

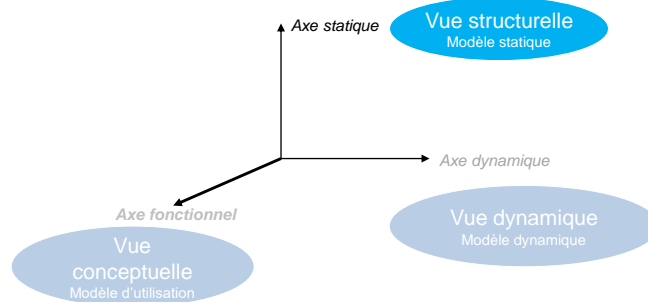
UML

Axe statique

Description statique des éléments du système

Certaines illustrations sont extraites du cours de D. Longuet LRI
Des livres de P. Roques (UML par la pratique)

Axe statique



De quoi est fait le système ?

Réalisation d'un diagramme de classes

- *il représente l'ensemble des classes d'un système et les relations entre ces classes*
- *il est réalisé progressivement à partir de morceaux de connaissances d'un domaine*
- *Il fait abstraction des aspects dynamiques et temporels*

Concept de Classe

- Décrire la structure des entités manipulées par le système
 - Conception orientée objet
 - Système = ensemble d'objets
 - Objet = données + fonctions
 - Représentation du système comme un ensemble d'objets interagissant
- Classe
 - Représentation abstraite d'objets qui ont les mêmes caractéristiques

3

Concept de Classe

- Type abstrait caractérisé par des propriétés (attributs et comportements) communes à un ensemble d'objets
- Rectangle à trois compartiments
 - nom
 - Attributs
 - Opérations

Nom de classe
Attribut 1 Attribut 2
Opération 1 Opération 2

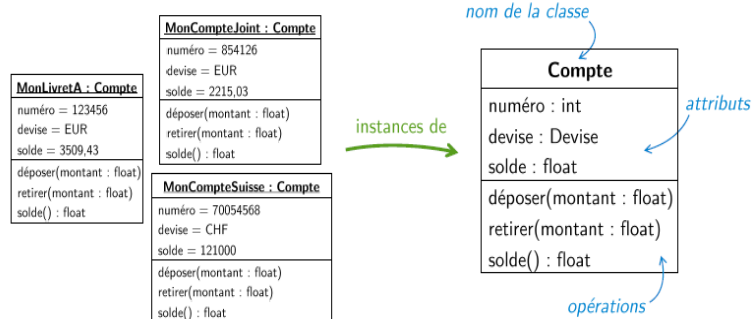
4

Objet

- Un objet permet de représenter une entité concrète ou abstraite du domaine d'application
 - Il est caractérisé un état et par un comportement
 - État (attributs)
 - ensemble des valeurs des propriétés de l'objet à un instant donné
 - Comportement (opérations)
 - détermine la façon dont l'objet réagit à des sollicitations (messages reçus de son environnement), ensemble d'opérations que l'objet peut effectuer
 - Il a une identité (adresse mémoire)
 - Des objets différents peuvent avoir les mêmes valeurs d'attributs
- Classe
 - Regroupement d'objets (instances)

5

Exemple d'objets et de classe



6

Concept de classe

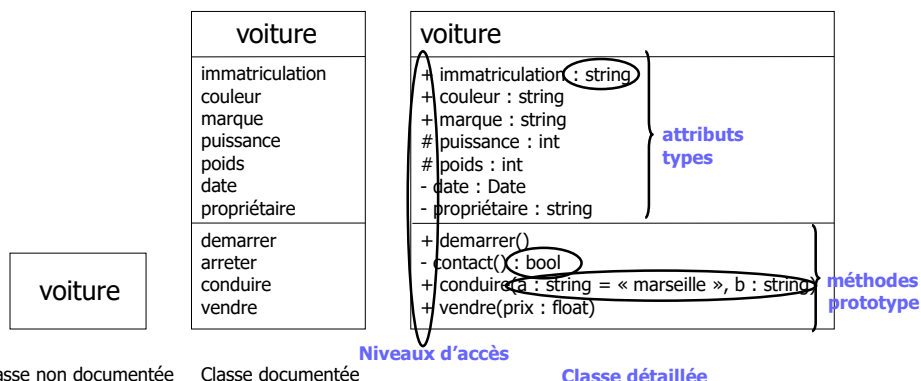
Nom de classe
Attribut 1 Attribut 2
Opération 1 Opération 2

- Classe
 - Nom
 - {Attribut - Attribut dérivé} – nom et type
 - {Opération} – services offerts par la classe – nom, arguments et valeur de retour
 - Visibilité des attributs et des opérations ('+', '-', '#')
 - + public – accessible pour tout utilisateur de la classe, et la classe elle-même
 - privé (accessible par la classe elle-même seulement)
 - # protected (accessible par la classe elle-même et ses sous-classes)
- Objet
 - instance de la classe

7

Exemples / niveaux d'abstraction

Ne pas représenter les attributs ou les opérations d'une classe sur un diagramme, n'indique pas que cette classe n'en contient pas (filtre visuel)



8

Concept de classe

- Classe
 - Représentation simplifiée
- Objet
 - instance de la classe
 - Représentation des **objets**
 - désignation directe et explicite *monlivretA*
 - la désignation inclut le nom de la classe *monlivretA : Compte*
 - désignation anonyme d'un objet d'une classe donnée *:Compte*

monlivretA

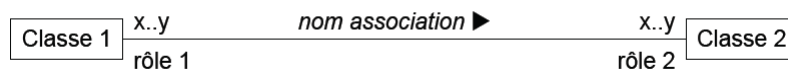
monlivretA:
Compte

: Compte

9

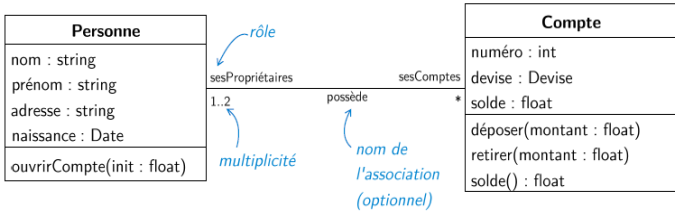
Concept d'association

- Association
 - Relation sémantique durable entre 2 classes, bidirectionnelle



10

Exemple



11

Lien entre objets



12

Remarque

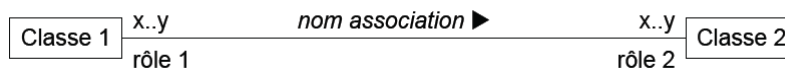
- L'association est instanciable dans un diagramme d'objets, sous forme de liens entre objets issus de classes associées
- Représenter les objets et leurs liens à un instant donné

13

Concept d'association

- Association
 - Multiplicité
 - * nombre d'objets liés par l'association
 - * pour chaque association, deux décisions à prendre : deux extrémités
 - Rôle

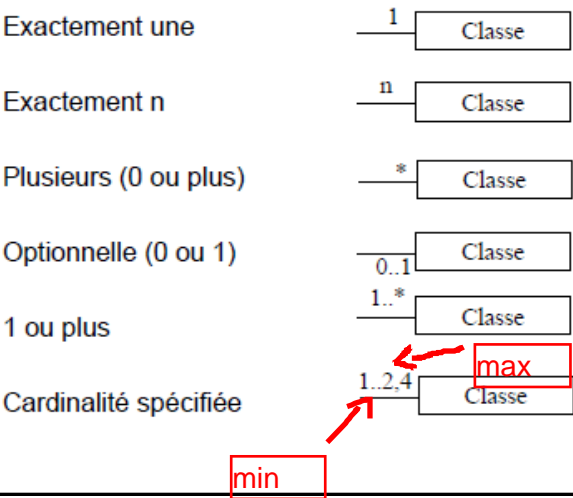
Nom	→ forme verbale, sens de lecture avec flèche
Rôles	→ forme nominale, identification extrémité association
Multiplicité	→ 1, 0..1, 0..*, 1..*, n..m



