
Cours 5 – Analyse

MODULE INTRODUCTION AU GÉNIE LOGICIEL

Objectifs du Cours

Appréhender
l'activité d'analyse

Se familiariser avec
les pratiques et les
livrables de
l'analyse

Utiliser UML pour
exprimer les
résultats de
l'analyse

Découverte des
diagrammes de
classe, d'activité, de
séquence et d'état

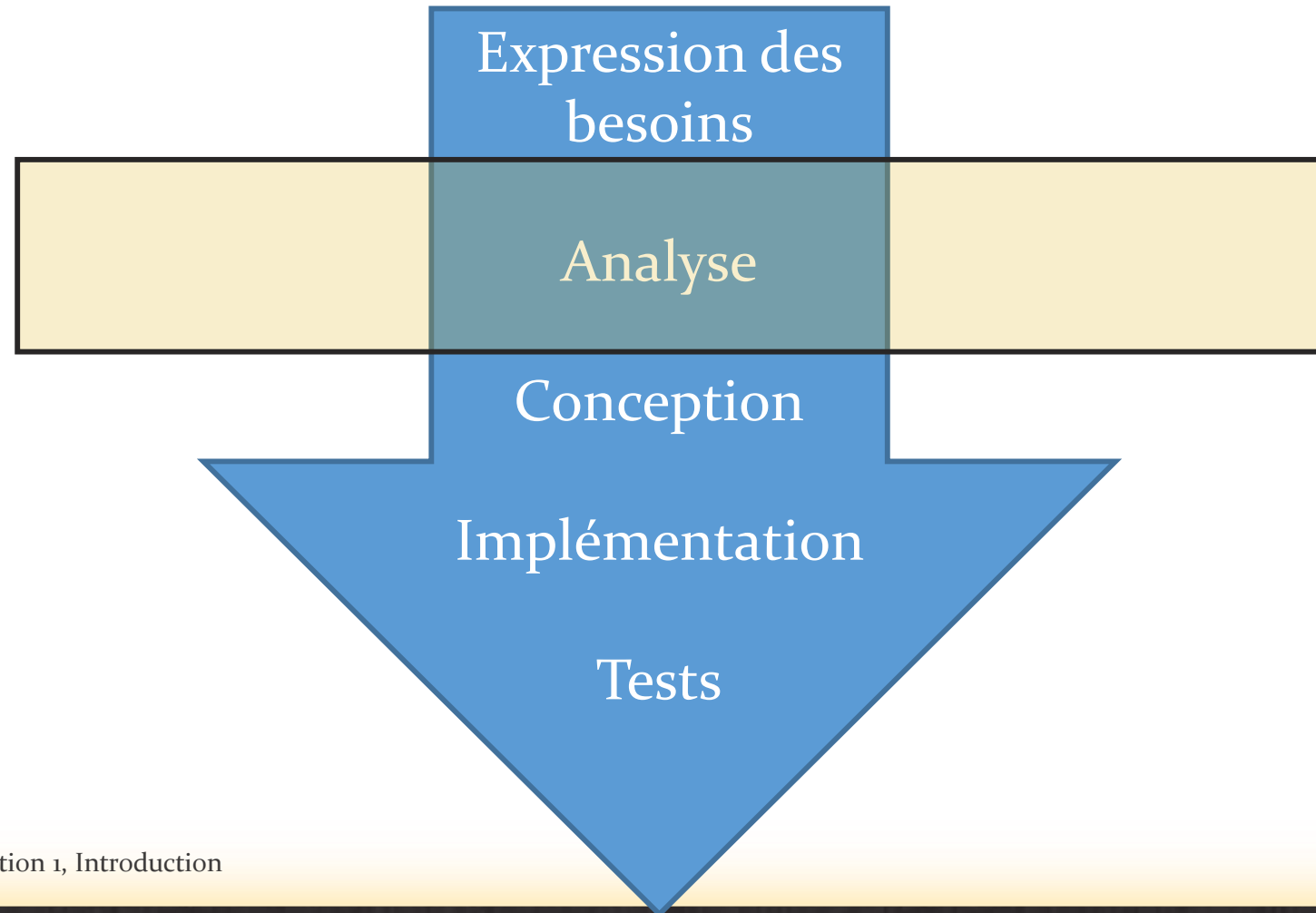
Plan du Cours



Introduction

SECTION 1

Cycle de Vie



L'activité d'analyse

L'analyse et l'expression de besoins sont très dépendants

Il y a une ambiguïté entre l'analyse et l'expression de besoins

L'analyse permet de clarifier les besoins d'une manière détaillée

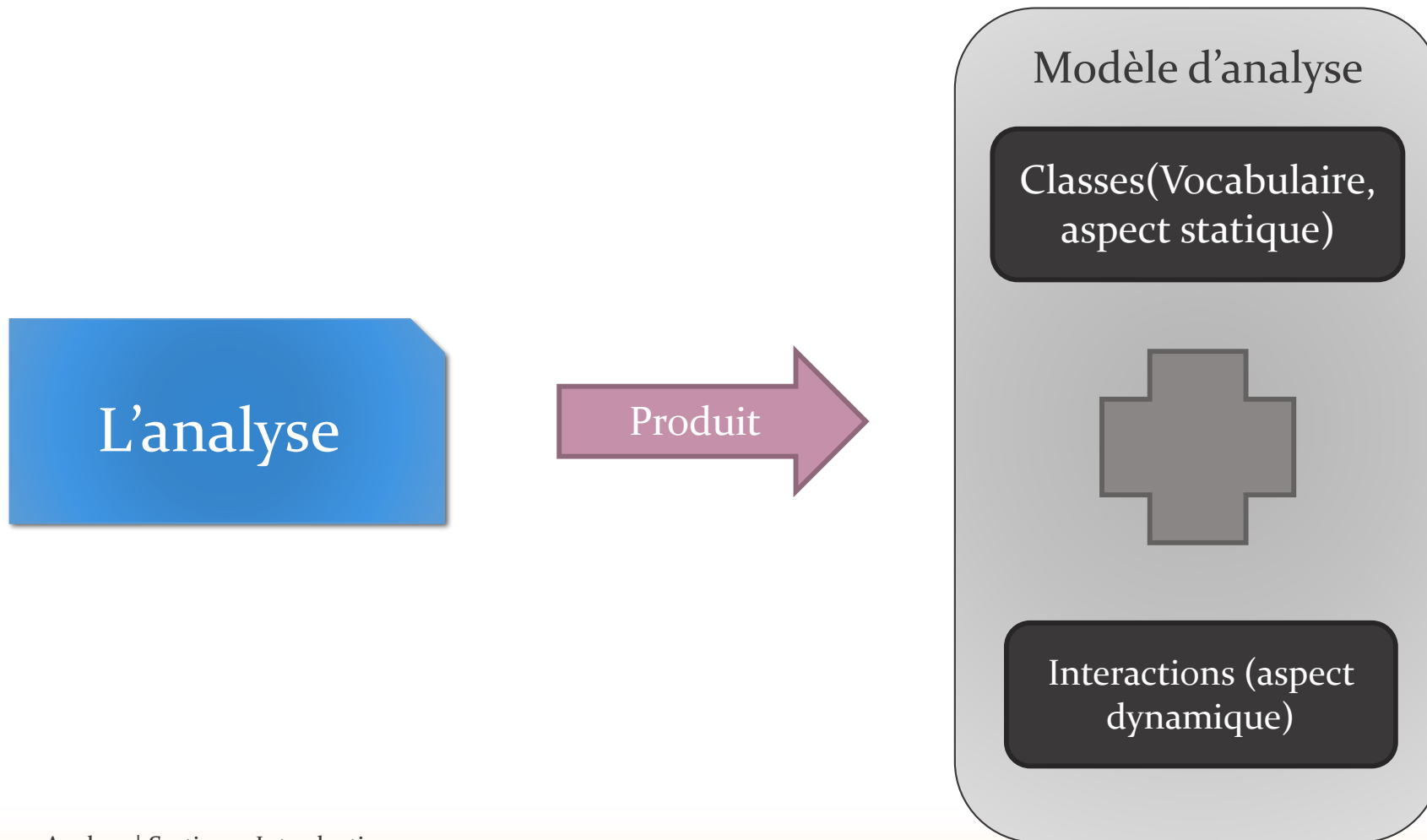
Il y a une ambiguïté entre l'analyse et la conception

L'analyse et la conception répondent à la question «comment »

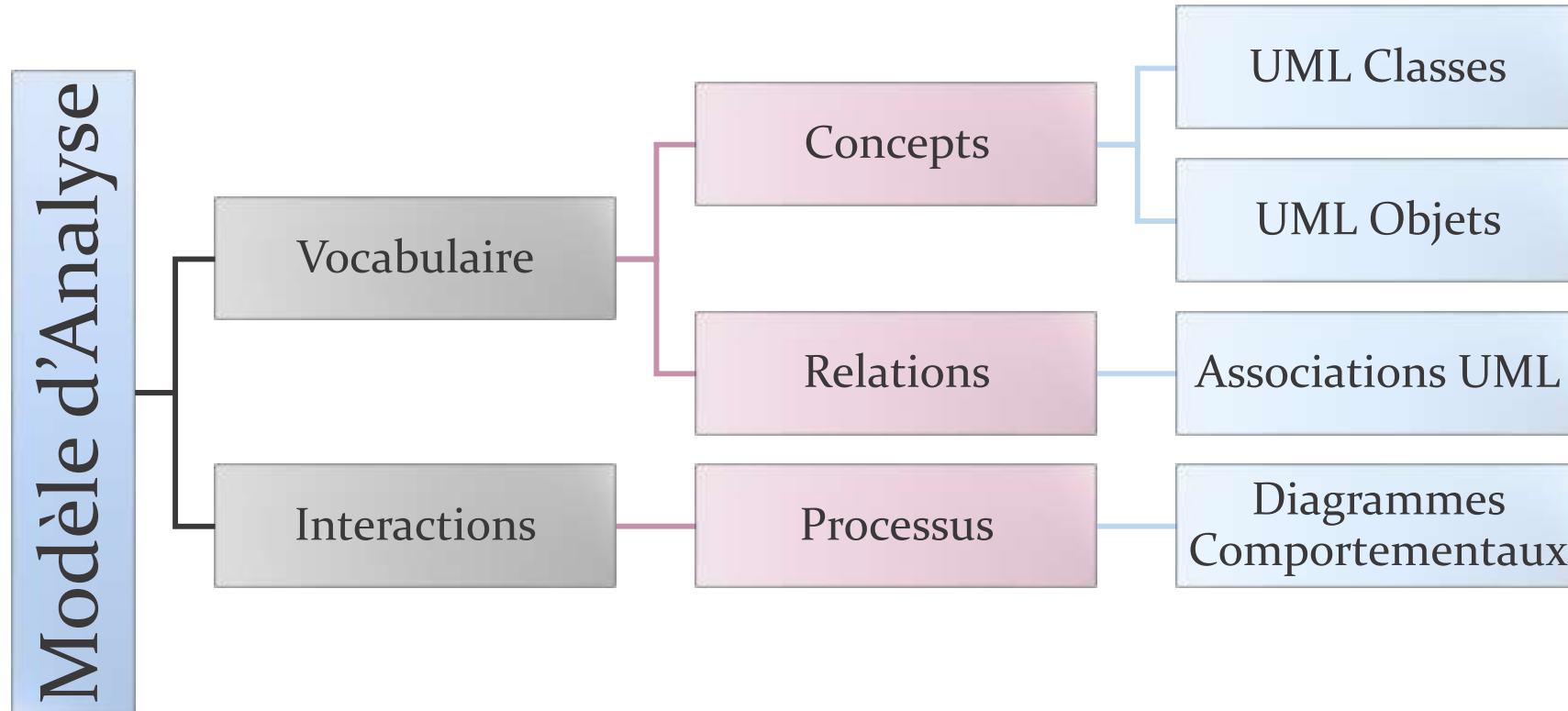
L'analyse se focalise sur l'aspect métier des fonctionnalités tandis que la conception se focalise sur l'aspect technique

Dans UP, le gros de l'analyse se fait durant la phase d'analyse de besoins et d'élaboration

L'activité d'analyse



Modèle d'Analyse



L'activité d'Analyse - Exemple

« à l'ESI, pour effectuer un stage, l'étudiant doit chercher un encadreur interne ou dans une entreprise externe et un sujet à réaliser. Une fois le sujet trouvé, l'étudiant le dépose au niveau du service des stages. Ce dernier s'occupe de la validation du sujet en faisant appel à un enseignant de l'ESI ».

L'activité d'Analyse - Exemple

- Le vocabulaire devra contenir les concepts suivants : enseignant, étudiant, encadreur, entreprise, sujet, service de stage et stage
- Il y a plusieurs interactions à souligner : l'interaction de recherche de sujet qui implique l'étudiant, l'entreprise, le sujet et l'encadreur. L'interaction de validation qui implique l'étudiant, le service de stages et l'enseignant.

Construire le modèle d'analyse - Règles

Se limiter aux concepts métier, s'éloigner des considérations techniques

Le langage du modèle d'analyse utilise le même langage que le métier

Le modèle capture une vision globale sur un concept ou un processus, ne pas aller trop dans le détail

Le modèle doit toujours être compréhensible et utile au client

Plus le modèle est simple, plus il est meilleur

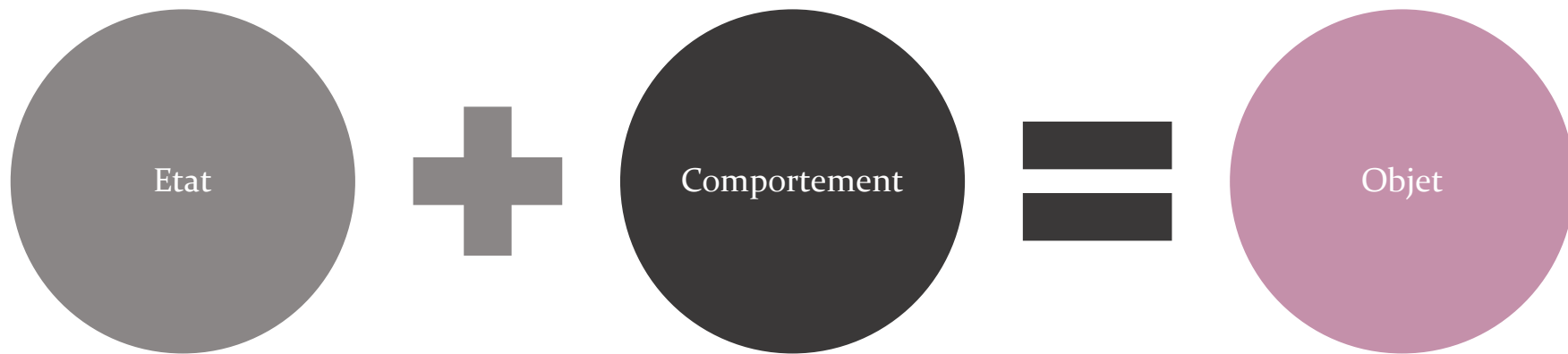
Introduction

SECTION 1 – DÉBAT (10 MNS)

Objets et Classes

SECTION 2

L'objet



Définition d'un Objet

- Rumbaugh définit l'objet comme étant une entité discrète ayant une limite bien définie qui possède un état et un comportement
- Un objet représente une entité du monde réel
- L'état de l'objet est l'ensemble des *valeurs de ses attributs*
- Le comportement d'un objet est représenté par les *opérations* qu'il peut effectuer. Souvent les opérations conduisent à un changement de l'état d'un objet
- L'objet a un *identifiant unique* qui permet de le distinguer des autres objets.

Objet - Exemple



Etat	Comportement
<ul style="list-style-type: none">• Numéro de série (Identifiant)• Marque• Modèle• Allumée• Mode (Photo / Vidéo)• Connectée à un ordinateur• Liste des photos en mémoire• Capacité• Photo en cours	<ul style="list-style-type: none">• Allumer()• Eteindre()• Connecter()• Filmer()• PrendreUnePhoto()

Notation UML des objets

compteTest: Compte

Cle = 80

Numero = 11256

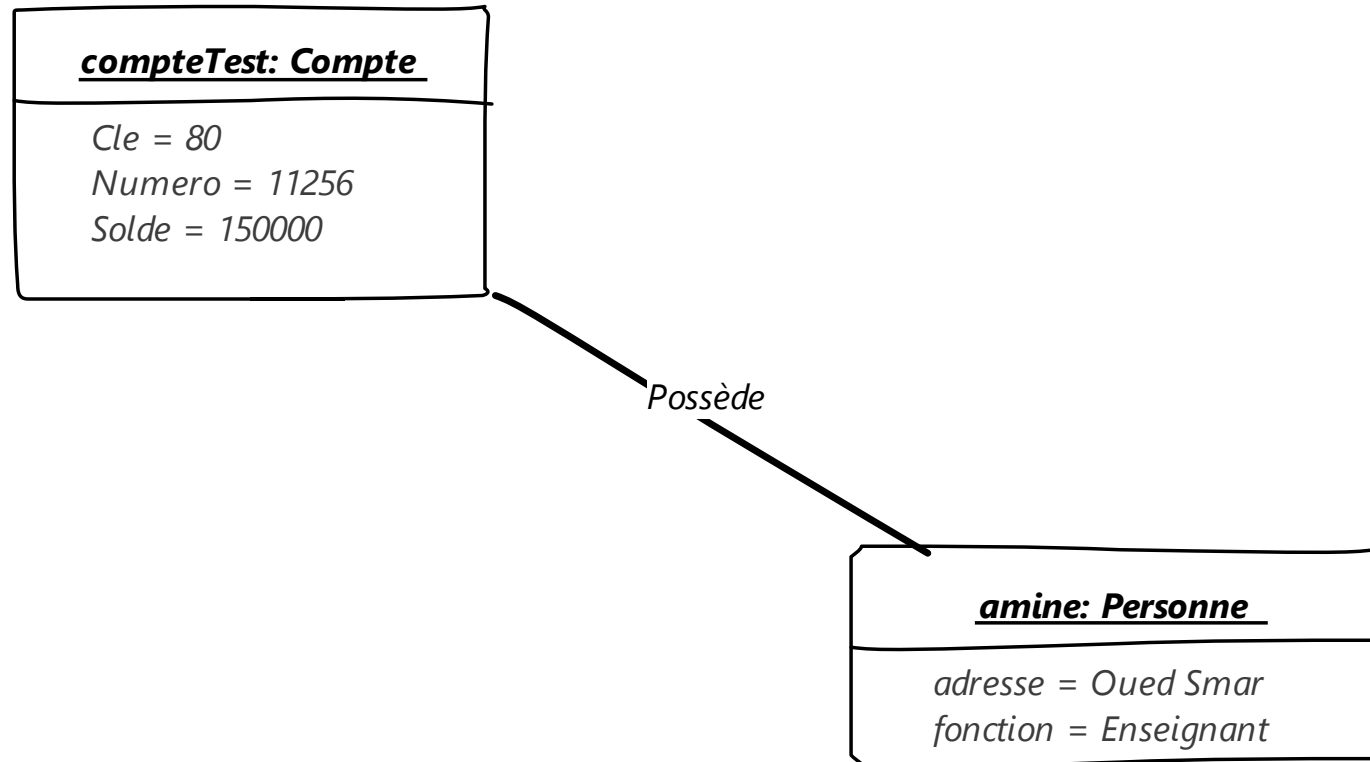
Proprietaire = [Mokhtar]

Solde = 150000

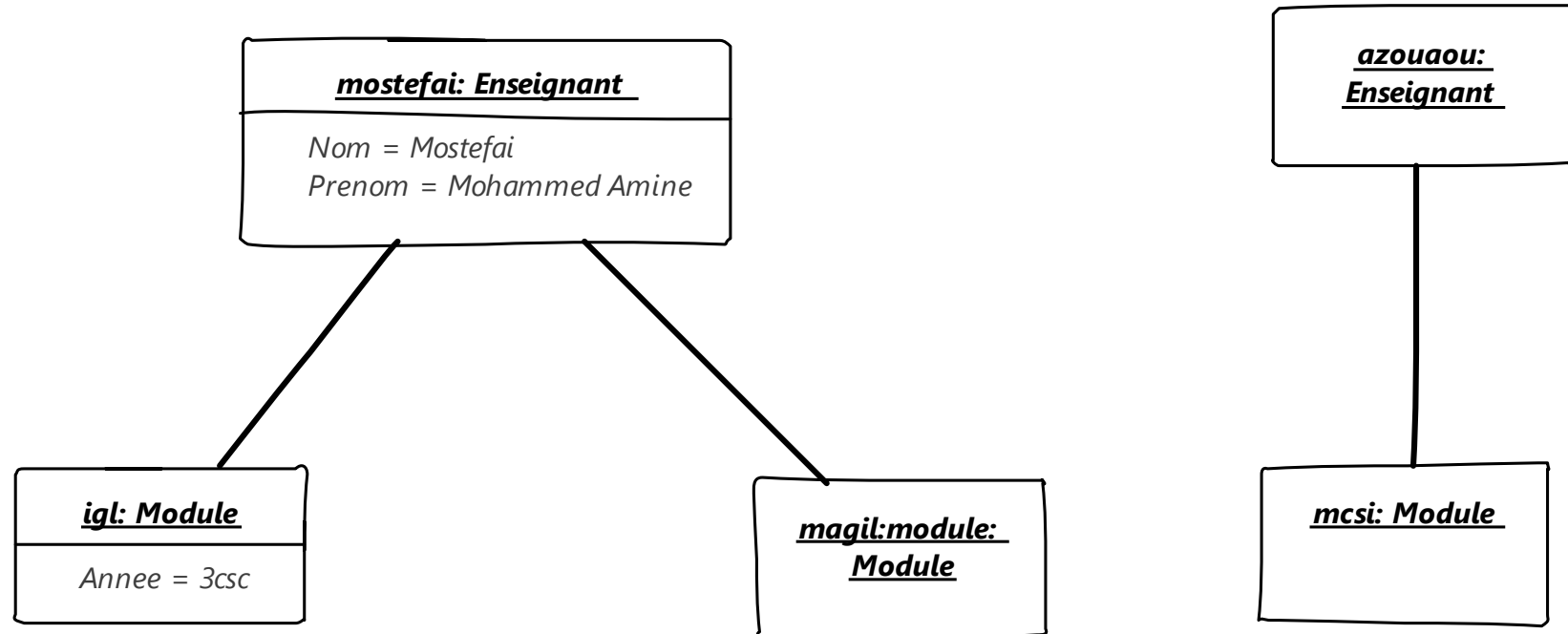
Notation UML des objets

- Les noms sont écrits en souligné
- Les noms des objets commencent par une minuscule. Si c'est un nom composé, le début de chaque mot suivant commence par une majuscule. Par exemple : compte11256:Compte ou clientFavori:Client. Le symbole « : » sépare le nom de l'instance du nom de sa classe.
- Le nom de l'objet peut être anonyme (ne comporte que le nom de la classe. Par exemple : :Compte ou :Client.
- L'objet peut ne pas avoir de classe : par exemple amine ou compteTest.
- Deux objets peuvent être associés via un lien

