# CHAPITRE 1 – INTRODUCTION

## Caractéristiques d’un modèle orienté objet

1. Éléments majeurs

* Abstraction
* Encapsulation
* Héritage
* Polymorphisme

1. Intervenants

* Utilisateurs
* Clients
* Analystes
* Concepteurs
* Programmeurs

1. Points de vue architecturaux

* Logique : Décomposition orientée objets
* Processus : Décomposition de l’exécution
* Implantation : Décomposition statique des modules et sous-systèmes
* Déploiement : Relation matériel-logiciel
* Scénario : Comportement du système pour les utilisateurs

# CHAPITRE 2 – FONCTIONNALITÉ & CAS D’UTILISATION

## Cas d’utilisation

• UNE DESCRIPTION RELATIVEMENT LONGUE

• ALLANT DU DÉBUT À LA FIN D’UN PROCESSUS

• COMPORTANT PLUSIEURS ÉTAPES OU TRANSACTIONS.

1. Cas d’utilisation

* Capturer le comportement désiré du système
* Spécifier ce que le système fait, mais pas comment il le fait
* Document narratif qui décrit la séquence d’évènements dans laquelle un acteur utilise un système pour accomplir un processus

1. Format de haut niveau

|  |  |
| --- | --- |
| Cas d’utilisation |  |
| Acteur |  |
| Type |  |
| Description |  |

1. Cas d’utilisation étendu

|  |  |
| --- | --- |
| Cas d’utilisation |  |
| Système |  |
| Niveau |  |
| Acteur primaire |  |
| Partie prenante |  |
| Précondition |  |
| Garanties en cas de succès |  |
|  |  |

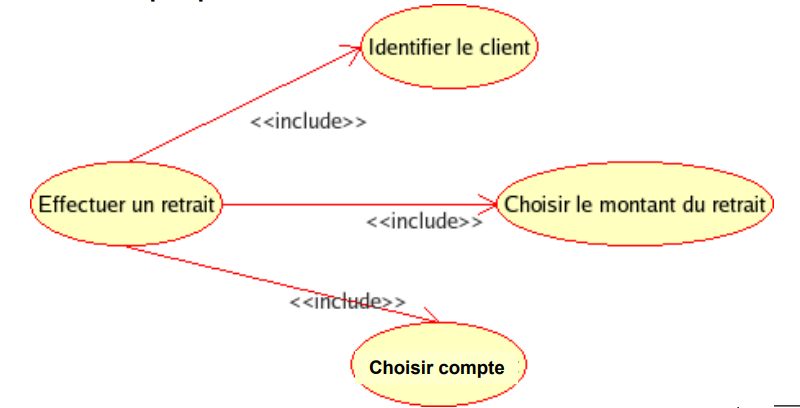
1. Scénario en colonne

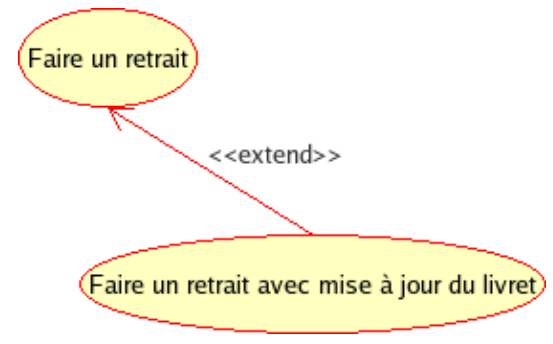
|  |  |
| --- | --- |
| CLIENT | SERVEUR |
| 1. blablablabla |  |
| 1. HOHOHOHO |  |
|  | 1. HEHEHEHHEHE |
|  | 1. HATE THIS COURSE |
|  |  |

## Diagramme de contexte

* Permet de visualiser les cas d’utilisations primaires
* Définit les limites du système modélisés, les principaux acteurs du modèle et les cas d’utilisation primaires

1. Relations entre les cas d’utilisations
2. «include » : Plus commune & plus utile, pour sous-fonctions qui peuvent être reutilisés
3. « extends » : Ajouter des étapes à un cas existants sans modifier le cas original



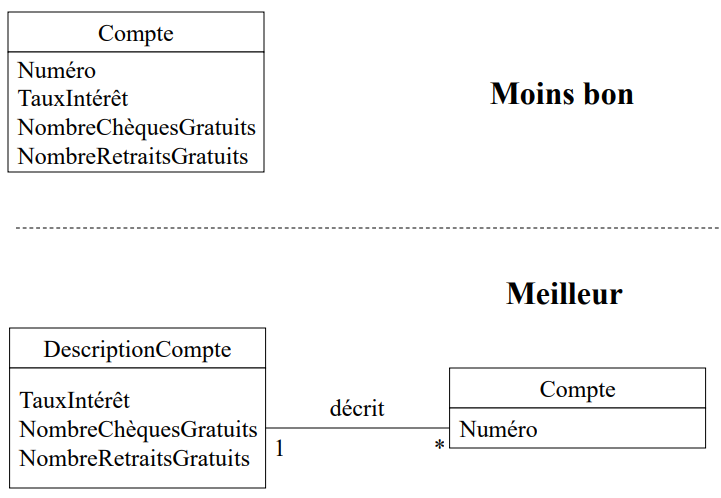


# CHAPITRE 3 – CONSTRUNCTION D’UN MODÈLE CONCEPTIONNEL

* ANALYSE ORIENTÉE OBJET
* L’ANALYSE ORIENTÉE OBJET EST LA DÉCOMPOSITION DU PROBLÈME EN CONCEPTS INDIVIDUELS
* LE MODÈLE CONCEPTUEL EST UNE REPRÉSENTATION DES ÉLÉMENTS DU MONDE RÉEL, PAS DE COMPOSANTES LOGICIELLES
* IL PERMET DE CLARIFIER LA TERMINOLOGIE & LE VOCABULAIRE

## Modèle conceptuel

1. Pour construire un modèle conceptuel
2. Faire la liste des concepts candidats
3. Les rassembler dans un diagramme de conceptuel
4. Ajouter les associations nécessaires pour conserver les relations qui méritent d’être mémorisée
5. Ajouter les attributs nécessaires pour remplir les requis informationnels



## Concepts, classes & types

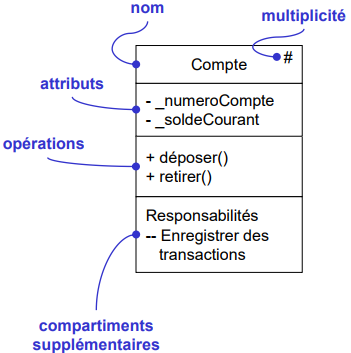
– Un concept: la référence à un élément du monde réel, issu du domaine du problème. Le terme concept n’est pas utilisé ni défini en UML

– Une classe: l’implémentation logicielle d’un concept, avec des attributs et des méthodes. Le terme classe est défini en UML

– Un type: similaire à une classe, mais sans les méthodes. Un type, en UML, est indépendant du langage.

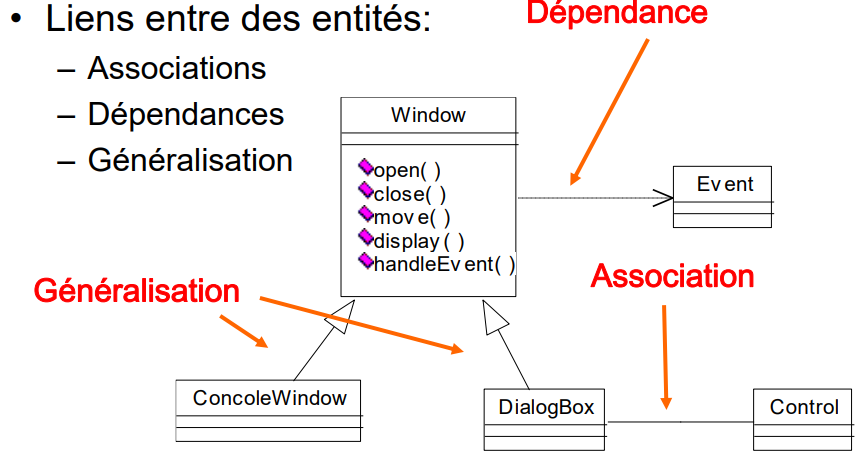
1. Classe

Une classe est la description d'une collection d'objets qui partagent les mêmes attributs, les mêmes opérations, les mêmes relations et la même sémantique.



