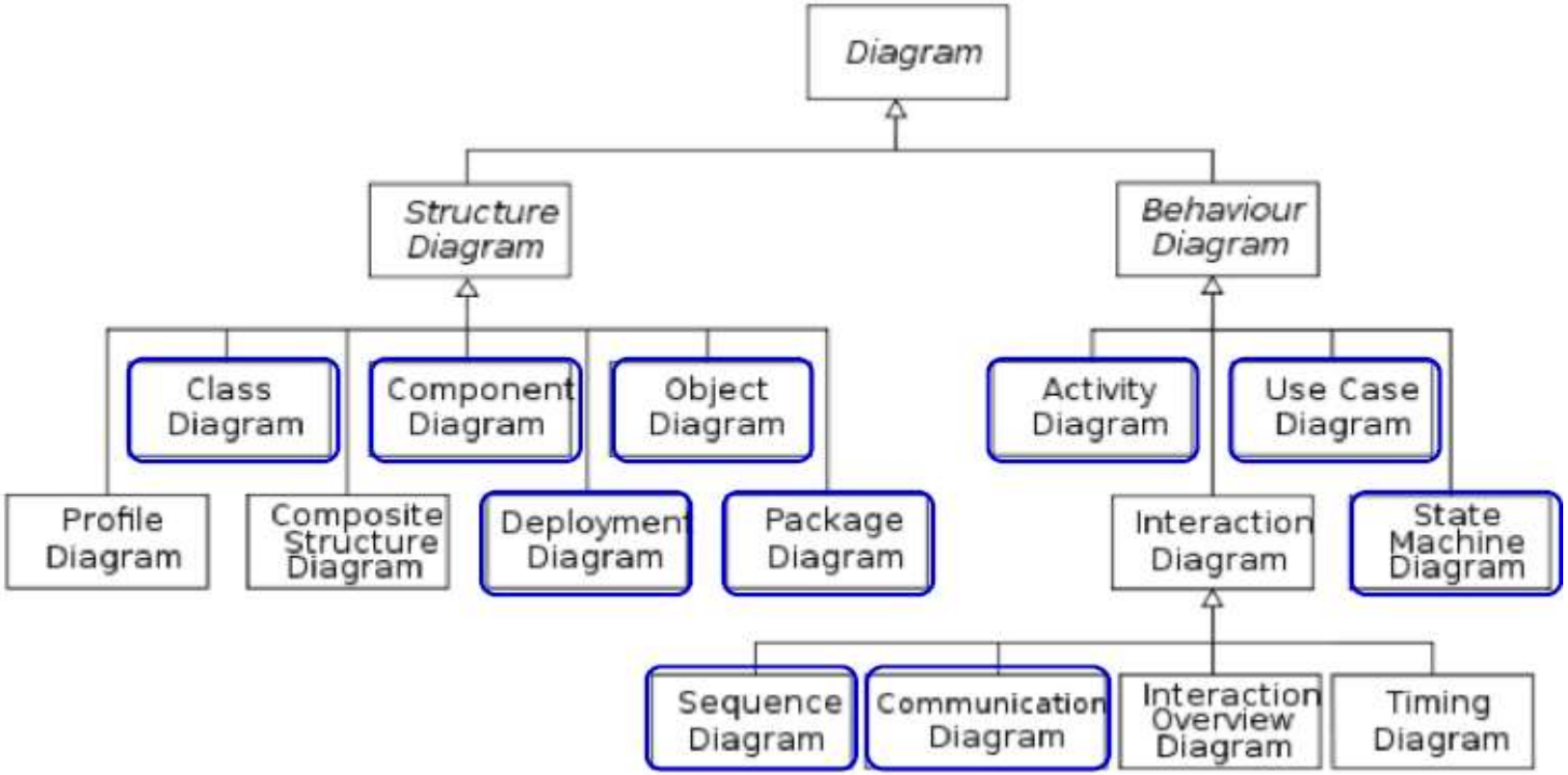
Génie Logiciel

Chapitre 5
Diagrammes UML : vue dynamique

Niveau: 3ième année Licence informatique

Année: 2025/2026

Les digrammes d'UML 2



Et un langage pour exprimer des contraintes : OCL

Plan

- Diagramme d'interaction
  - ✓ Séquence
  - ✓ Communication
- Diagrammes d'états/transitions
- Diagrammes d'activités

