Exercice Web

IDENTIFICATION DES CONCEPTS

Trouver pour le cas d'utilisation *Rechercher des ouvrages*, les concepts fondamentaux (*ou classes*).

SOLUTION

- Ouvrage
- Auteur
- Éditeur

IDENTIFICATION DES CONCEPTS

Trouver pour le cas d'utilisation *Gérer son panier*, les concepts fondamentaux.

SOLUTION

- Panier
- Livre

IDENTIFICATION DES CONCEPTS

Trouver pour le cas d'utilisation *Effectuer une commande*, les concepts fondamentaux.

SOLUTION

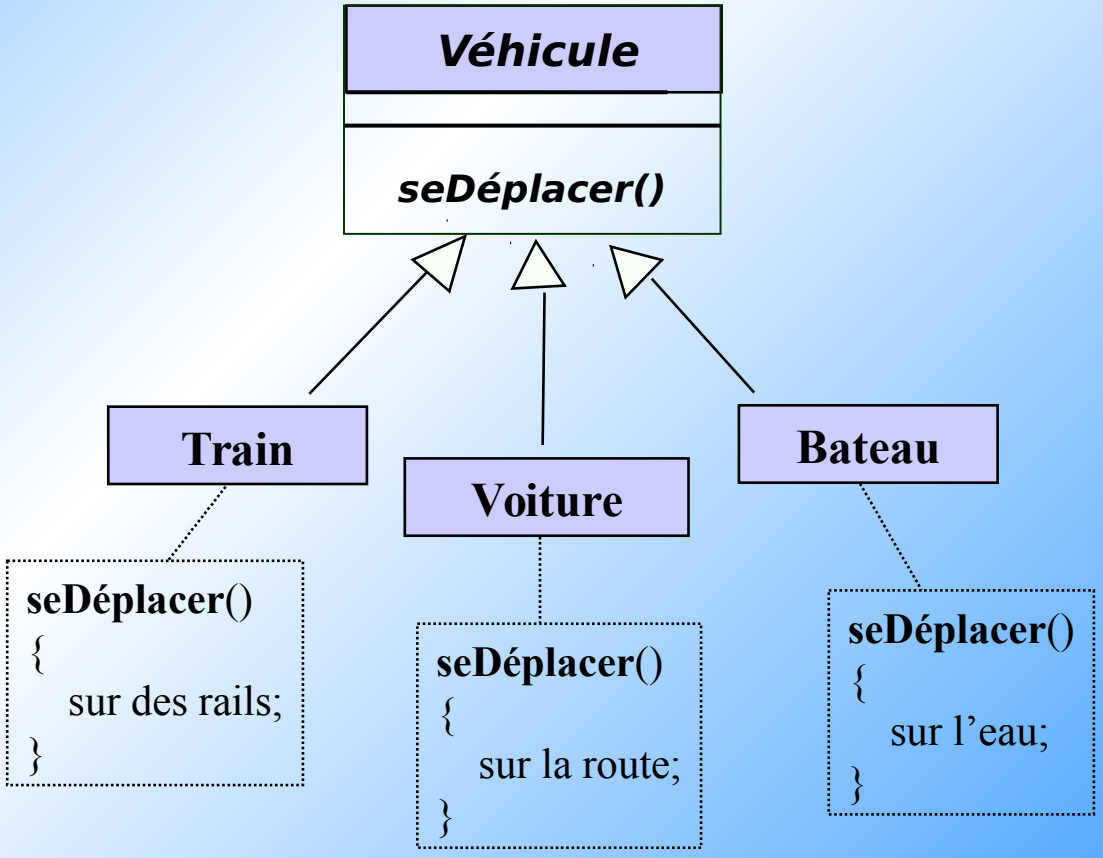
- Commande
- Panier
- Client
- CarteDeCrédit

POLYMORPHISME

Exemple

```
Véhicule convoi[3] = {
            Train("TGV"),
            Voiture("Corsa"),
            Bateau("Titanic")
            };

for (int i = 0; i < 3; i++)
{
  convoi[i].seDéplacer();
}
```



Associations et attributs site Web

Rechercher des ouvrages

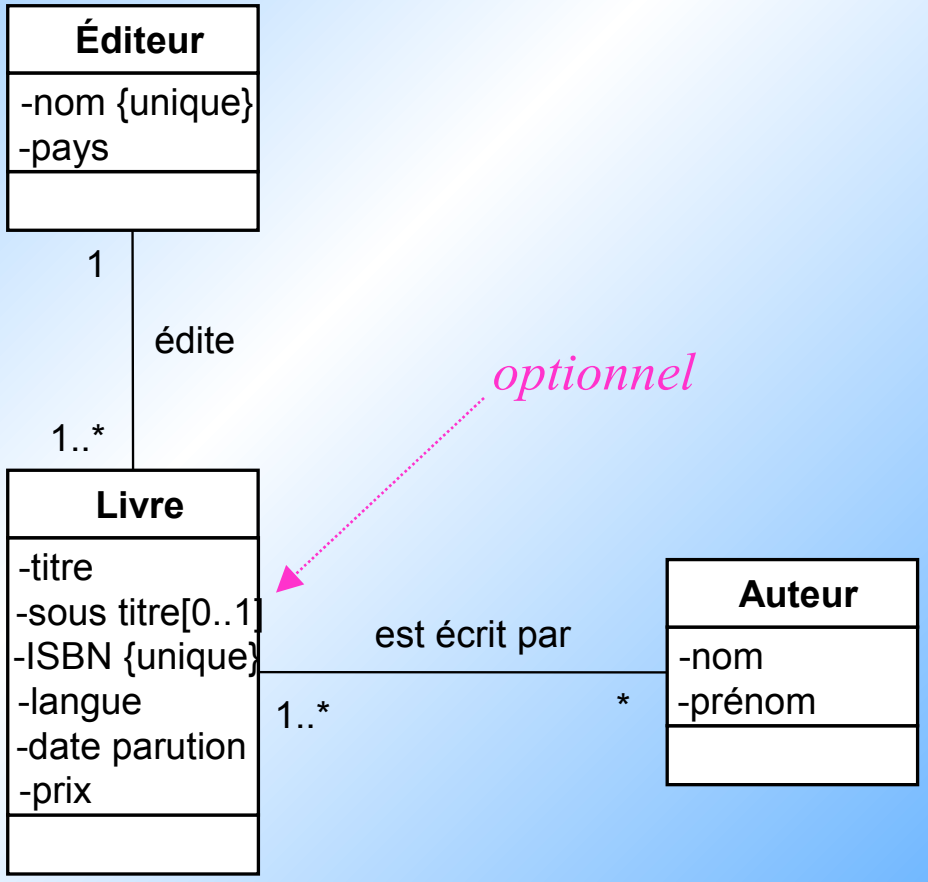
L'internaute peut saisir un critère (*titre*, *auteur*, *ISBN*, etc.) ou même plusieurs critères à la fois.

Les copies d'écran nous montrent les attributs *prix*, *date de parution*, *éditeur*, *langue*, *sous-titre* et *nombre de pages*.

Faire une première ébauche du modèle du domaine en utilisant les concepts Éditeur, Livre et Auteur

SOLUTION

Rechercher des ouvrages



SOLUTION

Rechercher des ouvrages

- L'attribut *sousTitre* est optionnel : tous les livres n'ont pas un sous-titre alors qu'ils ont tous un *titre*, une *langue*, etc.

- UML nous permet d'indiquer ce caractère optionnel en ajoutant une multiplicité [0..1] derrière l'attribut.

Associations et attributs site web

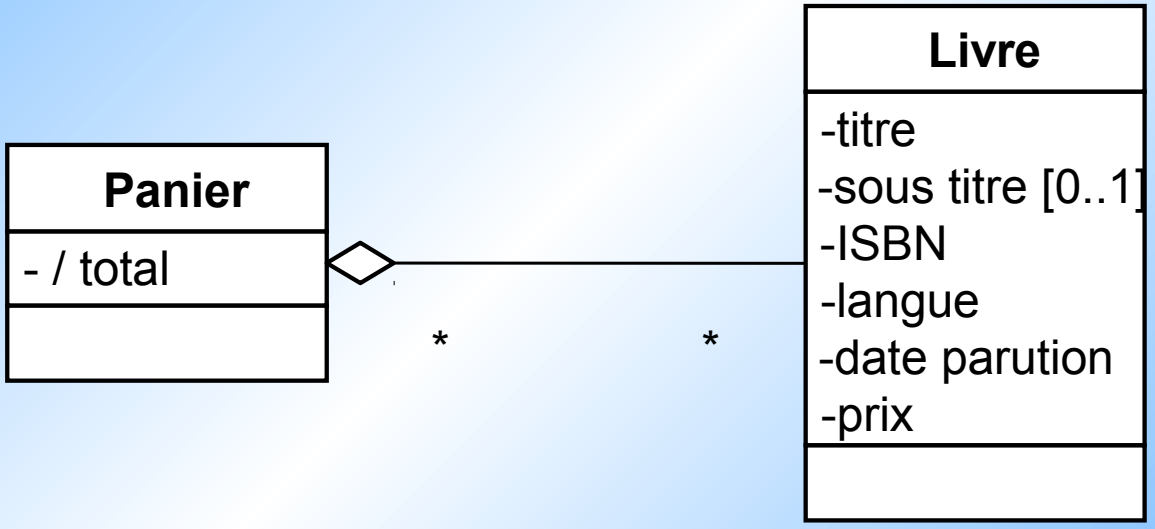
Gérer son panier

Le concept de panier est un concept du domaine car, dans les librairies réelles, le client remplit également un panier avant de passer à la caisse.
Le panier est simplement un conteneur (*vide au départ*) des livres sélectionnés par le client.

Faire le modèle du domaine
en utilisant au moins les concepts
Panier et Livre

SOLUTION

Gérer son panier



Associations et attributs site web

Gérer son panier

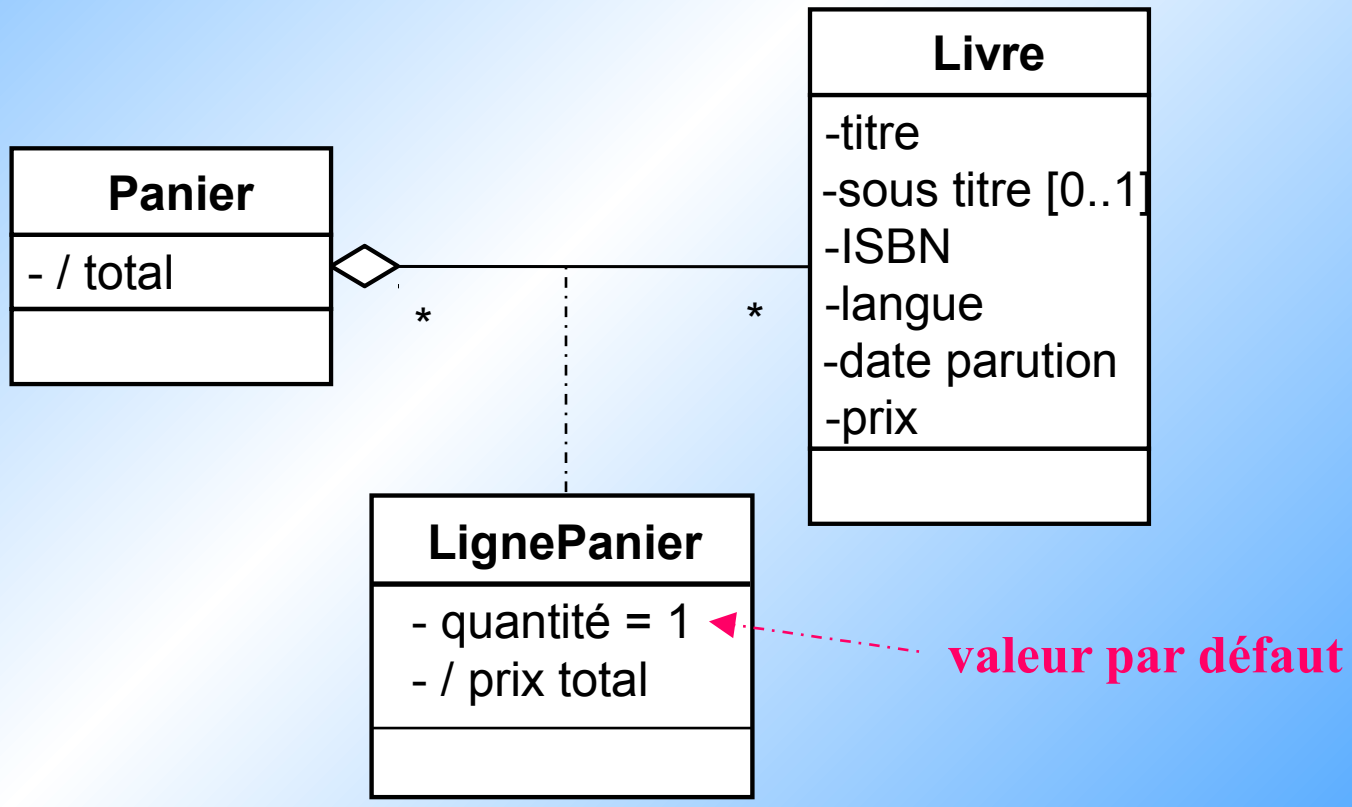
On doit prendre en compte le fait que le client peut choisir plusieurs exemplaires du même livre et qu'on veut le total du panier.

Compléter le modèle du domaine précédent

SOLUTION

Première solution

Introduire une classe d'association qui puisse porter l'attribut *quantité*.



Associations et attributs site web

Gérer son panier

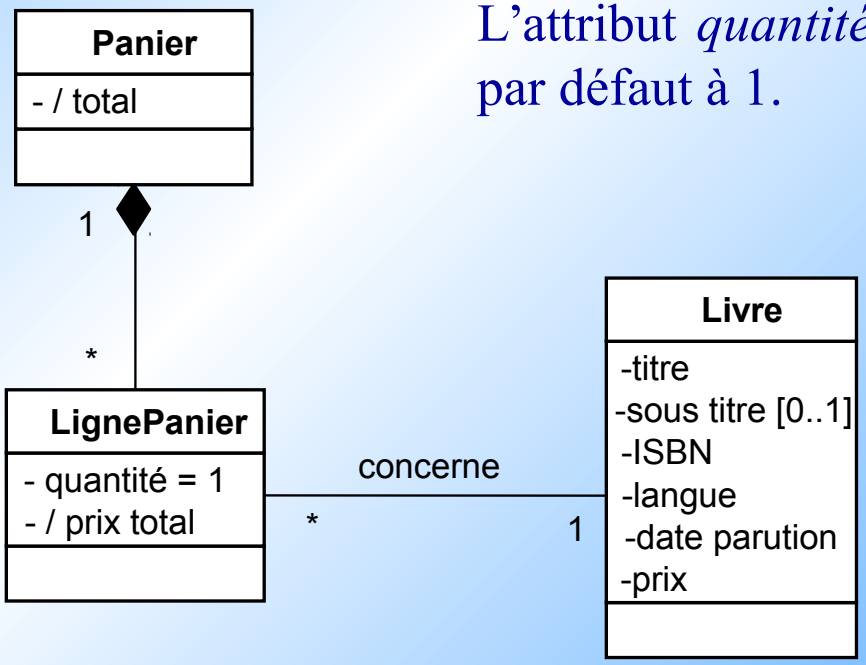
Quel est le lien entre le Panier et la LignePanier ?

Modifier le modèle précédent

SOLUTION

Gérer son panier

Ajouter un concept intermédiaire qui correspond à une ligne du panier et qui concerne donc un seul livre, mais avec un attribut quantité.



L'attribut *quantité* est positionné par défaut à 1.

SOLUTION

Gérer son panier

composition entre Panier et LignePanier :

1- une lignePanier ne peut appartenir qu'à un seul panier (*agrégation non partagée*) ;

2- la destruction du panier entraîne la destruction de toutes ses lignes (*le composite est responsable du cycle de vie des parties*).

Associations et attributs site web

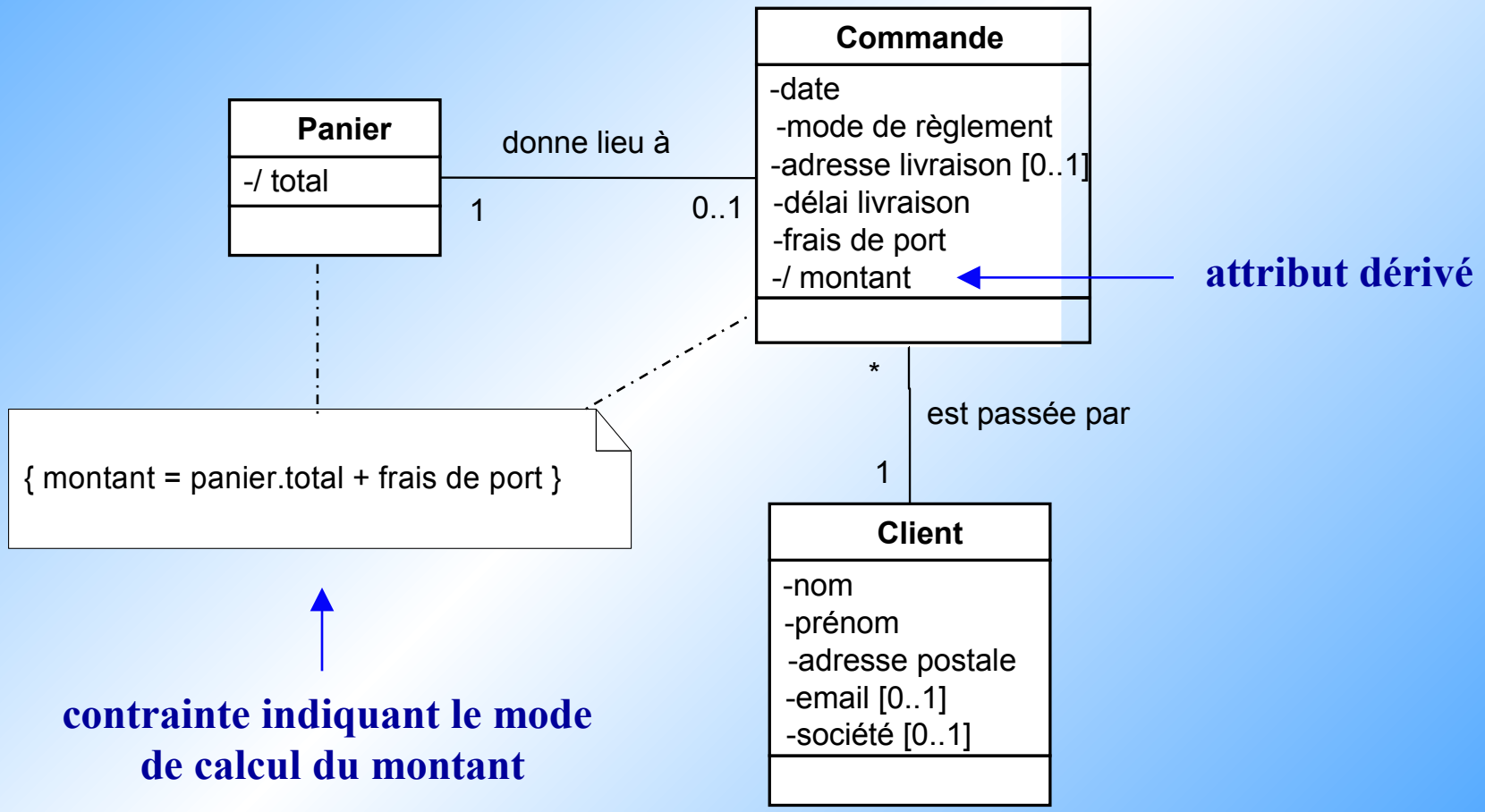
Effectuer une commande

Une fois que le client a un panier non vide, il peut passer sa commande.
Il lui faut pour cela saisir ses coordonnées et les informations nécessaires au paiement et à la livraison.

Faire le modèle
en se limitant, dans un premier temps,
aux concepts Panier, Commande et Client

SOLUTION

Effectuer une commande



Panier

PANIER › IDENTIFICATION › ADRESSES › PAIEMENT › CONFIRMATION

› Votre Panier : 1 article pour un achat maintenant

Articles	Quantité	Prix Unitaire	Montant	
Fouille de données complexes O. Boussaid, P. Gançarski, F. Masseglia, B. Trousse, G. Venturini, D. Zighed / Cépaduès	1	33,25 EUR	33,25 EUR	Mettre de côté Supprimer
TOTAL	1 articles		33,25 EUR	

Associations et attributs site web

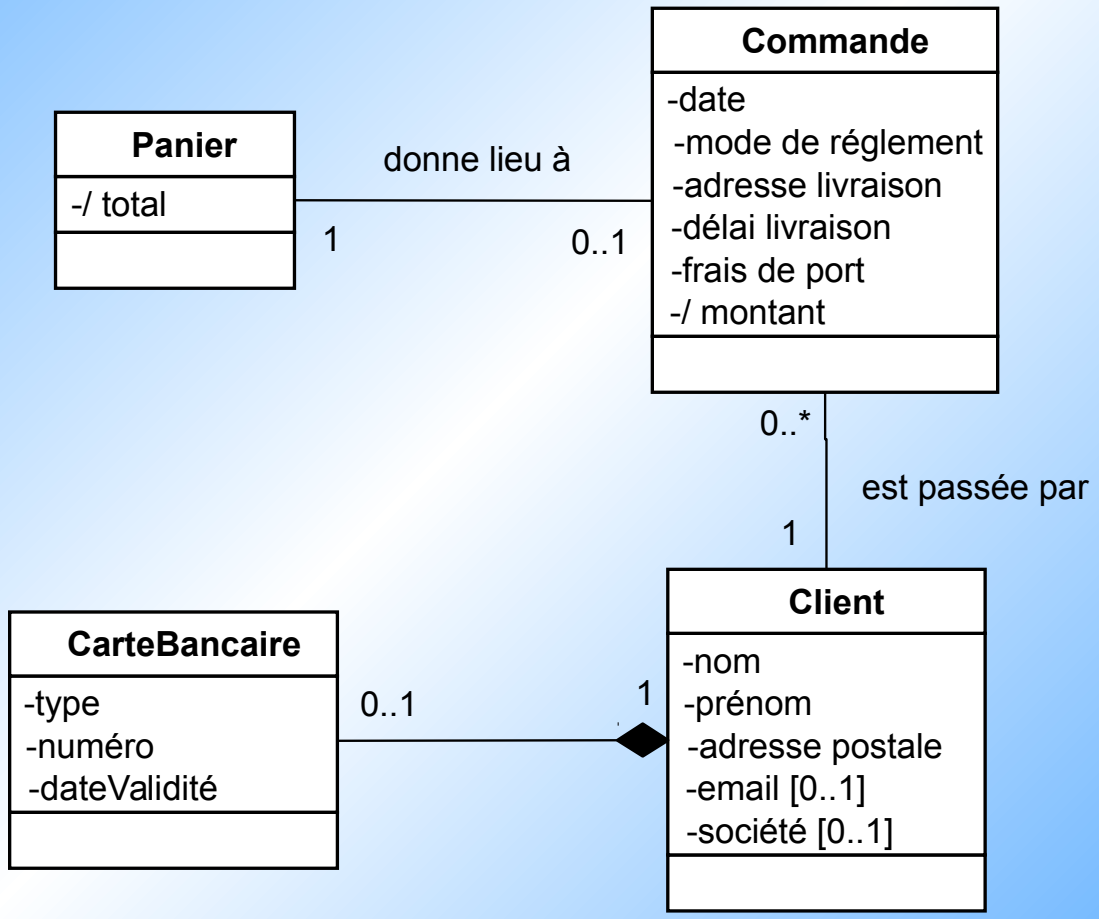
Effectuer une commande

On considère par la suite que le mode de règlement privilégié est la carte bancaire.