1. Association

* Relation entre des concepts qui indique une connexion sémantique
* Une association peut être décorée :

– Nom : décrit la nature de la relation.

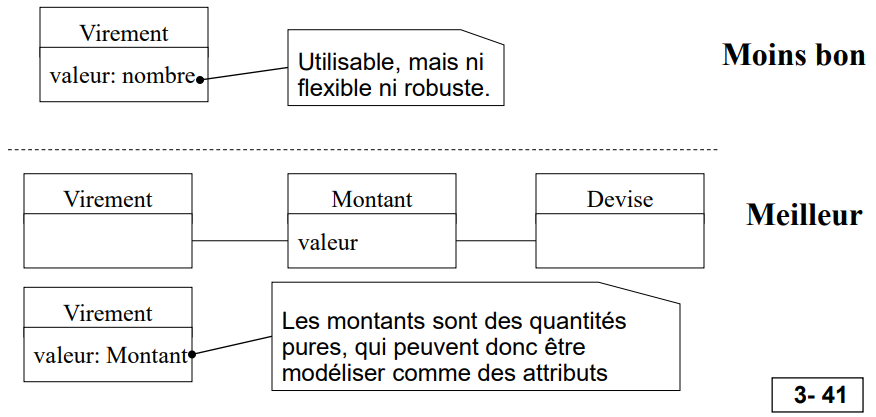
– Rôle : nomme la façon dont une classe participe à la relation.

– Multiplicité : indique le nombre d’objets participants.

– Direction : indique le sens de parcours de l’association.

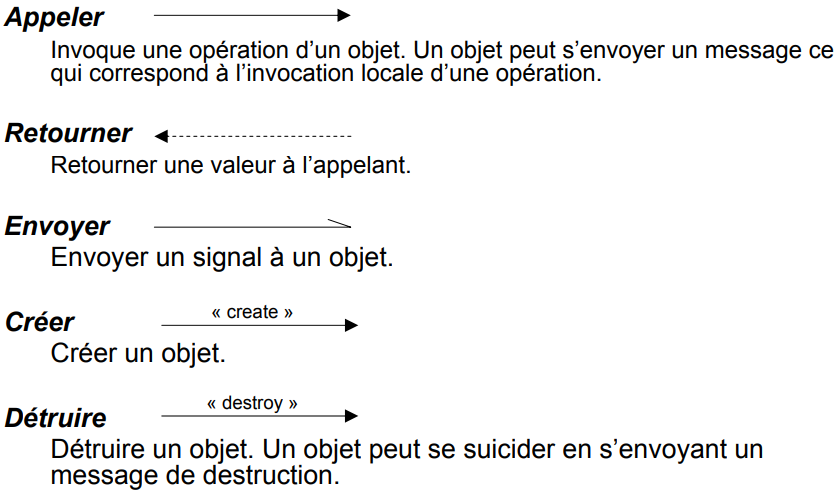
1. Attributs

* TOUTES LES PREMIERES LETTRESSDE MOTS DOIVENT DÉBUTER PAR UNE MAJUSCULE AUF LE PREMIER
* Modélisation de quantité et d’unités : Bien que le montant d’une transaction, par exemple, puisse être représenté avec un nombre, il est plus robuste et flexible de le modéliser à l’aide d’un concept séparé pour pouvoir tenir compte des unités:

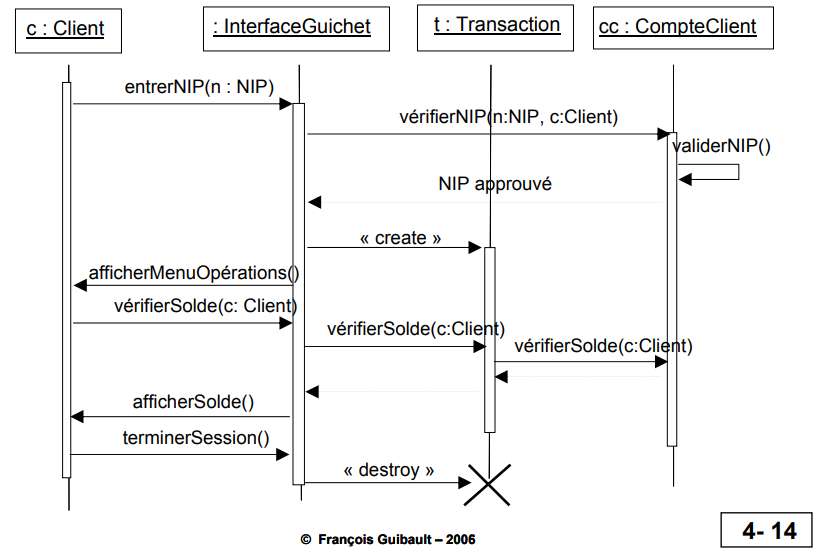


# CHAPITRE 4 – DIAGRAMMES D’INTERACTION

* Interaction : Une description du comportement incluant un ensemble de messages échangés entre un ensemble d’objets à l’intérieur d’un contexte afin d’accomplir un but.



## Diagramme de séquence

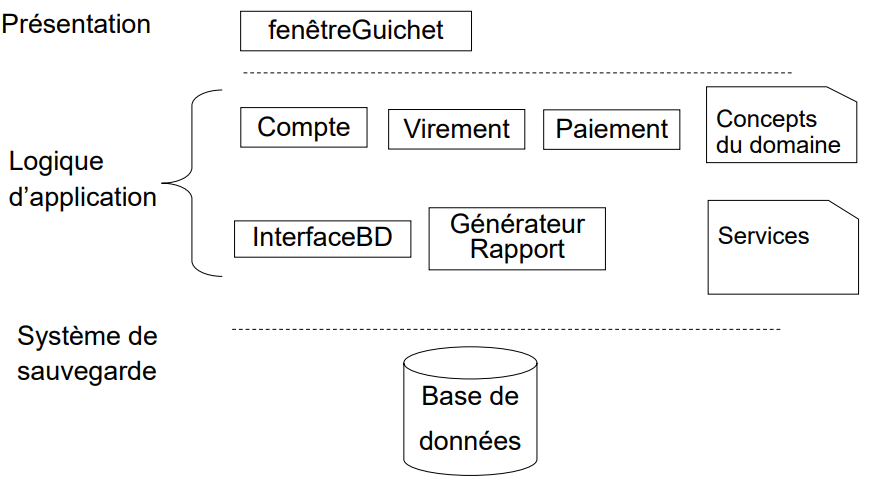




# CHAPITRE 5 – ARCHITECTURE

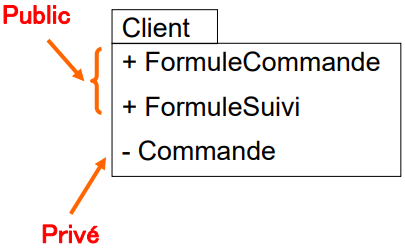
## Modèle Architectural

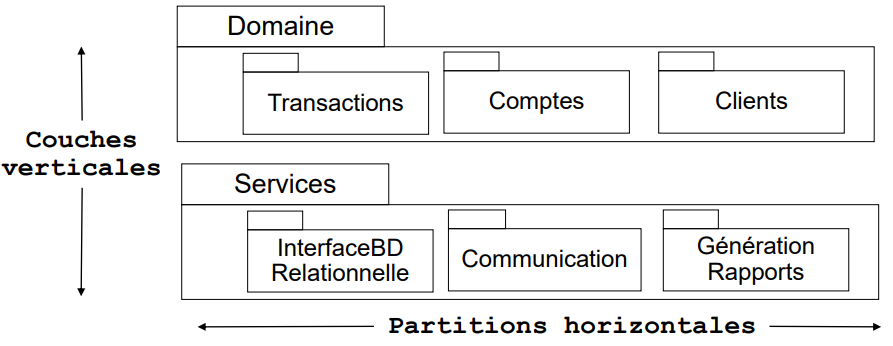
* L’architecture logicielle d’un système est sa décomposition en un certain nombre de sous-systèmes.
* Architecture multi Niveaux : Présentation, Logique d’application, Système de sauvegarde



