Université Sultan Moulay Slimane
Ecole Supérieure de Technologie Béni Mellal
Département ITG
DUT- IDIA 1 - S2
M122 - Ingénierie Logicielle avec Java et UML
Année universitaire: 2024-2025

Support de cours

Ingénierie logicielle avec Java et UML

Partie II: Modélisation UML

Chapitre VI: Diagramme de séquences

Pr. Siham BAKKOURI
Email: s.bakkouri@usms.ma

Plan

- INTRODUCTION
- CONCEPTS DE BASE
- LES ÉLÉMENTS GRAPHIQUES
- LES FRAGMENTS COMBINÉS
- LES UTILISATIONS PRATIQUES D'UN DSE
- EXERCICES

Introduction

- 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 spécifie **QUI** sera à l' œuvre dans le système pour réaliser les fonctionnalités décrites par les diagrammes de cas d'utilisation.
- 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 s'échangent des messages.
- Les ***acteurs*** interagissent avec le système au moyen d'IHM (Interfaces Homme-Machine).

Introduction

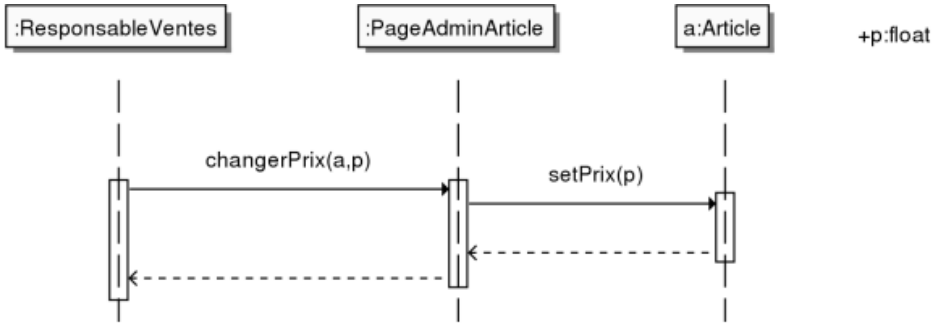
Les **diagrammes de séquences** montrent les interactions entre objets selon un point de vue temporel. Ils permettent la description de **scénarios types** ainsi que des **exceptions**. Ils ont deux utilisations principales :

1. **Documentation des Cas d'Utilisation (point de vue Fonctionnel):** Les diagrammes de séquences sont utilisés pour documenter les Cas d'Utilisation, offrant un aperçu visuel des interactions entre les acteurs et les systèmes, mettant l'accent sur le comportement fonctionnel du système.
2. **Représentation précise des interactions (point de vue Dynamique):** Ces diagrammes offrent une représentation détaillée des interactions dynamiques entre les objets, identifiant les messages échangés, les envois, les réceptions, etc. Ils sont essentiels pour comprendre en profondeur le comportement en temps réel des systèmes logiciels.

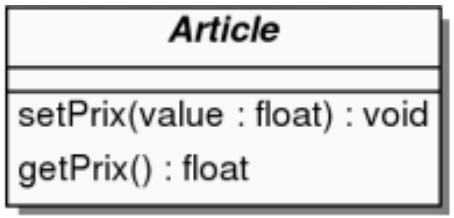
Exemple d'interaction

- **Cas d'utilisation :**

- **Diagramme de séquences** correspondant :



- Opérations nécessaires dans le **diagramme de classes** :



Concepts de base

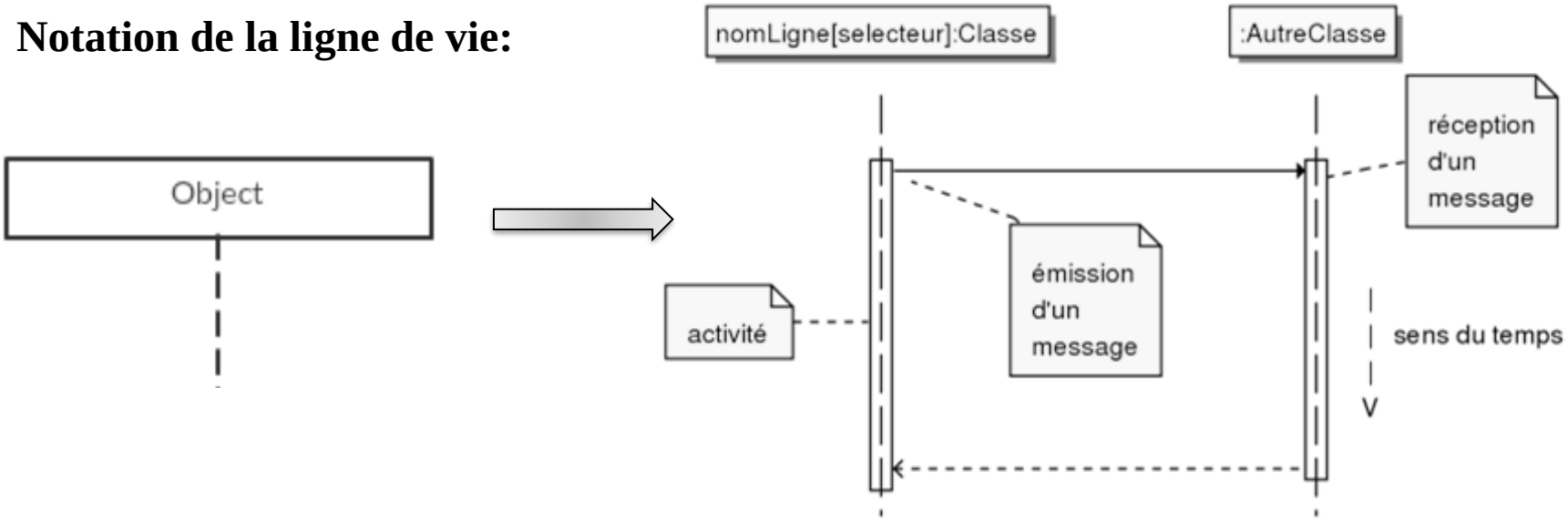
- **Les participants** (le plus souvent des objets)
  - Une ligne de vie : représente un participant à une interaction (objet ou acteur).
  - Des zones d'activation
- **Les messages**
  - L'opération et éventuellement ses paramètres
  - Éventuellement son résultat
- **Des Fragments combinés**
  - Alt : conditionnelle
  - Loop : boucle
  - Ref : référence a un autre diagramme de séquence (=appel de fonction)
  - …….

→ Ligne de vie

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

nomLigneDeVie [*selecteur*]: nomClasseOuActeur

- Dans le cas d'une collection de participants, un sélecteur permet de choisir un objet parmi n (par exemple objets[2]).
- **Notation de la ligne de vie:**

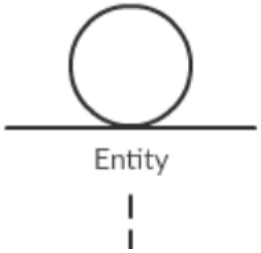


→ **Ligne de vie**

❑ Autres notations

- Une notation de ligne de vie avec un symbole d'élément **d'acteur** est utilisée lorsque le diagramme de séquence particulier appartient à un cas d'utilisation.

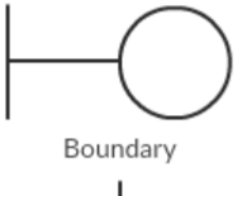
- Une ligne de vie avec un élément **d'entité** représente les données du système. Par exemple, dans une application de service à la clientèle, l'entité *Customer* gère toutes les données relatives à un client.



