

# CONCEPTION ORIENTÉE OBJET - COO

## DIAGRAMME UML DE PAQUETAGES

🎓 2A - Bachelor Universitaire de Technologie

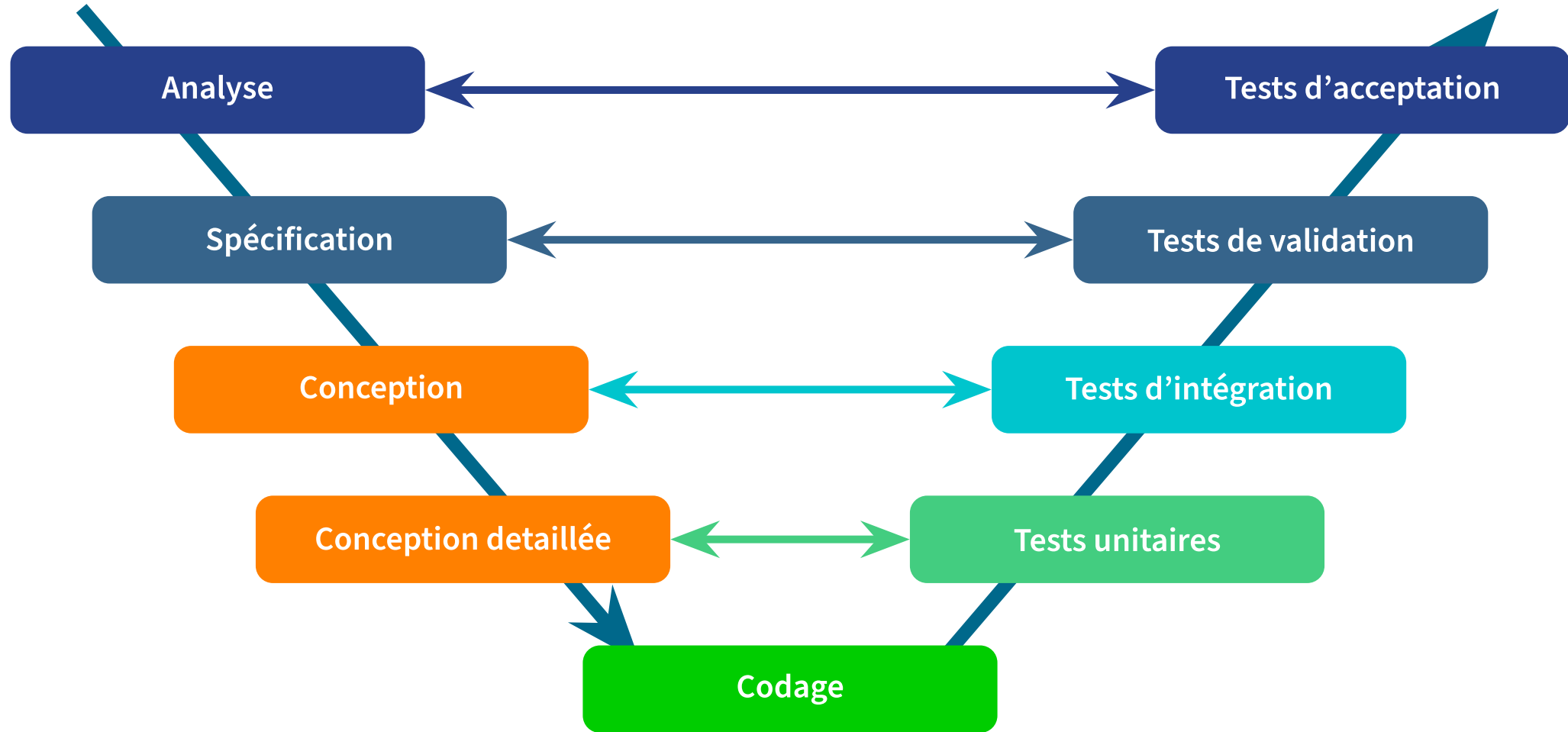
🏛️ IUT d'Orsay - Université Paris-Saclay - 2025/2026



**Idir AIT SADOUNE**

[idir.ait-sadoune@universite-paris-saclay.fr](mailto:idir.ait-sadoune@universite-paris-saclay.fr)

# CYCLE DE DÉVELOPPEMENT



# PLAN

- Les éléments de base
- Les relations entre paquetages
- Les principes de cohérence et d'indépendance
- Réduction de couplage entre paquetages

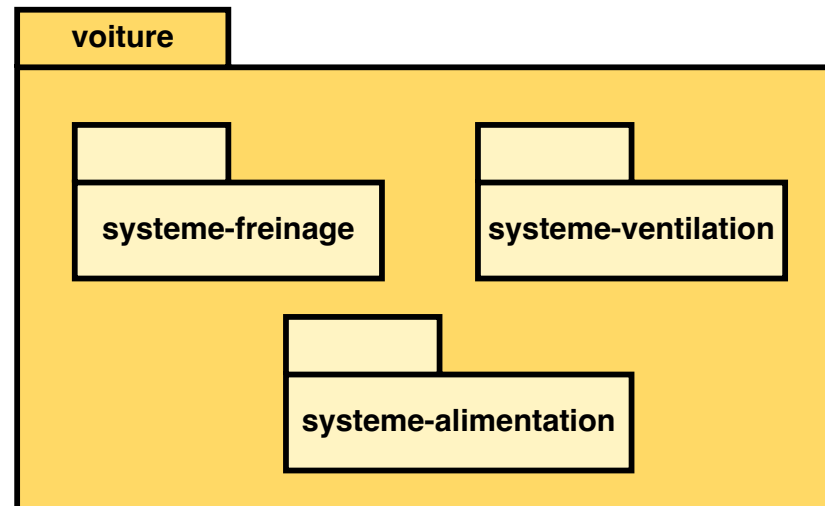
[Retour au plan](#) - [Retour à l'accueil](#)

# PLAN

- > Les éléments de base
- > Les relations entre paquetages
- > Les principes de cohérence et d'indépendance
- > Réduction de couplage entre paquetages

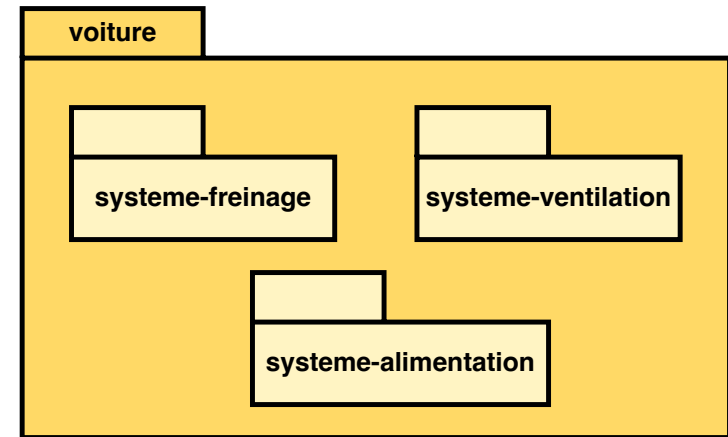
[Retour au plan](#) - [Retour à l'accueil](#)