## Diagramme de paquetages

* Utile pour regrouper les éléments communs en paquetages afin de pouvoir les comprendre
* Chaque paquetage doit avoir un nom unique
* Chaque élément d’un paquetage doit avoir un nom unique

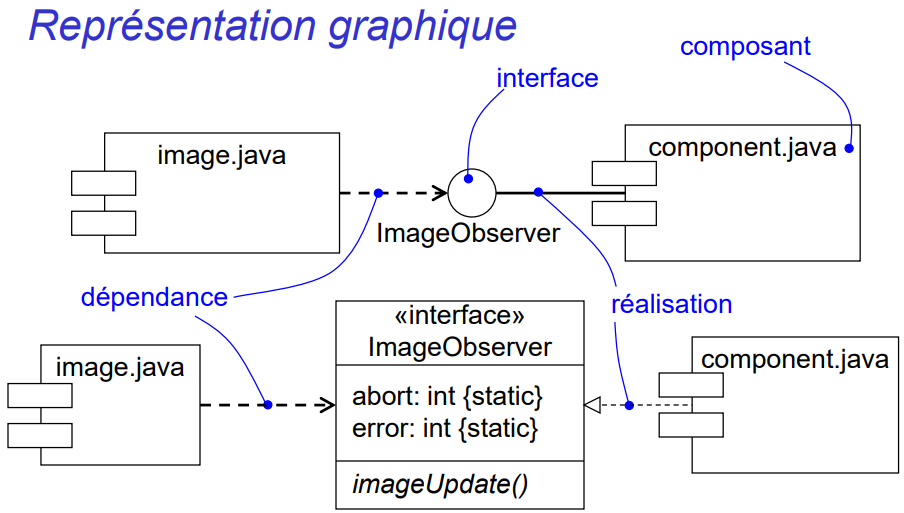




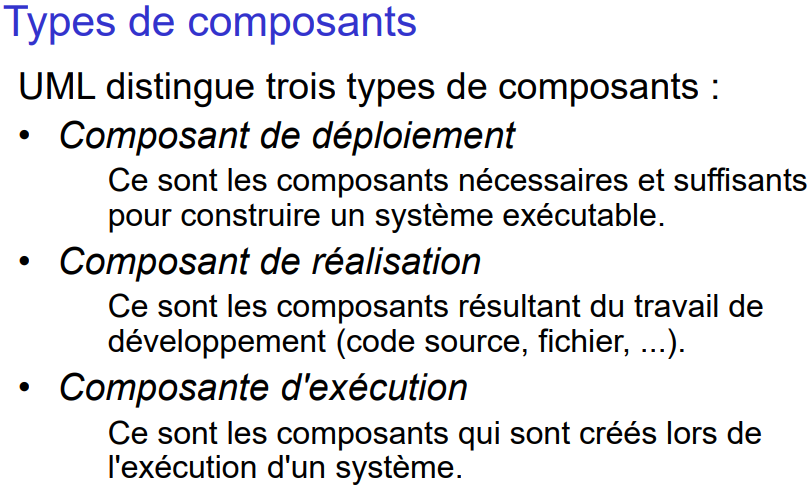
* On regroupe des éléments fournissant un service commun dans un paquetage.

