UML : Les diagrammes de séquence

 tv <tvaira@free.fr> - v.1.2

Diagrammes d'interactions

Les **diagrammes d'interactions** englobent deux types de diagrammes UML :

- les diagrammes de **séquence**,
- les diagrammes de **communication** (appelés diagramme de collaboration en UML 1),

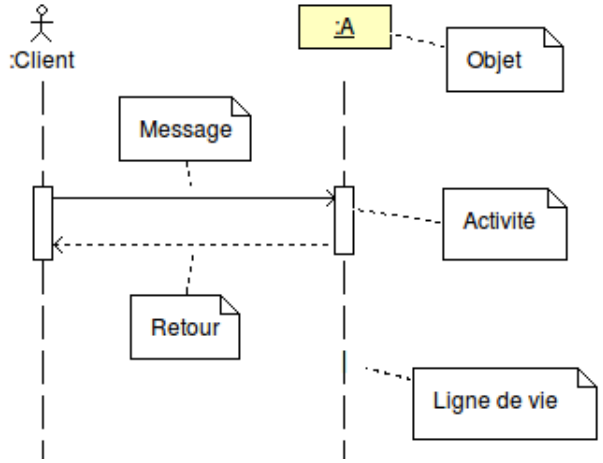


FIGURE 1 – *Diagramme de séquence*

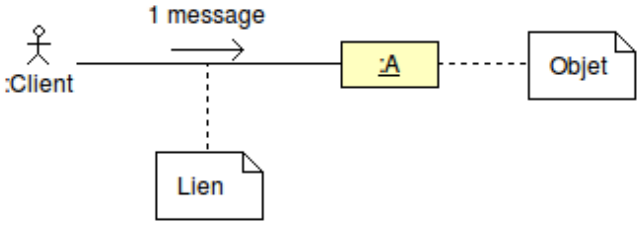


FIGURE 2 – *Diagramme de collaboration*

Chaque diagramme a ses points forts et ses points faibles. Lorsque le nombre d'objets devient conséquent, le diagramme de collaboration sera préférable. En revanche, la lecture du séquencement des messages sera plus difficile.

Diagramme de séquence

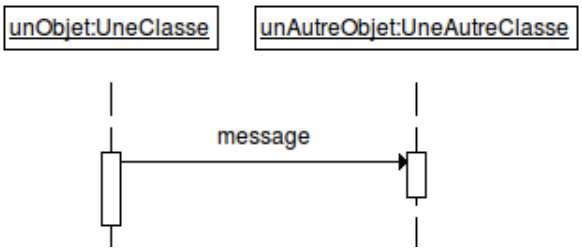
Un **diagramme de séquence** est un diagramme d'interaction dont le but est de **décrire comment les objets collaborent au cours du temps et quelles responsabilités ils assument**. Il décrit **un scénario d'un cas d'utilisation**.

Un **diagramme de séquence** représente donc les **interactions entre objets**, en insistant sur la chronologie des envois de **message**. C'est un diagramme qui représente la structure **dynamique** d'un système car il utilise une représentation temporelle. Les **objets**, intervenant dans l'interaction, sont matérialisés par une « **ligne de vie** », et les messages échangés au cours du **temps** sont mentionnés sous une forme textuelle.

Messages

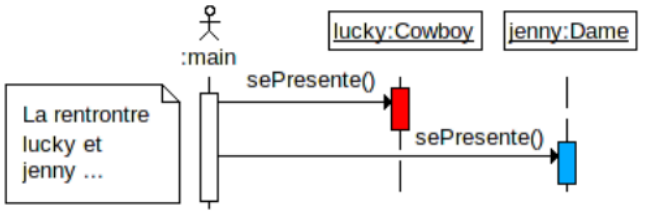
Un objet est une structure de données encapsulées qui répond à un **ensemble de messages**. Cette **structure de données (ses attributs) définit son état** tandis que l'**ensemble des messages (ses méthodes) décrit son comportement**.

