UML
Partie 2:
Analyse d'un projet

Analyse : Partie 2
UML (Unified Modeling Language)
Analyse d'un projet

SOMMAIRE

I) Analyser un projet

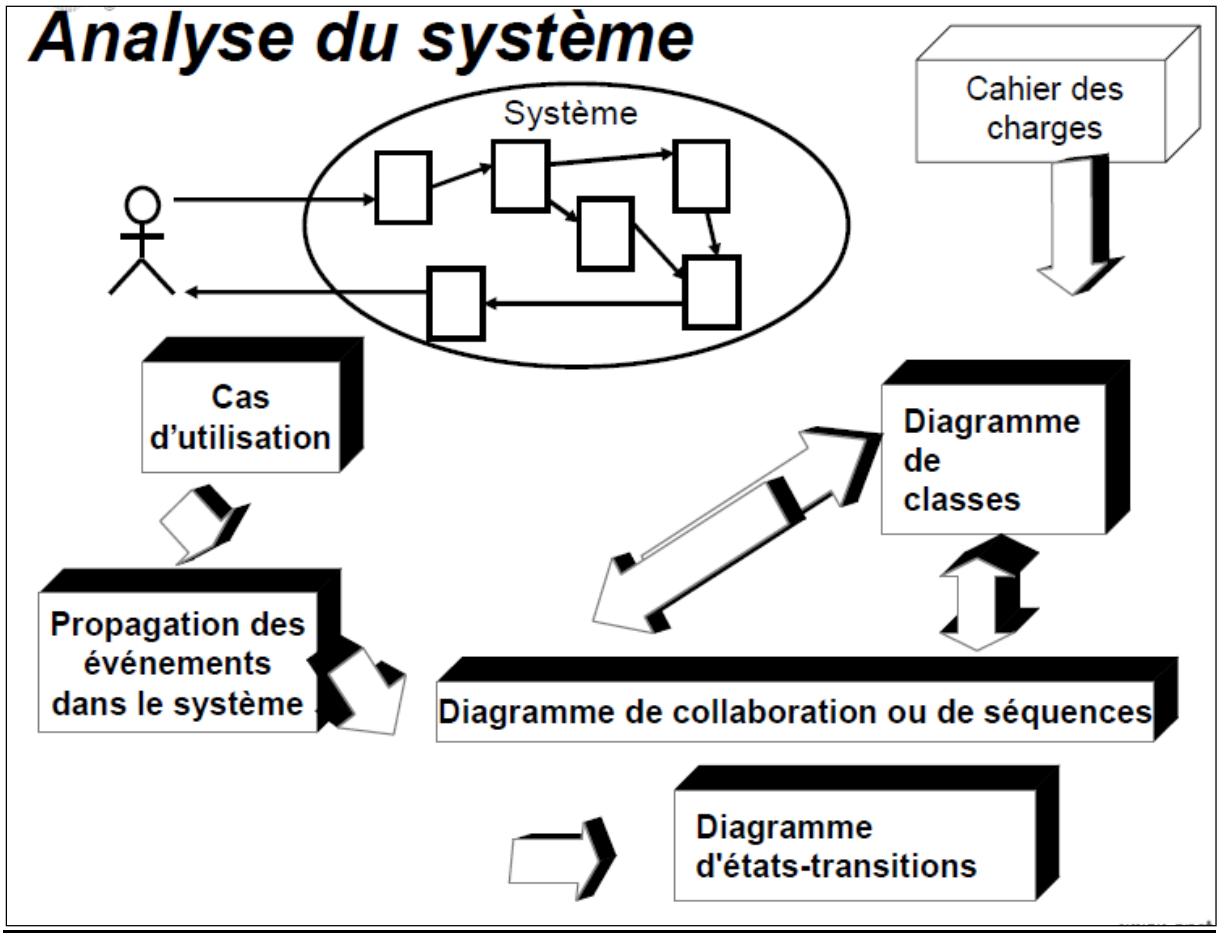
L'analyse d'un projet est constituée de différentes étapes.
Ces étapes donneront lieu à différents diagrammes.

Un diagramme donne à l'utilisateur un moyen de visualiser et de manipuler des éléments de modélisation.

On peut différencier ces diagrammes selon 3 points de vue :

- Point de vue fonctionnel
  - diagramme de cas d'utilisation
- Point de vue statique
  - diagramme de classe
  - diagramme de packages
- Point de vue dynamique
  - diagramme d'Etats-transitions
  -diagramme d'activité
  - diagramme de séquence

Analyser vos projets :

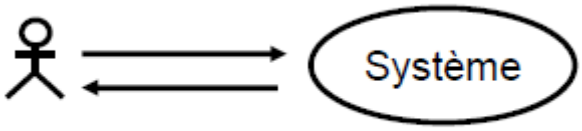


II) Cahier des charges : gestion de bibliothèque

- Un gérant de bibliothèque désire automatiser la gestion des prêts.
- Il commande un logiciel permettant aux utilisateurs de connaître les livres présents, d'en réserver jusqu'à 2. L'adhérent peut connaître la liste des livres qu'il a empruntés ou réservés.
- L'adhérent possède un mot de passe qui lui est donné à son inscription.
- L'emprunt est toujours réalisé par les employés qui travaillent à la bibliothèque. Après avoir identifié l'emprunteur, ils savent si le prêt est possible (nombre max de prêts = 5), et s'il a la priorité (il est celui qui a réservé le livre).
- Ce sont les employés qui mettent en bibliothèque les livres rendus et les nouveaux livres. Il leur est possible de connaître l'ensemble des prêts réalisés dans la bibliothèque.

A. Les diagrammes de cas d'utilisation :

Ce que doit faire le système sans spécifier comment il le fait.
Chaque cas d'utilisation peut être complété par un ensemble d'interactions successives d'une entité en dehors du système (l'utilisateur) avec le système lui-même.

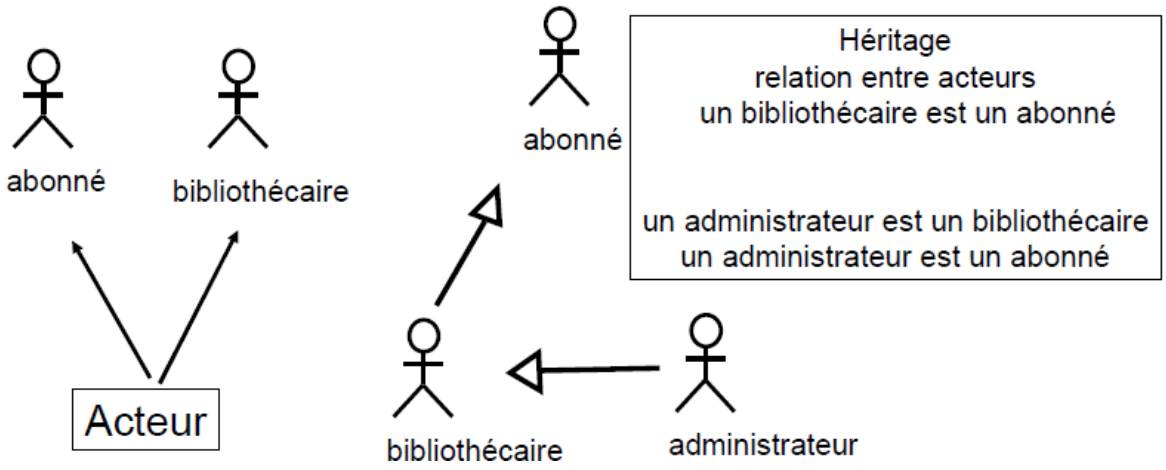


Un diagramme de cas d'utilisation définit :
. le système
. les acteurs
. les cas d'utilisations
. les liens entre acteurs et cas d'utilisations

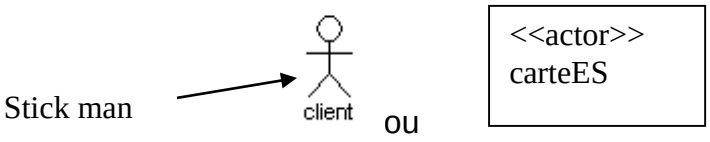
- *Les acteurs*

Un acteur représente un rôle joué par une entité externe (utilisateur humain, dispositif matériel ou autre système) qui interagit avec le système étudié.
Un acteur peut consulter et/ou modifier directement l'état du système, en émettant et/ou en recevant des messages susceptibles d'être porteur de données.

Relation entre acteurs : généralisation (héritage)



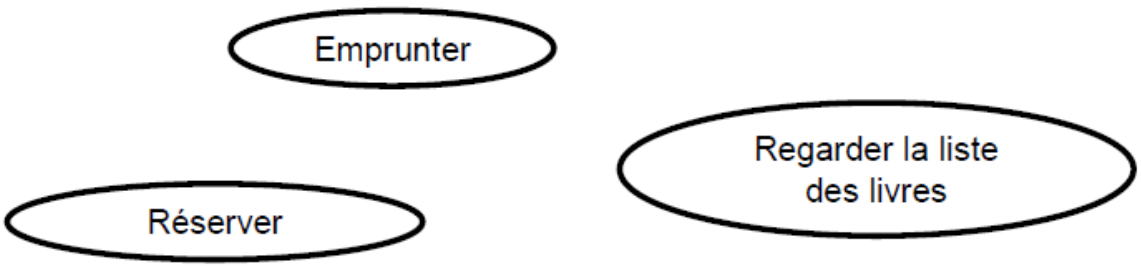
Représentation :



- *Les diagrammes de cas d'utilisation*

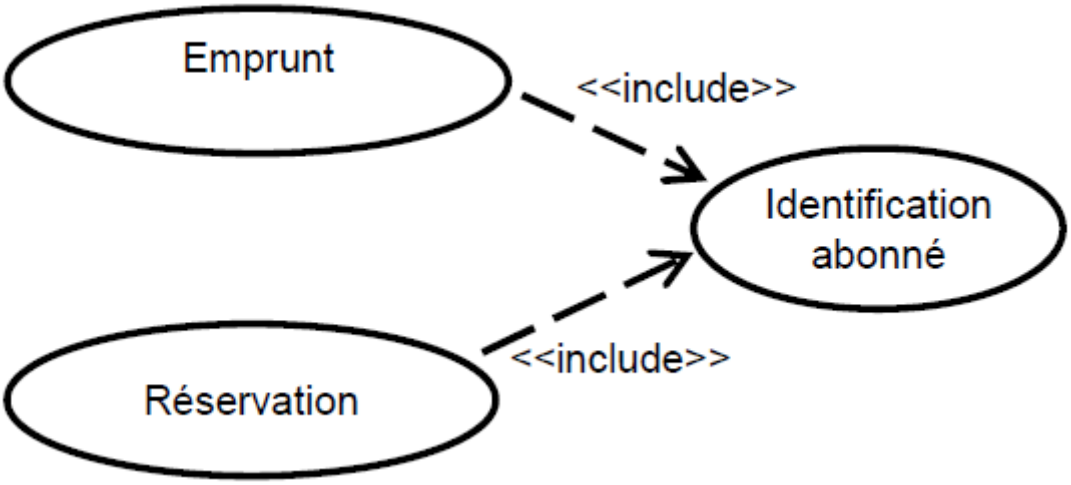
✓ Un cas d'utilisation est un moyen de représenter les différentes possibilités d'utiliser un système.
✓ Il exprime toujours une suite d'interactions entre un acteur et l'application.
✓ Il définit une fonctionnalité utilisable par un acteur.

Représentation :



1. « include »

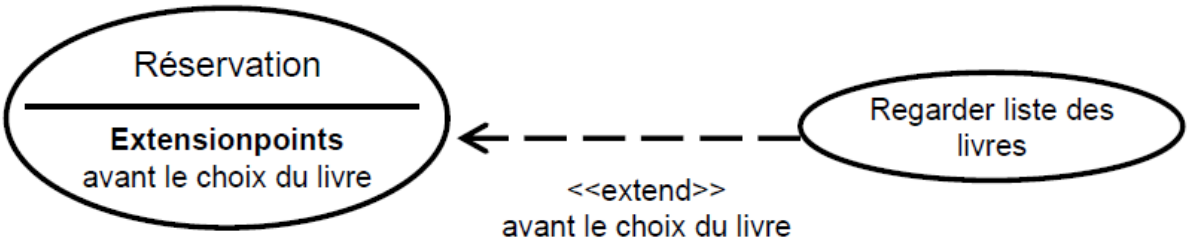
✓ La relation "*include*" précise qu'un cas d'utilisation contient le comportement défini dans un autre cas d'utilisation.
✓ Cette relation permet de mettre en commun des comportements communs à plusieurs cas d'utilisation..



---

2. « extend »

✓ La relation "*extend*" précise qu'un cas d'utilisation peut dans certains cas augmenter le comportement d'un autre cas d'utilisation.
✓ Une condition devra valider cette augmentation.
✓ Le point d'utilisation de cette augmentation peut être défini dans un "point d'extension".

