Associations UML et leur implémentation en Java
2e partie

Université de Montpellier / FDS

Mars 2022

Sommaire

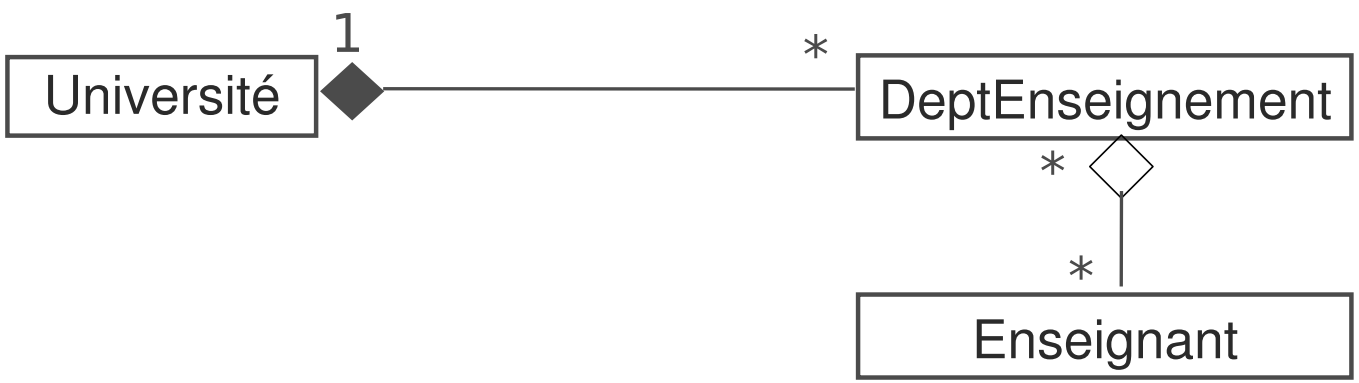
Rappels

- Association : relation entre 2 ou plusieurs classes qui décrit les connexions structurelles entre leurs instances
- Lien : connecte une ou plusieurs instances
- Notions à connaître : rôle, multiplicité, navigabilité
- Traduction en Java : attributs et/ou classe

Sommaire

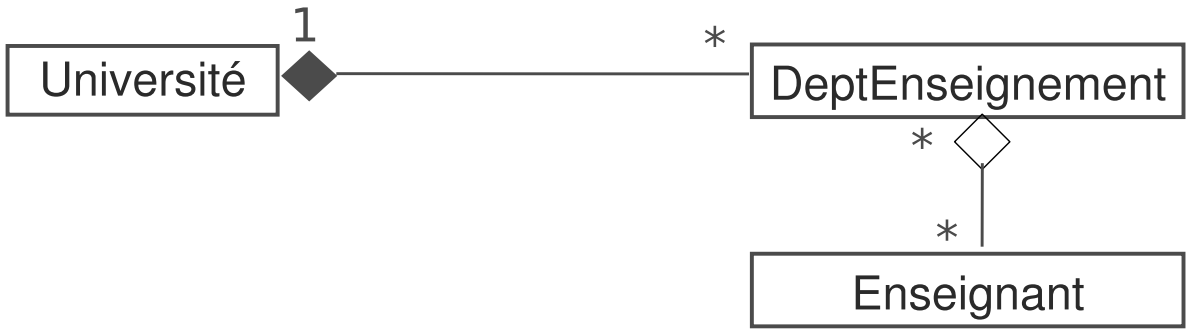
Agrégation et composition

- Une université se compose de départements d'enseignement
- Un département d'enseignement contient des enseignants
- le losange (blanc/vide ou noir/plein) nous épargne de nommer l'association
- Lire le losange : contient, se compose de, a pour éléments, agrège, a pour composants, a pour constituants, etc.



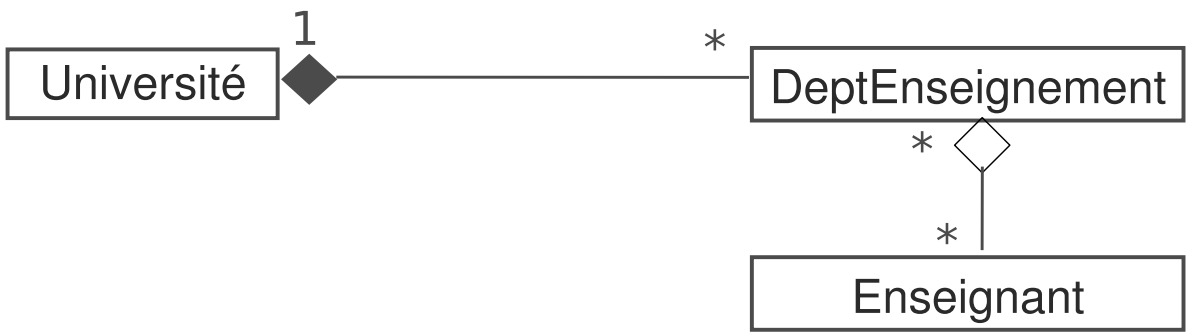
Agrégation

- Un département d'enseignement est un agrégat d'enseignants
- Un enseignant peut être rattaché à plusieurs départements
- Agrégation : relation *agrégat-élément*, dénotée par un **losange non rempli (blanc/vide)** du côté de l'agrégat.



