UML : diagramme de séquences

Achref El Mouelhi

Docteur de l'université d'Aix-Marseille
Chercheur en programmation par contrainte (IA)
Ingénieur en génie logiciel

elmouelhi.achref@gmail.com

UNIFIED
MODELING
LANGUAGE

Plan

UML

Diagramme de séquences ?

- Un diagramme dynamique d'**UML**
- Représentant l'interaction entre les acteurs et le système selon un ordre chronologique
- Pouvant servir à détailler un cas d'utilisation
- Le temps s'écoule selon une dimension verticale (non graduée) de haut en bas
- Les objets sont organisés horizontalement
- L'échange entre les différents éléments se fait moyennant des messages

UML

Mots-clés associés

- Ligne de vie (et activité)
- Message
- Fragment combiné

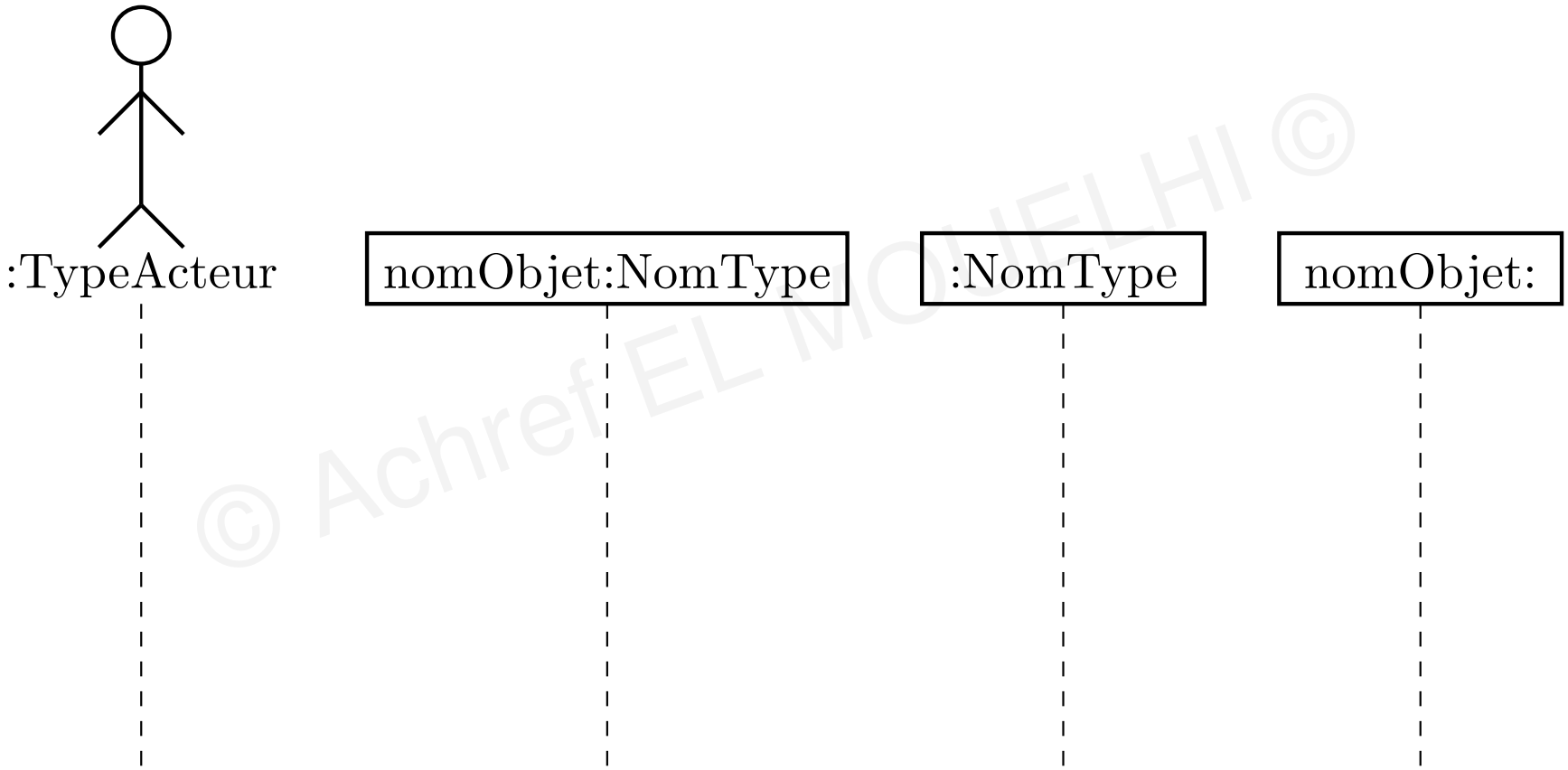
UML

Ligne de vie

- Rectangle au sommet contenant le nom de type (objet, acteur...) + ligne verticale pointillée
- Permettant de définir la zone d'envoi et de réception de messages échangés avec les autres lignes de vie

