

Diagrammes de séquences

Modélisation de la dynamique

- Modélisation des interactions entre objets de plusieurs classes
- Montrer la **succession** des enchaînements
- Montrer la **chronologie** des messages envoyés et reçus
- Montrer à quel moment un acteur est sollicité
- Diagrammes d'interaction
 - Diagramme de séquence
 - Diagramme de collaboration
- Modélisation du comportement des objets d'une classe

Interactions

- Interaction :
 - Un ensemble de communications entre instances
 - Appel de méthodes
 - Création /Destruction
- Partiellement ordonné dans le temps

Diagrammes d'interaction

- Montre les interactions entre instances dans un modèle
 - graphe d'instances et de stimuli
 - Création et destruction d'instances
- Types
 - Diagramme de séquence (point de vue temporel)
 - Diagramme de collaboration (point de vue structurel)

Diagrammes 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 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'échangeant des messages.
 - Les acteurs **interagissent** avec le système au moyen **d'IHM** (Interfaces Homme-Machine).

Objectifs

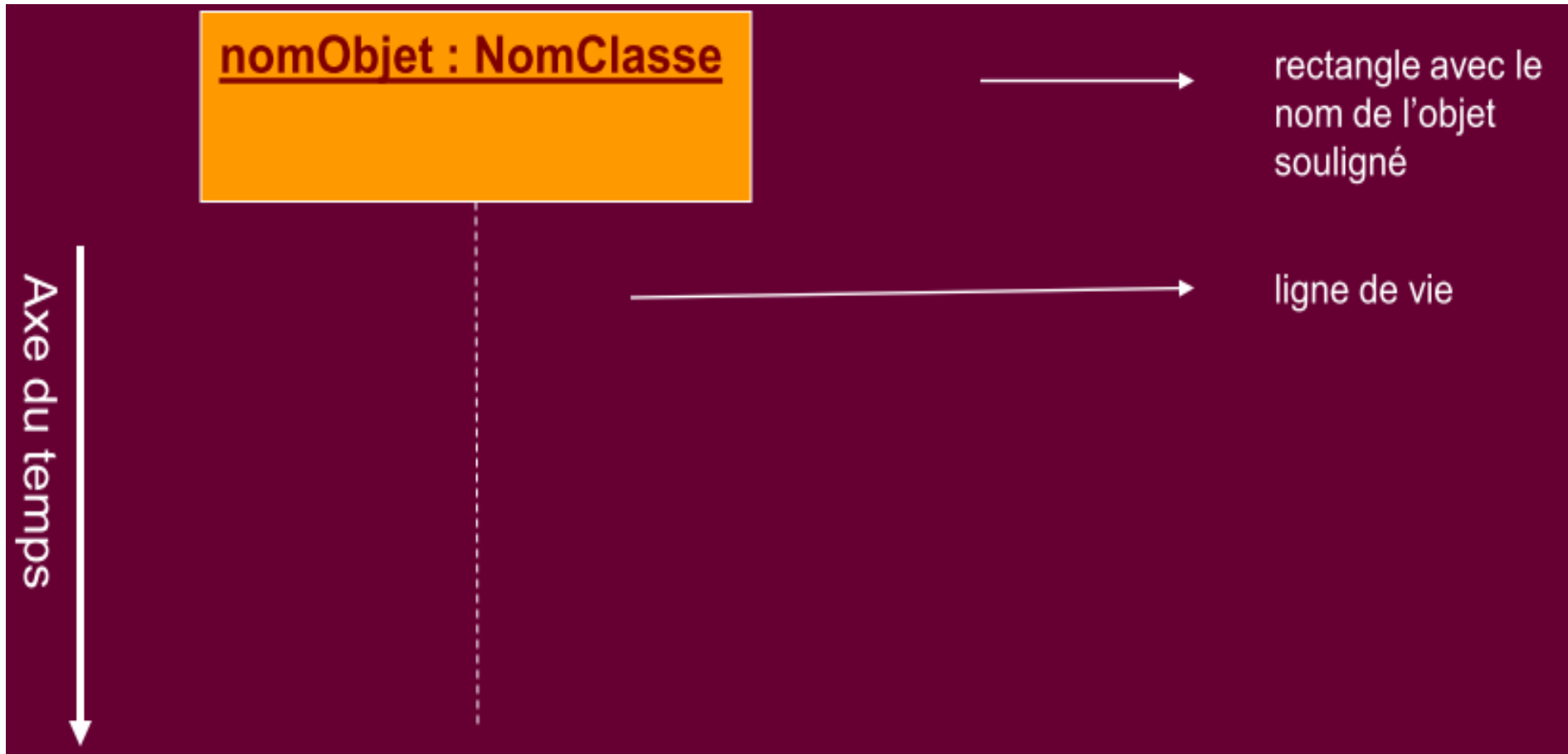
- Les diagrammes de séquences permettent de représenter des collaborations entre objets selon un **point de vue temporel**, on y met l'accent sur la **chronologie** des envois de messages
- Contrairement au diagramme de collaboration, **on n'y décrit pas le contexte ou l'état des objets**, la représentation **se concentre sur l'expression des interactions**
- Les diagrammes de séquences **peuvent servir à illustrer un cas d'utilisation**
- Insistance sur la **chronologie** des envois des messages
- Mise en évidence du fonctionnement du programme avec visualisation du temps

Objectifs

- L'ordre d'envoi d'un message est déterminé par sa position sur **l'axe vertical** du diagramme ; **le temps s'écoule** "de haut en bas" de cet axe
- La disposition des objets sur l'axe du temps n'a pas de conséquences pour la sémantique du diagramme

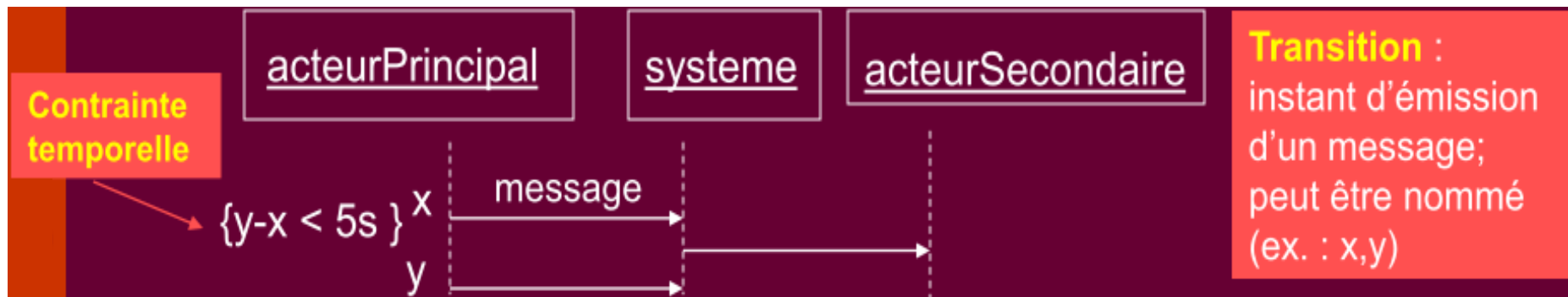
Représentation

- Représentation des objets :



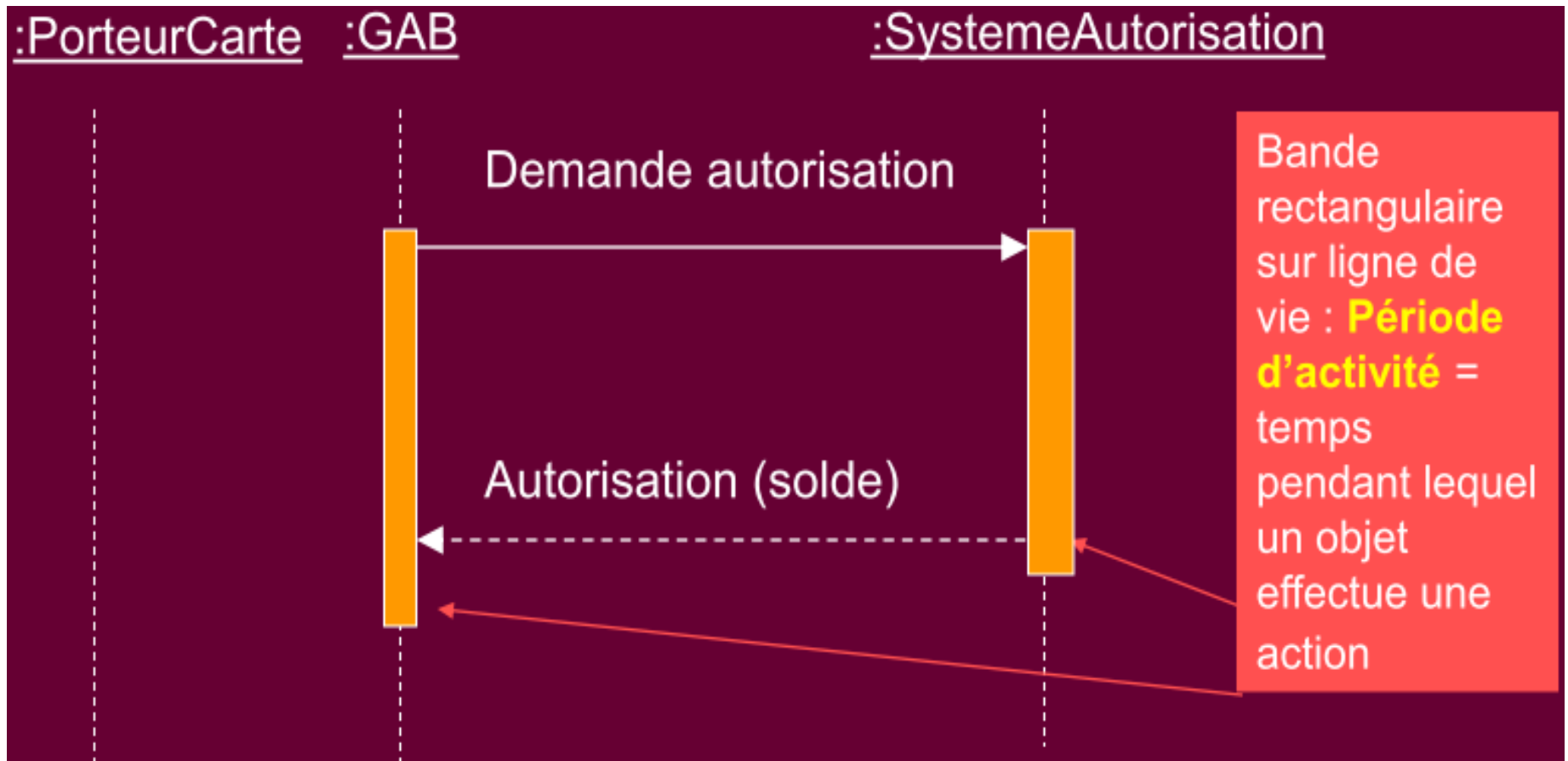
Représentation

- Acteur principal : à gauche
- Objet représentant le **système** en boîte noire au milieu
- Acteurs secondaires : à droite
- Echange de **message** :
 - flèche horizontale, de l'émetteur vers le destinataire