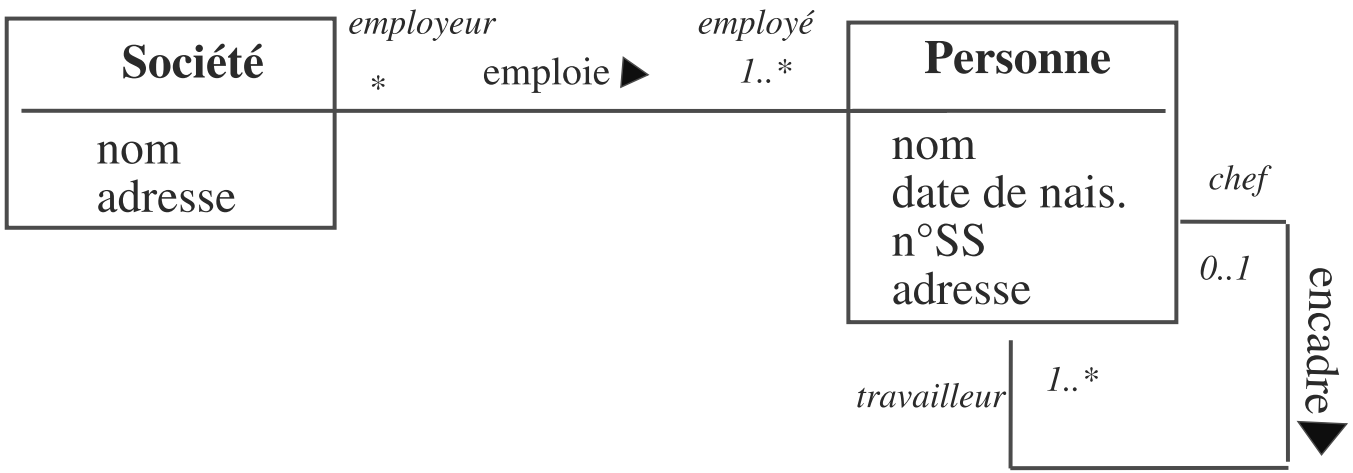
Composition

- Une université se compose de départements d'enseignement
- Le département n'est rattaché qu'à une université
- Le département disparaît si l'université disparaît
- Composition : relation *composite-élément*, on la note avec un **losange plein (noir)** du côté du composite.
- Notion d'exclusivité : un composant ne peut pas être partagé par plusieurs composites.



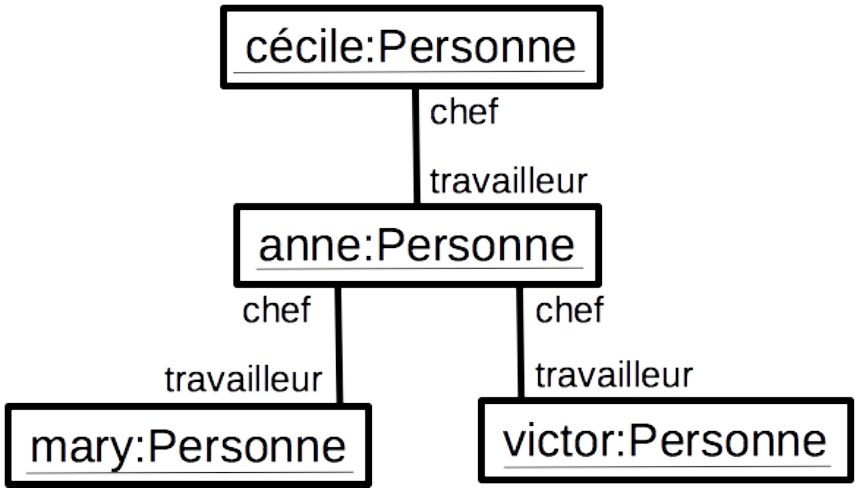
Sommaire

Association interne



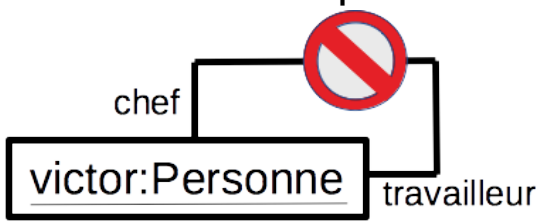
$$\text{encadre} \subseteq \text{Personne} \times \text{Personne}$$

