Conception Orienté Objet Diagramme de Sequence

AMOSSE EDOUARD

Introduction

- Permet de représenter les interactions entre
  - les entités,
  - les instances de classes,
  - Les sous systèmes
- Représente la séquence de messages entre les objets

Introduction

- Ils représentent :
  - Une suite spécifique d'événements survenant dans le système.
  - Une séquence spécifique d'actions et d'interactions entre les acteurs et le système.
- Suivent une chronologie de haut en bas
- Montre le flux de contrôle entre les participants d’un point de vue temporel

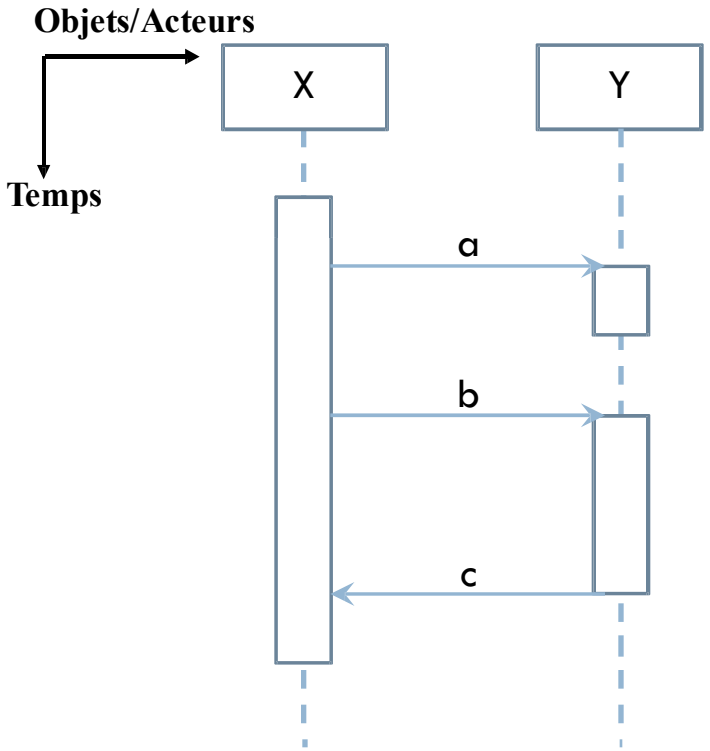
Représentation des Scenarios

Deux scenarios possibles:

- Normal : Tout se passe bien; correspond au déroulement du cas d'utilisation

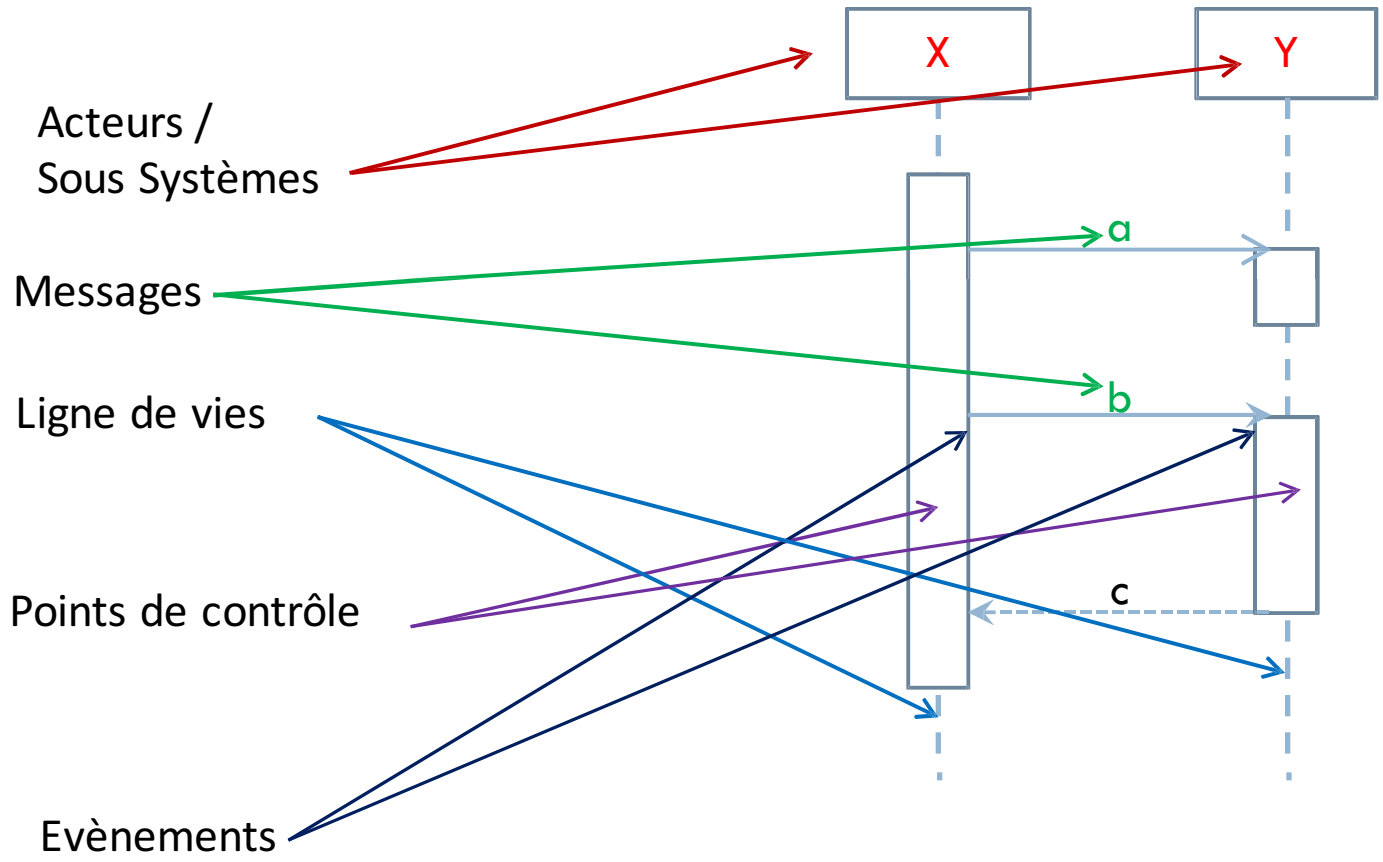
- Secondaires : Description des cas alternatifs (plusieurs choix), d'exceptions et d 'erreurs