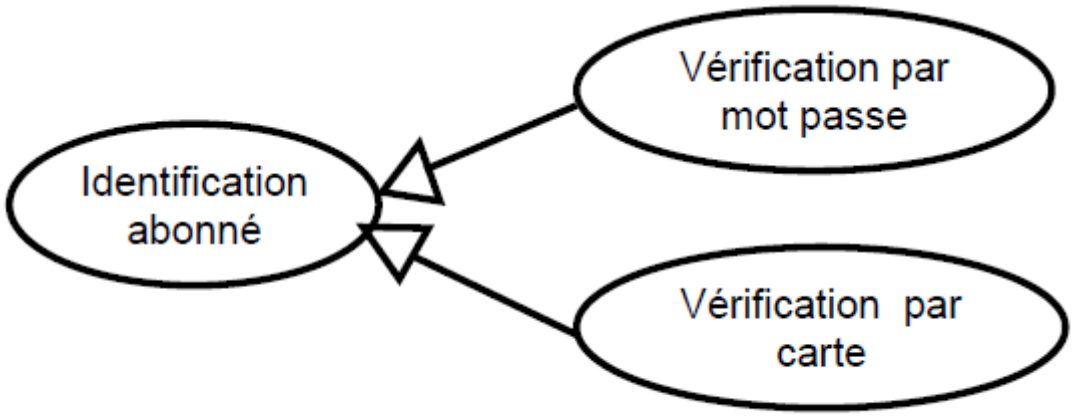


✓ Dans cet exemple, le cas d'utilisation "Regarder la liste des livres" augmente le cas d'utilisation d'une réservation, avant le choix du livre, si l'utilisateur en fait la demande.

3. « Généralisation »

✓ Cette relation "*est un*" introduit la notion d'héritage.
✓ Les cas d'utilisation "Vérification par mot passe" et "Vérification par carte" sont des spécialisations du cas d'utilisation "Identification abonné".

✓ Autrement dit : si l'on dit que l'on fait une "Identification abonné", ce peut être une "Vérification par carte" ou une "Vérification par mot passe".

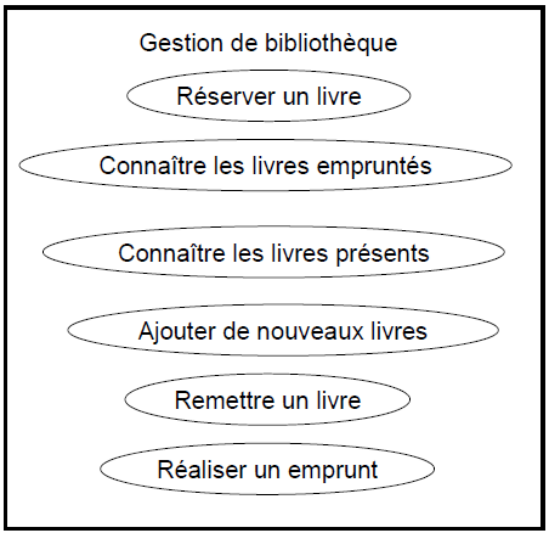


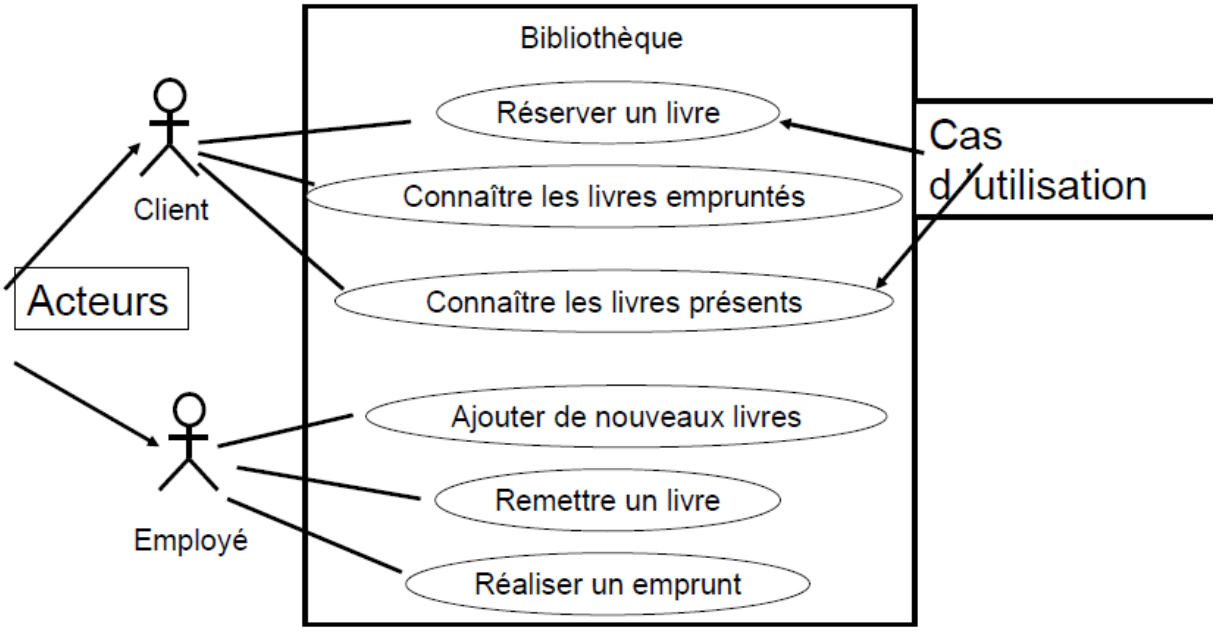
Résumé :

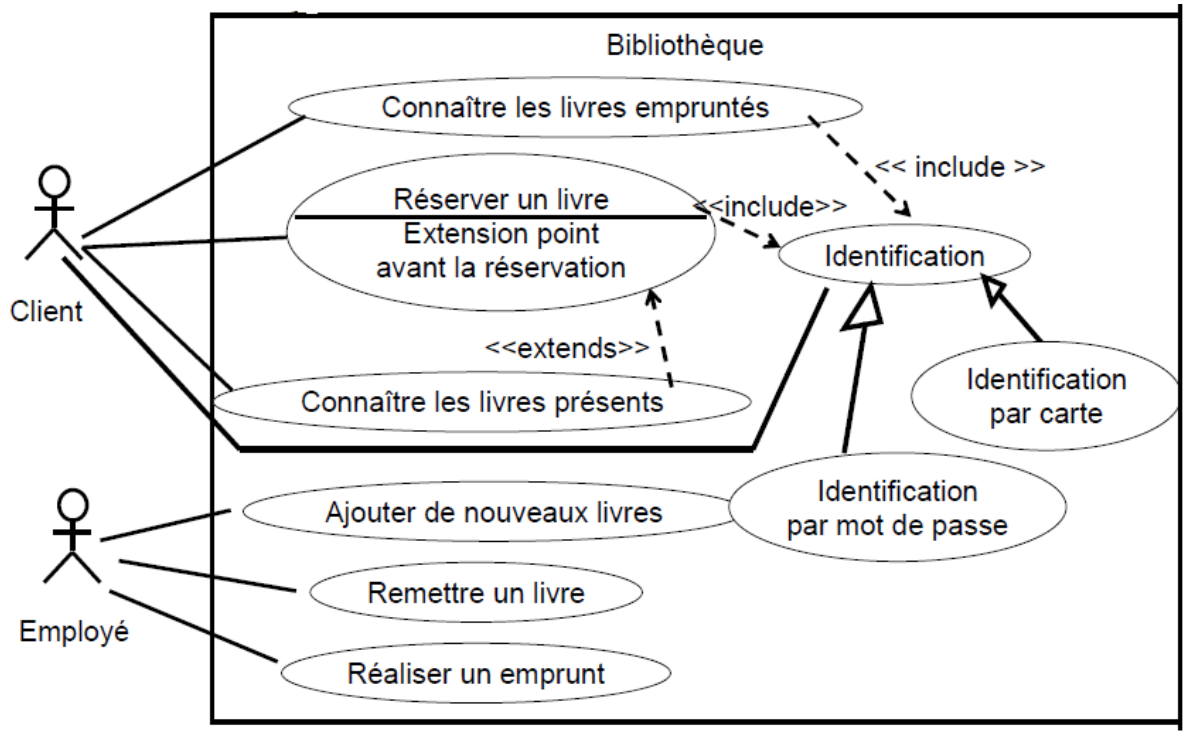
- ✓ Identifier les acteurs qui utilisent, qui gèrent, qui exécutent des fonctions spécifiques.
- ✓ Organiser les acteurs par relation d'héritage.
- ✓ Pour chaque acteur, rechercher les cas d'utilisation avec le système. En particulier, ceux qui modifient l'état du système ou qui attendent une réponse du système.
- ✓ Ne pas oublier les variantes d'interactions (cas d'erreur, cas interdits).
- ✓ Organiser ces interactions par héritage, par utilisation et par extension.

- Construisons le Diagramme des cas d'utilisation de notre cahier des charges

Le système définit l'application informatique, il ne contient donc pas les acteurs, mais les cas d'utilisation et leurs associations.

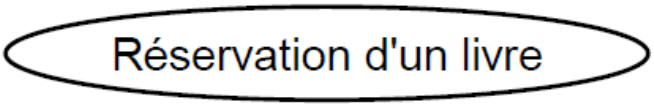



Bibliothèque
Réserver un livre
Connaître les livres empruntés
Connaître les livres présents
Ajouter de nouveaux livres
Remettre un livre
Réaliser un emprunt
Client
Employé
Acteurs
Cas d'utilisation


Bibliothèque
Connaître les livres empruntés
Réserver un livre
Extension point
avant la réservation
Connaître les livres présents
Ajouter de nouveaux livres
Remettre un livre
Réaliser un emprunt
Identification
Identification par carte
Identification par mot de passe
<< include >>
<<include>>
<<extends>>
Client
Employé

B. Les scénarios

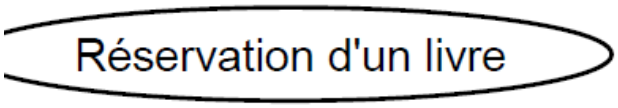
✓ La description d'un cas d'utilisation se fait par des scénarios qui définissent la suite logique des interactions qui constituent ce cas.
✓ On peut définir des scénarios simples ou des scénarios plus détaillés faisant intervenir les variantes, les cas d'erreurs,etc.
✓ Cette description se fait de manière simple, par un texte compréhensible par les personnes du domaine de l'application.
✓ Elle précise ce que fait l'acteur et ce que fait le système
✓ La description détaillée pourra préciser les contraintes de l'acteur et celles du système.



description simplifiée

Le client se présente devant un terminal:

- (1) Le système affiche un message d 'accueil.
- (2) Le client choisit l'opération réservation parmi les différentes opérations proposées.
- (3) Le système lui demande de s'authentifier.
- (4) Le client donne son identification (nom, mot de passe).
- (5) Le système lui demande de choisir un livre.
- (6) Le client précise le livre qu'il désire.
- (7) Le système lui précise si un exemplaire du livre lui est réservé.



description détaillée

Pré-conditions: Le client doit être inscrit à la bibliothèque
Le client ne doit pas avoir atteint le nombre maximum de réservation
Un exemplaire du livre doit être enregistré

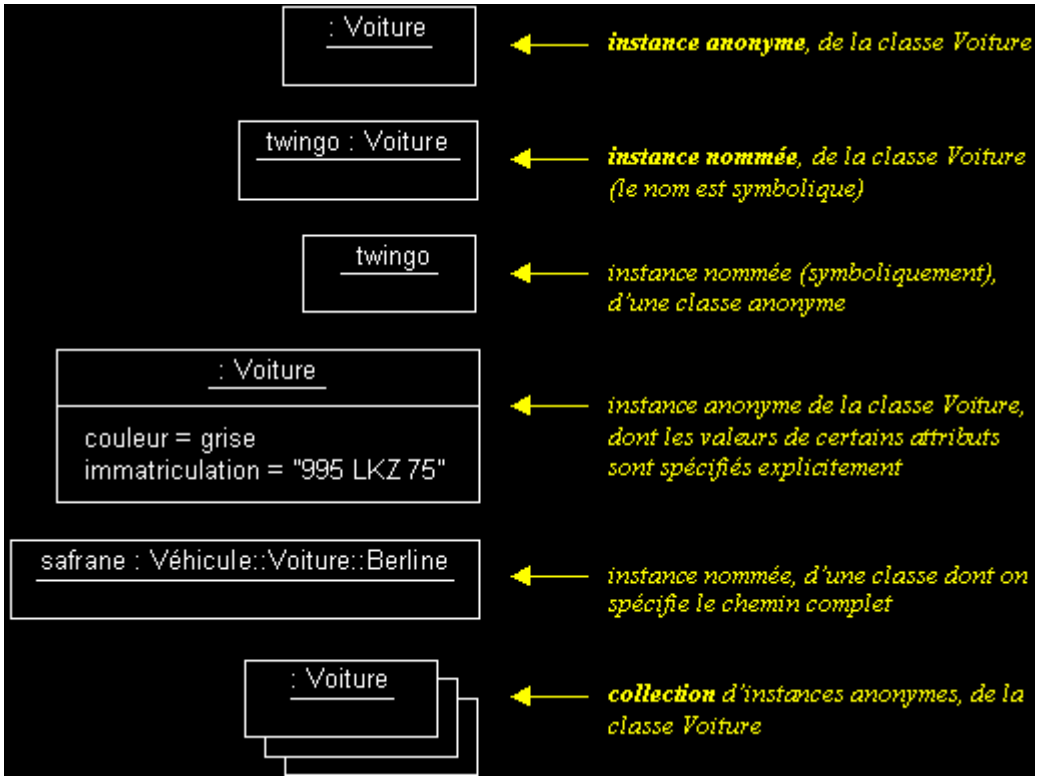
Post-conditions: (Si l'opération s'est bien déroulée)
Le client a une réservation supplémentaire
Le nombre d'exemplaires disponibles du livre est décrémenté de 1

