

## DOSSIER DE CANDIDATURE

Soumis à

La commission Nationale de  
Recrutement des Maîtres assistants

Dans la discipline

Informatique

Cours Modélisation Objet UML  
avec un Examen Corrigé

Elaboré par :

Dr. Ghassen HAMDI

Laboratoire MARS, Université de Sousse



---

# COURS

---

## *Modélisation Orienté Objet*

### *UML*

---

**Niveau : 4<sup>ème</sup> Génie Informatique**

**Préparé Par : HAMDI Ghassen**

*Docteur en Sciences Informatique.*

**Année universitaire : 2020/2021**

## **Cours**

### **« Modélisation Objet UML »**

**Objectif(s) :**

Au terme de ce module, l'étudiant pourra découvrir les bases de modélisation orientée objet des systèmes informatiques en UML :

- ✓ Maîtriser les connaissances théoriques fondamentales de la modélisation objet,
- ✓ Utiliser les diagrammes UML pour modéliser tout ou partie d'un système,
- ✓ faire preuve d'esprit critique face à une modélisation,
- ✓ Mettre en œuvre une démarche minimaliste pour le développement d'un système informatique à travers une étude de cas.

**Programme :**

- Chapitre 1- Introduction à la modélisation objet
- Chapitre 2- Diagramme de cas d'utilisation
- Chapitre 3- Diagramme de classe
- Chapitre 4 Diagramme de séquence
- Chapitre 5- Diagramme d'état transition
- Chapitre 6- Diagramme d'activité

**Bibliographie :****Livres :**

- UML par la pratique
- Modélisation objet avec UML
- UML en action
- De Merise à UML, ...
- Cours Pr. Wassim JAZIRI, ISIMS Sfax, 2009/2010

**Sur le Web :**

- <http://www.omg.org/uml/>
- <http://www.rational.com>
- <http://www.uml.free>

# *UML : Unified Modelling Language*



## Chapter 1

Introduction à la modélisation objet



**Ghassen HAMDI**

# PLAN

1. Notion d'objet
2. UML - l'historique
3. UML - caractéristiques
4. La genèse d'UML

# Introduction à la Modélisation Orientée Objet

## **Logiciel et Matériel :**

### **Systèmes informatiques :**

- **80 % de logiciel ;**
- **20 % de matériel.**

**Pendant quelques années, seule une poignée de fabricants ont produit des marchandises.**

- **Le matériel est pleinement satisfait**
- **Le marché est uni.**

**Les problèmes informatiques sont essentiellement des problèmes logiciels.**

# Introduction à la Modélisation Orientée Objet

- ▶ La "**crise du logiciel**" a été examinée dans une étude portant sur 8 380 projets (Standish Group, 1995) avec les résultats suivants :

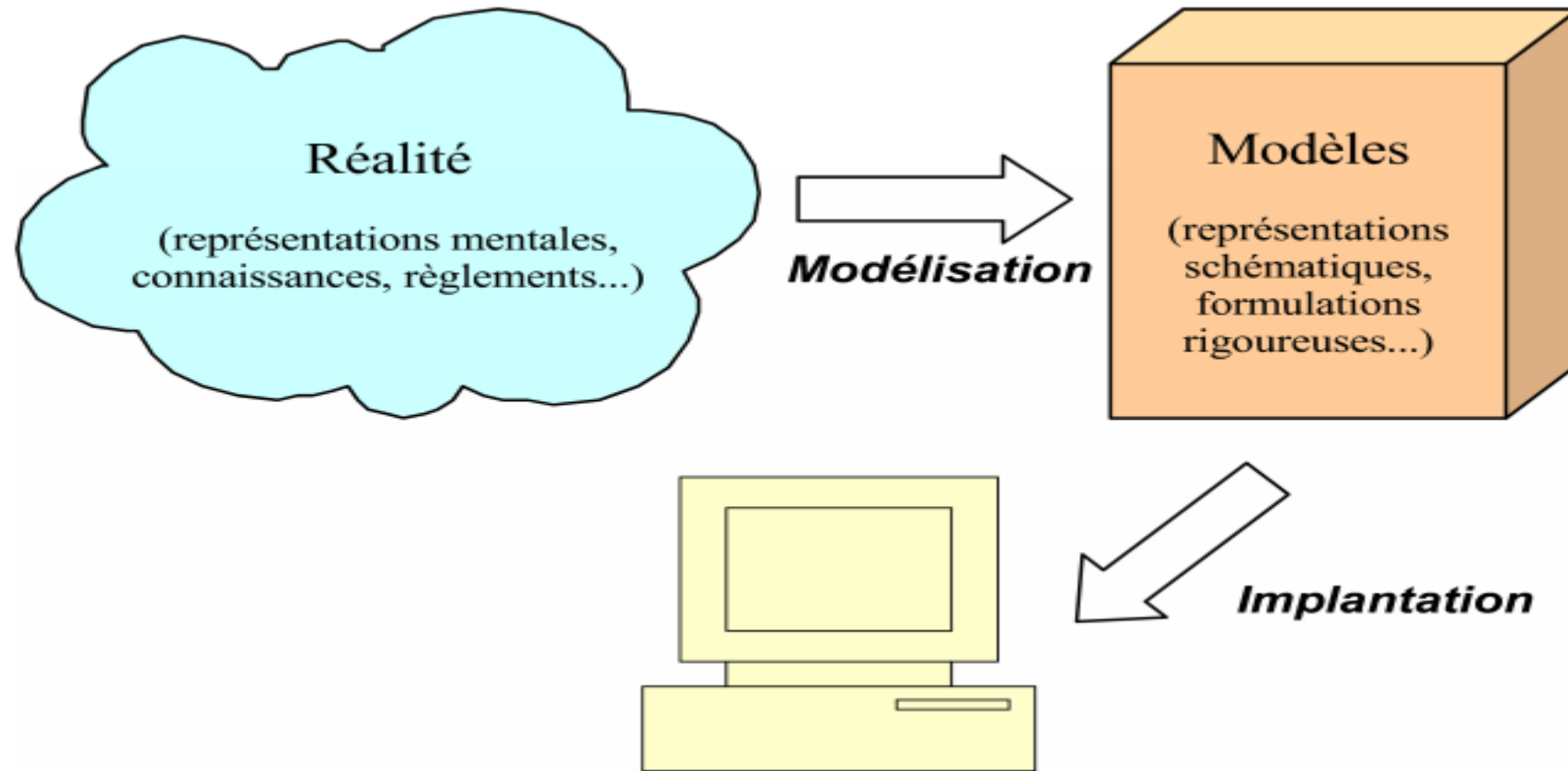
- Pourcentage de réussite : 16 % ;
- Les défis rencontrés : 53 % (Non respect du financement ou des échéances, problème de caractéristiques opérationnelles) ;
- Échec : 31 % (oublié).

**La part de réussite diminue avec la taille des projets et la taille des entreprises.**

- ▶ **Génie Logiciel (Software Engineering) :**

- Comment garantir la qualité d'un logiciel?
- Qu'attendons-nous des logiciels ? Quelles sont vos normes de qualité ?

# Introduction à la Modélisation Orientée Objet





# Introduction à la Modélisation Orientée Objet

## ► Modèle

- Un modèle se présente comme une **abstraction** de la réalité excluant certains détails du monde réel.
  - Cette simplification vise à diminuer la complexité du phénomène en éliminant les aspects qui n'influent pas de manière significative sur son comportement.
  - Le modèle articule ce que le concepteur estime être essentiel pour la compréhension et la prédiction du phénomène envisagé, tout en délimitant les frontières du phénomène projeté conformément aux objectifs du modèle.

# Introduction à la Modélisation Orientée Objet

## ► Langages de conceptualisation

- Un langage de **conceptualisation** nécessite de spécifier :
  - La signification des concepts ;
  - Une symbole utilisé pour exprimer des concepts ;
  - Des directives d'élaboration et d'emploi des concepts.
- Le secteur du logiciel compte de nombreux langages dédiés à la modélisation :
  - Accordé aux systèmes de processus(MERISE...) ;
  - Adapté aux systèmes en temps réel (ROOM, SADT...) ;
  - Décerné aux systèmes orientés objet (OMT, Booch, UML...).
- La mise en place d'un langage de modélisation nécessite la disponibilité d'outils (Atelier de Génie Logiciel).

# UML

**La réalisation d'une application peut passer par plusieurs étapes :**

**Définition des besoins**

**Analyse**

**Conception**

**Développement**

**Test**

**Validation**

**Déploiement**

**Maintenance ...**

# UML

- ▶ **Où se situe UML dans ce contexte ?**
- UML offre la possibilité de représenter toutes les étapes du développement d'une application, de l'analyse à l'implémentation, en utilisant divers types de graphes.

# UML

## ► UML : Unified Modeling Language

- Un langage de modélisation unifié
- Distinct d'un langage de programmation
- Non lié à un langage de programmation particulier, que ce soit orienté objet ou autre
- Un langage qui repose sur l'utilisation de représentations graphiques.
- Composé de plusieurs graphes (diagrammes) offrant une vision future de l'application sous différents points de vue.
- Une norme gérée par l'OMG (Object Management Group), une organisation mondiale établie en 1989 dans le but de normaliser le modèle objet.

# UML

- ▶ **Avant UML : plusieurs méthodes orientées objet (entre 1970 et 1995)**
  - Booch (présentée par Grady Booch)
  - OMT (introduite par James Rumbaugh)
  - OOSE (proposée par Ivar Jacobson)
  - OOA, OOD, HOOD...

# UML

## ► Début d'UML

- En 1995, Booch, Rumbaugh, et Jacobson amorcent le développement d'une approche unifiée, appelée Unified Method.
- En 1996, un consortium de collaborateurs est établi dans le but de collaborer à l'élaboration de la définition d'UML.
- En 1997, l'OMG officialise la standardisation de la méthode UML 1.1.

# UML

## ► Différente version d'UML

- UML 1.1 : 1997
- UML 1.2 : 1998
- UML 1.3 : 1999
- UML 1.4 : 2001
- UML 1.5 : 2003
- UML 2.0 : 2005
- UML 2.1 : 2006
- UML 2.2 : 2009
- UML 2.3 : 2010
- UML 2.4 : 2011
- UML 2.5 : 2015
- UML 2.5.1 : 2017



# UML

## ► Remarques

- 14 diagrammes à partir d'UML 2.3
- classes en deux catégories
  - 7 Diagrammes de structure (statiques) : Ils offrent une description de la configuration du système selon diverses perspectives (classes, composants, nœuds, objets, packages, etc.).
  - 7 Diagrammes de comportement (dynamique) : Ils servent à dépeindre les divers aspects du fonctionnement du système, que ce soit du point de vue temporel, des changements d'état, etc.

# La Genèse d'UML

- ▶ UML 2.0 propose 13 diagrammes répartis en trois catégories :
  - ▶ Les diagrammes de structure ou structurels : classes, objets, composants, déploiement, structure composite et packages.
  - ▶ Les diagrammes de comportement ou comportementaux : cas d'utilisation, états-transitions et activités.
  - ▶ Les diagrammes d'interactions (sous-catégories des diagrammes de comportement) : communication, séquence, diagramme global d'interaction (interaction overview), diagramme de temps.

# La Genèse d'UML

- ▶ UML 2.0 propose 13 diagrammes répartis en trois catégories :
  - ▶ **De cas d'utilisation** : décrit les vues utilisateur et les fonctionnalités système requises par ses futurs utilisateurs,
  - ▶ **De classes** : décrit la structure du système en termes de couches et la relation entre elles,
  - ▶ **D'objets** : représente des instances de diagrammes de classes (diagrammes de collaboration simplifiés sans représentation de la livraison des messages).
  - ▶ **D'états-transitions** : décrit les cycles de vie (comportements) des objets d'une classe, les conditions (et les transitions d'état) par lesquelles passe un objet au cours de son cycle de vie,
  - ▶ **De collaboration** : Décrit l'interaction entre les objets pour réaliser chaque fonction (CU) ,

# La Genèse d'UML

- ▶ **De séquence** : représente un schéma de coopération dans l'ordre chronologique. (Où nous précisons la durée et l'heure d'envoi et de réception ),
- ▶ **D'activités** : décrivez le comportement de la méthode CU,
- ▶ **De composants** : ou l'architecture des composants physiques (logiciels) de l'application. Il décrit les **interdépendances** des compositions ou **l'implémentation** entre les différents modules qui **constituent** le logiciel. ,
- ▶ **De déploiement** : Le matériel décrit la répartition des composants physiques de l'appareil.

# La Genèse d'UML

- ▶ **Package diagram** (Diagramme de modules ou paquets) : Il présente de manière logique l'organisation du modèle et les relations entre les packages.
- ▶ **Composite structure diagram** (Diagramme de structure composite/ d'architecture) : l'illustration de l'ajustement de détails, de connecteurs et de ressorts pour représenter l'organisation interne d'un élément statique complexe.
- ▶ **Interaction overview** (Diagramme de vue d'ensemble des interactions) : Utilise des diagrammes d'activité et des séquences afin de relier des parties d'interaction avec les décisions et les flux.
- ▶ **Timing diagram** (Diagramme de temps) : Il associe des diagrammes d'état et de séquence pour illustrer comment l'état d'un objet évolue au fil du temps et les messages qui influencent cette évolution.

# Pour conclure

- ▶ Les différents types de diagrammes UML combinés :
  - ▶ *Fournit un aperçu complet des aspects statiques et dynamiques du système.*
- ▶ L'UML ne requiert pas de méthode de travail spécifique :
  - ▶ *Il peut être inclus dans tout processus de développement logiciel.*
  - ▶ *En utilisant UML, vous pouvez améliorer votre façon de travailler progressivement sans changer votre façon de travailler.*

# Bibliographie

## ► Livres :

- UML par la pratique
- Modélisation objet avec UML
- UML en action
- De Merise à UML, ...

## ► Sur le Web :

- <http://www.omg.org/uml/>
- [http:// www.rational.com](http://www.rational.com)
- <http://www.uml.free>

# *UML : Unified Modelling Language*

## Chapter 2

Diagramme de cas d'utilisation



**Ghassen HAMDI**



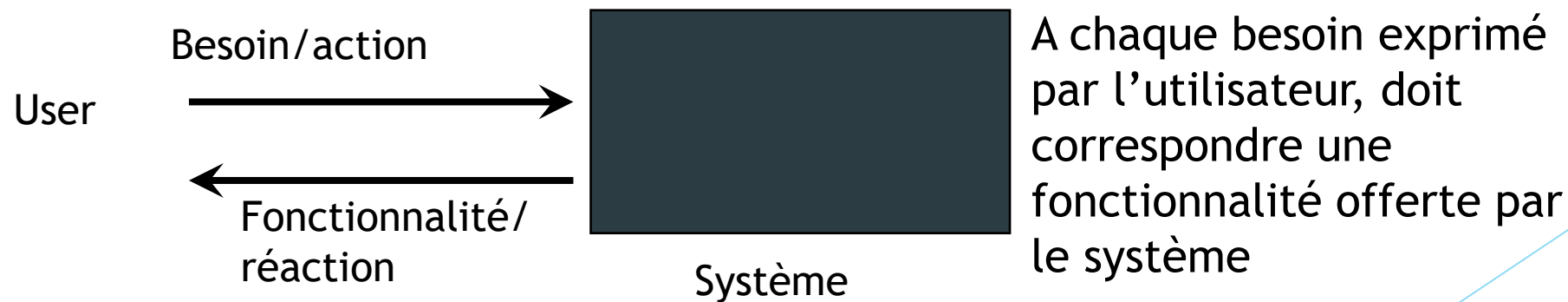
# Introduction

- ▶ Pour représenter et formaliser les besoins, il est indispensable de connaître les besoins que le système doit satisfaire lors de la phase d'analyse des besoins.
- ▶ Les moyens utilisés par UML pour modéliser ces besoins sont le diagramme de cas d'utilisation, qui représente l'organisation générale de la manipulation du système par les utilisateurs.

# Introduction

- ▶ Constat : Le système **subsiste** pour **aider** ses **utilisateurs**
- ▶ Idée : D'un point de vue de l'utilisateur, on peut décrire le **comportement du système**.

**Comportement = {Actions}+{Réactions}**



# Introduction

## ► Utilités des cas d'utilisation :

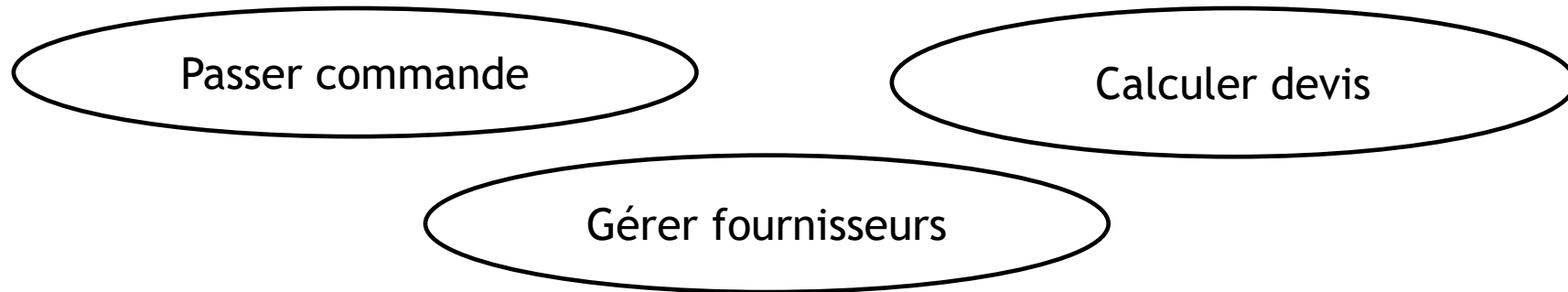
- **Focaliser** L'articulation des exigences concernant les **utilisateurs**.
- Éviter de regarder des spécifications non pertinentes et inutiles (ne convergeant pas dans le détail)
- ⇒ Approfondir sa connaissance des limites (domaine scientifique) et des exigences (fonctionnalités requises) du système

# Définitions

## Définition :

Un **cas d'utilisation** présente une série d'actions exécutées par le système, délivrant ainsi un résultat **observable** pour un acteur donné.

## Symbole des cas d'utilisation:



- ❑ En fait, le nom d'une UC est une petite phrase verbale active qui démontre une action trouvée dans le vocabulaire du système modélisé.
- ❑ Les CU sont excités par des acteurs

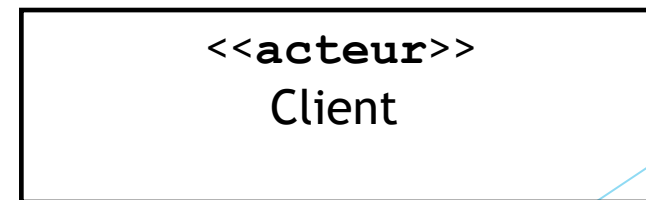
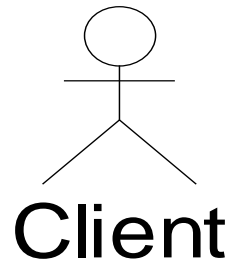
# Définitions

## ► Définition

Un acteur présente **un rôle** exécuté par une **entité externe** (utilisateur humain, dispositif matériel ou autre système) qui a un **impact direct sur le système étudié**..

- Un acteur est une entité externe (au système) reflétant sur le système et communiquant avec lui en :
  - Transmettre des données et interagir avec ce système
  - Interroger ou changer son état

## Symbole



# Définitions

- ▶ Un acteur peut occuper différentes positions :
  - ▶ Principal :
    - ▶ **Exécute** chaque fonction principale du système
  - ▶ Secondaire :
    - ▶ Exerce des tâches de maintenance ou administratives
  - ▶ Périphérique externe : ce sont les dispositifs matériels :
    - ▶ En plus des ordinateurs exécutant le programme
    - ▶ qui fait partie du domaine du programme, et
    - ▶ qui est requis pour exécuter le système
  - ▶ Systèmes externes (matériels et logiciels) avec qui le système étudié doit communiquer.

# Acteurs Vs Utilisateurs

- ▶ Il ne faut pas confondre **acteur** (=rôle) et **personne** qui adopte le système :
  - ▶ il est tout à fait possible pour une personne d'occuper plusieurs rôles.
  - ▶ il est important de comprendre qu'un rôle peut être joué par plusieurs personnes.
  - ▶ Il n'est pas obligatoire que l'acteur soit une personne physique.

# Exemple

## ► Illustration :

Situation d'utilisation : Prenons l'exemple du passage d'un client à la caisse.

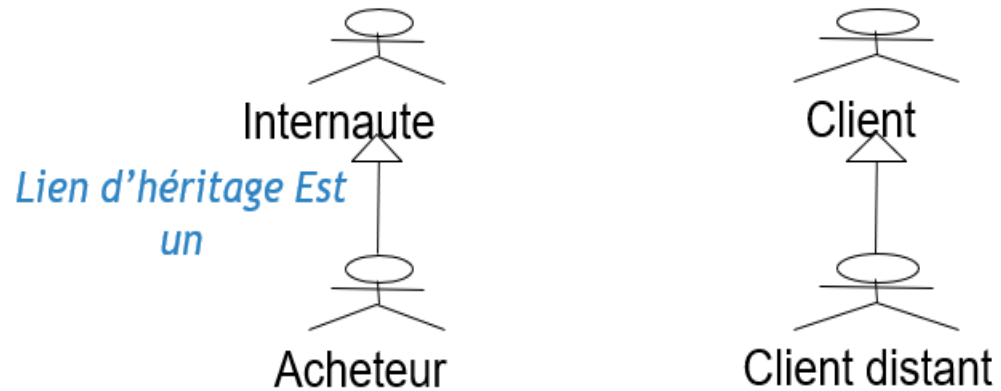
Les acteurs de ce cas d'utilisation sont :

- Principal :
  - Caissier
- Secondaire :
  - Gestionnaire des caisses (celui qui les RAZ)
- Systèmes externes :
  - Lecteur de carte bancaire (peut être qualifié d'appareil externe si et seulement si la caisse enregistreuse ne fonctionne qu'avec des lecteurs de carte : n'accepte pas les chèques ni les espèces)
  - Les autres systèmes auxquels le système spécifié doit interagir :
    - Système de gestion des stocks
    - Système d'autorisation des paiements, ...



