UML: Diagramme des classes

I. **Représentation des classes.**
II. **Relations entre classes.**
III. **Méthodologie.**

I . Représentation des classes

1. But du diagramme de classes.
2. Objets et classes d'objets.
3. Visibilité des membres.
4. Attributs.
5. Méthodes.
6. Classes abstraites.

I . Représentation des classes

1. But du diagramme de classes

- Le diagramme des classes fait partie des diagrammes UML permettant la modélisation statique des systèmes.
- Le diagramme de classes fournit une description de la structure du système.
- Il fournit une vue statique du système, en représentant les éléments (classes) du système et leurs relations.
- Il permet également une délimitation du système.

I . Représentation des classes

2. Objets et classes d'objets

Objet : entité identifiable du monde réel pouvant avoir ou pas une existence physique. Exemples : chat, table, courant de pensée…

Un objet possède **trois composantes** :

1. Une identité.
2. Des variables définissant sont état (**attributs**).
3. Des sous programmes gérant son comportement (**méthodes**).

I . Représentation des classes

2. Objets et classes d'objets

Classe : abstraction regroupant des objets ayant les mêmes attributs et les mêmes méthodes.

Un objet est alors une **instance** de la classe correspondante, et se distingue des autres instances par son identité et la valeur de ses attributs.

I . Représentation des classes

2. Objets et classes d'objets

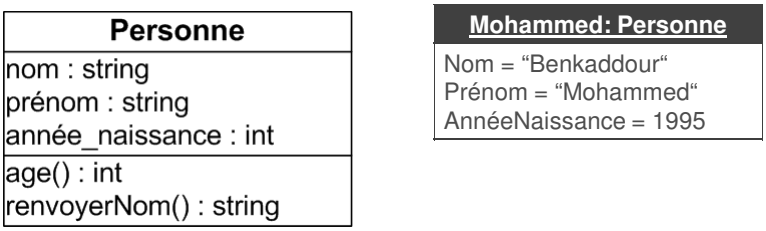
Représentation d'une classe :

Nom de la classe
déclaration des attributs ...
signature des méthodes() ...()

I . Représentation des classes

2. Objets et classes d'objets

Exemple d'une classe et d'une de ses instances



I . Représentation des classes

3. Visibilité des membres

Visibilité des attributs et méthodes :

Un attribut ou une méthode sont dits **privés** si leur utilisation est interdite en dehors de la classe.

Un attribut ou une méthode sont dits **publics** si leur utilisation est autorisée en dehors de la classe.

Un attribut ou une méthode sont dits **protégés** si leur utilisation est limitée à la classe et ses descendantes.

I . Représentation des classes

3. Visibilité des membres

Représentation de la visibilité :

- Public : +
- Privé : -
- Protégé : #

Exemple :

Personne
-nom : string #prénom : string -année_naissance : int
+age() : int +renvoyerNom() : string

I . Représentation des classes

4. Attributs

Possibilité d'initialisation des attributs

Lors de la déclaration d'un attribut on peut lui attribuer une valeur par défaut.

Exemple :

Salarié
-Nom: CHAINE # Prénom : CHAINE -année_naissance : ENTIER -Salaire: REEL= 8000,00
+ age() : ENTIER +renvoyerNom(): CHAINE

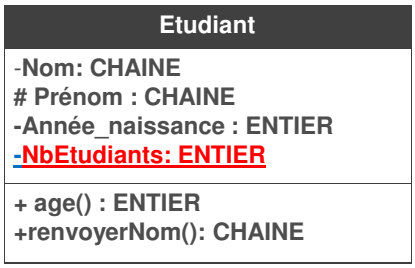
I . Représentation des classes

4. Attributs

Attributs de classes : ce sont des attributs particuliers qui ont la même valeur pour toutes les instances de la classe. L'exemple fondamental est un attribut qui compte le nombre d'objets instanciés de la classe.

Représentation : ils sont soulignés.

Exemple :



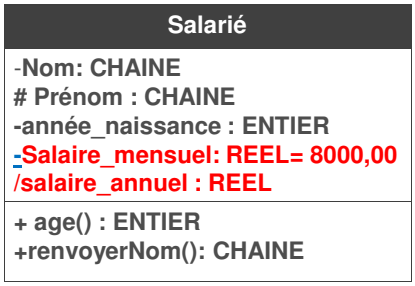
I . Représentation des classes

4. Attributs

Attributs dérivés : attributs dont la valeur peut être calculée à partir d'autres attributs et de formules de calcul.

