Analyse, Conception Objet

Diagrammes de Classes

Une partie du matériau de ce cours est issue du cours de S.Galland (Stephane.Galland@emse.fr)

Septembre 2003

Sommaire

Définition

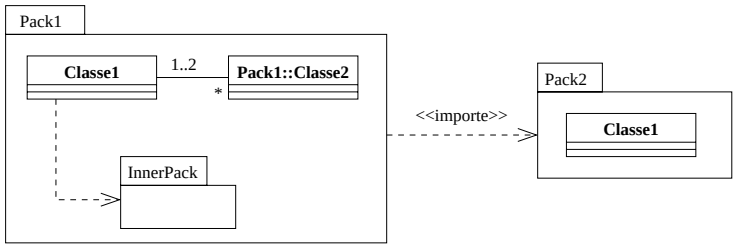
- Point central de la modélisation du système pour exprimer sa structure statique.
- Représentation d'un ensemble de classes, d'interfaces et de paquetages ainsi que leurs relations.
- Une classe décrit un ensemble d'objets (instances de la classe).
- Une association décrit un ensemble de liens (instances de l'association).

Utilisation

- Lors de l'analyse et de la conception:
  - Définitions formelles des objets qui composent le système à partir des cas d'utilisation et des diagrammes d'interaction (séquences et collaboration).
  - Bases conceptuelles pour les diagrammes d'état-transition, de déploiement, ...
- Lors de l'implantation :
  - Génération automatique des structures statiques du système (classes, relations, ...).

Paquetages

- Définition : Mécanisme de partitionnement des modèles et de regroupement des éléments de modélisation.
- Chaque paquetage peut contenir un ensemble de diagrammes et/ou de paquetages.
- Chaque élément d'un paquetage possède un nom unique dans ce paquetage.
- Possibilité de définir des relations entre paquetages (dépendances, cf. plus loin).

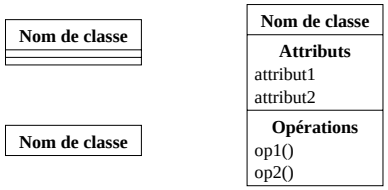


Classe

- Définition : description d'un ensemble d'objets partageant les mêmes attributs, opérations, méthodes, relations et sémantiques.
- Généralement, en fonction de l'objectif du diagramme, elle est décrite par : un nom (**obligatoire**), des attributs, des opérations, des exceptions, ...
- Un nom de classe est unique au sein du paquetage
  ⇝ `Nom de paquetage::nom de la classe`
- Rque : les objets sont représentés comme les classes avec leur nom souligné.

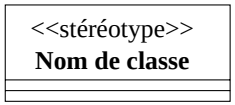
Classe (suite)

- Représentation UML :



Stéréotype

- Un stéréotype permet d'étendre les classes déjà existantes en leur donnant une signification sémantique différente.
- Si la classe `A` est un stéréotype de la classe `B`, alors `A` se comporte comme `B` tout en ayant une signification sémantique différente.
- Mécanisme proche de la généralisation/spécialisation sauf qu'il permet le changement de sémantique.
- Représentation UML :



Stéréotype (suite)

- Quelques stéréotypes prédéfinis :

  ≪**énumération**≫**:** classe définissant un ensemble d'identificateurs formant le domaine de valeur d'un type.

  ≪**utilitaire**≫**:** classe réduite au concept de module et qui ne peut être instanciée.

  ≪**acteur**≫**:** classe modélisant un ensemble de rôles joués par un acteur.

  ≪**interface**≫**:** classe contenant uniquement une description des opérations visibles.

  ≪**exception**≫**:** classe modélisant un cas particulier de signal : les exceptions.