Réservation d'un livre — Cas normal — description détaillée

1. (1) Le système affiche un message d 'accueil sur le terminal avec un choix d'opérations
2. (2) Le client choisit l'opération réservation parmi les différentes opérations proposées.
3. (3) Le système lui demande de s'authentifier.
4. (4) Le client donne son identification (nom, mot de passe).
5. (5) Le système demande le titre du livre en donnant la possibilité de choisir dans une liste.
6. (6) Le client précise le livre qu'il désire.
7. (7) Le système lui précise que un exemplaire du livre lui est réservé.

variante 1	en (6) le client demande à connaître les livres présents
variante 2	en (4) le client n'est pas reconnu, dans ce cas, tant qu'il n'est pas reconnu, on lui redemande de s'authentifier
variante 3	en (4) le client est reconnu, mais le mot de passe est incorrect, il peut avoir 5 essais, par la suite le client sera interdit pendant 1 heure.
variante 4	en (4) le client n'a plus le droit de réserver
variante 5	en (7) le livre n'est pas libre

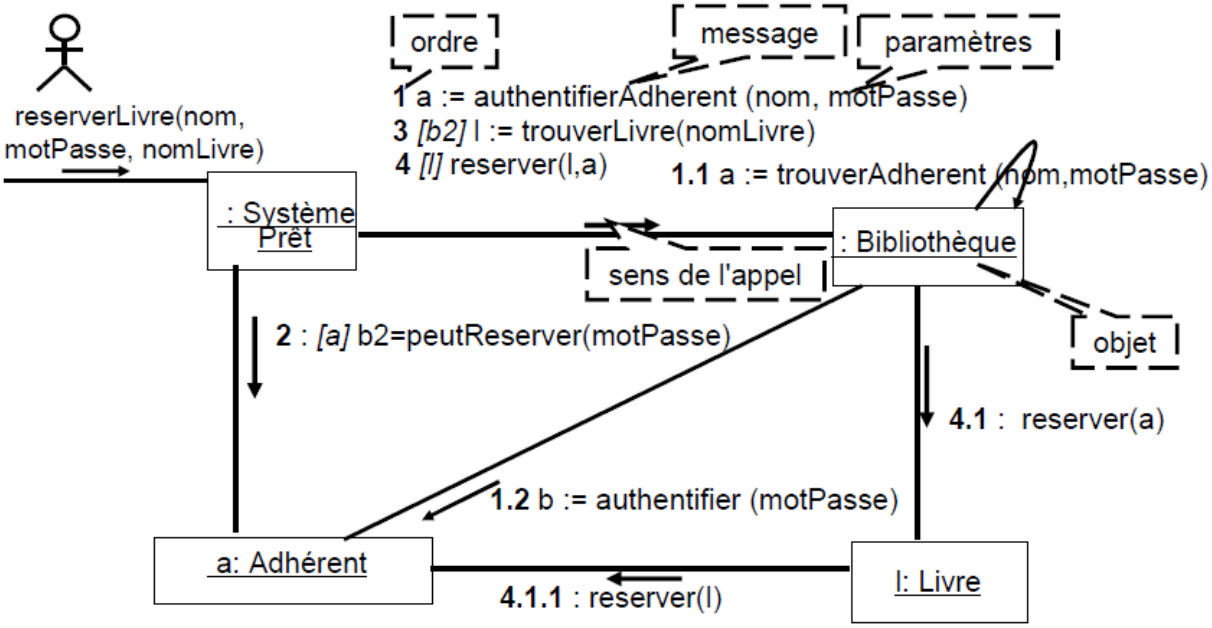
C) Instances et diagrammes d'objets



Les messages échangés entre les objets peuvent être de types différents :

synchrone →	L'expéditeur est bloqué pendant le traitement du message par l'expéditeur.
retour d'appel - - - - - - →	Un message synchrone peut être un appel de procédure, le retour peut être représenté (optionnel, le retour est implicite)
asynchrone <=UML1.3 ↘ / >=UML1.4 →	L'expéditeur continue son exécution pendant le traitement du message
minuté	Comme le synchrone, mais un chien de garde est positionné, c'est à dire que l'expéditeur se réveille au bout d'un certain temps s'il ne reçoit pas de réponse.

➔ Le diagramme de collaboration :



➔ Le diagramme de séquence :

Suite aux descriptions textuelles, le scénario peut être représenté en utilisant un diagramme de séquences.
Le diagramme de séquences :

- ✓ Met en évidence l'aspect temporel (haut vers le bas)
- ✓ Un objet a une ligne de vie représentée par une ligne verticale en pointillé.

- ✓ Une flèche reçue par un objet se traduit par l'exécution d'une opération.
- ✓ La durée de vie de l'opération est symbolisée par un rectangle.
- ✓ Certains objets vivent pendant tout le diagramme, d'autres sont activés et/ou meurent pendant la séquence.
- ✓ Il est possible de définir des choix et des itérations.

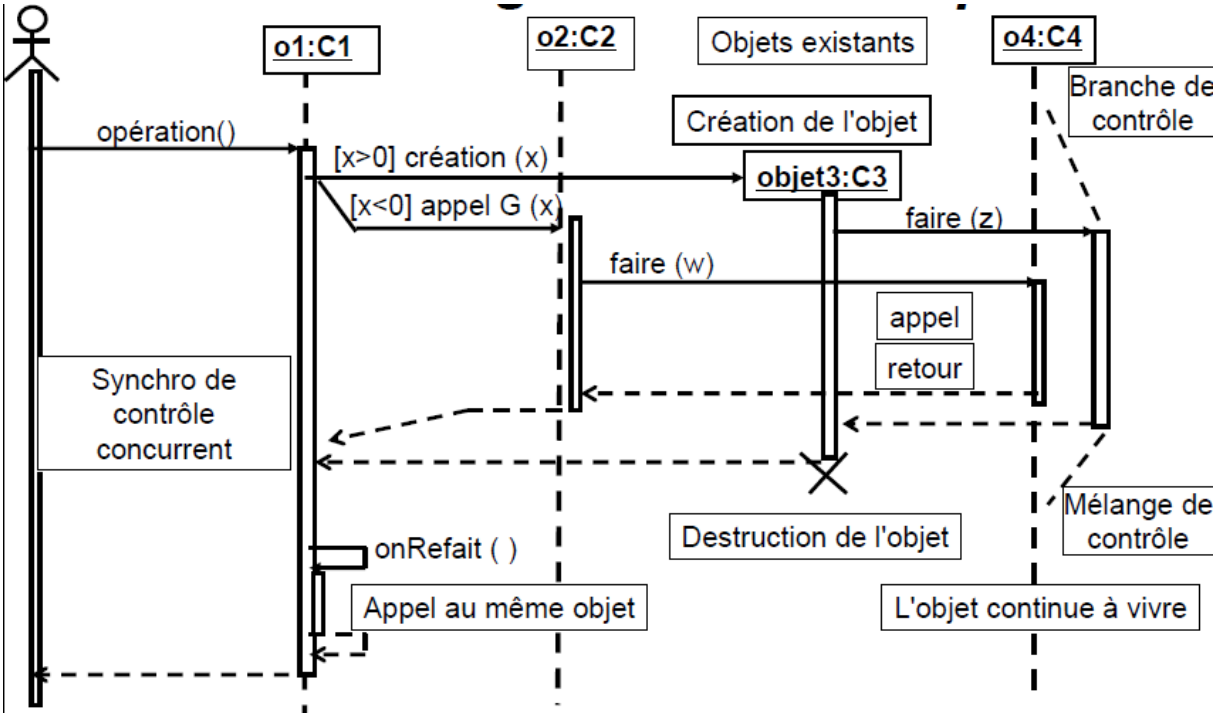
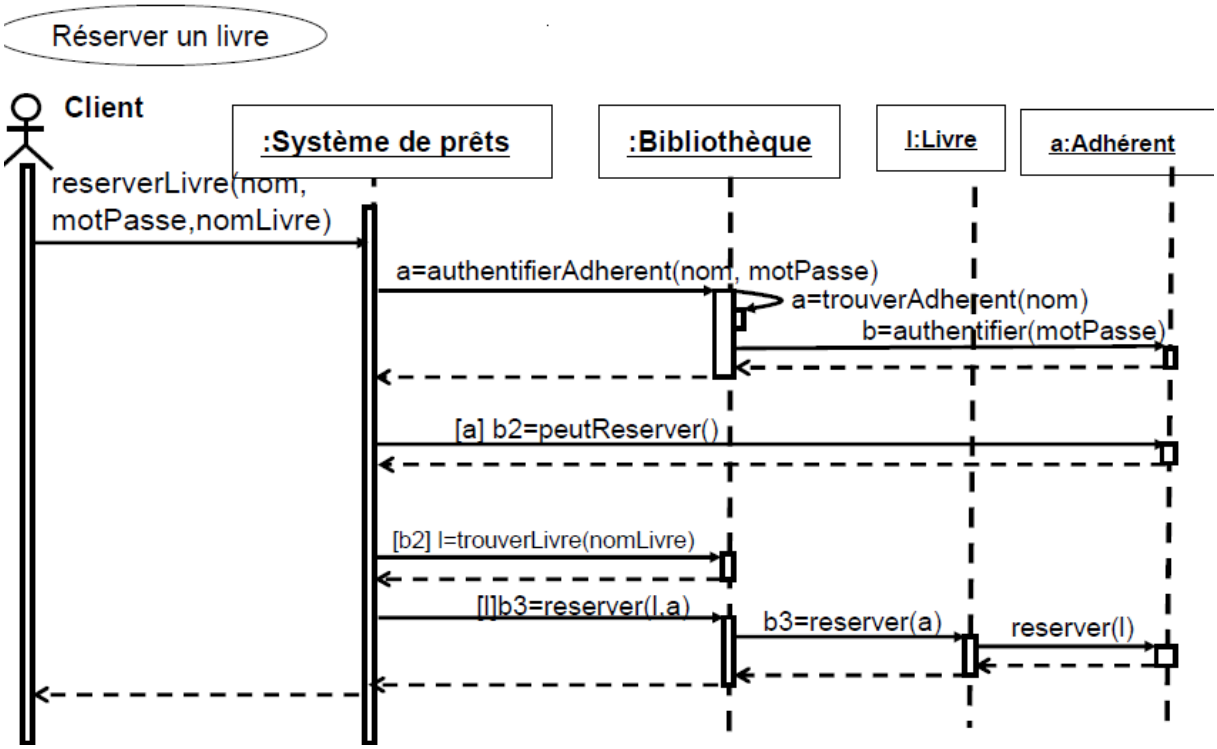
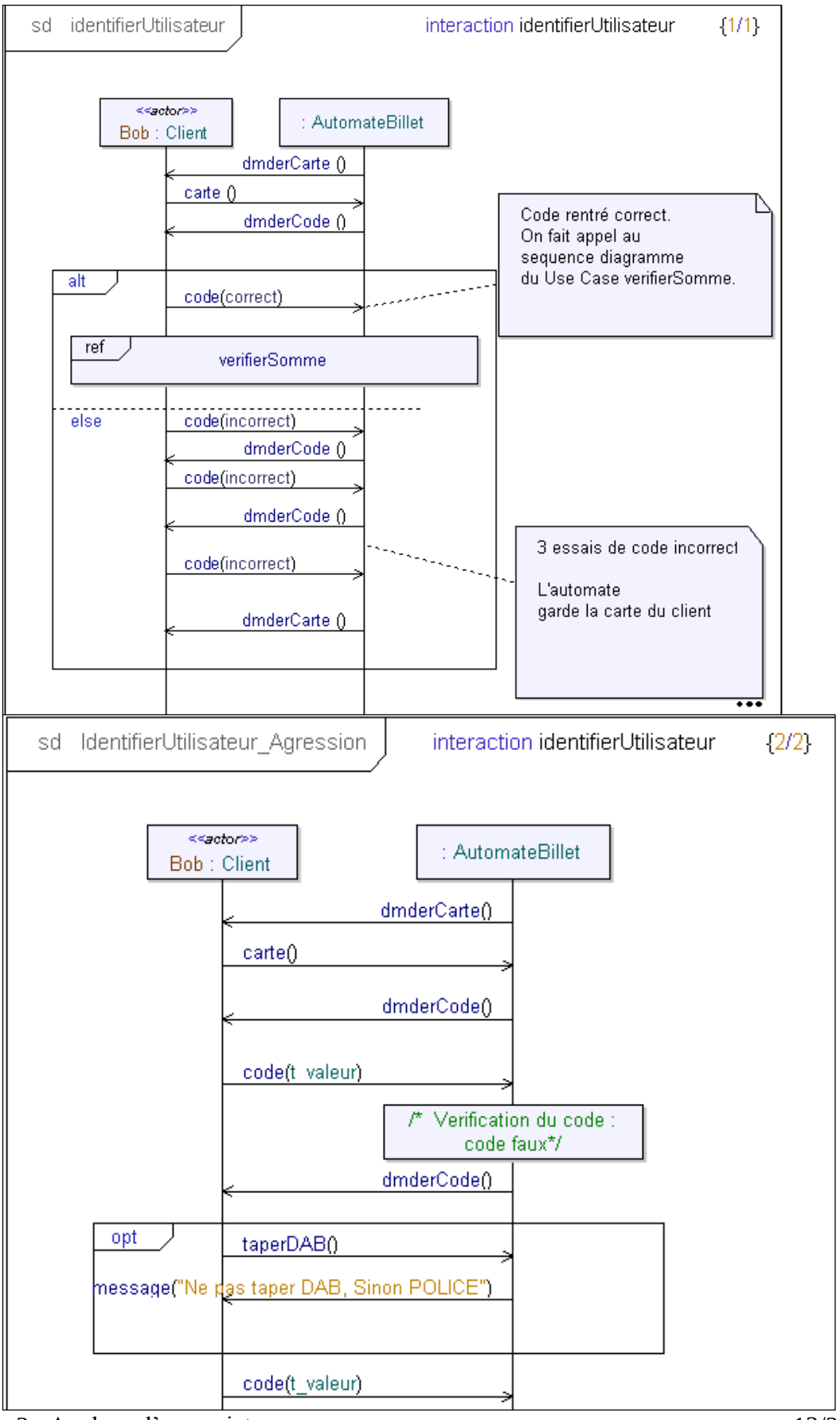
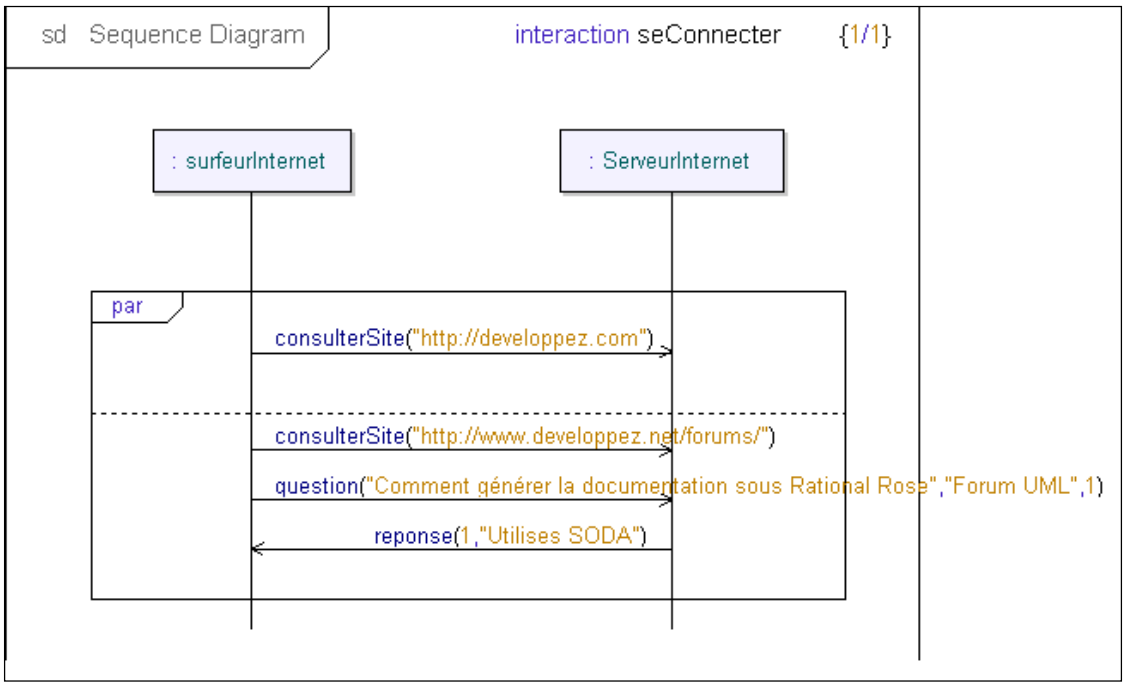


