Génie Logiciel et projet de synthèse

Analyse statique

Kévin Bailly
Institut des Systèmes Intelligents et de Robotique

kevin.bailly@upmc.fr

Exemple

- Système simplifié de réservation de vols (résumé des connaissances des experts du domaine)
  1. Les compagnies aériennes proposent différents vols
  2. Un vol est ouvert à la réservation et refermé sur ordre de la compagnie
  3. Un client peut réserver un ou plusieurs vols, pour des passagers différents
  4. Une réservation concerne un seul vol et un seul passager
  5. Une réservation peut être confirmée ou annulée
  6. Un vol a un aéroport de départ et un aéroport d'arrivée
  7. Un vol a un jour et une heure de départ et d'arrivée
  8. Un vol peut comporter des escales dans des aéroports
  9. Une escale a une heure d'arrivée et de départ
  10. Chaque aéroport dessert une ou plusieurs ville

Objectifs

UPMC SORBONNE UNIVERSITÉS

- L'objectif de la phase de modélisation statique est :
  - Identifier les **concepts** du domaine et les modéliser sous forme de **classes**
  - Identifier les **associations** entre les concepts et définir les **multiplicités** à chaque extrémité des associations
  - Définir les **attributs** et les **opérations**
  - Structurer en **packages** cohérents et indépendants
  - Concevoir des modules **réutilisables**

Plan

- Diagramme de classe
- Exemple de modélisation
- Patrons d'analyse
- Structuration en package

Diagramme de classe

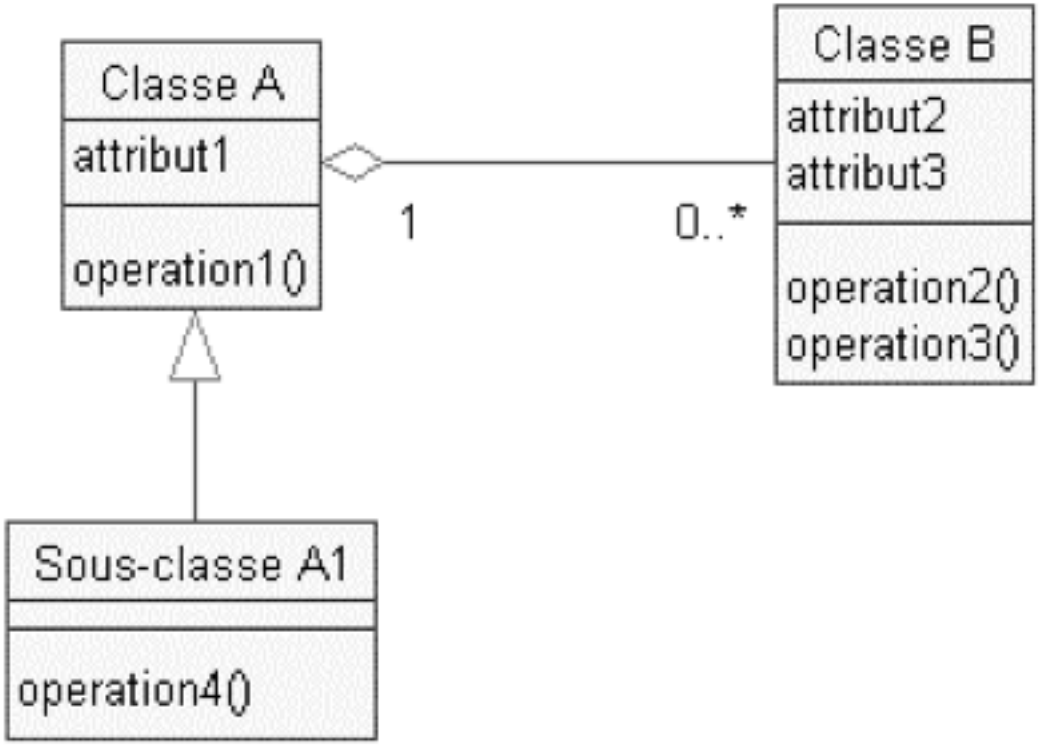
- Au cœur de la **modélisation orientée objet**
  Mais indépendant du langage de programmation

- Décrit la **structure du système** (vue interne)
  ≠ Diagramme des cas d'utilisation qui décrit le système du point de vue des acteurs (vue externe)

- **Statique** : l'évolution du système dans le temps n'est pas modélisée

- Représente principalement les **classes** et leurs **relation**

Représentation



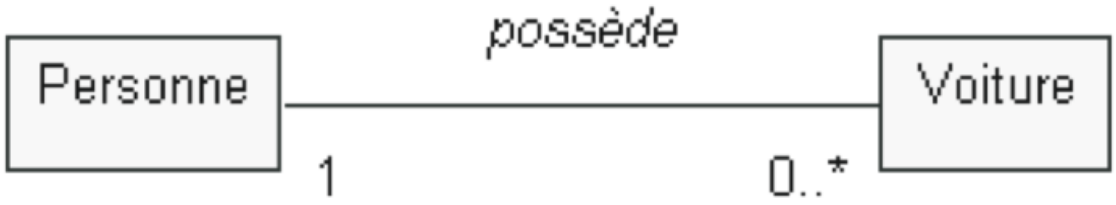
Classes

- Ensemble d'objets possédant les mêmes caractéristiques = type

- **Attributs** : types d'informations contenus dans une classe
- **Opérations** : types de comportement (services)

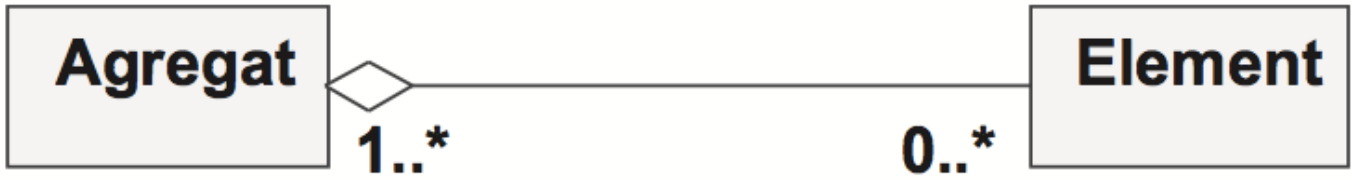
Association

- Relation sémantique durable entre deux classes
- Une association est nommée (verbe)
- Aux extrémités : information de multiplicité



Agrégation et composition

- Agrégation :
  - Exprime une relation de contenance
  - Relation non symétrique
  - Nommée implicitement : « est composé de »

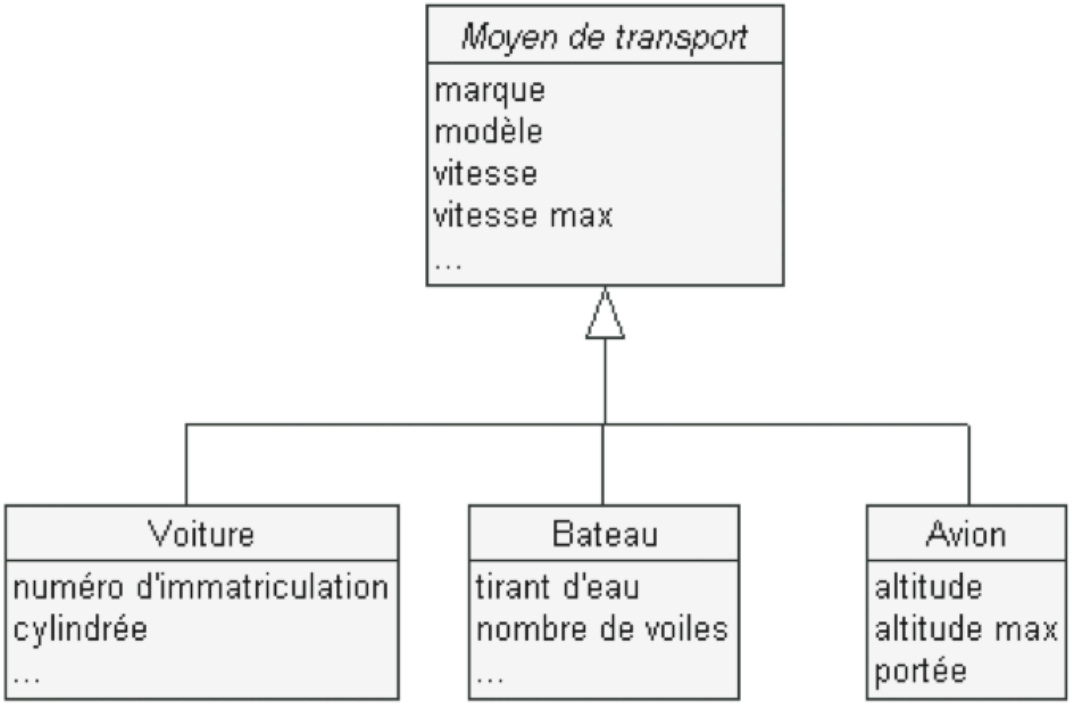


Agrégation et composition

- Composition :
  - Cas particulier d'agrégation
  - Un élément ne peut appartenir qu'à un seul agrégat
  - La destruction de l'agrégat implique la destruction de tous ses éléments

Généralisation

- Une **super-classe** est une classe plus générale reliée à une ou plusieurs **sous-classes** (classes plus spécialisées) par une relation de **généralisation**

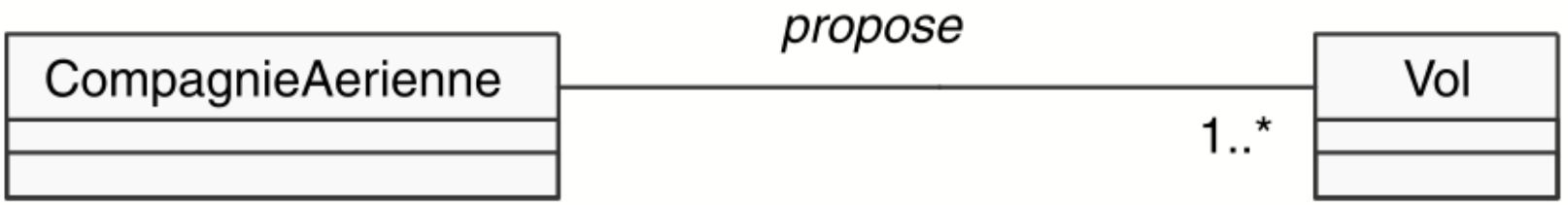


Exemple

- Système simplifié de réservation de vols (résumé des connaissances des experts du domaine)
  1. Les compagnies aériennes proposent différents vols
  2. Un vol est ouvert à la réservation et refermé sur ordre de la compagnie
  3. Un client peut réserver un ou plusieurs vols, pour des passagers différents
  4. Une réservation concerne un seul vol et un seul passager
  5. Une réservation peut être confirmée ou annulée
  6. Un vol a un aéroport de départ et un aéroport d'arrivée
  7. Un vol a un jour et une heure de départ et d'arrivée
  8. Un vol peut comporter des escales dans des aéroports
  9. Une escale a une heure d'arrivée et de départ
  10. Chaque aéroport dessert une ou plusieurs ville

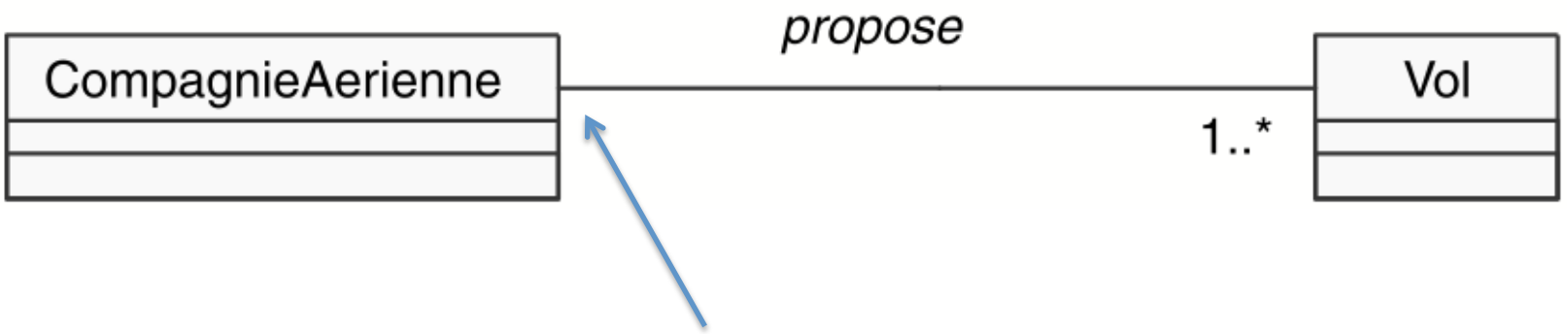
Exemple de modélisation

Les compagnies aériennes proposent différents vols



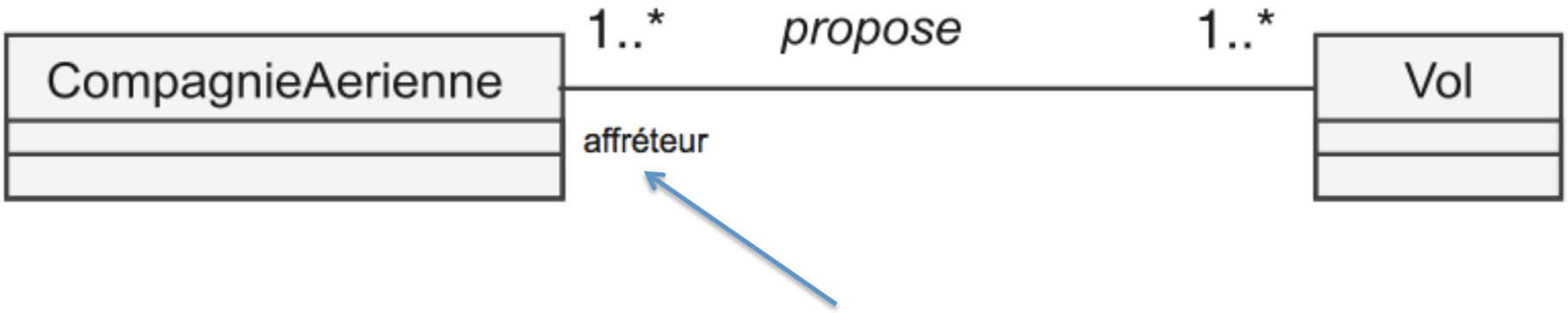
Exemple de modélisation

Les compagnies aériennes proposent différents vols



Pas d'indications sur la multiplicité : poser la question à l'expert

- Mais pour la suite….



