Introduction aux Designs Patterns

Composition d'un Design Pattern

- Sa **motivation** : un scénario qui illustre un cas de conception
- De son **intention** : ce qu'il fait, son but,
- Ses **constituants** : les différentes classes (abstraites et concrètes), interfaces et objets utilisés
- Sa **structure** : sa solution quasiment tout le temps exprimée en UML.
- Ses **collaborations** : comment les instances communiquent entre elles.

COO et Les Design Patterns

- COO
  - Programmer avec des interfaces plutôt qu'avec des classes spécifiques
  - Réutilisation : Penser plutôt Boite noire (Composition) que boite blanche (héritage)
- Design Patterns
  - Les problèmes sont récurrents s'ils sont exprimés de façons génériques ⇒ Utilisation d'interfaces
  - La bonne solution de beaucoup de problèmes récurrents est pensée en terme de réutilisation ⇒ compostion plutôt qu'héritage

Plan de la présentation

- Patterns Créateurs
  - Fabrication
  - Fabrique abstraite
  - Singleton
- Pattern Comportemental
  - Iterateur

Le pattern créateur Fabrique Abstraite

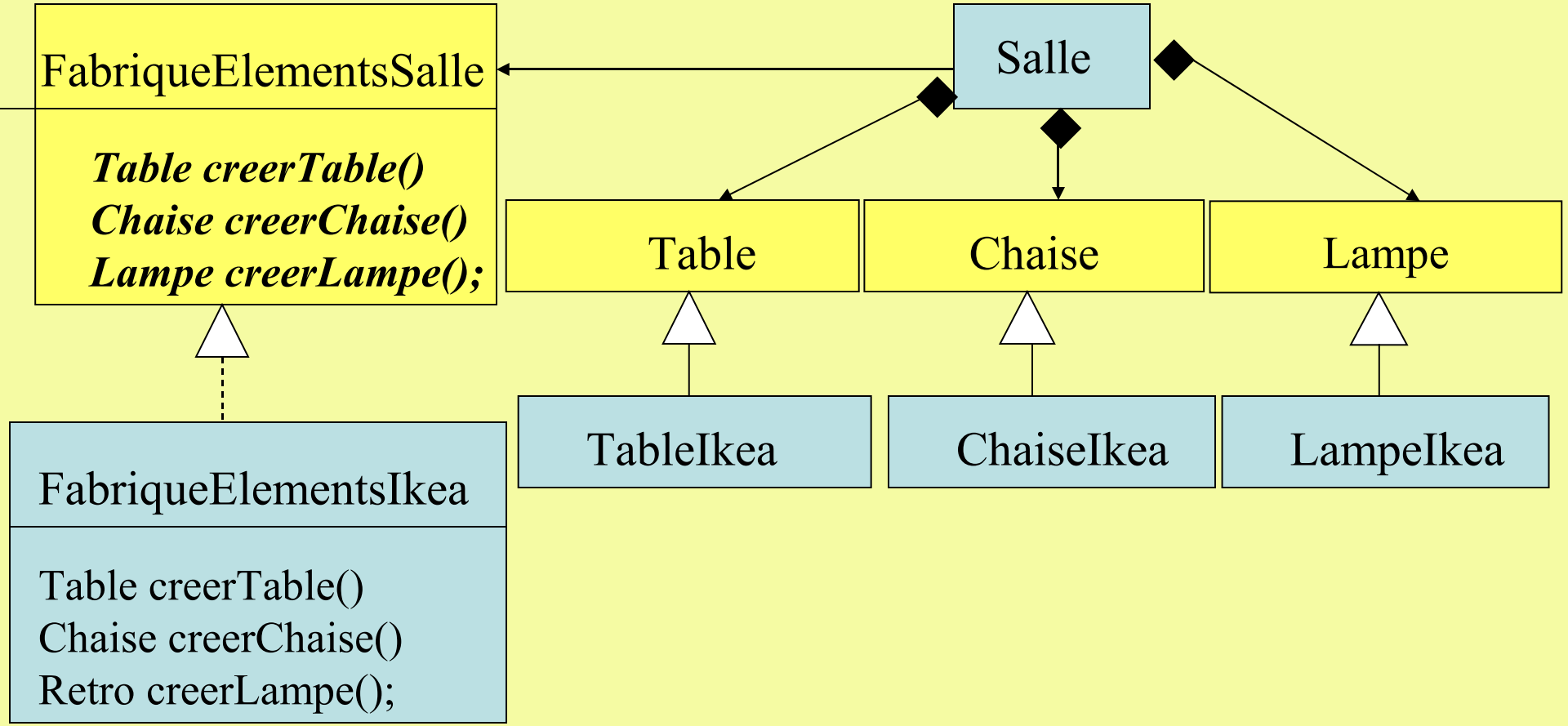
- **La motivation**
  - On désire définir à l'aide d'un logiciel, le contenu d'une salle de travail : tables, chaises et des lampes
  - D'une salle à l'autre ou d'une école à l'autre, les types des éléments peuvent être différents
  - Pour une salle donnée, il ne faut donc pas « coder en dur » les différents éléments de cette dernière. Cela rendrait difficile un changement ultérieur du type des éléments