L'ensemble des messages forme ce que l'on appelle l'**interface de l'objet**. Les objets interagissent entre eux en **s'échangeant des messages**.



La réponse à la réception d'un message par un objet est appelée **une méthode**. Une **méthode est donc la mise en oeuvre du message** : elle décrit la réponse qui doit être donnée au message.

UML



C++

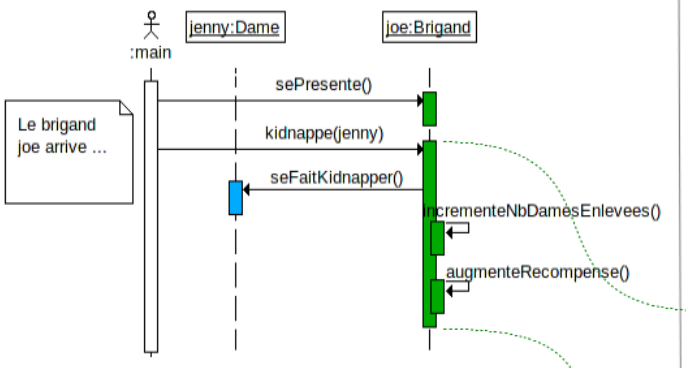
```
Cowboy lucky("Lucky Luke", "coca-cola");
Dame jenny("Jenny");

lucky.sePresente();

jenny.sePresente();
```

Une **activité** représente l'**exécution d'une méthode** :

UML



C++

```
Dame jenny("Jenny");
Brigand joe("Joe");

joe.sePresente();

joe.kidnappe(jenny);


void Brigand::kidnappe(Dame &dame)
{
  dame.seFaitKidnapper();
  incrementeNbDamesEnlevees();
  augmenteRecompense();
}
```


Un message retourne implicitement une valeur (ou aucune). Il est possible de préciser **explicitement** la valeur retournée par un message.

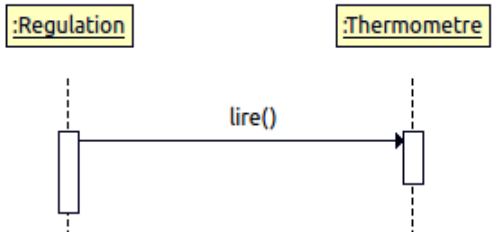


FIGURE 3 – *Retour implicite*

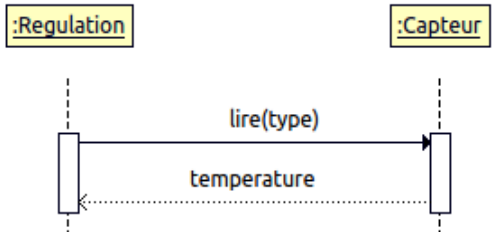


FIGURE 4 – *Retour explicite*

On distingue deux types de message :
- **synchrone** : l'objet émetteur se bloque en attendant la réponse de l'objet récepteur du message
- **asynchrone** : l'objet émetteur n'attend pas la réponse de l'objet récepteur du message et continue son activité

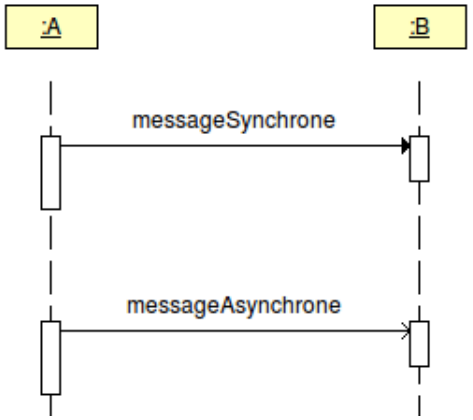


Diagramme de séquence système

Un **diagramme de séquence système** (DSS) permet de **décrire le comportement du système vu de l'extérieur** (par les acteurs) sans préjuger de comment il sera réalisé. Le système est vu comme une « **boîte noire** » qui sera ouverte (décrite) seulement en conception.

