Etude de cas 1

Cette étude de cas concerne un système simplifié de réservation de vols pour une agence de voyages.

Les interviews des experts métier auxquelles on a procédé ont permis de résumer leur connaissance du domaine sous la forme des phrases suivantes :

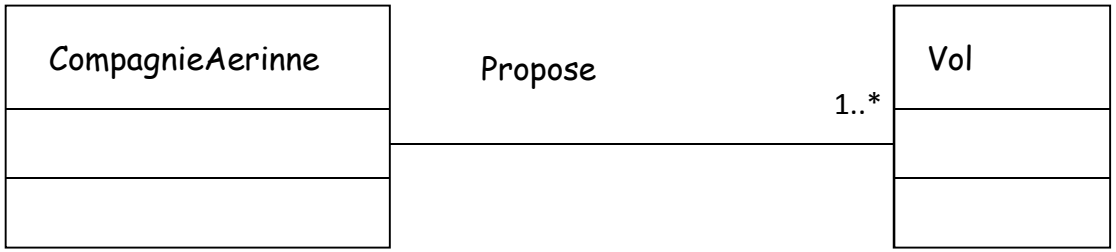
1. Des compagnies aériennes proposent différents vols.
2. Un vol est ouvert à la réservation et refermé sur ordre de la compagnie.
3. Un client peut réserver un ou plusieurs vols, pour des passagers différents.
4. Une réservation concerne un seul vol et un seul passager.
5. Une réservation peut être annulée ou confirmée.
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 un jour et une heure d'arrivée.
8. Un vol peut comporter des escales dans des aéroports.
9. Une escale a une heure d'arrivée et une heure de départ.
10. Chaque aéroport dessert une ou plusieurs villes.

➢ Modélisation de la phrase :

1° Des compagnies aériennes proposent différents vols.

•*CompagnieAerienne* et *Vols* sont des concepts importants du monde réel; ils ont des propriétés et des comportements.

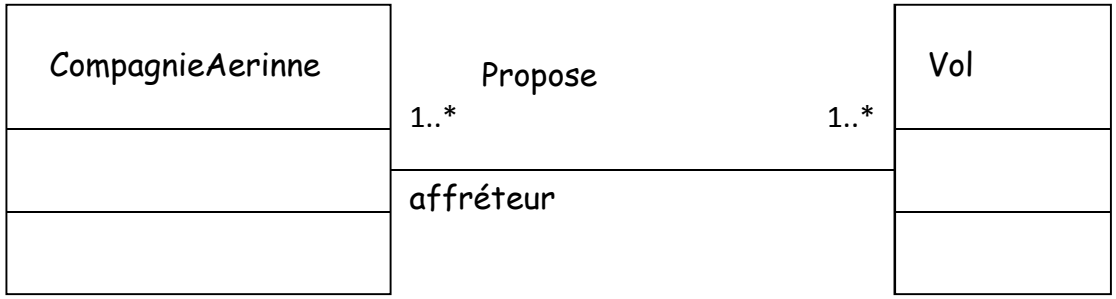
• Ce sont des objets métiers: 2 classes candidates pour cette modélisation statique.



La multiplicité « 1..* » du côté de la classe *Vol* a été préférée à une multiplicité « 0..* » car nous ne gérons que les compagnies aériennes qui proposent au moins un vol.

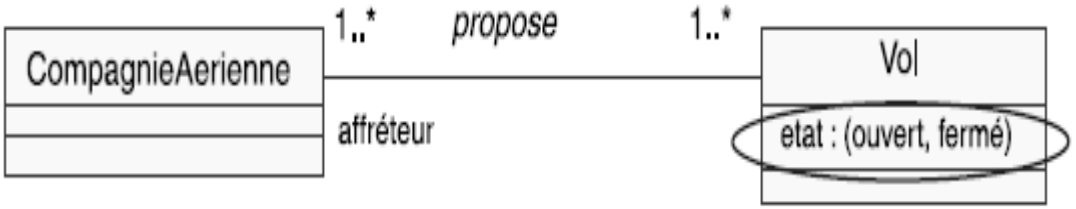
En revanche, la phrase ne nous donne pas d'indication sur la multiplicité du côté de la classe *CompagnieAerienne*. C'est là une première question qu'il faudra poser à l'expert métier.

Par la suite, nous partirons du principe qu'un vol est proposé le plus souvent par une seule compagnie aérienne, mais qu'il peut également être partagé entre plusieurs affréteurs. Au passage, on notera que le terme « affréteur » est un bon candidat pour nommer le rôle joué par la classe *CompagnieAérienne* dans l'association avec *Vol*.



➢ Modélisation de la phrase :

2° Un vol est ouvert à la réservation et fermé sur ordre de la compagnie.

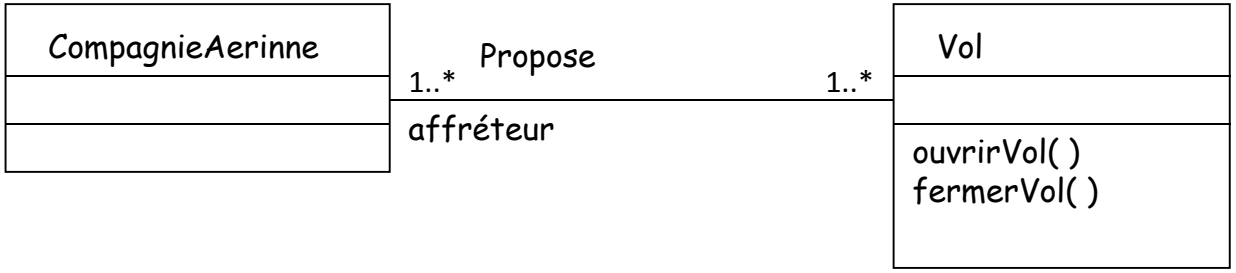


➢ Dans un diagramme de classes tout concept dynamique est modélisé en opération.