Le pattern créateur Fabrique Abstraite

- **L'intention**
  - La fabrique abstraite fournit une interface pour la création d'une famille d'objets liés entre eux ou apparentés sans qu'il soit nécessaire de spécifier le type des différents objets

Le pattern créateur Fabrique Abstraite

- **Retour sur notre motivation**

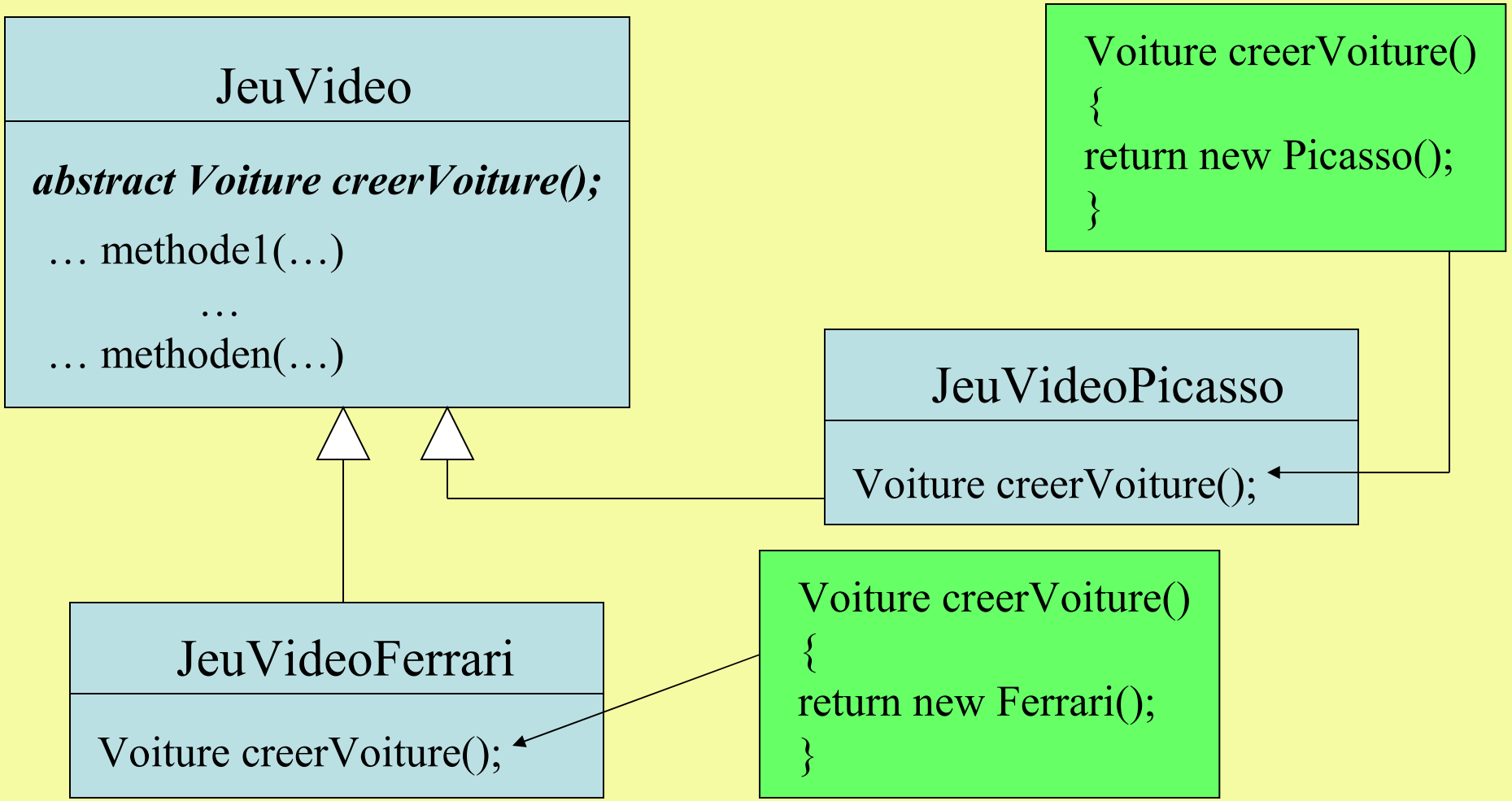


Le pattern créateur Fabrication

- **La motivation**
  - On conçoit un jeu vidéo qui permet de piloter des voitures. On veut concevoir de nouvelles voitures indépendamment de la conception du jeu vidéo.
  - Le jeu vidéo connaît donc l'interface d'une voiture mais ne connaît le type des voitures utilisées.
  - Pour disposer d'une voiture, le jeu vidéo doit pouvoir fabriquer la voiture sans en connaître le type !!!.

Le pattern créateur Fabrication

- **Retour sur notre motivation**



Le pattern créateur Fabrication

- **L'intention**
  - Définir une classe (abstraite) qui implémente toutes ses méthodes qui utilisent des objets et ne fait que déclarer les méthodes de fabrication des objets utilisés.
  - La méthode de fabrication permet à la classe abstraite de déléguer l'instanciation des objets utilisés à d'autres classes.

Le pattern créateur Fabrication

- **Les constituants**
  - Produit : l'interface des objets créés par la fabrication
  - ProduitConcret : Un classe qui implémente l'interface Produit.
  - Facteur : La classe qui déclare la méthode de fabrication et qui implémente les autres méthodes
  - FacteurConcret : La classe qui hérite de la classe Facteur en surchargeant la méthode de fabrication

