

UE Modélisation
Mardi 23 mars 2021

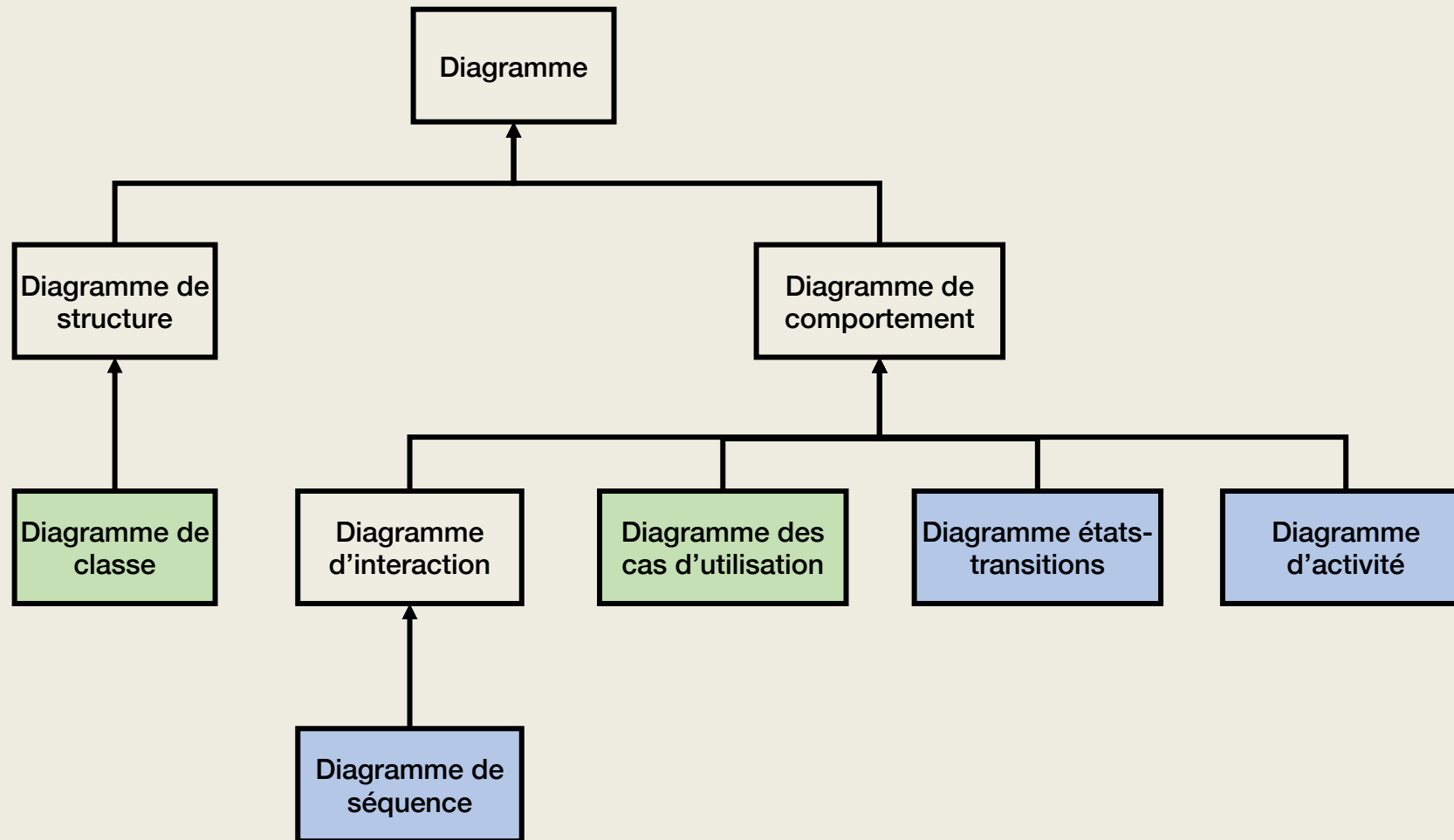
Diagramme de séquence

Damien MONDOU, Enseignant chercheur, La Rochelle Université

damien.mondou@univ-lr.fr

PARTIE I : INTRODUCTION

Les diagrammes UML (vus cette année)



Génie logiciel 1

Modélisation

Diagramme d'interaction

Objectif : Représenter les interactions entre groupe d'objets qui collaborent en échangeant des **messages**

2 types d'interaction entre objets :

- Interaction visualisée selon le point de vue de l'**espace** (objets et liens entre eux)

➔ Diagramme de collaboration

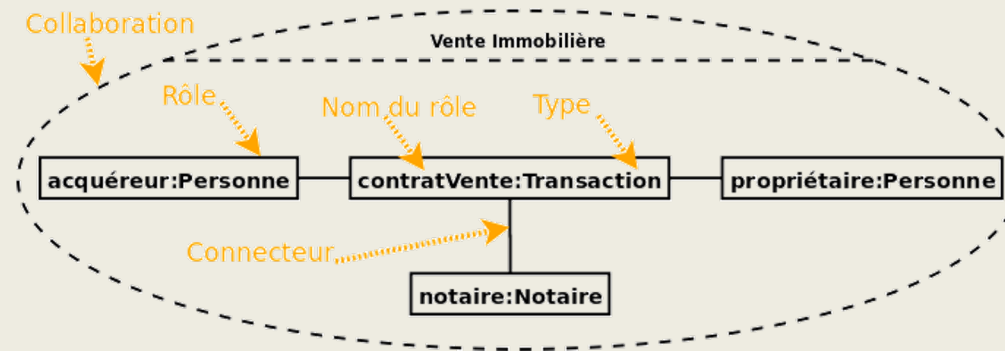


Diagramme de collaboration d'une transaction immobilière

(source : <https://laurent-audibert.developpez.com/Cours-UML/?page=diagrammes-interaction>)

Diagramme d'interaction

Objectif : Représenter les interactions entre groupe d'objets qui collaborent en échangeant des **messages**

2 types d'interaction entre objets :

- Interaction visualisée selon le point de vue de l'**espace** (objets et liens entre eux)