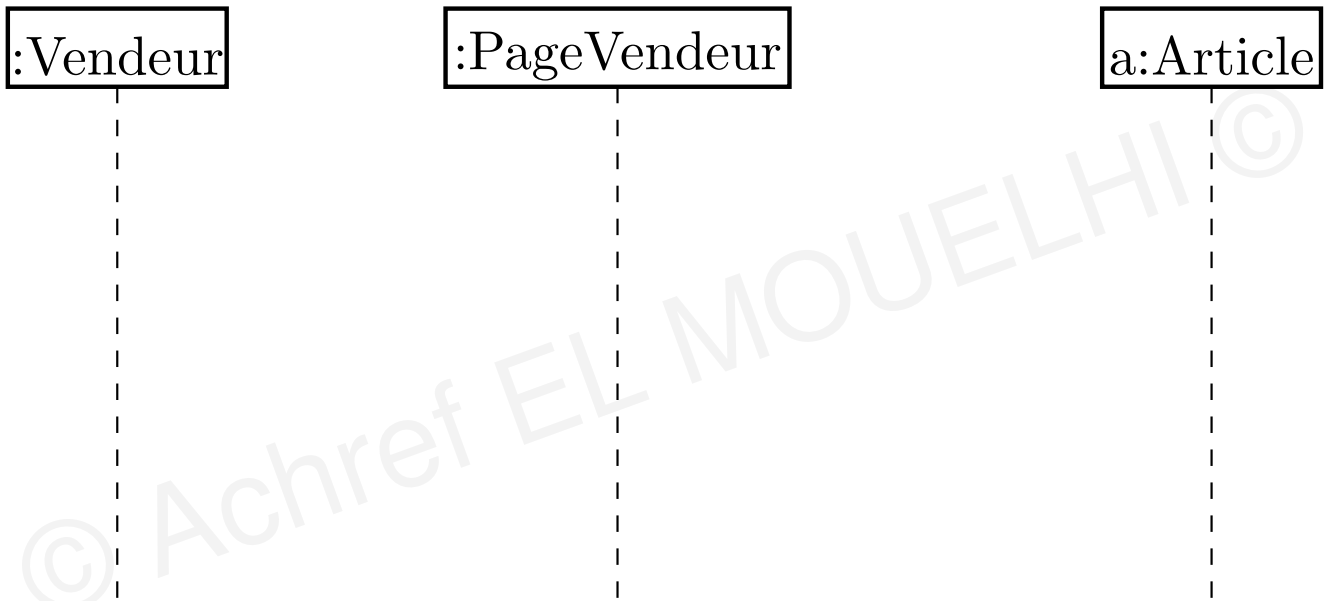
UML

Plusieurs représentations possibles en UML



UML

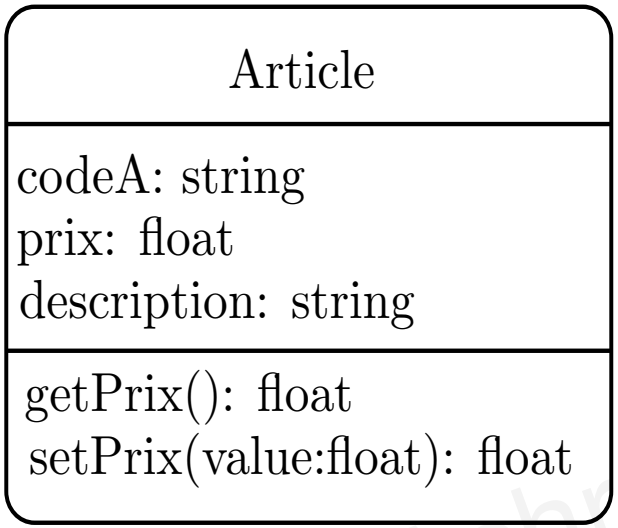
Exemple



L'acteur `Vendeur` peut être représenté comme un objet.

UML

Étant donné l'exemple suivant



classe `Livre`

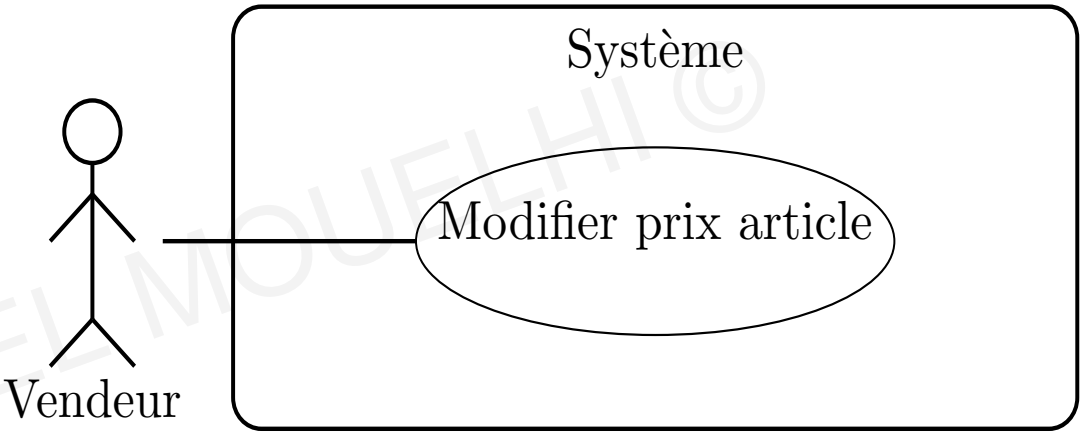
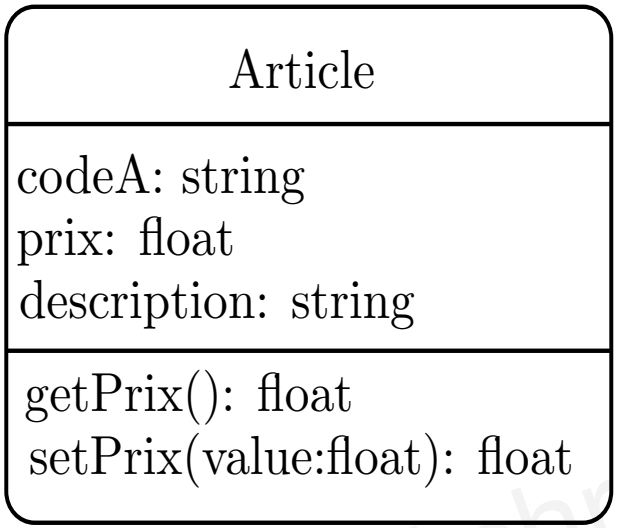


diagramme de cas d'utilisation

UML

Étant donné l'exemple suivant



classe `Livre`

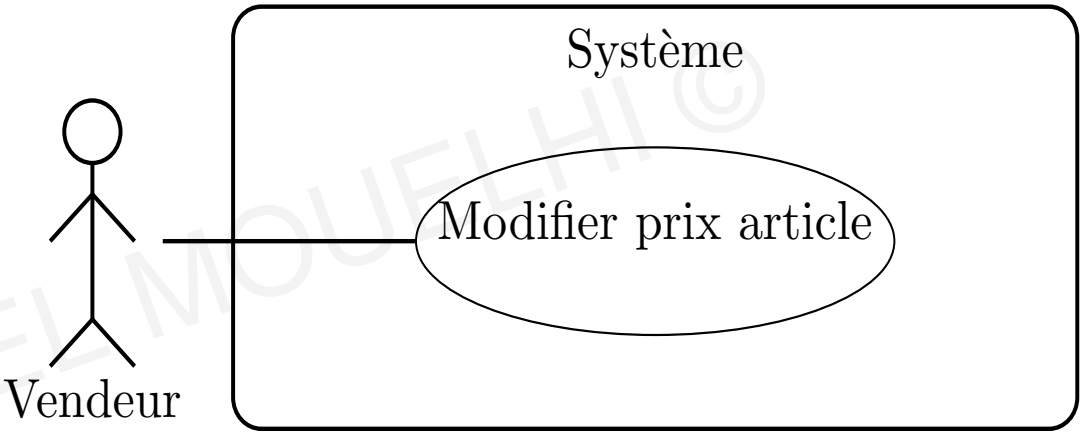


diagramme de cas d'utilisation

Objectif

Détailler le cas d'utilisation `Modifier prix article` avec un diagramme de séquences.

