Conception des systèmes d'information
Chapitre 3 : diagramme de classes

Dr. Mariem MAHFOUDH

mariem.mahfoudh@gmail.com

2 LSI-ADBD, ISIMS, 2023-2024

Bibliographie

Ce cours a été construit en se basant sur les références suivantes :

- Livre "UML 2 pratique pour la modélisation", Benoît Charroux, Aomar Osmani et Yann Thierry-Mieg
- Livre "UML 2 pour la pratique", Pascal Roques
- Livre " UML 2 pour les developeurs", Xavier Blanc et Isabelle Mounier
- Livre "UML 2 de l'apprentissage à la pratique", Laurent Audibert
- Livre "Modélisation Objet avec UML", Pr. Pierre Alain Muller
- Cours "Conception des systèmes d'information", Pr. Faiez Gargouri
- Cours "Analyse, Conception Objet", Stephane Galland
- Cours "Langage UML", Emmanuel Remy

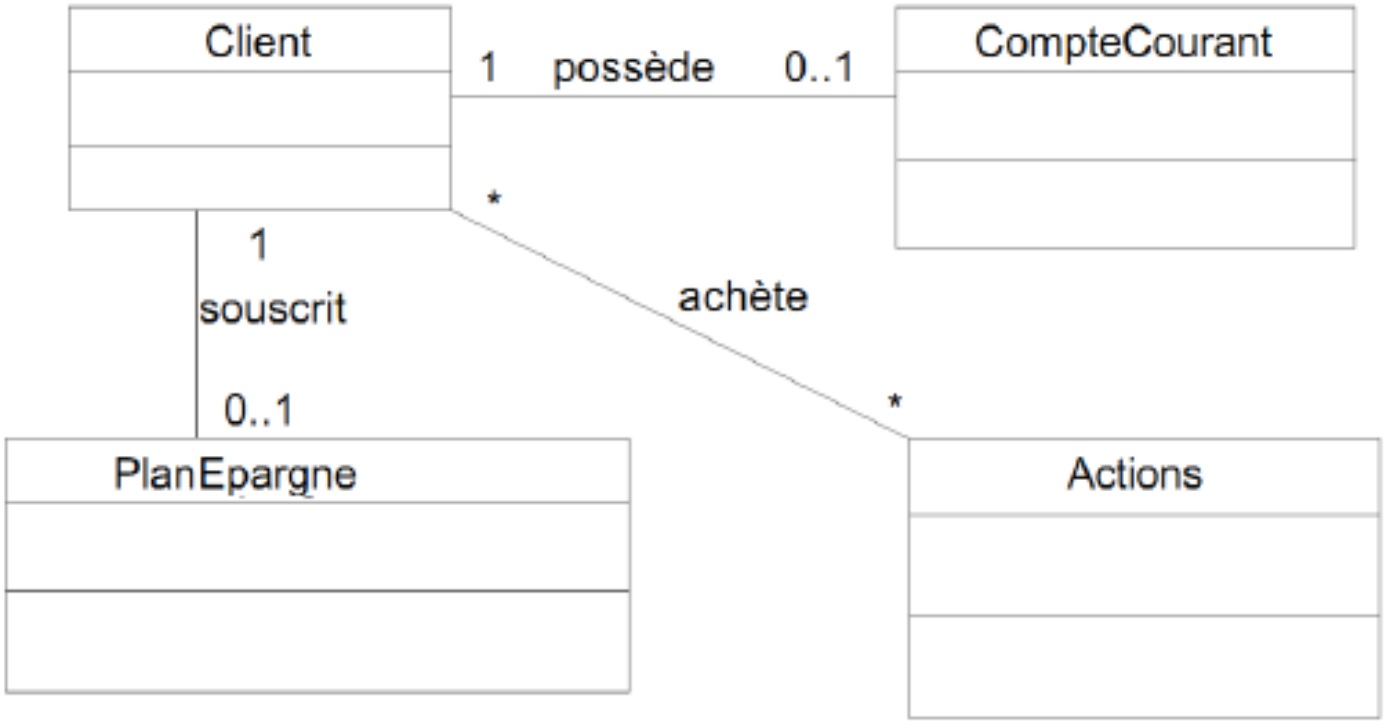
Plan

Diagramme de classe

- Le diagramme des classes est un diagramme structure (statique) qui permet de représenter :
  - les classes (attributs + méthodes) ;
  - les associations (relations) entre les classes.
- Il détermine les données qui seront manipulées par le système, des données stockés en classes.
- Il représente l'architecture conceptuelle du système, la structure interne.
- Il permet de fournir une représentation abstraite des objets du système qui vont interagir pour réaliser les cas d'utilisation.
- C'est un diagramme central du modèle du SI.

Exemple

Le diagramme de classe rassemble au modèle Entité-Association



Exemple de diagramme de classe

Classe

Une classe est un concept abstrait représentant des éléments variés comme :

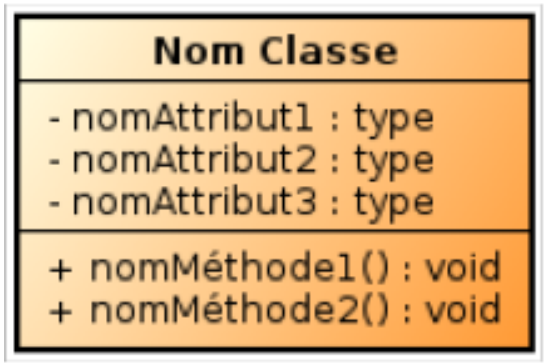
- des éléments concrets (ex. : des avions, des personnes),
- des éléments abstraits (ex. : des commandes de marchandises ou services),
- des composants d'une application (ex. : les boutons des boîtes de dialogue),
- des structures informatiques (ex. : des piles),
- des éléments comportementaux (ex. : des tâches),
- etc.

Classe

Représentation de classe

En UML, la classe est représentée par un rectangle (appelé aussi classeur) divisé essentiellement en 3 compartiments :

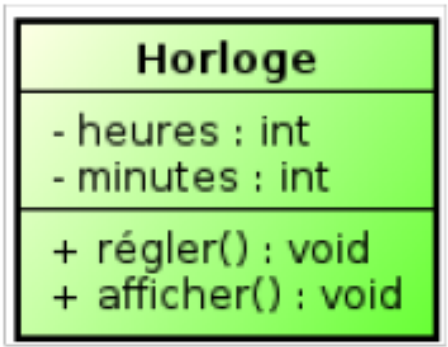
- Le premier compartiment contient le nom de la classe qui représente le type d'objet instancié.
- Le deuxième compartiment contient les attributs.
- Le troisième compartiment contient les méthodes.

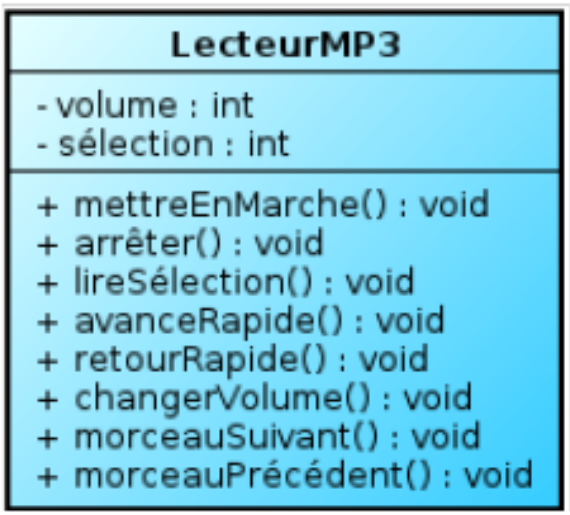


NB. On peut ne pas représenter les attributs et/ou les opérations d'une classe

Classe

- Le nom de la classe, selon la norme UML est en gras, mais on peut se limiter à l'écrire en majuscule.
- Un attribut d'une classe constitue un élément de l'état de ses objets
- Une opération représente un comportement service spécifique offert par les objets de la classe.





