

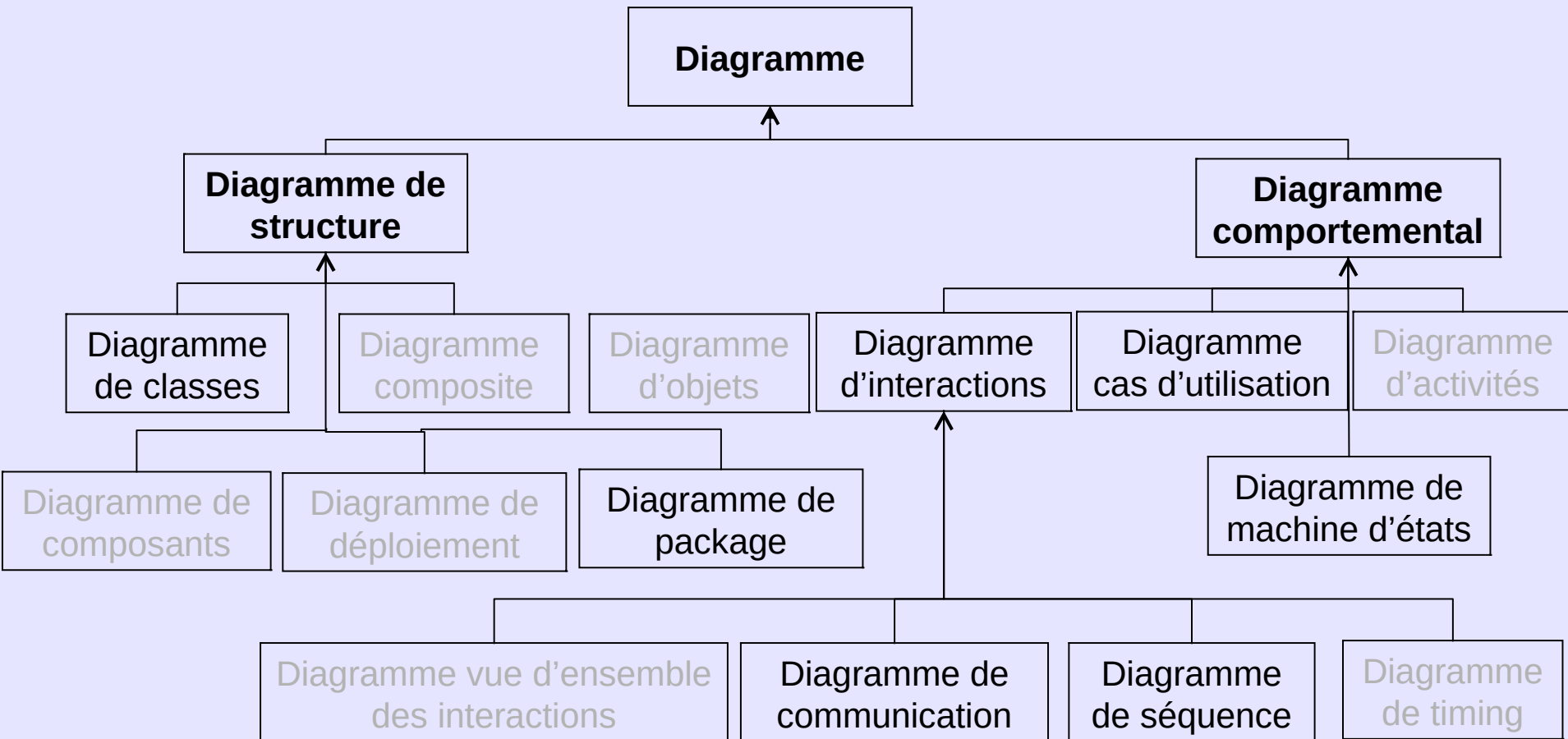
UML

- Cas d'utilisation et scénarios
- Diagramme de séquence
- Diagramme de communication
- Diagramme de classe
- Diagramme d'états-transitions

UML

- UML (Unified Modeling Language) est une méthode de modélisation orientée objet standardisée, adoptée par l'OMG en 1997.
20 ans de développement objet : méthodes, techniques et langages → Besoin de standardisation (Object Management Group)
 - 1996 : UML 1.0 (proposition à l'OMG)
 - 1997 : UML 1.1 (standardisée par l'OMG)
 - ...
 - 2005 : UML 2.0
- UML utilise 14 types de diagrammes pour la modélisation d'un système de l'analyse des besoins jusqu'à l'implémentation.

Les diagrammes d'UML



La démarche

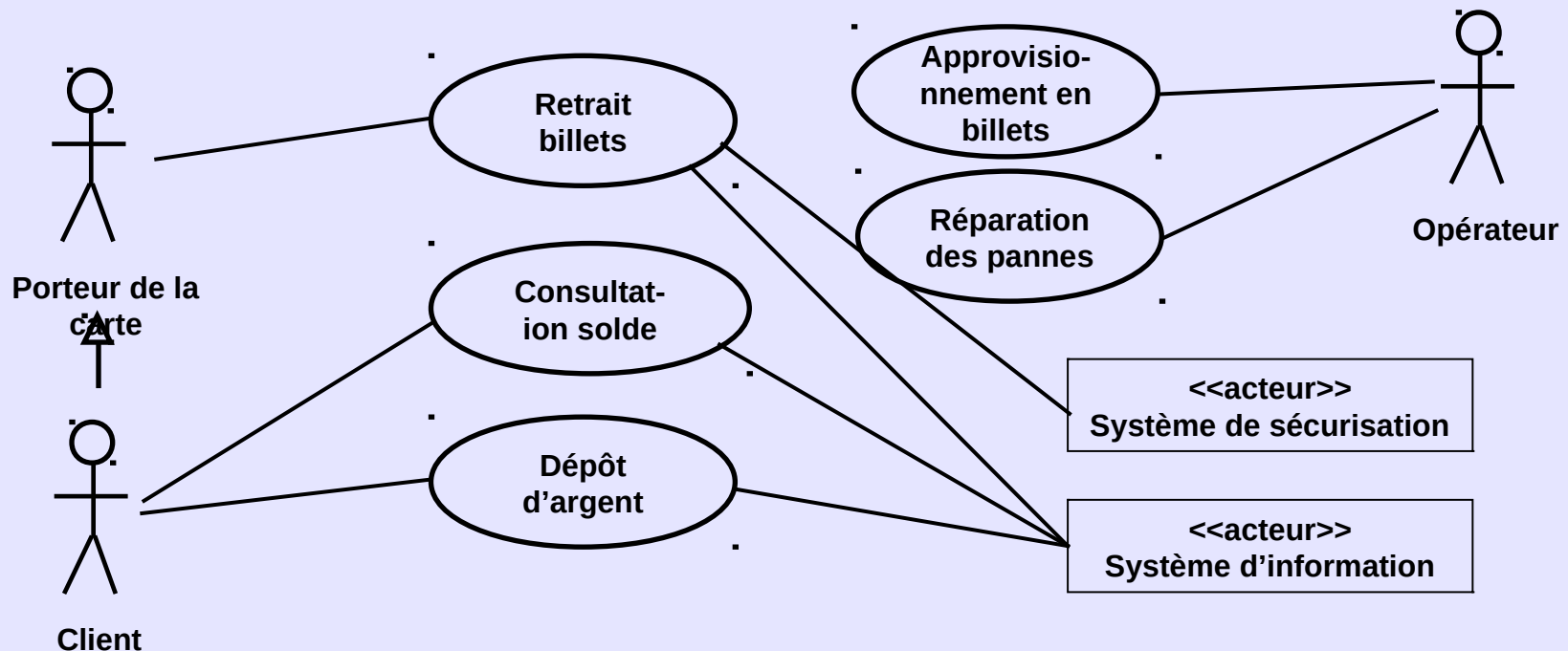
- Une démarche itérative et incrémentale
- Une démarche pilotée par les besoins des utilisateurs

Cas d'utilisation

- Un **cas d'utilisation** est utilisé pour définir le comportement d'une entité sans révéler la structure interne de l'entité.
- Chaque cas d'utilisation spécifie une séquence d'actions, y compris des variantes, que l'entité réalise, en interagissant avec les acteurs de l'entité.
- La description d'un cas d'utilisation peut se faire, de la plus informelle (textuelle) à la plus formelle (machines à états).
- Un **scénario** représente une séquence d'actions observable. Un cas d'utilisation est une abstraction de plusieurs chemins d'exécution, un scénario est une instance d'un cas d'utilisation.

Cas d'utilisation

- Un cas d'utilisation est un ensemble de scénarios partageant le même but. On distingue le scénario nominal des scénarios alternatifs des erreurs.

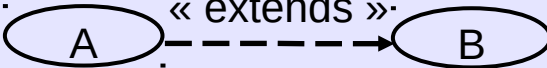


Cas d'utilisation

3 sortes de relation peuvent exister entre cas d'utilisation

1. Inclusion : 

Si un cas d'utilisation A fait appel pendant son exécution à un autre cas d'utilisation B pour réaliser son but, il s'agit alors d'une relation d'inclusion entre cas

2. Extension : 

Un cas d'utilisation A étend un autre cas d'utilisation B, si B peut faire appel (ie., déclencher l'exécution) à A

3. Héritage: 

Un cas d'utilisation B est une généralisation d'un autre cas d'utilisation A, si A est une spécialisation de B.

→ Il peut y avoir des généralisations entre acteurs

Scénario et diagramme de séquence

- Les scénarios peuvent être classés en scénarios principaux (le plus souvent, le chemin "normal" d'exécution du cas d'utilisation) et en scénarios secondaires (cas alternatif, cas exceptionnel ou cas d'erreur).
- Le **script** est la description du scénario en langage naturel.
- Chaque scénario est représenté sous forme d'un ou plusieurs **diagrammes de séquence**. Le diagramme de séquence ne couvre pas nécessairement toute l'exhaustivité du scénario ; il peut en représenter une perspective. Un diagramme de séquence montre une interaction entre les objets et les messages qu'ils échangent présentés en séquence dans le temps.
- UML introduit des mécanismes dans les diagrammes de séquence pour prendre en compte des collaborations sophistiquées entre les objets :
 - dates (pour l'émission et la réception de messages non instantanés),
 - lignes de vie des objets.

Description d'un Scénario

Un scénario est une série d'actions numérotées se déroulant en séquentiel généralement.

Titre :

Résumé :

Acteurs :

Date de création : Date de mise à jour:

Version : Responsable :

Préconditions : condition de début du scénario

Scénario nominal (principal) :

Enchainements alternatifs : permet de bifurquer à partir d'une action et de revenir ensuite au scénario nominal

Enchainements d'erreurs : dans le cas où la postcondition ne peut pas être vérifiée

Postconditions : condition de réussite du scénario

Titre : acheter un produit sur le web

Précondition :

Le client doit être titulaire d'une carte bancaire

Le client doit avoir une connexion internet

Le produit présent dans le catalogue doit être disponible

Scénario nominal :

1. Le client parcourt le catalogue et sélectionne des articles

2. Le système ajoute les produits sélectionnés au panier

3. Le client accède à son panier et décide de valider la commande

4. Le client fournit des informations de la livraison (adresse, livraison expresse, etc.)

5. Le système vérifie l'exactitude de l'adresse via son SIG

6. Le client introduit les informations de la carte bancaire, et l'adresse mail pour l'envoi de la facture électronique

7. Le système confirme la vente

8. Le système envoie un email au client de confirmation de la vente

Extensions :

4a: le client est un client régulier

1. Le système affiche les informations enregistrées, au client : lieu de livraison et @ mail

2. Le client valide les informations affichées ou décide de les modifier, et passe à l'étape 7

Alternatives :

7.a : le système n'autorise pas la vente

1. Le client peut re-saisir ses informations bancaires retour à l'étape 6 ou décide d'annuler la vente (quitter ce scénario).

Postcondition:

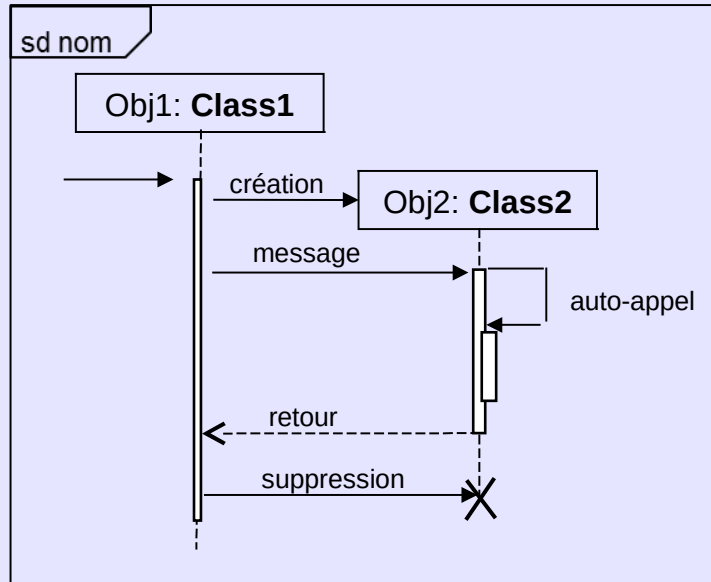
Vente réalisé, facture envoyée et compte client débité.

Diagramme de Séquence

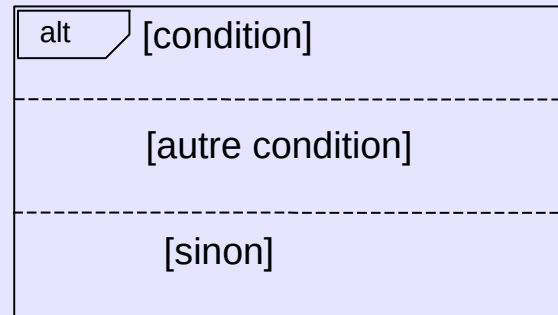
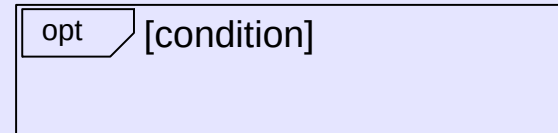
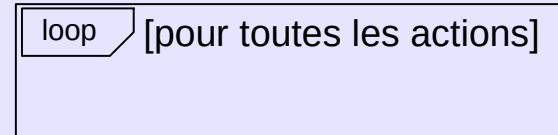
- Capture le comportement d'un **seul scénario**
- Contient un certain nombre d'objets (instances) et les messages échangés entre ces objets dans le cadre **d'un cas d'utilisation**

| Opérateur | Interprétation |
|-----------|---|
| alt | Plusieurs fragments possibles (alternatives). Celui dont la condition est vraie s'exécutera |
| opt | Optionel. Le fragment ne s'exécutera que si la condition associée est vraie |
| par | Parallèle. Chaque fragment est exécuté en parallèle |
| loop | Itération. Le fragment peut s'exécuter plusieurs fois et la garde indique la condition de l'itération |
| negative | Négatif. Le fragment représente une interaction invalide |
| region | Région critique. Le fragment ne peut avoir qu'un thread qui l'exécute à la fois |
| ref | Référence. Se réfère à une interaction définie dans un autre diagramme. Il est possible de définir des paramètres et une valeur de retour |
| sd | Diagramme de séquence. Sert à englober la totalité du diagramme |

Diagramme de Séquence

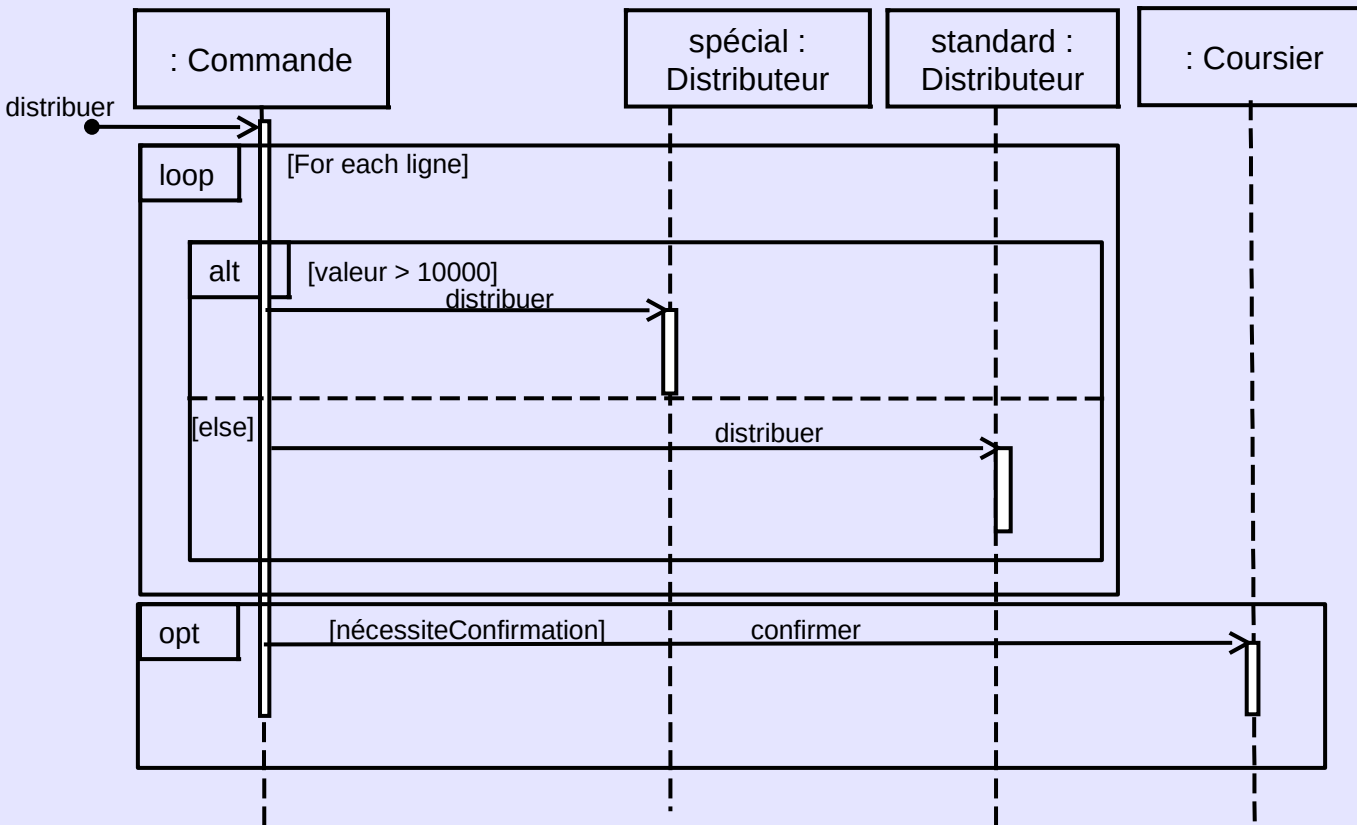


→ synchrone
→ asynchrone



- Un message synchrone signifie que l'appelant attend la réception de la réponse de l'appelé pour continuer son exécution
- Un message asynchrone permet à l'appelant de continuer son exécution en parallèle de l'activité de l'appelé

