Etape de Conception

à partir d'un modèle UML sous PowerAMC

Véronique Deslandres©, IUT,
Département Informatique
Université de Lyon

MàJ: mai 2019

Introduction

- En Analyse et Conception, il y a 2 étapes bien distinctes
  - L'**Analyse** effectuée avec les clients / utilisateurs
    - But : comprendre les besoins, le problème et le domaine
    - Maturation *progressive* de la connaissance
    - Englobe *plusieurs diagrammes* UML
  - La **Conception**, effectuée par les informaticiens
    - But : proposer une solution qui réponde aux besoins issus de l'analyse
    - Partir des DCL d'analyse pour produire des *DCL « de Conception »*

    + parfois des Diagrammes de Séquences très ciblés

1

Comment procéder avec l'AGL PAMC

Créer le modèle de **conception**

Création d'un DCL de Conception

- Dupliquer les DCL d'analyse pour la Conception
  - Obj.1 : conserver l'analyse **indépendamment** de la plateforme cible
    - Le DCL de Conception repose une technologie cible (java, XML)
  - Obj.2 : garder une modélisation de l'analyse **propre, lisible**
  - Or le DCL de Conception est *rarement* lisible

- 2 solutions :
  - *Générer* un MOO de Conception à partir du MOO d'Analyse via PAMC
    - Auquel cas, il faut explicitement **casser la dépendance** entre les 2 modèles
  - Dupliquer le projet sous l'OS (en dehors de PAMC), le renommer
    - **Plus simple**

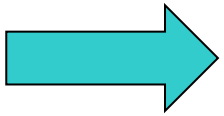
1.1 – Conception : enrichissement du DCL

- Changer le langage cible
  - Menu **Langage :** *Changer de langage objet courant*
  - Basculer du langage *Analyse* à la technologie cible (ex. java)
- On ajoute les **éléments de conception**
  - Identifiants, accesseurs, etc.
  - On précise les attributs / retours **multiples**
  - On vérifie la **navigabilité des associations**
  - On mentionne les classes persistantes
- Aussi :
  - On optimise les classes

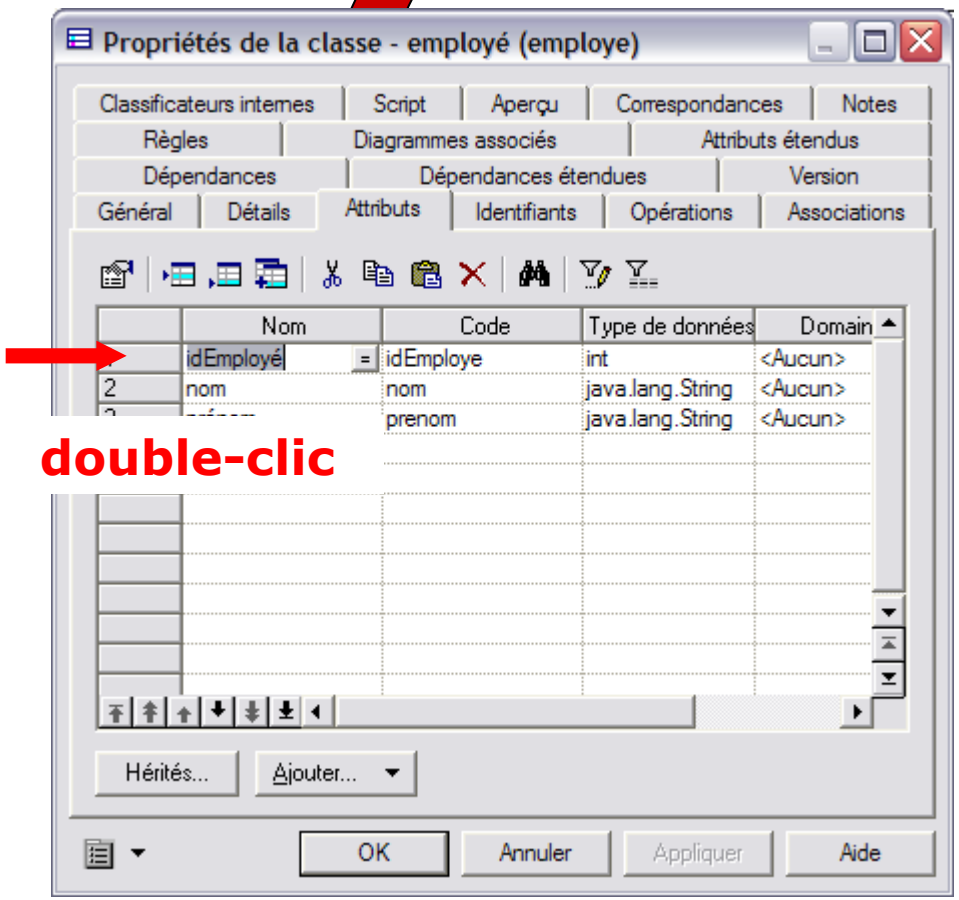
Ajouter un identifiant avec PAMC

- Fenêtre Propriété de l'attribut
- Case à cocher dans l'onglet Détail : « identifiant primaire »