14

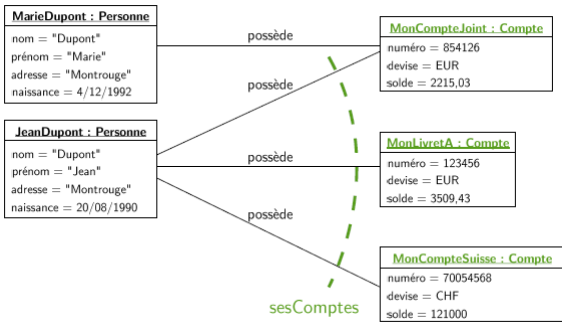
Concept d'association

• Multiplicité

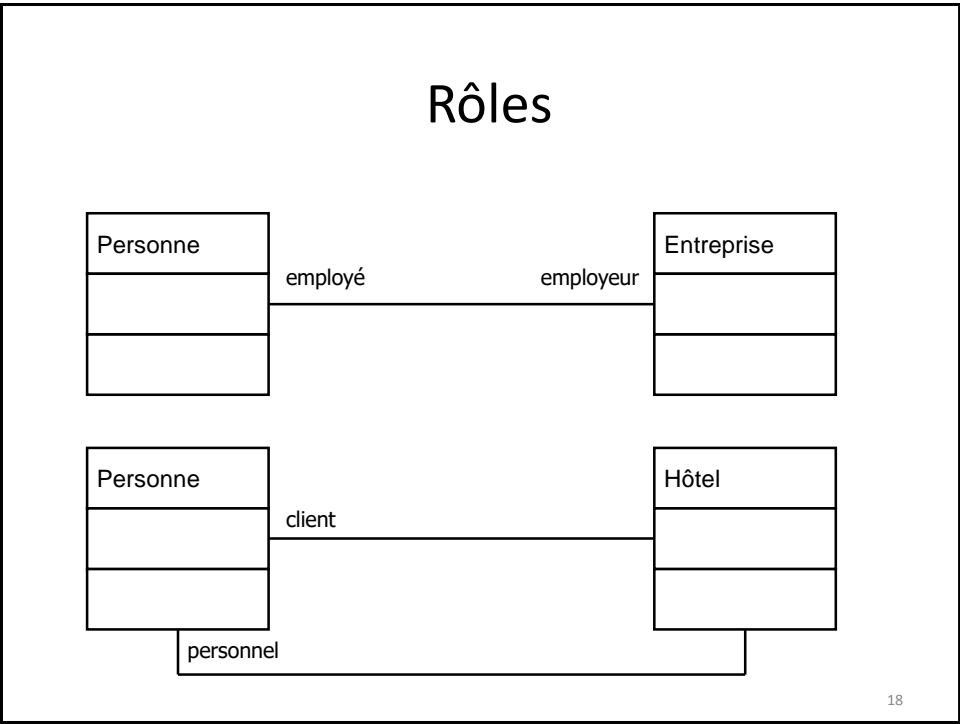
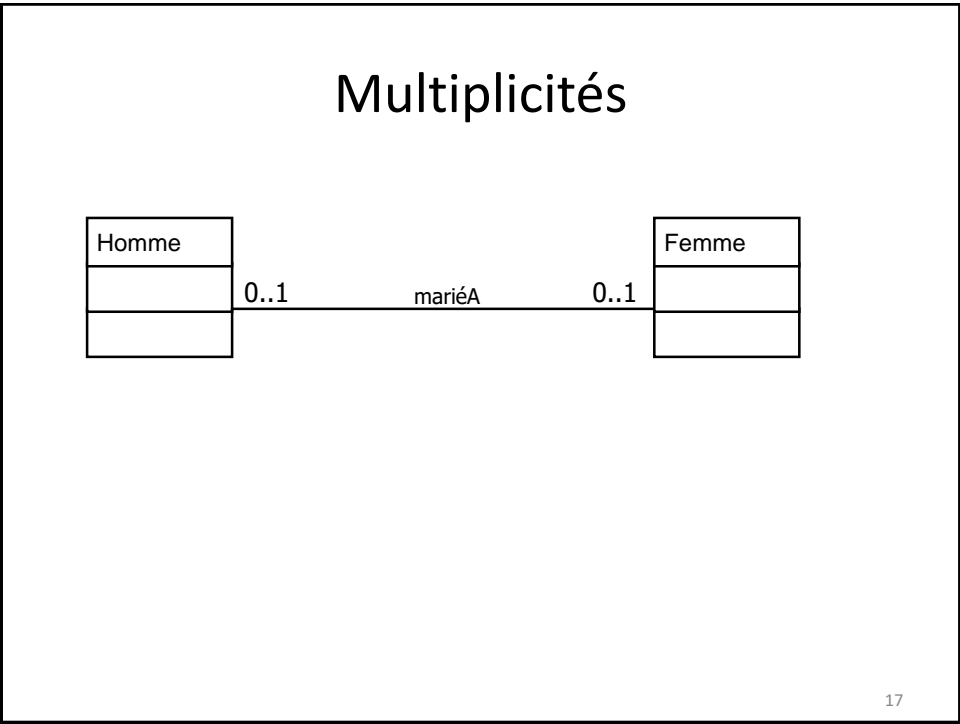


15

Lien entre objets

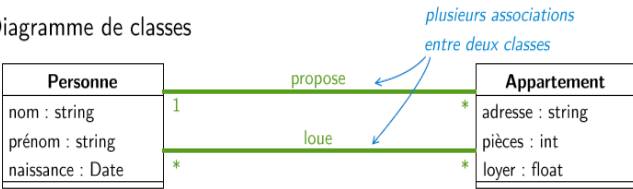


16



Autre exemple

Diagramme de classes



19

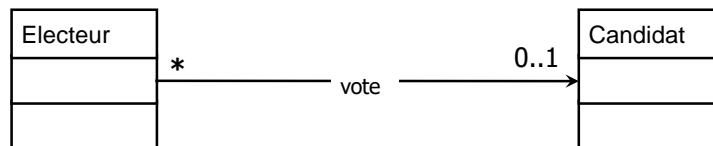
Divers

- Les associations ont une durée de vie, sont indépendantes les unes des autres
- Bidirectionnalité par défaut
- Restriction de la navigation à une direction

20

Association à navigabilité restreinte

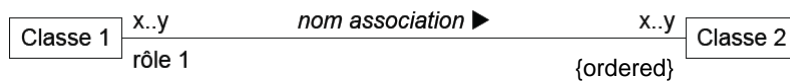
- La réduction de la portée de l'association peut être exprimée dans un modèle pour indiquer que les objets d'une classe ne "connaissent" pas les instances d'une autre
 - Depuis électeur, on a accès au candidat mais pas l'inverse



21

Concept d'association

- Contraintes sur les associations
 - {ordonné, ou_exclusif, gelé, plus, etc.}
 - à noter entre {} à une extrémité



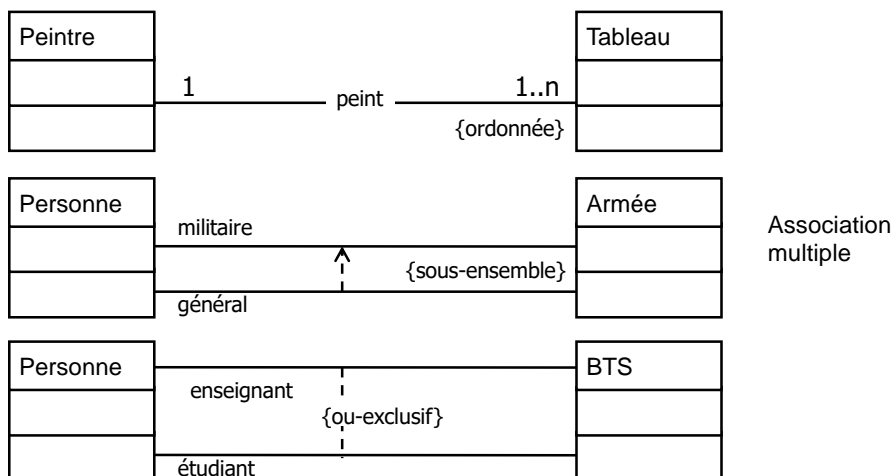
22

Contraintes

- Contrainte **{ordonnée} {ordered}**: une relation d'ordre décrit les objets
 - Sur une extrémité d'association
- Contrainte **{sous-ensemble} {subset}**: une collection est incluse dans une autre collection
 - Entre 2 associations
- Contrainte **{ou-exclusif} {xor}**: pour un objet donné, une seule association est valide
 - Entre 2 associations

23

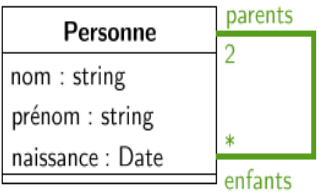
Exemples



24

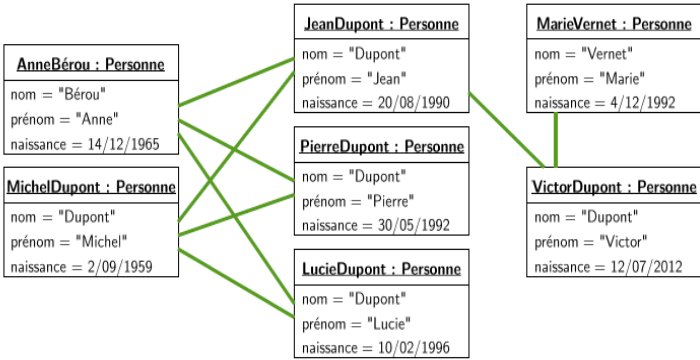
Association réflexive

- Elle lie des objets de même classe – rôles obligatoires



25

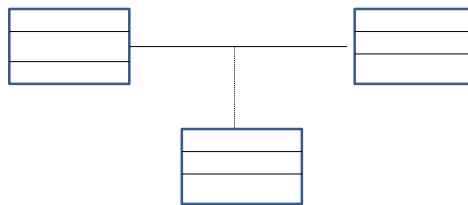
Exemple de diagrammes d’objets



26

Classes association

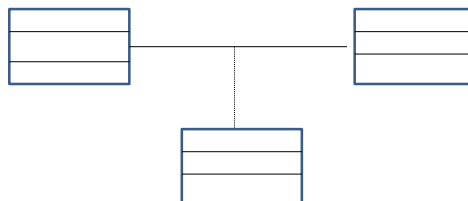
- Classes d'association
 - Pour ajouter attributs et opérations à des associations
 - Instance unique de la classe associatio pour chaque lien entre objets



