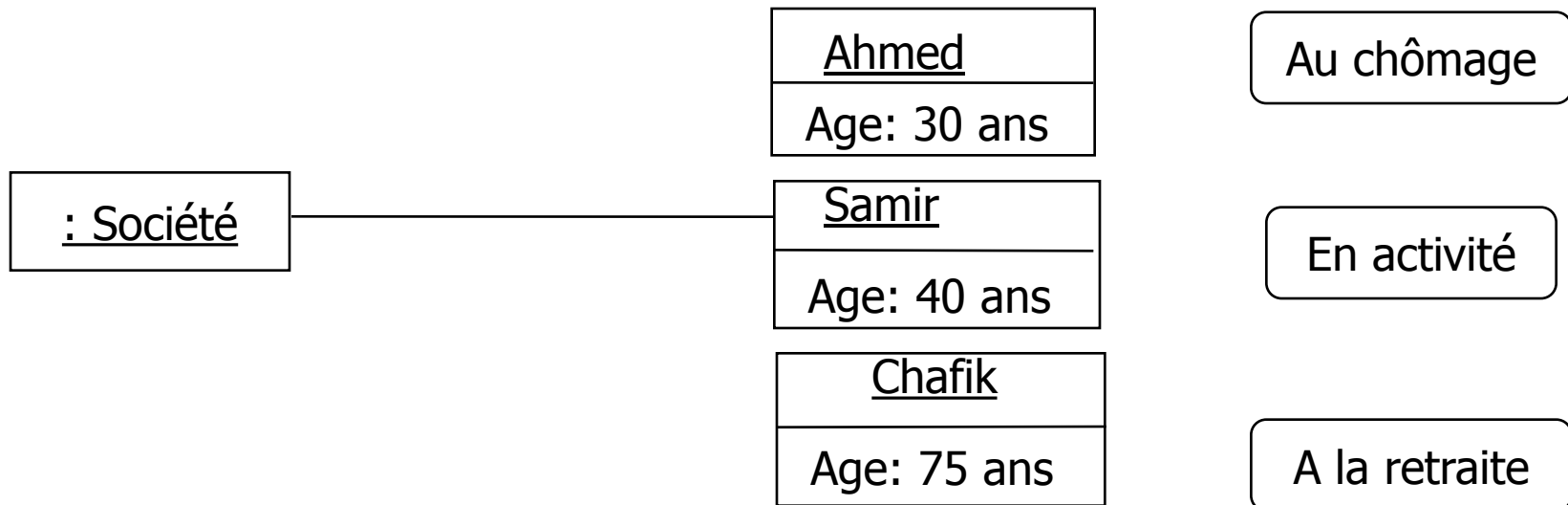


Notion d'état

Un état est l'image de la conjonction instantanée des valeurs des attributs de l'objet, et de la présence ou non de ses liens à d'autres objets

Exemple: L'état d'une personne résulte de la conjonction suivante :

- âge de la personne,
- présence d'un lien vers une société



Notion d'état

- Chaque diagramme d'états-transition contient un ***état initial***
- Pour un niveau hiérarchique donné, il y a ***un et un seul état initial*** mais il est possible qu'il y ait ***plusieurs états finaux*** qui correspondent chacun à une fin de vie d'objet différente
- Il est également possible de n'avoir ***aucun état final*** (par exemple, un système qui ne s'arrête jamais)



État initial

État intermédiaire

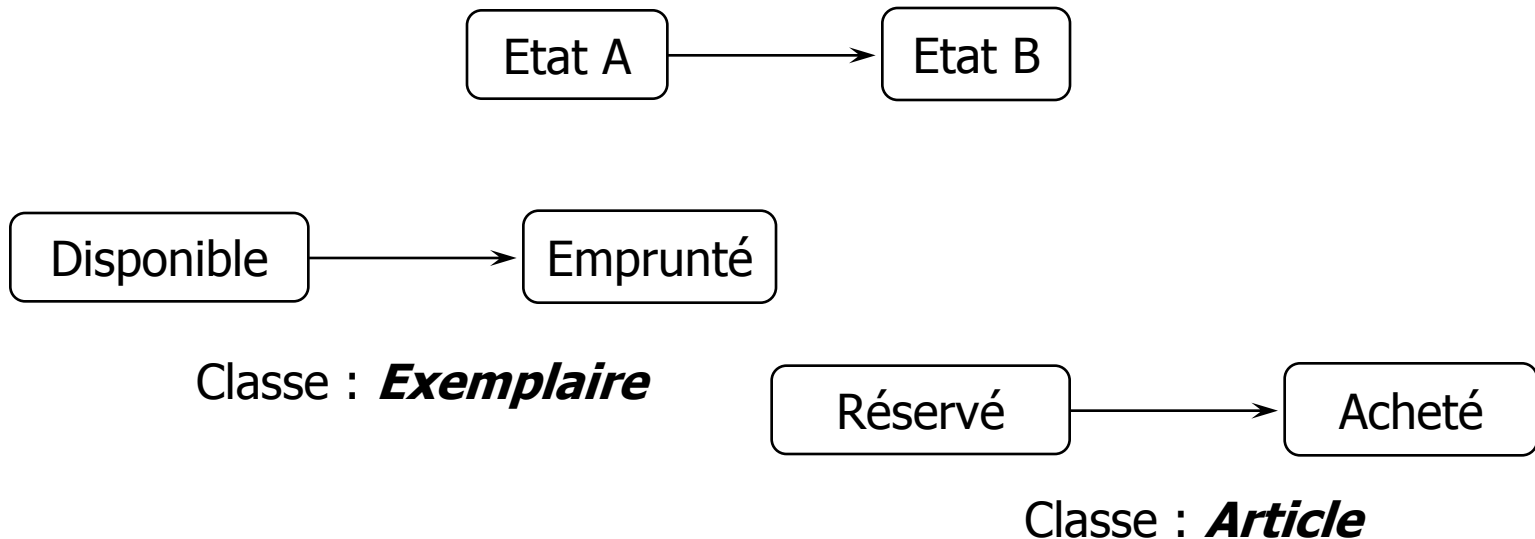


État final

Notion de transition

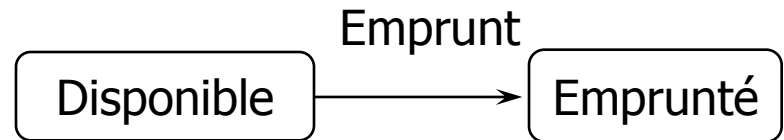
Lorsque des événements se produisent, les objets changent d'état en respectant les règles décrites dans l'automate associé à leur classe

- Les diagrammes d'états-transitions sont des graphes dirigés
- Les états sont reliés par des connexions unidirectionnelles, appelées **transitions**



Notion d'événement

- Un événement correspond à l'occurrence d'une situation donnée dans le domaine du problème
- Un événement est par nature une information instantanée qui doit être traitée sans plus attendre.
- L'événement est le déclencheur de la transition d' état à état Un objet, placé dans un état donné, attend l'occurrence d'un événement pour passer dans un autre état