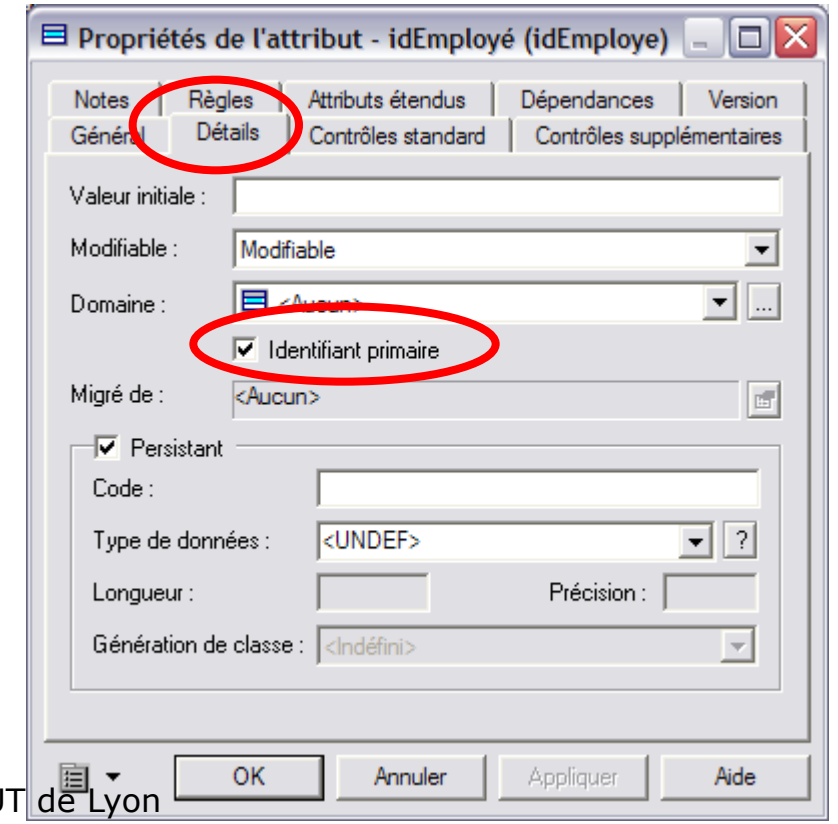
- On obtient la fenêtre Propriété en cliquant sur l'attribut de la classe dans le navigateur d'objet
  - (ou Double clic sur la ligne de l'attribut depuis la fenêtre Propriété de la Classe)