Diagramme d’interaction UML (Séquence et Communication)

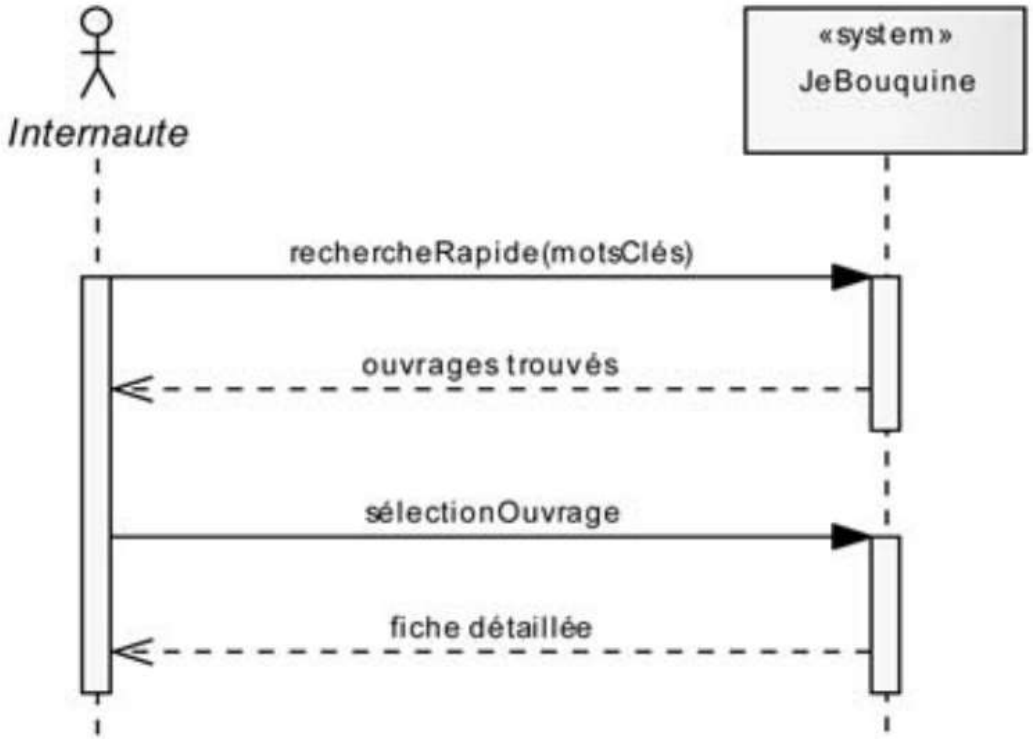
Objectif des diagrammes de séquence

- 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 du système ainsi que les liens entre les objets qui le composent.
  - ✓ Ils indiquent QUI interviendra dans le système pour réaliser les fonctionnalités décrites dans 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).