Compléter le modèle du domaine

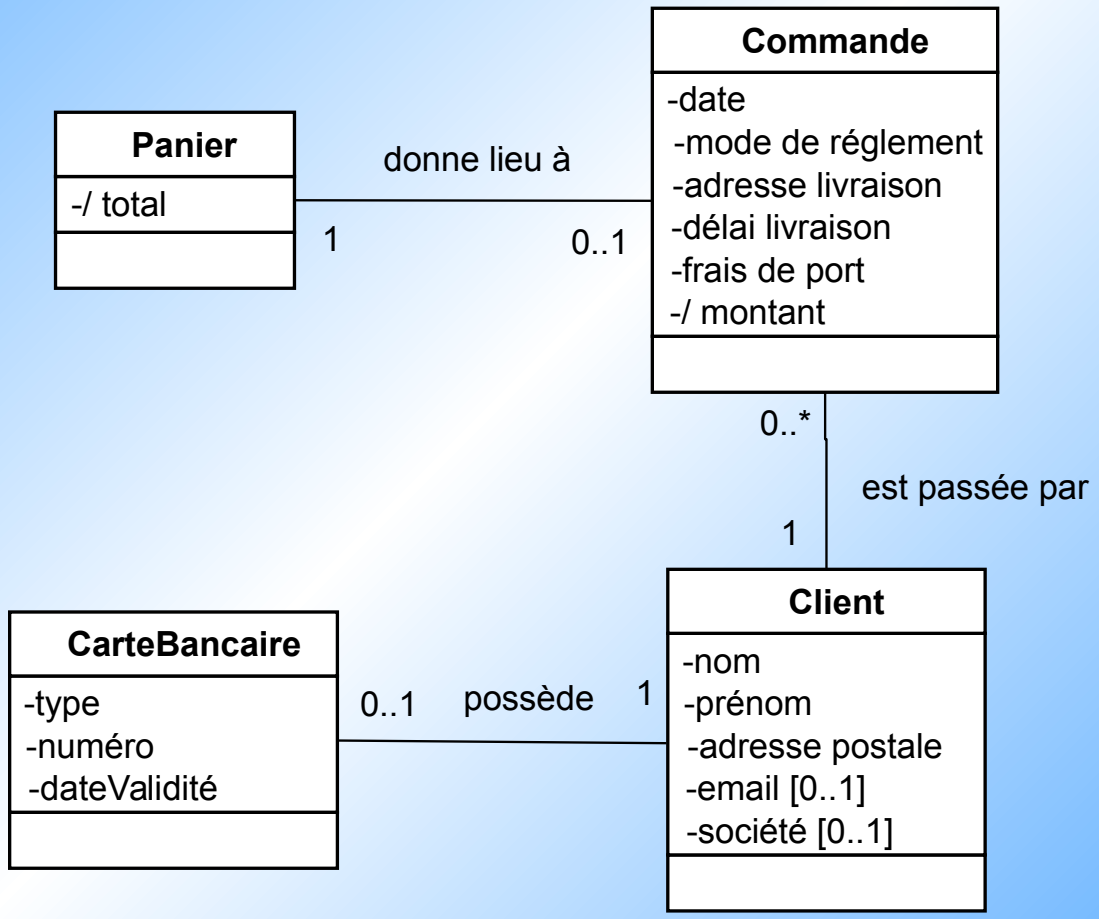
SOLUTION

Effectuer une commande



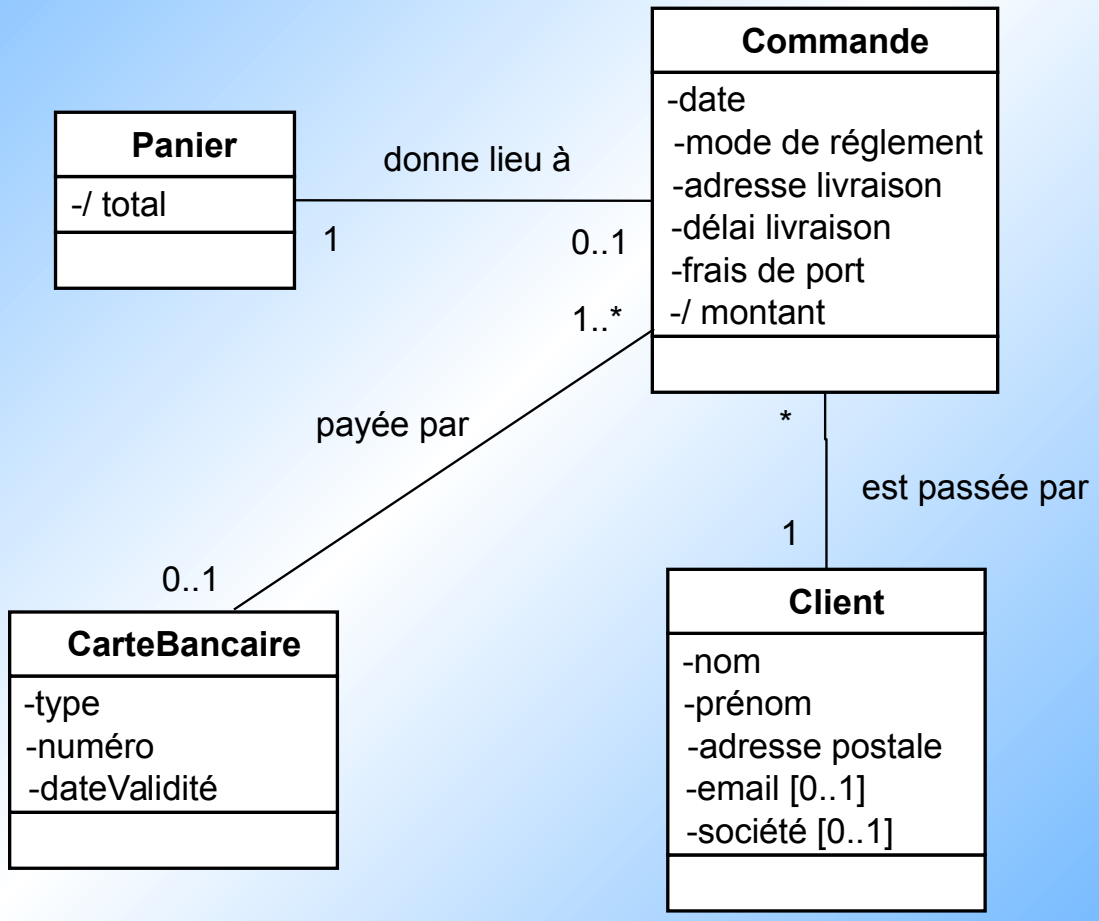
SOLUTION

Effectuer une commande



SOLUTION

Effectuer une commande



SOLUTION

Consulter ses commandes en cours

- La consultation des commandes en cours par le client ne fait pas intervenir de nouveau concept par rapport à la figure précédente.

- Par contre, on ajoute un attribut *motDePasse* dans la classe Client afin de sécuriser les consultations.

Associations et attributs site web

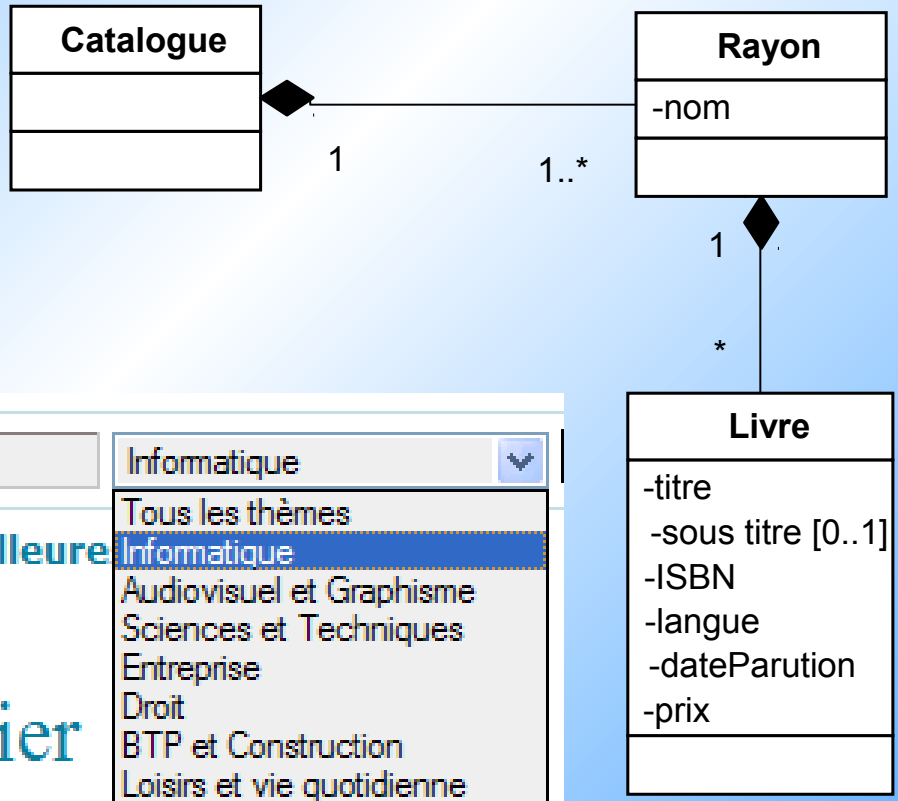
Maintenir le catalogue

La librairie *jeBouquine* a déjà ouvert un certain nombre de rayons bien séparés. Les livres sont donc classés en rayons au sein du catalogue.

Compléter le modèle du domaine

SOLUTION

Maintenir le catalogue



Associations et attributs site web

Maintenir le catalogue

On souhaite utiliser la notion de **thème**. Les livres peuvent appartenir à plusieurs thèmes car ceux-ci ne sont pas forcément disjoints.

Compléter le modèle du domaine

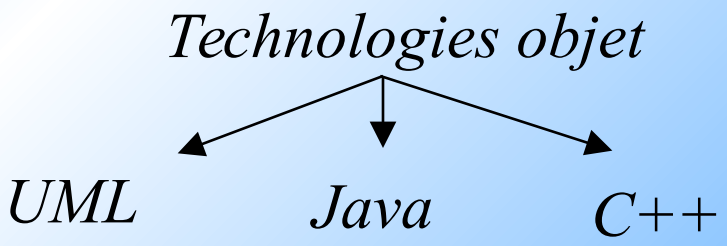
ASSOCIATIONS ET ATTRIBUTS SITE WEB

Exemple :
Un livre comme *"Bases de données avec UML"* appartiendrait au moins aux thèmes *"Technologies objet"* et *"Bases de données"*.

Technologies objet

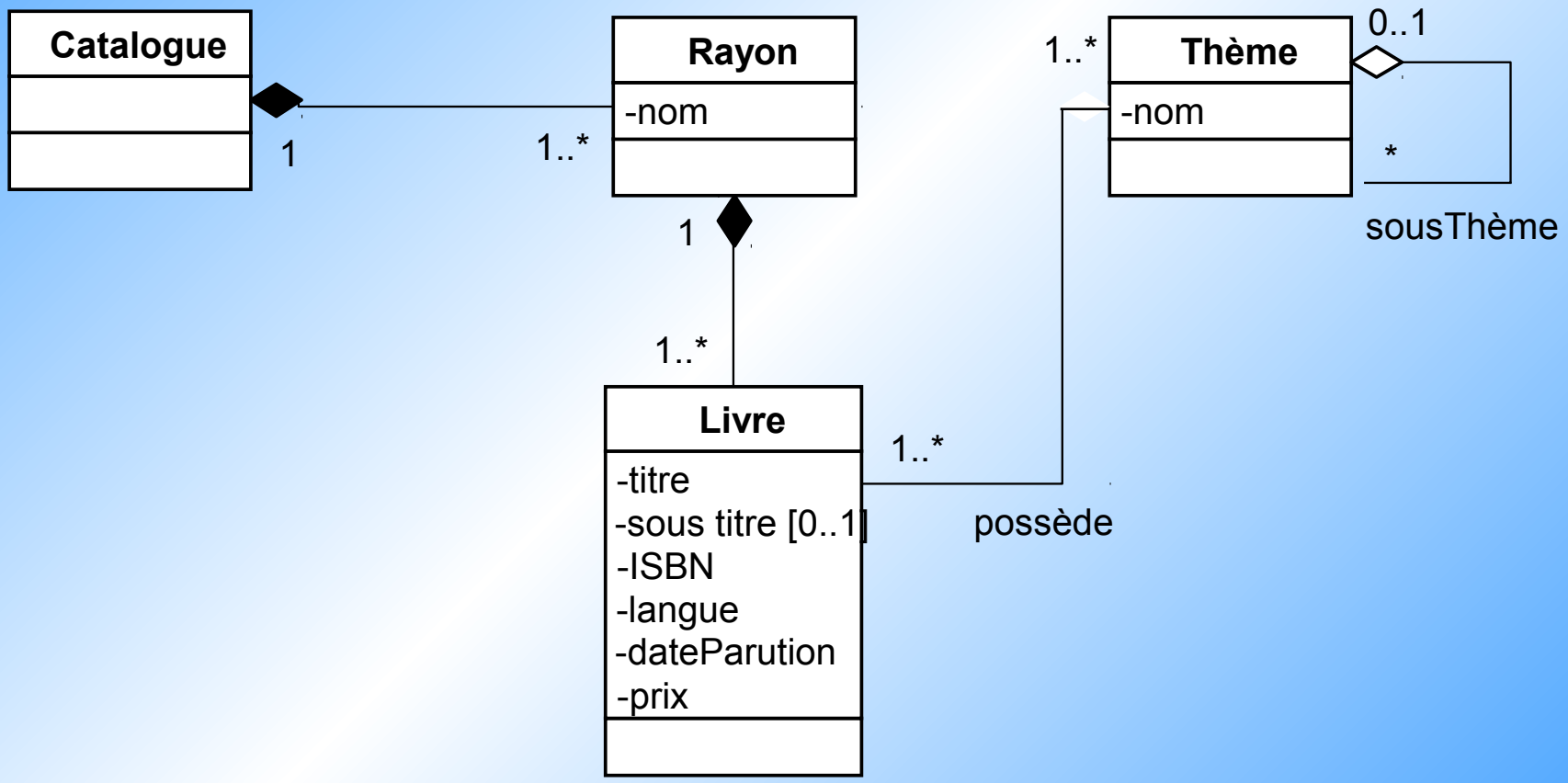
Bases de données

Un thème peut lui-même se décomposer en sous-thèmes : *"Technologies objet"* se décompose en *"UML"*, *"Java"*, *"C++"*, etc.



SOLUTION

Maintenir le catalogue



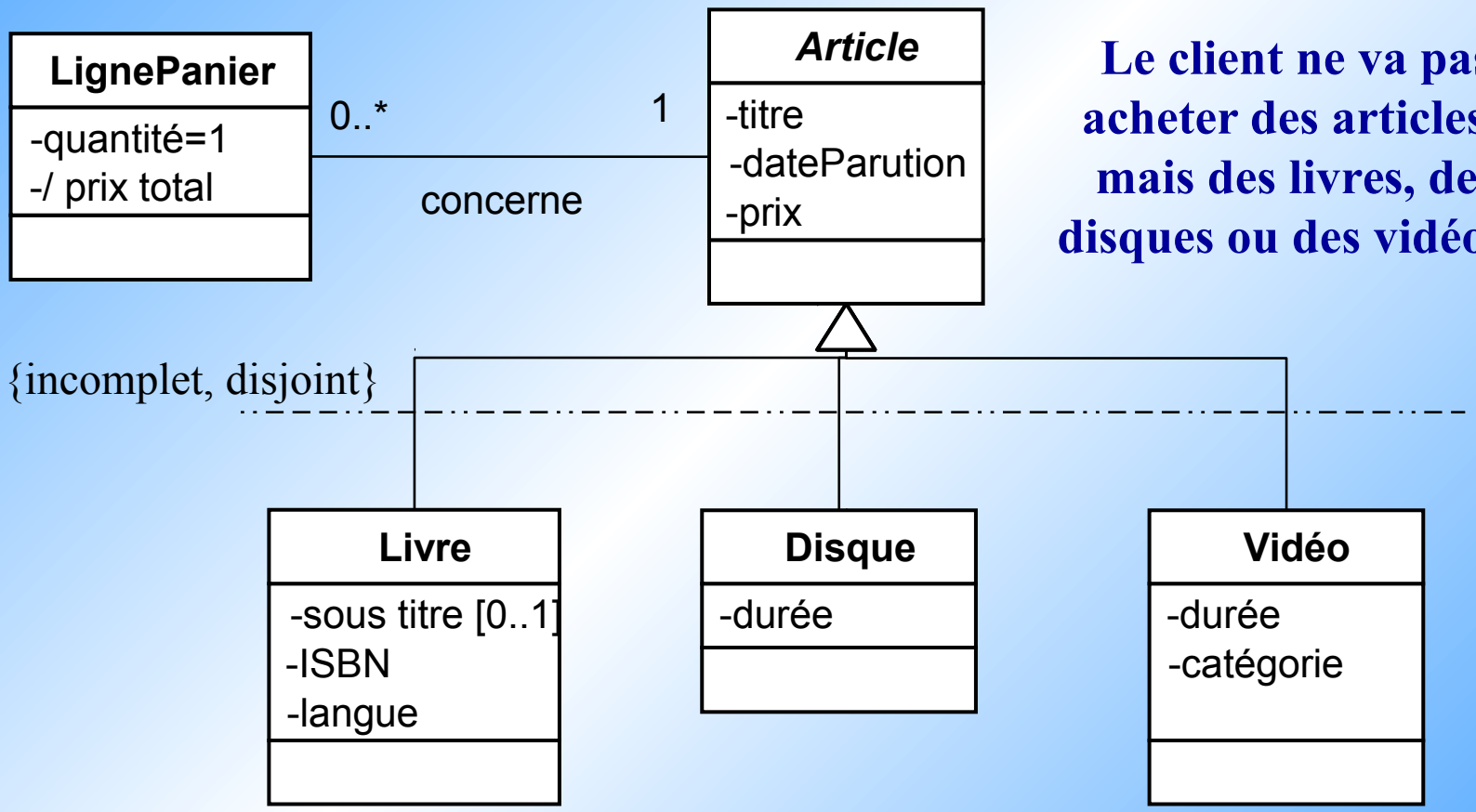
Associations et attributs site web

On va anticiper dès à présent une diversification du champ d'action de ***jeBouquine*** en proposant à la vente des disques et des vidéos.

Faire le modèle

SOLUTION

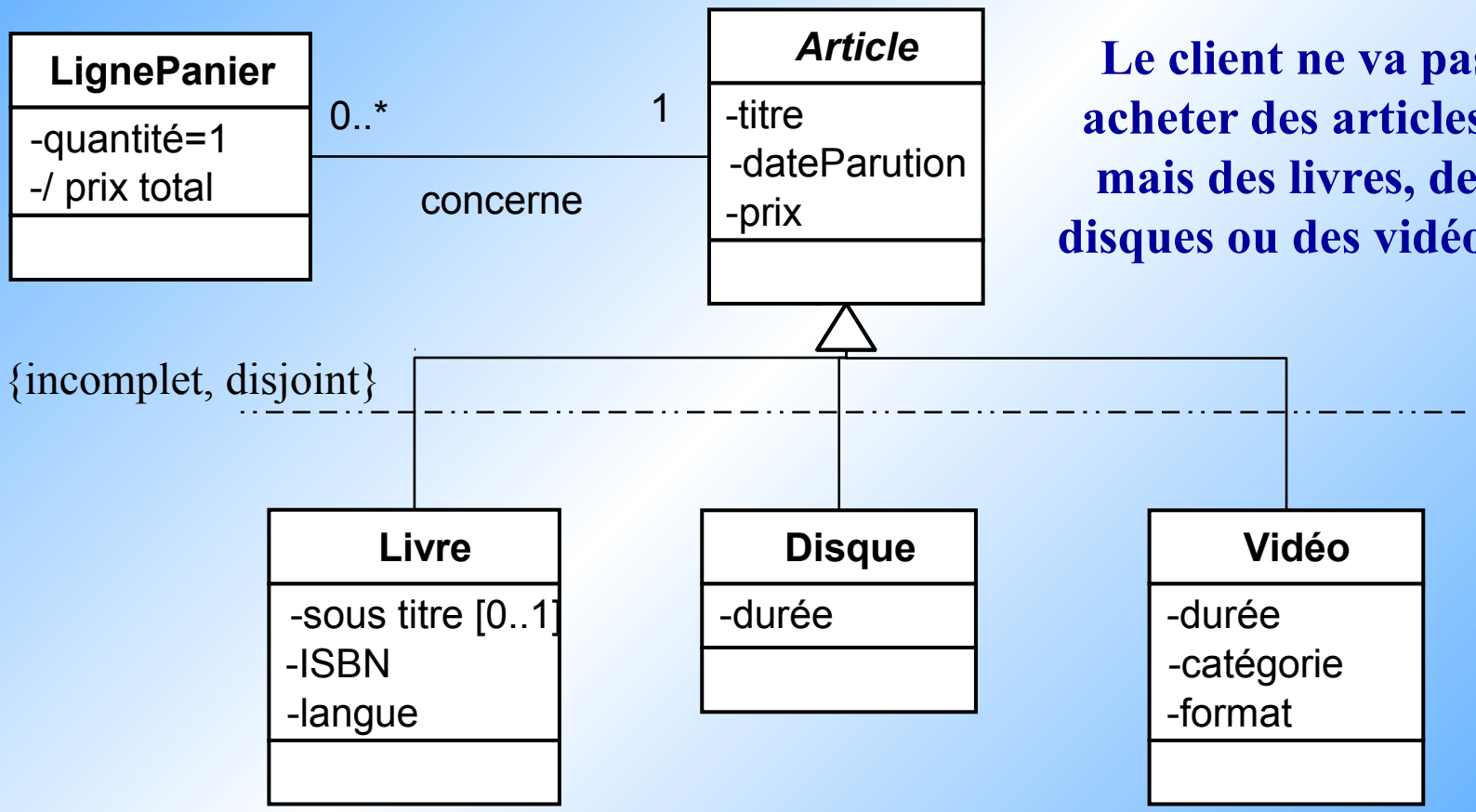
Article : classe abstraite



Le client ne va pas acheter des articles, mais des livres, des disques ou des vidéos !

