

Le diagramme de classes

- Utilité
- Classe : notions, attributs et opérations, opérations réalisées, représentation
- Encapsulation
- Association : définition, propriétés
- Dépendance
- Agrégation et composition
- Héritage : hiérarchie, généralisation, spécialisation, propriétés, polymorphisme
- Classe abstraite
- Comment construire le diagramme de classes

Utilité

- Représentation graphique de la vue de conception statique d'un système (modèle de données).
- Ce diagramme représente les classes avec les attributs et méthodes ainsi que les généralisations et les associations entre classes.

Classe - notions

- Si des objets présentent des caractéristiques communes (attributs, méthodes) facilement identifiables, il est intéressant de les modéliser sous la forme de classes.
- Cette démarche est appelée l'abstraction (terme aussi utilisé pour la création d'une hiérarchie).
- Une classe définit donc la structure (attributs), le comportement (méthodes) et les relations d'une série d'objets.
- Une classe peut être considérée comme un « moule » à partir duquel les objets sont créés.
- La création d'objets s'appelle donc l'instanciation (un objet est une instance de classe).

Classe - attributs et opérations

- Les attributs correspondent aux propriétés de la classe et comprennent :
 - un nom ;
 - un type ;
 - éventuellement une valeur initiale.
- La spécification du comportement d'une classe est définie par des opérations. La réalisation de ces opérations se fait au moyen de méthodes.
 - Opération = service offert par les instances de la classe (objets).
 - Méthode = implémentation d'une opération.

Attribut et méthode de classe

- Un attribut est de classe si sa valeur est partagée par toutes les instances de la classe (une seule valeur pour tous les objets créés à partir de la classe).
- Une méthode de classe est une méthode qui rend un service indépendamment des objets qui pourraient être créés. Elle ne peut manipuler que des attributs de classe.
- Dans la représentation UML d'une classe, les attributs et méthodes de classe sont soulignés.



Classe - opérations réalisées

- les constructeurs qui créent les objets ;
- les destructeurs qui détruisent les objets ;
- les accesseurs ou sélecteurs (ou 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 qui contient plusieurs objets.

Classe - représentation

- 3 zones: nom, attributs et méthodes.
- Première lettre du nom en majuscule.

Automobile
- marque : String - modèle : String - carburant : String - cylindrée : int - turbo : boolean - chassis : String
+ démarrer() : void + arrêter() : void + tourner() : void + faireLePlein() : void

Classe documentée

Automobile

- marque : String
- modèle : String
- carburant : String
- cylindrée : int
- turbo : boolean = false
- chassis : String

- + démarrer() : void
- + arrêter() : void
- + tourner() : void
- + faireLePlein() : void

Classe non documentée

Automobile

Encapsulation

- Certains attributs et certaines méthodes ont pour seul objectif des traitements internes à l'objet.
- Il n'est pas nécessaire de les rendre visibles de l'extérieur.
- Il est même souvent utile de les masquer vis-à-vis de l'extérieur pour conserver leur intégrité.
- L'encapsulation est un mécanisme consistant à rassembler les données et les méthodes au sein d'une structure en cachant l'implémentation de l'objet, c'est-à-dire, en empêchant l'accès aux données par un autre moyen que les services proposés.



- Pour réaliser l'encapsulation, les attributs doivent être privés et les méthodes qui permettent de changer leur valeur doivent être publiques.
- Il y a trois niveaux de visibilité des éléments d'une classe :
 - public (+) ;
 - protégé (#) ;
 - privé (-).

Automobile

- marque : String
- modèle : String
- carburant : String
- cylindrée : int
- turbo : boolean = false
- chassis : String

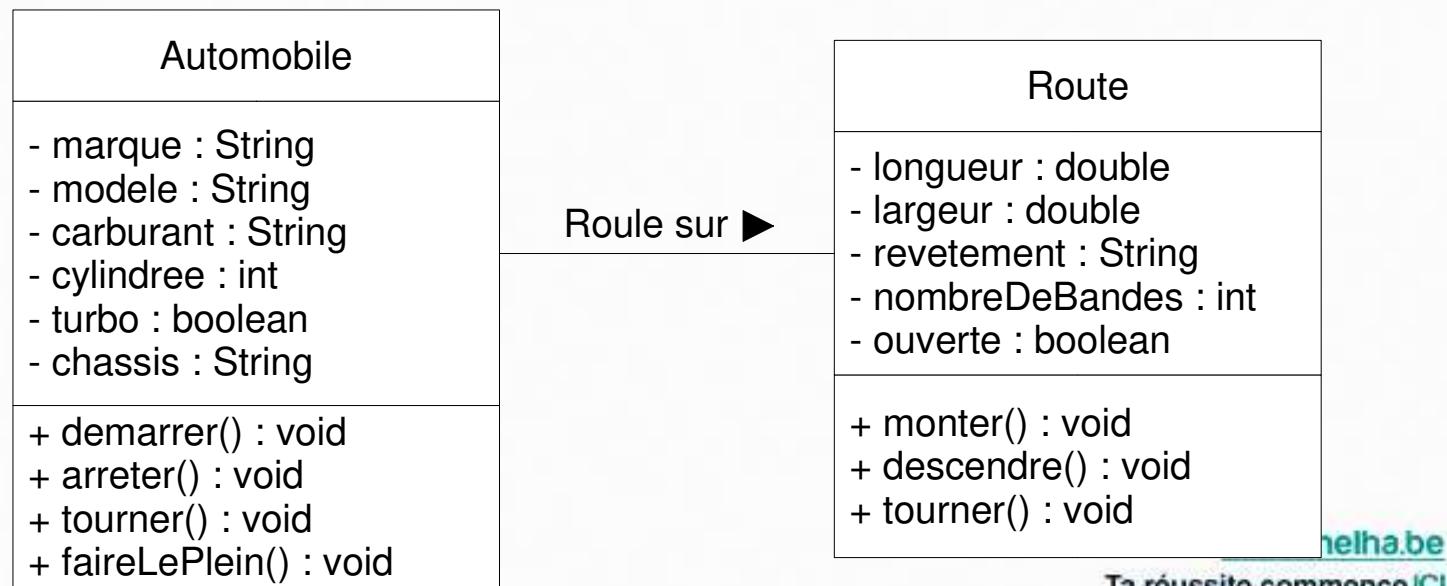
- + démarrer() : void
- + arrêter() : void
- + tourner() : void
- + faireLePlein() : void
- + accéler () : void
- chargerBatterie() : void
- embrayer() : void
- débrayer() : void
- changerVitesse() : void