27

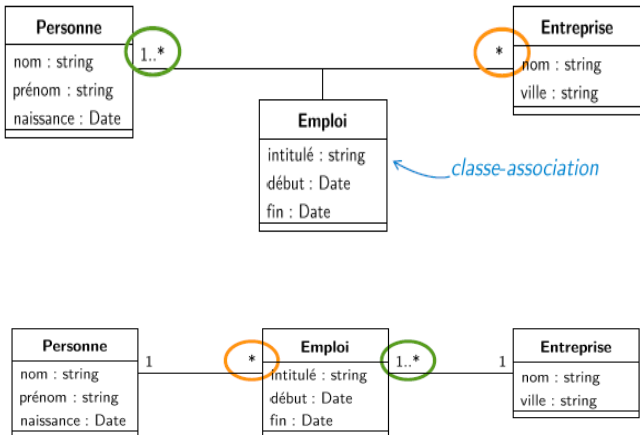
Classes association

- Quelques indices pour l'utilisation
 - un attribut est lié à une association
 - association *.* entre deux classes + informations liées à l'association



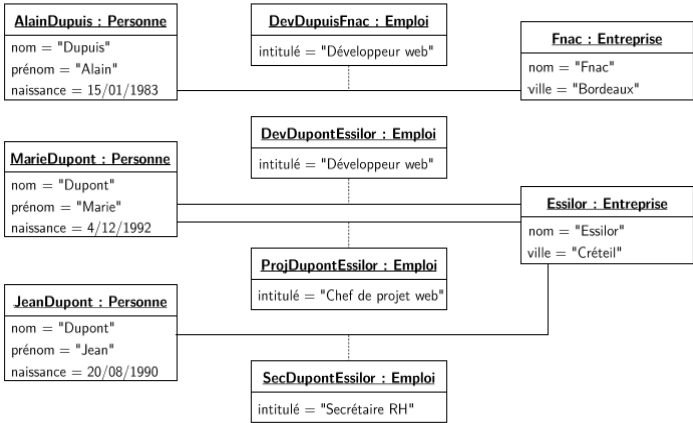
28

Exemples



29

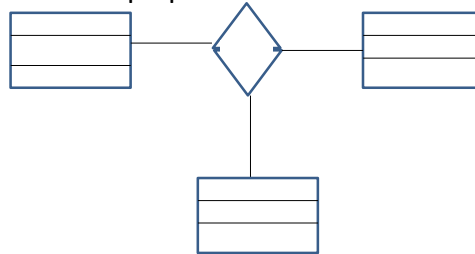
Diagramme d'objets



30

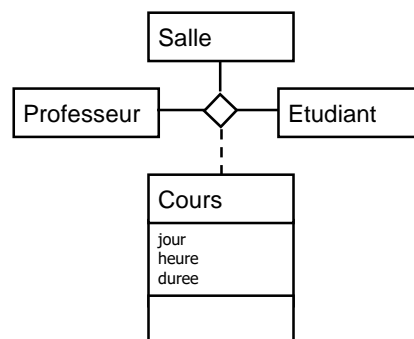
Association ternaire (n-aire)

- Les associations sont pour la plupart binaires (2 classes)
- Au delà → association ternaire
 - Groupe de liens entre au moins trois instances
 - Instance de l'association = n-uplet des attributs des instances impliquées



31

Exemple



32

Relations spécifiques

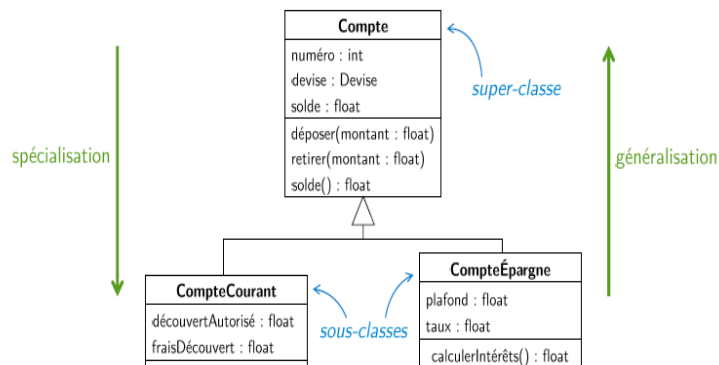
Généralisation - spécialisation

- Pour améliorer encore la finesse de modélisation
 - Spécialisation : raffinement d'une classe en une sous-classe
 - Généralisation : abstraction d'un ensemble de classes en super-classe
 - Partage des attributs, d'opérations, de contraintes



33

Exemple



34

Relations spécifiques

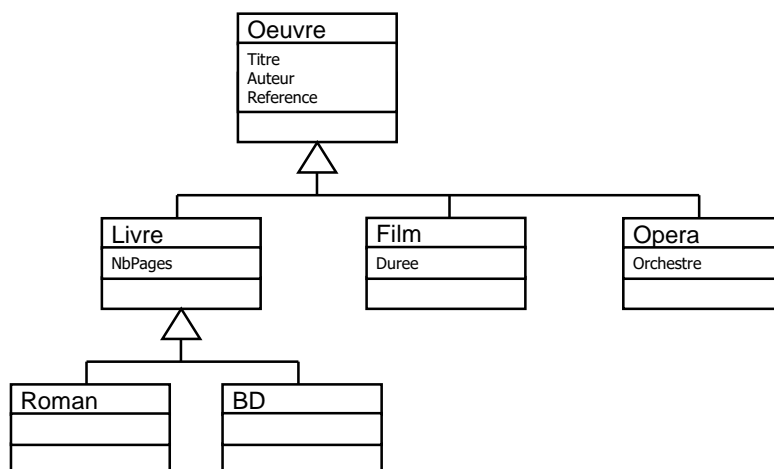
Généralisation - spécialisation

- Les instances de la sous-classe sont des instances de la super classe
- Toutes les propriétés de la classe parent doivent être valables pour les classes enfant
- Deux interprétations (héritage)
 - niveau conceptuel : un concept est plus général qu'un autre
 - Implémentation : héritage des attributs et méthodes



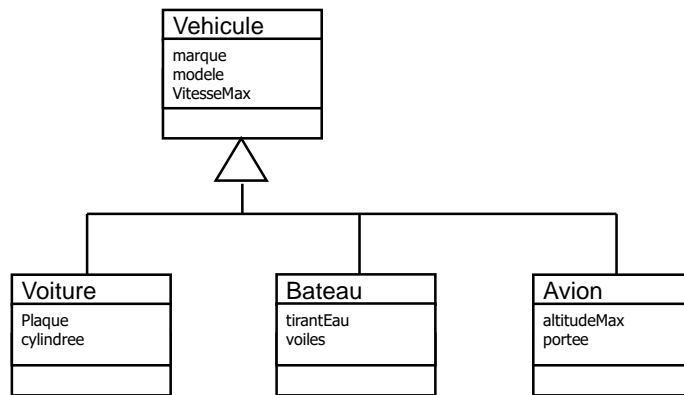
35

Spécialisation



36

Généralisation



37

Sémantique de l'héritage

- **Substitution**
 - Un étudiant est une personne, ce qui s'applique à une personne peut être appliqué à un étudiant.
- **Spécialisation**
 - Un étudiant est une personne car il a toutes les propriétés (attributs et opérations) d'une personne plus des propriétés spécifiques.
- **Inclusion**
 - Un étudiant est une personne car l'ensemble des étudiants est un sous-ensemble de celui des personnes.

38

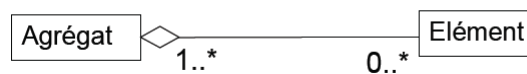
Classe abstraite

- Sans instances, seulement une base pour les classes héritées

39

Relations spécifiques Agrégation - Composition

- Associations asymétriques, fortes, non nommées
 - rôle prépondérant d'une extrémité
 - Agrégation
 - relation de type **composant-composite**



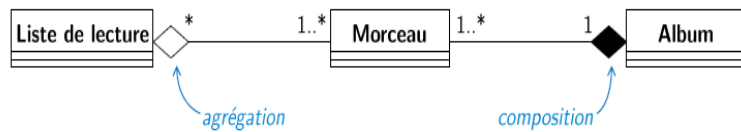
- Composition
 - Tous les éléments sont indispensables par nature à l'existence du composé



40

Exemple

Lecteur de contenu audio permettant de créer des listes de lecture

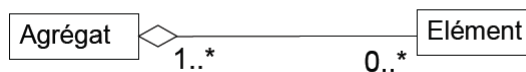


1

41

Relations spécifiques Agrégation - Composition

- Agrégation
 - structure d'arbre sous-jacente (pas de cycle)
 - La création ou destruction du composite est indépendante de la création ou destruction de ses composants



- Composition
 - Le composite contient ses composants
 - Sa suppression entraîne celle de ses composants



42

Agrégation

- Agrégation
 - Représente une relation de type **ensemble - élément**
 - Un agrégat est la somme de ses parties