Attribut

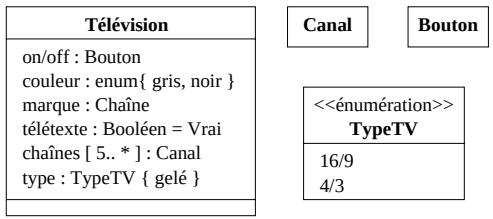
- Définition : propriété définie par un nom, un type et éventuellement une valeur initiale.
- Syntaxe UML :
  `[ visibilité ] nom_attribut [ multiplicité ] : type_attribut [ = valeur_initiale ]`
  - `visibilité` : cf. plus loin.
  - `nom_attribut` : identificateur de l'attribut, unique au sein de la classe.
  - `multiplicité` : l'attribut représente un ensemble de valeurs; exemple de tableau: `Parents[ 1..2 ] : Personne`.
  - `valeur_initiale` : valeur prise par l'attribut lors de l'instanciation de la classe (valeur en concordance avec le type de l'attribut).

Attribut : Type

- `type_attribut` : ensemble des valeurs pouvant être prises par l'attribut.
  Il peut être :
  - une classe : `Rectangle`, `Cercle`, `Personne`, ...
  - un type primitif : `Entier`, `Chaîne de caractères`, `Booléen`, ...
  - une expression complexe dont la syntaxe n'est pas précisée par UML : `ensemble de n points`, ...

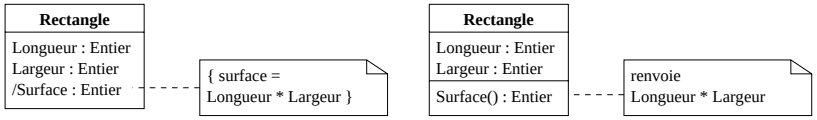
Attribut : Type (suite)

- Mutabilité : contrainte quant aux possibilités de modificiation d'un attribut.
  La mutabilité peut être :
  - {`gelé`} : attribut non modifiable (ou constante),
  - {`variable`} : modifiable à tout moment (mutabilité par défaut),
  - {`ajoutUniquement`} : seul l'ajout est possible (si multiplicité > 1)



Attribut : attribut dérivé

- Parfois des propriétés redondantes sont spécifiées lors de l'analyse.
- Les propriétés entièrement dépendantes d'autres propriétés peuvent être exprimées à l'aide d'attributs dérivés.
- Un attribut dérivé peut être traduit par une opération.



Opération

- Définition : spécification du comportement des instances de la classe.
- Cinq catégories d'opérations :
  - les constructeurs qui créent les objets,
  - les destructeurs qui détruisent les objets,
  - les sélecteurs (opérations de consultation) qui renvoient tout ou partie de l'état d'un objet,
  - les modificateurs qui changent tout ou partie de l'état d'un objet,
  - les itérateurs qui visitent l'état d'un objet ou le contenu d'une structure de données contenant des objets.

Opération (suite)

- Syntaxe UML :
  `[ visibilité ] nom_opération ( [ arguments ] ) : type_retourné propriétés`
  - `visibilité` : cf. plus loin.
  - `nom_opération` : identificateur de l'opération, unique au sein de la classe.
  - `type_retourné` : type de la valeur retournée par l'opération; si omis l'opération ne retourne aucune valeur.

Opération : arguments

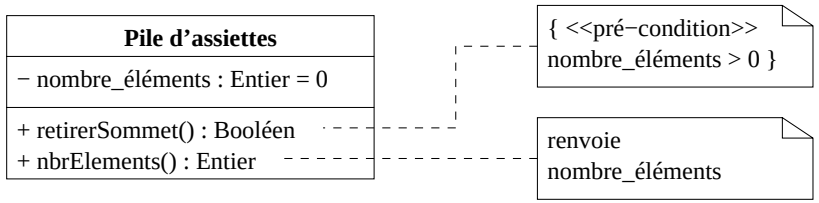
- `arguments` : description des valeurs nécessaire à l'opération.
- Syntaxe UML :
  `[ direction ] nom_argument : type_argument [ = valeur_par_défaut ] [ , autres_arguments ]`
  - `valeur_par_défaut` : valeur prise par l'argument si aucune valeur n'est donnée lors de l'utilisation de cette opération.
  - `direction` : UML définit trois directions pour les arguments
    - `in` (par défaut) : paramètre en entrée seule et non modifié par l'exécution de l'opération,
    - `out` : paramètre en sortie seule; l'appelant peut ainsi récupérer des informations,
    - `inout` : paramètre en entrée-sortie; l'argument est passé à l'opération et modifiable.