Diagramme de Séquence



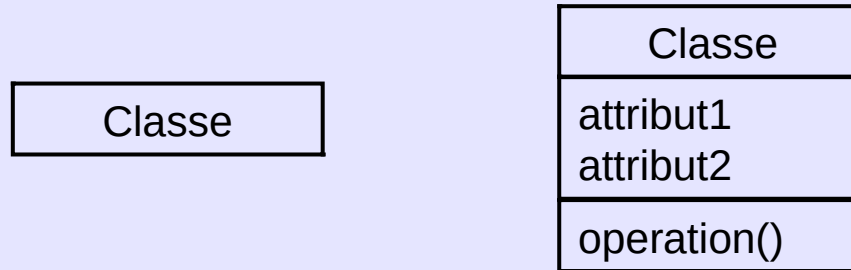
```
procedure distribuer{
  foreach(ligne) {
    if(produit.valeur > 10000)    spécial.distribuer()
    else        standard.distribuer()
  }
  if(nécessiteConfirmation)    coursier.confirmer()
}
```

Scénario et diagramme de communication

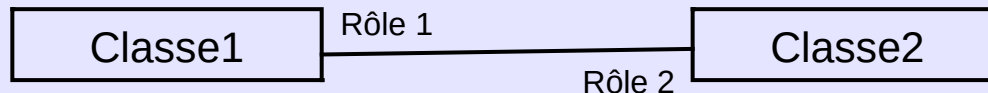
- Un diagramme de communication montre une interaction organisée autour des objets et de leurs liens
- Les diagrammes de communication sont utilisés pour présenter d'autres aspects, très difficiles à montrer dans un diagramme de séquence :
 - les objets composites représentés dans un diagramme de communication par imbrication graphique des objets
 - la concurrence
 - la nature des messages :
 - synchrone (l'émetteur attend que le récepteur reçoive le message),
 - asynchrone (l'émetteur n'attend pas que le récepteur reçoive le message),
 - borné (l'émetteur spécifie un temps d'attente de la réception, pas de la réponse).

Diagramme de classes

Classe : une entité du domaine “métier” à modéliser, composée d’attributs et pas nécessairement d’opérations.



Association : un lien logique ou physique entre 2 ou plusieurs entités. Elle permet d’attribuer des rôles aux entités classes qui y participent.



Multiplicités : le nombre de participation d’une **instance** dans une association (*, min..max)

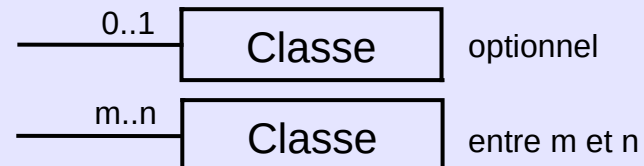
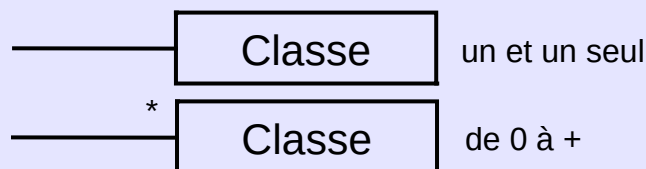
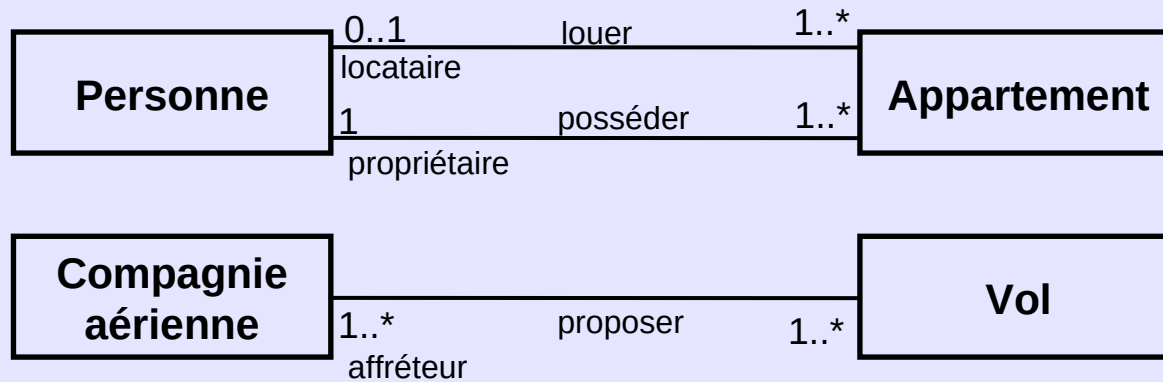
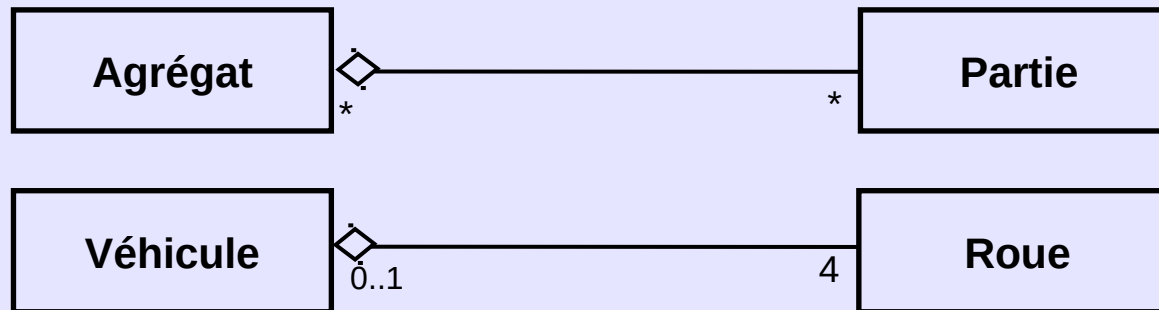


Diagramme de classes



Agrégation



Composition

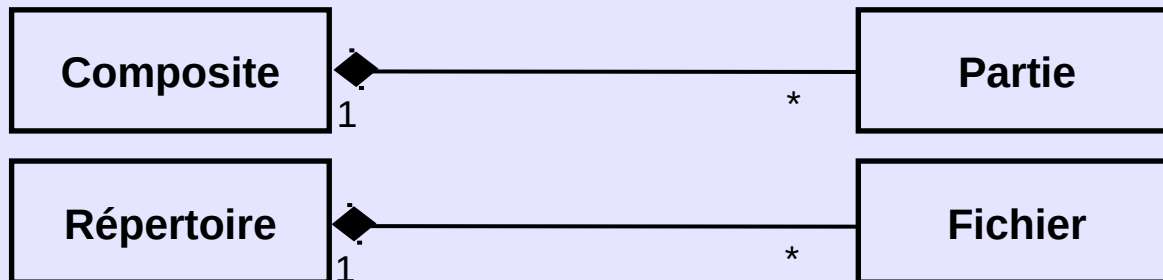
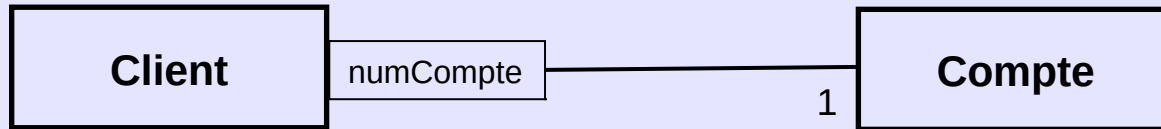


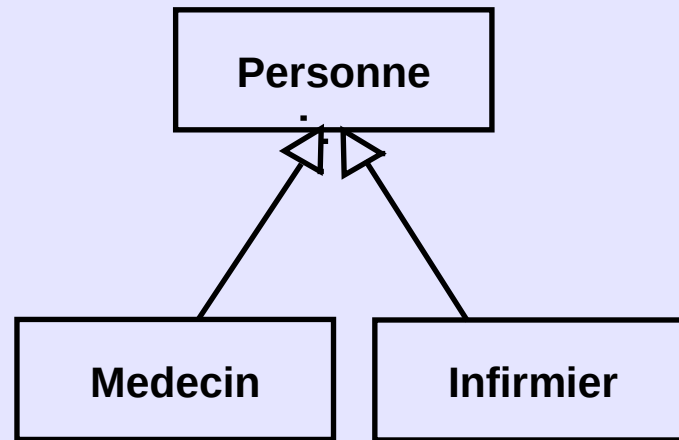
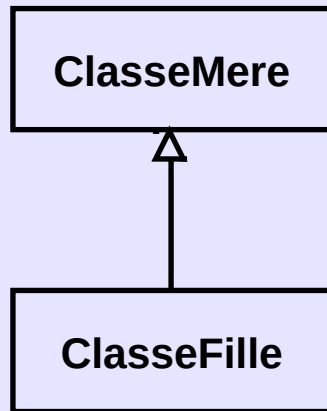
Diagramme de classes

Qualification



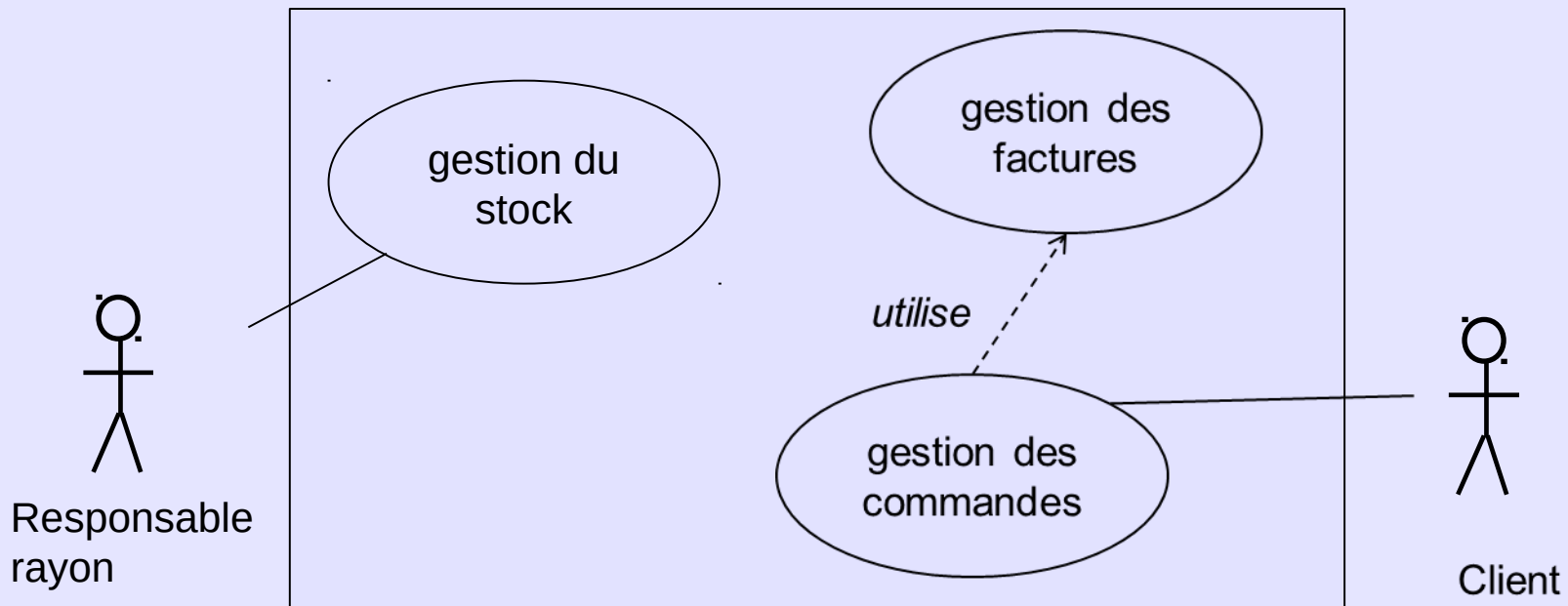
Généralisation et Spécialisation

Une classe peut être une généralisation d'une ou plusieurs classes. Ces classes sont des spécialisations (sous-types) de cette classe



Cas d'utilisation pour le problème de facturation de commandes

- Qui interagit avec le système pour gérer une commande, pour facturer une commande ou pour gérer le stock ?
 - une commande est faite par un client,
 - une commande est facturée par le système lui-même (si elle est réalisable)
 - une nouvelle personne (responsable rayon) est chargée de la gestion du stock.
- Vue du modèle de cas d'utilisation :



Définition des scénarios

| cas d'utilisation | scénario | acteur |
|-----------------------|-----------------------|----------------------------|
| gestion des commandes | entrer une commande | Client |
| | annuler une commande | Client |
| gestion des factures | facturer une commande | <i>Sans acteur externe</i> |
| gestion du stock | entrer un produit | Responsable rayon |

Diagramme de séquence du scénario "facturer une commande"

"L'état d'une commande passe à facturée si la quantité ordonnée est \leq au stock du produit ordonné"

→ vérifier si la commande peut être satisfaite (nombre en stock \geq nombre commandé),
et alors le stock doit être modifié et la commande facturée (on change son état).

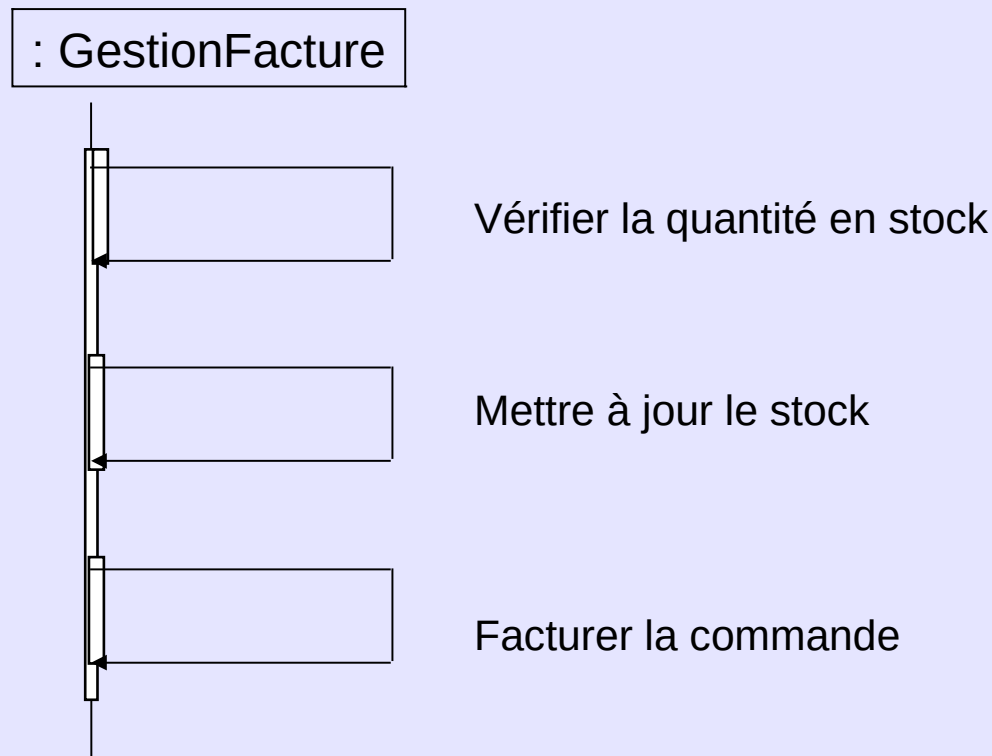


Diagramme de communication du scénario "facturer une commande"

On décrit seulement les objets nécessaires pour les cas d'utilisation :

- **Objet Facture ?**

Si le système envoie les factures, une facture est un objet avec des attributs (prix par exemple) et "facturer une commande" consiste à créer un nouvel objet "Facture". Sinon, "facturer une commande" est considéré comme une action sur l'état de la commande, il n'y a pas de classe "Facture" et la classe "Client" ne mandate pas de facturation.

- **Objets Produit et Commande ?**

Si la quantité du produit dans le stock est suffisante, la commande peut être facturée et le stock actualisé.