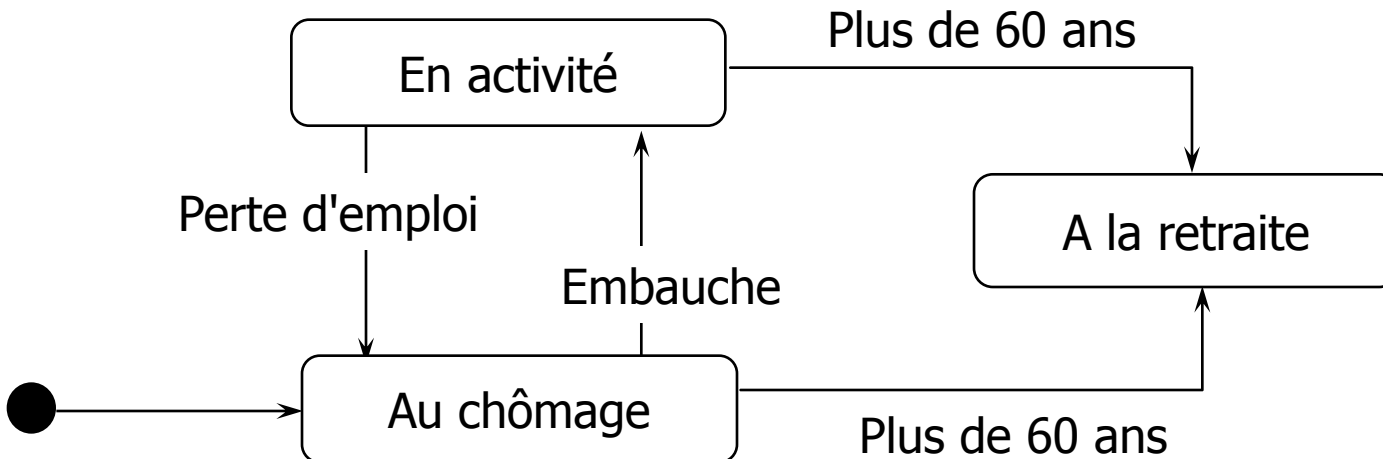
Classe : ***Exemplaire***

Notion d'événement

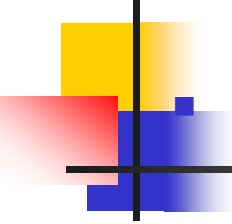
La syntaxe générale d'un événement est la suivante:

~~Nom de l'événement (Nom de paramètre : Type, ...)~~

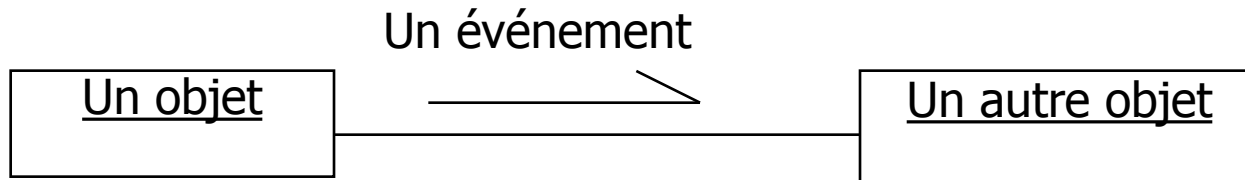
- La spécification complète d'un événement comprend:
 - le nom de l'événement
 - la liste des paramètres
 - l'objet expéditeur
 - l'objet destinataire
 - sa description



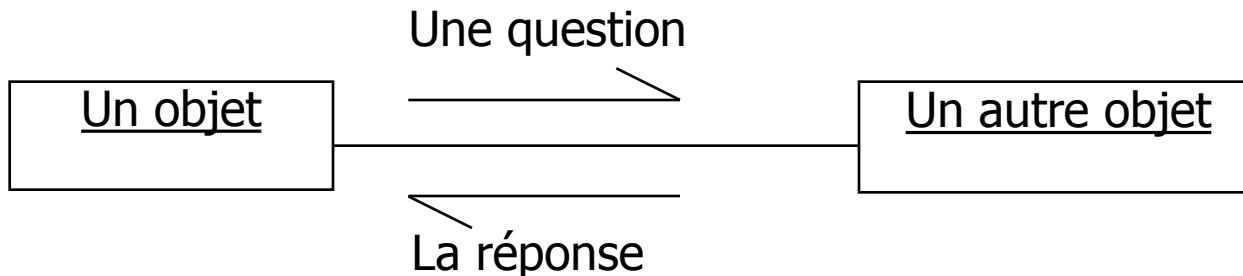
Communication entre objets par événement



La communication par événement est de type ***asynchrone***, ***atomique***, et ***unidirectionnelle***. Un objet peut envoyer un événement à un autre objet qui doit toujours être à même de l'interpréter

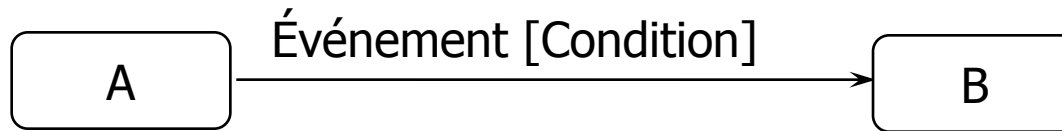


- ❑ Les besoins de communication par événements synchrones ou les échanges bidirectionnels peuvent se représenter au moyen de deux échanges asynchrones de directions opposées



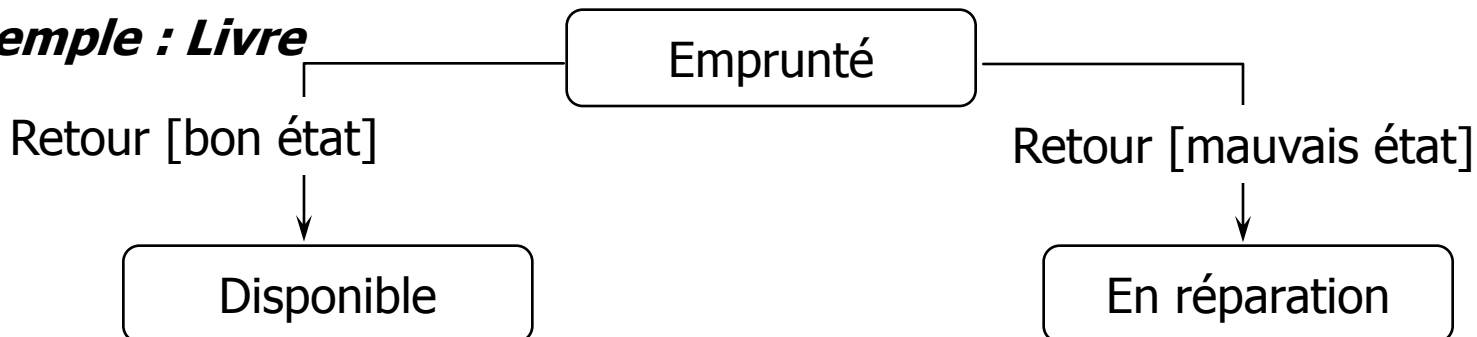
Notion de garde

- Une garde est une condition booléenne qui permet ou non le déclenchement d'une transition lors de l'occurrence d'un événement



- ❑ Les gardes permettent de maintenir l'aspect déterministe d'un automate d'états finis. Lorsque l'événement a lieu, les gardes - qui doivent être mutuellement exclusives - sont évaluées et une transition est validée puis déclenchée

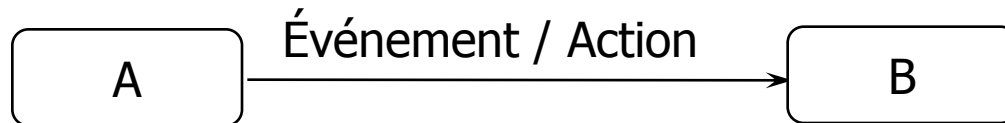
Exemple : Livre



Notions d'opération et d'action

Le lien entre les opérations définies dans la spécification d'une classe et les événements apparaissant dans le diagramme d'états-transitions de cette classe est assuré par le biais des *actions* et des *activités*

- Chaque transition peut avoir une action à exécuter lorsque la transition est déclenchée par un événement
- L'action est considérée comme instantanée et atomique



