Bases de la conception orientée objet

Aspects dynamiques : diagrammes de séquence

Petru Valicov
petru.valicov@univ-amu.fr

http://pageperso.lif.univ-mrs.fr/~petru.valicov/Teaching.html

2017-2018

Institut Universitaire de Technologie
Aix-Marseille Université
Département Informatique, Aix

Motivation

- Diagramme de cas d'utilisation ≈ à QUOI sert le système.
- Diagramme de classes ≈ QUI sera à l'œuvre dans le système pour réaliser les fonctionnalités voulues

Ce que nous ne verrons pas :

- Les *diagrammes d'objets* - éclairer un diagramme de classes avec des exemples
- Les *diagrammes de composants* - l'architecture logicielle du système

Jusque là on a vu que les **aspects statiques** du système.

Qu'en est-il de la dynamique des objets, leurs interactions, leurs cycles de vies ?

Diagrammes d'interaction

Rappel : Un objet interagit pour implémenter un comportement.

Objectif : Représenter les communications avec le logiciel et au sein du logiciel

- **Diagramme de communication**
  - Représentation spatiale des objets et de leurs interactions
  - Aide à valider les associations du diagramme de classe en les utilisant comme support de transmission des messages.
- **Diagramme de séquence**
  - Représentation temporelle des interactions entre les objets
  - Chronologie des messages échangés entre les objets et avec les acteurs

En phase de conception les deux diagrammes sont **équivalents**

Diagrammes d'interaction – exemple

À partir d'un diagramme de classes et d'un cas d'utilisation :

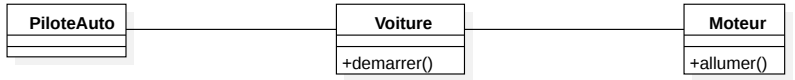
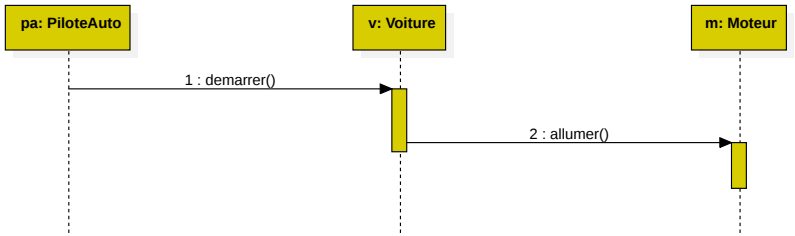


Diagramme de communication :

Diagramme de séquence :



Diagrammes d'interaction

Définition

Un diagramme d'interaction décrit la **réalisation des cas d'utilisation** sur le système décrit par le diagramme de classes

- Point de vue **interne** sur le fonctionnement du système
- Description au niveau de l'**instance** (état du système à un instant T)
- Description de **scénarios** particuliers d'une description textuelle d'un cas d'utilisation
- Représentation des **échanges de messages**
  - Entre les acteurs et le système, entre les objets du système
  - De façon chronologique

Diagrammes de séquence

Diagramme de séquence ≈ COMMENT les éléments du système interagissent entre eux et avec les acteurs

- Les objets du système interagissent en s'**échangeant des messages**
- Les acteurs interagissent avec le système au moyen d'une IHM (Interface Homme-Machine)
- On met l'accent sur la **chronologie** de l'envoi des messages

Diagrammes de séquence : premier exemple

Cas d'utilisation :

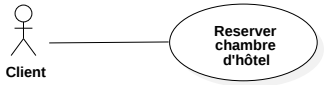
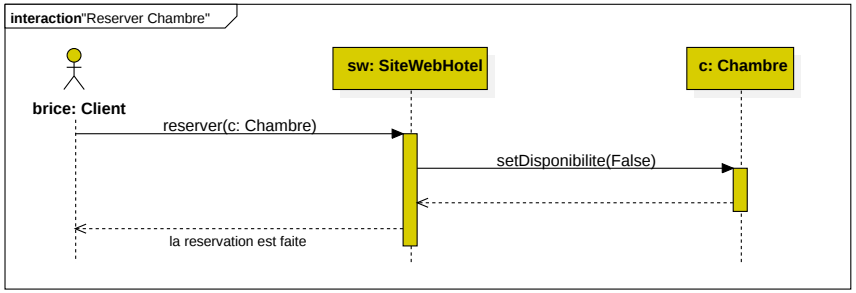