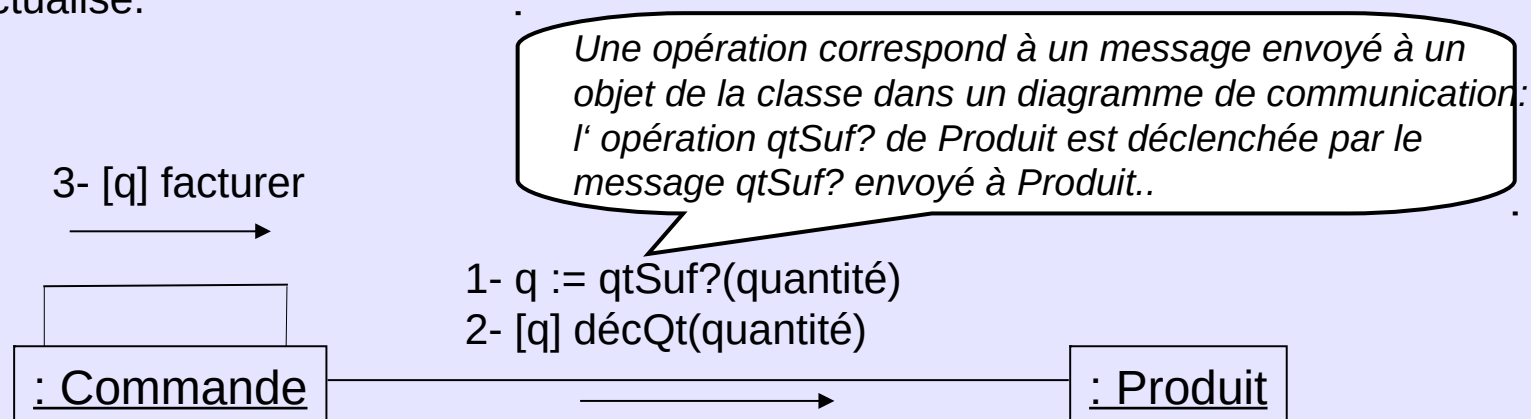
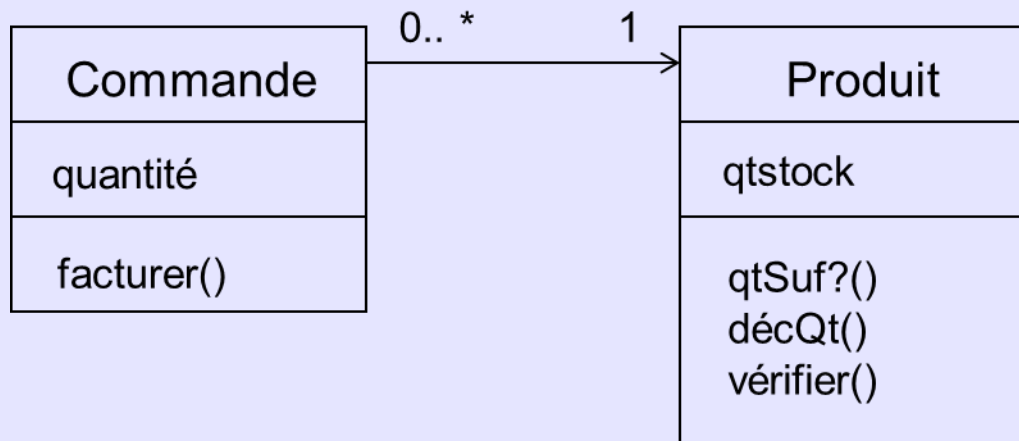


Diagramme partiel de classe pour la facturation d'une commande

- Un diagramme de classe est un modèle statique, formé des classes, des interfaces et leurs relations.
- Les attributs et associations peuvent être déduits à partir des paramètres des messages.
Les associations peuvent être déduites à partir de sens de messages.
Les identificateurs de classes peuvent être déduits à partir des paramètres.
- Sur une commande, on a une et seulement une référence à un produit commandé avec un nombre quelconque. On suppose un produit par commande, alors la cardinalité de Commande à Produit est 1.



Statechart d'une commande

- Un statechart spécifie le comportement local d'une classe. L'état d'un objet est une abstraction de la valeur de cet objet à un instant donné. Une transition représente le changement d'un état vers un autre. Ceci peut être associé à un événement avec une condition et des actions à exécuter.
- 2 états ont été identifiés : attente et facturée. Initialement, une commande est en attente et devient facturée si l'opération de facturation est lancée et si la quantité du produit dans le stock est suffisante (quantité en stock \geq quantité commandée)

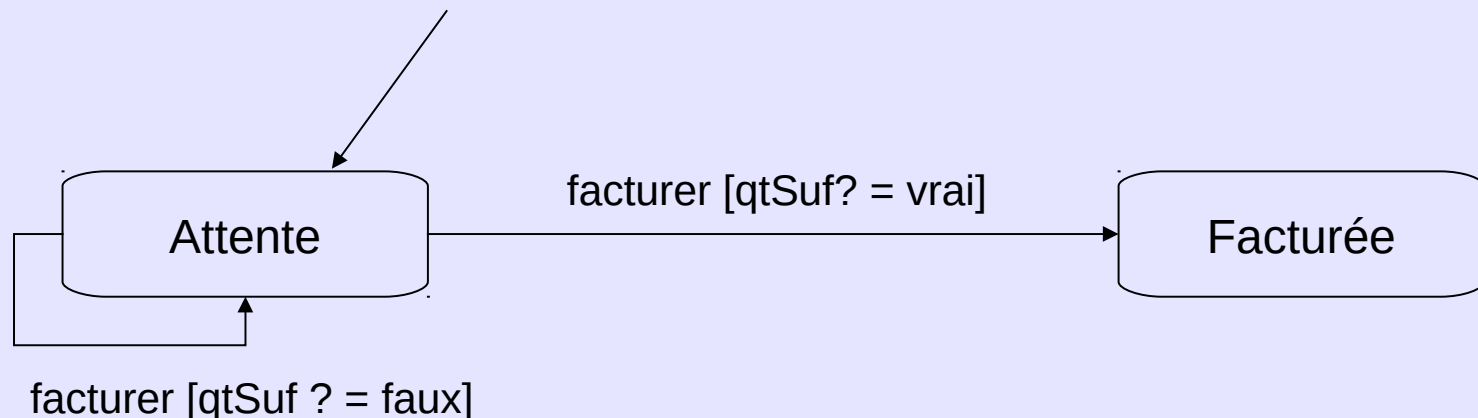


Diagramme de séquence du scénario "entrer une commande"

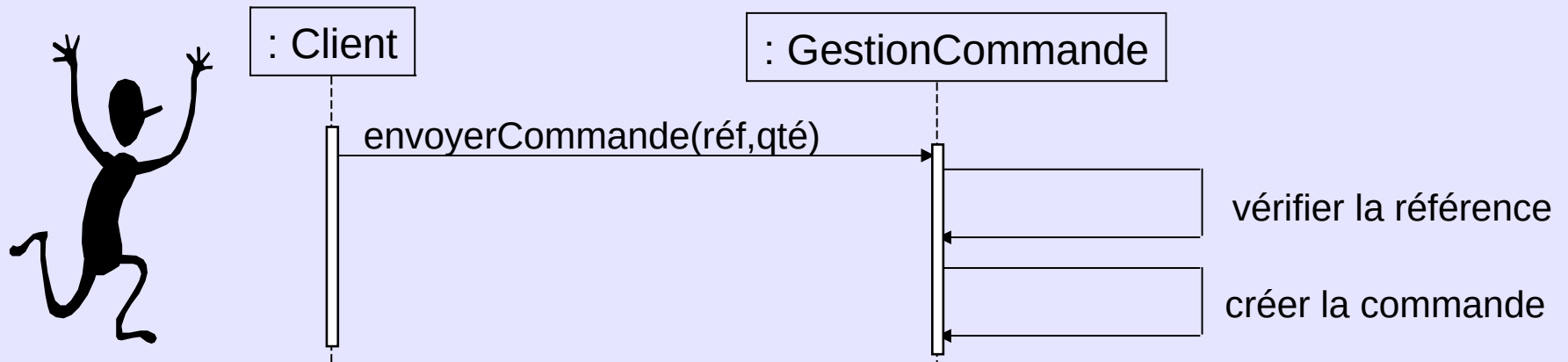


Diagramme de communication du scénario "entrer une commande"

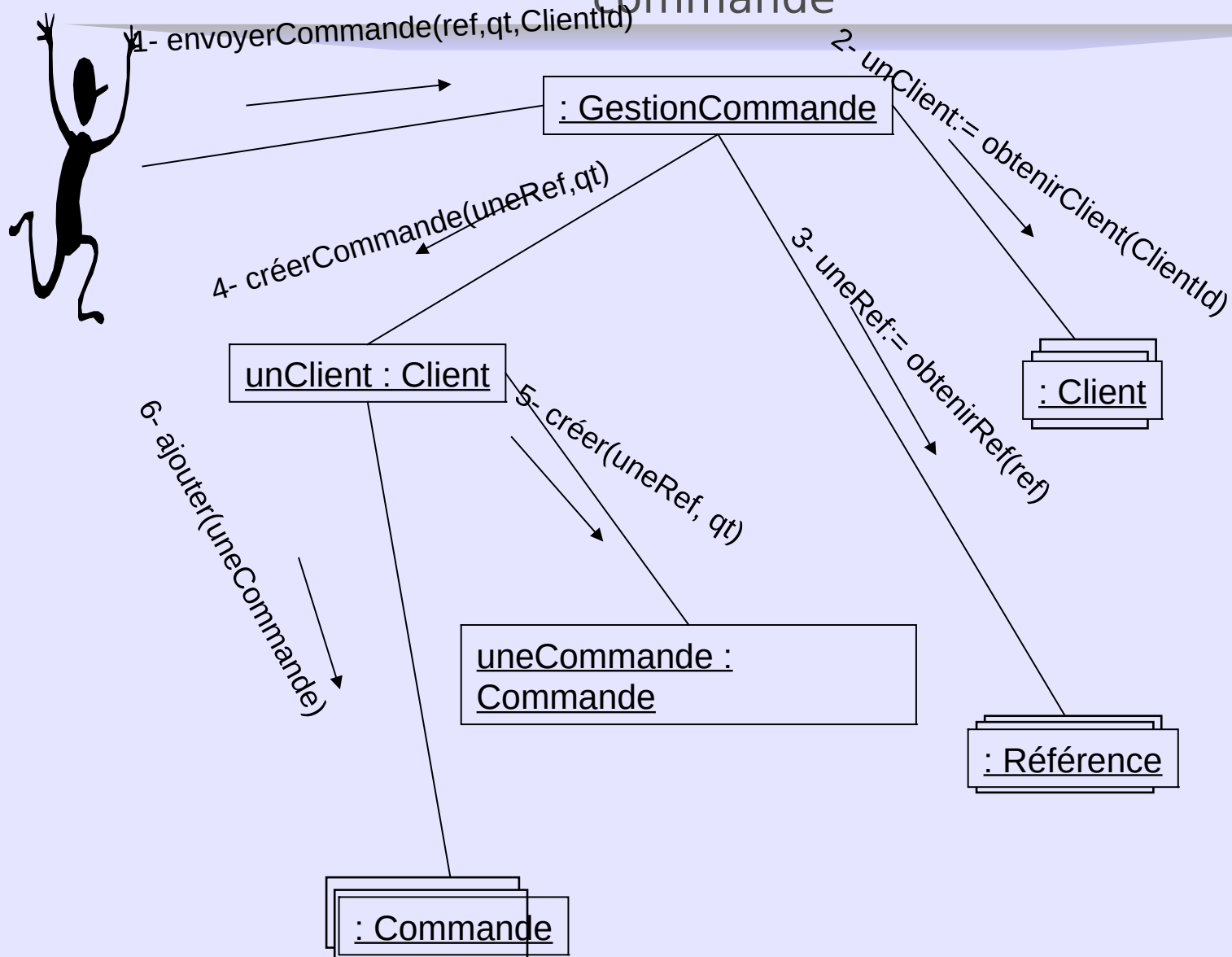


Diagramme partiel de classe pour l'entrée d'une commande

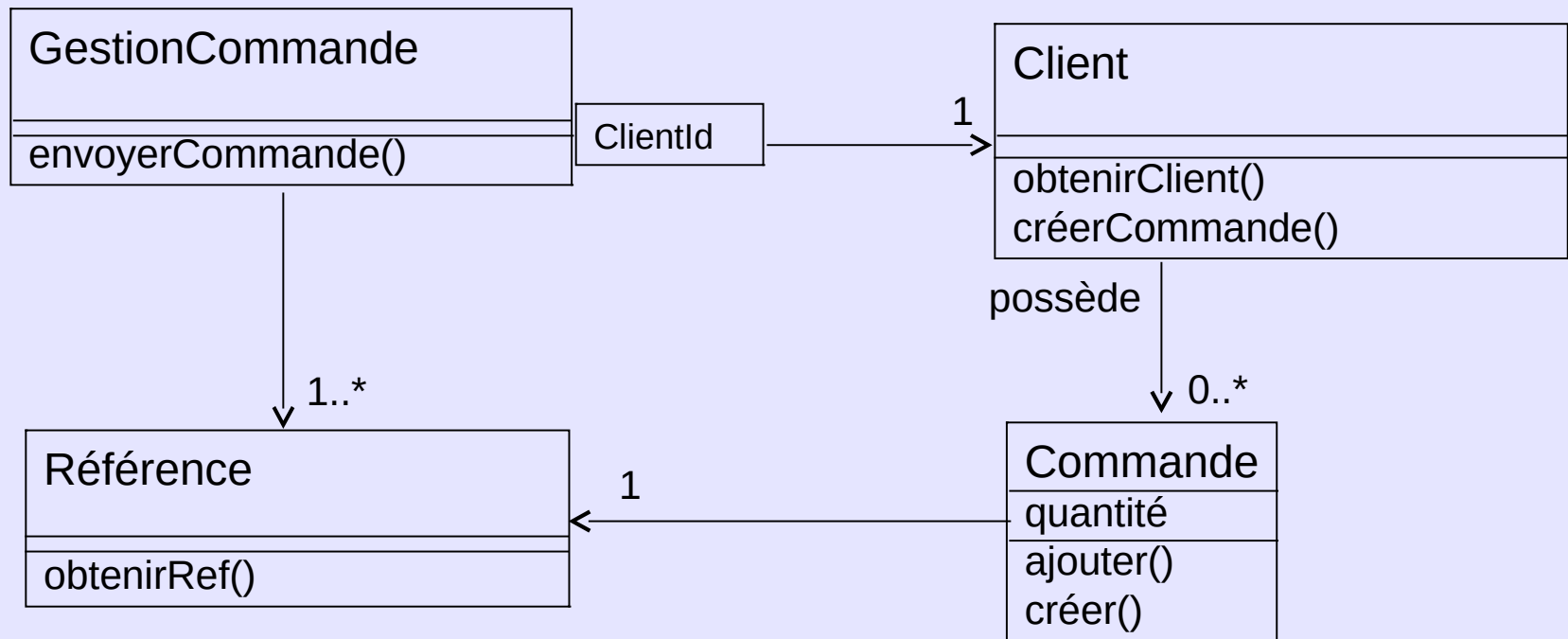


Diagramme de séquence du scénario "annuler une commande"

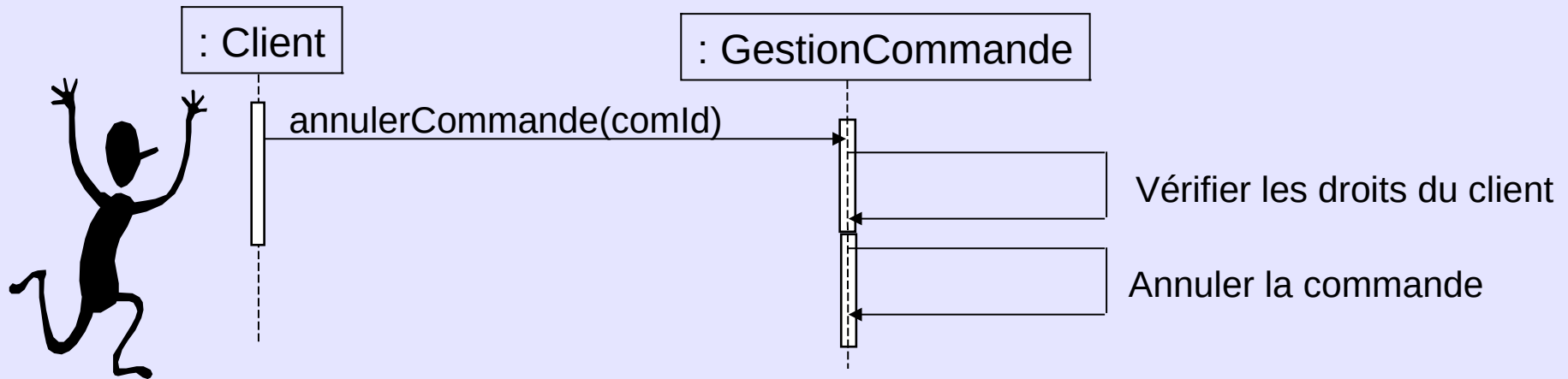


Diagramme de communication du scénario "annuler une commande"