Association - définitions

- Une association représente une relation entre classes.
- Elle est permanente dès qu'elle a été établie (contrairement à une dépendance).
- Une association dispose de plusieurs propriétés détaillées par après.

Association - propriété (1)

- Les associations peuvent être nommées (pas obligatoire), le sens de la lecture étant indiqué par une flèche.



Association - propriété (2)

- Dans les associations, les classes jouent des rôles spécifiques.
- Il est possible de spécifier ces rôles.
- Exemple: dans une relation de travail, la personne joue le rôle d'employé, l'entreprise joue le rôle d'employeur.



Association - propriété (3)

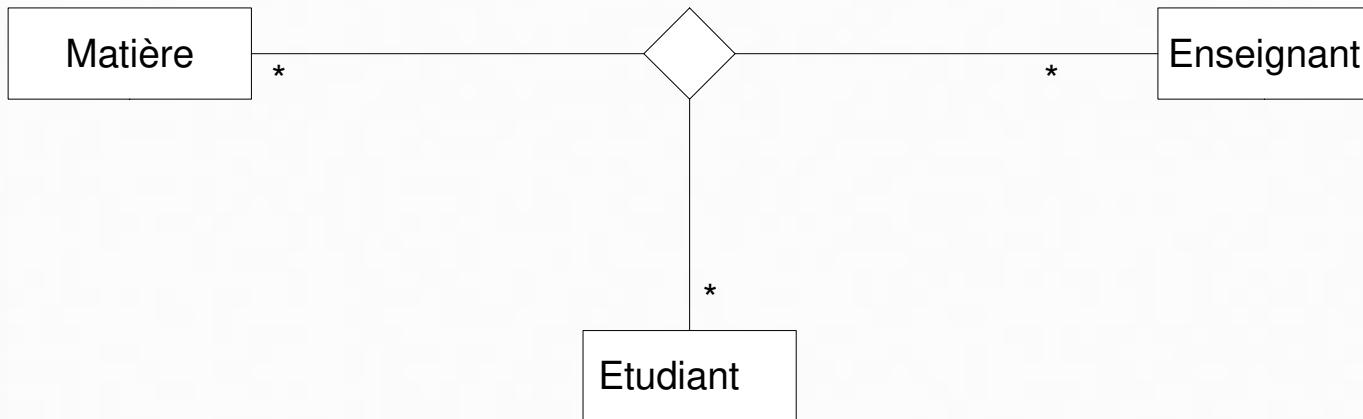
- La multiplicité spécifie le nombre d'objets que connaît un objet particulier.
- On parle aussi de *cardinalité*.
- Exemple: « *un pc équipé d'une interface IDE peut contenir jusqu'à 4 disques, un disque ne peut être connecté que sur un seul pc* » :



- 1 exactement 1
- 0..1 de 0 à 1
- * de 0 à l'infini
- 3..* de 3 à l'infini
- 0..2 de 0 à 2
- 2 exactement 2

Association - propriété (4)