→ **Ligne de vie**

❑ Autres notations

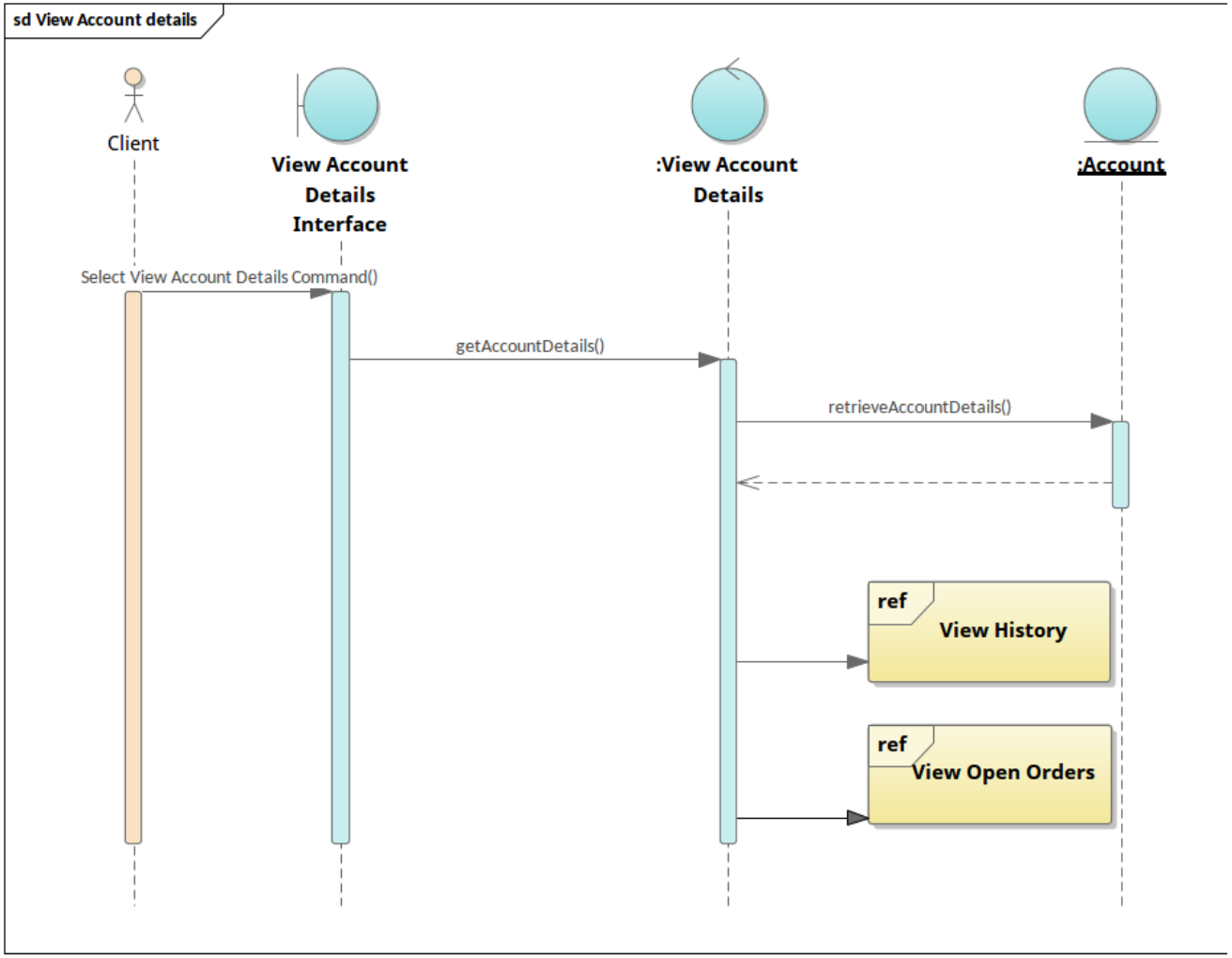
- Une ligne de vie avec un **élément de limite** indique une limite de système/élément logiciel dans un système ; par exemple, les écrans d'interface utilisateur, les passerelles de base de données ou les menus avec lesquels les utilisateurs interagissent, sont des limites.



- Une ligne de vie avec un **élément de contrôle** indique une entité ou un gestionnaire qui contrôle. Elle organise et planifie les interactions entre les frontières et les entités et sert de médiateur entre elles.

Les éléments graphiques

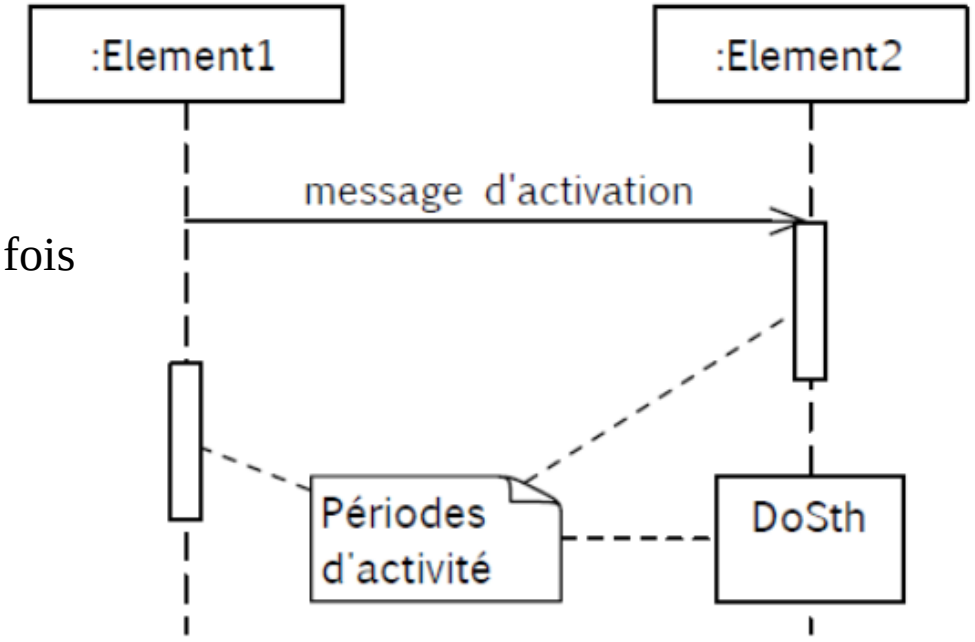
Exemple:



→ Barres d'activation/Période d'activité

- La **barre d'activation (**ou **période d'activité)** est la boîte placée sur la ligne de vie.
- Elle est utilisée pour indiquer qu'un objet est **actif** (ou **instancié**) lors d'une interaction entre deux objets → Période durant laquelle un objet effectue une action
- La longueur du rectangle indique la durée pendant laquelle les objets restent actifs.

  → Etat **actif** (durée de vie)
  → un objet peut être actif plusieurs fois



→ Les messages

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

→ Les messages

❑ Principaux types de messages

- **Un message synchrone** bloque l'expéditeur jusqu'à la réponse du destinataire. Le flot de contrôle passe de l'émetteur au récepteur. ***Typiquement*** : **appel de méthode**-

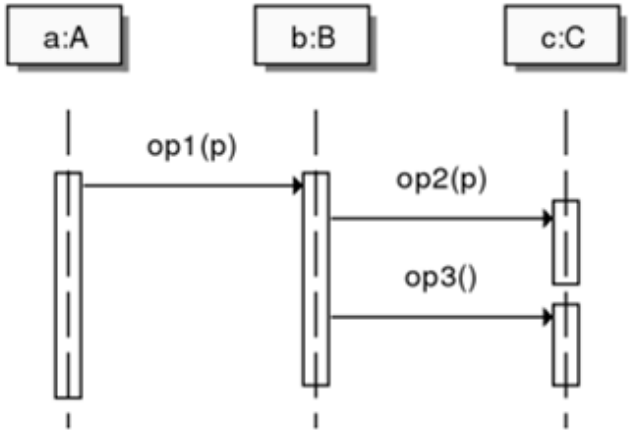
  → Si un objet ***A*** invoque une méthode d'un objet ***B***, ***A*** reste bloqué tant que ***B*** n'a pas terminé.

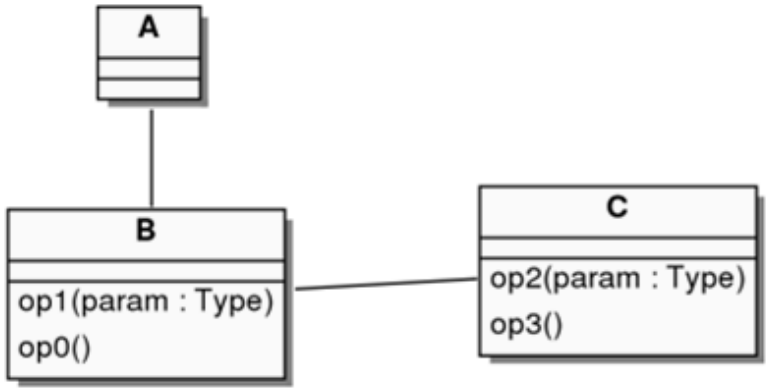
- On peut associer aux messages d'appel de méthode un ***message de retour*** (en ***pointillés***) marquant la reprise du contrôle par l'objet émetteur du message ***synchrone***.
- **Un message asynchrone** n'est pas bloquant pour l'expéditeur. Le message envoyé peut être pris en compte par le récepteur à tout moment ou ignoré. ***Typiquement*** : **envoi de signal**.

→ Les messages

❑ Correspondance messages / opérations

- 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.