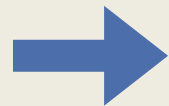


Diagramme de collaboration

- Interaction visualisée selon le point de vue du **temps** (cycle de vie des objets)

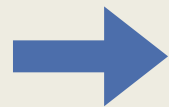


Diagramme de séquence

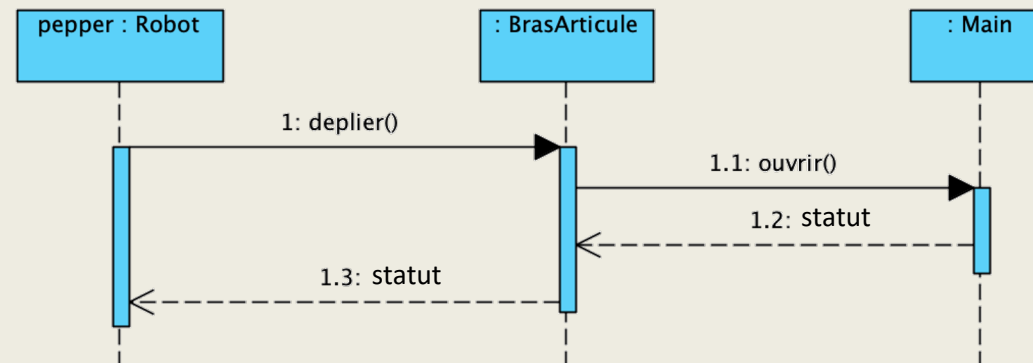


Diagramme de séquence d'une commande robotique

PARTIE II : ENTRONS DANS LE VIF DU SUJET...

Diagramme de séquence

Objectif : Description de la dynamique du système et des interactions entre un groupe d'objets en montrant, de façon séquentielle, les envois de message entre les objets (éventuellement les données échangées)

Dans un diagramme de séquence :


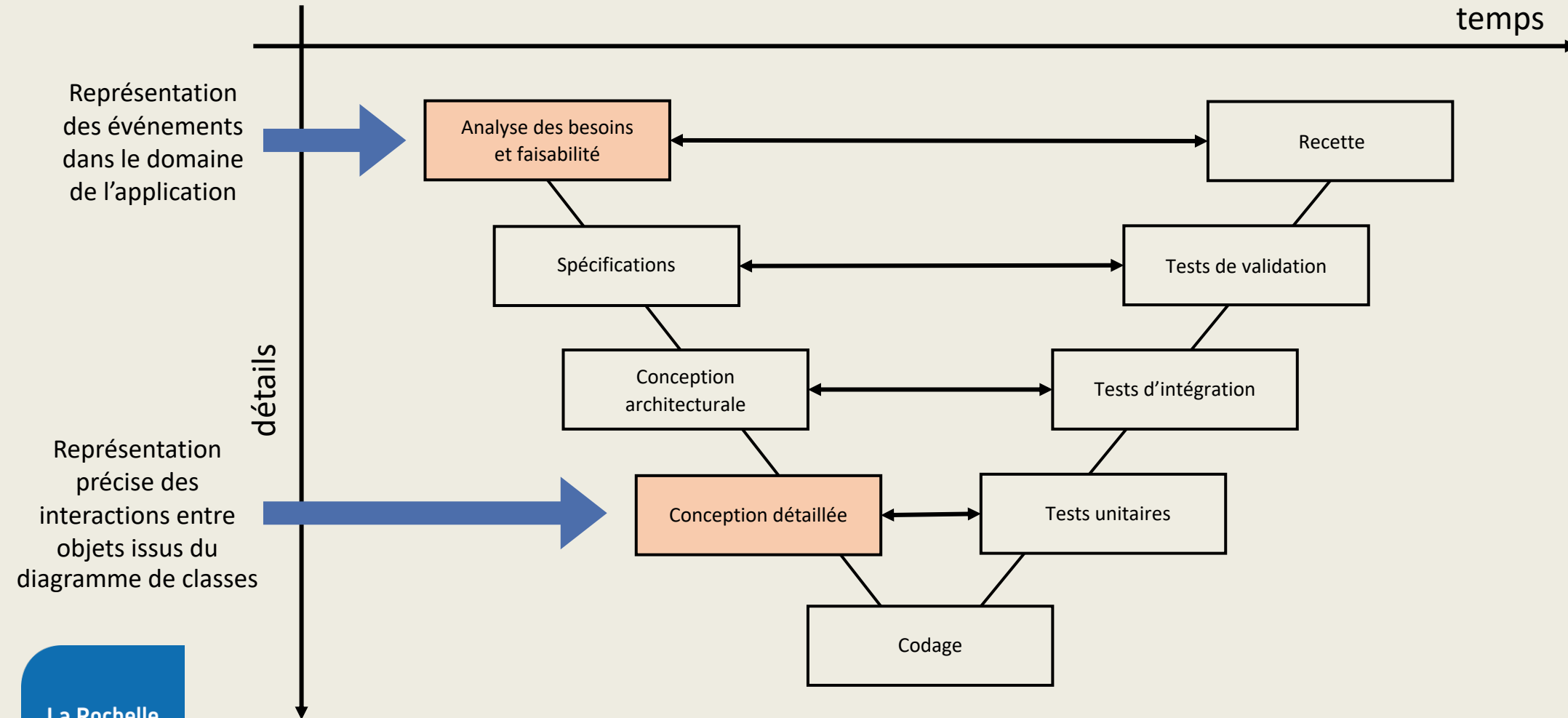
- ➔ Représentations des objets participants à l'interaction

- ➔ Lignes de temps : chronologie de l'échange (axe verticale)
- ➔ Messages activant des opérations chez l'objet receveur (communication entre objets)
- ➔ Eventuellement des fragments (structures conditionnelles, boucles...)

