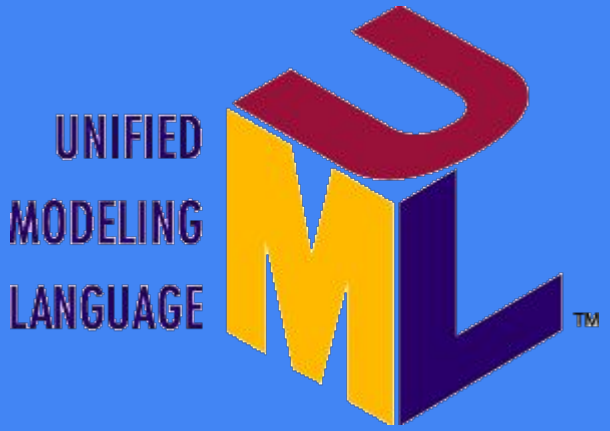
Modélisation avec UML

Samuel Toubon



Plan du cours

1. Vue d’ensemble
2. Le langage

Vue d’ensemble

Vue d’ensemble

1. Qu’est-ce que c’est ?
2. Pourquoi ?
3. Un peu d’Histoire
4. UML aujourd’hui
5. UML et Java

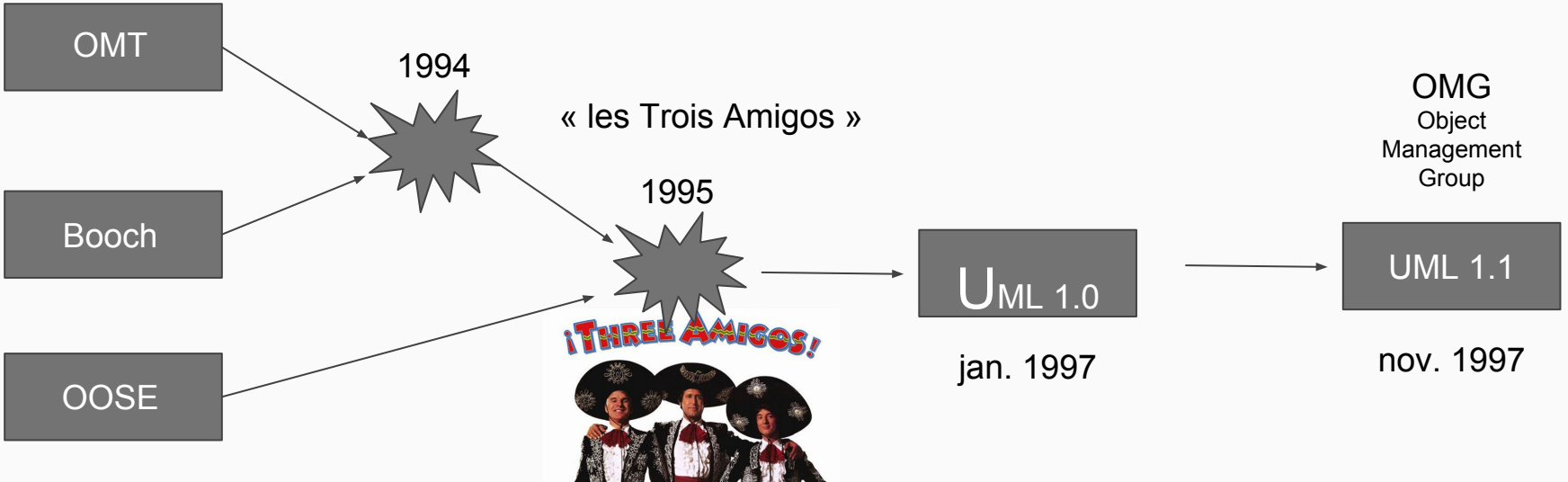
Qu’est-ce qu’UML ?

- Unified **Modeling Language**
- Une méthode de **modélisation**.
- Un ensemble de notations semi-graphiques et de règles : c’est la **syntaxe**.
- Un méta-modèle qui donne un sens à ces éléments : c’est la **sémantique**.
- Des outils qui s’y adaptent.

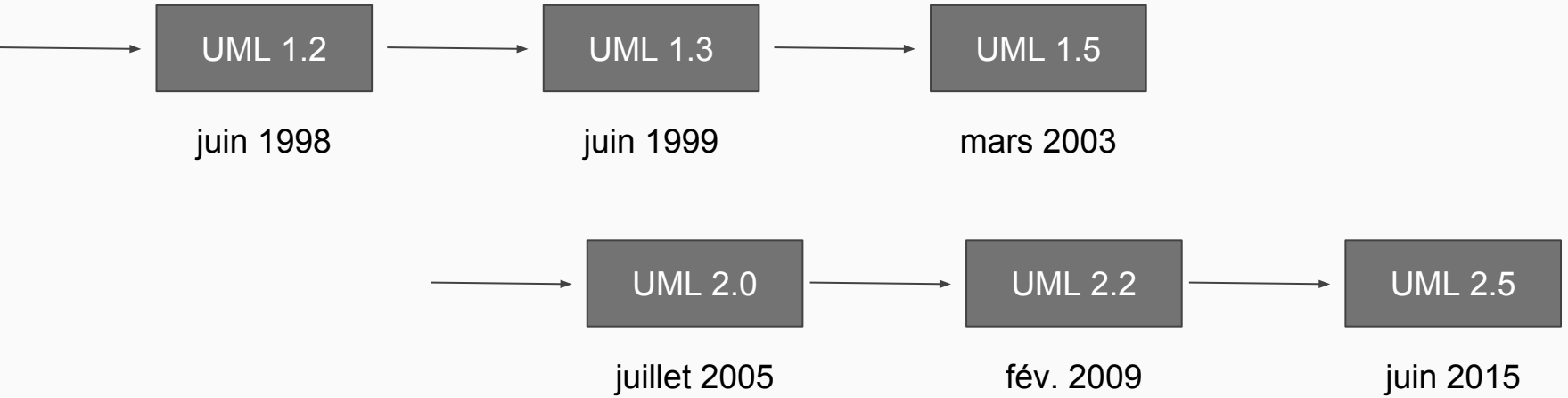
Pourquoi ?

- Aider à la réflexion.
- Savoir décrire ce que l'on veut faire, formaliser les besoins.
- Organiser les fonctionnalités.
- Documenter.
- **Communiquer avec les informaticiens !**

Un peu d'Histoire



Un peu d’Histoire



UML aujourd’hui

- Le langage de modélisation le plus connu et le plus utilisé.
- Peu d’utilisateurs connaissent vraiment le standard.
- UML est critiqué car pas assez formel.

10 ans de projets à l'Insee : quels enseignements ? Mai 2015

- L'intégralité des bilans de projet mentionnant l'utilisation d'UML le font de manière positive.
- Si l'utilisation d'UML nécessite bien souvent un temps de formation (voire de « tutorat ») non négligeable, UML est un outil très utile pour **« pouvoir échanger simplement » entre équipes statistiques et informatiques**.
- Le formalisme UML possède l'avantage « d'obliger la MOA à formuler ses problèmes et ses nouveaux besoins de façon précise afin qu'ils soient compris par l'équipe informatique ».
- L'erreur principale à absolument éviter est de « développer sans spécifications fonctionnelles détaillées ».

UML et Java

- UML n’est pas une syntaxe graphique pour Java !
- Mais c’est un langage basé sur l’approche objet : classe, encapsulation, héritage, polymorphisme, composition *etc* sont au coeur d’UML.

Néanmoins il existe des outils imparfaits :

- UML vers Java : génération de code
- Java vers UML : rétro-ingénierie

Le langage

Le langage

1. Diagramme de cas d’utilisation
2. Diagramme de classes
3. Diagramme d’objets
4. Diagramme de paquetages
5. Diagramme d’état-transition
6. Diagramme d’activité
7. Diagramme de séquence

Le langage

axe **fonctionnel**

axe **statique**

axe **dynamique**

1. Diagramme de cas d'utilisation
2. Diagramme de classes
3. Diagramme d'objets
4. Diagramme de paquetages
5. Diagramme d'état-transition
6. Diagramme d'activité
7. Diagramme de séquence

ordre du cours

1. Diagramme de cas d’utilisation

2. Diagramme d’activité
3. Diagramme d’état-transition

4. Diagramme de classes
5. Diagramme d’objets
6. Diagramme de paquetages

7. Diagramme de séquence

Diagramme de cas d’utilisation

Diagramme de cas d’utilisation

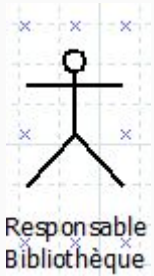
- Répond aux questions **« Qui peut faire quoi ? »** et **« Quelle est la limite du système ? »**.
- Une base pour le cahier des charges.

- Qui = les acteurs
- Quoi = les cas d’utilisations

Les acteurs

- Ils sont externes au système.

Acteur humain



Autre acteur
(un programme externe, du matériel…)

rôle de l'utilisateur par rapport au système

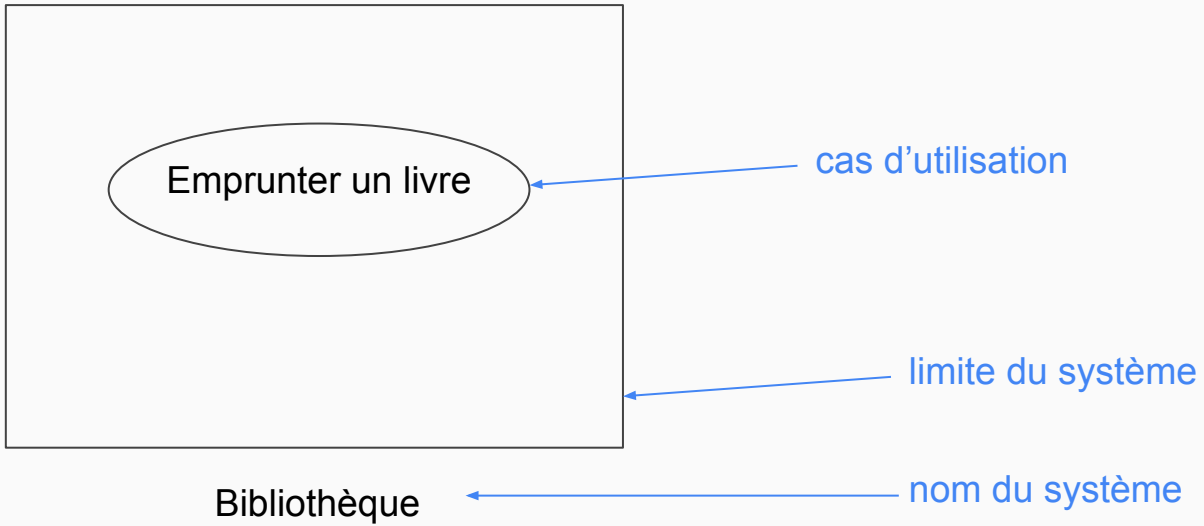
Les acteurs

- Ne pas confondre les notions d'acteur et de personne utilisant le système.

    - Une même personne peut jouer plusieurs rôles.
      Maurice est directeur mais peut jouer le rôle de guichetier.
    - Plusieurs personnes peuvent jouer un même rôle.
      Paul et Pierre sont deux clients.

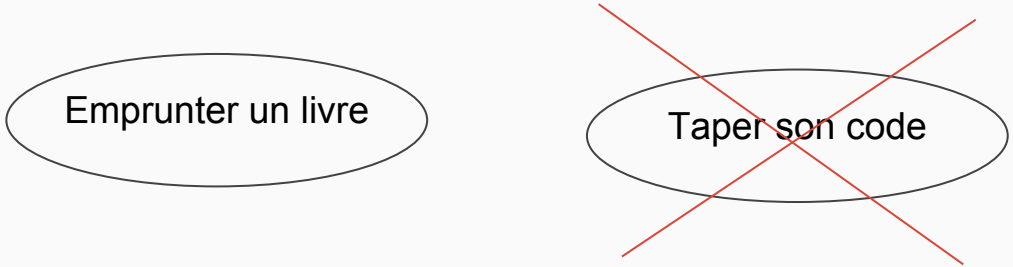
Les cas d’utilisation

- Un cas représente une fonctionnalité du système, c’est-à-dire une manière de l’utiliser visible de l’acteur.



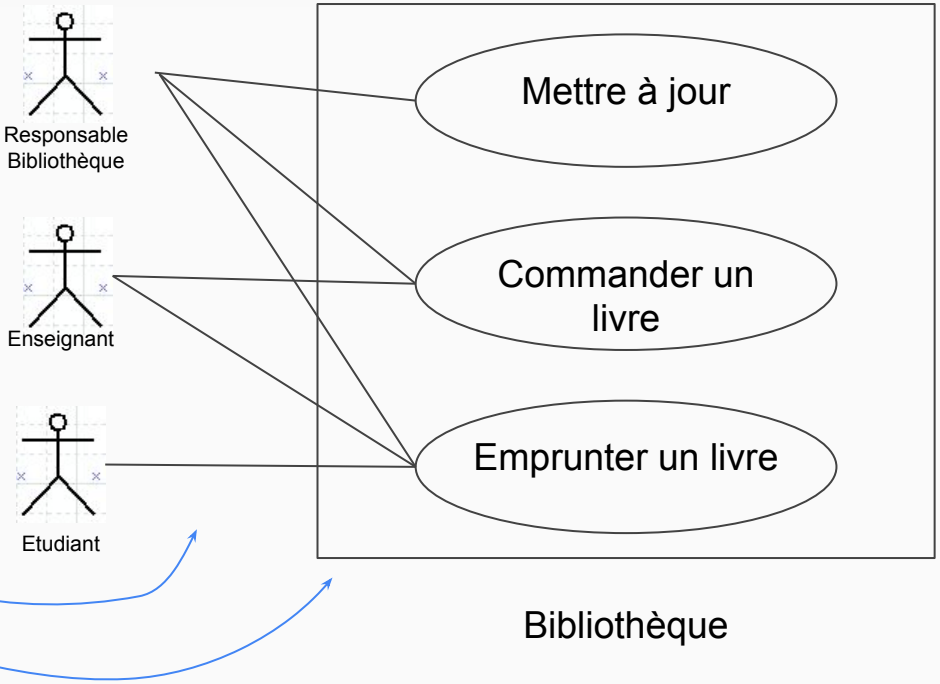
Les cas d’utilisation

- Un cas d’utilisation doit être utile en soi.



