3. Modélisation UML et programmation

Cours Stéphane Loiseau, univ. Angers

- Qu'est ce qu'UML ?
  - UML est une notation (Unified Modelling Language) issus de
    - BOOCH, d 'OMT, OOSE
    - OMG (Object Management Group)
  - UML est une méthode de construction :
    - l *'analyse*
    - la *conception*
    - en vue de la *programmation*
  - UML propose
    - des éléments de modélisation
    - un langage
    - un ensemble de règles de mise en œuvre de la méthode

- Rappel Orienté Objet
  - *classe* : un nom de concept, des *propriétés* (champs et méthodes)
    - *objet* : ses *champs*, ses *méthodes* (*opérations*)
  - L'héritage
    - La *généralisation* : processus qui factorise dans une classe les propriétés des classes considérés qui deviennent des sous classes
    - La *spécialisation* : processus qui fait dériver une sous classe d 'une classe
  - *Paquetage* regroupe des éléments cohérents, (classes…)

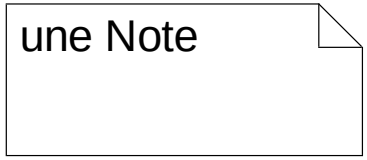
- Les Diagrammes

Un *diagramme* est une forme de modélisation UML :
une représentation visuelle d'éléments et de relations entre eux.

  - Modèle structurel (statique):
    - Diagramme de classes,
    - Diagramme d'objet.
  - Modèle dynamique :
    - Diagramme des cas d'utilisation,
    - Diagramme d'activité et d'états-transitions
    - Diagramme de communication, et [diagramme de séquences]
  - Modèle d'architecture
    - Diagramme de composant,
    - Diagramme de déploiement.

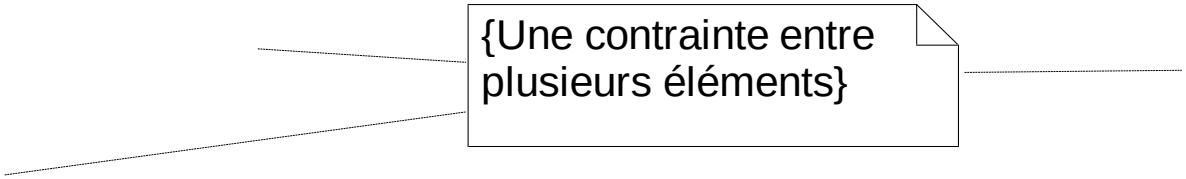
- Nota: un modèle est une description abstraite d'un système ou processus

- Quelques *éléments* visuels de base
  - une *Note* : mettre un commentaire qui explique un élément UML



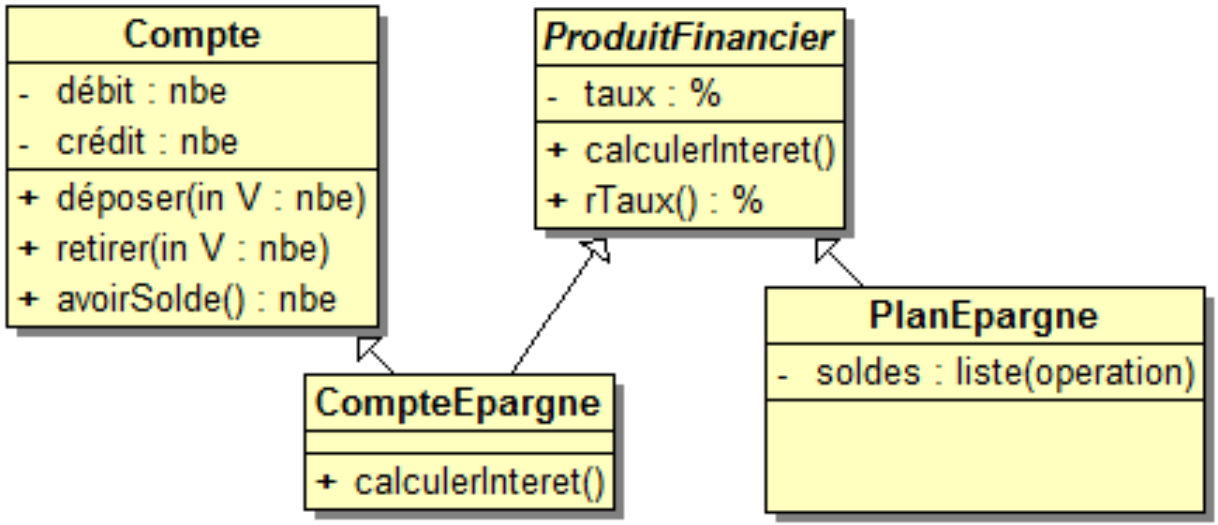
  - une *Contrainte :* texte qui a une valeur sémantique sur un ou des éléments

```
{ceci est une contrainte sur l'élément ou je suis}
```