Représentation : avec un « / » devant le nom.

Exemple :



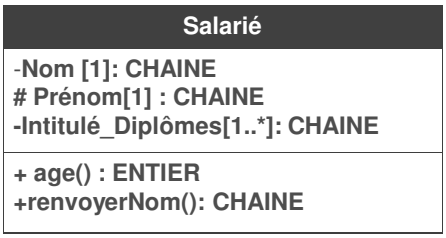
I . Représentation des classes

4. Attributs

Multiplicité : il est possible d'indiquer la multiplicité d'un attribut, c'est à dire le nombre de valeurs que la variable peut stocker.

Représentation : entre [ ].

Exemple :



I . Représentation des classes

6. Classes abstraites

Classe abstraite : classe qui ne peut être instanciée, car elle contient des méthodes abstraites, c'est à dire des méthodes non implémentées.

Une classe abstraite sert essentiellement à factoriser des méthodes et attributs communs à plusieurs classes, et ce dans une relation d'héritage.

I . Représentation des classes

6. Classes abstraites

Représentation : le nom de la classe est en italique, et on fait précéder les méthodes abstraites par le stéréotype « abstract ».

Exemple :

Figure Plane
-nom : string -épaisseur_trait : double -remplissage : bool
«abstract» +Dessiner()

I . Représentation des classes

6. Classes abstraites

Interface : il s'agit d'une classe totalement abstraite, c'est à dire d'une classe sans attributs qui ne contient que des méthodes abstraites.

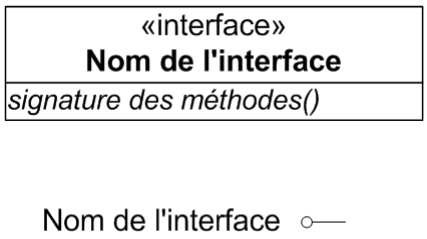
Son rôle est de regrouper un ensemble cohérent d'opérations.

On utilise des interfaces pour classer les opérations en catégories sans se soucier de leurs implémentations.

I . Représentation des classes

6. Classes abstraites

Représentations d'une interface



I . Représentation des classes

Vos Questions

UML : Diagramme de classes.

Ⅱ. Relations entre classes

Ⅱ. Relations entre classes

1. Association.
2. Agrégation.
3. Composition.
4. Dépendance.
5. Héritage.
6. Interfaces.

Ⅱ. Relations entre classes

Introduction

- Après avoir établi quelles classes participaient au système, il convient d'étudier leurs relations.
- Ces relations expriment des **liens structurels ou sémantiques**.
- Dans cette partie nous utiliserons souvent une représentation non documentée des classes (c'est à dire sans préciser les attributs et les méthodes) afin d'alléger les diagrammes.

Ⅱ. Relations entre classes

1. Association

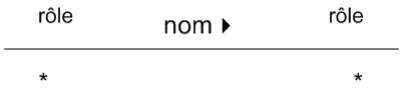
Association : relation sémantique durable entre deux classes.

Exemple : une personne peut travailler pour une entreprise. La relation « **travaille pour** » est une association entre les classes « **personne** » et « **entreprise** ».

II. Relations entre classes

1. Association

Représentation :

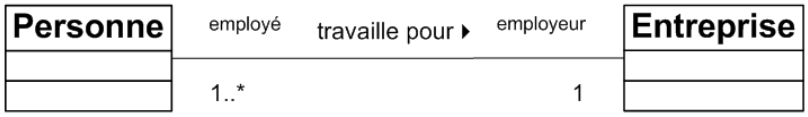


- « nom » est souvent un verbe complété par un sens de lecture.
- « rôle » indique les fonctions de chaque classe dans l'association.
- Les multiplicités précisent les nombres d'objets de chaque classe qui interviennent dans l'association.