SOLUTION

Article : classe abstraite



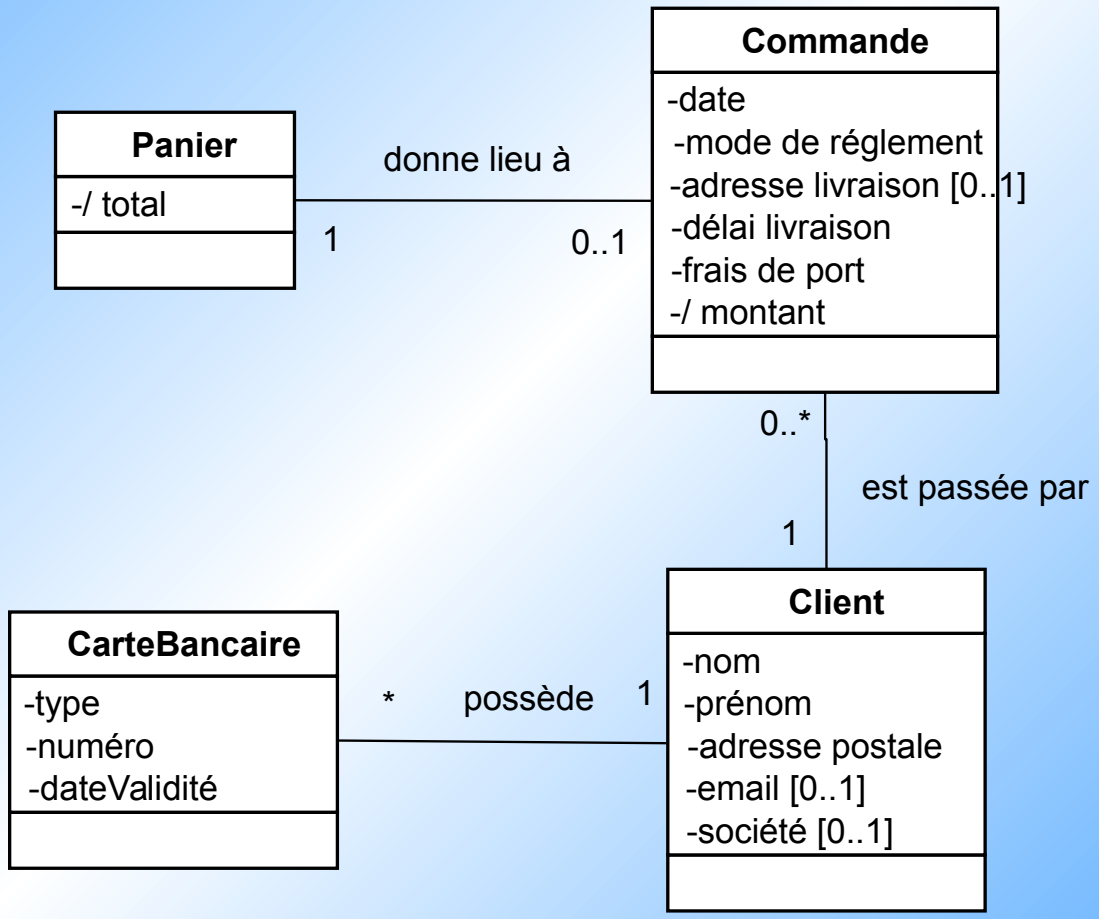
Le client ne va pas acheter des articles, mais des livres, des disques ou des vidéos !

Associations et attributs site Web

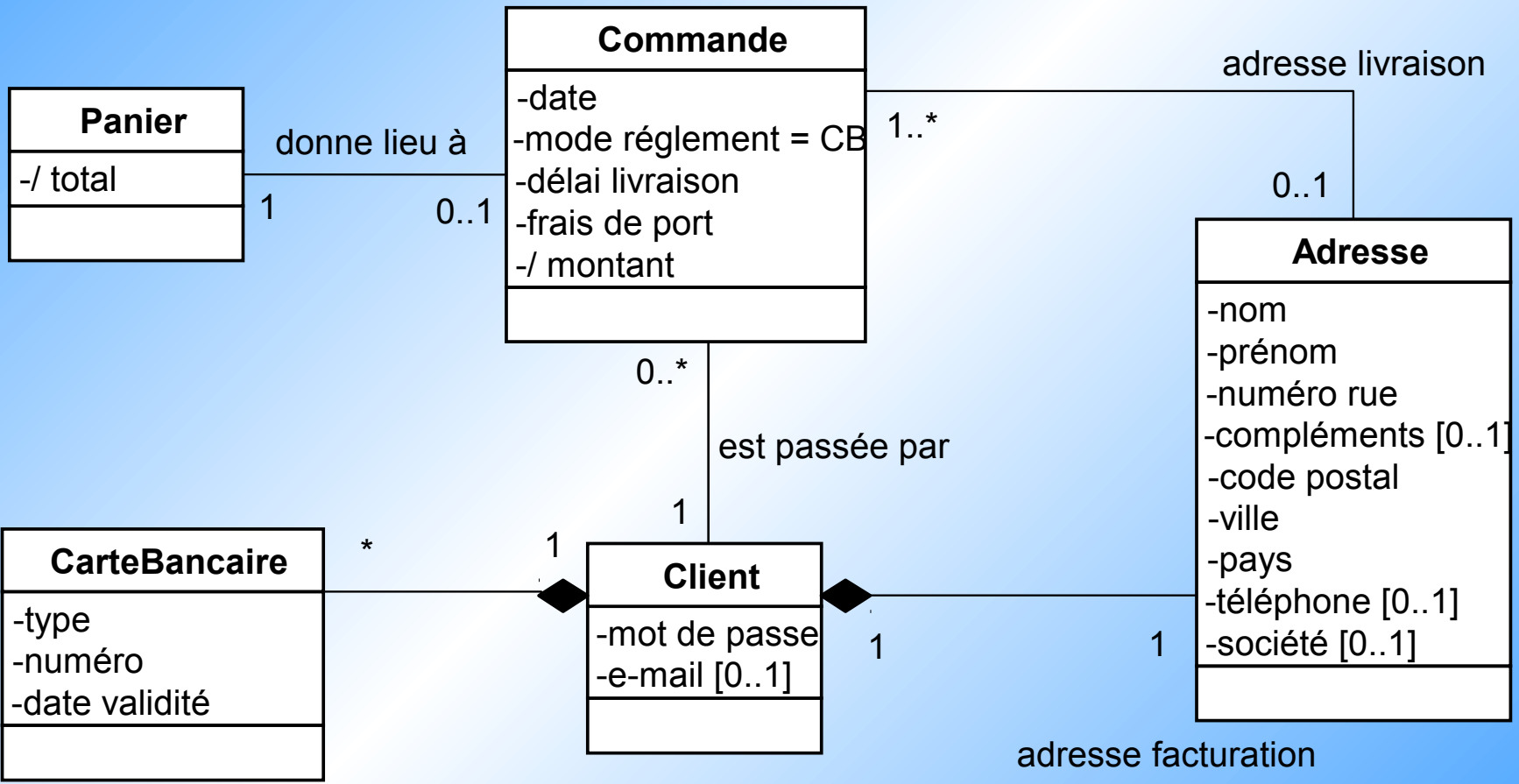
Factoriser les informations de facturation et de livraison d'une commande.

SOLUTION

Ce qu'on avait

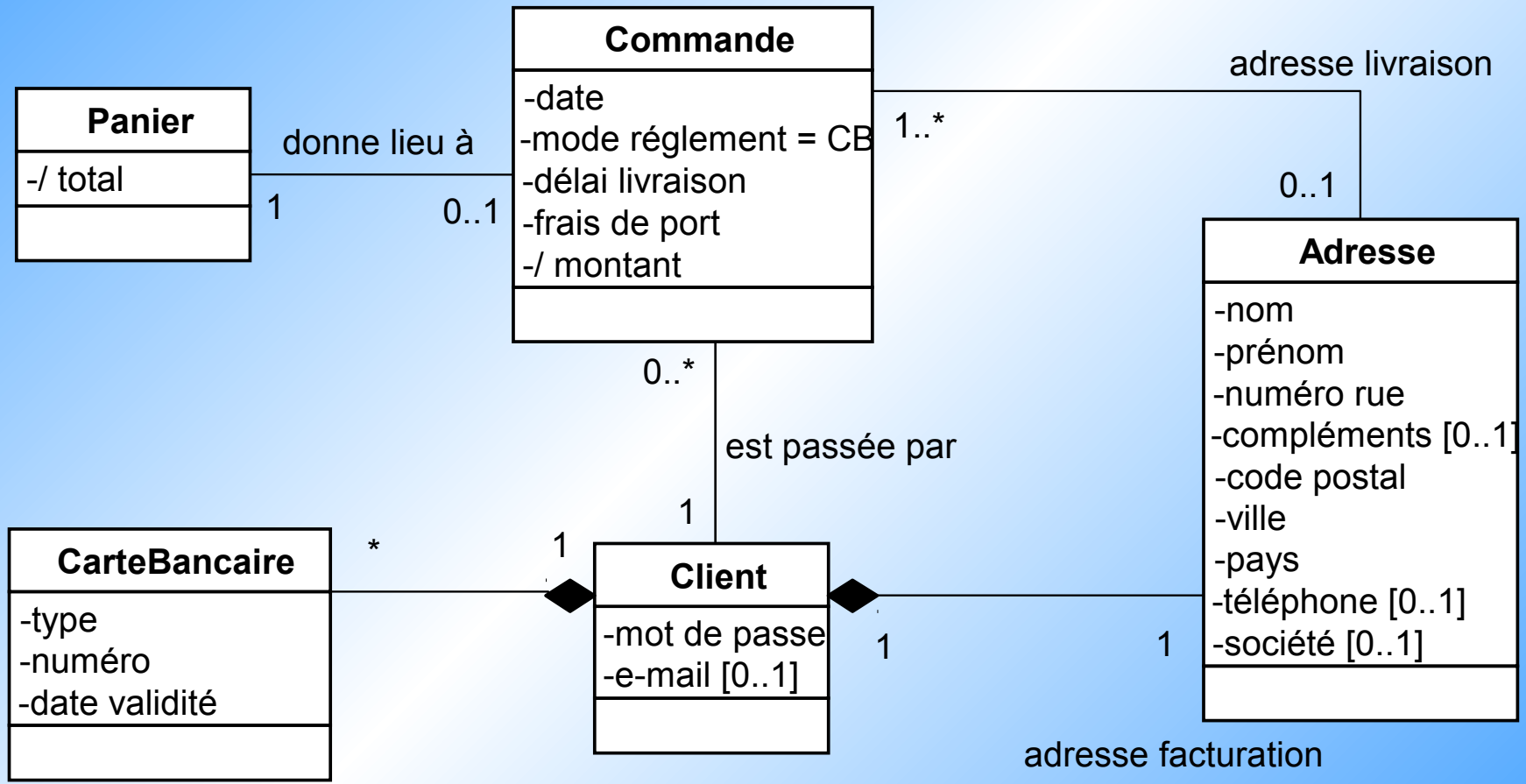


SOLUTION

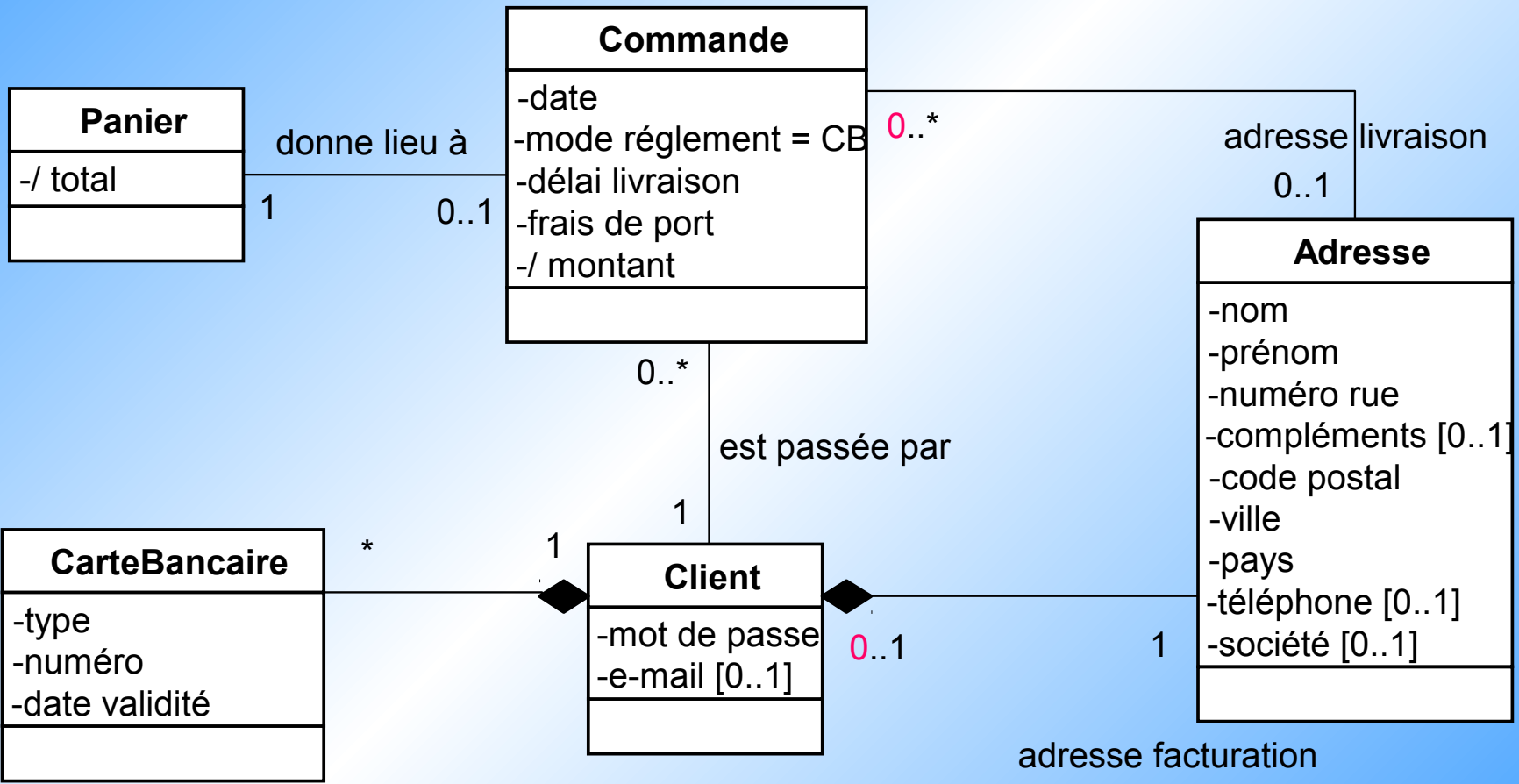

Commande
-date
-mode réglement = CB
-délai livraison
-frais de port
-/ montant
Panier
-/ total
donne lieu à
1
0..1
adresse livraison
1..*
0..1
Adresse
-nom
-prénom
-numéro rue
-compléments [0..1]
-code postal
-ville
-pays
-téléphone [0..1]
-société [0..1]
0..*
est passée par
1
CarteBancaire
-type
-numéro
-date validité
*
1
Client
-mot de passe
-e-mail [0..1]
1
1
adresse facturation

SOLUTION

Est-il correct ?



SOLUTION


Panier
-/ total
donne lieu à
1
0..1
Commande
-date
-mode réglement = CB
-délai livraison
-frais de port
-/ montant
0..*
adresse livraison
0..1
Adresse
-nom
-prénom
-numéro rue
-compléments [0..1]
-code postal
-ville
-pays
-téléphone [0..1]
-société [0..1]
0..*
est passée par
1
CarteBancaire
-type
-numéro
-date validité
*
1
Client
-mot de passe
-e-mail [0..1]
0..1
1
adresse facturation

SOLUTION

- On avait indiqué qu'une commande possède un attribut *adresseLivraison* et que le client a une *adressePostale*.

- Dans le cas d'un cadeau par exemple, l'adresse de livraison doit, **de plus**, comporter le nom et le prénom du destinataire.
Les informations de facturation et de livraison sont vraiment similaires.
Dans le cas fréquent où l'adresse de facturation est identique à l'adresse de livraison, la seconde est optionnelle.

- C'est un bon exemple de factorisation par association plutôt que par relation d'héritage.

SOMMAIRE

- Introduction et démarche
- Identification des concepts du domaine
- Ajout des associations et des attributs
- Généralisation des concepts
- **Structuration en paquetages**

STRUCTURATION EN PAQUETAGES

Paquetage = package en anglais

La structuration d'un modèle est une activité délicate. Elle doit s'appuyer sur deux principes fondamentaux :

1- cohérence

2- ***indépendance***.

STRUCTURATION EN PAQUETAGES

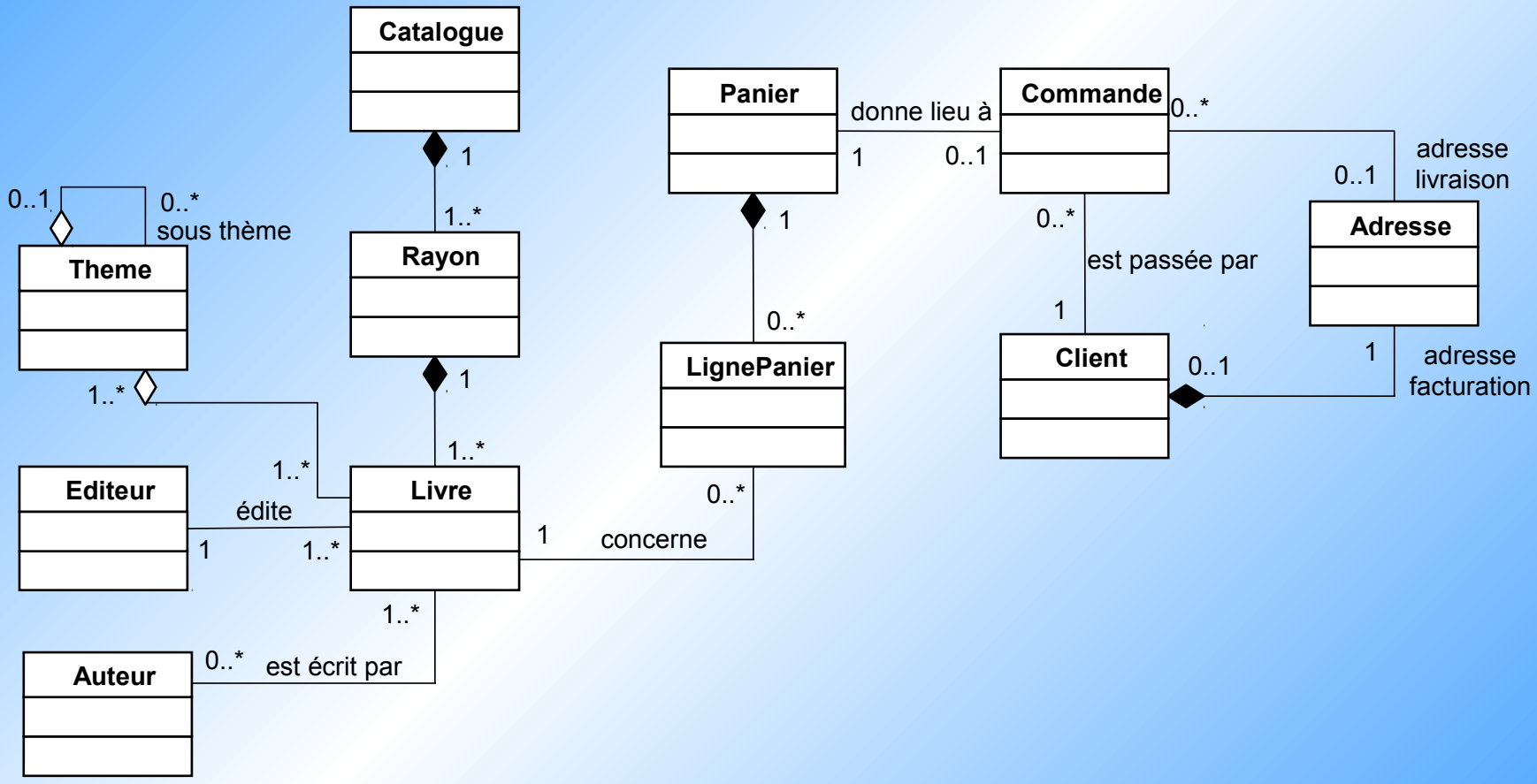
- Le **premier principe** (*cohérence*) consiste à regrouper les classes qui sont proches d'un point de vue sémantique.
Un critère intéressant consiste à évaluer les durées de vie des instances de concept et à rechercher l'homogénéité.

- Le **deuxième principe** (*indépendance*) s'efforce de minimiser les relations entre paquetages, c'est-à-dire plus concrètement les relations entre classes de paquetages différents.