Liens dans une association interne

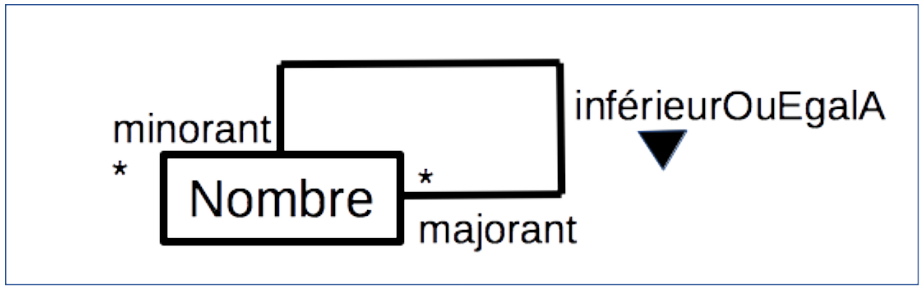


$$\text{encadre} \subseteq \text{Personne} \times \text{Personne}$$

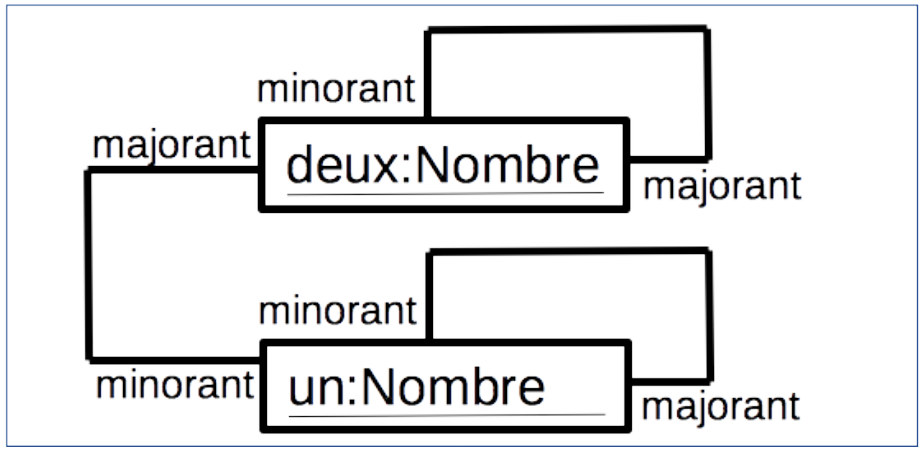
Le lien n'est pas forcément réflexif



Association interne avec liens réflexifs



Classes



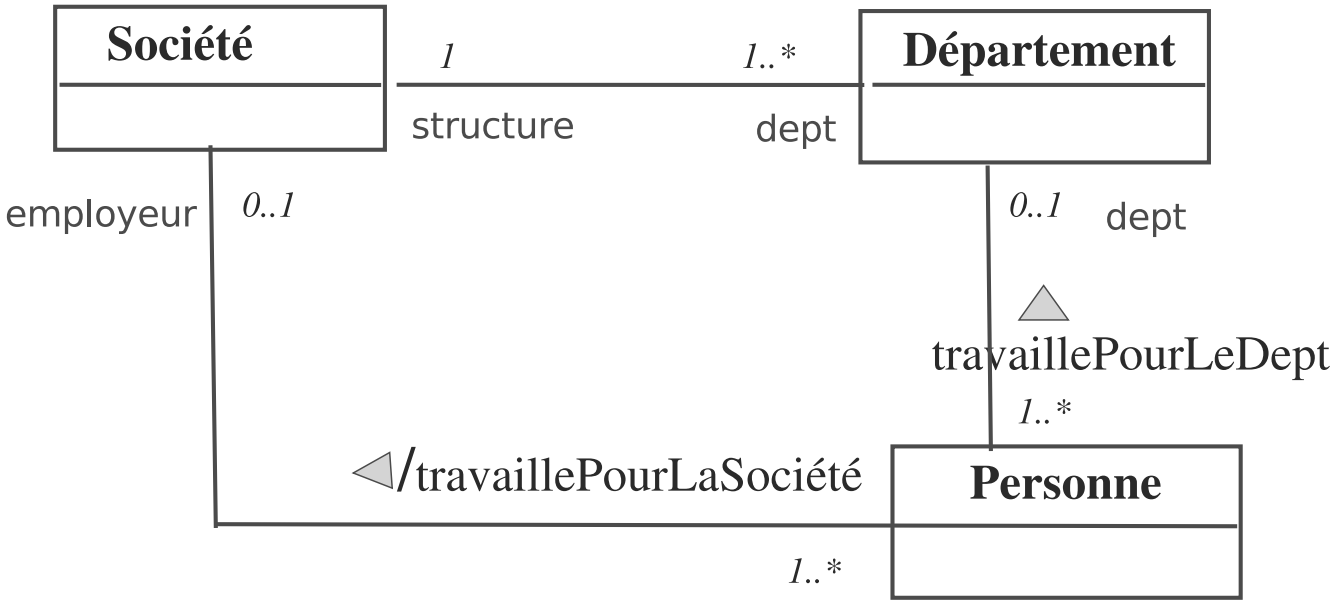
Instances

$$\text{inférieurOuEgalA} \subseteq \text{Nombre} \times \text{Nombre}$$

Sommaire

Association dérivée

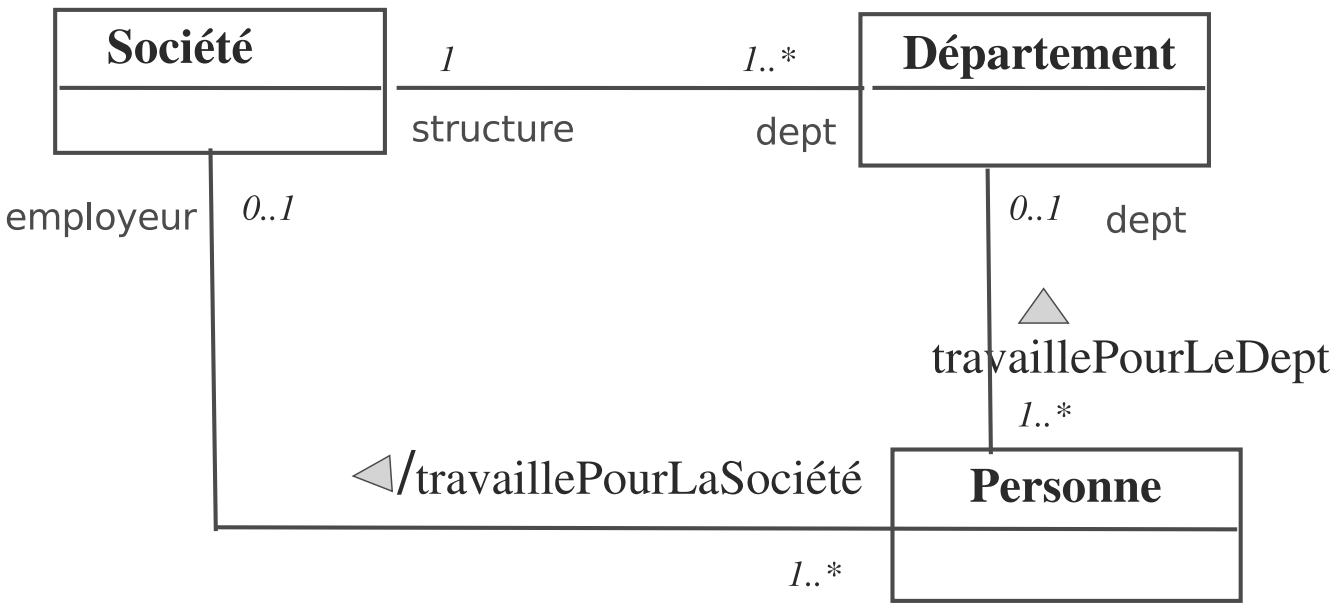
Association construite à partir d'autres associations
indiquée par le symbole /



{Personne.employeur=Personne.dept.structure}

Association dérivée

Une personne travaille pour une société, si elle travaille pour l'un des départements de cette société (la société est la structure accueillant ce département).

