Introduction au génie logiciel

Modélisation :
Rappels sur l'étude statique

INFO0504 2020-2021

Université de Reims Champagne-Ardenne

Supports utilisés

- Spécification UML 2.5
- UML 2 par la pratique, Pascal ROQUES, éditions Eyrolles, ISBN 978-2-212-13344-8
- Architecture Logicielle, Jacques PRINTZ, 3ème édition, DUNOD, ISBN 978-2-10-057865-8

- Design Patterns in JAVA, Vaskaran Sarcar, B/W Edition, IBSN 978-1-517-07144-8

La modélisation idéale

- Objectifs :
  - De représentation non ambigüe,
  - De compréhension générale,
  - D'expressivité intuitive,
  - D'indépendance vis-à-vis des domaines techniques.

La modélisation réaliste

- Objectifs :
  - De limiter les ambigüités
  - D'être accessible au plus grand nombre
  - De correspondre à des vues classiques
  - De représentativité d'une partie des domaines (informatiques)

Modélisation et UML

- UML ⇔ « *Unified Modeling language* »

- Mis en place par l'OMG (*Object Management Group*)

- Dernière version : 2.5.1 (décembre 2017)

⇔ modifications mineures de la 2.5 (juin 2015)

Modélisation et UML

- Ce qui définit UML (traduction littérale):
  - Une syntaxe abstraite basée sur une définition formelle des méta-modèle du MOF (*Meta-Object Facility*)

Modélisation et UML

- Ce qui définit UML (traduction littérale):
  - Des modèles à la sémantique clairement définie indépendante des technologies et s'intégrant dans un processus de génération par ordinateur.

Modélisation et UML

- Ce qui définit UML (traduction littérale):
  - Des éléments graphiques aisément compréhensibles à travers un ensemble de diagrammes décrivant les caractéristiques des systèmes modélisés.

Prise en main de l'UML

- 3 axes possibles :
  - Axe fonctionnel : définitions des actions du systèmes et relations entre les différents acteurs
  - Axe statique : représentation du système de manière global, de son architecture et éventuellement de son déploiement.
  - Axe dynamique : évolution du système en fonction des stimuli

Prise en main de l'UML

- 3 axes possibles :
  - Axe fonctionnel : définitions des actions du systèmes et relations entre les différents acteurs
  - **Axe statique : représentation du système de manière global, de son architecture et éventuellement de son déploiement.**
  - Axe dynamique : évolution du système en fonction des stimuli

Etude statique

- Vue architecturale du système
- Proche de la modélisation orientée objet
- Permet la génération automatique de code

➔ *diagramme de classe*

Etude statique

- Le diagramme de classe permet de :
  - visualiser les relations entre les objets :
    - Généralisation (héritage)
    - Agrégation/composition
    - Association
    - …
  - Guider le développeur indépendamment du langage

Etude statique

- Une classe possède au moins un nom

- Une classe peut aussi détailler :
  - Son type : interface, classe abstraite, …
  - Ses attributs (nom et accessibilité)
  - Ses capacités ⇔ opérations
  - Sa multiplicité

Etude statique

- Classe minimale

Pokemon

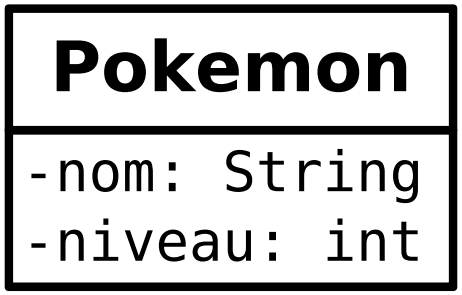
Etude statique

- Ajout d'attributs :
  - Privés : non visible pour les autres classes (même les sous-classes)

    ➔ au besoin créer des accesseurs
  - Protégés : comme « privé » sauf pour les sous-classes (les classes « filles »)
  - « Package » : visible pour toutes les classes d'un même « package » (voir diagramme correspondant)