Diagramme de séquence et cycle en V

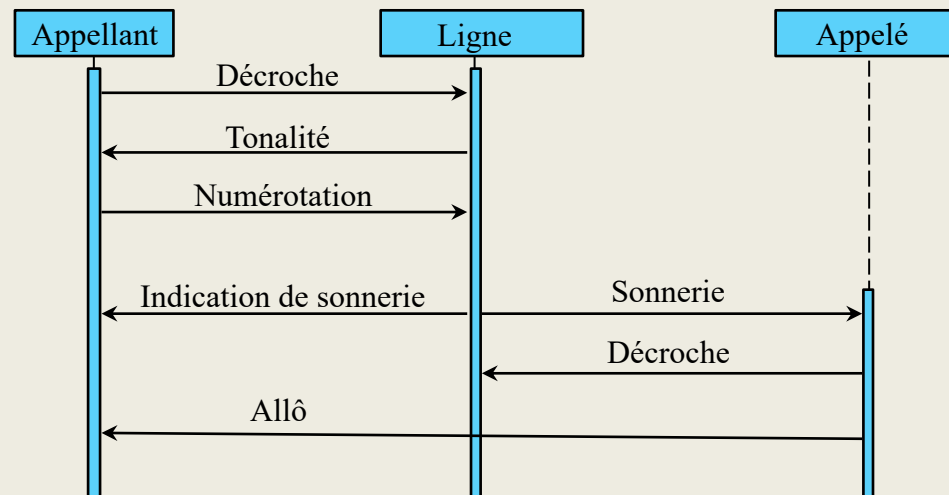


1^{ère} utilisation : analyse des besoins

Objectif : Description et documentation des cas d'utilisation :

- ➔ Langage naturel
- ➔ Pas de détails de synchronisation
- ➔ Ne correspond pas aux envois de message en programmation

Exemple : Appel téléphonique

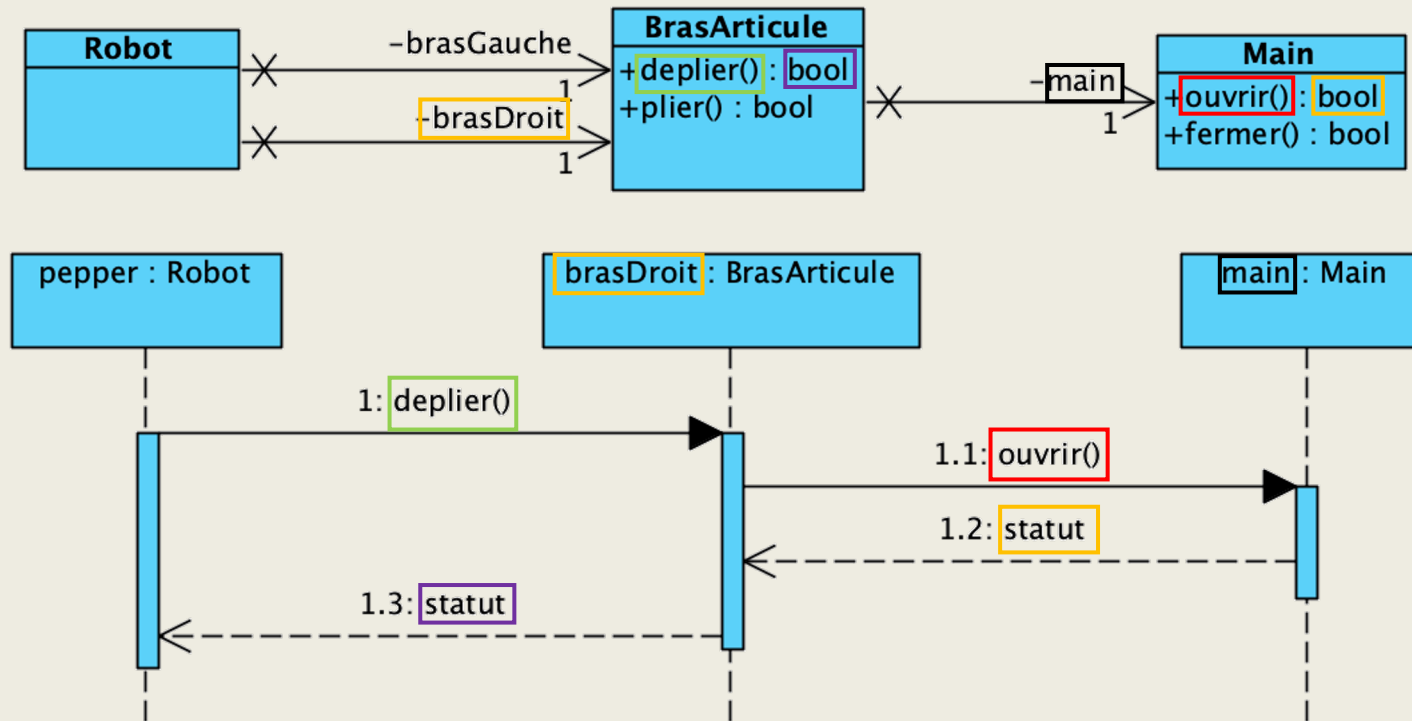


2^{ème} utilisation : conception détaillée

Objectif : Représentation précise des interactions entre objets issus du diagramme de classes :

➔ Concept de messages (appel de procédure, interruption, etc.)

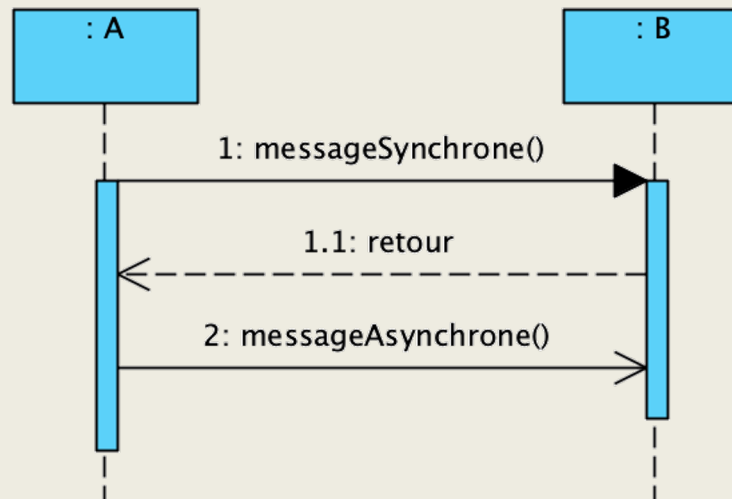
Exemple : Commande robotique



Les messages

Objectif : Représentation chronologique d'une communication particulière entre objets. 3 types d'envois de message :

