C. Diagrammes de classes

- Le diagramme de classes est le diagramme le plus important en conception objet.
- Il contient essentiellement des classes.

- Il décrit la structure interne du système
  - la partie données : *attributs*
  - La partie fonctions : *opérations*
  - *Les relations entre classes*

- Le diagramme de classes est une *description statique* du système
  - Il ne décrit pas l'utilisation des opérations
  - Il ne décrit pas l'utilisation de données

- 1 La classe
  - une *classe* correspond à un concept
    - C’est un moule (= matrice) à partir duquel seront générés les dynamiquement les objets.
      Un objet généré à partir d'une classe C est une *instance* de C.
  - Une classe est constituée d'un ensemble de
    - *Un nom*
    - *Attribut* (champs, variables d'instances) qui décrivent la structure des objets, on peut les typer, donner la valeur par défaut, la multiplicité…
    - *opérations*, les *méthodes* qui leur sont applicables.

- Le nom de la classe peut spécifiée
  - le paquetage dans lequel il figure Pac1::Pac2::NomClasse
  - Sous forme d'une contrainte { }
    - `auteur`
    - Si elle est en `etat` validée
    - Etc…
- Les propriétés (attribut, opération) ont une visibilité précisé par un modificateurs: +, -, # (public, privé, protégé); la visibilité se limite au paquetage dans lequel est la classe
  un attribut ou méthode de classe est souligné

- une opération peut être signée; la direction des paramètres peut être spécifiée: `in, out, inout`

y2

Classe abstraite

- une *classe abstraite* est définie comme une classe avec <<abstract>>
  - n’a pas d’instance
  - une *méthode abstraite* d'une classe C est une méthode ne devant pas possédée de corps (doit être redéfinie pour tout appel possible).

- Une interface est définie comme une classe avec <<interface>>
    - n’a pas d’instance
    - Toutes ses opérations sont abstraites
    - Ne possède aucun attribut

- Une classe peut avoir autant de compartiments qu'on le souhaite
  - En général
    - le premier est celui qui contient le nom de la classe
    - Le deuxième celui qui contient les attributs
    - Le troisième celui qui contient les opérations
  - On peut donc en créer d'autres: <<exception>>

2 Généralisation

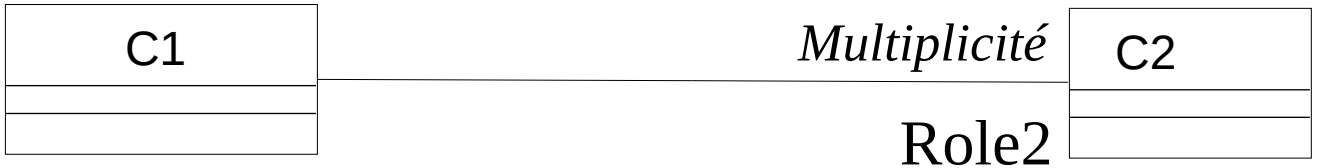
- La relation de classification (généralisation/spécialisation) est une relation qui correspond à un lien *est-une-sorte-de*
  - Elle décrit une relation entre
    - une classe générale (de base, sur-classe, superclasse ou classe parent) et,
    - une classe spécialisée (sous-classe, classe fils, classe enfant).
  - La sous-classe possède les propriétés de sa classe parent (mais ne peut accéder aux propriétés privées de celle-ci)
  - Une classe enfant peut re-définir les méthodes de la classe de base

- Il existe 5 contraintes types sur des classes de même niveau
  - {*incomplete*} : d'autres sous classes de même niveau **peuvent** exister
  - {*complete*} : d'autres sous classes de même niveau **ne peuvent pas** exister
  - {*disjoint*} : une sous classe **ne peut pas hériter** de deux sous classes de même niveau
  - {*Chevauchement*} : **deux sous classes peuvent avoir des instances identiques**
  - Le *discriminant* décrit comment sont identifiés les sous classes par rapport à la classe mère

3 Associations simples

-

- Une *association* [Utile pour le codage]
  - représente une relation sémantique entre des classes: le *trait* +
  - Elle peut posséder un *nom* (en général une forme verbale)
  - Elle peut posséder un/des *rôle(s*) explicitant le rôle d'une classe par rapport à l'autre
  - Elle peut posséder un *sens :* >
  - Elle peut posséder une *multiplicité*
- ***Ex: C1 voit C2 comme Role2; C2 voit C1 comme Role1***
  C1 NomDeLassociation C2