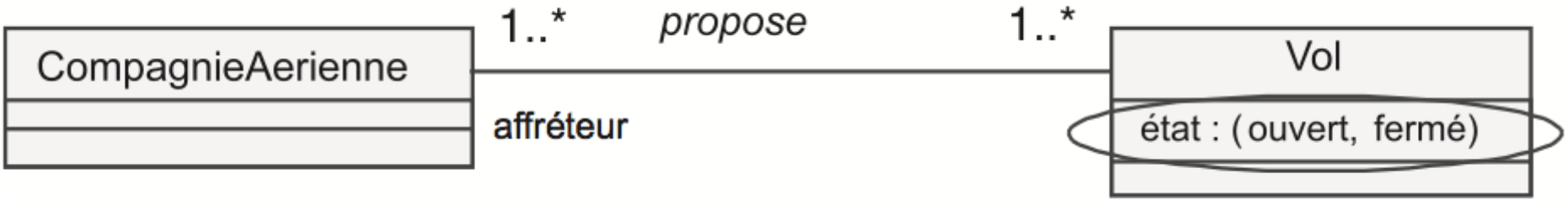
Role joué par la classe dans la relation

Exemple

- Système simplifié de réservation de vols (résumé des connaissances des experts du domaine)
  1. Les compagnies aériennes proposent différents vols
  2. Un vol est ouvert à la réservation et refermé sur ordre de la compagnie
  3. Un client peut réserver un ou plusieurs vols, pour des passagers différents
  4. Ue réservation concerne un seul vol et un seul passager
  5. Une réservation peut être confirmée ou annulée
  6. Un vol a un aéroport de départ et un aéroport d'arrivée
  7. Un vol a un jour et une heure de départ et d'arrivée
  8. Un vol peut comporter des escales dans des aéroports
  9. Une escale a une heure d'arrivée et de départ
  10. Chaque aéroport dessert une ou plusieurs ville

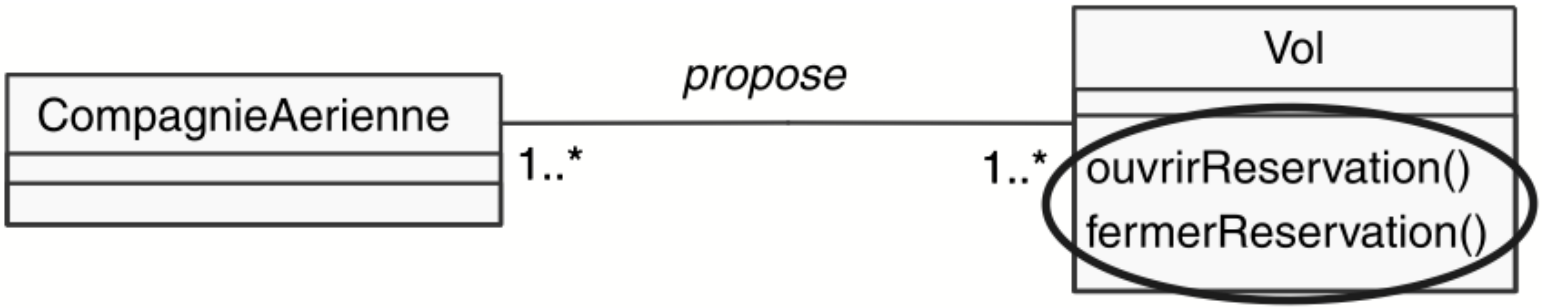
- Un vol est ouvert à la réservation et refermé sur ordre de la compagnie

1er solution : attribut état



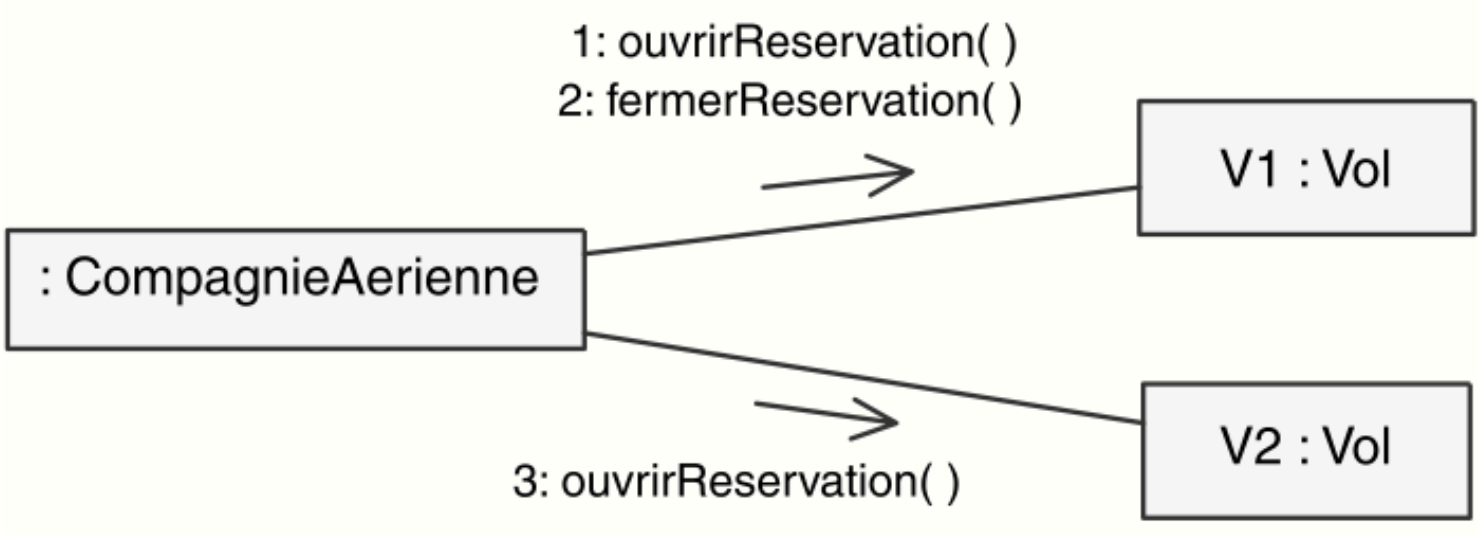
MAIS il s'agit d'un concept dynamique
→ à mettre avec les opérations

- Ouvrir et fermer réservation : responsabilité du vol, pas de la compagnie
- La compagnie enverra un message pour invoquer les opérations



Remarque

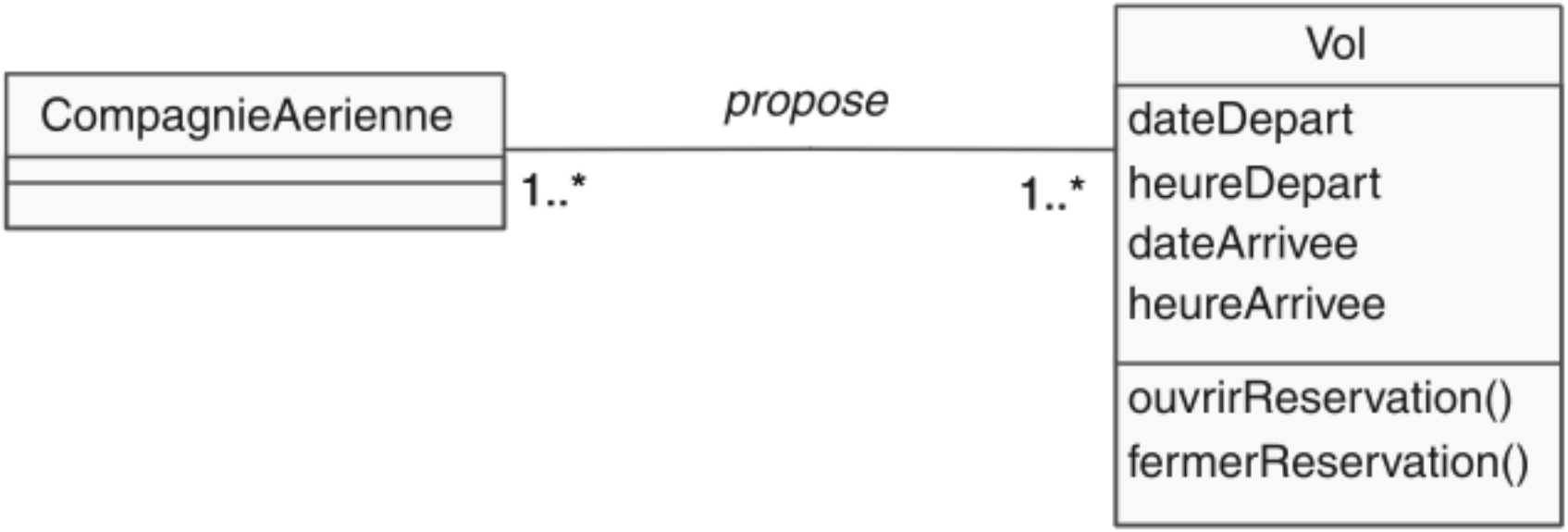
- Diagramme de communication : illustre les envois de messages entre instances



Exemple

- Système simplifié de réservation de vols (résumé des connaissances des experts du domaine)
  1. Les compagnies aériennes proposent différents vols
  2. Un vol est ouvert à la réservation et refermé sur ordre de la compagnie
  3. Un client peut réserver un ou plusieurs vols, pour des passagers différents
  4. Une réservation concerne un seul vol et un seul passager
  5. Une réservation peut être confirmée ou annulée
  6. Un vol a un aéroport de départ et un aéroport d'arrivée
  7. Un vol a un jour et une heure de départ et d'arrivée
  8. Un vol peut comporter des escales dans des aéroports
  9. Une escale a une heure d'arrivée et de départ
  10. Chaque aéroport dessert une ou plusieurs ville

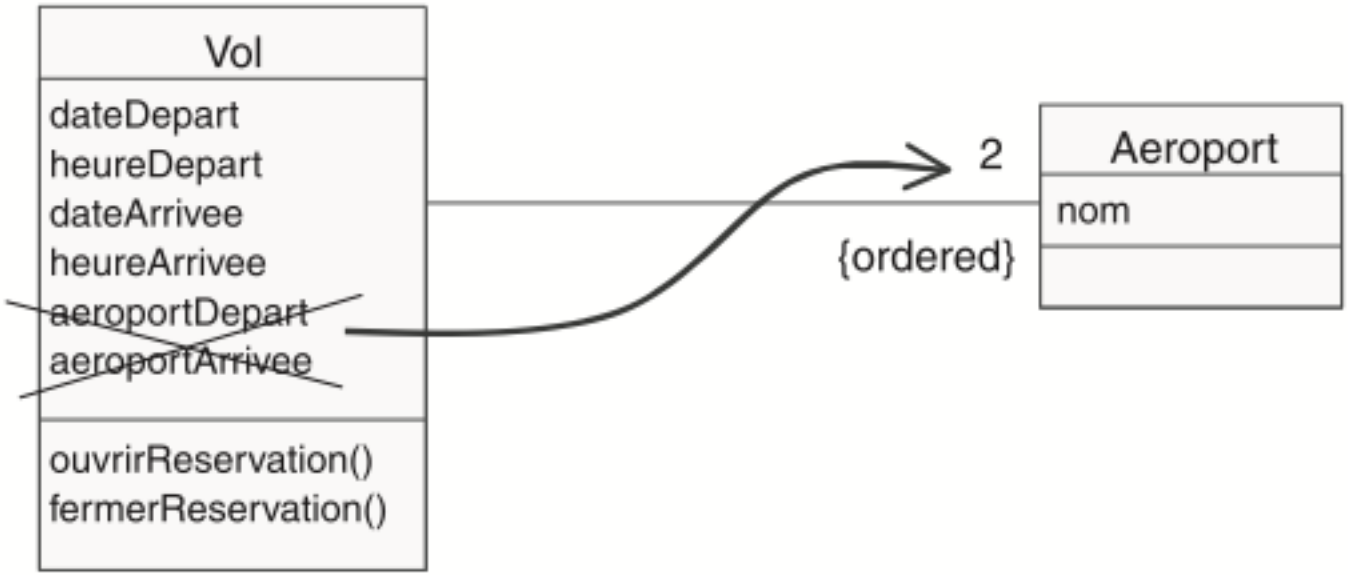
- 7) Un vol a un jour et une heure de départ et d'arrivée
  - Date et heures sont de simples valeurs : **attributs**



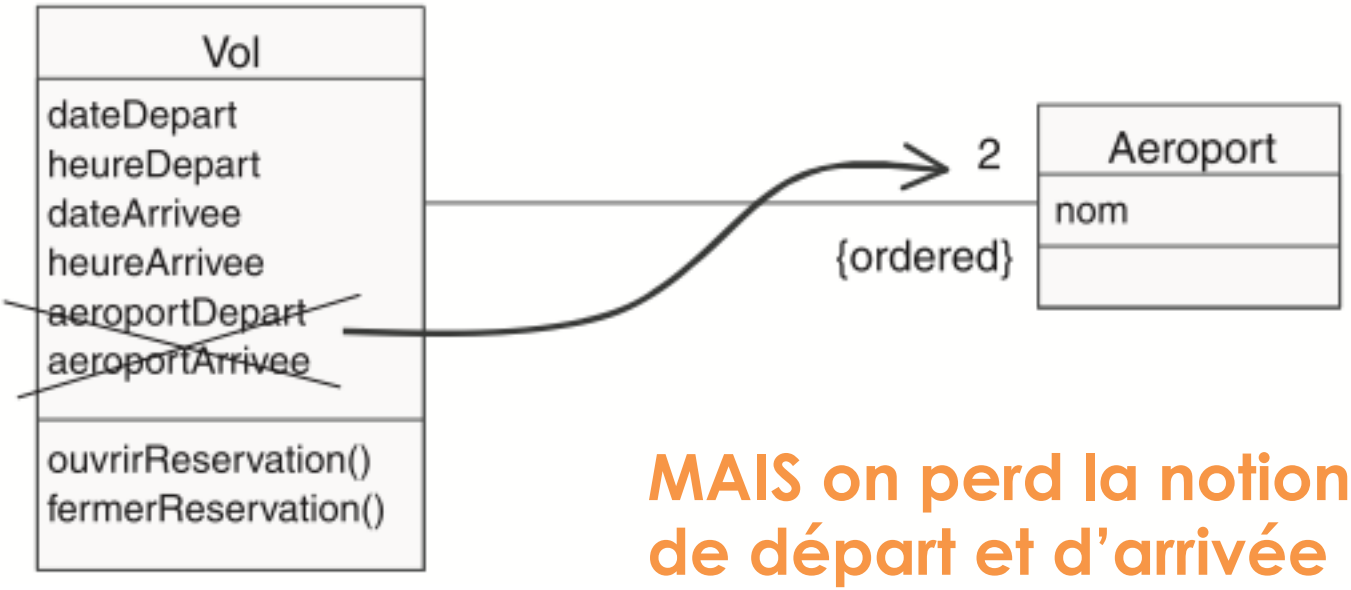
Exemple

- Système simplifié de réservation de vols (résumé des connaissances des experts du domaine)
  1. Les compagnies aériennes proposent différents vols
  2. Un vol est ouvert à la réservation et refermé sur ordre de la compagnie
  3. Un client peut réserver un ou plusieurs vols, pour des passagers différents
  4. Une réservation concerne un seul vol et un seul passager
  5. Une réservation peut être confirmée ou annulée
  6. Un vol a un aéroport de départ et un aéroport d'arrivée
  7. Un vol a un jour et une heure de départ et d'arrivée
  8. Un vol peut comporter des escales dans des aéroports
  9. Une escale a une heure d'arrivée et de départ
  10. Chaque aéroport dessert une ou plusieurs ville

- Pas un simple attribut : objet complexe (nom, capacité, villes desservies…)
  → nouvelle classe

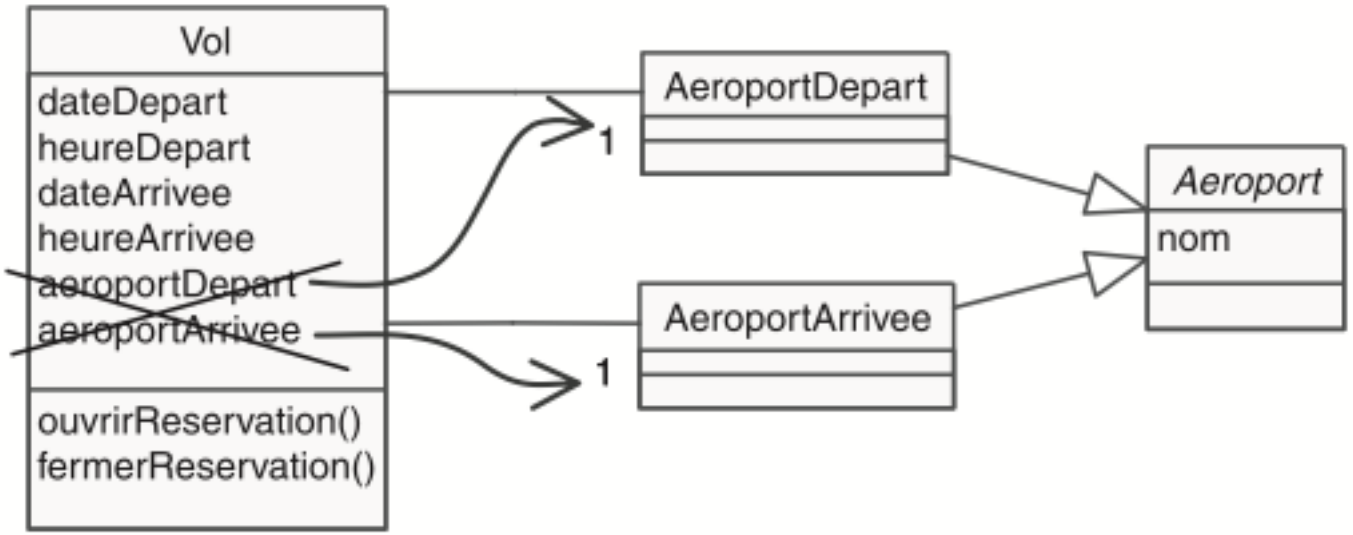


- Pas un simple attribut : objet complexe (nom, capacité, villes desservies…)
  → nouvelle classe



