ÉTS
Le génie pour l'industrie

LOG100 / GTI100
Programmation en génie logiciel et des TI
Automne 2024

Documenter la conception : les diagrammes UML

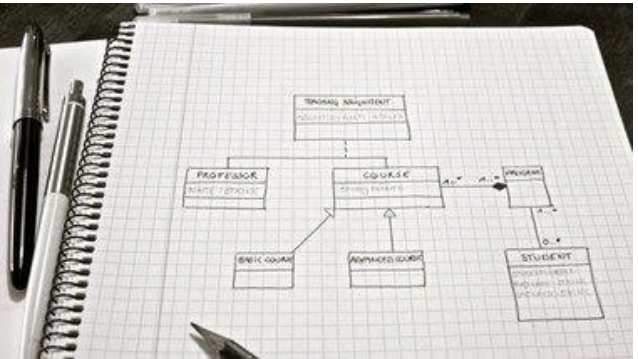
Chargé de cours : Anes Abdennebi
Crédits à: Ali Ouni, PhD

Plan

- Documenter la conception
- Le langage UML
- Diagramme de classes
- Diagramme de séquence
- Exercices

Documenter la conception

- Il est nécessaire de documenter et illustrer les décisions de conception
  - Cela permet de guider l'implémentation
  - Les illustrations graphiques facilitent la communication
  - Cela facilite aussi la maintenance



Plan

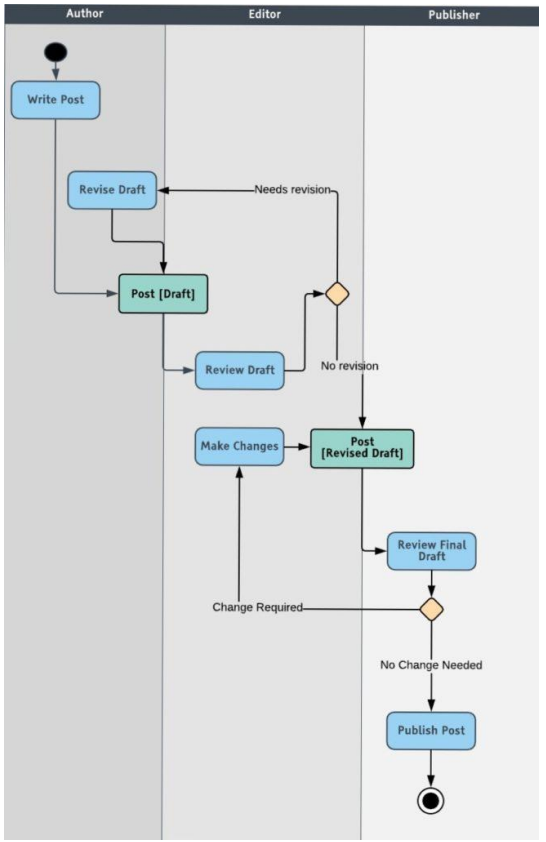
- Documenter la conception
- Le langage UML
- Diagramme de classes
- Diagramme de séquence
- Exercices

UML : Unified Modeling Language

- Standard incontournable (normalisé en 1997 par l'OMG)
  - Unification de trois approches: OOAD (Grady Booch), OMT (Jim Rumbaugh), et OOSE (Ivar Jacobson)
- Supporté par de très nombreux outils
- Indépendant des langages d'implémentation
- Toujours en évolution

UML : Unified Modeling Language

- Propose plusieurs vues complémentaires d'un système
  - Vue statique
    - Diagramme de classes
    - Diagramme de composants
  - Vue dynamique
    - Diagramme des cas d'utilisation
    - Diagramme de séquences
    - Diagramme de communications
    - Diagramme d'activités
    - Diagramme d'états
    - Etc.



Plan

- Documenter la conception
- Le langage UML
- Diagramme de classes
- Diagramme de séquence
- Exercices

Diagramme de classes

- « Il représente les classes et les interfaces d'un système ainsi que les différentes relations entre celles-ci. »

- Éléments de base : les **classes**

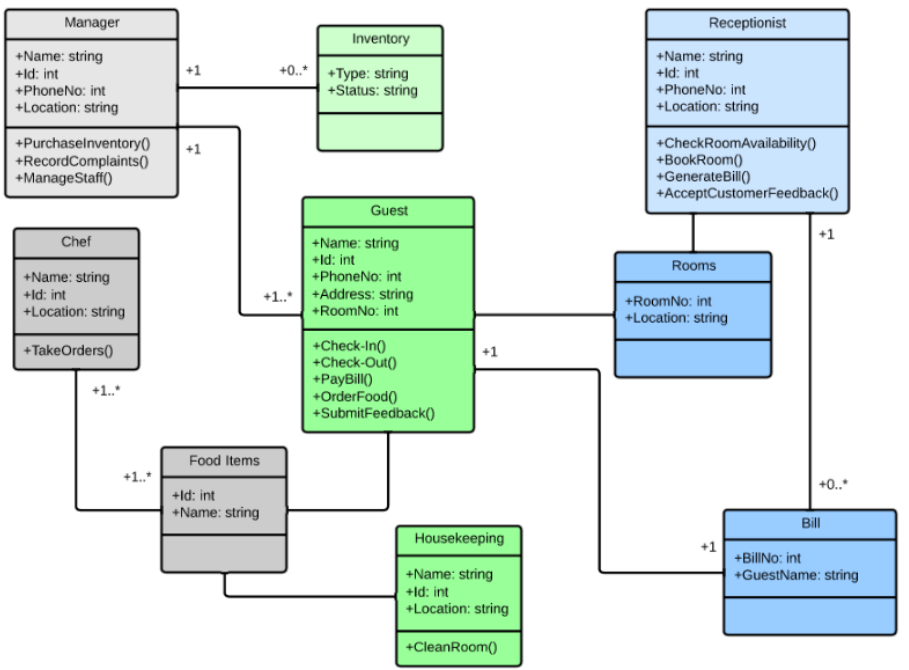


Diagramme de classes

- Une classe est représentée par un rectangle avec **trois parties**
  - 1ère partie : le nom de la classe
  - 2ième partie : les attributs
  - 3ième partie : les méthodes

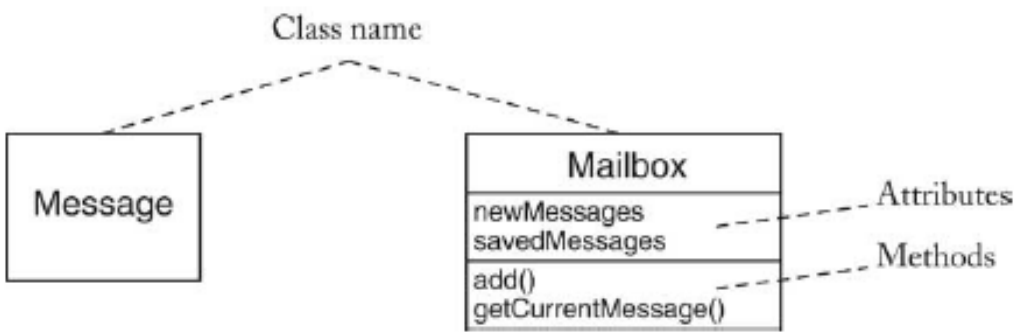


Diagramme de classes

- Parfois uniquement les méthodes et les attributs les plus importants

- Considérer le regroupement des attributs dans une autre classe quand il y en a trop
  - Regrouper les attributs numéro, rue, ville et code_postal dans une classe *Adresse*

