Polytech Paris-Sud



Formation initiale 3e année

Spécialité Informatique

Année 2016-2017

# UML

Cours 5

Diagrammes de séquence

Delphine Longuet delphine.longuet@lri.fr

http://www.lri.fr/~longuet/Enseignements/16-17/Et3-UML

# Diagrammes de séquence (conception)

Objectif : Représenter les communications avec et au sein du logiciel

* Représentation temporelle des interactions entre les objets
* Chronologie des messages échangés entre les objets et avec les acteurs

En conception : Décrire la réalisation des cas d'utilisation sur le système décrit par le diagramme de classes

* Point de vue interne sur le fonctionnement du système
* Description au niveau de l'instance (état du système à un instant)
* Description de scénarios particuliers
* Représentation des échanges de messages
* Entre les acteurs et le système, entre les objets du système ● De façon chronologique

# Diagrammes de séquence (conception)

Éléments du diagramme de séquence

* Acteurs
* Objets (instances)
* Messages (cas d'utilisation, appels d’opération)

Principes de base : Représentation graphique de la chronologie des échanges de messages avec le système ou au sein du système

* « Vie » de chaque entité représentée verticalement
* Échanges de messages représentés horizontalement

Diagramme de cas d'utilisation Diagramme de classes du système

Objectif : Description de la réalisation d'un cas d'utilisation sur le système décrit par le diagramme de classes

Acteur

Utiliser

Système

**Classe1**

**Classe3**

Problème : Communication entre les acteurs et le système vu comme un ensemble d'objets

Diagramme de cas d'utilisation Diagramme de classes du système

interface

Classe1

Classe3

Classe2

Acteur

Acteur

Utiliser

Système

**Classe1**

**Classe3**

Communication entre acteurs et système via une interface (texte, web, physique...)

Diagramme de cas d'utilisation Diagramme de classes du système

*classe de contrôle*

Acteur

Utiliser

Système

**Classe1**

**Classe3**

~~r~~

Acteu

**Interface**

Classe1

Classe3

Classe2

**Contrôle**

Écran

Clavier

*classes*

*d'interface*

Communication entre acteurs et système via une interface (texte, web, physique...)

Solution : Création de classes de contrôle et de classes d'interface qui :

* gèrent les interactions avec les acteurs
* encapsulent le résultat des opérations

