Chapitre 2 :

Diagramme de séquence

Plan

- Introduction
- Interactions et partenaires d'interaction
- Messages
- Fragments combinés
  - Branches et boucles
  - Concurrence et ordre
  - Filtres et assertions
- Autres éléments de langage
- Autres types de diagrammes d'interaction

Introduction

- **Les diagrammes de cas** d'utilisation modélisent à QUOI sert le système, en organisant les interactions possibles avec les acteurs.
- **Les diagrammes de séquence** permettent de décrire COMMENT les éléments du système interagissent entre eux et avec les acteurs pour réaliser l'objectif du système.
- Diagramme de séquence
  - Diagramme d'interaction
  - fait partie des diagrammes comportementaux (dynamiques)

Introduction

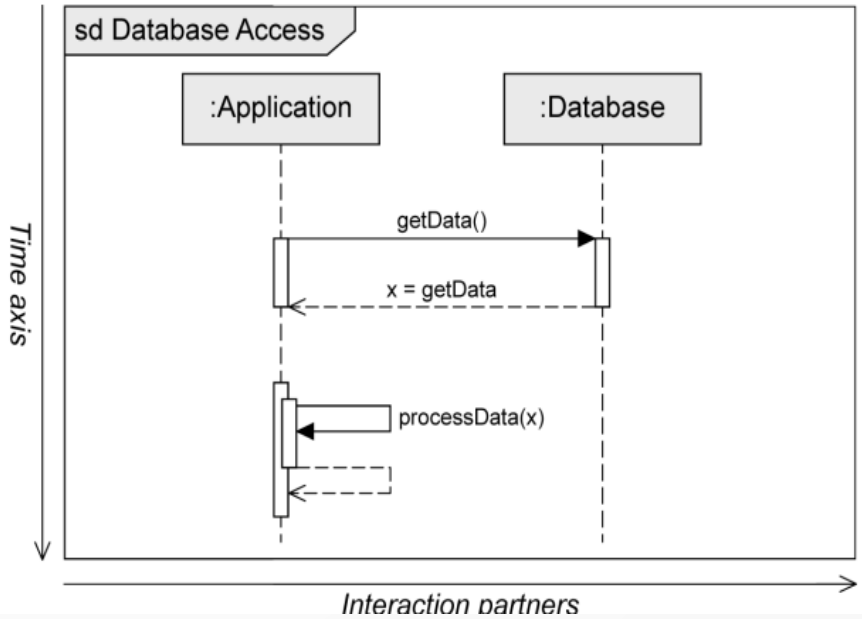
- Modélisation du comportement inter-objets
  - Interactions entre objets
- Interaction
  - Spécifie comment les messages et les données sont échangés entre les partenaires d'interaction
- Partenaires d'interaction
  - Humain (conférencier, administrateur, …)
  - Non humain (serveur, imprimante, logiciel exécutable, …)
- Exemples d'interactions
  - Conversation entre personnes
  - Échange de messages entre des humains et un système logiciel
  - Protocoles de communication
  - Séquence d'appels de méthode dans un programme
  - …

Diagramme d'interaction

- Utilisé pour spécifier les interactions
- Modélisation de scénarios concrets
- Décrit les séquences de communication à différents niveaux de détail
- Les diagrammes d'interaction montrent les éléments suivants :
  - Interaction d'un système avec son environnement
  - Interaction entre les parties du système afin de montrer comment un cas d'utilisation spécifique peut être mis en œuvre
  - Communication au niveau de la classe (appels d'opérations, comportement inter-objets)

Diagramme de séquence

- Diagramme d'interaction qui met en évidence l'ordre chronologique des messages.
- Représente un ensemble d'objets ainsi que les messages envoyés et reçus par ces objets
- Diagramme bidimensionnel
  - Axe horizontal :
    - partenaires d'interaction impliqués (les objets)
  - Axe vertical :
    - ordre chronologique de l'interaction

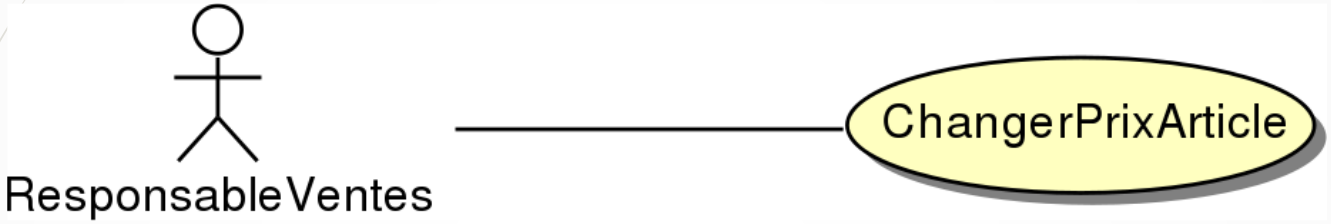


Utilisation des diagrammes de séquence

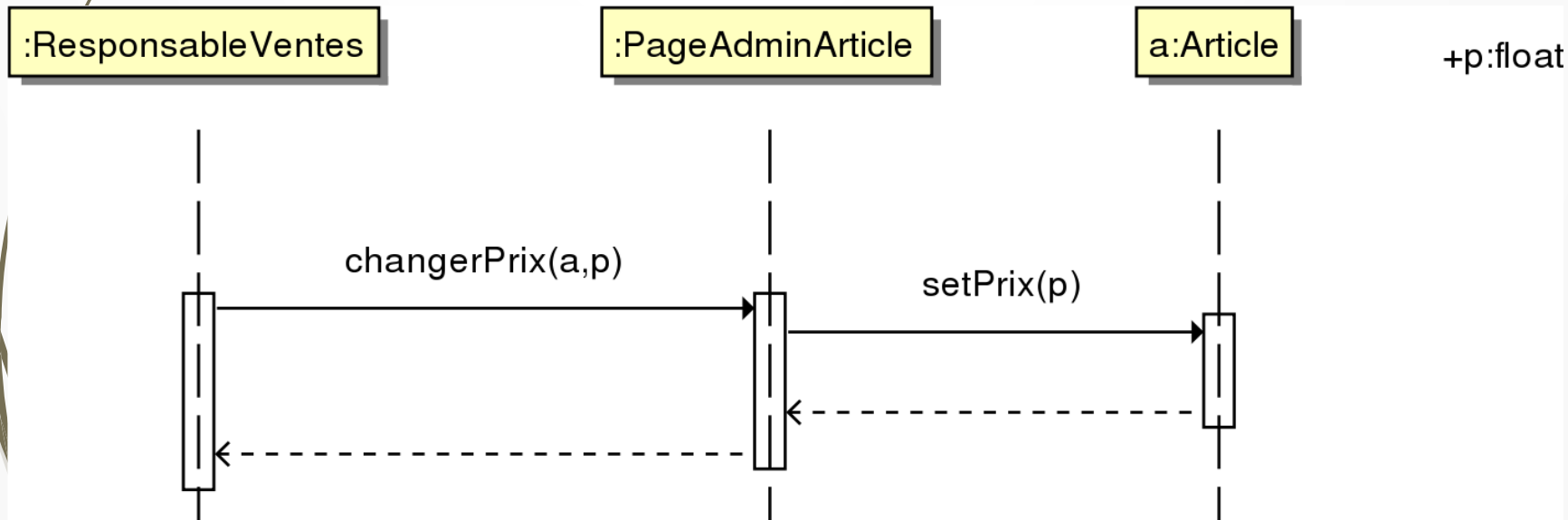
- Pendant la capture des besoins (système = boite noire) :
  - Interactions entre acteurs et système
  - Diagramme de séquence système (DSS)
  - Décrit les scénarios des cas d'utilisation
- Pendant la conception (système = boite blanche) :
  - Un diagramme de séquence par cas d'utilisation
    - Les classes concernées par le cas sont déterminées
    - Interactions entre les objets concernés par la réalisation du cas
- Elaboration en parallèle avec les diagrammes de classes

Exemple d'interaction

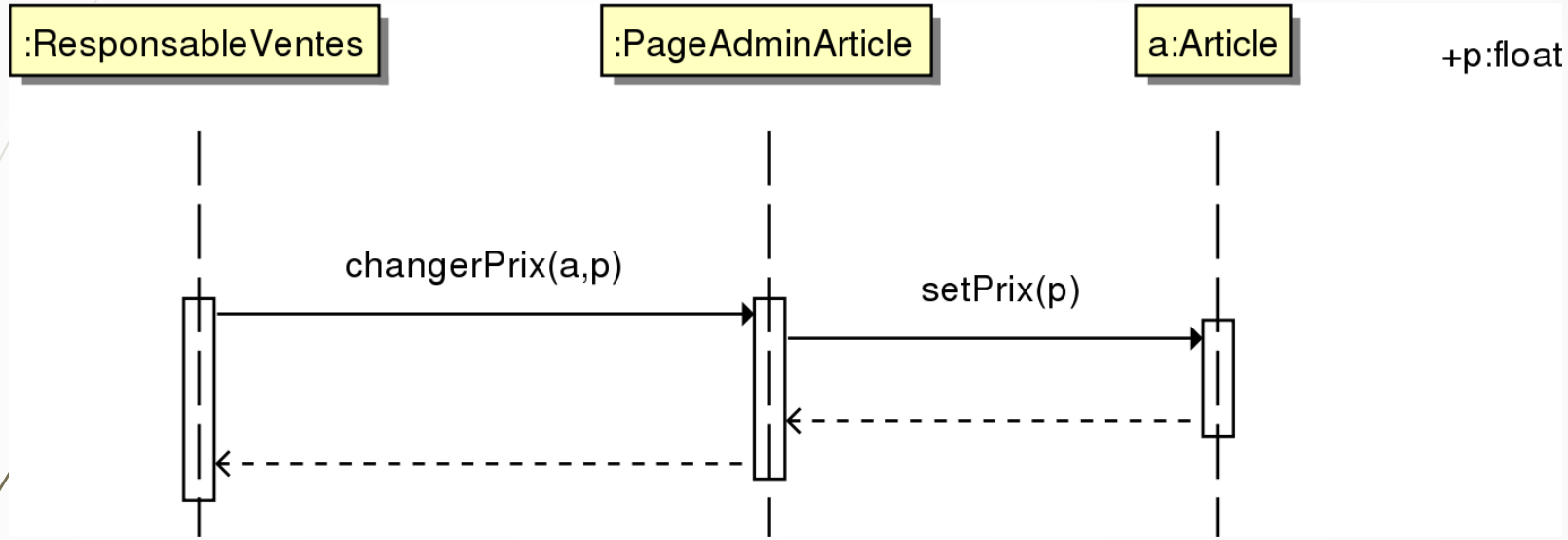
- Cas d'utilisation :



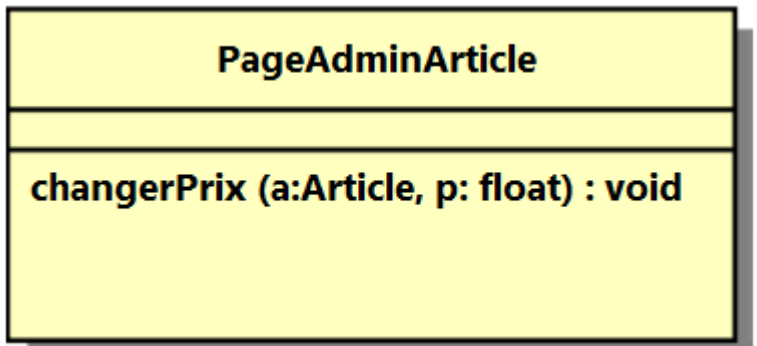
- Diagramme de séquence correspondant :
  - Les classes : ResponsableVente, PageAdminArticle et Article

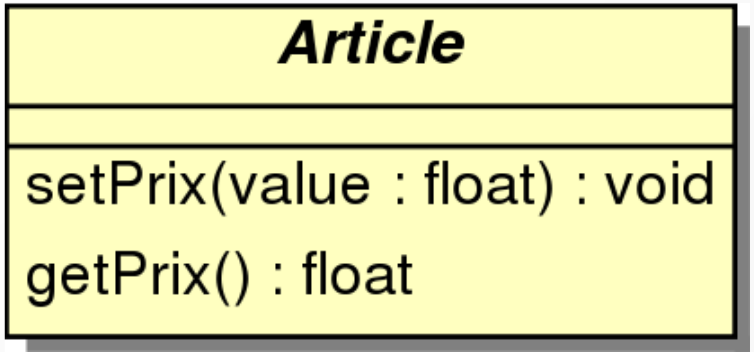


Exemple d'interaction



- Les messages envoyés correspondent à des appels des opérations définies dans le diagramme de classes

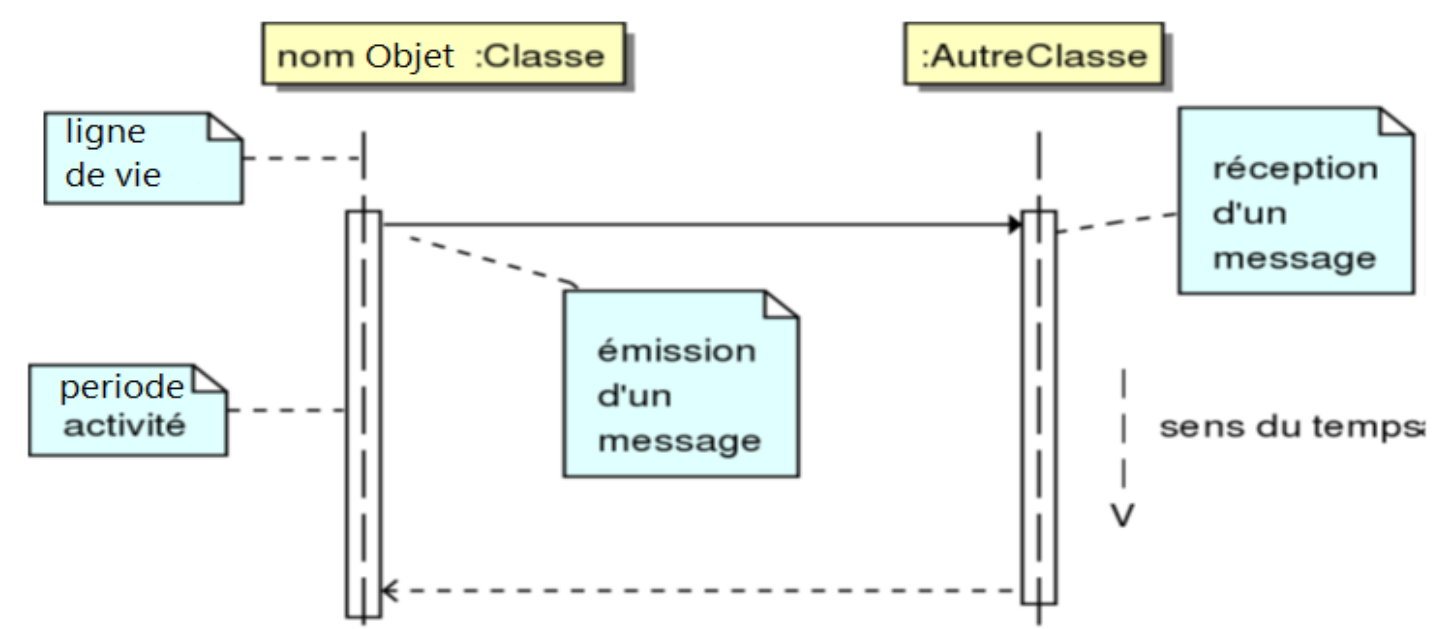




L'objet dans le diagramme de séquence

- Un objet est représenté par
  - un rectangle contenant le nom de l'objet sous la forme

    nomObjet : nomClasse (ou **:nomClasse** seulement)
  - une ligne verticale appelée ligne de vie de l'objet.
  - période d'activité : rectangle vertical sur la ligne de vie
- Le diagramme possède un axe de temps dirigé de haut en bas



Les messages

- Interaction : séquence d'événements
- Le message est défini via un événement d'envoi et un événement de réception

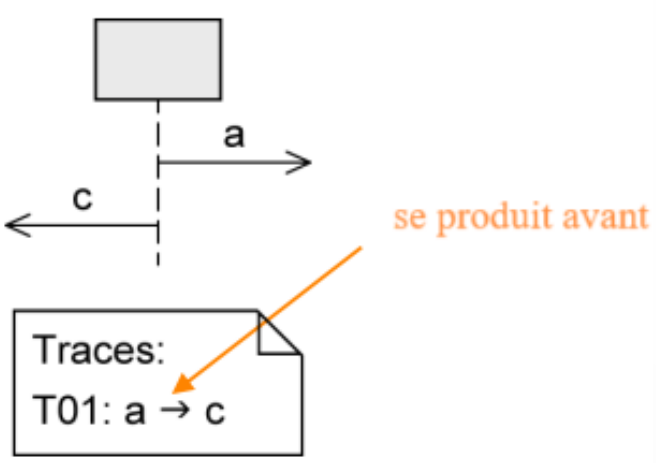
- Un message définit une communication particulière entre deux lignes de vie (objets ou acteurs).
  - Plusieurs types de messages existent :
    - l'invocation d'une opération (appel de méthode) ;
    - la création ou la destruction d'un objet.
    - l'envoi du signal

- La réception des messages provoque une période d'activité marquant le traitement du message (ou l'exécution dans le cas d'un appel de méthode).