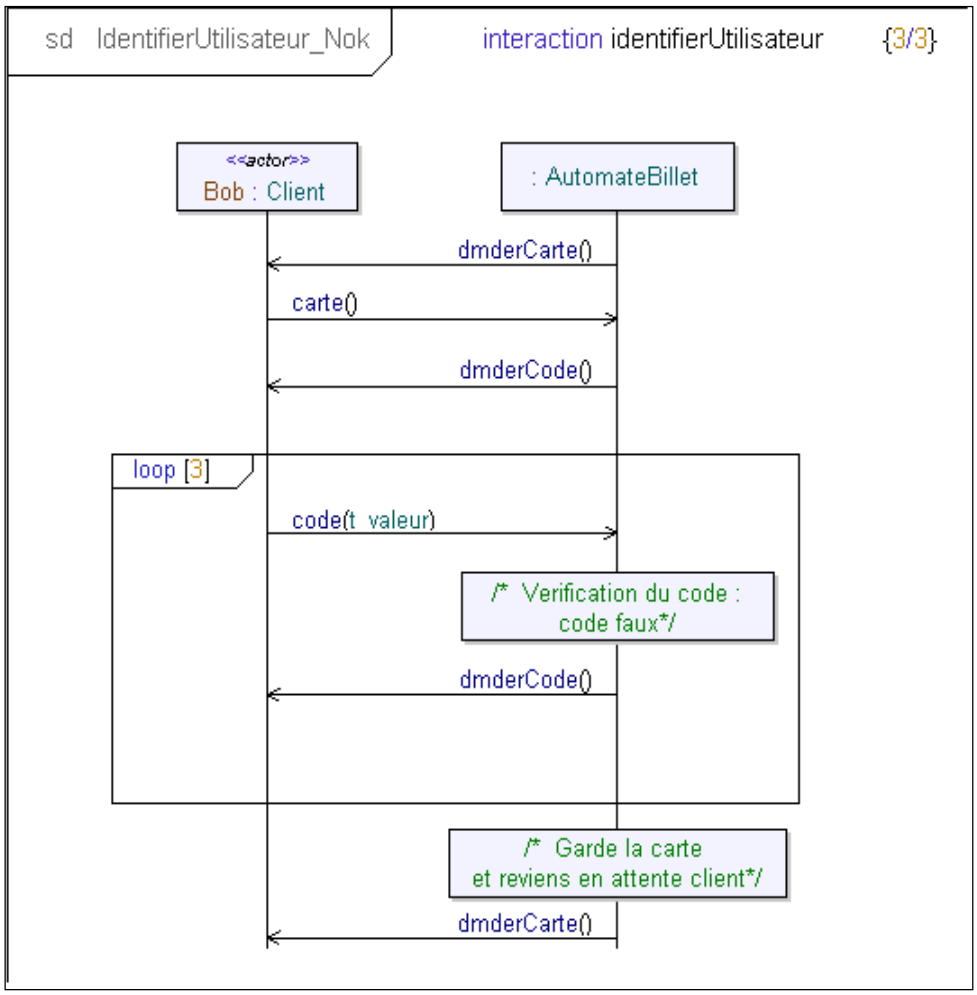
Diagramme de séquence pour le cas d'utilisation : Réserver un livre



Autres représentation en UML2 :

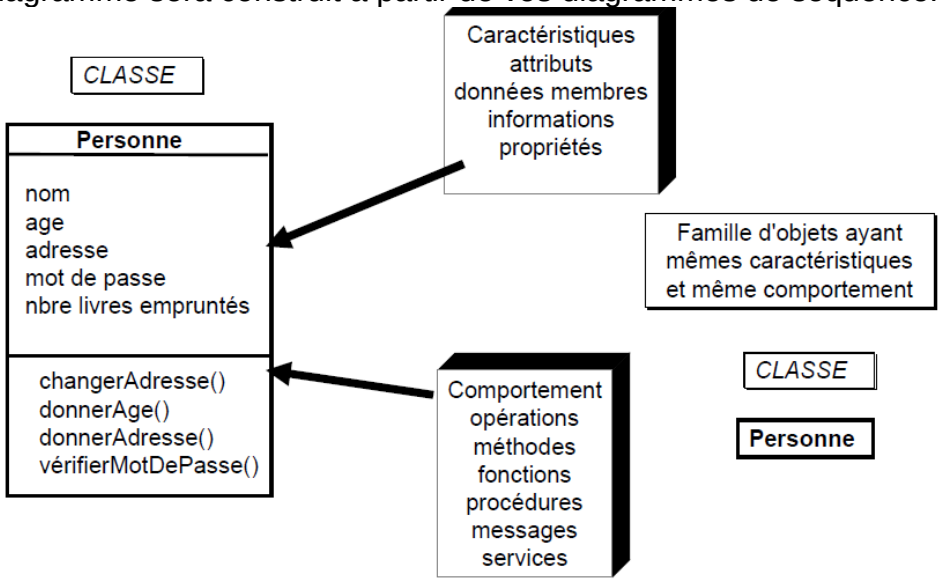


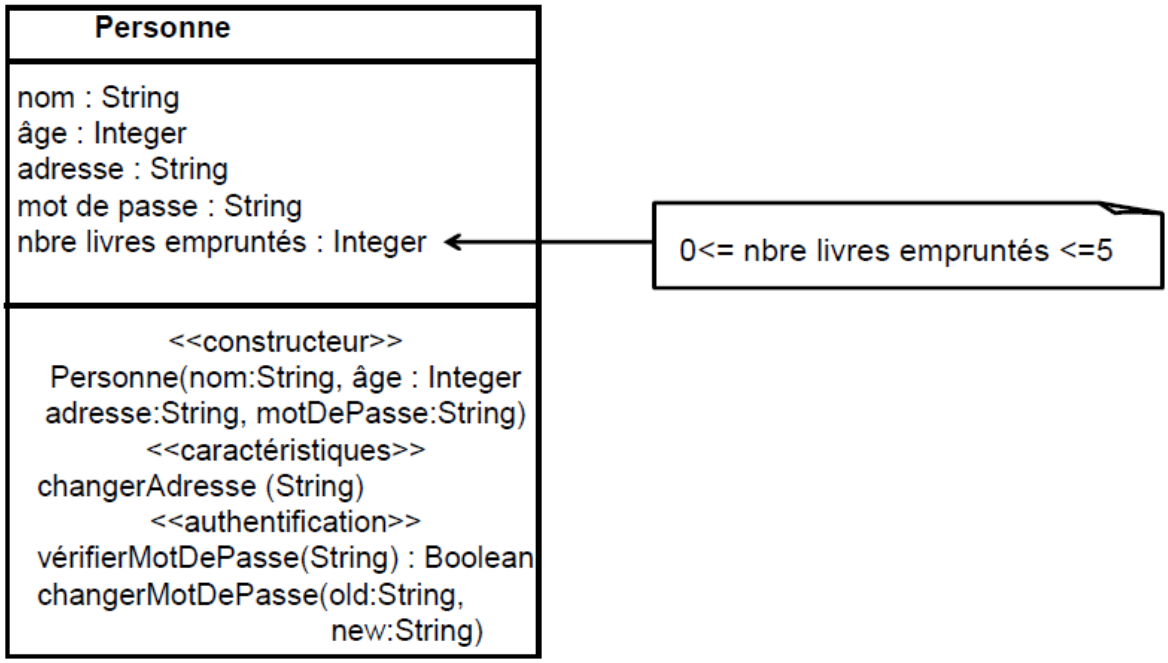

sd Sequence Diagram
interaction seConnecter {1/1}
: surfeurInternet
: ServeurInternet
par
consulterSite("http://developpez.com")
consulterSite("http://www.developpez.net/forums/")
question("Comment générer la documentation sous Rational Rose","Forum UML",1)
reponse(1,"Utilises SODA")


sd IdentifierUtilisateur_Nok
interaction identifierUtilisateur {3/3}
<<actor>>
Bob : Client
: AutomateBillet
dmderCarte()
carte()
dmderCode()
loop [3]
code(t_valeur)
/* Verification du code : code faux*/
dmderCode()
/* Garde la carte et reviens en attente client*/
dmderCarte()