Diagramme de séquence correspondant :



Le diagramme de classes correspondant :

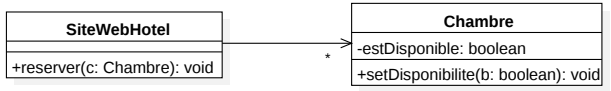
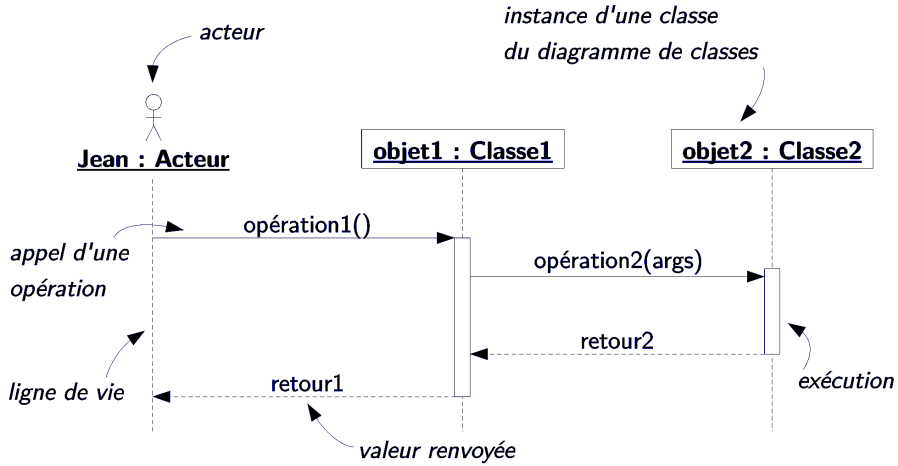


Diagramme de séquence : éléments de base

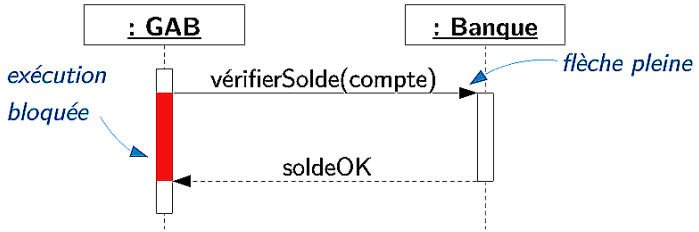


Messages

- Message ≈ une communication particulière entre des lignes de vie (objets ou acteurs)
- Les principales informations d'un diagramme de séquence
- Présentés dans un ordre chronologique
- 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**

Types de messages

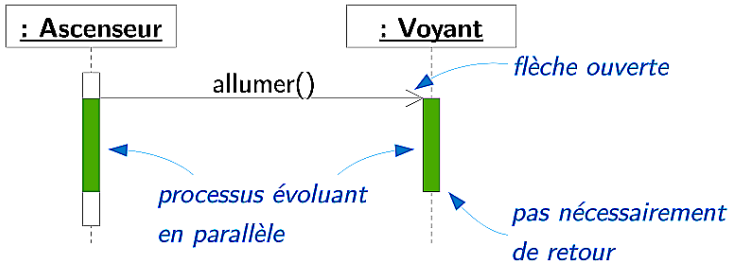
- Un message **synchrone** bloque l'expéditeur jusqu'à la réponse du destinataire.
- Le flot de contrôle passe de l'émetteur au récepteur.
- Le message de retour (en pointillés) marque la reprise du contrôle par l'émetteur



Typiquement : appel de méthode

Types de messages

Un message **asynchrone** n'est pas bloquant pour l'expéditeur : l'émetteur continue son exécution