43

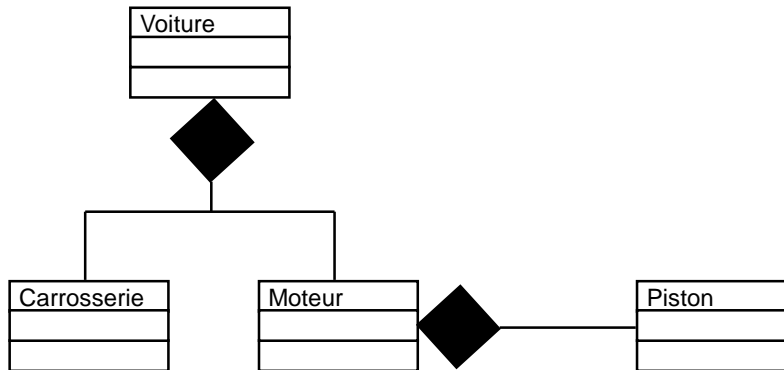
Agrégation

- A un même moment, une instance d'élément agrégé peut être liée à plusieurs instances d'autres classes
 - L'élément agrégé peut être **partagé**
- Une instance **ensemble** peut exister sans **élément** (et inversement)
 - Cycles de vies indépendants

44

Composition

- Les **cycles de vies** des composants et du composé sont **liés**
 - si le composé est détruit (ou copié), ses composants le sont aussi
- A un même moment, une instance de composant **ne peut être liée qu'à un seul composé**



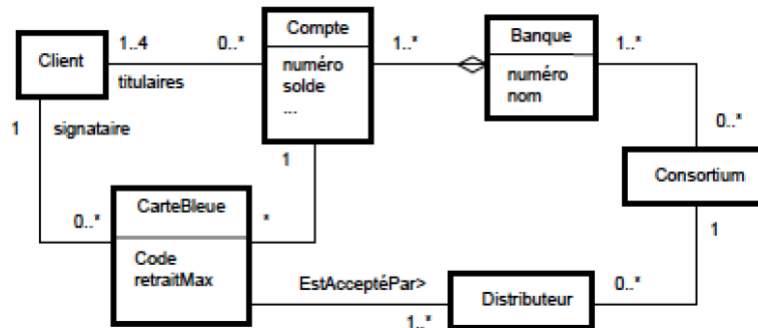
45

Association ? Agrégation ? Composition ?

- Questions à se poser
 - asymétrie et lien de subordination entre instances des deux classes (agrégation/composition) ou indépendance des objets (association) ?
 - création et destruction des parties avec le tout ? (composition)
- Dans le doute, toujours utiliser une association

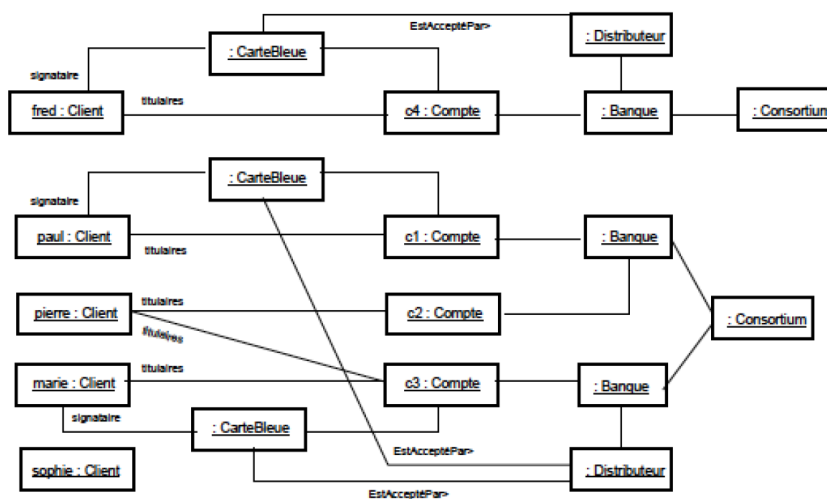
46

Exemple de diagramme de classe



47

Diagrammes d'objets



48

Diagramme de classes modèle du domaine

- Classes = objets du domaine
- Ignore les aspects temporels
- Démarche
 - Identifier les concepts du domaine
 - Les modéliser en tant que classes
 - Identifier les associations pertinentes entre les concepts (traduction de leur dépendance)
 - Proposer des multiplicités
 - Compléter le modèle avec des attributs

49

Diagramme de classes modèle du conception

- Pendant la conception/implémentation
- Classes = objets logiciels

50

Etude de cas – Enoncé

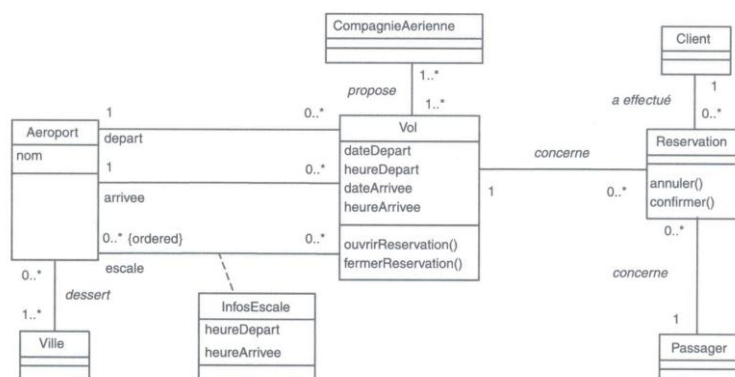
Système simplifié de réservation de vols pour une agence de voyage

- *Résumé de la connaissance du domaine :*
 1. des compagnies aériennes proposent différents vols
 2. un vol est ouvert à la réservation et refermé sur ordre de la cie
 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

Tiré des exemples de Pascal Roques

51

Diagramme de classes complet (ou presque)



52

Modélisation de la phrase 1

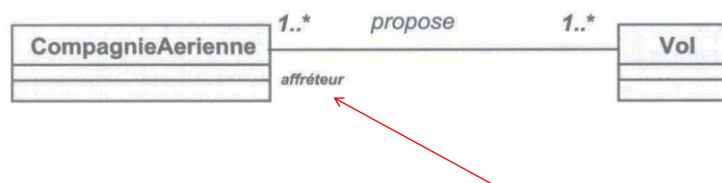
1. des **compagnies aériennes** proposent différents **vols**
- On recherche les concepts importants du monde réel avec des propriétés et des comportements
 - immédiat



53

Modélisation de la phrase 1

- Quelles multiplicité côté *compagnieAerienne*?
 - Un vol est proposé par une et une seule *cie* mais il peut être partagé par plusieurs affréteurs
 - *Affréteur*
 - Rôle joué par la classe *cie* dans l'association avec *vol*
 - Indique la façon dont un objet participe à l'association
 - Qd un rôle est donné, le nom de l'association peut être omis



54

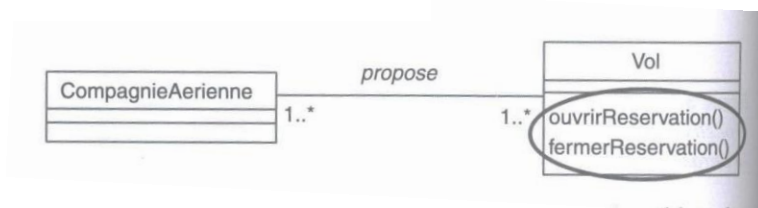
Modélisation de la phrase 2

2. un vol est ouvert à la réservation et refermé sur ordre de la *cie*

- l'ouverture et la fermeture de la réservation sont des concepts dynamiques
- Ce sont des actions
 - concept **d'opération**
 - *c'est la cie qui ouvre le vol et le vol qui est ouvert*
- on considère que
 - l'objet sur lequel on pourra réaliser un traitement doit le déclarer en tant qu'opération

55

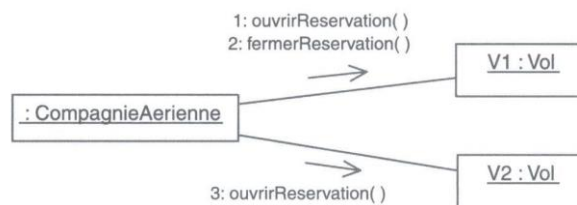
Modélisation de la phrase 2



56

Diagramme de collaboration

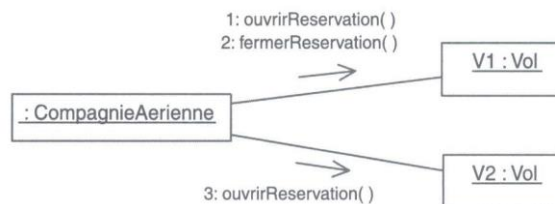
- Seuls concepts dynamiques : les opérations
- Diagramme de collaboration
 - Une autre façon de représenter l'interaction entre objets
 - Affiner un aspect délicat



57

Diagramme de collaboration

- Représentation des **objets**
 - désignation directe et explicite
monvol
 - la désignation inclut le nom de la classe
monvol : vol
 - désignation anonyme d'un objet d'une classe donnée
:vol



58

Modélisation de la phrase 2

- Une autre possibilité
 - Un attribut *état* avec deux valeurs possibles
 - Ouvert /fermé
 - mauvais choix
 - le vol et ses caractéristiques existent indépendamment de son ouverture et de sa fermeture

59

Diagramme de séquence versus Diagramme de collaboration

- Emphase sur la chronologie des envois de messages
- Emphase sur les interactions entre objets

