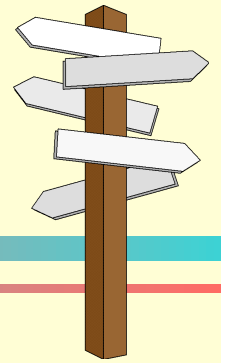
SLR202

Sous-ensemble de diagrammes UML

Etienne. Borde@telecom-paristech.fr
Sylvie.Vignes@telecom-paristech.fr

Institut Mines-Télécom
Télécom ParisTech
Département Informatique et Réseaux
Groupe Systems, Software,Services

Plan



1. Diagrammes de base
   - Diagramme de cas d' utilisation
   - Diagramme de classes
   - Diagramme d' objets
   - Diagramme de séquences
   - Diagramme d' états

2. Autres
   - Diagramme d' activités
   - Diagramme de communication

Diagramme cas d'utilisation

TELECOM ParisTech

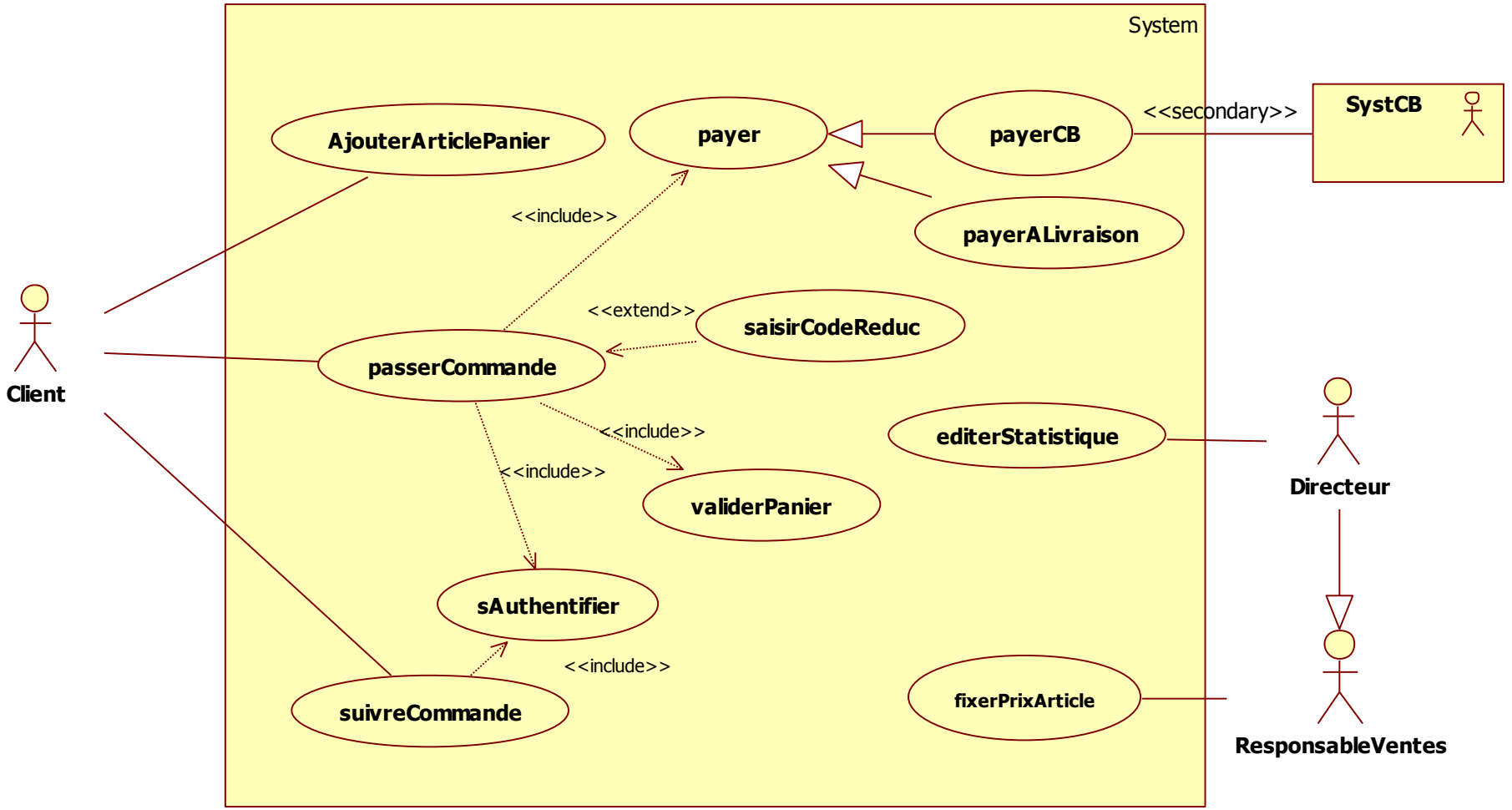
Modélisation des besoins

- Avant de développer:
  À quoi va servir le système ?
  - son objectif ?
  - les fonctionnalités attendues ?
  - qui va utiliser le système ?

- Modéliser les besoins pour « à gros grain »
  - Faire l' inventaire de toutes les fonctionnalités attendues, si possible dégager les principales
  - Organiser les fonctionnalités pour mettre en évidence des relations (envisager factorisation)

- Modéliser par des diagrammes d' utilisation

Exemple diagramme cas d'utilisation



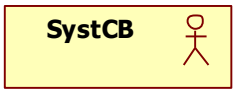
Cas d'utilisation

- Un cas d'utilisation
  - représente un service attendu du système par l'utilisateur
  - désigne un ensemble d'actions plus élémentaires

- Un acteur
  - entité extérieure au système modélisé
  - interagit directement avec lui

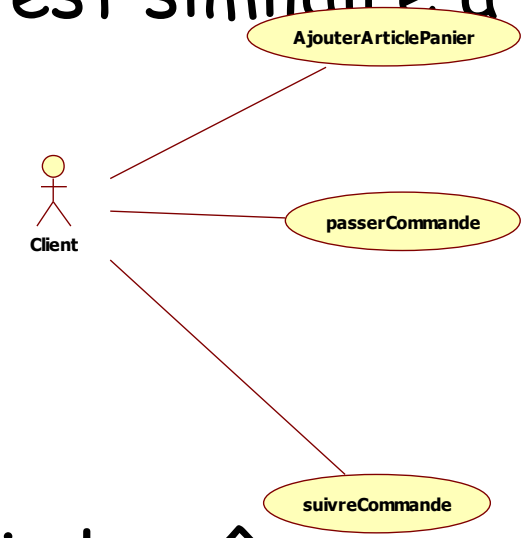

Client


Directeur



Relations entre acteurs et cas d'utilisation

- La relation acteur-cas d'utilisation est similaire à une transaction
  - initialisée par l'acteur
  - service de bout en bout :
    - déclenchement, déroulement, fin

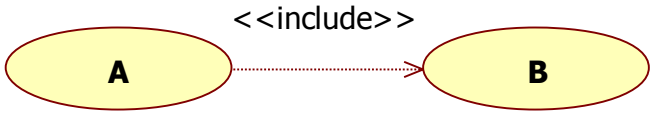


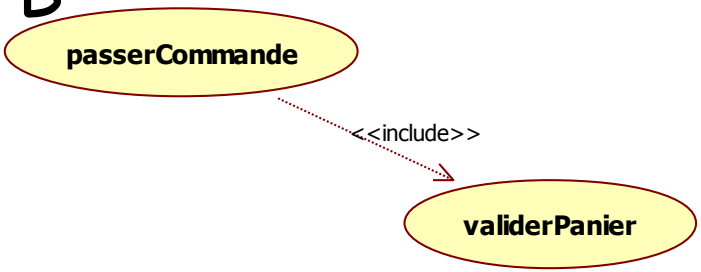
- Un acteur peut utiliser plusieurs fois le même cas d'utilisation
- Un cas peut être utilisé par plusieurs acteurs

- Modélisation par une association

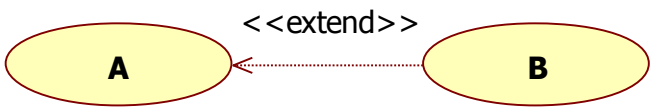
Relations entre cas d'utilisation

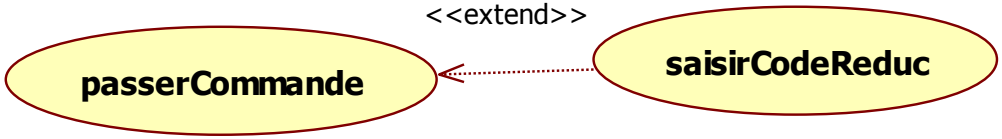
- Inclusion : le cas A inclut le cas B
  - B est une partie obligatoire de A



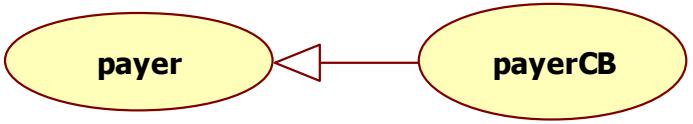


- Extension : le cas B étend le cas A
  - B est une partie optionnelle de A



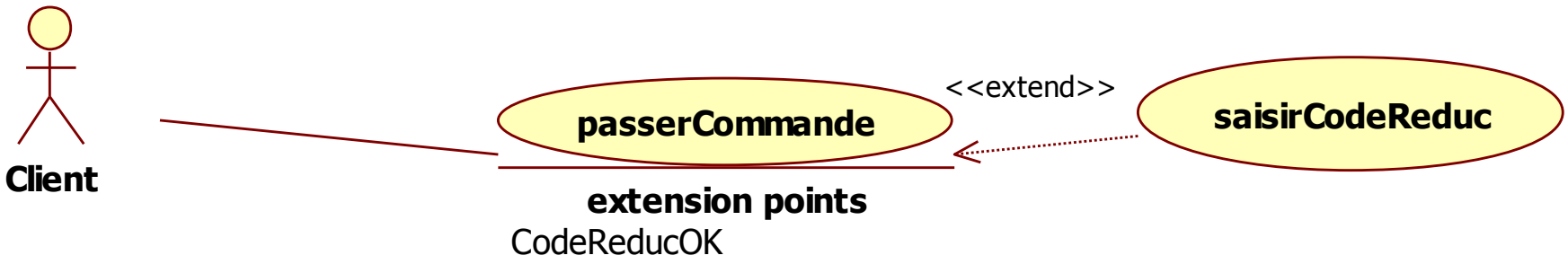


- Généralisation le cas A généralise le cas B
  - B est une sorte de A



Relation extension

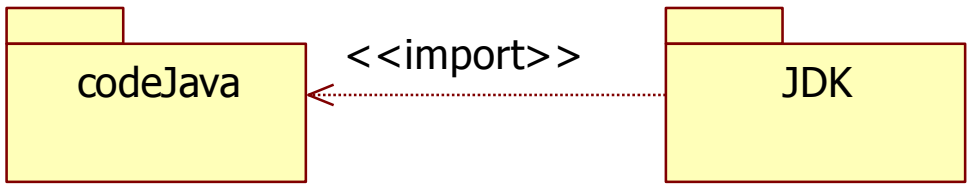
- Spécifier le point d'extension du cas qui peut être étendu par une « garde »



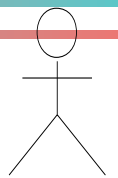
Concept général **de** dépendance

- si A dépend de B, toute modification de B est susceptible d'avoir un impact sur A

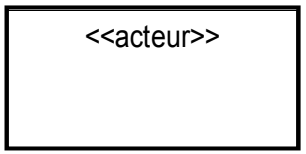
- Modélisé par une flèche pointillée qui indique le sens de la dépendance
- en général la dépendance est « stéréotypée »
  - La sémantique du stéréotype est fixée par le «profil » (contexte d'utilisation) UML
    - Ex cas d'utilisation : « include » « extend »
    - Ex notation UML entre packages de classes Java <<import »



Identification des acteurs



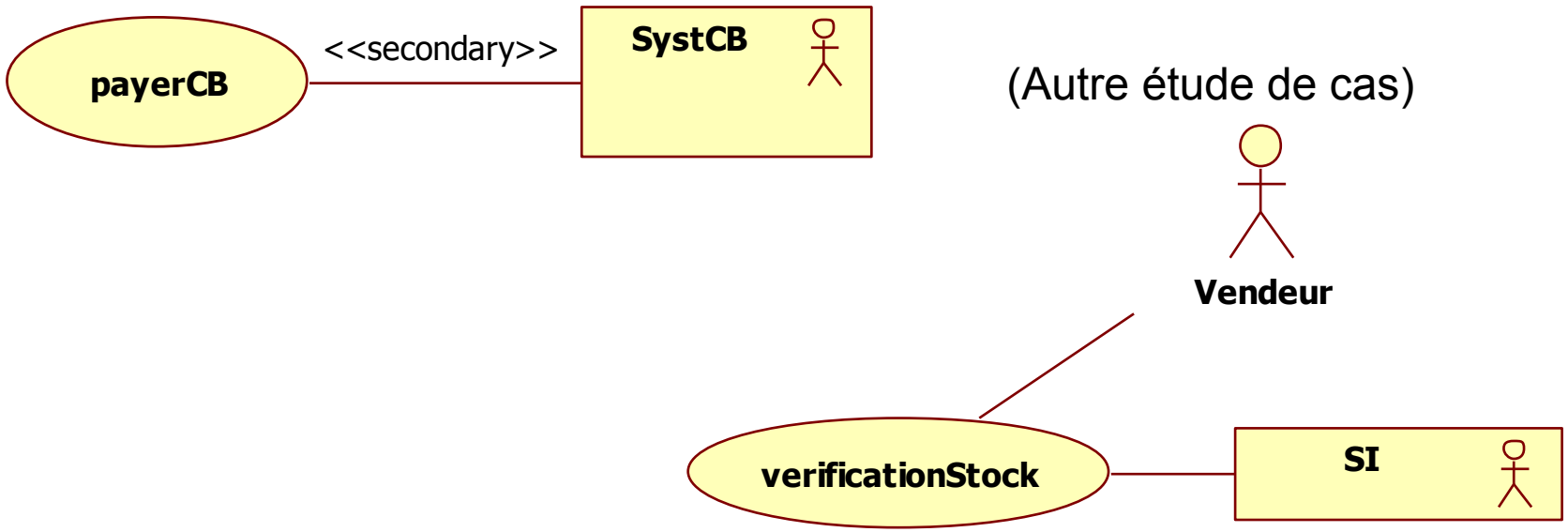
- Acteurs = utilisateurs du système
- Pour les identifier => frontières du système



- Acteur UML = un rôle
  - Une personne physique peut avoir plusieurs rôles
  - Plusieurs personnes peuvent avoir le même rôle
- La acteurs peuvent aussi être:
  - des systèmes informatiques externes
  - des logiciels déjà disponibles à intégrer dans le projet

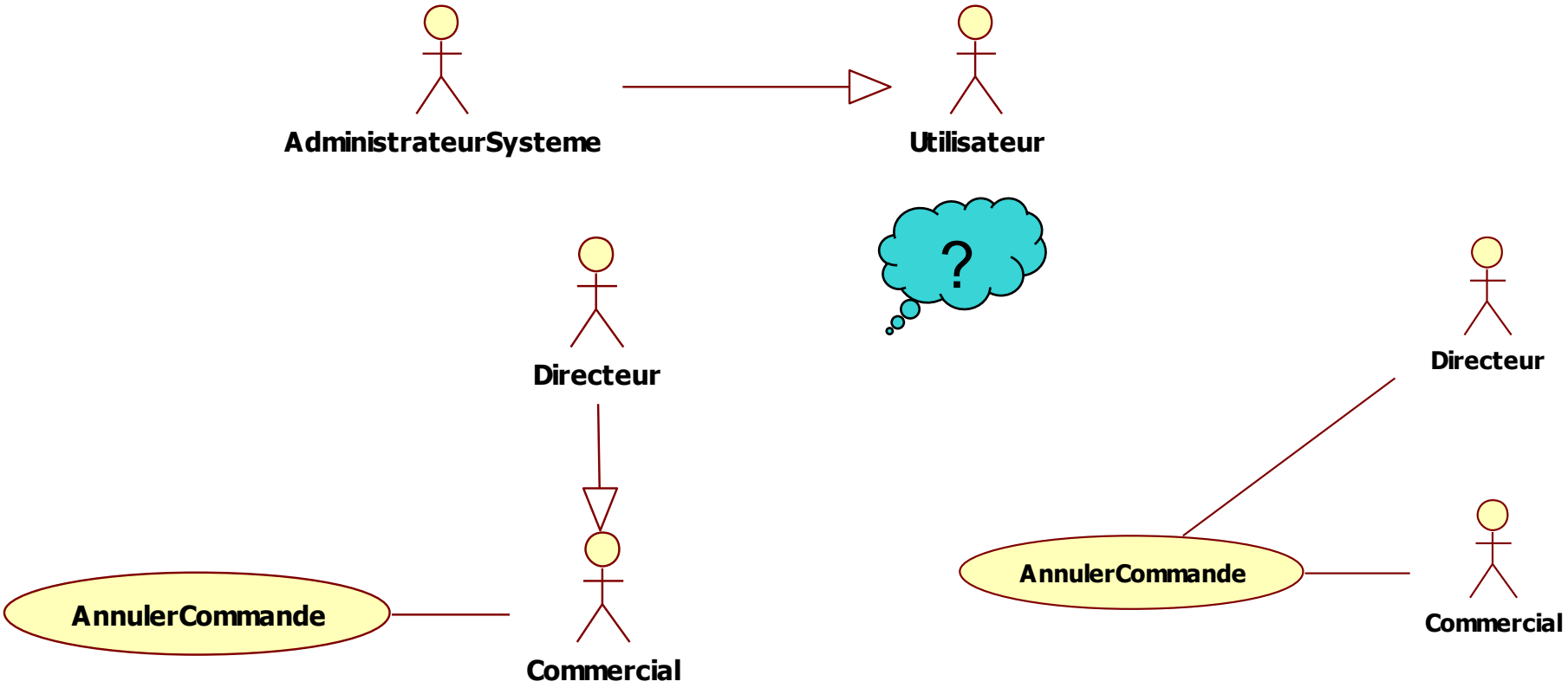
Acteurs principaux et secondaires

- Acteur principal si l'utilisateur a l'initiative de l'interaction avec le cas d'utilisation
- Acteur secondaire est sollicité par le système
  - En général des systèmes informatiques inter-connectés



Relation entre acteurs

- Seule relation : la *généralisation*



Recenser les cas d'utilisation

- Se placer du point de vue de chaque acteur
  - Enumérer les fonctionnalités qu'il attend du système (services), ses droits d'accès
  - Donner un nom significatif
- Surtout NE PAS décrire d'enchaînement, de dialogue
- Se situer au bon niveau d'abstraction
  - Pas trop de détails
  - Limiter le nombre de cas
    - pour rendre compte de l'objectif du système
    - Un cas est plus d'une action élémentaire
  - Enlever les redondances
- Enfin, décrire textuellement le cas
  - ensemble de scénarios nominaux et non nominaux
  - pas de cas d'utilisation indépendant de mauvais usages

