Diagramme de Classes

Diagrammes de classe aux différentes étapes

- **On peut utiliser les diagrammes de classes pour représenter un système à différents niveaux d'abstraction :**
  - Le point de vue **spécification** met l'accent sur les interfaces des classes plutôt que sur leurs contenus.
  - Le point de vue **conceptuel** capture les concepts du domaine et les liens qui les lient. Il s'intéresse peu ou prou à la manière éventuelle d'implémenter ces concepts et relations et aux langages d'implantation.
  - Le point de vue **implantation**, le plus courant, détaille le contenu et l'implantation de chaque classe.

- **Les diagrammes de classes s'étoffent à mesure qu'on va de hauts niveaux à de bas niveaux d'abstraction (de la spécification vers l'implantation)**

Objectif

- **Les diagrammes de cas d'utilisation modélisent à QUOI sert le système.**

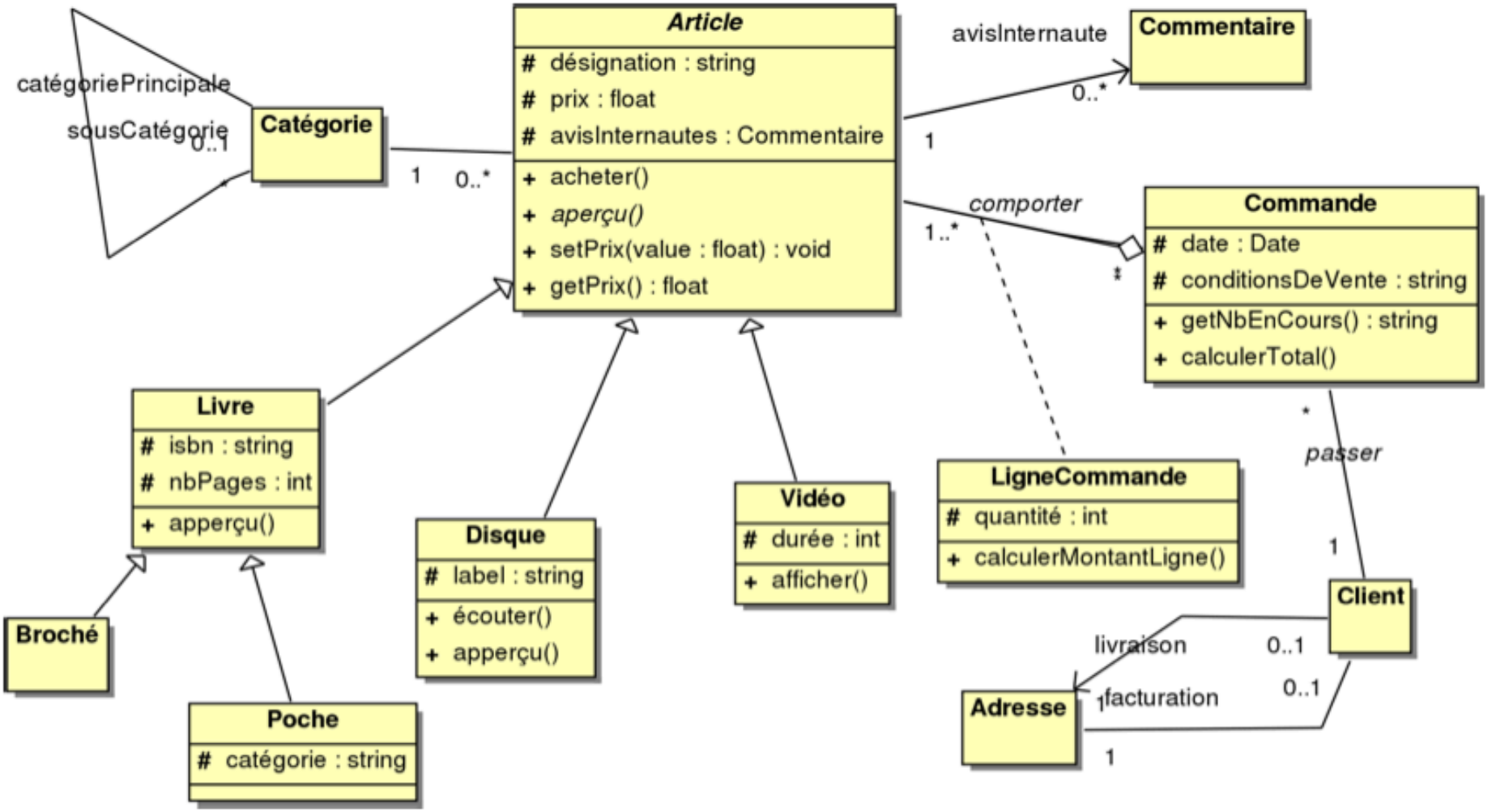
- **Le système est composé d'objets qui interagissent entre eux et avec les acteurs pour réaliser ces cas d'utilisation.**

- **Les diagrammes de classes permettent de spécifier**
  - la structure et
  - les liens entre les objets dont le système est composé.

Construction d'un diagramme de classes

- **Trouver les classes du domaine étudié ;**
  - Souvent, concepts et substantifs du domaine.
- **Trouver les associations entre classes ;**
  - Souvent, verbes mettant en relation plusieurs classes.
- **Trouver les attributs des classes ;**
  - Souvent, substantifs correspondant à un niveau de granularité plus fin que les classes.
  - Les adjectifs et les valeurs correspondent souvent à des valeurs d'attributs.
- **Organiser et simplifier le modèle en utilisant l'héritage ;**
- **Tester les chemins d'accès aux classées ;**
- **Itérer et raffiner le modèle.**

Un exemple pour commencer



Concepts et Instances

- **Une instance est la concrétisation d'un concept abstrait.**
  - Concept : Stylo
  - Instance : le stylo que vous utilisez est une instance du concept stylo : il a sa propre forme, sa propre couleur, son propre niveau d'usure, etc.
- **Un objet est une instance d'une classe**
  - Classe : Vidéo
  - Objets : Pink Floyd (Live à Pompey), 2001 Odyssée de l'Espace etc.

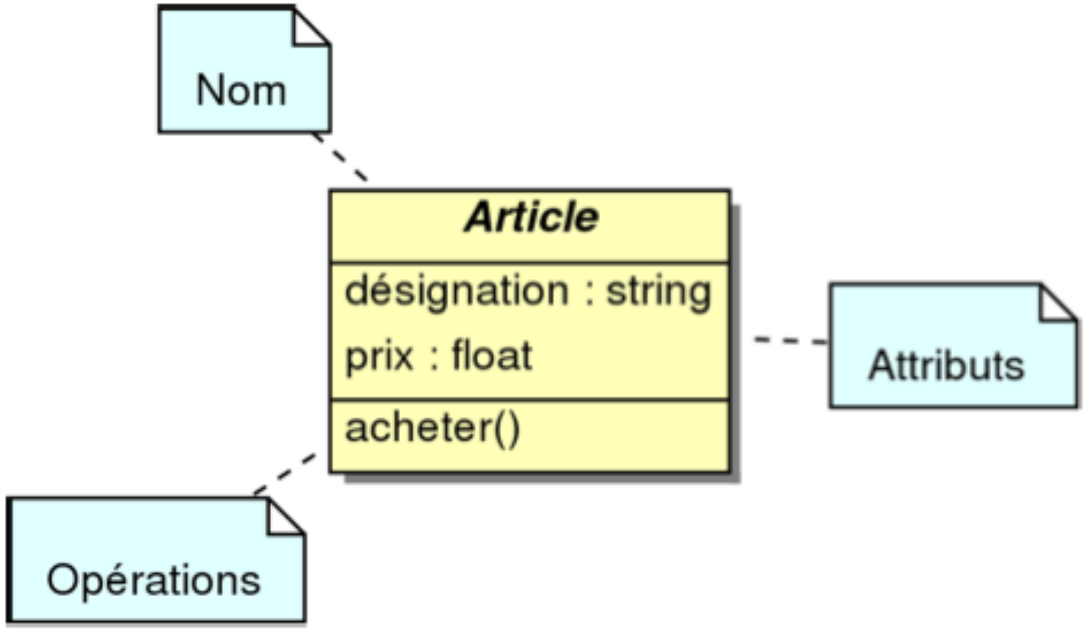
Une classe décrit un type d'objets concrets.

  - Une classe spécifie la manière dont tous les objets de même type seront décrits (désignation, label, auteur, etc).
- **Un lien est une instance d'association.**
  - Association : Concept « avis d'internaute » qui lie commentaire et article
  - Lien : instance [Ali avec son avis négatif], [Mohamed avec son avis positif]

Classes et Objets

- **Une classe est la description d'un ensemble d'objets ayant**
  - une sémantique, des attributs, des méthodes et des relations en commun.
  - Elle spécifie l'ensemble des caractéristiques qui composent des objets de même type.
- **Une classe est composée**
  - d'un nom,
  - d'attributs et
  - d'opérations.
  - Qui ne sont pas forcement toutes connues.