Créer un identifiant primaire



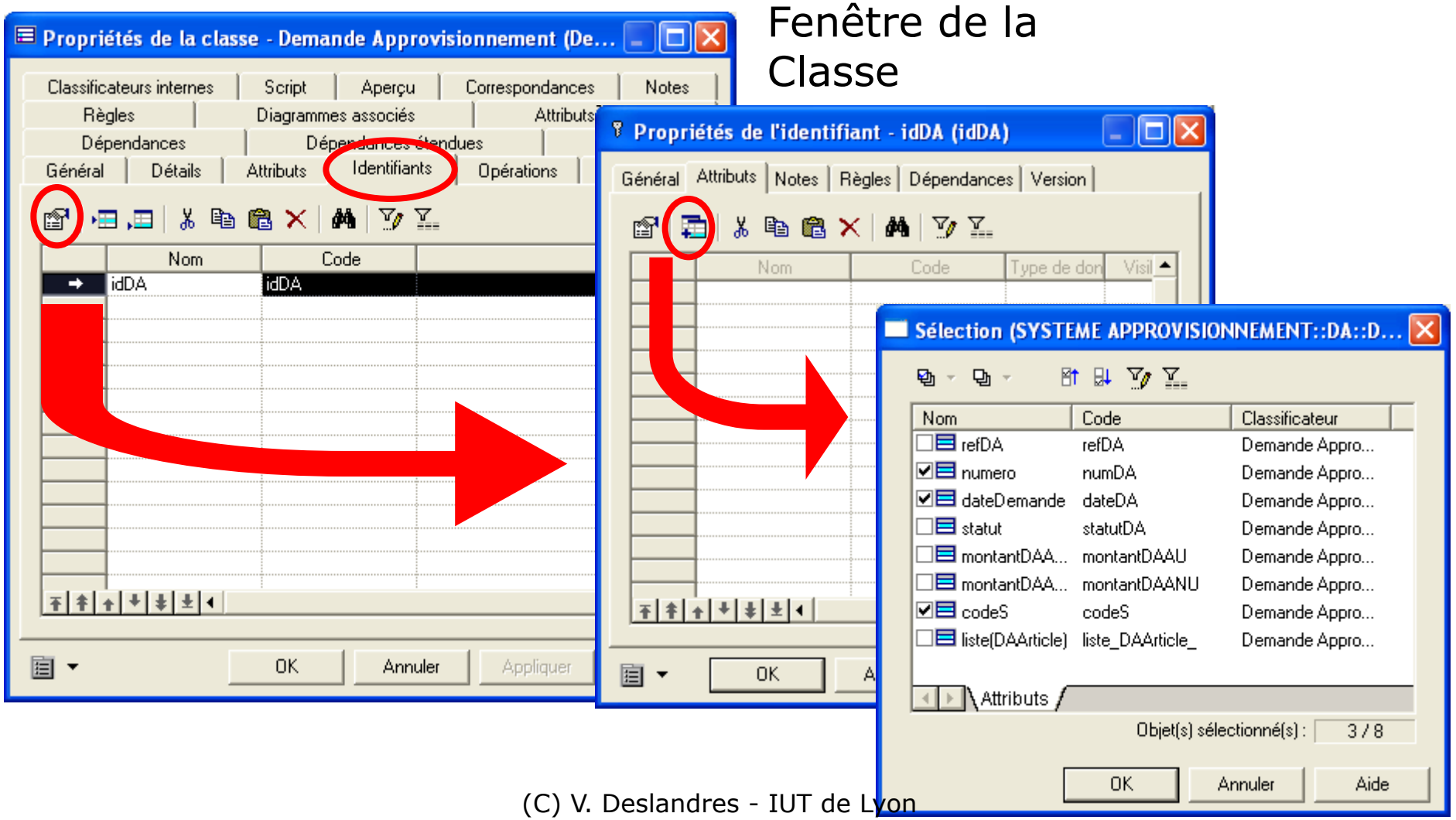
double-clic



Identifiants combinés

- Lorsqu'un identifiant est défini à partir d'une combinaison d'attributs de classe
- Créer un nouvel identifiant pour la classe (onglet Identifiant), puis cliquer sur l'outil `Afficher les propriétés`.
- Cliquer sur l'onglet `Attributs`
- Cliquer sur l'outil **`Ajouter des attributs`** et sélectionner les attributs qui composent l'identifiant

Création d'un identifiant combiné



Fenêtre de la Classe

Mentionner un attribut multiple

C'est un attribut qui peut avoir plusieurs valeurs.

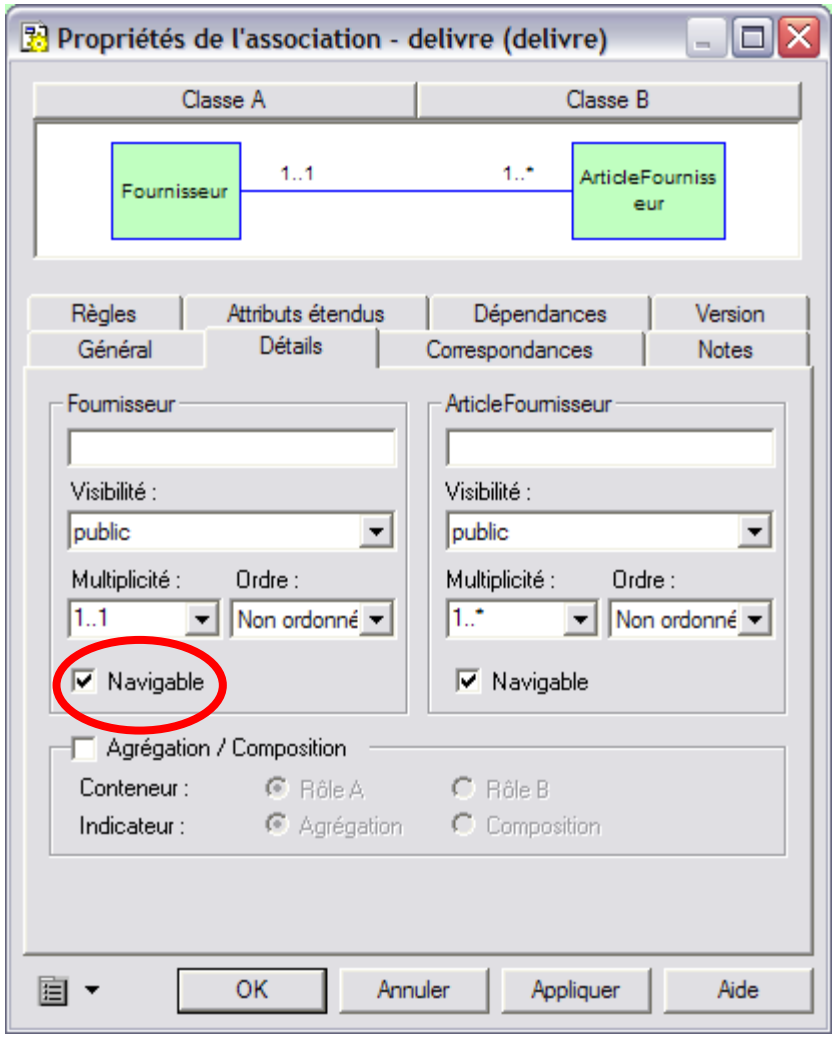
Dans PowerAMC, il faut aller dans la fenêtre de Propriétés de l'attribut, et choisir la « multiplicité » qui convient.