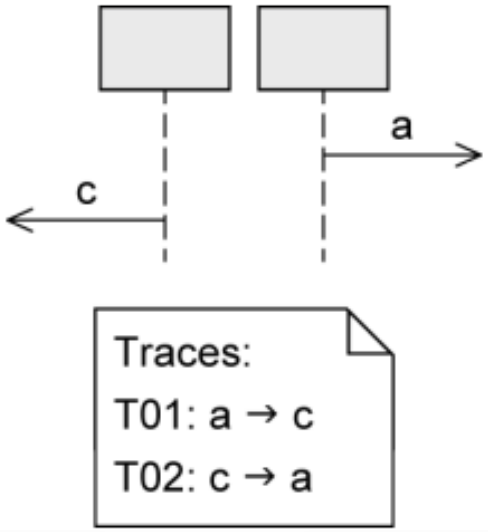
Les messages

- Ordre de messages

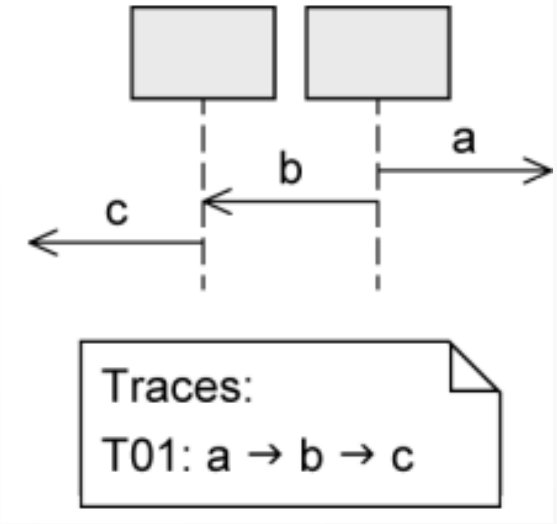
…sur une seule ligne de vie



…sur des lignes de vie différentes

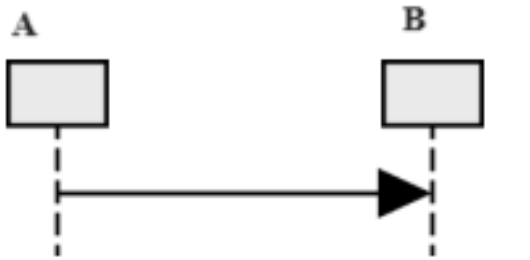


…sur des lignes de vie ≠ qui échangent des messages



Principaux types de messages

- **Message synchrone**
  - L'expéditeur attend jusqu'à ce qu'il reçoive un message de réponse avant de continuer
  - Syntaxe du nom du message : **msg(par1, par2)**
    - **msg** : le nom du message
    - **par** : paramètres séparés par des virgules
  - 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.



Principaux types de messages

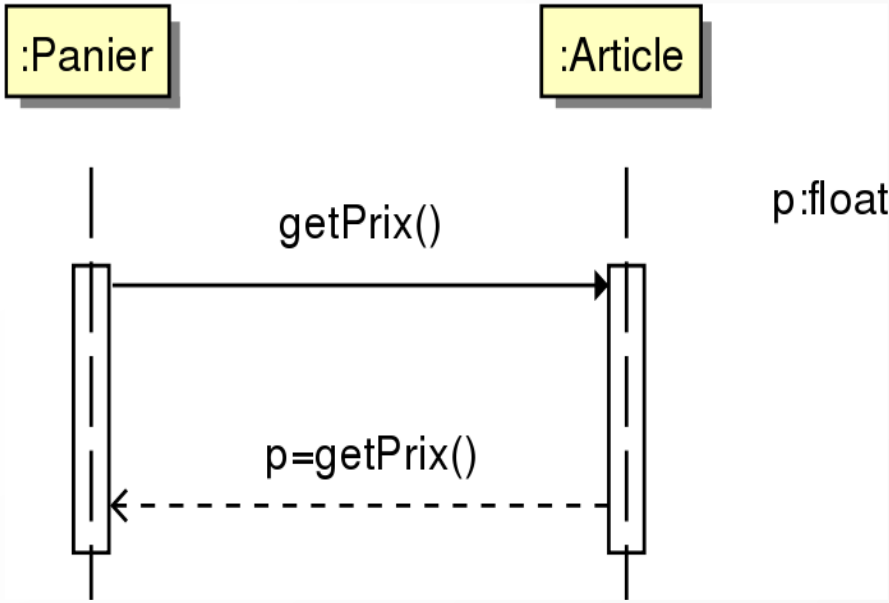
- **Message asynchrone**
  - L'expéditeur continue sans attendre un message de réponse
  - Le message envoyé peut être pris en compte par le récepteur à tout moment ou ignoré.
  - Syntaxe du nom du message : **msg (par1, par2)**
  - Typiquement : envoi de signal

- **Message de réponse ou de retour**
  - Peut être omis si le contenu et l'emplacement sont évidents
  - Syntaxe : **att = msg(par1,par2) : val**
    - **att** : la valeur de retour peut éventuellement être affectée à une variable
    - **msg** : le nom du message
    - **par** : paramètres séparés par des virgules
    - **val** : valeur de retour

Messages de retour

- Le récepteur d'un message synchrone rend la main à l'émetteur du message en lui envoyant un message de retour

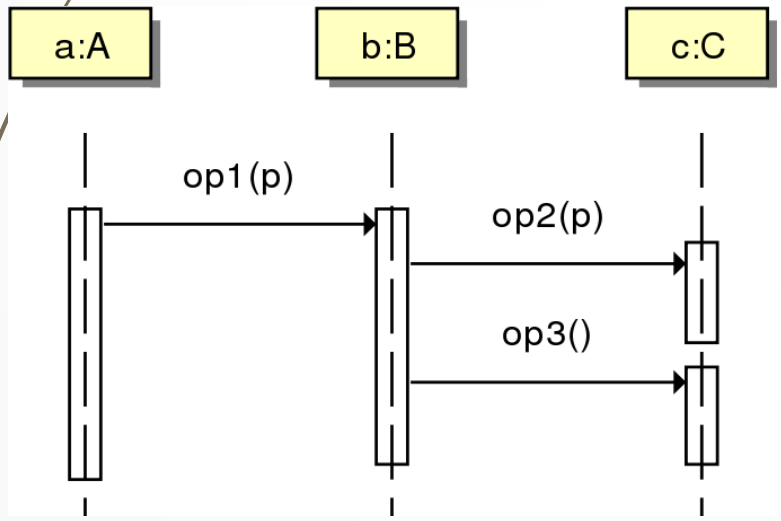
- Les messages de retour sont optionnels : la fin de la période d'activité marque également la fin de l'exécution d'une méthode.
- Ils sont utilisés pour spécifier le résultat de la méthode invoquée.

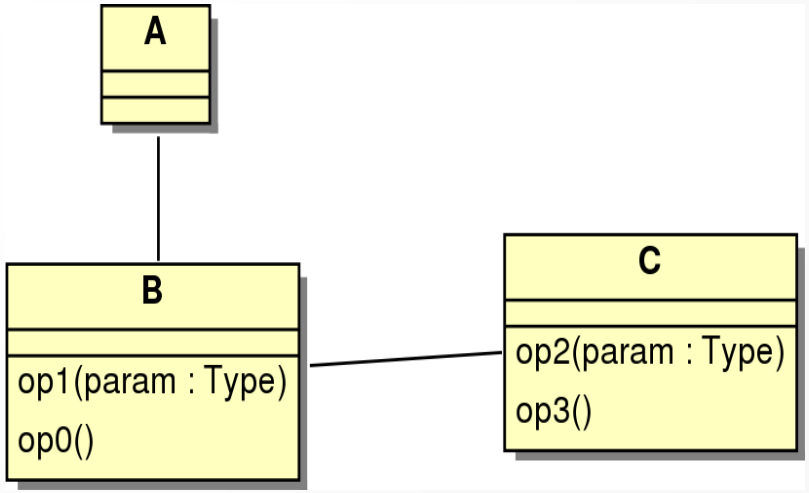


- Le retour des messages asynchrones s'effectue par l'envoi de nouveaux messages asynchrones.

Correspondance messages / opérations

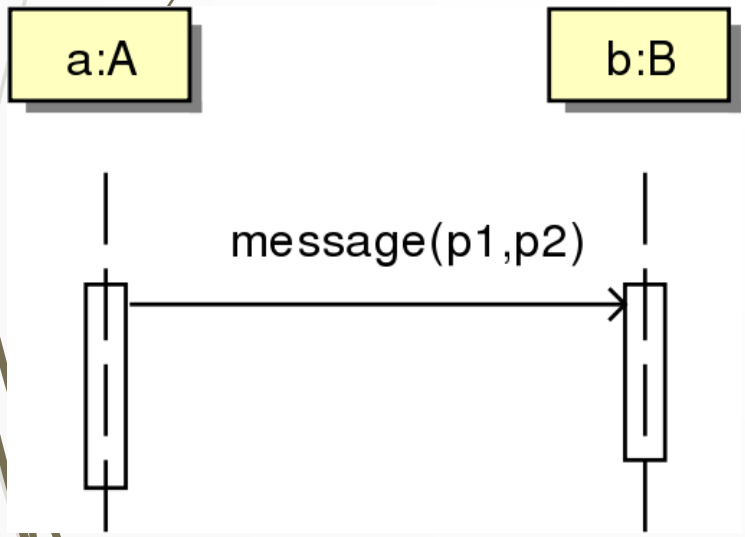
- Les **messages synchrones** correspondent à des opérations dans le diagramme de classes.
- Envoyer un message et attendre la réponse pour poursuivre son activité revient à invoquer une méthode et attendre le retour pour poursuivre ses traitements.

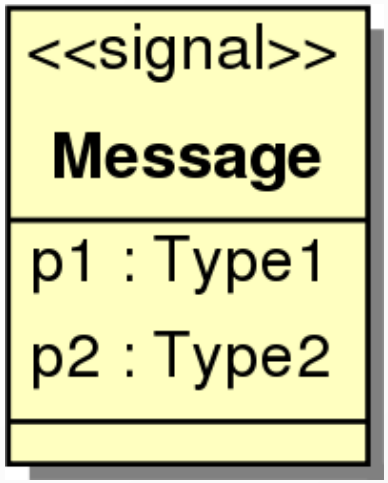




Correspondance messages / signaux

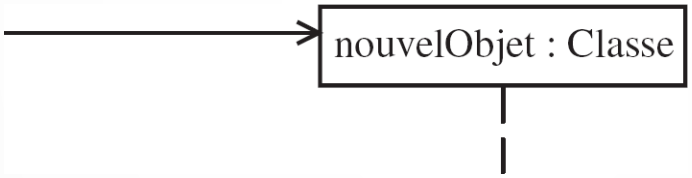
- Les messages asynchrones correspondent à des signaux dans le diagramme de classes.
- Les signaux sont des objets dont la classe est stéréotypée « signal » et dont les attributs (porteurs d'information) correspondent aux paramètres du message.



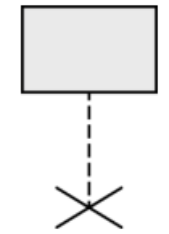


Principaux types de messages

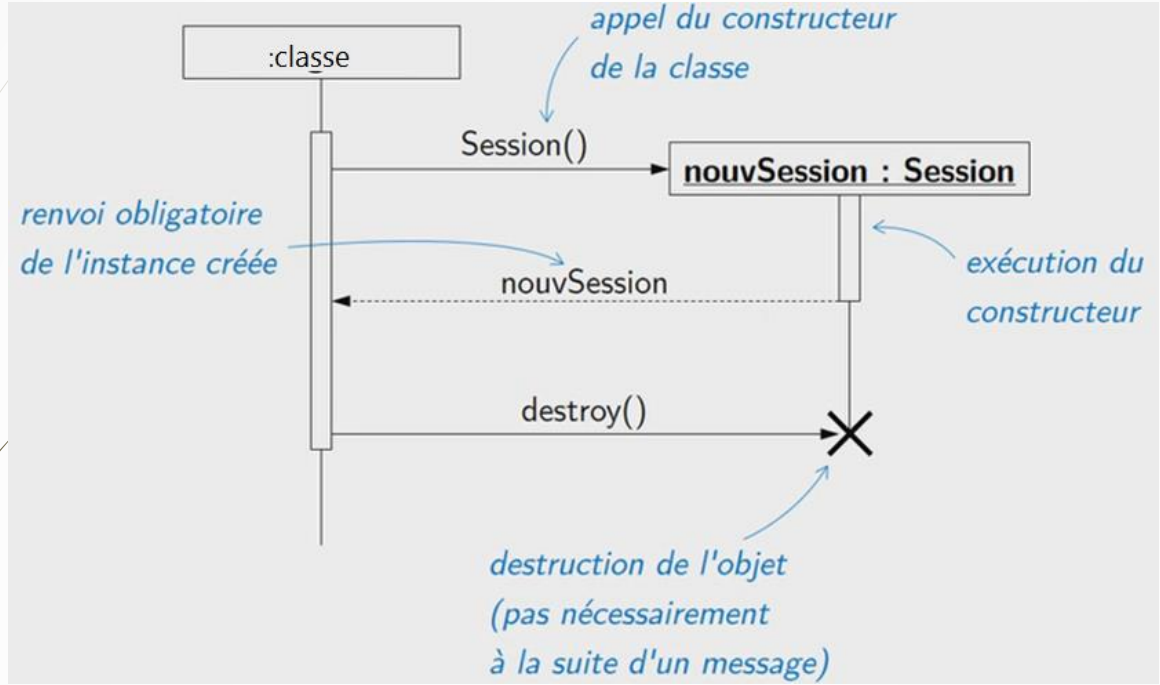
- **Création d'un objet**
  - La création d'un objet est matérialisée par une flèche qui pointe sur le sommet d'une ligne de vie.
  - Appel du constructeur de la classe classe **(synchrone**)
  - On peut aussi utiliser un message **asynchrone** ordinaire portant le nom <<create>> ou new

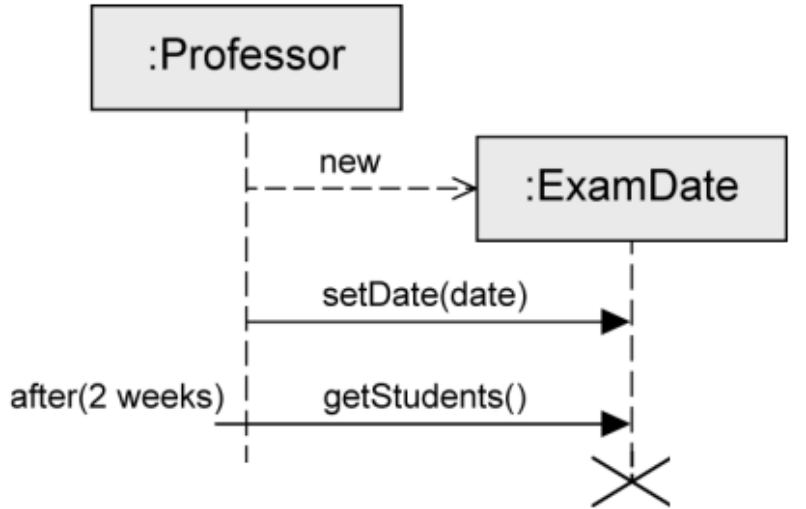


- **Destruction d'un objet**
  - La destruction d'un objet est matérialisée par une croix qui marque la fin de la ligne de vie de l'objet.



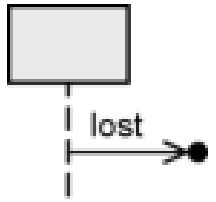
Création et destruction de lignes de vie



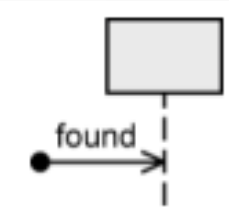


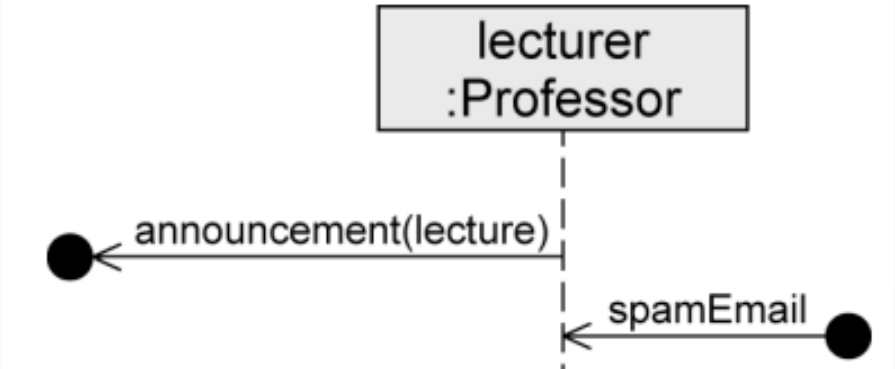
Principaux types de messages

- **Message complet**
  - L'expéditeur et le destinataire d'un message sont connus.
- **Message perdu**
  - Le destinataire d'un message est inconnu ou non pertinent
  - La flèche part d'une ligne de vie mais arrive sur un cercle indépendant marquant la méconnaissance du destinataire.