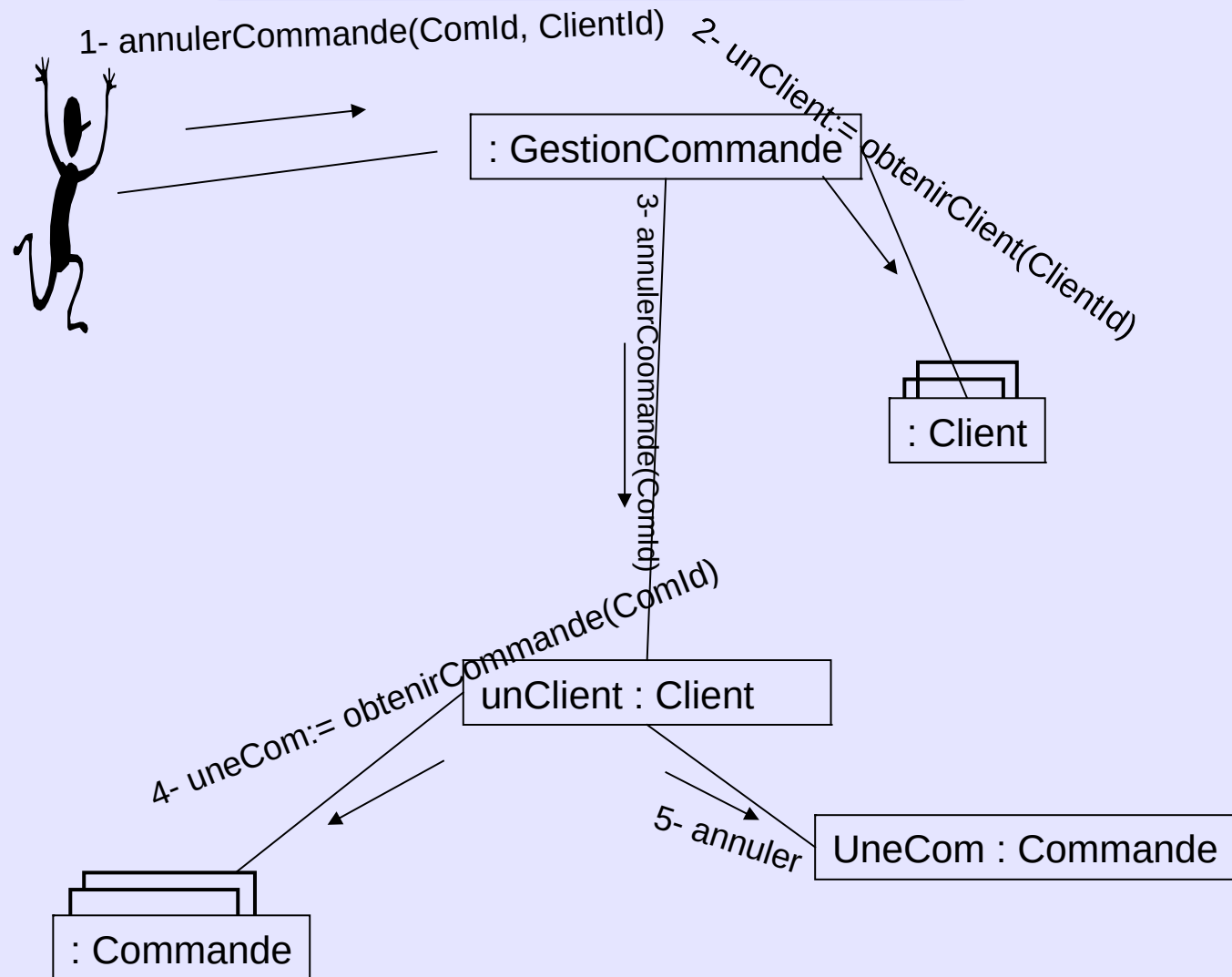


Diagramme partiel de classe pour l'annulation d'une commande

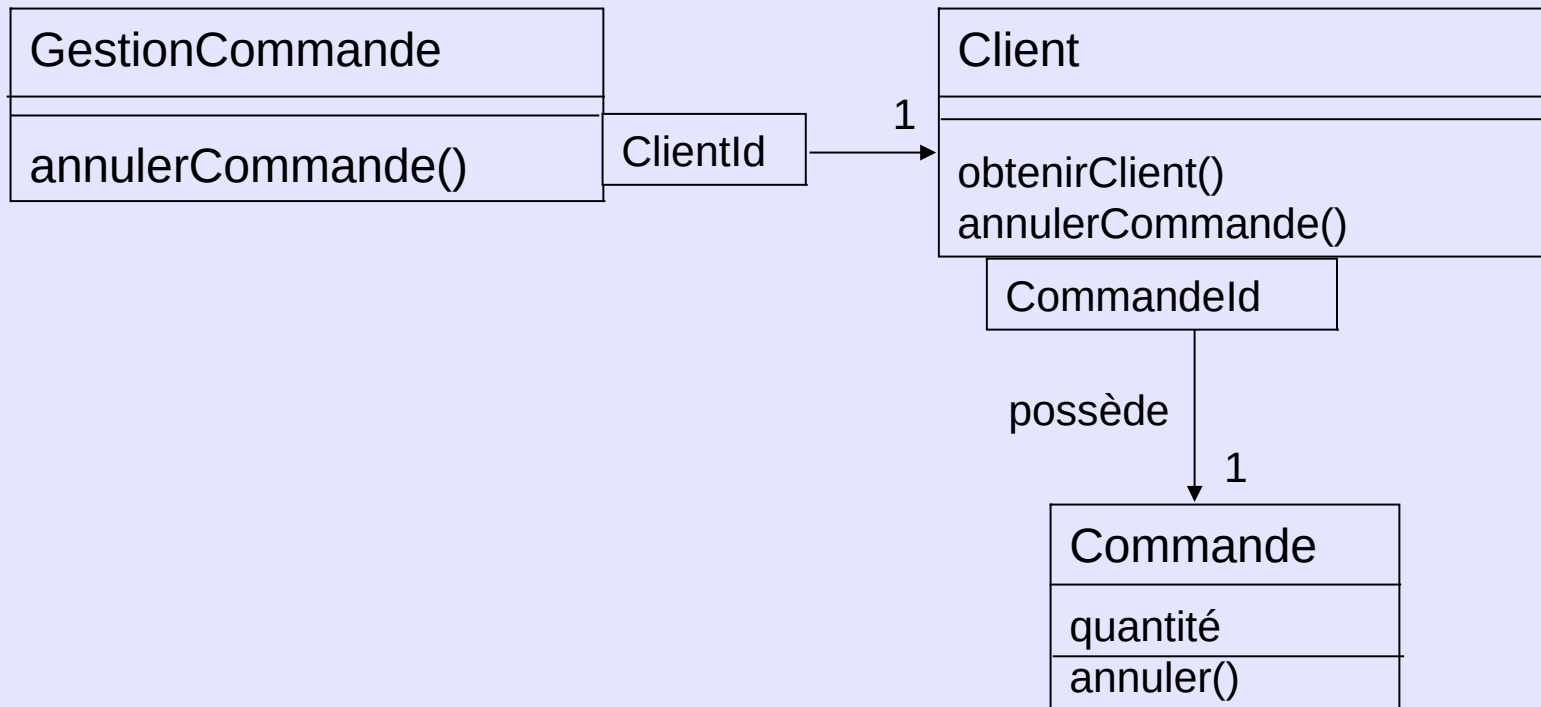
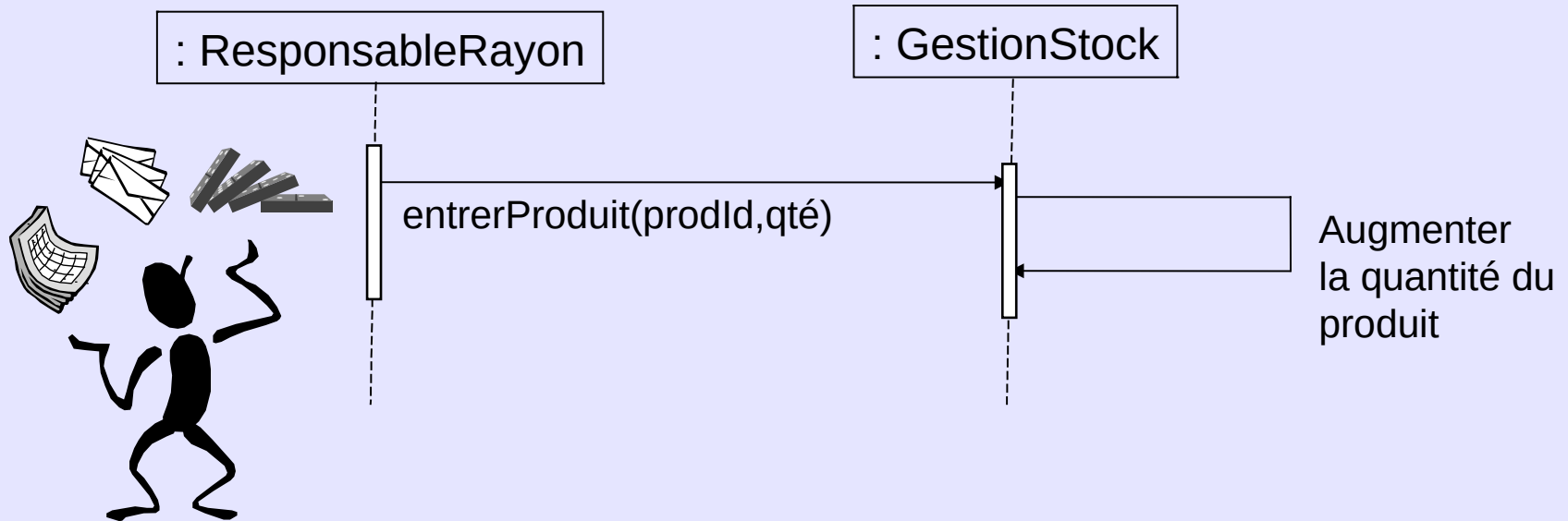


Diagramme de séquence du scénario "entrer un produit"



Exercice

- Donnez :
 - le diagramme de communication pour l'entrée d'un produit
 - le diagramme partiel de classe pour l'entrée d'un produit
 - le diagramme global de classe
 - le diagramme d'état pour une commande

Diagramme de communication du scénario "entrer un produit"

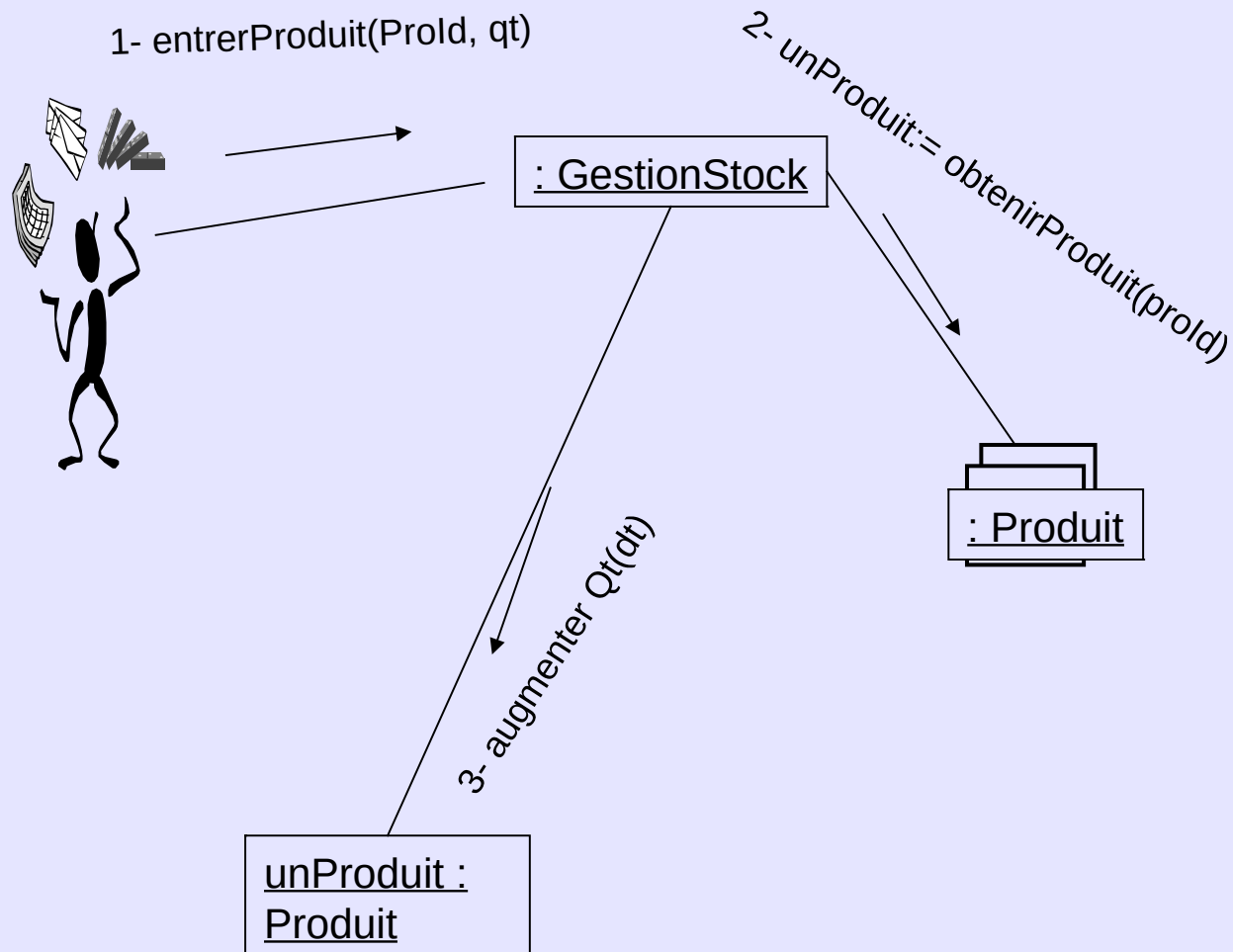


Diagramme partiel de classe pour l'entrée d'un produit

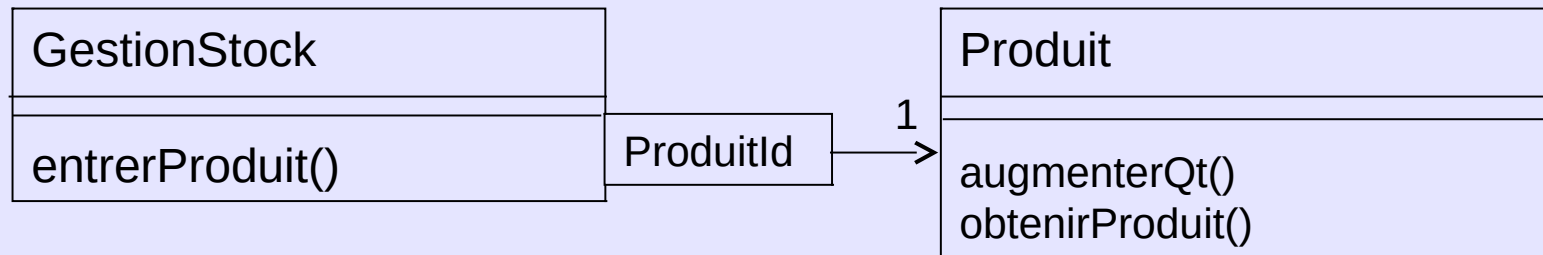


Diagramme global de classe

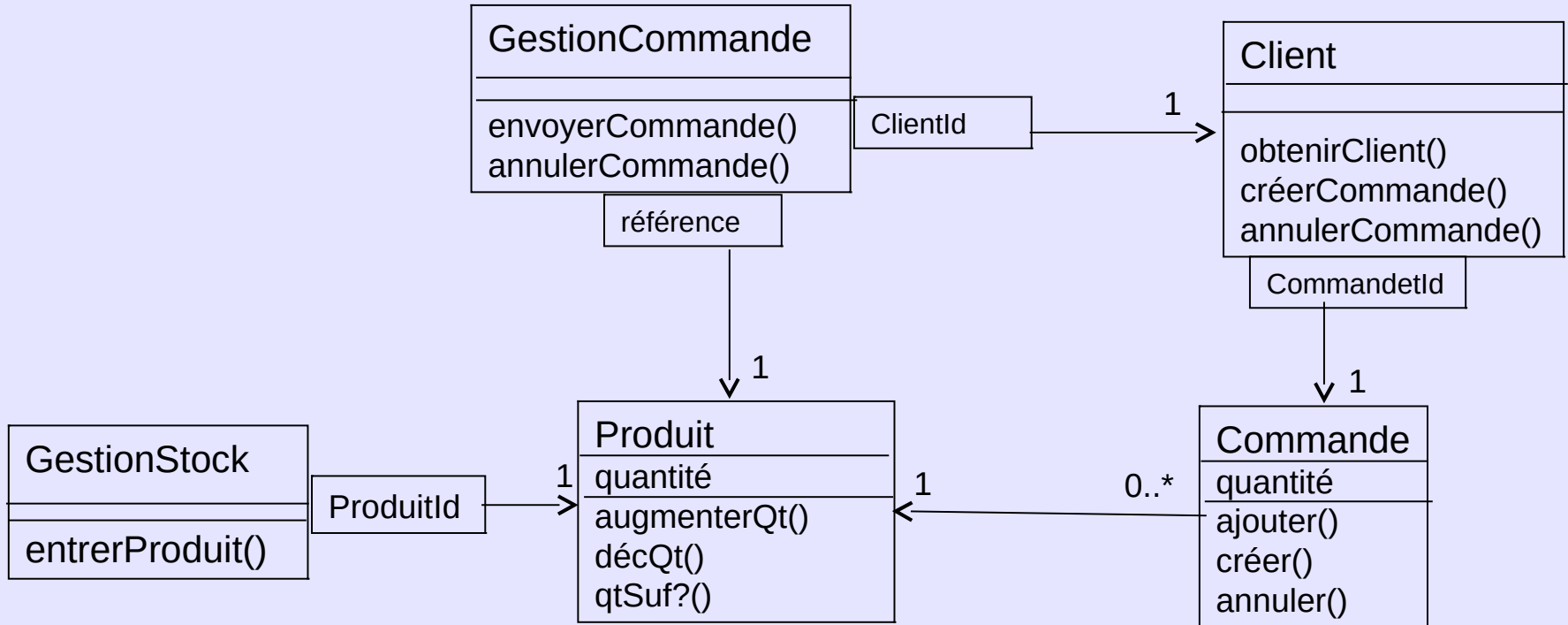
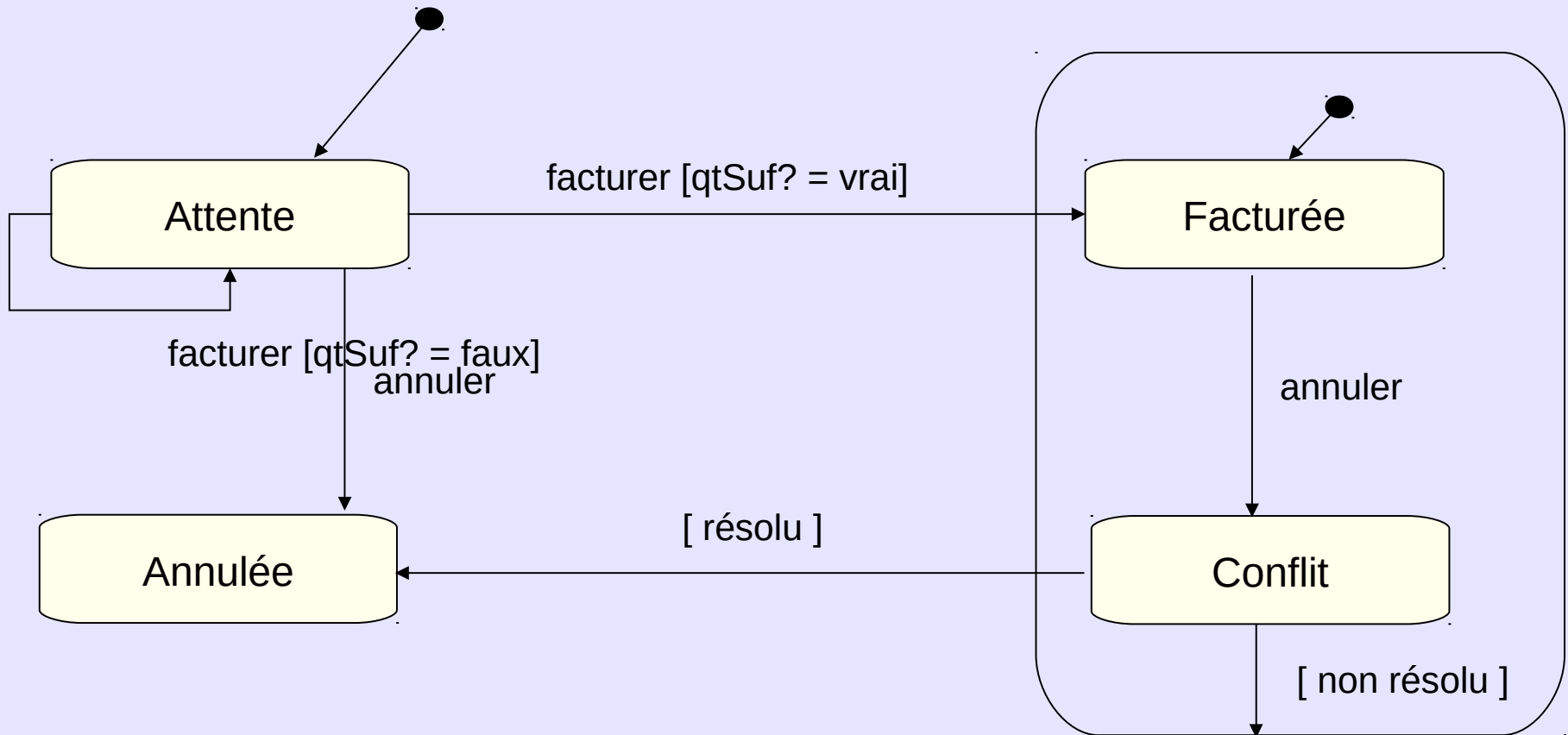