UML

Déroulement

- Le vendeur va utiliser la page vendeur pour appeler une méthode `changerPrix(a,p)` : `a` étant un objet de type `Article` et `p` le nouveau prix
- Pour modifier le prix de cet article, la page vendeur fera appel à la méthode `setPrix(p)` de l'objet `a`

UML

Déroulement

- Le vendeur va utiliser la page vendeur pour appeler une méthode `changerPrix(a,p)` : `a` étant un objet de type `Article` et `p` le nouveau prix
- Pour modifier le prix de cet article, la page vendeur fera appel à la méthode `setPrix(p)` de l'objet `a`

Comment modéliser tout ça ?

- Les échanges entre deux lignes de vie se fait via des messages

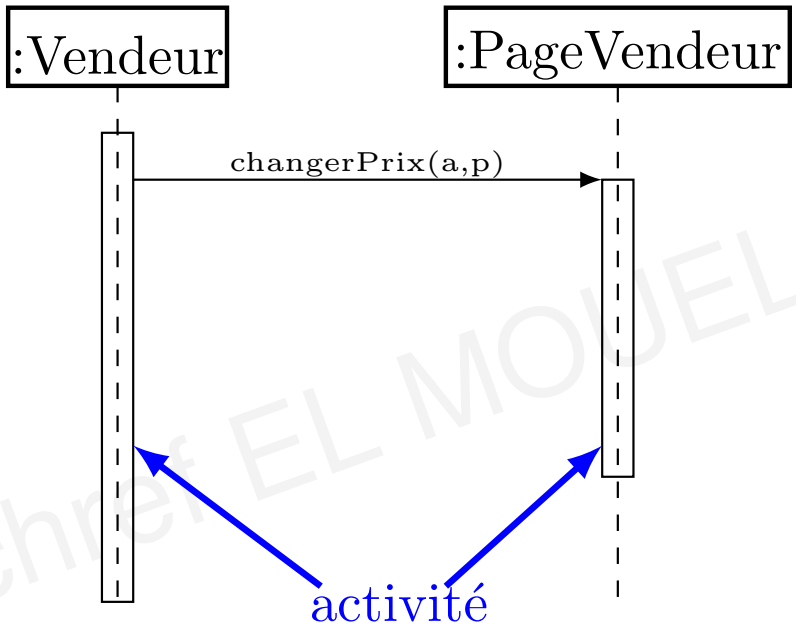
UML

Message

- Représenté par une flèche
- Définit une communication entre deux lignes de vie
- Déclenche une activité sur une ligne de vie
- Possède un nom (méthode, signal...)
- Peut avoir des paramètres

