



# DIAGRAMMES D'INTERACTION: SÉQUENCES COMMUNICATION

---

MCOO-Chapitre 4

# Diagrammes d'interaction

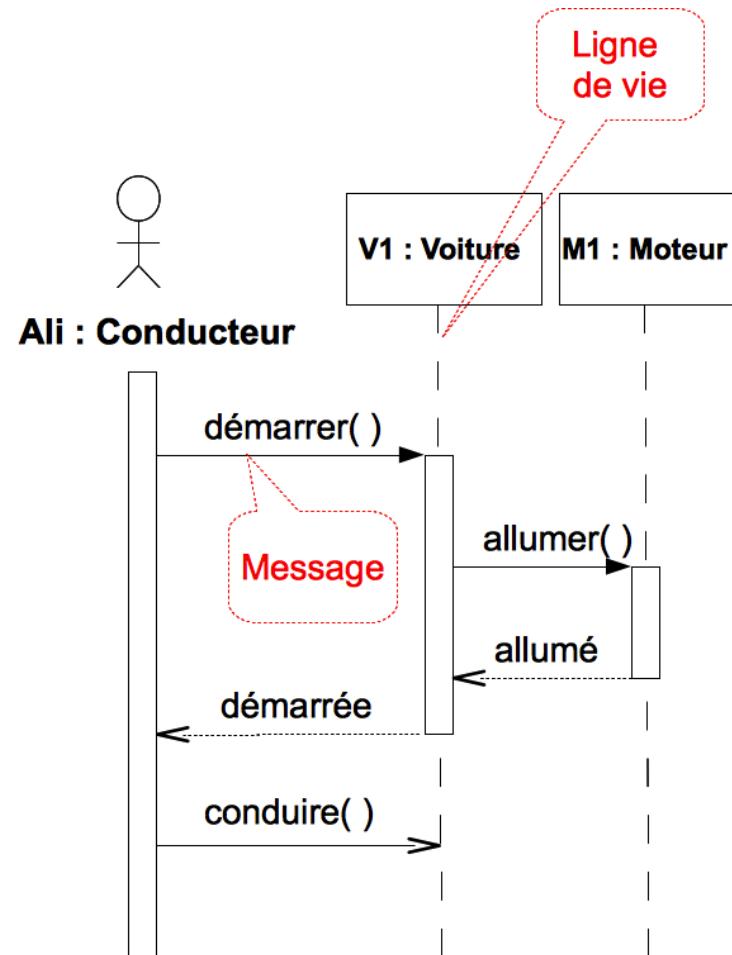
- Diagrammes dynamiques
- Aspect commun : **les messages** : sorte de communication entre deux entités (objet, acteur, sous-système)
- Différents diagrammes :
  - ✓ **Diagramme de Séquences**
  - ✓ **Diagramme de Communication**
  - ✓ Diagramme global d'Interaction
  - ✓ Diagramme de Temps

# Diagramme de Séquences

- Le diagramme d'interaction le plus commun
- Représentation temporelle de l'échange des messages entre les objets: séquencement des messages
- Représentation d'un seul scénario (avec la possibilité de combiner plusieurs)
- Écoulement du temps du haut vers le bas

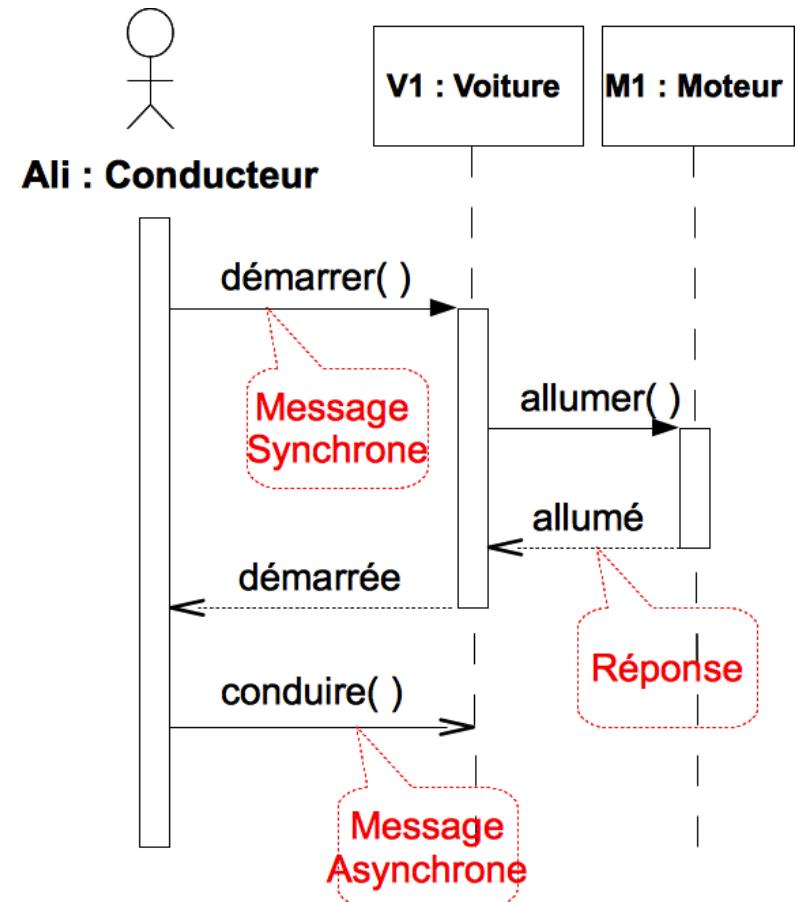
# Diagramme de Séquences : Contenu

- Lignes de vie
  - ✓ Rectangle + ligne pointillée
  - ✓ Etiquette [objet]:[classe]
- Messages
  - ✓ Communication entre les lignes de vie
  - ✓ Peuvent être :
    - Envoi de signal
    - Invocation d'une opération
    - Création ou destruction d'une instance



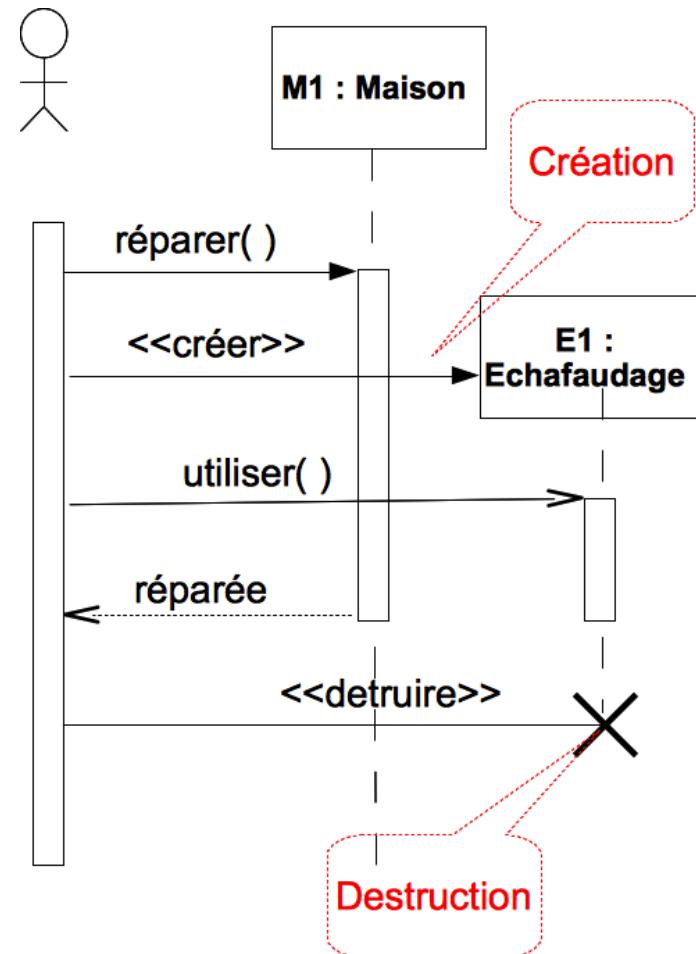
# Diagramme de Séquences : Messages (1)

- Message Asynchrone
  - ✓ N'attend pas de réponse
  - ✓ Ne bloque pas l'émetteur
  - ✓ **Exemple** : signal (interruption, évènement)
  - ✓ **Représentation** : Flèche en traits pleins et à l'extrémité ouverte
- Message Synchrone
  - ✓ Emetteur bloqué jusqu'à la réponse du récepteur
  - ✓ **Exemple** : invocation d'une opération
  - ✓ **Représentation** : Flèche en traits plats à l'extrémité pleine; suivie d'une flèche en pointillés