# Relations entre acteurs

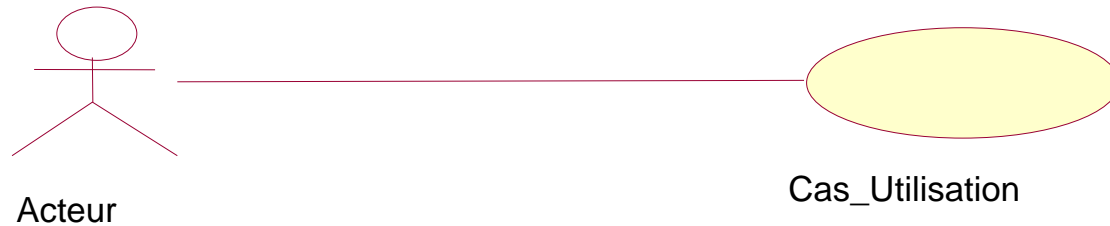
- Un acteur peut jouer un rôle dans la création de relations de généralisation avec d'autres acteurs.



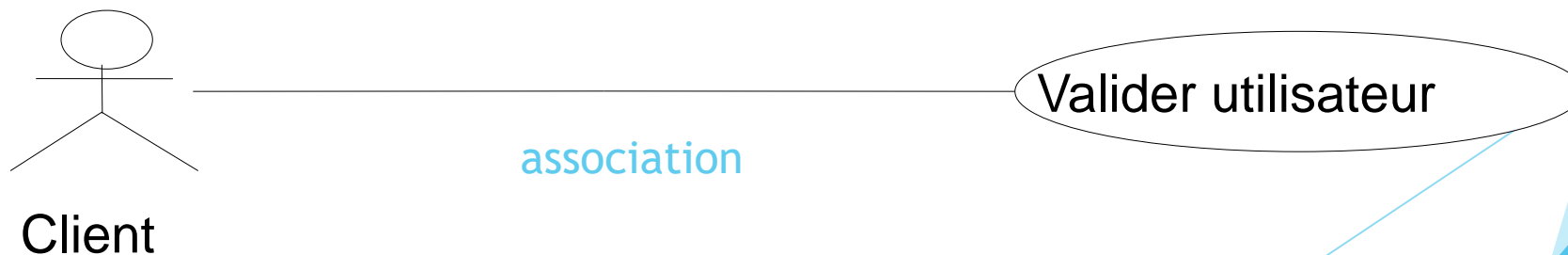
# Relation entre acteur et CU

## ▶ Acteur et Cas d'utilisation

- ▶ Association : lien entre un acteur et un cas d'utilisation
- ▶ Un acteur provoque un cas d'utilisation :

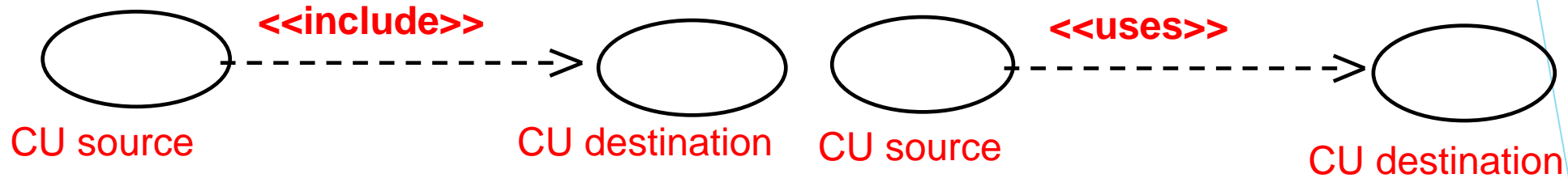


- ❑ Les acteurs sont associés à des cas d'utilisation uniquement par des associations qui décrivent quels acteurs et UC s'expriment, pouvant respectivement transférer et stocker des messages.



# Relations entre CU

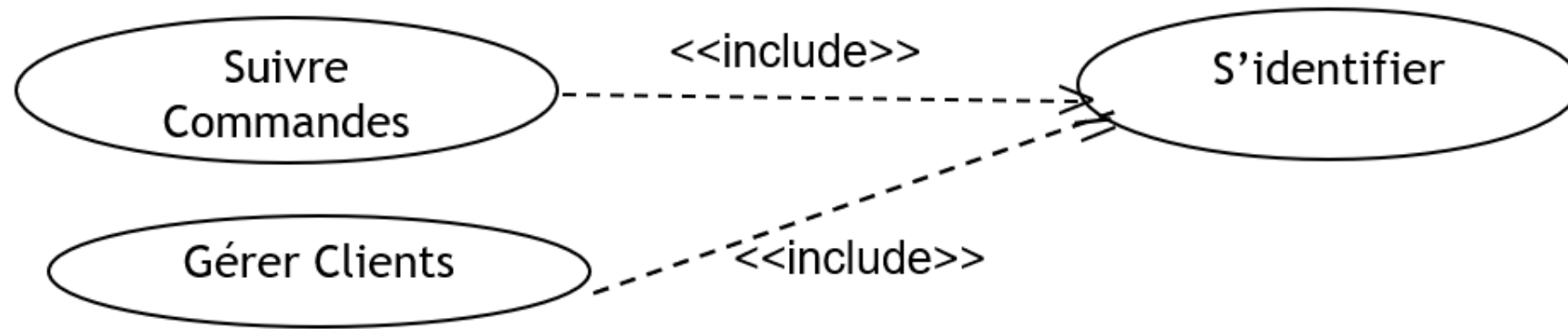
## ► Relation d'inclusion (include, uses):



- Le CU **source** **joindre** le comportement représenté par le **CU destination** en un point d'insertion désigné.
- **CU destination** est une partie obligatoire de CU **source** et on lit CU **source** *inclut* **CU destination** (dans le sens de la flèche).

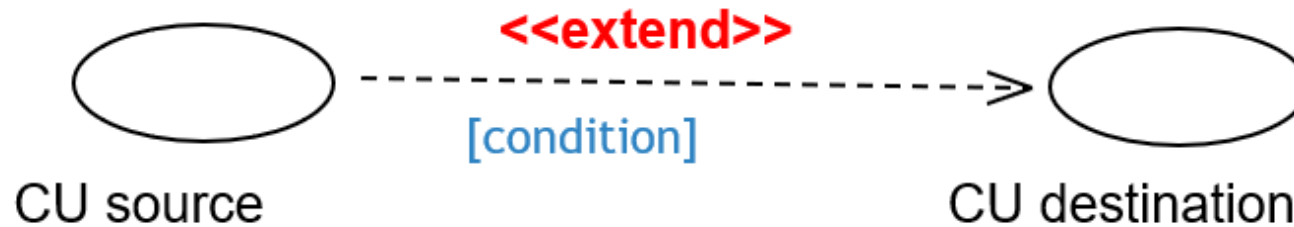
# Relations entre CU

## ► Exemple :



# Relations entre CU

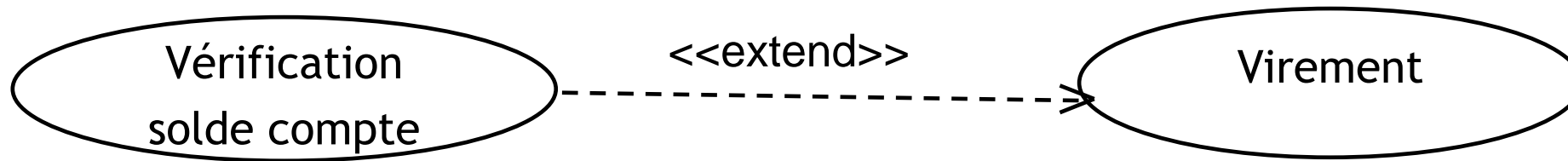
## ► Relation d'extension :



- La relation **extends** a pour fin la modélisation d'une extension à un CU.
- Il est possible d'imposer une condition à l'extension:
  - La condition d'extension est signalée à proximité du mot-clé **<<extend>>**
- Ce ratio est utilisé pour indiquer que le cas d'utilisation d'origine (l'origine de la flèche) n'est pas la base du cas d'utilisation principal actuel, mais peut l'être dans certaines circonstances.

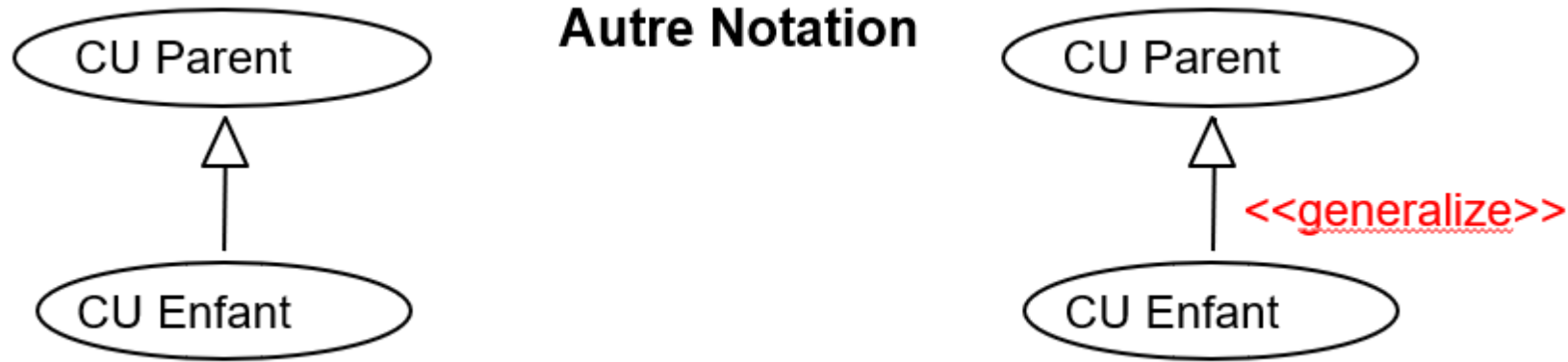
# Relations entre CU

## ► Exemples :



# Relations entre CU

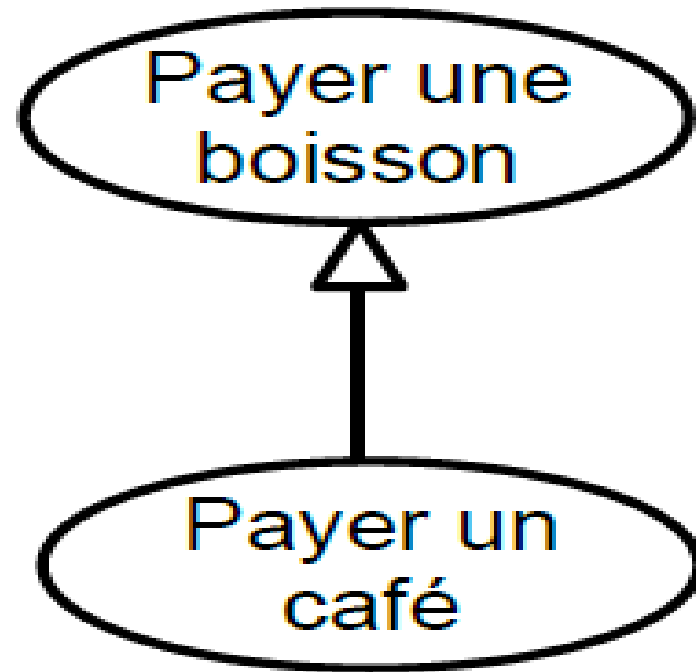
## ► Relation de généralisation (generalize):



- Le comportement du CU parent est transmis au CU enfant.
- Le comportement du CU parent peut être complété ou remplacé par le CU enfant.

# Relations entre CU

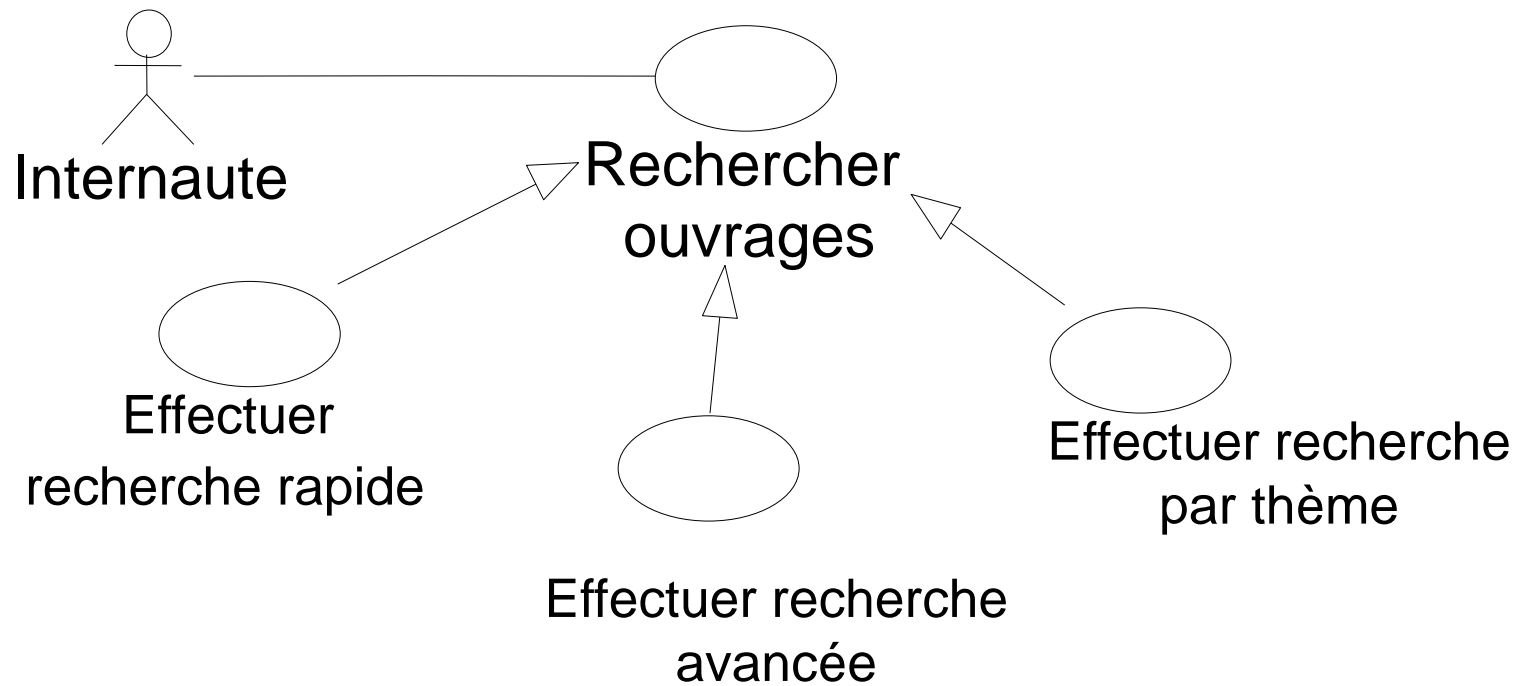
## ► Exemple 1:





# Relations entre CU

## ► Exemple 2:

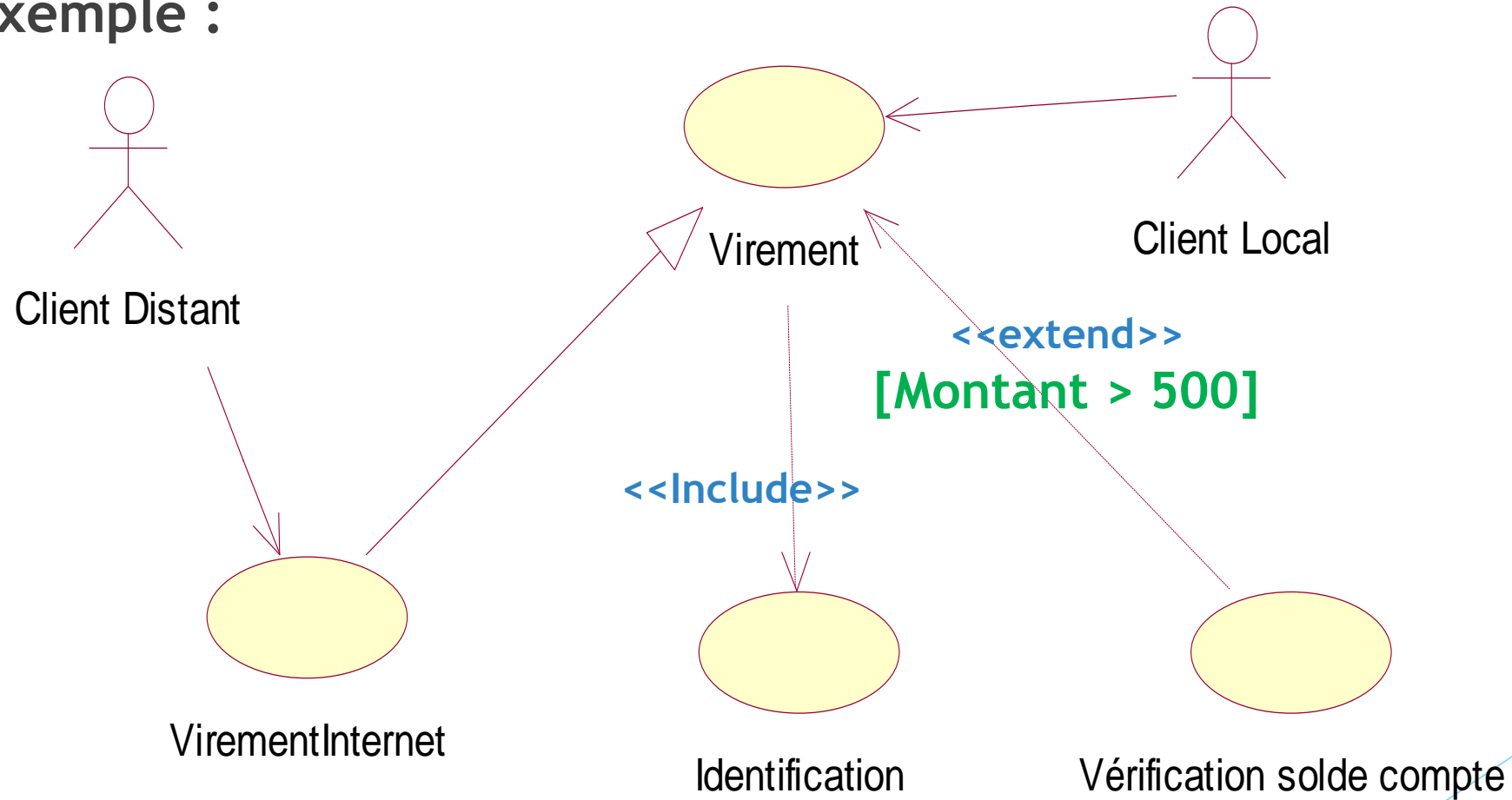


# Relations entre CU

- ▶ **Exemple** : Examinons les principes de gestion suivantes :
  - ▶ des clients locaux ou distincts
  - ▶ peuvent effectuer des virements interurbains en ligne.
  - ▶ Le transfert par Internet est un cas particulier de transfert
  - ▶ Tous les types de transfert nécessitent l'identification du client
  - ▶ En termes de montant de transfert supérieur à 500, c'est une promesse de vérification de solde du compte du client.
  
- **Question** : Donner le diagramme de CU correspondant à ces règles

# Relations entre CU

## Exemple :



# Cas d'utilisation et scénarios

- ▶ **Une illustration graphique des CU :**
  - ▶ donne un aperçu (sans détails) des différentes fonctionnalités du système.
  - ▶ peuvent être affinés à partir de scénarios
- ▶ Un CU est un ensemble de scénarios de réussite ou d'échec qui expriment comment un acteur spécifique met en œuvre le système afin d'accomplir un but
- ▶ CU → ++ scénarios : **de succès** (CU se produit) ou **d'échec** (CU ne se produit pas)
- ▶ **Définition**

Un scénario est une ordre spécifique d'échanges entre plusieurs acteurs et un système.

# Cas d'utilisations et scénarios

- ▶ Chaque scénario se compose d'étapes pouvant appartenir à l'une des trois catégories :
  - ▶ Message du sujet au système
  - ▶ Confirmation ou modification de l'état du système
  - ▶ Message du système au sujet
- ▶ Dans la présentation d'un CU vous pouvez voir :
  - ▶ Scénario nominal (optimal) : Un scénario qui atteint l'UC (objectif de l'acteur) avec le chemin le plus direct vers le succès.
    - ▶ Il assure la description l'interaction la plus courante.
  - ▶ Chaque extension (scénario dérivé) qui combine tous les autres scénarios de réussite (alternative : réussites) ou d'échec (exception).

# Spécification détaillée des cas d'utilisation

## ► En résumé

- Un scénario nominal : CU se déroule conformément aux attentes de l'utilisateur.
- Un scénario alternatif : CU se déroule de manière différente de ce qui était prévu par l'utilisateur
- Un scénario d'échec : le CU ne se réalise pas

# Spécification détaillée des cas d'utilisation

- ▶ **Nom du cas d'utilisation**
- ▶ **Acteurs**
- ▶ **Objectif**
- ▶ **Préconditions**
- ▶ **Postcondition**
- ▶ **Scénario nominal**
- ▶ **Extensions**
  - ▶ Scénario alternatif
    - Variantes et autres cas d'erreurs dans le déroulement du cas d'utilisation
  - ▶ Scénario d'échec

# Spécification détaillée des cas d'utilisation

- ▶ **Nom cas d'utilisation :** Authentification
- ▶ **Objectif :** Dans ce cas **d'utilisation donne l'occasion à l'utilisateur** d'accéder au menu principal à l'aide de son identifiant et de son mot de passe. Il contient toutes les informations pertinentes.
- ▶ La présence d'un médecin est obligatoire.
- ▶ **Post Condition :** Vérification réussie et accès direct au menu principal de l'application.