Représentation

- Flèche de retour en pointillé



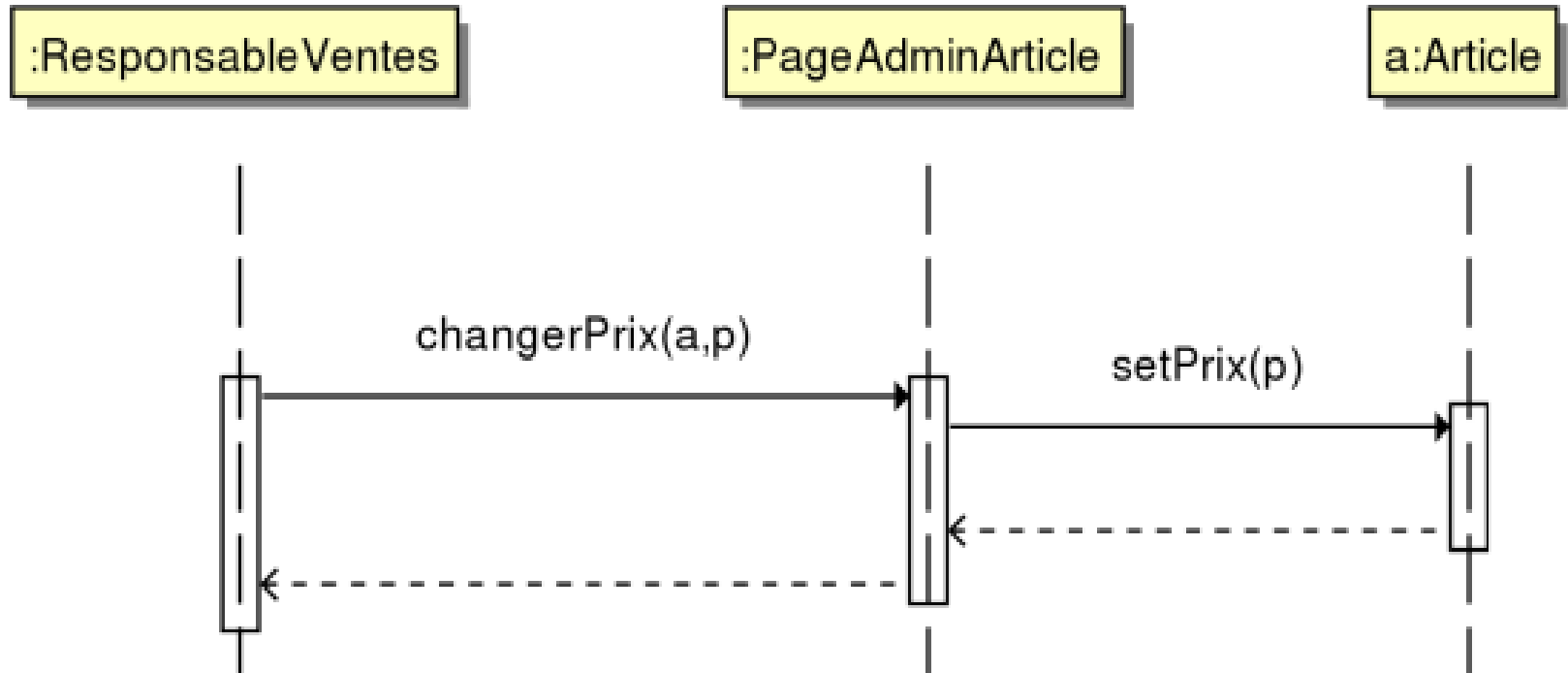
Exemple d'interaction

- Cas d'utilisation :

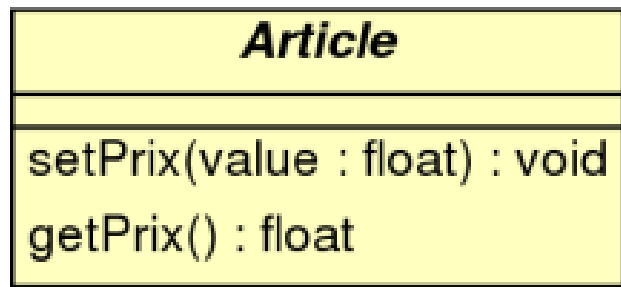


Exemple d'interaction

- Diagramme de séquences correspondant :



- Opérations nécessaires dans le diagramme de classes :

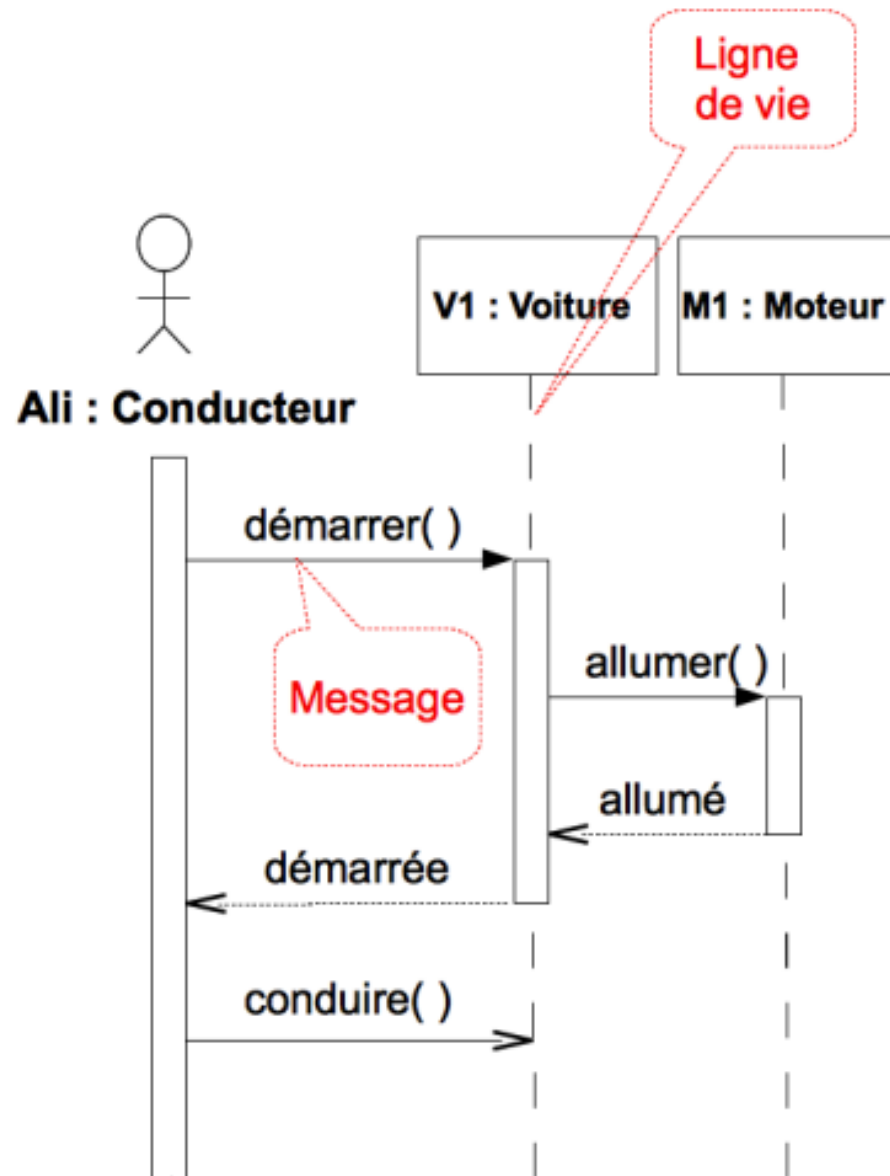


Diagrammes de séquences

- Présente les **interactions chronologiques** entre les objets
- Focalise sur les:
 - **objets** (les classes)
 - les **acteurs** (en partie)
 - échange de **messages**
 - **scénarii** d'exécution
 - **ligne de vie** des objets