60

Modélisation de la phrase 7

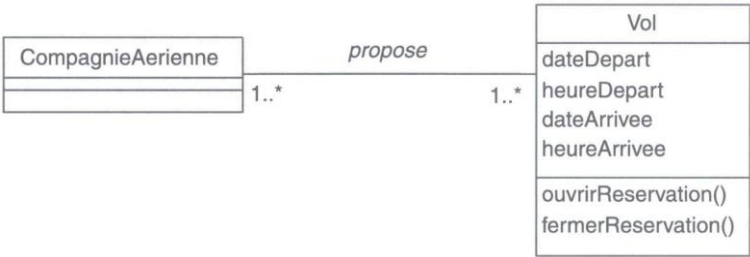
7. un vol a un jour et une heure de départ et un jour et une heure d'arrivée
- *date* et *heure* sont des attributs
 - Ils ont une valeur unique
 - Un élément dont on peut ne demander que sa valeur est un attribut

Vol
dateDepart
heureDepart
dateArrivee
heureArrivee
ouvrirReservation()
fermerReservation()

Un objet qui a des propriétés et un comportement est une instance de classe

61

Modélisation de la phrase 7



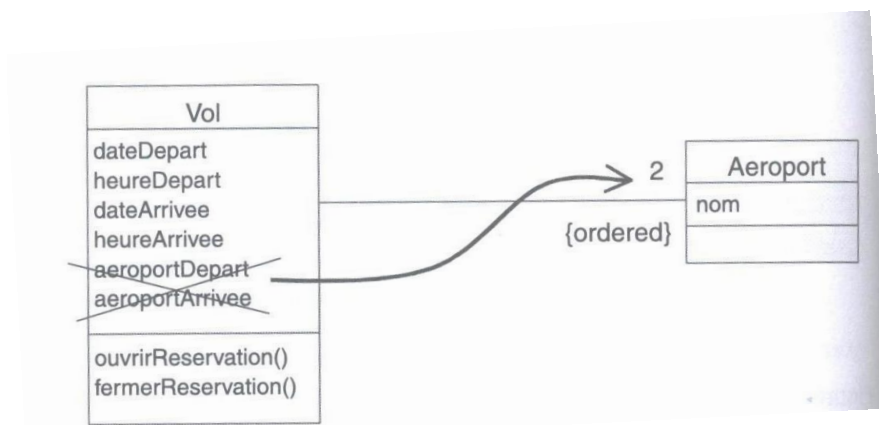
62

Modélisation de la phrase 6

6. un vol a un aéroport de départ et un aéroport d'arrivée
- *aéroport* est une classe
 - il possède un nom, une capacité, il dessert des villes etc.
 - 3 solutions

63

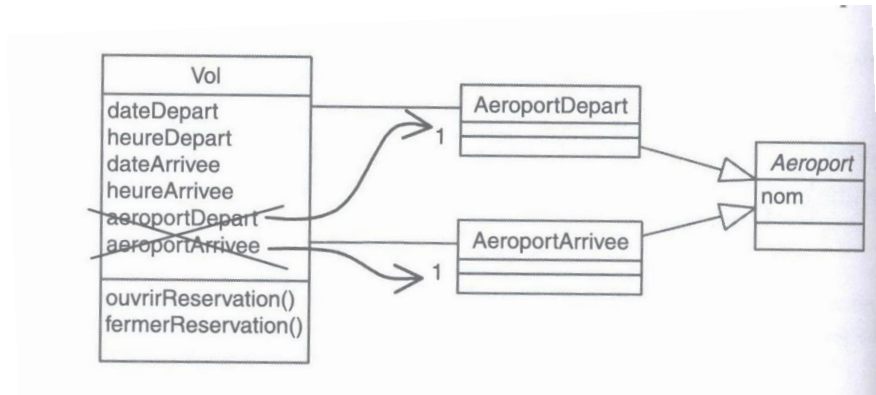
Solution 1



Les deux aéroports sont liés à un vol et ordonnés (le départ avant l'arrivée)
Est-ce parlant ?

64

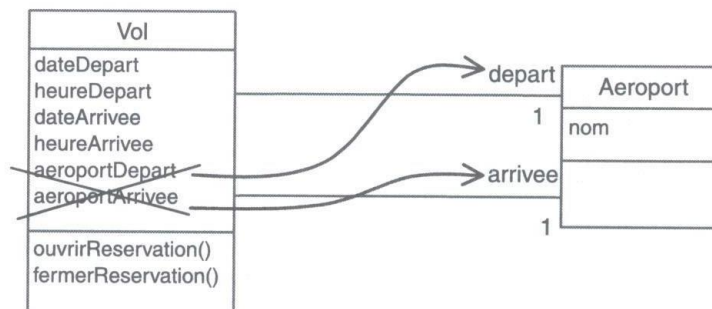
Solution 2



Ces deux sous classes ont exactement les mêmes instances
 Tout aéroport est de départ pour certains vols et de retour pour d'autres
 → Très mauvais choix

65

Solution 3



Le rôle distingue leur usage dans l'association

66

Modélisation de la phrase 6

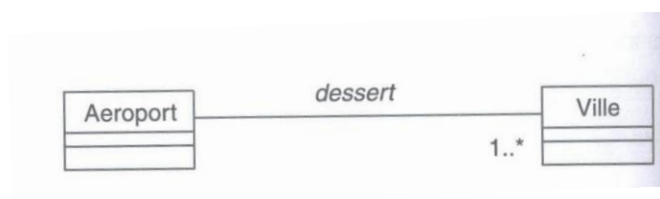
6. un vol a un aéroport de départ et un aéroport d'arrivée
- *aéroport* est une classe
 - il possède un nom, une capacité, il dessert des villes etc.
 - solution 1 : une *association* avec une **multiplicité** de 2, **ordonnée**
 - solution 2 : création de deux **sous-classes** de la classe *aéroport*
 - solution 3 : deux associations, un **rôle** pour chacune

67

Modélisation de la phrase 10

10. Chaque aéroport dessert une ou plusieurs villes

- immédiat



68

Modélisation de la phrase 10

10. Chaque aéroport dessert une ou plusieurs villes

- quelle multiplicité côté *aéroport* ?
- par combien d'aéroports une ville est-elle desservie ?

Desservir?

désigner le moyen de transport par les airs le plus proche
une ville est desservie par un et un seul aéroport



69

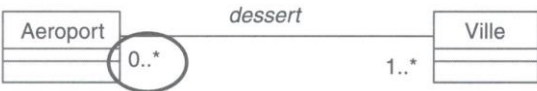
Modélisation de la phrase 10

10. Chaque aéroport dessert une ou plusieurs villes

- quelle multiplicité côté *aéroport* ?
- par combien d'aéroports une ville est-elle desservie ?

Desservir ?

désigner tout moyen de transport par les airs à moins de 30km
une ville est desservie par 0 ou plusieurs aéroports

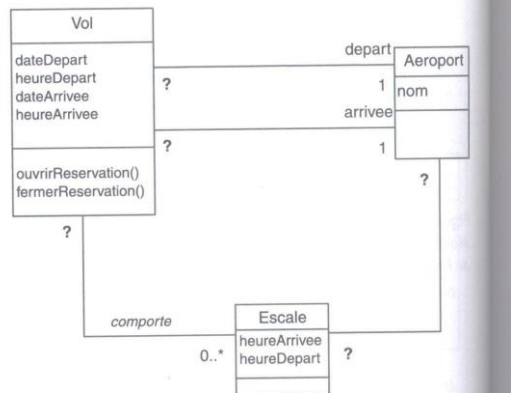


70

Modélisation des phrases 8 et 9

8. un vol peut comporter des escales dans des aéroports
 9. une escale a une heure d'arrivée et une heure de départ

Création
d'une classe *escale*
entre *vol* et *aéroport*

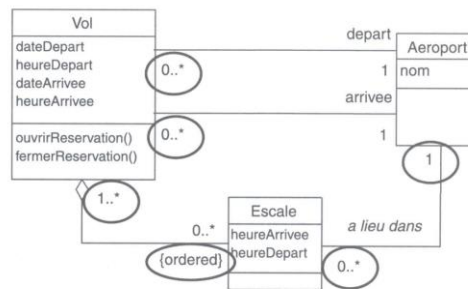


Modélisation des phrases 8 et 9

8. un vol peut comporter des escales dans des aéroports
 9. une escale a une heure d'arrivée et une heure de départ

- **Escale / Aéroport**
 - Une escale a lieu dans un et un seul aéroport
 - Un aéroport peut servir à plusieurs escales
- **Aéroport / Vol**
 - Un aéroport peut servir de départ et d'arrivée à plusieurs vols
- **Escale / Vol**
 - Une escale n'appartient pas à un seul vol

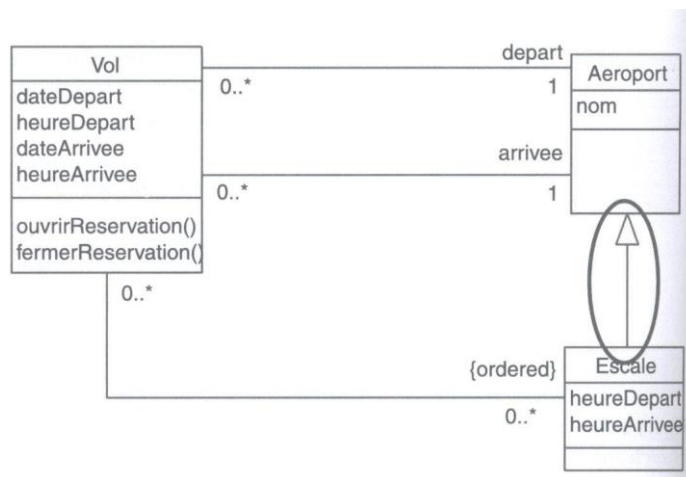
Modélisation des phrases 8 et 9



Attention en plus pas le même horaire d'arrivée pour tous

73

Pourquoi pas ?