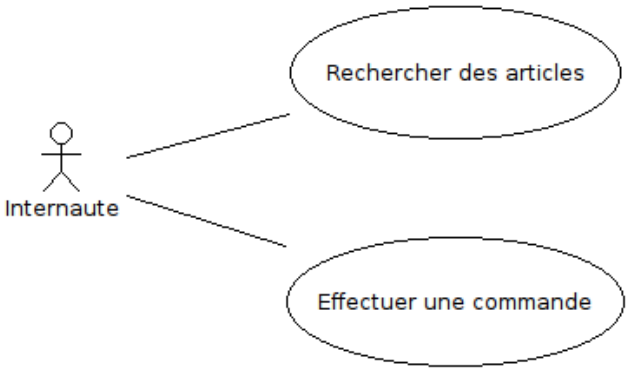


FIGURE 5 – *Diagramme de cas d'utilisation*

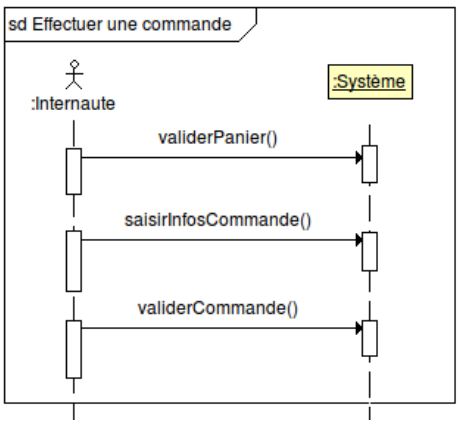
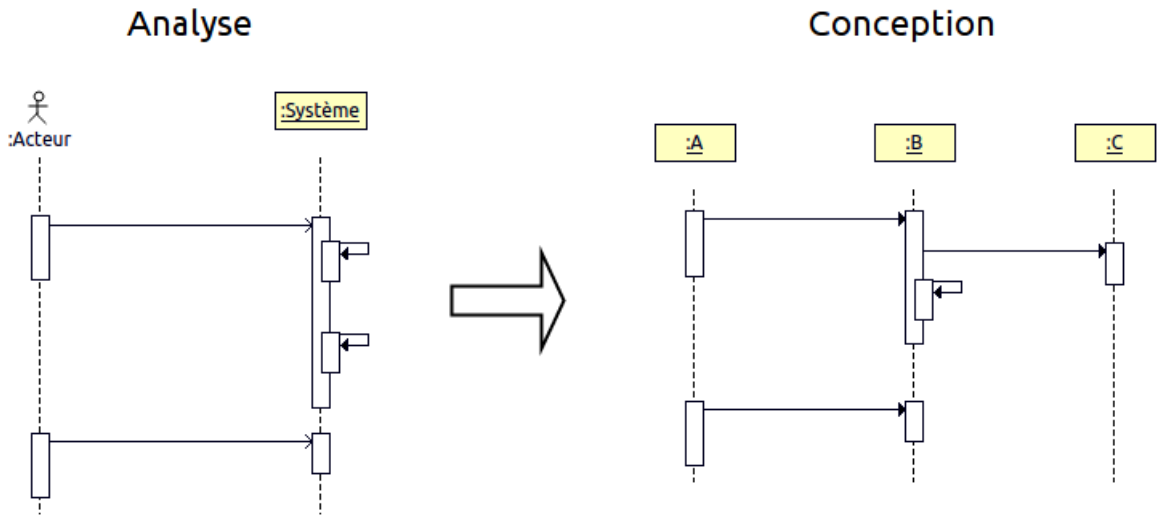


FIGURE 6 – *Diagramme de séquence système*

Passage de l'analyse à la conception : ouvrir la « boîte noire » pour « voir » les objets logiciels interagir



Fragment d'interaction

Un **fragment** (ou cadre) permet **d'identifier une sous-partie d'une interaction** afin que celle-ci soit référencées par d'autres interactions ou de lui spécifier des conditions particulières d'exécution (boucle, optionnel, ...).

