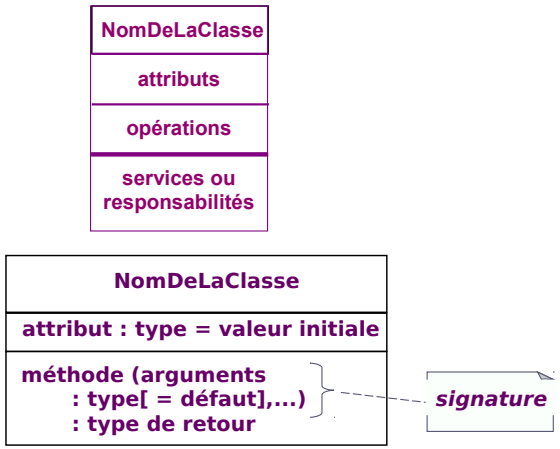
- *Diagramme de classes*
- *Diagramme de classes participantes*
- *Diagramme d'états - transitions*
- *Diagramme de classes vers le modèle relationnel*

DIAGRAMME DE CLASSES

Diagramme statique ou de structure

Représentation des classes



Représentation des classes

Visibilité des attributs et des opérations

ATTRIBUT DÉRIVÉ

- Un **attribut dérivé** est un attribut dont la **valeur** peut être **déduite** d'autres informations disponibles dans le modèle, par exemple d'autres attributs de la même classe, ou de classes en association.

- L'analyste garde cet attribut, pouvant être considéré comme redondant, s'il correspond à un concept important aux yeux de l'expert métier.

- **Exemple** :

L'*âge* d'une personne est un attribut dérivé de l'attribut *date de naissance*.

Notation : /âge

ATTRIBUT DE CLASSE

Par défaut, un attribut a une portée d'instance : chaque objet de la classe possède sa propre valeur pour la propriété.

Dans certains cas, l'attribut peut avoir une **portée de classe** : il existe alors une seule valeur commune de la propriété pour toutes les instances de la classe.

On parle dans ce cas d'attribut de classe, et on le souligne pour le distinguer des attributs d'instance.

attribut
montant facture → *portée d'instance*

attribut classe
TVA → *portée de classe*

ATTRIBUT DE CLASSE

- **Exemple** : la valeur de la variable PI = *3,14* définie dans la classe Math du langage Java.

- Cette valeur est accessible même s'il n'existe aucun objet de la classe Math.

- PI est un attribut de la classe Math et non un attribut de ses instances. Les instances ont accès à cet attribut, mais n'en possèdent pas une copie.

- En Java comme en C++, un attribut de classe s'accompagne du mot-clé *static*.

CONTRAINTES ATTRIBUT

Unicité : les valeurs de l'attribut n'ont pas de doublons.

{unique}
{not unique}

On peut trouver cette contrainte pour les valeurs retournées par une méthode.

Personne
- noSécurité : int {unique} - nom : String - prénoms : String[1..3] - dateNaissance : Date
+getPrénoms () : String[1..3] {unique, ordered}

CONTRAINTES ATTRIBUT

Types collection :

Bag ou Sac (*pas ordre, pas unicité*)
OrderedSet ou Ensemble ordonné (*ordre, unicité*)
Set ou Ensemble (*pas ordre, unicité*)
Sequence ou Séquence (*ordre, pas unicité*)

RegistreDesVotants
- votants : Personne [*] {OrderedSet}

Chaque votant n'est autorisé qu'à voter une seule fois.
Les votants sont classés par ordre alphabétique.

CONTRAINTES ATTRIBUT

frozen : attribut non modifiable une fois initialisé

readOnly : attribut modifiable à l'intérieur de la classe
mais ne peut être changé de l'extérieur de la classe

Contraintes diverses : {attribut > 0}, {précondition}

Les gens commentent des erreurs. Si on gèle un attribut tel que la date de naissance, on ne pourra pas le modifier si une erreur de saisie a été commise.

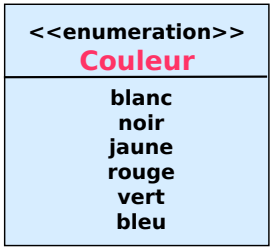
CONTRAINTES ATTRIBUT

Employé
n°employé : int {unique}{frozen} poste : String salaire : real {readOnly}

Le salaire ne peut être modifié à l'extérieur de la classe Employé.

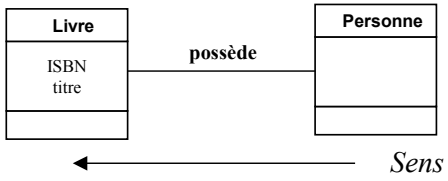
ÉNUMÉRATION

- Une **énumération** ou **type énuméré** est un type possédant un nombre fini et arbitraire de valeurs possibles.
- Classeur stéréotypé <<enumeration>>
- **Exemples** : les jours de la semaine, les mois.



IDENTIFICATION DES CONCEPTS

Exemple :
Une personne peut posséder des livres. La relation *possède* est une association entre les classes Personne et Livre.



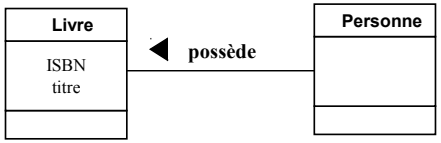
Par défaut bidirectionnelle

Sens lecture privilégié

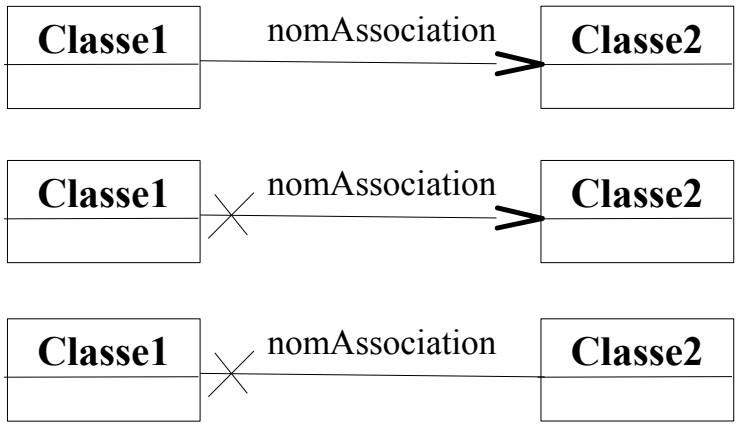
Même si le **verbe** qui nomme une association semble **privilégier un sens de lecture**, une **association** entre concepts dans un modèle du domaine est **par défaut bidirectionnelle**.
Donc, implicitement, l'exemple précédent inclut également le fait qu'un livre est possédé par une personne.

SENS DE LECTURE DES ASSOCIATIONS

- On peut faciliter la lecture des diagrammes en indiquant le sens de lecture des associations par le symbole : ▶



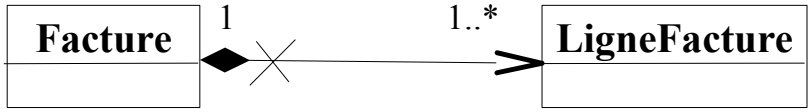
NAVIGABILITÉ DES ASSOCIATIONS



même sémantique

NAVIGABILITÉ DES ASSOCIATIONS

Exemple

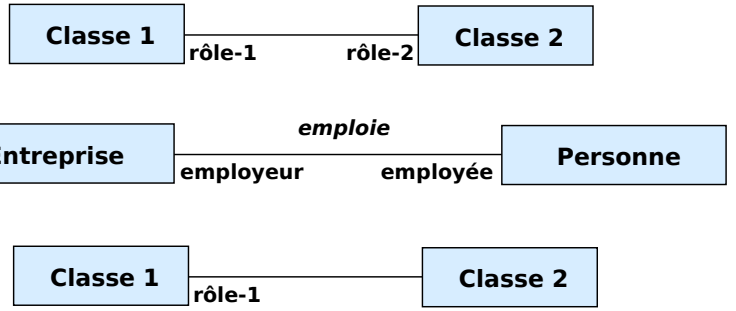


ASSOCIATION DÉRIVÉE

Les **associations dérivées** obéissent au même principe que les attributs dérivés : ce sont des associations redondantes mais considérées comme pertinentes par l'expert du domaine.

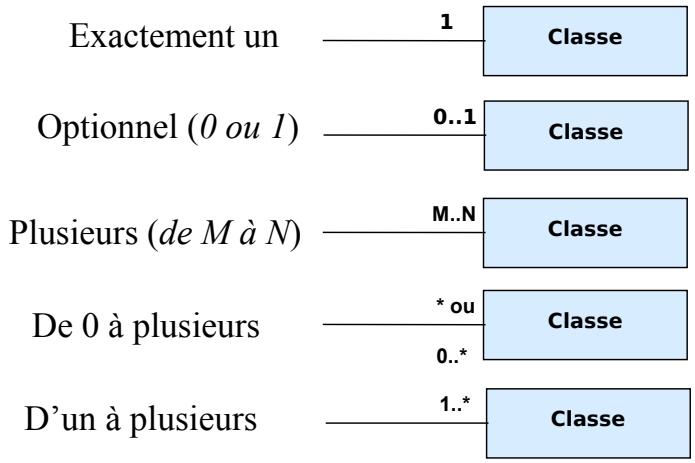
RÔLE

- Le **rôle** décrit comment une classe voit une autre classe au travers d'une association.
- Se place à une extrémité de l'association



CARDINALITÉS

Montrent combien d'objets d'une classe peuvent être liés à un objet de l'autre classe.

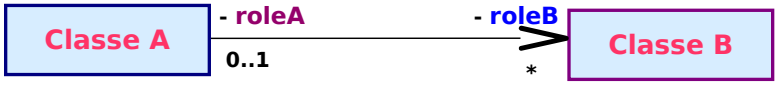


IMPLÉMENTATION


```
public class ClasseA {
  private Set<ClasseB> roleB ;
  ...
}
```

```
public class ClasseB {
  private ClasseA roleA ;
  ...
}
```

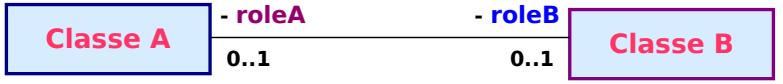

IMPLÉMENTATION




```
public class ClasseA {
  private Set<ClasseB> roleB ;
  ...
}
```

```
public class ClasseB {
  // ne connaît pas ClasseA
  ...
}
```


IMPLÉMENTATION



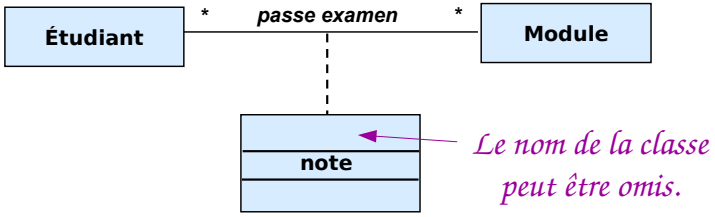

```
public class ClasseA {
  private ClasseB roleB ;
  ...
}
```

```
public class ClasseB {
  private ClasseA roleA ;
  ...
}
```

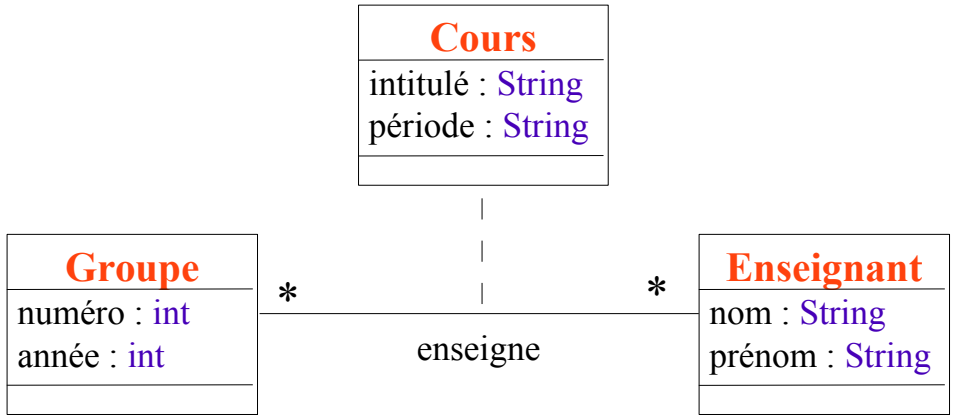
ASSOCIATION ATTRIBUÉE

- Une **association attribuée** est une association qui contient des attributs sans participer à des relations avec d'autres classes.

- Une association peut être raffinée et avoir ses propres propriétés, qui ne sont disponibles dans aucune des classes qu'elle lie.

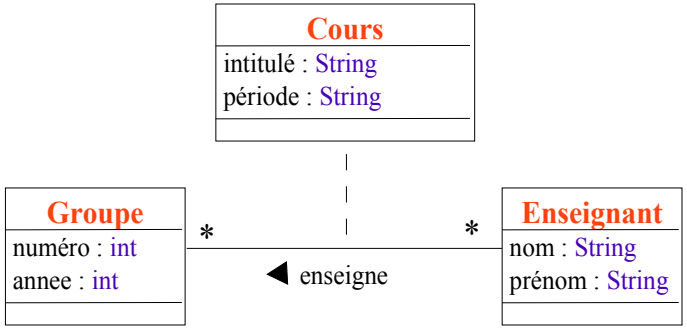


ASSOCIATION ATTRIBUÉE



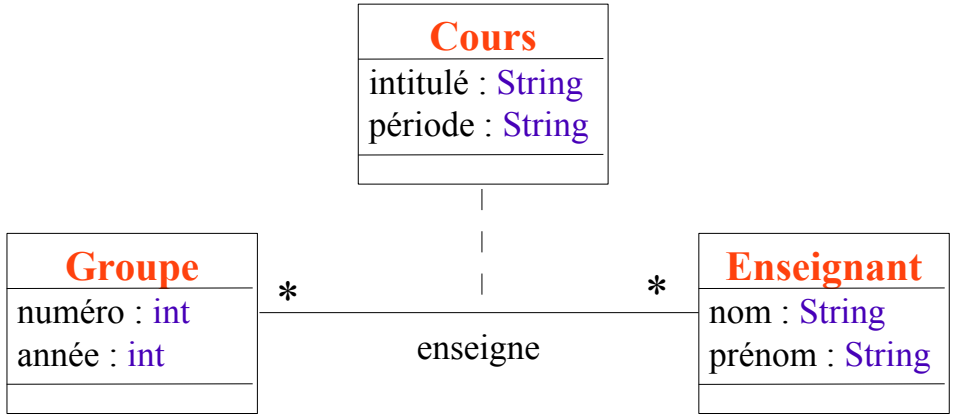
Quelle est la contrainte sur Cours ?

SOLUTION



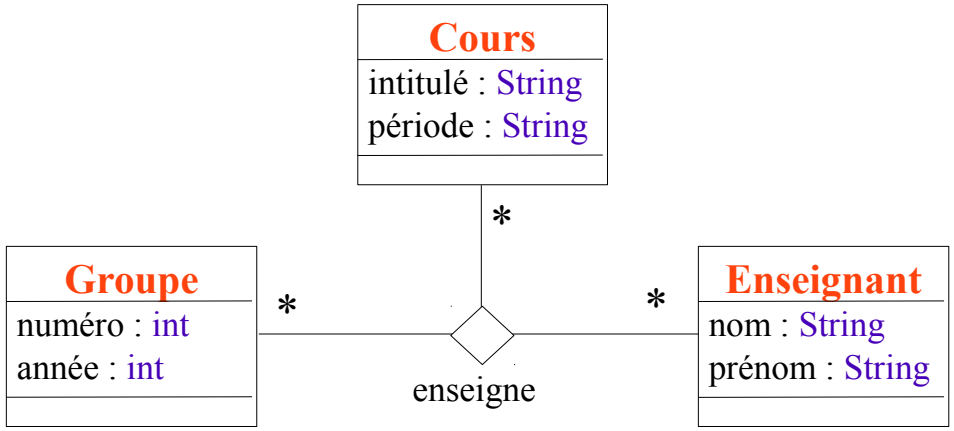
Un cours ne peut exister sans un lien Groupe – Enseignant
Un enseignant ne peut enseigner qu'un cours à un groupe donné

ASSOCIATION ATTRIBUÉE

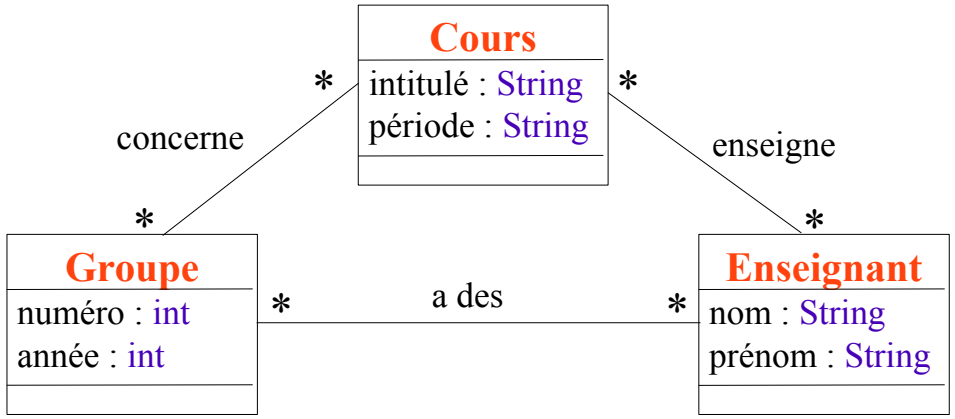


Comment y remédier ?

SOLUTION

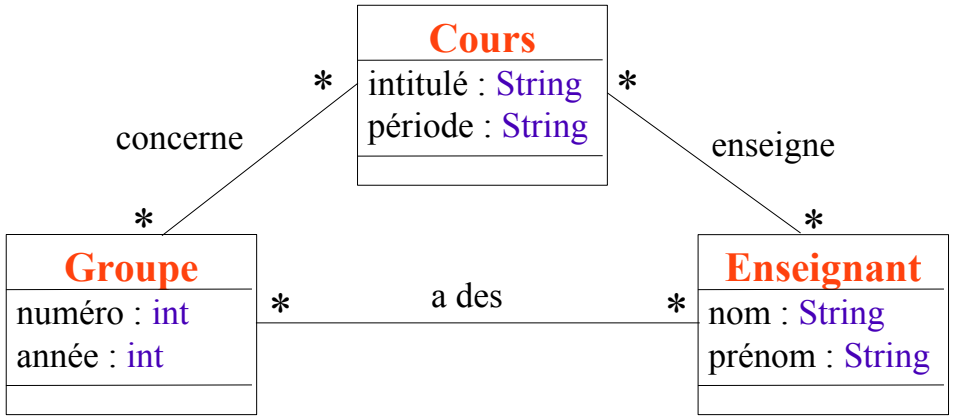


ASSOCIATION ATTRIBUÉE



Que pensez-vous de ce modèle ?

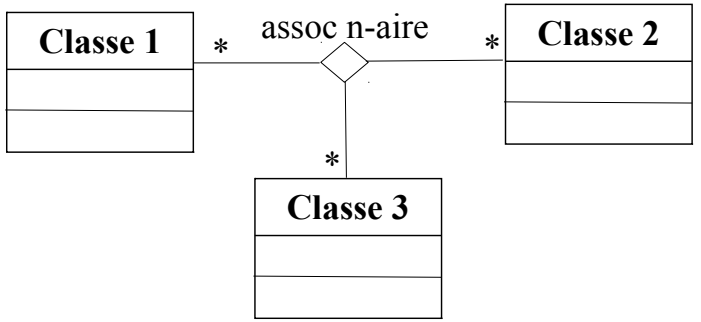
SOLUTION



On ne sait pas quels groupes a un enseignant dans un cours donnée.

ASSOCIATION N-AIRE

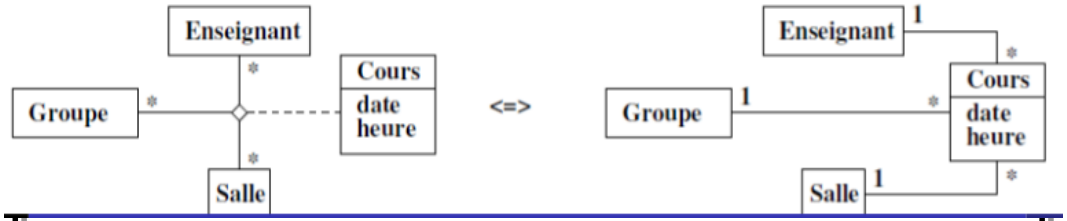
- Une **association n-aire** lie plus de *2* classes.



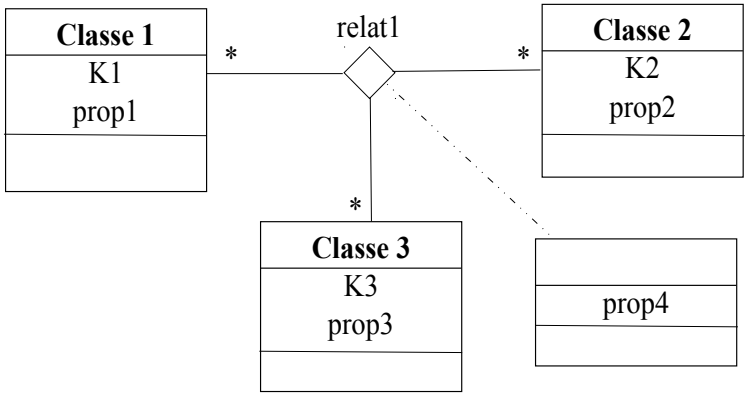
- Les multiplicités en UML sont
  - ✓ "*à l'envers*" (*par référence à Merise*) pour les associations binaires et
  - ✓ "*à l'endroit*" pour les associations n-aires (*n* > *2*).

ASSOCIATION N-AIRE

- Association n-aire quand on ne peut pas faire autrement.
- Privilégier toujours les associations binaires.



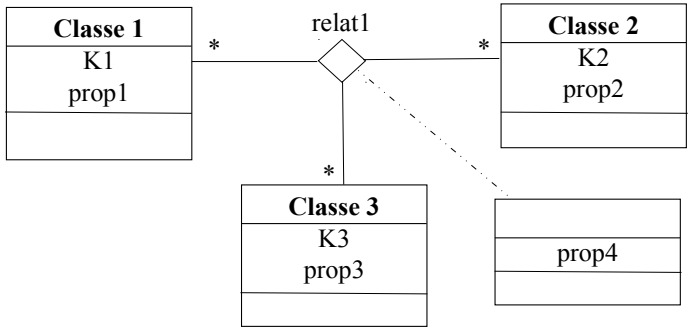
ASSOCIATION N-AIRE



Quelle contrainte a-t-on avec la propriété "prop4" ?

K1,K2,K3 → prop4

SOLUTION



Pour un triplet (K1, K2, K3), si K1, K2 et K3 sont des identifiants respectivement des classes Classe1, Classe2 et Classe3,

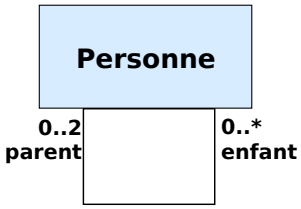
il n'y a qu'une valeur pour "prop4"

Privilégier
les associations binaires
aux associations n-aires

- ✦ plus simples à interpréter
- ✦ plus simples à implémenter
- ✦ autorisent la qualification
- ✦ autorisent la navigation

ASSOCIATION RÉFLEXIVE

Les associations peuvent relier une classe à elle-même.
Le nommage des rôles est ici très important pour distinguer les instances qui participent à la relation.

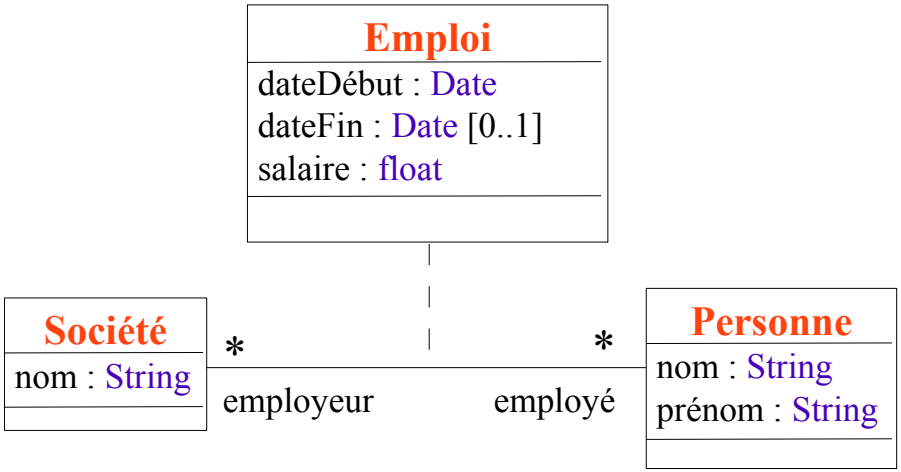


Une personne (*parent*) a combien d'enfants ? : *0* à plusieurs.
Une personne (*enfant*) a combien de parents ? : *0* à *2*

EXERCICE

Modéliser les derniers différents emplois d'un salarié dans ses diverses entreprises

SOLUTION

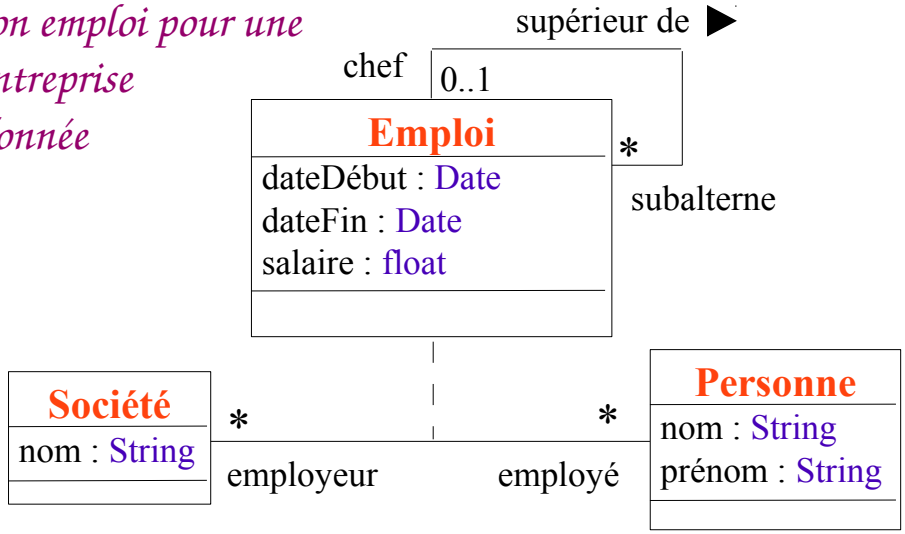


EXERCICE

Rajouter le concept de supérieur hiérarchique

SOLUTION

Une personne est le supérieur d'une autre dans le cadre de son emploi pour une entreprise donnée

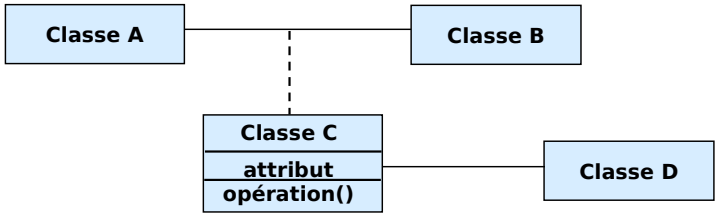


CLASSE D'ASSOCIATION

Classe d'association : association promue au rang de classe.
Elle possède tout à la fois les caractéristiques d'une association et d'une classe et peut donc porter des attributs qui se valorisent pour chaque lien.

