

Les Patrons de Conception (Design Patterns)

Des solutions standard à des problèmes de conception récurrents

Olivier Camp
Mickaël Clavreul



ESEO
I3 - LD et SE

2016/2017

1 Introduction

1 Introduction

2 Classification des patrons

- Les rôles : créateurs, structurels ou comportementaux
- Les domaines : objets ou classes
- Rôles et domaines

1 Introduction

2 Classification des patrons

- Les rôles : créateurs, structurels ou comportementaux
- Les domaines : objets ou classes
- Rôles et domaines

3 Format de description des patrons de conception

1 Introduction

2 Classification des patrons

- Les rôles : créateurs, structurels ou comportementaux
- Les domaines : objets ou classes
- Rôles et domaines

3 Format de description des patrons de conception

4 Catalogue et description des patron de conception GoF

1 Introduction

2 Classification des patrons

- Les rôles : créateurs, structurels ou comportementaux
- Les domaines : objets ou classes
- Rôles et domaines

3 Format de description des patrons de conception

4 Catalogue et description des patron de conception GoF

5 Les patrons les plus courants

- Les créateurs : Fabrication, Fabrique Abstraite
- Les structurels : Façade
- Les comportementaux : Itérateur, Observateur

1 Introduction

2 Classification des patrons

- Les rôles : créateurs, structurels ou comportementaux
- Les domaines : objets ou classes
- Rôles et domaines

3 Format de description des patrons de conception

4 Catalogue et description des patron de conception GoF

5 Les patrons les plus courants

- Les créateurs : Fabrication, Fabrique Abstraite
- Les structurels : Façade
- Les comportementaux : Itérateur, Observateur



6 D'autres patrons moins fréquents

- Les créateurs : Singleton
- Les structurels : Adaptateur, Composite, Procuration
- Les comportementaux : Commande, Stratégie, Visiteur

1 Introduction

2 Classification des patrons

- Les rôles : créateurs, structurels ou comportementaux
- Les domaines : objets ou classes
- Rôles et domaines

3 Format de description des patrons de conception

4 Catalogue et description des patron de conception GoF

5 Les patrons les plus courants

- Les créateurs : Fabrication, Fabrique Abstraite
- Les structurels : Façade
- Les comportementaux : Itérateur, Observateur

6 D'autres patrons moins fréquents

- Les créateurs : Singleton
- Les structurels : Adaptateur, Composite, Procuration
- Les comportementaux : Commande, Stratégie, Visiteur

7 Conclusion

- 1** Introduction
- 2** Classification des patrons
 - Les rôles : créateurs, structurels ou comportementaux
 - Les domaines : objets ou classes
 - Rôles et domaines
- 3** Format de description des patrons de conception
- 4** Catalogue et description des patron de conception GoF
- 5** Les patrons les plus courants
 - Les créateurs : Fabrication, Fabrique Abstraite
 - Les structurels : Façade
 - Les comportementaux : Itérateur, Observateur
- 6** D'autres patrons moins fréquents
 - Les créateurs : Singleton
 - Les structurels : Adaptateur, Composite, Procuration
 - Les comportementaux : Commande, Stratégie, Visiteur
- 7** Conclusion
- 8** Références Bibliographiques

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron de
conception
GoF

Les patrons
les plus
courants

D'autres
patrons
moins
fréquents

Conclusion

Références
Bibli-
ographiques

- La conception orientée objets est un art difficile
- Des objectifs parfois contradictoires :
 - encapsuler les données sans empêcher l'accès,
 - trouver le bon niveau de granularité des objets,
 - limiter les dépendances entre objets (faible couplage),
 - fournir des objets simples à utiliser,
 - offrir une implémentation performante.
- Une conception, à la fois, ***réutilisable, extensible, adaptable, performante*** est quelque chose de très difficile à faire



Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron de
conception
GoF

Les patrons
les plus
courants

D'autres
patrons
moins
fréquents

Conclusion

Références
Bibli-
ographiques

● La modélisation d'applications :

- est un processus complexe,
- requiert de l'expérience avant de devenir expert,
- soulève des problèmes de conceptions récurrents

Influence de l'architecture

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron de
conception
GoF

Les patrons
les plus
courants

D'autres
patrons
moins
fréquents

Conclusion

Références
Bibli-
ographiques

- **Christopher Alexander** : architecte à l'origine d'une théorie sur la conception architecturale s'appuyant sur des motifs (modèles, patrons, *patterns*) de conception :

Chaque motif décrit un problème que l'on rencontre sans cesse dans notre environnement ; puis il décrit le cœur de la solution à ce problème d'une telle façon qu'on peut utiliser la solution des millions de fois, sans jamais la mettre en œuvre de la même façon. "C. Alexander, *A Pattern Language*, 1977"



- un motif décrit *un problème récurrent*,
- un motif décrit *le noyaux d'une solution*,
- un motif peut engendrer *plusieurs conceptions distinctes*.

Qu'est-ce qu'un patron de conception ?

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron
de conception
GoF

Les patrons
les plus
courants

D'autres
patrons
moins
fréquents

Conclusion

Références
Bibli-
ographiques

Quatre éléments essentiels (d'après *Christopher Alexander*):

- ① **Le nom** : décrit en quelques mots le problème de conception traités, ses problèmes et ses solutions.
- ② **Le problème** : décrit les situations dans lesquelles le modèle s'applique (le sujet traité et le contexte).
- ③ **La solution** : décrit les éléments qui constituent la conception, les relations entre eux, leur coopération
- ④ **Les conséquences** : sont les effets résultants de sa mise en œuvre, les compromis à faire

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron
de conception
GoF

Les patrons
les plus
courants

D'autres
patrons
moins
fréquents

Conclusion

Références
Bibli-
ographiques

- 23 patrons de conception (design patterns) décrits dans un livre décisif en 1995 :



Design Patterns : Elements of Reusable Object Oriented Software, E. Gamma, R. Helm, R. Johnson, J. Vlissides (Gang of Four, GoF), Addison Wesley



- Les patrons de conception proposent :
 - des **solutions standard et génériques** à des problèmes de programmation et de conception courants,
 - un **vocabulaire partagé** facilitant la description d'une conception



1 Introduction

2 Classification des patrons

- Les rôles : créateurs, structurels ou comportementaux
- Les domaines : objets ou classes
- Rôles et domaines

3 Format de description des patrons de conception

4 Catalogue et description des patron de conception GoF

5 Les patrons les plus courants

- Les créateurs : Fabrication, Fabrique Abstraite
- Les structurels : Façade
- Les comportementaux : Itérateur, Observateur

6 D'autres patrons moins fréquents

- Les créateurs : Singleton
- Les structurels : Adaptateur, Composite, Procuration
- Les comportementaux : Commande, Stratégie, Visiteur

7 Conclusion

8 Références Bibliographiques

Patrons Créateurs (Creational Patterns)

Introduction

Classification
des patrons

Les rôles :
créateurs,
structurels
ou
comporté-
mentaux

Les
domaines :
objets ou
classes

Rôles et
domaines

Format de
description
des patrons
de
conception

Catalogue et
description
des patron de
conception
GoF

Les patrons

Création d'instance sans instanciation explicite (avec `new`) : les instructions gérant l'instanciation sont isolées pour rendre le reste du code indépendant du type d'objets créés

Patrons créateurs

- Singleton,
- Monteur,
- Prototype,
- Fabrique Abstraite,
- Fabrication.

Patrons Structurels (Structural Patterns)

Introduction

Classification
des patrons

Les rôles :
créateurs,
structurels
ou
comporte-
mentaux

Les
domaines :
objets ou
classes

Rôles et
domaines

Format de
description
des patrons
de
conception

Catalogue et
description
des patron de
conception
GoF

Les patrons

Mise en place de relations de composition entre objets et classes
afin de former des structures plus vastes:

Patrons Structurel (Structural Patterns)

- Décorateur,
- Proxy,
- Façade,
- Composite,
- Poids-Mouche,
- Pont,
- Adapteur.

Introduction

Classification
des patrons

Les rôles :
créateurs,
structurels
ou
comporte-
mentaux

Les
domaines :
objets ou
classes

Rôles et
domaines

Format de
description
des patrons
de
conception

Catalogue et
description
des patron de
conception
GoF

Les patrons

Définition de l'interconnexion d'objets et de classes en vue de les faire communiquer et collaborer :

Patrons comportementaux

- Commande,
- Itérateur,
- Médiateur,
- Patron de Méthode,
- Visiteur,
- Chaîne de Responsabilité,
- État,
- Observateur,
- Interprète,
- Memento,
- Stratégie.

1 Introduction

2 Classification des patrons

- Les rôles : créateurs, structurels ou comportementaux
- **Les domaines : objets ou classes**
- Rôles et domaines

3 Format de description des patrons de conception

4 Catalogue et description des patron de conception GoF

5 Les patrons les plus courants

- Les créateurs : Fabrication, Fabrique Abstraite
- Les structurels : Façade
- Les comportementaux : Itérateur, Observateur

6 D'autres patrons moins fréquents

- Les créateurs : Singleton
- Les structurels : Adaptateur, Composite, Procuration
- Les comportementaux : Commande, Stratégie, Visiteur

7 Conclusion

8 Références Bibliographiques

Domaines des Patrons

Introduction

Classification
des patrons

Les rôles :
créateurs,
structurels
ou
comportementaux

Les
domaines :
objets ou
classes

Rôles et
domaines

Format de
description
des patrons
de
conception

Catalogue et
description
des patron de
conception
GoF

Les patrons

Le domaine précise si un patron s'applique à des classes ou à des objets :

Les patrons de classe

- définissent des relations entre classes grâce à l'*héritage*
- les relations sont définies à *la compilation* (elles sont statiques)

Les patrons d'objets

- décrivent des relations entre objets, principalement à l'aide de la *composition*
- les relations sont établies à *l'exécution* (elles sont dynamiques) 

1 Introduction

2 Classification des patrons

- Les rôles : créateurs, structurels ou comportementaux
- Les domaines : objets ou classes
- Rôles et domaines**

3 Format de description des patrons de conception

4 Catalogue et description des patron de conception GoF

5 Les patrons les plus courants

- Les créateurs : Fabrication, Fabrique Abstraite
- Les structurels : Façade
- Les comportementaux : Itérateur, Observateur

6 D'autres patrons moins fréquents

- Les créateurs : Singleton
- Les structurels : Adaptateur, Composite, Procuration
- Les comportementaux : Commande, Stratégie, Visiteur

7 Conclusion

8 Références Bibliographiques

[Introduction](#)

Classification des patrons

Les rôles : créateurs, structurels ou comportementaux

Les domaines : objets ou classes

Rôles et domaines

Format de description des patrons de conception

Catalogue et description des patron de conception GoF

Les patrons

Les patrons créateurs

- **de classes** : déléguent certaines parties de la construction d'objets aux *sous-classes*.
- **d'objets** : déléguent certaines parties de la construction d'objets à *d'autres objets*.

Introduction

Classification
des patrons

Les rôles :
créateurs,
structurels
ou
comporte-
mentaux

Les
domaines :
objets ou
classes

Rôles et
domaines

Format de
description
des patrons
de
conception

Catalogue et
description
des patron de
conception
GoF

Les patrons

Les patrons structurels

- **de classes** : utilisent l'*héritage* pour composer des *classes*.
- **d'objets** : décrivent des façons d'*assembler des objets*.

[Introduction](#)[Classification des patrons](#)[Les rôles : créateurs, structurels ou comportementaux](#)[Les domaines : objets ou classes](#)[Rôles et domaines](#)[Format de description des patrons de conception](#)[Catalogue et description des patron de conception GoF](#)[Les patrons](#)

Les patrons comportementaux

- **de classes** : utilisent l'*héritage* pour définir des *algorithmes* et des *structures de contrôle*.
- **d'objets** : décrivent la façon dont un *groupe d'objets collaborent pour réaliser une tâche* qu'un objet ne peut réaliser seul.

Un format de description unique(1/4)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron de
conception
GoF

Les patrons
les plus
courants

D'autres
patrons
moins
fréquents

Conclusion

Références
Bibli-
ographiques

- Les **notations graphiques** sont souvent **insuffisantes** pour décrire les patrons de conception
 - elles ne décrivent que les relations entre classes et objets
 - elles ne décrivent pas les orientations, les alternatives et les compromis qui conduisent à l'utiliser
 - elles ne permettent pas de décrire des exemples concrets
- Le GoF propose un **format unique de description** des patrons de conception

Un format de description unique (2/4)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron de
conception
GoF

Les patrons
les plus
courants

D'autres
patrons
moins
fréquents

Conclusion

Références
Bibli-
ographiques

- **Nom du modèle et classification** : le nom d'un patron doit tenter de décrire son principe. Un bon nom est essentiel à la communication.
- **Intentions** :
 - que fait le patron ?
 - quels sont sa raison d'être et son but ?
 - quel problème de conception concerne-t-il ?
- **Alias** : d'autres nom du patron (s'ils existent)
- **Motivation** : un scénario illustratif de l'utilisation du patron, qui montre comment il résoud le problème.

Un format de description unique (3/4)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron de
conception
GoF

Les patrons
les plus
courants

D'autres
patrons
moins
fréquents

Conclusion

Références
Bibli-
ographiques

● Indications d'utilisation :

- quels sont les cas qui justifient son utilisation ?
- dans quelles situation devrait on utiliser le patron ?
- comment reconnaître ses situations ?

● Structure : une représentation graphique des classes du patron (généralement en UML) .

● Constituants : la liste des classes et objets participants au patron.

● Collaborations : les classes et objets intervenant dans le patron et leurs responsabilités.

● Conséquences : quelles sont les conséquences (positives et négatives) de l'utilisation du patron ? quels sont les compromis qu'impliquent l'utilisation du patron ?

Un format de description unique (4/4)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron
de
conception
GoF

Les patrons
les plus
courants

D'autres
patrons
moins
fréquents

Conclusion

Références
Bibli-
ographiques

- **Implémentation** : quels sont les pièges à éviter, les astuces et techniques à connaître ? y a-t-il des solutions typiques pour certains langages.
- **Exemples de code** : fragments de code illustrant des implémentations (C++, Smalltalk, Java, ...) possibles du patron.
- **Utilisations remarquables** : exemples d'utilisation du patron dans des systèmes existants.
- **Modèles apparentés** : quels patrons de conception sont en relation avec celui traité ?

Double classification

Introduction

Classification des patrons

Format de description des patrons de conception

Catalogue et description des patron de conception GoF

Les patrons les plus courants

D'autres patrons moins fréquents

Conclusion

Références Bibliographiques

Type de Patron			
	<i>Créateurs</i>	<i>Structurels</i>	<i>Comportementaux</i>
Classes	Fabrication	Adaptateur	Interprète Patron de Méthode
Objets 	Fabrique Abstraite Monteur Prototype Singleton	Adaptateur Composite Décorateur Façade Procuration Poids-Mouche Pont	Chaîne de Responsabilité Commandes Itérateur Médiateur Memento Visiteur Etat Observateur Stratégie

[Introduction](#)[Classification des patrons](#)[Format de description des patrons de conception](#)[Catalogue et description des patron de conception GoF](#)[Les patrons les plus courants](#)[D'autres patrons moins fréquents](#)[Conclusion](#)[Références Bibliographiques](#)

- **Fabrication (*Factory Method*)** - *classe, créateur* : définit une interface pour la création d'un objet, tout en laissant à des sous-classes le choix de la classe à instancier.
- **Fabrique Abstraite (*Abstract Factory*)** - *objet, créateur* : interface pour créer des familles d'objets apparentés sans avoir à spécifier leurs classes concrètes.
- **Monteur (*Builder*)** - *objet, créateur* : dissocie la construction et la représentation d'un objet complexe : le même procédé de construction peut ainsi générer des représentations différentes.