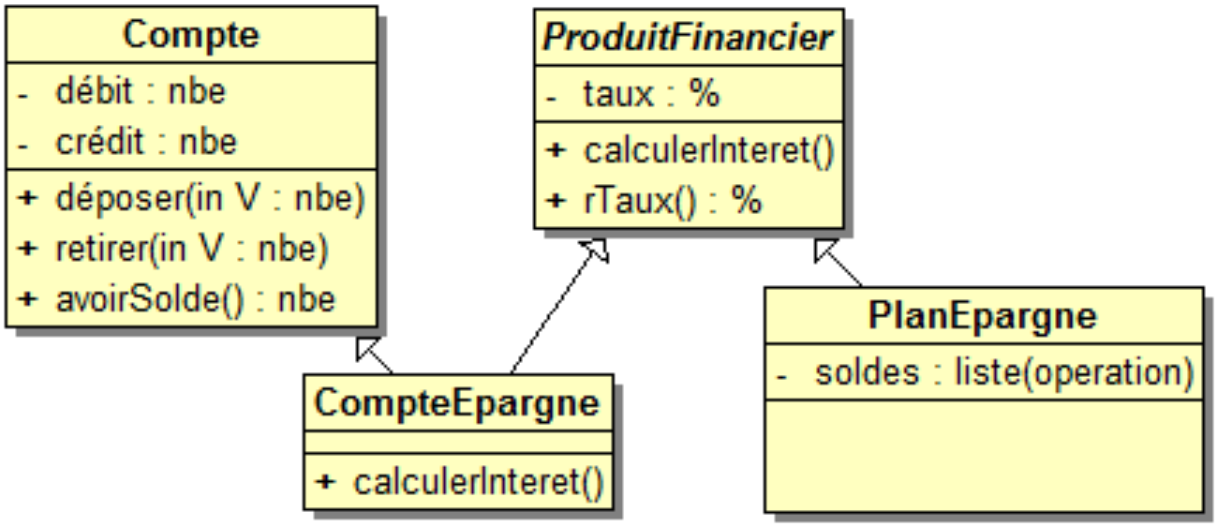
  - Exemples de *classes*, *paquetage*, *relation de dépendance* entre source et cible (cible dépend du contenu de source)