=> Il faut représenter la 2° phrase par 2 opérations : *ouvrirVol( )* et *fermerVol( )*

➢Dans quelle classe ? Responsabilité d'une classe

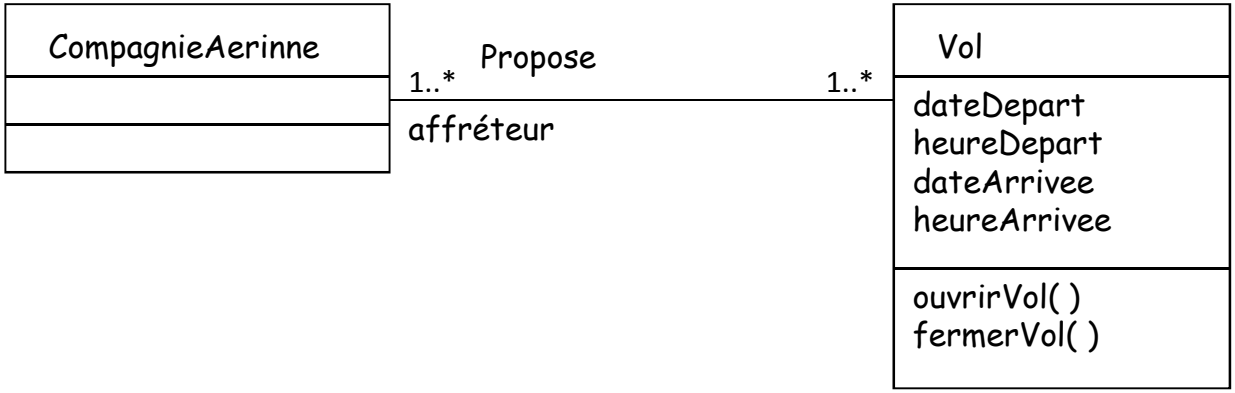
- Qui est ouvert à la réservation? C'est le vol et non la compagnie.
- Les opérations sont déclarées dans l'objet sur lequel elles doivent s'exécuter.
- Les autres pourront déclencher ces opérations par envoi de messages.
- Il faut s'assurer que la classe CompagnieAerienne a une association avec la classe vol.



➢ Modélisation des phrases :

7° Un vol a un jour et une heure de départ et un jour et une heure d'arrivée.

➢ Les dates et les heures de départ et d'arrivée ne représentent que des valeurs : attributs.



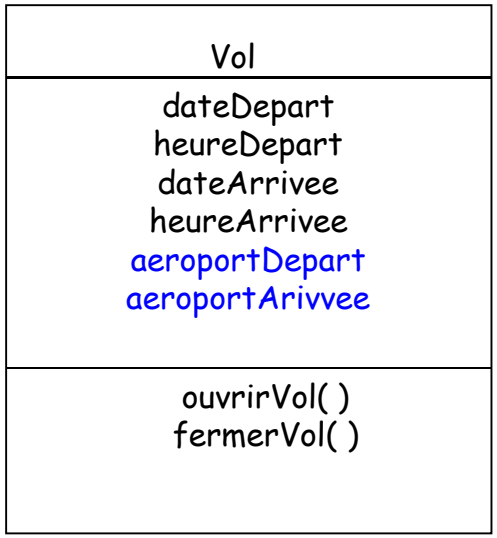
➢ Pour savoir si un élément doit être représenté en attribut ou en objet :

- ➢ S'il n' y a que sa valeur qui est intéressante : c'est plutôt un attribut.
- ➢ Si plusieurs questions peuvent concerner l'élément, alors il faut le représenter en objet.

➢ Modélisation des phrases :

6° Un vol a un aéroport de départ et un aéroport d'arrivée.

Contrairement aux notions d'heure et de date qui sont des types « simples », la notion d'aéroport est complexe ; elle fait partie du « métier ». Un aéroport ne possède pas seulement un nom, il a aussi une capacité, dessert des villes, etc. C'est la raison pour laquelle nous préférons créer une classe *Aeroport* plutôt que de simples attributs *aeroportDepart* et *aeroportArrivee* dans la classe *Vol*.



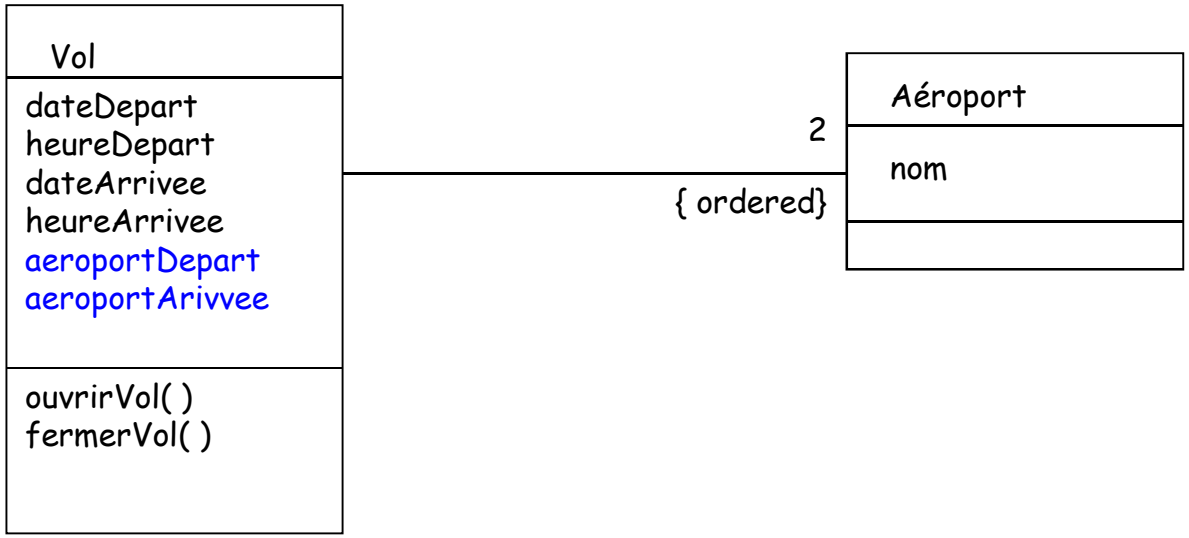
➢ Modélisation des phrases :

6° Un vol a un aéroport de départ et un aéroport d'arrivée.

➢ Par quoi peut-on représenter l'élément ''Aéroport'' ?

3 réponses sont envisageables :

1. Soit avec une classe et une association de multiplicité 2

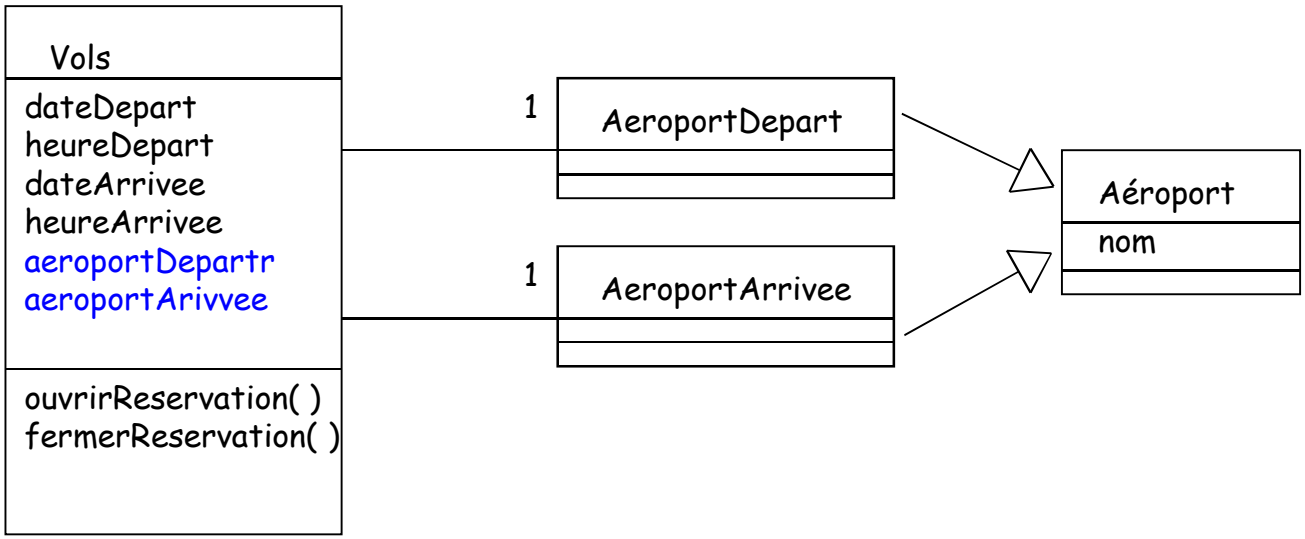


☞ Modélisation peu parlante.

