Diagrammes de classes

- **Introduction**
- **Diagramme de Classes**
  - -Classes
  - -Attributs
  - -Opérations/Méthodes
  - -Attributs dérivés
  - -Visibilité
- **Relations dans les diagrammes de classes**
  - -Association
  - -Héritage
  - -Agrégation
  - -Composition
  - -Dépendance
- **Types de classes**

Introduction

- Les diagrammes de classes se composent d'un ensemble des classes, des interfaces et leurs relations
- Les diagrammes de classes représentent la vue statique du système
- Les diagrammes de classes permettent de produire/ construire le squelette du système
- La modélisation des diagrammes de classes est l'étape essentielle dans la conception orientée objets

Les éléments de base dans les diagrammes de classes sont:

- Classes (Encapsulation des données et des fonctions)
- Attributs (Des données simples dans les classes)
- Opérations / Méthodes (Des fonctions dans les classes / interfaces)
- Relations (Représentent les liens entre les instances des classes)
- Interfaces (Déclaration des fonctions)

Classes

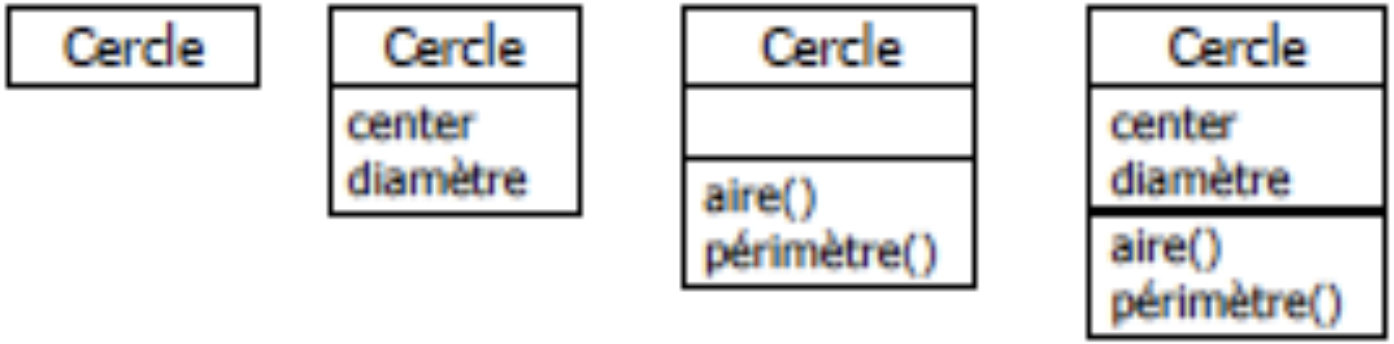
Une classe se compose:

- des attributs
- des opérations

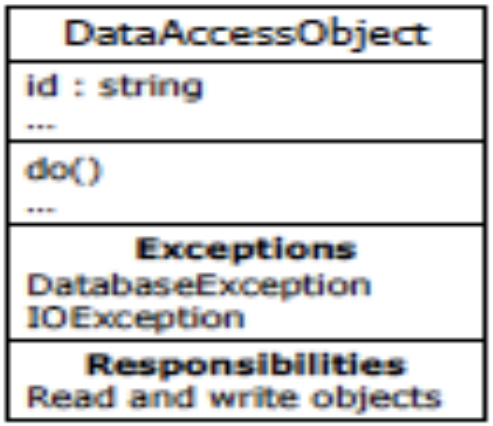
Notation:

- Une classe se représente par une boîte comprenant le nom de la classe
- Plusieurs formes de représentation des classes : du moins complète au plus complète

Classes (suite)



Une classe peut avoir des compartiments supplémentaires:



Attributs

Définition Les attributs représentent les données nécessaires des instances d'une Classe.

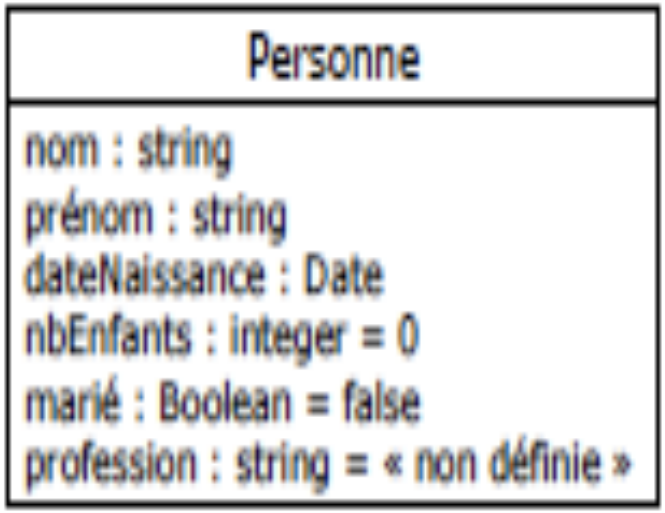
Les attributs peuvent avoir:

- Un type
- type simple
  nombre : integer;
  longueur : real;
  texte : string;
- type complexe
  center : Point;
  date : Date;

Attributs (suite 1)

- Une valeur par défaut
  nombre : integer = 10;
- Une liste des valeurs possibles
  coulor : Color = {rouge, bleu, violet, jaune};

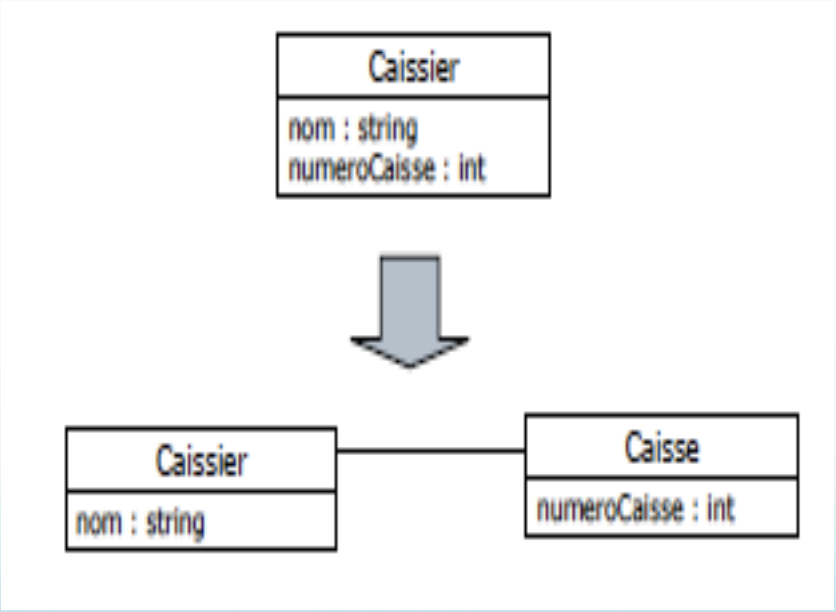
Exemple:



Attributs (suite 2)

- Un attribut ne représente que des données relatives à la classe qui possède cet attribut

Exemple:



Opérations/Méthodes

- Les opérations représentent le comportement des instances de la classe
- Le comportement d'une classe comprend:
  - Les accesseurs et les modificateurs qui sont manipulent sur les données des instances des classes
  - Un certain nombre de tâches, associées à la responsabilité de la classe, etc

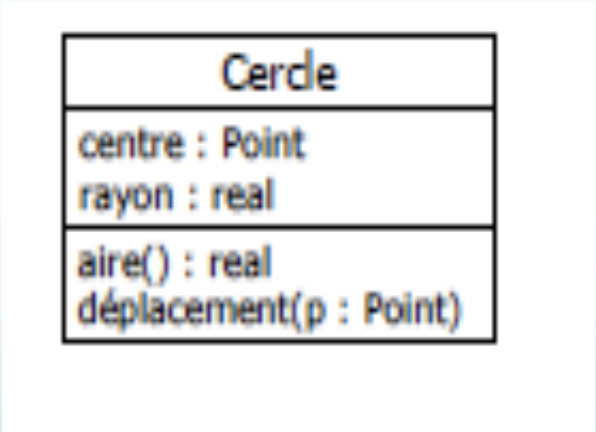
Opérations/Méthodes (Suite)

- Les opérations peuvent avoir

-Un nom (aire, calculer, ...)