- L'action correspond à une des *opérations* déclarées dans la classe de l'objet destinataire de l'événement. L'action a accès aux paramètres de l'événement, ainsi qu'aux attributs de l'objet

Actions dans un état

Les états peuvent également contenir des actions; elles sont exécutées à l'entrée ou à la sortie de l'état ou lors de l'occurrence d'un événement pendant que l'objet est dans l'état :

- L'action d'entrée (**entry:**) est exécutée de manière instantanée et atomique dès l'entrée dans l'état
- L'action de sortie (**exit:**) est exécutée à la sortie de l'état
- L'action sur événement interne (**on:**) est exécutée lors de l'occurrence d'un événement qui ne conduit pas à un autre état

Nom d'un état

entry : Action d'entrée

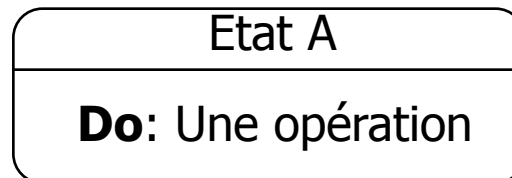
on Nom d'un événement : Action

exit : Action de sortie

Opération et activité

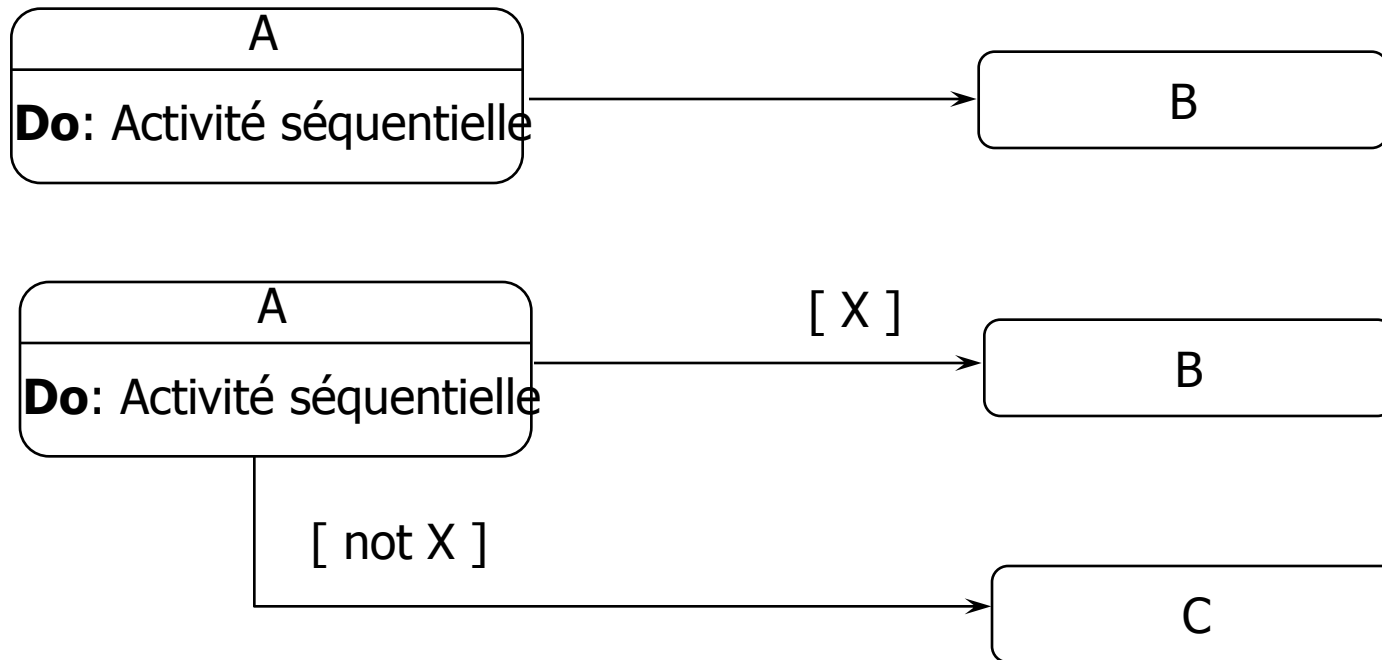
Une opération qui prend un temps non négligeable et qui est exécutée pendant que l'objet est dans un état donné est appelée une *activité*

- Le mot-clé **do:** indique une *activité*
- Contrairement aux actions, les activités peuvent être interrompues à tout moment, dès qu'une transition de sortie de l'état est déclenchée
 - Activité cyclique: elle ne s'arrête que lorsqu'une transition de sortie est déclenchée
 - Activité séquentielle: elle démarre à l'entrée dans l'état. Lorsqu'elle parvient à son terme, l'état peut être quitté si une des transitions est franchissable. C'est une transition automatique



Notion d'activité

Lorsque l'activité se termine, les transitions automatiques, sans événements, mais éventuellement protégées par des gardes, sont déclenchées



Points d'exécution des opérations

6 façons d'associer une opération à une transition:

- l'action associée à la transition d'entrée (**Op1**)
- l'action d'entrée de l'état (**Op2**)
- l'activité dans l'état (**Op3**)
- l'action de sortie de l'état (**Op4**)
- l'action associée aux événements internes (**Op5**)
- l'action associée à la transition de sortie de l'état (**Op6**)