- voir démo StarUML

Le diagramme de classes

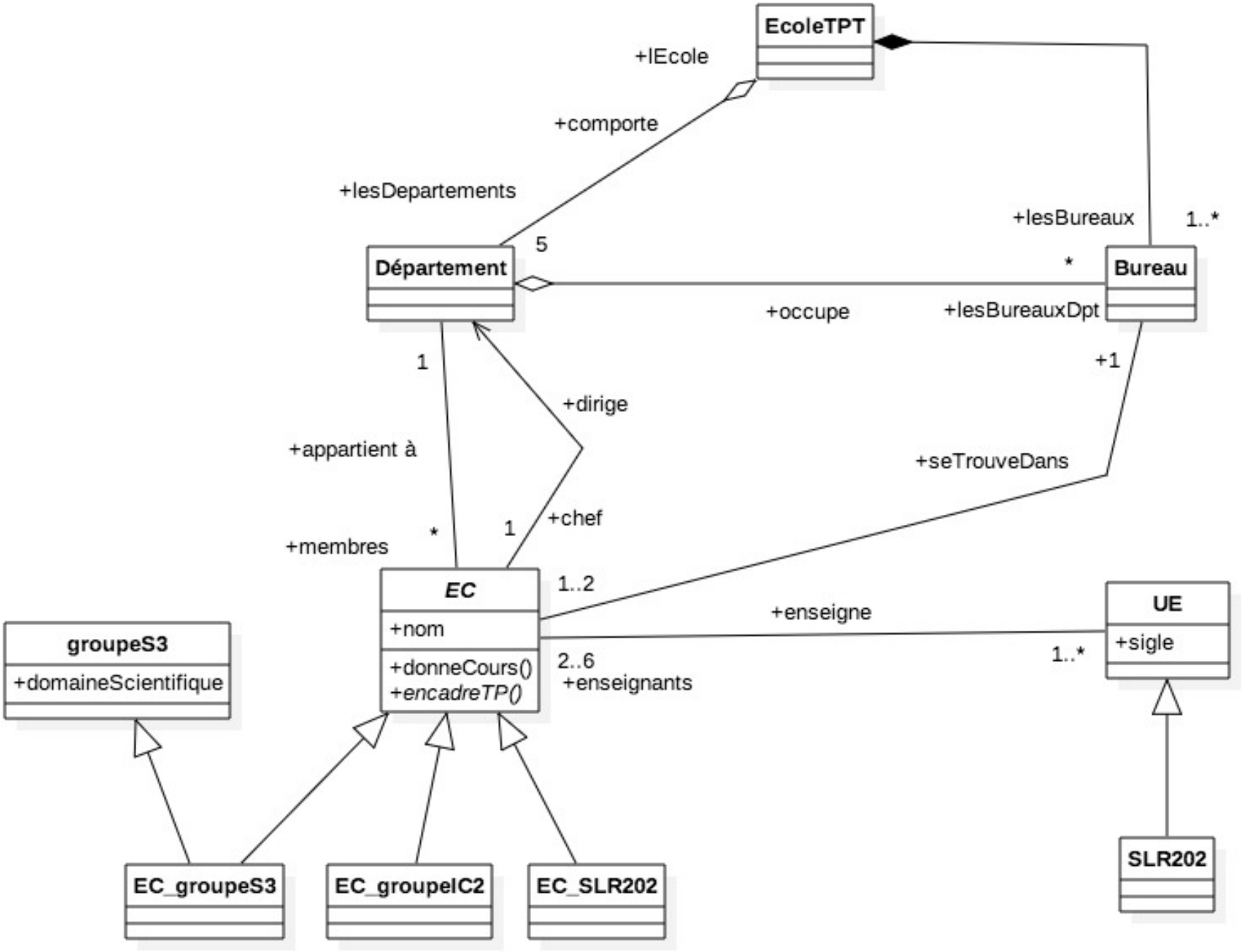
TELECOM ParisTech

Objectif

- Les diagrammes de cas d'utilisation modélisent à quoi sert le système.
- Le système est composé d'objets qui interagissent entre eux et avec les acteurs pour réaliser ces cas d'utilisation.
- Les diagrammes de classes décrivent la structure, par abstraction : classes et associations, des objets et liens dont le système est composé

Concepts supportés par le diagramme de classes

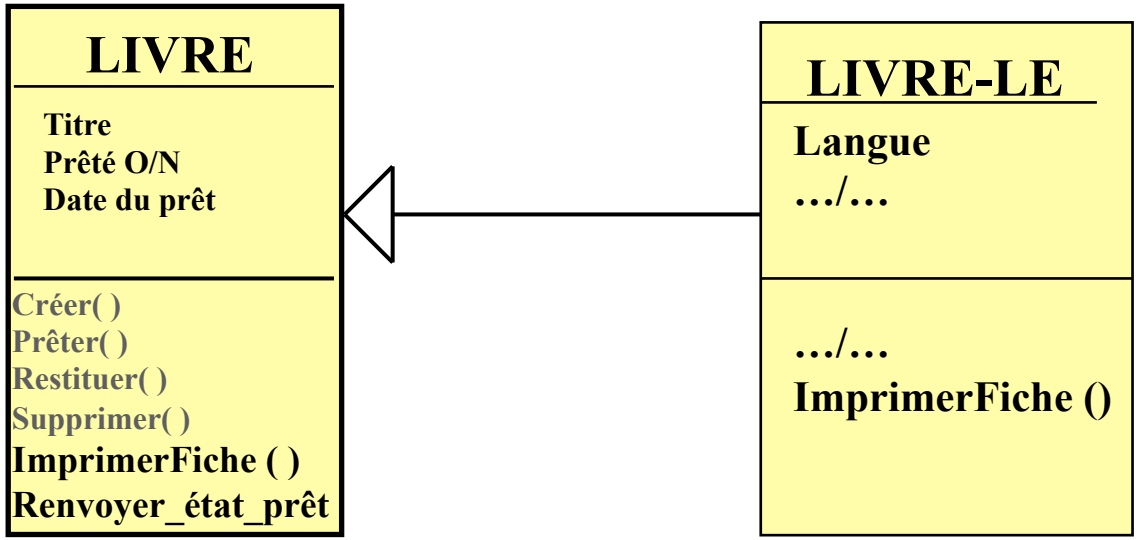
- Les classes :
  - classes réelles
  - classes interfaces
  - classes abstraites
  - classes paramétrables/génériques
- Les associations :
  - associations binaires
  - généralisation/spécialisation
  - agrégation/composition
  - associations n-aires
  - Classe d'association

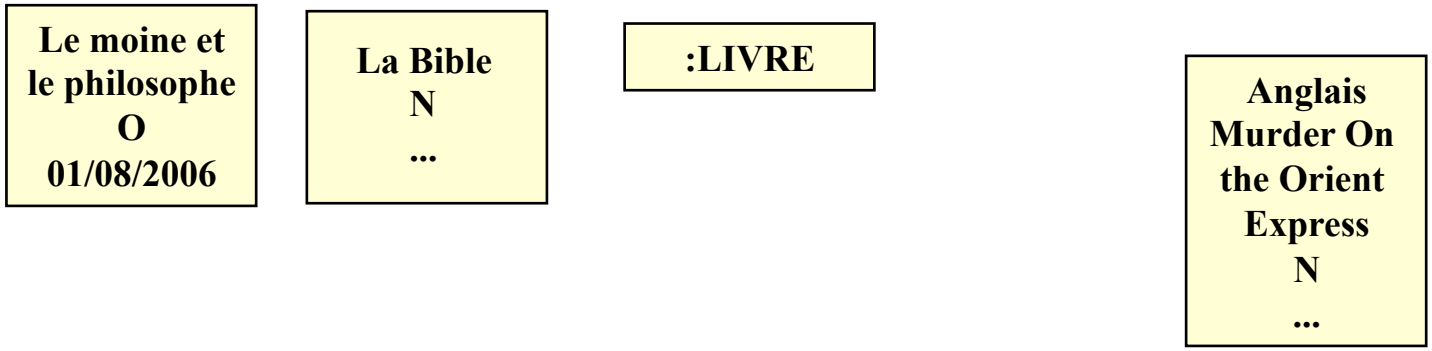

EcoleTPT
+lEcole
+comporte
+lesDepartements
5
Département
+lesBureaux
1..*
*
Bureau
+occupe
+lesBureauxDpt
1
+1
+dirige
+appartient à
+seTrouveDans
+membres
*
1
+chef
EC
+nom
+donneCours()
+encadreTP()
1..2
2..6
+enseignants
+enseigne
1..*
UE
+sigle
groupeS3
+domaineScientifique
EC_groupeS3
EC_groupeIC2
EC_SLR202
SLR202

Objet; classe; héritage;

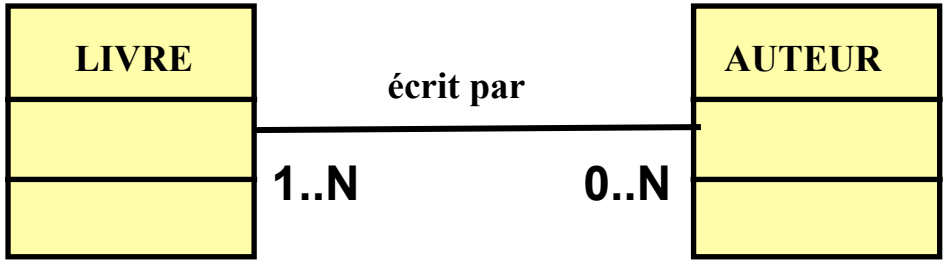
Description
des attributs d'un livre

Opérations pour
la classe «LIVRE»





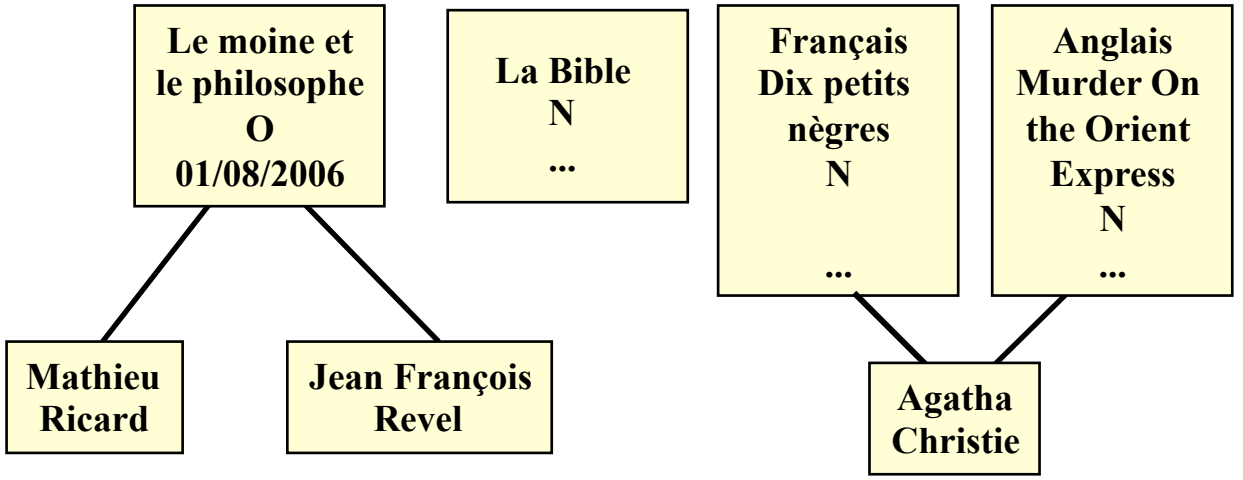
Association entre 2 classes
Liens entre objets



Une association décrit un groupe de liens ayant une structure et une sémantique Commune.

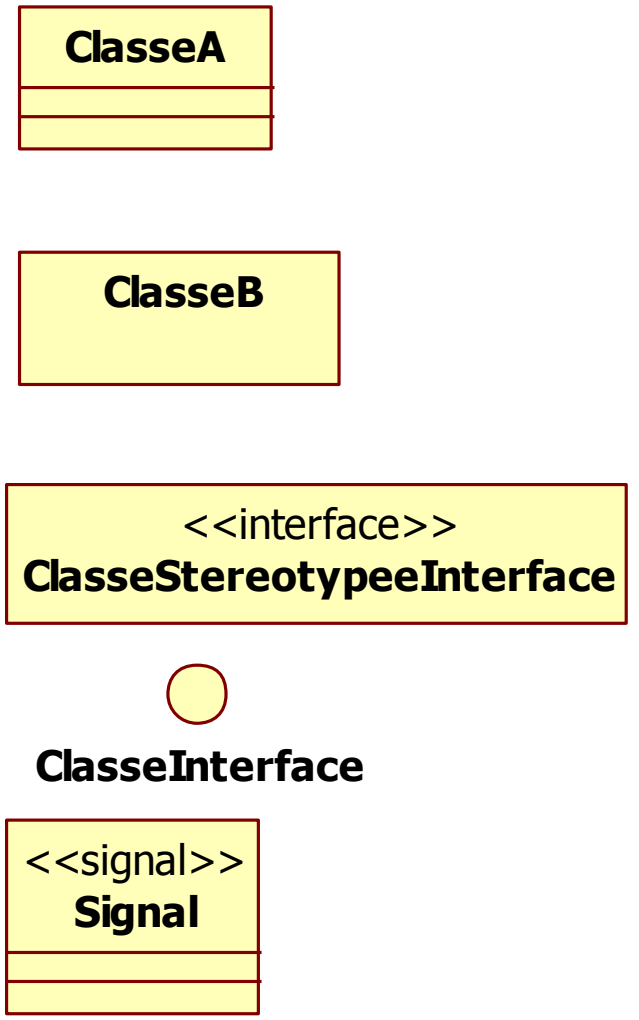
Un lien est une instance d'association.

Un lien est "stable".



Représentation graphique d'une classe

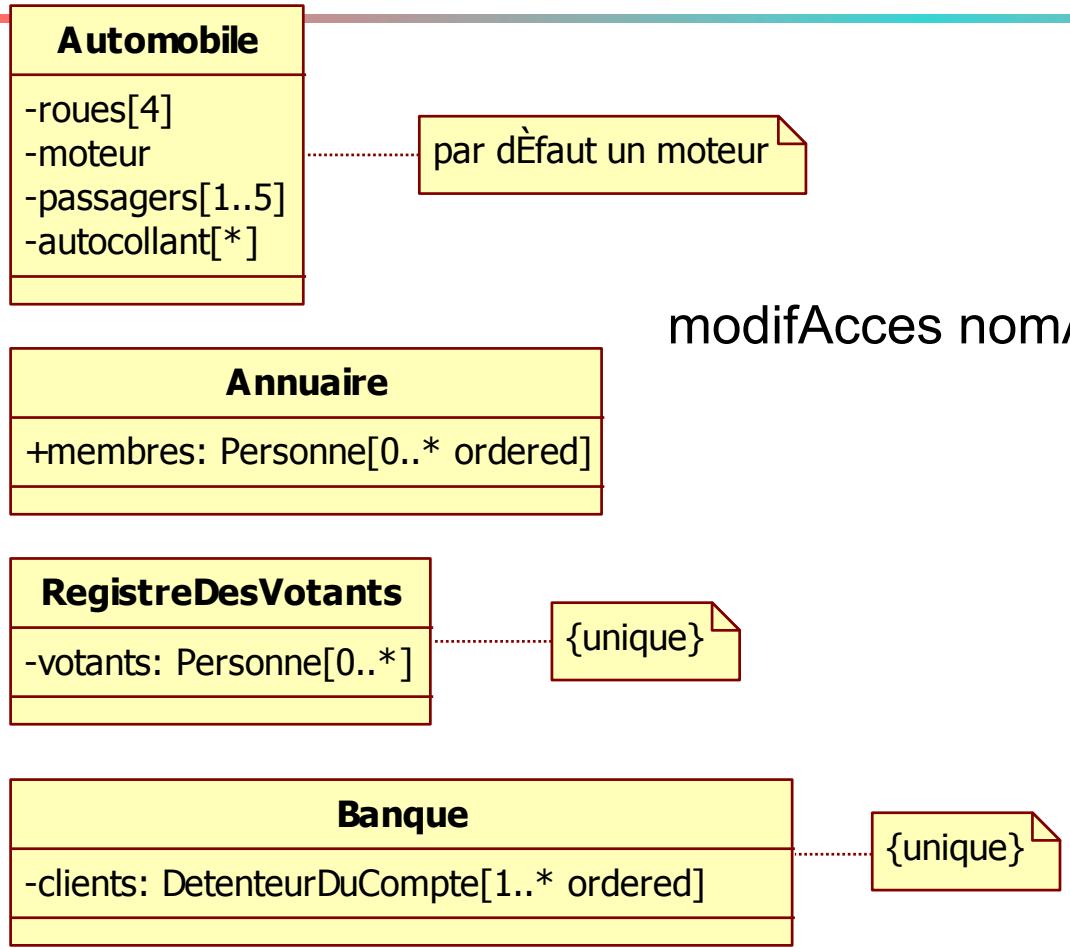
- 1 à n compartiments :
  - nom de la classe : obligatoire
  - 0 ou n attributs
  - 0 ou n opérations
  - responsabilités de la classe, exceptions, …
- Présence de stéréotype
  - <<class>> : implicite
  - <<interface>>
  - « abstract »
  - <<signal>>
  - <<utility>>
  - « acteur »
  - ….



Notion de classeur

- Terme générique UML
- = Groupe d'objets possédant des propriétés communes
- Une classe est un (=hérite de) classeur
- La classe Livre est une instance de classeur
- => *Évocation furtive de méta-modèle*
- Un classeur peut être décoré (ex stéréotype <<abstract>> pour une classe)

Attributs



Syntaxe

modifAcces nomAtt : nomClasse [multi] =valeurInit

Un attribut avec multiplicitÈ dÈfinit une collection (sequence, Set, orderdSet)

Sémantique des opérations

- En UML par défaut: Direction des paramètres
  - in :fourni par la procédure appelante non modifiable
  - inout: fourni par la opération appelante modifiable puis restituée
  - out : pas fournie, calculée puis fournie
  - return : une valeur retournée

- En Java
  - Les valeurs sont données par l'appelant
  - in
  - Return

- en pratique on choisit une sémantique opérationnelle
- (ex Java)

Encapsulation

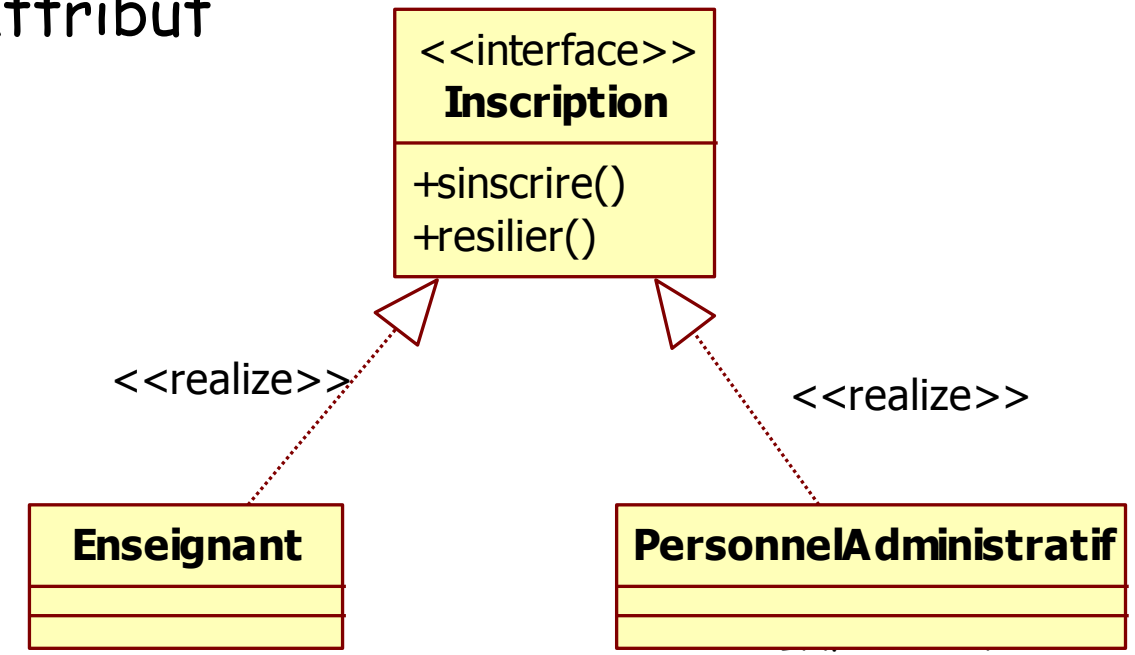
- Visibilité des **classes**, de leurs **propriétés** (attributs et associations) et de leurs opérations

Mot clé	ou Caractère	Signifie visible
public	+	Partout dans le modèle
	~	Dans le paquetage, dans une sous-partie du modèle
protected	#	dans la classe Ou dans ces descendants du paquetage ou d'un autre paquetage
private	-	Uniquement dans la classe