Exemple de Lien



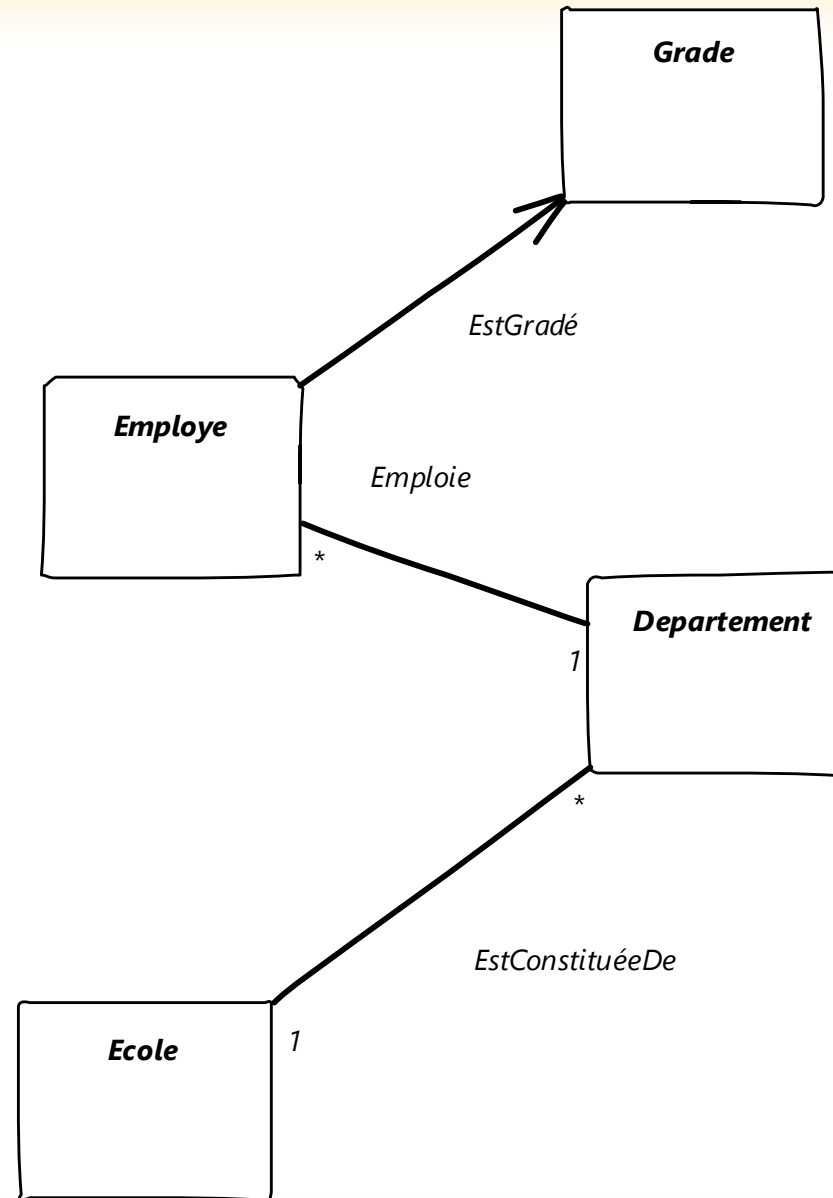
Exemple



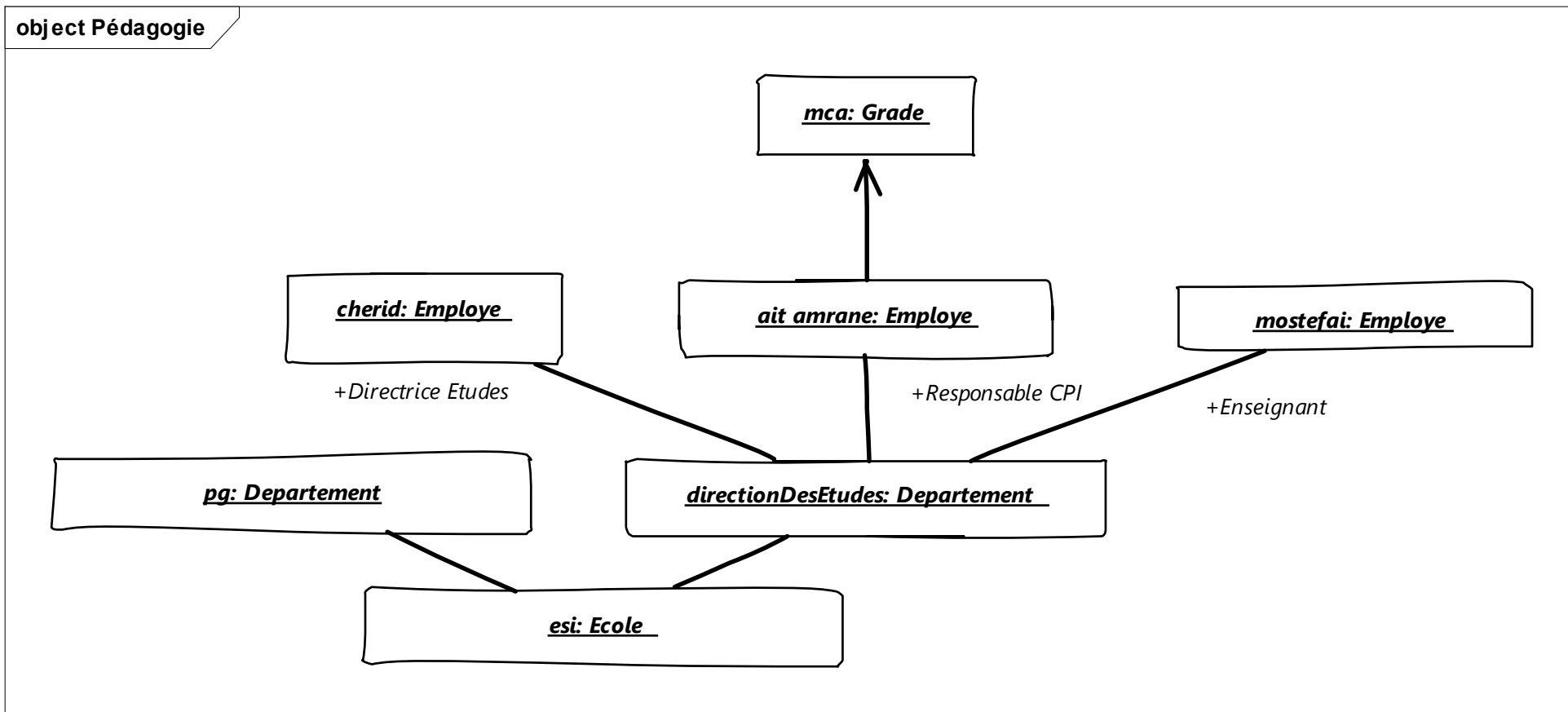
Objets – Notation UML - Diagrammes

- Un diagramme d'objets présente des **objets** et leurs **relations** à un instant T
- Le diagramme d'objets est idéal pour donner des **exemples** sur des situations particulières
- Les liens **bidirectionnels** entre A B quand A et B s'envoient des messages
- Un lien **unidirectionnel** entre A vers B indique que A peut envoyer un message vers B mais pas l'inverse

Exemple



Exemple – Suite



Section 2 – objets et classes

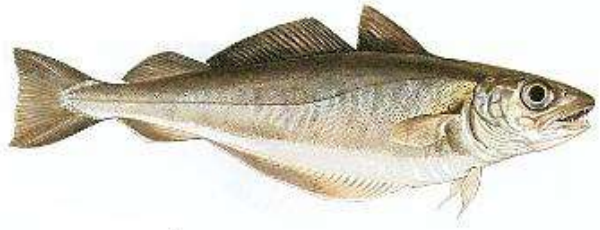
- Rumbaugh définit la classe comme étant un descripteur d'un ensemble d'objets qui partagent les mêmes attributs, méthodes, relations et comportement
- La classe est le modèle d'un ensemble d'objets similaires
- Dans l'exemple précédent : Mostefai et Azouaou sont deux instances de la classe « Enseignant »
- Un objet appartient à une seule classe
- La classe définit la structure d'un objet (aspect statique) , son comportement (aspect dynamique) et ses relations

Les classes

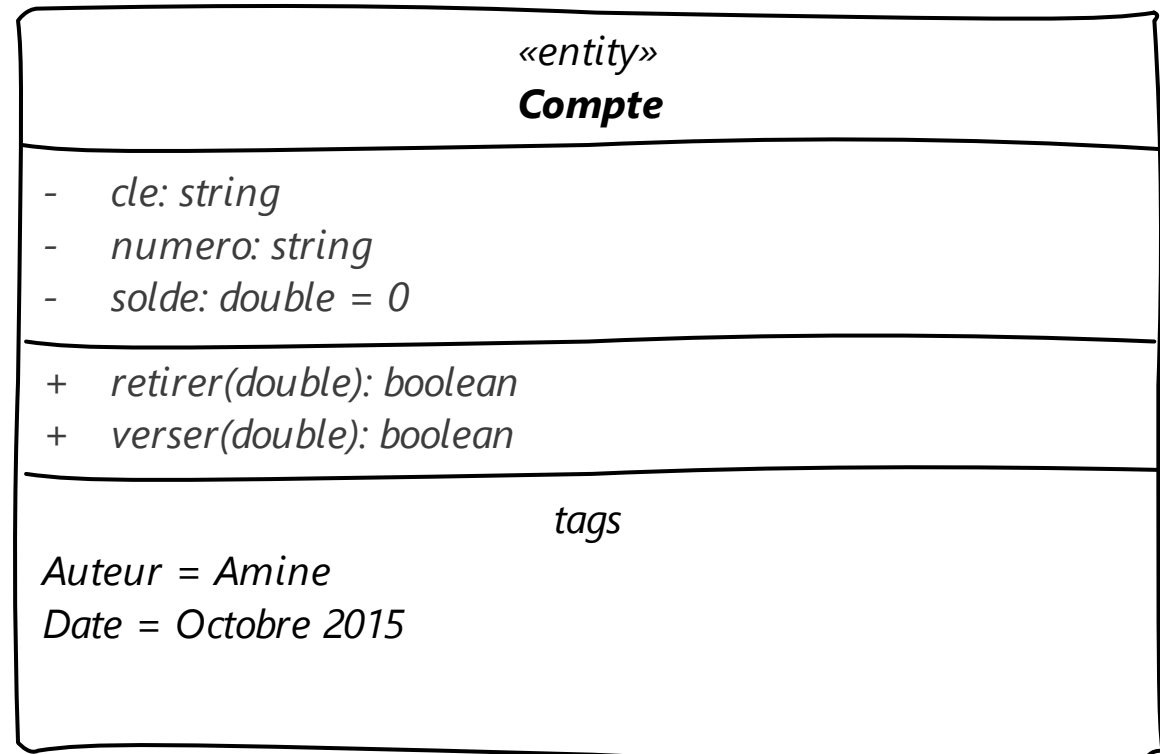
- Rumbaugh définit la classe comme étant un descripteur d'un ensemble d'objets qui partagent les mêmes attributs, méthodes, relations et comportement
- La classe est le modèle d'un ensemble d'objets similaires
- Dans l'exemple précédent : Mostefai et Azouaou sont deux instances de la classe « Enseignant »
- Un objet appartient à une seule classe
- La classe définit la structure d'un objet (aspect statique) , son comportement (aspect dynamique) et ses relations

Modélisation orientée objet

Comment classer ces animaux ?



Classes – Notation UML



Classes – Notation UML - Nom

- De préférence, utiliser la convention **UpperCamelCase**. Le nom de classe est en minuscules et la première lettre en majuscules. Si le nom de la classe est composite, le nom de chaque mot composant est en minuscule et la première lettre en majuscule. Par exemple : *Agent*, *Compte*, *LigneFacture*, *MandatPostalValide*.
- Eviter les abbréviations : par exemple utiliser *FactureValideDetaillee* au lieu de *FactureVD*.
- Ne pas utiliser des noms verbaux car les classes représentent des « choses ».

Classes – Notation UML - Visibilité

- La visibilité s'applique aussi bien aux attributs qu'aux opérations
- Durant la phase d'analyse, *la visibilité n'est pas importante*
- Les langages de programmation peuvent avoir une interprétation *différente* de la visibilité

Classes – Notation UML - Visibilité

Visibilité	Nom	Description
+	Visibilité publique	Accès depuis la même classe et à l'extérieur de la classe
-	Visibilité privée	Accès depuis la même classe uniquement
#	Visibilité protégée	Accès depuis la même classe et les classes descendantes
~	Visibilité paquet	Accès depuis la même classe et toutes les classes appartenant au même paquet

Visibilité - Exemple

Classes – Notation UML - Attributs

- Le type peut être un type simple défini par UML ou l'un des types suivants : *boolean, byte, char, double, float, int, long, short*
- *Le type peut être complexe (une autre classe)*
- La valeur initiale indique la valeur de l'attribut quand une instance de la classe est créée
- Un attribut est dit statique s'il appartient à la classe, pas à une instance en particulier. Les attributs statiques sont noté en souligné.
- Syntaxe : *Visibilité Nom_Attribut : type [multiplicité] = valeur_initiale*

Classes – Notation UML – Attributs, multiplicité

- La multiplicité indique les attributs multiples.
- Les multiplicités sont équivalentes aux tableaux mais ont plus de sémantique.
- Exemple 1 : `int valeurs[7]`. La classe contient ***exactement*** 7 valeurs.
- Exemple 2 : `int valeurs[2..*]` : la classe contient ***au moins*** 2 valeurs.
- Exemple 3 : `int valeurs[2..7]` la classe contient ***au minimum*** 2 valeurs et ***au maximum*** 7 valeurs.
- Exemple 4 : `int valeurs [0..1]` la classe contient ***une*** valeur ou ***null***.

Exemples d'attributs

Etudiant
<ul style="list-style-type: none">+ matricule : int# nom : string# prenom : string~modules : int [2..9]- age :int = 18- chambreAffectee : Chambre [0..1]+ tuteursDeSuivi : Tuteur [2]+ <u>nombreEtudiants</u> : int+ inscriptions : Inscription [1..*]

Opérations

- **Syntaxe** : *Visibilité Nom_Operation (direction nom_paramètre: type = valeur_défaut,...) : type_retour*
- Les opérations sont nommées en utilisant une formulation *lowerCamelCase*
- De préférence, utiliser des *expressions verbales* pour les opérations
- Les paramètres sont nommés en utilisant *lowerCamelCase*
- *Ne pas utiliser des expressions verbales* pour les paramètres
- Des valeurs par défaut peuvent être renseignées pour une opération.
- Les opérations statiques sont *soulignées*

Opérations, direction des paramètres

Direction	Description
in p	Le paramètre p est un paramètre d'entrée. p est utilisé par l'opération mais n'est pas modifié par l'opération.
inout p	Le paramètre p est un paramètre d'entrée / sortie. p est utilisé par l'opération et sa valeur peut être changée par l'opération.
out p	Le paramètre p est un paramètre de sortie. La valeur de p pourrait être modifiée par l'opération.
return p	Le paramètre p est un paramètre de retour. L'opération doit retourner la valeur de p.

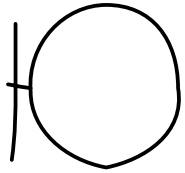
Stéréotype

- Les stéréotypes représentent un mécanisme pouvant étendre UML
- Un stéréotype peut cibler une classe, un attribut ou une opération
- Plusieurs dizaines de stéréotypes sont utilisés mais les plus connus sont : *actor*, *boundary*, *entity* et *control*.
- Le stéréotype « acteur » représente un acteur
- Le stéréotype « entity » représente une entité. Une entité est un concept métier, par exemple « Compte », « Client », « Fournisseur »,...

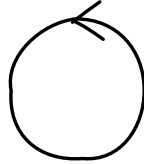
Stéréotypes d'Analyse UP

- Le stéréotype « control » représente un contrôleur. Un contrôleur est un intermédiaire entre les limites et les entités. Le contrôleur se charge de l'exécution des commandes provenant de la limite.
- Le stéréotype « boundary » représente une limite du système qui s'interface avec l'acteur. Par exemple, interface utilisateur.

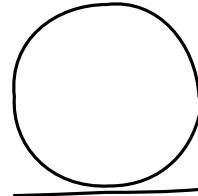
Stéréotypes UP



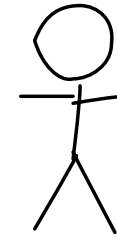
InterfaceWeb



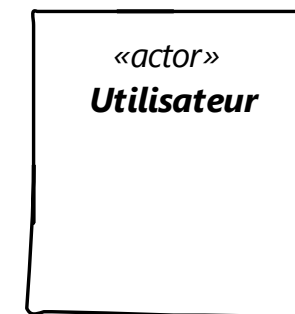
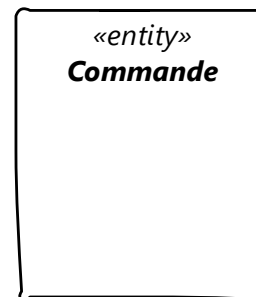
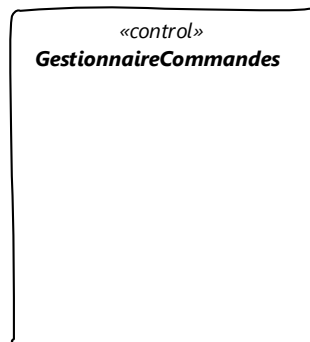
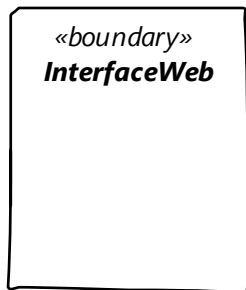
GestionnaireCommandes



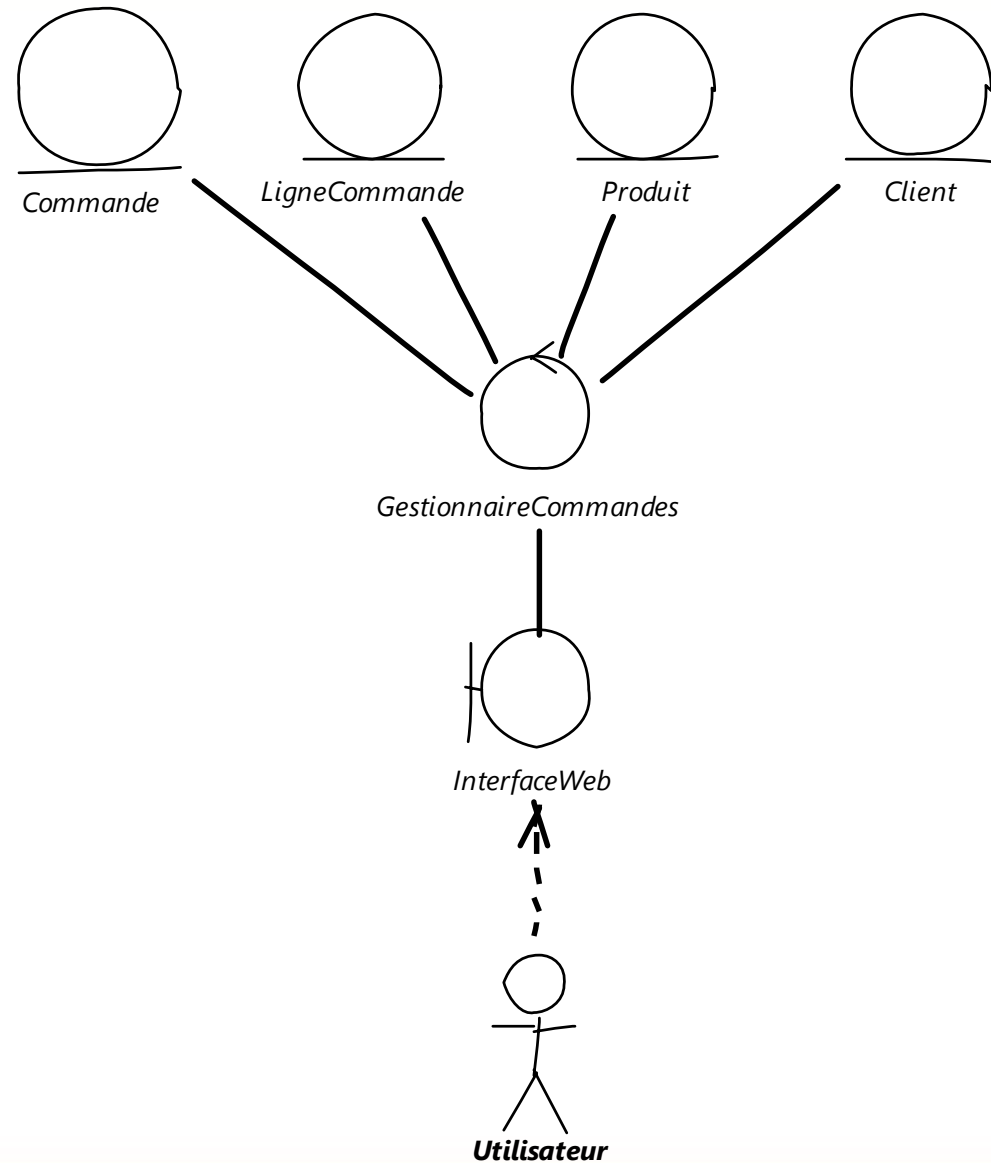
Commande



Utilisateur



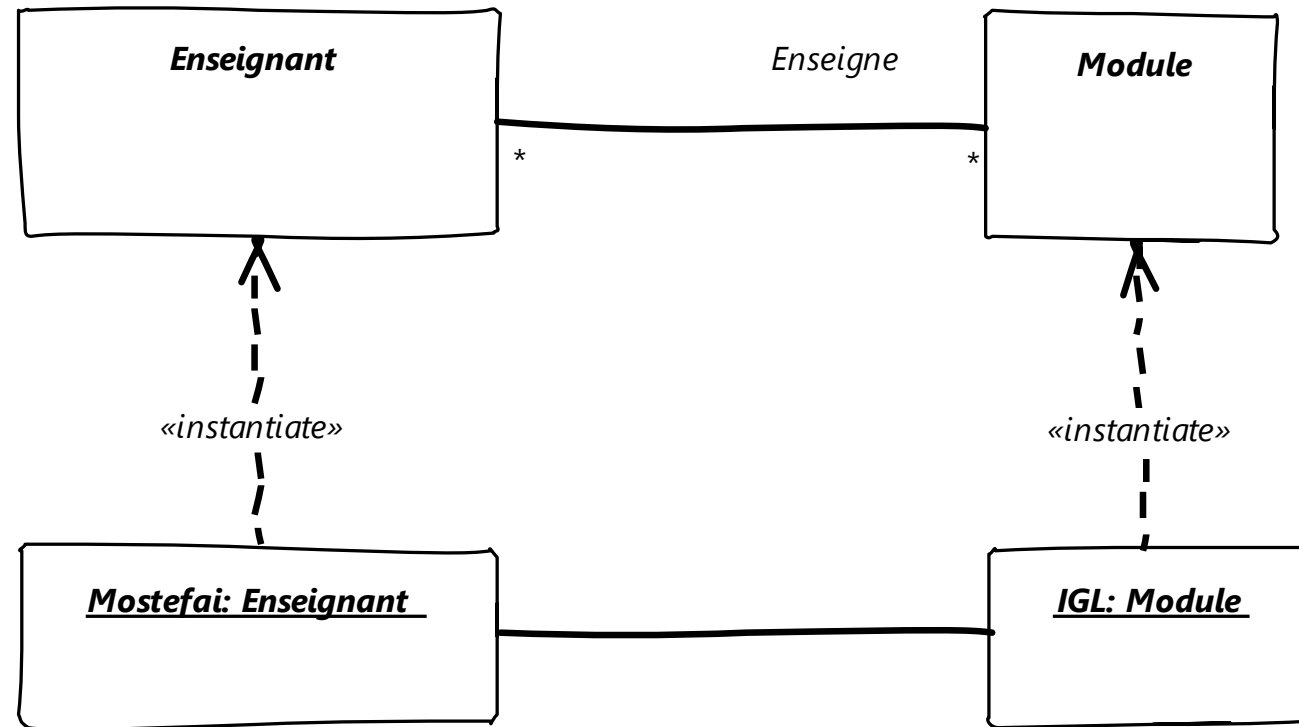
Exemple du monde réel



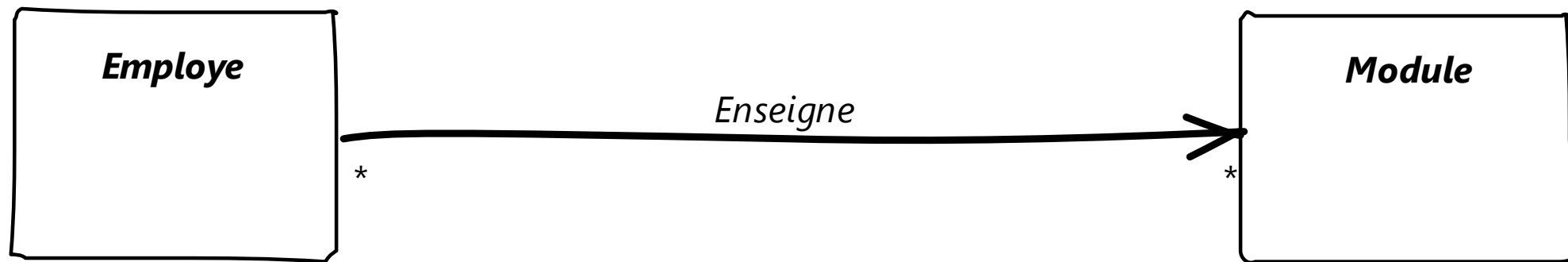
Association

- Une association est une *relation* entre deux classes
- Une *association* entre deux classes se traduit par un *lien* entre deux instances de ces classes
- Une association décrit une *relation* qui a une *sémantique* entre les deux classes
- Une association peut être entre une classe et elle-même. L'association est dite *réflexive*.
- La direction détermine la *navigation* d'un objet vers un autre.

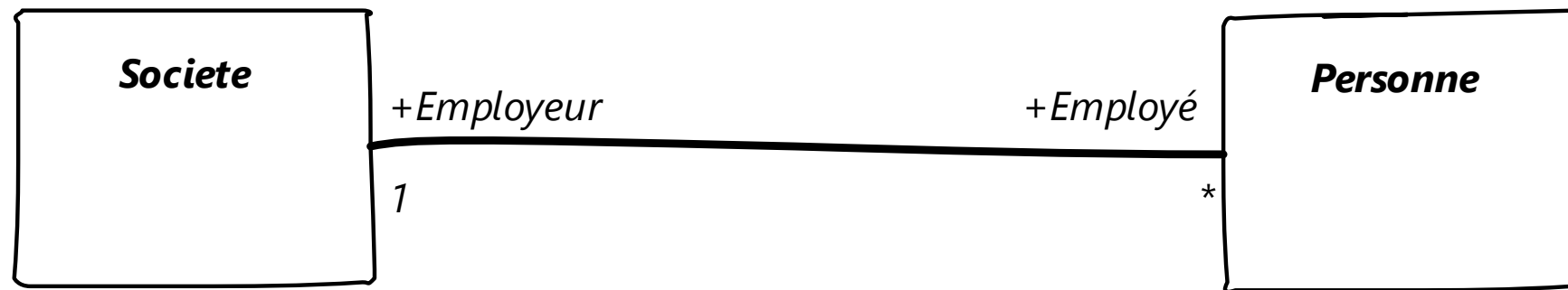
Associations entre Objets et Classes



Associations



Associations – Suite



Nom des Associations

Utiliser des *phrases verbales pour les noms et nominales pour les rôles*

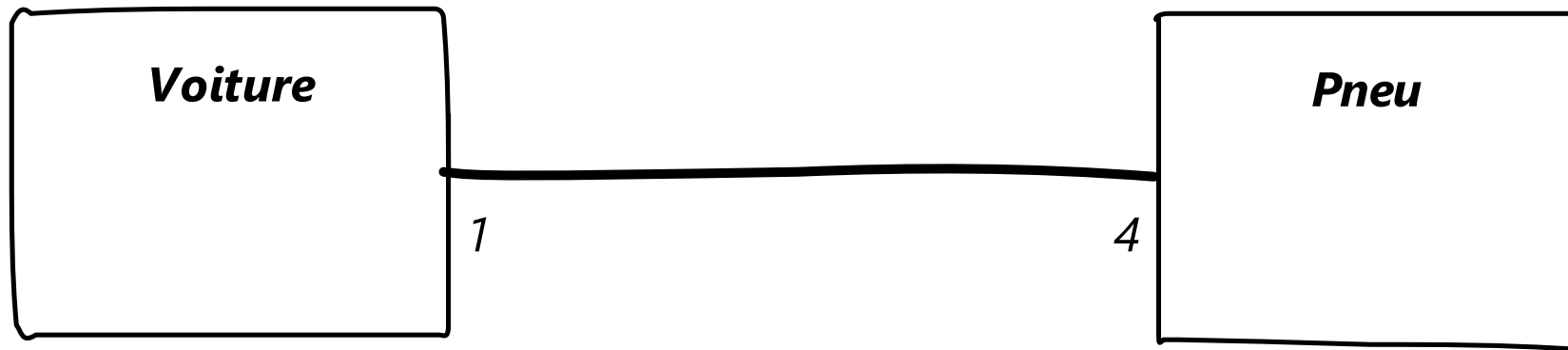
Utiliser soit le nom
soit le rôle.

