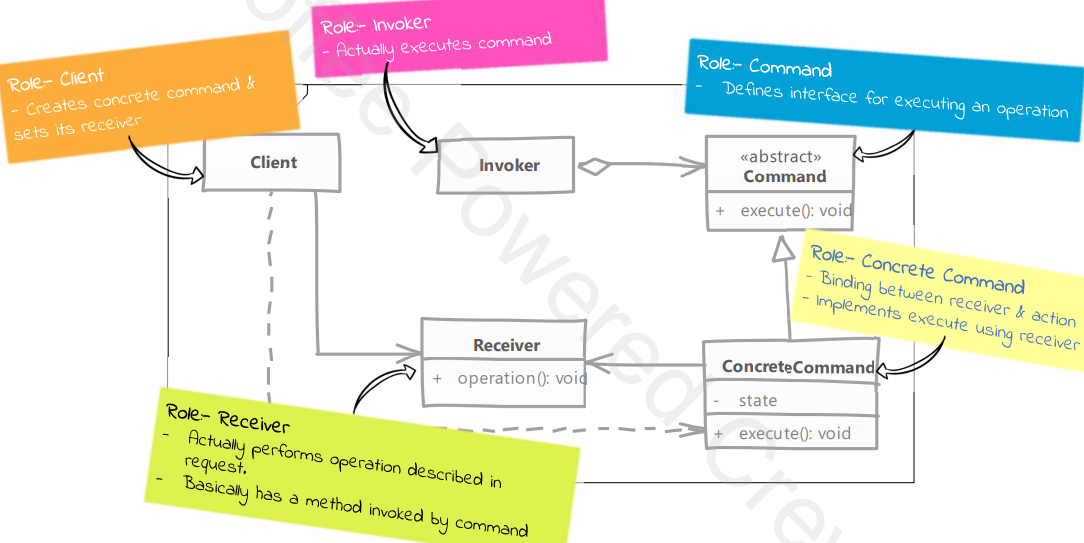
Command

Behavioral Design Pattern

**What is Command:**

1. Represent a request or a method call as an object. Information about parameters passed and the actual operation is encapsulated in an object called command.
2. This object can be stored for later execution or sent to other parts of code.
3. This object can be queued and execute them later

**UML:**



**Implementation steps:**

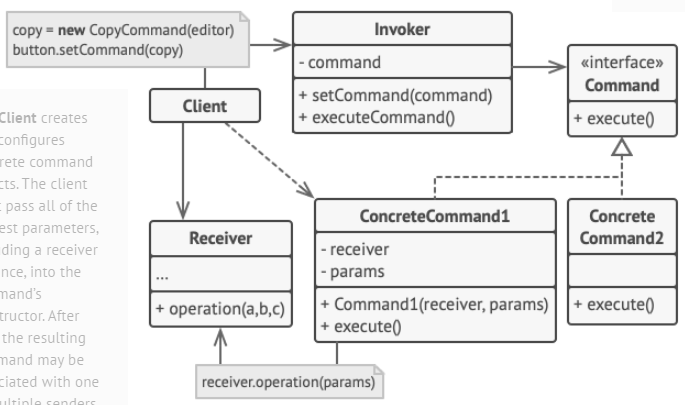
1. Write Command interface:
   1. Define method which executes the command
2. Implement this interface in class for each request or operation type we want to implement. Command should also allow for undo operation if your system needs it.
3. Each concrete command knows exactly which operation it needs. All it needs is parameters for the operation if required and the receiver instance on which operation is invoked.
4. Client creates the command instance and sets up receiver and all required parameters.
5. The command instance is then ready to be sent to other part of code.

**Implementation considerations:**

1. You can support “undo” & “redo” in your commands.
2. Pay attention to size of state maintained by commands, if they going to be queued for a long time.

**Intent:** turns a request into a stand-alone object that contains all information about the request. This transformation lets you pass requests as a method argument, delay or queue a request’s execution, and support undoable operations.

**Structure:**



1. Sender class (Invoker) – responsible for initiating requests. This class must have a field for storing a reference to a command object. The sender triggers that command instead of sending the request directly to the receiver. Isn’t responsible for creating the command object. Usually it gets a pre-created command from the client via the constructor.
2. Command interface – declares a single method for executing the command.
3. Concrete command – implement various kind of requests. Not supposed to perform the work on its own, but rather to pass the call to one of the business logic objects.
4. Receiver – contains some business logic. Almost any object may act as a receiver. Most commands only handle the details of how a request is passed to the receiver, which the receiver itself does the actual work.
5. Client – creates and configures concrete command objects.

**Example:**