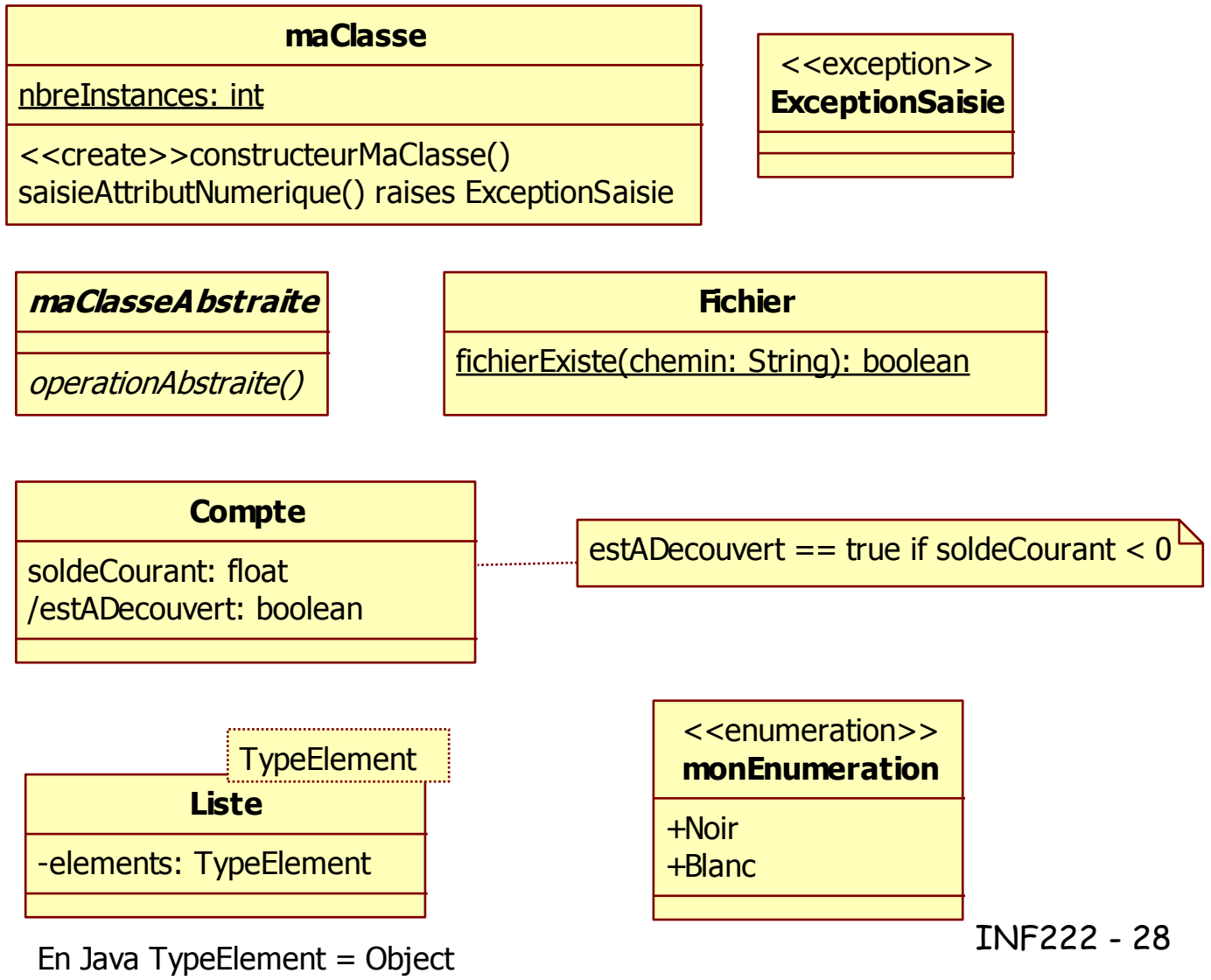
Classe Interface

- Exemple de stéréotype prédéfini <<interface>>

- Regroupe des opérations assurant un service cohérent
  - n'y figurent que opérations publiques (et abstraites)
  - En UML, des propriétés
  - En Java, pas d'attribut



Concepts avancés sur les classes



En Java TypeElement = Object

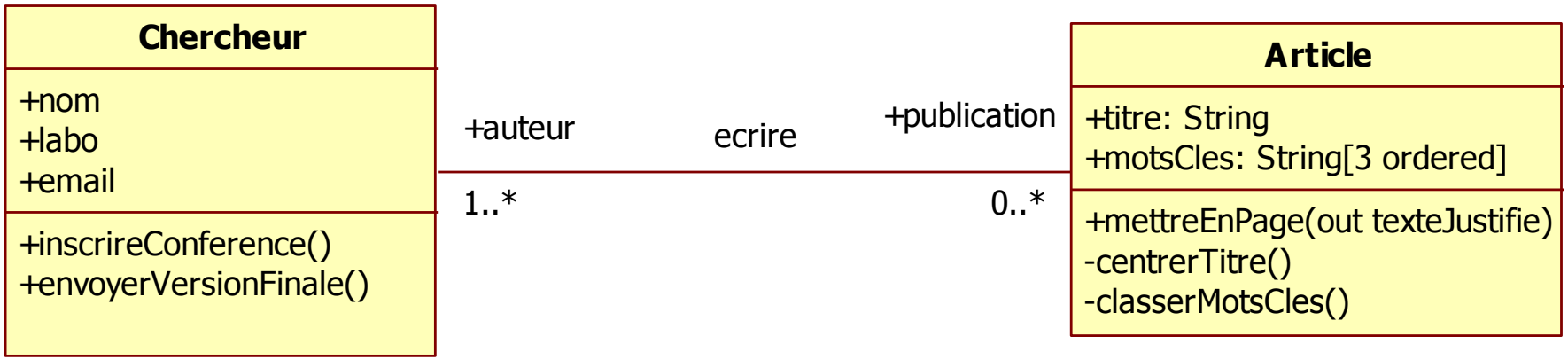
Les associations

- représentent des liens **statiques/structurels** entre objets et à longue durée de vie

- sont de 3 types et traduisent :
  - Une relation entre deux classes
    y compris
    - d' une classe avec elle-même (réflexivité)
    - ou entre plusieurs classes (on n' utilise jamais!)
  - le concept de spécialisation/généralisation
  - la notion d'assemblage/composition

- Remarque : les associations ne sont en général pas directement supportées par les langages de programmation orienté-objet.
- En quelque sorte, ce sont des attributs de classes représentés par une notation relationnelle

Exemple d'association

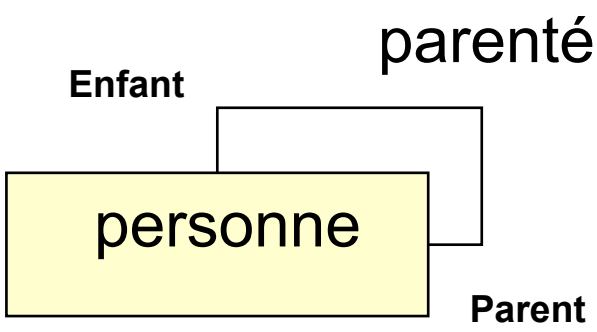


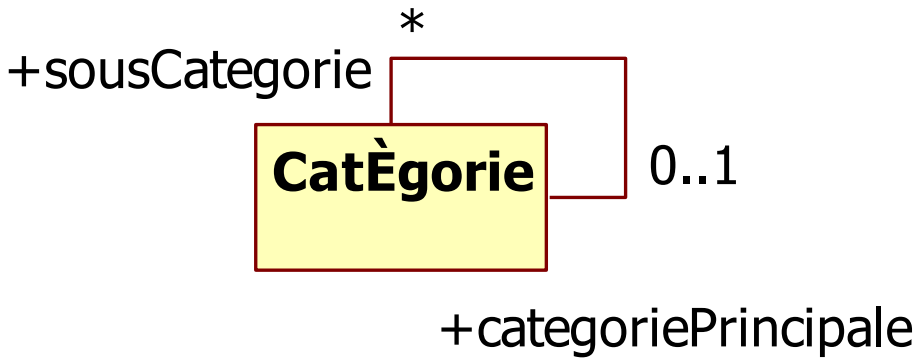
- Un chercheur écrit 0 ou n articles
- Un article est écrit par un ou plusieurs chercheurs
- C'est dans son rôle d'auteur qu'un chercheur écrit un article

Différents types d'associations

- binaire : associe deux classes
- n-aire : associe plus de deux classes

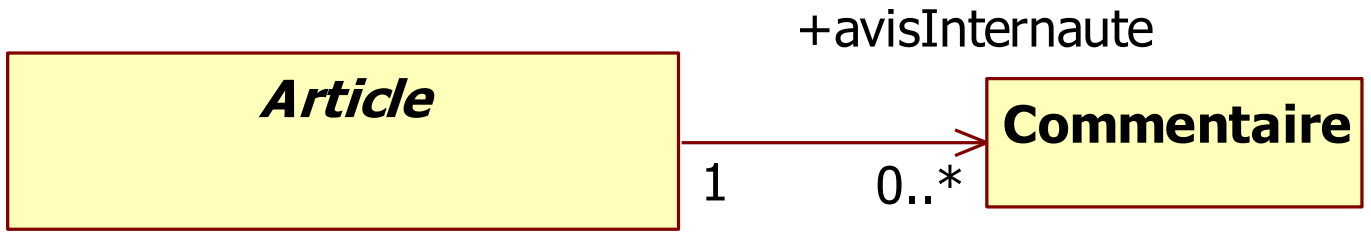
- réflexive : associe une classe à elle-même.
  - Dans ce cas, obligatoire d'indiquer les *rôles* joués par la classe





Navigabilité d'une association

- La navigabilité définit l'accès aux informations
  - Par défaut dans les deux sens
  - Peut être orientée



Les cardinalités des associations expriment

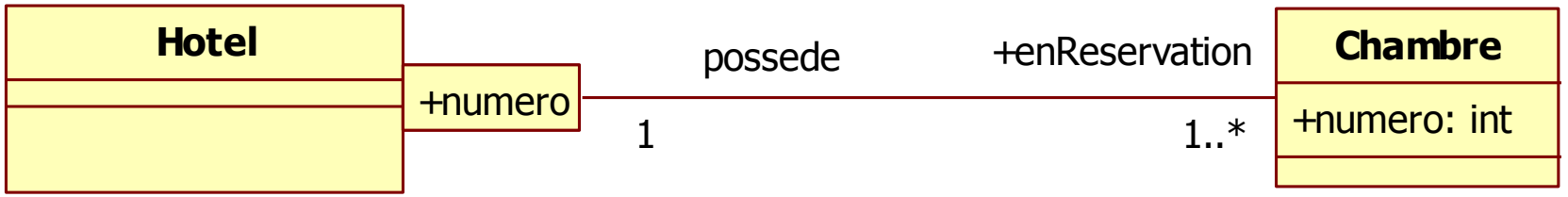
- le nombre d'instances cible rattachées à une instance de la classe source

1	exactement 1
*	0 à n quelconque
0..1	optionnel mais pas multiple
1..*	au moins 1
25..59	intervalle

- Remarque : Si on suppose qu'une classe représente un ensemble d'objets, une association est un sous-ensemble du produit cartésien des ensembles d'objets des classes qu'elle relie.

Diagramme de classes
association qualifiée

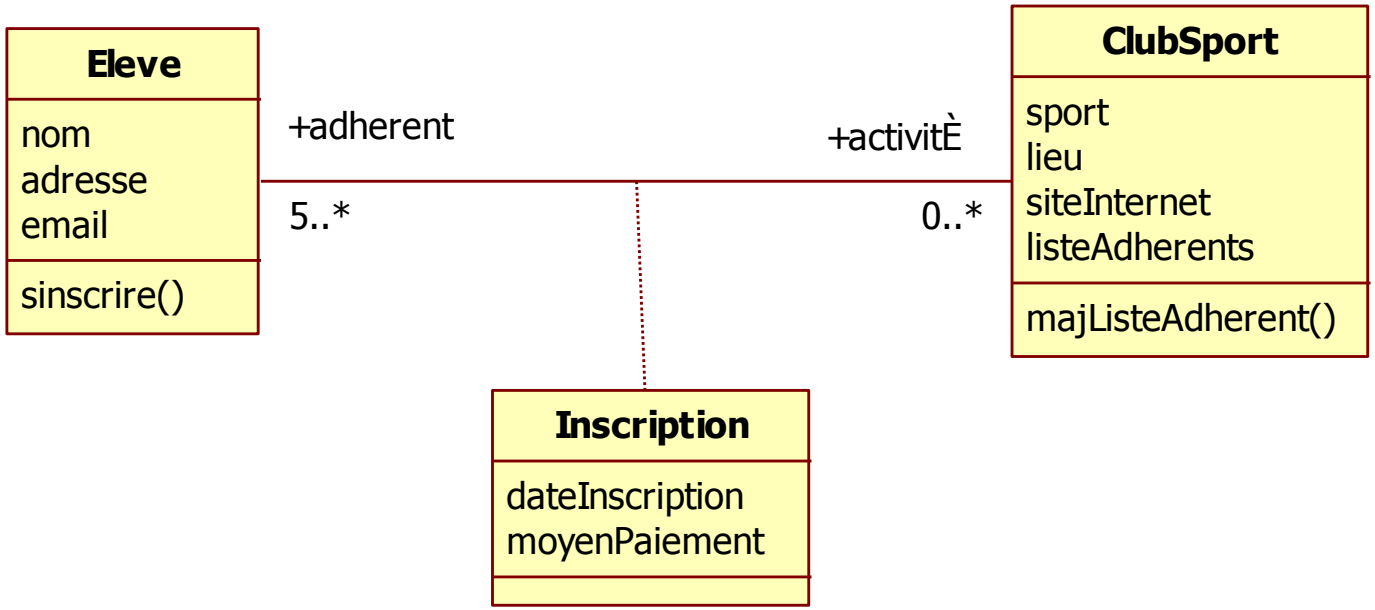
- Un qualificatif d'association est un attribut qui permet par l'association d'identifier un objet particulier à atteindre



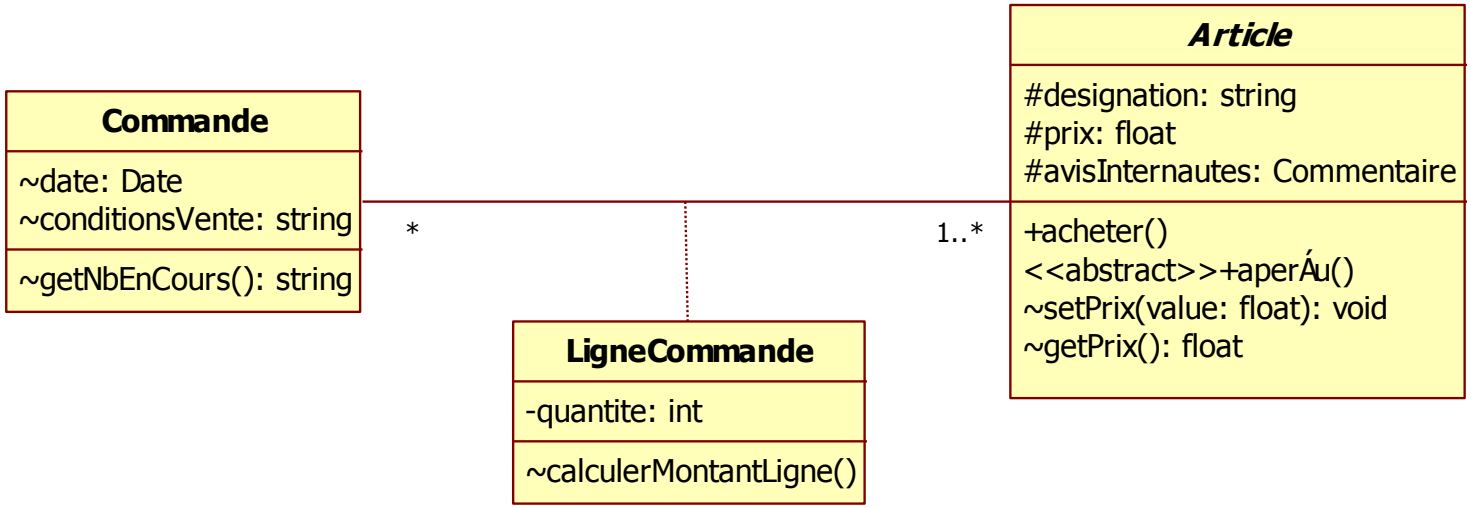
  - Numero est un attribut de chambre
  - La conjonction d' un attribut ID_Hotel de la classe Hotel et de l' attribut numero est équivalent à une clé primaire de Base de Données.

Exemple classe d'association

- Représente le n-uplet des instances mises en jeu dans l'association
- Un étudiant peut s'inscrire à plusieurs clubs; pour chaque inscription on enregistre la date et le paiement

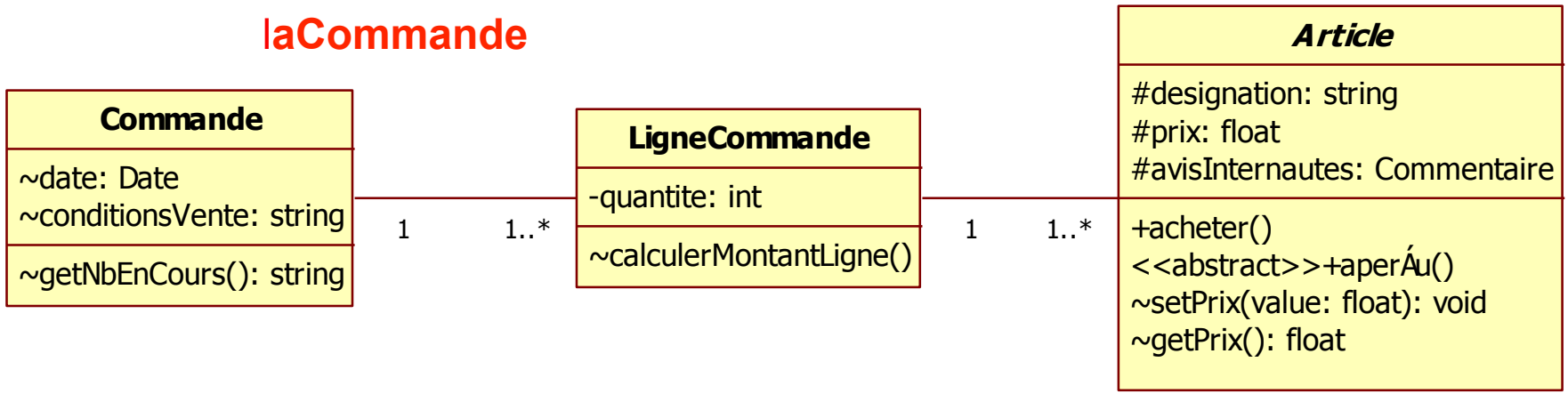


Comment transformer une classe d'association pour l'implémenter en Java?



Dans le modèle il y a des factures; Une facture est une suite de lignes.
Une ligne concerne un article en n exemplaires.
On veut calculer le montant par ligne, et le montant de la facture au fur et à mesure lorsque les lignes sont complètes

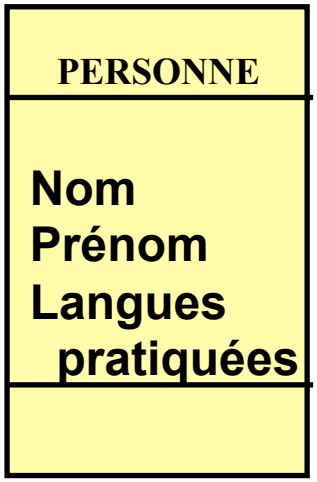
Une proposition de transformation



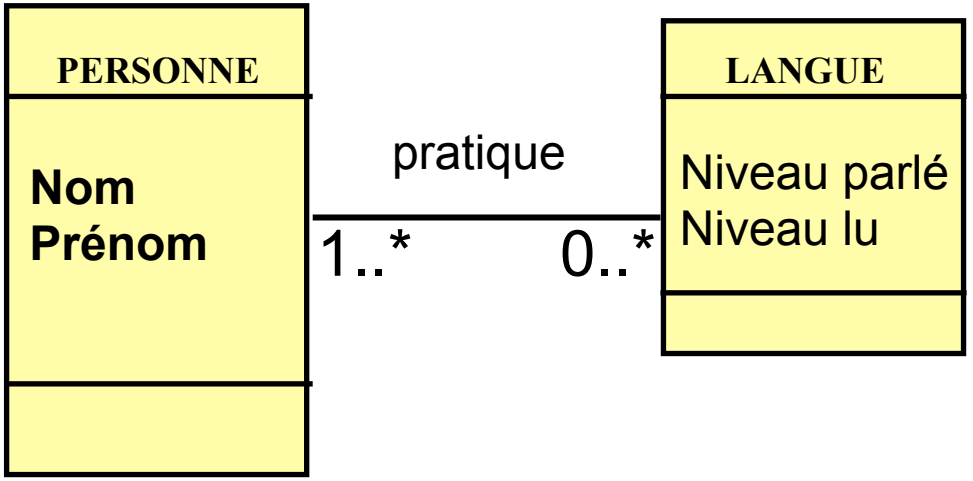
Discuter cette proposition, complétez si besoin et trouvez l'erreur !

Diagramme de classes
Règle de construction 1

- Tout attribut dans une classe doit avoir une valeur unique



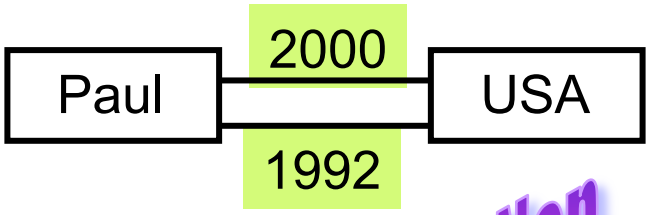
limité!