# Spécification détaillée des cas d'utilisation

Les opérations effectuées par l'utilisateur	Les opérations effectuées par le système
1. L'acteur lance l'application.	
	2. Le système lui fournit un formulaire d'authentification et lui demande un nom d'utilisateur et un mot de passe.
3. L'acteur saisie son identifiant et son mot de passe.	
	4. Le système vérifier la validité des informations saisies.

# Spécification détaillée des cas d'utilisation

- ▶ **Scénario alternatif** : représente les différentes interactions alternatives pour l'utilisateur / système.
  - Identifiant et mot de passe incorrects : le système indique, après vérification des informations à l'étape 4, à l'utilisateur que les informations saisies sont incorrectes et affiche à nouveau le formulaire d'authentification.
  - Le scénario nominal recommence à l'étape 3.
- ▶ **Scénario d'échec** :
  - L'acteur oublie son mot de passe ou bien son nom d'utilisateur .

# *UML : Unified Modelling Language*

## Chapter 3

### Diagramme de classe



hamdighassan@gmail.com

**Ghassen HAMDI**

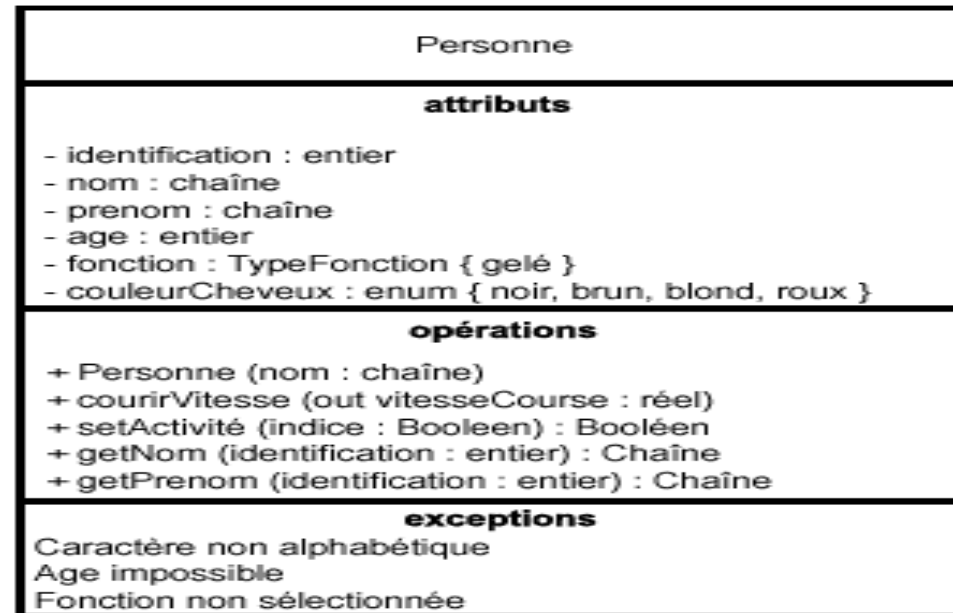
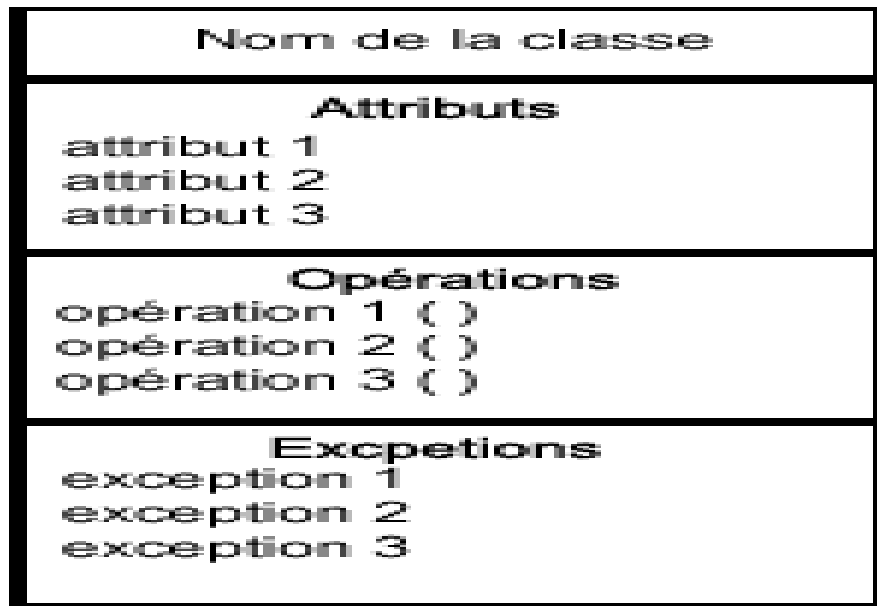
Academic Year: 2021/2022

# Sémantique

- ▶ Le diagramme de classes donne une représentation statique.
- ▶ Un ensemble de composants de modélisation statique tels que des classes et des packages qui servent à représenter la structure du modèle.
- ▶ Permet au concepteur de visualiser les relations entre les classes dans un système.
- ▶ Pour les modèles complexes, plusieurs diagrammes de classes complémentaires peuvent être construits.

# Représentation

- ▶ Les rectangles avec des séparations (sections) sont utilisés pour représenter une classe avec UML.
- ▶ Chaque section regroupe un ensemble d'éléments de même genre : Attributs, Opérations, Exceptions, etc.
- ▶ Possibilité d'ajouter d'autres sections : Responsabilité, ...



# Représentation

## ► Exemple :

**Classe = propriétés + méthodes + instantiation (constructeur)**

Scénario d'échec



GAB
Attributs
Opérations
<b>Exceptions</b> Carte non valide Code secret incorrect

# Syntaxe des attributs

- ▶ En dérivant un **attribut (/Attribut)** :
  - ▶ on peut déduire un attribut en appliquant une formule sur d'autres attributs.
  - ▶ Cette opération conduit à l'implémentation d'une opération.

RECTANGLE
Longueur
Largeur
/Surface
Opérations

Niveau Analyse

Note

Surface = longueur \* largeur

RECTANGLE
Longueur
Largeur
Surface ()

Niveau Conception

# Syntaxe des opérations

- Analyse de la visibilité et de l'impact des attributs et des opérations :

CLASSE
+ <u>Attributpublic</u> # <u>Attributprotégé</u> - <u>Attributprivé</u> <u>AttributDeClasse</u>
Idem pour les opérations

Attribut de classe

Visibilité globale : l'attribut est considéré comme un objet partagé par les instances d'une classe

PRODUIT
NumProduit IntituléProduit PrixProduit <u>NbreDeProduits</u>
Créer() Supprimer()



# Relations entre classes

**Les relations qui se trouvent entre les classes :**

- ▶ Association
- ▶ Agrégation
- ▶ Composition
- ▶ Dépendance
- ▶ Héritage

# Les associations

## ► Les associations :

- Une association établit expression d'une connexion sémantique bidirectionnelle entre plusieurs  $n$  classes ( $n \geq 1$ ).
- Illustre une relation conceptuelle qui dure dans le temps entre  $n$  classes ( $n \geq 1$ ).

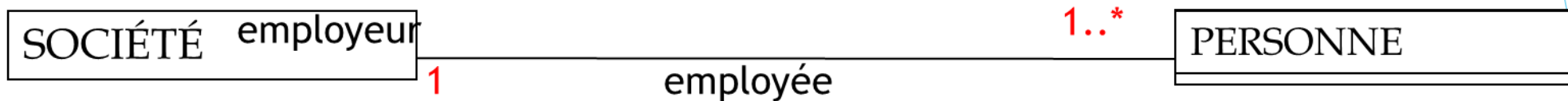


Des instances de l'association **emprunter** :  
{ (A1, E1), (A1, E2), .... (A2, E1), (A2, E4), ....}

# Les associations

## ► Les cardinalités

- spécifient combien le nombre d'objets d'une classe pouvant être liés à un autre objet.



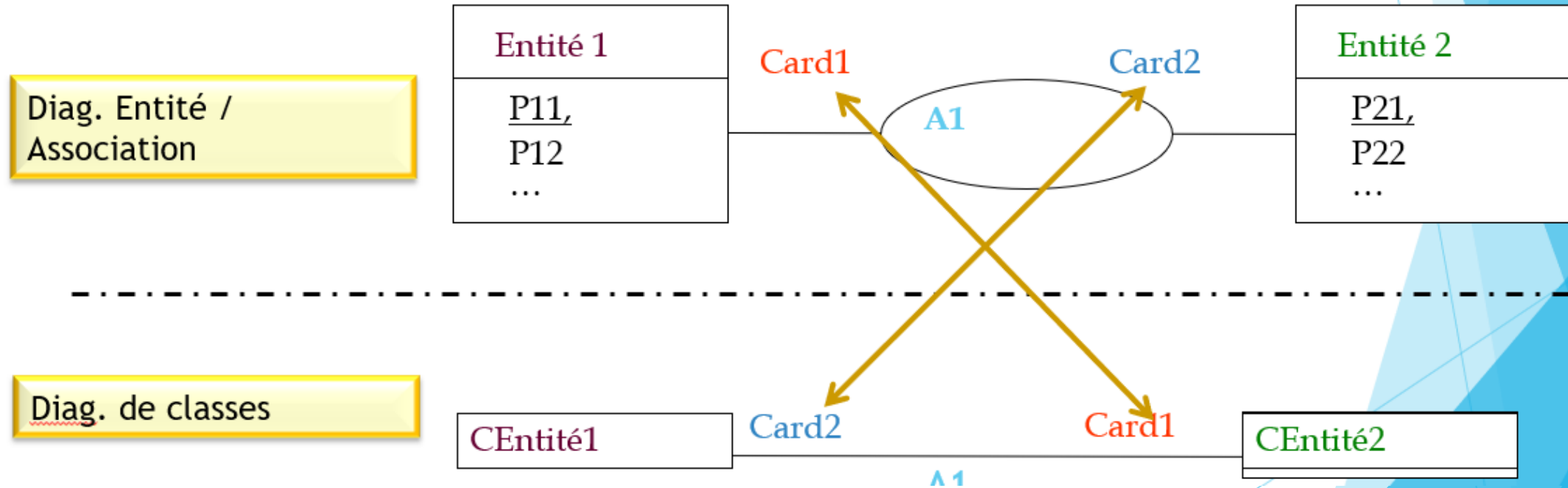
### Valeurs de cardinalité conventionnelles

1	Un et un seul
0 .. 1	Zéro ou un
N (exemple : 8)	N (entier naturel)
M .. N (exemple : 3..7)	De M à N (entiers naturels)
*/0 .. *	De 0 à plusieurs
1 .. *	De 1 à plusieurs

# Les associations

## ► Les cardinalités

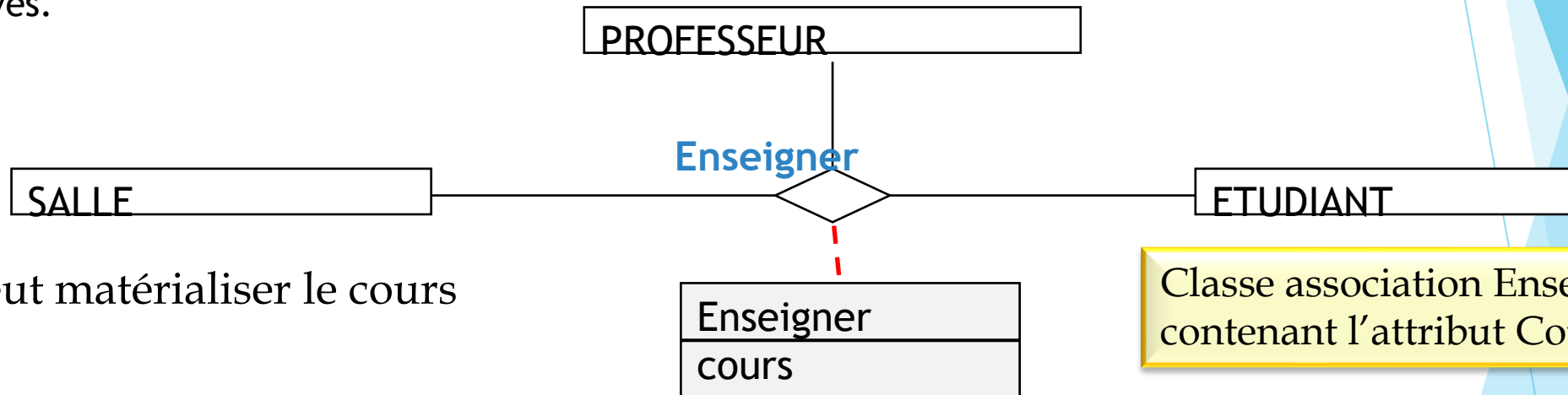
- Comparaison avec le modèle entité/association :



# Les associations

## ► Types des associations :

- Les associations peuvent être récursives, binaires ou racontées (doivent être évitées).
- **Exemple** : Nous voulons représenter la relation éducative entre l'enseignant, la classe et les élèves.



Si on veut matérialiser le cours

Classe association Enseigner contenant l'attribut Cours

Association Enseigner porteuse de données

- ✓ **Remarque** : Lorsqu'il faut relier plusieurs classes, le symbole d'un losange permet de faire la connexion :

# Les associations

## ► Dénomination des associations

- La nomination d'une association peut être :



- Lorsqu'une association est ambiguë, il est possible de préciser **le sens de lecture**.

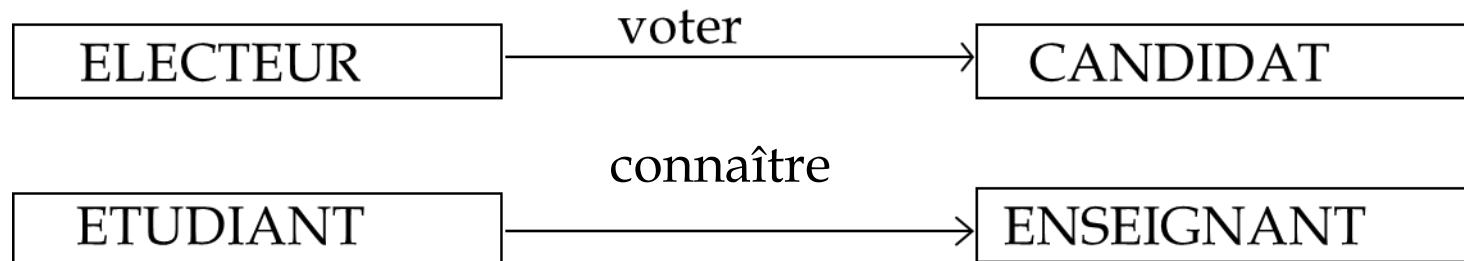


# Les associations

## ► Association à navigabilité restreinte :

- Par défaut, une association permettant la navigation bidirectionnelle.
- Il est envisageable de la restreindre unidirectionnel dans un modèle.

➔ Il est stipulé que les instances d'une classe ne sont pas conscientes des instances de l'autre.



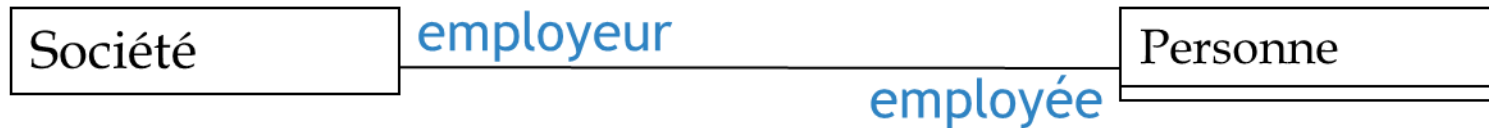
# Les associations

## ► Le concept de rôle :

- Le bout d'une association est défini par un nom appelé "rôle", permettant ainsi de décrire la perception d'une classe source envers une classe destination au sein de l'association.



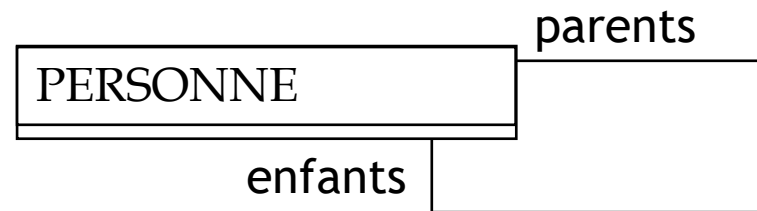
- Rôle 1 : La contribution de Classe 1 à l'association
- Rôle 2 : La contribution de Classe 2 à l'association





# Les associations

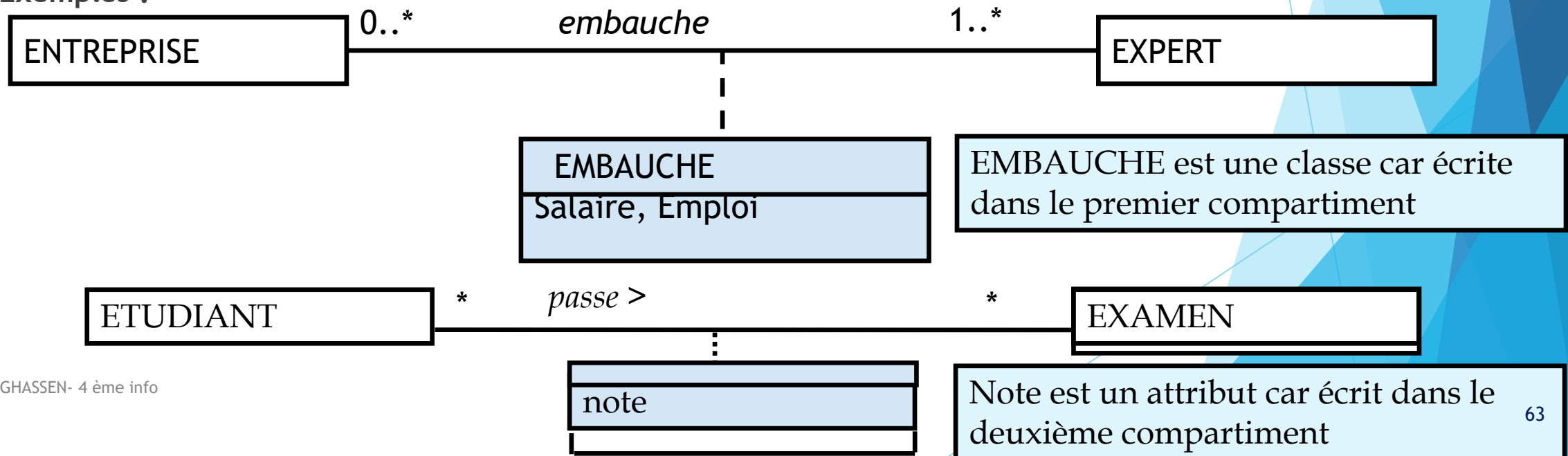
- Pour les associations ambiguës, il est nécessaire de spécifier les rôles.



# Les classes-association

## ► Une classe-association :

- Lorsqu'une association doit inclure des attributs ou s'associer à d'autres associations, elle se transforme en classe associative ou en classe association.
- Une classe peut être utilisée pour représenter une association grâce à une classe-association, ce qui permet d'ajouter des attributs et des opérations dans l'association.
- présente les attributs d'une classe et d'une association
- Exemples :



# L'agrégation

- ▶ L'agrégation est une forme spéciale de connexion. Cela représente le rapport de participation entre les éléments du groupe. Le nombre
- ▶ est représenté par l'ajout d'un losange vide sur le côté de l'appareil.
- ▶ L'agrégation signifie la relation du tout à ses parties. Le tout est la multitude, et la partie est la multitude.



# La composition

- ▶ Les éléments, également appelés agrégats mixtes, sont des combinaisons fortes.
- ▶ décrit les structures de quarantaine entre les occurrences.
- ▶ Ainsi, la suppression d'un organisme composite engendre la destruction de ses constituants.
- ▶ Une occurrence de la pièce reste toujours affilié à au plus une occurrence de l'élément composite :
- ▶ La cardinalité de l'arête composite ne peut excéder 1 (c'est-à-dire 1 ou 0,1).

# La composition

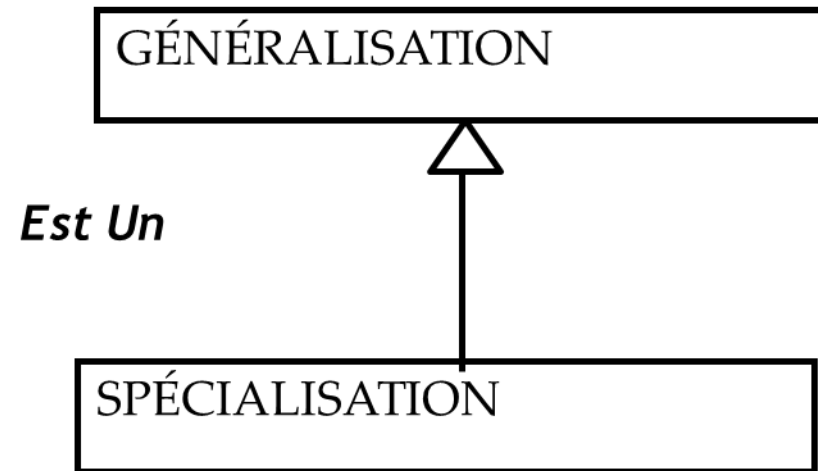
- ▶ Les caractéristiques des composants et du composé coïncident :
  - ▶ En cas de destruction (ou de duplication) du composé, ses composants subissent également cette action
  - ▶ une instance de composant ne peut être associée qu'à un seul composé.
  - ▶ Les "**objets composites**" représentent des occurrences de classes composées.