Donner des noms
explicites et lisibles.

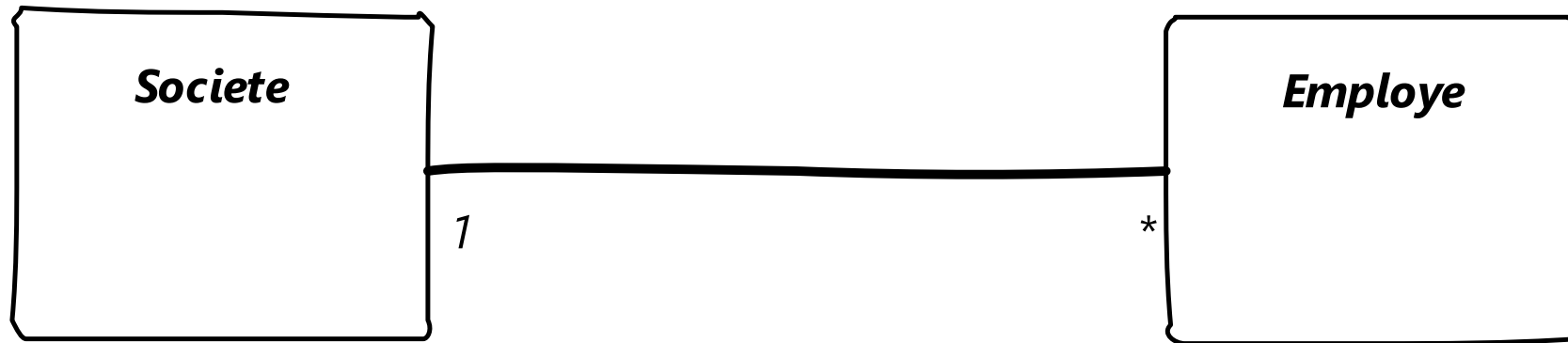
Multiplicités

Multiplicité	Signification
0..1	Zéro ou 1
1	Exactement 1
0..*	Zéro ou plusieurs
*	Zéro ou plusieurs
1..*	1 ou plusieurs
1..9	1 à 9
9	Exactement 9
1..5, 8, 20..*	1 à 5, exactement 8 ou plus de 20

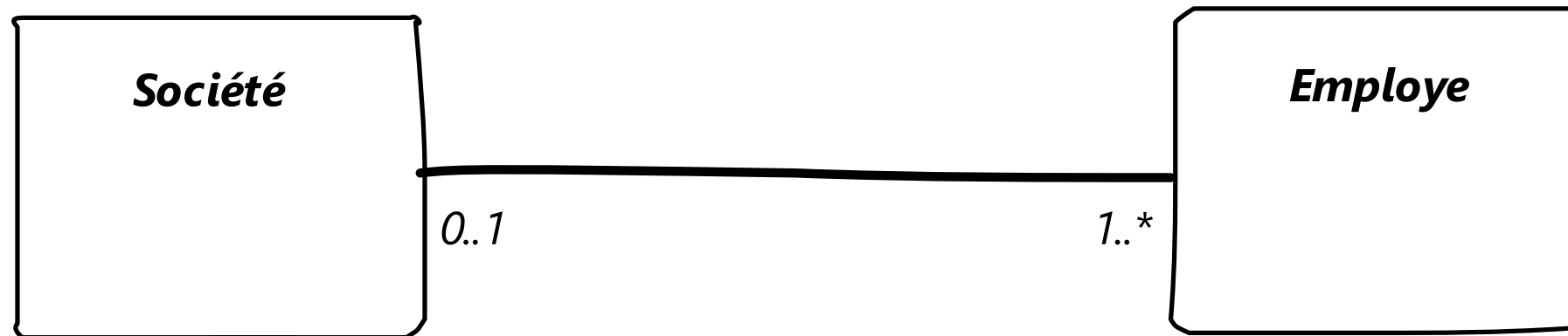
Multiplicités – Exemple 1



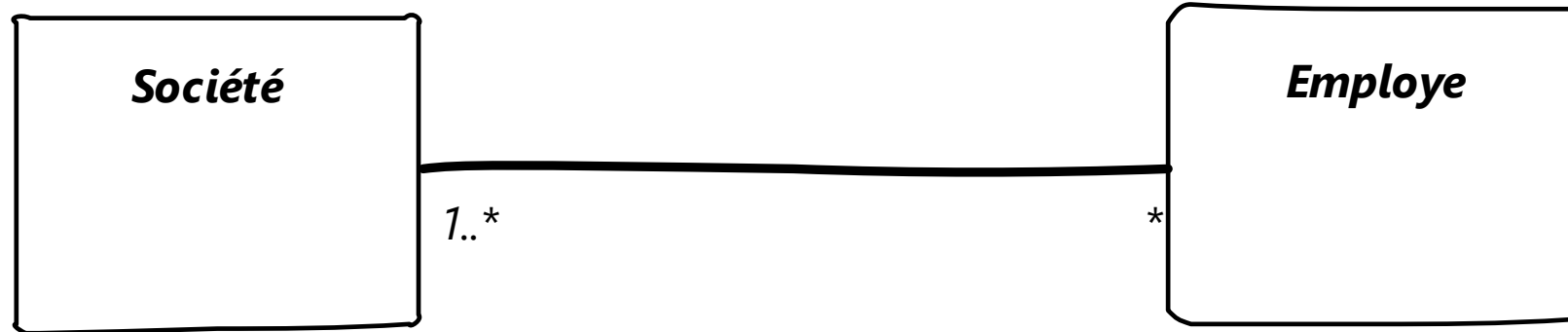
Multiplicité – Exemple 2



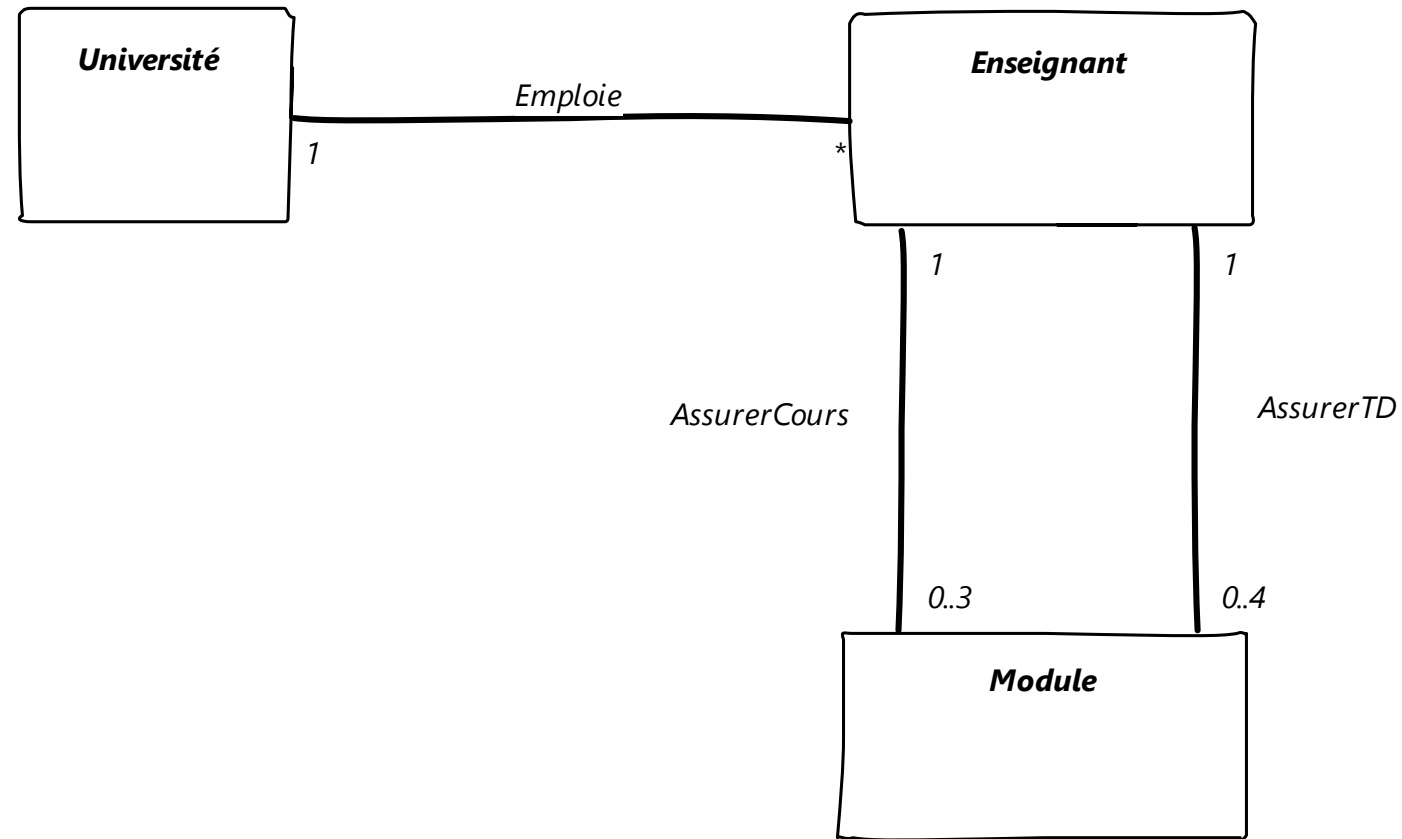
Multiplicité – Exemple 3



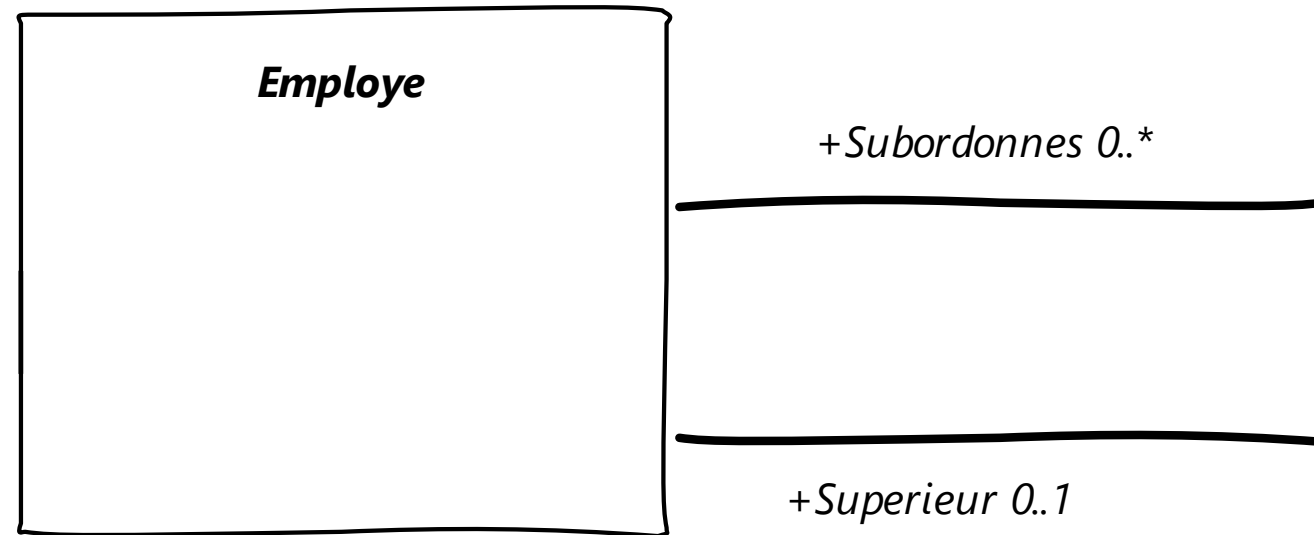
Multiplicité – Exemple 4



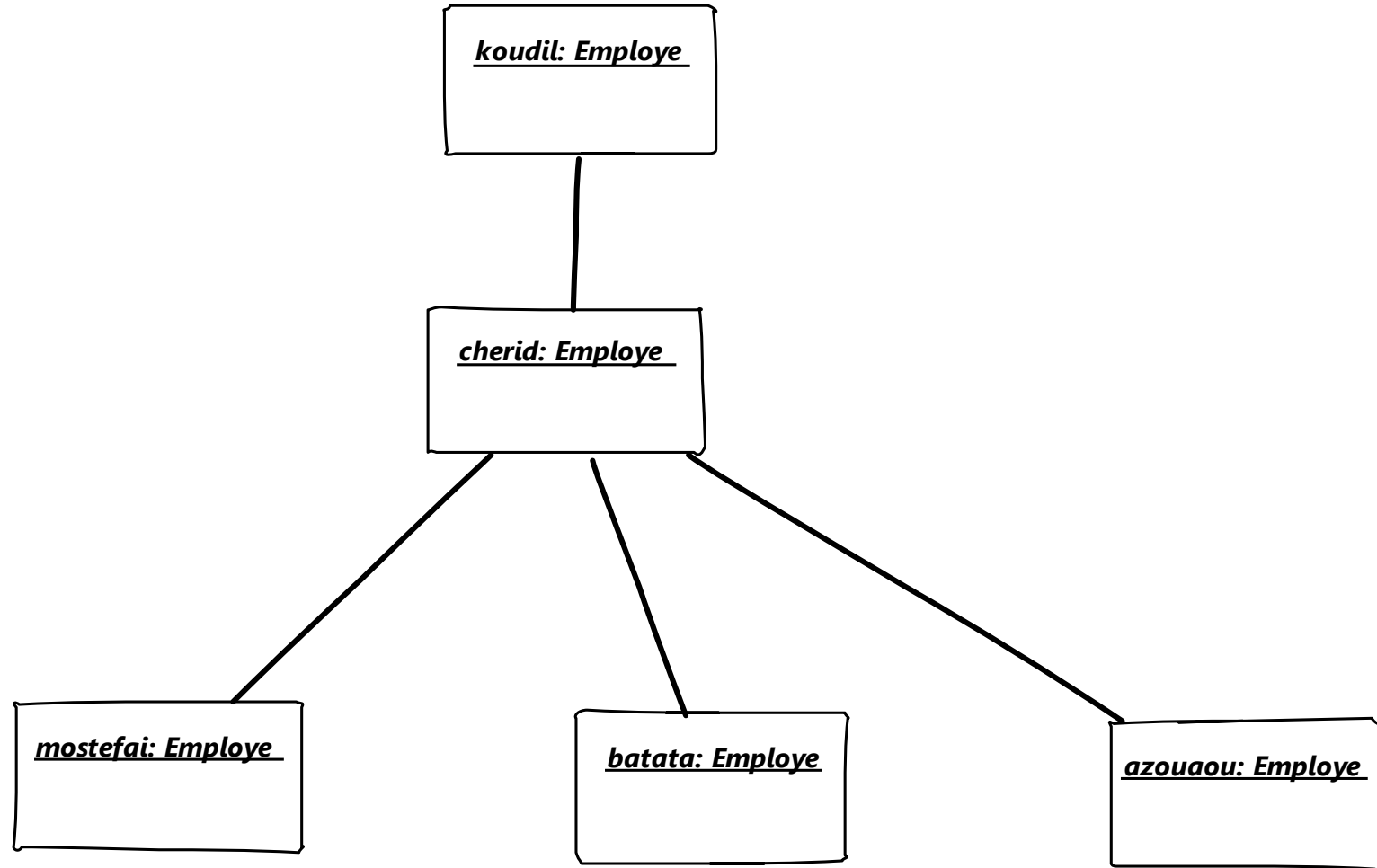
Multiplicité, Exemple 5



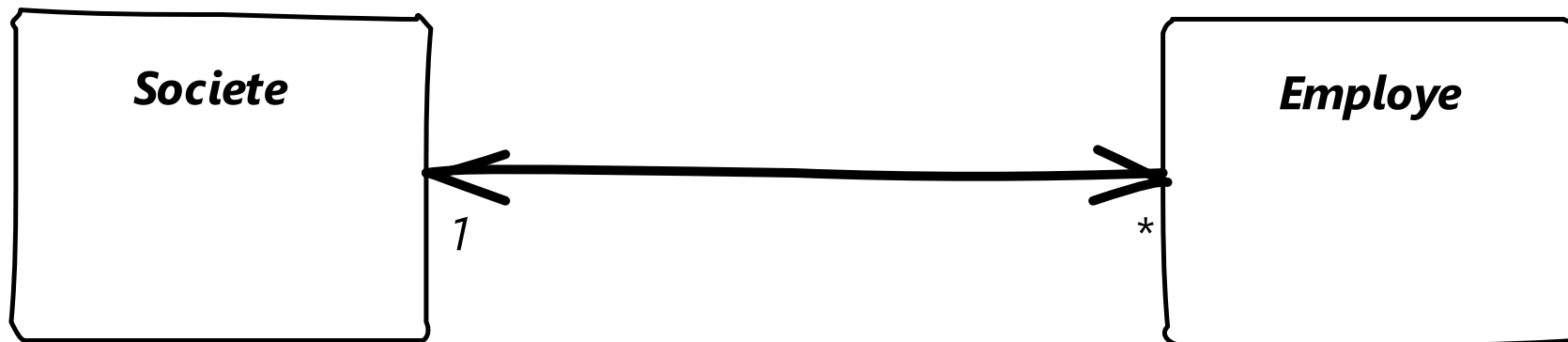
Associations Réflexives



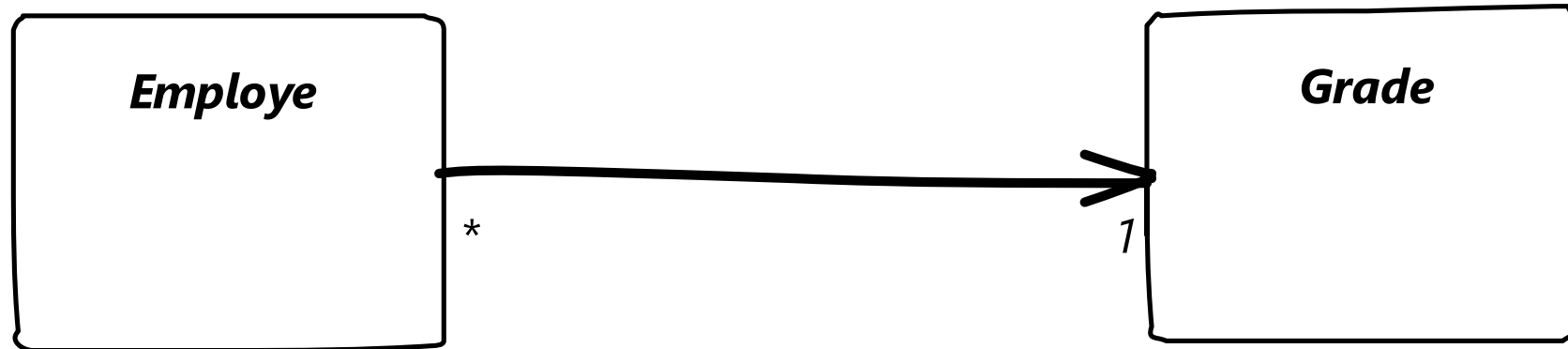
Associations Réflexives – Objet



Navigation, Exemple 1



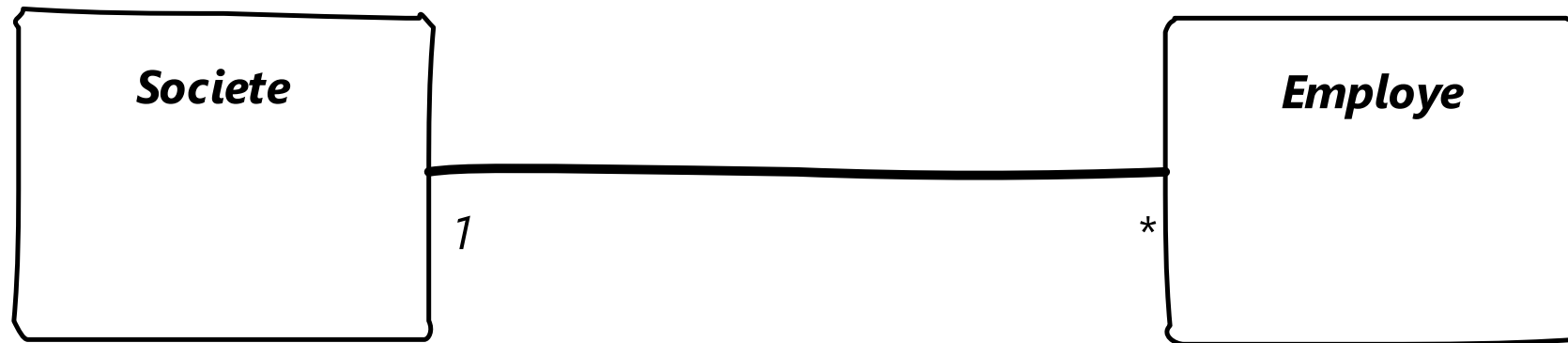
Navigation, Exemple 2



Navigation, Exemple 3



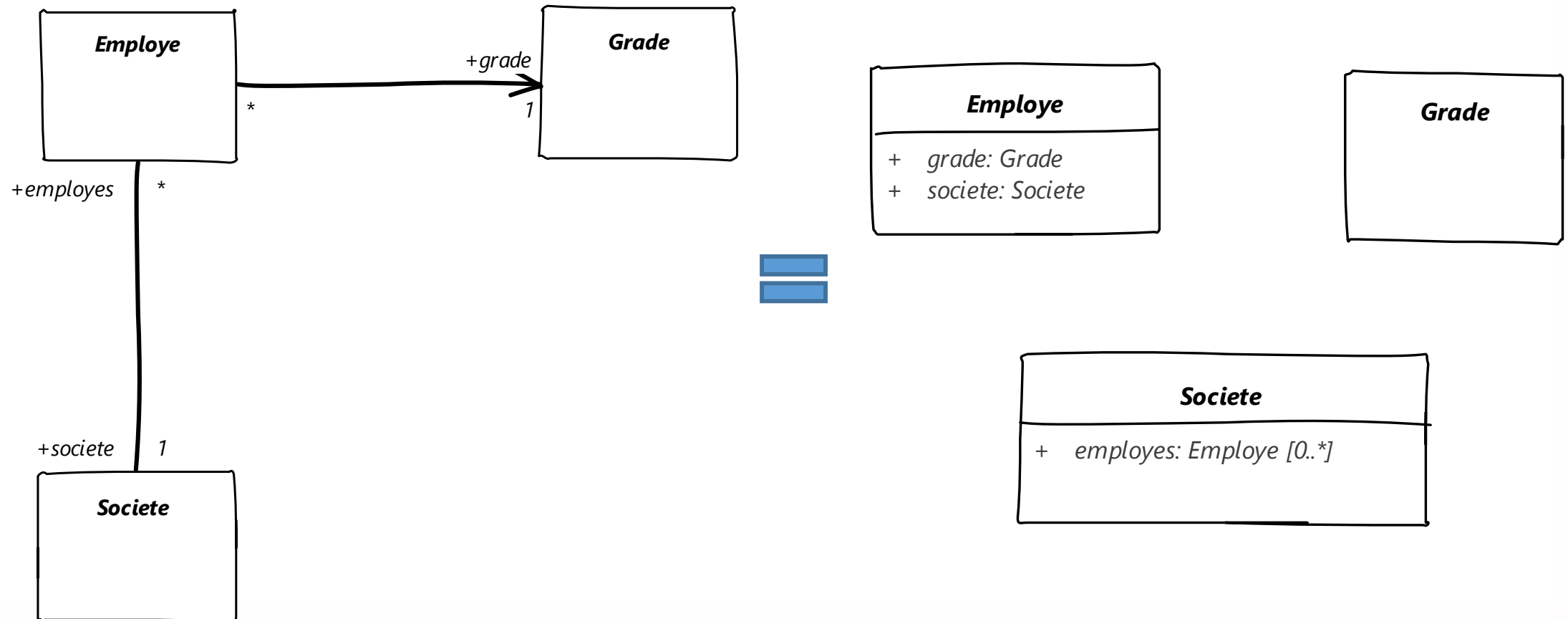
Navigation, Exemple 4



Associations et attributs

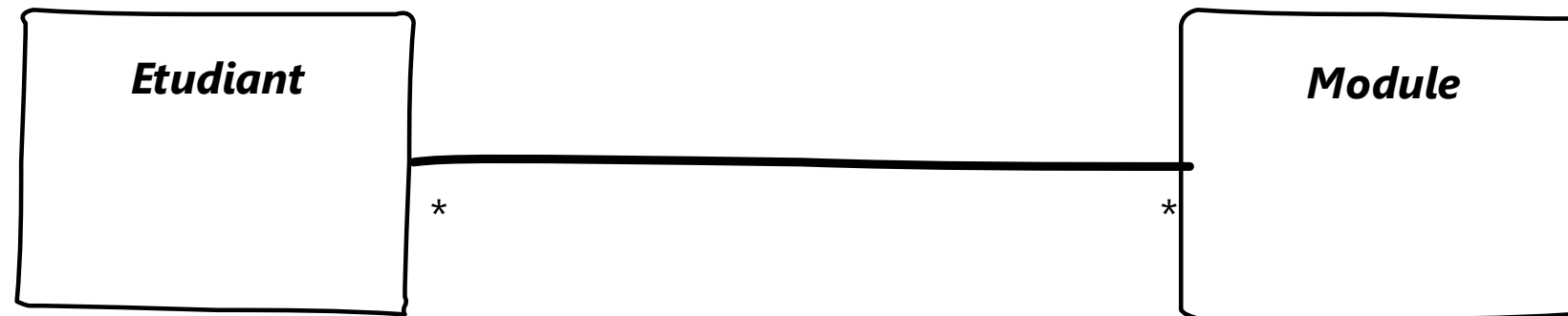
- Les attributs peuvent être un autre moyen de représenter une association
- Les associations à multiplicité multiple peuvent être représentées par des tableaux ou des collections
- Lors de la génération de code, les associations sont générées en tant qu'attributs

Associations et Attributs, Exemple



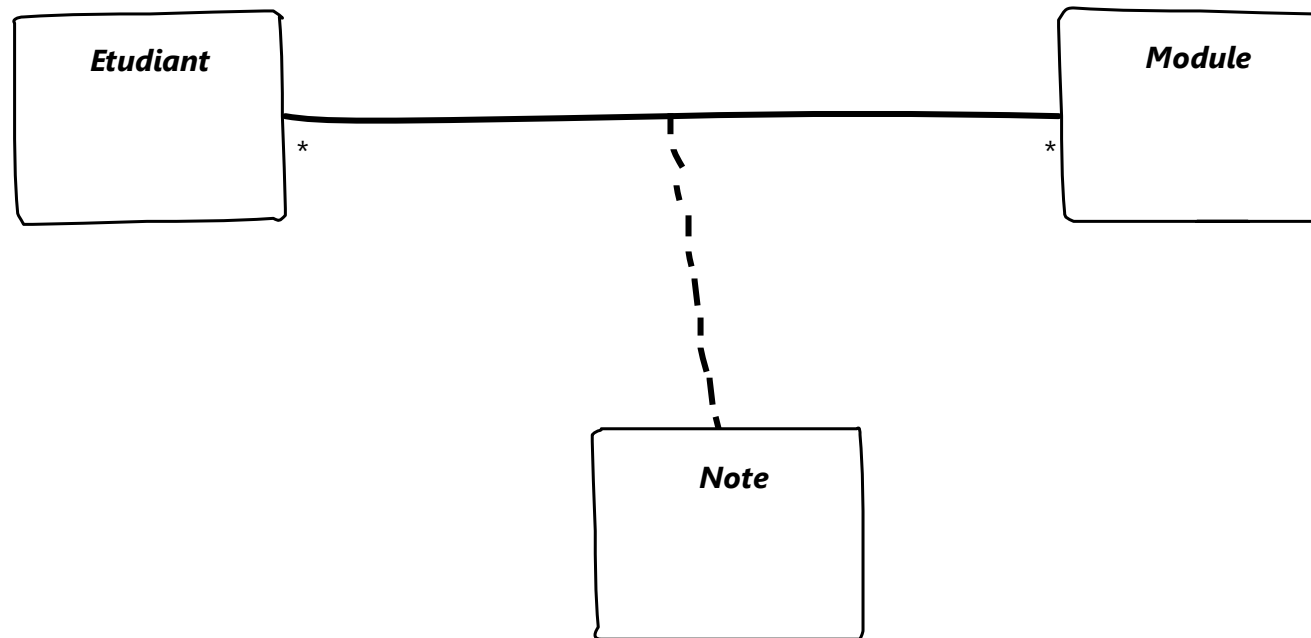
Classes d'Associations

- Parfois, quand il y a une association entre deux classes, l'attribut ne peut être dans aucune des classes.
- Exemple ci-dessous : où mettre la note ?

