



# Conception Orientée Objet

## 3IIR

---

### Partie I

### Diagrammes de base

**Pr. Khalid SRAIDI**

**2022/2023**

# Introduction

Les technologies de programmation n'ont pas cessé d'évoluer depuis l'apparition des ordinateurs. Cette évolution est fortement sollicitée pour répondre à un besoin incessant de produire des applications plus complexes.

La principale avancée des vingt dernières années réside dans la **programmation orientée objet (P.O.O.)**. Face à ce nouveau mode de programmation, de très nombreuses méthodes ont vu le jour comme Booch, OMT ...

Dans ce contexte et devant le foisonnement de nouvelles méthodes de conception « **orientée objet** », l'Object Management Group (OMG) a eu comme objectif de définir une notation standard utilisable dans les développements informatiques basés sur l'objet.



*C'est ainsi qu'est apparu UML.....*

## Définition

“

«Le Langage de Modélisation Unifié, de l'anglais **Unified Modeling Language (UML)**, est un langage de modélisation graphique à base de pictogrammes conçu pour fournir des modèles normalisés pour visualiser la conception d'un système. Il est couramment utilisé en développement logiciel et en conception orientée objet»



# Versions UML

2.5.1	Décembre 2017
2.4.1	Juillet 2011
2.3	Mai 2010
2.2	Janvier 2009
2.1.2	Octobre 2007
2.0	Juillet 2005
1.5	Mars 2003
1.4	Septembre 2001
1.3	Février 2000
1.2	Juillet 1999
1.1	Décembre 1997

## UML en oeuvre

UML n'est pas une méthode. Il est un langage de modélisation pour fournir des modèles de conception orientée objet unifiés afin de :

- **Visualiser**

- Chaque symbole graphique possède une sémantique.

- **Spécifier**

- De manière précise et complète, sans ambiguïté.

- **Construire**

- Une partie du code des classes peut être généré automatiquement.

- **Documenter**

- Les différents diagrammes, notes, contraintes, exigences sont conservés dans un document.

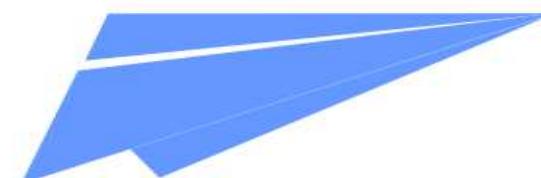
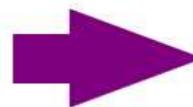
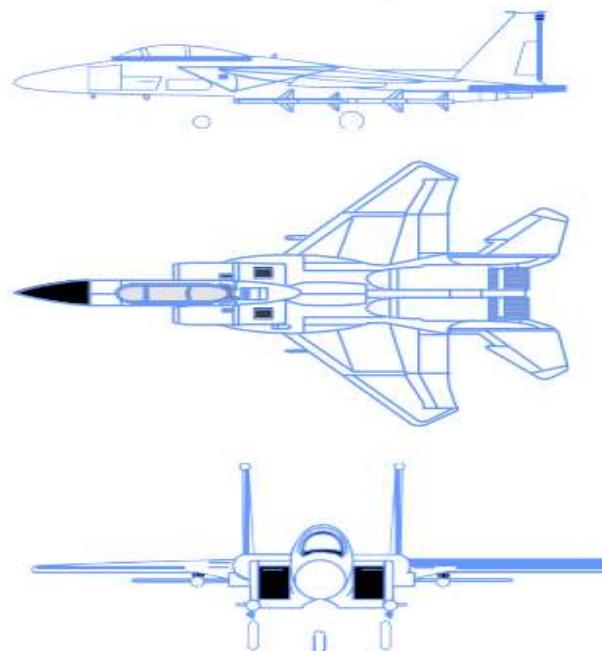
## Qu'est ce qu'un modèle ?

“

«Un modèle est une abstraction de la réalité. Il définit une frontière entre la réalité et la perspective de l'observateur. Ce n'est pas "la réalité", mais une vue très subjective de la réalité. Bien qu'un modèle ne représente pas une réalité absolue, un modèle reflète des aspects importants de la réalité, il en donne donc une vue juste et pertinente»

# Qu'est ce qu'un modèle ?

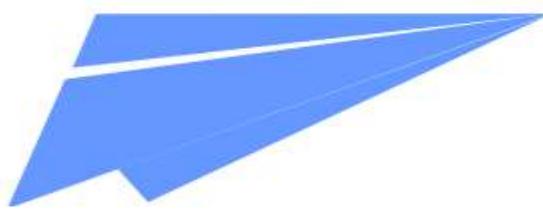
→ Une simplification de la réalité



# Importance d'un modèle ?

moins important

Plus important



Avion papier



Avion militaire

Le développement logiciel AUSSI nécessite des modèles bien pensés !

## Importance de la modélisation

- Le modèle peut remplacer une longue **description textuelle** par un **schéma ou diagramme** plus facile et plus rapide à interpréter.
- Le code tout seul ne suffit pas. **Il faut une documentation compréhensible et non ambiguë** pour assurer une bonne communication entre les différents intervenants.
- La modélisation permet de morceler une application en fragments maitrisable.

# **UML est un support de communication**

## → **Performance de sa notation graphique**

Il permet de s'exprimer clairement à l'aide des concepts objets

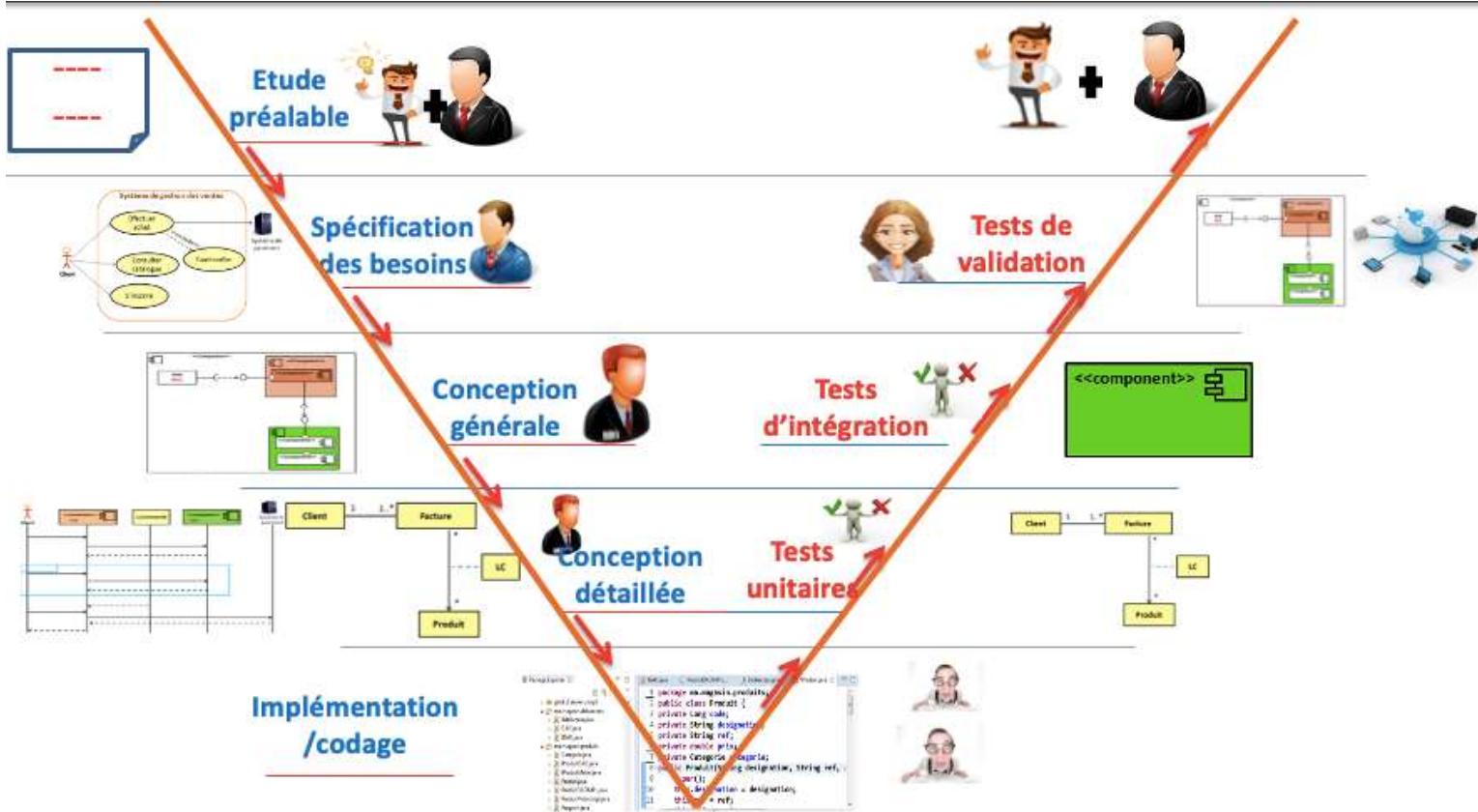
## → **Limiter les ambiguïtés**

Parler un langage commun, au vocabulaire précis.

## → **Indépendance**

Il est indépendant des langages de programmation, du domaine d'application, du processus de développement etc, ce qui le rend un langage universel.

# UML et Cycle de vie d'un logiciel

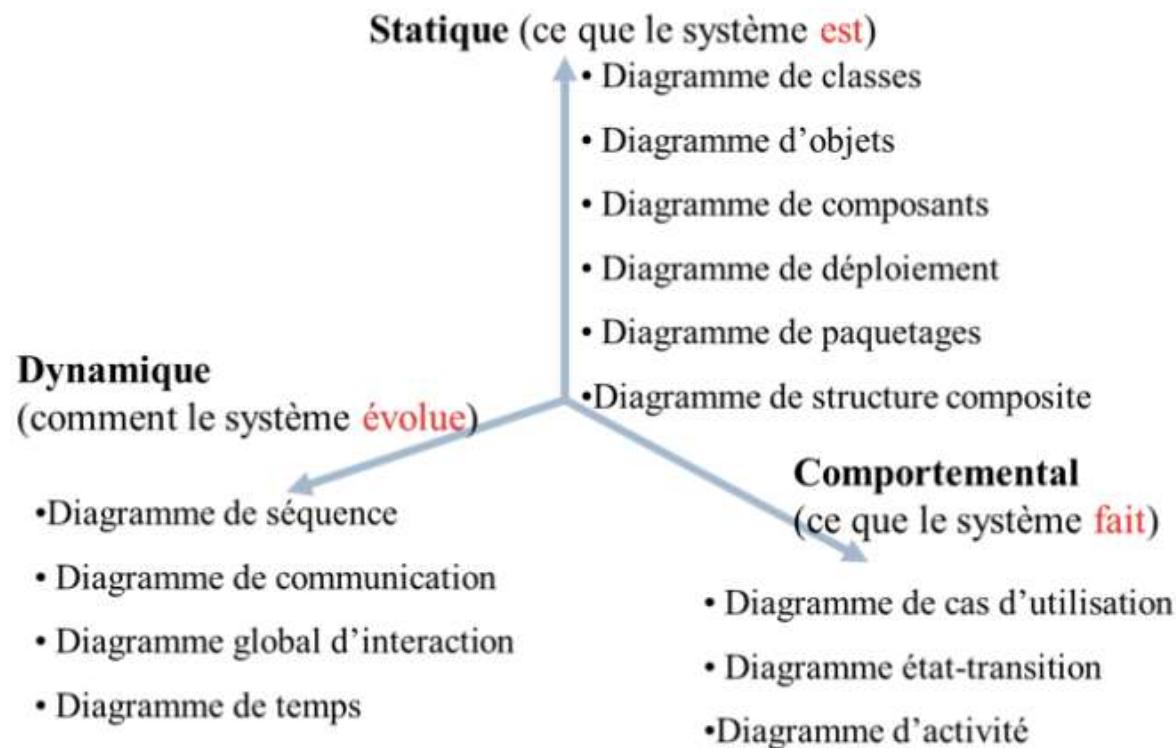


## Les catégories des diagrammes UML

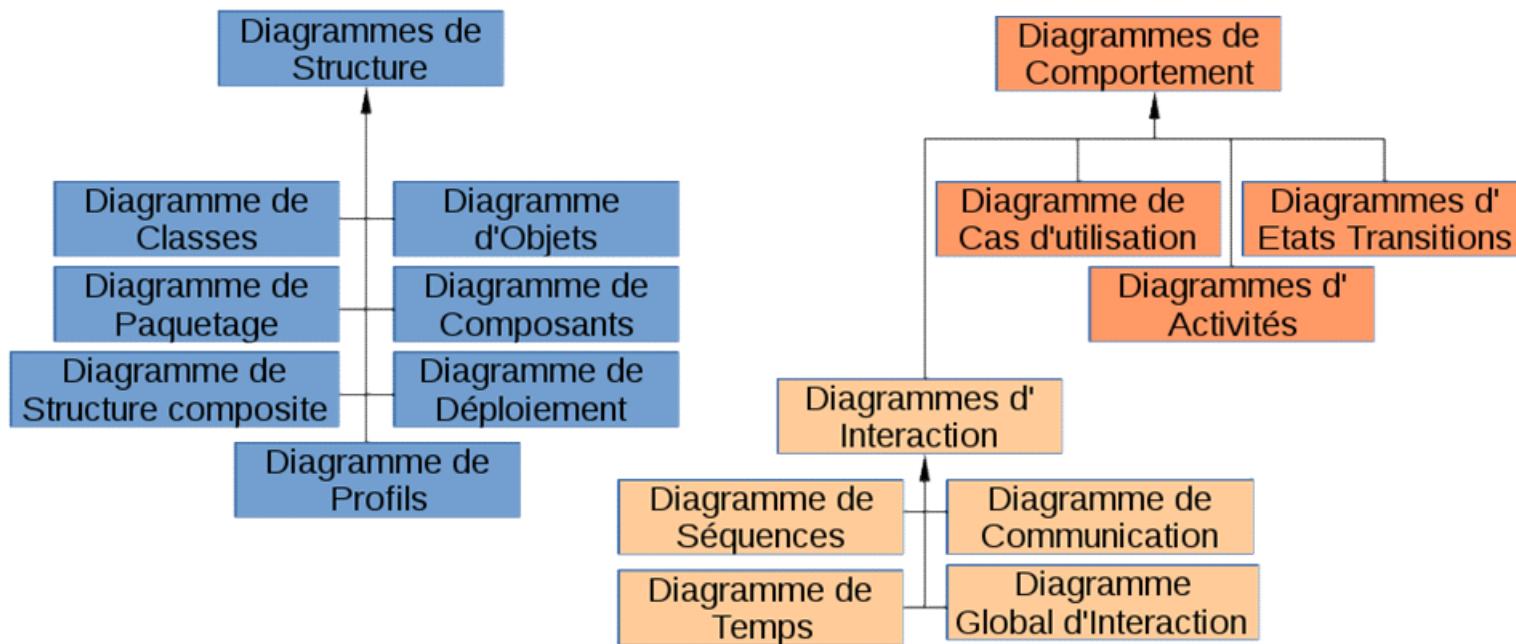
Les diagrammes UML sont tous réalisés à partir du besoin des utilisateurs et peuvent être regroupés selon les deux aspects suivants :

- ✓ **Les aspects fonctionnels** : Qui utilisera le logiciel et pour quoi faire ? Comment les actions devront-elles se dérouler ? Quelles informations seront utilisées pour cela ?
  
- ✓ **Les aspects liés à l'architecture** : Quels seront les différents composants logiciels à utiliser (base de données, librairies, interfaces, etc.) ? Sur quel matériel chacun des composants sera installé ?

# Les catégories des diagrammes UML



# Les diagrammes UML depuis UML 2.3



# UML: Diagrammes de Comportement

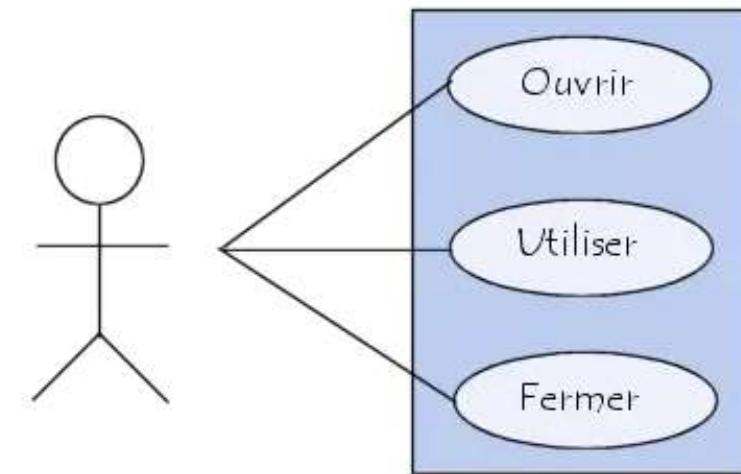
Les diagrammes de comportement (behavior diagrams) rassemblent :

- **Diagramme de cas d'utilisation (use case diagram).**
- **Diagramme d'état-transition (state machine diagram).**
- **Diagramme d'activité (activity diagram).**

## UML: Diagrammes de Comportement

### Diagramme de cas d'utilisation (use case diagram)

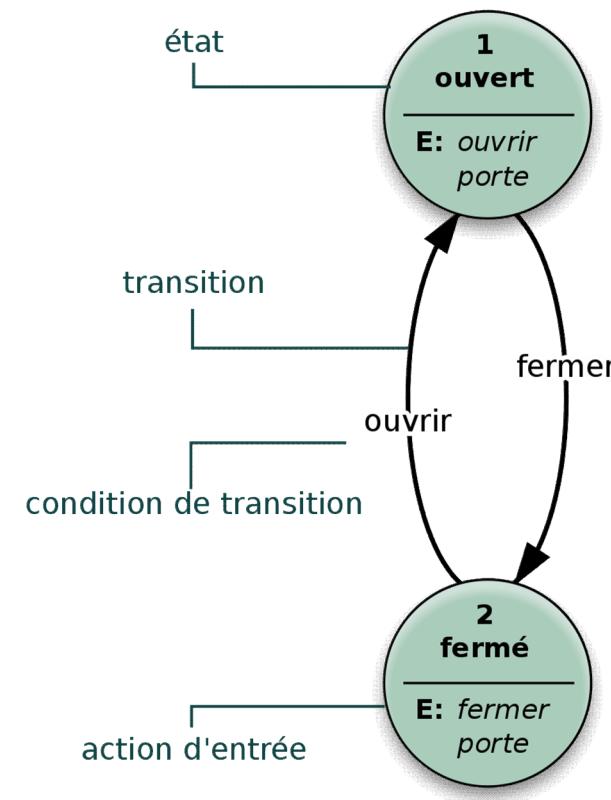
Il représente les interactions entre le système et les acteurs (intervenants extérieurs au système), c'est-à-dire toutes les fonctionnalités que doit fournir le système.



## UML: Diagrammes de Comportement

### Diagramme état-transition (state machine diagram)

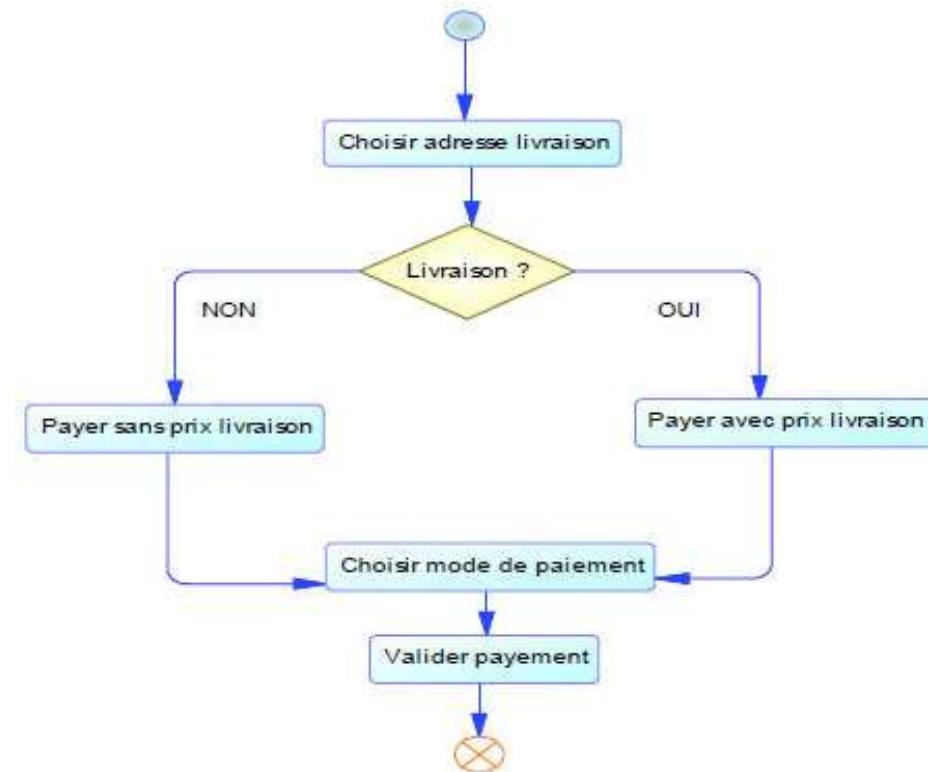
Il représente sous forme de machine à états finis le comportement du système ou de ses composants.



## UML: Diagrammes de comportement

### Diagramme d'activité (activity diagram)

IL représente sous forme de flux ou d'enchaînement d'activités le comportement du système ou de ses composants



# UML: Diagrammes Statiques

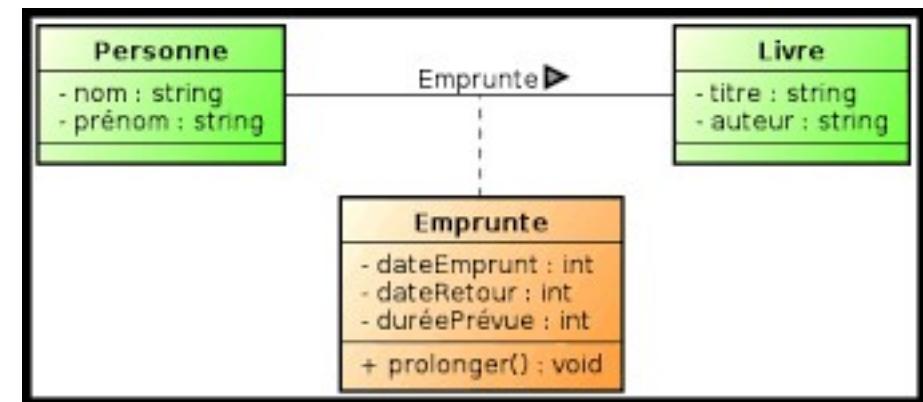
Les diagrammes statiques (static diagrams) ou diagrammes de structure (structure diagrams) rassemblent :

- **Diagramme de classes (class diagram).**
- **Diagramme d'objets (object diagram).**
- **Diagramme de composants (component diagram).**
- **Diagramme de déploiement (deployment diagram).**
- **Diagramme des paquets (package diagram).**
- **Diagramme de structure composite (composite structure diagram).**
- **Diagramme de profils (profile diagram)**

# UML: Diagrammes Statiques

## Diagramme de classes (class diagram)

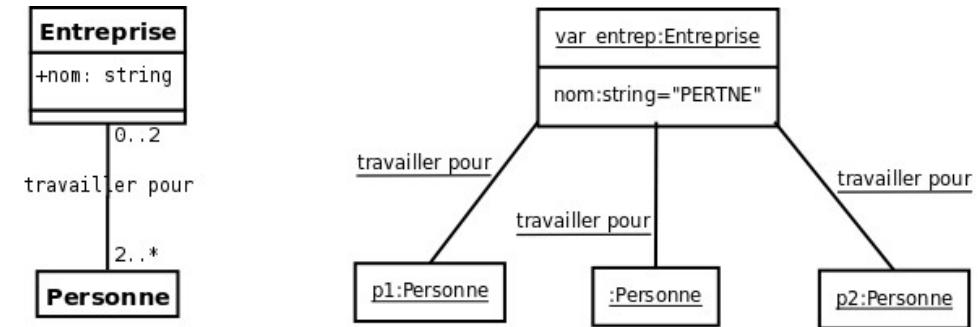
Il représente les classes intervenantes dans le système.



# UML: Diagrammes Statiques

## Diagramme d'objet (Object diagram)

Il représente les objets qui sont des instances des classes et leurs liens pour donner les états de ces objets dans un contexte d'exécution.

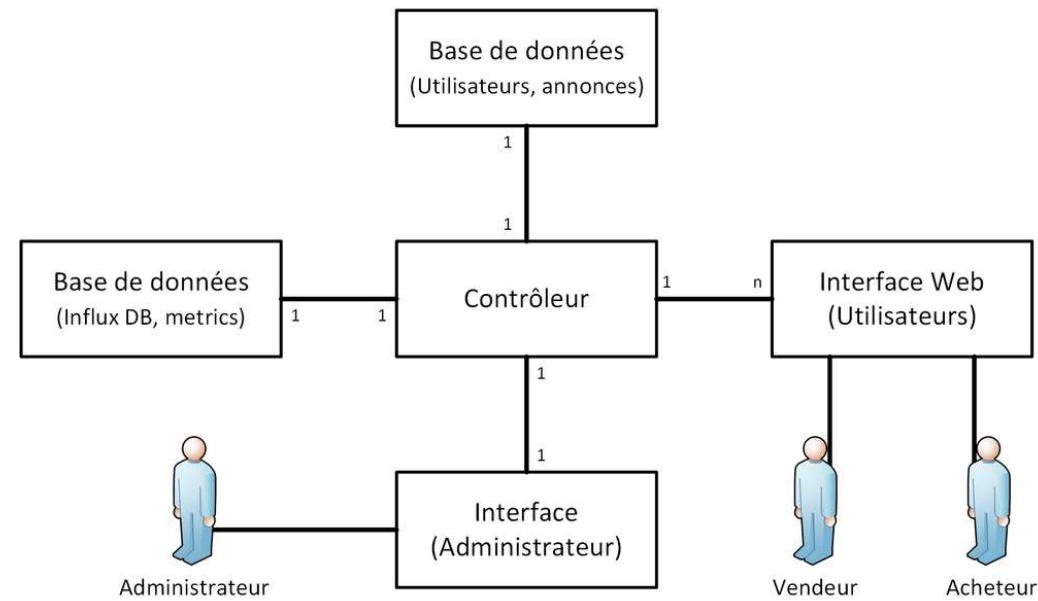


# UML: Diagrammes Statiques

## Diagramme de composant (component diagram)

Il représente les composants logiciels du système d'un point de vue physique, tels qu'ils sont mis en œuvre (fichiers, bibliothèques, bases de données...).

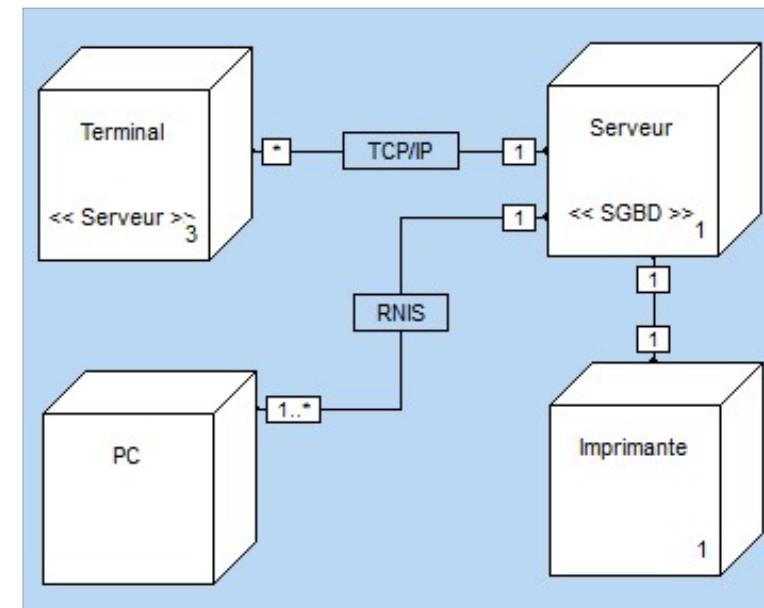
## Diagramme de composants



## UML: Diagrammes Statiques

### Diagramme de déploiement (deployment diagram)

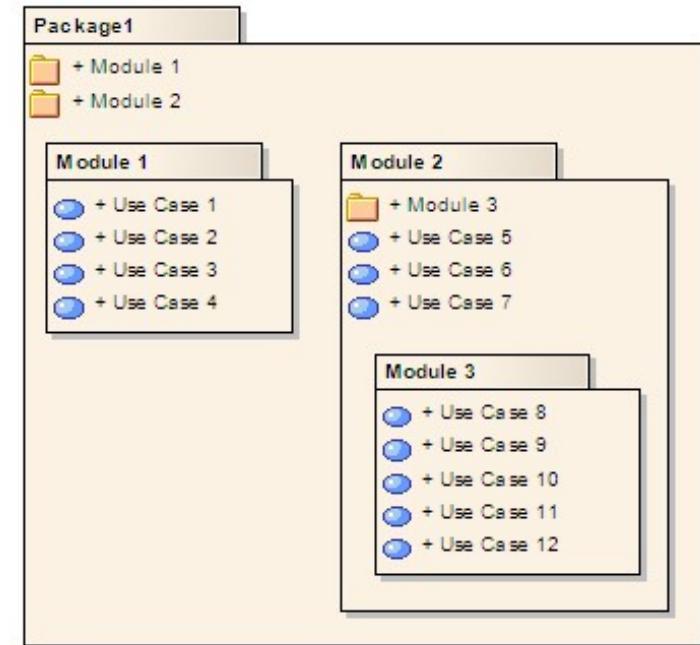
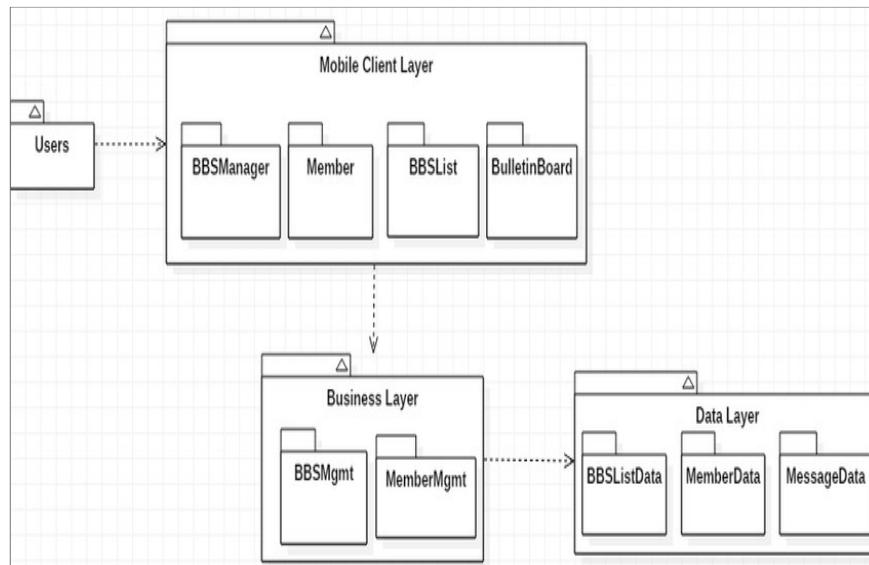
Il représente des éléments matériels (ordinateurs, périphériques, réseaux, systèmes de stockage...) et la manière dont les composants du système sont installés sur ces éléments matériels et interagissent entre eux.