# EXEMPLE D'INTRODUCTION



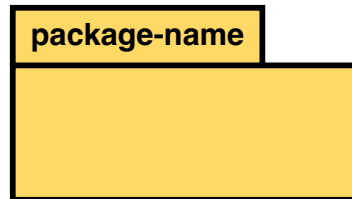
# DÉFINITIONS

- Un **paquetage** regroupe des éléments de la modélisation appelés **membres**, portant sur **un sous-ensemble du système**.
- Le découpage en **paquetage** doit traduire **un découpage logique** du système correspondant à **des espaces de nommage homogènes**.
- Un **paquetage** permet de grouper **n'importe quelle élément** de modélisation d'**UML** dans des éléments de **plus haut niveau**.
  - cas d'utilisation, objets, classes, composants, ..., et d'autres paquetages.

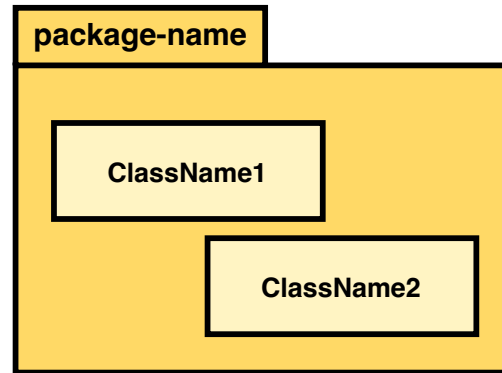


# PRÉSENTATION D'UN PAQUETAGE

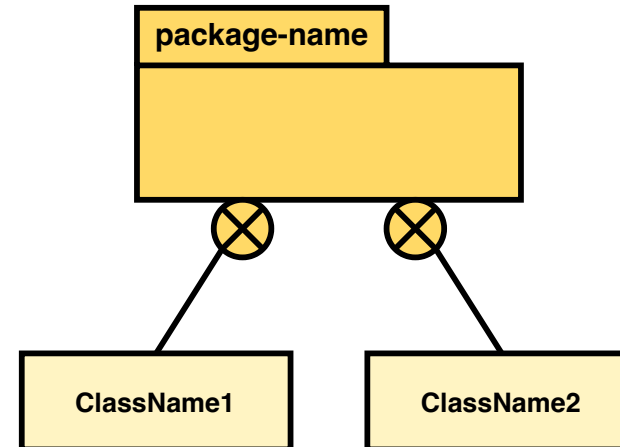
Représentation  
globale



Représentation  
détaillée



Représentation  
éclatée



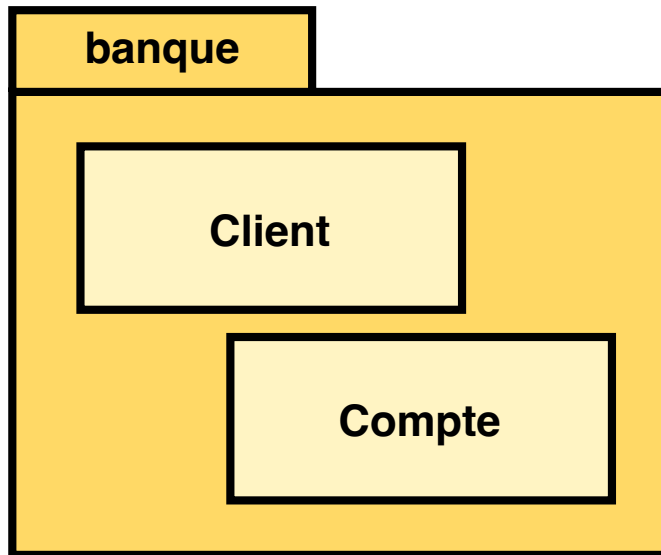
# NOM D'UN PAQUETAGE

- Chaque **paquetage** doit avoir un **nom différent**.
- Un **paquetage** définit un **espace de nommage** (name space).
- Les **membres d'un paquetage** appartiennent au **paquetage englobant**:
  - **deux éléments** dans **deux paquetages** peuvent porter **le même nom**.
  - **deux éléments** dans **le même paquetage** doivent porter **des noms différents**.