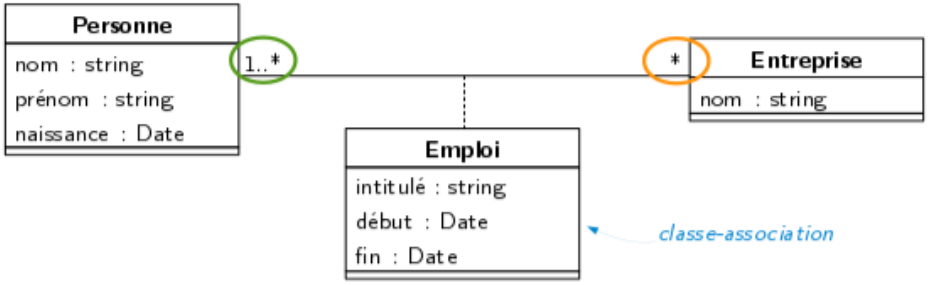
CLASSE D'ASSOCIATION

Une association peut être représentée par une classe (*classe C*) pour ajouter des attributs et des opérations dans l'association.

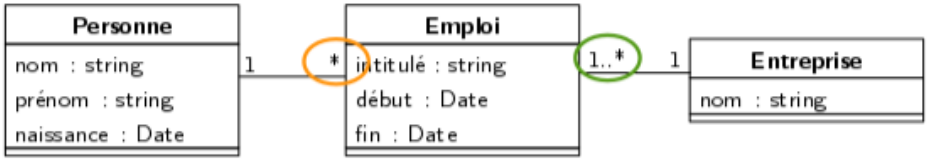


C'est une classe comme les autres et donc elle peut avoir d'autres relations (*avec la classe D*).

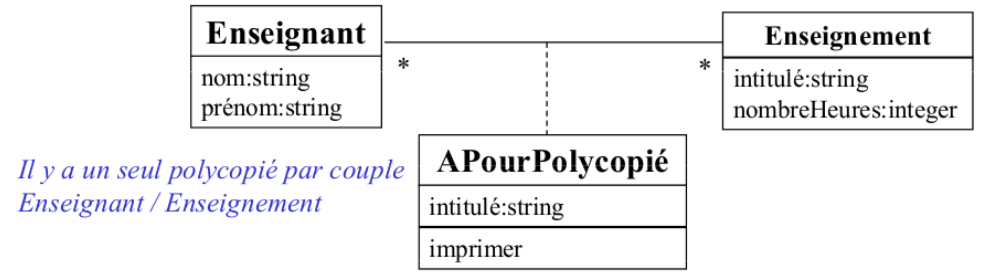
CLASSE D'ASSOCIATION



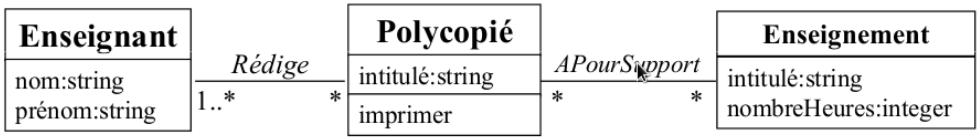
Équivalence en termes de classes et d'associations



Une classe d'association peut être remplacée par une classe intermédiaire qui sert de pivot.



Il y a un seul polycopié par couple Enseignant / Enseignement



Un nombre quelconque d'occurrences de Polycopié pour chaque Enseignant et chaque Enseignement

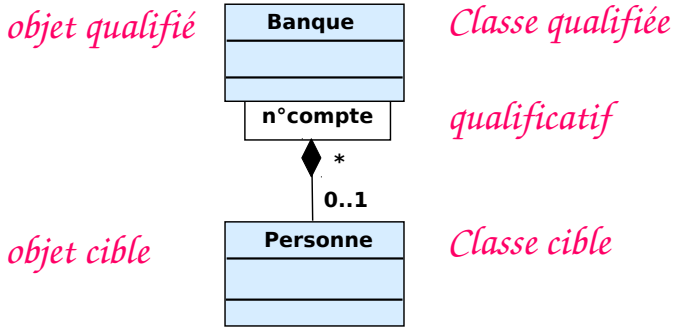
QUALIFICATIF

- Soit une association binaire qui fait correspondre un objet à un ensemble d'objets liés.
- Parfois, il peut être préférable de sélectionner un objet dans cet ensemble en fournissant une valeur qui permette de distinguer les objets dans l'ensemble.
- Il peut s'agir d'un attribut de la classe cible mais en général, c'est un attribut d'association dont la valeur est fournie par le créateur lorsqu'il ajoute un nouveau lien.
- Dans une association binaire, un tel attribut s'appelle un qualificatif.

QUALIFICATIF

- Un objet associé à une valeur qualifiée détermine un objet lié unique ou (*moins souvent*) un sous-ensemble d'objets liés.
- Cette valeur qualifie l'association.
- Dans un contexte d'implémentation, on peut qualifier ces attributs de valeur d'index.
- Un qualificatif permet de sélectionner un ou des objet(s) dans l'ensemble des objets reliés à un objet (*appelé* ***objet qualifié***) par une association.

QUALIFICATIF



- L'objet sélectionné par la valeur du qualificatif est appelé objet cible.
- Un qualificatif agit toujours sur une association dont la multiplicité est *many* dans la direction cible.

QUALIFICATIF

- Dans le cas le plus simple, chaque valeur de qualificatif sélectionne un seul objet dans l'ensemble cible des objets liés.

- En d'autres termes, un objet qualifié et une valeur de qualificatif génèrent un objet cible lié unique.

- On peut modéliser un tableau comme une association qualifiée. Le tableau est l'objet qualifié, l'index du tableau est le qualificatif et l'élément du tableau est l'objet cible.

- Le qualificatif sert de sélecteur dans l'ensemble des objets reliés par l'association. Il scinde l'ensemble en sous-ensembles par valeur de qualificatif.

QUALIFICATIF

- Dans la plupart des cas, l'objectif d'un qualificatif est de sélectionner un objet unique dans l'ensemble des objets liés pour qu'une association qualifiée puisse se comporter comme une table de consultation.

- Un qualificatif possède un nom et un type mais pas de valeur initiale car les qualificatifs ne sont pas des objets indépendants et chaque valeur de qualificatif doit être explicite lorsqu'on ajoute un lien à l'association.

- On n'utilise pas de qualificatifs dans des associations n-aires.

QUALIFICATIF

- La classe qualifiée et le qualificatif forment à eux deux une valeur composite reliée à la classe cible.

- Les valeurs de multiplicité habituelles sont :
  - 0..1 : on peut sélectionner un objet unique mais toute valeur de qualificatif associée à l'objet qualifié ne sélectionne pas nécessairement un objet
  - 1 chaque valeur de qualificatif associée à l'objet qualifié sélectionne un objet cible unique. **Le domaine des valeurs de qualificatif doit être fini**.
  - * la valeur de qualificatif est un index qui partitionne les objets cibles en sous-ensembles.

QUALIFICATIF

- Dans l'autre direction de l'association qualifiée (*cible vers objet qualifié*), la multiplicité indique le nombre de paires (*objet qualifié, qualificatif*) qu'on peut relier à l'objet cible, et non pas le nombre d'objets qualifiés.

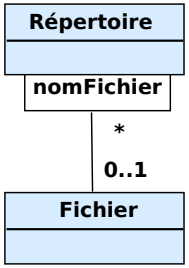
- Sur une association qualifiée, les multiplicités sont traitées comme si l'objet qualifié et le qualificatif étaient une seule entité, une clé composite.

- La valeur du qualificatif est une propriété de lien et non pas de l'objet cible.

QUALIFICATIF

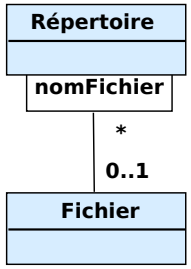
- Soit un système de fichiers UNIX dans lequel chaque répertoire est une liste d'entrées avec des noms. On peut utiliser les mêmes noms dans d'autres répertoires. Chaque entrée pointe vers un fichier qui peut être un fichier de données ou un autre répertoire. Plusieurs entrées peuvent pointer vers le même fichier (*fichier possédant plusieurs liens symboliques*).

Le répertoire qualifié par le nom du fichier génère un fichier.



Le nom du fichier ne fait pas partie du fichier. Il appartient à la relation entre un répertoire et un fichier. Un fichier ne possède pas de nom unique.

QUALIFICATIF



(répertoire, nomFichier) → 0 ou 1 fichier
répertoire → plusieurs fichiers
fichier → plusieurs (répertoire, nomFichier)
fichier → plusieurs répertoires
fichier → plusieurs nomFichiers

QUALIFICATIF

- Une association qualifiée est une table de consultation.

- On implémente les tables de consultation comme des tables de hachage, des arborescence-B et des listes triées.

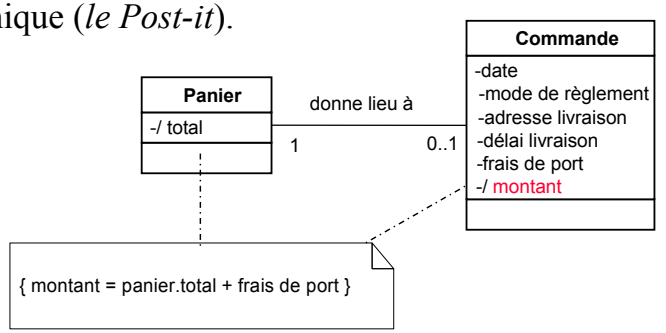
- Une association qualifiée désigne une structure de données indexée optimisée pour la consultation fondée sur la valeur qualifiée.

- Un qualificatif d'attribut ne doit généralement pas être inclus comme attribut de la classe cible car sa présence dans l'association suffit.

CONTRAINTE

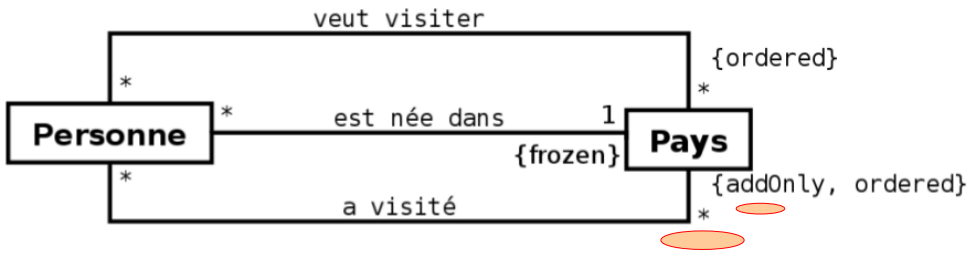
Une **contrainte** est une condition entre éléments du modèle, qui doit être vérifiée par les éléments concernés.

Elle est définie entre accolades { }, et est insérée dans une note graphique (*le Post-it*).



CONTRAINTE

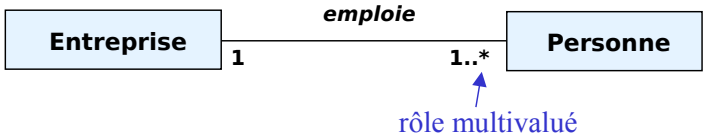
Exemple



On peut ajouter de nouveaux liens
mais pas en supprimer

RÔLE MULTIVALUÉ

- Un **rôle multivalué** est un rôle dont la multiplicité a une valeur maximale supérieure strictement à *1* (* *par exemple*).



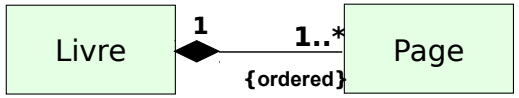
- La convention usuelle est de considérer les rôles multivalués comme des ensembles. Les objets ne sont pas ordonnés, et aucun objet n'apparaît plus d'une fois dans l'ensemble.

- On peut modifier cette convention en attachant une contrainte au rôle.

CONTRAINTE D'ORDRE

- La contrainte {ordered} peut être placée sur le rôle pour spécifier qu'une relation d'ordre décrit les objets placés dans la collection.

- On ne spécifie pas comment les éléments sont ordonnés (*numéro, ordre alphabétique, etc*) car c'est un choix de conception, mais seulement que l'ordre doit être maintenu durant l'ajout et/ou la suppression des objets.



CONTRAINTE D'IMMUABILITÉ

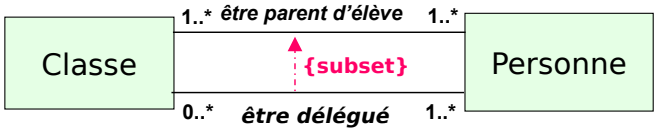
- La contrainte {frozen} (*gelé*) peut être placée sur un attribut, une extrémité d'association ou une classe.

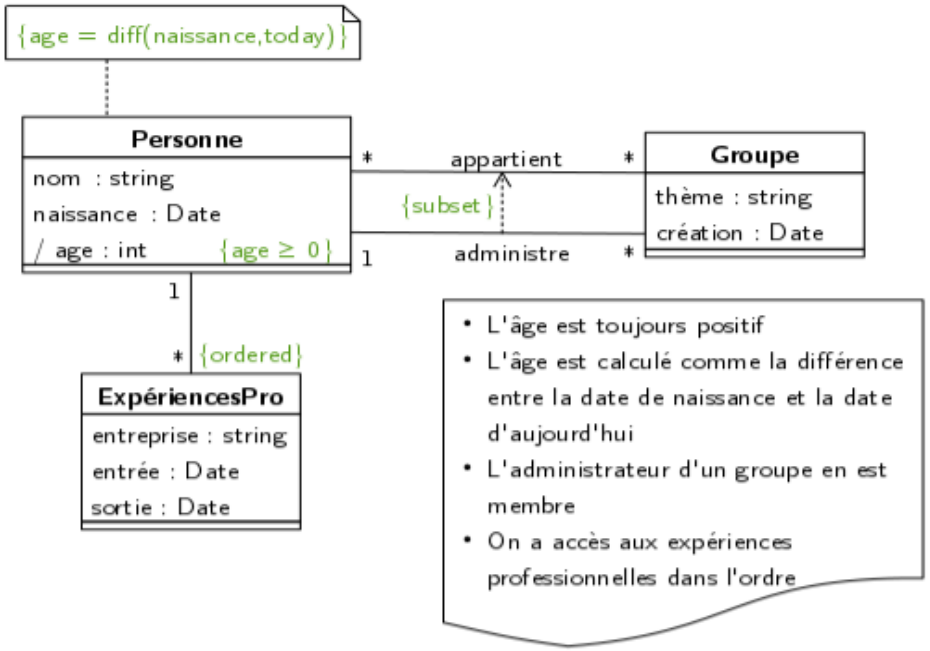
- Sur un attribut ou une extrémité d'association, la contrainte d'immuabilité indique que la valeur de cet attribut ou de cette association est immuable pendant toute la durée de vie de l'objet. Elle doit être définie à la création de l'objet et ne peut jamais être modifiée.

- Appliquée à une classe, la contrainte d'immuabilité indique que toutes les terminaisons d'association et tous les attributs de la classe sont gelés.

CONTRAINTE DE SOUS-ENSEMBLE

- La contrainte de *sous-ensemble* {subset} indique qu'une collection est incluse dans une autre collection.

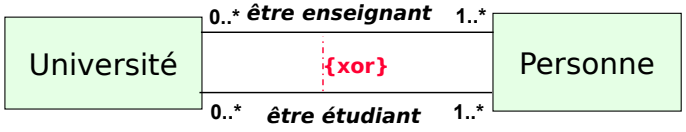




CONTRAINTE DU OU EXCLUSIF

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

- Cette contrainte évite l'introduction de sous-classes artificielles pour matérialiser l'exclusivité.



AGRÉGATION

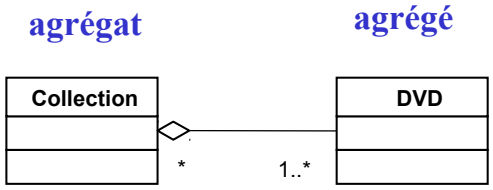
- Si l'une des classes joue le rôle d'**ensemble composé d'instances** de l'autre classe, on utilise l'**agrégation**.

- Une agrégation n'est plus sémantiquement symétrique puisqu'elle privilégie l'une des deux classes en l'élevant au rang de conteneur.

- Les agrégations n'ont pas besoin d'être nommées : implicitement, elles signifient *"contient"*, *"est composé de"*.

AGRÉGATION

- Forme particulière d'association entre un tout et ses parties, dans laquelle le tout est composé de parties.
- Association non symétrique, une extrémité joue un rôle prédominant par rapport à l'autre.



AGRÉGATION

- L'**agrégation** garde cependant les **propriétés d'une association** et n'influe :

  ✓ ni sur l'expression des multiplicités,
  ✓ ni sur la navigabilité,
  ✓ ni sur le cycle de vie des instances reliées.

- Par conséquent, il est possible de partager l'agrégation : une partie peut appartenir simultanément à plusieurs agrégats.

COMPOSITION

La **composition** est une **agrégation particulière** où l'objet ne peut appartenir qu'à un seul composite **et** le cycle de vie est très imbriqué.

(*la fermeture de l'entreprise entraîne la suppression des services*).

COMPOSITION

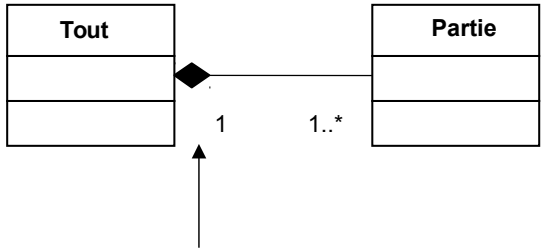
Avec une composition, on introduit les deux caractéristiques suivantes :

1 - la composition n'est pas partageable : un objet ne peut appartenir qu'à un seul composite à la fois.

2 - le cycle de vie des parties est forcément lié à celui du composite : la destruction du composite entraîne en particulier la destruction de ses parties.

COMPOSITION

La composition implique une contrainte sur la valeur de la multiplicité du côté du composite : elle ne peut prendre que les valeurs 0 ou 1.



Cardinalités : 0..1 ou 1 seulement

0 en cardinalité minimale : attribut non renseigné

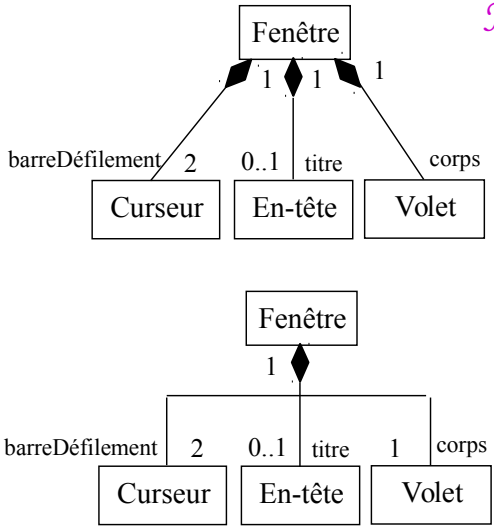
COMPOSITION

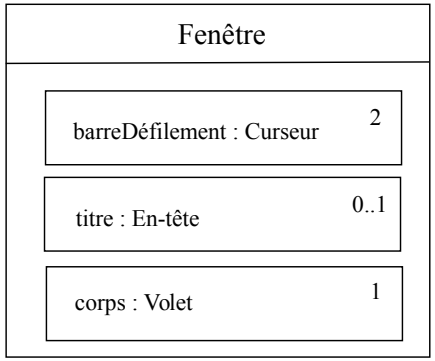
- La composition et les attributs sont sémantiquement équivalents, leurs représentations graphiques sont interchangeables.

- La notation par composition s'emploie dans un diagramme de classes lorsqu'un attribut participe à d'autres relations dans le modèle.

COMPOSITION

Notations





HÉRITAGE

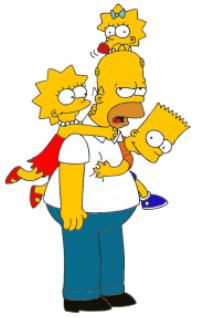
- C'est le procédé de partage ou de factorisation de l'information dans les technologies à objets

ou

- Mécanisme de dérivation d'une nouvelle classe ou d'un nouveau objet à partir d'une classe existante.

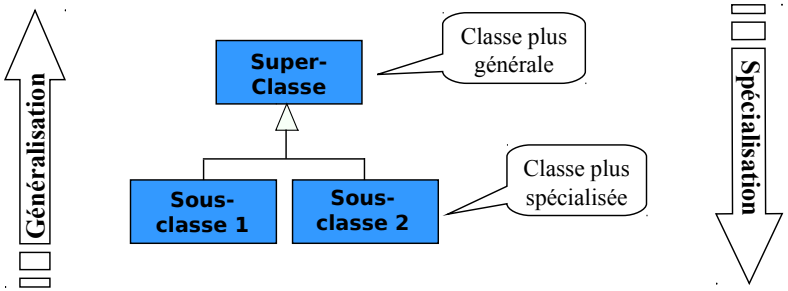
- L'idée provient de l'existence d'une hiérarchie de classes.

Comportement
Apparence
Attitude



HÉRITAGE

- Hiérarchie de classes : super-classe et sous-classe(*s*).

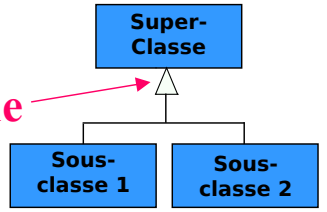


- **Généralisation** : factorisation des attributs et des opérations communs à plusieurs classes dans une classe plus générale.

- **Spécialisation** : raffinement d'une classe par création d'une classe plus spécialisée.

HÉRITAGE

Pas de flèche mettre un triangle



super-classe
classe de base
classe mère

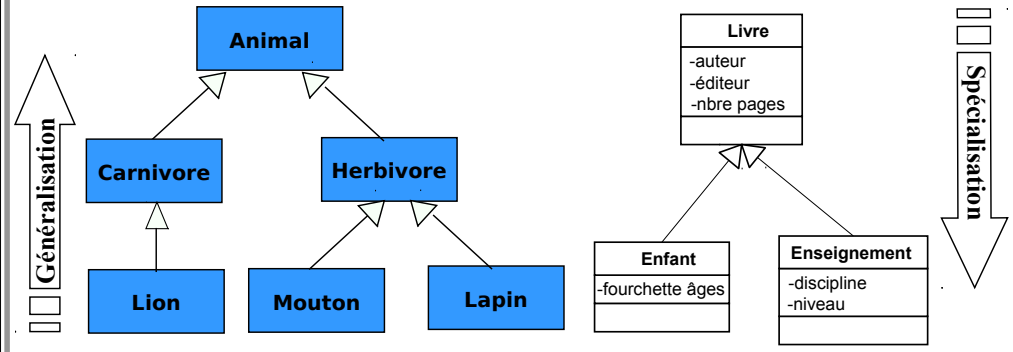
sous-classe
classe dérivée
classe fille

- La classe dérivée **est une** version spécialisée de sa classe de base.

- Relation "**est-un**" pour traduire le principe de généralisation / spécialisation.

HÉRITAGE

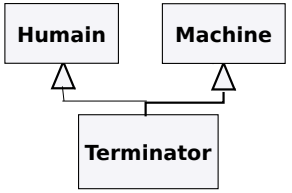
Exemples



Chaque instance d'une sous-classe est aussi une instance de toutes les super-classes.

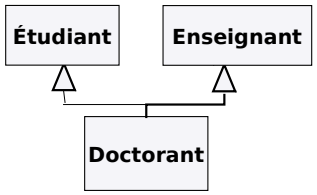
HÉRITAGE

Héritage multiple



HÉRITAGE

Héritage multiple



HÉRITAGE

- Un objet hérite les attributs et les méthodes de la classe dont il est issu

  ou

  la classe spécialisée hérite des propriétés qu'elle n'a pas substituées.

- Une sous-classe peut ajouter des attributs et/ou des méthodes.

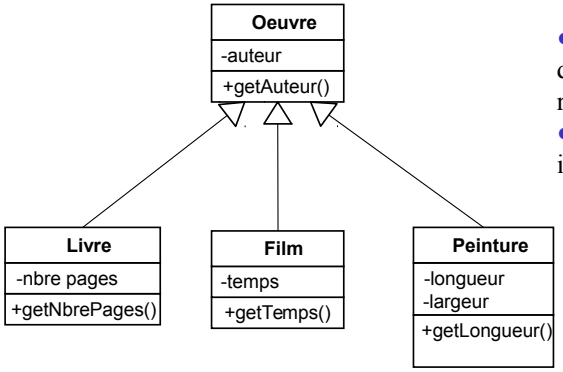
- Une sous-classe peut redéfinir une méthode par surcharge.