UML

Voici comment modéliser les messages en UML

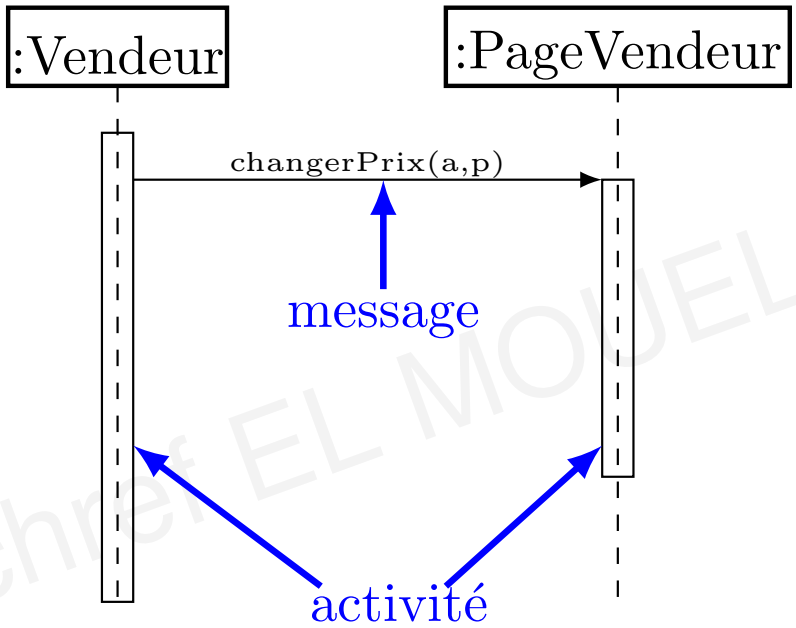


Remarque

La réception d'un message a déclenché une activité dans l'objet `:PageVendeur`

UML

Voici comment modéliser les messages en UML



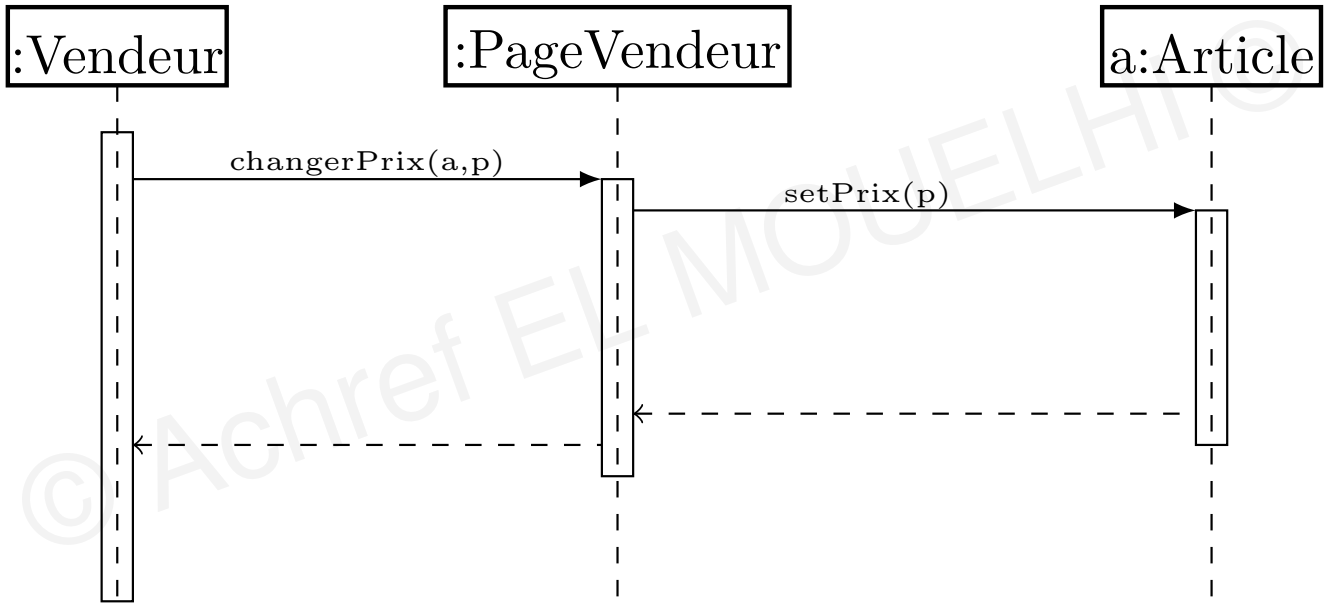
Remarque

La réception d'un message a déclenché une activité dans l'objet `:PageVendeur`

UML

Le diagramme de séquence correspondant au cas d'utilisation

`Modifier prix article`

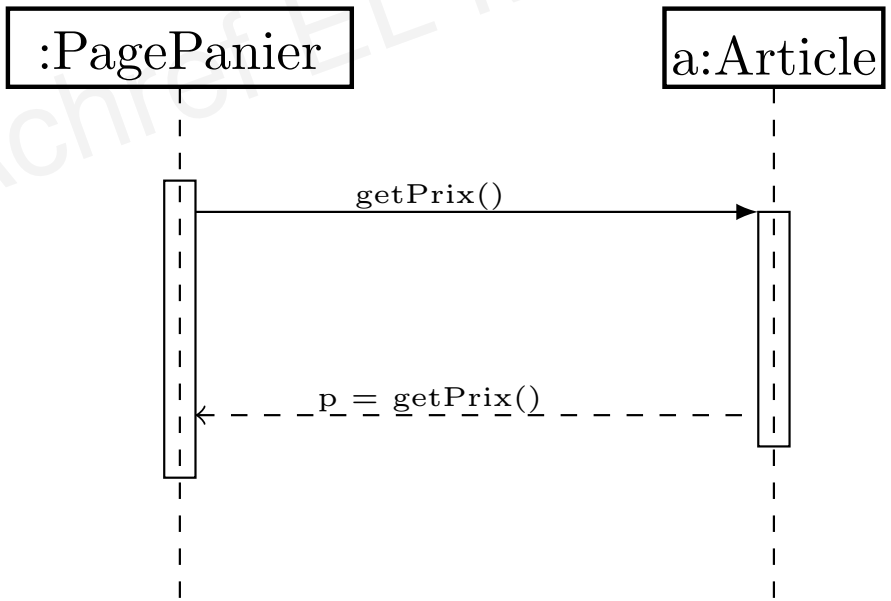


La ligne en pointillé correspond à la réponse au message synchrone

UML

Message Synchrone

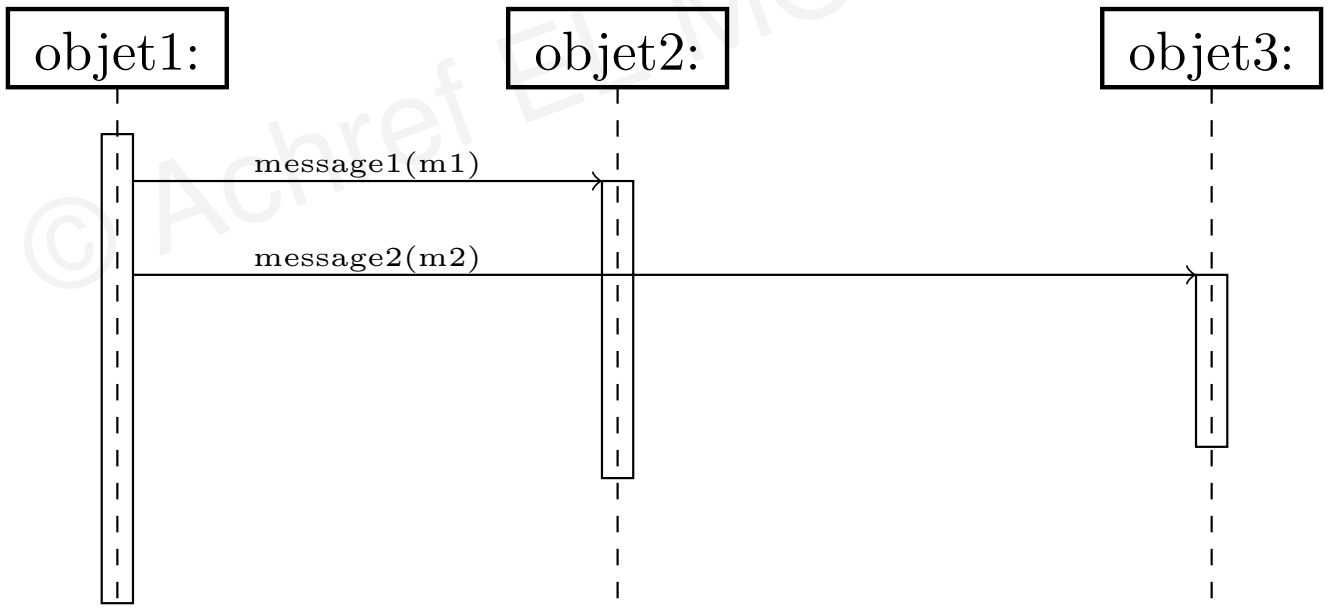
- Bloque l'émetteur
- Le récepteur rend la main à l'émetteur par un message de retour (sa représentation est optionnelle)
- Peut spécifier le résultat de la méthode invoquée



UML

Message Asynchrone

- Ne bloque l'émetteur
- Représentée graphiquement par une flèche continue à extrémité ouverte



UML

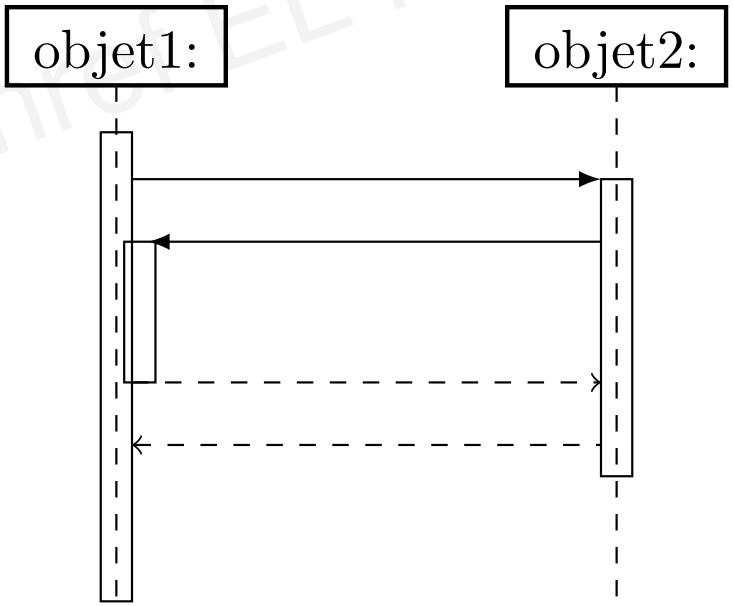
Explication

- `:Objet1` a envoyé un premier message à `:Objet2` qui a déclenché une activité chez ce dernier
- Ensuite, il a envoyé un deuxième message à `:Objet3` sans attendre ni la réponse de `:Objet2` ni la fin de sans activité

UML

Qu'arrive t-il en cas de message concomitants ?