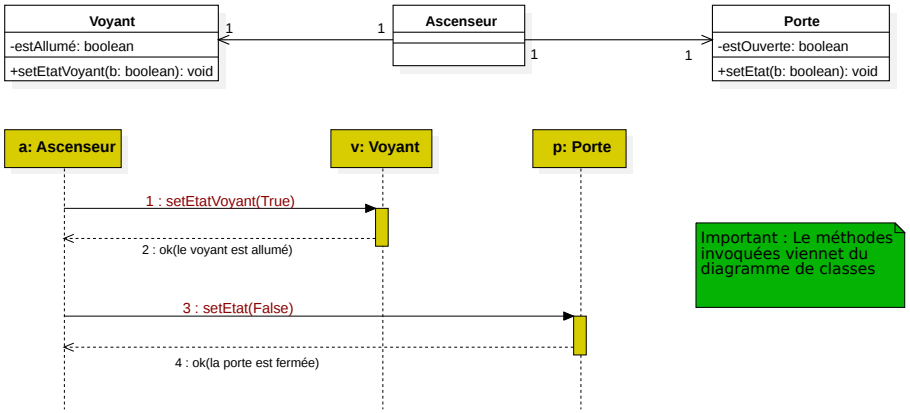


Le message envoyé peut être pris en compte par le récepteur à tout moment ou ignoré.

Correspondance messages / opérations

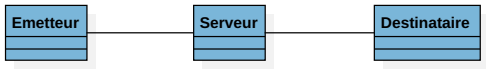
Les messages correspondent à des opérations dans le diagramme de classes.

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



Correspondance messages / opérations

Service de messagerie :



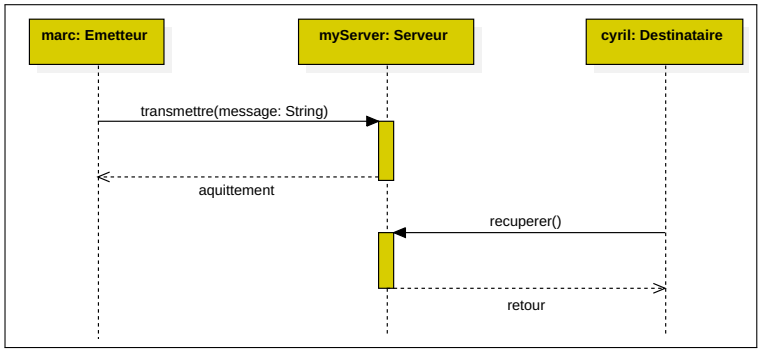
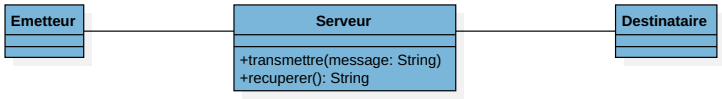
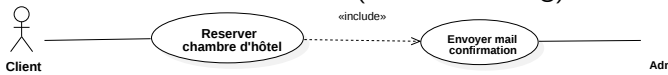


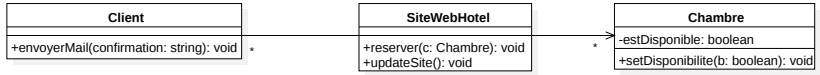
Diagramme de classe complété :

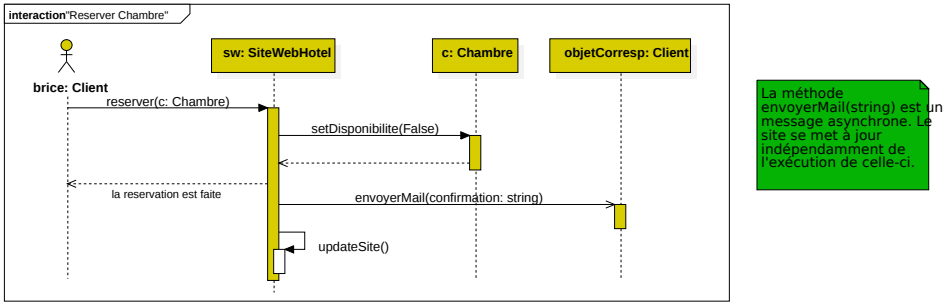


Correspondance messages / opérations

Envoyer un message asynchrone et continuer son activité sans attendre la réponse revient à invoquer une méthode dans un environnement multi-tâches (multi-threading).