Comportement
Apparence
Attitude

HÉRITAGE

Exemple

- Une œuvre est la composition d'un artiste. Parmi les œuvres, on peut citer les livres qui ont un certain nombre de pages, les films qui durent un certain temps, les peintures d'une certaine taille …
- Lorsqu'on dit tout livre est une œuvre, et qu'une œuvre a un auteur, on veut se passer de redire qu'un livre a un auteur.



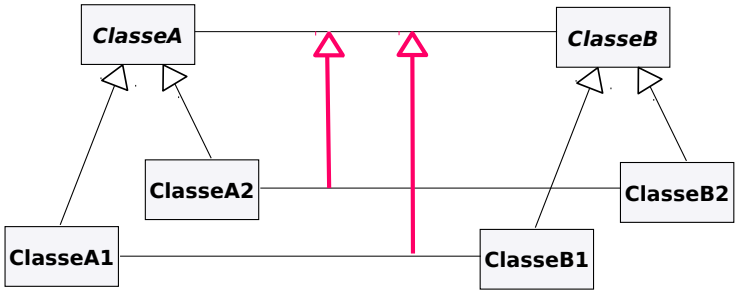
- Le but est de ne pas écrire deux fois la même chose : réutiliser le code existant.
- Éviter de faire des bêtises identiques à plusieurs endroits.

SPÉCIALISATION DE L'ASSOCIATION

- Généralisation peu courante.

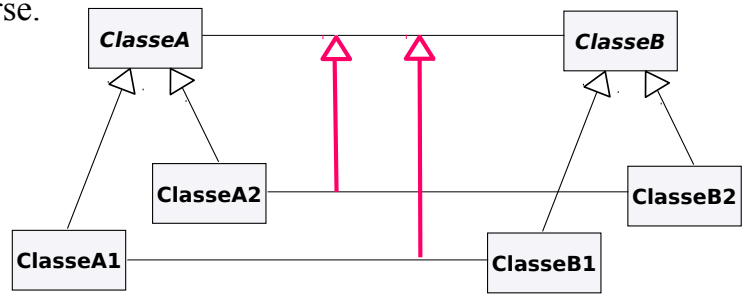
- L'élément enfant doit apporter des contraintes supplémentaires au parent et constituer un sous-ensemble de l'extension du parent.

- Une association enfant est plus contrainte que son parent.



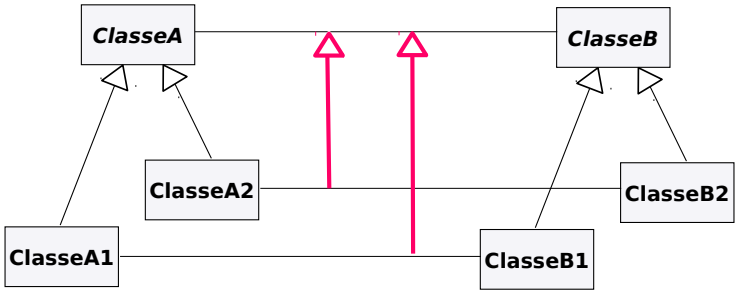
SPÉCIALISATION DE L'ASSOCIATION

- Définir des sous-ensembles de l'étendue signifie que chaque lien de l'association enfant est un lien de l'association parent, mais pas l'inverse.



- Tout lien connectant ClasseA1 et ClasseB1 connectera également *ClasseA* et *ClasseB*, mais tous les liens connectant *ClasseA* et *ClasseB* ne se connecteront pas à ClasseA1 et ClasseB1.

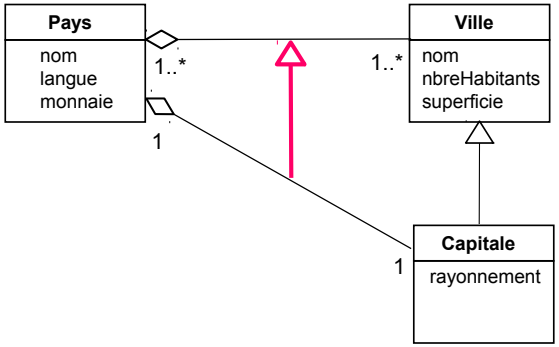
SPÉCIALISATION DE L'ASSOCIATION



- L'association générale peut être considérée comme une **association abstraite** tandis que les deux **associations** enfant sont **concrètes**.
- **Pattern** de hiérarchies de classes par paires connectées par des associations est **relativement courante** dans la généralisation d'associations.

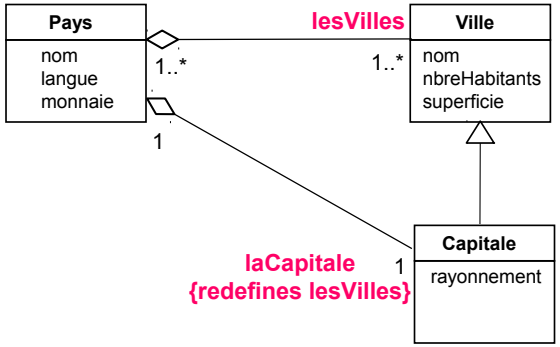
SPÉCIALISATION DE L'ASSOCIATION

- La distinction entre la définition de sous-ensembles et la spécialisation d'une association n'est pas clairement décrite dans la spécification UML2.



SPÉCIALISATION DE L'ASSOCIATION

- Peu implémenté par les outils du marché, d'où la notion de redéfinition de propriété. Le mot-clé *redefines* remplace le symbole de spécialisation entre associations



CONTRAINTE ENTRE SOUS-CLASSES

- {disjoint} ou {exclusif} : une classe descendante d'une classe A ne peut être descendante que d'une seule sous-classe de A (*défaut*).

- {chevauchement} ou {inclusif} : une classe descendante d'une classe A appartient au produit cartésien des sous-classes de la classe A. Un objet concret est alors construit à partir d'une classe obtenue par mélange de plusieurs super-classes.

- {incomplète} indique une généralisation extensible.

- {complète} indique qu'une instance est forcément d'une des sous classes (*la super classe est alors abstraite*).

CONTRAINTE ENTRE SOUS-CLASSES

Il est possible de représenter un certain nombre de contraintes d'intégrité entre sous-classes d'objets.

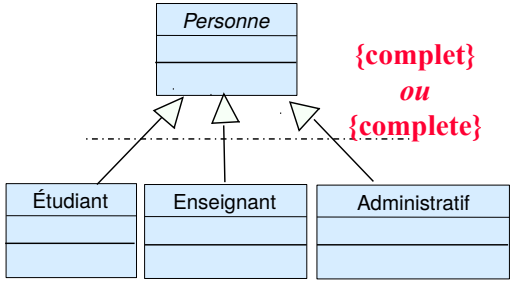
Ces contraintes correspondent aux différentes combinaisons possibles autour des contraintes de couverture et de disjonction.

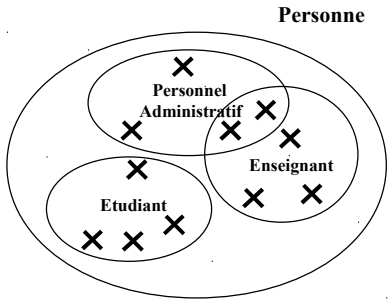
Contrainte de couverture
Contrainte de disjonction

CONTRAINTE DE COUVERTURE

Contrainte de couverture : tout objet de la super-classe doit appartenir à au moins l'une des sous-classes.
(*liste exhaustive de classes*)

Par défaut : incomplet



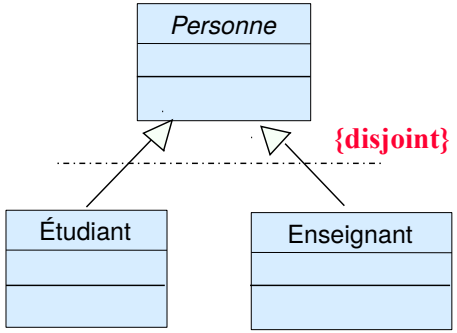


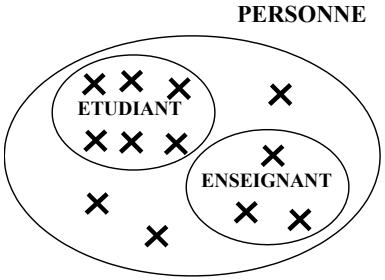
non couverture : {incomplete}

CONTRAINTE DE DISJONCTION

Contrainte de disjonction : tout objet de la super-classe doit appartenir à une seule sous-classe : les sous-classes sont mutuellement exclusives.

Par défaut : disjoint





non disjonction : {overlapping}

STRUCTURATION EN PAQUETAGES

Paquetage = package en anglais

La structuration d'un modèle est une activité délicate. Elle doit s'appuyer sur deux principes fondamentaux :

1- cohérence

2- ***indépendance***.

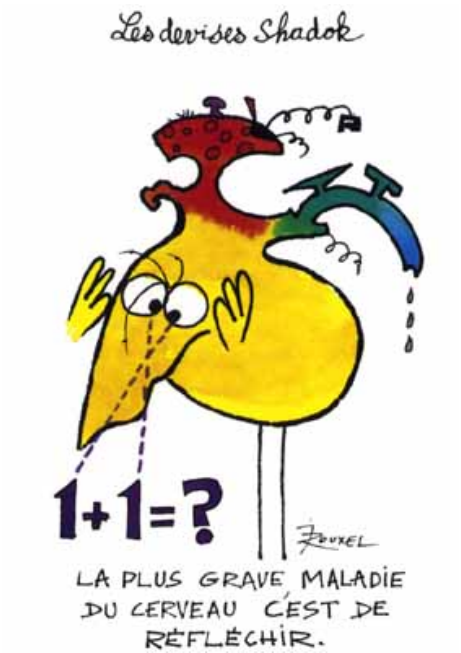
STRUCTURATION EN PAQUETAGES

- Le **premier principe** (*cohérence*) consiste à regrouper les classes qui sont proches d'un point de vue sémantique.
Un critère intéressant consiste à évaluer les durées de vie des instances de concept et à rechercher l'homogénéité.

- Le **deuxième principe** (*indépendance*) s'efforce de minimiser les relations entre paquetages, c'est-à-dire plus concrètement les relations entre classes de paquetages différents.

Diagramme de classes participantes

DIAGRAMME DE CLASSES PARTICIPANTES



SOMMAIRE

INTRODUCTION

- Identification des classes d'analyse participant à la réalisation des cas d'utilisation.

- ***3* types de classes d'analyse** :

  ✓ ***Dialogues*** : qui représentent les moyens d'interaction avec le système,
  ✓ ***Contrôles*** : qui contiennent la logique applicative et
  ✓ ***Entités*** : qui sont les objets métier manipulés.

- Résultat de cette investigation : **diagramme de classes participantes**.

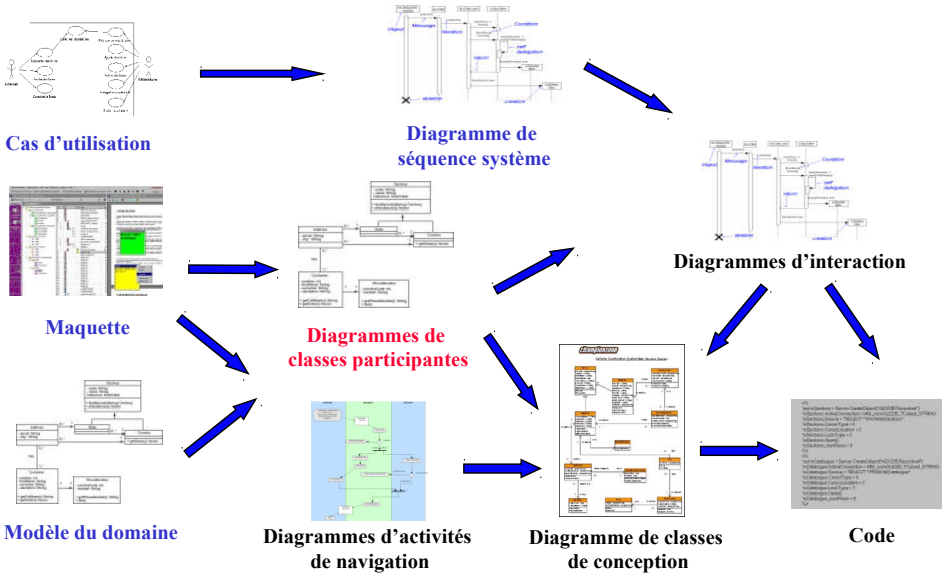
DÉMARCHE

Déjà réalisé :
- Identification des cas d'utilisation
- Description détaillée des cas d'utilisation
- Première version du modèle du domaine

Pour passer en douceur à la conception

Identification
- des principales **classes d'IHM**
- des classes décrivant la **cinématique de l'application**

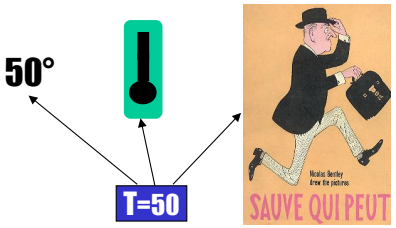
DÉMARCHE



SOMMAIRE

LE MODÈLE MVC

3 Sous-Systèmes



- Le **Modèle** contient la connaissance du domaine, les données et fonctionnalités fondamentales, indépendamment des entrées/sorties.

- La **Vue** est responsable de la présentation en sortie et du maintien de la cohérence de la présentation quand le modèle change.

- Le **Contrôleur** est responsable de la gestion des séquences d'interactions avec l'utilisateur.

TYPES DE CLASSES D'ANALYSE

❶ Les dialogues

- Classes qui permettent les **interactions entre le système et ses utilisateurs**.

- Écrans proposés à l'utilisateur :
  - ✓ les formulaires de saisie,
  - ✓ les résultats de recherche, etc.

- Elles proviennent directement de l'analyse de la maquette.

TYPES DE CLASSES D'ANALYSE

❶ Les dialogues

- Il y a au moins un dialogue pour chaque paire (*acteur – cas d'utilisation*).

- Ce dialogue peut avoir des objets subalternes auxquels il délègue une partie de ses responsabilités.

- En général, les dialogues vivent aussi longtemps que le cas d'utilisation concerné.

TYPES DE CLASSES D'ANALYSE

❷ Les contrôles

- Classes qui contiennent la **cinématique de l'application**.

- Elles font la **transition** entre les **dialogues** et les **classes métier**, en permettant aux écrans de manipuler des informations détenues par un ou plusieurs objets métier.

- Elles contiennent les **règles métier**, et permettent de les isoler à la fois des objets d'interface et des données persistantes.

TYPES DE CLASSES D'ANALYSE

❷ Les contrôles

- Les contrôles ne donneront pas forcément lieu à de vrais objets de conception,
mais assurent qu'on n'oublie pas de fonctionnalités ou de comportements requis par les cas d'utilisation.

TYPES DE CLASSES D'ANALYSE

❸ Les entités

- Classes qui représentent les **objets métier**.

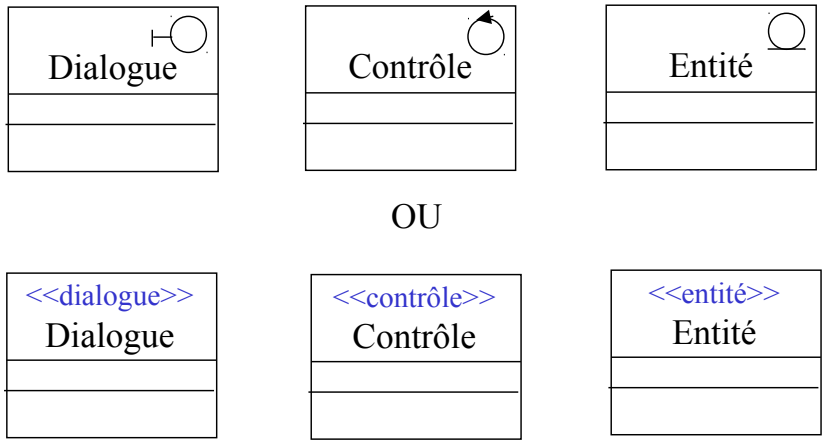
- Elles proviennent du modèle du domaine.

- Elles sont confirmées et complétées par cas d'utilisation**.**

- Très souvent des **entités persistantes** :
  - ✓ elles survivent à l'exécution d'un cas d'utilisation particulier
  - ✓ elles permettent à des données et des relations d'être stockées dans des fichiers ou des bases de données.

TYPES DE CLASSES D'ANALYSE

- Utilisation des représentations graphiques de I. Jacobson.



SOMMAIRE

- Introduction
- Typologie des classes d'analyse
- **Diagramme de classes participantes**
- Classes d'analyse du site Web

DIAGRAMME DE CLASSES PARTICIPANTES

- Chaînon manquant de la démarche : le diagramme de classes participantes.

- Diagrammes de classes UML qui décrivent, par cas d'utilisation, les principales classes d'analyse et leurs relations.

- Avantage important pour le chef de projet :
  découper le travail de son équipe d'analystes
  suivant les différents **cas d'utilisation**
  (*plutôt que de vouloir tout traiter d'un bloc*).

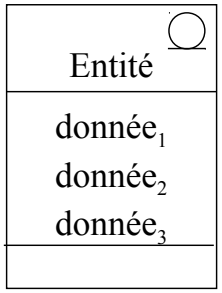
DIAGRAMME DE CLASSES PARTICIPANTES

- De plus, le modèle du domaine joue le rôle de **référence commune** en ce qui concerne les entités découvertes par les différentes équipes.

- Diagrammes de classes participantes particulièrement importants :
  ils font la jonction entre :
  - ✓ les cas d'utilisation,
  - ✓ le modèle du domaine,
  - ✓ la maquette et
  - ✓ les diagrammes de conception logicielle
    (*diagrammes d'interaction et diagrammes de classes*).

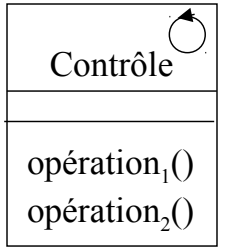
ENTITÉS

- Possèdent uniquement des attributs.
- Ces attributs représentent en général des informations persistantes de l'application.



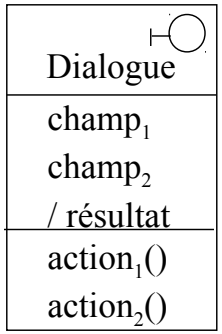
CONTRÔLES

- Possèdent uniquement des opérations.
- Ces opérations montrent
  - ✓ la logique de l'application,
  - ✓ les règles métier,
  - ✓ les comportements du système informatique.
- Il y a un ou plusieurs contrôles par cas d'utilisation, en fonction du nombre et de la cohérence des comportements associés.



DIALOGUES

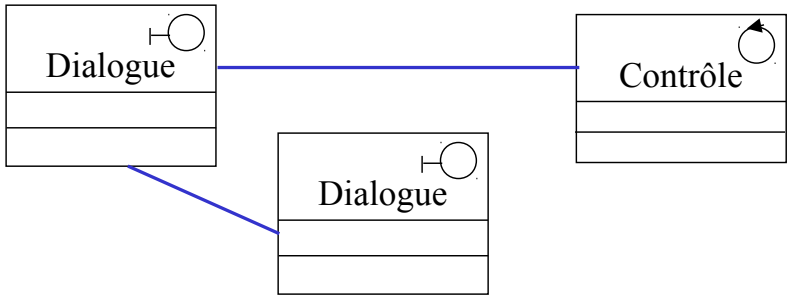
- Possèdent des attributs et des opérations.
- Les attributs représentent les champs de saisie ou des résultats.
- Les résultats seront distingués en utilisant la notation de l'attribut dérivé.
- Les opérations représentent des actions de l'utilisateur sur l'IHM.



ASSOCIATION ENTRE CLASSES D'ANALYSE

Respect des 4 règles suivantes :

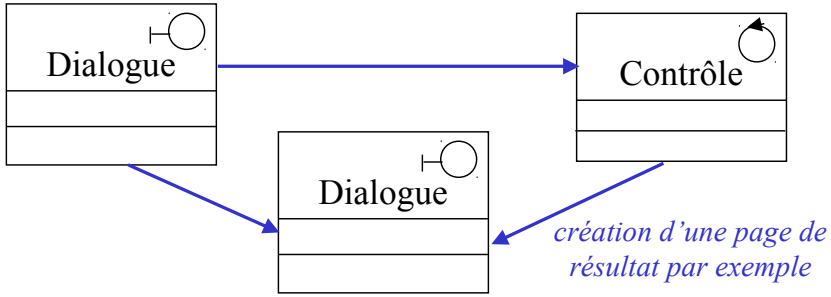
❶ Les ***dialogues*** ne peuvent être reliés
- ✓ qu'aux contrôles ou
- ✓ à d'autres dialogues,
- ✓ mais pas directement aux entités.



ASSOCIATION ENTRE CLASSES D'ANALYSE

Respect des 4 règles suivantes :

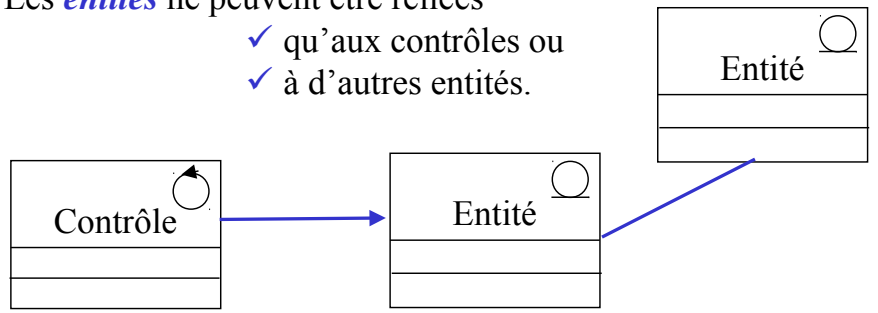
❶ En général les associations sont unidirectionnelles dans le sens dialogue vers contrôle, sauf quand ce dernier crée un nouveau dialogue (*par exemple une page de résultats*).



ASSOCIATION ENTRE CLASSES D'ANALYSE

Respect des 4 règles suivantes :

❷ Les ***entités*** ne peuvent être reliées
- ✓ qu'aux contrôles ou
- ✓ à d'autres entités.

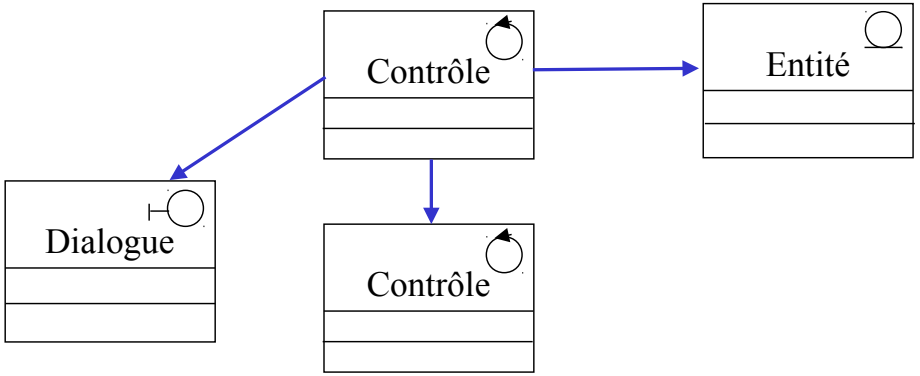


● Les associations sont toujours unidirectionnelles dans le sens contrôle vers entité.

ASSOCIATION ENTRE CLASSES D'ANALYSE

Respect des 4 règles suivantes :

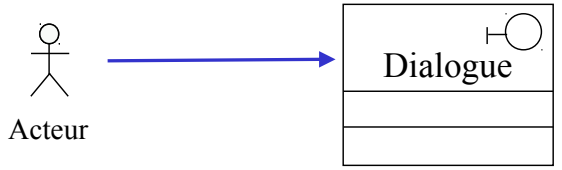
❸ Les ***contrôles*** ont accès à tous les types de classes, y compris d'autres contrôles.



ASSOCIATION ENTRE CLASSES D'ANALYSE

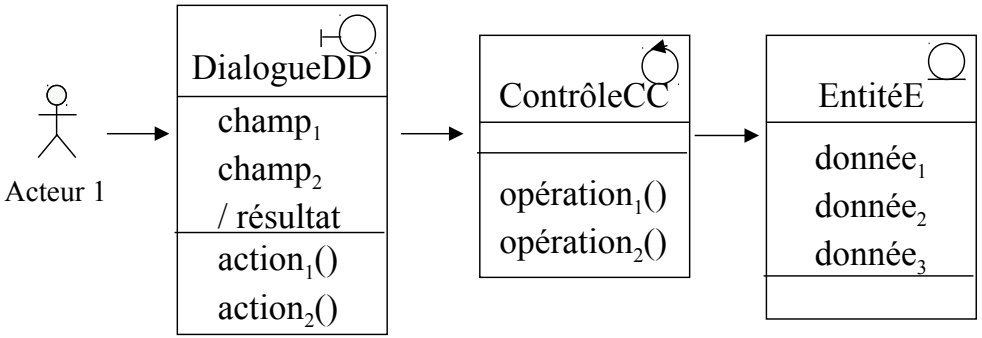
Respect des 4 règles suivantes :

❹ Les ***acteurs*** ne peuvent être liés qu'aux dialogues.