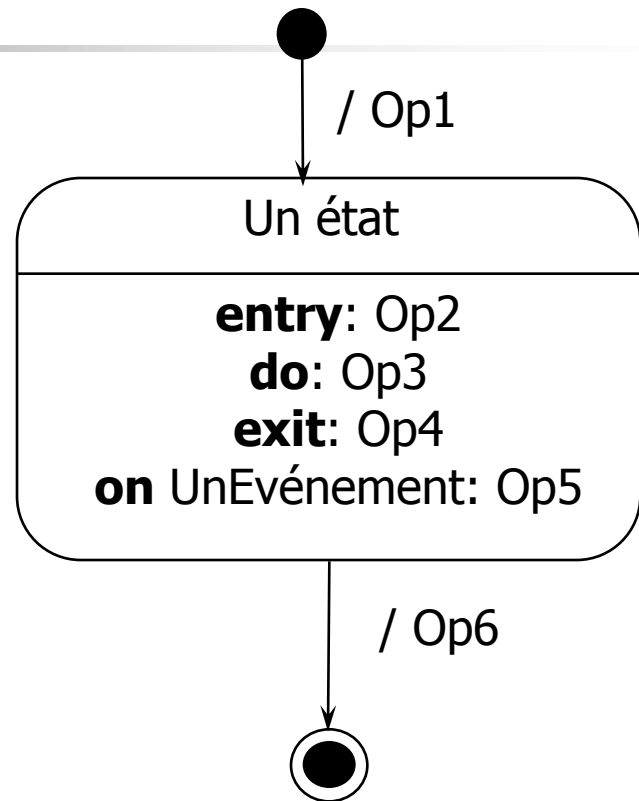
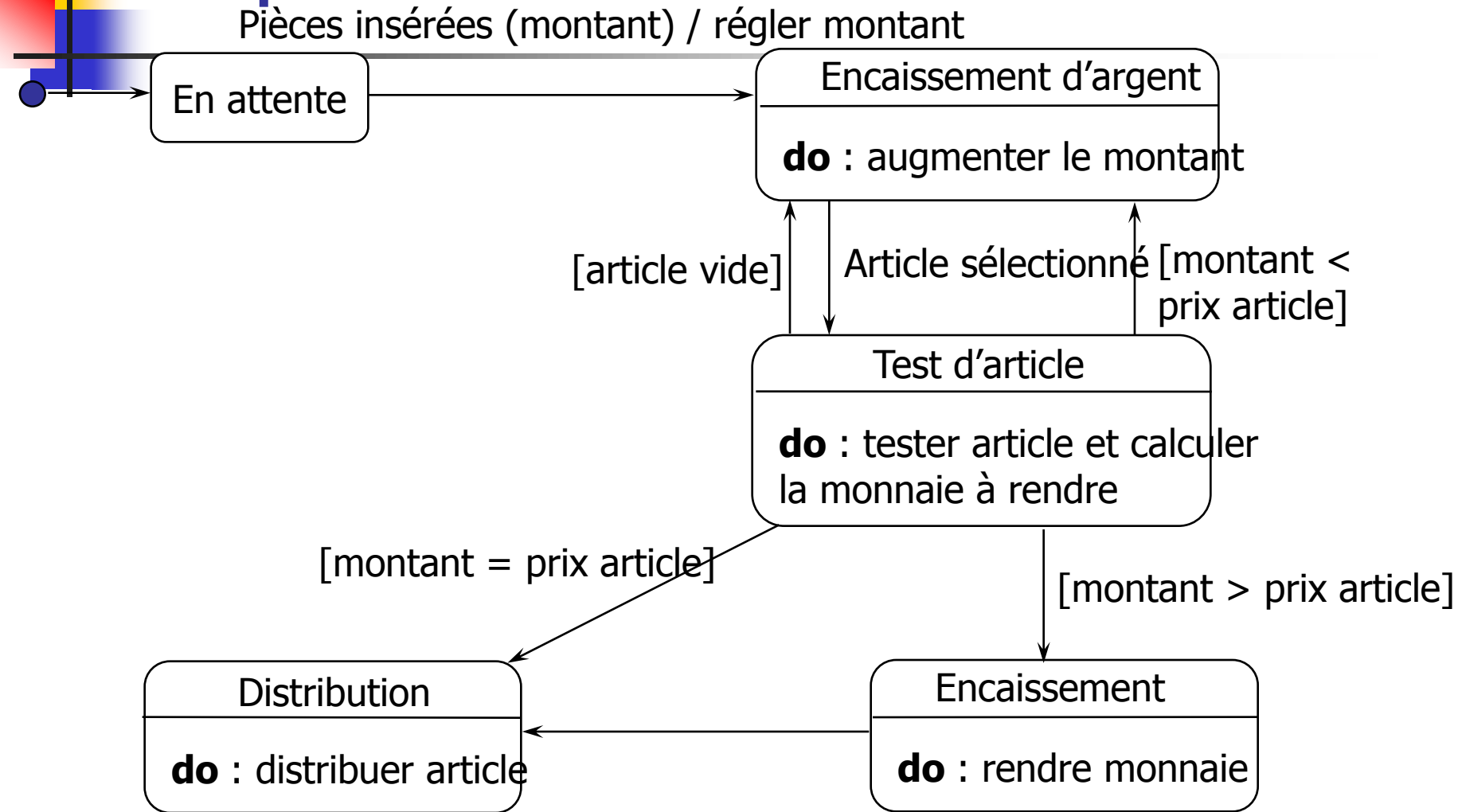


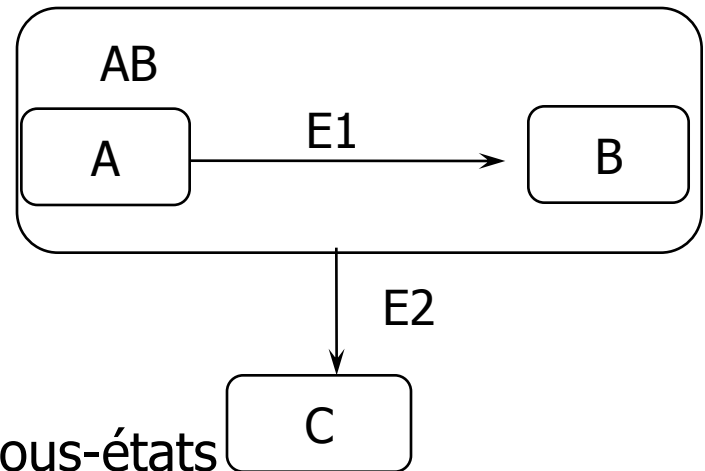
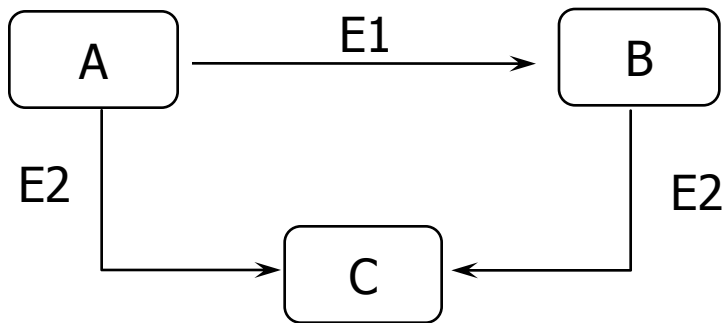
Diagramme d'états-transitions du distributeur automatique des boissons



Généralisation d'états

Un état peut être décomposé en plusieurs sous-états disjoints; les sous-états héritent des caractéristiques de leur super-état

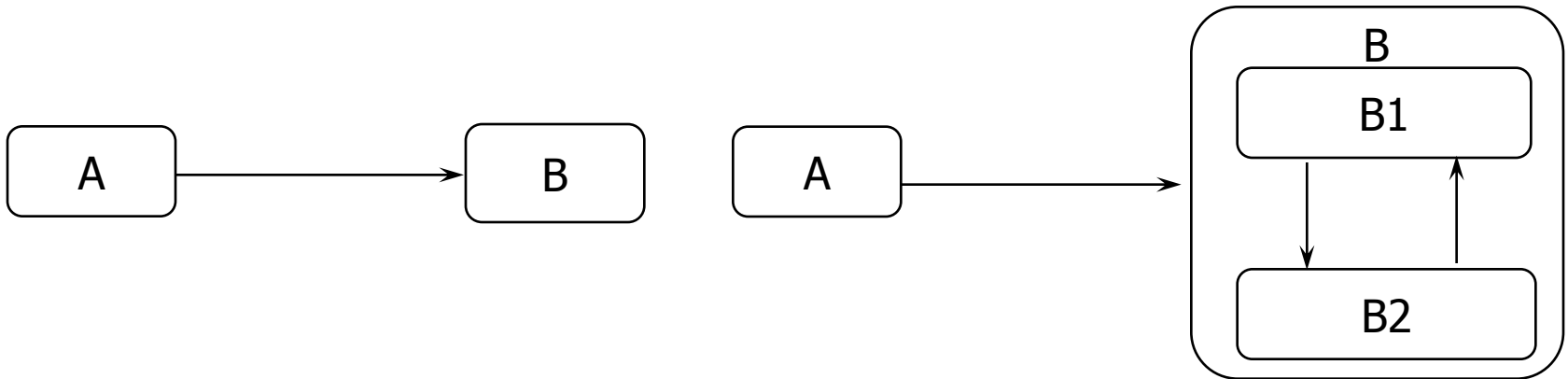
- Décomposition disjonctive: l'objet doit être dans un seul sous-état à la fois



AB est un super état dont A et B sont des sous-états
Cette généralisation permet de factoriser la transition E2