Escale a peu de réalité par elle-même
Elle existe par rapport à un vol et est fortement liée à un aéroport

74

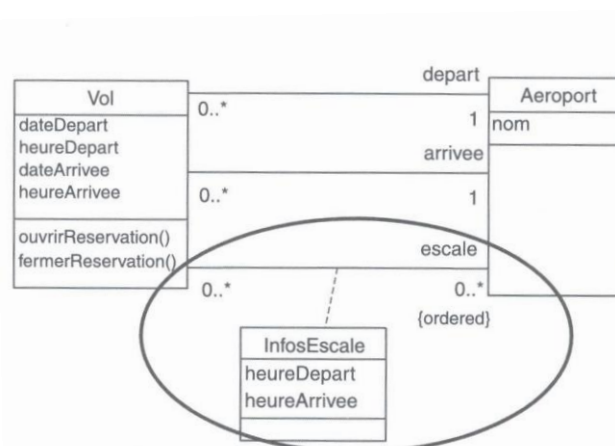
Modélisation des phrases 8 et 9

- 8. un vol peut comporter des escales dans des aéroports
- 9. une escale a une heure d'arrivée et une heure de départ

- Escale / Vol : Solution 2
 - *escale* un rôle joué par un *aéroport* par rapport à un *vol*
 - on crée une classe d'association *infosEscale*

75

Solution 2



Les attributs dépendent fonctionnellement des deux classes

76

Modélisation des phrases 3, 4 et 5

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

- Distinguer le concept de client de celui de passager ?
- Le concept de client est lié à une notion de facturation

77

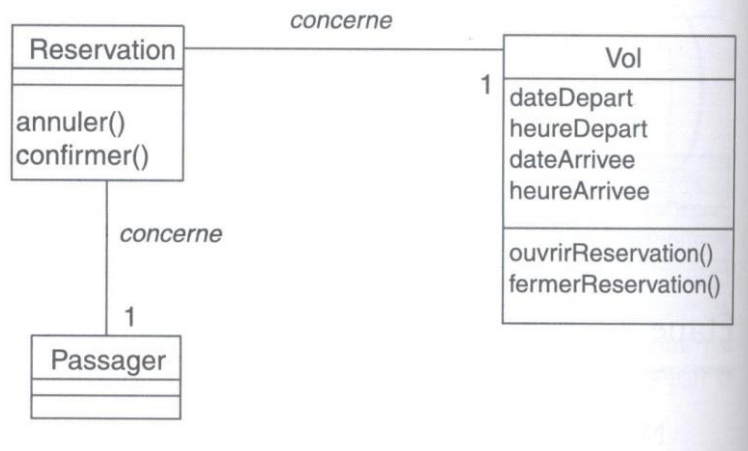
Modélisation des phrases 3, 4 et 5

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

- Phrase 3
 - Un client peut effectuer plusieurs réservations, chacune portant sur un vol
- Phrase 4
 - Immédiat
- Phrase 5
 - concept d'opération

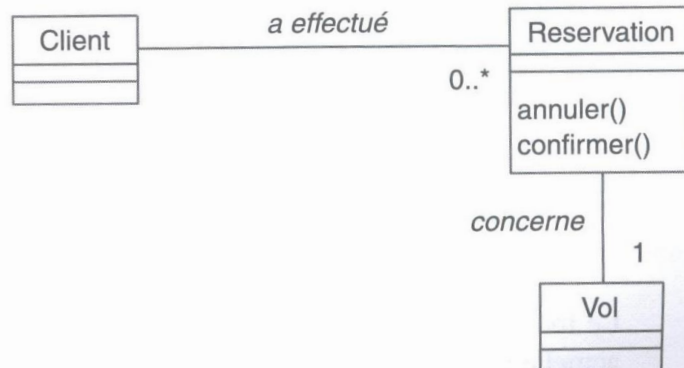
78

Modélisation des phrases 3, 4 et 5



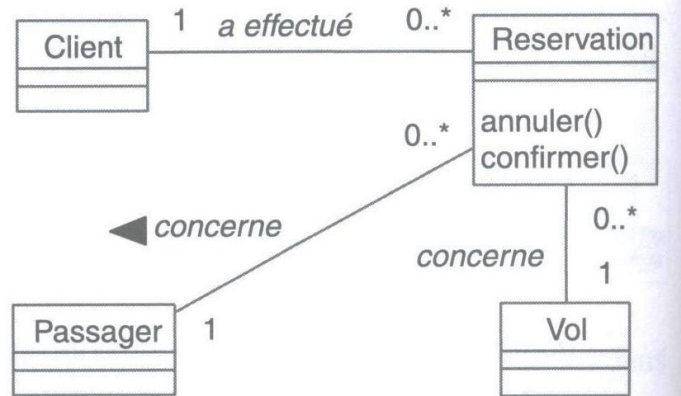
79

Modélisation des phrases 3, 4 et 5



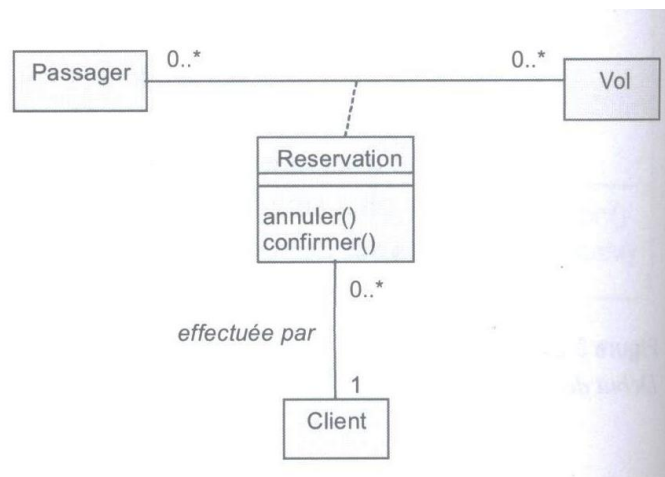
80

Modélisation des phrases 3, 4 et 5



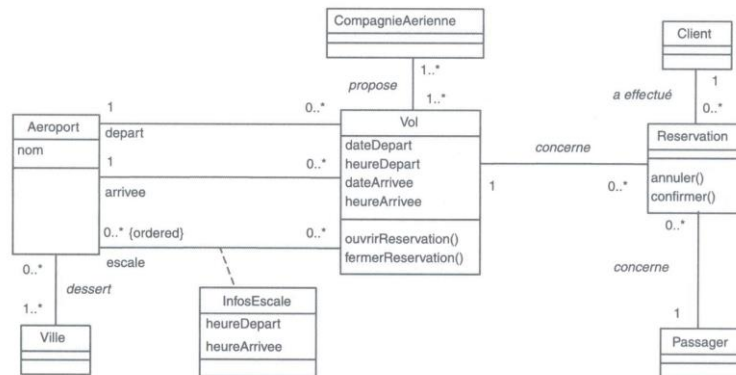
81

Modélisation des phrases 3, 4 et 5 autre choix



82

Diagramme de classes complet (ou presque)



83

Ajout d'attributs

- Aéroport : *nom*
- Client : *nom, prénom, adresse, numéro de téléphone, numéro de fax*
- Cie : *nom*
- InfosEscale : *heure départ, heure d'arrivée*
- Passager : *nom, prénom*
- Réservation : *date, numéro*
- Ville : *nom*
- Vol : *numéro, date arrivée, heure d'arrivée, date départ, heure départ*

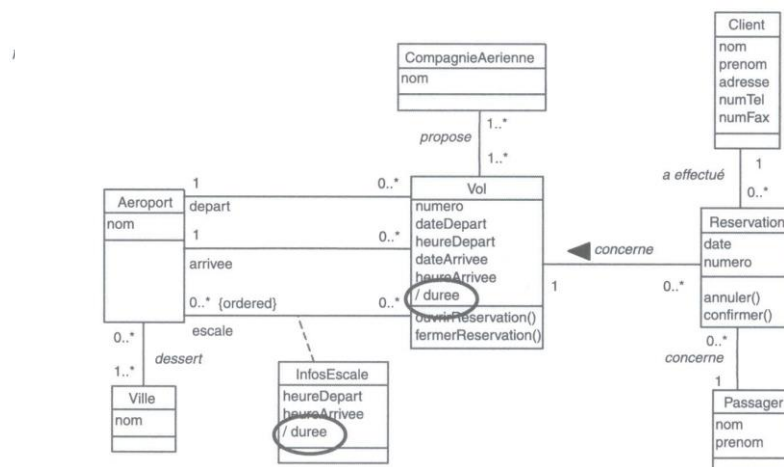
84

Ajout d'attributs

- Un **attribut dérivé** est une propriété évaluée intéressante pour l'analyste, mais redondante car sa valeur peut être déduite d'autres informations disponibles dans le modèle
 - Valeur constamment disponible
 - Calcul complexe et coûteux