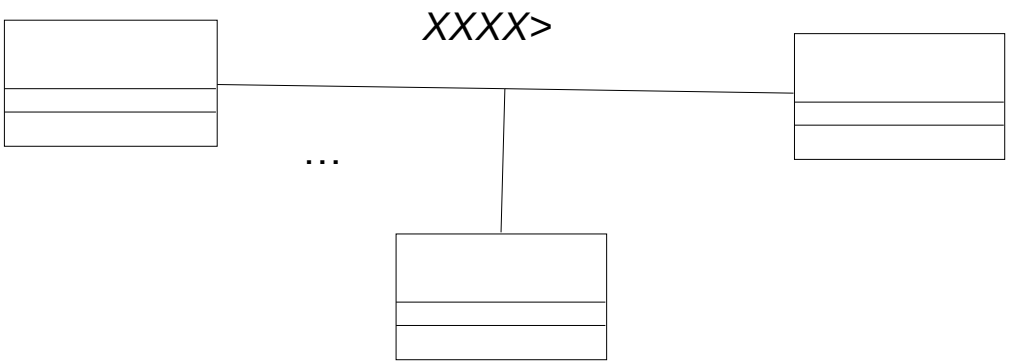


- La *multiplicité* fournit la cardinalité portée par un Role
  - **1** signifie que C1 a 1 et 1 seul Role2
  - **M..N** signifie que C1 a M à N Role2
  - ***** signifie que C1 a 0 à plusieurs Role2
  - **1..*** signifie que C1 a 1 a plusieurs Role2

- Elle peut aussi être utile ailleurs,

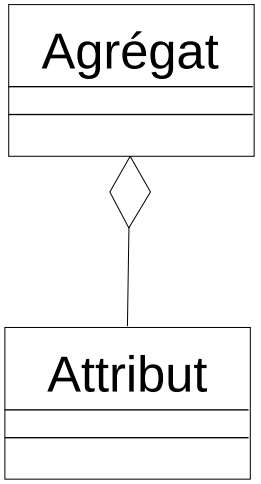
- La *classe association*
  - Une association peut vouloir avoir ses propres propriétés
  - En modèle objet: seules les classes ont des propriétés

  => On associe une classe (dite association) à la propriété

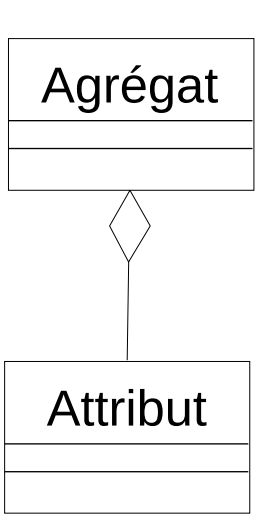


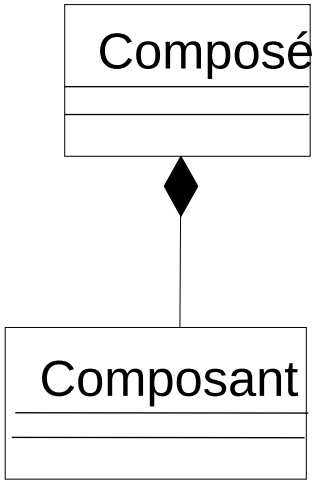
4 Autres relations

- La *relation d'agrégation* est une association qui
  - représente la composition d'une classe (agrégat) par un ou plusieurs composants (une classe agrégée)
  - Agrégat **Est-Propriétaire-D'un** (possède) Attribut
    - on ne peut pas enlever les objets agrégés sans son accord,
    - d'autres objets peuvent s'en servir
  - Est transitive



- La *relation de composition* est une relation plus forte que l'agrégation
  - Composé **Contient** (possède) Composant
    - L'objet composite est responsable de la création, copie et destruction des composants
    - Une instance d'un composant n'appartient qu'à un objet composite





- La *relation 3-aire* est une relation connectant 3 classes
  - Elle porte informations comme les autres associations
  - Elle est peu utilisée, voire déconseillée

- L’association dérivée peut être déduite d’autres relations
  - À un but de lisibilité
  - Est déclarée par un « / » devant le nom de la relation

Bibliographie

- P.A. Muller Modélisation objet avec UML Eyrolles 2003
- J. Gabay Merise vers OMT et UML InterEditions 98
- N. Kettani, D. Mignet, P.Paré, C. Rosenthal-Sabroux De Merise à UML Eyrolles 98
- Développer des systèmes interactifs V. Van-Rylen . Anghkor 98
- Penser objet avec UML et JAVA M. Lai Intereditions 98
- Base de données objet et relationnel G. Gardarin Eyrolles 99
- Introduction à UML Tom Penders. OEV 2002
- UML 2 D. Pilone, O'Reilly 2006
- UML2 C. Morley, J. Hugues, B. Leblanc, Dunod 2008
- Transparents cours S. Loiseau, licence informatique 2000
- Transparent cours, C. Devred, 2011
- UML2 Modélisation des objets, L. de Brauher & al 3eme édition, eni 2013