II. Relations entre classes

1. Association

Exemple :



Ⅱ. Relations entre classes

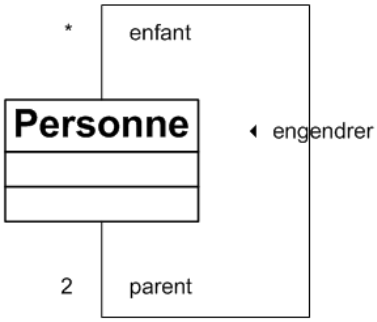
1. Association

Une association entre deux classes peut être multiple :

Ⅱ. Relations entre classes

1. Association

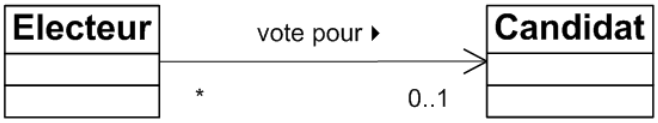
Une association peut être réflexive :



II. Relations entre classes

1. Association

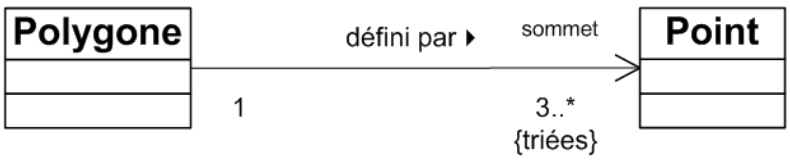
Association à navigabilité restreinte : par défaut une association est navigable dans les deux sens. Mais on peut aussi indiquer que les instances d'une classe ne connaissent pas les instances d'une autre.



II. Relations entre classes

1. Association

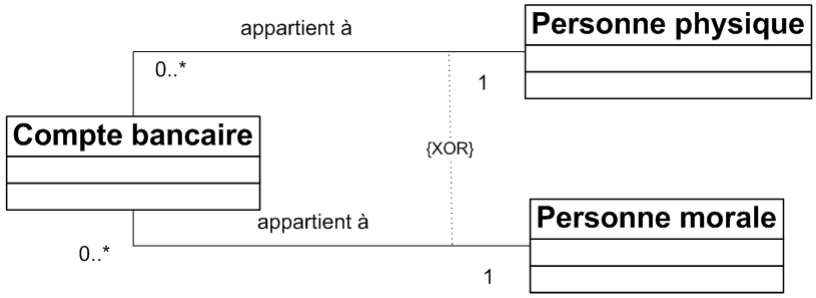
Association avec contraintes : permet de mieux préciser le sens et la portée de l'association.



Ⅱ. Relations entre classes

1. Association

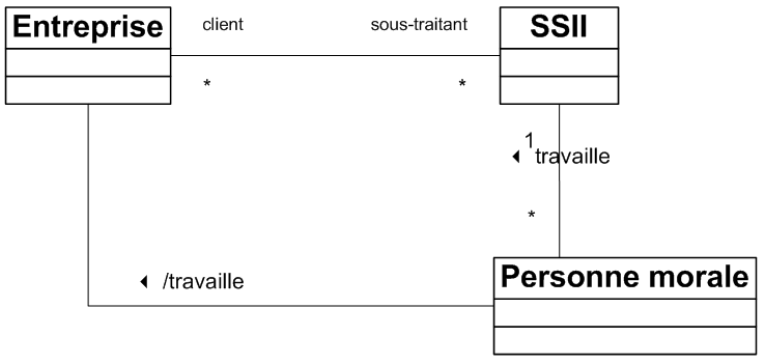
Contrainte entre associations : permet par exemple de faire un choix entre deux associations possibles.



Ⅱ. Relations entre classes

1. Association

Association dérivée : association redondante que l'on peut déduire d'autres associations. Elle permet de faciliter la compréhension de la navigabilité entre classes.



Ⅱ. Relations entre classes

1. Association

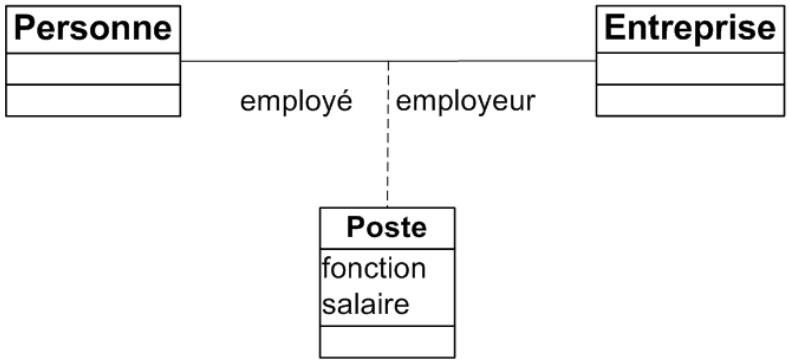
Classe d'association

- Certaines associations nécessitent de stocker des informations complémentaires qui, d'un point de vue conceptuel, ne peuvent faire partie d'aucune des classes de la relation.
- On crée alors une nouvelle classe dite classe d'association, destinée à contenir ces informations.

