|  |
| --- |
| Modèle de conception réutilisables |
| Command pattern |
| Rapport d’implémentation |
|  |
| **Brito Carvalho Bruno Ricardo – Decorvet Grégoire – Ngo Quang Dung – Schweizer Thomas** |
| **24/06/2014** |

|  |
| --- |
|  |

Table des matières

[1 Introduction 2](#_Toc391302190)

[2 Motivation 2](#_Toc391302191)

[3 Procédure 2](#_Toc391302192)

[4 Application 4](#_Toc391302193)

[5 Diagramme de classes 4](#_Toc391302194)

[6 Conclusion 4](#_Toc391302195)

# Introduction

# Motivation

En génie logiciel, Commande est un patron de conception (design pattern) de type comportemental qui encapsule la notion d’invocation. Il permet de séparer complètement le code initiateur de l’action, du code de l’action elle-même.

Ce patron de conception est souvent utilisé dans les interfaces graphiques où, par exemple, un élément de menu peut être connecté à différentes Commandes de façon à ce que l’objet d’élément de menu n’ait pas besoin de connaître les détails de l’action effectuée par la Commande.

(Source : Wikipedia)

Donc, le design pattern permet de :

* Encapsuler une requête sous la forme d’objet
* Paramétrer facilement des requêtes diverses
* Permettre des opérations réversibles

# Procédure

* Command : définit l’interface d’une commande avec une méthode « execute() » qui demande au « Receiver » correspondant de réaliser la tâche
* Invoker : déclenche la commande en appelant la méthode « execute() » d’un objet « Command »
* Receiver : reçoit la commande et réalise les opérations associées. Chaque objet « Command » concret possède un lien avec un objet « Receiver »
* Client : crée des « ConcreteCommand » et définit un « Receiver » pour la commande. « ConcreteCommand » est le lien entre des actions à réaliser et le « Receiver »

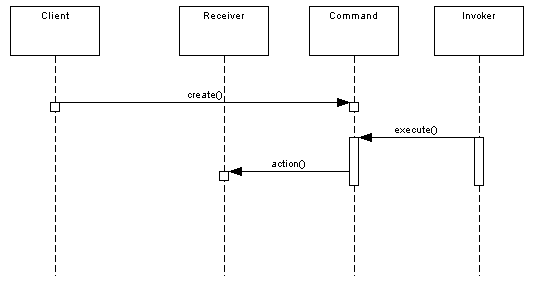


Diagramme de séquence

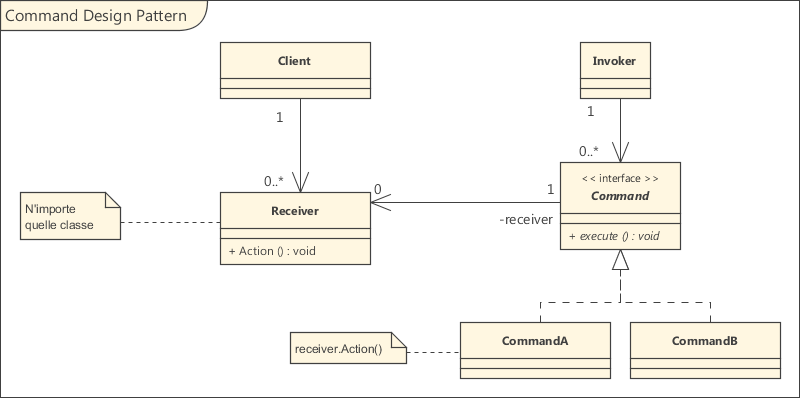


Diagramme de classes

# Application

# Diagramme de classes

# Conclusion