STRUCTURATION EN PAQUETAGES

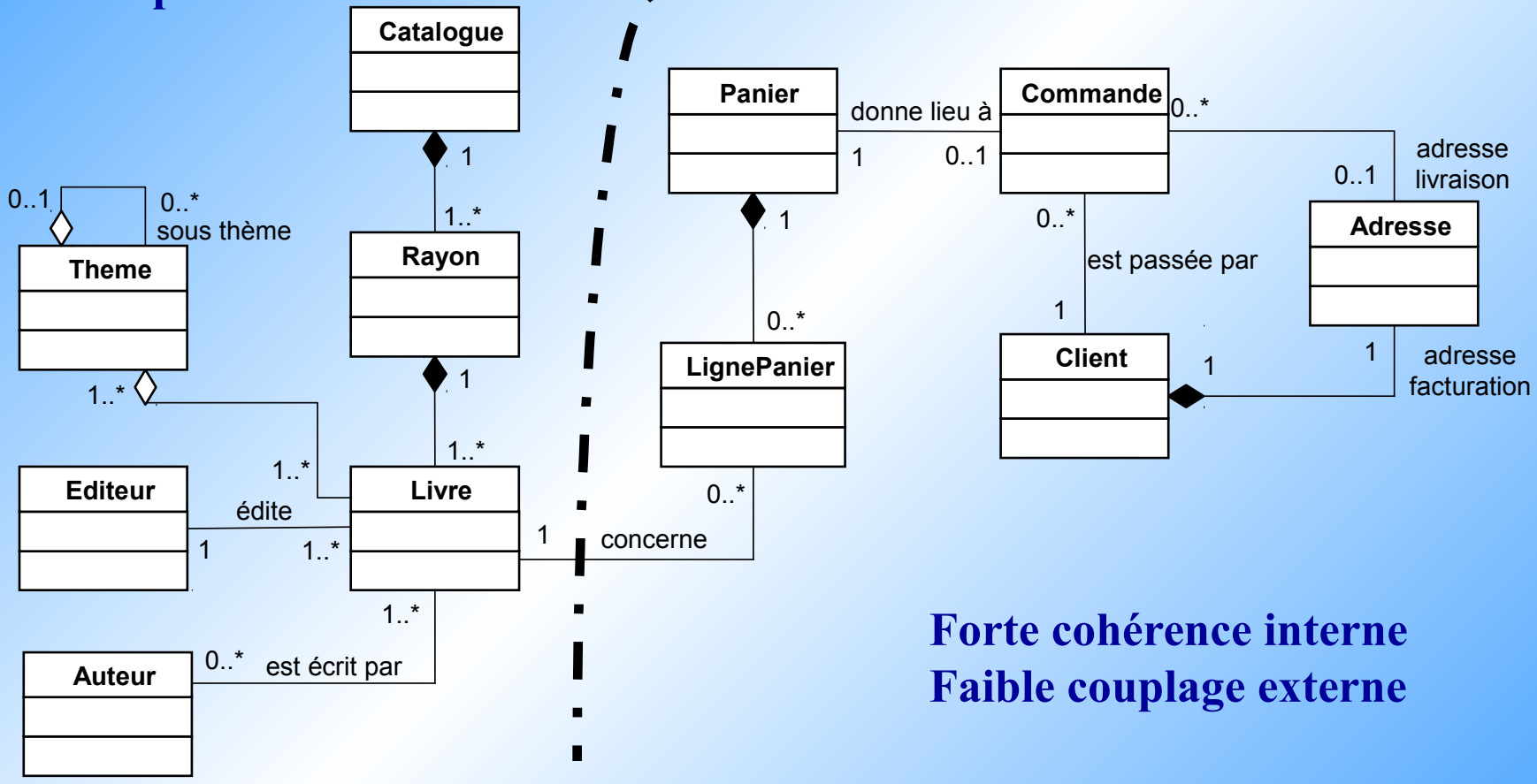
Faire le découpage en paquetages

SOLUTION


Catalogue
1
1..*
Rayon
1
1..*
Livre
1..*
Auteur
0..*
est écrit par
0..1
0..*
sous thème
Theme
1..*
1..*
Editeur
édite
1
1..*
Panier
donne lieu à
1
0..1
Commande
0..*
1
0..*
LignePanier
0..*
1
concerne
0..*
est passée par
1
Client
0..1
0..1
adresse
livraison
Adresse
1
adresse
facturation

SOLUTION

Durée de vie de plusieurs mois, voire de plusieurs années



Forte cohérence interne
Faible couplage externe

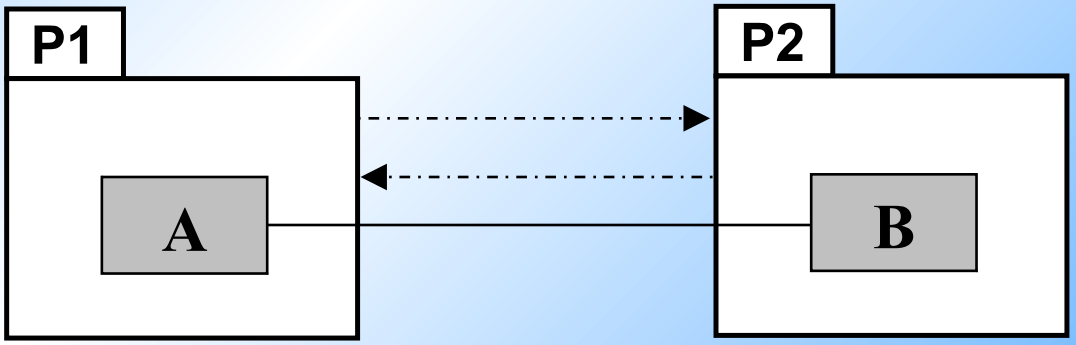
CATALOGUE

GESTION

STRUCTURATION EN PAQUETAGES

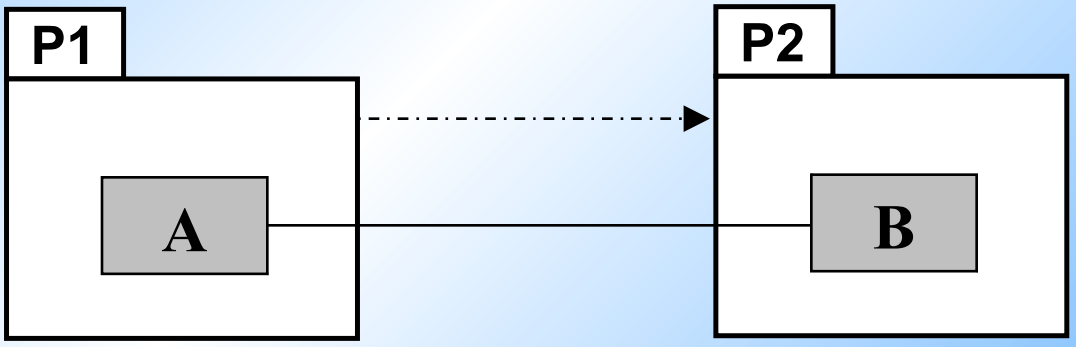
Une association entre deux classes *A* et *B* permet par défaut de naviguer dans les deux sens entre des objets de la classe *A* et des objets de la classe *B*.

Il est possible de limiter cette navigabilité à une seule des deux directions, en particulier pour les associations qui traversent les paquetages, sans quoi nous récupérons systématiquement une paire de dépendances.



STRUCTURATION EN PAQUETAGES

UML nous permet de représenter explicitement cette navigabilité en ajoutant sur l'association une flèche indiquant le seul sens possible.

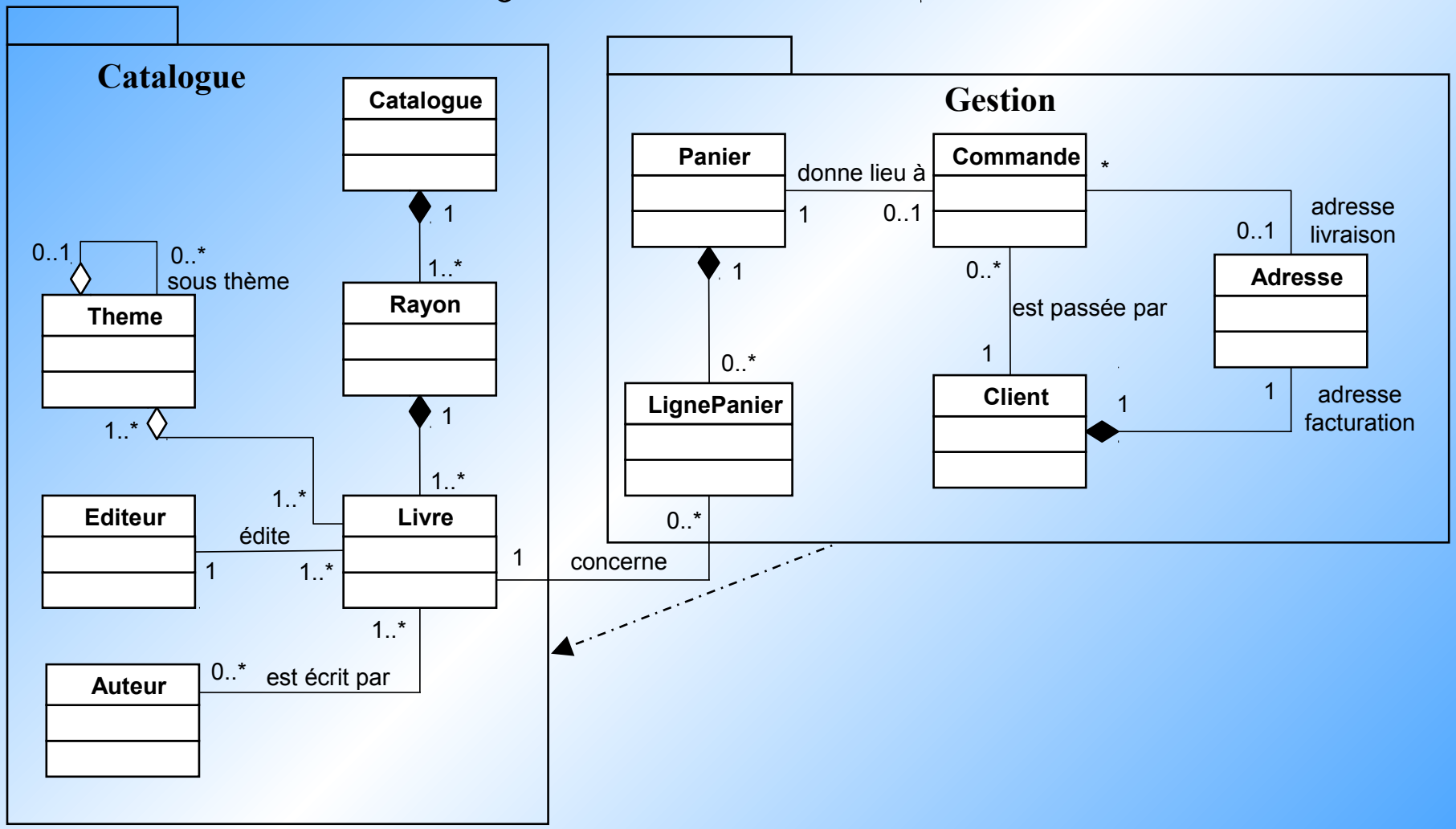


Exercice Web

STRUCTURATION EN PAQUETAGES

Peut-on simplifier la navigation ?

SOLUTION

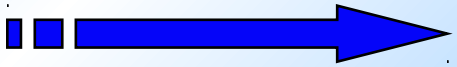

Catalogue
Catalogue
1
1..*
0..1
0..*
sous thème
Theme
Rayon
1..*
1
1..*
1..*
Editeur
édite
Livre
1
1..*
1
concerne
1..*
Auteur
0..*
est écrit par
Gestion
Panier
donne lieu à
Commande
1
0..1
*
0..1
adresse
livraison
Adresse
0..*
est passée par
1
1
0..*
LignePanier
Client
1
1
adresse
facturation
0..*

SOLUTION

Une seule association entre les deux paquetages : entre *Livre* et *LignePanier*.

Est-elle bidirectionnelle ?

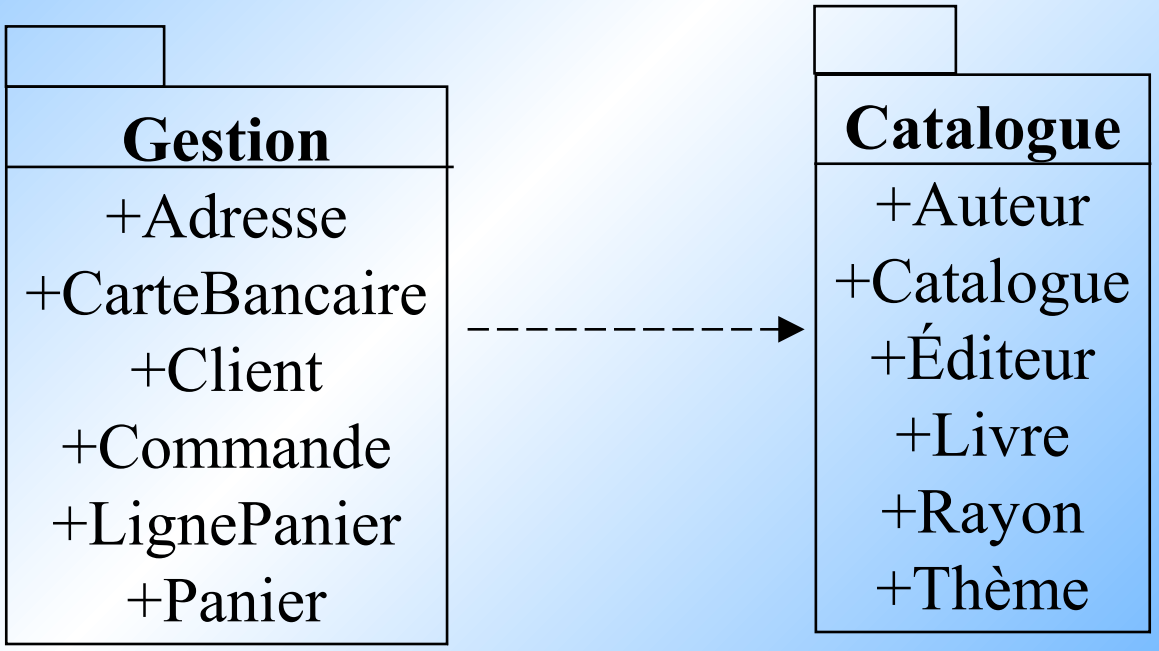
La ligne panier dépend du livre sélectionné mais par contre la définition d'un livre est auto-suffisante.



L'association n'a donc besoin d'être navigable que dans le sens *LignePanier* vers *Livre*, ce qui enlève la dépendance du paquetage *Catalogue* vers le paquetage *Gestion*.

SOLUTION

Représentation plus synthétique des deux paquetages :



SOMMAIRE

- Introduction et démarche
- Identification des concepts du domaine
- Ajout des associations et des attributs
- Généralisation des concepts
- Structuration en paquetages

- **Relations de dépendance, de réalisation et notions d'interfaces**

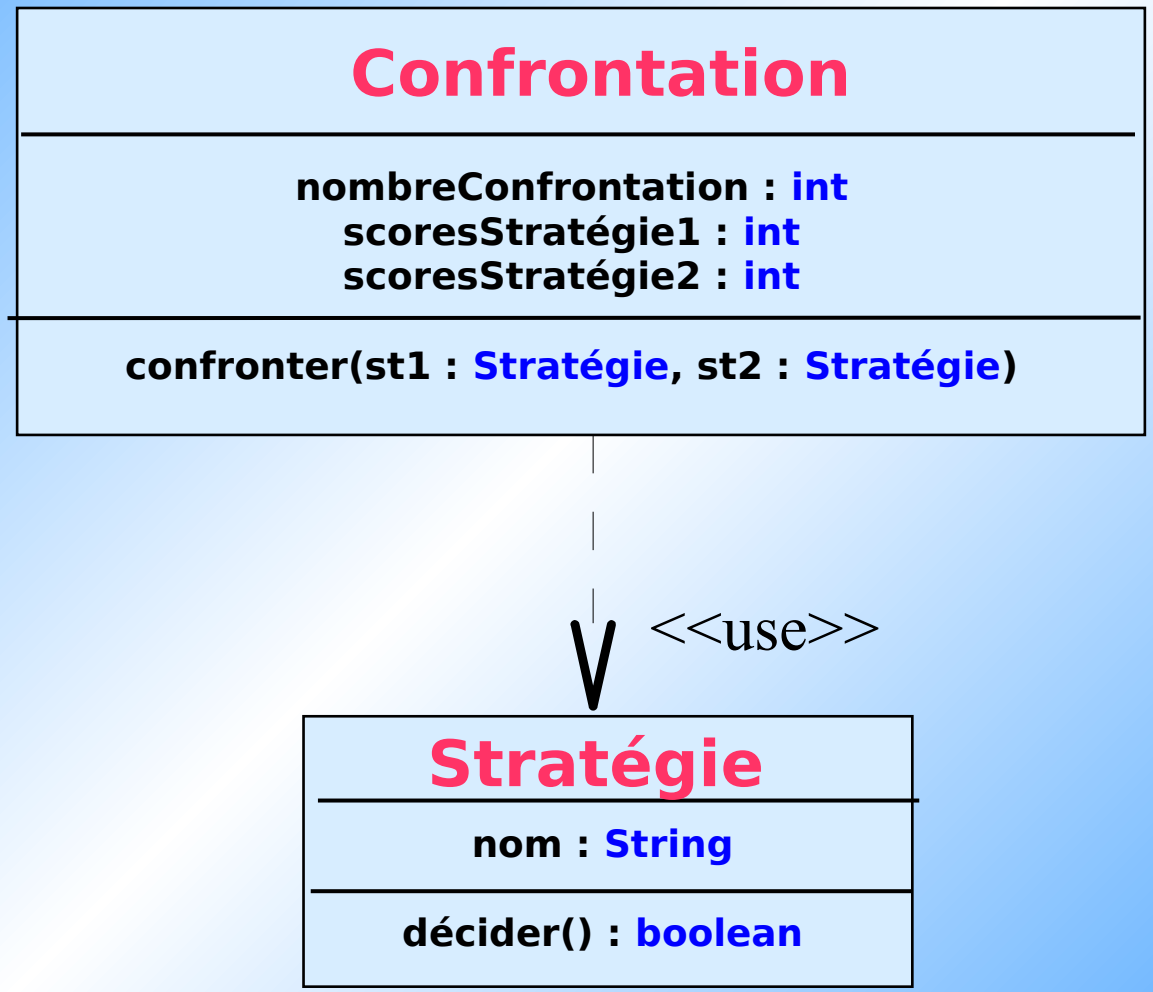
Relations de dépendance

- Une dépendance est une relation unidirectionnelle exprimant une dépendance sémantique entre des éléments du modèle.

- Une dépendance est habituellement utilisée quand une classe en utilise une autre comme argument dans la signature d'une opération.

- Elle indique que la modification de la cible peut impliquer une modification de la source.

Relations de dépendance



INTERFACES

- Liste d'opérations constituant un contrat à respecter par les classes implémentant l'interface

- Déclarations de propriétés et de méthodes **publiques** mais aucune implémentation de celles-ci (*méthodes abstraites*)

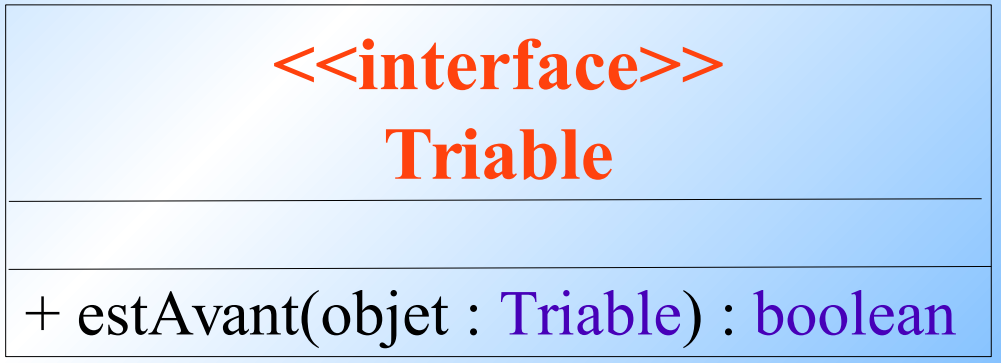
- Prendre en compte le langage d'implémentation du modèle
  (*C++ pas de concept d'interface,*
  Java pas de propriété, constantes uniquement)

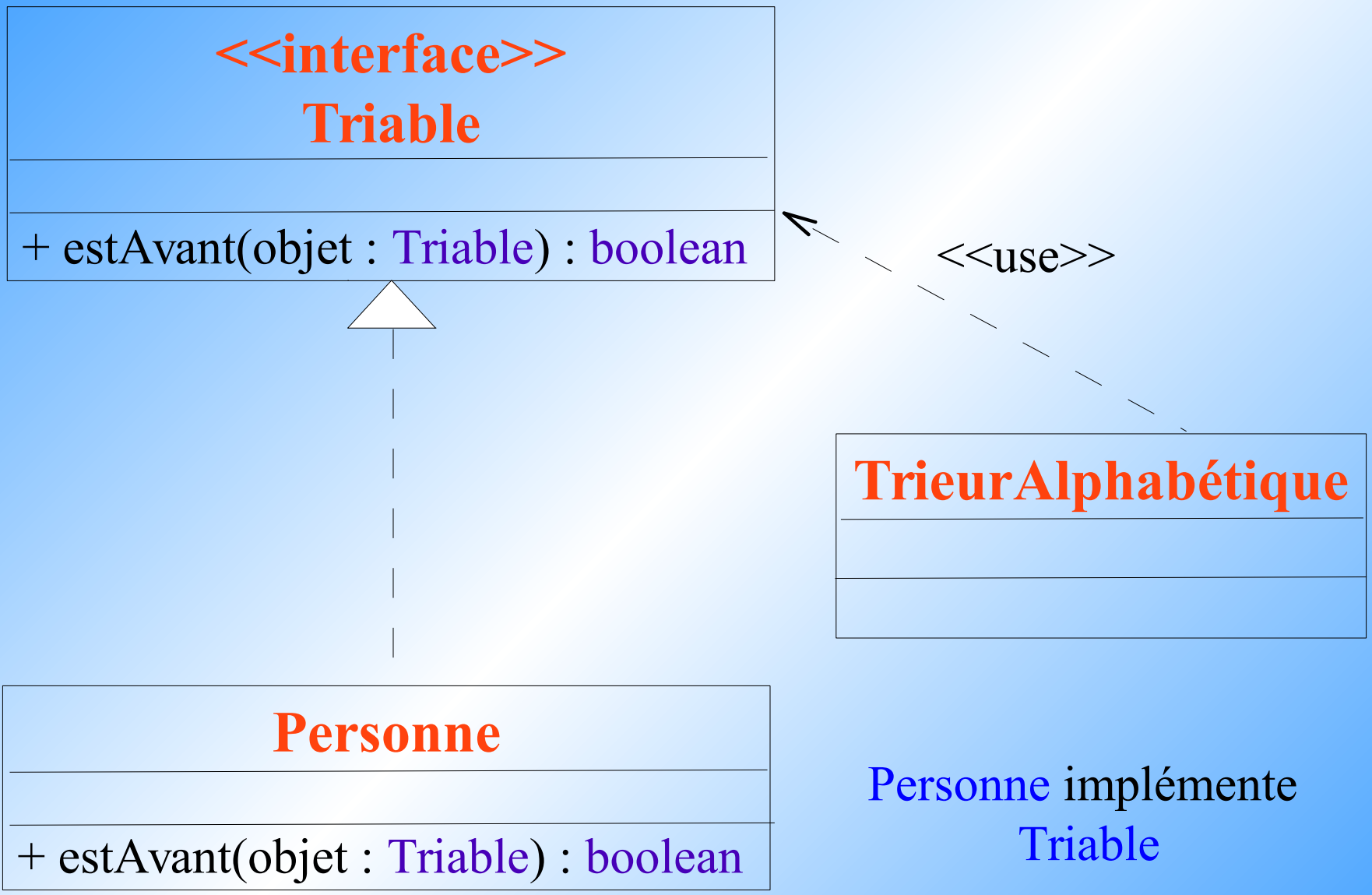
- Une propriété d'interface indique que toute classe implémentant celle-ci doit stocker l'information et fournir les moyens de la manipuler.

