

C.16

## Finaliser le diagramme de classes de conception

### Objectif

Tenir compte des principes OO et UML.

### Préambule

Le **diagramme de classes** de conception est un document indispensable qui représente la **vue de conception statique** d'un système.

Le **diagramme d'objets** représente la **vue de conception** ou la **vue de processus** statiques d'un système. Le diagramme d'objets peut être efficace pour donner un exemple ou d'affiner un aspect délicat d'un diagramme de classes.

### Méthode

#### Conventions de nommage

Les noms d'attributs et d'opérations (et de relations) commencent toujours par une **minuscule** et peuvent contenir ensuite plusieurs mots concaténés par une **Majuscule**.

Les noms des classes commencent systématiquement par une **Majuscule**. Ensuite, on applique la même convention.

Il est conseillé de ne pas utiliser d'accents ni de caractères spéciaux !

#### Objet ou attribut ?

Un objet est un élément plus « important » qu'un attribut. Un bon critère à appliquer :

- si l'on peut demander à un élément que sa valeur, il s'agit d'un simple attribut
- si l'on peut lui poser plusieurs questions, il s'agit plutôt d'un objet qui possède à son tour plusieurs attributs.

#### Héritage

La super-classe n'est pas toujours abstraite (voir polymorphisme).

#### Agrégation ou composition ?

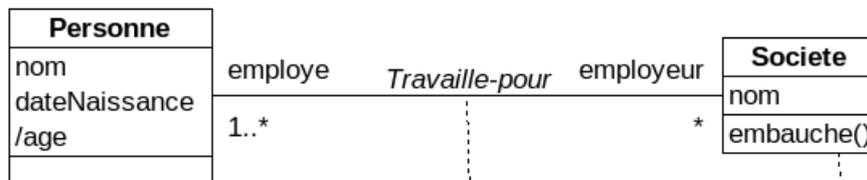
Une agrégation est un cas particulier d'association non symétrique exprimant une relation de contenance. Elle n'a pas besoin d'être nommée car elle signifie implicitement « contient » ou « est composée de ».

Une composition est une agrégation plus forte impliquant :

- une partie ne peut appartenir qu'à un seul composite (agrégation non partagée)
- la destruction du composite entraîne la destruction de toutes ses parties (responsable du cycle de vie de ses parties).

Pour qu'une agrégation soit une composition, il faut donc vérifier les deux critères suivants :

- la multiplicité ne doit pas être supérieure à 1 du côté composite
- le cycle de vie des parties doit dépendre de celui du composite (et notamment pour la destruction).



### Multiplicité

Aux deux extrémités d'une association, on doit faire figurer une indication de multiplicité. Elle spécifie le nombre d'objets qui peuvent participer à une relation avec un autre objet.

Une Societe peut employer 1 ou plusieurs Personne et 1 Personne peut Travailler-pour plusieurs Societe.

Relation entre classes qui précise que les objets d'une classe sont reliés aux objets d'une autre classe  
Par exemple, une Personne Travaille-pour une Société.

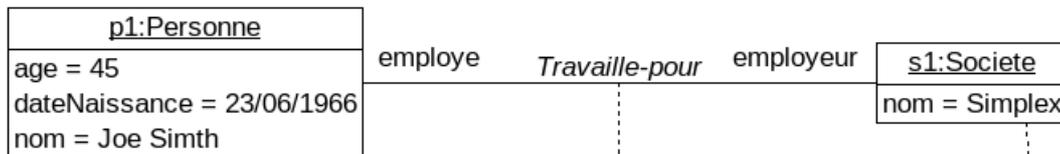
Quand une classe participe à une association, elle y joue un rôle spécifique et on peut le nommer explicitement.  
Par exemple, une Personne, qui joue le rôle d'employé, est associé à une Société qui tient le rôle d'employeur.

Un attribut représente un type d'information contenu dans une classe.

Un attribut dérivé (noté /) est un attribut dont la valeur peut être déduite d'autres informations disponibles dans le modèle.

Une opération représente un élément de comportement (un service) contenu dans une classe. On ajoute les opérations en phase de conception (attribution des responsabilités aux objets).  
La même opération peut s'appliquer à des classes différentes. On dit qu'elle est polymorphe car elle peut prendre différentes formes.  
Une méthode est l'implémentation d'une opération pour une classe.

Diagramme de classes



Un lien est une connexion entre instances d'objets. Il y a donc une relation d'association entre les classes Personne et Societe.  
Par exemple, la personne Joe Smith Travaille-pour la société Simplex.