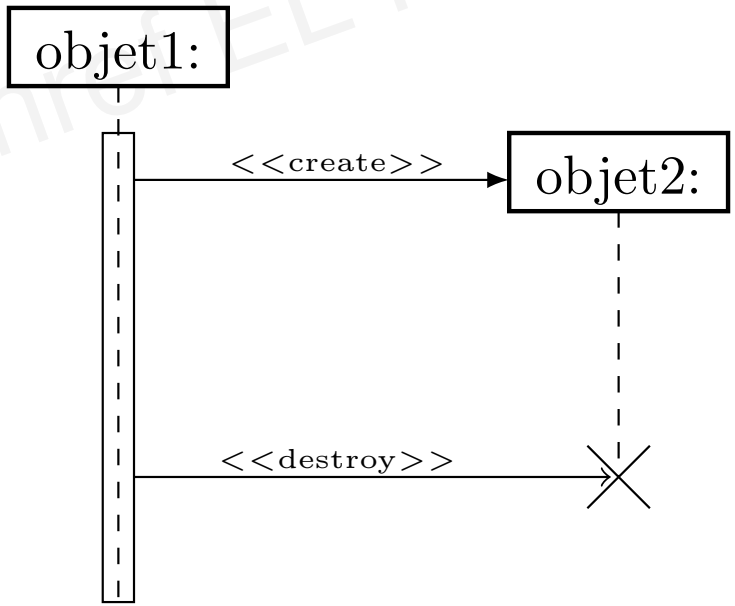
- Si un objet est actif, la réception d'un message implique la création d'une nouvelle activité
- La nouvelle activité sera représentée par un rectangle chevauchant le premier



UML

Un objet peut créer et détruire un deuxième objet

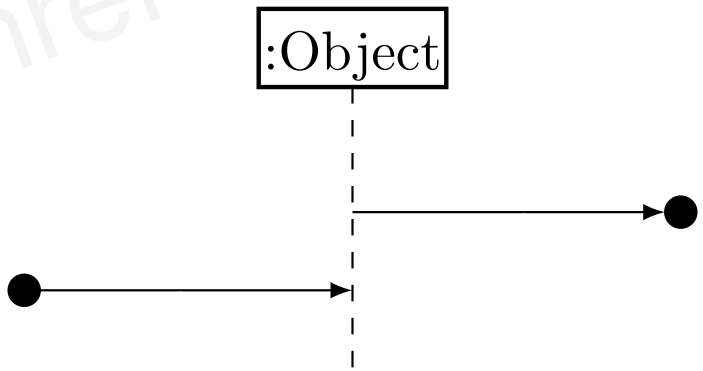
- La création d'un objet se fait par l'envoi d'un message avec le stéréotype `<<create>>`
- La destruction se fait par l'envoi d'un message avec le stéréotype `<<destroy>>`



UML

Qu'est ce qu'un message complet, perdu et trouvé ?

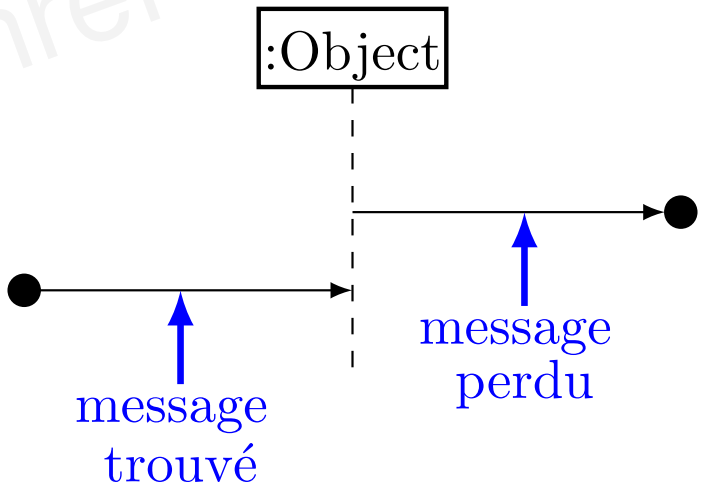
- Complet : évènements d'envoi et de réception connus
- Perdu : évènement d'envoi connu mais pas la réception
- Trouvé : évènement de réception connu mais pas l'envoi



UML

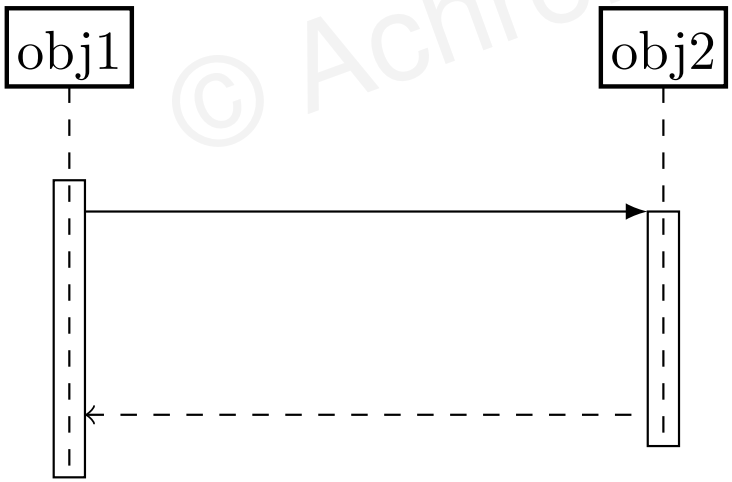
Qu'est ce qu'un message complet, perdu et trouvé ?

