Command

Objective

Encapsulate a request in an object by making it easier to set up customer classes with different queries.

Function

Encapsulate an operation in an object, allowing it to be executed without the need to know its content.

Structure

As shown in figure 1

The Summoner does not know who the Recipient is or the action to be performed, he merely invokes a Command that executes the appropriate action.

The structure that meets this pattern is shown in Figure 1

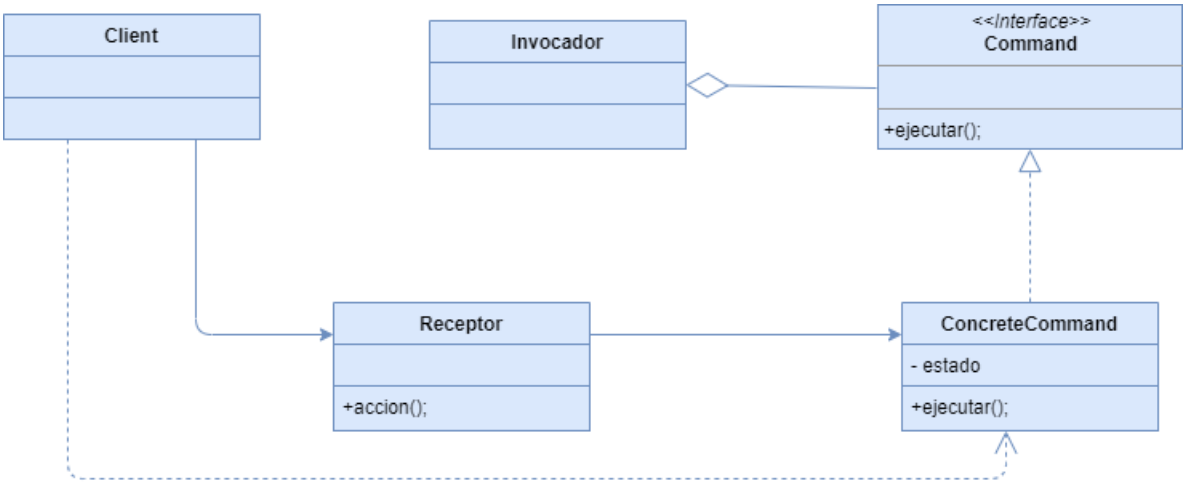

Figure 1: UML Diagram Command Pattern

Applications

The use of the Command pattern is recommended when:

- You have commands that different receivers can handle in different ways.
- A set of high-level commands are available that are implemented through primitive operations.
- You want to specify, stack and execute commands at various times.

- It is necessary to have the option to undo the execution of commands.

- It is necessary to manage audits and records of all changes through commands.

Design Patterns Collaborators

- A composite pattern can be used to implement composite commands or transactions, the use of the memento pattern preserves the state that a command requires in order to undo the effects of its execution. The use of the prototype pattern allows a command to be copied before it is recorded in an action log.

Scope of action

Applied at the object level.

Problem

For the interaction between objects, direct references are handled between them, a requesting object references the methods of a receiving one; by means of conditioning statements (if or switch, case), complexity is increased when trying to add a new receiving class; since the summoning class must be modified by increasing the conditioning statement that discriminates the new class; which contradicts the two principles of object-oriented programming.

Solution

The command design pattern solves this contradiction by creating an abstract class called command, which can be instantiated into a concrete class, allowing to parameterize a command object to be executed for the client class.

Diagram or Implementation

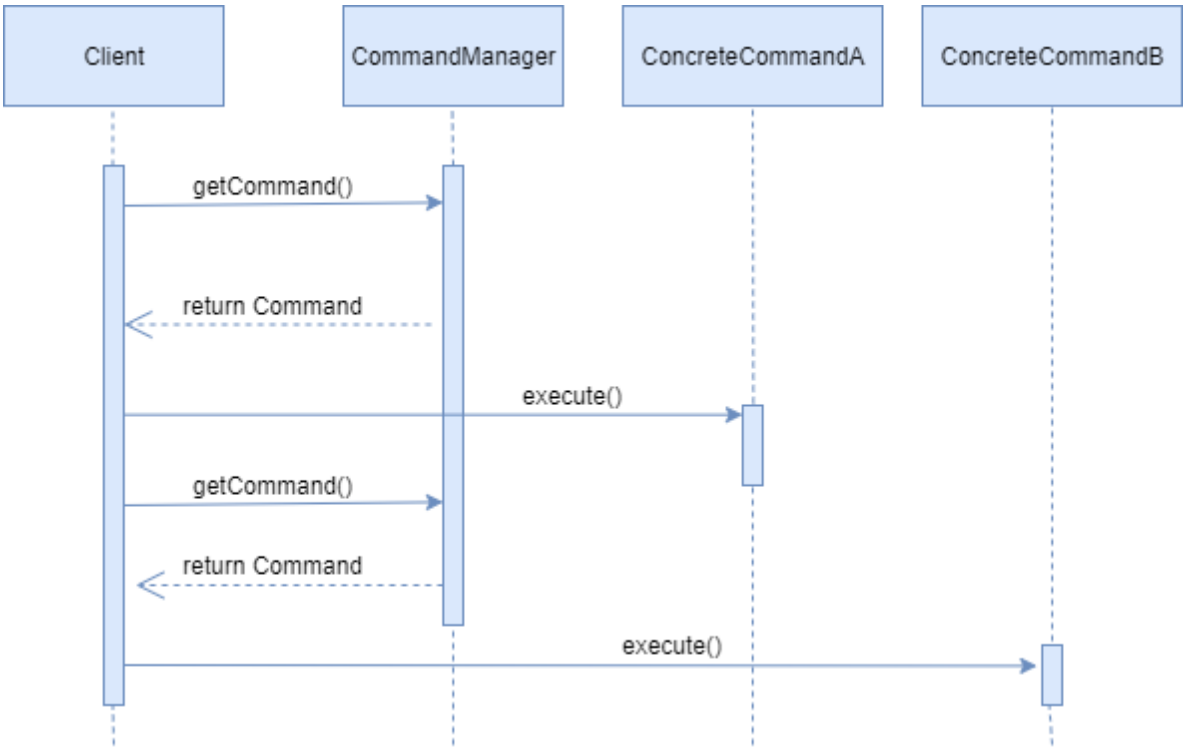

Figure 2: UML Diagram Command Pattern

Figure 2 explains the behaviour of the pattern by means of a sequence diagram.

- The invoker component gets a Command of the CommandManager class.
- The invoker component executes the command.
- The invoker component gets another Command of the CommandManager class.
- The invoker component executes the command.