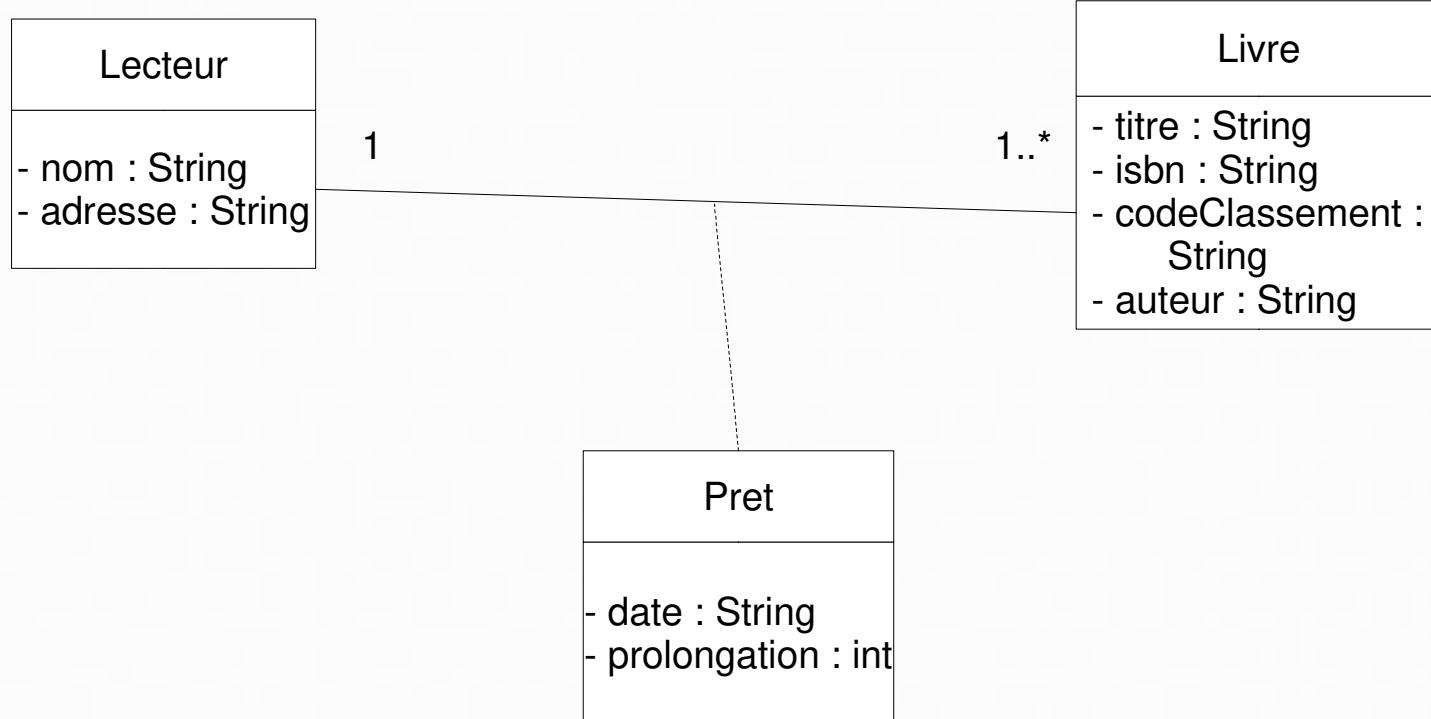
- Les associations peuvent mettre en présence plus de deux classes.
- On parle d'association n-aire.



Association - propriété (5)

- Une association peut bénéficier des propriétés d'une classe. Elle peut posséder:
 - des attributs ;
 - des opérations ;
 - des associations vers d'autres classes.
- On parle alors de classe d'association.
- Notation: relier la classe d'association à l'association au moyen d'un trait discontinu.
- Exemple: « *Un lecteur emprunte un livre à l'occasion d'un prêt. Ce prêt est caractérisé par sa date et le fait d'être prolongé ou non* ».

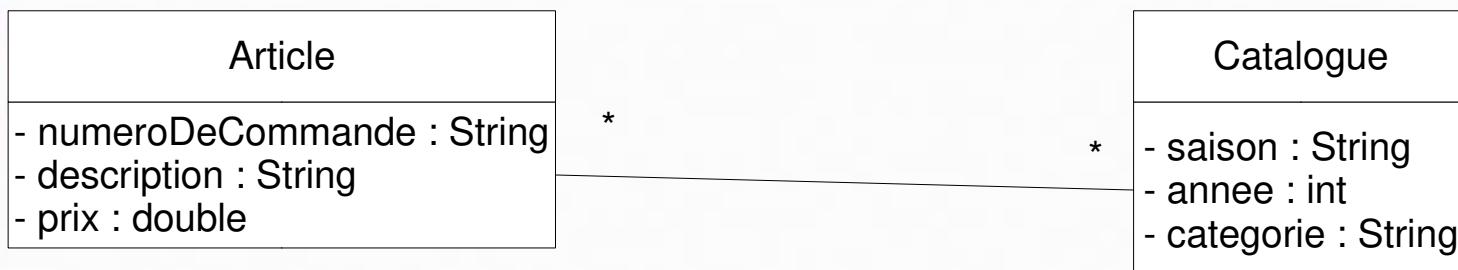




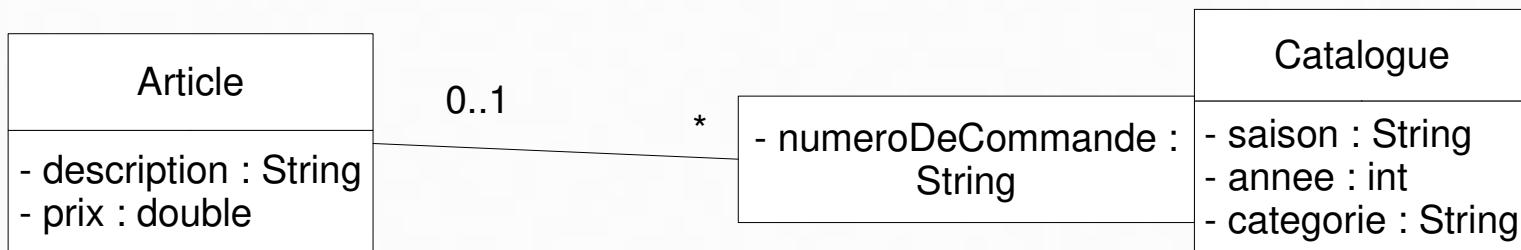
Association - propriété (6)

- La qualification d'une instance est une valeur ou un ensemble de valeurs qui permettent de retrouver une instance.
- A l'aide d'une qualification, une cardinalité (multiplicité) maximale non finie peut être ramenée à une cardinalité (multiplicité) maximale finie.
- Exemple: « *il est possible de retrouver un article dans un catalogue grâce à son numéro de commande* ».