Un exemple

- 3/7 à 10/12 CU par diagramme.

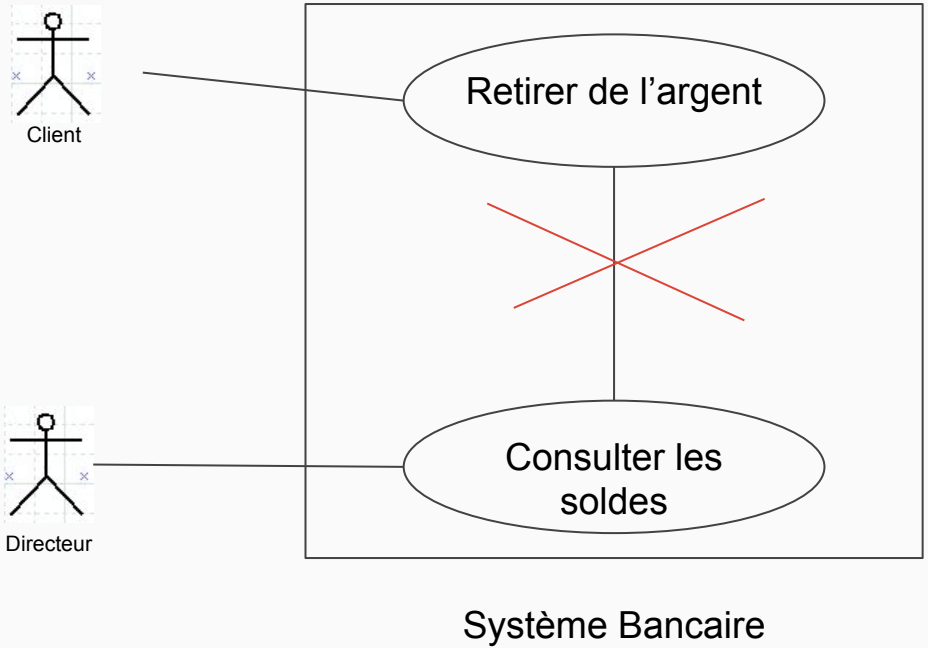


Qui peut faire quoi ?

Quelle est la limite du système ?

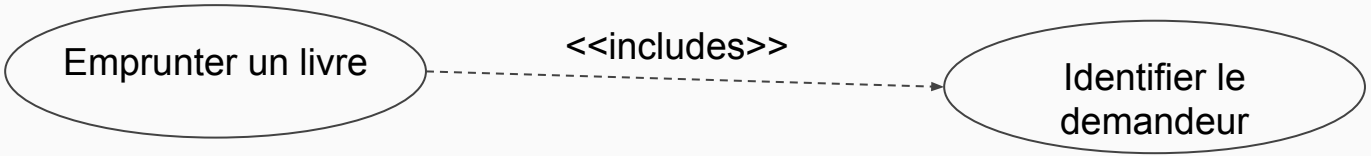
Associations internes non modélisées

- Ce diagramme se concentre sur la description du système et de ses interactions avec l'extérieur.
- Formalisme en général bien trop pauvre pour décrire l'intérieur du système.
- Quelques exceptions slides suivantes.



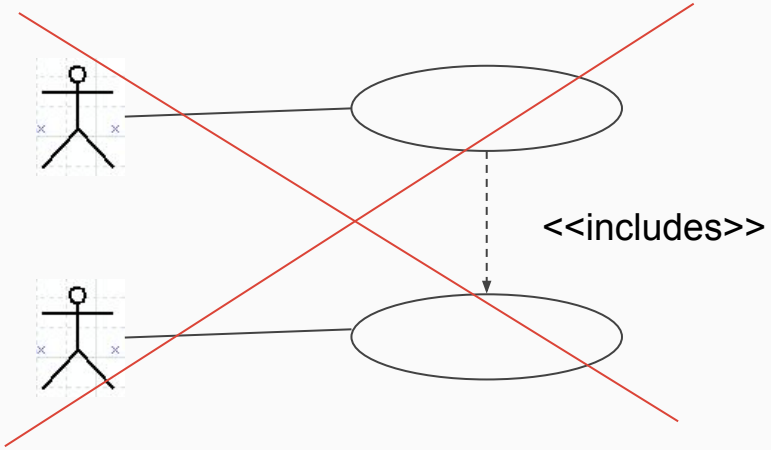
Association *includes*

- Association entre cas d'utilisations pour **détailler** ou **partager**.
- Relation cause-conséquence : quand on fait A alors on fait B.
- Pas de notion de temps explicite.

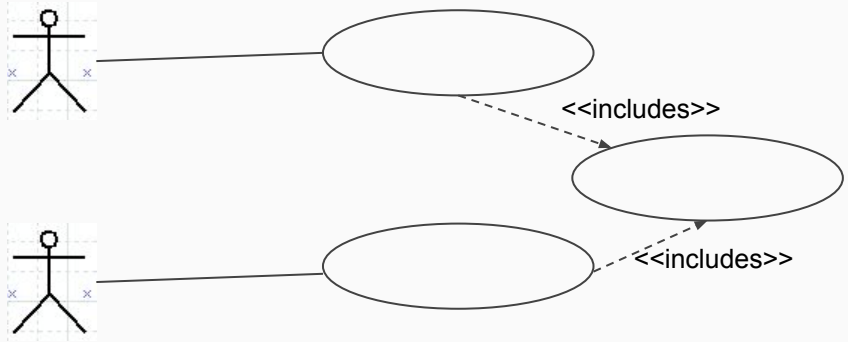


Association *includes*

- Impossible d’accéder directement à un cas inclus.



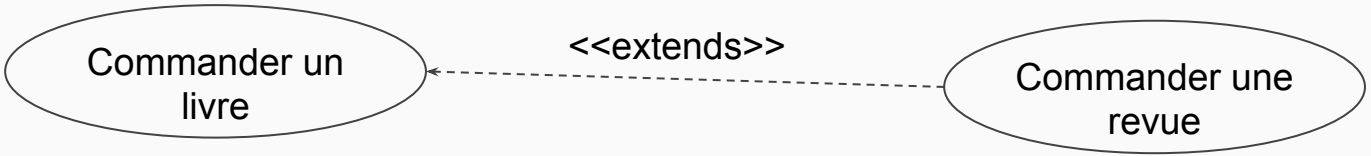
FAUX !



CORRECT

Association *extends*

- Association entre cas d'utilisations.
- Relation cause-conséquence dépendant du scénario : quand on fait A alors, *de manière optionnelle*, on peut faire B.
- Pas de notion de temps explicite.



attention au sens de la flèche !

Héritage entre acteurs

- Le directeur peut faire tout ce que peut faire le client.
- Il peut donc prendre le rôle de client pour retirer de l'argent (sur son propre compte).

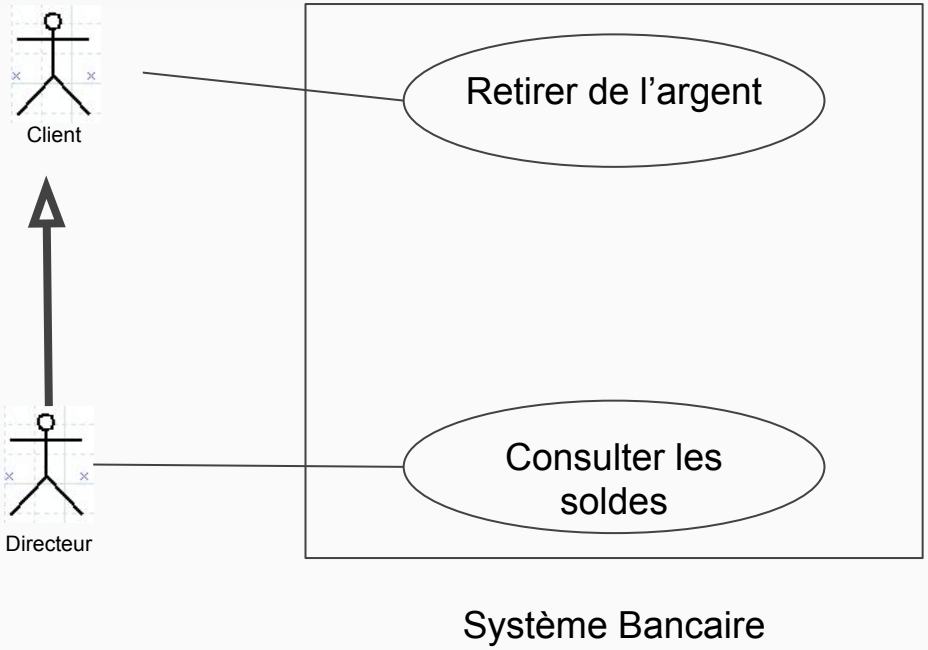
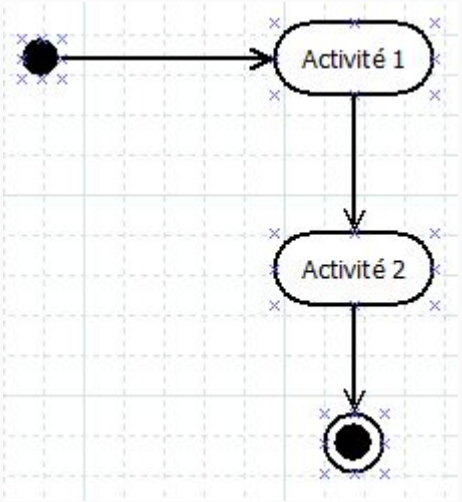


Diagramme d’activité

Diagramme d’activité

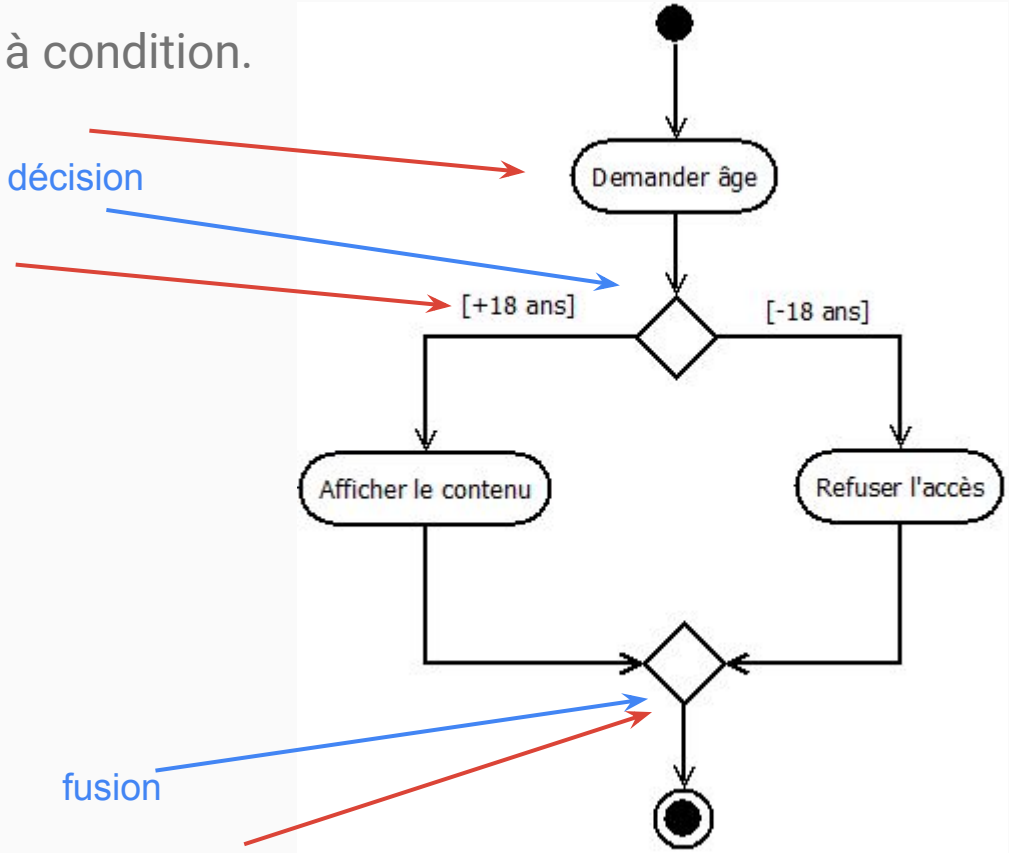
- Détaille le comportement interne d’un cas d’utilisation, d’un ensemble de cas d’utilisation, ou d’une méthode.
- Indique l’enchaînement des opérations.