Opération : propriétés

- UML définit cinq propriétés pour les opérations:
  - `requête [ = vrai | faux ]` : si `vrai` (par défaut), l'opération n'altère pas l'état de l'instance.
  - `abstrait [ = vrai | faux ]` : si `vrai` (par défaut), indique une opération non implémentée c-à-d, une opération dont l'implémentation devra être obligatoirement réalisée par les classes filles.
  - `estFeuille [ = vrai | faux ]` : si `vrai` (par défaut), indique une opération qui ne peut pas être réimplémentée par une classe fille.
  - `estRacine [ = vrai | faux ]` : si `vrai` (par défaut), indique qu'une opération est définie pour la première fois dans une hiérarchie de classes.
  - `concurrence = séquentiel | gardé | concurrent` : précise le mécanisme d'exécution concurrente de l'opération.

Opération : implémentation

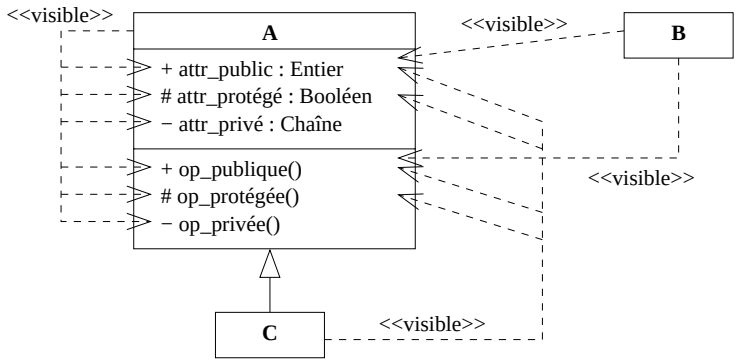
- Spécification d'une pré-condition : condition qui doit être toujours vraie avant l'exécution de l'opération.
- Spécifciation d'une post-condition : condition qui est toujours vraie après l'exécution de l'opération.
- Implémentation avec des diagrammes état-transition, des diagrammes de collaboration, du pseudo-code, ...



Visibilité des membres

- UML définit trois niveaux de visibilité :
  - `public` : l'élément est visible pour tous les clients/utilisateurs de la classe, noté par `+`.
  - `protégé` : l'élément est visible pour les sous-classes de la classe, noté par `#`.
  - `privé` : l'élément est visible pour la classe seule, noté par `-`.

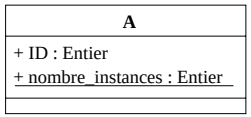
Visibilité des membres (suite)



Portée des membres

- UML définit deux niveaux de portée :
  - portée d'instance (par défaut) : les éléments sont valides pour d'une seule instance de la classe; les éléments n'ont aucune existance en dehors de l'instance.
  - portée de la classe : les éléments sont toujours valides; ils ne sont pas attachés à une instance particulière mais à une classe.
- Syntaxe UML : un élément ayant une portée de classe est souligné

```
+ Parents [ 1..2 ] :  Personne
```



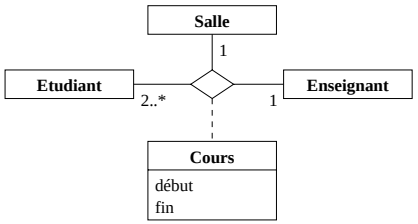
Association

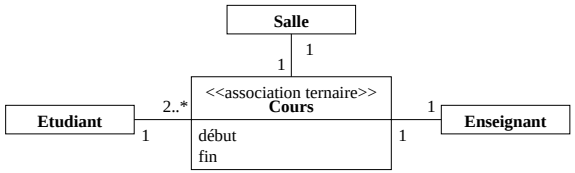
- Définition : relation entre au moins deux classes qui entraînent des connexions entre leurs instances.
- Représentation UML : trait reliant les deux classes en relation.