Diagramme d'état de Commande



Comportement global

A quel moment une commande est-elle facturée ?

Une entrée de produit peut déclencher la facturation de commandes en attente
=> les interactions entre les scénarios (leur synchronisation) induisent plusieurs règles de gestion à prendre en compte. Les scénarios décrits précédemment sont indépendants, ils sont à la base de scénarios plus complexes.

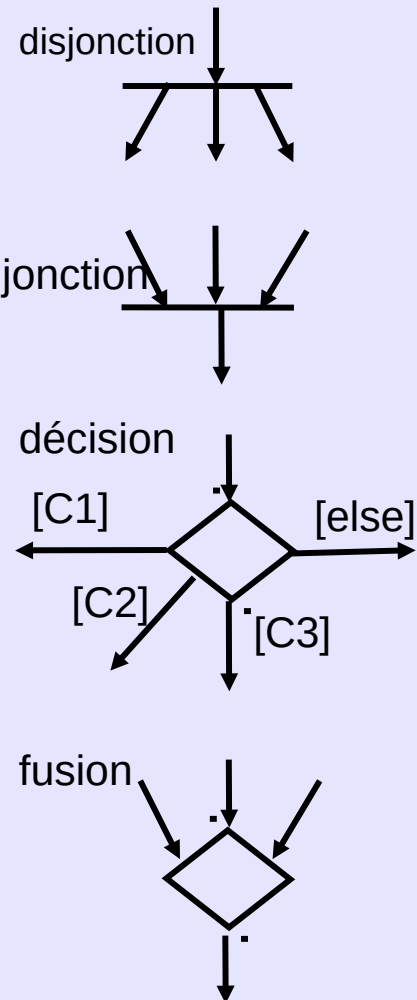
UML

- Diagramme d'activités
- Diagramme d'objets
- Diagramme de packages
- Diagramme de déploiement

Diagramme d'activité

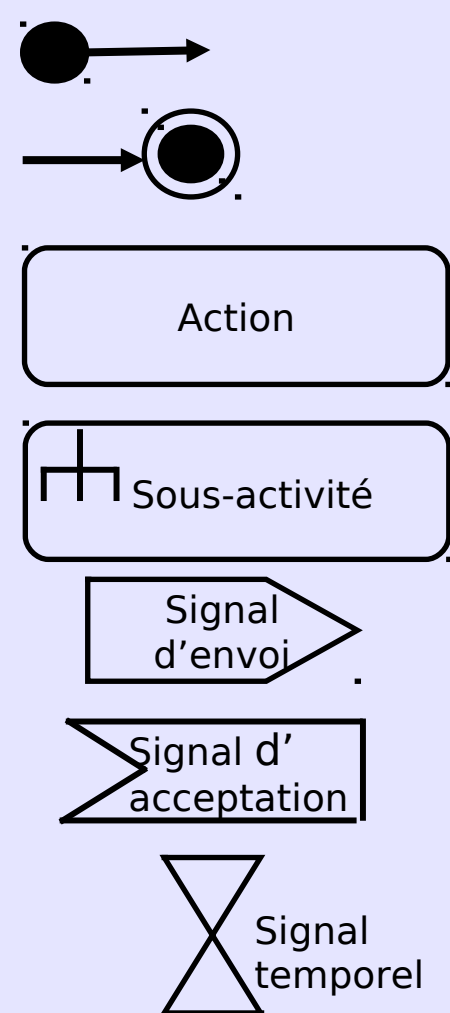
- Décrit la logique procédurale, les processus métier et les enchaînement d'activité (workflows).
- Il est comparable à un organigramme.
- Identifie les comportements possibles du processus métier modélisé, i.e., les règles d'exécution minimales d'un processus.

Diagramme d'activité



- Les opérateurs de disjonction et de jonction expriment la synchronisation d'un sous processus métier (début et fin de la synchronisation respectivement)
- Les opérateurs conditionnels décision et fusion permet de bifurquer sur un processus métier conditionnel et de le clôturer. Les points de décision comportent des conditions disjointes (une seule est vraie à la fois, si aucune n'est vraie la garde [else] est déclenchée)

Diagramme d'activité



- Les points d'entrée et de terminaison du diagramme d'activité permettent d'identifier le début et la fin de l'activité
- Une activité peut être décomposée en sous-activités et actions
- Les signaux permettent de synchroniser les actions du diagramme d'activité avec des événements de l'extérieur et des contraintes de temps

Diagramme d'activité

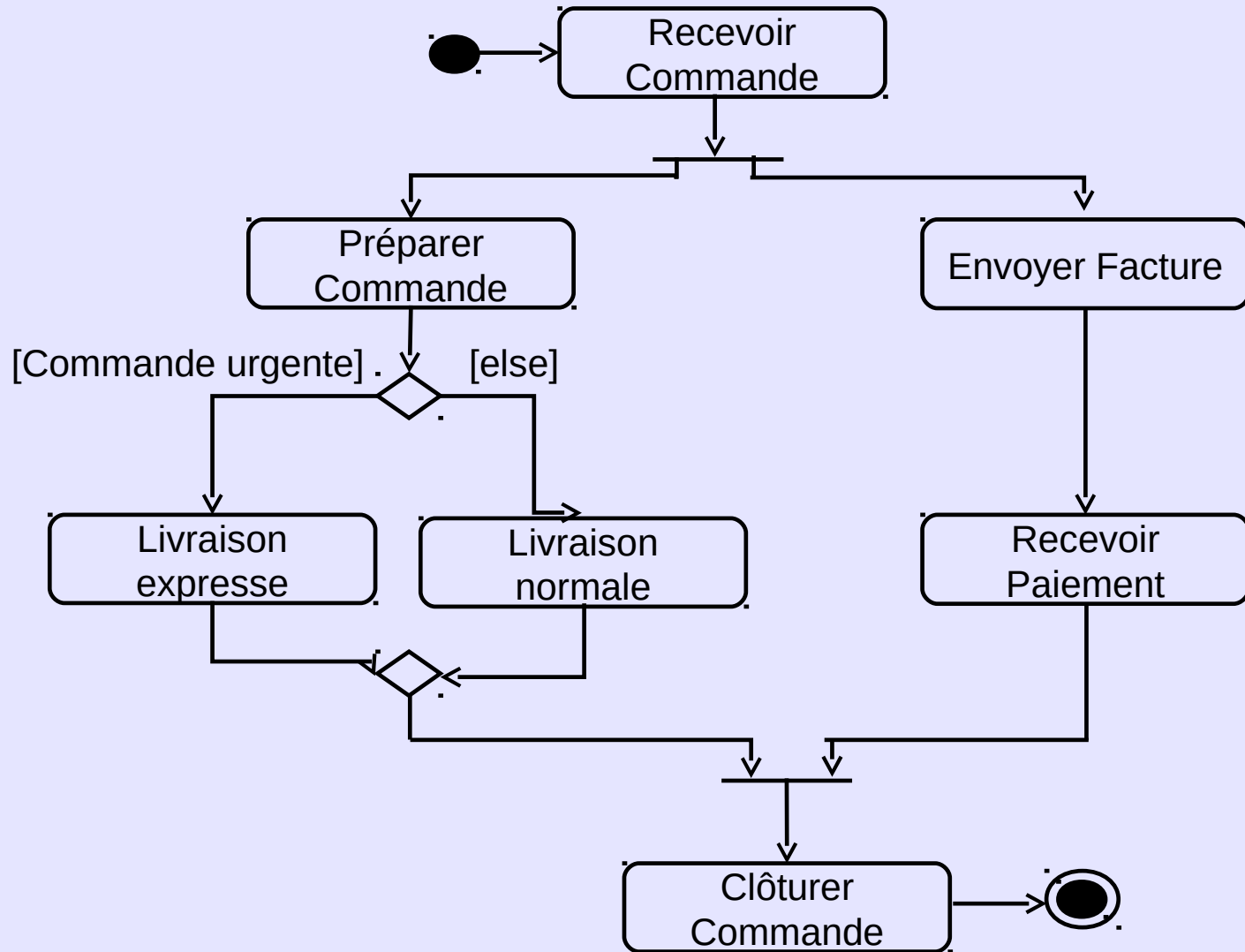


Diagramme d'activité

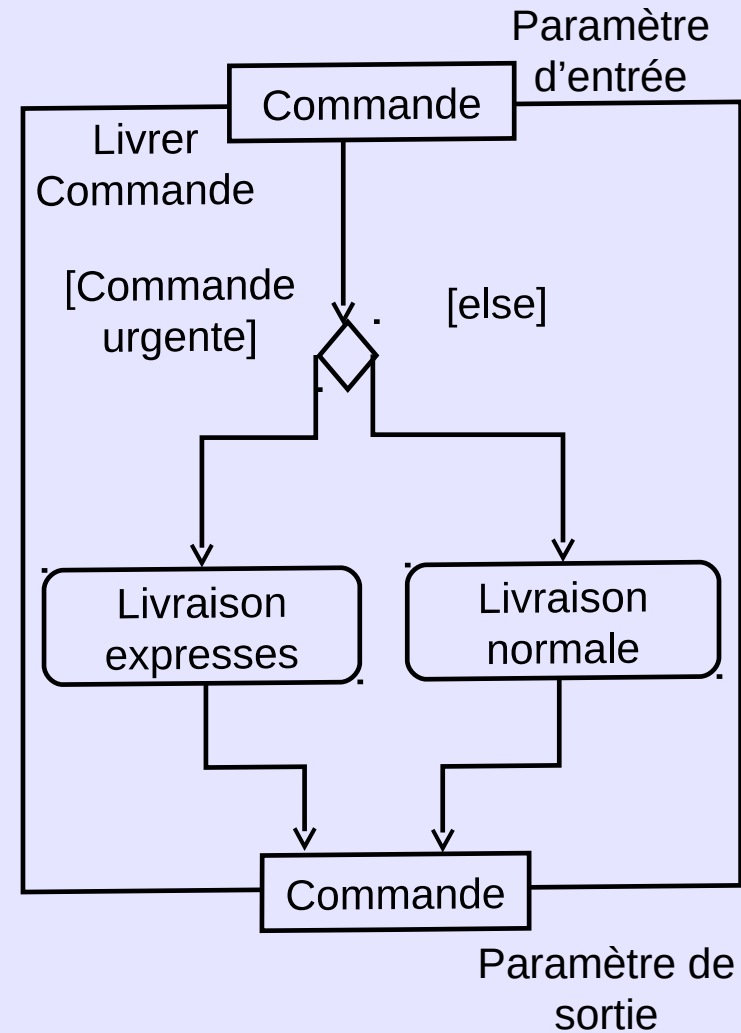
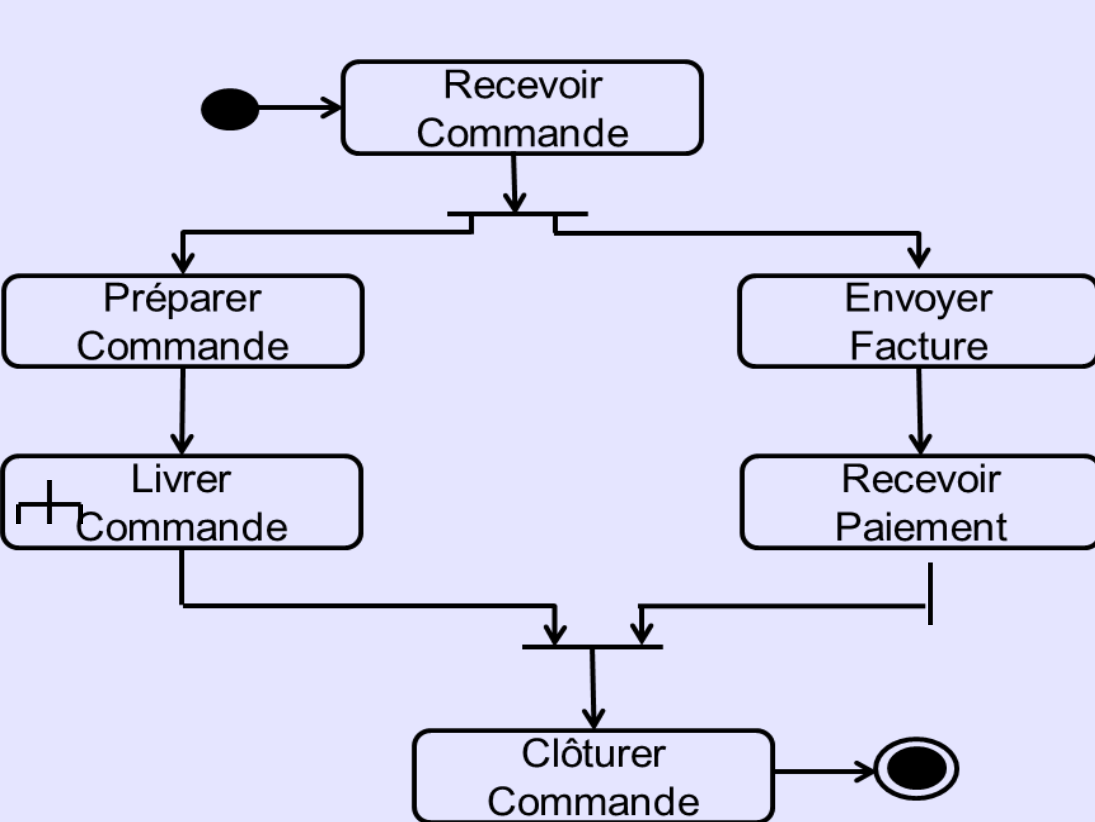


Diagramme d'activité

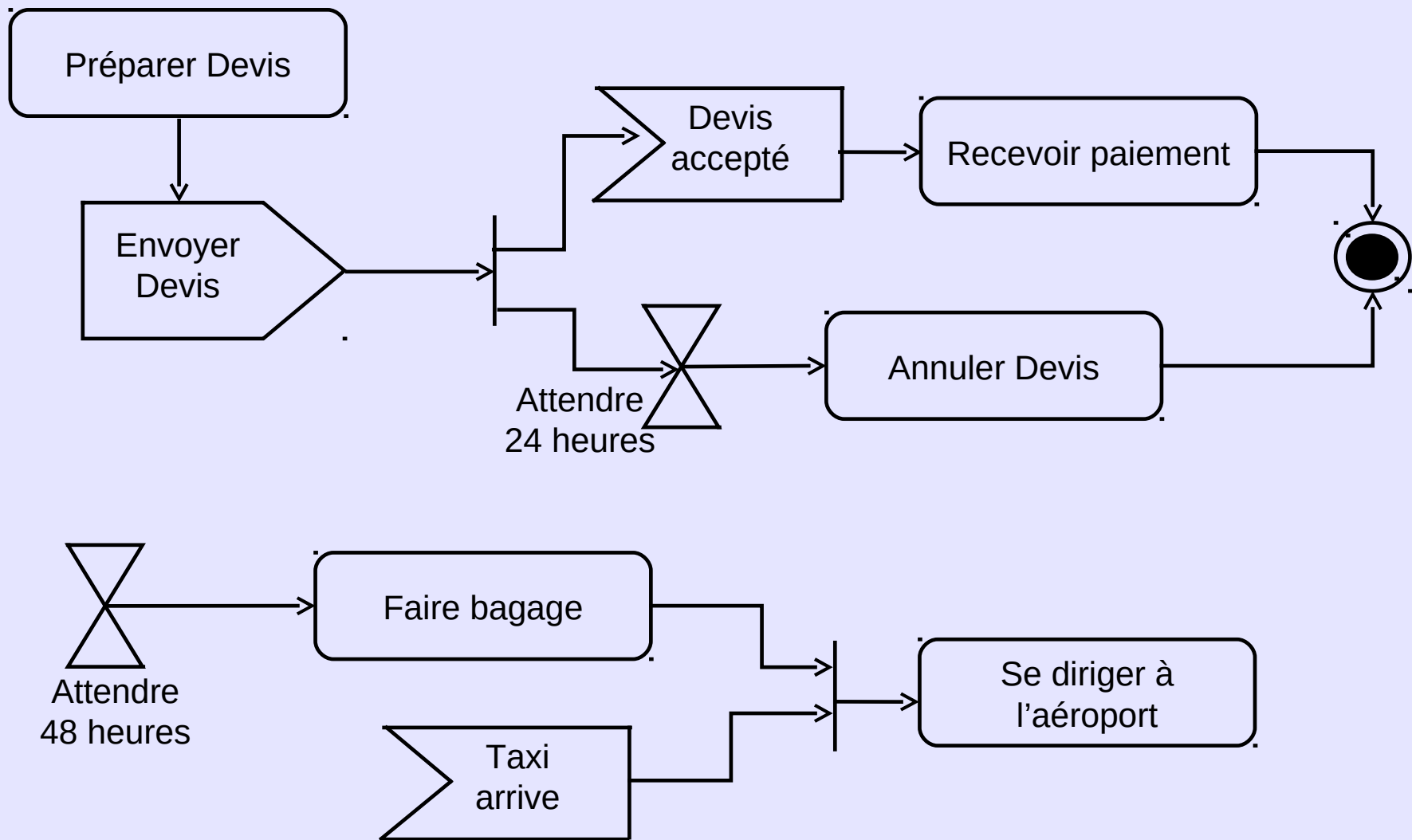


Diagramme d'objet (d'instances)

- Une image instantanée des objets de l'application à un moment donné. Il représente les instances des classes et non pas les classes.