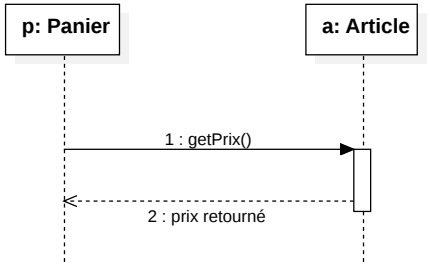




Messages de retour

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
- Ils sont utilisés pour spécifier le résultat de la méthode invoquée

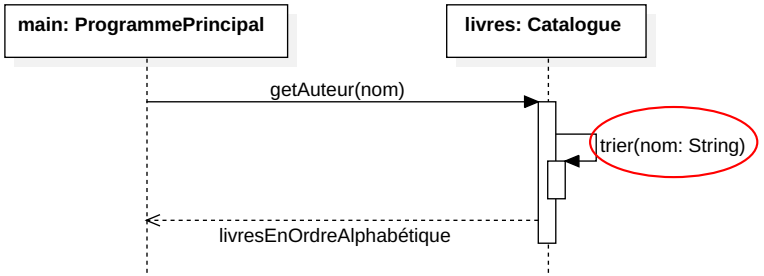


Le retour des messages asynchrones s'effectue par l'envoi de nouveaux messages asynchrones.

Message réflexif

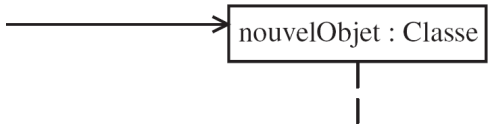
Dans le diagramme de classes :

Dans le diagramme de séquence :

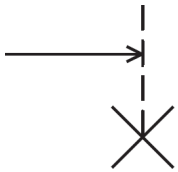


Création et destruction de lignes de vie

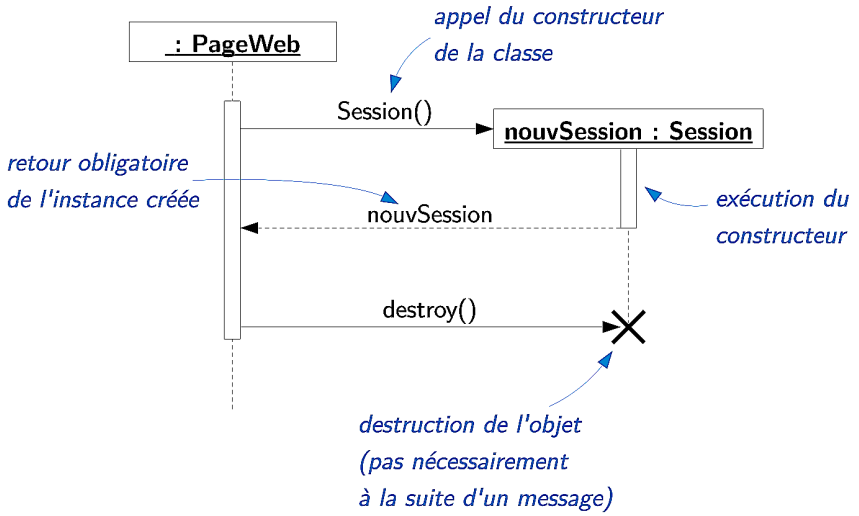
- La **création** d'un objet est matérialisée par une flèche qui pointe sur le sommet d'une ligne de vie.
  - On peut aussi utiliser un message asynchrone ordinaire portant le nom « create ».