-Un type de retour (un type de retour)

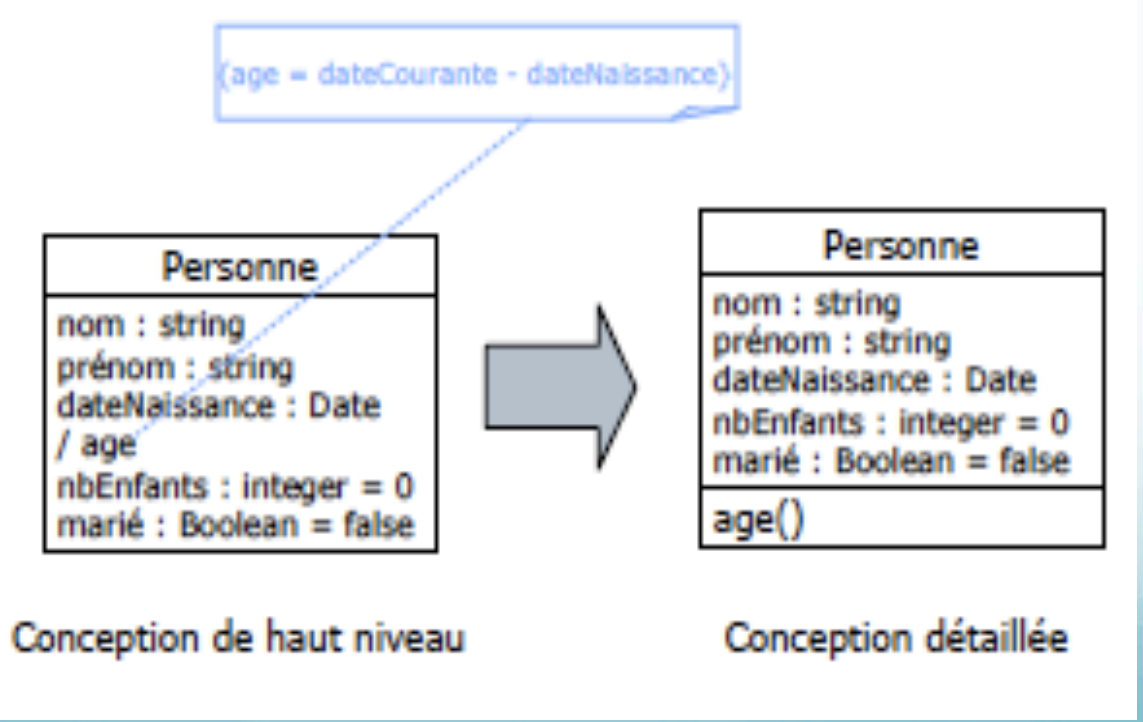
-Des arguments avec leurs types (déplacement(p : Point)



Attributs dérivés

- Les attributs peuvent être déduits d'autres attributs.

L'age d'une personne peut être dérivé de la date de naissance



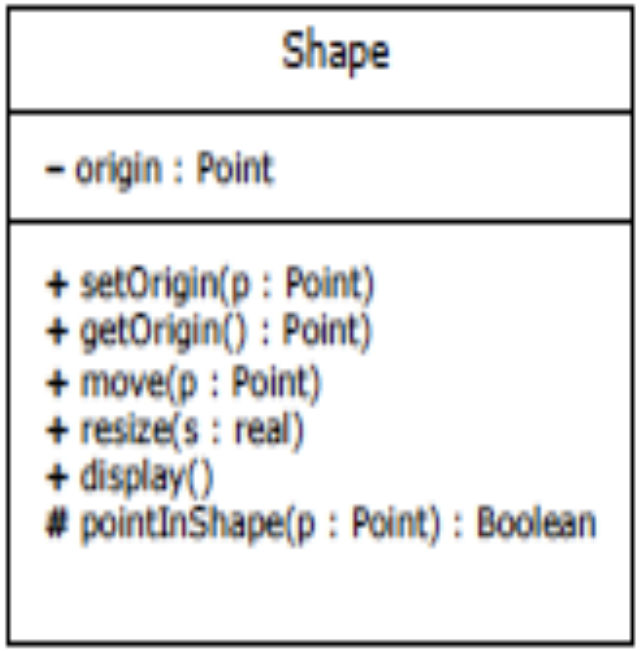
Visibilité

Les attributs et les opérations ont la visibilité

- Publique (public)
  visible à l'extérieur de la classe (notation « + »).
- Protégé (protected)
  visible seulement par les objets de la classe et des objets des sous classes (notation « # »).
- Privé (private)
  visible seulement par les objets de la classe (notation « - »).

Visibilité (Suite)

Exemple:



Types de relation

Les relations entre les classes:

- Association
  Relation sémantique entre les classes
- Héritage
  Une classe peut hériter d'un ou plusieurs classes
- Agrégation
  Une association montre qu'une classe est une partie d'une autre classe
- Composition
  Une forme forte d'agrégation
- Dépendance
  Montre la dépendance entre les classes

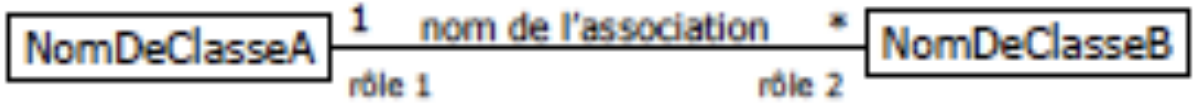
Association

- Une association est utilisée pour montrer comment deux classes sont liées entre elles:

  - Une association exprime une connexion sémantique bidirectionnelle entre classes.

  - Une association est une abstraction des liens entre des instances des classes.

  -Notation:



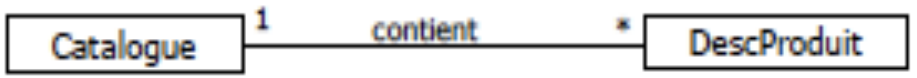
- Chaque extrémité d'une association est appelée un rôle

  - Un rôle montre le but de l'association

  - Un rôle peut avoir (Un nom, L'expression de multiplicité)

Association (Suite 1)

- Multiplicité (définir combien d'instances de la classe A est associé à un instance de la classe B)



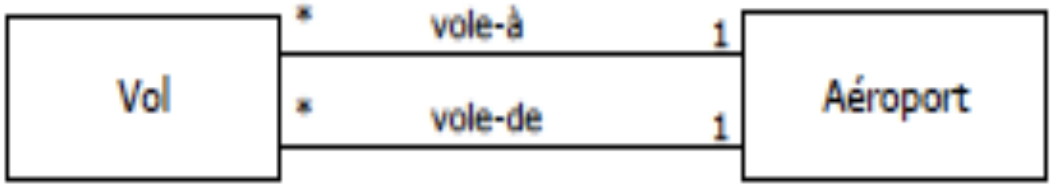
- Différentes expressions de multiplicité

1	:	un et un seul
0..1	:	zéro ou un seul
m..n	:	de m à n (entiers naturels)
n	:	exactement n (entier naturel)
*	:	zéro ou plusieurs
1..*	:	d'un à plusieurs

Association (Suite 2)

- Associations multiples
- Deux classes peuvent avoir plusieurs associations entre elles

Exemple:



Association (Suite 3)

Association directionnelle et Attributs

- Par défaut, les associations sont bidirectionnelles
- Pourtant, les associations peuvent être directionnelles

- La navigabilité pointant de Caisse à Vente montre que **Caisse** possède un attribut de **type Vente**
- Cet attribut s'appelle venteCourrante

Autre façon de représentation: utilisation de l'attribut

Association (Suite 4)

Quand utilise-t-on l'association directionnelle ou l'attribut?

- On utilise l'attribut pour les types de données, comme Boolean, Time, Date, Real, Integer, ...