# Diagramme de Séquences : Messages (2)

- Création d'instance
  - ✓ Création d'un objet qui n'existe pas
  - ✓ **Représentation** : flèche qui pointe sur le sommet d'une ligne de vie
- Destruction d'instance
  - ✓ Destruction d'un objet qui n'existera plus
  - ✓ N'est pas toujours provoquée par un message
  - ✓ **Représentation** : une croix qui marque la fin de la ligne de vie de l'objet détruit



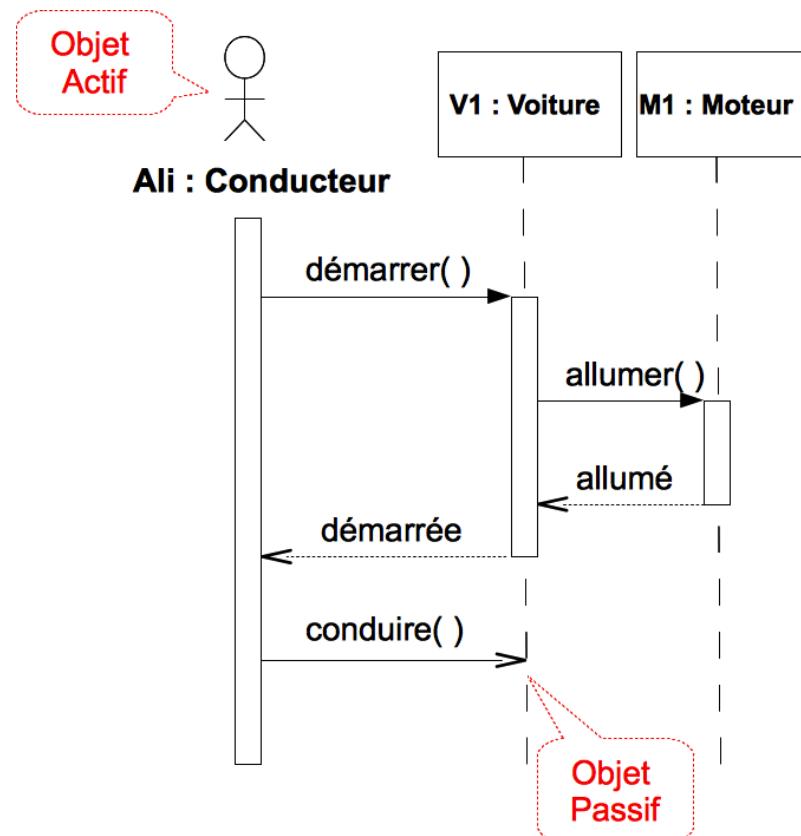
# Diagramme de Séquences : Objets Actif et Passifs

## ■ Objet actif

- ✓ Initie et contrôle le flux d'activités
- ✓ **Représentation** : un rectangle à la place de la ligne de vie verticale

## ■ Objet passif

- ✓ A besoin d'un flux d'activités pour pouvoir exécuter une méthode
- ✓ À l'exécution d'une méthode, un rectangle blanc est placé sur la ligne en pointillés

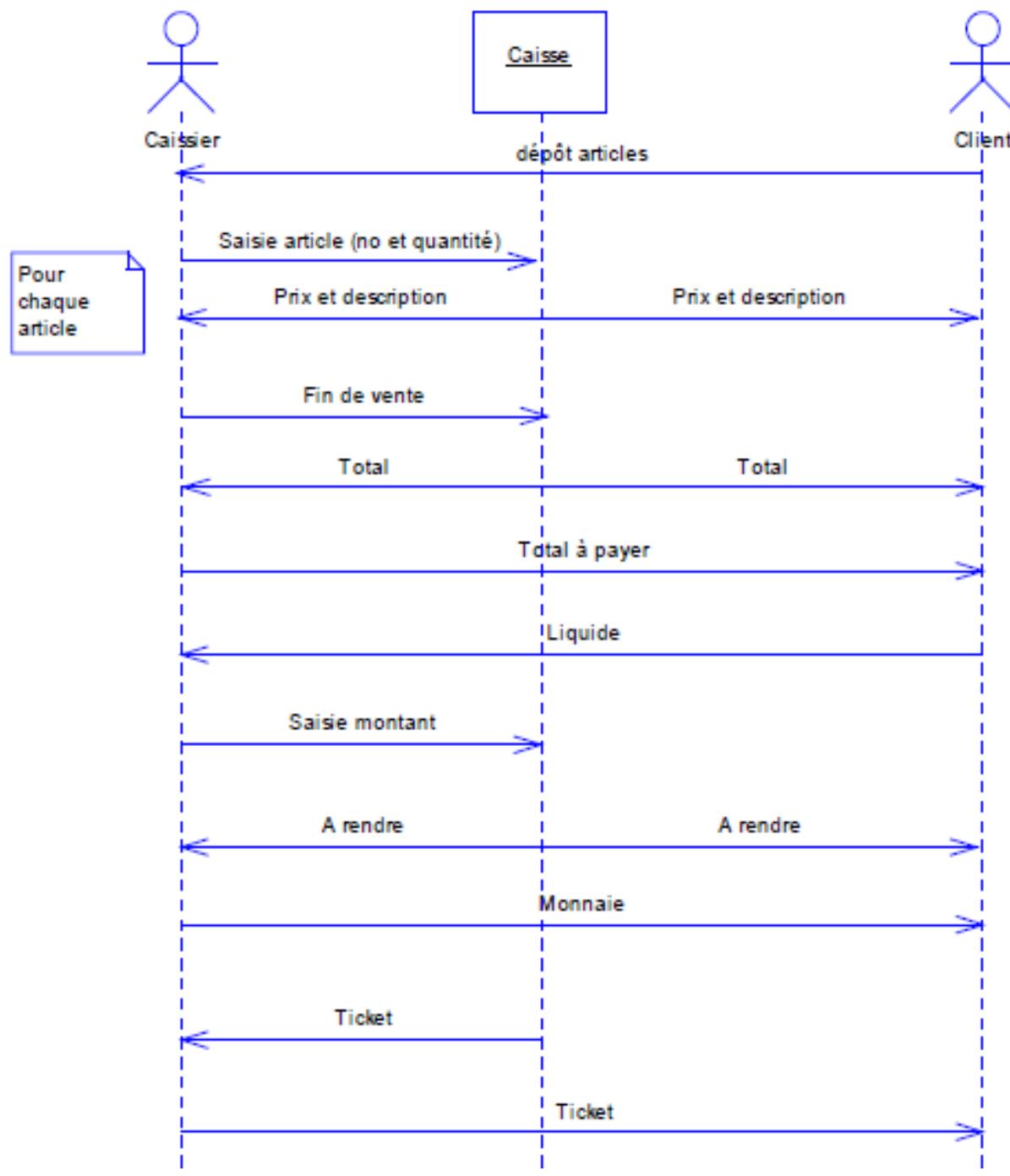


# Exercice D'application

Le déroulement normal d'utilisation d'une caisse de supermarché est le suivant :

