



université  
PARIS-SACLAY

université  
PARIS-SACLAY  
IUT D'ORSAY

# QUALITÉ DE DÉVELOPPEMENT

## DIAGRAMME UML DE PAQUETAGE

🎓 2A - Bachelor Universitaire de Technologie  
🏛️ IUT d'Orsay - Université Paris-Saclay - 2022/2023



**Idir AIT SADOUNE** 🌐

[idir.aitsadoune@centralesupelec.fr](mailto:idir.aitsadoune@centralesupelec.fr) ✉️

# PLAN

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

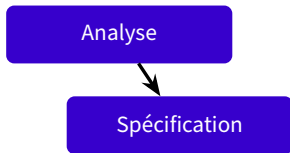
[Retour à l'accueil](#) - [Retour au plan](#)

# CYCLE DE DÉVELOPPEMENT EN V

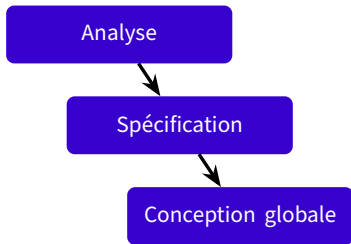
# CYCLE DE DÉVELOPPEMENT EN V

Analyse

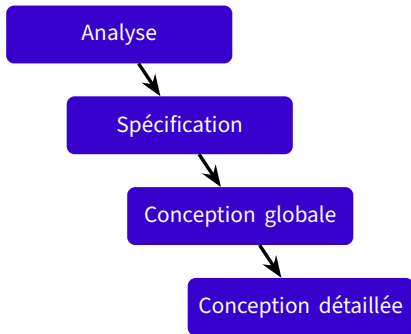
# CYCLE DE DÉVELOPPEMENT EN V



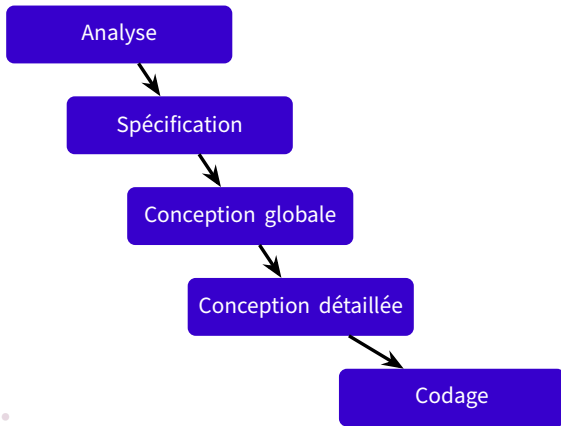
# CYCLE DE DÉVELOPPEMENT EN V



# CYCLE DE DÉVELOPPEMENT EN V

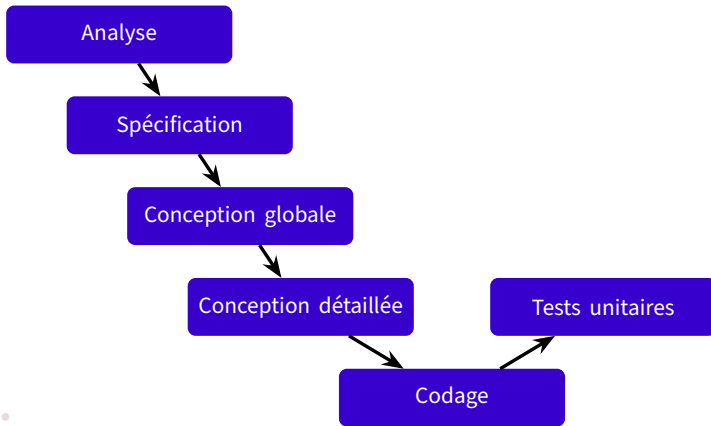


# CYCLE DE DÉVELOPPEMENT EN V

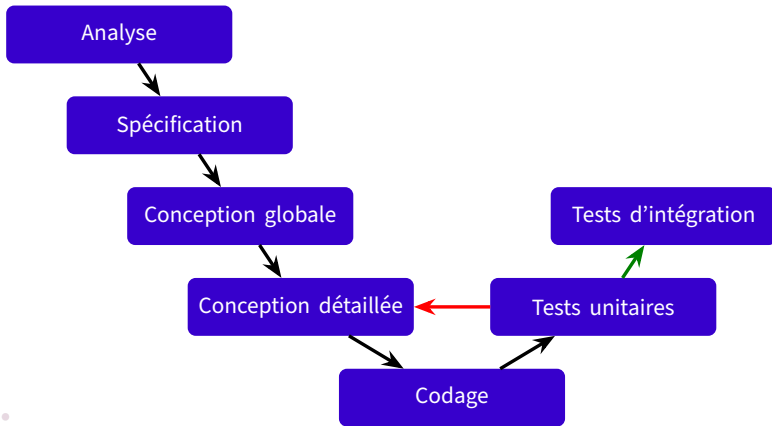




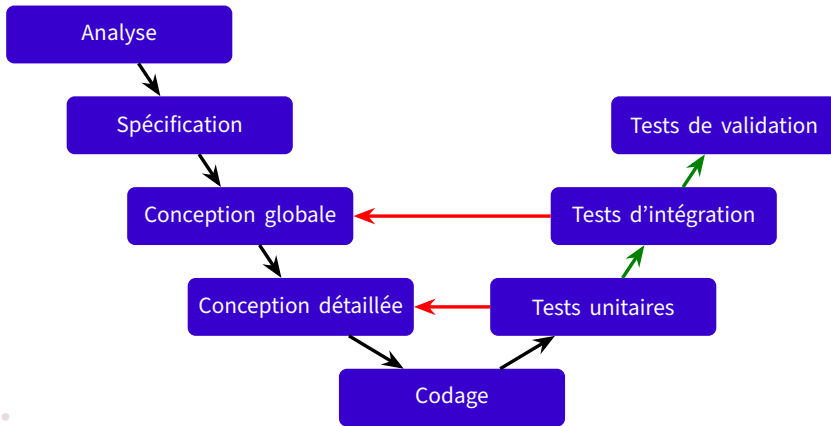
# CYCLE DE DÉVELOPPEMENT EN V



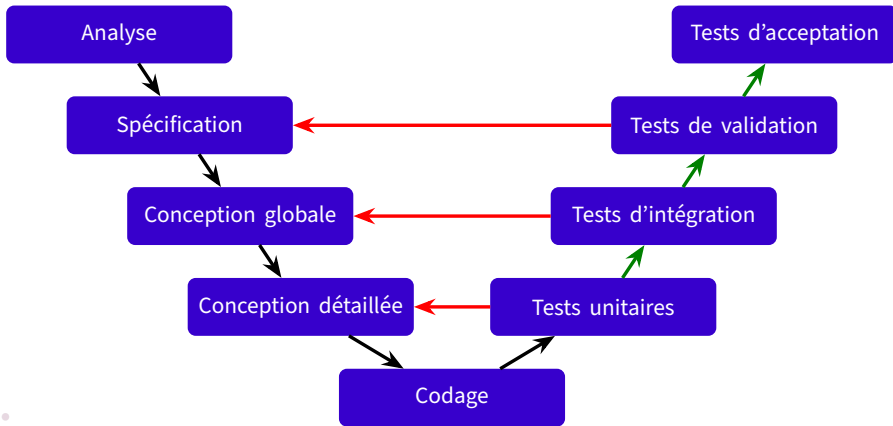
# CYCLE DE DÉVELOPPEMENT EN V



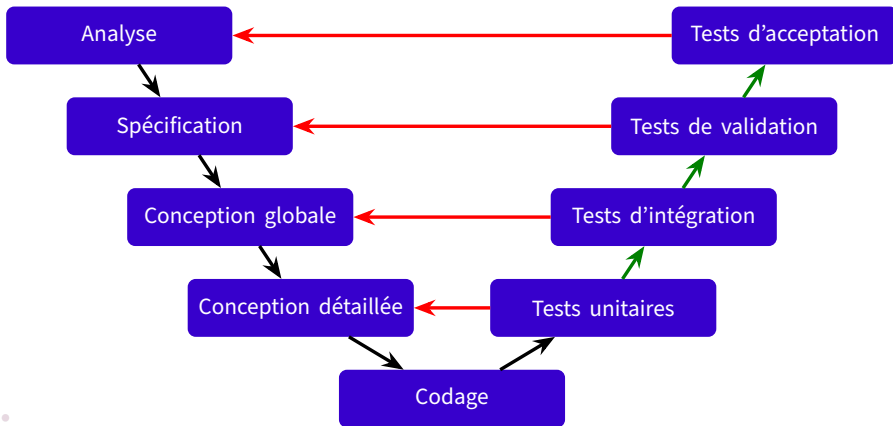