**Remarque :** tous les accords de puissance restent valables pour l'agrégation et la composition.

# La généralisation

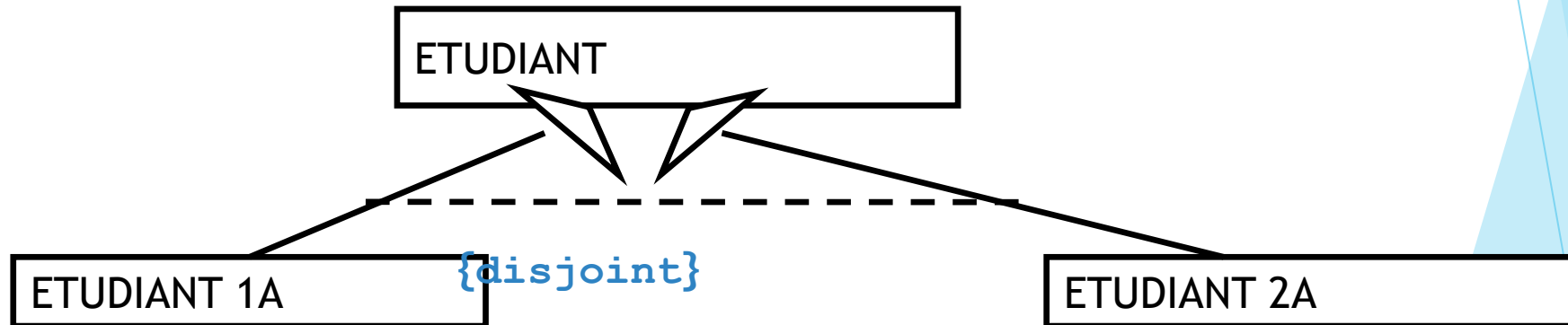
- ▶ En utilisant UML, on peut désigner l'héritage par la généralisation qui peut être :
  - ▶ Simple



- ▶ Multiple : la spécialisation possède plus qu'une généralisation

# La généralisation

- ▶ Limites et caractéristiques de la généralisation :
  - ▶ Les critères ≠ peuvent être utilisés pour assigner une classe générale.
  - ▶ Contrainte exprimée dans le mot-clé {disjoint}
  - ▶ Tout objet qui est une descendance directe d'une seule sous-classe
  - ▶ La généralisation traduit une fragmentation exclusive par défaut.



Inter (ETUDIANT 1A, ETUDIANT 2A) = vide

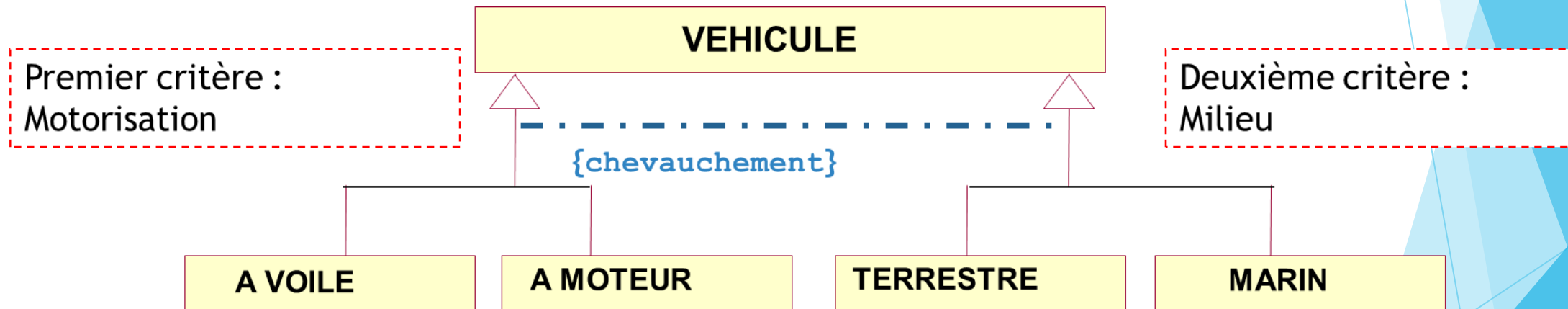
Par rapport à Merise : contrainte de disjonction et non pas de couverture

# La généralisation

## ► Limites et caractéristiques de la généralisation :

- La contrainte décrite par le mot-clé `{chevauchement}`

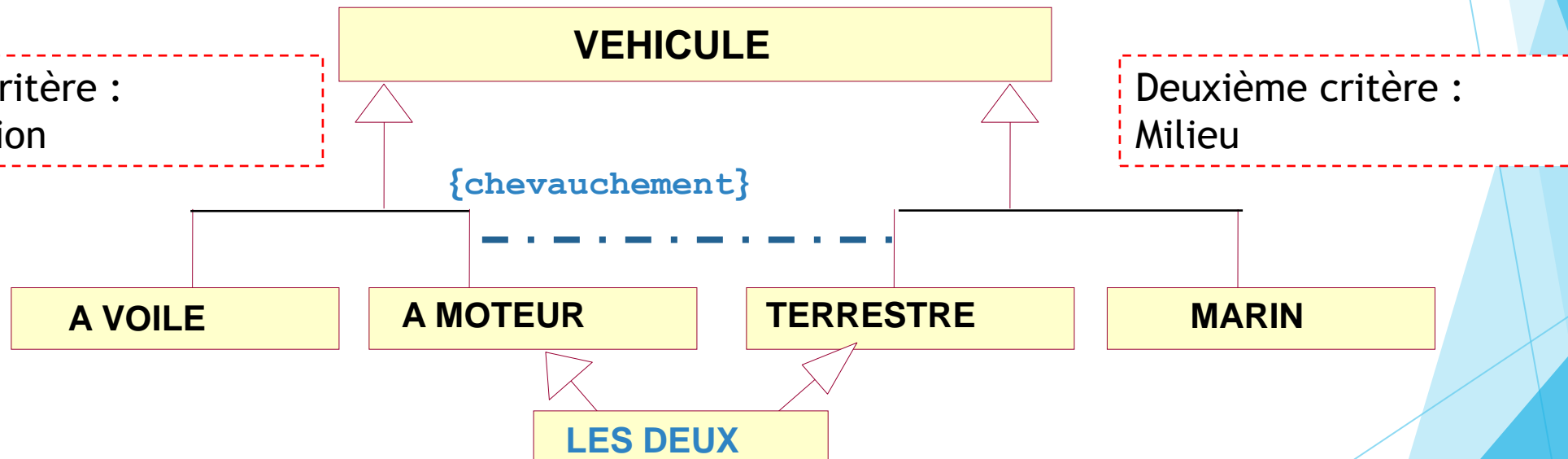
- Il est possible d'avoir simultanément une instance de l'une des spécialisations et une instance d'une autre.





# La généralisation

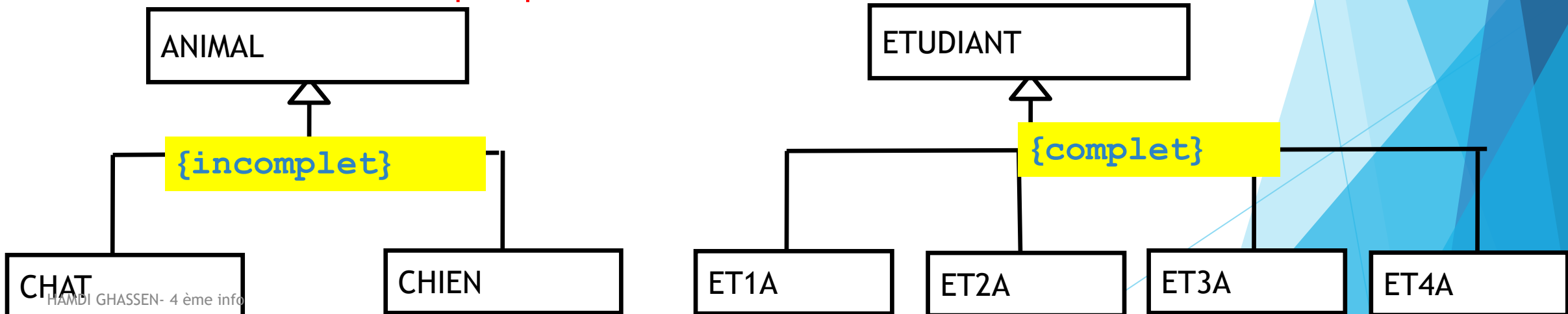
- ▶ Si le chevauchement ne peut se faire qu'entre les classes : A MOTEUR et TERRESTRE, le schéma devient :



# La généralisation

## ► Limites et caractéristiques de la généralisation :

- La contrainte désignée par le mot-clé `{complet}`
  - Indique que la généralisation est terminée :
    - Aucune autre sous-classe ne peut être ajoutée
- La contrainte désignée par le mot-clé `{incomplet}`
  - Montre que la généralisation peut être étendue : elle peut avoir des contraintes autres que spécialisations.



# *UML : Unified Modelling Language*

## Chapter 5

### Diagramme de séquence



hamdighassan@gmail.com

**Ghassen HAMDI**

Academic Year: 2021/2022

# Introduction

## **Cas d'utilisation autorisée :**

utilisateurs ont exprimé leur souhait.

**Pour établir une liste initiale des acteurs (objets et entités) qui font partie du système.**

## **Exigence :**

Ensemble de maintenance demandé par l'utilisateur (que dois-je faire ?)

Cette exigence doit maintenant être expliquée par un professionnel de l'informatique (Comment ?)

Diagramme d'interaction.

# Sémantique

- Les **objets** qui collaborent sont identifiés par leur nom, leur fonction et/ou leur catégorie.

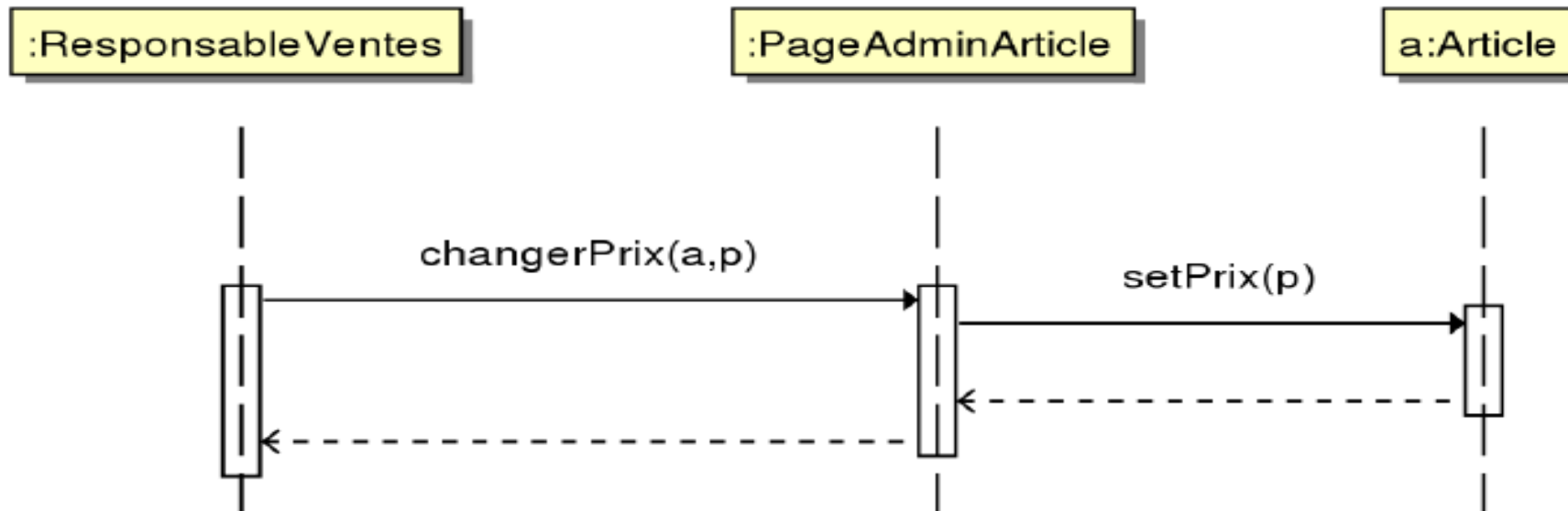
<u>O1</u>	Un objet O1	Ali
<u>:C</u>	Un objet anonyme de la classe C	:PERSONNE
<u>/R:C</u>	Un objet anonyme de la classe C jouant le rôle R	/Lecteur:PERSONNE
<u>/R</u>	Un objet anonyme jouant le rôle R	/Lecteur
<u>O/R:C</u>	Un Objet O, instance de la classe C, jouant le rôle R	Ali/Lecteur:PERSONNE

# Exemple d'interaction

- Cas d'utilisation :

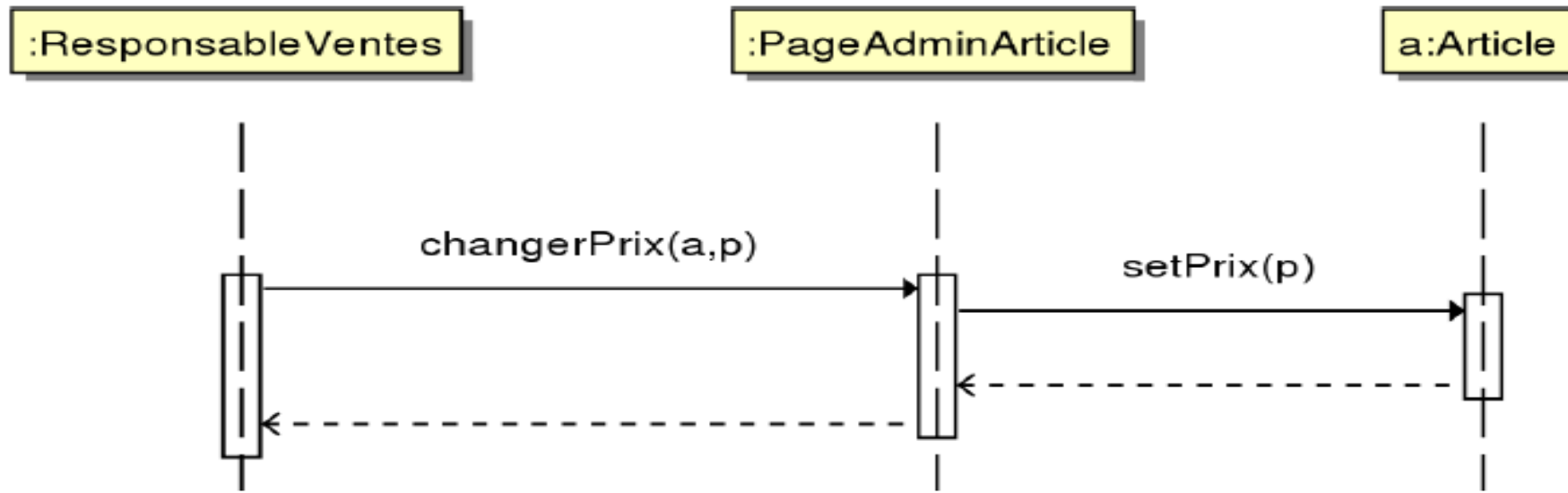


- Diagramme de séquences correspondant :

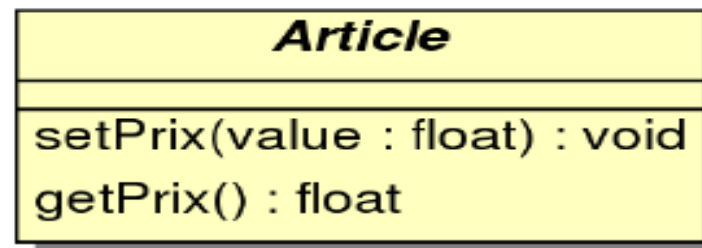


# Exemple d'interaction

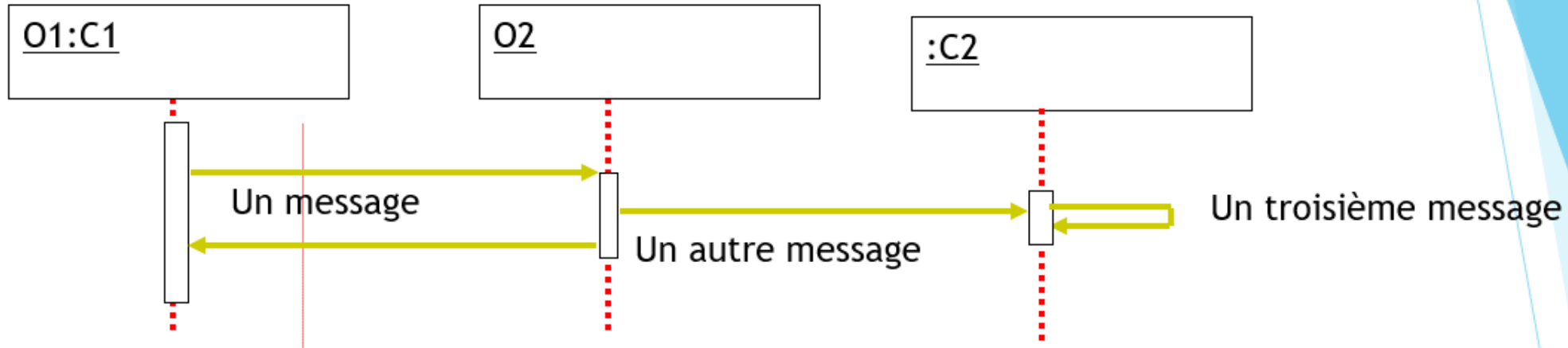
- Diagramme de séquences correspondant :



- Opérations nécessaires dans le diagramme de classes :



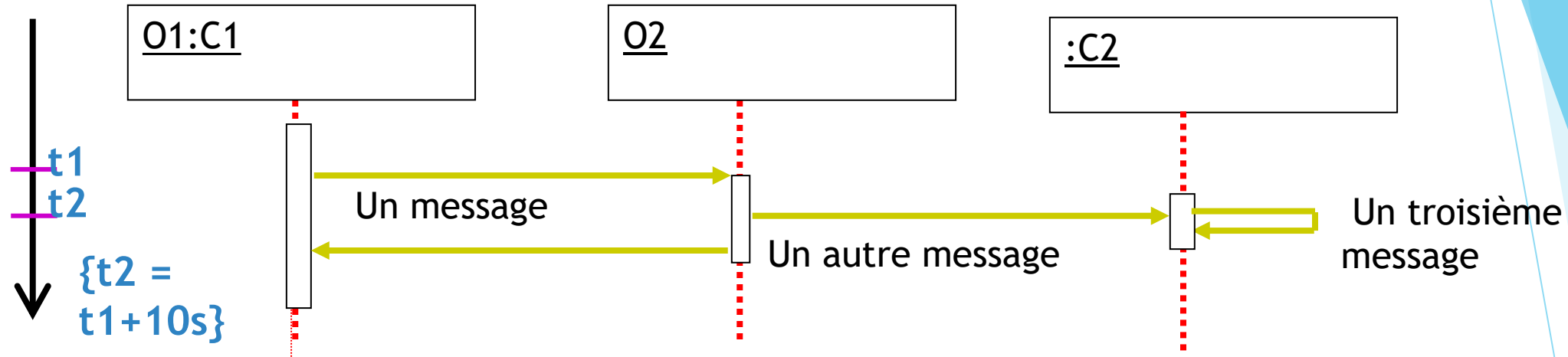
# Représentation générale



- ▶ Les objets qui initient la conversation sont placés à gauche
- ▶ Un rectangle et une ligne de vie verticale sont utilisés pour représenter chaque objet.
- ▶ L'importance de l'ordre horizontal des objets n'est pas vraiment significative.
- ▶ Il n'y a aucune implication pour la sémantique du diagramme.



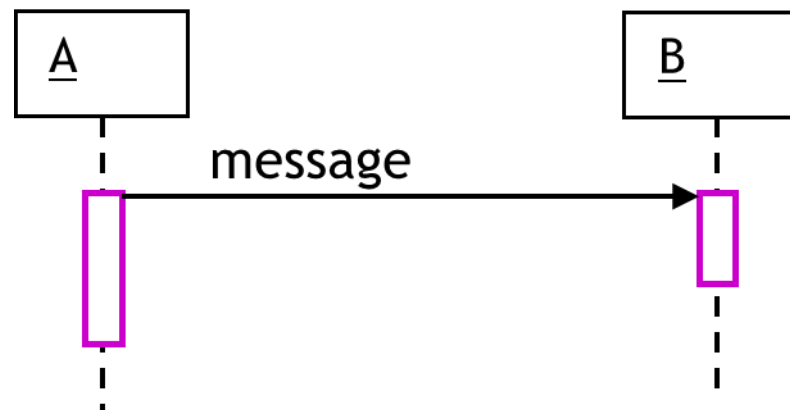
# Représentation générale



- ▶ L'ordre d'envoi des messages est déterminé par leur position sur les lignes de vie des objets (sur l'axe vertical du graphique) :
- ▶ Le temps va 'de haut en bas' de cet axe.
- ▶ La dimension verticale représente le passage du temps.
- ▶ peuvent être classés pour exprimer des contraintes de temps.