- En entrée/sortie d’activité : exactement 1 **enchaînement**.
- Les enchaînements sont **automatiques**.

Condition

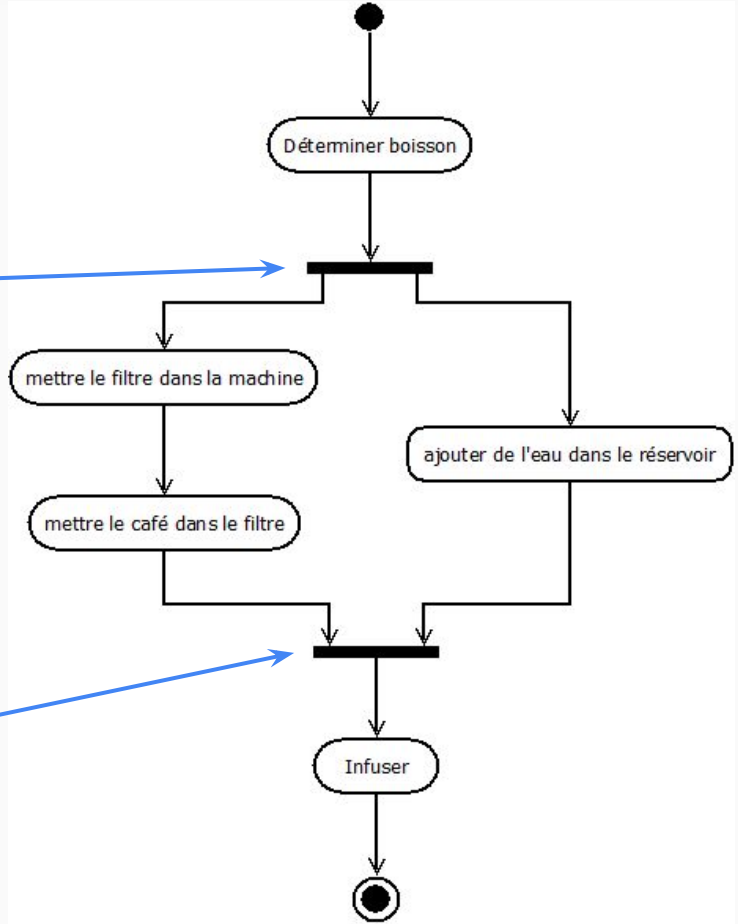
- Changement d'activité soumis à condition.



Synchronisation

barre de synchronisation

barre de fusion



Partitions

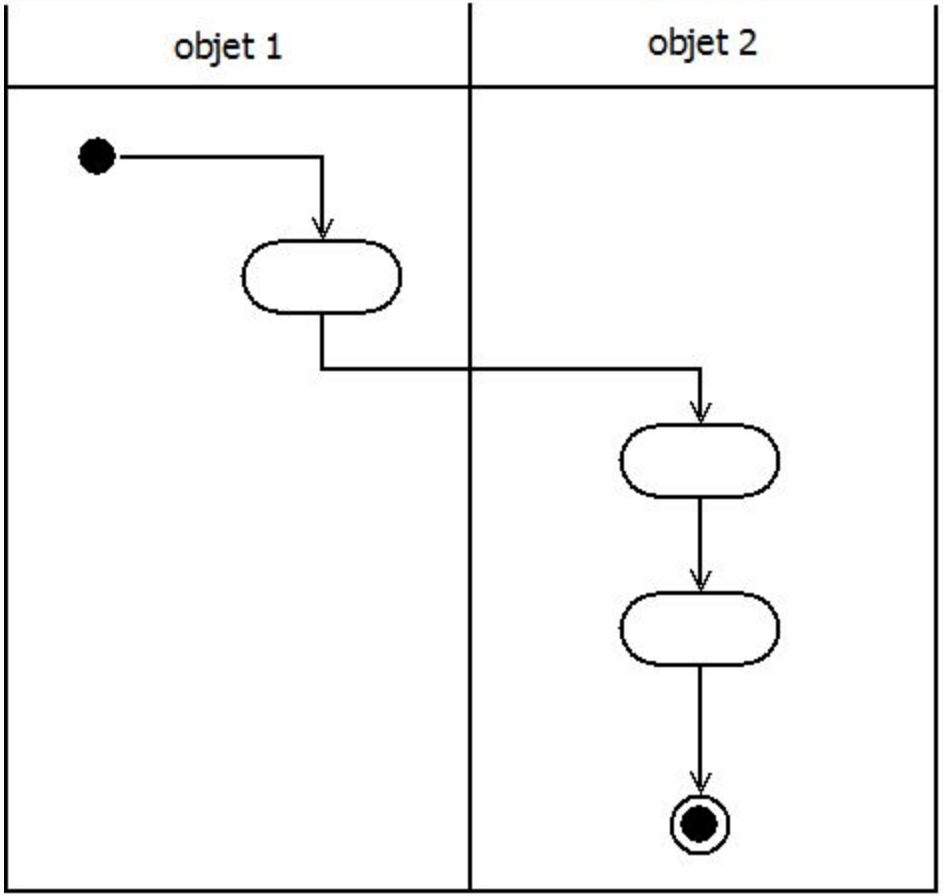
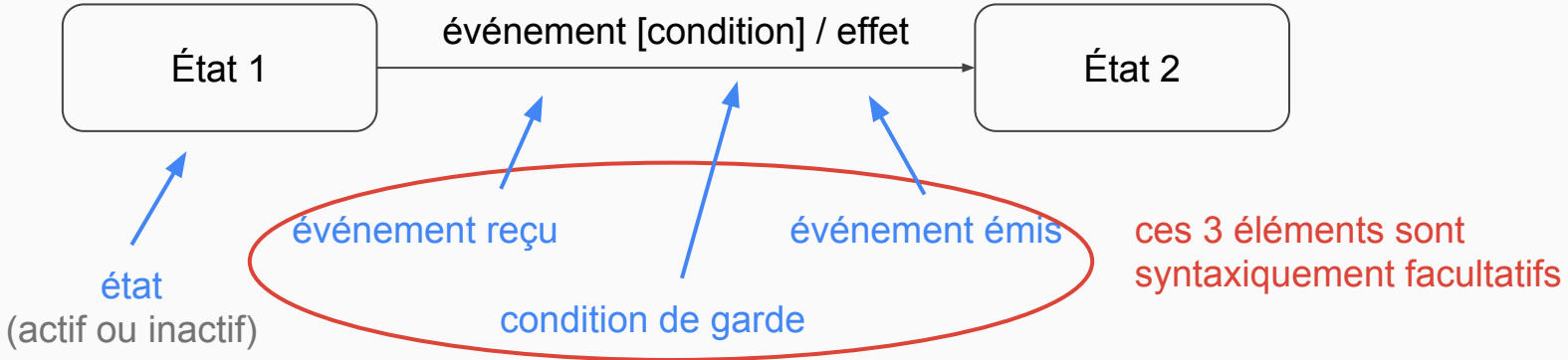


Diagramme d’état-transition

Diagramme d’état-transition

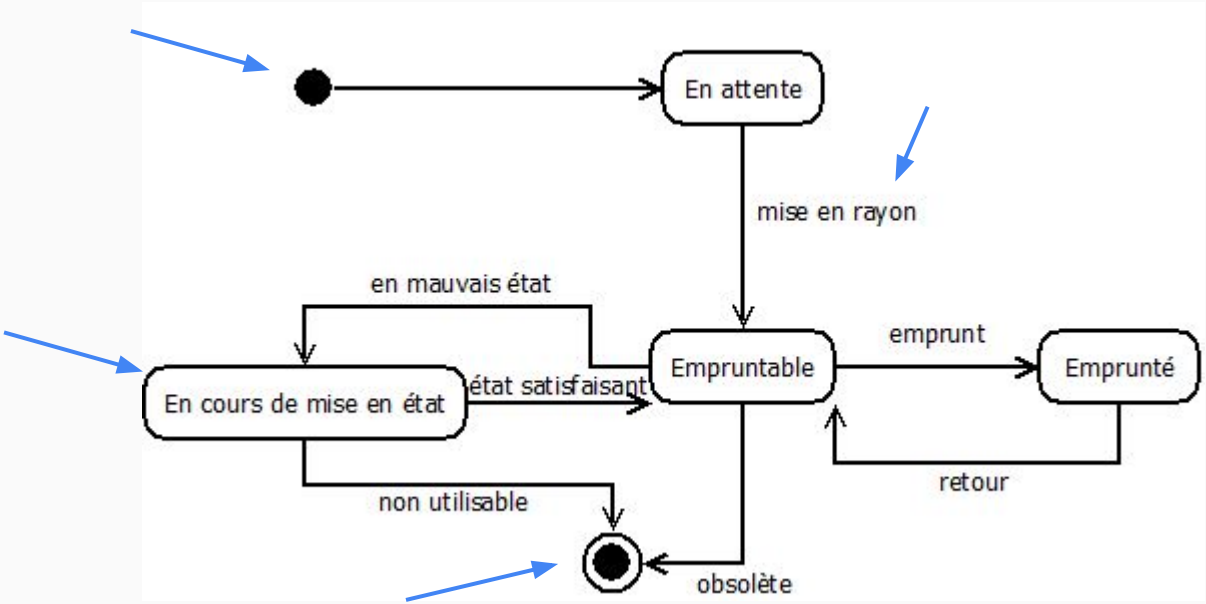
- Description des réactions d’un objet aux changements de son environnement. **État = moment du cycle de vie d’un objet.**
- En réponse à des stimuli reçus.



Événement reçu

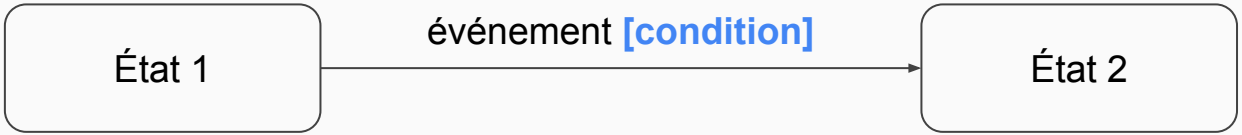
- Déclenche la transition qui lui est associée.
- Cette transition permet de passer d'un état à l'autre.
- Un objet peut réagir à certains événements lorsqu'il est dans certains états.

Exemple simple : cycle de vie d'un livre



Condition de garde

- Pour que la transition soit franchie, il faut que la condition soit remplie en plus de la réalisation de l'événement associé.



Condition de garde et effets : exemple

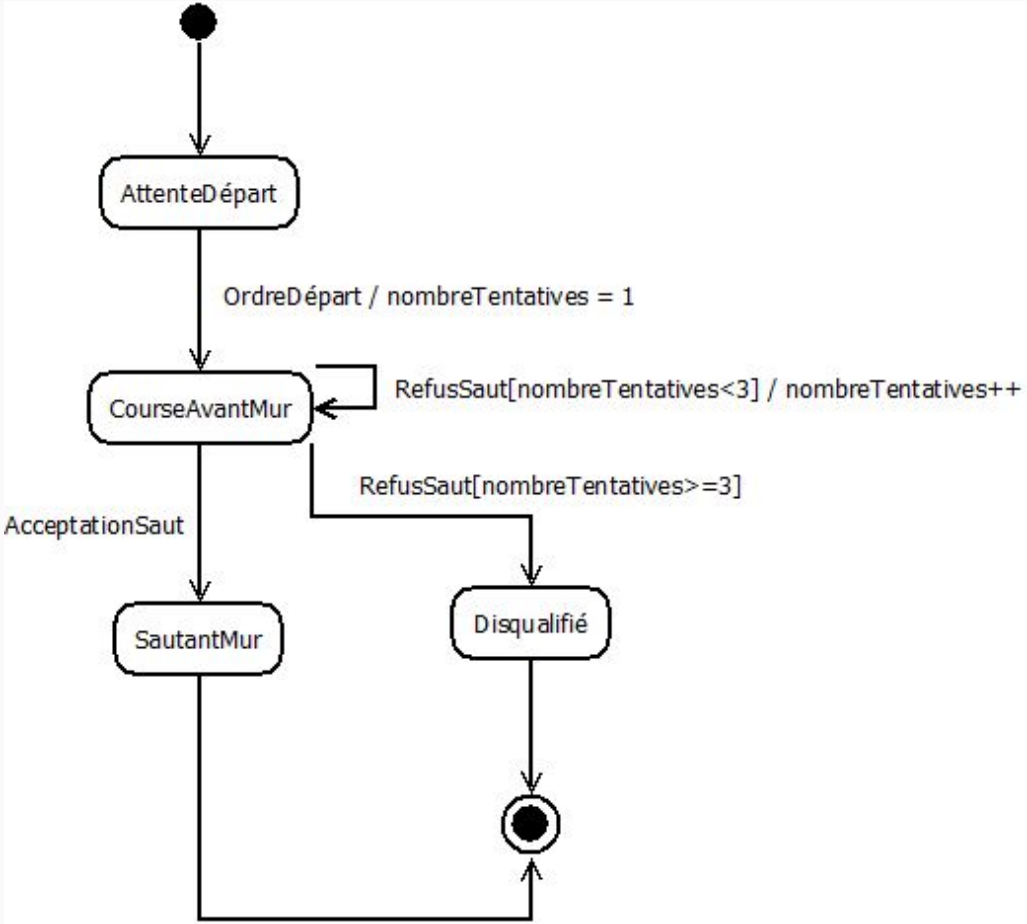


Diagramme de classes

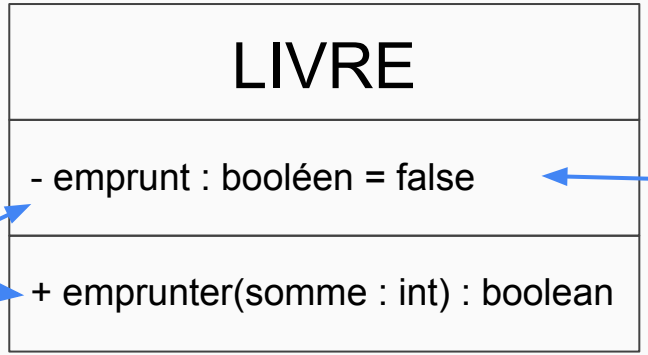
Diagramme de classes

On cherche à représenter :

- les classes (avec les attributs, les méthodes)
- les relations entre les classes
- les liens d’héritage

Ce diagramme est statique.