# CYCLE DE DÉVELOPPEMENT EN V



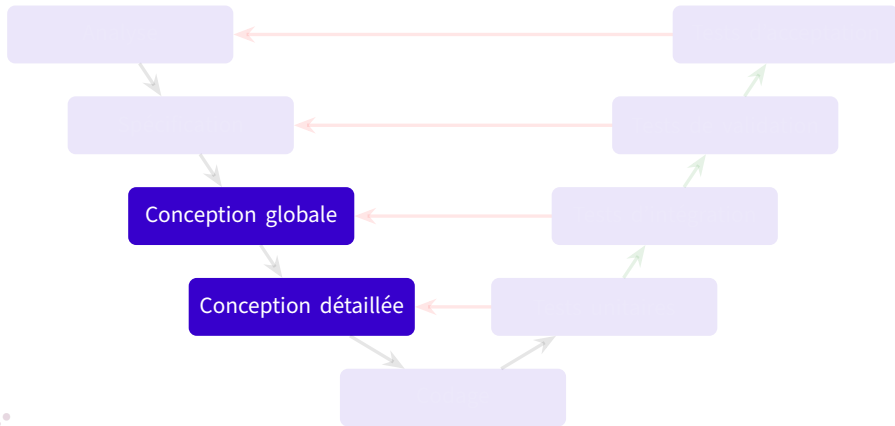
# CYCLE DE DÉVELOPPEMENT EN V



# CYCLE DE DÉVELOPPEMENT EN V



# LES ACTIVITÉS DE CONCEPTION

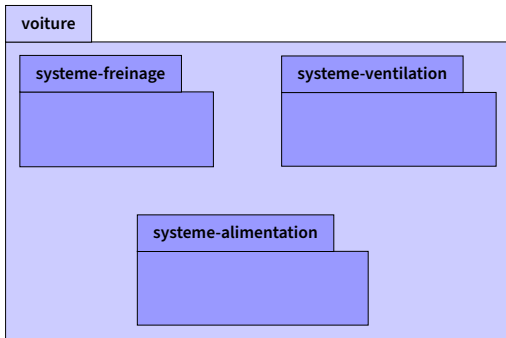


# PLAN

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

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

# EXEMPLE D'INTRODUCTION





# DÉFINITIONS

# DÉFINITIONS

- Un **paquetage** regroupe des éléments de la modélisation appelés **membres**, portant sur **un sous-ensemble du système**.

# DÉFINITIONS

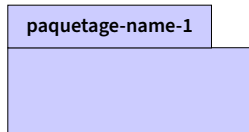
- Un **paquetage** regroupe des éléments de la modélisation appelés **membres**, portant sur **un sous-ensemble du système**.
- Le **découpage en paquetage** doit traduire **un découpage logique** du système correspondant à **des espaces de nommage homogènes**.

# DÉFINITIONS

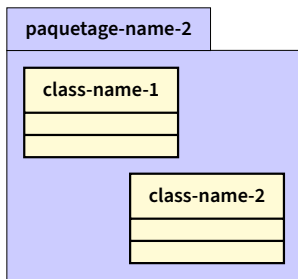
- Un **paquetage** regroupe des éléments de la modélisation appelés **membres**, portant sur **un sous-ensemble du système**.