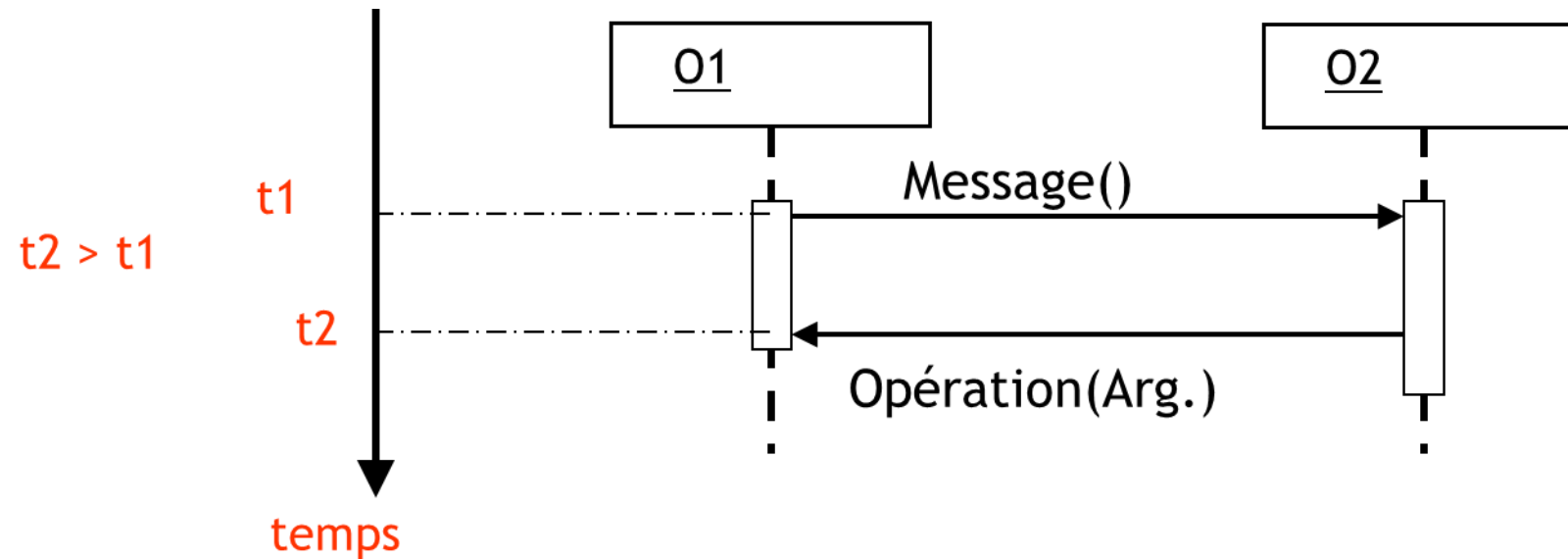
# Activation d'un objet

- ▶ Il est possible d'afficher clairement les différents intervalles d'activité d'un objet en utilisant une barre rectangulaire superposée à la ligne de vie de l'objet.
  - ▶ Une période d'activité représente le temps pendant lequel un objet exécute une action, que ce soit directement ou par l'intermédiaire d'un autre objet.
- ❑ Un exemple d'objet qui active un autre est celui où la période d'activité de A recouvre celle de B.



# Activation et envoi de messages

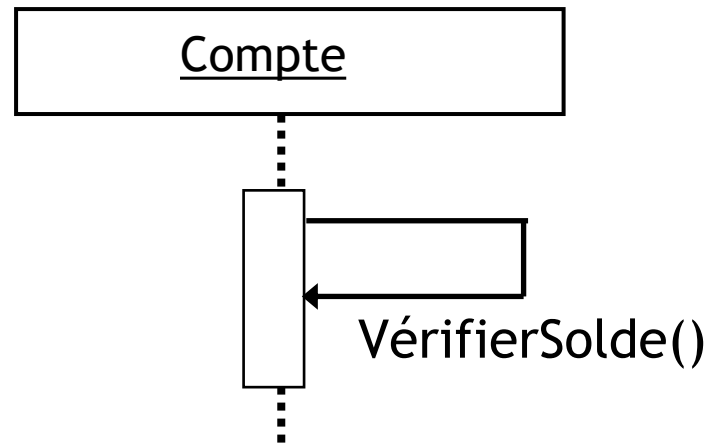
- L'ordre d'affichage des messages (ligne verticale : de haut en bas) peut être utilisé à la place de la numérotation des messages.



❖ N.B. Temps d'**envoi** de Message() par O1 = Temps de **réception** de Message() par O2 =  $t_1$

# Activation et envoi de messages

- Un message peut être considéré comme étant **réflexif**.



# Activation et envoi de messages

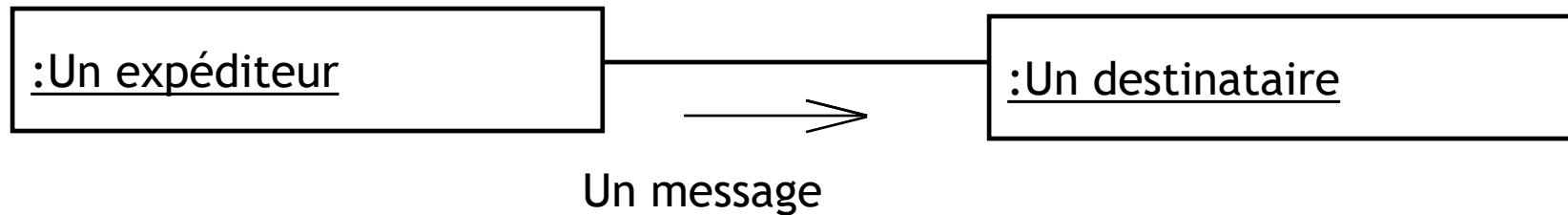
## ► Types d'envoi de messages :

1. Les flots de contrôle à plat.
2. Appel de procédure ou flot de contrôle emboîté.
3. Retour d'un appel de procédure.

# Activation et envoi de messages

## 1. Les flots de contrôle à plat :

- ▶ Signifiant l'avancement vers la prochaine étape d'une séquence.
- ▶ les messages de cette catégorie sont représentés par une flèche simple.

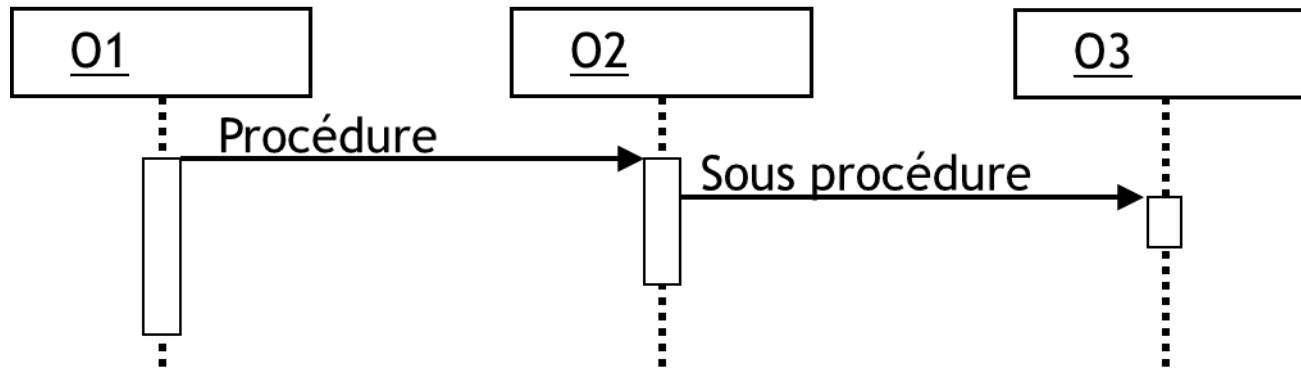


Les messages asynchrones ne sont pas bloqués pour les expéditeurs. Les destinataires peuvent soit envoyer le message sur leur compte, soit ignorer le à tout moment.

# Activation et envoi de messages

## 2. Appel de procédure ou flot de contrôle emboîté :

- ▶ L'appelant ne reprend le contrôle que si le flux emboîté se termine
- ▶ Une flèche pleine repère les messages de cette catégorie.

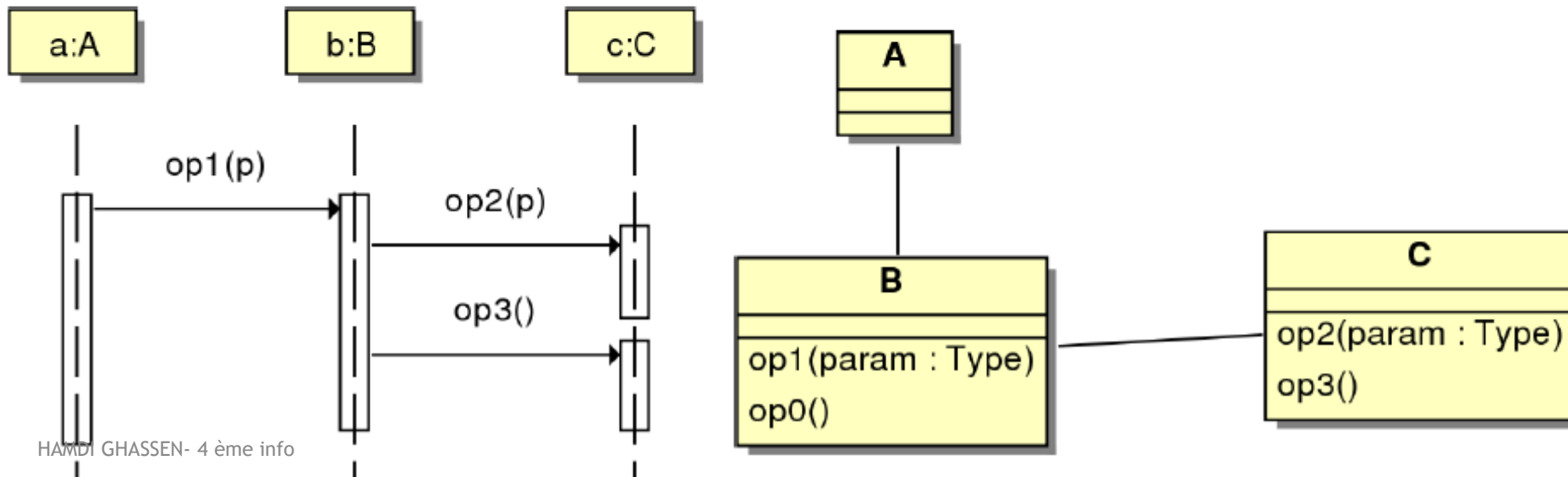


- O2 reprend le contrôle lorsque O3 termine sa mission.
- La fin du sous-programme est indiquée par la fin du temps d'activation de O3.
- O1 reprend le contrôle lorsque O2 termine la mission.

# Activation et envoi de messages

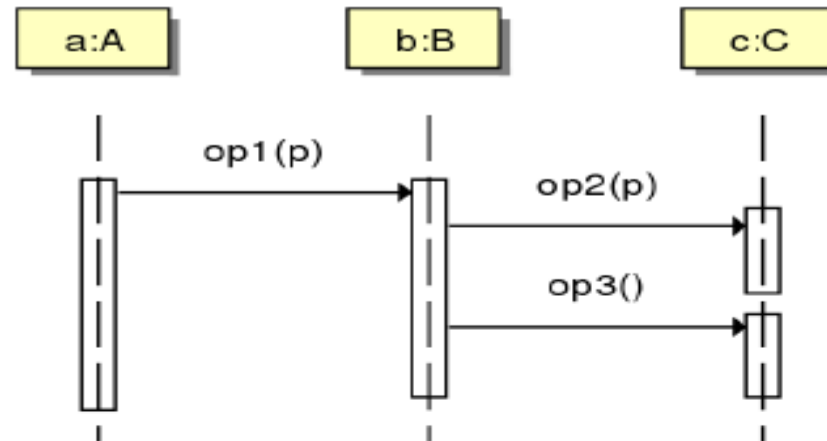
- Les **messages synchrones** correspondent à des **opérations** dans le diagramme de classes.

Envoyer un message et attendre la réponse pour poursuivre son activité revient à invoquer une méthode et attendre le retour pour poursuivre ses traitements.





# Activation et envoi de messages



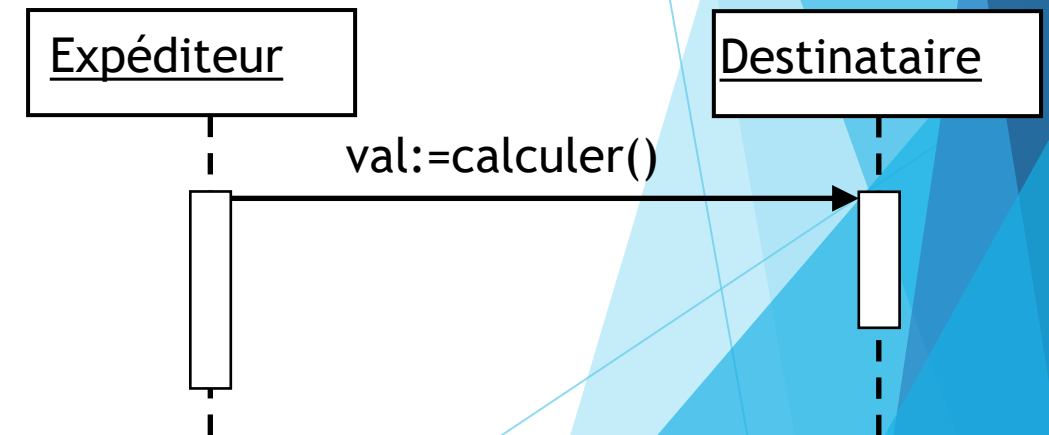
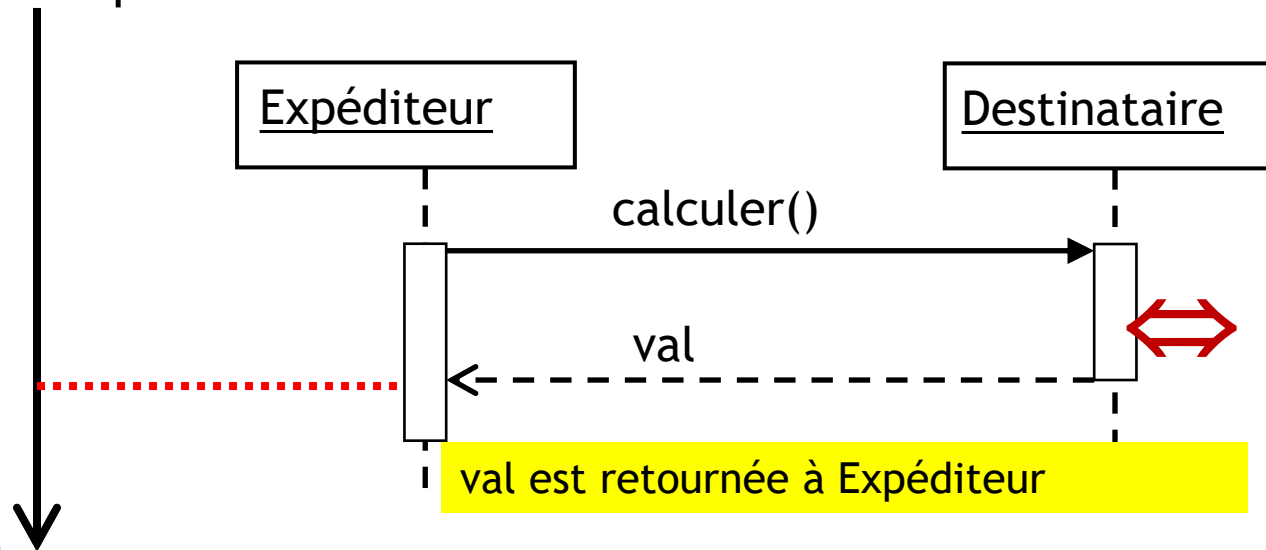
```
class B {
    C c;
    op1(p:Type){
        c.op2(p);
        c.op3();
    }
}
```

```
class C {
    op2(p:Type){
        ...
    }
    op3(){
        ...
    }
}
```

# Activation et envoi de messages

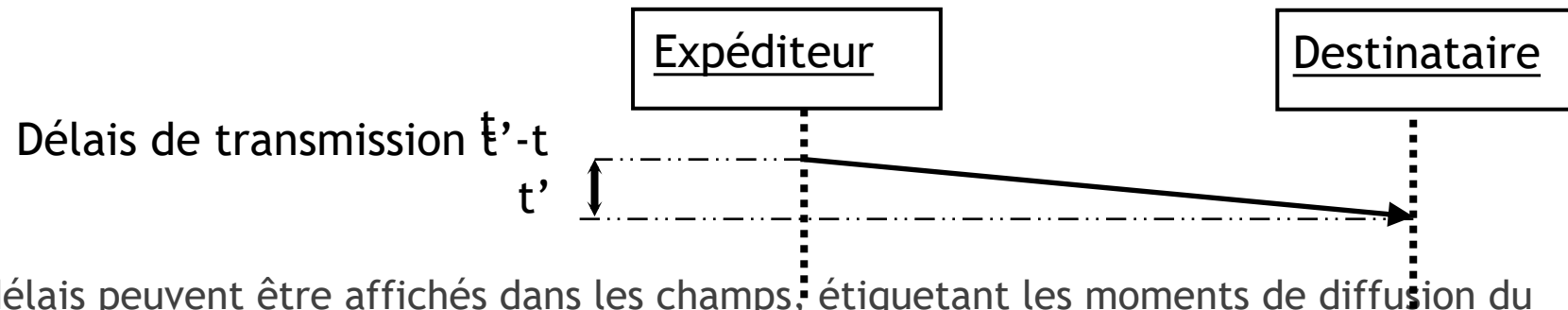
## 3. Retour d'un appel de procédure :

- ✓ La fin de l'activation de l'objet récepteur marque implicitement le retour de l'opération.
- ✓ Les paramètres renvoyés sont également représentés par cette notation.
- ✓ Tous les messages de cette catégorie sont soulignés avec une flèche en pointillés.

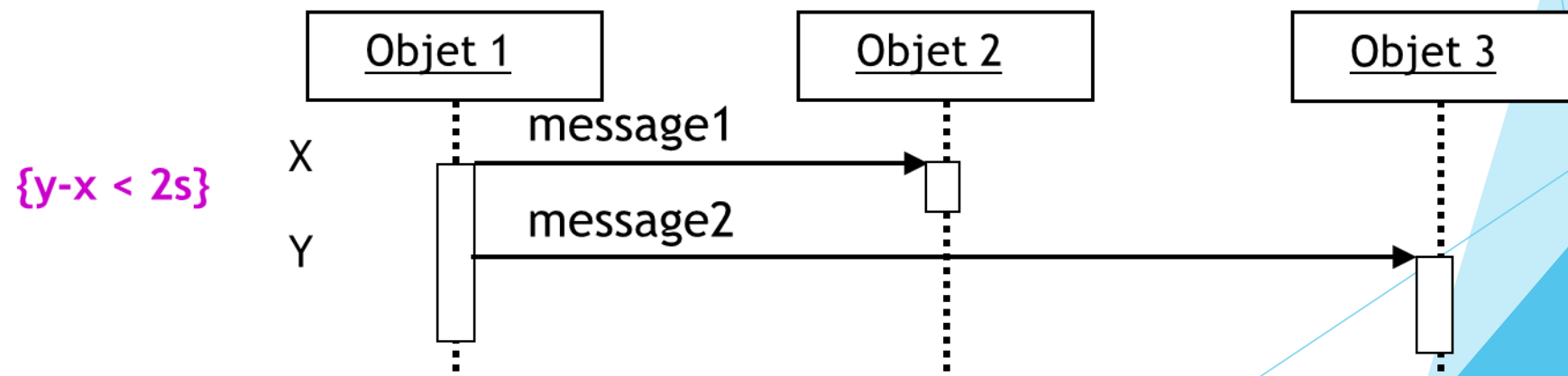


# Contraintes temporelles

- La flèche représentant le message peut être affichée en diagonale: délai de livraison anormale



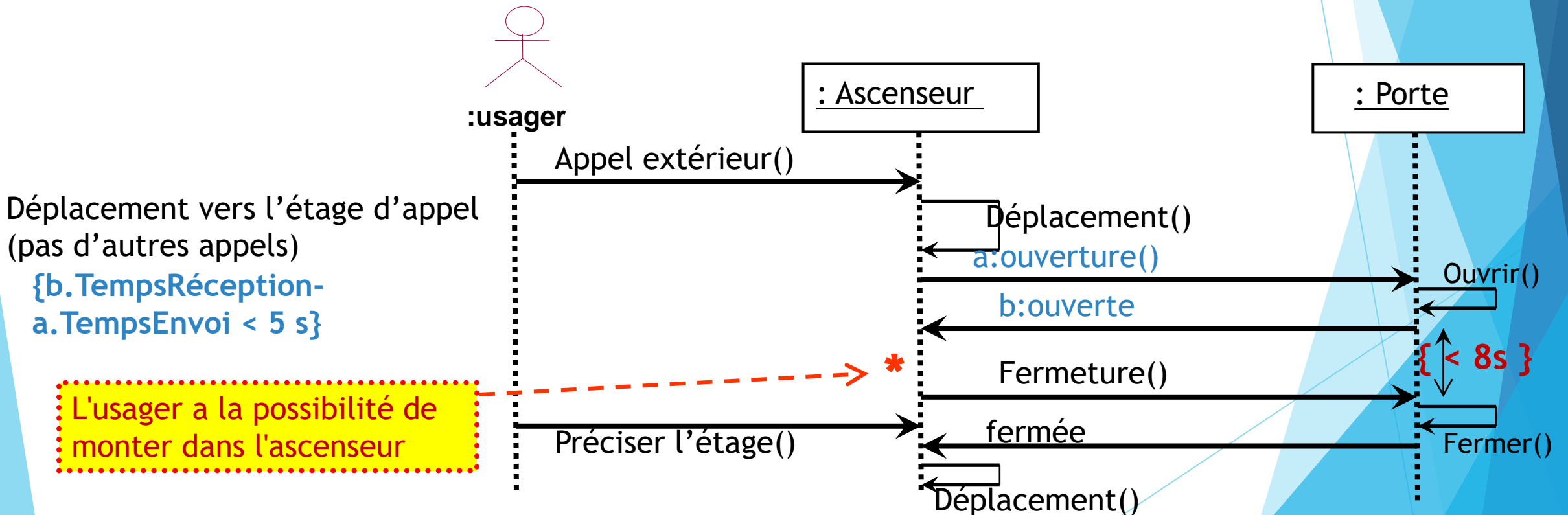
- Les délais peuvent être affichés dans les champs, étiquetant les moments de diffusion du message



# Contraintes temporelles

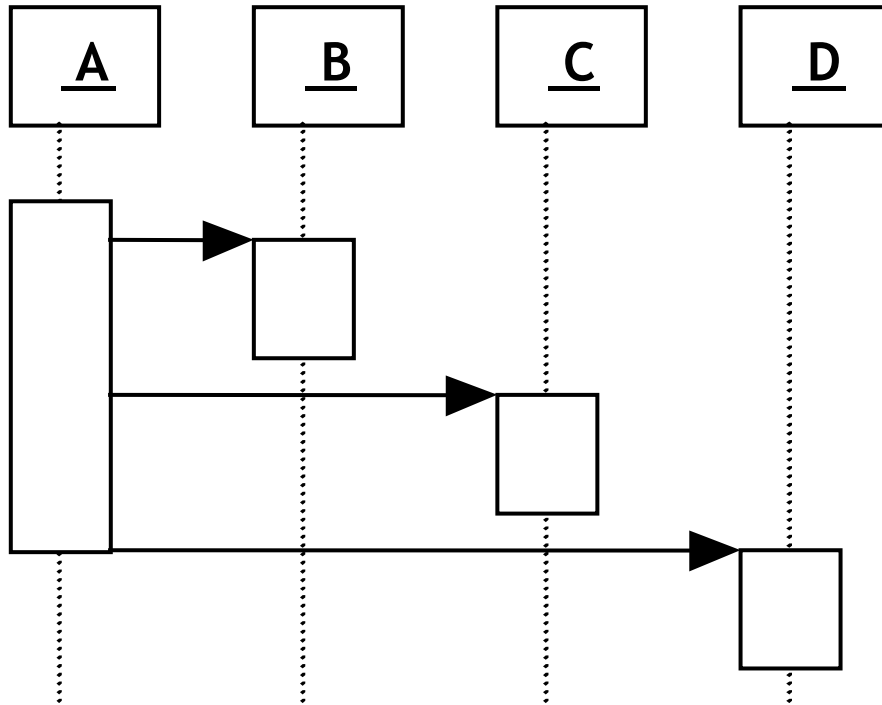
- Les annotations temporelles (**temps : message**) et les fonctions temporelles (**TempsEnvoi**, **TempsRéception**, ...) peuvent être utilisées pour exprimer les contraintes temporelles.