85

Attributs



86

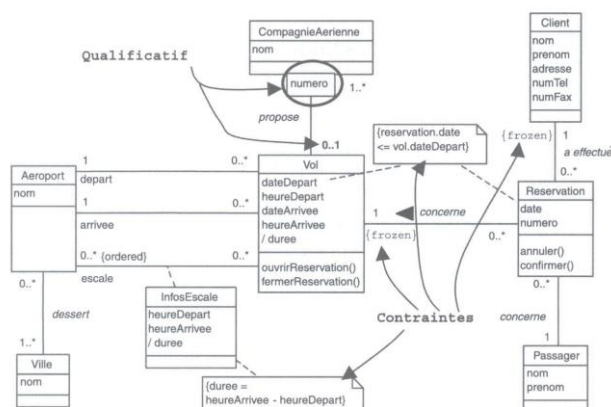
Contraintes

- Répertorier les principales sur le diagramme de classes
 - Sous la forme de note, entre {}
`{Reservation.date <= Vol.datedep }`
 entre *Vol* et *Réservation*
 - pour qu'un lien entre deux objets ne puisse jamais être modifié après sa création `{frozen}`
 - Pour changer de vol ou de client, il faut annuler la réservation en question et en créer une nouvelle

87

Contraintes - Qualificatif

La combinaison d'une cie et d'un vol génère au plus au numéro



Chaque vol est identifié de façon unique par un numéro propre à la cie

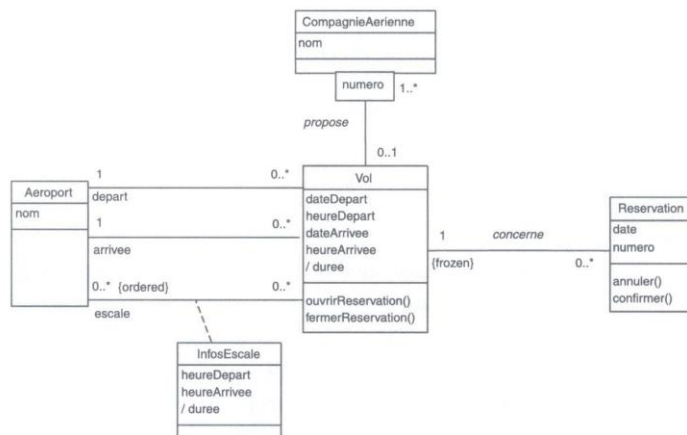
88

Utilisation de patterns d'analyse

- *Améliorer le modèle*
- La classe *Vol* a beaucoup de responsabilités
- Cela viole le principe de forte cohésion
 - Elle contient tout ce qui se retrouve dans les catalogues des cies aérienne
 - par exemple les vols génériques qui reviennent à l'identique toutes les semaines ou presque
 - *Il existe un vol Toulouse Paris-Orly sans escale tous les lundi matin à 7h10 proposé par Air France*
 - Elle contient tout ce qui concerne les réservations, par exemple
 - *On réserve le vol Toulouse Paris-Orly sans escale, départ, 7h10 proposé par Air France le 14 janvier 2002*

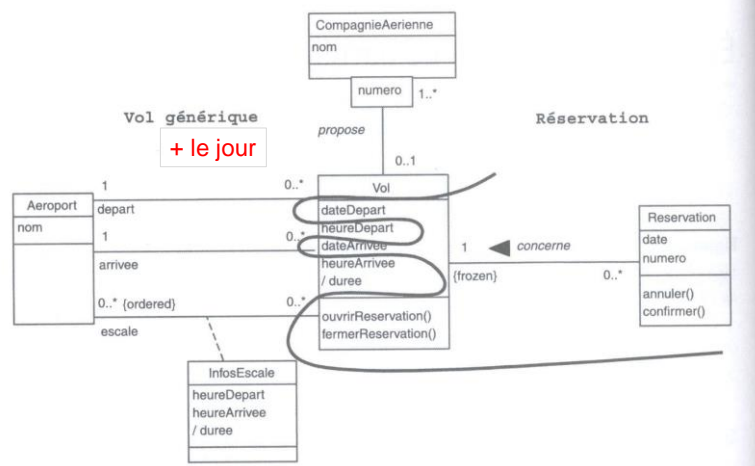
89

Analyse de ce fragment



90

Analyse de ce fragment



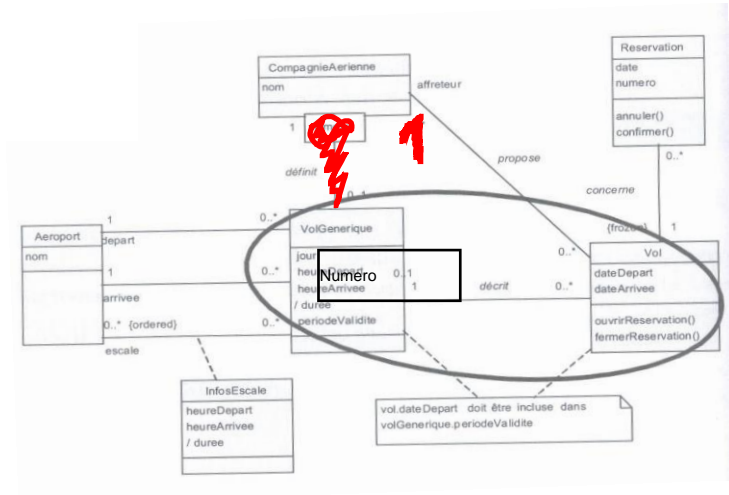
91

Utilisation de patterns d'analyse

- On sépare ces deux aspects
 - Création d'une classe *VolGénérique*
 - Partage des attributs et opérations et associations
 - + jour semaine, période de l'année disponible pour *VolGénérique*
- On parle de **méta classe**
- Le vol générique est décrit avec des propriétés identifiées pour plusieurs vols réels
- Si un aéroport est fermé pour travaux, aucun vol réel ne sera créé mais le vol générique ne sera pas touché

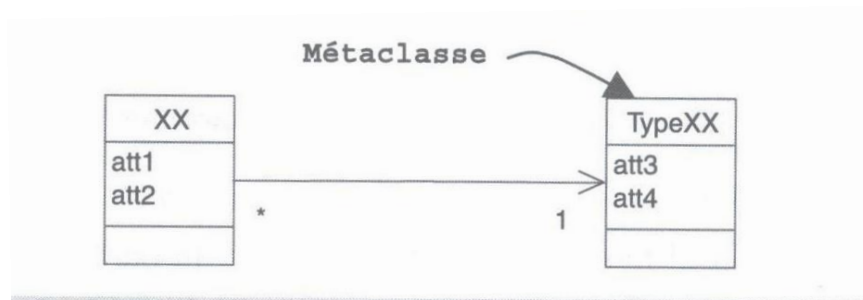
92

Utilisation de patterns d'analyse



93

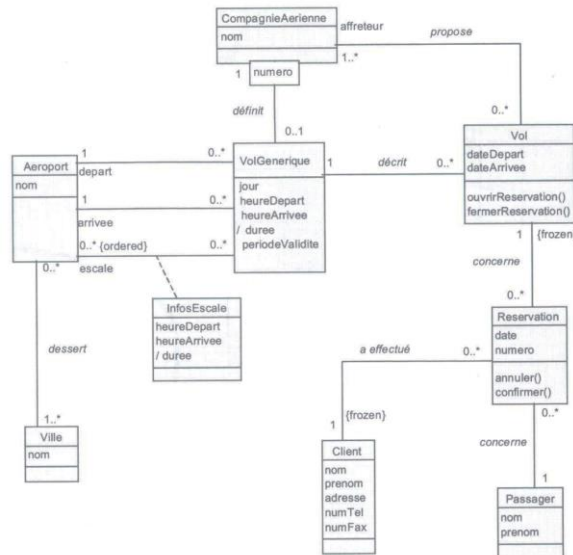
Principe



On parle de méta classe car
elle contient des informations décrivant une autre classe

94

Modèle d'analyse



95

Structuration en paquetages

- Vers une meilleure convivialité / Réutilisation
- Eléments d'organisation des modèles
 - Regroupement d'éléments de modélisation
- Le découpage du modèle doit se faire en respectant
 - Cohérence
 - Regrouper les classes proches d'un point de vue sémantique
 - Indépendance
 - Minimiser les dépendances entre paquetages

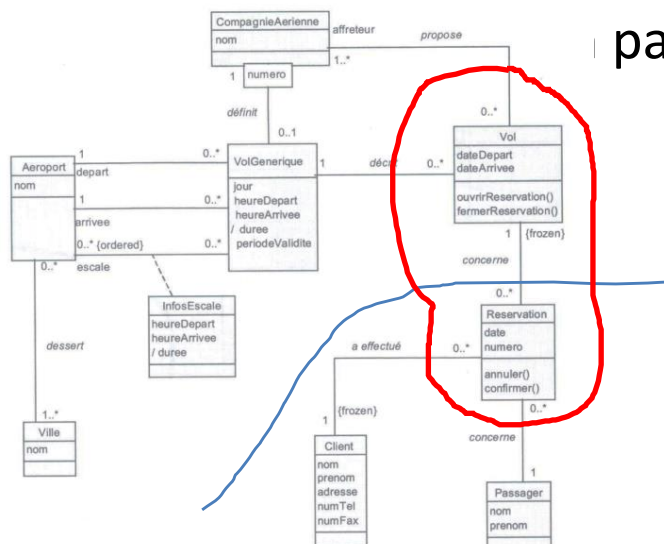
96

Structuration en paquetages

- Découpage en deux **paquetages**
 - L'un pour la définition des vols, stables dans le temps
 - L'un pour les réservations
 - Discussion sur l'association (ou les associations) qui traverse(nt) la frontière des paquetages
 - *Ne pas conserver de dépendance mutuelle entre paquetages*