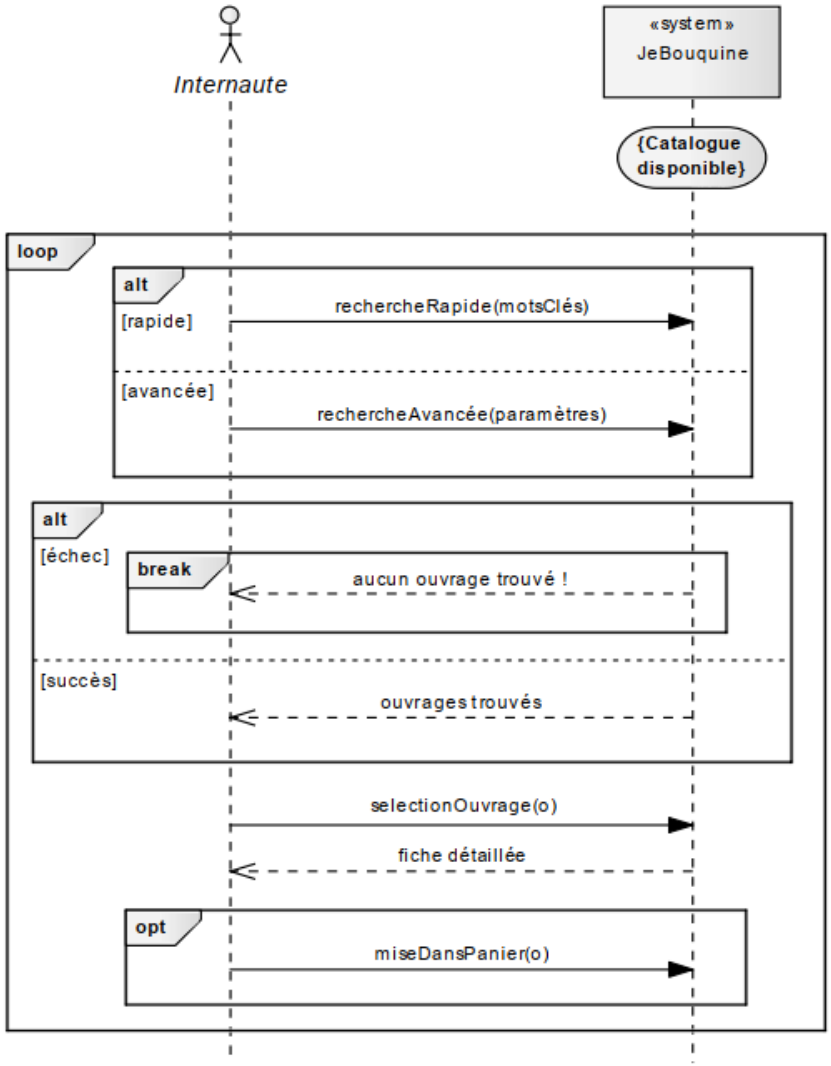
Exemple de motivation

Diagramme de séquence système de Chercher des ouvrages



Exemple de motivation (2)

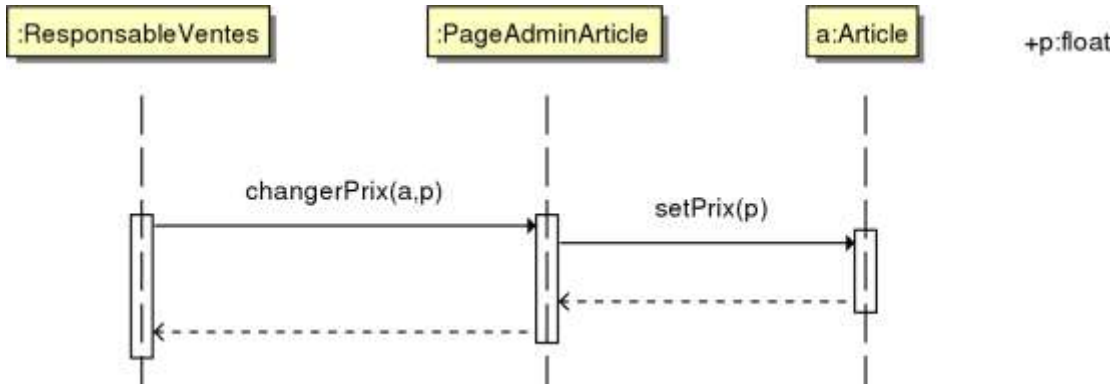
DSS plus complet de
Chercher des ouvrages