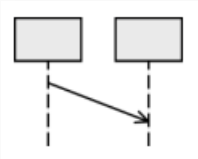
- **Message trouvé**
  - L'expéditeur d'un message est inconnu ou n'est pas pertinent



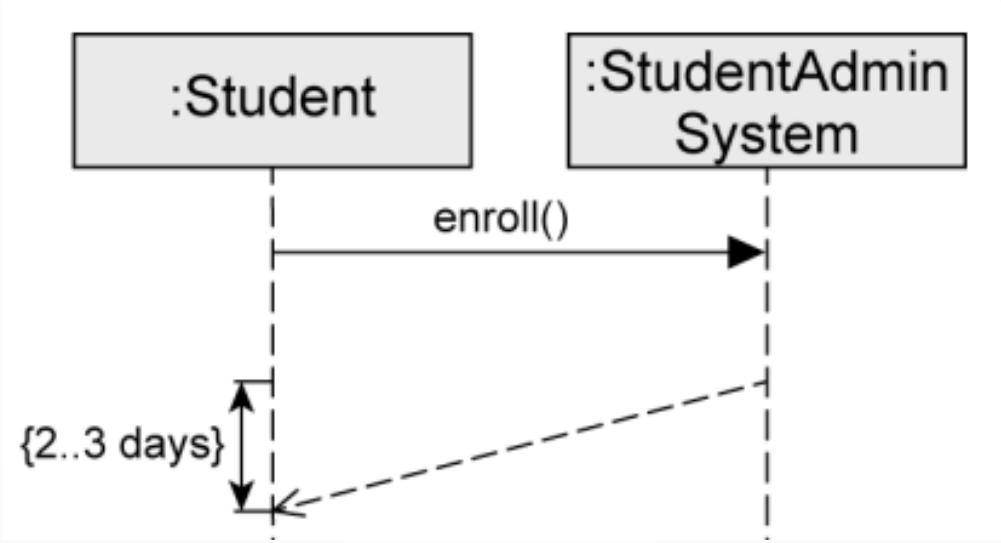


Principaux types de messages

- **Message qui prend du temp**



  - "Message avec durée"
  - Habituellement, les messages sont supposés être transmis sans aucune perte de temps
  - Exprime qu'un délai s'écoule entre l'envoi et la réception d'un message



Messages réflexifs

- Un message peut être envoyé à l'objet lui-même.
- On parle alors d'un message réflexif
- Appel d'une méthode sur l'objet lui-même (par exemple, msg1 est une méthode de la classe T1)

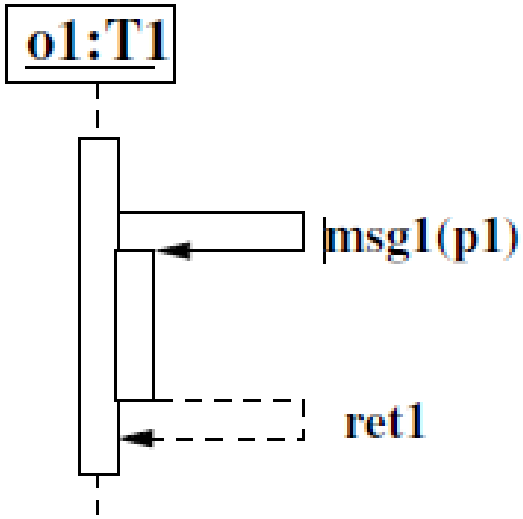


Diagramme de séquence système

- Les DSS permettent de spécifier les opérations système.
- Le système est considéré comme un tout (une boite noire) :
  - On s'intéresse à ses interactions avec les acteurs ;
  - On utilise une classe « Système » qui - à part les acteurs – donnera lieu à la seule ligne de vie des DSS.
- Un DSS est produit pour chacun des cas d'utilisation.

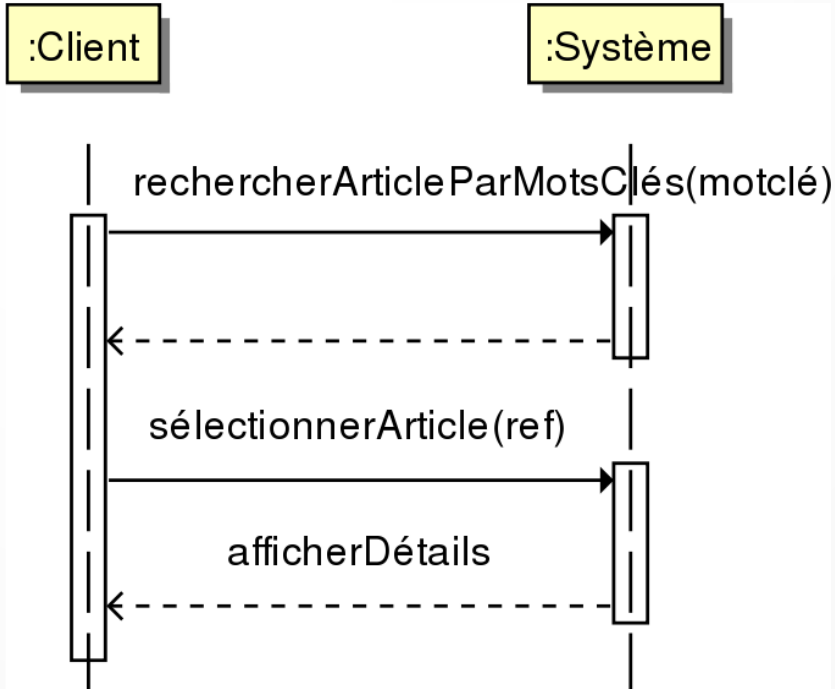
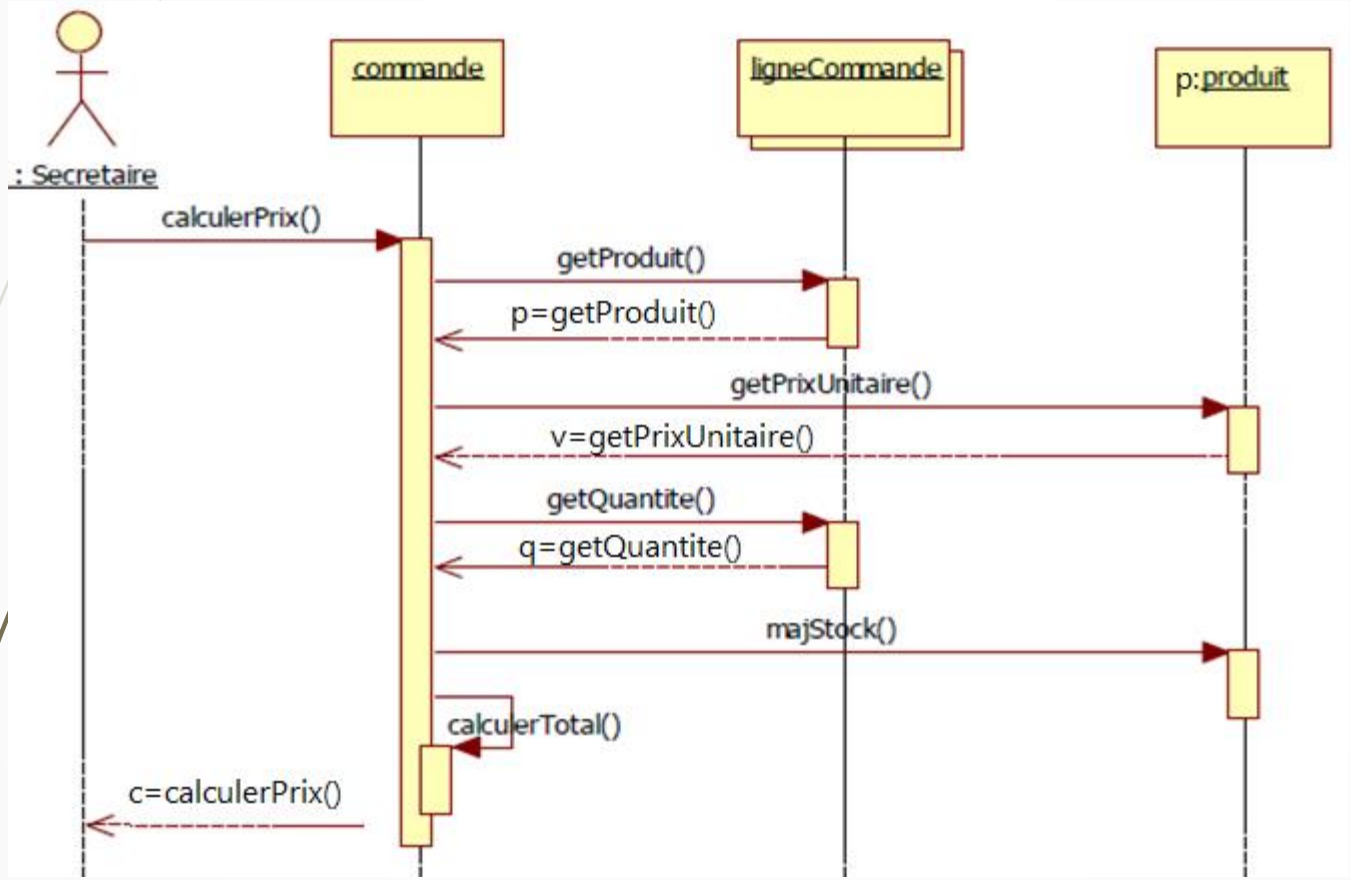


Diagramme de séquence : exemple

- Exemple simplifié sans boucle : Calculer le prix d'un produit



- getProduit(), getQuantite() : méthodes de la classe ligneCommande
- getPrixUnitaire(), majStock() : méthodes de la classe produit
- calculerTotal(), calculerPrix(): méthodes de la classe commande

Fragment combiné

- Les éléments du diagramme de séquence présentés jusqu'à maintenant ne modélise que des scenarios qui sont composés des séquences de messages

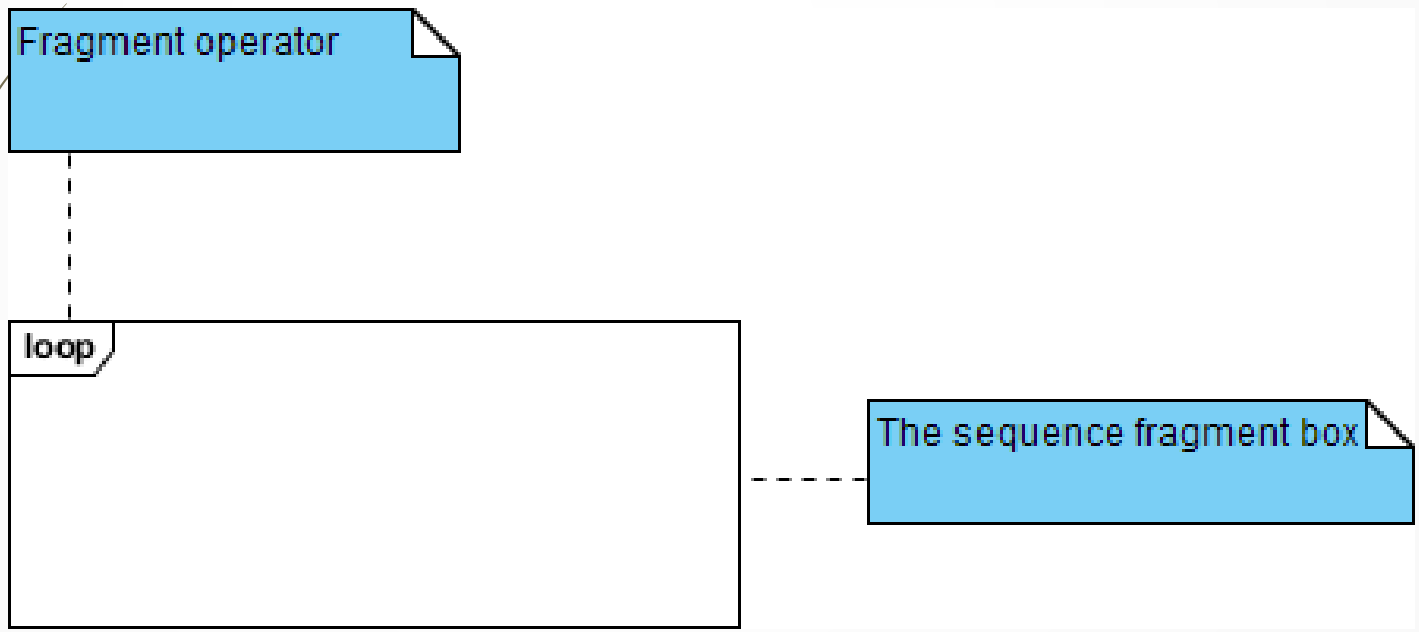
- Pour prendre en compte tous les scenarios possibles d'un cas d'utilisation dans le même diagramme de séquence
  - On a besoin de plus d'éléments : **les fragments combinés**

- Un fragment combiné permet de décomposer un diagramme complexe en fragments suffisamment simples pour être compris.

- 12 fragments combinés

Fragment combiné

- Un fragment combiné est représenté par un rectangle dont le coin supérieur gauche contient un pentagone.
  - Dans le pentagone figure le type du fragment (appelé «opérateur d'interaction»).



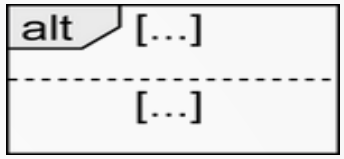
Type d'opérateurs d'interaction

	Operator	Purpose
Branches and loops	**alt**	Alternative interaction
	opt	Optional interaction
	loop	Repeated interaction
	break	Exception interaction
Concurrency and order	**seq**	Weak order
	strict	Strict order
	par	Concurrent interaction
	critical	Atomic interaction
Filters and assertions	**ignore**	Irrelevant interaction
	consider	Relevant interaction
	assert	Asserted interaction
	neg	Invalid interaction

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.