Une classe



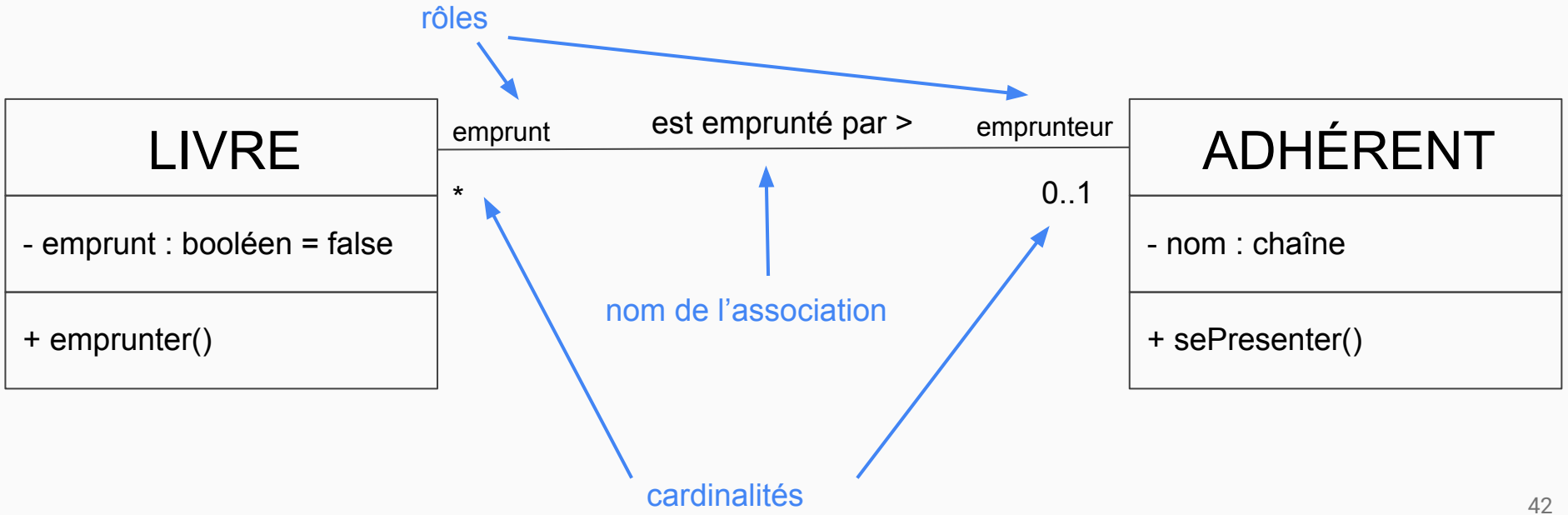
attributs : aucun à plusieurs, typés ou non, avec une valeur par défaut ou non

visibilités

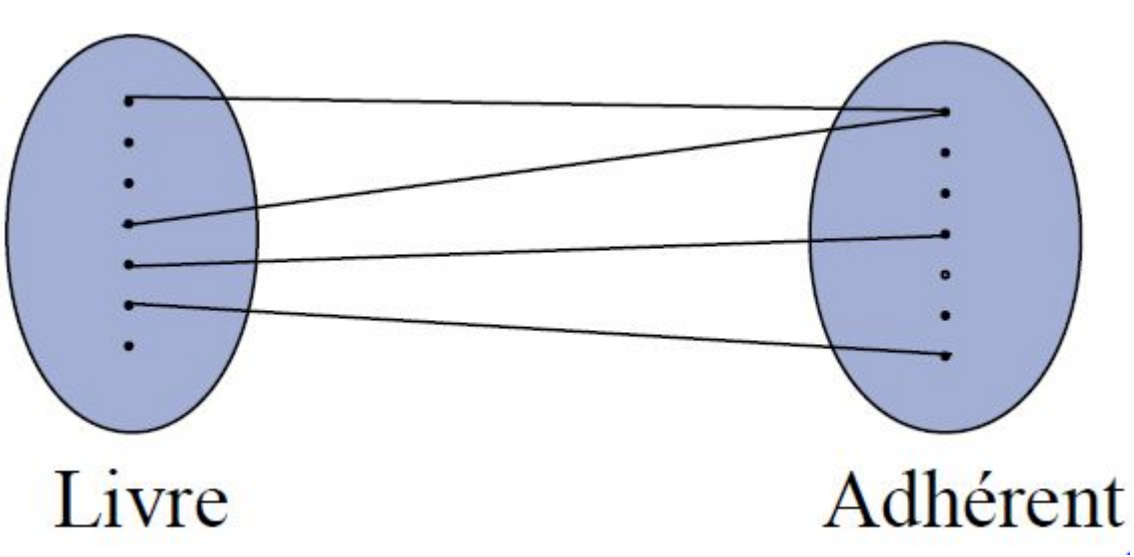
- public +
- protégé #
- privé -
- paquetage ~

NB : on utilise de plus le souligné pour indiquer un attribut ou une méthode `static`