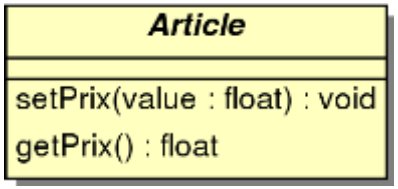
Exemple d'interaction

- Cas d'utilisation

- Diagramme de séquences correspondant :



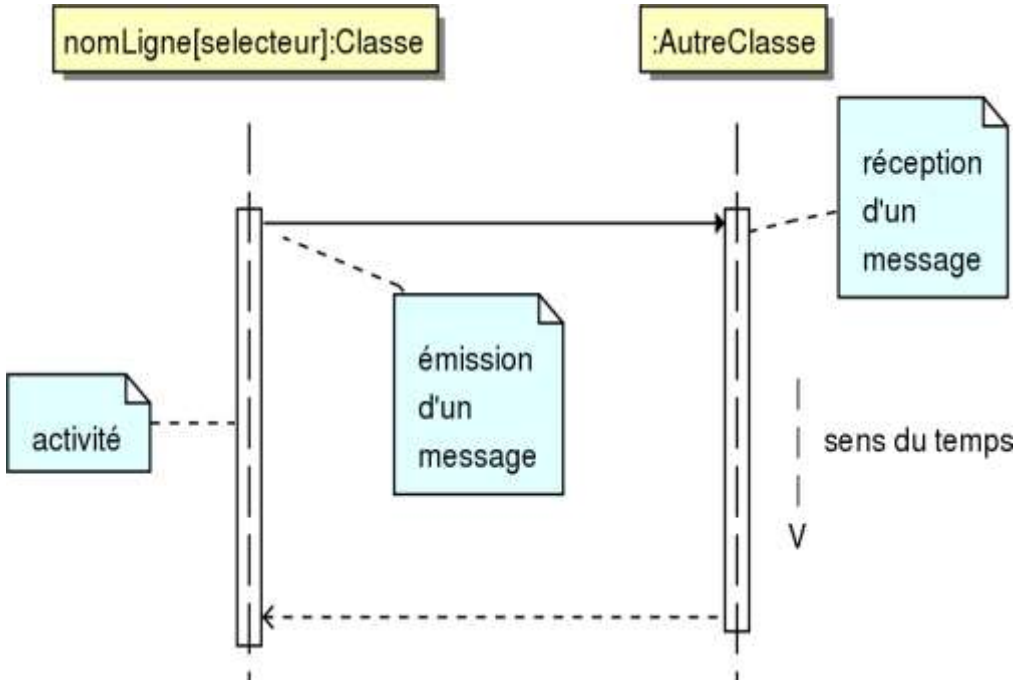
- Opérations nécessaires dans le diagramme de classes :



Ligne de vie

- Représente un participant à une interaction (objet ou acteur).
  - ✓ **maLigneDeVie [selecteur]: nomClasseOuActeur**

- Dans le cas d'une collection de participants, un sélecteur permet de choisir un objet parmi n
  - ✓ 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 :
  - ✓ Envoi d'un signal
  - ✓ Invocation d'une opération (appel de méthode)
  - ✓ Création ou la destruction d'un objet