# UML: Diagrammes Statiques

## Diagramme de Paquetage (package diagram)

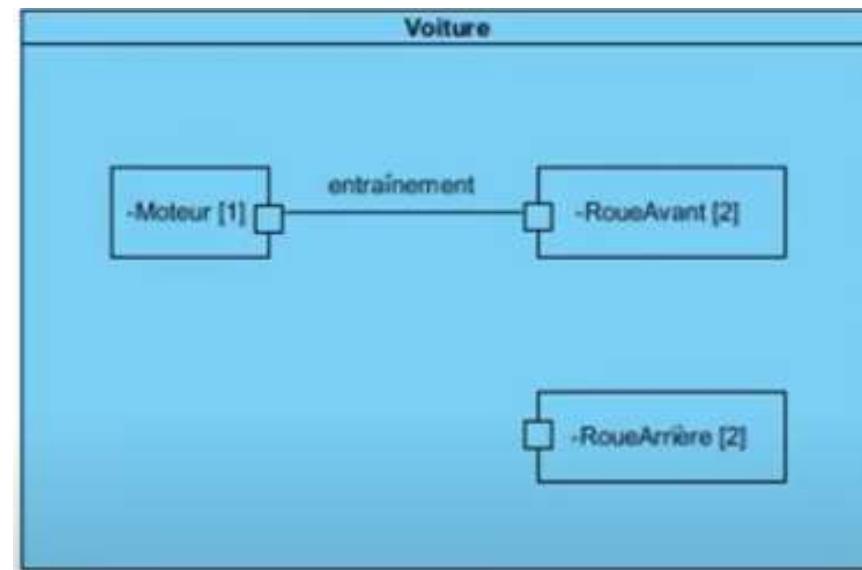
Il représente des dépendances entre les paquets (un paquet étant un conteneur logique permettant de regrouper et d'organiser les éléments dans le modèle UML), c'est-à-dire entre les ensembles de définitions.



## UML: Diagrammes Statiques

### Diagramme de structure composite (composite structure diagram)

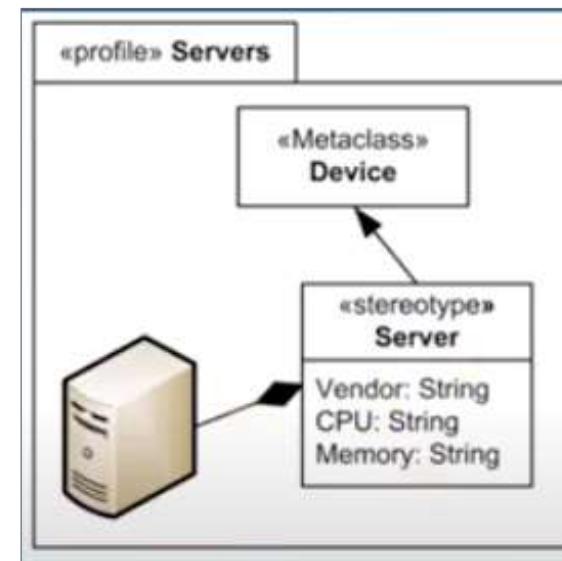
Il représente sous forme de boîte blanche des relations entre composants d'une classe (depuis UML 2.x).



# UML: Diagrammes Statiques

## Diagramme de profils (profile diagram)

Ce diagramme fournit une représentation des concepts utilisés dans la définition des profils (packages, stéréotypes, application de profils, etc.)



# UML: Diagrammes Dynamiques

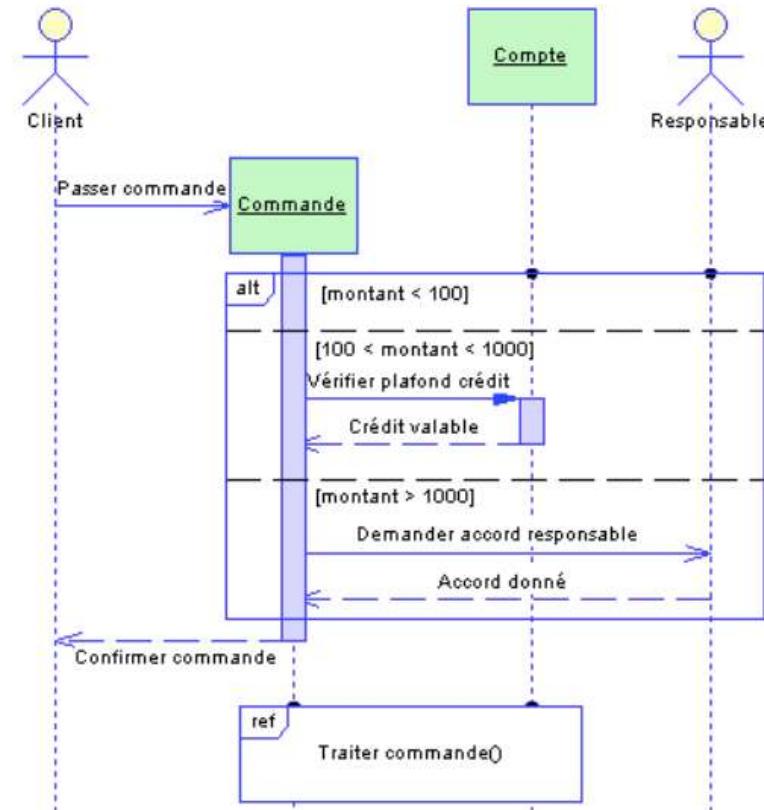
Les diagrammes dynamiques (dynamic diagrams) ou d'interaction (interaction diagrams) rassemblent :

- Diagramme de séquence (sequence diagram).
- Diagramme de communication (communication diagram).
- Diagramme global d'interaction (interaction overview diagram).
- Diagramme de temps (timing diagram) .

# UML: Diagrammes Dynamiques

## Diagramme de séquence (sequence diagram)

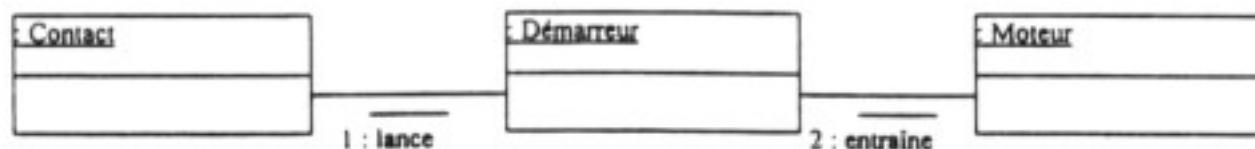
Représentation de façon séquentielle du déroulement des traitements et des interactions entre les éléments du système et/ou de ses acteurs.



## UML: Diagrammes Dynamiques

### Diagramme de communication (communication diagram)

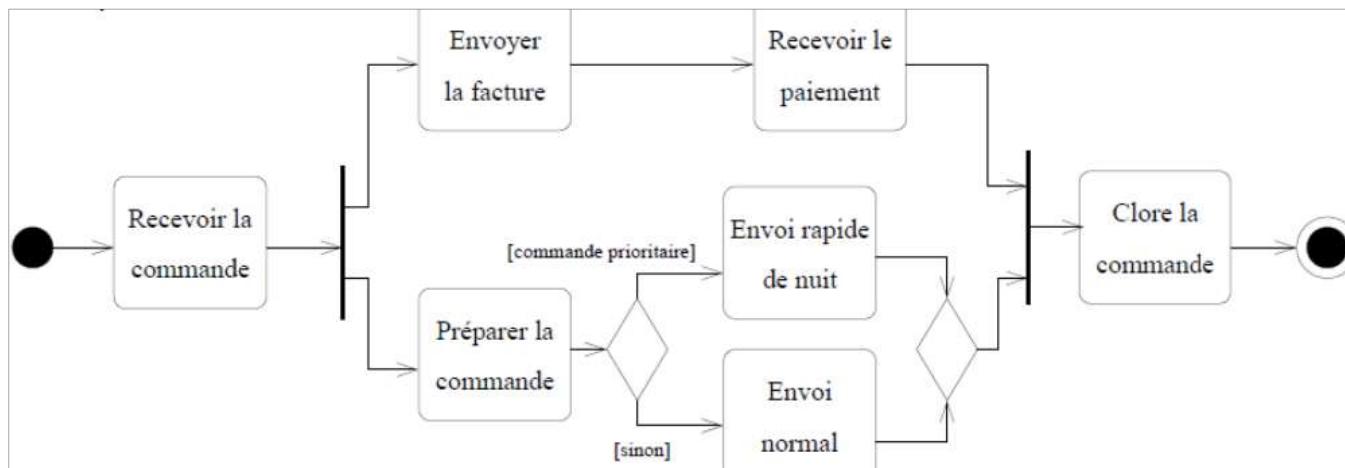
Représentation de façon simplifiée d'un diagramme de séquence se concentrant sur les échanges de messages entre les objets (depuis UML 2.x).



# UML: Diagrammes Dynamiques

## Diagramme global d'interaction (interaction overview diagram)

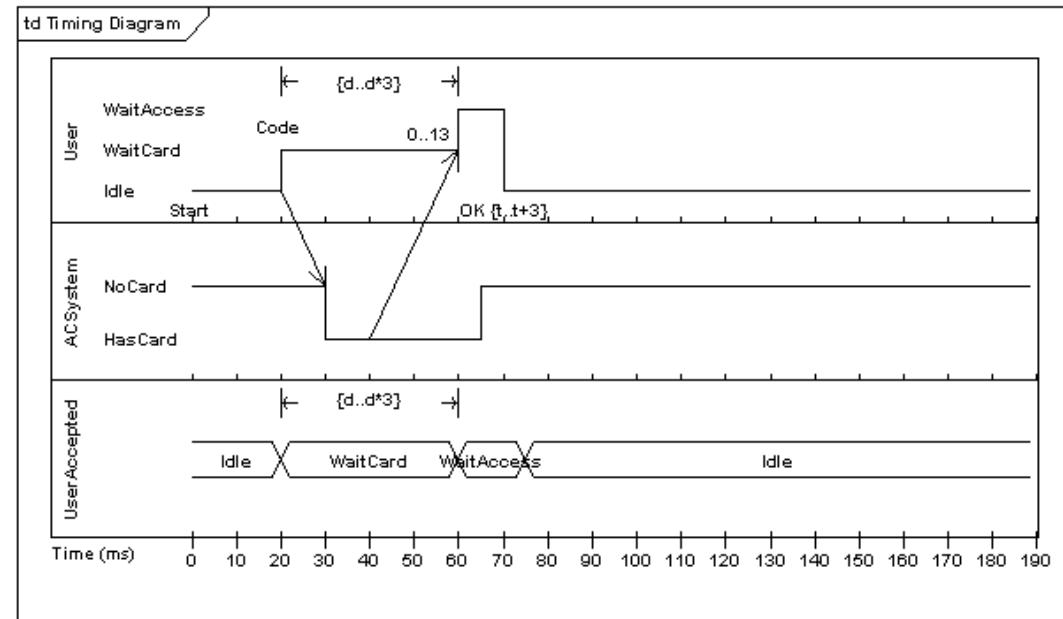
Représentation des enchaînements possibles entre les scénarios préalablement identifiés sous forme de diagrammes de séquences.



# UML: Diagrammes Dynamiques

## Diagramme de temps (timing diagram)

Représentation des variations d'une donnée au cours du temps (depuis UML 2.3).





## UML en pratique

- Les diagrammes UML sont tous utiles selon le besoin mais ils ne sont pas nécessairement tous produits dans un même projet.
- Les diagrammes de **cas d'utilisation, d'activités, de classes, d'objets, de séquence et d'états-transitions** sont utiles pour **modéliser les exigences fonctionnelles** du projet .
- Les diagrammes de **composants, de déploiement et de communication** sont utiles pour **modéliser les exigences techniques** du projet.

## Chap 2: Diagramme de cas d'utilisation

“

«Il décrit les fonctionnalités d'un système d'un point de vue utilisateur. Il se base sur le cahier des charges pour être construit. Le but est d'identifier les catégories d'utilisateurs : chacune d'entre elles, appelée acteur, est susceptible de mettre en œuvre une ou plusieurs fonctionnalités du système ; et les besoins du système : chaque fonctionnalité, appelée cas d'utilisation, doit répondre à l'un des besoins nécessités par une ou plusieurs catégories d'utilisateurs.»

## Eléments constitutifs: Acteur

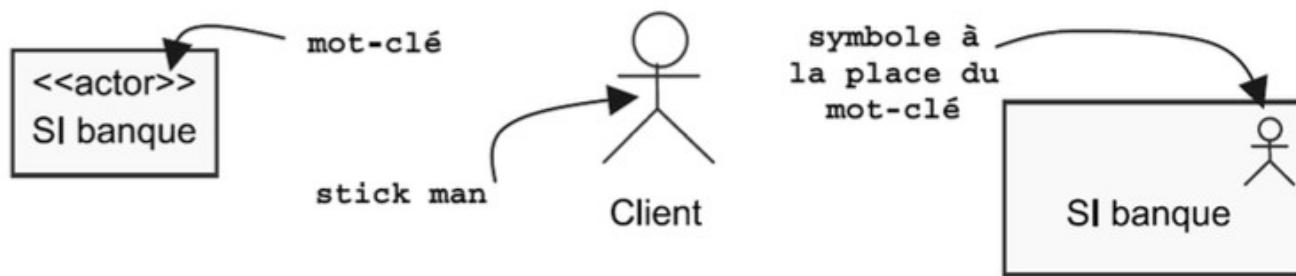
Un acteur représente un rôle joué par une entité externe (utilisateur humain, dispositif matériel ou autre système) qui interagit directement avec le système étudié. Il existe trois types d'acteurs:

- **Humain** : utilisateur du système, au travers des différentes interfaces IHM (logicielle ou matérielle) ; exemple : client, administrateur, technicien, etc. ;
- **Logiciel** : entité logicielle existante et fonctionnelle qui communique avec le système grâce à une interface logicielle ; exemple : application de gestion, base de données, etc.,
- **Matériel** : entité matérielle qui exploite les données du système, ou est pilotée par le système ; exemple : imprimante, robot, serveur, etc.

Pour identifier les acteurs, on se fonde sur les frontières du système.

# Représentation graphique d'un acteur

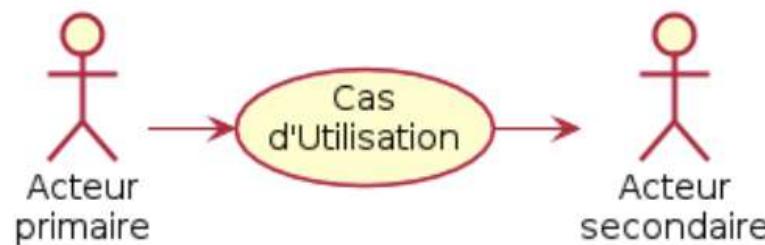
- La représentation graphique standard de l'acteur en UML est l'icône appelée stick man, avec le nom de l'acteur sous le dessin.
- On peut également figurer un acteur sous la forme rectangulaire d'une classe, avec le mot-clé «actor».



- Une bonne recommandation consiste à faire prévaloir l'utilisation de la forme graphique du stick man pour les acteurs humains et une représentation rectangulaire pour les systèmes connectés.

## Acteur principal ou secondaire

- Les acteurs demandant des services aux systèmes, ils sont le plus souvent à l'initiative des échanges avec le système : ils sont dits **acteurs primaires**.
- Lorsqu'ils sont sollicités par le système (dans le cas de serveurs externes par exemple), ils sont dits **acteurs secondaires**.



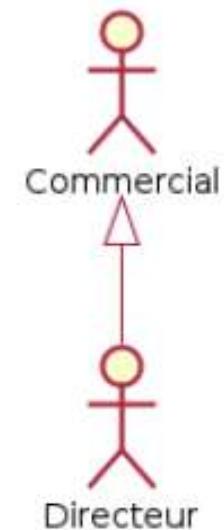
## Relations entre acteurs

Il n'y a qu'un seul type de relation possible entre acteurs : la relation de **généralisation**.

Il y a généralisation entre un cas A et un cas B lorsqu'on peut dire : A est une sorte de B.

**Exemple :**

Un directeur est une sorte de commercial : il peut faire avec le système tout ce que peut faire un commercial, plus d'autres choses



## Eléments constitutifs Un cas d'utilisation

“

«Un cas d'utilisation est un service rendu à un acteur : c'est une fonctionnalité d'un système»

**Graphiquement:** un cas d'utilisation est représenté par une ellipse comportant un verbe à l'infinitif.



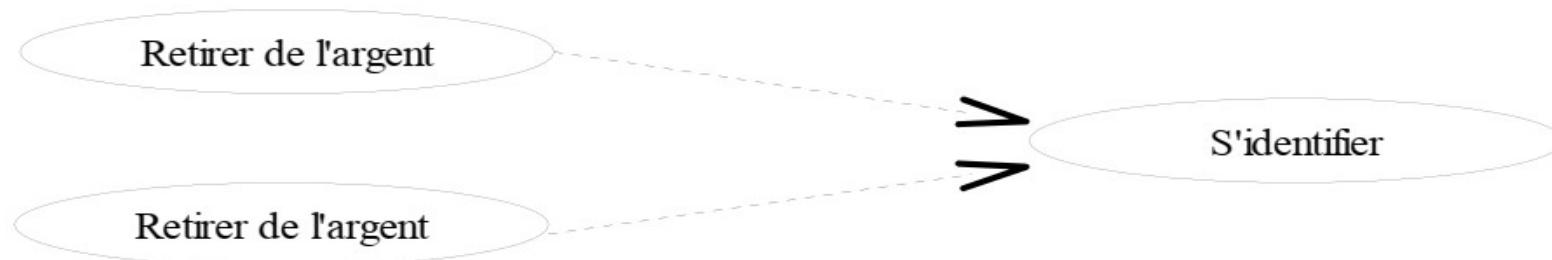
# Relation entre cas d'utilisation

Il y a trois types de relation entre cas d'utilisation :

i°) **Inclusion:** B est une partie obligatoire de A et on lit A inclut B (dans le sens de la flèche).

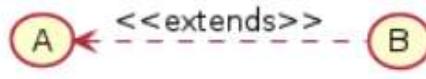


*Exemple :*



# Relation entre cas d'utilisation

ii°) Extension: B est une partie optionnelle de A et on lit B étend (prolonge) A (dans le sens de la flèche).

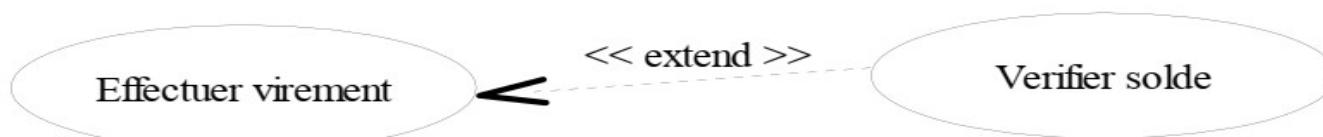


Autrement dit, un cas d'utilisation B étend un cas d'utilisation A, lorsque le cas d'utilisation B peut être appelé au cours de l'exécution du cas d'utilisation A.

## Exemple



Si le client existe on n'a pas besoin de le créer.



Dans certains cas, on peut être appelé à vérifier le solde lors d'un virement.

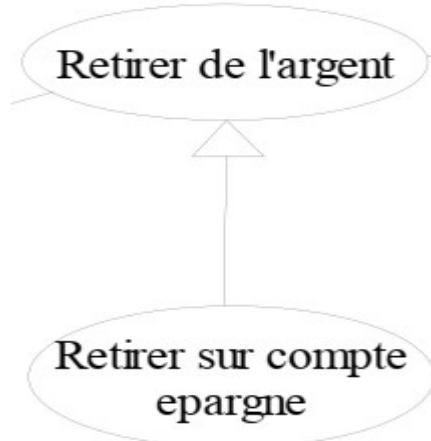
# Relation entre cas d'utilisation

iii°) **Généralisation:** le cas A est une généralisation du cas B et on lit B est une sorte de A, si B est un cas particulier de A, c'est à dire lorsque A peut être substitué par B pour un cas

e.



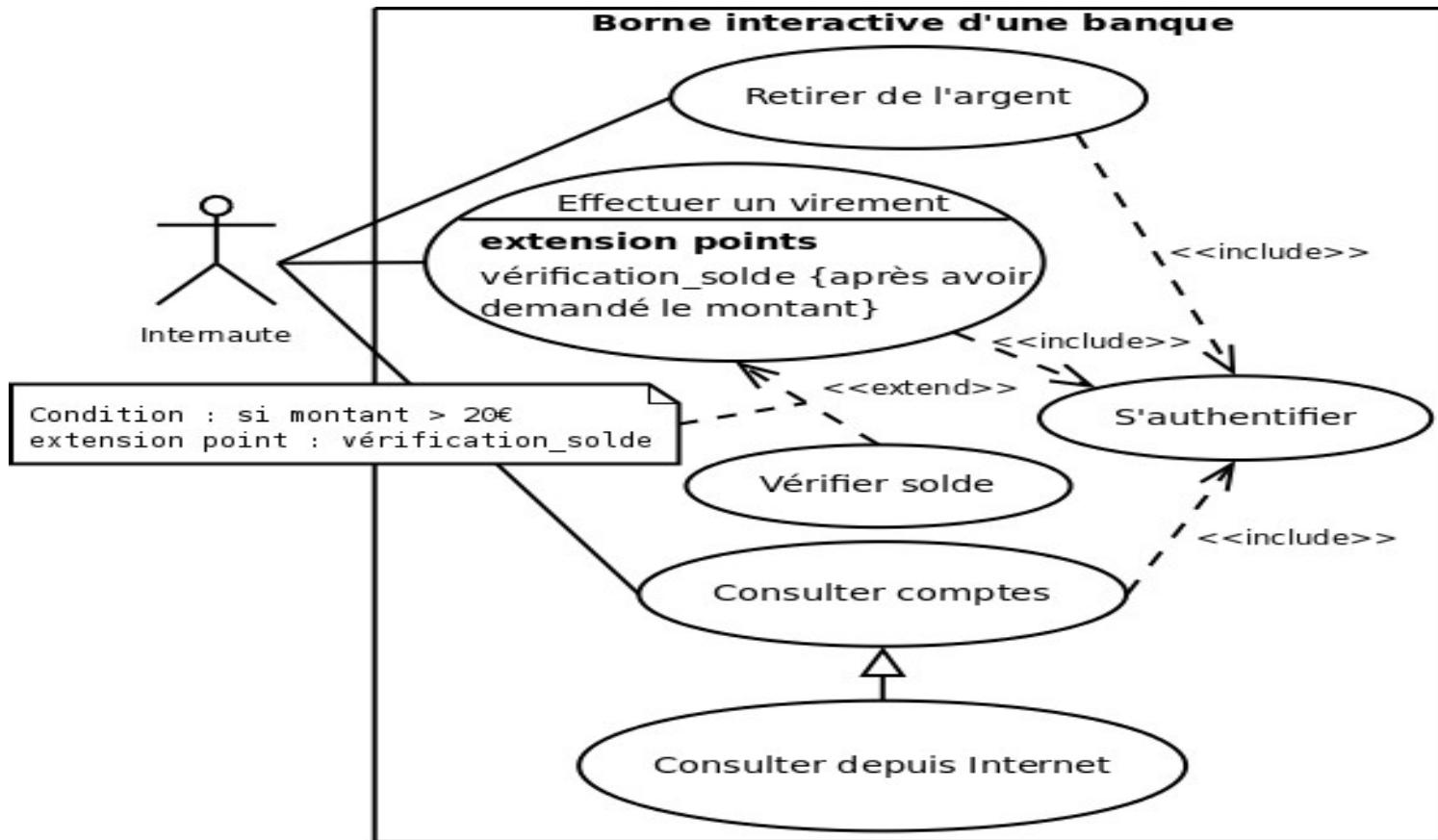
## Exemple



Les flèches en pointillés dénotent en fait une relation de dépendance, et les mentions *includes* et *extends* sont des stéréotypes et à ce titre placés entre guillemets.

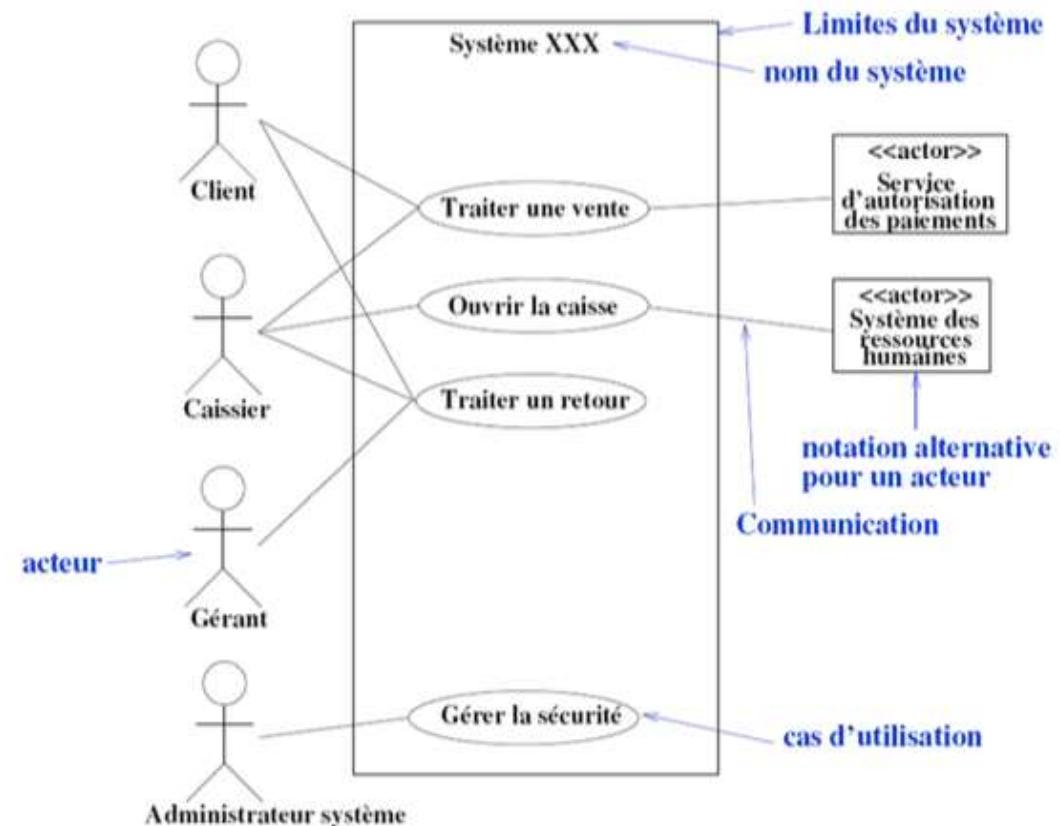
N.B : Contrainte au deux autres relations. La généralisation n'est pas un stéréotype.

## Relation entre cas d'utilisation: Exemple



# Système

Un système représente une application dans le modèle UML. Il est identifié par un nom et regroupe un ensemble de cas d'utilisation qui correspondent aux fonctionnalités offertes par l'application à son environnement. L'environnement est spécifié sous forme d'acteurs liés aux cas d'utilisation. Un système se représente par un rectangle contenant le nom du système et les cas d'utilisation de l'application.



# Identification d'un cas d'utilisation

- ❑ Il faut se placer du point de vue de chaque acteur et déterminer :
  - comment il se sert du système,
  - dans quels cas il l'utilise,
  - à quelles fonctionnalités il doit avoir accès.
- ❑ Pour chaque acteur, il convient de :
  - Rechercher les différentes intentions avec lesquelles il utilise le système
  - Déterminer dans le cahier des charges, les services fonctionnels attendus du système
- ❑ Les cas d'utilisation sont repérés à partir du cahier du charge
- ❑ Il n'y a pas une manière unique et totalement objective de repérer les cas d'utilisation

## Exercice

Un magasin XYZ a besoin de mettre en place un système de gestion des ventes en ligne.

Ce système doit permettre à ces utilisateurs les fonctionnalités suivantes:

- Le client doit être capable de consulter le catalogue et créer un bon de commande.
- Si le client possède une carte bancaire, le système doit aussi lui offrir la possibilité d'effectuer le paiement en ligne à l'aide d'un système de paiement en ligne.
- Pour une commande non payée, le commercial doit prendre contact avec le client pour établir les procédures

de paiement. Une fois le paiement reçu, le commercial valide la vente.

- Pour une commande payée et non livrée, le magasinier livre le client et enregistre les informations concernant la livraison sur le système.
- Le magasinier peut aussi, créer un nouveau produit s'il n'existe pas dans le catalogue.
- Le comptable doit être capable de visualiser les ventes et les chiffres d'affaires réalisés.
- Le directeur peut consulter des rapports fournis par le système pour les analyser.