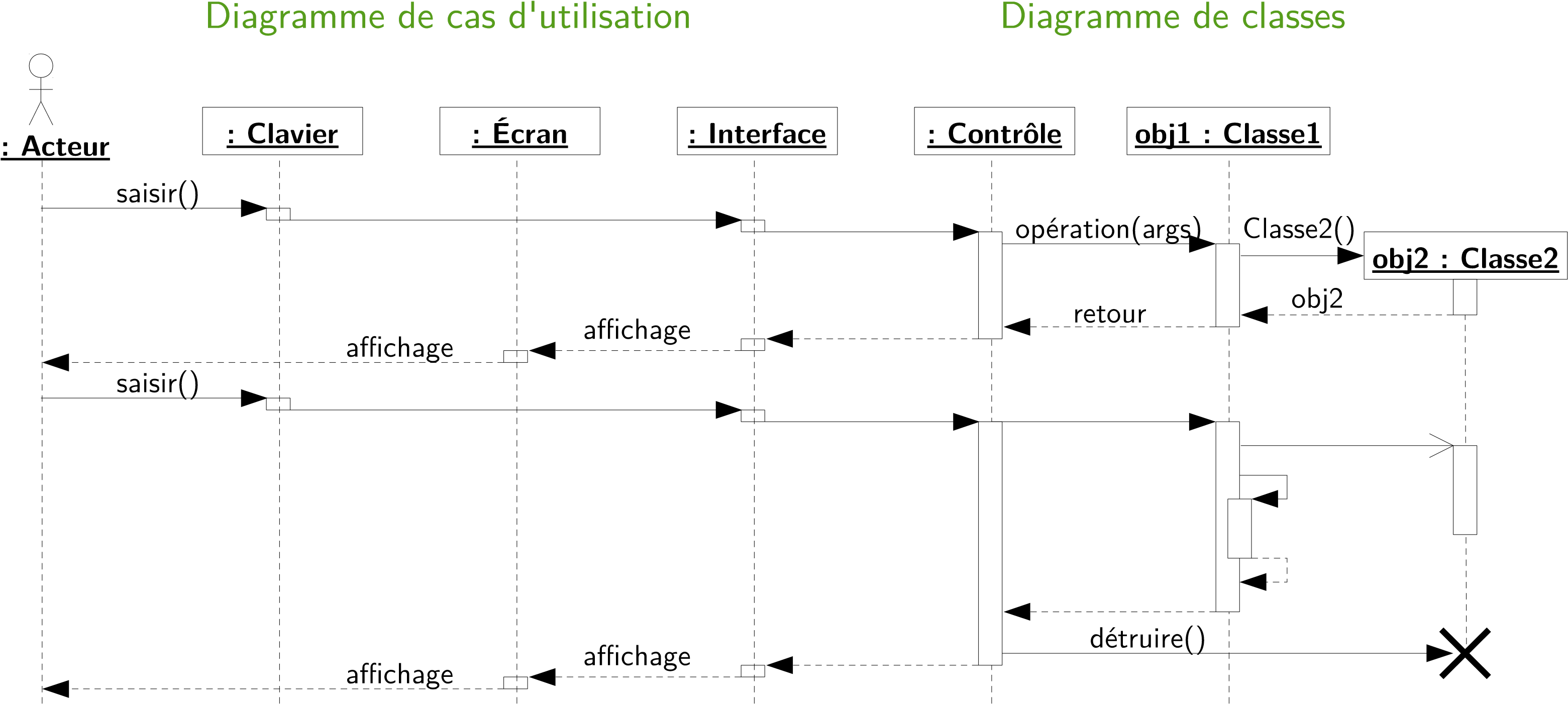


Diagramme de séquence du cas d'utilisation Utiliser

Acteur

Utiliser

Système

**Classe1**

**Classe3**

# Éléments de base

Jean : Acteur

objet1 : Classe1

objet2 : Classe2

opération1()

opération2(args)

retour2

retour1

*instance d'acteur*

*instance d'une classe*

*du diagramme de classes*

*valeur renvoyée*

*exécution*

*appel d'une*

*opération*

*ligne de vie*

# Types de messages

Message synchrone : Émetteur bloqué en attente du retour

*exécution flèche pleine*

vérifierSolde(compte)

soldeOK

**:**

**GAB**

**:**

**Banque**

*bloquée*

Message asynchrone : Émetteur non bloqué, continue son exécution

allumer()

*processus évoluant*

*en parallèle*

*flèche ouverte*

*pas nécessairement*

*de retour*

**:**

**Ascenseur**

**:**

**Voyant**

# Création et destruction d'objet

**:**

**PageWeb**

**nouvSession : Session**

nouvSession

Session()

destroy()

*appel du constructeur*

*de la classe*

*retour obligatoire*

*de l'instance créée*

*destruction de l'objet*

*(*

*pas nécessairement*

*à la suite d'un message)*

*exécution du*

*constructeur*

# Message réflexif

**:**

**Catalogue**

**livres : Ensemble**

chercher(auteur)