Association : arité

- Les association ont le plus souvent une arité binaire : deux classes en relation.
- Représentation UML des associations d'arité supérieure (n-aire) : losange.
- Traduire une association d'arité n-aire en un ensemble d'associations binaires.
- Note : la difficulté de trouver un nom différent pour chaque extrémité d'une association n-aire est souvent le signe d'une association d'arité inférieure.

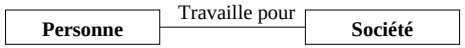
Association : arité (suite)





Association : nommage

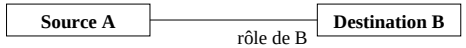
- Les associations peuvent être nommées c.-à-d. identifiées par un texte unique décrivant la sémantique de l'association.

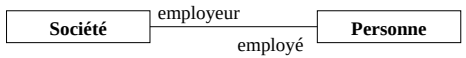


- Note : utiliser une forme verbale active ("travaille pour") ou passive ("employé par").
- Si ambiguïté, indiquer le sens de lecture avec les signes ◀ ou ▶ (par défaut lecture de gauche à droite).

Association : rôles des extrémités

- Les extrémités des associations peuvent être qualifiées par des rôles.
- Un rôle indique comment une classe `Source` voit une classe `Destination`.

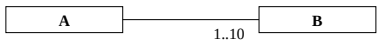


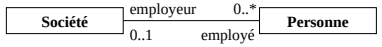


- Le rôle est un pseudo-attribut de la classe source (utilisé comme un attribut).
- Note : ne pas utiliser à la fois le nommage d'une association et les rôles des extrémités de la même association.

Association : multiplicité

- Définition : la multiplicité précise le nombre d'instances pouvant être liées par une extrémité d'association à une instance pour chaque autre extrémité d'association.
- Sous sa forme générale, elle s'exprime par une suite d'intervalles disjoints sur l'ensemble des entiers naturels.
- Sous-estimer la multiplicité des associations réduit la flexibilité du modèle, la sur-estimer conduit à des développements inefficaces.





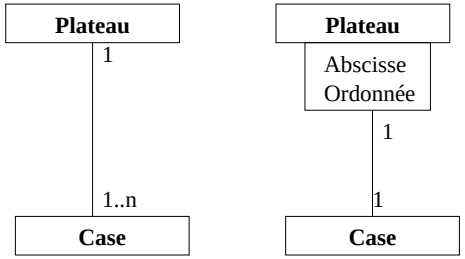
Association : multiplicité (suite)

- Multiplicités définies par UML :

`1`	un et un seul (multipicité par défaut)
`0..1`	zéro ou un
`N`	exactement `N`
`M..N`	de `M` à `N`
`*`	zéro ou plus, c.-à-d. de 0 à $+\infty$
`0..*`	zéro ou plus, c.-à-d. de 0 à $+\infty$
`1..*`	un ou plus, c.-à-d. de 1 à $+\infty$

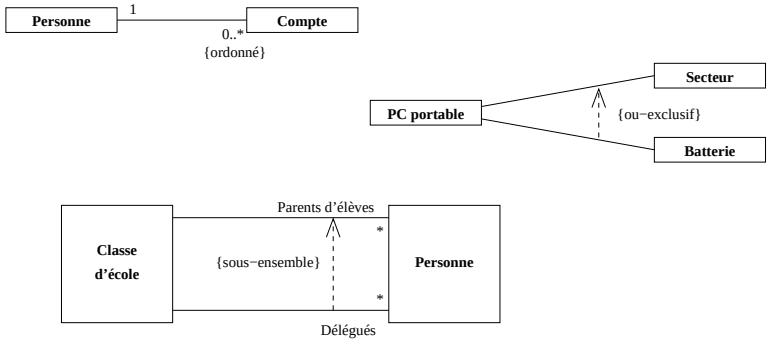
Association : qualification

- Clé d'accés sur l'ensemble des instances que représente l'association.