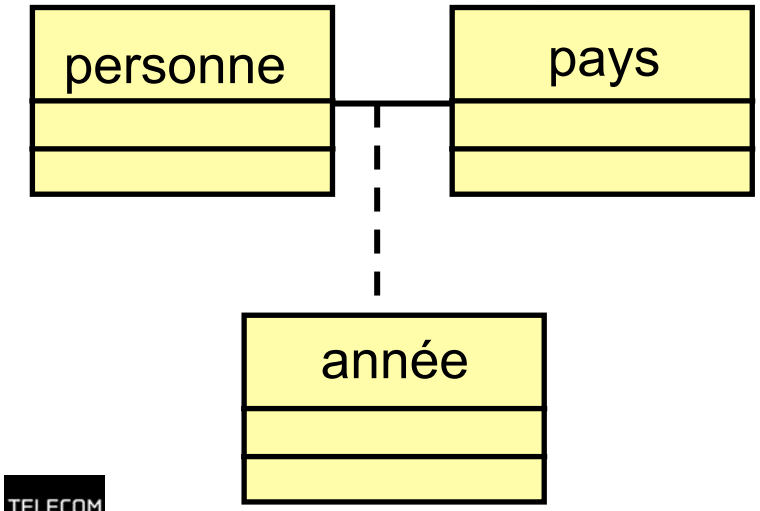
oui

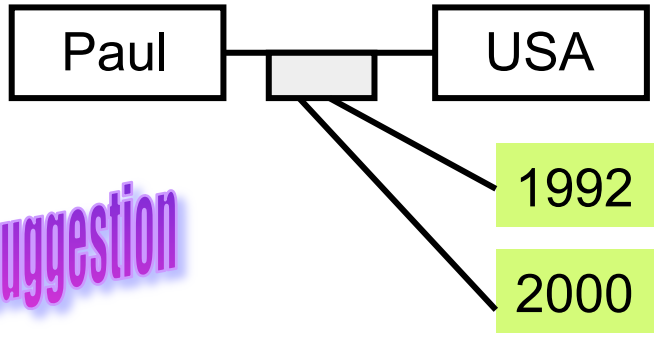
Diagramme de classes
Règle de construction 2

- Entre deux objets, il ne peut exister qu'un lien pour une même association



Non





Une suggestion

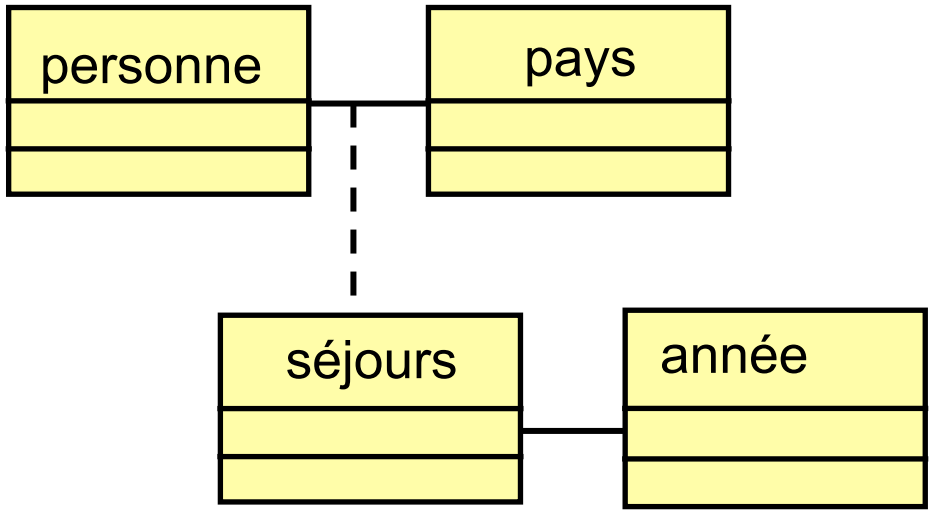
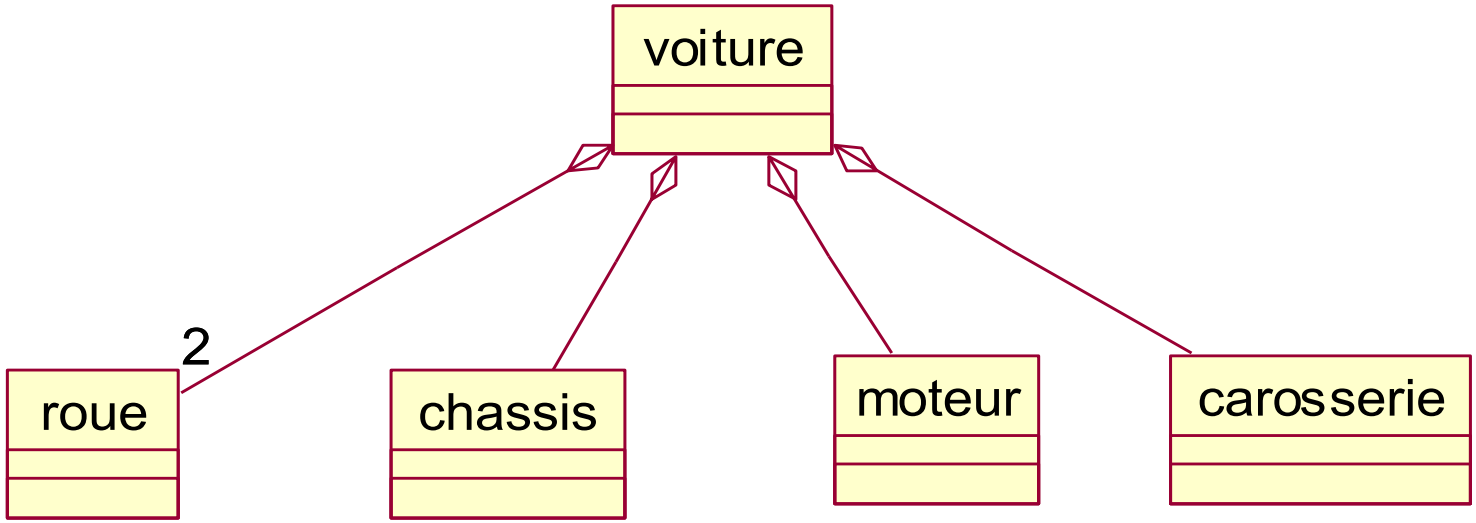


Diagramme de classes

Agrégation

- Type d'association qui spécifie une relation composé-composant
- Association **faible** : le cycle de vie de chaque partie est **indépendant** de celui de l'agrégat



Exemple d'agrégation

- Une relation « tout-partie » avec des propriétés
  - indépendance existentielle
  - « anti-symétrie »
  - transitivité

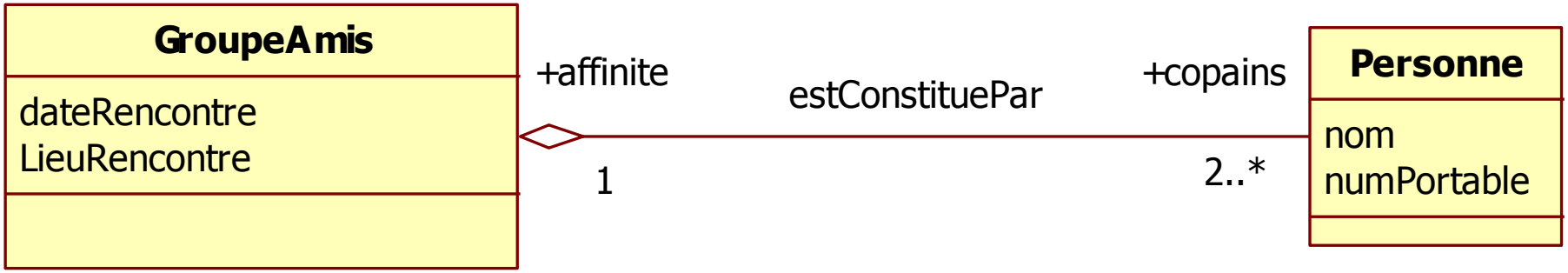
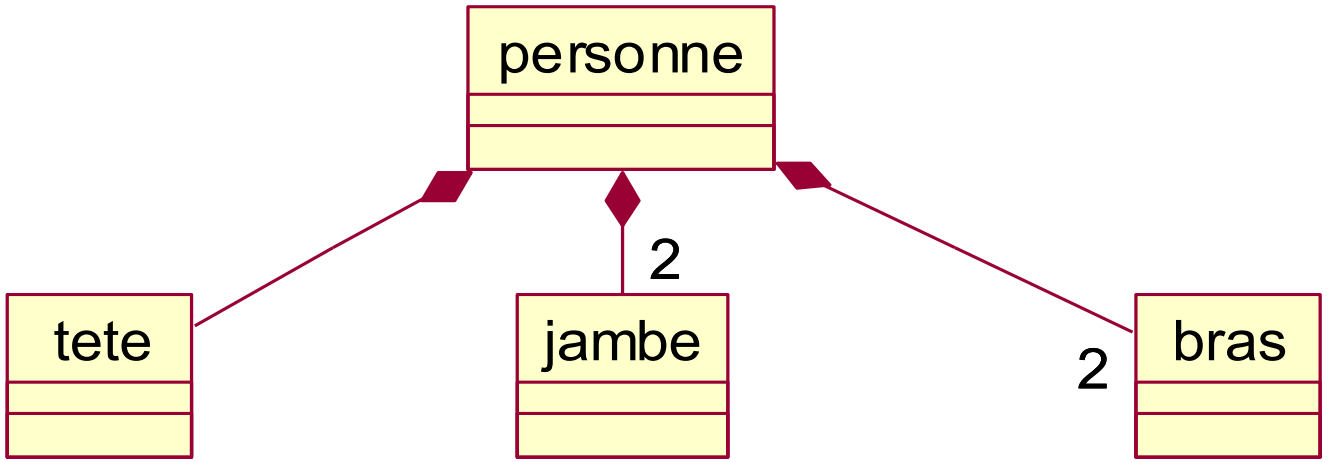


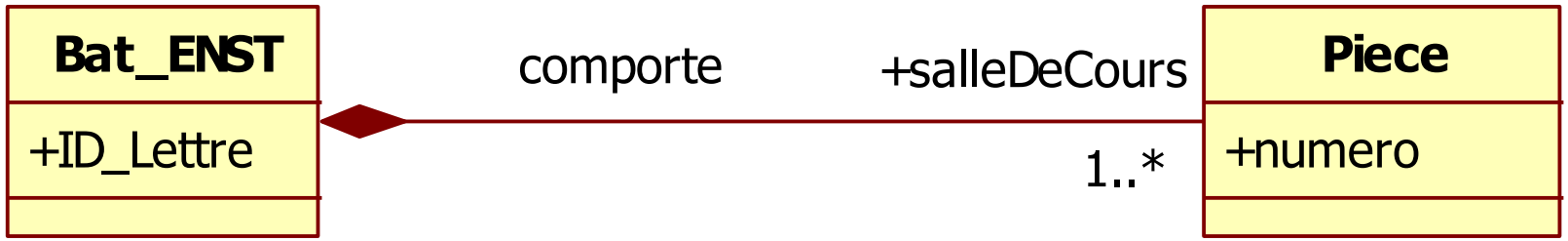
Diagramme de classes
composition

- Type d'association qui spécifie une relation composé-composant
- Association **forte** : le cycle de vie de chaque partie est **dépendant** de celui de l'agrégat



Exemple de composition

- Une relation « tout-partie » avec des propriétés
  - « anti-symétrie »
  - transitivité
    mais
  - dépendance existentielle entre un composant et son agrégat
  - exclusivité



Composition ou agrégation?

- Est-ce que la destruction de l'objet composite (le tout) implique la destruction des objets composants (les parties)?
  - les composants n'ont pas d'autonomie vis-à-vis des composites

  vrai => composition
- Lorsque l'on copie le composite, doit-on copier les composants?

  vrai => composition
  - Ou peut-on les « réutiliser », auquel cas un composant peut faire partie de plusieurs composites?

  vrai => agrégation

Diagrammes de classes à différentes étapes de conception

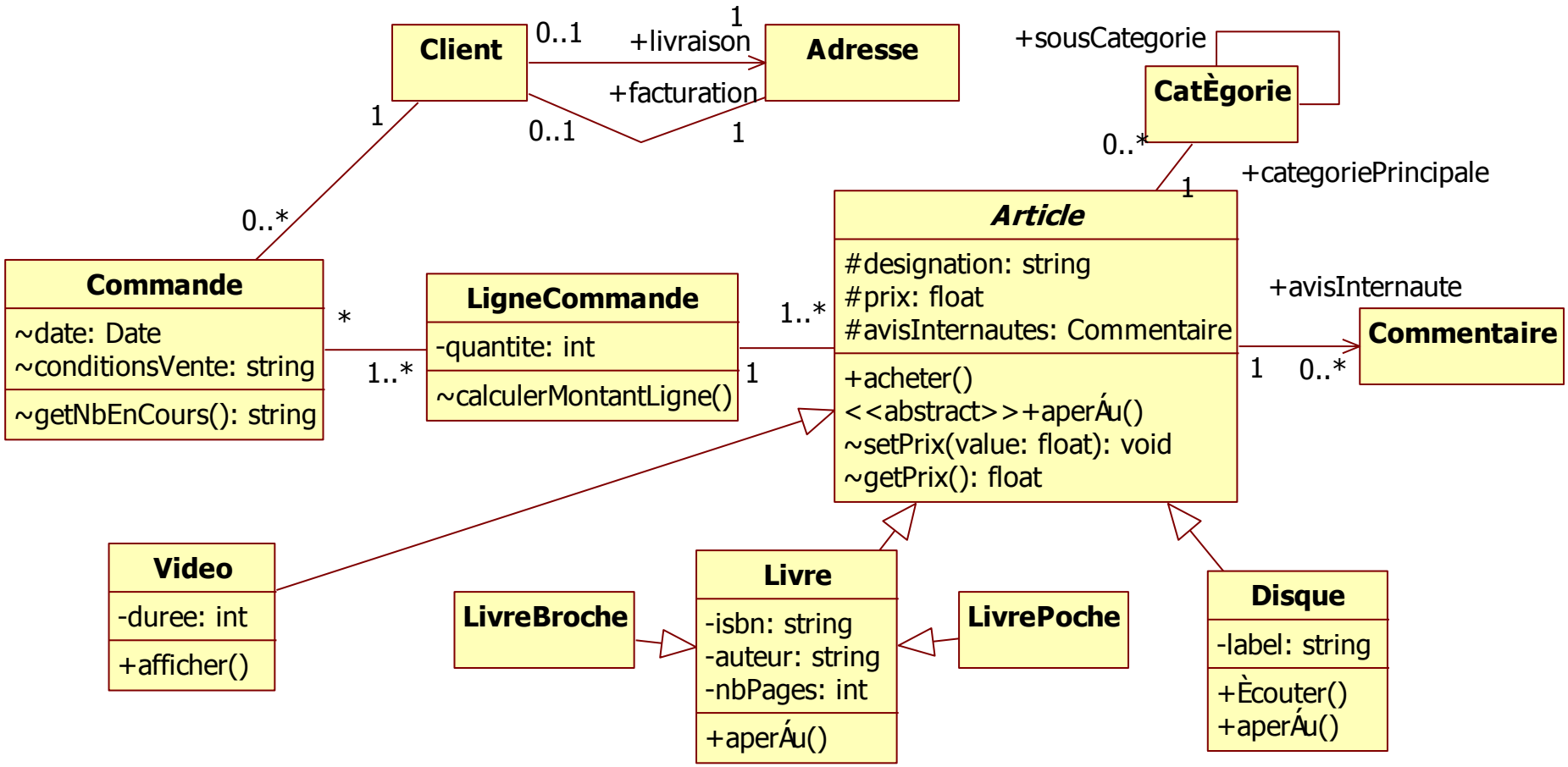
- On peut utiliser les diagrammes de classes pour représenter un système à différents niveaux d'abstraction :
  - Le point de vue spécification met l'accent sur les interfaces des classes plutôt que sur leurs contenus.
  - Le point de vue conceptuel capture les concepts du domaine et les liens qui les lient. Il s'intéresse pas à la manière éventuelle d'implémenter ces concepts et relations et aux langages d'implantation.
  - Le point de vue implantation, le plus courant, détaille le contenu et l'implantation de chaque classe.

- Les diagrammes de classes s'étoffent à mesure qu'on va de hauts niveaux à de bas niveaux d'abstraction (de la spécification vers l'implantation)

Construire un diagramme de classes

1. Trouver les classes du domaine étudié ;
   - Souvent, concepts et substantifs du domaine.
2. Trouver les associations entre classes ;
   - Souvent, verbes mettant en relation plusieurs classes.
3. Trouver les attributs des classes ;
   - Souvent, substantifs correspondant à un niveau de granularité plus fin que les classes. Les adjectifs et les valeurs correspondent à des valeurs d'attributs
4. Organiser et simplifier le modèle en utilisant l'héritage ;
5. Tester les chemins d'accès aux classes ;
6. Itérer et raffiner le modèle.

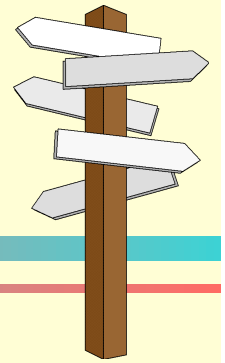
Autre exemple diagramme de classes



Structuration en packages

- Critères de cohérence :
  - Finalité : les classes doivent rendre des services de même nature aux utilisateurs
  - Évolution : isoler les classes stables du métier les distinguer des classes applicatives
  - Cycle de vie des objets
- Minimiser les dépendances

Plan



1. Modélisation : diagrammes de base
   - Diagrammes de cas d'utilisation
   - Diagrammes de classes
   - Diagrammes d'objets
   - Diagrammes de séquences

2. Modélisation avancée
   - Diagrammes d'états
   - Diagrammes d'activités
   - Diagrammes de communication

- UML et méthodologie

  Bonnes pratiques de modélisation
  - Design Patterns

Diagramme d'objets; Diagramme de séquence

TELECOM ParisTech

Objectif des diagrammes d'objets

- Les diagrammes de cas d'utilisation modélisent à QUOI sert le système, en organisant les interactions possibles avec les acteurs.
- Les diagrammes de classes spécifient quelles classes réaliseront les fonctionnalités décrites par les diagrammes de cas d'utilisation.
- Les diagrammes d'objets décrivent COMMENT les éléments du système interagissent entre eux et avec les acteurs.
  - Les objets internes au système interagissent en s'échangent des messages.
  - Les acteurs interagissent avec le système au moyen d'IHM

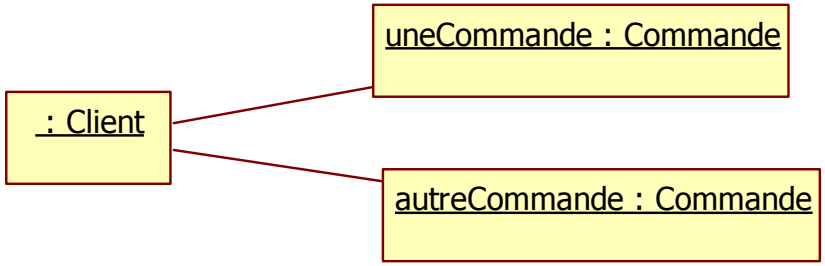
Le diagramme d'objets

- représente les objets d'un système qui coopèrent par envoi de messages à un instant donné.

- Pour :
  - Illustrer le modèle de classes (en montrant un exemple qui explique le modèle) ;
  - Préciser certains aspects du système (des détails) ;
  - Exprimer une exception (en modélisant des cas particuliers. . .).