ASSOCIATION ENTRE CLASSES D'ANALYSE

Exemple de diagramme de classes participantes



SOMMAIRE

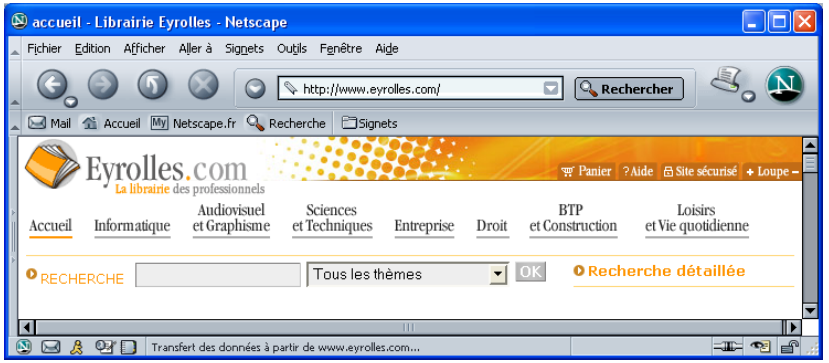
- Introduction
- Typologie des classes d'analyse
- Diagramme de classes participantes
- **Classes d'analyse du site Web**

RECHERCHER DES OUVRAGES

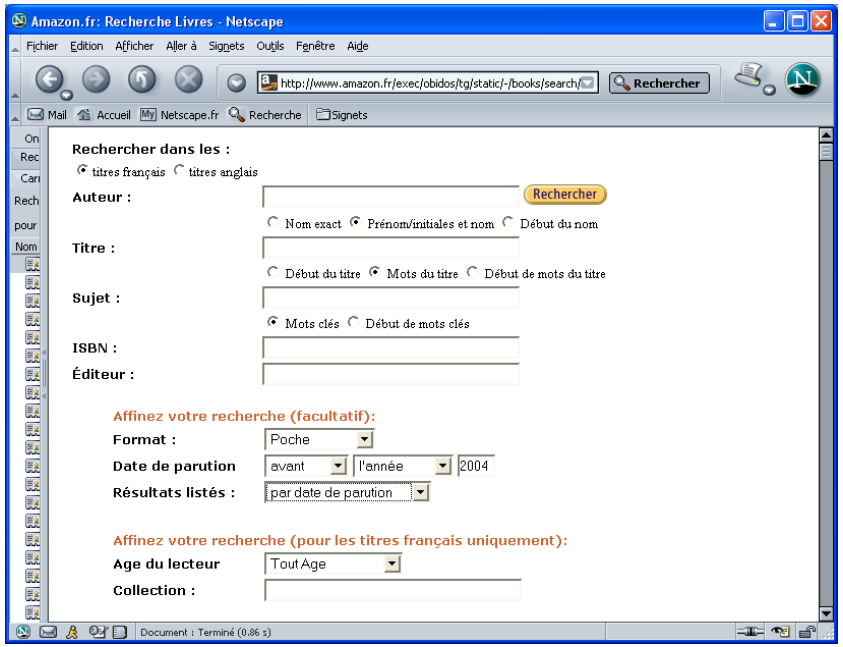
- L'internaute veut trouver le plus rapidement possible un ouvrage recherché dans l'ensemble du catalogue.
- Il veut aussi pouvoir consulter la fiche détaillée d'un ouvrage particulier avant de le mettre dans son panier.

- On suppose que la maquette nous a présenté *4* écrans principaux
  - ✓ Les écrans de recherche rapide et de recherche avancée.
  - ✓ Le résultat de la recherche, sur une ou plusieurs pages, qui permet en particulier d'accéder à la fiche détaillée d'un ouvrage particulier.

RECHERCHER DES OUVRAGES



Exemple d'écran de recherche rapide



Autre exemple d'écran de recherche détaillée

Exemple d'écran de résultat de recherche

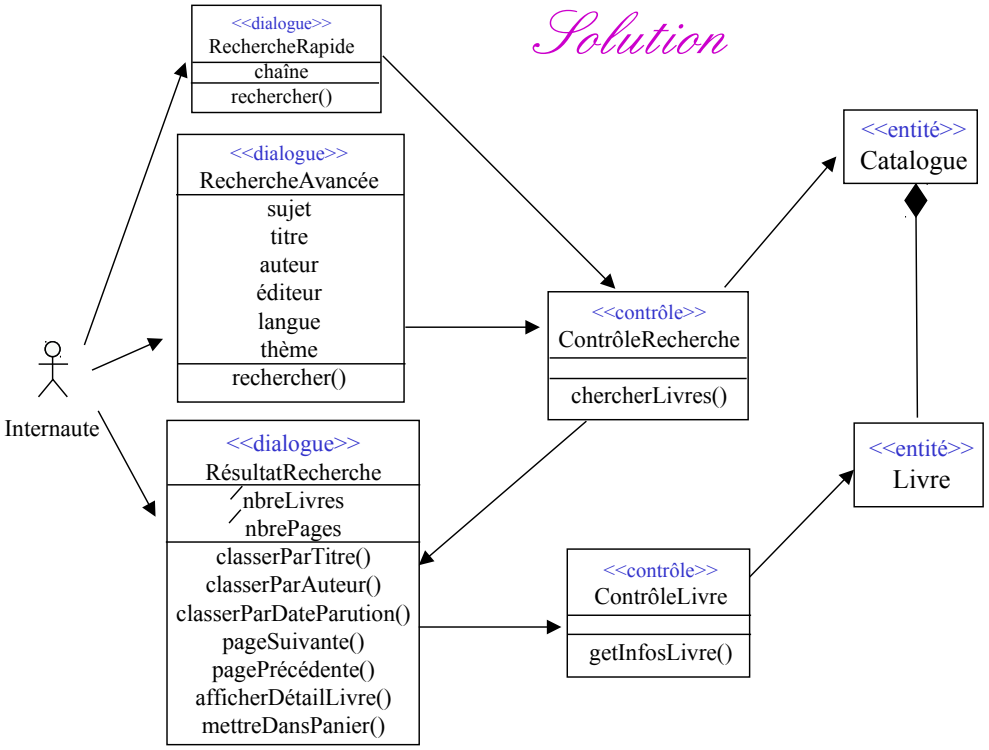
Exemple d'écran de fiche détaillée

CLASSES PARTICIPANTES SITE WEB

Faire le diagramme des classes participantes

Rechercher des ouvrages

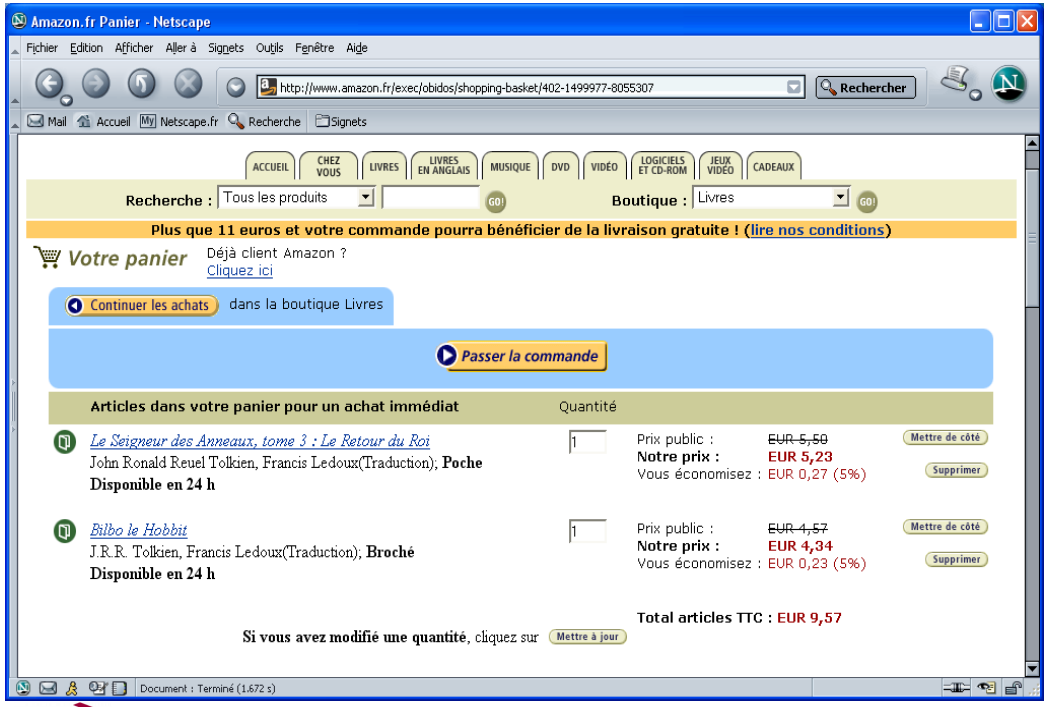
Solution

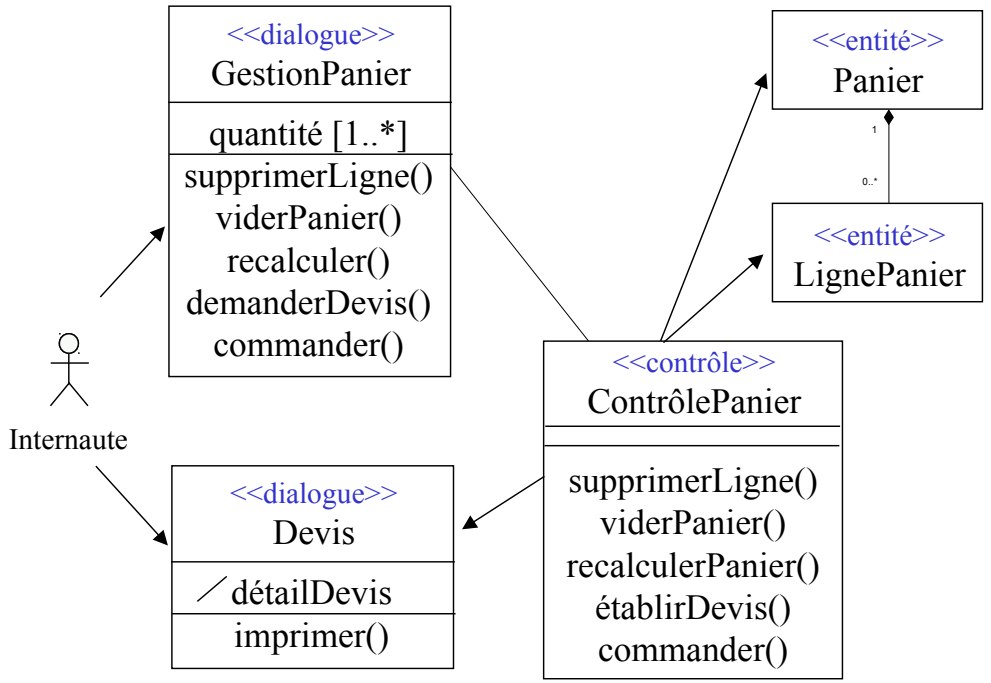


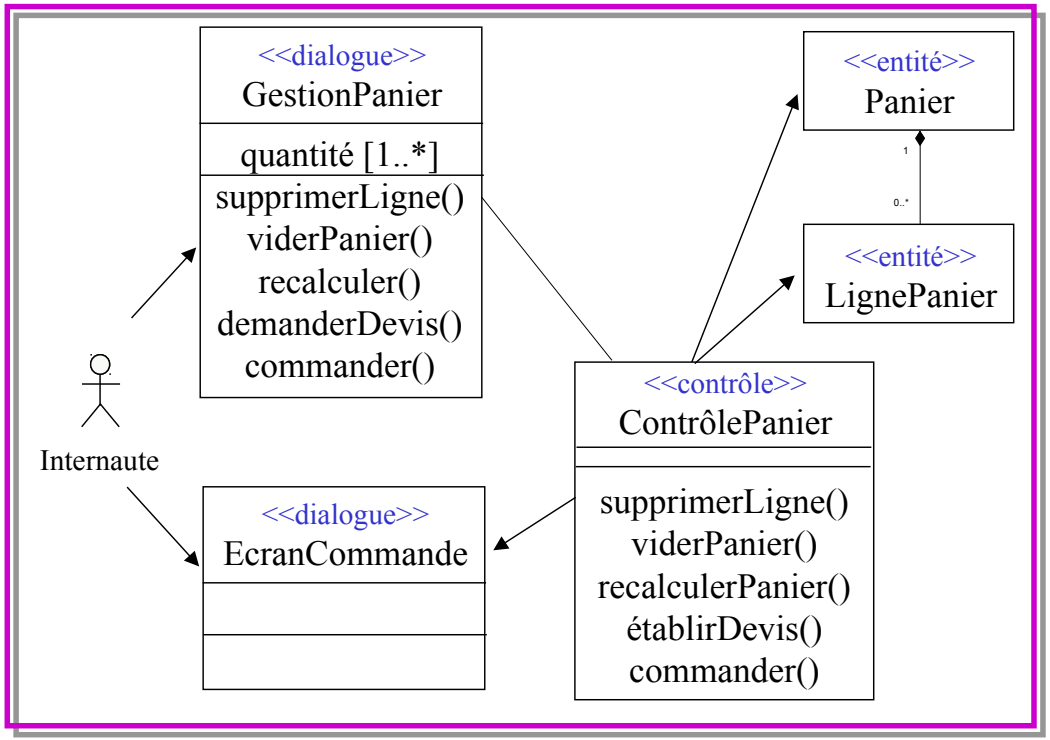
GÉRER SON PANIER

Faire le diagramme des classes participantes
Gérer son panier

- Lorsque l'internaute est intéressé par un ouvrage, il doit avoir la possibilité
  - ✓ de l'enregistrer dans un panier virtuel,
  - ✓ puis d'ajouter d'autres livres,
  - ✓ en supprimer ou encore
  - ✓ en modifier les quantités avant de passer commande.



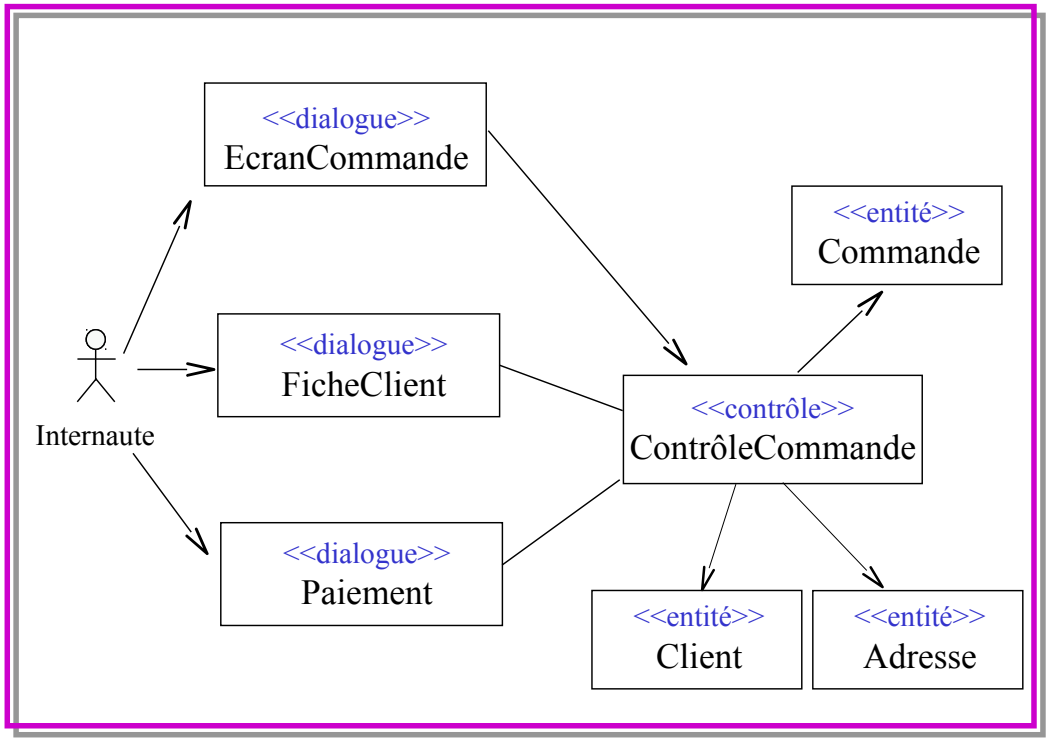


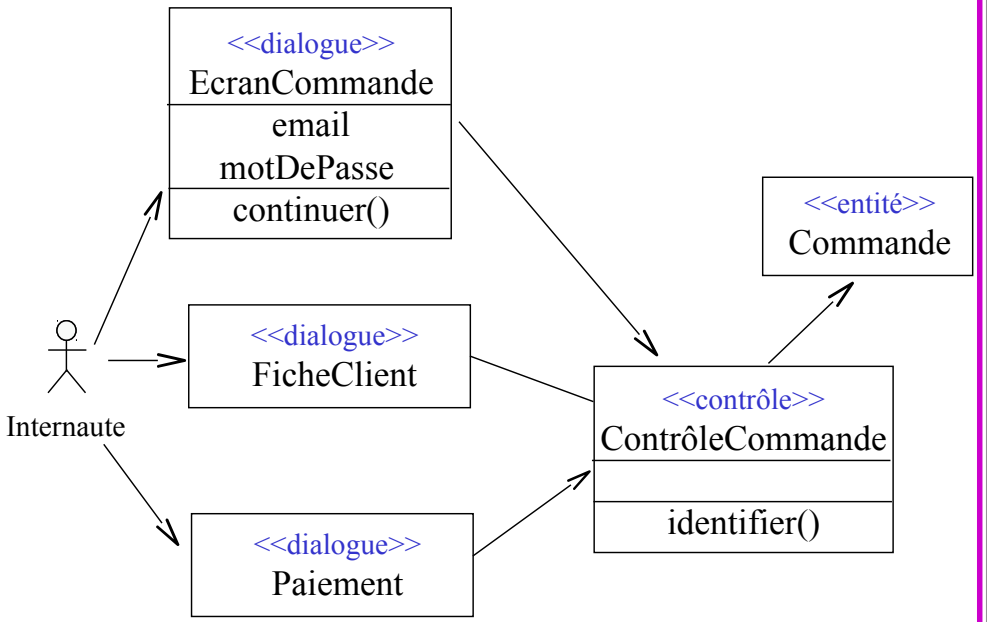


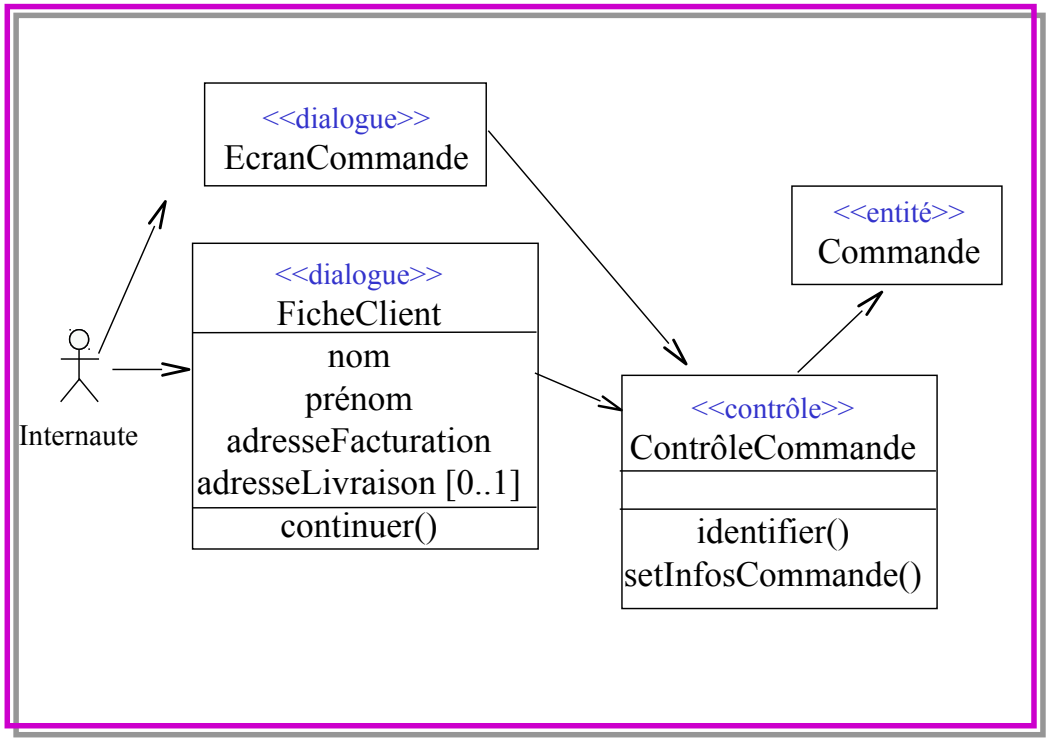
EFFECTUER UNE COMMANDE

- A tout moment, le client doit pouvoir accéder au formulaire du bon de commande, dans lequel il saisit ses coordonnées et les informations nécessaires au paiement et à la livraison.

- S'il est déjà client, il peut modifier les informations sauvegardées le concernant.






<<dialogue>>
EcranCommande
<<entité>>
Commande
<<dialogue>>
FicheClient
nom
prénom
adresseFacturation
adresseLivraison [0..1]
continuer()
Internaute
<<contrôle>>
ContrôleCommande
identifier()
setInfosCommande()

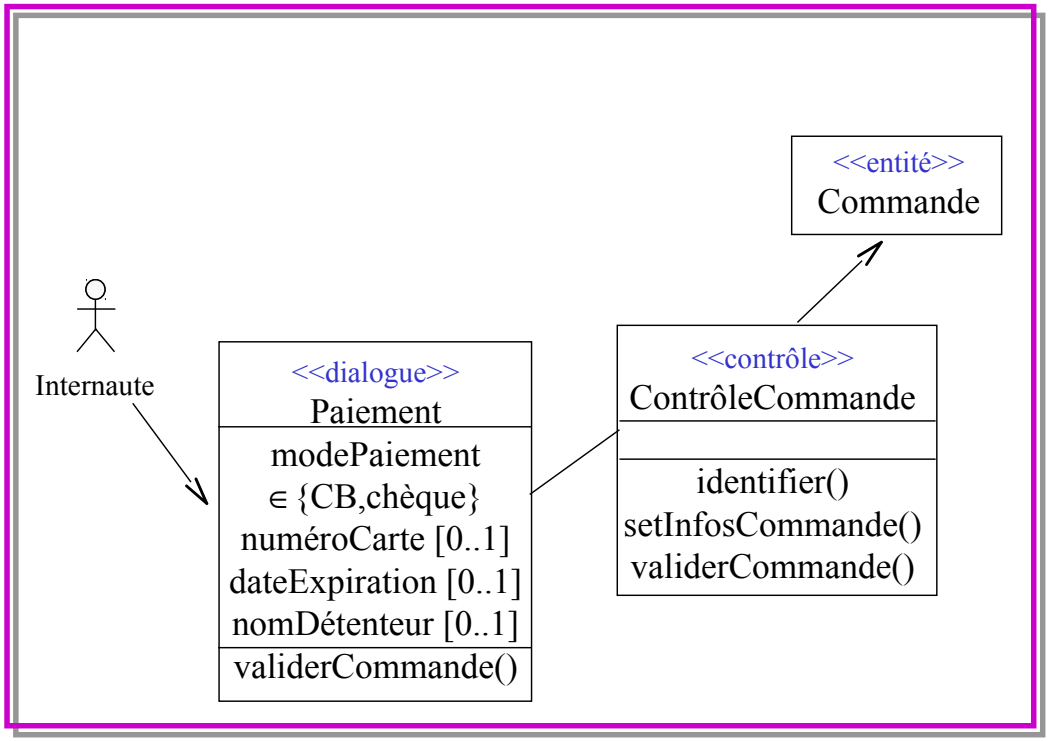

<<entité>>
Commande
Internaute
<<dialogue>>
Paiement
modePaiement
∈ {CB,chèque}
numéroCarte [0..1]
dateExpiration [0..1]
nomDétenteur [0..1]
validerCommande()
<<contrôle>>
ContrôleCommande
identifier()
setInfosCommande()
validerCommande()

Diagramme d'états-transitions

DIAGRAMME ÉTATS-TRANSITIONS

(Statecharts ou State machines)

Diagramme comportemental

Modélisation du cycle de vie des éléments fortement réactifs

SOMMAIRE

INTRODUCTION

- Les diagrammes d'états sont **utilisés** pour compléter aussi bien :
  - le diagramme de classes d'analyse que
  - le diagramme de classes technique.

- Ils sont élaborés **pour une classe** afin de visualiser le comportement d'un objet au cours du temps.

INTRODUCTION

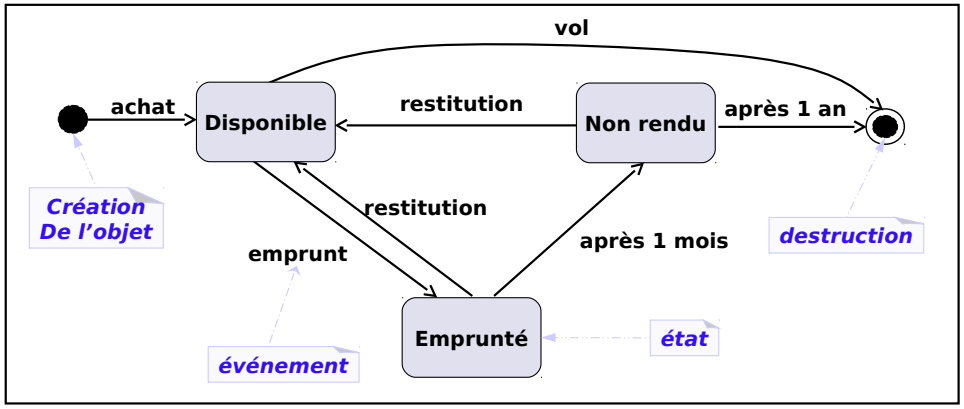
- UML a repris le concept de *machine à états finis* : qui consiste à s'intéresser au **cycle de vie d'une instance générique d'une classe** particulière au fil de ses interactions avec le reste du monde, dans tous les cas possibles.

- Cette vue locale d'un objet, qui **décrit comment il réagit à des événements** en fonction de son état courant et comment il passe dans un nouvel état, est représentée graphiquement sous la forme d'un *diagramme d'états*.

INTRODUCTION

Exemple de diagramme d'états

Bibliothèque – objet exemplaire



INTRODUCTION

- Le diagramme d'états représente le cycle de vie des objets d'une classe.

- Les objets subissant les contraintes extérieures passent par une succession d'états **modifiant la valeur des attributs** ou des **règles composant l'objet**.

ÉTATS D'UN OBJET

Ils sont fonction de :

- **La valeur des propriétés de l'objet**
  ex : un élève est dit *"recalé "* si sa moyenne est inférieure à *10/20*.

