

Diagramme de classes et d'objets

Dame Samb
dcsamb@yahoo.fr
UCAD/FST/DMI

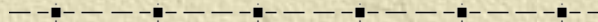


Diagramme de classes (DCL) et d'objets (DOB)

- ✦ Les classes et les objets modélisent les entités matérielles ou immatérielles qui existent dans le système qu'on essaye de décrire.
- ✦ Le DCL fournit une représentation abstraite des objets du système qui doivent interagir pour réaliser les fonctions du système.
- ✦ Le DCL montre la structure statique du système car on ne tient pas compte du facteur temporel dans le comportement du système.
- ✦ Le DOB est un exemple du DCL qui donne une photographie du système dans le temps.

Diagramme de classes (DCL)

- ✦ Le DCL permet de modéliser les classes du système et leurs relations, indépendamment d'un langage de programmation particulier.
- ✦ Le DCL décrit la vue statique du système en terme de classes et de relations entre elles
- ✦ Pour créer un diagramme de classes, il faut :
 - Identifier et décrire les classes
 - Identifier et décrire les relations qui existent entre ces classes

Représentation d'une classe

✧ Une classe se représente à l'aide d'un rectangle comportant trois compartiments de base

✧ Le nom

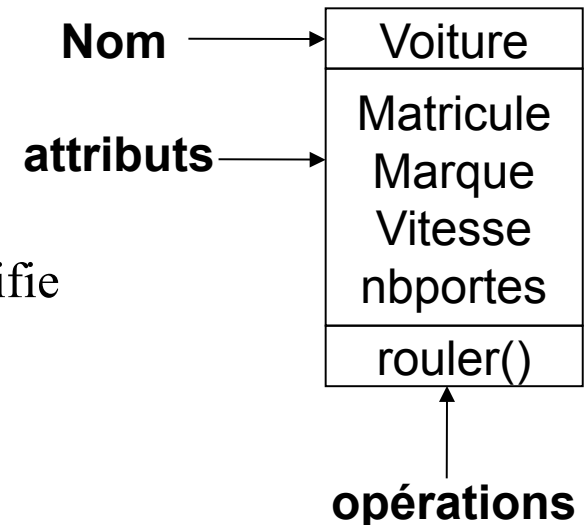
- Doit évoquer le concept qui décrit la classe
- Commence par une majuscule

✧ Les attributs

- Décrit les caractéristiques des objets
- Captent l'information qui décrit et identifie une instance spécifique d'une classe

✧ Les opérations

- décrivent ce qu'une classe peut faire et quels services elle offre aux autres classes.



Attributs d'une classe

- ✦ Ils représentent les données encapsulées dans les objets de cette classe.
- ✦ Chaque donnée est définie par un **nom** (unique dans la classe), un **type**, une **visibilité** et peut être initialisée.
- ✦ La syntaxe de la déclaration d'un attribut est la suivante :
 - `<visibilité> [/] <nom_attribut>:<Type> ['[' <multiplicité> ']' ['{' <propriété> '}']] [= <valeur_par_défaut>]`
- ✦ Peuvent être qualifiés de
 - **Dérivés**: l'attribut est calculé à partir d'autres attributs et de formules de calcul.
 - **Classe**: l'attribut possède une valeur unique et partagée par toutes les instances de la classe

Abonné
+Nom:string
+Prénom:string
+age: int
#login:string
#passwd:string

Propriétés d'un attribut

- ✦ UML définit trois propriétés pour un attribut les attributs :
 - **Variable** : aucune restriction sur les modifications de la valeur de l'attribut
 - **AjoutUniquement** : pour les attributs dont la multiplicité est supérieur à 1, des valeurs additionnels peuvent être ajoutée, mais ne peuvent ensuite être supprimées ou altérées
 - **Gelé** : la valeur de l'attribut ne peut pas être changée après l'initialisation de l'objet

✦ Exemples:

onOff : Bouton
- x : Réel
coord[3] : Réel

+ pi : réel = 3.14 {gelé}
inscrits[2..8] : Personne
/age : Entier

Méthodes d'une classe

- ✧ **Méthode:** fonctionnalité assurée par une classe
- ✧ **Opération:** implémentation d'une méthode. Une opération (même nom et mêmes types de paramètres) doit être unique dans une classe.
- ✧ La syntaxe de déclaration d'une opération est la suivante :
 - `<visibilité> <nom_méthode> ([<paramètre> [, <paramètre> [, <paramètre> ...]]]) : [<valeur_renvoyé>] [{ <propriétés> }]`
- ✧ Une méthode peut être
 - de **classe**: Elle ne peut manipuler que des attributs de classe et ses propres paramètres.
 - **abstraite**: on connaît son entête mais pas la manière dont elle peut être réalisée (i.e. on connaît sa déclaration mais pas sa définition, voir classe abstraite).

