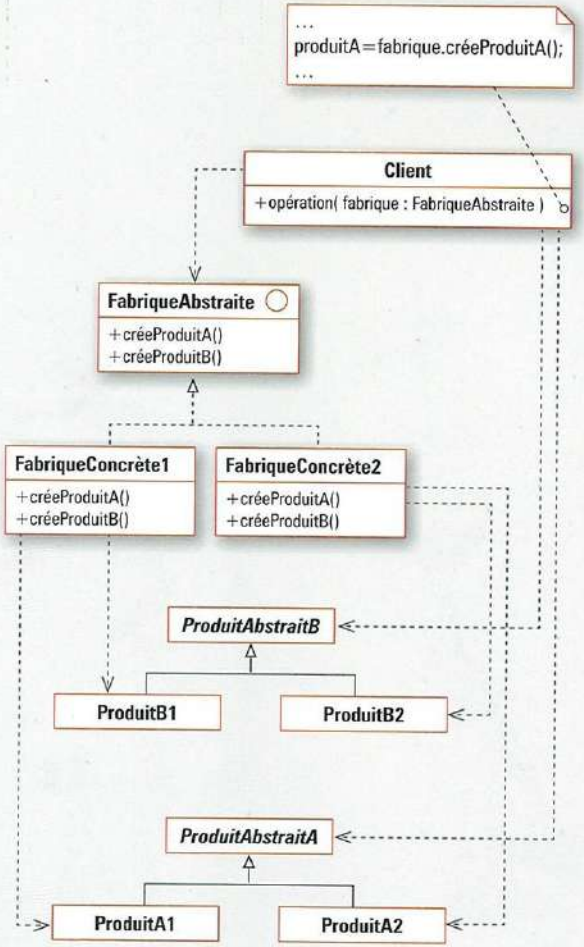
Patterns de création

Abstract Factory

Fournir une interface de création d'objets regroupés en familles sans devoir connaître les classes concrètes destinées à la création de ces objets.

Utilisations :

- Un système utilisant des produits a besoin d'être indépendant de la manière dont ces produits sont créés et regroupés.
- Un système est paramétré par plusieurs familles de produits qui peuvent évoluer.



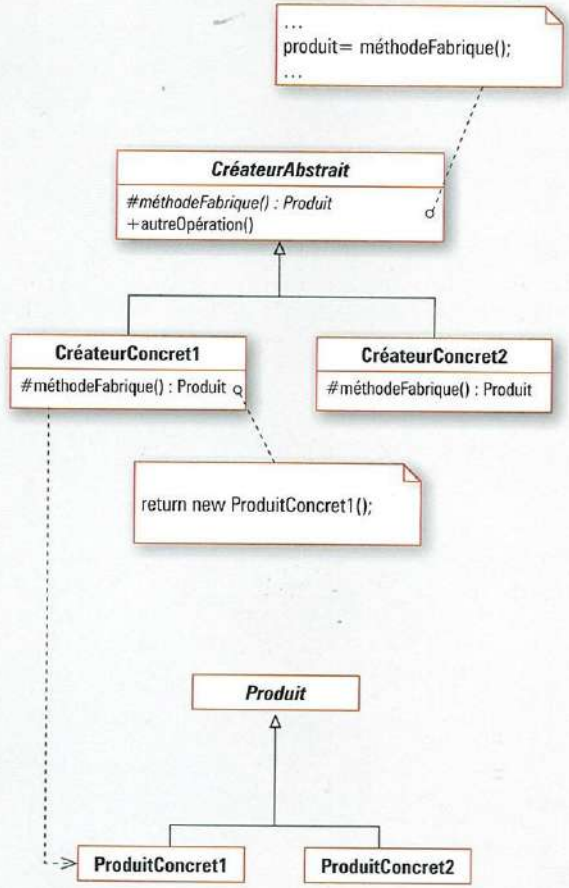
Patterns de création

Factory Method

Introduire une méthode abstraite de création d'un objet en déléguant aux sous-classes concrètes la création effective.

Utilisations :

- Une classe ne connaît que les classes abstraites des objets avec lesquels elle possède des relations.
- Une classe veut transmettre à ses sous-classes les choix d'instanciation en profitant du mécanisme de polymorphisme.