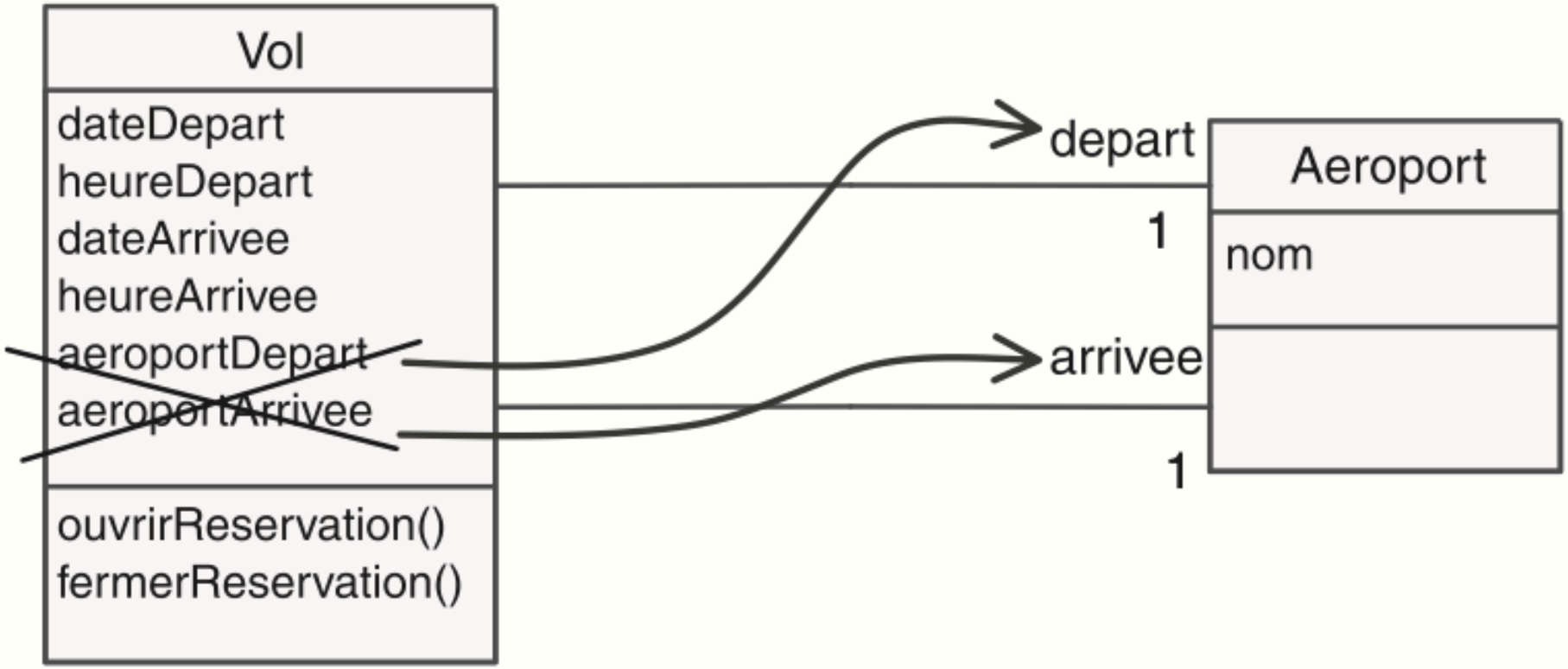
MAIS on perd la notion de départ et d'arrivée

- 2ème solution : création de 2 sous classes



MAIS un aéroport peut être à la fois AeroportDepart et AeroportArrivé
→ REDONDANCE

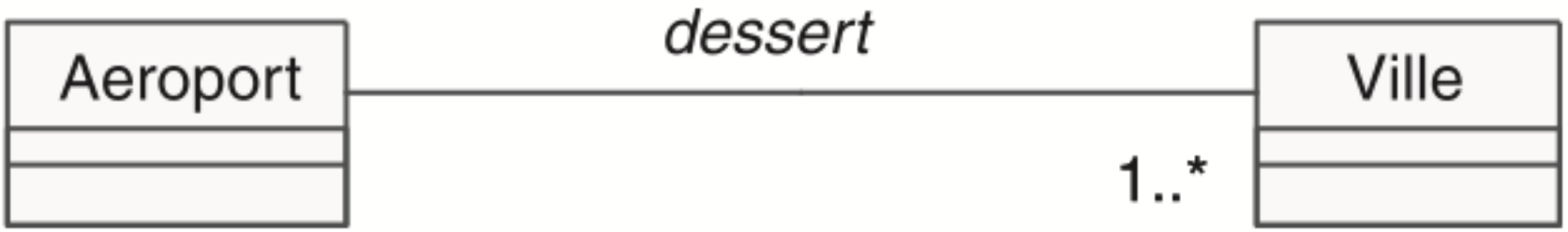
- 3ème solution : utilisation des rôles



Exemple

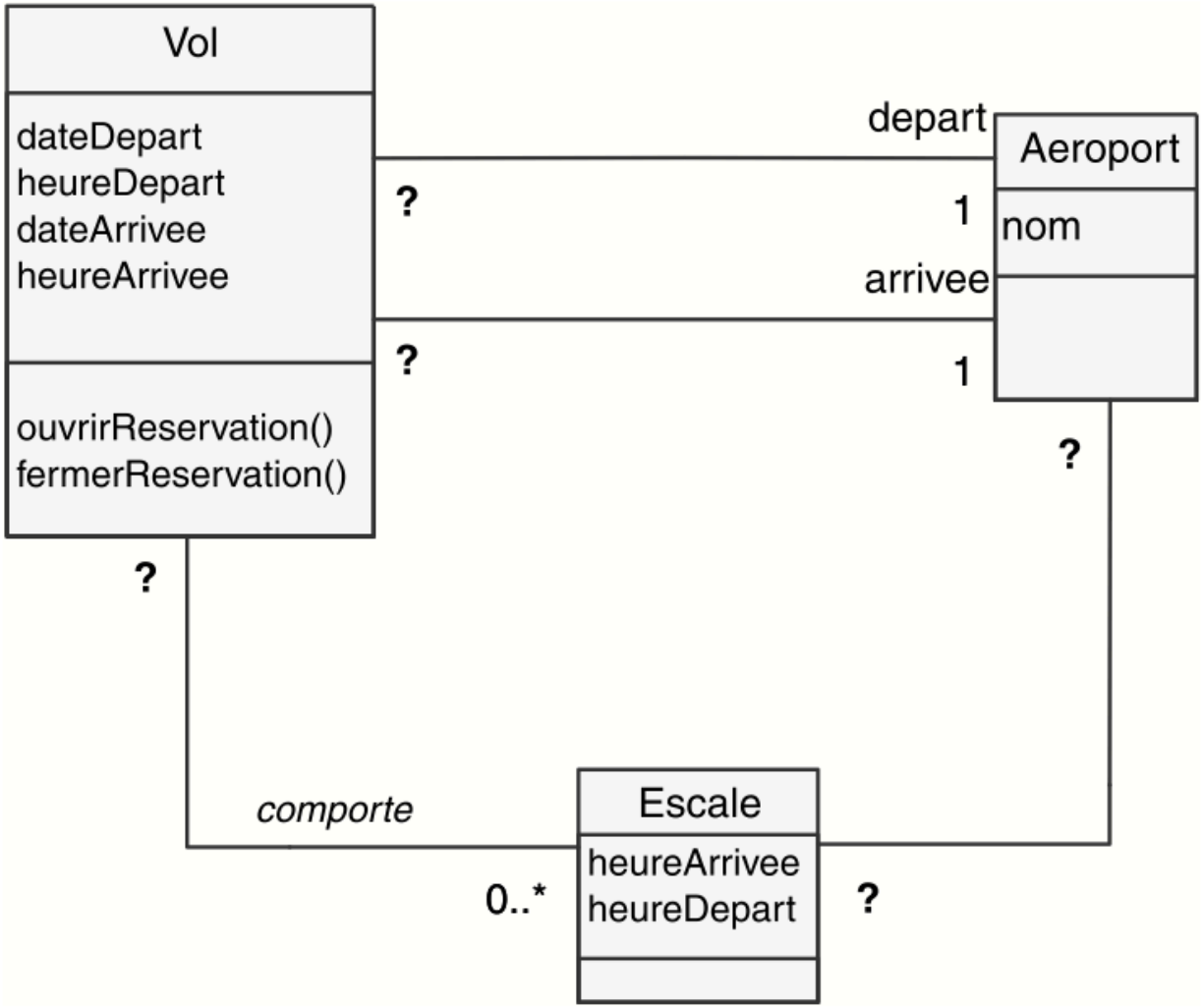
- Système simplifié de réservation de vols (résumé des connaissances des experts du domaine)
  1. Les compagnies aériennes proposent différents vols
  2. Un vol est ouvert à la réservation et refermé sur ordre de la compagnie
  3. Un client peut réserver un ou plusieurs vols, pour des passagers différents
  4. Une réservation concerne un seul vol et un seul passager
  5. Une réservation peut être confirmée ou annulée
  6. Un vol a un aéroport de départ et un aéroport d'arrivée
  7. Un vol a un jour et une heure de départ et d'arrivée
  8. Un vol peut comporter des escales dans des aéroports
  9. Une escale a une heure d'arrivée et de départ
  10. Chaque aéroport dessert une ou plusieurs ville

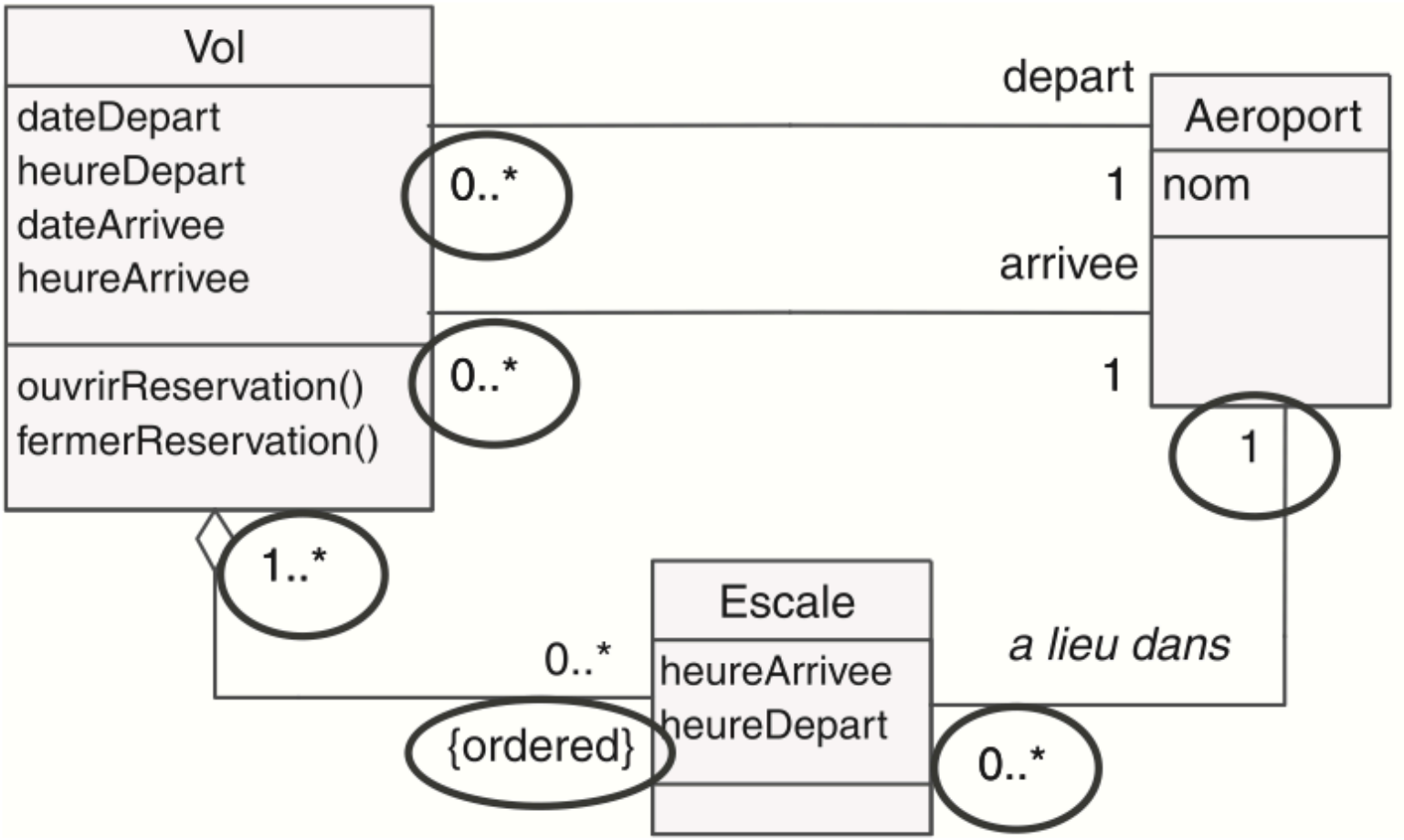
- Détermine la multiplicité côté Ville



Exemple

- Système simplifié de réservation de vols (résumé des connaissances des experts du domaine)
  1. Les compagnies aériennes proposent différents vols
  2. Un vol est ouvert à la réservation et refermé sur ordre de la compagnie
  3. Un client peut réserver un ou plusieurs vols, pour des passagers différents
  4. Une réservation concerne un seul vol et un seul passager
  5. Une réservation peut être confirmée ou annulée
  6. Un vol a un aéroport de départ et un aéroport d'arrivée
  7. Un vol a un jour et une heure de départ et d'arrivée
  8. Un vol peut comporter des escales dans des aéroports
  9. Une escale a une heure d'arrivée et de départ
  10. Chaque aéroport dessert une ou plusieurs ville


Vol
dateDepart
heureDepart
dateArrivee
heureArrivee
ouvrirReservation()
fermerReservation()
depart
?
1
Aeroport
nom
arrivee
?
1
?
?
comporte
Escale
heureArrivee
heureDepart
0..*
?


Vol
dateDepart
heureDepart
dateArrivee
heureArrivee
ouvrirReservation()
fermerReservation()
depart
arrivee
0..*
0..*
1
1
Aeroport
nom
1
1..*
Escale
heureArrivee
heureDepart
0..*
{ordered}
a lieu dans
0..*

Peut-on améliorer le diagramme ?

UPMC SORBONNE UNIVERSITÉS

- Escale est un intermédiaire entre Vol et Aéroport
- Peut-on la modéliser différemment ?

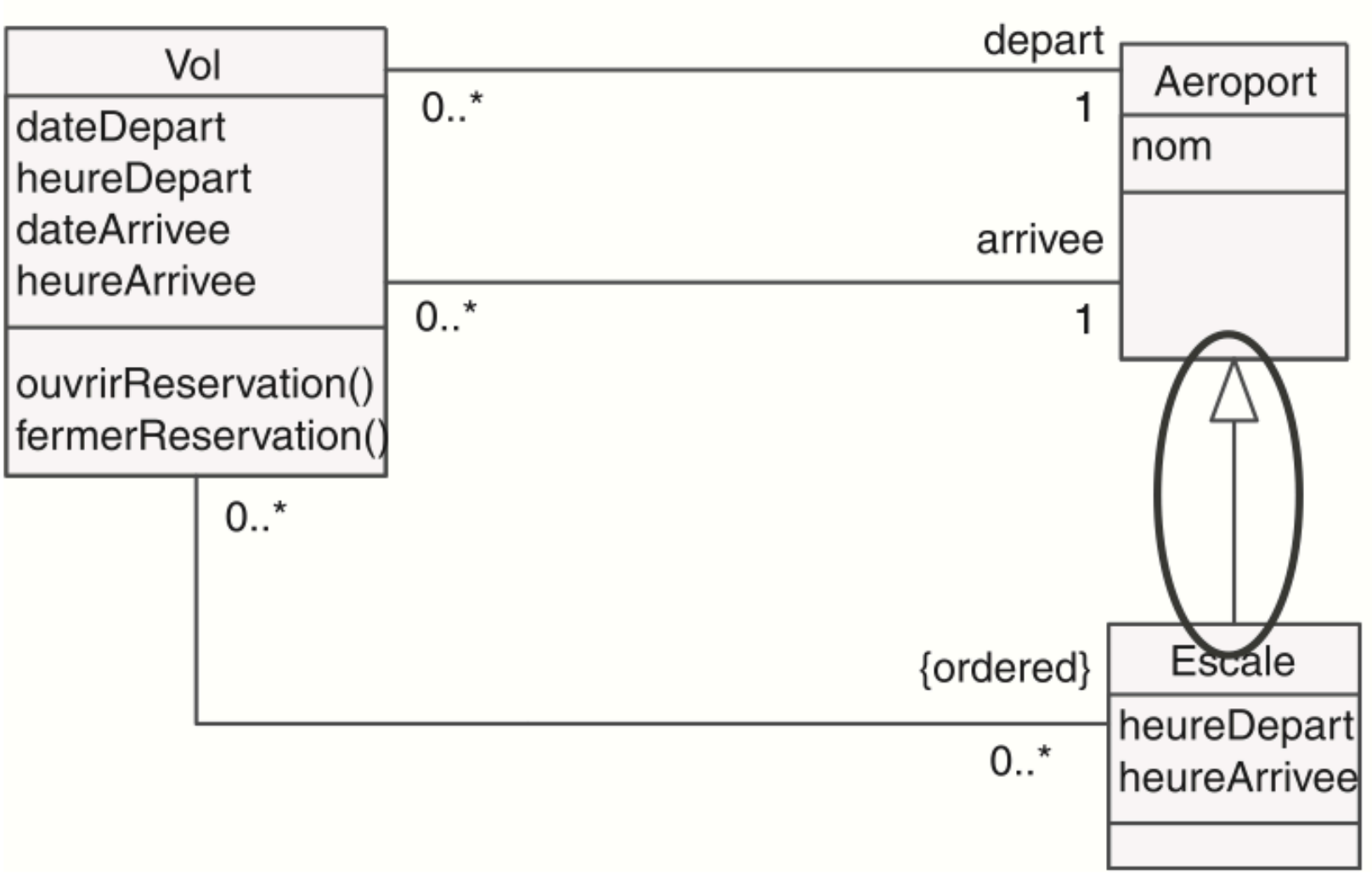
- Solution 1 : héritage

Peut-on améliorer le diagramme ?