- Le **découpage en paquetage** doit traduire **un découpage logique** du système correspondant à **des espaces de nommage homogènes**.
- Un **paquetage** permet de grouper **n'importe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

# PRÉSENTATION D'UN PAQUETAGE

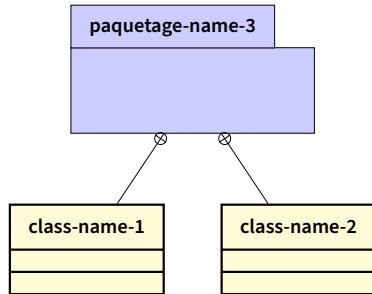
Représentation globale



Représentation détaillée



Représentation éclatée



# NOM D'UN PAQUETAGE

# NOM D'UN PAQUETAGE

- Chaque **paquetage** doit avoir un **nom différent** (**espace de nommage**).

# NOM D'UN PAQUETAGE

- Chaque **paquetage** doit avoir un **nom différent** (**espace de nommage**).
- Les éléments contenus dans un paquetage se distinguent par leur **appartenance** au **paquetage englobant**.



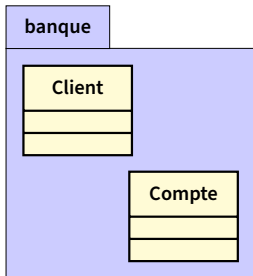
# NOM D'UN PAQUETAGE

- Chaque **paquetage** doit avoir un **nom différent** (**espace de nommage**).
- Les éléments contenus dans un paquetage se distinguent par leur **appartenance** au **paquetage englobant**.
  - ➡ **deux éléments** dans **deux paquetages** peuvent porter **le même nom**.

