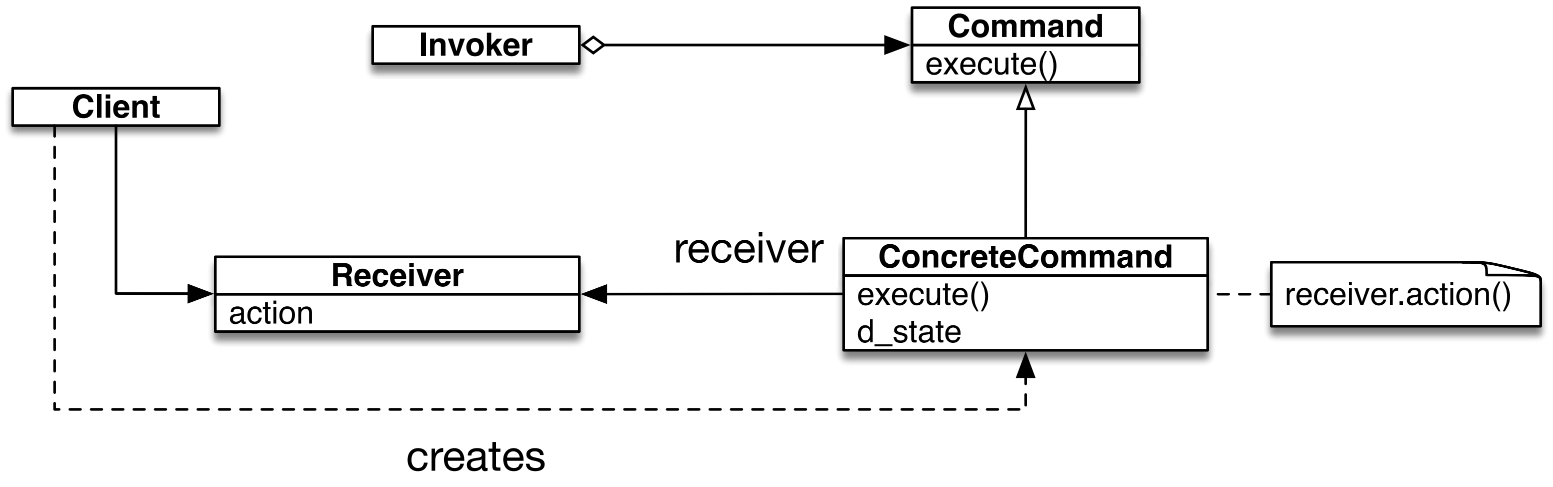
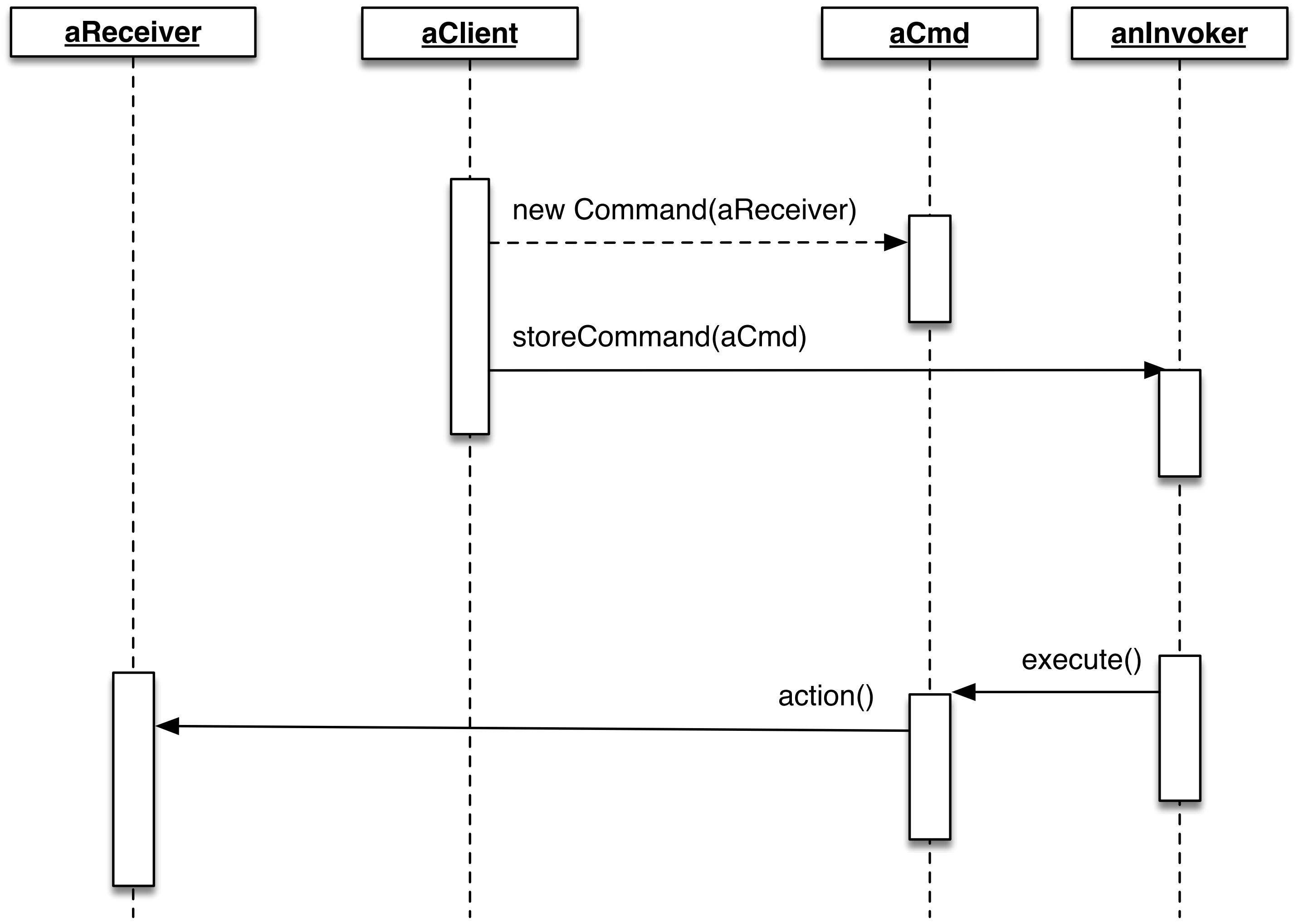
Command / Action / Transaction

Encapsulate a request as an object, thereby letting you parameterize clients with different requests, queue or log requests, and support undo-able operations.





Applicable when you want to
- Parameterize objects by an action (callback) to perform. Commands are OO replacement for callbacks
- Specify, queue, and execute requests at different times. (Command object lifetime is independent of the request)
- Support undo: implement unexecute in Command
- Support logging changes: implement load and store methods. In case of a crash, replay
- Structure a system around high-level operations built on primitive operations (e.g., a system that supports transactions); Command let you model transactions

Command decouples the invoker and the receiver, can be assembled into composite commands, can be manipulated and extended, and adding new commands doesn't require changing existing classes.

In the implementation consider undo / redo, and if the Command has to be copied or not when placed onto a command history (copy if the Command's state changes on execution, like a Delete; only store a reference when the state does not change); and consider using templates for implementing undo / redboale commands.

Client creates a ConcreteCommand object and set its receiver.
Invoker asks the Command to carry out the request.
Receiver knows how to perform the operations associated with carrying out a request (can be any class).
Note the Command decouples the Invoker and the Receiver