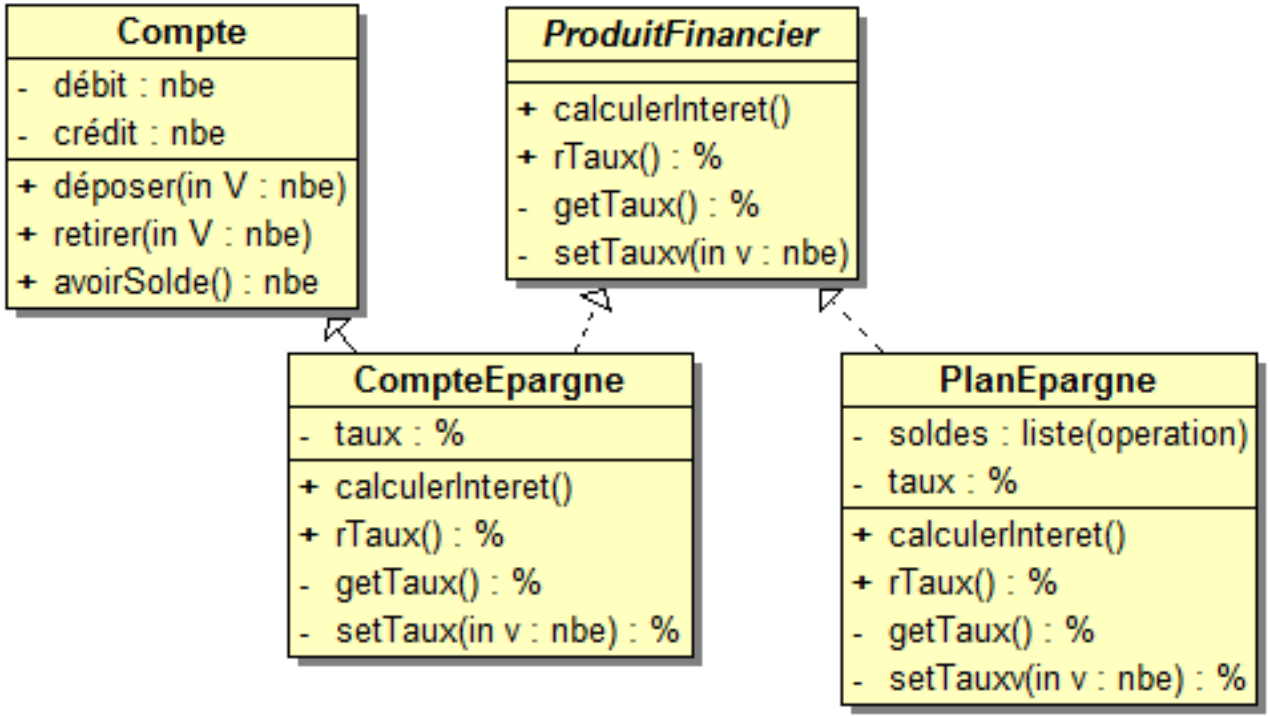
A-. Transformation d'UML basique vers un langage de programmation quelconque



UML (C++)

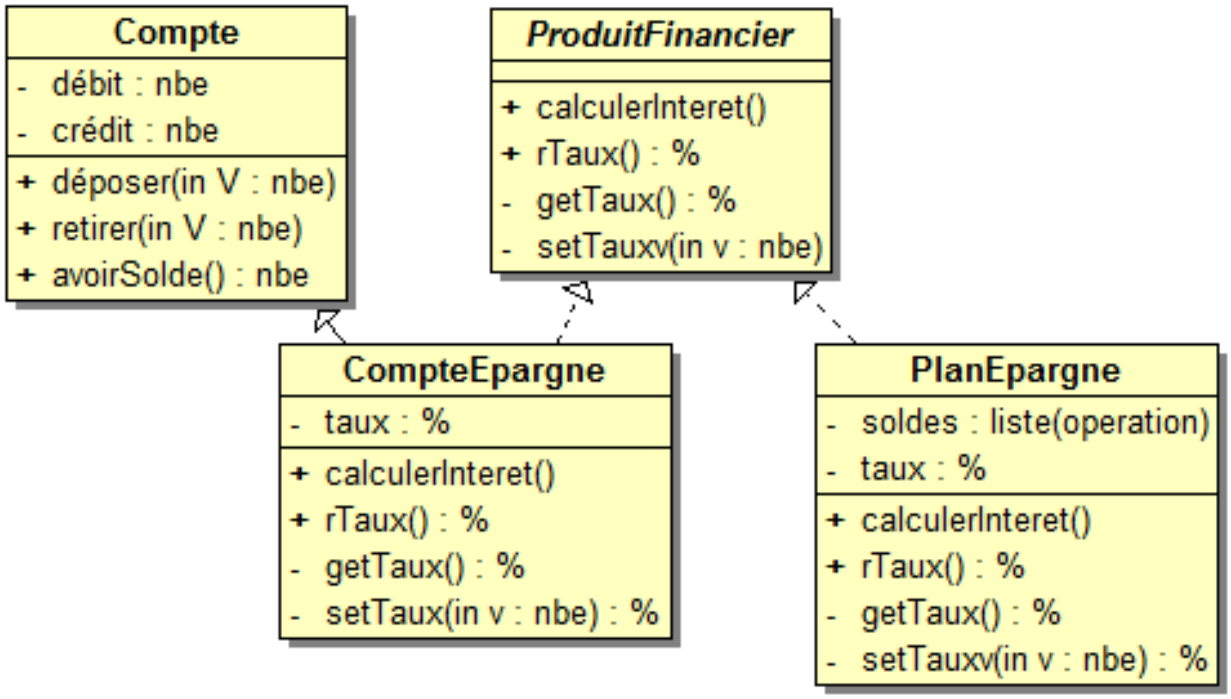
UML-> interfaces
(JAVA)