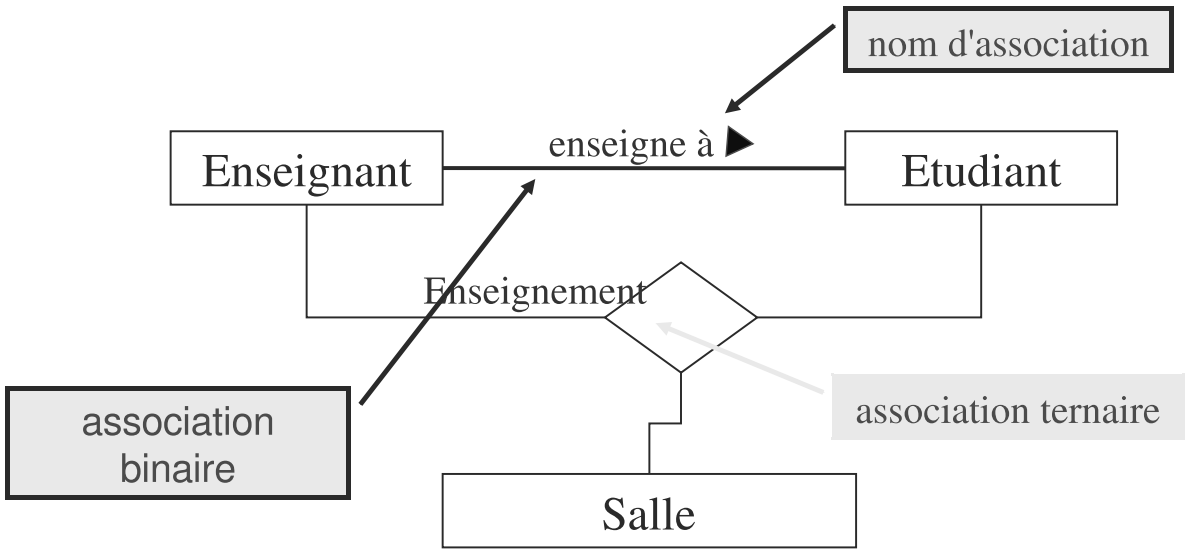


{Personne.employeur=Personne.dept.structure}

Sommaire

Associations n-aires

Les associations peuvent être d'arité 3, 4, ...
L'arité est donnée par le nombre d'ensembles du produit cartésien

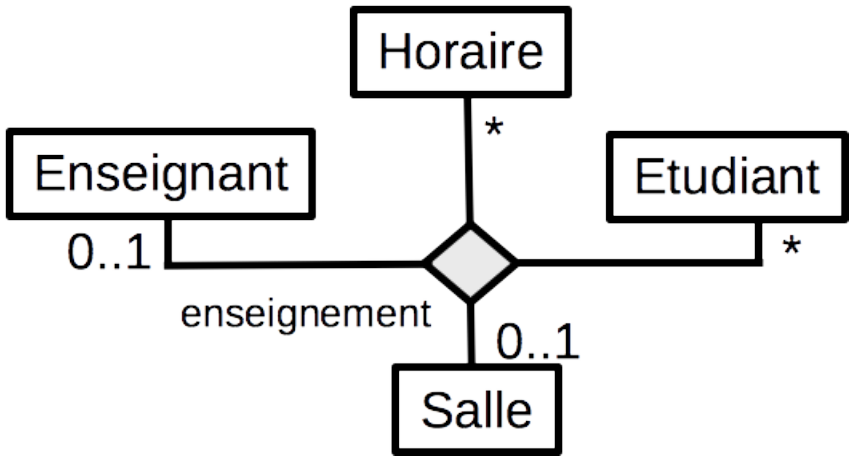


Arité 2 : enseigne-à $\subseteq$ Enseignant $\times$ Etudiant
Arité 3 : Enseignement $\subseteq$ Enseignant $\times$ Etudiant $\times$ Salle

Associations n-aires

Multiplicité d'une relation n-aire (variante de Enseignement)

Arité 4 : enseignement $\subseteq$ Enseignant $\times$ Horaire $\times$ Etudiant $\times$ Salle



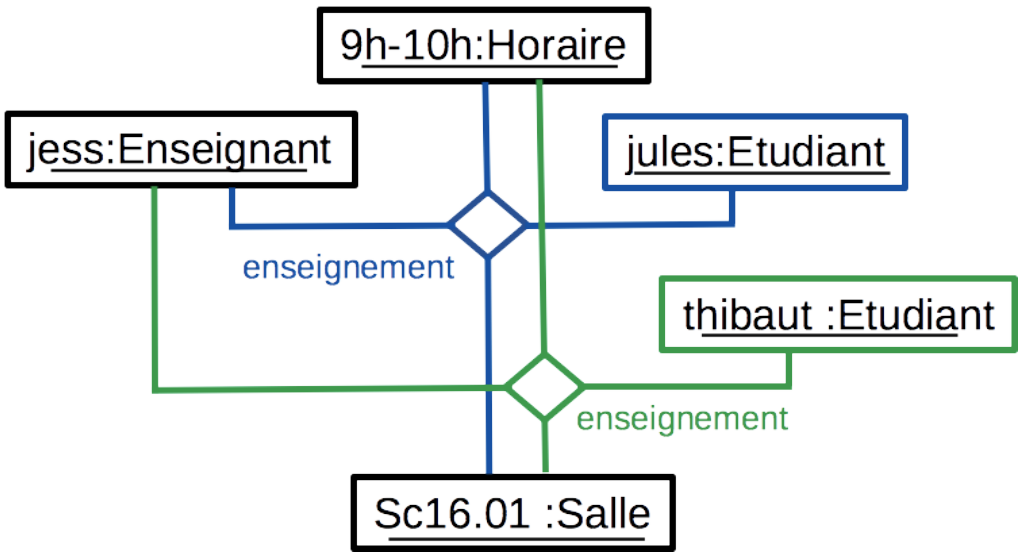
Fixer un tuple de 3 instances et se demander combien d'instances de la classe restante peuvent être associées :

- 1 enseignant, 1 horaire, 1 salle $\rightarrow$ 0 ou plusieurs étudiants (*)
- 1 étudiant, 1 horaire, 1 salle $\rightarrow$ 0 ou 1 enseignant (0..1)
- 1 enseignant, 1 horaire, 1 étudiant $\rightarrow$ 0 ou 1 salle (0..1)
- 1 enseignant, 1 salle, 1 étudiant $\rightarrow$ 0 ou plusieurs horaires (*)