Création (2/2)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron
de conception
GoF

Les patrons
les plus
courants

D'autres
patrons
moins
fréquents

Conclusion

Références
Bibli-
ographiques

- **Prototype (*Prototype*)** - *objet, créateur* : construction d'objets par clonage d'un objet prototype.
- **Singleton (*Singleton*)** - *objet, créateur* : garantit qu'une classe n'a qu'une seule instance et fournit un moyen d'obtenir cette unique instance.

Structuration (1/2)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron de
conception
GoF

Les patrons
les plus
courants

D'autres
patrons
moins
fréquents

Conclusion

Références
Bibli-
ographiques

- **Adaptateur (*Adapter*)** - *classe, structurel* : convertit l'interface d'une classe en une interface distincte conforme à l'attente de l'utilisateur. Permet à des classes "incompatibles" de travailler ensemble.
- **Pont (*Bridge*)** - *objet, structurel* : découple une abstraction de son implémentation afin que les deux puissent être modifiées indépendamment.
- **Composite (*Composite*)** - *objet, structurel* : organise des objets en structure arborescente. Ainsi un utilisateur manipule un objet individuel et une structure d'objets de la même manière.
- **Décorateur (*Decorator*)** - *objet, structurel* : attache de responsabilités supplémentaires à un objet de façon dynamique.

Structuration (2/2)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron
de
conception
GoF

Les patrons
les plus
courants

D'autres
patrons
moins
fréquents

Conclusion

Références
Bibli-
ographiques

- **Façade (*Facade*)** - *objet, structurel* : fournit une interface unifiée pour un ensemble d'interface d'un sous système. Façade définit une interface de plus haut niveau, qui facilite l'utilisation du sous-système.
- **Poids-Mouche (*Fly-Weight*)** - *objet, structurel* : assure, en mode partagé, le support efficace d'un grand nombre d'objets à fine granularité.
- **Procuration (*Proxy*)** - *objet, structurel* : fournit un remplaçant ou un mandataire pour un autre objet afin d'en contrôler l'accès.

[Introduction](#)[Classification des patrons](#)[Format de description des patrons de conception](#)[Catalogue et description des patron de conception GoF](#)[Les patrons les plus courants](#)[D'autres patrons moins fréquents](#)[Conclusion](#)[Références Bibliographiques](#)

- **Patron de Méthode (*Template Method*)** - *classe, comportemental* : définit le squelette de l'algorithme d'une opération, en déléguant le traitement de certaines étapes à des sous classes. Le patron de méthode permet aux sous classes de redéfinir certaines étapes de l'algorithme sans en modifier la structure globale.
- **Interprète (*Interpreter*)** - *classe, comportemental* : définit une représentation de la grammaire d'un langage donné, ainsi qu'un interprète utilisant cette représentation pour analyser la syntaxe du langage.
- **Chaîne de Responsabilité (*Chain of Responsibility*)** - *objet, comportemental* : permet à plusieurs objets de prendre en charge une requête en cascade.

Comportement (2/4)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron
de
conception
GoF

Les patrons
les plus
courants

D'autres
patrons
moins
fréquents

Conclusion

Références
Bibli-
ographiques

- **Commande (Command)** - *objet, comportemental* : encapsule une requête dans un objet.
- **Itérateur (Iterator)** - *objet, comportemental* : fournit un moyen de donner accès aux éléments d'un objet de type aggrégat sans en révéler son implémentation.
- **Médiateur (Mediator)** - *objet, comportemental* : définit un objet qui encapsule les modalités d'interaction de divers objets.

Comportement (3/4)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron
de conception
GoF

Les patrons
les plus
courants

D'autres
patrons
moins
fréquents

Conclusion

Références
Bibli-
ographiques

- **Memento (*Memento*)** - *objet, comportemental* : acquiert et délivre à l'extérieur une information sur l'état interne d'un objet, pour qu'il puisse être rétablit ultérieurement dans cet état.
- **Observateur (*Observer*)** - *objet, comportemental* : définit une relation un à plusieurs, de façon que, lorsqu'un objet change d'état, les autres, qui en dépendent, puissent être notifiés et modifiés en conséquence.
- **État (*State*)** - *objet, comportemental* : permet à un objet de changer de comportement lorsque son état interne change (il paraîtra changer de classe).

[Introduction](#)[Classification des patrons](#)[Format de description des patrons de conception](#)[Catalogue et description des patron de conception GoF](#)[Les patrons les plus courants](#)[D'autres patrons moins fréquents](#)[Conclusion](#)[Références Bibliographiques](#)

- **Stratégie (*Strategy*)** - *objet, comportemental* : Définit une famille d'algorithmes, encapsule chacun d'entre eux et les rend interchangeables. Le patron stratégie permet de modifier un algorithme indépendamment de ses clients.
- **Visiteur (*Visitor*)** - *objet, comportemental* : représente une opération à effectuer sur les éléments d'une structure d'objet. Le patron visiteur permet de définir une nouvelle opération sans modifier les classes des éléments sur lesquels il opère.

- 1** Introduction
- 2** Classification des patrons
 - Les rôles : créateurs, structurels ou comportementaux
 - Les domaines : objets ou classes
 - Rôles et domaines
- 3** Format de description des patrons de conception
- 4** Catalogue et description des patron de conception GoF
- 5** Les patrons les plus courants
 - **Les créateurs** : Fabrication, Fabrique Abstraite
 - Les structurels : Façade
 - Les comportementaux : Itérateur, Observateur
- 6** D'autres patrons moins fréquents
 - Les créateurs : Singleton
 - Les structurels : Adaptateur, Composite, Procuration
 - Les comportementaux : Commande, Stratégie, Visiteur
- 7** Conclusion
- 8** Références Bibliographiques

Fabrication - Factory Method (1/5)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron de
conception
GoF

Les patrons
les plus
courants

Les
créateurs :
Fabrication,
Fabrique
Abstraite

Les
structuraux :
Façade

Les
comporte-

Catégories

Créateur, Classe

Alias

Constructeur Virtuel

Intention

Définit une **interface** pour la création d'un objet, mais en laissant aux **sous-classes** le choix des classes à instancier. Le patron *Fabrication* permet à une classe de **déléguer l'instanciation à des sous-classes**

Fabrication - Factory Method (2/5)

Introduction

Classification des patrons

Format de description des patrons de conception

Catalogue et description des patron de conception GoF

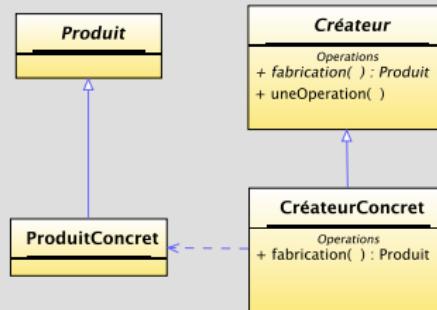
Les patrons les plus courants

Les créateurs : Fabrication, Fabrique Abstraite

Les structuraux : Façade

Les comportementaux

Structure



Collaborations

- Le *Créateur* confie la définition de la fabrication à ses sous-classes afin qu'il renvoie une instance du *ProduitConcret* approprié.

Fabrication - Factory Method (3/5)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron de
conception
GoF

Les patrons
les plus
courants

Les
créateurs :
Fabrication,
Fabrique
Abstraite

Les
structurels :
Façade

Les
comportem-

Un exemple

On souhaite écrire une application permettant d'afficher les caractéristiques, au choix, d'un rectangle ou d'une ellipse de dimensions aléatoires.

Fabrication - Factory Method (4/5)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron de
conception
GoF

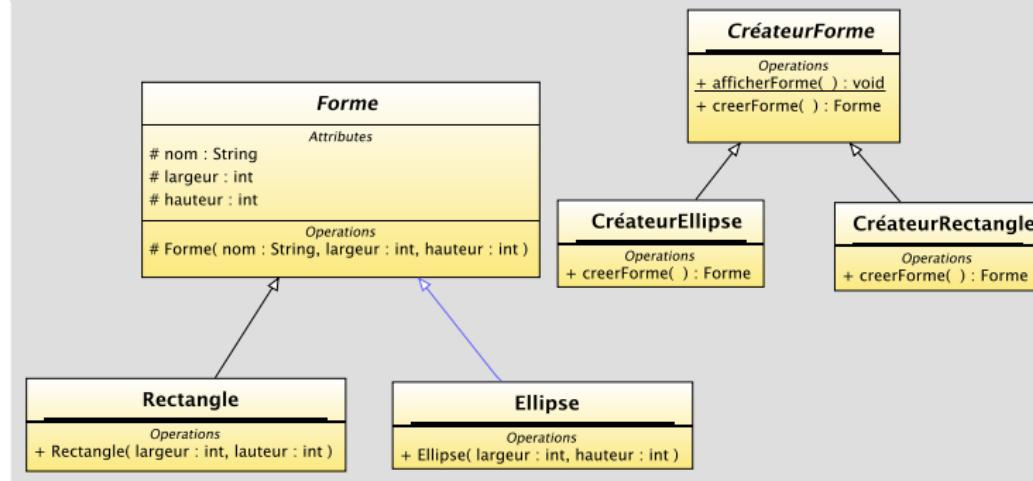
Les patrons
les plus
courants

Les
créateurs :
Fabrication,
Fabrique
Abstraite

Les
structuraux :
Façade

Les
comporte-

Exemple de Fabrication



Fabrication - Factory Method (4/5)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron de
conception
GoF

Les patrons
les plus
courants

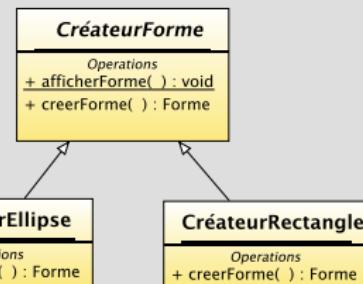
Les
créateurs :
Fabrication,
Fabrique
Abstraite

Les
structuraux :
Façade

Les
comporta-

Exemple de Fabrication

```
public abstract class CréeleurForme {  
    protected Random alea=new Random();  
  
    public abstract Forme créerForme();  
  
    public static void afficherForme(String type){  
        CréeleurForme fabrique ;  
        if (type.equals("rectangle")) {  
            fabrique=new CréeleurRectangle();  
        } else if (type.equals("ellipse")) {  
            fabrique=new CréeleurEllipse();  
        } else {  
            System.err.println("Cette forme n'existe pas");  
            return;  
        }  
        Forme f = fabrique.créerForme();  
        f.afficheInfo();  
    }  
}
```



Fabrication - Factory Method (4/5)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron de
conception
GoF

Les patrons
les plus
courants

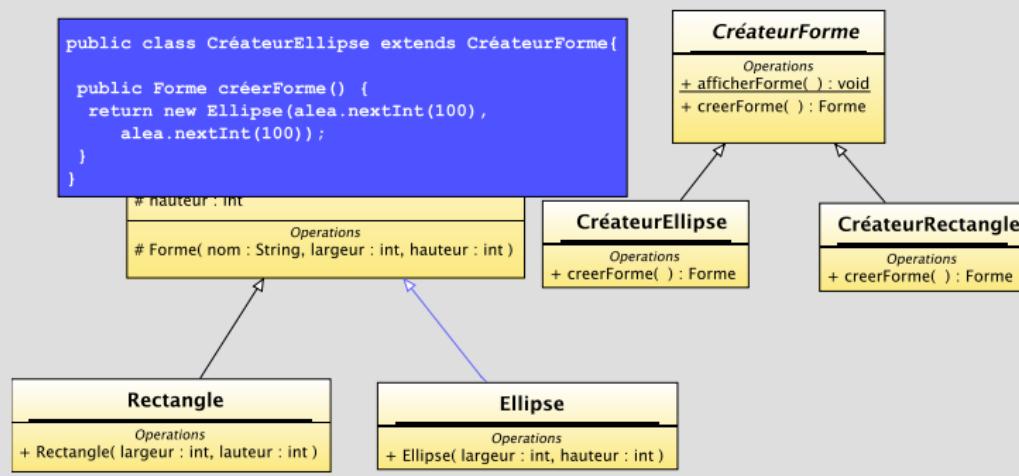
Les
créateurs :
Fabrication,
Fabrique
Abstraite

Les
structuraux :
Façade

Les
comporte-

Exemple de Fabrication

```
public class CréeleurEllipse extends CréeleurForme{  
  
    public Forme créerForme() {  
        return new Ellipse(alea.nextInt(100),  
                          alea.nextInt(100));  
    }  
}
```



Fabrication - Factory Method (4/5)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron de
conception
GoF

Les patrons
les plus
courants

Les
créateurs :
Fabrication,
Fabrique
Abstraite

Les
structuraux :
Façade

Les
comporta-

Exemple de Fabrication

```
public class CréeleurEllipse extends CréeleurForme{  
  
    public Forme créerForme() {  
        return new Ellipse(alea.nextInt(100),  
                           alea.nextInt(100));  
    }  
}
```

Forme(nom : String, largeur : int, hauteur : int)
Operations

CréeleurForme
Operations
+ afficherForme() : void
+ créerForme() : Forme

CréeleurEllipse
Operations
+ créerForme() : Forme

CréeleurRectangle
Operations
+ créerForme() : Forme

```
public class CréeleurRectangle extends CréeleurForme {  
  
    public Forme créerForme() {  
        return new Rectangle(alea.nextInt(100),  
                            alea.nextInt(100));  
    }  
}
```

Forme(nom : String, largeur : int, hauteur : int)

Fabrication - Factory Method (4/5)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron de
conception
GoF

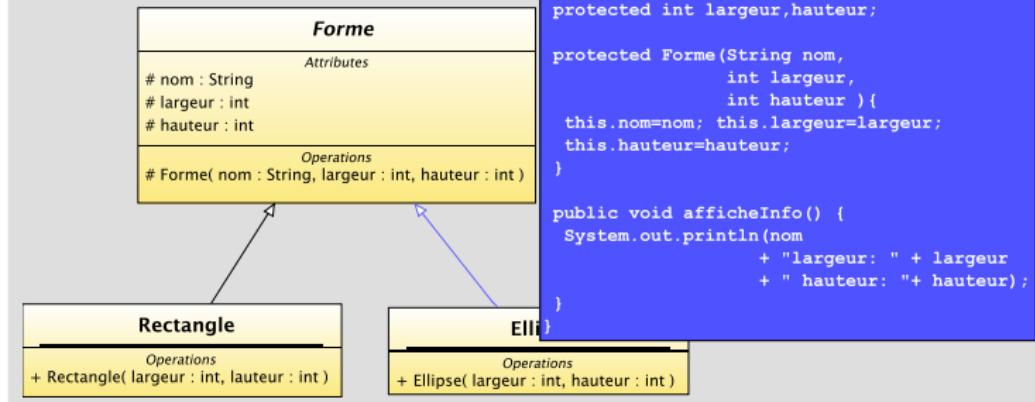
Les patrons
les plus
courants

Les
créateurs :
Fabrication,
Fabrique
Abstraite

Les
structuraux :
Façade

Les
comporte-

Exemple de Fabrication



Fabrication - Factory Method (4/5)