Diagramme de classes

- Attributs d'une classe :

Visibilité nom_Attribut [multiplicité] : type_Attribut [= Initialisation]

- Visibilité : Public (+), Protected (#) ou Private (-)
- Multiplicité : le nombre de fois où cet attribut peut être utilisé au sein du même objet
  - Par exemple, pour permettre de donner deux prénoms à une instance de la classe Personne, voilà la définition de l'attribut prénom :
    - # prenom [2] : string

Diagramme de classes

- Méthodes d'une classe :

Visibilité nom_méthode ([liste_de_paramètres]) : [type_retour]

- Exemple
  - + getMessage(index : int) : Message
- Quand la méthode a des paramètres, la liste de paramètres est spécifiée comme suit :

nom_param1 : type_param1, nom_param2 : type_param2, etc.

- Une méthode peut ne rien retourner
  - Correspond à une méthode en Java avec le mot clé « void »

Diagramme de classes

- Relations entre classes :

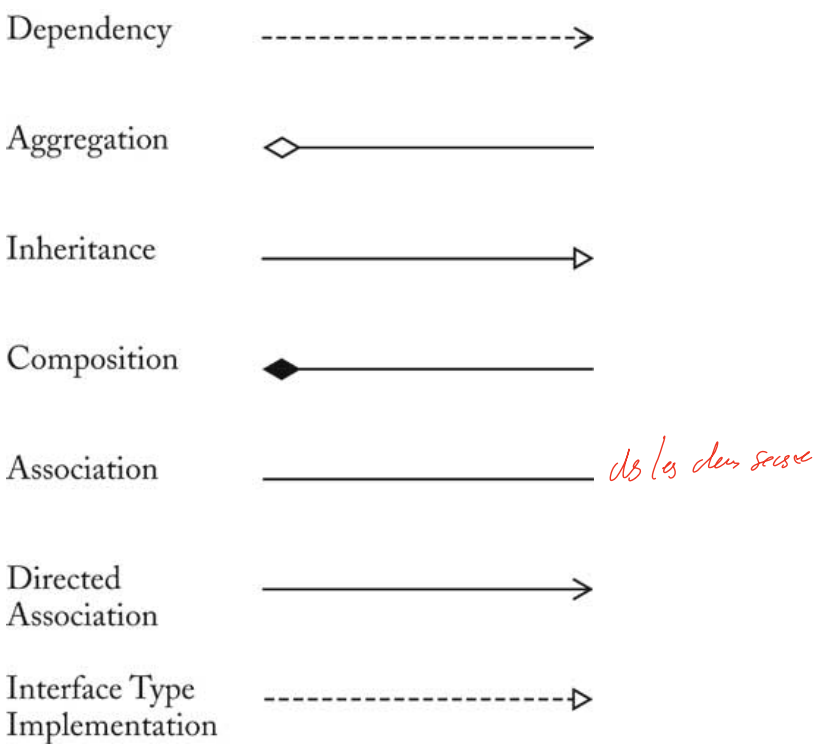
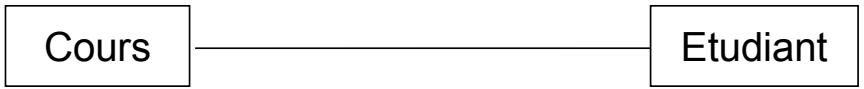


Diagramme de classes

- Association
  - Relation structurelle
    - Exemple:



  - Elle peut être nommée
  - On peut aussi nommer ses extrémités en leur associant des rôles

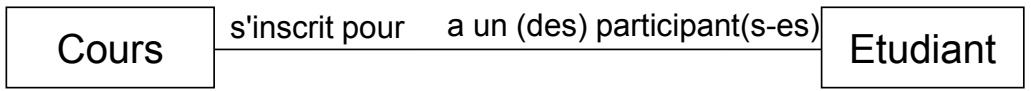
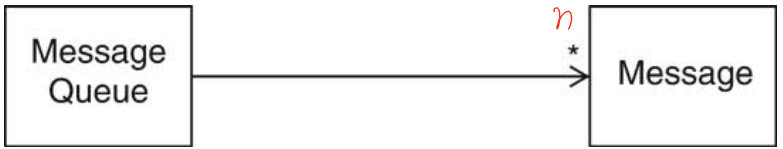


Diagramme de classes

- Association
  - Elle peut être **bidirectionnelle** ou elle peut avoir une seule direction de navigation
  - Exemple de navigation **unidirectionnelle** : la file *MessageQueue* connait l'ensemble des instances de *Message* qu'elle contient mais Message ignore tout de la file qui le contient



  - Exemple de **bidirection** : *Course* possède un ensemble d'étudiants et *Etudiant* a un ensemble de *Courss*

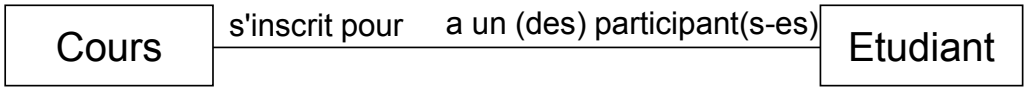


Diagramme de classes

- Relation d'agrégation
  - On spécifie les multiplicités aux deux extrémités de cette relation pour indiquer combien d'instances de l'agrégé sont contenues dans une instance de l'agrégat
    - n'importe quel nombre (zéro ou plus) : *
    - un ou plusieurs : 1..*
    - zéro ou un : 0..1
    - exactement un : 1

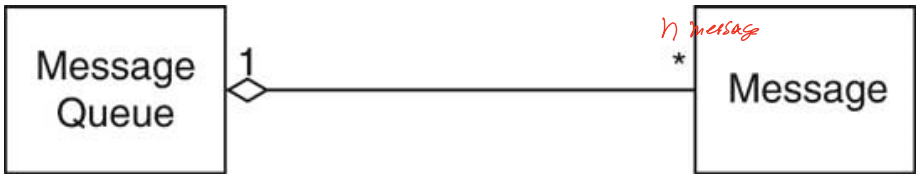


Diagramme de classes

- Relation de composition
  - C'est un cas particulier d'agrégation
  - Les objets agrégés n'existent pas en dehors du conteneur
  - Si le conteneur est supprimé, les objets agrégés sont aussi supprimés
  - Exemple: une file de messages est contenue de manière permanente dans une boîte vocale