Visibilité des attributs et des opérations

✦ Il existe trois niveaux de visibilité:

- **Public (+)** : attribut ou opération visible par tous.
- **Protégé (#)** : attribut ou opération visible seulement à l'intérieur de la classe et par les sous classe de la classe.
- **Privé (-)** : attribut ou opération visible à l'intérieur de la classe.

✦ Les attributs sont en général inaccessibles tant que les opérations sont accessibles (principe de l'encapsulation)

Relations entre classes

- ✦ Les relations suivantes peuvent exister entre des classes :
- ✦ **L'association simple:** lien entre instances de classes.
- ✦ **La généralisation/spécialisation:** factorisation des propriétés communes à plusieurs classes.
- ✦ **L'agrégation:** lien de type ensemble/élément
- ✦ **La composition:** cas particulier de l'agression avec un couplage fort.

Relations entre classes

✦ **Multiplicité :**

- indique un domaine de valeur pour les cardinalités dans les relations entre classes.
- peut être associée à une terminaison **d'association simple, d'agrégation ou de composition.**

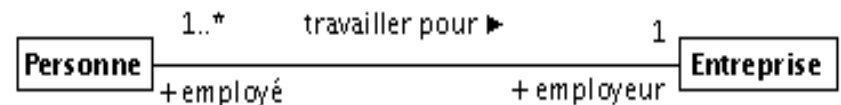
1	Un et un seul
0..1	zéro ou un
n	n (entier naturel)
m..n	De m à n (entier naturel)
*	plusieurs
0..*	De zéro à plusieurs
1..*	De un à plusieurs

Association simple

✧ Une association décrit un groupe de liens ayant une même structure et une même sémantique.

✧ **Formalisme et exemples:**

- **Association binaire**

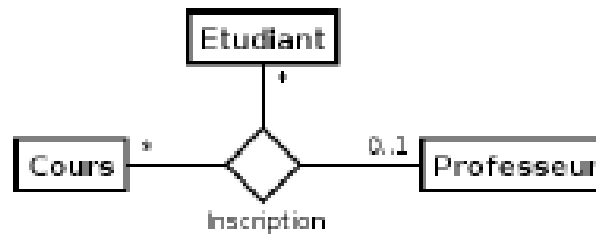


- Une association binaire est matérialisée par un trait plein entre les classes associées.
- Elle peut être ornée d'un **nom**, avec éventuellement une précision du sens de lecture ► ou ◄.
- Le **rôle** tenu par une classe vis-à-vis d'une association peut être précisé.

Association simple

✦ Formalisme et exemples:

- Association ternaire

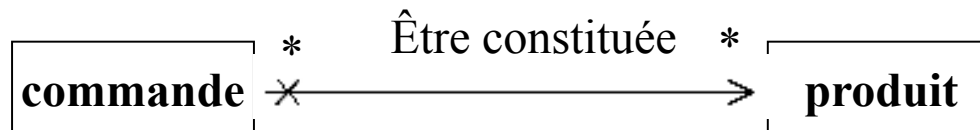


- On représente une association n-aire par un grand losange avec un chemin partant vers chaque classe participante.
- Le nom de l'association, le cas échéant, apparaît à proximité du losange.

Association simple

✦ **Navigabilité:** indique s'il est possible de traverser une association.

– exemple:

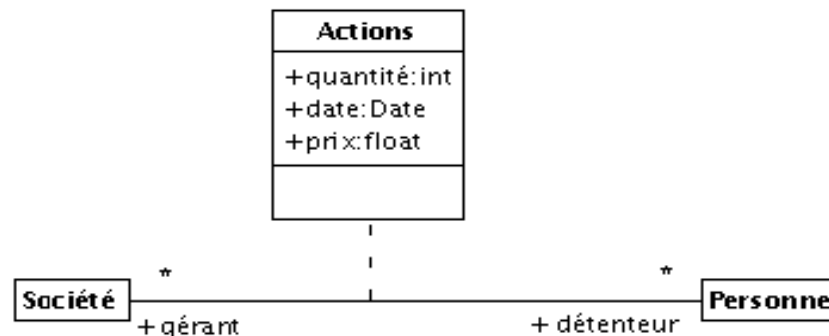


- la terminaison du côté de la classe *Commande* n'est pas navigable : cela signifie que les instances de la classe *Produit* ne sont pas constituées d'une liste d'objets du type *Commande*.
- Inversement, la terminaison du côté de la classe *Produit* est navigable : chaque objet commande est constitué d'une liste de produits.