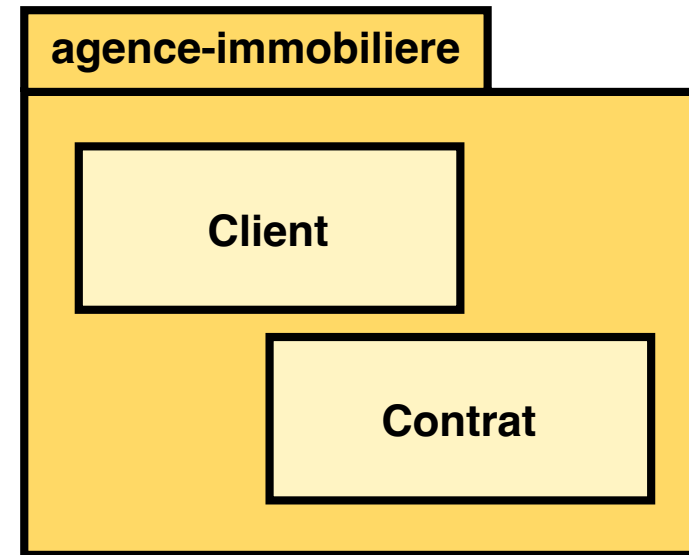


# NOM D'UN PAQUETAGE

## EXEMPLE

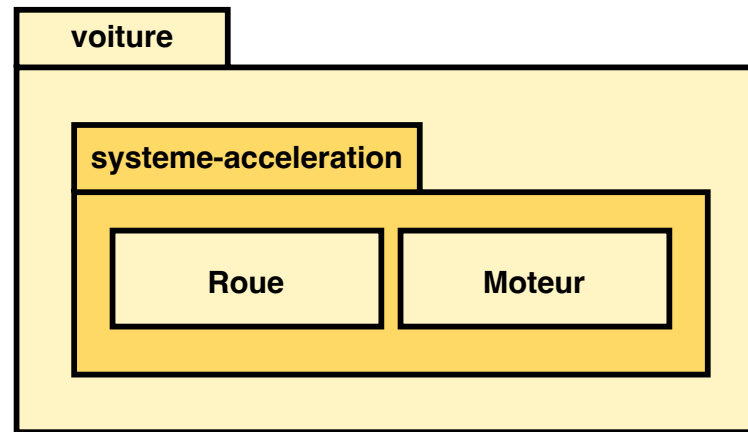


**banque::Client**



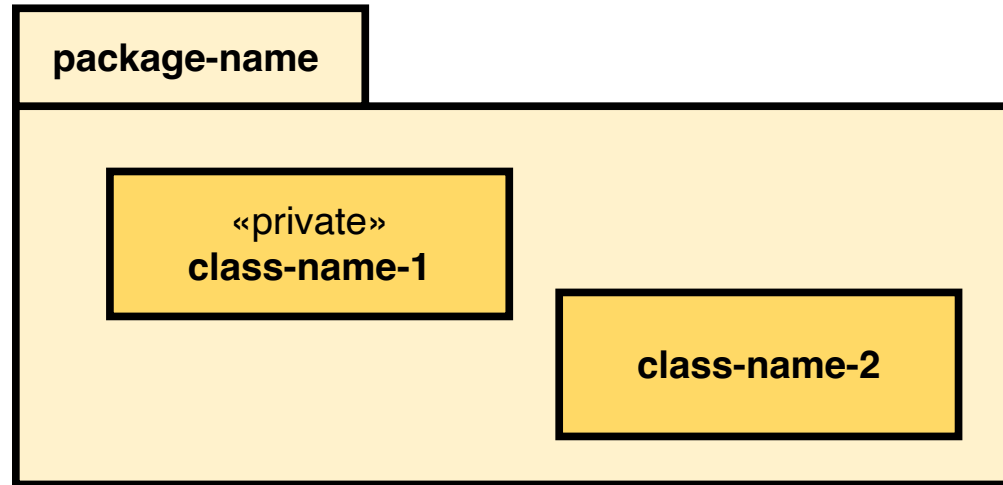
**agence-immobiliere::Client**

# NOM D'UN ÉLÉMENT



- Le **nom** d'un élément est **simple** s'il est utilisé **seul**.  
➡ la classe **Roue**
- Le **nom** d'un élément est **complet** s'il est précédé par **les noms des paquetages englobants (name space)**.  
➡ la classe **voiture::systeme-acceleration::Roue**

# LA VISIBILITÉ



- Les éléments d'un **paquetage** peuvent avoir une **visibilité** déclarée:
  - de type **public** (par défaut) → visible dans tout le modèle.
  - de type **privé** (**private**) → non visible à l'extérieur du paquetage.

# PLAN

- Les éléments de base
- Les relations entre paquetages
- Les principes de cohérence et d'indépendance
- Réduction de couplage entre paquetages

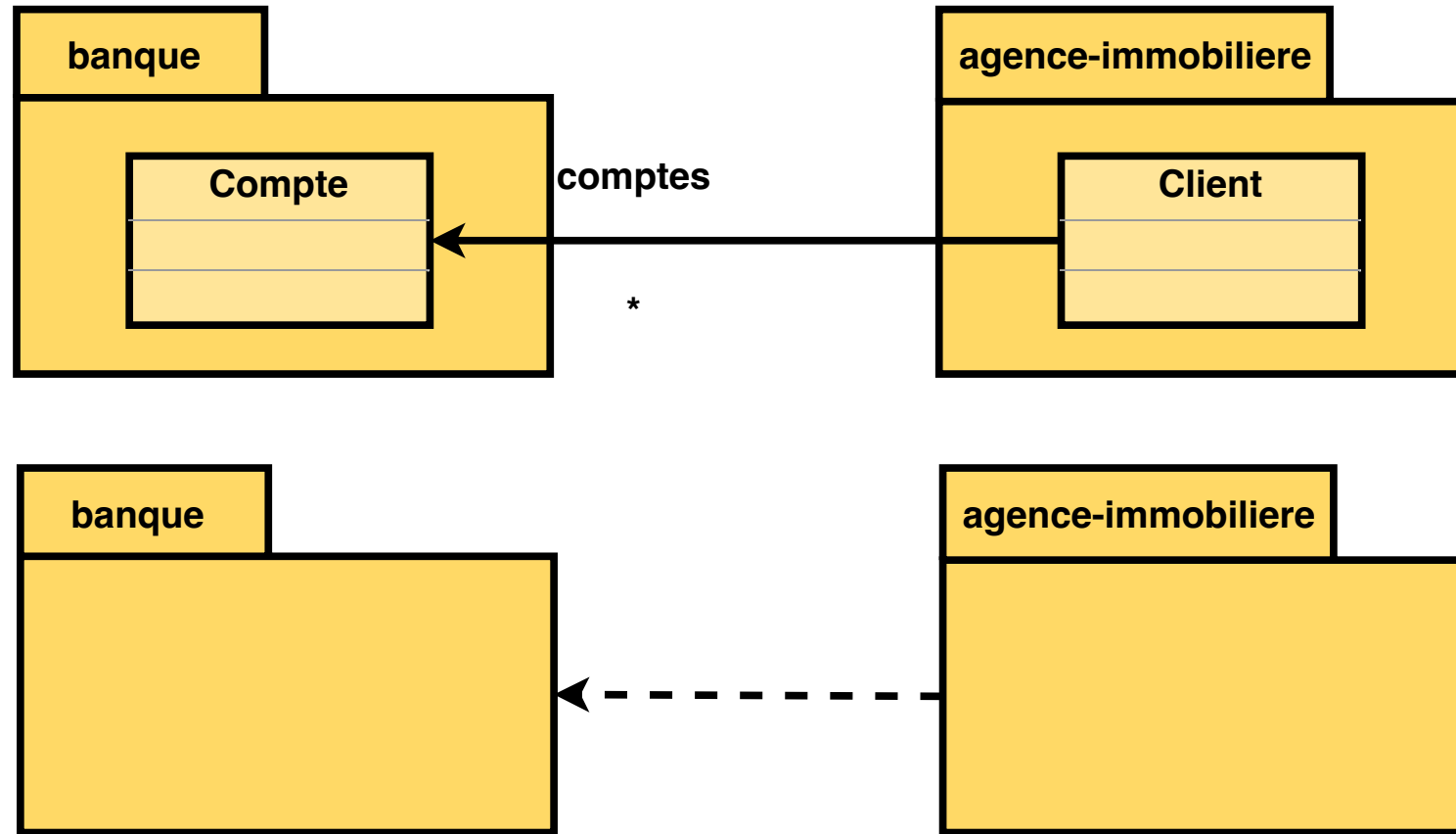
[Retour au plan](#) - [Retour à l'accueil](#)

# RELATION DE DÉPENDANCES

- **Une relation de dépendance** entre deux paquetages existe dès que **deux éléments de modélisation** issus de deux paquetages sont **associés**.
  - ▢► hormis les cas de **dépendances implicites** (emboîtement de paquetages)
- **La dépendance** est une **relation unidirectionnelle** entre **paquetages**.
  - ▢► une modification de la cible peut impliquer une modification de la source
- Une relation de **dépendance** se représente par **une flèche en pointillé**.