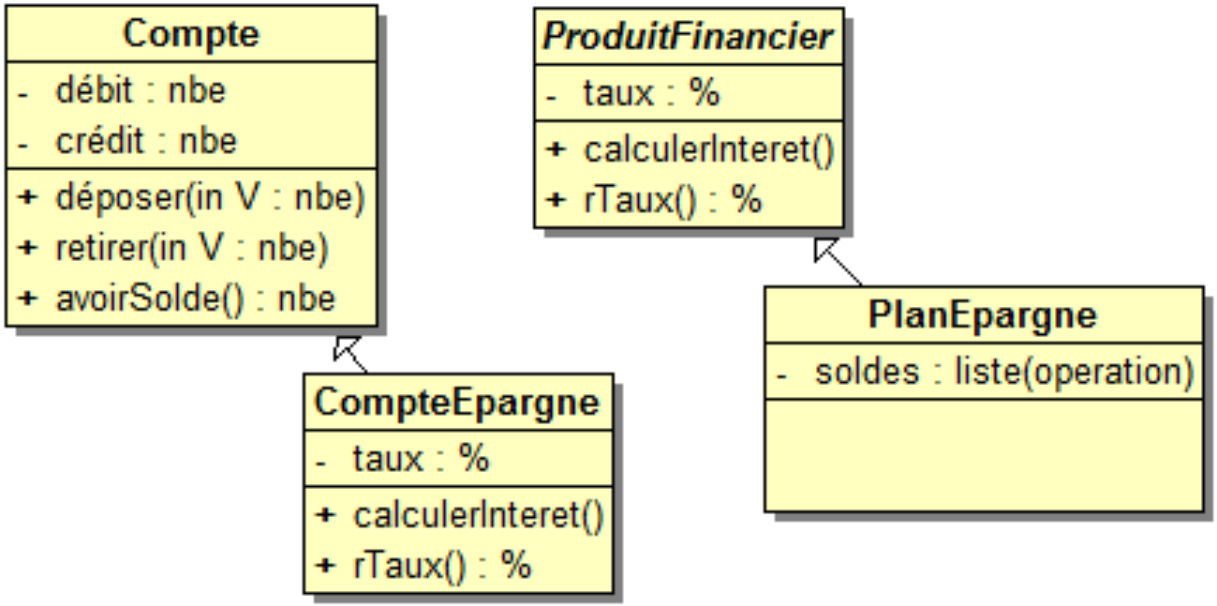


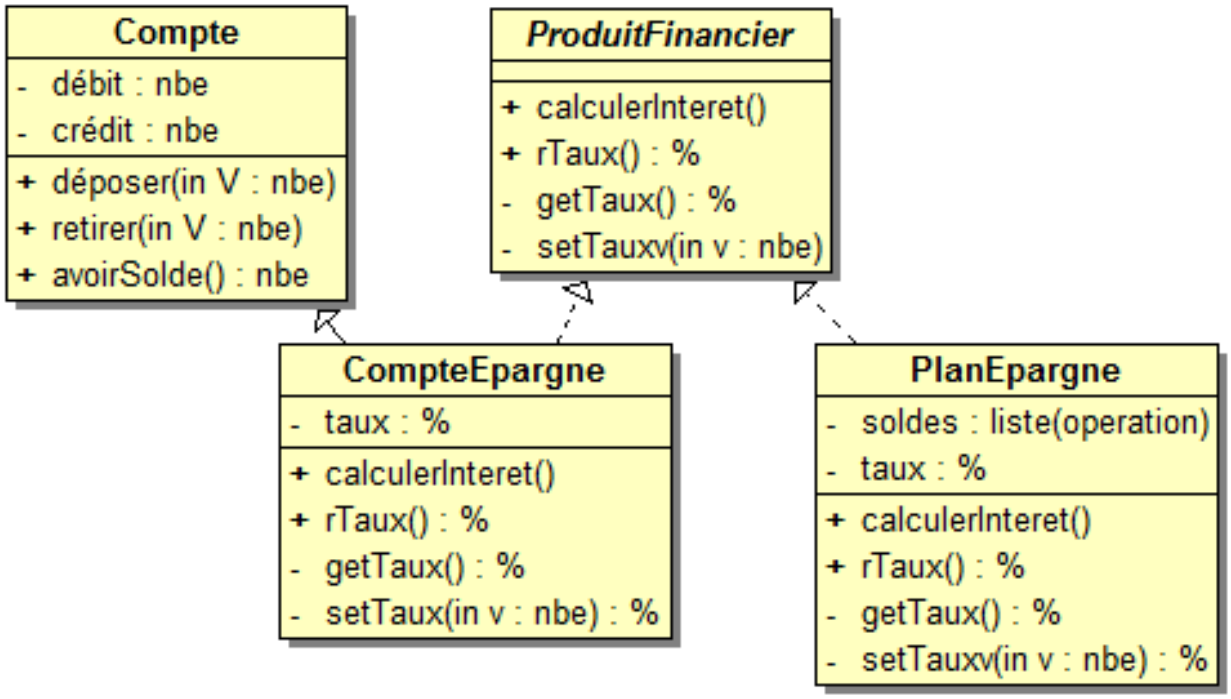


Sol1
Interface -> classe