Associations n-aires

Liens d'une relation n-aire

enseignement $\subseteq$ Enseignant $\times$ Horaire $\times$ Etudiant $\times$ Salle



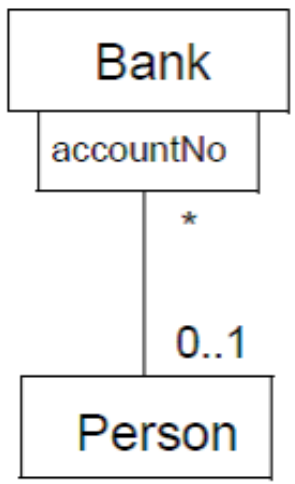
4-Tuples :

- (jess, 9h-10h, jules, Sc16.01)
- (jess, 9h-10h, thibaut, Sc16.01)

Sommaire

Associations qualifiées

- Un qualifieur sur une association permet de sélectionner un sous-ensemble de tuples dans l'association.
- Un qualifieur peut être typé (comme un attribut).
- Il apparaît comme un rectangle collé à une classe.
- Il se compose d'un ou plusieurs attributs.

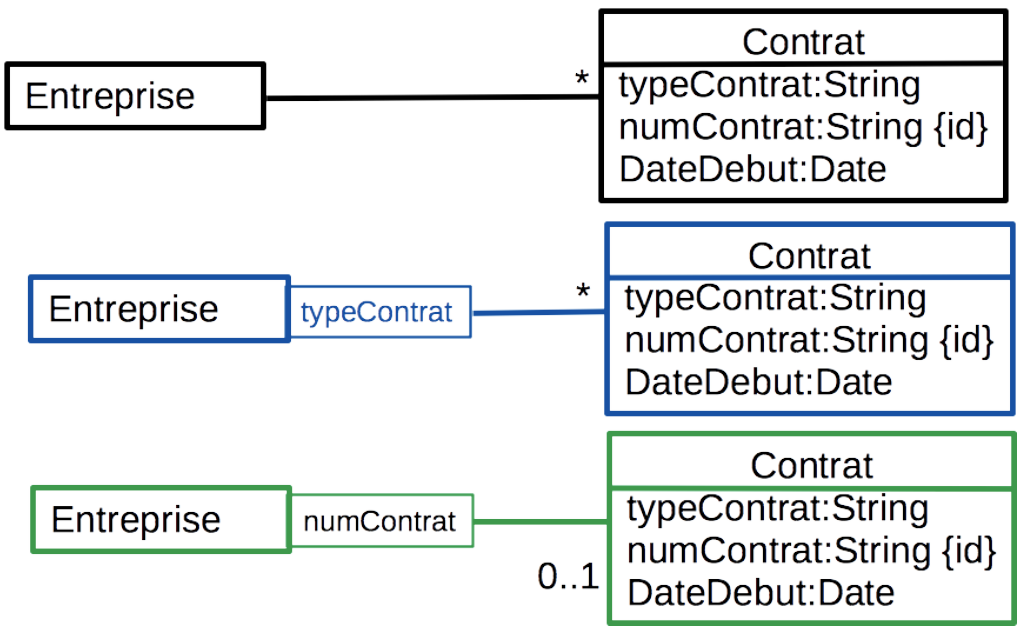


Source : OMG Document Number : ptc/2013-09-05

Normative Reference : http ://www.omg.org/spec/UML/2.5

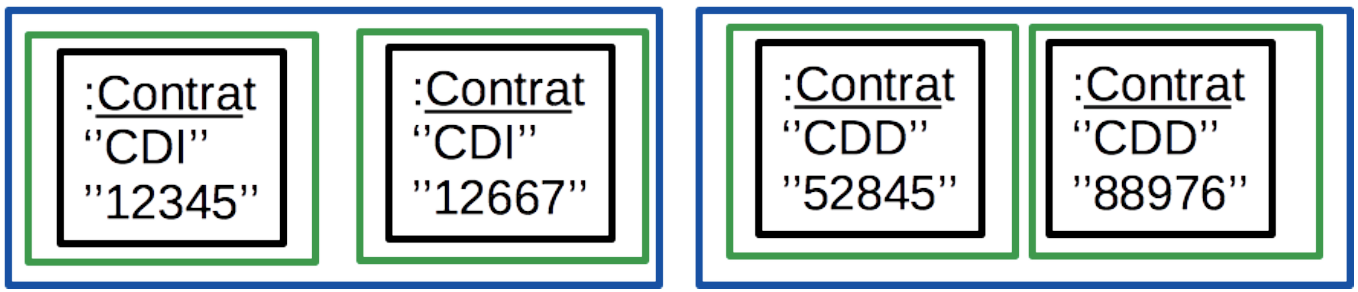
Associations qualifiées

- Déterminer la multiplicité : fixer 1 entreprise et 1 valeur du qualifieur
- 1 entreprise → plusieurs contrats (*)
- 1 entreprise, 1 type de contrat → 0 ou plusieurs contrats (*)
- 1 entreprise, 1 numéro de contrat → 0 ou 1 contrat (0..1)
  `{id}` indique que l'attribut est un identifiant



Associations qualifiées

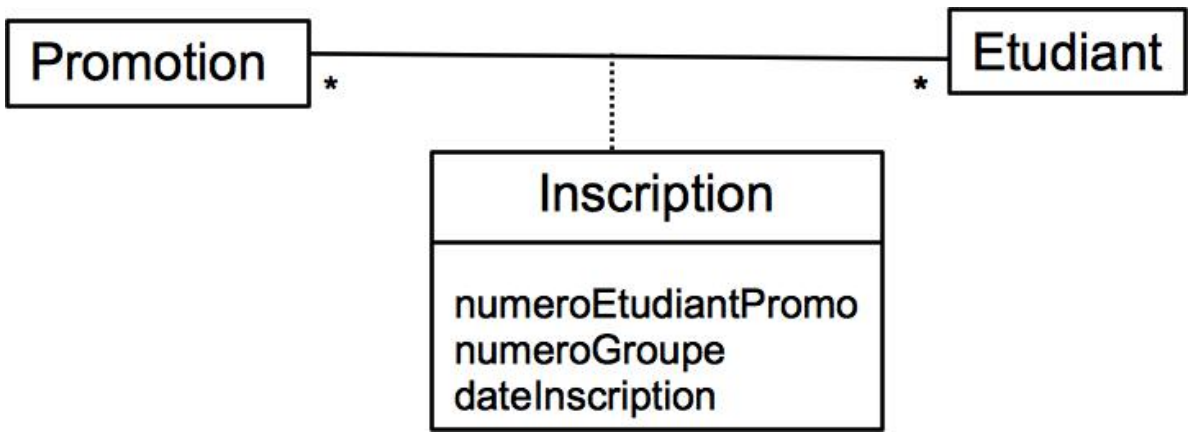
- Chaque valeur de qualifieur détermine un sous-ensemble des instances de contrat
- typeContrat
- numContrat



