Summer Semester 2015

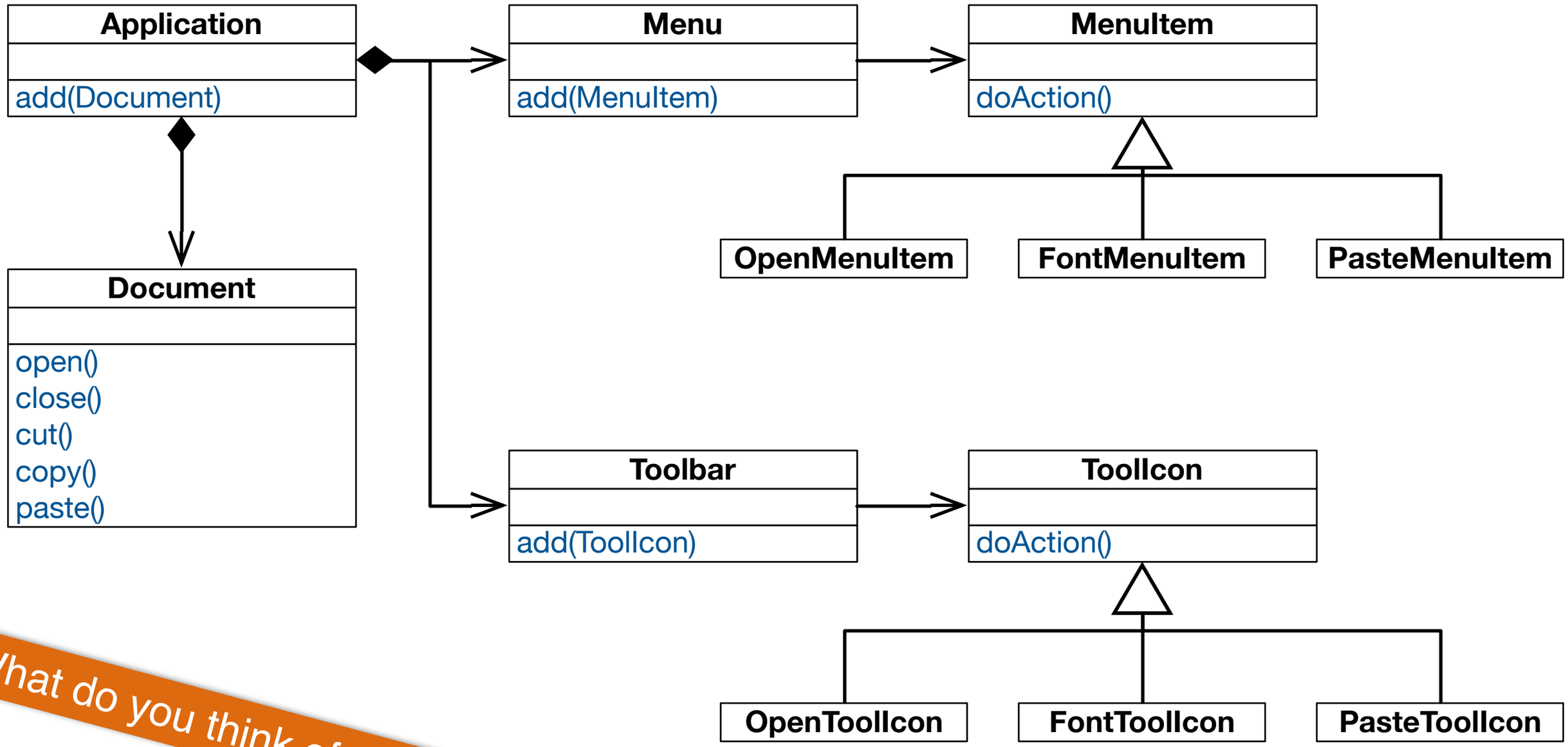
Software Engineering Design & Construction