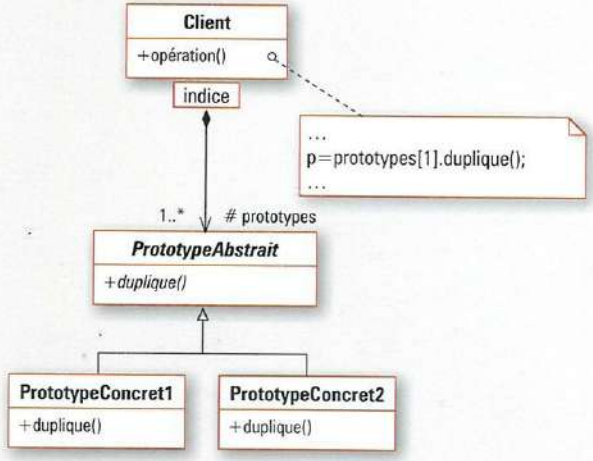
Patterns de création

Prototype

Créer des nouveaux objets par duplication d'objets existants appelés prototypes qui disposent de la capacité de clonage.

Utilisations :

- Un système doit créer des instances sans connaître la hiérarchie des classes les décrivant.
- Un système doit rester simple et ne pas inclure une hiérarchie parallèle de classes de fabrique.

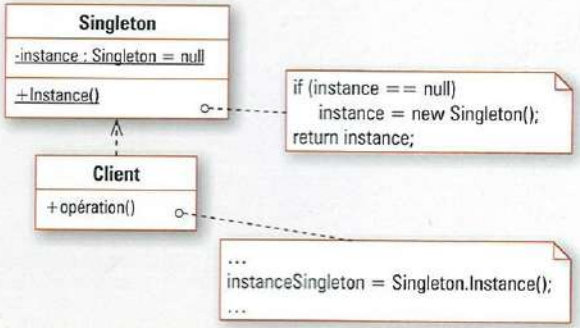


Singleton

Garantir qu'une classe ne possède qu'une seule instance et que cette classe fournisse une méthode retournant cette instance unique.

Utilisation :

- Limiter le nombre d'instances d'une classe à une seule.



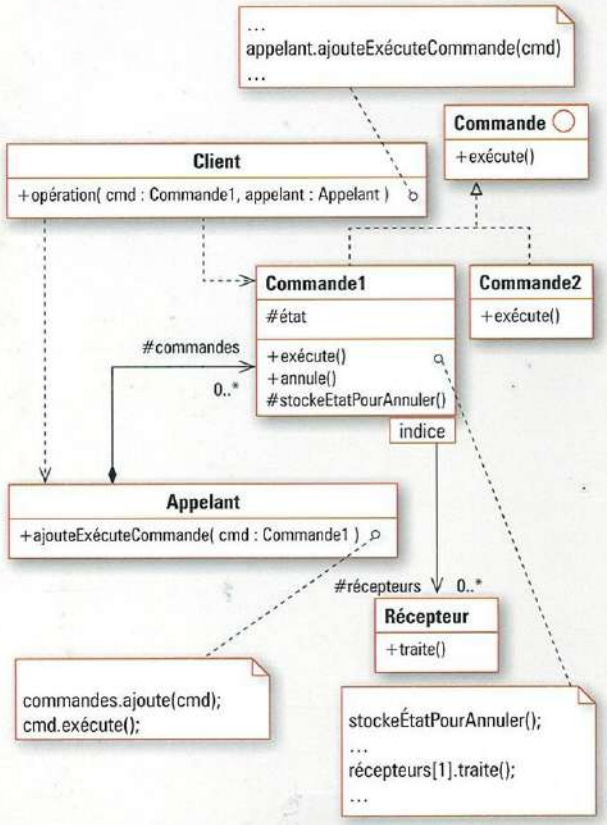
Patterns de comportement

Command

Transformer une commande (requête) en un objet, facilitant des opérations comme l'annulation, la mise en file des requêtes et le suivi.