Le pattern créateur Singleton

- **La motivation**
  - La quasi-totalité des logiciels communique lors de leur exécution avec une base de données.
  - Pour gérer une seule et même transaction lors d'une exécution d'un logiciel, il faut être assuré qu' à tout instant de l'exécution on manipule la même session (au sens base de données).

Le pattern créateur Singleton

- **L'intention de ce pattern**
  - Ce pattern permet pour une classe donnée de ne fabriquer qu'un seul objet et donc de manipuler à tout instant d'une exécution le même objet.

- **Les constituants de ce pattern**
  - Il n'y a qu'un seul constituant : la classe qui fournit l'unique objet.

Le pattern créateur Singleton

- **La collaboration**
  - Cette classe collabore avec les autres classes à l'aide :
    - D'une méthode statique qui permet de récupérer une référence sur l'unique objet de la classe
    - De l'ensemble des méthodes qui définissent le comportement de cet objet.

Le pattern Iterateur

- **Motivation**
  - On peut vouloir parcourir un conteneur de plusieurs façons différentes
    - Parcourir un arbre en profondeur ou parcourir un arbre en largeur
    - Parcourir une liste du premier élément au dernier élément ou parcourir du dernier élément au premier

Le pattern Iterateur

- **Motivation**
  - Au même instant, on peut avoir en cours, plusieurs parcours d'un même conteneur
  - On peut enfin vouloir parcourir un conteneur sans risque de modification intempestive de la structure interne du conteneur