- ➔ Message synchrone (bloque l'expéditeur jusqu'à la réponse du destinataire)
- ➔ Message asynchrone (non bloquant)
- ➔ Message de retour

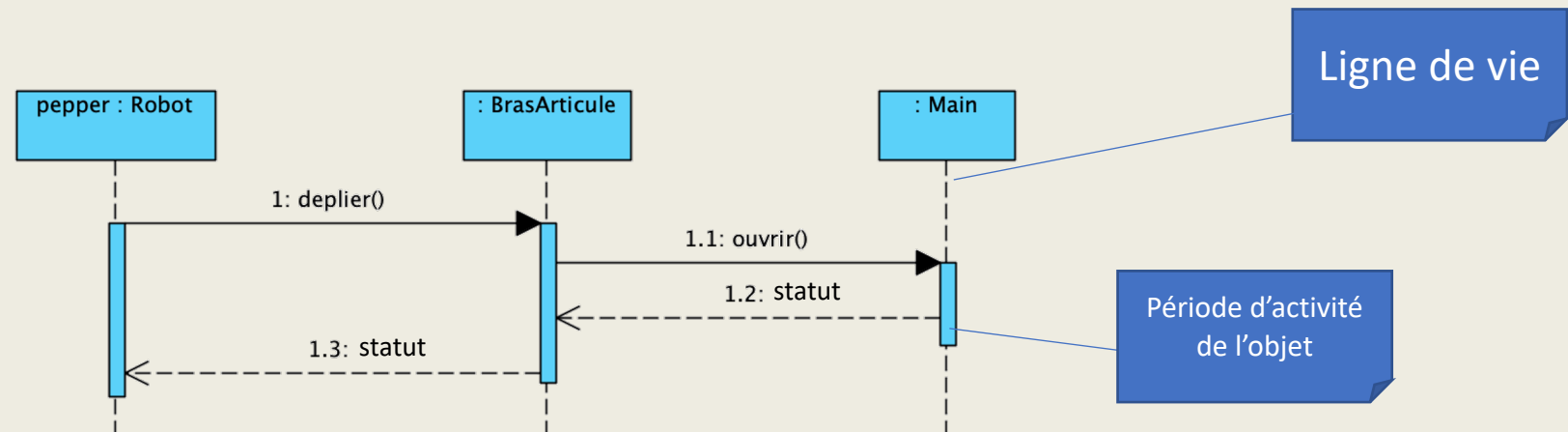


Les périodes d'activité

Définition : Une période d'activité représente une durée pendant laquelle un objet effectue une action, soit directement, soit par l'intermédiaire d'un autre objet qui lui sert de sous-traitant.

➔ Bandes rectangulaires placées sur les lignes de vie

➔ Le début et la fin d'une bande correspondent respectivement au début et à la fin d'une période d'activité.



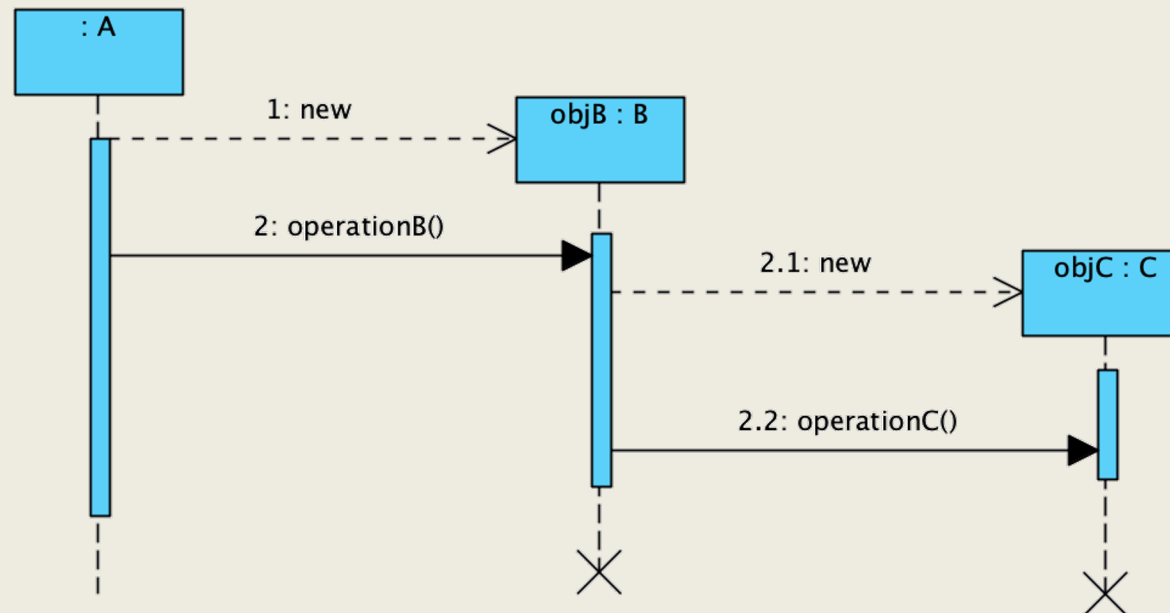
Création / destruction d'objets

Exemple : Comment représenter la suite d'échanges réalisée après l'appel de `operationA()` ?

```
public class A {  
    public void operationA(){  
        System.out.println("Operation A");  
        B objB = new B();  
        objB.operationB();  
    }  
}
```

```
public class B {  
    public void operationB(){  
        System.out.println("Operation B");  
        C objC = new C();  
        objC.operationC();  
    }  
}
```