Diagramme de séquence



- L'axe vertical représente le temps.
- L'axe horizontal représente les objets/acteurs impliqués dans l'interaction.
- Une ligne verticale est attachée à chaque objet/acteur et représente sa ligne de vie «lifeline».

Diagramme de séquence



Objets / Acteurs / Sous Systèmes

Dans un diagramme de séquence:

- Les objets, acteurs et/ou sous systèmes apparaissent dans la partie supérieur du diagramme

- Ils correspondent aux classes participant à l'interaction

Messages

- Représentés par des flèches directionnelles
- Représentant la communication entre objets (instances des classes)
- Différents de la relation structurelle entre les classes
- Un texte sur la flèche caractérise le message envoyé à l'objet appelé

Messages (2)

- Le diagramme étant orienté de haut vers le bas, l'ordre d'apparition des messages représente l'ordre d'appel
- Un objet peut envoyer des messages à lui même (ces messages sont dits réflexifs).
- Les flèches peuvent être orientées dans un sens ou dans l'autre

Messages (3)

Différents types de messages :

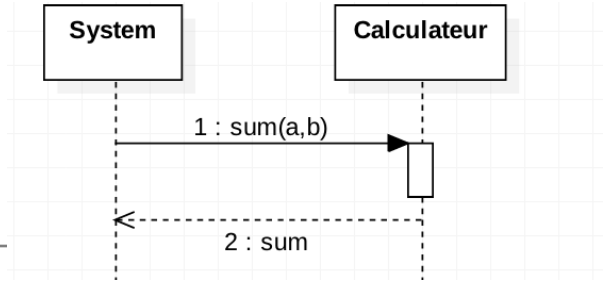
- Message simples
- Messages synchrones
- Messages asynchrones
- Messages de retour
- Message de création
- Messages de destruction

Messages (4)

Autres types de messages :