Classes : les attributs

Syntaxe

La syntaxe de description des attributs est :

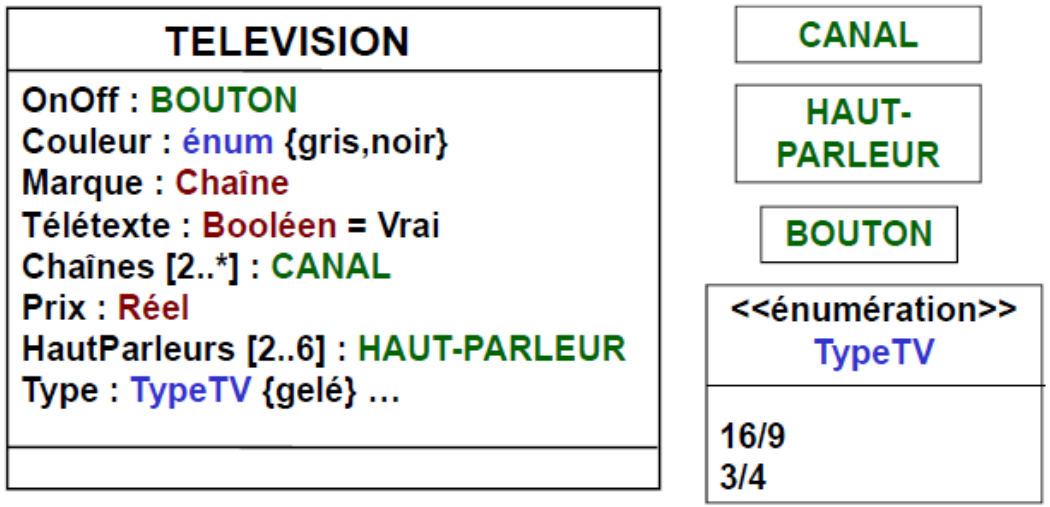
[Visibilité] NomAttribut [Multiplicité] [: Type [=Valeur Initiale] [Propriété]*]

- Visibilité = type d'accessibilité :
  - + : public, visible et modifiable par tout objet
  - - : private, seulement visible et modifiable par les opérations de l'objet auquel il appartient.
  - # : protected, seulement accessible et modifiable par les opérations des classes descendantes
  - Aucun caractère, ni mot-clé. Propriété ou classe visible uniquement dans le paquetage où la classe est définie.
- Multiplicité : intervalle ou nombre
  Multiplicité := (Intervalle|nombre)

Classes : les attributs

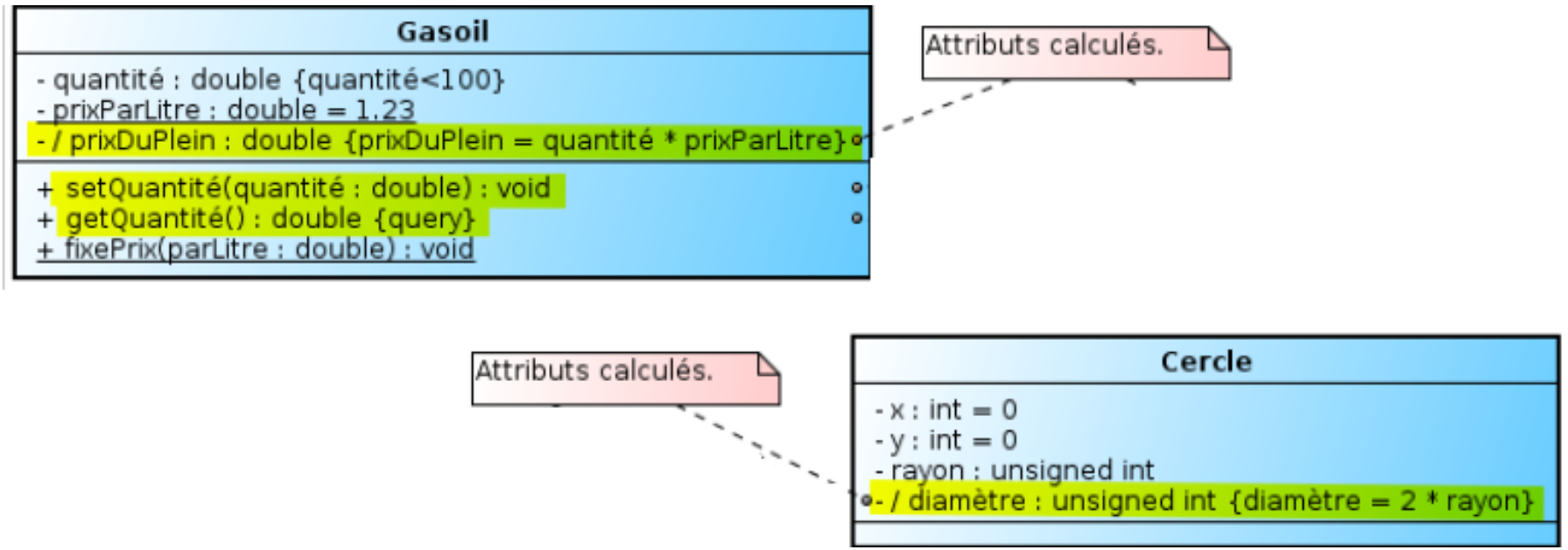
[Visibilité] NomAttribut [Multiplicité] [: Type [=Valeur Initiale] [Propriété]*]

- Le type des attributs peut être :
  - Un type primitif: entier, chaîne,
  - Une classe (type utilisateur) : Personne, Bouton, etc.
- Propriété : Mutabilité (gelé, variable, ajout Uniquement,)
  - Gelé : attribut non modifiable (const de C++).
  - Variable : propriété par défaut, l'attribut est modifiable
  - Ajout Uniquement : seul l'ajout est possible (multiplicité > 1).



Les attributs calculés (dérivé)

- Une classe peut avoir des attributs calculés, c'est-à-dire que leurs valeurs sont proposées au travers d'une fonction utilisant les autres attributs précédemment exprimés.
- Un tel attribut possède un nom précédé du signe " / " et suivi d'une contrainte permettant de le calculer.