- **La valeur des propriétés des objets qui lui sont reliés**
  ex : un client est dit *"suspendu "* si le solde d'un de ses comptes est négatif depuis plus de *5* jours.

- **La valeur des états des objets qui lui sont reliés**
  ex : un client est dit *"inactif "* s'il n'existe pas pour ce client d'occurrences de relation avec un contrat à l'état *"en cours "*.

ÉTATS D'UN OBJET

Ils sont fonction de :

- **Ses relations ou de son absence de relation avec d'autres objets**
  ex : une commande est dite *"en attente "* s'il n'existe pas pour cette commande d'occurrences de relation avec une livraison.

- **La valeur des propriétés de ses relations avec d'autres objets**
  ex : un produit est dit *"en rupture de stock "* dans un dépôt si la quantité en stock < seuil critique.

INTRODUCTION

- Les diagrammes d'états constituent une **modélisation du comportement d'un objet**.

- Un diagramme d'états relie des événements à des états en spécifiant la **séquence d'états** provoquée par une **séquence d'événements** (*ou déclencheurs ou triggers*).

- Une transition représente une modification d'état provoquée par un événement.

POUR QUELLES CLASSES ?

Toutes les classes du modèle statique ne requièrent pas nécessairement une machine à états, représentée par un diagramme d'états.

Un diagramme d'états n'a d'utilité que si l'**objet passe par au moins trois états**.

POUR QUELLES CLASSES ?

- Classes qui ont un **comportement dynamique complexe** nécessitant une description poussée.

Cela correspond à l'un des deux cas suivants :

1- **Les objets de la classe peuvent-ils réagir différemment à l'occurrence du même événement** ?

Chaque type de réaction caractérise un état particulier.

2- **La classe doit-elle organiser certaines opérations dans un ordre précis** ?

états séquentiels précisant la chronologie forcée des événements d'activation.

COMMENT LE CONSTRUIRE ?

1- Représenter tout d'abord la **séquence d'états** qui décrit le **comportement nominal d'un objet**, avec les transitions qui y sont associées.

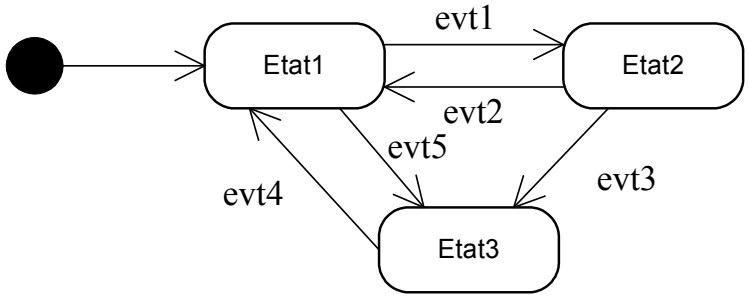
2- Ajouter progressivement les transitions qui correspondent aux **comportements alternatifs ou d'erreur**.

3- Compléter les **actions** sur les transitions et les **activités** dans les états.

4- **Structurer le diagramme** en sous-états s'il devient trop complexe.

COMMENT LE REPRÉSENTER ?

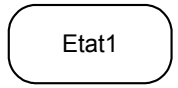
- Un diagramme d'états est un **graphe** dont les **nœuds** sont des **états** et les **arcs** orientés des **transitions** désignées par les noms d'événements.



- Le passage d'un état à un autre est **instantané** car le système doit toujours être dans un état connu.

ÉTAT

- Un *état* représente une situation durant la vie d'un objet pendant laquelle :
  - ✓ il satisfait une certaine **condition**,
  - ✓ il exécute une certaine **activité**,
  - ✓ ou bien il attend un certain **événement**.



- Un objet passe par une succession d'états durant son existence. Un état a une **durée finie**, **variable** selon la vie de l'**objet**, en particulier en fonction des événements qui lui arrivent.

COMMENT IDENTIFIER LES ÉTATS ?

Comment trouver les états d'une classe ?

Pour faire un parallèle, on peut dire qu'il est **aussi difficile** de trouver les bons états dans le modèle dynamique que les bonnes **classes** dans le modèle statique !

COMMENT IDENTIFIER LES ÉTATS ?

Il n'y a donc pas de recette miracle, cependant **trois démarches complémentaires** peuvent être mises en œuvre :

1- La **recherche intuitive** repose sur l'expertise métier. Certains états fondamentaux font partie du vocabulaire des experts du domaine et sont identifiables a priori.

2- L'étude des attributs et des associations de la classe peut donner des indications précieuses : chercher des **valeurs seuils d'attributs** qui modifient la dynamique, ou des **comportements** qui sont induits par **l'existence** ou **l'absence** de **certains liens**.

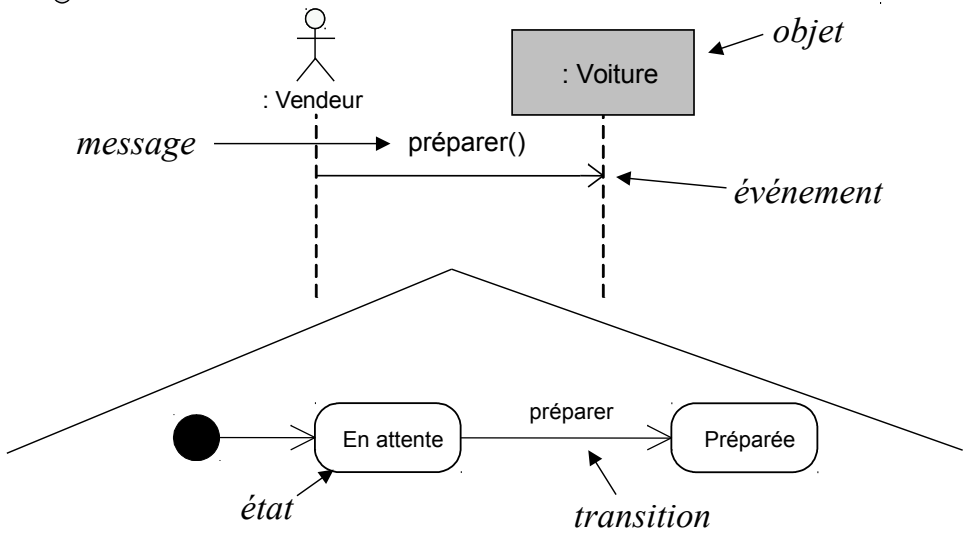
COMMENT IDENTIFIER LES ÉTATS ?

3- Une **démarche systématique** peut également être utilisée, classe par classe :

- ✓ chercher le diagramme d'interaction le plus représentatif du comportement des instances de cette classe,
- ✓ associer un état à chaque intervalle entre événements émis ou reçus par une instance et
- ✓ placer les transitions.
- ✓ reproduire ensuite cette démarche avec tous les scénarios faisant intervenir des instances de la classe, afin d'ajouter de nouvelles transitions ou de nouveaux états.

La difficulté principale consiste à trouver ensuite les boucles dans le diagramme, afin de ne pas multiplier les états.

COMMENT REPRÉSENTER LES CONCEPTS DYNAMIQUES DE BASE ?



ÉTAT INITIAL ET ÉTAT FINAL

- ***2* pseudo-états** :

✓ l'**état initial** du diagramme d'états : **création** de l'instance.

✓ l'**état final** du diagramme d'états : **destruction** de l'instance.

ÉVÉNEMENT / TRANSITION

- Un **événement** spécifie qu'il s'est passé quelque chose de significatif, localisé dans le temps et dans l'espace.

- Un **événement** représente l'occurrence d'un stimulus qui peut déclencher une transition entre états.

- Une **transition** décrit la réaction d'un objet lorsqu'un événement se produit (*généralement l'objet change d'état*).

ÉVÉNEMENT

- *6* sortes d'événements (*ou déclencheurs ou triggers*) :

1- La **réception d'un message** (*signal event*) envoyé par un autre objet, ou par un acteur. L'envoi d'un message est en général asynchrone (*l'émetteur n'est pas bloqué et peut continuer son exécution*).

2- L'**appel d'une opération** (*call event*) sur l'objet récepteur. L'événement d'appel est en général synchrone (*l'émetteur est bloqué et attend que l'appelé ait fini de traiter le message)*.
Ex : calculerMontant() (*message synchrone*)
imprimer() (*message asynchrone*)

ÉVÉNEMENT

3- Le **passage du temps** (*time event*), qui se modélise en utilisant le mot-clé *after* suivi d'une expression représentant une durée, décomptée à partir de l'entrée dans l'état courant.
Ex : after(durée), after(3 minutes)

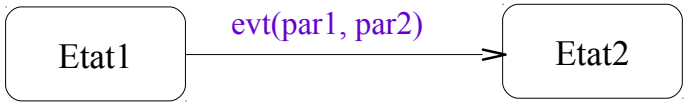
4- Un **changement** dans la satisfaction d'une condition (*change event*). On utilise alors le mot-clé *when*, suivi d'une expression booléenne. L'événement de changement se produit lorsque la condition passe à vrai.
Ex : when (solde <= 1 000 €)
Différence avec une condition de garde : un événement de changement est évalué continuellement jusqu'à ce qu'il devienne vrai, et c'est à ce moment-là que la transition se déclenche.

ÉVÉNEMENT

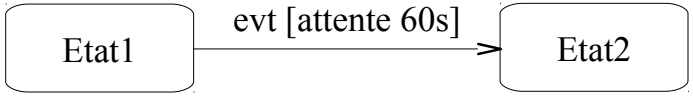
5- La **fin d'une activité** (*completion event*) de type *do/*, interne à un état. Cela peut déclencher une transition dite automatique qui ne porte pas de déclencheur explicite.

6- **at (date)**
Ex : at (date = 2 / 10 / 2016)

Un **événement** peut porter des **paramètres** qui matérialisent le flot d'informations ou de données entre objets.



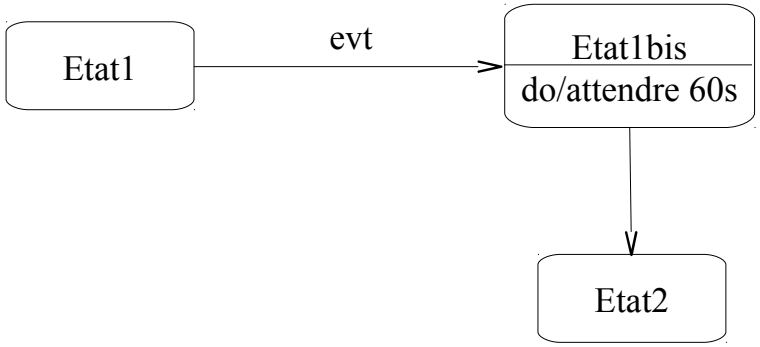
EXERCICE



Que pensez-vous de ce modèle ?

SOLUTION

Transition doit être instantanée



MESSAGE

- Un **message** est une transmission d'information unidirectionnelle entre deux objets, l'objet émetteur et l'objet récepteur.

- Le **mode de communication** peut être :
  - **synchrone** ou
  - **asynchrone**.

- La réception d'un message est un événement qui doit être traité par le récepteur.

CONDITION

Une **condition** est une expression booléenne qui doit être vraie lorsque l'événement arrive pour que la transition soit déclenchée.

OU

Les gardes

- Une **condition de garde** est une expression booléenne.

- Une **condition de garde** doit être vraie pour qu'une transition gardée ait lieu quand survient un événement.

- Une **condition gardée** est symbolisée par une expression booléenne entre crochets :
  événement(attributs)[condition]

Exemples
- infraction [nombre points < 20]
- désistement [assez de places]

OPÉRATION

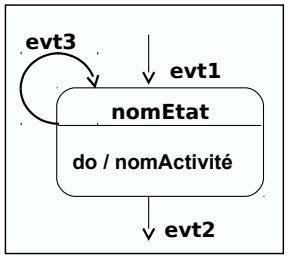
- La description comportementale d'un objet doit spécifier ce que fait l'objet en réponse aux événements.

- Des opérations attachées aux états ou aux événements sont exécutées en réponse aux états ou aux événements correspondants.

- *2* types d'**opérations** :
  - *les **activités***
  - *les **actions***

ACTIVITÉ

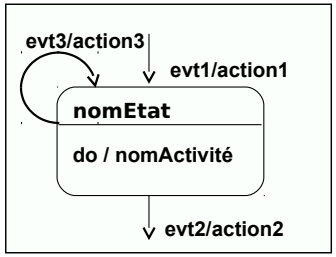
- Une **activité** est une opération qui nécessite du temps pour s'exécuter.
- Une **activité** est associée à un état.
- Une **activité** est exécutée dans un état tant que l'on se trouve dans cet état, ou jusqu'à ce que le calcul associé soit terminé.



- Une **activité** est placée à l'intérieur d'une boite d'état après un *do* : *"do / nomActivité "*

ACTION

- Une **action** est une opération instantanée.
- Une **action** est associée à un événement.



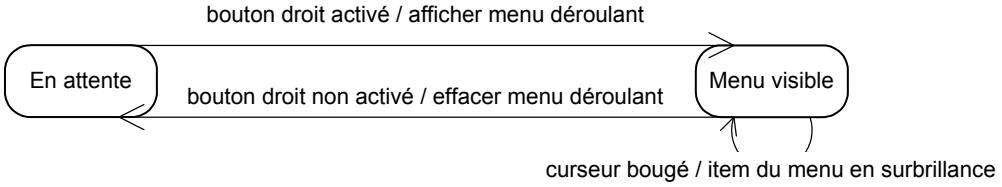
- Une **action** est généralement exécutée pendant une transition.
- Une **action** est placée après l'événement séparée par une barre oblique : *"nomEvénement / nomAction"*.

ACTION / ACTIVITÉ

- Les **actions** sont associées aux transitions et sont considérées comme **atomiques**, c'est-à-dire ininterruptibles, ou encore instantanées par rapport à l'échelle de temps considérée.
- Elles peuvent représenter des mises à jour de valeurs d'attributs, appels d'opérations, la création ou la destruction d'un autre objet, ainsi que l'envoi d'un signal à un autre objet.
- Les **activités**, au contraire des actions, ont une certaine durée, sont **interruptibles** et sont associées aux états.
- Les notions de durée et d'atomicité étant contextuelles et relatives, le travail du modélisateur consiste à choisir le bon concept au moment opportun.

EXEMPLE

Diagramme d'états d'un menu déroulant sur une station de travail

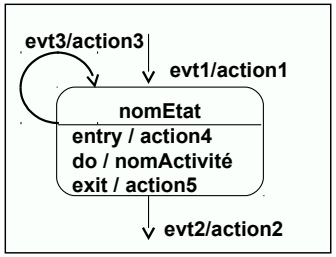


ACTION D'ENTRÉE / SORTIE

- Une **action en entrée** (*sortie*) est exécutée chaque fois que l'on rentre (*quitte*) dans l'état.

- Elles sont utiles lorsque toutes les transitions qui entrent (*sortent*) dans un état utilisent une **action commune**.

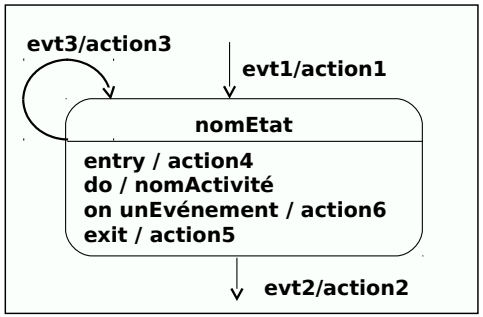
- L'action est indiquée après les mots clés "*entry* " ou "*exit* " suivis du caractère "/ ".



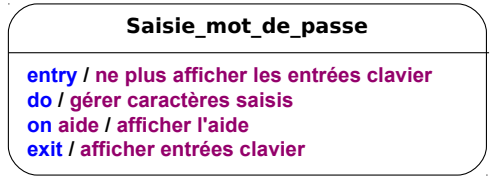
ACTION

- Un état peut contenir une autre **action** exécutée lors de l'occurrence d'un événement pendant que l'**objet est dans l'état** : c'est un **événement interne**.

- L'action est indiquée après le mot clé "*on unEvénement /* ".



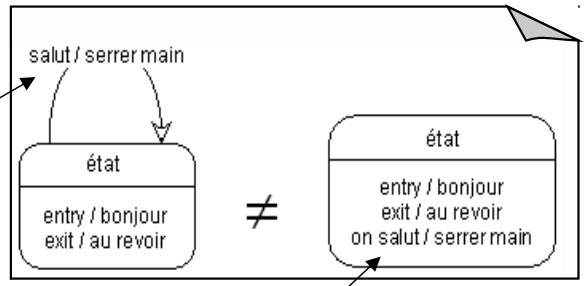
Exemple



ÉVÉNEMENT INTERNE

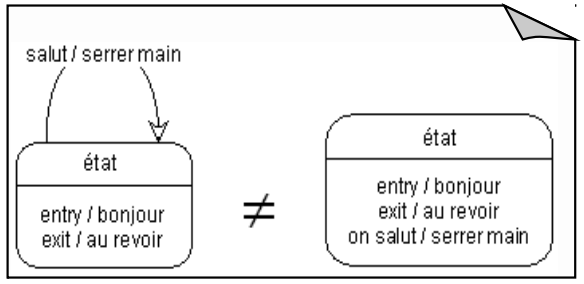
- Un **événement interne** n'entraîne pas l'exécution des actions d'entrée et de sortie, contrairement au déclenchement d'une transition réflexive.

transition réflexive ou transition propre



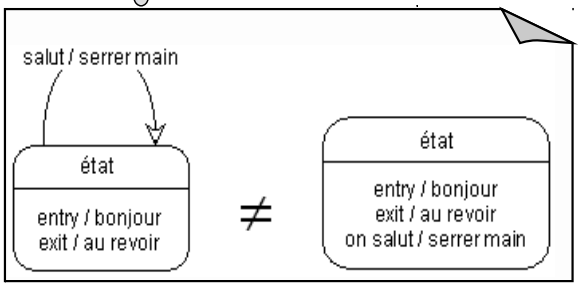
événement interne ou transition interne

EXERCICE



On rentre dans l'état "*état* " et l'événement "*salut* " arrive.
Que se passe-t-il pour les deux modèles ?

SOLUTION



- bonjour
- evt salut → au revoir
  serrer main
  bonjour
- evt salut → au revoir
  serrer main
  bonjour

- bonjour
- evt salut → serrer main
  …
- evt salut → serrer main

EXÉCUTION OPÉRATION

- Si plusieurs opérations sont spécifiées dans un état, elles sont **exécutées dans l'ordre suivant** :

✓ action sur la transition d'entrée — évt1 survient

✓ action d'entrée dans l'état
✓ activité "*do*" dans l'état — entrée dans l'état
✓ action associée aux événements internes

✓ action de sortie de l'état
✓ action sur la transition de sortie de l'état — évt2 survient

A savoir

DIAGRAMME D'ÉTATS

- Un diagramme d'états peut **s'exécuter une seule fois** ou en **boucles continues**.

- Les diagrammes qui **s'exécutent une seule fois** représentent les objets qui ont une **vie finie**. Ils possèdent un **état initial** et un **état final**.

LECTURE DES DIAGRAMMES D'ÉTATS

- Si un **objet** est **dans un état** et qu'un **événement** indiqué sur l'une des transitions **se produit**, alors l'**objet** entre dans un **nouvel état**.

- Si **plus d'une transition** sortent d'un état, alors le premier événement survenant provoque le franchissement de la transition correspondante.

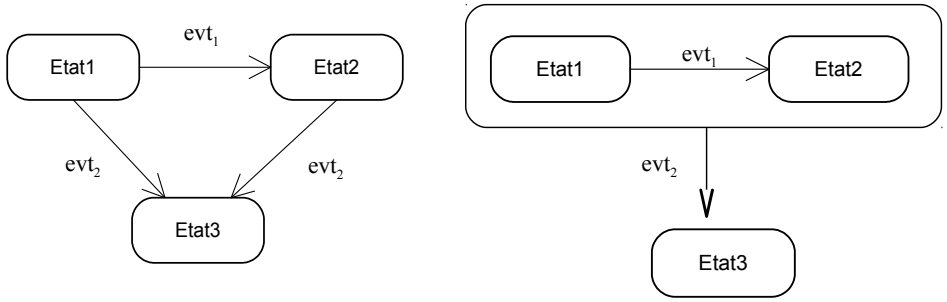
- Si un **événement** se produit et n'a **pas de transition** à partir de l'état courant, alors l'**événement** est **ignoré**.

GÉNÉRALISATION D'ÉTATS

- Dans le cas d'un **comportement dynamique complexe**, les diagrammes d'états sur un niveau deviennent vite **illisibles**.
- Pour éviter ce problème, il est nécessaire de **structurer les diagrammes d'états**.
- Un **diagramme d'états imbriqué** est en fait une **généralisation d'états**.
- La **généralisation** permet d'organiser les états et les événements de façon hiérarchique en utilisant l'**héritage**.
- Les états dans le diagramme imbriqué sont tous des **raffinements** du diagramme de haut niveau.
- Un **super-état** est la généralisation de plusieurs sous-états.

GÉNÉRALISATION D'ÉTATS

Exemple

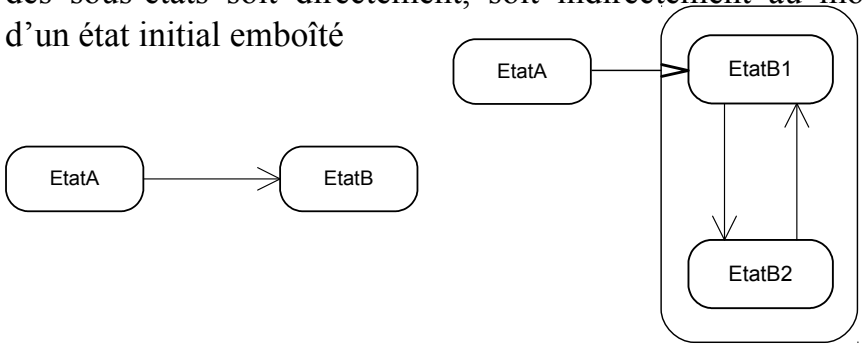


GÉNÉRALISATION D'ÉTATS

Exemple

- L'état *B* est divisé en *2* sous-états B_1 et B_2.

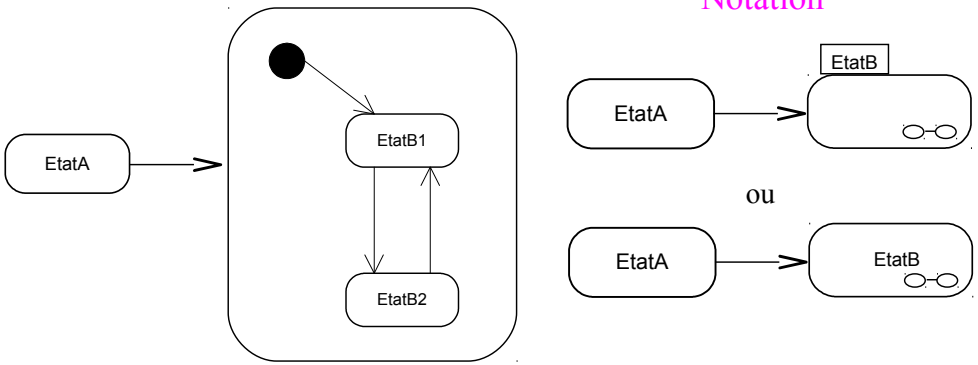
- La transition d'entrée dans l'état *B* doit être reportée sur un des sous-états soit directement, soit indirectement au moyen d'un état initial emboîté



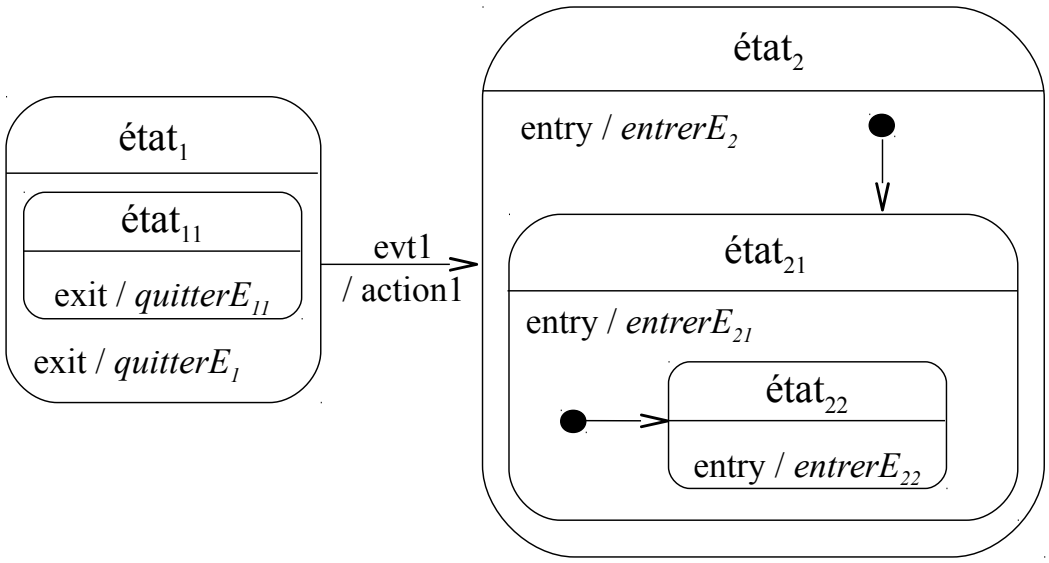
GÉNÉRALISATION D'ÉTATS

Exemple

- Il est préférable de limiter les liens entre niveaux hiérarchiques en définissant systématiquement un état initial pour chaque niveau.