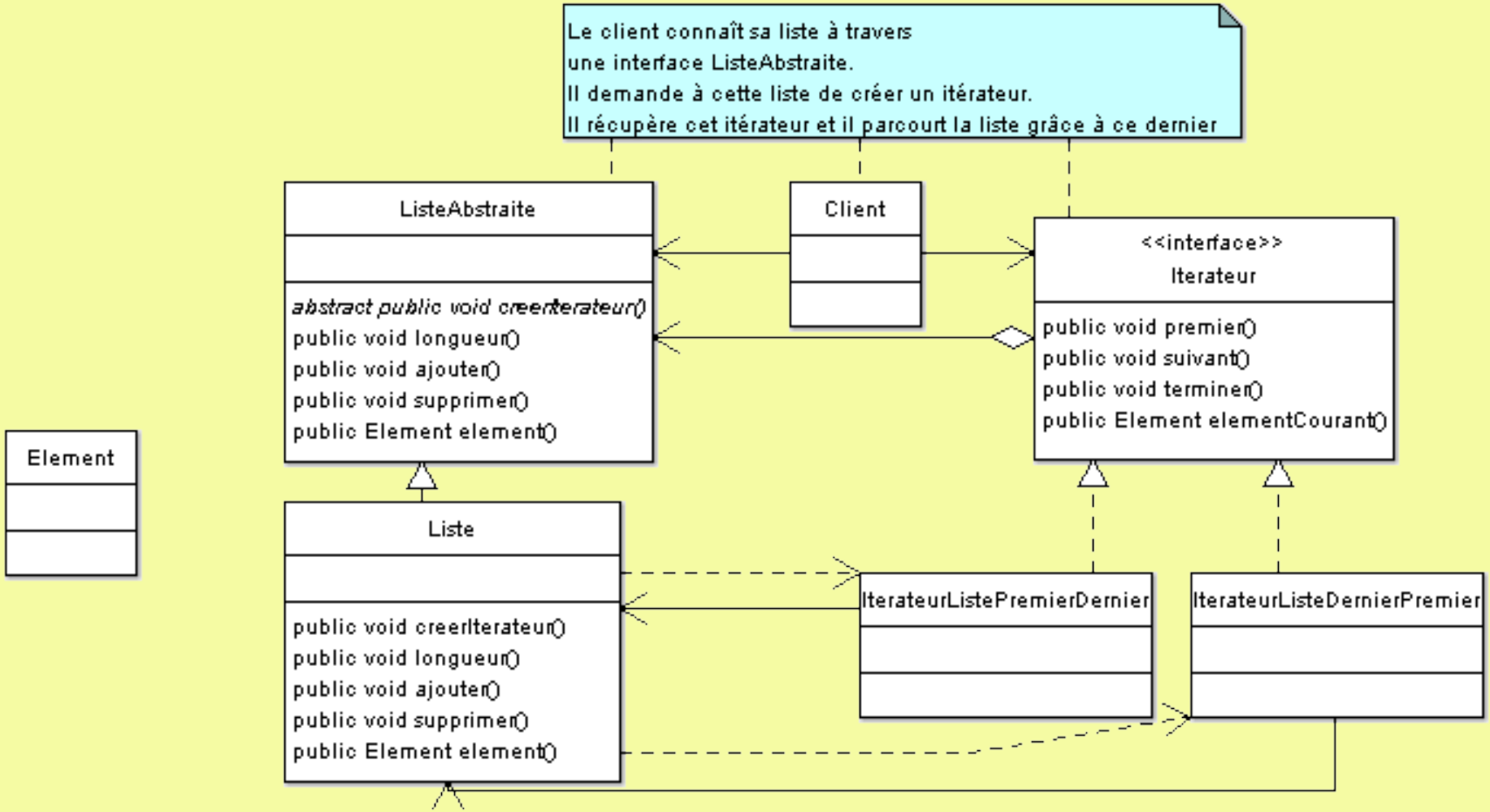
Le pattern Iterateur

- **Intention**

  Fournir un moyen d'accès séquentiel à un agrégat d'objets sans mettre à découvert la représentation interne du ce dernier