Association simple

✦ Classe d'association:

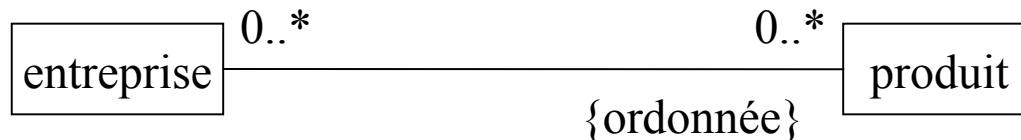
- possède à la fois les caractéristiques d'une association et celles d'une classe.
- se connecte à deux ou plusieurs classes et peut posséder des attributs et des opérations.
- est attachée à l'association avec une ligne en pointillée.
- Exemple:



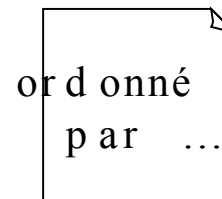
Association simple (contraintes)

✦ Association ordonnée:

- C'est une contrainte qui spécifie que les objets sont ordonnés (selon la clé, le nom, la date, etc.)
- Cette contrainte est spécifiée par le stéréotype *{Ordonnée}* du côté de la classe dont les instances sont ordonnées



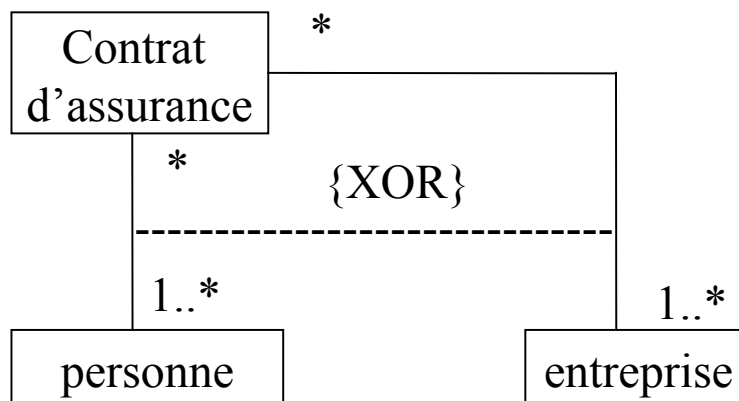
- La manière dont les objets sont ordonnés est spécifiée à travers une note.



Association simple (Contraintes)

✦ Association « ou exclusif »

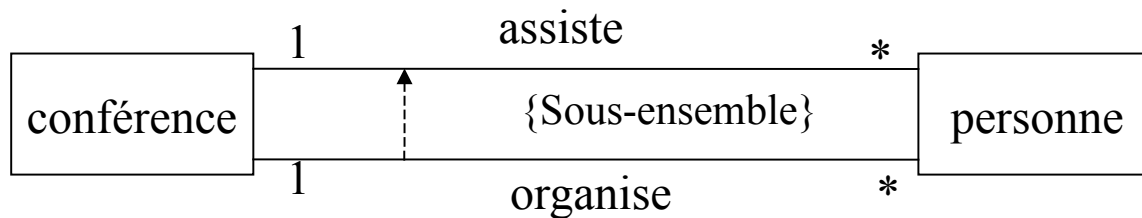
- précise que, pour un objet donné, une seule association parmi un groupe d'associations est valide
- Exemple : Un contrat d'assurance concerne une entreprise ou une personne mais pas les deux en même temps



Association simple (Contraintes)

✦ Association « sous sous-ensemble »

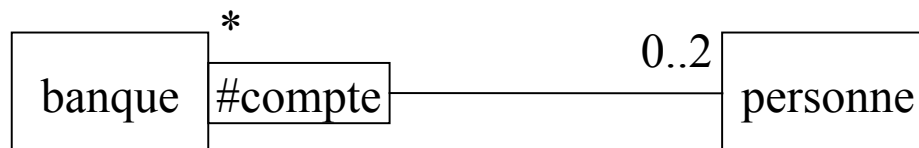
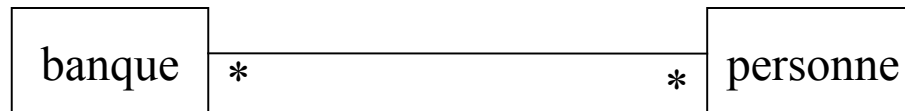
- C'est une contrainte qui indique qu'une collection est incluse dans une autre collection.
- La contrainte est placée à proximité d'une relation de dépendance entre deux associations.
- La flèche de la relation de dépendance indique le sens de la contrainte.



