Chapitre 4-2
Le pattern Chain of Responsibility

1. Description

Le pattern `Chain of Responsibility` construit une chaîne d'objets telle que si un objet de la chaîne ne peut pas répondre à une requête, il puisse la transmettre à son successeur et ainsi de suite jusqu'à ce que l'un des objets de la chaîne y réponde.

2. Exemple

Nous nous plaçons dans le cadre de la vente de véhicules d'occasion. Lorsque le catalogue de ces véhicules est affiché, l'utilisateur peut demander une description de l'un des véhicules mis en vente. Si une telle description n'a pas été fournie, le système doit alors renvoyer la description associée au modèle de ce véhicule. Si à nouveau, cette description n'a pas été fournie, il convient de renvoyer la description associée à la marque du véhicule. Une description par défaut est renvoyée s'il n'y a pas non plus de description associée à la marque.

Ainsi, l'utilisateur reçoit la description la plus précise qui est disponible dans le système.

Le pattern `Chain of Responsibility` fournit une solution pour mettre en œuvre ce mécanisme. Celle-ci consiste à lier les objets entre eux du plus spécifique (le véhicule) au plus général (la marque) pour former la chaîne de responsabilité. La requête de description est transmise le long de cette chaîne jusqu'à ce qu'un objet puisse la traiter et renvoyer la description.

Le diagramme d'objets UML de la figure 4-2.1 illustre cette situation et montre les différentes chaînes de responsabilité (de la gauche vers la droite).

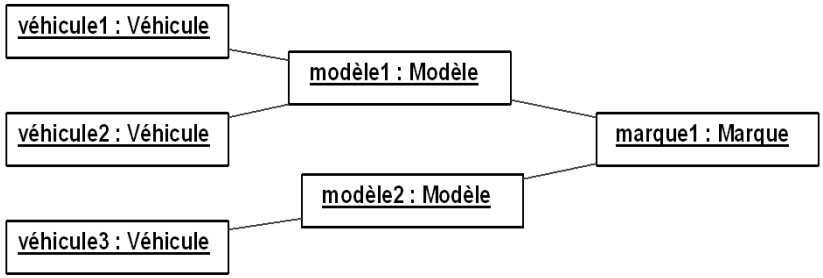

Figure 4-2.1 - Diagramme d'objets de véhicules, modèles et marques avec les liens de la chaîne de responsabilité

La figure 4-2.2 représente le diagramme des classes du pattern `Chain of Responsibility` appliqué à l'exemple. Les véhicules, modèles et marques sont décrits par des sous-classes concrètes de la classe `ObjetBase`. Cette classe abstraite introduit l'association `suivant` qui implante la chaîne de responsabilité. Elle introduit également trois méthodes :

– `getDescription` est une méthode abstraite. Elle est implantée dans les sous-classes concrètes. Cette implantation doit retourner soit la description si elle existe soit la valeur `null` dans le cas contraire.

– `descriptionParDéfaut` retourne une valeur de description par défaut, valable pour tous les véhicules du catalogue.

– `donneDescription` est la méthode publique destinée à l'utilisateur. Elle invoque la méthode `getDescription`. Si le résultat est `null`, alors s'il y a un objet `suivant`, sa méthode `donneDescription` est invoquée à son tour sinon c'est la méthode `descriptionParDéfaut` qui est utilisée.

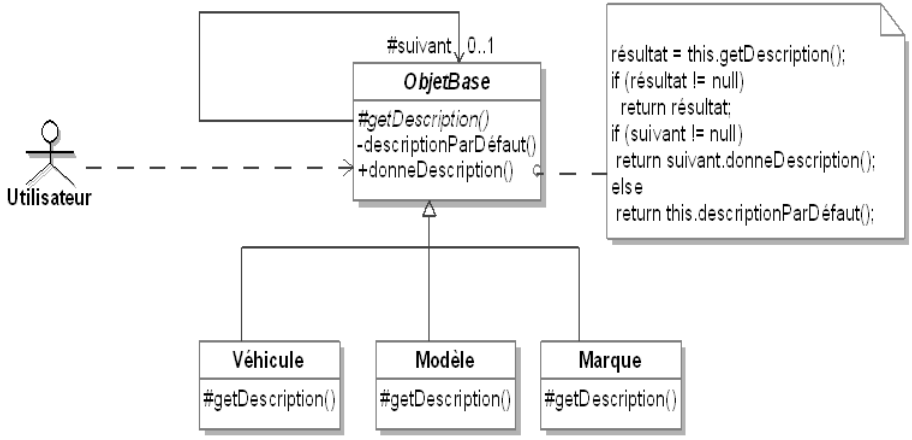

Figure 4-2.2 - Le pattern `Chain of Responsibility` *pour organiser la description de véhicules d'occasion*

La figure 4-2.3 montre un diagramme de séquence qui est un exemple de requête d'une description basée sur le diagramme d'objets de la figure 4-2.1.

Dans cet exemple, ni le `véhicule1`, ni le `modèle1` ne possèdent de description. Seule `marque1` possède une description qui est donc utilisée pour le `véhicule1`.