Le diagramme d'objets

- Le diagramme de classes contraint la structure et les liens entre objets



- Rôle du diagramme d'objets dans une méthode de développement: montrer comment les objets réalisent les fonctionnalités

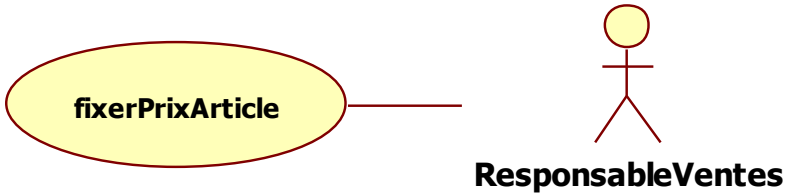
Les diagrammes d'objets d'UML

- constituent les diagrammes d'interaction
- 2 sortes de diagrammes complémentaires
  - Diagrammes de communication
    - Organisation structurelle des objets
  - Diagrammes de séquences
    - Chronologie de l'envoi de messages entre objets
- De plus, on peut se limiter à un contexte précis :
  - Classeur structuré
  - Collaboration

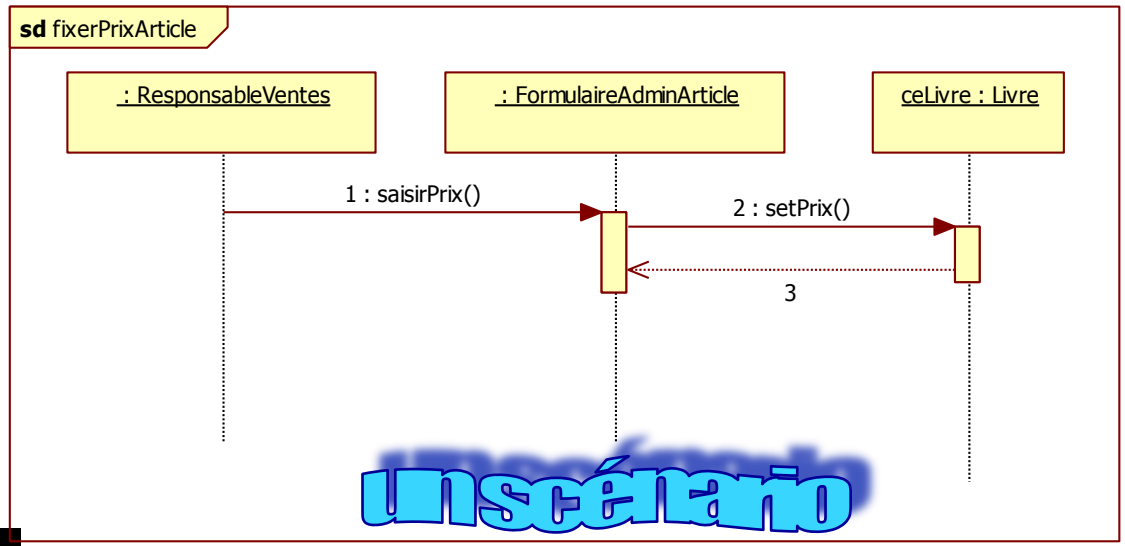
La méthode et l'outil de cette UE utilisent :
le diagramme de séquence
dans le contexte de collaboration

Complémentarité des diagrammes

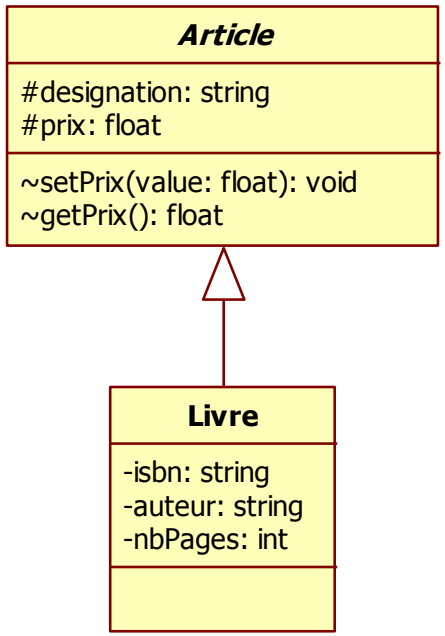
- Diagramme Cas d'utilisation



- Diagramme de séquences

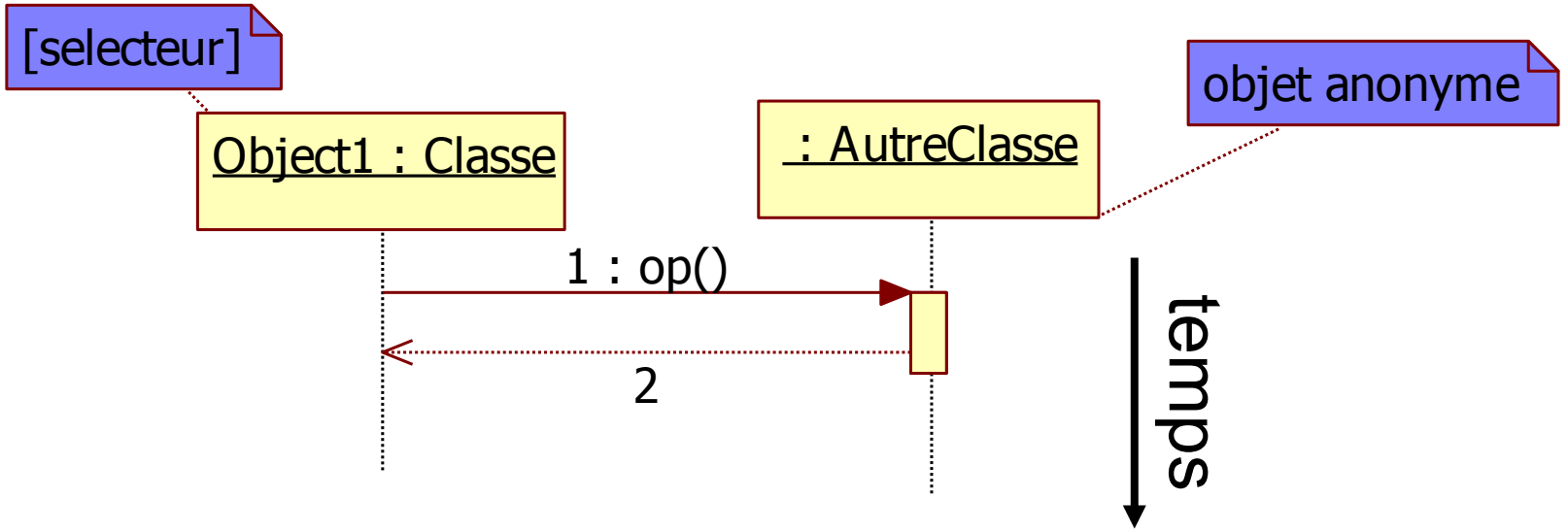


- Diag. de Classes



Ligne de vie

- Une ligne de vie représente un participant à une interaction (objet ou acteur).



- Dans le cas d'une collection de participants, un sélecteur permet de choisir un objet parmi n

nomLigneVie [selecteur]: nomClasseOuActeur

Les messages

- Les principales informations contenues dans un diagramme de séquences sont les messages échangés entre les lignes de vie, présentés dans un ordre chronologique.
- Un message définit une communication particulière entre des lignes de vie (objets ou acteurs).
- Plusieurs types de messages existent
  - l'envoi d'un signal ;
  - l'invocation d'une opération (appel de méthode) ;
  - la création ou la destruction d'un objet.
- La réception des messages provoque une période d'activité (rectangle vertical sur la ligne de vie) marquant le traitement du message

Principaux types de messages

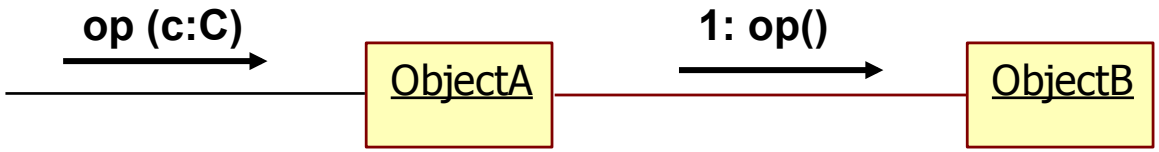
- Un message synchrone bloque l'expéditeur jusqu'à la réponse du destinataire. Le flot de contrôle passe de l'émetteur au récepteur.
  - Typiquement : appel de méthode
  - Si un objet A invoque une méthode d'un objet B, A reste bloqué tant que B n'a pas terminé.

- On *peut* associer aux messages d'appel de méthode un message de retour (en pointillés) marquant la reprise du contrôle par l'objet du message synchrone.

- Un message asynchrone n'est pas bloquant pour l'expéditeur. Le message envoyé peut être pris en compte par le récepteur à tout moment ou ignoré.
  - Typiquement : envoi de signal (voir stéréotype de classe signal)

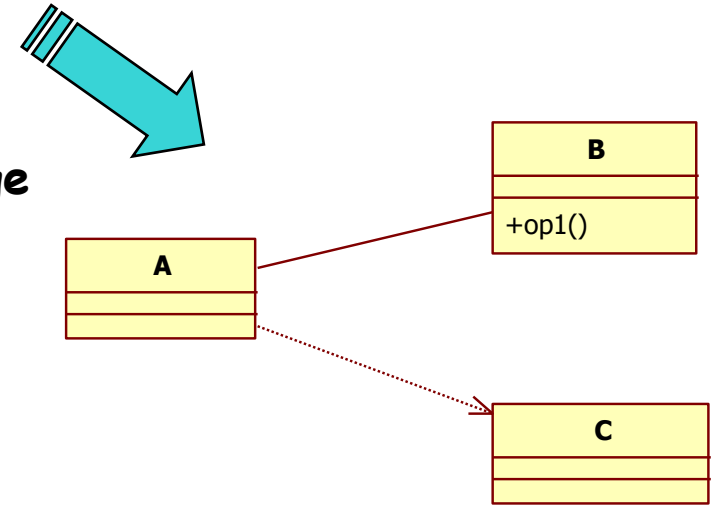
Remarque: Liens durables ou temporaires

- Lien durable = association navigable entre classes
- Lien temporaire = dépendance



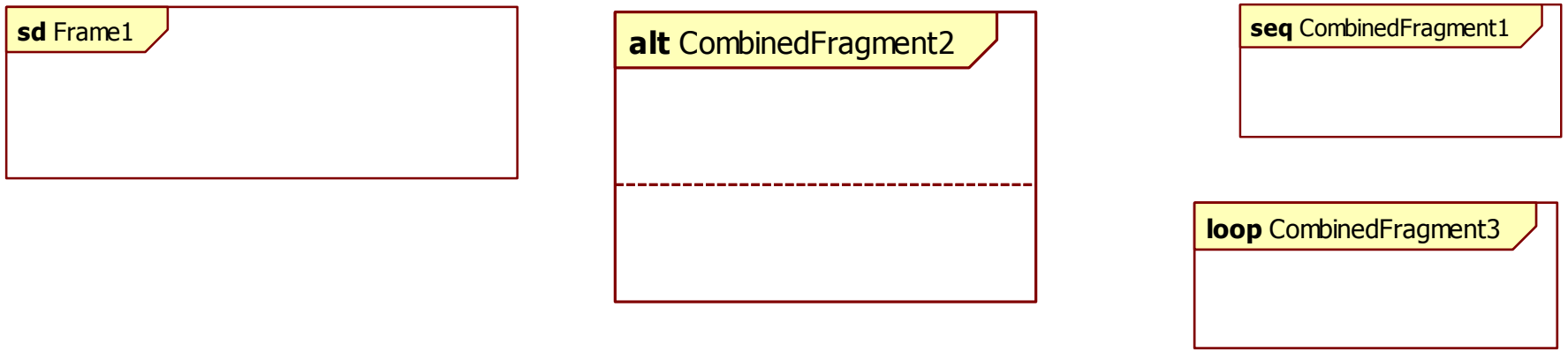
objetA reçoit en paramètre d'un message une référence sur un objet de classe C

-> induit une dépendance



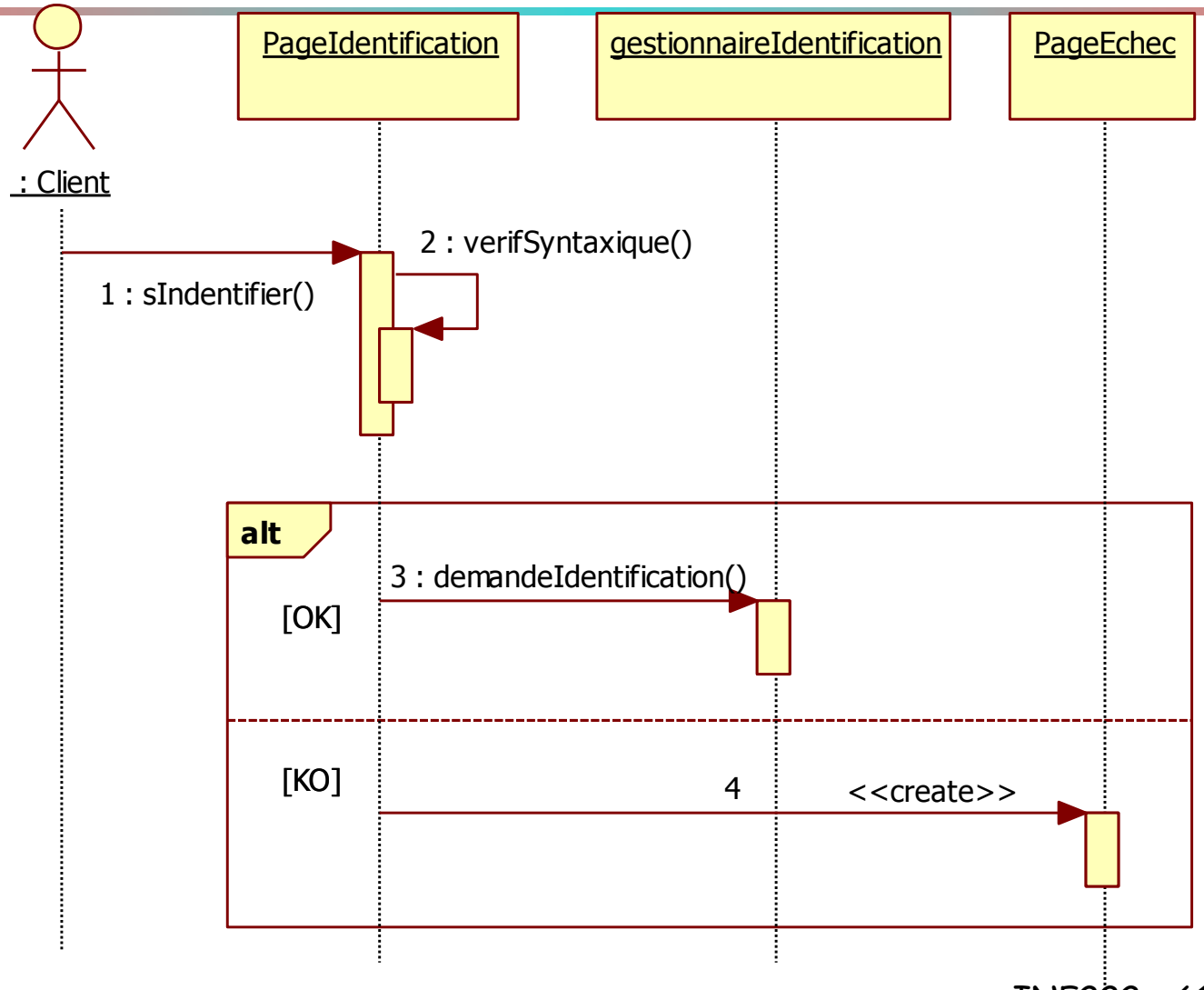
Fragment combiné

- permet de décomposer une interaction complexe en fragments plus simples pour être compris.
- se représente de la même façon qu'une interaction.

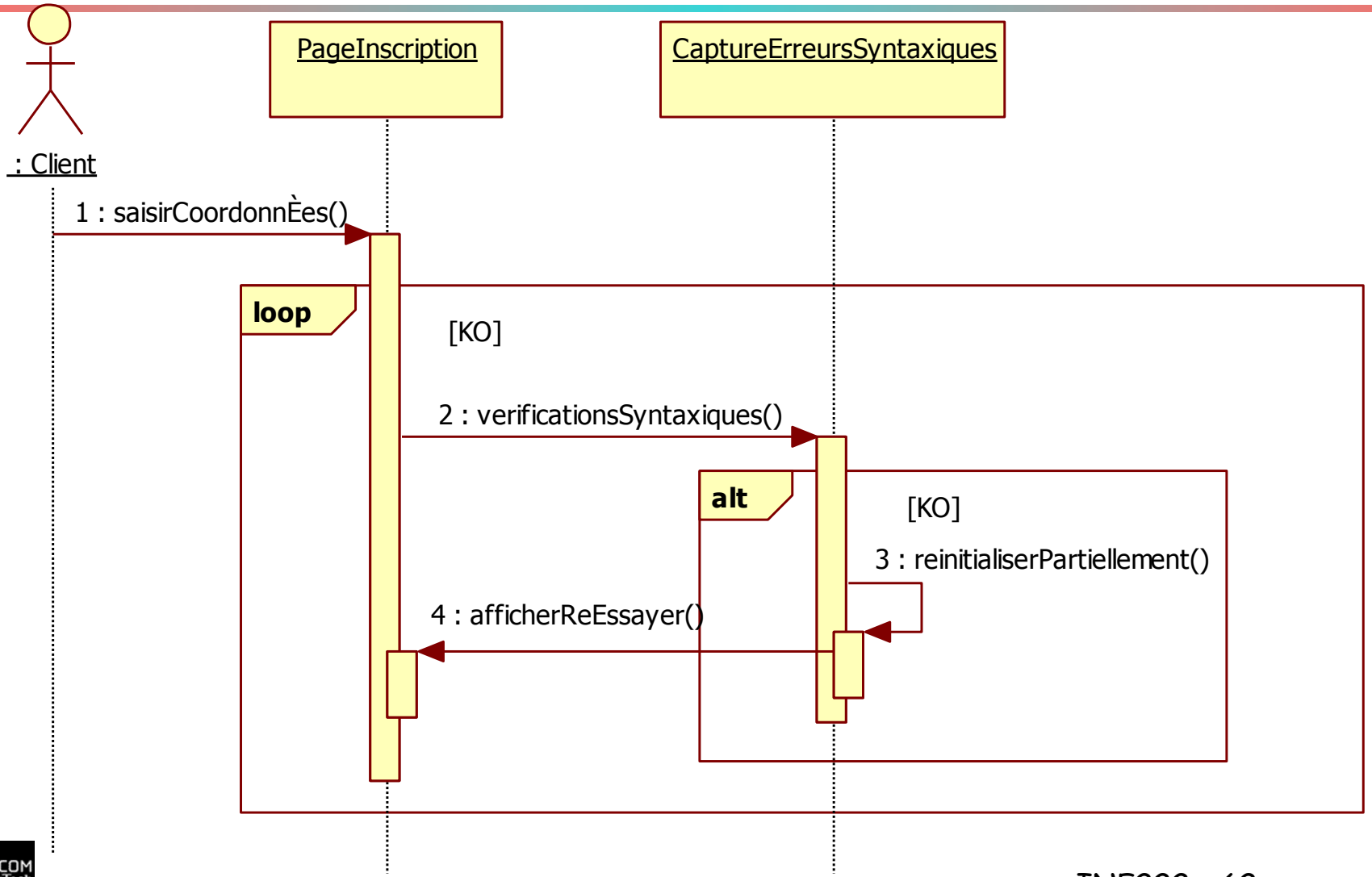


- Il est représenté un rectangle dont le coin supérieur gauche contient un pentagone.
- Dans le pentagone figure le type de la combinaison (appelé opérateur d'interaction).

Exemple de fragment avec l'opérateur conditionnel



Opérateur de boucle



Syntaxe d'une boucle

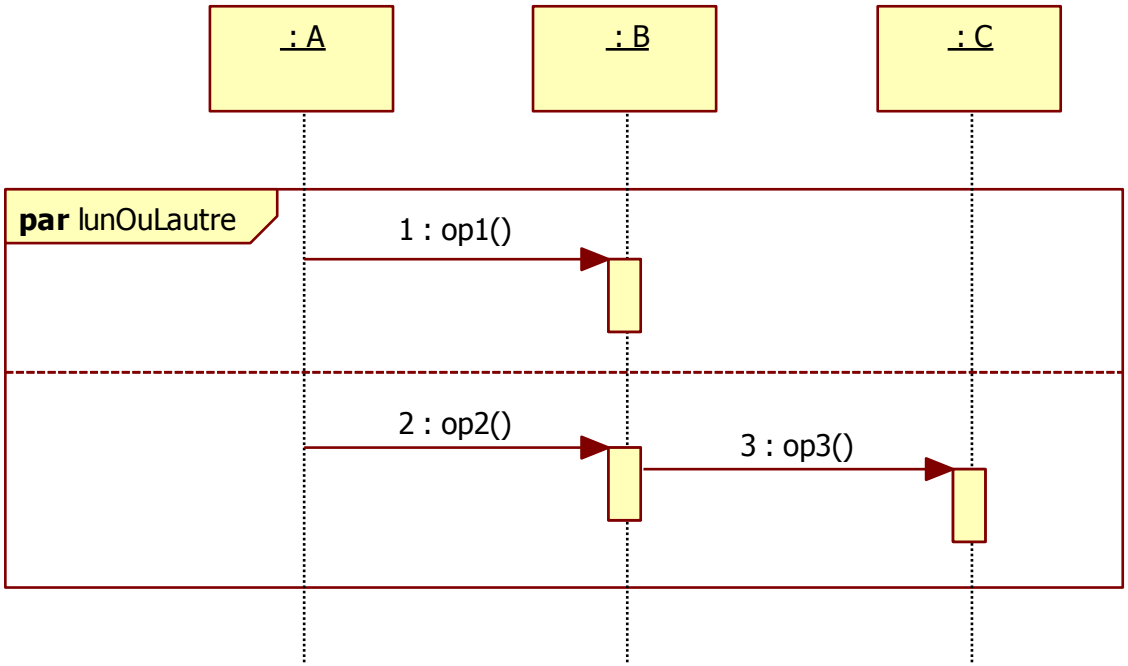
loop (minNbIterations , maxNbIterations)

- La boucle est répétée au moins minNbItérations fois avant qu'une éventuelle condition booléenne ne soit testée
- Tant que la condition est vraie, la boucle continue, au plus maxNbItérations fois.