Etude statique

- Ajout d'attributs :



- Accessibilité :
  - Prive ⇔ -
  - Protégée ⇔ #
  - Publique ⇔ +
  - « Package » ⇔ ~

Etude statique

- Ajout d'attributs :
  - d'instances : type par défaut où la valeur est spécifique à chaque instance
  - De classe : attribut dont la valeur est liée à la classe et non à l'instance

    ➔ attribut statique
- Exemples d'attribut de classe :
  - Une constante;
  - Un compteur d'instance;

Etude statique

- Ajout de compétences (opérations) :

- Accessibilité et type comme les attributs

- Opérations utilitaires (accesseurs, …)

Etude statique

- Relations entre les classes :
  - Dépendance

    ⇔ relation « utilise une instance de »

  - Association

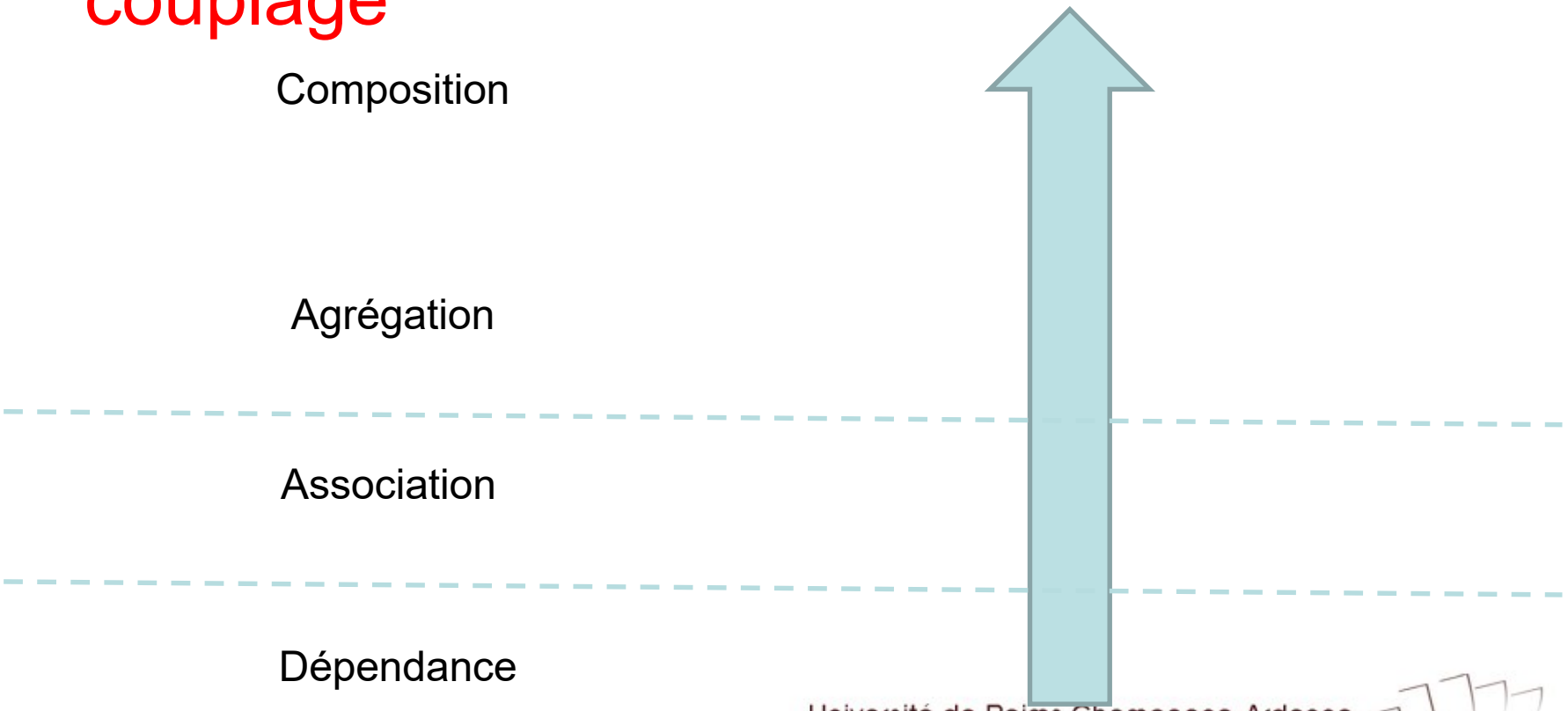
    ⇔ relation « possède une instance de »