- Message conditionnel
- Message réflexif
- Message itératif

Messages Synchrones

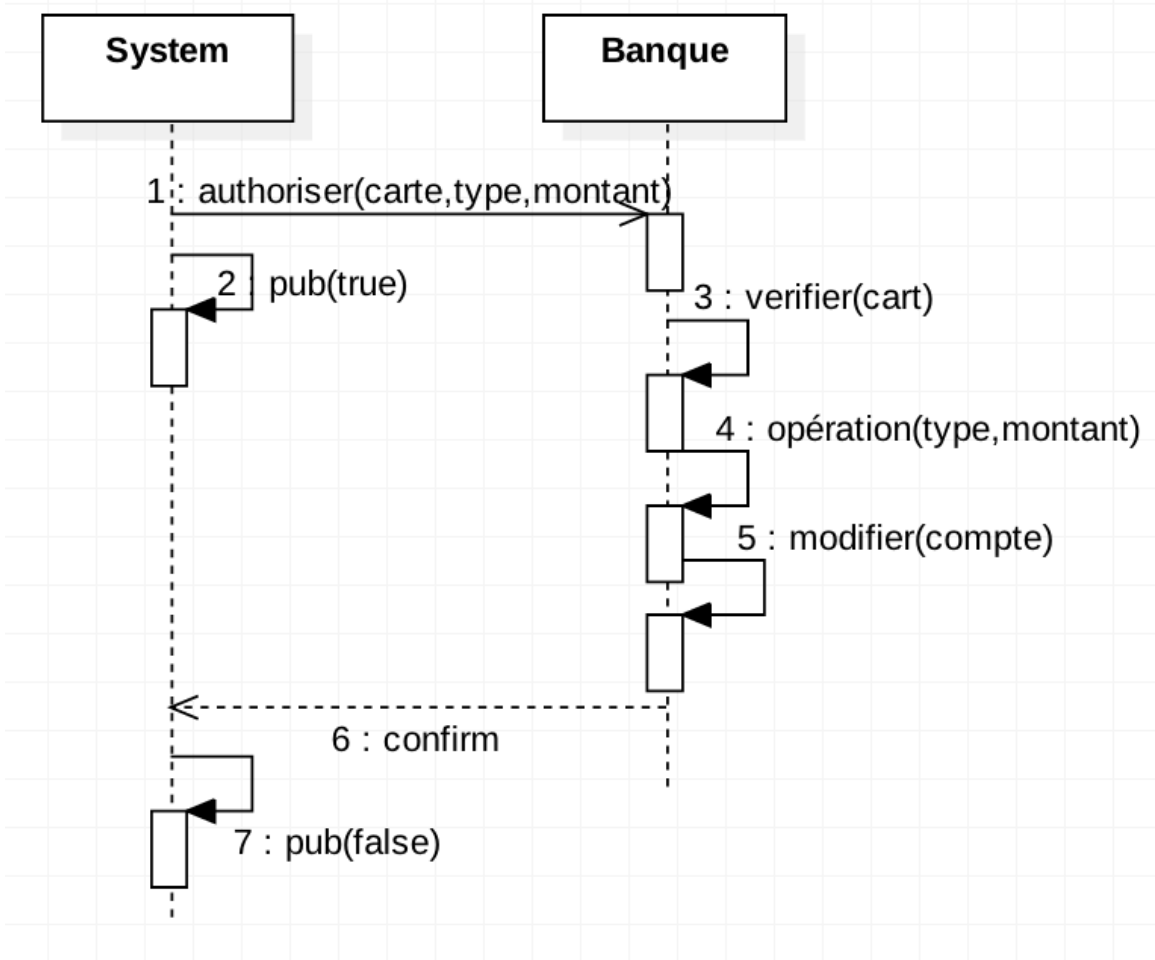


- Appel bloquant
- L'emeteur attend la fin de l'exécution du récepteur
- Les messages synchrones peuvent renvoyer un message de retour
- Les messages de retour sont représentés par des flèches en traits discontinus

Messages Asynchrones

- Appel non bloquant
- Les messages asynchrones sont des messages non bloquants.
- Ils n’attendent pas la fin de l’exécution du récepteur
- Les messages asynchrones peuvent également envoyer un message de retour

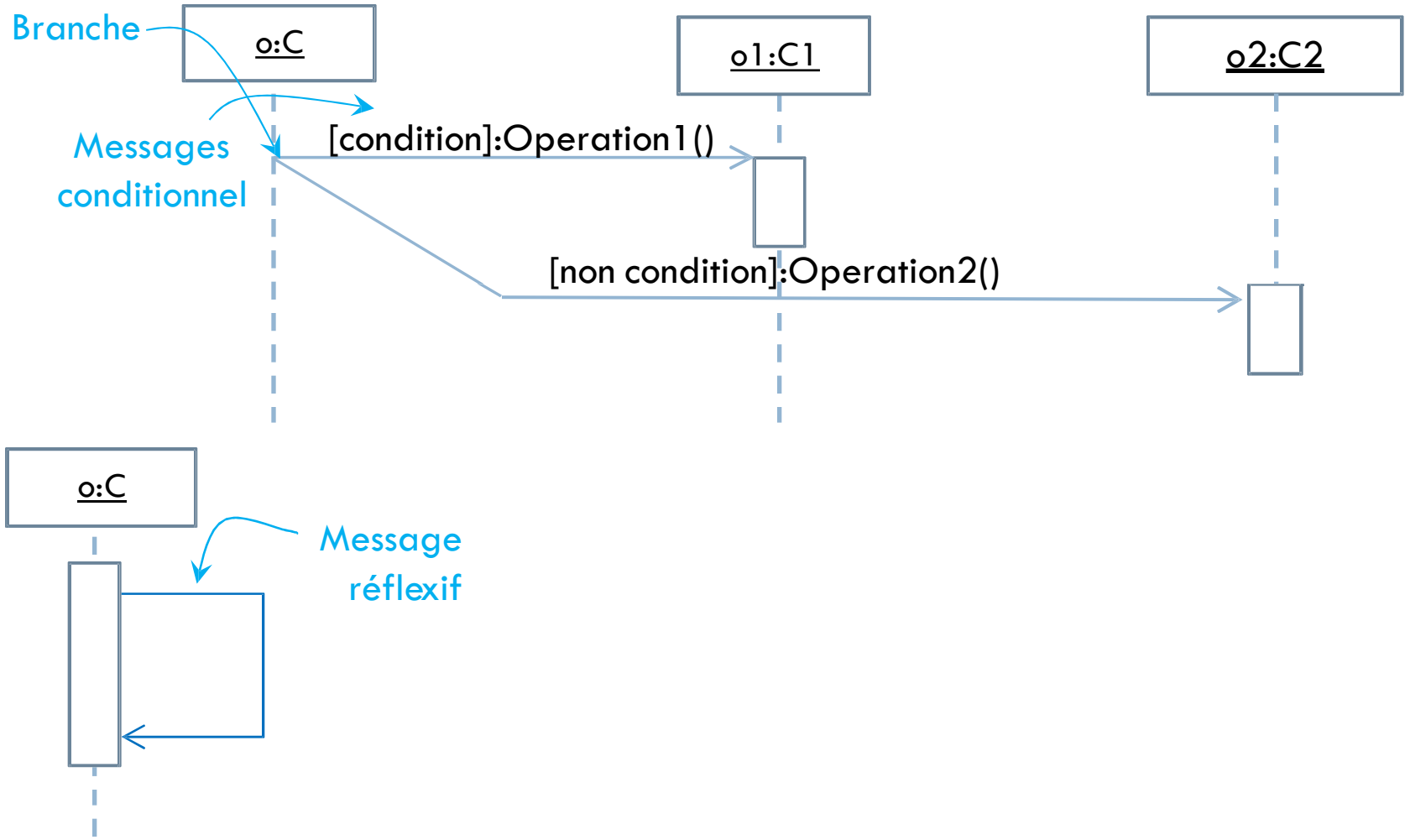
Messages Asynchrones (2)