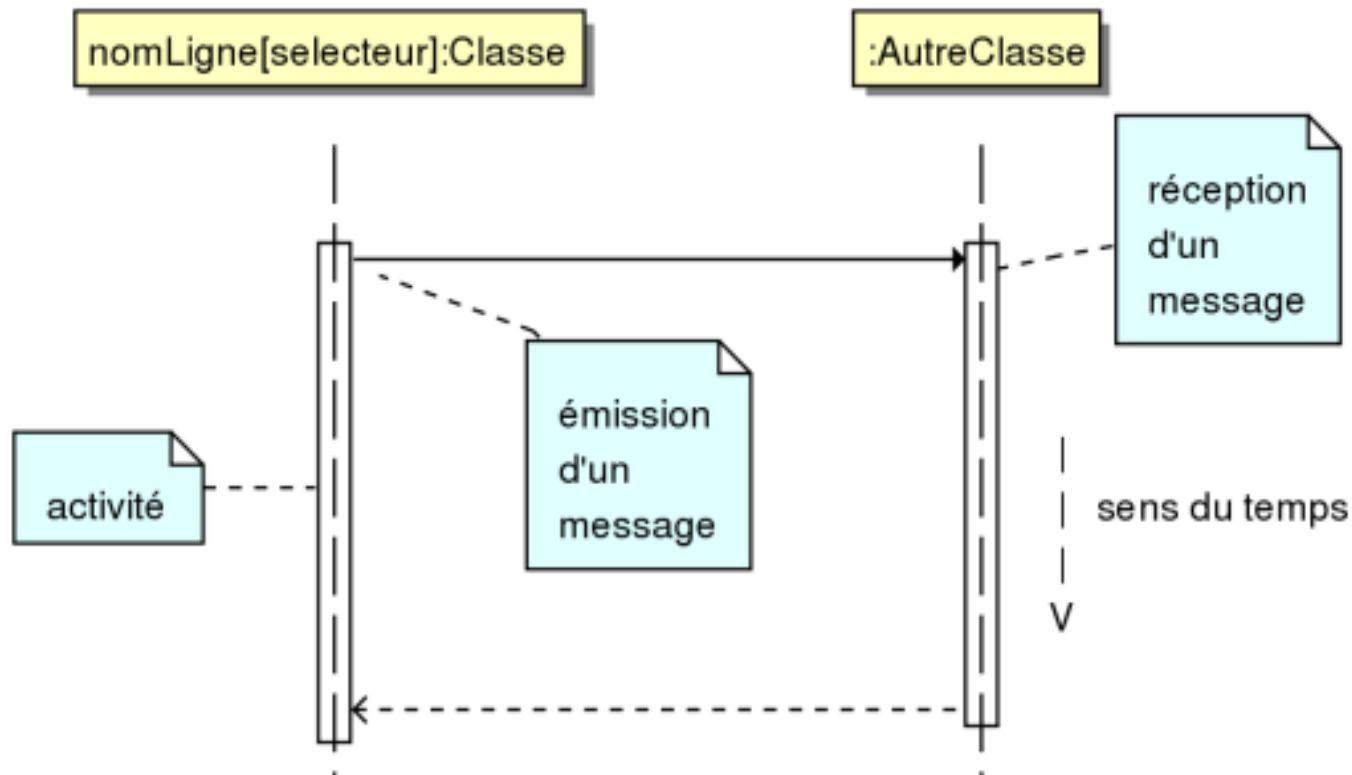
Contenu

- **Lignes de vie**
 - Rectangle + ligne pointillée
 - Etiquette [objet]:[classe]
- **Messages**
 - Communication entre les lignes de vie
 - Peuvent être :
 - Envoi de signal
 - Invocation d'une opération
 - Création ou destruction d'une instance



Ligne de vie



- Une ligne de vie représente un participant à une interaction (objet ou acteur).
 - *nomLigneDeVie [selecteur]: nomClasseOuActeur*
- Dans le cas d'une collection de participants, un sélecteur permet de choisir un objet parmi n (par exemple objets[2])



Messages

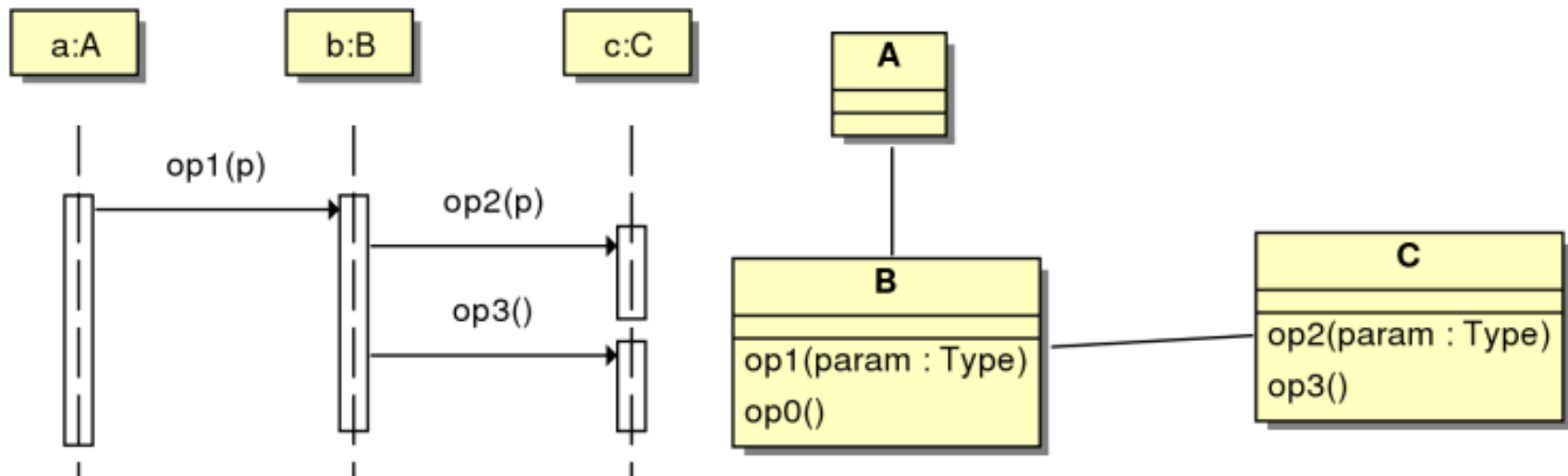
- Les principales informations contenues dans un diagramme de séquence sont les **messages** échangés entre les lignes de vie, présentés dans un ordre **chronologique**.
 - Un message définit une communication particulière entre des lignes de vie (objets ou acteurs).
 - Plusieurs types de messages existent, 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.

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 (voir stéréotype de classe « signal »).

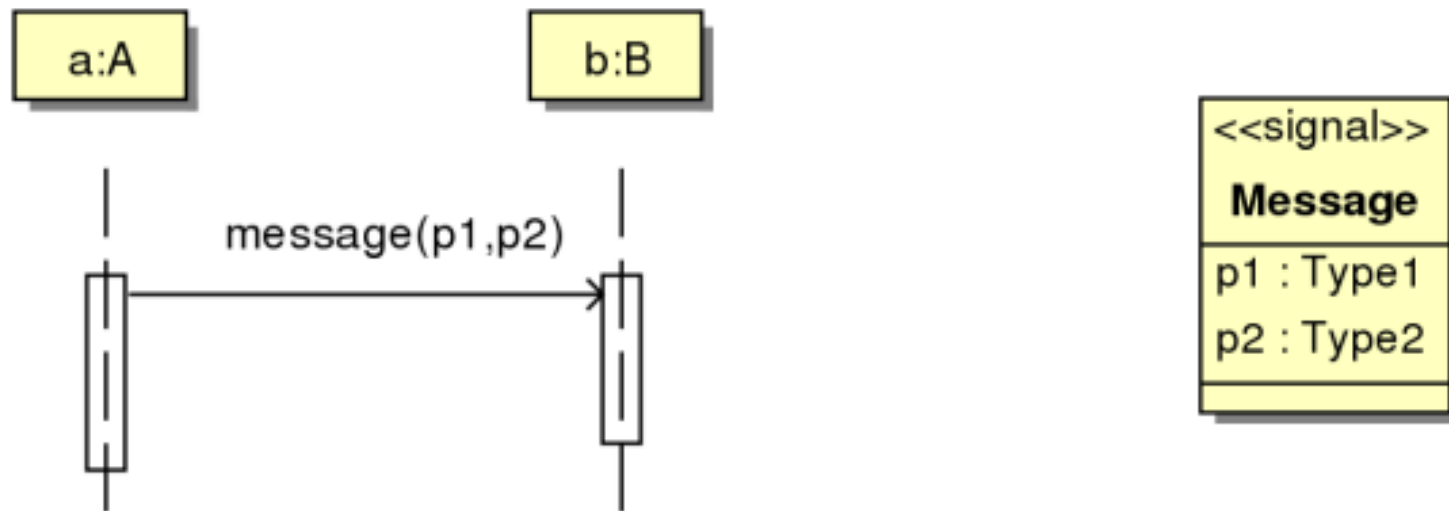
Correspondance messages / opérations

- 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.