  - Agrégation / Composition

    ⇔ relation « est composée de »

Etude statique

- Relations entre les classes : évolution du couplage



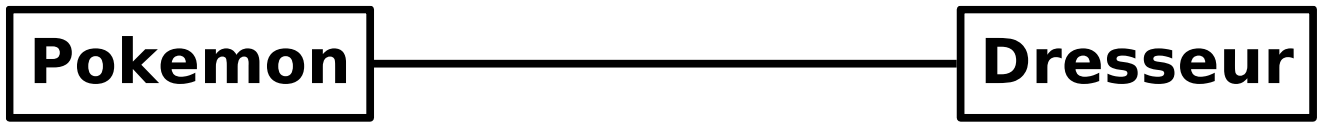
Etude statique

- Exemple de dépendance

- La classe « Compétence » peut être utilisée comme :
  - Argument d'une opération
  - Ou au sein d'une opération
  - Ou encore comme valeur de retour

Etude statique

- Exemple d'association non orientée anonyme

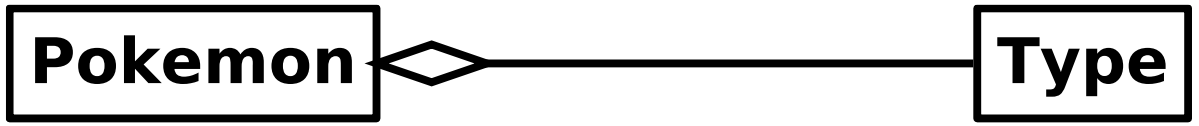


- La classe « Pokemon » possède une instance de dresseur
- La classe « Dresseur » possède une instance de Pokemon

➔ Cette relation n'est pas structurelle

Etude statique

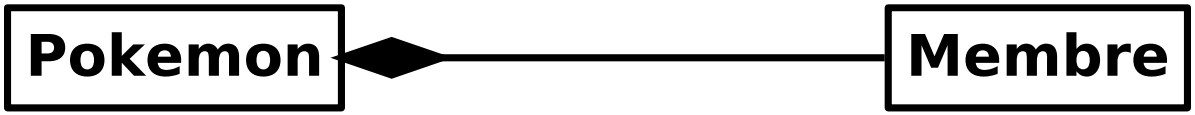
- Exemple d'agrégation



- Une instance de « Pokemon » possède forcément un type
- Une instance de « Type » peut exister indépendamment d'une instance de « Pokemon »

Etude statique

- Exemple de composition



- Une instance de « Pokemon » possède forcément (au moins) un « Membre »;
- Une instance de « Membre » est liée à instance de « Pokemon » et ne peut exister sans.

Etude statique

- Toutes les relations peuvent être :
  - Nommées
  - Orientées
  - Multiple entre objets
  - Associées à des contraintes
  - Qualifiées

Etude statique

- Relations spécifiques à la construction des classes :
  - Généralisation :

    ⇔Une classe est une spécialisation d'une classe plus générique (pouvant être abstraite);