Ici il peut y avoir 2 adresses (par ex. une privée, une professionnelle)

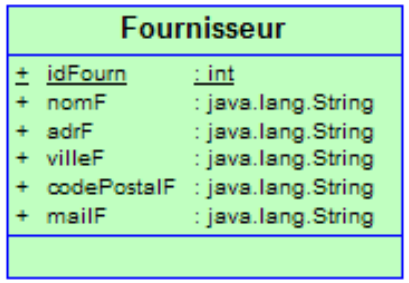
Navigabilité des associations

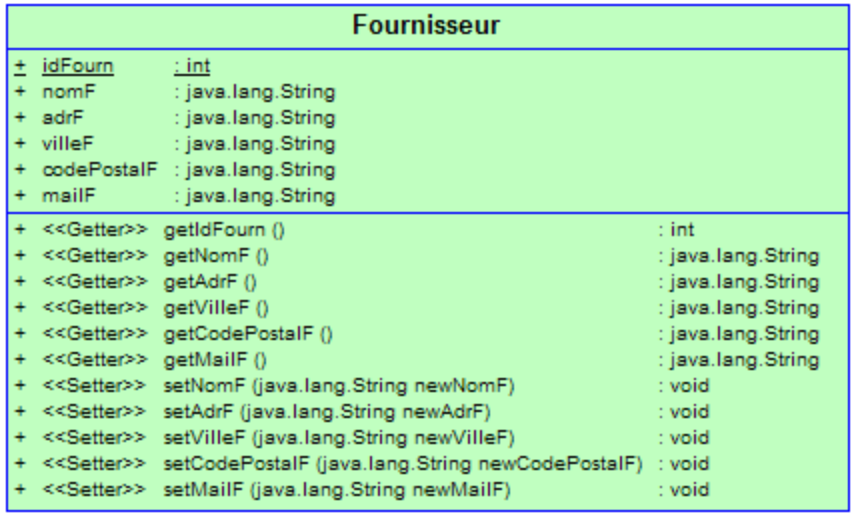
Dans **PowerAMC**, pour qu'une classe récupère les instances avec lesquelles elle est liée, il faut que les associations soient qualifiées de « navigable »

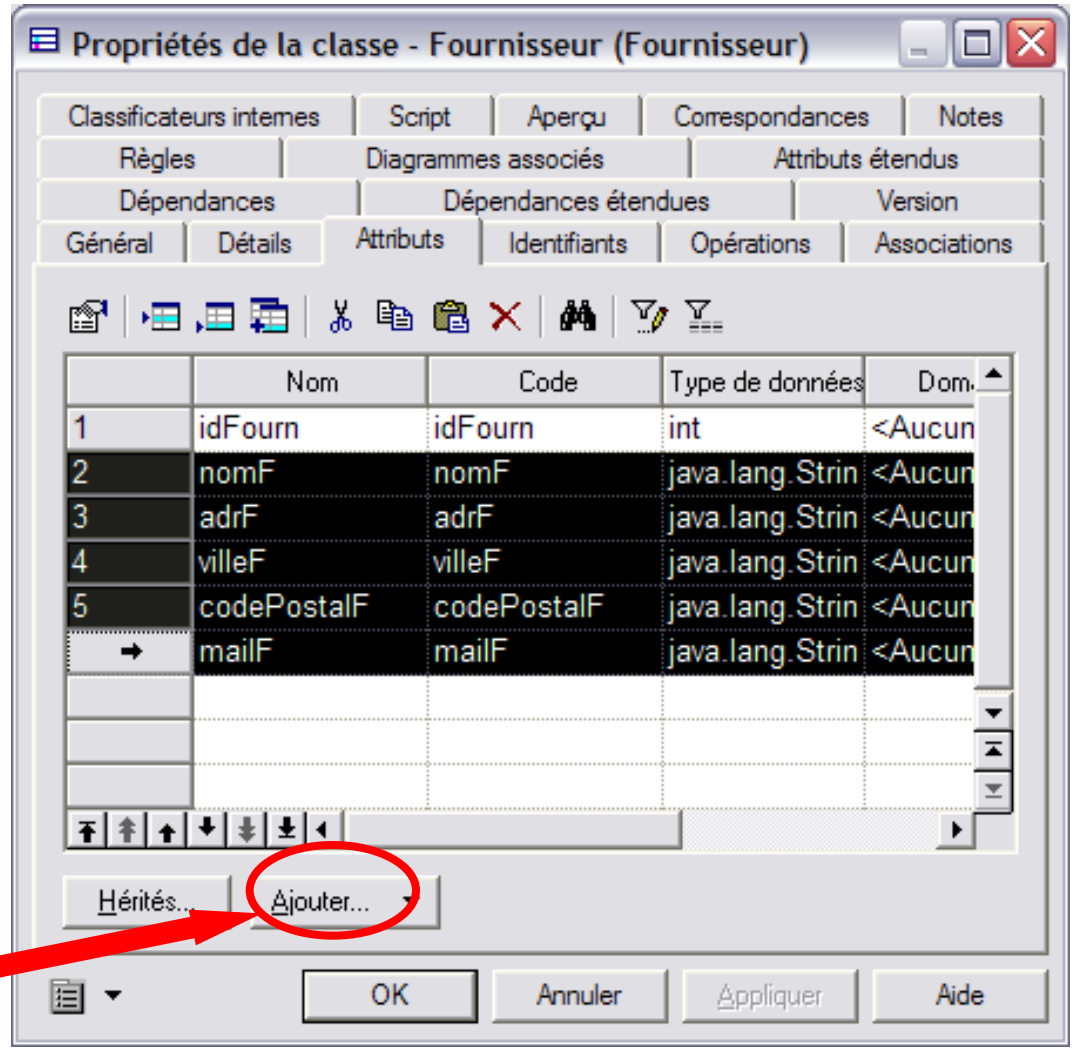
- Les **associations non navigables** sont des associations d'analyse, qui ne se traduisent pas nécessairement par un lien concret au niveau de l'implémentation.