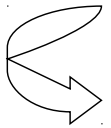
EXERCICE



état$_{11}$, réception de l'événement *evt*$_1$

SOLUTION

- quitterE$_{11}$
- quitterE$_1$
- action$_1$
- entrerE$_2$
- entrerE$_{21}$
- entrerE$_{22}$



état$_{22}$

AGRÉGATION D'ÉTATS

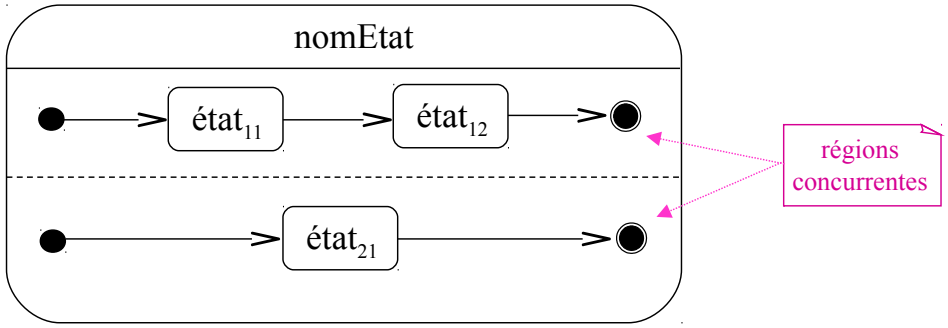
- L'**agrégation d'états** est la composition d'un état à partir de plusieurs autres états indépendants.

- L'objet doit être simultanément dans tous les états composant l'agrégation d'états.

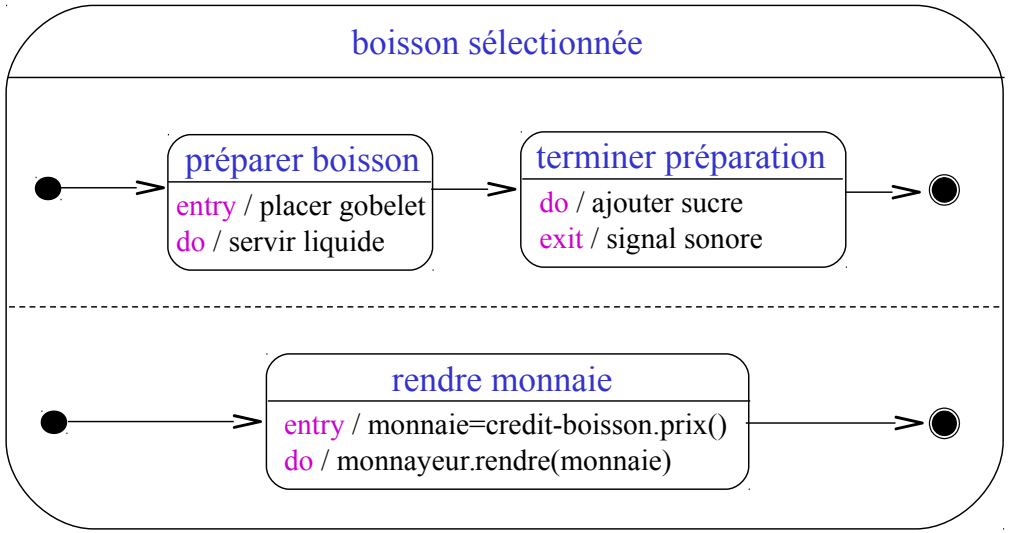
- L'agrégation d'états, tout comme la généralisation d'états, permet de **simplifier la représentation** des automates.
  - ✓ la **généralisation** simplifie par **factorisation**,
  - ✓ l'**agrégation** simplifie par **segmentation** de l'espace des états.

AGRÉGATION D'ÉTATS

- L'**agrégation d'états** permet de décrire efficacement les mécanismes concurrents. Cela permet de représenter des zones où l'action est réalisée par des **flots d'exécution parallèles**.

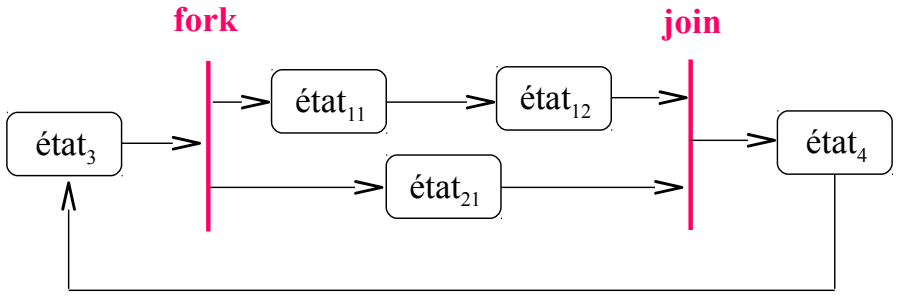


EXEMPLE



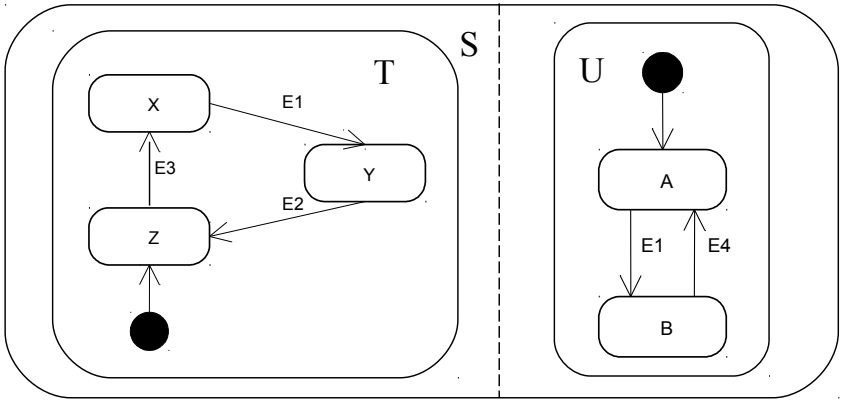
AGRÉGATION D'ÉTATS

- Il est également possible de représenter ce type de comportement au moyen de transitions concurrentes dites complexes.



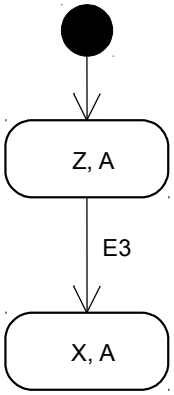
EXERCICE

- L'état *S* est un agrégat formé de *2* états indépendants *T* et *U*.
- *T* est composé des sous-états *X*, *Y* et *Z*.
- *U* est composé des sous-états *A* et *B*.



EXERCICE

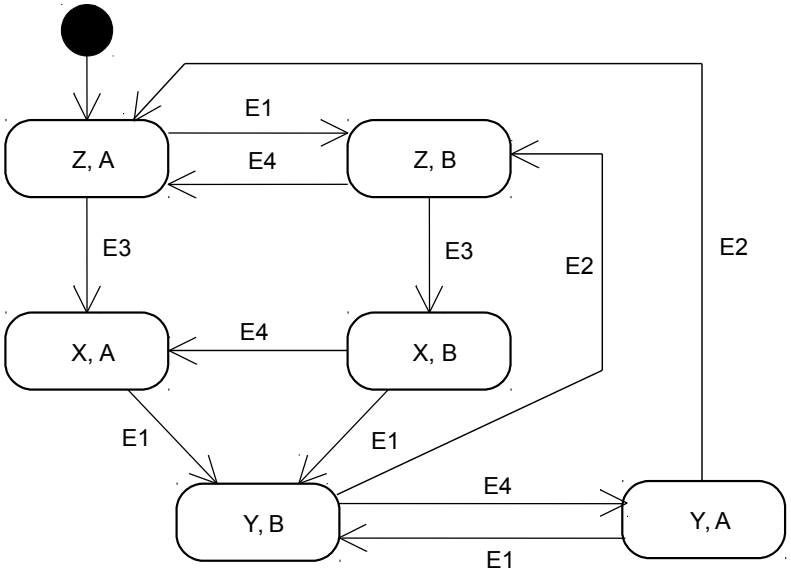
Sans agrégation d'états,
chercher l'automate équivalent



A finir

6 états et *11* transitions

SOLUTION



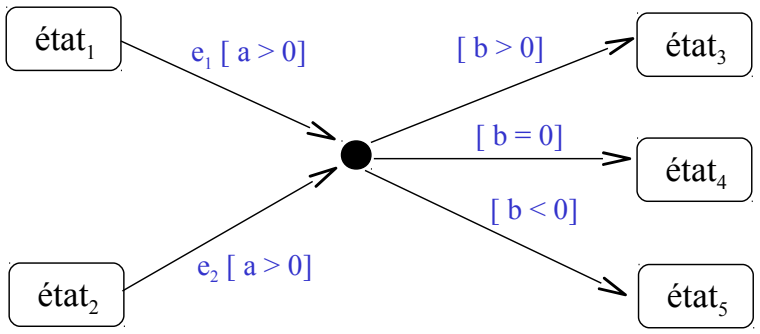
POINTS DE DÉCISION

- Représentation des **alternatives pour le franchissement d'une transition**.
- Pseudo-états particuliers :
  - ✓ le point de jonction ●
  - ✓ le point de choix ◇

POINT DE JONCTION

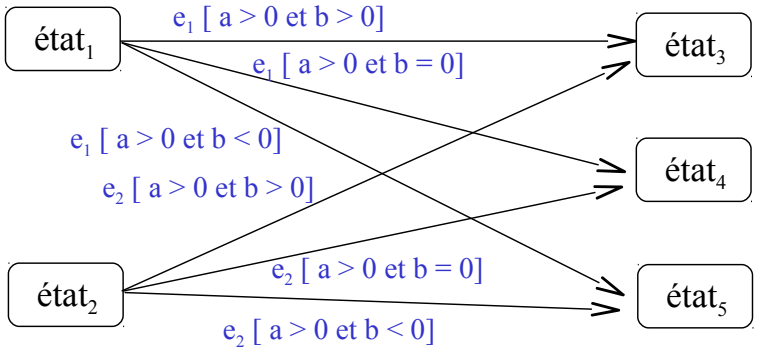
- Les **points de jonction** sont un artefact graphique qui permet de **partager des segments de transitions**.
- L'intérêt est de permettre une **notation plus compacte** et de rendre **plus visibles les chemins alternatifs**.
- Un **point de jonction** peut avoir **plusieurs segments** de transition **entrante** et **plusieurs segments** de transition **sortante**.
- Lorsqu'un chemin passant par un point de jonction est emprunté, toutes les gardes le long de ce chemin doivent s'évaluer à vrai dès le franchissement du premier segment.

EXERCICE



Sans point de jonction,
chercher l'automate équivalent

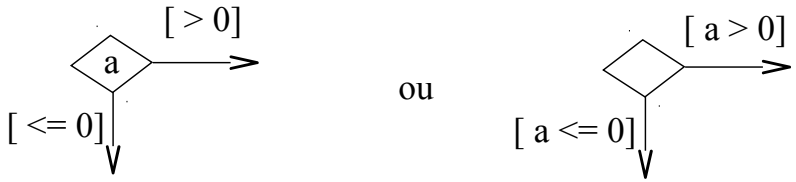
SOLUTION



POINT DE CHOIX

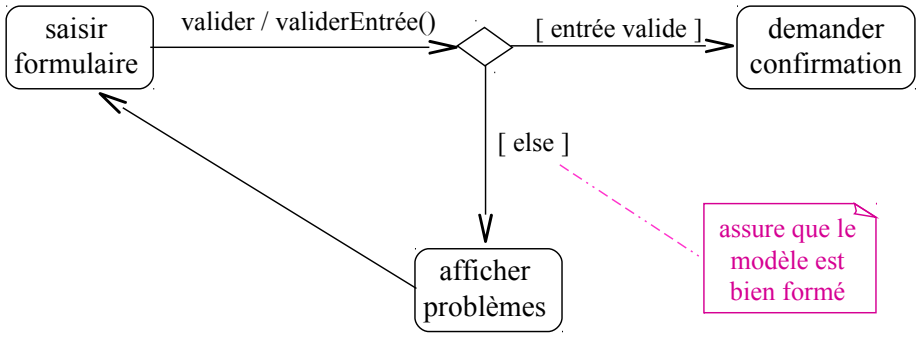
- Un **point de choix ou décision** possède **une entrée** et **au moins deux sorties**.
- Contrairement aux points de jonction, les **points de choix** dits "*dynamiques*", sont **évalués au moment où ils sont atteints**.
- Si quand le point de choix est atteint, aucun segment en aval n'est franchissable, le modèle est dit mal formé.
- Au contraire, si plusieurs segments sont franchissables, l'une d'entre elles est choisie aléatoirement.

POINT DE CHOIX

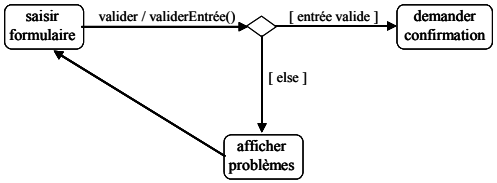


POINT DE CHOIX

Exemple



POINT DE CHOIX



- L'utilisateur remplit un formulaire. Il valide son formulaire en appuyant sur le bouton *valider*.
- *validerEntrée()* effectue une vérification de la cohérence des données.
- Si les informations paraissent correctes, on lui demande de confirmer, sinon on affiche les erreurs détectées et il doit remplir de nouveau le formulaire.

HISTORIQUE

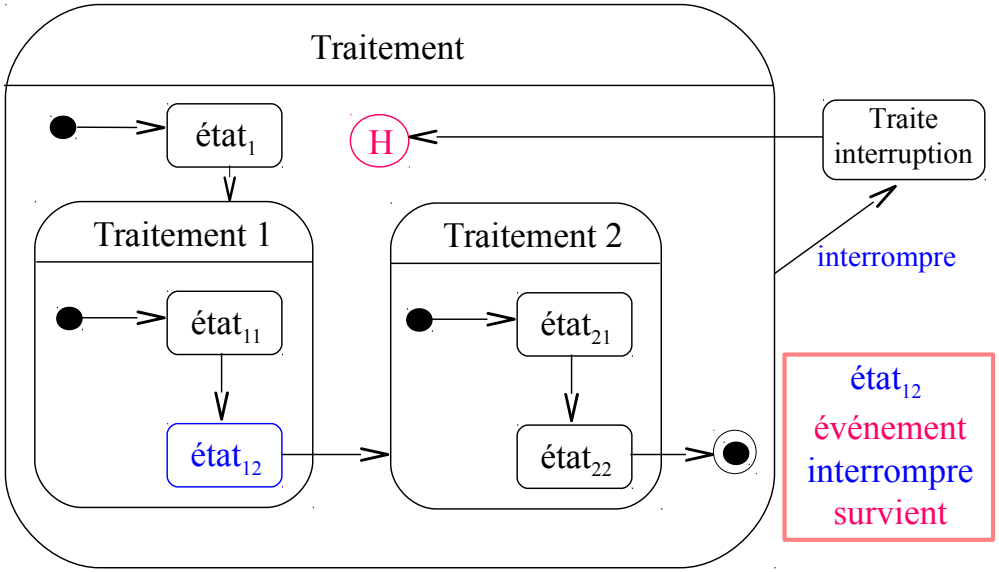
- Une transition ayant pour cible le **pseudo-état historique**, noté par un cercle contenant un H, est équivalente à une transition qui a pour **cible le dernier état visité dans la région** contenant le H.

Ⓗ *historique de surface*

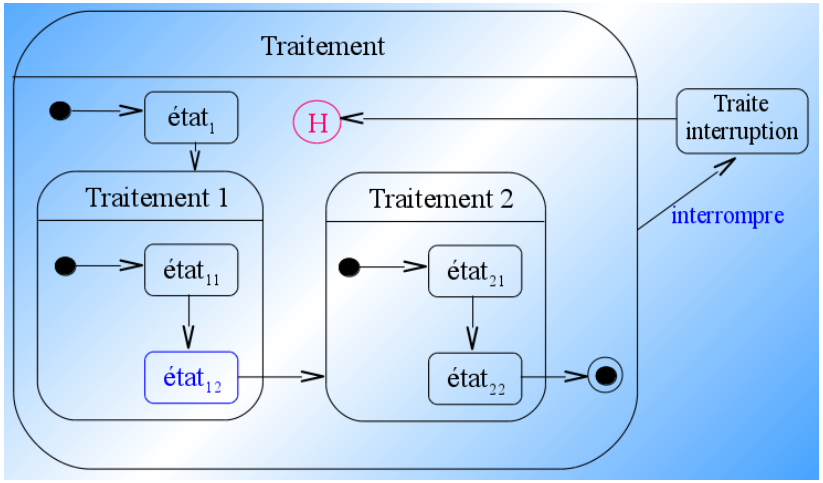
- Une transition ayant pour cible le **pseudo-état historique profond**, noté par un cercle contenant un H*, permet d'atteindre le **dernier état visité dans la région**, **quel que soit son niveau d'imbrication**, alors que le pseudo-état H limite l'accès aux états de son imbrication dans la région.

Ⓗ* *historique profond*

EXERCICE

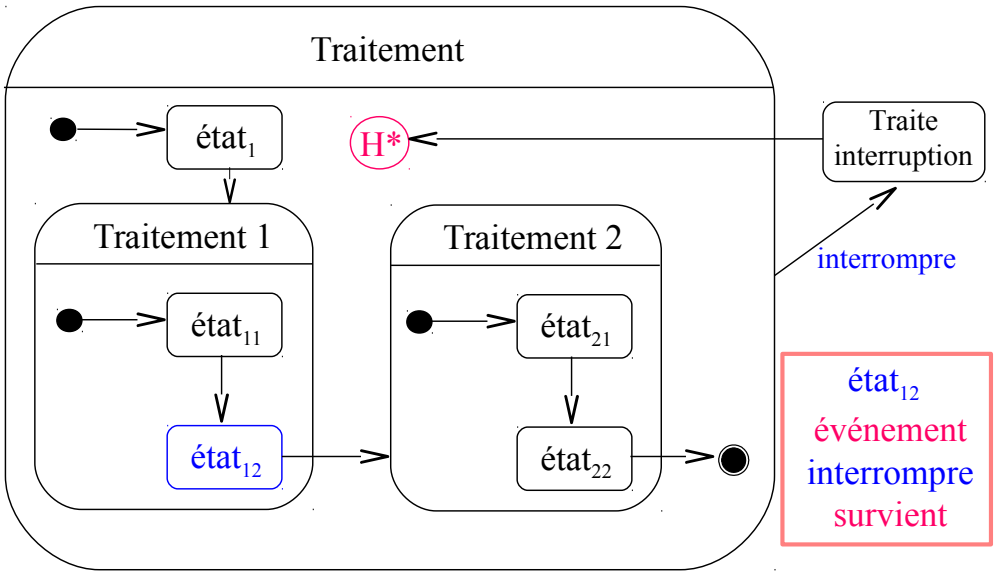


SOLUTION

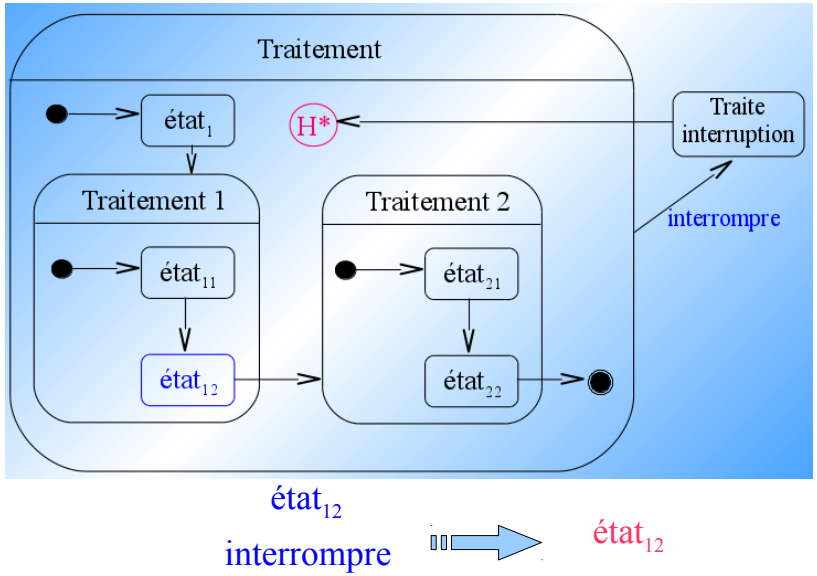


état12
interrompre
état11

EXERCICE



SOLUTION



ÉTUDE DE CAS

Système simplifié de Publiphone à pièces

1- Le prix minimal d'une communication interurbaine est de *1* € .
2- Après l'introduction de la monnaie, l'utilisateur a deux minutes pour composer son numéro (*ce délai est décompté par le standard*).
3- La ligne peut être libre ou occupée.
4- Le correspondant peut raccrocher le premier.
5- Le Publiphone consomme de l'argent dès que l'appelé décroche et à chaque unité de temps (*UT*) générée par le standard.
6- On peut ajouter des pièces à tout moment.
7- Lors du raccrochage, le solde de monnaie est rendu.

ÉTUDE DE CAS

1- Identifier les acteurs

2- Construire le diagramme de contexte statique

3- Identifier les cas d'utilisation

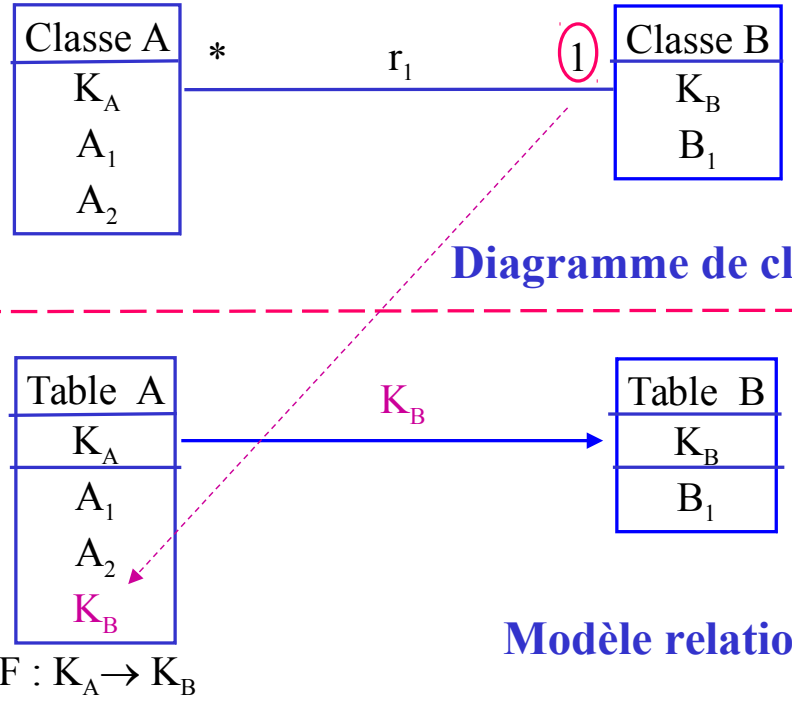
4- Construire un diagramme de séquence système.

5- Construire le diagramme de contexte dynamique.

6- Élaborer le diagramme d'états du Publiphone.

Diagramme de classes vers le modèle relationnel

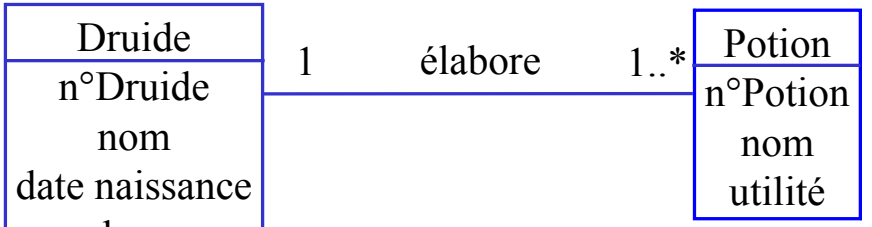
Diagramme de classes vers le modèle relationnel



Druide			Potion
n°Druide {unique} nom date naissance adresse	1	élabore	1..*	n°Potion {unique} nom utilité

Quel est le modèle relationnel ?

Solution



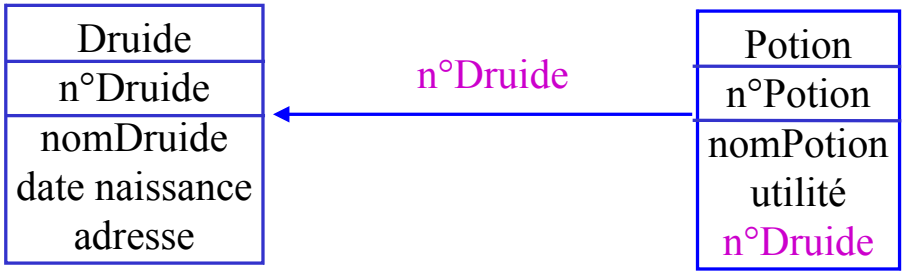
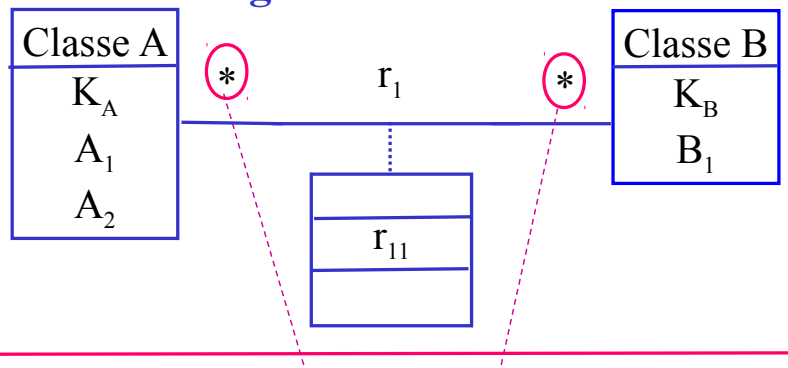
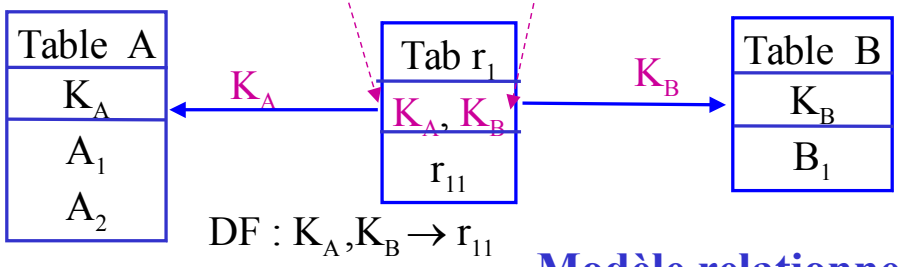


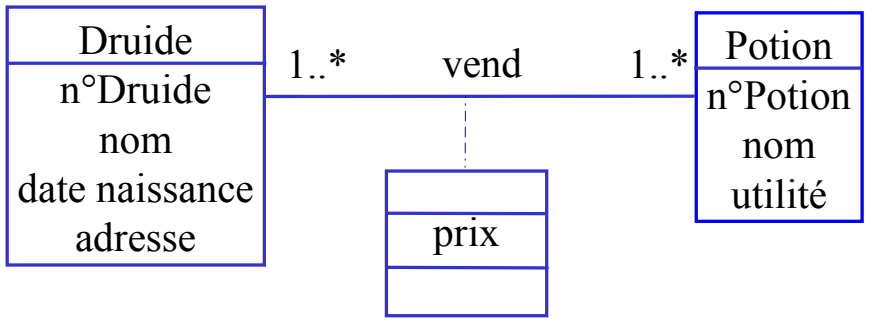
Diagramme de classes





DF : $K_A, K_B \rightarrow r_{11}$

Modèle relationnel



Quel est le modèle relationnel ?