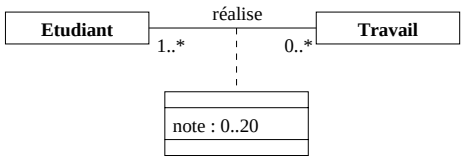
Association : contraintes

- Expression de contraints sur les extrémités, entre associations, . . . (cf. cours OCL).



Association : classe-association

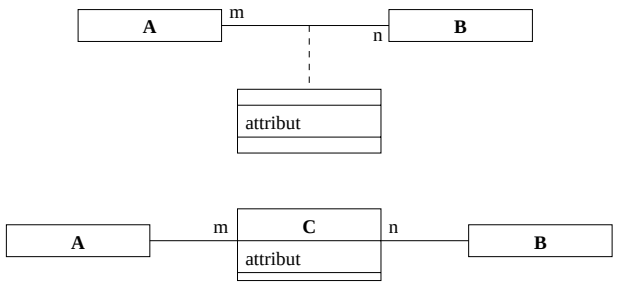
- Si une association possède des propriétés ou des opérations, il est possible de la qualifier à l'aide d'une classe-association.
- Une classe-association possède les mêmes caractéristiques que les associations et les classes.



- Une classe-association qui ne participe pas à d'autres relations avec d'autres classes peut ne pas porter de nom.

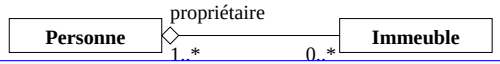
Association : classe-association (suite)

- Lors de la conception, une classe-association peut être remplacée par une classe intermédiaire.



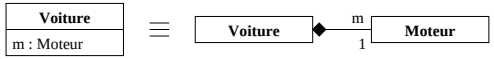
Aggrégation

- Définition : forme spéciale d'association exprimant une relation de composition entre aggrégats (part-whole).
- Association irréflexive, antisymétrique dans laquelle une des extrémitiés joue un rôle prédominant, pouvant être récursive.
- Permet de modéliser les contraintes d'intégrité et de désigner les aggrégats comme garant de ces contraintes.
- A travers une aggrégation, il est possible de représenter :
  - la propagation des valeurs d'attributs d'une classe vers l'autre;
  - une action sur une classe qui implique une action sur une autre classe (ex: copie profonde);
  - une subordination des objets d'une classe à ceux d'une autre.



Composition

- Cas particulier d'aggrégation. La classe ayant le rôle prédominant est la classe composite ou classe conteneur.
- La composition implique :
  - la durée de vie des composants est la même que celle du composite.
  - la multiplicité du côté du composite prend ses valeurs dans `0` ou `1`.
- La composition et les attributs sont sémantiquement équivalents.

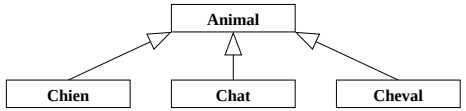


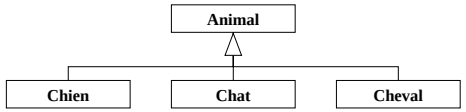
Généralisation

- Définition : relation (irréflexive, antisymmétrique, transitive) entre une classe plus générale et une classe plus spécifique (signifie "est un" ou "est une sorte de"). Ce n'est pas une association.
- Exemple : un animal est un concept plus général qu'un chat ou un chien. Inversement un chien est un concept plus spécialisé qu'un animal. La classe `Animal` est une généralisation de la classe `Chat` ou la classe `Chien`. La classe `Chien` est une spécialisation de la classe `Animal`.
- L'élément plus spécifique peut contenir des informations qui lui sont propres, à condition que ces informations et la description des éléments plus généraux soient totalement cohérentes.
- Deux types de généralisation : simple ou multiple.

Généralisation : simple

- La généralisation simple est une relation binaire.
- Représentation UML : flèche triangulaire blanche orientée vers la classe la plus générale.