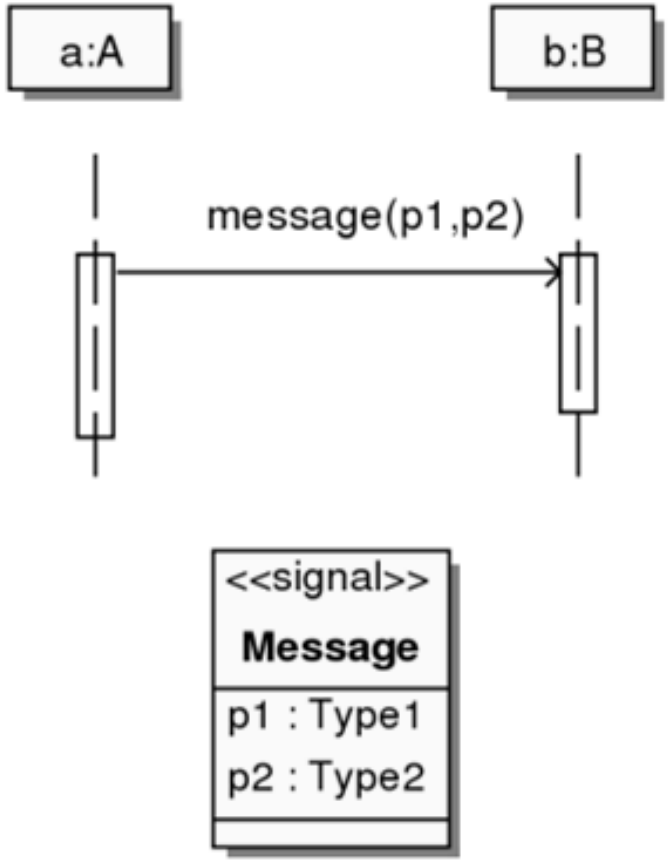




→ Les messages

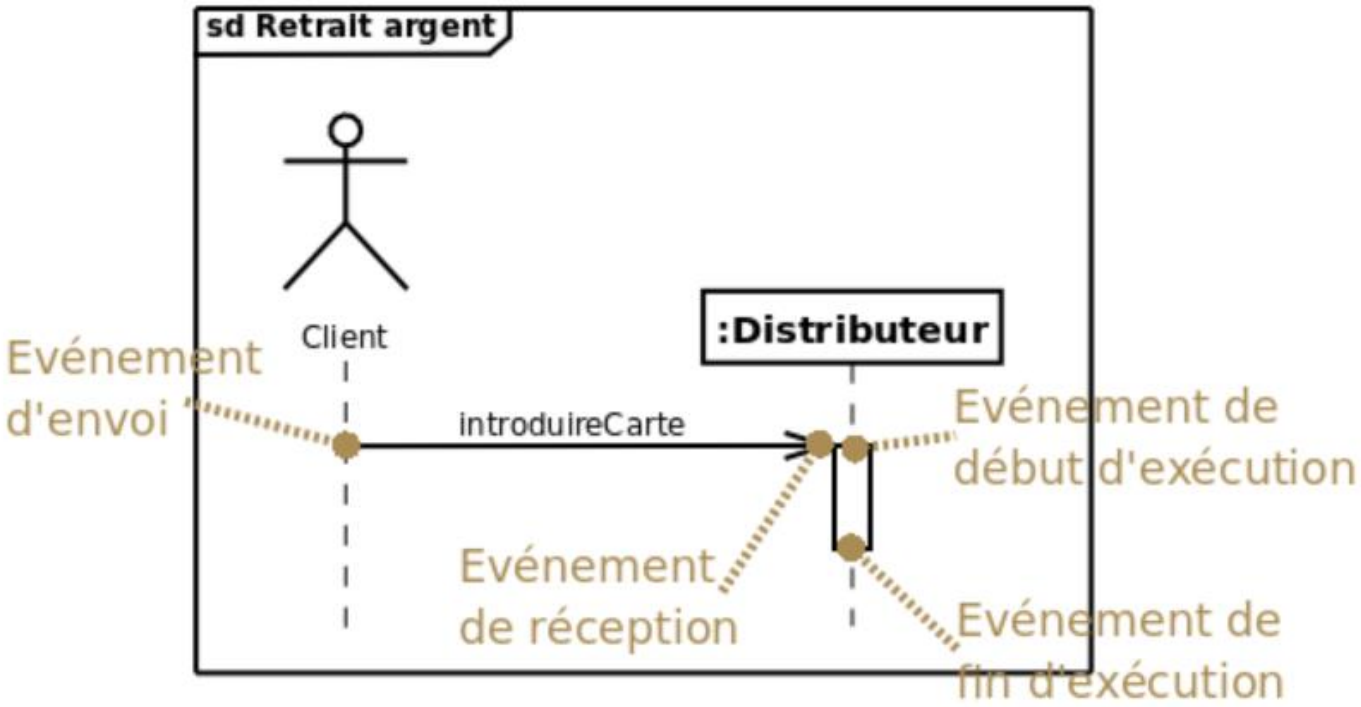
❑ Correspondance messages / signaux

- Les **messages asynchrones** correspondent à des signaux dans le diagramme de classes.
- Les **signaux** sont des objets dont la classe est stéréotypée signal et dont les attributs (porteurs d'information) correspondent aux paramètres du message.



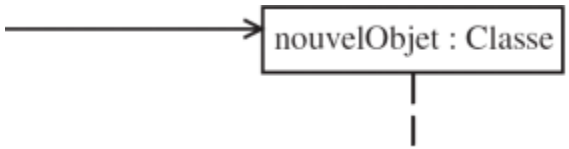
→ Événements et messages

- UML permet de séparer clairement l'envoi du message, sa réception, ainsi que le début de l'exécution de la réaction et sa fin

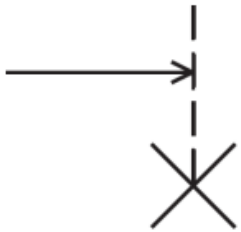


→ Messages de création et destruction de lignes de vie

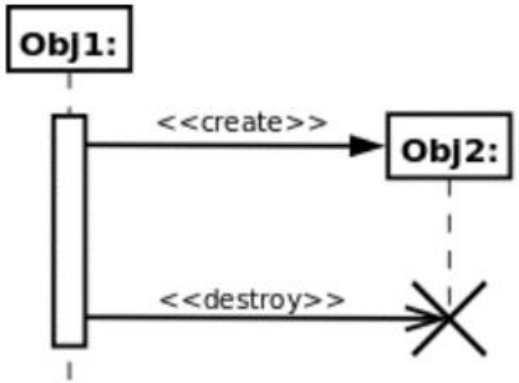
- La **création** d'un objet est matérialisée par une flèche qui pointe sur le sommet d'une ligne de vie.
- On peut aussi utiliser un message asynchrone ordinaire portant le nom ***create*** .



- La **destruction** d'un objet est matérialisée par une croix qui marque la n de la ligne de vie de l'objet.



Exemple:



→ Messages complets, perdus et trouvés

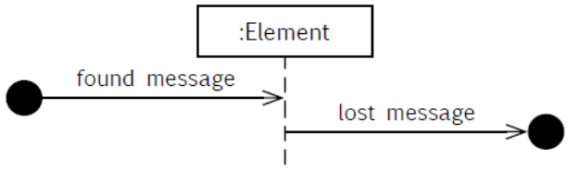
- Un **message complet** est tel que les événements d'envoi et de réception sont connus.
- Un message complet est représenté par une **flèche** partant ***d'une ligne de vie et arrivant à une autre ligne de vie.***
- Un **message perdu** est tel que l'événement d'envoi est connu, mais pas l'événement de réception.

→ La flèche part d'une ligne de vie mais arrive sur un cercle indépendant marquant la méconnaissance du destinataire. *Exemple* : broadcast.

- Un **message trouvé** est tel que l'événement de réception est connu, mais pas l'événement d'émission.

→ **Messages complets, perdus et trouvés**

❑ Syntaxe des messages



- **La syntaxe des messages est :**

nomSignalOuOperation (parametres)

- **La syntaxe des arguments est** :

nomParametre = valeurParametre

- **Pour un argument modifiable** :

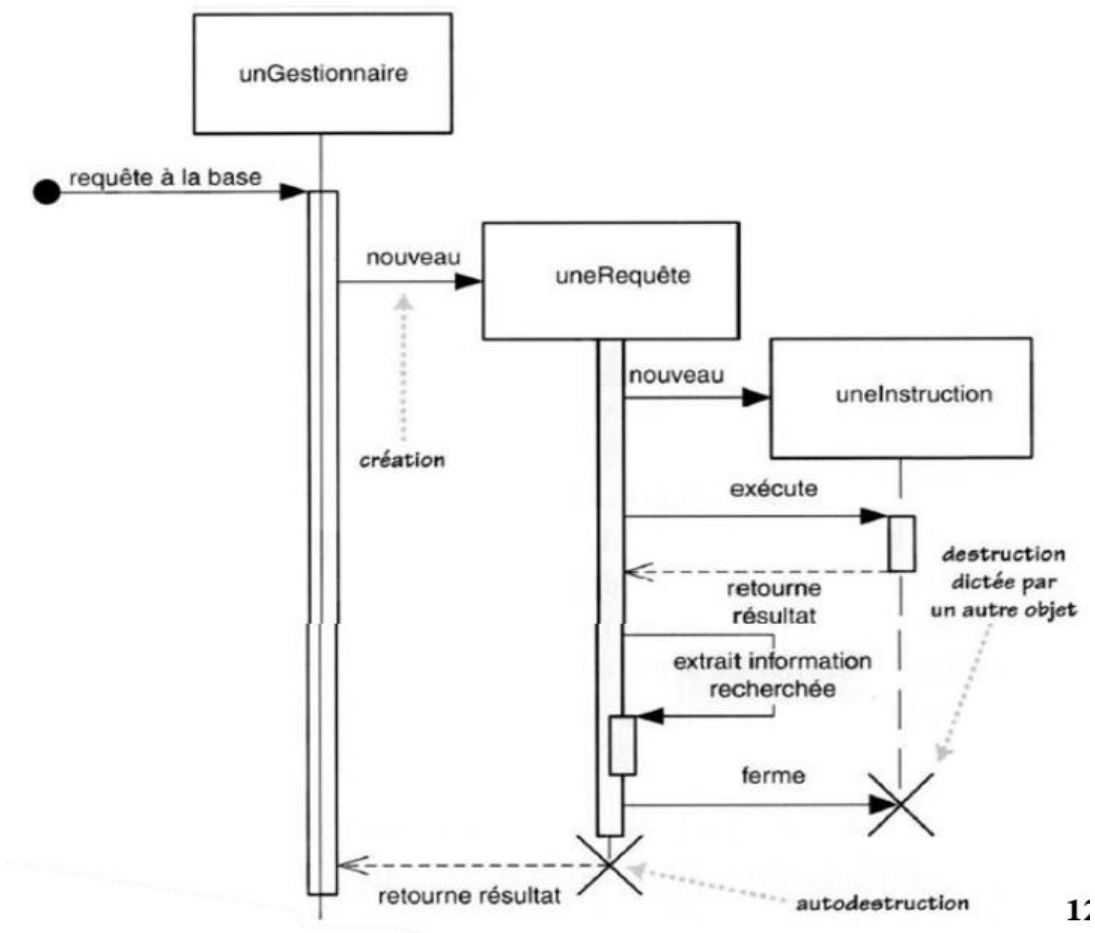
nomParametre : valeurParametre

Exemples :

appeler("NN", 54214110), afficher(x,y), initialiser(x=100), f(x:12)