(SmallTalk)

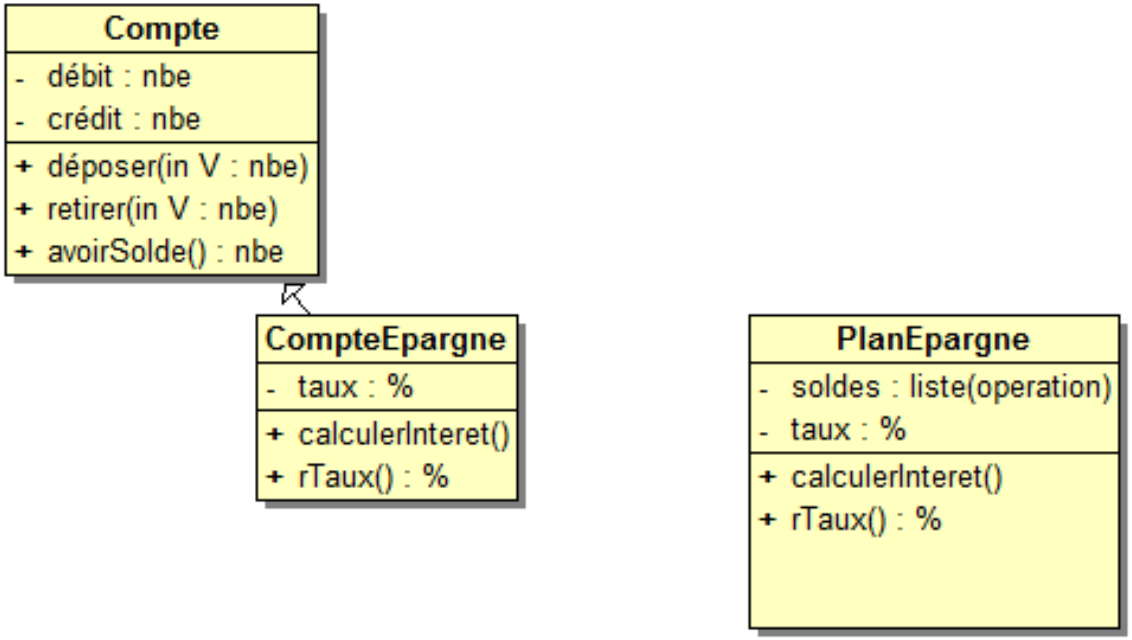
On ne tient plus compte des accesseurs pour plus de simplicité