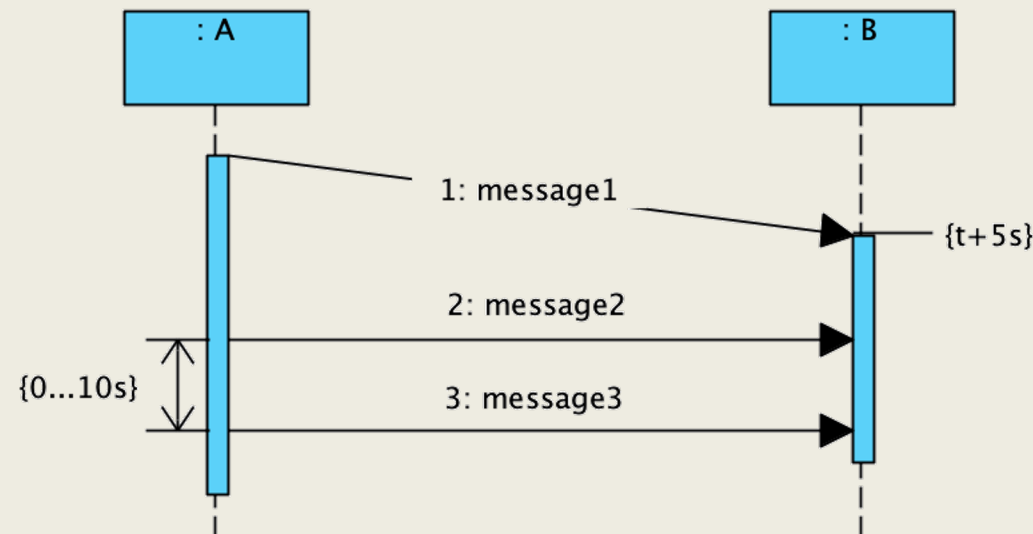
```
public class C {  
    public void operationC() {  
        System.out.println("Operation C");  
    }  
}
```



Contraintes temporelles

Exemple : Comment représenter la suite d'échanges suivante :

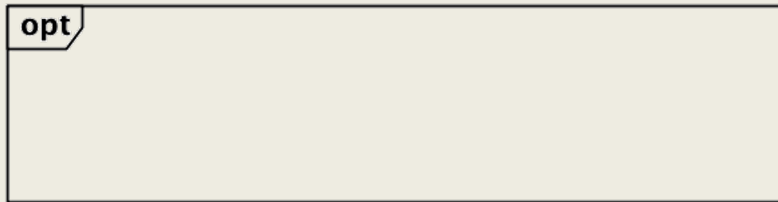
- ➔ Le message1 est reçu 5 secondes après son émission
- ➔ Le message3 est envoyé au plus tard 10 secondes après message2



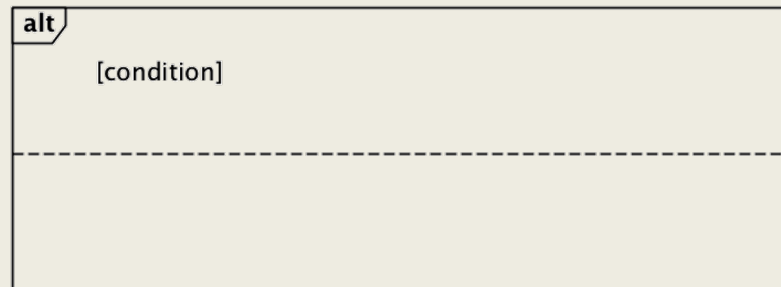
Les fragments combinés

Définition : Les fragments combinés offrent le support de nombreuses constructions comme les alternatives, les boucles, le parallélisme, etc.

➔ Représentés par un rectangle dont le coin supérieur gauche contient le type de fragment



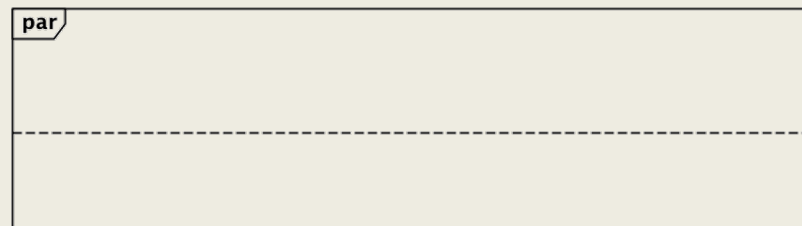
option



alternative



boucle



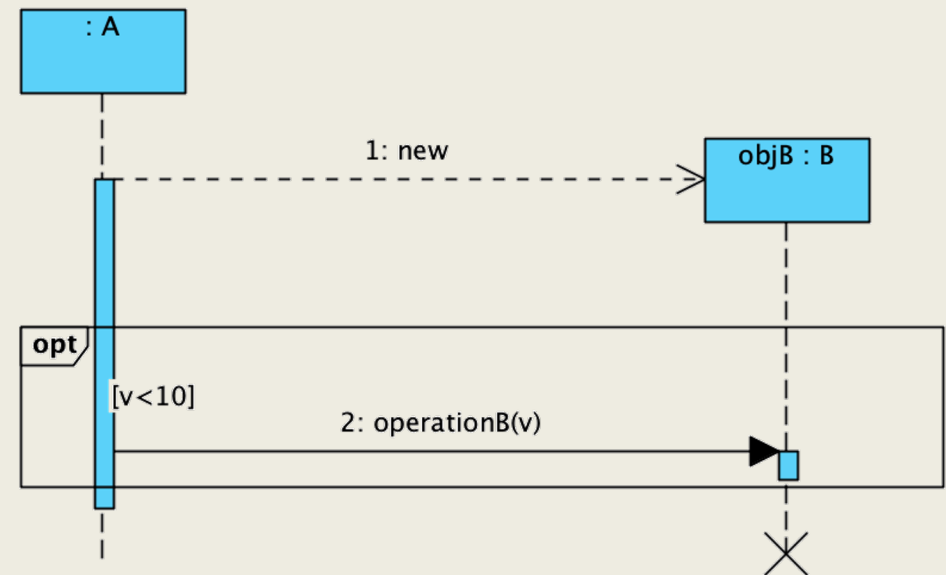
parallélisme

Les fragments combinés : l'option

Définition : L'option s'obtient en utilisant le fragment **opt** suivi d'une condition sous la forme d'une expression logique. Si la condition est vérifiée, le contenu du fragment est exécuté.