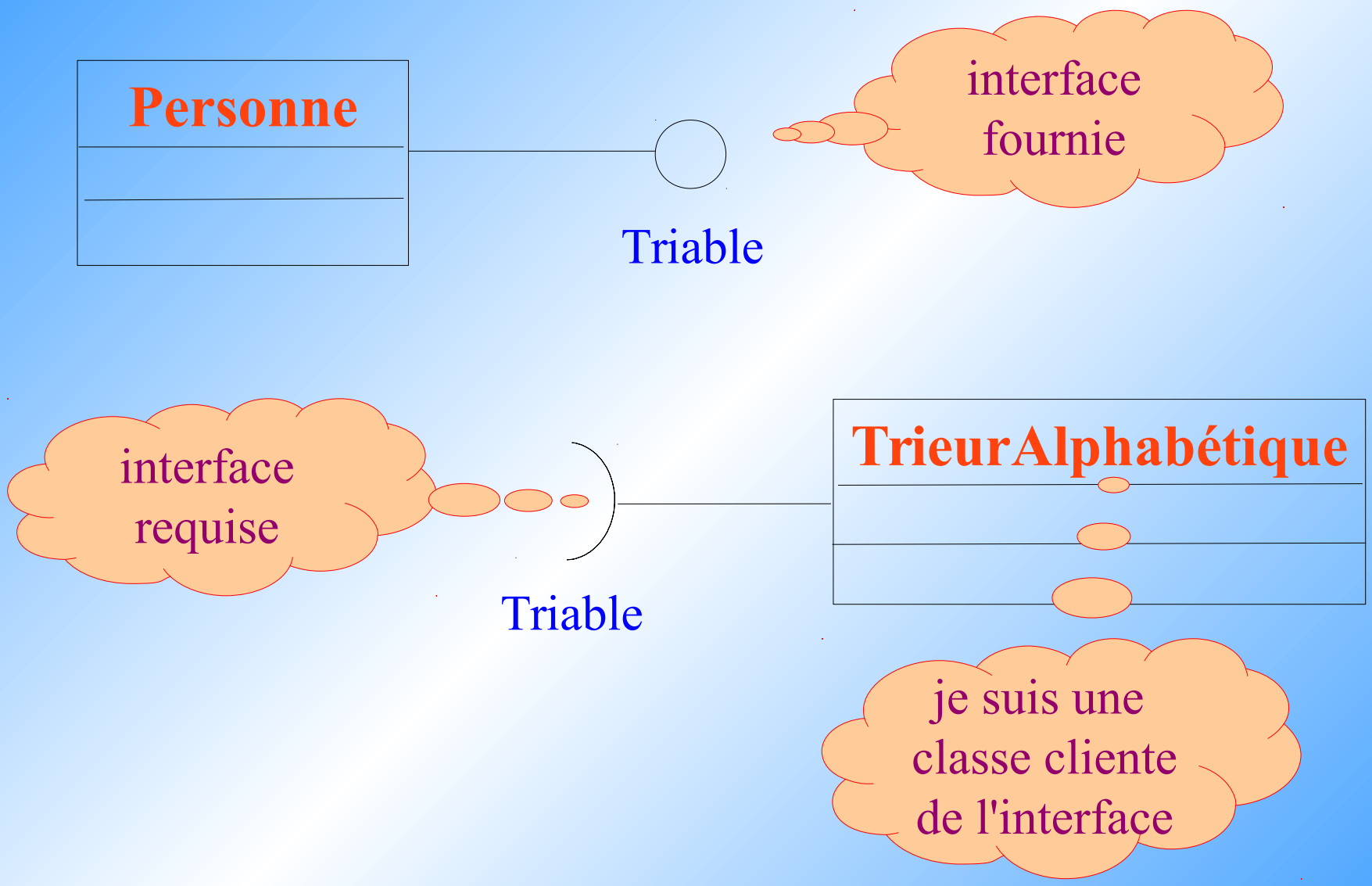
INTERFACES

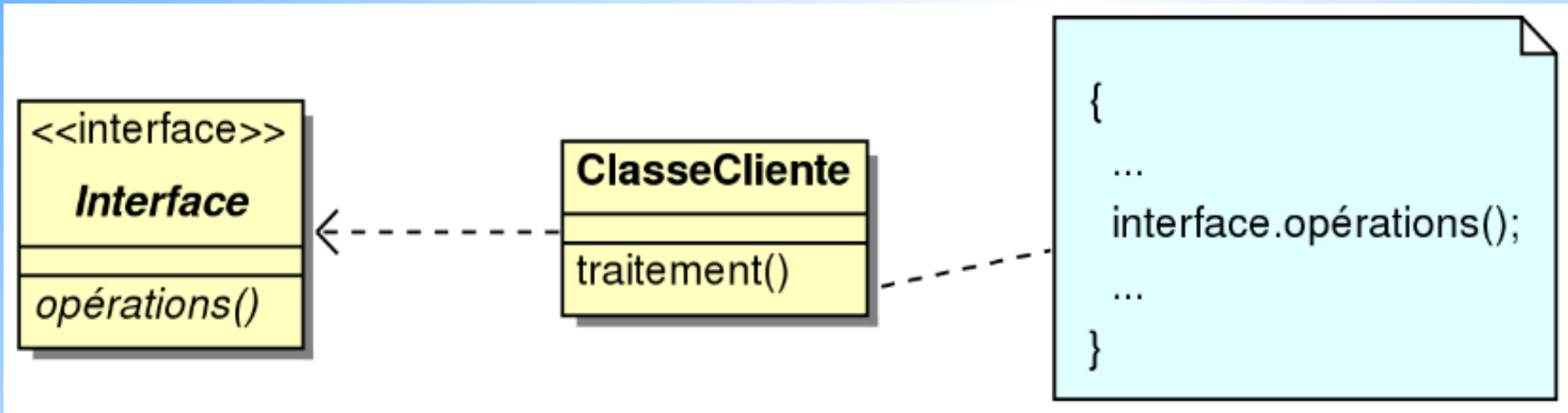
- 2 types de représentations
  (*fonction de ce que l'on doit mettre en lumière*) :
  - Classe avec stéréotype interface
  - Notation à rotule

Classe avec stéréotype

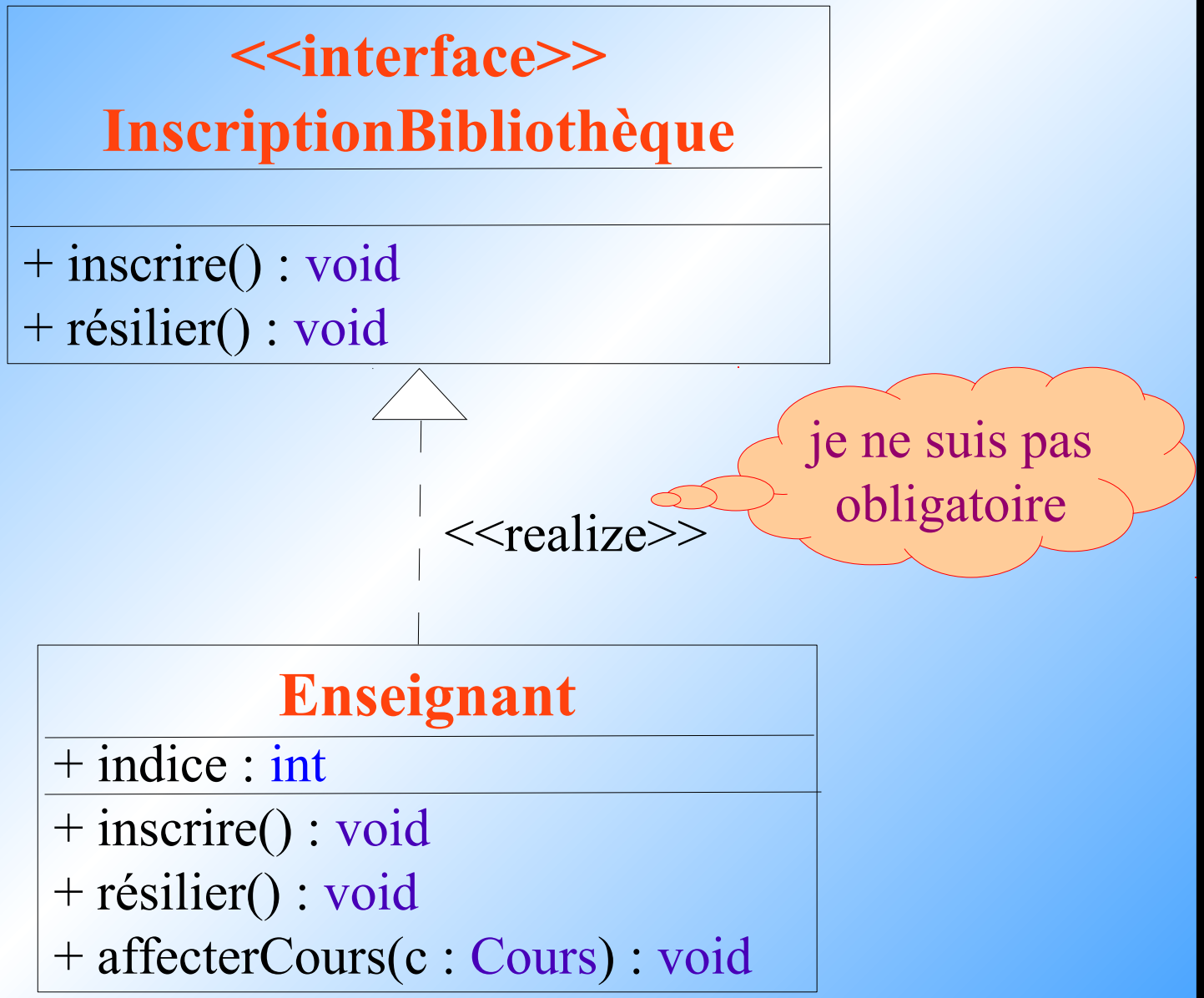


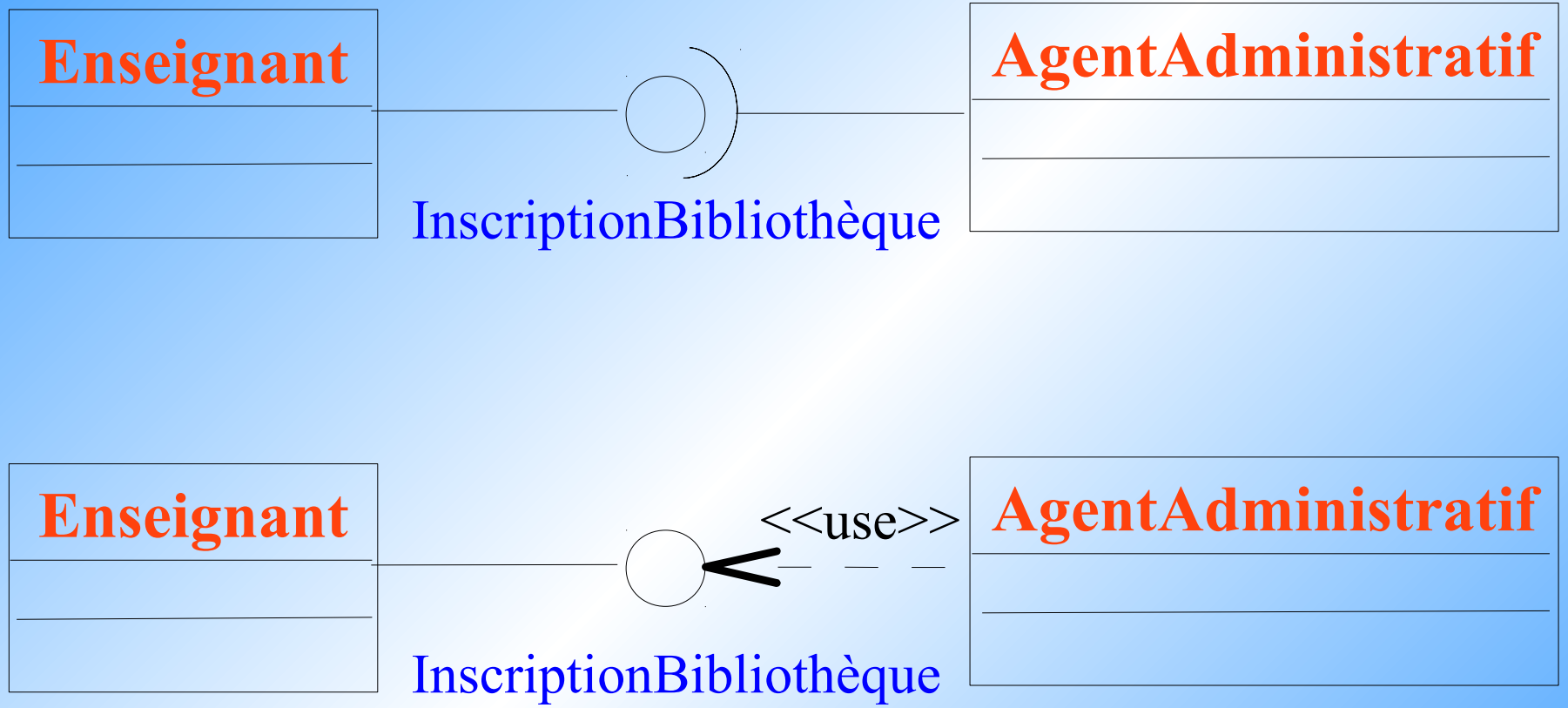

<<interface>>
Triable
+ estAvant(objet : Triable) : boolean
<<use>>
TrieurAlphabétique
Personne
+ estAvant(objet : Triable) : boolean
Personne implémente
Triable


Personne
interface fournie
Triable
interface requise
Triable
TrieurAlphabétique
je suis une classe cliente de l'interface



(Transparent de Pierre Gérard – IUT de Villetaneuse)


<<interface>>
InscriptionBibliothèque
+ inscrire() : void
+ résilier() : void
<<realize>>
je ne suis pas obligatoire
Enseignant
+ indice : int
+ inscrire() : void
+ résilier() : void
+ affecterCours(c : Cours) : void


Enseignant
AgentAdministratif
InscriptionBibliothèque
Enseignant
<<use>>
AgentAdministratif
InscriptionBibliothèque

Exercices

EXERCICE

```
public class Un {
    public int r1() {
        return this.test();
    }
    public int test() {
        return 1;
    }
}

public class Deux extends Un {
    public int test() {
        return 2;
    }
}
```

```
public class Trois extends Deux {
    public int r2() {
        return this.r1();
    }
    public int r3() {
        return super.test();
    }
}

public class Quatre extends Trois {
    public int test() {
        return 4;
    }
}
```

EXERCICE


```
public class TestSuper {
    public static void main(java.lang.String[] args) {
        Un e1 = new Un();
        Deux e2 = new Deux();
        Trois e3 = new Trois();
        Quatre e4 = new Quatre();

        System.out.println(e1.test());
        System.out.println(e1.r1());
        System.out.println(e2.test());
        System.out.println(e2.r1());
        System.out.println(e3.test());
        System.out.println(e4.r1());
        System.out.println(e3.r2());
        System.out.println(e4.r2());
        System.out.println(e3.r3());
        System.out.println(e4.r3());
    }
```

SOLUTION

System.out.println(e1.test()); ?

```
public class Un {
    public int r1() {
        return this.test();
    }
    public int test() {
        return 1;
    }
}

public class Deux extends Un {
    public int test() {
        return 2;
    }
}
```

```
public class Trois extends Deux {
    public int r2() {
        return this.r1();
    }
    public int r3() {
        return super.test();
    }
}

public class Quatre extends Trois {
    public int test() {
        return 4;
    }
}
```

SOLUTION

System.out.println(e1.r1()); ?

```
public class Un {
    public int r1() {
        return this.test();
    }
    public int test() {
        return 1;
    }
}

public class Deux extends Un {
    public int test() {
        return 2;
    }
}
```

```
public class Trois extends Deux {
    public int r2() {
        return this.r1();
    }
    public int r3() {
        return super.test();
    }
}

public class Quatre extends Trois {
    public int test() {
        return 4;
    }
}
```

SOLUTION

System.out.println(e2.test()); ?

```
public class Un {
    public int r1() {
        return this.test();
    }
    public int test() {
        return 1;
    }
}

public class Deux extends Un {
    public int test() {
        return 2;
    }
}
```

```
public class Trois extends Deux {
    public int r2() {
        return this.r1();
    }
    public int r3() {
        return super.test();
    }
}

public class Quatre extends Trois {
    public int test() {
        return 4;
    }
}
```

SOLUTION

System.out.println(e2.r1()); ?

```
public class Un {
    public int r1() {
        return this.test();
    }
    public int test() {
        return 1;
    }
}

public class Deux extends Un {
    public int test() {
        return 2;
    }
}
```

```
public class Trois extends Deux {
    public int r2() {
        return this.r1();
    }
    public int r3() {
        return super.test();
    }
}

public class Quatre extends Trois {
    public int test() {
        return 4;
    }
}
```

SOLUTION

System.out.println(e2.r1()); ?

```
public class Un {
    public int r1() {
        return this.test();
    }
    public int test() {
        return 1;
    }
}

public class Deux extends Un {
    public int test() {
        return 2;
    }
}
```

```
public class Trois extends Deux {
    public int r2() {
        return this.r1();
    }
    public int r3() {
        return super.test();
    }
}

public class Quatre extends Trois {
    public int test() {
        return 4;
    }
}
```

SOLUTION

System.out.println(e3.test()); ?

```
public class Un {
    public int r1() {
        return this.test();
    }
    public int test() {
        return 1;
    }
}

public class Deux extends Un {
    public int test() {
        return 2;
    }
}
```

```
public class Trois extends Deux {
    public int r2() {
        return this.r1();
    }
    public int r3() {
        return super.test();
    }
}

public class Quatre extends Trois {
    public int test() {
        return 4;
    }
}
```

SOLUTION

System.out.println(e4.r1()); ?

```
public class Un {
    public int r1() {
        return this.test();
    }
    public int test() {
        return 1;
    }
}

public class Deux extends Un {
    public int test() {
        return 2;
    }
}
```

```
public class Trois extends Deux {
    public int r2() {
        return this.r1();
    }
    public int r3() {
        return super.test();
    }
}

public class Quatre extends Trois {
    public int test() {
        return 4;
    }
}
```

SOLUTION

System.out.println(e3.r2()); ?

```
public class Un {
    public int r1() {
        return this.test();
    }
    public int test() {
        return 1;
    }
}

public class Deux extends Un {
    public int test() {
        return 2;
    }
}
```

```
public class Trois extends Deux {
    public int r2() {
        return this.r1();
    }
    public int r3() {
        return super.test();
    }
}

public class Quatre extends Trois {
    public int test() {
        return 4;
    }
}
```

SOLUTION

System.out.println(e4.r2()); ?

```
public class Un {
    public int r1() {
        return this.test();
    }
    public int test() {
        return 1;
    }
}

public class Deux extends Un {
    public int test() {
        return 2;
    }
}
```

```
public class Trois extends Deux {
    public int r2() {
        return this.r1();
    }
    public int r3() {
        return super.test();
    }
}

public class Quatre extends Trois {
    public int test() {
        return 4;
    }
}
```

SOLUTION

System.out.println(e3.r3()); ?

```
public class Un {
    public int r1() {
        return this.test();
    }
    public int test() {
        return 1;
    }
}

public class Deux extends Un {
    public int test() {
        return 2;
    }
}
```

```
public class Trois extends Deux {
    public int r2() {
        return this.r1();
    }
    public int r3() {
        return super.test();
    }
}

public class Quatre extends Trois {
    public int test() {
        return 4;
    }
}
```

SOLUTION

System.out.println(e4.r3()); ?

```
public class Un {
    public int r1() {
        return this.test();
    }
    public int test() {
        return 1;
    }
}

public class Deux extends Un {
    public int test() {
        return 2;
    }
}
```

```
public class Trois extends Deux {
    public int r2() {
        return this.r1();
    }
    public int r3() {
        return super.test();
    }
}

public class Quatre extends Trois {
    public int test() {
        return 4;
    }
}
```

SOLUTION


```
System.out.println(e1.test());      1
System.out.println(e1.r1());        1
System.out.println(e2.test());      2
System.out.println(e2.r1());        2
System.out.println(e3.test());      2
System.out.println(e4.r1());        4
System.out.println(e3.r2());        2
System.out.println(e4.r2());        4
System.out.println(e3.r3());        2
System.out.println(e4.r3());        2
```

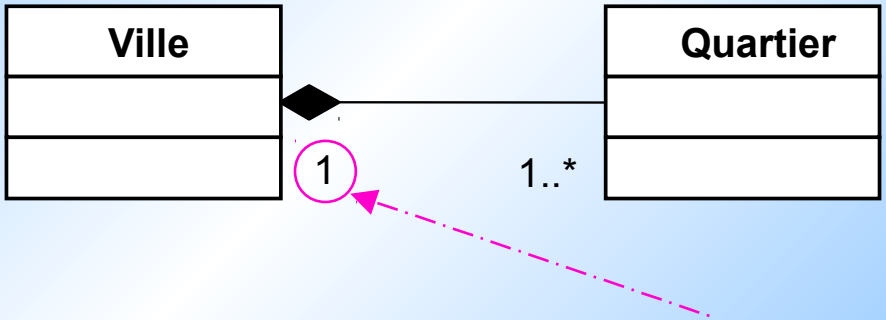
EXERCICE

Une ville possède des quartiers.

Faire deux classes Ville et Quartier
et trouver la bonne association

SOLUTION

Une ville possède des quartiers.



quartier est un objet non partagé et
le cycle de vie des objets est imbriqué

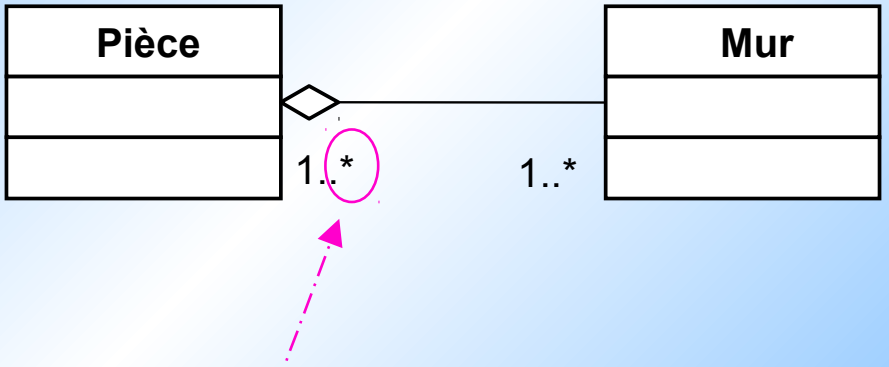
(Si la ville est détruite, alors ses quartiers le seront également)

EXERCICE

Une pièce contient des murs.

SOLUTION

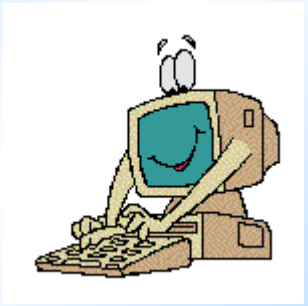
Une pièce contient des murs.



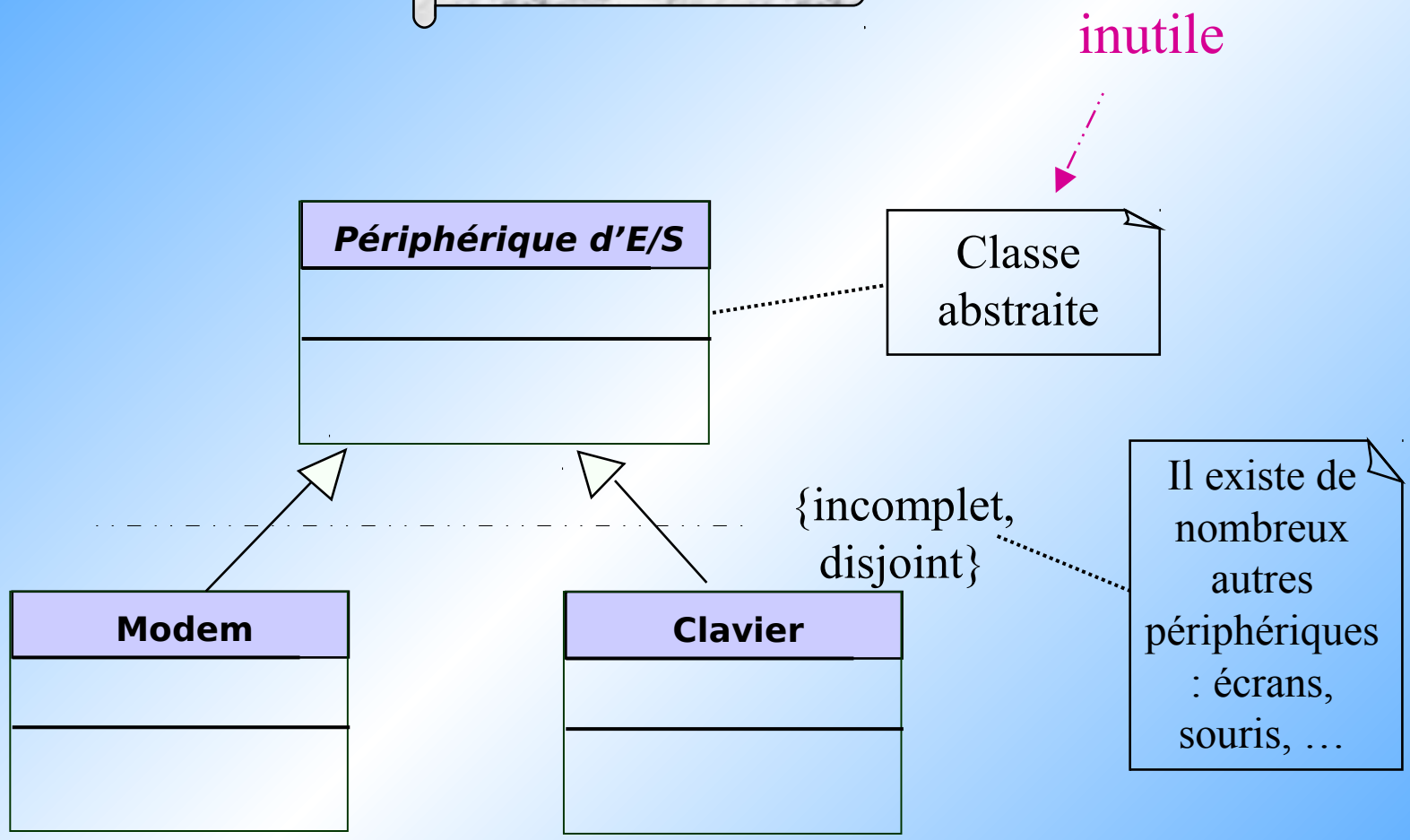
Le mur est un objet partagé donc cela ne pourra jamais être une composition

EXERCICE

Les modems et les claviers sont des périphériques d'entrée / sortie.



