# LE PATTERN COMMANDE

## Intention

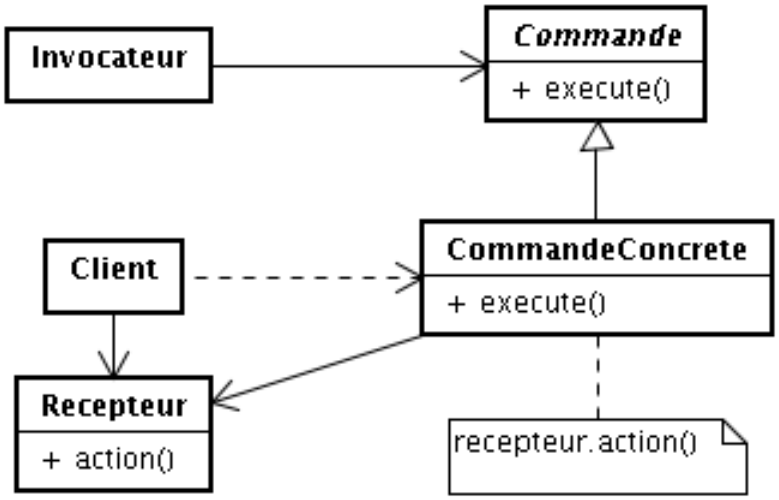
Encapsuler une requête comme un objet, autorisant ainsi le paramétrage des clients par différentes requêtes, files d'attente et récapitulatifs de requêtes, et de plus permettant la réversion des opération.

## Indications d'utilisation

Il convient d'utiliser le modèle Commande, lorsque l'on veut :

- Introduire dans des objets, sous la forme de paramètres, des actions à effectuer. Un tel paramétrage peut s'exprimer en langage procédural par l'intermédiaire de fonctions callback, c'est-à-dire, des fonctions enregistrées dans un certain contexte pour être appelées ultérieurement. Les commandes sont la version orientée objet des fonctions callbacks.
- Spécifier, mettre en file d'attente, et exécuter les requêtes a différents instants.
- Assurer le service « défaire » (undo). L'opération Execute de Commande peut stocker un état pour inverser son effet dans la commande elle-même. L'interface de Commande doit posséder une opération supplémentaire, **reversion**, qui supprime les effets d'un précédent appel à Execute. Les commandes exécutées sont stockées dans une liste historique.
- Permettre une mémorisation de modifications, afin de pouvoir les appliquer à nouveau après un éventuel crash. En ajoutant à l'interfaceCommande les opération **charger** et **stocker**, on peut réaliser un enregistrement persistant des modifications. Récupérer d'un crash implique de recharger les commandes enregistrées depuis le disque et de les exécuter à nouveau à l'aide de l'opération **execute**.
- Structurer un système autour d'opération de haut niveau, construites à l'aide de primitives. Une structure de ce types est courante dans les systèmes d'information qui autorisent les transactions. Une transaction encapsule un ensemble de modifications à effectuer sur des données. Le patternCommande fournit le moyen de modéliser les transactions. Les commandes ont une interface commune qui permet d'invoquer toutes les transactions de la même manière. Le pattern favorise également l'extension du système par apport de nouvelles transactions.

## Structure




## Constituants

*Commande*

- Déclare une interface pour exécuter une opération.

CommandeConcrete

- Définit une astreinte entre un objet récepteur et une action
- Concrétise execute par invocation des opérations adéquates de récepteur.

*Client*

- Créer un objet CommandeConcrete et positionne son récepteur.

*Invocateur*

- Demande à la commande d'entreprendre la requête.

*Recepteur*

- Sait comment effectuer les opérations associées avec le traitement d'une requête. Tout type de classe peut servir de récepteur.

## Collaborations

Le client crée un objet CommandeConcrete et établit les spécifications de son récepteur.

Un objet Invocateur stocke l'objet CommandeConcrete.

L'invocateur lance une requête en appelant la fonction exécute de la commande. Lorsque les commandes doivent pouvoir être inversées, CommandeConcrete, avant d'invoquer execute, stocke l'état qui permettra de renverser l'exécution de la commande.

L'objet CommandeConcrete invoque les opérations de son récepteur pour satisfaire la requête.