Exemple : Représenter la suite d'échanges réalisée après l'appel de `operationA(v)`

```
public class A
{
    public void operationA(int v)
    {
        System.out.println("Operation A");
        B objB = new B();
        if (v < 10)
            objB.operationB(v);
    }
}
```

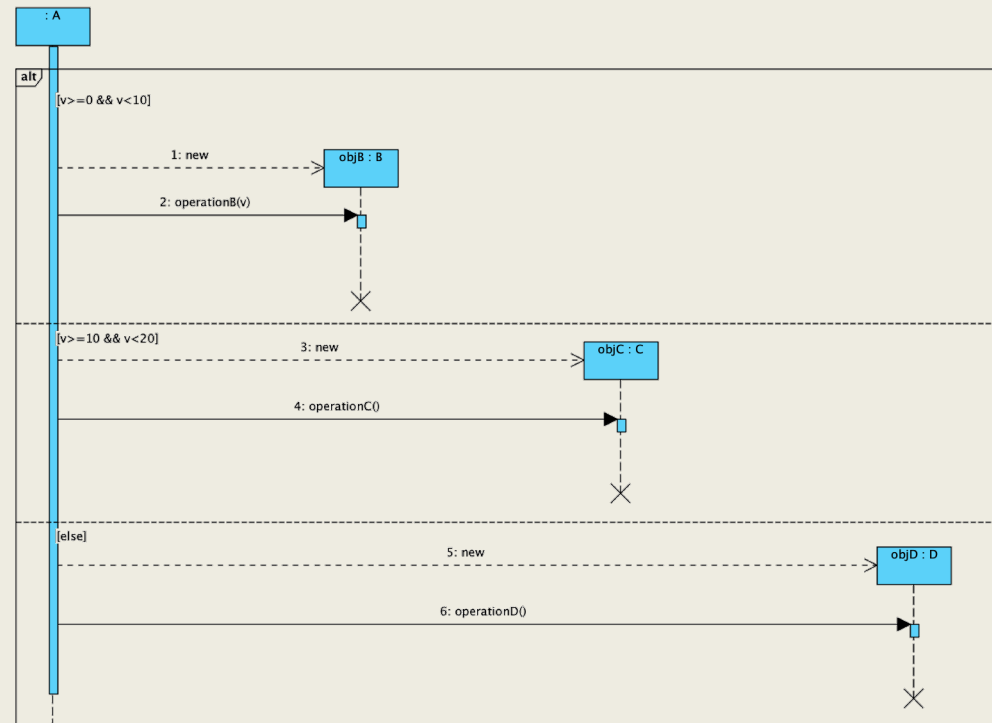


Les fragments combinés : l'alternative

Définition : Le fragment combiné de l'alternative comprend plusieurs conditions de test (au moins une) puis le mot-clé `else`. Le fragment est scindé en plusieurs parties dont le contenu n'est exécuté que si la condition associée est remplie. Le contenu de la dernière partie est associé au mot-clé `else`. Il est exécuté uniquement si aucune des conditions précédentes n'est vérifiée.

Exemple : Représenter la suite d'échanges réalisée après l'appel de `operationA(v)`

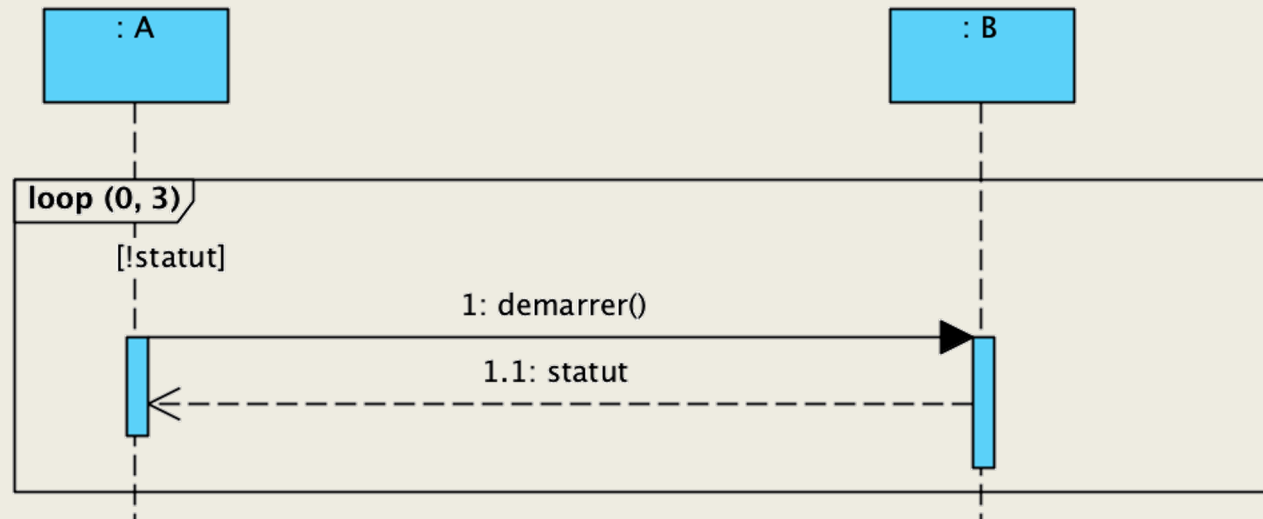
```
public class A {  
    public void operationA(int v){  
        if(v >= 0 && v<10){  
            B objB = new B();  
            objB.operationB(v);  
        }  
        else if(v >=10 && v <20){  
            C objC = new C();  
            objC.operationC();  
        }  
        else{  
            D objD = new D();  
            objD.operationD();  
        }  
    }  
}
```



Les fragments combinés : la boucle

Définition : La boucle est réalisée par l'opérateur **loop** suivi des paramètres **min**, **max** et associé à une condition. Le contenu du fragment est exécuté **min** fois, puis tant que la condition de test est vérifiée et tant que le nombre maximal d'exécutions de la boucle ne dépasse pas **max**. Tous les paramètres et la condition sont optionnels.