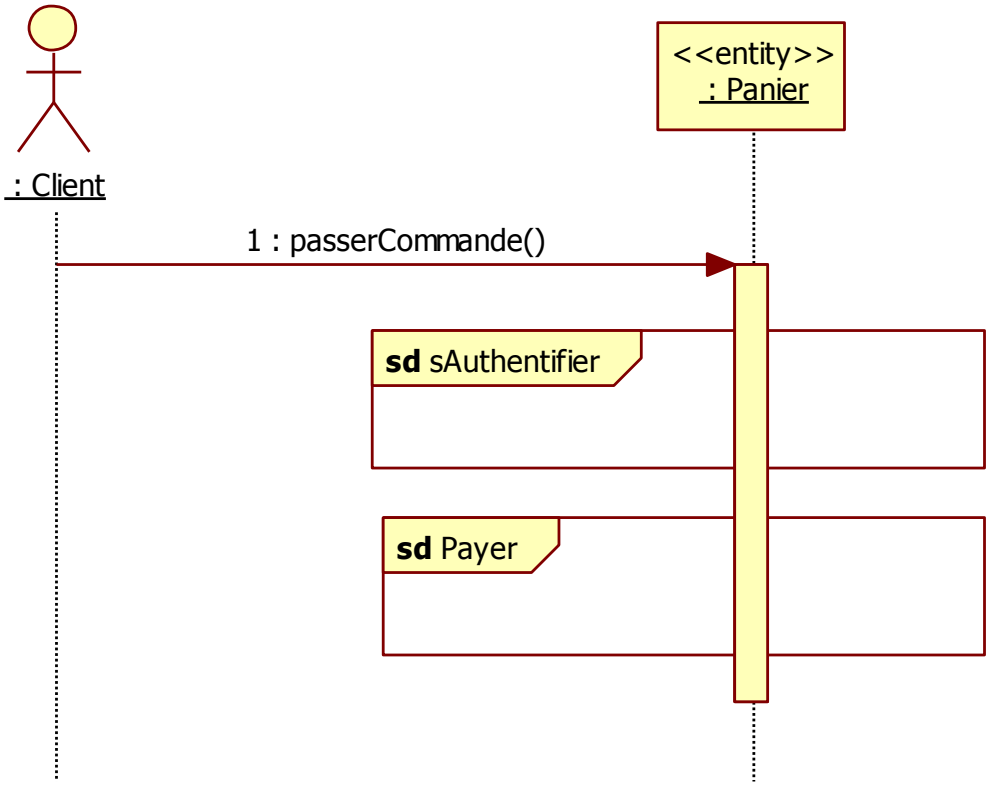
- Notations :
  - la condition est placée entre crochets dans le fragment
  - loop(valeur) est équivalent à loop(valeur,valeur).
  - loop est équivalent à loop(0,*).

L'opérateur parallèle

- L'opérateur **par** permet d'envoyer des messages en parallèle.
  - Ce qui se passe de part et d'autre de la ligne pointillée est indépendant.

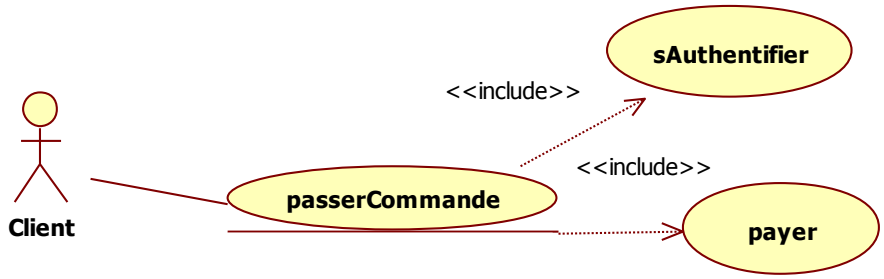


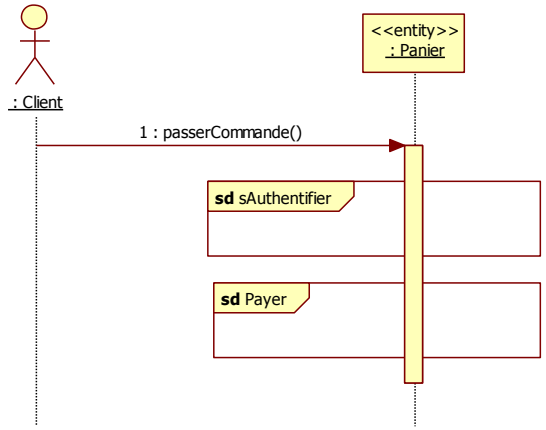
Réutilisation d'une interaction



Utilisation d'un DS pour modéliser un cas d'utilisation

- Chaque cas d'utilisation donne lieu au minimum à un diagramme de séquences (en général plusieurs)

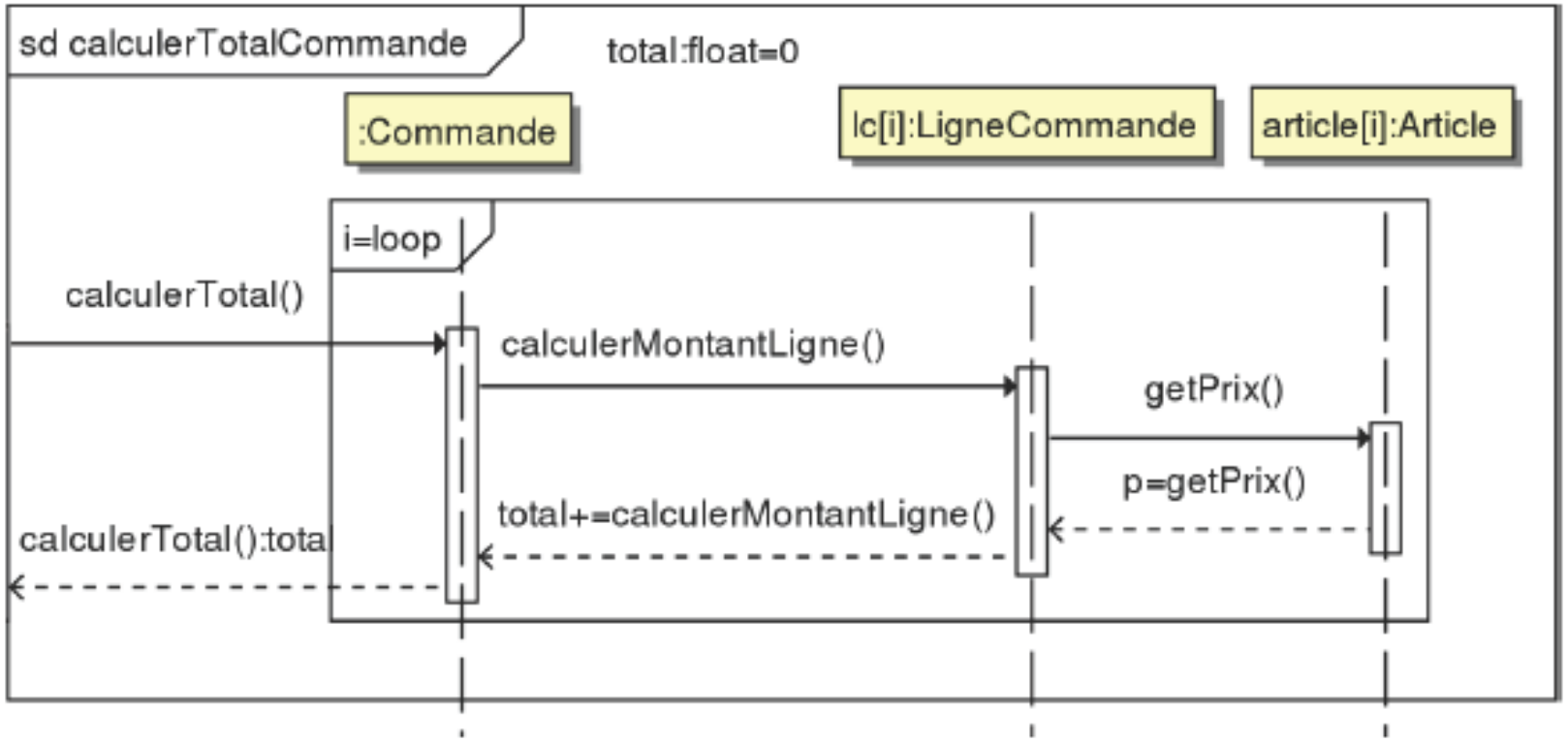




- inclusions et les extensions => réutilisation de fragment par référencement

Utilisation d'un DS pour modéliser une méthode

- Une interaction peut être identifiée par un fragment sd précisant son nom
- Un message peut partir du bord de l'interaction, spécifiant le comportement du système après réception du message, quel que soit l'expéditeur



Exo

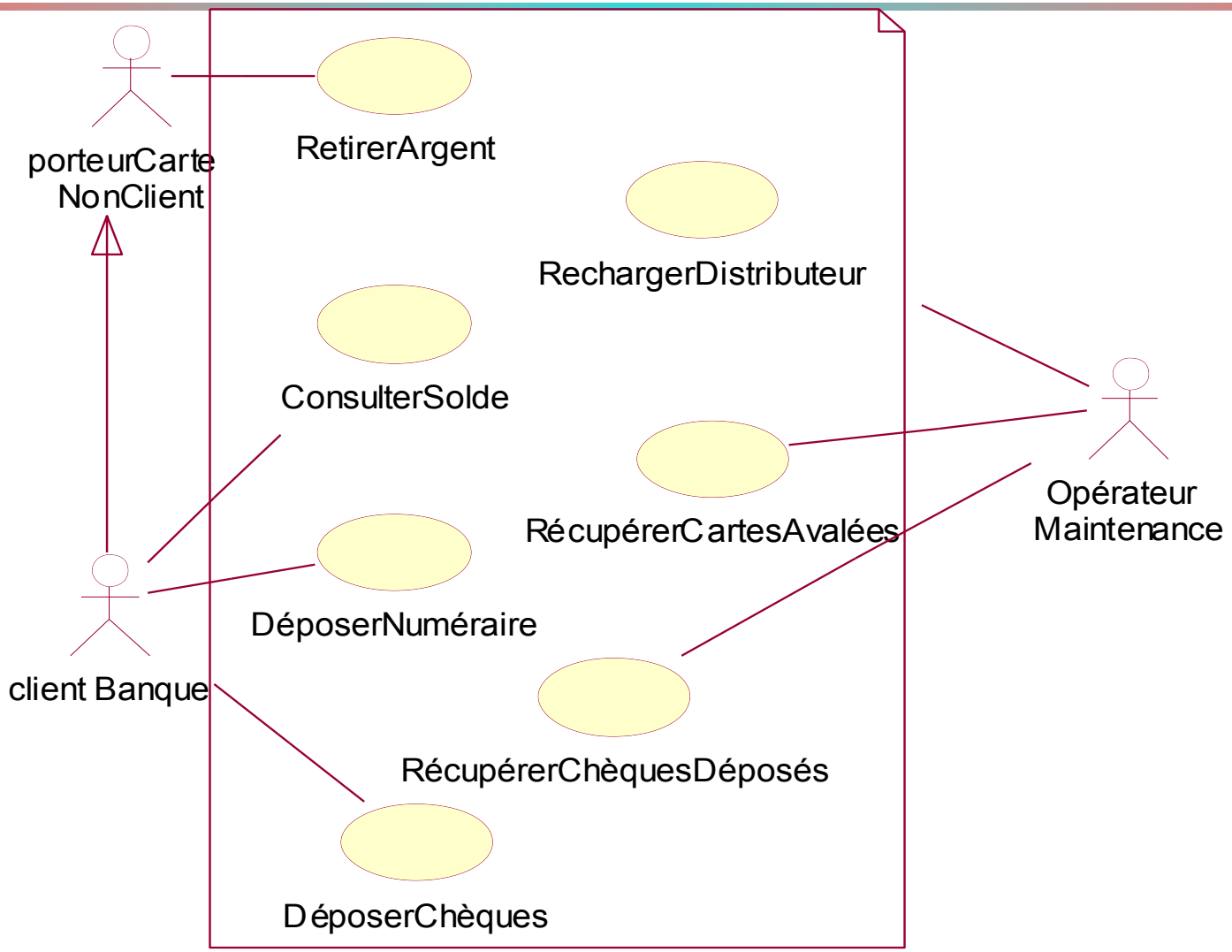
Guichet Automatique de Banque

TELECOM ParisTech

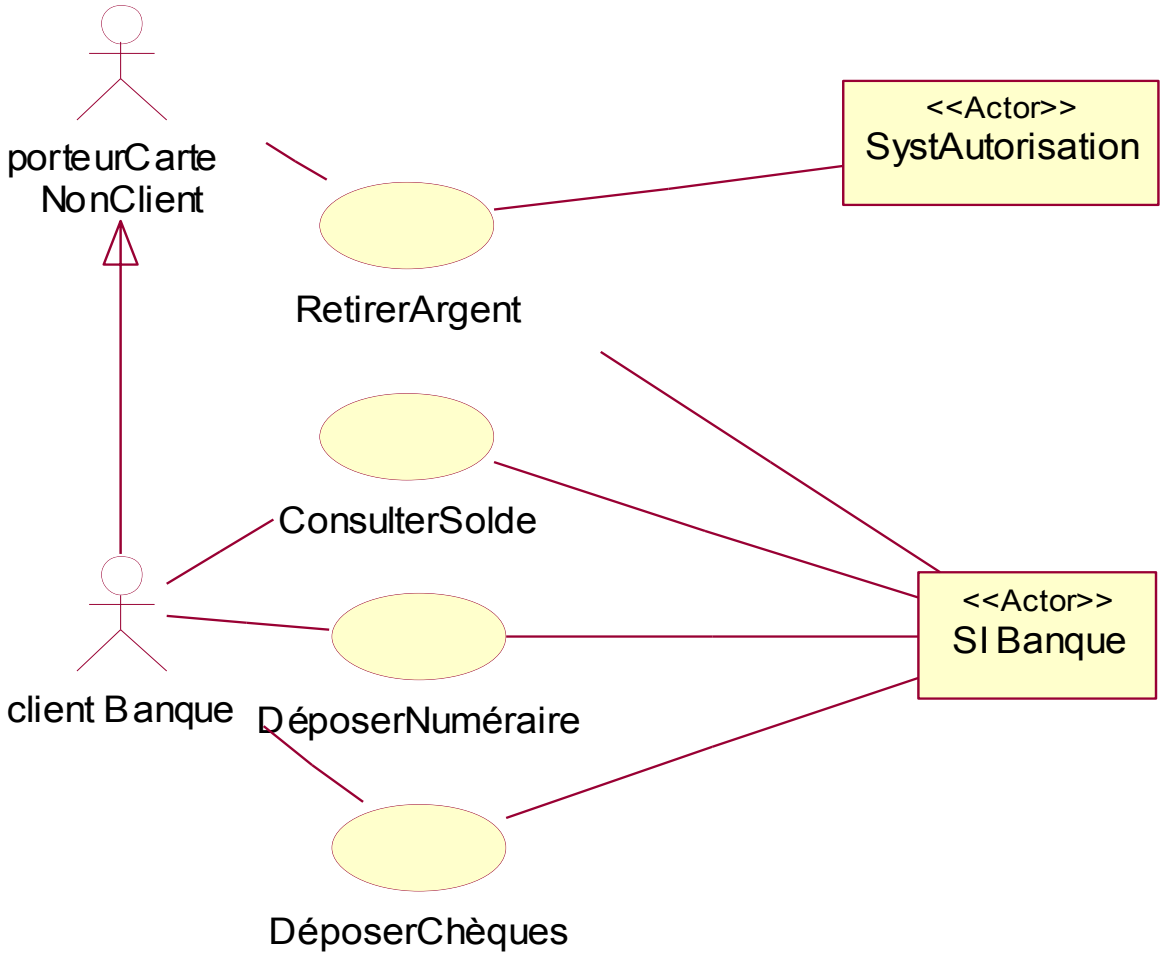
Cahier des charges du Système simplifié de GAB

- Le GAB offre les services suivants
  - Distribution d'argent à tout porteur d'une carte de crédit, via un lecteur de carte et un distributeur de billets
  - Consultation de solde de compte, dépôt en numéraire et dépôt de chèques pour les clients porteurs d'une carte de crédit de la banque propriétaire du GAB
- Autres services
  - Les transactions sont sécurisées
  - Il faut recharger le distributeur