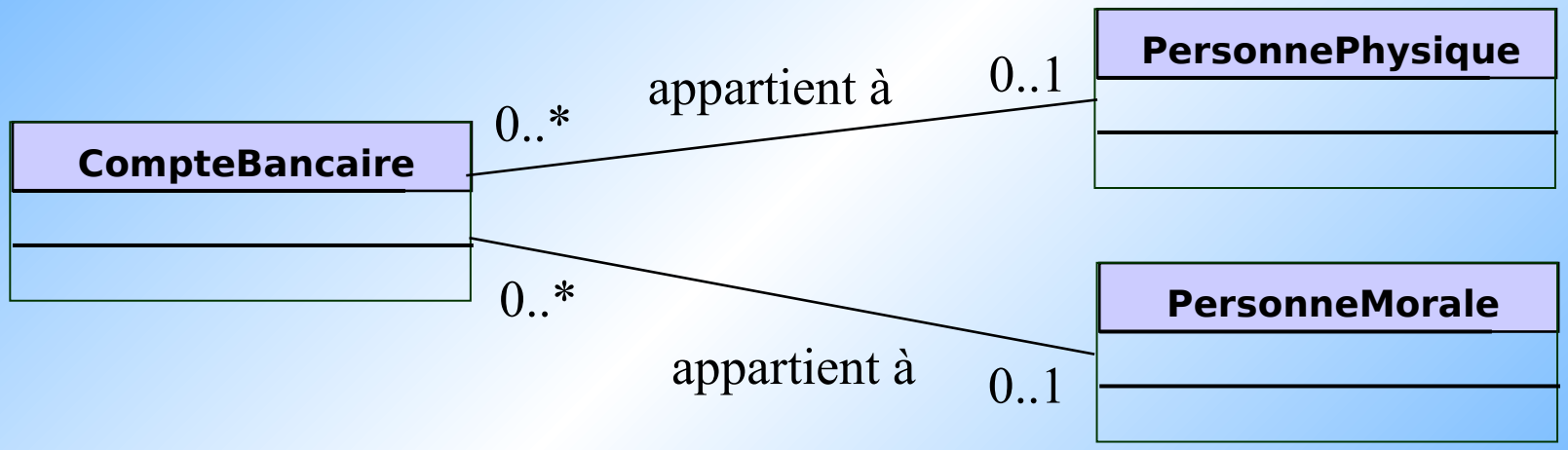
SOLUTION


inutile
Périphérique d'E/S
Classe abstraite
{incomplet, disjoint}
Il existe de nombreux autres périphériques : écrans, souris, …
Modem
Clavier

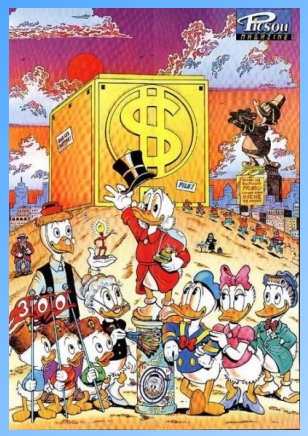
EXERCICE

Un compte bancaire peut appartenir soit à une personne physique, soit à une personne morale.

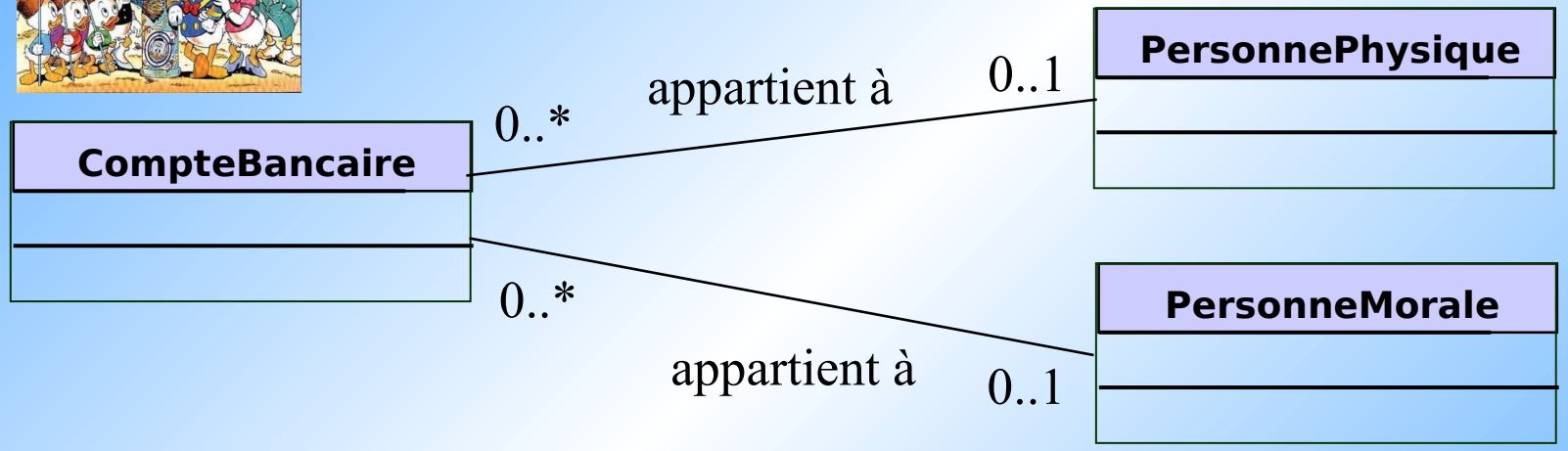
SOLUTION



Est-ce correct ?



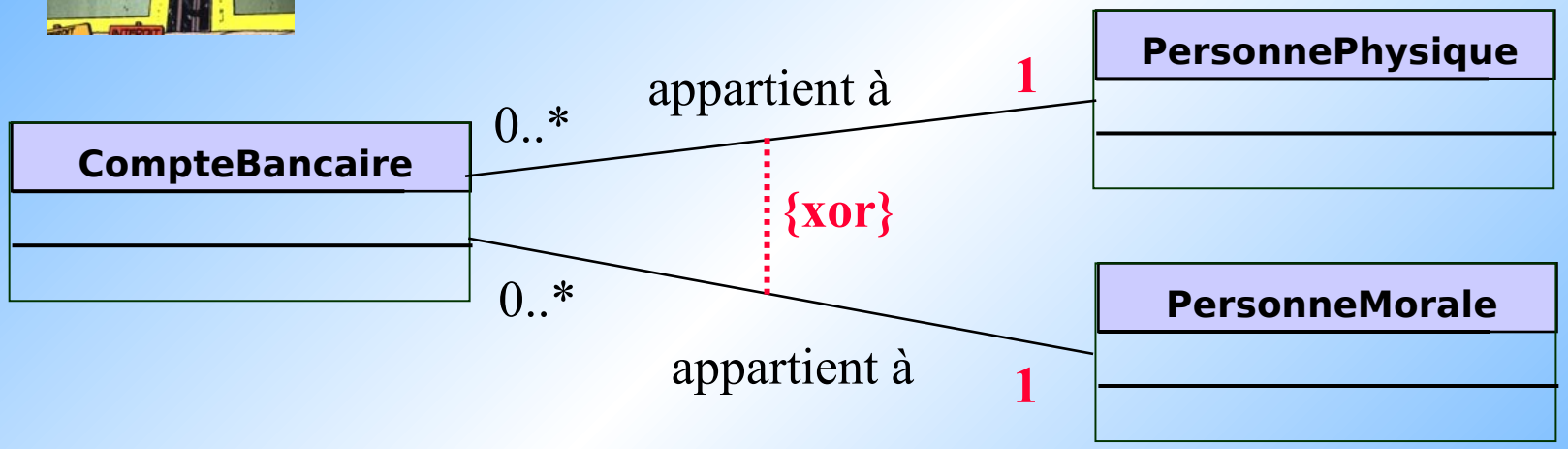
SOLUTION



*Un compte bancaire peut être associé à aucune personne ou également être associé à une personne physique **et** morale.*

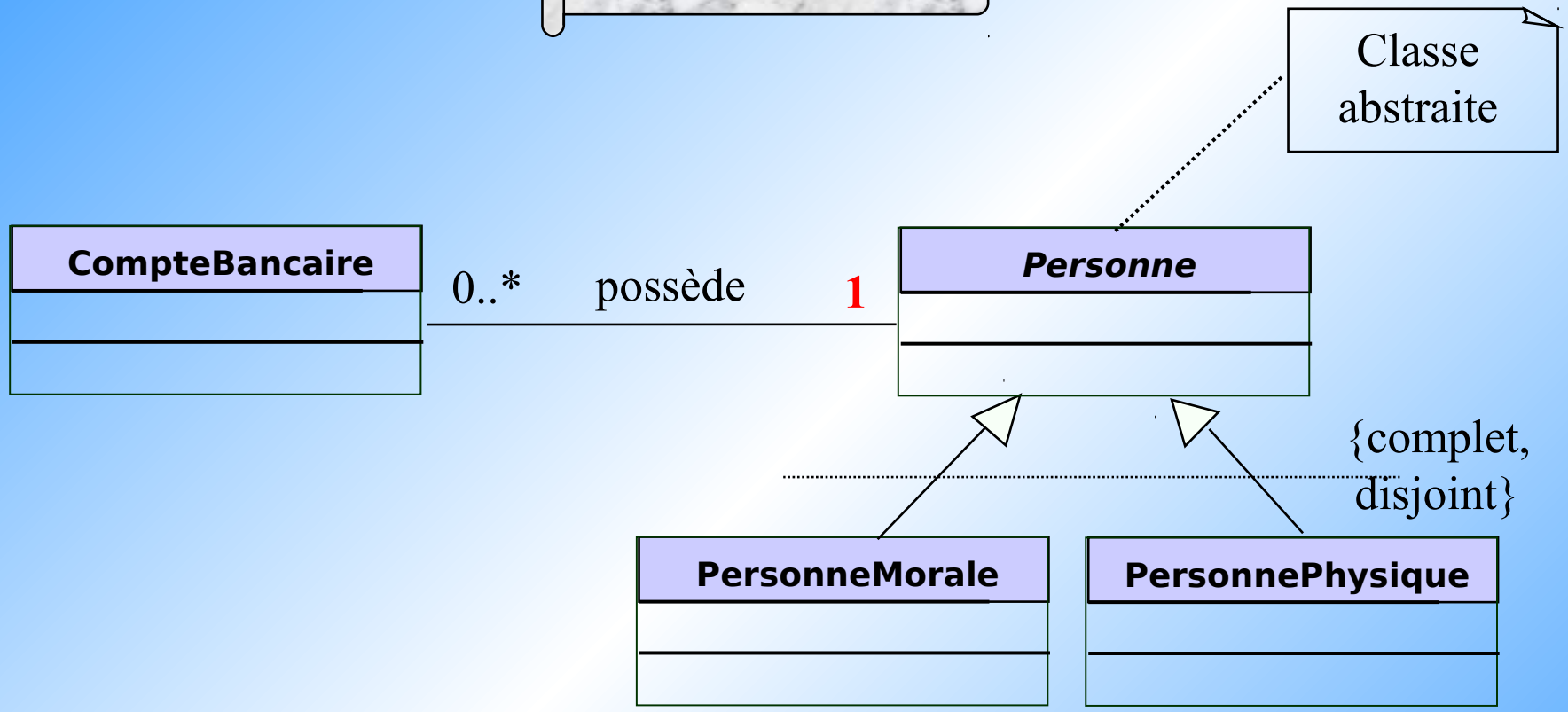
Cette solution ne reflète pas le "*ou exclusif*".

SOLUTION

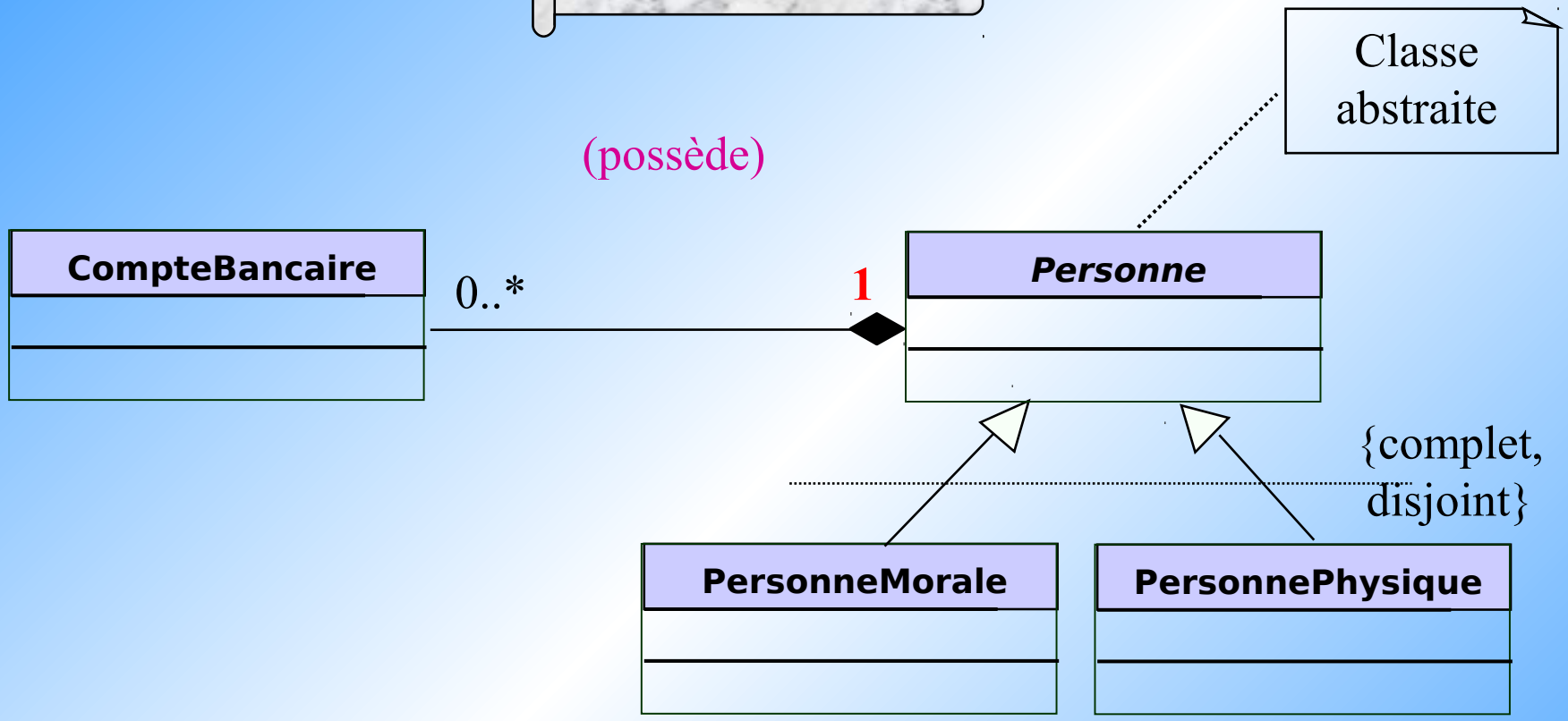


Introduire la contrainte prédéfinie {*xor*} entre les deux associations qui portent maintenant une multiplicité strictement égale à *1*.

SOLUTION


Classe abstraite
CompteBancaire
0..*
possède
1
Personne
{complet, disjoint}
PersonneMorale
PersonnePhysique

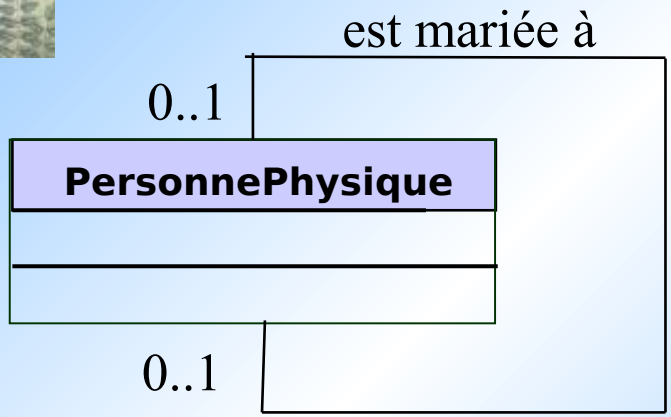
SOLUTION


Classe abstraite
(possède)
CompteBancaire
0..*
1
Personne
{complet, disjoint}
PersonneMorale
PersonnePhysique

EXERCICE

Deux personnes peuvent être mariées.

SOLUTION

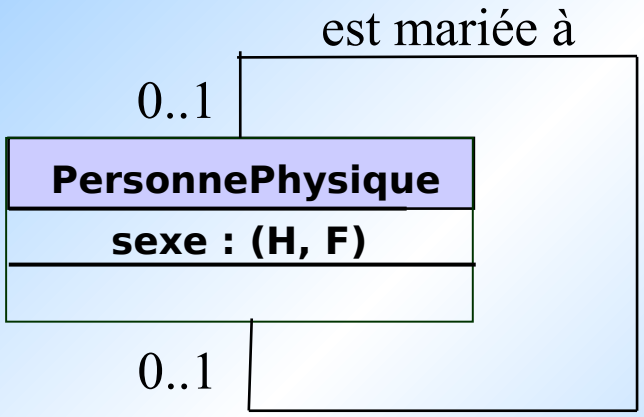


Droit français : une personne n'est pas tenue d'être mariée, mais ne peut pas être mariée avec plusieurs personnes à la fois.

Ajouter la contrainte suivante :

Le mariage ne peut unir que des personnes de sexe opposé.

SOLUTION

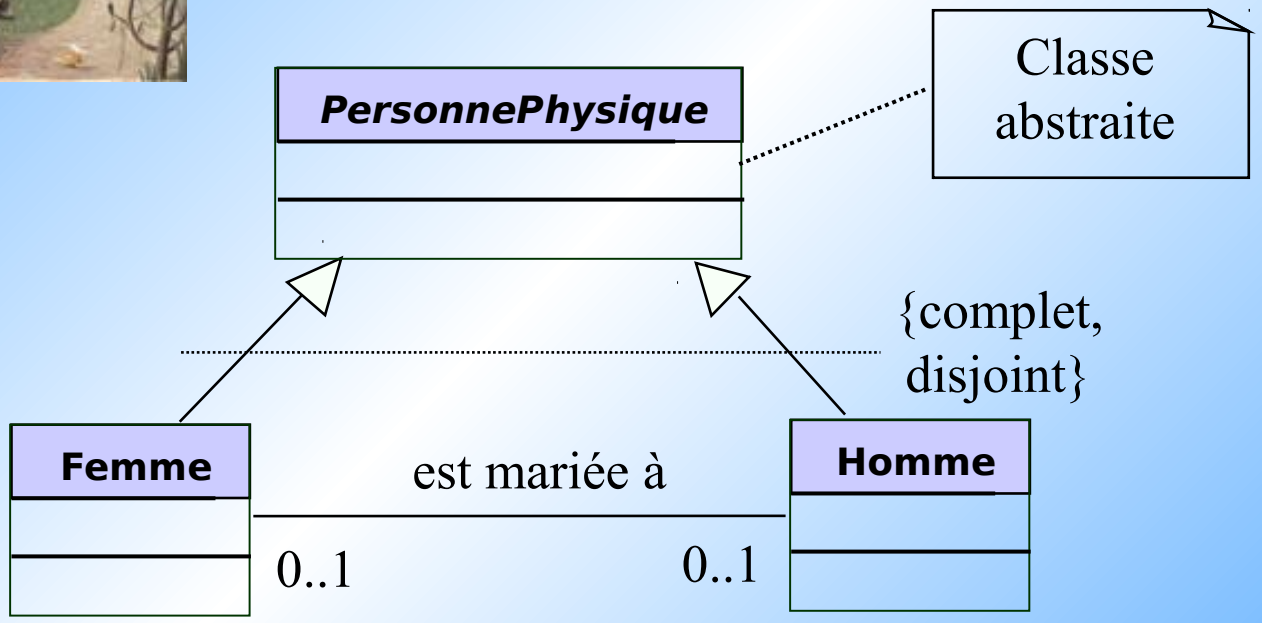


{deux personnes de même sexe ne peuvent pas être mariées}

{Personne1.sexe ≠ Personne2.sexe}

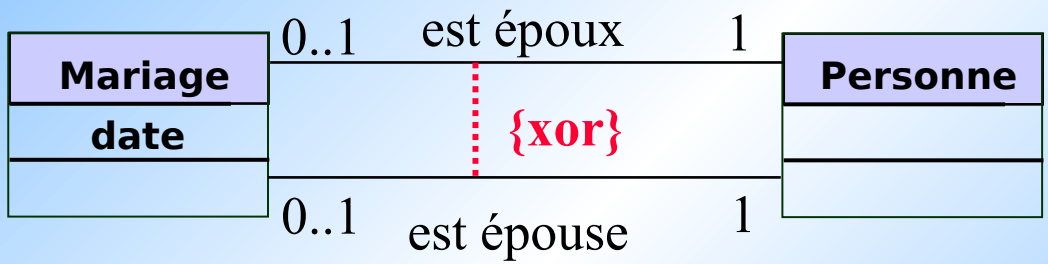
Introduire explicitement un attribut énuméré *sexe* : (*H*, *F*) et une contrainte sur l'association.

SOLUTION



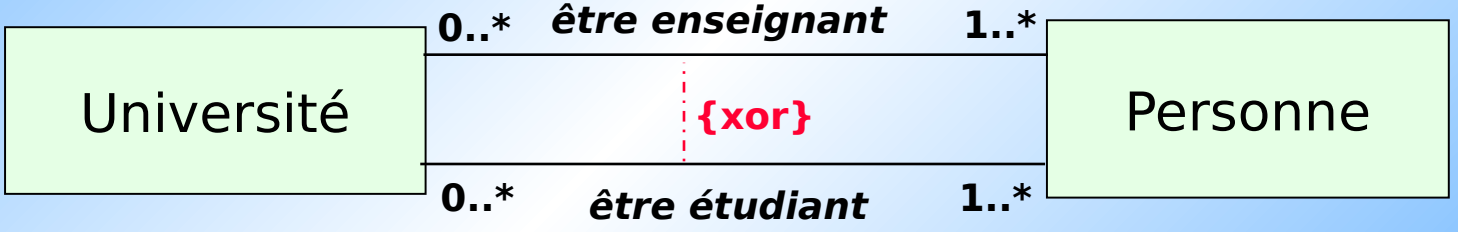
Introduire *2* sous-classes Homme et Femme.