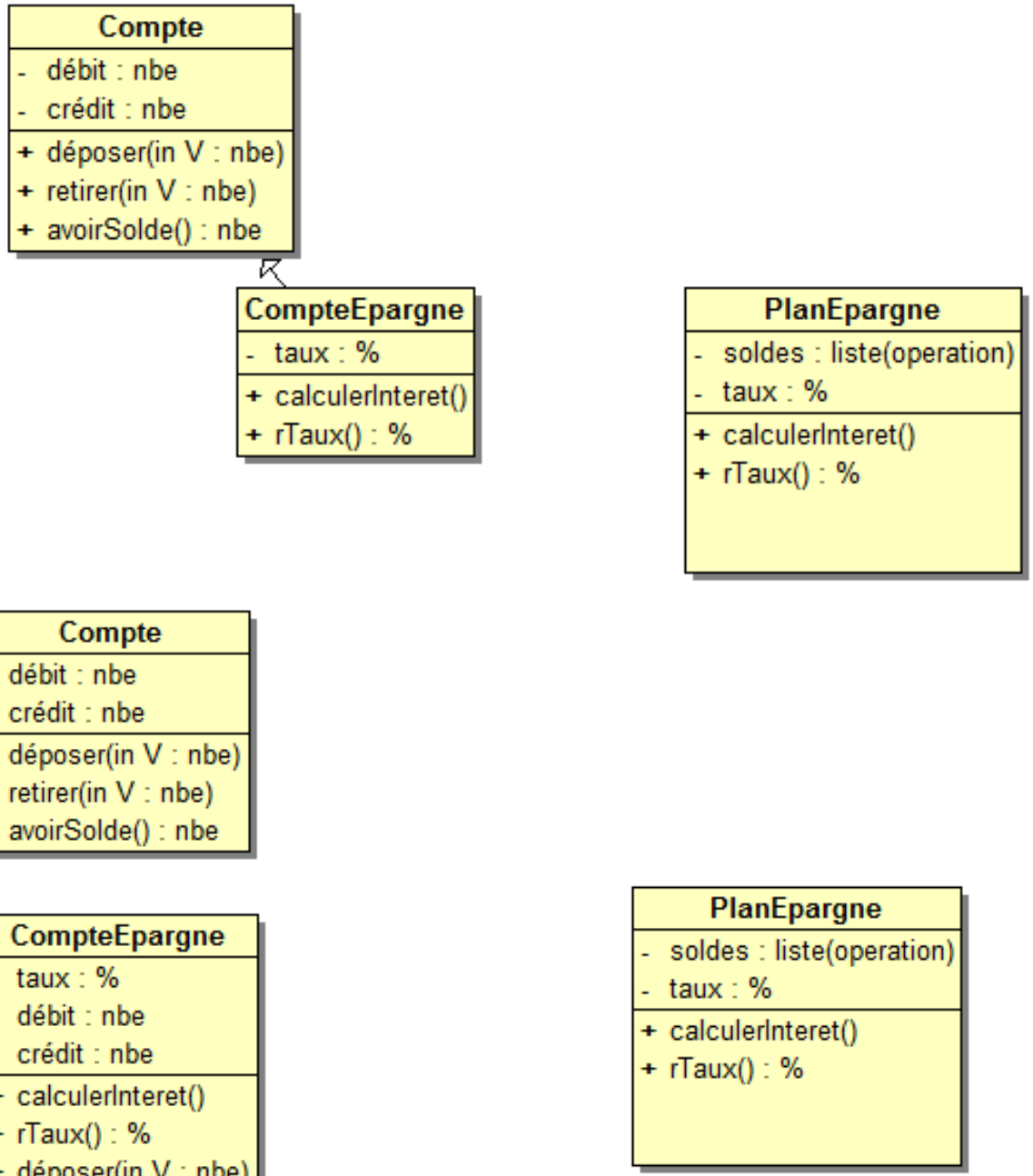




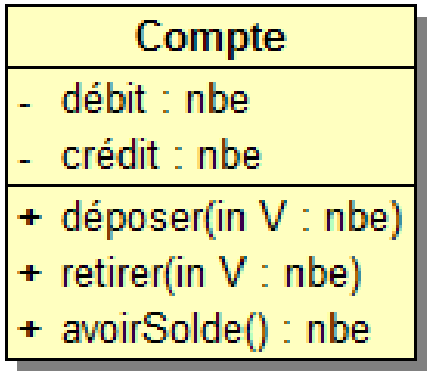
Sol2
Interface -> classe
(SmallTalk)



Sol2
classe (SmallTalk)
-> classe Sans Héritage
(ADA)



classe Sans Héritage
(ADA)
-> Sans Classes
(PASCAL)




```
Type Compte
  debit: nbe, credit: nbe
FinType
Procedure déposer(var O Compte, V:nbe) ...
Procédure retirer(var O Compte, V: nbe)...
Fonction(var O Compte): nbe ...
```

...

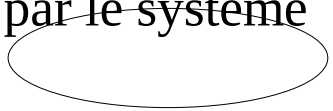
B. Diagramme de cas d’utilisation

- Objectif : A partir des besoins des utilisateurs, spécifications du comportement du système pour eux.