- Escale est un intermédiaire entre Vol et Aéroport
- Peut-on la modéliser différemment ?

- Solution 1 : héritage
  - Une escale est un cas particulier d'aéroport

Héritage


Vol
dateDepart
heureDepart
dateArrivee
heureArrivee
ouvrirReservation()
fermerReservation()
depart
0..*
1
Aeroport
nom
arrivee
0..*
1
0..*
{ordered}
Escale
heureDepart
heureArrivee
0..*

Problèmes ?

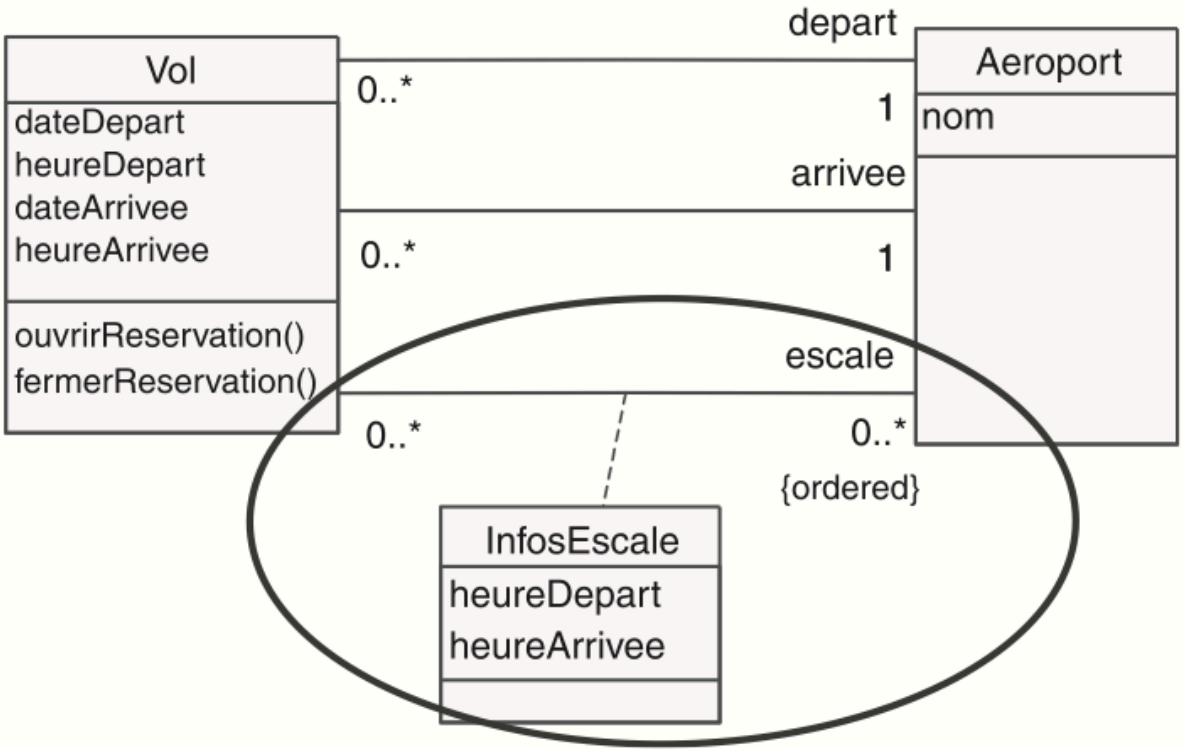
UPMC SORBONNE UNIVERSITÉS

- Une escale n'est pas un aéroport !
  - Une escale ne dessert pas de villes
  - Une escale ne peut pas servir de départ ni d'arrivée
- Juste une facilité d'implémentation pour récupérer les informations d'aéroport
  - Héritage d'implémentation
  - A NE PAS FAIRE

Classes d'associations

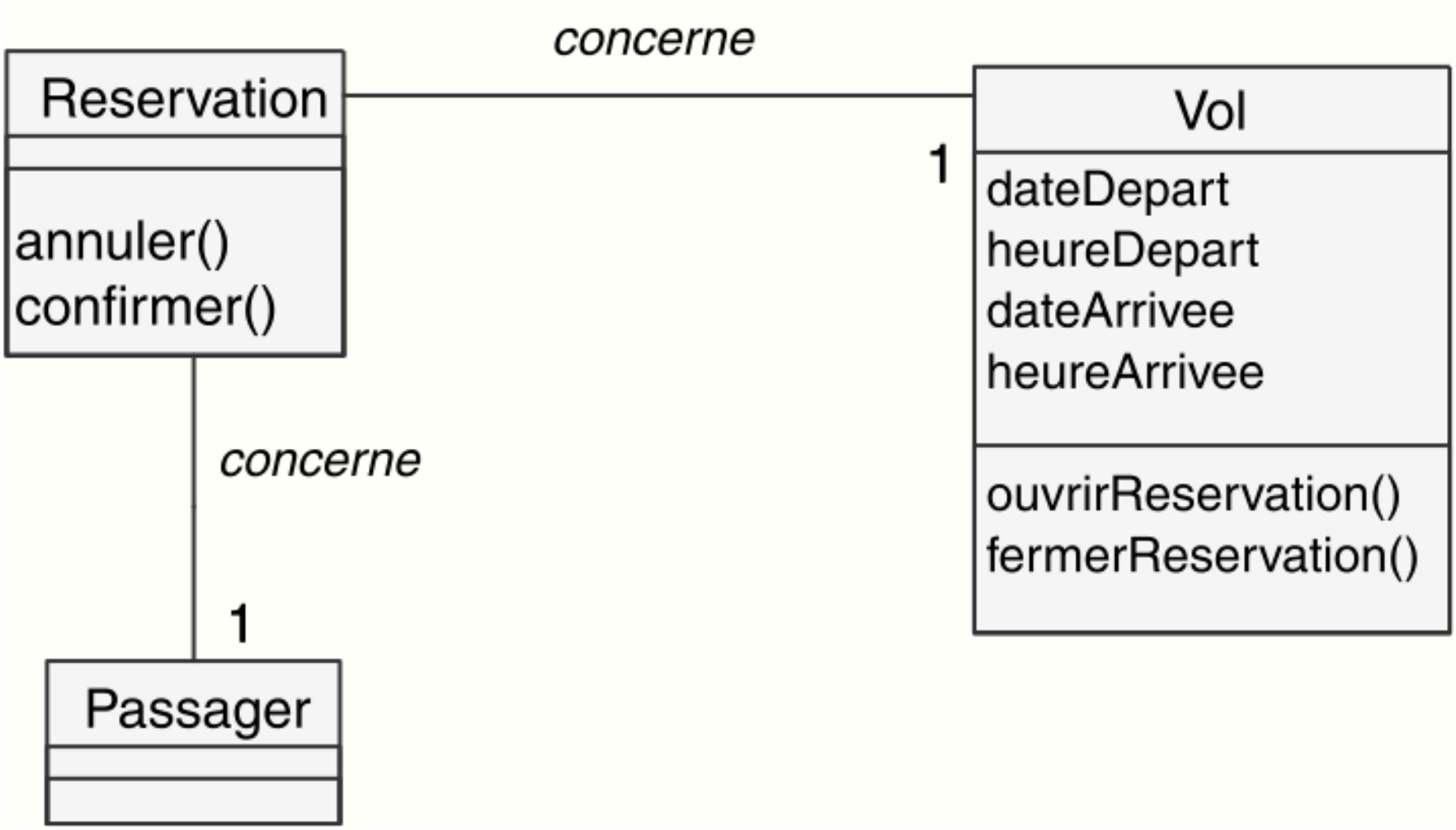
- 2ème solution : l'escale est un **rôle joué** par l'aéroport :
  - heureArrivee et heureDepart sont des **attributs d'association**

On pourrait le faire aussi pour les heures et dates de départ et d'arrivée



Exemple

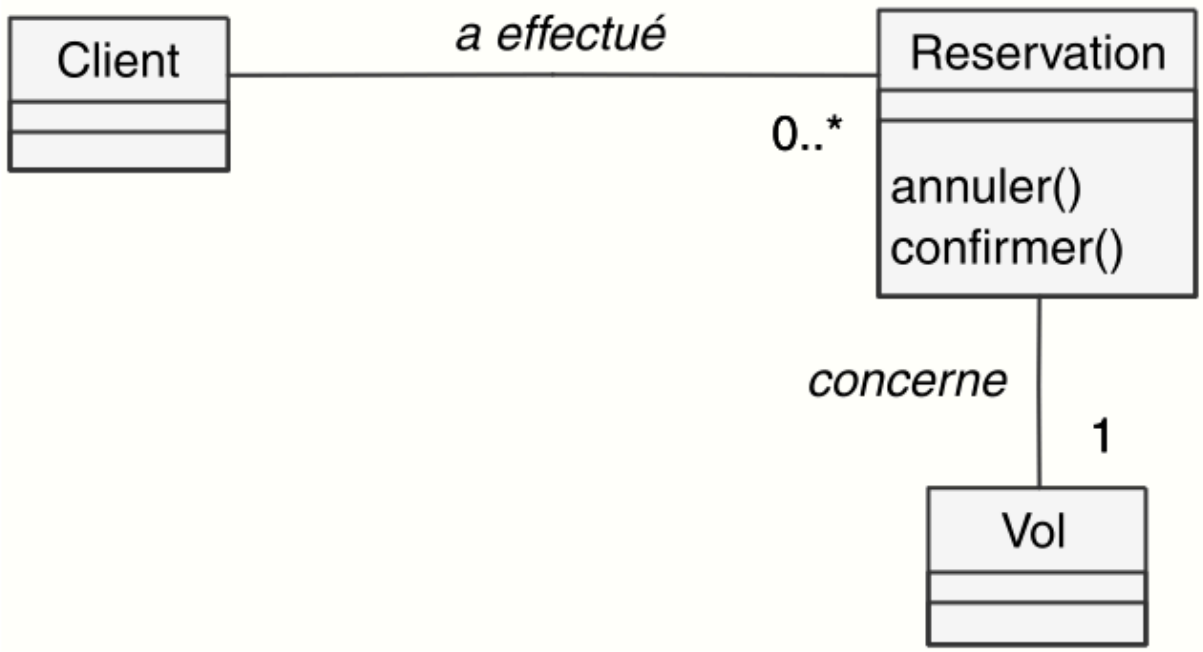
- Système simplifié de réservation de vols (résumé des connaissances des experts du domaine)
  1. Les compagnies aériennes proposent différents vols
  2. Un vol est ouvert à la réservation et refermé sur ordre de la compagnie
  3. Un client peut réserver un ou plusieurs vols, pour des passagers différents
  4. Une réservation concerne un seul vol et un seul passager
  5. Une réservation peut être confirmée ou annulée
  6. Un vol a un aéroport de départ et un aéroport d'arrivée
  7. Un vol a un jour et une heure de départ et d'arrivée
  8. Un vol peut comporter des escales dans des aéroports
  9. Une escale a une heure d'arrivée et de départ
  10. Chaque aéroport dessert une ou plusieurs ville


concerne
Reservation
annuler()
confirmer()
Vol
dateDepart
heureDepart
dateArrivee
heureArrivee
ouvrirReservation()
fermerReservation()
1
concerne
1
Passager

Exemple

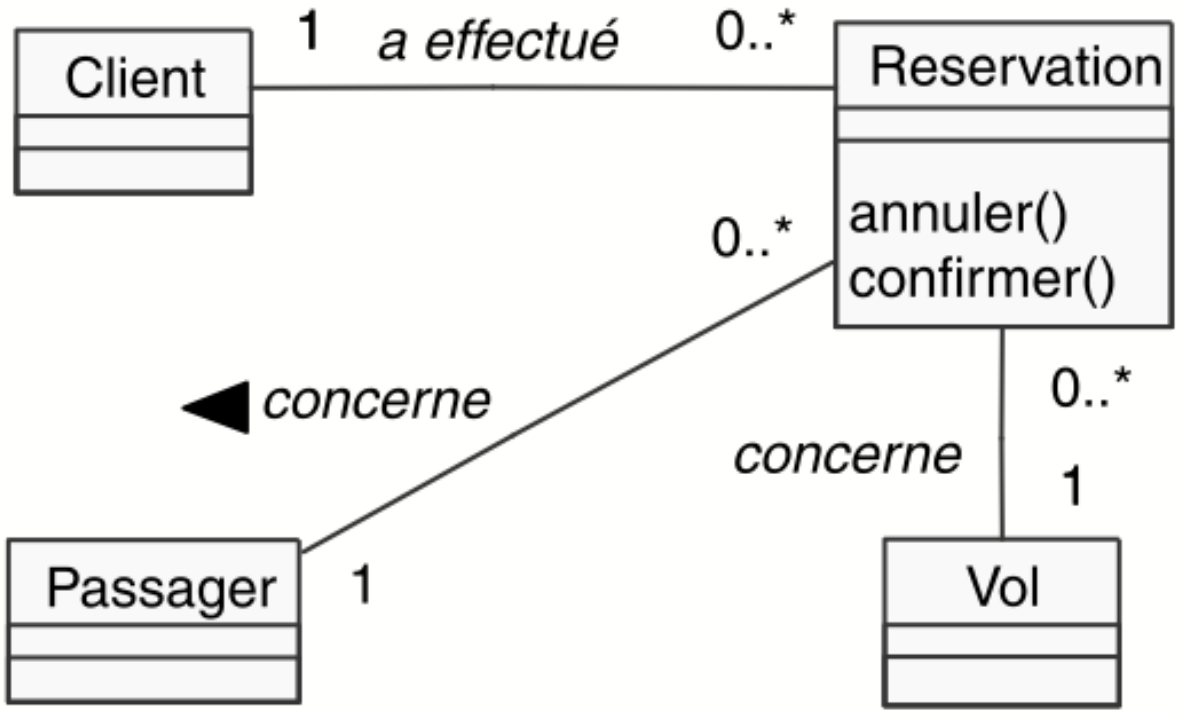
- Système simplifié de réservation de vols (résumé des connaissances des experts du domaine)
  1. Les compagnies aériennes proposent différents vols
  2. Un vol est ouvert à la réservation et refermé sur ordre de la compagnie
  3. Un client peut réserver un ou plusieurs vols, pour des passagers différents
  4. Une réservation concerne un seul vol et un seul passager
  5. Une réservation peut être confirmée ou annulée
  6. Un vol a un aéroport de départ et un aéroport d'arrivée
  7. Un vol a un jour et une heure de départ et d'arrivée
  8. Un vol peut comporter des escales dans des aéroports
  9. Une escale a une heure d'arrivée et de départ
  10. Chaque aéroport dessert une ou plusieurs ville

On réutilise le concept de réservation et on reformule l'étape 3 par :
« Un client peut effectuer plusieurs réservations »