Utilisations :

- Un objet doit être paramétré par un traitement à réaliser. C'est l'appelant (voir diagramme ci-dessous) qui est ici paramétré par une commande qui contient la description d'un traitement à réaliser sur un ou plusieurs récepteurs.
- Les commandes doivent être stockées dans une file et pouvoir être exécutées à un moment quelconque, éventuellement plusieurs fois.
- Les commandes sont annulables et/ou peuvent être tracées.
- Les commandes doivent être regroupées sous la forme d'une transaction.



ISBN : 978-2-7460-5294-9
Prix : 6 €
www.editions-eni.fr

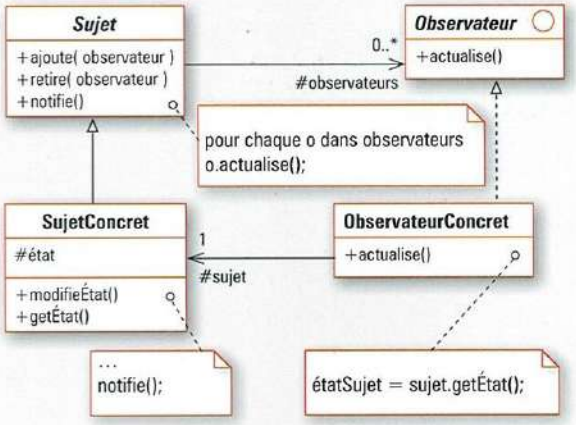
Patterns de comportement

Observer

Construire une dépendance entre un sujet et des observateurs de sorte que chaque modification du sujet soit notifiée aux observateurs afin qu'ils puissent mettre à jour leur état.

Utilisations :

- Une modification dans l'état d'un objet engendre des modifications dans d'autres objets qui sont déterminés dynamiquement.
- Un objet veut prévenir d'autres objets sans devoir connaître leur type, c'est-à-dire sans être fortement couplé à ceux-ci.
- Au sein d'un système, il n'est pas souhaitable de fusionner deux objets en un seul.

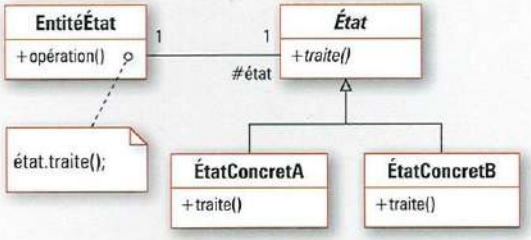


State

Adapter le comportement d'un objet (machine à états) en fonction de son état interne.

Utilisations :

- Le comportement d'un objet dépend de son état.
- L'implantation de cette dépendance à l'état par des instructions conditionnelles est trop complexe.



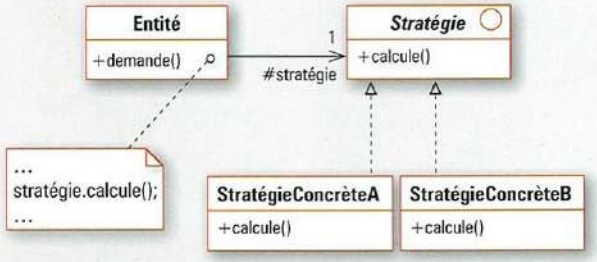
Patterns de comportement

Strategy

Adapter le comportement et les algorithmes d'un objet en fonction d'un besoin sans modifier les interactions de cet objet avec ses clients.

Utilisations :

- Le comportement d'une classe peut être implanté par différents algorithmes dont certains sont plus efficaces en temps d'exécution ou en consommation mémoire ou encore contiennent des mécanismes de mise au point.
- L'implantation du choix de l'algorithme par des instructions conditionnelles est trop complexe.

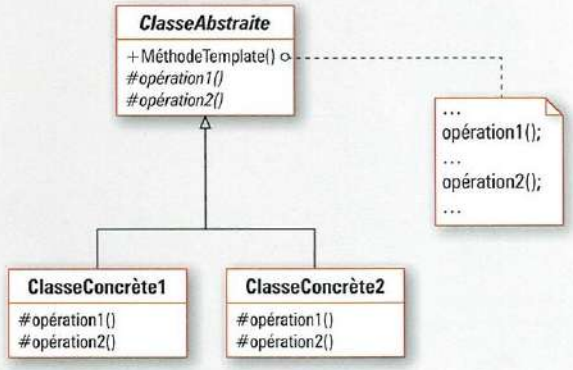


Template Method

Reporter dans des sous-classes certaines étapes de l'une des opérations d'un objet, ces étapes étant alors décrites dans les sous-classes.