listeLivresAuteurTriée

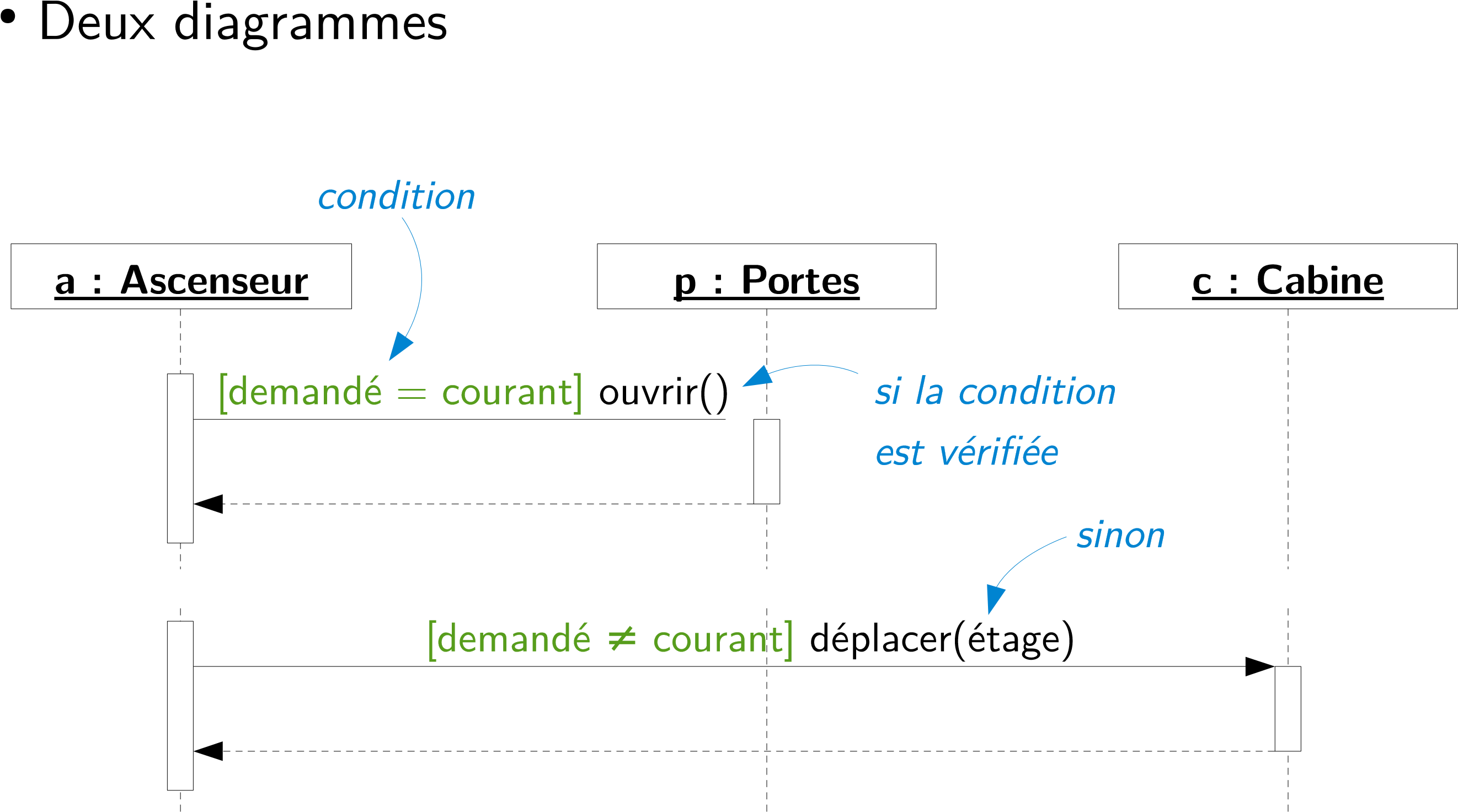
trier()

*appel d'une*

*opération interne*

# Alternative

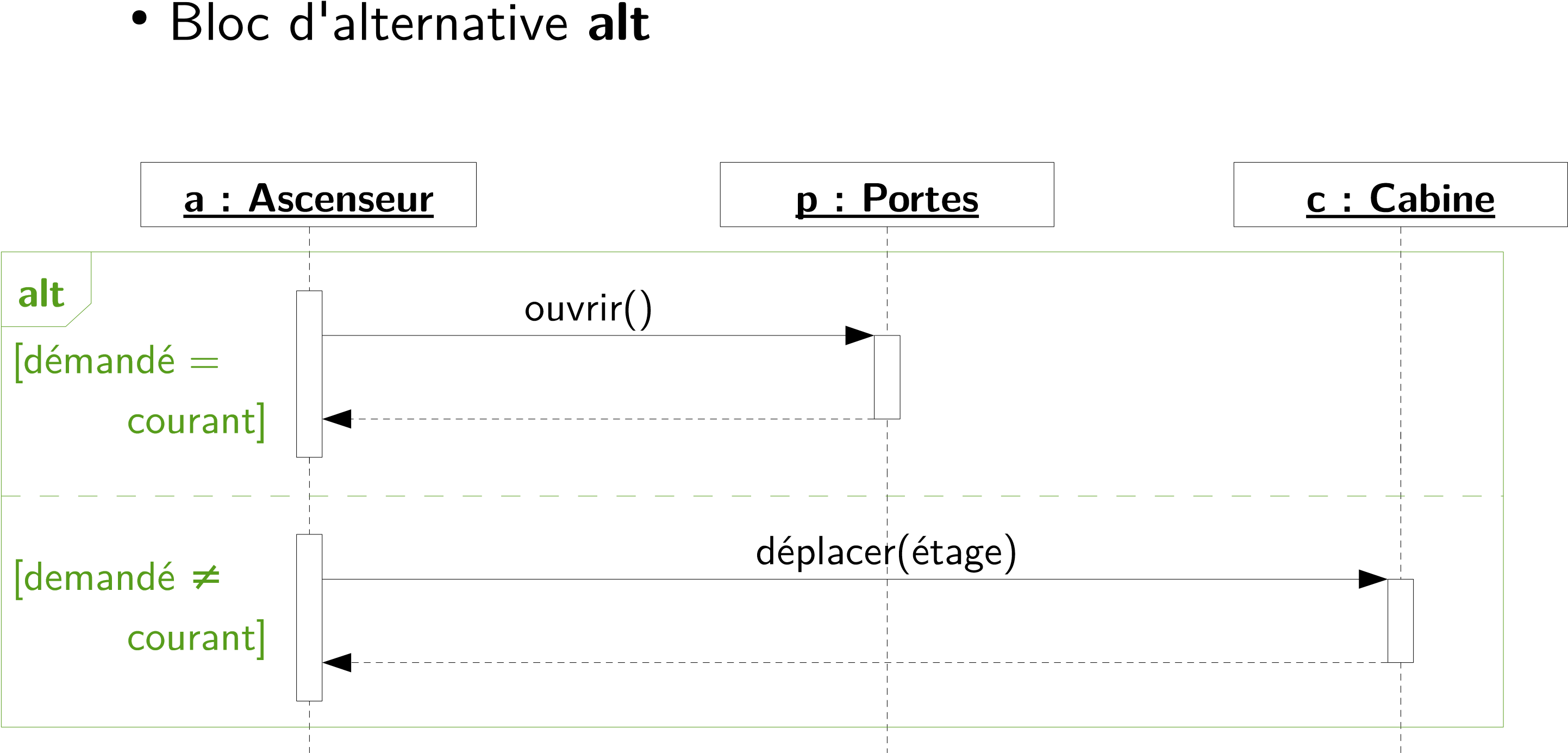
Principe : Condition à l'envoi d'un message Notation :