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



Création et destruction d'objet

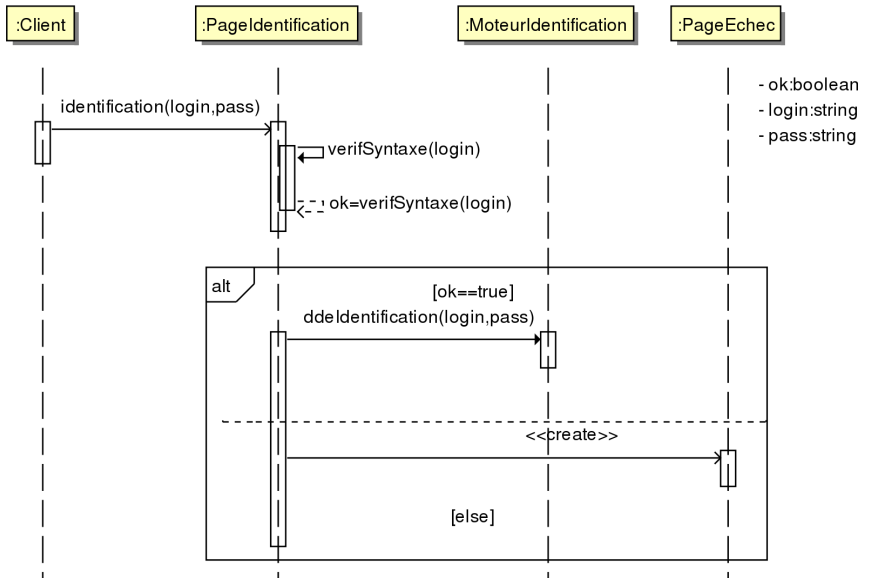


Fragment combiné

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).
- *opérateur d'interaction* - le type de la combinaison

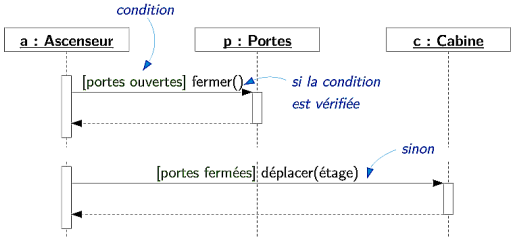
Exemple de fragment avec l'opérateur conditionnel



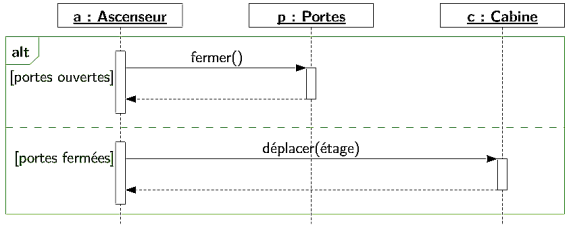
Type d'opérateurs d'interaction - Alternative

Principe : Condition à l'envoi d'un message

Deux diagrammes :



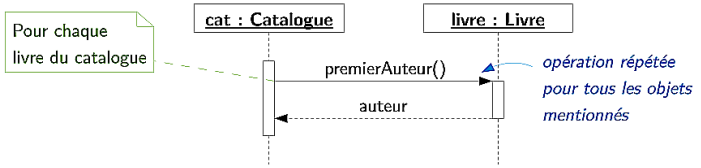
Bloc d'alternative **alt** :



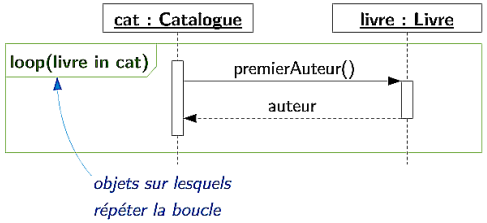
Type d'opérateurs d'interaction - Boucle