Association simple (Contraintes)

✧ Association qualifiée

- La qualification (ou restriction) d'une association consiste à sélectionner un sous-ensemble d'objets parmi l'ensemble des objets qui participent à une association.
- La qualification est réalisée au moyen d'un ensemble d'attributs particuliers appelé **qualitatif** ou **clé**
- Elle réduit la multiplicité effective du modèle.



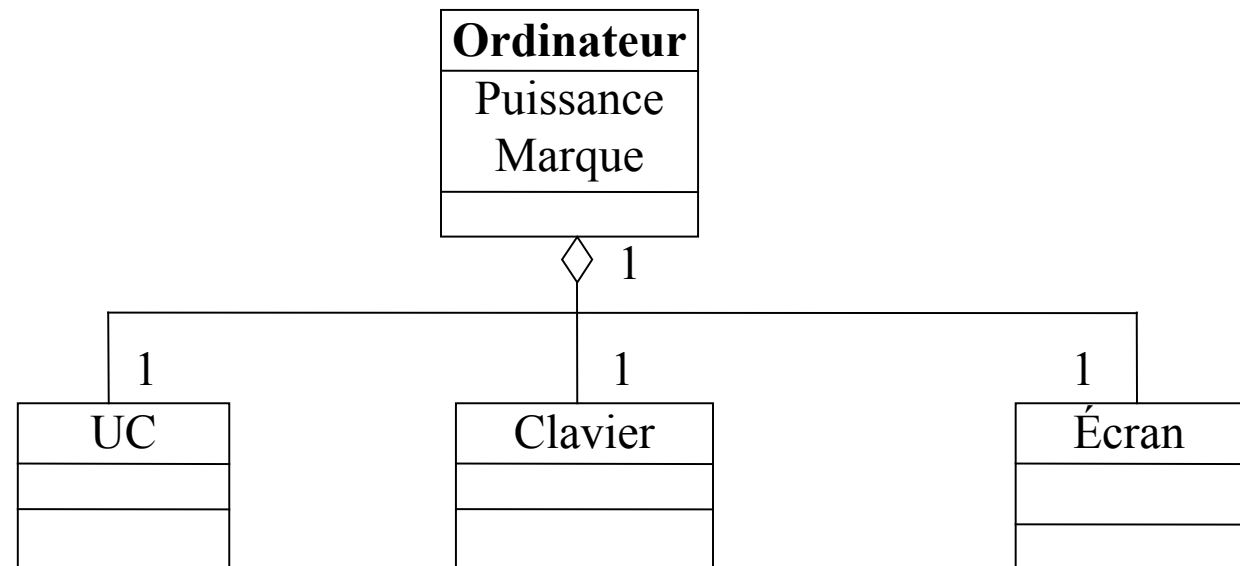
Agrégation

✦ C'est une **association non symétrique** dans laquelle **l'une** des **extrémités** joue un rôle **prédominant** par rapport à **l'autre**.

✦ Les critères suivants impliquent une agrégation :

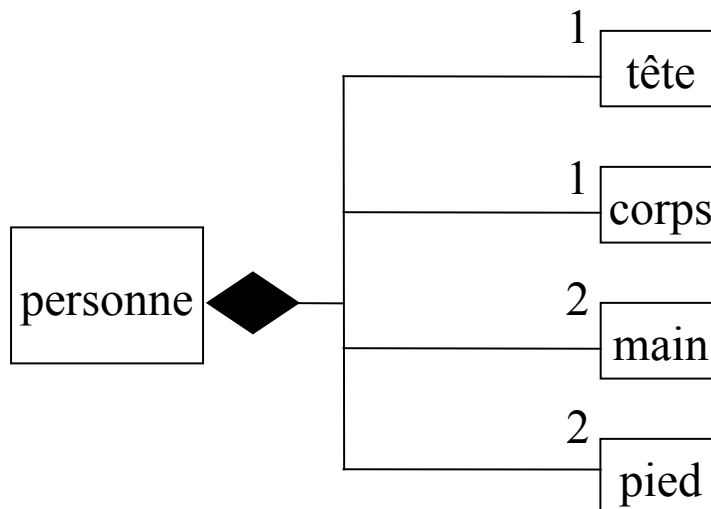
- une classe fait partie d'une autre classe,
- une action sur l'ensemble est appliquée à l'élément,
- les changements d'état sont liés.