Sommaire

Classes d'association

- Classe attachée visuellement par un trait en pointillé
- Permet d'ajouter des informations (attributs, opérations) sur une association complexe (de réifier l'association)
- C'est une classe à part entière ⇒ elle peut être liée à d'autres classes



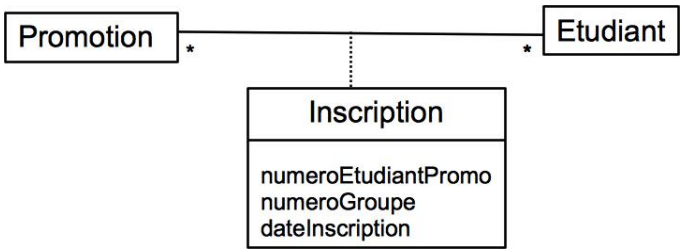
UML vers Java pour les associations complexes

On transforme en général en classes :

- Les classes d'association
- Les associations n-aires, avec n>2

Certaines connexions aux autres classes peuvent se faire sous forme d'attributs, comme vu au cours précédent.

Traduction de la classe-association Inscription

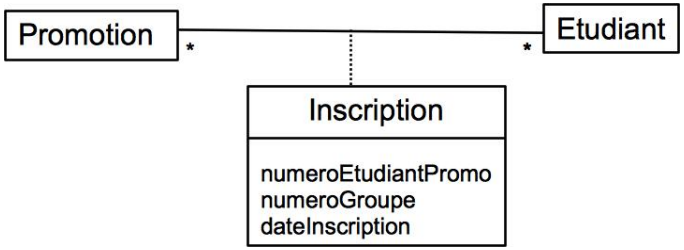



```
public class Inscription{

// Attributs UML
  private String numeroEtudiantPromo ;
  private String numeroGroupe ;
  private LocalDate dateInscription ;

// Roles UML
  private Promotion promotion ;
  private Etudiant etudiant ;
}
```

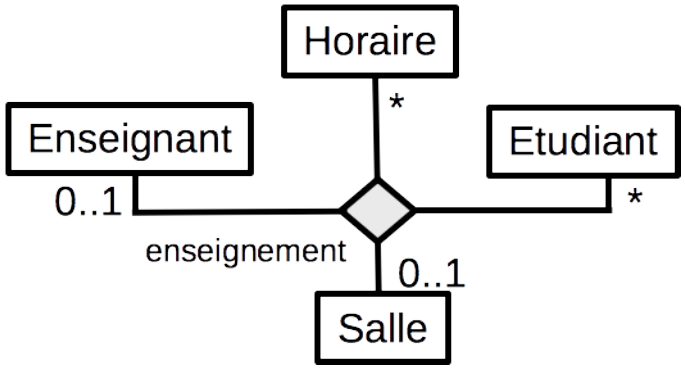
Traduction de la classe-association Inscription




```
public class Promotion{
// Attributs UML
  private int année ;
  ...
// Roles UML
  private ArrayList<Inscription> listeInscriptions ;
}

public class Etudiant{
// Attributs UML
  private String nom ;
  ...
// Roles UML
// Un étudiant peut avoir plusieurs inscriptions
  private ArrayList<Inscription> inscription ;
}
```

Traduction de l'association n-aire




```
public class Enseignement{

// Attributs UML
  private String code ;
  ...

// Roles UML
  private Horaire horaire ;
  private Enseignant enseignant ;
  private Etudiant etudiant ;
  private Salle salle ;
}
// Les 4 classes peuvent avoir un attribut vers les enseignements
// qui les concernent
```