Introduction

Classification des patrons

Format de description des patrons de conception

Catalogue et description des patron de conception GoF

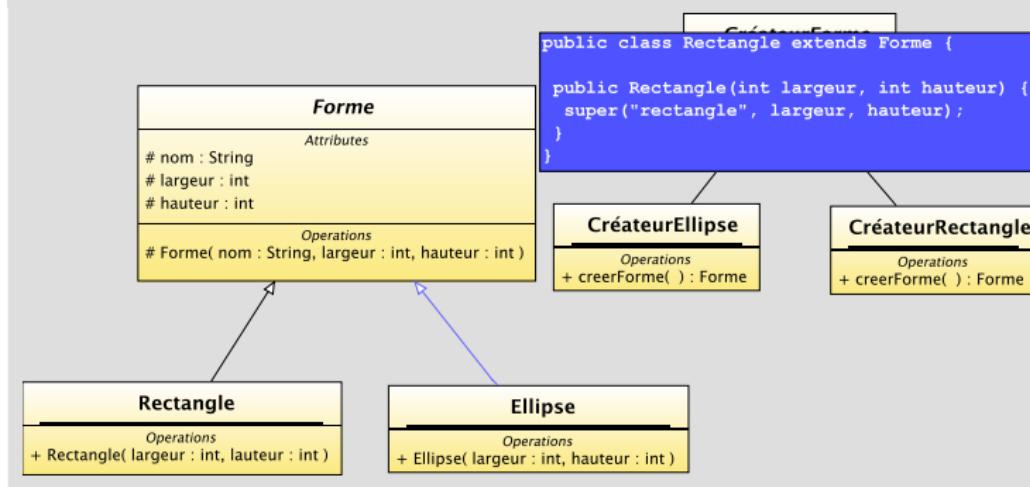
Les patrons les plus courants

Les créateurs : Fabrication, Fabrique Abstraite

Les structuraux : Façade

Les comportementaux

Exemple de Fabrication



Fabrication - Factory Method (4/5)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron de
conception
GoF

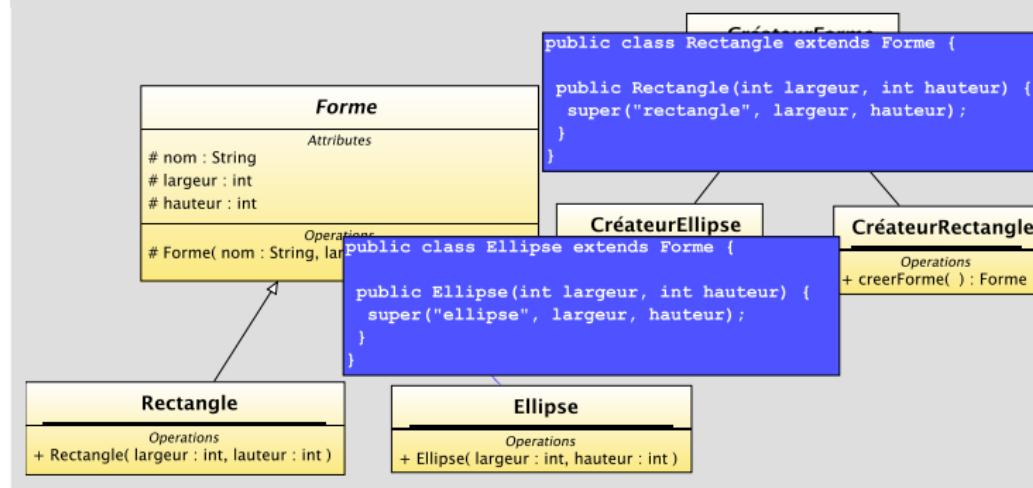
Les patrons
les plus
courants

Les
créateurs :
Fabrication,
Fabrique
Abstraite

Les
structuraux :
Façade

Les
comporte-

Exemple de Fabrication



[Introduction](#)[Classification des patrons](#)[Format de description des patrons de conception](#)[Catalogue et description des patron de conception GoF](#)[Les patrons les plus courants](#)[Les créateurs : Fabrication, Fabrique Abstraite](#)[Les structures : Façade](#)[Les comportements](#)

Indications d'utilisation

La Fabrication doit être utilisée lorsque :

- une classe ne peut prévoir la classe des objets qu'elle aura à créer,
- une classe attend de ses sous-classes qu'elles spécifient les objets qu'elles créent.

Conséquences

- La Fabrication dispense d'avoir à incorporer au code des classes spécifiques de l'application. Le code ne concerne que la classe *Produit* et peut donc fonctionner avec tout *ProduitConcret*.

Fabrique Abstraite - Abstract Factory (1/8)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron de
conception
GoF

Les patrons
les plus
courants

Les
créateurs :
Fabrication,
Fabrique
Abstraite

Les
structuels :
Façade

Les
comporte-

Catégories

Créateur, Objet

Alias

Kit

Intention

La fabrique abstraite fournit une interface pour la création de familles d'objets apparentés ou interdépendants, sans qu'il soit nécessaire de spécifier leur classes concrètes.

Fabrique Abstraite - Abstract Factory (2/8)

Introduction

Classification des patrons

Format de description des patrons de conception

Catalogue et description des patron de conception GoF

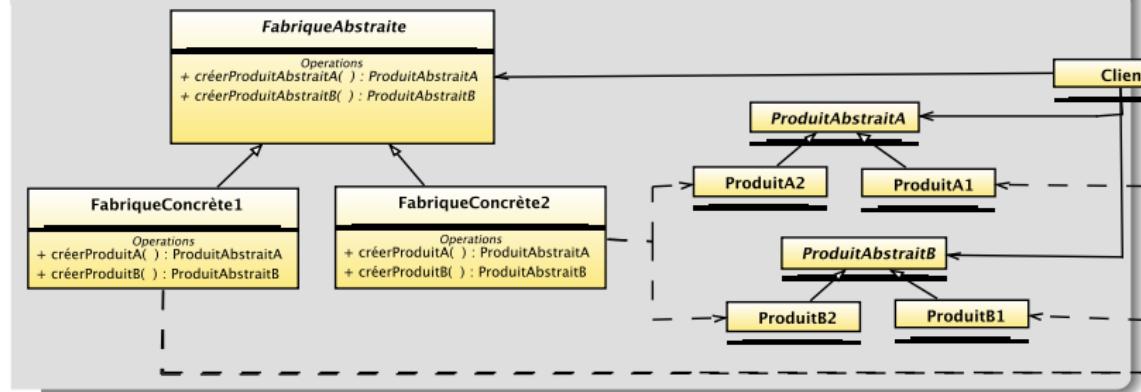
Les patrons les plus courants

Les créateurs : Fabrication, Fabrique Abstraite

Les structurales : Façade

Les comportementales

Structure



[Introduction](#)[Classification des patrons](#)[Format de description des patrons de conception](#)[Catalogue et description des patron de conception GoF](#)[Les patrons les plus courants](#)[Les créateurs : Fabrication, Fabrique Abstraite](#)[Les structurels : Façade](#)[Les comportementaux](#)

Collaborations

- une instance unique de *FabriqueConcrète* créée à l'exécution. Elle se charge de la création des *Produits* concrets. Pour créer des *Produits* concrets différents il faut des *FabriqueConcrètes* différentes.
- Une *FabriqueAbstraite* délègue la création des instances de *Produit* à sa sous-classe *FabriqueConcrète*.

[Introduction](#)[Classification des patrons](#)[Format de description des patrons de conception](#)[Catalogue et description des patron de conception GoF](#)[Les patrons les plus courants](#)[Les créateurs : Fabrication, Fabrique Abstraite](#)[Les structurels : Façade](#)[Les comportementaux](#)

Exemple

On souhaite écrire une application capable de changer son “look and feel” en fonction du choix de l’utilisateur. Elle devra pouvoir créer, au choix et de manière transparente, des boutons avec un “look and feel” Unix ou Windows. Cette application pourrait être généralisée pour gérer tous les gadget graphiques (menus, fenêtres, ascenseurs, boîtes de dialogues...) de la même manière.

Fabrique Abstraite - Abstract Factory (5/8)

Introduction

Classification des patrons

Format de description des patrons de conception

Catalogue et description des patron de conception GoF

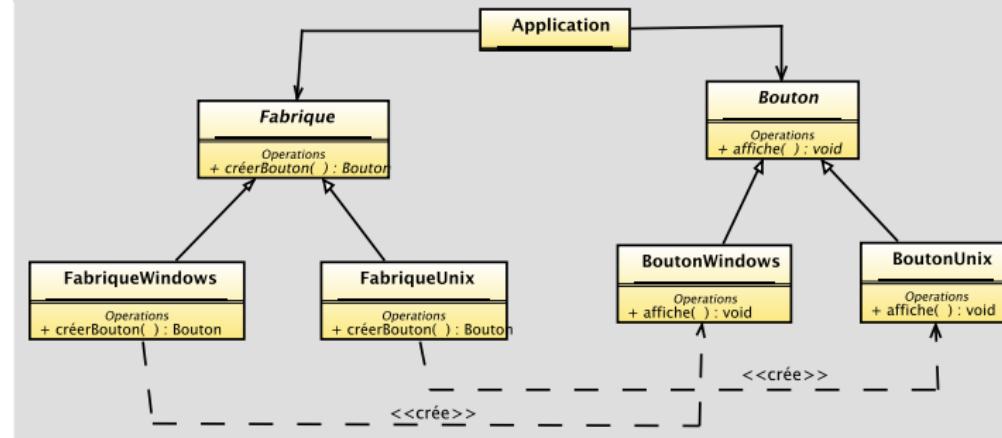
Les patrons les plus courants

Les créateurs : Fabrication, Fabrique Abstraite

Les structurals : Façade

Les comportementaux

Exemple de Fabrique Abstraite



Fabrique Abstraite - Abstract Factory (5/8)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron de
conception
GoF

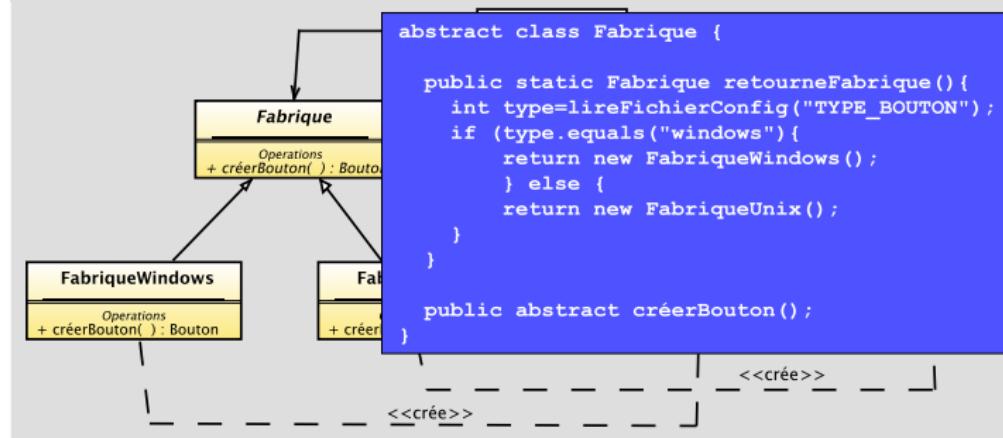
Les patrons
les plus
courants

Les
créateurs :
Fabrication,
Fabrique
Abstraite

Les
structuraux :
Façade

Les
comporta-

Exemple de Fabrique Abstraite



Fabrique Abstraite - Abstract Factory (5/8)

Introduction

Classification des patrons

Format de description des patrons de conception

Catalogue et description des patron de conception GoF

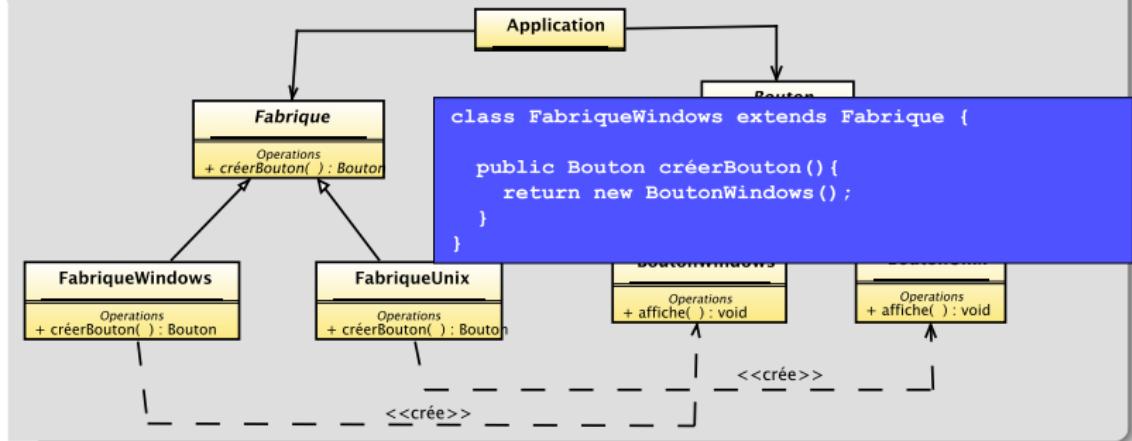
Les patrons les plus courants

Les créateurs : Fabrication, Fabrique Abstraite

Les structurals : Façade

Les comportementaux

Exemple de Fabrique Abstraite



Fabrique Abstraite - Abstract Factory (5/8)

Introduction

Classification des patrons

Format de description des patrons de conception

Catalogue et description des patron de conception GoF

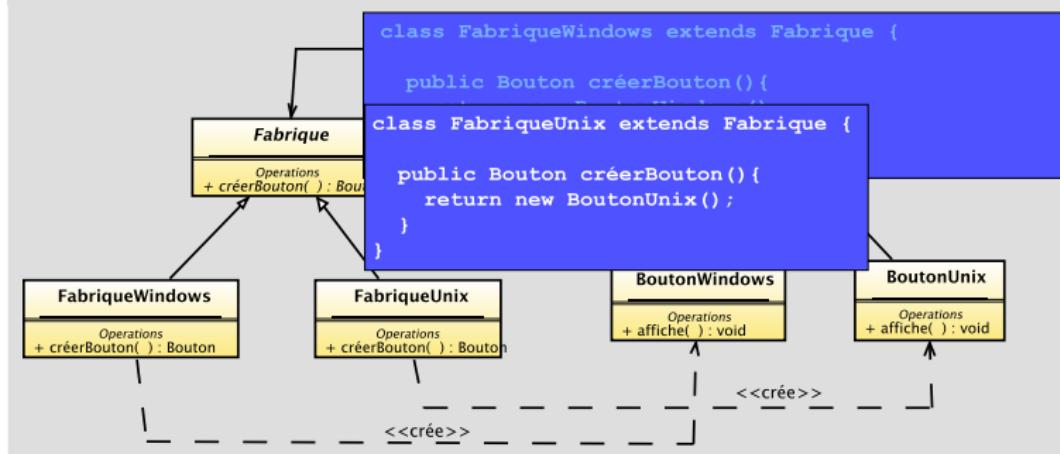
Les patrons les plus courants

Les créateurs : Fabrication, Fabrique Abstraite

Les structures : Façade

Les comportements

Exemple de Fabrique Abstraite



Fabrique Abstraite - Abstract Factory (5/8)

Introduction

Classification des patrons

Format de description des patrons de conception

Catalogue et description des patron de conception GoF

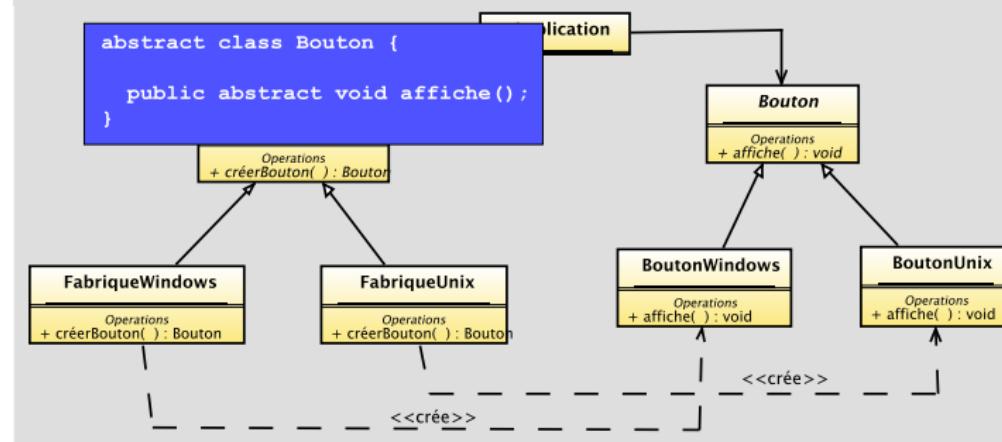
Les patrons les plus courants

Les créateurs : Fabrication, Fabrique Abstraite

Les structurals : Façade

Les comportementaux

Exemple de Fabrique Abstraite



Fabrique Abstraite - Abstract Factory (5/8)