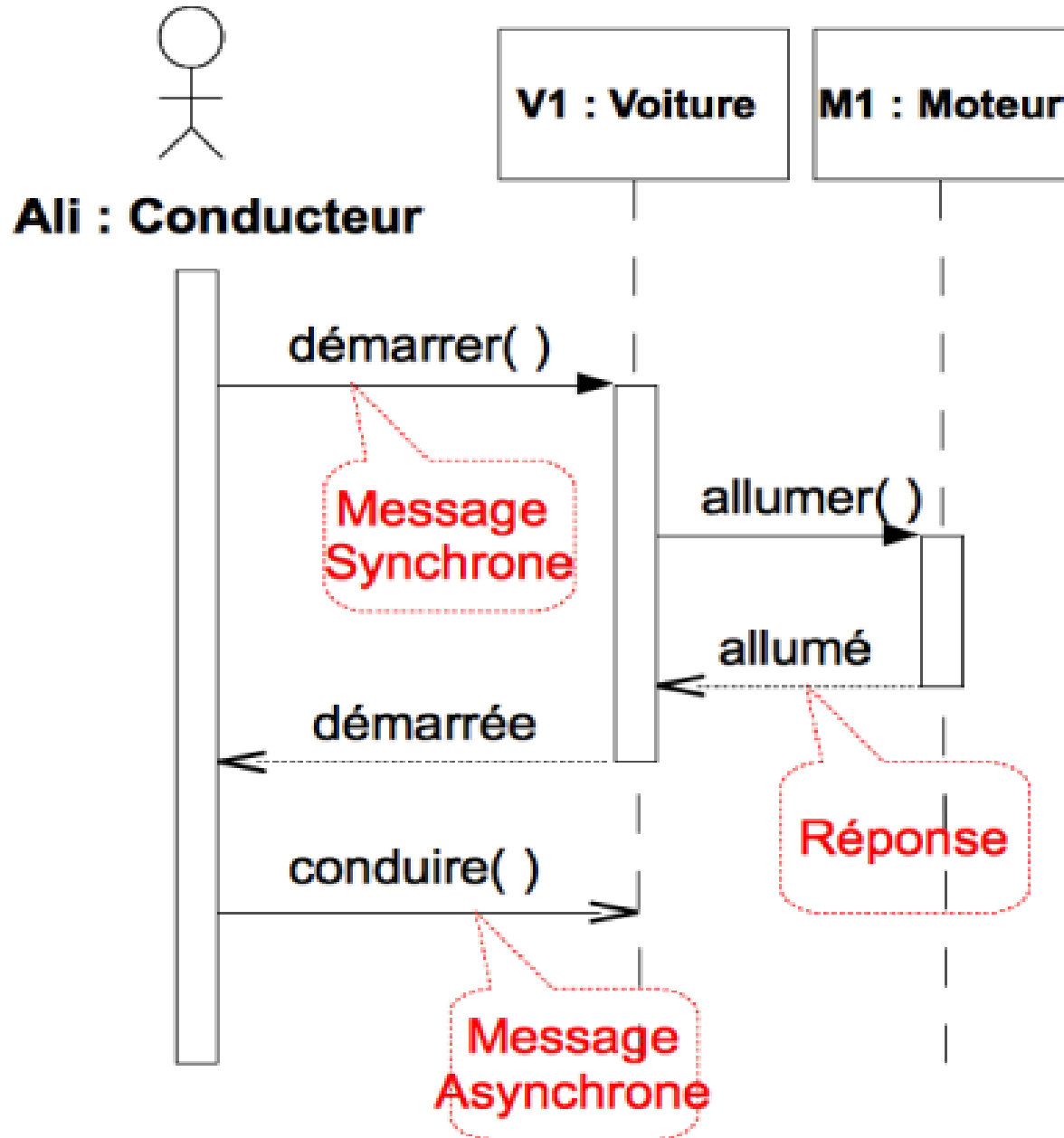


Correspondance messages / signaux

- 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.

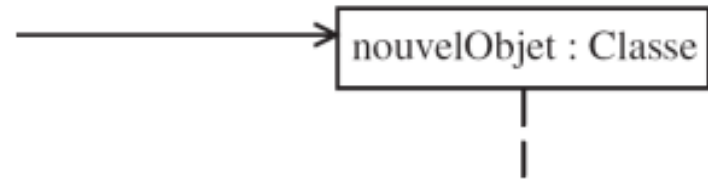


Messages : exemple

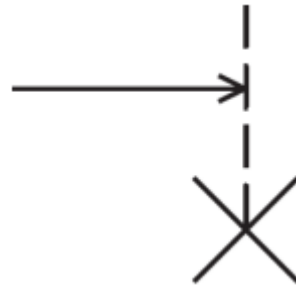


Création et destruction de lignes de vie

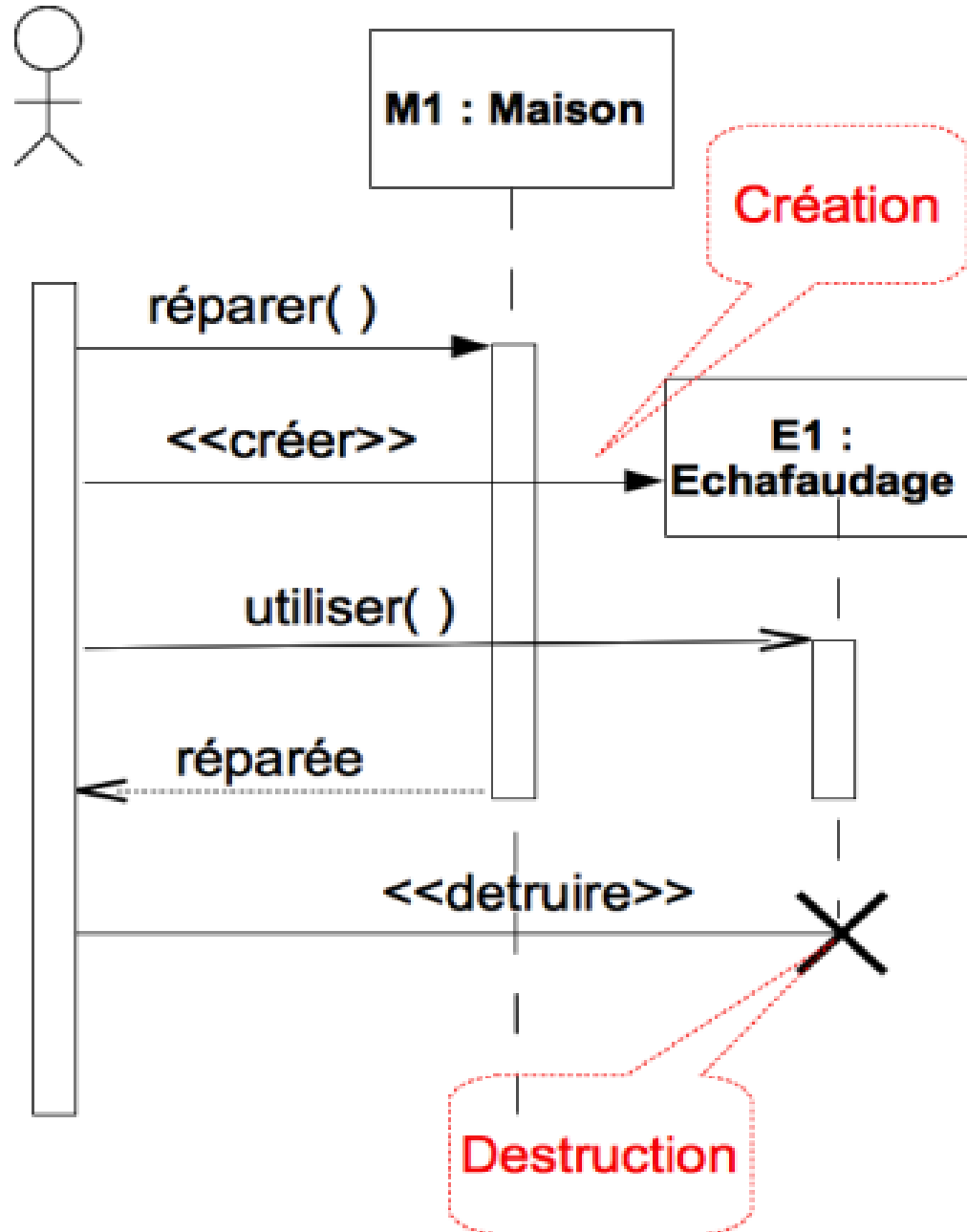
- La **création** d'un objet est matérialisée par une flèche qui pointe sur le sommet d'une ligne de vie.
 - On peut aussi utiliser un message asynchrone ordinaire portant le nom « create ».



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

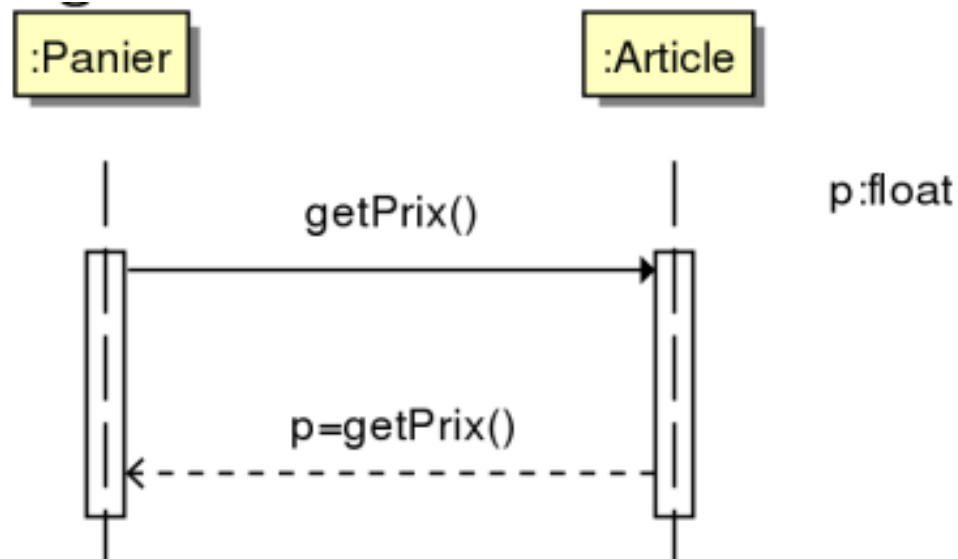


Exemple



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.