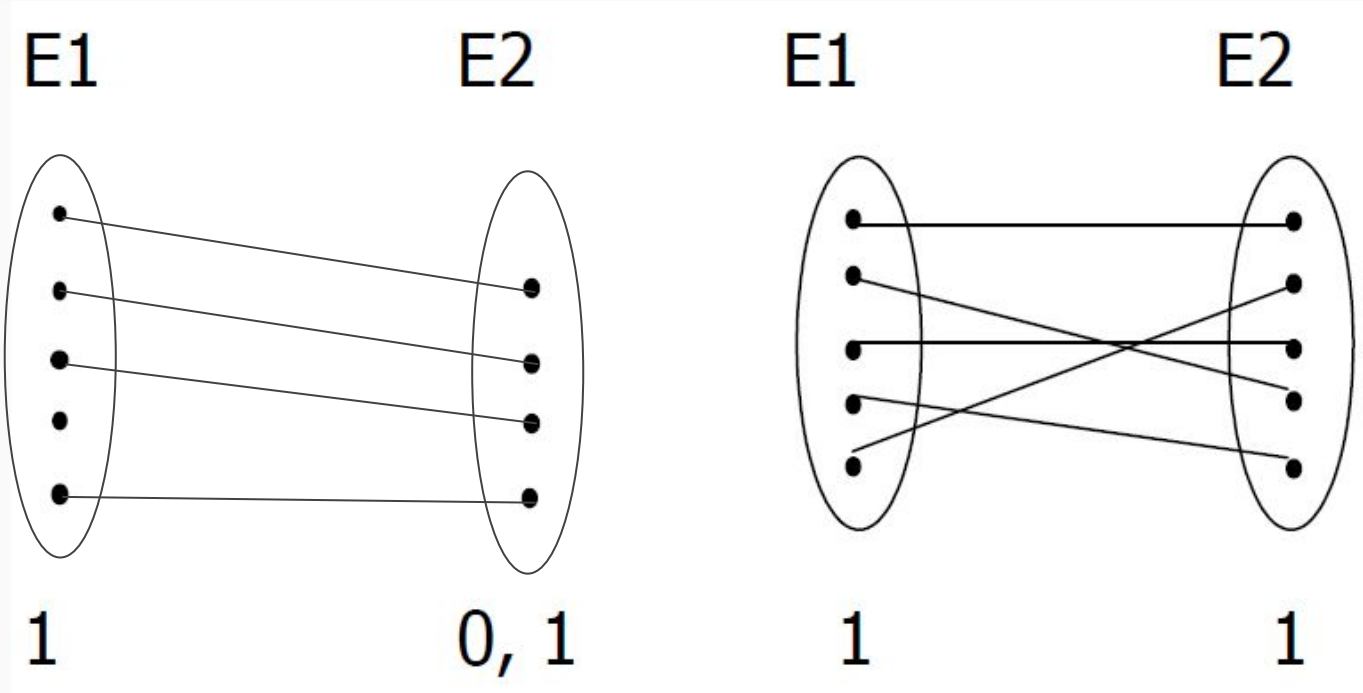
Associations entre classes



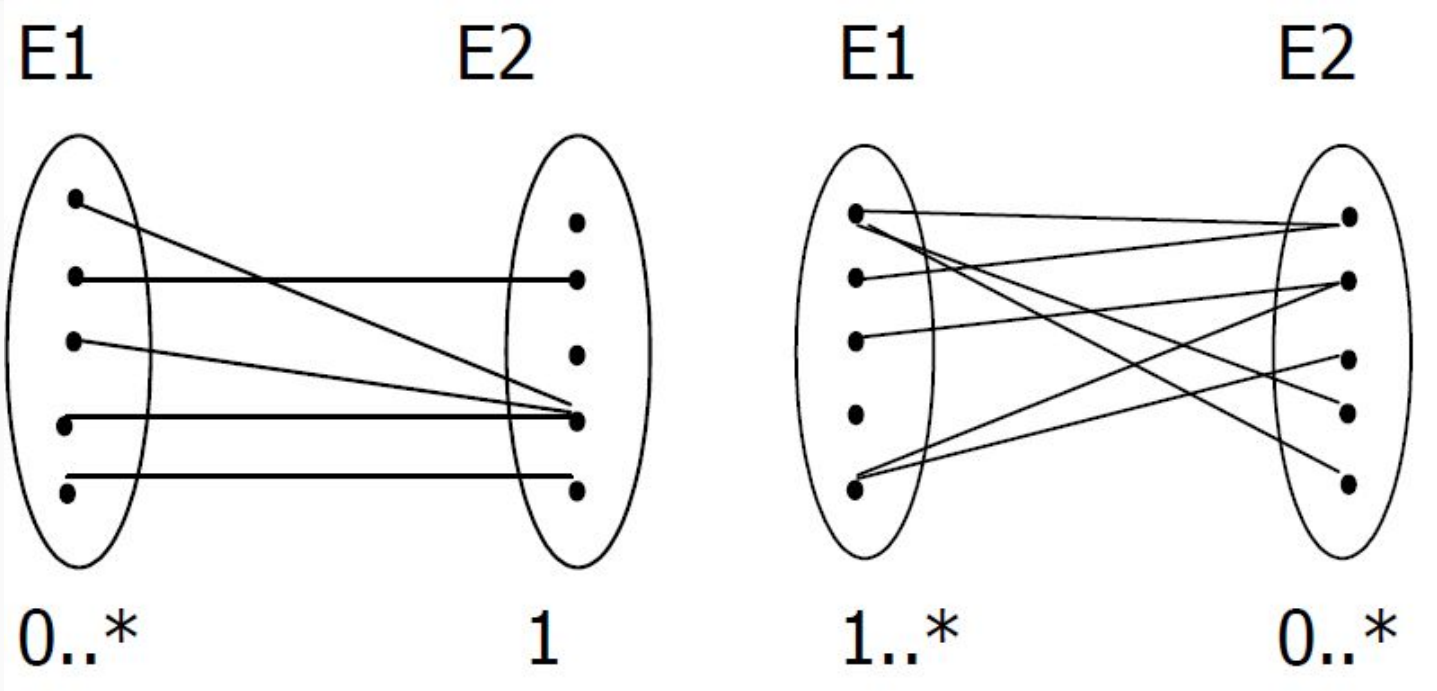
Associations entre classes



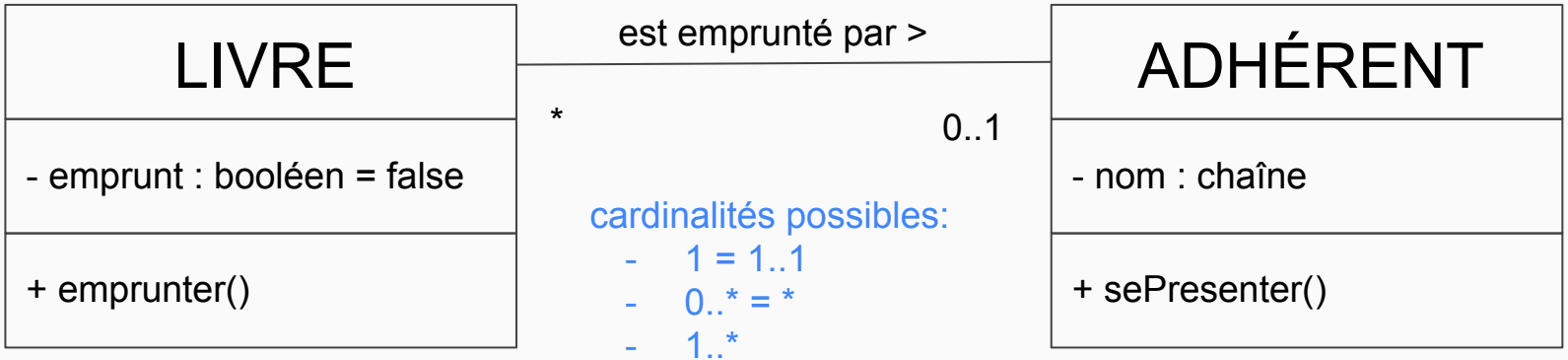
Associations entre classes



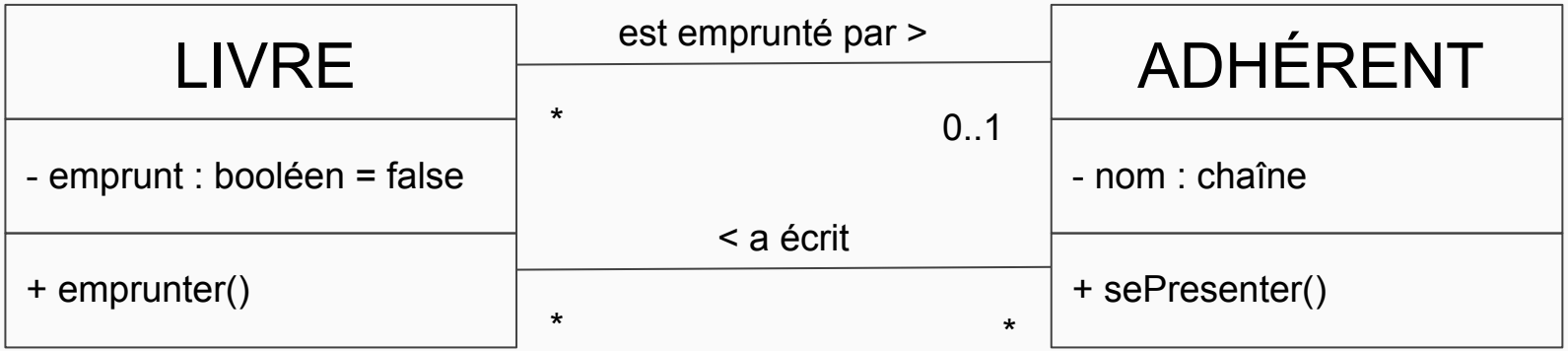
Associations entre classes



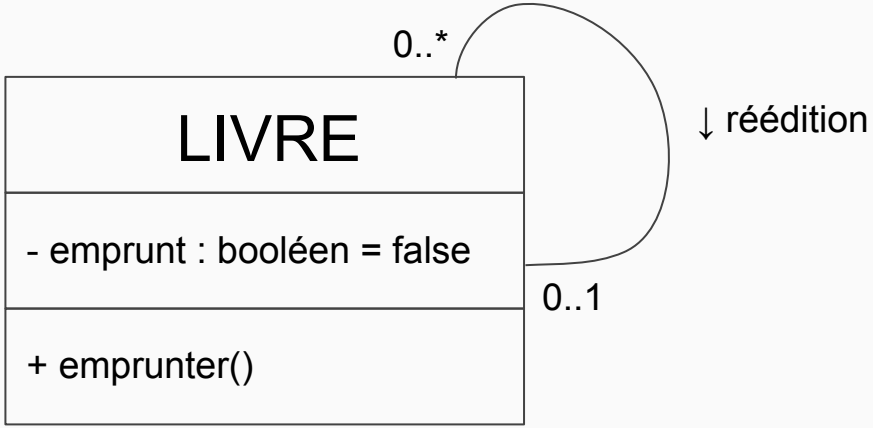
Cardinalités



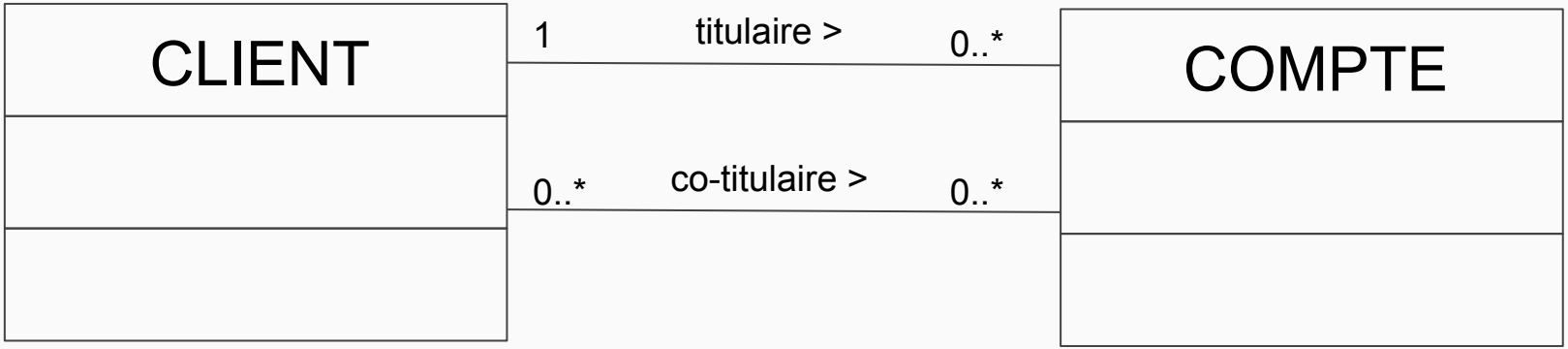
Associations multiples



Associations réflexives



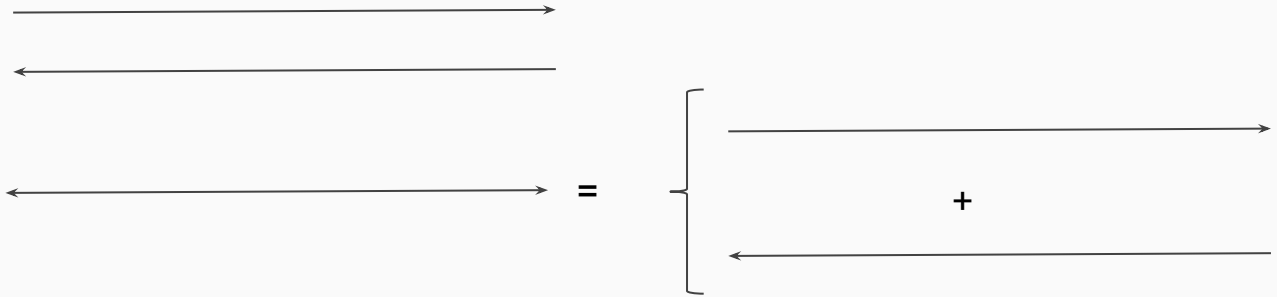
Les associations ne sont pas toujours suffisantes



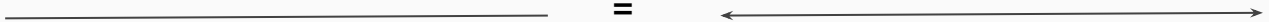
Exemple : Un client ne peut pas être à la fois titulaire et co-titulaire d'un même compte.

Associations navigables

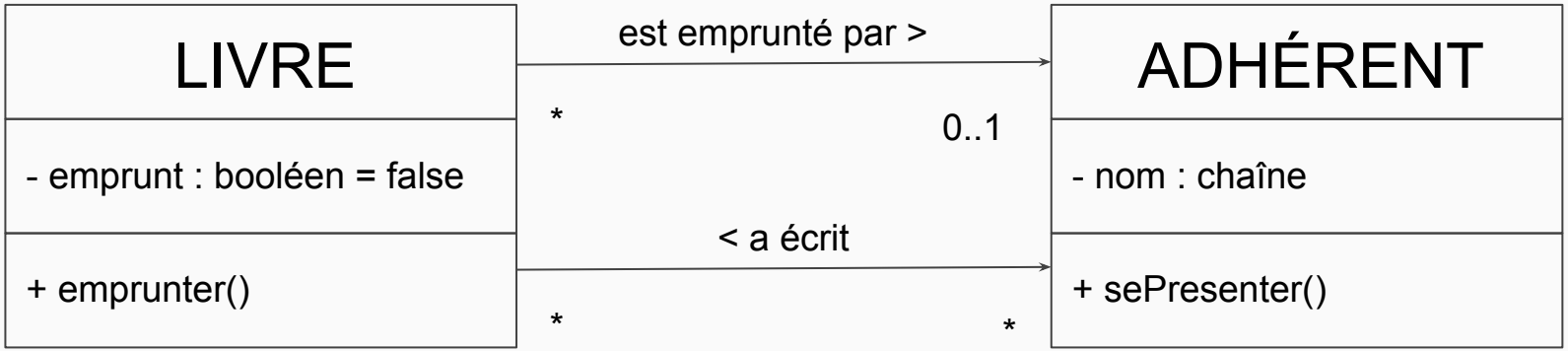
Associations navigables orientées :



Associations non orientées :

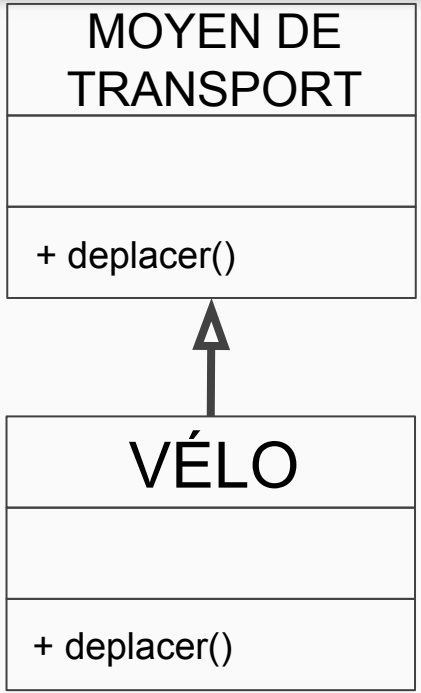


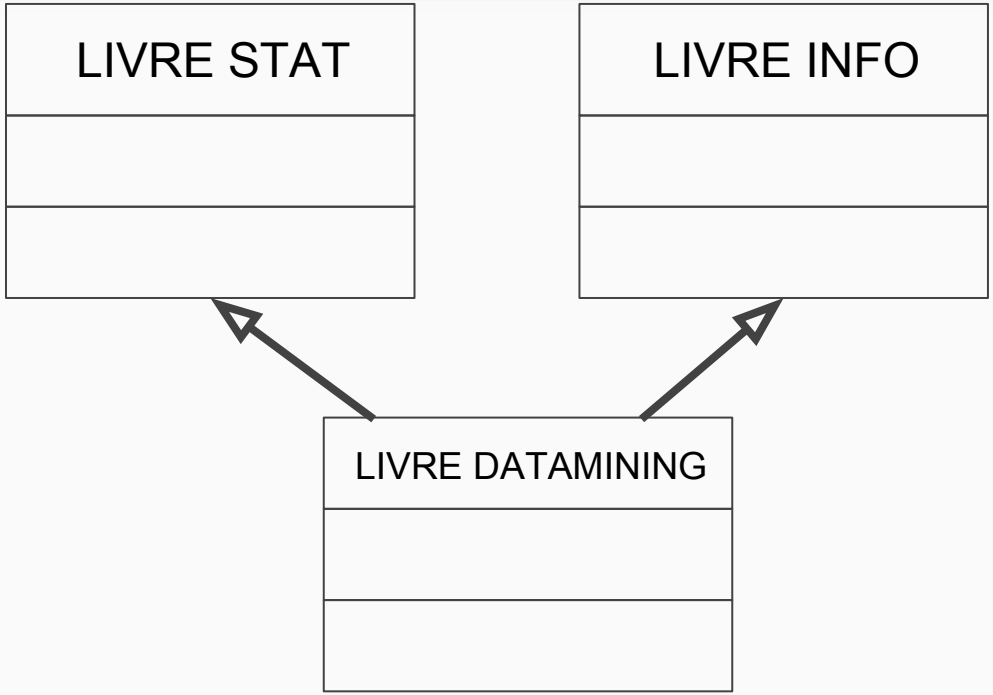
Associations navigables : exemple



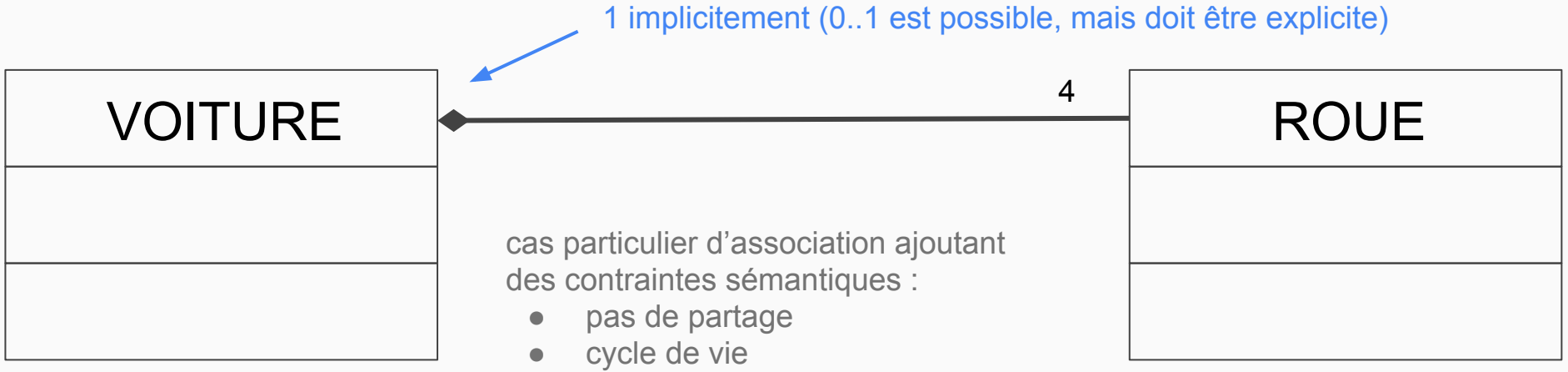
Q : quelle implémentation ?