  (niveau de la modélisation)



- **D'autres compartiments peuvent être ajoutés :**
  - responsabilités, exceptions, etc.

Propriétés : attributs et opérations

- **Les attributs et les opérations sont les propriétés d'une classe.**
  - Leur nom commence par une minuscule (convention).

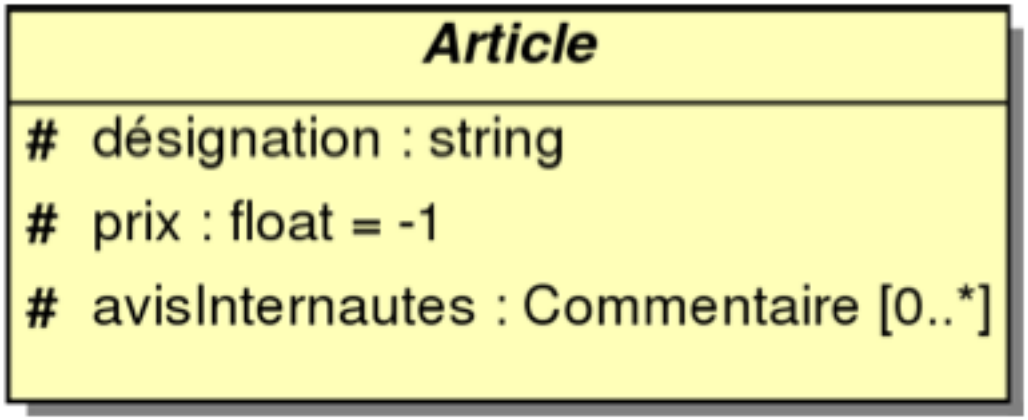
- **Un attribut décrit une donnée de la classe.**
  - Les types des attributs et leurs initialisations ainsi que les modificateurs d'accès peuvent être précisés dans le modèle
  - Les attributs prennent des valeurs lorsque la classe est instanciée : ils sont en quelque sorte des « variables » attachées aux objets.

- **Une opération est un service offert par la classe**
  - un traitement que les objets correspondant peuvent effectuer.

Compartiment des attributs

- **Un attribut peut être initialisé et sa visibilité est définie lors de sa déclaration.**

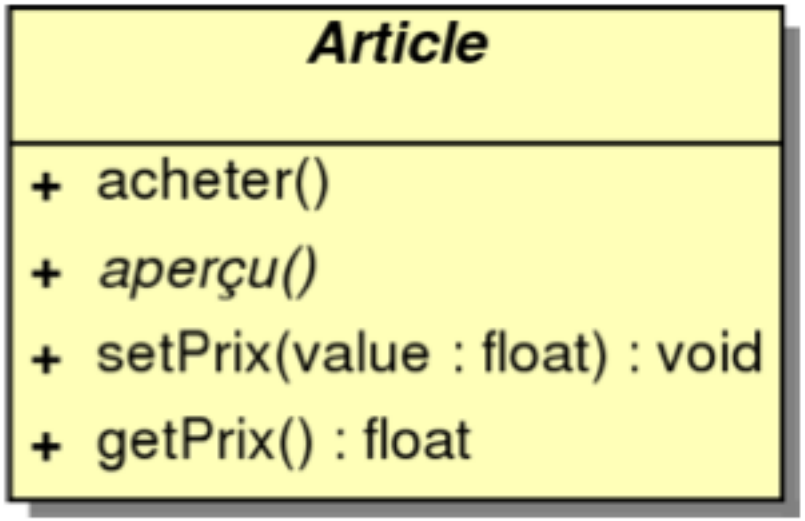
- **Syntaxe de la déclaration d'un attribut :**
  - `modifAcces nomAtt : nomClasse[multi]=valeurInit`



Compartiment des opérations

Une opération est définie par son nom ainsi que par les types de ses paramètres et le type de sa valeur de retour.

- **La syntaxe de la déclaration d'une opération est la suivante :**
  - `modifAcces nomOperation(parametres) : ClasseRetour`
- **La syntaxe de la liste des paramètres est la suivante :**
  - `nomClasse1 nomParam1 ,..., nomClasseN nomParamN`



Méthodes et Opérations

- **Une opération est la spécification d'une méthode (sa signature) indépendamment de son implantation.**
  - UML 2 autorise également la définition des opérations dans n'importe quel langage donné.
- **Exemples de méthodes pour l'opération `fact(n:int):int`**

```
{ // implementation iterative
int resultat =1;
for (int i = n; i>0; i--)
  resultat*=i;
return resultat;
}
{ // implementation recursive
  if (n==0 || n==1)
   return 1;
  return (n * fact(n-1));
}
```

Relations entre classes

- **Une relation d'héritage est une relation de généralisation/ spécialisation permettant l'abstraction.**

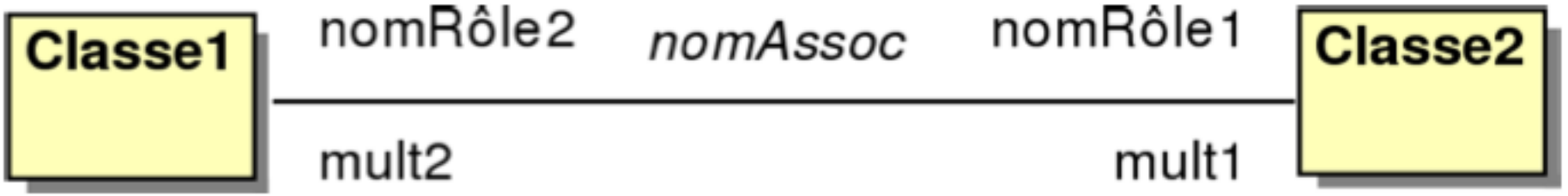
- **Une dépendance est une relation unidirectionnelle exprimant une dépendance sémantique entre les éléments du modèle (flèche ouverte pointillée).**

- **Une association représente une relation sémantique entre les objets d'une classe.**

- **Une relation d'agrégation décrit une relation de contenance ou de composition.**

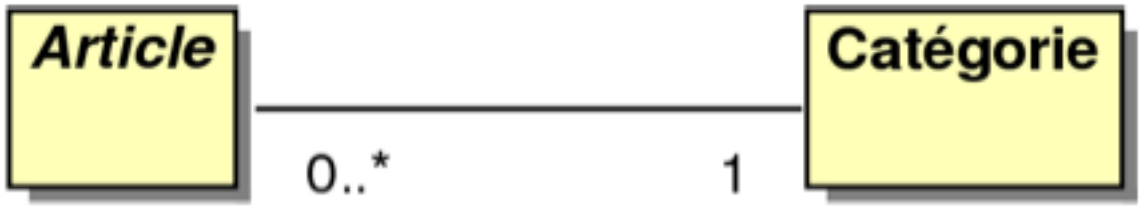
Association

- **Une association est une relation structurelle entre objets.**
- **Une association est souvent utilisée pour représenter les liens possibles entre objets de classes données.**
- **Elle est représentée par un trait entre classes.**



Multiplicités des associations

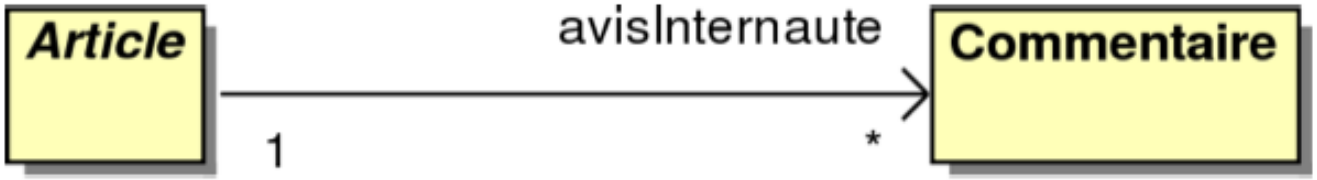
- **La notion de multiplicité permet le contrôle du nombre d'objets intervenant dans chaque instance d'une association.**
  - Exemple : un article n'appartient qu'à une seule catégorie (1) ; une catégorie concerne plus de 0 articles, sans maximum (*).



- **La syntaxe est MultMin..MultMax.**
  - « * » à la place de MultMax signifie « plusieurs » sans préciser le nombre.
  - « n..n » se note aussi « n », et « 0..* » se note « * ».

Navigabilité d'une association

- **La navigabilité permet de spécifier dans quel(s) sens il est possible de traverser l'association à l'exécution.**
  - On restreint la navigabilité d'une association à un seul sens à l'aide d'une flèche.



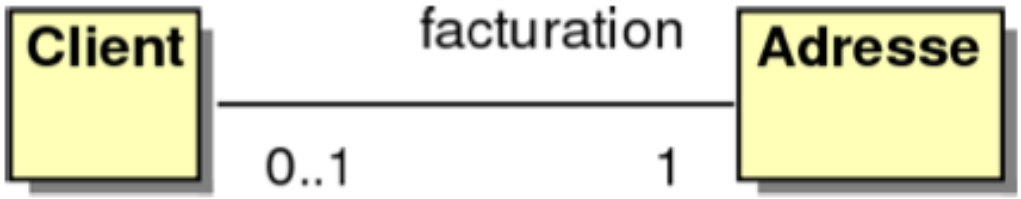
  - Exemple : Connaissant un article on connaît les commentaires, mais pas l'inverse.
- **On peut aussi représenter les associations navigables dans un seul sens par des attributs.**
  - Exemple : En ajoutant un attribut « avisInternaute » de classe « Article » à la place de l'association.