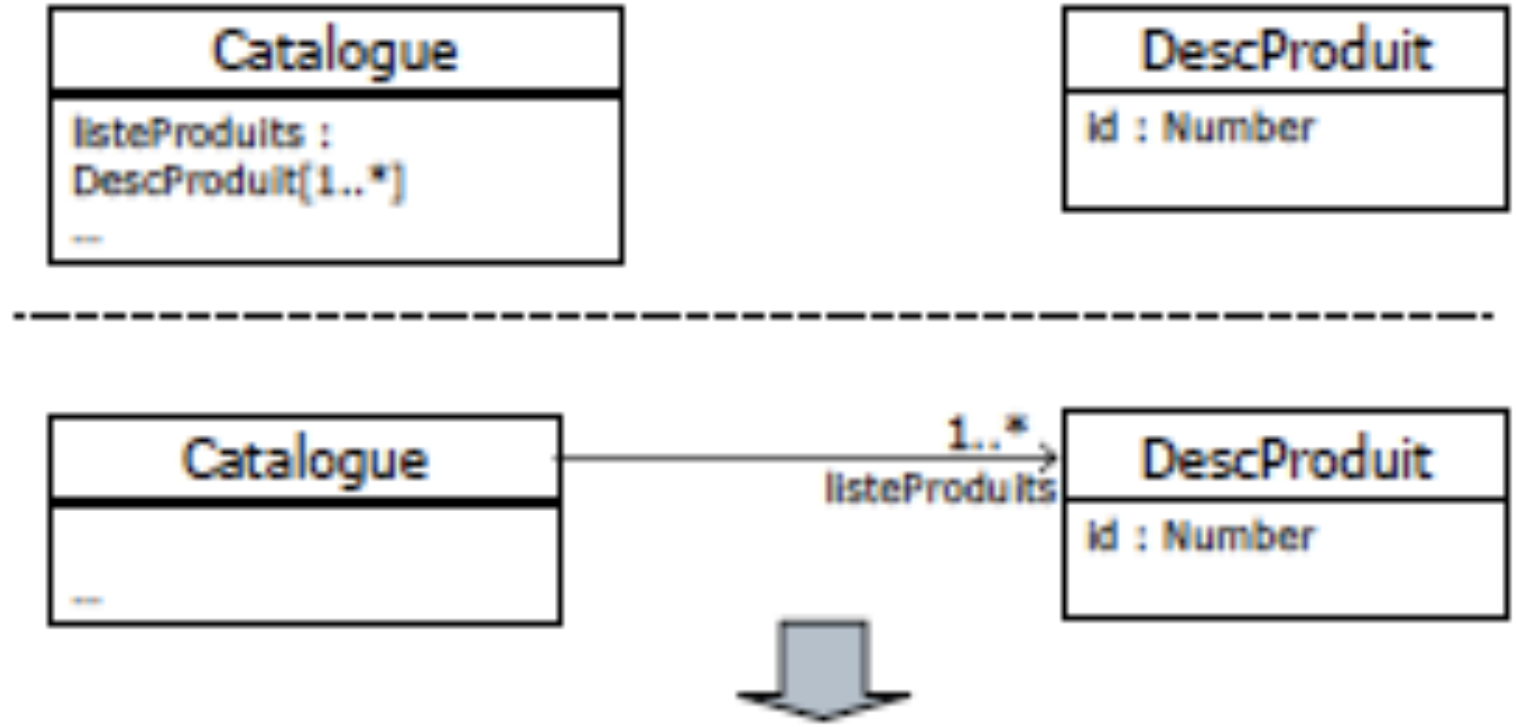
- On utilise l'association directionnelle pour d'autres classes Pour voir mieux les connexions entre les classes

- C'est juste pour mieux représenter, ces deux façons sont sémantiquement équivalentes

Exemple:

Association (Suite 5)

Autre exemple:



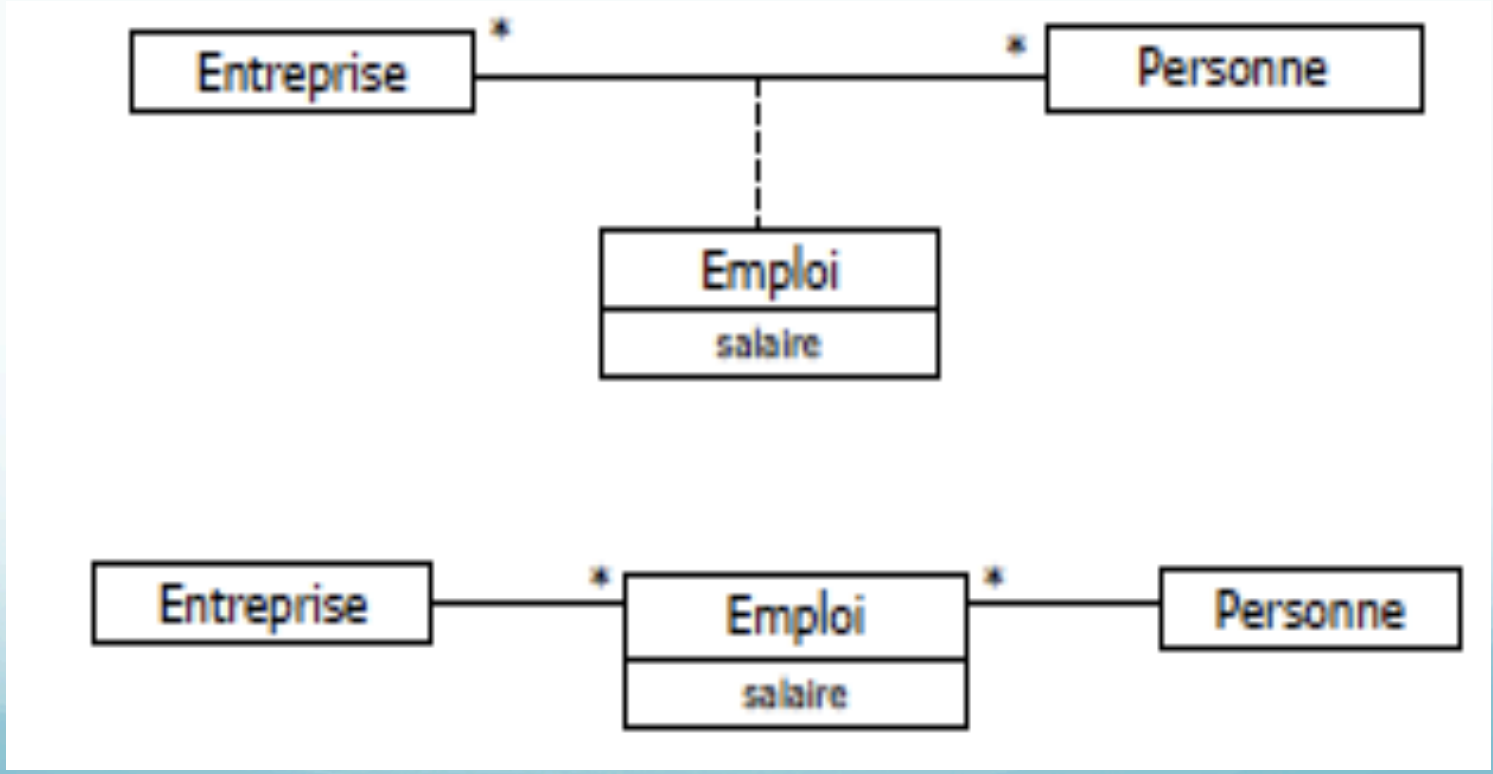

```
public class Catalogue {
        private List<DescProduit> listeProduits =
                                        new ArrayList<DescProduit>();
// ...
}
```

Association (Suite 6)

Classes d'association

- Une classe d'association permet de considérer une association comme une classe lorsqu'un attribut ne peut pas être attaché à aucune de deux classes d'une association