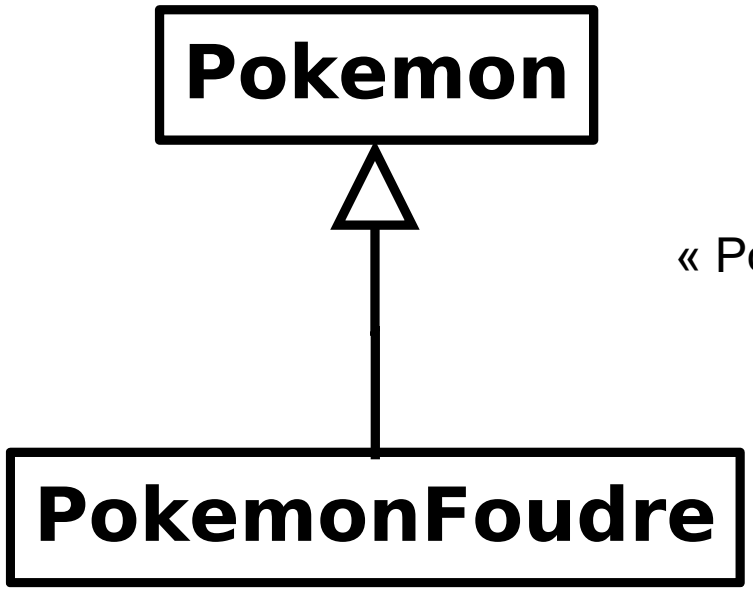
    ⇔Relation « est un ».
  - Réalisation :

    ⇔Une classe définie l'implémentation d'opérations (méthodes) définies dans une interface;

    ⇔Relation « peut être un ».

Etude statique

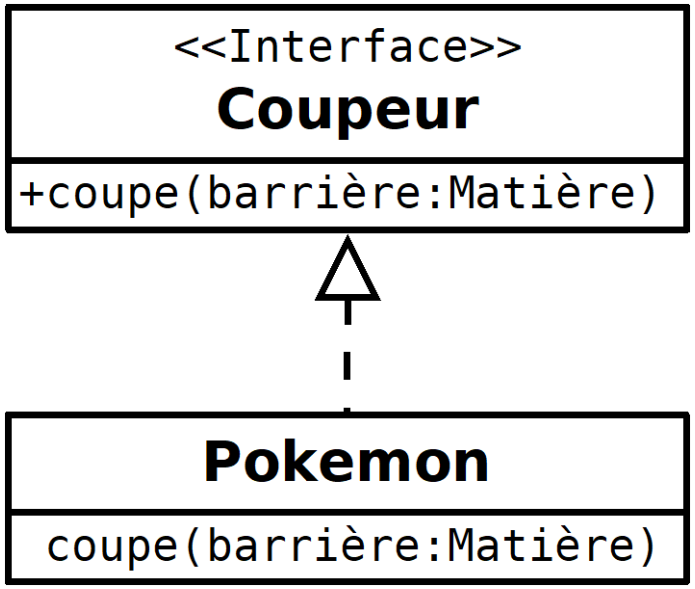
- Exemple de généralisation :



« PokemonFoudre » = « Pokemon » + …

Etude statique

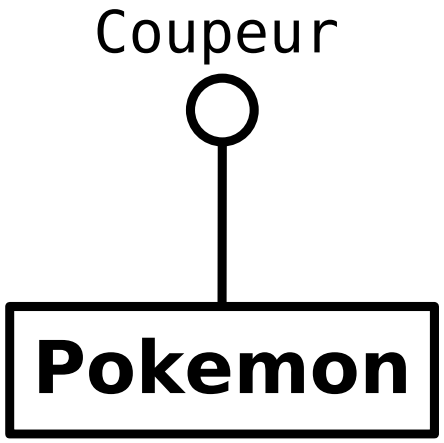
- Exemple de réalisation :



Un « Pokemon » peut être un « Coupeur » est donne l'implémentation de l'opération « coupe »

Etude statique

- Exemple de réalisation :