# RELATION DE DÉPENDANCES

## EXEMPLE

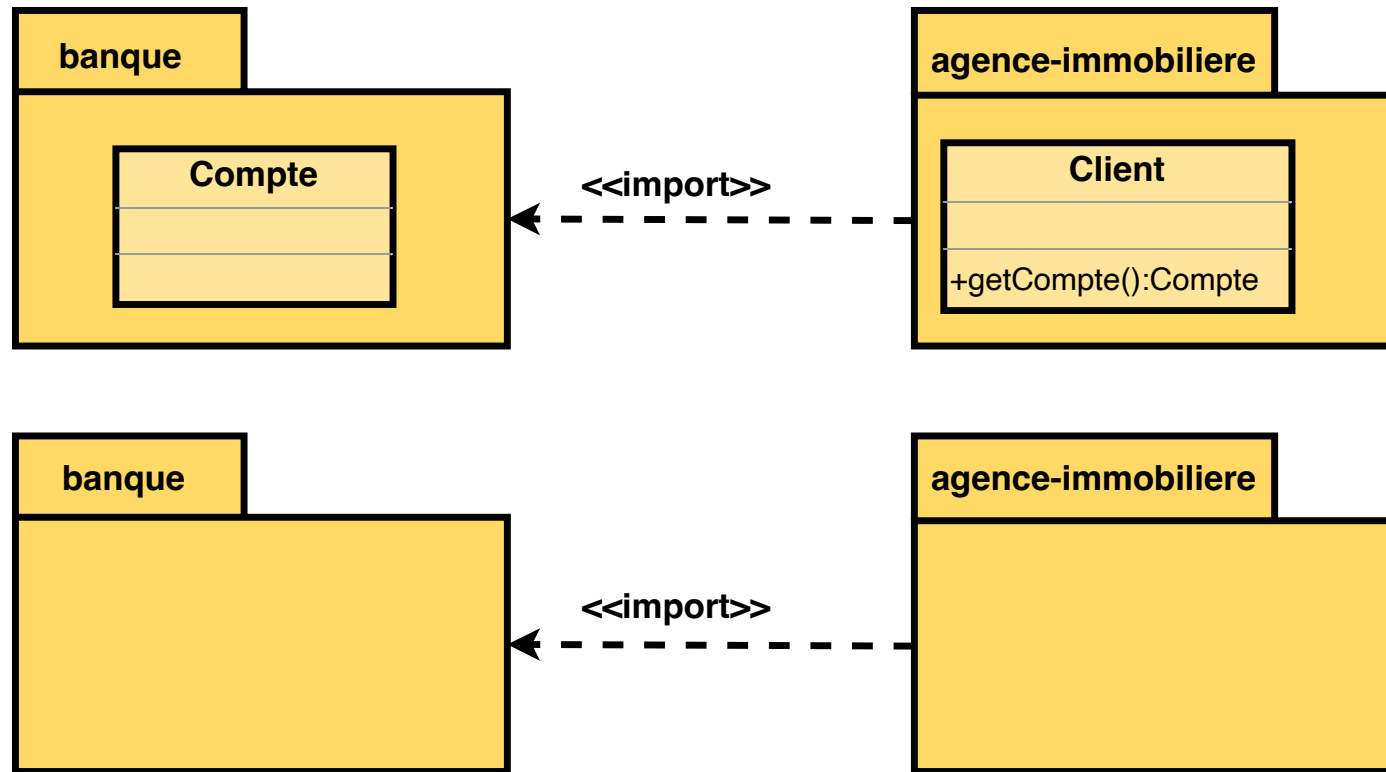


# IMPORTATION DE PAQUETAGE

- Pour **utiliser** des éléments de modélisation définis dans un autre paquetage, on peut utiliser la relation d'**importation**.
  - ▢ permet d'**importer l'espace de nommage** d'un autre paquetage.
- Les **éléments de modélisation** du paquetage importé deviennent **accessible** à tous les membres du paquetage réalisant l'importation.
  - ▢ référençable **sans utiliser explicitement le nom du paquetage importé**.