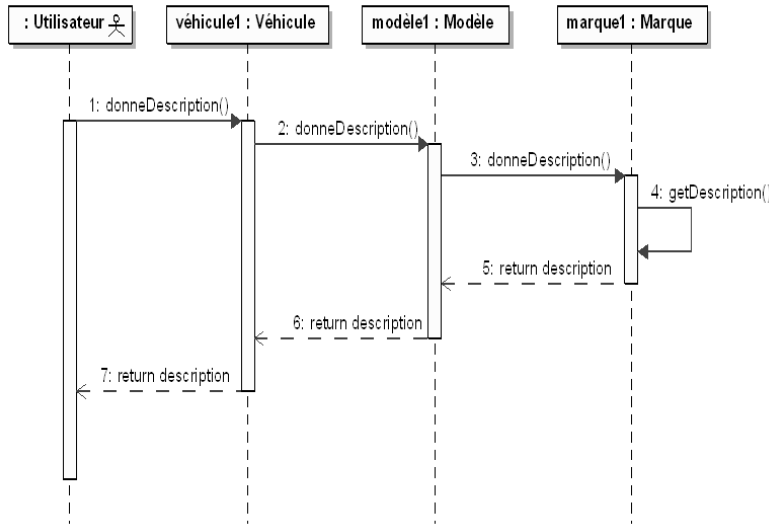

Figure 4-2.3 - Diagramme de séquence illustrant sur un exemple le pattern ***`Chain of Responsibility`***

3. Structure

3.1 Diagramme de classes

La figure 4-2.4 détaille la structure générique du pattern.

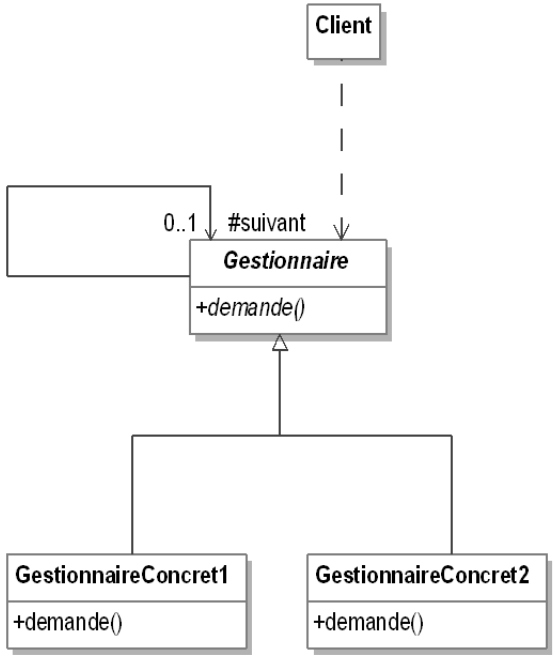

Figure 4-2.4 - Structure du pattern `Chain of Responsibility`

3.2 Participants

Les participants au pattern sont les suivants :

– `Gestionnaire` (`ObjetBase`) est une classe abstraite qui implante sous forme d'une association la chaîne de responsabilité ainsi que l'interface des requêtes.
– `GestionnaireConcret1` et `GestionnaireConcret2` (`Véhicule`, `Modèle` et `Marque`) sont les classes concrètes qui implantent le traitement des requêtes en utilisant la chaîne de responsabilité si elles ne peuvent pas les traiter.
– `Client` (`Utilisateur`) initie la requête initiale auprès d'un objet de l'une des classes `GestionnaireConcret1` ou `GestionnaireConcret2`.

3.3 Collaborations

Le client effectue la requête initiale auprès d'un gestionnaire. Cette requête est propagée le long de la chaîne de responsabilité jusqu'au moment où l'un des gestionnaires la traite.

4. Domaines d'application

Le pattern est utilisé dans les cas suivants :

– Une chaîne d'objets gère une requête selon un ordre qui est défini dynamiquement.
– La façon dont une chaîne d'objets gère une requête ne doit pas être connue de ses clients.

5. Exemple en C#

Nous introduisons maintenant l'exemple en langage C#. La classe `ObjetBase` est décrite à la suite. Elle implante la chaîne de responsabilité par la propriété `suivant`. Les autres méthodes correspondent aux spécifications introduites dans la figure 4-2.2.

```
using System;

public abstract class ObjetBase
{
 public ObjetBase suivant { protected get; set; }

 private string descriptionParDefaut()
 {
   return "description par défaut";
 }

 protected abstract string description { get; }

 public string donneDescription()
 {
   string resultat;
   resultat = this.description;
   if (resultat != null)
     return resultat;
   if (suivant != null)
     return suivant.donneDescription();
   else
     return this.descriptionParDefaut();
 }
}
```

Les trois sous-classes concrètes de la classe `ObjetBase` sont `Vehicule`, `Modele` et `Marque`, leur code source C# est présenté à la suite. La classe `Vehicule` gère une description simple fournie au moment de sa construction (le paramètre `null` est utilisé en cas d'absence de description).

```
using System;

public class Vehicule : ObjetBase
{
 protected string laDescription;
```