✦ Exemple



Composition

- ✦ C'est un cas particulier de l'agrégation avec une contrainte de durée de vie entre la classe composite et la ou les classes composées.
- ✦ La destruction de la classe composite implique automatiquement la destruction de tous ses composants.
- ✦ Elle implique une contrainte sur la valeur de la multiplicité du côté de la classe composite qui doit être 0 ou 1.



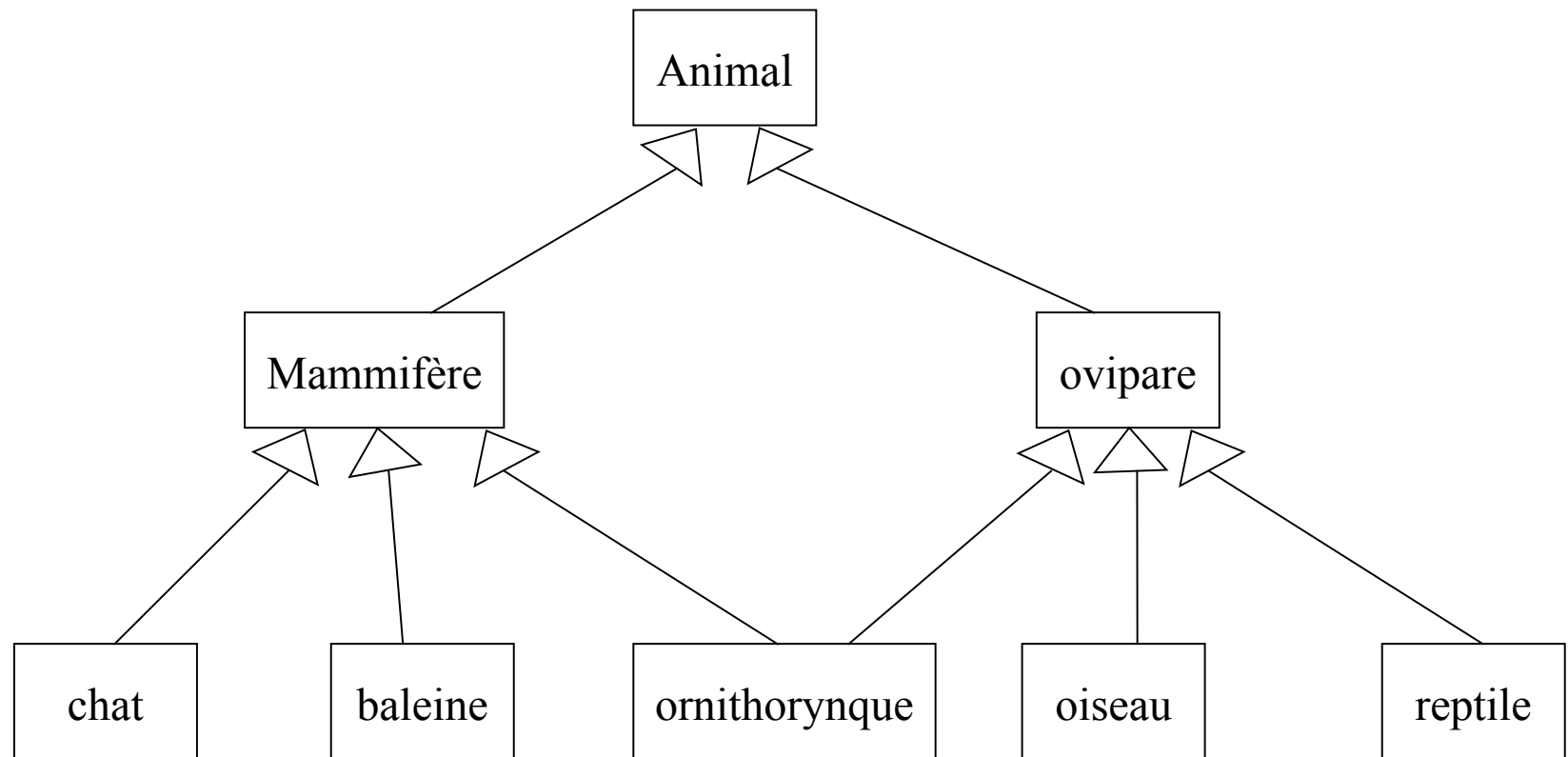
Règles obligatoires pour la composition

- La suppression du composé entraîne la suppression des composants
- Les attributs du composé sont utilisés dans les composants
- Les composants sont des instances du composé
- Un composant ne peut pas être en relation avec d'autres classes externes au composé.