# Questions

1. Identifier les acteurs.
2. Identifier les cas d'utilisation
3. Donnez le diagramme de cas d'utilisation.

# Exercice

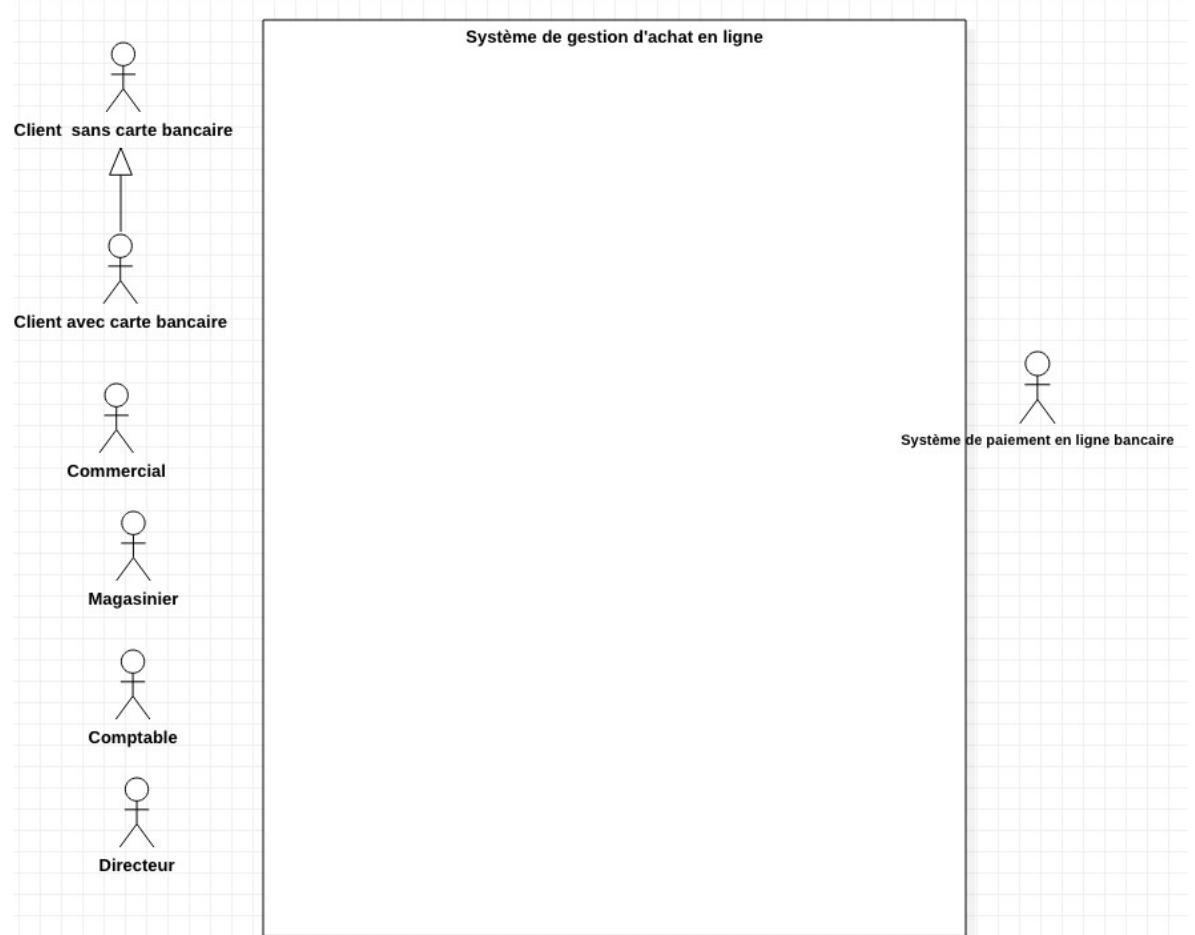
Un magasin XYZ a besoin de mettre en place un système de gestion des ventes en ligne.

Ce système doit permettre à ces utilisateurs les fonctionnalités suivantes:

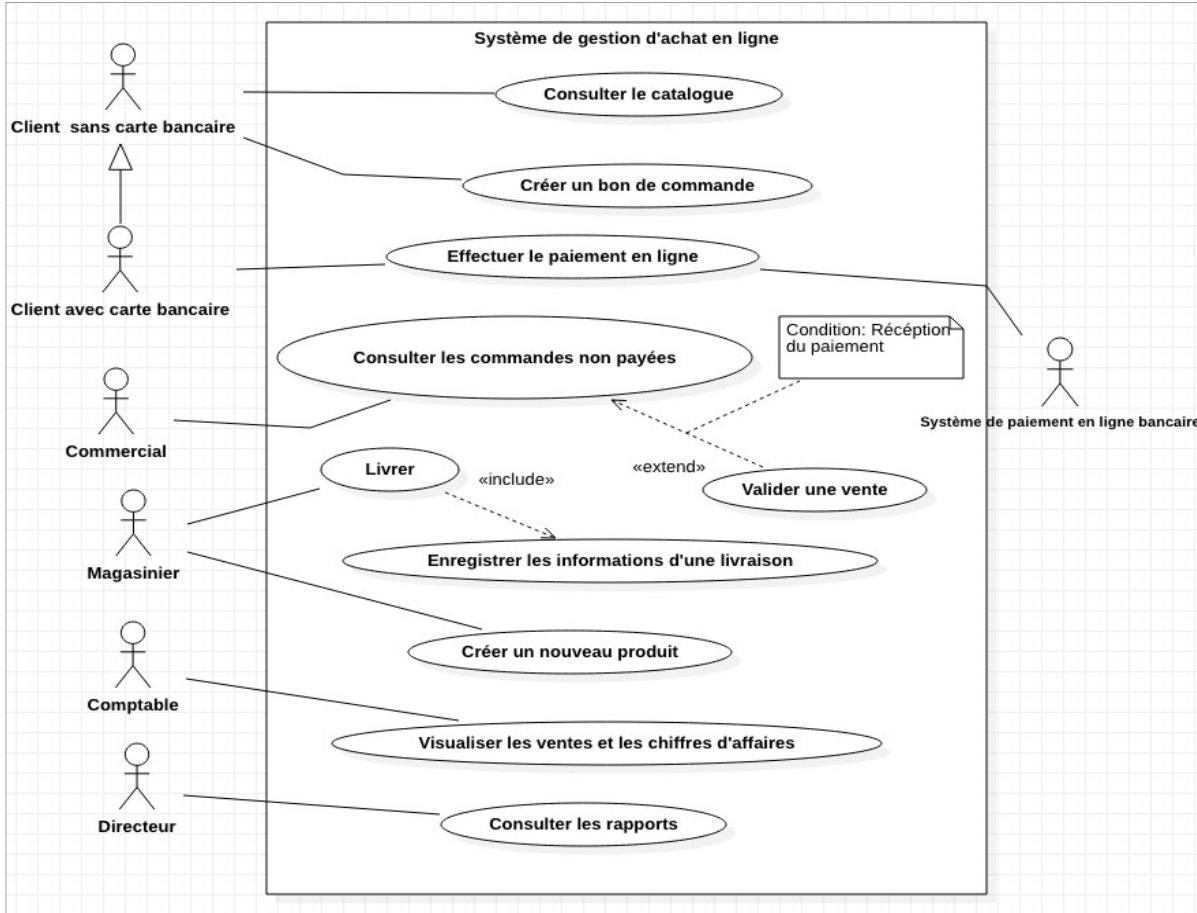
- Le client doit être capable de consulter le catalogue et créer un bon de commande.
- Si le client possède une carte bancaire, le système doit aussi lui offrir la possibilité d'effectuer le paiement en ligne à l'aide d'un système de paiement en ligne.
- Pour une commande non payée, le commercial doit prendre contact avec le client pour établir les procédures de paiement. Une fois le paiement reçu, le commercial valide la vente.
- Pour une commande payée et non livrée, le magasinier livre le client et enregistre les informations concernant la livraison sur le système.
- Le magasinier peut aussi, créer un nouveau produit s'il n'existe pas dans le catalogue
- Le comptable doit être capable de visualiser les ventes et les chiffres d'affaires réalisés
- Le directeur peut consulter des rapports fournis par le système pour les analyser

## Solution: Identification des acteurs

- Cette représentation graphique est appelée aussi diagramme de contexte
- L'objectif d'un diagramme de contexte est de préciser les acteurs d'un système sans connaître les détails de ce dernier.



### 3°) Diagramme de cas d'utilisation



# Documenter un cas d'utilisation

Le développement d'un nouveau système, ou l'amélioration d'un système existant, doit répondre à un ou à plusieurs besoins. Par exemple, une banque a besoin d'un guichet automatique pour que ses clients puissent retirer de l'argent même en dehors des heures d'ouverture de la banque. Celui qui commande le logiciel est le maître d'ouvrage. Celui qui réalise le logiciel est le maître d'œuvre. Le maître d'ouvrage intervient constamment au cours du projet, notamment pour :

- Définir et exprimer les besoins ;
- Valider les solutions proposées par le maître d'œuvre ;
- Valider le produit livré.

# Documenter un cas d'utilisation

La description d'un cas d'utilisation permet de :

- Clarifier le déroulement de la fonctionnalité
- Décrire la chronologie des actions qui devront être réalisées
- D'identifier les parties redondantes pour en déduire des cas d'utilisation plus précises qui seront utilisées par inclusion, extension ou généralisation/spécialisation.

# Documenter un cas d'utilisation

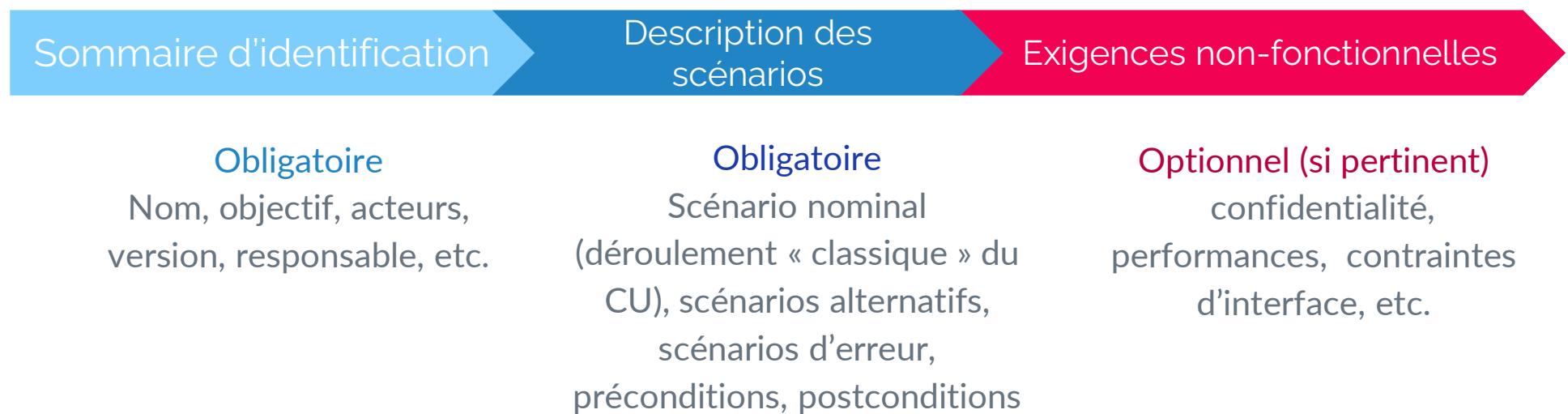
## □ Scénario d'un cas d'utilisation :

- Un scénario est une instance de cas d'utilisation.
- Il représente une séquence d'interactions entre le système et ses acteurs.
- Il décrit une exécution particulière d'un cas d'utilisation du début à la fin.
- Chaque unité de description de séquences d'actions est appelée enchaînement.
- Un scénario représente une succession particulière d'enchaînements, qui s'exécute du début à la fin du cas d'utilisation.

# Documenter un cas d'utilisation

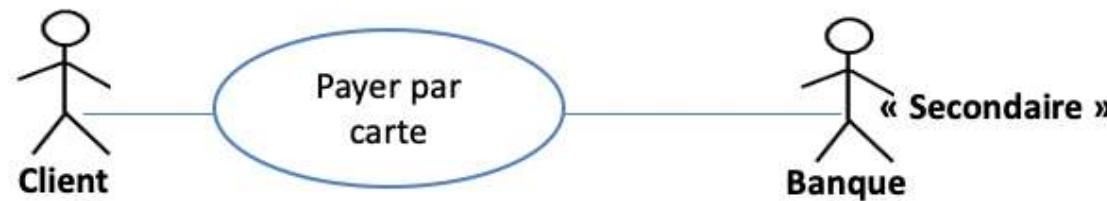
- Un cas d'utilisation contient en général :
- Un début et une fin clairement définies
- Les différentes variantes des scénarios (nominal, alternatif, d'erreur)
- Les messages échangés pendant les interactions
- L'acteur principal d'un cas d'utilisation dispose donc de l'ensemble des enchaînements pour réaliser une certaine tâche métier.

# Structure de la documentation



## Structure de la documentation:

- ❑ Exemple: Payer par carte un achat effectué en ligne



# Documenter un cas d'utilisation

## Sommaire d'identification

- Nom du cas : Payer par Carte
- Objectif : Détails les étapes permettant à un client de payer par une carte bancaire
- Acteurs : Client, Banque (secondaire)
- Date : 20/10/2020
- Responsables : Ahmed
- Version : 1.0

# Documenter un cas d'utilisation

## Description du scénario

Le cas d'utilisation commence lorsqu'un client demande le paiement par carte bancaire

### Pré-conditions :

Le client a validé sa commande

### Enchaînement nominal :

1. Le client saisit les informations de sa carte bancaire
2. Le système vérifie que le numéro de CB est correct
3. Le système vérifie le solde auprès du système bancaire
4. Le système demande au système bancaire de débiter le client
5. Le système notifie le client du bon déroulement de la transaction

### Enchaînements alternatifs :

- En (2) : si le numéro est incorrect, le client est averti de l'erreur, et invité à recommencer
- En (3) : si le solde est insuffisant, afficher message d'erreur

### Post-conditions:

- La commande est validée
- Le compte de l'entreprise est crédité

# Documenter un cas d'utilisation

## Exigences non-fonctionnelles

- Contraintes non fonctionnelles :**
  - Fiabilité : les accès doivent être sécurisés
  - Confidentialité : les informations concernant le client ne doivent pas être divulgués
- Contraintes liées à l'interface homme-machine :**
  - Toujours demander la validation des opérations bancaires

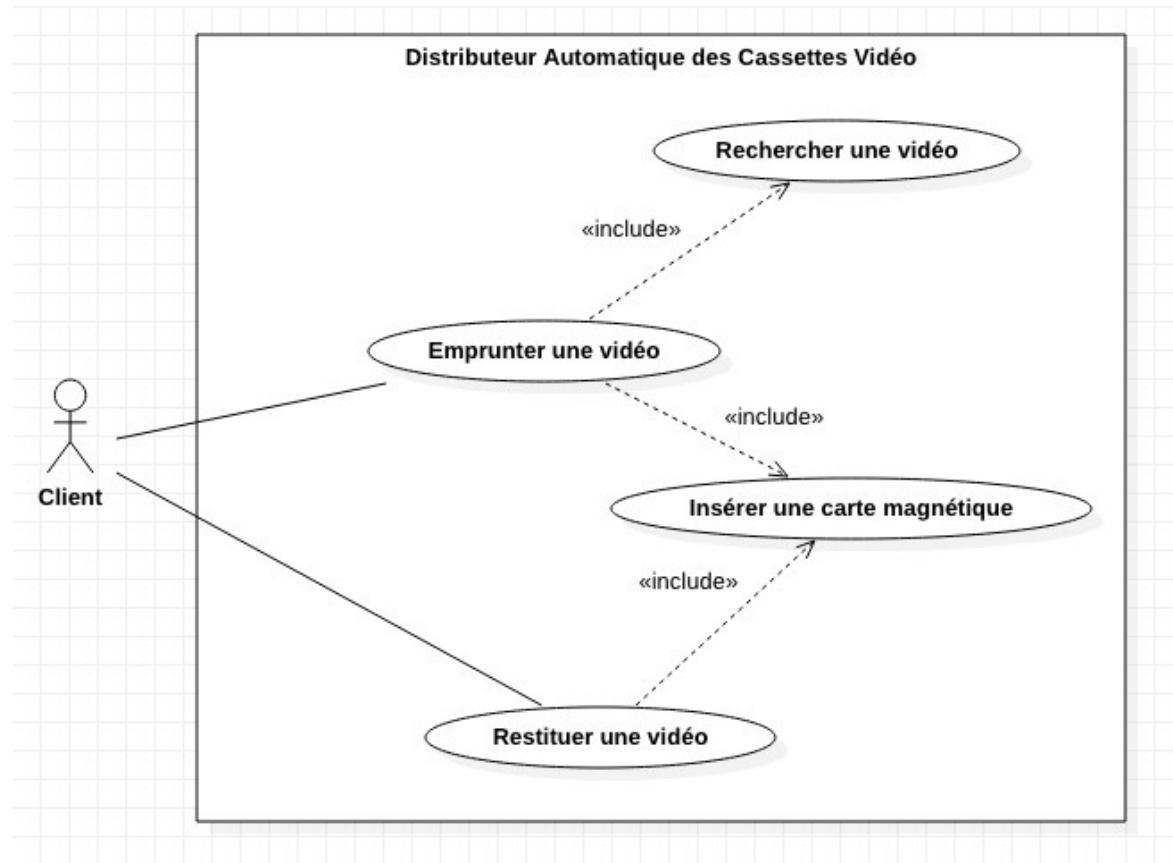
## Exemple Détailé: Distributeur automatique de cassettes vidéo

**La description du fonctionnement d'un distributeur automatique de cassettes vidéo est la suivante:**

« Une personne souhaitant utiliser le distributeur doit avoir une carte magnétique spéciale. Les cartes sont disponibles au magasin qui gère le distributeur. Elles sont créditées d'un certain montant en euros et rechargeables au magasin. Le prix de la location est fixé par tranches de 6 heures (1 euro par tranche). Le fonctionnement du distributeur est le suivant : le client introduit sa carte ; si le crédit est supérieur ou égal à 1 euro, le client est autorisé à louer une cassette (il est invité à aller recharger sa carte au magasin sinon) ; le client choisit une cassette et part avec ; quand il la ramène, il l'introduit dans le distributeur puis insère sa carte ; celle-ci est alors débitée ; si le montant du débit excède le crédit de la carte, le client est invité à régulariser sa situation au magasin et le système mémorise le fait qu'il est débiteur ; la gestion des comptes débiteurs est prise en charge par le personnel du magasin. On ne s'intéresse ici qu'à la location des cassettes, et non à la gestion du distributeur par le personnel du magasin (ce qui exclut la gestion du stock des cassettes) ».

**Q1. Modélisez avec un diagramme de cas d'utilisation le fonctionnement d'un distributeur automatique de cassettes vidéo dont la description est donnée ci-dessus.**

## Exemple détaillé: Distributeur automatique de cassettes vidéos



## Exemple détaillé: Distributeur automatique de cassettes vidéos

**Q2.** Décrivez sous forme textuelle les cas d'utilisation « **Emprunter une vidéo** » et « **Rechercher une vidéo** ». La recherche d'une vidéo peut se faire par genres ou par titres de film. Les différents genres sont : action, aventure, comédie et drame. Quand une liste de films s'affiche, le client peut trier les films par titres ou par dates de sortie en salles.

## Exemple détaillé: Distributeur automatique de cassettes vidéos

### □ La description du cas d'utilisation « Emprunter une vidéo »

#### ▪ Sommaire d'identification

- Nom du cas : « Emprunter une vidéo ».
- Objectif : décrire les étapes permettant au client du magasin d'emprunter une cassette vidéo *via* le distributeur automatique.
- Acteur principal : Client.
- Acteur secondaire : néant.
- Date de création : le 31/09/2020.
- Date de mise à jour : le 10/10/2020.
- Responsable : M. Dupont.
- Version : 1.1.

## Exemple détaillé: Distributeur automatique de cassettes vidéos

### □ La description du cas d'utilisation « Emprunter une vidéo »

#### ▪ Enchaînements alternatifs

##### A1 : Le crédit de la carte est inférieur à 1 euro.

L'enchaînement démarre après le point 2 de la séquence nominale :

3. Le système indique que le crédit de la carte ne permet pas au client d'emprunter une vidéo.
4. Le système invite le client à aller recharger sa carte au magasin.

La séquence nominale reprend au point 8.

#### ▪ Enchaînements d'exception

##### E1 : La carte introduite n'est pas valide.

L'enchaînement démarre après le point 1 de la séquence nominale :

2. Le système indique que la carte n'est pas reconnue.
3. Le distributeur éjecte la carte.

##### E2 : La cassette n'est pas prise par le client.

L'enchaînement démarre après le point 6 de la séquence nominale :

7. Au bout de 15 secondes le distributeur avale la cassette.
8. Le système annule la transaction (toutes les opérations mémorisées par le système sont défaites).
9. Le distributeur éjecte la carte.

## Exercice: Distributeur automatique de cassettes vidéos

### La description du cas d'utilisation « Emprunter une vidéo »

#### **E3 : La carte n'est pas reprise par le client.**

L'enchaînement démarre après le point 8 de la séquence nominale :

9. Au bout de 15 secondes le distributeur avale la carte.

10. Le système consigne cette erreur (date et heure de la transaction, identifiant du client, identifiant du film).

#### **E4 : Le client a annulé la recherche (il n'a pas choisi de vidéo).**

L'enchaînement démarre au point 4 de la séquence nominale :

5. Le distributeur éjecte la carte.

#### **▪ Post-conditions**

Le système a enregistré les informations suivantes :

La date et l'heure de la transaction, à la minute près : les tranches de 6 heures sont calculées à la minute près.

L'identifiant du client.

L'identifiant du film emprunté.

#### **▪ Rubriques optionnelles**

#### **Contraintes non fonctionnelles**

Le distributeur doit fonctionner 24 heures sur 24 et 7 jours sur 7.

La vérification de la validité de la carte doit permettre la détection des contrefaçons.

#### **Contrainte liée à l'interface homme-machine**

Avant de délivrer la cassette, demander confirmation au client.

## Exercice: Distributeur automatique de cassettes vidéos

### La description du cas « Rechercher une vidéo »

#### **Sommaire d'identification**

Nom du cas : « Rechercher une vidéo ».

But : décrire les étapes permettant au client de rechercher une vidéo via le distributeur automatique.

Acteur principal : néant (cas interne inclus dans le cas « Emprunter une vidéo »).

Acteur secondaire : néant.

Date de création : le 20/10/2020.

Responsable : M. Dupont.

Version : 1.0.

# Exercice: Distributeur automatique de cassettes vidéos

## □ La description du cas « Rechercher une vidéo »

### Séquencement

Le cas démarre au point 3 de la description du cas « Emprunter une vidéo ».

### Enchaînement nominal (le choix du film se fait par genres)

1. Le système demande au client quels sont ses critères de recherche pour un film (les choix possibles sont : par titres ou par genres de film).
2. Le client choisit une recherche par genres.
3. Le système recherche les différents genres de film présents dans le distributeur.
4. Le système affiche une liste des genres (les choix possibles sont action, aventure, comédie et drame).
5. Le client choisit un genre de film.
6. Le système affiche la liste de tous les films du genre choisi présentes dans le distributeur.
7. Le client sélectionne un film.

### Enchaînements alternatifs

#### A1 : Le client choisit une recherche par titres.

L'enchaînement démarre après le point 1 de la séquence nominale :

2. Le client choisit une recherche par titres.
3. Le système affiche la liste de tous les films classés par ordre alphabétique des titres.

La séquence nominale reprend au point 7.

## Exercice: Distributeur automatique de cassettes vidéos

### La description du cas « Rechercher une vidéo »

#### **Enchaînements d'exception**

##### **E1 : Le client annule la recherche.**

L'enchaînement peut démarrer aux points 2, 5 et 7 de la séquence nominale :  
Appel de l'exception E4 du cas « Emprunter une vidéo ».

#### **Post-conditions**

Le système a mémorisé le film choisi par le client.

#### **Rubriques optionnelles**

#### **Contraintes non fonctionnelles**

#### **Contraintes liées à l'interface homme-machine**

Quand une liste de films s'affiche, le client peut trier la liste par titres ou par dates de sortie en salles.

Le client peut se déplacer dans la liste et la parcourir de haut en bas et de bas en haut.

Ne pas afficher plus de 10 films à la fois dans la liste.

## Regroupement des cas d'utilisation en paquetage

“

«Un paquetage permet d'organiser des éléments de modélisation en groupe. Un paquetage peut contenir des classes, des cas d'utilisations, des interfaces, etc. »

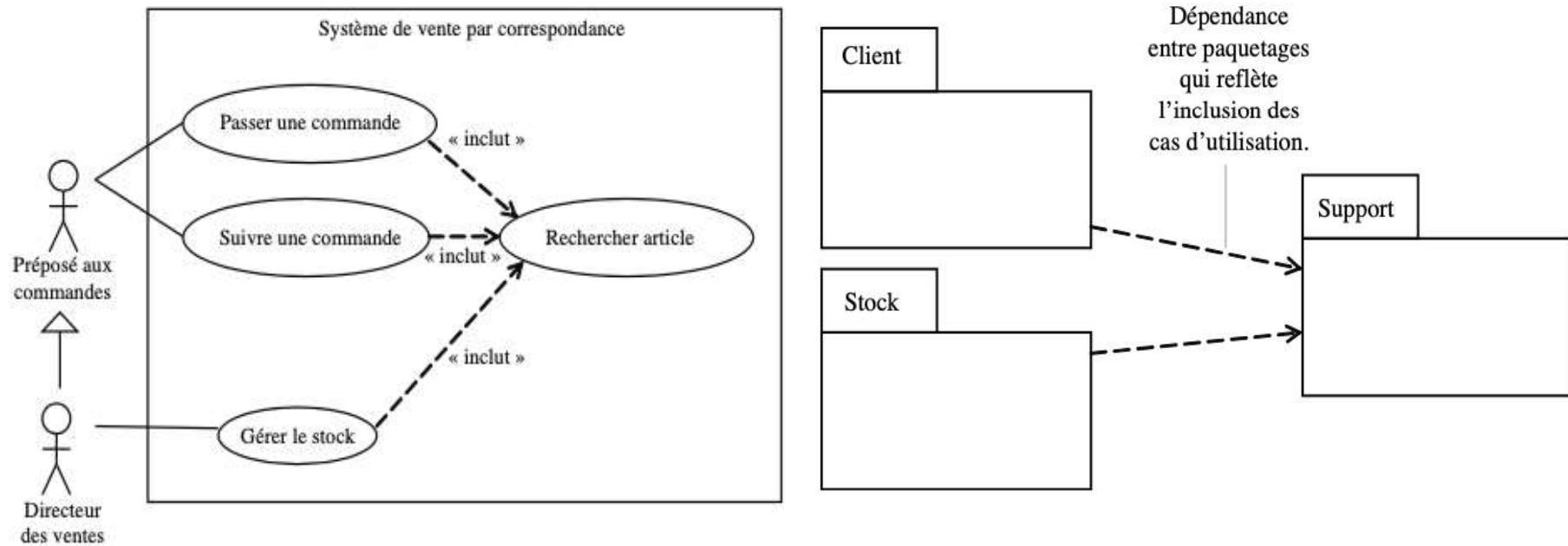
## Le regroupement des cas d'utilisation en paquetage

- UML permet de regrouper des cas d'utilisation dans une entité appelée « **paquetage** ».
- Le regroupement peut se faire par acteur ou par domaine fonctionnel.
- Un diagramme de cas d'utilisation peut contenir plusieurs paquetages et des paquetages peuvent être inclus dans d'autres paquetages.

### □ Exemple:

Le directeur des ventes d'un système de vente par correspondance est un préposé aux commandes avec un pouvoir supplémentaire (en plus de pouvoir passer et suivre une commande, il peut gérer le stock). Le préposé aux commandes ne peut pas gérer le stock.

# Le regroupement des cas d'utilisation en paquetage



- Il y a trois paquetages ont été créés : Client, Stock et Support.
- Ces paquetages contiennent les cas d'utilisation :
  - Client contient les cas « Passer une commande » et « Suivre une commande »
  - Stock contient le cas « Gérer le stock »,
  - Support contient le cas « Rechercher article ».

## Chap. 3:Diagramme de Classes

- Le diagrammes de cas d'utilisation modélisent à **QUOI** sert le système.
- Le système est composé d'objets qui interagissent entre eux et avec les acteurs pour réaliser ces cas d'utilisation :
  - Les diagrammes de classes font partie des diagrammes statiques permettant de spécifier **QUI** intervient à l'intérieur du système.
  - Ils spécifient également quels liens peuvent entretenir les objets du système.

# Diagramme de Classes: Exemple

Société XYZ,  
N°Facture: 0034/2019, Le 01/03/2019

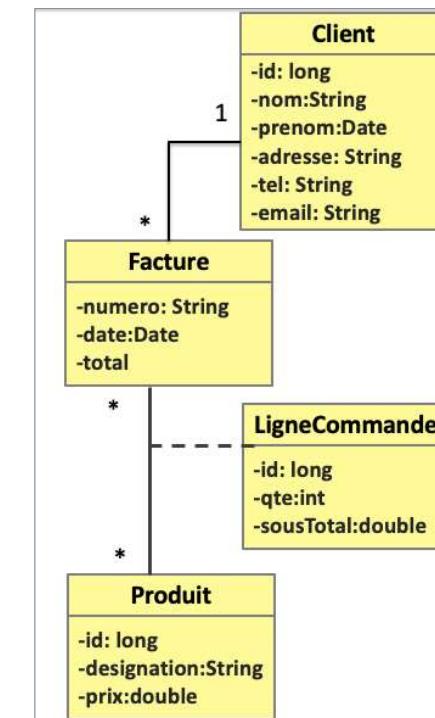
Client  
Nom : Ali Baba  
Adresse : Route 1, N°100 - Casablanca  
Tél : 0666666  
Email : uvw@m.ma

DÉSIGNATION	PU	QTE	SOUS TOTAL
GALAXY S4	1000,00	2	2000,00
GALAXY S5	1500,00	2	3000,00
PC HP 4555	4500,00	1	4500,00
IMPRIMANTE LASER 200	1300,00	1	1300,00
BUREAU PLAX-55	1200,00	1	1200,00

Total : 12000,00

Règlement Espèce

Réalité (exécution)



Modèle

# Diagramme de Classes

- La construction du diagramme de classes constitue l'objectif de tous processus de modélisation « objet ».
- Le système est composé d'objets qui interagissent entre eux et avec les acteurs pour réaliser ces cas d'utilisation.
- Les diagrammes de classes permettent de spécifier la structure et les liens entre les objets dont le système est composé.

## Qu'est ce qu'une Classe ?

“

*«Une classe est une description d'un ensemble d'objets ayant une sémantique, des attributs, des méthodes et des relations en commun. Un objet est une instance d'une classe»*

# Exemple d'une classe

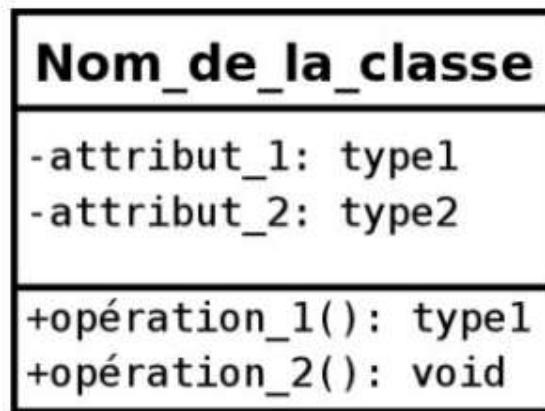
Voiture
+ marque : String
+ immatriculation : String
+ couleur : String
# puissanceFiscale : entier
# poidsVide : entier
- dateFabrication : Date
- propriétaire : Personne
+ demarrer()
+ arreter()
+ conduire( <i>a</i> : Lieu, <i>b</i> : Lieu)
- vendre(prix : entier long)



Objet ou une instance de la classe voiture

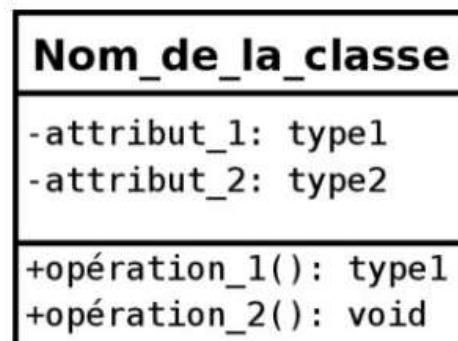
# Représentation graphique d'une Classe

- Une classe est représentée graphiquement par un rectangle divisé en trois compartiments. Le premier indique le nom de la classe, le deuxième ses attributs et le troisième ses méthodes (opérations).