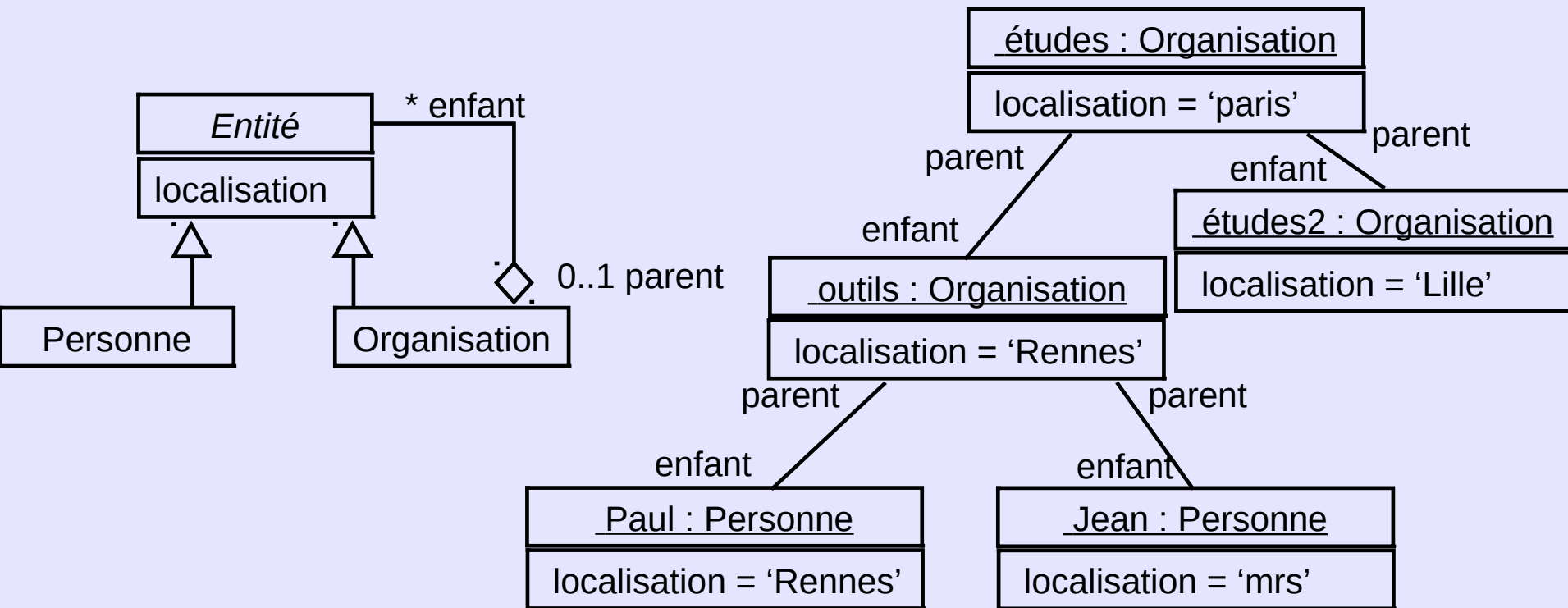
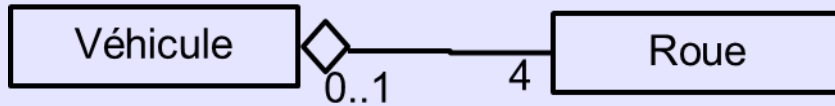
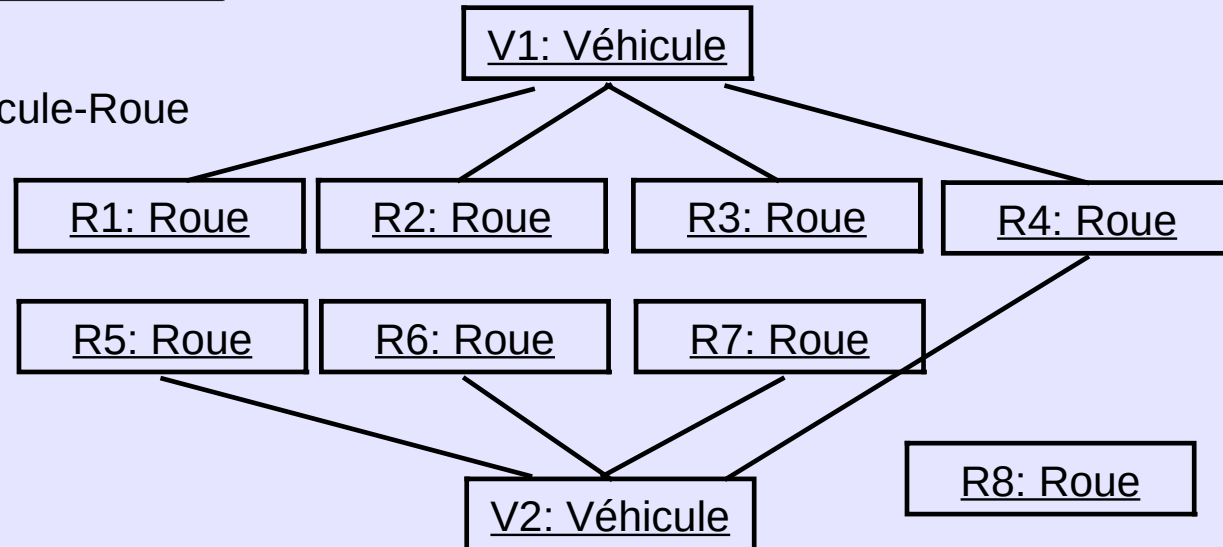


Diagramme d'objet (d'instances)



-- Diagramme de classes Véhicule-Roue



-- Diagramme d'objets

La roue R4 ne peut pas faire partie des véhicules V1 et V2 à la fois
→ **Diagramme d'objet incorrect**

Diagramme de packages

- Espace de noms
 - Il permet de prendre tout modèle d'UML et de grouper ses éléments dans des unités de plus haut niveau
- Regroupement des classes d'une application
- Un package contient des sous-packages et des classes.
 - Le nom d'un sous-package et d'une classe est unique au sein du même package

Diagramme de packages

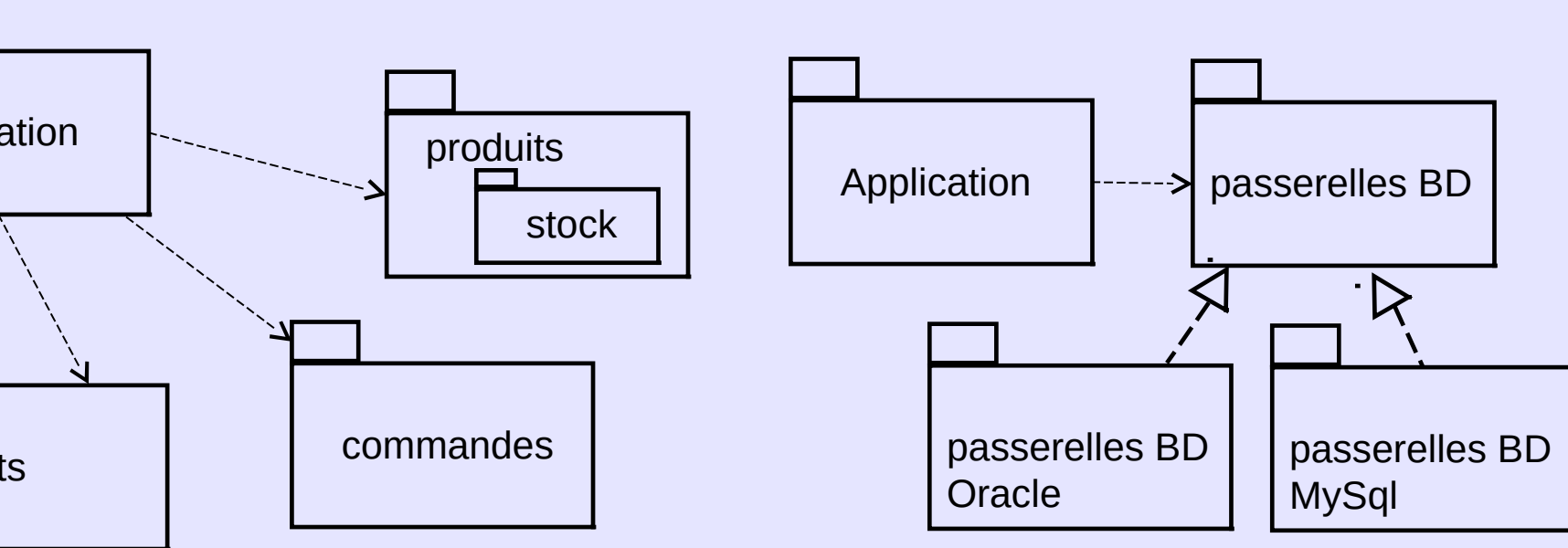
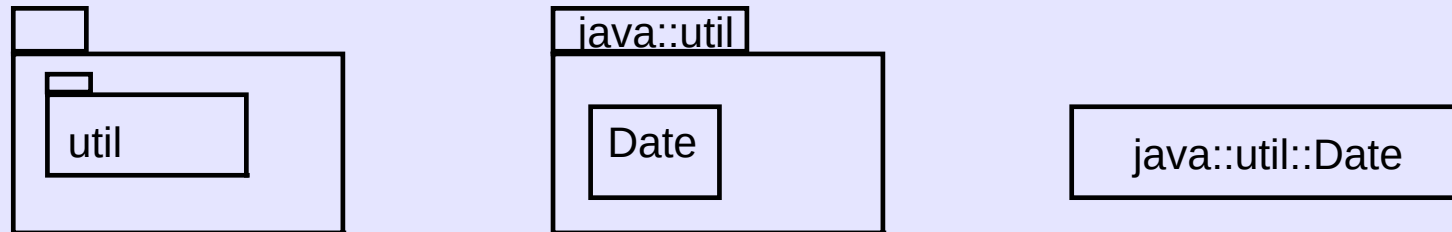


Diagramme de packages

- **Identification des packages**

- L'expérience du concepteur est primordial
- Deux principes courants :
 - Principe de fermeture commune :
les classes du même package ne doivent changer que pour des raisons similaires
 - Principe de réutilisation communes :
les classes du même package doivent être réutilisées ensemble

- **Dépendance entre packages**

- Les packages doivent présenter des dépendances unidirectionnelles et non cycliques
- Les relations de dépendances ne sont pas transitives

Diagramme de déploiement

- Représente l'agencement physique d'une application montrant sur quels composant matériels, les différents composant logiciels s'exécutent et les voies de communication.
- Nœud : unité susceptible d'héberger un logiciel
 - Equipement : unité matériel (ordinateur, unité de stockage, etc.)
 - Environnement d'exécution : logiciel qui héberge d'autres logiciels (OS, processus, etc.)