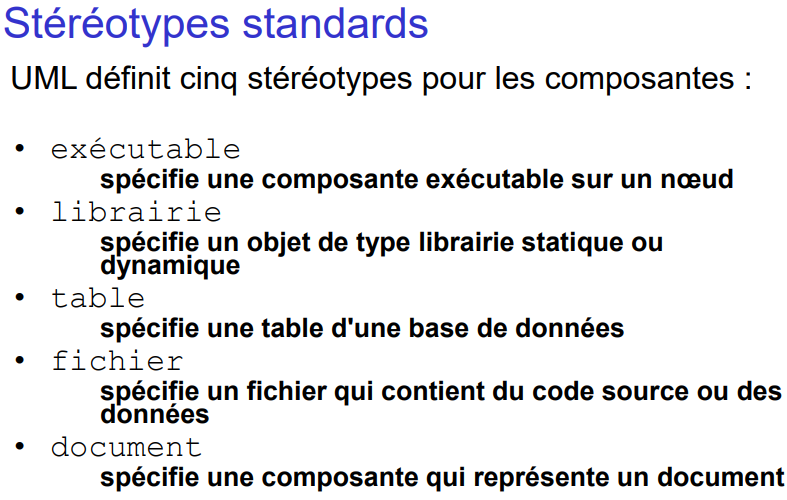
## Composants & Déploiement

1. Diagramme de composant

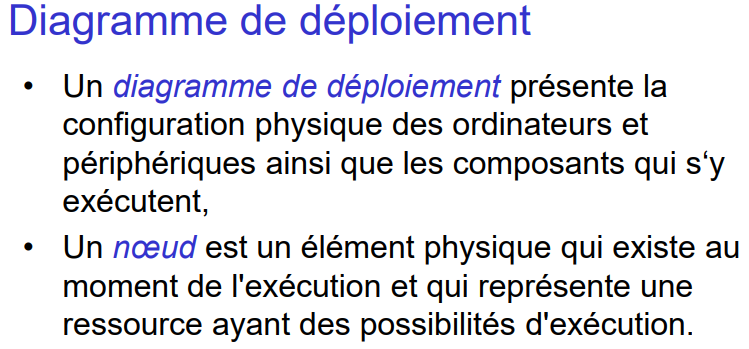


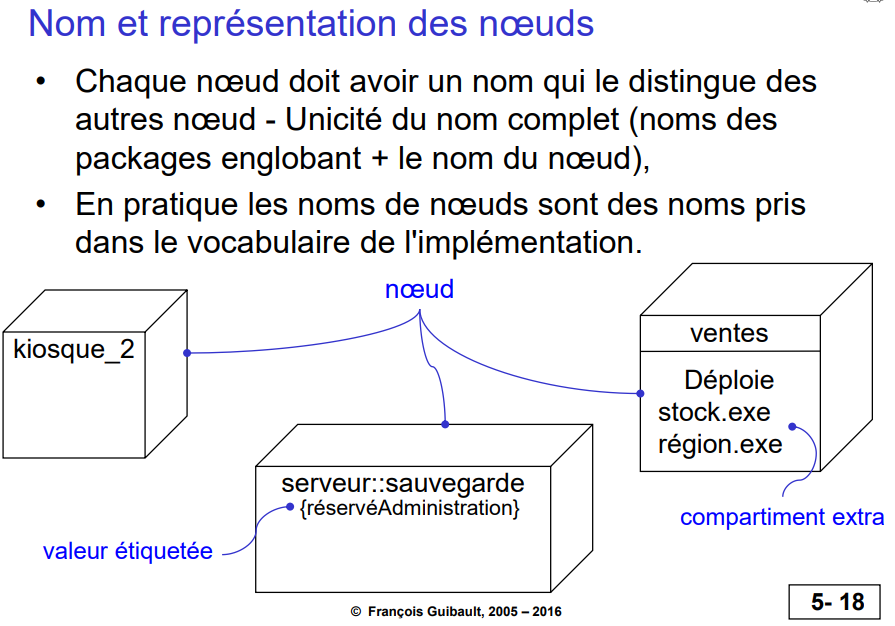
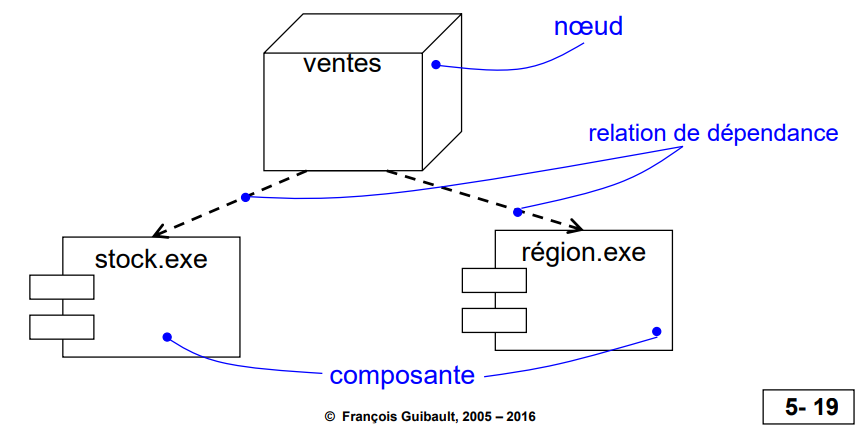
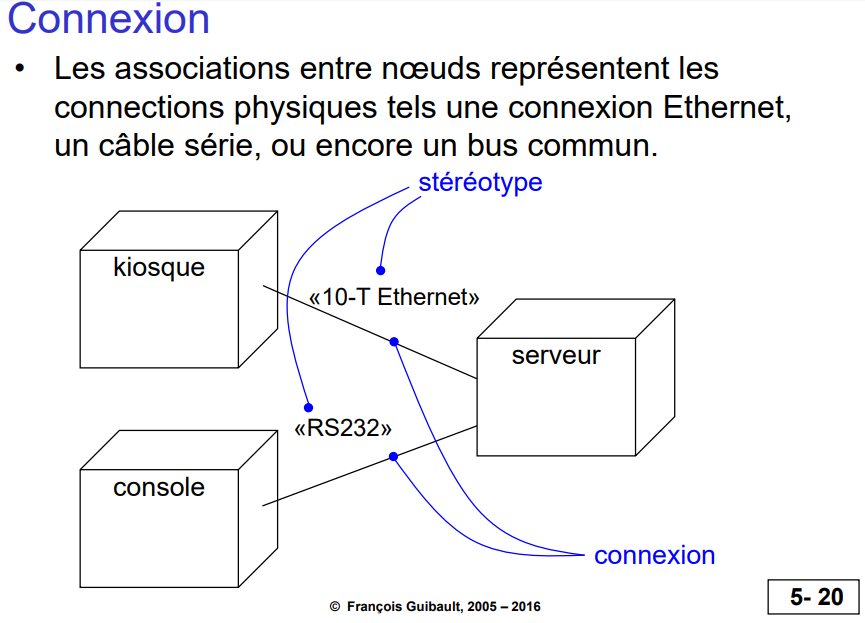
* Un diagramme de composants présente une vue statique de l'implémentation d'un système.
* Un composant est une partie physique et remplaçable d'un système qui se conforme et réalise un ensemble d'interfaces
* Un composant est l'implémentation physique de classes ou de paquetages.
* Une interface est une collection de spécifications d'opérations qui définissent le service rendu par une classe ou un composant.

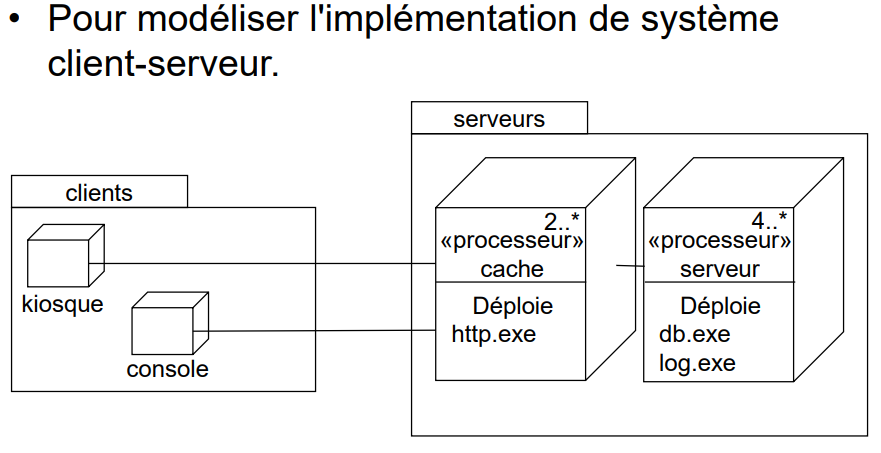




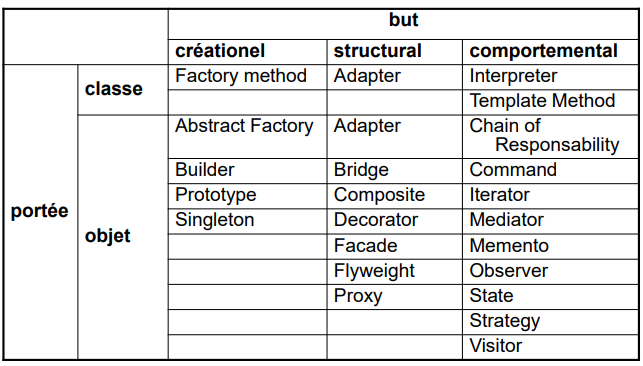
1. Diagramme de déploiement





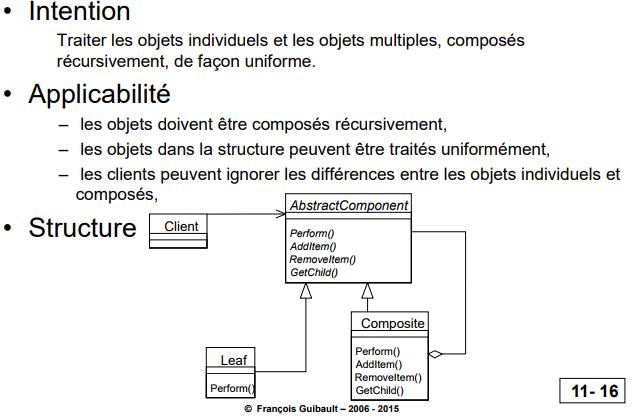
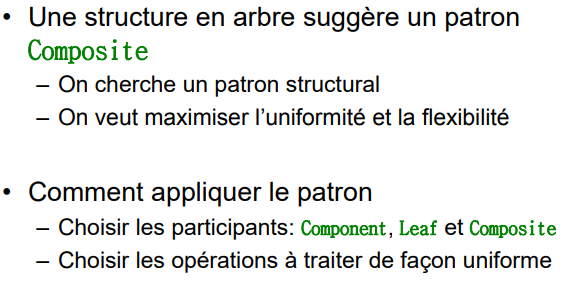
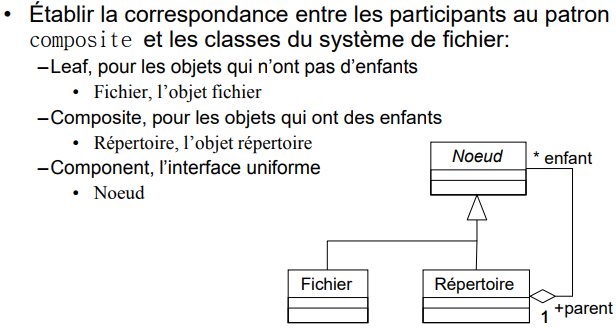