Ⅱ. Relations entre classes

1. Association

Exemple :



Ⅱ. Relations entre classes

1. Association

Qualification d'une association : renseigne sur l'implication effective des classes dans la relation.

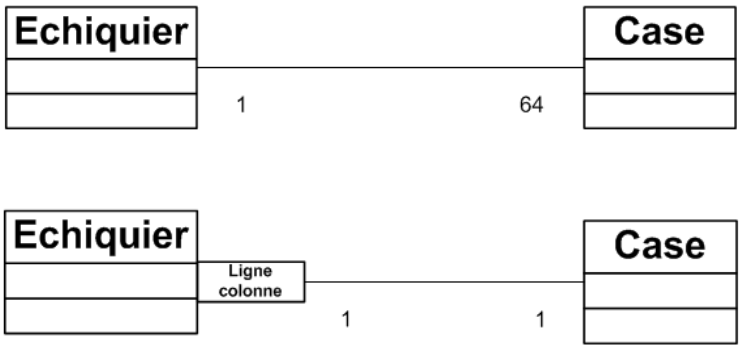
Deux types de précisions apportés par une qualification :

- Sur la multiplicité d'une des extrémités de l'association.
- Sur la portée de l'association par rapport aux classes en relation.

Ⅱ. Relations entre classes

1. Association

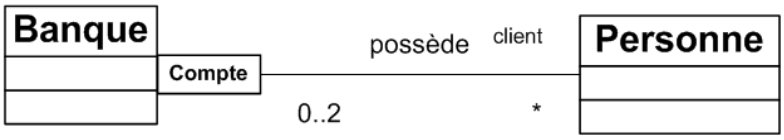
Exemple 1 : changement de la multiplicité



II. Relations entre classes

1. Association

Exemple 2 : restriction de la portée d'une association

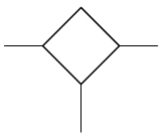


II. Relations entre classes

1. Association

Association n-aire : association qui relie plus de deux classes.

Représentation :

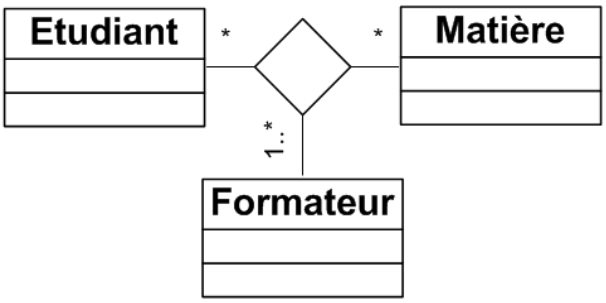


Attention : La détermination de la multiplicité n'est pas immédiate.

Ⅱ. Relations entre classes

1. Association

Exemple :

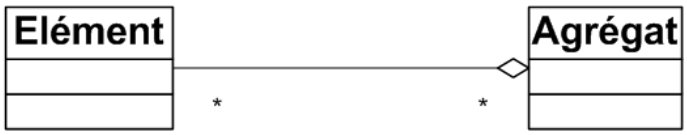


Ⅱ. Relations entre classes

2. Agrégation

Agrégation : forme particulière d'association, non symétrique, qui exprime une relation d'inclusion structurelle ou comportementale d'un élément dans un ensemble.

Représentation :



Ⅱ. Relations entre classes

2. Agrégation

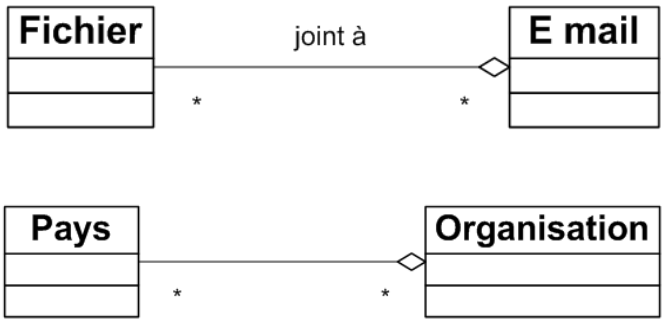
Propriétés de l'agrégation :

- Les cycles de vie de l'agrégat et de ses éléments agrégés sont indépendants : ils peuvent exister l'un sans l'autre.
- Un élément peut appartenir à plusieurs agrégats.