# IMPORTATION DE PAQUETAGE

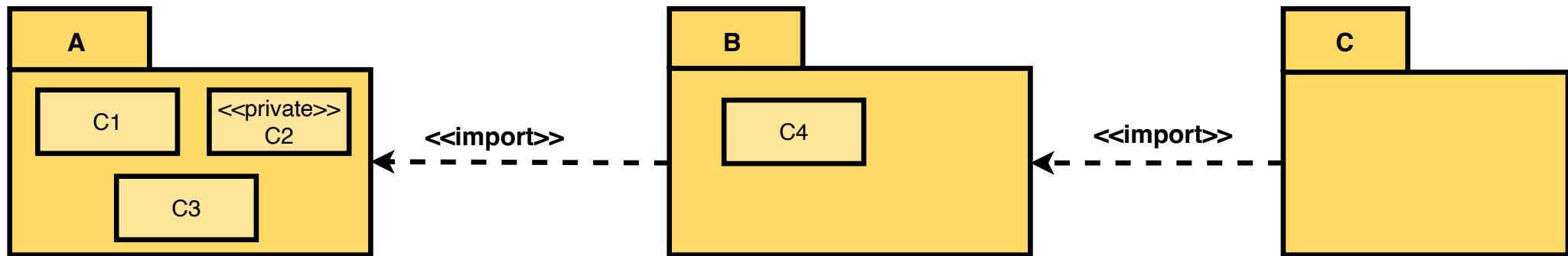
## EXEMPLE 1





# IMPORTATION DE PAQUETAGE

## EXEMPLE 2



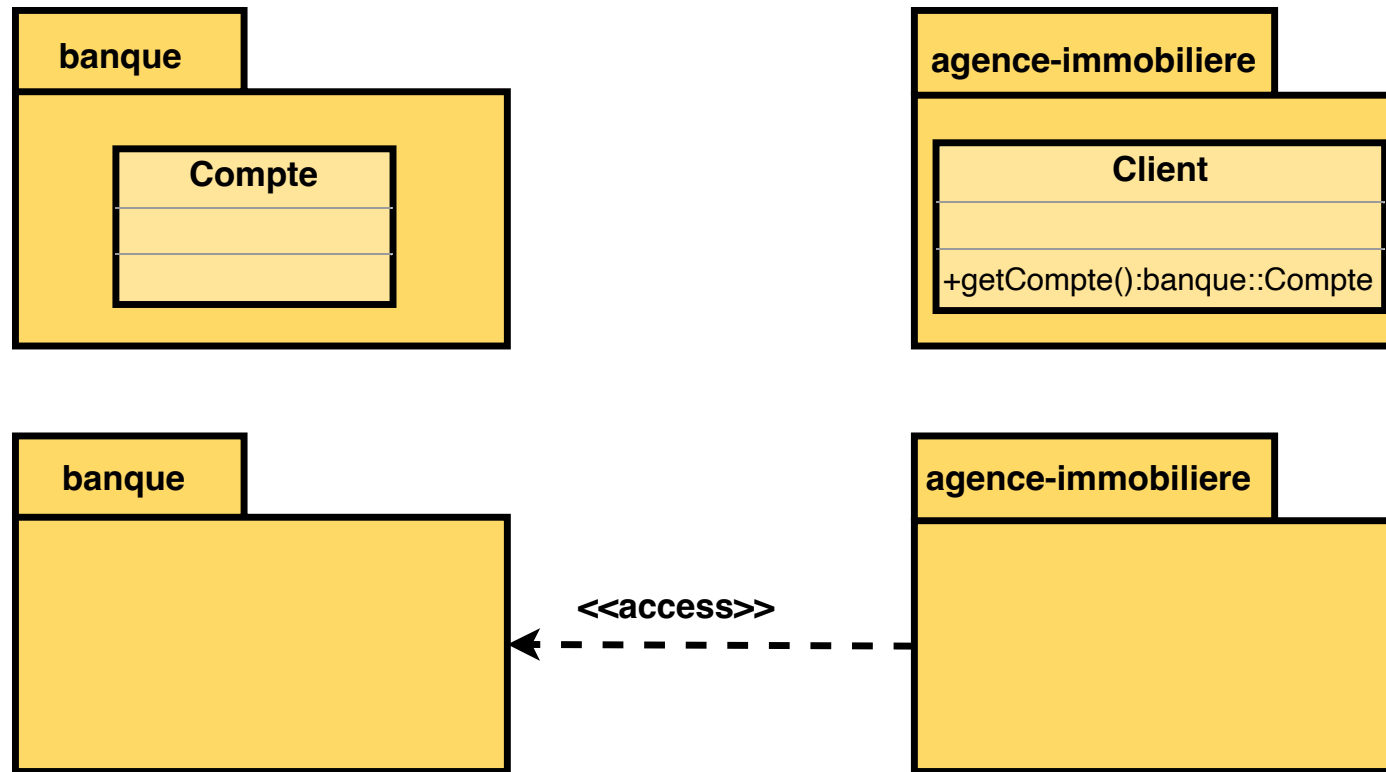
Le paquetage **C** a accès aux classes **C1**, **C3** et **C4**.

# ACCÈS À UN PAQUETAGE

- Il est possible d'**accéder** aux éléments de modélisation d'un paquetage à partir d'un autre paquetage en **utilisant** le **nom complet** de l'élément.
  - ▢ utilisation de **l'espace de nommage** d'un autre paquetage
- L'**accès à un élément** d'un autre espace de nommage **n'est pas une importation** et ne peut être transmis à d'autres paquetages par transitivité.

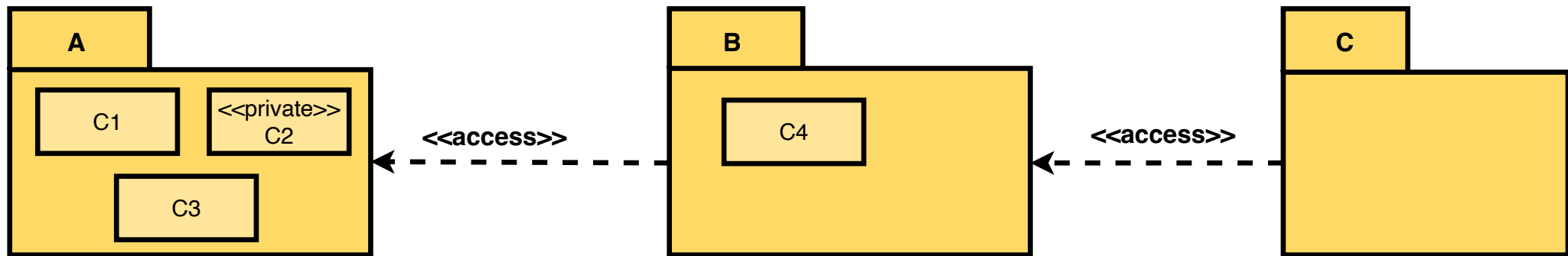
# ACCÈS À UN PAQUETAGE

## EXEMPLE 1



# ACCÈS À UN PAQUETAGE

## EXEMPLE 2



Le paquetage **C** a accès à la classe **C4** seulement.

# PLAN

- Les éléments de base
- Les relations entre paquetages
- Les principes de cohérence et d'indépendance
- Réduction de couplage entre paquetages

[Retour au plan](#) - [Retour à l'accueil](#)