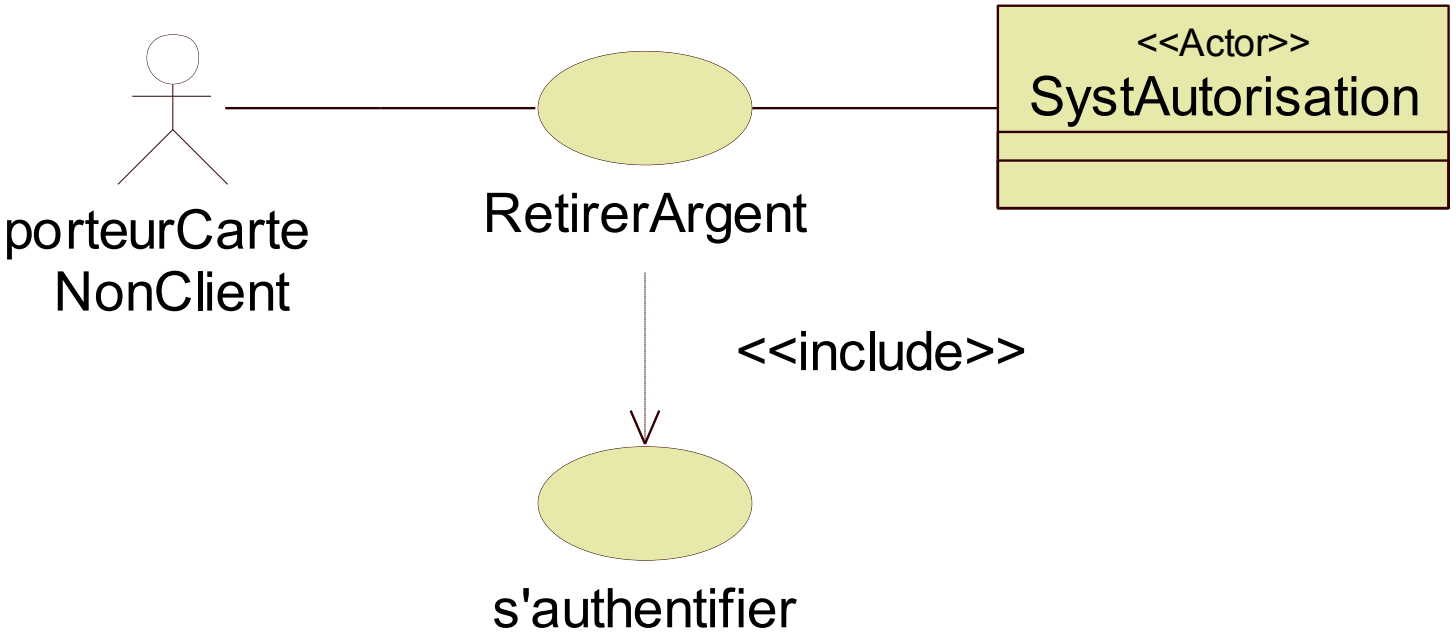
Cas d'utilisation préliminaire du GAB



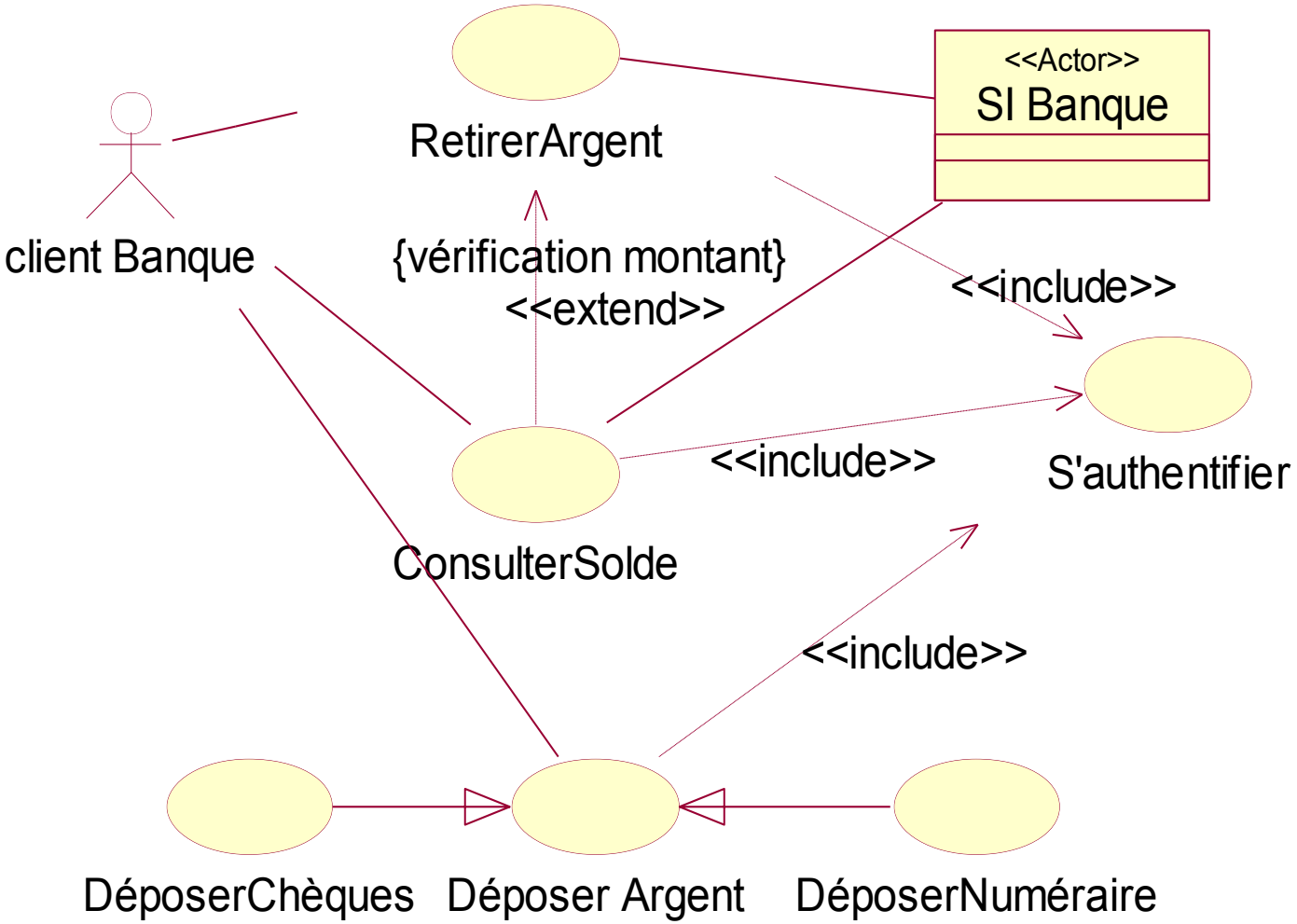
Cas d'utilisation: les acteurs secondaires



Exemple <<include>>



Exemple <<extend>> et généralisation



Description textuelle d'un cas d'utilisation (1/3)

- Titre: retirerArgent
- Résumé: un porteur de carte qui n'est pas client de la banque peut retirer de l'argent si son crédit le permet
- Acteurs: PorteurCarte Nonclient (P), SysAutorisation (S)

- Description des scénarios
  - …
- Exigences non fonctionnelles
  - …
- Besoins en IHM

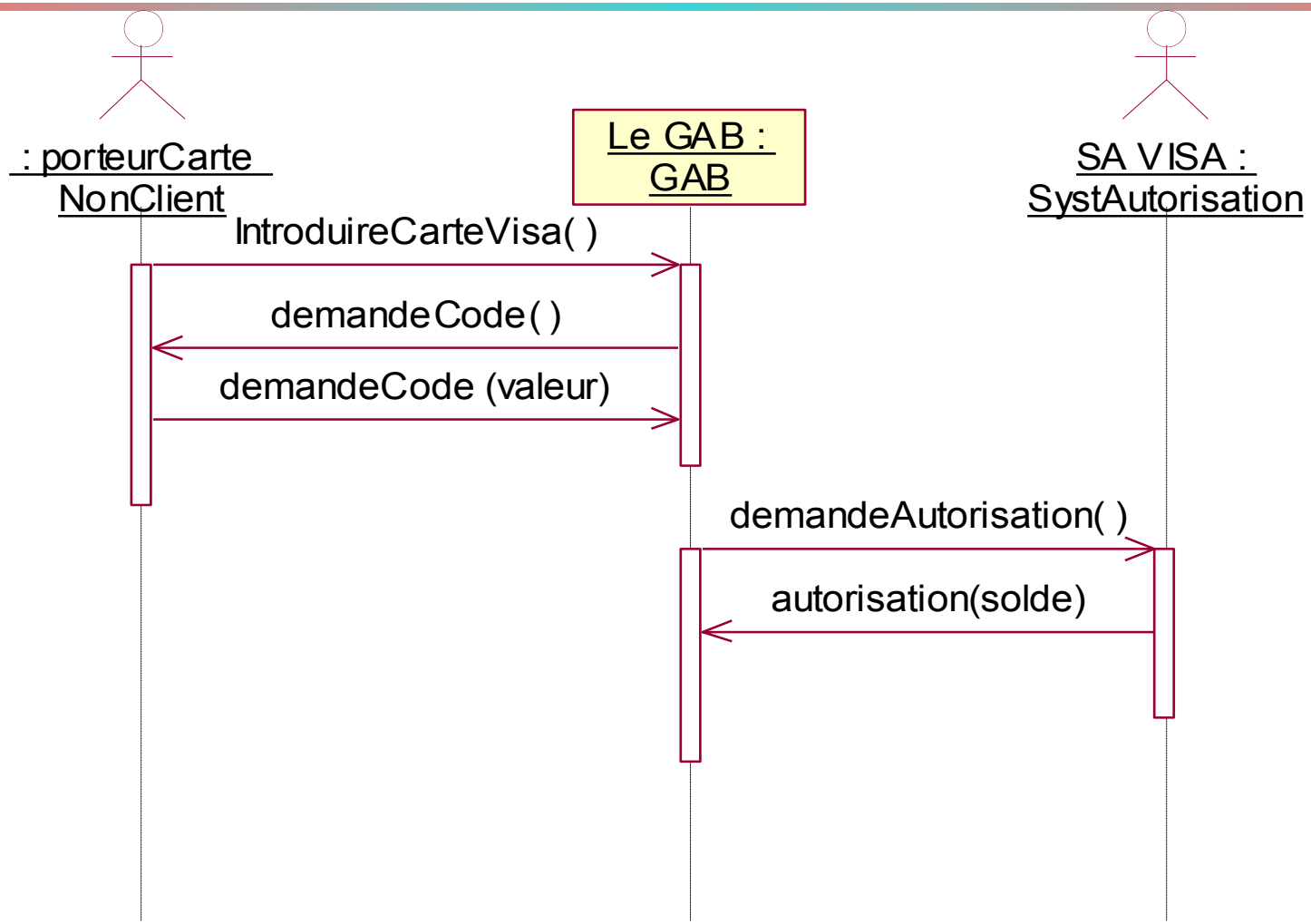
Description textuelle des cas d'utilisation (2/3)

- Description des Scénarios
  - Préconditions
    - La caisse GAB est alimentée (au moins quelques billets)
    - Aucune carte ne se trouve déjà dans le lecteur
  - Scénario nominal
    - …
  - Enchaînements alternatifs (ou scénario non nominal)
    - …
  - Enchaînements d'erreur
    - …
  - Postconditions
    - …

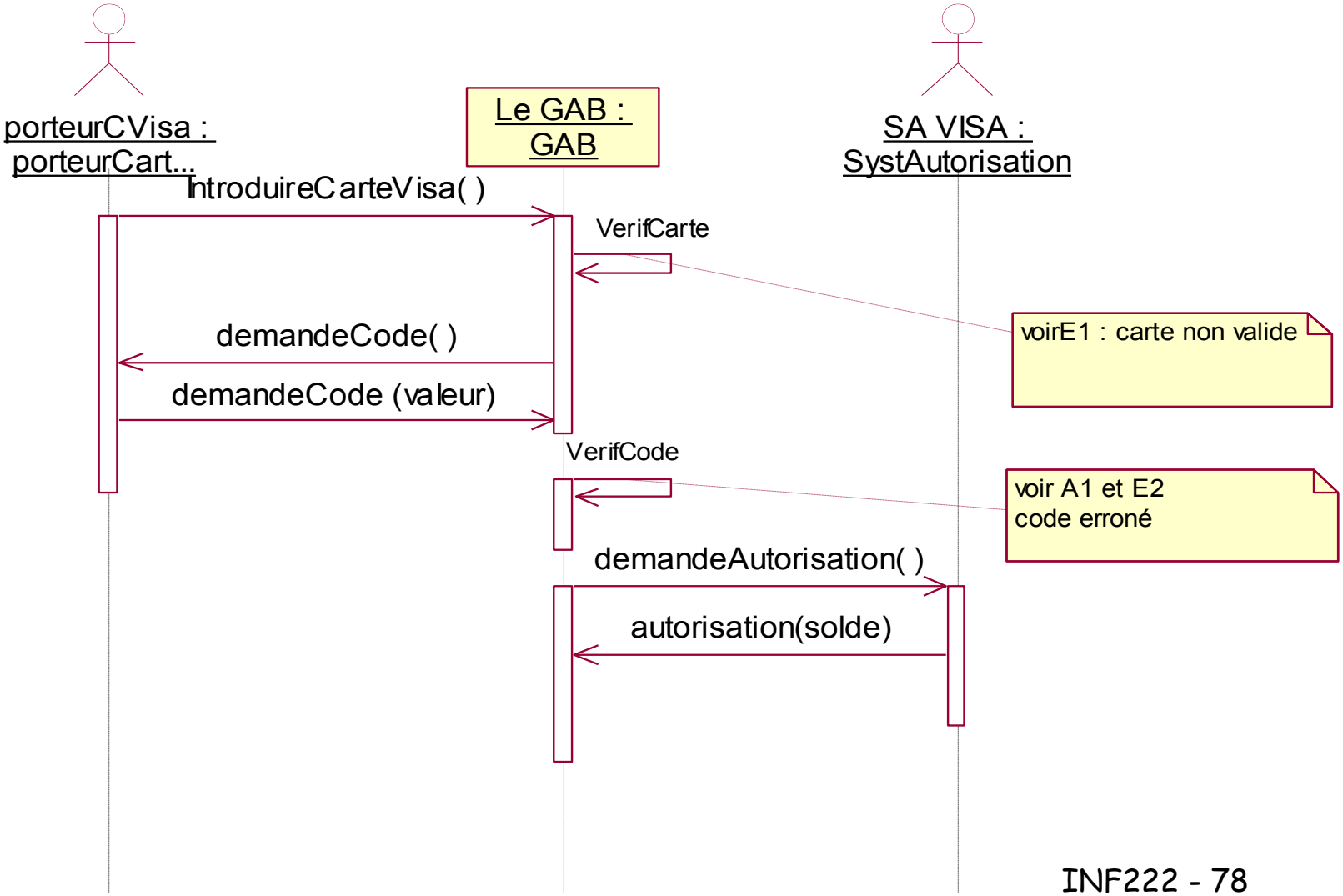
Description textuelle des cas d'utilisation (3/3)

- Scénario nominal
  1. Le porteur de carte introduit sa carte dans le lecteur
  2. Le GAB vérifie que c'est une carte bancaire
  3. Le GAB demande le code d'identification
  4. Le porteur de carte saisit son code
  5. Le GAB compare ce code avec celui codé sur la puce
  6. Le GAB demande une autorisation au SysAutorisation
  7. Le sysAutorisation donne accord et montant hebdomadaire
  8. ...
- Enchaînements Alternatifs
  - A1 : code provisoirement erroné => point 5 du SN
  - A2 : montant demandé > solde hebdomadaire => point 10 du SN
  - ...
- Enchaînements d'erreur
  - E1: carte non-valide (illisible, périmée)
  - E2: code définitivement erroné
  - E3: retrait non autorisé
  - E4 : carte non reprise
- Postconditons
  - Billets manquants = somme retirée
  - Transaction enregistrée par le GAB

Diagramme de séquences
(début du scénario nominal pour un porteur de carte VISA)



(Début du scénario enrichi)



Le diagramme d'activités

un autre diagramme pas Orienté Objet …

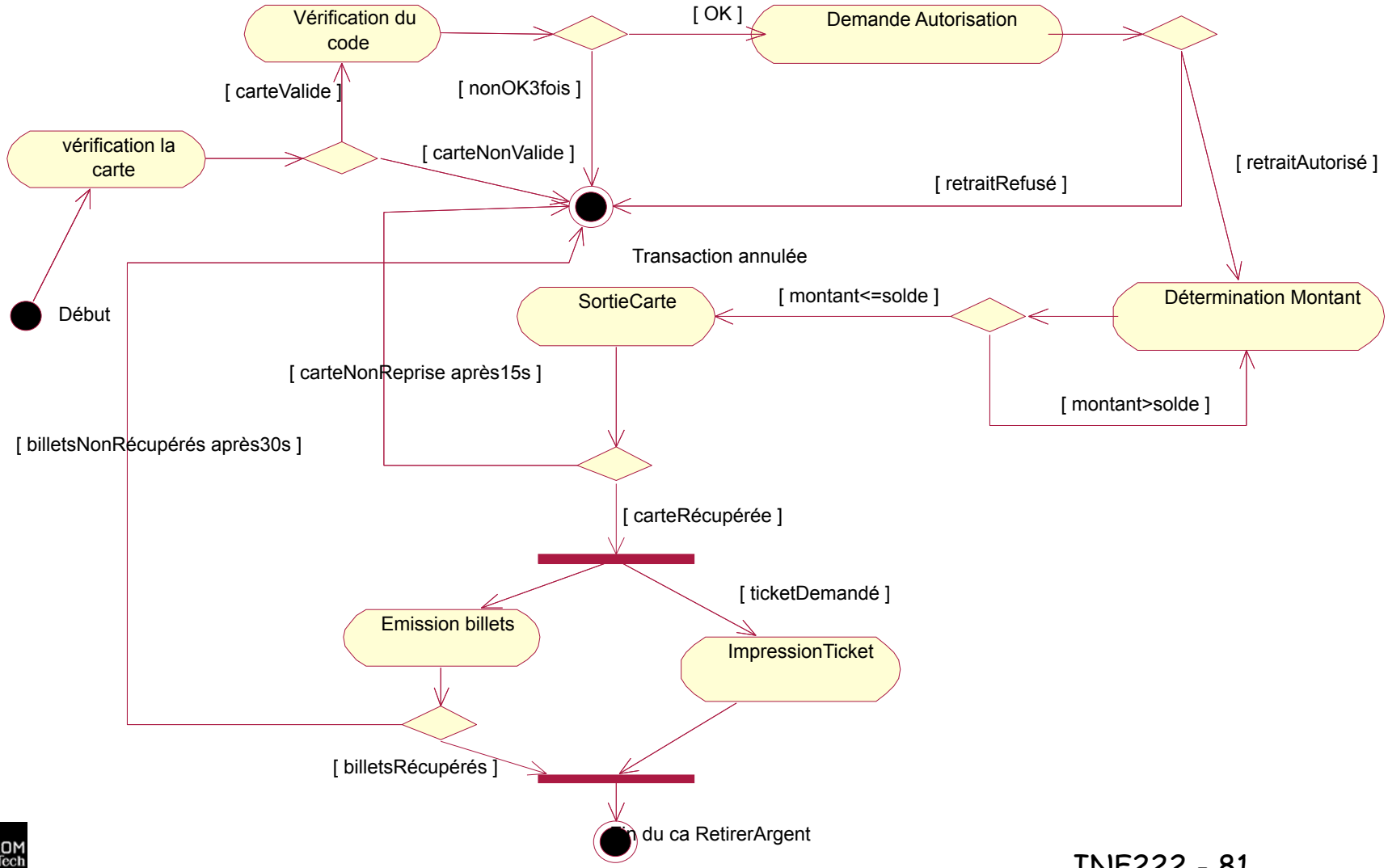
TELECOM ParisTech

But du diagramme d'activités

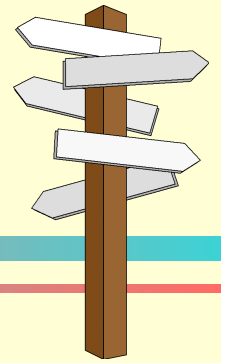
- Diagramme d'activité est utilisé pour:
  - Modéliser un workflow dans un cas d'utilisation ou entre plusieurs cas d'utilisation.
  - Spécifier une opération (décrire sa logique)

- Eventuellement et temporairement, pendant le cycle de développement,
  - approprié pour modéliser la dynamique d'une tâche, d'un cas d'utilisation lorsque le diagramme de classe n'est pas encore stabilisé.
  - dans un but d'explication entre dévloppeurs du modèle

Diagramme d'activités
(retirerArgent)



Plan



1. Modélisation : diagrammes de base
   - Diagrammes de cas d'utilisation
   - Diagrammes de classes
   - Diagrammes d'objets
   - Diagrammes de séquences