- Faires Le diagramme =
    - Quels sont les cas d’utilisation : séquences d’action, traitements attendus…
    - Identifiés les acteurs, leurs nombres, leurs connaissances, leurs rôles…
    - Structuré les cas d’utilisation en paquetage en mettant les cas en relation
    - Les autres diagrammes dynamiques détaillent le diagramme de cas d’utilisation

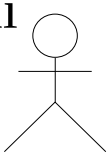
1. Cas d’utilisation et acteurs

- Définition: un *cas d ’utilisation* modélise un service rendu par le système



Cas d’utilisation

- Définition: un *acteur* est une personne ou un élément externe à lui qui interagit avec le système

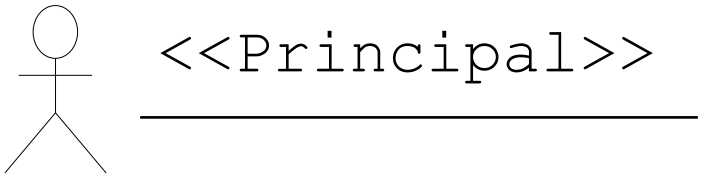


Acteur

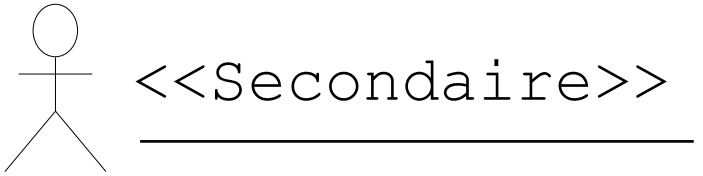
Différents acteurs

- Définitions

  L'acteur principal, est l'acteur à qui un cas d'utilisation rend service
  Les autres acteurs sont appelés *acteurs secondaires*.
- Règle: un seul acteur principal par cas d'utilisation



Acteur



Acteur

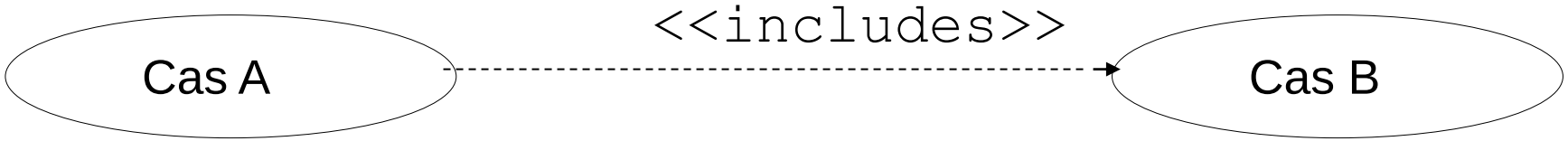
- Remarques:
  - En général, l'acteur principal initie le cas d'utilisation par ses sollicitations
  - Acteur principal obtient un résultat observable.
  - Acteur secondaire est sollicité pour des informations supplémentaires

- 2. Relations entre cas d’utilisation et acteurs

  – l’inclusion (une dépendances stéréotypé)
  – l’extension (une dépendances stéréotypé)
  – La généralisation/spécialisation
  – Les relations entre acteurs

La relation d'inclusion entre cas

- Définition:
  une *relation d'inclusion* exprime qu'un cas d'utilisation (A) incorpore un autre cas (B).
- Règles:
  - Le cas qui **incorpore** l'autre ne peut pas être utilisé seul.
  - Si A inclut B, lorsque A est sollicité, B l'est obligatoirement