Sans qualification

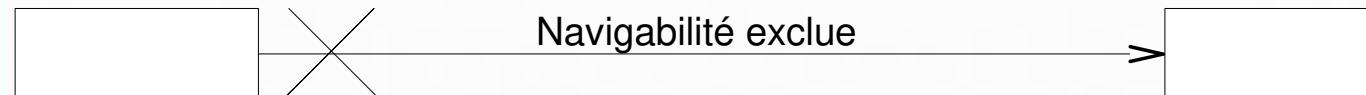
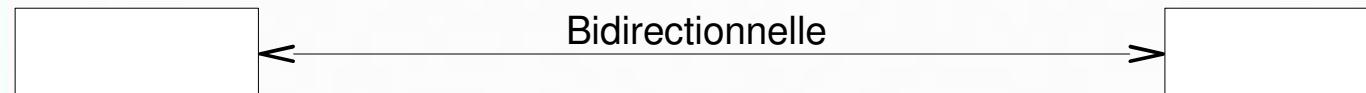
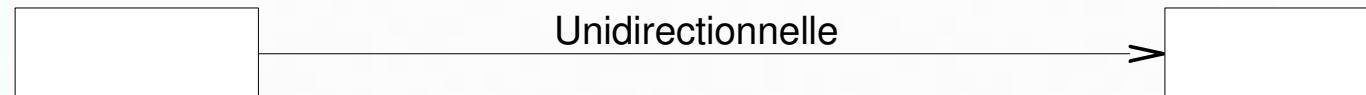
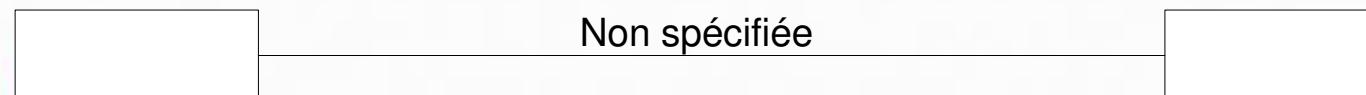


Avec qualification



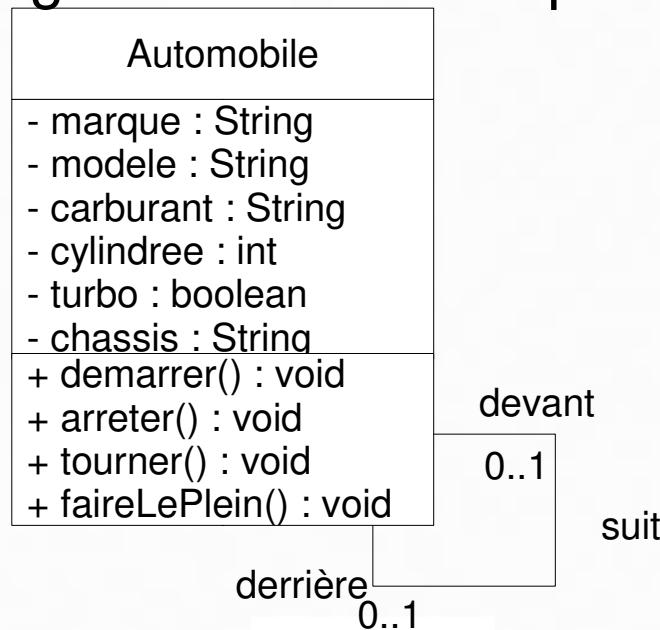
Association - propriété (7)

- Dans certains cas, il peut être possible de retrouver les informations d'une classe à partir d'une autre classe (via l'envoi de message).
- Dans d'autres cas ce n'est pas possible.
- Cet aspect de l'association est assuré par la notion de navigabilité.
- Représentation : flèche ajoutée à l'association dans le sens de la navigabilité.



Association - propriété (8)

- Une association peut également être réflexive (porte sur des objets de même type). Les rôles doivent obligatoirement être spécifiés.



- Cette association réflexive donne un lien tel que celui-ci.

<u>voiture1 : Automobile</u>
marque = "Volkswagen" modele = "Polo" carburant = "Diesel" cylindree = 1850 turbo = true chassis = "DX1232456852P"

suit

<u>voiture2 : Automobile</u>
marque = "Renault" modele = "Clio" carburant = "Diesel" cylindree = 1900 turbo = false chassis = "SX1232456852P"

Association - propriété (9)

- Les contraintes permettent d'ajouter de nouvelles règles sémantiques à celles qui existent.
- Elles précisent les conditions indispensables pour que le modèle soit correctement formé.
- Elles peuvent être écrites sous forme de texte informel bien qu'il existe un langage formel (OCL pour Object Constraint Language).
- Représentation: chaîne de caractères entourée d'accolades.