# Représentation graphique d'une Classe

- Les **attributs** et les **opérations** sont les propriétés d'une classe. Leur nom commence par une minuscule.
- Un attribut décrit une donnée de la classe. Les types des attributs et leurs initialisations ainsi que les modificateurs d'accès peuvent être précisés dans le modèle.



# Représentation graphique d'une Classe

- Les attributs prennent des valeurs lorsque la classe est instanciée : ils sont en quelque sorte des « variables » attachées aux objets.
- Une **opération** est un service offert par la classe (un traitement que les objets correspondant peuvent effectuer).



# Syntaxe des attributs et des méthodes

Nous pouvons détailler la classe en indiquant :

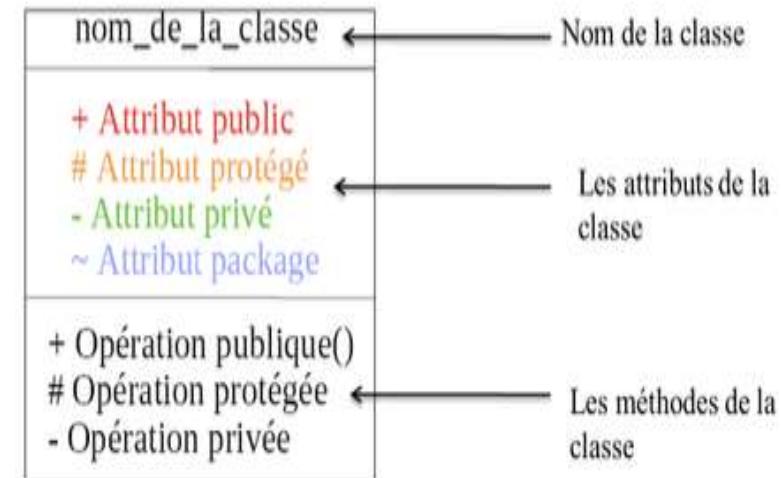
- ❑ La visibilité (encapsulation) des méthodes et des attributs.
- ❑ Le type de chaque attribut.
- ❑ La signature de chaque méthode.
- ❑ Le type de valeur renvoyée par chaque méthode.

Nom DeLaClasse
-nomAttribut1 -nomAttribut2: type -nomAttribut3: type = valeur
+nomOperation1() #nomOperation2(parametre1) -nomOperation3(parametre2: type, parametre3: type) #nomOperation4(): typeRetour -nomOperation5(parametre2: type, parametre3: type): typeRetour2

# Syntaxe des attributs et des méthodes: Visibilité

## □ La visibilité (encapsulation) des méthodes et des attributs

Type de visibilité	Symbol à placer devant l'attribut ou la méthode
public : élément non encapsulé visible par tous	+
privé : élément encapsulé visible seulement dans la classe.	-
protégé : élément encapsulé visible dans la classe et dans les sous-classes.	#
package : élément encapsulé visible dans les classes du même paquetage.	~

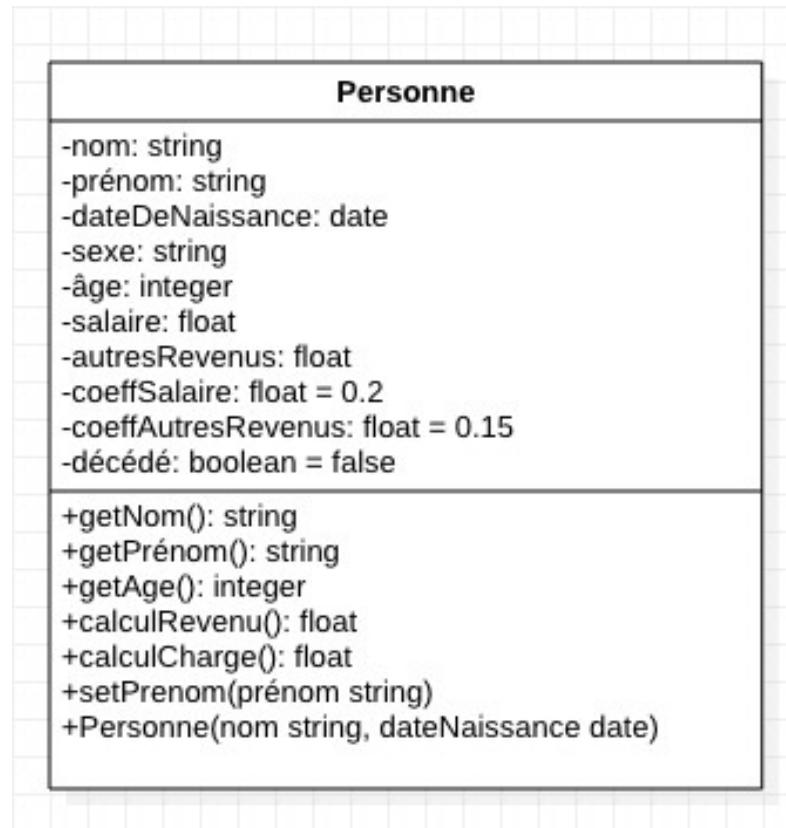


## Exercice 1:

Une personne est caractérisée par son nom, son prénom, son sexe et son âge. Les objets de classe Personne doivent pouvoir calculer leurs revenus et leurs charges. Les attributs de la classe sont privés; le nom, le prénom ainsi que l'âge de la personne doivent être accessibles par des opérations publiques.

1. Donnez une représentation UML de la classe Personne.
2. En plus des informations fournies précédemment, deux types de revenus sont envisagés: d'une part le salaire et d'autre part toutes les autres sources de revenus. les deux revenus sont représentés par des nombres réels. Pour calculer les charges globales, on applique un coefficient fixe de 20% sur les salaires et de 15% sur les autres revenus.  
Enrichissez la représentation précédente pour prendre en compte ces nouveaux éléments.
3. Un objet de la classe Personne peut être créé à partir du nom et de la date de naissance. il est possible de changer le prénom d'une personne. par ailleurs, le calcul des charges ne se fait pas de la même manière lorsque la personne décède.  
Enrichissez encore la représentation précédente pour prendre en compte ces nouveaux éléments.

# Solution



# Relations entre les classes

En COO avec UML, il existe plusieurs types de relations entre les classes dont on cite:

- Dépendance
- Association
- Héritage
- Agrégation
- Composition

## La Relation de Dépendance

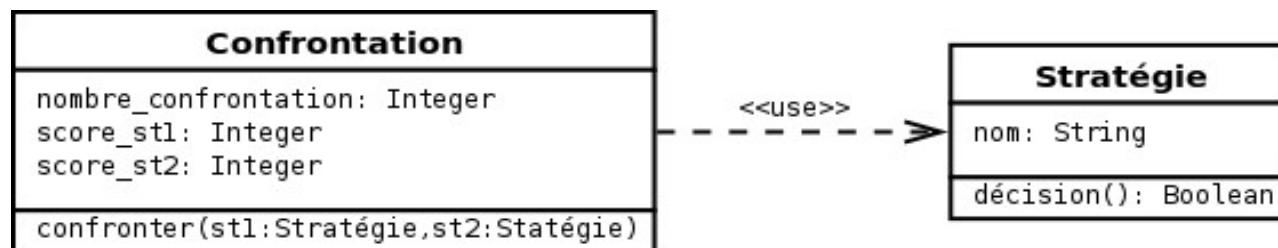
Une dépendance peut s'interpréter comme une relation de type «utilise un ». Elle est habituellement utilisée lorsqu'une classe utilise un objet d'une autre classe comme argument dans la signature d'une méthode ou alors lorsque l'objet de l'autre classe est créé à l'intérieur de la méthode. Dans les deux cas la durée de vie de l'objet est très courte, elle correspond à la durée d'exécution de la méthode.

# La Relation de Dépendance

## □ Notation:

Elle est représentée par un trait discontinu orienté, reliant les deux classes. La dépendance est souvent stéréotypée « use » pour mieux expliciter le lien sémantique entre les éléments du modèle.

## Exemple:



# La Relation d'Association

- Une association est une relation structurelle entre objets. Une association est souvent utilisée pour représenter les liens possibles entre objets de classes données.
- Elle est représentée par un trait entre classes
- Il y a deux type d'associations:
  - Binaire: est une associations qui lie deux classes
  - n-aire: est une association qui lie plusieurs classes

# La Relation d'Association

Nous pouvons détailler l'association en indiquant :

- **Le nom de l'association** : L'association peut être orné d'un texte, avec un éventuel sens de lecture, qui permet de nous informer de l'intérêt de cette relation. Nous rajoutons une phrase courte permettant de préciser le contexte de cette association.

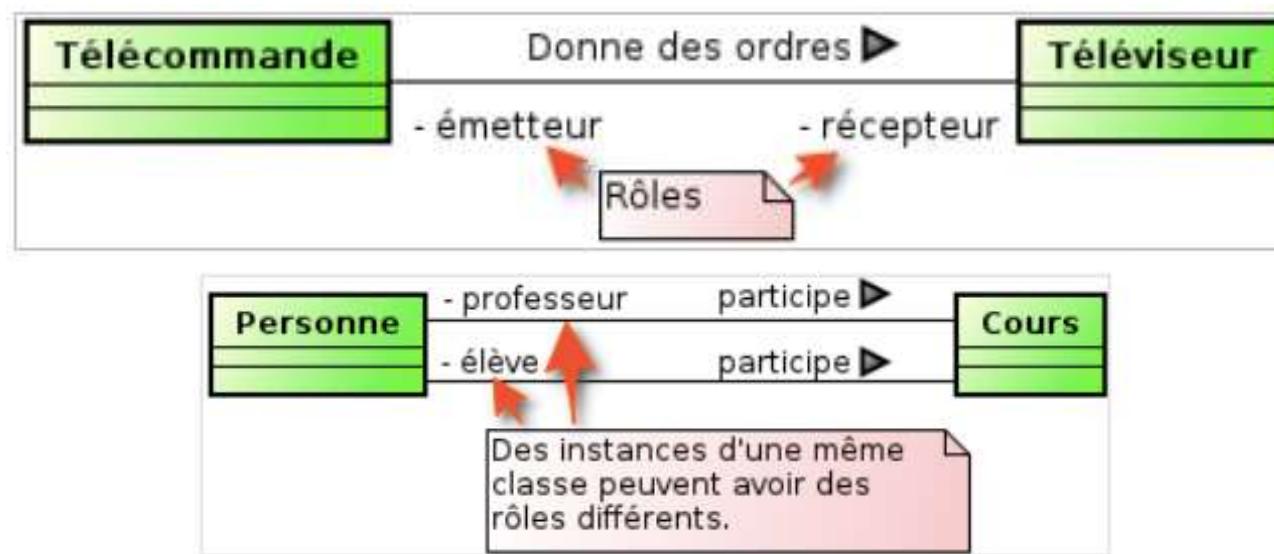


Le nom de l'association peut être suivi ou précédé des symboles « > » ou « < » (qui précise le sens de lecture).



# La Relation d'Association

- **Le rôle :** Chaque extrémité d'une association peut être nommée. Ce nom est appelé rôle et indique la manière dont l'objet est vu de l'autre côté de l'association. Lorsqu'un objet A est lié à un autre objet B par une association, cela se traduit souvent par un attribut supplémentaire dans A qui portera le nom du rôle B. (et inversement).



# La Relation d'Association

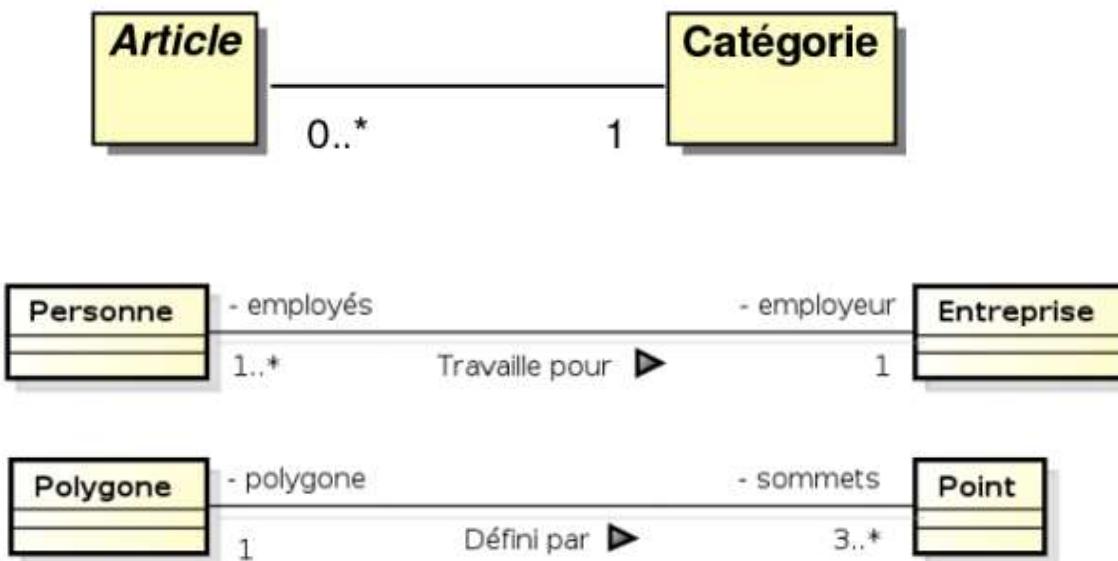
- **La cardinalité (ou multiplicité)** : La notion de multiplicité permet le contrôler du nombre d'objets intervenant dans chaque instance d'une association.

Cardinalité	Signification
<b>0..1</b>	Zéro ou une fois
<b>1..1 (ou 1)</b>	Une et une seule fois
<b>0..* (ou *)</b>	De zéro à plusieurs fois
<b>1..*</b>	De une à plusieurs fois
<b>m..n</b>	Entre <b>m</b> et <b>n</b> fois
<b>n..n (ou n)</b>	<b>n</b> fois

# La Relation d'Association

## □ Cardinalité

**Exemple:**



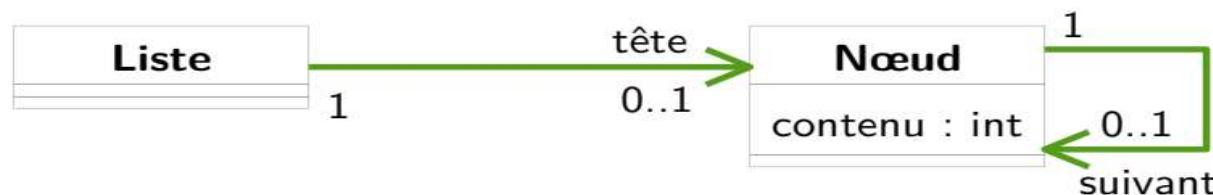
# La Relation d'Association

## Navigabilité:

- La **navigabilité** permet de spécifier dans quel(s) sens il est possible de traverser l'association à l'exécution.
- **Orientation d'une association:**
  - Restreint l'accessibilité des objets
  - Depuis un A, on a accès aux objets de B qui lui sont associés, mais pas



## Exemple (listes chaînées)

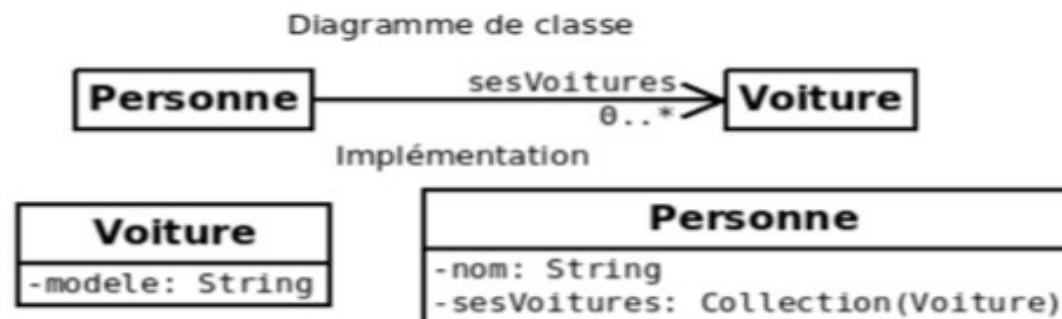


# La Relation d'Association

## Navigabilité:

Selon la navigabilité nous spécifions deux types d'associations:

1. **Association Unidirectionnelle** :On restreint la navigabilité d'une association à un seul sens à l'aide d'une flèche.
- **Exemple:** On peut accéder à ses voitures à partir d'une personne ; pas à ses propriétaires à partir d'une voiture

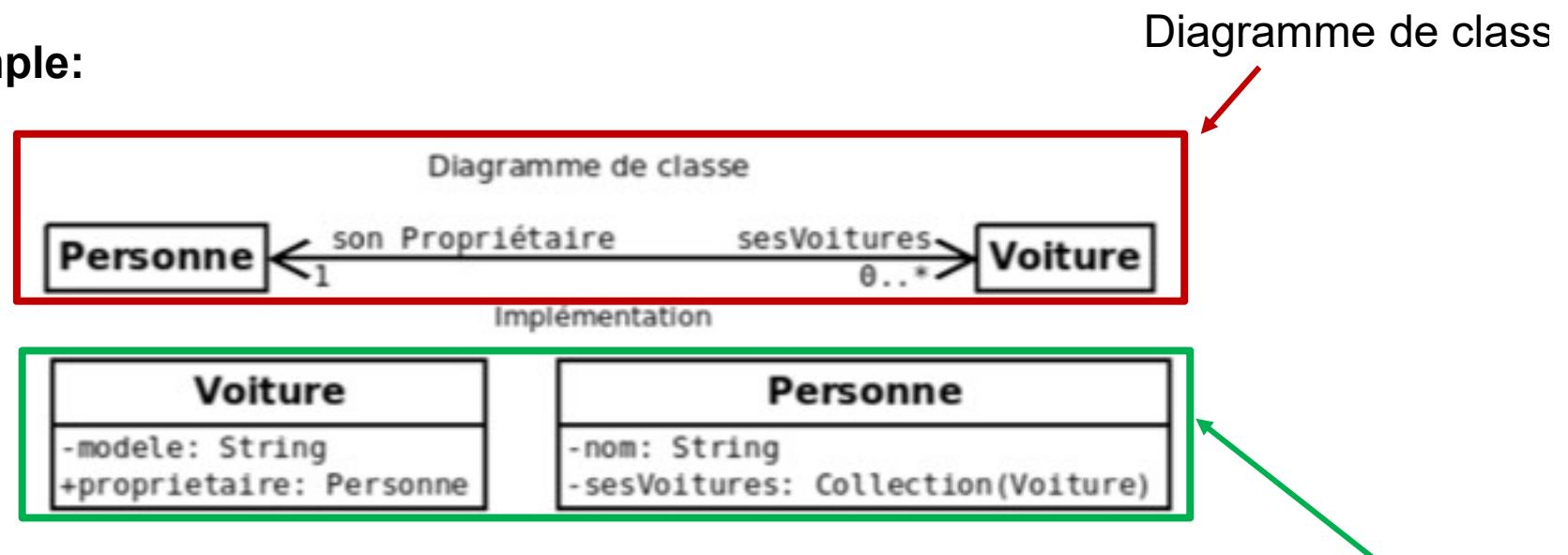


# La Relation d'Association

## **Navigabilité:**

- **2. Association bidirectionnelle** : la navigabilité est faite dans les deux sens.

- **Exemple:**



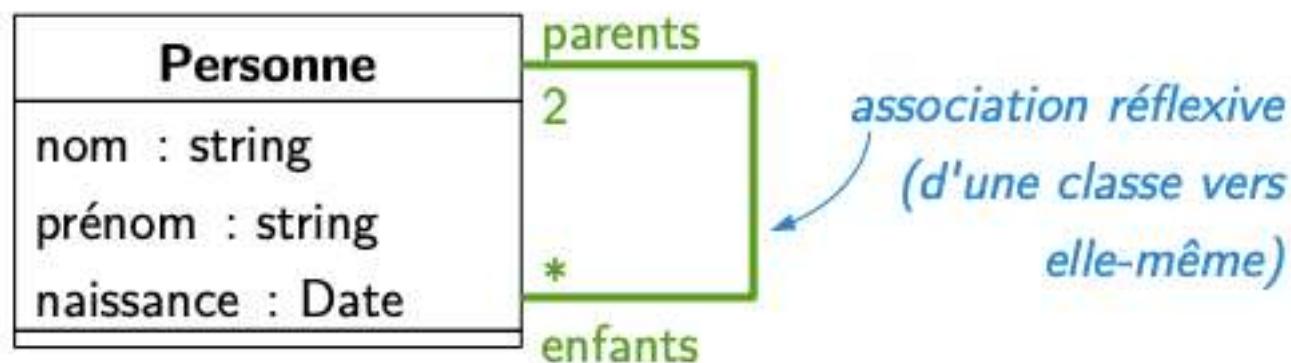
L'implémentation de diagramme de classes et l'interprétation des associations entre les classes

# La Relation d'Association

- **Association réflexive:**

Une classe peut être associée à elle-même. Une classe peut contenir une référence à des objets de même classe.

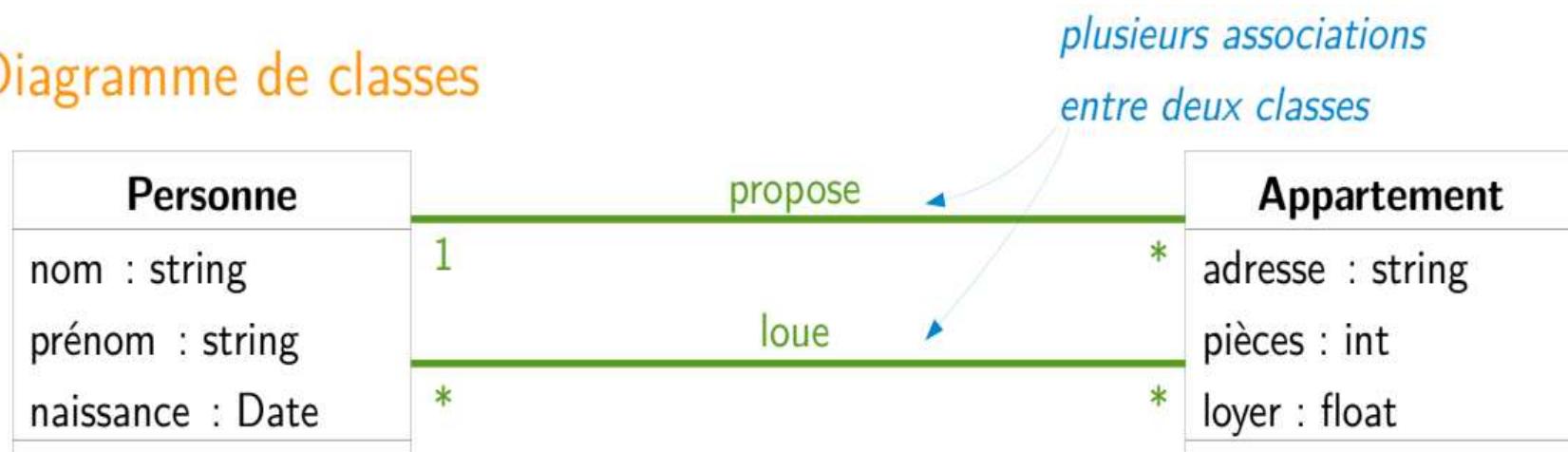
- **Exemple**



# La Relation d'Association

- **Association multiple:** les classes ont plusieurs relations distinctes entre elles.

Diagramme de classes

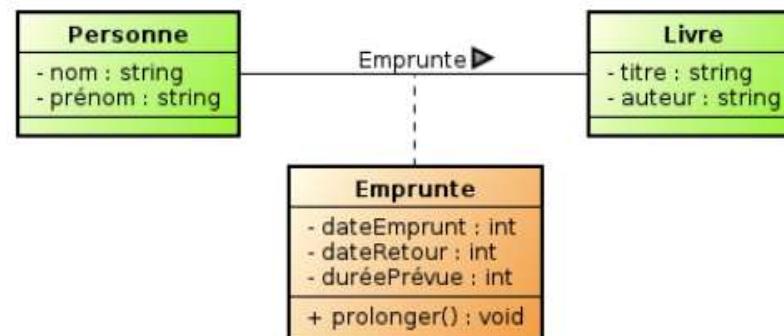


# La Relation d'Association

## Classes- Association:

Une association peut apporter de nouvelles informations (**attributs** et **méthodes**) qui n'appartiennent à aucune des deux classes qu'elle relie et qui sont spécifiques à l'association. Ces nouvelles informations peuvent être représentées par une nouvelle classe attachée à l'association via un trait en pointillés.

### ■ Exemple

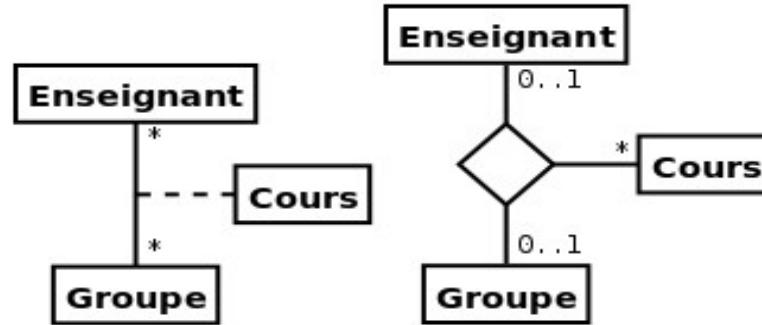


# La Relation d'Association

## Association n-aire :

Une association n-aire est une association entre 3 classes ou plus.

- **Exemple**



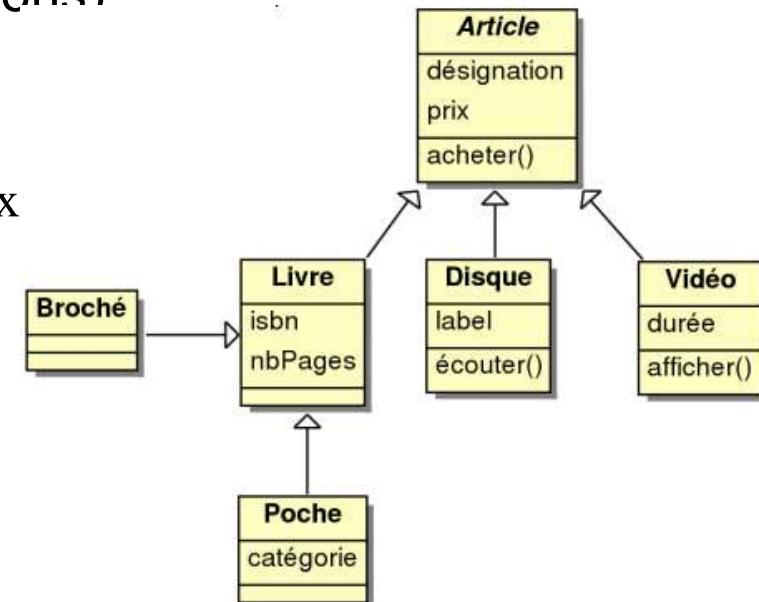
Dans le diagramme de gauche de la figure, un cours ne peut exister que s'il existe un lien entre un objet Enseignant et un objet Groupe. Quand le lien est rompu (effacé), le cours l'est également. Si un cours doit pouvoir exister indépendamment de l'existence d'un lien il faut opter pour une association ternaire (modèle de droite dans figure).

# Héritage

L'héritage est une relation de spécialisation/généralisation. Les éléments spécialisés héritent de la structure et du comportement des éléments plus généraux (attributs et opérations)

- **Exemple:**

Par héritage d'Article, un livre a d'office un prix  
une désignation et une opération acheter(), sans  
qu'il soit nécessaire de le préciser.

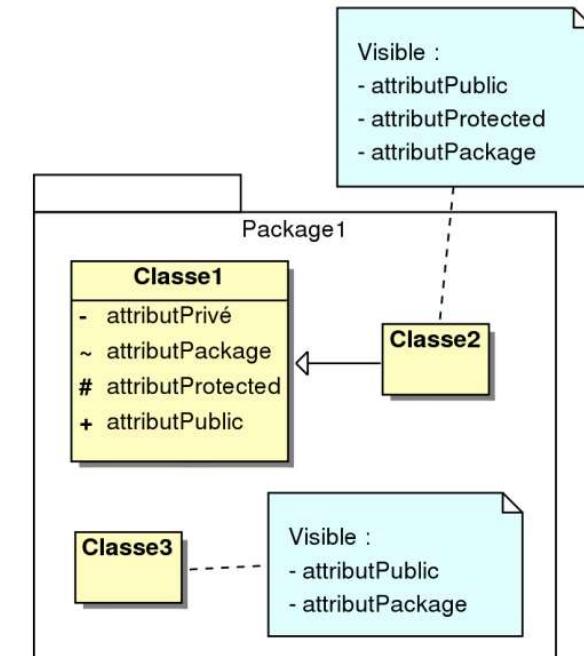


# Héritage

La classe enfant possède toutes les propriétés de ses classes parents (attributs et opérations):

- La **classe enfant** est la **classe spécialisée**
- La **classe parent** est la **classe générale**

Toutefois, elle n'a pas accès aux propriétés privées.

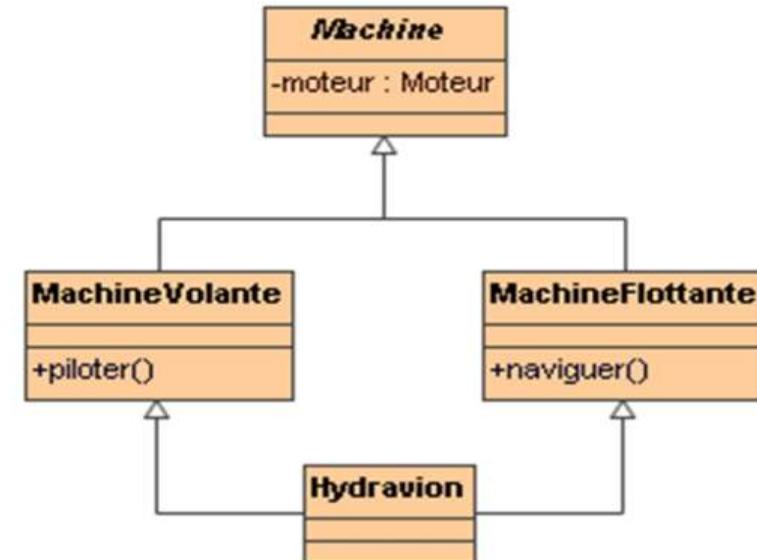
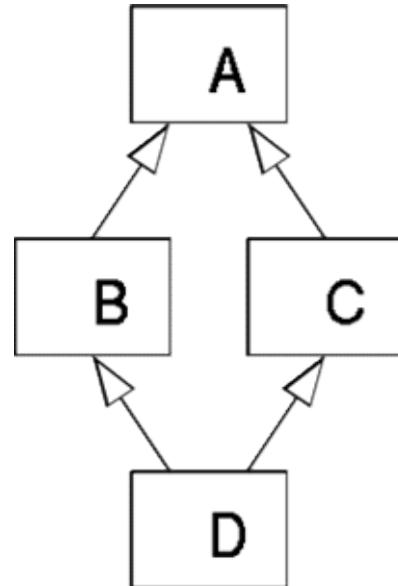


# Héritage multiple

Une classe peut avoir plusieurs classes parents. On parle alors d'héritage multiple.