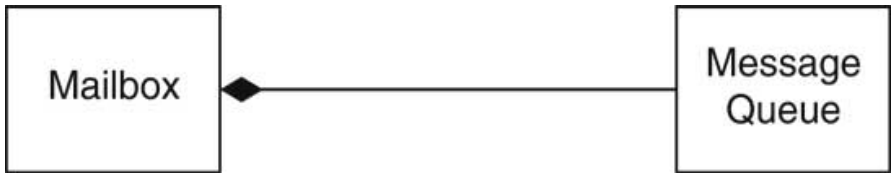


Diagramme de classes

- Relation d'implémentation
  - C'est une relation entre une classe et une **interface**
  - Une interface décrit un ensemble de méthodes sans spécifier **aucune implémentation**
  - Une classe qui implémente une interface doit implémenter **toutes les méthodes** de cette interface
  - Pour représenter une interface en UML, on ajoute le stéréotype « interface » au dessus du nom de l'interface

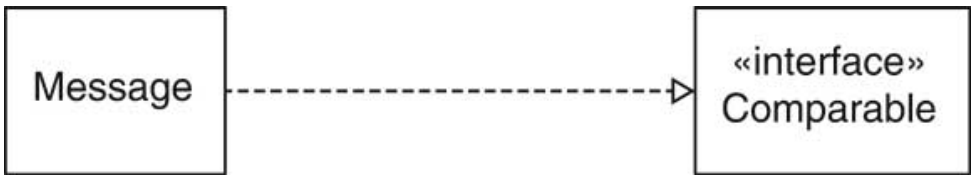


Diagramme de classes

- Relation d'héritage :

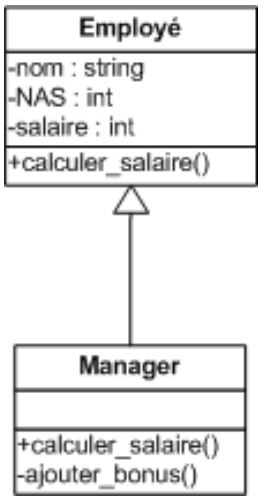


Diagramme de classes

- Relation de dépendance
  - Ce n'est pas une relation stable dans le temps
    - Une instance d'une classe peut avoir besoin d'une instance d'une autre classe de façon ponctuelle