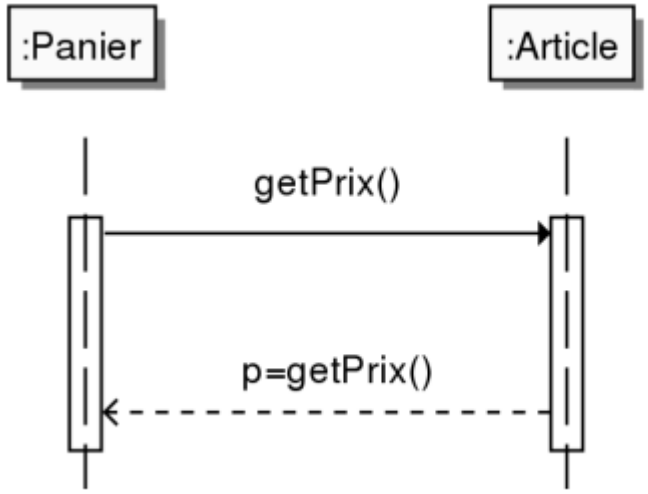
→ **Messages complets, perdus et trouvés**

Exemple:



→ Messages de retour

- 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.



p:float

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

→ Messages de retour

❑ Syntaxe des messages de retour

- **La syntaxe des messages de retour est :**

attributCible = nomOperation (params): valeurRetour

- **La syntaxe des paramètres est :**

nomParam = valeurParam

ou

nomParam : valeurParam

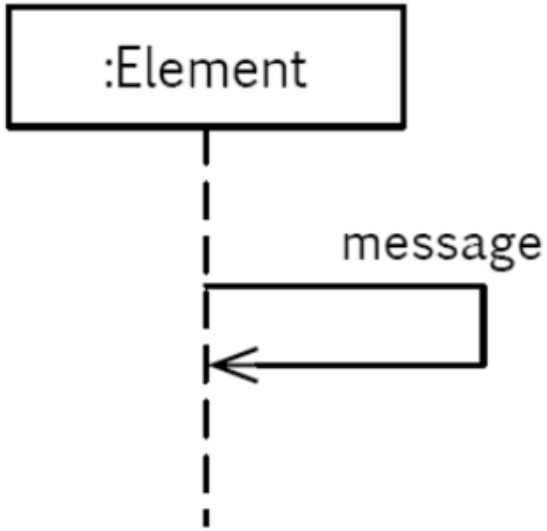
→ Les messages de retour sont représentés en pointillés.

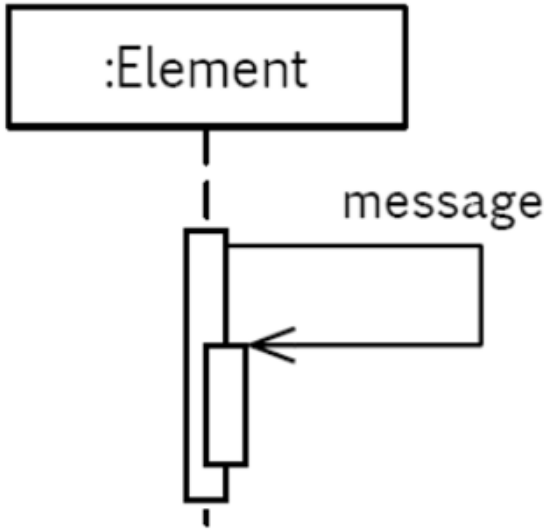
→ Messages réflexif

- Un objet peut s'envoyer des messages

  → appel à une autre méthode de l'objet

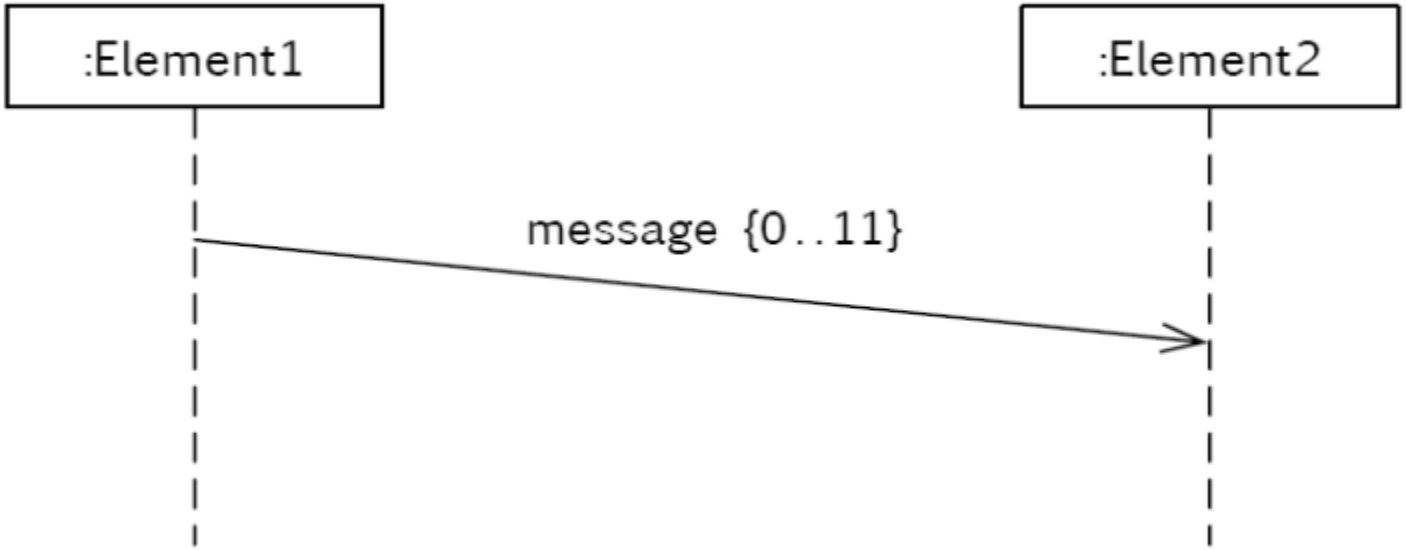
  → appel récursif





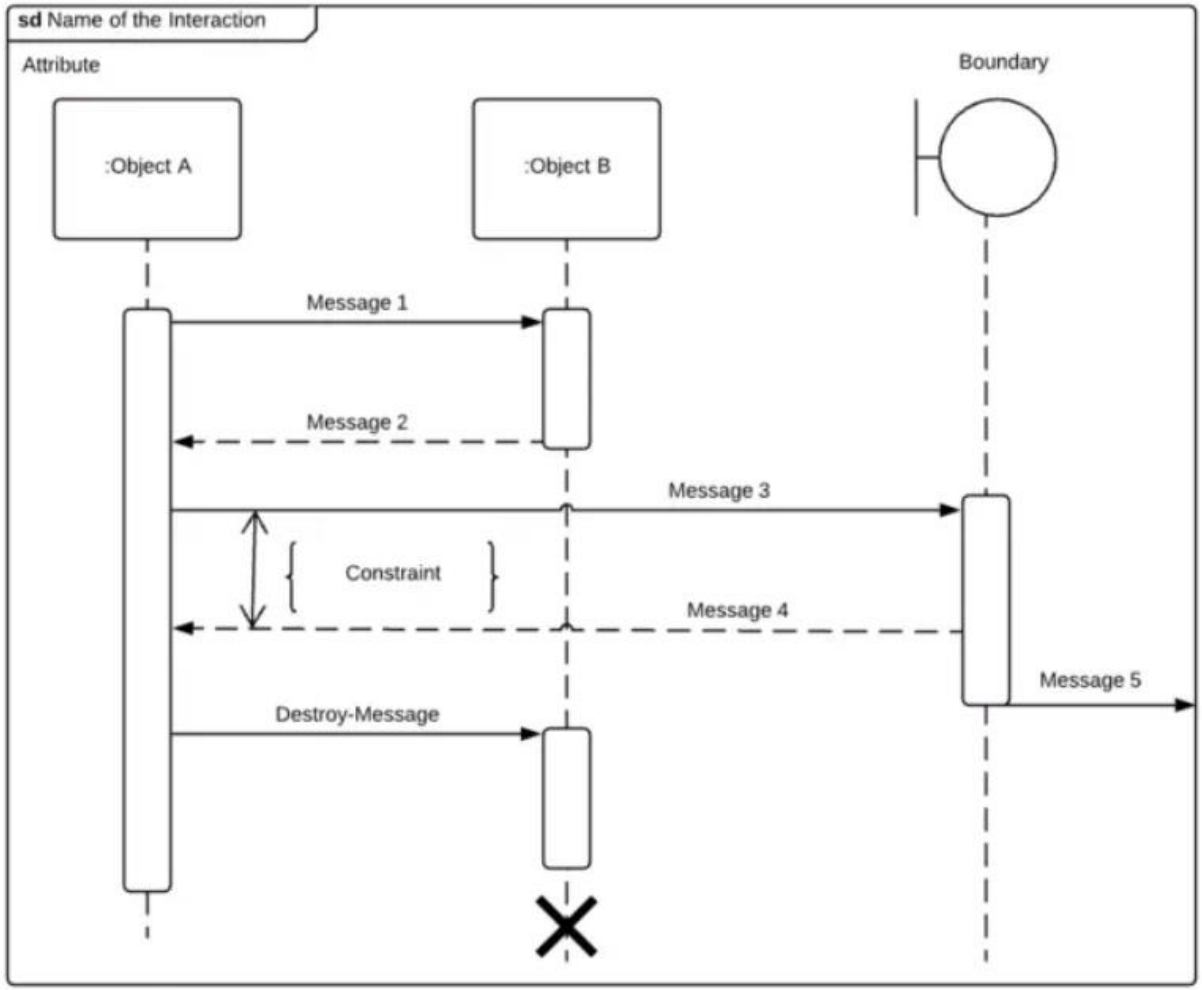
→ Durée et contraintes temporelles

- Représentation des délais de transmission :