Généralisation d'états

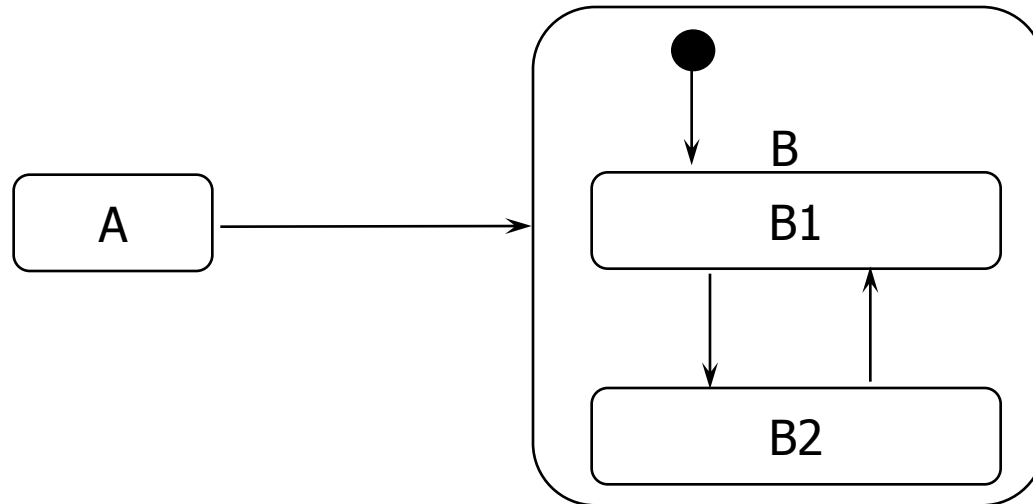
- Les transitions d'entrée ne sont pas héritées par tous les états, seul un état peut être cible de la transition



L'état B est divisé en deux sous-états B1 et B2. La transition d'entrée dans l'état B doit être reportée directement sur un des sous états.

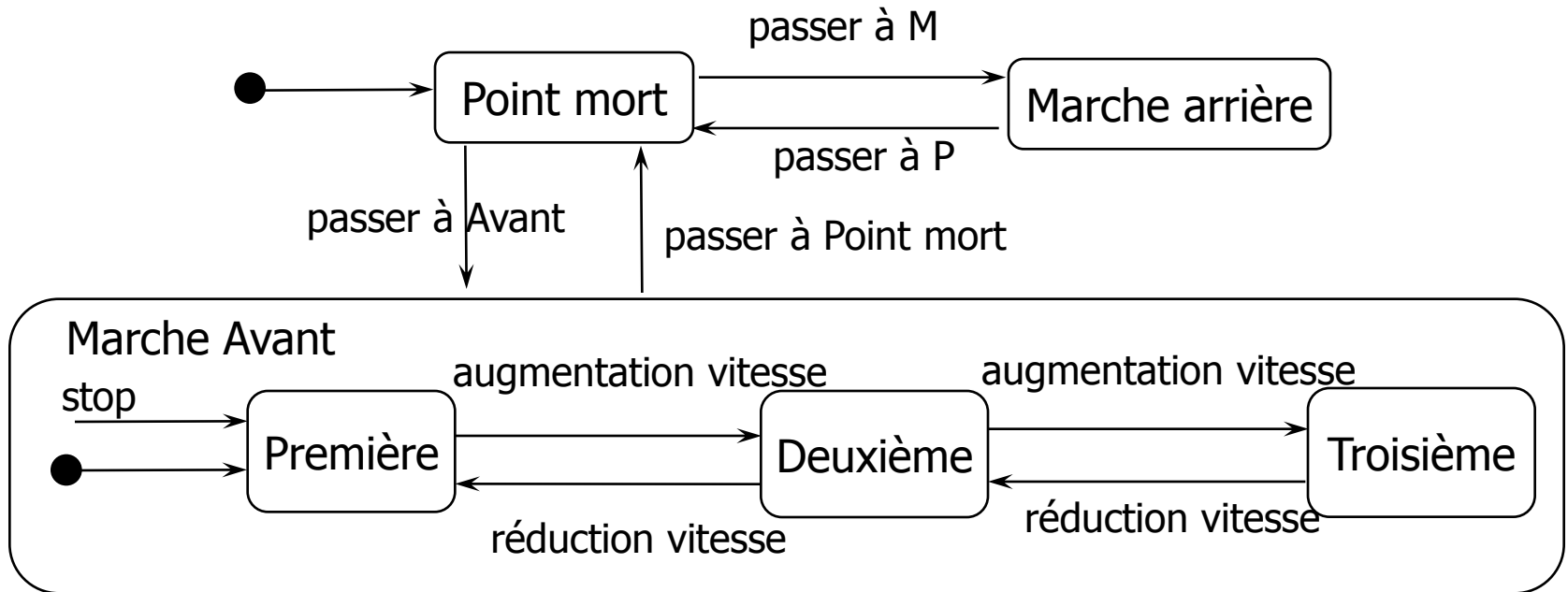
Généralisation d'états

Il est préférable de limiter les liens entre niveaux hiérarchiques d'un automate en définissant systématiquement un état initial pour chaque niveau



Généralisation d'états

Exemple: boîte automatique de transmission

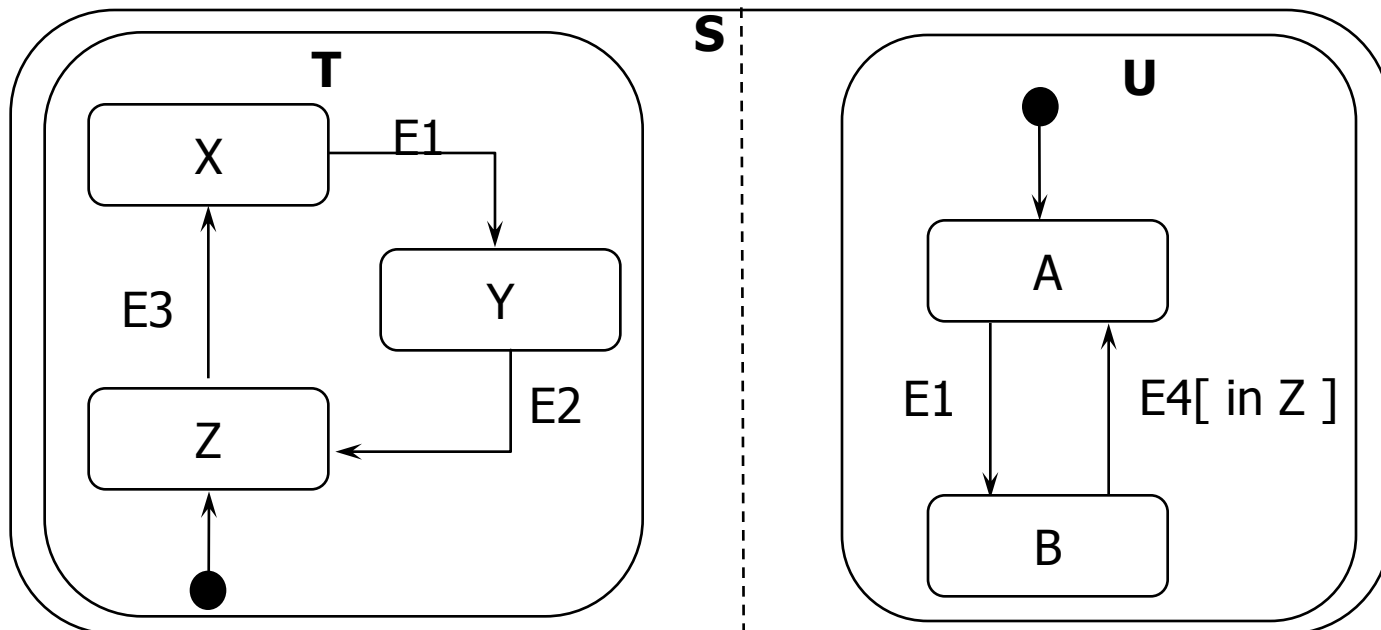


La généralisation d'état permet de mettre en facteur les deux transitions 'passer à Avant' et 'passer à Point mort'