# LA COHÉRENCE ET L'INDÉPENDANCE

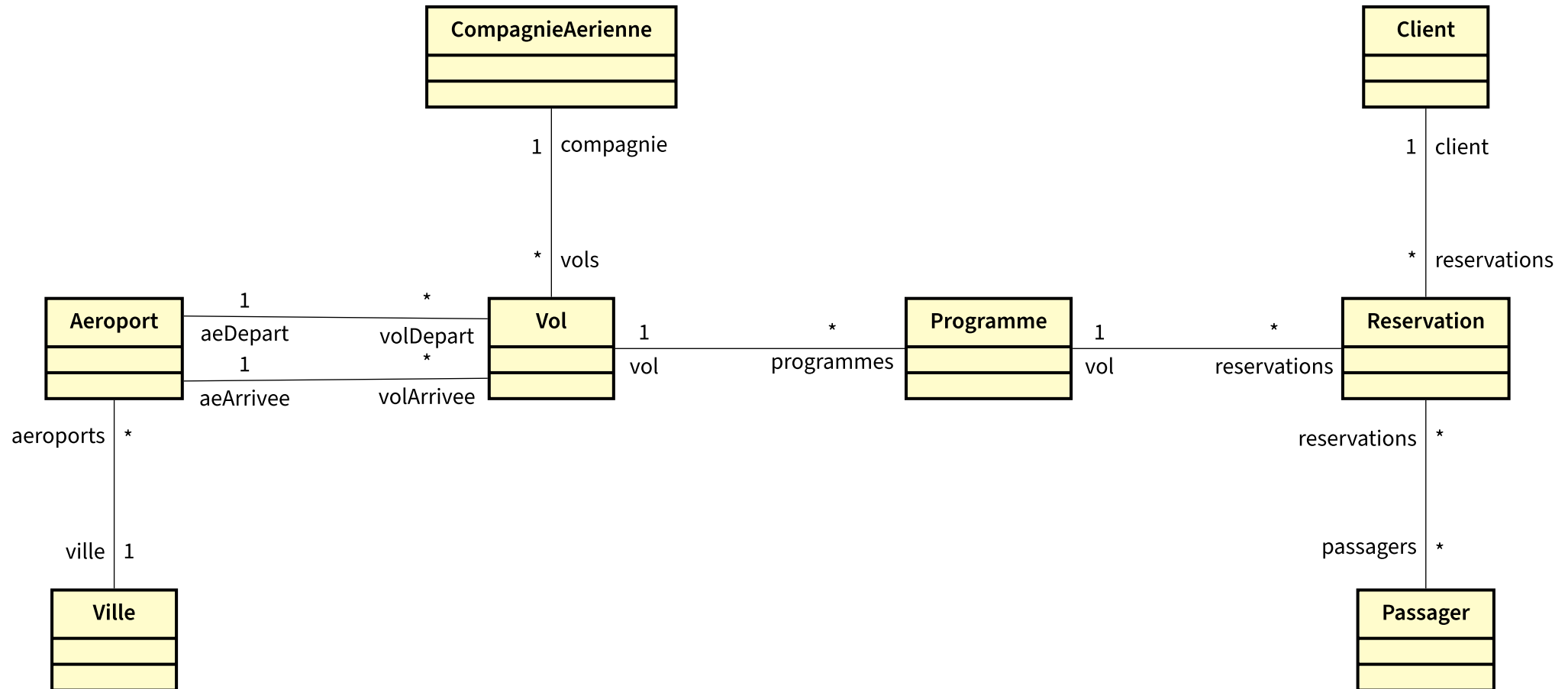
- Le découpage en paquetages doit traduire un découpage logique du système à construire (des espaces de nommage homogènes)
- La structuration d'un modèle dans un diagramme de paquetages s'appuie sur deux principes fondamentaux :
  1. la cohérence → regrouper les éléments proches sémantiquement.
  2. l'indépendance → minimiser les dépendances entre les paquetages.

# LE PRINCIPE DE COHÉRENCE

La **cohérence** consiste à **regrouper** les éléments **proches d'un point de vue sémantique** en suivant les critères suivants:

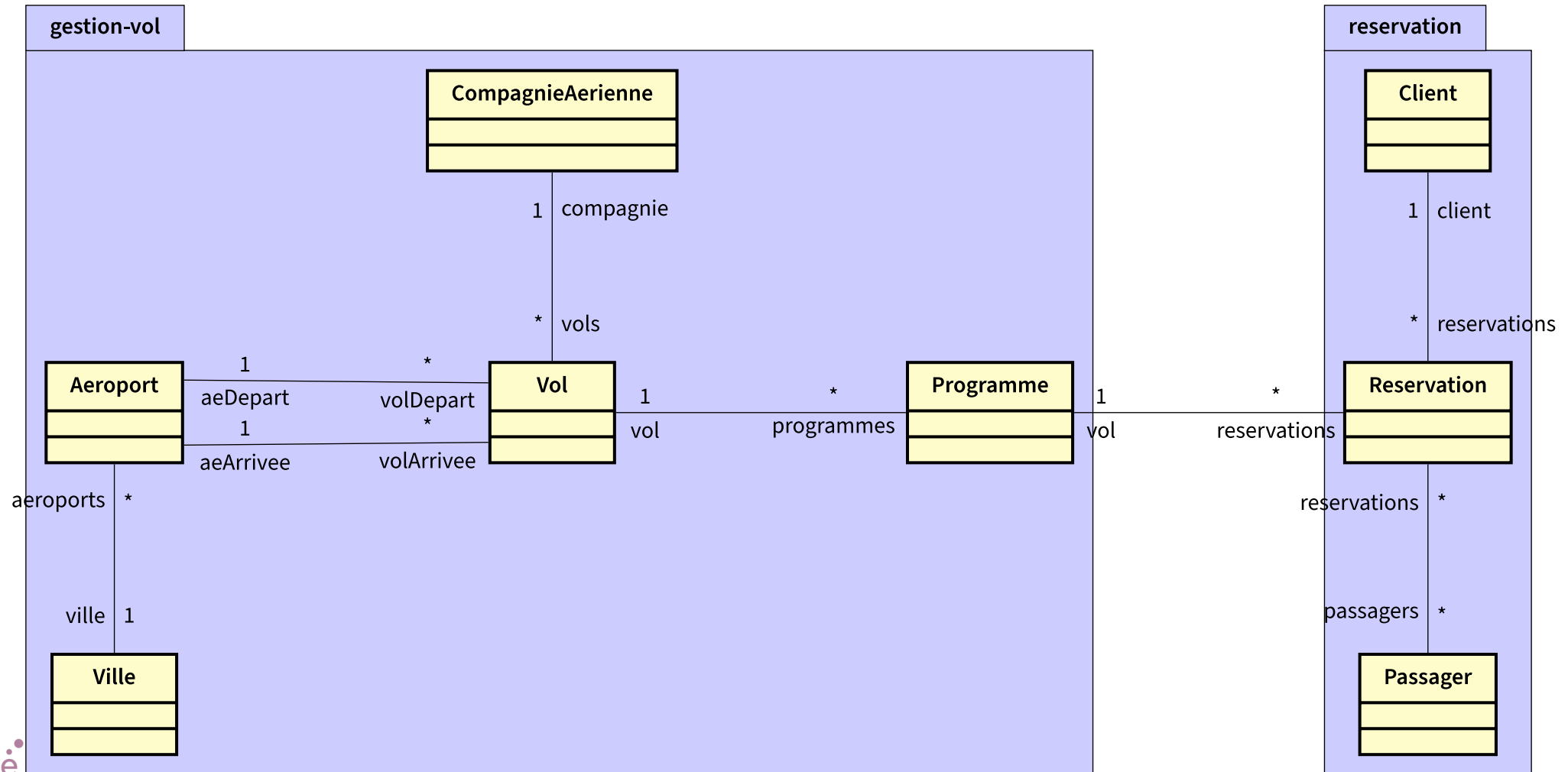
- **finalité** → les classes doivent rendre des **services de même nature**.
- **évolution** → les **classes stables** doivent être isolées de celles qui vont évoluer (les **classes métiers** et les **classes applicatives**).
- **cycle de vie des objets** → les classes doivent être distinguées selon que leurs **objets** ont une **durée de vie** identique ou pas.