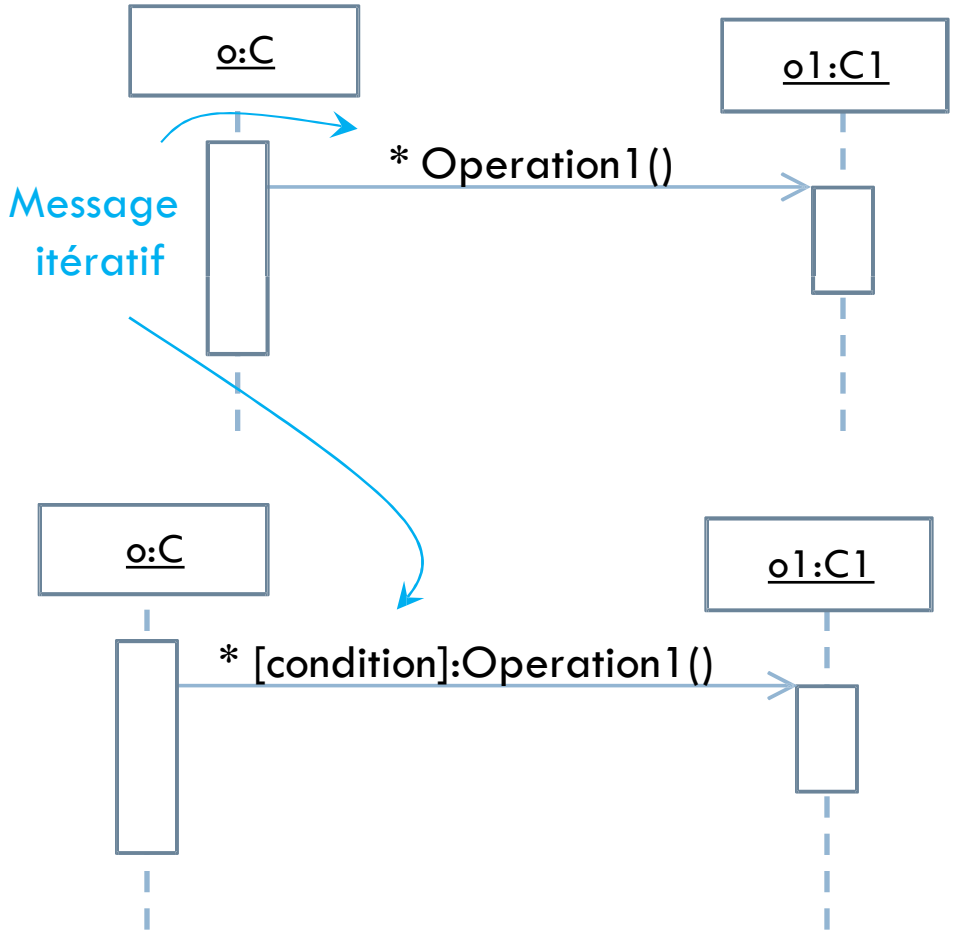
Messages et Classes

- La notion de message dans un diagramme de classes correspond aux méthodes dans les classes
- La méthode est implémentée du coté du récepteur
- Il est aussi possible de préciser des paramètres à l'envoi des messages

Message Conditionnel et Réflexif



Message itératif



Ligne de vie d’un objet

- Représente la durée de vie d’un objet dans un diagramme de séquence
- De nouveaux objets peuvent être créés pendant le cycle
- Un objet est créé quand un message lui est envoyé
- La fin de la ligne correspond à la description de l’objet