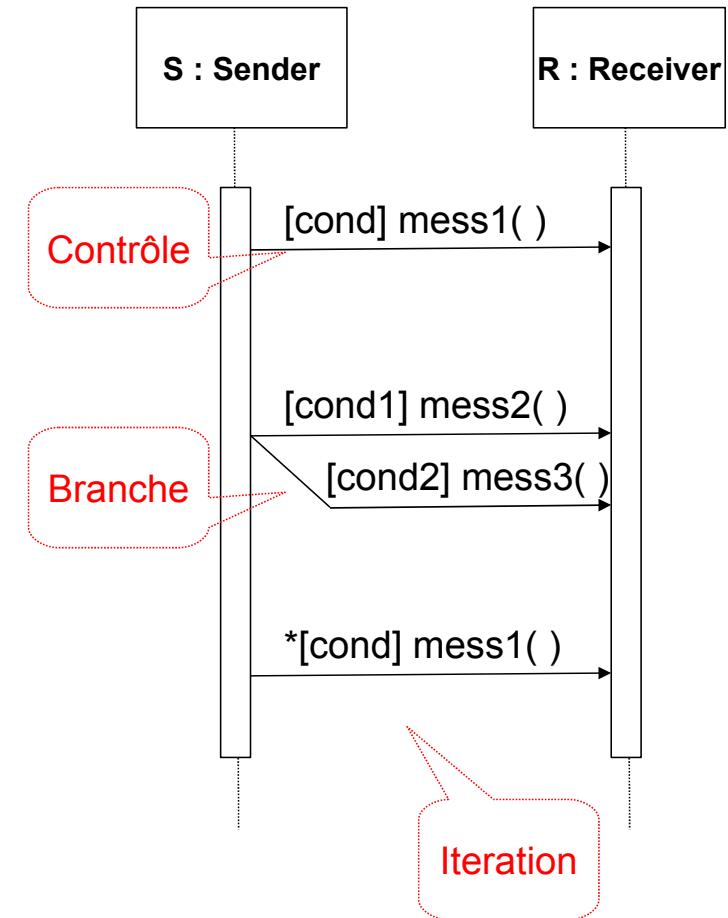
- un client arrive à la caisse avec ses articles à payer
- le caissier enregistre le numéro d'identification de chaque article, ainsi que la quantité si elle est supérieure à 1
- la caisse affiche le prix de chaque article et son libellé
- lorsque tous les achats sont enregistrés, le caissier signale la fin de la vente
- la caisse affiche le total des achats
- le caissier annonce au client le montant total à payer
- le client choisit son mode de paiement
  - ✓ liquide : le caissier encaisse l'argent, la caisse indique le montant à rendre au client
  - ✓ chèque : le caissier note le numéro de pièce d'identité du client
  - ✓ carte de crédit : la demande d'autorisation est envoyée avant la saisie
- la caisse enregistre la vente et l'imprime
- le caissier donne le ticket de caisse au client

*Modéliser cette situation à l'aide d'un diagramme de séquence en ne prenant en compte que le cas du paiement en liquide.*



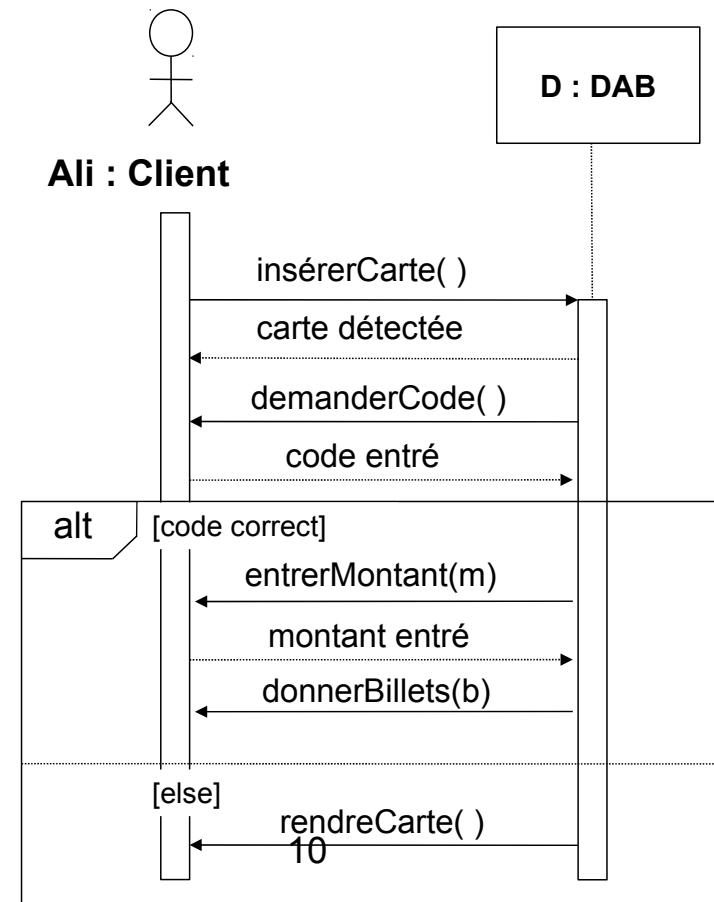
# Structures de Contrôle

- Structure de contrôle
  - ✓ mess1 envoyé ssi la condition de garde [entre crochets] est respectée
- Branche
  - ✓ On envoie soit mess2 soit mess3, selon les conditions
- Itération
  - ✓ Le mess4 est envoyé tant que la condition est vraie



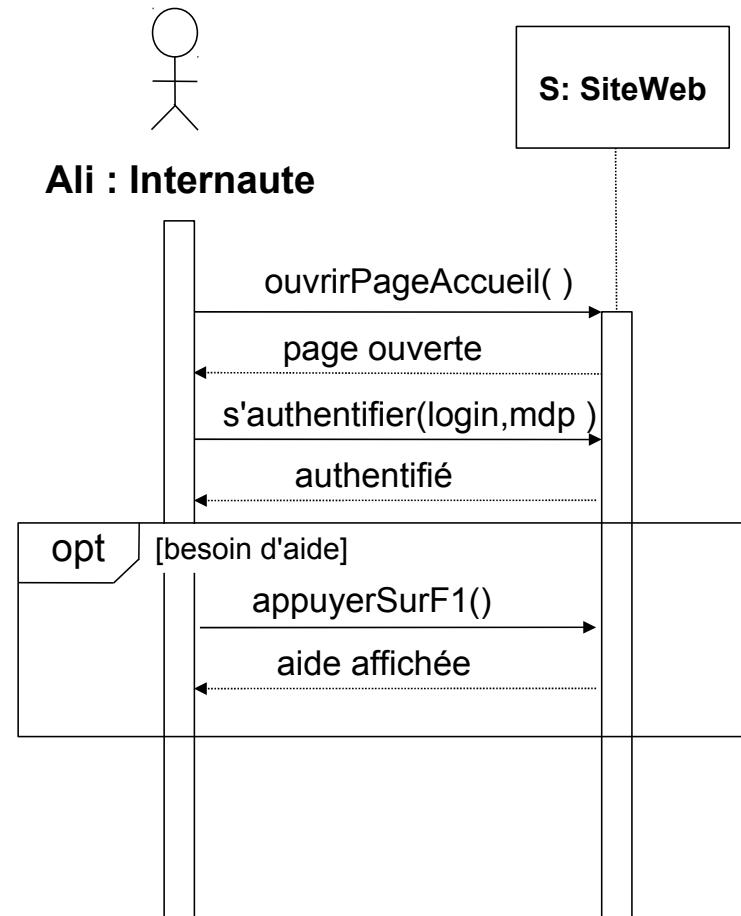
# Opérateur « Alternative »

- Alternative (ou *alt*)
  - ✓ Opérateur conditionnel
    - Équivalent d'une exécution à choix multiples (*switch*)
  - ✓ Peut posséder plusieurs opérandes, chacune détient une condition de garde
  - ✓ Absence de condition de garde: condition vraie
  - ✓ Condition *else*: vraie si aucune autre condition n'est vraie



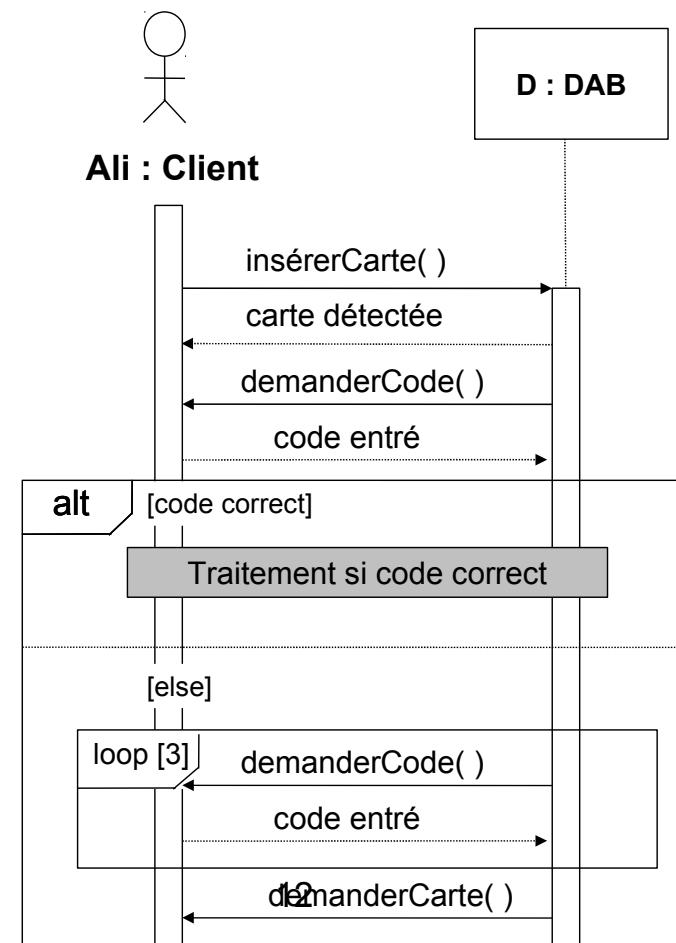
# Opérateur « Option »