# QUEL DÉCOUPAGE ?

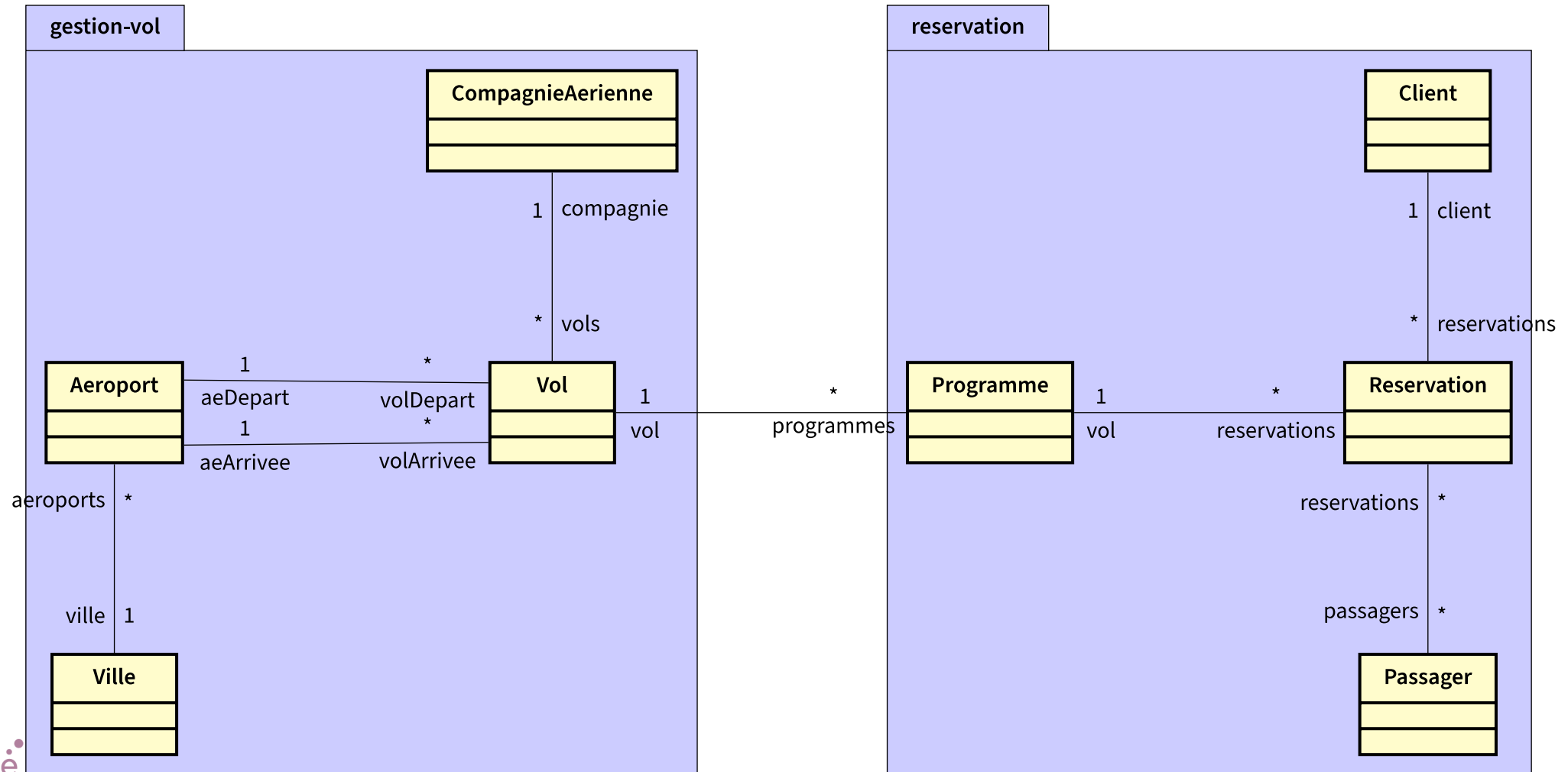




# AVANTAGER LA FINALITÉ



# AVANTAGER L'ÉVOLUTION



# UN DÉCOUPAGE QUI AVANTAGE L'INDÉPENDANCE

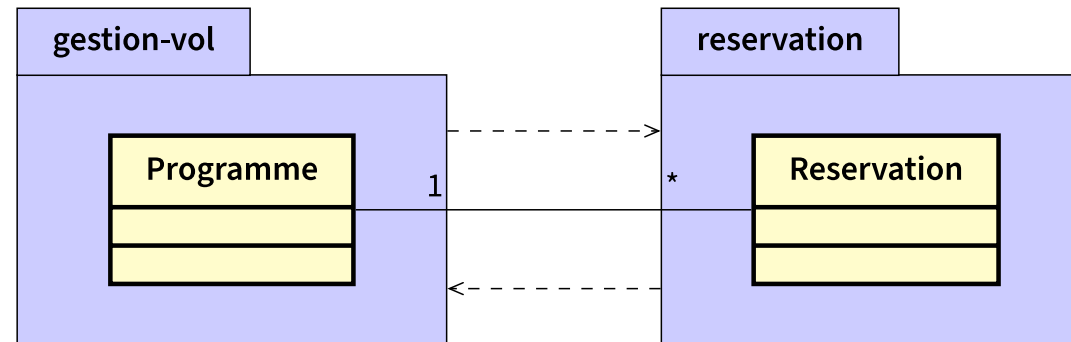
Est ce qu'on peut faire mieux que le résultat obtenu à partir  
des découpages précédents ?

# PLAN

- Les éléments de base
- Les relations entre paquetages
- Les principes de cohérence et d'indépendance
- Réduction de couplage entre paquetages

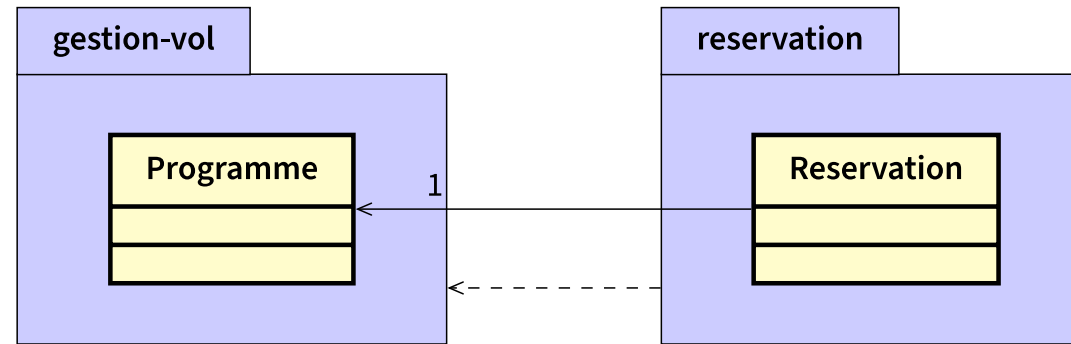
[Retour au plan](#) - [Retour à l'accueil](#)

# RÉDUCTION DE COUPLAGE



- Les associations qui traversent deux paquetages peuvent induire **des dépendances mutuelles**, si elles sont **bidirectionnelles**.
- Le concepteur doit **réduire les dépendances mutuelles**, afin **d'augmenter la modularité et l'évolutivité** de son application.