{calcul modulo division par 97 exact}

CompteBancaire

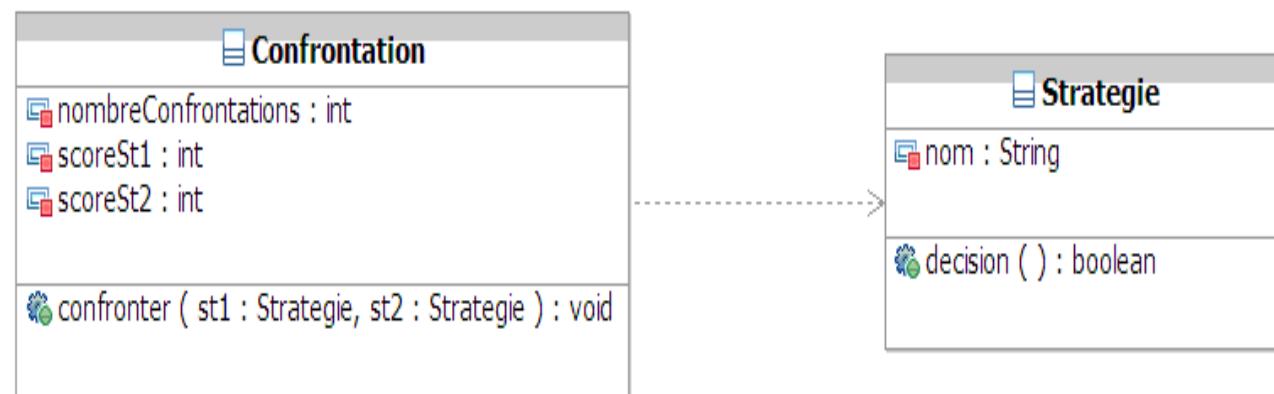
Societe

{XOR}

Personne

Dépendance

- Relation d'utilisation qui établit qu'un changement de spécification d'un élément peut en influencer un autre.
- L'inverse n'est pas nécessairement vrai.
- Indique qu'une classe en utilise une autre comme argument dans la signature d'une méthode.
- La durée de vie d'une telle relation est courte (durée d'exécution de la méthode).
- Représentation: flèche en pointillé dirigée vers l'élément en regard duquel existe la dépendance. Pas nécessairement de nom.



Agrégation et composition

- Les associations décrivent des relations simples entre des classes. Or, il est parfois intéressant de modéliser des associations plus fortes.
- Dans certains cas, une des deux classes joue un rôle plus important que l'autre. C'est le cas lorsqu'il faut modéliser une association « tout/partie ».
- Une classe représente un élément plus grand (le « tout » ou composite) et est constituée d'autres classes plus petites (les « parties » ou composants).
- Il faut établir qu'une classe est composée d'autres classes ou qu'une classe fait partie d'une autre.
- Ce type d'association est représenté par une agrégation ou une composition.

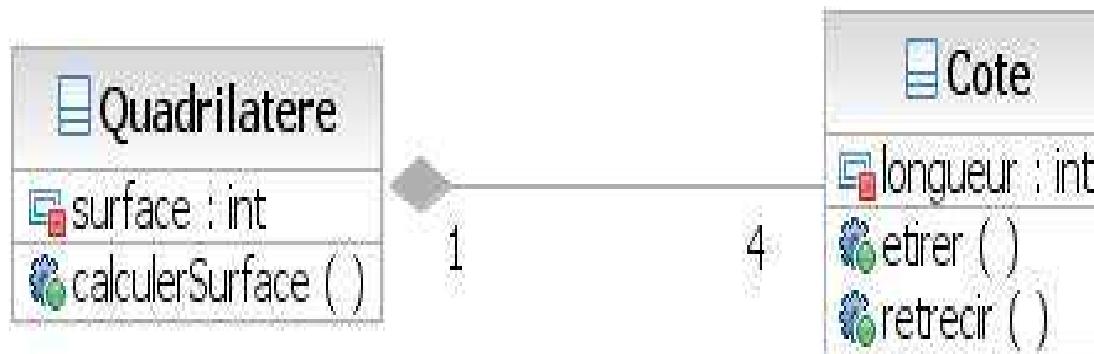
Composition - définition

- A un moment donné, une instance du composant ne peut être liée qu'à un seul composite (la valeur maximale de la multiplicité du côté du composite ne peut donc excéder 1).
- Le « tout » est responsable de la création des parties (et donc de leur stockage en mémoire).
- Si l'agrégat est détruit, les composants le sont également.
- L'agrégat et les composants ont ici la même durée de vie.

Composition - représentation

- Représentation:

Trait continu avec un losange noir du côté de la classe qui joue le plus grand rôle (le « tout »)



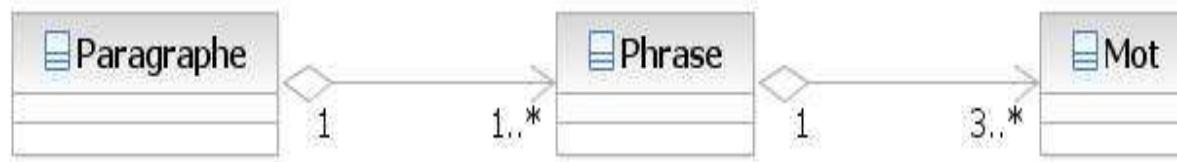
Agrégation - définition

- Le composite ne dispose que d'une copie des composants. Autrement dit, la destruction du composite n'entraîne pas celle des composants.
- Un composant peut donc faire partie de plusieurs composites.
- L'agrégation est aussi appelée composition faible.

Agrégation - représentation

- Représentation:

Trait continu avec un losange blanc du côté de la classe qui joue le plus grand rôle (le « tout »)



Hiérarchie de classes

- Dans la réalité, il peut arriver que des classes différentes aient des éléments communs (attributs, méthodes et contraintes).
- Point de vue programmation:
 - Inutile de programmer plusieurs fois des classes qui ont ces éléments en commun.
 - Considérer ces classes comme faisant partie d'une même « famille ».
 - Programmation unique de ces attributs, méthodes et contraintes.
 - Répercussion des éléments programmés sur toutes les classes d'une même famille (réutilisation).
 - Utilisation de mécanismes qui permettent d'adapter ces éléments si nécessaire (extension).