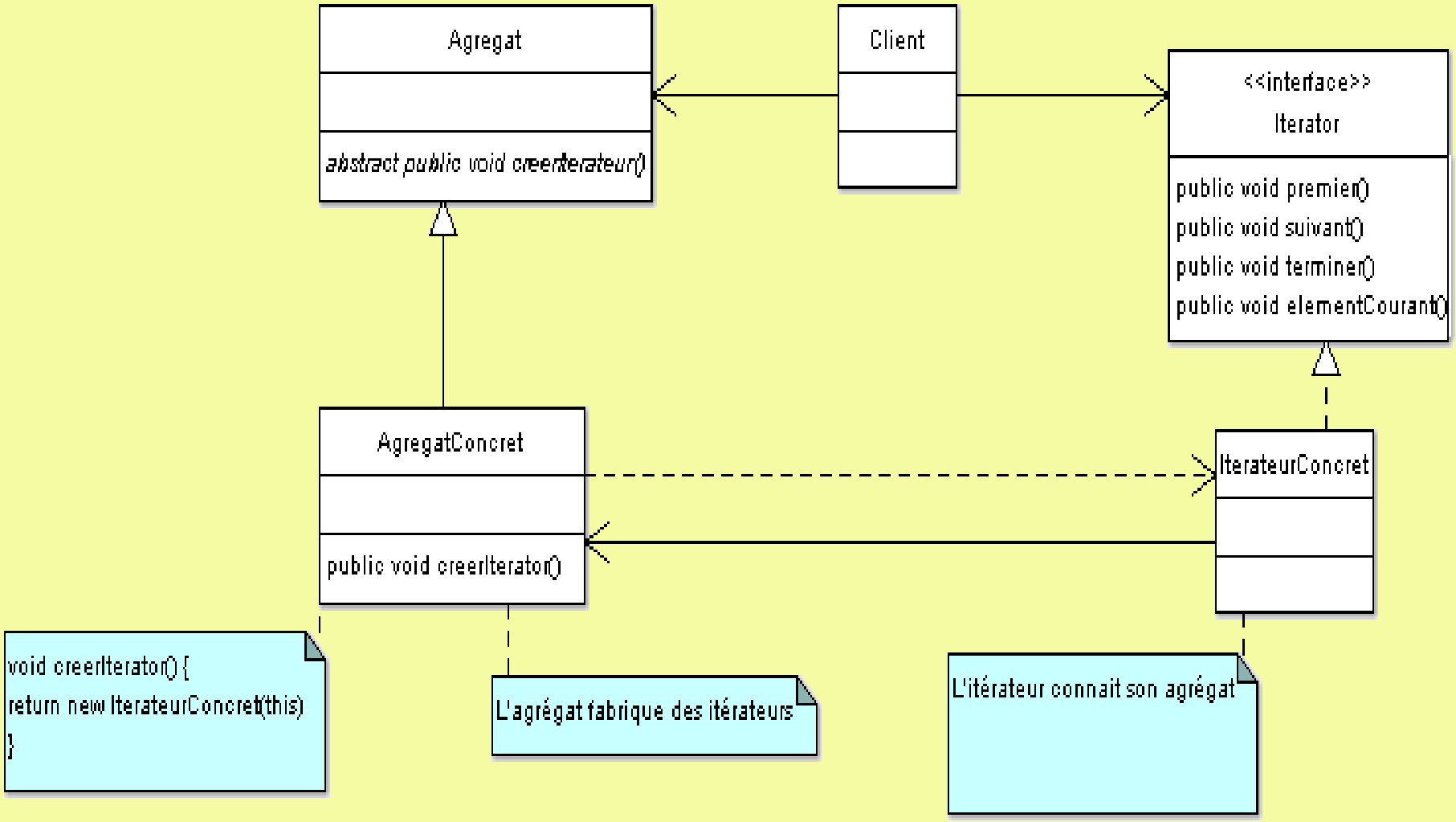
Le pattern Iterateur

La structure du Pattern et ses composants avec une liste



Le pattern Iterateur

La structure générique du Pattern et ses composants



Le pattern Iterateur

- **Composant**
  - La classe IterateurNul est un itérateur dégénéré qui est toujours terminé
  - Les itérateurs en Java