Ⅱ. Relations entre classes

2. Agrégation

Exemples d'agrégation :

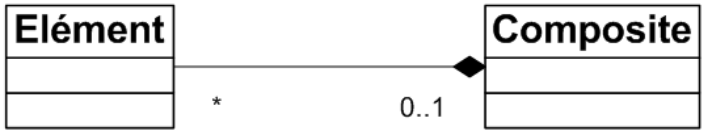


Ⅱ. Relations entre classes

3. Composition

Composition : forme particulière d'agrégation qui exprime l'appartenance exclusive d'une instance à une autre.

Représentation :



Ⅱ. Relations entre classes

3. Composition

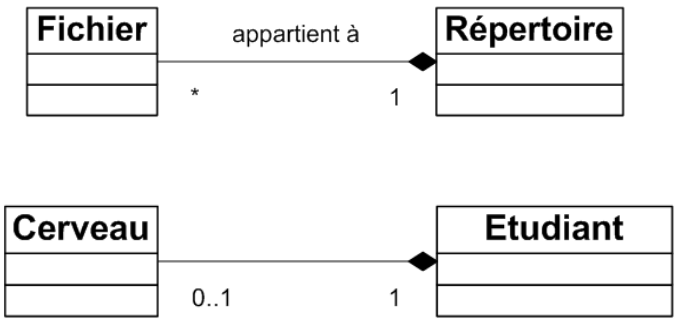
Propriétés de la composition :

- Les cycles de vie du composite et de ses composants sont liés : la création ou la destruction d'un composite implique celle de ses composants.
- Un élément ne peut appartenir qu'à un seul composite.

Ⅱ. Relations entre classes

3. Composition

Exemples de composition :

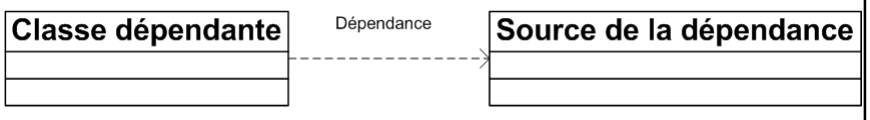


Ⅱ. Relations entre classes

4. Dépendance

Dépendance : relation unidirectionnelle indiquant qu'une classe est modifiée si l'élément dont elle dépend est lui même modifié.

Représentation :



Ⅱ. Relations entre classes

4. Dépendance

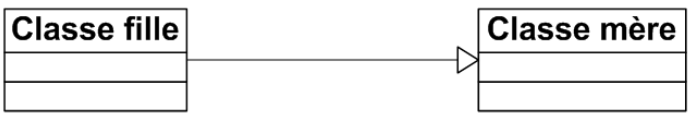
Exemple :

Ⅱ. Relations entre classes

5. Héritage

Héritage : relation de spécialisation/généralisation entre deux classes. Elle indique qu'une classe dite classe fille spécialise une autre classe dite classe mère, i.e. qu'elle possède les attributs et les méthodes de la classe mère plus d'autres qui lui sont propres.

Représentation :



Ⅱ. Relations entre classes

5. Héritage

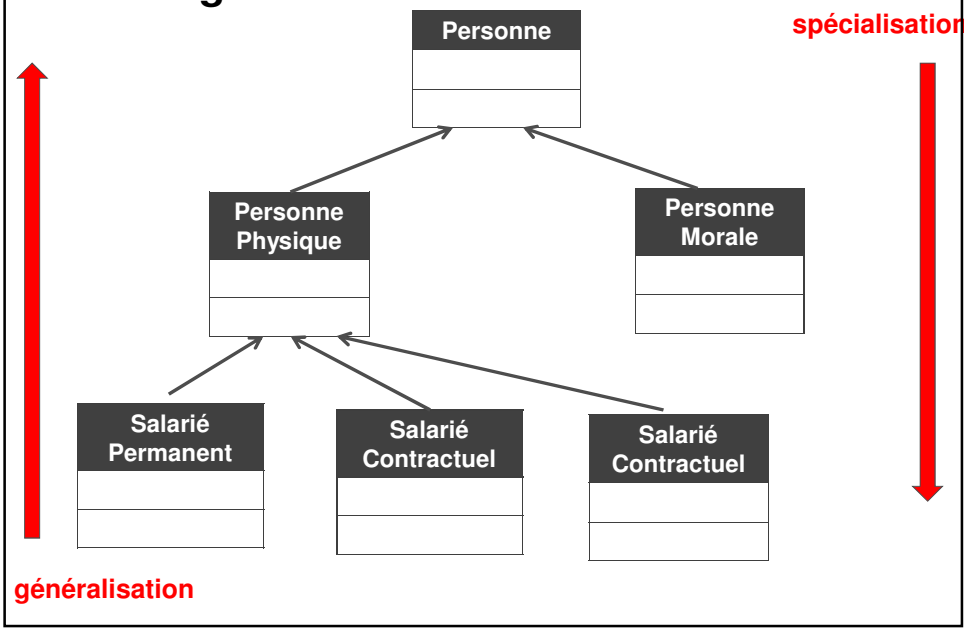
Les deux visions de l'héritage :