Intégrer les accesseurs get/set

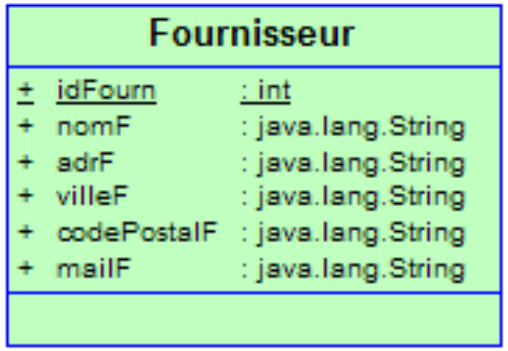


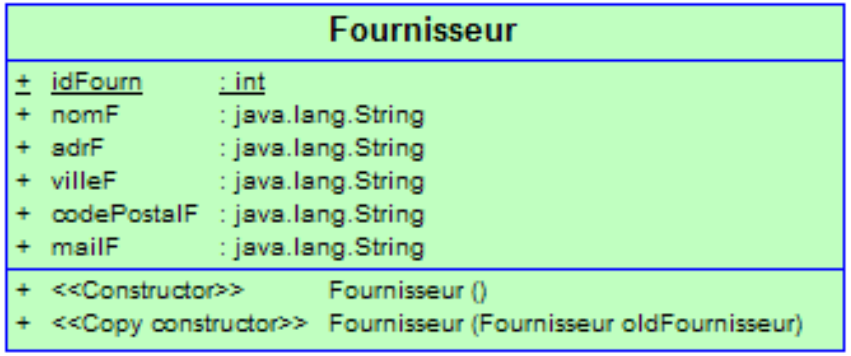




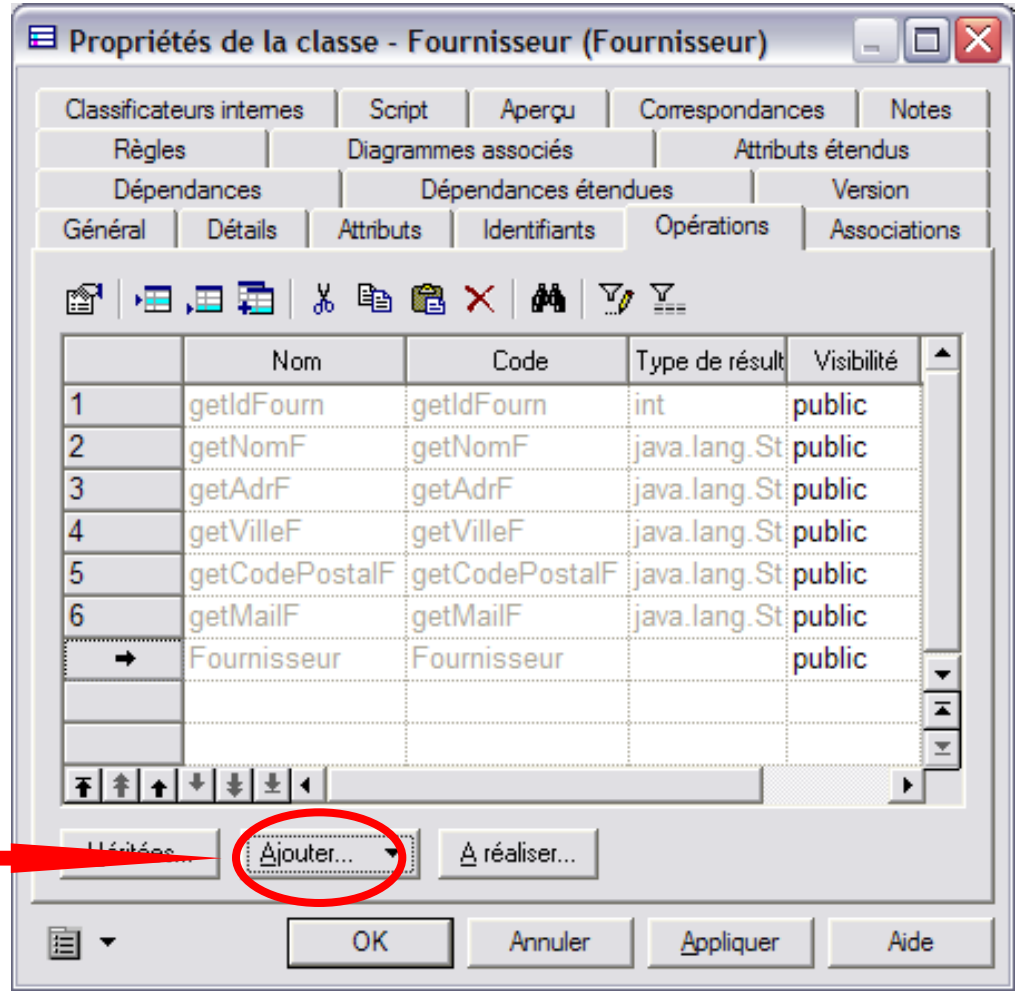
Ajouter le accesseurs get/set

Intégrer les constructeurs





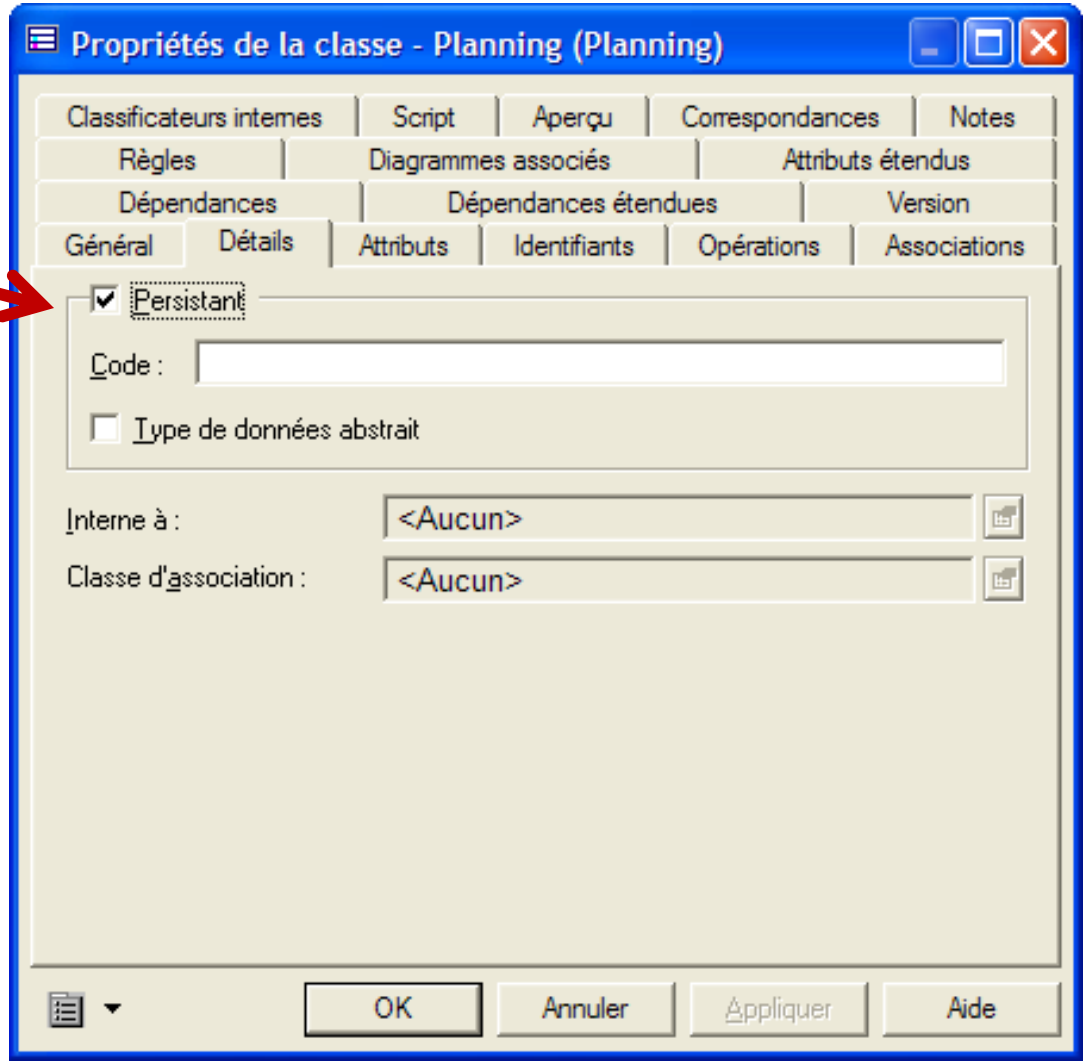
Ajouter le constructeur par défaut



Mentionner une classe Persistante dans PowerAMC

Par défaut **toutes les classes** créées dans PowerAMC sont **persistantes**

Il faudra **décocher** la case des classes non persistantes.



1.2 – Ré organiser les classes (1/2)

- **Généraliser** les classes, **spécialiser** des classes existantes par ailleurs
  - Objectif : réutilisation
- **Supprimer** des éléments du DCL d'analyse