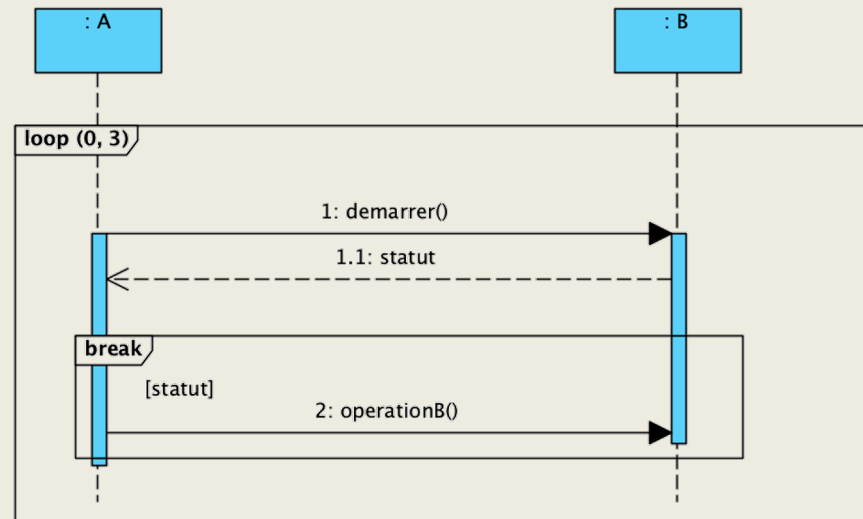
Exemple : un objet A peut appeler une méthode `demarrer()` d'un objet B tant que le statut retourné n'a pas la valeur `true` et au maximum 4 fois.



Les fragments combinés : l'opérateur break

Définition : Si la condition du fragment est vérifiée, alors son contenu est exécuté puis, et à la différence de l'option, l'exécution du fragment qui le contient se termine immédiatement. S'il n'est pas contenu dans un fragment, alors c'est l'exécution du diagramme de séquence qui s'achève.

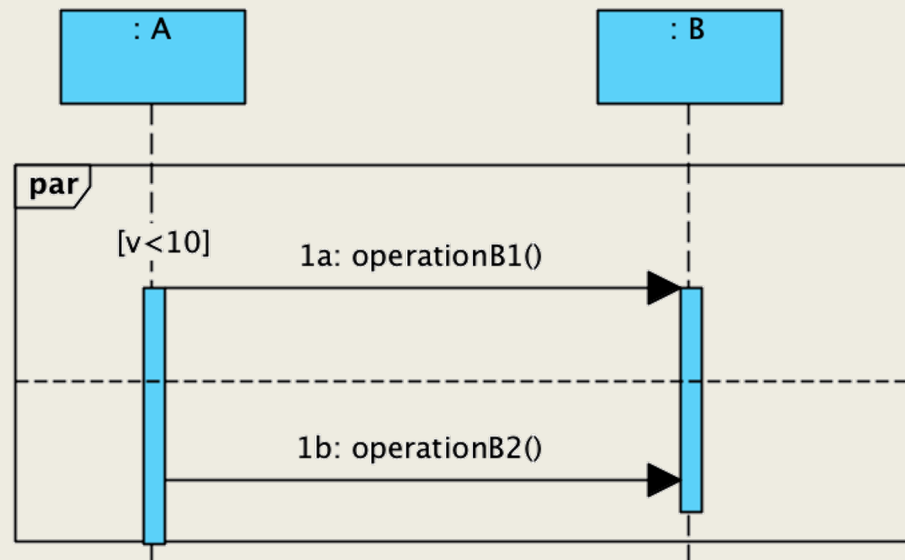
Exemple : un objet A peut appeler une méthode `demarrer()` d'un objet B au maximum 4 fois. Si le statut retournée possède la valeur `true`, alors la méthode `operationB()` est exécutée et la boucle s'arrête.



Les fragments combinés : la parallélisme

Définition : Le parallélisme est un fragment combiné scindé en plusieurs parties dont chaque contenu est exécuté simultanément. L'exécution de chaque partie peut être soumise à une condition.

Exemple : un objet `A` appelle une méthode `operationB1()` d'un objet `B` si l'entier `v` est inférieur à 10 tout en appelant `operationB2()`.



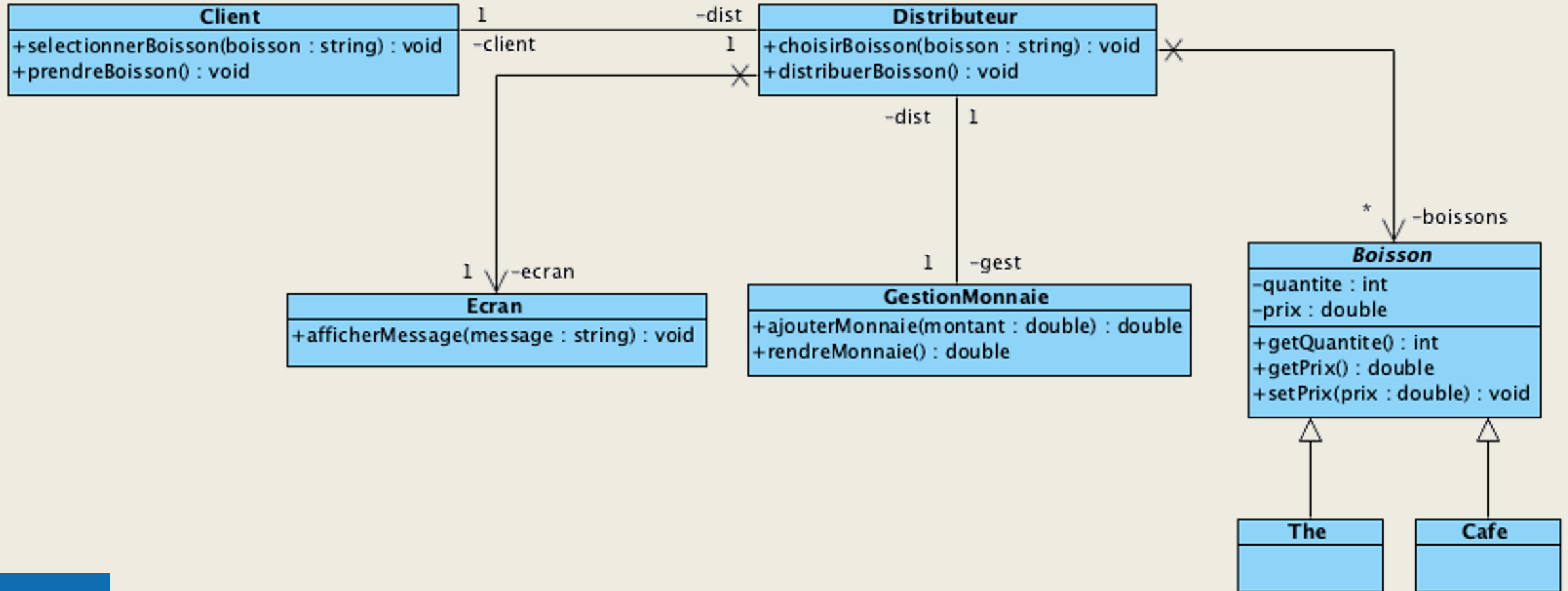
PARTIE III : DES EXOS ...