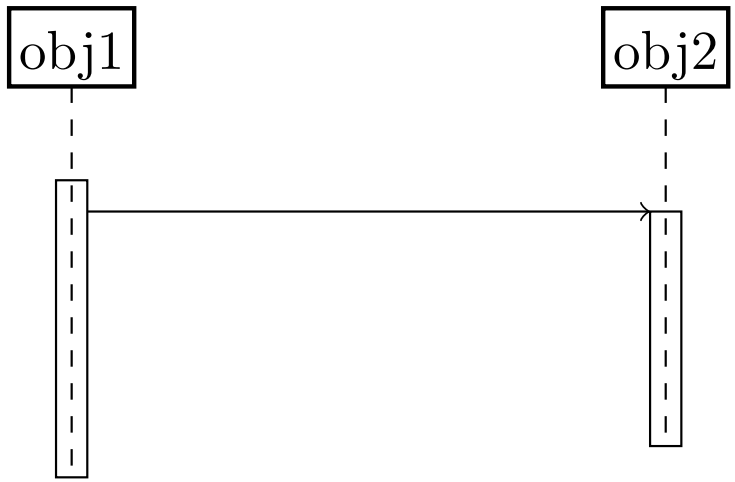
- Complet : évènements d'envoi et de réception connus
- Perdu : évènement d'envoi connu mais pas la réception
- Trouvé : évènement de réception connu mais pas l'envoi



Exercice

- Lequel de ces deux diagrammes correspond à l'envoi d'un message asynchrone ?
- Ne pas représenter un message de retour synchrone signifie t-il que le type de retour est `void` ?
- `obj1` est un acteur, car seuls les acteurs peuvent envoyer de messages
- `obj1` est le type de l'objet qui envoi le message
- La ligne de vie de `obj1` dans le premier diagramme prendra fin après sa période d'activité ?





UML

Question

Comment faire pour répéter une partie du diagramme de séquence ou pour ajouter un bloc conditionnel ?

UML

Question

Comment faire pour répéter une partie du diagramme de séquence ou pour ajouter un bloc conditionnel ?

Solution

Utiliser les fragments combinés

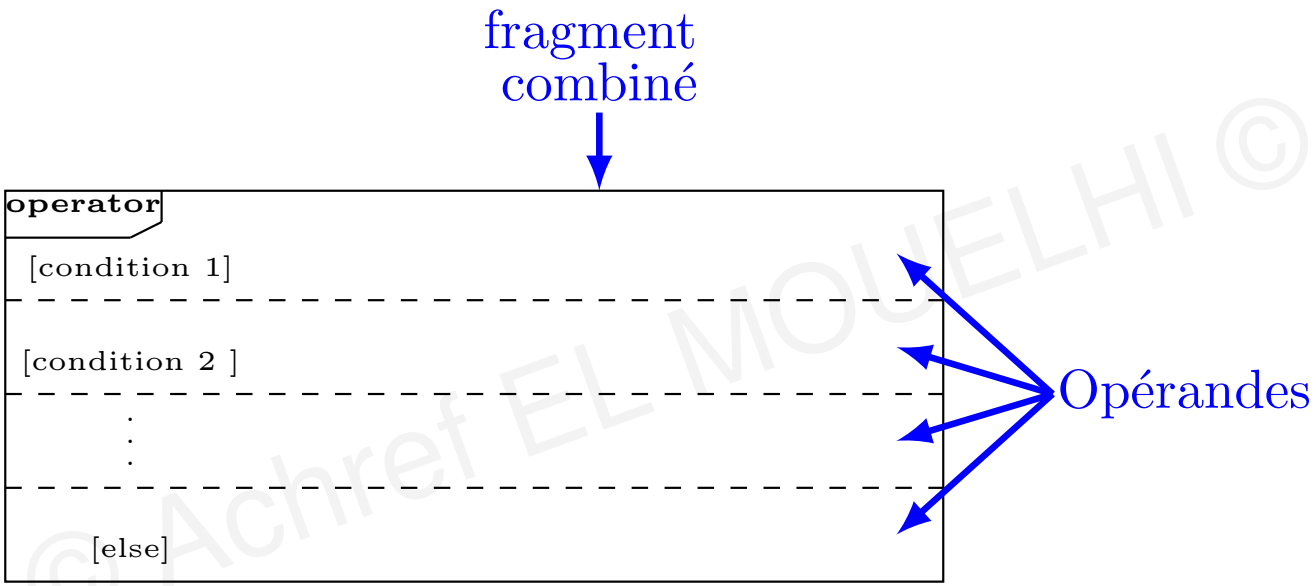
UML

Fragment combiné

- Fragment d'interaction
- Utilisant un opérateur [et des opérandes]
- Exemple d'opérateurs :
  - `alt` : opérateur conditionnel à plusieurs opérandes (équivalent d'un bloc `if` ... `else if` ... `else` en programmation)
  - `opt` : opérateur conditionnel à une seule opérande (`if` sans `else`)
  - `loop` : opérateur répétitif acceptant au max deux valeurs min et max
  - `par` : opérateur d'exécution parallèle
  - ....