Principe : Répéter un enchaînement de messages

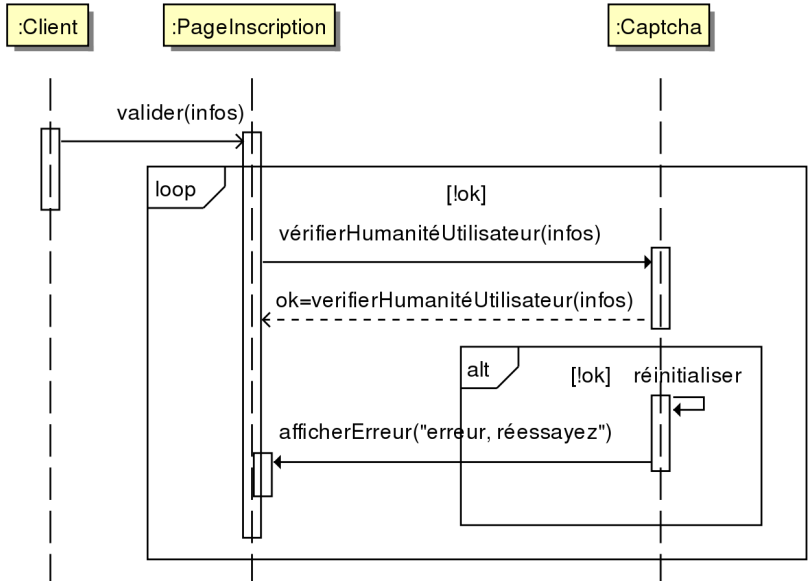
Note :



Bloc de boucle **loop** :

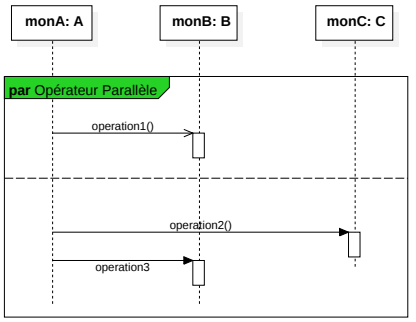


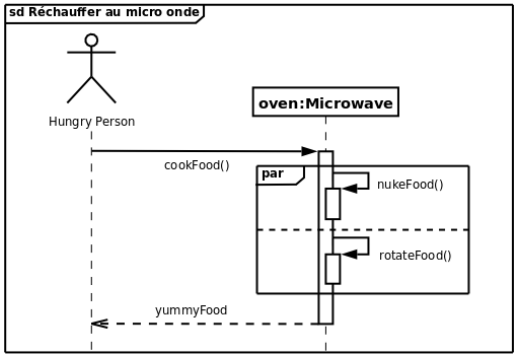
Opérateur de boucle



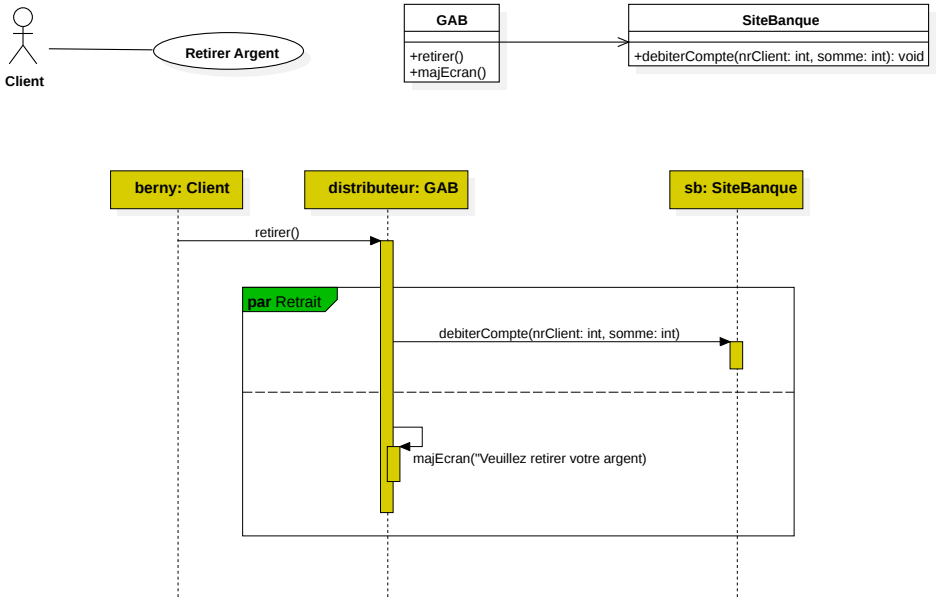
Opérateur parallèle

- 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.