Création / Destruction d'objets

- **Création**

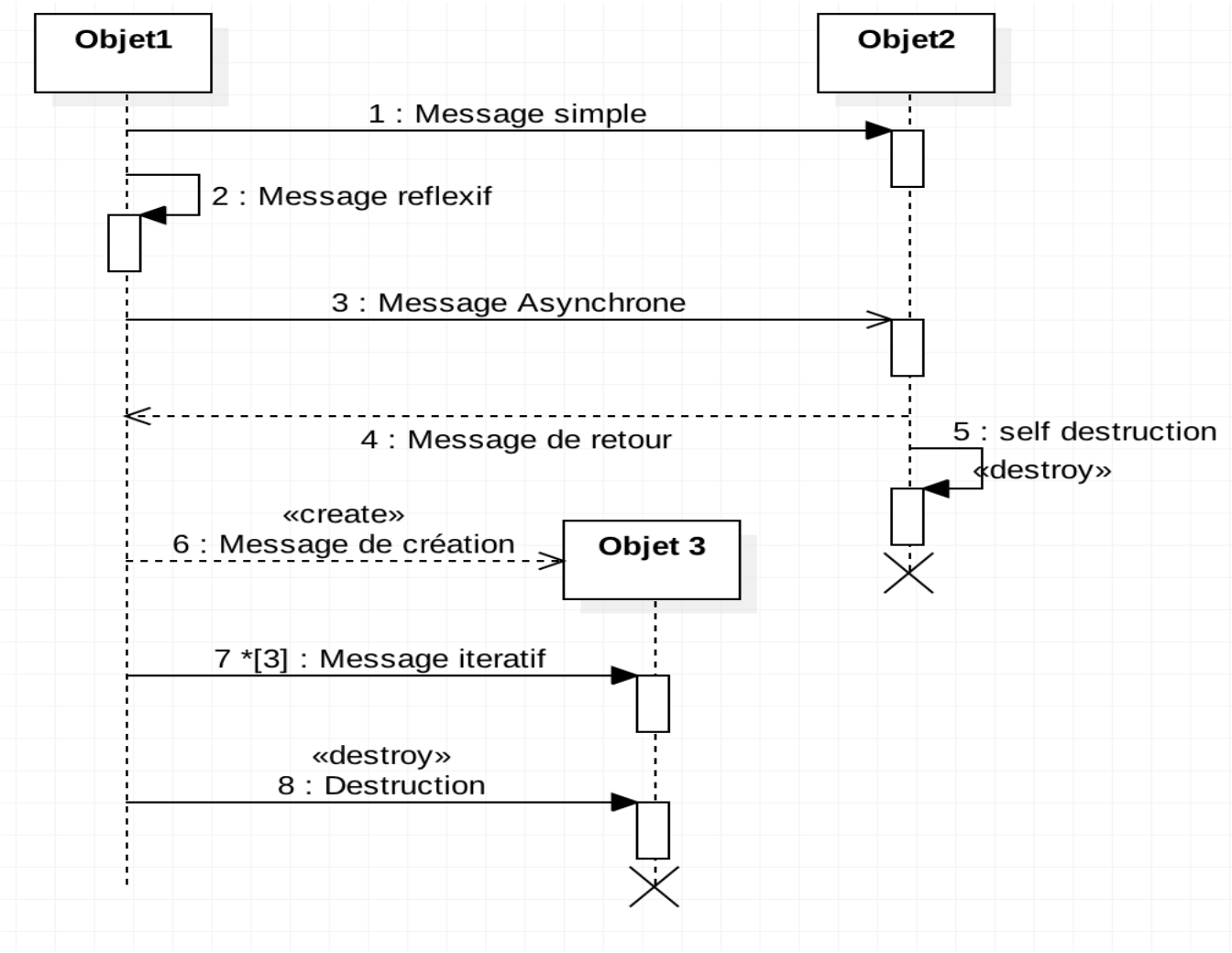
En faisant pointer un message de création sur le rectangle symbolisant l'objet dans le diagramme.

- **Destruction**
  - Représenté par une croix à la fin de la ligne de vie
  - Un objet peut s'auto détruire
  - Un objet peut se détruire par un message pointé sur la croix

Le point de contrôle

- Période pendant laquelle un objet effectue une action
- La mention du point de contrôle est facultative
- Il est modélisé graphiquement par un rectangle sur la ligne de vie de l’objet