- `sd` : fragment du diagramme de séquence en entier
- `alt` : fragment alternatif (Si ... Alors ... Sinon ...)
- `opt` : fragment optionnel
- `par` : fragment parallèle (traitements concurrents)
- `loop` : le fragment s'exécute plusieurs fois (boucle)
- `region` : région critique (un seul thread à la fois)
- `ref` : référence à une interaction dans un autre diagramme

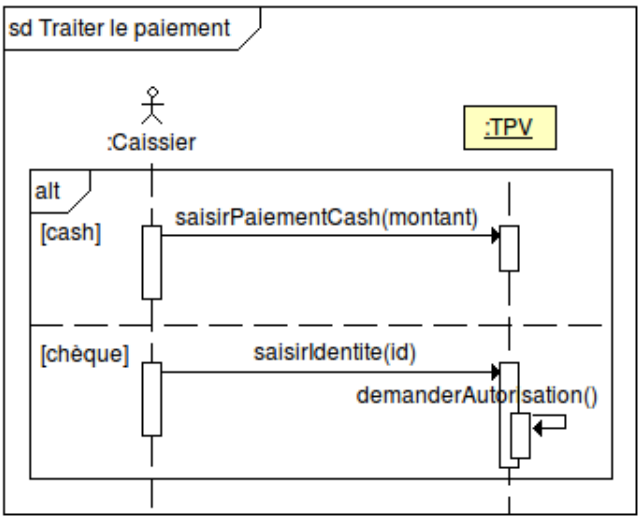


FIGURE 7 – *Exemple d'utilisation du fragment* `alt`

Travail demandé

Vous devez être capable de lire, commenter et compléter un diagramme de séquence à partir d'expressions textuelles et / ou de la définition des objets. Les compétences terminales visées sont : C3.1 et C3.2. Vous devez aussi être capable de produire du code à partir d'un diagramme de séquence.

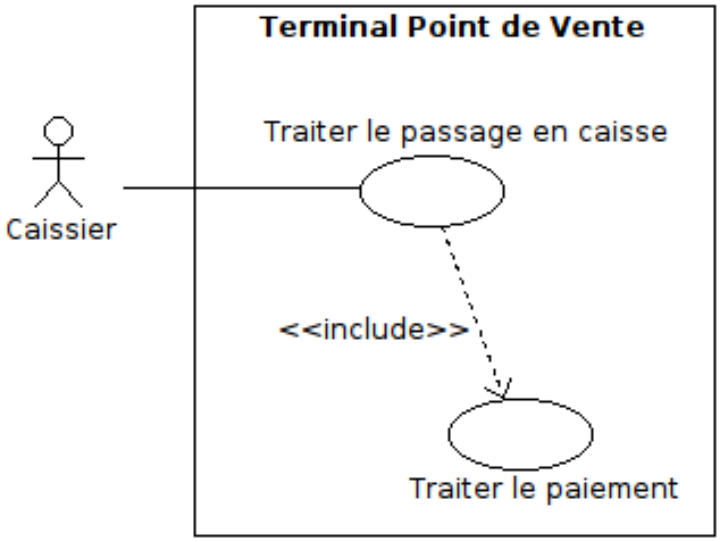
Étude d'un terminal point de vente (TPV)

Cette étude de cas concerne un système simplifié de caisse enregistreuse de supermarché. Le déroulement normal de la caisse est le suivant :