Association unidirectionnelle de 1 vers 1

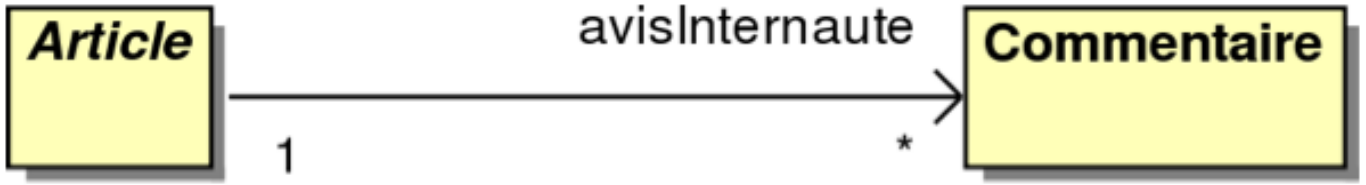
```
public class Adresse {...}

public class Client{
  private Adresse livraison;

  public void setAdresse(Adresse adresse){
      this.livraison = adresse;
  }

  public Adresse getAdresse (){
      return livraison;
   }
}
```

Association bidirectionnelle de 1 vers 1

```
public class Client{
   Adresse facturation;
   public void setFacturation(Adresse uneAdresse){
       if(uneAdresse!=null){
             this.facturation = uneAdresse;
             facturation.client = this; // correspondance
       }
   }
}
public class Adresse{
   Client client;
   public void setClient(Client unClient){
       this.client = unClient;
       client.facturation = this; // correspondance
   }
}
```



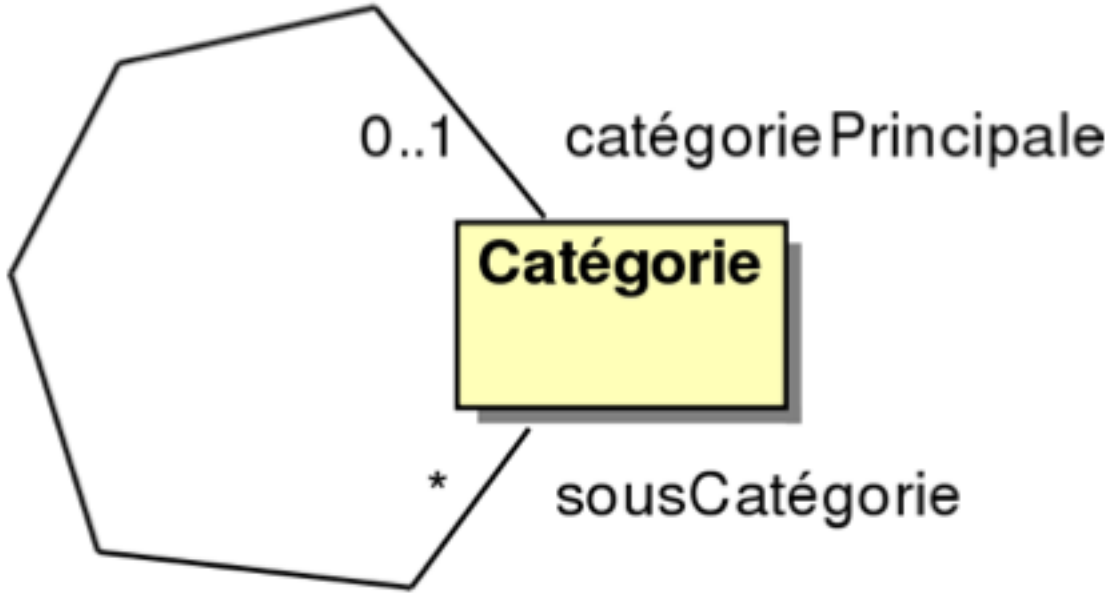
Association unidirectionnelle de 1 vers *




```
public class Commentaire {...}

public class Article{
   private Vector avisInternaute =newVector ();
   public void addCommentaire(Commentaire commentaire){
       avisInternaute.addElement(commentaire);
   }
   public void removeCommentaire(Commentaire commentaire{
       avisInternaute.removeElement(commentaire);
   }
}
```

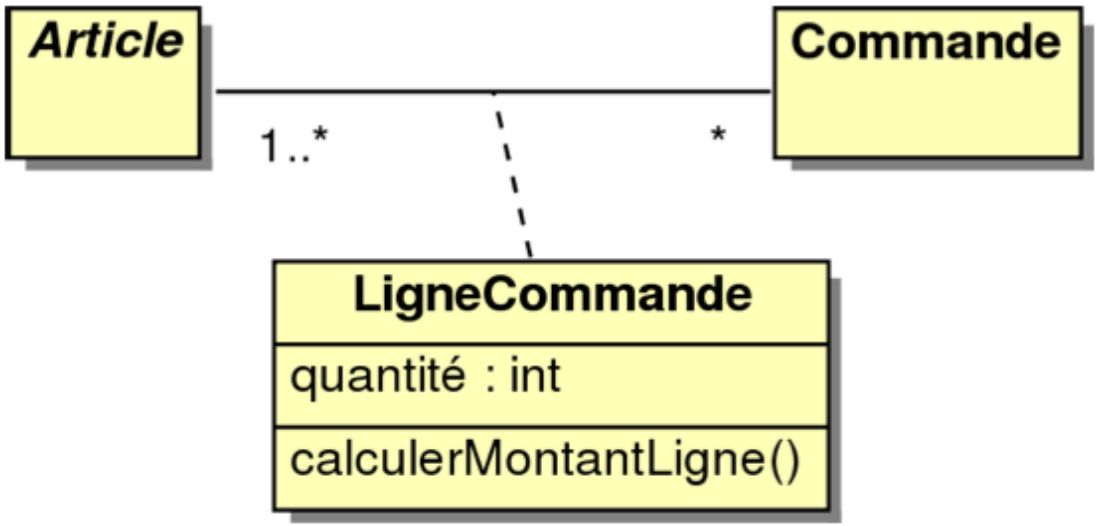
Associations réflexives

- **L'association la plus utilisée est l'association binaire (reliant deux classs).**
- **Parfois, les deux extrémités de l'association pointent vers le même classeur.**
  - Dans ce cas, l'association est dite « réflexive ».



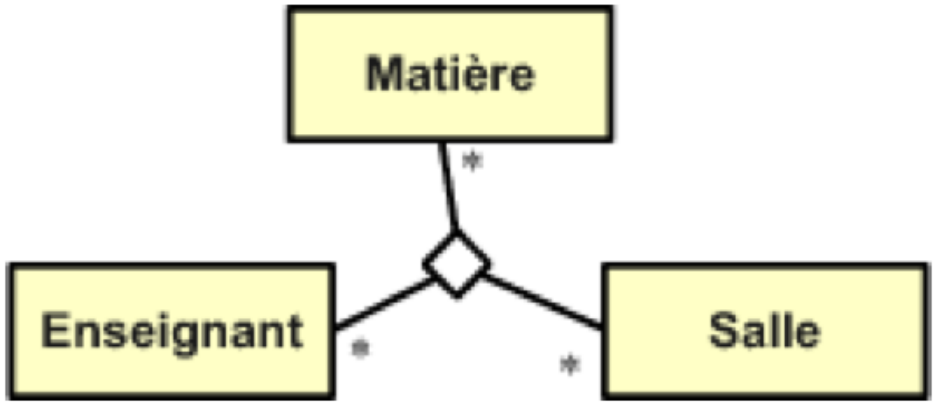
Classe-association

- **Une association peut être raffinée et avoir ses propres attributs,**
  - qui ne sont disponibles dans aucune des classes qu'elle lie.
  - Or seules les classes peuvent avoir des attributs, cette association devient alors une classe appelée « classe-association ».



Associations n-aires

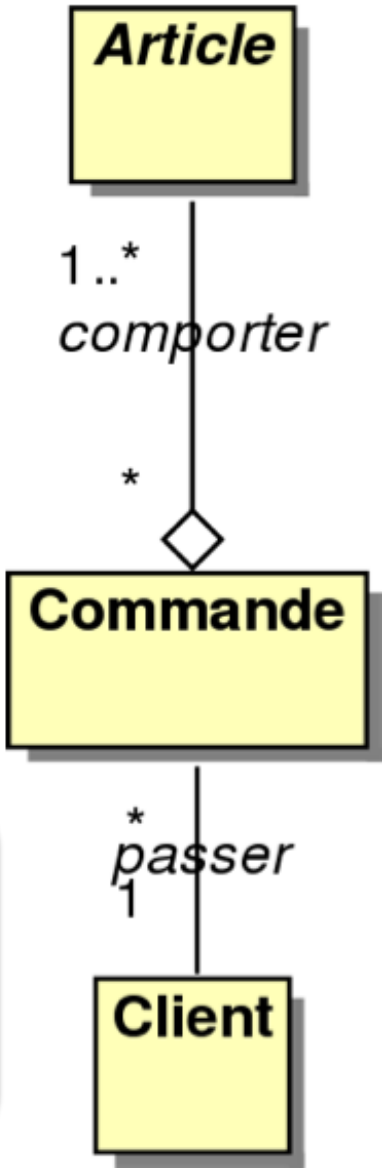
- **Une association n-aire lie plus de deux classes.**
  - Notation avec un losange central pouvant éventuellement accueillir une classe-association.
  - La multiplicité de chaque classe s'applique à une instance du losange.