# Alternative

Principe : Condition à l'envoi d'un message Notation :

● Deux diagrammes



# Boucle

Principe : Répéter un enchaînement de messages Notation : ● Note

*opération répétée pour tous les objets mentionnés*

**cat : Catalogue**

**livre : Livre**

premierAuteur()

Pour chaque

livre du catalogue

auteur

# Boucle

Principe : Répéter un enchaînement de messages Notation :

* Note
* Bloc de boucle **loop**

**cat : Catalogue**

**livre : Livre**

premierAuteur()

**loop(livre in cat)**

auteur

*objets sur lesquels répéter la boucle*

# Référence à un autre diagramme

**b : Banque**

**c : Compte**

**cl : Client**

**vérification du solde**

**ref**

**prélèvement agios notifié au client**

**ref**

*référence à un*

*diagramme décrit ailleurs*

# Exemple - Analyse

|  |  |  |
| --- | --- | --- |
| **Compte** | | |
| numéro : int devise : Devise solde : float | | |
|  | | |
| 1  0..\* |  | 1  0..\* |
| **Virement** | | |
| montant : float date : Date | | |
|  | | |

|  |
| --- |
| **Client** |
| nom : string naissance : Date |

1 1..\*

Employé

SGB

effectuerVirementP(id,noC1,noC2,mnt)

true

Effectuer un

virement personnel

Système de gestion bancaire

Employé

Cas d'utilisation *Effectuer un virement personnel* (niveau analyse)

**Client**

Effectuer un

virement personnel

Système de gestion bancaire

Employé

nom : string 1

naissance : Date **Compte**

1..\* numéro : int devise : Devise solde : float

1 1

0..\* 0..\*

**Virement** montant : float date : Date

**:**

**Employé**

**SGB**

**:**

effectuerVirementP(id,noC1,noC2,mnt)

true

chercher client

**ref**

créer virement

**ref**

**c : Client**

chercher comptes

**ref**

**v : Virement**

effectuer virement

**ref**

Réalisation du

cas d'utilisation

*Effectuer un*

*virement personnel*

(

niveau conception

)