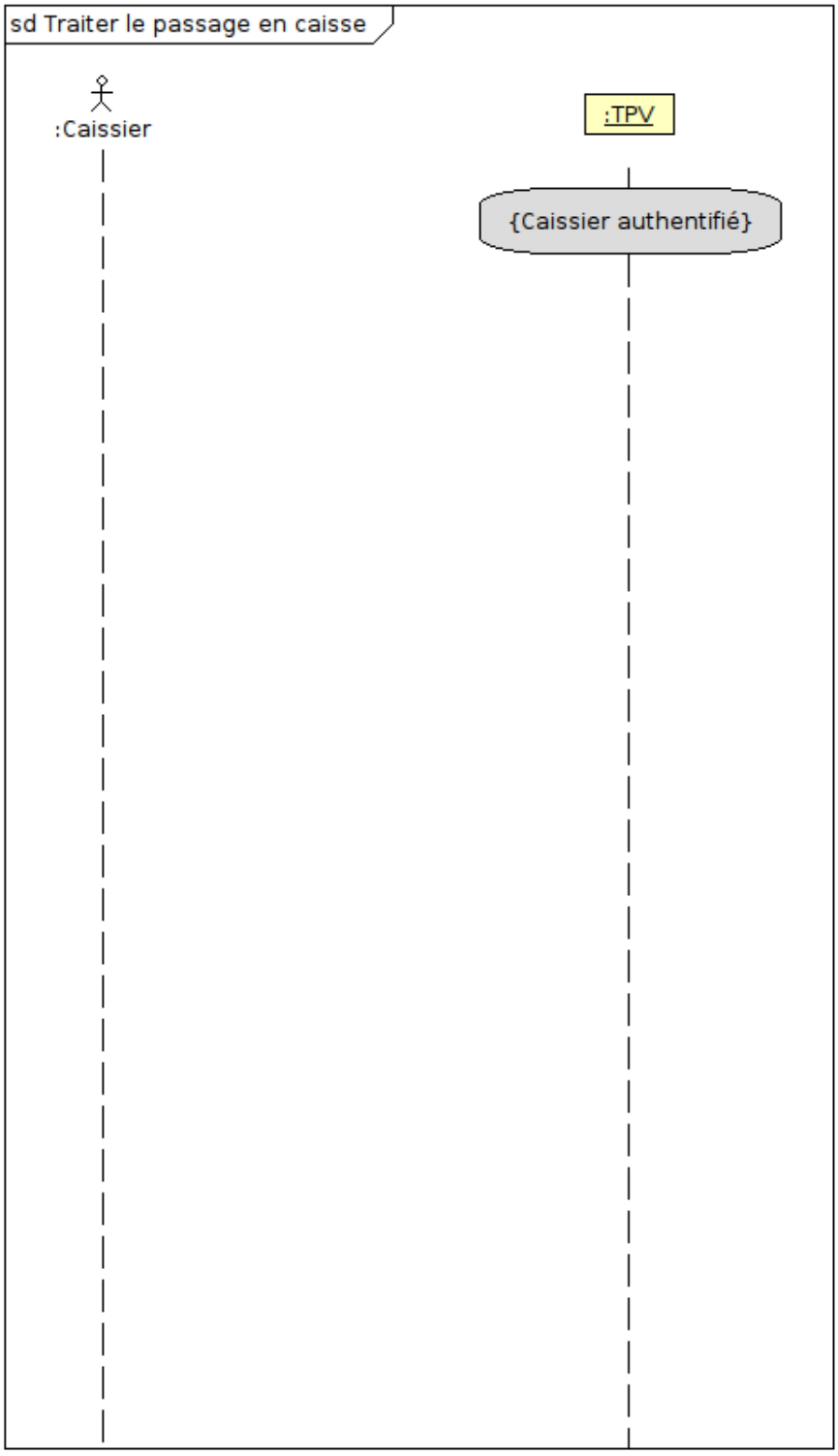
- Un client arrive à la caisse avec des articles à payer
- Le caissier enregistre le numéro d'identification (CPU) de chaque article, ainsi que la quantité si elle est supérieure à 1
- La caisse affiche le prix de chaque article et son libellé
- Lorsque tous les achats sont enregistrés, le caissier signale la fin de la vente
- La caisse affiche le total des achats
- Le client choisit son mode de paiement :
  - numéraire : le caissier encaisse l'argent reçu, la caisse indique la monnaie à rendre au client
  - chèque : le caissier vérifie la solvabilité du client en transmettant une requête à un centre d'autorisation via la caisse
  - carte de crédit : un terminal bancaire fait partie de la caisse. Il transmet une demande d'autorisation à un centre d'autorisation en fonction du type de carte
- Le caissier signale la fin du paiement et la caisse enregistre la vente et imprime un ticket
- Le caissier donne le ticket de caisse au client

L'énoncé est volontairement incomplet et imprécis, comme il est dans les projets réels !