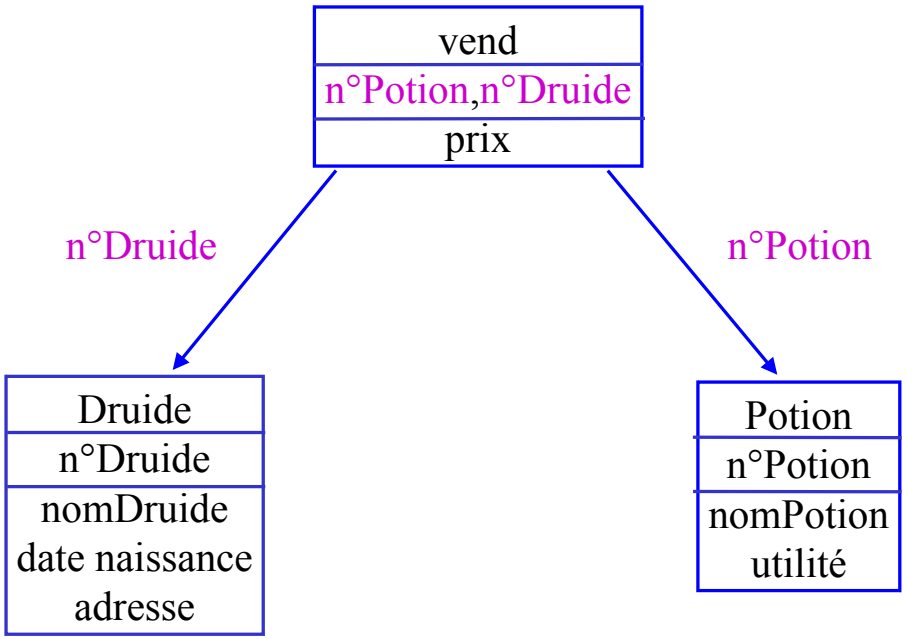
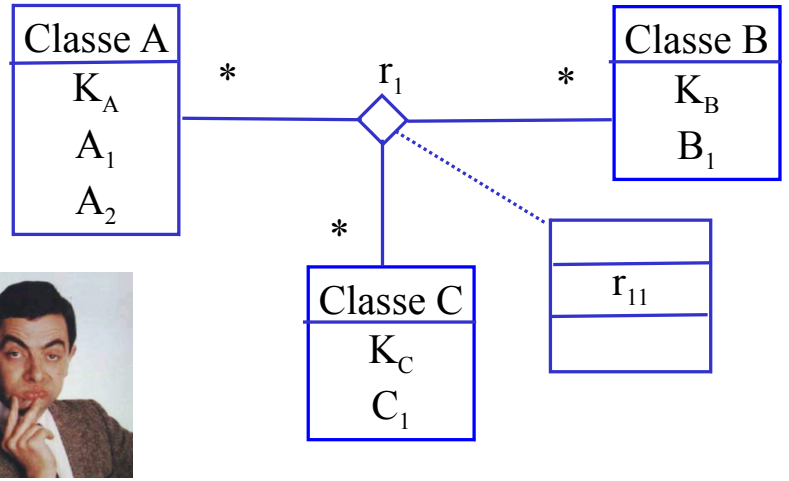
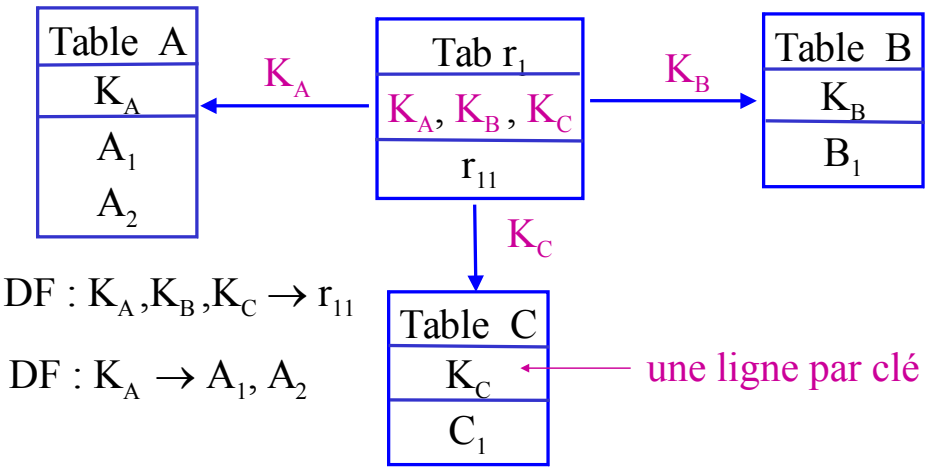


Diagramme de classes



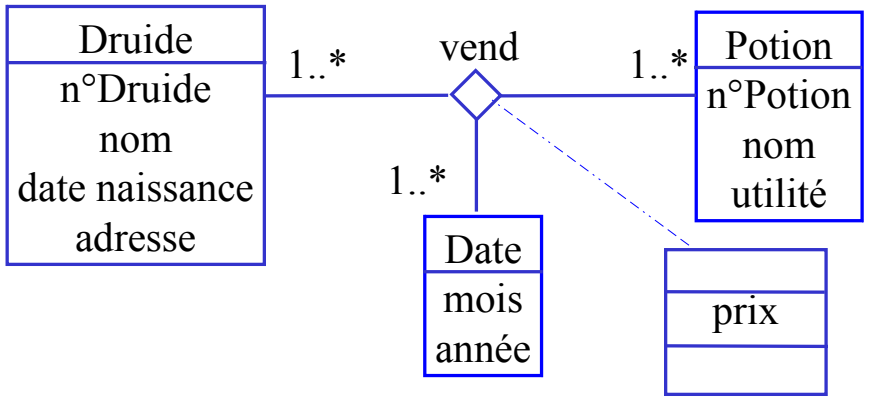
Quel est le modèle relationnel ?

Modèle relationnel



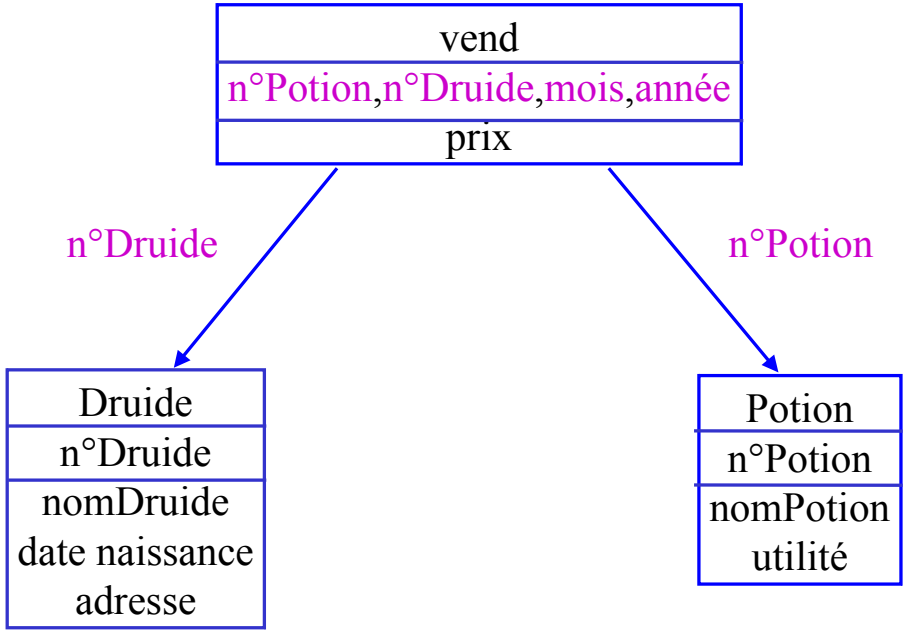
DF : $K_A, K_B, K_C \rightarrow r_{11}$

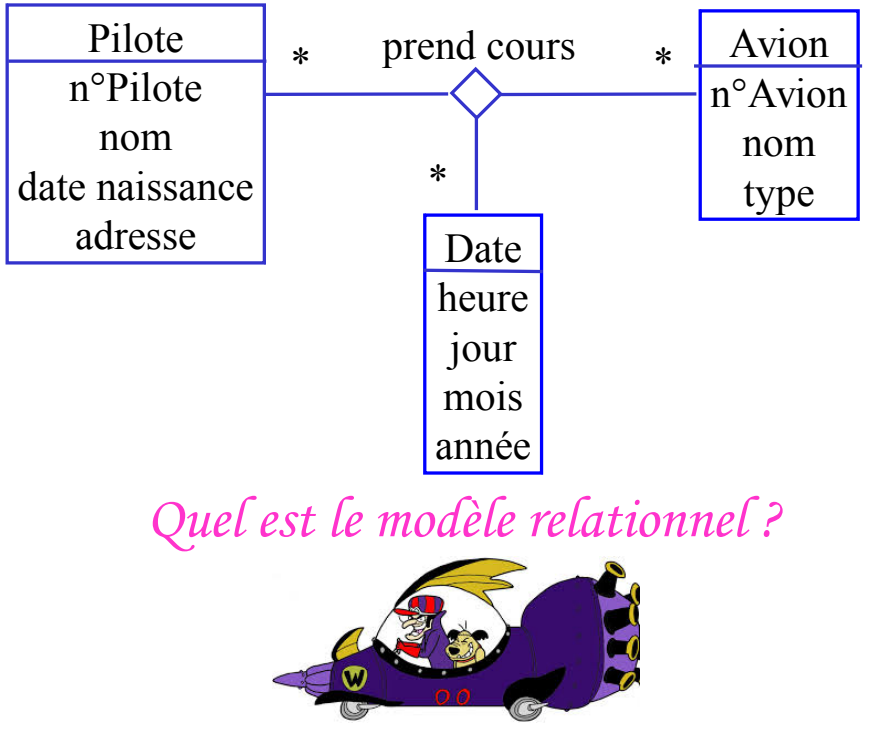
DF : $K_A \rightarrow A_1, A_2$



Quel est le modèle relationnel ?

Solution





Quel est le modèle relationnel ?

Solution

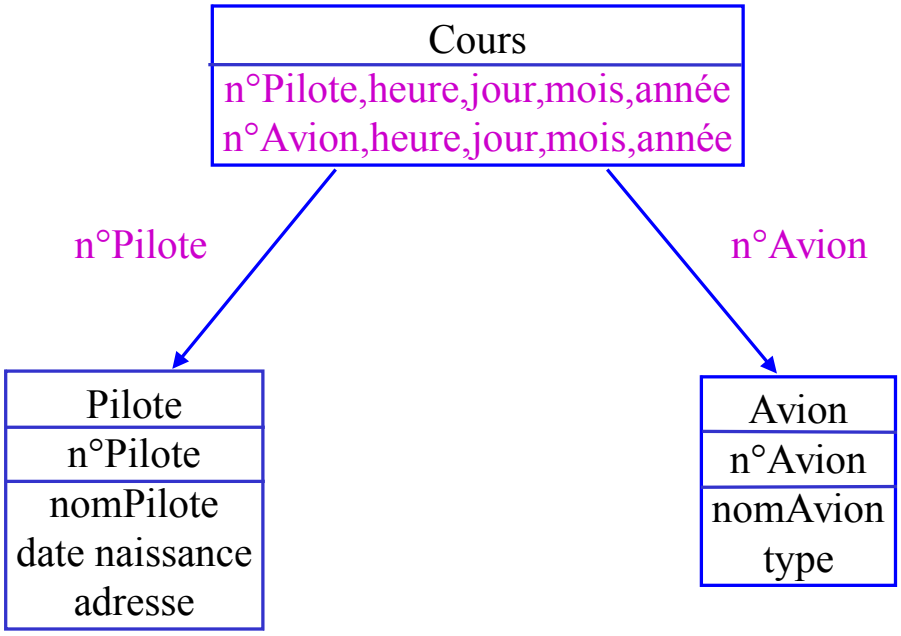
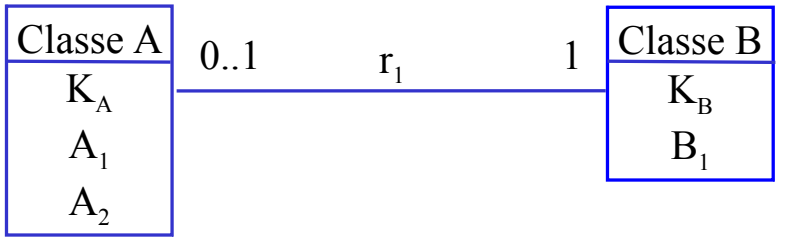
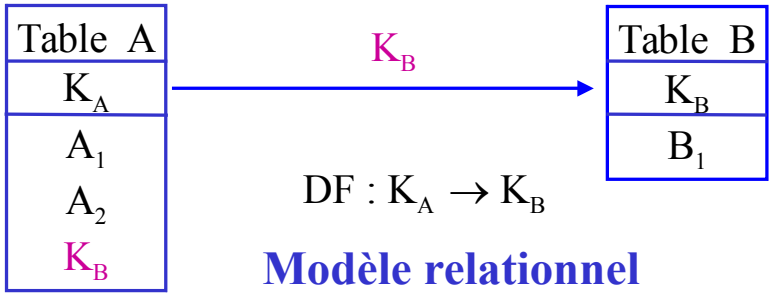


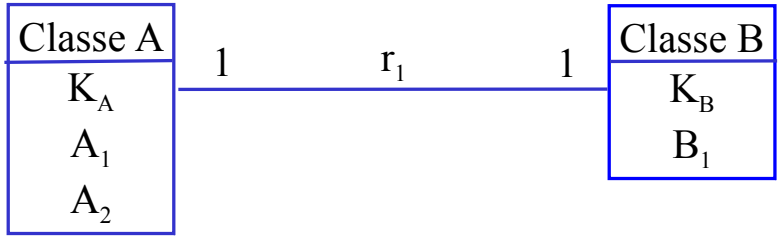
Diagramme de classes





Modèle relationnel

Diagramme de classes



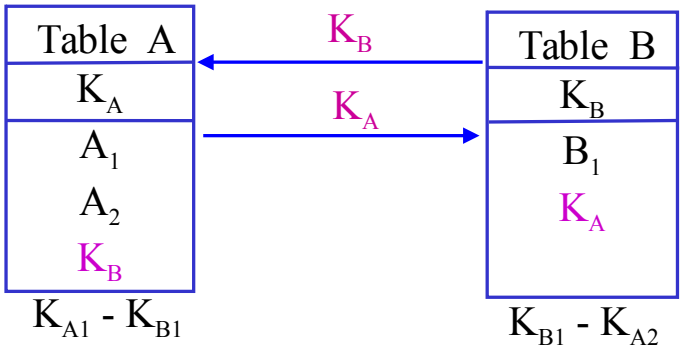
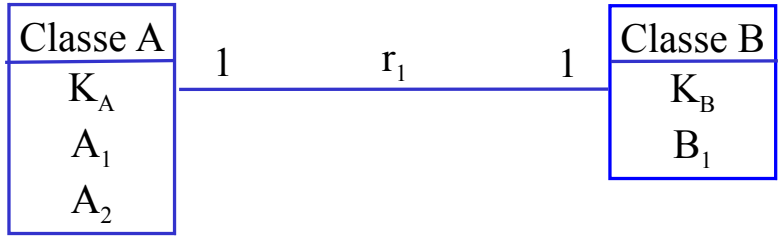


Diagramme de classes



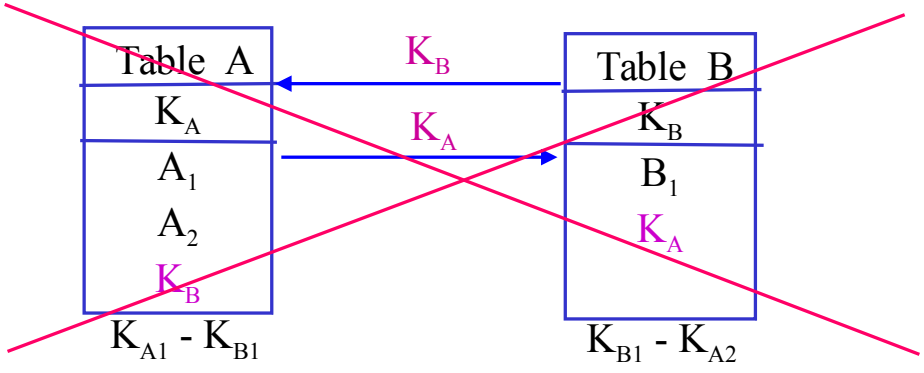
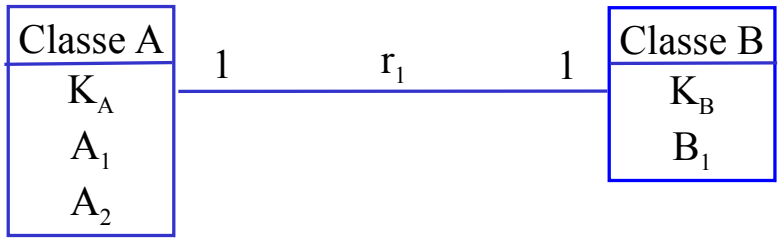
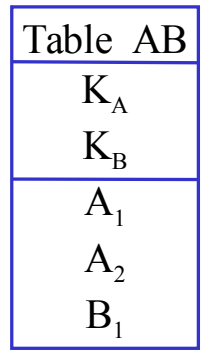


Diagramme de classes





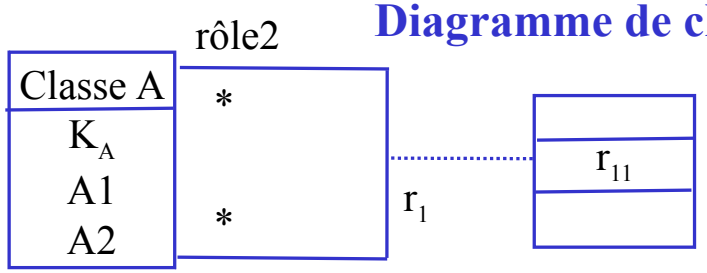
2 clés candidates : K_A et K_B

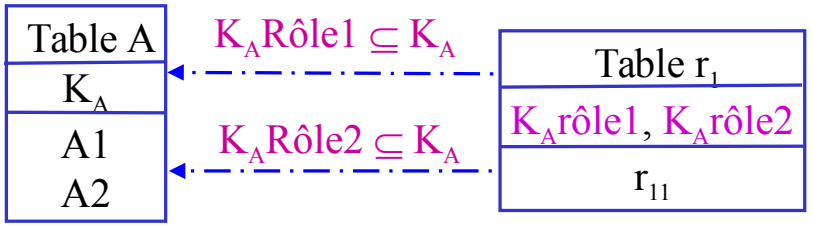
1 clé primaire : K_A ou K_B

K_{A1} - K_{B1}

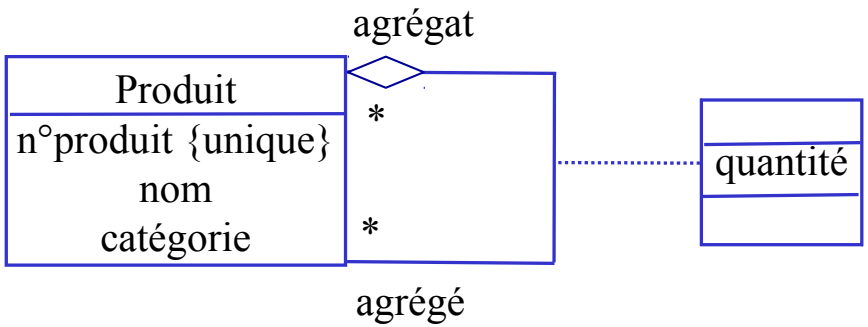
Modèle relationnel

Diagramme de classes





Modèle relationnel



Quel est le modèle relationnel ?

Solution

Modèle relationnel

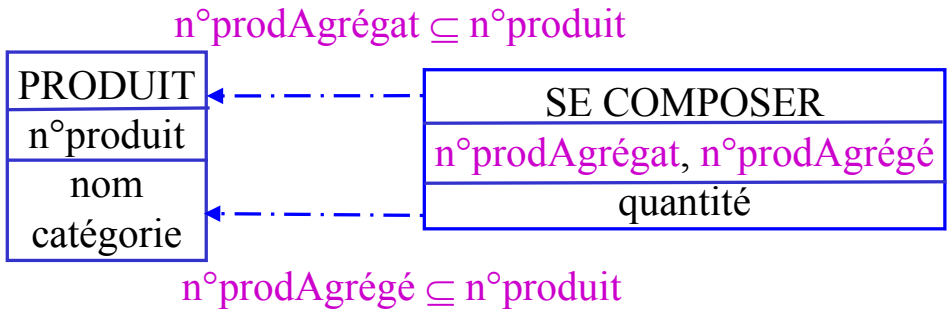
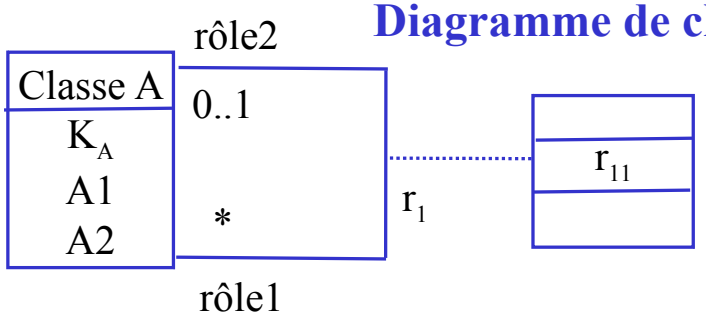


Diagramme de classes

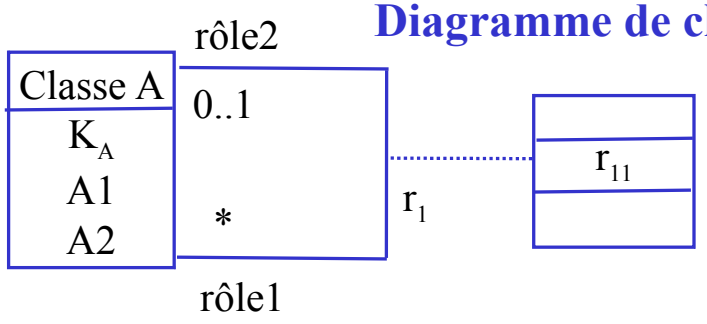


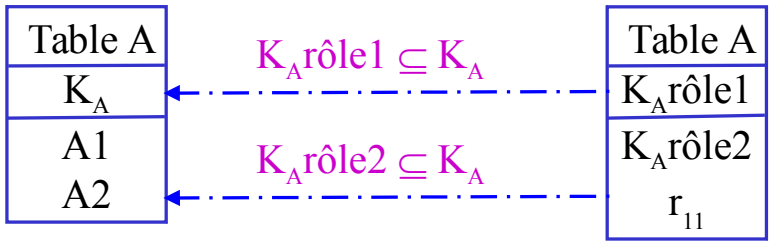
Identifiant de r_1 : $K_A$rôle1

Table A
K_A
A1
A2
$K_A$rôle2 [0..1]
r_{11} [0..1]

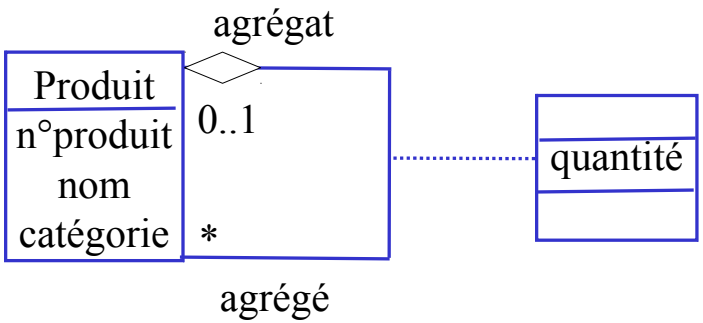
Modèle relationnel

Diagramme de classes





Modèle relationnel



Quel est le modèle relationnel ?