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

Association de type agrégation

- **Une agrégation est une forme particulière d'association.**
  - C'est une association asymétrique
- **Elle représente la relation d'inclusion d'un élément dans un ensemble.**
- **On représente l'agrégation par l'ajout d'un losange vide du côté de l'agrégat.**



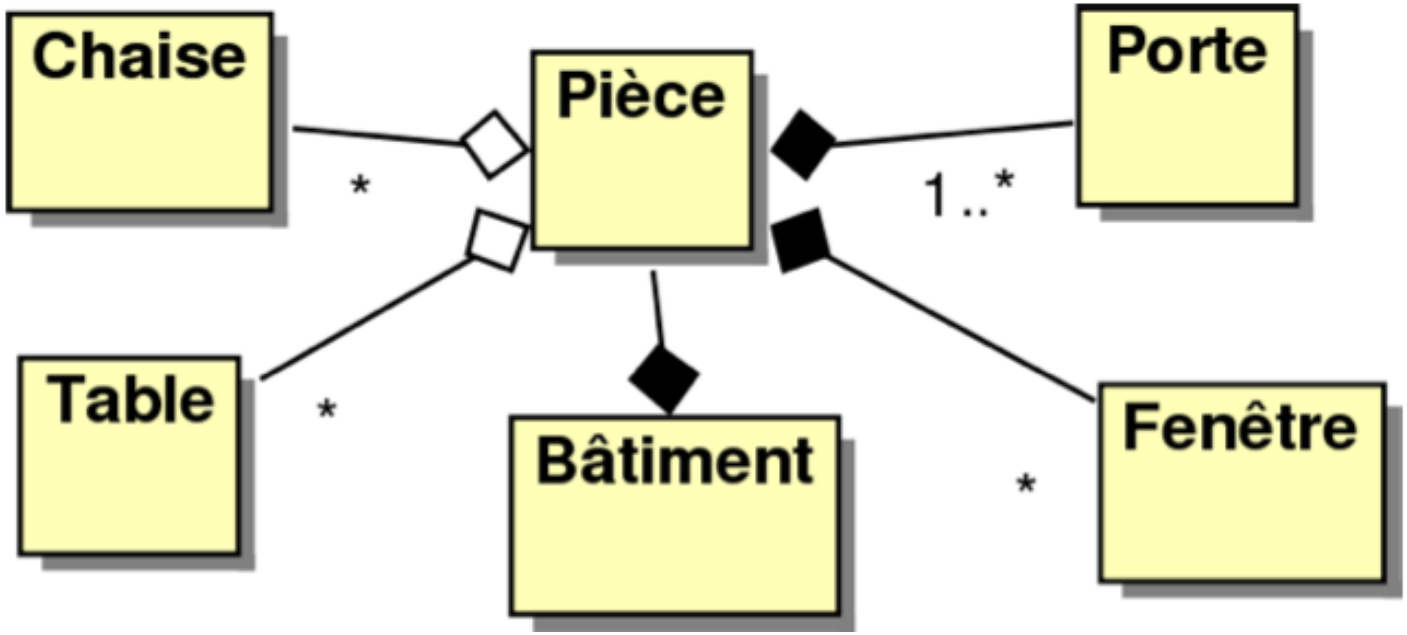
Une agrégation dénote une relation d'un ensemble à ses parties. L'ensemble est **l'agrégat** et la partie **l'agrégé**.

Association de type composition

- **La relation de composition décrit une contenance structurelle entre instances. On utilise un losange plein.**

Remarques

• La **destruction** et la **copie** de l'objet composite (l'ensemble) impliquent respectivement la destruction ou la copie de ses composants (les parties).
• Une instance de la partie **n'appartient jamais** à plus d'une instance de l'élément composite.



Composition et agrégation

- **Dès lors que l'on a une relation du tout à sa partie, on a une relation d'agrégation ou de composition.**

  La composition est aussi dite **« agrégation forte »**.

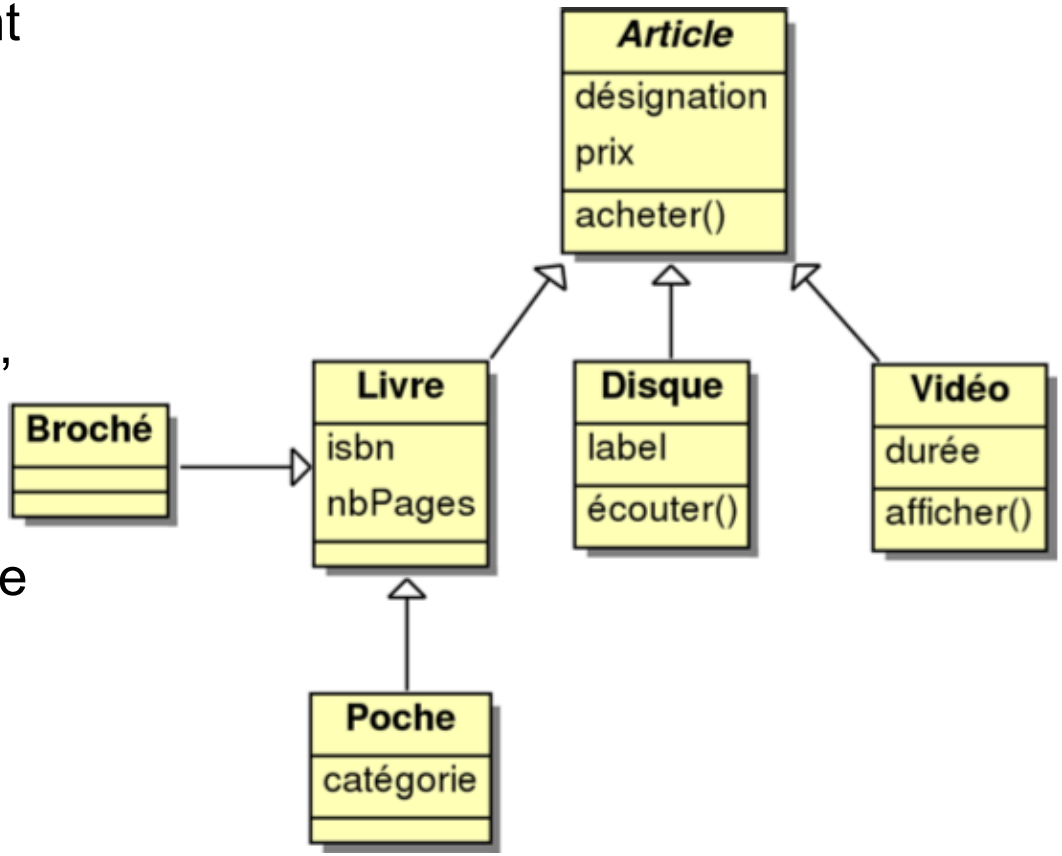
- **Pour décider de mettre une composition plutôt qu'une agrégation, on doit se poser les questions suivantes :**
  - Est-ce que la destruction de l'objet composite (du tout) implique nécessairement la destruction des objets composants (les parties) ? C'est le cas si les composants n'ont pas d'autonomie vis-à-vis des composites.
  - Lorsque l'on copie le composite, doit-on aussi copier les composants, ou est-ce qu'on peut les « réutiliser », auquel cas un composant peut faire partie de plusieurs composites ?
- **Si on répond par l'affirmative à ces deux questions, on doit utiliser une composition.**

Relation d'héritage

- **L'héritage une relation de spécialisation/généralisation.**

  ↳ Les éléments spécialisés héritent de la structure et du comportement des éléments plus généraux (attributs et opérations)

  ↳ Exemple : Par héritage d'Article, un livre a d'office un prix, une désignation et une opération acheter(), sans qu'il soit nécessaire de le préciser