➢ Modélisation des phrases :

6° Un vol a un aéroport de départ et un aéroport d'arrivée.

2. Soit avec 2 classes

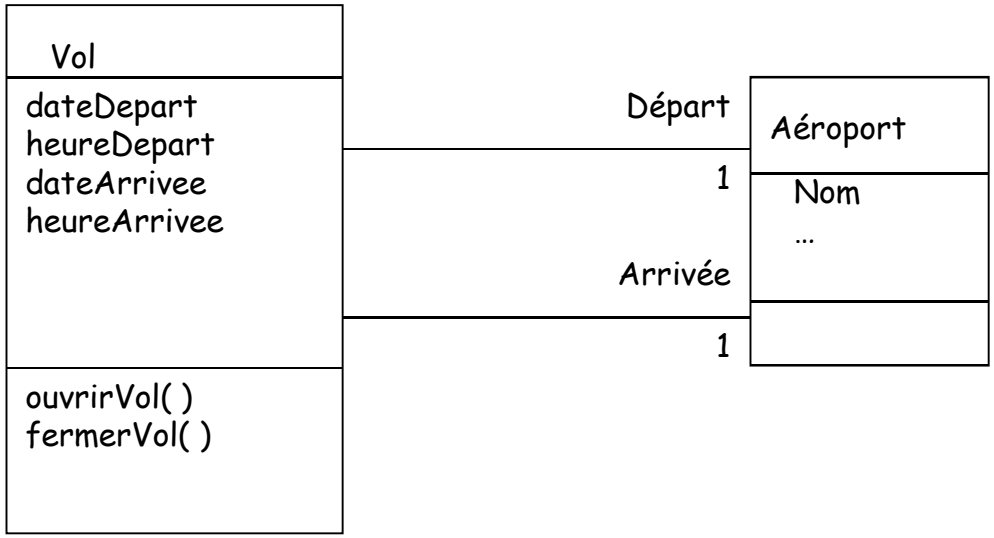


☞ Modélisation incorrecte (héritage). Tout aéroport peut être de départ et d'arrivée.

➢ Modélisation des phrases :

6° Un vol a un aéroport de départ et un aéroport d'arrivée.

2. Soit avec 2 associations

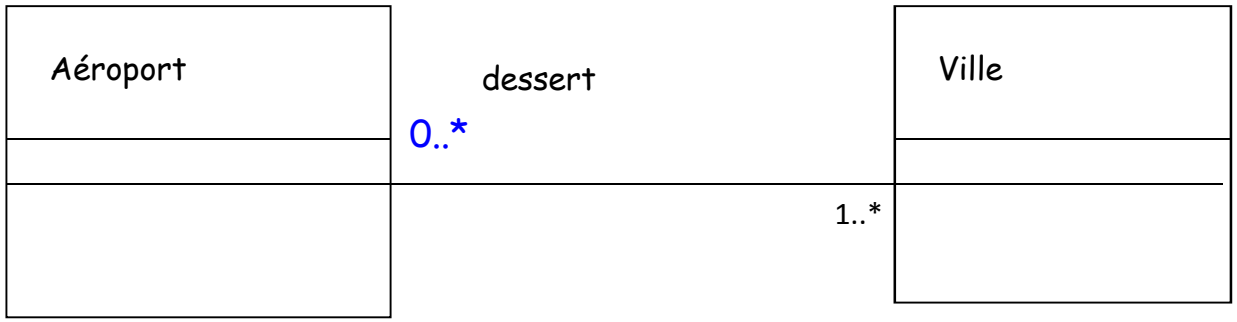


☞ Le concept de rôle s'applique parfaitement dans cette situation.

➢ Modélisation des phrases :

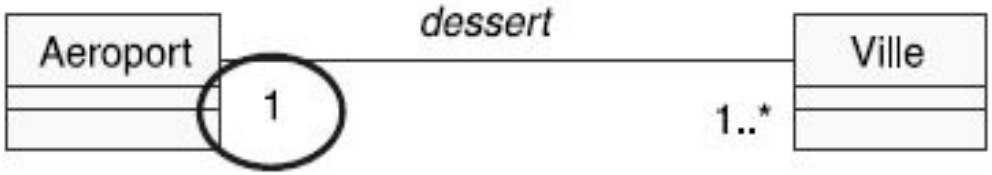
10° Chaque aéroport dessert une ou plusieurs villes

➢On ne peut pas savoir la multiplicité de ''*Aéroport*"



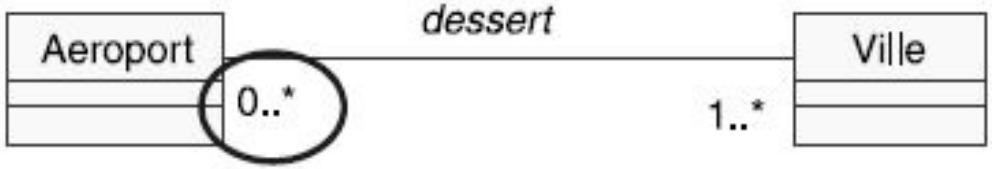
N.B Desservir des villes= Assurer un moyen de transport pour ces villes.

La question est moins triviale qu'il n'y paraît au premier abord… Tout dépend en effet de la définition exacte que l'on prête au verbe « desservir » dans notre système ! En effet, si « desservir » consiste simplement à désigner le moyen de transport par les airs le plus proche, toute ville est desservie par un et un seul aéroport.



Mais si « desservir » vaut par exemple pour tout moyen de transport aérien se trouvant à moins de trente kilomètres, alors une ville peut être desservie par 0 ou plusieurs aéroports.

C'est cette dernière définition que nous retiendrons.



➢ Modélisation des phrases :

8° Un vol peut comporter des escales dans des aéroports

9° Une escale a une heure d'arrivée et une heure de départ.

➢ Une escale a les propriétés heure d'arrivée et heure de départ, c'est donc une classe.