UML

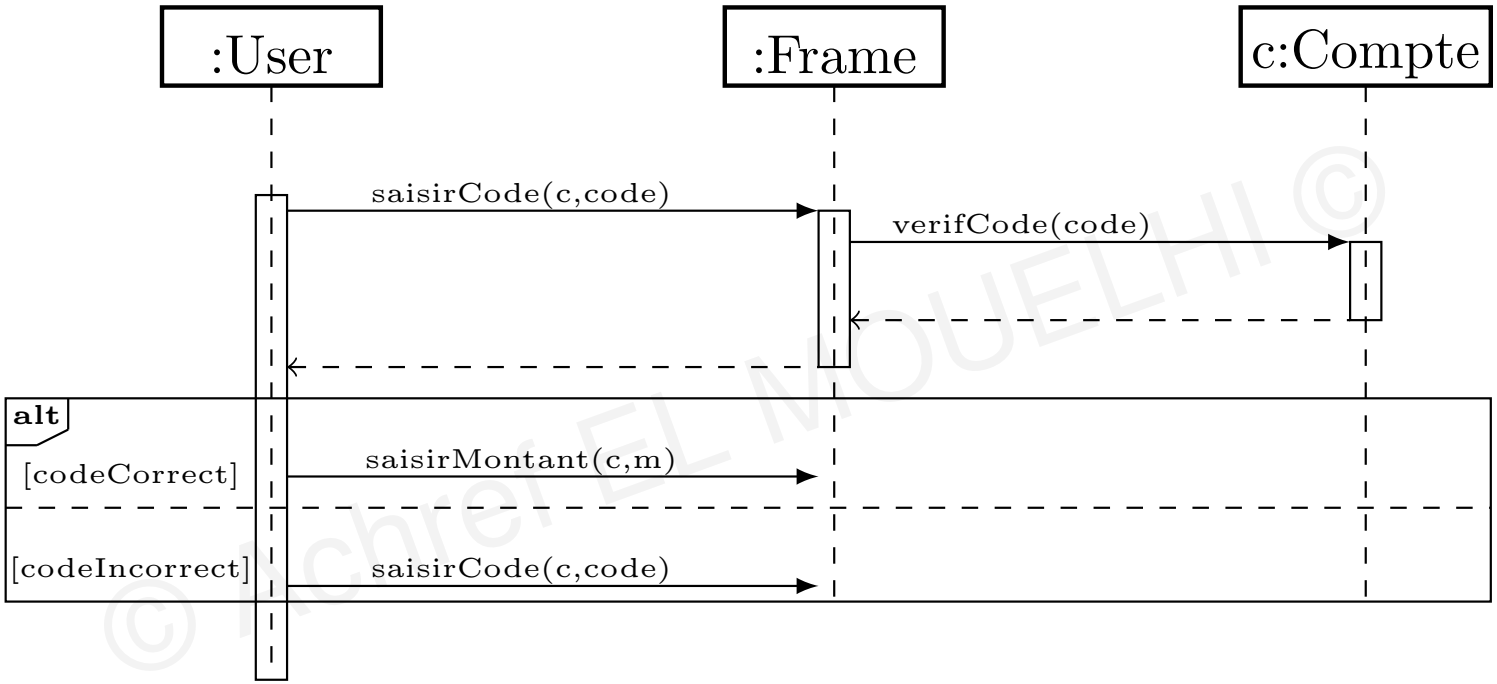
Structure d'un fragment combiné



Certains fragments n'ont pas besoin d'une condition ni de plusieurs opérandes.

UML

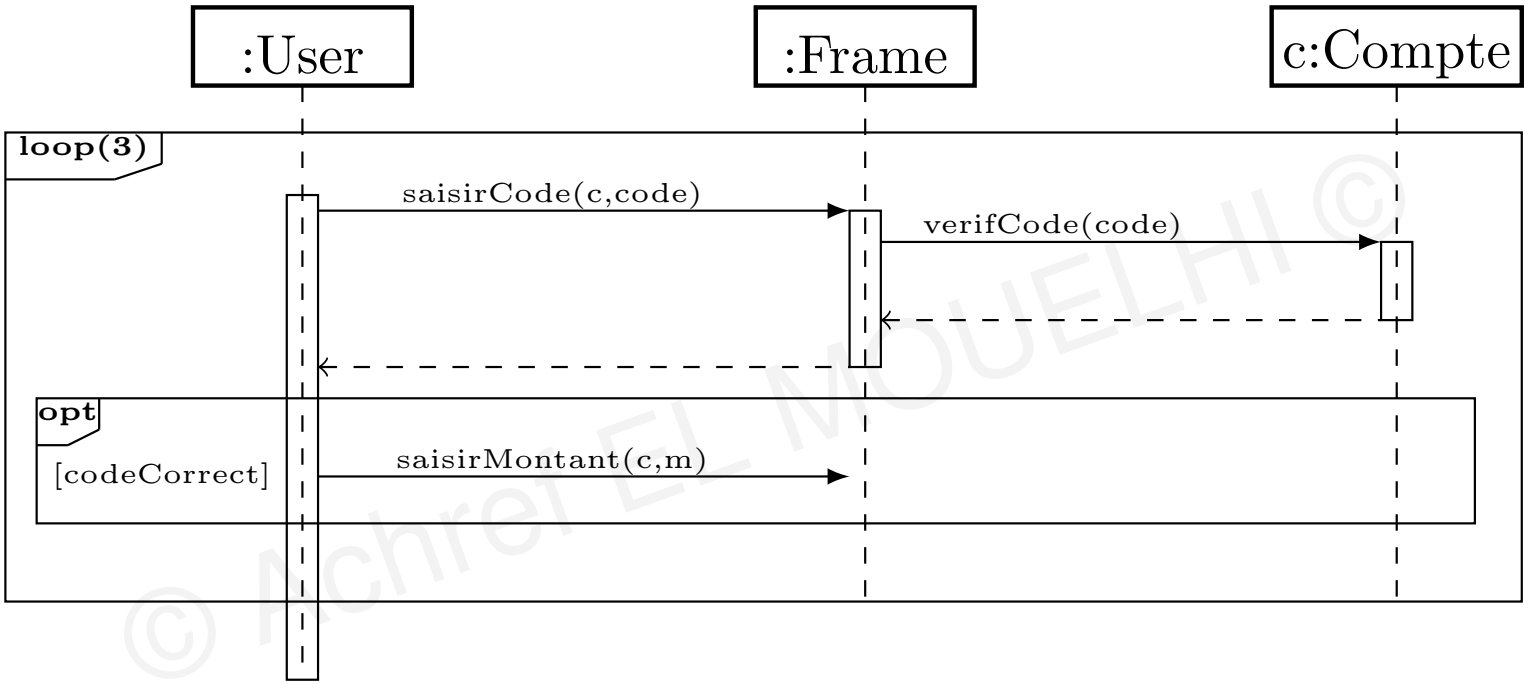
Exemple de `alt` avec un distributeur de billets



Si le code est incorrect , il faut recommencer (c'est répétitif) ⇒ On peut utiliser l'opérateur `loop`.

UML

Exemple de `alt` avec un distributeur de billets



- En ajoutant le paramètre (3) pour `loop`, on a défini un nombre exact d'itération.
- Si on ne précise aucun paramètre, alors il s'agit d'une boucle infinie.