- Il est donc intéressant de modéliser des classes comme sous-ensembles d'autres classes.
- Nous considérons donc ici les sous-classes comme appartenant à une même famille, matérialisée par une super-classe.
- Ces mécanismes sont assurés par la généralisation et la spécialisation.
- Des relations entre des classes et des sous-classes constituent une hiérarchie de classes.

Généralisation

- = relation entre un élément général et un élément dérivé de celui-ci mais plus spécifique.
- Cas où on peut dire « *est un type de ...* », « *est une sorte de ...* », « *est un(e) ...* ».
- Termes qui désignent l'élément général
 - Classe générale
 - Super-classe
 - Sur-classe
 - Classe-mère

- Termes qui désignent l'élément spécifique
 - Sous-classe
 - Classe spécialisée
 - Classe-fille
- Exemples
 - « *Cheval* » et « *Chien* » sont des sous-classes de « *Mammifère* », elle-même sous-classe de « *Animal* ».
 - Une « *baie vitrée* » est une sorte de « *fenêtre* ».

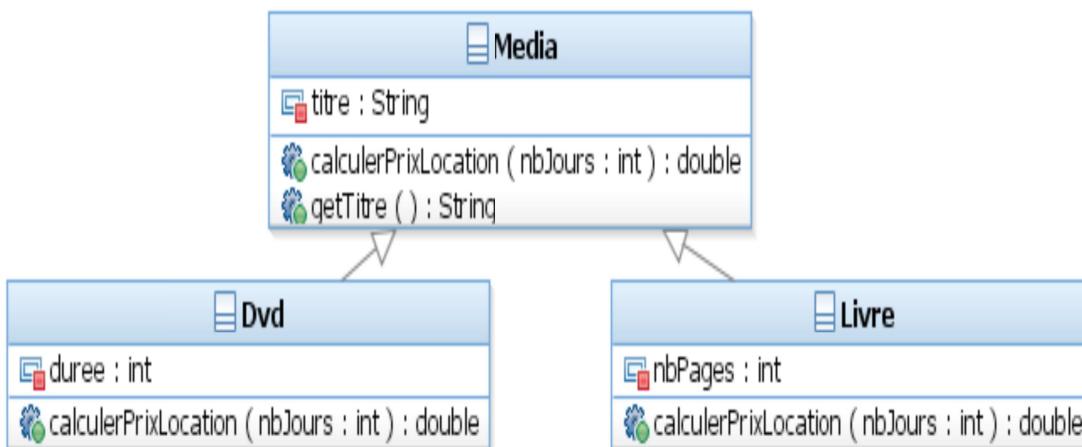
Spécialisation

- = relation par laquelle une classe est une sous-classe d'une autre.
- Par cette relation, elle « hérite » d'attributs, méthodes et contraintes de sa classe-mère.
- Une sous-classe partage donc des attributs et des méthodes avec les sous-classes :
 - de la même famille ;
 - qui proviennent de la même super-classe.

- Par la spécialisation, une sous-classe reste cohérente avec sa classe générale. Elle peut utiliser toutes les méthodes et les attributs définis dans la classe générale.
- De plus :
 - Il peut y avoir apport de nouveaux attributs et méthodes.
 - Des méthodes peuvent être redéfinies par rapport à la classe-mère.

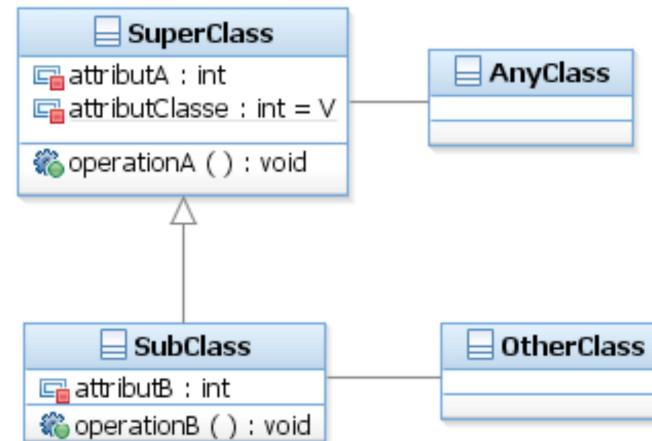
- Propriété qui fait bénéficier à une sous-classe de la structure et du comportement de sa super-classe.
- Conséquence de la spécialisation.
- Représentation:
 - Trait continu avec un triangle du côté de la super-classe.
 - Base du triangle vers la sous-classe.
 - Sommet du triangle vers la super-classe.

- Les méthodes et attributs hérités:
 - sont représentés dans la super-classe ;
 - ne sont pas reproduits dans les sous-classes à moins qu'ils ne soient redéfinis.