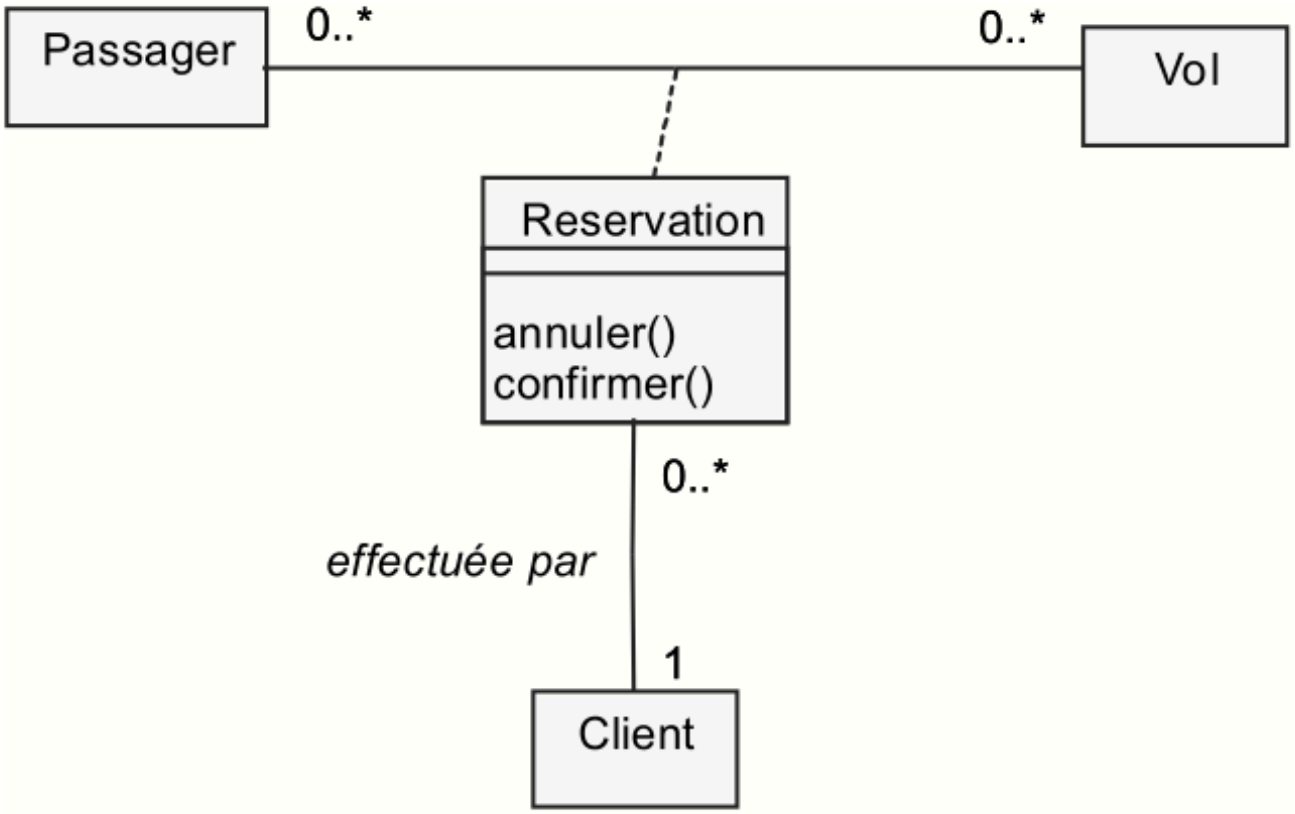
Exercices

- Proposer une modification du diagramme faisant intervenir une classe d'association

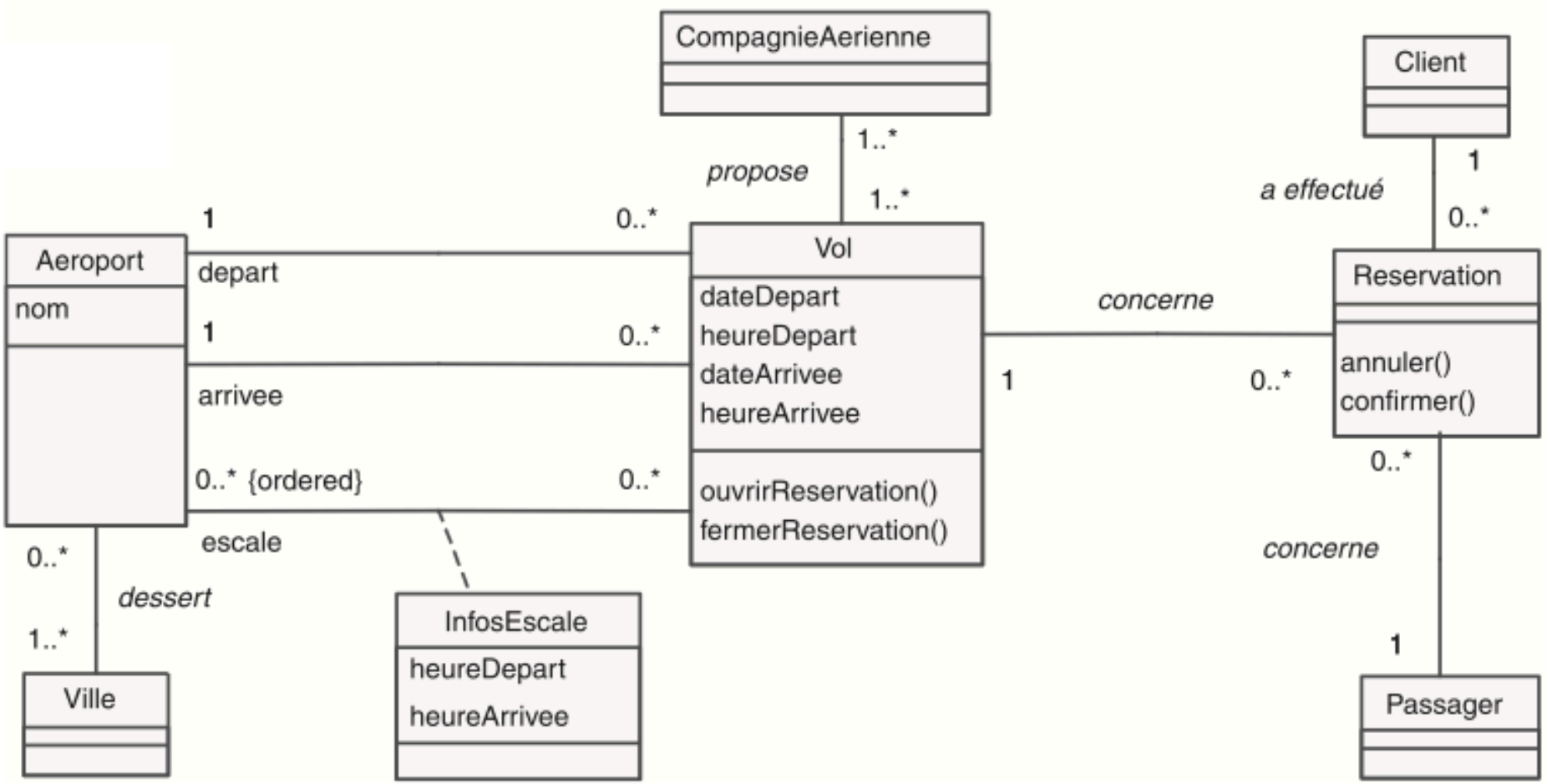


Solution

- On peut modéliser Reservation comme une classe d'association entre Vol et Passager



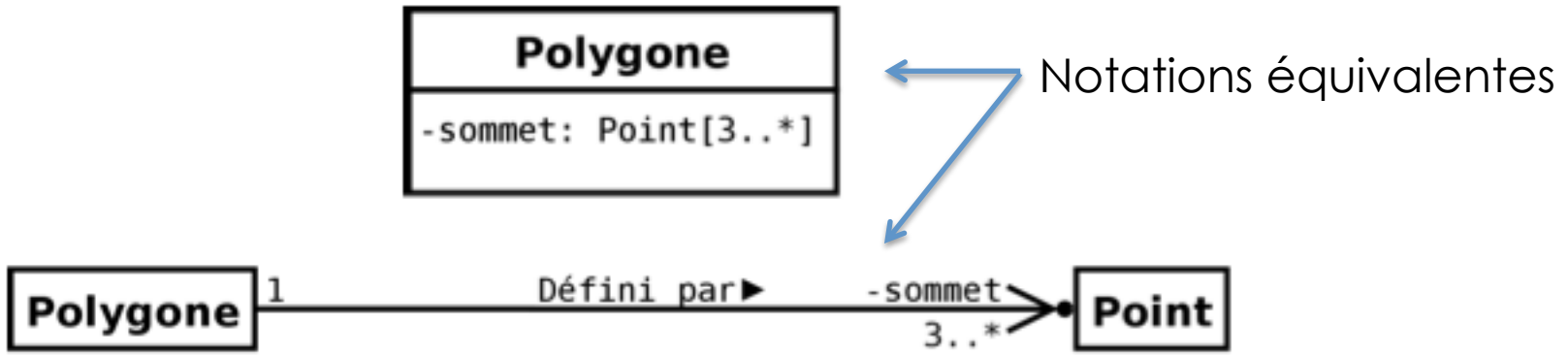
Résultat intermédiaire

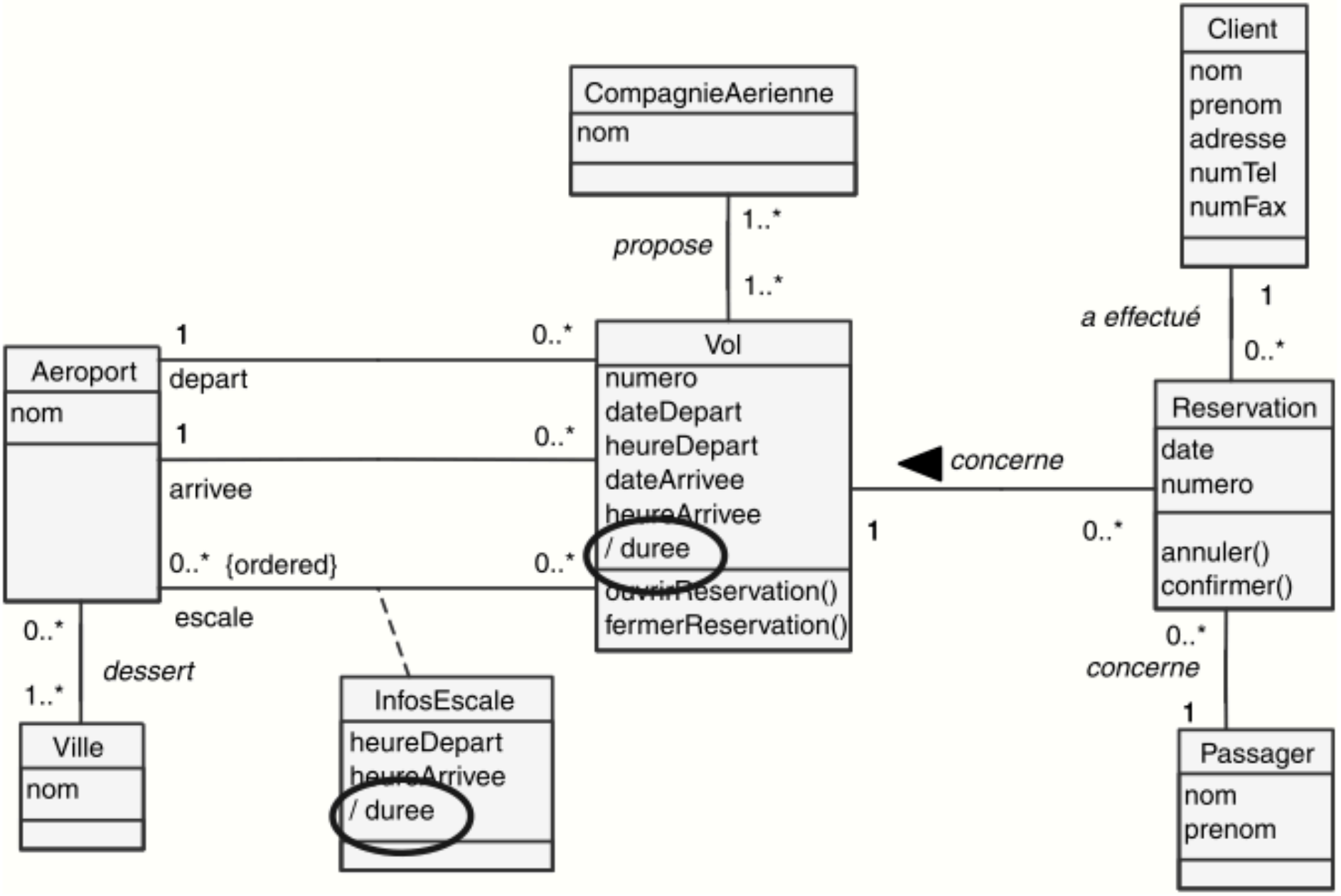


Architecture globale correcte, mais il manque certaines informations (attributs métiers, contraintes…)

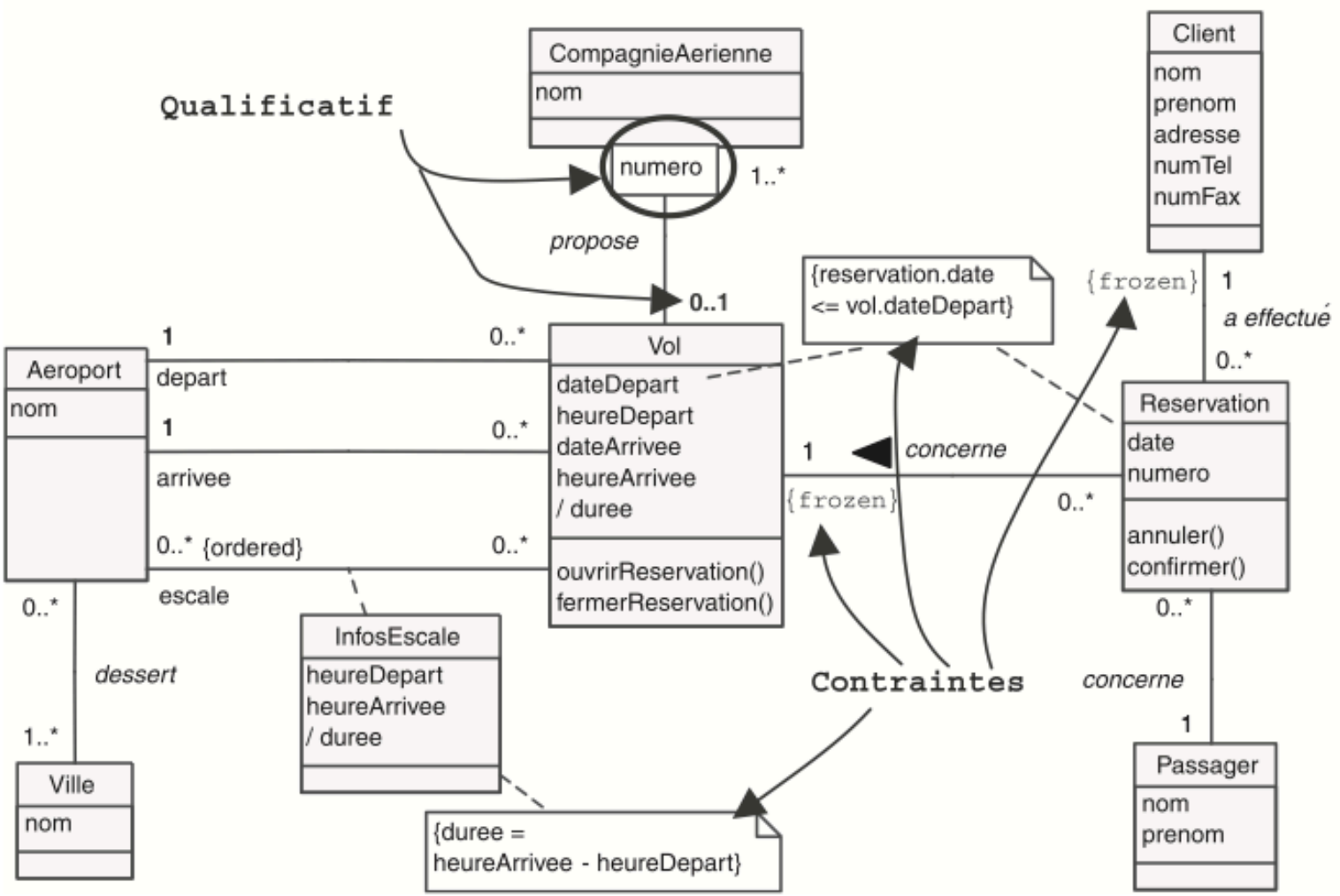
Les attributs

- Les attributs métiers
  - Définissent des informations qu'une classe ou un objet doivent connaître (ex pour Vol : numero, heureDepart…)
  - Pas de références à d'autres classes (rôle des associations)
- Les attributs dérivés
  - Peuvent être calculés à partir d'autres informations du modèle (ex pour un vol : durée)