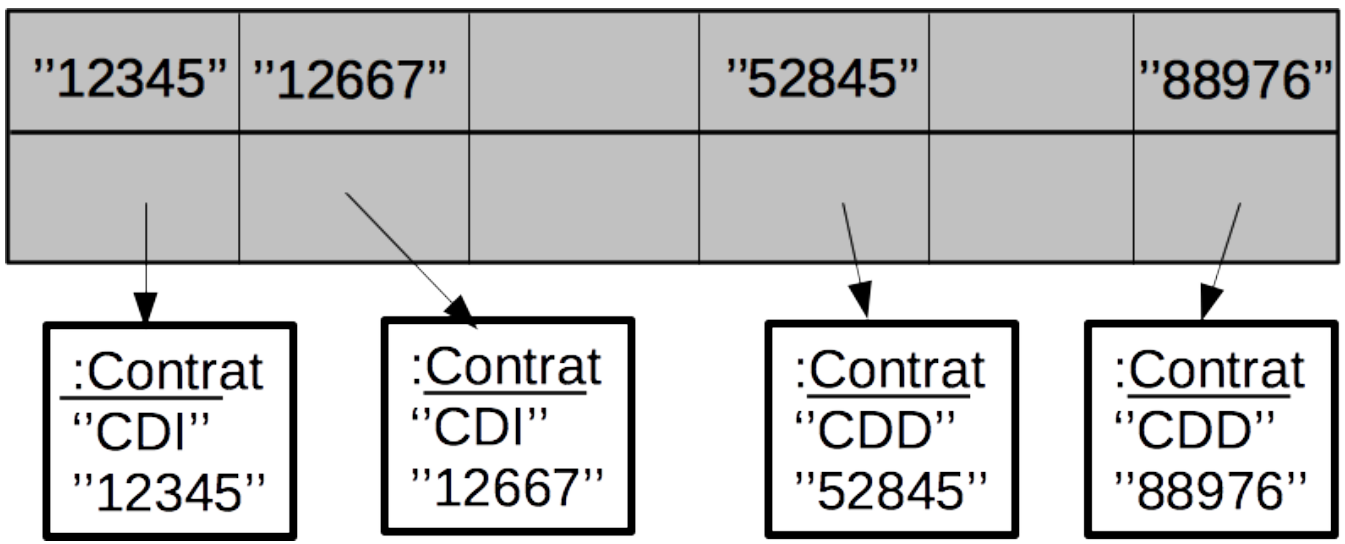
Sommaire

Dictionnaire associatif : principe

- Un dictionnaire associatif est une table indexée par une clef : il associe des clefs (un index) à des valeurs.
- Notamment utilisé pour traduire une association qualifiée.
- Exemple : Un dictionnaire associatif de contrats, indexés par leur numéro de contrat, que l'on sait unique.
  - clef = numéro de contrat, de type String
  - valeur = le contrat qui a ce numéro
- Intérêt : accès très rapide à une valeur à partir de la clef
- Ils s'implémentent souvent avec des tables de hachage ou des arbres de recherche équilibrés ordonnés par les clefs

Dictionnaire associatif : illustration

Vue logique d'un dictionnaire associatif



En Java : `Map`, `HashMap`, `TreeMap`

Une classe pour gérér les dictionnaires associatifs

```
HashMap<TClef, TValeur> table=new HashMap<>();
```

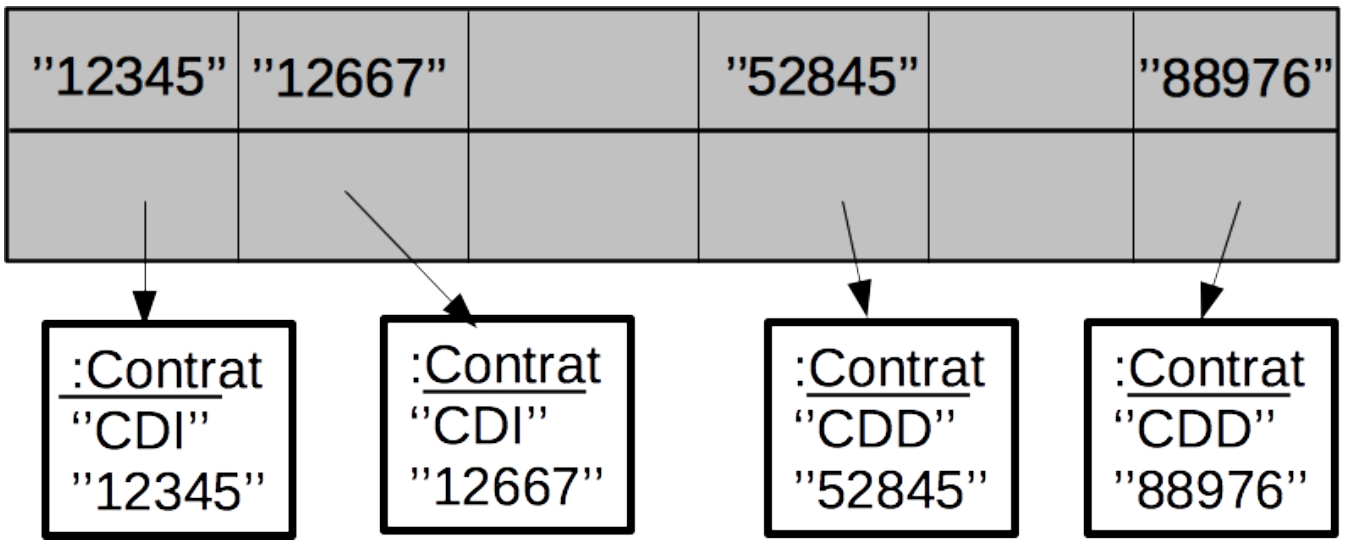
- Map implémentée avec une table de hashage
- Classe générique
  - `TClef` : type des clefs
  - `TValeur` : type des valeurs

Quelques méthodes importantes des `Map`

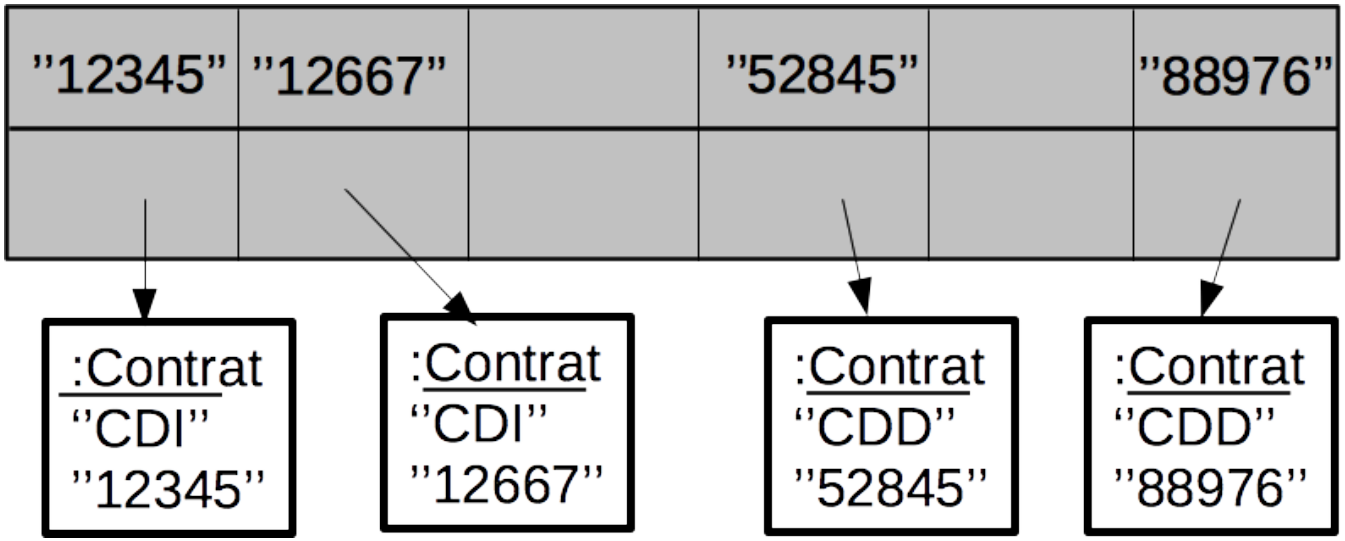
- `Object get(TClef key)` Retourne l'objet de clef `key` ou `null` s'il n'y en a pas.
- `TValeur put(TClef key, TValeur value)` Ajoute l'élément `value` avec comme clef `key`. S'il existait déjà un élément de même clef, cet élément est retourné (et écrasé dans la table par `value`).
- `TValeur remove(Object key)` Retire l'élément de clef `key` de la table.
- `Collection<TValeur> values()` Retourne une collection des valeurs contenues dans la table
- `Set<TClef> keySet()` Retourne une collection des clefs contenues dans la table