# PRIVILÉGIER UN SENS DE NAVIGATION

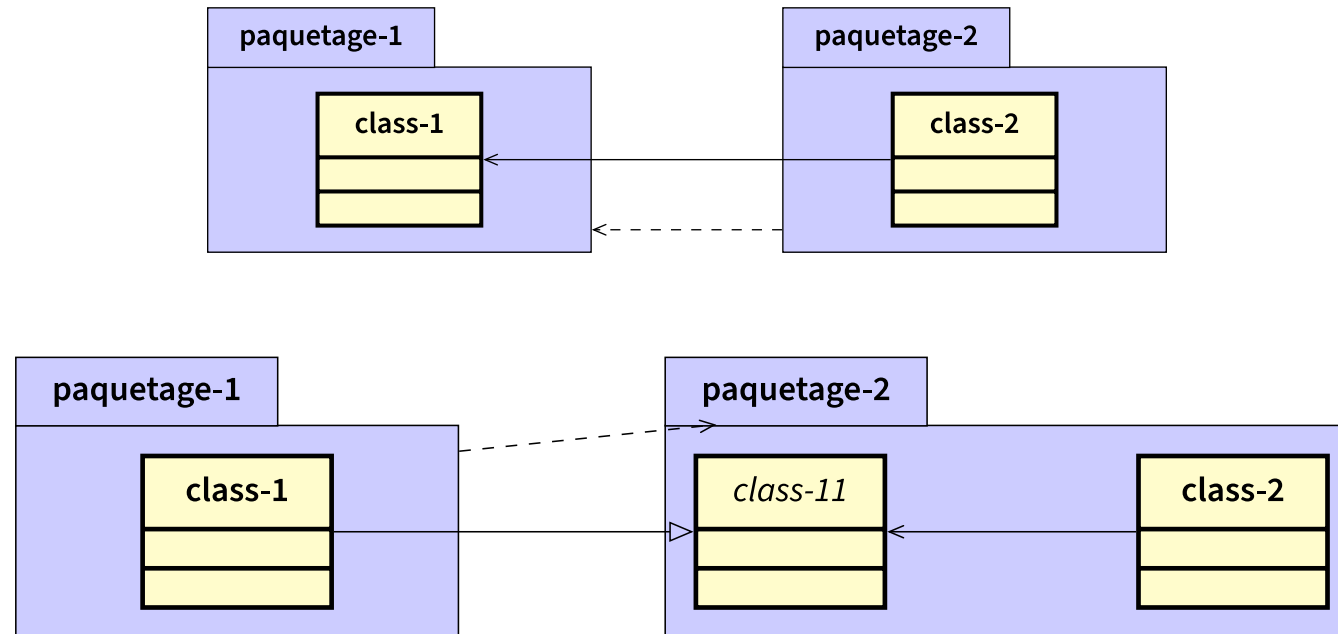


On fait un choix en **priviliégiant un sens de navigation** afin d'éliminer une des deux dépendances

*"il est certain qu'une réservation est en relation forte avec le vol concerné, alors que le vol existe par lui-même, indépendamment de toute réservation"*

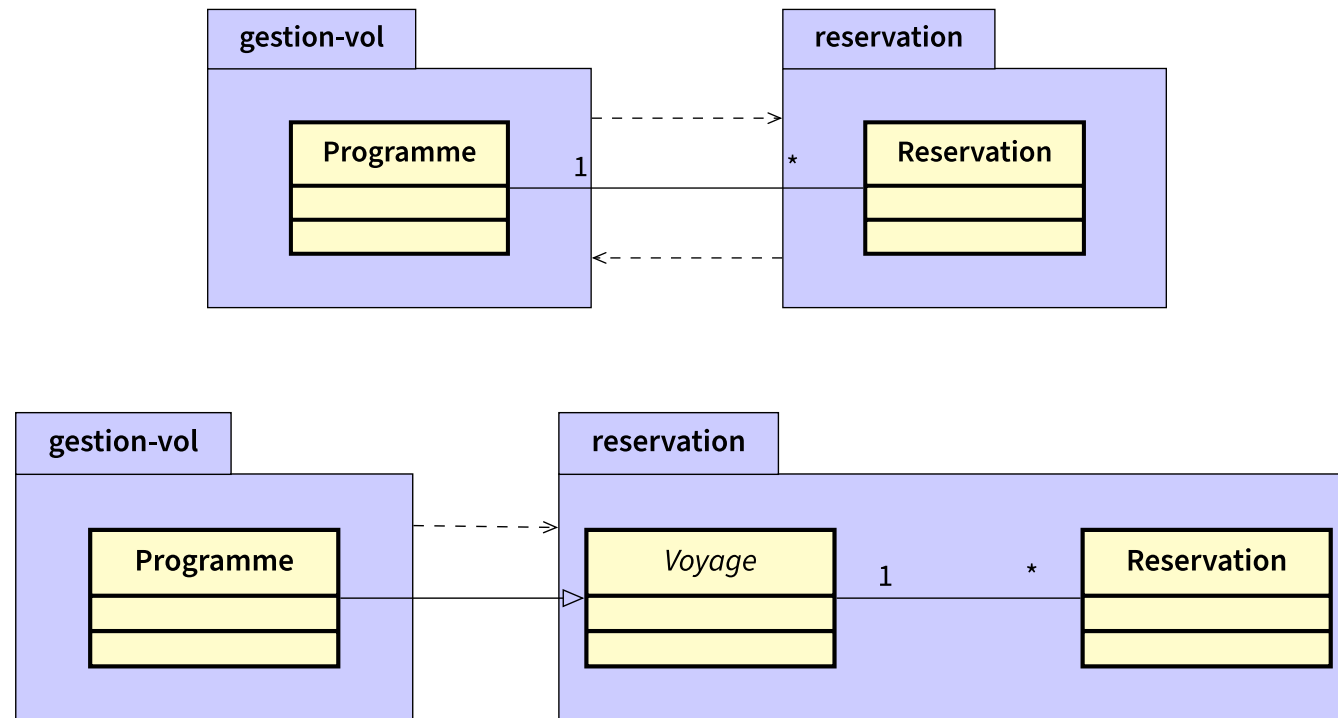
# INVERSER UNE DÉPENDANCE

L'inversion d'une dépendance s'effectue, en **introduisant une classe abstraite** (ou une interface), de la façon suivante :



# INVERSER UNE DÉPENDANCE

## EXEMPLE





# MERCI

[Version PDF des slides](#)

[Retour à l'accueil](#) - [Retour au plan](#)