Introduction

Classification des patrons

Format de description des patrons de conception

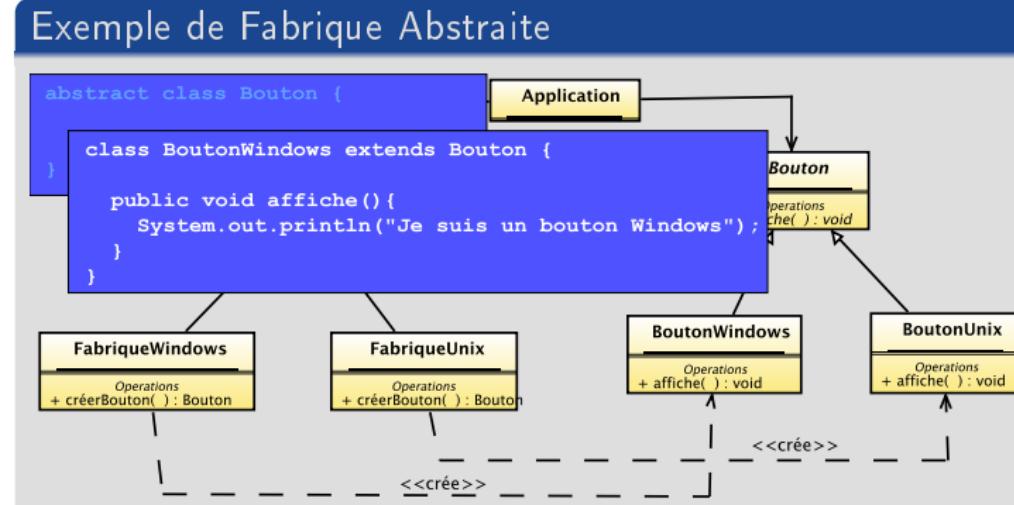
Catalogue et description des patron de conception GoF

Les patrons les plus courants

Les créateurs : Fabrication, Fabrique Abstraite

Les structurales : Façade

Les comportementaux



Fabrique Abstraite - Abstract Factory (5/8)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron de
conception
GoF

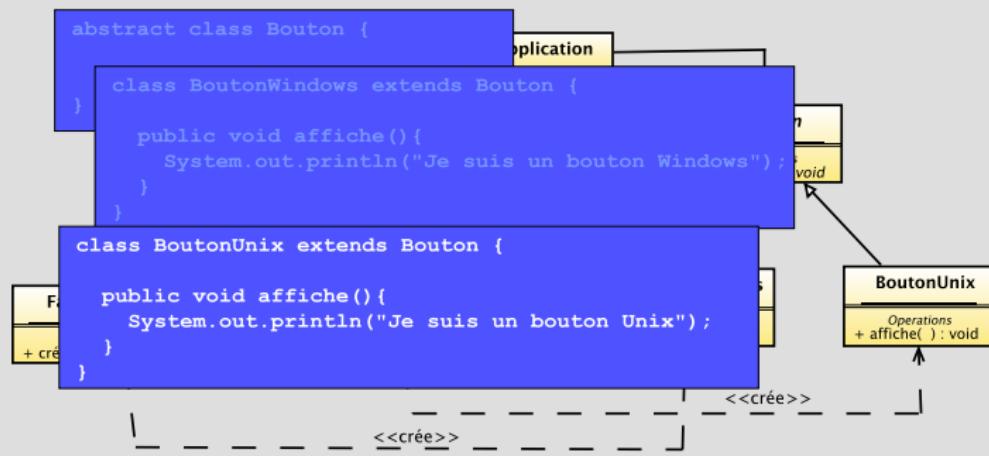
Les patrons
les plus
courants

Les
créateurs :
Fabrication,
Fabrique
Abstraite

Les
structuraux :
Façade

Les
comporta-

Exemple de Fabrique Abstraite



Fabrique Abstraite - Abstract Factory (5/8)

Introduction

Classification des patrons

Format de description des patrons de conception

Catalogue et description des patron de conception GoF

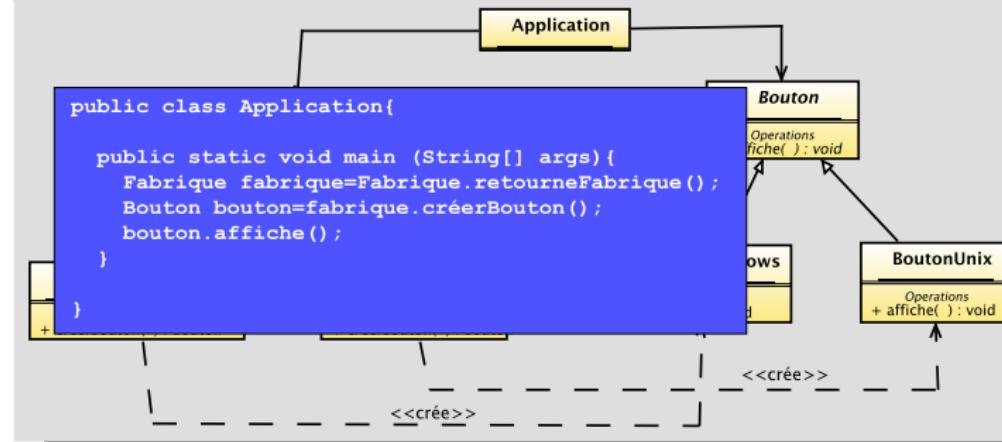
Les patrons les plus courants

Les créateurs : Fabrication, Fabrique Abstraite

Les structurales : Façade

Les comportementaux

Exemple de Fabrique Abstraite



Fabrique Abstraite- Abstract Factory (6/8)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron
de
conception
GoF

Les patrons
les plus
courants

Les
créateurs :
Fabrication,
Fabrique
Abstraite

Les
structuraux :
Façade

Les
comporta-

Indications d'utilisation

L'utilisation de la Fabrique Abstraite est recommandée lorsque :

- un système doit être indépendant de la façon dont ses produits ont été créés, combinés et représentés,
- un système doit être constitué d'une famille de produits parmi plusieurs,
- on souhaite renforcer le caractère de communauté d'une famille d'objets conçus pour être utilisés ensemble.

[Introduction](#)[Classification des patrons](#)[Format de description des patrons de conception](#)[Catalogue et description des patron de conception GoF](#)[Les patrons les plus courants](#)[Les créateurs : Fabrication, Fabrique Abstraite](#)[Les structures : Façade](#)[Les comportements](#)

Conséquences positives

La Fabrique Abstraite :

- **isole les classes concrètes** : les clients manipulent les classes à travers leurs interfaces abstraites. Les noms des classes produits sont isolés dans l'implémentation de la fabrique concrète et n'apparaissent pas dans le code.
- **facilite la substitution de familles de produits** : une classe de fabrique concrète n'apparaît, dans un code, qu'à l'endroit où elle est instanciée. Il est donc aisé de modifier la fabrique concrète utilisée par une application.
- **favorise le maintien de la cohérence entre les objets** : si les objets produit d'une même famille sont conçus pour travailler ensemble, il est important qu'une application utilise ceux d'une seule et même famille à la fois. La Fabrique Abstraite permet de satisfaire au mieux cette condition.

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron
de conception
GoF

Les patrons
les plus
courants

Les
créateurs :
Fabrication,
Fabrique
Abstraite

Les
structurels :
Façade

Les
comporte-

Conséquence négative

La Fabrique Abstraite :

- permet difficilement de gérer de nouveaux type de produits : faire fabriquer aux Fabriques Abstraites de nouveaux types de produits est difficile. Il faut pour cela modifier l'interface de la fabrique, ce qui implique de modifier la classe Fabrique Abstraite et toutes ses sous-classes.

Fabrique Abstraite Vs Fabrication

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron
de conception
GoF

Les patrons
les plus
courants

Les
créateurs :
Fabrication,
Fabrique
Abstraite

Les
structuraux :
Façade

Les
comporta-

Fabrique Abstraite

- Fournit une interface abstraite pour créer **une famille de produit**,
- Chaque sous classe de *FabriqueAbstraite* se charge de **créer une famille de produit**.

Fabrication

- Fournit une interface abstraite pour créer **un produit**
- Délègue l'instanciation à ses sous-classes

- 1 Introduction
- 2 Classification des patrons
 - Les rôles : créateurs, structurels ou comportementaux
 - Les domaines : objets ou classes
 - Rôles et domaines
- 3 Format de description des patrons de conception
- 4 Catalogue et description des patron de conception GoF
- 5 **Les patrons les plus courants**
 - Les créateurs : Fabrication, Fabrique Abstraite
 - **Les structurels : Façade**
 - Les comportementaux : Itérateur, Observateur
- 6 D'autres patrons moins fréquents
 - Les créateurs : Singleton
 - Les structurels : Adaptateur, Composite, Procuration
 - Les comportementaux : Commande, Stratégie, Visiteur
- 7 Conclusion
- 8 Références Bibliographiques

Façade (1/6)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron
de conception
GoF

Les patrons
les plus
courants

Les
créateurs :
Fabrication,
Fabrique
Abstraite

Les
structuraux :
Façade

Les
comporte-

Catégories

Structurel, Objet

Intention

Fournit une **interface unifiée** à l'ensemble des interfaces d'un sous-système. La façade fournit une **interface de plus haut niveau** qui rend le **sous système plus facile à utiliser**.

Façade (2/6)

Introduction

Classification des patrons

Format de description des patrons de conception

Catalogue et description des patron de conception GoF

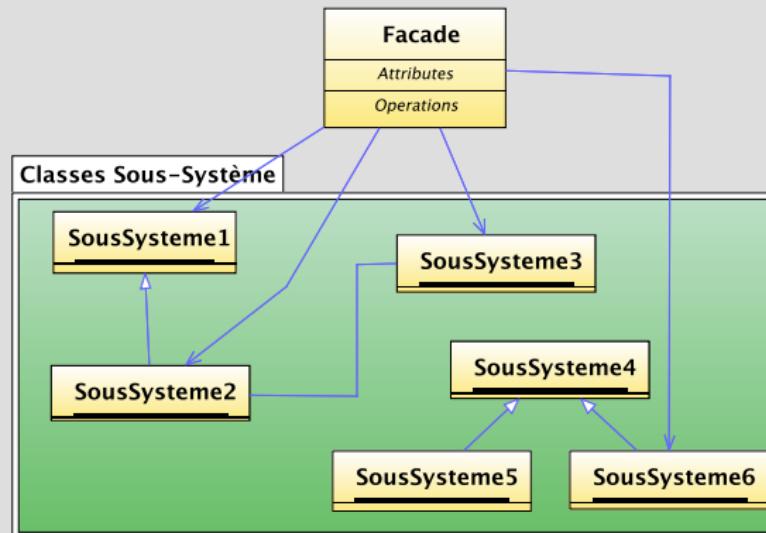
Les patrons les plus courants

Les créateurs : Fabrication, Fabrique Abstraite

Les structurales : Façade

Les comportementaux

Structure



Façade (3/6)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron
de conception
GoF

Les patrons
les plus
courants

Les
créateurs :
Fabrication,
Fabrique
Abstraite

Les
structuraux :
Façade

Les
comporta-

Collaborations

Les clients communiquent avec les sous-système en envoyant des requêtes à la façade qui répercute celles-ci aux objets appropriés du sous-système.

Les clients qui utilisent la façade n'ont pas à accéder directement aux objets de son sous-système.

Façade (4/6)

Introduction

Classification des patrons

Format de description des patrons de conception

Catalogue et description des patron de conception GoF

Les patrons les plus courants

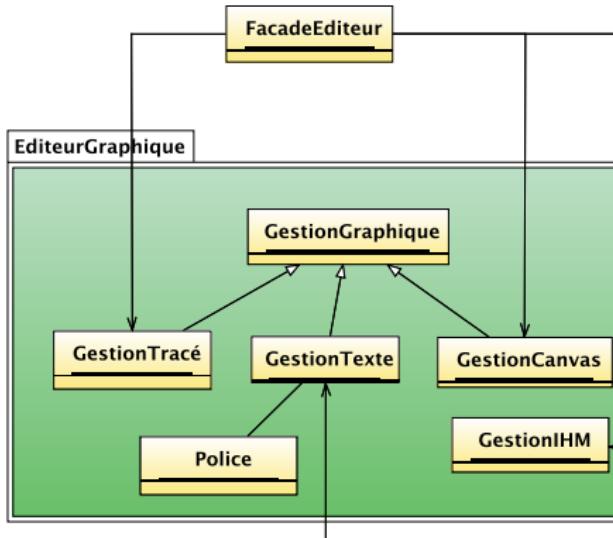
Les créateurs : Fabrication, Fabrique Abstraite

Les structures : Façade

Les comportements

Exemple de Façade

On souhaite faciliter la configuration d'un éditeur graphique à l'aide du patron de conception façade. Celui-ci se chargera d'aiguiller les demandes de configuration vers le gestionnaire approprié (gestion du texte, du canvas, du tracé, de l'IHM).



Façade (4/6)

Introduction

Classification des patrons

Format de description des patrons de conception

Catalogue et description des patron de conception GoF

Les patrons les plus courants

Les créateurs : Fabrication, Fabrique Abstraite

Les structures : Façade

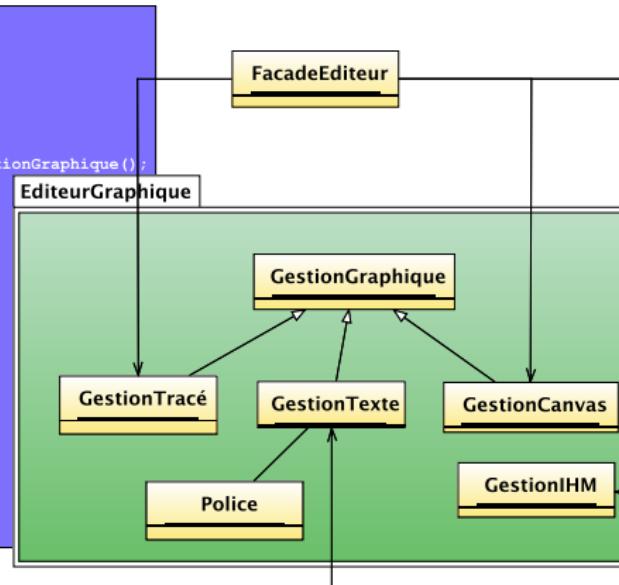
Les comportements

```
public class FacadeEditeur{
    GestionTracé traceMgr;
    GestionCanvas canvasMgr;
    GestionTexte texteMgr;
    GestionIHM ihmMgr

    public FacadeEditeur(){
        GestionGraphique graphicMgr= GestionGraphique.getGestionGraphique();
        traceMgr = graphicMgr.getTraceManager();
        canvasMgr = graphicMgr.getCanvasManager();
        texteMgr = graphicMgr.getTexteManager();
        ihmMgr= GestionIHM.getIHMManager();
    }

    public void changerPolice(String nom, int taille,
                             String style){
        Police police = new Police(nom,taille,style);
        texteMgr.setPolice(police);
    }

    public void setCouleur(Color couleur){
        traceMgr.setCouleur(couleur);
        canvasMgr.setCouleur(couleur);
        texteMgr.setCouleur(couleur);
    }
}
```



Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron de
conception
GoF

Les patrons
les plus
courants

Les
créateurs :
Fabrication,
Fabrique
Abstraite

Les
structurels :
Façade

Les
comporte-

Indications d'utilisation

On utilise le modèle **Façade** lorsque :

- on souhaite disposer d'une interface simple pour un système complexe.
- on souhaite découpler un sous-système des clients et des autres sous-systèmes. Ceci tend à favoriser l'indépendance et la portabilité du sous-système.
- on cherche à structurer un sous-système en sous-systèmes. On utilisera la façade pour définir un point d'entrée à chaque niveau du sous-système.