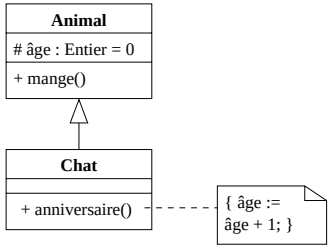


Généralisation : simple (suite)

- La classe la plus générale peut être nommée : “classe mère”, “classe parent”, “superclasse”, “classe de base”.
- La classe la plus spécialisée peut être nommée : “classe fille”, “classe enfant”, “sous-classe”, “classe dérivée”.
- La classe la plus élévée dans la hiérarchie (pas de généralisation) est souvent nommée “classe racine”.

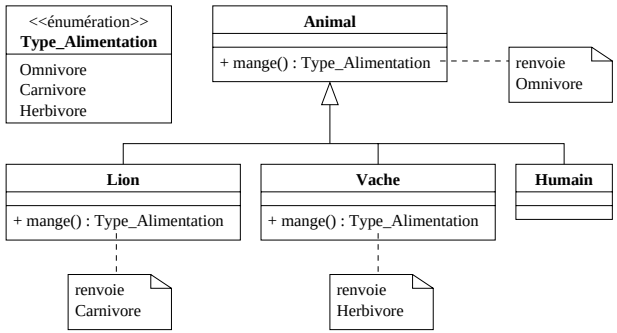
Généralisation : simple (suite)

- Les sous-classes héritent des attributs, des opérations, des relations et des contraintes définies dans la classe mère.
- Héritage: mécanisme permettant à une classe d’utiliser les membres de sa classe mère sans avoir à les redéfinir.



Polymorphisme

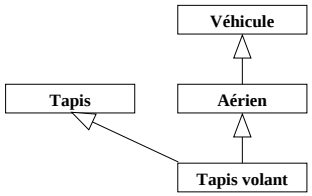
- Définition : mécanisme permettant à une classe fille la spécialisation d’opérations.



- L’exécution de l’opération `mange()` déprendra du type réel de l’objet : `Lion::mange()` pour un `Lion` et `Animal::mange()` pour un `Humain`.

Généralisation : multiple

- Définition : mécanisme permettant à une classe d’avoir plusieurs classes mères.
- Les superclasses n’ont pas forcément d’ancêtres communs.



- `Tapis volant` hérite des propriétés de `Tapis` et de `Aérien`.

Relation de dépendance

- Définition : relation unidirectionnelle indiquant qu'un changement dans la cible (fournisseur) provoque un changement dans la source (client).
- Relation non structurelle existant entre plusieurs éléments.
- Représentation UML : une flèche pointillé éventuellement stéréotypée.
- Quatre types de relations de dépendances :

  Abstraction: relation entre éléments qui représentent un même concept à différents niveaux d'abstraction ou selon des points de vue distincts;

  Liaison: dépendance entre une classe paramètrable (cible) et une classe paramètrée (source);

  Permission: l'élément source a le droit d'accéder à l'espace de nommage de l'élément cible;

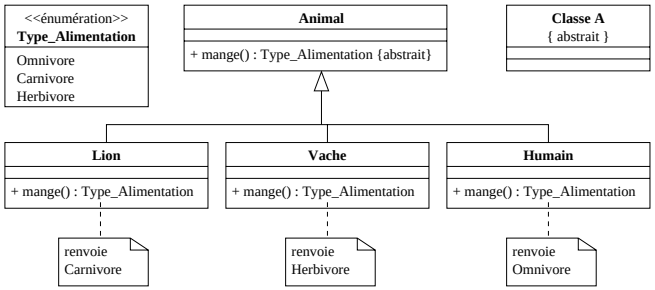
  Utilisation: l'élément source requiert la présence d'un élément cible.