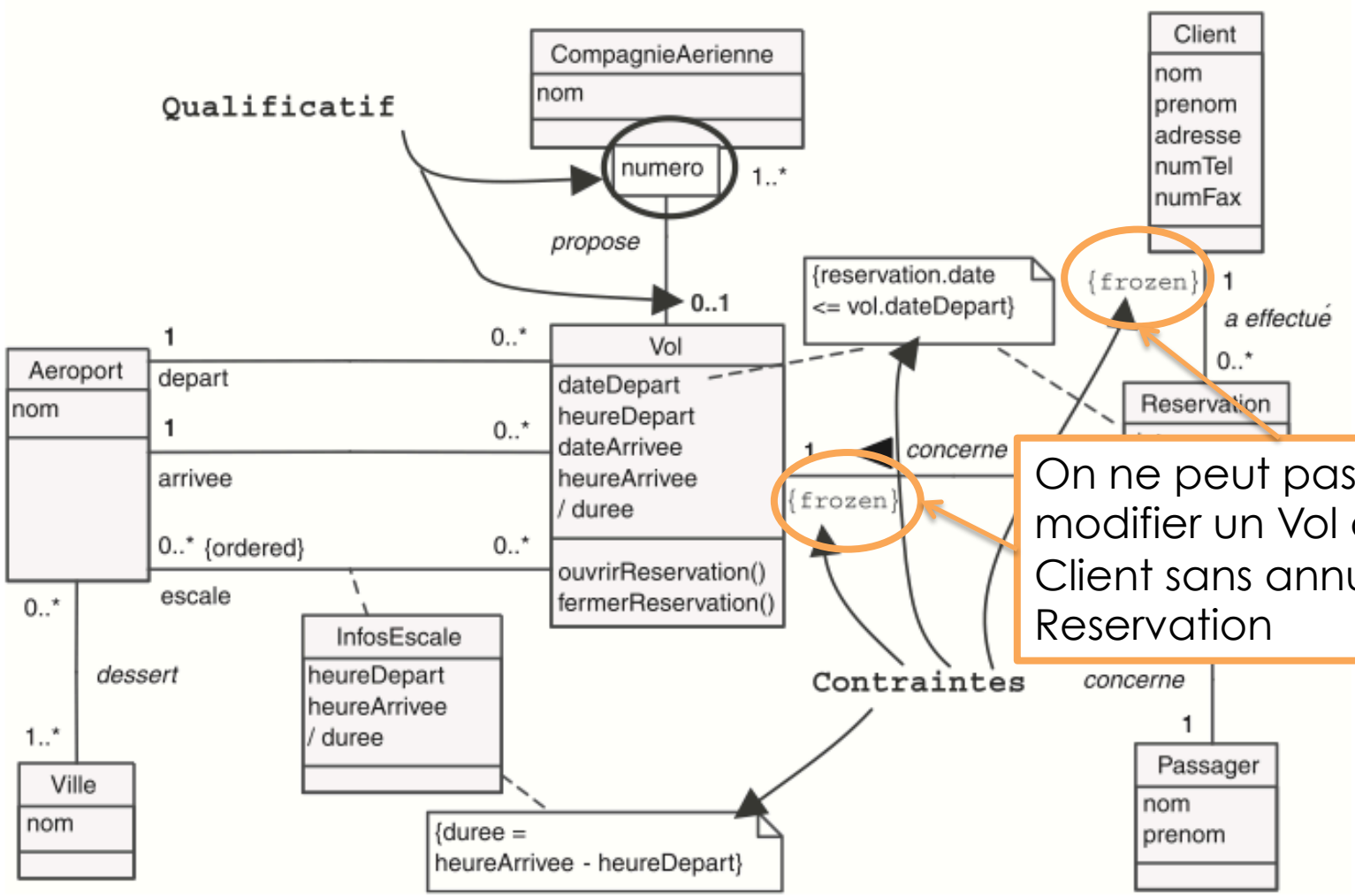



CompagnieAerienne
nom
propose
1..*
1..*
Client
nom
prenom
adresse
numTel
numFax
a effectué
1
0..*
Aeroport
nom
1
depart
0..*
1
arrivee
0..*
0..*
{ordered}
escale
Vol
numero
dateDepart
heureDepart
dateArrivee
heureArrivee
/ duree
ouvrirReservation()
fermerReservation()
concerne
1
0..*
Reservation
date
numero
annuler()
confirmer()
0..*
concerne
1
Passager
nom
prenom
0..*
dessert
1..*
Ville
nom
InfosEscale
heureDepart
heureArrivee
/ duree

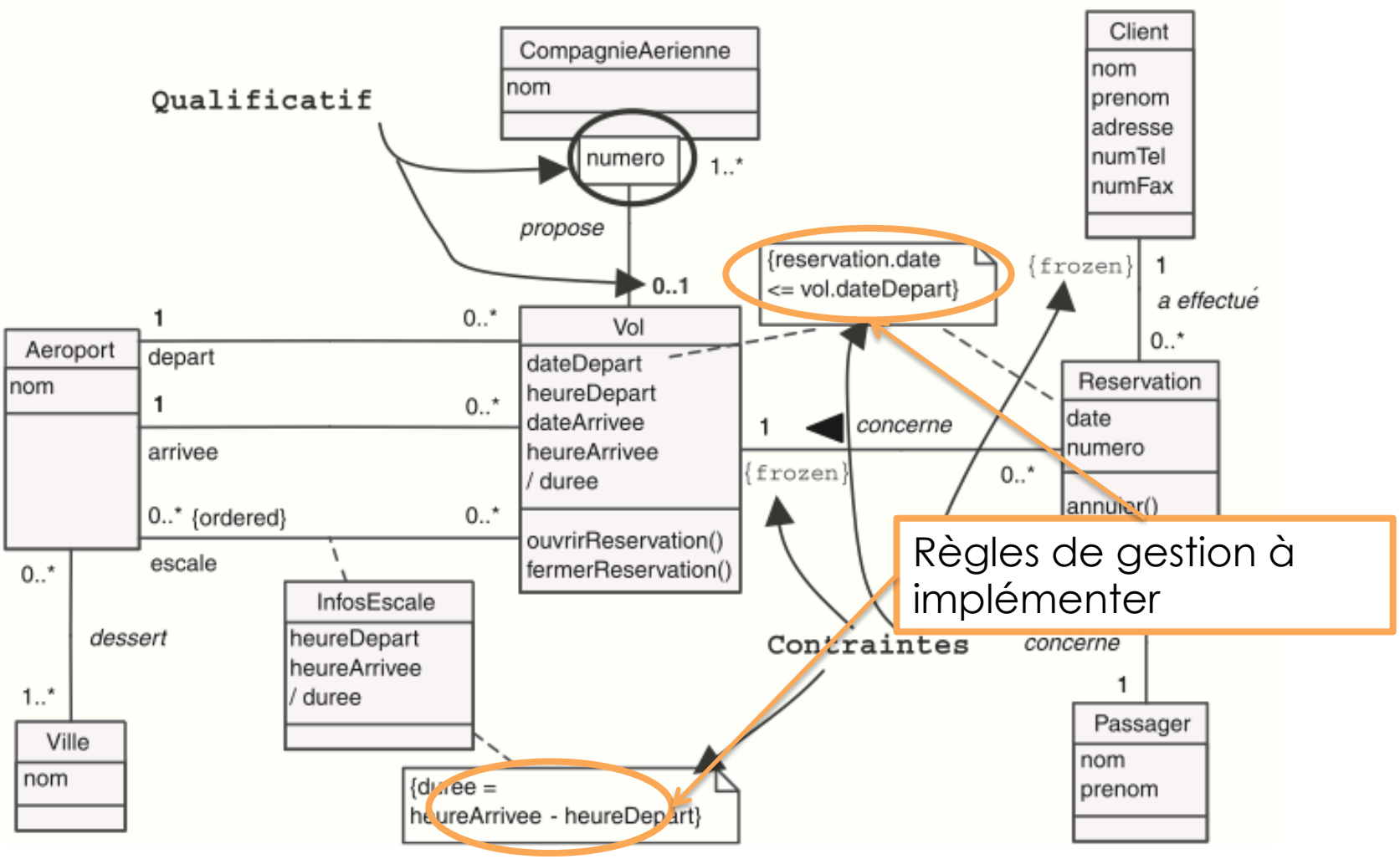
Contraintes et qualificatifs



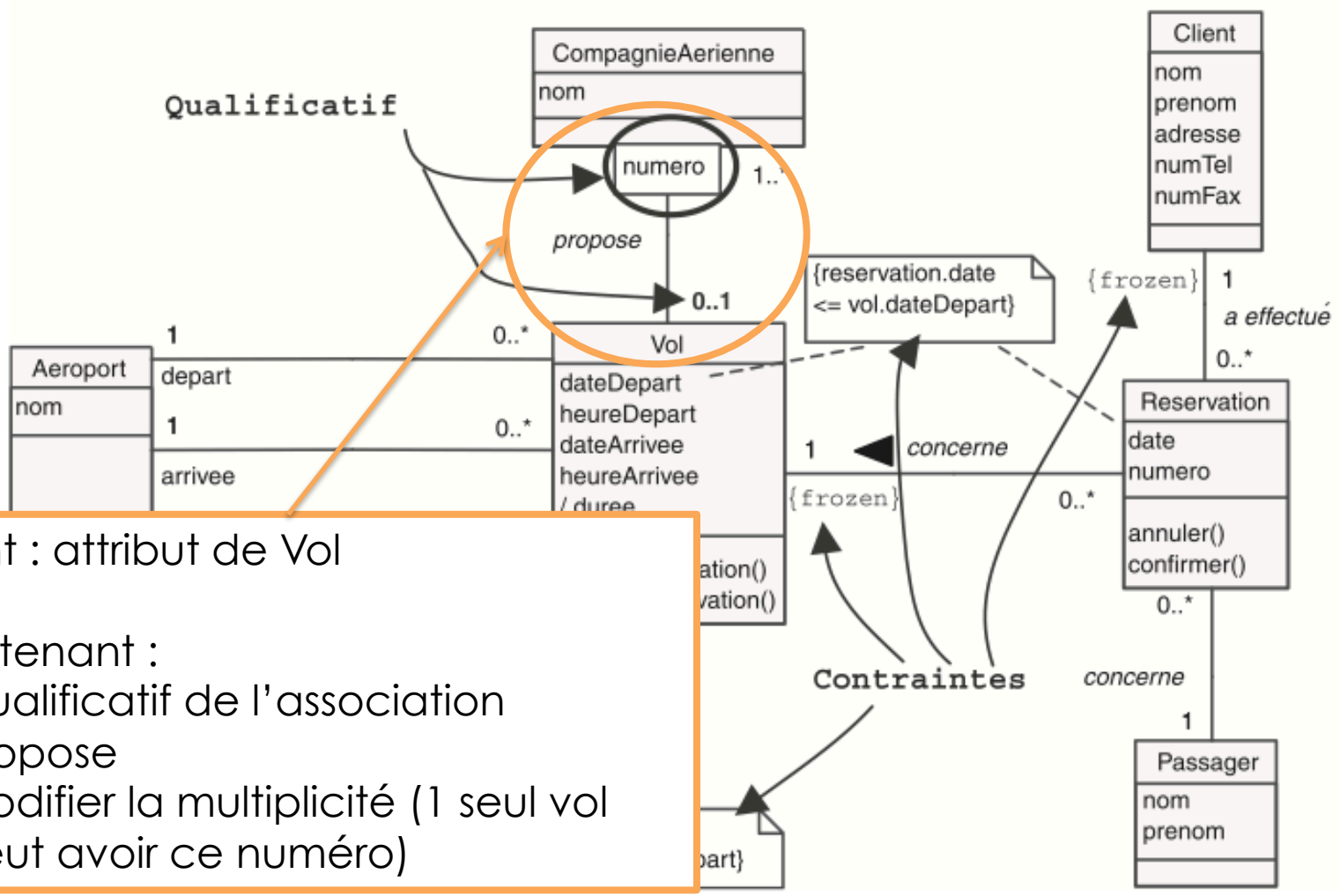
Contraintes et qualificatifs



Contraintes et qualificatifs



Contraintes et qualificatifs



Avant : attribut de Vol

Maintenant :

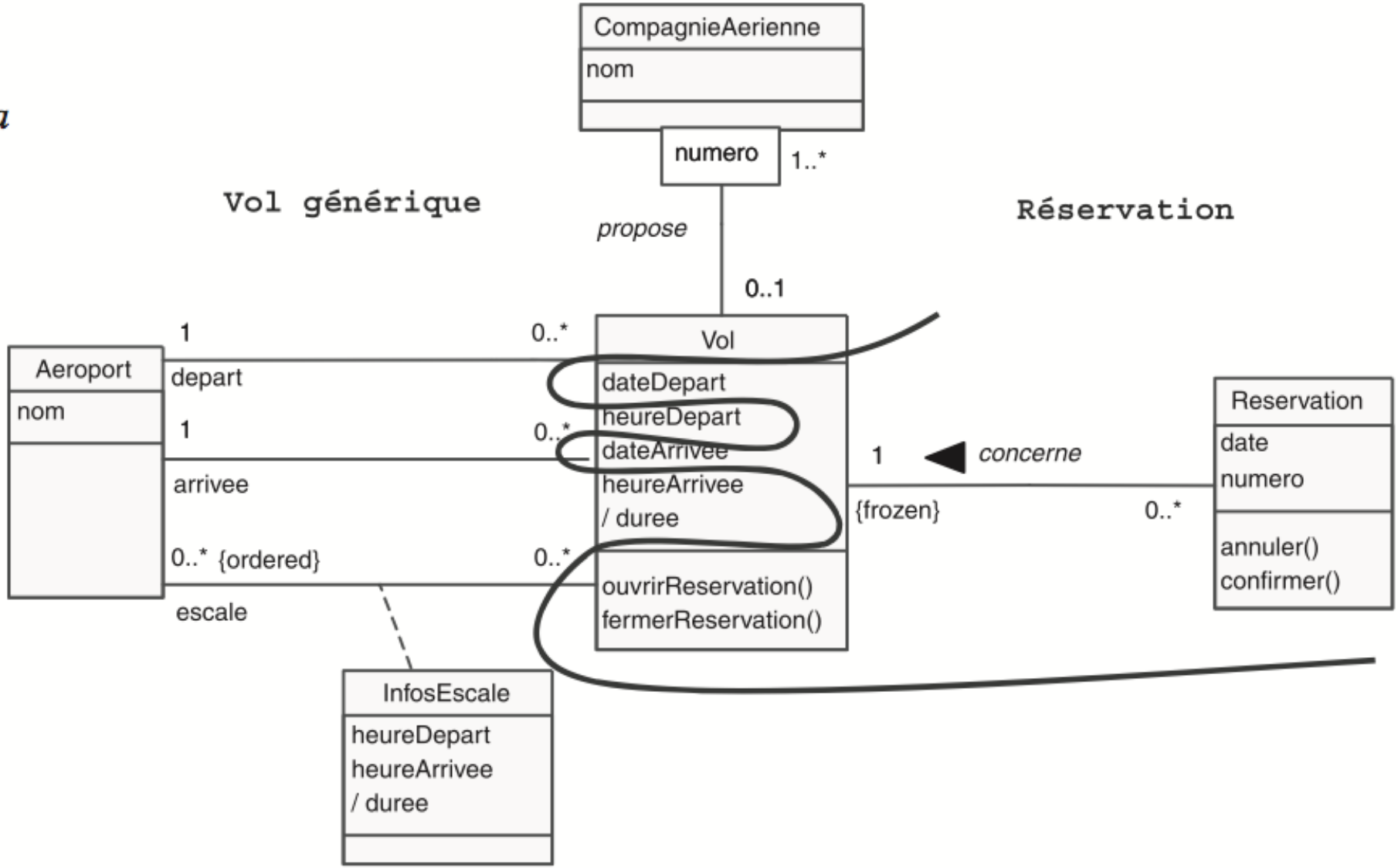
- Qualificatif de l'association propose
- Modifier la multiplicité (1 seul vol peut avoir ce numéro)

Peut-on encore améliorer le modèle ?