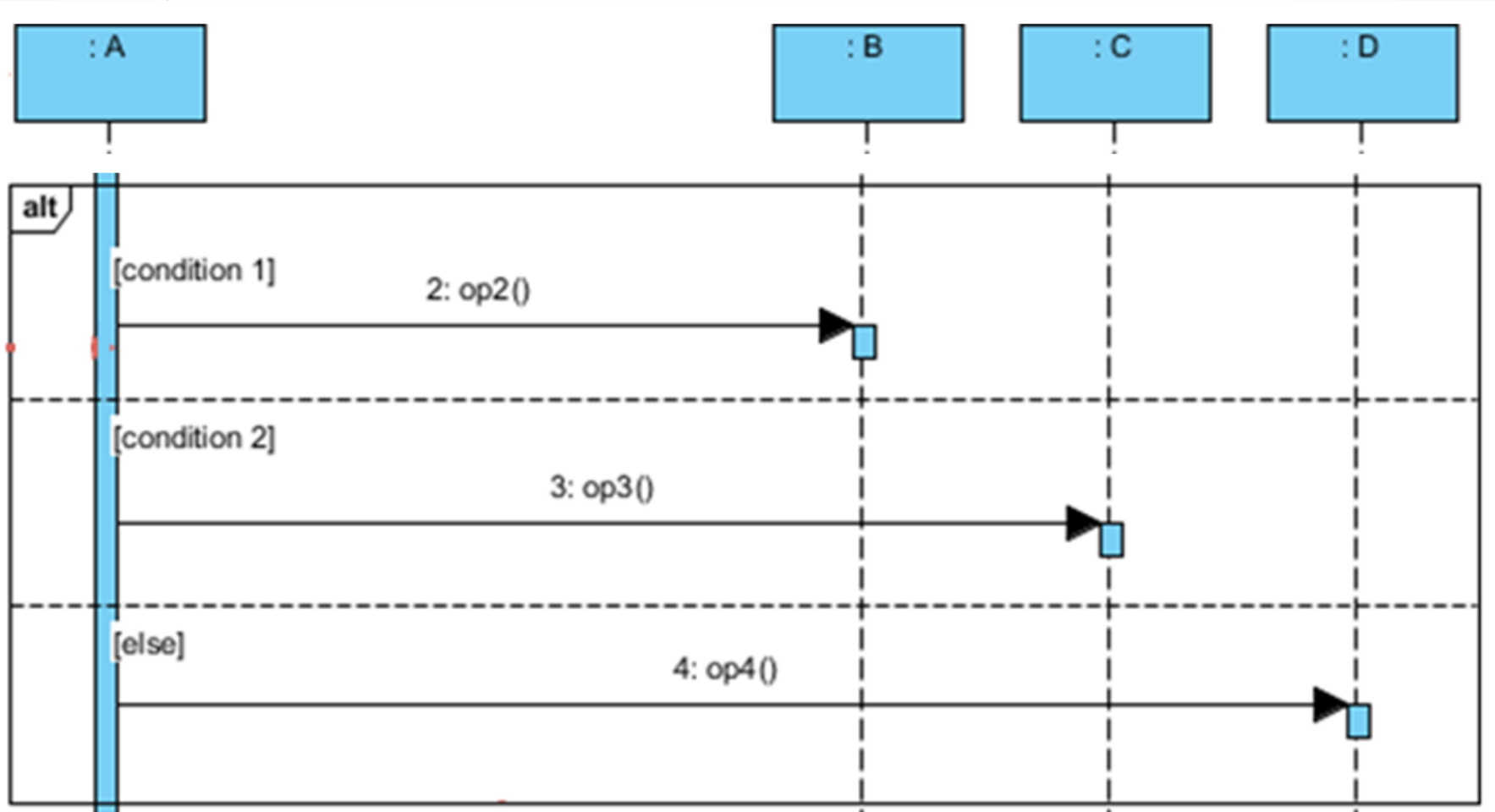
Le fragment alt



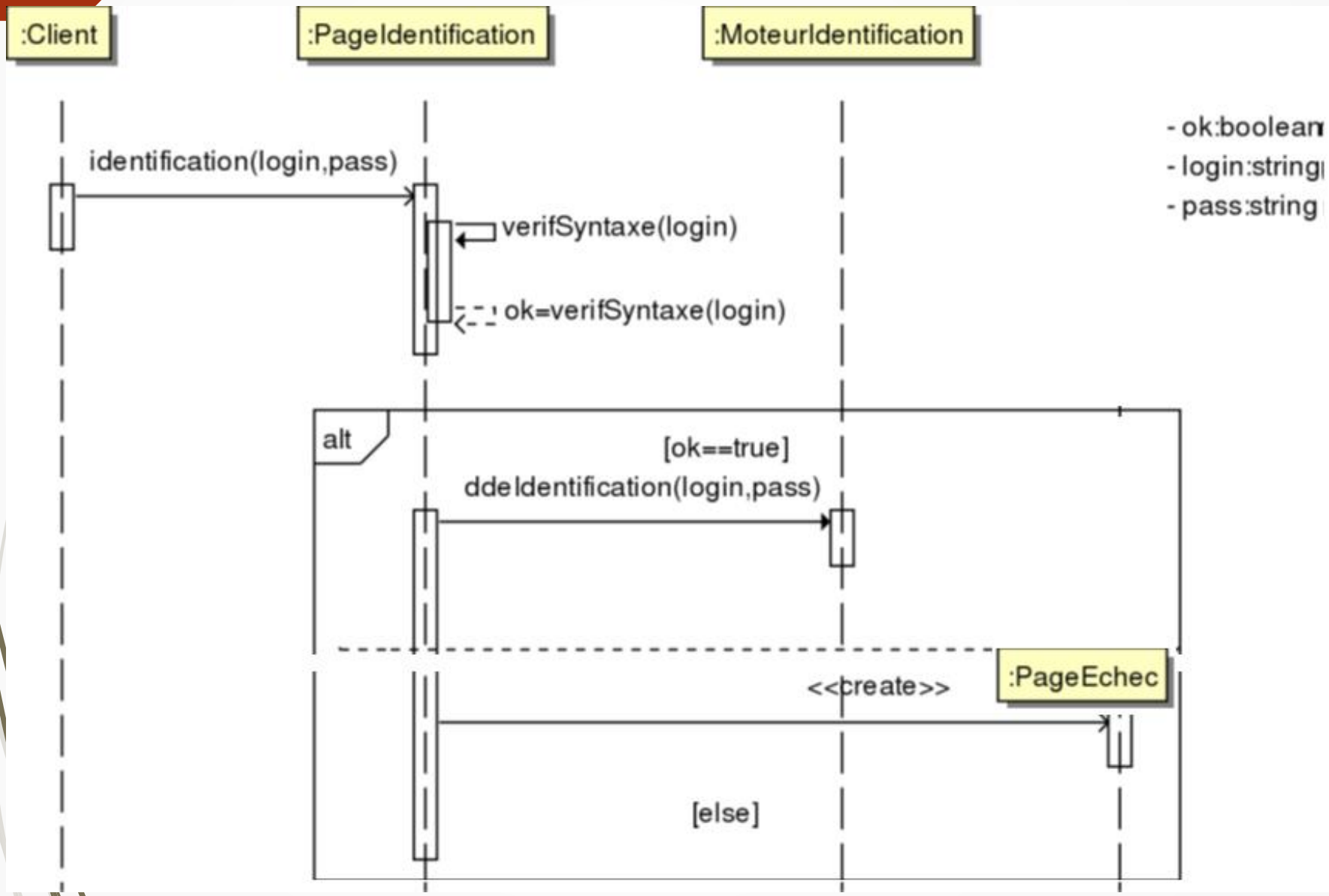
- Pour modéliser une séquence alternative
- Semblable à l'instruction switch dans Java
- Les gardes sont utilisées pour sélectionner le chemin à exécuter
- Gardes
  - Mise entre crochets
  - par défaut : true
  - prédéfinie : [else]
- Plusieurs opérandes, au moins deux
- Les gardes doivent être disjointes pour éviter les comportements indéterministes

Les fragments alt et opt

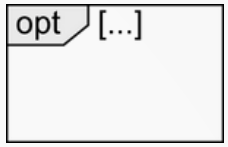
- Op2() est exécutée si condition1 est vraie
- Op3() est exécutée si condition2 est vraie
- Op4() est exécutée autrement



Le fragment alt : exemple d'authentification



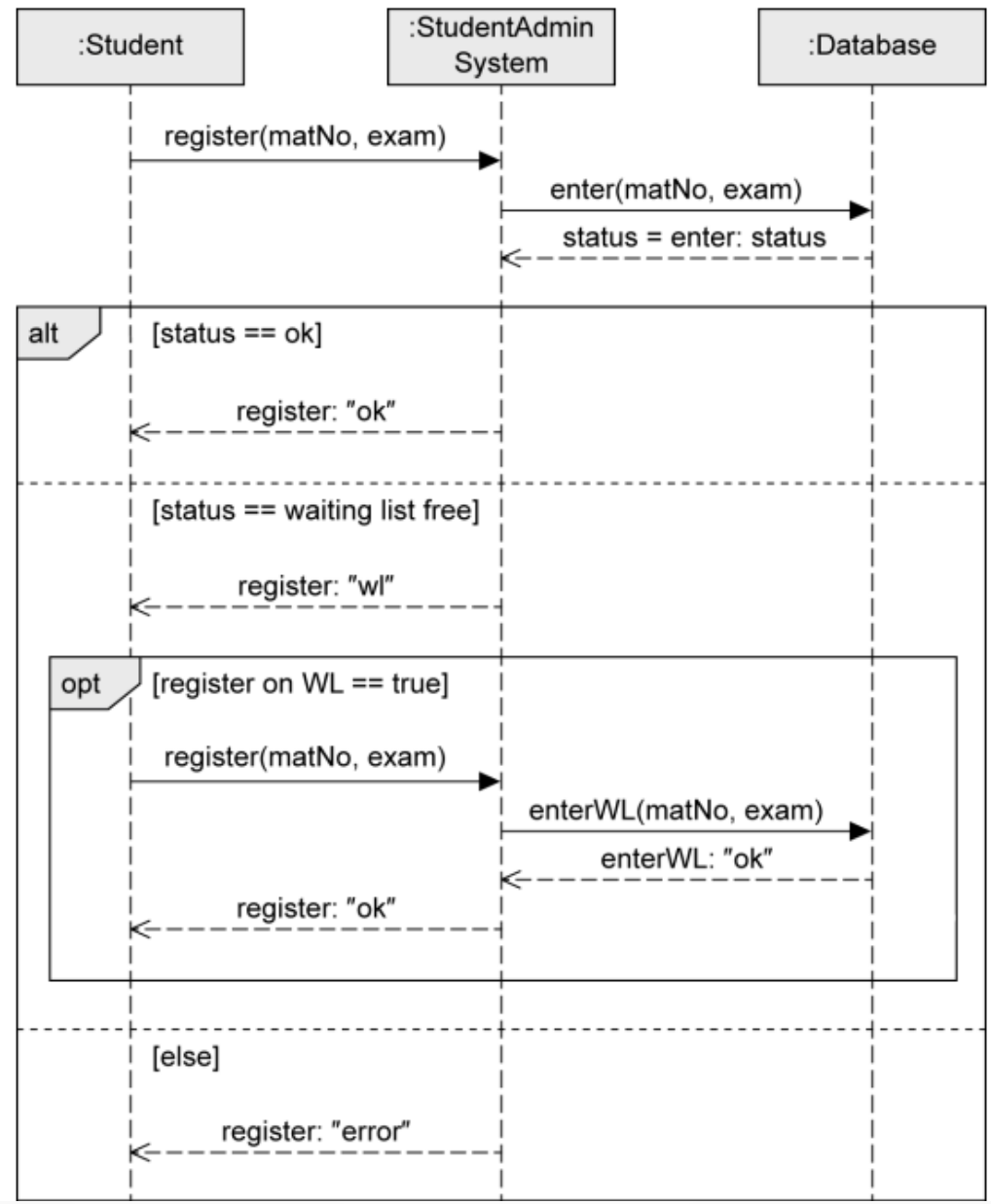
Le fragment opt



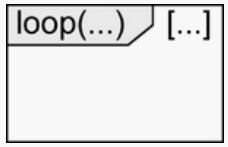
- Pour modéliser une séquence optionnelle
- Exécution réelle dépend de la condition de garde
- Exactement un opérande
- Semblable à l'instruction **if** sans la branche **else**
- équivalent au fragment alt avec deux opérandes, l'un d'eux est vide

Op1() est exécutée si la garde (**condition)** est vraie

Les fragments alt et opt : exemple



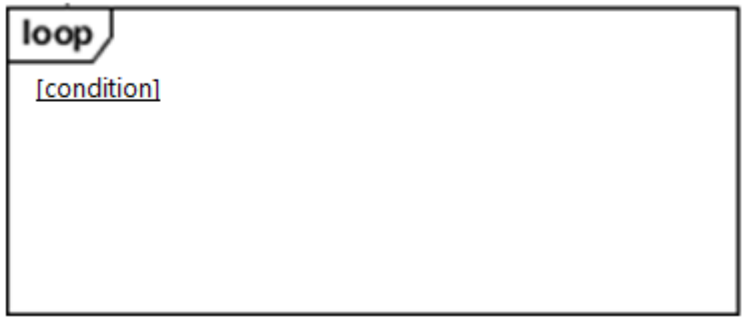
Le fragment loop



- Pour exprimer qu'une séquence doit être exécutée à plusieurs reprises
- Syntaxe d'une boucle :

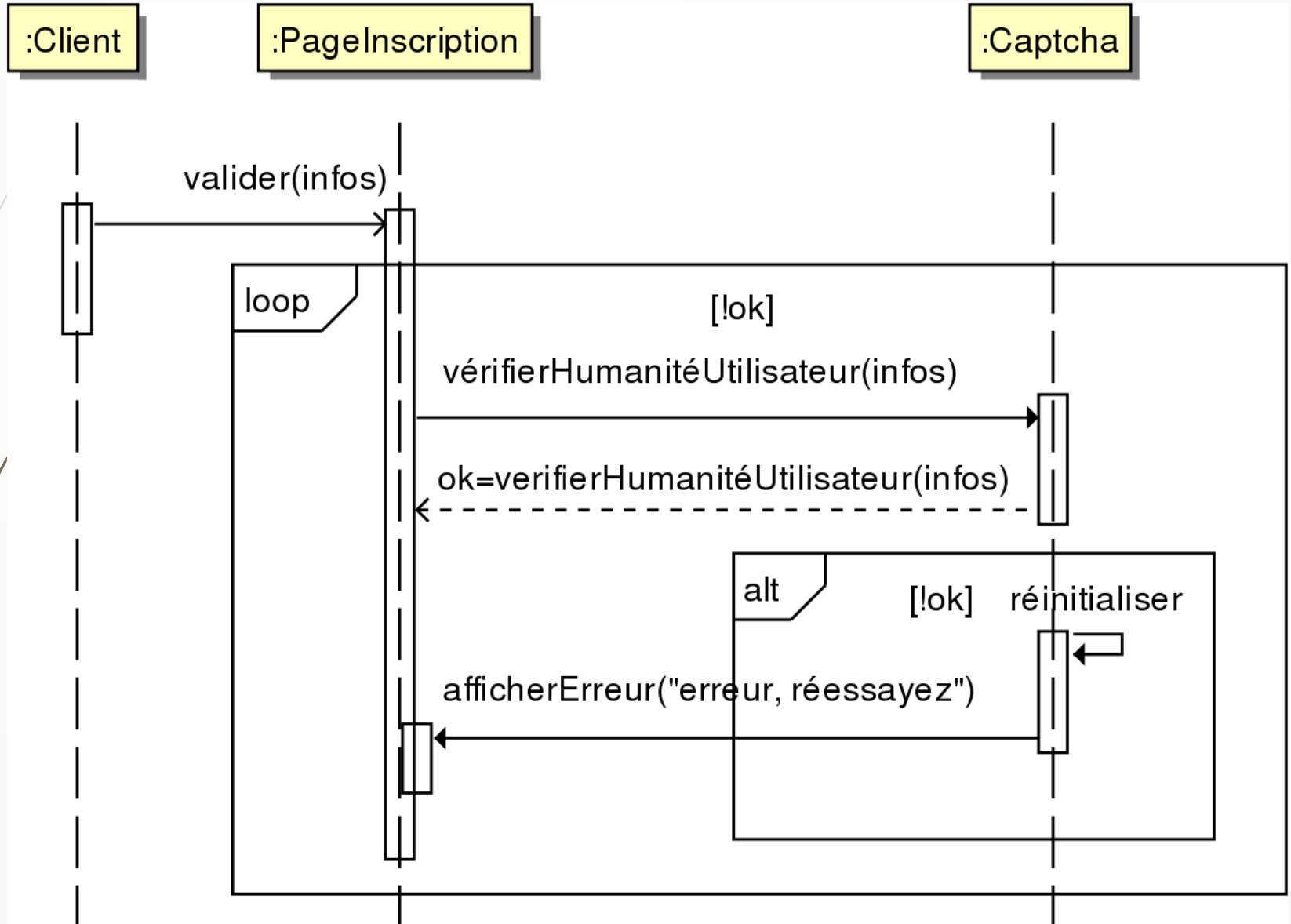
  loop (min , max)

  - La boucle est répétée au moins **min** fois avant qu'une éventuelle condition booléenne ne soit testée (la condition est placée entre crochets dans le fragment)
  - Tant que la condition est vraie, la boucle continue, au plus **max** fois.

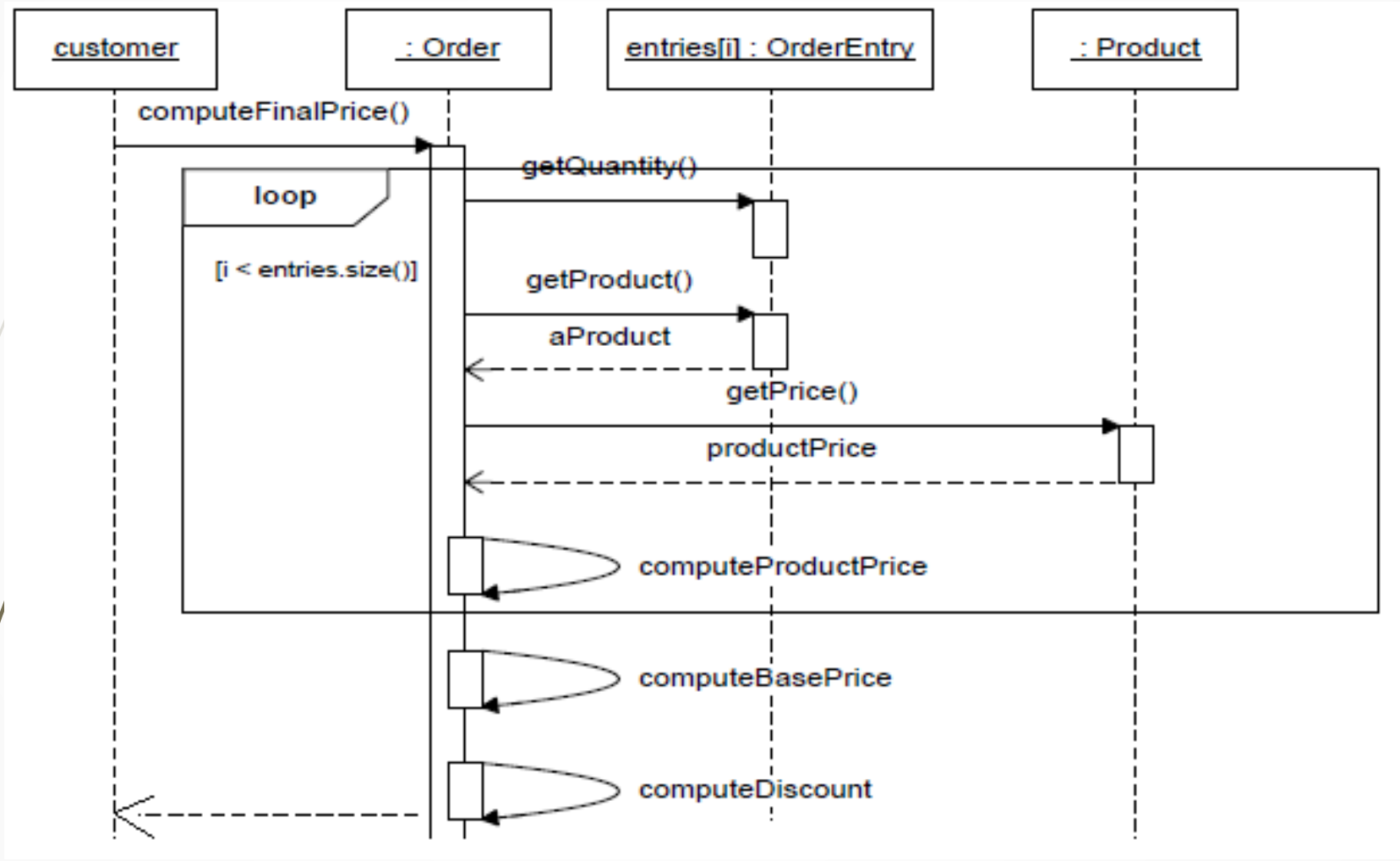


- Notations :
  - loop(valeur) est équivalent à loop(valeur,valeur).
  - loop est équivalent à loop(0,*), où * signifie illimité .