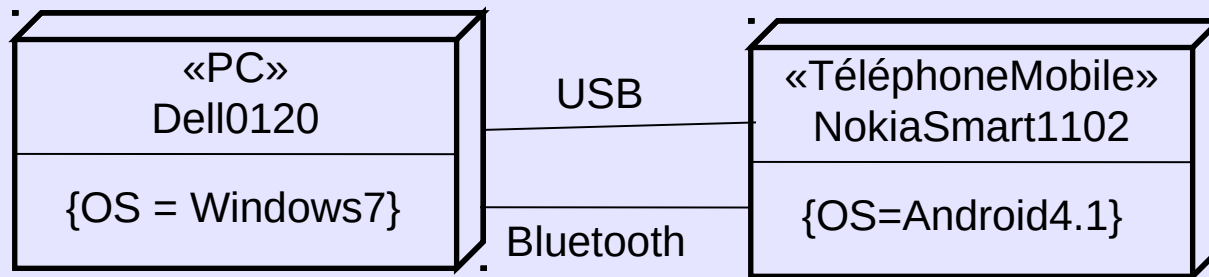
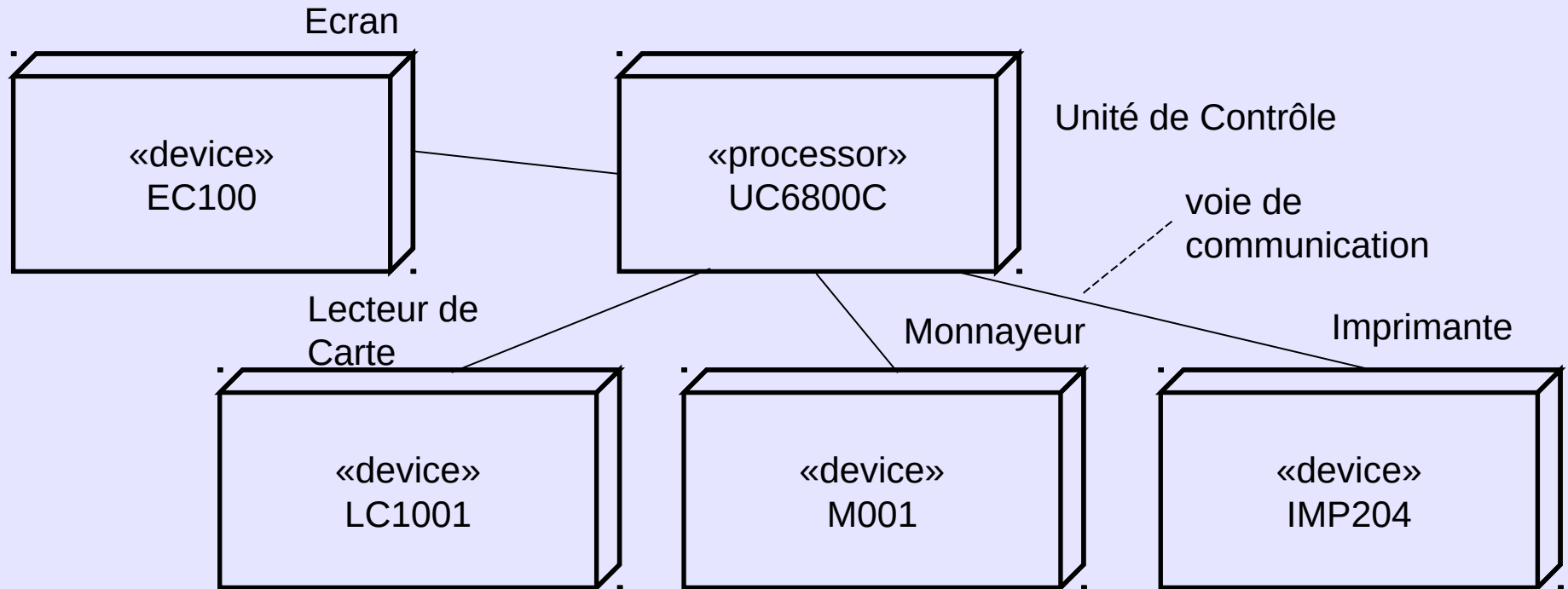
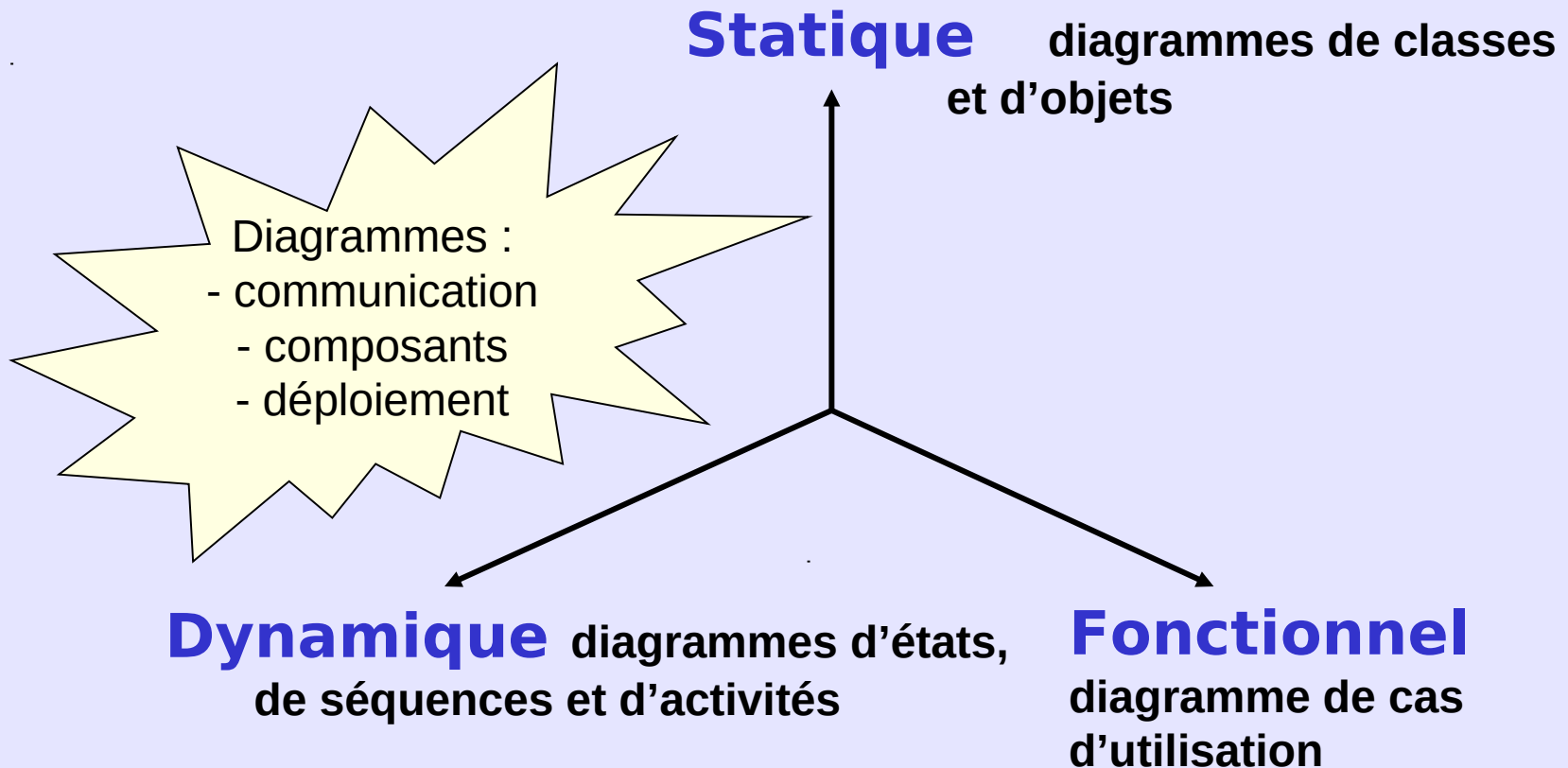
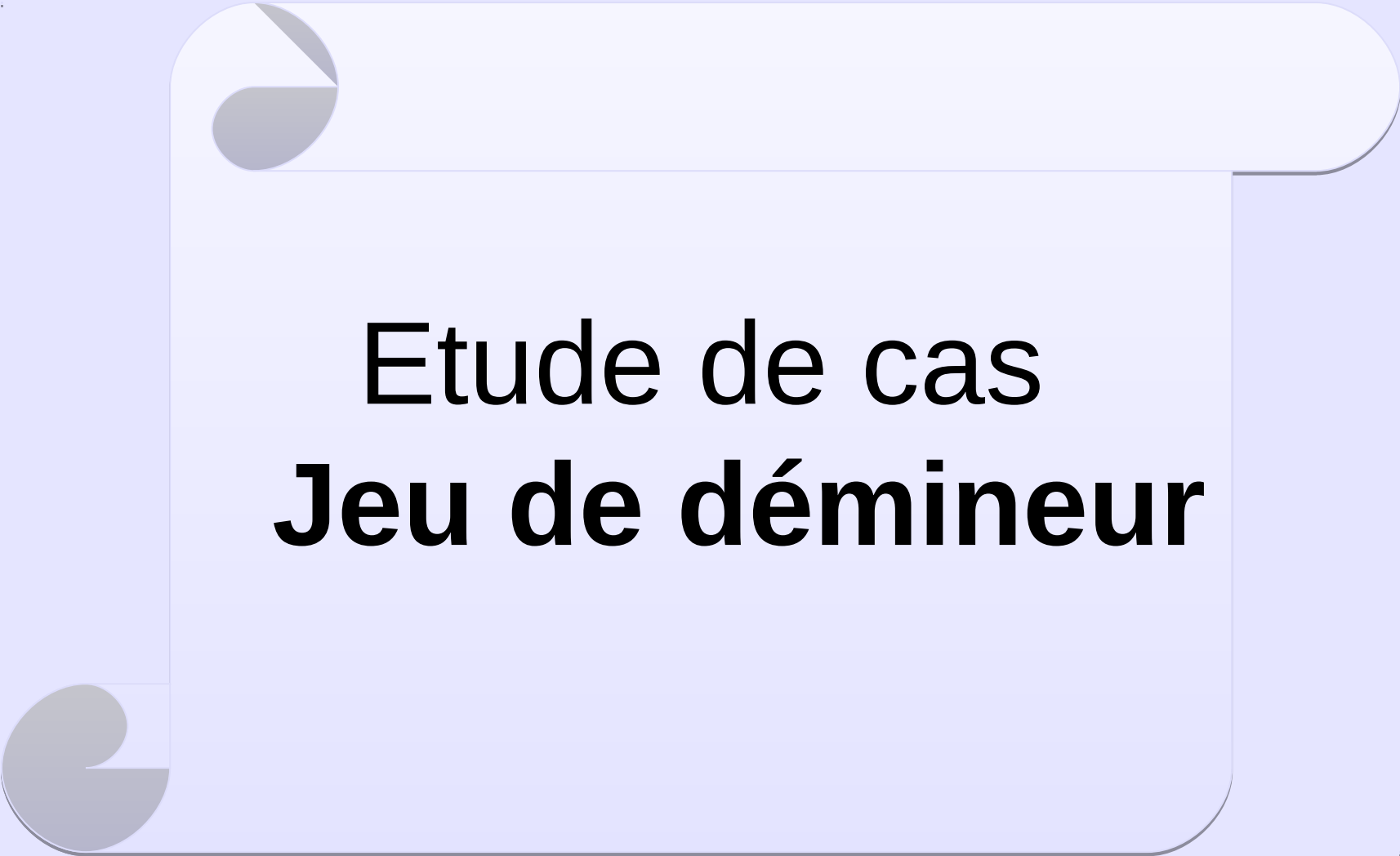


Diagramme de déploiement



UML pour la spécification






Etude de cas

Jeu de démineur


Partie de jeu de démineur

Bombs Left: **40**

 rastyle.com

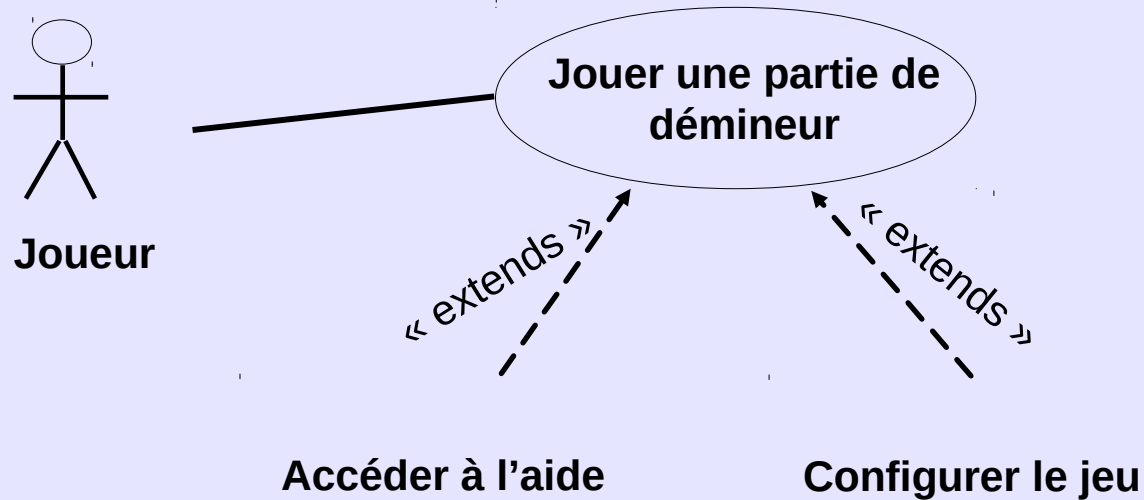
Restart
Splash

Mouse **click** opens a spot,
press SPACE to mark a spot,
hold SPACE and click to open
neighbor spots

 00

Elapsed Time:
4

Diagramme de cas d'utilisation



Description du scénario jouer une partie

Titre : joueur une partie de démineur

Résumé: jouer une partie de démineur consiste à découvrir les cases non-minées et marquer les cases minées d'un drapeau

Acteurs: joueur (acteur principal)

Description du scénario:

Préconditions :

- jeu lancé

Scénario nominal:

1. Le joueur clique sur une case
2. Le système lance le chronomètre si premier click
3. Le système identifie la case appuyée (à découvrir)
4. Le système découvre la case
5. Le système arrête le jeu et affiche le résultat (score) final
6. Si meilleur score, le système propose au joueur de saisir son nom

Extensions:

0.a : le joueur décide de reconfigurer le jeu

1. Le joueur précise le niveau de jeu
2. Le joueur redimensionne la taille de la matrice des cases.
3. Le système enregistre les nouvelles données et met à jour la configuration initiale du jeu.

Description du scénario jouer une partie

0.b: le joueur consulte l'aide de la partie

1. Le système affiche la page d'index d'aide
2. Le joueur défile les pages d'aide
3. Le joueur quitte l'aide
4. Le système continue

Alternatives:

1.b: le joueur marque une case

1. Le système marque la case choisie d'un drapeau et reste à l'étape 1

4.a: la case découverte est minée

1. le système découvre toutes les cases et déclare le joueur perdant

4.b: la case découverte est non-minée

1. le système découvre la case choisie et revient à l'étape 1.

Postconditions :

Diagramme de séquences « Système »