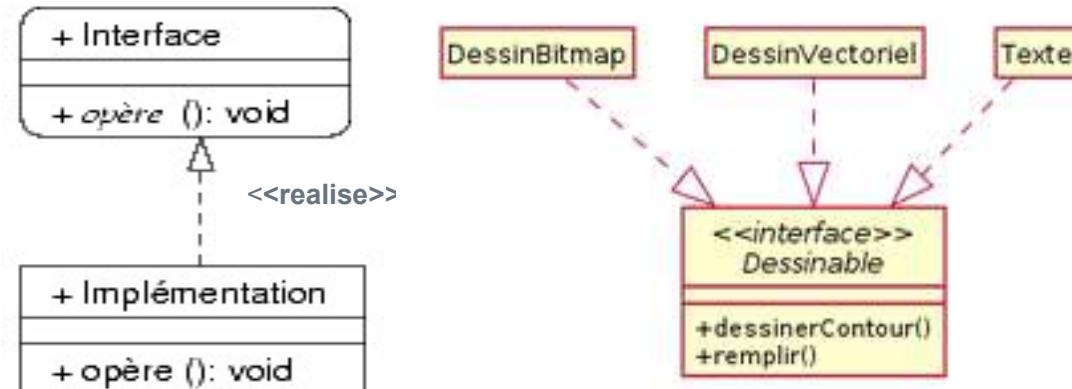
- Le langage C++ est un des langages objet permettant son implantation effective.
- Java ne le permet pas.

Problème: Combien de moteurs à l'hydravion ?



# Interface

- Le rôle d'une **interface** est de regrouper un ensemble d'opérations assurant un service cohérent.
- Une interface est définie comme une classe, mais sans des attributs.  
On ajoute le stéréotype **<< interface >>** avant le nom de l'interface.
- On utilise une association de type **réalisation** en ajoutant le stéréotype **<<realise>>** entre l'interface et la classe qui l'implémente.
- Les classes implémentant une interface doivent implémenter toutes les opérations décrites dans l'interface.

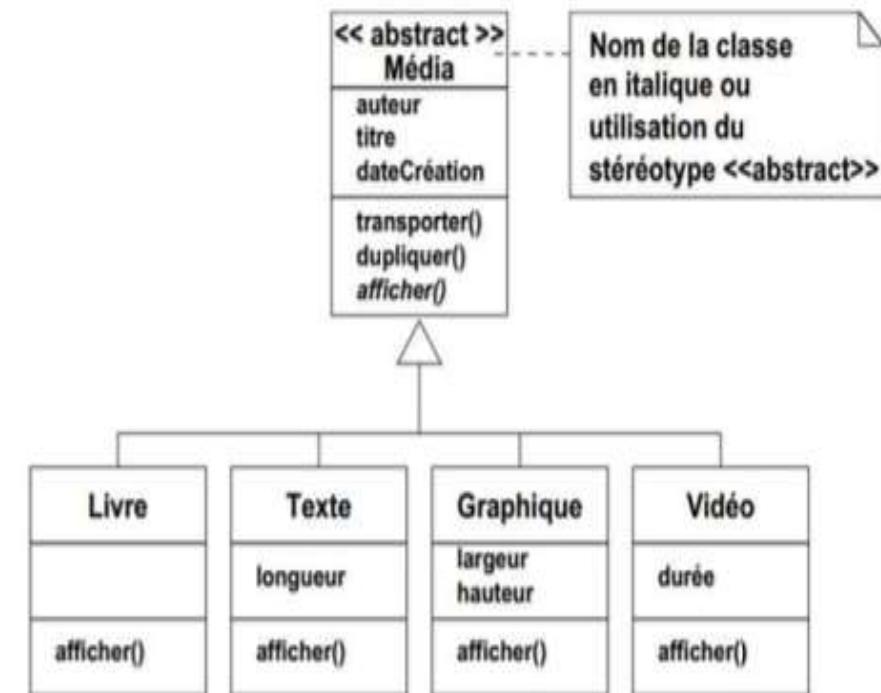


# Classes Concrètes et Classes Abstraites

- Une classe **concrète** possède des instances. Elle constitue un modèle complet d'objet: tous les attributs et méthodes sont complètement décrits.
- Une classe **abstraite** ne peut pas posséder d'instance directe car elle ne fournit pas une description complète.
- Elle a pour vocation de posséder des sous classes concrètes et sert à factoriser des attributs et méthodes communs à ses sous classes.
- En UML, une classe ou une méthode abstraite sont représentées avec une mise en ***italique*** du nom de la classe ou de la méthode ou par le stéréotype **<<Abstract>>**.

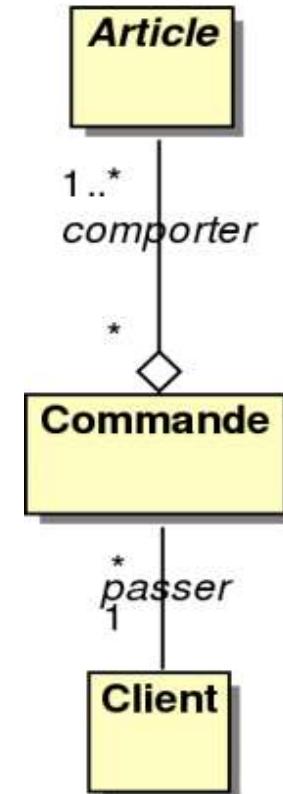
# Classes Concrètes et Classes Abstraites

**Exemple** Un Media peut être transporté, dupliqué, affiché. Le transport et la duplication sont indépendants du type de Média (copie de fichiers). Par contre, tout Média peut être affiché et ce n'est pas la même chose pour Livre, Vidéo, Graphique ou Texte. Un Média ne peut pas définir comment il s'affiche tant qu'il ne sait pas ce qu'il est. Il n'existe pas d'instance de la classe Média. Un Média n'existe qu'en tant que Livre, Texte, Graphique ou Vidéo.



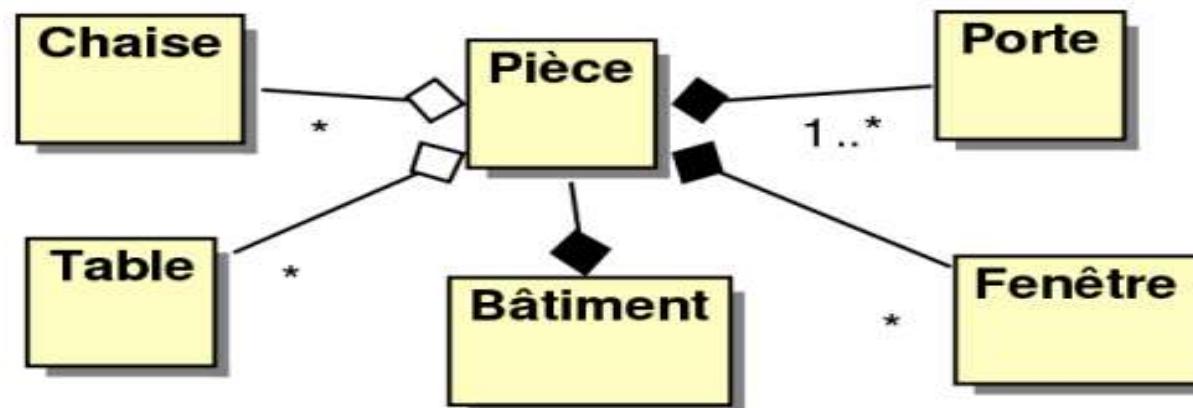
# Agrégation

- Une agrégation est une forme particulière d'association.  
Elle représente la relation d'inclusion d'un élément dans un ensemble.
- On représente l'agrégation par l'ajout d'un losange vide du côté de l'agrégat.
- Une agrégation dénote une relation d'un ensemble à ses parties. L'ensemble est l'agrégat et la partie l'agrégé.



# Composition

- La relation de composition décrit une contenance structurelle entre instances. On utilise un losange plein.
- La destruction et la copie de l'objet composite (l'ensemble) impliquent respectivement la destruction ou la copie de ses composants (les parties).
- Une instance de la partie n'appartient jamais à plus d'une instance de l'élément composite.



# Agrégation ou Composition ?

Pour décider de mettre une composition plutôt qu'une agrégation, on doit se poser les questions suivantes :

1. Est-ce que la destruction de l'objet **composite** (du tout) implique nécessairement la destruction des objets **composants** (les parties) ?

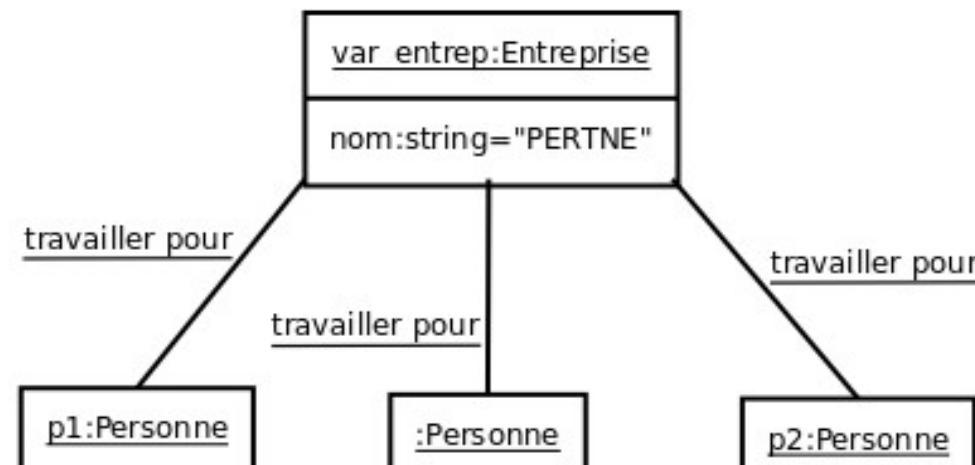
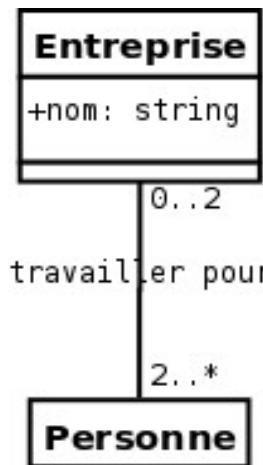
→ C'est le cas si les composants n'ont pas **d'autonomie** vis-à-vis des composites.

1. Lorsque l'on copie le composite, doit-on aussi copier les composants, ou est-ce qu'on peut les réutiliser , auquel cas un composant peut faire partie de plusieurs composites ?

=> **Si on répond par l'affirmative à ces deux questions, on doit utiliser une composition**

# Diagramme d'objet

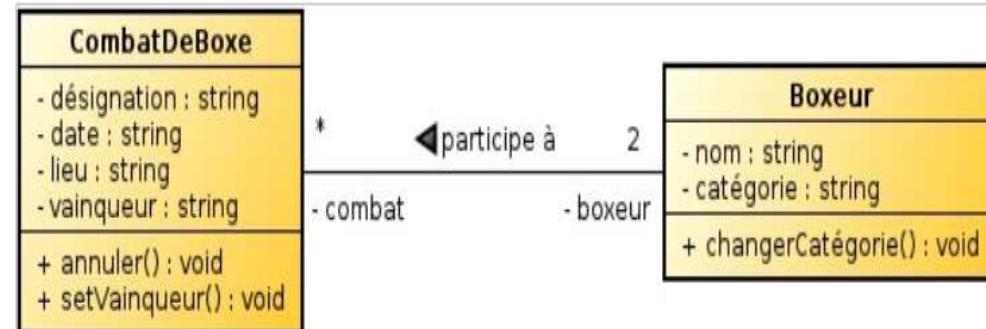
- Un diagramme d'objets représente des objets et leurs liens pour donner une vue de l'état du système à un instant donné de l'exécution.
- A un diagramme de classe correspond une infinité de diagrammes d'objets.



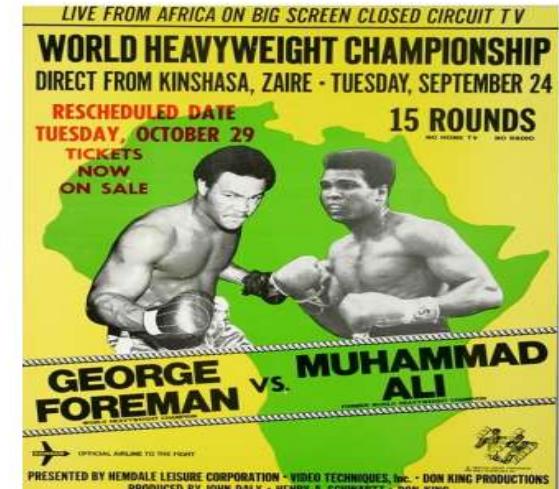
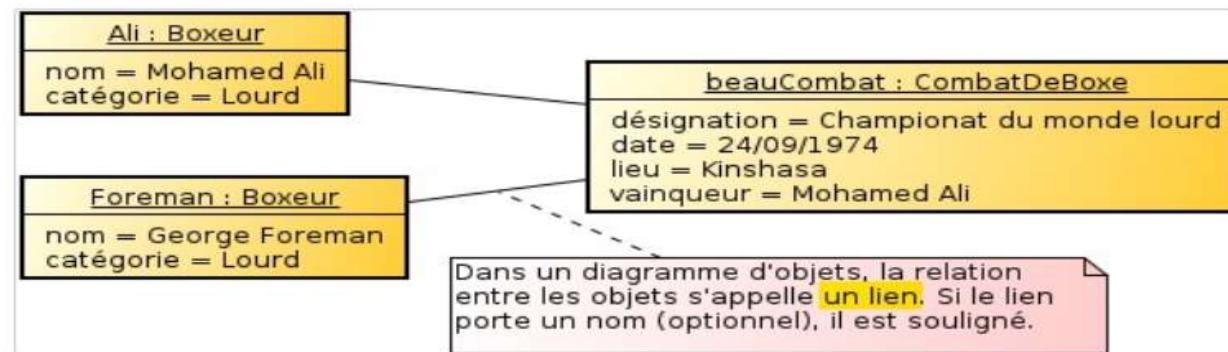
# Diagramme d'objet

## ■ Exemple:

Soit le diagramme de classe suivant :



Avec le diagramme objet ci-dessous on définit une instance particulière du diagramme de classe (qui est le combat entre **Mohamed Ali** et **George Foreman** qui a eu lieu le 24 septembre 1974 à Kinshasa).



## **Exercice 1:**

Dessiner les diagrammes de classe correspondant aux situations suivantes :

1. Tout écrivain a écrit au moins une œuvre ;
2. Les personnes peuvent être associées à des universités en tant qu'étudiants aussi bien qu'en tant que professeurs ;
3. Les cinémas sont composés de plusieurs salles qui projettent des films à une heure déterminée ;
4. Tous les jours, le facteur distribue des recommandés dans une zone géographique qui lui est affectée. Les habitants sont aussi associés à une zone géographique. Les recommandés sont de deux sortes : lettres ou colis. Comme plusieurs facteurs peuvent intervenir sur la même zone, on souhaite, pour chaque recommandé, le facteur qui l'a distribué, en plus du destinataire.

## Solution 1

110

## Solution 1

4. Tous les jours, le facteur distribue des recommandés dans une zone géographique qui lui est affectée. Les habitants sont aussi associés à une zone géographique. Les recommandés sont de deux sortes : lettres ou colis. Comme plusieurs facteurs peuvent intervenir sur la même zone, on souhaite, pour chaque recommandé, le facteur qui l'a distribué, en plus du destinataire.

---

---

---

---

---

---

---

---

---

## **Exercice 2**

Une start-up de développeurs travaille pour une entreprise dans un cadre de sous-traitance. Cette start-up possède un identifiant et un nom. Chaque développeur est caractérisé par son numéro de carte d'identité, son nom, son prénom, et son adresse email. Il utilise un ordinateur portable qui lui est personnel. Un développeur peut être un programmeur spécialisé dans un langage de programmation, ou un concepteur expert dans une méthode ou un langage de modélisation. les concepteurs créent les modèles de conception et les programmeurs écrivent le code.

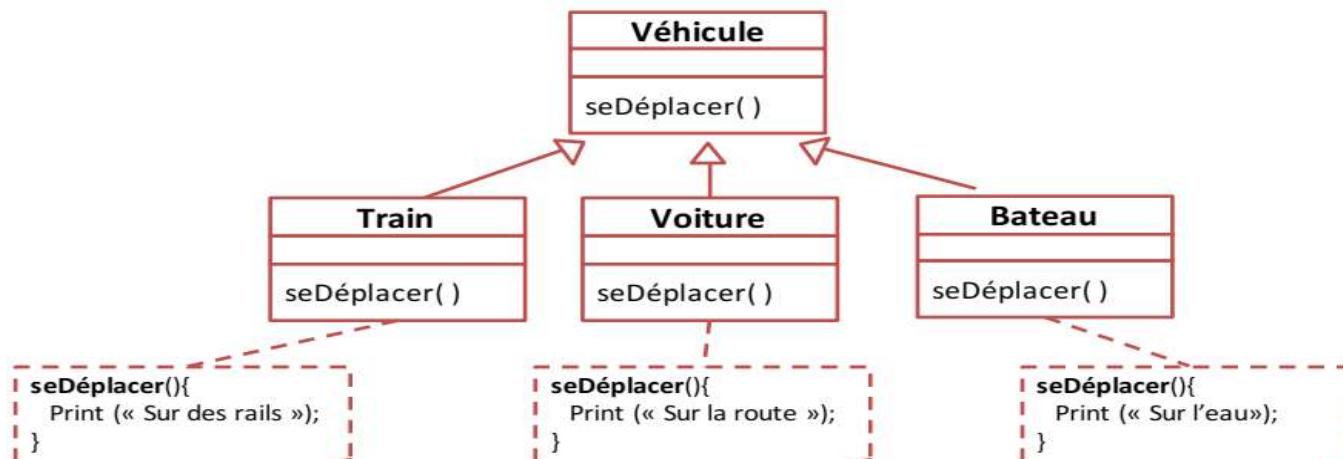
1. Créer le diagramme de classe correspondant à la description faite en dessus.
2. Créer le diagramme d'objets correspondant au texte suivant : Ali et Mostapha sont des programmeurs spécialisés respectivement dans les langages Ada et Java. Mohamed est un concepteur spécialisé dans le langage de modélisation UML . Ils font tous partie du start up « DevSoft » qui travaille pour le compte de l'entreprise «AtlasTechno »

## Solution 2

# Polymorphisme et Héritage

## Définition :

- Poly : plusieurs
  - Morphisme : Forme
- Faculté d'une méthode à pouvoir s'appliquer à des objets de classes différentes.
- Capacité d'une classe fille à **redéfinir** une méthode héritée à partir d'une classe mère.



L'opération **se déplacer** est polymorphe car sa réalisation peut prendre plusieurs formes

## Le modèle relationnel et le modèle objet

Les problèmes de correspondance entre le modèle objet et le modèle relationnel:

**Le modèle objet propose plus de fonctionnalités :**

- l'héritage, le polymorphisme
- Les relations entre deux entités sont différentes

**Les objets ne possèdent pas d'identifiant unique contrairement au modèle relationnel**

# Le Mapping Objet Relationnel

## Définition :

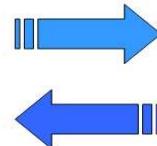
- Concept permettant de connecter un modèle objet à un modèle relationnel.
- Couche qui va interagir entre l'application et la base de données.

## Pourquoi utiliser ce concept?

- Pas besoin de connaître l'ensemble des tables et des champs de la base de données
- Faire abstraction de toute la partie SQL d'une application

table

Id	name
..	..
..	..
..	..



```
public class Obj {  
    private int id;  
    private String name;  
  
    public Obj(){  
    }  
    public Obj(String name) {  
        this.name = name;  
    }  
    public void setId(int i) {  
        id = i;  
    }  
    ...  
}
```

## Le Mapping Objet Relationnel

Les différents outils existants :

- Python : SQLAlchemy
- Java : TopLink, Hibernate
- C# : NHibernate, DLinq

## Chapitre 4: Diagramme de séquences

- Les **diagrammes de cas d'utilisation** modélisent à **QUOI** sert le système, en organisant les interactions possibles avec les acteurs.
- Les **diagrammes de classes** permettent de spécifier la structure et les liens entre les objets dont le système est composé : ils spécifient **QUI** sera à l'œuvre dans le système pour réaliser les fonctionnalités décrites par les diagrammes de cas d'utilisation.
- Par contre, les **diagrammes de séquences** permettent de décrire **COMMENT** les éléments du système interagissent entre eux et avec les acteurs.
- Les objets au cœur d'un système interagissent en échangeant des messages.
- Les acteurs interagissent avec le système au moyen d'IHM (Interfaces Homme-Machine).

## Diagramme de Séquence

Un diagramme de séquence fait parties des diagrammes comportementaux (dynamique) et plus précisément des diagrammes d'interactions.

- ✓ Un diagramme de séquence va permettre de représenter graphiquement un scénario.
- ✓ Il permet de représenter des échanges entre les différents objets et acteurs du système en fonction du temps.
- ✓ Vue la difficulté de modéliser la dynamique globale du système dans un seul diagramme. Nous ferons donc appel à un ensemble de diagrammes de séquences chacun correspondant généralement à un cas d'utilisation.

.

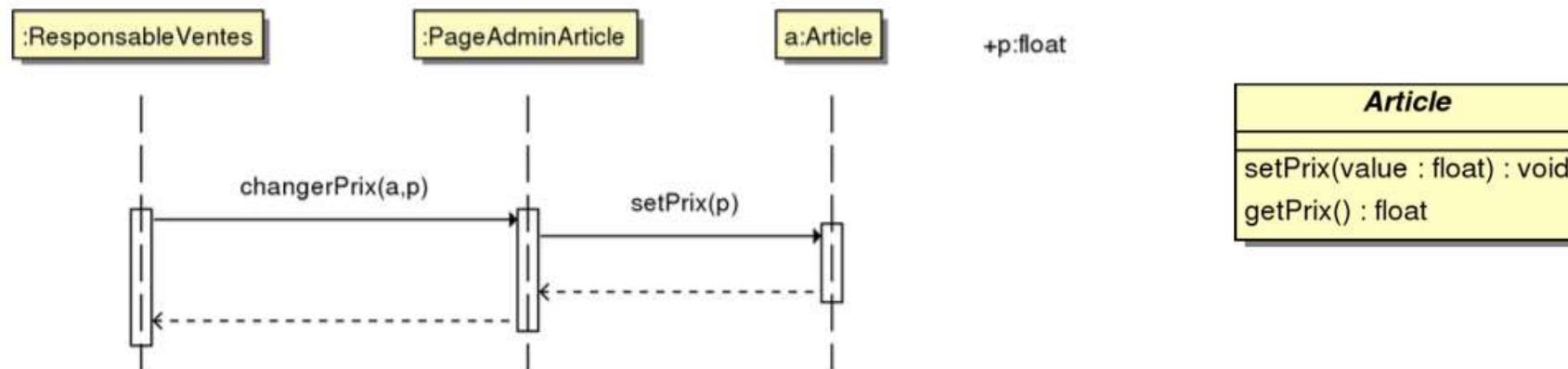
120

# Exemple d'interaction

- Soit le cas d'utilisation « ChangerPrixArticle » qui est représenté dans la figure ci-dessous:

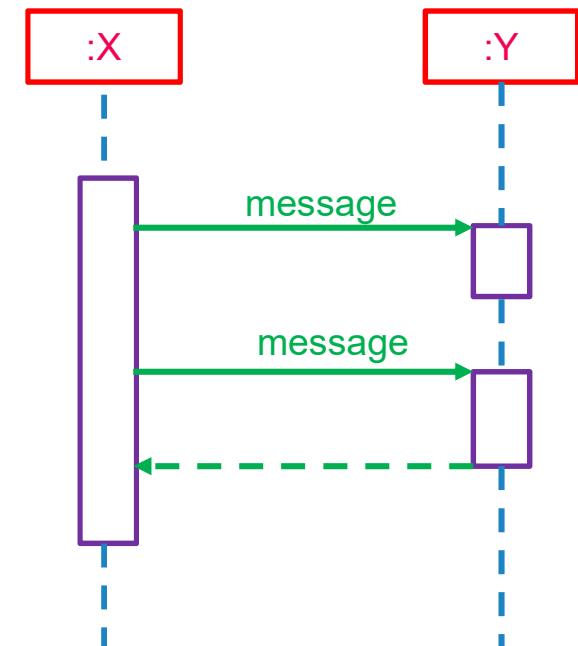


- Son diagramme de séquences correspondant est le suivant:



# Concepts Principaux

- Les objets / acteurs
- La ligne de vie de l'objet/acteur
- Les messages
- Le point de contrôle (barre d'activation)



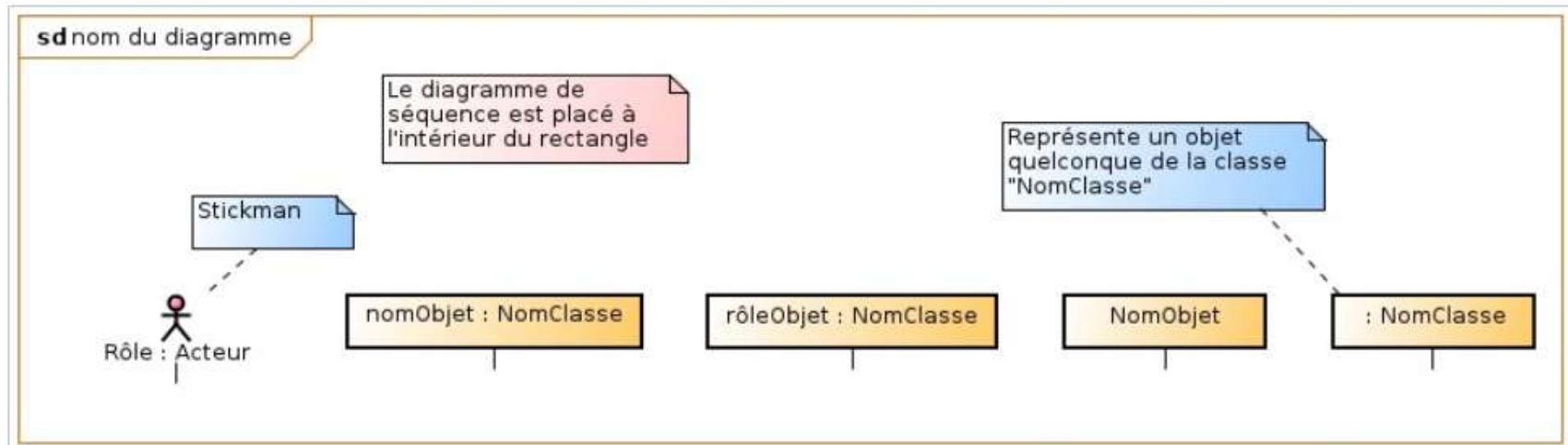
## Délimitation du diagramme de séquence

Le diagramme de séquence est placé dans un rectangle qui dispose d'une étiquette **sd** en haut à gauche (qui signifie **sequence diagramm**) suivi du nom du diagramme.



# Objet /Acteur

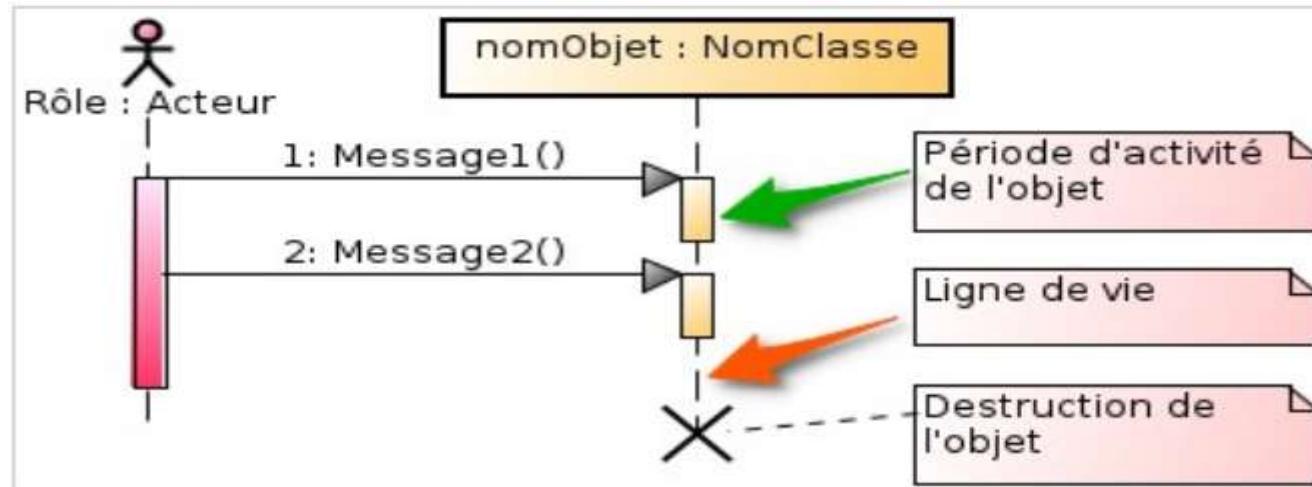
Dans un diagramme de séquence, l'objet est présenté par un rectangle dans lequel figure le nom de l'objet. Le nom de l'objet est généralement souligné et peut prendre l'une des quatre formes suivantes :



Les diagrammes de séquences représentant les échanges entre les objets mais aussi les échanges avec les acteurs, nous trouverons aussi la représentation du **stickman** (qui peut être considéré comme un objet).