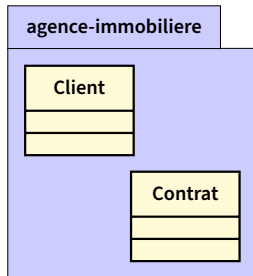
# NOM D'UN PAQUETAGE

- Chaque **paquetage** doit avoir un **nom différent** (**espace de nommage**).
- Les éléments contenus dans un paquetage se distinguent par leur **appartenance** au **paquetage englobant**.
  - ➡ **deux éléments** dans **deux paquetages** peuvent porter **le même nom**.
  - ➡ **deux éléments** dans **le même paquetage** doivent porter **des noms différents**.

# EXEMPLE

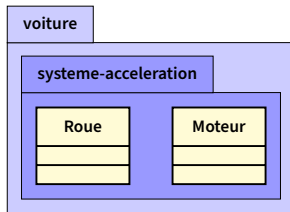


`banque::Client`

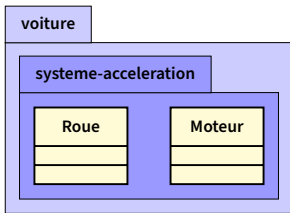


`agence-immobiliere::Client`

# NOM D'UN ÉLÉMENT

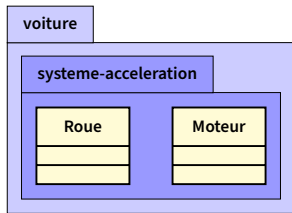


# NOM D'UN ÉLÉMENT



- Le **nom** d'un élément est dit **simple** s'il est utilisé **seul**.
  - ➡ la classe **Roue**

# NOM D'UN ÉLÉMENT

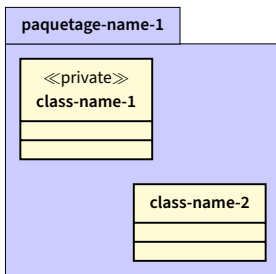


- Le **nom** d'un élément est dit **simple** s'il est utilisé **seul**.
  - ➡ la classe **Roue**
- Le **nom** d'un élément est dit **complet** s'il est précédé par **les noms des paquetages englobants (name space)**.

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



# RELATION DE DÉPENDANCE

# RELATION DE DÉPENDANCE

- Une relation de dépendance doit exister dès que deux éléments issus de deux paquetages différents sont associés.
  - ➡ hormis les cas de dépendances implicites : emboîtement de paquetages.

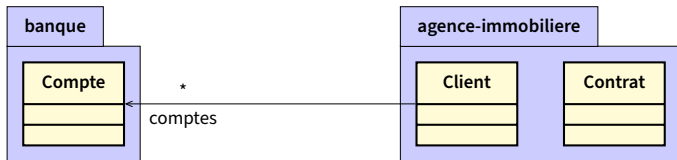
# RELATION DE DÉPENDANCE

- Une relation de dépendance doit exister dès que deux éléments issus de deux paquetages différents sont associés.
  - ➡ hormis les cas de dépendances implicites : emboîtement de paquetages.
- C'est une relation unidirectionnelle entre paquetages.
  - ➡ une modification de la cible peut impliquer une modification de la source.

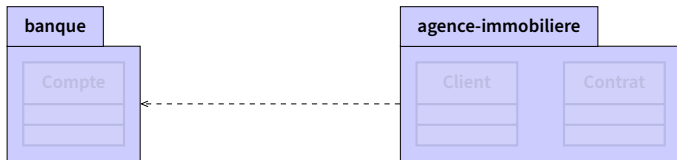
# RELATION DE DÉPENDANCE

- Une relation de dépendance doit exister dès que deux éléments issus de deux paquetages différents sont associés.
  - ➡ hormis les cas de dépendances implicites : emboîtement de paquetages.
- C'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.