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

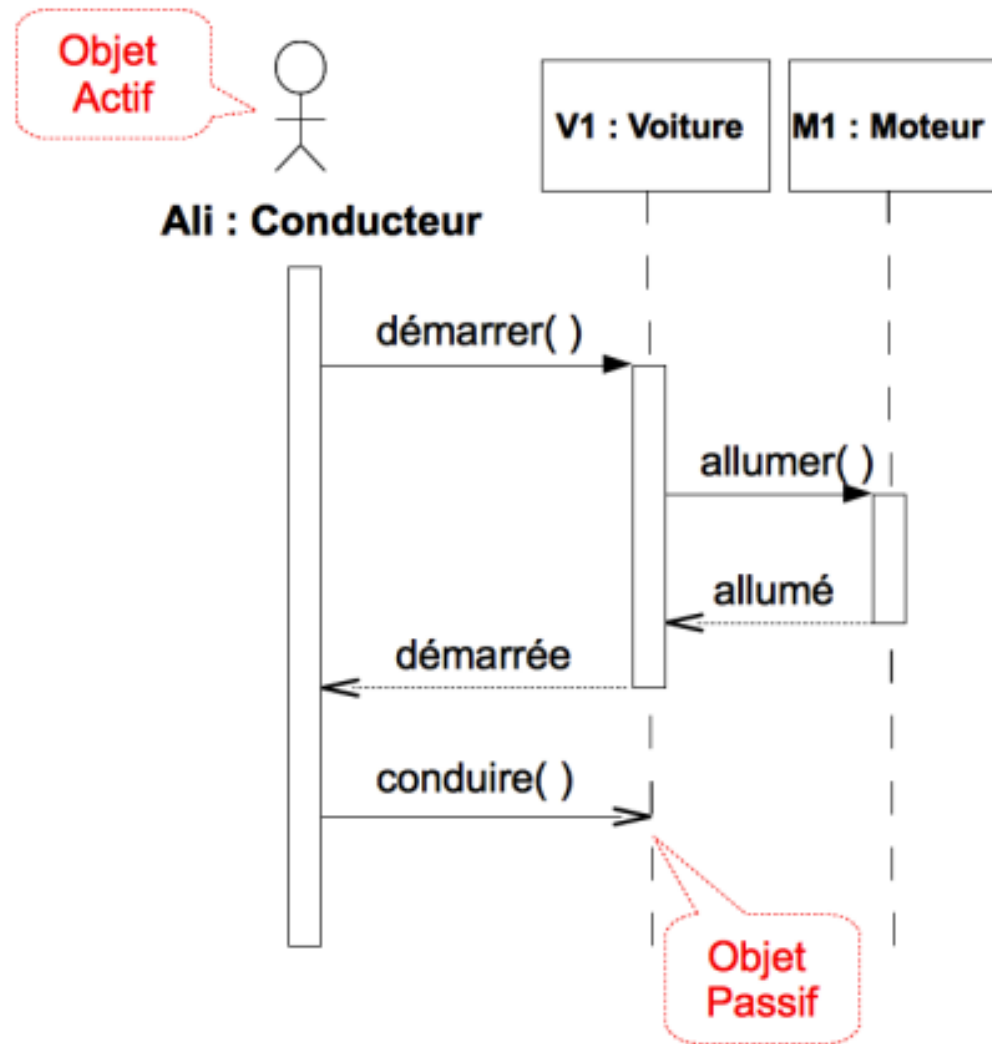
Objets Actif et Passifs

- **Objet actif**

- Initie et contrôle le flux d'activités
- **Représentation** : un rectangle à la place de la ligne de vie verticale

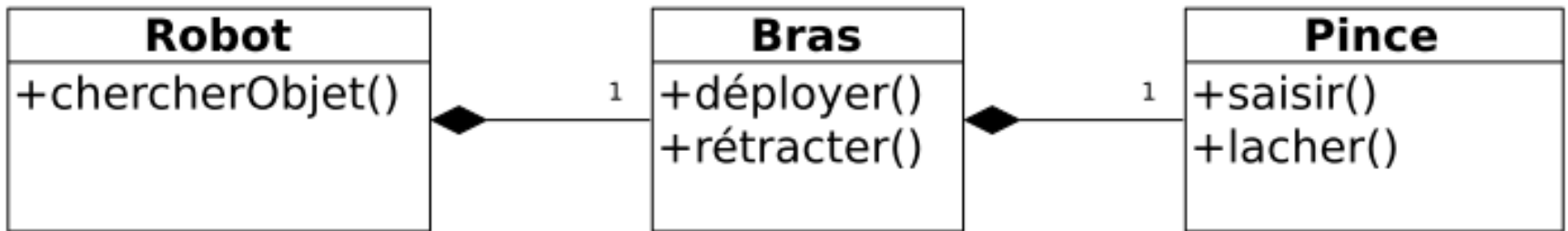
- **Objet passif**

- A besoin d'un flux d'activités pour pouvoir exécuter une méthode
- À l'exécution d'une méthode, un rectangle blanc est placé sur la ligne de vie en pointillés



Exercice

- Soit un qbras articulé (ou petit robot) capable de déployer ou de rétracter son bras et d'ouvrir ou de fermer sa pince pour aller chercher des objets lorsque l'ordre lui en est donné. Le diagramme de classe ci-dessous “modélise” ce robot.

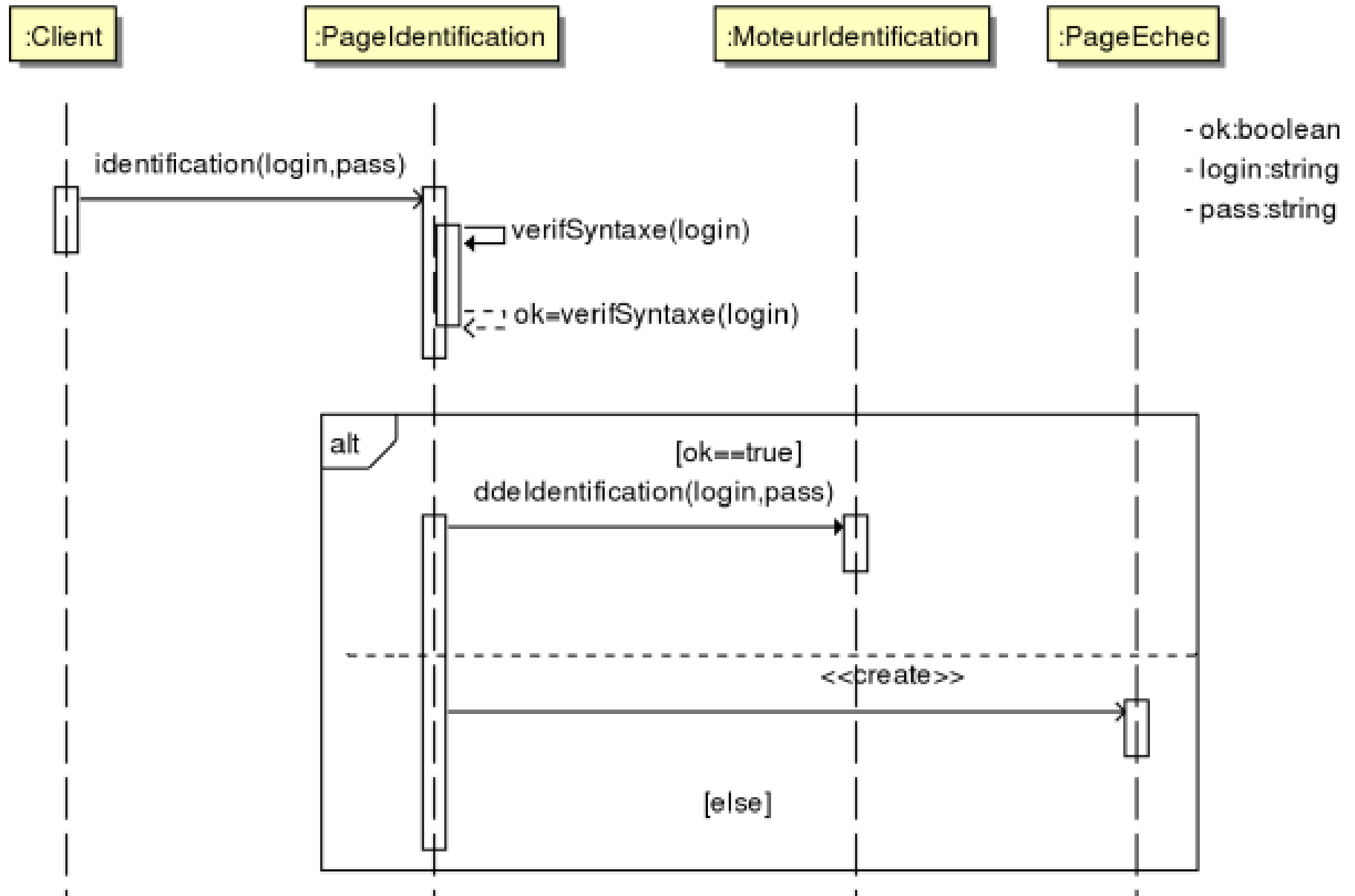


- Question: *Illustrer par un diagramme de séquence le scénario suivant :*
 - *L'ordre est envoyé au robot d'aller chercher un objet*
 - *le robot déploie son bras*
 - *le robot saisit l'objet avec sa pince*
 - *le robot rétracte son bras*
 - *le robot lâche l'objet*

Fragment combiné

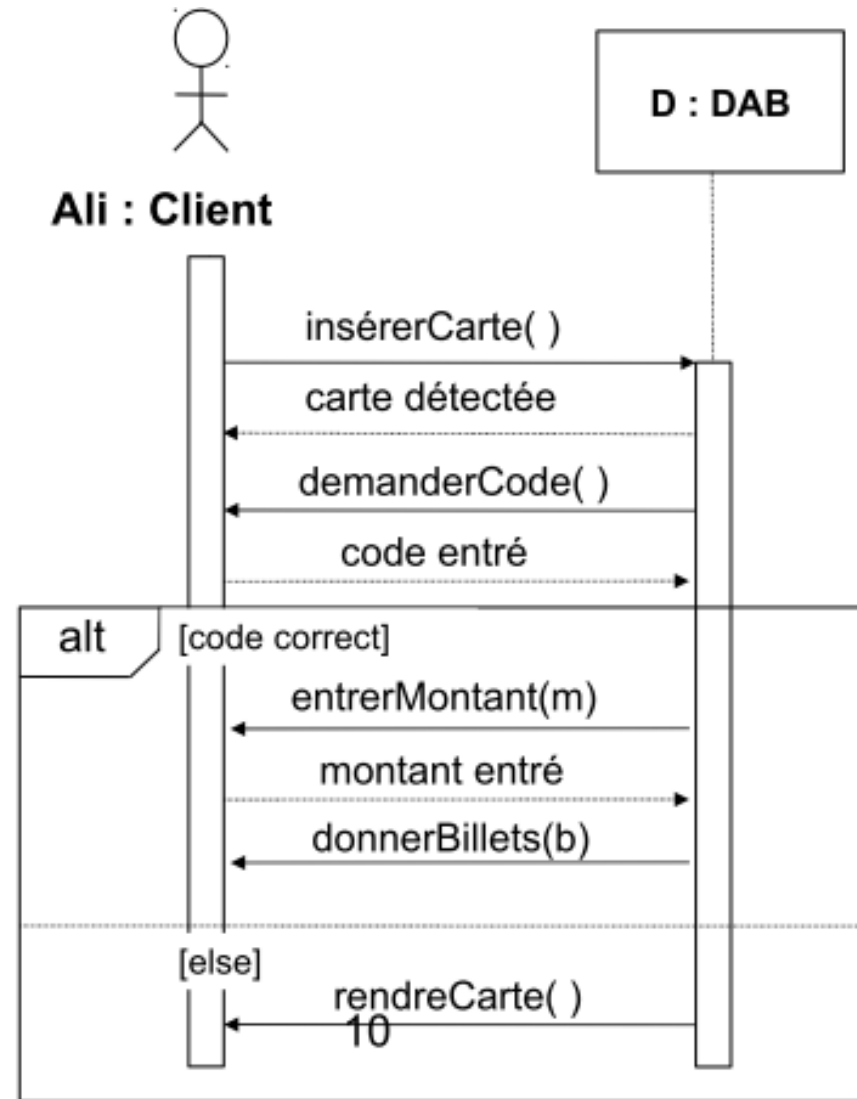
- Un fragment combiné permet de **décomposer une interaction complexe** en **fragments** suffisamment **simples** pour être compris.
 - **Recombinaison** les fragments restitue la complexité.
 - Syntaxe complète avec UML 2 : représentation complète de processus avec un langage simple (ex : processus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