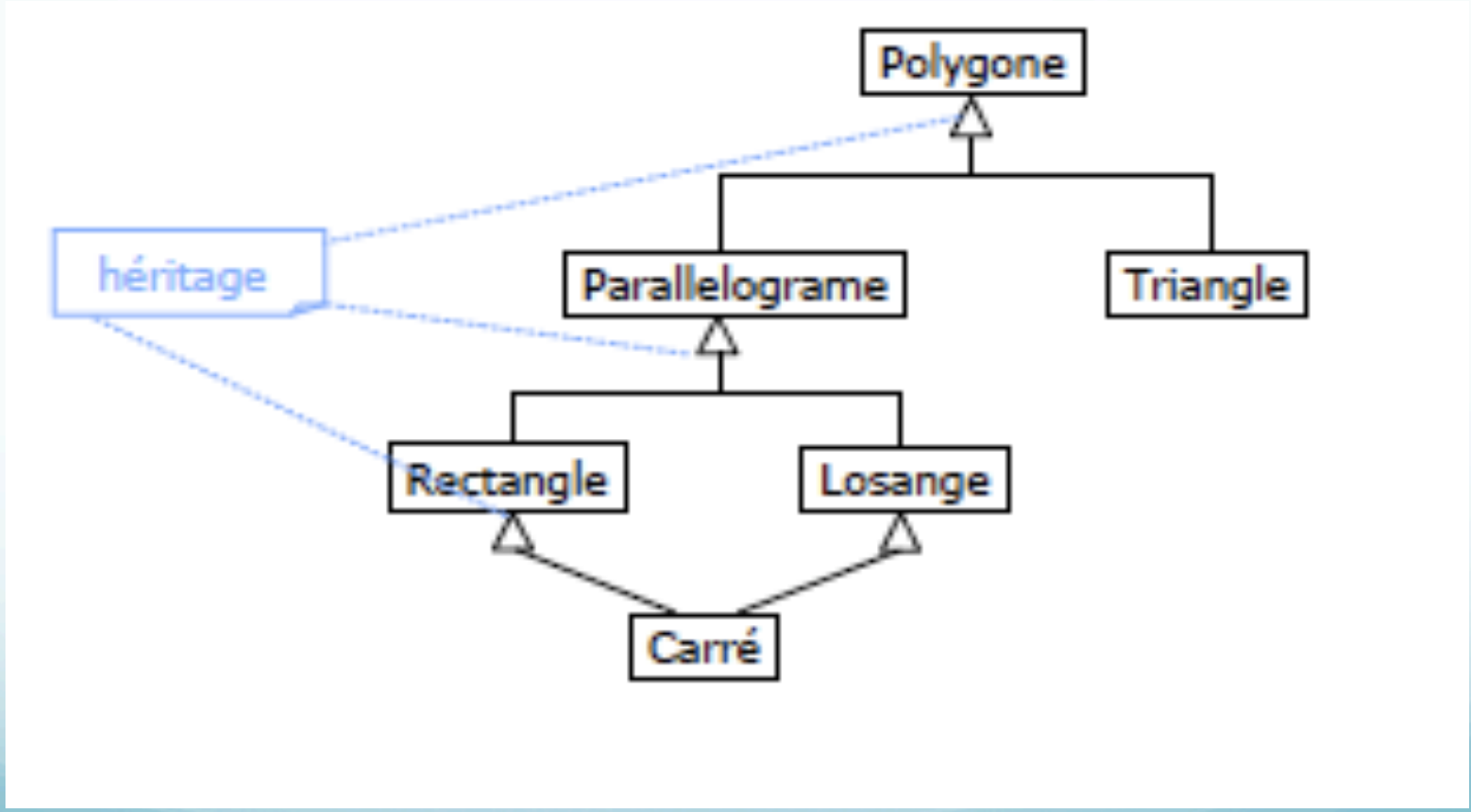
Exp:



Héritage

Héritage

Une super classe se spécialise en sous classes:



Héritage(Suite 1)

Principe de substitution

-Tout objet d'une sous classe peut jouer le rôle d'un objet de sa super classe.

-Un objet d'une sous classe peut substituer un objet de sa super classe.

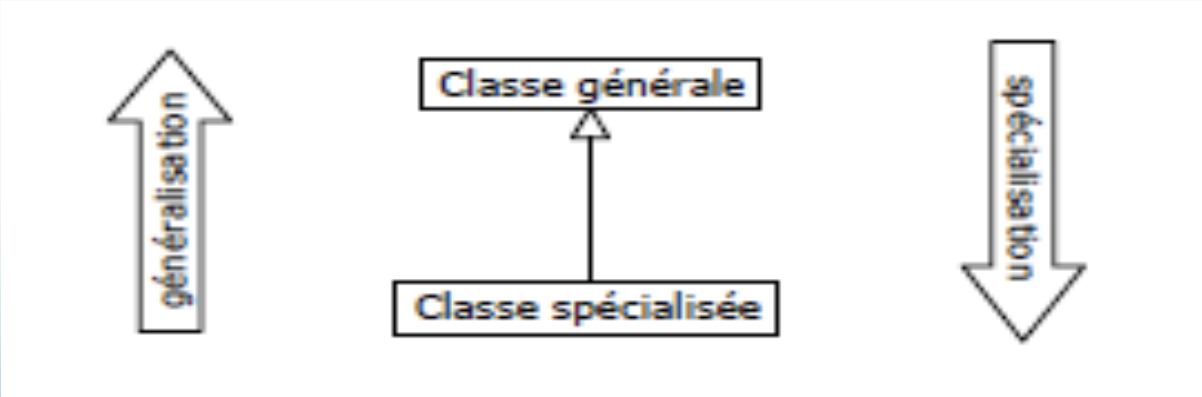
- Informellement

Une sous classe est une sorte de super classe.

Exemple: Un triangle est une sorte du polygone.

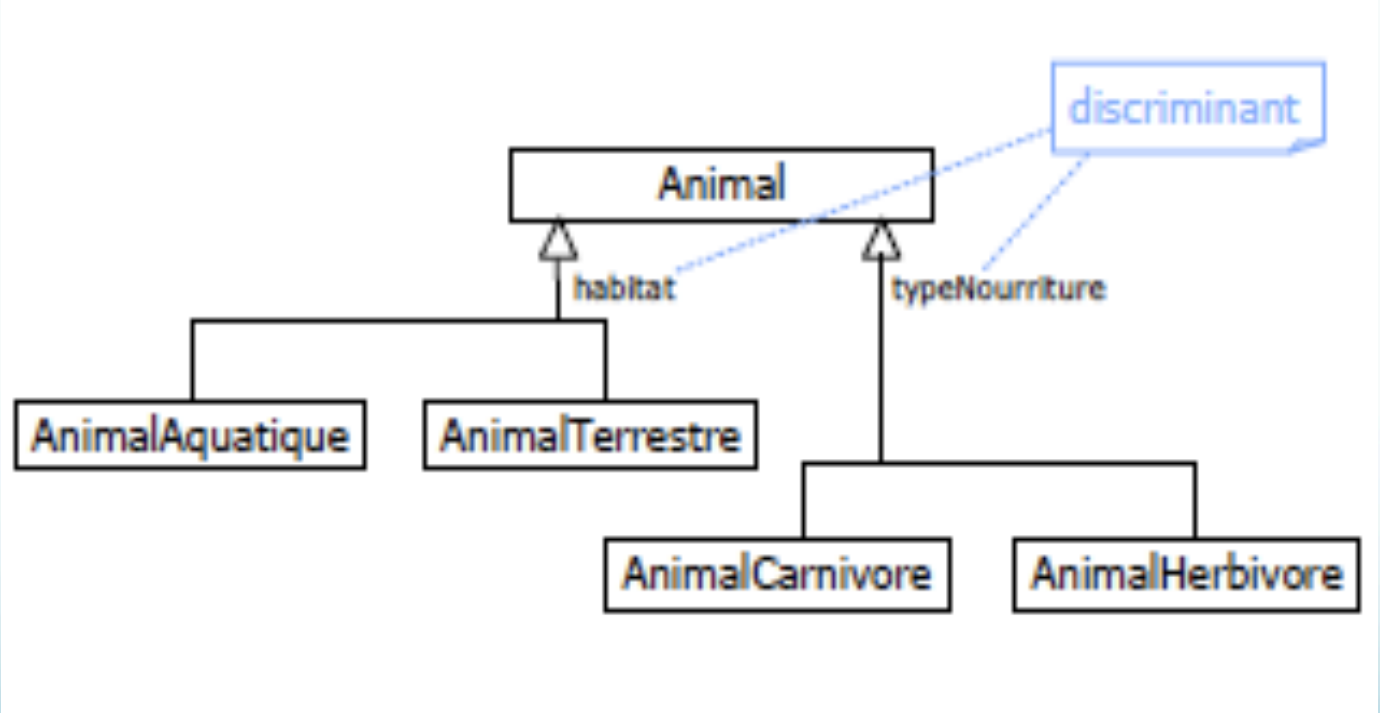
Héritage(Suite 2)