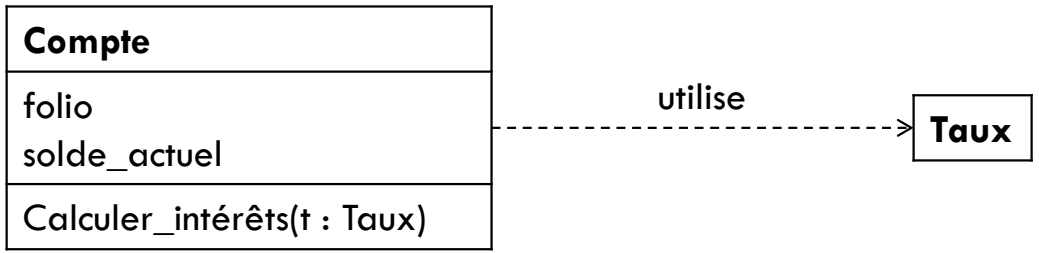


Diagramme de classes

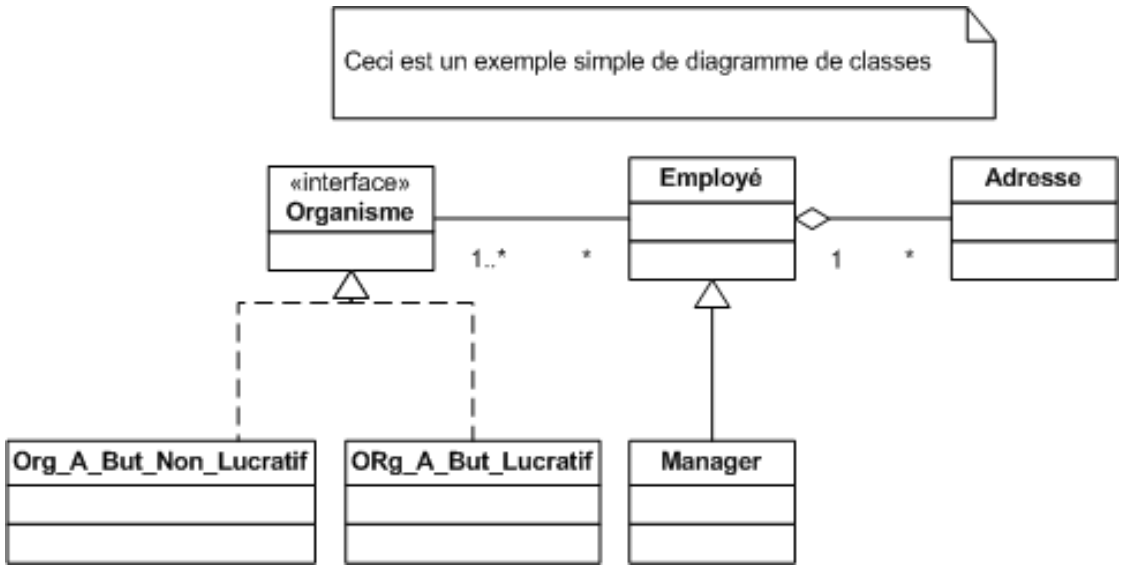


Diagramme de classes

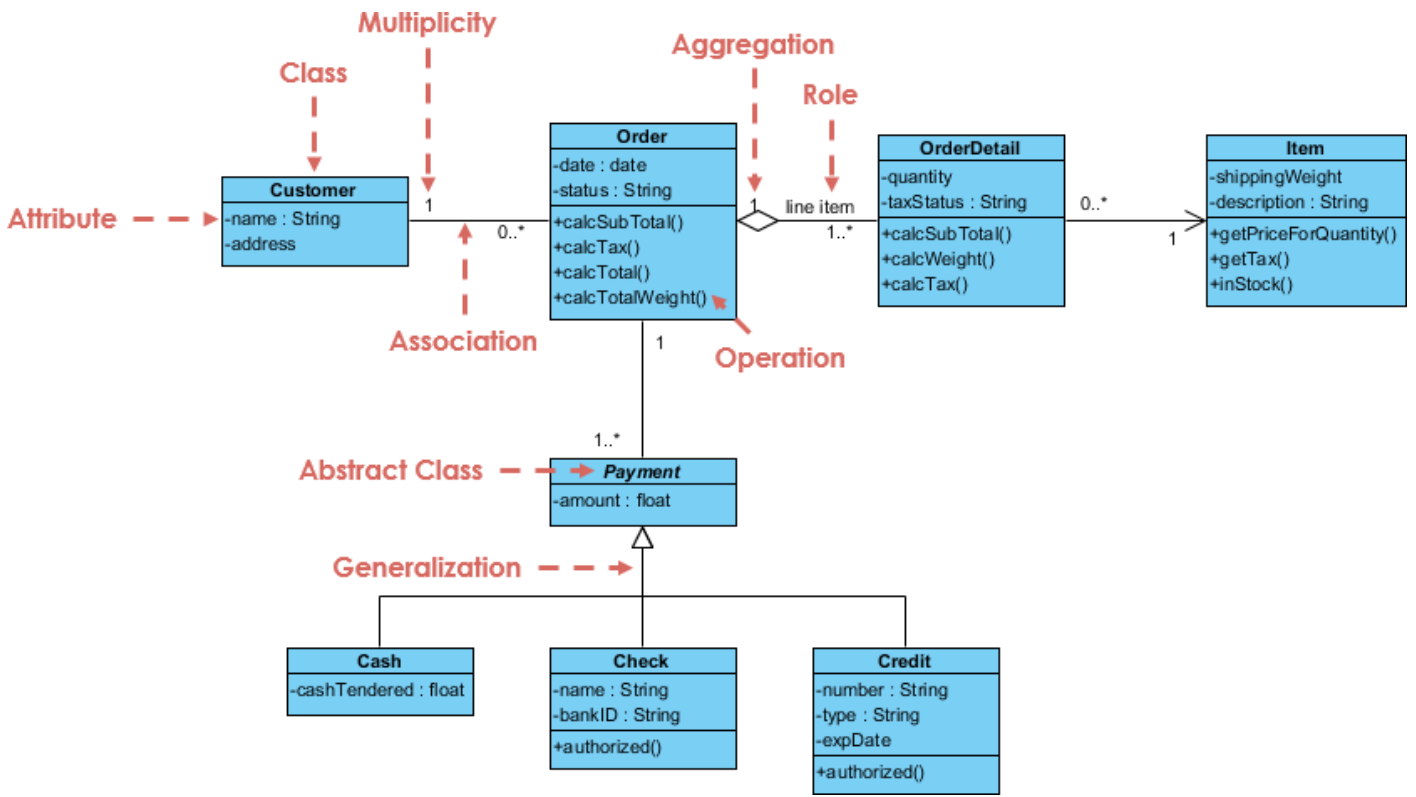


Diagramme de classes

- Quelques conseils
  - Construire des diagrammes faciles à comprendre
    - Éviter de vouloir mettre toutes les classes et les relations dans un seul diagramme
  - Construire des diagrammes qui communiquent adéquatement la conception
    - Inclure juste les éléments pertinents selon l'objectif du diagramme
  - Accompagner les diagrammes d'explications textuelles

Plan

- Documenter la conception
- Le langage UML
- Diagramme de classes
- Diagramme de séquence
- Exercices

Diagramme de séquence

- Un diagramme de séquence montre la dynamique d'un scénario
  - Montre comment se déroule le **scénario**
  - C'est un diagramme d'**objets**

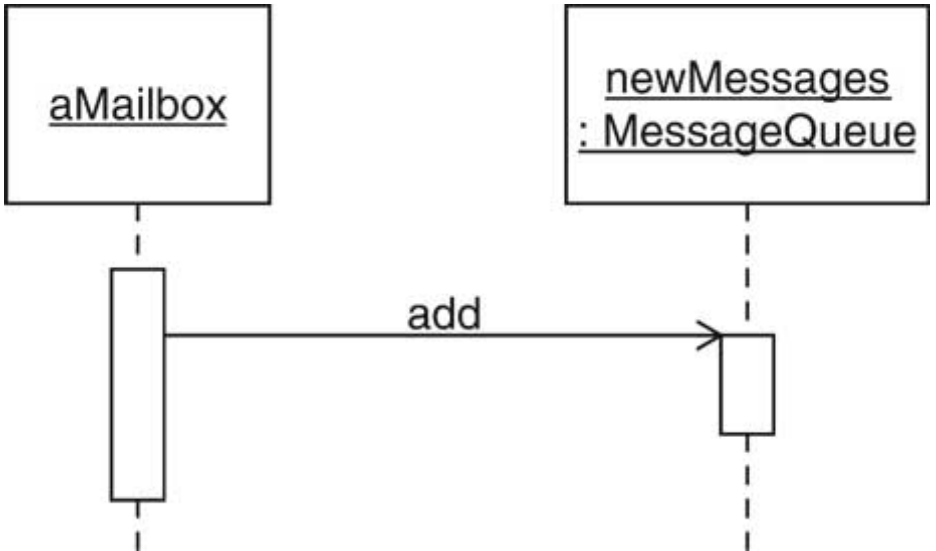


Diagramme de séquence

- Il décrit dans un ordre **chronologique** l'interaction entre les objets
- Il montre les **objets impliqués** dans un scénario
- Il montre l'**ordre d'appel** des **méthodes** de ces objets