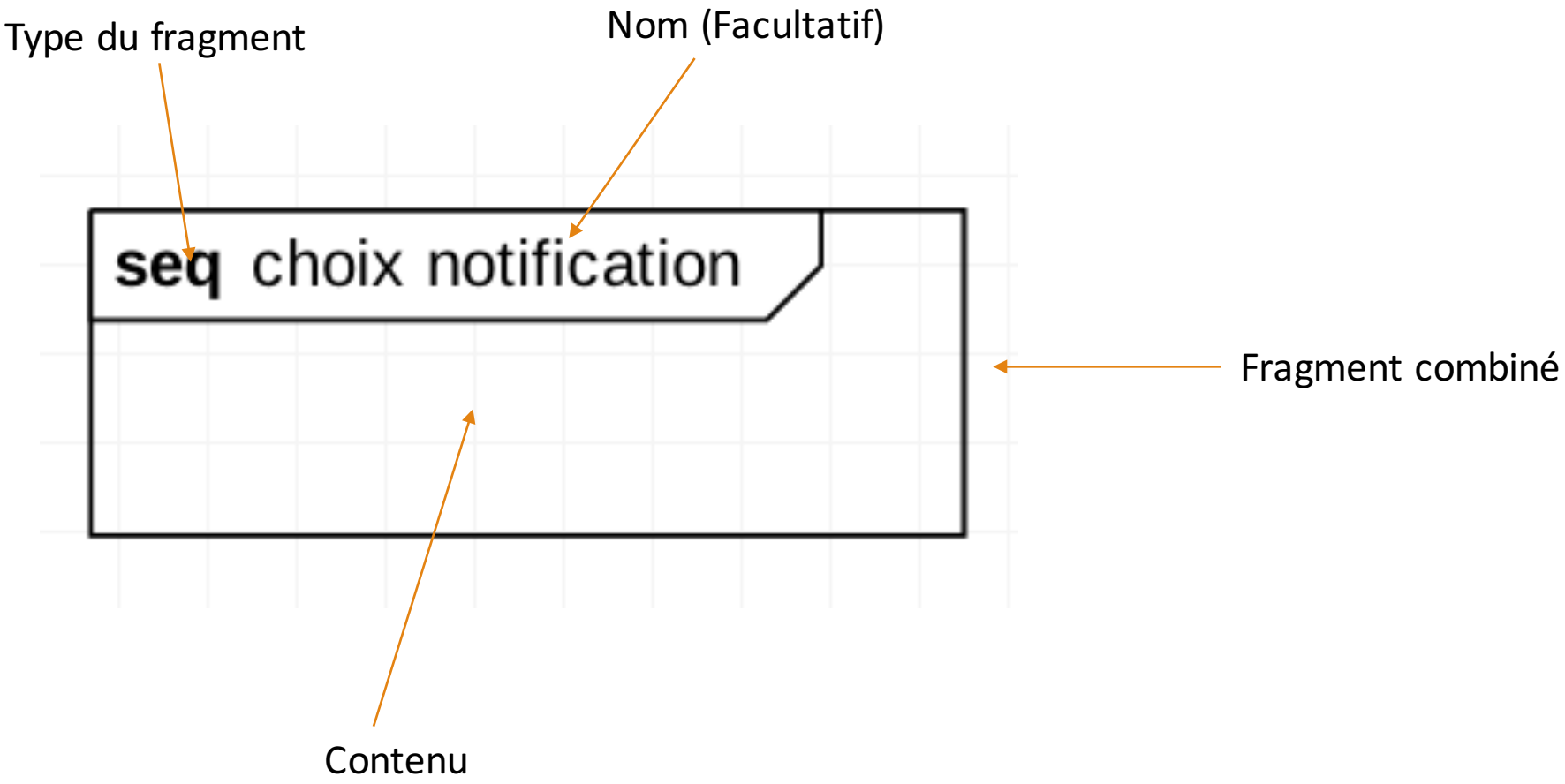
Révision


Objet1
Objet2
1 : Message simple
2 : Message reflexif
3 : Message Asynchrone
4 : Message de retour
5 : self destruction
«destroy»
«create»
6 : Message de création
Objet 3
7 *[3] : Message iteratif
«destroy»
8 : Destruction

Fragments

- Ils permettent de découper un diagramme de séquence en unité simple
- Les fragments sont représentés dans des rectangles
- Le type du fragment est précisé dans un pentagone à l'intérieur du rectangle