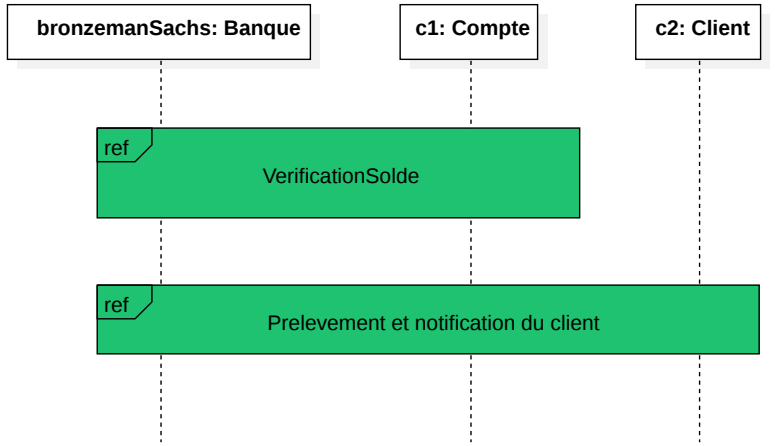


Opérateur parallèle : exemple



Référence à un autre diagramme de séquence

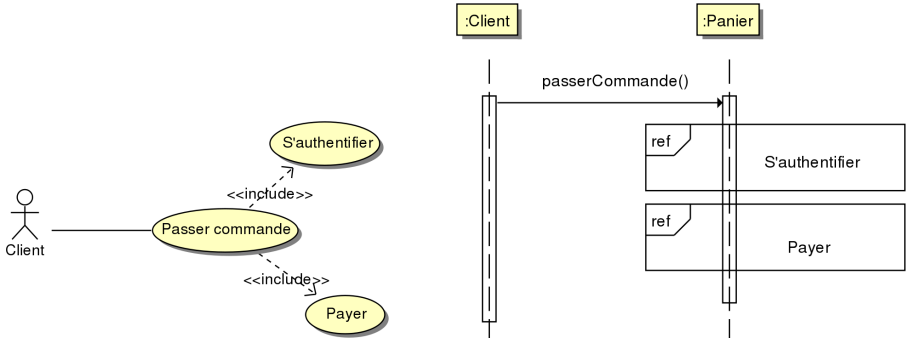
Réutiliser un diagramme de séquence consiste à placer un fragment portant la référence **ref** là où l'interaction est utile.



La référence n'a un sens que si le diagramme auquel on se réfère existe déjà !

Référence à un autre diagramme de séquence

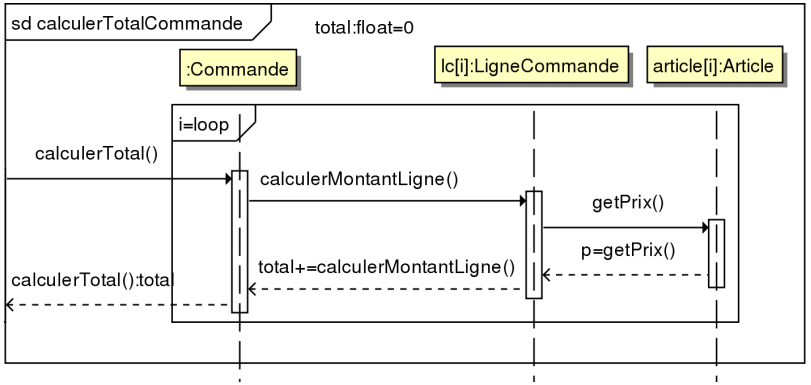
Chaque cas d'utilisation donne lieu à un diagramme de séquences