Agrégation d'états

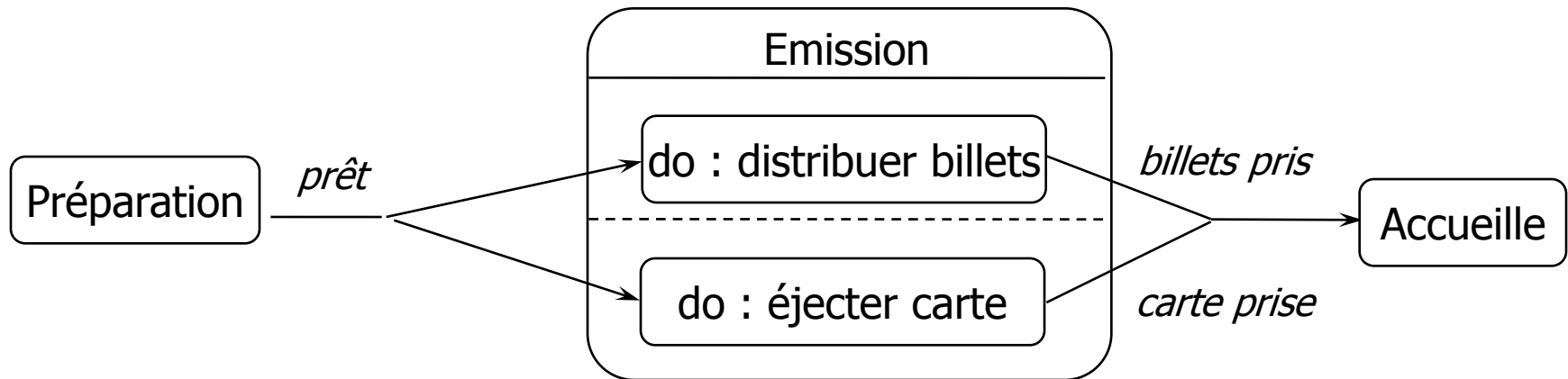
L'agrégation d'états est la composition d'un état à partir de plusieurs autres états indépendants

- La composition est de type conjonctive ce qui implique que l'objet doit être simultanément dans tous les états composant l'agrégation d'états
- Forme de parallélisme entre automates



Agrégation d'états

Exemple: activité d'émission des billets





Manifeste du système de BD OO: 13 règles [Atkinson 1989]

- Il gère des objets complexes: objets de base + constructeurs:
 - Ensemble minimal de constructeurs:
 - Ensemble (SET)
 - L'occurrence (TUPLE)
 - La liste (LIST ou ARRAY)
- Il gère l'identité d'objet:
 - Tout objet doit posséder une identité unique, indépendante des valeurs d'un attribut.



Manifeste du système de BD OO

- Il permet l'encapsulation :
 - Accès aux méthodes qu'à travers l'interface de l'objet.
 - Données et implémentation des méthodes sont masqués dans l'objet.
- Il gère les types ou les classes :
 - Le schéma de BD comporte un ensemble de classes ou de types.



Manifeste du système de BD OO


- **Les types ou les classes doivent pouvoir hériter de leurs ancêtres:**
 - Un sous-type ou une sous-classe doit hériter des attributs et des méthodes, respectivement, de son super type ou de sa superclasse.
- **Il doit assurer les liens dynamiques :**
 - Les méthodes s'appliquent à des objets de types différents. A chaque type un code différent, donc la liaison s'effectue à l'exécution: liaison dynamique



Manifeste du système de BD OO

- **Le LMD est complet au niveau calcul :**
 - Le LMD du SGBDOO doit être un langage de programmation à vocation générale et non pas seulement un langage de requêtes.

- **L'ensemble des types de données est extensible :**
 - L'utilisateur doit pouvoir construire de nouveaux types à partir des types prédéfinis par le système.



Manifeste du système de BD OO: règles des SGBD traditionnels

- **Le système permet la persistance des données :**
 - Comme dans un SGBD traditionnel, les données doivent continuer d'exister donc persister après la fin de l'application qui les a créées.
- **Le SGBD doit être capable de gérer de très vastes bases de données:**
 - Les données sont stockées en mémoire secondaire, moyennant l'utilisation de méthodes d'accès et de stockage sophistiquées. Le SGBDOO doit disposer de mécanismes semblables.
- **Le SGBD accepte des utilisateurs simultanés et concurrents :**
 - Existence de mécanismes de contrôle de concurrence semblables à ceux des SGBD classiques.



Manifeste du système de BD OO

- Le SGBD assure la reprise en cas de défaillances matérielles et/ou logicielles:
 - Existence de mécanismes de récupération semblables à ceux des SGBD traditionnels.
- Le SGBD fournit un moyen simple pour interroger les données :
 - Existence d'un utilitaire de requêtes de haut niveau (déclaratif), efficace, indépendant de l'application ; (langage ou navigateur graphique)



Manifeste du système de BD OO

- caractéristiques facultatives prônées par le manifeste :
 - Héritage multiple
 - Distribution de données
 - Conception de transactions et gestion de versions

Ce qui manque:

- sécurité, intégrité, vue



Approches de développement de SGBDOO

[Khoshafian 1990]

- **Étendre un langage de programmation OO existant avec des fonctionnalités de BD** : par exemple, Smalltalk, C++, Java; (GemStone):
- **Proposer des bibliothèques de SGBD Orientées Objet:**
Ajouter des fonctionnalités à un langage de POO existant mais à travers des bibliothèques de classes acceptant la persistance, l'agrégation, les transactions, la sécurité etc. par exemple Ontos, ObjectStore.
- **Intégrer des constructions objet du langage de BD dans un langage hôte conventionnel.** Par exemple extensions intégrées au langage C: système O2



Approches de développement de SGBDOO

[Khoshafian 1990]

- **Étendre le langage de BD existant avec des caractéristiques OO. Étendre SQL par de constructions OO** (norme SQL 1999). L'object Data Management Group (ODMG) spécifie une norme Object SQL. Nombre d'éditeurs de SGBDOO se mettent en conformité avec cette norme.
- **Développer un nouveau modèle et un nouveau langage avec des caractéristiques totalement orientées objet.** Approche suivie par Semantic Information Manager (SIM) : modèle sémantique plus un LDD et un LMD nouveaux. [Jagannathan 1988].



Perspectives des SGBDOO

Les SGBD de manière générale sont concernés par la création et l'entretien de vastes collections de données: ils se caractérisent par :

- Le modèle de données
- La persistance de données
- Le partage des données
- La fiabilité
- L'évolutivité
- La sécurité et l'intégrité
- La distribution