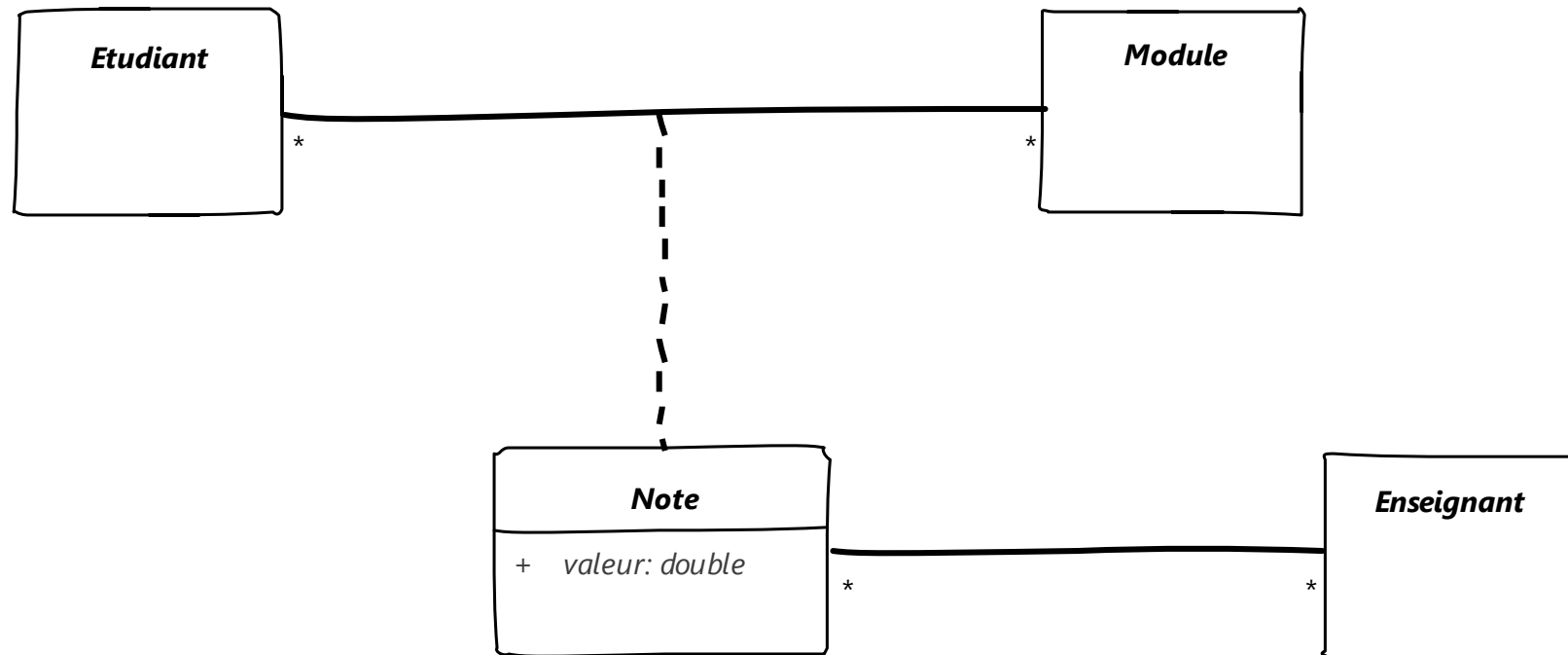


Classes d'Association – Suite

- Les classes d'association sont de vraies classes qui peuvent avoir des attributs, opérations et même d'autres opérations



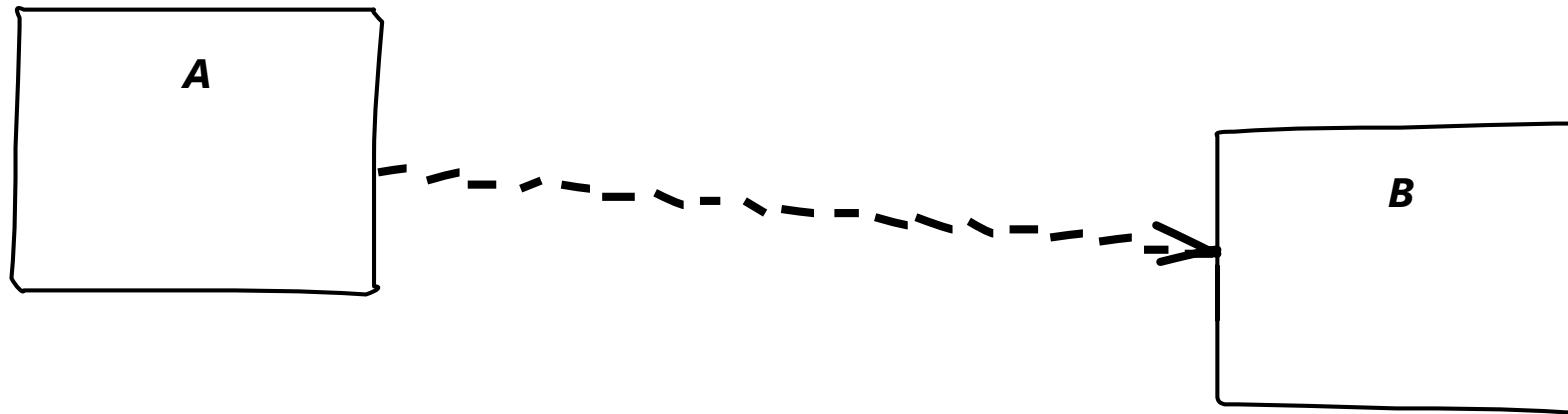
Classe d'Association - Suite



Dépendance

- En plus des associations, les classes peuvent être liées par des relation de **dépendance**
- Plusieurs cas peuvent induire à une **dépendance** entre une classe A et une classe B : A manipule B dans une opération, B est un paramètre dans une opération de A, B est le type de retour d'une opération, A appelle une méthode B, ...etc.
- La dépendance n'est pas réservée aux classes, elle peut être utilisée avec les *paquets* et les *cas d'utilisation*

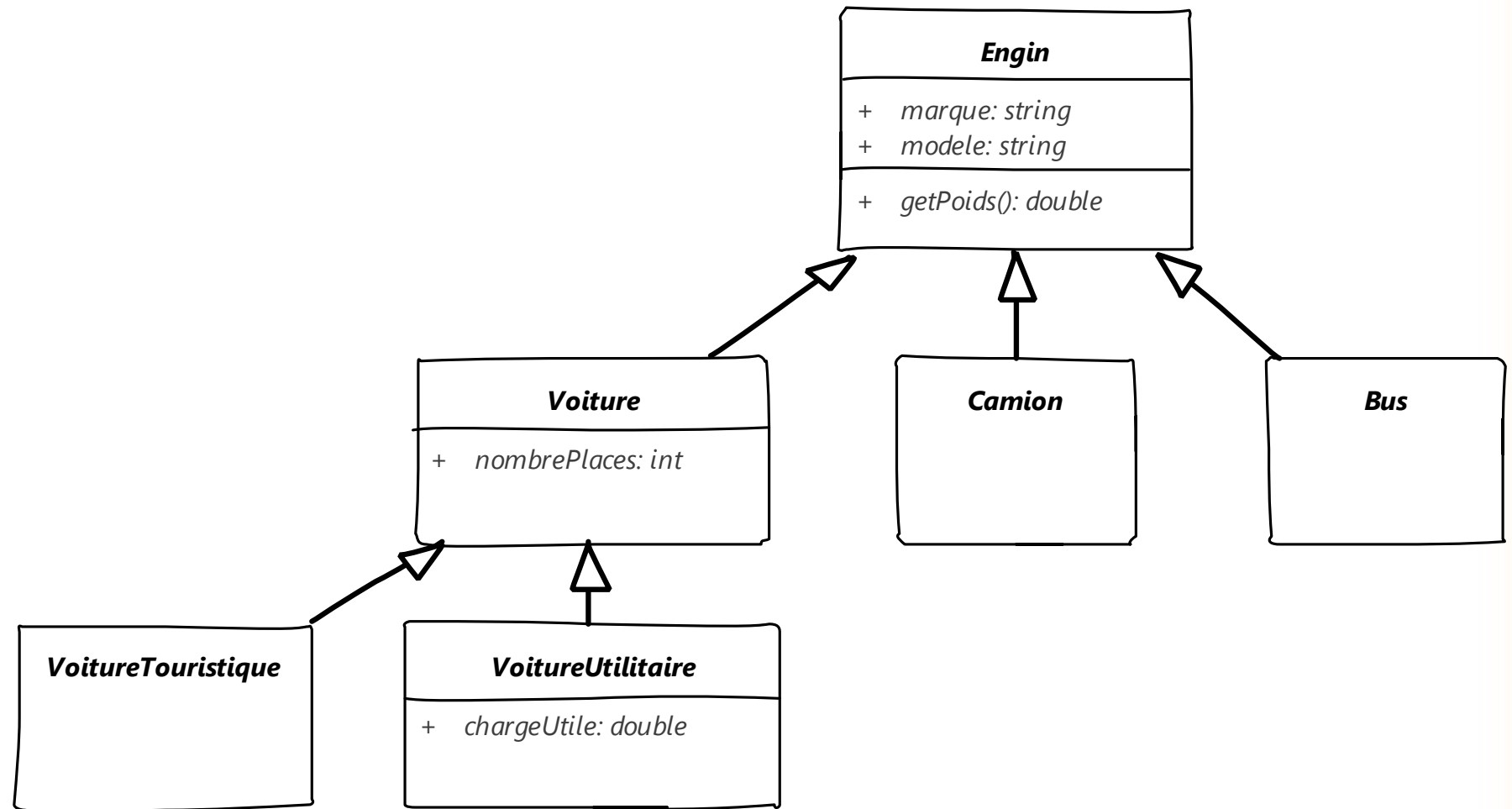
Exemple de Dépendance



Généralisation

- L'héritage est une relation entre une classe *générale* et une classe *plus spécifique*
- Entre les deux, une relation de *substitution* : on peut *substituer* toute utilisation de l'élément général par l'élément plus spécifique
- L'héritage est symbolisé par *le lien de généralisation*
- La classe spécialisée hérite de *tous* les attributs et les opérations de la classe *parente*

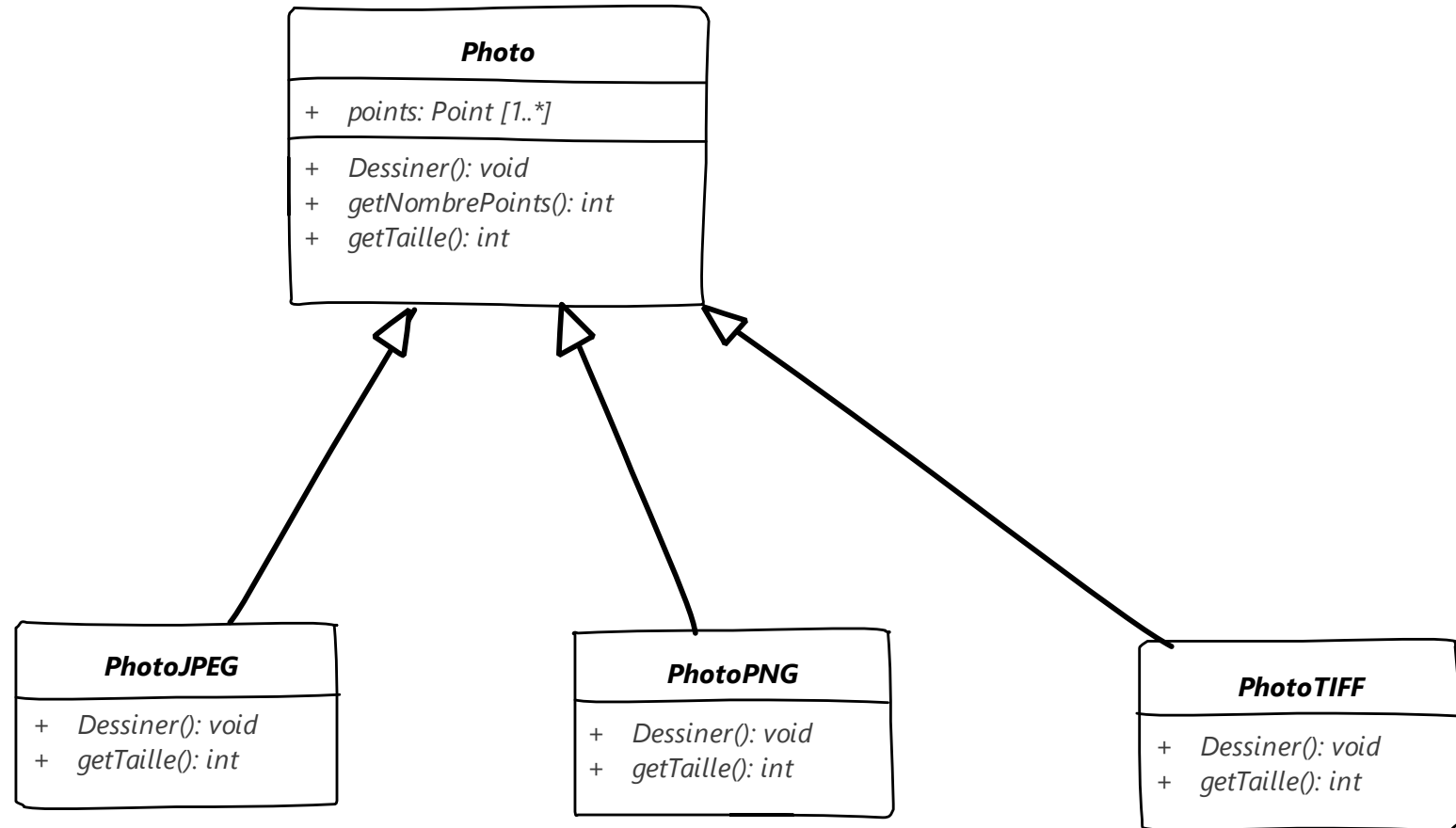
Généralisation, Exemple



Abstraction

- Parfois une classe ne peut exister que par l'existence des classes descendantes
- Une classe peut aussi déléguer l'implémentation d'une opération à ses classes descendantes
- Une opération qui est sans implémentation dans la classe actuelle est une ***opération abstraite***
- Une classe qui contient une ou plusieurs opérations abstraites est dite ***classe abstraite***

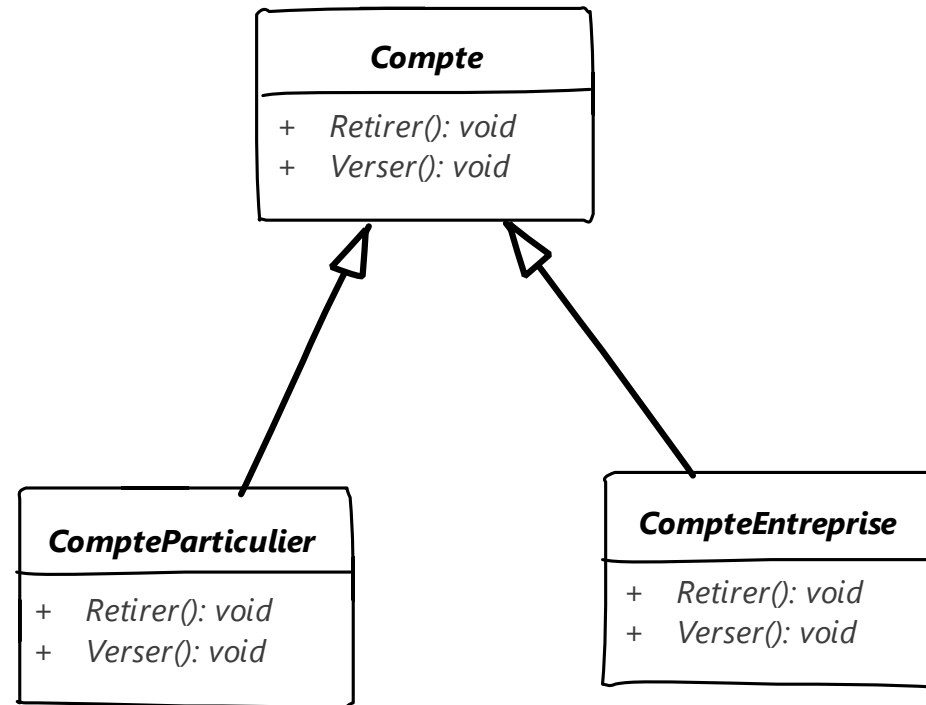
Abstraction, Exemple



Polymorphisme

- Une opération polymorphique est une opération qui possède *plusieurs* implémentations
- Par exemple, n'importe quel compte accepte les retraits et les versements. Par contre, les comptes entreprises ont un comportement différent des comptes particuliers. Les comptes entreprises acceptent les soldes négatifs.

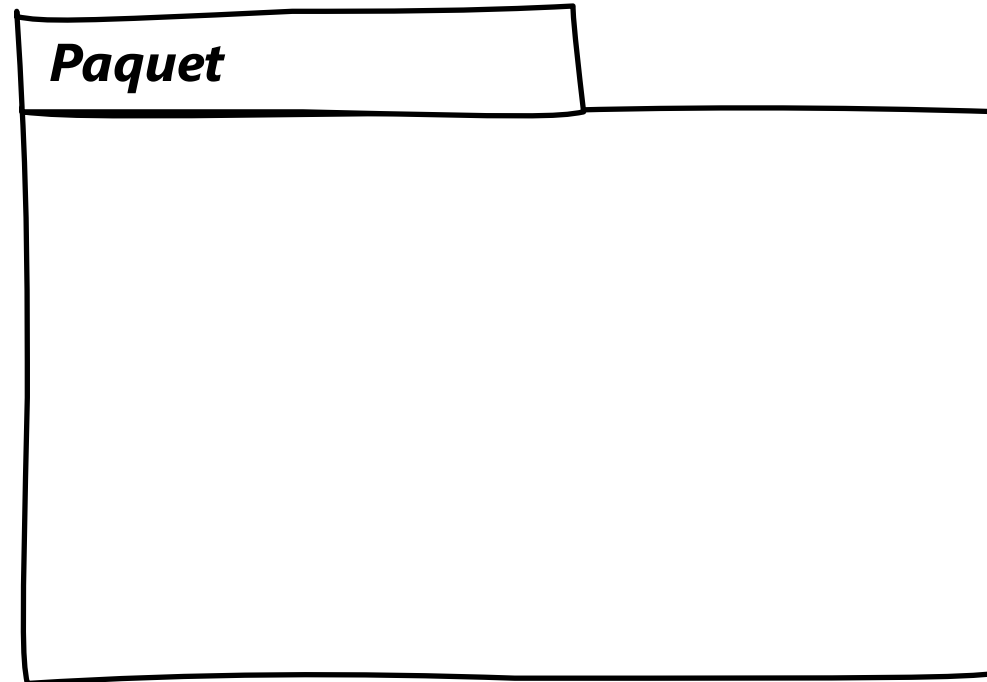
Exemple - Polymorphisme



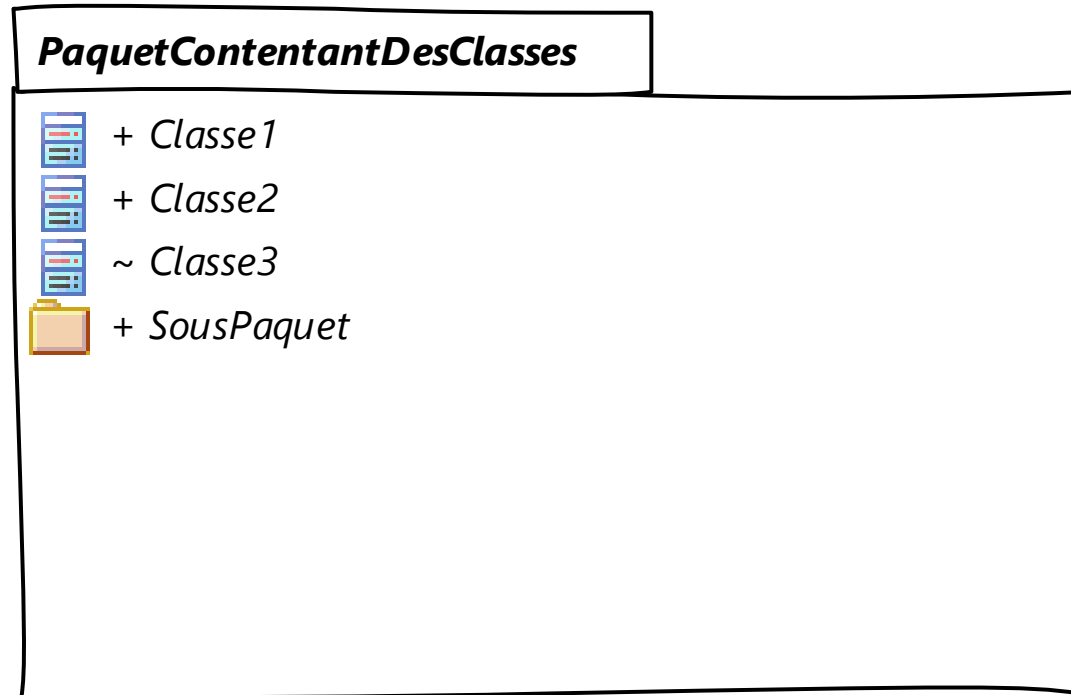
Paquets

- Un paquet UML est un élément de **groupage** qui contient **plusieurs** éléments UML dont éventuellement d'autres paquets
- Les paquets servent à **organiser** les éléments UML
- Les paquets définissent des « **frontières sémantiques** » du modèle
- Chaque élément UML appartient à exactement **un** paquet. Les éléments de haut niveau appartiennent à un paquet implicite appelé « root » ou « topLevel »