# EXEMPLE



# EXEMPLE



# IMPORTATION DE PAQUETAGE

# IMPORTATION DE PAQUETAGE

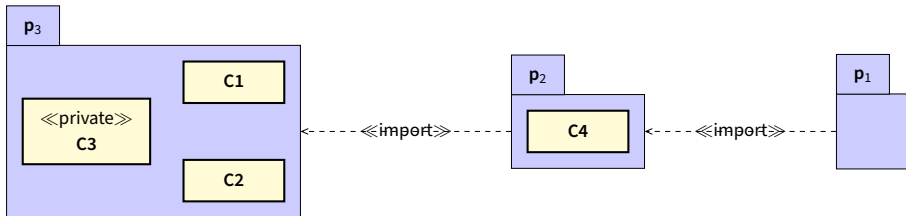
- Pour **simplifier l'utilisation des éléments** contenus dans un autre paquetage, on peut utiliser l'**importation**.
  - ➡ permet d'**importer l'espace de nommage** d'un autre paquetage.



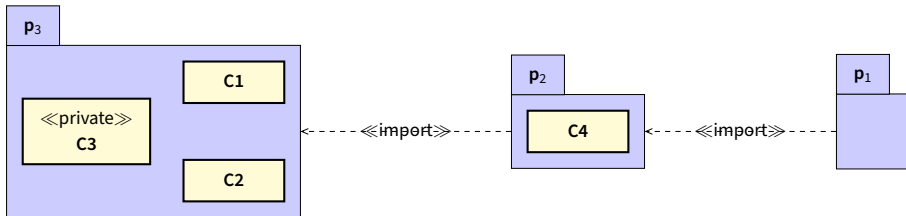
# IMPORTATION DE PAQUETAGE

- Pour **simplifier l'utilisation des éléments** contenus dans un autre paquetage, on peut utiliser l'**importation**.
  - ➡ permet d'**importer l'espace de nommage** d'un autre paquetage.
- Tous les membres d'un paquetage ont **accès à tous les membres du paquetage importé**.
  - ➡ sans utiliser explicitement le nom du paquetage importé.

# EXEMPLE



# EXEMPLE



Le paquetage p<sub>1</sub> a accès aux classes C1,C2 et C4.

# ACCÈS À UN PAQUETAGE

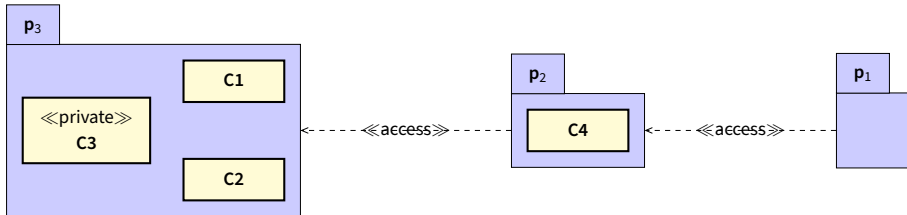
# ACCÈS À UN PAQUETAGE

- Pour **accéder** aux éléments d'un paquetage à partir d'un autre paquetage, il faut **utiliser le nom complet**.
  - ▶ permet pour un paquetage d'avoir **accès à l'espace de nommage d'un autre paquetage**.

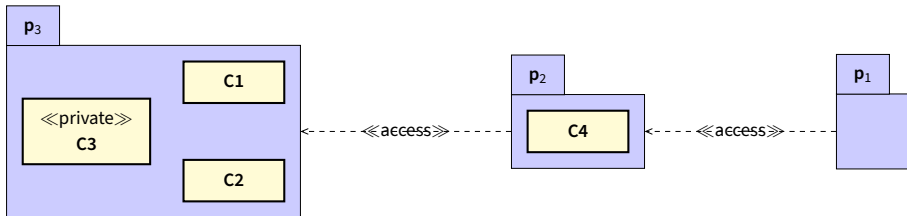
# ACCÈS À UN PAQUETAGE

- Pour **accéder** aux éléments d'un paquetage à partir d'un autre paquetage, il faut **utiliser le nom complet**.
  - permet pour un paquetage d'avoir **accès à l'espace de nommage d'un autre paquetage**.
- L'espace de nommage n'est **pas importé et ne peut être transmis à d'autres paquetages** par transitivité.

# EXEMPLE 1



# EXEMPLE 1



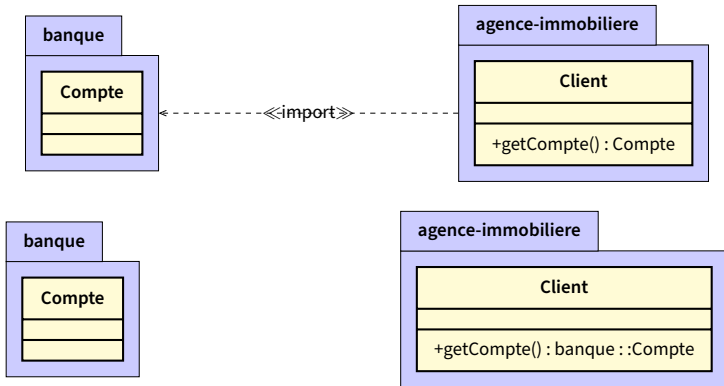
Le paquetage p<sub>1</sub> a accès à la classe C4 seulement.



