



EXPERT

4^e édition

Design Patterns en Java

Les 23 modèles de conception :
descriptions et solutions illustrées
en **UML 2** et **Java**

Version numérique

OFFERTE !

www.editions-eni.fr

Fichiers complémentaires
à télécharger



Laurent DEBRAUWER

Les éléments à télécharger sont disponibles à l'adresse suivante :
<http://www.editions-eni.fr>
Saisissez la référence ENI de l'ouvrage **EI4DES** dans la zone de recherche et validez. Cliquez sur le titre du livre puis sur le bouton de téléchargement.

Avant-propos

Partie 1 : Introduction

Chapitre 1

Introduction aux patterns de conception

- 1. Design Patterns ou patterns de conception 15
- 2. La description des patterns de conception 17
- 3. Le catalogue des patterns de conception 18
- 4. Comment choisir et utiliser un pattern de conception
pour résoudre un problème 20
- 5. Organisation du catalogue des patterns de conception 23

Chapitre 2

Une étude de cas : la vente en ligne de véhicules

- 1. Description du système 25
- 2. Cahier des charges 25
- 3. Prise en compte des patterns de conception 27

2 _____ Design Patterns en Java

Les 23 modèles de conception

Partie 2 : Patterns de construction

Chapitre 3

Introduction aux patterns de construction

1. Présentation	29
2. Les problèmes liés à la création d'objets	30
2.1 Problématique	30
2.2 Les solutions proposées par les patterns de construction	31

Chapitre 4

Le pattern Abstract Factory

1. Description	33
2. Exemple	33
3. Structure	36
3.1 Diagramme de classes	36
3.2 Participants	37
3.3 Collaborations	37
4. Domaines d'utilisation	37
5. Exemple en Java	38

Chapitre 5

Le pattern Builder

1. Description	45
2. Exemple	45
3. Structure	47
3.1 Diagramme de classes	47
3.2 Participants	47
3.3 Collaborations	48
4. Domaines d'utilisation	49
5. Exemple en Java	49

Chapitre 6
Le pattern Factory Method

- 1. Description 55
- 2. Exemple 55
- 3. Structure 57
 - 3.1 Diagramme de classes 57
 - 3.2 Participants 58
 - 3.3 Collaborations 58
- 4. Domaines d'utilisation 58
- 5. Exemple en Java 59

Chapitre 7
Le pattern Prototype

- 1. Description 63
- 2. Exemple 63
- 3. Structure 66
 - 3.1 Diagramme de classes 66
 - 3.2 Participants 67
 - 3.3 Collaboration 67
- 4. Domaines d'utilisation 67
- 5. Exemple en Java 68

Chapitre 8
Le pattern Singleton

- 1. Description 73
- 2. Exemple 73
- 3. Structure 74
 - 3.1 Diagramme de classe 74

4 _____ Design Patterns en Java

Les 23 modèles de conception

3.2 Participant.....	74
3.3 Collaboration	75
4. Domaine d'utilisation	75
5. Exemples en Java	75
5.1 La liasse vierge.....	75
5.2 La classe Vendeur	76

Partie 3 : Patterns de structuration

Chapitre 9

Introduction aux patterns de structuration

1. Présentation	79
2. Composition statique et dynamique.....	80

Chapitre 10

Le pattern Adapter

1. Description	83
2. Exemple.....	83
3. Structure	85
3.1 Diagramme de classes.....	85
3.2 Participants.....	85
3.3 Collaborations.....	86
4. Domaines d'application	86
5. Exemple en Java	87

Chapitre 11
Le pattern Bridge

- 1. Description 91
- 2. Exemple..... 91
- 3. Structure 94
 - 3.1 Diagramme de classes..... 94
 - 3.2 Participants..... 95
 - 3.3 Collaborations..... 95
- 4. Domaines d'application 95
- 5. Exemple en Java 96

Chapitre 12
Le pattern Composite

- 1. Description 101
- 2. Exemple..... 101
- 3. Structure 104
 - 3.1 Diagramme de classes..... 104
 - 3.2 Participants..... 104
 - 3.3 Collaborations..... 105
- 4. Domaines d'application 106
- 5. Exemple en Java 107

Chapitre 13
Le pattern Decorator

- 1. Description 109
- 2. Exemple..... 109
- 3. Structure 114
 - 3.1 Diagramme de classes..... 114