Paquets – Notation UML



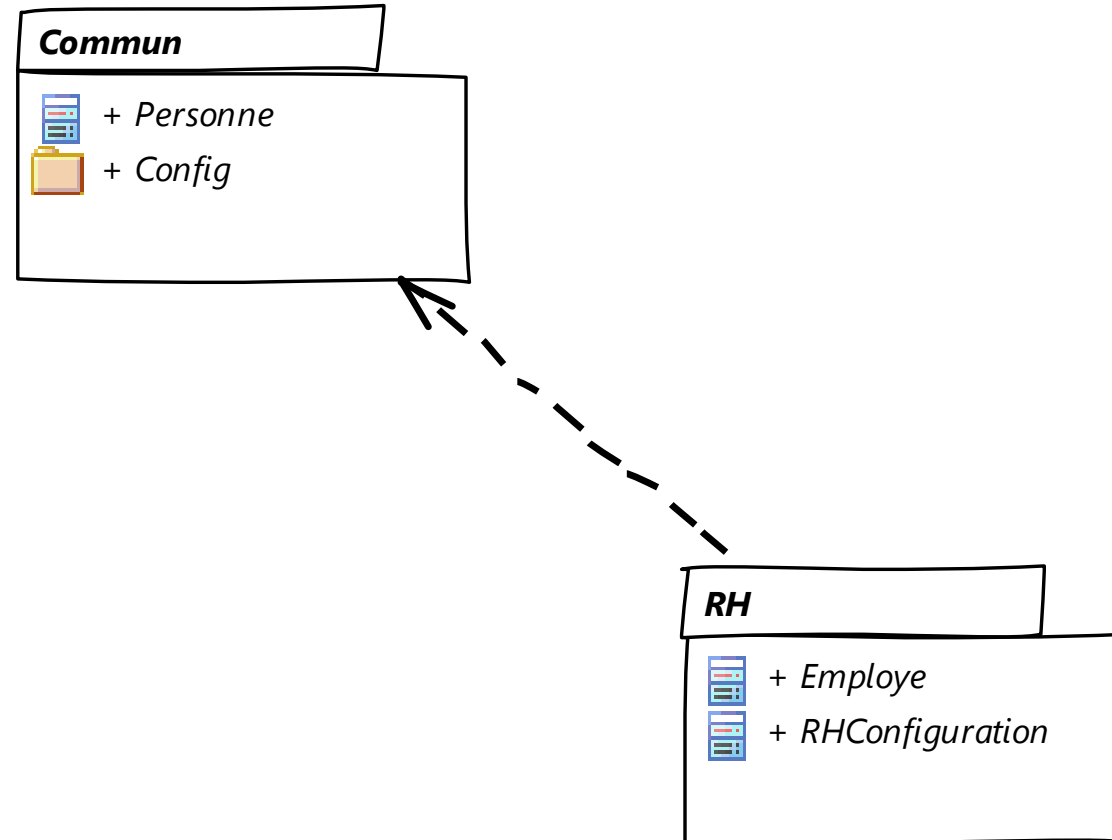
Paquets – Notation UML



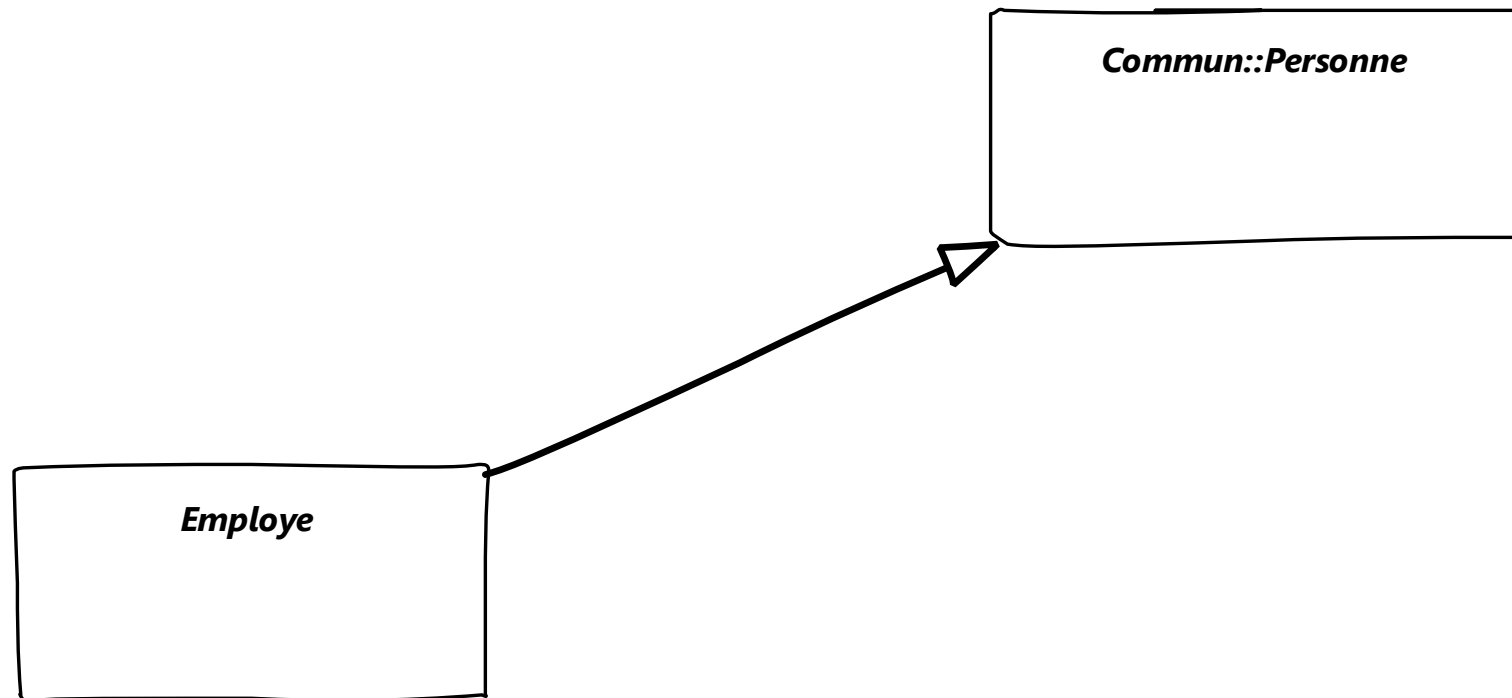
Paquets – Espaces de noms

- Le paquet définit une *frontière* où les noms des éléments doivent être *uniques*
- Si un élément doit référer à un autre élément se trouvant dans un autre paquet, il doit utiliser un « *nom qualifié* ».
- Le nom qualifié se fait en préfixant le nom de l'élément par les noms des paquets qui le contiennent
- Par exemple, si un paquet P₁ contient un sous-paquet P₂ contenant une classe C₁, le nom qualifié de C₁ est **P₁:P₂:C₁**.

Paquets – Dépendance



Éléments d'autres paquets



Recensement des Classes



SECTION 3

Les classes d'analyse

- Les classes d'analyse sont les classes **découvertes** durant l'activité d'analyse
- Les classes d'analyse correspondent à des **concepts** « réels »
- L'analyse fait ressortir les éléments suivants d'une classe : Nom, Attributs les plus importants, Opérations les plus importantes, stéréotypes (non techniques)
- Les éléments suivants ne sont pas importants dans l'analyse : métadonnées, paramètres des opérations, visibilité

Sources de recensement

Modèle de
spécifications

Modèle des cas
d'utilisation

Toute autre source
d'information
relative au
domaine

Représentation d'une classe d'analyse

- Les classes d'analyse sont représentés en utilisant le diagramme des classes
- L'ensemble des classes sont appelés « **vocabulaire** », « **glossaire métier** » ou « **taxonomie du domaine** »
- Les classes d'analyse sont une représentation de haut niveau : elles sont caractérisées par un nom, des attributs et éventuellement des opérations.
- Le **nom de la classe** est **obligatoire**.
- Les **noms des attributs** sont **obligatoires**. Leurs types facultatifs.
- Les **paramètres et les types de retour** des opérations ne sont montrés que s'ils apportent une compréhension au système

Représentation d'une classe d'analyse

- La **visibilité** est généralement omise
- Les **stéréotypes** ne sont montrés que s'ils augmentent la représentativité
- Les **métadonnées** ne sont montrés que s'ils augmentent la représentativité

Exemple d'une Classe d'Analyse

Compte
numero proprietaire solde
verser() retirer()

Techniques de recensement

Technique
des noms /
verbes

Technique
des
stéréotypes

Techniques des Noms / Verbes

- Technique simple qui **analyse du texte** pour trouver des classes, des attributs et des opérations
- Les noms et les **phrases nominales** représentent les **classes** et les **attributs**
- Les **verbes** et les **phrases verbales** représentent les **opérations**
- Difficultés avec les termes difficiles, les synonymes et les homonymes
- Difficulté à trouver les classes « cachées »
- S'il y a des termes qui ne sont pas compris, voir avec les experts du domaine

Technique des noms / verbes – Suite

Collecte d'information

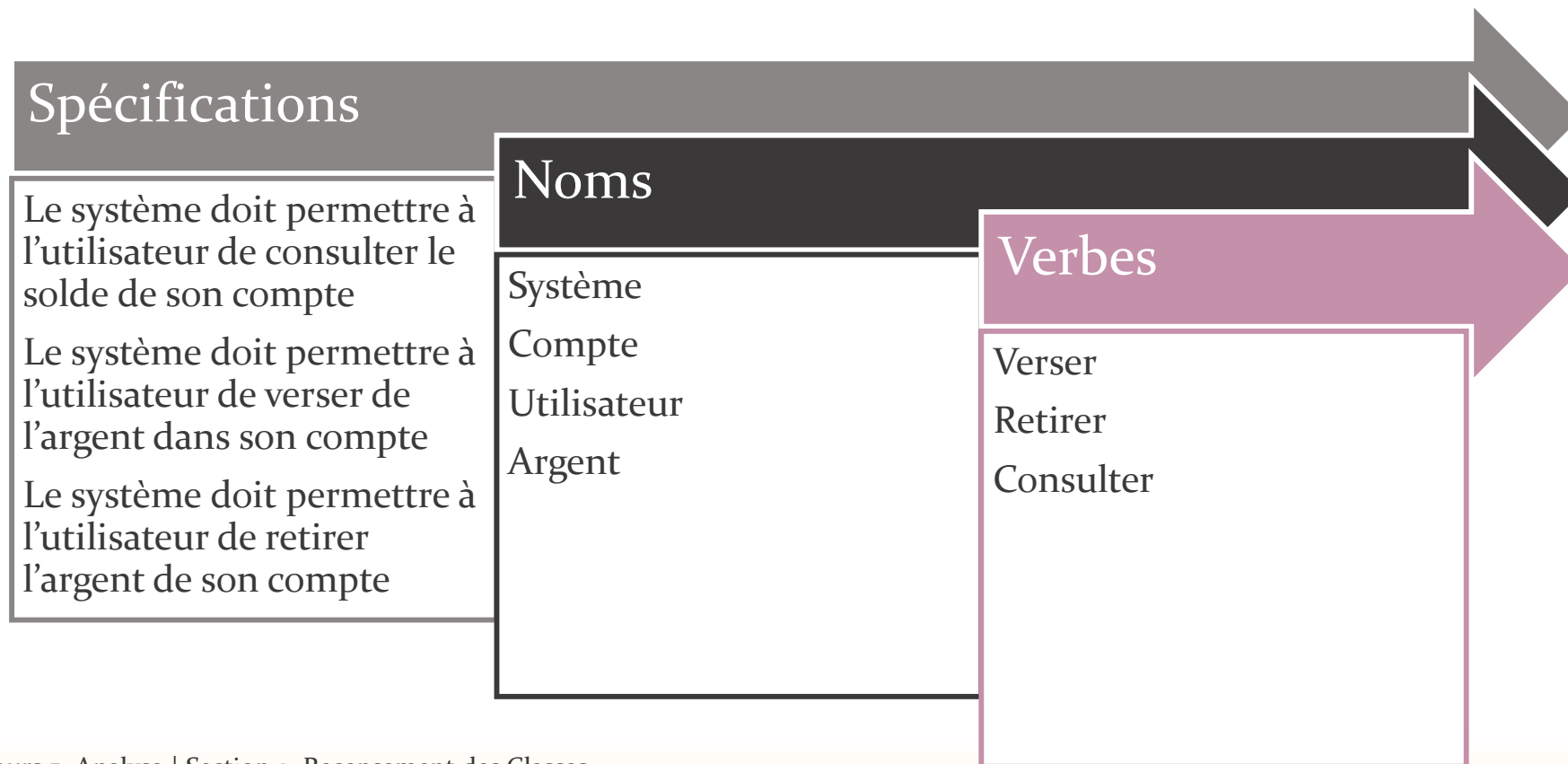
- Spécifications formelles
- Cas d'utilisation
- ...



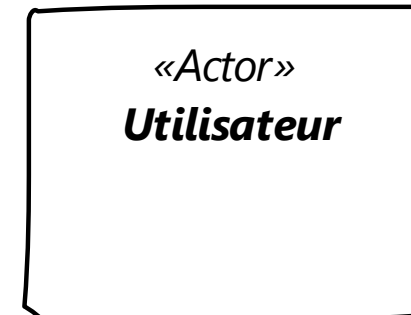
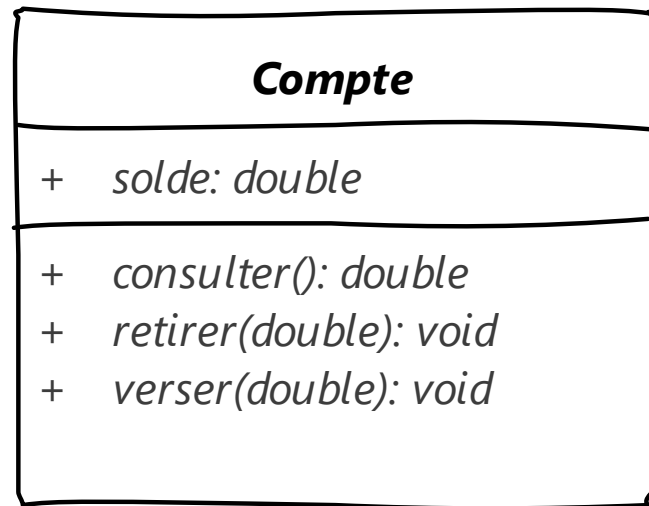
Analyse des informations

- Noms et phrases nominales
- Verbes et phrases verbales

Technique des noms / verbes - Exemple



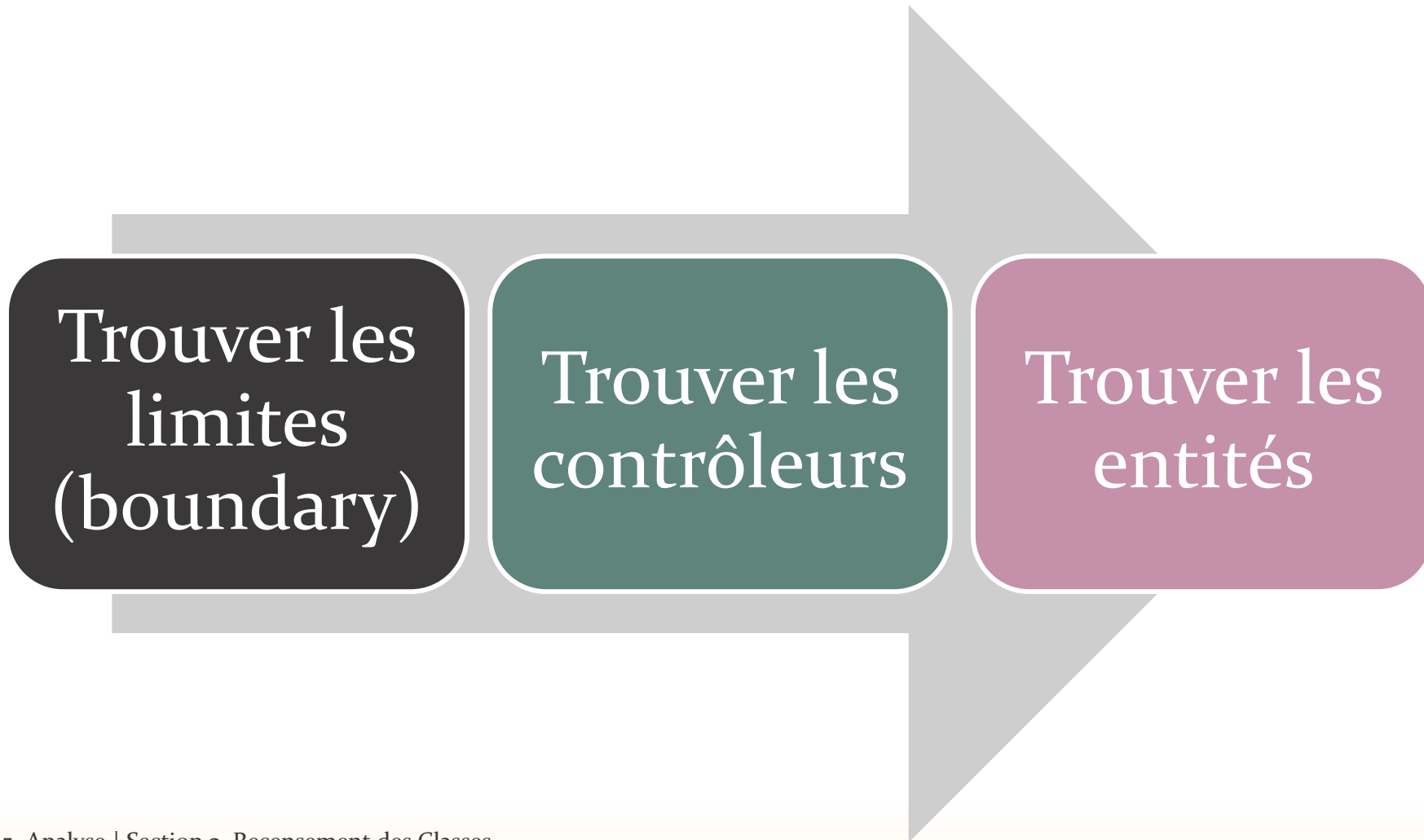
Technique des noms / verbes - Exemple



Technique des stéréotypes

- Cette technique se concentre sur trois **stéréotypes** : « boundary », « entity » et « control »
- Cette technique est **complémentaire** avec la technique des verbes / noms

Technique des stéréotypes



Technique des stéréotypes - Limites

- Les classes « boundary » représentent les **limites** du système
- Ces classes communiquent avec les acteurs
- Trois types de classes « boundary » : classes *d'interface utilisateur*, classes qui *s'interfacent avec d'autres systèmes* et des classes qui *s'interfacent avec un matériel externe* (périphérique,...)
- Ces classes doivent rester de **haut niveau** (ne pas s'occuper des détails)

Technique des stéréotypes - Contrôleurs

- Les contrôleurs sont responsable de la **coordination** des action permettant d'accomplir un cas d'utilisation
- Si un contrôleur est trop compliqué, il est suggéré de le **diviser** en plusieurs contrôleurs

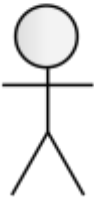
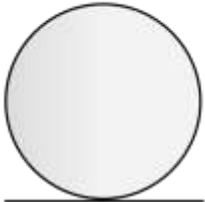
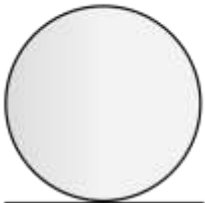

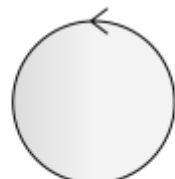
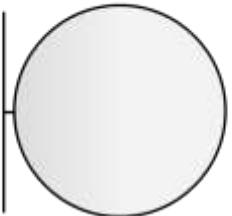
Technique des stéréotypes - Entités

- Représentent les **concepts** manipulés par le système
- Utilisés par plusieurs **cas d'utilisation** et plusieurs **contrôleurs**
- Ces classes sont souvent **persistantes** (enregistrées dans des fichiers ou des BDD)

Technique des stéréotypes - Exemple

1. Le système doit permettre à l'utilisateur de s'inscrire
2. Le système doit permettre à l'utilisateur connecté de consulter les formations assurés, leurs dates et les formateurs associés
3. Le système doit envoyer un message mensuellement qui avertit les utilisateurs inscrits des nouvelles formations

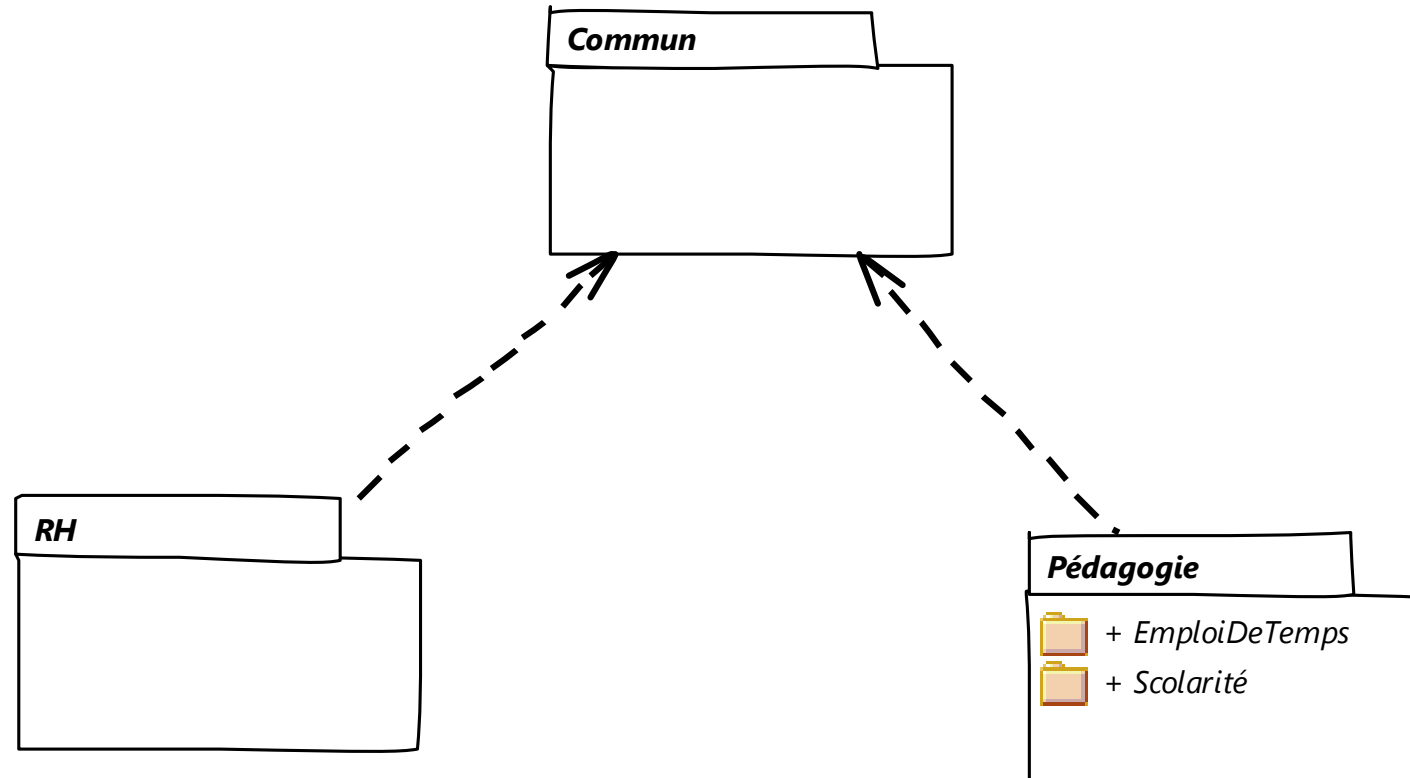
Technique des stéréotypes - Exemple

Acteurs	Entités	Contrôleurs	Limites
 Utilisateur	 Formation  Formateur	 InscriptionManager  NotificationManager	 InterfaceInscription

Paquets d'analyse

- Les classes d'analyse sont *regroupées* dans des « paquets d'analyse »
- Les classes se trouvant dans le même paquet sont des classes ayant un certain *lien sémantique*
- Un bon regroupement des paquets produit une haute « cohésion » à l'intérieur d'un paquet et un faible « couplage » entre paquets

Paquets d'analyse - Exemple



Recensement des Classes



SECTION 3, DÉBAT 05 MNS

Comportements et Interactions



SECTION 4

Introduction

- Les classes d'analyse représentent la *structure statique* du système
- Les interactions expriment comment les instances de ces classes *interagissent* pour réaliser une fonction du système
- Les interactions expriment l'aspect dynamique du système

Objectifs des interactions

- Trouver quelles classes qui **interagissent** pour un cas d'utilisation donné
- Trouver les **messages** envoyés entre les classes pour réaliser un certain comportement (attributs, opérations, relations)
- Eventuellement, **mettre à jour** les modèles de besoins et d'analyse
- Ne pas créer d'interactions pour tous les cas d'utilisation, **Uniquement** pour les CUs les plus importants et les plus complexes

Diagrammes comportementaux

- Les **diagrammes comportementaux** est la meilleure façon de représenter les comportements et interactions
- Les **diagrammes de séquence** illustrent une séquence de messages ordonnée dans le temps
- Le **diagramme de communication** décrit les relations entre objets dans un contexte donné
- Le **diagramme d'activité** décrit le flux d'actions menant à la réalisation d'une fonction métier
- Le **diagramme d'état** démontre le changement d'état d'une entité dans le temps

Comportements et Interactions



SECTION 4, DÉBAT 05 MNS

Diagrammes de Séquence



SECTION 5

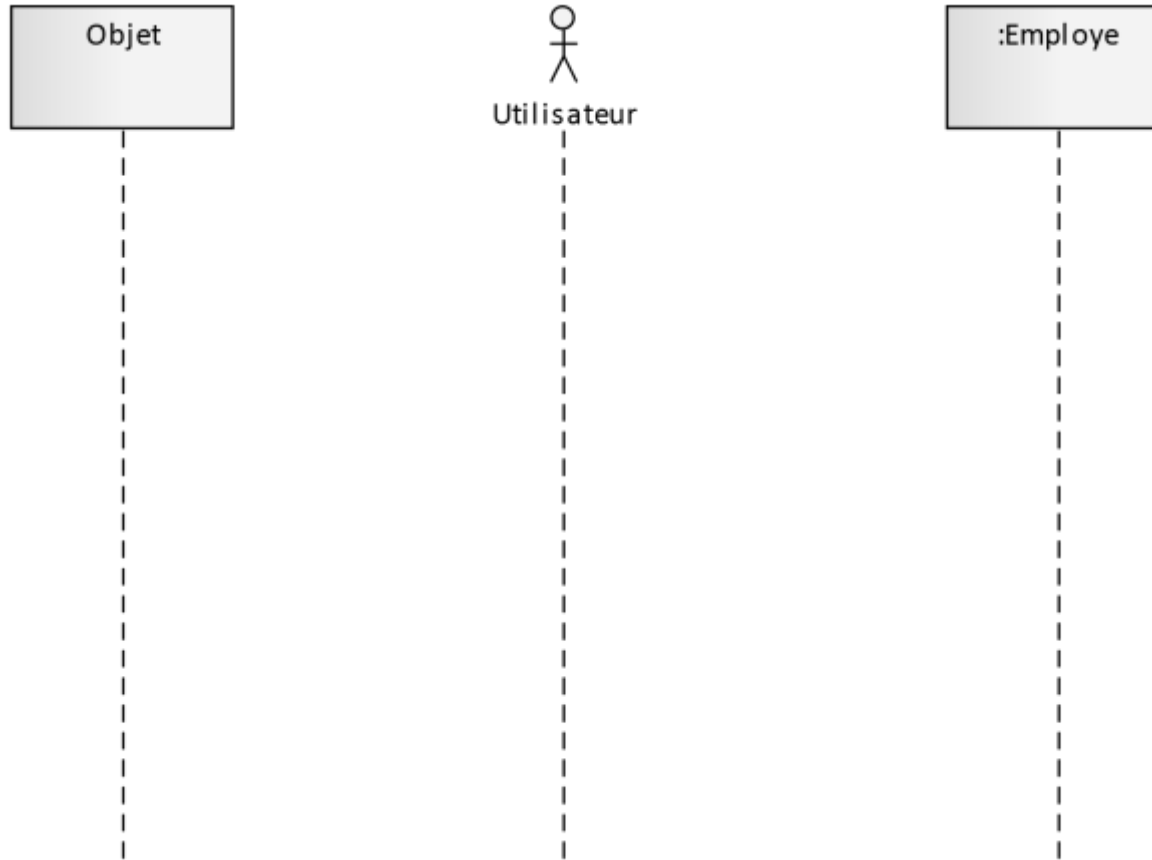
Présentation

- Les diagrammes de séquence (DSQ) décrivent une action *ordonnée* dans le temps
- Les DSQ document les CU et font partie du modèle d'analyse
- Les DSQ sont composés de trois concepts principaux : les *lignes de vie*, les *messages* et *les fragments*