Fragment combiné



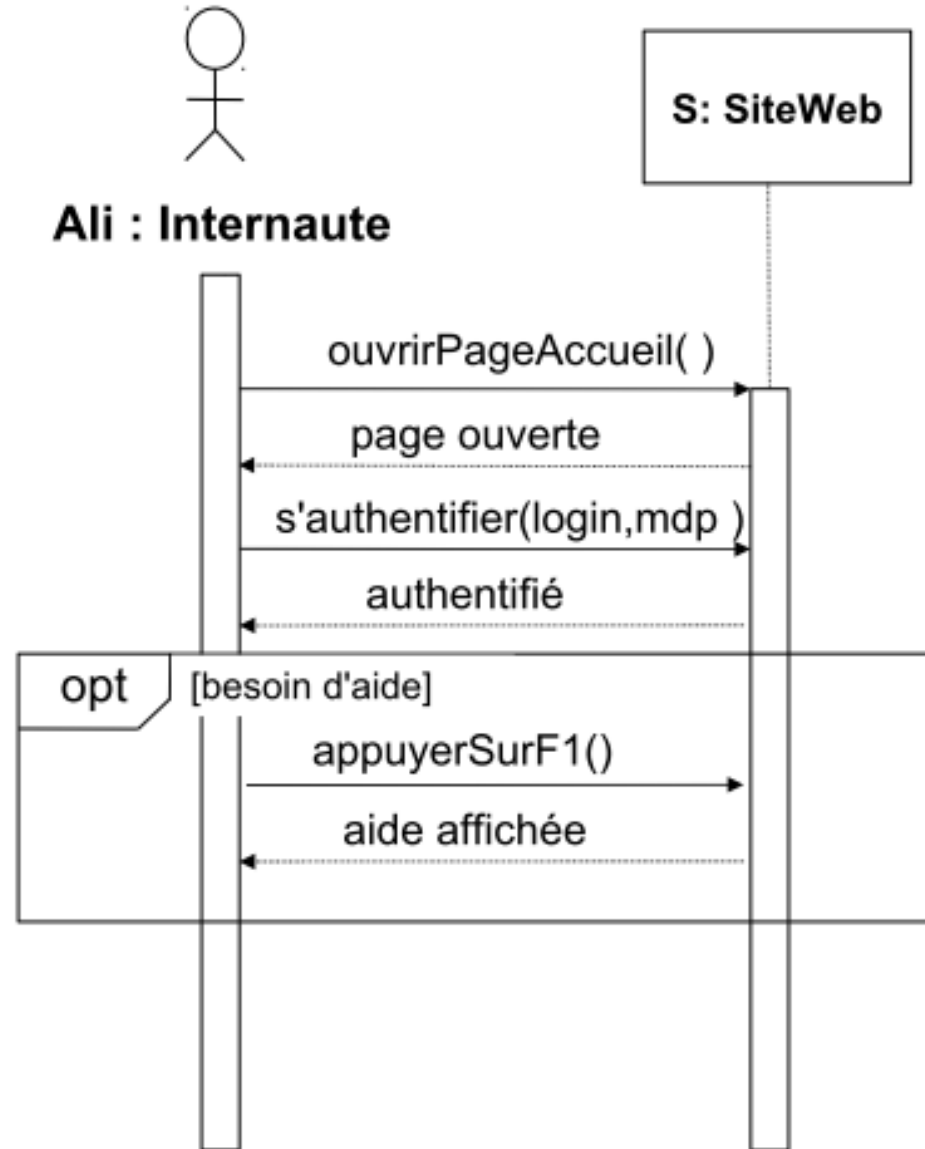
Opérateur « Alternative »

- Alternative (ou alt)
 - Opérateur conditionnel
 - Équivalent d'une exécution à choix multiples (switch)
 - Peut posséder plusieurs opérandes, chacune détient une condition de garde
 - Absence de condition de garde: condition vraie
 - Condition else: vraie si aucune autre condition n'est vraie



Opérateur « Option »

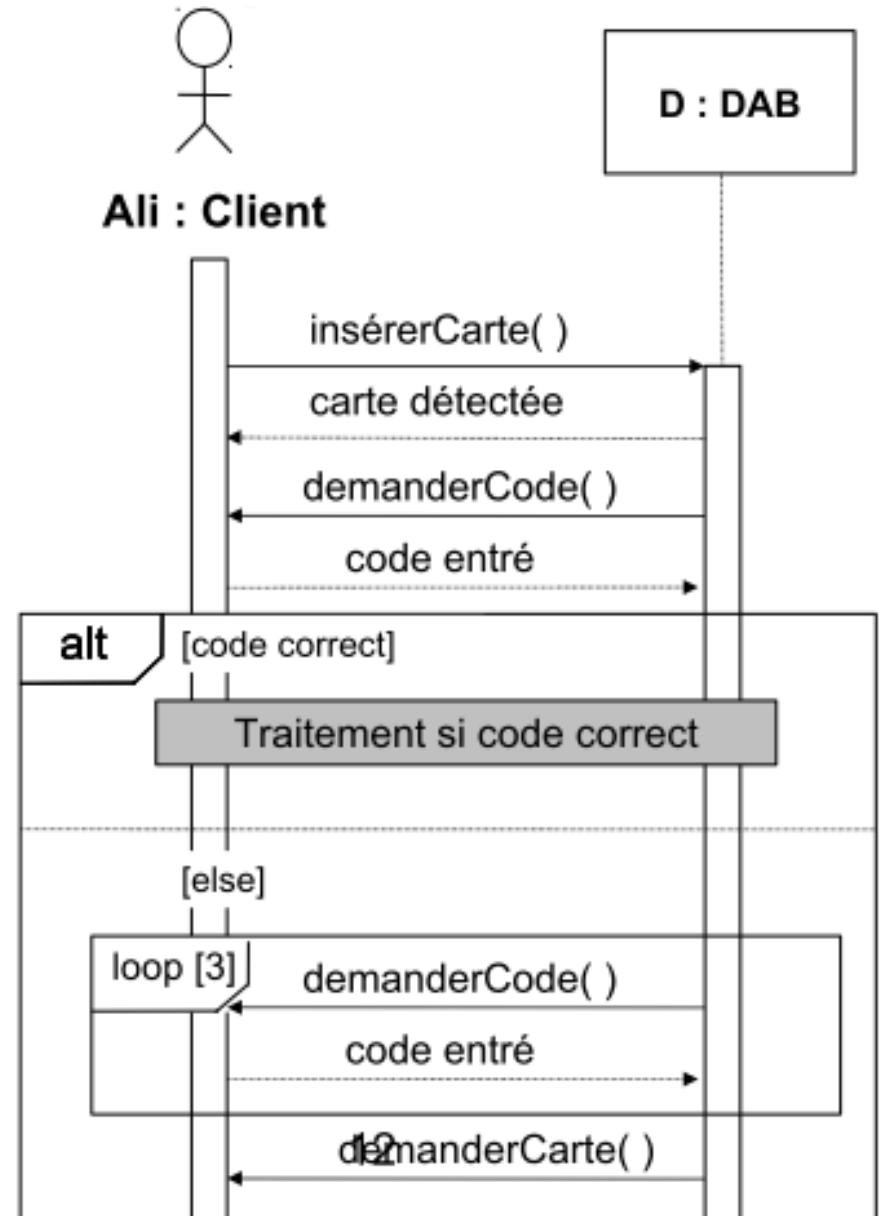
- Option (ou opt)
 - Représente un comportement qui peut se produire ou pas.
 - Équivalent à un alt à une seule branche et sans else