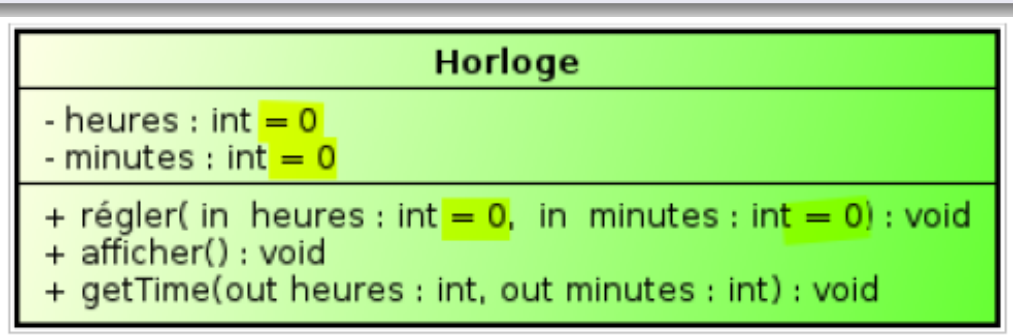
Syntaxe de description des opérations

[Visibilité] NomOpération [[Arguments] : TypeRetourné [Propriété*]]

- Visibilité : +, -, #
- Arguments : [Direction] NomArgument [: TypeArgument] [= ValeurDefaut]

  Direction:

  - In (la valeur par défaut): l'argument est un paramètre en entrée seule non modifié par l'exécution de cette opération ;
  - Out : l'argument est un paramètre en sortie seule l'appelant peut récupérer sa valeur ;
  - InOut : l'argument est un paramètre en entrée-sortie.

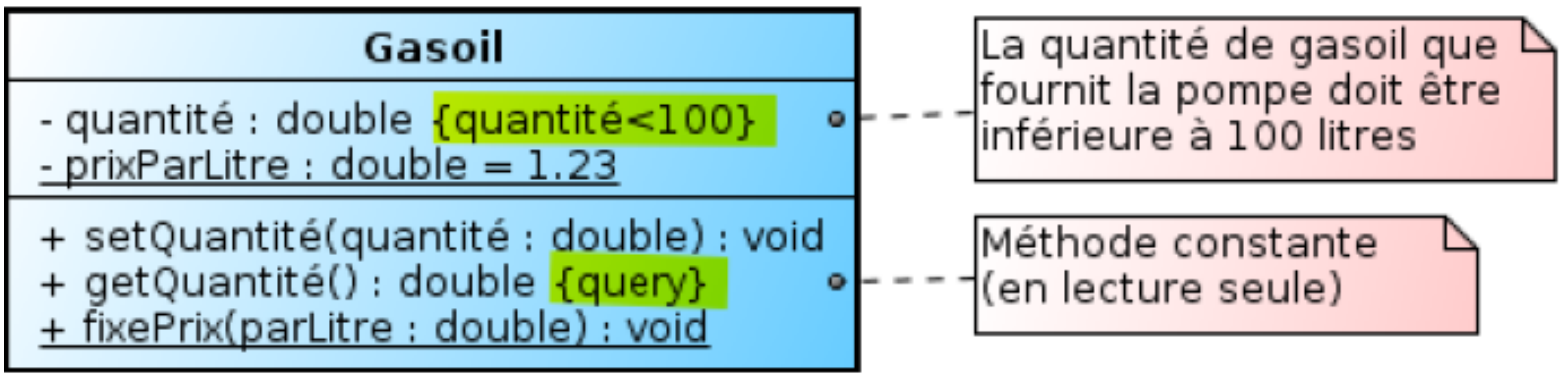


Classes : les opérations

Syntaxe

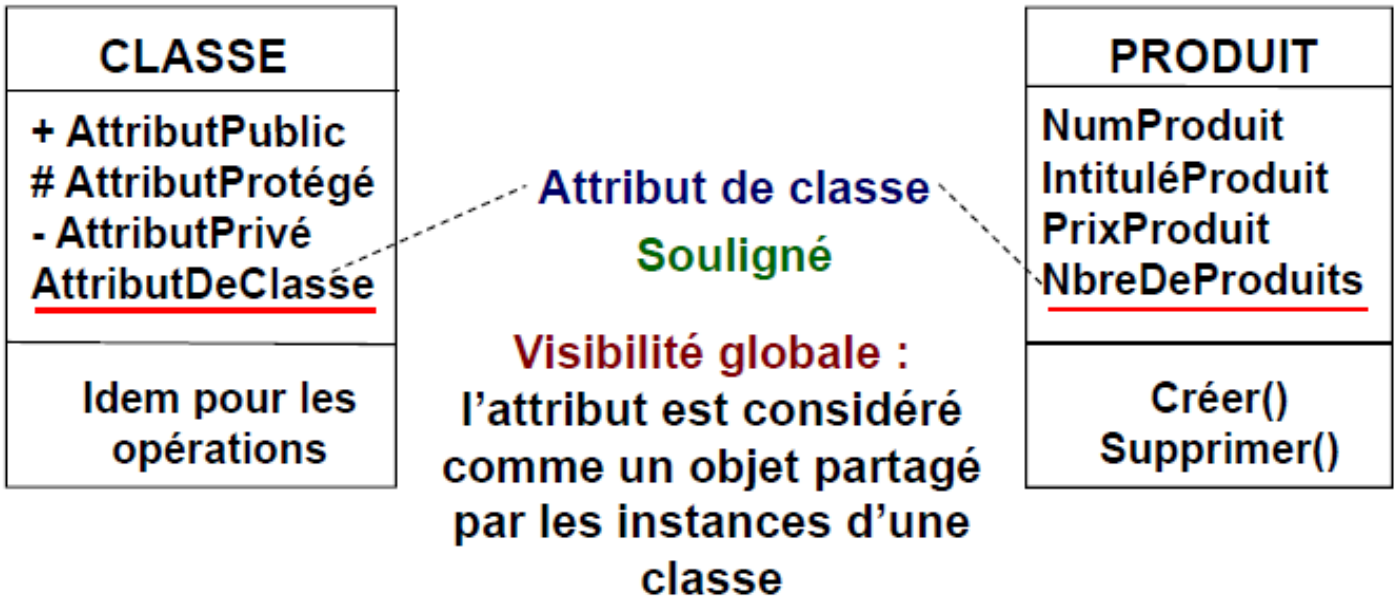
[Visibilité] NomOpération [[Arguments] : TypeRetourné [Propriété*]]

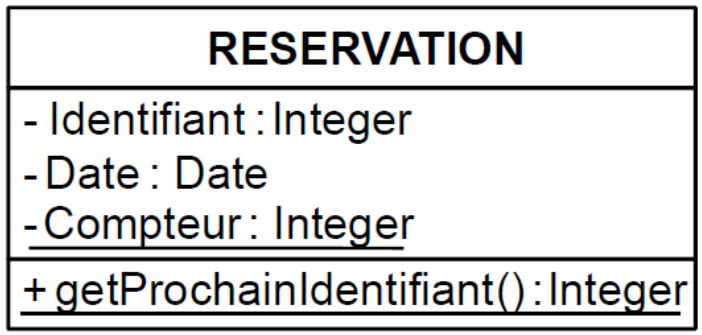
- Le typeRetourné peut être :
  - Un type primitif : entier, chaîne,
  - Une classe (type utilisateur) : Personne, Bouton, etc.
  - Void
- Propriété (ou bien contrainte):
  - requête (query): l'opération ne modifie pas les attributs
  - abstrait : l'opération n'est pas implémentée dans la classe ;
  - récursive : l'opération est récursive ;
  - etc.