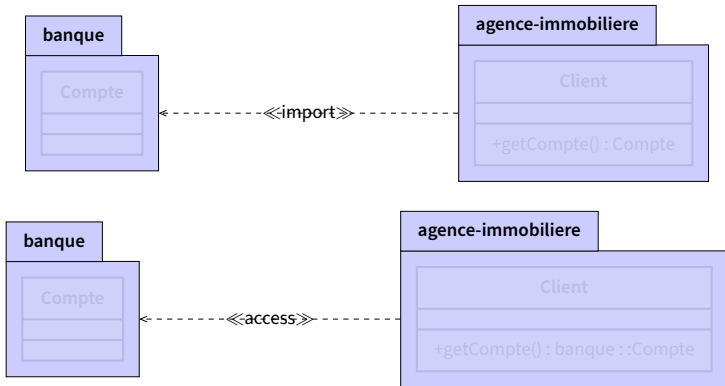
## EXEMPLE 2



# EXEMPLE 2



## EXEMPLE 2



# PLAN

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

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

# LA COHÉRENCE ET L'INDÉPENDANCE

# LA COHÉRENCE ET L'INDÉPENDANCE

- Le découpage en paquetages doit traduire un découpage logique du système à construire.
  - des espaces de nommage homogènes.

# 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 :

# 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.



# 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

# LE PRINCIPE DE COHÉRENCE

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

# LE PRINCIPE DE COHÉRENCE

- **La cohérence** consiste à **regrouper** les éléments de modélisation **proches** d'un point de vue **sémantique** en suivant les critères suivants :
  - **Finalité** : les classes doivent rendre des **services de même nature**.

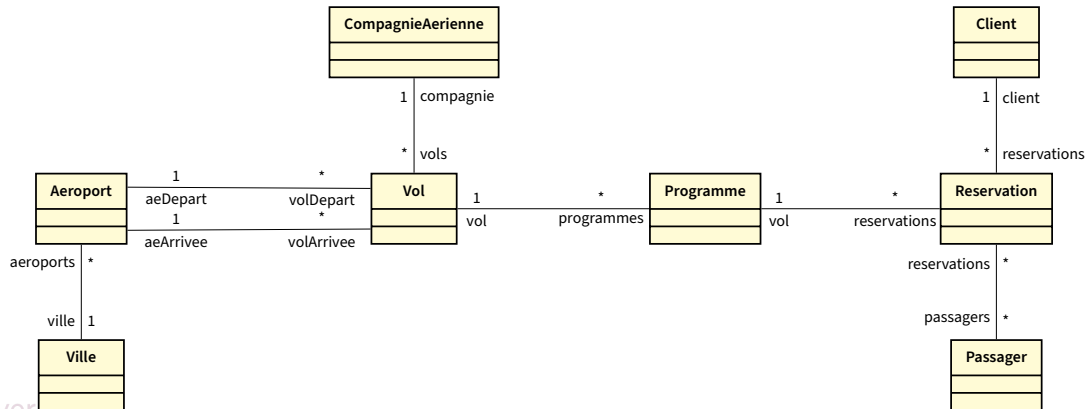
# LE PRINCIPE DE COHÉRENCE

- **La cohérence** consiste à regrouper les éléments de modélisation proches d'un point de vue sémantique en suivant les critères suivants :
  - **Finalité** : les classes doivent rendre des services de même nature.
  - **Evolution** : les classes stables doivent être isolées de celles qui vont évoluer (les classes métiers et les classes applicatives).