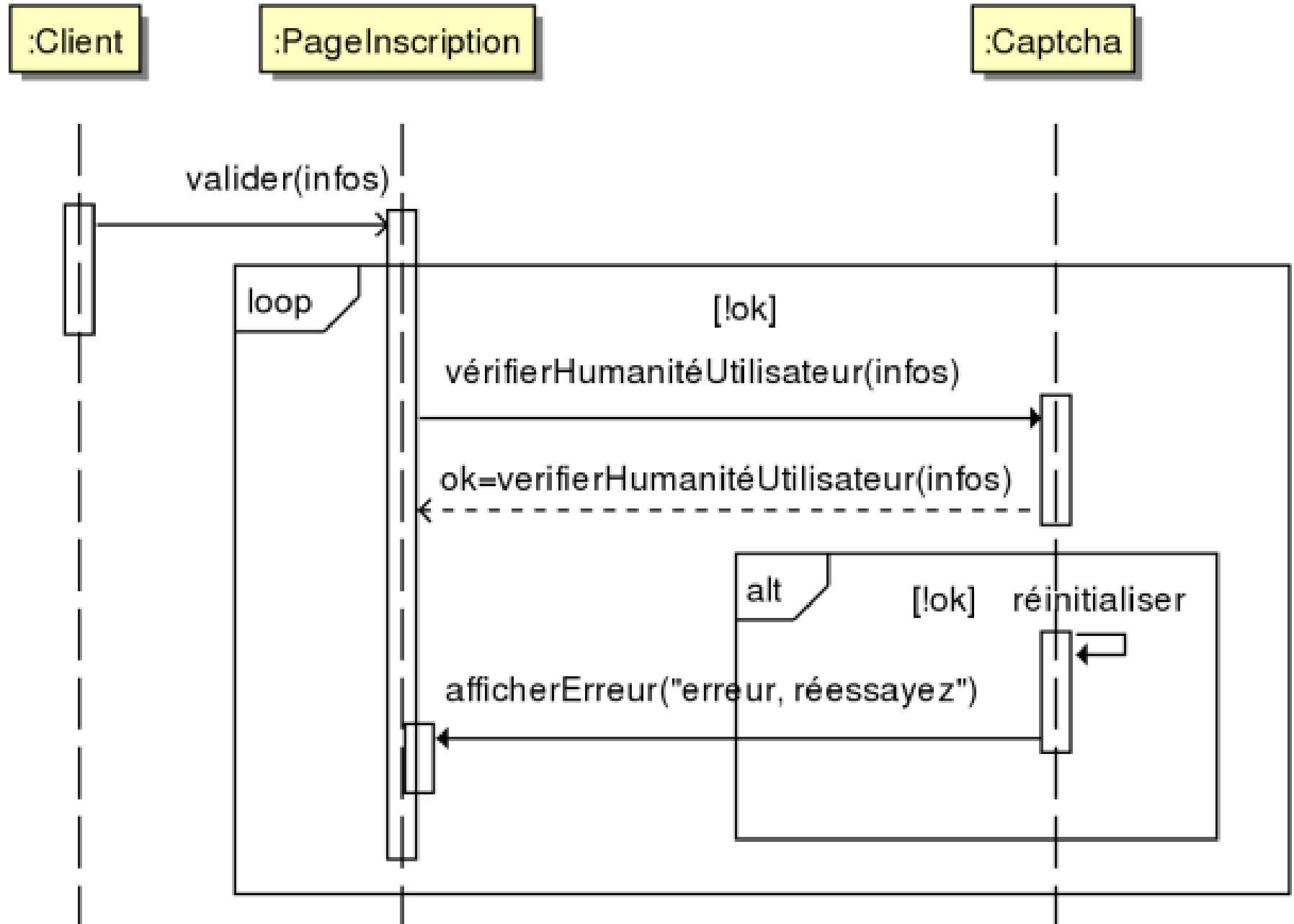
Opérateur « Loop »

- Loop

- Équivalent d'une boucle *for*
- Décrit des interactions qui s'exécutent en boucle
- La condition (garde) indique le nombre de répétitions (min et max) ou une condition booléenne à respecter

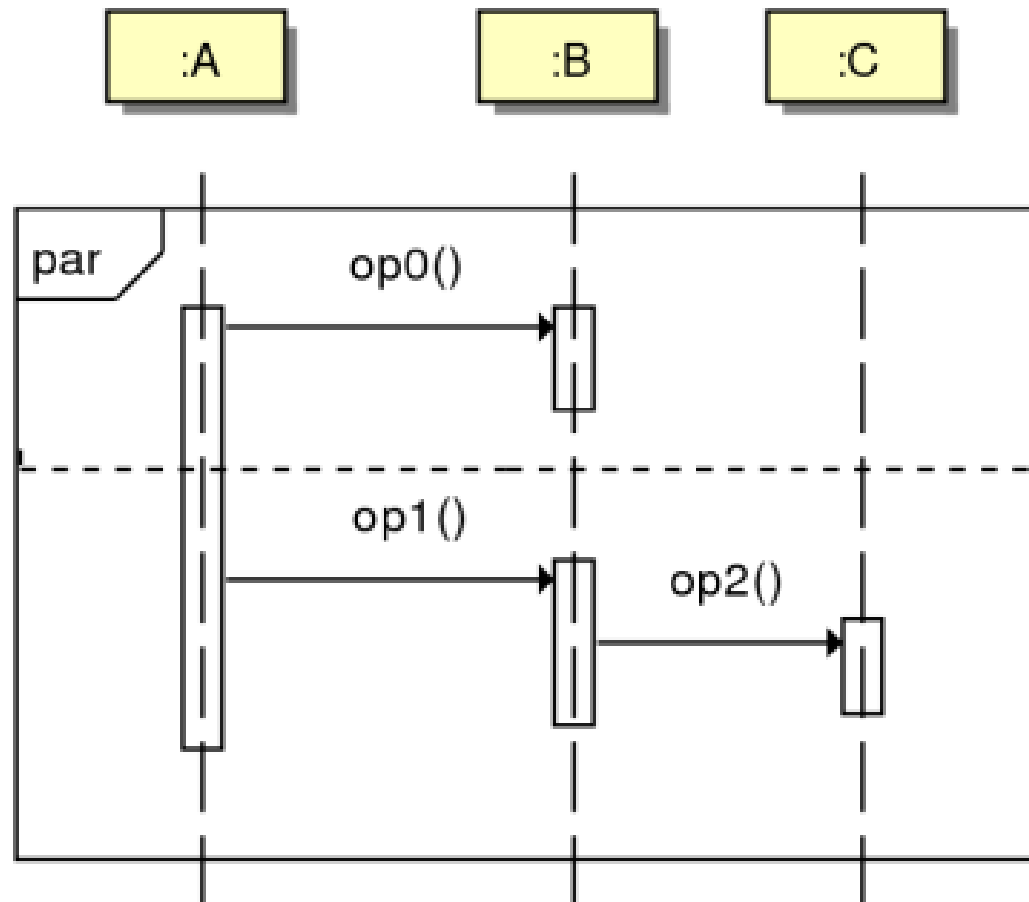


Opérateur « Loop »



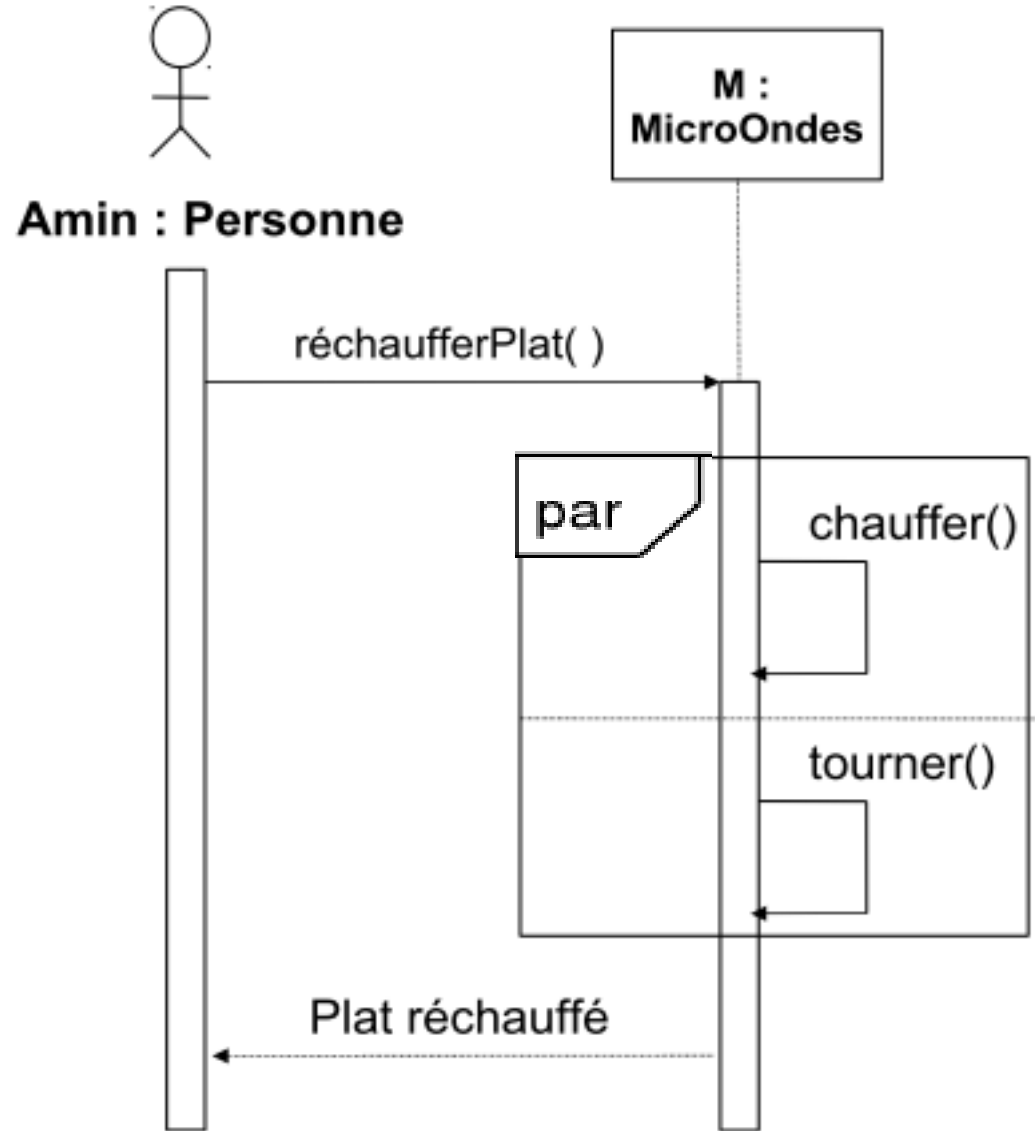
Opérateur parallèle

- 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.