- Période d'activité
  - ✓ 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

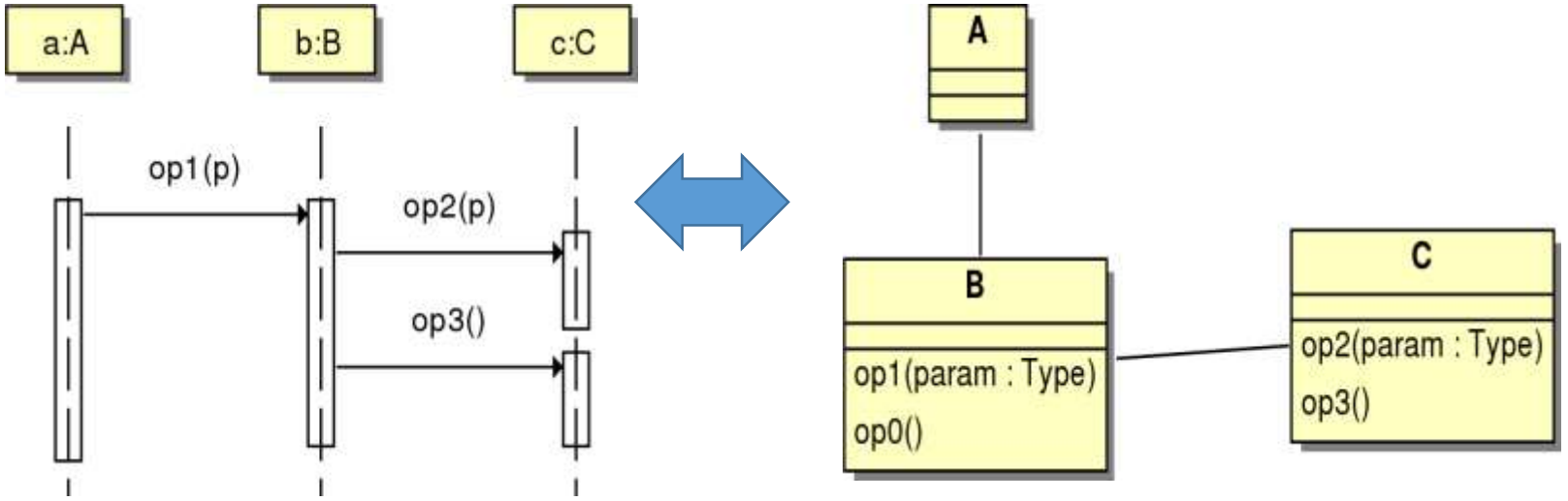
- Message synchrone bloque l'expéditeur jusqu'à la réponse du destinataire.
  - ✓ Le flot de contrôle passe de l'émetteur au récepteur.
  - ✓ Typiquement : appel de méthode: Si un objet A invoque une méthode d'un objet B, A reste bloqué tant que B n'a pas terminé.

  - ✓ 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 »**).