2. Modélisation avancée
   - Diagrammes d'états
   - Diagrammes d'activités
   - Diagrammes de communication

Modélisation comportementale

Diagramme d'états

voir cours de Elie Najm

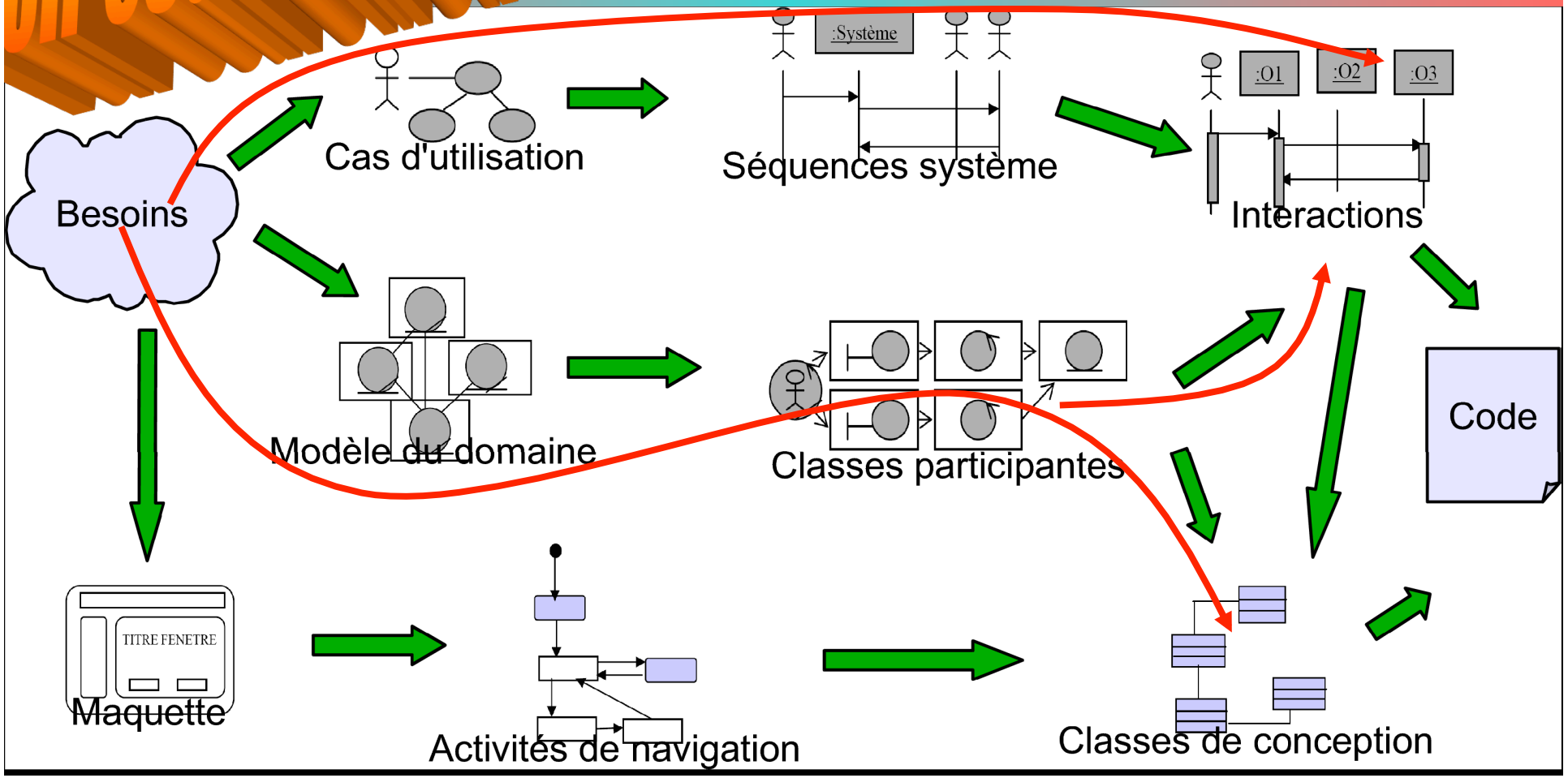
TELECOM ParisTech

Enchaînement de diagrammes

Cohérence du modèle UML

TELECOM ParisTech

Enchaînement et itérations entre diagrammes UML



+ diagramme d'états de classes principales

Exo diagramme de classes

TELECOM ParisTech

Exo

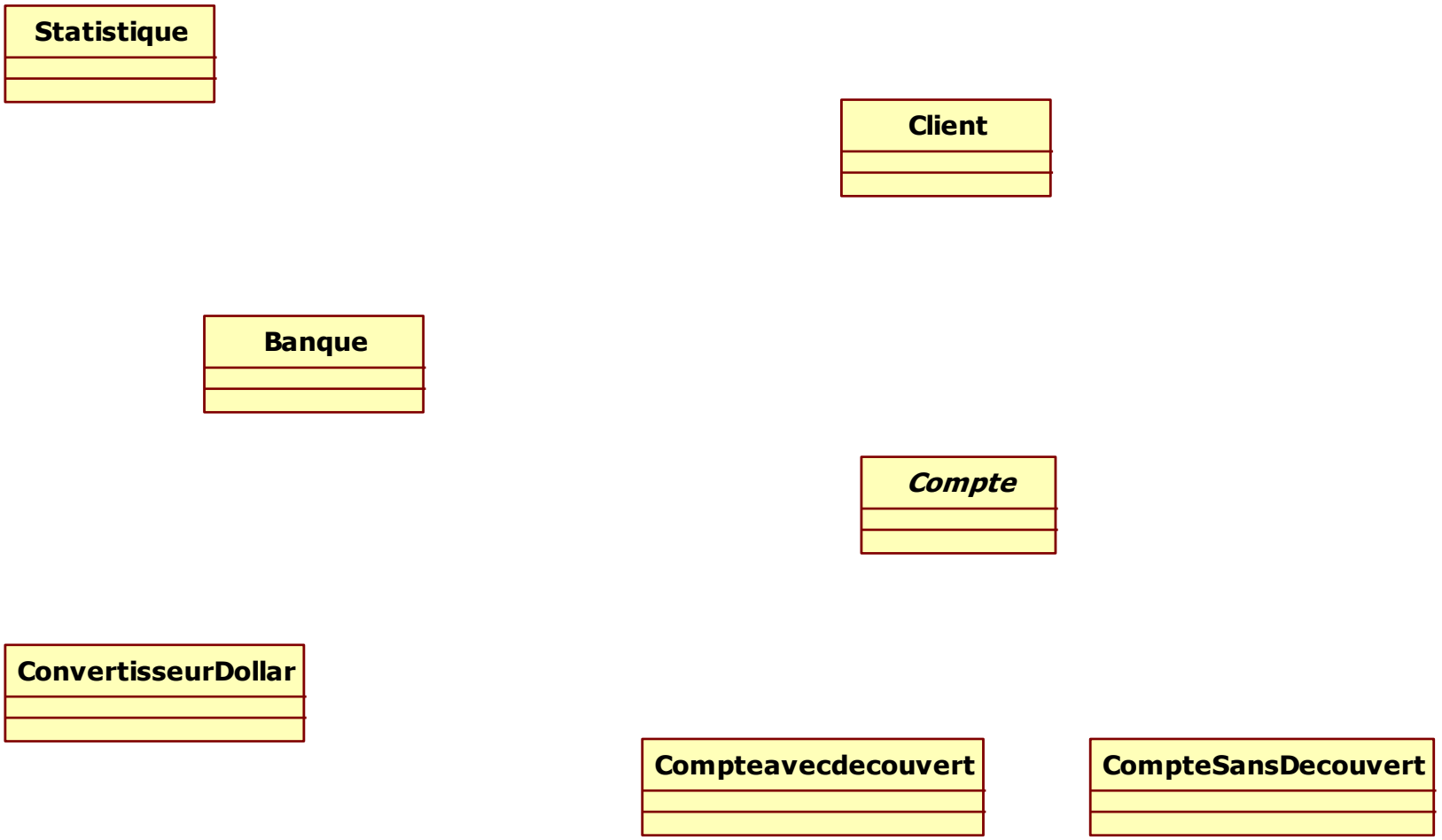
Il s'agit de décrire les classes et les associations entre classes d'une application simplifiée de gestion de comptes bancaires, à partir du texte suivant :

- Chaque client peut posséder plusieurs comptes en banque. Un client est défini par un numéro, un nom et une adresse
- Un compte est défini par un numéro et le solde. Deux types de comptes existent, l'un autorise un découvert, l'autre pas.
- L'application offre la possibilité d'établir des statistiques concernant les mouvements des comptes de clients. Pour cela on développera une classe Statistique.
- L'application contient aussi fournit aussi une classe *ConvertisseurDollar* qui permet de convertir les montants des transactions de l'Euro vers le Dollar et inversement.

Démarche

- dans le texte repérer les termes du métier bancaire qui sont les classes et les relier avec des associations nommées à partir d'expressions verbales
- au fur et à mesure compléter les attributs et opérations
- vérifier la nature et la navigabilité de l'association : binaire, généralisation/spécialisation, agrégation (association faible entre composant et composés)
- mettre les cardinalités aux extrémités des associations et donnez des noms de rôle à ces extrémités quand cela vous semble avoir du sens.

À compléter


Statistique
Client
Banque
Compte
ConvertisseurDollar
Compteavecdecouvert
CompteSansDecouvert

Exo2 association unidirectionnelle 1-1

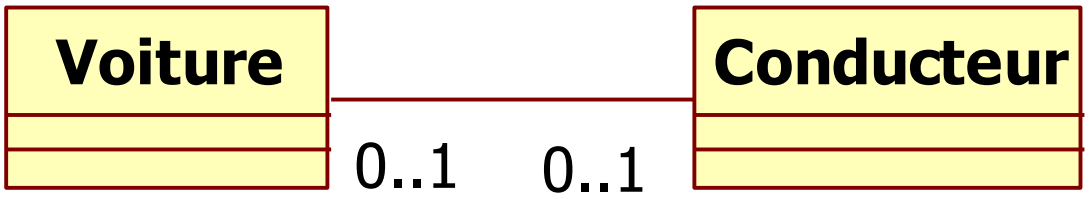
- Comment transformer cette relation en code Java?

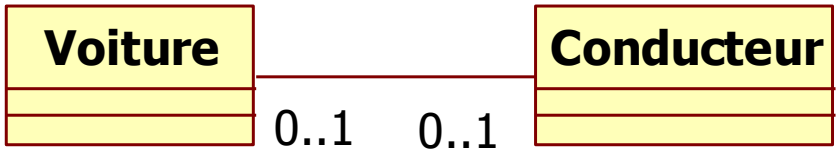

```
public class Emplacement {...}

public class Exemplaire {
        private Emplacement rangee ;
        public void setEmplacement ( Emplacement emplacement ){
                this . rangee = emplacement ;
        }
        public Emplacement getEmplacement (){
                return rangee ;
        }
public static void main ( String [] argv ){
        Emplacement emplacement = new Emplacement ();
        Exemplaire exemplaire = new Exemplaire ();
        exemplaire . setEmplacement ( emplacement );
        Emplacement place = exemplaire . getEmplacement ();
        }
```

Exo3 :association bidirectionnelle 1-1

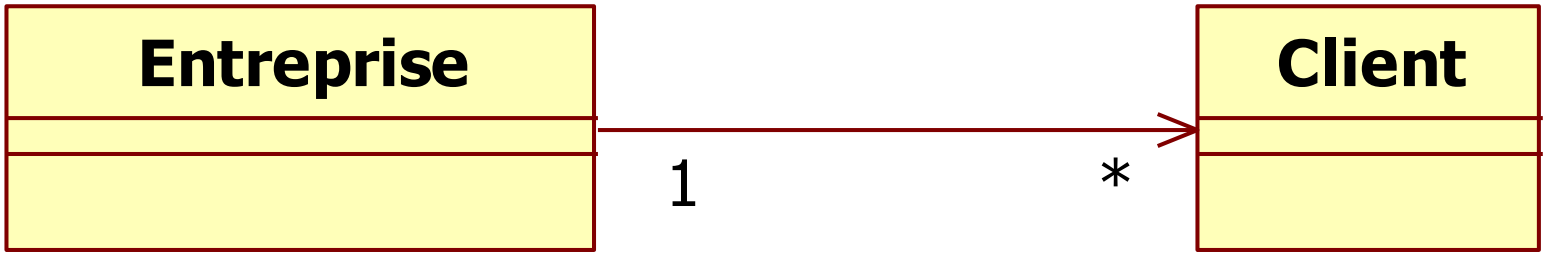
- Transformer en Java …

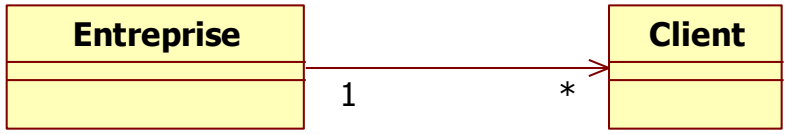





```
public class Conducteur {
        Voiture voiture ;
        public void addVoiture ( Voiture voiture ){
                if( voiture != null ){
                this . voiture = voiture ;
                voiture . conducteur = this ;
                }
        }
        public static void main ( String [] argv ){
                Voiture voiture = new Voiture ();
                Conducteur conducteur = new Conducteur ();
                conducteur . addVoiture ( voiture );
                }
}
public class Voiture {
        Conducteur conducteur ;
        public void addConducteur ( Conducteur conducteur ){
        this . conducteur = conducteur ;
        conducteur . voiture = this ;
        }
}
```

Exo4 Association unidirectionnelle 1-*






```
public class Client {...}

public class Entreprise {
    private Vector clients = new Vector ();
    public void addClient ( Client client ){
        clients . addElement ( client );
    }
    public void removeClient ( Client client ){
        clients . removeElement ( client );
    }
    public static void main ( String [] argv ){
        Entreprise monEntreprise = new Entreprise ();
        Client client = new Client ();
        monEntreprise . addClient ( client );
        monEntreprise . removeClient ( client );
    }
}
```

De l'analyse d'un problème à sa généralisation

Exemple d'un Patron de Conception

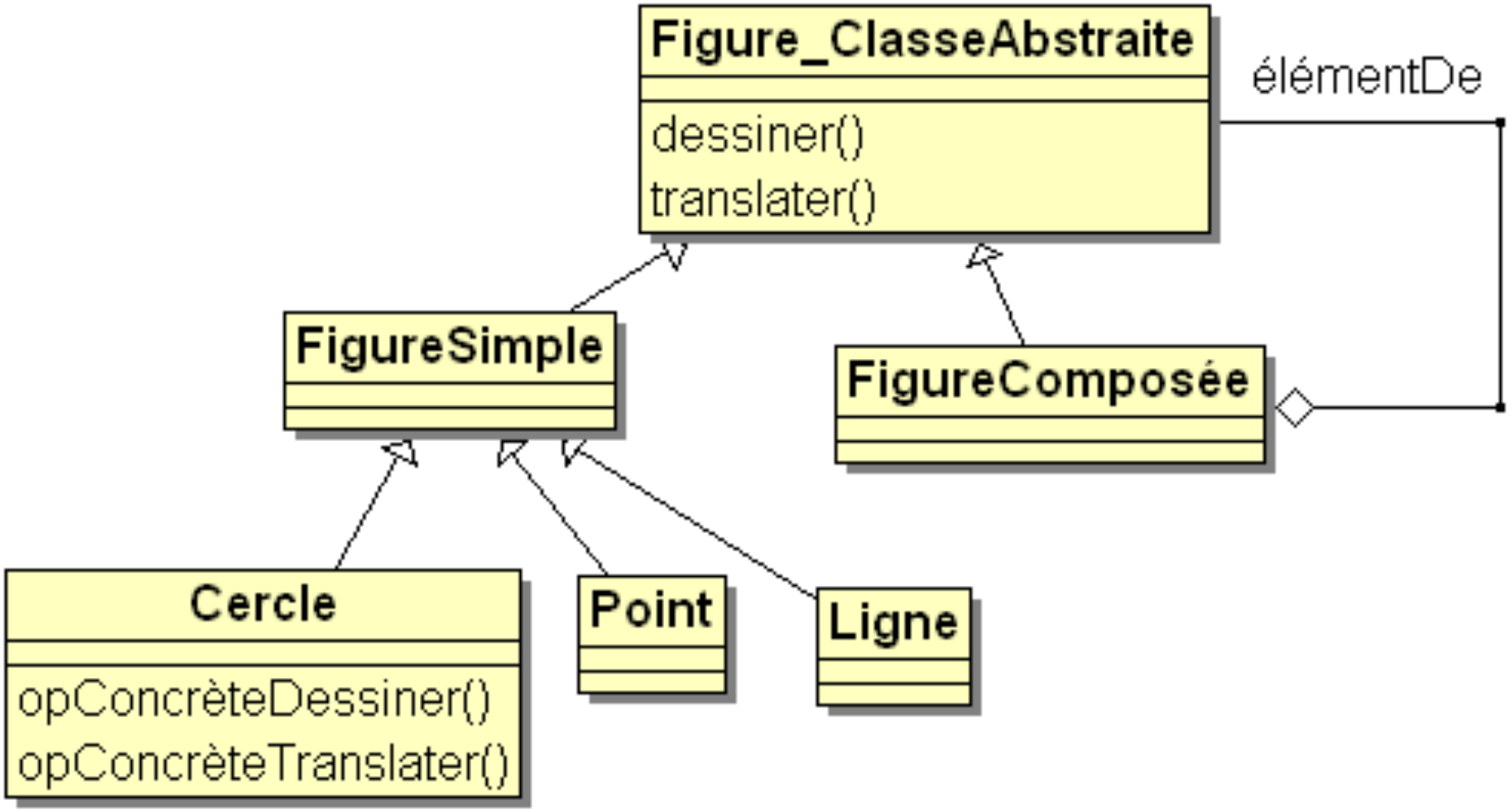
Sur un petit exemple

TELECOM ParisTech

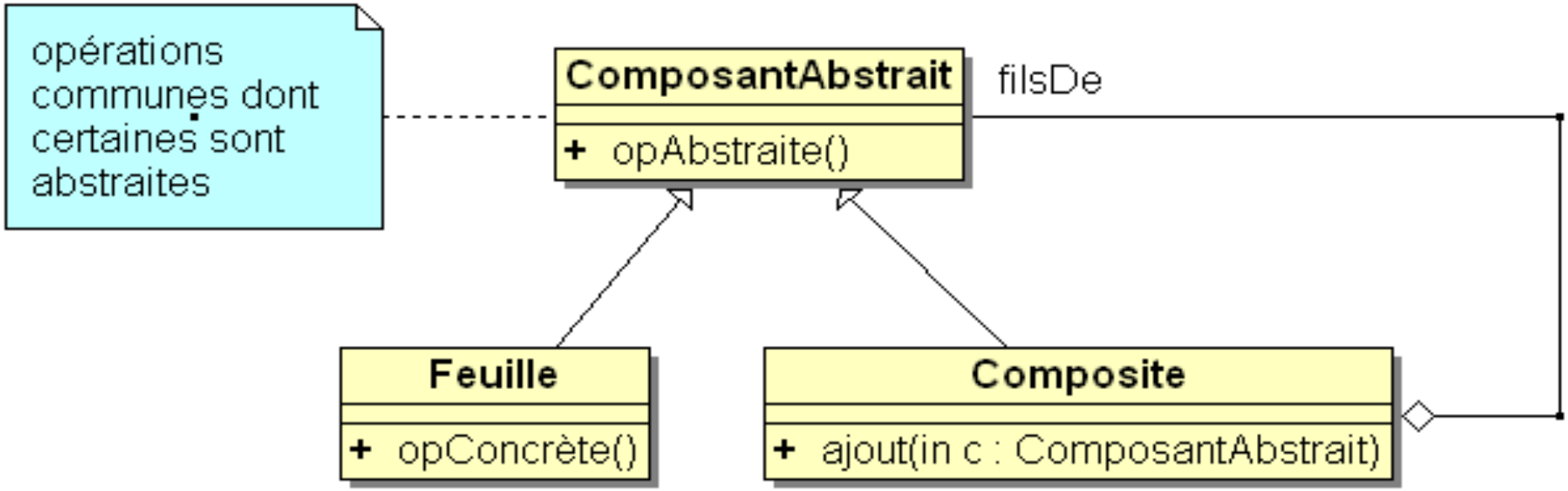
Ex Représentation de figures géométriques

- Une figure géométrique est composée de figures simples ou composées
- Une figure simple peut être un point, une ligne ou un cercle
- Une figure composée est constituée de plusieurs figures
- Une figure peut être dessinée ou translatée

De la modélisation …

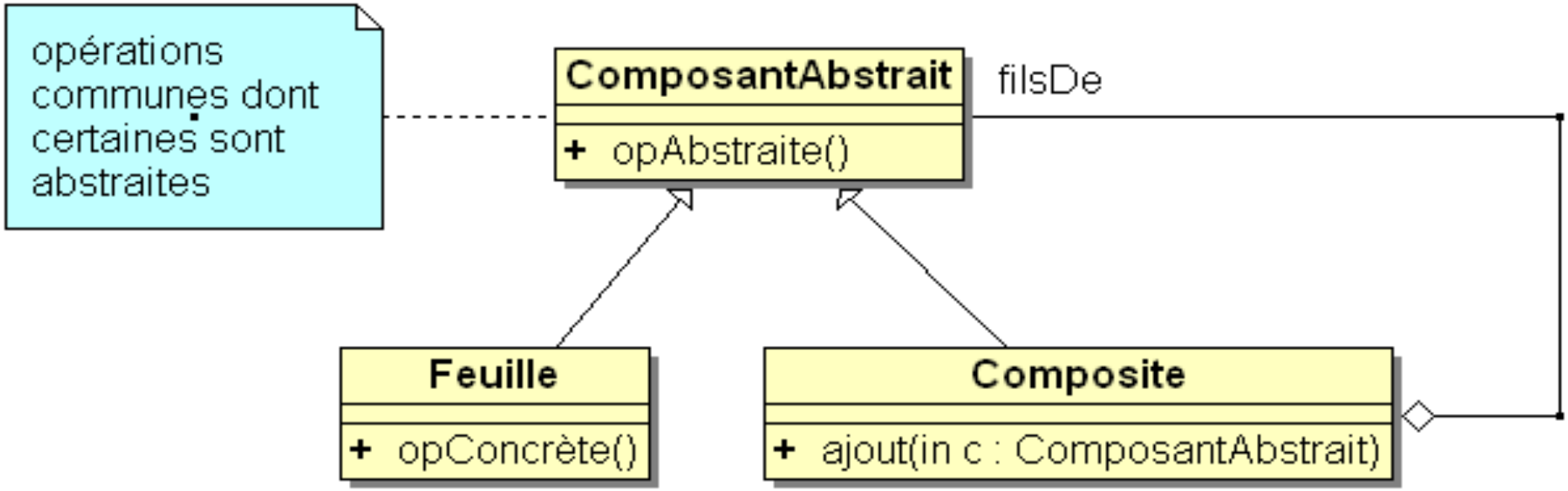


… au patron de conception « objets composites »



Hiérarchie d'agrégation d'objets ayant tous le même comportement

… au patron de conception « objets composites »



Hiérarchie d'agrégation d'objets ayant tous le même comportement

Ex de design Pattern