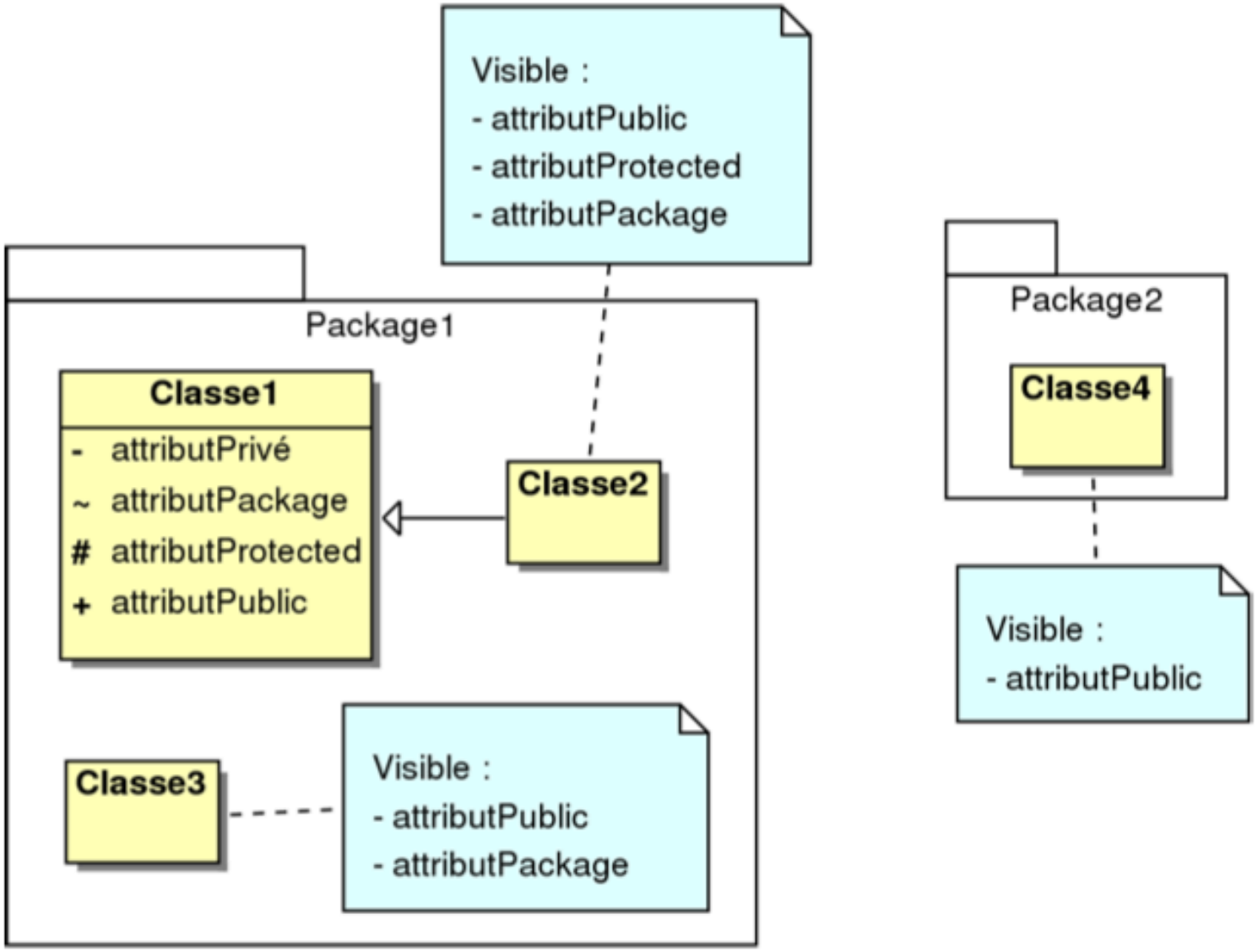


Encapsulation

- **L'encapsulation est un principe de conception consistant à protéger le cœur d'un système des accès intempestifs venant de l'extérieur.**

- **En UML, utilisation de modificateurs d'accès sur les attributs ou les classes :**
  - **public** ou « + » : propriété ou classe visible partout
  - **protected** ou « # ». propriété ou classe visible dans la classe et par tous ses descendants.
  - **private** ou « - » : propriété ou classe visible uniquement dans la classe
  - **package**, ou « ~ » : propriété ou classe visible uniquement dans le paquetage
  - Il n'y a pas de visibilité « par défaut ».

Exemple d'encapsulation



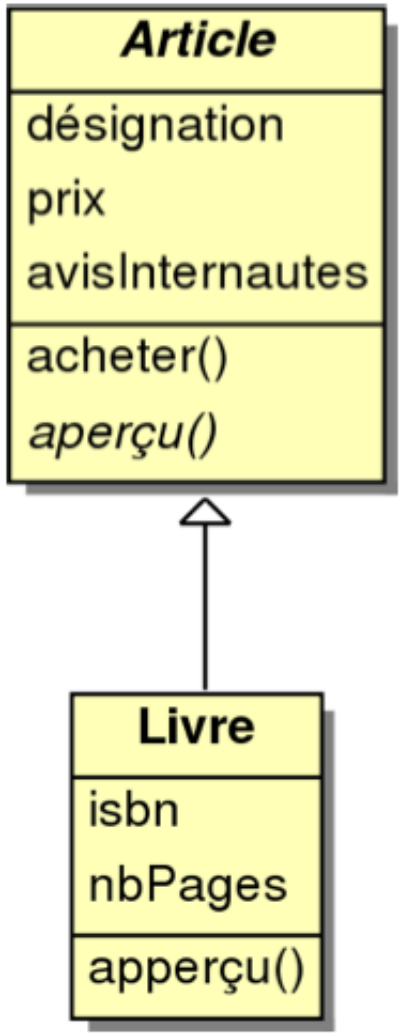
Relation d'héritage et propriétés

- **La classe enfant possède toutes les propriétés de ses classes parents (attributs et opérations)**

  - La classe enfant est la classe spécialisée (ici Livre)

  - La classe parent est la classe générale (ici Article)

  Toutefois, elle n'a pas accès aux propriétés privées.

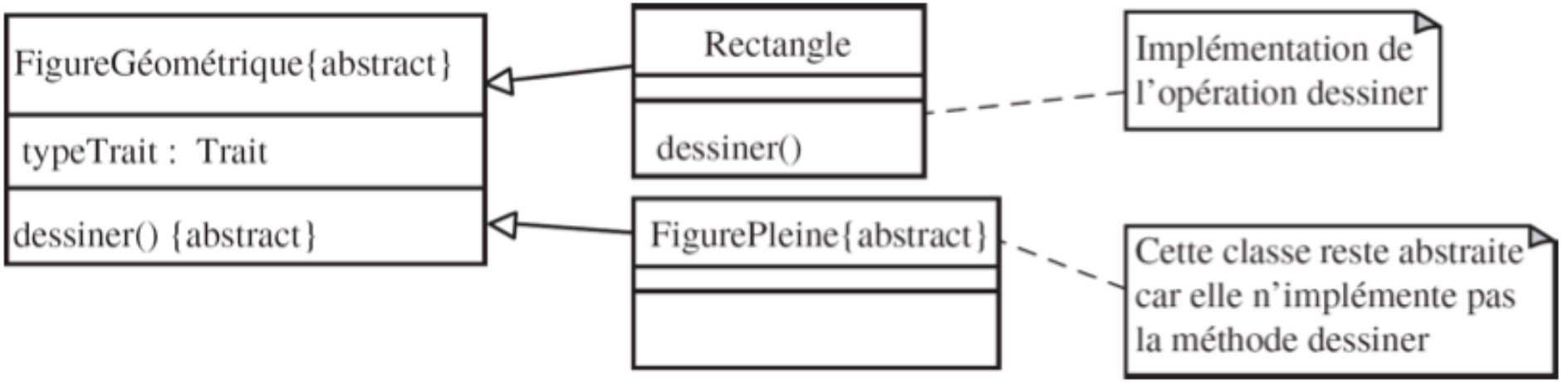


Terminologie de l'héritage

- **Une classe enfant peut redéfinir (même signature) une ou plusieurs méthodes de la classe parent.**
  - Sauf indications contraires, un objet utilise les opérations les plus spécialisées dans la hiérarchie des classes.
  - La **surcharge** d'opérations (même nom, mais signatures des opérations différentes) est possible dans toutes les classes.
- **Toutes les associations de la classe parent s'appliquent, par défaut, aux classes dérivées (classes enfant).**
- **Principe de substitution ou liaison dynamique : une instance d'une classe peut être utilisée partout où une instance de sa classe parent est attendue.**
  - Par exemple, toute opération acceptant un objet d'une classe Animal doit accepter tout objet de la classe Chat (l'inverse n'est pas toujours vrai).

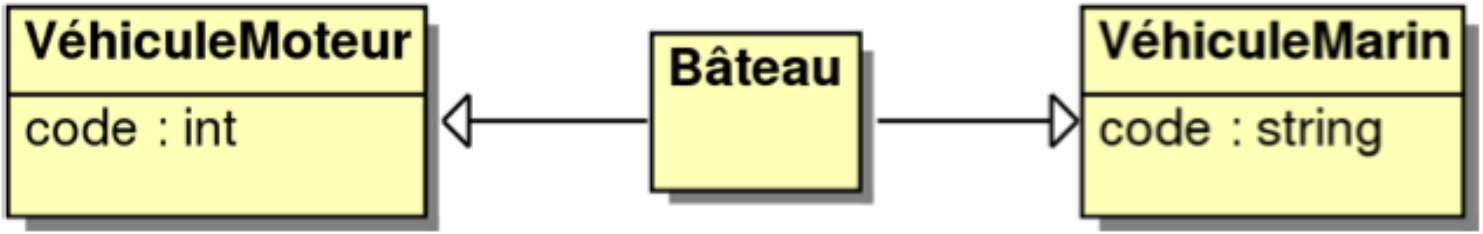
Classes abstraites

- **Une méthode est dite abstraite lorsqu'on connaît son entête mais pas la manière dont elle peut être réalisée.**
- **Il appartient aux classes enfant de définir les méthodes abstraites.**
- **Une classe est dite abstraite**
  - lorsqu'elle définit au moins une méthode abstraite ou
  - lorsqu'une classe parent contient une méthode abstraite non encore réalisée.