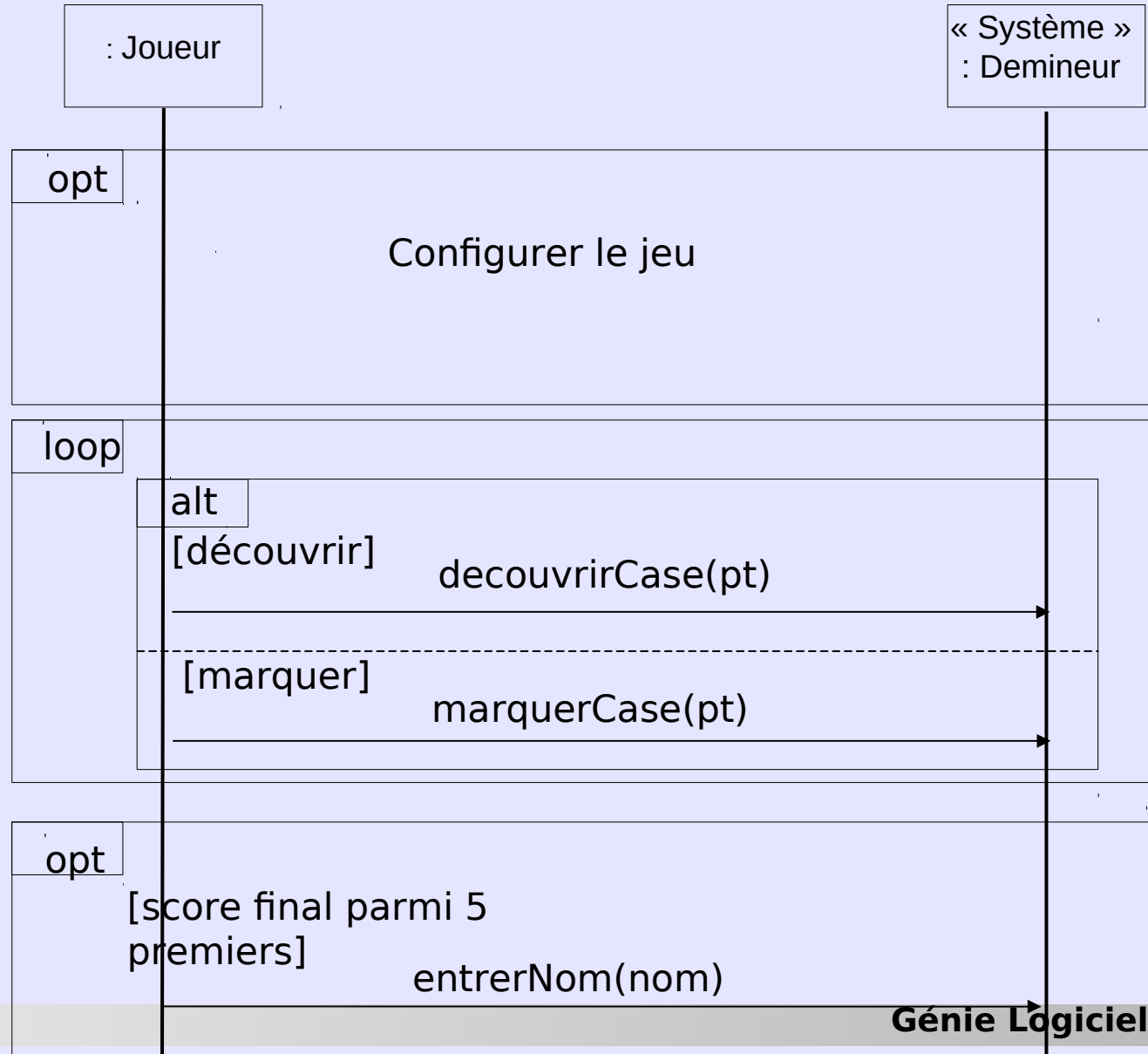


Diagramme de classes de domaine

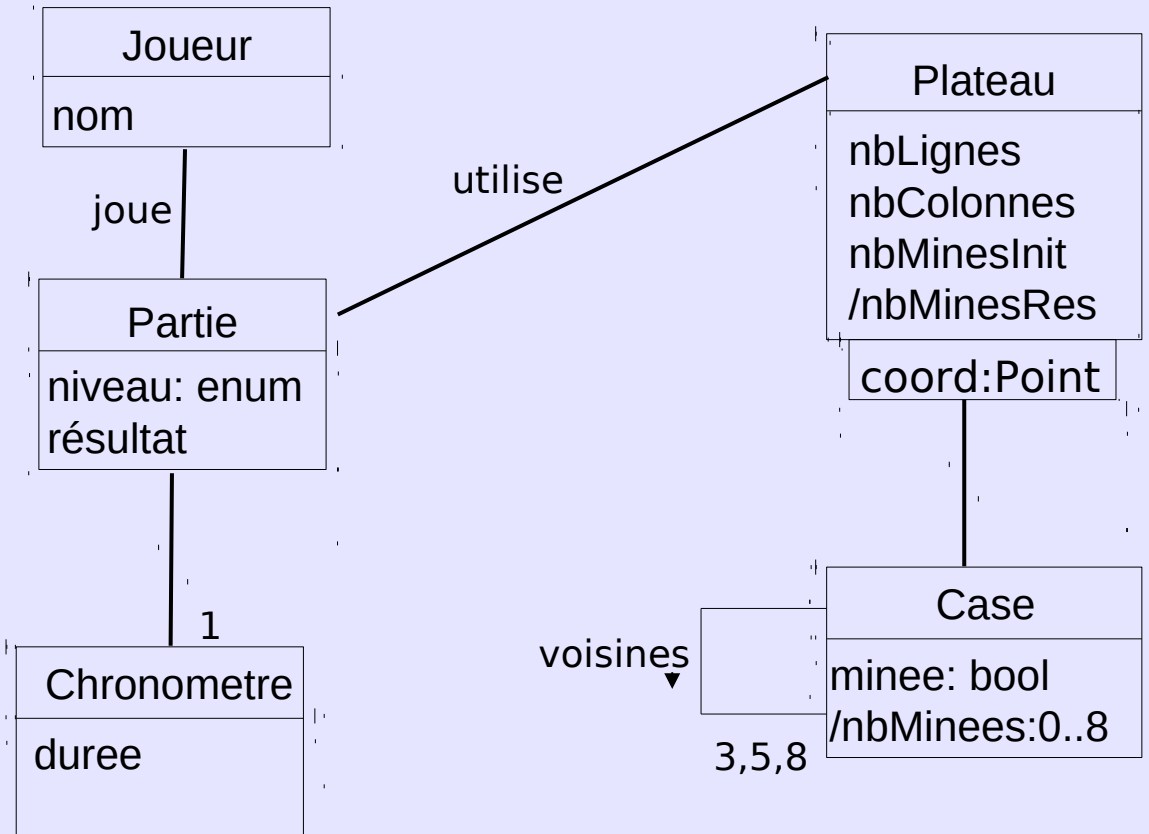


Diagramme d'état d'une case

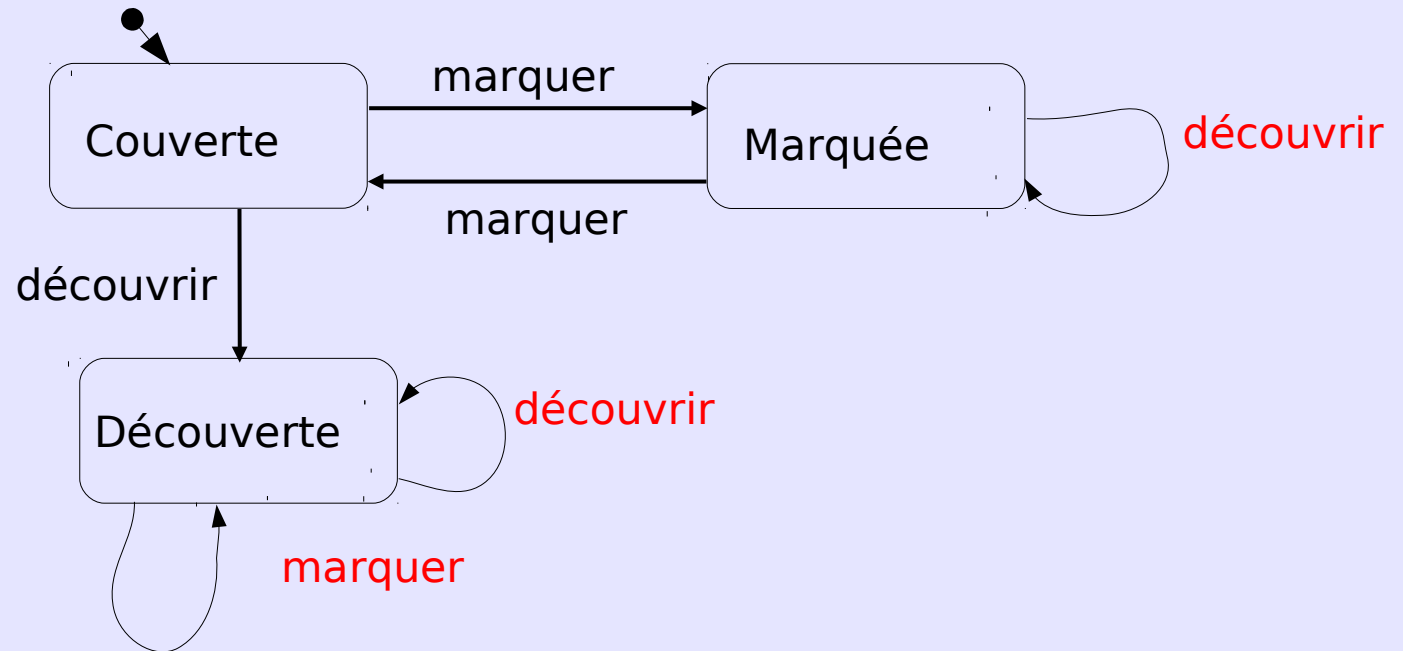


Diagramme de séquences scénario

Marquer une case

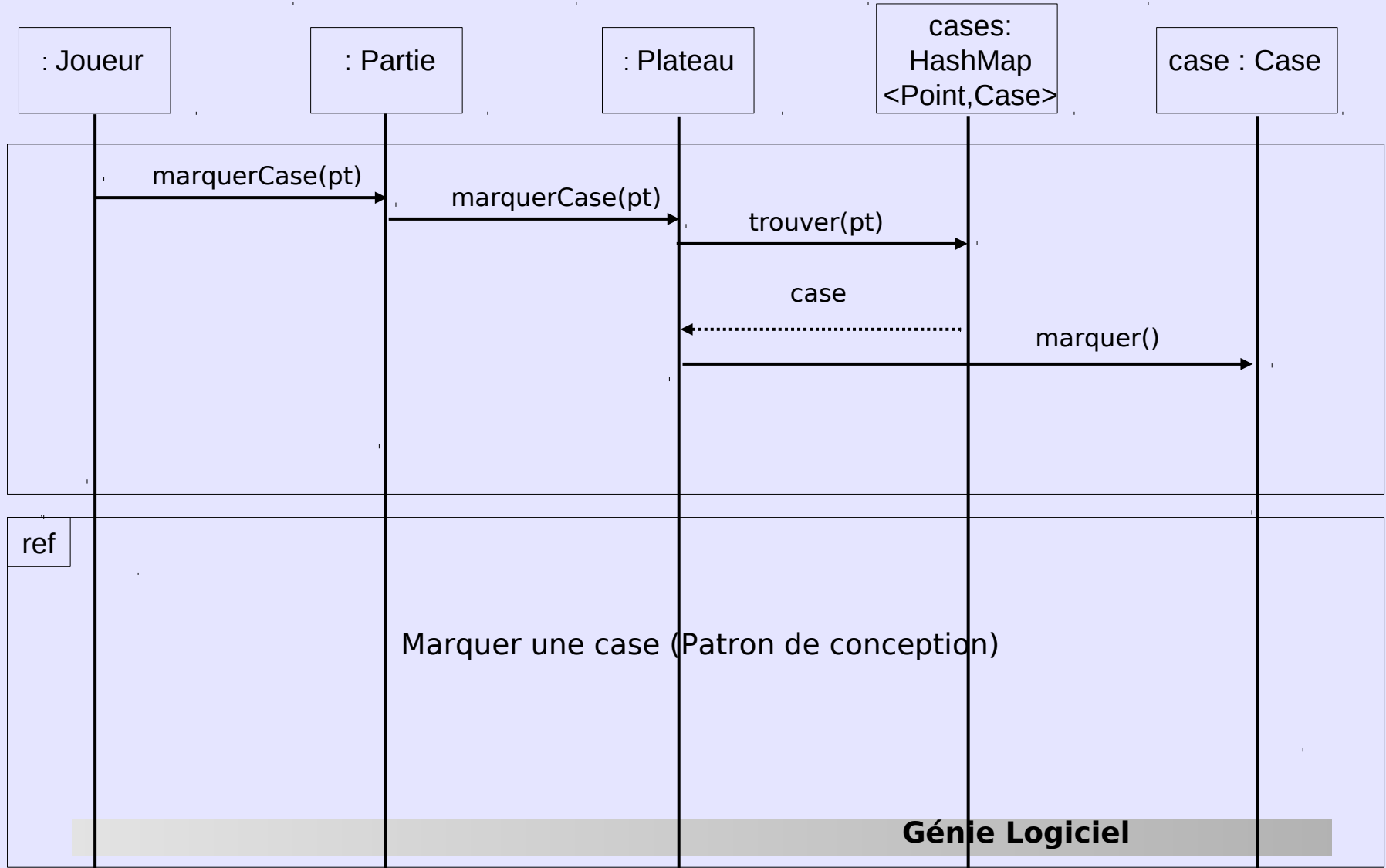


Diagramme de séquences scénario Marquer une case (Patron de conception)

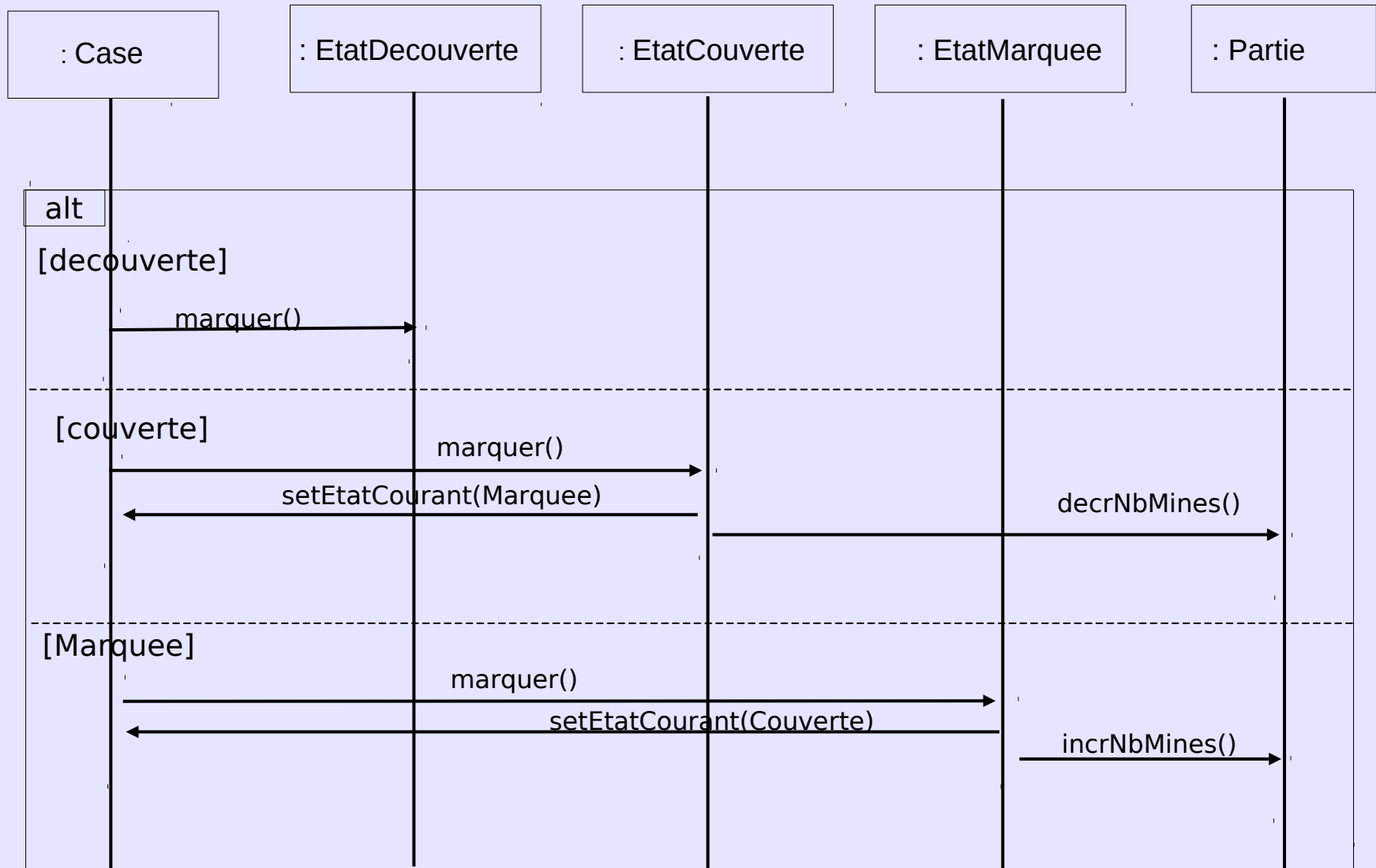


Diagramme de séquences scénario

Découvrir une case

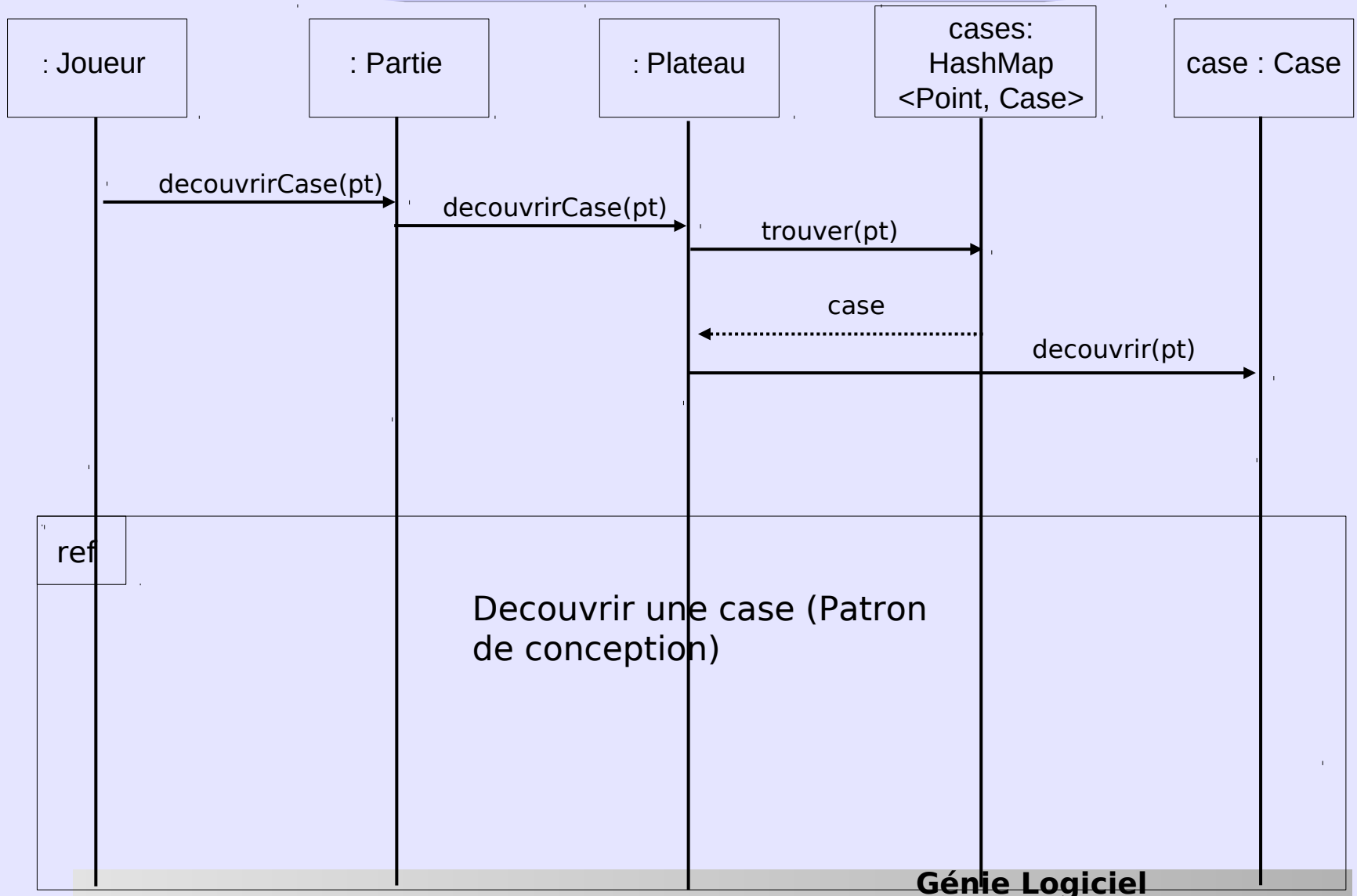


Diagramme de séquences scénario Découvrir une case (Patron de conception)

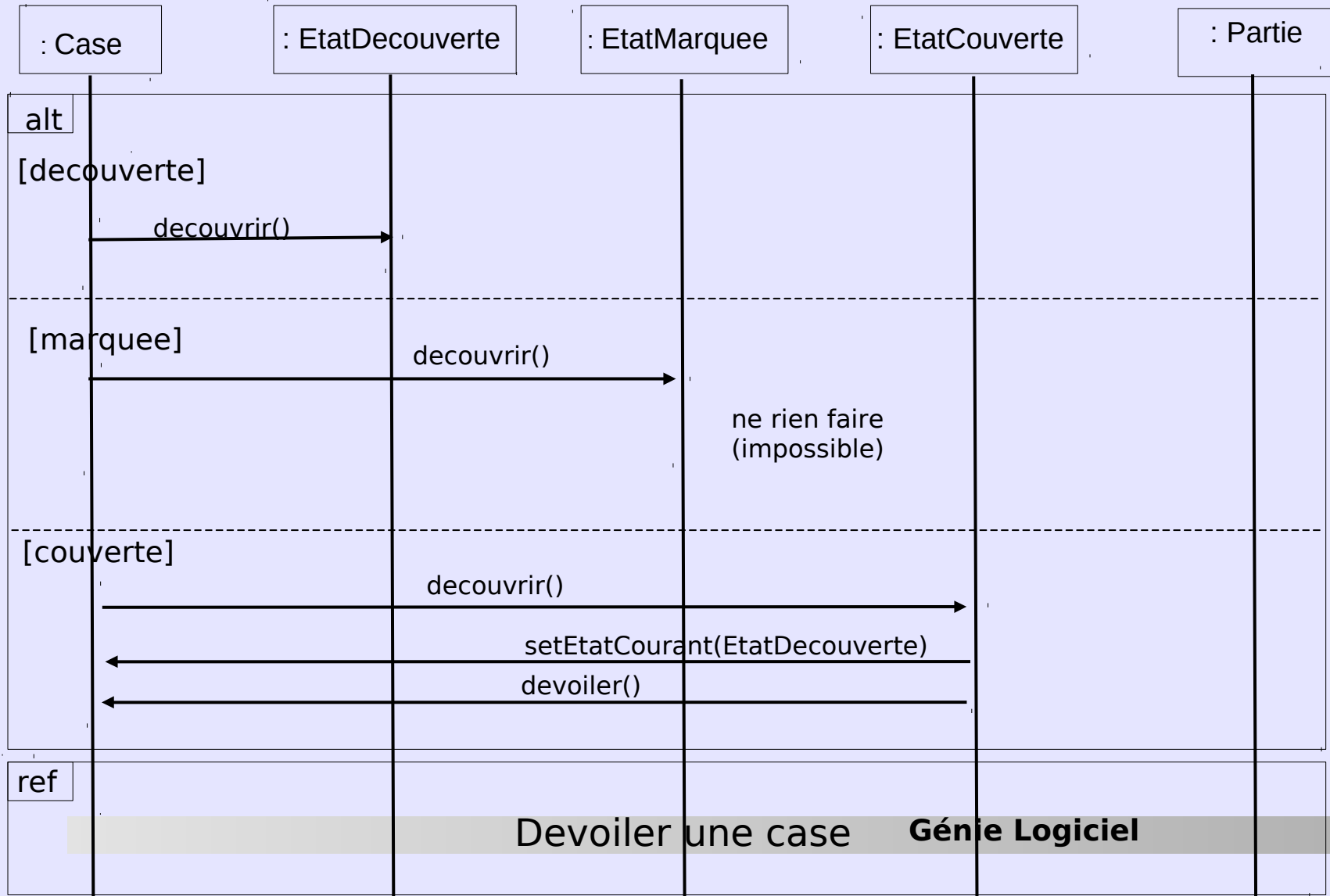


Diagramme de séquences scénario

Dévoiler une case