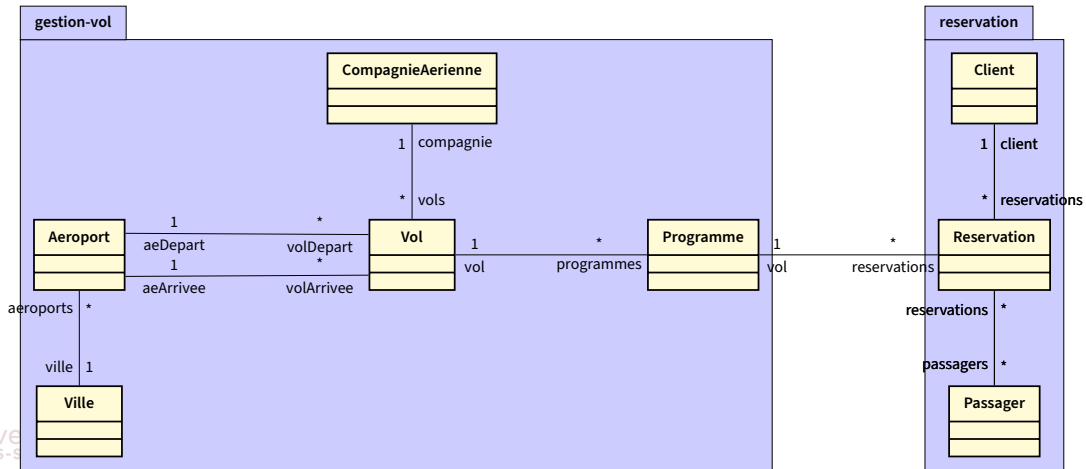
# LE PRINCIPE DE COHÉRENCE

- **La cohérence** consiste à **regrouper** les éléments de modélisation **proches** d'un **point de vue sémantique** en suivant les critères suivants :
  - **Finalité** : les classes doivent rendre des **services de même nature**.
  - **Evolution** : 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 ?

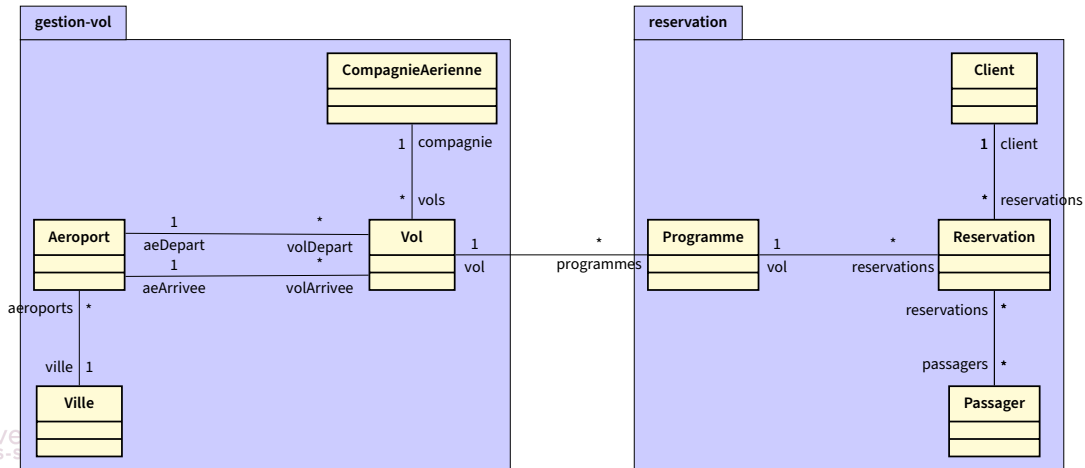


# AVANTAGER LA FINALITÉ





# AVANTAGER L'ÉVOLUTION



# UN DÉCOUPAGE QUI AVANTAGE L'INDÉPENDANCE

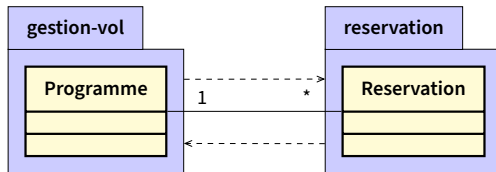
Est ce qu'on peut faire mieux que le résultat obtenu à partir  
des découpages précédents ?

# PLAN

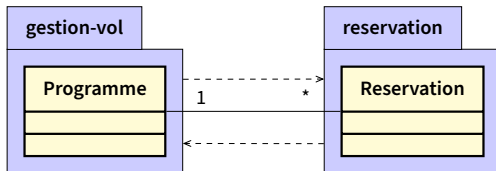
- Les éléments de base
- Les relations entre paquetages
- Les principes de cohérence et d'indépendance
- Réduction de couplage entre paquetages

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

# RÉDUCTION DE COUPLAGE

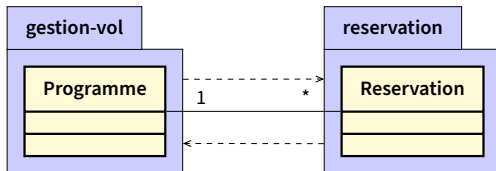


# RÉDUCTION DE COUPLAGE



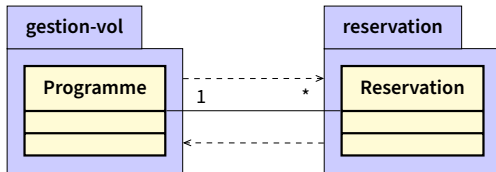
- Les associations qui traversent deux paquetages peuvent induire des dépendances mutuelles, si elles sont bidirectionnelles.