[Introduction](#)[Classification des patrons](#)[Format de description des patrons de conception](#)[Catalogue et description des patron de conception GoF](#)[Les patrons les plus courants](#)[Les créateurs : Fabrication, Fabrique Abstraite](#)[Les structures : Façade](#)[Les comportements](#)

Conséquences

Le modèle Facade :

- masque au client les composants du sous-système et le rend ainsi plus facile à utiliser,
- favorise le couplage faible entre le sous-système et ses clients. On peut ainsi faire évoluer le sous-système sans affecter ses clients.
- n'empêche pas aux clients d'utiliser directement les classes du sous-système si nécessaire. On choisit ainsi entre exhaustivité et confort d'utilisation.

- 1** Introduction
- 2** Classification des patrons
 - Les rôles : créateurs, structurels ou comportementaux
 - Les domaines : objets ou classes
 - Rôles et domaines
- 3** Format de description des patrons de conception
- 4** Catalogue et description des patron de conception GoF
- 5** Les patrons les plus courants
 - Les créateurs : Fabrication, Fabrique Abstraite
 - Les structurels : Façade
 - Les comportementaux : Itérateur, Observateur**
- 6** D'autres patrons moins fréquents
 - Les créateurs : Singleton
 - Les structurels : Adaptateur, Composite, Procuration
 - Les comportementaux : Commande, Stratégie, Visiteur
- 7** Conclusion
- 8** Références Bibliographiques

Itérateur - Iterator (1/4)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron de
conception

GoF

Les patrons
les plus
courants

Les
créateurs :
Fabrication,
Fabrique
Abstraite

Les
structures :
Façade

Les
comporte-

Catégories

Comportemental, Objet

Alias

Curseur

Intention

Fournit un moyen d'accès séquentiel aux éléments d'un agrégat d'objets sans mettre à découvert la représentation interne de celui-ci.

Implémentation existantes

Les interfaces *Iterator* et *Enumeration* de Java.

Itérateur - Iterator (2/4)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron de
conception
GoF

Les patrons
les plus
courants

Les
créateurs :
Fabrication,
Fabrique
Abstraite

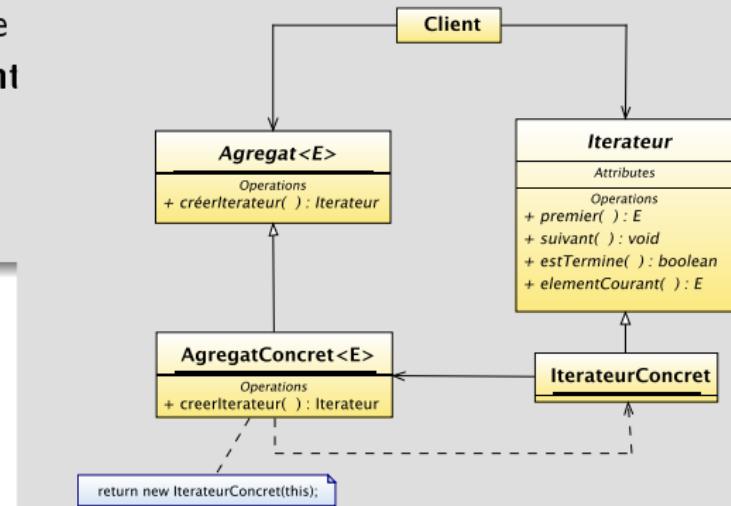
Les
structurels :
Façade

Les
comporte-

Collaboration

Un *ItérateurConcret* assure
le suivi de l'objet courant
de l'agrégat et peut
déterminer l'objet
suivant dans le parcours.

Structure



Itérateur - Iterator (3/4)

Introduction

Classification des patrons

Format de description des patrons de conception

Catalogue et description des patron de conception GoF

Les patrons les plus courants

Les créateurs : Fabrication, Fabrique Abstraite

Les structures : Façade

Les comportements

```
public class Tableau<E> extends Agregat<E>{
    private Object[] tableau ;

    public Tableau(int size){
        tableau =new Object[size];
    }

    public E getElement(int i){
        return (E)tableau[i];
    }

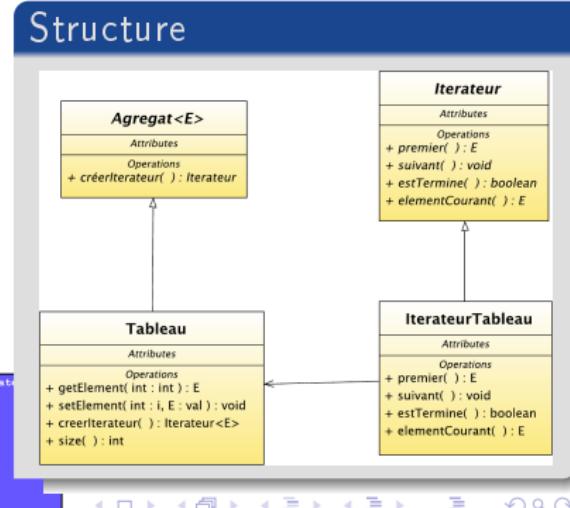
    public void setElement(int i, E val){
        tableau[i]=val;
    }

    public Iterateur<E> creerIterateur(){
        return new IterateurTableau<E>(this);
    }

    public int size(){
        return tableau.length;
    }
}
```

```
public class Tableau<E> extends Agregat<E>{
    private Object[] tableau ;
    public Tableau(int size){
        tableau =new Object[size];
    }

    public E premier(){
        indice=0;
        return montTableau.getElement(indice);
    }
```



Itérateur - Iterator (3/4)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron de
conception
GoF

**Les patrons
les plus
courants**

Les
créateurs :
Fabrication,
Fabrique
Abstraite

Les
structures :
Façade

Les
comporté-

```
public class IterateurTableau<E> extends Iterateur<E>{
    Tableau<E> monTableau ;
    int indice = 0 ;

    public IterateurTableau(Tableau<E> tab) {
        monTableau=tab;
    }

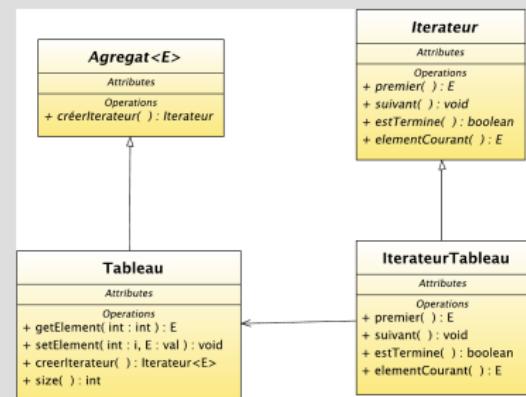
    public E premier(){
        indice=0;
        return monTableau.getElement(indice);
    }

    public void suivant(){
        indice=indice+1;
    }

    public boolean estTermine(){
        return indice==monTableau.size();
    }

    public E elementCourant(){
        return monTableau.getElement(indice);
    }
}
```

Structure



Itérateur - Iterator (4/4)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron de
conception
GoF

Les patrons
les plus
courants

Les
créateurs :
Fabrication,
Fabrique
Abstraite

Les
structurels :
Façade

Les
comporte-

Indications d'utilisation

- Plusieurs itérateurs différents peuvent permettre des parcours différents d'un objet agrégat
 - un agrégat de type arbre pourrait prévoir plusieurs itérateurs afin de le parcourir de diverses façon : parcours infixé, postfixé, préfixé
- Plusieurs itérateurs peuvent parcourir le même agrégat simultanément,
- Les itérateurs offrent une interface uniforme pour parcourir diverses structures agrégats.

- Observer (1/5)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron de
conception
GoF

Les patrons
les plus
courants

Les
créateurs :
Fabrication,
Fabrique
Abstraite

Les
structures :
Façade

Les
comporte-

Catégories

Comportemental, Objet

Alias

Dépendants, Diffusion-Souscription

Intention

Définit une interdépendance de type *un à plusieurs*, afin que lorsqu'un objet change d'état, tous ceux qui en dépendent en soient notifiés et soient mis à jour.

Implémentations existantes

L'interface *Observer* et la classe *Observable* de l'API Java mettent en oeuvre ce patron de conception.

Observateur - Observer (2/5)

Introduction

Classification des patrons

Format de description des patrons de conception

Catalogue et description des patron de conception GoF

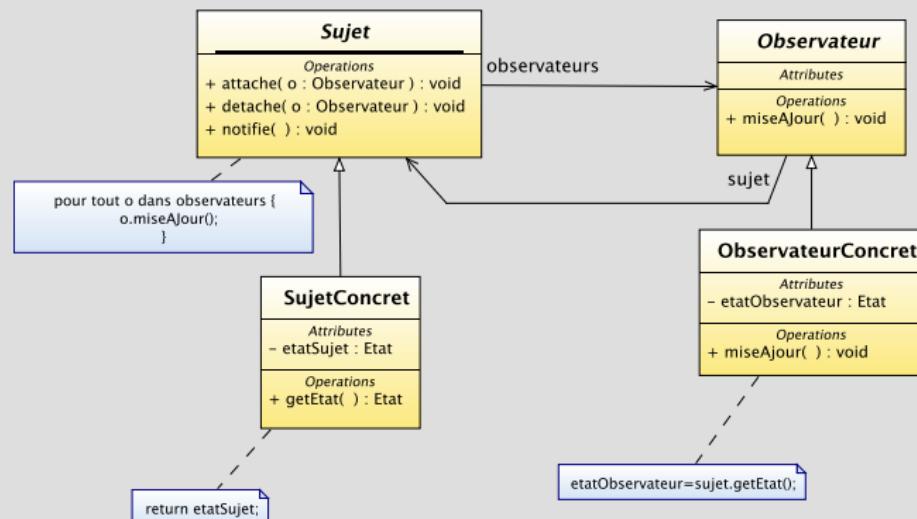
Les patrons les plus courants

Les créateurs : Fabrication, Fabrique Abstraite

Les structures : Façade

Les comportements

Structure



Observateur - Observer (3/5)

Introduction

Classification des patrons

Format de description des patrons de conception

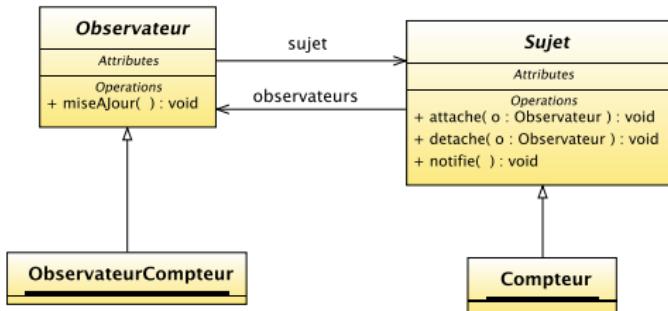
Catalogue et description des patron de conception GoF

Les patrons les plus courants

Les créateurs : Fabrication, Fabrique Abstraite

Les structuraux : Façade

Les comportementaux



Observateur - Observer (3/5)

Introduction

Classification des patrons

Format de description des patrons de conception

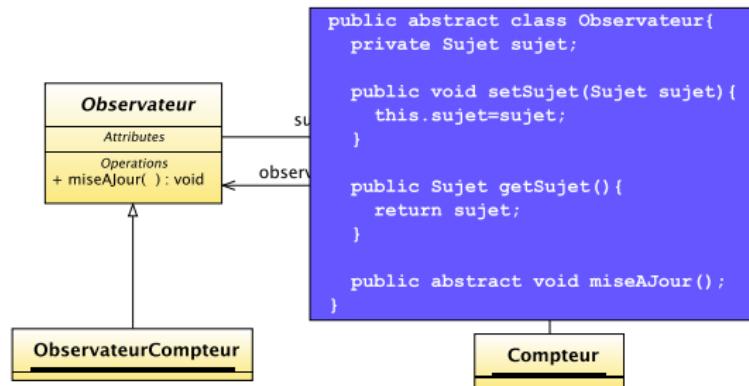
Catalogue et description des patron de conception GoF

Les patrons les plus courants

Les créateurs : Fabrication, Fabrique Abstraite

Les structuraux : Façade

Les comportementaux



Observateur - Observer (3/5)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron de
conception
GoF

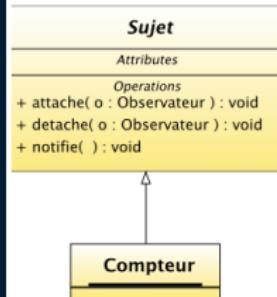
Les patrons
les plus
courants

Les
créateurs :
Fabrication,
Fabrique
Abstraite

Les
structuraux :
Façade

Les
comporta-

```
public abstract class Sujet {  
  
    private List<Observateur> observateurs;  
    observateurs=new ArrayList<Observateur> ();  
  
    public void attache(Observateur o){  
        observateurs.add(o);  
        o.setSujet(this);  
    }  
  
    public void detache(Observateur o){  
        observateurs.remove(o);  
        o.setSujet(null);  
    }  
  
    public void notifie(){  
        for (Observateur o : observateurs){  
            o.miseAJour();  
        }  
    }  
}
```



Observateur - Observer (3/5)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron de
conception
GoF

**Les patrons
les plus
courants**

Les
créateurs :
Fabrication,
Fabrique
Abstraite

Les
structurels :
Façade

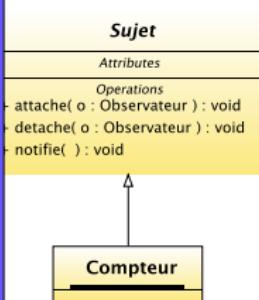
Les
comporta-

```
public class Compteur extends Sujet
    implements Runnable{
    private int compte=0;

    private void incremente(){
        compte++;
        notifie();
    }

    public int getValue(){
        return compte;
    }

    public void run(){
        Random generateur=new Random();
        while (true){
            try{
                Thread.sleep(generateur.nextInt(2)*1000);
            } catch (InterruptedException e){
                System.exit(0);
            }
            incremente();
        }
    }
}
```



Observateur - Observer (3/5)

Introduction

Classification des patrons

Format de description des patrons de conception

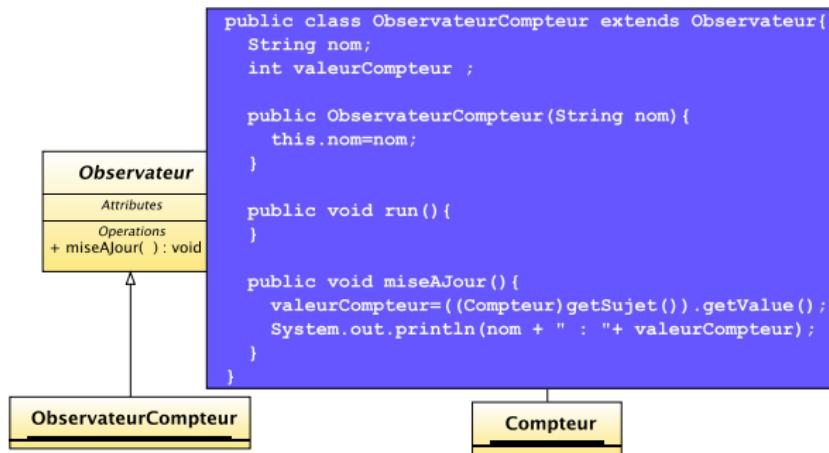
Catalogue et description des patron de conception GoF

Les patrons les plus courants

Les créateurs : Fabrication, Fabrique Abstraite

Les structuraux : Façade

Les comportements



Observateur - Observer (3/5)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron de
conception
GoF

Les patrons
les plus
courants

Les
créateurs :
Fabrication,
Fabrique
Abstraite

Les
structurels :
Façade

Les
comporte-

```
public class EssaiCompteur {  
    public static void main (String[] args){  
        Compteur compteur = new Compteur();  
        Thread threadCount=new Thread(compteur);  
        ObservateurCompteur o1=new ObservateurCompteur("compteur 1"),  
                           o2=new ObservateurCompteur("compteur 2"),  
                           o3=new ObservateurCompteur("compteur 3"),  
                           o4=new ObservateurCompteur("compteur 4");  
  
        threadCount.start();  
        compteur.attache(o1);  
        try{  
            Thread.sleep(3000);  
        } catch (InterruptedException e){  
            System.exit(0);  
        }  
        compteur.attache(o2);  
        compteur.attache(o3);  
        try{  
            Thread.sleep(2000);  
        } catch (InterruptedException e){  
            System.exit(0);  
        }  
        compteur.attache(o4);  
    }  
}
```

[Introduction](#)[Classification des patrons](#)[Format de description des patrons de conception](#)[Catalogue et description des patron de conception GoF](#)[Les patrons les plus courants](#)[Les créateurs : Fabrication, Fabrique Abstraite](#)[Les structures : Façade](#)[Les comportements](#)

Conséquences

- **Faible couplage entre Sujet et Observateurs** : un sujet sait qu'il possède une liste d'Observateurs, il ne les connaît pas leurs classes concrètes.
- **Support de la diffusion vers les Observateurs** : il n'est pas nécessaire de spécifier les destinataires, les notifications sont transmises à tous les objets qui ont souscrits.
- **Mises à jour inopinées** : le fait que les différents Observateurs d'un même sujet ne se connaissent pas peut entraîner des lourdeurs lors des mises à jour.