DS avec des boucles

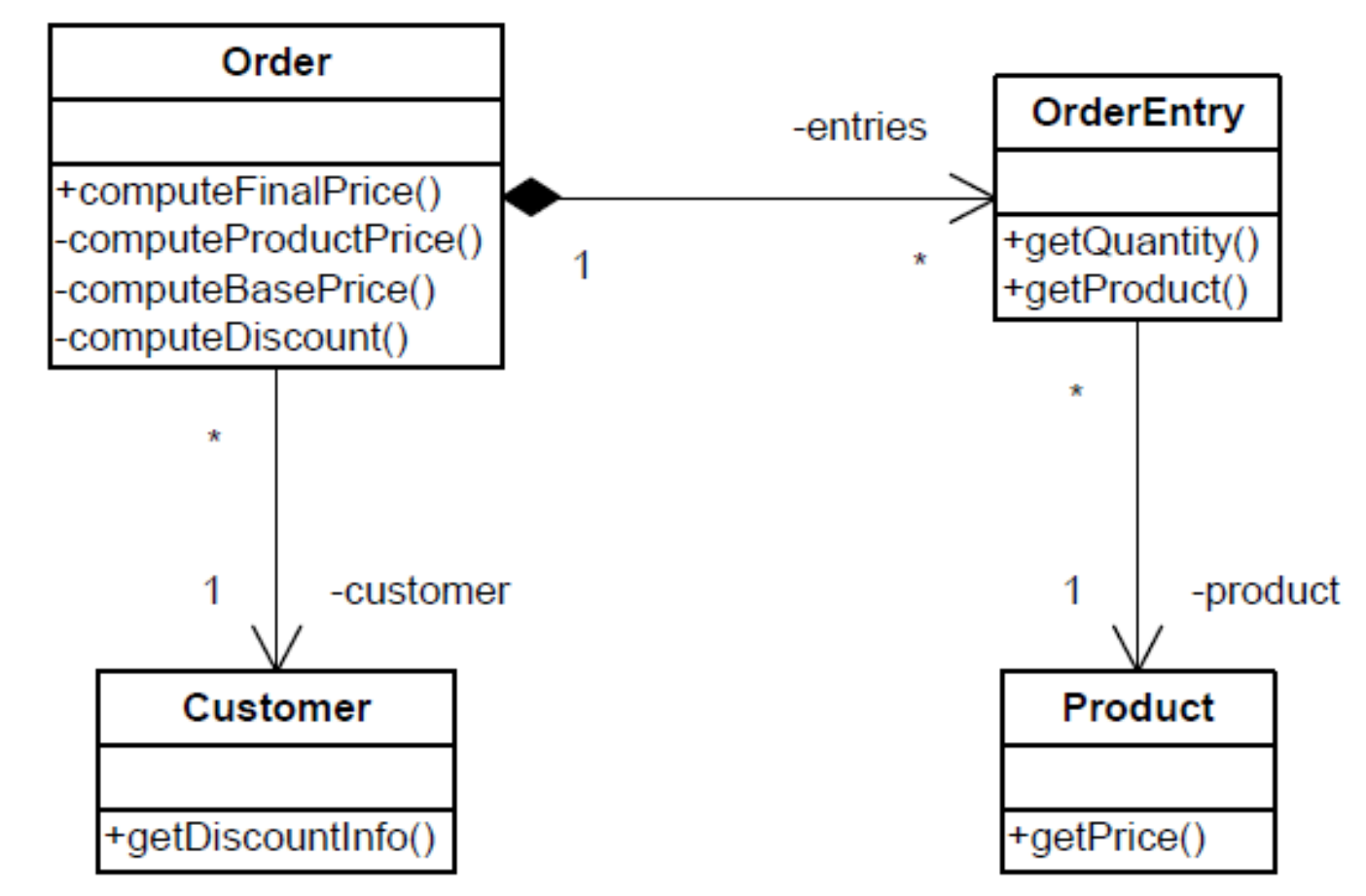


Exemple : Commander un article

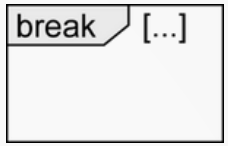


Exemple : Commander un article

Diagramme de classes correspondant

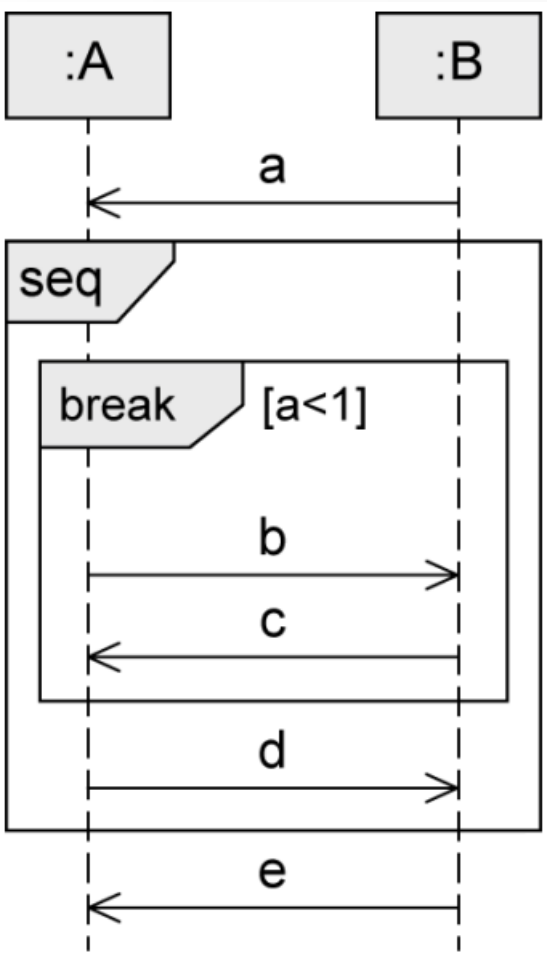


Le fragment Break

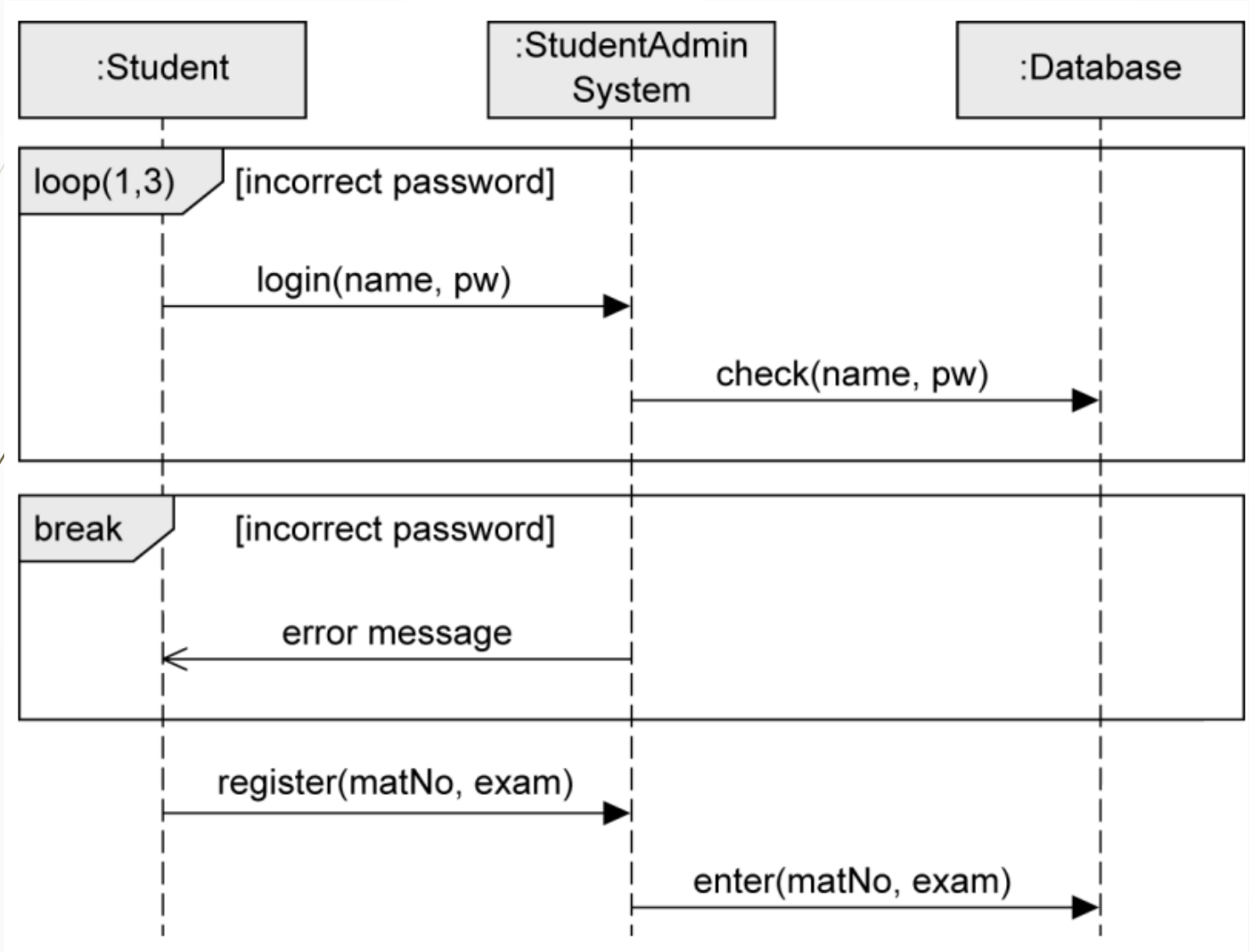


- Ce fragment interrompt le fragment parent.
- Forme simple de gestion des exceptions
- Exactement un opérande avec une condition de garde
- Si la condition de garde est vraie :
  - Les interactions au sein de cet opérande sont exécutées
  - Les opérations restantes du fragment parent sont ignorées
  - L'interaction se poursuit dans le fragment de niveau supérieur suivant

Not executed if break is executed

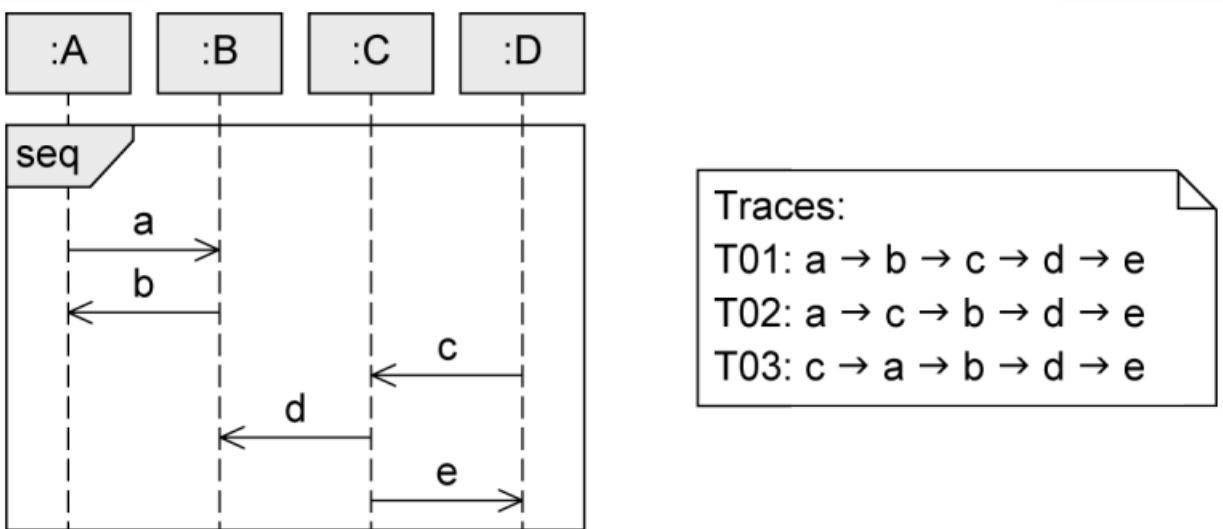


Les fragments loop et break : exemple

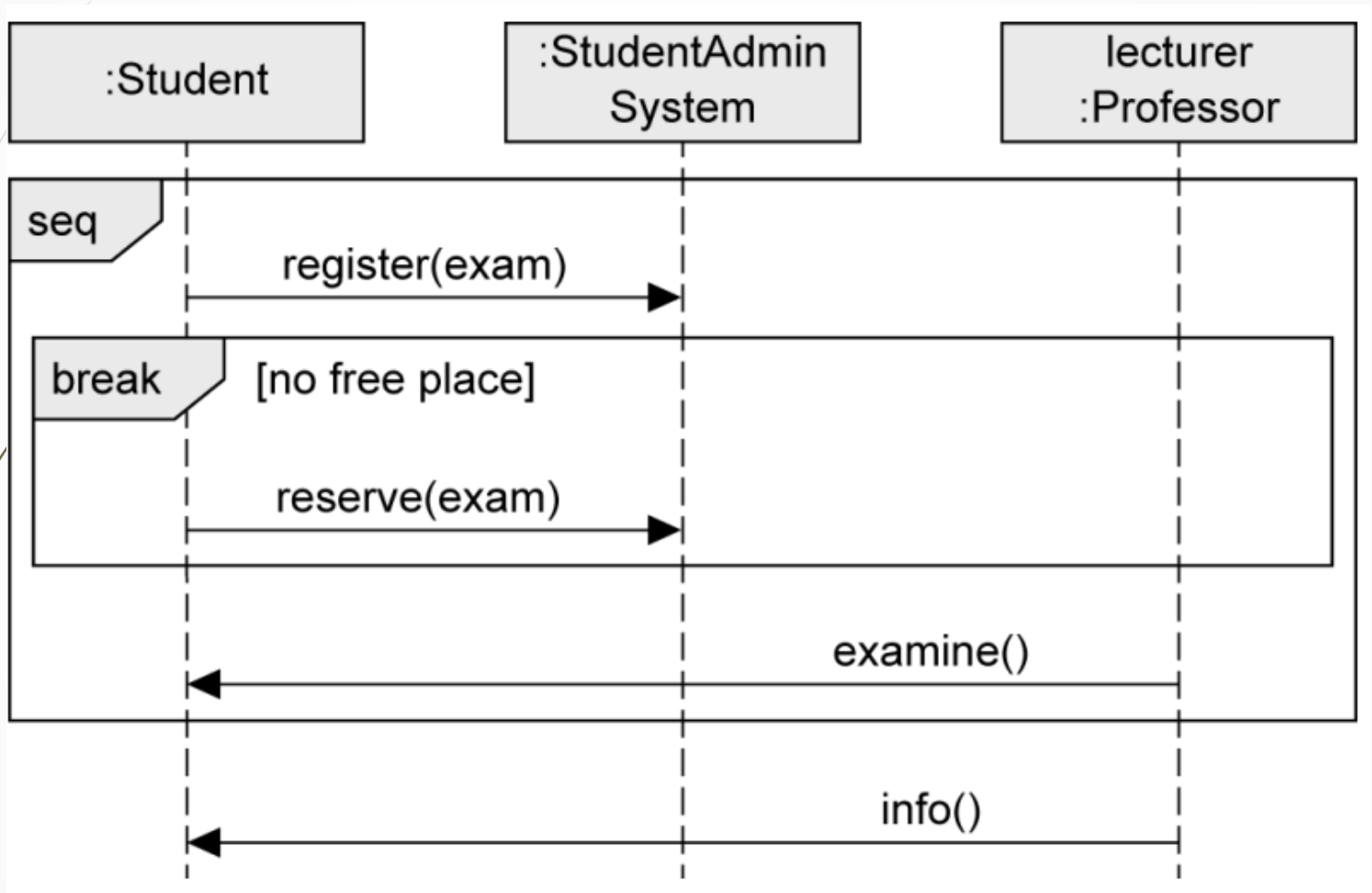


Le fragment seq

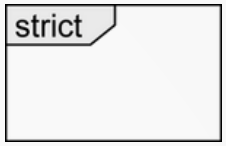
- Ordre par défaut des événements
- Séquencement faible :
  1. L'ordre des événements au sein de chacun des opérandes est conservé dans le résultat.
  2. Les événements sur différentes lignes de vie provenant de différents opérandes peuvent survenir dans n'importe quel ordre.
  3. Les événements sur la même ligne de vie provenant de différents opérandes sont ordonnés de telle sorte qu'un événement du premier opérande précède celui du deuxième opérande.



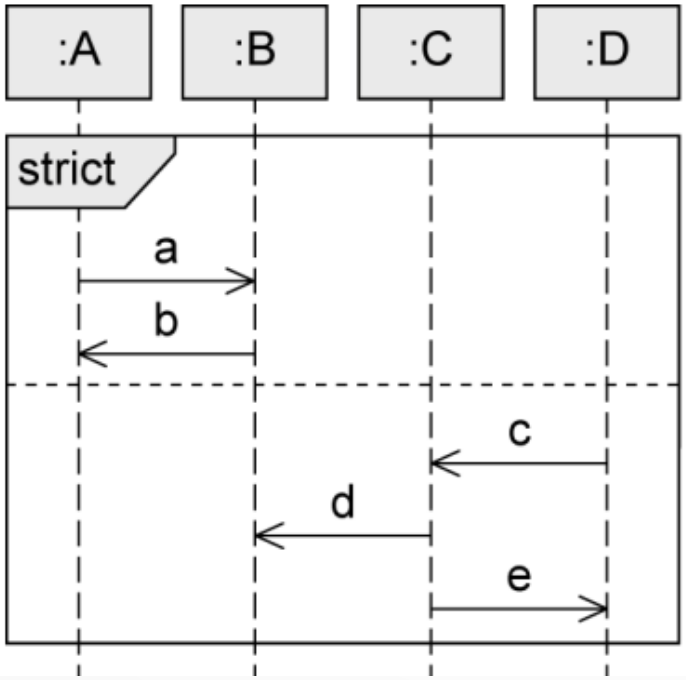
Le fragment seq : exemple



Le fragment strict

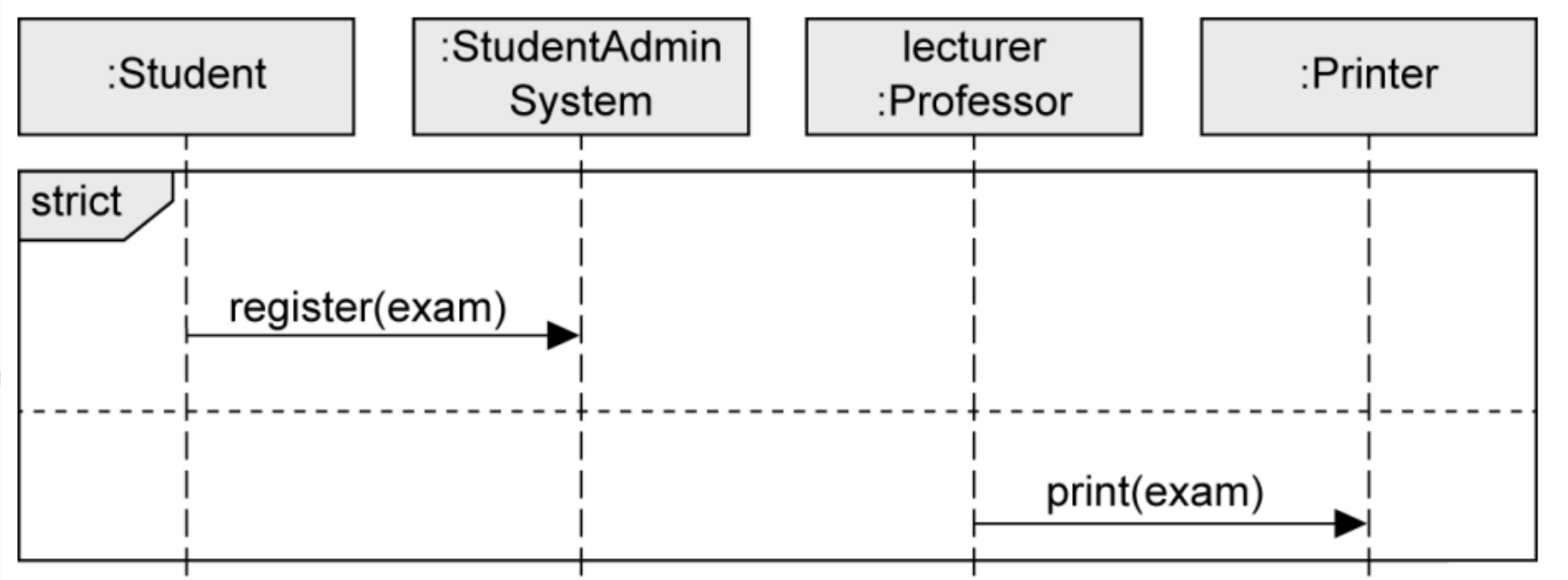


- Interaction séquentielle dans l'ordre
- L'ordre des occurrences d'événements sur différentes lignes de vie entre différents opérandes est significatif
- Les messages d'un opérande plus haut sur l'axe vertical sont toujours échangés avant les messages d'un opérande plus bas sur l'axe vertical.