Visibilité et portée des attributs et des opérations statiques

Correspondent aux membres static en C++ ou Java





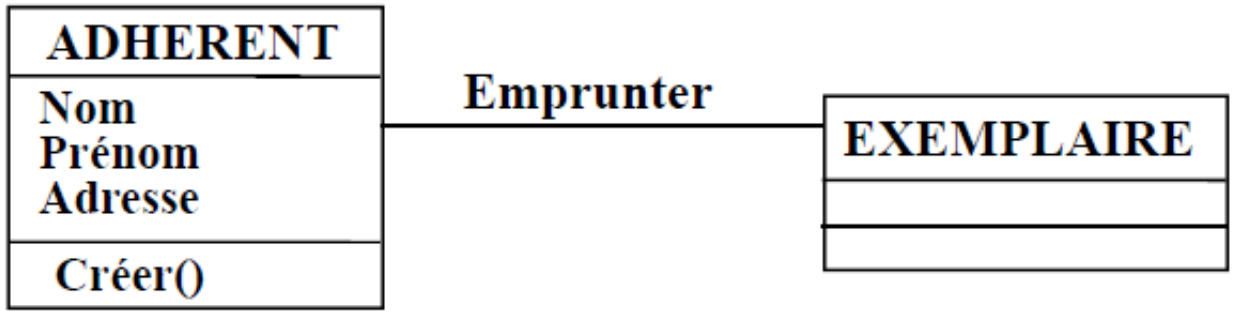
Relations

Les relations entre classes :

- Association
- Agrégation
- Composition
- Héritage

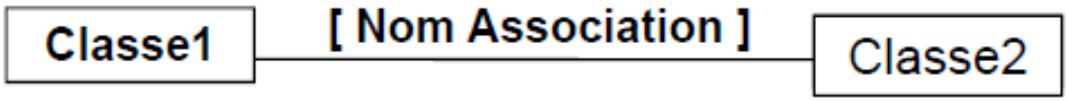
Relations : Associations

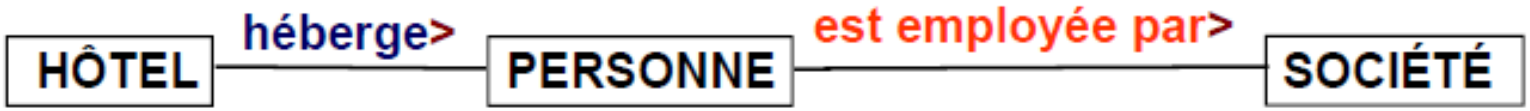
- Une association exprime une connexion sémantique bidirectionnelle entre n classes ($n >= 1$).
- Elle indique qu'une classe est en relation avec une autre pendant un certain laps de temps.
- La ligne de vie des deux objets concernés ne sont cependant pas associés étroitement (un objet peut être détruit sans que l'autre le soit nécessairement).
- L'association est représentée par un simple trait continu, reliant les deux classes.



Nommage des associations

- Une association peut être nommée : c'est optionnel.
- Les noms peuvent être en forme verbale active ou passive
- Le sens de lecture d'une association peut être précisé lorsqu'il est ambigu

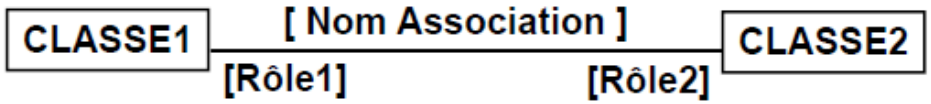


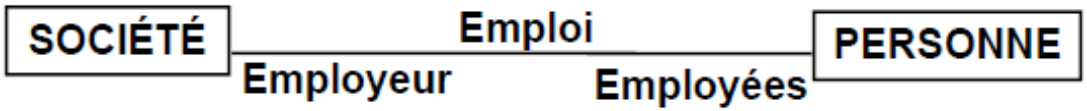


Les rôles

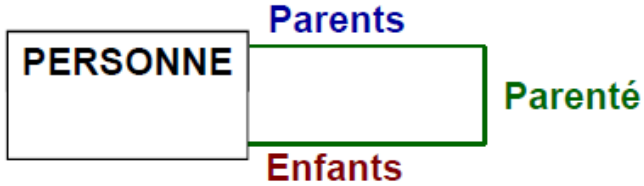
L'extrémité d'une association peut avoir un nom, appelé rôle, qui décrit comment une classe source voit une classe destination au travers de l'association.

- Rôle 1 : le rôle joué par Classe 1 dans l'association
- Rôle 2 : le rôle joué par Classe 2 dans l'association





L'indication des rôles est nécessaire pour les associations ambiguës.

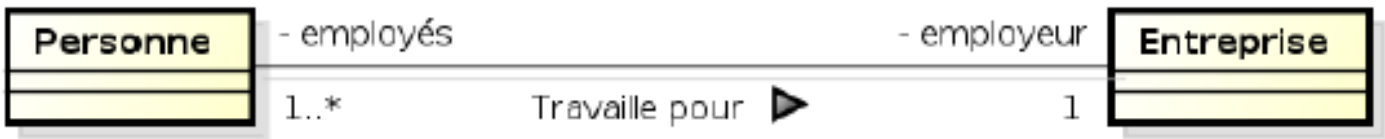


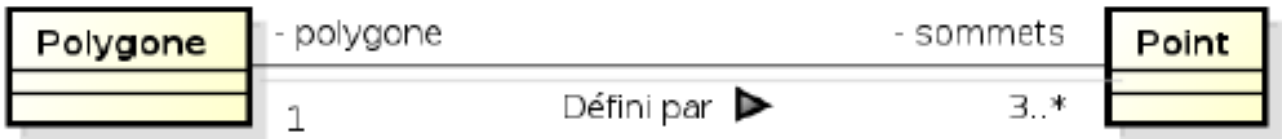
Les multiplicités ou cardinalités

précisent les nombres min et max d'objets d'une classe qui peuvent être liés à un objet de l'autre.

Valeurs de cardinalité conventionnelles

1	Un et un seul
0 .. 1	Zéro ou un
N	N (entier naturel)
M .. N (3..7)	De M à N (entiers naturels)
*	De 0 à plusieurs
0 .. *	De 0 à plusieurs
1 .. *	De 1 à plusieurs





Les multiplicités ou cardinalités

Par rapport au modèle Entité/Association :

Diagramme Entité / Association

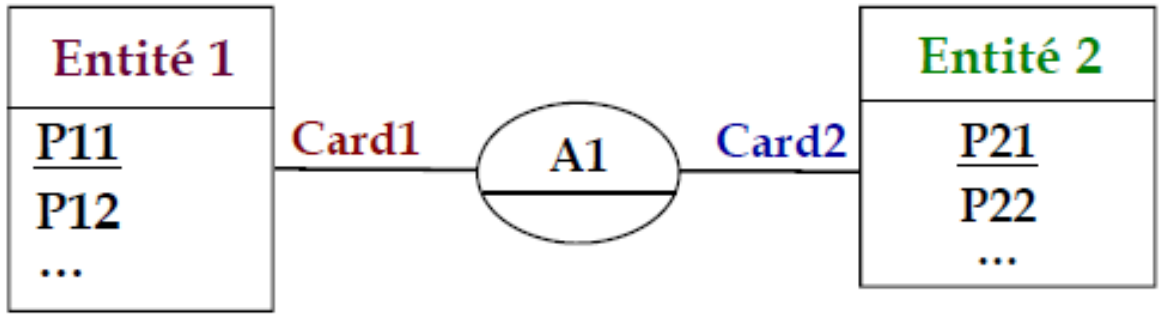
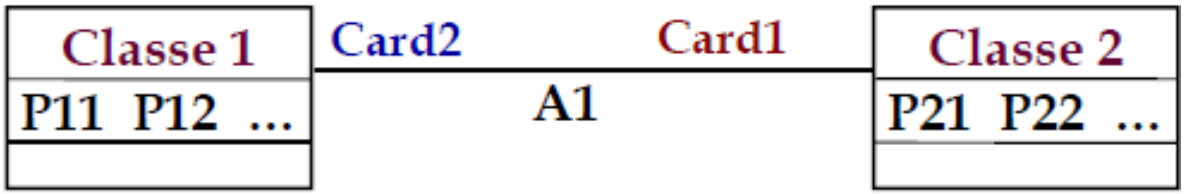
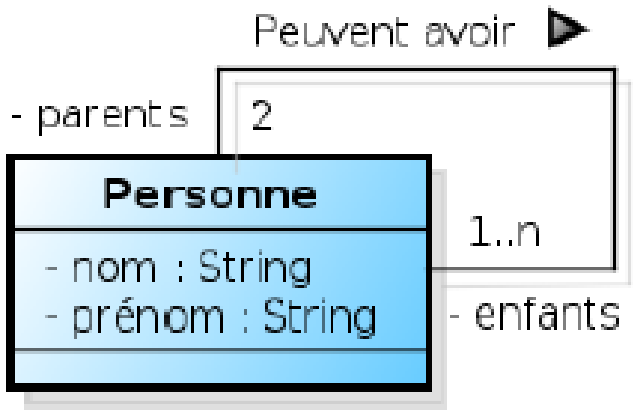