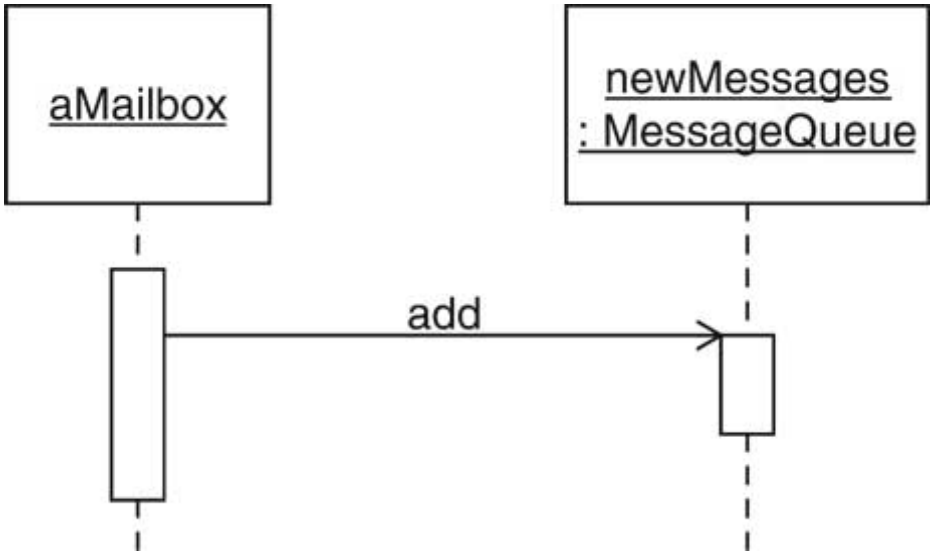
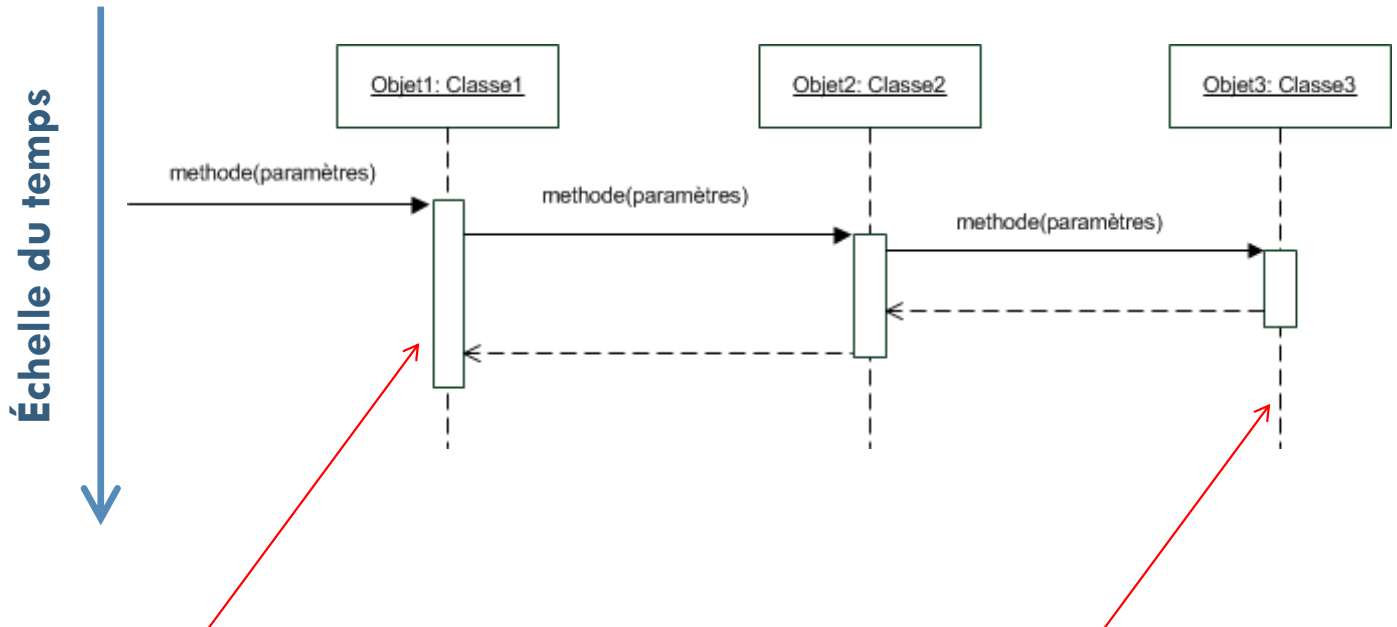


Diagramme de séquence



Barre d'**activation** : désigne un période d'activité de l'objet

Ligne de vie de l'objet

Diagramme de séquence

Nom du diagramme de séquence et ses **paramètres**

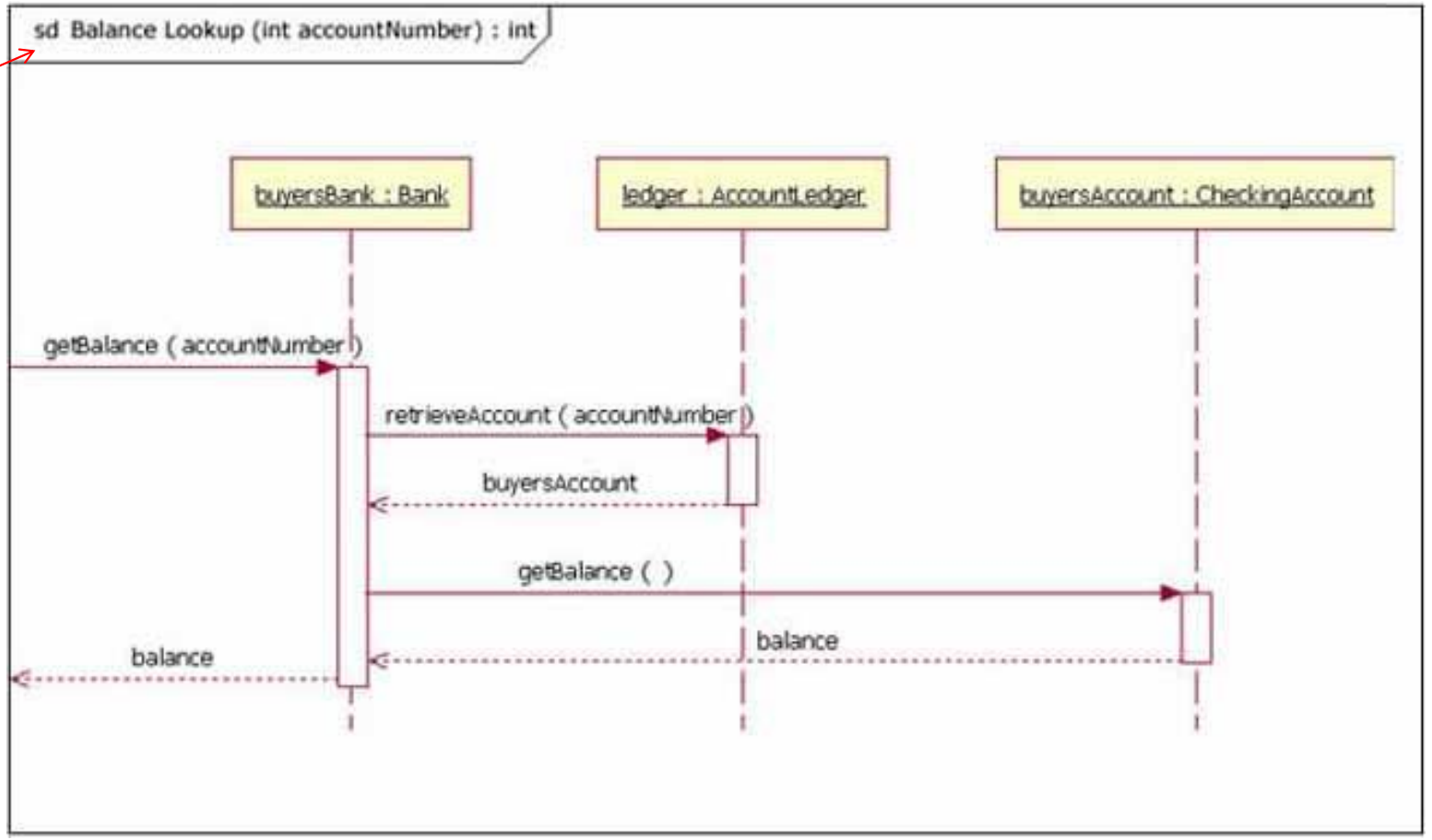


Diagramme de séquence

- Notation :