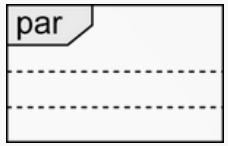


Traces:
T01: a → b → c → d → e

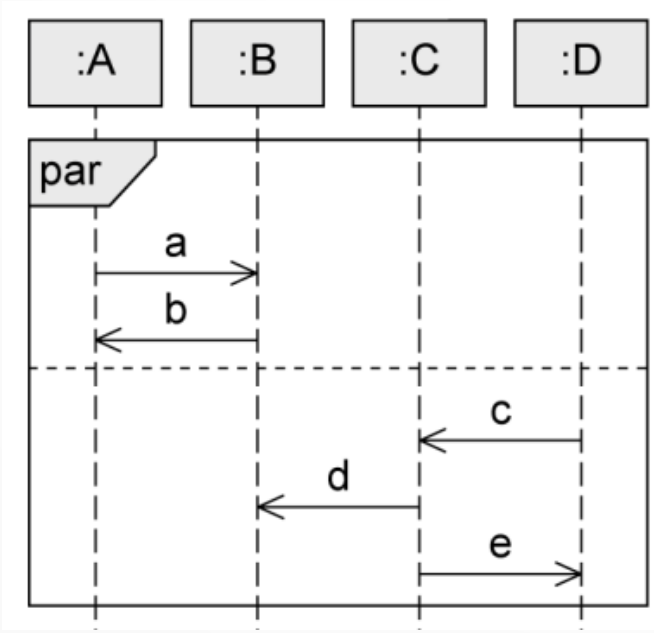
Le fragment strict : exemple



Le fragment par

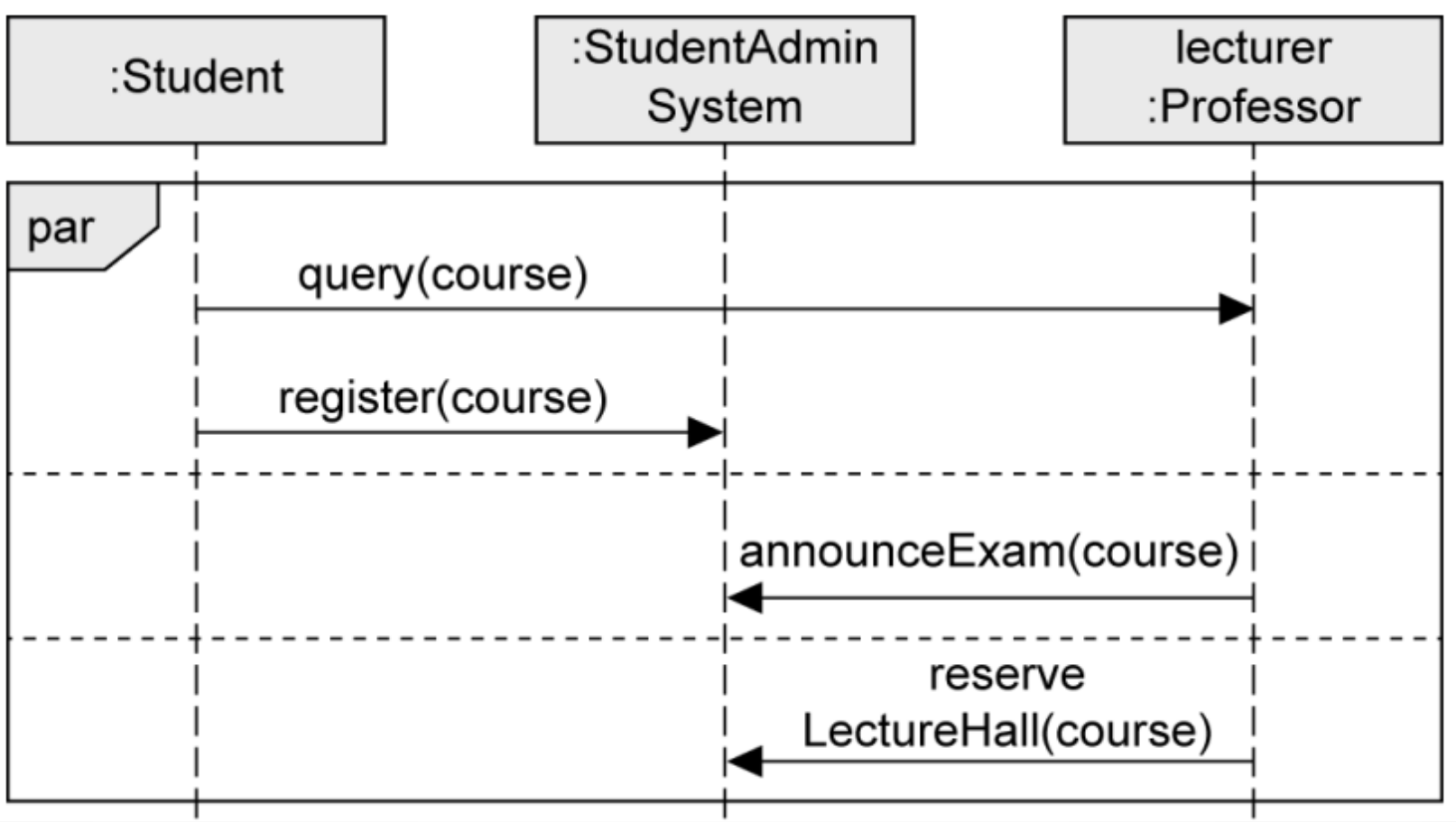


- L'opérateur **par** permet d'envoyer des messages en parallèle.
- Ce qui se passe de part et d'autre de la ligne pointillée est indépendant.
- Les restrictions de chaque opérande doivent être respectées
- L'ordre des différents opérandes n'a pas d'importance
- Concurrence, pas de véritable parallélisme



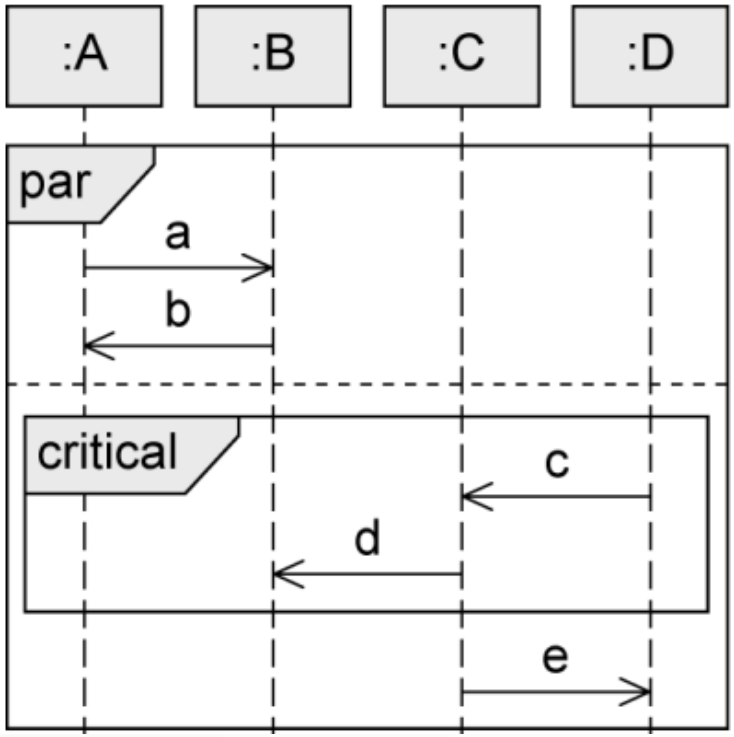
Traces:
T01: a → b → c → d → e
T02: a → c → b → d → e
T03: a → c → d → b → e
T04: a → c → d → e → b
T05: c → a → b → d → e
T06: c → a → d → b → e
T07: c → a → d → e → b
T08: c → d → a → b → e
T09: c → d → a → e → b
T10: c → d → e → a → b

Le fragment par : exemple



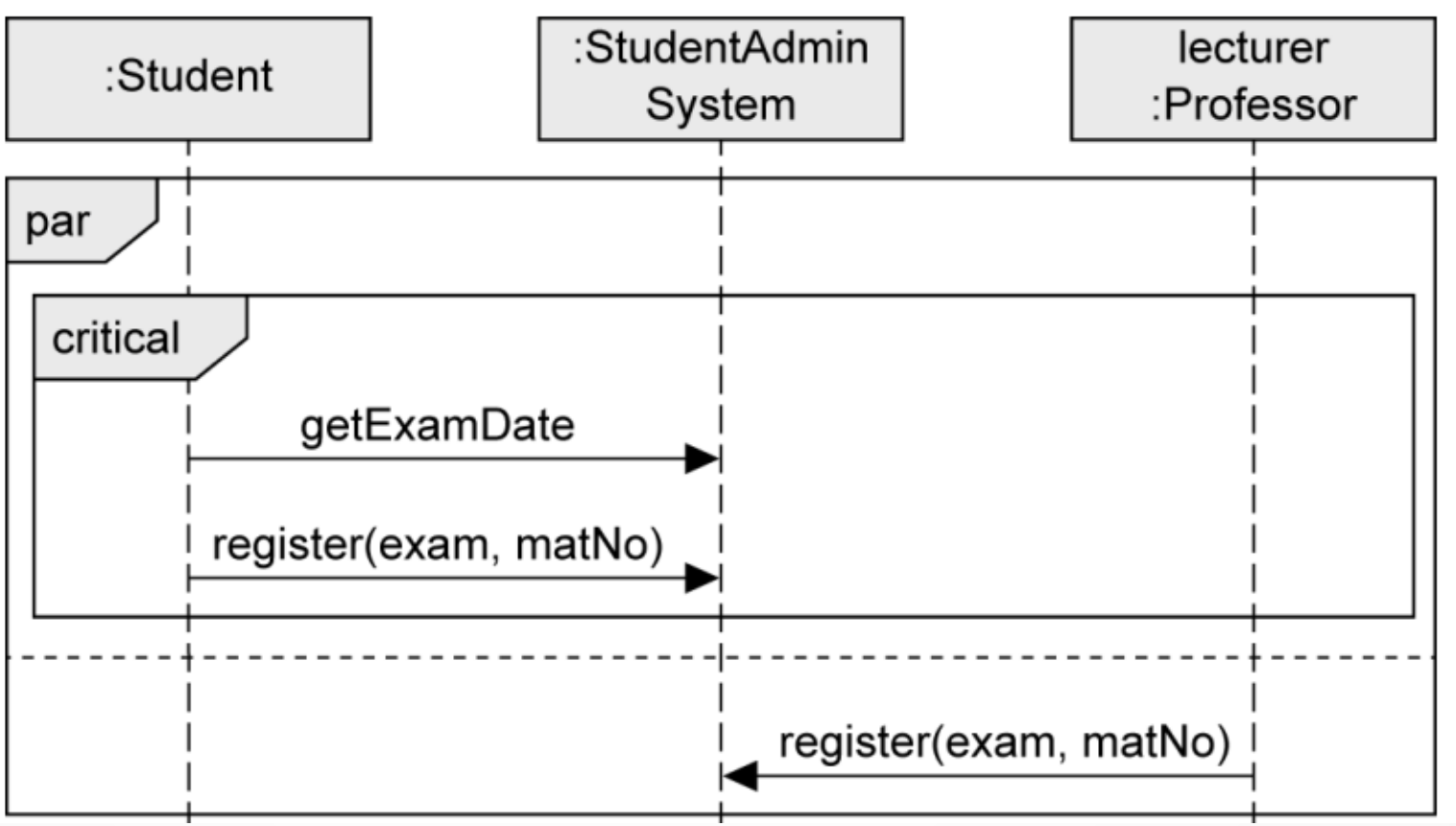
Le fragment critical : section critique

- Exécution de la section critique est atomique. Elle ne peut pas être divisée
- Pour s'assurer que certaines parties d'une interaction ne sont pas interrompues par des événements inattendus
- L'ordre dans **critical** : ordre par défaut **seq**

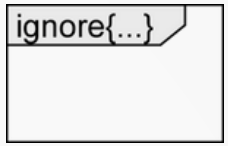


Traces:
T01: a → b → c → d → e
T02: a → c → d → b → e
T03: a → c → d → e → b
T04: c → d → a → b → e
T05: c → d → a → e → b
T06: c → d → e → a → b

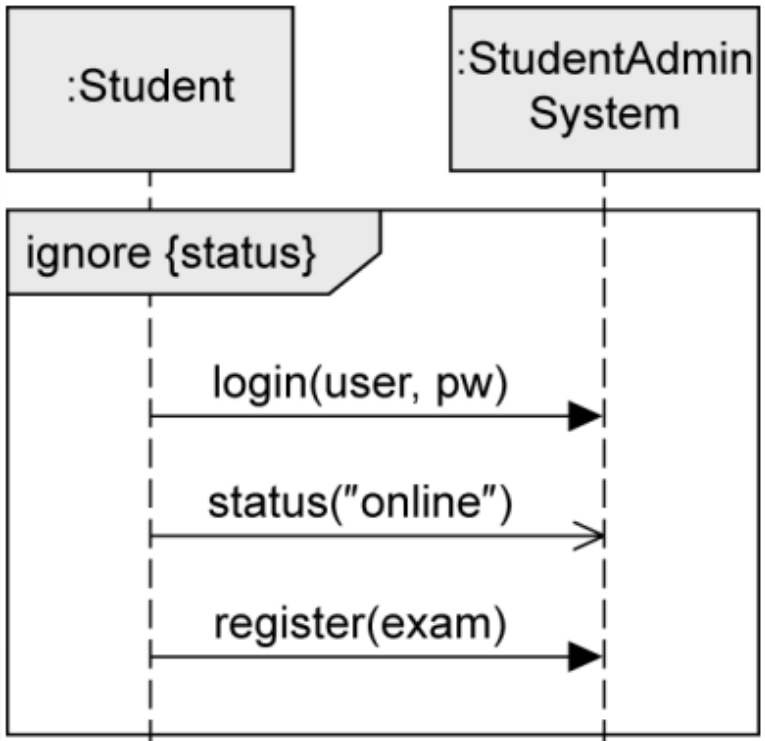
Le fragment critical : exemple



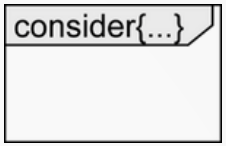
Le fragment ignore



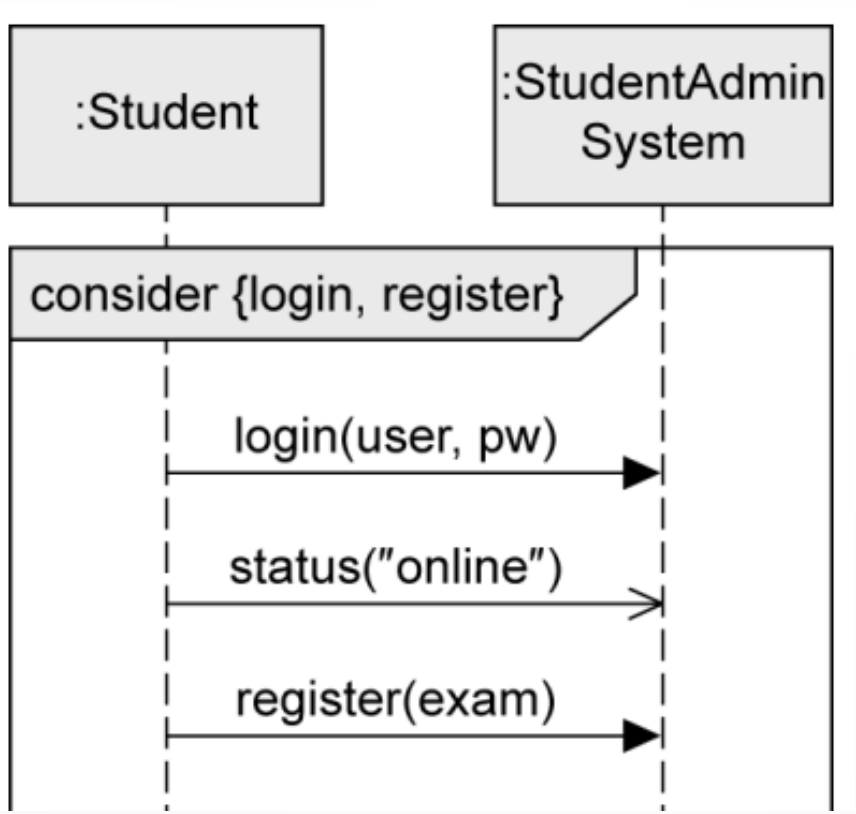
- Pour signaler des messages non pertinents
- Des messages peuvent apparaître au moment de l'exécution mais n'ont aucune autre signification
- Exactement un opérande
- Messages non pertinents entre accolades après le mot-clé **ignore**