[Introduction](#)[Classification des patrons](#)[Format de description des patrons de conception](#)[Catalogue et description des patron de conception GoF](#)[Les patrons les plus courants](#)[Les créateurs : Fabrication, Fabrique Abstraite](#)[Les structures : Façade](#)[Les comportements](#)

Indications d'utilisation

On utilise le modèle **Observateur** lorsque

- un concept a deux représentations, l'une dépendant de l'autre. Encapsuler ces deux représentations dans deux objets différents permet de les faire évoluer indépendamment;
- la modification d'un objet nécessite des modifications sur d'autres objets dont on ignore le nombre;
- un objet doit être capable de notifier d'autres objets sans faire d'hypothèses sur la nature de ceux ci;

- 1** Introduction
- 2** Classification des patrons
 - Les rôles : créateurs, structurels ou comportementaux
 - Les domaines : objets ou classes
 - Rôles et domaines
- 3** Format de description des patrons de conception
- 4** Catalogue et description des patron de conception GoF
- 5** Les patrons les plus courants
 - Les créateurs : Fabrication, Fabrique Abstraite
 - Les structurels : Façade
 - Les comportementaux : Itérateur, Observateur
- 6** D'autres patrons moins fréquents
 - **Les créateurs : Singleton**
 - Les structurels : Adaptateur, Composite, Procuration
 - Les comportementaux : Commande, Stratégie, Visiteur
- 7** Conclusion
- 8** Références Bibliographiques

Singleton (1/4)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron
de conception
GoF

Les patrons
les plus
courants

D'autres
patrons
moins
fréquents

Les
créateurs :
Singleton

Les
structurels :
Adaptateur,



Catégories

Créateur, Objet

Intention

Garantit qu'une classe n'a qu'une seule instance et fournit un point d'accès de type global à cette classe

Singleton (2/4)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron de
conception
GoF

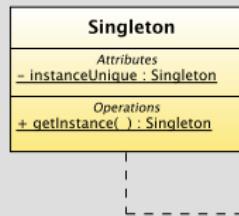
Les patrons
les plus
courants

D'autres
patrons
moins
fréquents

Les
créateurs :
Singleton

Les
structurels :
Adaptateur,

Structure



`return instanceUnique`

Collaborations

Les clients accèdent à l'unique instance du Singleton à travers sa méthode de classe `getInstance()`.

[Introduction](#)[Classification des patrons](#)[Format de description des patrons de conception](#)[Catalogue et description des patron de conception GoF](#)[Les patrons les plus courants](#)[D'autres patrons moins fréquents](#)[Les créateurs : Singleton](#)[Les structures : Adaptateur.](#)

Conséquences

Le modèle **Singleton** a de nombreux avantages :

- il fournit un accès contrôlé à son unique instance;
- il propose une alternative aux variables globales;
- il autorise un nombre variables d'instances : le modèle permet de changer de stratégie en autorisant plusieurs instances de la classe **Singleton**;

Singleton (4/4)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron de
conception
GoF

Les patrons
les plus
courants

D'autres
patrons
moins
fréquents

Les
créateurs :
Singleton

Les
structuraux :
Adaptateur,

Exemple de mise en œuvre

```
public class Singleton {  
    private static Singleton monInstance;  
    private Singleton(){  
    }  
    synchronized public static Singleton getInstance()  
    {  
        if (monInstance ==null)  
            monInstance=new Singleton();  
        return monInstance;  
    }  
}
```



- 1 Introduction
- 2 Classification des patrons
 - Les rôles : créateurs, structurels ou comportementaux
 - Les domaines : objets ou classes
 - Rôles et domaines
- 3 Format de description des patrons de conception
- 4 Catalogue et description des patron de conception GoF
- 5 Les patrons les plus courants
 - Les créateurs : Fabrication, Fabrique Abstraite
 - Les structurels : Façade
 - Les comportementaux : Itérateur, Observateur
- 6 D'autres patrons moins fréquents
 - Les créateurs : Singleton
 - **Les structurels : Adaptateur, Composite, Procuration**
 - Les comportementaux : Commande, Stratégie, Visiteur
- 7 Conclusion
- 8 Références Bibliographiques

Adaptateur - Adapter (1/4)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron
de conception
GoF

Les patrons
les plus
courants

D'autres
patrons
moins
fréquents

Les
créateurs :
Singleton

Les
structurels :
Adaptateur,

Catégories

Structurel, Objet ou Classe

Alias

Empaqueteur

Intention

Convertit l'interface d'une classe en une autre conforme à l'attente du client

Adaptateur - Adapter (2/4)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron de
conception
GoF

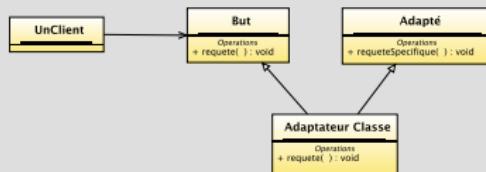
Les patrons
les plus
courants

D'autres
patrons
moins
fréquents

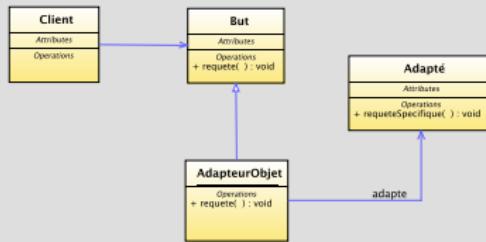
Les
créateurs :
Singleton

Les
structuraux :
Adaptateur,

Structure de l'adaptateur de classe



Structure de l'adaptateur d'objet



[Introduction](#)[Classification des patrons](#)[Format de description des patrons de conception](#)[Catalogue et description des patron de conception GoF](#)[Les patrons les plus courants](#)[D'autres patrons moins fréquents](#)[Les créateurs : Singleton](#)[Les structures : Adaptateur,](#)

Indication d'utilisation

On utilise le patron Adaptateur lorsque :

- on veut utiliser une classe existante dont l'interface ne coïncide pas avec celle souhaitée;
- on souhaite créer une classe réutilisable qui collabore avec des classes sans relations avec elle et encore inconnues (c'est-à-dire dont les interfaces ne seront peut être pas compatibles);
- (*pour le cas des adaptateurs d'objets*) on a besoin d'utiliser plusieurs sous-classes existantes, mais l'adaptation de chacune d'entre elles par dérivation n'est pas possible. Un adaptateur objet peut adapter l'interface de sa classe parente.

Adaptateur - Adapter (4/4)

Introduction

Classification des patrons

Format de description des patrons de conception

Catalogue et description des patron de conception GoF

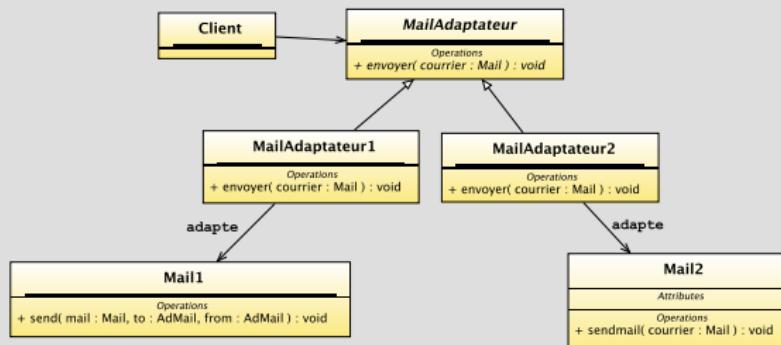
Les patrons les plus courants

D'autres patrons moins fréquents

Les créateurs : Singleton

Les structuraux : Adaptateur,

Exemple d'utilisation



Adaptateur - Adapter (4/4)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron de
conception
GoF

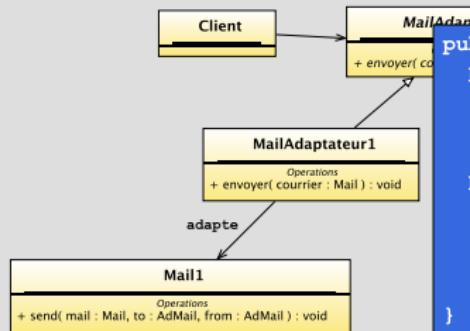
Les patrons
les plus
courants

D'autres
patrons
moins
fréquents

Les
créateurs :
Singleton

Les
structurels :
Adaptateur,

Exemple d'utilisation



```
public class MailAdaptateur1 extends MailAdaptateur {
    public MailAdaptateur1(Mail1 monAdapté) {
        adapte=monAdapté;
    }

    public void envoyer(Mail courrier) {
        adapte.send(courrier.getBody(),
                    courrier.getTo(),
                    courrier.getFrom());
    }
}
```

Adaptateur - Adapter (4/4)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron de
conception
GoF

Les patrons
les plus
courants

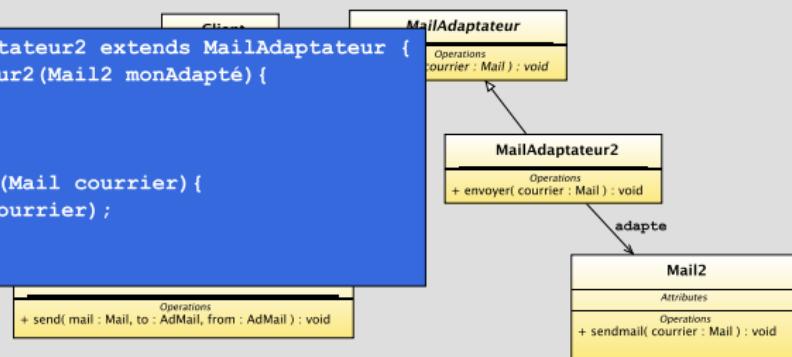
D'autres
patrons
moins
fréquents

Les
créateurs :
Singleton

Les
structurels :
Adaptateur,

Exemple d'utilisation

```
public class MailAdaptateur2 extends MailAdaptateur {  
    public MailAdaptateur2(Mail2 monAdapté){  
        adapte=monAdapté;  
    }  
  
    public void envoyer(Mail courrier){  
        adapte.sendmail(courrier);  
    }  
}
```



Composite (1/3)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron
de conception
GoF

Les patrons
les plus
courants

D'autres
patrons
moins
fréquents

Les
créateurs :
Singleton

Les
structurels :
Adaptateur,

Catégories

Structurel, Objet

Intention

Le patron Composite compose des objets en des structures arborescentes pour représenter des hiérarchies composant/composé. Il permet au client de traiter de la même et unique façon les objets individuels et les combinaisons de ceux-ci.

Composite (2/3)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron de
conception
GoF

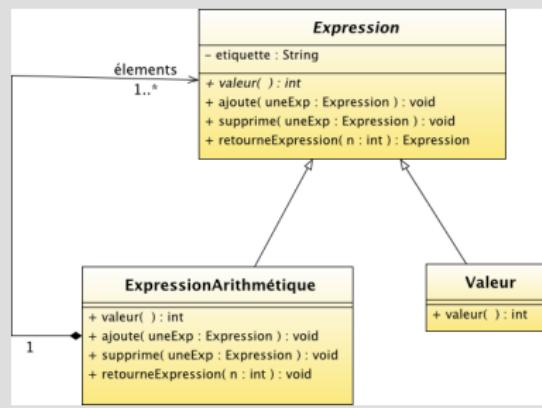
Les patrons
les plus
courants

D'autres
patrons
moins
fréquents

Les
créateurs :
Singleton

Les
structurels :
Adaptateur,

Structure



Composite (3/3)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron de
conception
GoF

Les patrons
les plus
courants

D'autres
patrons
moins
fréquents

Les
créateurs :
Singleton

Les
structurels :
Adaptateur,

Indication d'utilisation

On utilise le patron Composite lorsque :

- on souhaite représenter des hiérarchies de l'individu à l'ensemble;
- on souhaite que le client n'ait pas à se préoccuper de la différence qui existe entre une combinaison d'objets et un objet individuel. le client pourra traiter tous les objets de la structure composite de la même manière.

Procuration - Proxy (1/3)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron de
conception
GoF

Les patrons
les plus
courants

D'autres
patrons
moins
fréquents

Les
créateurs :
Singleton

Les
structurels :
Adaptateur,

Catégories

Structurel, Objet

Alias

Subrogé

Intention

Fournit à un tiers objet un mandataire ou un remplaçant pour contrôler l'accès à cet objet

Procuration - Proxy (2/3)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron de
conception
GoF

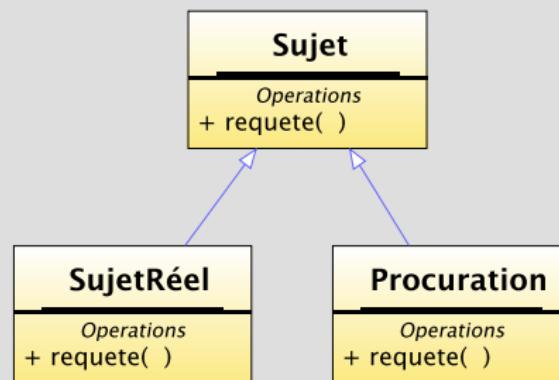
Les patrons
les plus
courants

D'autres
patrons
moins
fréquents

Les
créateurs :
Singleton

Les
structurels :
Adaptateur,

Structure



Procuration - Proxy (3/3)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron
de conception
GoF

Les patrons
les plus
courants

D'autres
patrons
moins
fréquents

Les
créateurs :
Singleton

Les
structurels :
Adaptateur,

Plusieurs types de procurations

Procuration à distance: représentant local d'un objet situé dans un espace d'adresse différent;

Procuration virtuelle: création d'objets lourds à la demande (une procuration d'image pour ne pas manipuler l'image complète tant que ce n'est pas nécessaire - chargement au besoin);

Procuration de protection: contrôle l'accès à l'objet original (rend possible la vérification des droits d'accès aux objets);

Références intelligentes: remplace un pointeur brut (permet le comptage de référence, chargement et construction paresseuses);

- 1** Introduction
- 2** Classification des patrons
 - Les rôles : créateurs, structurels ou comportementaux
 - Les domaines : objets ou classes
 - Rôles et domaines
- 3** Format de description des patrons de conception
- 4** Catalogue et description des patron de conception GoF
- 5** Les patrons les plus courants
 - Les créateurs : Fabrication, Fabrique Abstraite
 - Les structurels : Façade
 - Les comportementaux : Itérateur, Observateur
- 6** D'autres patrons moins fréquents
 - Les créateurs : Singleton
 - Les structurels : Adaptateur, Composite, Procuration
 - Les comportementaux : Commande, Stratégie, Visiteur**
- 7** Conclusion
- 8** Références Bibliographiques

Commande - Command (1/3)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron de
conception
GoF

Les patrons
les plus
courants

D'autres
patrons
moins
fréquents

Les
créateurs :
Singleton

Les
structurels :
Adaptateur,

Catégories

Comportemental, Objet

Alias

Action, Transaction

Intention

Encapsuler une requête comme un objet, autorisant ainsi le paramétrage des clients par différentes requêtes, files d'attente et récapitulatifs de requêtes et, de plus, permettant l'annulation des opérations.