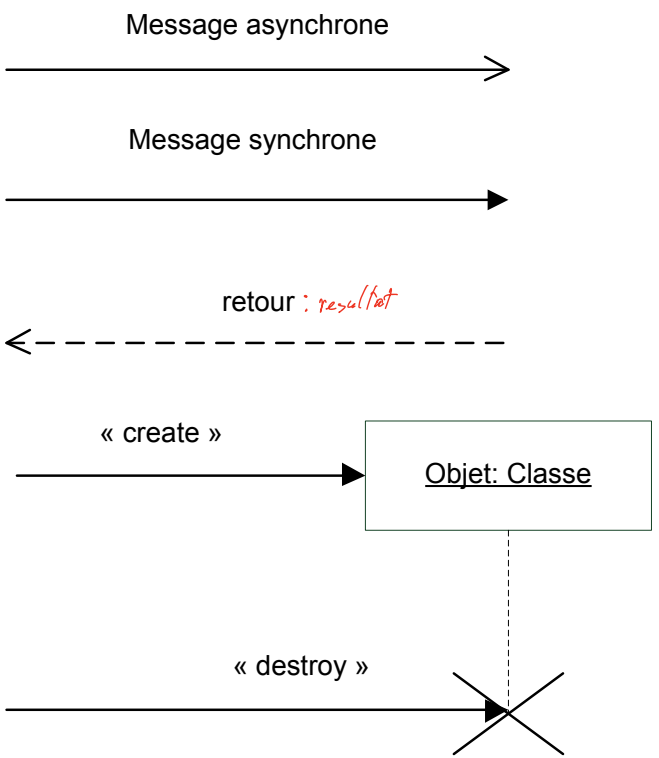


Diagramme de séquence

- Deux façons de désigner l'objet interagissant :
  - Instance **nommée** d'une classe

  - Instance **anonyme** d'une classe

Diagramme de séquence

- Deux façons d'illustrer le retour d'un message :

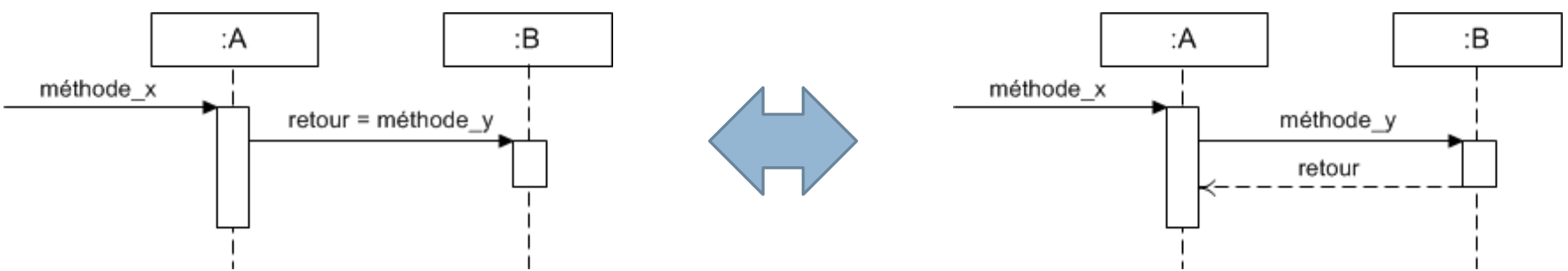


Diagramme de séquence

- Une méthode peut créer un objet :

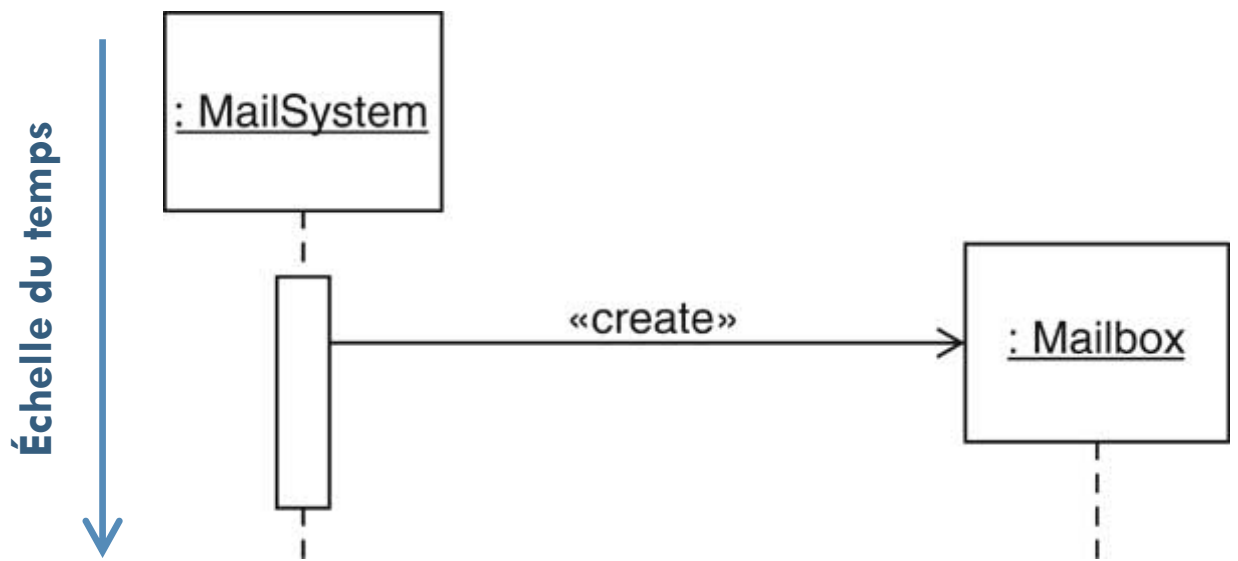


Diagramme de séquence

- Une méthode peut appeler une autre méthode du même objet

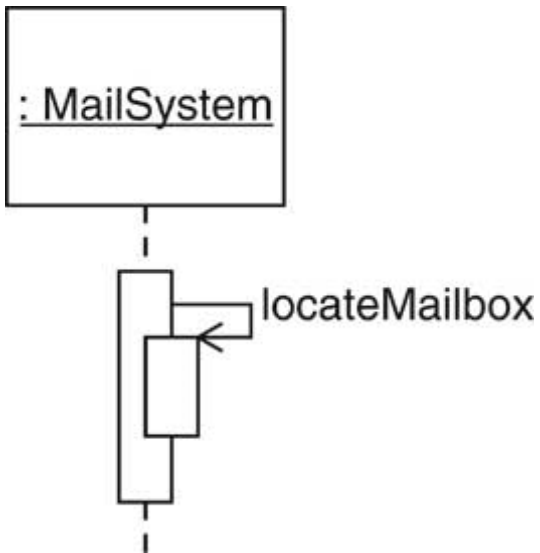


Diagramme de séquence

- UML offre d'autres notations qui permettent de :
  - représenter des boucles et des alternatives
  - faire appel à un autre diagramme de séquence pour des raisons de lisibilité et compréhension
  - représenter des fragments strictement séquentiels ou parallèles

