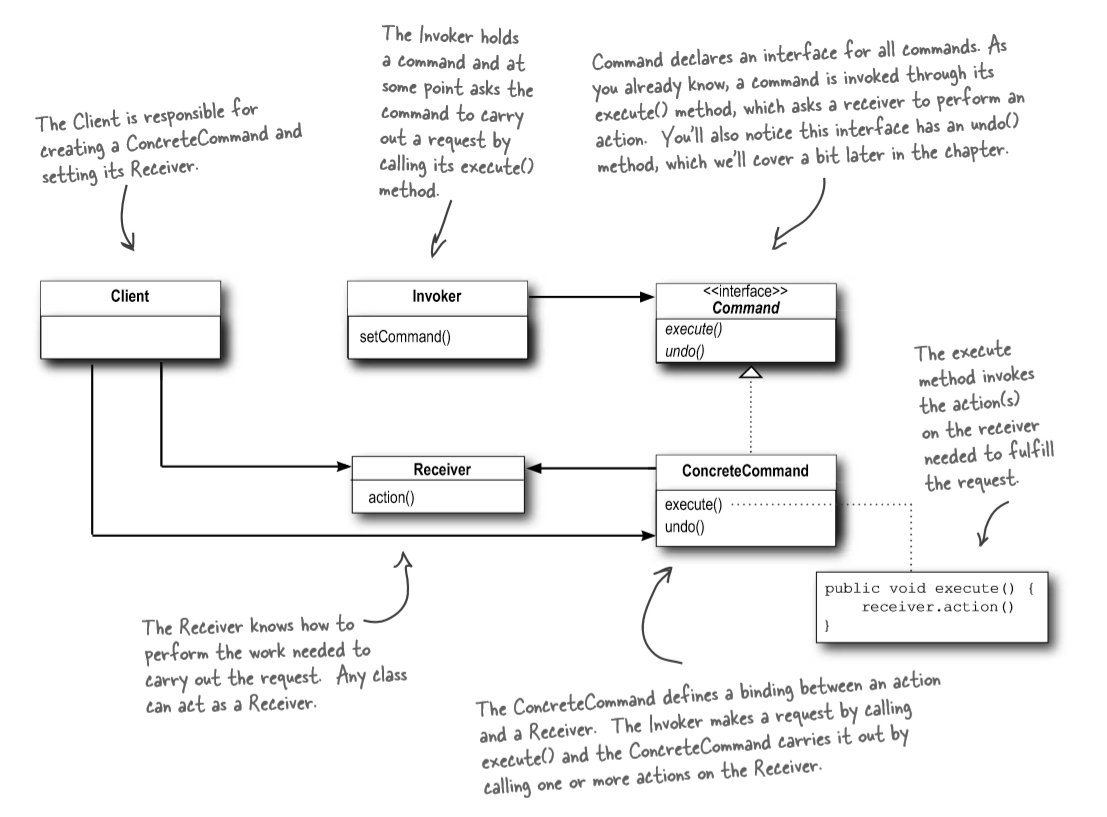
**Command Pattern**

The Command Pattern encapsulates a request as an object, thereby letting you parameterize other objects with different requests, queue or log requests, and support undoable operations.

There are many a times in our application that we want to perform multiple operations on same data. I mean the scenarios when the user has some data and he have an option of doing one out of many operations in that data. Example could be an image processor, I could choose to rotate, flip and/or invert colours of the photo. Also, I could find myself not happy with the action and I may want to undo my action.



* **Command**: This is an interface for executing an action
* **ConcreteCommand**: This object specifies the binding between a Receiver/action taker and an action invoker. This object is responsible for executing corresponding operation on Receiver.
* **Client**: creates a ConcreteCommand object and sets its receiver
* **Invoker**: It will use Command object to carry out the request
* **Receiver**: It contains the real operational logic that need to be performed on the data.

Command pattern is a very good way of decrease the coupling between sender and receiver. The most important thing to remember while implementing the command pattern is that the Command is just a link between sender and receiver. It should only tell the receiver what the sender is expecting. It should never alter the logic of sender and receiver in any way.

<https://www.codeproject.com/Articles/339390/Understanding-and-Implementing-Command-Pattern-in>