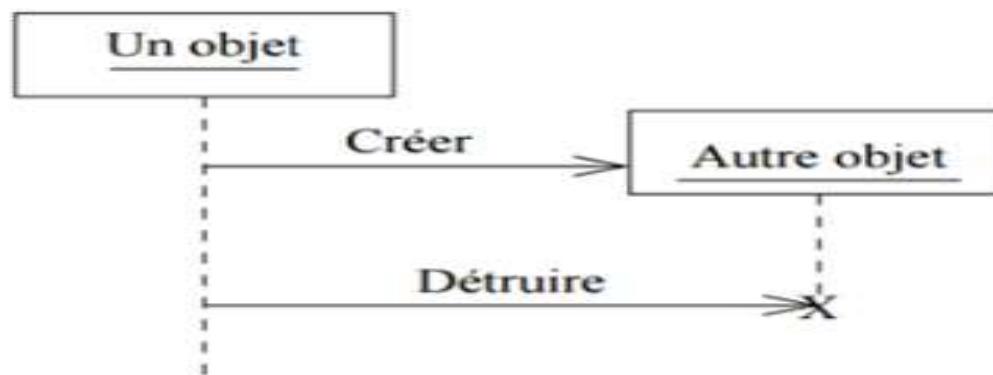
# Ligne de vie des objets

- A chaque objet est associé une ligne de vie (en trait pointillés à la verticale de l'objet) qui peut être considéré comme un axe temporel (le temps s'écoule du haut vers le bas).
- La ligne de vie indique **les** périodes d'activité de l'objet.
- Lorsque l'objet est détruit, la ligne de vie s'achève par un croix.



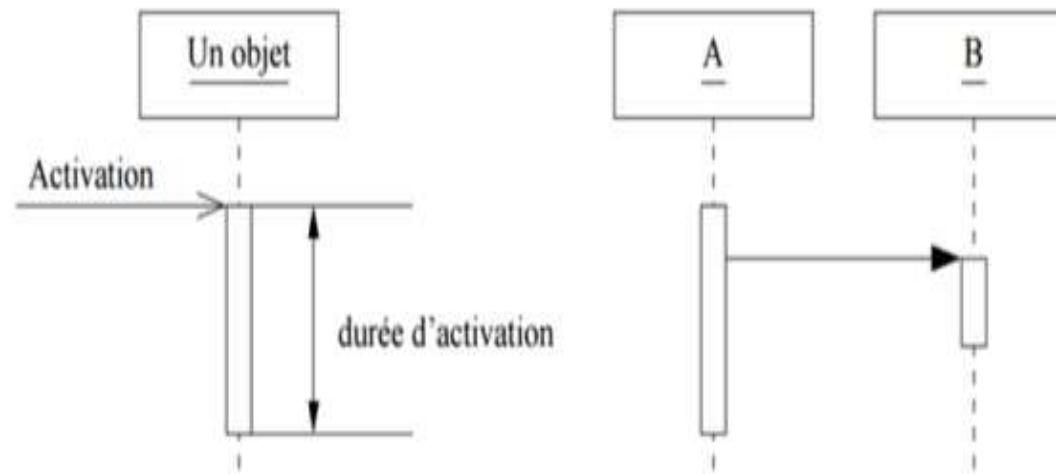
# Ligne de vie des objets

- La ligne de vie peut se terminer à l'endroit où l'objet est détruit.
- De même, la création d'un objet peut être représentée simplement par l'envoi d'un nouveau message à un objet.
- Les objets qui sont créés vers la fin de la séquence n'apparaissent pas toujours en haut du diagramme, mais peuvent apparaître à l'endroit où ils sont créés.



## Le point de contrôle (Bande / barre d'activation)

- La dimension verticale représente l'écoulement du temps. Une période d'activité correspond au temps pendant lequel un objet effectue une action.
- Graphiquement elle est représentée en plaçant un rectangle au-dessus de la ligne de vie de l'objet.

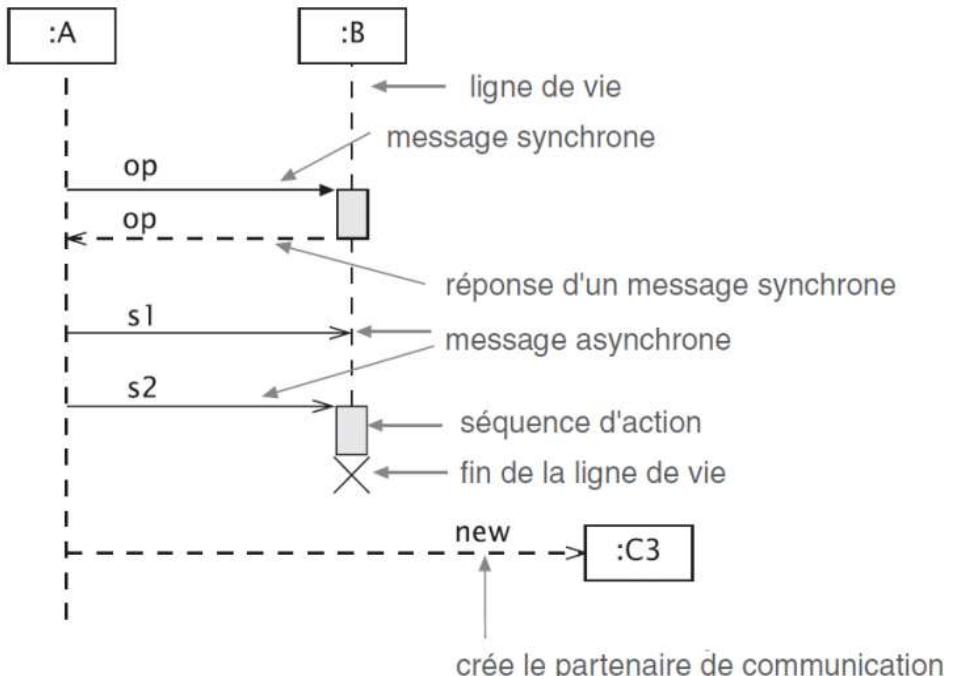


# Les Messages

Un message définit une communication particulière entre des lignes de vie (objets ou acteurs).

Ce message peut être défini par :

- l'envoi d'un signal.
- l'invocation d'une opération (appel de méthode).
- la création ou la destruction d'un objet.

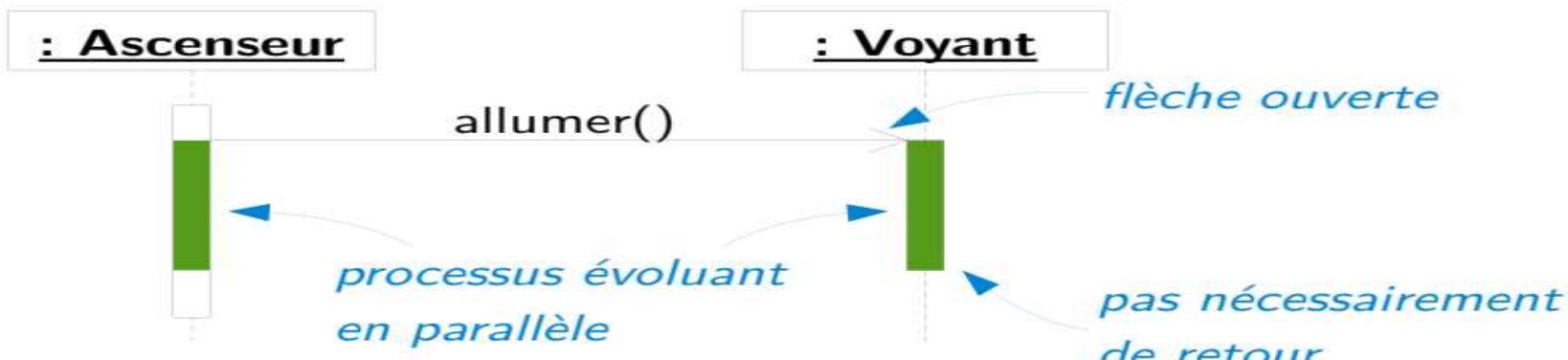


## Les Principaux Types de Message

- **Un message synchrone** Émetteur bloqué en attente du retour.



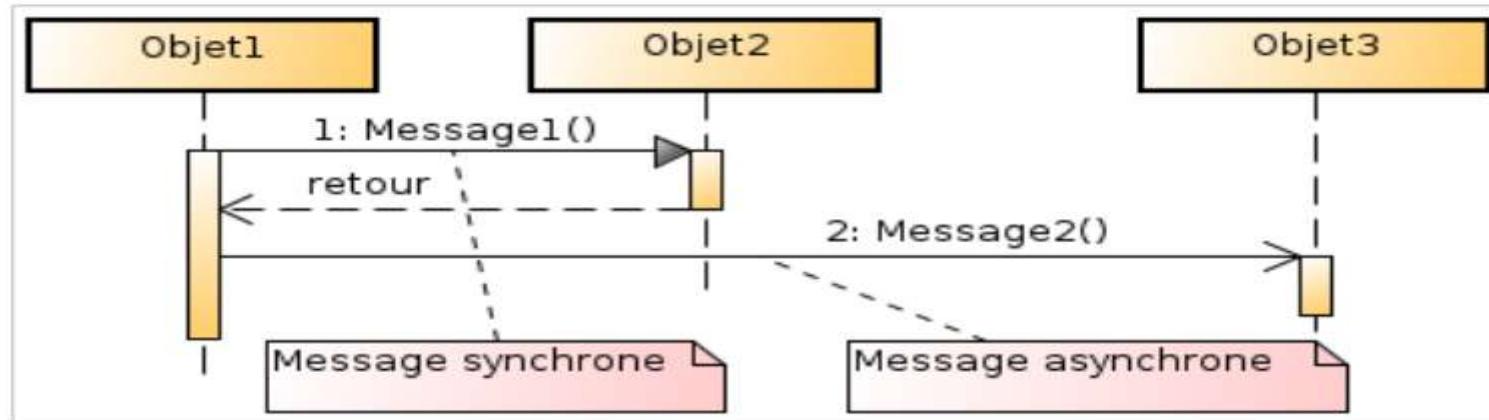
- **Un message asynchrone** Émetteur non bloqué, continue son exécution



# Présentation Graphique

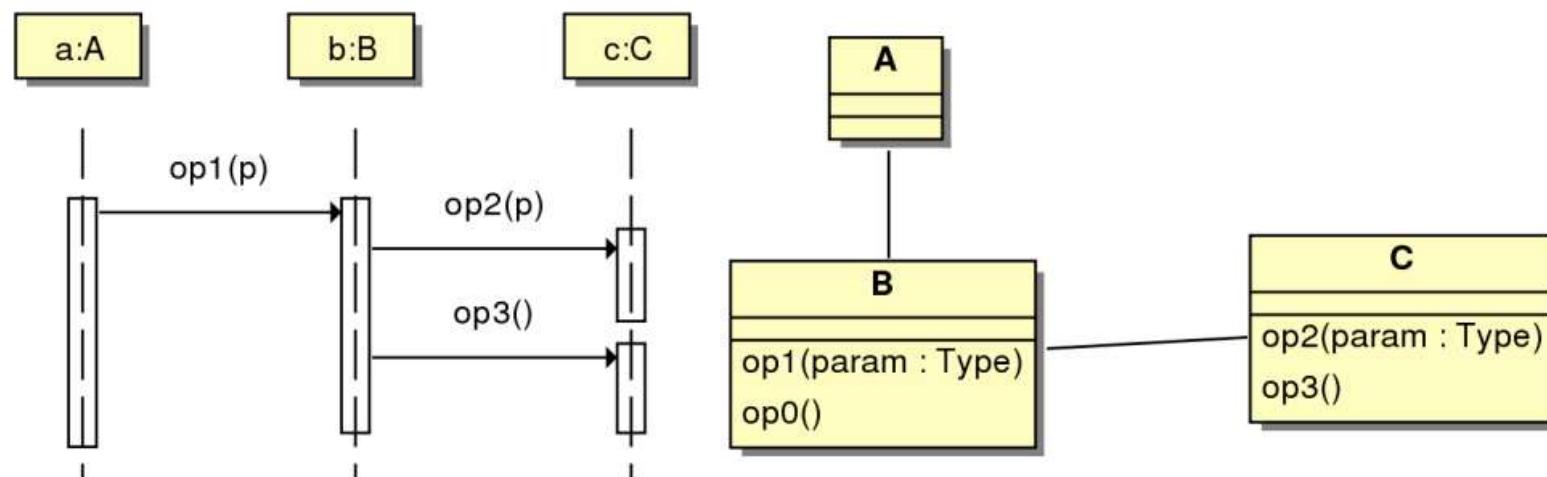
Dans le diagramme de séquence, les envois de messages sont représentés par des flèches horizontales qui vont de la ligne de vie de l'objet émetteur vers la ligne de vie de l'objet récepteur du message.

- **Les messages synchrones** : (*flèche avec un triangle plein à son extrémité*).
- **Les messages asynchrones** : (*simple flèche*)
- **Les messages de retour (réponses)** : (*simple flèche en pointillés*).



# Les Messages Synchrones

- 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.



# Messages Asynchrones

Dans le cas d'un message asynchrone, l'expéditeur n'attend pas la fin de l'activation de la méthode invoquée chez le destinataire. Un message asynchrone peut être :

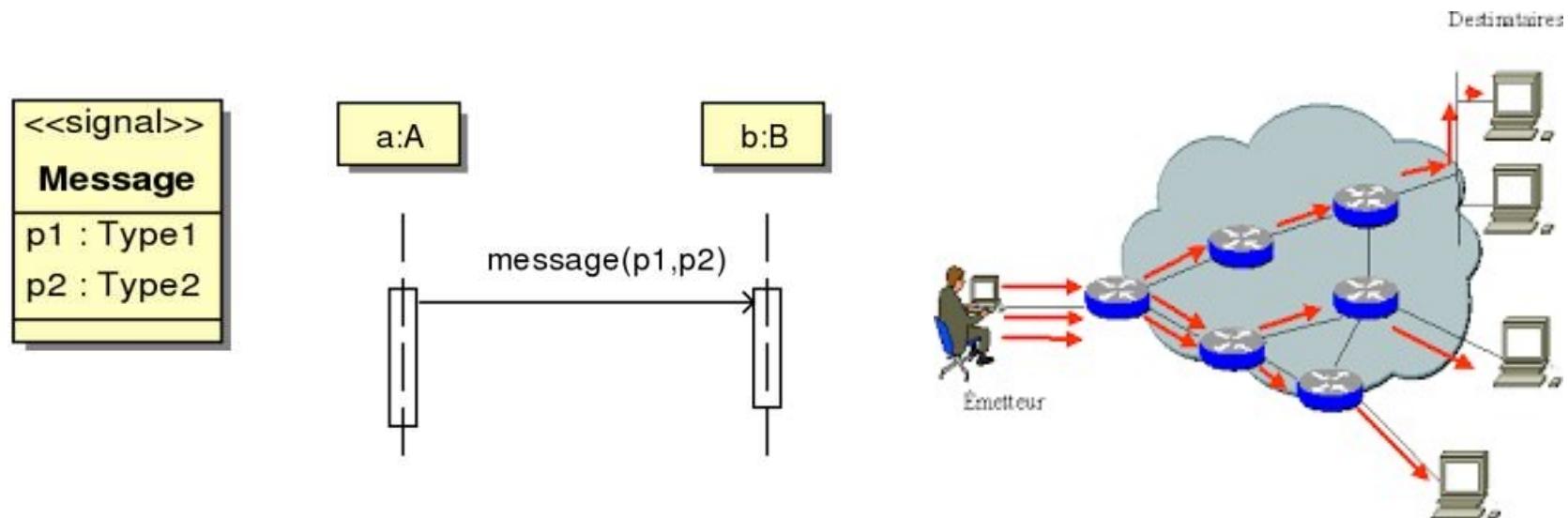
- **Un appel de méthode : Fréquent dans un système multithreads (multitâche):**

Ainsi, l'objet expéditeur n'étant pas bloqué pendant l'exécution de la méthode, continuer ainsi à envoyer d'autres messages.

# Messages Asynchrones

Un message asynchrone peut être :

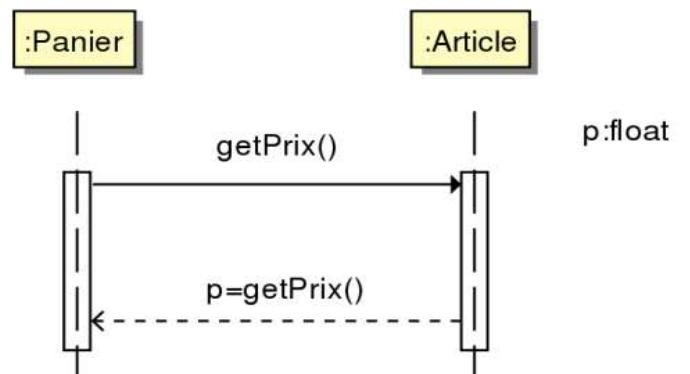
- **Un signal (cas le plus fréquent)** : dans le cas de broadcasting par exemple  
L'objet expéditeur transmet juste une information à l'objet destinataire.



# Message de retour

## Pour Message Synchrone:

- Le récepteur d'un message **synchrone** rend la main à l'émetteur du message en lui envoyant un message de retour.
- Les messages de retour sont **optionnels** : la fin de la période d'activité marque également la fin de l'exécution d'une méthode.
- Ils sont utilisés pour spécifier le résultat de la méthode invoquée.



## Pour Message Asynchrone

- Le retour des messages asynchrones s'effectue par l'envoi de nouveaux messages asynchrones.

## Syntaxe des messages synchrones et asynchrones:

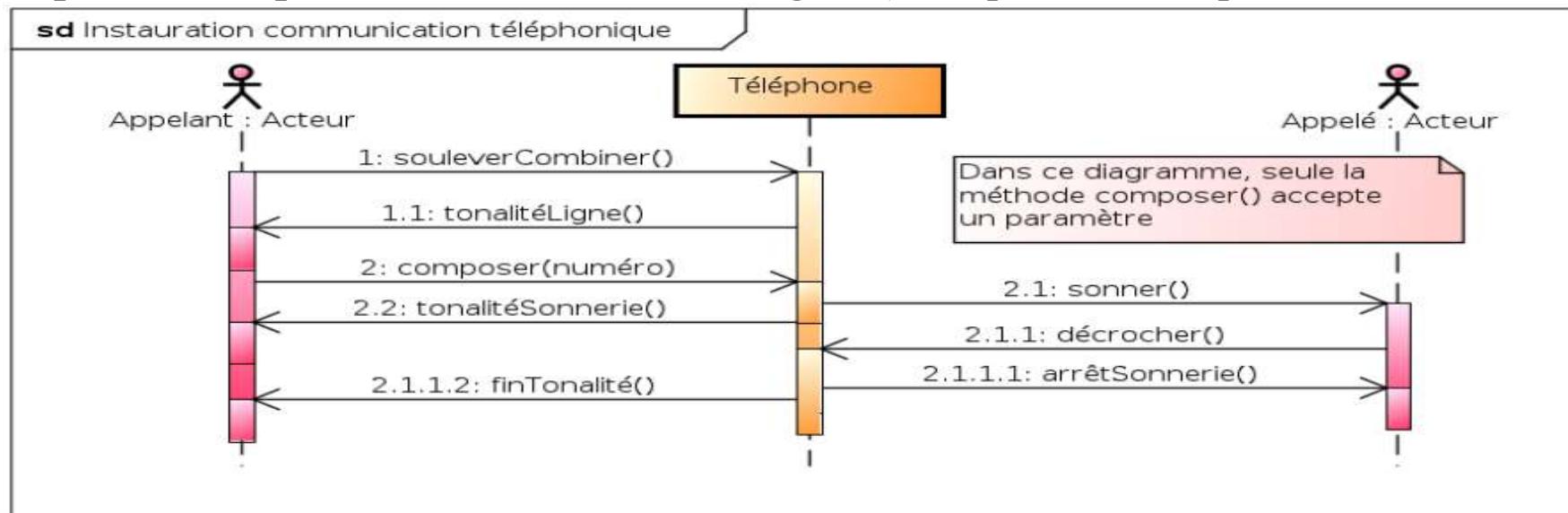
La plupart du temps, dans un diagramme de séquence, nous pouvons nous contenter de définir un message par :

- **Son nom** (qui est le nom de la méthode appelée ou du signal envoyé).

Nous pouvons lui adjoindre **facultativement** :

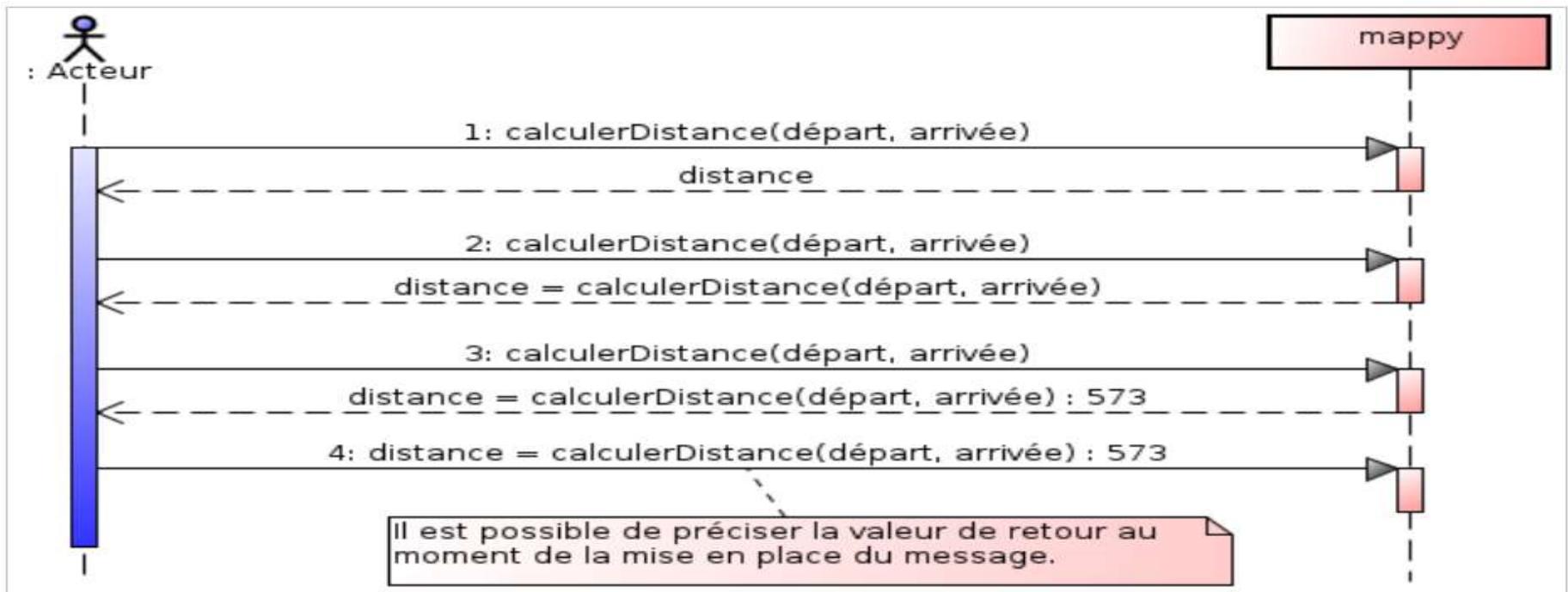
- **Une numérotation** devant le nom message (séparé du nom du message par 2 point ":" :). La numérotation s'effectue séquentiellement à partir de 1.

- **Les paramètres passés à la méthode ou au signal** (entre parenthèses après le nom du message).



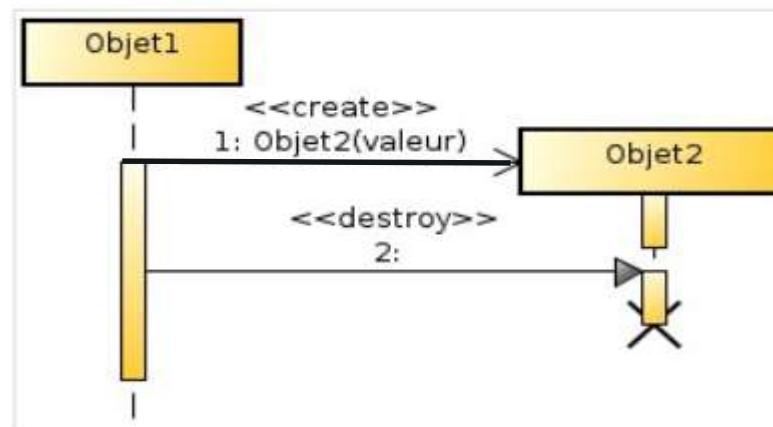
## Syntaxe des réponses: Messages retour

Comme pour les messages synchrones ou asynchrones, nous pouvons nous contenter de donner au message retour un simple nom, mais nous pouvons aussi les caractériser plus précisément en utilisant la syntaxe suivante : ***numéro : attribut = nomMessage (paramètres) : valeurDeRetour***.

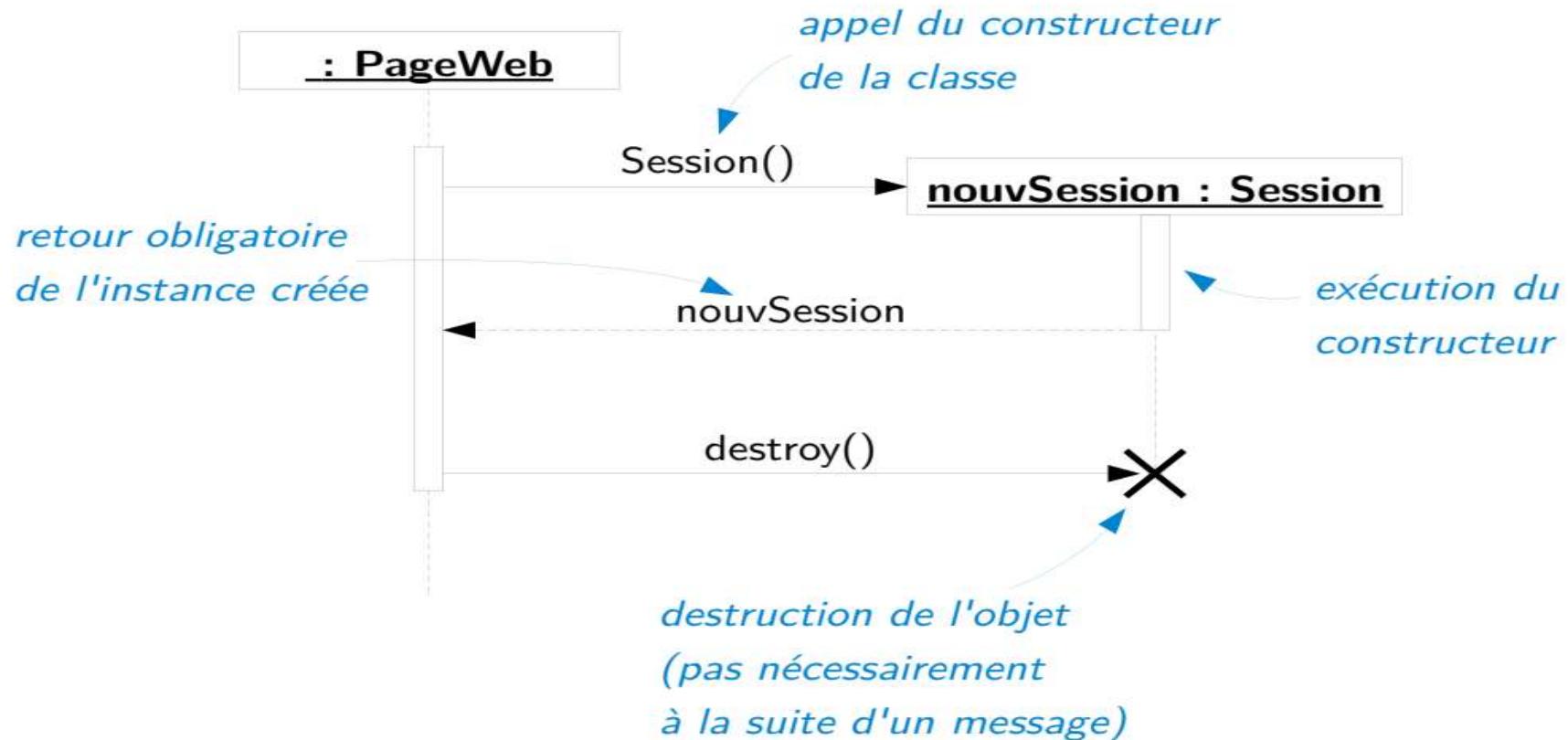


## Les Message de Création et de destruction d'objet

- **La création** d'un objet est matérialisée par un message spécifique, appel d'un **constructeur**, généralement accompagné du stéréotype « *create* » qui pointe sur le début (le sommet) de la ligne de vie de l'objet créé.
- **La destruction** d'un objet est représentée par une croix à la fin de sa ligne de vie. Dans ce cas là, il porte le stéréotype « *destroy* ».

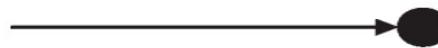


# Création et de destruction d'objet



# Messages Complet, perdu et trouvé

- **Un message complet:** est tel que les évènements d'envoi et de réception sont connus. Il est représenté par une flèche partant d'une ligne de vie et arrivant à une autre ligne de vie.
- **Un message perdu:** est tel que l'événement d'envoi est connu, mais pas l'événement de réception. La flèche part d'une ligne de vie mais arrive sur un cercle indépendant marquant la méconnaissance du destinataire.