# CHAPITRE 6 – PATRONS

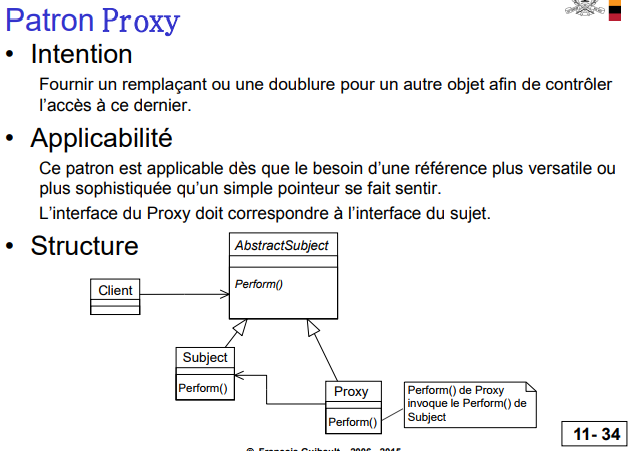


## Composite et Proxy

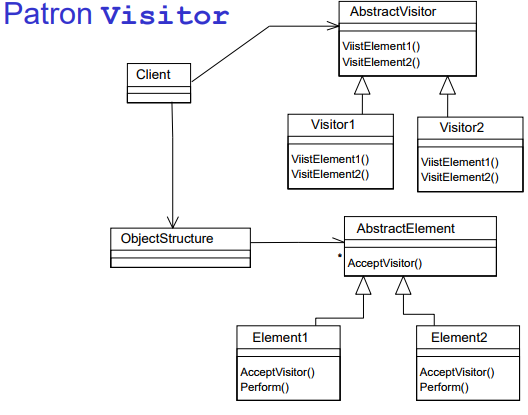
1. Composite

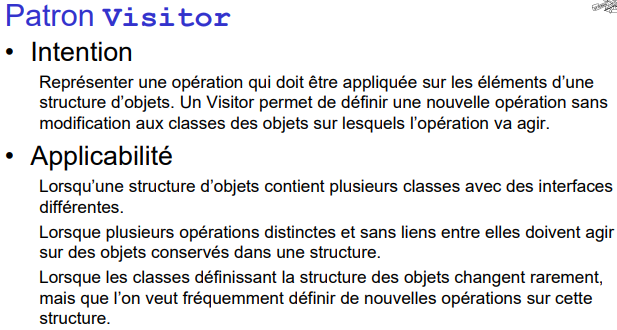
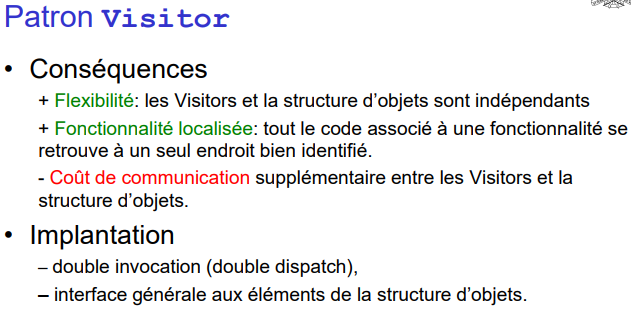


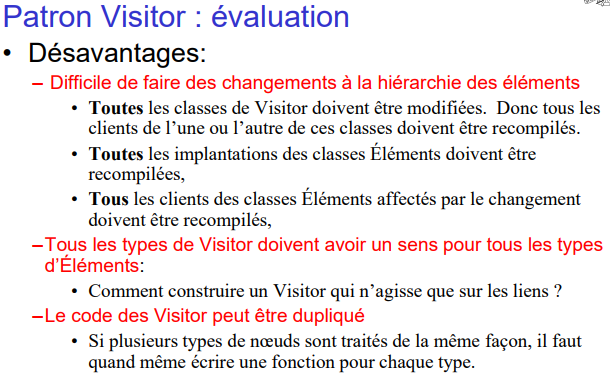
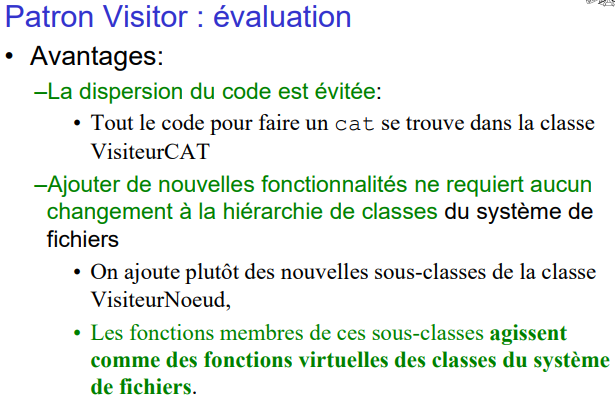
1. Proxy



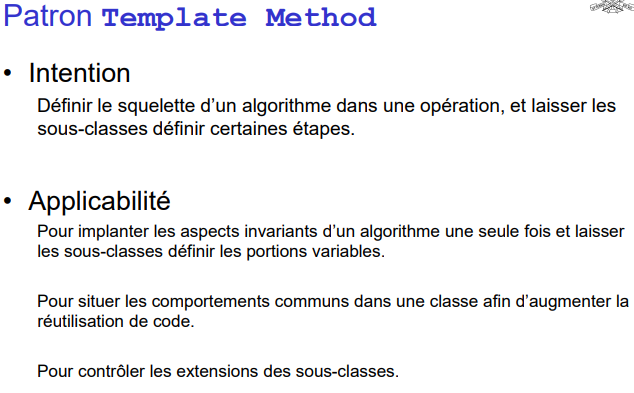
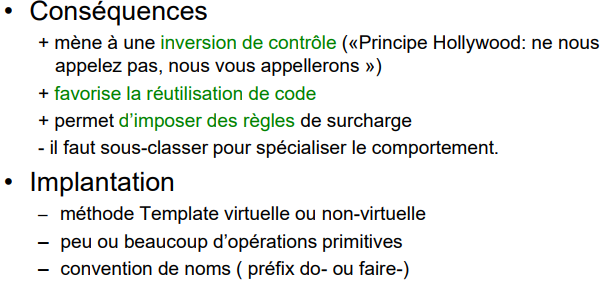
## Visitor, Template Method and Singleton