Héritage multiple

- **Une classe peut avoir plusieurs classes parents. On parle alors d'héritage multiple.**
  - Le langage C++ est un des langages objet permettant son implantation effective.
  - Java ne le permet pas.



Construction d'un diagramme de classes

- **Trouver les classes du domaine étudié ;**
  - Souvent, concepts et substantifs du domaine.
- **Trouver les associations entre classes ;**
  - Souvent, verbes mettant en relation plusieurs classes.
- **Trouver les attributs des classes ;**
  - Souvent, substantifs correspondant à un niveau de granularité plus fin que les classes.
  - Les adjectifs et les valeurs correspondent souvent à des valeurs d'attributs.
- **Organiser et simplifier le modèle en utilisant l'héritage ;**
- **Tester les chemins d'accès aux classées ;**
- **Itérer et raffiner le modèle.**

Exercice de Structuration
Diagramme de Classes d'Analyse

Gestion des demandes de formations

1. Le processus de formation est initialisé lorsque le responsable formation reçoit une demande de formation de la part d'un employé.
2. Cette demande est instruite par le responsable qui qualifie la demande et transmet son accord ou son désaccord à l'intéressé.
3. En cas d'accord, le responsable recherche dans le catalogue des formations agréées un stage correspondant à la demande.
4. Il informe l'employé du contenu de la formation et lui propose une liste des prochaines sessions.
5. Lorsque l'employé retourne son choix, le responsable formation inscrit le participant à la session auprès de l'organisme de formation concerné.
6. Le responsable formation contrôle par la suite la facture que lui a adressée l'organisme de formation avant de la transmettre au comptable des achats.

Gestion des demandes de formations

1. **Le processus de formation est initialisé lorsque le responsable formation reçoit une demande de formation de la part d'un employé.**

   ↳ L'article indéfini « un/une »

   Signifie souvent « un en général »

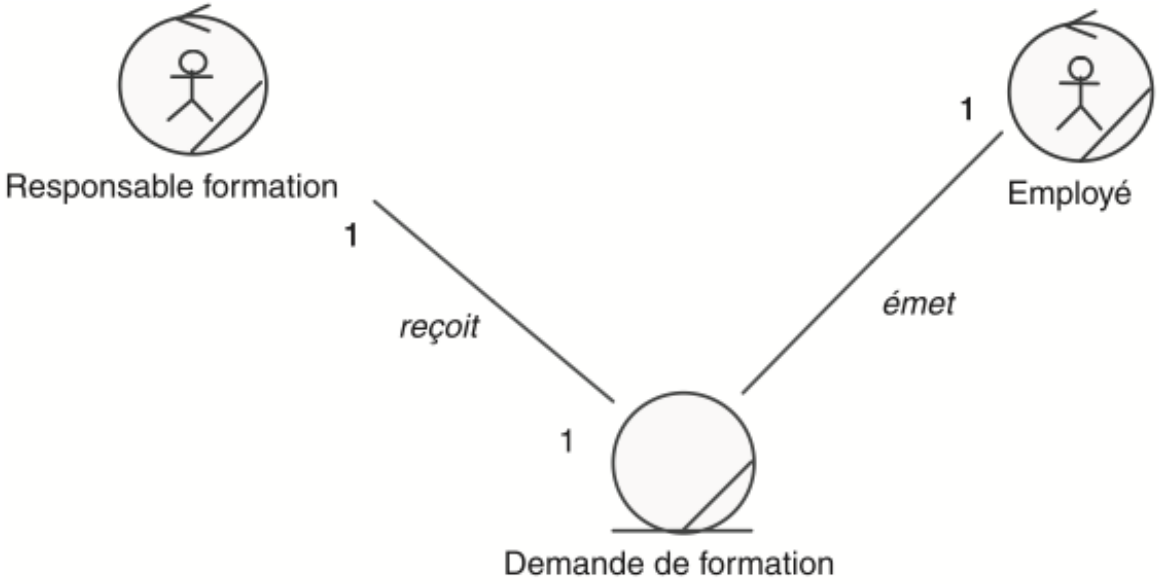
   Mais parfois « un est un seule »

   Et interdit le pluriel

   ↳ L'article défini « le »

   Indique que le nom est

   Unique dans le contexte



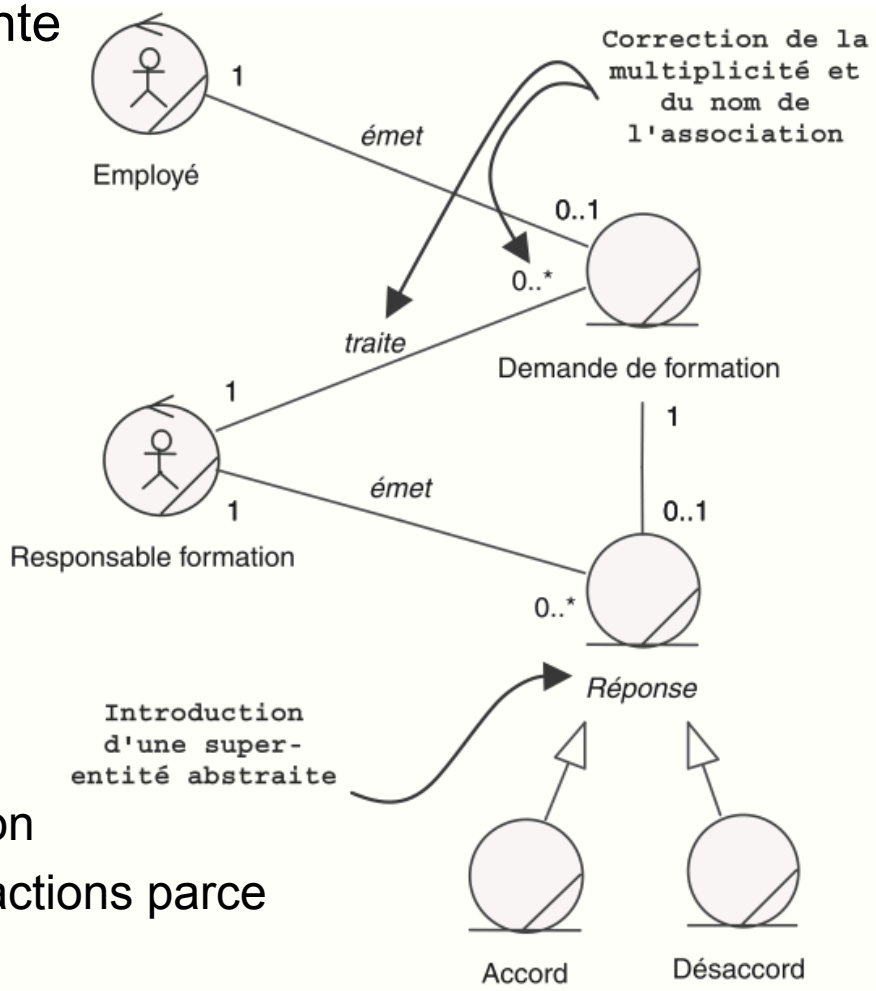
Gestion des demandes de formations

2. Cette demande est instruite par le responsable qui qualifie la demande et transmet son accord ou son désaccord à l'intéressé.

- « Ce » référence à la phrase précédente
- ! Synonymes ! : Responsable = Responsable Formation
- « Son/sa » : association ou attribut
  - Association si 2 concepts
  - Attribut si simple caractéristique
- Conjonction « OU »
  - Association de généralisation/ Spécialisation
  - Type énuméré
- Verbe
  - Un verbe est traduit par une association

Mais on fait pas 3 associations pour les 3 actions parce

On est dans une vue statique



Gestion des demandes de formations

3. En cas d'accord, le responsable recherche dans le catalogue des formations agréées un stage correspondant à la demande.