- Les sous classes sont aussi appelées **classes Spécialisées**
- Les super classes sont aussi appelées **classes générales.**
- L'héritage est aussi appelé **la spécialisation** ou **généralisation**



Héritage(Suite 3)

- Le discriminant (optionnel) est une étiquette décrivant le critère suivant lequel se base la spécialisation



Agrégation

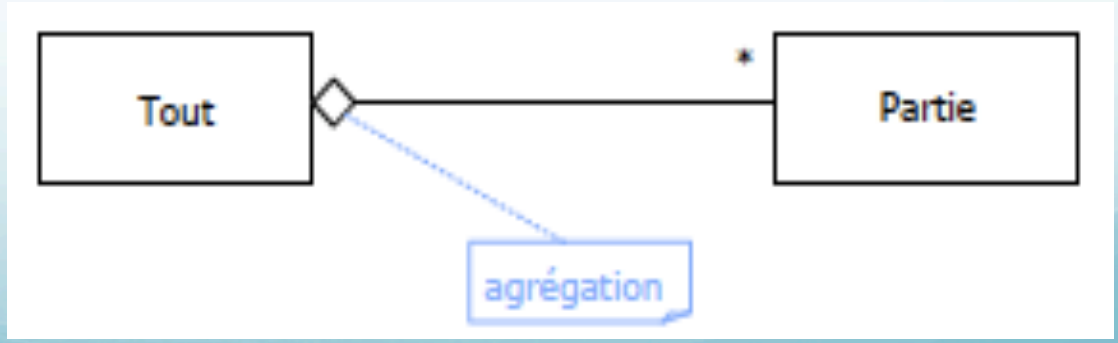
Une agrégation est une forme d'association qui exprime un couplage plus entre les classes

- Une agrégation est utilisée entre deux classes lorsque la relation qu'elle représente est de type

  - Maître et esclaves : « appartient à »
  - Tout et parties : « est une partie de »

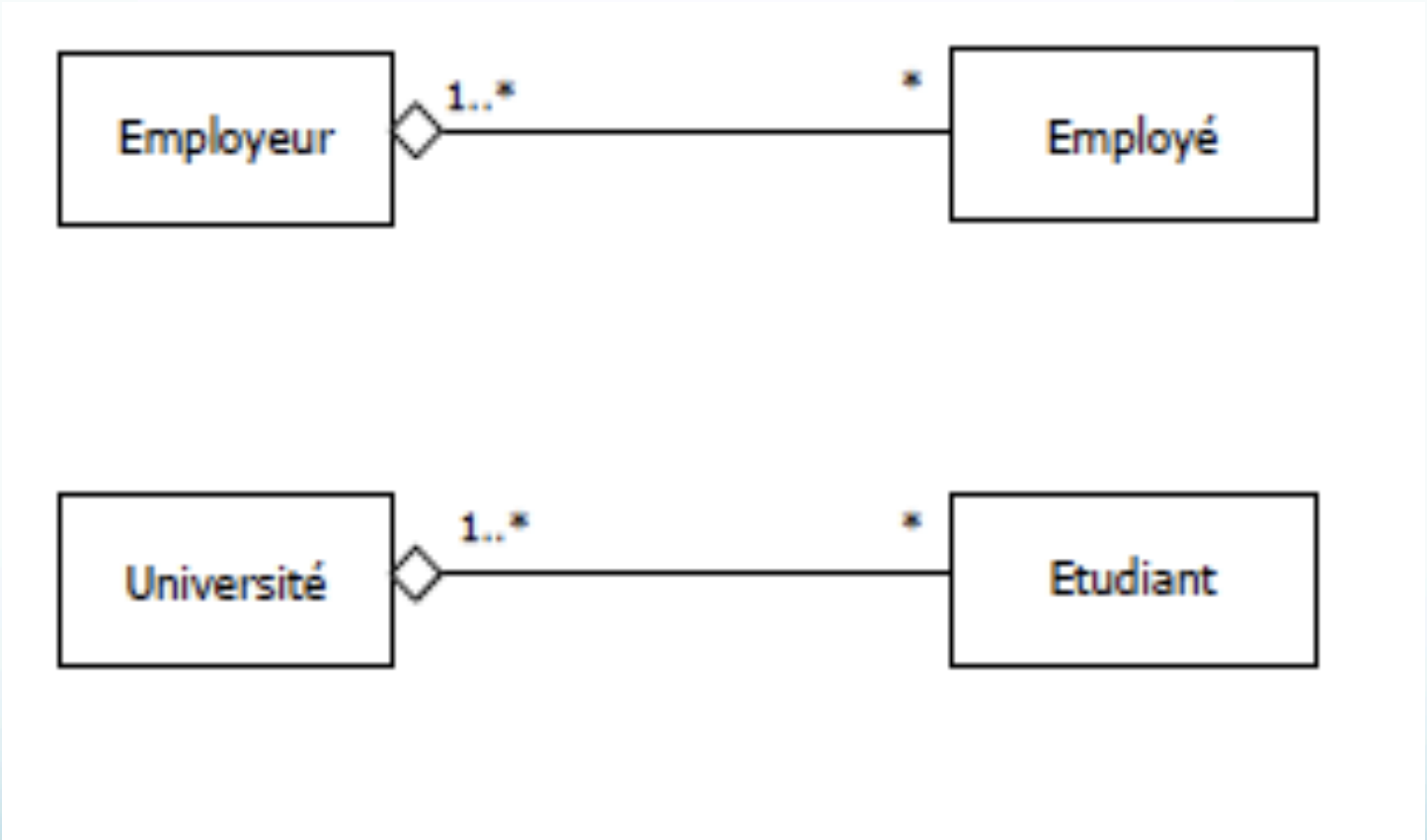
- Notation:

Le symbole désignant l'agrégation se place du côté de l'agrégat



Agrégation (Suite 1)

Exemple:



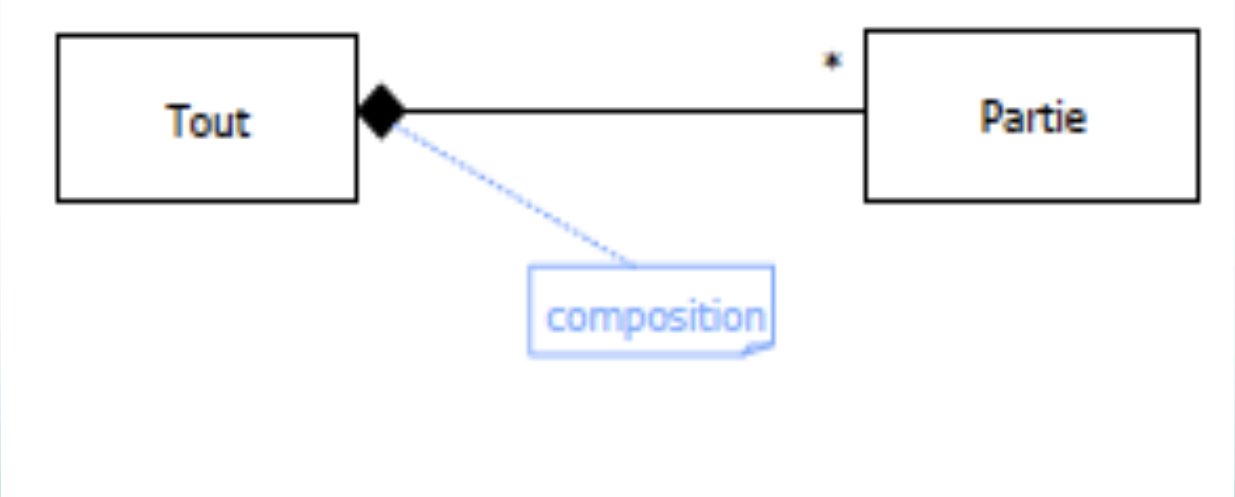
Composition

Une composition est une forme forte d'agrégation

- Une composition est aussi une relation « tout-partie » mais l'agrégat est plus important