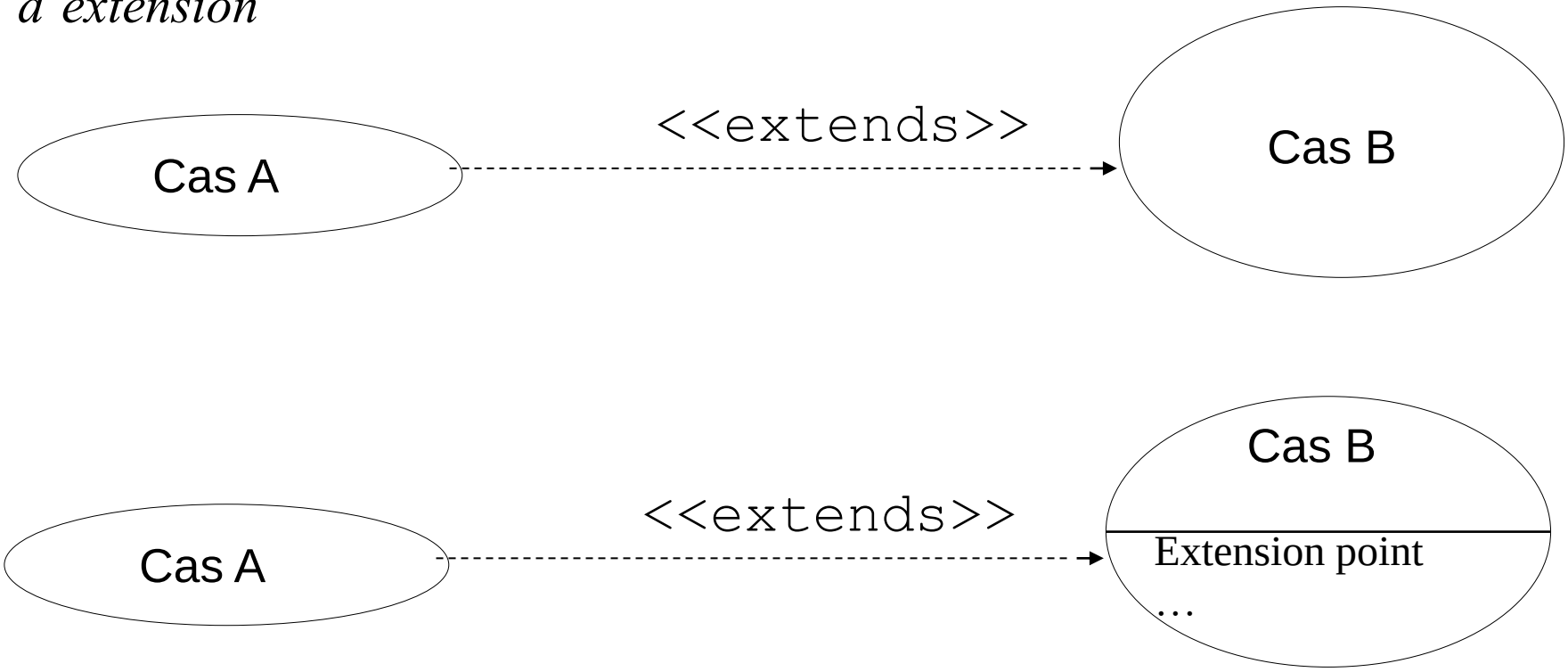


- Note : les cas d'utilisation ne s'enchainent pas: pas de représentation temporelle.

La relation d'extension entre cas

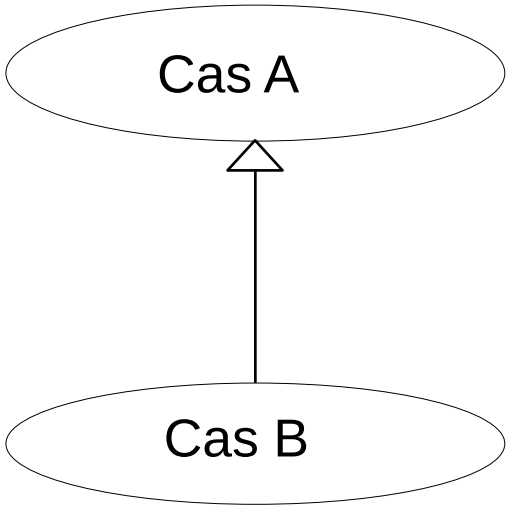
- Définition:
  Une *relation d'extension* exprime qu'un cas (A) étend un autre cas B
- Règle: A **peut être appelé** au cours de B.

- L'extension peut intervenir à un point précis du cas étendu : le *point d'extension*



La Relation de généralisation/Spécialisation

- Définition
  Un cas A est une *généralisation* de B, si B **est un cas particulier** de A



Relations entre acteurs

- Définition:
  L'acteur A est une *généralisation* de l'acteur B si A peut être remplacé par B.
- Règle:
  Tous les cas d'utilisation accessible par A **sont accessibles** par B. L'inverse n'est pas vrai