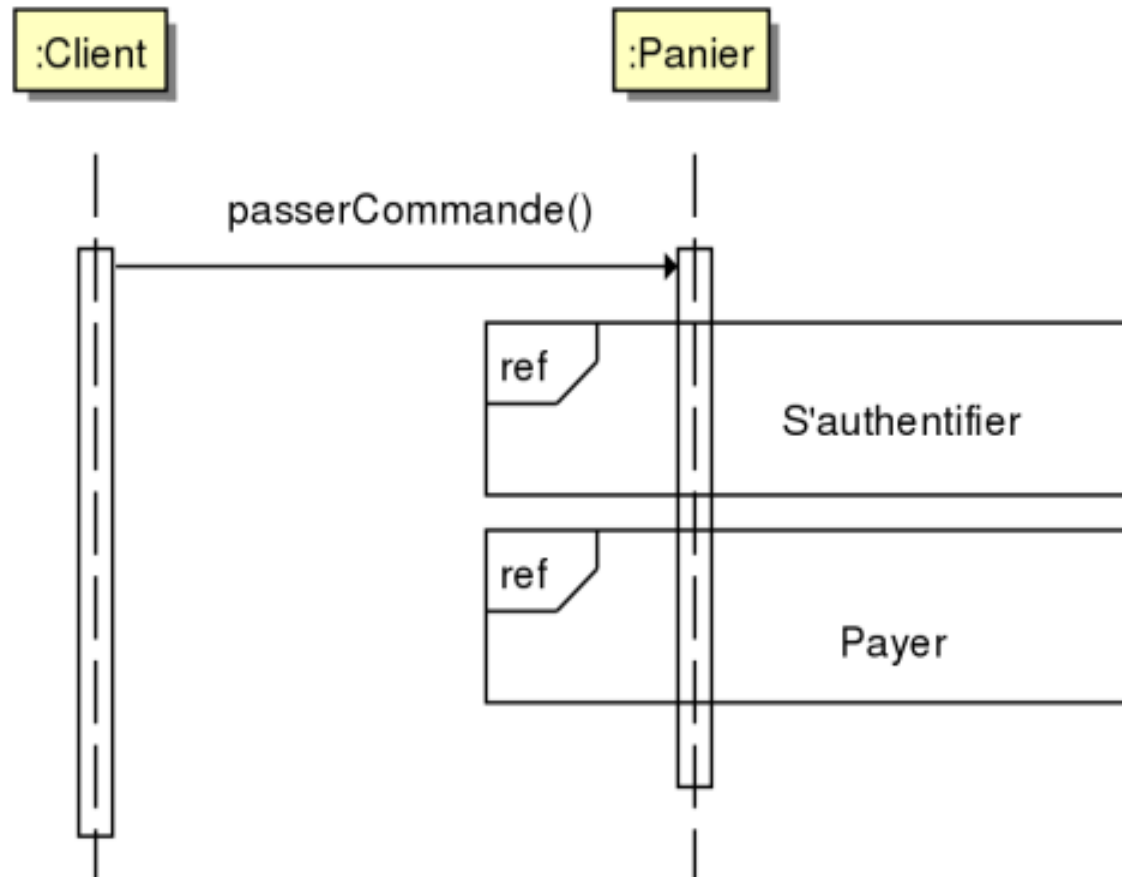
Opérateur parallèle

- Parallèle (ou par)
 - A au moins 2 sous-fragments exécutés simultanément
 - Simule une exécution parallèle



Réutilisation d'une interaction

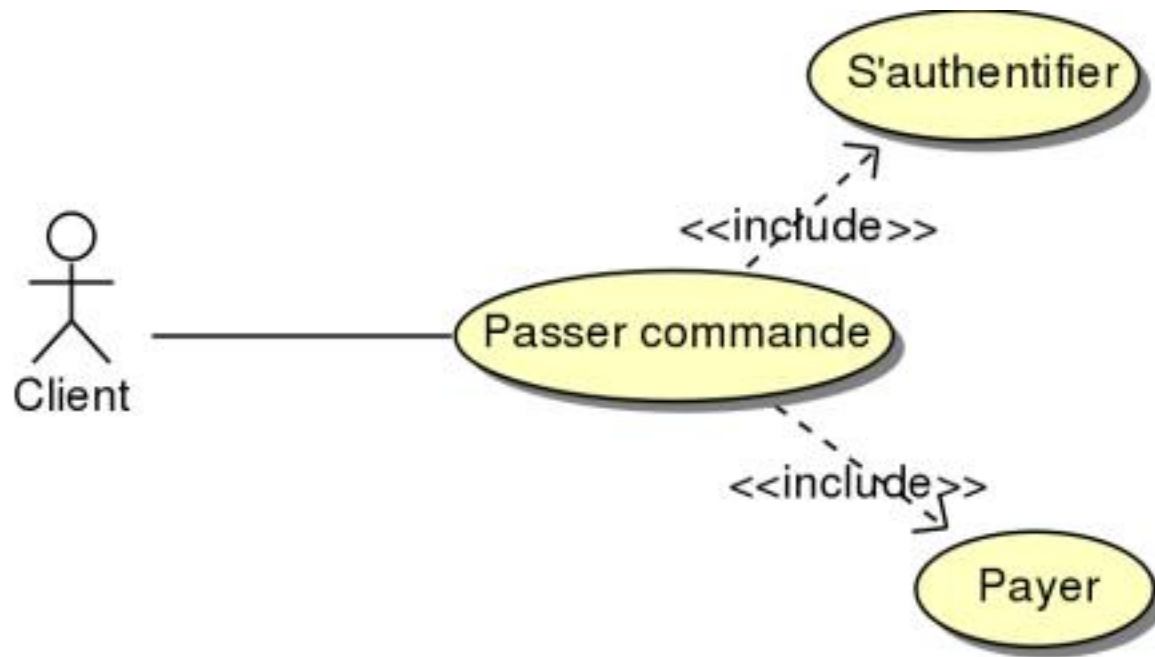
- **Réutiliser une interaction** consiste à placer un fragment portant la référence « **ref** » là où l'interaction est utile.



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

Utilisation d'un Diagramme de Séquence pour modéliser un cas d'utilisation

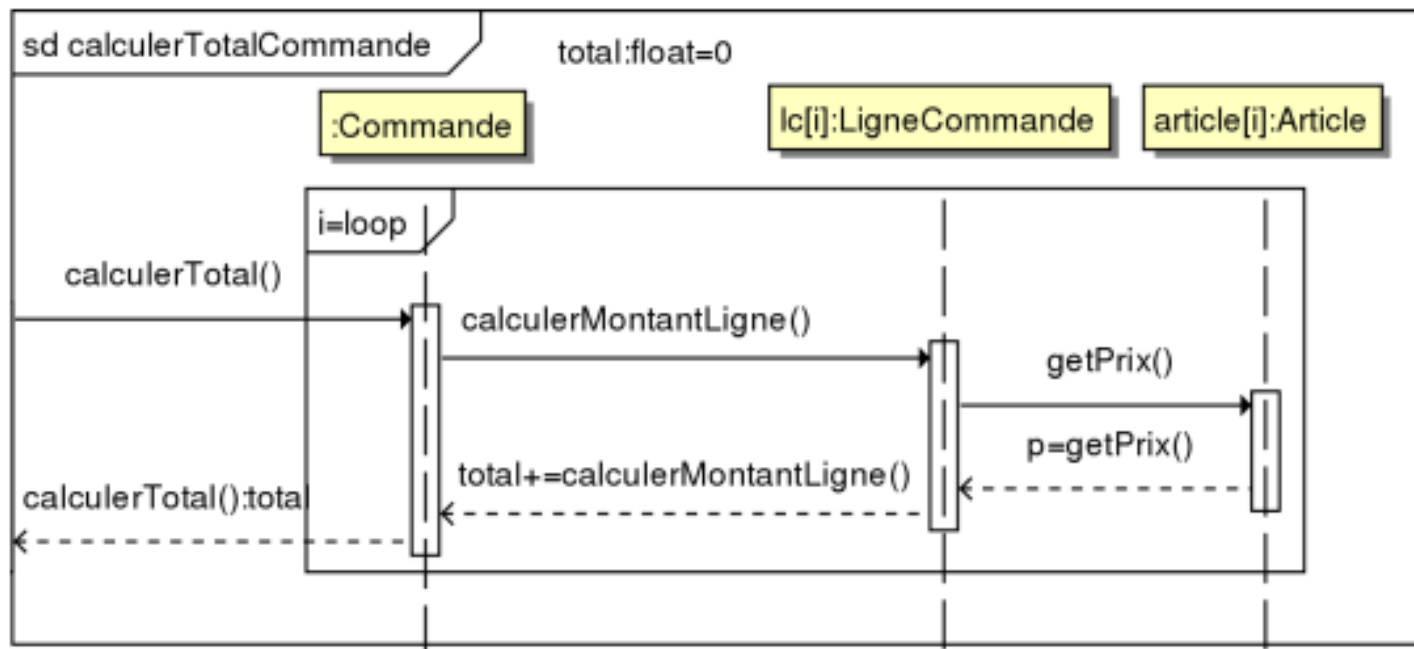
- Chaque cas d'utilisation donne lieu à un diagramme de séquences.



- 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

- Une entreprise désire développer un logiciel de gestion de formation de ses employés.
- Un employé saisie sa demande de formation dans un formulaire après avoir consulté le catalogue des formations proposées par l'entreprise.
- Le module de gestion des formations, ajoute cette demande dans la base de données et envoie une notification à la direction par email (date de saisie, code de l'employé)
- La direction consulte les détails de la formation à travers le même catalogue et répond à l'email du module de gestion par un refus ou un accord.
- Cet email est transmis par ce module à l'employé.
- En cas d'accord, le module envoie les formulaires d'inscription nécessaires avec le planning des sessions de formation à l'employé.