6 _____ Design Patterns en Java

Les 23 modèles de conception

3.2 Participants	115
3.3 Collaborations.	115
4. Domaines d'application	115
5. Exemple en Java	116

Chapitre 14 **Le pattern Facade**

1. Description	119
2. Exemple.	119
3. Structure	122
3.1 Diagramme de classes.	122
3.2 Participants	123
3.3 Collaborations.	123
4. Domaines d'application	124
5. Exemple en Java	125

Chapitre 15 **Le pattern Flyweight**

1. Description	129
2. Exemple.	129
3. Structure	132
3.1 Diagramme de classes.	132
3.2 Participants	132
3.3 Collaborations.	133
4. Domaine d'application.	133
5. Exemple en Java	133

Chapitre 16
Le pattern Proxy

- 1. Description 137
- 2. Exemple 137
- 3. Structure 140
 - 3.1 Diagramme de classes 140
 - 3.2 Participants 141
 - 3.3 Collaborations 142
- 4. Domaines d'application 142
- 5. Exemple en Java 142

Partie 4 : Patterns de comportement

Chapitre 17
Introduction aux patterns de comportement

- 1. Présentation 145
- 2. Distribution par héritage ou par délégation 146

Chapitre 18
Le pattern Chain of Responsibility

- 1. Description 149
- 2. Exemple 149
- 3. Structure 153
 - 3.1 Diagramme de classes 153
 - 3.2 Participants 153
 - 3.3 Collaborations 154
- 4. Domaines d'application 154
- 5. Exemple en Java 154

8 _____ Design Patterns en Java

Les 23 modèles de conception

Chapitre 19

Le pattern Command

1. Description	159
2. Exemple.....	159
3. Structure	163
3.1 Diagramme de classes.....	163
3.2 Participants.....	164
3.3 Collaborations.....	164
4. Domaines d'application	165
5. Exemple en Java	166

Chapitre 20

Le pattern Interpreter

1. Description	171
2. Exemple.....	171
3. Structure	174
3.1 Diagramme de classes.....	174
3.2 Participants.....	175
3.3 Collaborations.....	175
4. Domaines d'application	176
5. Exemple en Java	176

Chapitre 21

Le pattern Iterator

1. Description	183
2. Exemple.....	183
3. Structure	186
3.1 Diagramme de classes.....	186

- 3.2 Participants 187
- 3.3 Collaborations. 187
- 4. Domaines d'application 187
- 5. Exemple en Java 188

Chapitre 22
Le pattern Mediator

- 1. Description 193
- 2. Exemple 193
- 3. Structure 197
 - 3.1 Diagramme de classes. 197
 - 3.2 Participants 197
 - 3.3 Collaborations. 198
- 4. Domaines d'application 198
- 5. Exemple en Java 198

Chapitre 23
Le pattern Memento

- 1. Description 205
- 2. Exemple 205
- 3. Structure 208
 - 3.1 Diagramme de classes. 208
 - 3.2 Participants 208
 - 3.3 Collaborations. 209
- 4. Domaines d'application 209
- 5. Exemple en Java 209

Chapitre 24

Le pattern Observer

1. Description	213
2. Exemple	213
3. Structure	216
3.1 Diagramme de classes	216
3.2 Participants	217
3.3 Collaborations	217
4. Domaines d'application	217
5. Exemple en Java	218

Chapitre 25

Le pattern State

1. Description	221
2. Exemple	221
3. Structure	224
3.1 Diagramme de classes	224
3.2 Participants	224
3.3 Collaborations	225
4. Domaines d'application	225
5. Exemple en Java	225

Chapitre 26

Le pattern Strategy

1. Description	231
2. Exemple	232
3. Structure	234
3.1 Diagramme de classes	234

- 3.2 Participants 234
- 3.3 Collaborations. 235
- 4. Domaines d'application 235
- 5. Exemple en Java 236

Chapitre 27
Le pattern Template Method

- 1. Description 241
- 2. Exemple 241
- 3. Structure 245
 - 3.1 Diagramme de classes. 245
 - 3.2 Participants 246
 - 3.3 Collaborations. 246
- 4. Domaines d'application 247
- 5. Exemple en Java 247

Chapitre 28
Le pattern Visitor

- 1. Description 251
- 2. Exemple 251
- 3. Structure 255
 - 3.1 Diagramme de classes. 255
 - 3.2 Participants 256
 - 3.3 Collaborations. 256
- 4. Domaines d'application 257
- 5. Exemple en Java 257