Fragment - Représentation



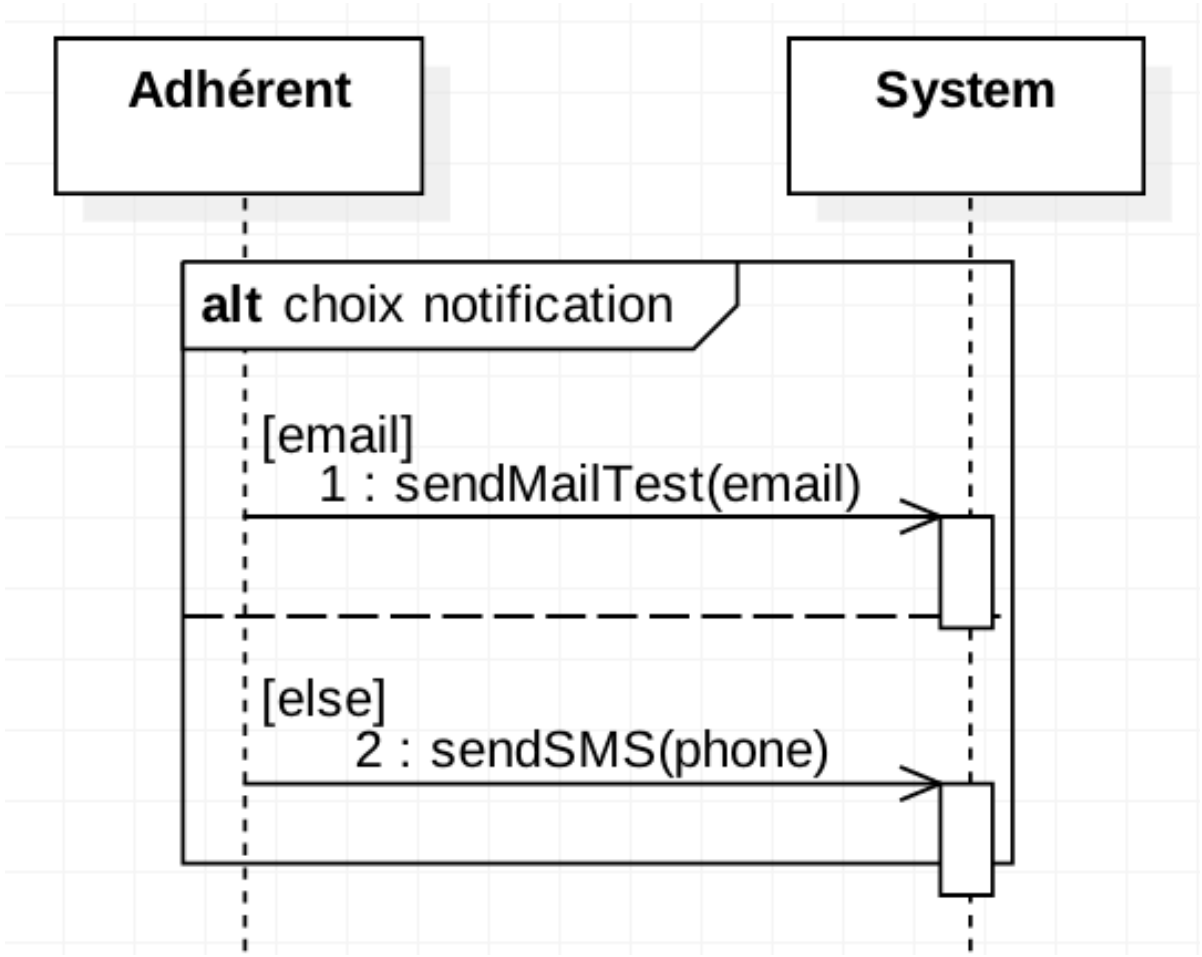
Types de fragments

- alt : Opérateur conditionnel
- loop : Opérateur d'itération
- opt : Opérateur optionnel (facultatif)
- break : Si exécuté, Le reste de la séquence est abandonné
- par : Exécution en parallèle
- seq : Message d'une même ligne de vie s'execute sequentiellement

Opérateur alt

- Modélise des alternatifs (des choix)
- Se base sur des conditions
- Les conditions sont spécifiés entre []
- Les alternatives sont spécifiées dans des zones en pointillés
- Une clause [else] peut être spécifiée

Opérateur alt - Exemple

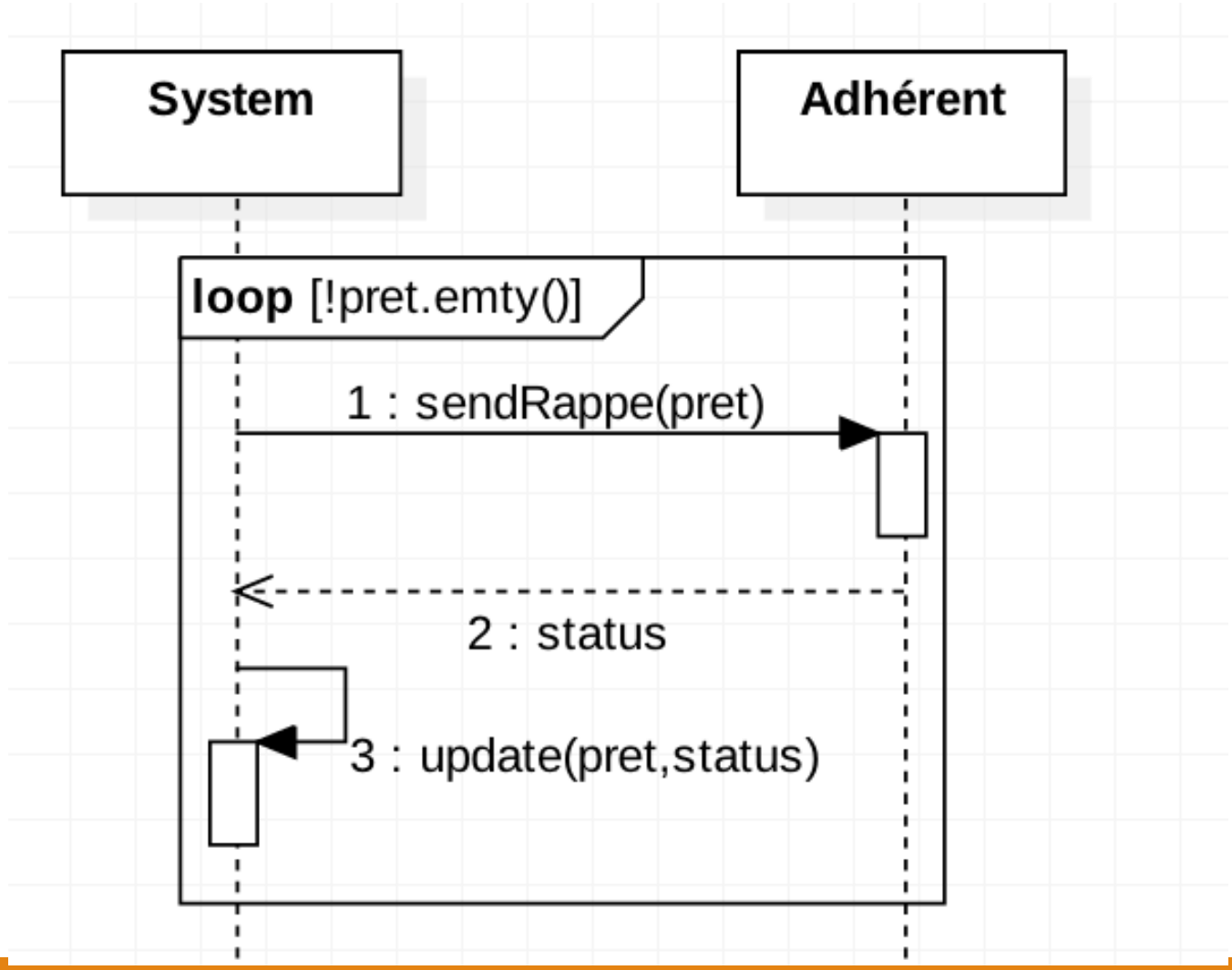


Opérateur loop

- Modélise les itérations (répétitions)