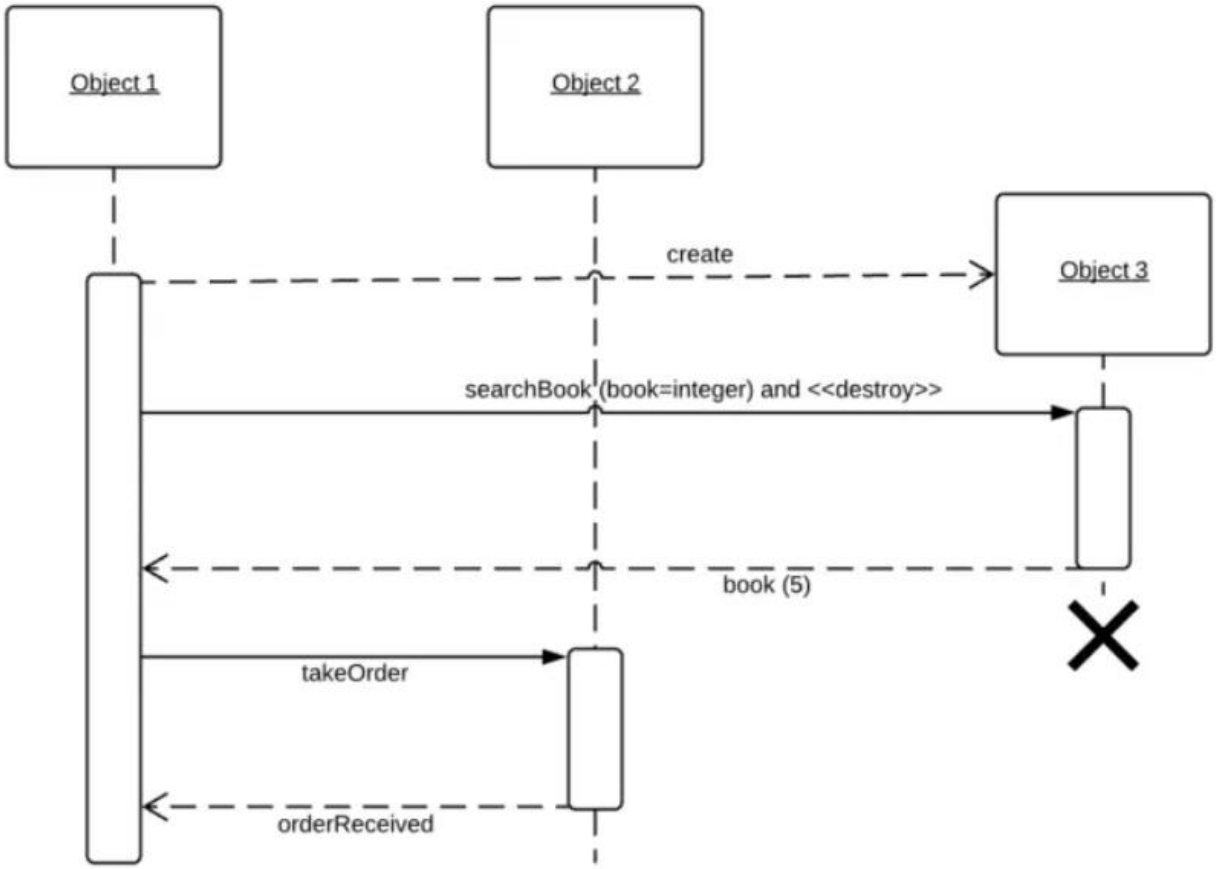


❑ Diagramme de séquences

- Les éléments de base d'un diagramme de séquence UML2 sont les **lignes de vie** et les **messages**.
- Si le diagramme doit être séparé optiquement des autres, un cadre avec l'étiquette « **sd** » dans l'en-tête est utilisé



Exemple de diagramme de séquences



Les fragments combiné

- Un **fragment combiné** permet de décomposer une interaction complexe en fragments suffisamment simples pour être compris.
  - Recombiner les fragments restitue la complexité.
  - Syntaxe complète avec UML 2 : représentation complète de processus avec un langage simple (ex : processus parallèles).
- Un fragment combiné se représente de la même façon qu'une **interaction**. Il est représenté un rectangle dont le coin supérieur gauche contient un pentagone.
- Dans le pentagone figure le type de la combinaison (appelé opérateur d'interaction).

❑ Type d'opérateurs d'interaction

- **Opérateurs de branchement** (choix et boucles) :

  → *alternative*, *option*, *break* et *loop*

- **Opérateurs contrôlant l'envoi en parallèle de messages :**

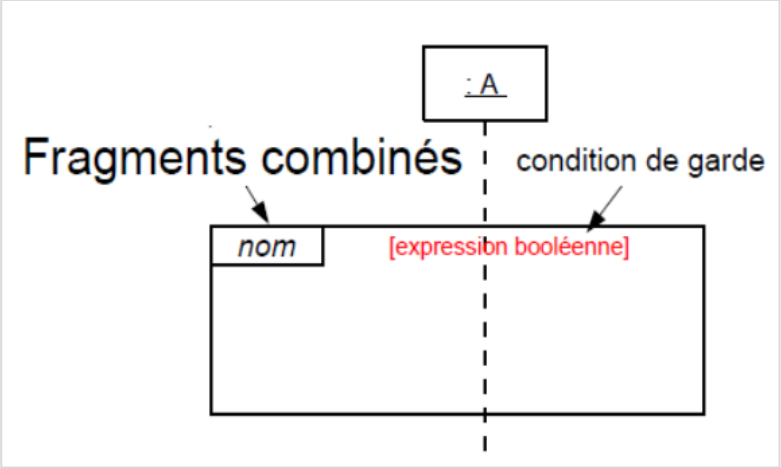
  → *parallel* et *critical region* ;

- **Opérateurs contrôlant l'envoi de messages :**

  → *ignore*, *consider*, *assertion* et *negative*

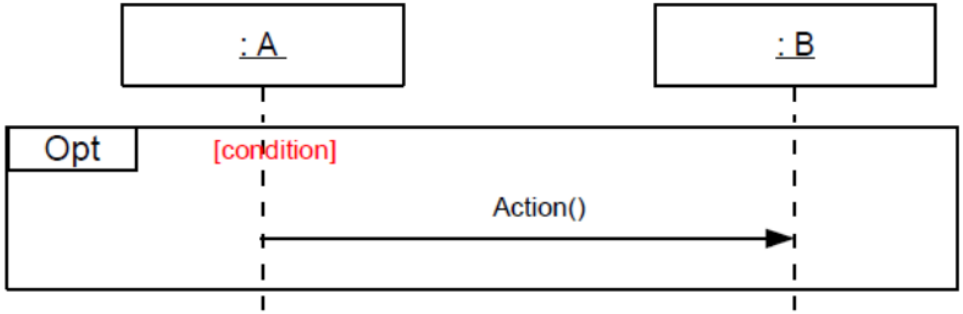
- **Opérateurs fixant l'ordre d'envoi des messages**

  → *weak sequencing* et *strict sequencing*.

