|  |
| --- |
| DEFOIS, FARISSE, HAMANI, LOPPY, VERGINE |
| Documentation D’un Design Pattern |
| Adapter |

|  |
| --- |
| Romain DEFOIS  26/10/2017 |

Table des matières

[Nom du Pattern 2](#_Toc497744406)

[Description du Pattern 2](#_Toc497744407)

[Exemple de diagramme UML (Problème et Solution) 2](#_Toc497744408)

[Les Domaines d’Application du Pattern 3](#_Toc497744409)

[Exemple de Code en C# 3](#_Toc497744410)

# Nom du Pattern

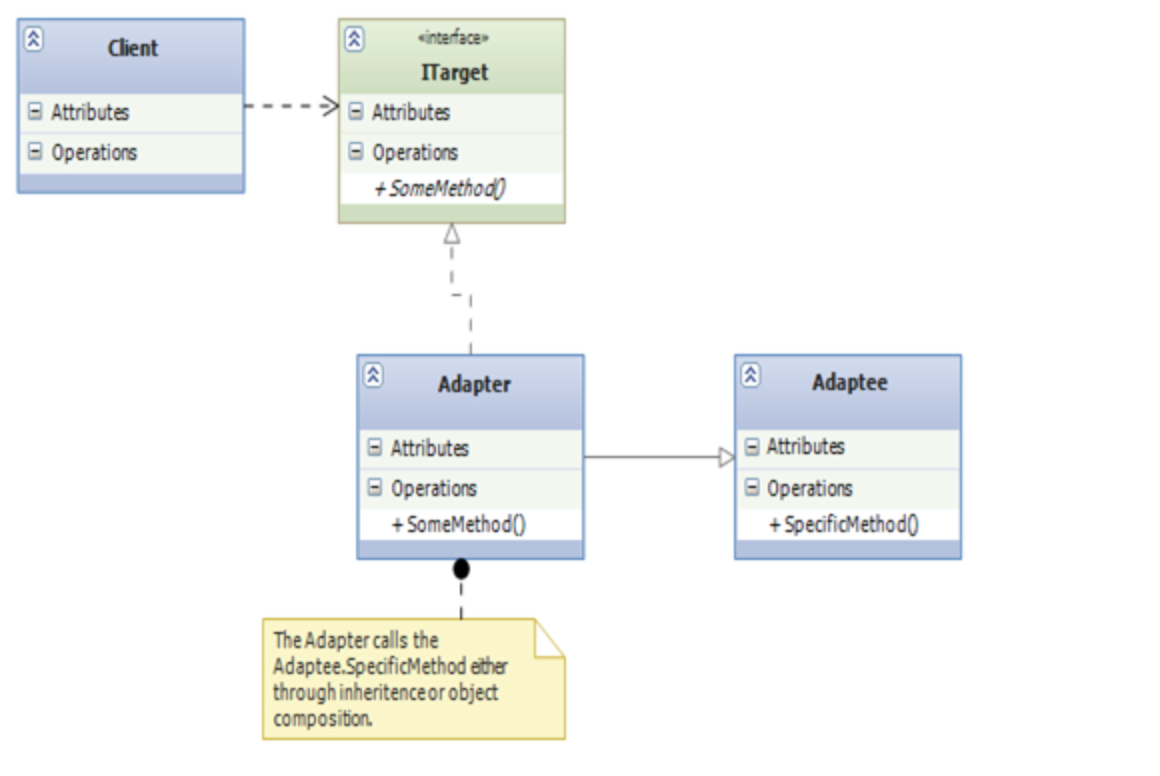
Adaptateur

# Description du Pattern

En génie logiciel, adaptateur (ou wrapper[1](https://fr.wikipedia.org/wiki/Adaptateur_(patron_de_conception)#cite_note-1)) est un patron de conception (*design pattern*) de type structure (*structural*). Il permet de convertir l'interface d'une classe en une autre interface que le client attend.