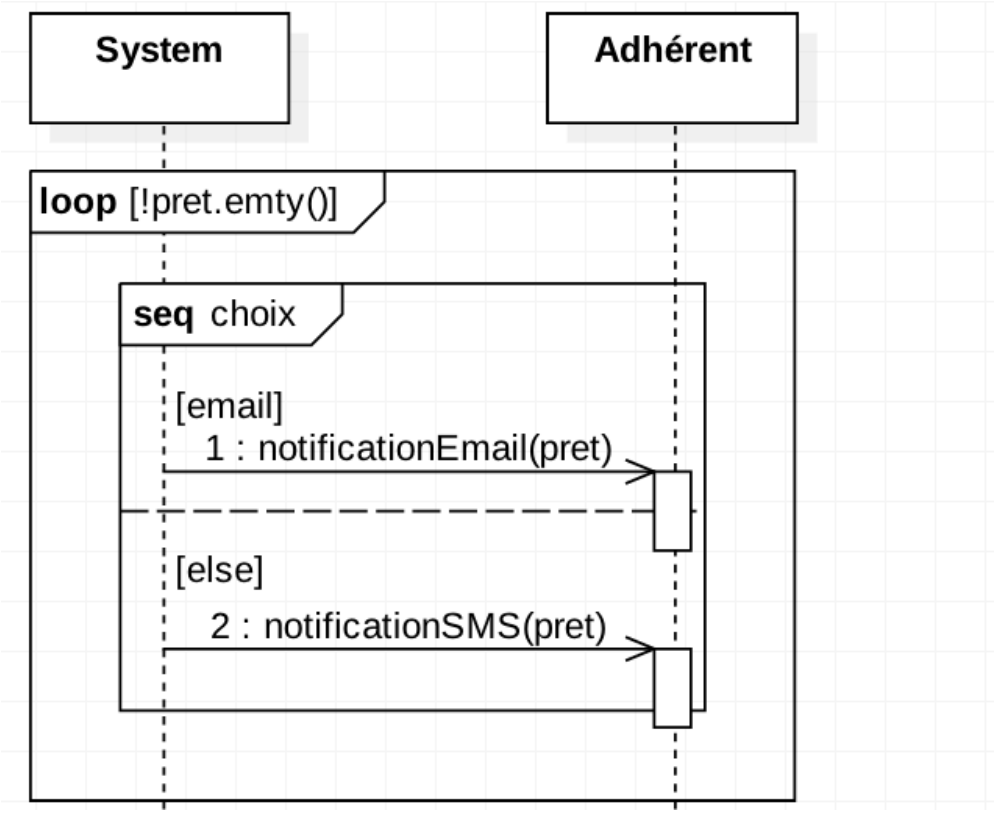
- La séquence d’activité se répète tant que la condition d’arrêt n’est pas vérifiée

Opérateur loop - Exemple



Opérateurs

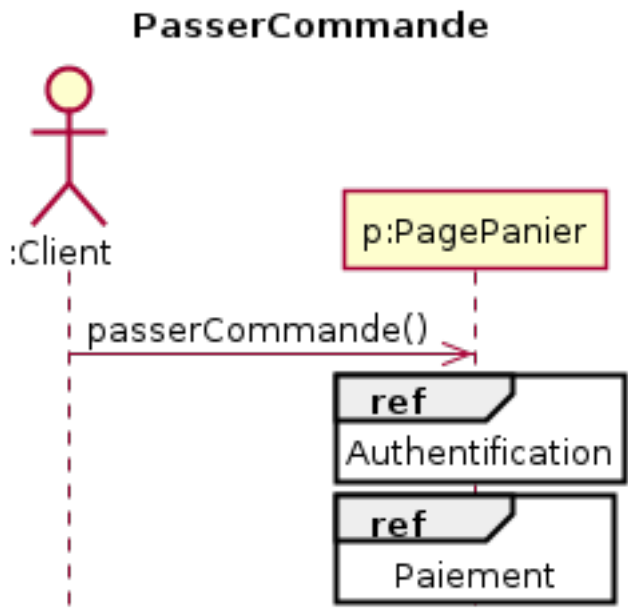
Il est possible de combiner les opérateurs



Réutilisation des séquences

- Il est possible de réutiliser des séquences déjà définies
- Le fragment combiné « ref » permet de faire référence à une séquence déjà définie
- Permet de définir des diagrammes de séquences simple et plus lisibles

Réutilisation des Séquences



Exercice

Concevoir un diagramme de séquence d'une transaction dans un distributeur.

- Les cartes bancaires peuvent gérer deux types de comptes :
  - Cash
  - Crédit
- Une transaction peut être : un retrait ou un dépôt
- Un dépôt peut être de deux types
  - Dépôt de chèque
  - Dépôt de cash
- Afficher une pub au moment de la validation avec le système de la banque
- On peut se tromper de codes trois fois au maximum
  - Après 3 tentatives, la carte est gardée par la machine