- Option (ou *opt*)
  - ✓ Représente un comportement qui peut se produire ou pas.
  - ✓ Équivalent à un *alt* à une seule branche et sans *else*



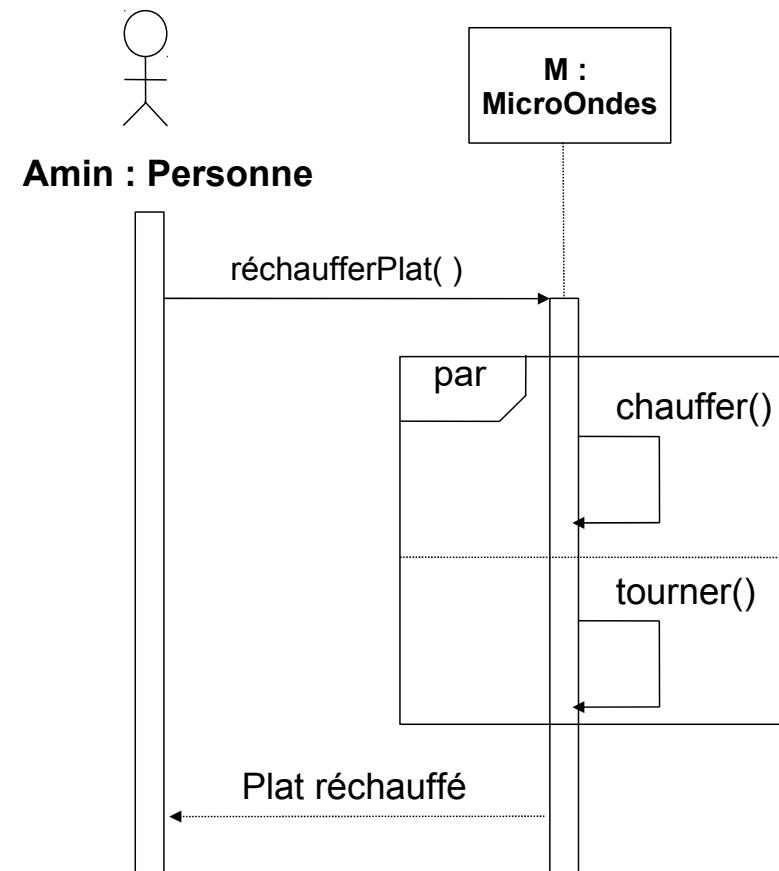
# Opérateur « Loop »

- Loop
  - ✓ Équivalent d'une boucle *for*
  - ✓ Décrit des interactions qui s'exécutent en boucle
  - ✓ La condition (garde) indique le nombre de répétitions (min et max) ou une condition booléenne à respecter



# Opérateur « Parallèle »

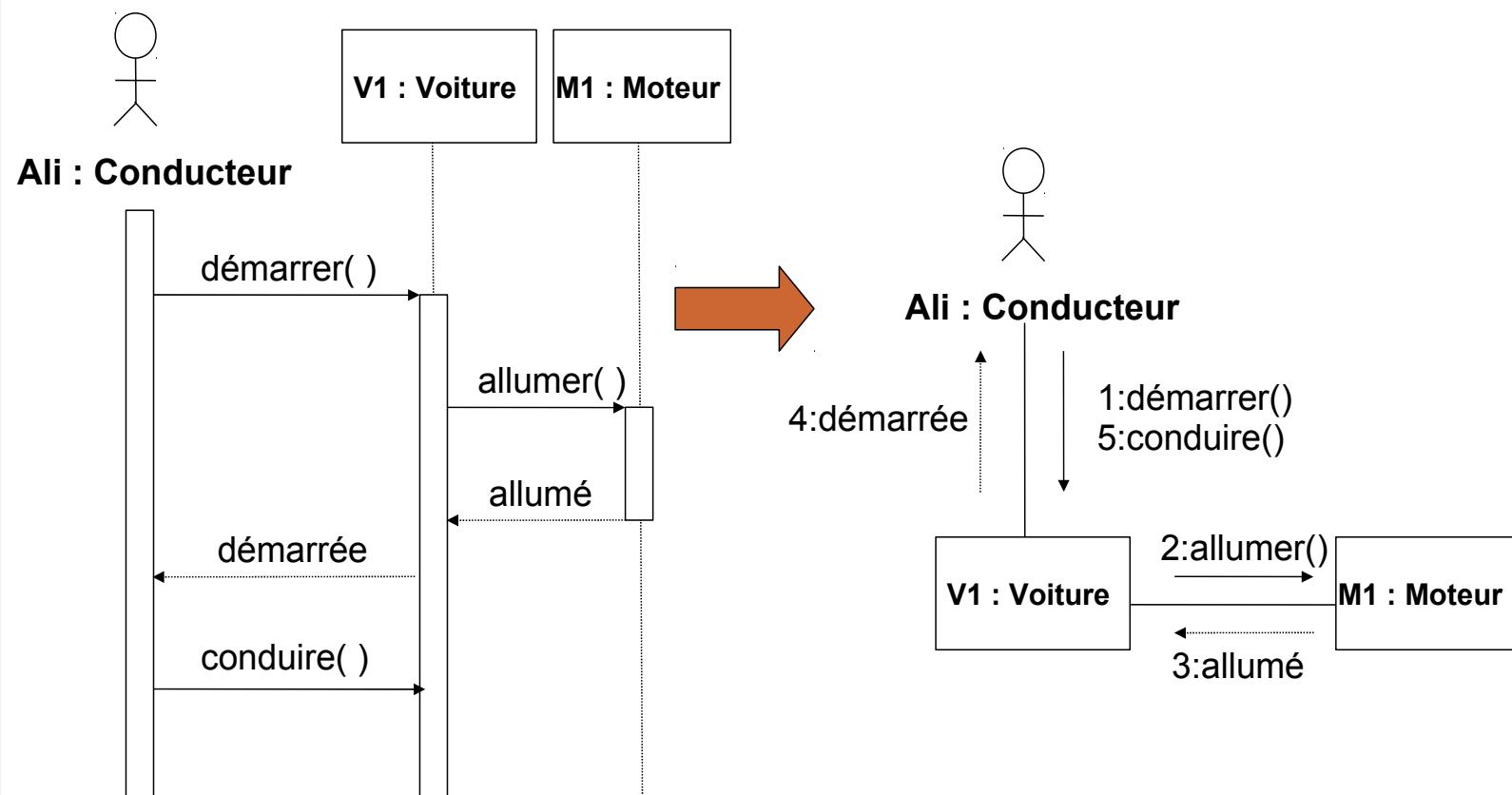
- Parallèle (ou *par*)
  - ✓ A au moins 2 sous-fragments exécutés simultanément
  - ✓ Simule une exécution parallèle