Partie 5 : Application des patterns

Chapitre 29

Compositions et variations de patterns

1. Préliminaire	263
2. Le pattern Pluggable Factory	264
2.1 Introduction	264
2.2 Structure	269
2.3 Exemple en Java	270
3. Reflective Visitor	277
3.1 Discussion	277
3.2 Structure	281
3.3 Exemple en Java	283
4. Le pattern Multicast	290
4.1 Description et exemple	290
4.2 Structure	293
4.3 Exemple en Java	294
4.4 Discussion : comparaison avec le pattern Observer	301

Chapitre 30

Le pattern composite MVC

1. Introduction au problème	303
2. Le pattern composite MVC	304
3. Le framework Vaadin	311
4. Exemple en Java	312
4.1 Introduction	312
4.2 Architecture	313
4.3 Étude du code	315

Chapitre 31

Les patterns dans la conception de logiciels

- 1. Modélisation et conception avec les patterns de conception 327
- 2. Autres apports des patterns de conception 330
 - 2.1 Un référentiel commun 330
 - 2.2 Un ensemble récurrent de techniques de conception. 330
 - 2.3 Un outil pédagogique de l'approche à objets 330

Partie 6 : Annexe

Annexe 1

Java avancé et conception par objets

- 1. Les concepts avancés de la programmation par objets 331
 - 1.1 Le typage des variables 331
 - 1.2 La liaison dynamique 332
 - 1.3 La surcharge des méthodes. 334
 - 1.4 La généricité 337
 - 1.4.1 La notion de classe générique. 337
 - 1.4.2 L'instanciation des paramètres de type 341
 - 1.4.3 La généricité et l'héritage 347
 - 1.5 Les interfaces. 349
 - 1.5.1 La réalisation des interfaces 349
 - 1.5.2 La spécialisation des interfaces 350
 - 1.5.3 La différence entre les classes abstraites
et les interfaces 350
 - 1.6 Les classes internes : un support pour
la composition d'objets. 351
- 2. Les principes de la conception par objets 353
 - 2.1 La réification 353
 - 2.2 La conception modulaire 354

14 _____ Design Patterns en Java

Les 23 modèles de conception

2.3	L'abstraction	356
2.4	La réutilisation des classes	358

Annexe 2 **Exercices**

1.	Énoncés des exercices	361
1.1	Création de cartes de paiement	361
1.1.1	Création en fonction du client	361
1.1.2	Création à l'aide d'une fabrique.	362
1.2	Autorisation des cartes de paiement	362
1.3	Système de fichiers	362
1.4	Browser graphique d'objets	363
1.5	États de la vie professionnelle d'une personne	364
1.6	Cache d'un dictionnaire persistant d'objets	364
2.	Correction des exercices	367
2.1	Création de cartes de paiement	367
2.1.1	Création en fonction du client	367
2.1.2	Création à l'aide d'une fabrique.	368
2.2	Autorisation des cartes de paiement	368
2.3	Système de fichiers	369
2.4	Browser graphique d'objets	375
2.5	États de la vie professionnelle d'une personne	377
2.6	Cache d'un dictionnaire persistant d'objets	378
	Index	381

Chapitre 4

Le pattern Abstract Factory

1. Description

Le but du pattern `Abstract Factory` est la création d'objets regroupés en familles sans devoir connaître les classes concrètes destinées à la création de ces objets.

2. Exemple

Le système de vente de véhicules gère des véhicules fonctionnant à l'essence et des véhicules fonctionnant à l'électricité. Cette gestion est confiée à l'objet `Catalogue` qui crée de tels objets.

Pour chaque produit, nous disposons d'une classe abstraite, d'une sous-classe concrète décrivant la version du produit fonctionnant à l'essence et d'une sous-classe décrivant la version du produit fonctionnant à l'électricité. Par exemple, à la figure 4.1, pour l'objet scooter, il existe une classe abstraite `Scooter` et deux sous-classes concrètes `ScooterÉlectricité` et `ScooterEssence`.

L'objet `Catalogue` peut utiliser ces sous-classes concrètes pour instancier les produits. Cependant si, par la suite, de nouvelles familles de véhicules doivent être prises en compte par la suite (diesel ou mixte essence-électricité), les modifications à apporter à l'objet `Catalogue` peuvent être assez lourdes.

Le pattern `Abstract Factory` résout ce problème en introduisant une interface `FabriqueVehicule` qui contient la signature des méthodes pour définir chaque produit. Le type de retour de ces méthodes est constitué par l'une des classes abstraites de produit. Ainsi, l'objet `Catalogue` n'a pas besoin de connaître les sous-classes concrètes et reste indépendant des familles de produit.