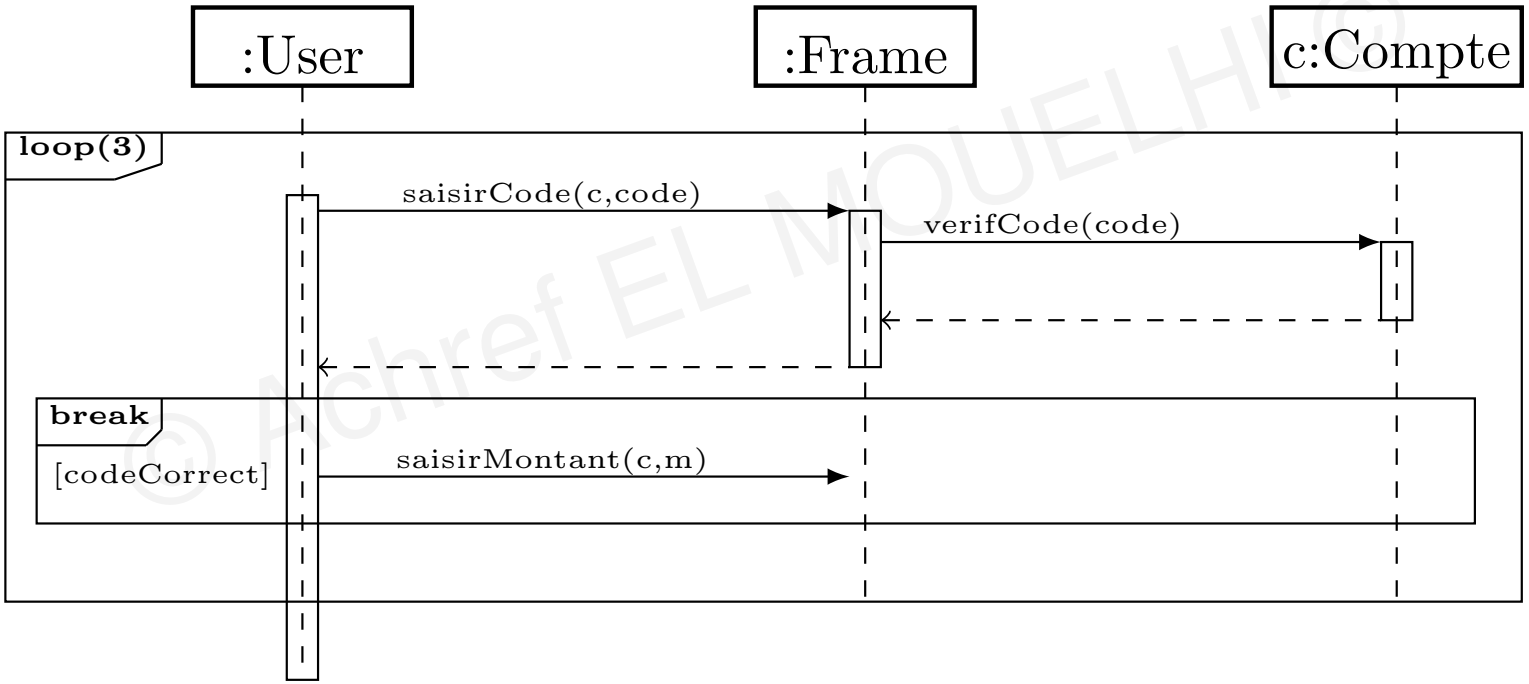
UML

Remarque

La solution précédente oblige l'utilisateur à refaire l'opération 3 fois même s'il saisit correctement le code.

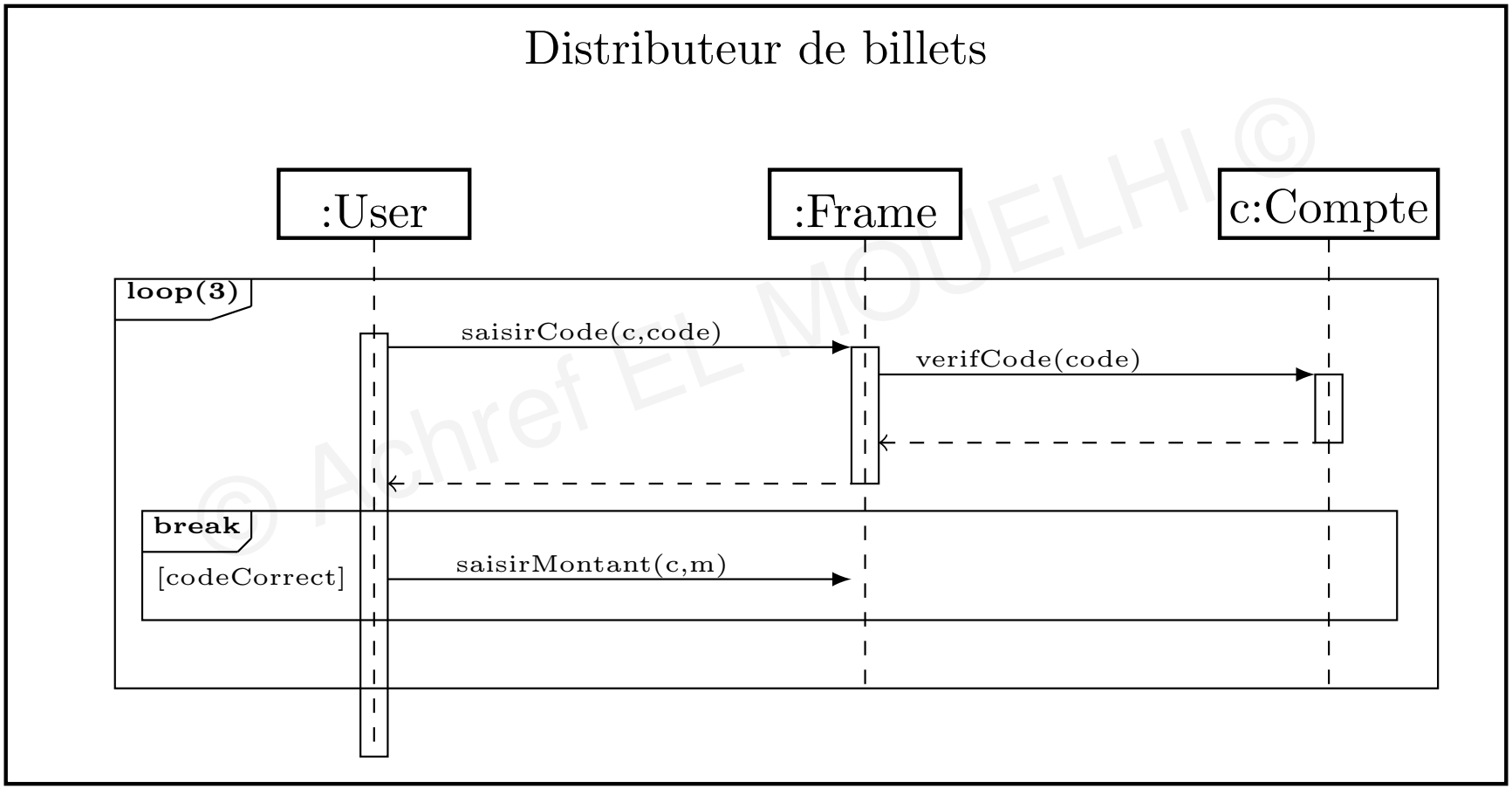
UML

Pour corriger ça, on peut utiliser un fragment `break` qui sera exécuté avant de quitter la boucle



UML

On peut aussi définir le contexte comme un fragment principal



UML

Exercice

Compléter le diagramme précédent en étudiant les différents cas possibles pour le retrait d'argent (solde insuffisant, possibilité d'imprimer un ticket...).