Dr. Michael Eichberg
Fachgebiet Softwaretechnik
Technische Universität Darmstadt

Command Pattern

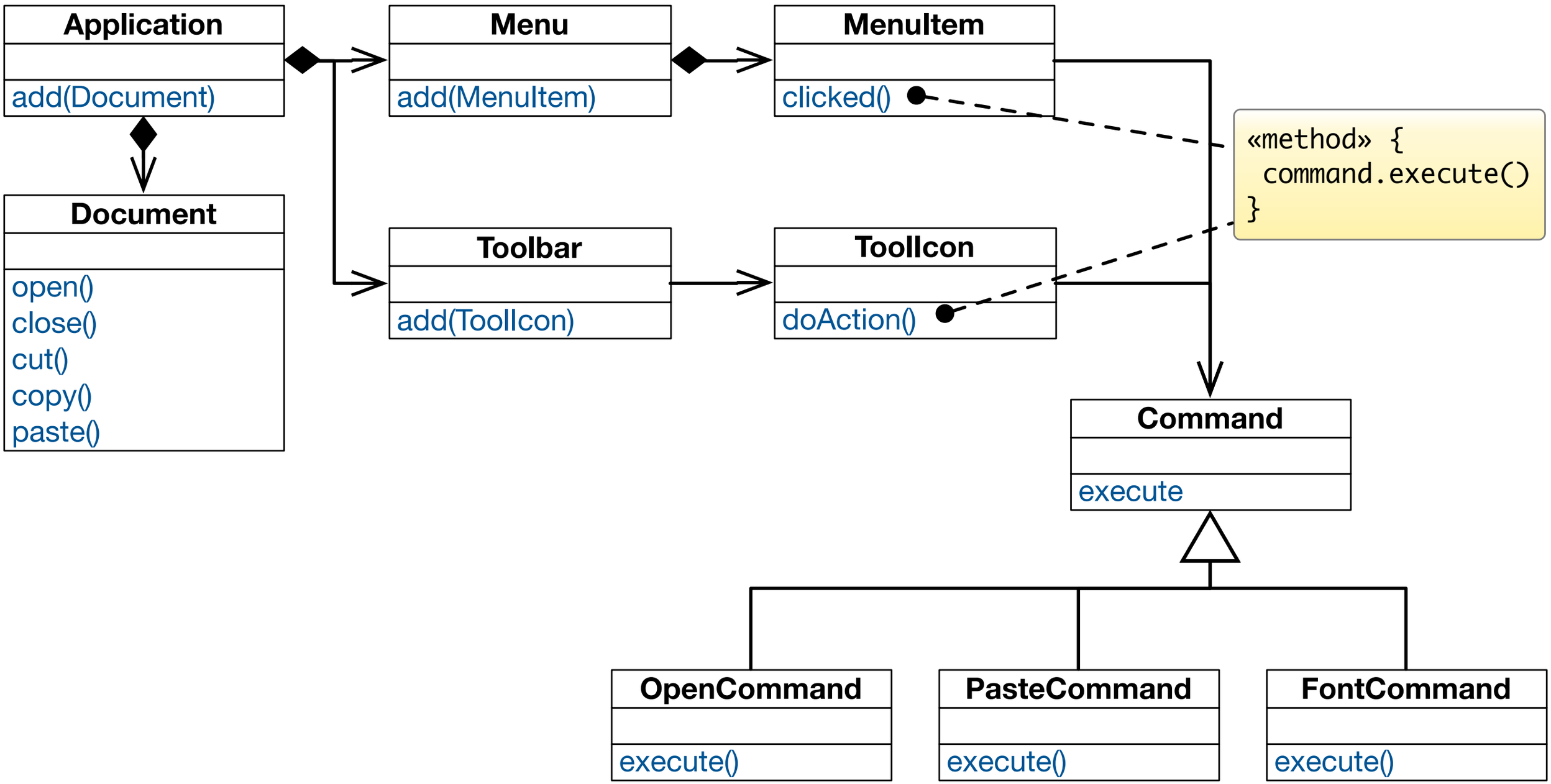