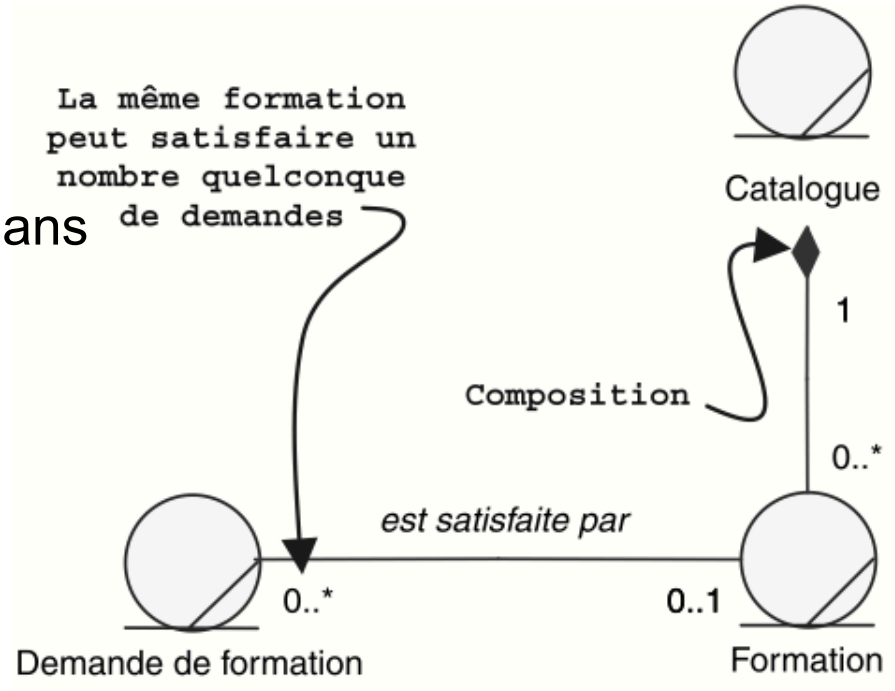
- Relation de type conteneur et

Contenu : Catalogue et Formation

  - Agrégation ou composition
- Pluriel : multiplicité « 0..* »
- Verbe : d'action ne se traduise pas dans

le DCA. C'est pour la vue dynamique

- Adjectif :
  - Soit attribut d'une entité
  - Soit association
  - Soit Bruit
- Participe : souvent une association
- ! Synonyme !
  - Formation = Stage

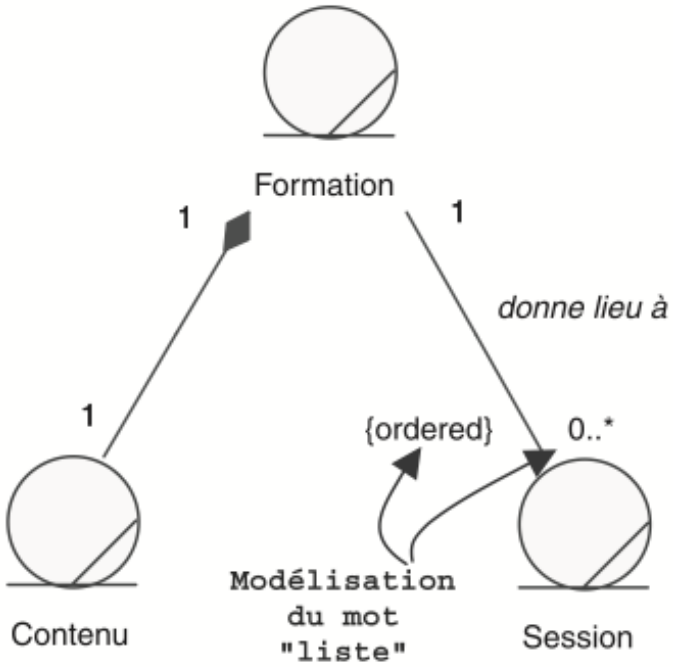


Gestion des demandes de formations

4. Il informe l'employé du contenu de la formation et lui propose une liste des prochaines sessions.

- Il = Responsable
- Contenance ou possession
  - Entité à part ou Attibut
- Conteneur : mot liste
  - multiplicité « * »
  - Souvent contrainte d'ordonnancement
- ! Faux Synonyme!
  - Session != Stage / Formation
- Verbes :
  - Échange de message entre instance et

NON pas une association

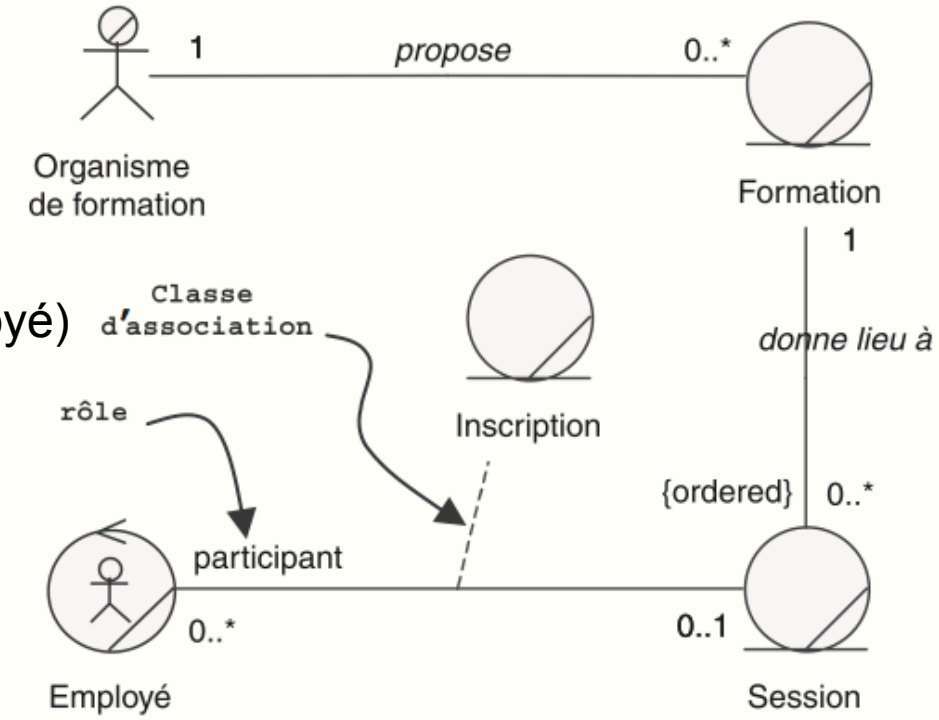


Gestion des demandes de formations

5. Lorsque l'employé retourne son choix, le responsable formation inscrit le participant à la session auprès de l'organisme de formation concerné.

- Verbes :
  - Souvent un verbe cache un nom : Inscrit → Inscription
- Terme Vague
  - Choix : A situer dans le contexte
  - Choix = **une** Session particulière
- Rôles :
  - !! N'est pas automatiquement

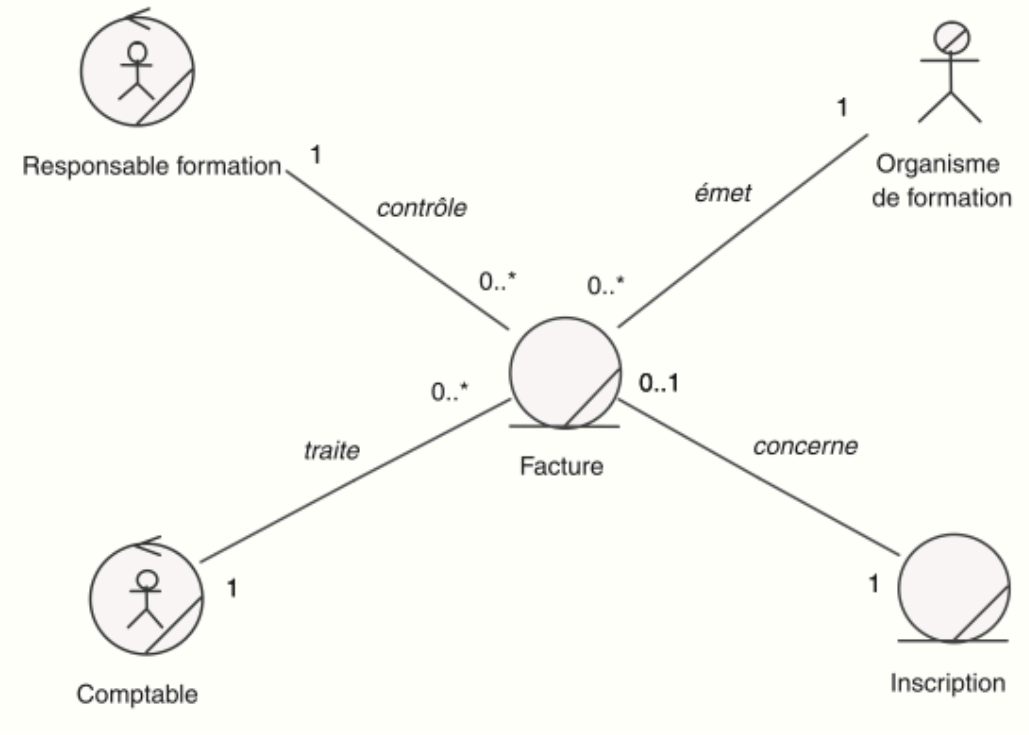
  une nouvelle entité (Participant = Employé)
- Acteurs
  - Organisme en relation avec
    - Session ou Formation ?
  - Formation (voir Phrase 4)