Commande - Command (2/3)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

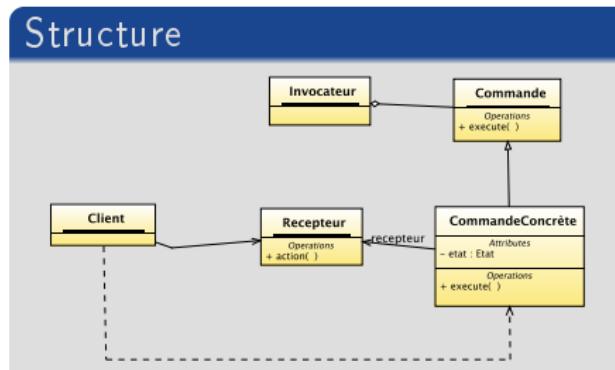
Catalogue et
description
des patron de
conception
GoF

Les patrons
les plus
courants

D'autres
patrons
moins
fréquents

Les
créateurs :
Singleton

Les
structurels :
Adaptateur,



Commande concrète:

CommandeColler,
Commande-
Copier

Client: Application

Invocateur: MenuElement

Récepteur: document,
application

Commande - Command (3/3)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron de
conception
GoF

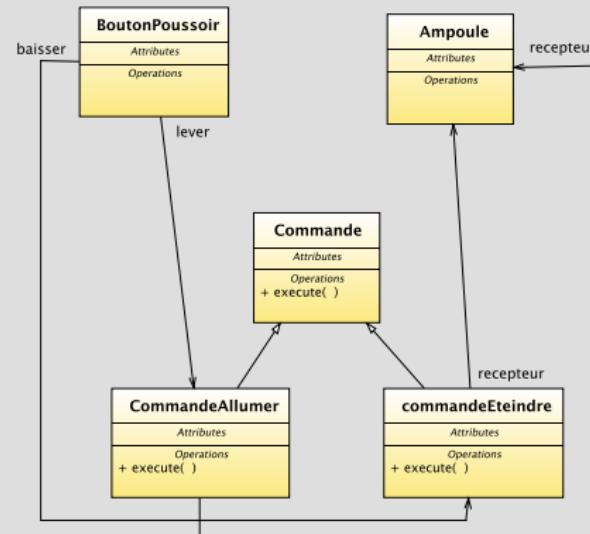
Les patrons
les plus
courants

D'autres
patrons
moins
fréquents

Les
créateurs :
Singleton

Les
structures :
Adaptateur,

Exemple d'utilisation



Commande - Command (3/3)

Introduction

Classification des patrons

Format de description des patrons de conception

Catalogue et description des patron de conception GoF

Les patrons les plus courants

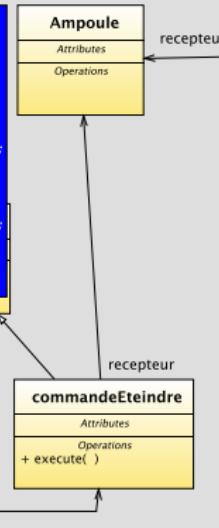
D'autres patrons moins fréquents

Les créateurs : Singleton

Les structures : Adaptateur,

Exemple d'utilisation

```
/* Le Récepteur */
public class Ampoule{
    public Ampoule(){}
    public void allumer(){
        System.out.println("La lumière est allumée");
    }
    public void eteindre(){
        System.out.println("La lumière est éteinte");
    }
}
```



Commande - Command (3/3)

Introduction

Classification des patrons

Format de description des patrons de conception

Catalogue et description des patron de conception GoF

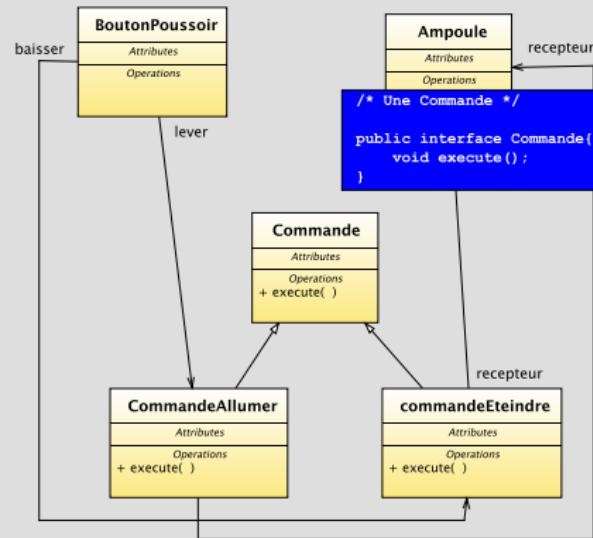
Les patrons les plus courants

D'autres patrons moins fréquents

Les créateurs : Singleton

Les structures : Adaptateur,

Exemple d'utilisation



Commande - Command (3/3)

Introduction

Classification des patrons

Format de description des patrons de conception

Catalogue et description des patron de conception GoF

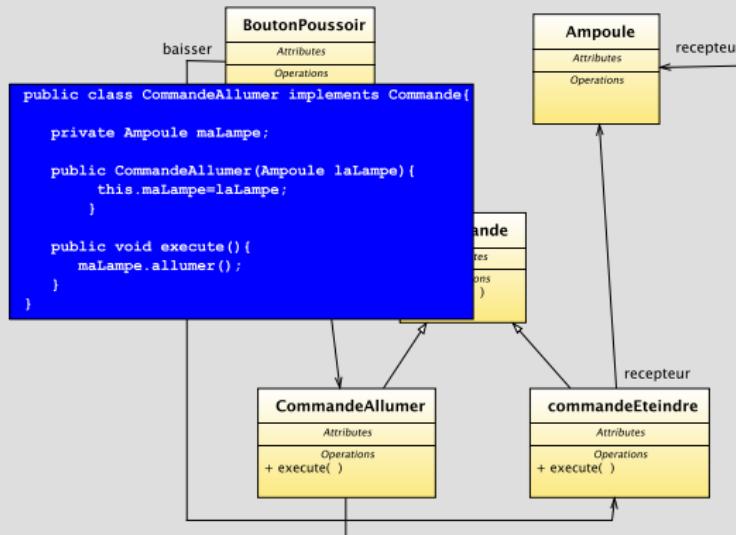
Les patrons les plus courants

D'autres patrons moins fréquents

Les créateurs : Singleton

Les structures : Adaptateur,

Exemple d'utilisation



Commande - Command (3/3)

Introduction

Classification des patrons

Format de description des patrons de conception

Catalogue et description des patron de conception GoF

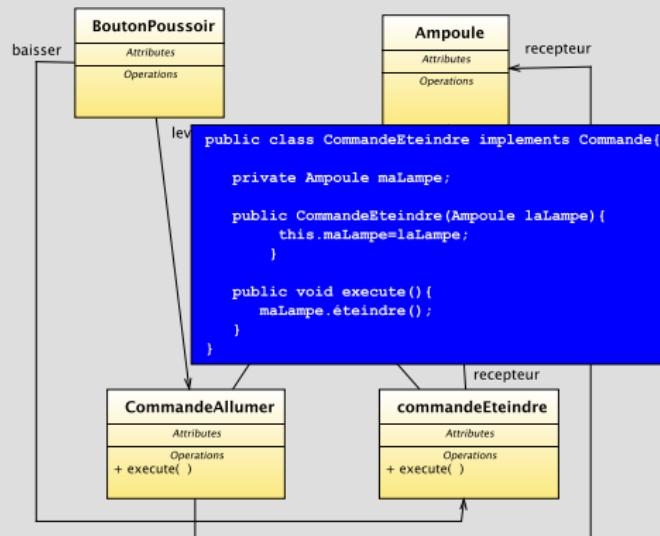
Les patrons les plus courants

D'autres patrons moins fréquents

Les créateurs : Singleton

Les structures : Adaptateur,

Exemple d'utilisation



Commande - Command (3/3)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron de
conception
GoF

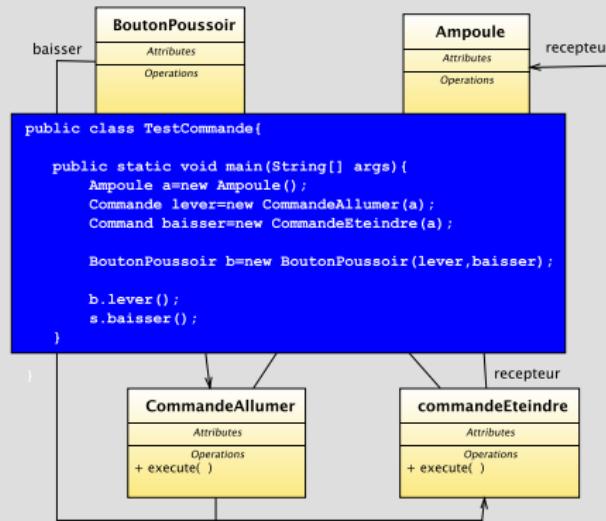
Les patrons
les plus
courants

D'autres
patrons
moins
fréquents

Les
créateurs :
Singleton

Les
structurels :
Adaptateur,

Exemple d'utilisation



Stratégie - Strategy (1/3)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron de
conception
GoF

Les patrons
les plus
courants

D'autres
patrons
moins
fréquents

Les
créateurs :
Singleton

Les
structurels :
Adaptateur,

Catégories

Comportemental, Objet

Alias

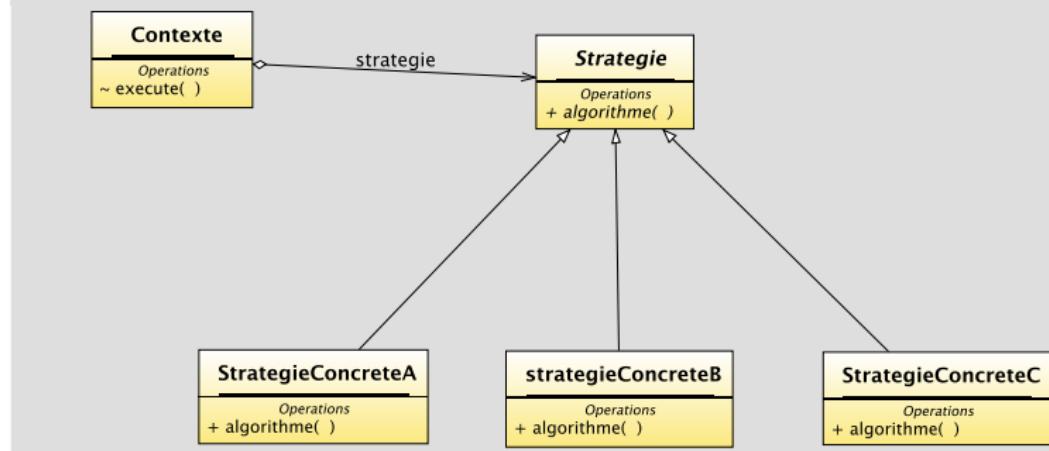
Politique

Intention

Définit une famille d'algorithmes, encapsule chacun d'entre eux et les rend interchangeables. Le modèle Stratégie permet aux algorithmes d'évoluer indépendamment des clients qui les utilisent.

[Introduction](#)[Classification des patrons](#)[Format de description des patrons de conception](#)[Catalogue et description des patron de conception GoF](#)[Les patrons les plus courants](#)[D'autres patrons moins fréquents](#)[Les créateurs : Singleton](#)[Les structures : Adaptateur,](#)

Structure



Stratégie - Strategy (3/3)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron de
conception
GoF

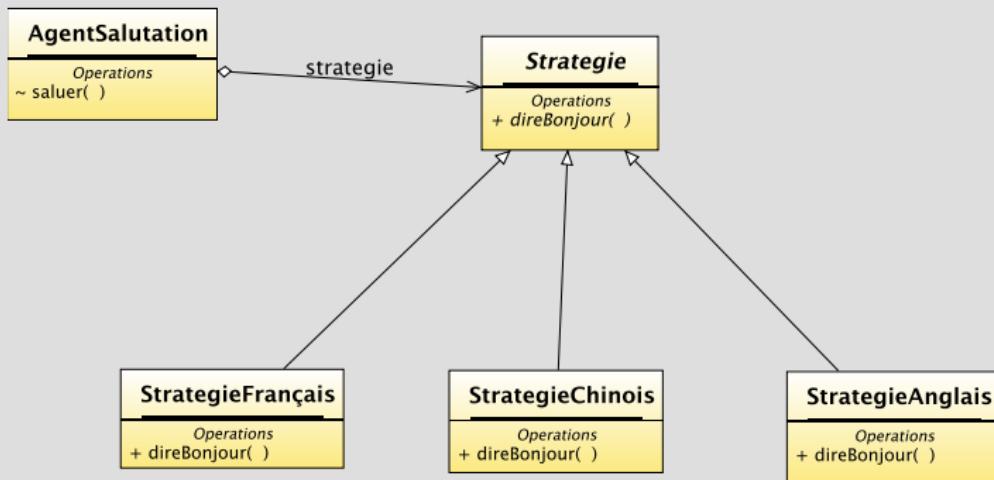
Les patrons
les plus
courants

D'autres
patrons
moins
fréquents

Les
créateurs :
Singleton

Les
structuraux :
Adaptateur,

Exemple d'utilisation



Stratégie - Strategy (3/3)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron de
conception
GoF

Les patrons
les plus
courants

D'autres
patrons
moins
fréquents

Les
créateurs :
Singleton

Les
structurels :
Adaptateur,

Exemple d'utilisation



```
public class AgentSalutation {  
    Strategie maStrategie;  
  
    public AgentSalutation(Strategie strategie){  
        maStrategie=strategie;  
    }  
  
    public void changeStrategie(Strategie newStrat){  
        maStrategie=newStrat;  
    }  
  
    public void saluer(){  
        maStrategie.direBonjour();  
    }  
}
```



glais

Stratégie - Strategy (3/3)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron de
conception
GoF