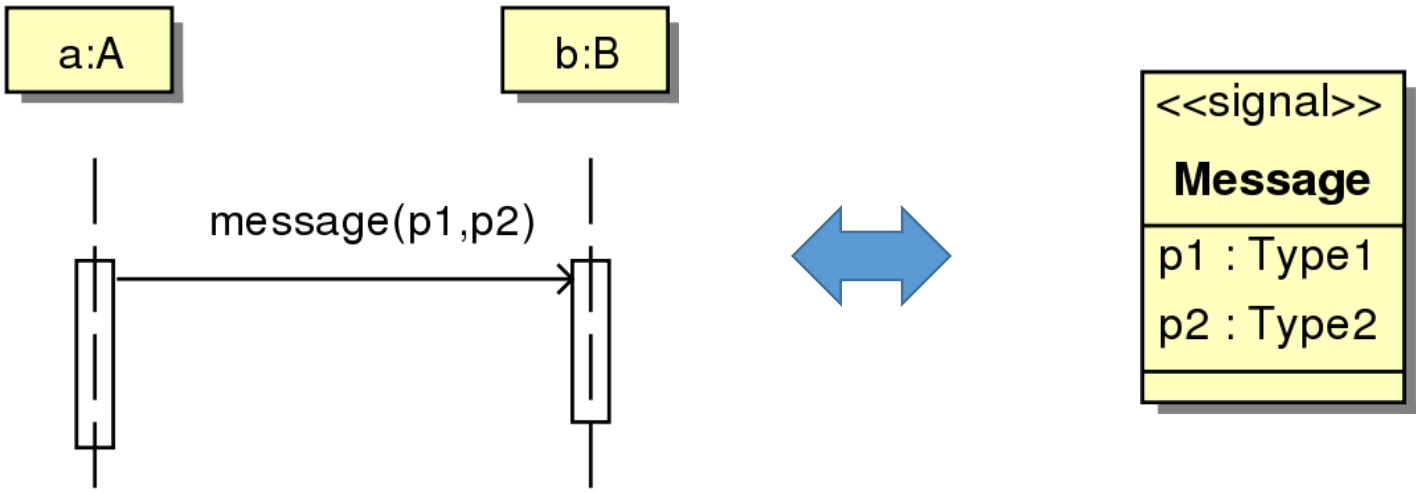
Correspondance messages / opérations

- 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



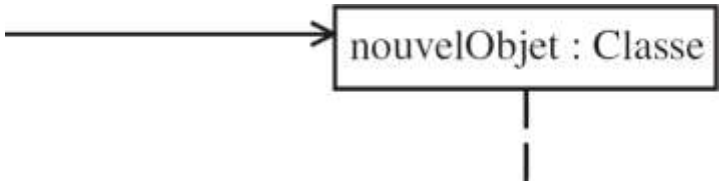
Correspondance messages / signaux

- 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.

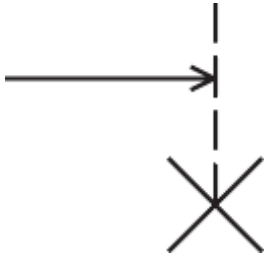


Création et destruction de lignes de vie

- 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.



Messages complets, perdus et trouvés

- Un message complet est tel que les événements d'envoi et de réception sont connus.
  - ✓ Représenté par une flèche partant d'une ligne de vie et arrivant à une autre ligne de vie.
  - ✓ Exemple: Un objet Client appelle la méthode payer() sur ServeurPaiement.

- 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 : Un broadcast radio, UDP non fiable,

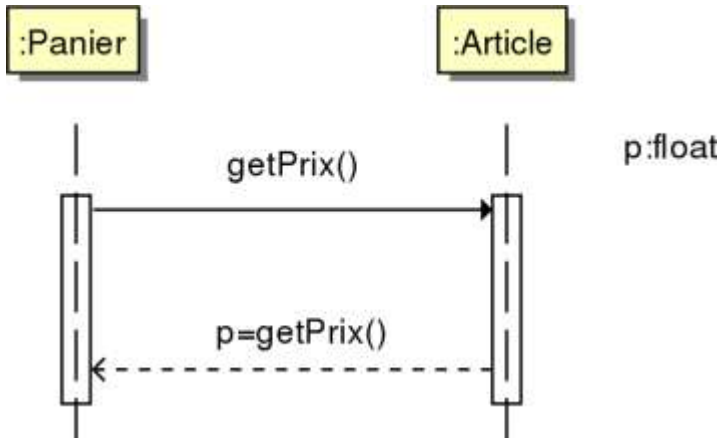
- Un message trouvé est tel que l'événement de réception est connu, mais pas l'événement d'émission.
  - ✓ Exemple:
    - Un utilisateur reçoit une notification, mais on ne modélise pas qui l'a envoyée.
    - Un système reçoit un signal venant de l'extérieur (capteur, alarme, API inconnue).

Syntaxe des messages

- Syntaxe des messages :
  - ✓ nomSignalOuOperation (parametres)
- Syntaxe des arguments :
  - ✓ nomParametre = valeurParametre
- Pour un argument modifiable :
  - ✓ nomParametre : valeurParametre
- Exemples :
  - ✓ appeler("Capitaine Hadock", 54214110)
  - ✓ afficher(x,y)
  - ✓ initialiser(x=100)
  - ✓ f(x:12)

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.