Command Design Pattern

Motivating Example: A Document Editor

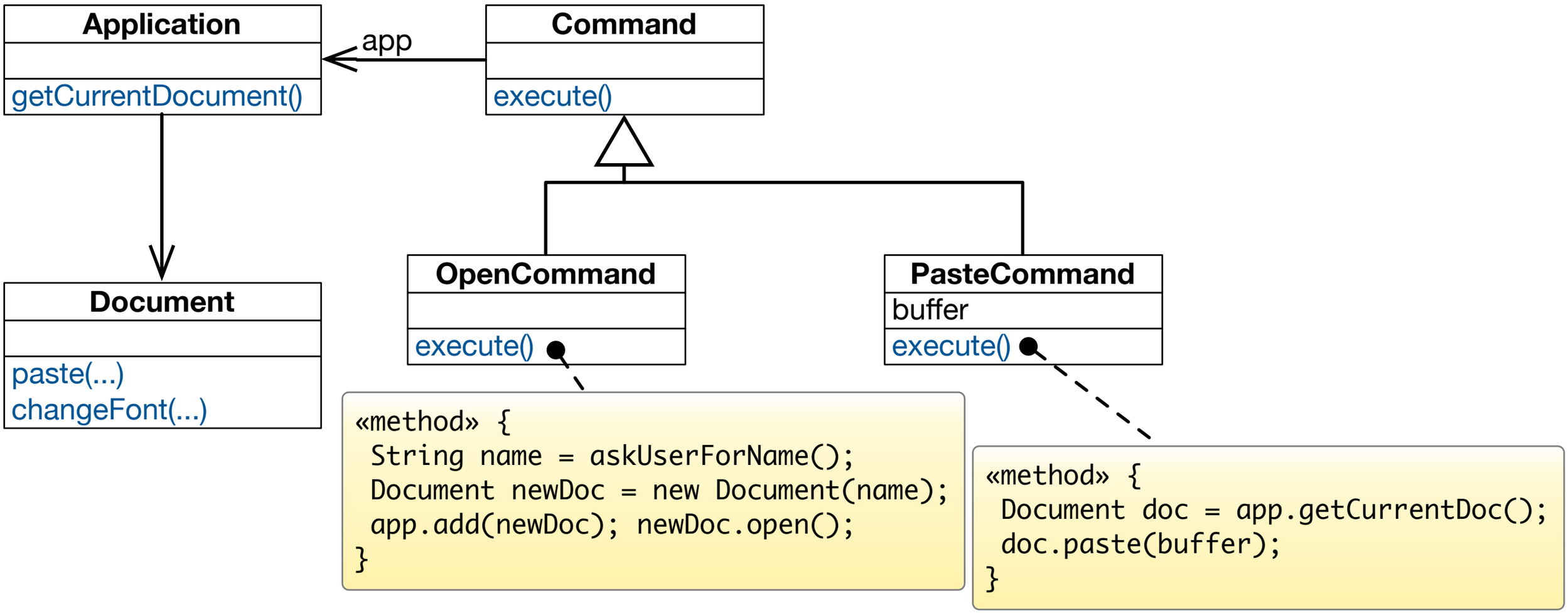


What do you think of this design?

