目的・範囲: *Objective & Scope*	Design Pattern(Command パターン)
分類名: *Classification Name*	振る舞いに関するパターン
著作者: *Author*	鈴木　皓太
実施日: *Enforcement day*	2014年6月7日
バージョン: *Version*	
初版発行日: *Original Release*	2014年6月7日
現行版発行日: *Current Release*	2014年6月7日
キーワード: *Key Words*	手続きの管理 コマンド・命令　自体をオブジェクトとして分離
背景情報: *Background*	

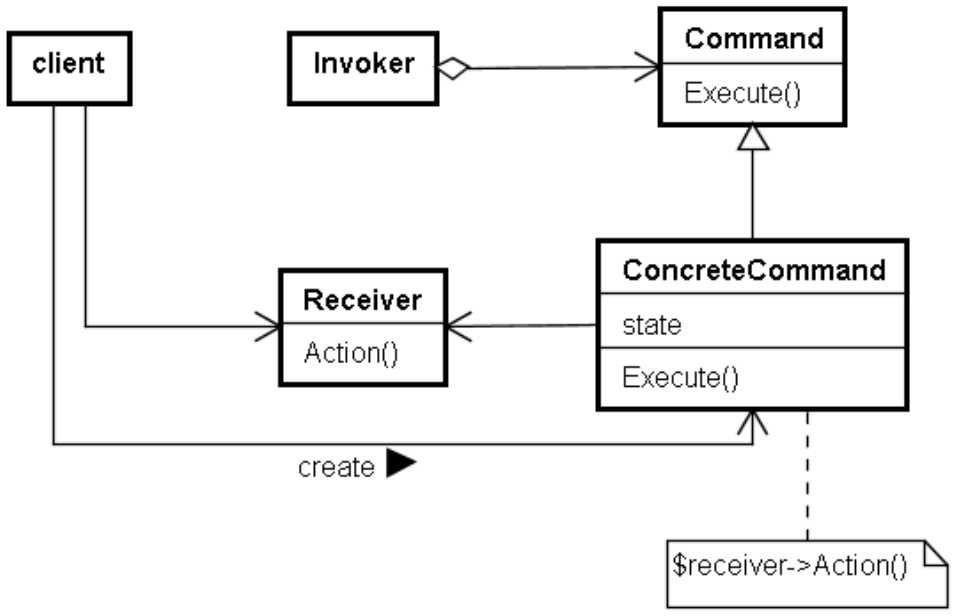
◇目的
・一つ以上の「命令・動作・コマンド・振る舞い」 を、カプセル化して、オブジェクトにする。
例えば、
・「データベースにInsert文を流す」とか「テーブルをtruncateする」とか「印刷する」のような、そのままインスタンスとして成立できる程度に抽象度の低い「ある動作」そのもの と、
・その「ある動作」に必要なパラメータ
を、カプセル化したもの。

上記のオブジェクトを中心に、実装レベルまで、オブジェクトを追加したもの全体が、「Commandパターン」

◇効果
・動作とかをオブジェクトとしているので、UNDO/REDOや、Job Queueを、簡単に実装できる。
・動作のネスト・再帰（結果的に木構造になる方）も、簡単に実装できる。

◇背景
「『コマンド群・クラス群・メソッド群』などのクラスを実装したときに、抽象化をした結果」。

◇Commandパターンの実際のコードと考え方
Command : 命令を実行する用のAPIを定義する。
ConcreteCommand : Commandクラスのサブクラス。定義されてるAPIを実装する
Invoker : 命令実行の要求を出すクラス。
Receiver : 任意の「命令・動作・コマンド・振る舞い」のクラスを実装する場所。



◇問題点の改善

変数のスコープをprivateとすべきかprotectedにすべきか、難しい部分だが、デフォルトではprivateでいい。

◇Commandパターンのまとめ

C++でいうところの、「一定の動作の塊を、namespaceにぶち込む」イメージの、デザインパターン。
「命令・動作・コマンド・振る舞い」ひとつひとつの拡張性とか汎用性とか再利用性とかを、コンスタントに高められる。
「ファイル操作をするコマンド」群とか、「Mathクラス」とか、そういう粒度のものは、とりあえずCommandパターンにしておけ。

◇注意

◇総括

基本用語

用語	説明

※引用文献e-words、WikiPedia