Diagramme de classes



Association réflexive

Une association qui lie une classe avec elle-même est une association réflexive.



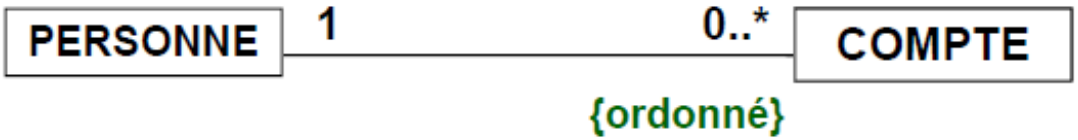
Contraintes

Les types de contraintes exprimables sur les associations :

- Ordonné ;
- Sous-ensemble ;
- Partition (Ou-exclusif) ;
- ...

Ordonné

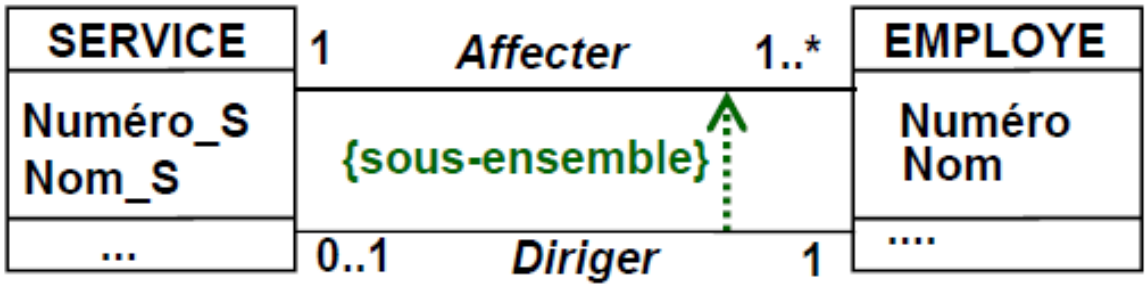
La collection des comptes d'une personne est triée



Contraintes

On désire représenter les règles de gestion suivantes :

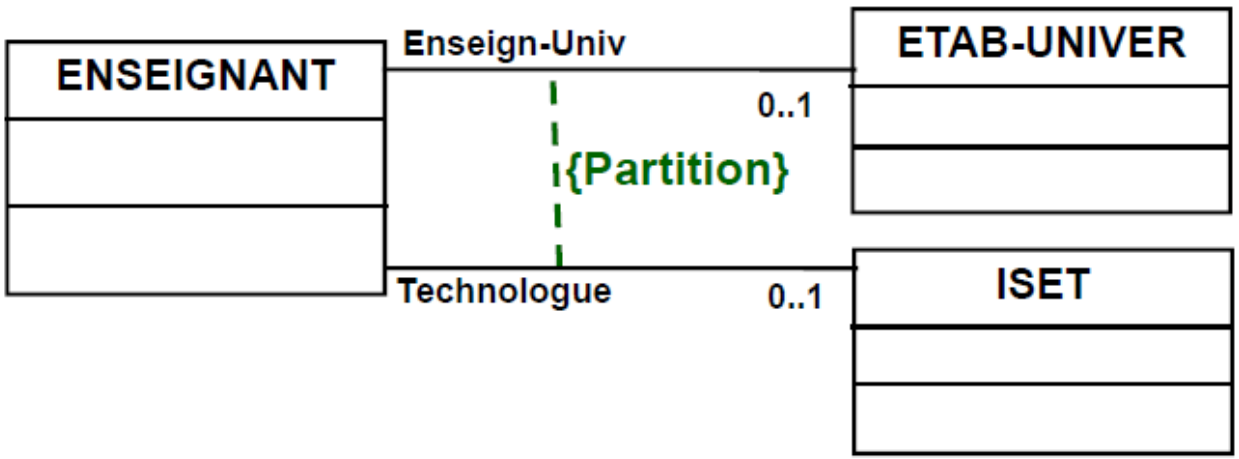
- Un employé est affecté à un seul service.
- Plusieurs employés sont affectés à un service.
- Un service est dirigé par un seul employé.
- Le directeur d'un service est obligatoirement l'un des employés affectés à ce service



Toute instance de diriger est aussi une instance d'affecter

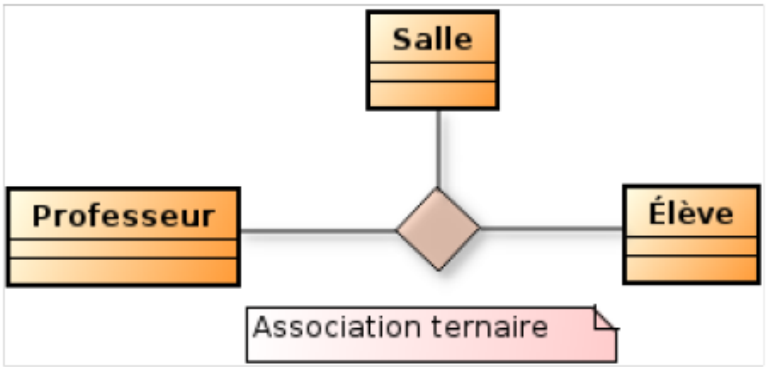
Contraintes

Partition (Ou-exclusif) : indique que pour un objet donné, un seul lien est valide.

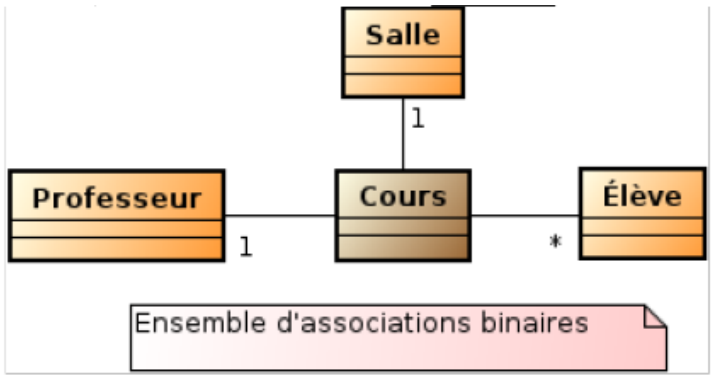


Association n-aire

- Une association qui lie plus de 2 classes entre elles, est une association n-aire.
- L'association n-aire se représente par un losange d'où part un trait allant à chaque classe.
- L'association n-aire est imprécise et souvent source d'erreur. Elle peut être donc remplacée par un ensemble d'associations binaires.