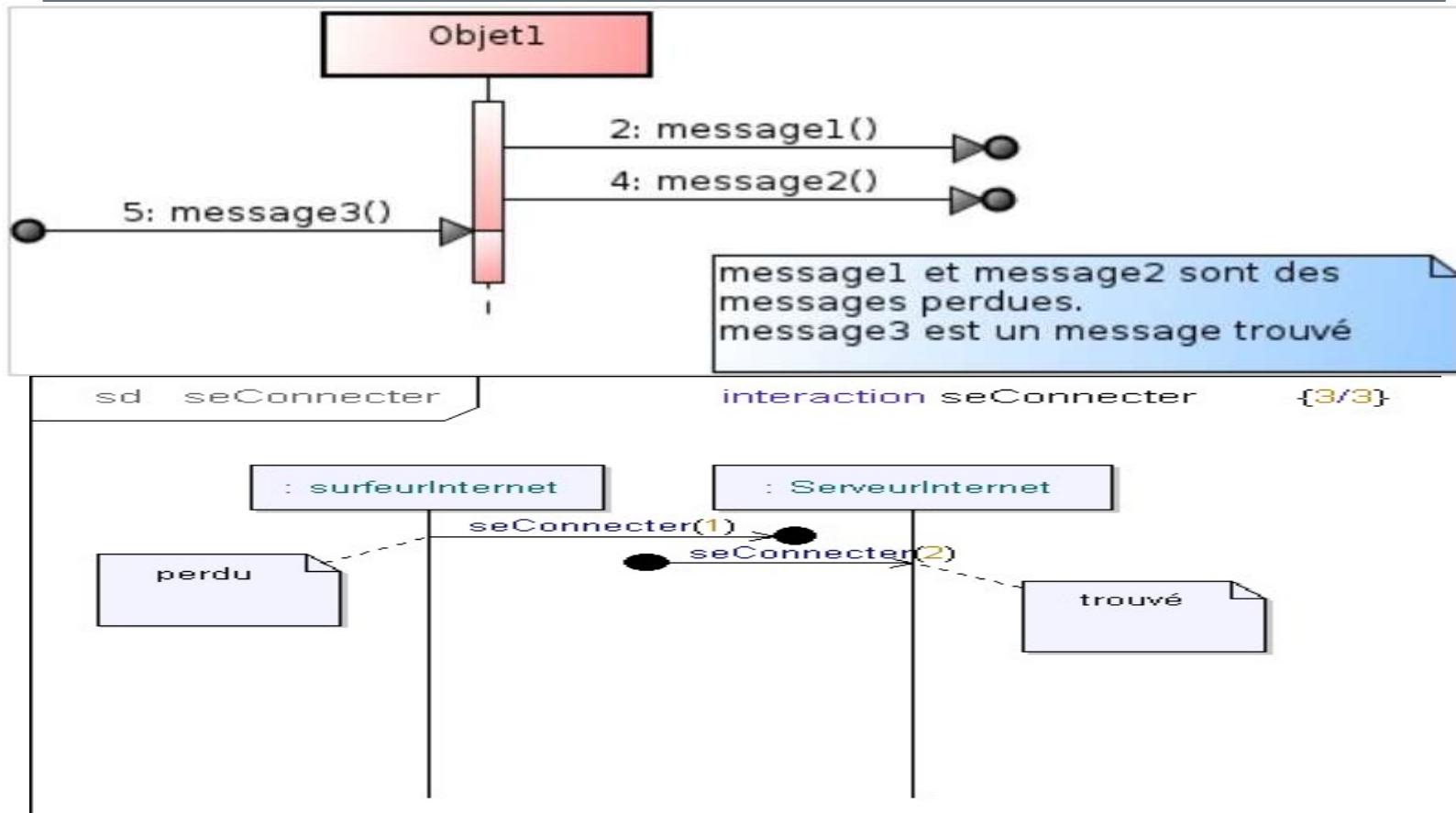
*Cette sorte de message permet de modéliser, par exemple, les scenarii de pertes de message sur un réseau.*

- **Un message trouvé:** est tel que l'événement de réception est connu, mais pas l'événement d'émission. Un message trouvé est un message pour lequel soit l'émetteur est inconnu, soit le message provient d'une source aléatoire.



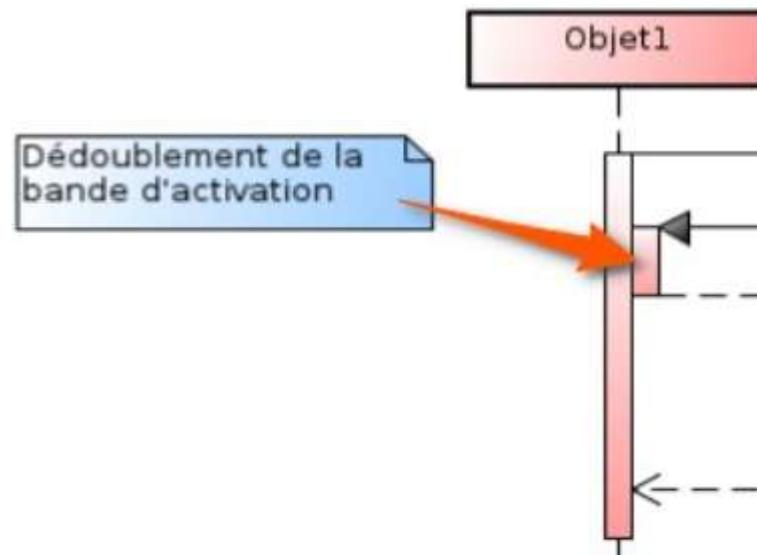
- *Ce message peut être utilisé pour modéliser le comportement d'un élément suite à la réception d'un message d'exception.*

# Messages Complet, perdu et trouvé



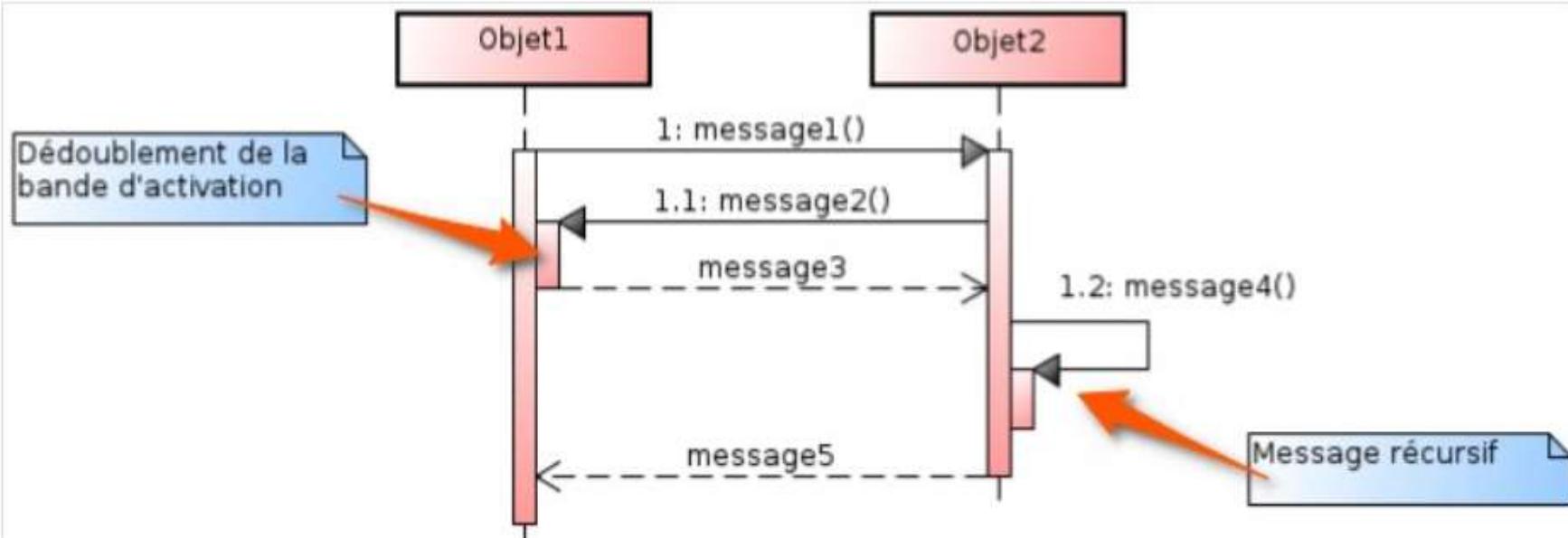
## Recouvrement des bandes d'activations

Lorsqu'un objet est déjà activé il peut quand même recevoir d'autres messages (*appel d'une autre de ses méthodes*), cela se représente par un dédoublement de la bande d'activation.

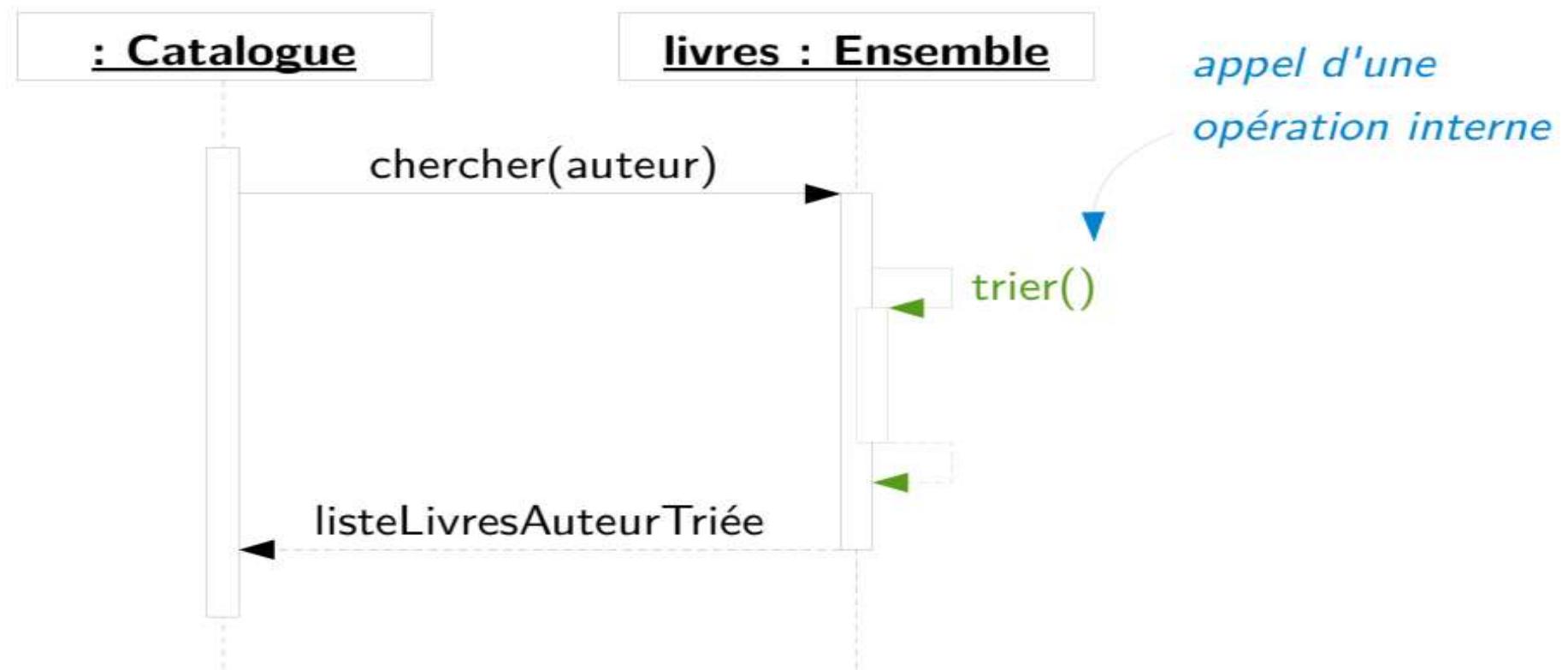


## Message récursif

- Appelé aussi message réflexif envoyé d'un objet vers lui-même. On peut montrer qu'un objet s'envoie un message à lui-même (***utilisation d'une autre méthode du même objet***) à l'aide d'une flèche circulaire et Cela se représente là aussi par un dédoublement de la bande d'activation.

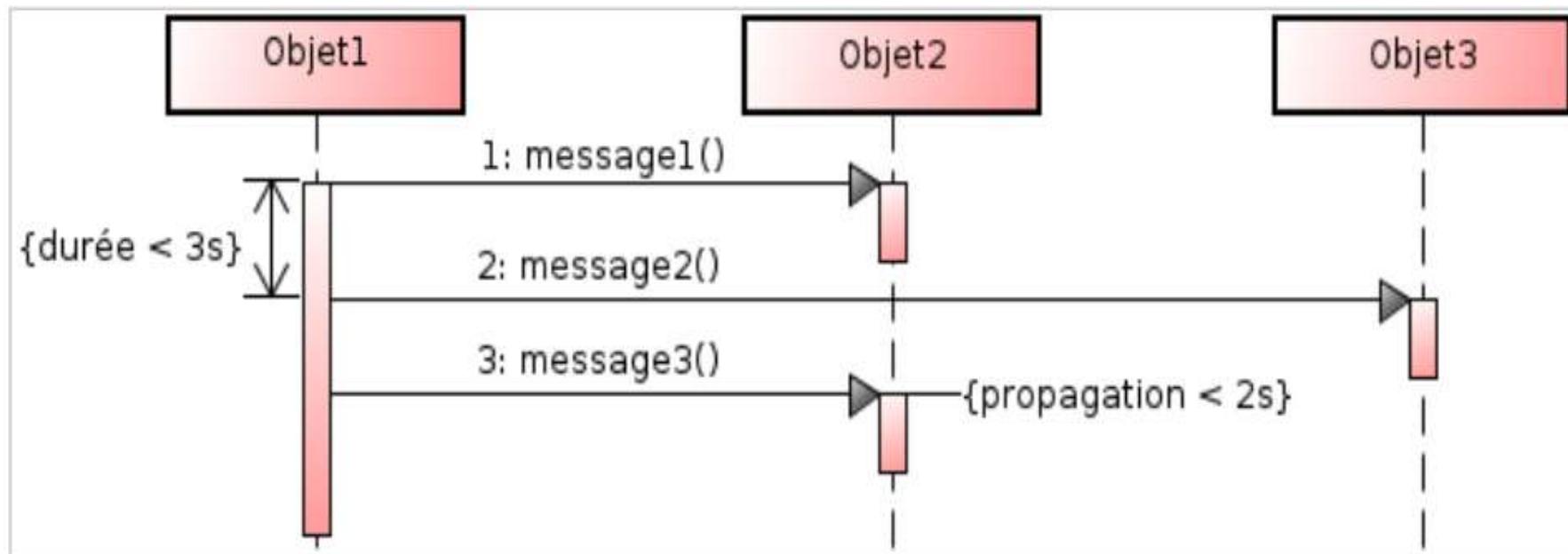


# Message réflexif



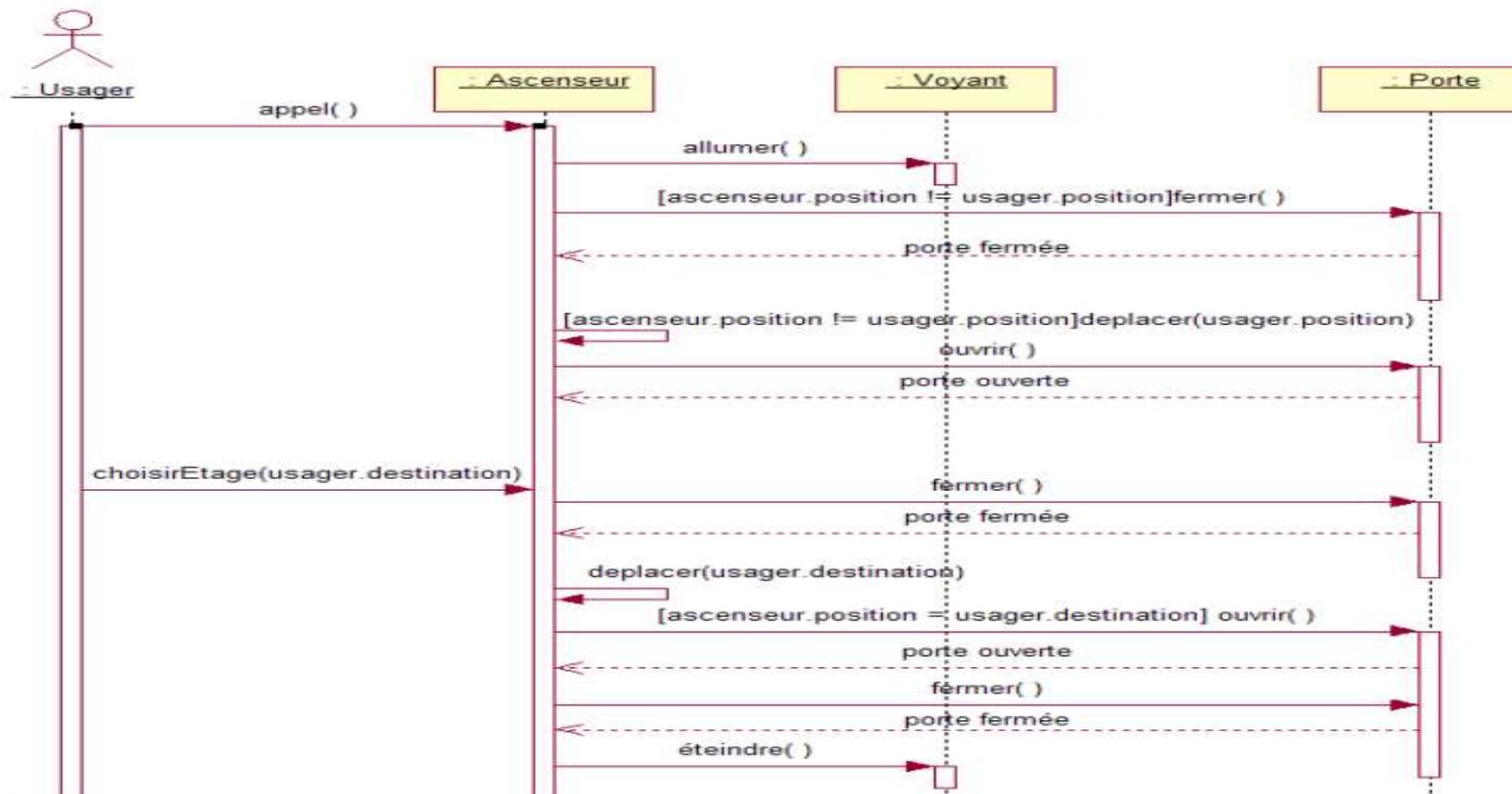
## Contraintes temporelles

Des repères temporels avec des contraintes peuvent être placés le long de la ligne de vie. Un message avec un temps de propagation non négligeable peut être représenté par une flèche oblique ou en l'écrivant explicitement.



# Diagramme de Séquence: Exemple 1

- Sur l'exemple suivant, le DS permet de représenter graphiquement un scénario:



## **Exercice 1:**

### **Description textuelle du cas d'utilisation : « Emprunter un livre»**

La rubrique « **enchaînement nominal** » du cas d'utilisation « **Emprunter un livre** » se fait selon le scénario suivant;

- 1) L'adhérent se présente au comptoir et la bibliothécaire saisit la fonctionnalité pour emprunter un livre de l'application.
- 2) D'abord, il faut vérifier si l'adhérent a le droit d'emprunter des livres (carte valide, nombre de livres déjà empruntés ne dépasse pas un seuil fixé, ...).
- 3) Ensuite, il faut vérifier si le livre est disponible.
- 4) Si tout va bien, on crée un nouveau prêt avec la date de prêt et la date de retour, associé avec l'adhérent et le livre choisi.
- 5) On rend le livre indisponible.
- 6) On incrémente le nombre de livres empruntés par l'adhérent.

**Etablir le diagramme de séquence de ce scénario de cas d'utilisation Emprunter livre**

# Solution 1

## **Exercice 2:**

### **Description textuelle du cas d'utilisation : « Retirer de l'argent »**

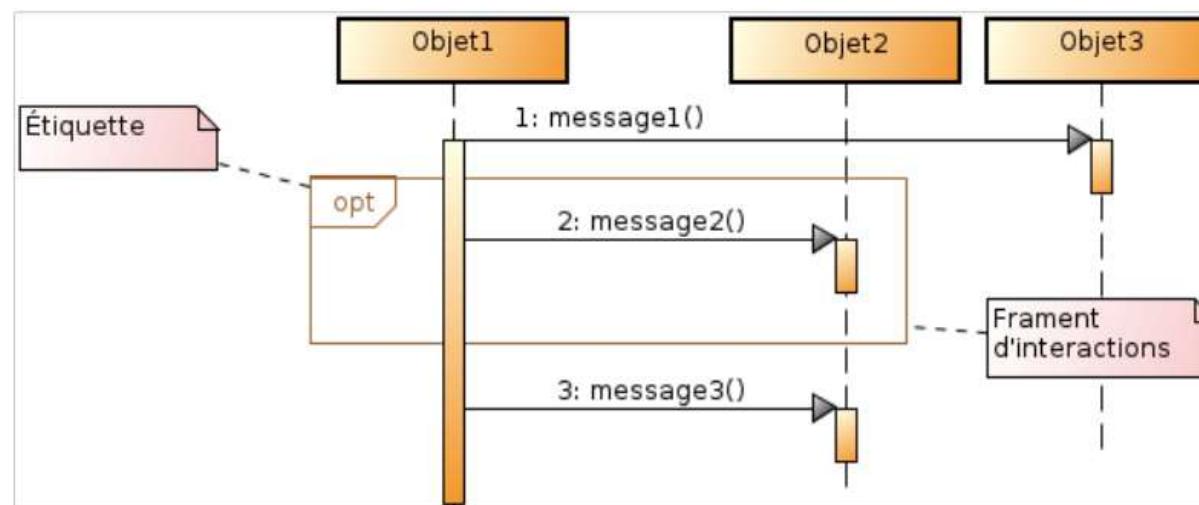
- La rubrique « **enchaînement nominal** » du cas d'utilisation « **retrait d'espèces** » contient les éléments suivants :
  1. Le guichetier saisit le numéro de compte du client ;
  2. L'application valide le compte auprès du système central ;
  3. Le guichetier demande un retrait de 1000 dh ;
  4. Le système « guichet » interroge le système central pour s'assurer que le compte est suffisamment approvisionné ;
  5. Le système central effectue le débit du compte ;
  6. En retour, le système notifie au guichetier qu'il peut délivrer le montant demandé.

**Question : Donner le diagramme de séquences associé à cette description textuelle.**

# Solution 2

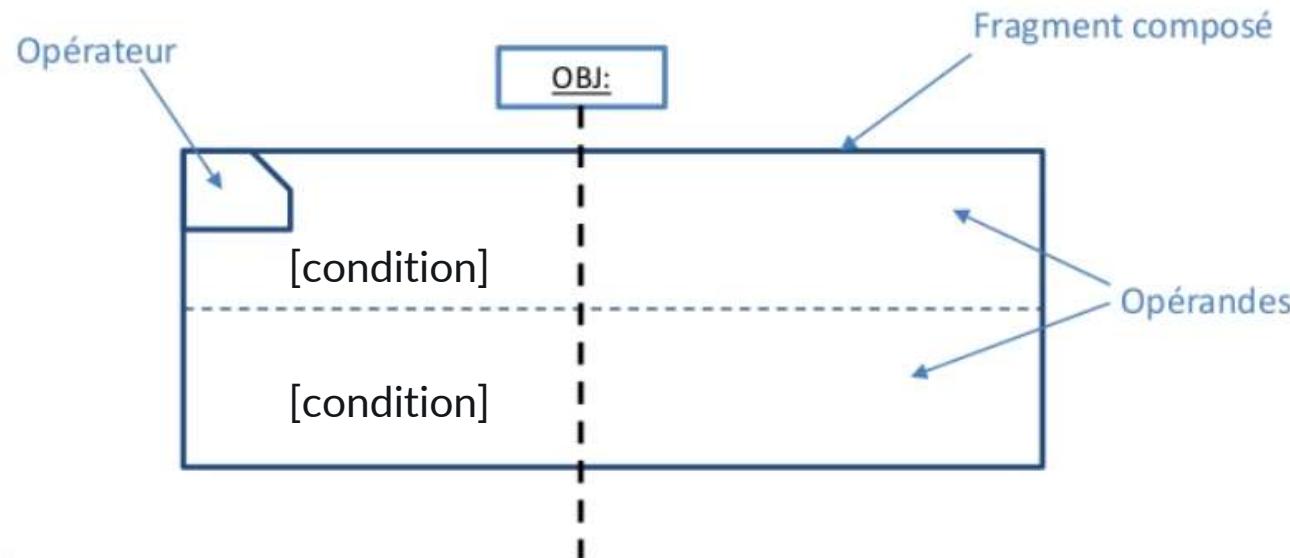
# Fragments d'interactions combinés

Les fragments combinés sont des unités d'interaction définies dans UML. Ils offrent le support de nombreuses constructions comme les alternatives, les boucles, le parallélisme, etc. et confèrent au diagramme de séquence le statut d'un véritable modèle des interactions.



## Fragments d'interactions combinés

Un fragment d'interactions est une partie du diagramme de séquence (délimitée par un rectangle) associée à une étiquette (dans le coin supérieur gauche). L'étiquette contient un **opérateur d'interaction** qui permet de décrire des modalités d'exécution des messages à l'intérieur du cadre.



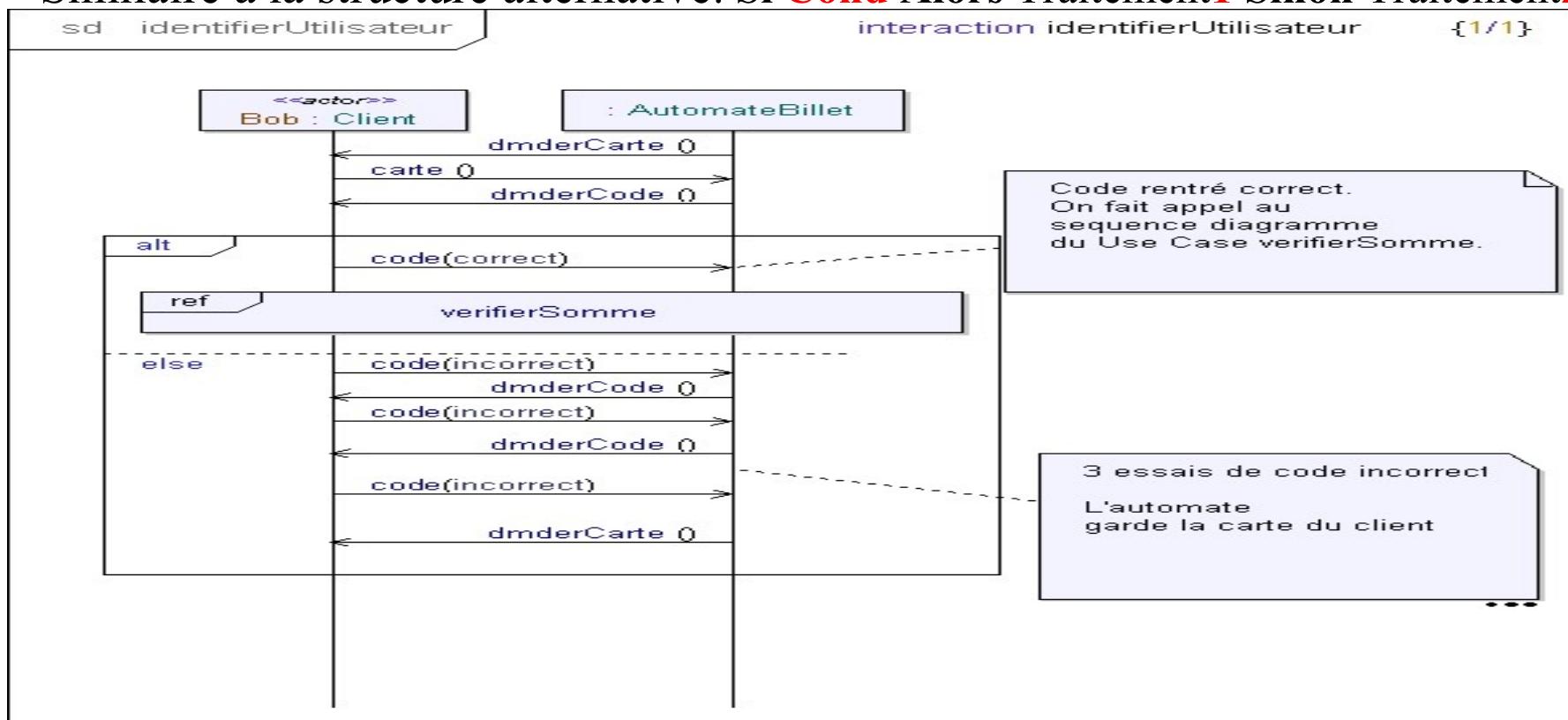
## Fragments d'interactions combinés

Il existe 13 opérateurs parmi lesquels on cite les opérateurs les plus utilisés:  
Principalement 8 utilisés :

- Opérateurs de choix et de boucle : **Alternative, Option, Loop,Break**
- Opérateurs contrôlant l'envoi en parallèle de messages: **Parallel, Critical region**
- Opérateurs fixant l'ordre d'envoi des messages: **strict sequencing,weak sequencing**

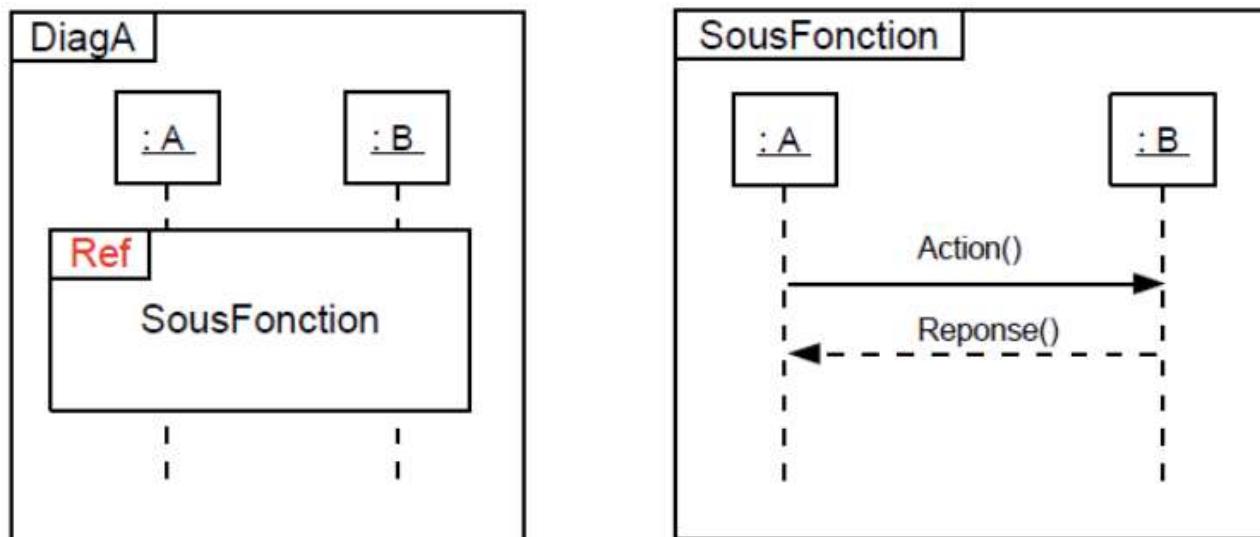
# Opérateur Alternative

- Le fragment représente des choix.
- Similaire à la structure alternative: Si **Cond** Alors Traitement1 Sinon Traitement2