- Exemple :

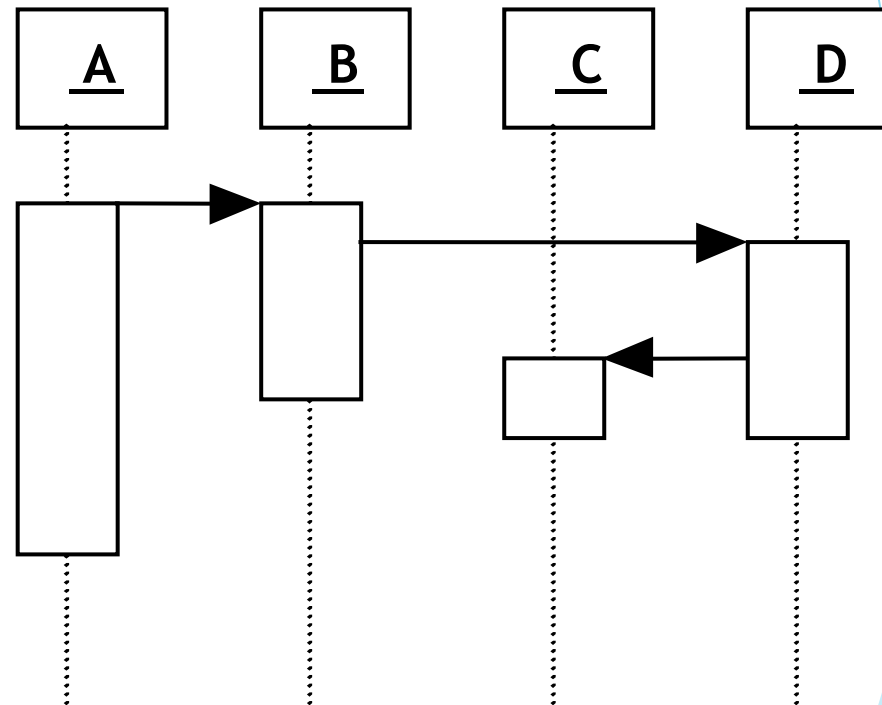


# Structures de contrôle

- Contrôle **centralisé** (I) et contrôle **décentralisé** (II)



(I)



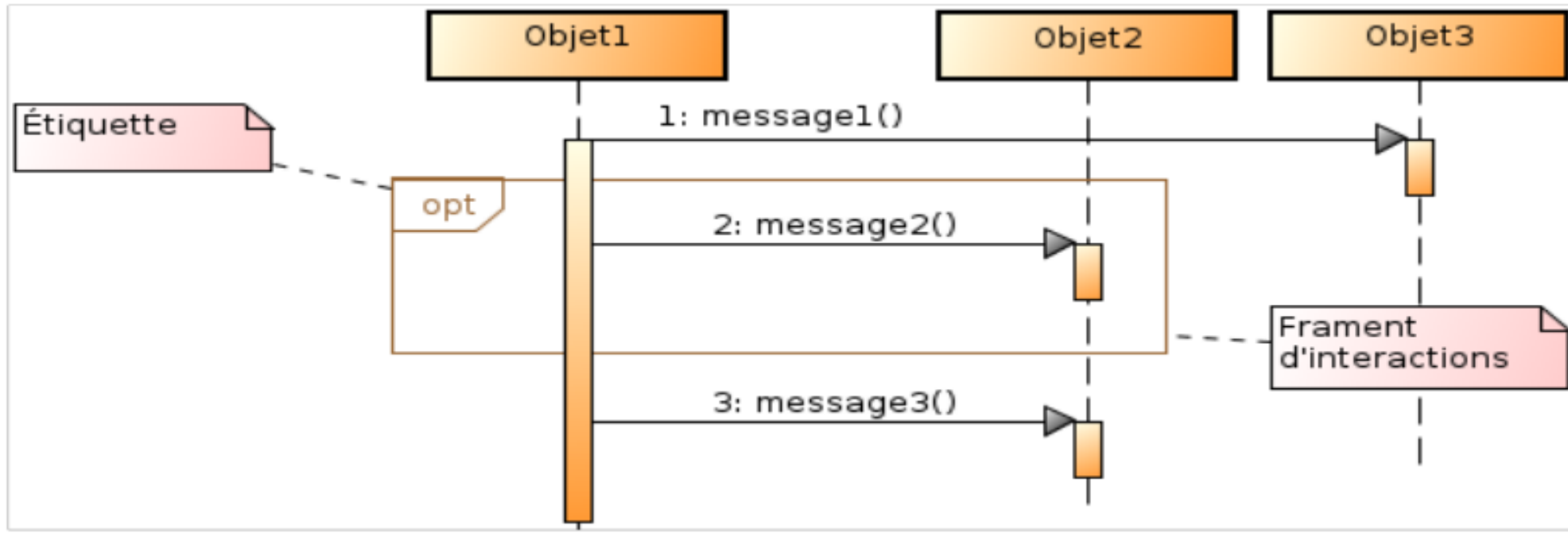
(II)

A **centralise** les envois de messages

L'envoi de messages est **décentralisé**

# Fragments d'interactions combinés

- Le volet d'interaction fait partie d'un diagramme de séquence (encadré d'un rectangle) lié à une balise (dans le coin supérieur gauche). L'étiquette contient une déclaration d'interaction qui vous permet de montrer comment les messages sont exécutés dans le cadre.



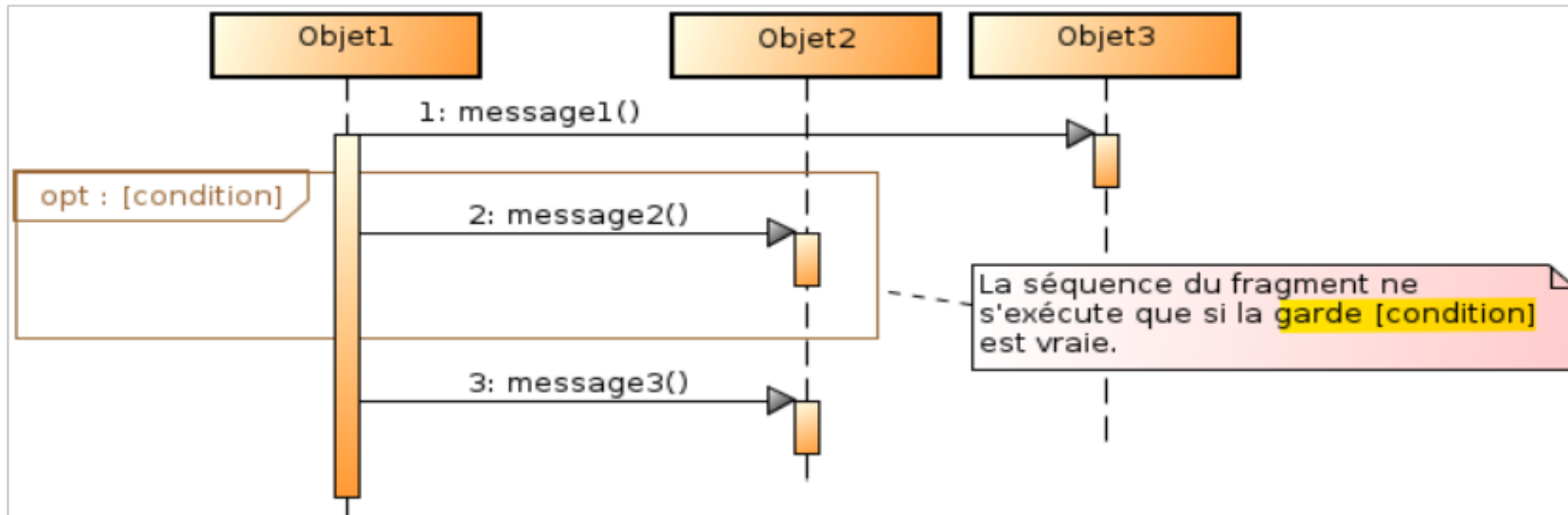
# Fragments d'interactions combinés

- ▶ Les coefficients du facteur d'interaction sont distingués par une ligne en pointillés. La condition de sélection d'un opérande (le cas échéant) est définie comme une expression booléenne entre crochets ([ ]).
- ▶ Les principales méthodes sont les boucles, les sauts conditionnels, les alternatives, le dispatching parallèle, etc.

# Fragments d'interactions combinés

## ► Fragment d'interaction avec opérateur « **opt** » :

Le sous-fragment est exécuté si la condition de garde est vraie et n'est exécuté d'aucune autre manière.

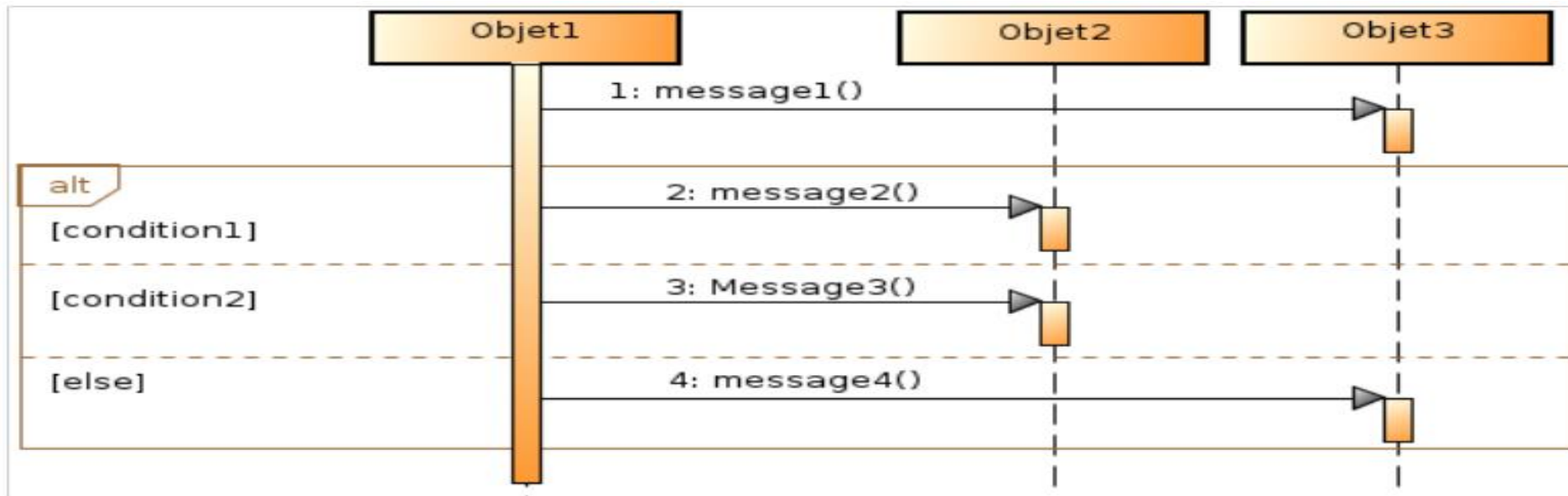




# Fragments d'interactions combinés

## ► Fragment d'interaction avec opérateur « alt » :

L'opérateur de choix (alt) est un opérateur conditionnel où plusieurs opérandes sont délimités par des traits d'union. Seule la partie du code associée à la première condition vraie est exécutée. La clause "else" est activée uniquement en l'absence de conditions valides supplémentaires.



# Fragments d'interactions combinés

## ► Fragment d'interaction avec opérateur « **loop** » :

- On formule la garde de la manière suivante :

**loop** [**min**, **max**, **condition**] : Chaque paramètre (**min**, **max** et **condition**) est optionnel.

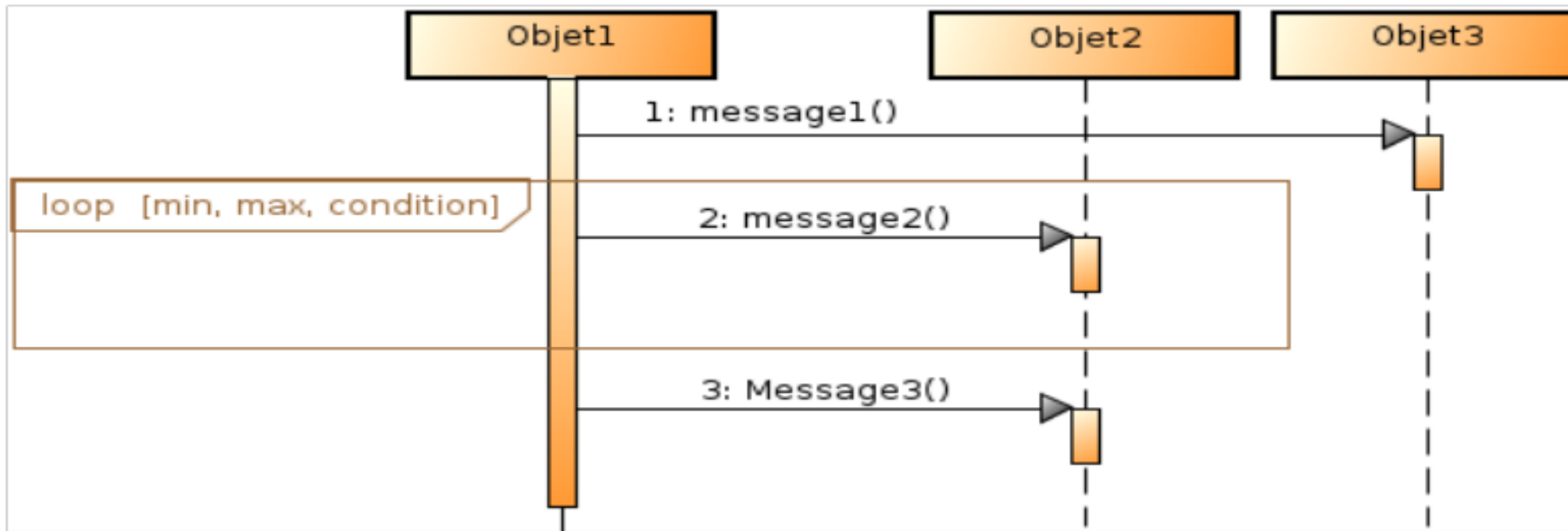
- Exemple de gardes :

**loop**[**3**]→La séquence s'exécute **3** fois.

**Loop**[**1**, **3**, **code=faux**] La séquence s'exécute **1** fois puis un maximum de **2** autres fois si **code=faux**.

# Fragments d'interactions combinés

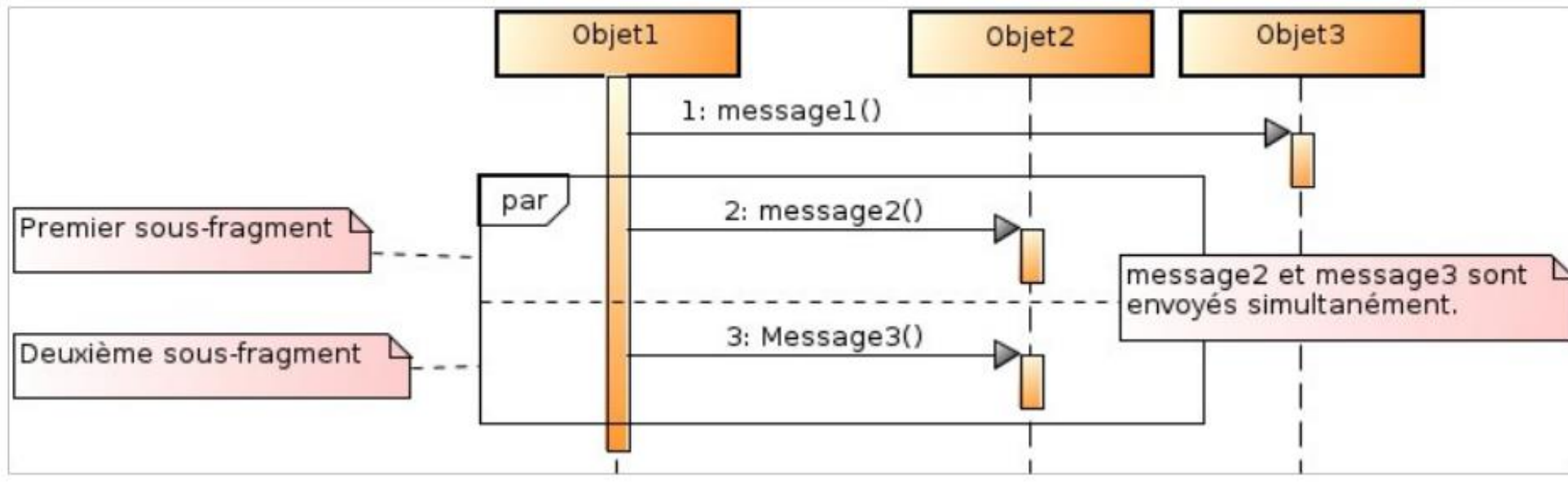
- Fragment d'interaction avec opérateur « **loop** » :



# Fragments d'interactions combinés

## ► Fragment d'interaction avec opérateur « **par** » :

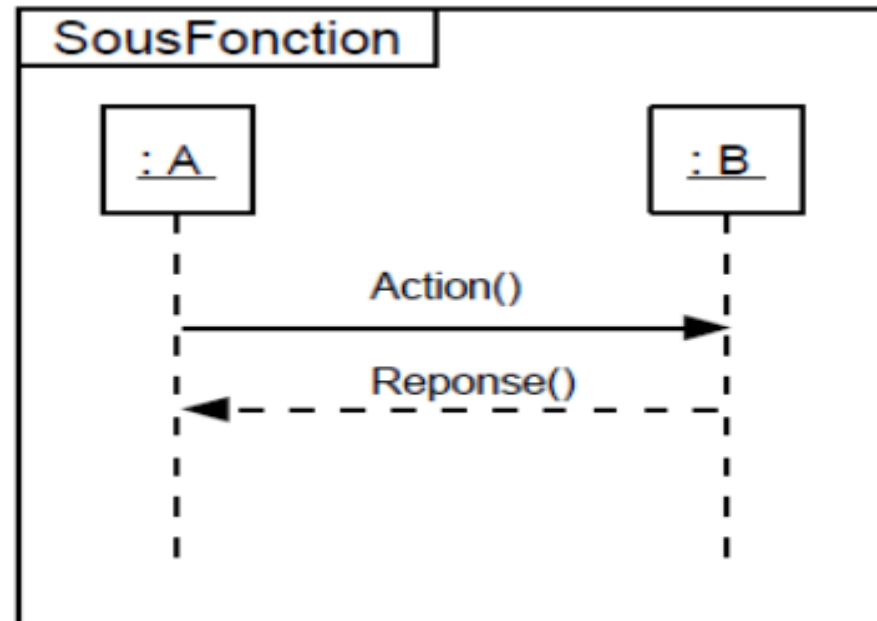
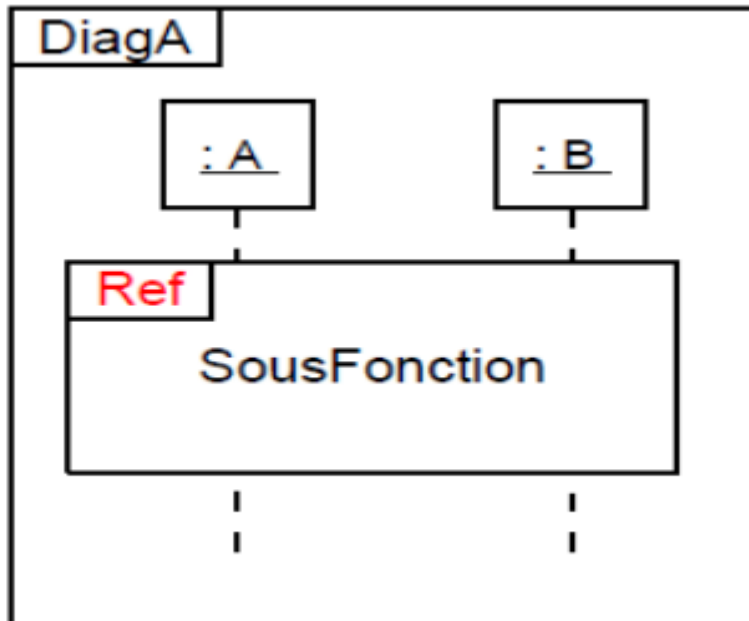
Un fragment interactif avec un opérateur de parallélisme (par) qui contient deux ou plusieurs sous-parties séparées par une ligne pointillée, qui sont exécutées d'une façon parallèle.