À remplacer par :



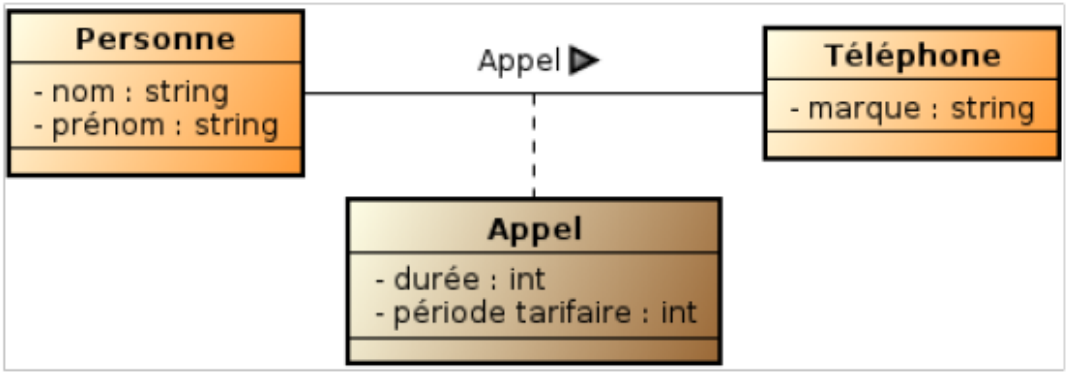
Un cours peut correspondre à l'association ternaire de 3 classes.

Classe d'association ou classe associative

- Permet de représenter une association par une classe pour définir des attributs et/ou des opérations dans l'association.
- Possède les caractéristiques d'une classe et d'une association.

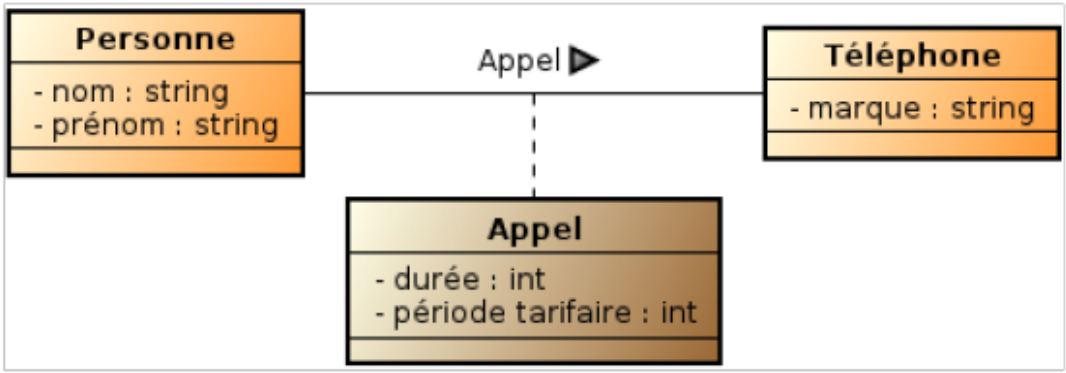
Exemple

Lorsqu'une personne utilise un téléphone, il faut pouvoir mesurer la durée de l'appel et savoir à quel moment il a lieu afin de le tarifier. Nous ajoutons donc deux attributs durée et période tarifaire qui n'appartiennent ni à la classe Personne ni à la classe Téléphone. Ces deux attributs sont mis dans une nouvelle classe (Appel) qui est attachée à l'association.



Classe d'association ou classe associative

Comme pour l'association ternaire, nous pouvons convertir la classe association en un ensemble d'associations binaires

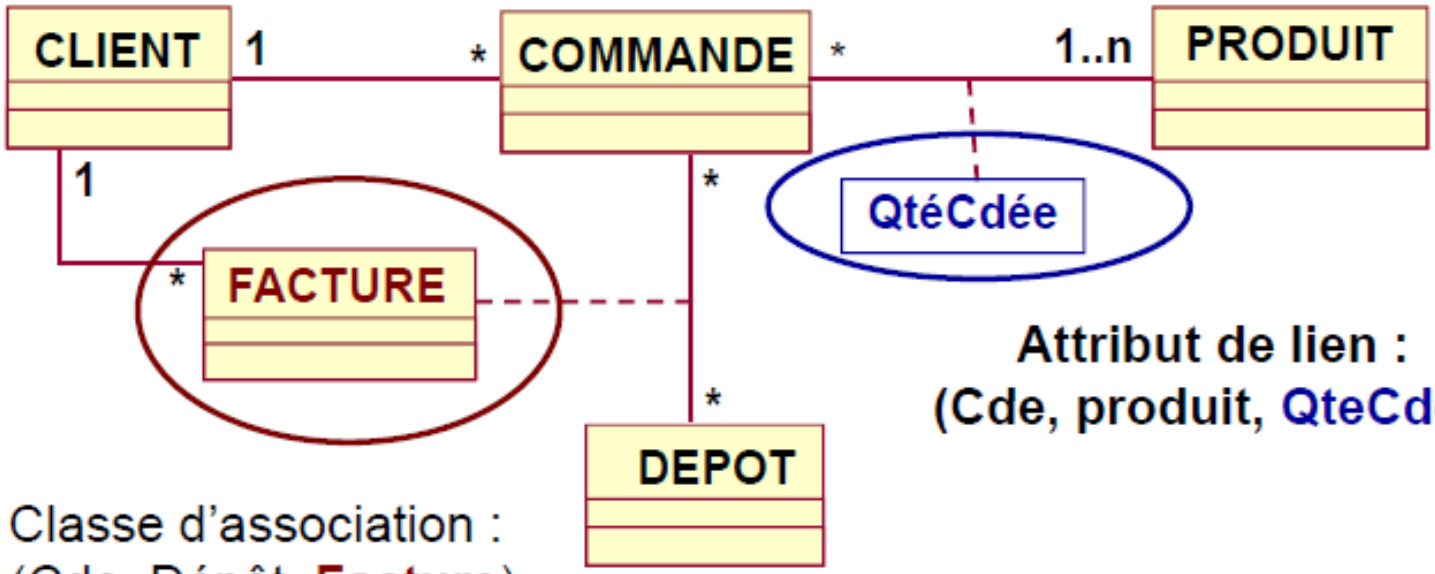


Exercice

Considérons les règles de gestion suivantes :

- Un client passe une ou plusieurs commandes.
- Une commande est passée par un seul client et concerne un ou plusieurs produits.
- Un produit peut être commandé par plusieurs commandes.
- Chaque commande est envoyée à un ou plusieurs dépôts pour être satisfaite.
- Chaque dépôt ayant satisfait la partie qui le concerne d'une commande, génère une facture correspondant à cette partie.
- Toute facture générée sera envoyée au client correspondant

Relations : Associations



Facture est une classe à part entière :
elle a ses propres attributs, opérations et liens

L'aggrégation

- L'agrégation indique qu'un objet A possède un autre objet B, l'objet B peut exister indépendamment de l'objet A. La suppression de l'objet A n'entraîne pas la suppression de l'objet B.
- L'objet A est plutôt à la fois possesseur et utilisateur de l'objet B.
- L'aggrégation représente une relation de type "ensemble/élément".
- Elle se représente par un losange vide du coté de l'objet conteneur.