Spécialisation : on étend les propriétés d'une classe à des sous-classes plus spécifiques. Cela permet donc la réutilisation de modèles déjà existants.

Généralisation : on factorise les propriétés communes d'un ensemble de classes dans une super-classe plus abstraite. Cela permet de gagner en généricité.

Ⅱ. Relations entre classes

5. Héritage



Ⅱ. Relations entre classes

5. Héritage

Propriétés de l'héritage :

- Une classe fille possède les attributs et méthodes de sa classe mère mais n'a accès à ceux-ci que s'ils sont déclarés « public » ou « protégé ».
- Une classe fille peut redéfinir (avec la même signature) des méthodes de la classe mère. C'est le mécanisme de **polymorphisme**.
- Les associations de la classe mère s'appliquent par défaut aux classes filles.

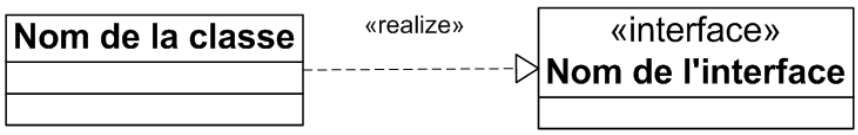
Ⅱ. Relations entre classes

6. Interfaces

Une interface est donc une classe sans attributs et ne contenant que des méthodes abstraites.

Ses méthodes seront alors implémentées par au moins une autre classe. On dit alors que cette dernière **réalise** l'interface.

Représentation :



Ⅱ. Relations entre classes

6. Interfaces

D'autres classes utiliseront l'interface pour réaliser leurs opérations. On les dit **classes clientes de l'interface**.

C'est un lien de dépendance qui unit une classe cliente à une interface, complété par le stéréotype « **uses** ».

Représentation :

Ⅱ. Relations entre classes

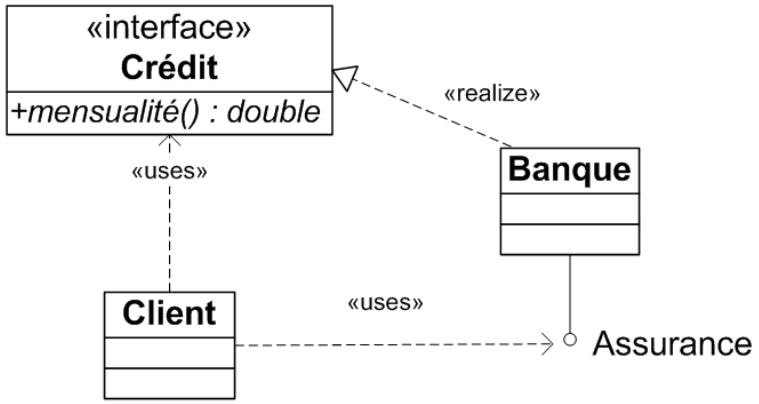
6. Interfaces

But d'une interface

L'interface est utilisée pour diminuer le couplage entre les classes car une classe « cliente » qui utilise les services spécifiés dans une interface n'a pas besoin de connaître quelle classe « serveur » implante réellement ce service ni de quelle manière ces services sont implantés.

6. Interfaces

Exemple :



Vos Questions

Ⅲ. Méthodologie.

Ⅲ. Méthodologie

1. Plan d'élaboration.
2. Diagramme d'objets.

Ⅲ. Méthodologie

1. Plan d'élaboration

- Identifier les classes. Cela peut se faire à partir d'une liste de classes « candidates » fournies par un expert du domaine. On éliminera les classes redondantes et les classes superflues (celles qui ne sont pas en rapport direct avec le problème).
- établir les associations entre les classes.
- Identifier les attributs des classes.