# Diagramme de Communication

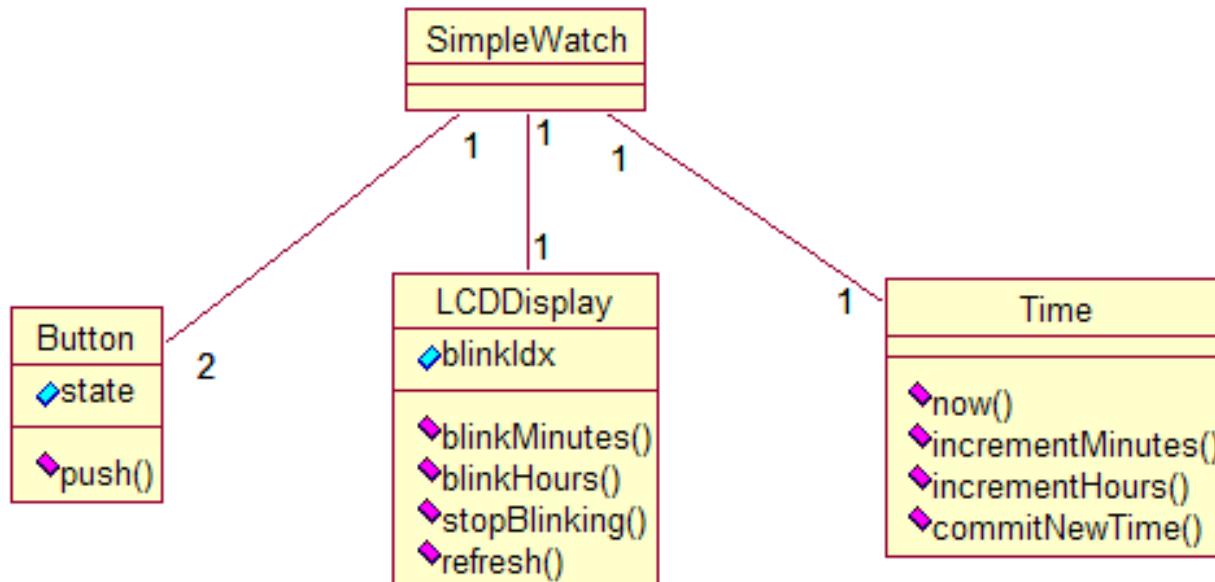
- Appelé diagramme de *collaboration* avant UML2
- Montre les interactions entre objets
- Insiste sur la structure spatiale pour mettre en collaboration un groupe d'objets
  - ✓ Messages : liens reliant les objets
  - ✓ Temps : représenté implicitement par une numérotation des messages

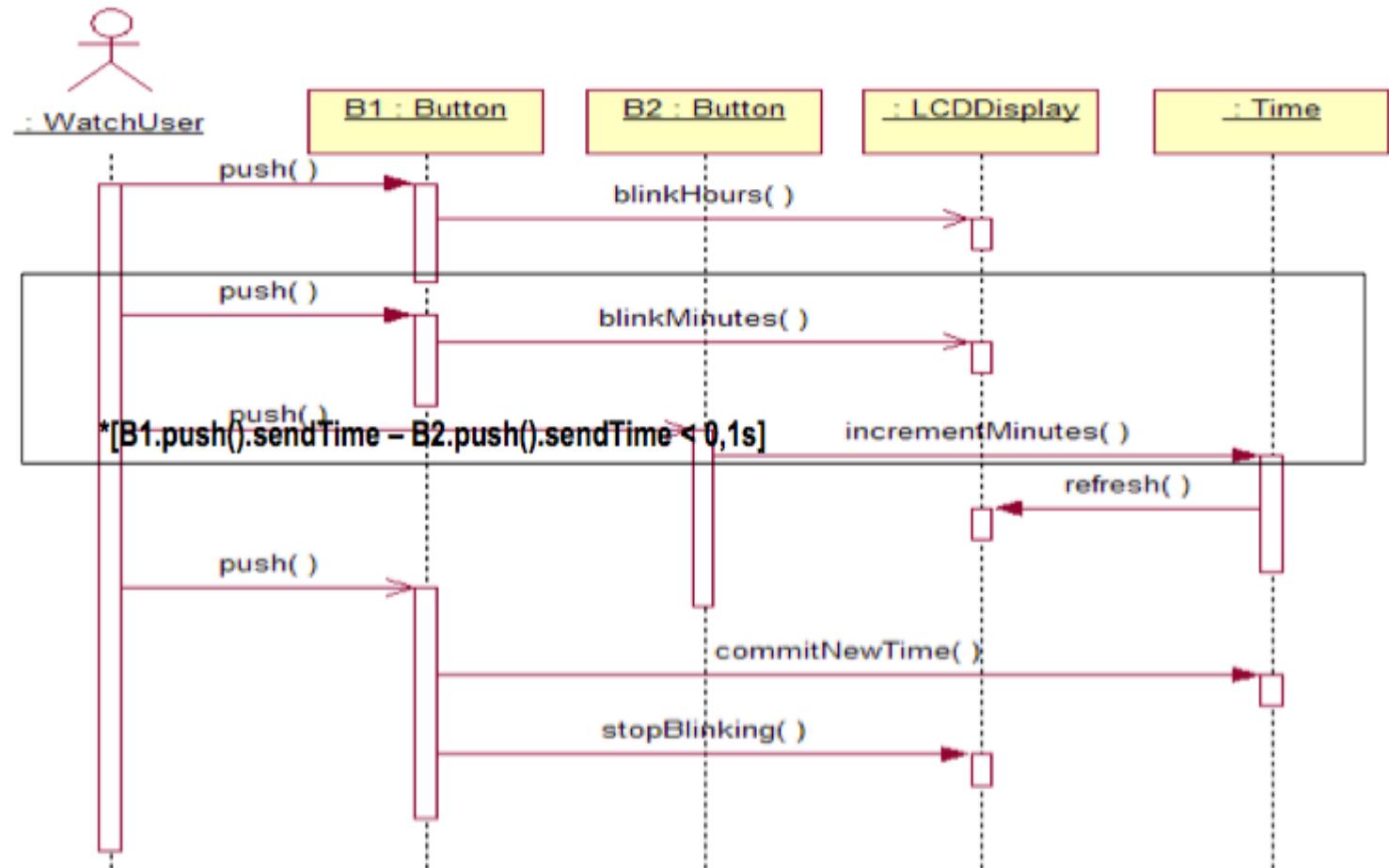
# Exemple de Diagramme de Communication

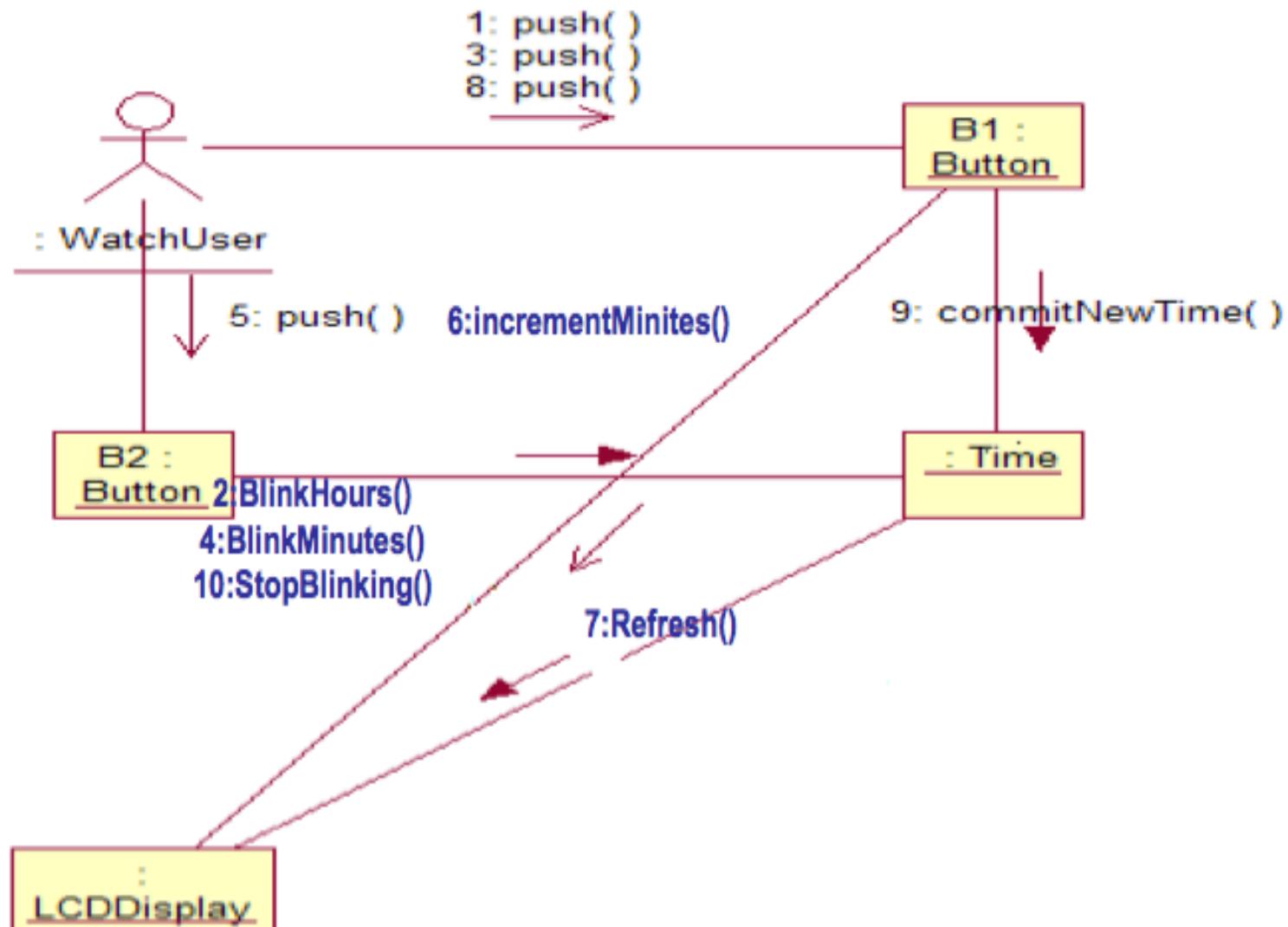


# Application : Montre

- A partir du diagramme de classes suivant :
  - ✓ Rédiger un diagramme de séquences pour modéliser un scénario où un utilisateur voudrait régler les minutes sur sa montre :
    - En appuyant 2 fois sur le bouton 1, il accède au réglage des minutes (l'heure clignote puis la minute clignote). Ensuite, avec le bouton 2, il incrémente les minutes d'une valeur, le LCD display est rafraîchi après chaque pression. En appuyant sur les deux boutons à la fois, l'heure est enregistrée et l'affichage s'arrête de clignoter
  - ✓ En déduire le diagramme de communication