➢ *''Escale''* a peu d'informations propres. Elle n'est qu'une partie de *''Vol''*.

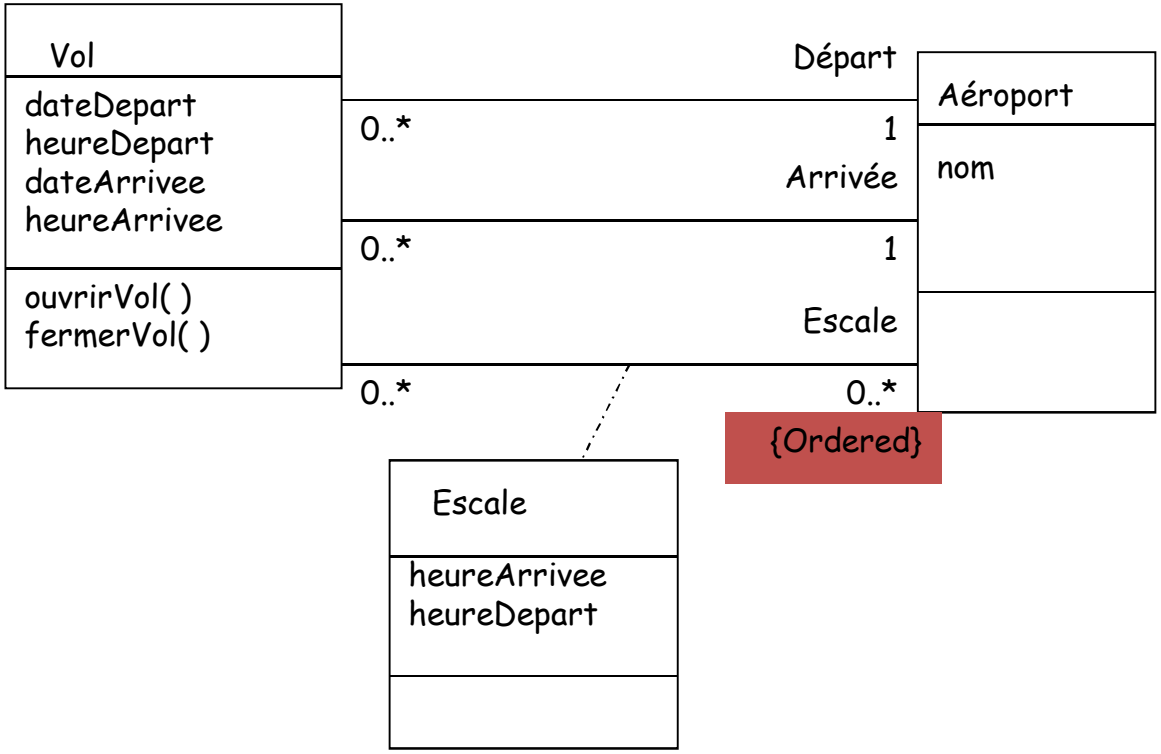
➢ On peut la représenter comme une spécialisation de *''Aéroport''* . Mais elle n'est pas totalement un aéroport.

➢ Modélisation des phrases :

8° Un vol peut comporter des escales dans des aéroports

9° Une escale a une heure d'arrivée et une heure de départ.

➢ La meilleure solution serait de la modéliser comme une classe d'association entre *''Vols''* et *''Aéroport''*.

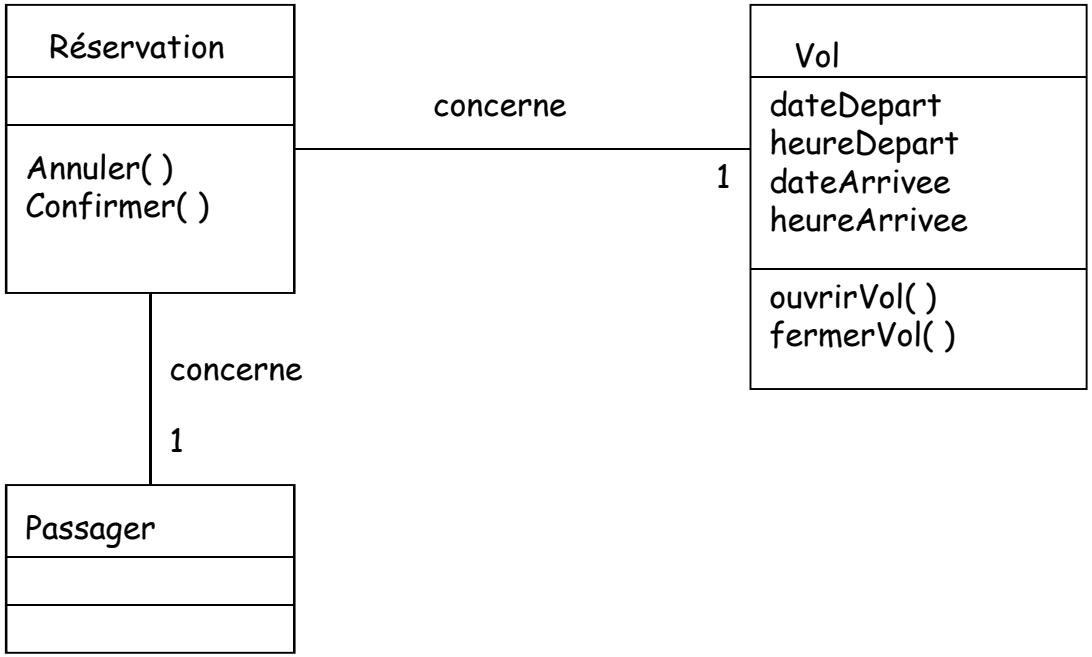


➢ Modélisation des phrases :

4° Une réservation concerne un seul vol, et un seul passager.

5° Une réservation peut être annulée ou confirmée.