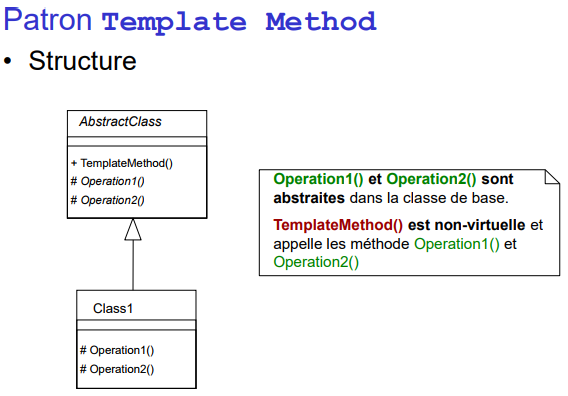
1. Patron VIsitor



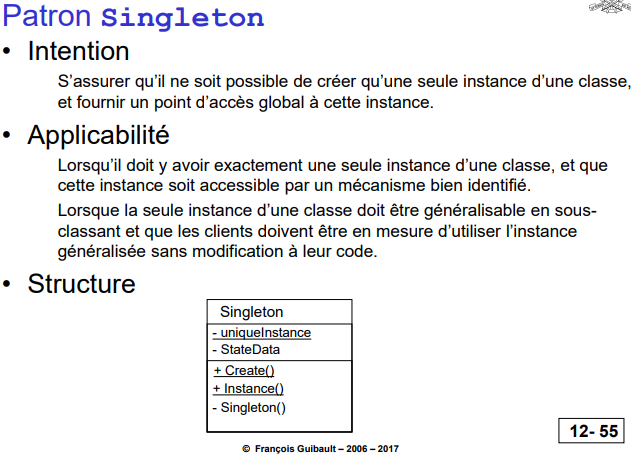


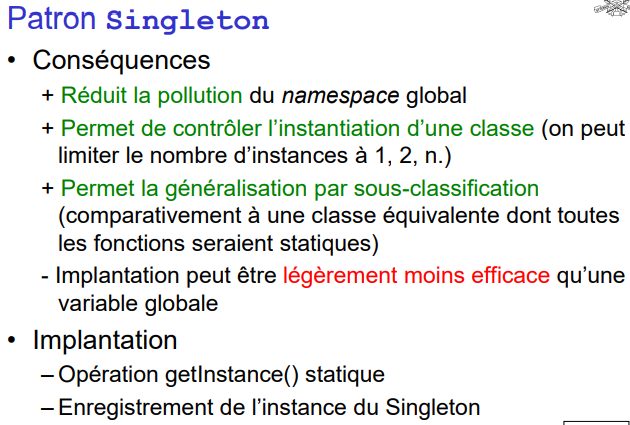
1. Patron Template Method

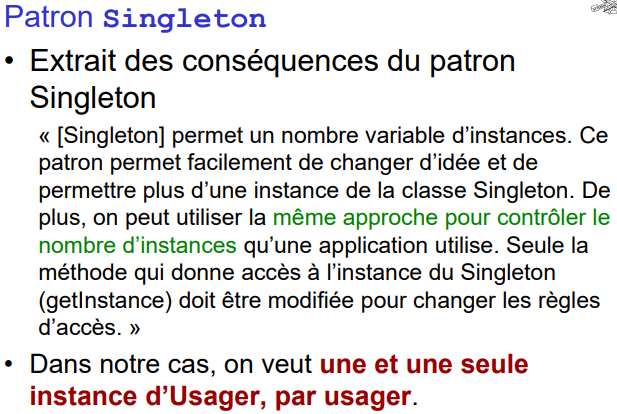
 



1. Patron Singleton

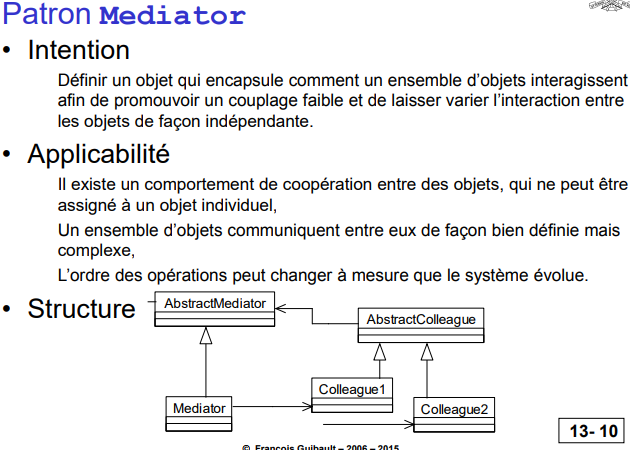


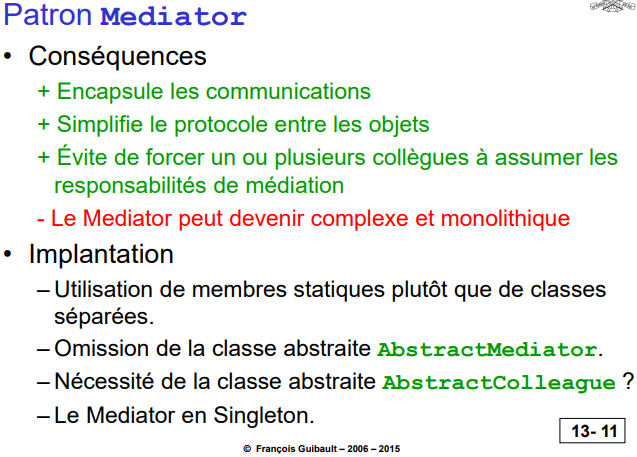


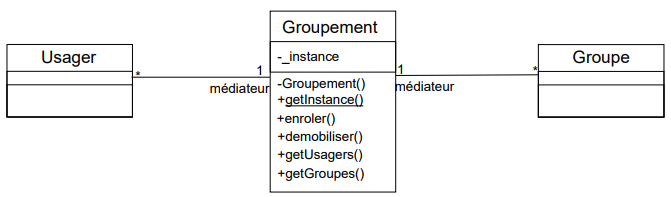


## Mediator, Observer, Abstract Factory et Prototype

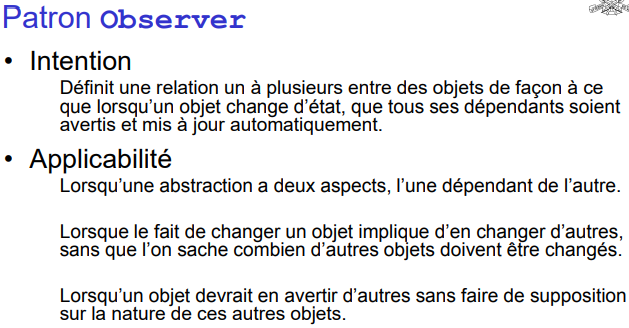
1. Mediator

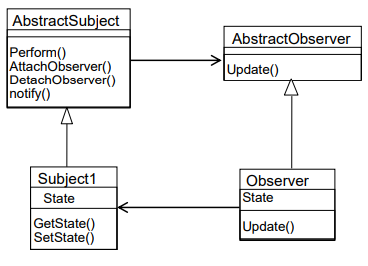
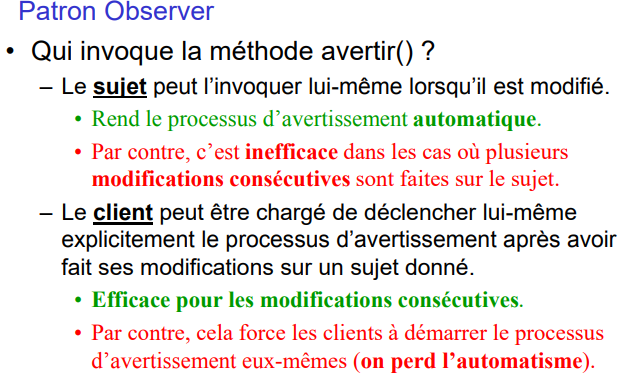