Héritage simple et multiple





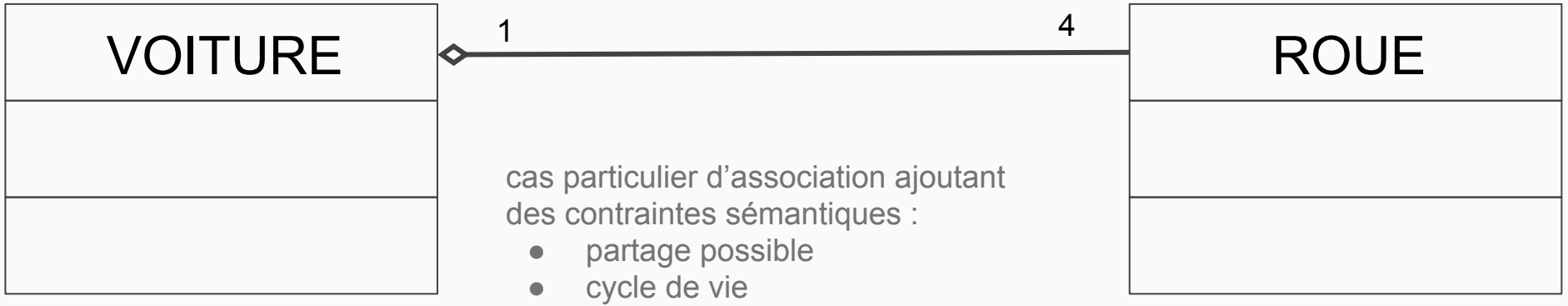
Composition : vente de voiture



cas particulier d'association ajoutant des contraintes sémantiques :

- pas de partage
- cycle de vie

Agrégation : casse automobile



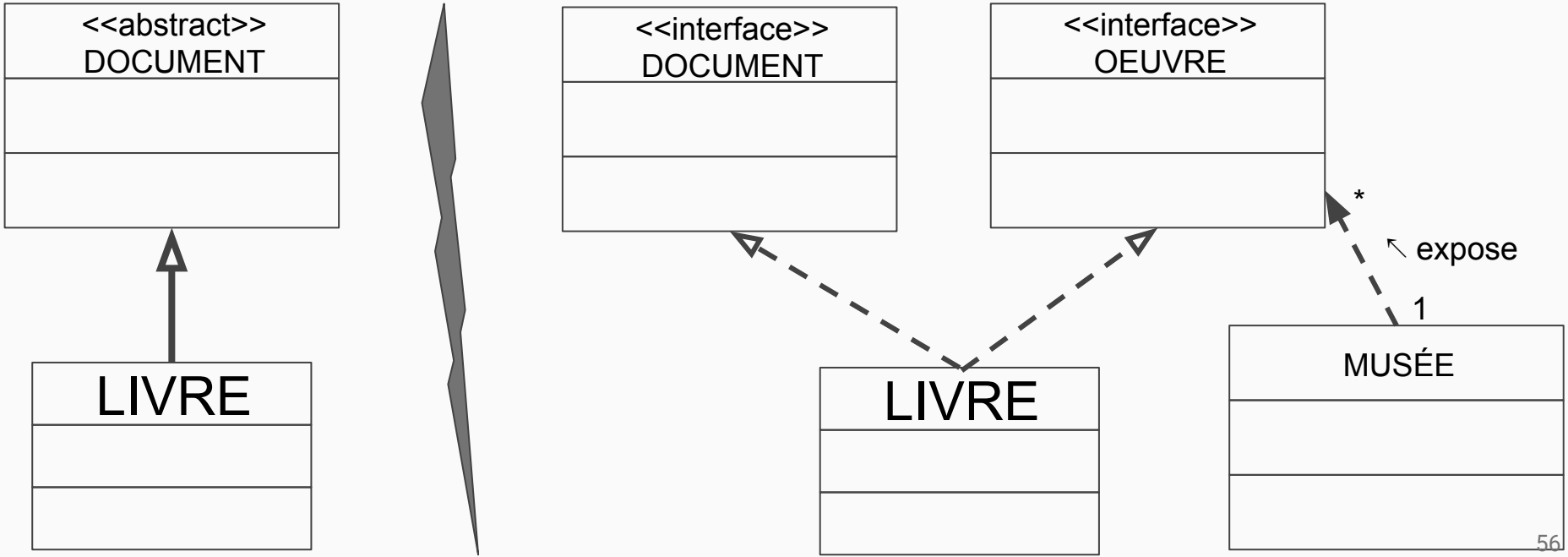
cas particulier d'association ajoutant des contraintes sémantiques :

- partage possible
- cycle de vie

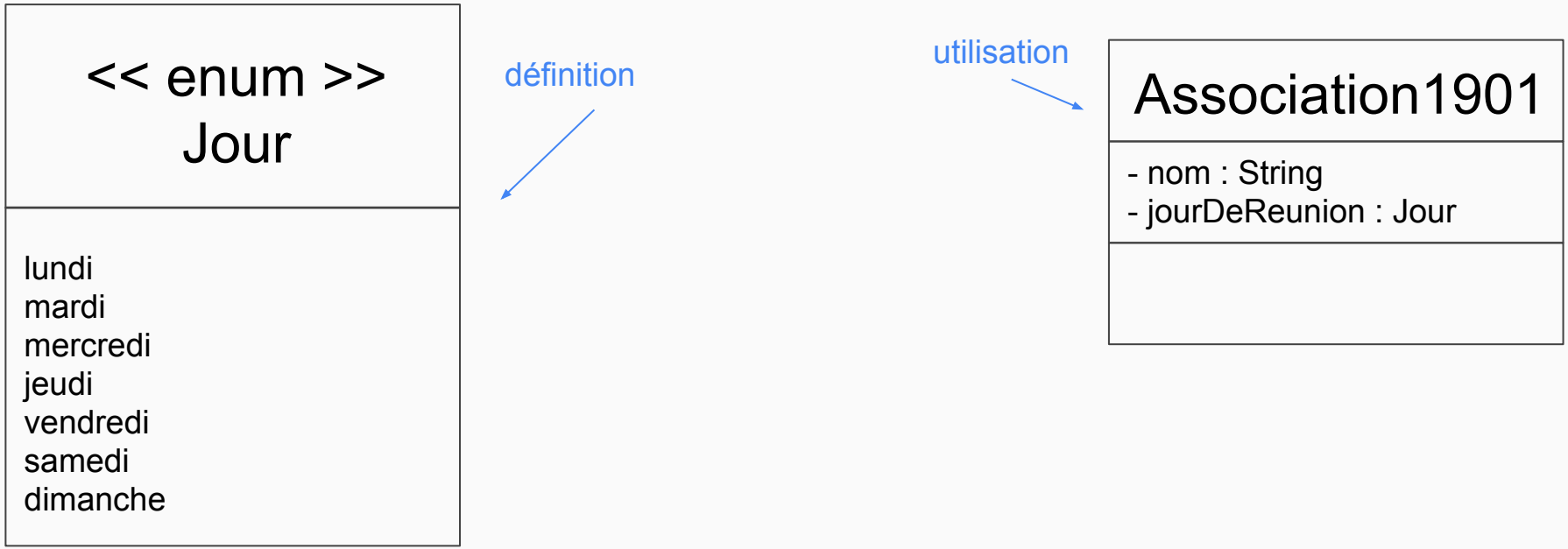
Composition *vs* agrégation

	Agrégation	**Composition**
Partage	Oui	Non
Destruction	Non	Oui
Cardinalité	Quelconque	1 ou 0..1

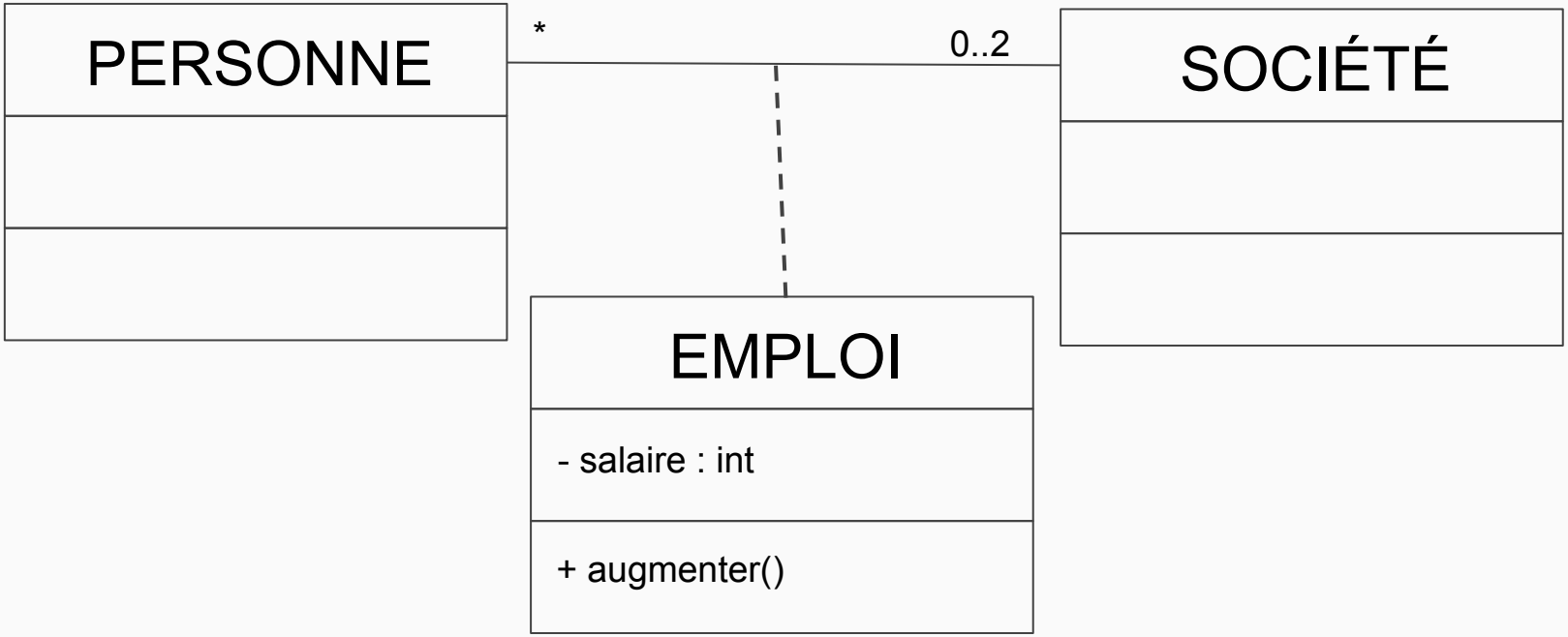
Classe abstraite, interface



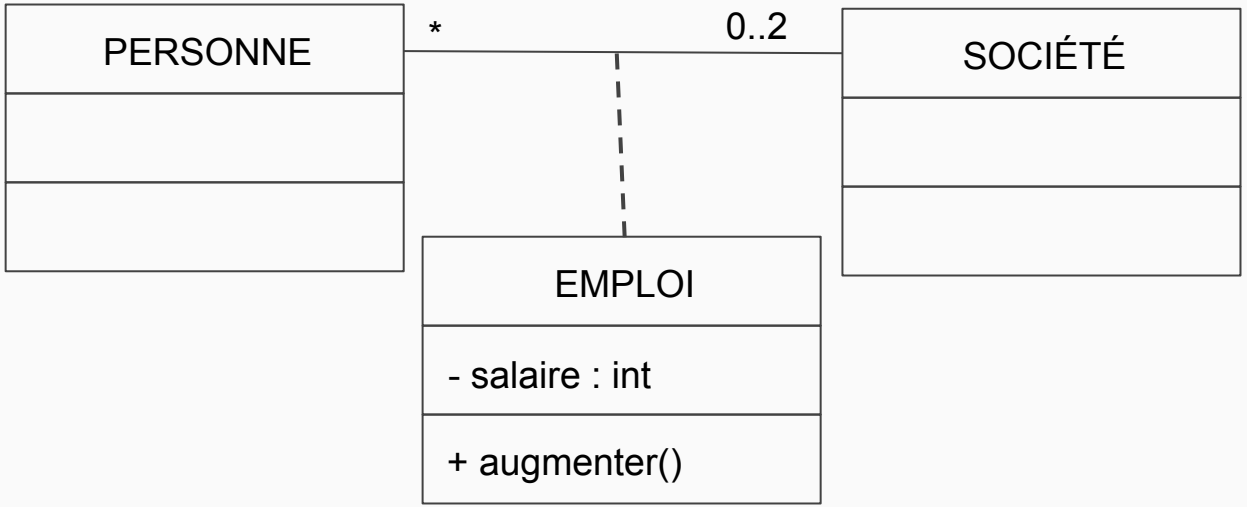
Enumération



Classes d’association



Classe d'association



Q : Quelle sémantique pour chacun de ces diagrammes ?