Relation de dépendance (suite)

- Stéréotypes prédéfinis dans UML pour les relations de dépendance :
  - **Liaison** : ≪`lie`≫
  - **Utilisation** : ≪`utilise`≫, ≪`appelle`≫ (une opération source appelle une autre opération cible), ≪`crée`≫ (une instance de la classe source crée une instance de la classe cible), ≪`instancie`≫ (une opération de l'élément source crée une instance de l'élément cible), ≪`instance de`≫ (l'élémént source est une instance de l'élément cible).
  - **Permission** : ≪`ami`≫ (permet à l'élément source d'ignorer la propriété de visibilité de l'élément cible),
  - **Abstraction** : ≪`dérive`≫ (un élément source est défini ou calculé à partir d'un élément cible), ≪`raffine`≫ relation de dépendance entre deux éléments à des niveaux sémantiques différents, ≪`réalise`≫ : l'élément source implémente la spécification désignée par l'élément cible.

Classe abstraite

- Définition : classe non instanciable définissant au moins un mécanisme général instanciable par des classes filles.
- Représentation UML : définition de la classe avec la propriété {`abstrait`} ou si une de ses opérations (héritée ou non) possède la propriété {`abstrait = vrai`}.



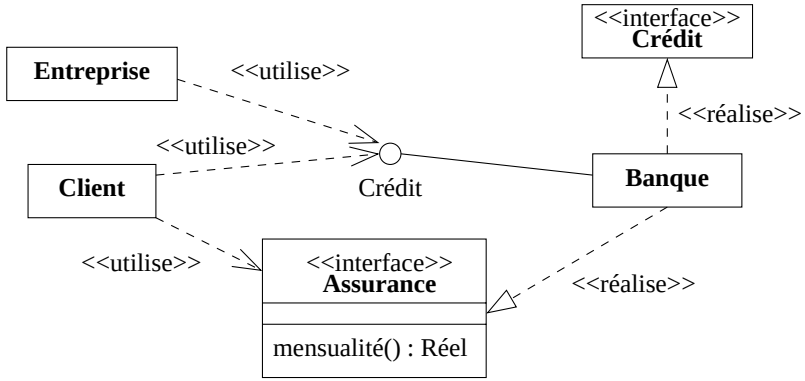
Classe abstraite (suite)

- On définit une classe abstraite lorsque :
  - il est impossible de connaître l'implantation d'une méthode (abstractisation d'une opération).
    Exemple : l'opération `manger()` de la classe `Animal` n'a pas de sens sémantique propre; il s'agit d'une propriété existant chez tous les animaux; il est impossible de préciser le comportement de ce service au niveau de la classe `Animal`.
  - l'instanciation d'une classe n'a aucun sens sémantique ou pratique.

Interface

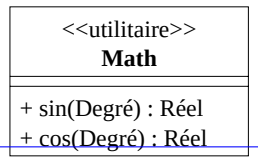
- Définition : description d'un ensemble d'opérations utilisées pour spécifier un service offert par une classe.
- Ne contient ni attribut, ni association, ni implémentation des opérations (les opérations sont abstraites).
- Une classe réalisant une interface doit :
  - soit implémenter les opérations de l'interface,
  - soit définir les opérations de l'interface comme des opérations abstraites (implémentables par les classes filles).
- Représentation UML :
  - classe ayant le stéréotype ≪`interface`≫, ou par un cercle pour faire référence à l'interface utilisée dans la classe,
  - flèche d'héritage en pointillés pour la réalisation d'une interface par une classe,
  - flèche de dépendance en pointillés pour l'utilisation.

Interface (suite)



Classe utilitaire

- Définition : définition d'une classe dont tous les membres ont une portée de classe ⇝ les attributs et les opérations deviennent des variables et des procédures globales.
- Une classe utilitaire ne peut pas être instanciée.
- *En Java, une classe utilitaire correspond à une classe qui ne contient que des membres statiques.*
- Représentation UML : classe ayant le stéréotype ≪`utilitaire`≫.



Pour en savoir plus ...

"Modélisation objet avec UML". Muller, Gaertner. 2000. Éditions Eyrolles (http://www.editions-eyrolles.com/).

"Le guide de l'utilisateur UML". Booch. 2000. Éditions Eyrolles.

"UML en action – De l'analyse des besoins à la conception en Java". Roques. 2000. Éditions Eyrolles.

"Intégrer UML dans vos projets". Lopez. 1997. Éditions Eyrolles.