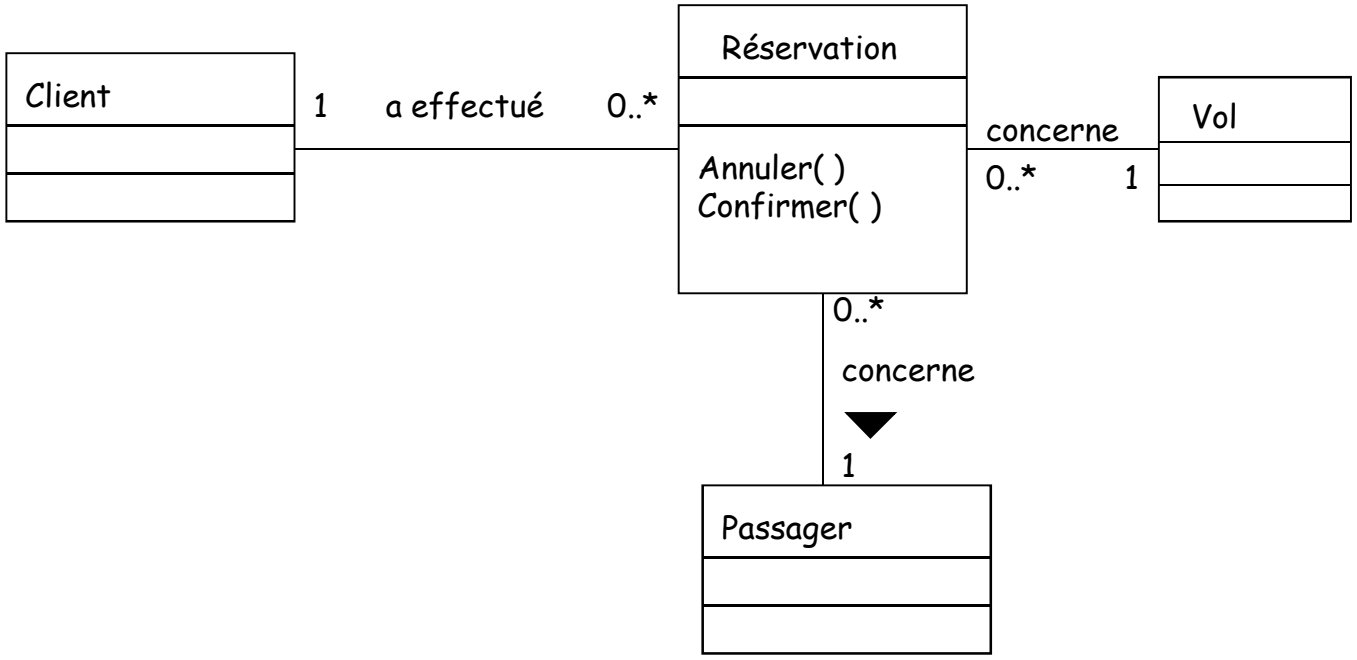
- La réservation et le passager sont 2 concepts métier : 2 classes d'objets
- Un réservation concerne un seul vol et un seul passager: donc 2 associations entre ''*Vol*'' et ''*Réservation*'' et entre ''*Réservation*'' et ''*Passager*''.
- La 5° phrase se traduit par l'ajout de 2 opérations *annuler( )* et *confirmer( )* dans ''*Reservation*''.



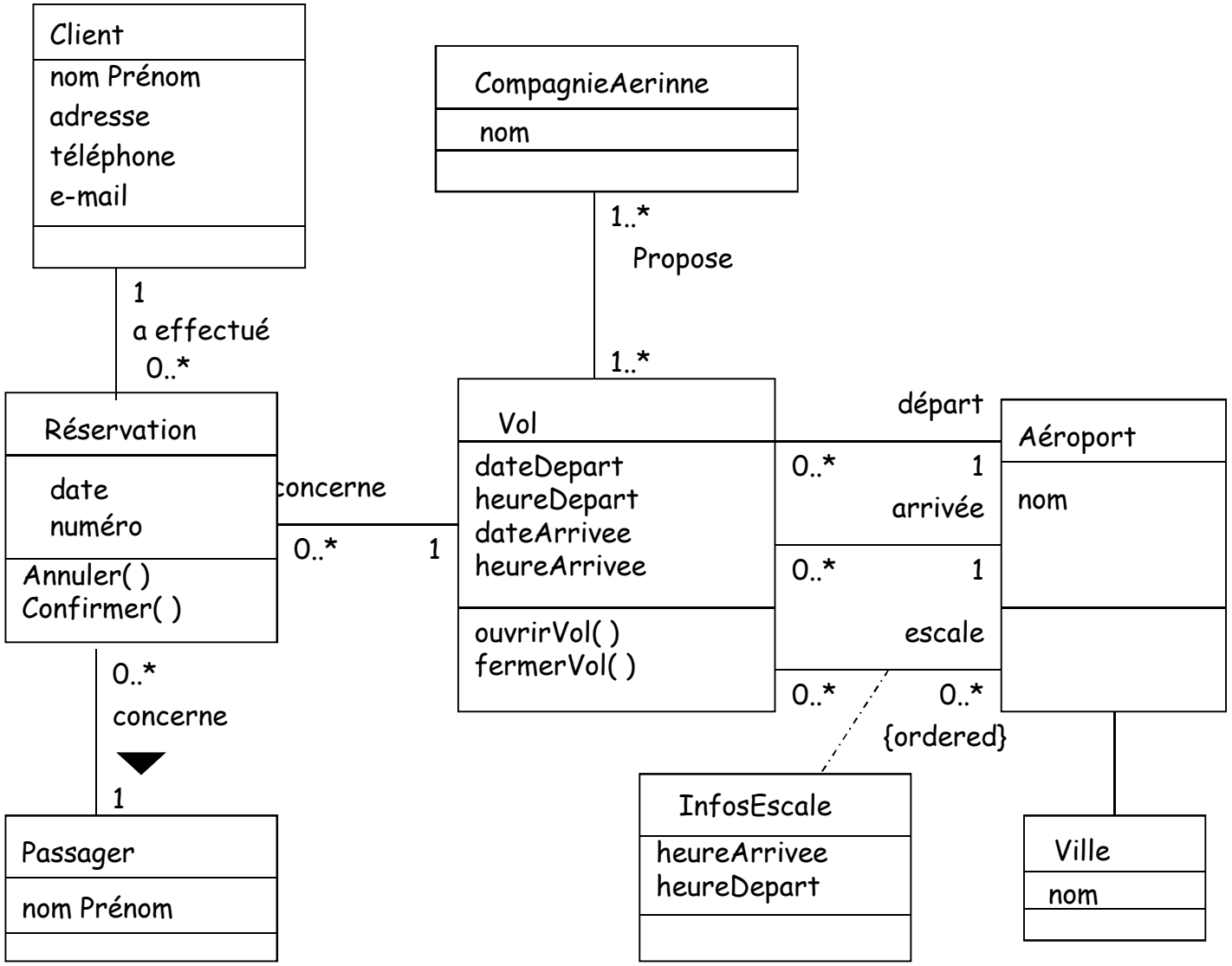
➢ Modélisation des phrases :

3° Un client peut réserver un ou plusieurs vols, pour des passagers différents.