# Fragments d'interactions combinés

- Fragment d'interaction avec opérateur « **ref** » :

ref : permet de faire appel à un autre diagramme de séquence..



# *UML : Unified Modelling Language*

## Chapter 6

### Diagramme d'état transition



hamdighassan@gmail.com

**Ghassen HAMDI**

Academic Year: 2021/2022

# Introduction

- ▶ Jusqu'à présent :
  - ▶ On a étudié le système comme « **un tout** » :
    - ▶ Cas d'utilisation **du système**
    - ▶ Diagrammes d'interaction **du système**
    - ▶ Diagramme de classes **du système**
  - ▶ **Une vue d'ensemble sur le système**
- ▶ Pour s'occuper de chaque classe du système :
  - ▶ Selon des perspectives variées
    - ▶ Aspect dynamique de chaque classe :
      - ▶ Diagramme états – transitions de cette classe

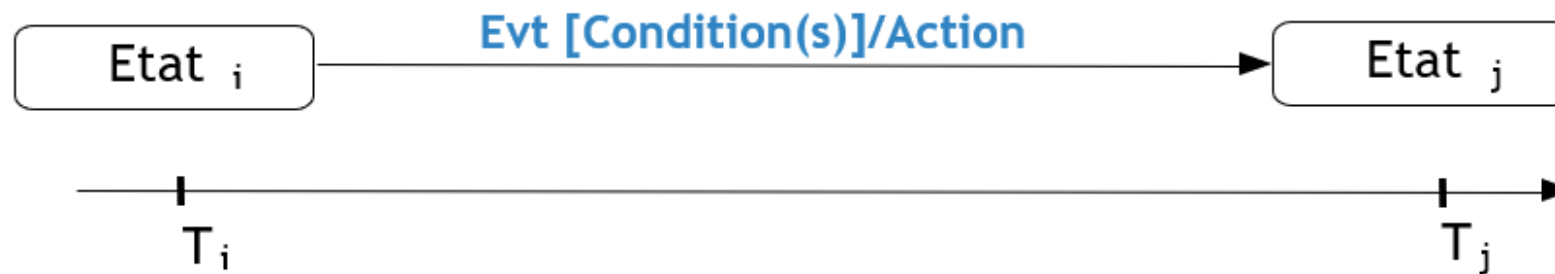
# Introduction

- ▶ Les objets d'une classe ne sont pas fixés :
  - ▶ Au cours de leur cycle de vie, ils ont la capacité de se développer et de passer d'un état à un autre.
  - ▶ L'utilisation d'un DE-T pour une classe permet de décrire l'évolution possible des objets:
  - ▶ L'énumération des états possibles de l'objet au cours de son CV ;
    - ▶ les **événements** Initiant les modifications d'état;
    - ▶ les **éventuelles conditions** se sont les conditions qu'il doit vérifier avant d'effectuer une transition d'état. et
    - ▶ les **opérations** qui entraînent sa transition d'un état à un autre.



# Sémantique

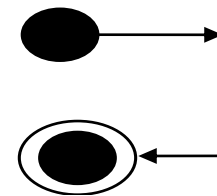
- ▶ Une classe ne doit pas nécessairement avoir un mouvement égalitaire, mais elle peut en contenir plusieurs (avec des significations différentes).
- ▶ La règle graphique pour indiquer le changement du statut est :



A l'instant  $T_i$ , suite à l'arrivée d'un événement *Evt*, et sous certaines *conditions*, l'objet passe de l'*Etat i* à l'*Etat j* par l'activation de l'action *Action*

# L'état d'un objet

- ▶ **L'état** d'un objet est une situation particulière au cours de sa vie.
- Dans certaines circonstances, un objet satisfait une condition, effectue une action ou attend simplement un événement.
- L'état d'un objet est déterminé par les valeurs de tous ses attributs et l'existence de références à d'autres objets.
- Cette condition se caractérise par la durabilité et la stabilité.
- Deux conditions particulières s'appliquent au IT.
  - ***L'état initial*** : état avant la création (dans la BD du système) de l'objet et
  - ***L'état final*** : état après la destruction (de la base) de l'objet.



# L'état d'un objet

## ► Remarques :

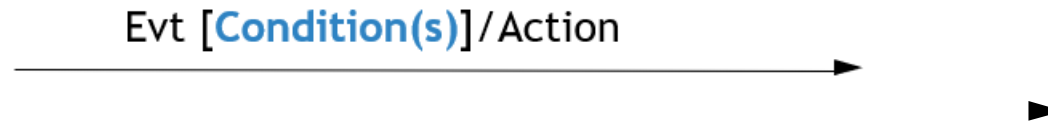
- L'état final correspond à un état dans lequel l'objet ne peut plus grandir.
- Une transition vers un nouvel état ne peut pas se produire à partir d'un état final.
- Certaines actions mènent au dernier emplacement, par exemple supprimer, archiver, annuler....
- Dans une IT il est possible de ne pas avoir de licence définitive de la même manière qu'il est possible d'en avoir plusieurs.

# Les événements

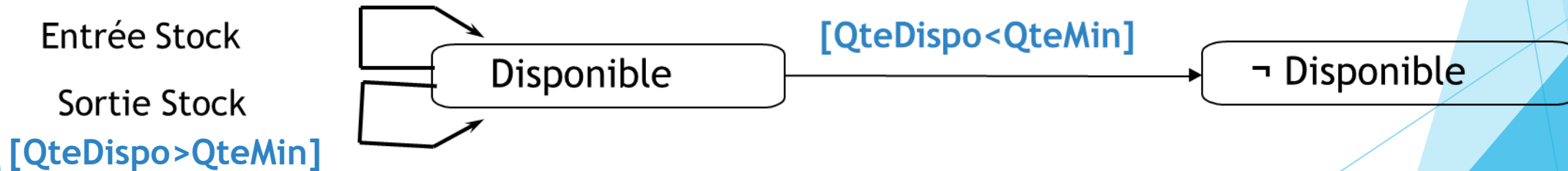
- ▶ **Un événement se définit comme l'apparition d'une situation donnée dans le domaine du problème.**
- ▶ Cette information est disponible immédiatement
  - ▶ Classification des événements :
    - ▶ **Événement externe :**
      - ▶ Exemple : Obtention d'un bon de commande client
    - ▶ **Événement interne :**
      - ▶ Exemple : réception d'une demande d'achat du magasin au service commercial
    - ▶ **Événement temporel :**
      - ▶ Exemple : Supprimer toutes les réservations non confirmées 24 heures avant la date du voyage.

# Les transitions

- ▶ Une transition est un processus qui permet de passer instantanément d'un état à un autre.



- ▶ Un événement est à l'origine de la déclenchement : l'arrivée d'un événement qui bloque la transition.
- ▶ Peut être climatisé avec l'aide d'un gardien.

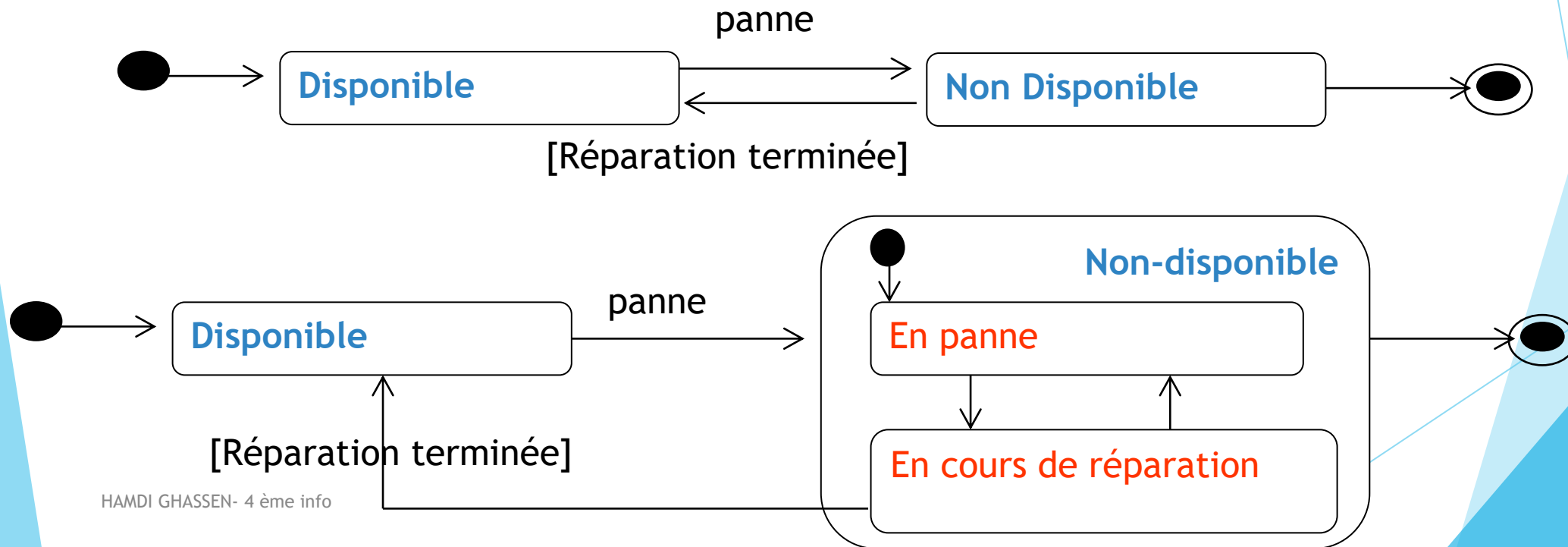


# Concepts avancés sur les états

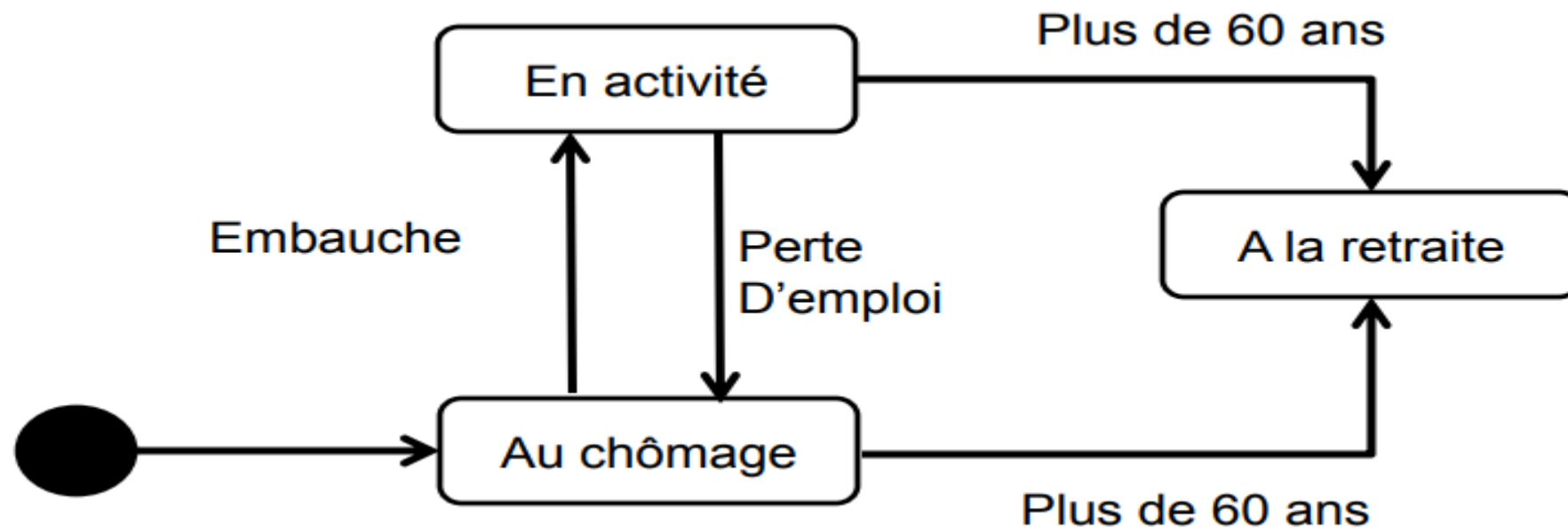
## ► Etat composite (ou composé)

- Chaque état composite est divisé en sous-états.

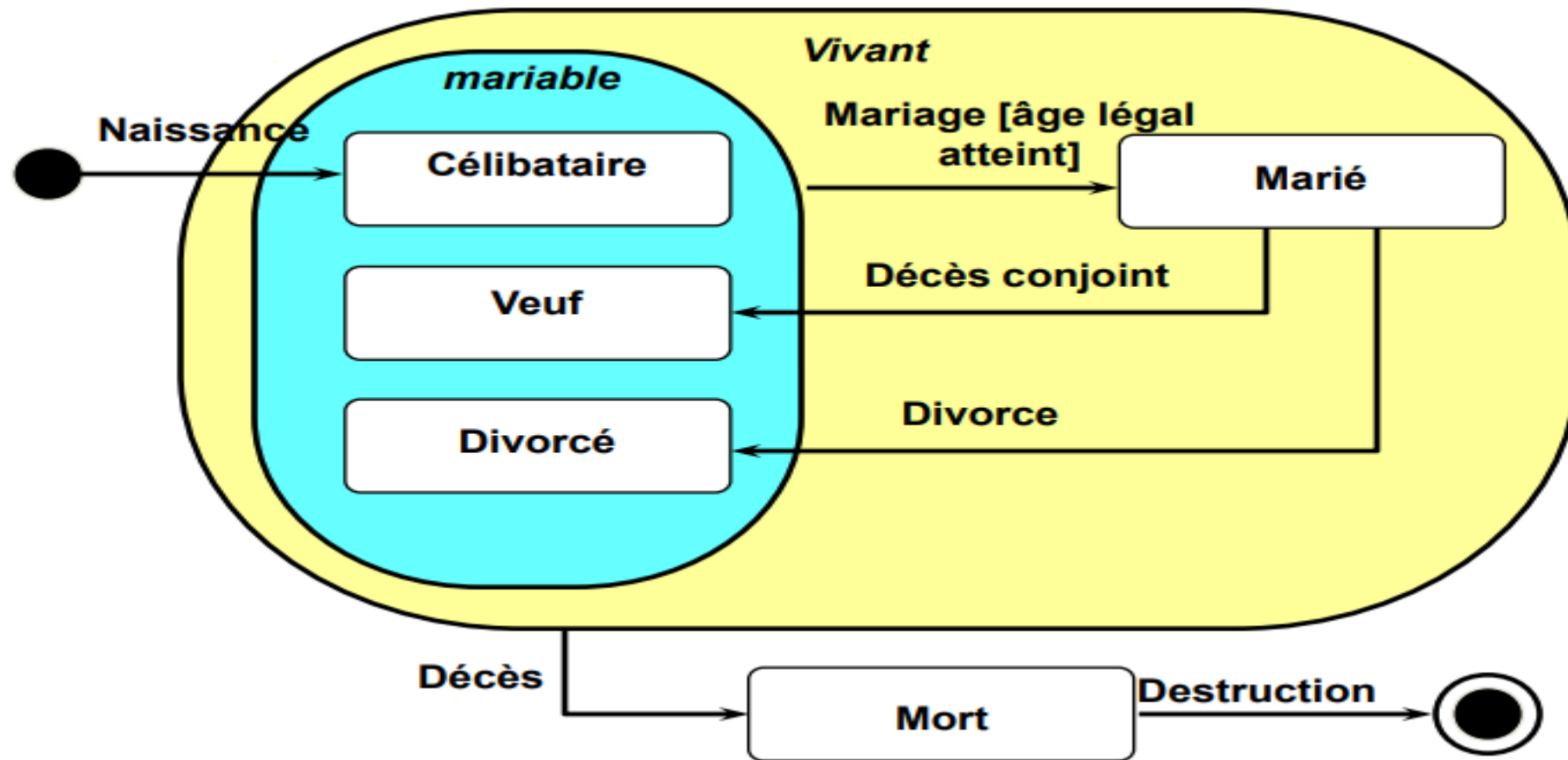
Exemple :



# Exemple



# Exemple





# *UML : Unified Modelling Language*

## Chapter 7

### Diagramme d'activité



hamdighassan@gmail.com

**Ghassen HAMDI**

Academic Year: 2021/2022

# Introduction

- ▶ On peut observer le flux d'une activité à la suivante grâce au diagramme d'activité (DA).
- ▶ Les activités sont des opérations qui ne comportent pas de risque nucléaire
- ▶ Est une variante de DET :
- ▶ Dans DET, on distingue les états et les transitions, tandis que dans YES, on distingue les activités et les transitions.

Un DA est un diagramme d'état vu sous "forme procédurale" avec des actions/activités marquées (d'où le nom).

# Les états d'action (et d'activités)

- ▶ **Un état d'action** modélise l'étape d'exécution d'un algorithme ou d'un thread.
- ▶ C'est un état simplifié qui se caractérise par une action d'entrée et qui débute au moins une transition automatique vers un autre état.

Etudier devis

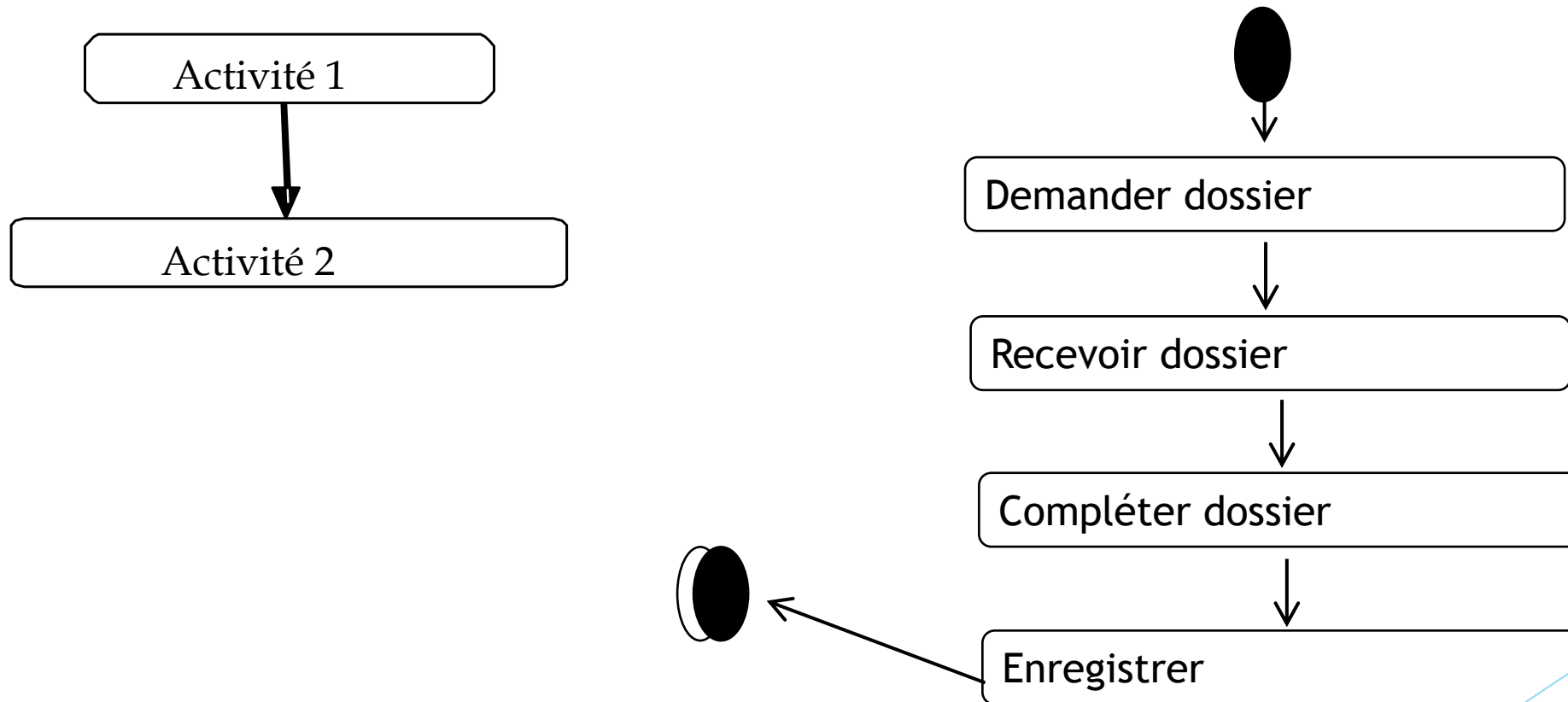
Exemple d'état d'action

# Les transitions

- ▶ La migration peut s'effectuer de manière automatique ou suspendue.
- ▶ Une transition automatique est :
  - ▶ traversée lorsque l'activité précédente est terminée.
- ▶ Transition surveillée :
  - ▶ se croise lorsque l'action précédente est terminée et
  - ▶ si la condition est vraie.