Généralisation et héritage

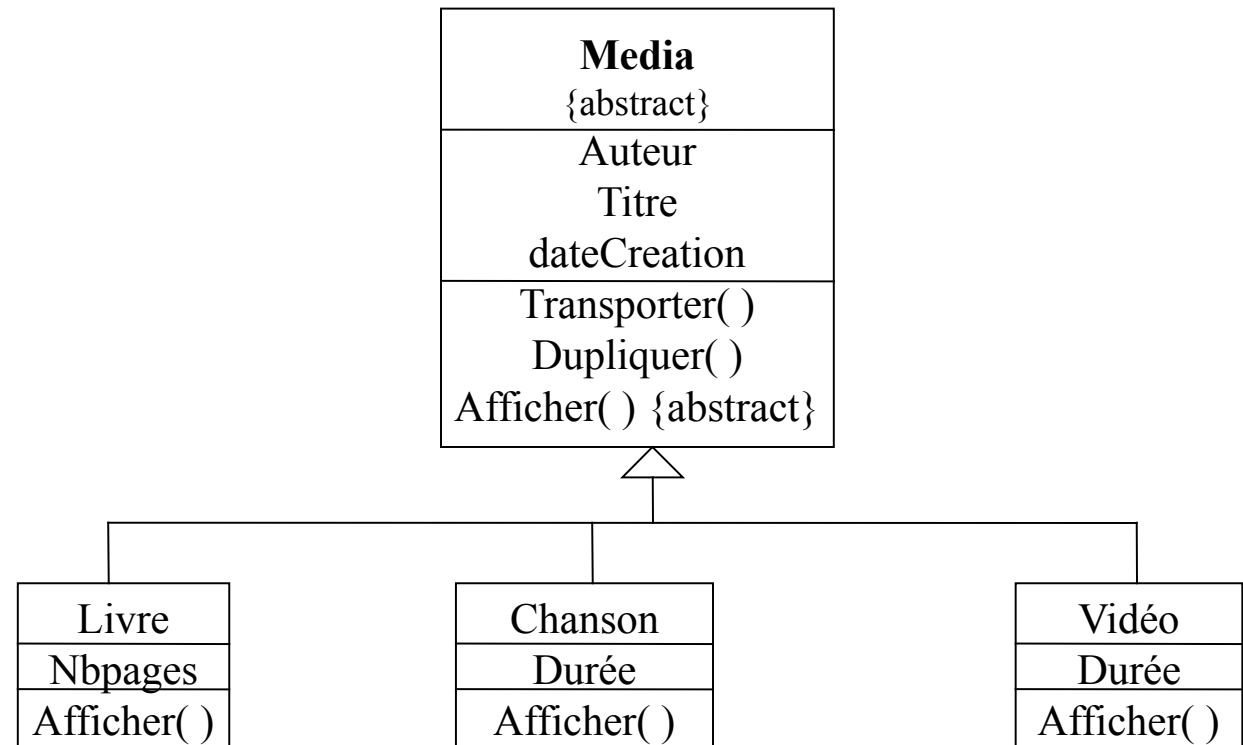
- ✦ La **généralisation** est la relation qui existe entre une classe générale (superclasse) et plusieurs autres classes (sous-classes) plus spécifiques.
- ✦ Les attributs et les opérations d'une superclasse sont transmis aux sous-classes par **héritage**.
- ✦ Une sous-classe possède toutes les propriétés de la superclasse, mais elle ne peut accéder aux **propriétés privées** de celle-ci.
- ✦ Une sous-classe peut redéfinir (même signature) une ou plusieurs méthodes de la superclasse.
- ✦ Toutes les associations de la superclasse s'appliquent aux sous-classes.
- ✦ Une instance d'une sous-classe peut être utilisée partout où une instance de sa superclasse est attendue.
- ✦ Une classe peut avoir plusieurs parents, on parle alors **d'héritage multiple**.