Pour modéliser les inclusions et les extensions des cas d'utilisation on utiliser le référencement des diagrammes de séquence correspondants

Utilisation d'un DS pour spécifier une méthode

- Un diagramme de séquence est identifié par un fragment **sd** 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



Exemple - Analyse et Conception

Cas d'utilisation

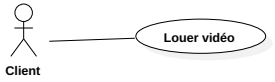


Diagramme de classes

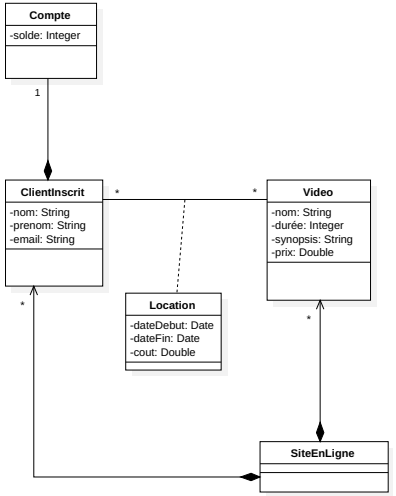
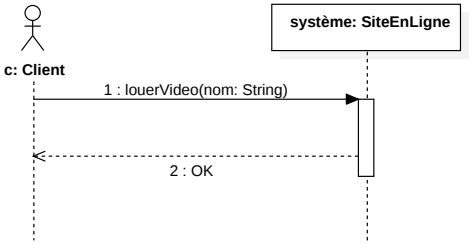
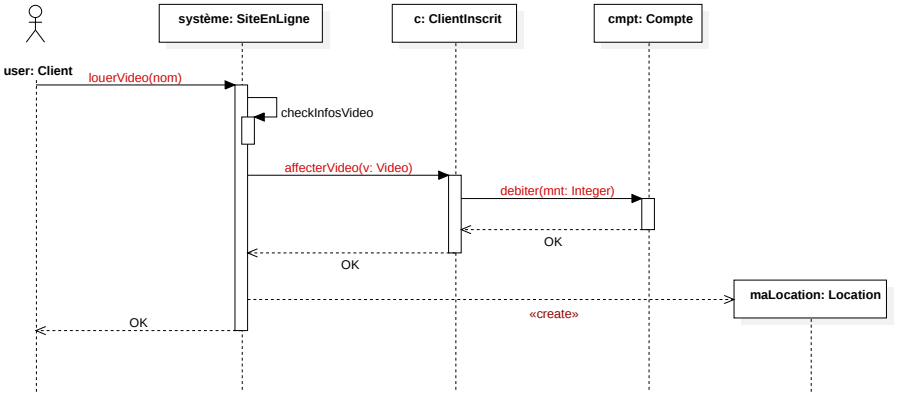


Diagramme de séquence simplifié





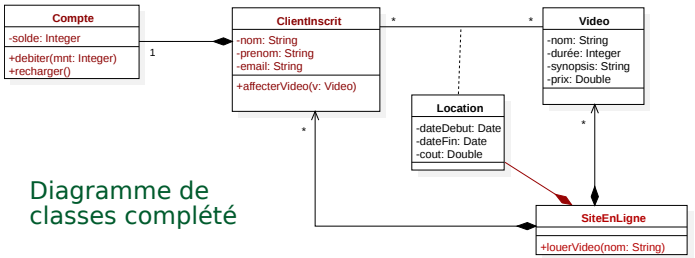


Diagramme de classes complété

Diagramme de séquence - règles

Messages entre acteurs et interface

- ≪ *Fausses* ≫ opérations liées au cas d'utilisation (même nom)
- Arguments et valeurs de retour simples : texte, nombre

Messages au sein du système

- Opérations du diagramme de classes
- Si message d'une ligne de vie de `ClasseA` vers une ligne de vie de `ClasseB`, alors
  - `ClasseA` et `ClasseB` liées par une association
  - Opération du message dans `ClasseB`