Utilisations :

- Une classe partage avec une autre ou plusieurs autres classes du code commun qui peut être factorisé après que la ou les parties spécifiques à chaque classe aient été déplacées dans des nouvelles méthodes.
- Un traitement possède une partie invariable et des parties spécifiques à différents types d'objets.



© Editions ENI - www.editions-eni.fr

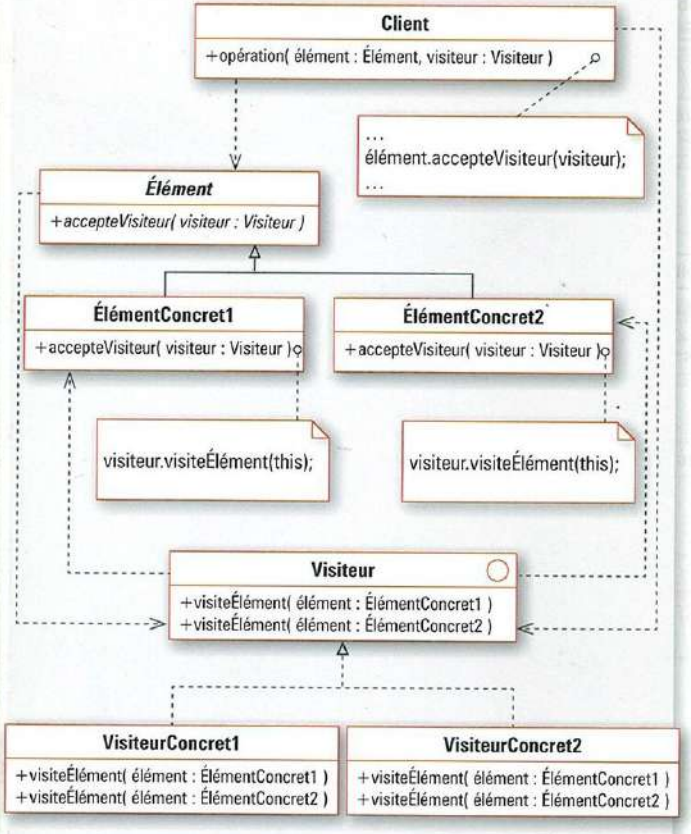
Patterns de comportement

Visitor

Construire une ou plusieurs opérations à réaliser sur les éléments d'un ensemble d'objets. De nouvelles opérations sont ainsi ajoutées sans modifier les classes de ces objets.

Utilisations :

- De nombreuses fonctionnalités doivent être ajoutées à un ensemble de classes sans que ces ajouts viennent alourdir ces classes.
- Un ensemble de classes possède une structure fixe et il est nécessaire d'adjoindre à chaque classe des fonctionnalités sans en modifier l'interface. Cependant, si la structure de l'ensemble des classes auxquelles il est nécessaire d'adjoindre des fonctionnalités est souvent modifiée, le pattern Visitor n'est pas adapté. En effet, toute modification de la structure implique une modification de chaque visiteur.



Les neuf autres patterns

Pattern de création

Builder

Abstraire la construction d'objets complexes de leur représentation de sorte qu'un client puisse créer ces objets complexes sans devoir se préoccuper des différences de représentation.

Patterns de structure

Bridge

Séparer l'aspect d'implantation d'un objet de son aspect de représentation et d'interface afin que ces deux éléments puissent évoluer indépendamment l'un de l'autre.

Facade

Regrouper les interfaces d'un ensemble d'objets en une interface unifiée rendant cet ensemble plus simple à utiliser pour un client.

Flyweigth

Partager de façon efficace un ensemble important d'objets de grain fin.

Patterns de comportement

Chain of Responsibility

Construire une chaîne d'objets telle que si un objet de la chaîne ne peut pas répondre à une requête, il puisse la transmettre à son successeur et ainsi de suite jusqu'à ce que l'un des objets de la chaîne la traite.