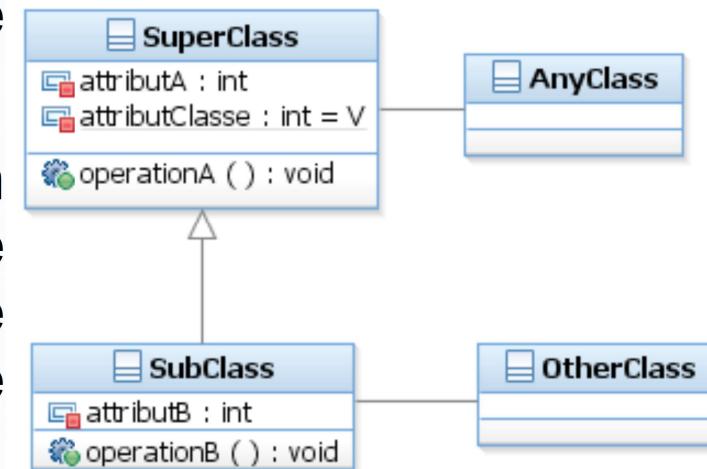


Héritage: caractéristiques

- Si tous les objets de SuperClass possèdent un attributA, alors tous les objets de SubClass le possèdent également.
- La valeur de attributA n'est, elle, pas héritée.
- Toutes les méthodes utilisées sur les objets de SuperClass sont également utilisables sur les objets de SubClass.

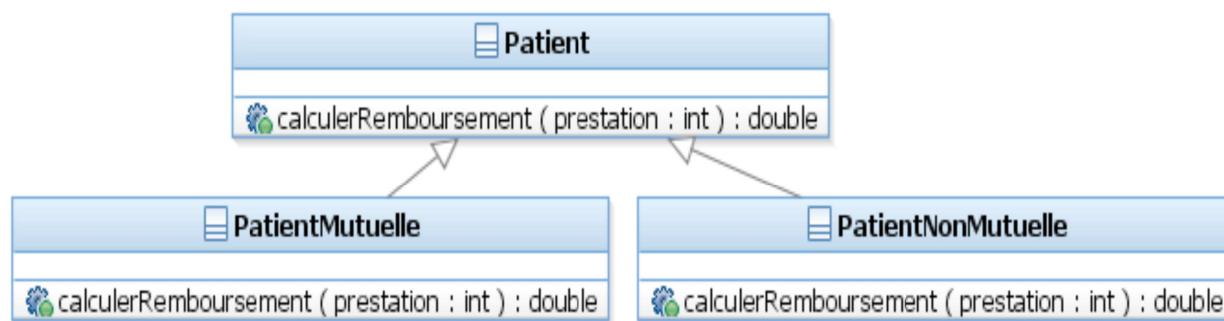


- Si SuperClass possède un attribut de classe avec la valeur V, alors SubClass possède cet attribut de classe avec la valeur V.
- S'il existe une association entre SuperClass et une classe AnyClass, cette association est transmise à SubClass.
- On peut utiliser operationA() et operationB() sur les objets de SubClass.



Polymorphisme

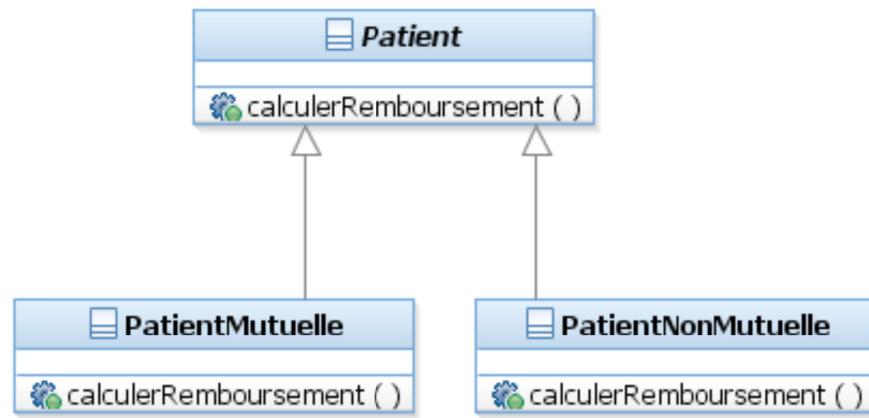
- Des méthodes peuvent produire différents effets ou se dérouler différemment en fonction des sous-classes (=comportement différent).
- Lorsque la méthode d'une sous-classe possède la même signature et le même type de retour que celle d'une super-classe, c'est celle de la sous-classe qui prévaut (redéfinition de méthode).
- Exemple: *le calcul du remboursement d'une prestation dépend du fait qu'un patient soit affilié ou non à une mutuelle.*



Classe abstraites et concrètes

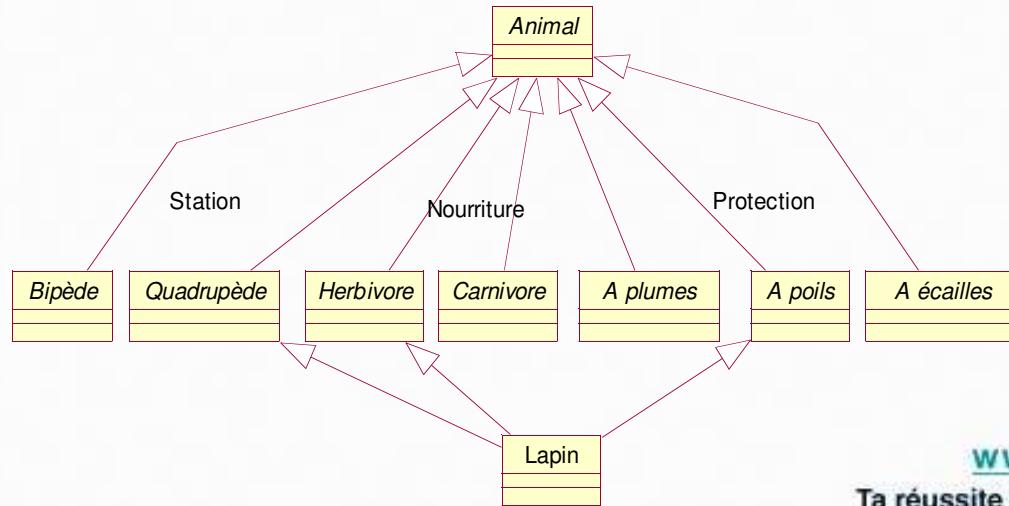
- Une classe abstraite n'a pas directement d'instances. Ce sont les sous-classes qui possèdent des instances.
- La classe abstraite met à la disposition de ses sous-classes des opérations et des propriétés communes.
- A l'opposé, une classe concrète est obligatoirement instanciée.