➢ Il faut discerner un client d'un passager

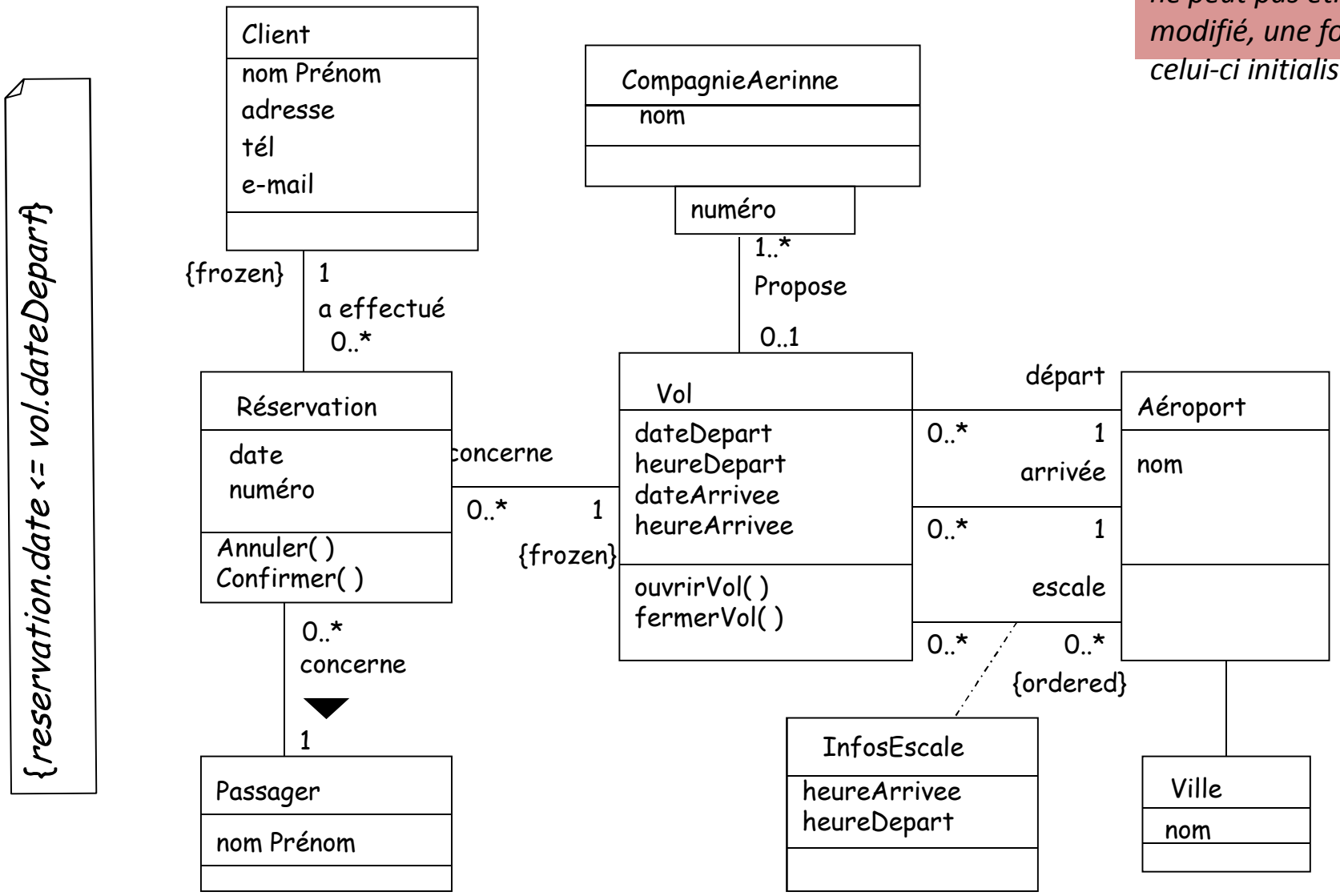


➢ Le diagramme des classe complet est :