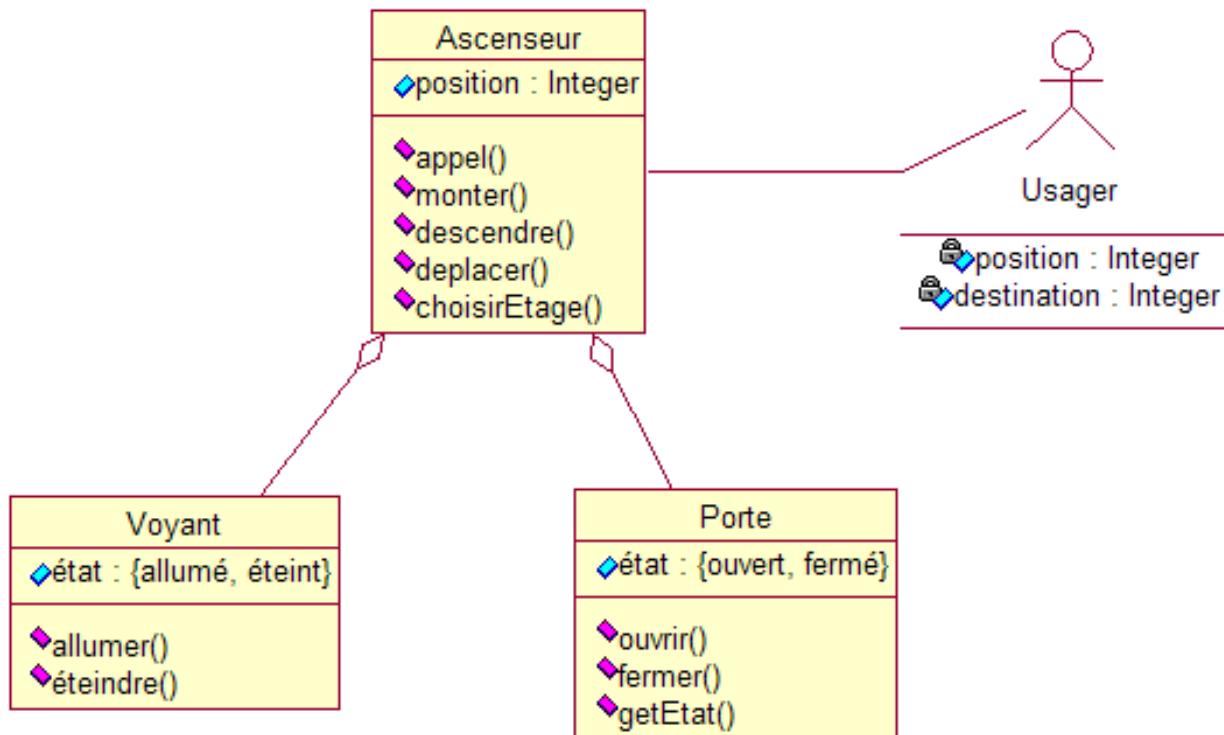


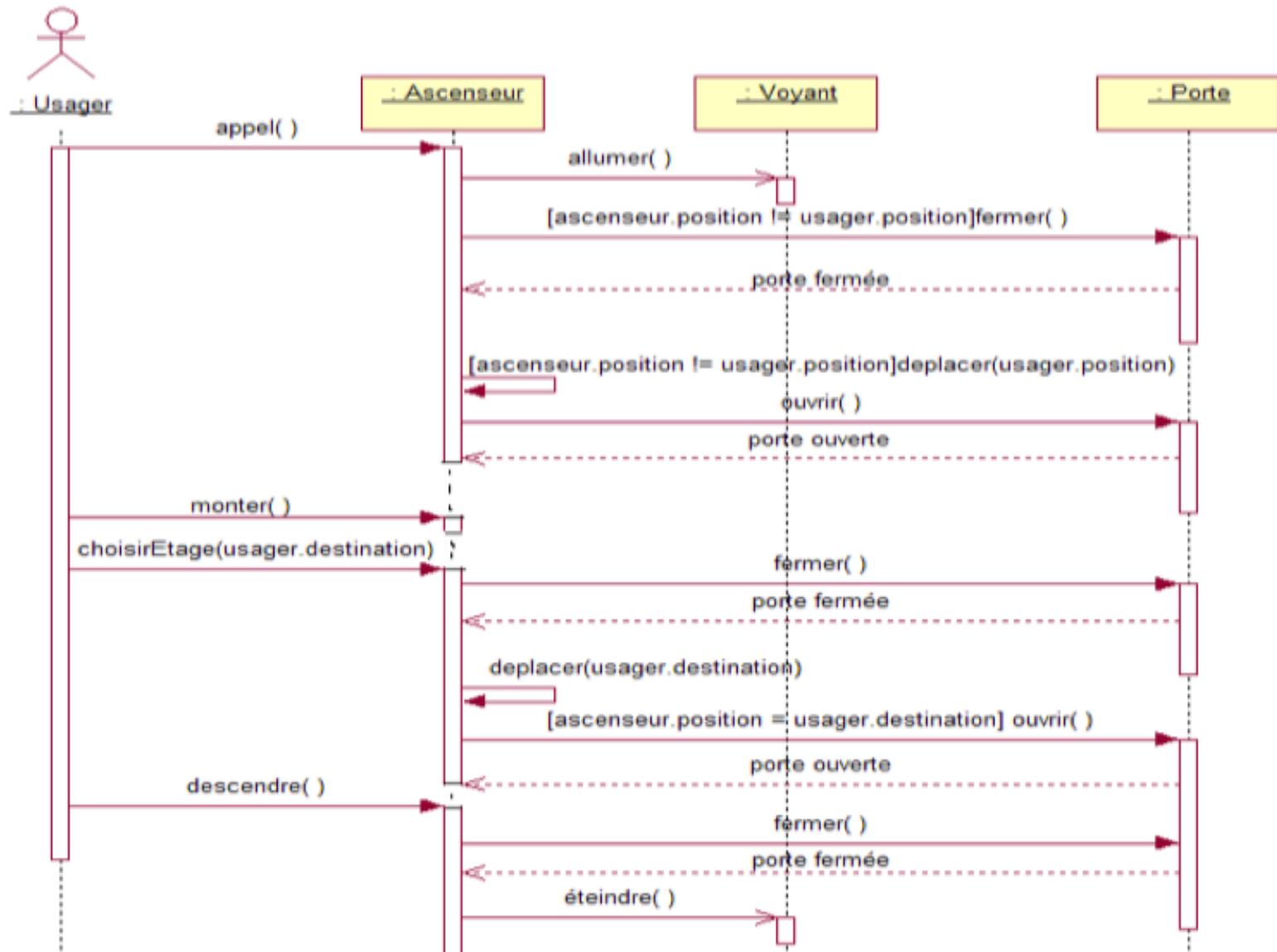


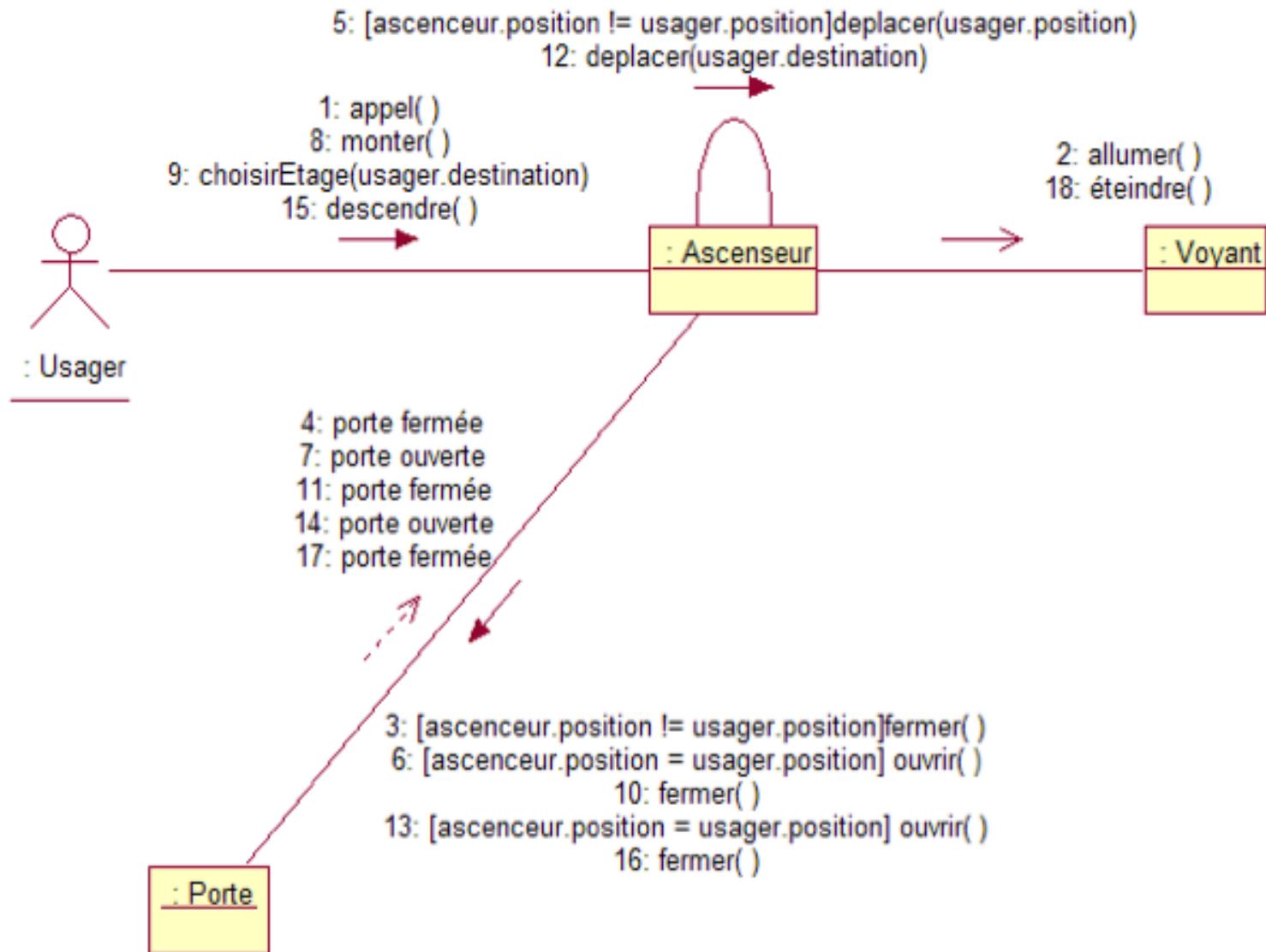
# Application: Ascenseur

- D'après le diagramme de classes :

- ✓ Rédiger le diagramme de séquences pour modéliser un scénario où un usager veut monter en utilisant l'ascenseur
- ✓ En déduire le diagramme de communication







# *Exemple*

---

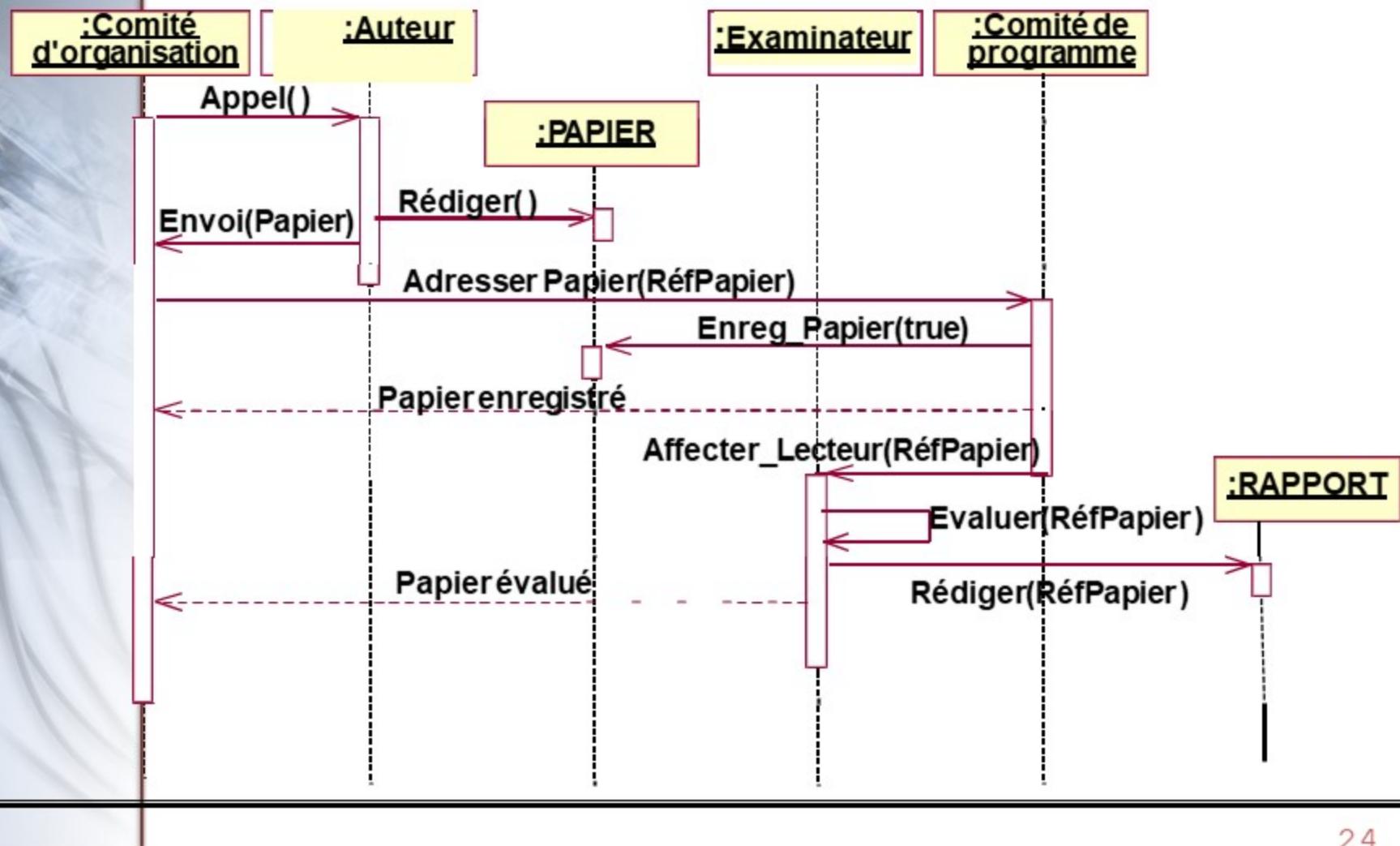
## D **Organisation d'une conférence internationale.**

**Considérons les Règles de gestion suivantes :**

- Le comité d'organisation adresse un **appel à communication** aux auteurs intéressés par les thèmes de la conférence.