L’adaptateur fait fonctionner ensemble des classes qui n'auraient pas pu fonctionner sans lui, à cause d'une incompatibilité d'interfaces[2](https://fr.wikipedia.org/wiki/Adaptateur_(patron_de_conception)#cite_note-2).

# Exemple de diagramme UML (Problème et Solution)



* ITarget : C'est l'interface qui est utilisée par le client pour réaliser la fonctionnalité
* Adaptee : C'est la fonctionnalité que le client désire mais son interface n'est pas compatible avec le client
* Client : C'est la classe qui veut réaliser certaines fonctionnalités en utilisant le code de l'adaptee.
* Adapter : C'est la classe qui implémenterait ITarget et appellerait le code Adaptee que le client veut appeler.

---------------Détecter la langueAfrikaansAlbanaisAllemandAmhariqueAnglaisArabeArménienAzériBasqueBengaliBiélorusseBirmanBosniaqueBulgareCatalanCebuanoChinois (simplifié)Chinois (traditionnel)CinghalaisCoréenCorseCréole haïtienCroateDanoisEspagnolEspérantoEstonienFinnoisFrançaisFrison occidentalGaélique écossaisGalicienGalloisGéorgienGoudjeratiGrecHaoussaHawaïenHébreuHindiHmongHongroisIgboIndonésienIrlandaisIslandaisItalienJaponaisJavanaisKannadaKazakhKhmerKirghizeKurdeLaoLatinLettonLituanienLuxembourgeoisMacédonienMalaisMalayalamMalgacheMaltaisMaoriMaratheMongolNéerlandaisNépalaisNorvégienNyanjaOurdouOuzbekPachtoPendjabiPersanPolonaisPortugaisRoumainRusseSamoanSerbeShonaSindhiSlovaqueSlovèneSomaliSotho du SudSoundanaisSuédoisSwahiliTadjikTagalogTamoulTchèqueTélougouThaïTurcUkrainienVietnamienXhosaYiddishYorubaZoulouAnglais

# Les Domaines d’Application du Pattern

Une classe GestionDesStocks permettra de manipuler depuis d’autres endroits dans une chaîne de gestion commerciale l’ensemble des modules afférents à la gestion des stocks. Bien implémentée cette DP permet une incroyable modularité et la réutilisation simplifiée du code dans des applications complexes.

# Exemple de Code en C#

Ce code structurel illustre le modèle Adapter qui map l'interface d'une classe sur une autre afin qu'elles puissent fonctionner ensemble. Ces classes incompatibles peuvent provenir de différentes bibliothèques ou frameworks.

1. using System;
3. namespace DoFactory.GangOfFour.Adapter.Structural
4. {
5. /// <summary>
6. /// MainApp startup class for Structural
7. /// Adapter Design Pattern.
8. /// </summary>
9. class MainApp
10. {
11. /// <summary>
12. /// Entry point into console application.
13. /// </summary>
14. static void Main()
15. {
16. // Create adapter and place a request
17. Target target = new Adapter();
18. target.Request();
20. // Wait for user
21. Console.ReadKey();
22. }
23. }
25. /// <summary>
26. /// The 'Target' class
27. /// </summary>
28. class Target
29. {
30. public virtual void Request()
31. {
32. Console.WriteLine("Called Target Request()");
33. }
34. }
36. /// <summary>
37. /// The 'Adapter' class
38. /// </summary>
39. class Adapter : Target
40. {
41. private Adaptee \_adaptee = new Adaptee();
43. public override void Request()
44. {
45. // Possibly do some other work
46. //  and then call SpecificRequest
47. \_adaptee.SpecificRequest();
48. }
49. }
51. /// <summary>
52. /// The 'Adaptee' class
53. /// </summary>
54. class Adaptee
55. {
56. public void SpecificRequest()
57. {
58. Console.WriteLine("Called SpecificRequest()");
59. }
60. }
61. }