  - Souvent en analyse, on prévoit **trop de classes**
    - Les transformer par ex. en attribut si c'est possible (pas d'autres attributs ou relations nécessaires avec la classe). Ex .: des sous-classes *VIP*, attribut *typeVIP* suffit ?
  - On a aussi parfois des **redondances d'association**
    - Vérifier leur utilité en fonction de la **fréquence des traitements**
  - Supprimer des méthodes ?
    - Certaines structures de données **simplifient les choix d'analyse →**

Importance des structures de données

Certaines structures de données réalisent les méthodes initialement envisagées

- Ex. : une classe d'analyse Contact avec un nom, un tél. (=ID) et une **méthode VeriferDoublon()**
  - Il suffit d'enregistrer la classe Contact dans un conteneur *set* de Java (automatiquement géré, ne tolère pas les doublons)

- Ex.: pour un **contrôle d'accès de salle**, on choisira une *HashTable* avec les numéros de salle (clef) et le code d'accès (valeur).

Ré organiser les classes (2/2)

- Transformer des classes en **interfaces**
  - Certaines classes ont finalement des comportements partagés par d'autres
  - On souhaite réutiliser des « comportements » existants dans d'autres applications
    - Sous la forme d'interfaces

Concevoir proprement

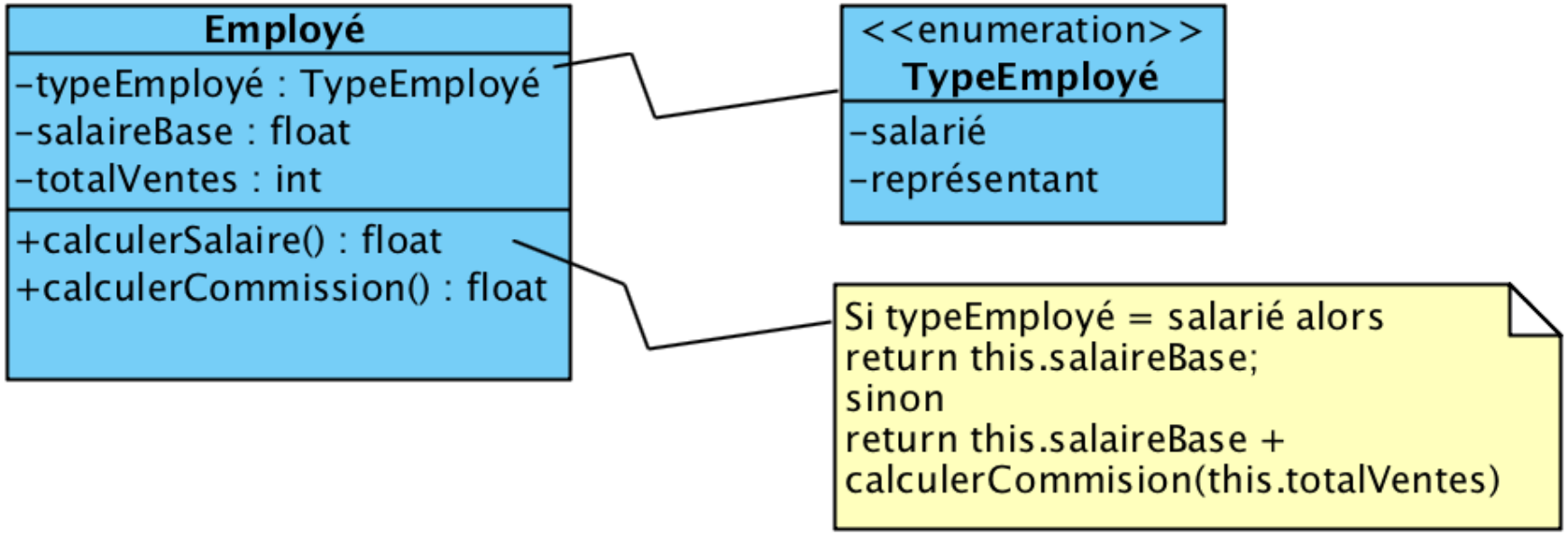
- Objectif : faciliter la maintenance du logiciel