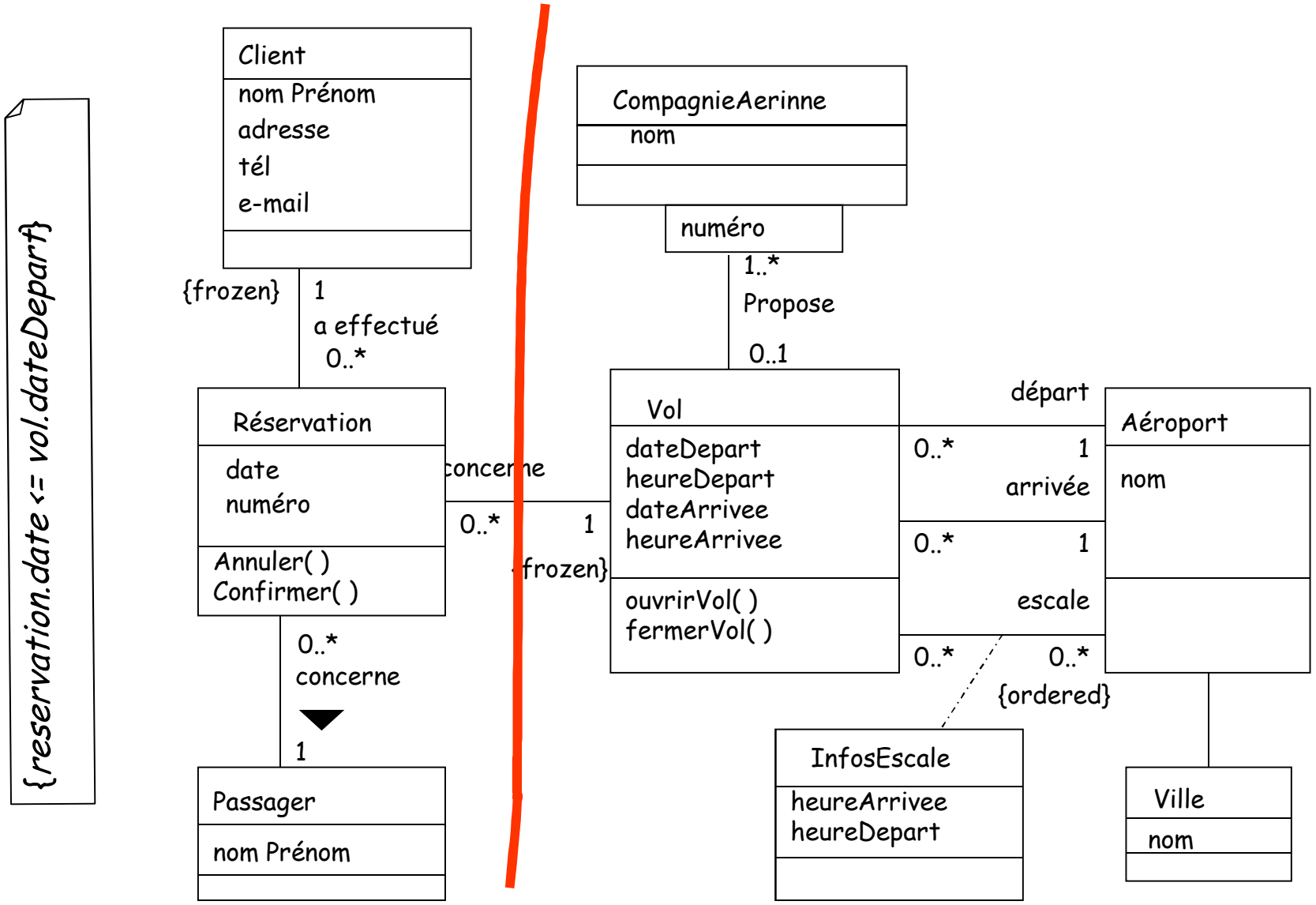


➢ Diagramme des classe complet et annoté

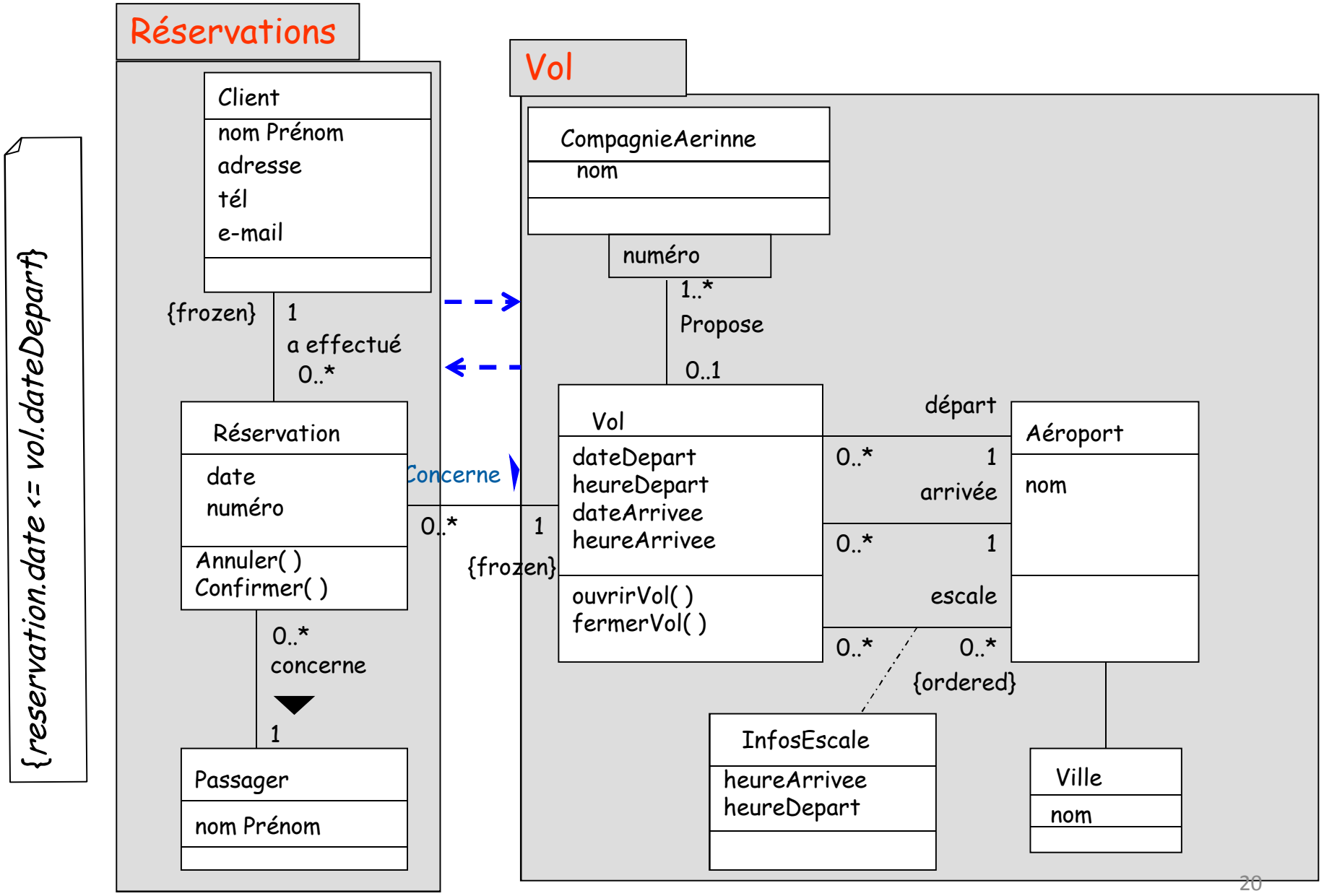
Frozen (readOnly, depuis UML2) spécifie que l'attribut ne peut pas être modifié, une fois celui-ci initialisé.

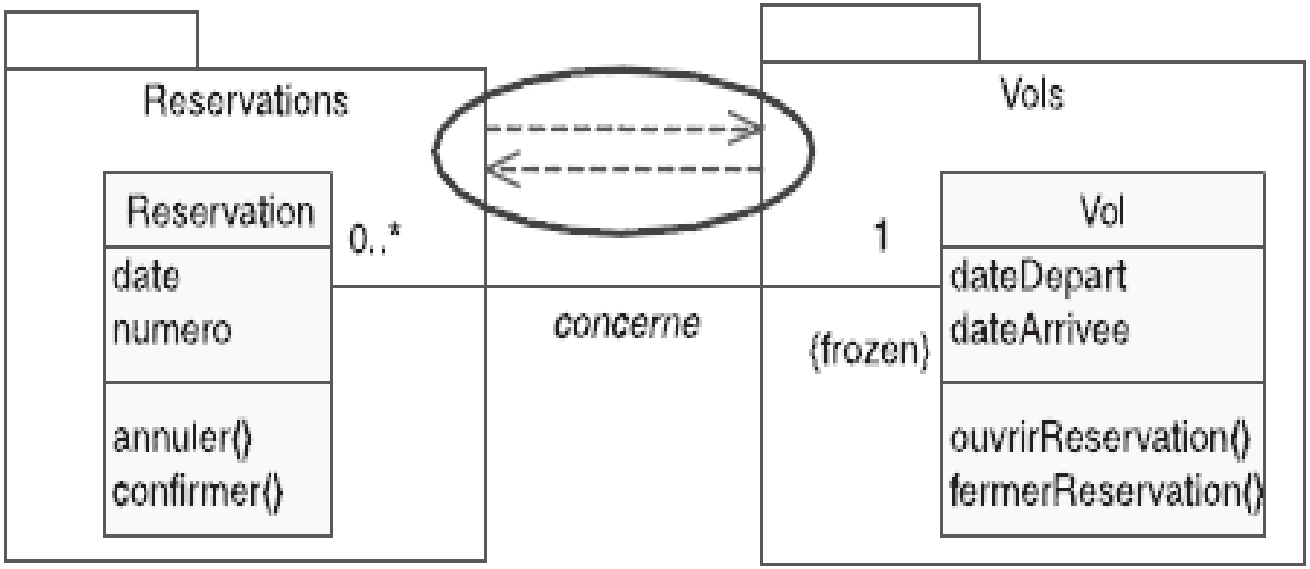


➢ Structuration en package



➢ Diagramme des classe complet et annoté





- Réduire la dépendance mutuelle afin d'augmenter la modularité et l'évolutivité d'une application

- Il faudrait limiter l'association à un seul sens.