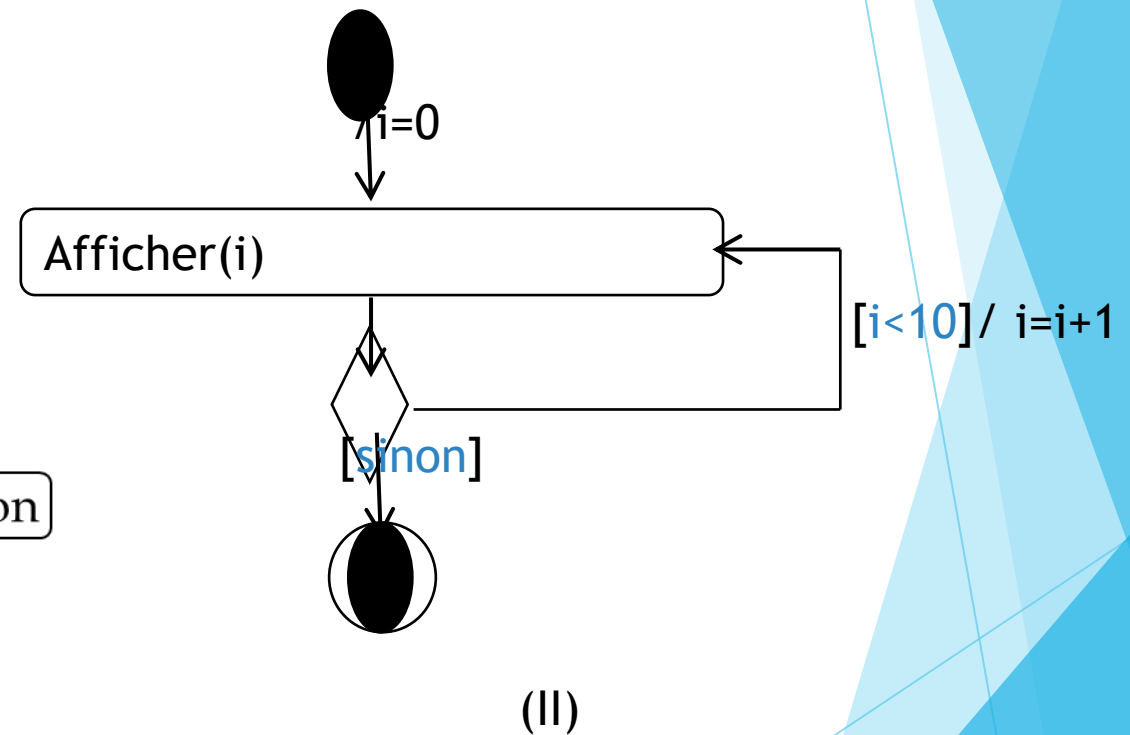
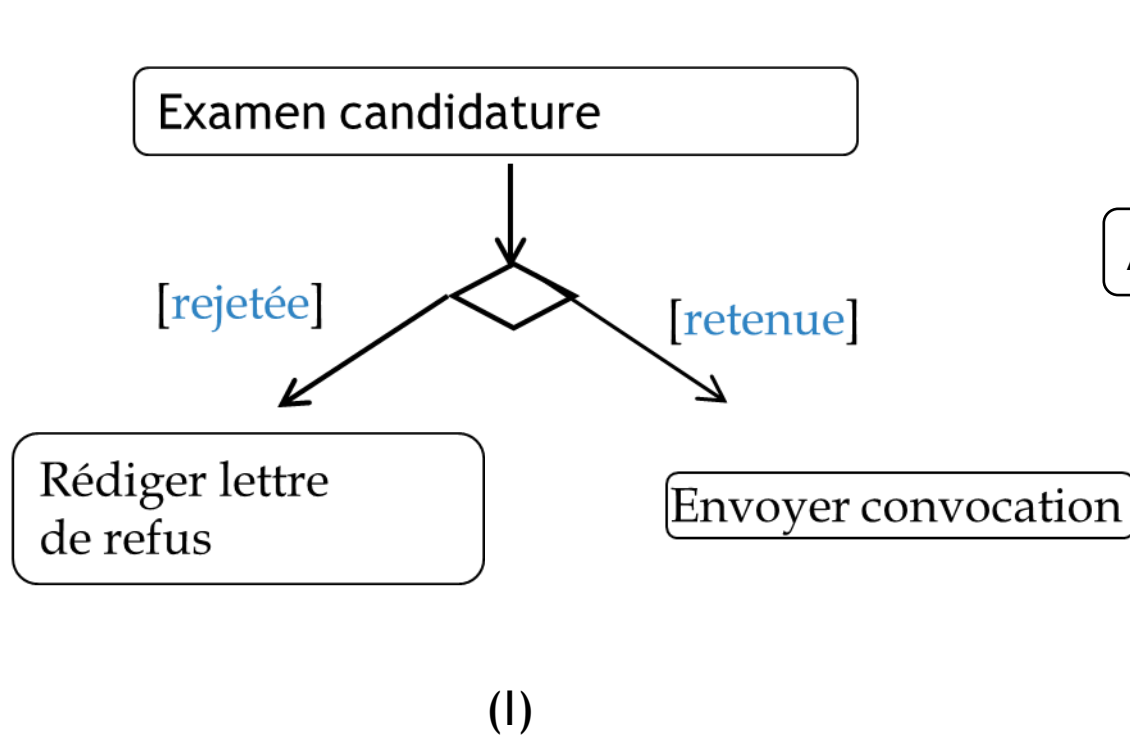
# Les transitions

## Transitions automatiques



# Les transitions

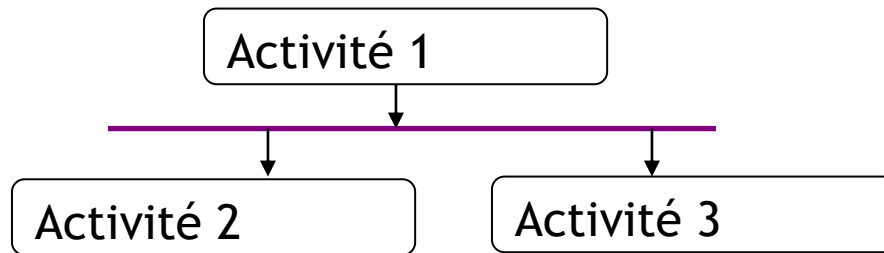
## Transitions **gardées**



Le niveau d'abstraction d'un DA peut être : de haut niveau (I) ou de bas niveau (II) : algorithmique

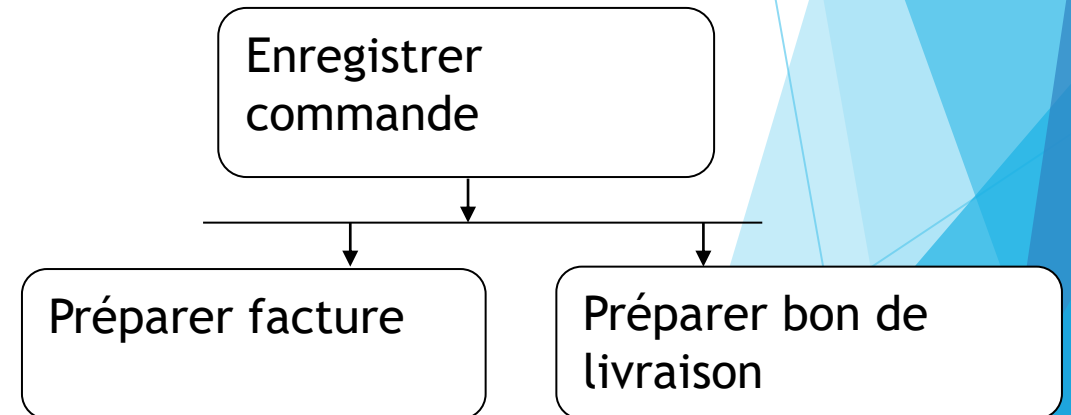
# Synchronisation

- ▶ Vous pouvez utiliser la barre de synchronisation (BS)
- ▶ pour synchroniser vos transitions. BS permet l'ouverture et/ou la fermeture de branches parallèles dans le flux d'exécution.
- ▶ Les transitions commençant à la ligne de synchronisation ont lieu simultanément.



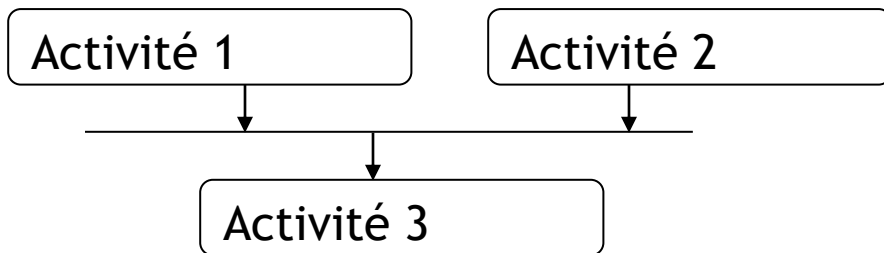
La fin de Activité 1 engendre les débuts **simultanés** de Activité 2 et Activité 3

HAMDI GHASSEN- 4 ème info



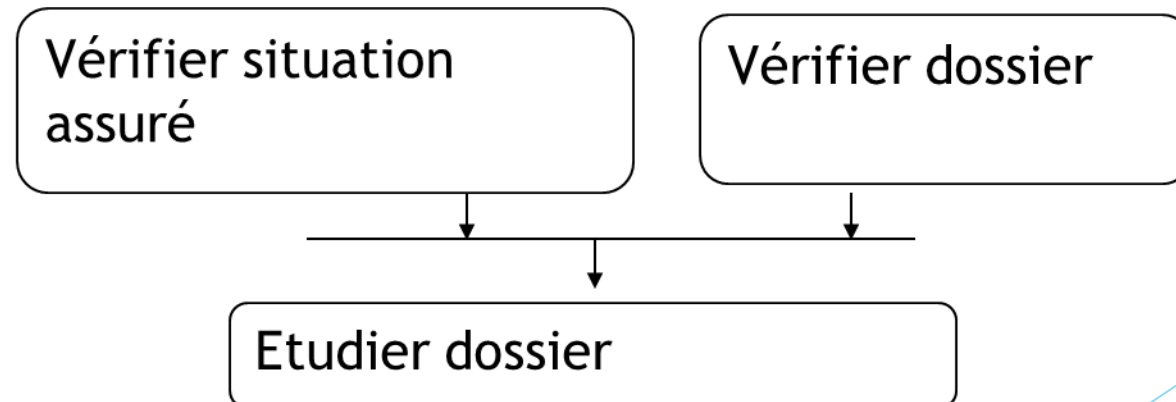
# Synchronisation

- ▶ Une ligne de synchronisation n'est franchie qu'après que toutes les transitions qui lui sont associées sont terminées.



L'action 3 ne démarre que lorsque les étapes 2 et 1 sont terminées

- ▶ **Exemple :**

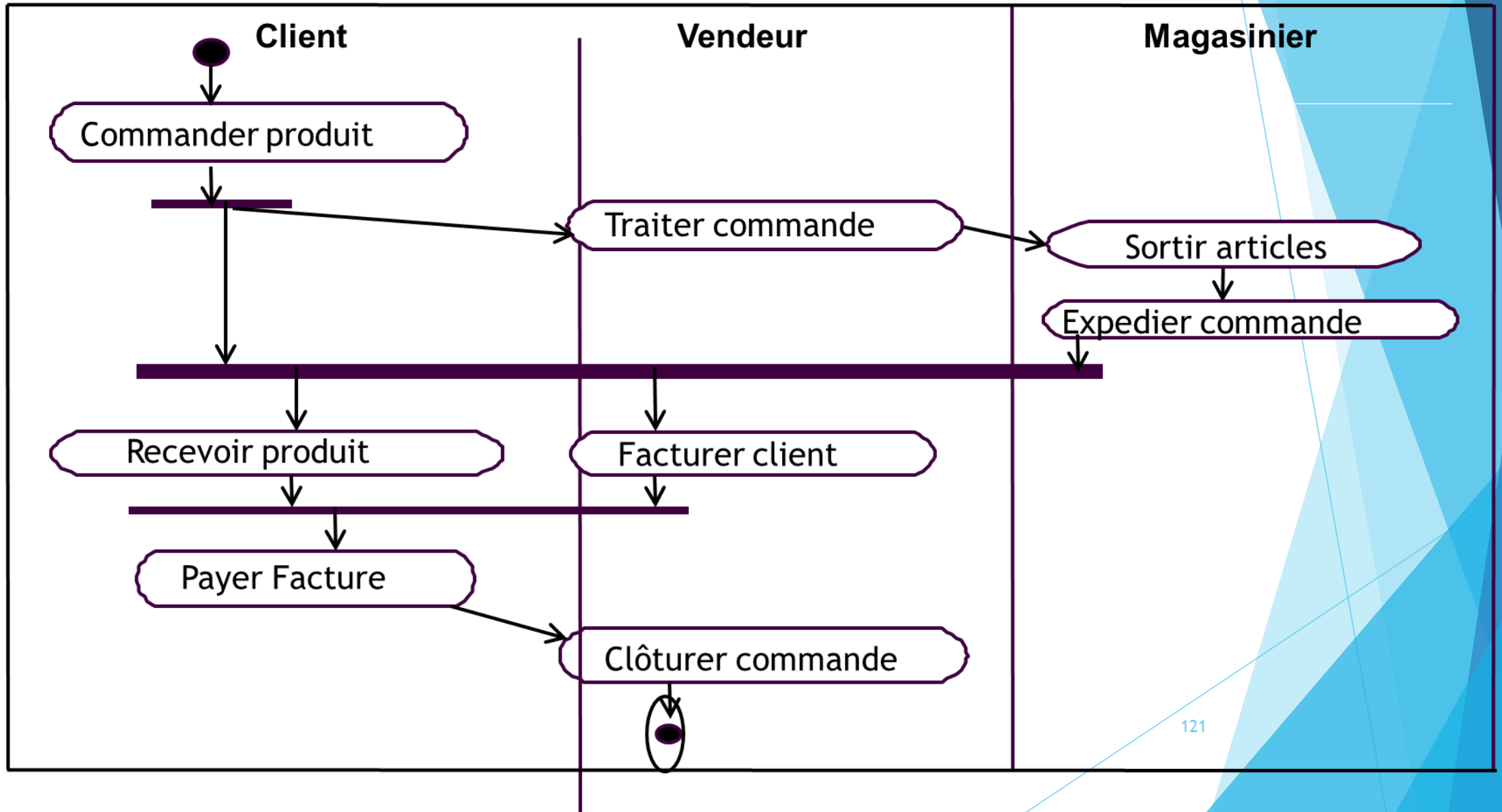




# Les travées

- ▶ Les diagrammes d'activités peuvent être fractionnés en sections (ou couloirs d'activités) pour illustrer la responsabilité (pour chaque classe/acteur) au sein d'un processus ou d'une structure organisationnelle.
- ▶ **Exemple : Traitement d'une commande client**
- ▶ 3 acteurs :
  - ▶ Client (commande la marchandise et paie),
  - ▶ Vendeur (remplit la commande et émet une facture à l'acheteur) et
  - ▶ Entrepôt (sélectionne la marchandise et envoie la commande)

## Couloirs d'activité



---

# Examen

---

## *Modélisation Orienté Objet*

### *UML*


---

**Niveau : 4<sup>ème</sup> Génie Informatique**

**Préparé Par : HAMDI Ghassen**

*Docteur en Sciences Informatique.*

**Année universitaire : 2020/2021**

Ecole Supérieure Privée d'Ingénierie et des Technologies Appliquées IHE-ESPITA		<b>Formulaire</b>	EXA-FR-02-00
		<b>Examen</b>	07/01/2021

Année Universitaire 2020 - 2021

Session : ☒ Principale / ☐ Rattrapage

## Matière : Modélisation Objet UML

<b>Enseignant</b>	: Ghassen HAMDI	<b>Date</b>	: 07/01/2021
<b>Filière</b>	: Génie Informatique	<b>Section</b>	: 4
<b>Barème</b>	: Ex. 1 : 10pts ; Ex. 2 : 10pts	<b>Documents</b>	: non autorisés
		<b>Calculatrice</b>	: non autorisée
<b>Durée/Ex.</b>	: Ex. 1 : 45mn ; Ex. 2 : 45mn	<b>Durée</b>	: 1h30mn
		<b>Nbre de pages</b>	: 2

### N.B.

Il sera tenu compte de la lisibilité, la présentation, et de la clarté des réponses.  
Les exercices sont indépendants et peuvent être traités dans n'importe quel ordre.  
Numérotez vos feuilles (par exemple 1/2, 2/2 ou 1/3, 2/3, 3/3 ou ...)

### Exercice 1 :

On considère le système suivant de gestion d'un DAB (Distributeur automatique de billets) :

- Le distributeur délivre de l'argent à tout porteur de carte (carte Visa ou carte de la banque)
- Pour les clients de la banque, il permet :
  - La consultation du solde du compte
  - Le dépôt d'argent (chèque ou numéraire)
- Toute transaction est sécurisée et nécessite par conséquent une authentification

Dans le cas où une carte est avalée par le distributeur, un opérateur de maintenance se charge de la récupérer. C'est la même personne qui collecte également les dépôts d'argent et qui recharge le distributeur.

**Modéliser cette situation par un diagramme de cas d'utilisation.**

### Exercice 2 :

On désire automatiser la gestion d'une petite bibliothèque municipale. Pour cela, on a analysé son fonctionnement pour obtenir la liste suivante de règles et d'affirmations :


- Les adhérents ont un prénom (chaîne de caractères) et un nom (chaîne de caractères).
- La bibliothèque comprend un ensemble de documents et un ensemble d'adhérents.
- De nouveaux documents sont ajoutés régulièrement à la bibliothèque.
- Ces documents sont soit des journaux, soit des volumes.
- Les volumes sont soit des dictionnaires, soit des livres, soit des BD.
- Les documents sont caractérisés par un titre (chaîne de caractères).

- Les volumes ont en plus un auteur (chaîne de caractères). Les Bd ont en plus un nom de destinataire (chaîne de caractères).
- Les journaux ont, outre les caractéristiques des documents, une date de parution (une date).
- Seuls les livres sont empruntables.
- Un adhérent peut emprunter ou restituer un livre.
- Les adhérents peuvent emprunter des livres (et uniquement des livres) et on doit pouvoir savoir à tout moment quels sont les livres empruntés par un adhérent.
- Un adhérent peut emprunter au plus 3 livres.
- La date de restitution d'un livre emprunté est fixée au moment du prêt. Cette date peut être prolongée sur demande.

### **Travail demandé**

Réalisez le diagramme de classes permettant d'automatiser la bibliothèque municipale.

Définissez les attributs et les méthodes de chaque classe de ce diagramme, ainsi que le type et les cardinalités des associations entre les classes.

Ecole Supérieure Privée d'Ingénierie et des Technologies Appliquées IHE-ESPITA		Formulaire	EXA-FR-02-00
		Correction Examen	07/01/2021

Année Universitaire 2020 - 2021

Session : ☒ Principale / ☐ Rattrapage

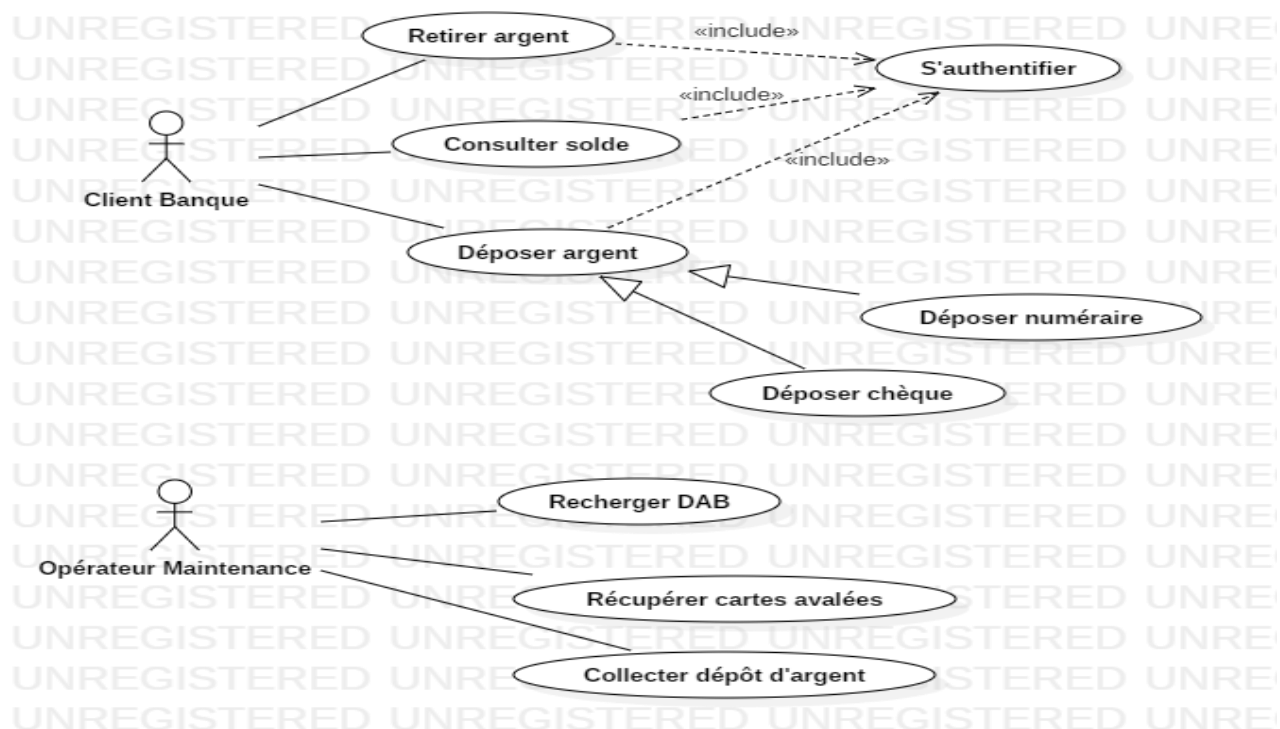
## Matière : Modélisation Objet UML

<b>Enseignant</b>	: Ghassen HAMDI	<b>Date</b>	: 07/01/2021
<b>Filière</b>	: Génie Informatique	<b>Section</b>	: 4
<b>Barème</b>	: Ex. 1 : 10pts ; Ex. 2 : 10pts	<b>Documents</b>	: non autorisés
		<b>Calculatrice</b>	: non autorisée
<b>Durée/Ex.</b>	: Ex. 1 : 45mn ; Ex. 2 : 45mn	<b>Durée</b>	: 1h30mn
		<b>Nbre de pages</b>	: 2

### N.B.

Il sera tenu compte de la lisibilité, la présentation, et de la clarté des réponses.  
Les exercices sont indépendants et peuvent être traités dans n'importe quel ordre.  
Numérotez vos feuilles (par exemple 1/2, 2/2 ou 1/3, 2/3, 3/3 ou ...)

### Exercice 1 :



## Exercice 2 :