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

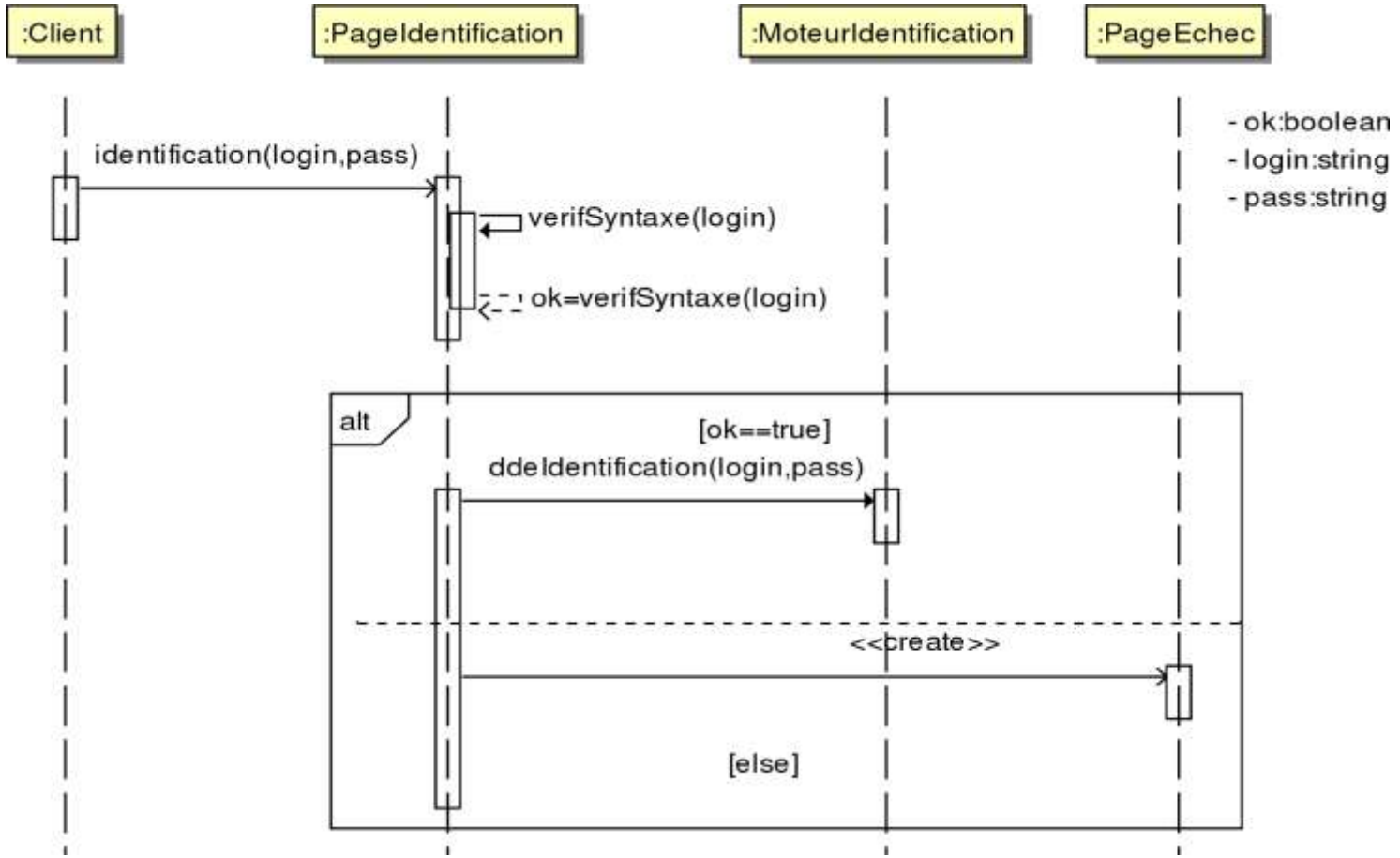
- Les messages de retour sont représentés en pointillés.

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).
- 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).