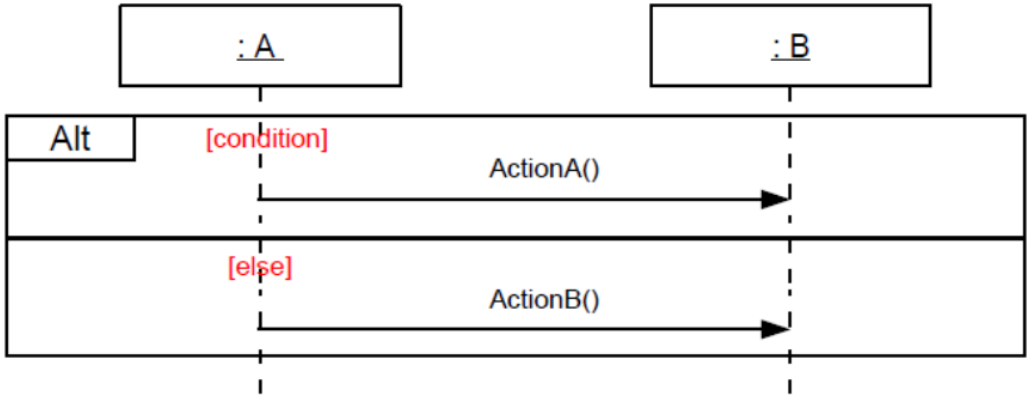


❑ Opérateurs conditionnels

- **Opt** : Fragment parcouru si une condition est vérifiée

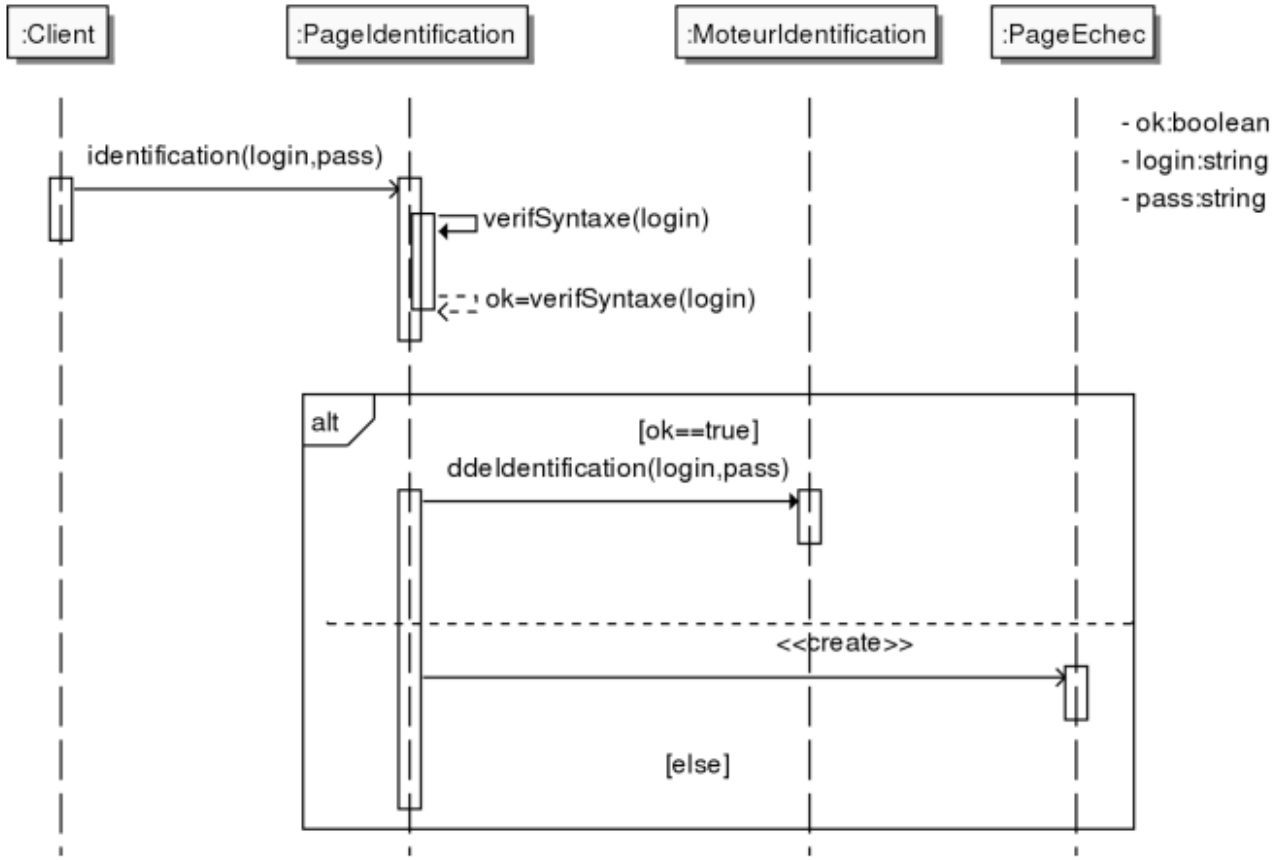


- **Alt** : Equivalent à la structure de contrôle "**si** .. **alors** .. **sinon**"