Lignes de vie

- Une ligne de vie représente *un seul participant* dans une interaction
- Une ligne de vie peut représenter un *objet*, une *instance d'une classe* ou un acteur
- Pour représenter une interaction, des *messages* lient les lignes de vie

Lignes de vie – Représentation UML



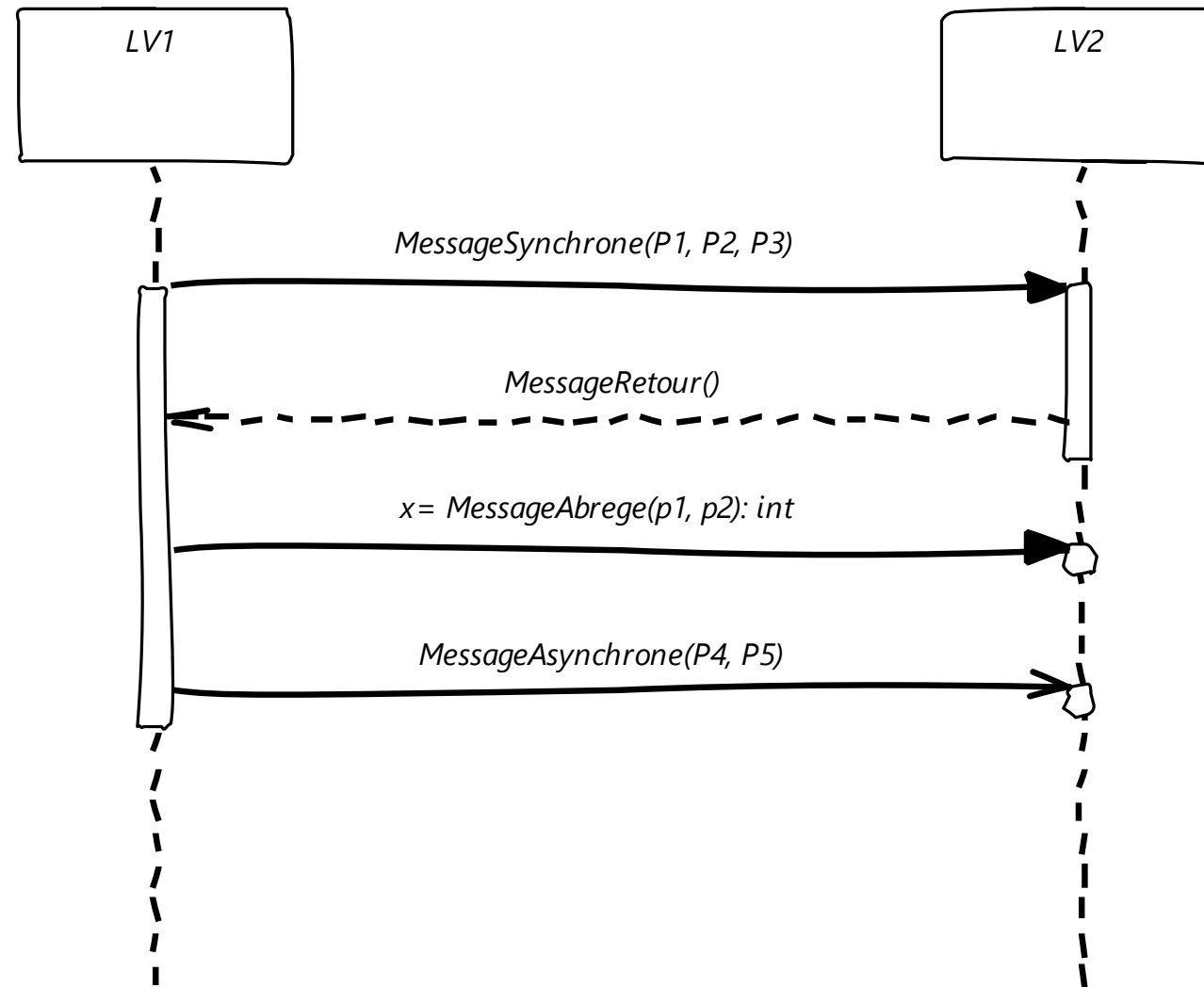
Messages

- Un message représente une **communication** entre deux lignes de vie durant une interaction
- Un message peut s'agir de ***l'appel d'une opération***
- Un message peut s'agir de la **création** ou de la **destruction** d'instance
- Un message peut s'agir ***d'envoi d'un signal***
- Quand une ligne de vie reçoit un message, ça correspond généralement à l'appel d'une ***opération ayant la même signature***
- Quand une ligne de vie reçoit un message, elle obtient son **activation**. L'activation change de ligne de vie à travers le temps.

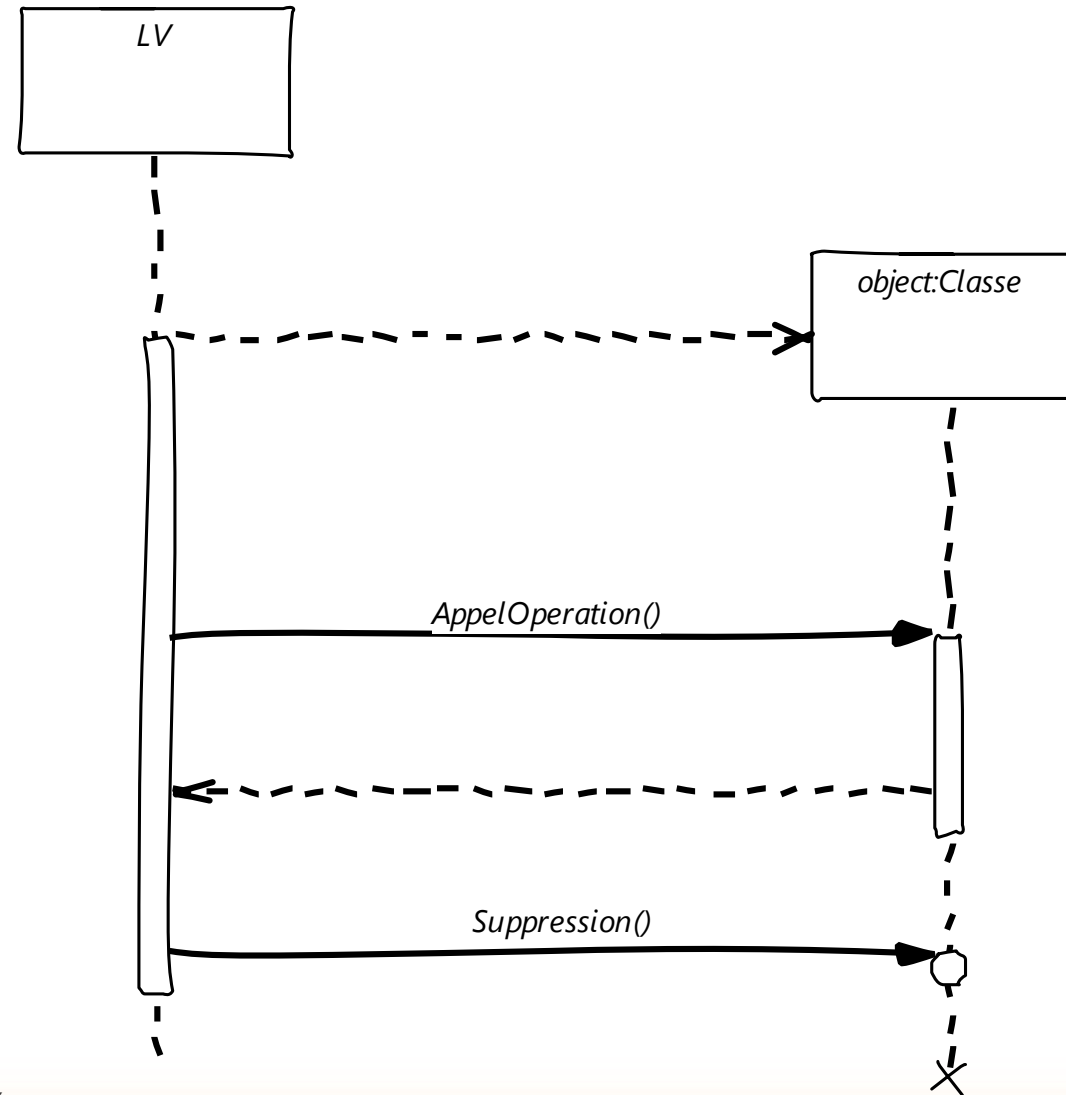
Messages

Type de message	Description
Message synchrone	L'émetteur attend que le récepteur termine l'opération pour passer à l'étape suivante
Message asynchrone	L'émetteur envoie le message puis continue son exécution sans attendre la fin chez le récepteur
Message de retour	L'émetteur récupère l'activation suite à l'avoir perdu en envoyant un message au destinataire
Message de création	Le message provoque la création du destinataire
Message de destruction	Le message provoque la destruction du destinataire
Message found	L'émetteur de ce message ne fait pas partie de l'interaction.
Message lost	Le destinataire ne reçoit jamais ce message. Peut indiquer des situation d'erreur.

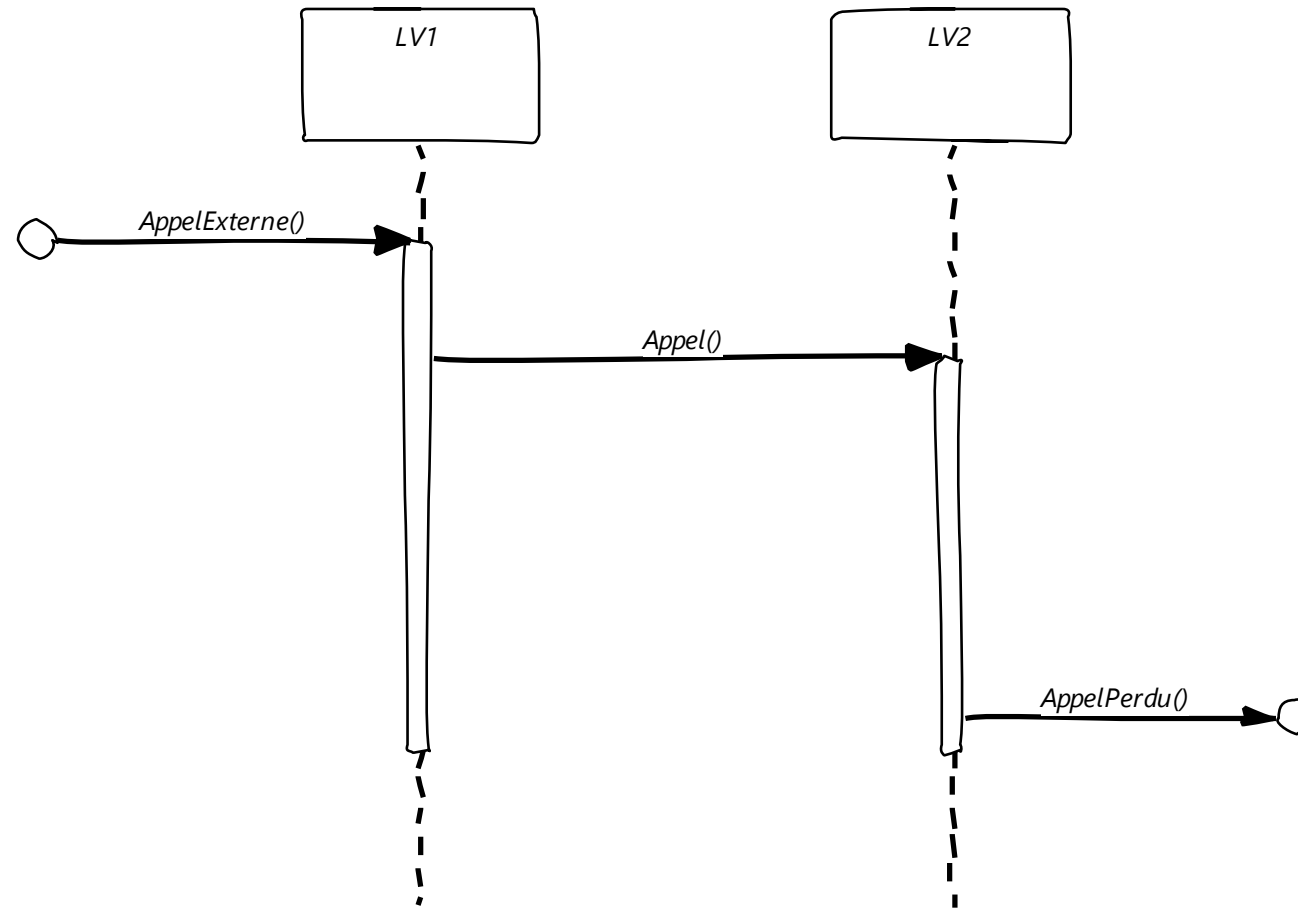
Messages



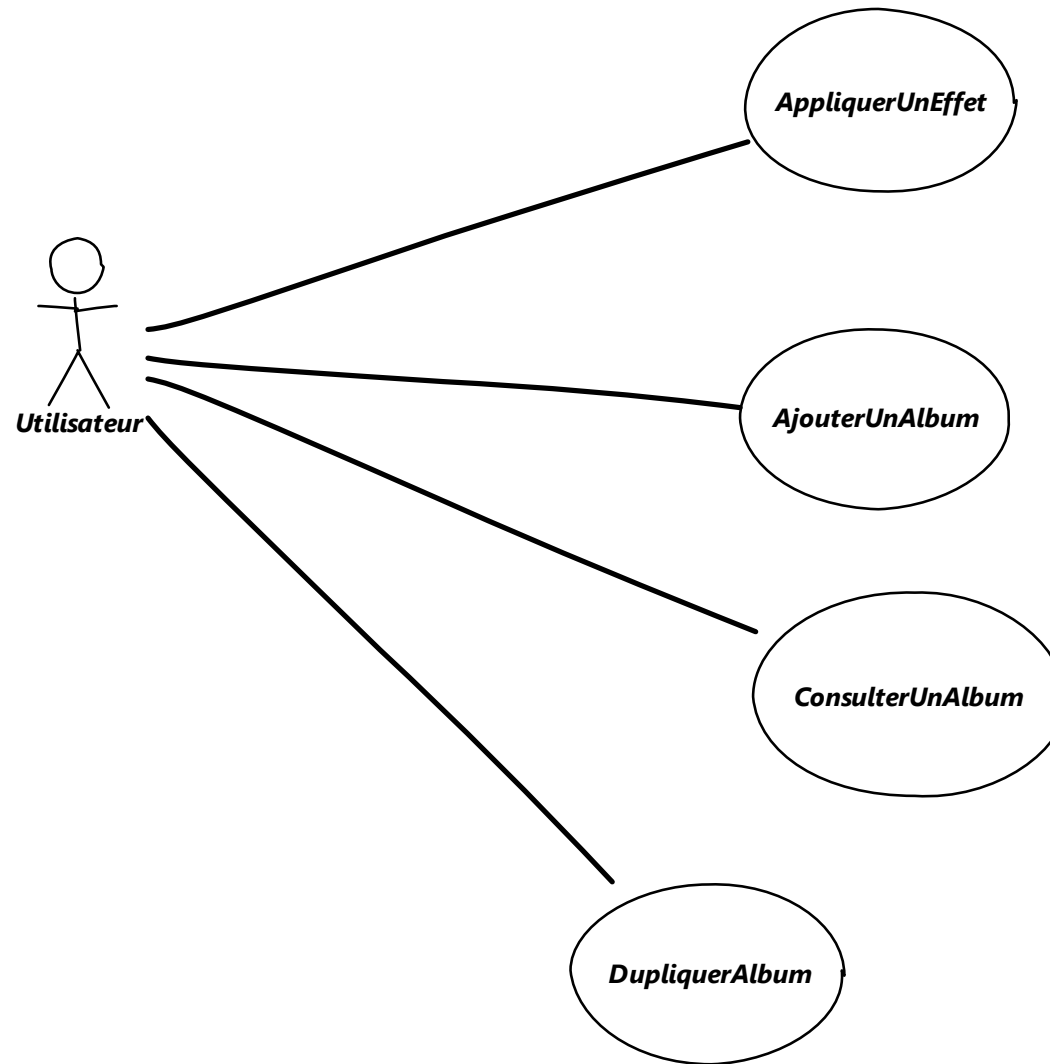
Messages de Création / Destruction



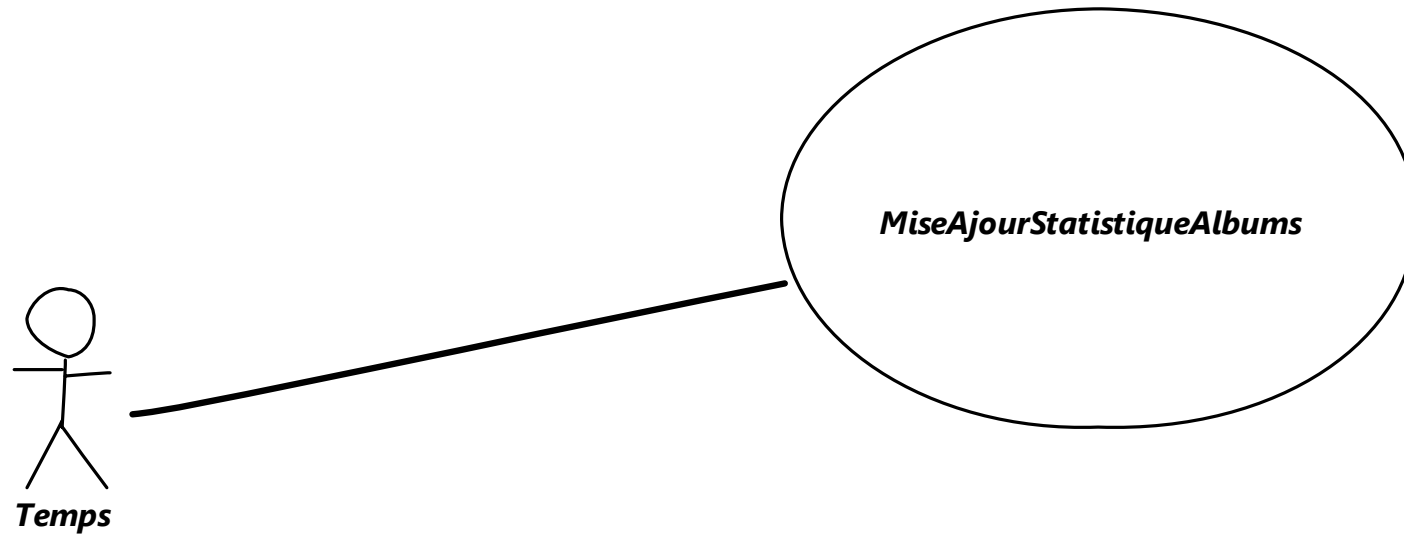
Messages Found / Lost



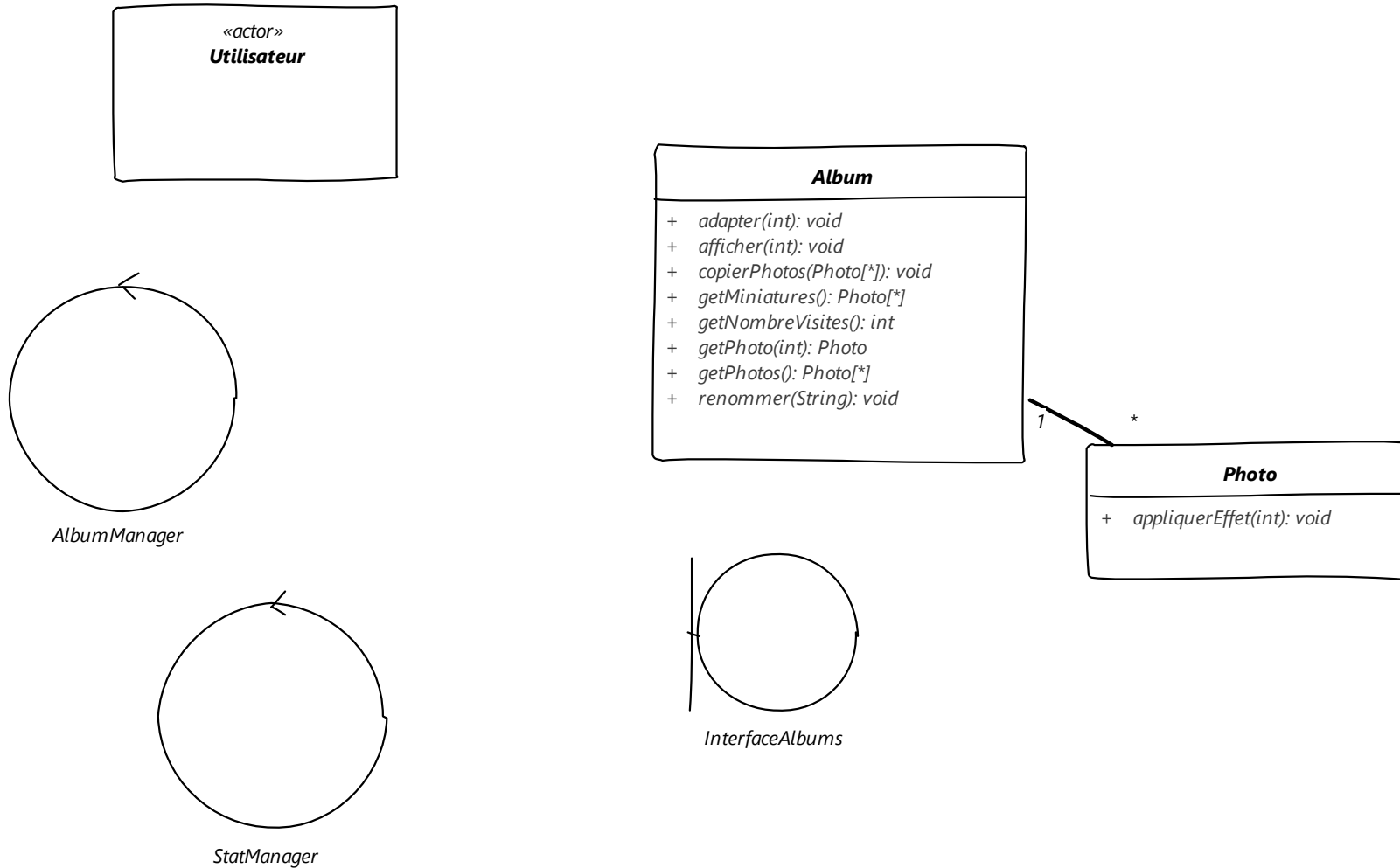
DSQ - Exemple



DSQ – Exemple - Suite



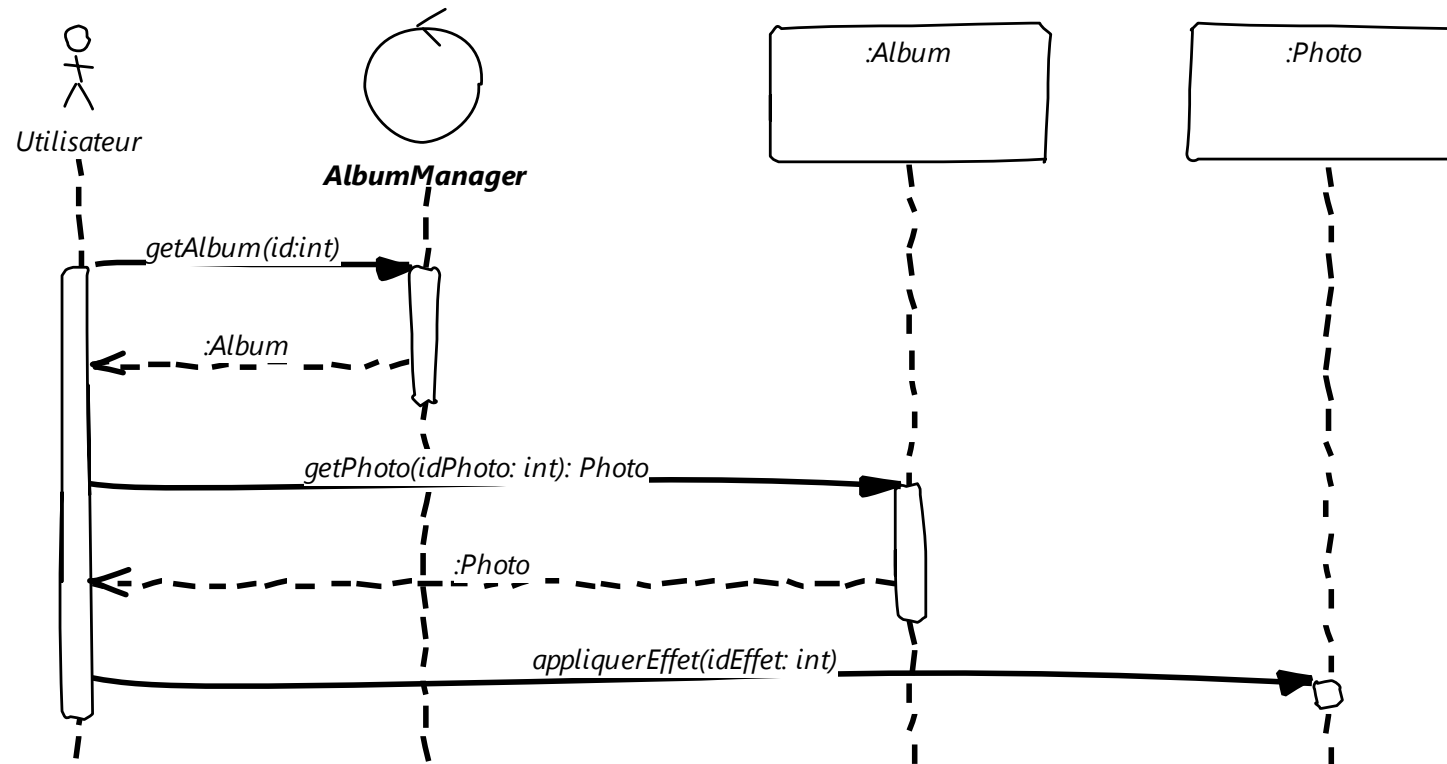
DSQ – Exemple – Suite



DSQ – Exemple - Suite

CU: AppliquerUnEffet
ID: 1
Description brève : Appliquer un effet spécial à une photo d'un album. Chaque effet est identifié par un numéro.
Acteurs primaires : Utilisateurs
Acteurs secondaires : Aucun
Préconditions : le client doit être authentifié
Enchaînement principal 1. L'utilisateur sélectionne un album 2. L'utilisateur sélectionne la photo désirée de l'album 3. L'utilisateur applique l'effet désiré en indiquant son numéro.
Postconditions :
Enchaînement s alternatifs :