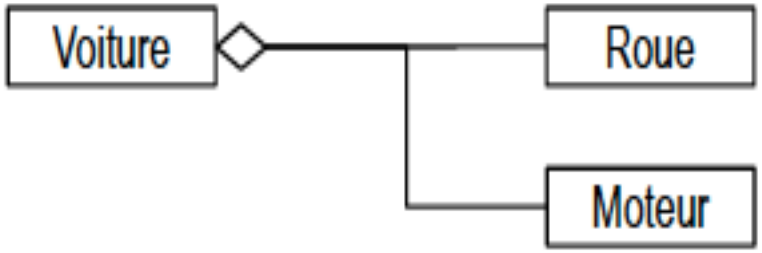
Si le tout est détruit, les parties sont aussi détruites

Notation:



L'agrégation

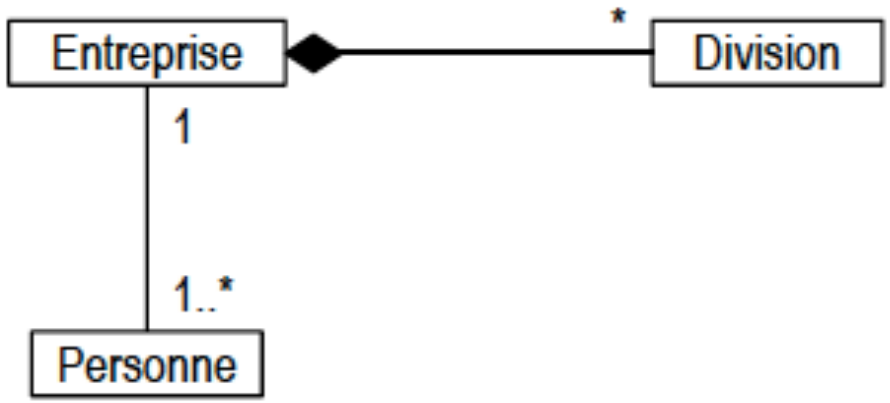
- L'**agrégation** dénote une connexion plus forte que l'association :
  - Plusieurs objets sont groupés afin de former un objet plus important
  - On peut y voir un **tout** et des **parties**, le tout valant plus que la somme des parties
  - En UML on met un losange évidé côté tout



- Simple association ou agrégation ?
  - Y a-t-il un tout et des parties ? Non ⇒ simple association

Composition

- La **composition** est une forme forte d'agrégation :
  - Une partie ne peut pas appartenir à plus d'un tout
  - La destruction du tout entraîne celle des parties
  - En UML on utilise un losange plein côté tout



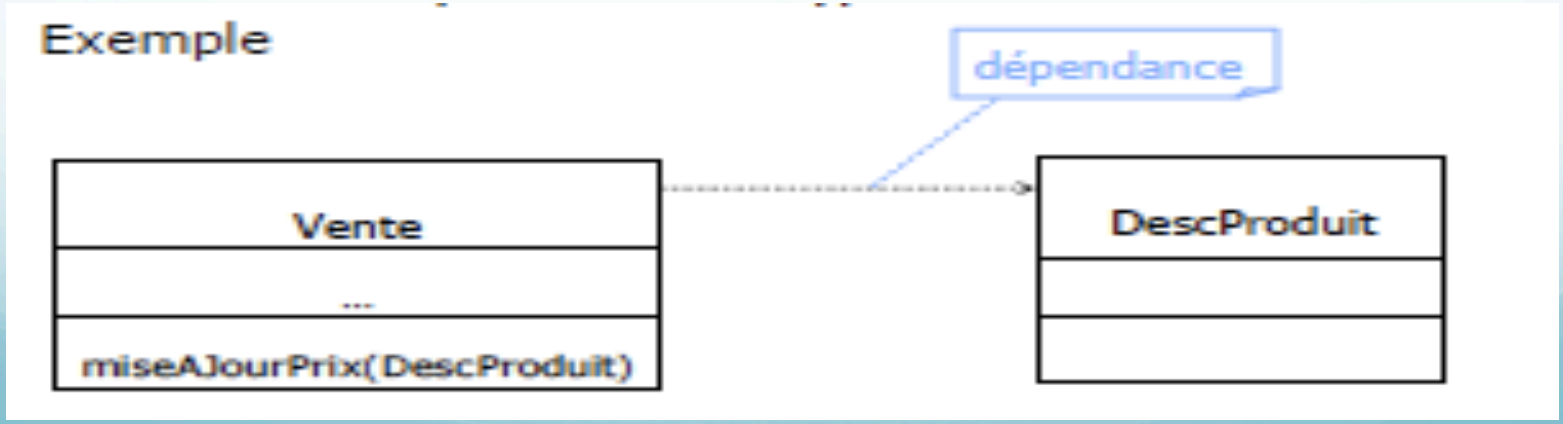
- Agrégation ou composition ?
  1. La « partie » peut-elle faire partie de plus d'un « tout » ?
     Oui ⇒ agrégation, non ⇒ question suivante
  2. La destruction du tout entraîne-t-elle celle des parties ?
     Oui ⇒ composition, non ⇒ agrégation

Dépendance

Une classe peut dépendre d'une autre classe

Plusieurs types de dépendance

- Avoir un attribut de type d'une autre classe
- Envoyer un message en utilisant un attribut, une variable locale, une variable globale d'une autre classe ou des méthodes statiques
- Recevoir un paramètre de type d'une autre classe

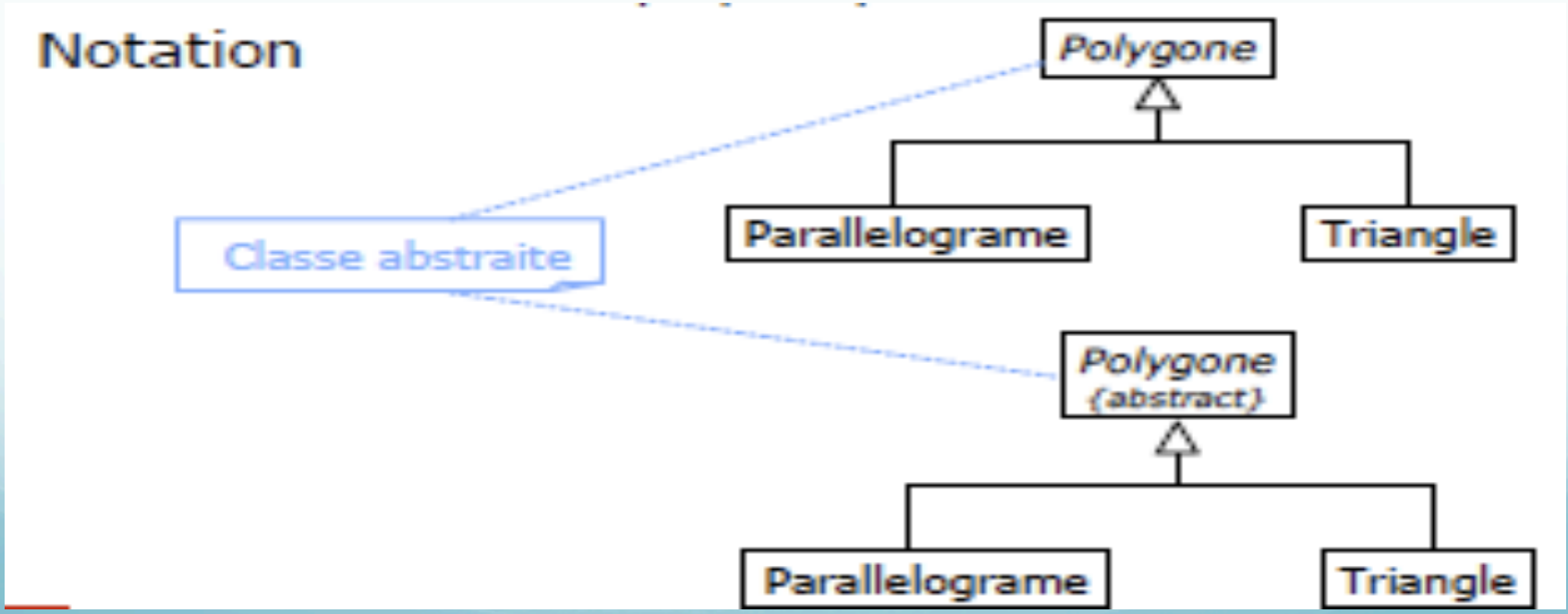


Types de Classes

Classe abstraite

- Une classe abstraite est une classe qui n'a pas des instances:

Liée à la notion de l'héritage
Liée à la notion du polymorphisme

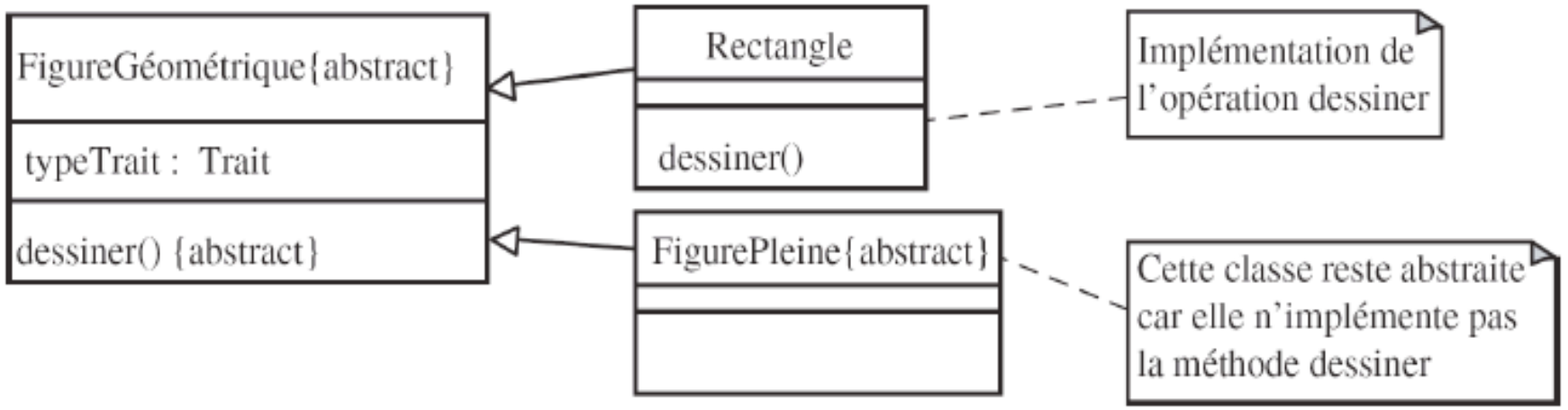


Classes abstraites

- Une méthode est dite **abstraite** lorsqu'on connaît son entête mais pas la manière dont elle peut être réalisée.

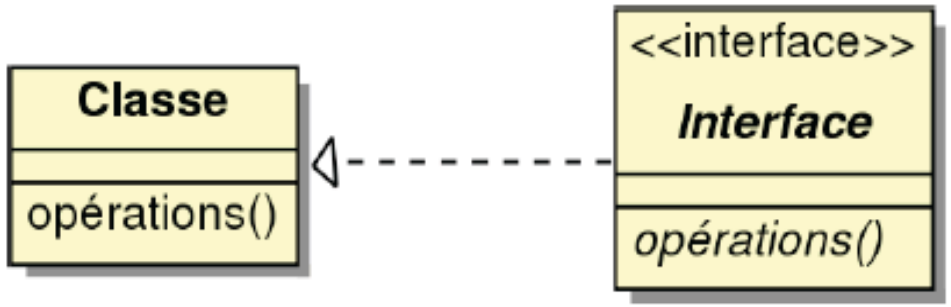
Il appartient aux classes enfant de définir les methodes abstraites.

- Une classe est dite **abstraite** lorsqu'elle définit au moins une méthode abstraite ou lorsqu'une classe parent contient une méthode abstraite non encore réalisée.



Interface

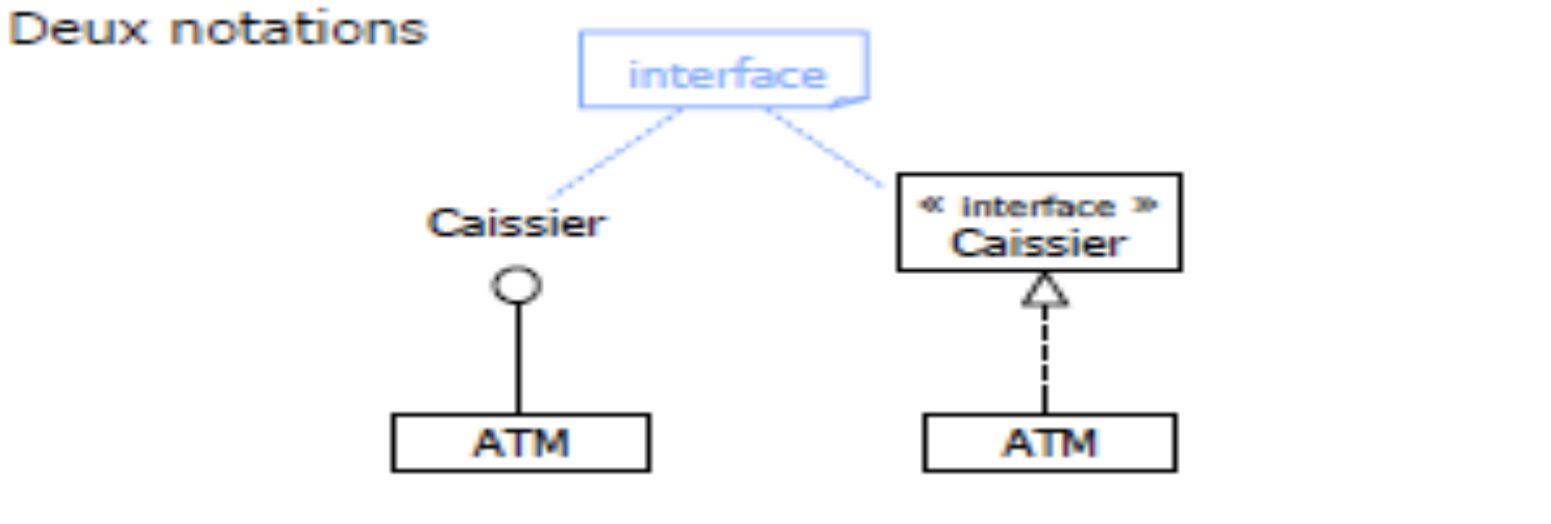
- Le rôle d'une **interface** est de regrouper un ensemble d'opérations assurant un service cohérent offert par un classeur et une classe en particulier.
- Une interface est définie comme une classe, avec les mêmes compartiments. On ajoute le stéréotype « interface » avant le nom de l'interface.
- On utilise une relation de type **réalisation** entre une interface et une classe qui l'implémente.