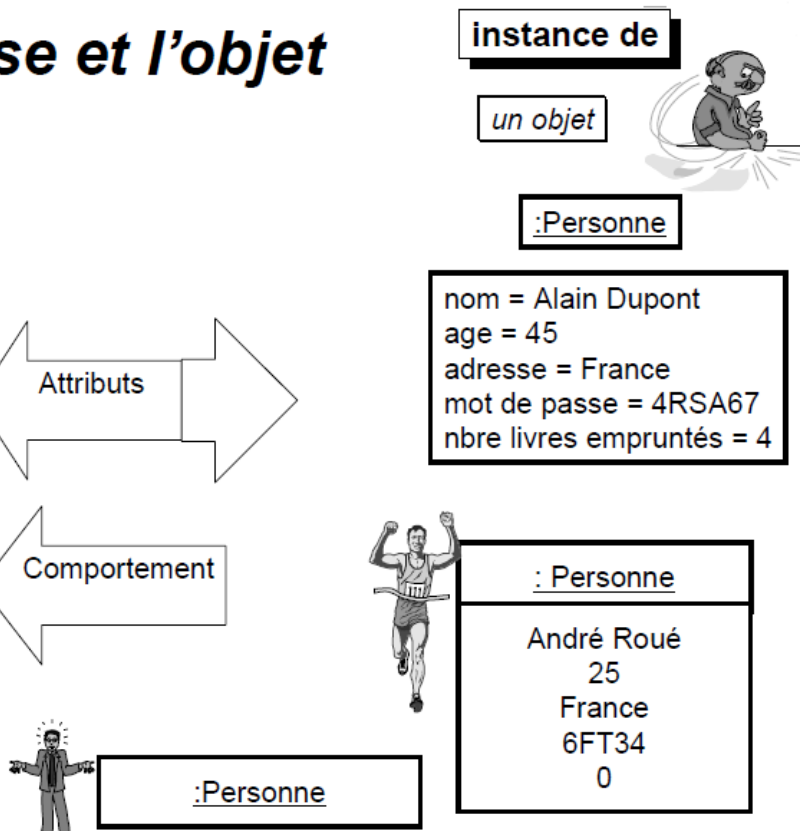
D) Diagramme de classes

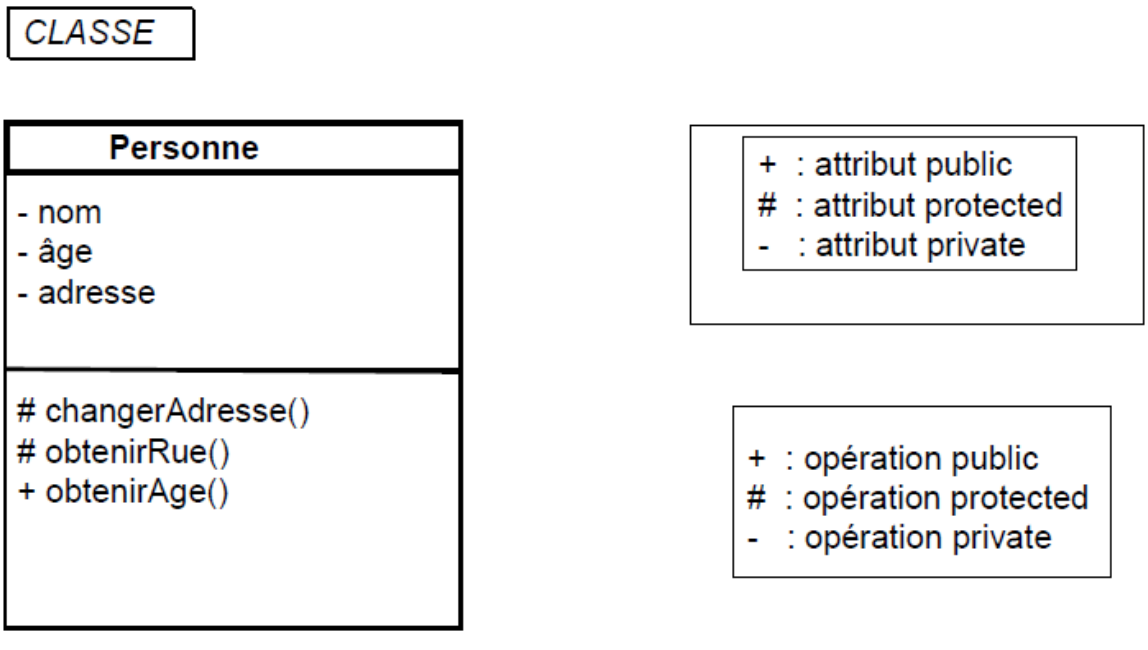
Diagramme de classe : qui représente les associations entre les classes. Ce diagramme sera construit à partir de vos diagrammes de séquence.



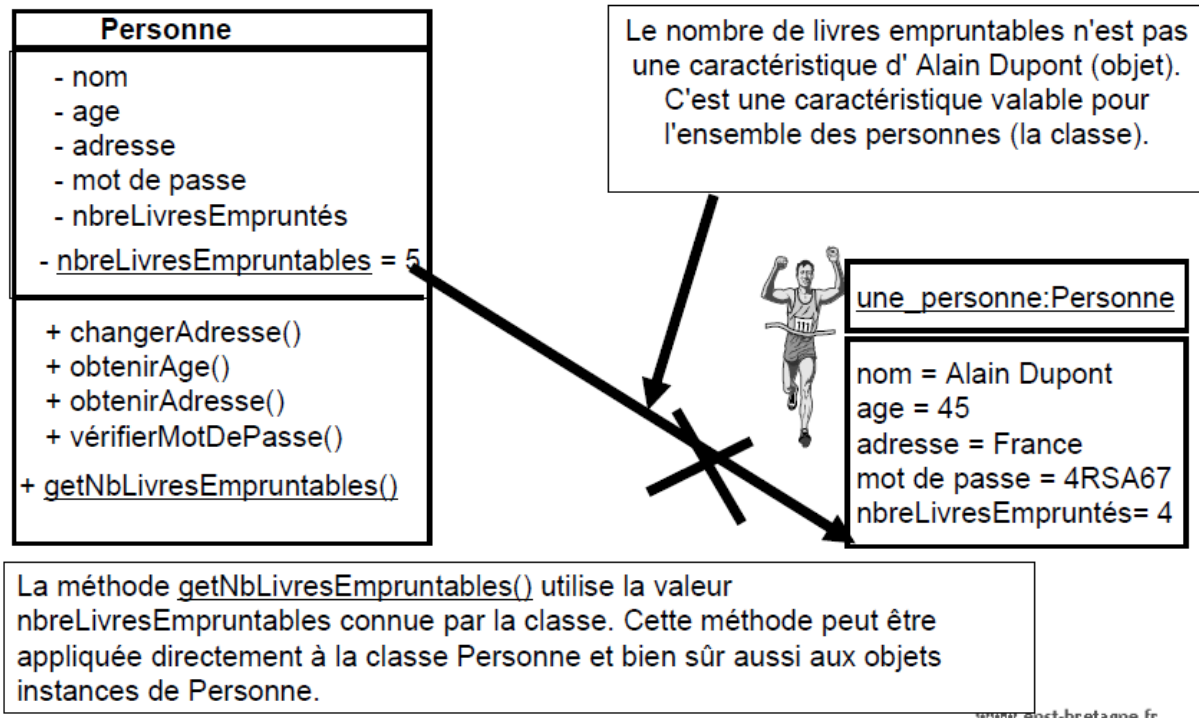


La classe et l'objet

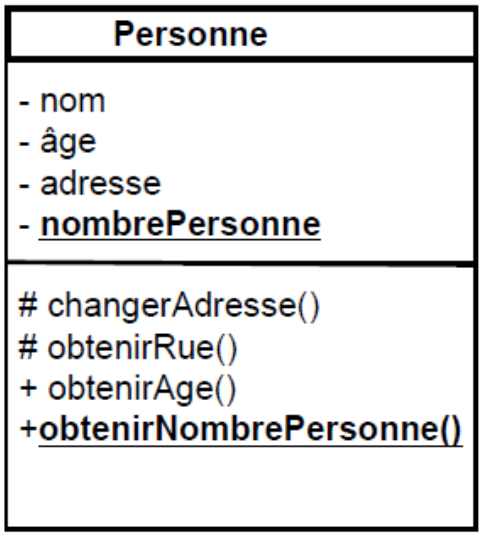


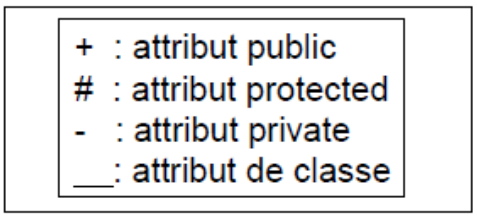


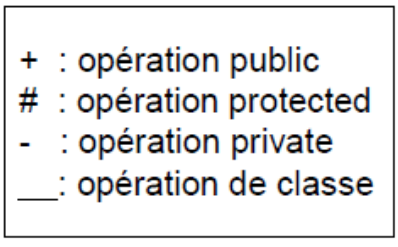
Attributs et opérations de classe

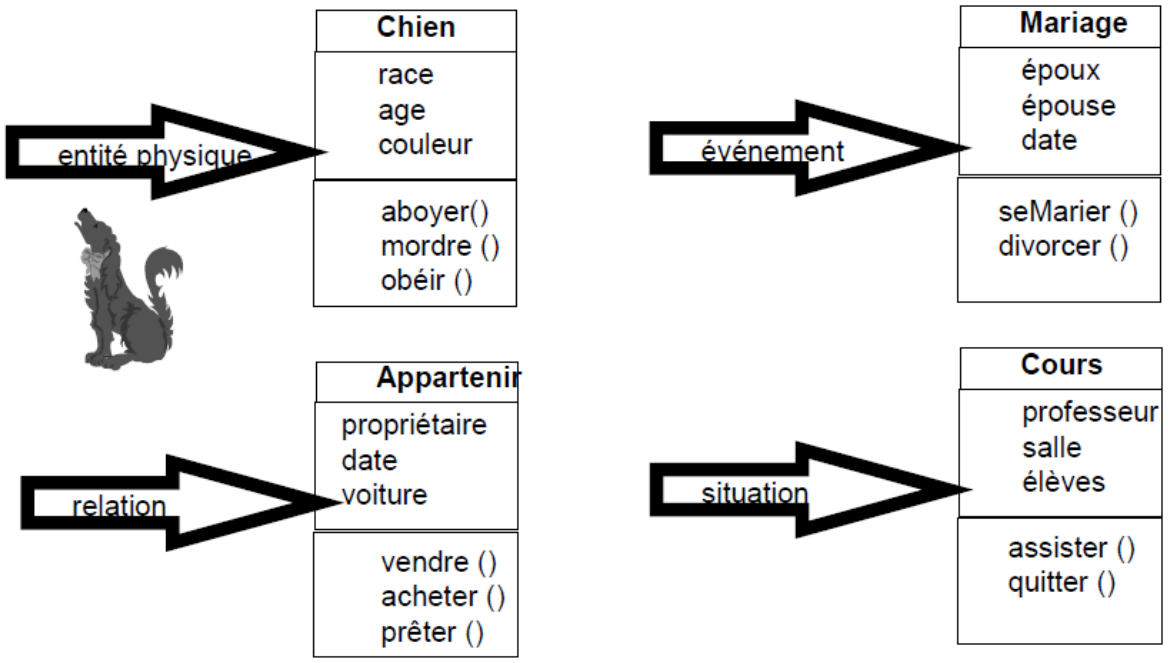


CLASSE









1) Héritage

BUT

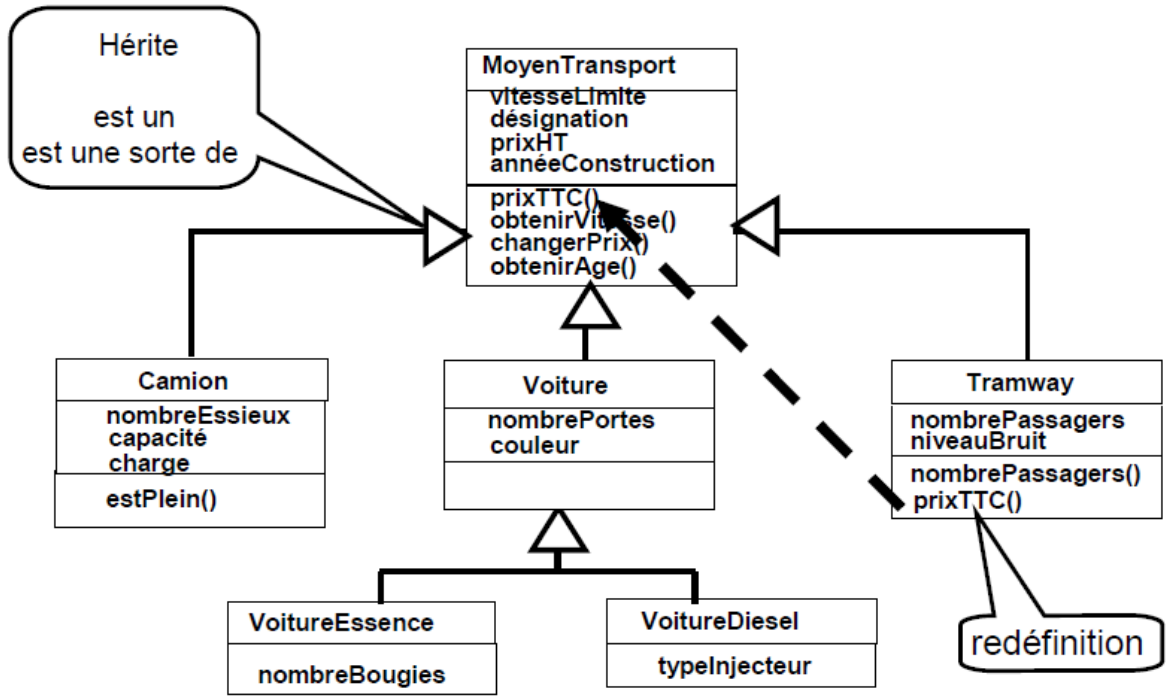
- ✓ Permettre une réutilisation optimale des classes déjà écrites, utilisées et validées
- ✓ Réutilisation de la structure des données héritées
- ✓ Réutilisation du code des services hérités

PRINCIPE

- ✓ Ne pas modifier les classes déjà écrites cela modifierait l'utilisation qui en est faite.