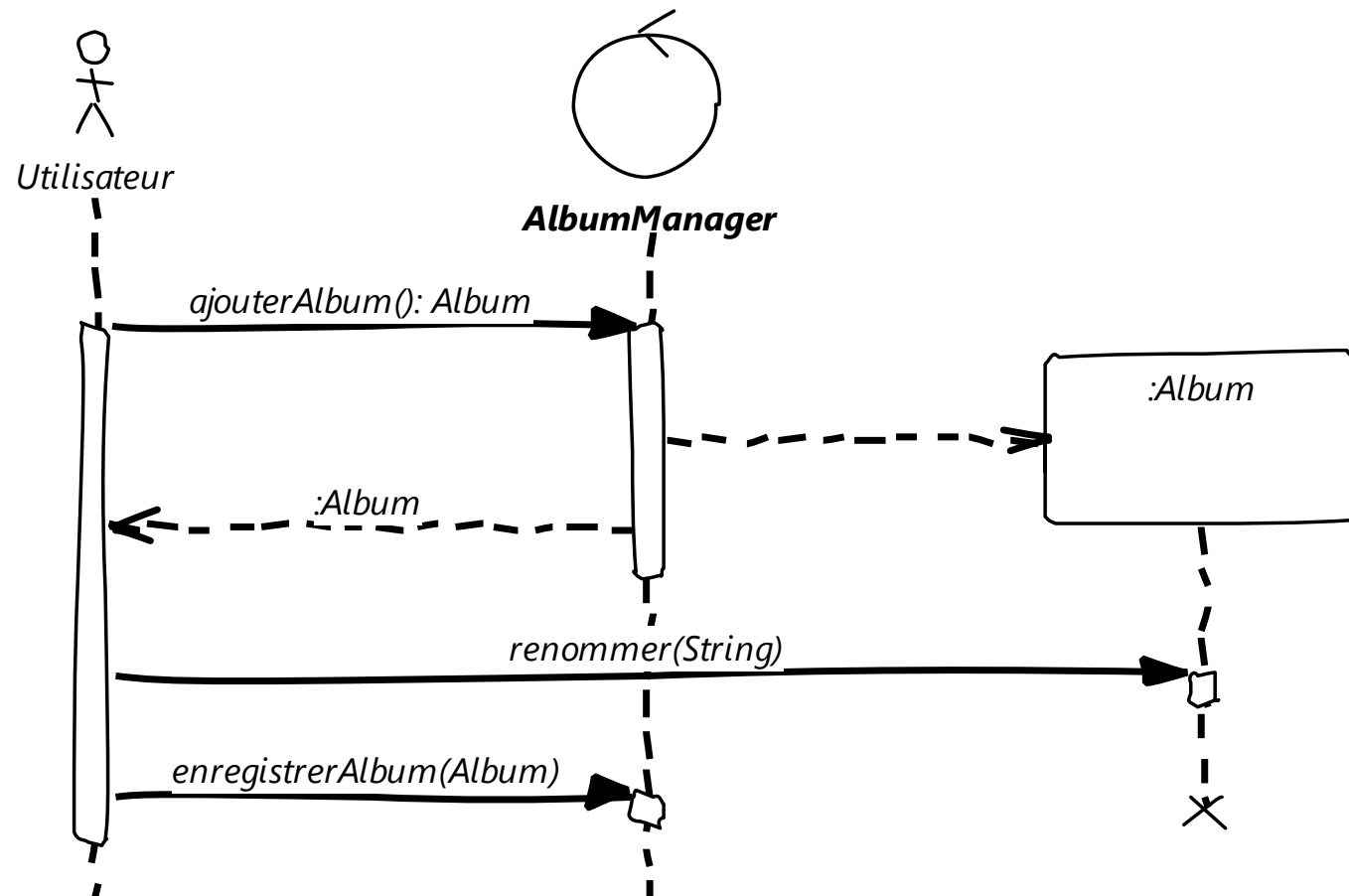
DSQ – Appliquer un effet



DSQ – Ajouter un album

CU: AjouterUnAlbum
ID: 2
Description brève : Création d'un nouvel album par l'utilisateur
Acteurs primaires : Utilisateurs
Acteurs secondaires : Aucun
Préconditions : le client doit être authentifié
Enchaînement principal <ol style="list-style-type: none">1. Le CU démarre quand l'utilisateur clique sur le bouton « nouveau »2. L'utilisateur donne un nom à l'album3. L'utilisateur enregistre l'album
Postconditions :
Enchaînement s alternatifs :

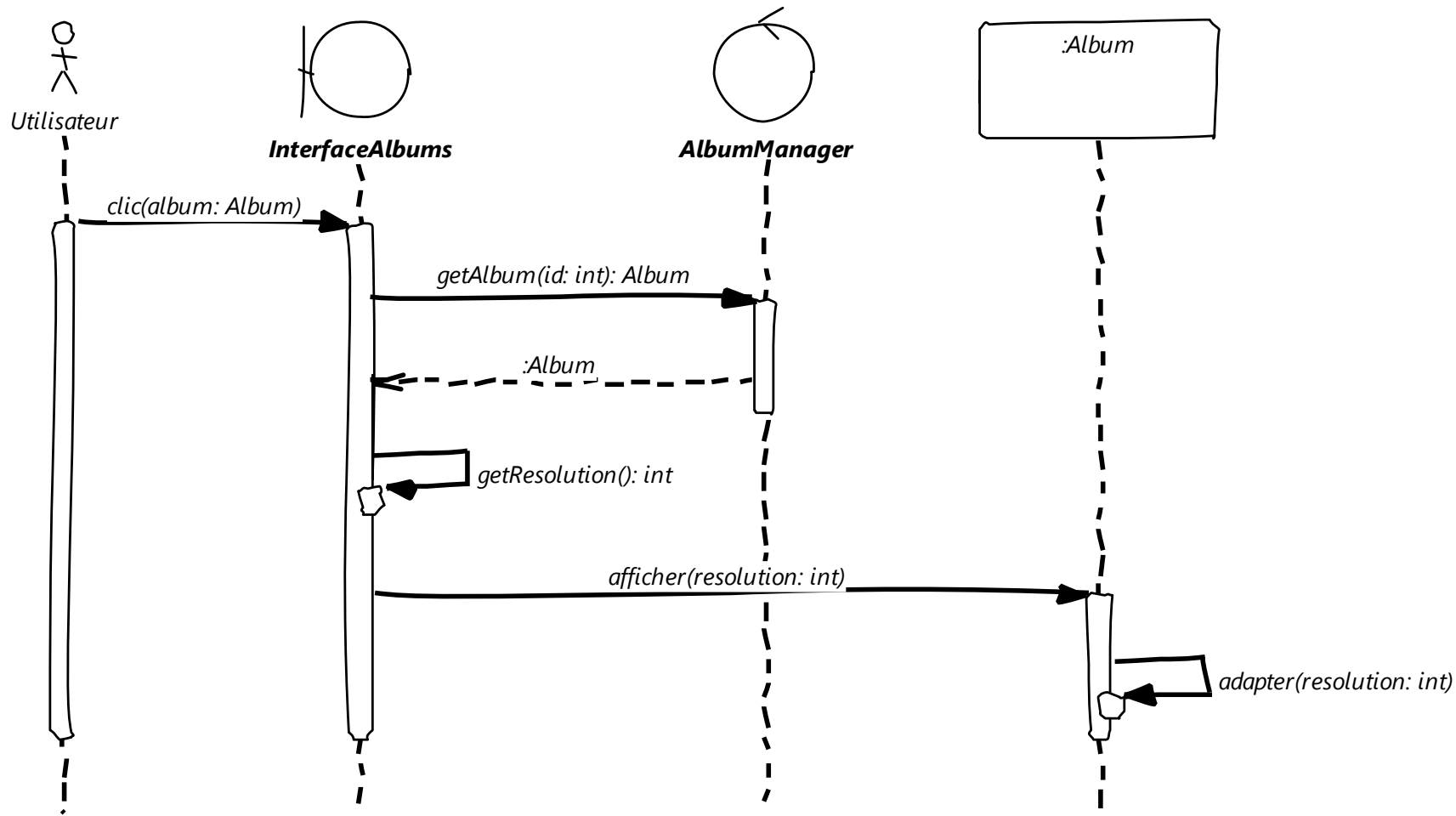
DSQ – Ajouter un album



DSQ – Consulter un album

CU: Consulter un album
ID: 3
Description brève : L'utilisateur consulte un album et les photos qui le composent
Acteurs primaires : Utilisateurs
Acteurs secondaires : Aucun
Préconditions : le client doit être authentifié
Enchaînement principal <ol style="list-style-type: none">1. Le CU démarre quand l'utilisateur clique sur l'album sélectionné2. L'album adapte son affichage selon la résolution3. L'album affiche les photos le composant selon la résolution
Postconditions :
Enchaînement s alternatifs :

DSQ – Consulter un album



Fragments

- Un DSQ peut être composé de plusieurs fragments
- Un fragment est caractérisé par un **nom**
- Un fragment peut contenir **un** ou **plusieurs** messages
- Un fragment est composé d'un **opérateur** et de un ou plusieurs **opérandes**
- L'opérateur décide comment les opérandes sont exécutées
- Un fragment peut être caractérisé par une ou plusieurs **conditions**
- Les fragments peuvent être **imbriqués**

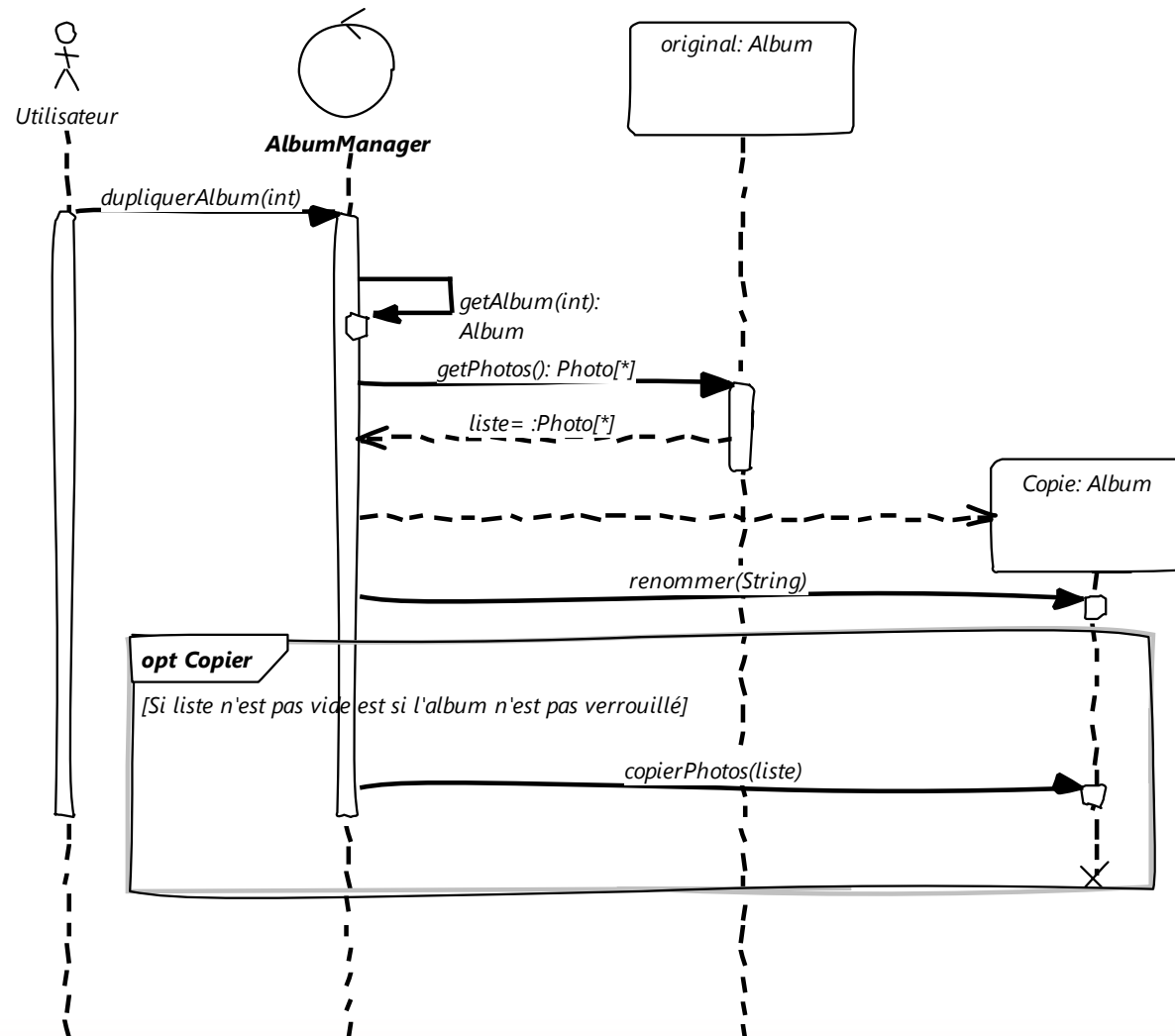
Fragments – Opérateurs

Opérateurs	Nom	Description
opt	optionnel	L'opérande n'est exécutée que si la condition est vérifiée
alt	Alternatives	Plusieurs alternatives. Uniquement l'opérande dont la condition est vérifiée s'exécute
loop	Itération	Itère l'exécution tant que la condition est vérifiée
ref	Référence	Réfère à une autre interaction

DSQ – Dupliquer un album

CU: DupliquerUnAlbum
ID: 4
Description brève : L'utilisateur crée un nouvel album portant le même nom et contenant les mêmes photos
Acteurs primaires : Utilisateurs
Acteurs secondaires : Aucun
Préconditions : le client doit être authentifié
Enchaînement principal <ol style="list-style-type: none">1. Le CU démarre quand l'utilisateur clique sur le bouton dupliquer2. Le système crée un nouvel album3. Le système donne le même nom au nouvel album4. Si l'album original contient des photos et s'il n'est pas verrouillé<ol style="list-style-type: none">4.1 Copier les photos de l'album original vers la copie
Postconditions :
Enchaînement s alternatifs :

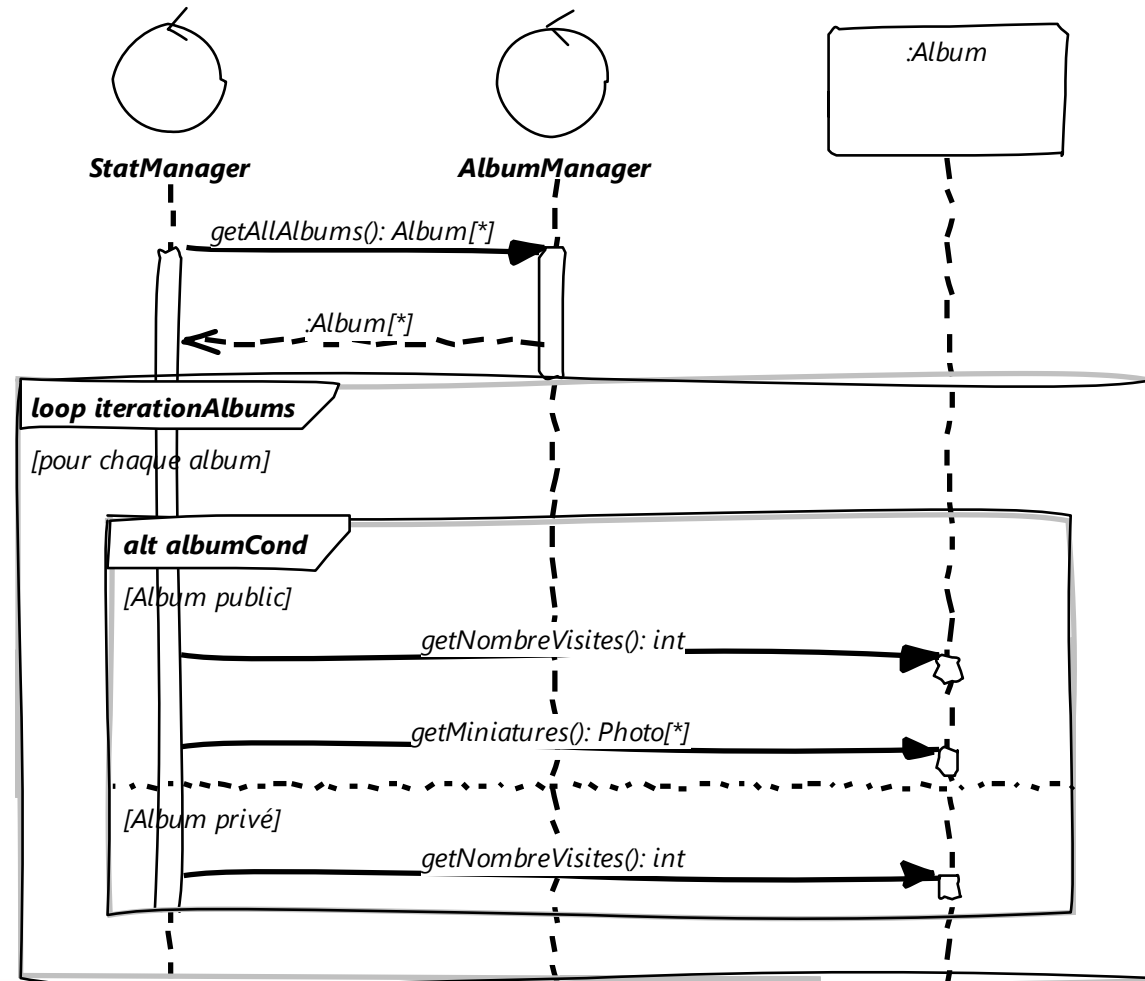
DSQ – Dupliquer un album



DSQ – MAJ Statistiques Album

CU: MAJ statistiques
ID: 5
Description brève : Chaque jour, des statistiques sont mises à jour pour tous les albums. Si l'album est public, on prend le nombre de visite et les miniatures sinon on prend uniquement le nombre de visites.
Acteurs primaires : Utilisateurs
Acteurs secondaires : Aucun
Préconditions : le client doit être authentifié
Enchaînement principal <ol style="list-style-type: none">1. Le CU démarre chaque fin de journée2. Le système récupère la liste des albums3. Pour chaque album<ol style="list-style-type: none">3.1 si l'album est public,<ol style="list-style-type: none">3.1.1 prendre le nombre de visites et les miniatures3.2 sinon<ol style="list-style-type: none">3.1.2 prendre le nombre de visites
Postconditions :

DSQ – MAJ Statistiques Album



Diagrammes de Séquence



SECTION 5, DÉBAT 05 MNS

Diagrammes d'Activité



SECTION 6

Présentation

- Les diagrammes d'activité (AD) représentent des processus en tant qu'une activité composée de plusieurs **nœuds** connectés.
- Dans UML 1, les AD étaient des sous-ensembles des diagrammes d'état. Dans UML 2, c'est des diagrammes à part entière.
- Une activité peut être attachée à n'importe quel élément du modèle (cas d'utilisation, classe, ...). Une activité modélise essentiellement un **comportement**.
- Les diagrammes d'activité sont idéaux pour représenter des **processus métier**.
- Les ADs n'ont pas besoin de connaître la structure statique du modèle (classes, objets) ce qui permet leur utilisation dans les premiers moments de l'analyse.
- Les ADs sont très simples à comprendre par le client.

Activités

- Les activités sont des réseaux composés de *nœuds* et de *connecteurs*.
- Il y a trois types de nœuds : des *nœuds d'action*, des *nœuds de contrôle* et des *nœuds d'objet*.
- Un « nœud d'action » représente une *unité de travail* atomique (indivisible)
- Un « nœud de contrôle » contrôle le flux au sein d'une activité
- Un « nœud d'objet » représente un objet manipulé dans l'activité.

Activités - Suite

- Les connecteurs représentent le *flux* entre les nœuds.
- Il y a deux types de connecteurs : des *connecteurs de contrôle* et des *connecteurs d'objet*.
- Un connecteur de contrôle représente le flux d'activité
- Un connecteur d'objet représente le flux d'objet
- Une activité démarre avec un nœud spécial appelé *nœud initial*
- L'activité se termine avec un nœud spécial appelé *nœud final*

Activités – Exemple

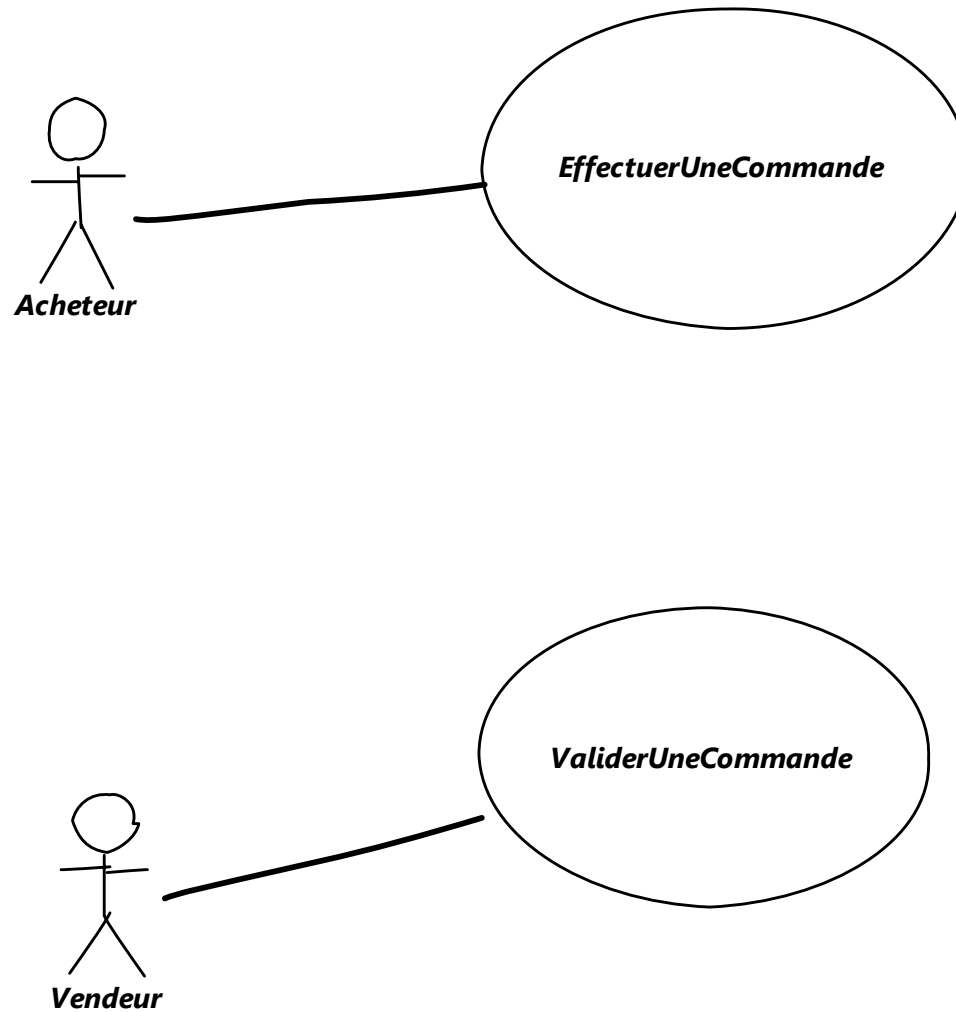
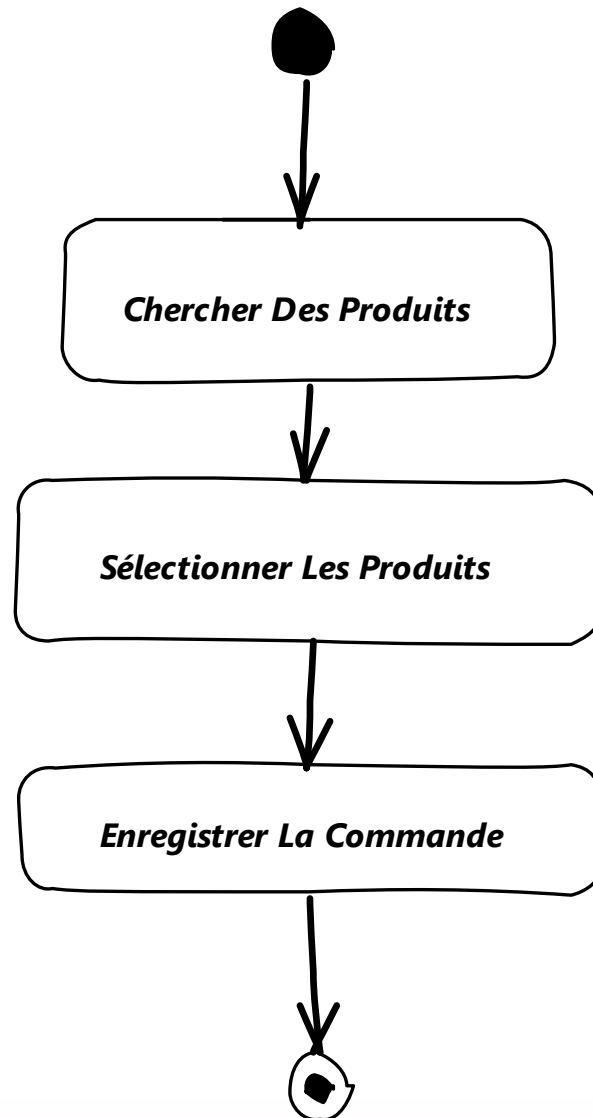