Relations : Composition

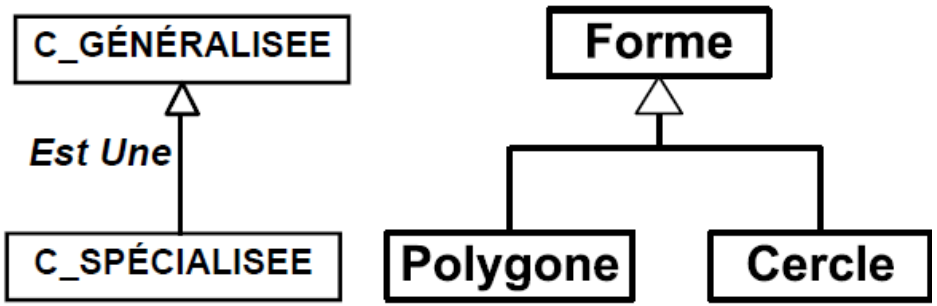
La composition

- Définit une contenance structurelle entre les instances
- La composition indique qu'un objet A (appelé conteneur) est constitué d'un autre objet B. Cet objet A n'appartient qu'a l'objet B et ne peut pas être partagé avec un autre objet.
- Elle se représente par un losange plein du coté de l'objet conteneur

La destruction de l'objet composite implique la destruction de ses composants

L'héritage

- L'héritage est un mécanisme de transmission des caractéristiques (attributs, méthodes) d'une classe à une sous classe
- L'héritage peut être simple ou multiple
- Une sous-classe hérite les attributs, les opérations et les références de ses parents
- Une sous-classe peut : ajouter des attributs, des opérations, des références, redéfinir des opérations héritées



Classes concrètes et abstraites

Classes concrètes

Une classe concrète possède des instances. Elle constitue un modèle complet d'objet (tous les attributs et méthodes sont complètement décrits).

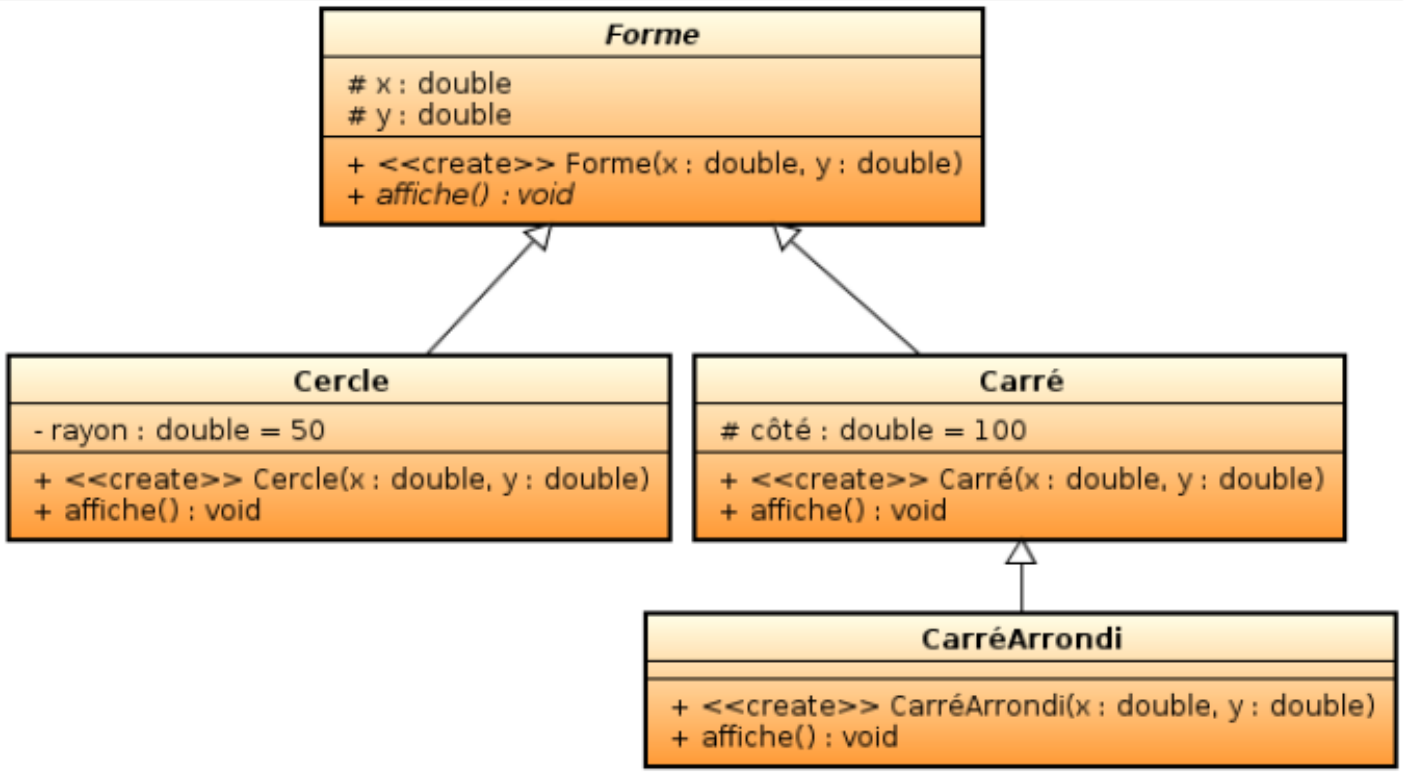
Classes abstraites

- Les classes abstraites ne sont pas instanciables directement.
- Elles servent de spécification générale pour manipuler les objets instances de leurs sous-classes.
- Le nom d'une classe abstraite s'écrit en italique.

Classes concrètes et abstraites

Exemple

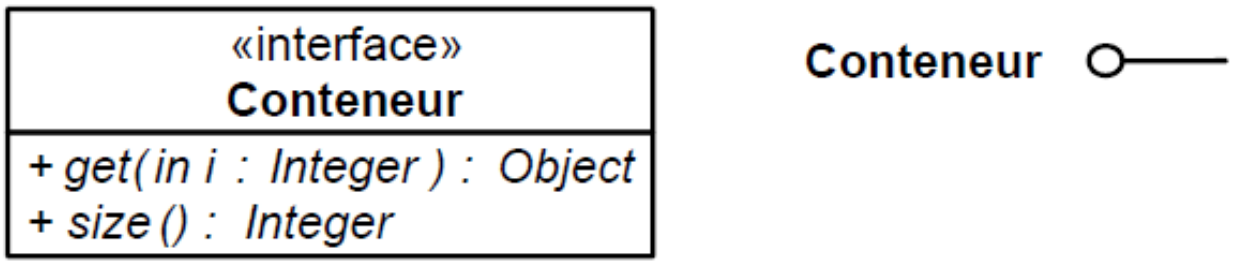
La classe Forme est abstraite puisqu'elle constituée de la méthode abstraite affiche(). Cette méthode est spécifiée abstraite puisqu'il est impossible de décrire un tracé particulier, ne connaissant pas la forme exacte. Ce n'est qu'avec une classe concrète comme Cercle que nous sommes capable de réaliser le tracé correspondant.