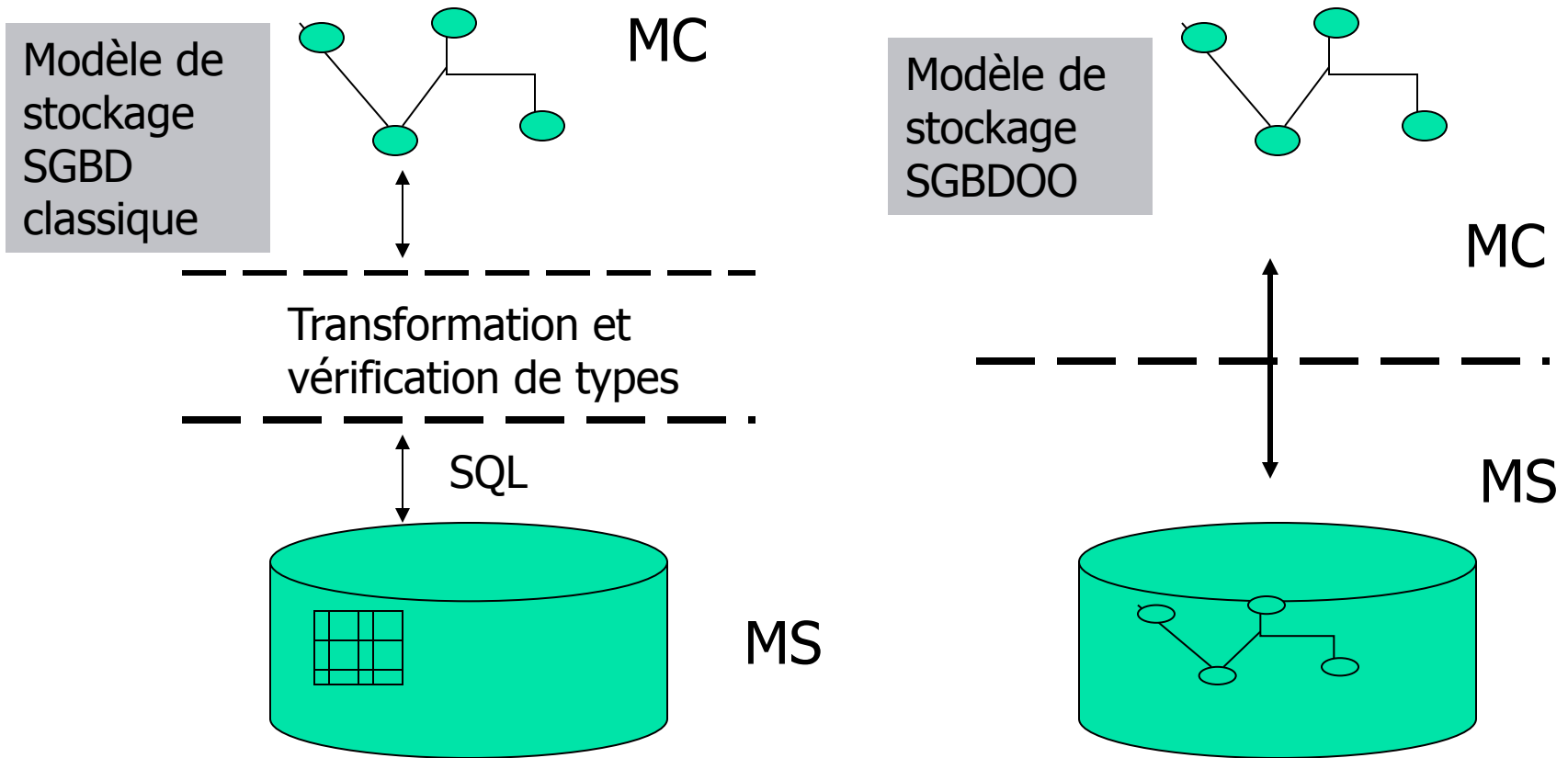
Les langages de programmation traditionnels ne permettent pas la prise en compte de ces caractéristiques. Aujourd'hui les applications exigent à la fois des fonctionnalités de langage de programmation classique et de SGBD; des efforts sont consentis pour développer des systèmes tenant compte de ces deux domaines.



Caractéristiques des SGBDOO

- Les SGBD classiques utilisent un modèle de stockage à deux niveaux :
 - Le modèle de stockage de l'application en mémoire principale
 - Le modèle de stockage des données en mémoire secondaire.
- Les SGBDOO utilise un modèle de stockage à un seul niveau:

Caractéristiques des SGBDOO





Caractéristiques des SGBDOO

- 1-Gestion des OID : 2 types
 - OID logique : indépendants de l'emplacement physique de l'objet sur MS: donc indirection nécessaire pour connaître l'adresse physique.
 - OID physique qui codifie l'emplacement.

- Dans les deux cas, pour atteindre des performances raisonnables et optimiser les accès aux objets, le SGBDOO doit convertir les OID en pointeurs en mémoire et vice-versa. Cette technique de conversion porte le nom de « **pointer swizzling** »



Techniques de conversion de pointeurs

- Action qui consiste à convertir les OID en pointeurs en mémoire principale et vice-versa
- Utilisation d'une table de correspondance appelée « table d'objets résidents »

Table d'objets résidents référençant 4 objets présents en mémoire centrale

Agence :OID1
Numfi : F003
Adresseafi :
Employé : (OID4, OID5)
Propriété : (OID2, OID3)
Gérant : OID6

Employe : OID4
Numemp : EMP14
Prénom : Mohamed
Nom : Rami
::
Agence : OID1

Table d'objets résidents	
OID	Adresse en mémoire
1	
2	
3	
4	
..	

PropriétéAlouer:OID2
Numprop : LM1
Adresse
Type: appartement
Employe : OID4
Agence: OID1

PropriétéAlouer: OID3
Numprop : LM3
Adresse
Type: appartement
Employe : OID4
Agence: OID1

Si nous voulons accéder depuis agence (OID1) à employé OID5, une recherche dans la table indique qu'il n'est pas résident en MC. Il faut donc accéder à l'objet en MS et transférer son adresse dans la table. Par contre si nous voulons accéder à employé OID4 à partir de Agence, une recherche dans la table indique que l'objet est en MC.



Caractéristiques des SGBDOO

- 2- Persistance : 3 schémas de réalisation
 - Etablissement de points de contrôle
 - Basée sur l'accessibilité
 - Basée sur l'allocation

Rappel : objets persistants sont des objets qui survivent après la fin du programme qui les a créés.



Caractéristiques des SGBDOO : persistance par établissement de points de contrôle

- Copie de la totalité ou d'une partie de l'espace d'adressage d'un programme d'application dans la MS. En cas d'interruption, le programme peut redémarrer à partir de « checkpoint ».



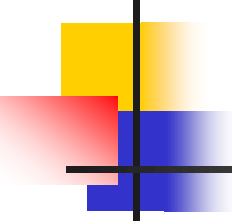
Caractéristiques des SGBDOO : persistance basée sur l'accessibilité

- Un objet est persistant s'il est accessible par un objet racine persistant. Le système gère alors une arborescence d'accessibilité.



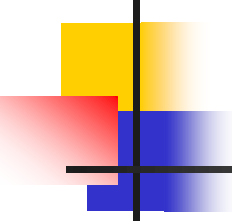
Caractéristiques des SGBDOO : persistance basée sur l'allocation

- Un objet ne devient persistant que s'il a été déclaré comme tel au sein du programme d'application.



Notions spécifiques aux SGBDOO : transactions longues, la gestion de versions, l'évolution de schémas