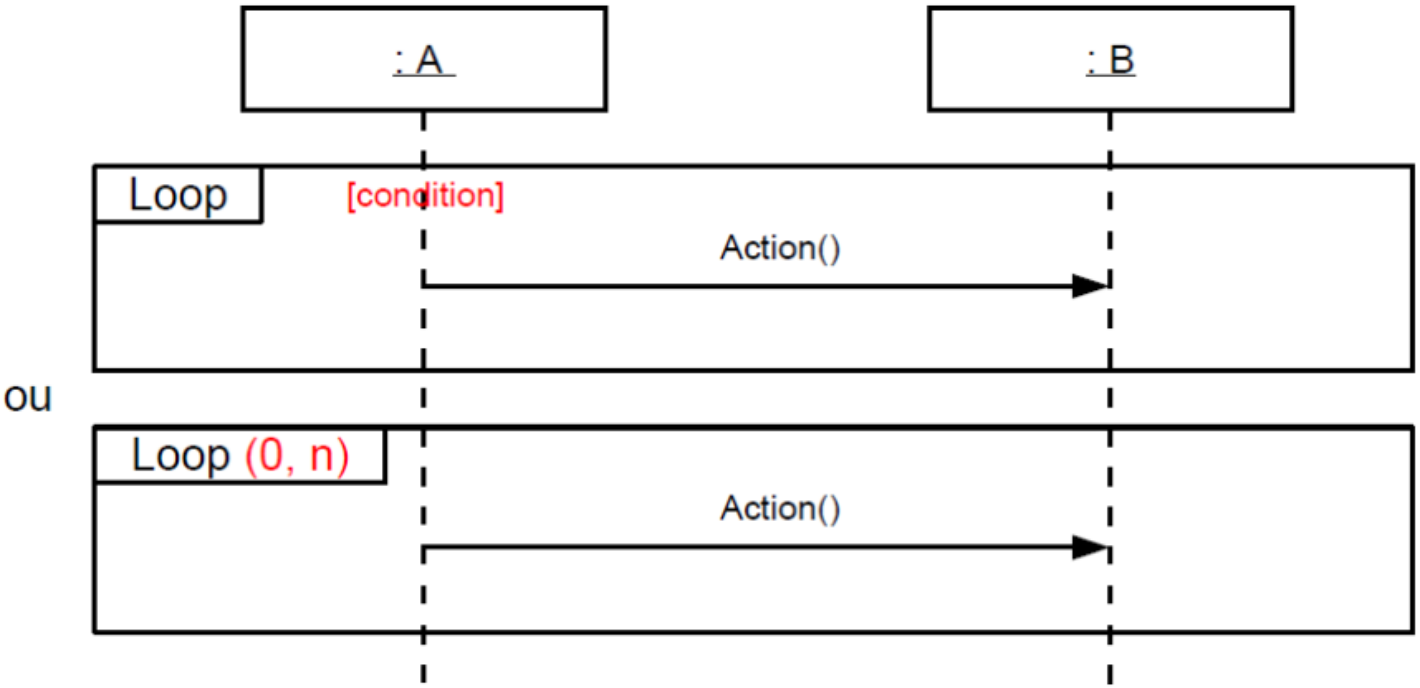
Les fragments combiné

Exemple de fragment avec l'opérateur conditionnel



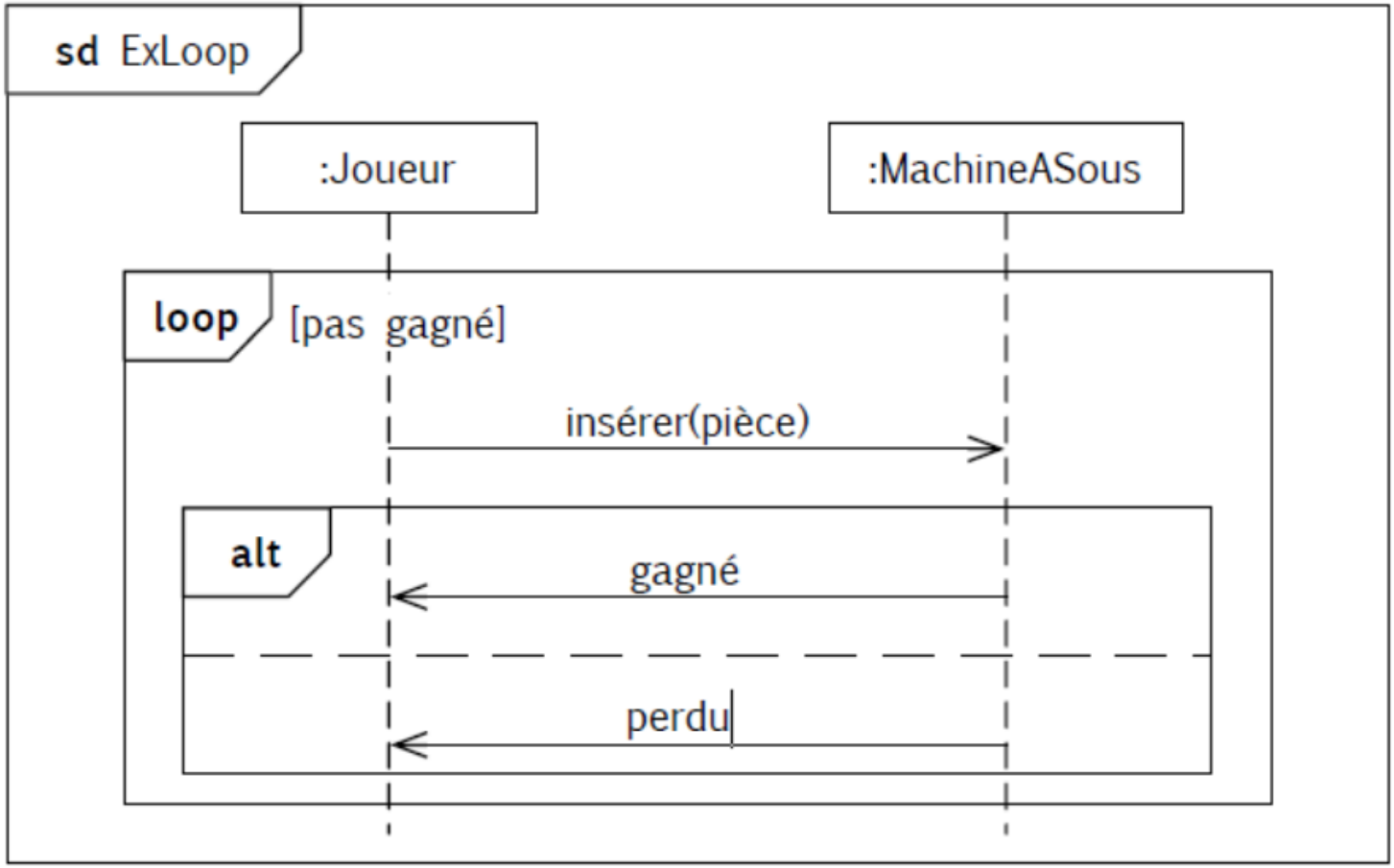
❑ Opérateur de boucle

- **Loop** : Répétition du fragment tant que la condition est vérifiée



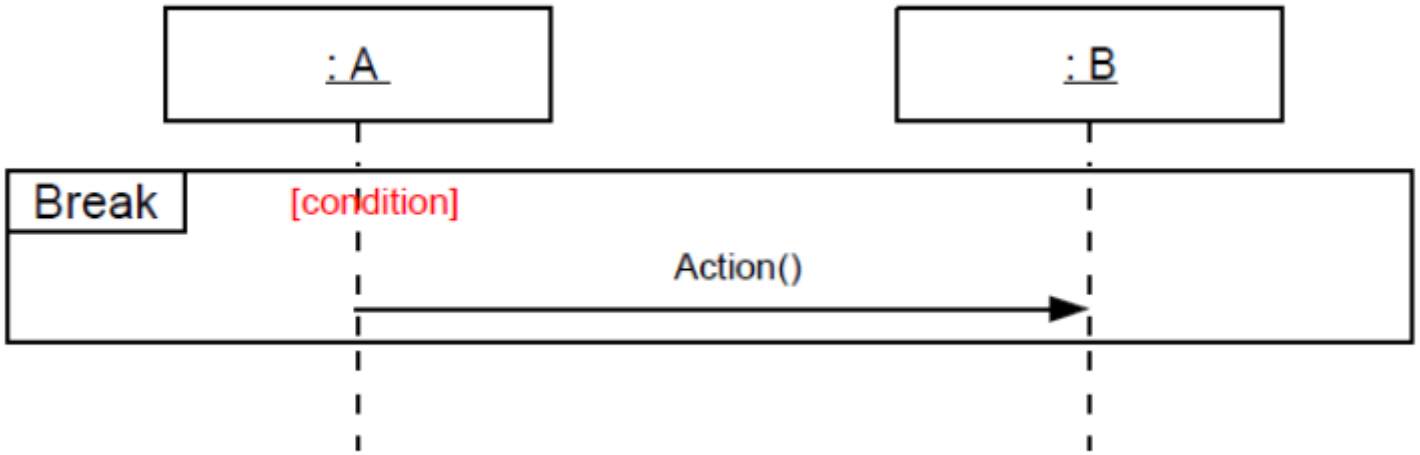
Les fragments combiné

Exemple



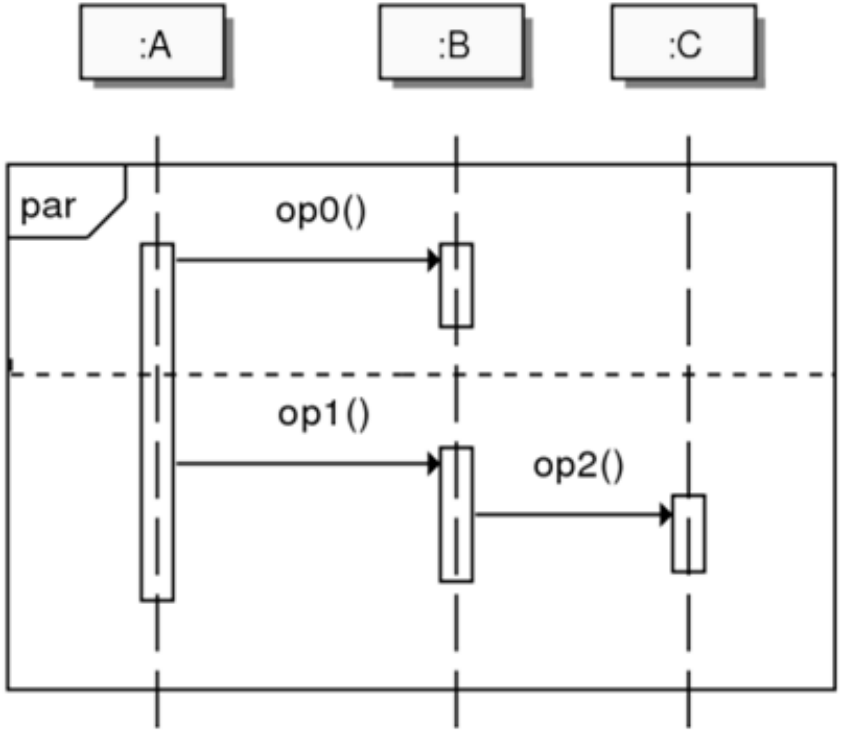
❑ Opérateur *break*

- **Break** : Fragment exécuté et met fin au fragment englobant



❑ Opérateur parallèle

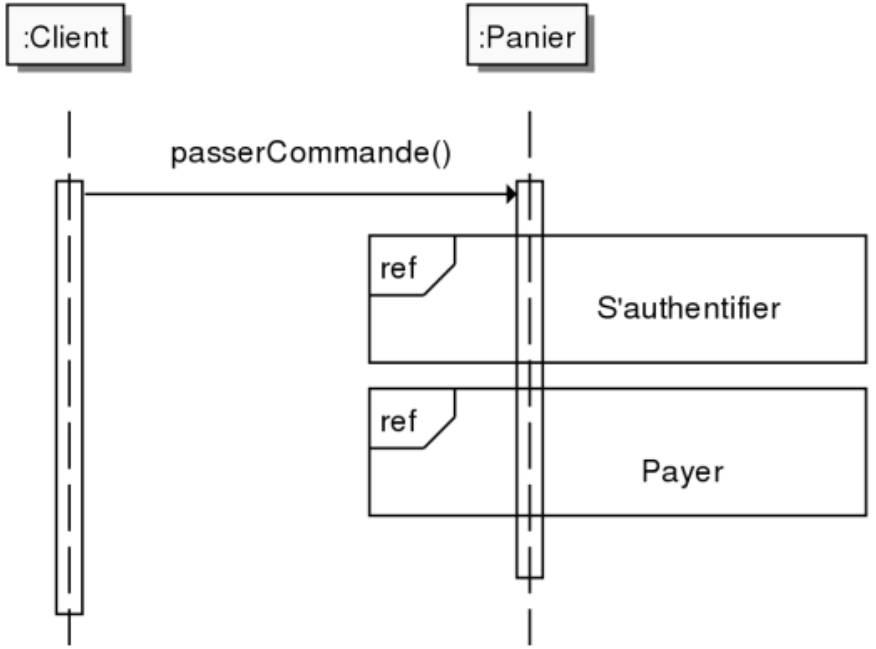
- L'opérateur par permet d'envoyer des messages en parallèle.
- Ce qui se passe de part et d'autre de la ligne pointillée est indépendant.