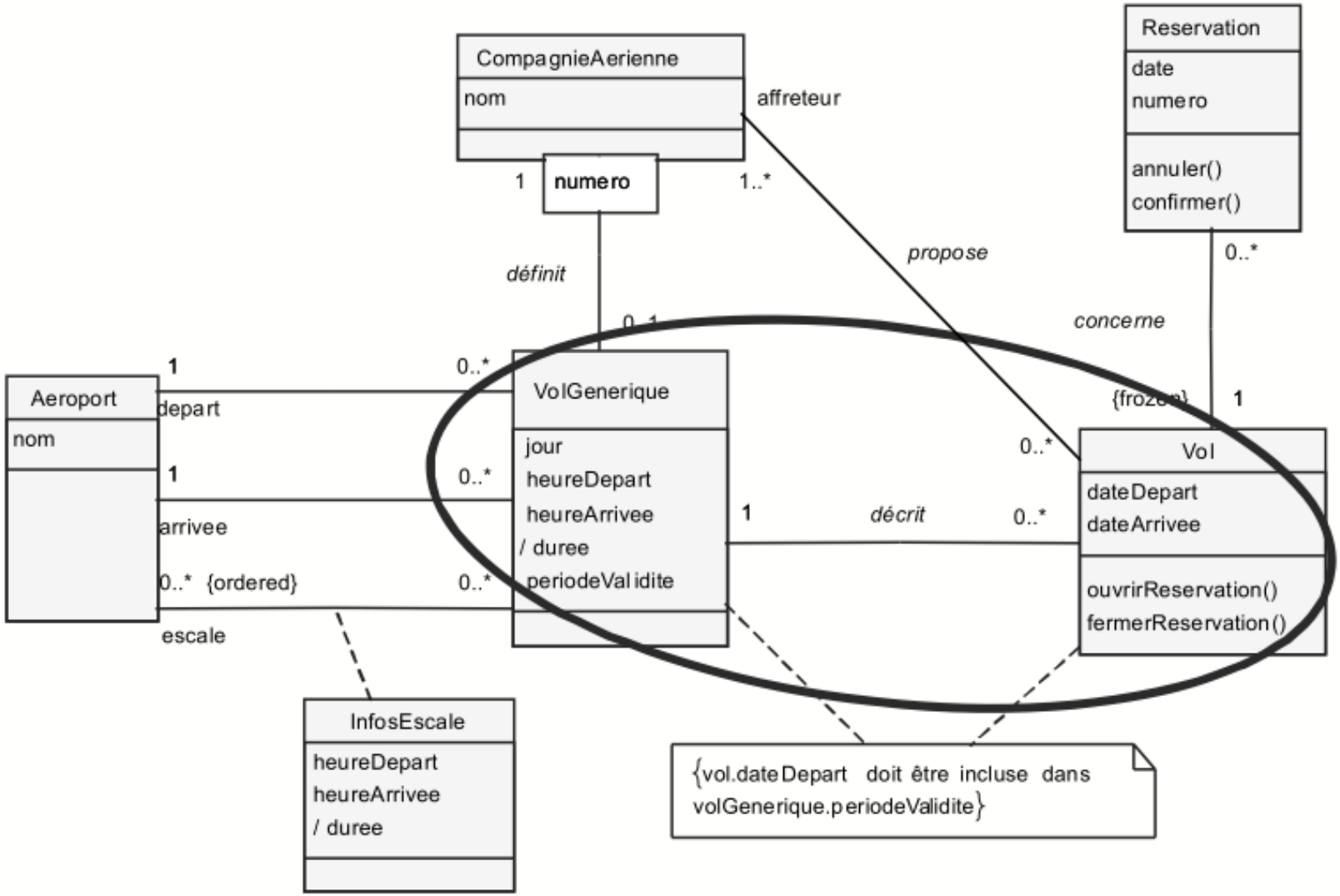
- La classe Vol a de nombreuses responsabilités (attributs et associations)

- A l'encontre de l'un des objectif de conception en POO : **forte cohérence**

Solution : pattern de la métaclasse



Solution : pattern de la métaclasse

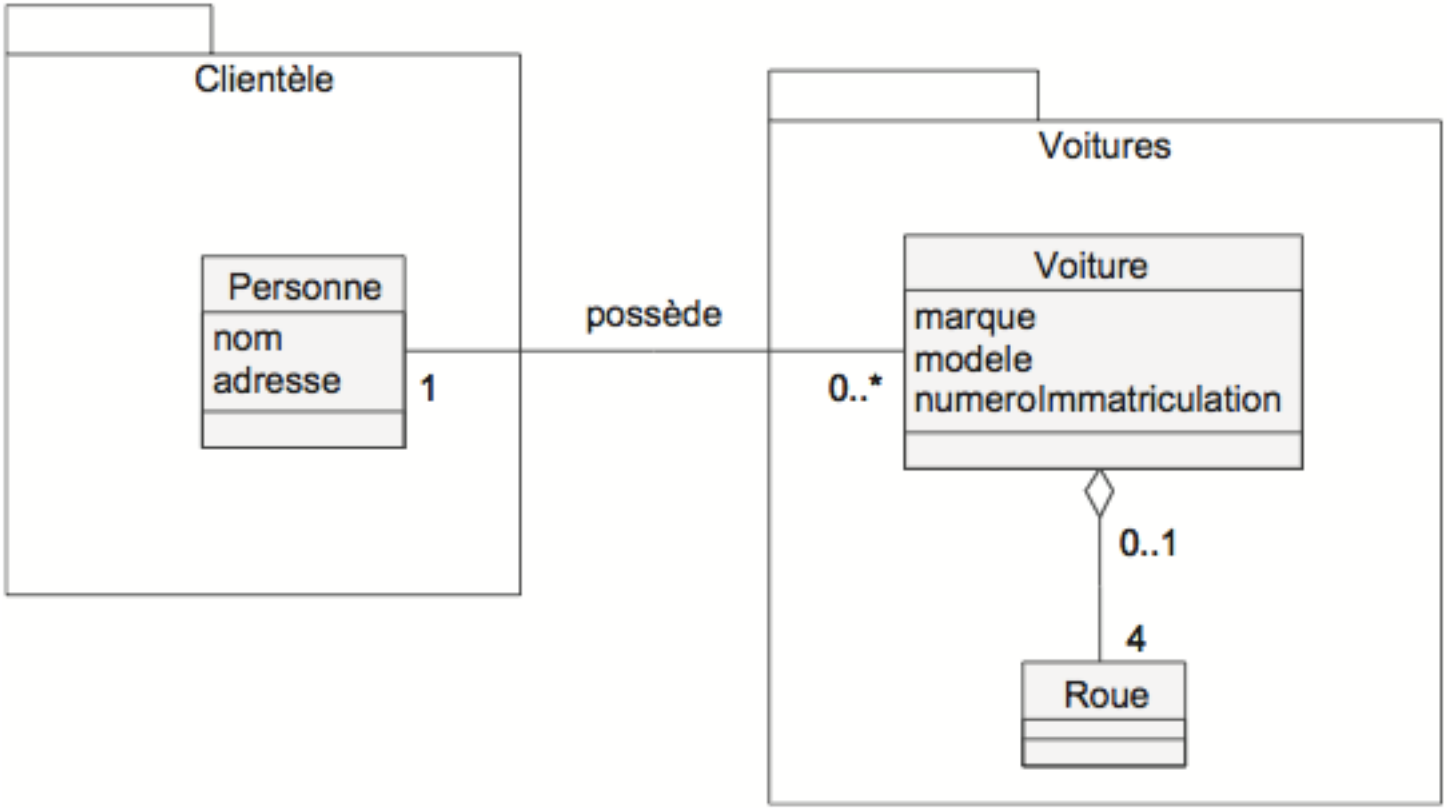


Package

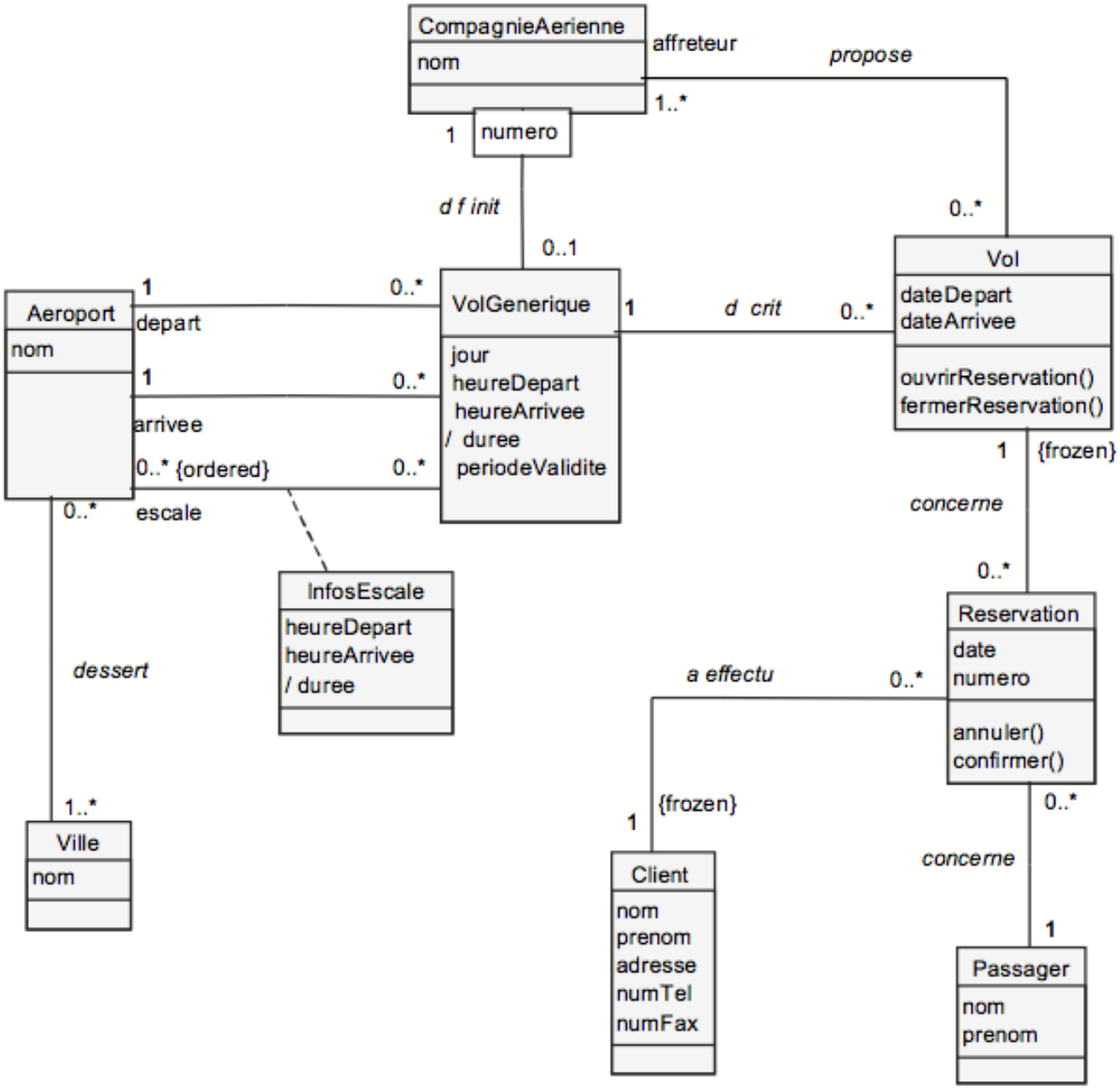
UPMC SORBONNE UNIVERSITÉS

- Permet de regrouper des classes et des relations
- Les packages sont des espaces de noms
- La structure en package doit se faire suivant deux principes :
  - **Cohérence** : forte proximité sémantique entre les classes d'un même package
  - **Indépendance** : faibles relations entre les classes de packages différents

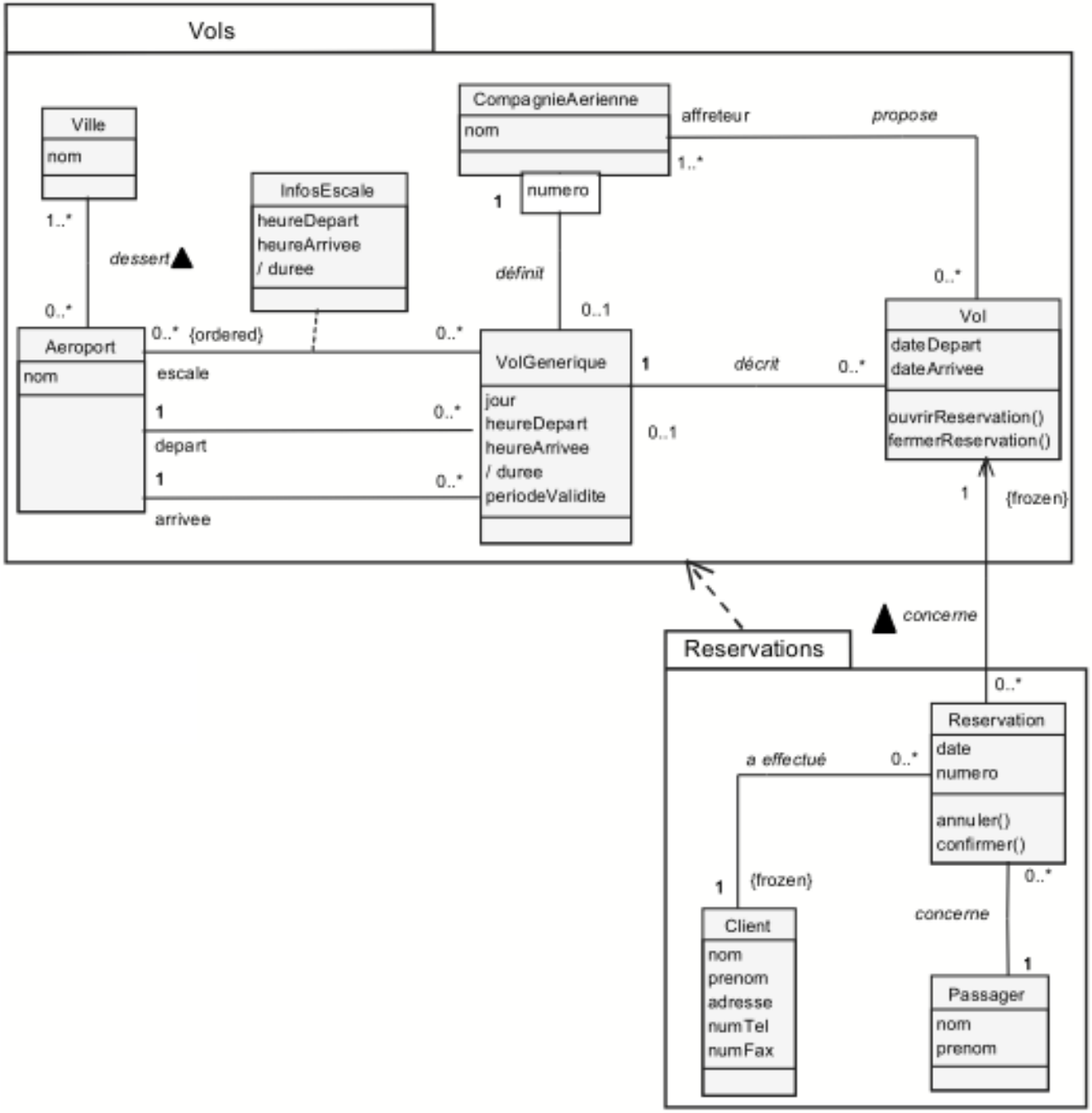
Représentation



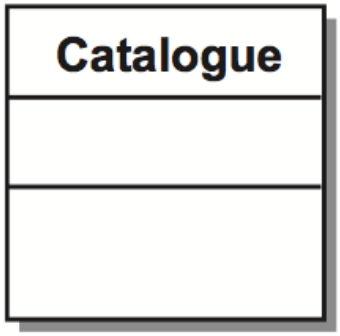
Proposer un découpage en package



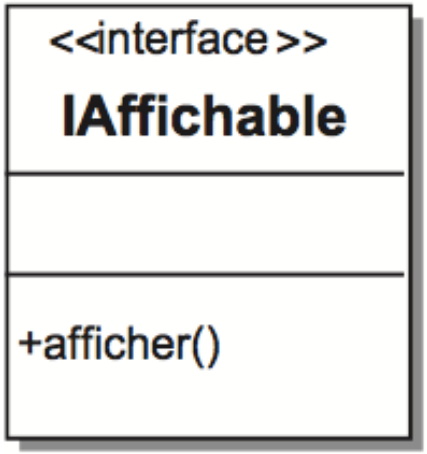
Solution


Vols
Ville
nom
1..*
0..*
dessert
InfosEscale
heureDepart
heureArrivee
/ duree
CompagnieAerienne
nom
affreteur
propose
1..*
1
numero
défini
0..1
0..*
Vol
dateDepart
dateArrivee
ouvrirReservation()
fermerReservation()
Aeroport
nom
0..*
{ordered}
0..*
escale
1
0..*
depart
1
0..*
arrivee
VolGenerique
jour
heureDepart
heureArrivee
/ duree
periodeValidite
1
décrit
0..*
0..1
1
{frozen}
concerne
Reservations
0..*
Reservation
date
numero
annuler()
confirmer()
a effectué
0..*
1
{frozen}
0..*
concerne
1
Client
nom
prenom
adresse
numTel
numFax
Passager
nom
prenom