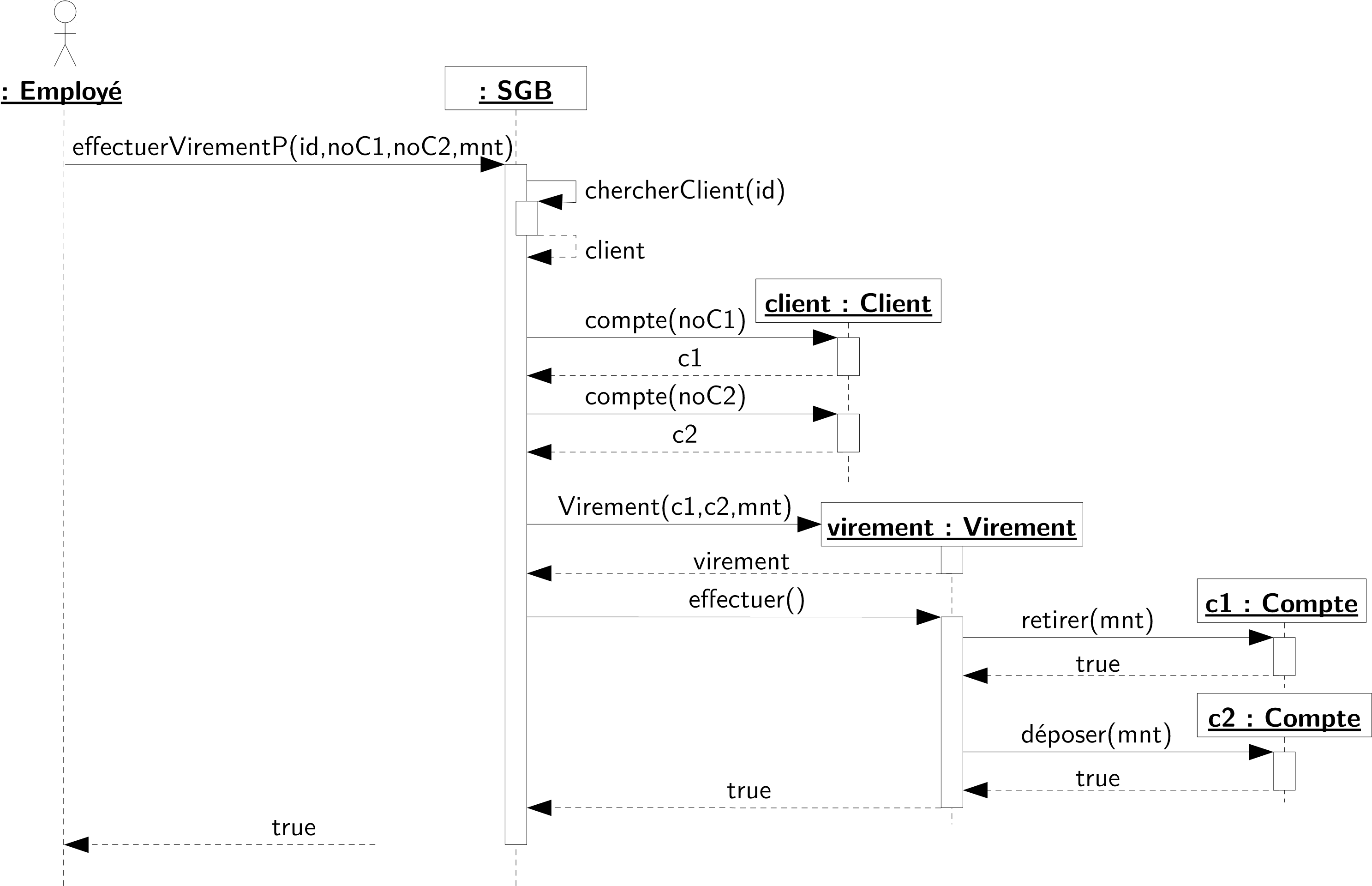


Diagramme de classes complété avec les classes techniques et les opérations nécessaires

**Client**

nom : string

naissance : Date

**Compte**

numéro : int

devise : Devise

solde : float

déposer(mnt : float) : bool

retirer(mnt : float) : bool

**Virement**

montant : float

date : Date

effectuer() : bool

1..\*

1

0..\*

0..\*

compte(no : int) : Compte

1

1

**SGB**

effectuerVirementP() : bool

chercherClient(id : int) : Client

\*

1

1

\*

# Quelques règles

Messages entre acteurs et interface

* « Fausses » opérations liées au cas d'utilisation (même nom) ● Arguments (saisis) et valeurs de retour (affichées) simples : texte, nombre

Messages au sein du système

* Opérations du diagramme de classes
* Si message de **objA : ClasseA** vers **objB : ClasseB**, alors opération du message dans ClasseB