- Un **auteur soumet un papier** décrivant ses travaux dans l'un des thèmes de la conférence.
- Un **comité de programme est responsable de l'affectation des papiers reçus à des examinateurs spécialisés pour évaluation.**
  - Chaque **examinateur évalue les papiers attribués par le comité de programme.**

# Exemple

D Le diagramme de séquence correspondant est :



# TD: Diagramme de séquences

# Ex1:Messages synchrones et asynchrones

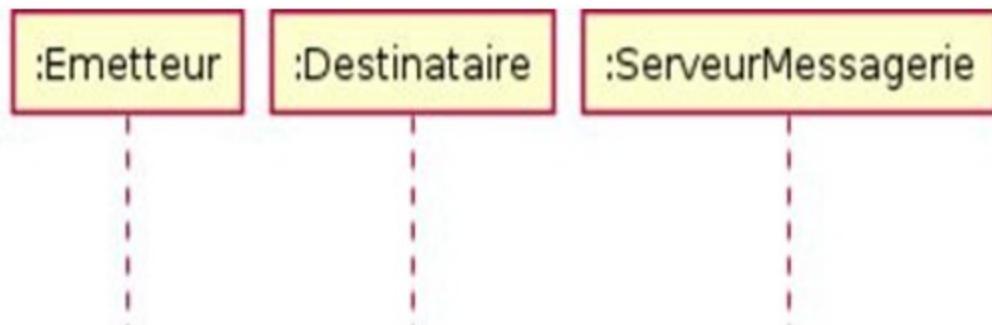
Un système de messagerie procède par envois de messages directement de l'émetteur vers le destinataire, sans intermédiaire.