Interpreter

Fournir un cadre pour donner une représentation par objets de la grammaire d'un langage afin d'interpréter des expressions écrites dans ce langage.

Iterator

Fournir un accès séquentiel à une collection d'objets à des clients sans que ceux-ci aient à se préoccuper de l'implantation de cette collection.

Mediator

Élaborer un objet dont la vocation est la gestion et le contrôle des interactions dans un ensemble d'objets sans que ces éléments aient à se connaître mutuellement (couplage faible).

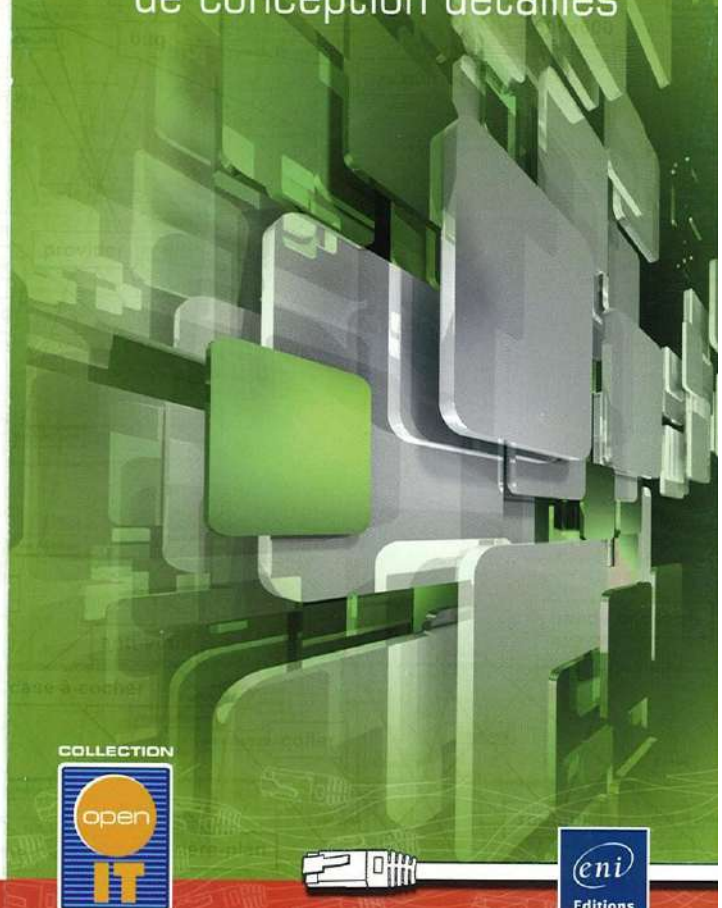
Memento

Sauvegarder et restaurer l'état d'un objet sans en violer l'encapsulation.

© Editions ENI - www.editions-eni.fr

Design Patterns

14 patterns
de conception détaillés



COLLECTION open IT

eni Editions

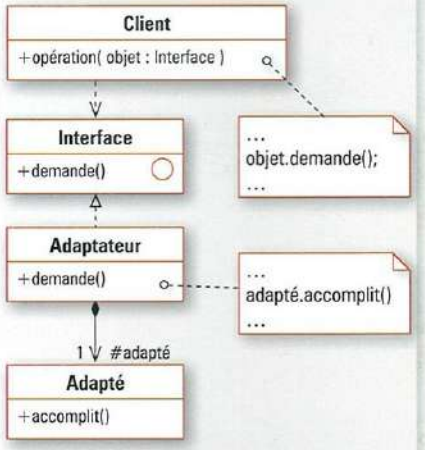
Patterns de structure

Adapter

Convertir l'interface d'une classe existante en l'interface attendue par des clients également existants afin qu'ils puissent coopérer.

Utilisations :

- Intégrer dans un système un objet dont l'interface ne correspond pas à l'interface requise au sein de ce système.
- Fournir des interfaces multiples à un objet lors de sa conception.