Généralisation et héritage



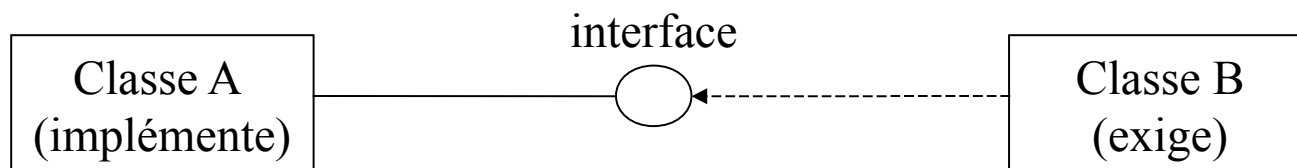
Classe abstraite

✦ Une classe est abstraite si elle dispose d'une méthode abstraite et dans ce cas, elle n'est pas instanciable. Permet de spécifier un comportement commun à plusieurs classes



Interface

- ✦ Elle décrit le comportement visible d'une classe.
- ✦ Elle n'est pas une classe réelle mais une liste de services accessibles par les autres classes.
- ✦ Le comportement visible d'une interface est décrit par des **opérations abstraites** dont la **visibilité est publique**
- ✦ Une interface est représentée par un petit cercle ayant un nom
- ✦ Une classe qui utilise l'interface (implantée par une autre classe) est connectée via une relation de dépendance vers le cercle représentatif de cette interface.



Interface

- ✦ Pour montrer les opérations dans une interface, on la spécifie comme une classe avec le stéréotype «interface».
- ✦ Une interface peut être réalisée par plusieurs classes et une même classe peut réaliser plusieurs interfaces.
- ✦ Exemple:

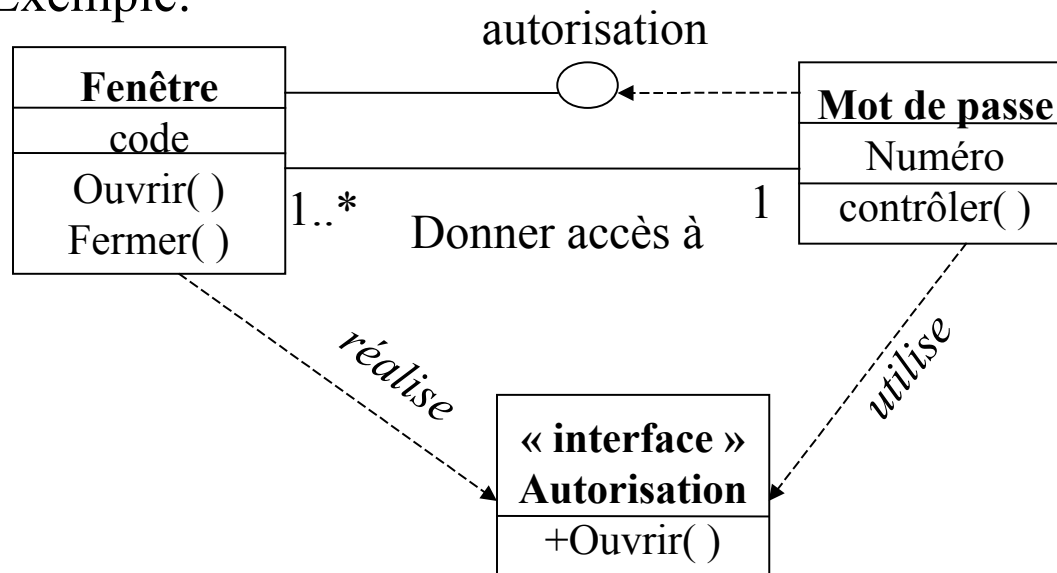


Diagramme d'objets

- ✦ Un DOB est une instance d'un DCL et illustre l'état d'un système à un instant t donnée dans le temps.
- ✦ Les DOB sont utilisées pour montrer un contexte (avant ou après une interaction entre objets par exemple).
- ✦ Un DOB est composé :
 - d'objets (instances de classes),
 - de liens (instances d'associations).
- ✦ Représentation d'un objet
 - Un message vers le groupe d'objet atteint l'ensemble des objets du groupe

Nom de l'objet

Nom de l'objet:classe

:classe

:Personne

Objet anonyme

Groupe d'objets

Diagramme d'objets

- ✧ Dans un DOB, le compartiment des opérations n'est pas utile, mais les attributs doivent recevoir des valeurs.
- ✧ Dans un DOB, les relations du diagramme de classes deviennent des liens.
- ✧ Naturellement, on ne représente pas les multiplicités des extrémités des liens (elles valent toujours 1).