Une sous-classe d'implantation de `FabriqueVehicule` est introduite pour chaque famille de produit, à savoir les sous-classes `FabriqueVehiculeÉlectricité` et `FabriqueVehiculeEssence`. Une telle sous-classe implante les opérations de création du véhicule appropriée pour la famille à laquelle elle est associée.

L'objet `Catalogue` prend alors pour paramètre une instance répondant à l'interface `FabriqueVehicule`, c'est-à-dire soit une instance de `FabriqueVehiculeÉlectricité`, soit une instance de `FabriqueVehiculeEssence`. Avec une telle instance, le catalogue peut créer et manipuler des véhicules sans devoir connaître les familles de véhicules et les classes concrètes d'instanciation correspondantes.

L'ensemble des classes du pattern Abstract Factory pour cet exemple est détaillé à la figure 4.1.

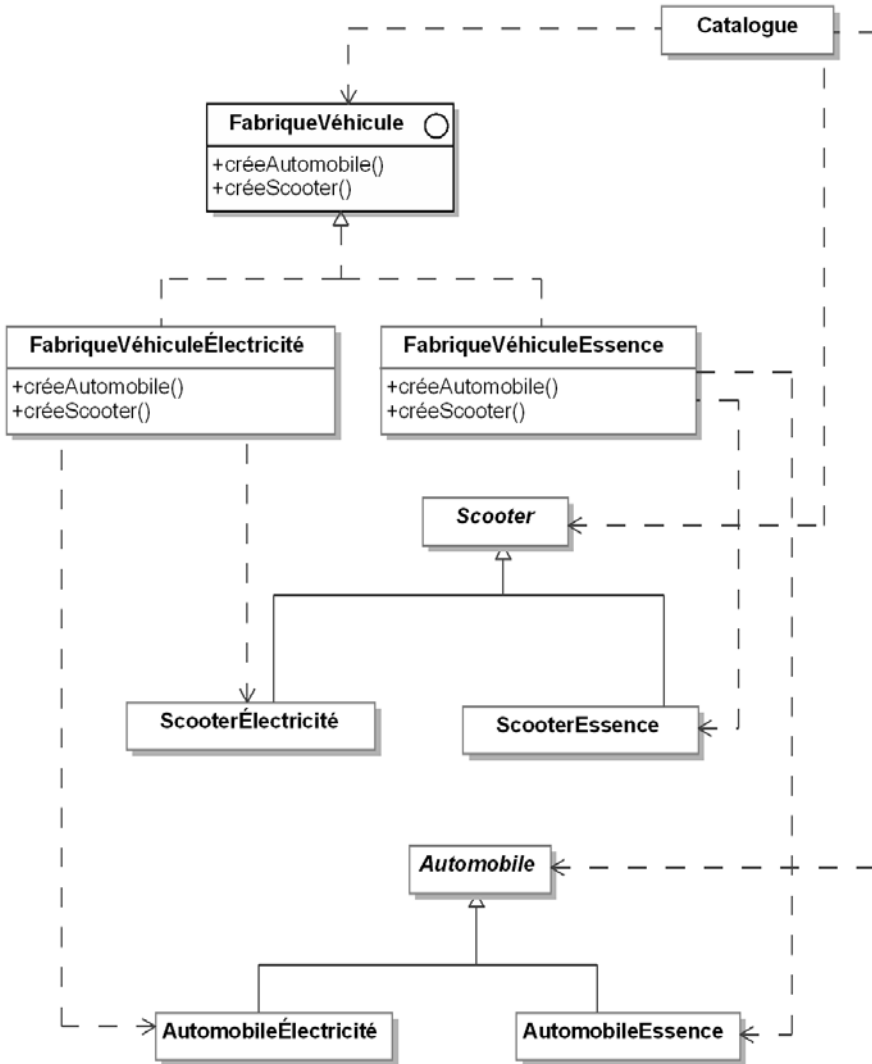


Figure 4.1 - Le pattern Abstract Factory appliqué à des familles de véhicules

3. Structure

3.1 Diagramme de classes

La figure 4.2 détaille la structure générique du pattern.