La machine à café

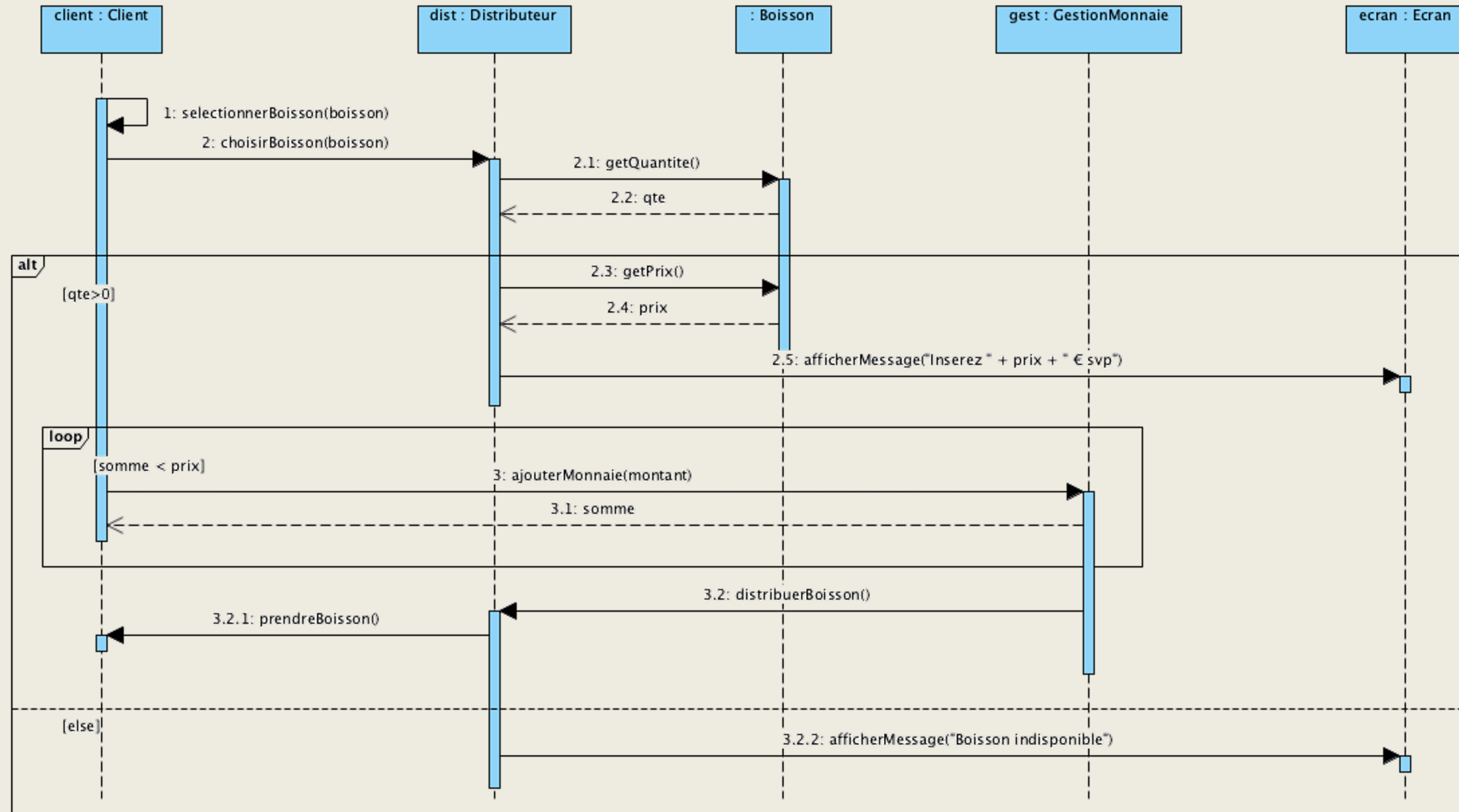
Cahier des charges : Nous souhaitons modéliser le système d'un distributeur de boissons, délivrant du café ou du thé. La spécification est la suivante :

- L'utilisateur choisit sa boisson sur le distributeur. Pour la boisson sélectionnée, le distributeur vérifie la quantité en stock.
 - Si la boisson est disponible (stock >0), alors :
 - Le distributeur récupère le prix de la boisson;
 - l'écran du distributeur affiche le prix de la boisson;
 - le client insère son argent;
 - Le distributeur rend la monnaie et fait couler la boisson;
 - Le distributeur donne la boisson à l'utilisateur;
 - Le distributeur met à jour la quantité de boisson disponible.
 - Sinon, un message d'erreur est affiché sur l'écran.

La machine à café : diagramme de classe



La machine à café : diagramme de séquence



Source

- UML 2.5, Initiation, exemples et exercices corrigés – 5^{ème} édition, Laurent Debrauwer, Fien Van Der Heyde, Edition eni, mars 2020
- Cours d'Armelle Prigent