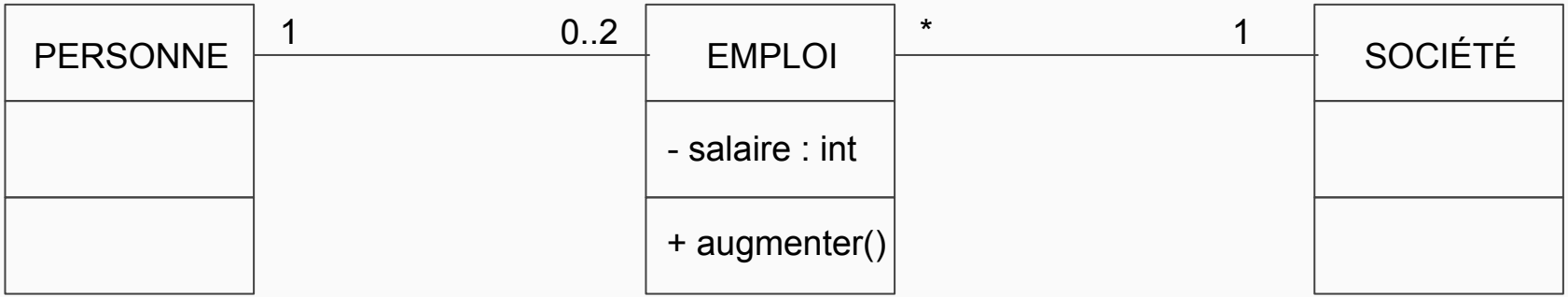


Diagramme d’objets

Diagramme d’objets

Montre, *à un instant donné*, les **instances** créées et leurs **liens**.

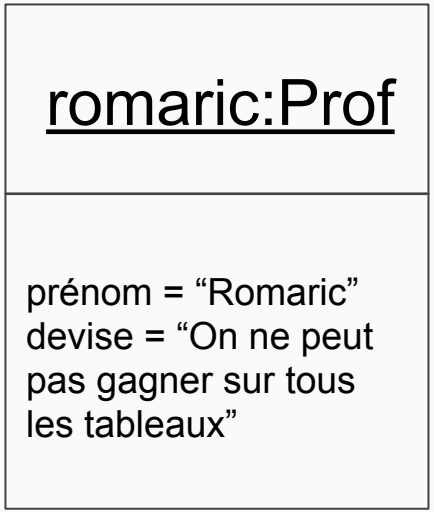


Diagramme d'objets : exemple

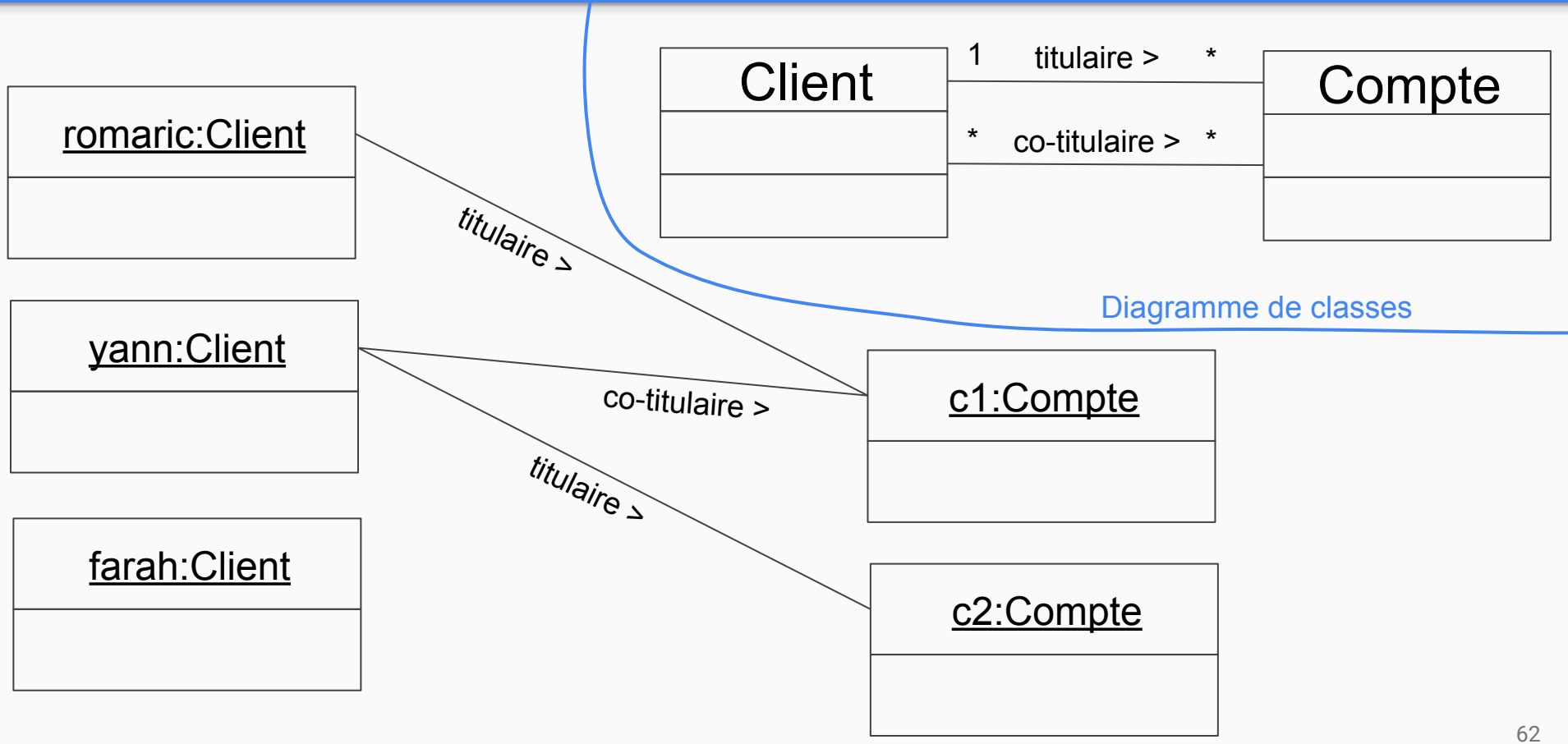


Diagramme de paquetages

Diagramme de paquetages

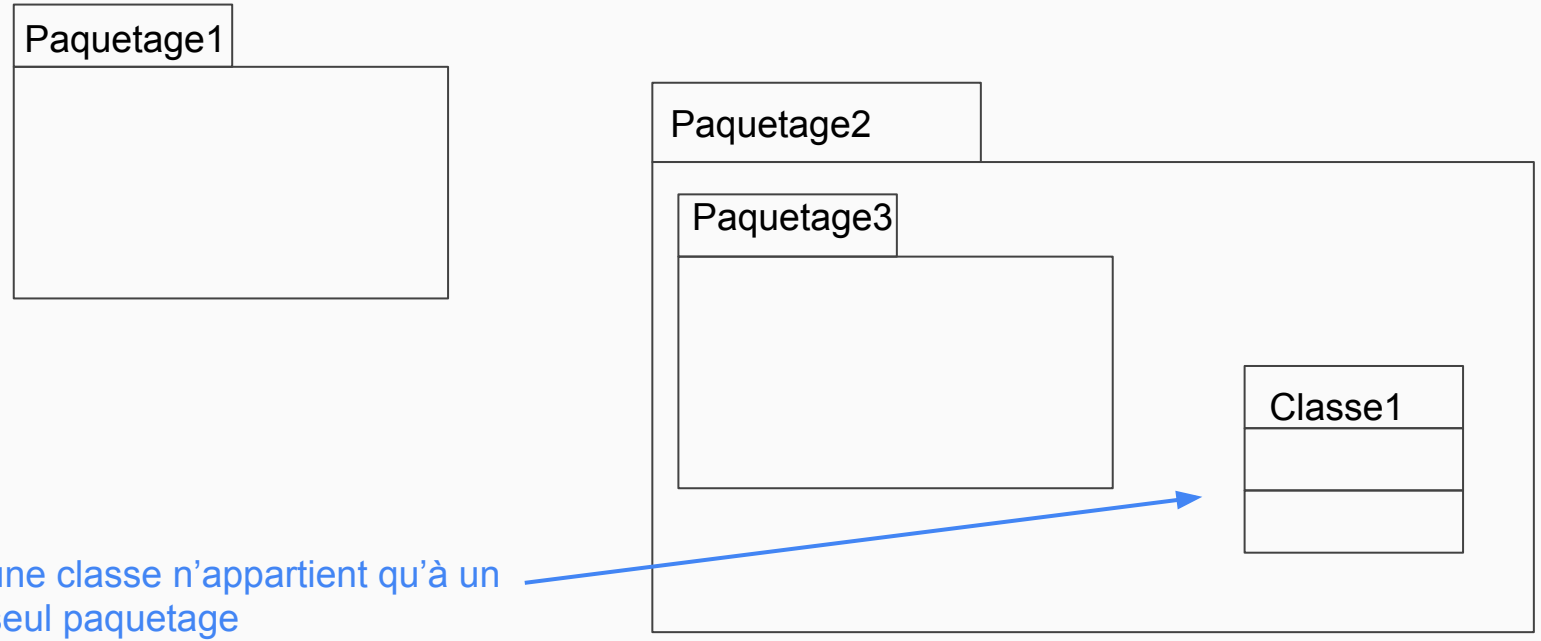
Permet de décomposer logiquement l'application :

- selon une logique métier
- en tâchant de minimiser les dépendances entre paquetages

Chaque paquetage peut contenir :

- des classes
- des paquetages

Diagramme de paquetages : exemple



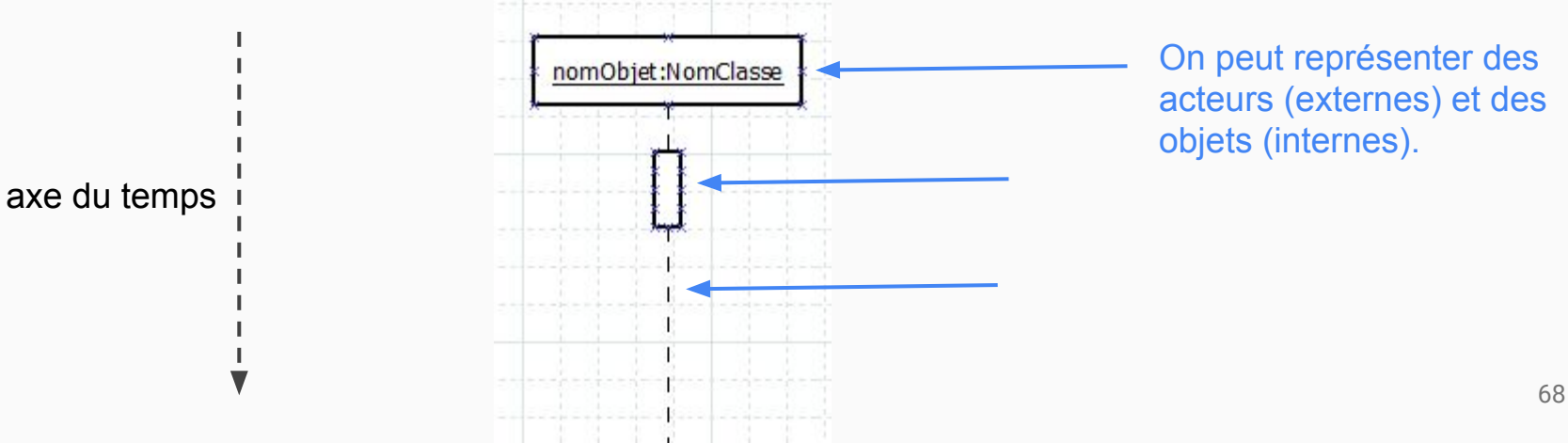
Lien d’utilisation entre paquetages

NB : Lorsque c’est possible, éviter les dépendances circulaires !

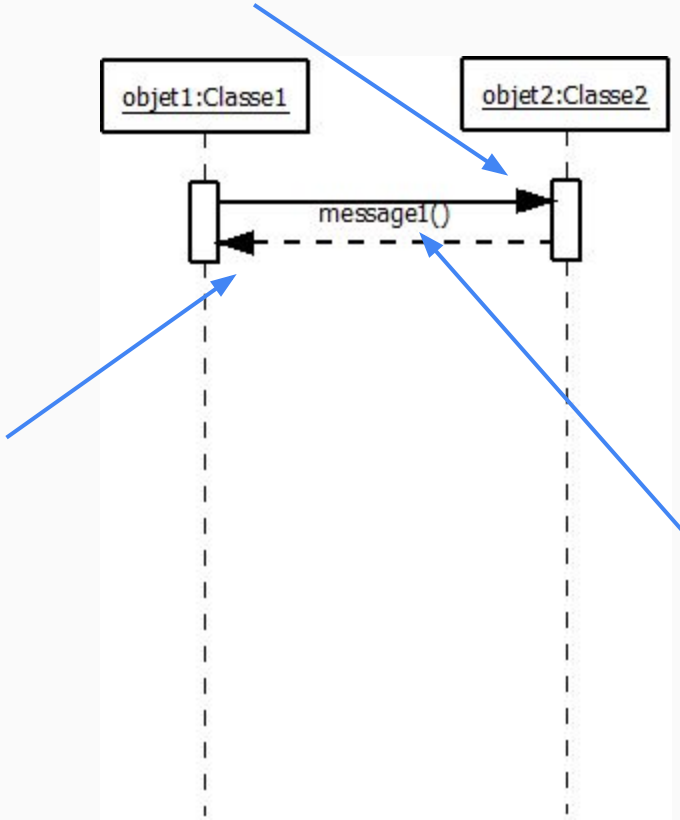
Diagramme de séquence

Diagramme de séquence

- Représentation d'**un** scénario (parmi tous les possibles, cf diagramme d'activité).
- Suivant un axe **temporel** (de haut en bas).