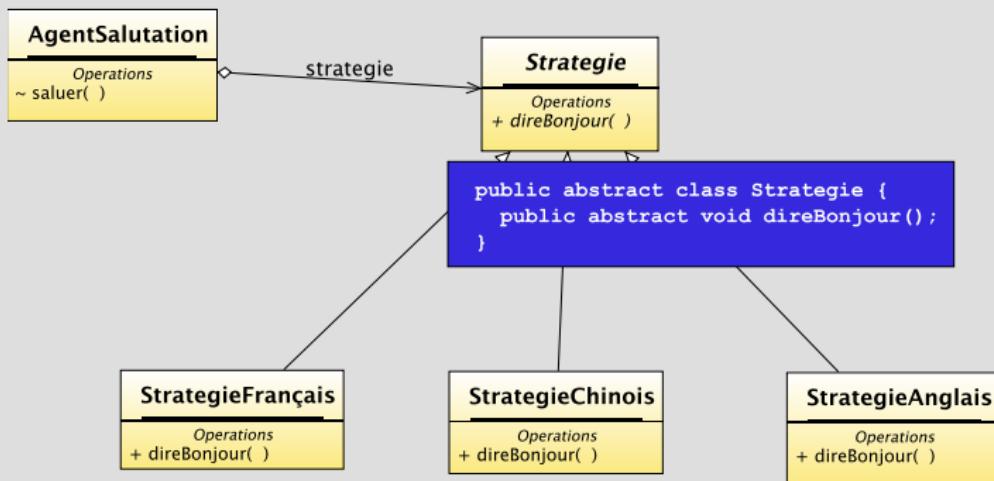
Les patrons
les plus
courants

D'autres
patrons
moins
fréquents

Les
créateurs :
Singleton

Les
structurels :
Adaptateur,

Exemple d'utilisation



Stratégie - Strategy (3/3)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron de
conception
GoF

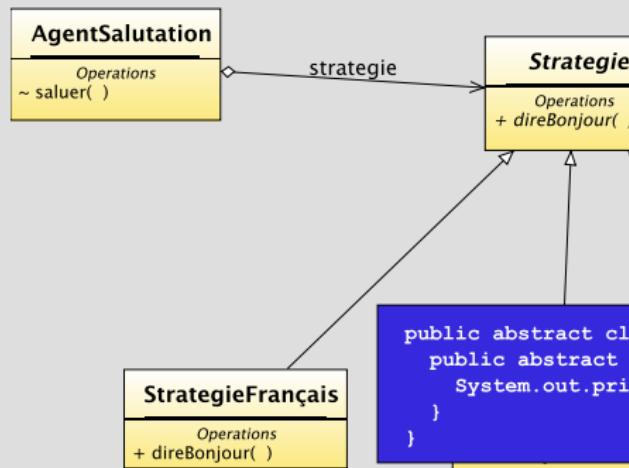
Les patrons
les plus
courants

D'autres
patrons
moins
fréquents

Les
créateurs :
Singleton

Les
structures :
Adaptateur,

Exemple d'utilisation



```
public abstract class StrategieFrançais extends Strategie {
    public abstract void direBonjour(){
        System.out.println("Bonjour !!!");
    }
}
```

Stratégie - Strategy (3/3)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron
de conception
GoF

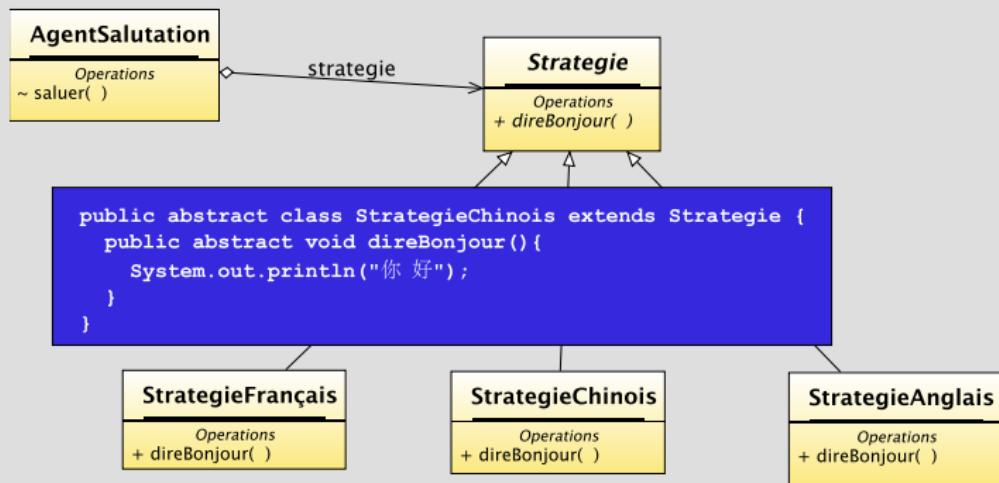
Les patrons
les plus
courants

D'autres
patrons
moins
fréquents

Les
créateurs :
Singleton

Les
structuraux :
Adaptateur,

Exemple d'utilisation



Stratégie - Strategy (3/3)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron de
conception
GoF

Les patrons
les plus
courants

D'autres
patrons
moins
fréquents

Les
créateurs :
Singleton

Les
structuraux :
Adaptateur,

Exemple d'utilisation



```
public abstract class StrategieAnglais extends Strategie {
    public abstract void direBonjour(){
        System.out.println("Hello");
    }
}
```



Stratégie - Strategy (3/3)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron de
conception
GoF

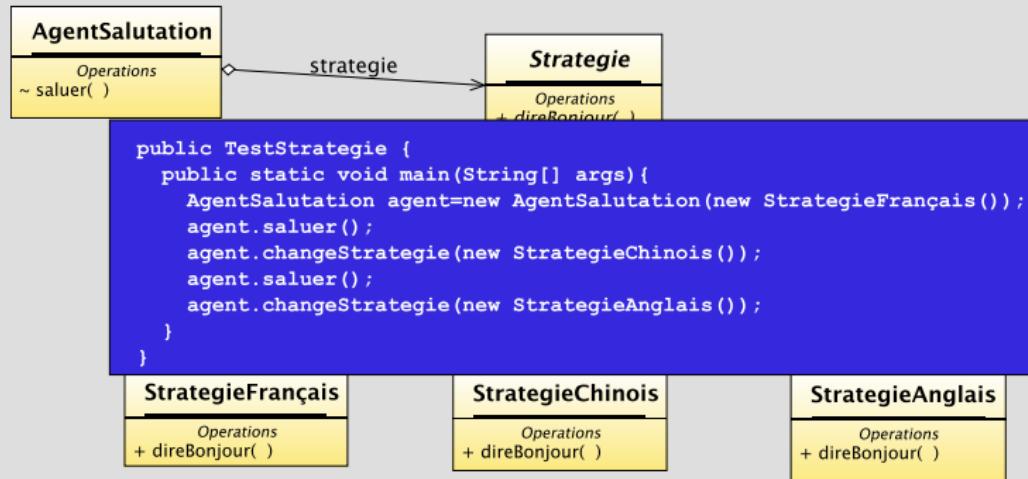
Les patrons
les plus
courants

D'autres
patrons
moins
fréquents

Les
créateurs :
Singleton

Les
structurels :
Adaptateur,

Exemple d'utilisation



[Introduction](#)[Classification des patrons](#)[Format de description des patrons de conception](#)[Catalogue et description des patron de conception GoF](#)[Les patrons les plus courants](#)[D'autres patrons moins fréquents](#)[Les créateurs : Singleton](#)[Les structures : Adaptateur.](#)

Catégories

Comportemental, Objet

Intention

Représente un opération à appliquer aux différents éléments d'une structure d'objets. Le patron Visiteur permet de définir l'application d'une opération sur les instances d'une classe sans avoir à modifier la classe sur laquelle l'opération s'applique.

Visiteur - Visitor (2/4)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron de
conception
GoF

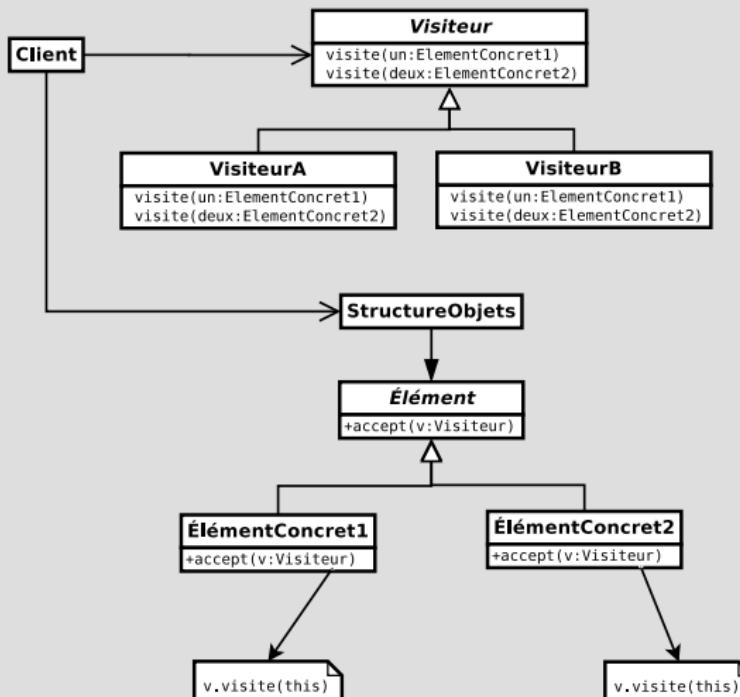
Les patrons
les plus
courants

D'autres
patrons
moins
fréquents

Les
créateurs :
Singleton

Les
structurels :
Adaptateur,

Structure



Visiteur - Visitor (3/4)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron de
conception
GoF

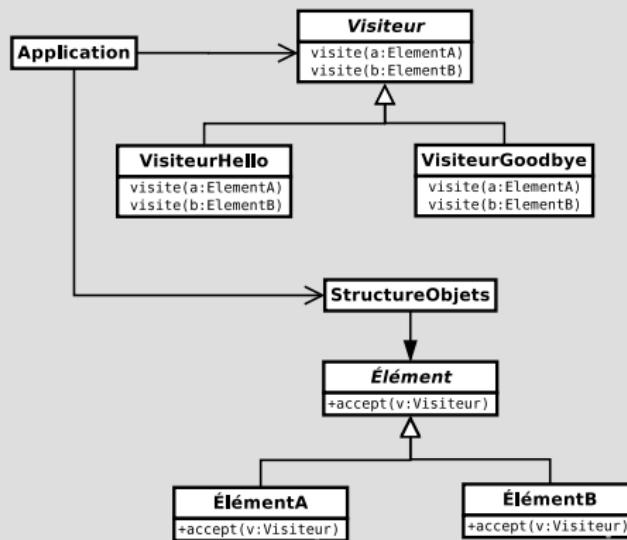
Les patrons
les plus
courants

D'autres
patrons
moins
fréquents

Les
créateurs :
Singleton

Les
structurels :
Adaptateur,

Exemple d'utilisation



Visiteur - Visitor (3/4)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron de
conception
GoF

Les patrons
les plus
courants

D'autres
patrons
moins
fréquents

Les
créateurs :
Singleton

Les
structurels :
Adaptateur,

Exemple d'utilisation

```
public class Client {  
    public static void main(String[] args) {  
        Random r=new Random();  
        ArrayList<Element> liste=new ArrayList<>();  
        // liste est remplie aléatoirement  
        // de ElementA et des ElementB  
        for (int i=0;i<10;i++){  
            if (r.nextBoolean()){  
                liste.add(new ElementA());  
            } else {  
                liste.add(new ElementB());  
            }  
        }  
        // On applique aléatoirement l'opération goodbye  
        // et hello aux élément de liste  
        for (Element e:liste){  
            if (r.nextBoolean()){  
                e.accept(new VisitorGoodbye());  
            } else {  
                e.accept(new VisitorHello());  
            }  
        }  
    }  
}
```

Visiteur - Visitor (3/4)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron de
conception
GoF

Les patrons
les plus
courants

D'autres
patrons
moins
fréquents

Les
créateurs :
Singleton

Les
structurels :
Adaptateur,

Exemple d'utilisation

```
public abstract class Element {  
    abstract void accept (Visitor v);  
}  
  
public class ElementA extends Element {  
    void accept(Visitor v){  
        v.visit(this);  
    }  
}  
  
public class ElementB extends Element {  
    void accept(Visitor v){  
        v.visit(this);  
    }  
}
```

Visiteur - Visitor (3/4)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron de
conception
GoF

Les patrons
les plus
courants

D'autres
patrons
moins
fréquents

Les
créateurs :
Singleton

Les
structures :
Adaptateur,

Exemple d'utilisation

```
public abstract class Visitor {  
    abstract void visit(ElementA a);  
    abstract void visit(ElementB b);  
}  
  
public class VisitorGoodbye extends Visitor {  
  
    void visit(ElementB b) {  
        System.out.println("Goodbye b");  
    }  
  
    void visit(ElementA a) {  
        System.out.println("Goodbye a");  
    }  
}  
  
public class VisitorHello extends Visitor {  
  
    void visit(ElementB b) {  
        System.out.println("Hello b");  
    }  
  
    void visit(ElementA a) {  
        System.out.println("Hello a");  
    }  
}
```

Visiteur - Visitor (4/4)

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron de
conception
GoF

Les patrons
les plus
courants

D'autres
patrons
moins
fréquents

Les
créateurs :
Singleton

Les
structurels :
Adaptateur,

Conséquences

- L'ajout de nouvelles opération est simplifié par le patron Visitor. Il suffit d'ajouter l'implémentation de la nouvelle opération dans chacune des classes visiteur
- Les opérations d'un objet (ou d'une classe) sont centralisées dans son visiteur
- L'ajout d'une nouvelle classe d'élément concret est fastidieux. en effet, il est nécessaire d'ajouter une méthode, prenant en compte le nouvel objet concret dans chacune des classes visiteur.

Conclusion

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron de
conception
GoF

Les patrons
les plus
courants

D'autres
patrons
moins
fréquents

Conclusion

Références
Bibli-
ographiques

Christopher Alexander

On peut faire des batiments en entassant, sans liens, des modèles. Une construction ainsi faite, n'est qu'un assemblage de modèles. Elle n'est pas dense; elle n'a pas de consistance. Mais il est aussi possible, d'associer les modèles de telle façon que, plusieurs se recouvrent dans le même espace physique : la construction devient alors très dense ; elle réunit de nombreuses significations dans un petit espace ; et du fait de cette densité, elle prend corps.

Une application est constituée de nombreux modèles qui s'imbriquent et cohabitent pour fournir un tout plus grand !

Ouvrages de référence

Introduction

Classification
des patrons

Format de
description
des patrons
de
conception

Catalogue et
description
des patron de
conception
GoF

Les patrons
les plus
courants

D'autres
patrons
moins
fréquents

Conclusion

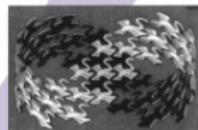
Références
Bibli-
ographiques

Version Originale du Gang des Quatre

Design Patterns

Elements of Reusable
Object-Oriented Software

Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides



Foreword by Grady Booch

WESLEY PROFESSIONAL COMPUTING SERIES

Traduction française

DESIGN PATTERNS

Catalogue de modèles de conception réutilisables

Erich Gamma, Richard Helm, Ralph Johnson et John Vlissides
Traduction de Jean-Marie Las vergères



[Introduction](#)[Classification des patrons](#)[Format de description des patrons de conception](#)[Catalogue et description des patron de conception GoF](#)[Les patrons les plus courants](#)[D'autres patrons moins fréquents](#)[Conclusion](#)[Références Bibliographiques](#)

Pattern-Oriented Software Architecture (POSA)

Buschmann, Meunier, Rohnert, Sommerlad, Stal



D'autres ouvrages

[Introduction](#)

[Classification des patrons](#)

[Format de description des patrons de conception](#)

[Catalogue et description des patron de conception GoF](#)

[Les patrons les plus courants](#)

[D'autres patrons moins fréquents](#)

[Conclusion](#)

[Références Bibliographiques](#)