Chaque attribut a une valeur pour chaque instance d'objet.

p1 et p2 sont deux instances de la classe Personne et possède leurs propres valeurs aux attributs communs de cette classe.

Un objet est une entité possédant une identité et encapsulant un état et un comportement. Un objet est une instance d'une classe.

s1 est une instance de la classe Societe ou s1 est un objet de type Societe.

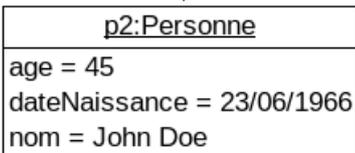
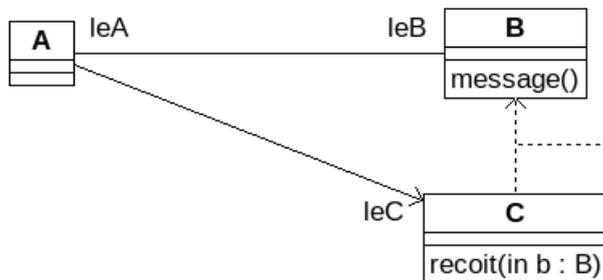
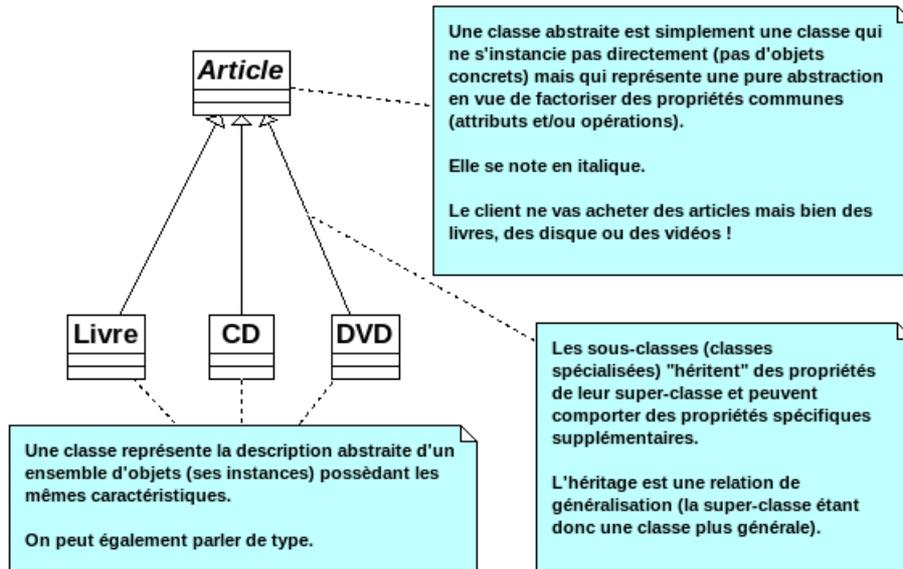


Diagramme d'objets



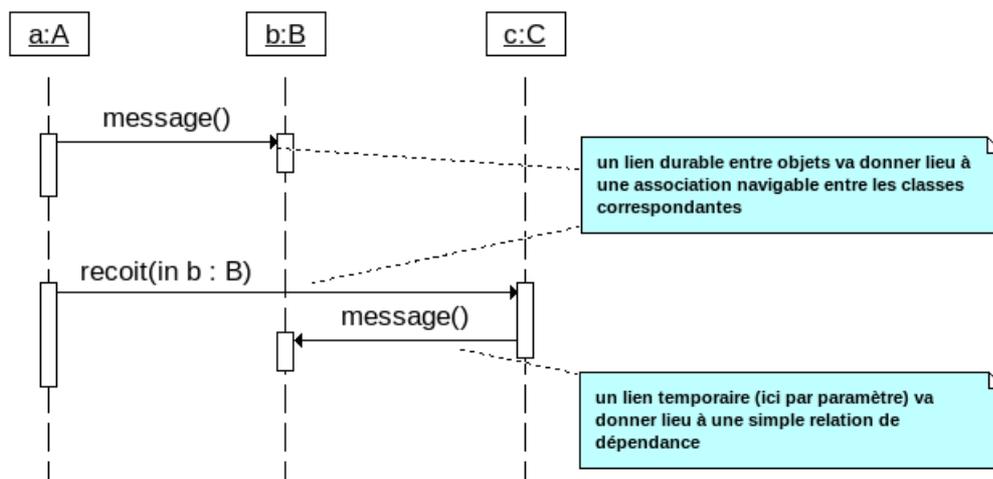
Une dépendance est une relation d'utilisation. Ici, la classe C utilise la classe B.  
La plupart du temps, les dépendances servent à montrer qu'une classe en utilise une autre comme argument (paramètre) dans la signature d'une opération. On parle aussi de lien temporaire. Cela peut aussi être le cas d'une variable locale dans une opération.