# RÉDUCTION DE COUPLAGE

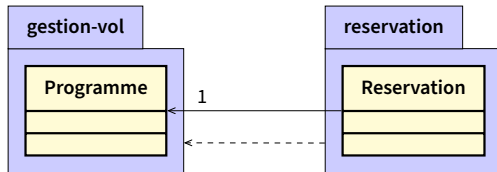


- Les associations qui traversent deux paquetages peuvent induire des dépendances mutuelles, si elles sont bidirectionnelles.
- Le concepteur doit réduire les dépendances mutuelles ou cycliques, afin d'augmenter la modularité et l'évolutivité de son application.

# PRIVILÉGIER UN SENS DE NAVIGATION



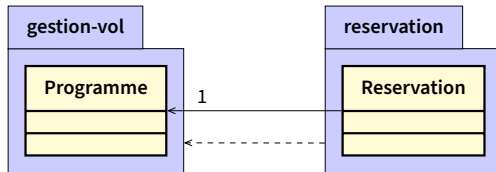
# PRIVILÉGIER UN SENS DE NAVIGATION



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



# PRIVILÉGIER UN SENS DE NAVIGATION



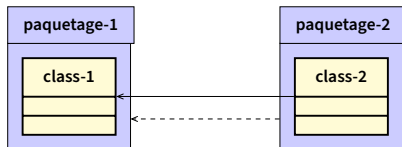
- On fait un choix en **priviliégiant un sens de navigation** afin d'éliminer une des deux dépendances :
  - ➡ *il est certain qu'une réservation est en relation forte avec le vol concerné, alors que le vol existe par lui-même, indépendamment de toute réservation.*

# INVERSER UNE DÉPENDANCE

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

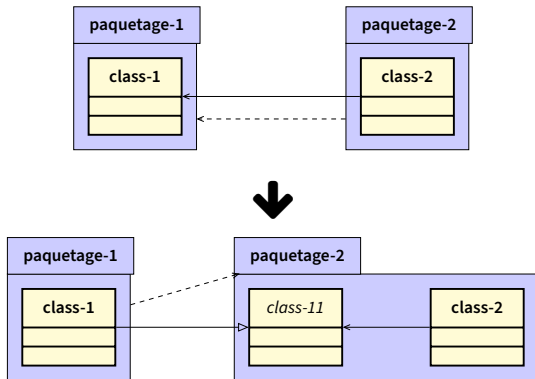
# INVERSER UNE DÉPENDANCE

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

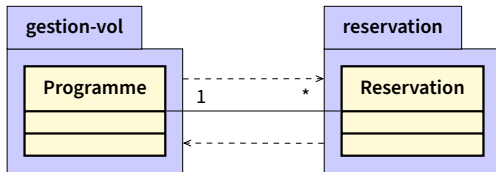


# INVERSER UNE DÉPENDANCE

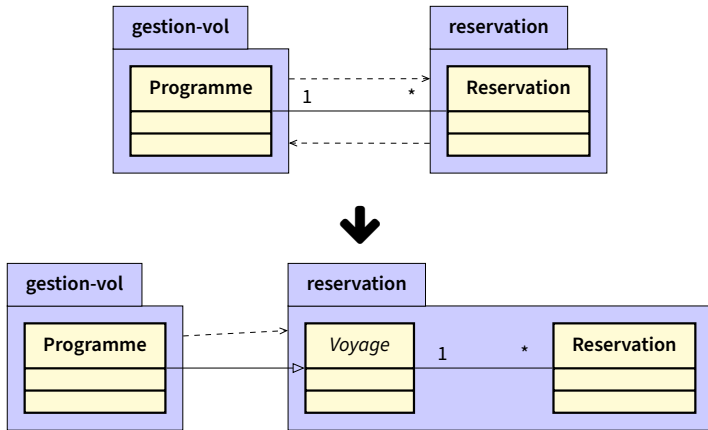
- L'inversion d'une dépendance s'effectue, en introduisant une classe abstraite (ou une interface), de la façon suivante :



# EXEMPLE



# EXEMPLE



# THE END

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