Diagramme de séquence

- Notation des alternatives :

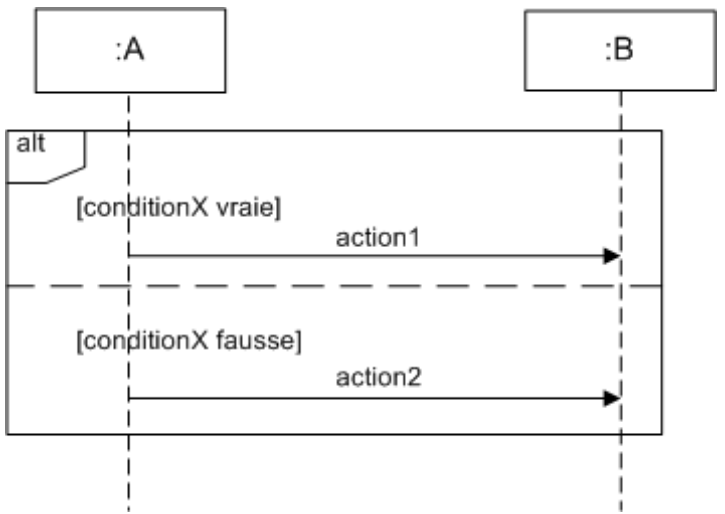


Diagramme de séquence

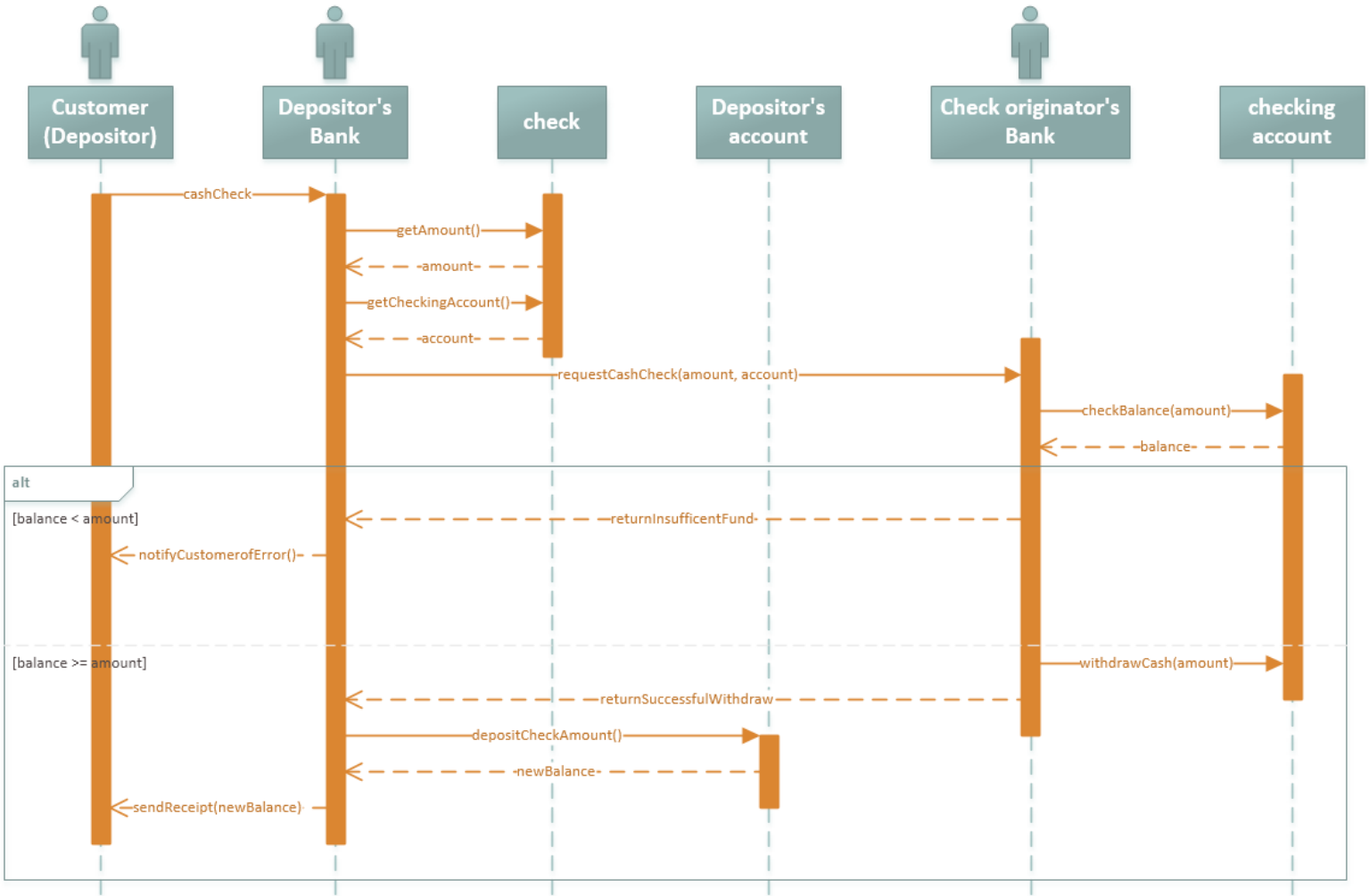


Diagramme de séquence

- Notation des alternatives :

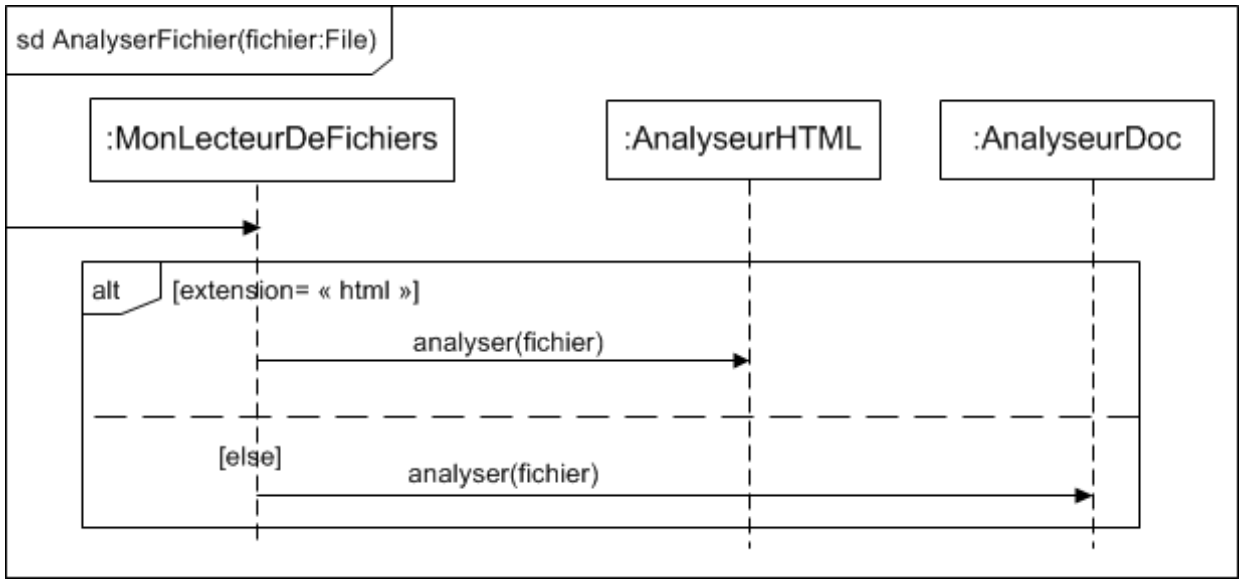


Diagramme de séquence

- Notation des boucles :