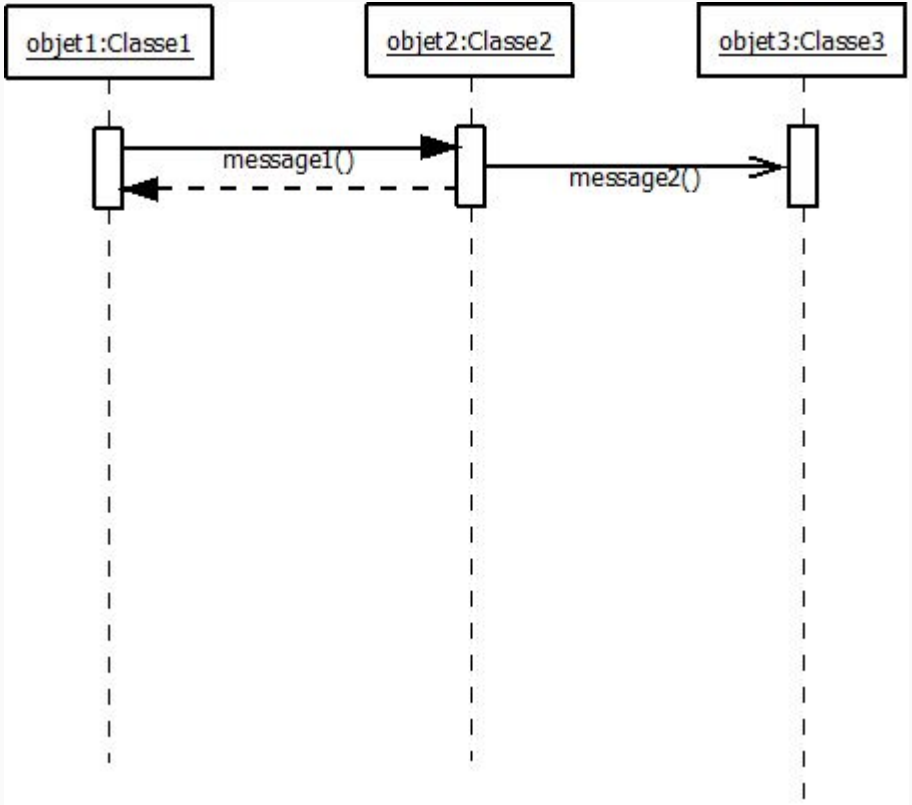


Échange de messages

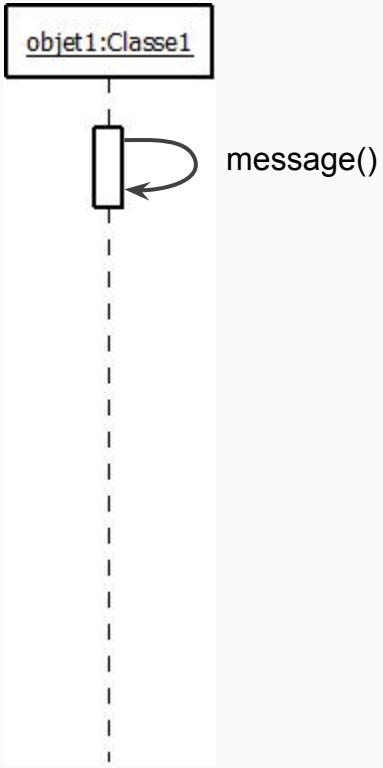


On peut ajouter des paramètres.
Envoi de message = appel de méthode.

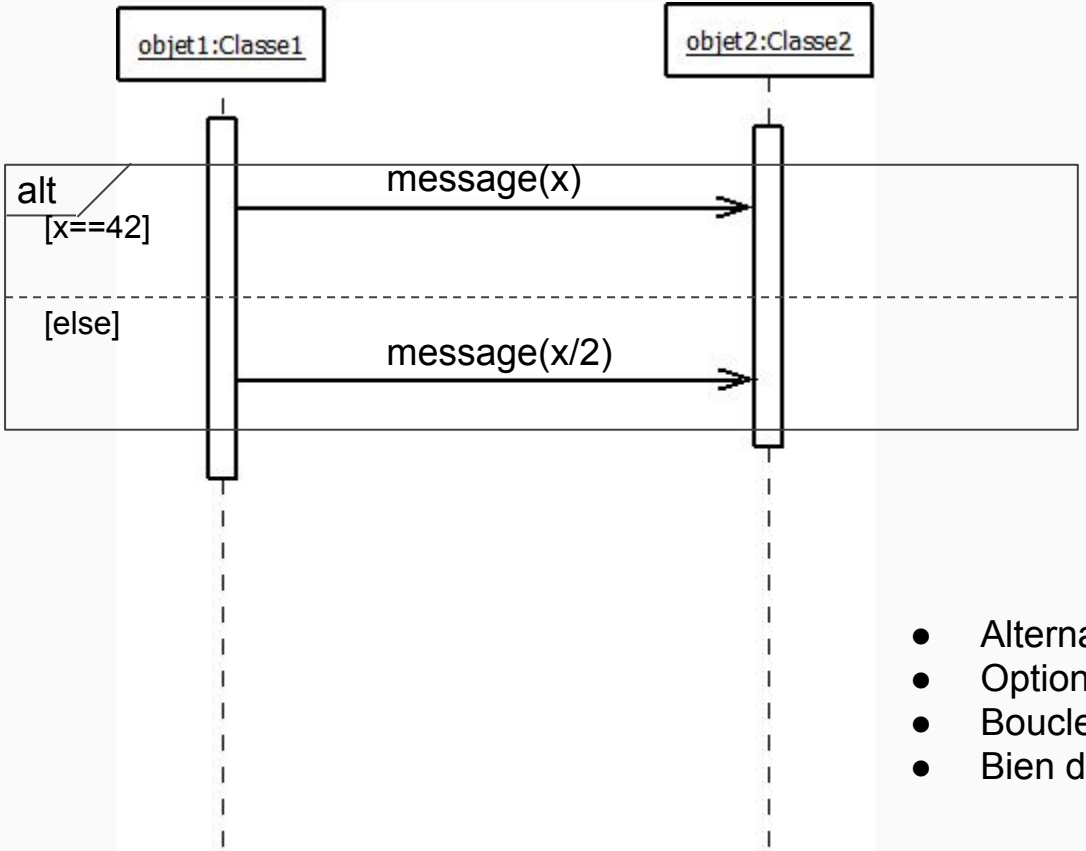
Messages synchrones et asynchrones



Messages réflexifs

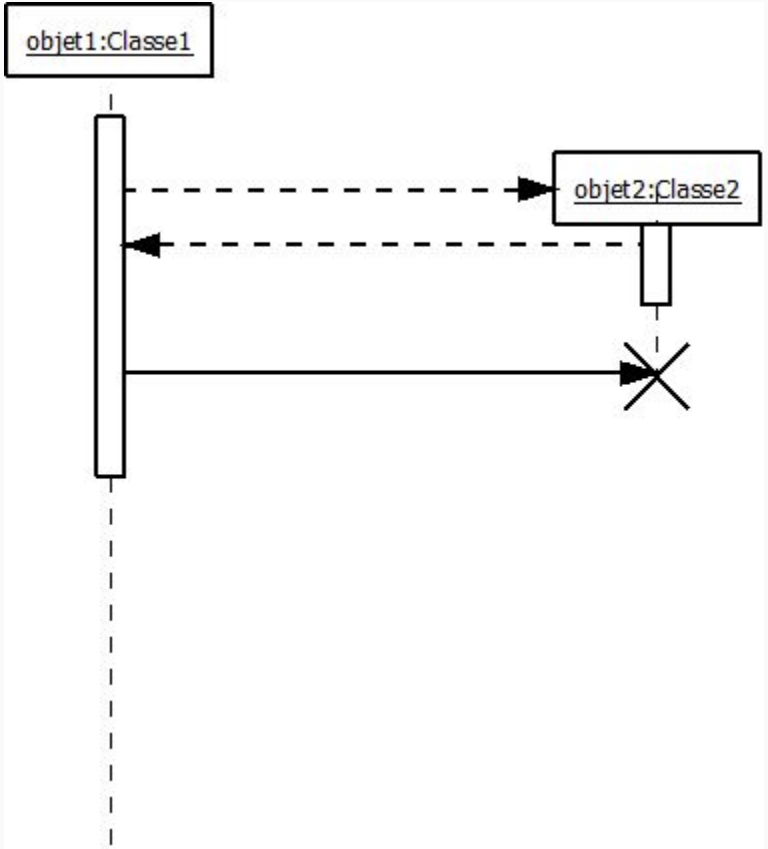


Fragments combinés (depuis UML 2)



- Alternative (**alt**)
- Option (**opt**)
- Boucle (**loop**)
- Bien d'autres...

Création et destruction d’objets



UNITÉ
BIBLIOTHÈQUE

BUREAU
132A

TÉLÉPHONE
3236

Jean-Luc DUVAL
Responsable de l'Unité

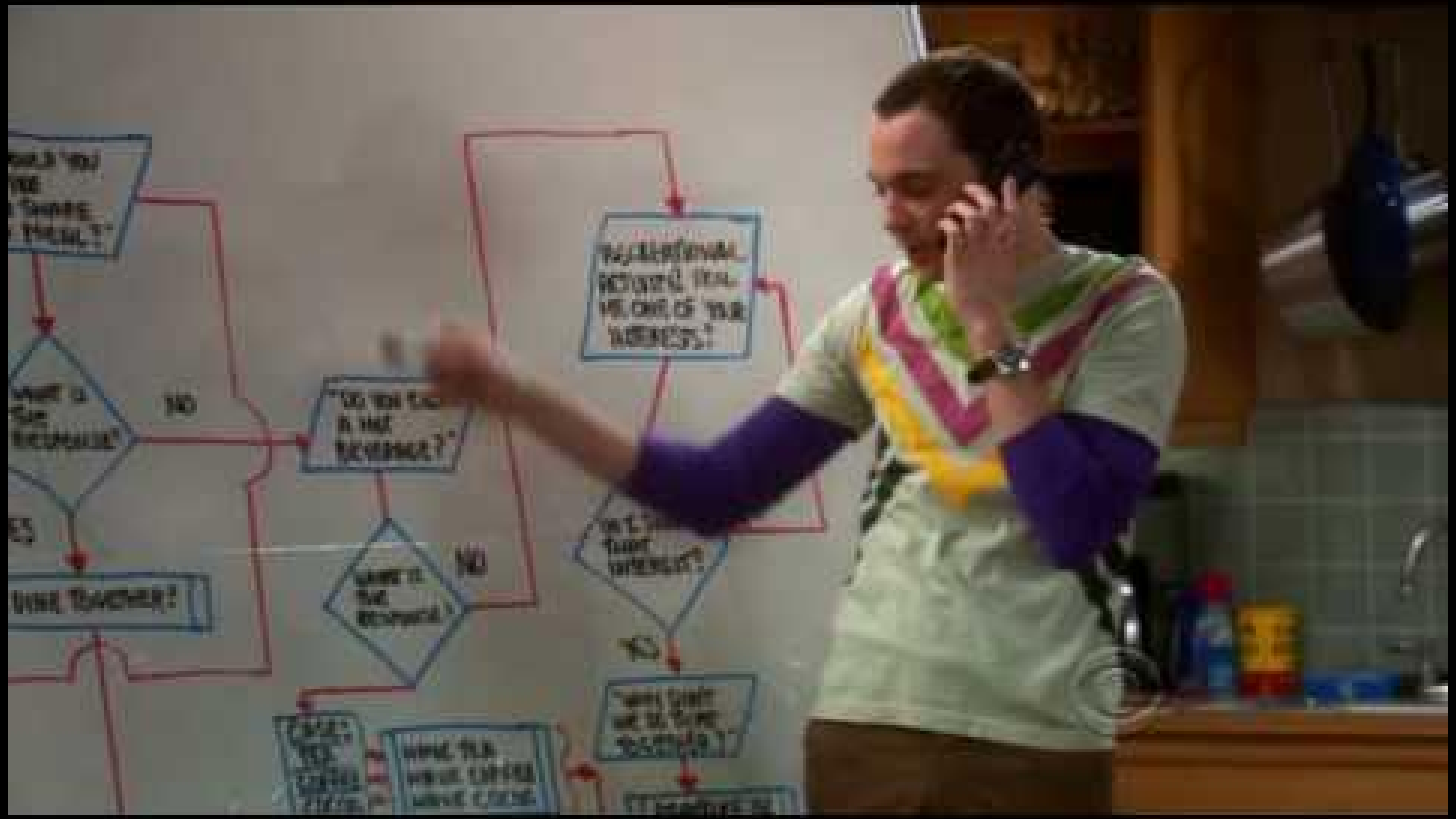
BUREAU
132B

TÉLÉPHONE
3237

Caroline DÉZON
Bibliothécaire

↑ Vos meilleurs alliés ↑

Demandez le « dépliant orange » !


NO
NO
NO

Bibliographie

- Cours d'UML de Laurence Duval (Université de Bretagne Occidentale)
- *UML Basics*, support de cours d'Olivier Barais et Benoît Combemale (Université de Rennes I)
- *UML 2.5* (4è édition) de Laurent Debrauwer et Fien Van Der Heyde aux éditions Eni
- *UML 2.0* de Pascal Roques aux éditions Eyrolles