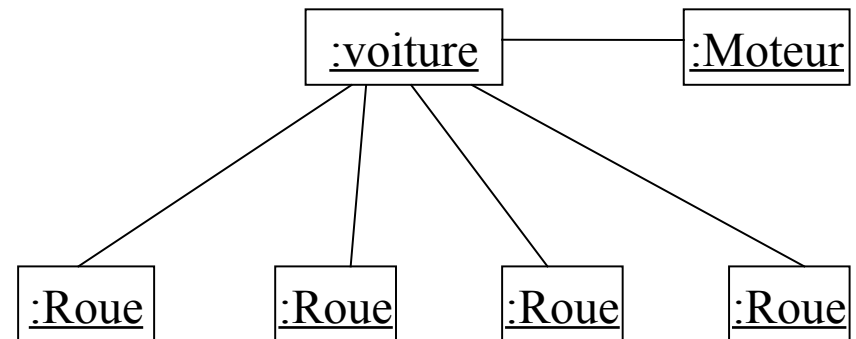
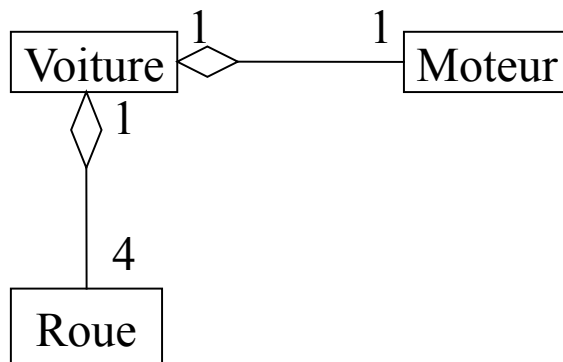
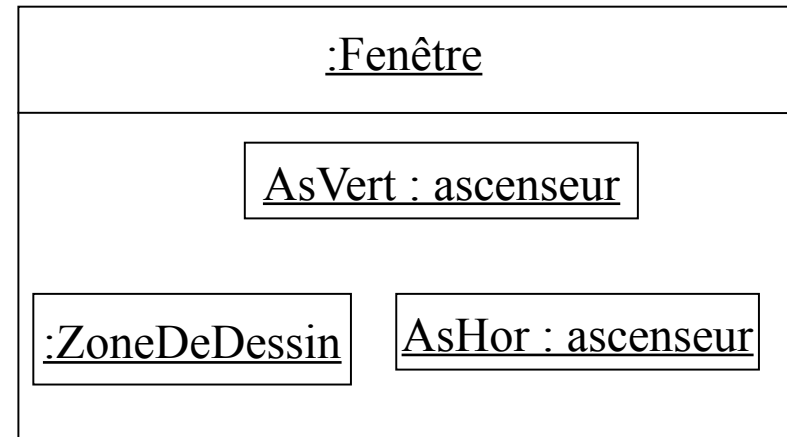
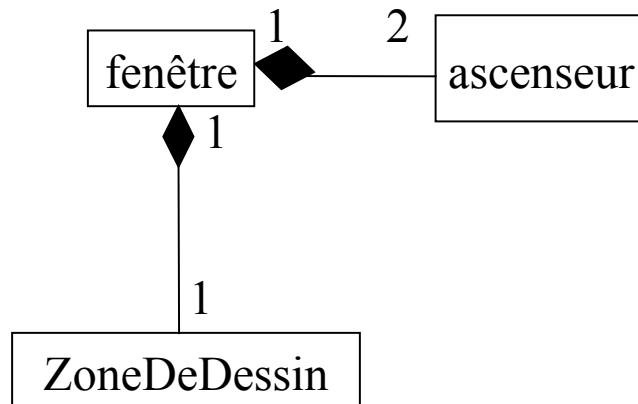


Diagramme d'objets

- ✦ Il est possible de représenter les objets composés de sous objets au moyen d'un objet composite.
- ✦ L'objet composite se présente comme un objet habituel avec différence que les attributs sont remplacés par les sous objets.
- ✦ Exemple



Exercice

- Chaque étudiant du département IF suit un ensemble d'unités d'enseignement (UE).
- Chaque UE a un coefficient et est constituée de cours, de travaux dirigés (TD) et de travaux pratiques (TP).
- Chaque cours, TD ou TP a une date. Les cours sont faits en amphithéâtre, les TD en salle de classe et les TP en salle machine.
- Pour les TP et TD, les étudiants sont répartis dans des groupes. Pour chaque TP, chaque étudiant est en binôme avec un autre étudiant.
- Les cours et les TD sont assurés par un enseignant. Les TP sont assurés par deux enseignants.
- Pour chaque UE, l'étudiant a une note de devoir surveillé ; pour chaque TP, le binôme a une note de TP.