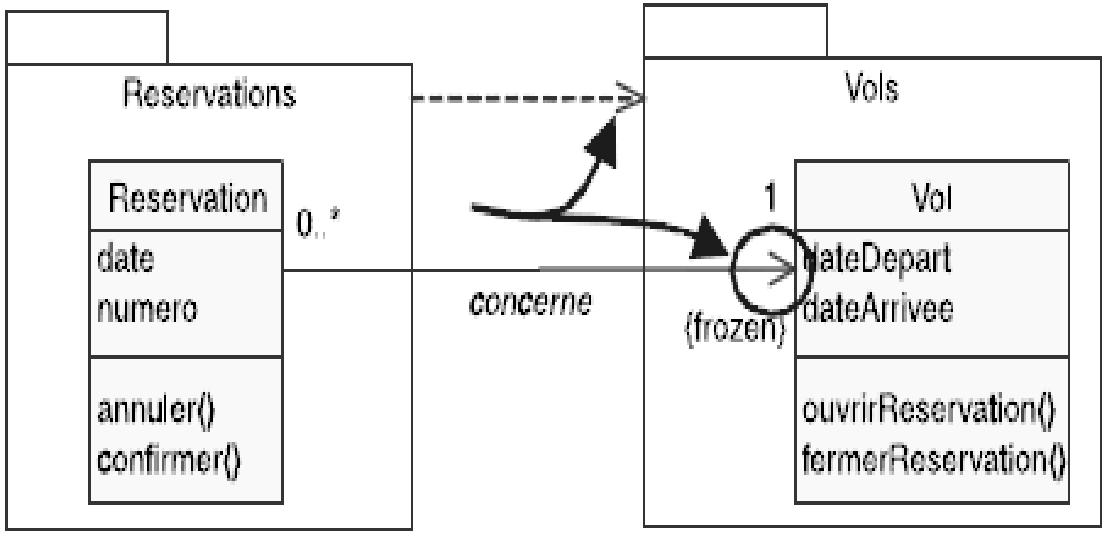
Remarque: Navigabilité Et Dépendance

- Une association entre deux classes A et B permet par défaut de naviguer dans les deux sens entre des objets de la classe A et des objets de la classe B.

- Cependant, il est possible de limiter cette navigation à une seule des deux directions, pour éliminer une des deux dépendances induites par l'association.

- Dans notre exemple, nous allons faire un choix et privilégier un sens de navigation afin d'éliminer une des deux dépendances.

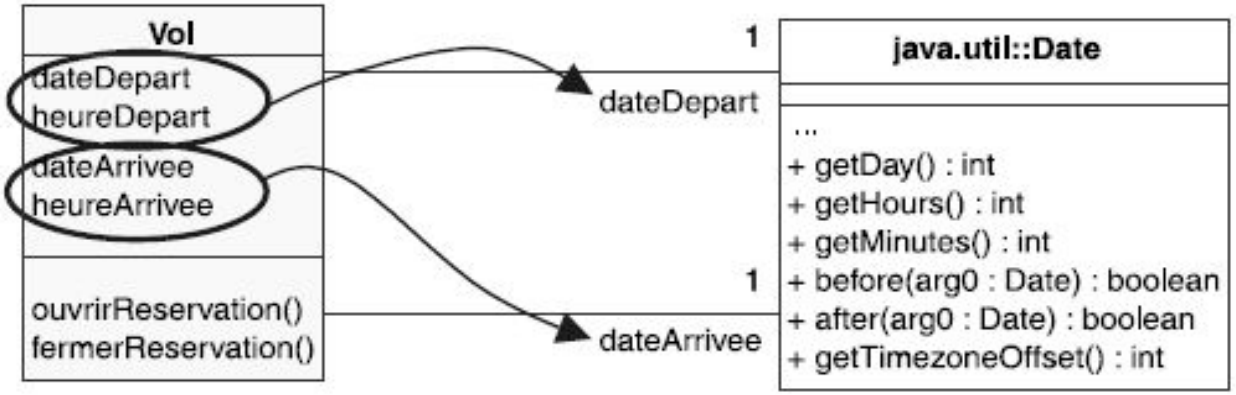
- Il est certain qu'une réservation ne va pas sans connaissance du vol qui est concerné, alors qu'un vol existe par lui-même, indépendamment de toute réservation.



Modèle d'analyse et de conception

DIFFÉRENCE ENTRE MODÈLE D'ANALYSE ET DE CONCEPTION

Dans le code Java de l'application finale, il est certain que nous utiliserons explicitement la classe d'implémentation `Date` (du package `java.util`).



Fragment possible de diagramme de classe de conception détaillée avec Java

Il ne s'agit pas là d'une contradiction, mais d'une différence de niveau d'abstraction, qui donne lieu à deux modèles différents, un modèle d'analyse et un modèle de conception détaillée, pour des types de lecteurs et des objectifs distincts.

2° Un vol est ouvert à la réservation et fermé sur ordre de la compagnie.

Nous allons maintenant nous intéresser à la deuxième phrase. Les notions d'ouverture et de fermeture de la réservation représentent des concepts dynamiques. Il s'agit en effet de changements d'état d'un objet *Vol* sur ordre d'un objet *CompagnieAérienne*. Une solution naturelle consiste donc à introduire un attribut énuméré *etat*, comme cela est montré sur la figure suivante.

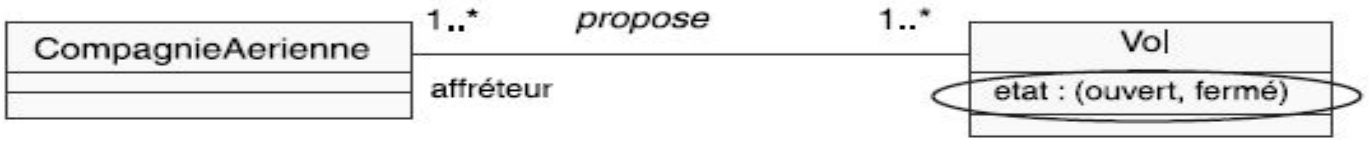


Figure 3-3. Première modélisation de la phrase 2

En fait, ce n'est pas la bonne approche : tout objet possède un état courant, en plus des valeurs de ses attributs. Cela fait partie des propriétés intrinsèques des concepts objets. La notion d'état ne doit donc pas apparaître directement en tant qu'attribut sur les diagrammes de classes : elle sera modélisée dans le point de vue dynamique grâce au diagramme d'états (voir chapitres 5 et 6). Dans le diagramme de classes UML, les seuls concepts dynamiques disponibles sont les opérations.

Or, les débutants en modélisation objet ont souvent du mal à placer les opérations dans les bonnes classes ! Plus généralement, l'assignation correcte des responsabilités aux bonnes classes est une qualité distinctive des concepteurs objets expérimentés…

Source: UML2 par la pratique