Diagramme d'activité



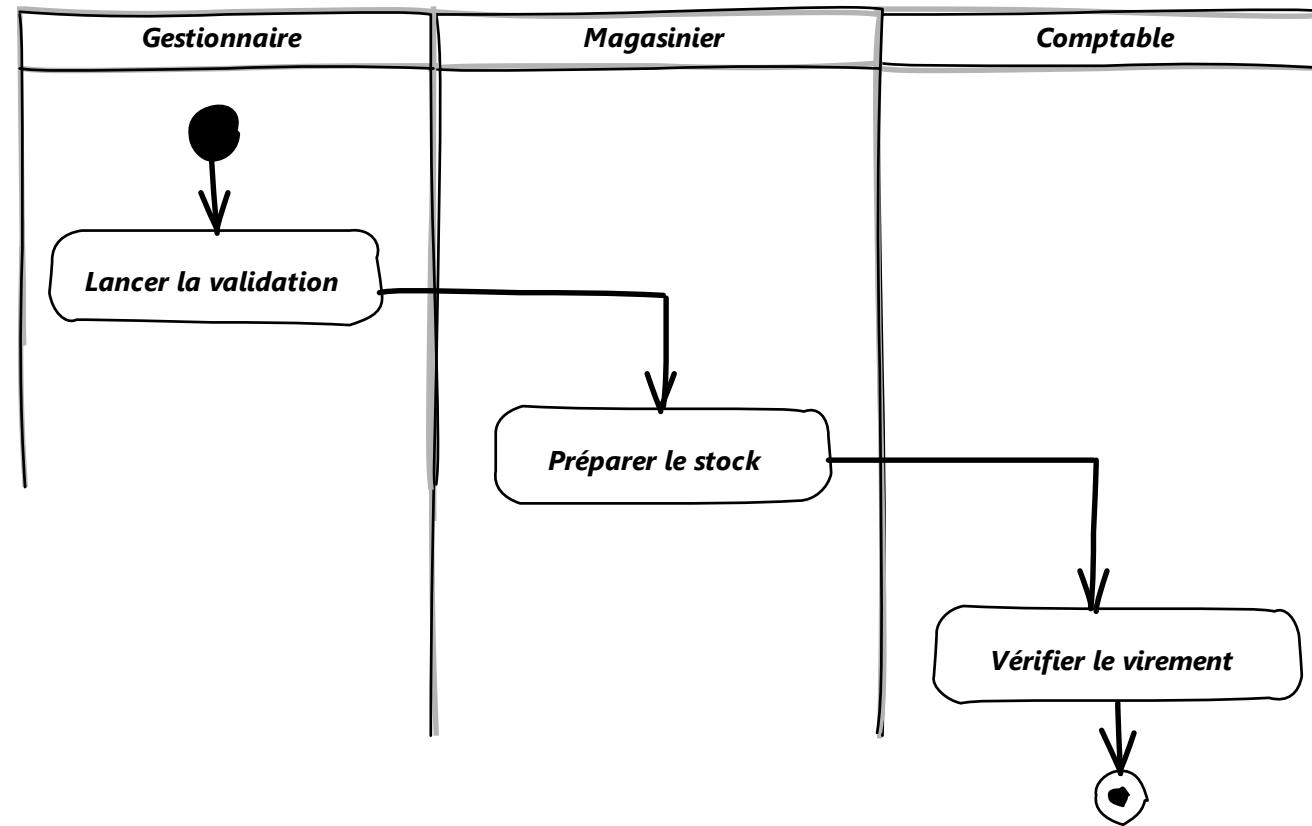
Fonctionnement

- Le diagramme d'activité suit une séquence déterminée par un *jeton*
- Le jeton transite du nœud initial vers le nœud final
- Le jeton peut être un des éléments suivants : le *flux* actuel, un *objet* ou des *données*
- L'état d'un AD est déterminé par ses jetons
- Les jetons transitent d'un nœud à l'autre via les connecteurs

Partitions

- Les activités sémantiquement liées peuvent être regroupées en « *partitions* »
- Une partition désigne généralement le rôle exécutant l'action en cours
- Les partitions rendent les ADs plus faciles à lire et plus expressifs
- Les partitions peuvent être représentées d'une manière horizontale ou verticale
- Exemple : pour valider une facture, le gestionnaire lance la validation, le magasinier prépare le stock et enfin, le comptable vérifie le paiement

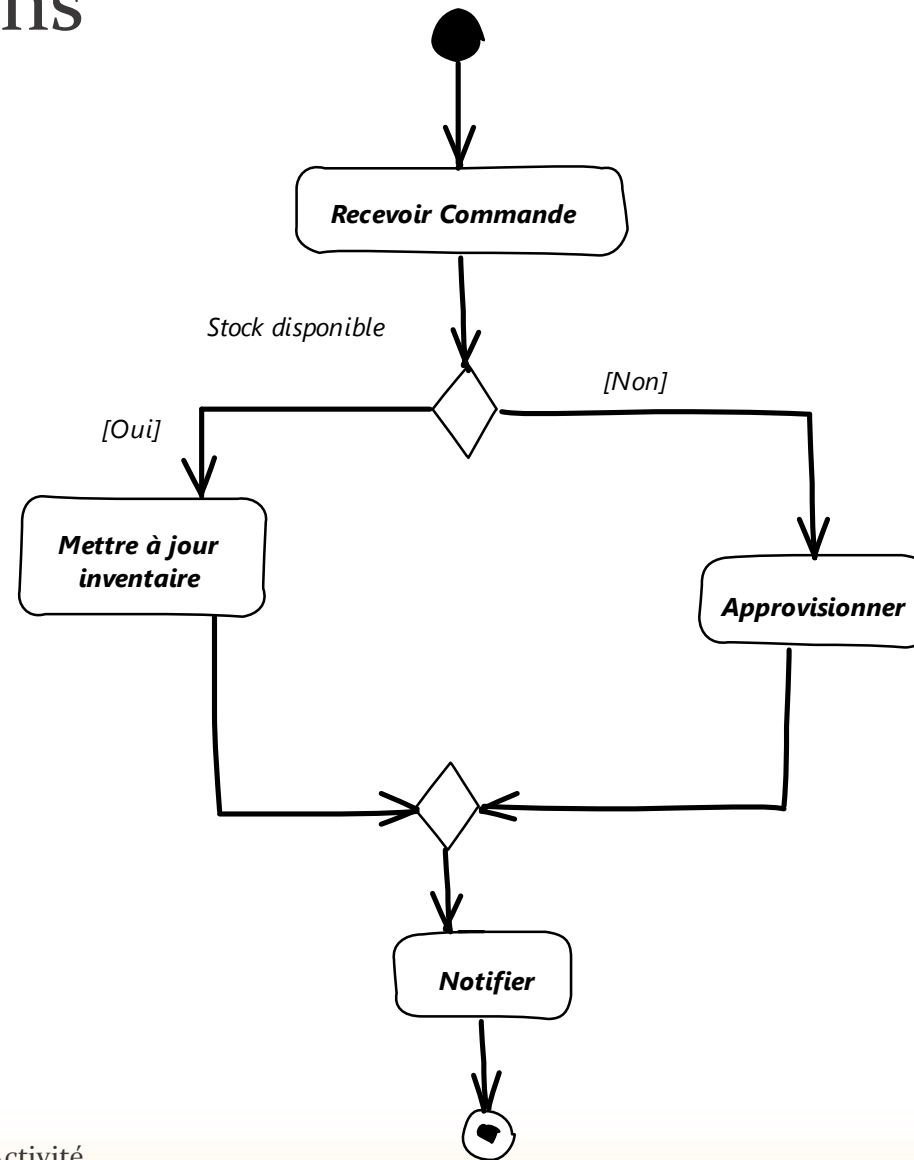
Partitions – Exemple



Décisions et fusions

- Un nœud de **décision** a un connecteur en entrée et deux (ou plusieurs e sortie)
- Le chemin à suivre par le jeton est celui de la condition vérifiée du nœud de décision
- **Plusieurs chemins** sortent du nœud de décision
- Un nœud de fusion fusionnent les connecteurs sortis d'un nœud de décision
- Exemple : une fois une commande reçue, si le stock est disponible faire l'inventaire du reste, sinon lancer un approvisionnement

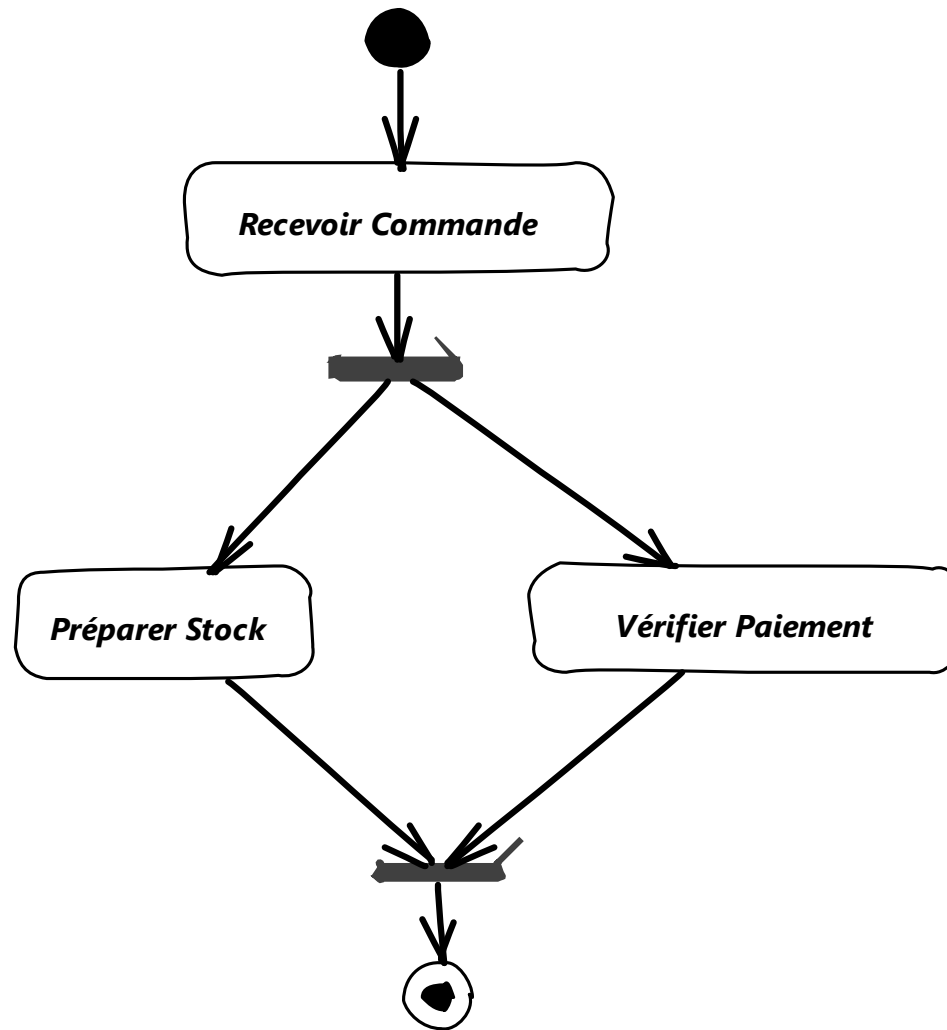
Décisions et fusions



Activités parallèles

- Dans un DA, on peut créer des flux parallèles en utilisant le nœud « fork »
- Pour fusionner les flux parallèles, un nœud « join » est utilisé
- Exemple : après réception d'une commande, deux activités sont lancées en parallèle, une pour la vérification du paiement et l'autre pour la préparation du stock

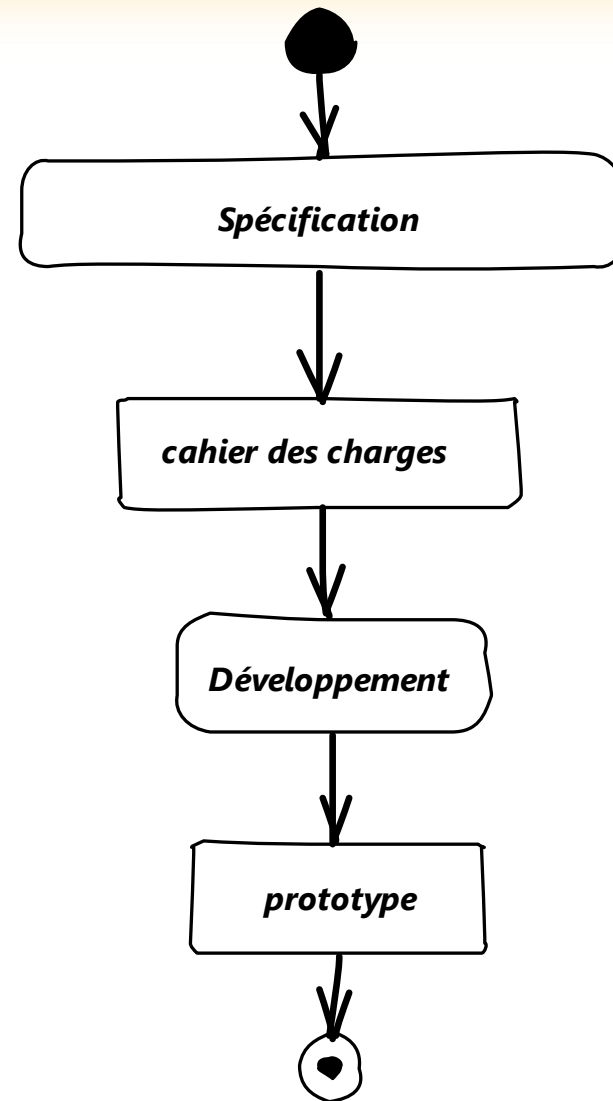
Activités parallèles



Nœuds d'objet

- Les nœuds d'objet sont des nœuds spéciaux qui indiquent que des instances d'une classe données sont disponible à un point donné du temps
- Exemple, dans une activité industrielle, la spécification produit un cahier de charges et le développement produit un prototype

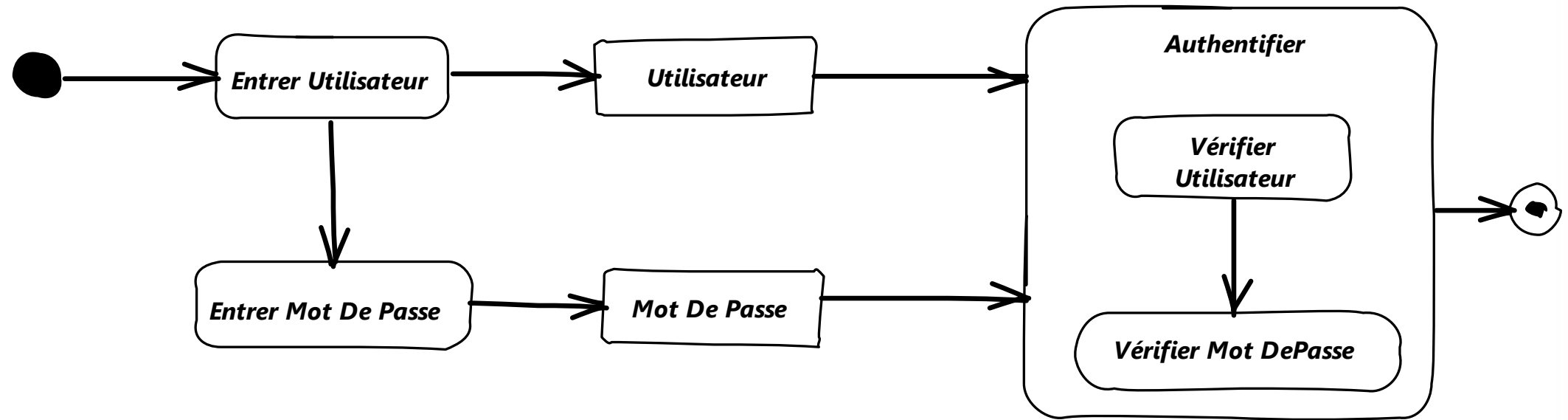
Nœuds d'objet



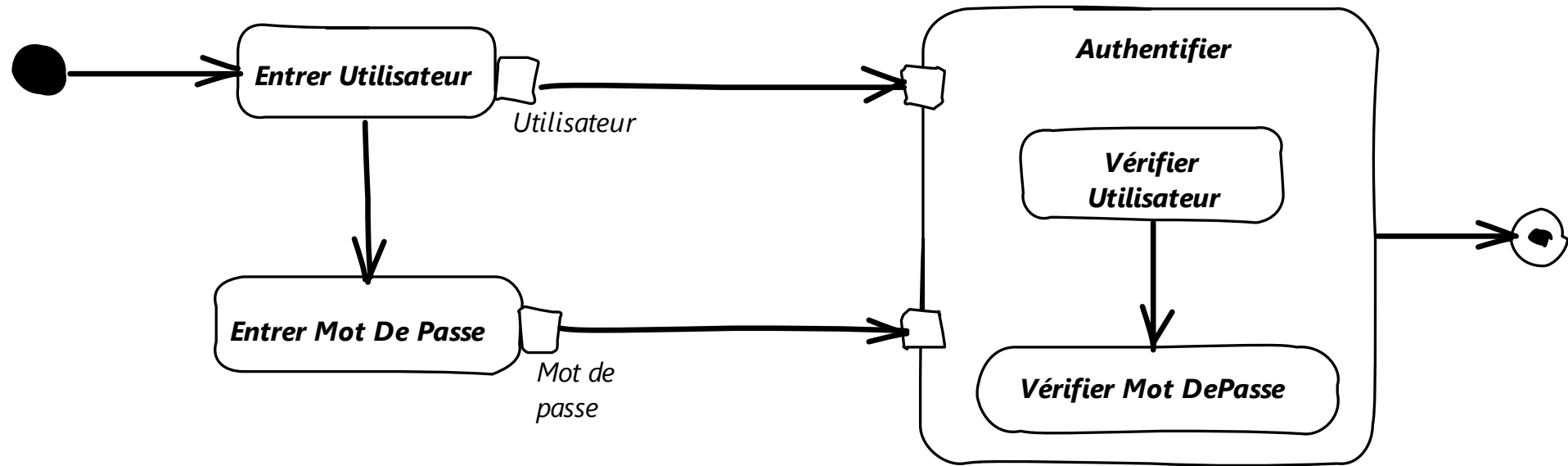
Nœuds d'objet

- Lorsque les activités produisent beaucoup d'objets, il serait intéressant de les organiser en **PINS**
- Par exemple, pour l'authentification, le username et le mot de passe sont des objets produits

Nœuds d'objet



Nœuds d'objet



Quand utiliser le diagramme d'activité ?

- Pour modéliser des processus parallèles
- Pour modéliser des processus métier indépendamment de la structure statique (classes d'analyse)

Diagrammes d'Activité



SECTION 6, DÉBAT 05 MNS

Diagrammes d'état-transition



SECTION 7

Introduction

- Un diagramme d'états-transitions (*DET*) modélise le *cycle de vie* d'un objet dit *objet réactif*
- Un objet réactif est un objet qui change d'état suite à un *évènement* donné
- Une machine d'état représente un nombre *fini* de ces états

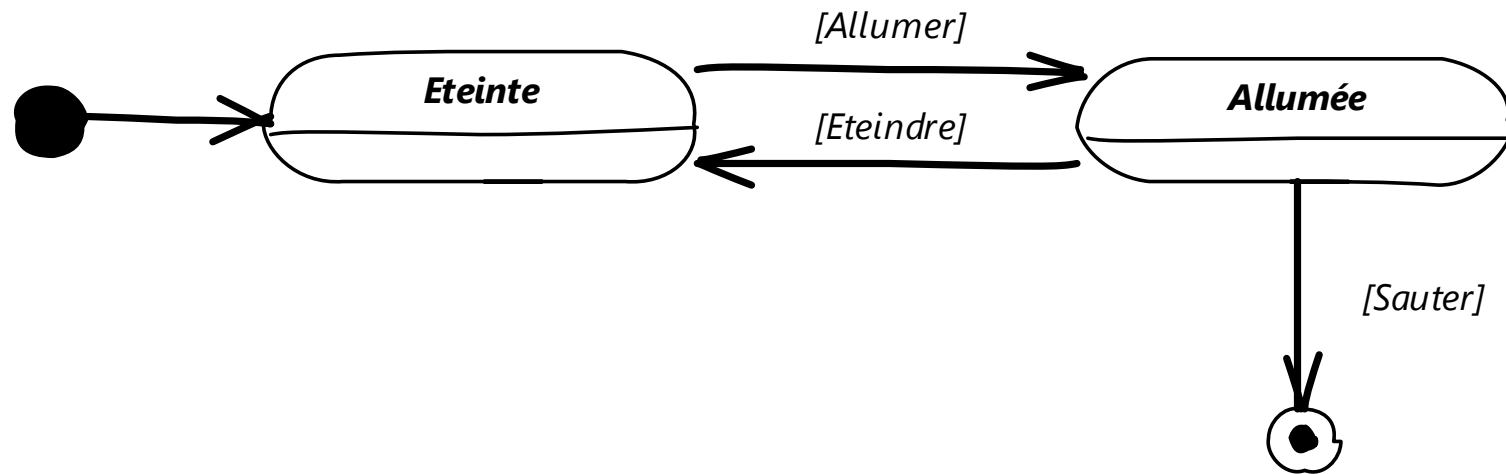
Diagrammes d'état-transition

- Les DETs sont composés de trois éléments principaux : **états**, **transitions** et **évènements**
- Un état représente **une condition** dans laquelle se trouve un objet à un instant T
- Un évènement est une **action** particulière qui se déclenche sous des conditions spécifiques
- Une transition est le **mouvement** d'un état vers un autre suite à un évènement donné

Exemple

- Une lampe est initialement éteinte. Si on l'allume, elle passe dans l'état allumée. Si on l'éteint, elle repasse dans l'état éteinte. Si elle saute, elle passe définitivement à l'état final défectueux

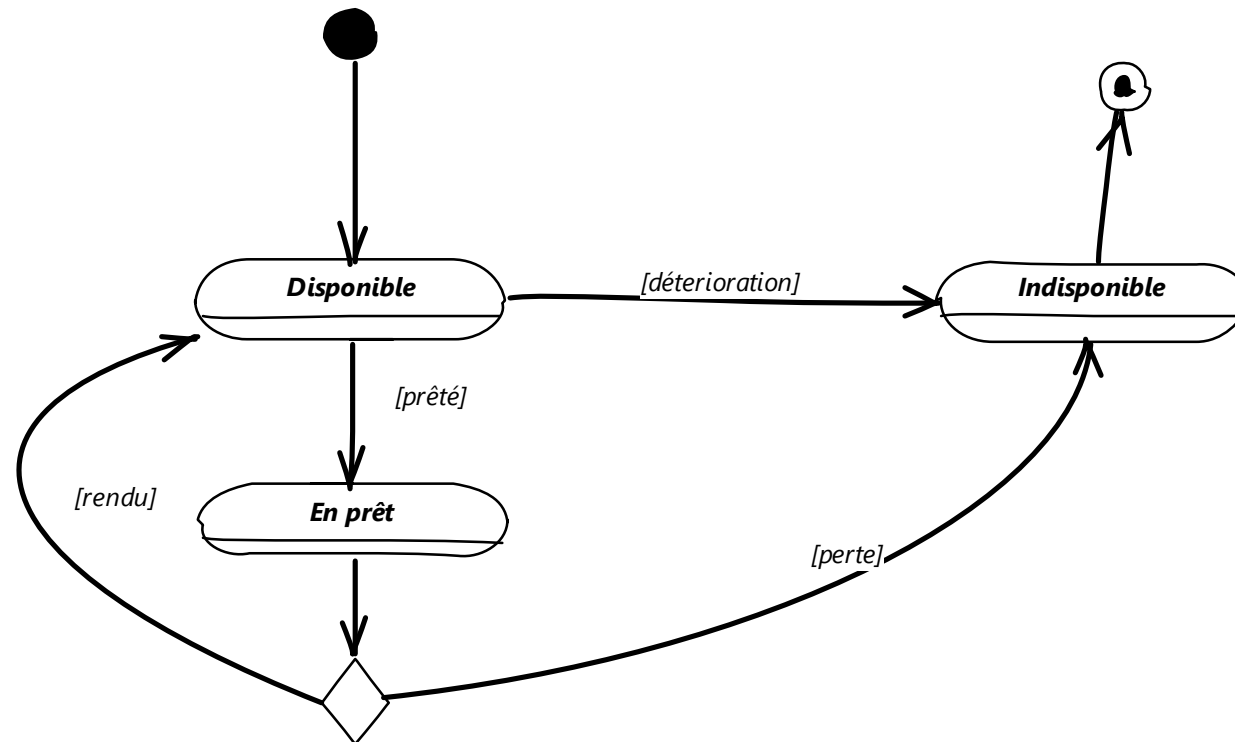
DET – Exemple



Etats

- Durant le temps, un objet peut envoyer un message à un autre objet qui peuvent changer sont état
- L'état d'un objet est déterminé par ses *attributs*, ses *relations* à d'autres objets et les *activités* dans lesquelles il est impliqué
- Il ne faut modéliser que les états qui ajoutent de l'expressivité au système
- Les états de jonction (pseudo-états) peuvent connecter plusieurs transitions
- Un pseudo-état de choix exprime un ou plusieurs choix mutuellement exclusifs

DET – Exemple



Quand utiliser le DET ?

- Pour modéliser des entités réactives (dont les états changent)

Diagrammes d'état-transition



SECTION 7, DÉBAT 05 MNS

Bibliographie

- Software Engineering Right Edition, Ian Sommerville, Addison Wesley, 2007
- Software Development and Professional Practice, John Dooley, APress, 2010
- Software Development Life Cycle (SDLC), Togi Berra, course session 2
- Rational Unified Process - Best Practices for Software
- Development Teams, IBM / Rational, 1998