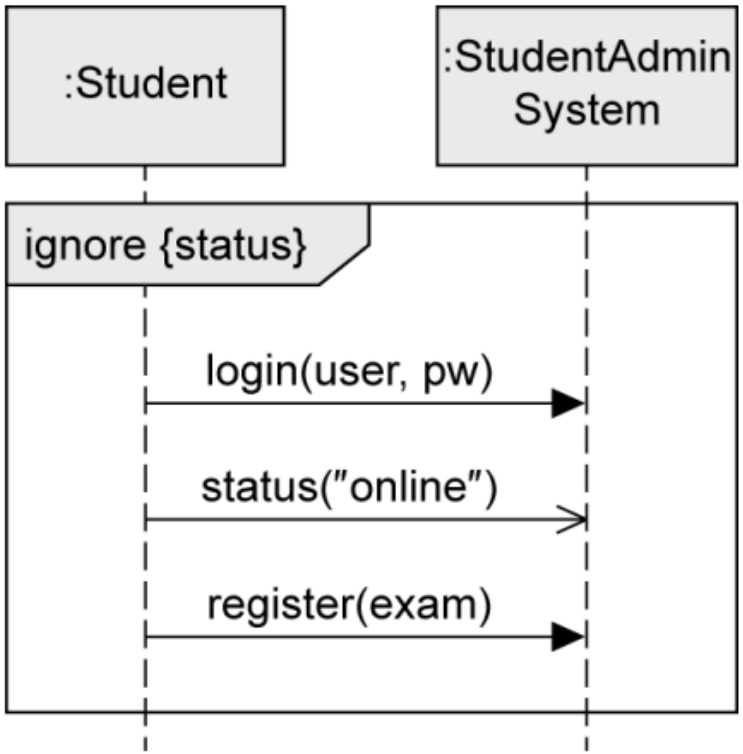
Le fragment consider

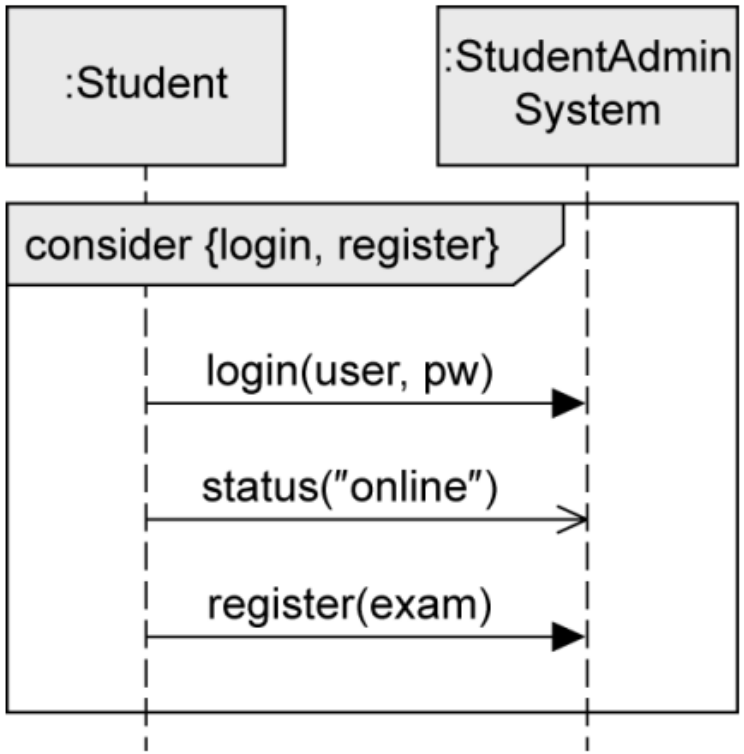


- Pour spécifier les messages qui revêtent une importance particulière pour l'interaction considérée
- Messages considérés entre accolades après le mot-clé **consider**

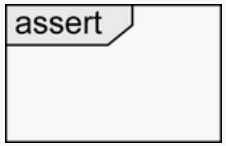


Consider vs ignore

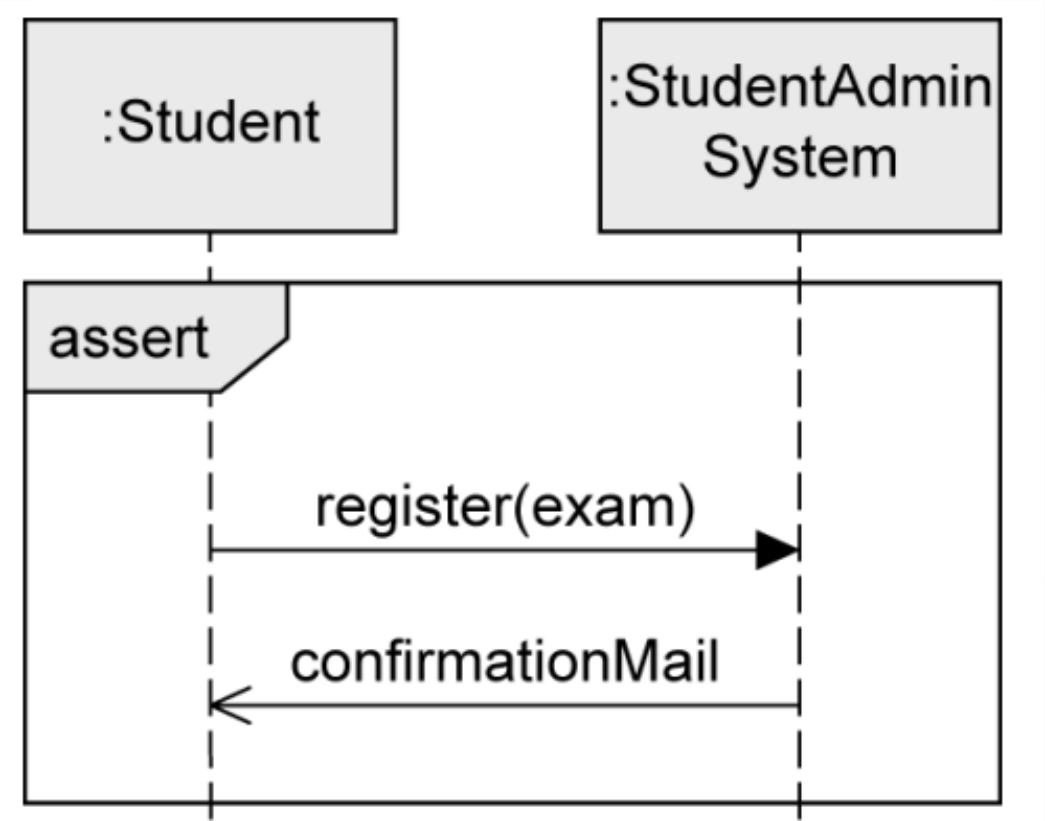




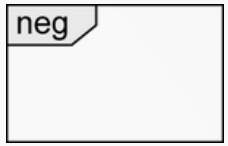
Le fragment assert



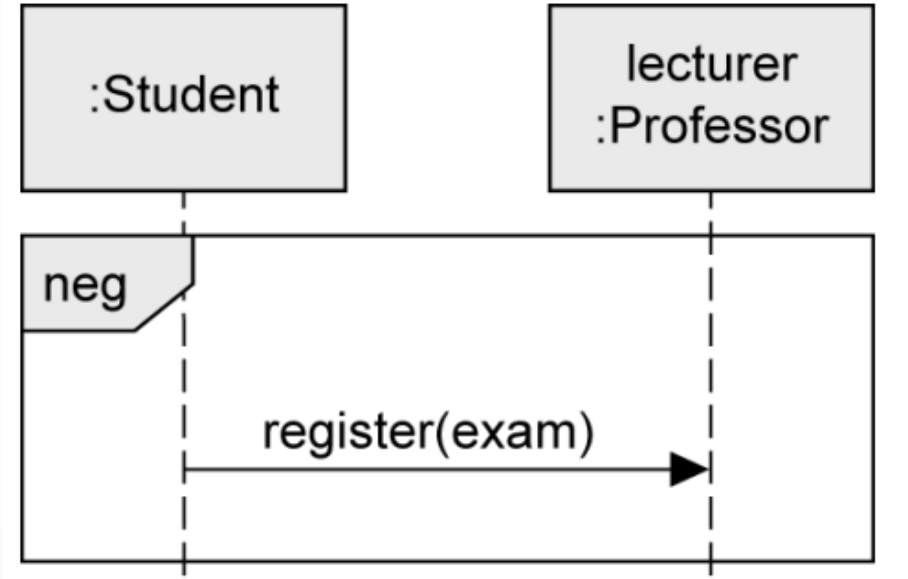
- Identifier certaines traces modélisées comme obligatoires
- Les écarts qui se produisent dans la réalité mais qui ne sont pas inclus dans le diagramme ne sont pas autorisés
- Exactement un opérande



Le fragment neg

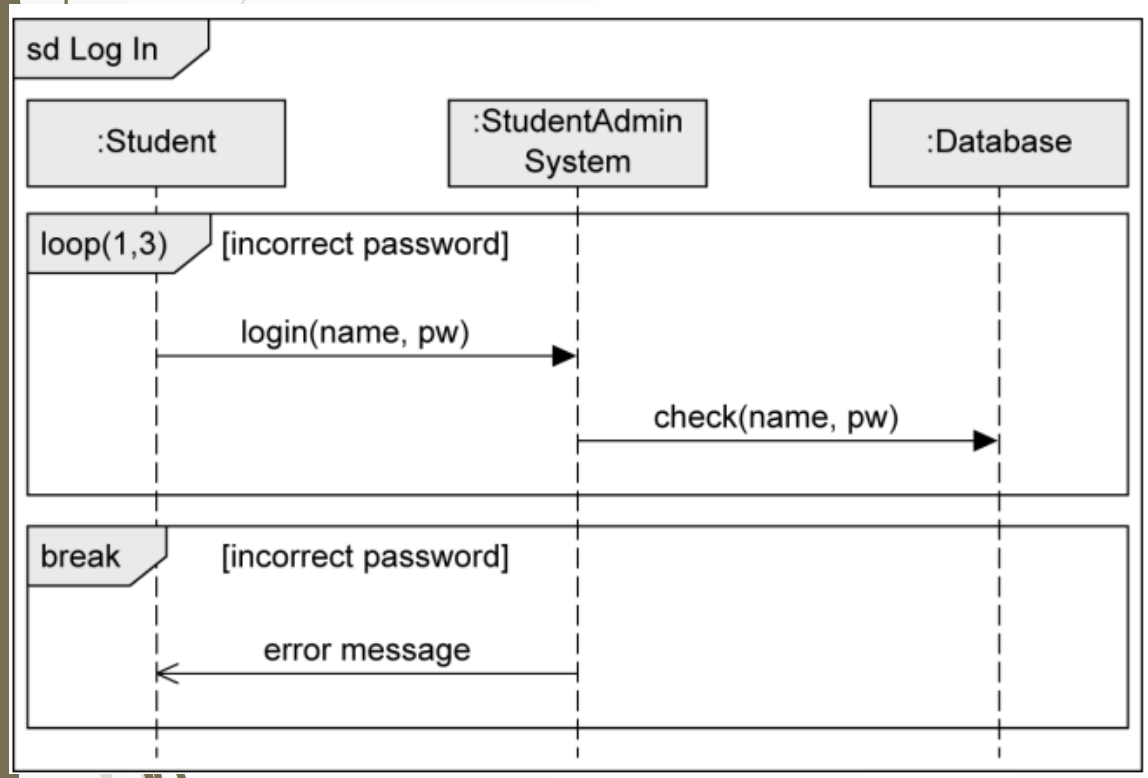


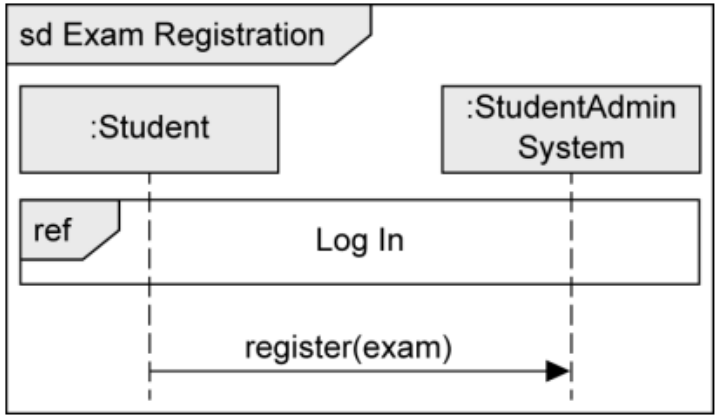
- Pour modéliser des interactions non valides
- Décrire des situations qui ne doivent pas se produire
- Exactement un opérande
- But
  - Mettre explicitement en évidence les erreurs fréquentes
  - Représenter des séquences pertinentes et incorrectes



Référence d'interaction : Réutilisation d'un fragment

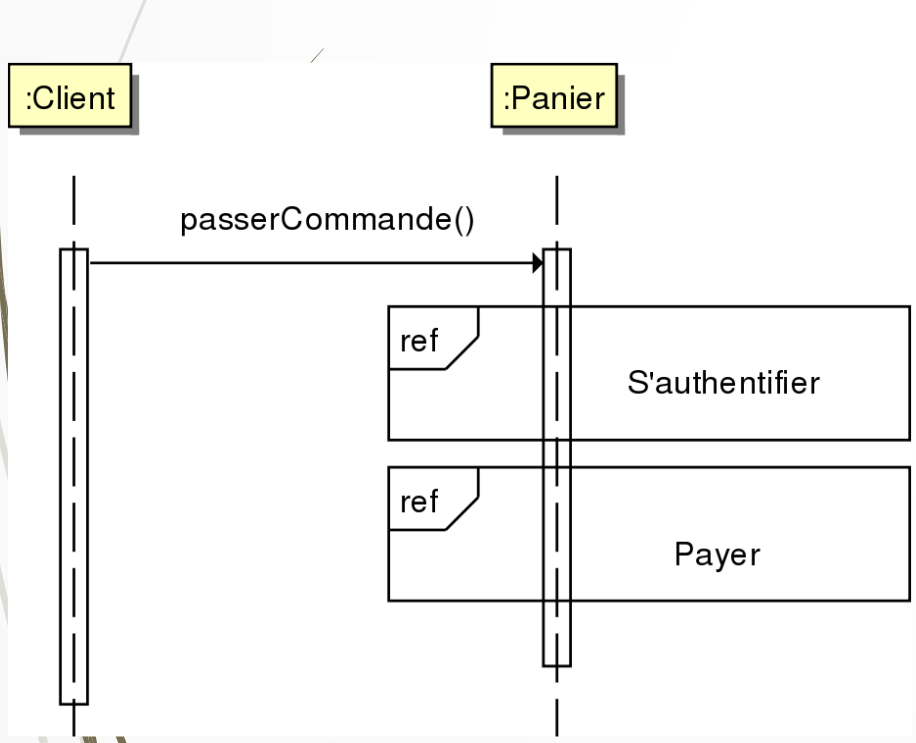
- Intègre un diagramme de séquence dans un autre diagramme de séquence

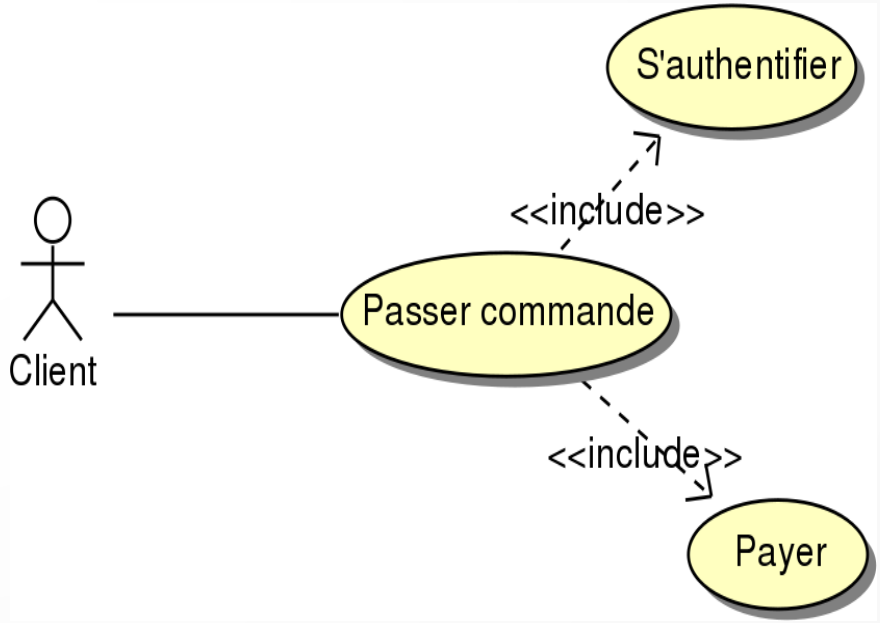




Réutilisation d'un fragment d'interaction

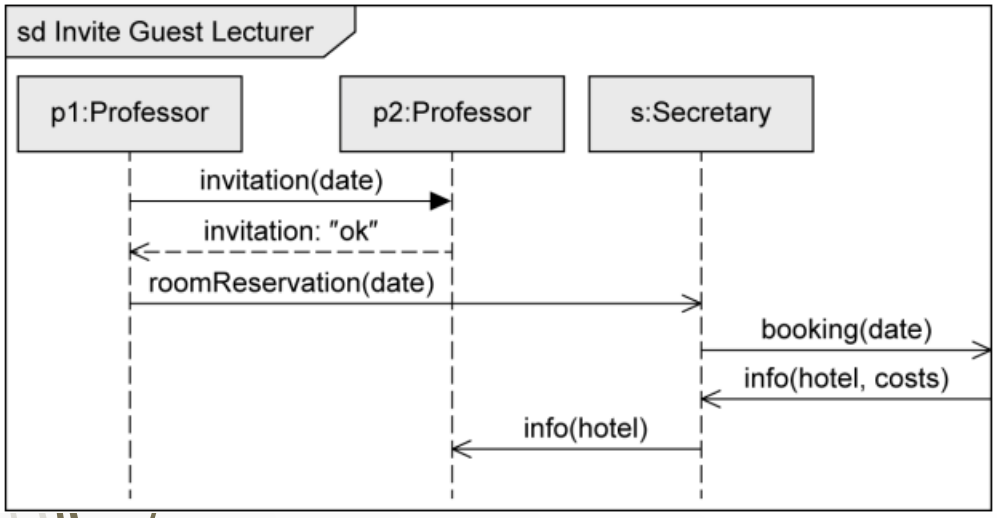
- Chaque cas d'utilisation donne lieu à un diagramme de séquence
- Les *inclusions* et les *extensions* sont des cas typiques d'utilisation de la réutilisation par référencement.