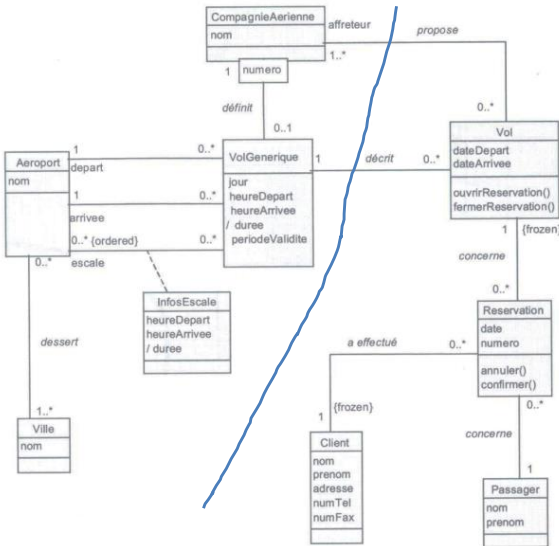
97

Structuration paquetages



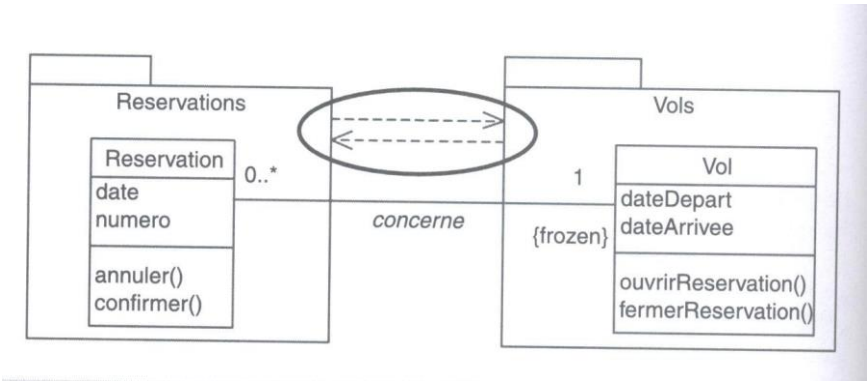
98

Structuration en paquetages



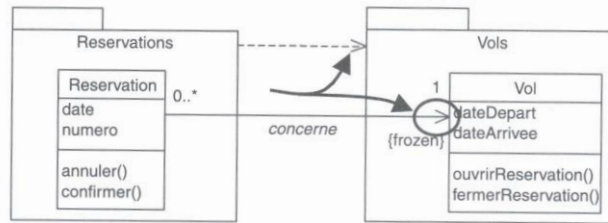
99

Paquetage – structuration 1



100

Structuration 1

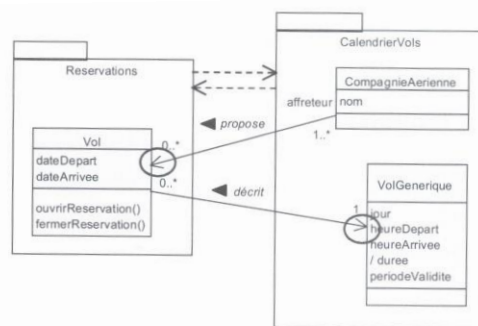


Une seule association mais elle provoque une dépendance mutuelle
On privilégie un sens de navigation

- une réservation n'existe pas sans vol
- un vol existe en lui-même

101

Structuration 2



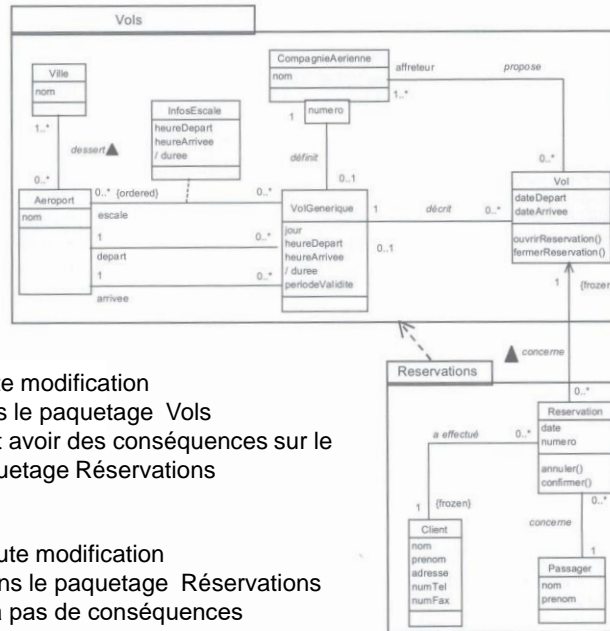
Le vol est décrit par VolGenerique
Le Vol générique existe en lui-même

Phrase 2 : navigabilité obligatoire de *cie* vers *vol*, cf. diag de collaboration
Le vol est ouvert par le compagnie

→ 2 associations différentes, navigables en sens inverse

102

Bilan



Toute modification
dans le paquetage Vols
peut avoir des conséquences sur le
paquetage Réservations

Toute modification
dans le paquetage Réservations
n'a pas de conséquences
sur le paquetage Vols

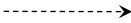

103

Paquetages

- Les paquetages sont des éléments d'organisation des modèles
- Ils regroupent des éléments de modélisation, selon des critères purement logiques
- Ils permettent d'encapsuler des éléments de modélisation
- Ils servent de "briques" de base dans la construction d'une architecture
- Ils représentent le bon niveau de granularité pour la réutilisation

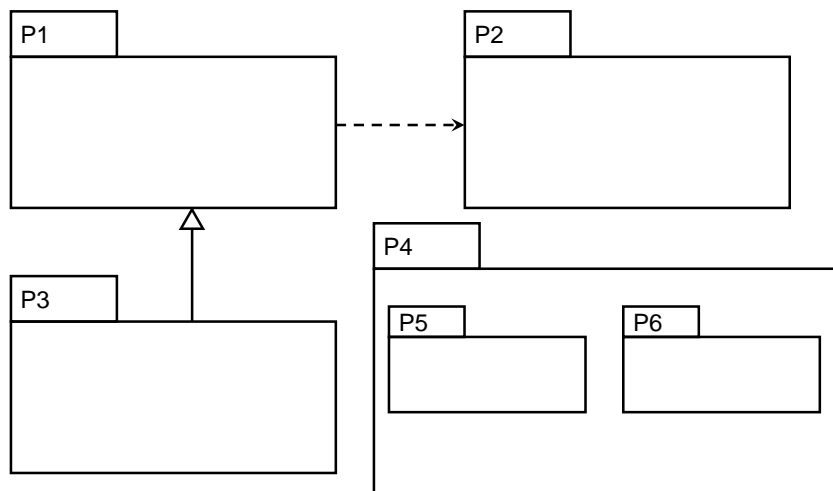
104

Relation entre paquetage

- Dépendance : 
 - Au moins un élément du paquetage source utilise les services d'au moins un des éléments du paquetage destination
- Héritage : 
 - Au moins un élément du paquetage source spécialise (est dérivée d') au moins un des éléments du paquetage destination

105

Convention graphique



106

D'un schéma UML vers un schéma relationnel

- Chaque classe du diagramme UML devient une relation. Il faut choisir un attribut de la classe pour jouer le rôle de la clé
- Associations
 - 1,*
 - ajouter comme attribut de type clé étrangère dans la relation correspondant au côté *, la clé de la relation correspondant au côté 1

107

D'un schéma UML vers un schéma relationnel

- Associations
 - 1,*
 - *,* et classes association
 - l'association devient une relation
 - sa clé : les clés des relations associées aux classes participant à l'association (chacune est clé étrangère)
 - s'il s'agit d'une classe association et qu'elle possède des attributs, ces attributs complètent la relation

108

D'un schéma UML vers un schéma relationnel

- Associations
 - 1,*
 - *,* et classes association
 - 1,1
 - ajouter comme attribut de type clé étrangère, dans une des deux relations correspondant au côté 1, la clé de la relation correspondant à l'autre côté
 - le choix se fait en regardant les multiplicités minimales

109

D'un schéma UML vers un schéma relationnel

- Associations d'héritage
 - Chaque sous-classe devient une relation
 - Sa clé est celle la relation associée à la classe dont elle a été dérivée

110

Classes d'analyse

- Principes d'identification des classes, attributs et associations
- Les diagrammes de classes d'analyse représentent des choses, des concepts et des idées du monde réel
 - Exemples : Vol, Réservation, Client, ..
- Un concept fait partie du domaine de l'analyse
 - S'il est nécessaire à la compréhension du problème
 - S'il répond à des exigences fonctionnelles visibles par un utilisateur
- Identifier les classes candidates dans les cas d'utilisation
 - Chercher les noms communs et les groupes nominaux
 - Vérifier les propriétés « objet » de chaque concept
 - Identité, attributs, opérations, états, ...

111

Conclusion

- Pour un modèle complexe, plusieurs diagrammes de classes complémentaires doivent être construits
- On peut par exemple se focaliser sur :
 - les classes qui participent à un cas d'utilisation
 - les classes associées dans la réalisation d'un scénario précis
 - les classes qui composent un paquetage
 - la structure hiérarchique d'un ensemble de classe

112