- Les classes implémentant une interface doivent implémenter toutes les opérations décrites dans l'interface

Interface

- Une interface décrit une portion du comportement visible d'un ensemble d'objets
- Une interface est très similaire à une classe abstraite ne contenant que des opérations abstraites (virtuelles)
- Une interface spécifie seulement les opérations sans implémentation

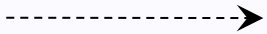


Notion de paquetage

Packages

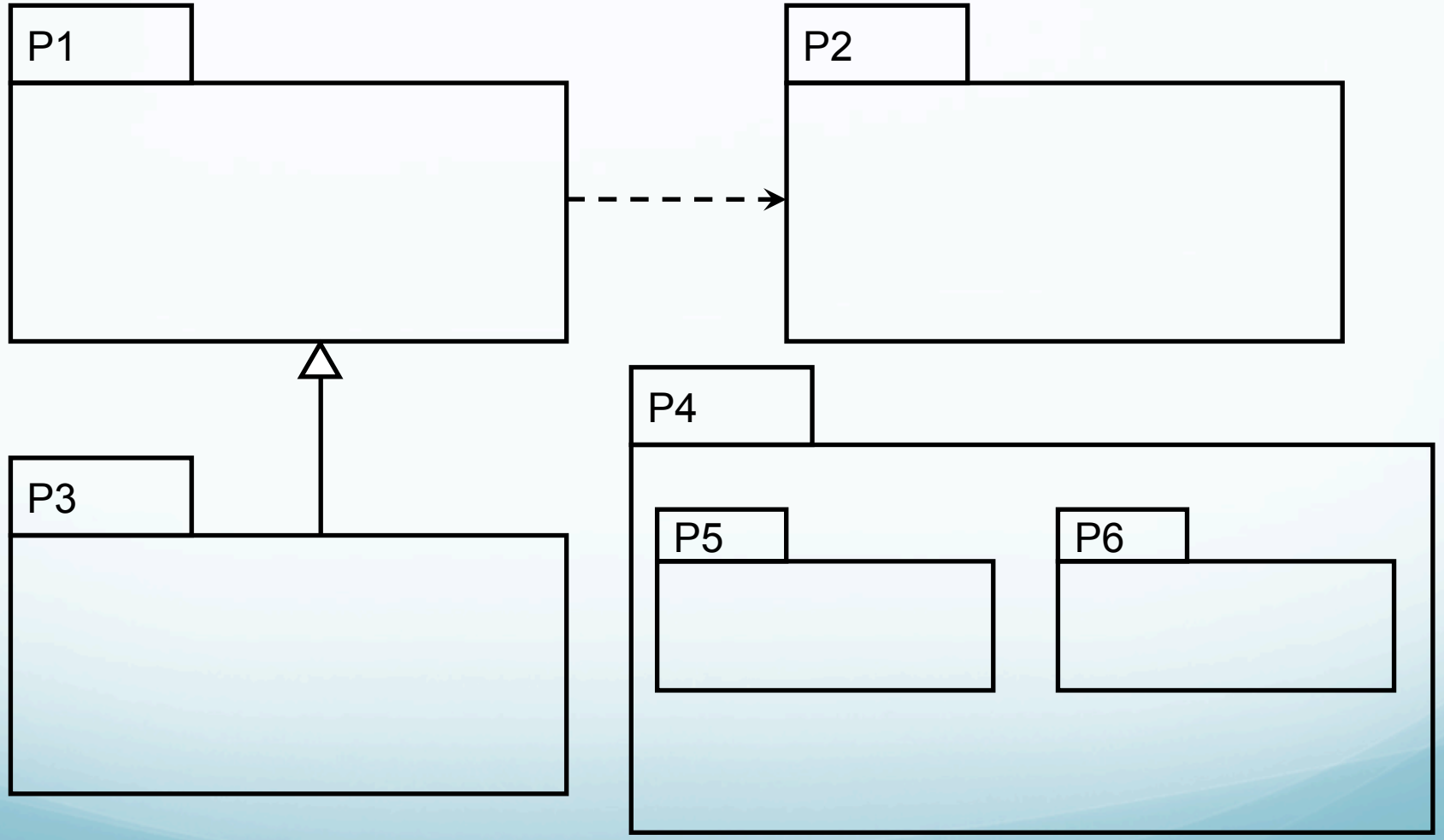
- Les paquetages sont des éléments d'organisation des modèles
- Ils regroupent des éléments de modélisation, selon des critères purement logiques
- Ils permettent d'encapsuler des éléments de modélisation
- Ils permettent de structurer un système en catégories (vue logique) et sous-systèmes (vue des composants)
- Ils servent de "briques" de base dans la construction d'une architecture
- Ils représentent le bon niveau de granularité pour la réutilisation

Relation entre paquetage

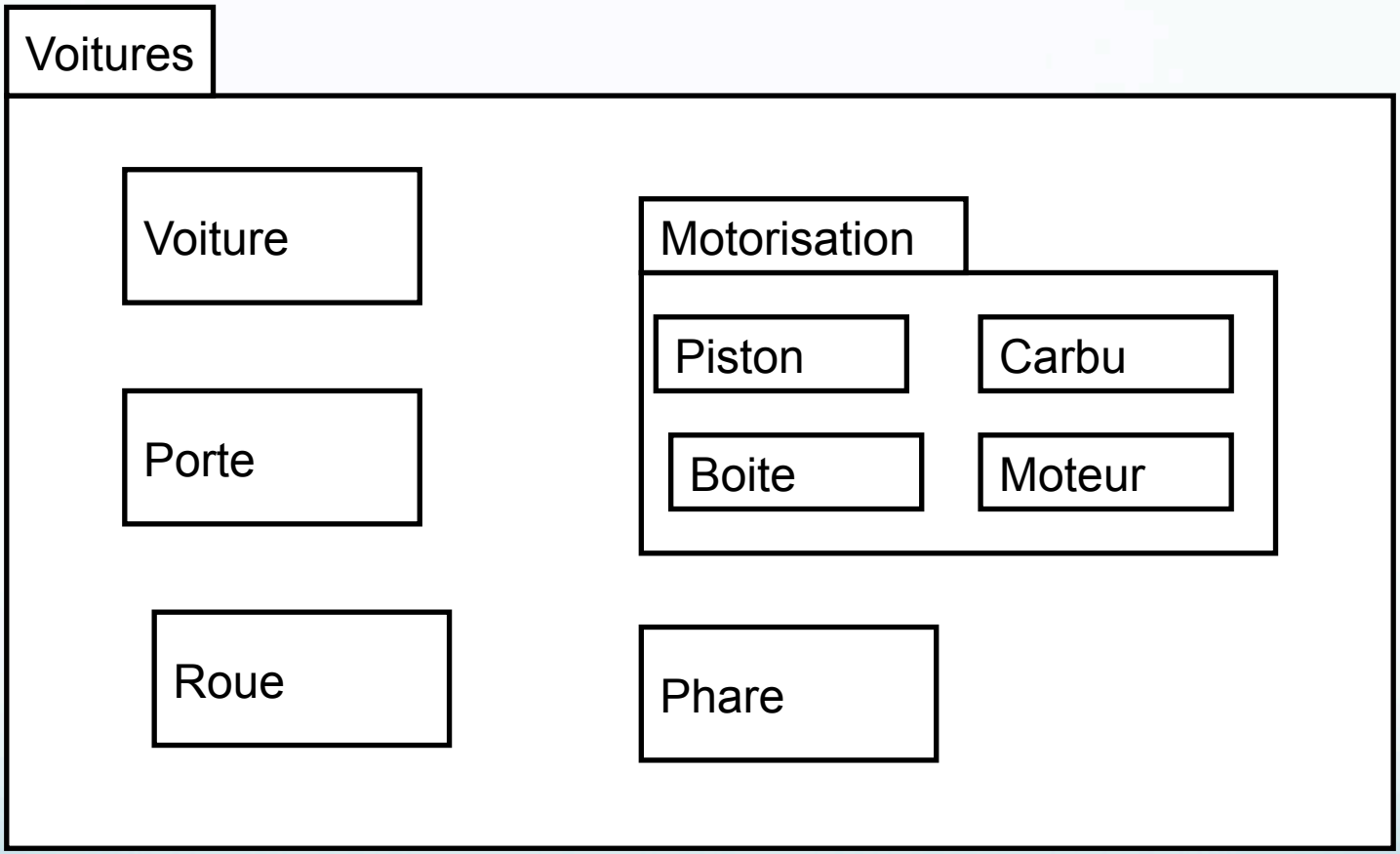
- Dépendance : 
  - Au moins un élément du paquetage source utilise les services d'au moins un des éléments du paquetage destination

- Héritage : 
  - Au moins un élément du paquetage source spécialise (est dérivée d') au moins un des éléments du paquetage destination

Convention graphique



Exemple


Voitures
Voiture
Porte
Roue
Motorisation
Piston
Carbu
Boite
Moteur
Phare

Exemple de package