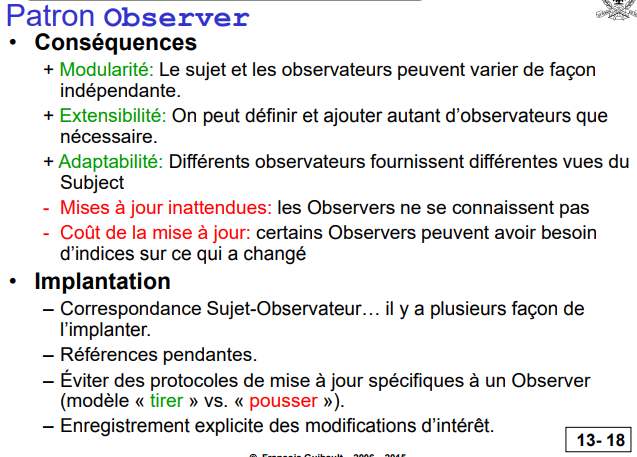




1. Observer

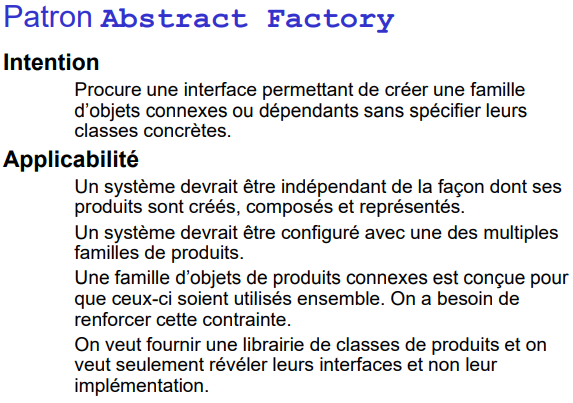
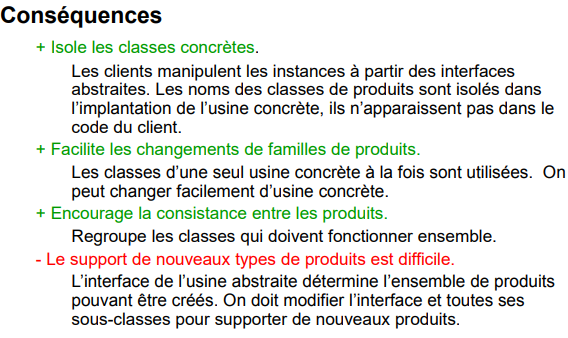
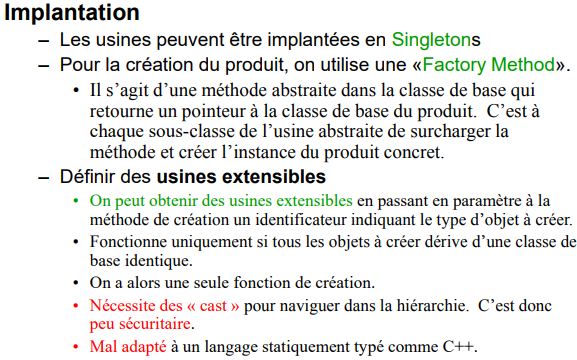


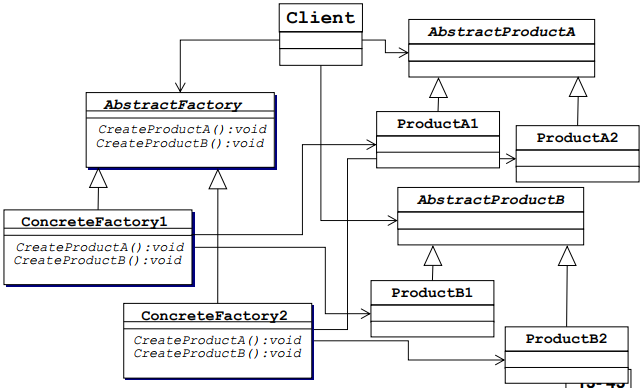




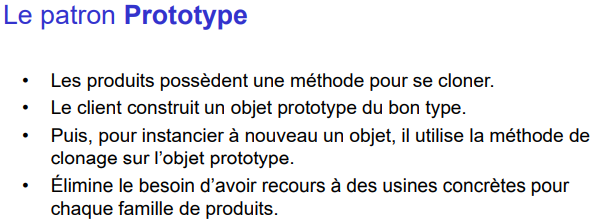


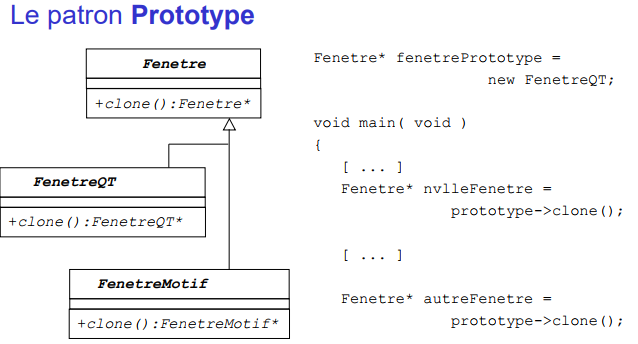
1. Abstract Factory





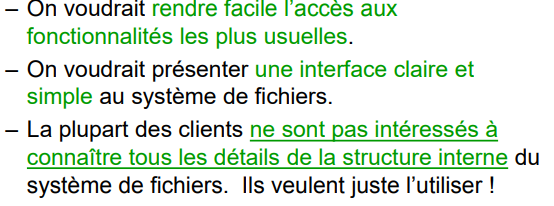
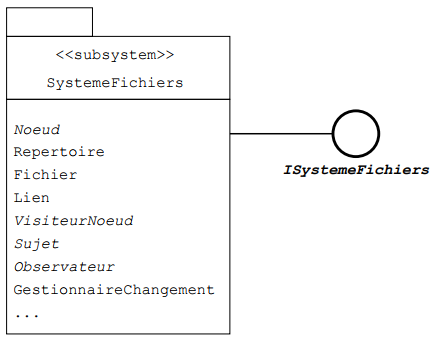
1. Prototype