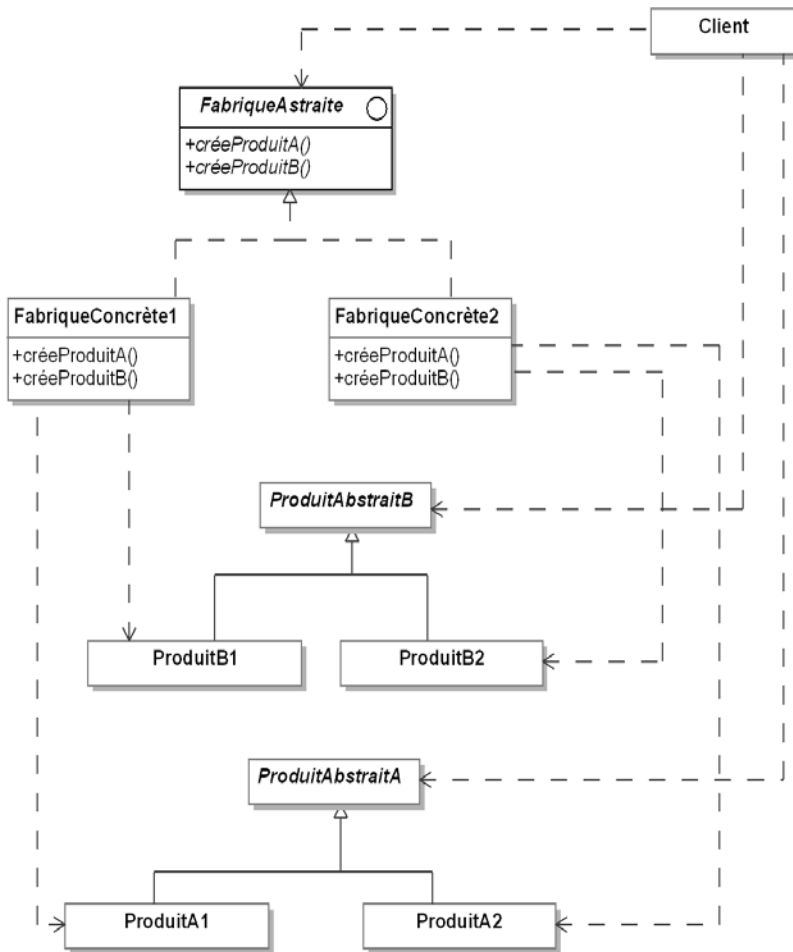


Figure 4.2 - Structure du pattern Abstract Factory

3.2 Participants

Les participants au pattern sont les suivants :

- `FabriqueAbstraite` (`FabriqueVehicule`) est une interface spécifiant les signatures des méthodes créant les différents produits.
- `FabriqueConcret1`, `FabriqueConcret2` (`FabriqueVehiculeElectricite`, `FabriqueVehiculeEssence`) sont les classes concrètes implantant les méthodes créant les produits pour chaque famille de produits. Connaissant la famille et le produit, elles sont capables de créer une instance du produit pour cette famille.
- `ProduitAbstraitA` et `ProduitAbstraitB` (`Scoter` et `Automobile`) sont les classes abstraites des produits indépendamment de leur famille. Les familles sont introduites dans leurs sous-classes concrètes.
- `Client` est la classe qui utilise l'interface de `FabriqueAbstraite`.

3.3 Collaborations

La classe `Client` utilise une instance de l'une des fabriques concrètes pour créer ses produits au travers de l'interface de `FabriqueAbstraite`.

■ Remarque

Normalement, il ne faut créer qu'une seule instance des fabriques concrètes, celle-ci pouvant être partagée par plusieurs clients.

4. Domaines d'utilisation

Le pattern est utilisé dans les domaines suivants :

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

5. Exemple en Java

Nous introduisons maintenant un petit exemple d'utilisation du pattern écrit en Java. Le code Java correspondant à la classe abstraite `Automobile` et ses sous-classes est donné à la suite. Il est très simple, décrit les quatre attributs des automobiles ainsi que la méthode `afficheCaracteristiques` qui permet de les afficher.

```
public abstract class Automobile
{
    protected String modele;
    protected String couleur;
    protected int puissance;
    protected double espace;

    public Automobile(String modele, String couleur, int
        puissance, double espace)
    {
        this.modele = modele;
        this.couleur = couleur;
        this.puissance = puissance;
        this.espace = espace;
    }

    public abstract void afficheCaracteristiques();
}

public class AutomobileElectricite extends Automobile
{
    public AutomobileElectricite(String modele, String
        couleur, int puissance, double espace)
    {
        super(modele, couleur, puissance, espace);
    }

    public void afficheCaracteristiques()
    {
        System.out.println(
            "Automobile électrique de modele : " + modele +
            " de couleur : " + couleur + " de puissance : " +
            puissance + " d'espace : " + espace);
    }
}
```