Le diagramme de cas d'utilisation sera le suivant :

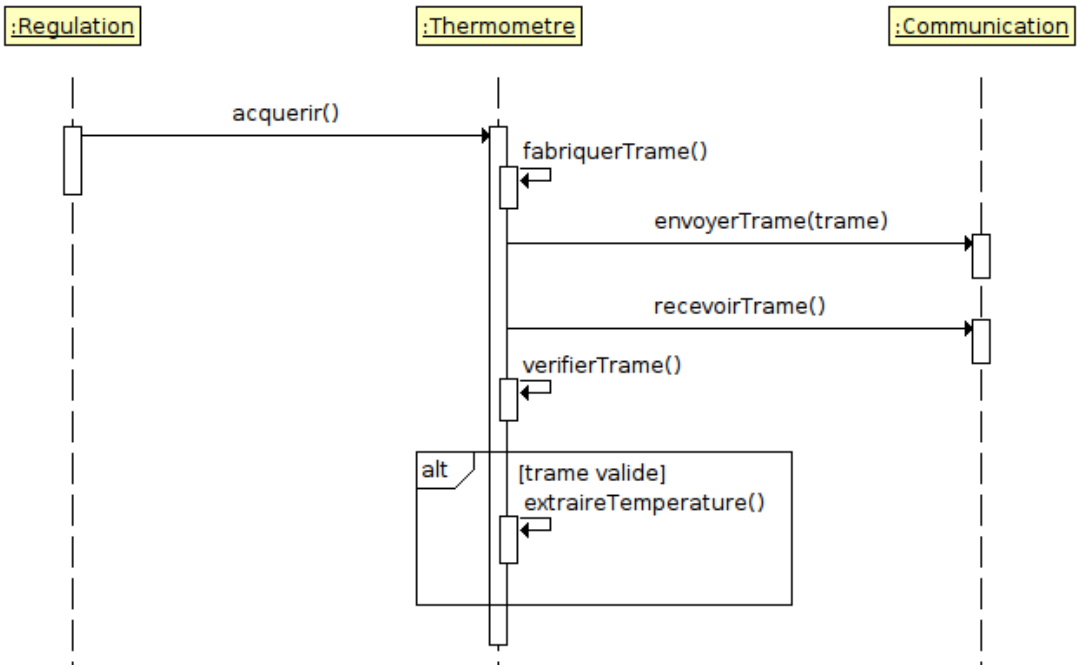


Question 1. Compléter le diagramme de séquence système pour le cas d'utilisation "Traiter le passage en caisse". Le diagramme de séquence de "Traiter le paiement" est fourni figure 7 (page 4).

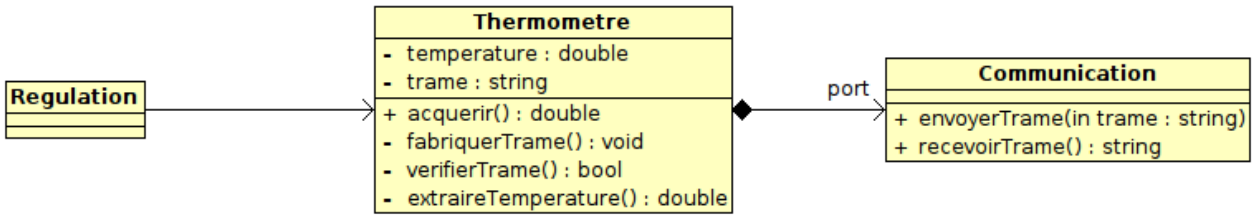


Étude d'une acquisition de mesure

Question 2. Coder la méthode `acquerir()` de la classe `Thermometre` à partir du diagramme de séquence ci-dessous.



Le diagramme de classes partiel est le suivant :



L'exploitation du diagramme de classes est indispensable.