Dictionnaire associatif

```
HashMap<String, Contrat> fichierContrats  = new HashMap<>();
fichierContrats.put("12345", new Contrat("12345","CDI"));
fichierContrats.put("12667", new Contrat("12667","CDI"));
fichierContrats.put("52845", new Contrat("52845","CDD"));
fichierContrats.put("88976", new Contrat("88976","CDD"));
```

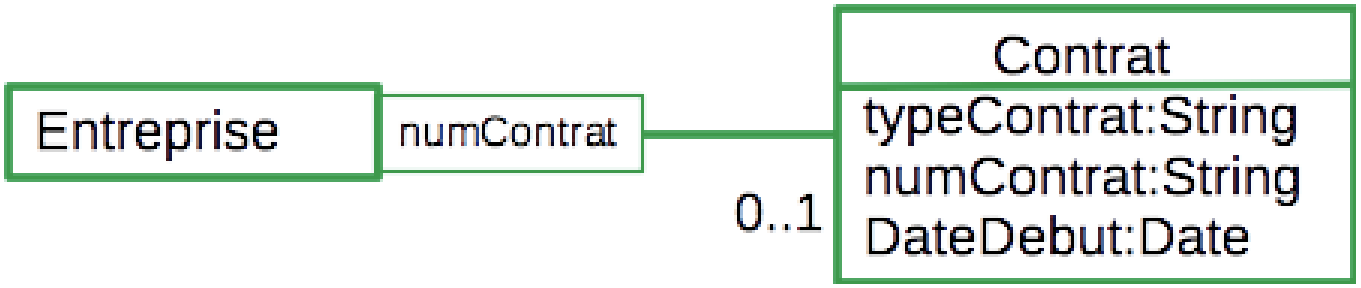


Dictionnaire associatif




```
Contrat contrat = fichierContrats.get("12667");
if (contrat != null) {
   System.out.println("contrat numéro 12667 = " + contrat);
}
```

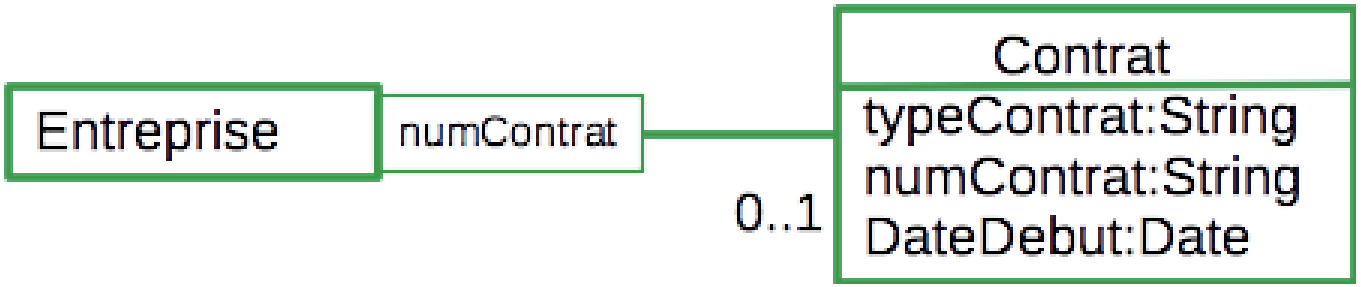
La classe entreprise




```
public class Entreprise{
 private HashMap<String, Contrat> fichierContrats
                                  = new HashMap<>();

 public Entreprise(){}
 //....
```

La classe entreprise




```
public class Entreprise{
 private HashMap<String, Contrat> fichierContrats
                                  = new HashMap<>();
 //...   void
 public ajoute(Contrat c){
    fichierContrats.put(c.getNumContrat(),c);
 }
  public Contrat getContrat(String num){
    return fichierContrats.get(num);
 }
  public boolean contientContrat(String num){
    return fichierContrats.containsKey(num);
 }
```

Parcourir un dictionnaire associatif

```
public class Entreprise{

 // en parcourant ses valeurs, avec la méthode values()

 public ArrayList<Contrat> contratsSupSMIC_v1(){
    ArrayList<Contrat> resultat = new ArrayList<>();
    for (Contrat c : fichierContrats.values())
         if (c.getSalaire() >= ValeurSMIC)
              resultat.add(c);
    return resultat;
 }
```

Parcourir un dictionnaire associatif

```
public class Entreprise{

  // en parcourant ses clefs, avec la méthode keySet()

 public ArrayList<Contrat> contratsSupSMIC_v2(){
   ArrayList<Contrat> resultat = new ArrayList<>();
   for (String num : fichierContrats.keySet())
      if (fichierContrats.get(num).getSalaire() >= ValeurSMIC)
           resultat.add(fichierContrats.get(num));
   return resultat;
 }
```

Synthèse

- Association, agrégation, composition, binaire, n-aire
- Associations internes, dérivées, qualifiées
- Classes d'association
- Traduction par des attributs et/ou une classe
- Multiplicités > 1 : usage de liste ou de dictionnaire associatif