Solution

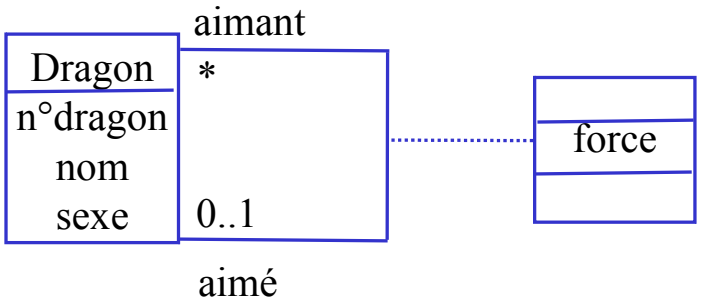
n°produit→nom, catégorie

n°produitComposant →n°produitComposite,quantité
f

Modèle relationnel

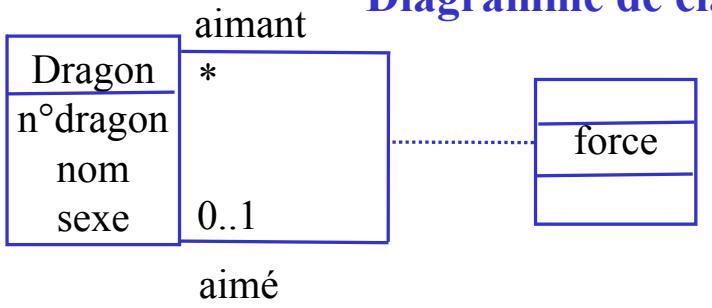
PRODUIT
n°produit
nom
catégorie
n°produitComposite [0..1]
quantité [0..1]

n°produitComposite ⊆ n°produit

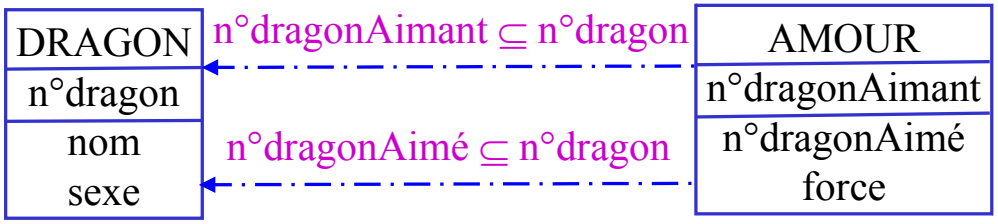


Quel est le modèle relationnel ?

Diagramme de classes



Modèle relationnel



TRANSFORMATION DE L'HÉRITAGE

1- Décomposition par distinction

2- Décomposition descendante (push-down)

3- Décomposition ascendante (push-up)

1- Décomposition par distinction

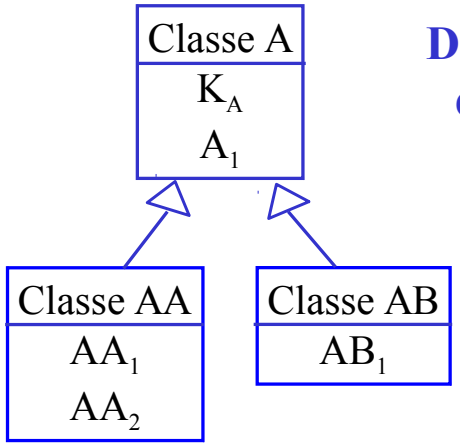


Diagramme de classes

1- Décomposition par distinction

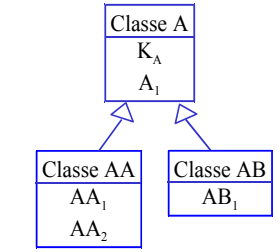
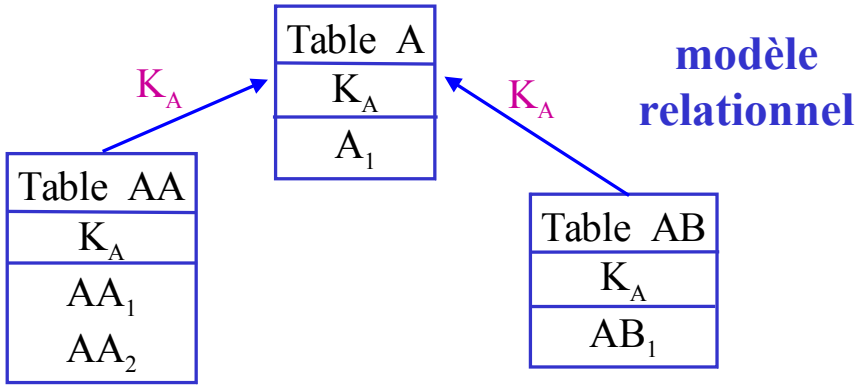


Diagramme de classes



modèle relationnel

1- Décomposition par distinction

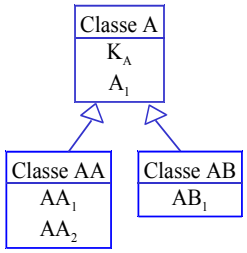
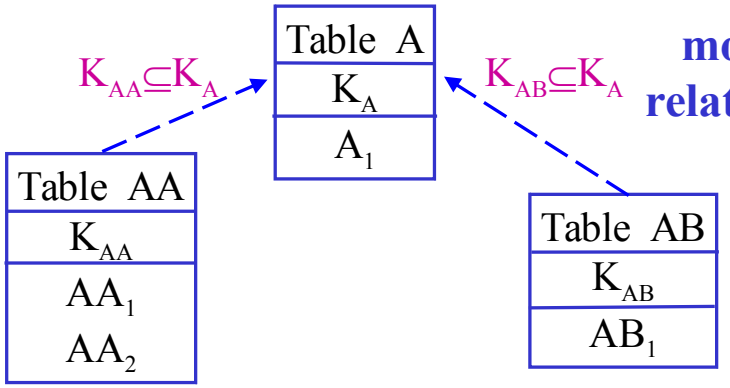


Diagramme de classes



modèle relationnel

1- Décomposition par distinction

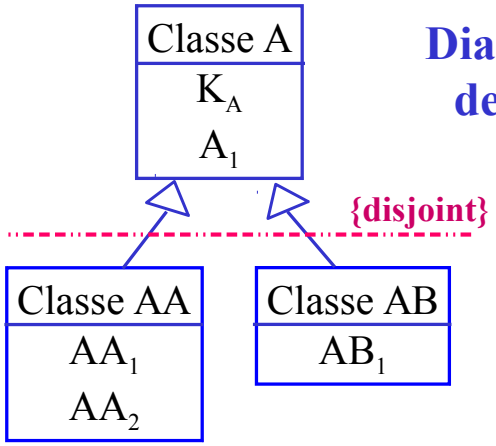


Diagramme de classes

1- Décomposition par distinction

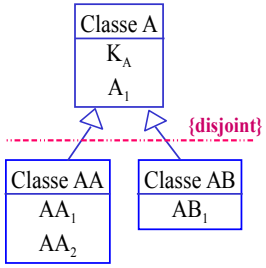
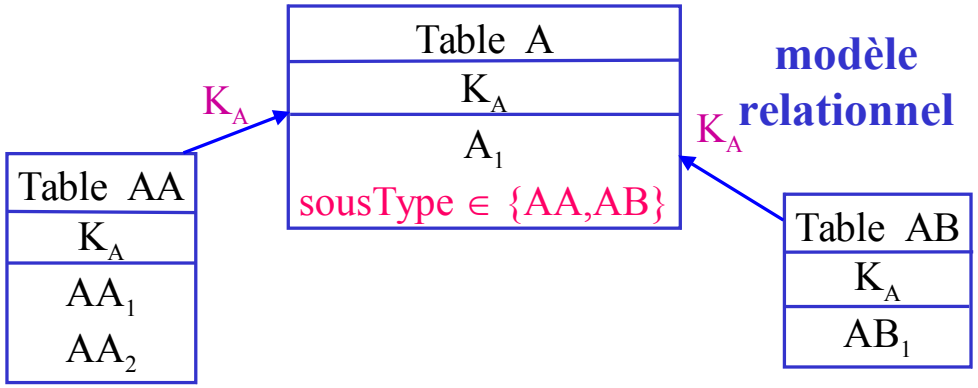


Diagramme de classes



modèle relationnel

2- Décomposition descendante

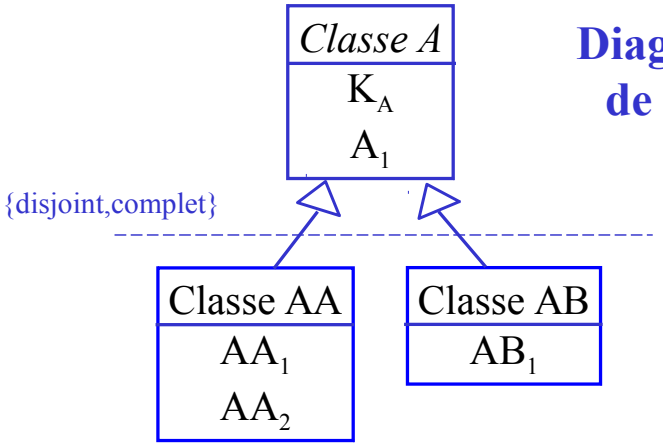


Diagramme de classes

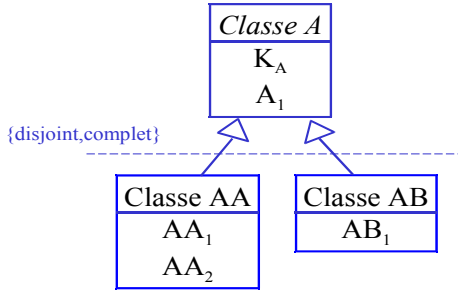
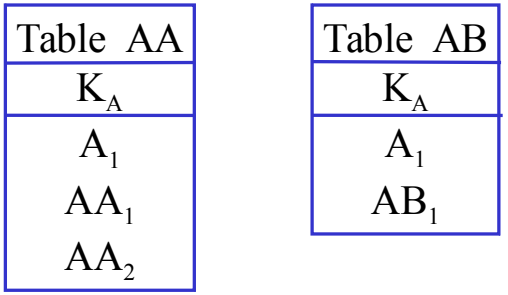


Diagramme de classes



modèle relationnel

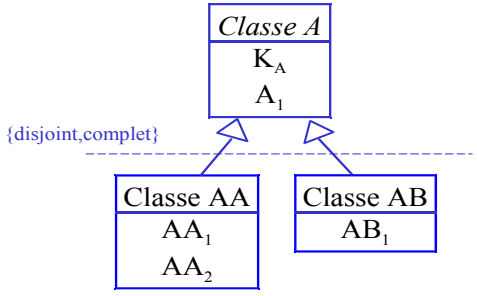
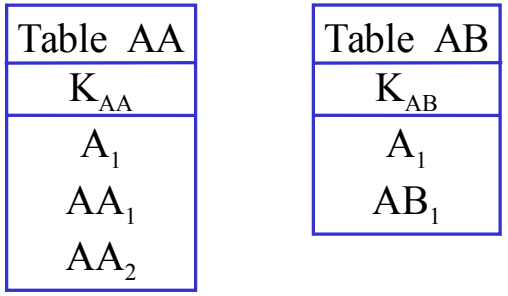


Diagramme de classes



modèle relationnel

3- Décomposition ascendante

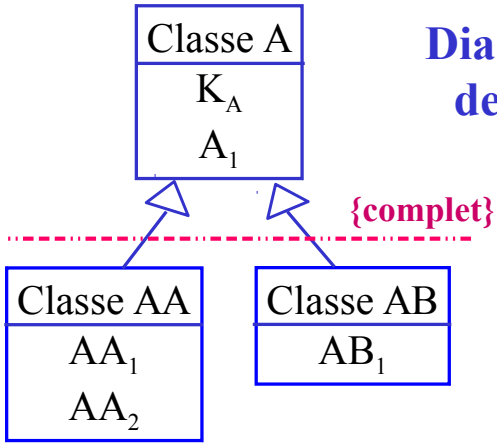


Diagramme de classes

3- Décomposition ascendante

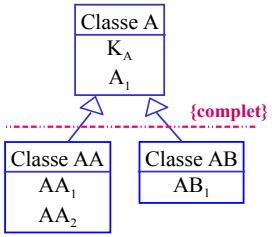


Diagramme de classes

Table A
K_A
A_1
AA_1
AA_2
AB_1

modèle relationnel

3- Décomposition ascendante

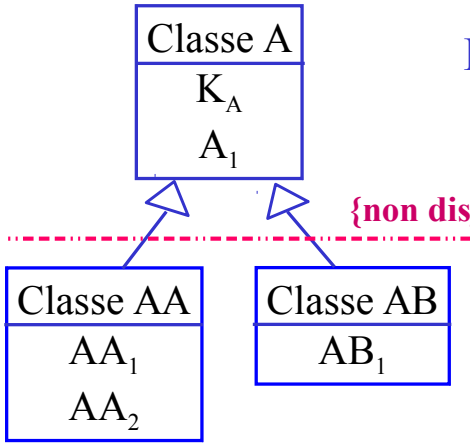


Diagramme de classes

3- Décomposition ascendante

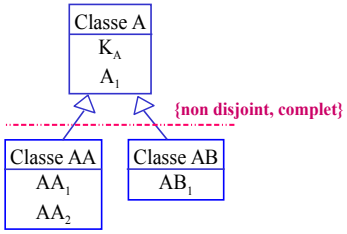


Diagramme de classes

Table A
K_A
A_1
AA_1
AA_2
AB_1
AA ?
AB ?

booléen

modèle relationnel

3- Décomposition ascendante

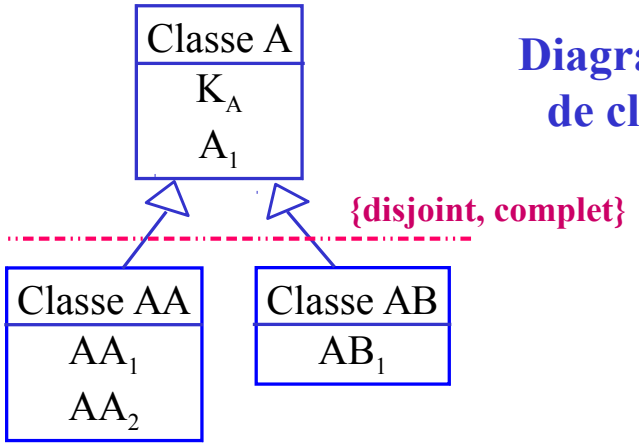


Diagramme de classes

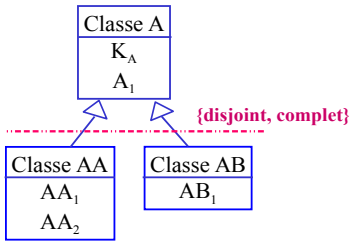


Diagramme de classes

Table A
K_A
A_1
AA_1
AA_2
AB_1
sousType ∈ {AA,AB}

modèle relationnel

3- Décomposition ascendante

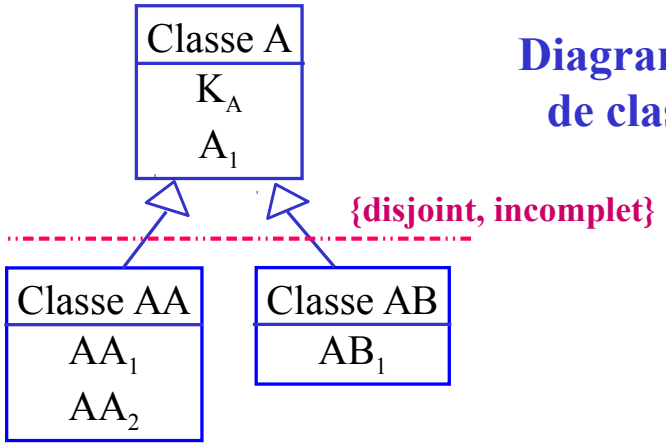


Diagramme de classes

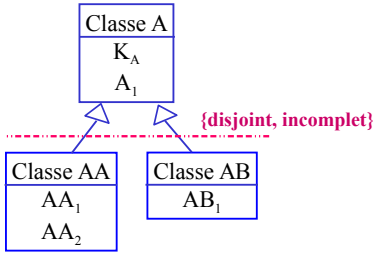
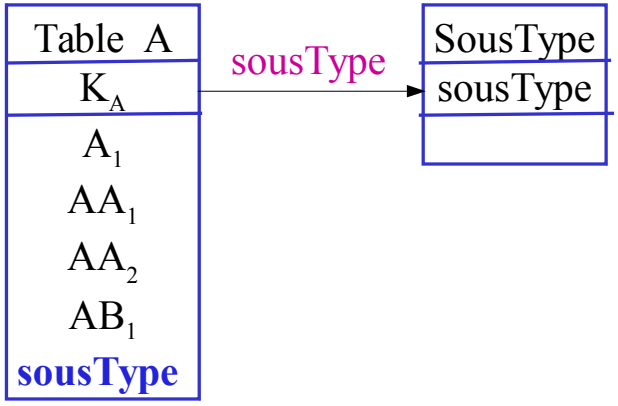
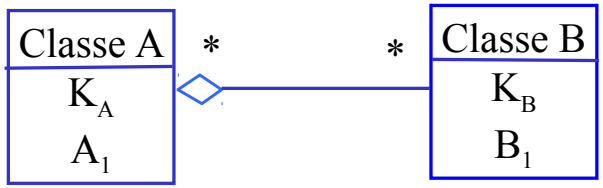


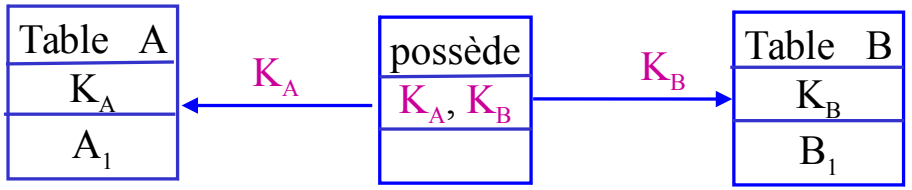
Diagramme de classes



modèle relationnel

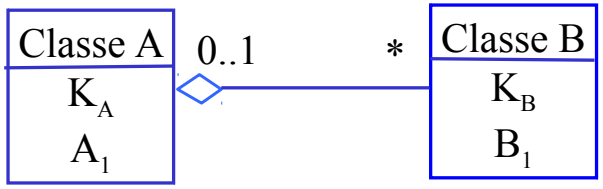
diagramme de classes

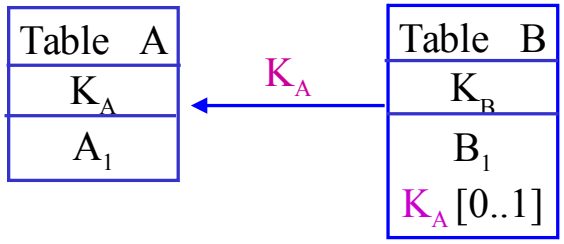




modèle relationnel

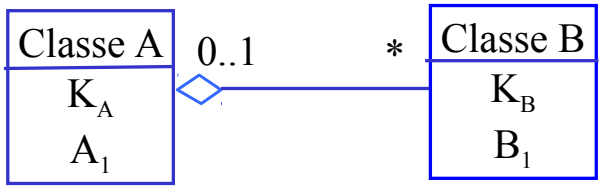
diagramme de classes

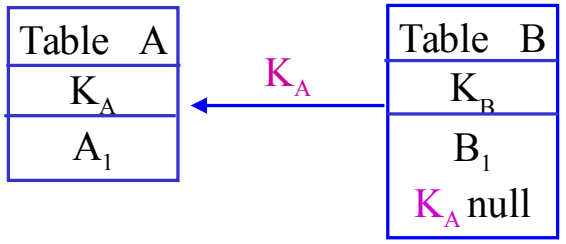




modèle relationnel

diagramme de classes





modèle relationnel

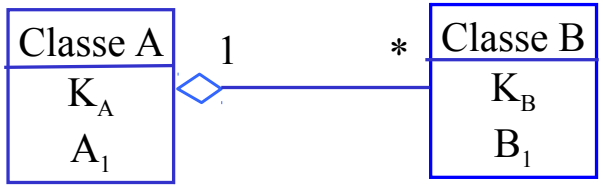
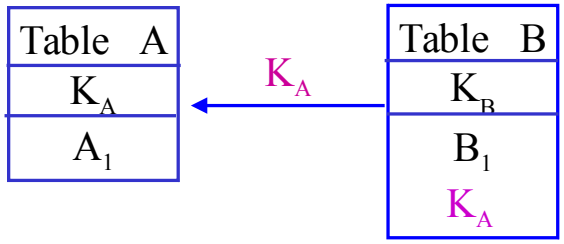


diagramme de classes



modèle relationnel

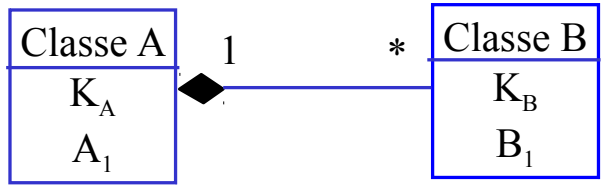
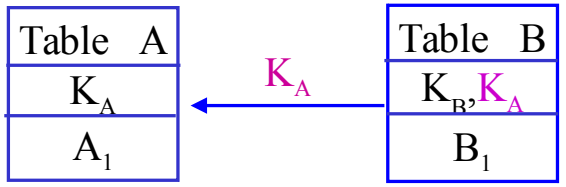
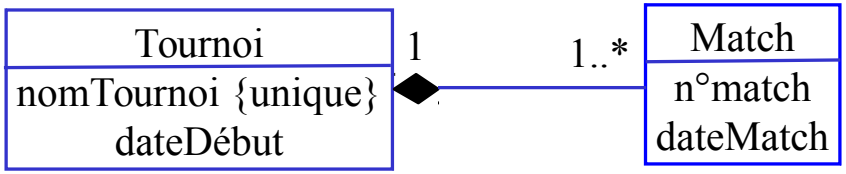


diagramme de classes



modèle relationnel

Diagramme de classes



Quel est le modèle relationnel ?

Solution

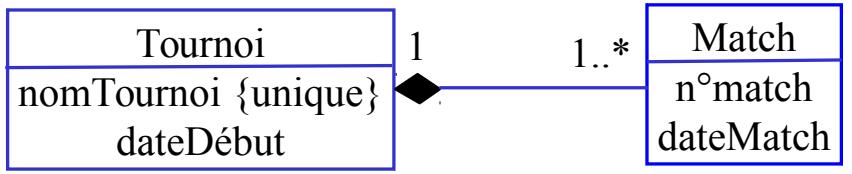
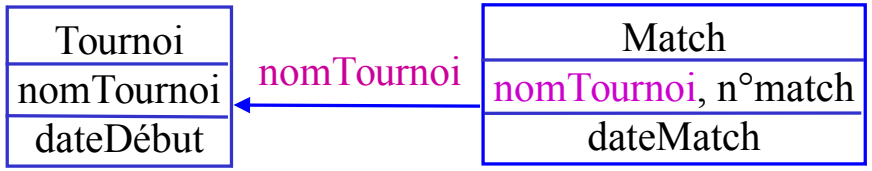
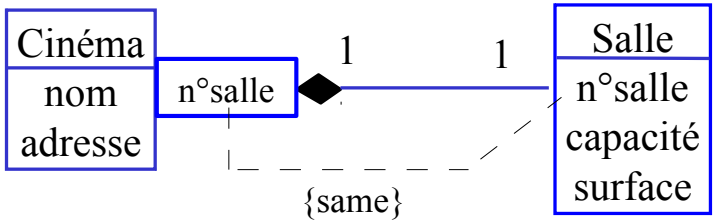


diagramme de classes



modèle relationnel

Diagramme de classes



Quel est le modèle relationnel ?

Solution

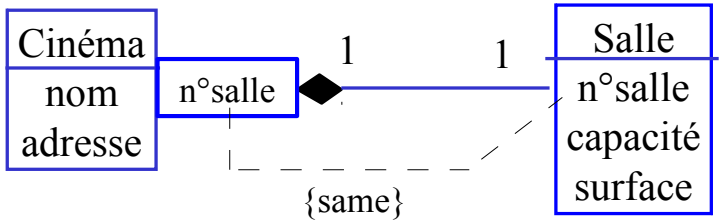
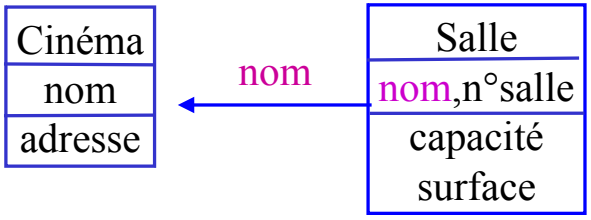


diagramme de classes



modèle relationnel

EXERCICE STYLOS / FEUTRES

- On s'intéresse aux deux classes suivantes : *Feutre* et *Stylo*.

- Un crayon (*feutre, stylo ou autre*) est utilisé pour écrire. Bien évidemment, pour pouvoir écrire, la pointe du stylo devra être sortie et le feutre débouché!

- Ces deux classes ont un certain nombre de propriétés communes (*couleur, marque, niveau d'encre*, etc.) mais comptent aussi un certain nombre de différences : les feutres ont un bouchon (*de la même couleur que celle de l'encre*) alors que les stylos n'ont qu'une pointe rétractable.

- Un feutre peut perdre son bouchon et un bouchon le corps de son feutre d'origine.

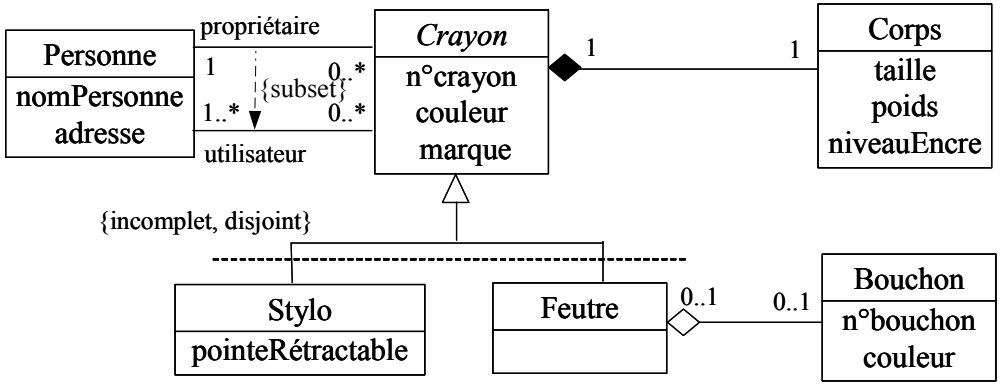
EXERCICE STYLOS / FEUTRES

- Le corps des feutres et des stylos a une certaine taille et un certain poids.

- Évidemment, ces feutres et ces stylos appartiennent à un propriétaire et peuvent être utilisés par un certain nombre de personnes.



Quel est le modèle relationnel ?



{Crayon.couleur = Bouchon.couleur}