- ✓ Ne pas hésiter à créer des classes, extensions d'autres déjà validées

Héritage : adaptation



✓

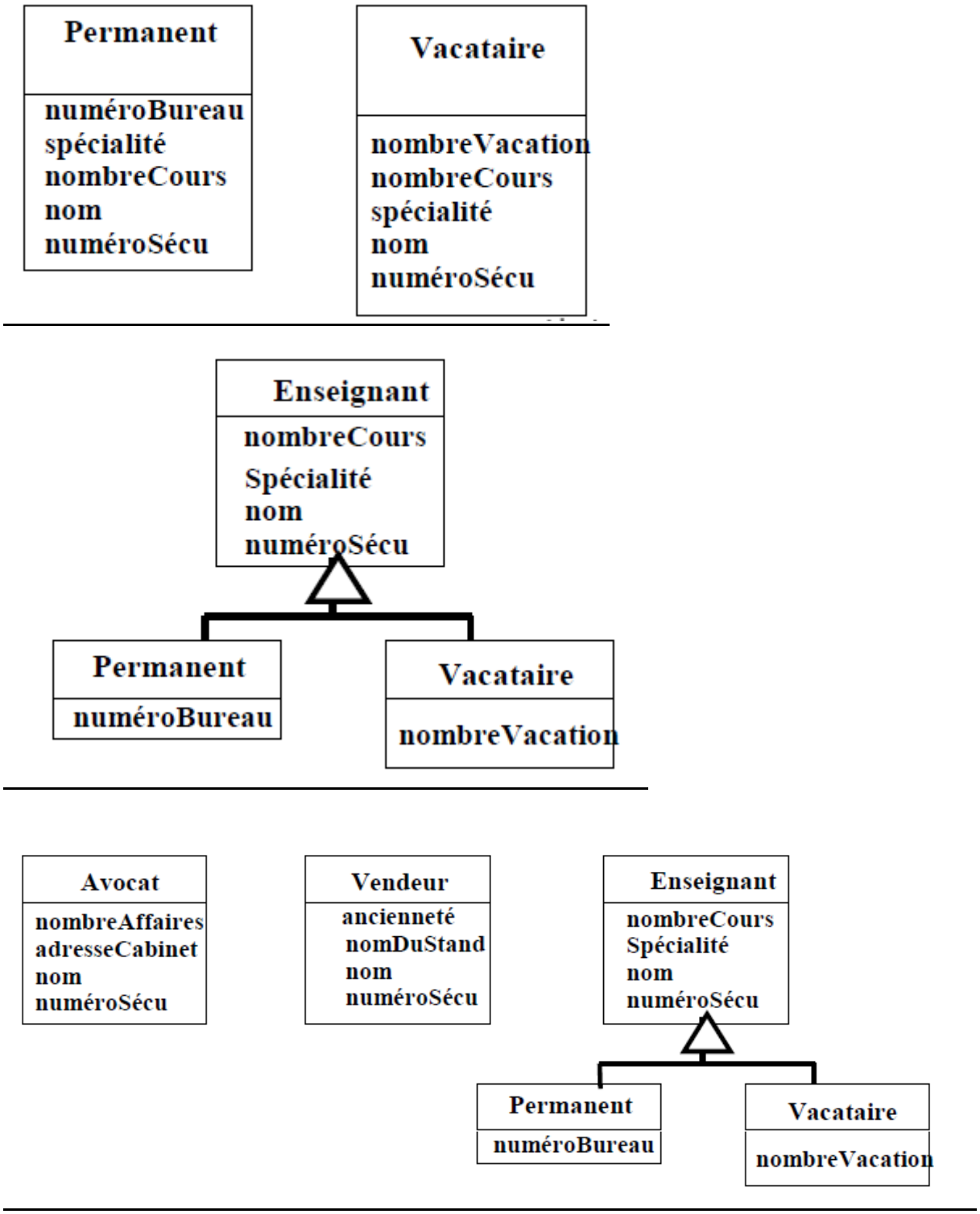
Ajout d'une classe de base

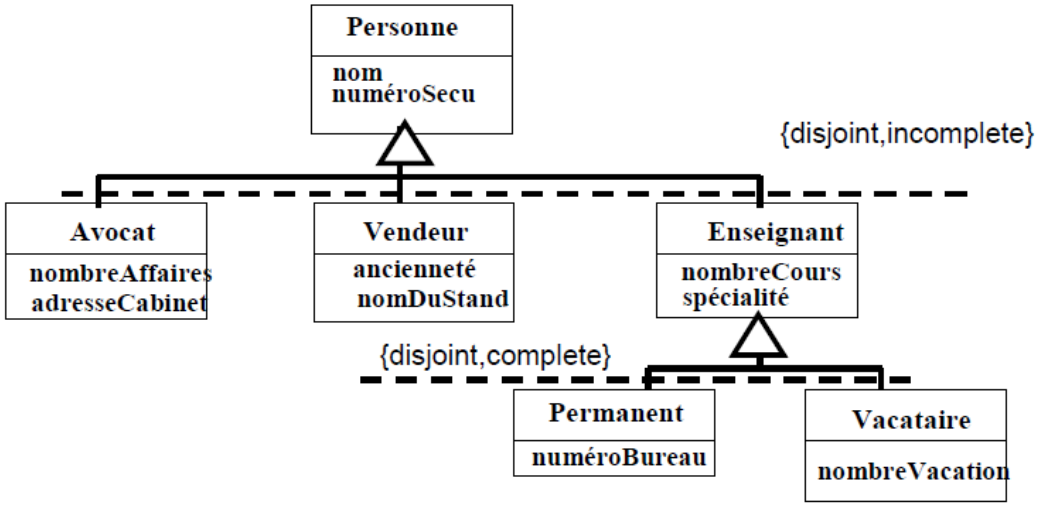
BUT

- ✓ Permettre une factorisation des caractéristiques et des comportements communs à plusieurs classes
- ✓ Mise en commun des structure des données
- ✓ Mise en commun du code des services

PRINCIPE

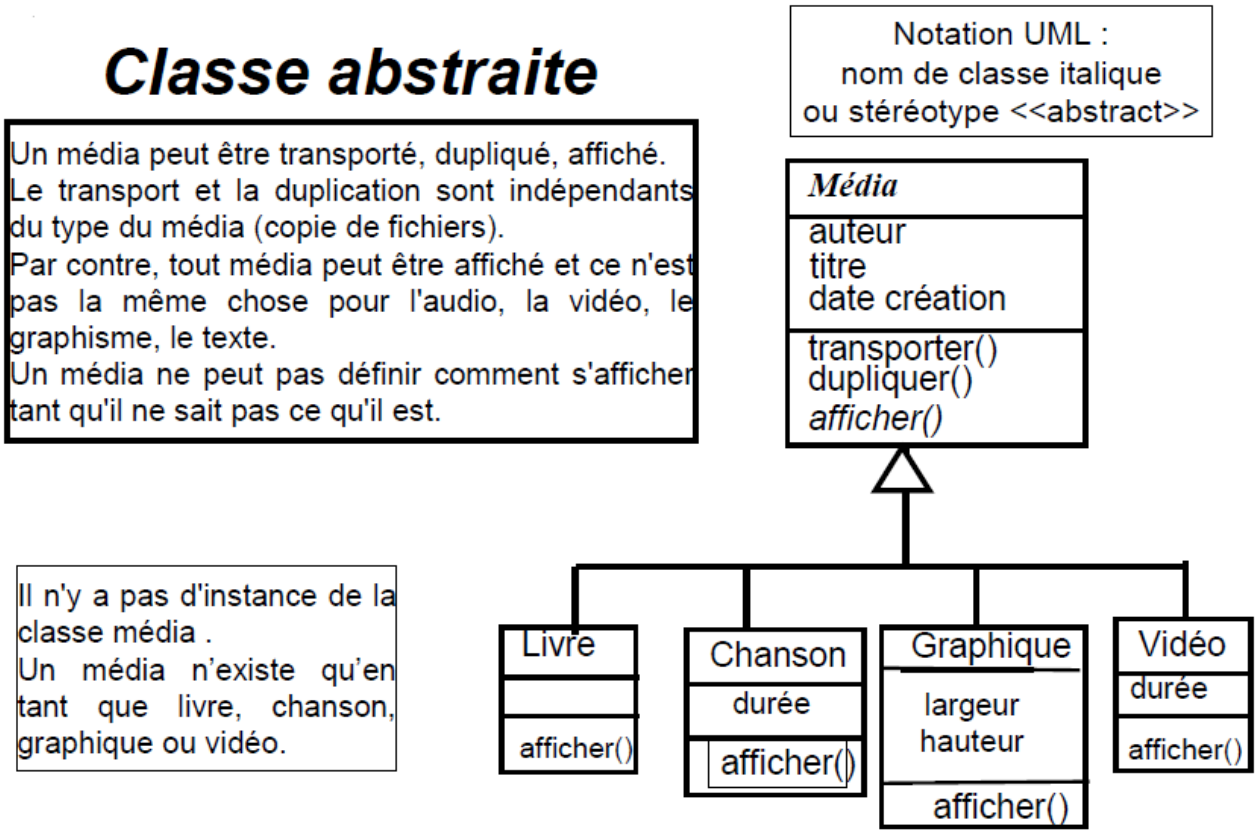
- ✓ Lorsque plusieurs classes ont des caractéristiques et des comportements communs la création d'une classe ancêtre permet de regrouper ce qui est commun.
- ✓ Cette classe ancêtre peut correspondre à une classe concrète ou à une classe abstraite

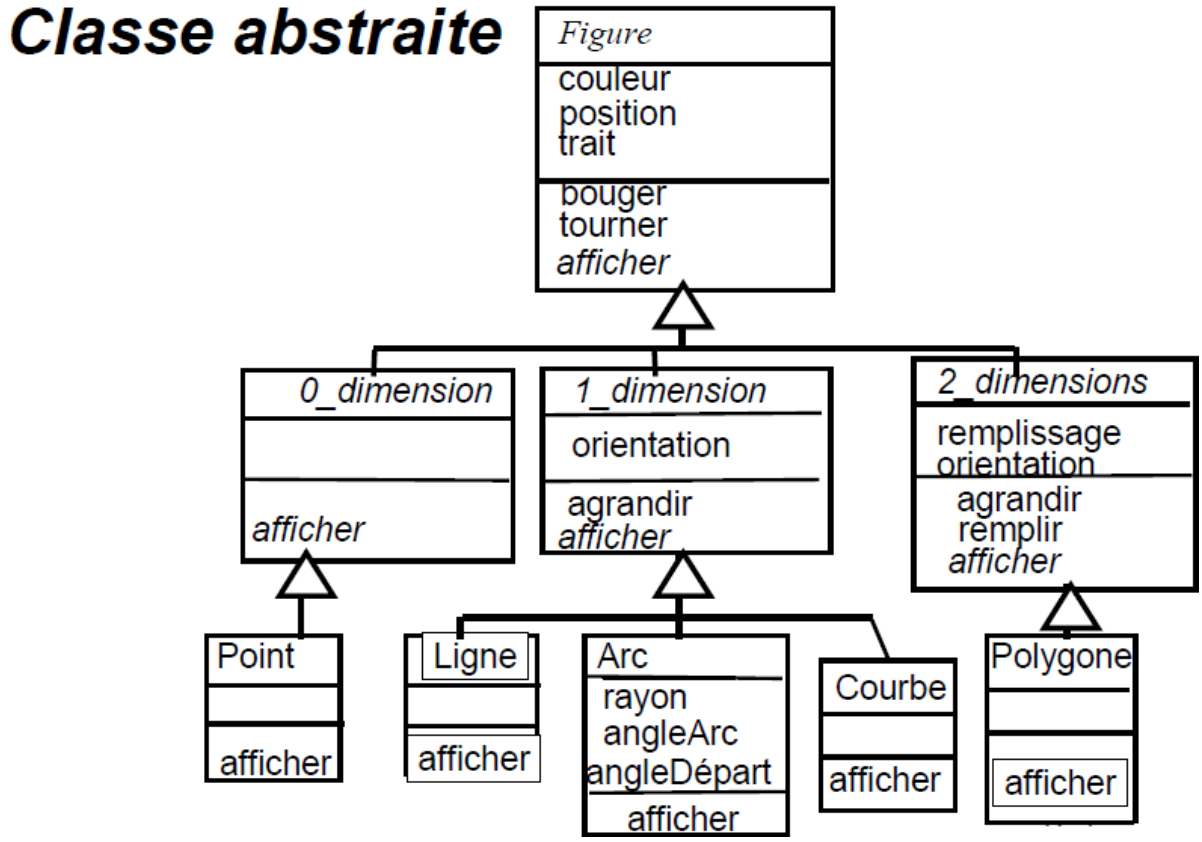

Permanent
numéroBureau
spécialité
nombreCours
nom
numéroSécu
Vacataire
nombreVacation
nombreCours
spécialité
nom
numéroSécu
Enseignant
nombreCours
Spécialité
nom
numéroSécu
Permanent
numéroBureau
Vacataire
nombreVacation
Avocat
nombreAffaires
adresseCabinet
nom
numéroSécu
Vendeur
ancienneté
nomDuStand
nom
numéroSécu
Enseignant
nombreCours
Spécialité
nom
numéroSécu
Permanent
numéroBureau
Vacataire
nombreVacation



---

2) Classes abstraites





3) Polymorphisme

BUT

- ✓ Créer des sous-types (sous-classes). Une sous-classe sera du même type que la classe dont elle hérite (super-classe).
- ✓ Ceci permet de mettre en oeuvre le polymorphisme et la liaison dynamique

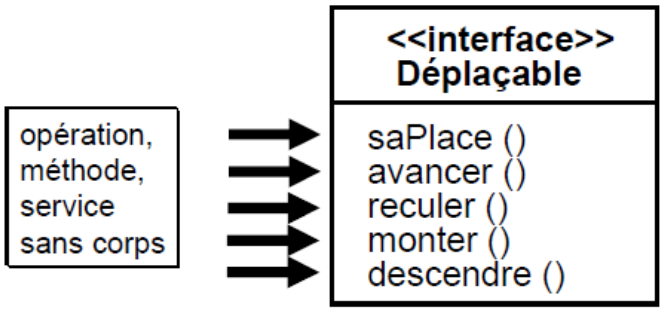
PRINCIPE

- ✓ Un objet d'une classe donnée pourra toujours faire référence à des objets de ses sous classes (polymorphisme).

✓ Une opération exécutée par un objet sera celle que connaît l'objet dont il fait référence (liaison dynamique)

5) Les Interfaces

- ✓ Une **interface** permet de décrire le **comportement** d'une entité (classe, paquetage ou composant), c'est à dire un savoir-faire sous la forme d'une **liste d'opérations**.
- ✓ Une interface ne peut donner lieu à **aucune implémentation**.
- ✓ Une interface est équivalente à une classe abstraite sans attributs où toutes les méthodes sont abstraites.
- ✓ **Une classe** peut déclarer qu'elle **implémente une interface**. Elle doit alors implémenter toutes les opérations de cette interface. Elle peut ensuite être utilisée partout où ce comportement est exigé.