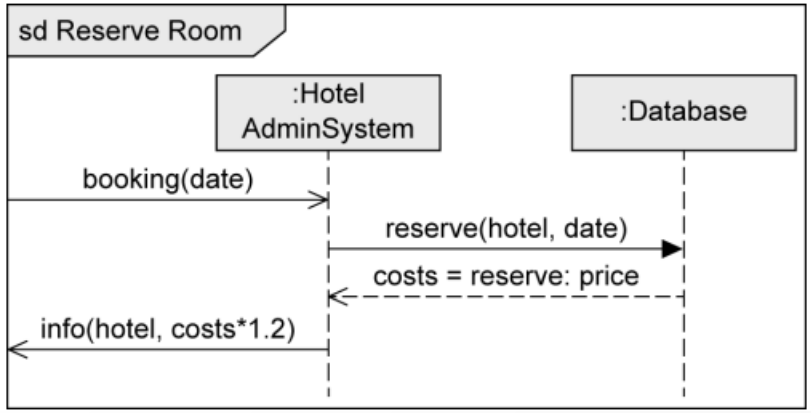




Portes

- Permet d'envoyer et de recevoir des messages au-delà des limites du fragment d'interaction

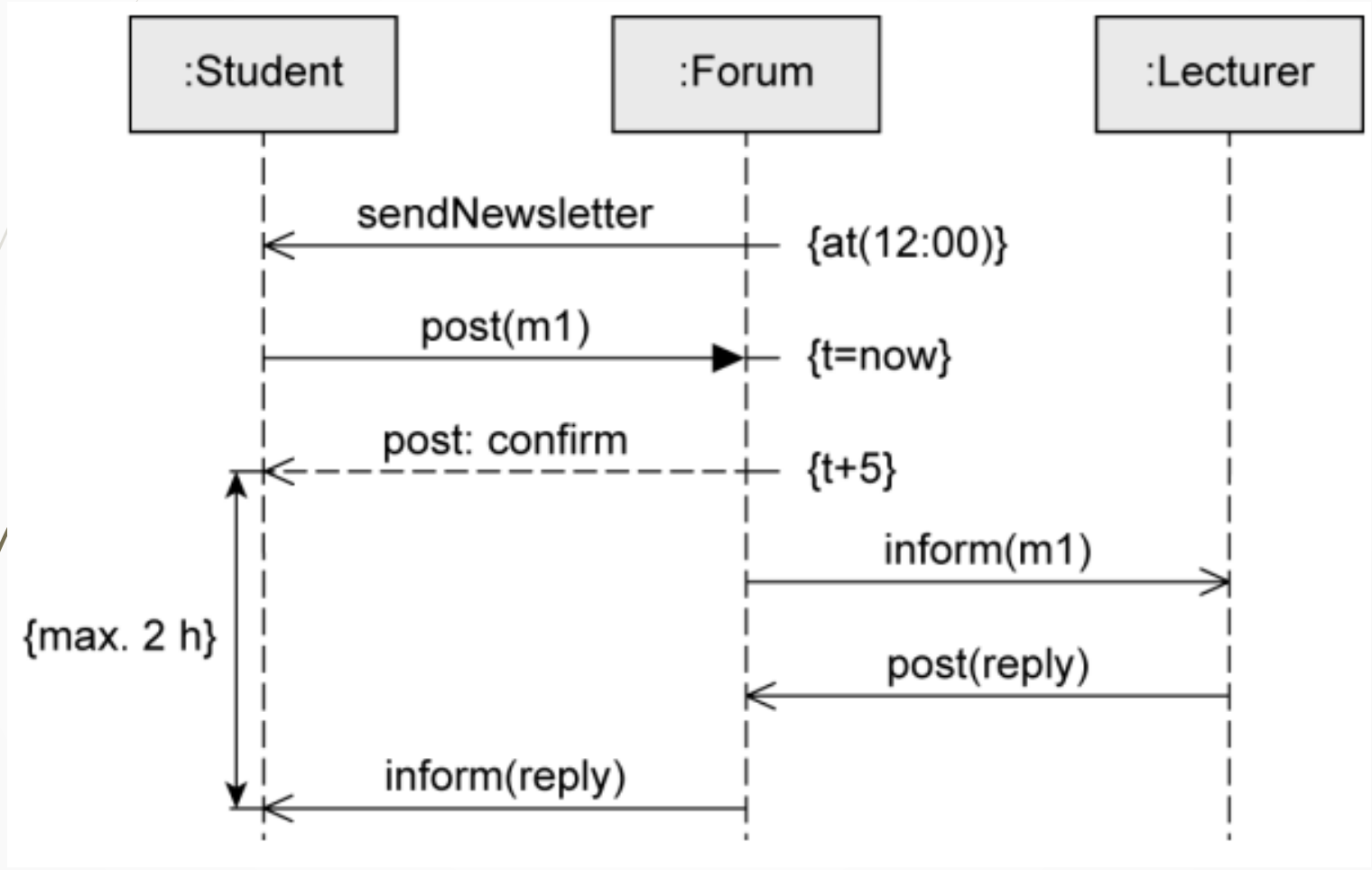




Contraintes temporelles

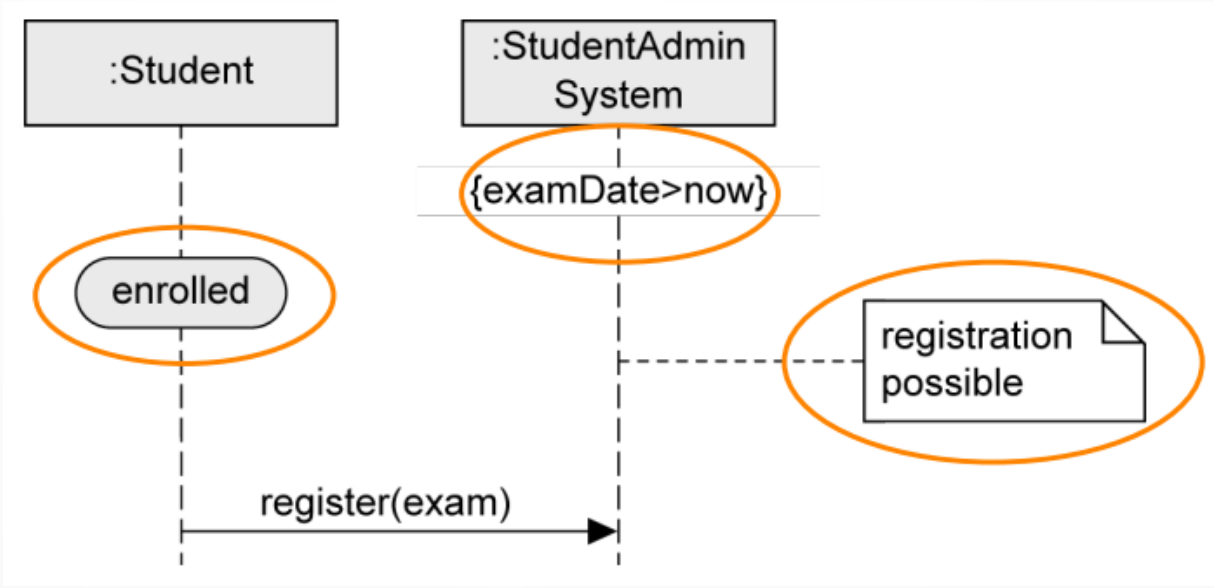
- Les types
  - Moment dans le temps pour l'occurrence de l'événement
    - Relatif : par exemple, **after** (5 sec)
    - Absolu : par exemple, **at** (12.00)
  - Délai entre deux événements
    - **{inférieur .. supérieur}**
    - Par exemple, {12.00 .. 13.00}
- Actions prédéfinies
  - **now** : heure actuelle
    - Peut être affecté à un attribut puis utilisé dans une contrainte temporelle
  - Durée : calcul de la durée d'émission d'un message

Contraintes temporelles : exemple



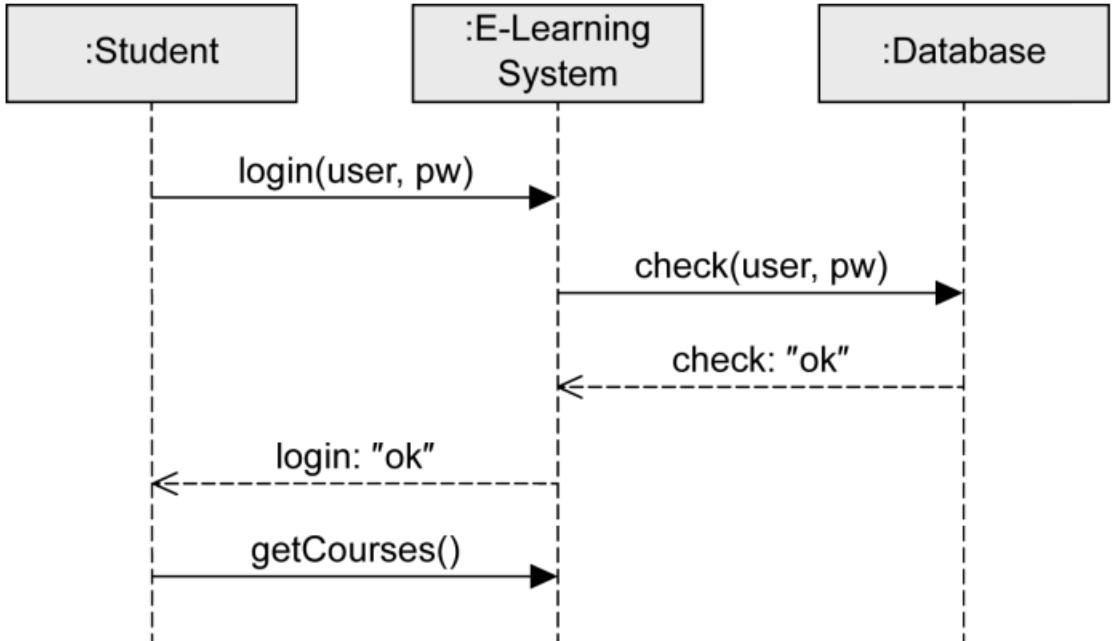
Etat invariant

- Affirme qu'une certaine condition doit être remplie à un certain moment
- Toujours affecté à une ligne de vie spécifique
- Évaluation avant que l'événement ultérieur ne se produise
- Si l'invariant d'état n'est pas vrai, soit le modèle, soit l'implémentation est incorrect
- Trois notations alternatives



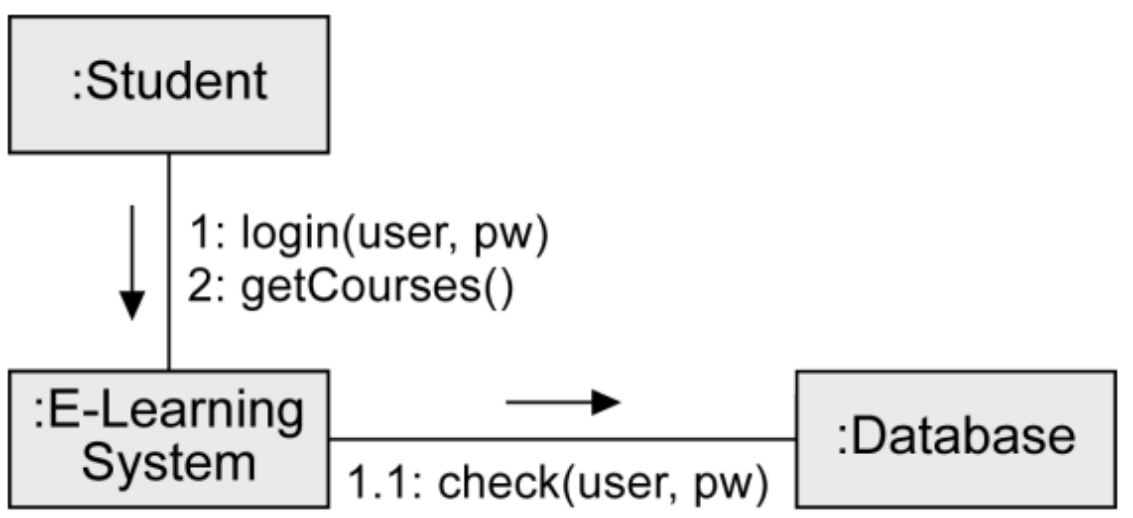
Quatre types de diagrammes d'interaction

- Basé sur les mêmes concepts
- Généralement équivalent pour des interactions simples, mais objectif différent
- Diagramme de **séquence**
  - Axe vertical: ordre chronologique
  - Axe horizontal: partenaires d'interaction



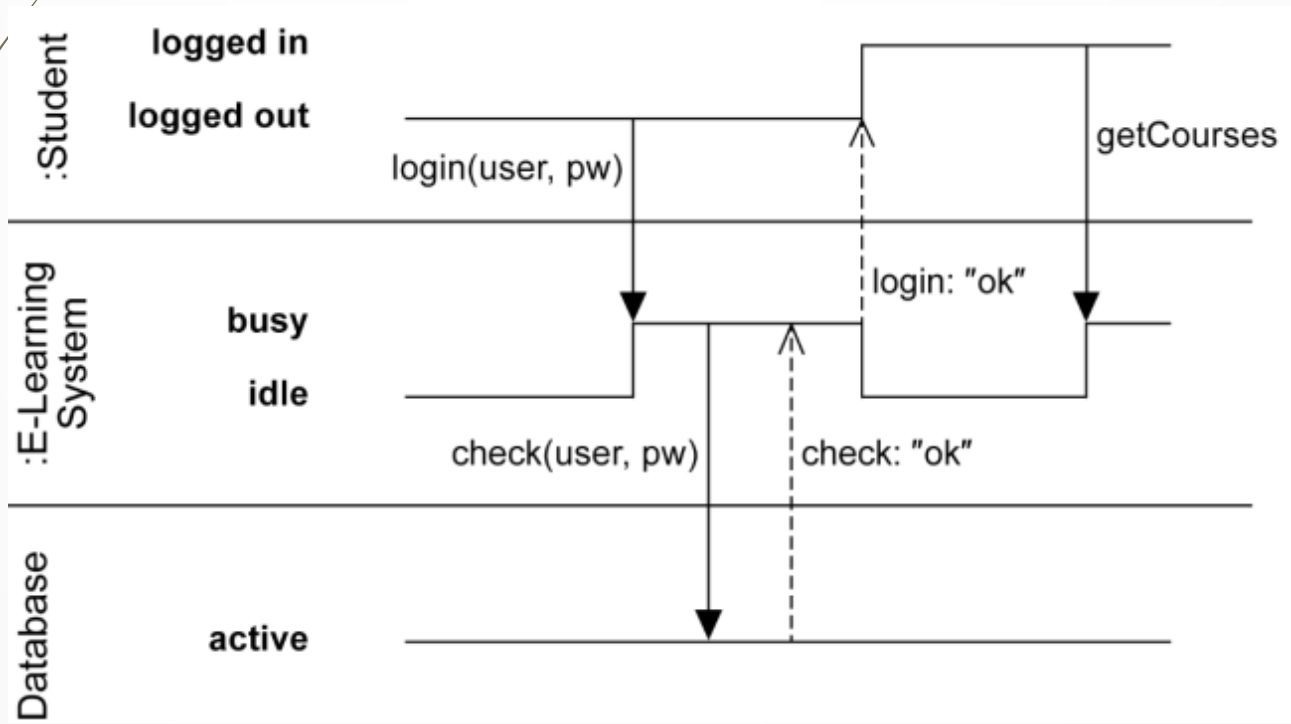
Quatre types de diagrammes d'interaction

- Diagramme de **communication**
  - Modélise les relations entre les partenaires de communication
  - Focus : Qui communique avec qui
  - Le temps n'est pas une dimension distincte
  - Ordre des messages via classification décimale



Quatre types de diagrammes d'interaction

- Diagramme de **timing**
  - Affiche les changements d'état des partenaires d'interaction qui résultent de l'apparition d'événements
  - Axe vertical : partenaires d'interaction
  - Axe horizontal : ordre chronologique



Quatre types de diagrammes d'interaction

- Diagramme de **Interaction overview**
  - Visualise l'ordre des différentes interactions
  - Permet de placer divers diagrammes d'interaction dans un ordre logique
  - Concepts de notation de base du diagramme d'activité