Relation de réalisation et notion d'interfaces

Interface

- "Sorte" de classe définie exclusivement par des opérations abstraites.
- Une interface décrit une partie du comportement visible d'une classe, d'un paquetage, ... Ce comportement est défini par une liste d'opérations publiques.
- Elle ne fournit aucune implémentation des opérations.
- Elle ne définit aucun attribut.

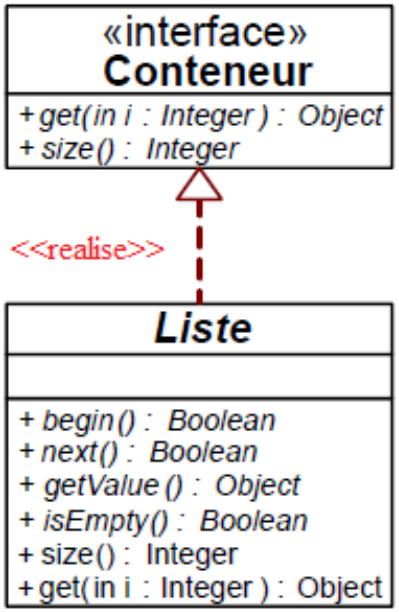


Représentation graphique d'une interface

Interfaces

Interface : relation de réalisation

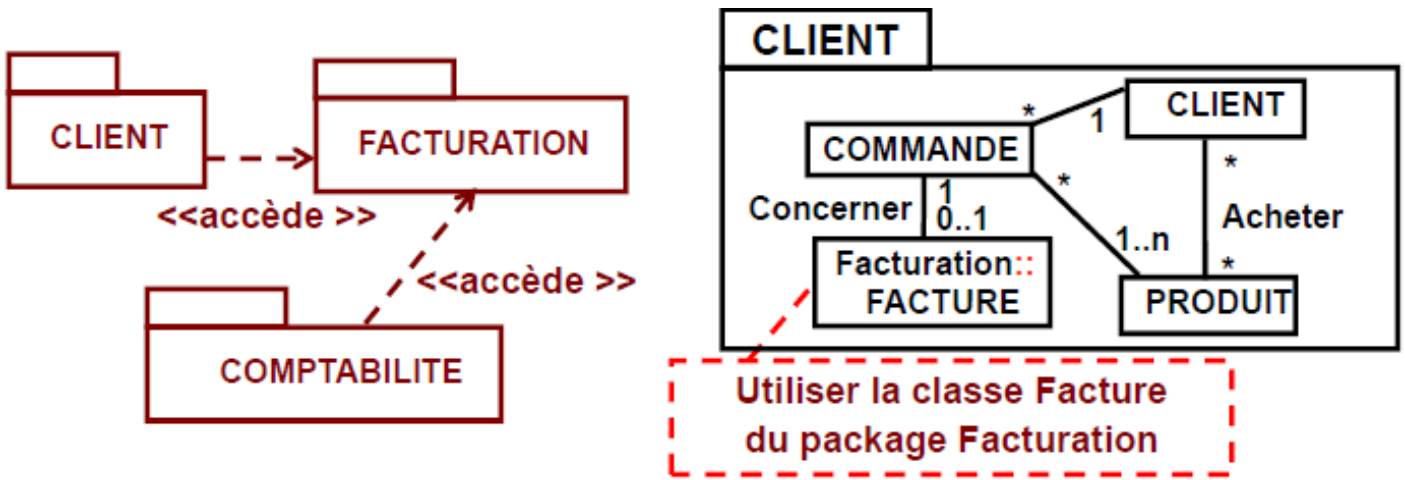
- Une classe peut utiliser tout ou une partie des services décrits dans une interface.
- La classe qui réalise une interface est indiquée par une relation de réalisation : << *realise* >>
- Une classe réalisant une interface doit implémenter tous les services décrits par cette interface.



Les paquetages

Le paquetage

- Un paquetage (ou package) regroupe des éléments de modélisation (EM) : classes, associations et packages
- Un paquetage contient un sous ensemble d'un modèle.
- L'objectif de la décomposition en paquetages est d'avoir une cohérence forte entre éléments d'un même paquetage et couplage faible entre les paquetages



La relation <<accède>> permet de référencer des éléments du package destination.

Construction de diagramme de classe

Démarche

Une démarche couramment utilisée pour bâtir un diagramme de classes consiste à trouver :

- les classes, puis
- les associations entre elles,
- à trouver ensuite les attributs de chaque classe, et enfin
- à organiser et à simplifier le diagramme en utilisant notamment le principe de généralisation.

Il faut réitérer ce processus afin d'améliorer la qualité du modèle.

Construction de diagramme de classe

1. Trouver les classes du domaine

- La première étape consiste, avec l'aide d'un expert du domaine, à trouver une liste de classes candidates.
  La démarche est souvent empirique !
- Dans un premier temps, ne soyez pas trop restrictif. Les classes correspondent souvent à des concepts ou à des substantifs du domaine. Parfois, elles correspondent à des opérations : une transaction pour une banque est une classe, tout comme un appel téléphonique pour un opérateur de téléphonie.
- Après avoir établi une première liste, éliminez les classes redondantes (celles qui modélisent des concepts similaires), ainsi que les classes superflues (celles qui ne sont pas en rapport direct avec le domaine).
- Le nom des classes est important ; il ne doit pas être vague.

2. Trouver les associations entre les classes

- Les associations correspondent souvent à des verbes mettant en relation plusieurs classes, comme "pilote", ou "travail pour".
- Les relations entre classes doivent être modélisées par des associations, et non par des attributs (conducteur pour une classe Pilote est un mauvais choix d'attribut mais conduire en tant qu'association entre les classes Pilote et Voiture est approprié).
- Ne confondre pas une association avec un attribut car un attribut est propre à une classe tandis qu'une association met en jeu plusieurs classes.

Construction de diagramme de classe

3. Trouver les attributs des classes

- Les attributs correspondent généralement à des substantifs tels que "la masse d"une voiture"ou "le montant d'une transaction".
- Les adjectifs tels que "rouge" ou des expressions telles que "50 euros" représentent souvent des valeurs d'attribut.

4. Organiser et simplifier le diagramme en utilisant l'héritage

- Un héritage dénote une relation de généralisation spécialisation entre plusieurs classes.
- Vous pouvez construire un graphe d'héritage "vers le bas", en partant d'une classe du domaine et en la spécialisant en une ou plusieurs $sous - classes$, ou "vers le haut", en factorisant les propriétés de plusieurs classes pour former une classe plus générale.

Exercice

Etablir les diagrammes de classes correspondants aux situations suivantes:

1. Une personne est caractérisée par son nom, son prénom, son sexe et son âge.

   Les responsabilités de la classe sont entre autres le calcul de l'âge, le calcul des charges.

   Les attributs de la classe sont privés ; le nom, le prénom ainsi que l'âge de la personne font partie de l'interface de la classe Personne.

   Deux types de revenus sont envisagés, le salaire et toutes les sources de revenus autres que le salaire, qui sont tous deux représentés par des entiers. On calcule les charges en appliquant un coefficient fixe de 15 % sur les salaires et un coefficient de 20 % sur les autres revenus.

2. Un bateau contient des cabines, occupées par des personnes. Ces dernières sont des guides, des animateurs, ou des passagers. Les guides proposent des visites aux passagers. Les animateurs organisent des animations pour les passagers.

Chapitre suivant

Chapitre 3 : Diagrammes d'objet