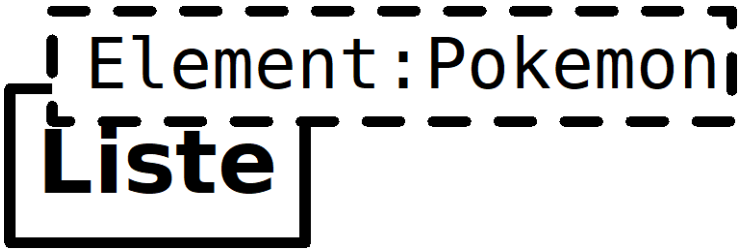
Un « Pokemon » peut être un « Coupeur »
est donne l'implémentation de l'opération
« coupe »

Etude statique

- Définition de traitements génériques

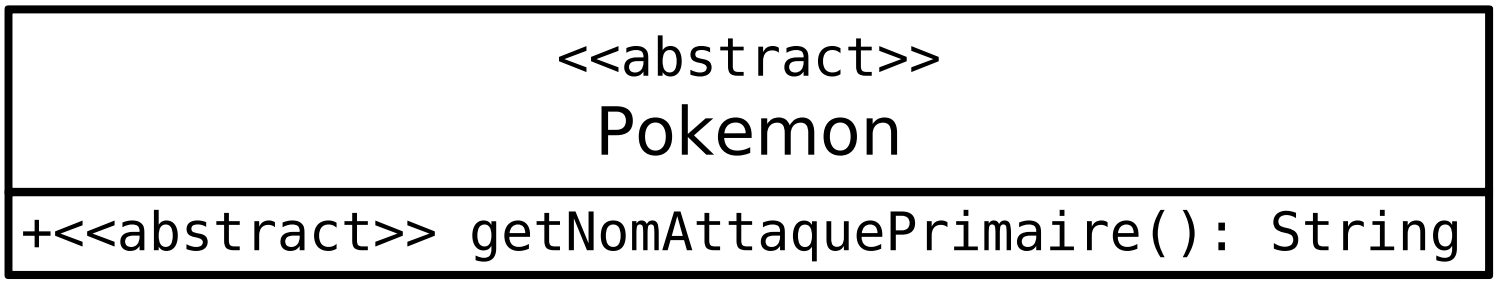
  ⇔ non liés à une classe spécifique

- Notion de « template » ➔ typé à l'instantiation



Etude statique

- Définition d'une entité sans pouvoir en définir une partie du fonctionnement interne.
  - ⇔ Certaines parties dépendent d'une spécialisation
  - ⇔ existence d'opérations abstraites



Etude statique

- Le diagramme de classe repose sur :
  - Le niveau d'abstraction choisi;
  - Le découpage cohérent en « package »;
  - L'expérience du modélisateur;
  - L'utilisation de « recettes » connues de la Conception Orientée Objet (COO).

Etude statique

- En COO, une « recette » ⇔ design pattern
- Design pattern (Patron de Conception)

  ⇔ un modèle valide et reconnu

  ⇔ un couple problème/solution

  ⇔ une vue indépendante d'un langage de programmation (objet)

Les « design patterns »

- Basés sur le livre (1994) :

  « Design Patterns: Elements of Reusable Object-Oriented Software »

  ISBN : 0201633612

- Du « Gang Of Four » ⇔ GOF :
  - Erich Gamma
  - Richard Helm
  - Ralph Johnson
  - John Vlissides

Les « design patterns »

- Classés en 3 familles de patrons:
  - De création : lié à la construction des objets qui se veut être indépendante de leur structure
  - Structurels : détaille comment gérer la composition (au sens générique) des objets
  - De comportement : se basent sur la communication entre les objets et leurs rôles respectifs.

Les « design patterns »

- Il est important de :
  - Comprendre les « design pattern »
    ⇔ cas pratique de la COO
  - (re)connaître ces « design pattern » pour :
    - Au mieux éviter de les réinventer
    - Au pire proposer des modélisation non fonctionnelle pour l'objectif fixé

Les « design patterns »

- Certains peuvent sembler triviaux
- D'autres très techniques et abstraits ☺
- Leur point commun ⇔ proposer pour la COO des solutions :
  - de haut niveau
  - indépendante
  - fonctionnelle