1. Précisez le type de message transitant de l'émetteur au destinataire dans le diagramme ci-dessous.

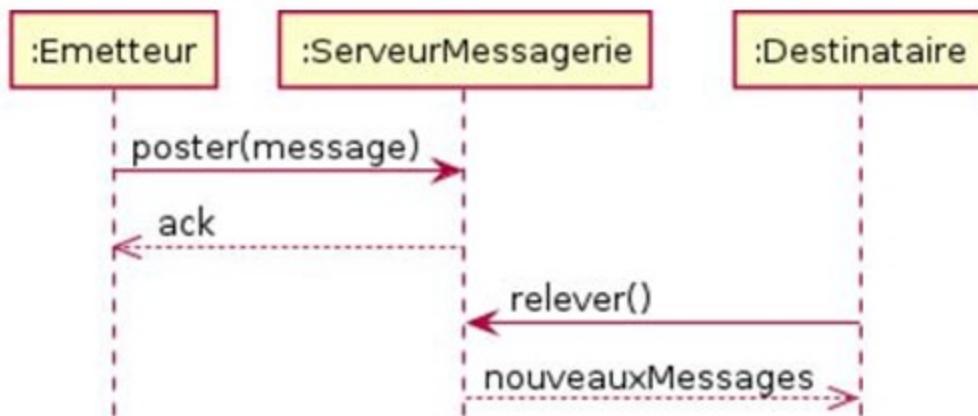




On suppose maintenant que l'on dispose d'un serveur de messagerie faisant permettant de stocker temporairement les messages. Les lignes de vie sont donc les suivantes :



2. Représentez les messages modélisant l'envoi et la réception d'un message.
3. Qu'est-il nécessaire de prévoir dans le diagramme de classes pour que cette solution soit correcte ?



ServeurMessagerie

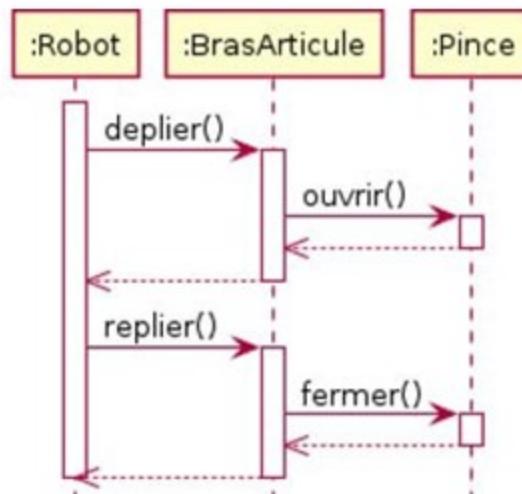
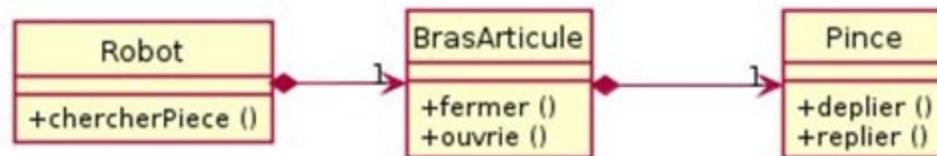
+poster(message:String)  
+relever():String[\*]

# Ex2: Séquences et activités

Considérons le pseudo-code suivant :

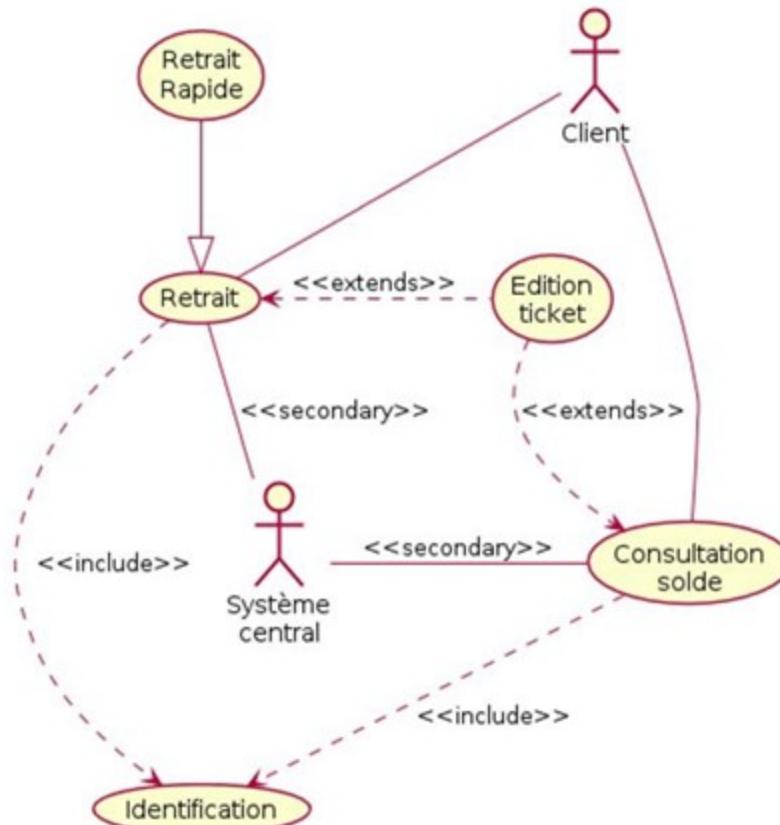
```
1 class Robot {
2     private :
3         BrasArticule brasArticule ;
4     public :
5         void chercherPiece() {
6             brasArticule.deplier() ;
7             brasArticule.replier() ;
8         }
9     }
10
11    class BrasArticule {
12        private :
13            Pince pince ;
14        public :
15            void deplier() {
16                ...
17                pince.ouvrir() ;
18            }
19            void replier() {
20                ...
21                pince.fermer() ;
22            }
23        }
24
25    class Pince {
26        private :
27        ...
28    public :
29        void fermer() { ... }
30        void ouvrir() { ... }
31    }
32
33 Début programme principal
34     Robot robot ;
35     robot.chercherPiece() ;
36 Fin programme principal
```

1. Donnez un diagramme de classes correspondant au code ci-dessus.
2. Donnez un diagramme de séquences pour modéliser la séquence d'activités après l'appel de `robot.chercherPiece()` dans le programme principal.  
Représentez les barres d'activités des différentes lignes de vie.



# Ex3: Documentation de cas d'utilisation

- Les diagrammes de séquence sont souvent utilisés pour documenter des cas d'utilisation. Soit le diagramme suivant pour modéliser les fonctionnalités attendues d'un DAB (distributeur automatique de banque) :



- La description textuelle du cas d'utilisation Retrait inclut les éléments suivants :
  - 1. Le client demande un retrait
  - 2. On procède à l'identification du client
  - 3. Le DAB demande au client quel compte débiter
  - 4. A l'invitation du DAB, le client donne le montant à débiter
  - 5. Le DAB procède aux vérifications nécessaires auprès du système central
  - 6. Le client est informé des suites données à sa demande
  - 7. Le DAB demande au client s'il souhaite un ticket
  - 8. Si le retrait a été autorisé, le DAB restitue la carte bancaire
  - 9. Immédiatement après que le client ait pris la carte, le DAB met les billets à disposition
  - 10. Retrait autorisé ou non, si le client a demandé un ticket, il est édité
- On suppose qu'il existe déjà des diagrammes de cas d'utilisation intitulés Identification et EditionTicket documentant les cas d'utilisation correspondants.
  1. Donnez un diagramme de séquences formalisant en UML le scénario ci-dessus.