❑ Réutilisation d'une interaction

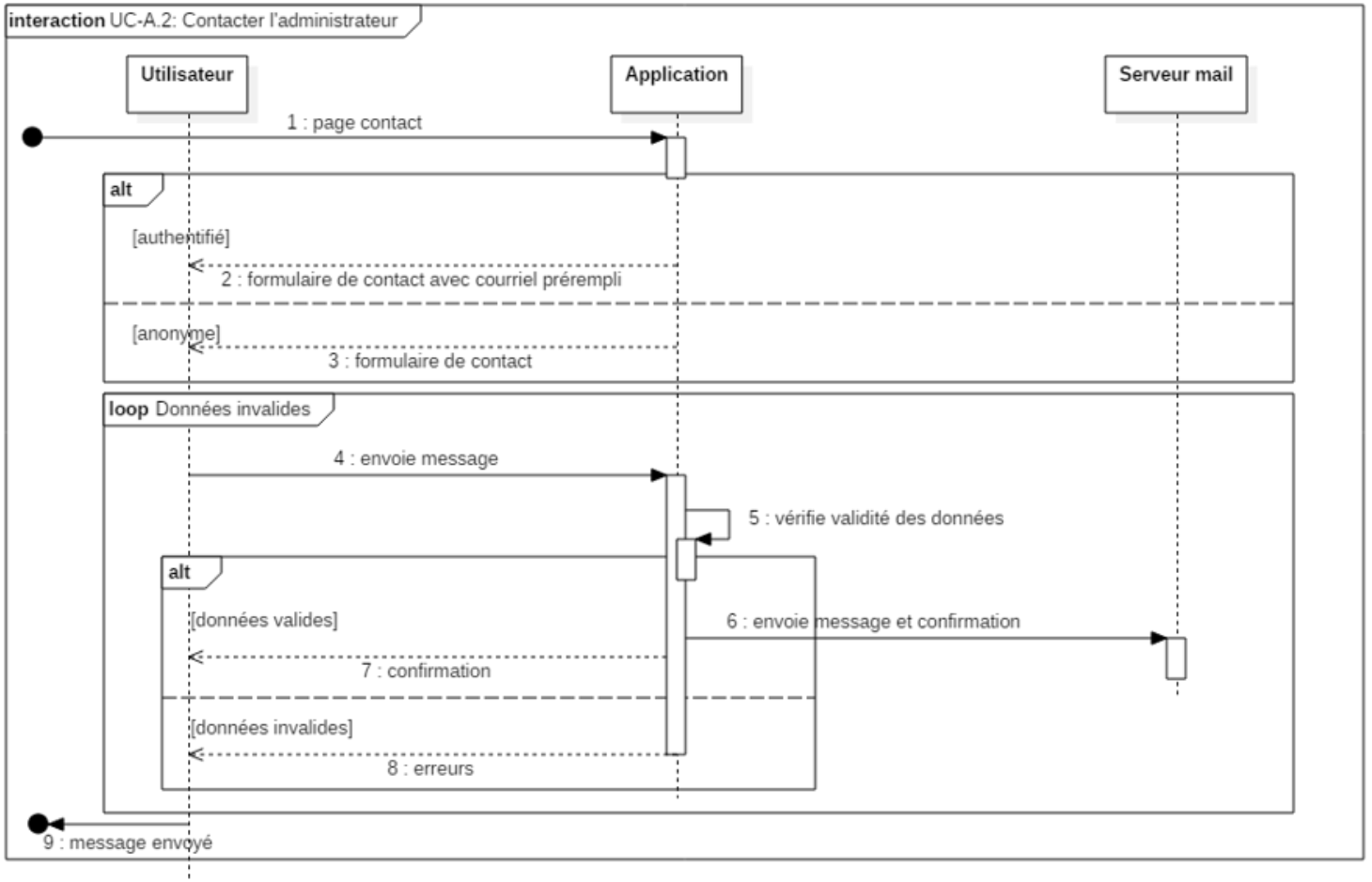
- Réutiliser une interaction consiste à placer un fragment portant la référence ***ref*** là où l'interaction est utile.

- On spécifie le nom de l'interaction dans le fragment.



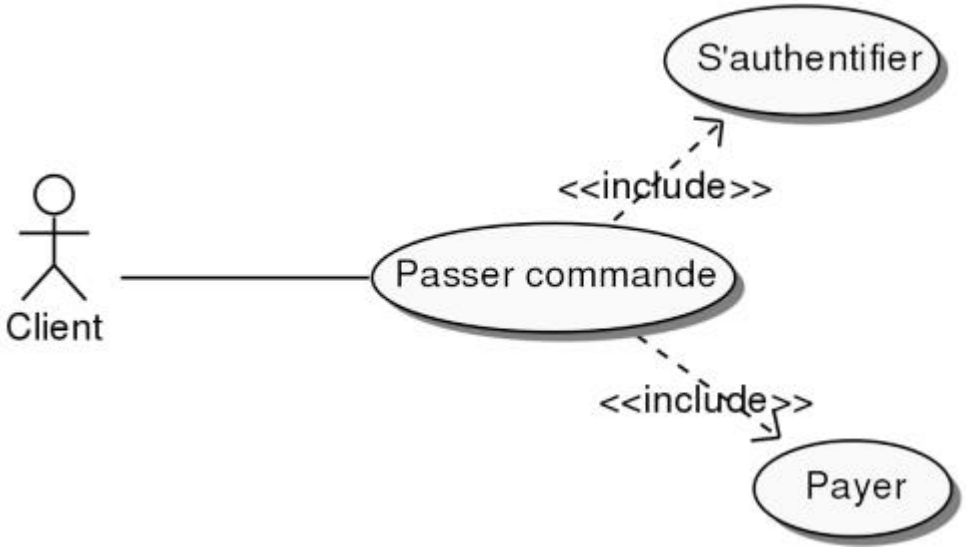
Les fragments combiné

Exemple



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

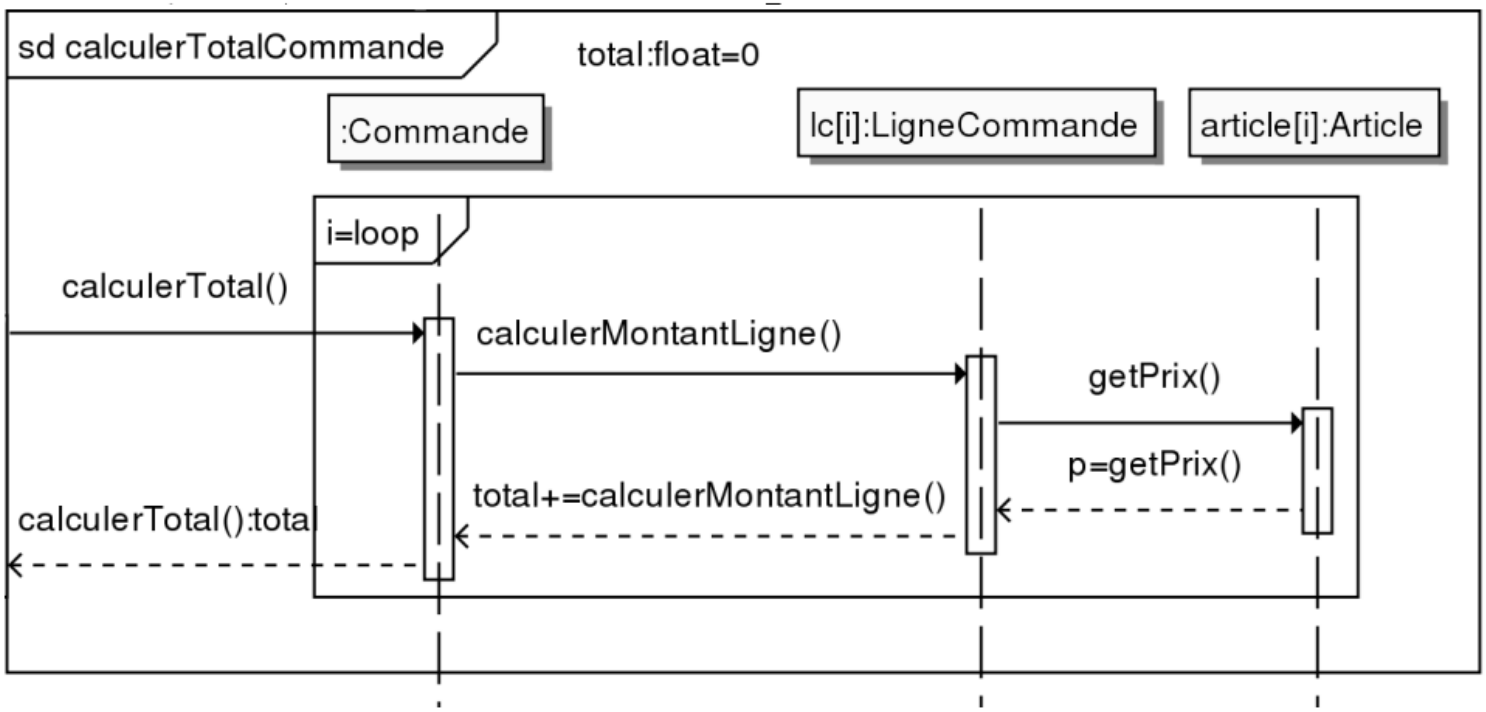
- Chaque cas d'utilisation donne lieu à un diagramme de séquences



- Les inclusions et les extensions sont des cas typiques d'utilisation de la réutilisation par référencement

→ Utilisation d'un DS pour spécifier une méthode

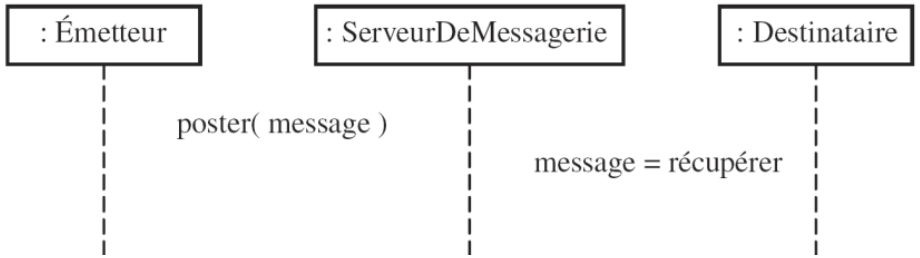
- Une interaction peut être identifiée par un fragment **sd** (pour « sequence diagram) précisant son nom
- Un message peut partir du bord de l'interaction, spécifiant le comportement du système après réception du message, quel que soit l'expéditeur



Exercice 1

1. Quand un courrier électronique est envoyé par l'émetteur, celui-ci ne veut pas attendre que destinataire l'ait reçu et il n'y a pas d'intermédiaire. Peut-on utiliser un message synchrone ? Complétez la figure ci dessous par des flèches représentant des messages.

2. Un serveur de messagerie sert d'intermédiaire entre l'émetteur et le récepteur d'un email. Le serveur est toujours en fonction. Est-ce qu'on peut utiliser des messages synchrones pour l'envoi et la récupération de emails ? Complétez la figure ci-dessous par des flèches représentant des messages.



Exercice 2

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 MAD
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.

Donner le diagramme de séquences associé à cette description textuelle.

Exercice 2 – Correction