- Par ex., respecter le **principe d'Ouverture/Fermeture (OCP)** :

« Les entités logicielles (classes, packages, etc.) doivent être **ouvertes à l'extension** mais **fermées à la modification** »

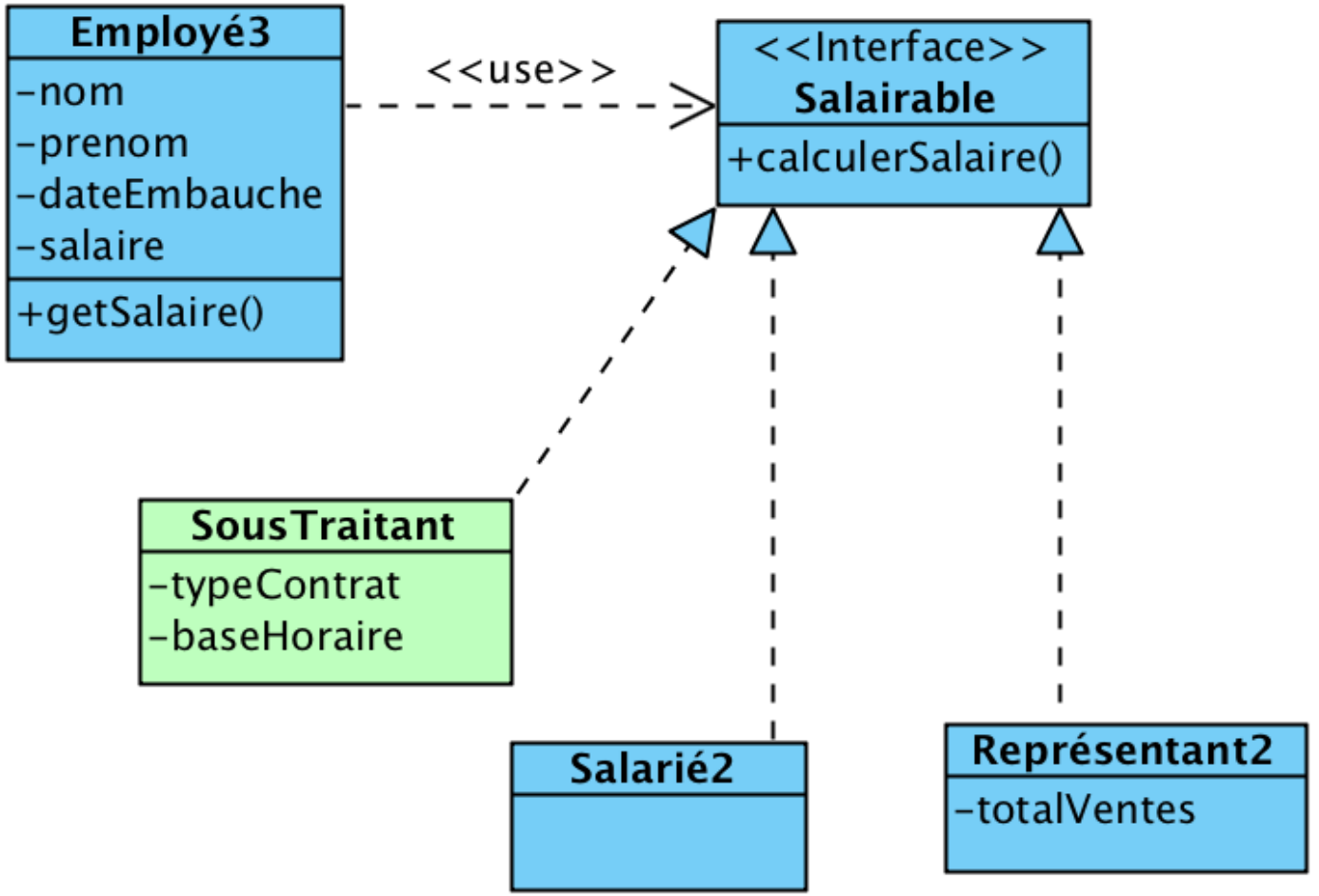
Un des principes SOLID (*Single Responsibility Principle, Open/Closed Principle, Liskov Substitution Principle, Interface Segregation Principle et Dependency Inversion Principle*)

Exemple de non respect d'OCP



Si on doit considérer un nouveau type d'employé ?

Avec le principe OCP



Concevoir proprement

- Il existe beaucoup d'autres méthodes d'optimisation de la conception que vous apprendrez plus tard
  - Notamment, les **Design Patterns** (Strategy, Composite, TemplateMethod, Factory Method, etc.)

- Avec un AGL, une fois le DCL de Conception fini, on peut **générer du code** (cours suivant)