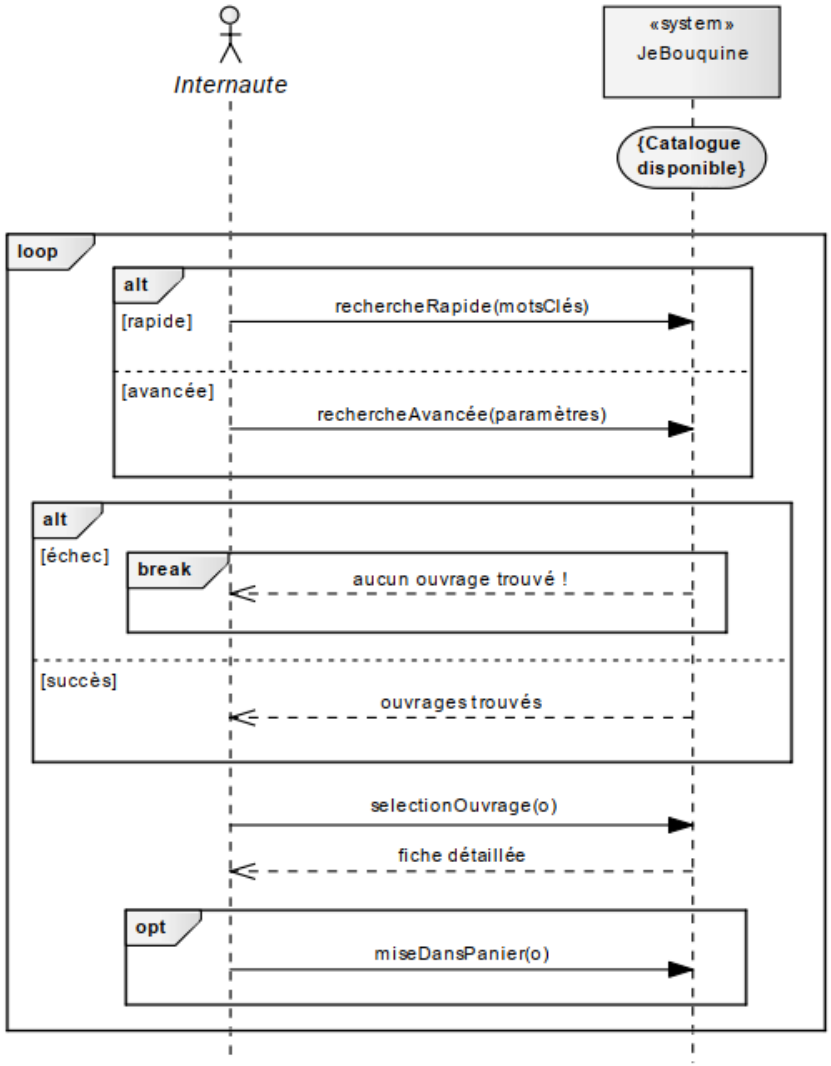
Exemple de fragment

- Opérateur conditionnel



Exemple de fragment

DSS plus complet de Chercher des ouvrages



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.

Questions

Exercice

- Expliquez la syntaxe des messages suivants extraits d'un diagramme de séquence

```
1)   f
2)   f( 0 )
3)   f( x )
4)   f( x = 0 )
5)   f( y = x )
6)   f( - )
7)   f( x, y )
8)   *
9)   y = f
10)  y = f( 0 )
11)  y = f( x = 0 )
12)  y = f( x ) : 0
```

Exercice-Solution

La plupart des messages portent f comme nom.

1) f est un message sans argument.
2) f(0) est un message qui reçoit en argument la valeur 0.
3) f(x) est un message qui reçoit la valeur de x en argument.
4) f(x = 0) est un message qui reçoit un argument x ayant pour valeur 0.
5) f(y = x) est un message ayant un argument y qui prend la valeur de x.
6) f(-) est un message avec un argument non défini.
7) f(x, y) est un message qui reçoit en arguments les valeurs de x et de y.
8) * est un message de type quelconque.
9) y = f est un message de réponse à un message f ; la valeur de retour est affectée à y.
10) y = f(0) est un message de réponse à un message f(0) ; la valeur de retour est affectée à y.
11) y = f(x = 0) est un message de réponse à un message f(x = 0) ; la valeur de retour est affectée à y.
12) y = f(x) : 0 est un message de réponse à un message f(x) ; la valeur de retour 0 est affectée à y