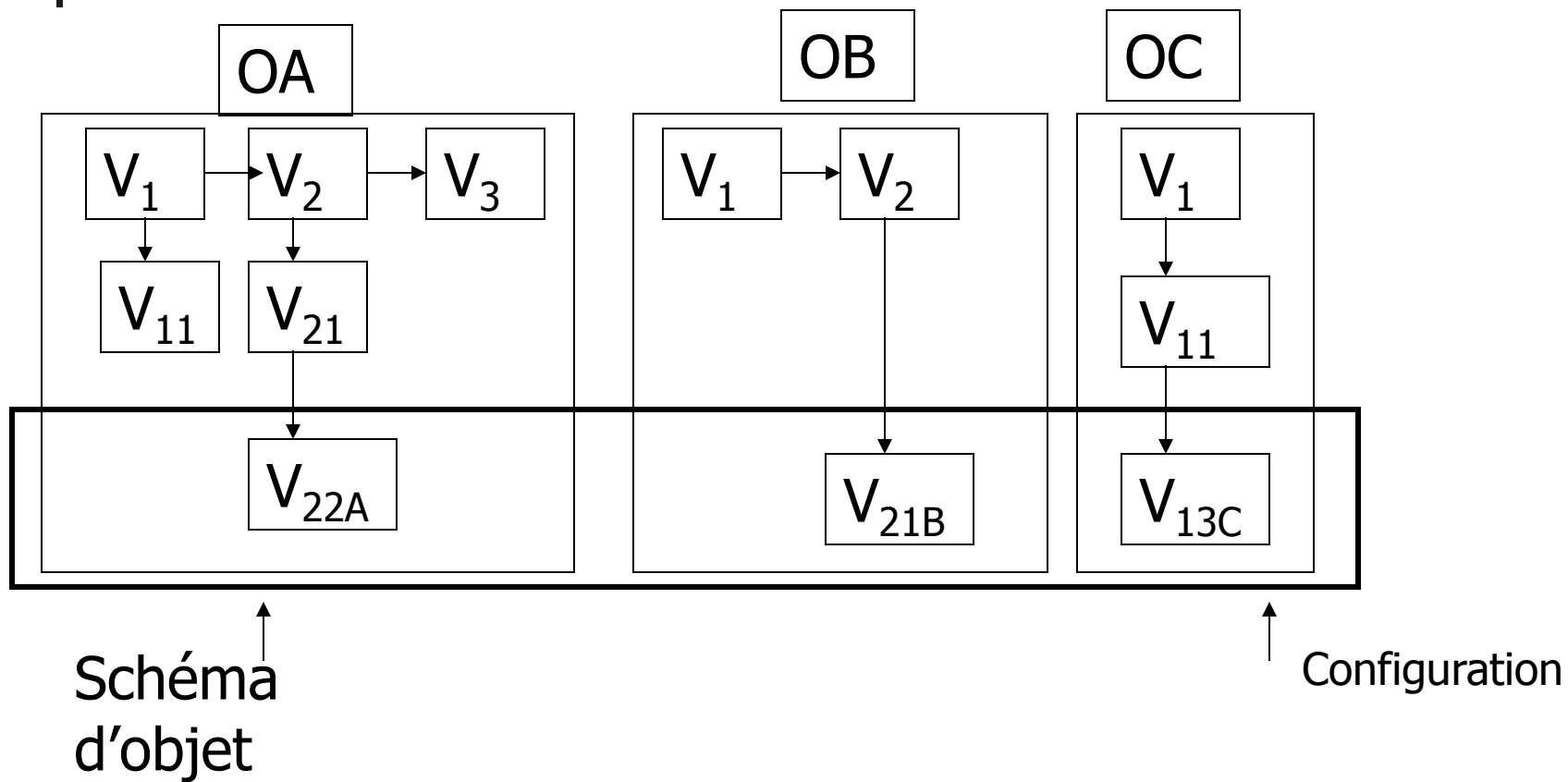
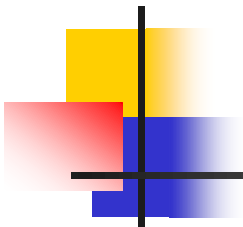
- **Les transactions longues** : les transactions faisant intervenir des objets complexes peuvent se poursuivre sur de longues périodes de temps: nécessité de protocoles différents de ceux des transactions courtes qui caractérisent les SGBD classiques.
 - Dans un SGBDOO, l'unité de contrôle de concurrence et de récupération est l'objet.
 - Dans un SGBDOO, les protocoles basés sur le verrouillage sont utilisés. Cependant ceux-ci ont montré leur limites dans le cas de transactions longues.



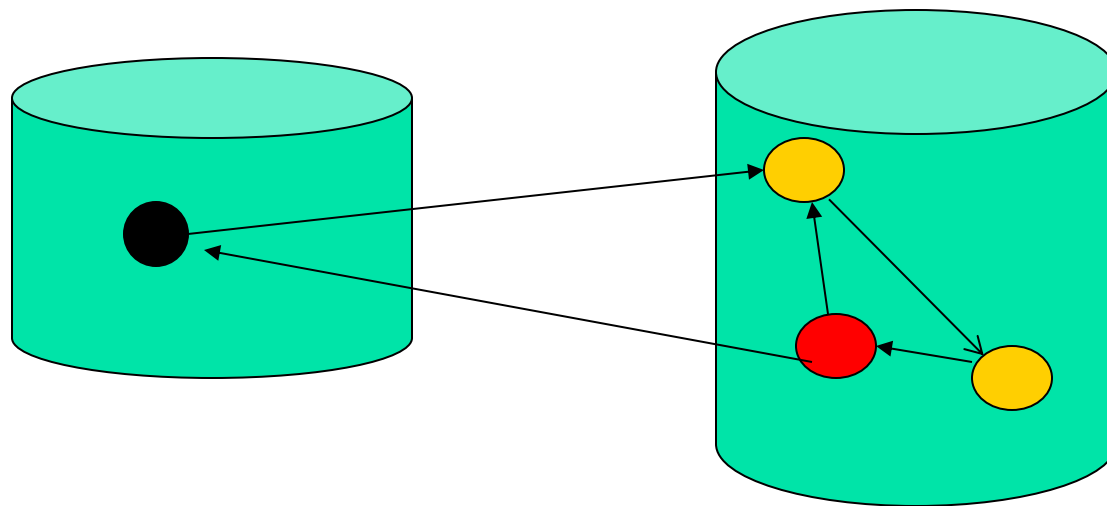
Notions spécifiques aux SGBDOO : transactions longues, la gestion de versions, l'évolution de schémas

■ Versions :

- Le processus de conservation de l'évolution des objets s'appelle la gestion des versions.
- Une version d'un objet représente un état identifiable de l'objet.
- Un historique des versions représente l'évolution de l'objet



Gestion de versions



● Version transitoire

● Version opérationnelle

● Version officielle

Base de données
publique

Espace de travail
privé



Notions spécifiques aux SGBDOO

■ Evolution de schéma :

- Modification de la définition de classes
 - Modifier les attributs
 - Modifier les méthodes
- Modification de la hiérarchie d'héritage
 - Rajouter une superclasse
 - Supprimer une classe de la liste des superclasses
 - Modifier l'ordre des superclasses
- Modification sur l'ensemble des classes :
création, suppression de classes, modification des
noms des classes.



SGBD relationnels objet

- Les SGBDOO n'ont pas eu dans le commerce l'engouement qu'avaient eu avant eux les SGBD relationnels.
- SGBDRO : système hybride entre SGBDR et SGBDOO.
- SGBDRO préserve le capitale de connaissances des SGBDR
- SGBDRO offre certaines fonctionnalités des SGBDOO pour les applications évoluées.
- Extension de la norme SQL : SQL 99 et SQL 2003.



Avantages des SGBDRO

- Résolution des faiblesses des SGBDR (données simples,)
- Prise en compte de la réutilisation et du partage par l'offre de fonctionnalités standard de manière centralisée au niveau du serveur de SGBD. (dans le SGBDOO, codée dans chaque application)
- Préservation du noyau de connaissances et d'expérience investi dans le développement d'applications relationnelles.



Inconvénients des SGBDRO

- Accroissement des coûts et de la complexité
- Simplicité et normalité du modèle relationnel disparaît (reproche des puristes du modèle relationnel)
- La complexité excessive pour étendre le modèle relationnel ne satisfait pas les puristes du modèle objet.



Manifeste des bases de données de troisième génération [Stonebraker 1990]: quelques éléments

1. Un SGBD de 3ième génération doit posséder un système de type riche
2. L'héritage est une bonne idée
3. Les fonctions, y compris les procédures et les méthodes de BD, ainsi que l'encapsulation sont de bonnes idées
4. Des identificateurs uniques d'enregistrements ne doivent être affectés automatiquement par le SGBD que si une clé primaire définie par l'utilisateur n'est pas disponible.
5. L'essentiel de tout accès programmé à une BD doit être effectué par le biais d'un langage non procédural et de haut niveau
6. Les vues modifiables sont essentielles et non contournables.