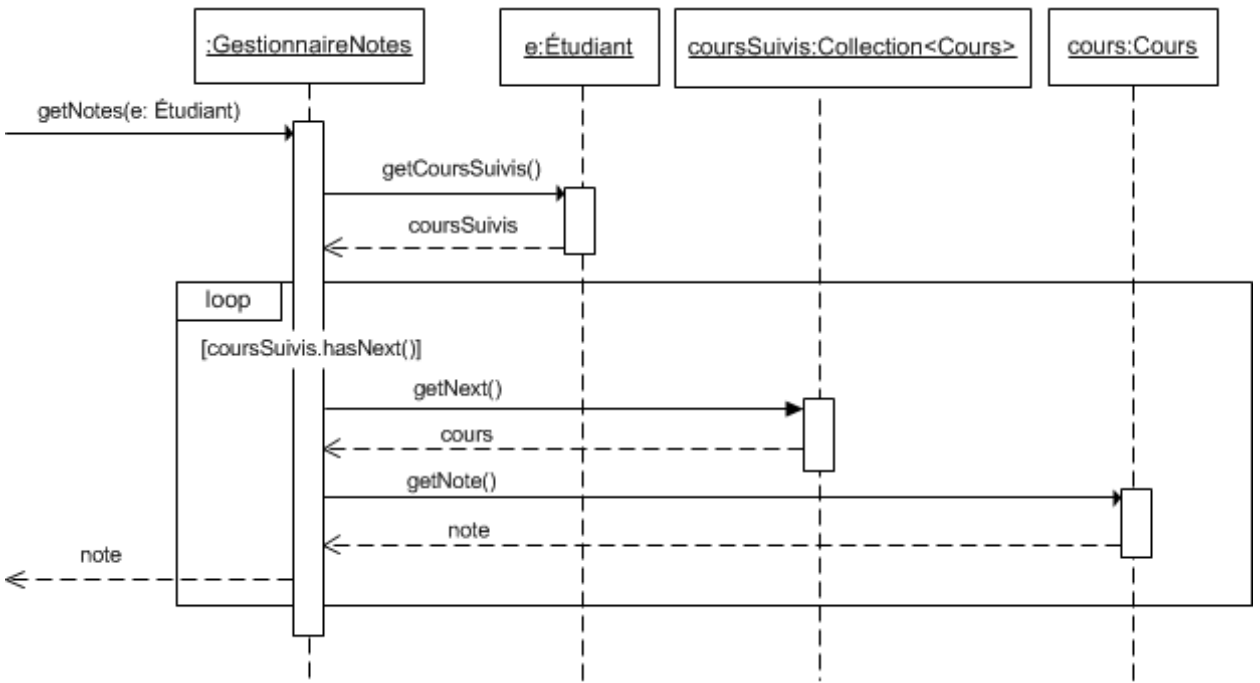
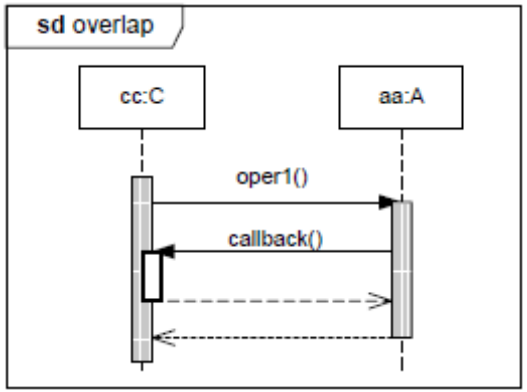


Diagramme de séquence

- Exemple d'un appel chevauchant (*callback*) :



Plan

- Documenter la conception
- Le langage UML
- Diagramme de classes
- Diagramme de séquence
- Exercices

Diagramme de classes

Une **cabine téléphonique** contient un **téléphone** composé de 9 **touches** pouvant être **enfoncées** par l'**utilisateur**. Le téléphone peut être utilisé par un utilisateur, lequel doit réaliser plusieurs opérations avec le dispositif : le **décrocher, tenter une connexion** avec un **numéro**, et **communiquer** un **message** si la personne répond.

Diagramme de classes

10 joueurs de hockey utilisent une patinoire sur laquelle se trouvent deux buts et une rondelle, à des positions précises, pouvant être soit : tirée au but, passée à un autre joueur, volée d'un autre joueur.

Un joueur a un nom, un numéro, un bâton, 4 coéquipiers, 5 adversaires et, occasionnellement, une rondelle. Un autre joueur peut tenter de le plaquer, de lui voler la rondelle (s'il la possède) et de lui passer la rondelle.

Le joueur utilise son bâton pour déplacer la rondelle à une certaine vitesse (en m/s) et selon un angle (en degré). Les joueurs sont bien conscients de la position des 2 buts. Un but peut contenir une rondelle ou non.

Diagramme de séquence

Dessinez le diagramme de séquence correspondant à l'appel de la méthode *gerant.recruter("Dubois")* dans le code suivant :

```
public class Gerant {
  private Compagnie maCompagnie = new Compagnie("Ma compagnie");
  public void recruter(String nom){
    Employee nouveau = new Employee(nom);
    maCompagnie.addEmployee(nouveau);
    System.out.println("Un nouvel employée a été recruté");
  }
}
```

Diagramme de séquence

Dessinez le diagramme de séquence correspondant à l'appel de la méthode *main()* suivante :

```
public class MyClass {
  public void method1() {
    String chaine = "allo";
    Integer lg = new Integer(chaine.length());
    System.out.print(chaine + " " + lg);
  }
  public static void main(String[] args) {
    MyClass maClasse = new MyClass();
    maClasse.method1();
  }
}
```

Diagramme de séquence

Dessinez le diagramme de séquence correspondant à l'appel de la méthode *main()* suivante :

```
public class MyClass {
  public void method2() {
    String chaine = "allo";
    for (int i = 0; i < chaine.length(); i++) {
      System.out.print(chaine.charAt(i));
    }
  }
  public static void main(String[] args) {
    MyClass maClasse = new MyClass();
    maClasse.method2();
  }
}
```