Correspondances UML/Java

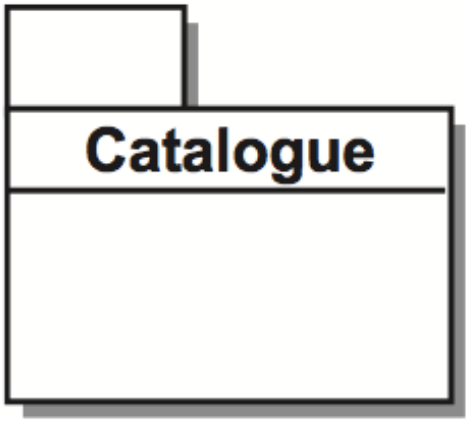



```
public class Catalogue {
    …
}
```

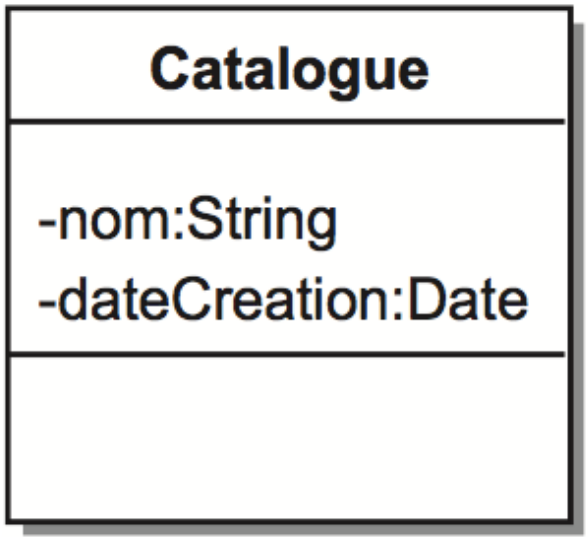



```
interface IAffichable {
    void afficher();
}
```

Correspondances UML/Java

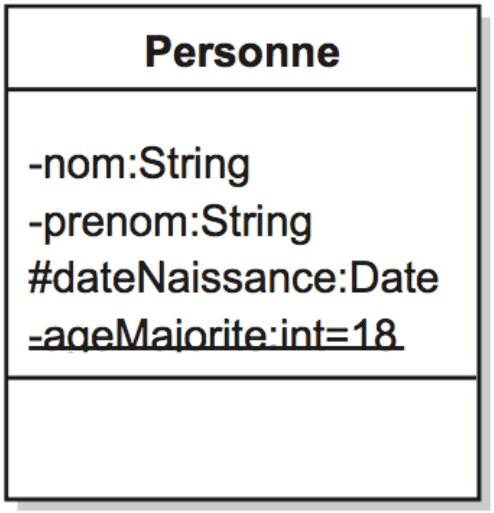



```
package catalogue;

…
```

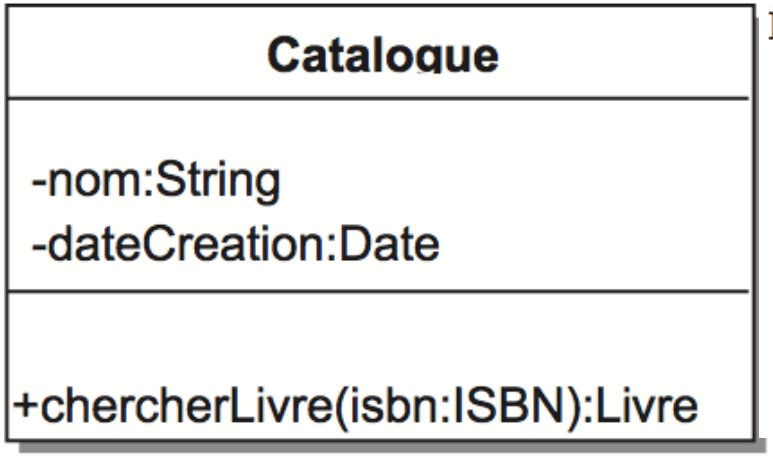



```
import java.util.Date;

public class Catalogue {
    private String nom;
    private Date dateCreation;
    …
}
```

Correspondances UML/Java

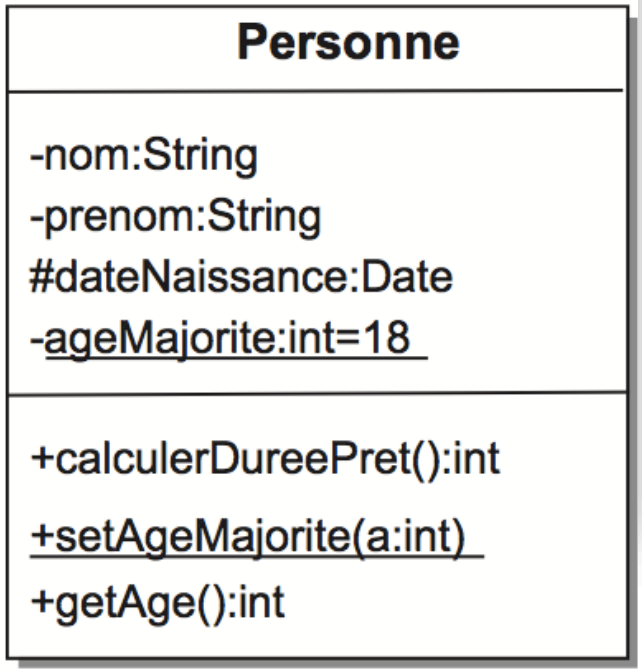



```
abstract public class Personne {
    private String nom;
    private String prenom;
    protected Date dateNaissance;
    private static int ageMajorite = 18;
}
```



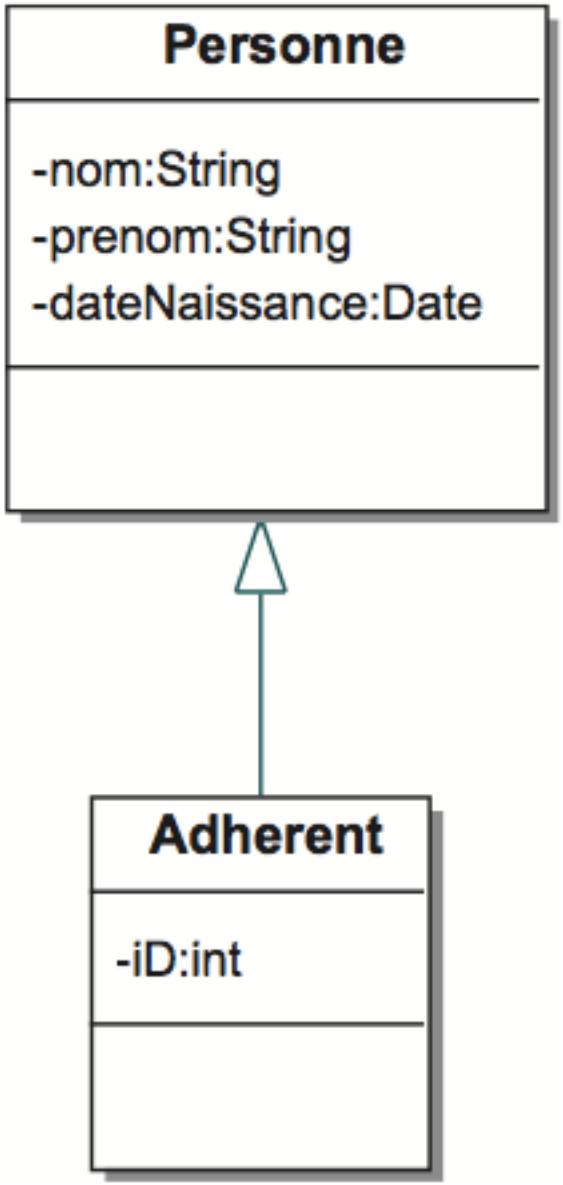

```
public class Catalogue {
    private String nom;
    private Date dateCreation;
    public Livre chercherLivre(ISBN isbn) {
        …
    }
    …
}
```

Correspondances UML/Java



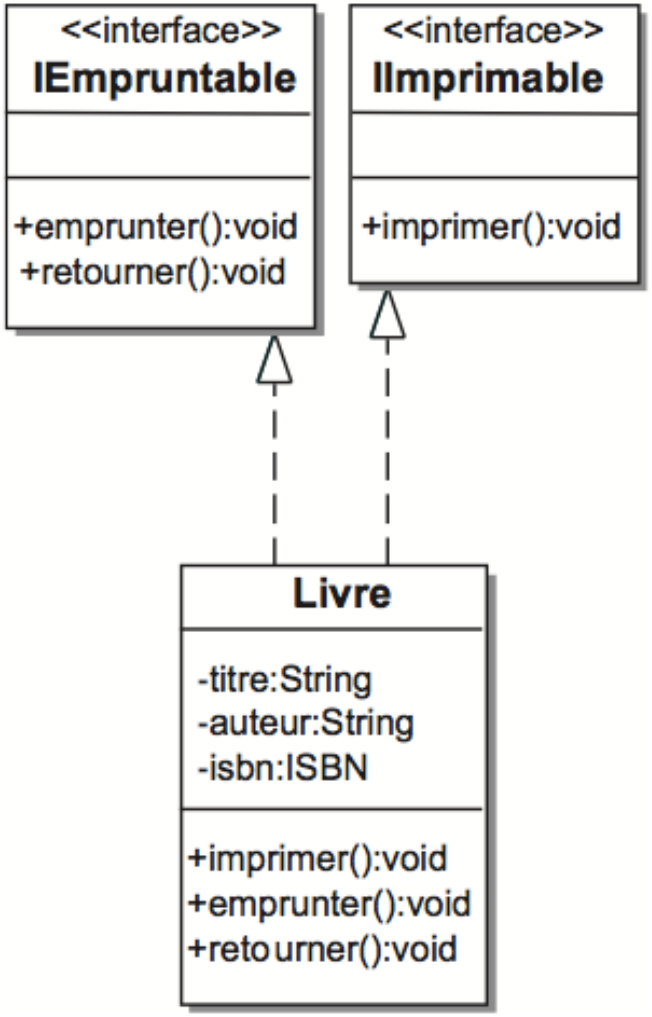

```
abstract public class Personne {
    private String nom;
    private String prenom;
    protected Date dateNaissance;
    private static int ageMajorite = 18;
    public abstract int calculerDureePret();
    public static void setAgeMajorite(int aMaj) {
    …
    }
    public int getAge() {
    …
    }
}
```

Correspondances UML/Java



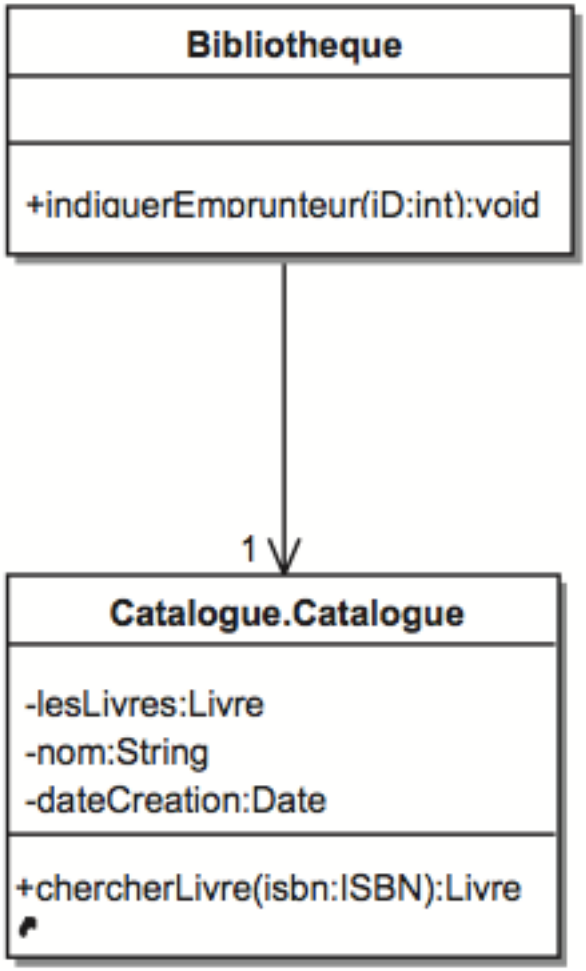

```
public class Adherent extends Personne {
    private int iD;
}
```

Correspondances UML/Java



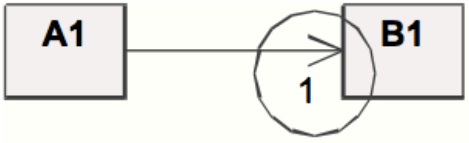

```
public class Livre implements
 IImprimable, IEmpruntable {
    private String titre;
    private String auteur;
    private ISBN isbn;
    public void imprimer(){
    …
}
    public void emprunter(){
    …
}
    public void retourner(){
    …
 }
}
```

Correspondances UML/Java



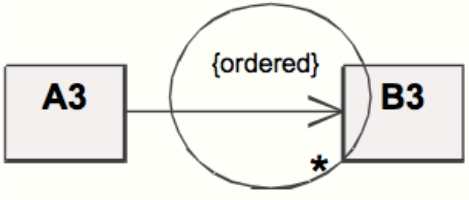

```
package bibliotheque;

import catalogue;
public class Bibliotheque {
    private Catalogue leCatalogue;
    …
    }
}
```

Correspondances UML/Java

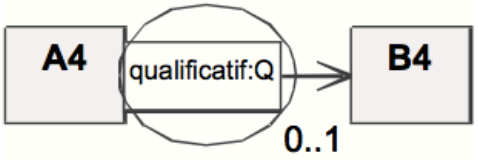



```
public class A1 {
    private B1 leB1;
    …
}
```


```
public class A2 {
    private B2 lesB2[];
    …
}
```

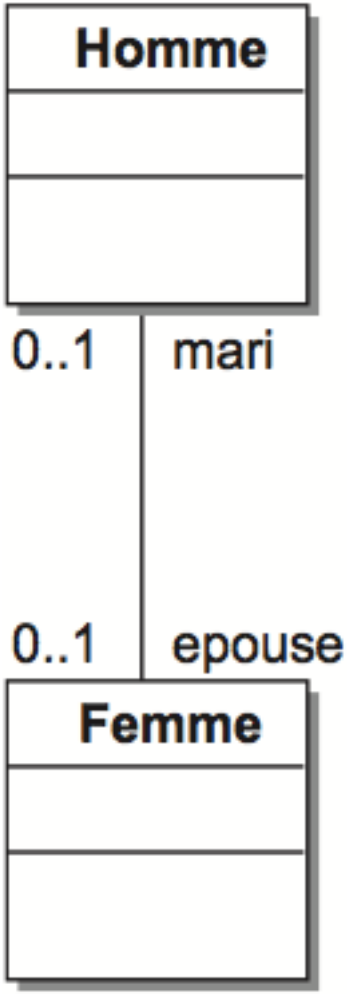



```
public class A3 {
    private List<B3> lesB3
    = new ArrayList<B3>();
    …
}
```




```
public class A4 {
    private Map<Q,B4> lesB4
    = new HashMap<Q,B4>();
```

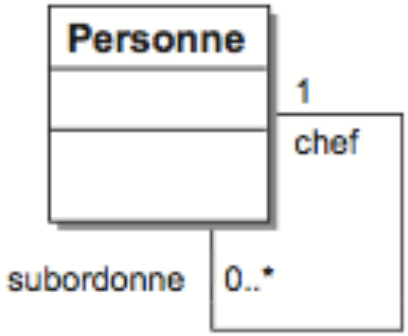
Correspondances UML/Java



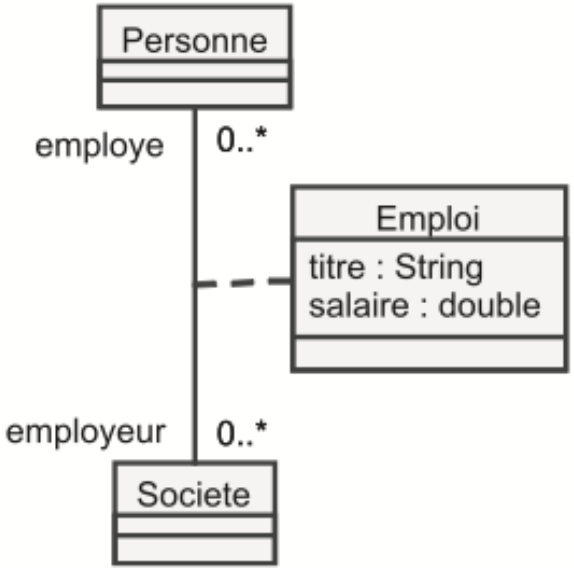

```
public class Homme {
    private Femme epouse;
    …
}
```

```
public class Femme {
    private Homme mari;
    …
}
```

Correspondances UML/Java




```
public class Personne {
    private Personne subordonne[];
    private Personne chef ;
    …
}
```




```
public class Emploi {
    private String titre;
    private double salaire;
    private Personne employe;
    private Societe employeur ;
    …
}
```