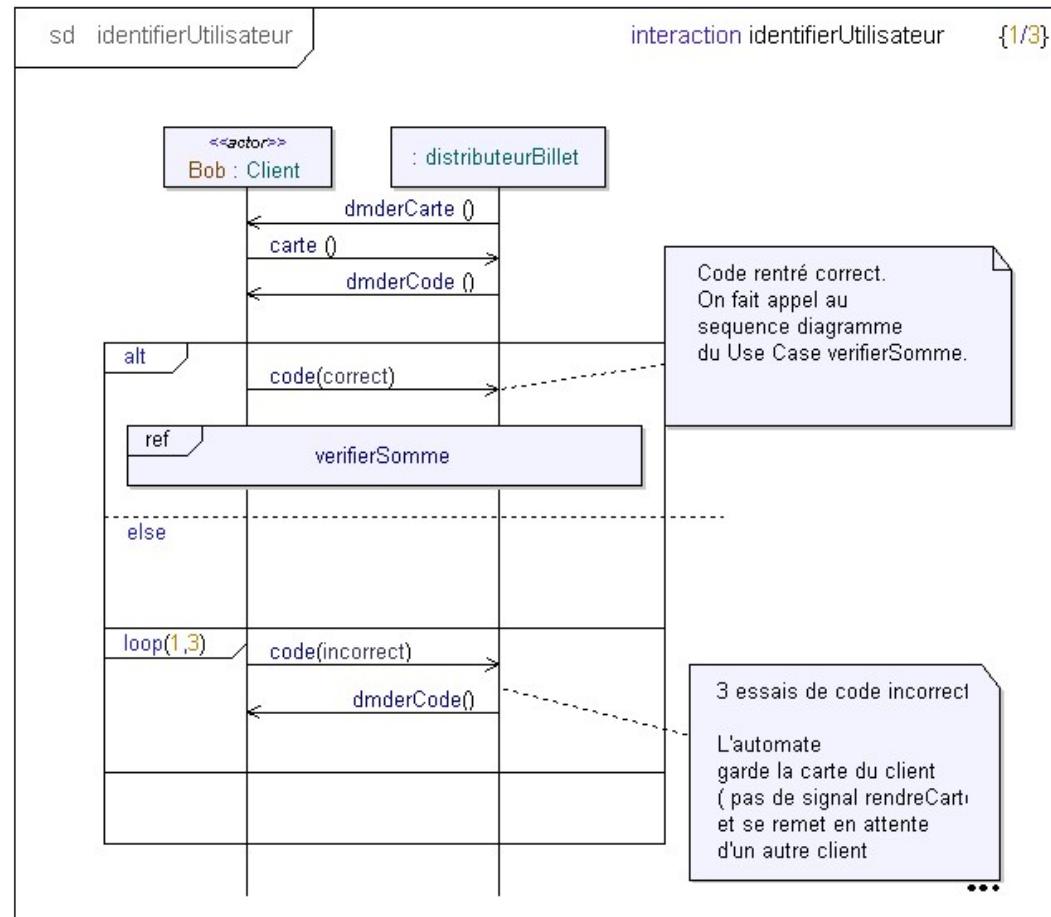


# Opérateur Référence (ref)

- Un fragment ref permet d'inclure un autre diagramme de séquences. Il indique une référence vers un autre diagramme de séquence existant

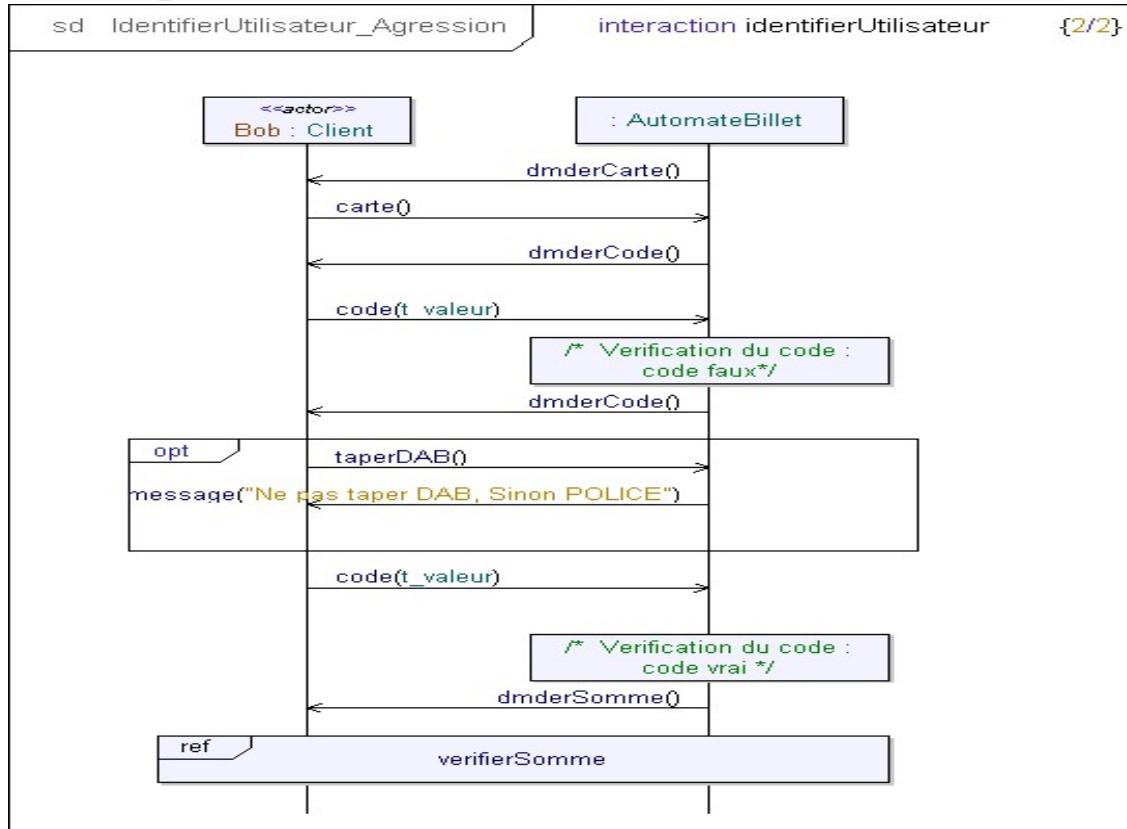


# Opérateur référence (ref)



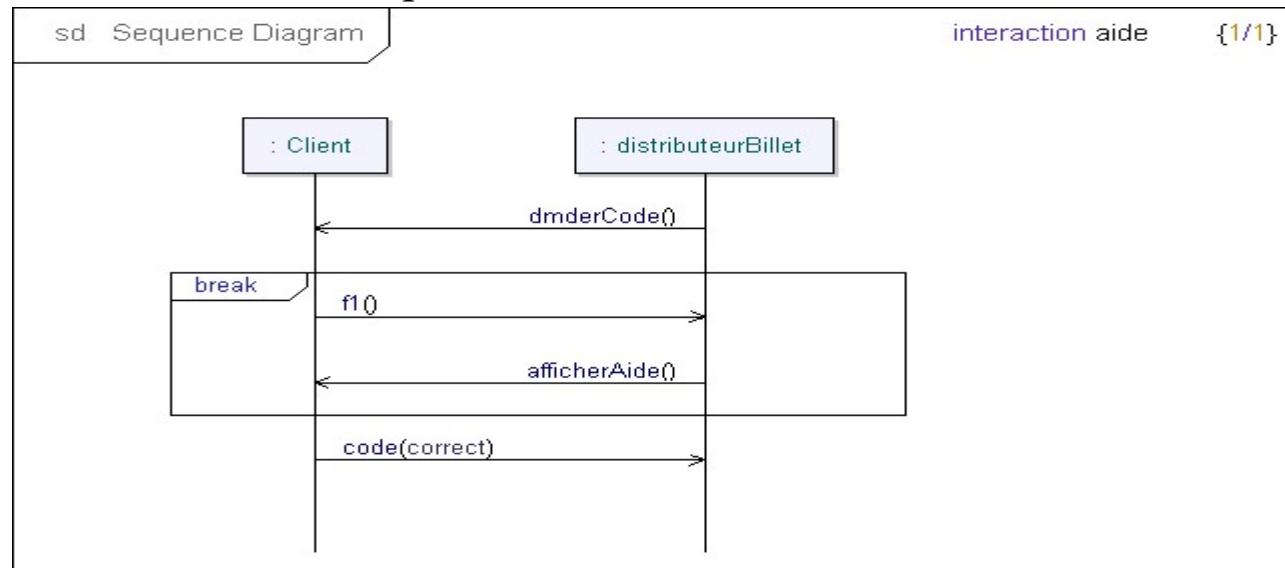
# Opérateur Option (opt)

- Équivalent à un opérateur alternative avec une seule condition



# Opérateur Break

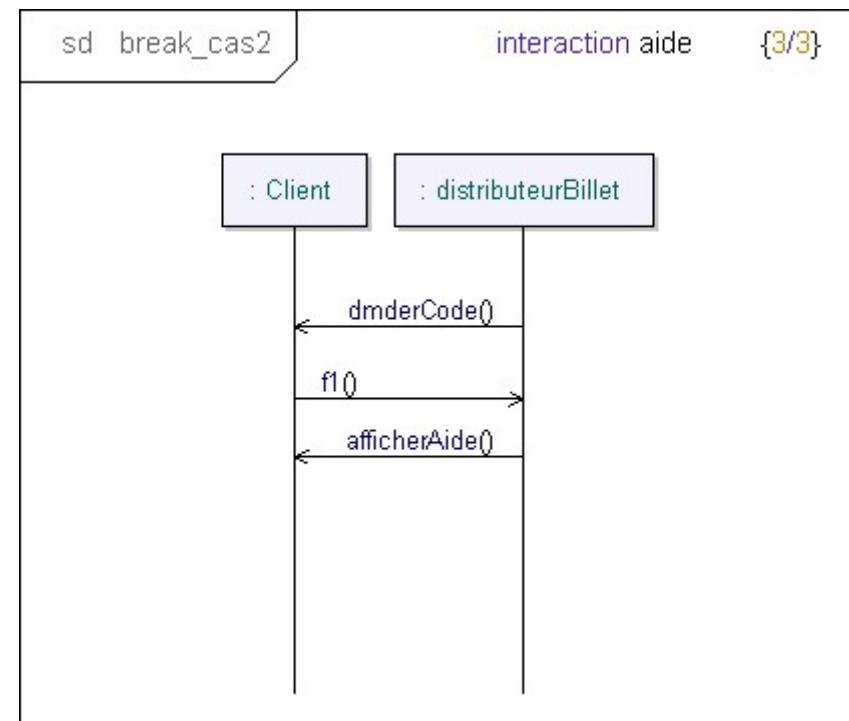
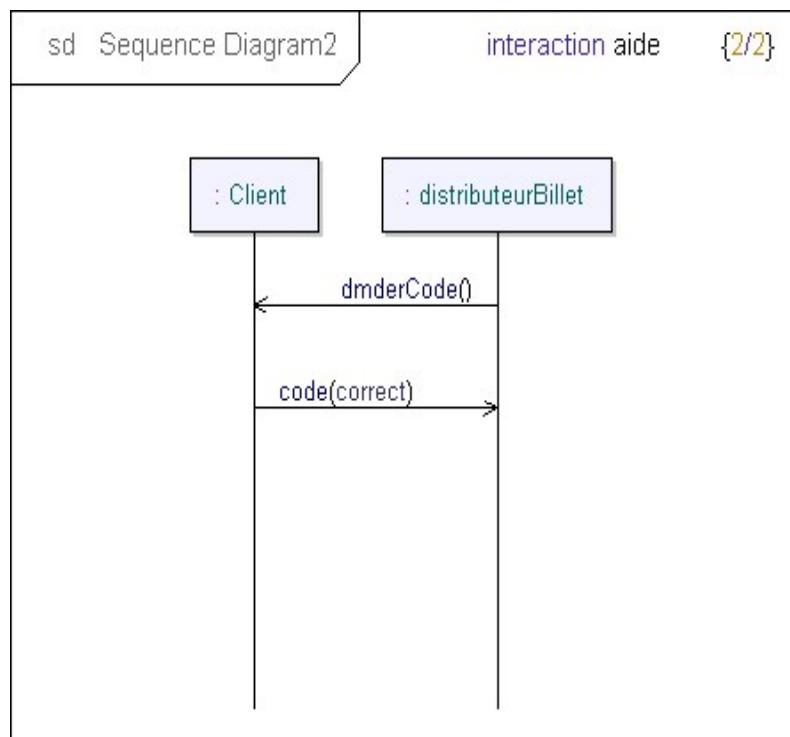
- Il est utilisé dans les fragments combinés qui représentent des scénarios d'**exception** en quelque sorte.
- Les interactions de ce fragment seront exécutées à la place des interactions décrites en dessous.
- Il y a donc une notion d'interruption du flot « normal » des interactions.



Si le client choisit de consulter l'aide au lieu de rentrer le code, le flot d'interaction relatif à la saisie du code est interrompu

# Opérateur Break

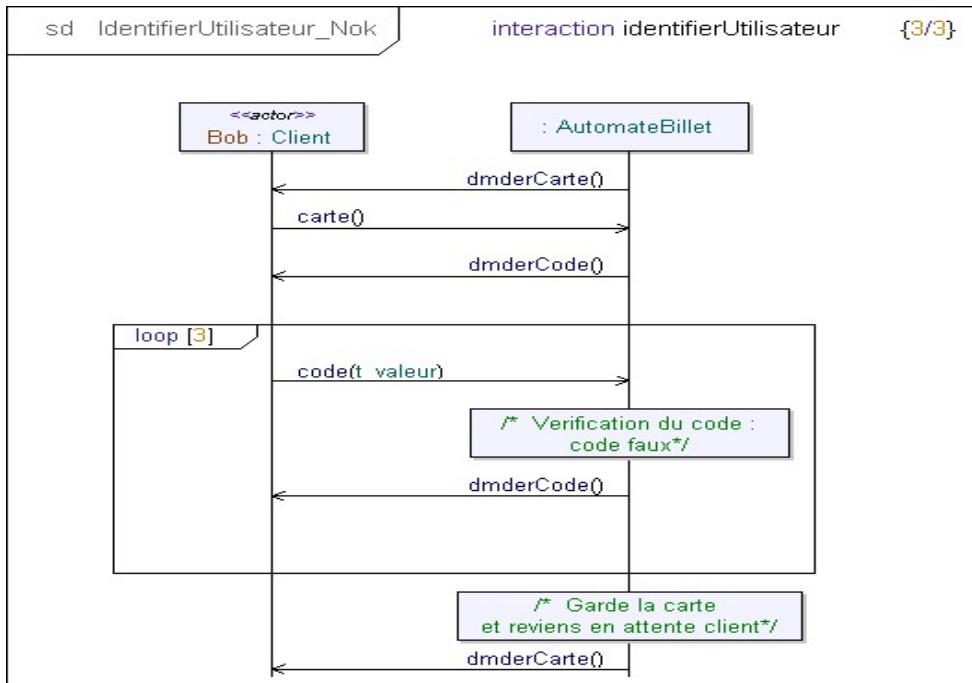
L'équivalent du diagramme de séquence précédent sans l'opérateur **break** correspond aux deux diagrammes de séquence ci-après :



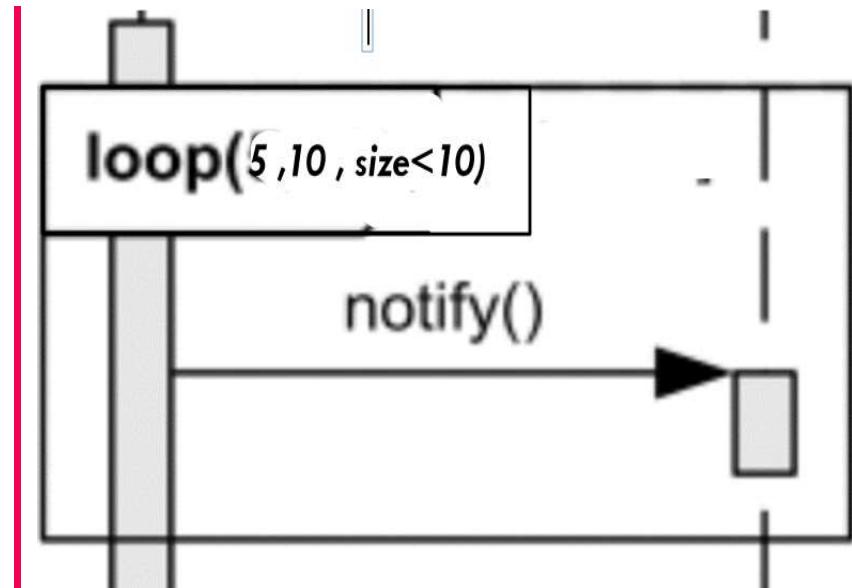
# Opérateur Loop

Permet de spécifier une boucle, le nombre de répétition.

Syntaxe: **loop [min, max, condition]**

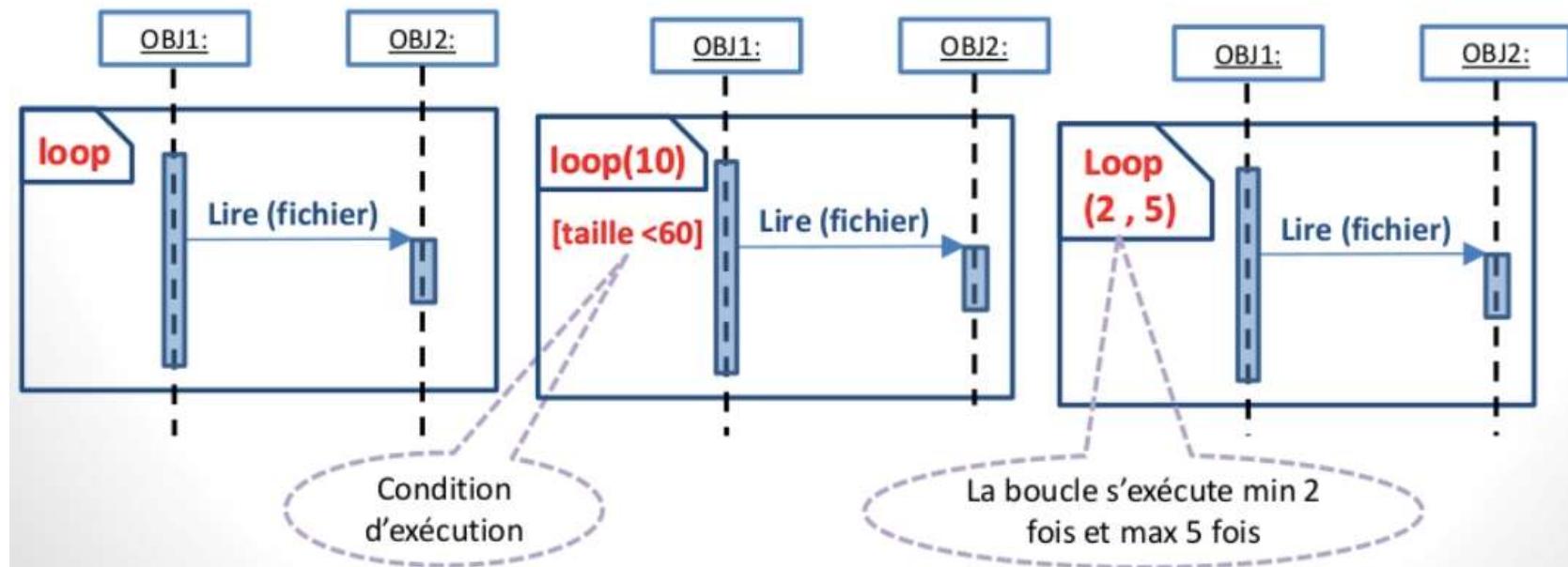


Si l'utilisateur se trompe trois fois de code, la carte est gardée et le distributeur se remet en mode d'attente d'une carte



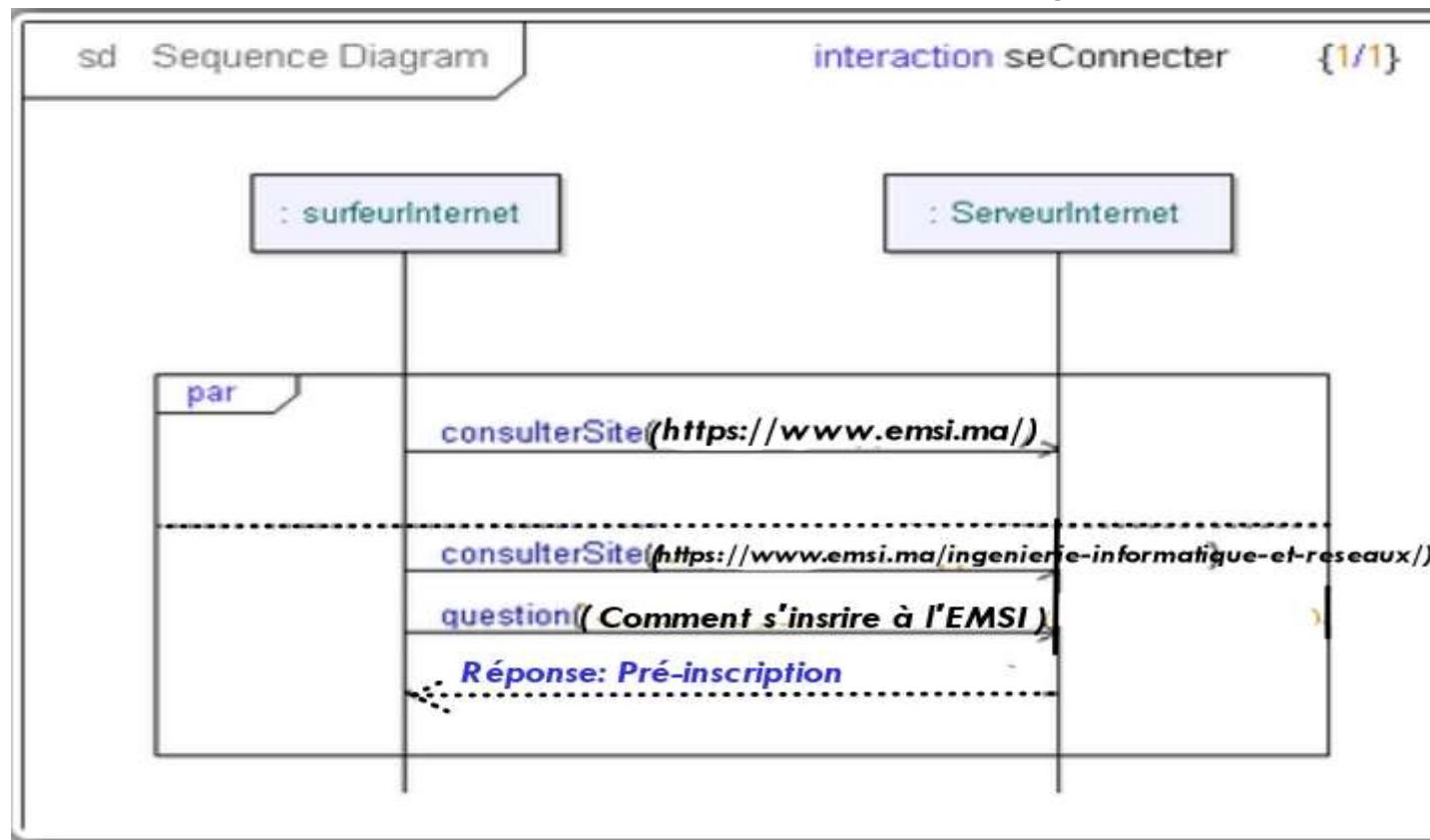
- La boucle s'exécute au minimum 5 fois et au maximum 10 fois
- Si la condition est fausse, on sort de la boucle, quel que soit le nombre d'exécutions de la boucle

## Fragment d'interaction avec opérateur « loop »



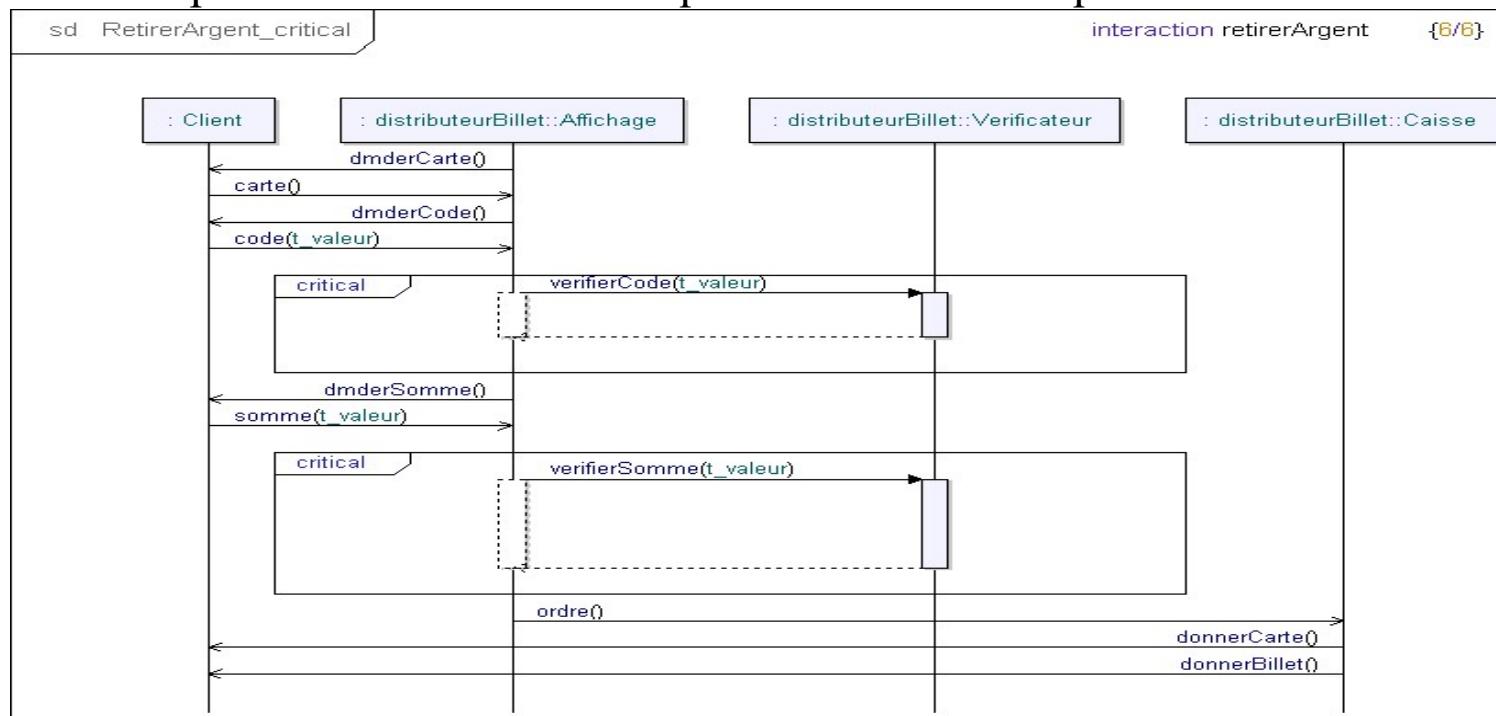
# Opérateur Parallel (par)

Spécifie l'exécution en parallèle de plusieurs sous fragments .



# Opérateur Critical

- Une section critique permet d'indiquer que les interactions décrites dans cet opérateur ne peuvent pas être interrompues par d'autres interactions décrites dans le diagramme.
- L'opérateur impose un traitement atomique des interactions qu'il contient.



L'utilisateur ne peut pas obtenir des billets avec un code erroné et une somme demandée incorrecte.

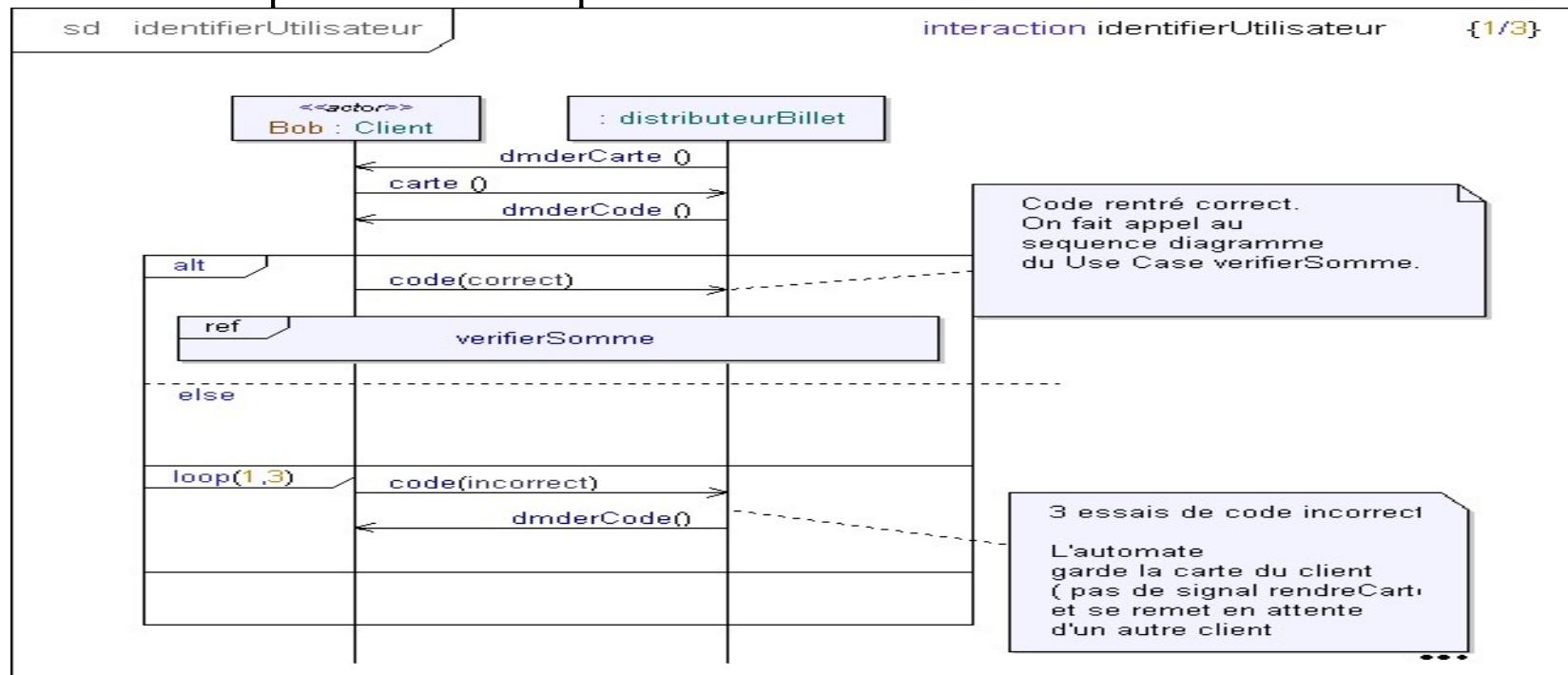
## Les fragments combinés en exemple

Le diagramme de séquence présenté précédemment dans le cours ne correspond qu'au scénario nominal du cas d'utilisation. Des notes (optionnelles) permettent d'indiquer les scénarios alternatifs et d'exceptions.

Chaque scénario nominal ou d'exception peut être documenté par son propre diagramme de séquence. Grâce aux fragments d'interactions, il est possible de documenter un scénario alternatif (ou d'exception) dans le diagramme de séquence du scénario nominale.

# Combiner les opérateurs

- Les fragments combinés et leurs opérateurs peuvent être combinés/mixés en vue de décrire des comportements complexes



Si l'utilisateur saisi le code correctement alors dans ce cas le SD (**verifierSomme**) relatif à la vérification du code est appelé, sinon, s'il se trompe trois fois de code, la carte est gardée et le distributeur se remet en mode d'attente d'une carte.