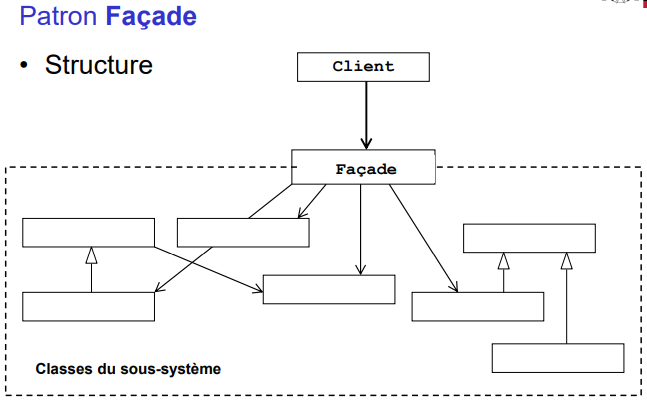




## Façade et Chain of Responsability

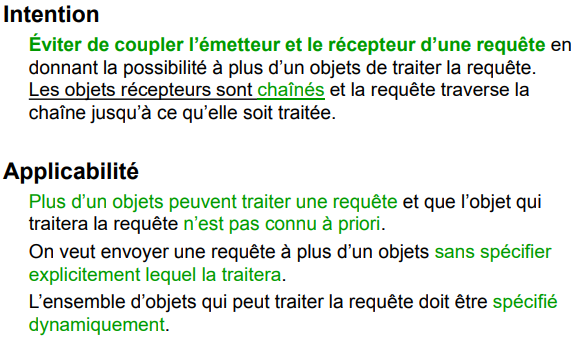
1. Façade

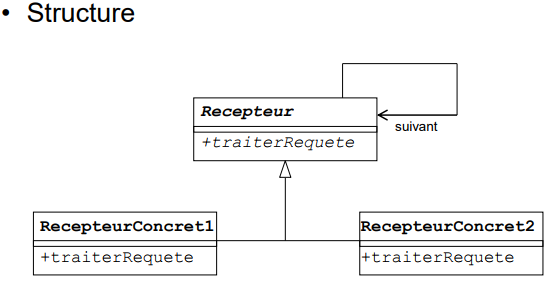


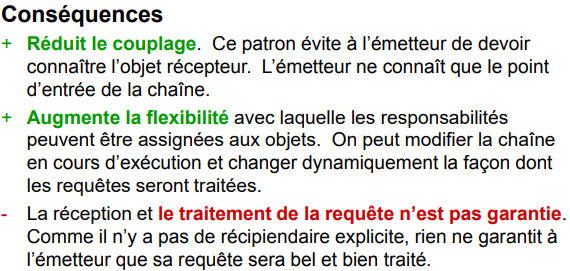




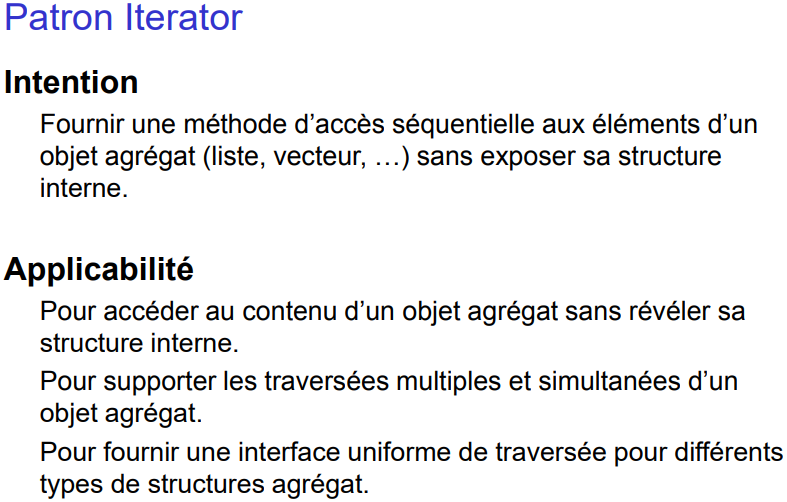
1. Chain of Responsability

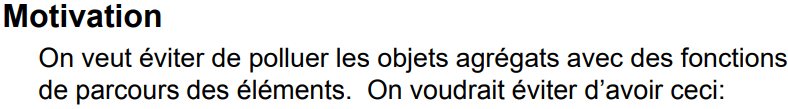


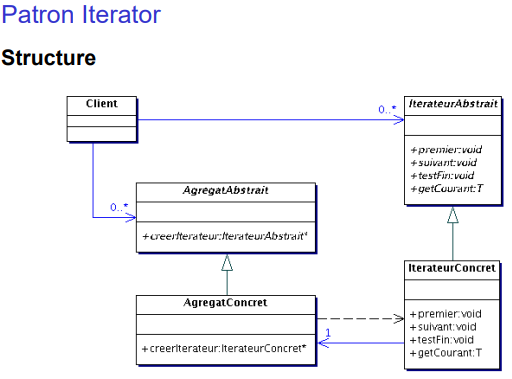


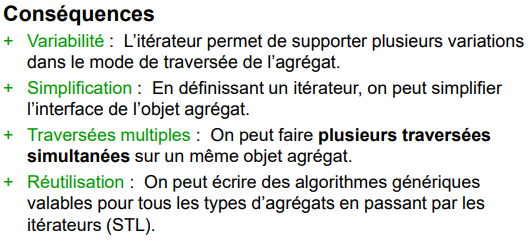


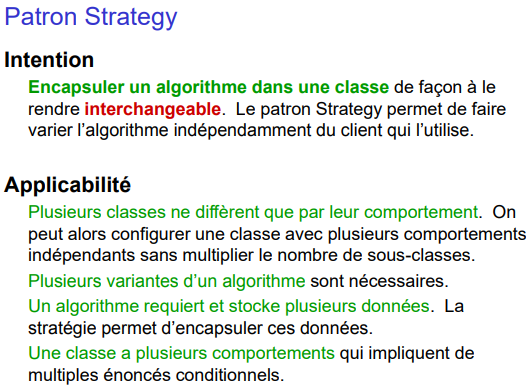
## Iterator, Strategy, State et Command

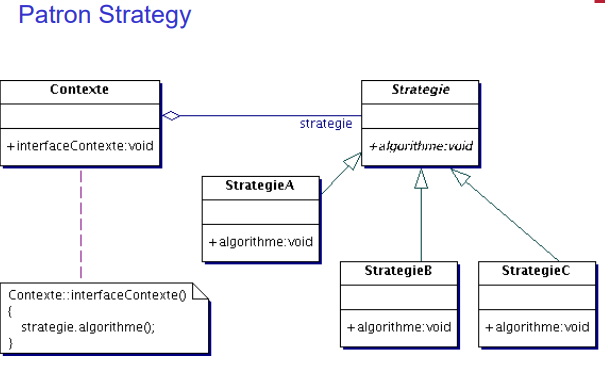


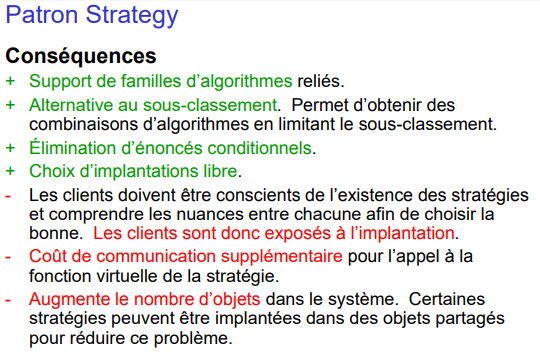


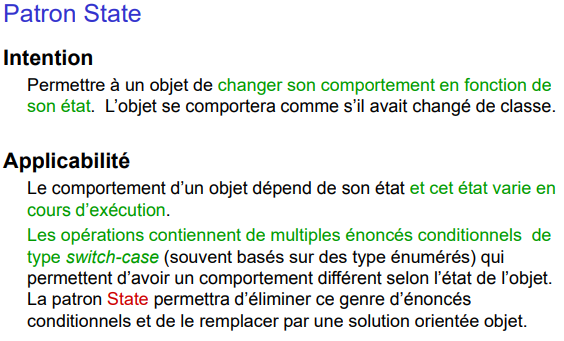


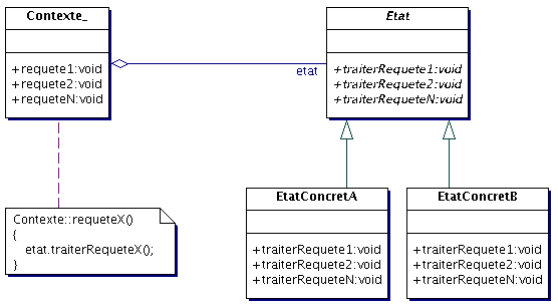






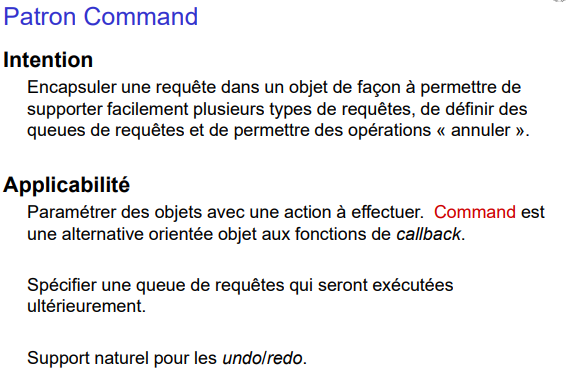
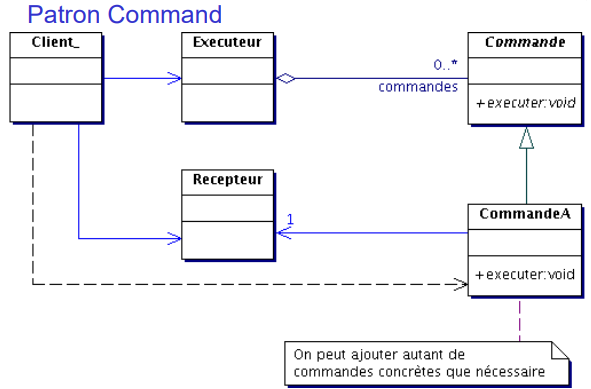
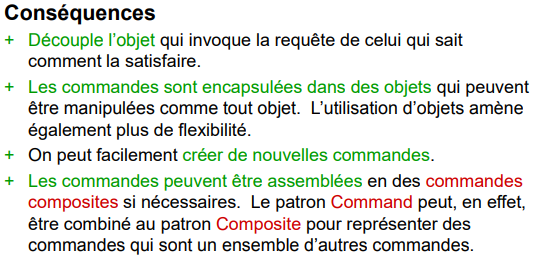












# CHAPTER 7 – EXCEPTIONS