SOLUTION

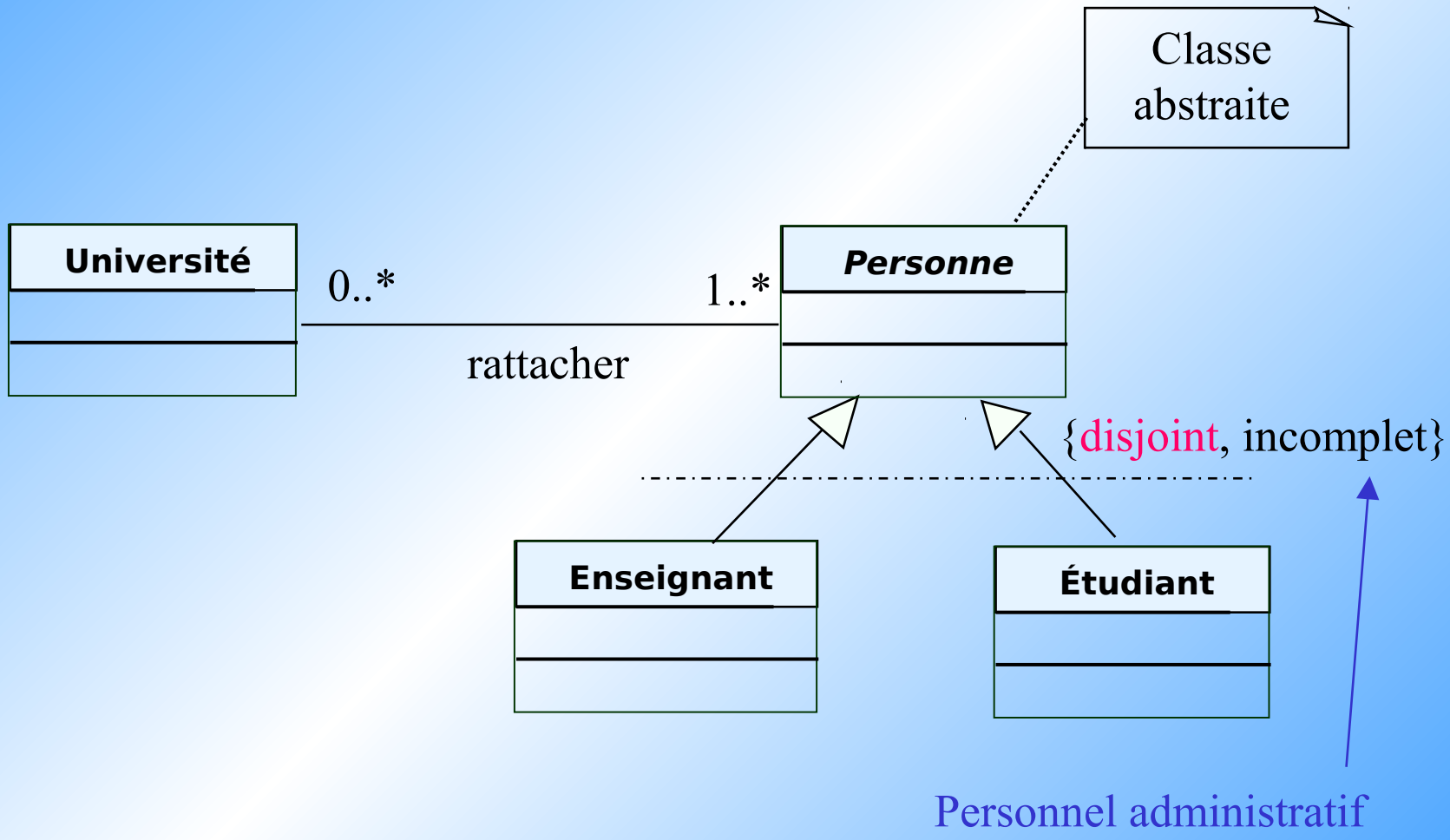


EXERCICE

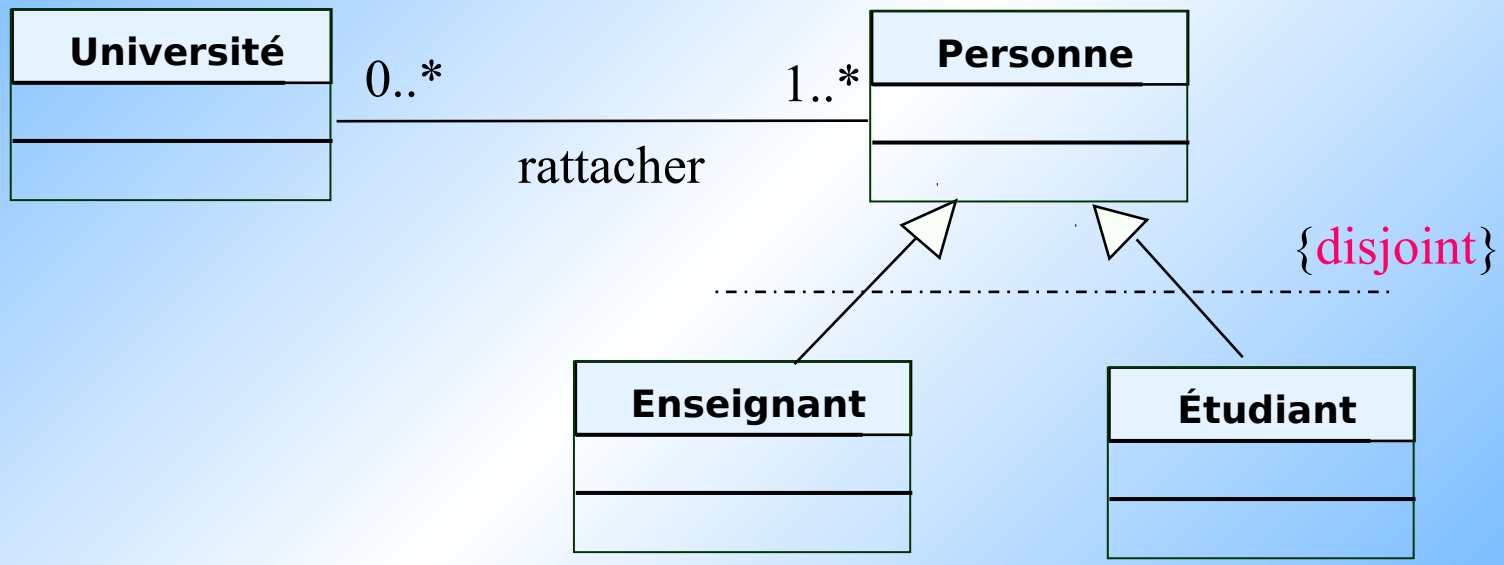
Trouver la hiérarchie de classes qui reflète la contrainte du "*ou exclusif* "



SOLUTION


Classe abstraite
Université
0..*
1..*
Personne
rattacher
{disjoint, incomplet}
Enseignant
Étudiant
Personnel administratif

SOLUTION

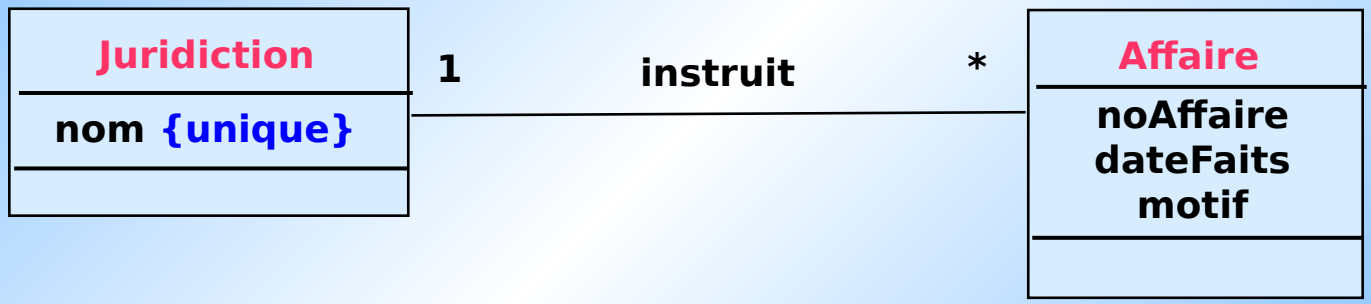

Université
0..*
1..*
Personne
rattacher
{disjoint}
Enseignant
Étudiant

EXERCICE

Différentes juridictions peuvent instruire des affaires. Une juridiction est caractérisée par un nom unique. Une affaire est caractérisée par un numéro, une date et un motif. Le numéro d'une affaire est unique pour une juridiction donnée, mais pas dans l'absolu. En effet, deux affaires disctinctes peuvent se voir attribuer le même numéro si elles sont instruites par deux juridictions distinctes.

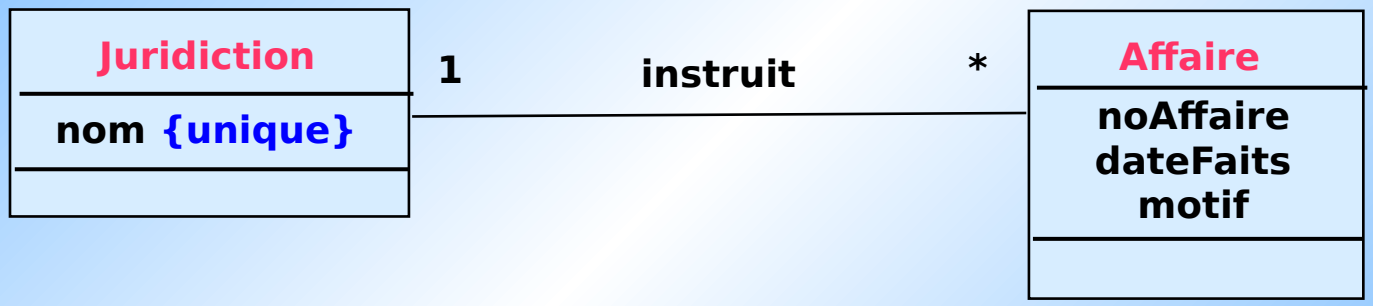
SOLUTION

Que pensez-vous de ce modèle ?



SOLUTION

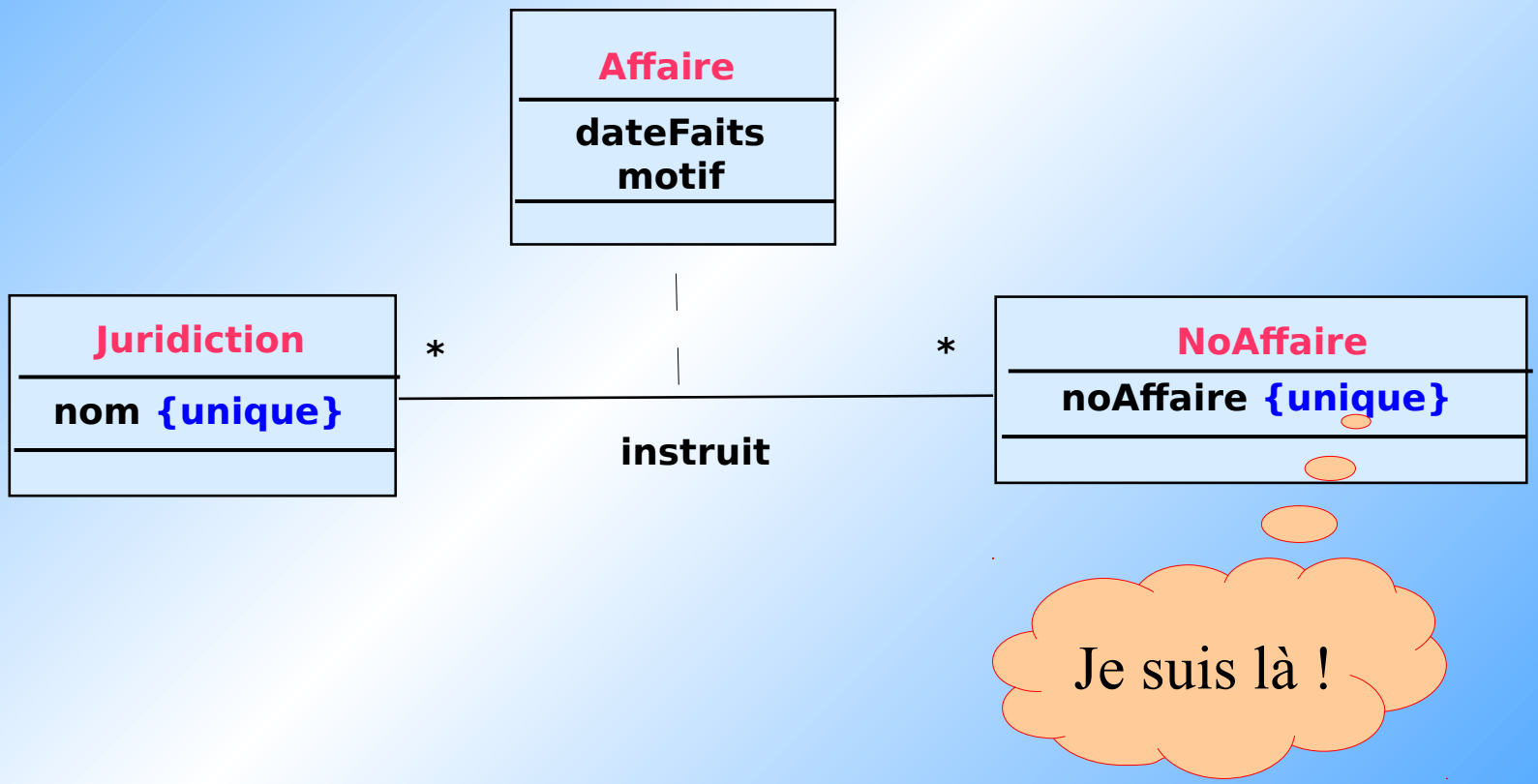
Que pensez-vous de ce modèle ?



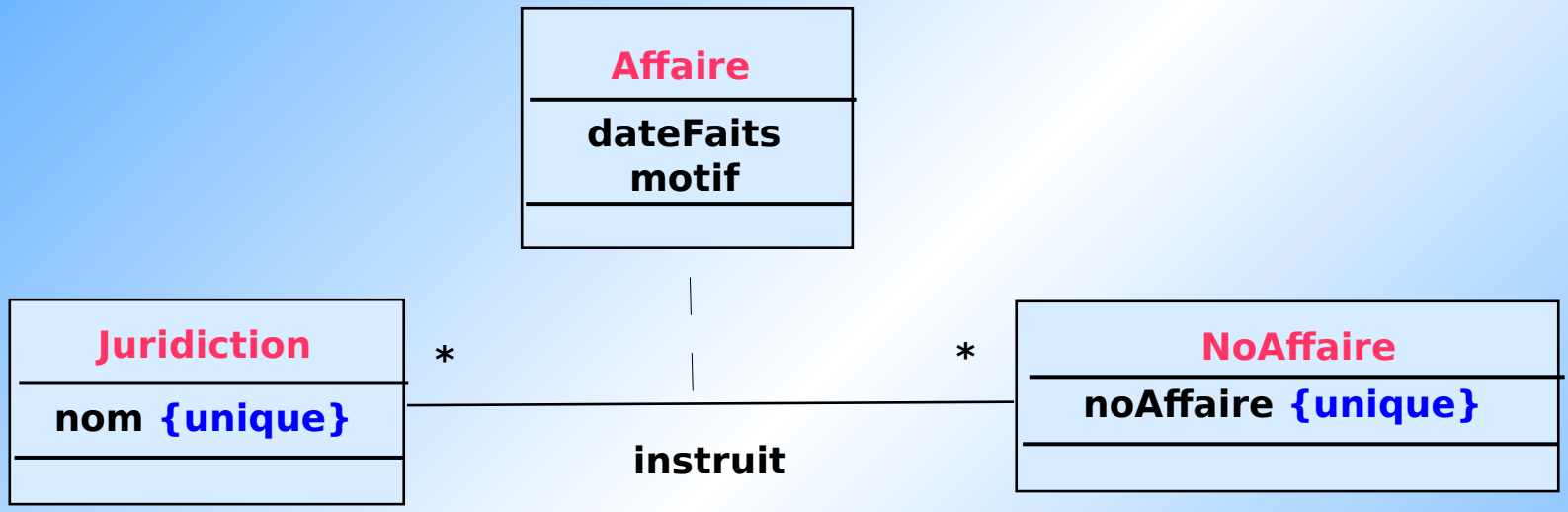
Il ne rend pas compte de l'unicité de noAffaire pour une juridiction donnée

SOLUTION

Que pensez-vous de ce modèle ?



SOLUTION

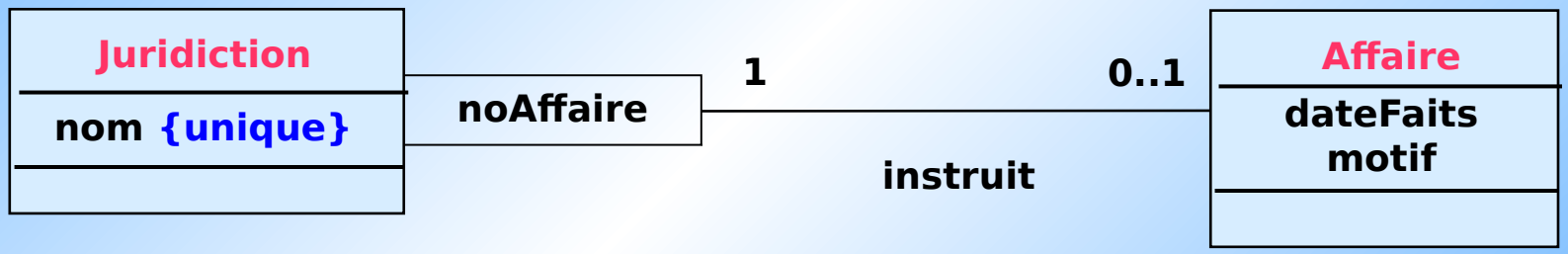


Prend en compte l'unicité de noAffaire pour une juridiction donnée

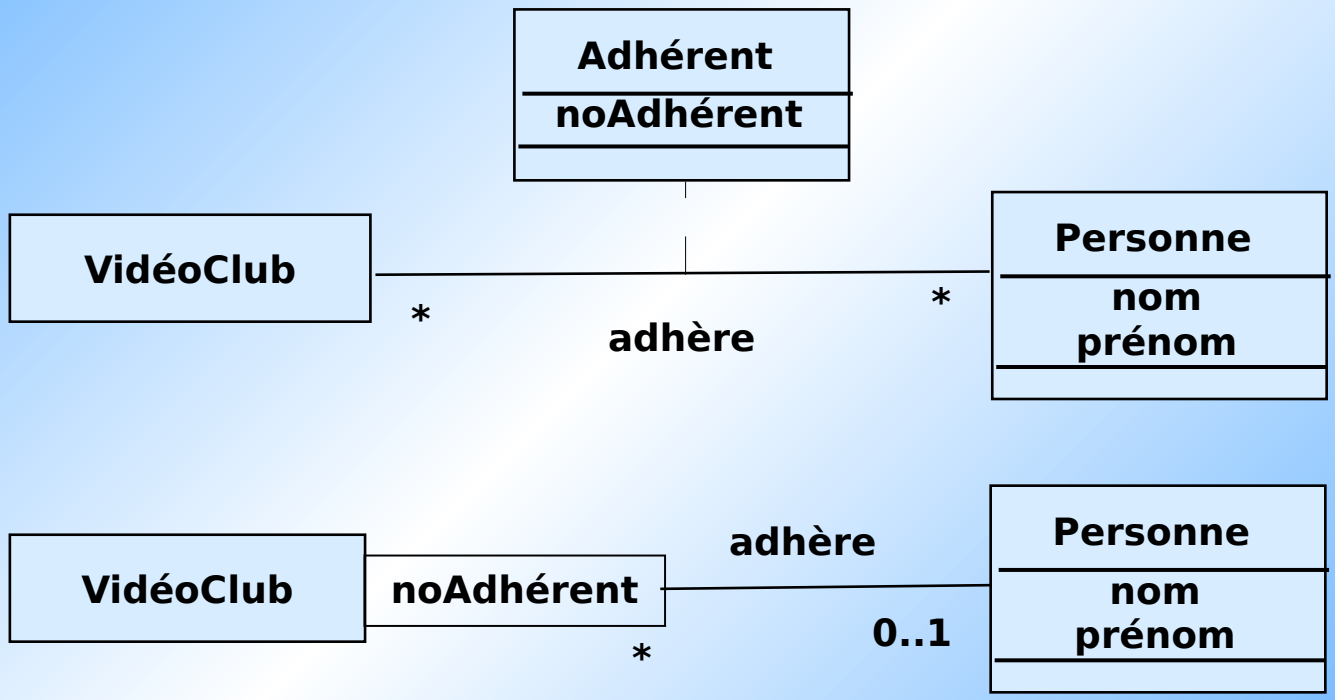
Introduction d'une classe très artificielle : NoAffaire

SOLUTION

Que pensez-vous de ce modèle ?



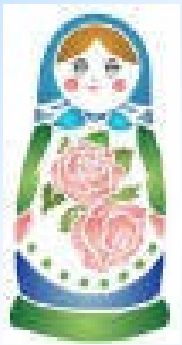
SOLUTION


Adhérent
noAdhérent
VidéoClub
*
adhère
*
Personne
nom
prénom
VidéoClub
noAdhérent
*
adhère
0..1
Personne
nom
prénom

EXERCICE

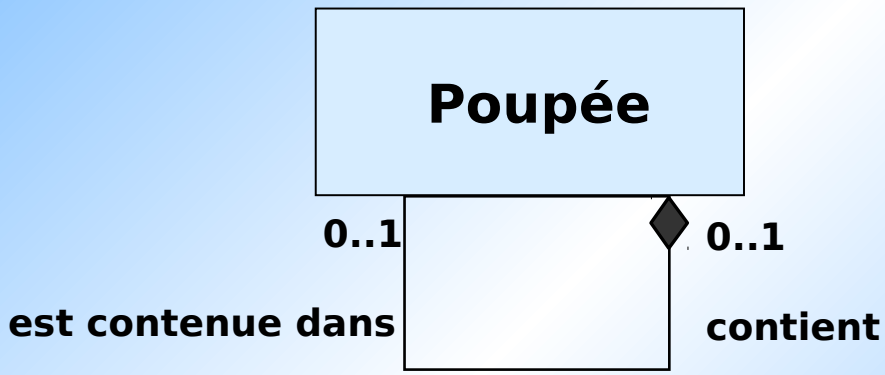
Modéliser les poupées russes.