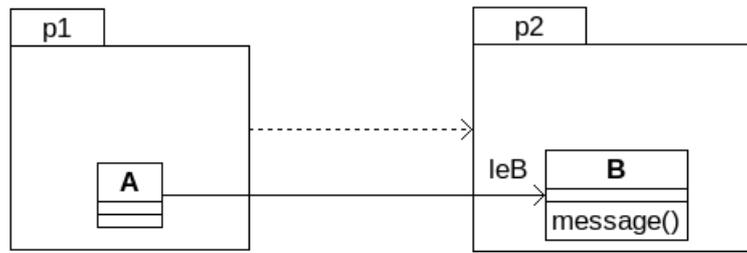
**Lien durable**

Un lien durable entre objets va donner lieu à une **association** entre les classes qui correspondent.

**Lien temporaire**

Un lien temporaire va donner lieu à une simple relation de **dépendance**. Une dépendance se caractérise par le passage en paramètre d'une méthode ou par une instance (variable) locale d'un objet.



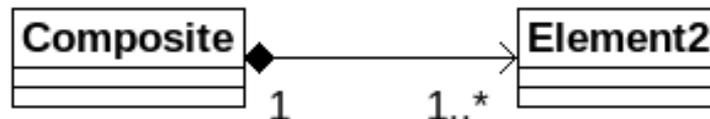
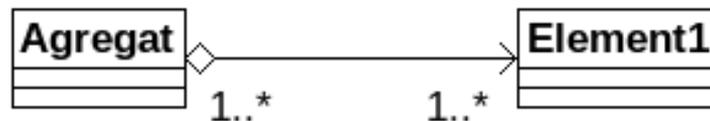


### Navigabilité et dépendance

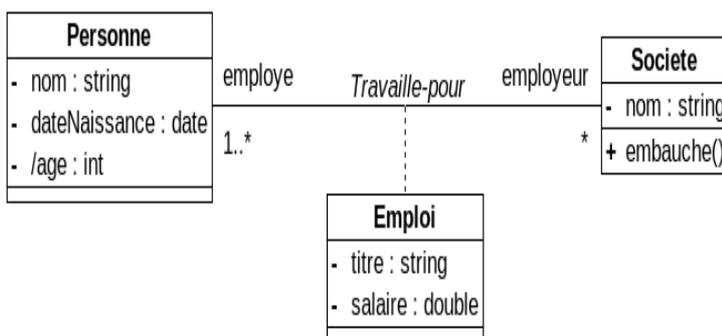
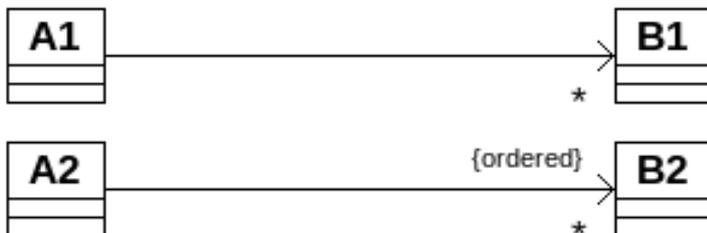
Une association entre deux classes est par défaut bi-directionnelle ce qui permet de naviguer dans les deux sens entre des objets de la classe A et des objets de la classe B.

Il est possible de limiter cette navigabilité à une seule des deux directions en ajoutant une flèche indiquant le seul sens possible. Seuls les objets de la classe A « connaîtront » les objets de la classe B.

Il y a alors une dépendance unique entre les deux packages p1 et p2.



### Exemples d'agrégation et de composition



### Propriétés

UML 2 définit 4 propriétés qui se rapportent à une des extrémités d'une association. Elles sont représentées avec la notation des contraintes {...}.

La propriété **ordered** indique que l'ensemble des objets est classé selon un ordre explicite (ordonné). C'est bien une contrainte pour le choix du conteneur à implémenter.

Les autres propriétés sont : **changeable** (ajout, suppression et modification autorisés), **addOnly** et **frozen** (modification et suppression interdites).

### Classe d'association

Il s'agit d'une association promue au rang de classe. Ce concept avancé d'UML n'existe pas dans les langages de programmation objet.