- Notation
 - Une classe abstraite se représente en caractères *italiques*.



Héritage multiple

- Principe par lequel une classe-fille peut avoir plusieurs classes-mères.
- Ce principe n'est pas supporté par tous les langages orientés objet (Java,...).



2019-2020 - HELHa - Mons - A&CP

www.helha.be

Ta réussite commence ICI



Modélisation de l'héritage

- Déetecter les classes similaires d'un point de vue structurel et comportemental.
- Abstraction: extraire toutes les caractéristiques comportementales et structurelles communes et les placer dans des classes plus générales dont les classes spécialisées héritent.

Comment construire un diagramme de classes

- Sélectionnez les classes.
- Identifiez les relations entre les différentes classes. N'oubliez pas que les associations entre les classes représentent une faculté d'obtenir des informations ou des services d'une autre classe.
- Identifiez les attributs (propriétés mesurables).
- Identifiez les actions réalisées par les classes.
- Regroupez les classes partageant des caractéristiques communes en hiérarchies.

Comment sélectionner des classes ?

- Une classe est porteuse de propriétés, assume des responsabilités fonctionnelles et peut changer d'état.
- Identifiez tous les concepts représentant une définition de structure statique.
- Supprimez tous les concepts redondants.
- Supprimez toutes les classes qui ne font pas référence au métier mais à un choix de réalisation.
- Tout ce qui est quantifiable sera représenté par un attribut.

Pièges à éviter

- Ne représentez pas d'acteur du système dans votre diagramme de classes (sauf s'il est nécessaire de disposer d'informations le concernant).
- Les associations entre les classes sont durables, n'utilisez donc pas d'association pour représenter une relation instantanée entre 2 classes.
- Ne placez pas d'association qui puisse être retrouvée par navigation.
- N'oubliez pas que les cardinalités doivent être valables à tout moment pour les objets créés (ex : différence entre une cardinalité minimale égale à 0 ou à 1).

- N'utilisez l'agrégation et la composition que pour montrer qu'une des classes joue un rôle de conteneur par rapport à l'autre classe. De plus, la composition lie la durée de vie des composants à celle du conteneur.
- Ne placez pas d'attributs représentant des listes d'objets (double emploi avec les cardinalités des associations).
- Ne représentez pas dans les attributs d'une classe des propriétés venant d'une autre classe avec laquelle elle est en relation.
- Afin de rendre vos diagrammes plus lisibles, il n'est pas nécessaire de représenter les constructeurs ainsi que les getters et les setters.

Le diagramme d'objets



HELHa

Haute École Louvain
en Hainaut

Plan

- Notion
- Caractéristiques des objets
- Représentation
- Lien

Notion d'objet

- Le principe consiste en une représentation abstraite sous forme d'entités, d'éléments d'une partie du monde réel (le domaine); entités qui ont une existence matérielle (une voiture, un animal) ou virtuelle (la sécurité sociale, le temps, une connexion).
- Représentation:

objet 1

objet 2

objet 3

Caractéristiques fondamentales

- État
 - Valeur à un moment donné de ses attributs (caractéristiques).
 - Liens avec d'autres objets.
 - La structure est l'ensemble des attributs d'un objet.
- Comportement
 - Réactions de l'objet à son environnement.
 - Actions ou opérations réalisées par l'objet.
 - Le comportement des objets sera matérialisé par des méthodes.

- Identité

- Distingue sans ambiguïté tout objet des autres objets, même si l'état, les valeurs d'attributs sont identiques.
- Généralement un attribut particulier permet d'identifier de manière unique un objet exemple: un numéro de moteur, un numéro dans le registre national, un numéro de client...

Conventions de représentation des objets

- Rectangle en 2 parties: pour le nom et les attributs.
- Noms des objets en minuscules.
- 3 possibilités pour les noms d'objets :
 - objet
 - objet:Classe
 - :Classe

- 3 possibilités pour les attributs :
 - attribut : type = valeur
 - attribut = valeur
 - attribut
- Exemple

<u>voiture1 : Automobile</u>
marque = « Volkswagen »
modele = « Polo »
carburant = « Diesel »
cylindree = 1900
turbo = true
chassis = « DX1232456852P »

Liens entre objets

- Les objets sont en relation les uns avec les autres (liens). Ces liens sont des instances des associations qui relient les classes.
- Cette relation se représente par un trait continu.
- Ceci donne un diagramme d'objets.

voiture : Automobile

marque = « Volkswagen »
modele = « Polo »
carburant = « Diesel »
cylindree = 1900
turbo = true
chassis = « DX1232456852P »

pierre : Utilisateur

nom = « Dupont »
age = 38
metier = « cadre »



www.helha.be

Ta réussite commence  ICI