SOLUTION




$P_1 \longleftrightarrow P_2 \longleftrightarrow P_3 \longleftrightarrow P_4$

SOLUTION



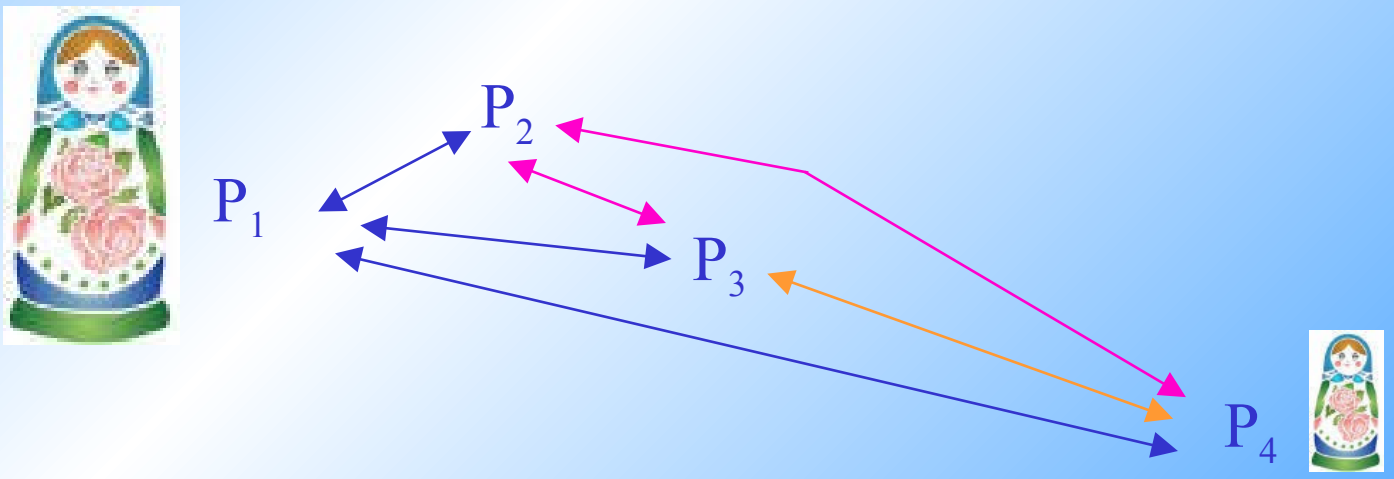
$P_1 \longleftrightarrow P_2 \longleftrightarrow P_3 \longleftrightarrow P_4$

SOLUTION

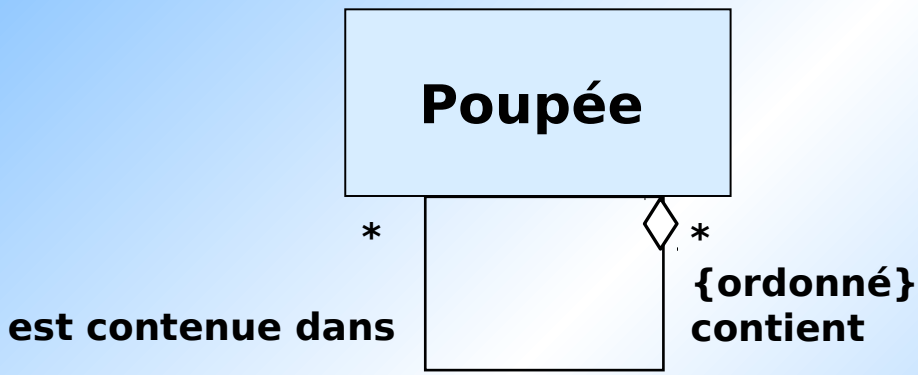
On souhaite connaître :

- toutes les poupées qui sont contenues dans une poupée
- toutes les poupées la contenant

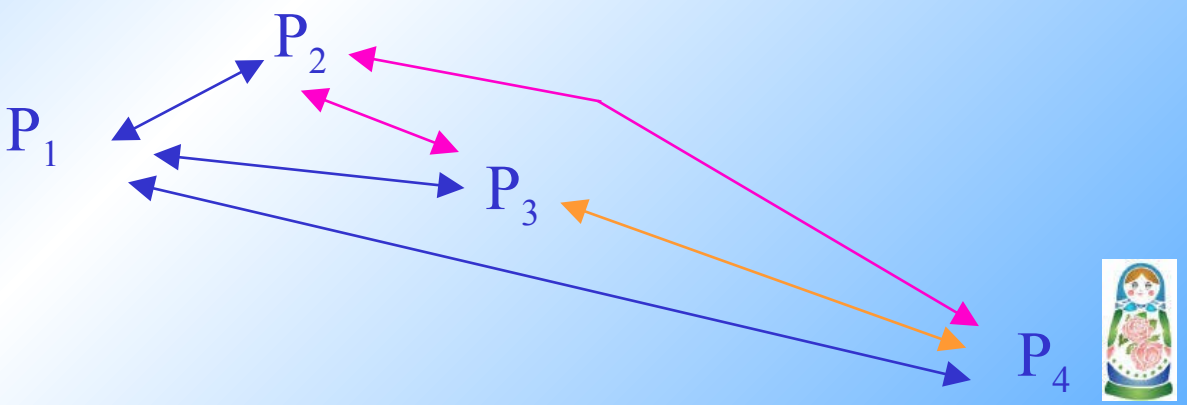
avec la notion d'ordre



SOLUTION



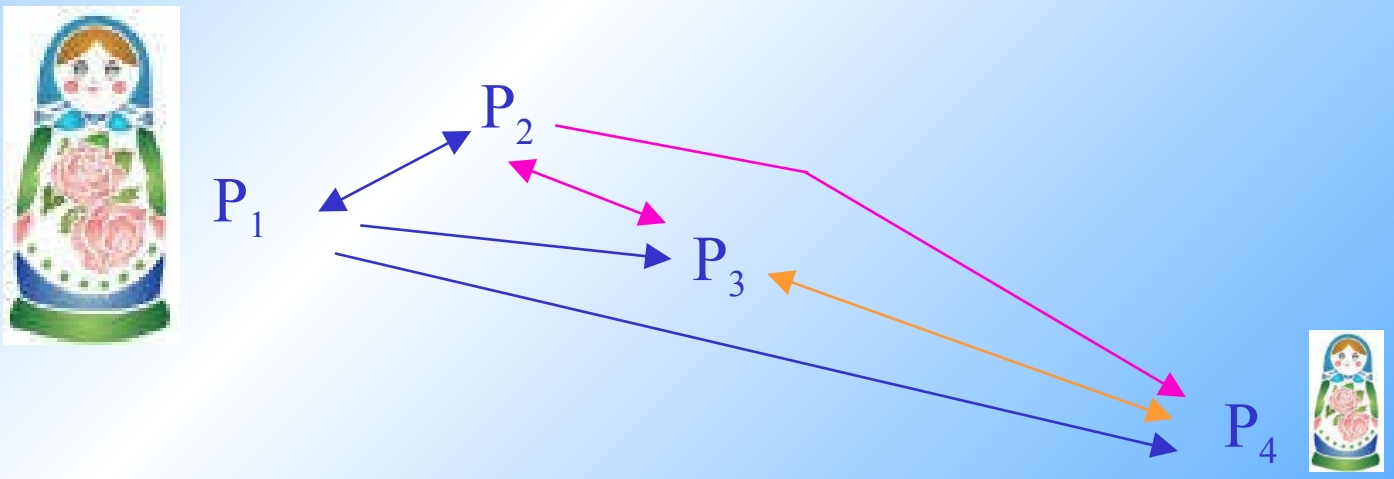
Existence d'un ordre (*petite poupée vers grosses*)



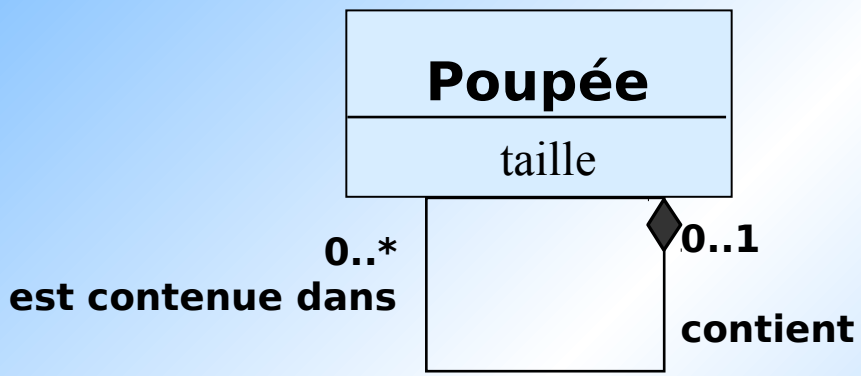
SOLUTION

On souhaite connaître :

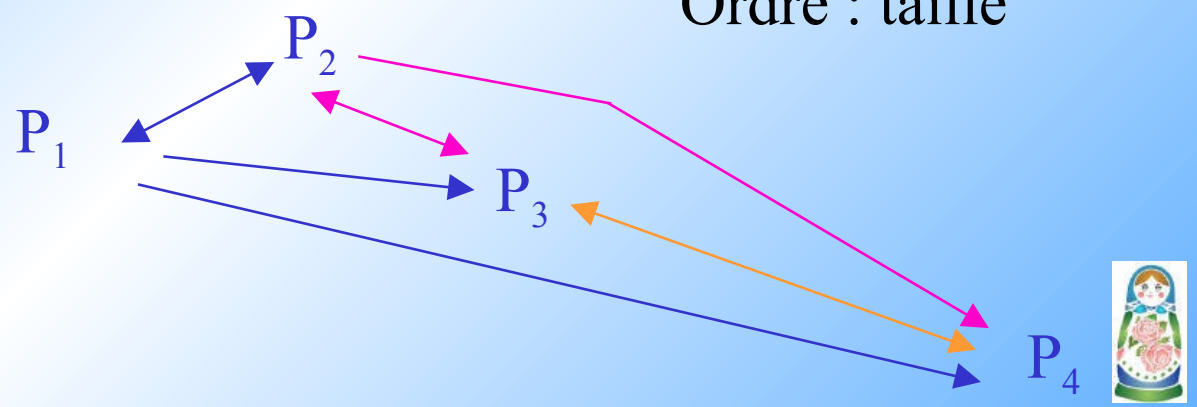
- toutes les poupées qui sont contenues dans une poupée
- la poupée la contenant de taille immédiatement supérieure



SOLUTION

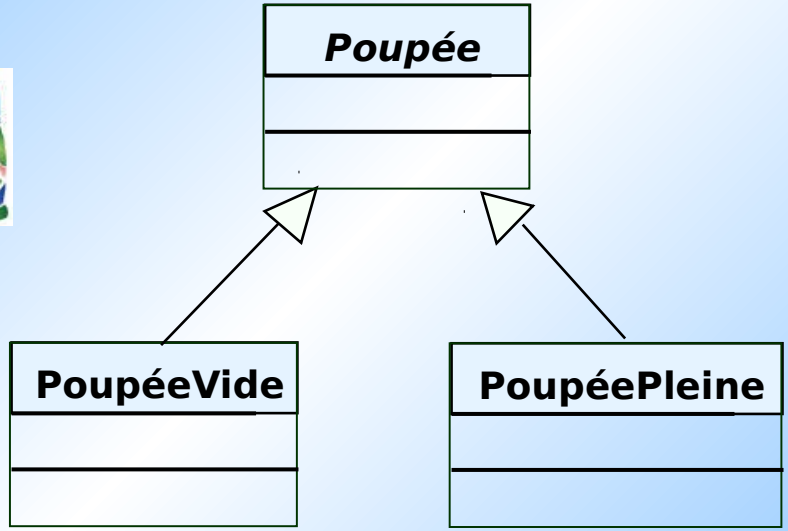


{PoupéeComposite.taille > PoupéeComposant.taille}



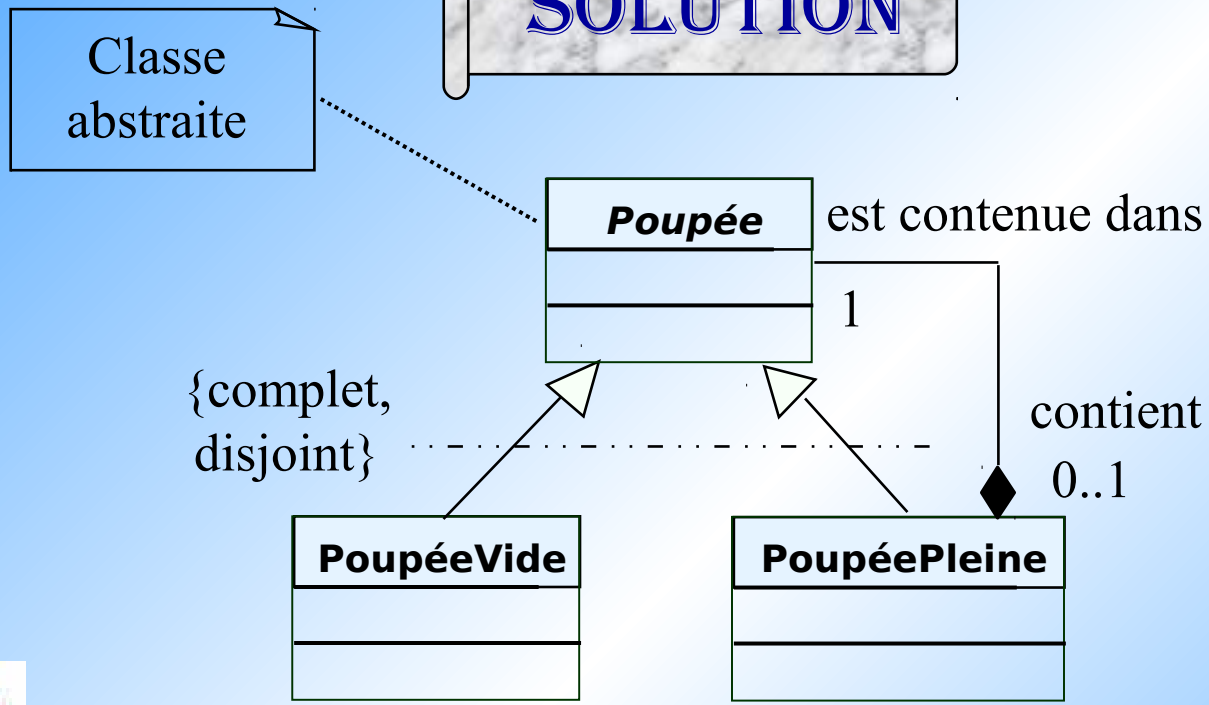
EXERCICE

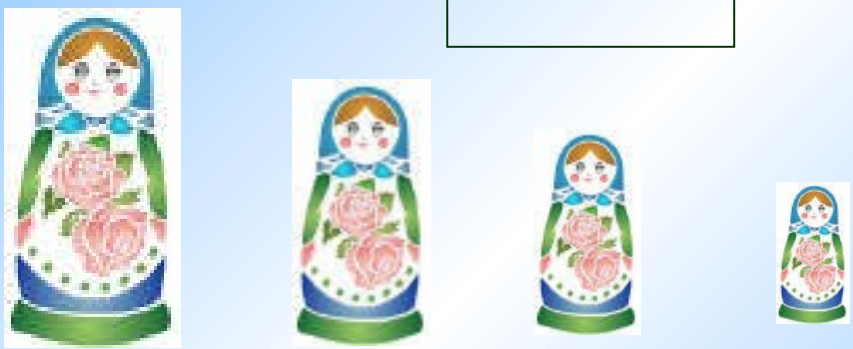
Modéliser les poupées russes avec cette hiérarchie.



Une poupée pleine peut contenir au moins une poupée
Une poupée vide ne contient pas de poupée

SOLUTION





$P_1 \longleftrightarrow P_2 \longleftrightarrow P_3 \longleftrightarrow P_4$

EXERCICE

Modéliser l'énoncé suivant

- Une figure géométrique est composée de figures simples ou composées.

- Une figure composée est constituée de plusieurs figures et une figure simple peut être un point, une ligne ou un cercle.

- Une figure peut être dessinée ou translatée.

SOLUTION

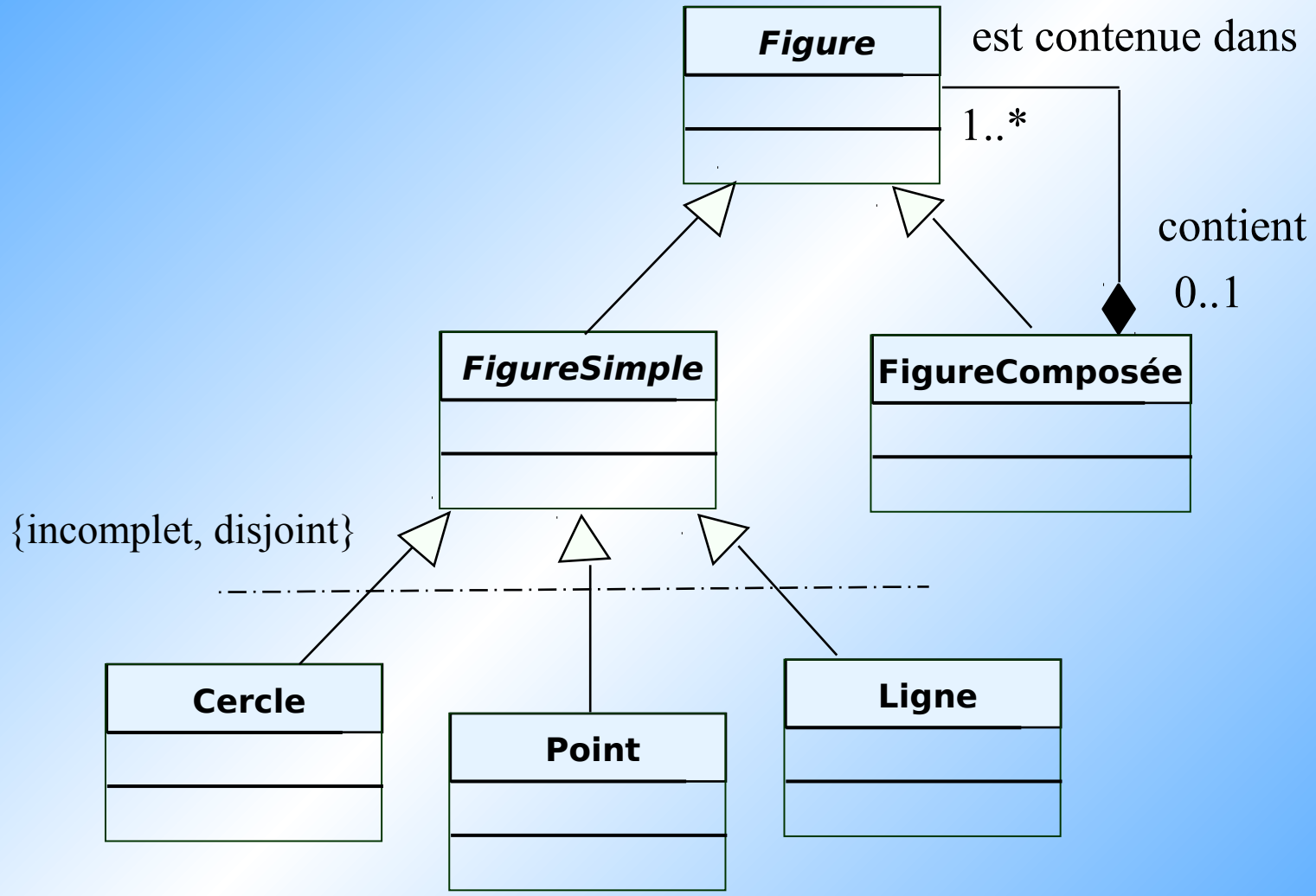


Figure
est contenue dans
1..*
contient
0..1
FigureSimple
FigureComposée
{incomplet, disjoint}
Cercle
Point
Ligne

SOLUTION

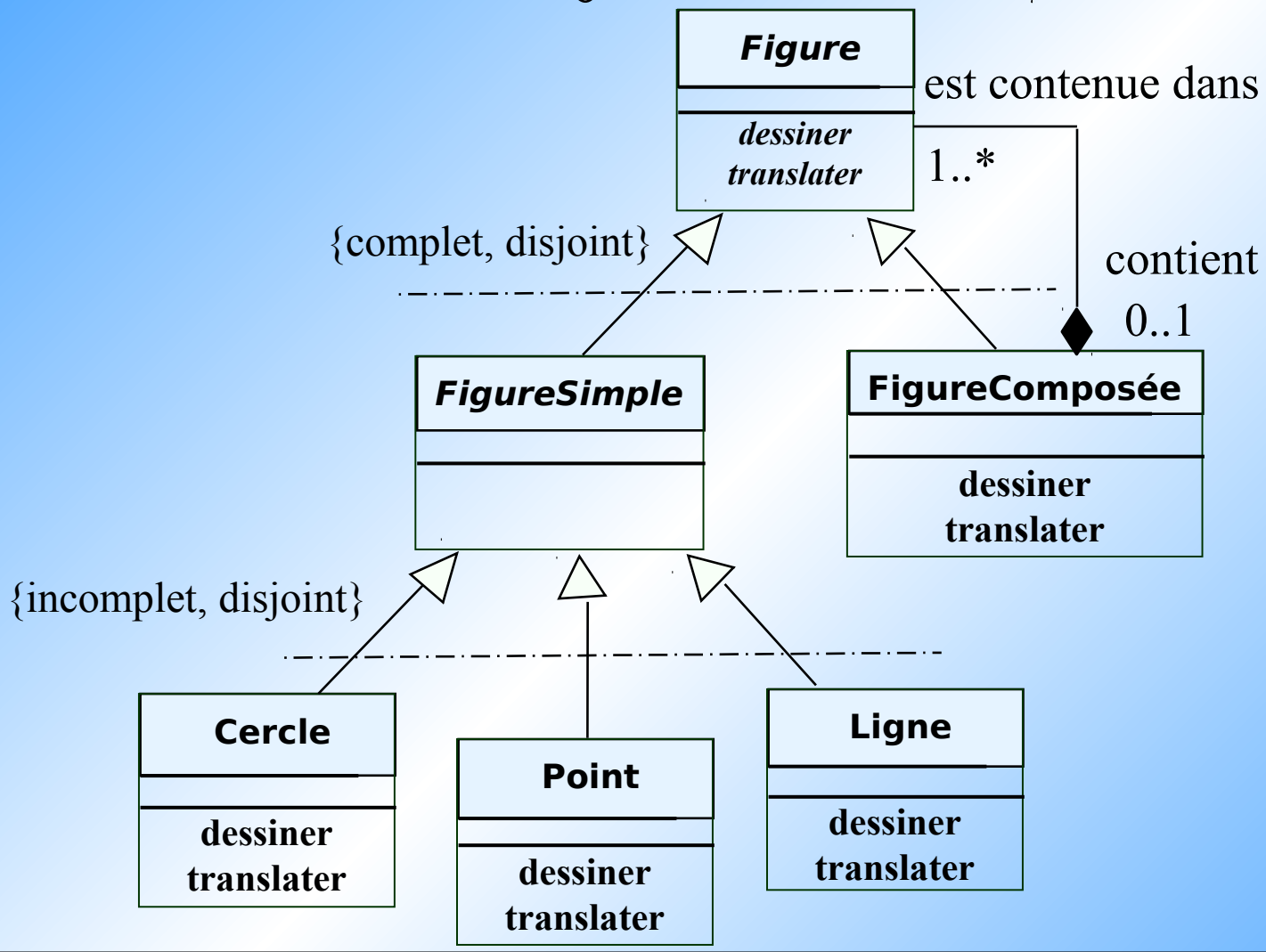


Figure
dessiner
translater
est contenue dans
1..*
{complet, disjoint}
contient
0..1
FigureSimple
FigureComposée
dessiner
translater
{incomplet, disjoint}
Cercle
dessiner
translater
Point
dessiner
translater
Ligne
dessiner
translater

PATRON COMPOSITE

proposé par Erich Gamma