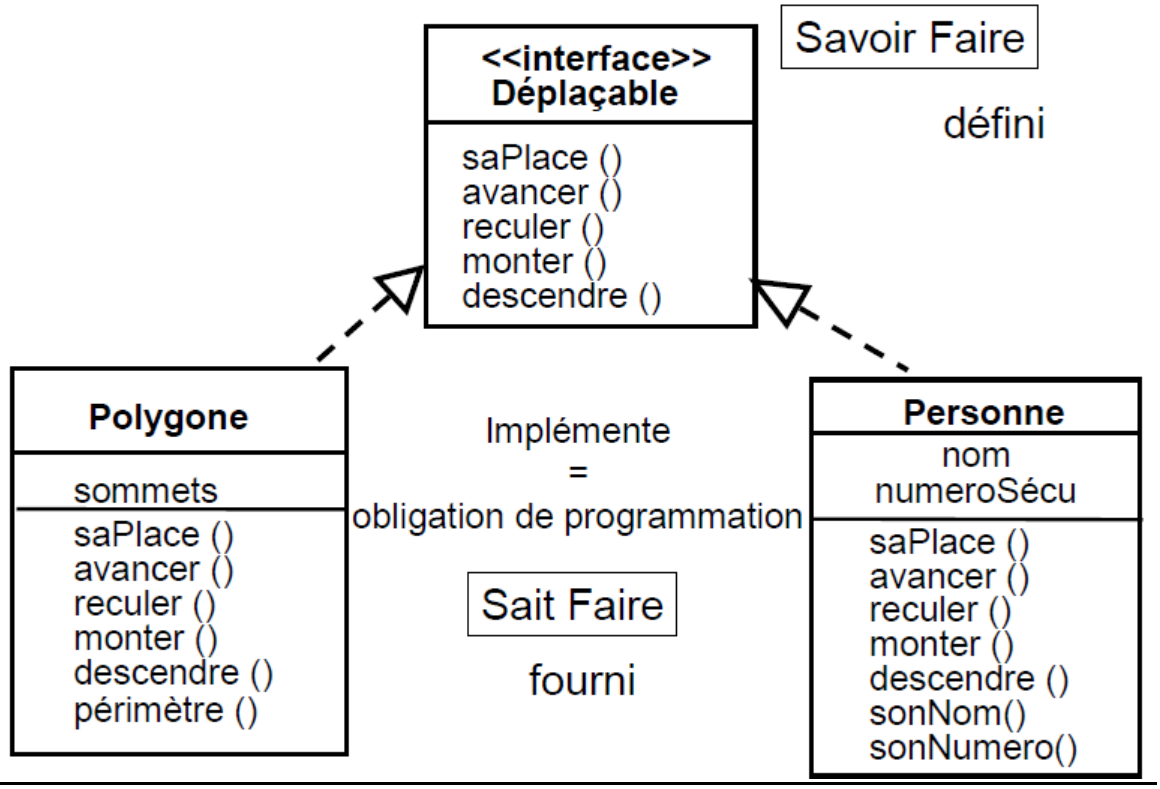
Une interface n'est PAS une classe
C'est une liste de services

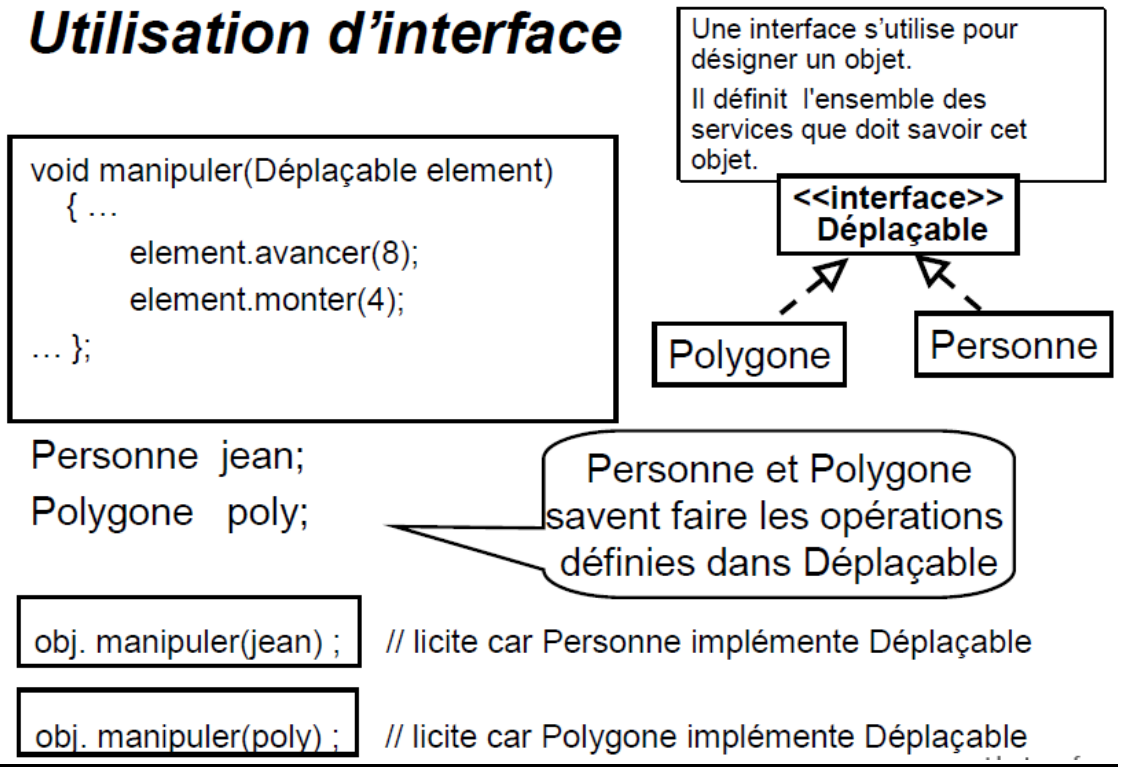
Elle ne peut pas servir à créer un objet

Une interface exprime un savoir faire

TYPE = CLASSE + INTERFACE

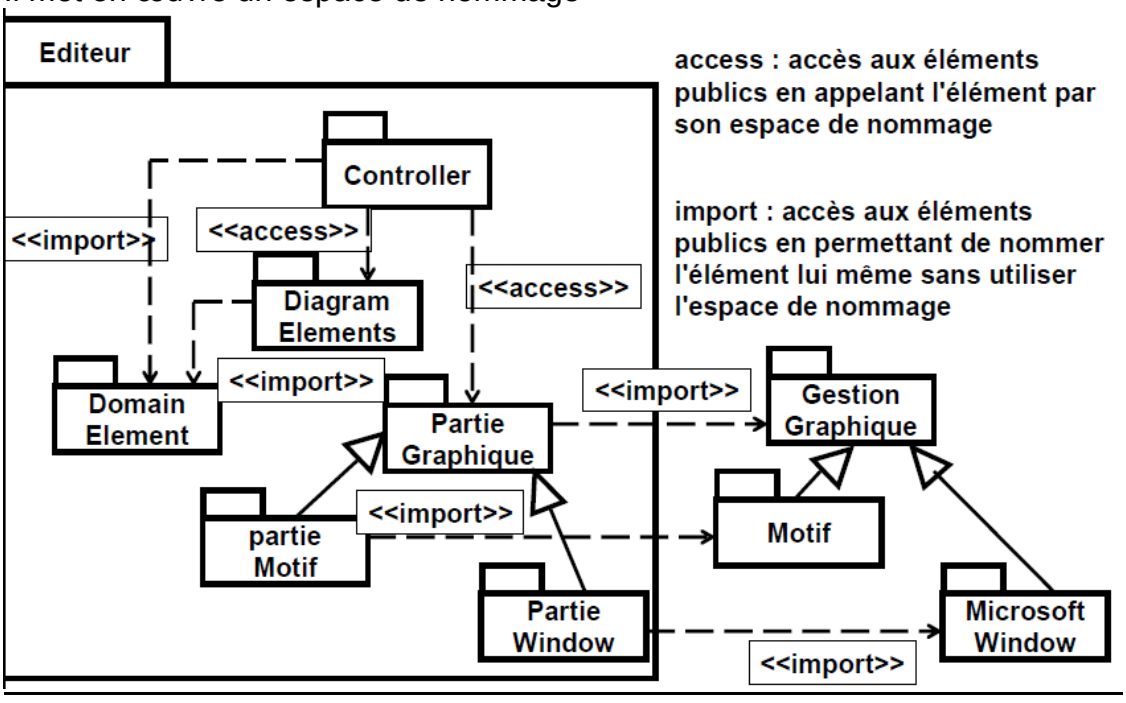
Implémentation d'interface

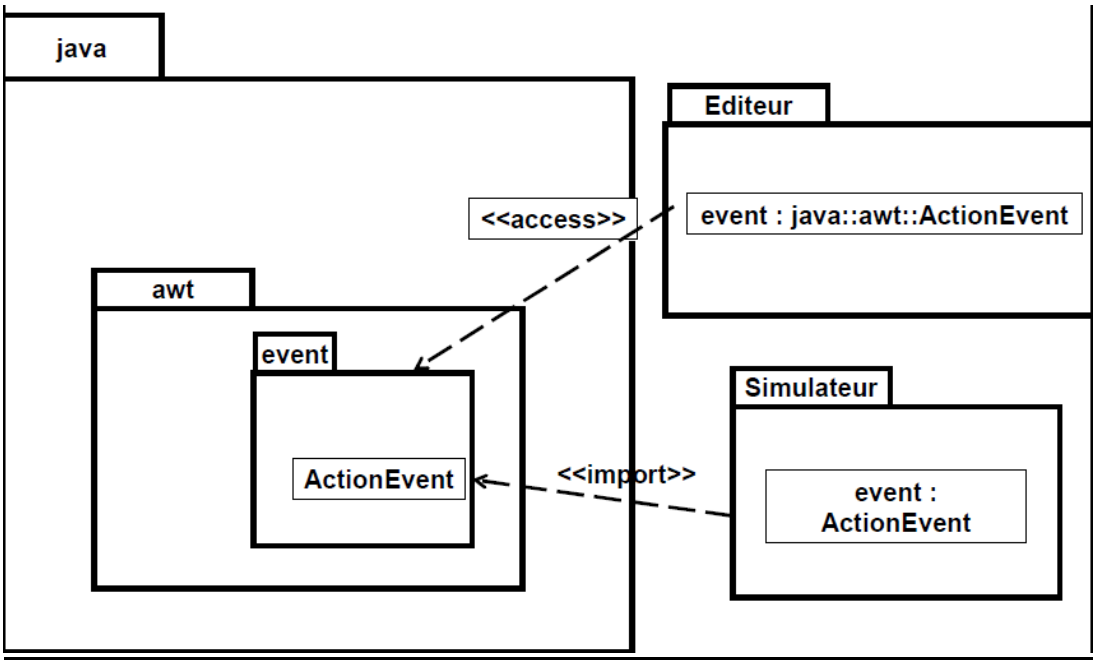




Les Paquetages

- ✓ Une application est constituée de plusieurs classes, des dizaines ou des centaines. Il est important de les organiser en groupes (en fonction de certains critères surtout logiques).
- ✓ C'est le paquetage (package) qui permet ce regroupement.
- ✓ Un paquetage regroupe des classes, des interfaces.
- ✓ Il permet d'encapsuler certains éléments de la modélisation. Un élément du paquetage peut être inaccessible de l'extérieur du paquetage, il n'est alors connu que par les éléments du même paquetage.
- ✓ Il met en œuvre un espace de nommage





Diagrammes de classes de la gestion de bibliothèque, recherche à partir du cahier des charges.

Phases de la modélisation objet :

- Identifier les classes candidates.
- Préparer le dictionnaire de données : classes retenues.
- Identifier les associations entre classes (en incluant les agrégations).
- Identifier les attributs.
- Organiser et simplifier les classes en utilisant l'héritage.
- Supprimer les associations inutiles
- Vérifier que le diagramme inclut toutes les demandes du cahier des charges.
- Itérer et affiner le modèle.
- Grouper les classes en modules (paquetages).

Identifier les classes : les classes candidates

- Un gérant de bibliothèque désire automatiser la gestion des prêts.
- Il commande un logiciel permettant aux utilisateurs de connaître les livres présents, d'en réserver jusqu'à 2. L'adhérent peut connaître la liste des livres qu'il a empruntés ou réservés.
- L'adhérent possède un mot de passe qui lui est donné à son inscription.
- L'emprunt est toujours réalisé par les employés qui travaillent à la bibliothèque. Après avoir identifié l'emprunteur, ils savent si le prêt est possible (nombre max de prêts = 5), et s'il a la priorité (il est celui qui a réservé le livre).
- Ce sont les employés qui mettent en bibliothèque les livres rendus et les nouveaux livres. Il leur est possible de connaître l'ensemble des prêts réalisés dans la bibliothèque

Gérant bibliothèque gestion prêts logiciel utilisateurs livres
adhérent liste mot de passe inscription emprunt employés emprunteur ensemble

Les classes retenues

- Gérant — non pertinente, n'intervient pas
- *bibliothèque* — *oui* responsabilité : gérer les livres, adhérents, prêts
- gestion — non vague
- *prêts* — *oui* responsabilité : contenir les infos et actions sur les prêts
- logiciel — non vague
- utilisateurs — (choix entre utilisateur, adhérent, emprunteur)
- *livres* — *oui* responsabilité : permettre de connaître son état
- *adhérent* — *oui* responsabilité : permettre à la personne d'être identifiée
- liste — non implémentation ou conception
- mot de passe — non attribut
- Inscription — non action
- emprunt — non action
- *employés* — *oui* responsabilité : reconnaître qui a fait un prêt, etc..
- emprunteur — (choix entre utilisateur, adhérent, emprunteur)
- Ensemble — non implémentation ou conception

Dictionnaire des données

- bibliothèque : organisme gérant une collection de livres qui peuvent être empruntés par ses adhérents. Une bibliothèque est gérée par ses employés.
- prêt : un prêt est caractérisé par le numéro du livre, la date, la durée. Il ne peut être fait que par un adhérent.
- livre ouvrage pouvant être emprunté.
- adhérent personne inscrite à la bibliothèque.
- employé personne travaillant à la bibliothèque.