Ⅲ. Méthodologie

1. Plan d'élaboration

- Organiser et simplifier le modèle, en particulier en utilisant la relation d'héritage.
- Recommencer toute cette procédure du début à la fin autant de fois que nécessaire. Au fil des itérations, le modèle s'affinera.
- Utiliser un diagramme d'objets pour donner un exemple, ou pour préciser un aspect délicat du diagramme de classes.

Ⅲ. Méthodologie

2. Diagramme d'objets.

Un **diagramme d'objets** représente des instances de classes et leurs relations.

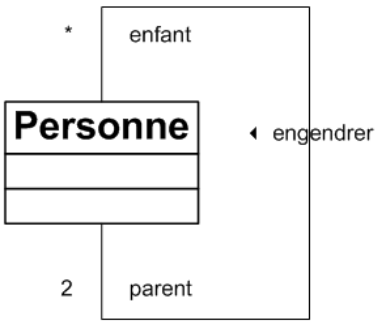
Il sert entre autres à illustrer le diagramme de classes en montrant un exemple explicatif du modèle.

Il va permettre également de **clarifier certaines relations entre classes**, en particulier les associations **réflexives** et **multiples**.

Ⅲ. Méthodologie

2. Diagramme d'objets.

Exemple : reconsidérons cette relation réflexive.

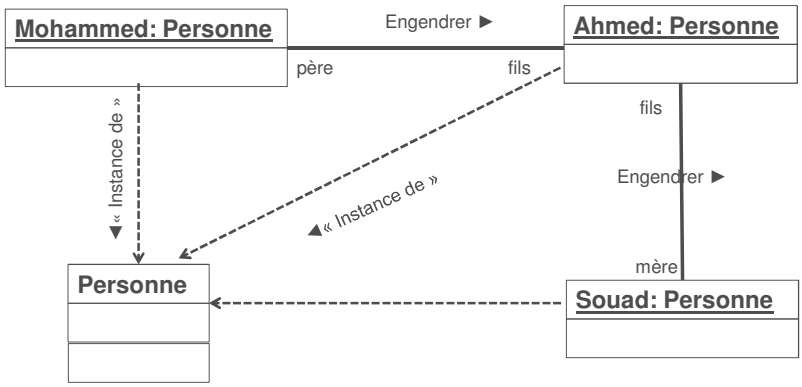


Un diagramme d'objets permettra d'illustrer ce diagramme des classes un peu complexe.

Ⅲ. Méthodologie

2. Diagramme d'objets.

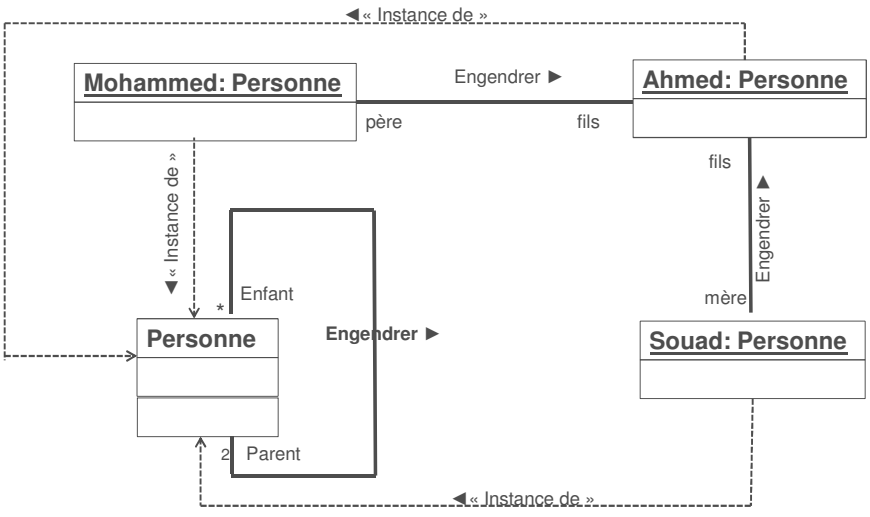
Exemple (suite) : le diagramme d'objets correspondant.



Ⅲ. Méthodologie

2. Diagramme d'objets.

Exemple (suite et fin) : le diagramme d'objets correspondant.



Ⅲ. Méthodologie

Vos Questions