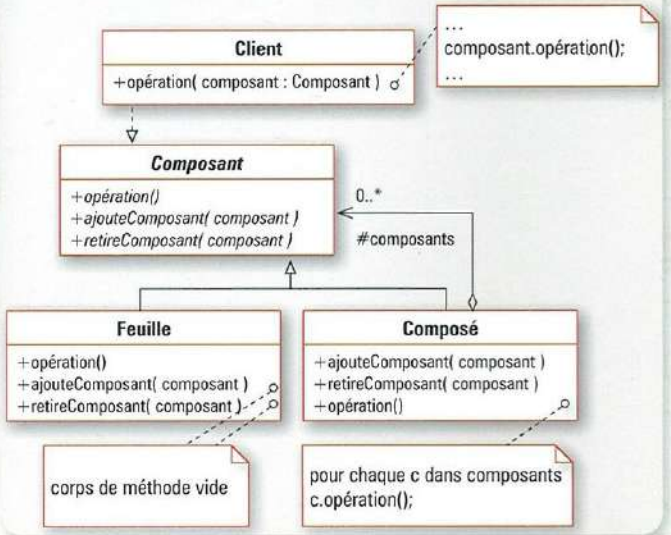


Composite

Offrir un cadre de conception d'une hiérarchie de composition arborescente d'objets sans que les clients aient à traiter différemment les composés et les individus.

Utilisations :

- Il est nécessaire de représenter au sein d'un système des hiérarchies de composition.
- Les clients d'une composition doivent ignorer s'ils communiquent avec des objets composés ou des individus.



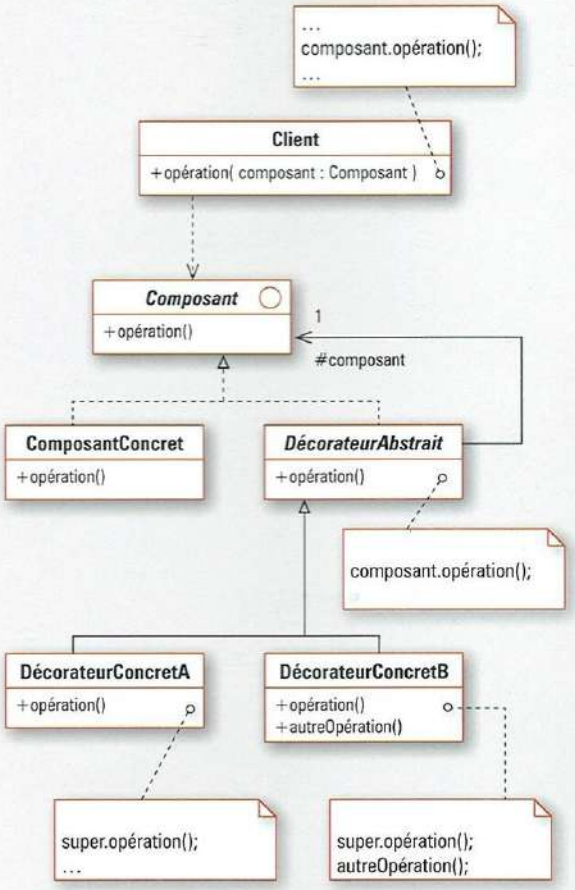
Patterns de structure

Decorator

Ajouter dynamiquement des fonctionnalités supplémentaires à un objet tel que cet ajout de fonctionnalités ne modifie pas l'interface de l'objet et reste donc transparent vis-à-vis des clients.

Utilisations :

- Un système ajoute dynamiquement des fonctionnalités à un objet, sans modifier son interface, c'est-à-dire sans que les clients de cet objet aient à être modifiés.
- Un système gère des fonctionnalités qui peuvent être retirées à des objets.
- L'utilisation de l'héritage pour étendre des objets n'est pas pratique, ce qui peut arriver quand la hiérarchie des objets est déjà complexe.



Patterns de structure

Proxy

Concevoir un objet qui se substitue à un autre objet (le sujet réel) et en contrôle l'accès.

Utilisations :

- Le proxy virtuel permet de créer un objet de taille importante au moment approprié.
- Le proxy remote permet d'accéder à distance à un objet s'exécutant dans un autre environnement. Ce type de proxy est mis en œuvre dans les systèmes d'objets distants (CORBA, Java RMI).
- Le proxy de protection permet de sécuriser l'accès à un objet, par exemple par des techniques d'authentification.