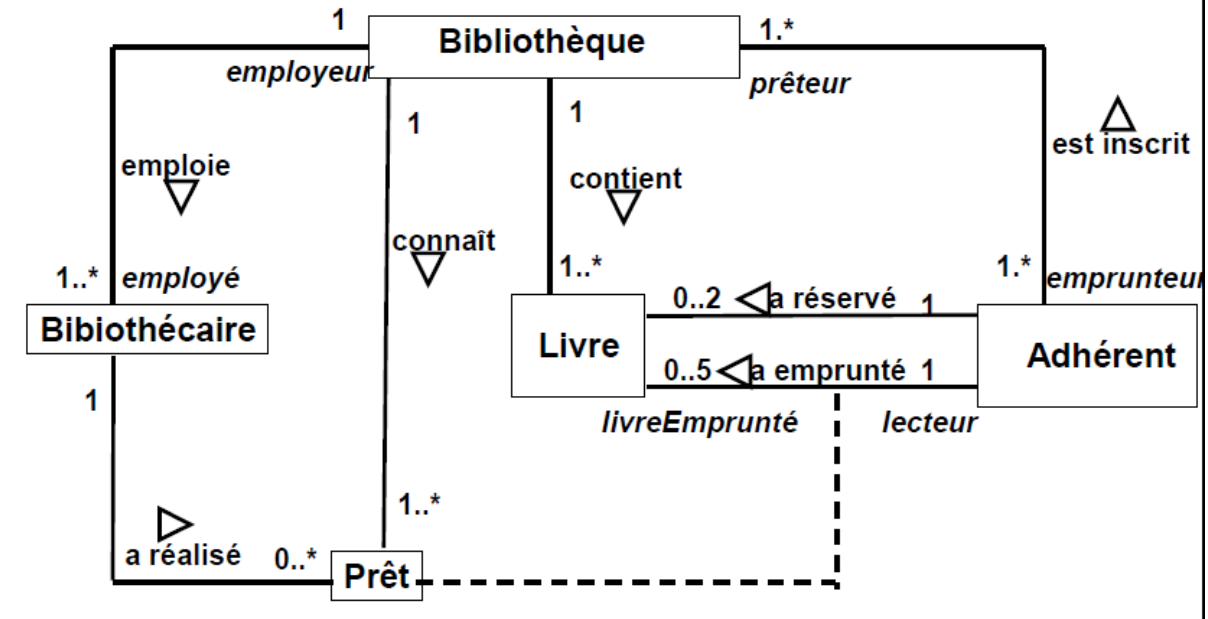
Chercher les associations

- Un gérant de bibliothèque désire automatiser la gestion des prêts.
- Il commande un logiciel permettant aux utilisateurs de *connaître* les livres présents, d'en *réserver* jusqu'à 2. L'adhérent peut connaître la liste des livres qu'il *a empruntés* ou *réservés*.
- L'adhérent possède un mot de passe qui lui est donné à son inscription.
- L'emprunt *est* toujours *réalisé* par les employés qui *travaillent* à la bibliothèque. Après avoir identifié l'emprunteur, ils savent si le prêt est possible (nombre max de prêts = 5), et s'il a la priorité (il est celui qui a réservé le livre).
- Ce sont les employés qui mettent en bibliothèque les livres rendus et les nouveaux livres. Il leur est possible de connaître l'ensemble des prêts *réalisés* dans la bibliothèque

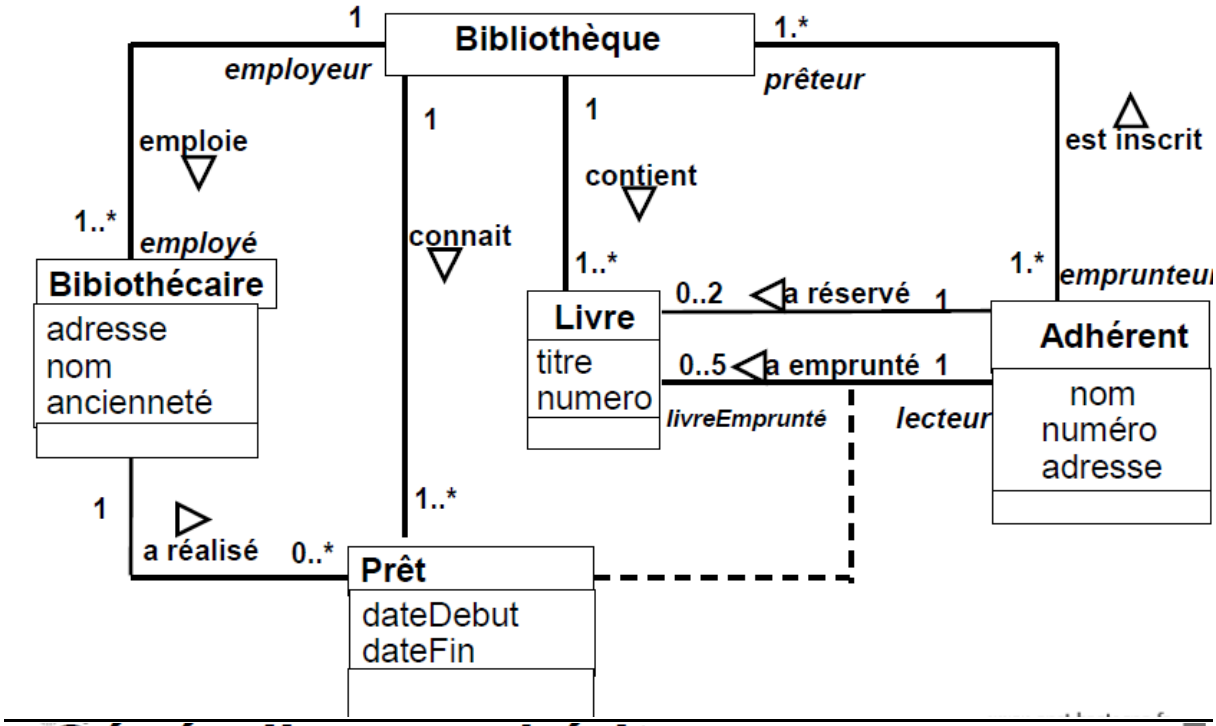
Associations sous entendues

Une adhérent *est inscrit* à la bibliothèque.
La bibliothèque *contient* des livres

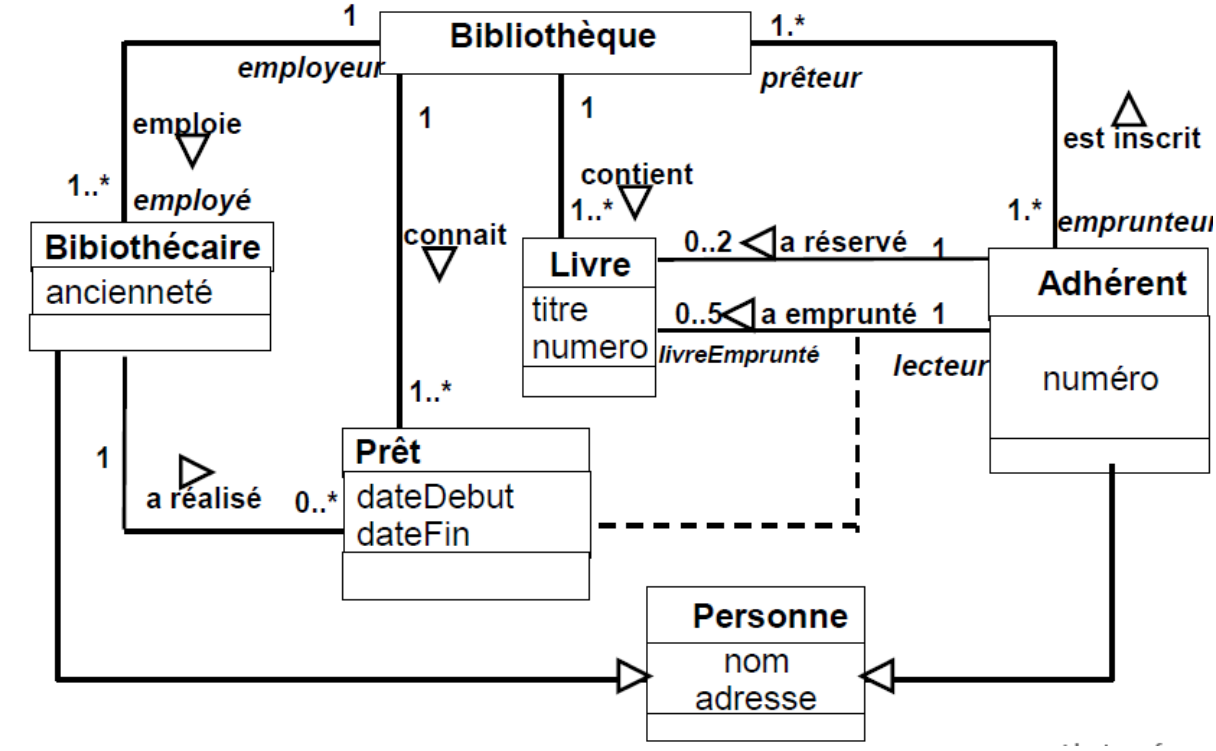
Les associations

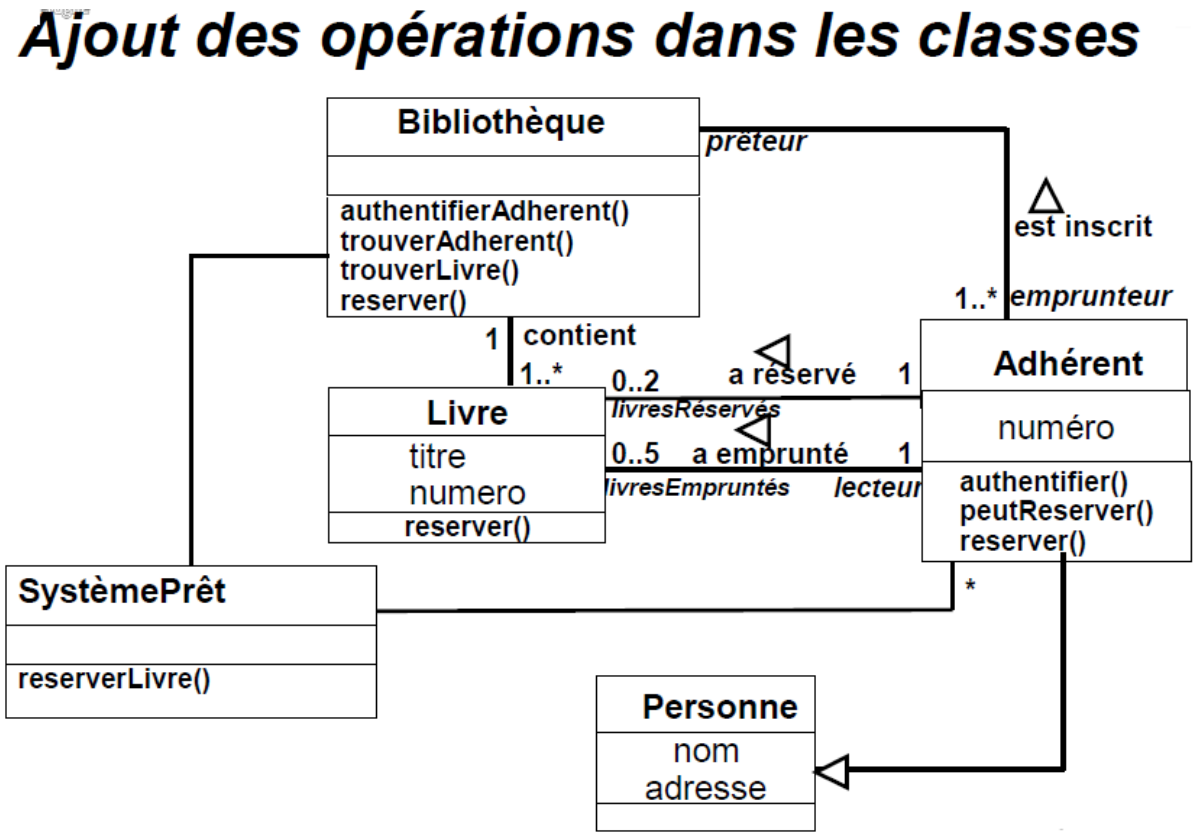


Chercher les attributs



Généraliser par héritage





Conclusion

- ✓ Méthode pour appréhender la réalisation d'un système informatique
- ✓ Diagrammes qui permettent d'aider à la réflexion, de permettre la discussion (clients, développeurs)
- ✓ Pas de solution unique mais un ensemble de solutions plus ou moins acceptables suivant les contraintes du client et les logiciels et matériels disponibles

La Partie 3 présentera les diagrammes suivants :

- ✓ Diagramme d'états transitions
- ✓ Diagramme de composants
- ✓ Diagramme de déploiement
- ✓ Diagramme d'activités