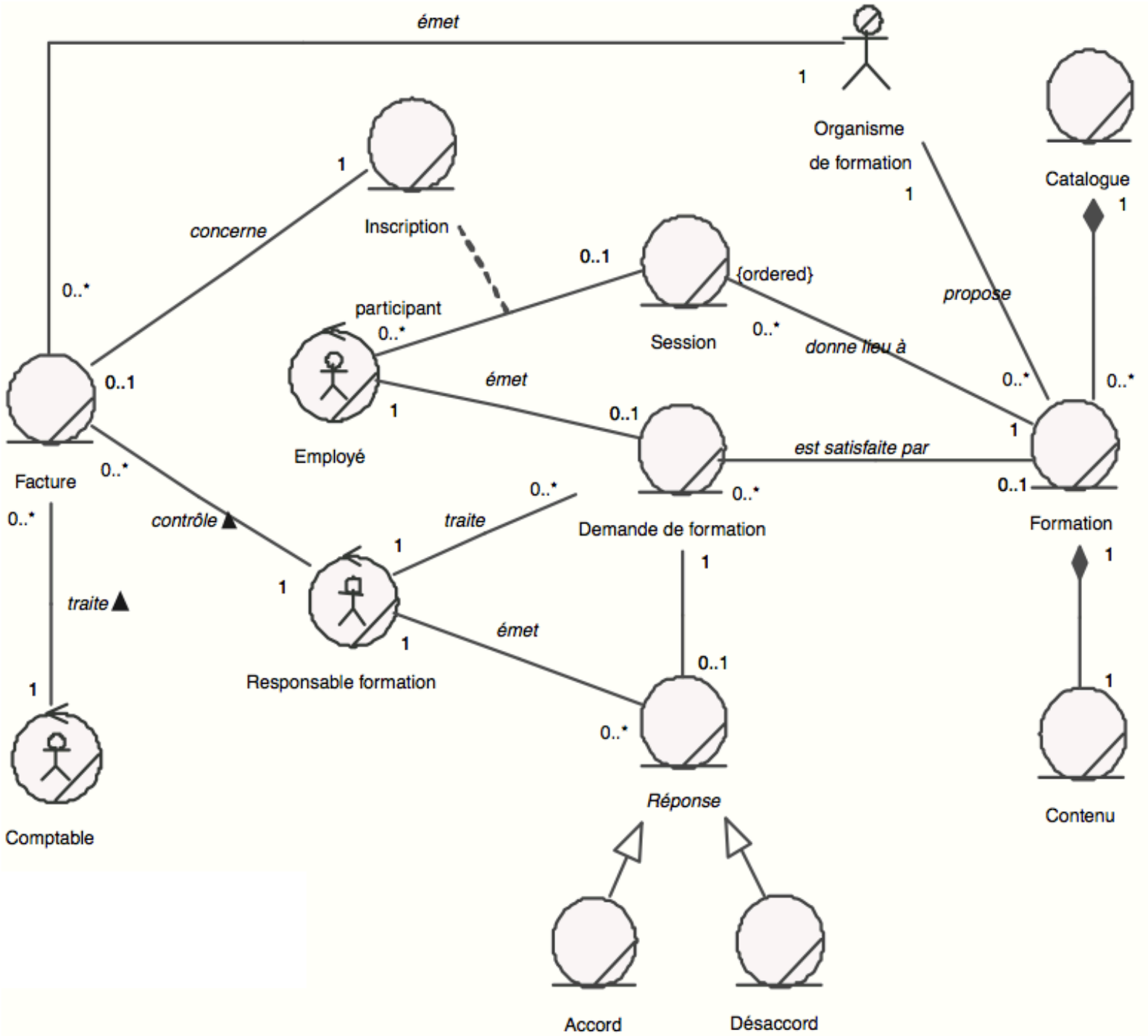


Gestion des demandes de formations

6. Le responsable formation contrôle par la suite la facture que lui a adressée l'organisme de formation avant de la transmettre au comptable des achats.

- Le temps → c'est pour la vu dynamique
- « contrôle par la suite... » ne fait qu'indiquer une succession temporelle de messages. Elle permet implicitement de relier la facture à l'inscription (voir phrase 5).




émet
1
Organisme
de formation
1
Catalogue
1
1
Inscription
concerne
0..*
0..1
participant
0..*
0..1
{ordered}
Session
0..*
propose
donne lieu à
0..*
0..*
émet
1
0..1
Employé
1
Facture
0..*
0..1
est satisfaite par
0..*
0..1
Formation
0..*
contrôle
traite
0..*
Demande de formation
1
1
1
traite
émet
1
1
0..1
Responsable formation
0..*
Réponse
Contenu
1
1
Comptable
Accord
Désaccord

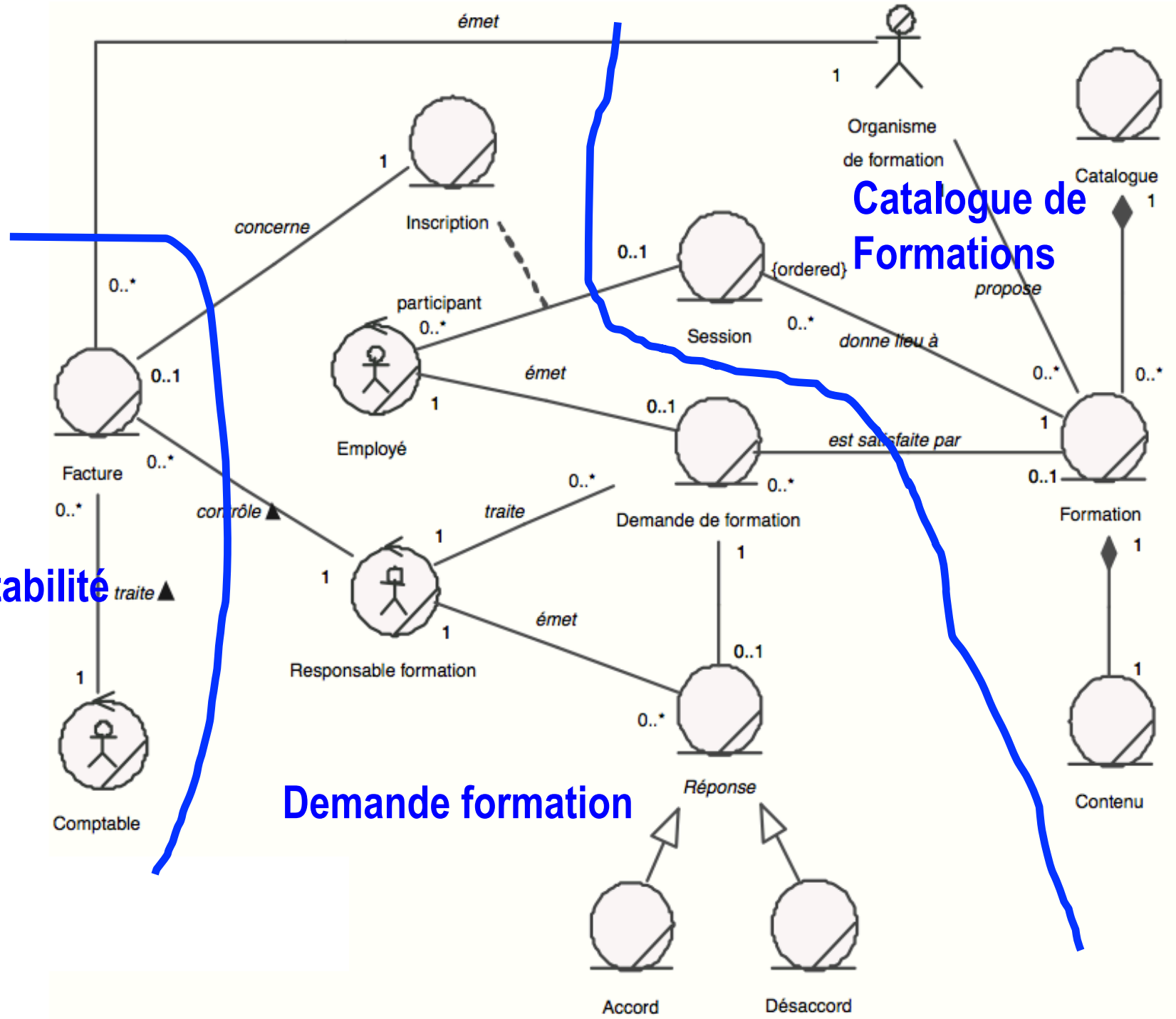

émet
1
Organisme
de formation
Catalogue
1
Catalogue de
Formations
concerne
1
Inscription
0..1
{ordered}
propose
0..*
participant
0..*
Session
0..*
donne lieu à
0..*
0..*
0..1
émet
1
0..1
Employé
est satisfaite par
1
0..1
Facture
0..*
0..*
0..*
traite
Demande de formation
Formation
0..*
contrôle
1
1
traite
1
1
émet
1
0..1
Responsable formation
0..*
Réponse
Contenu
Comptabilité
traite
Comptable
Demande formation
Accord
Désaccord

Diagramme de Package