Solution: Decouple Invoker from Receiver



Solution: Decouple Invoker from Receiver

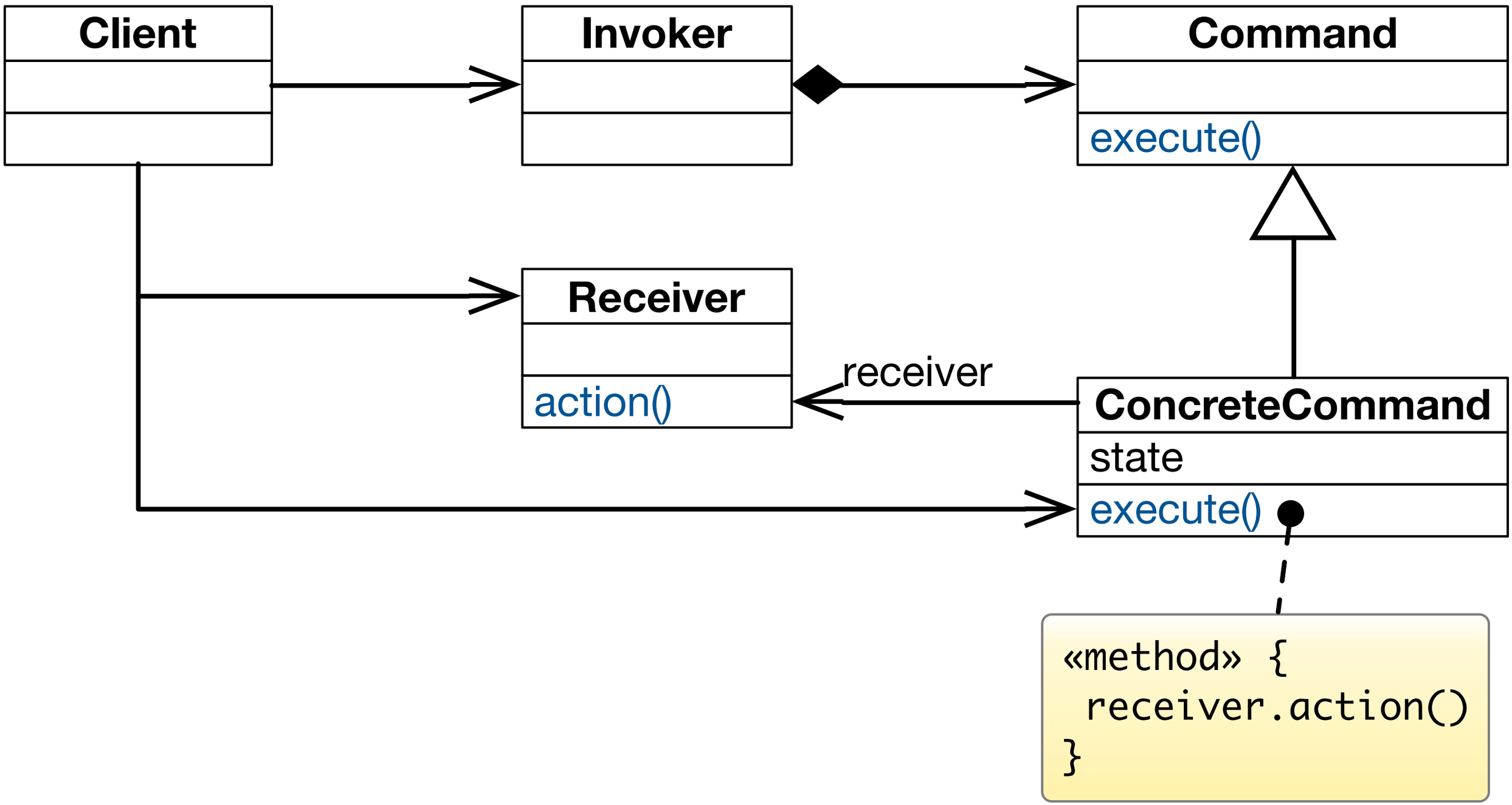


Command Pattern - Intent

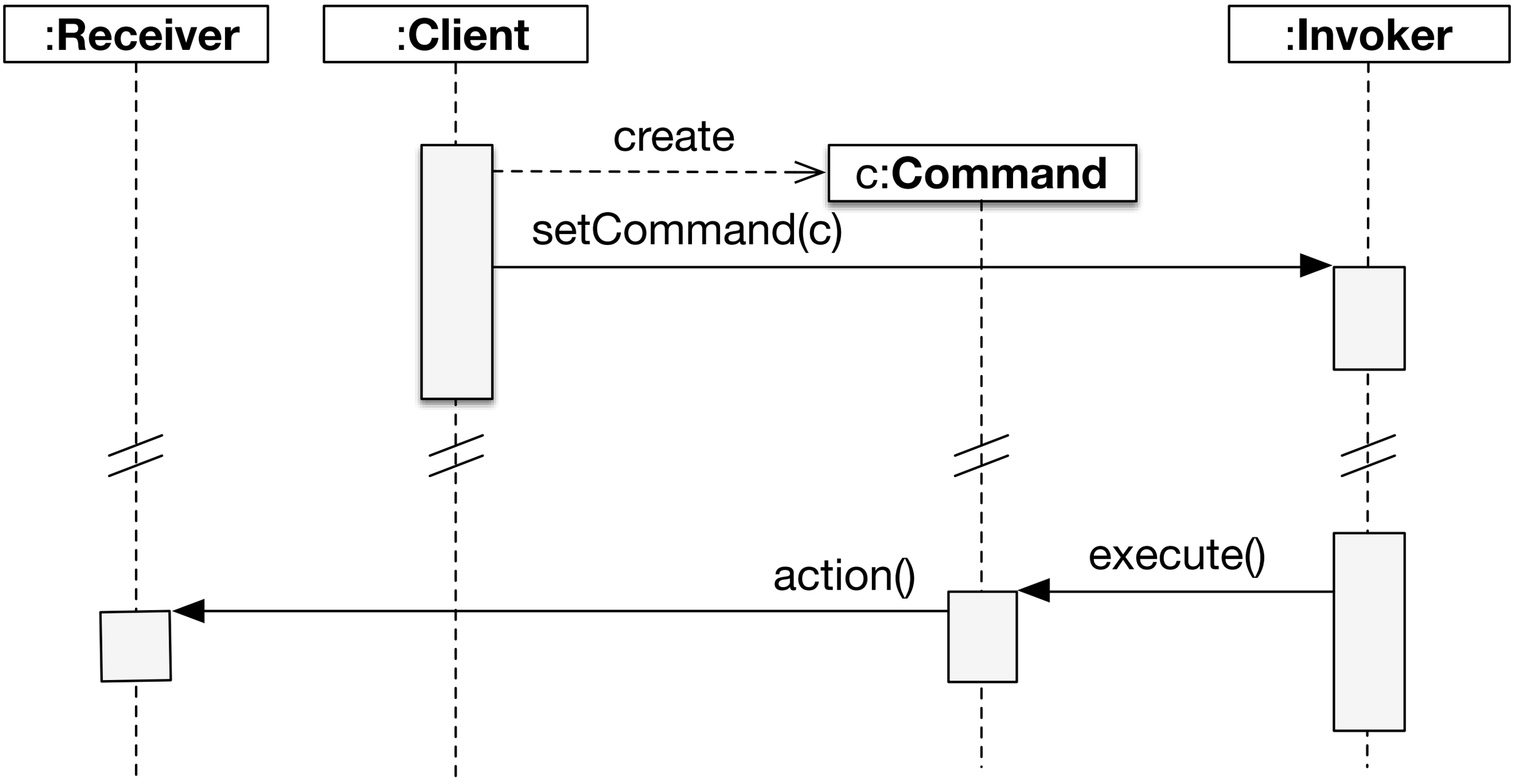
Encapsulate a request to an object, thereby allowing to:

- Issue requests without knowing the receiver or the operation being requested.
- Parameterize clients with different requests.
- Queue or log requests and support undoable requests.

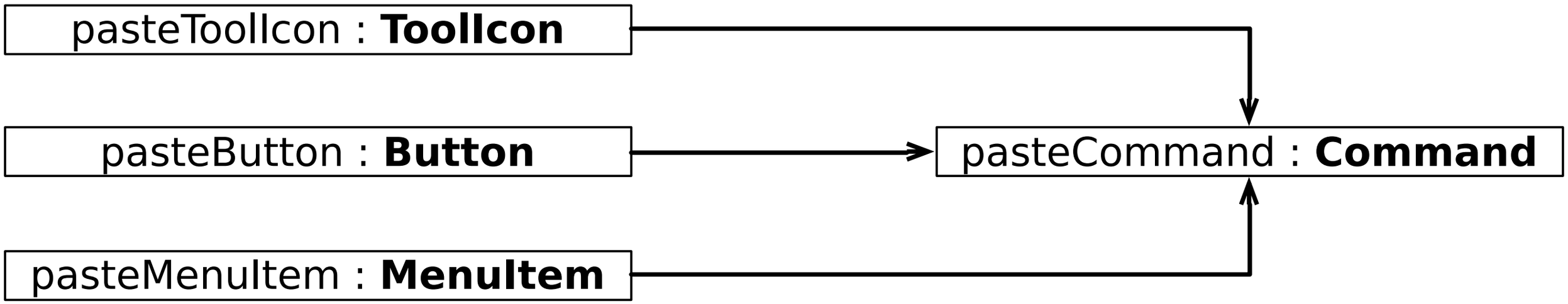
Command Pattern - Structure



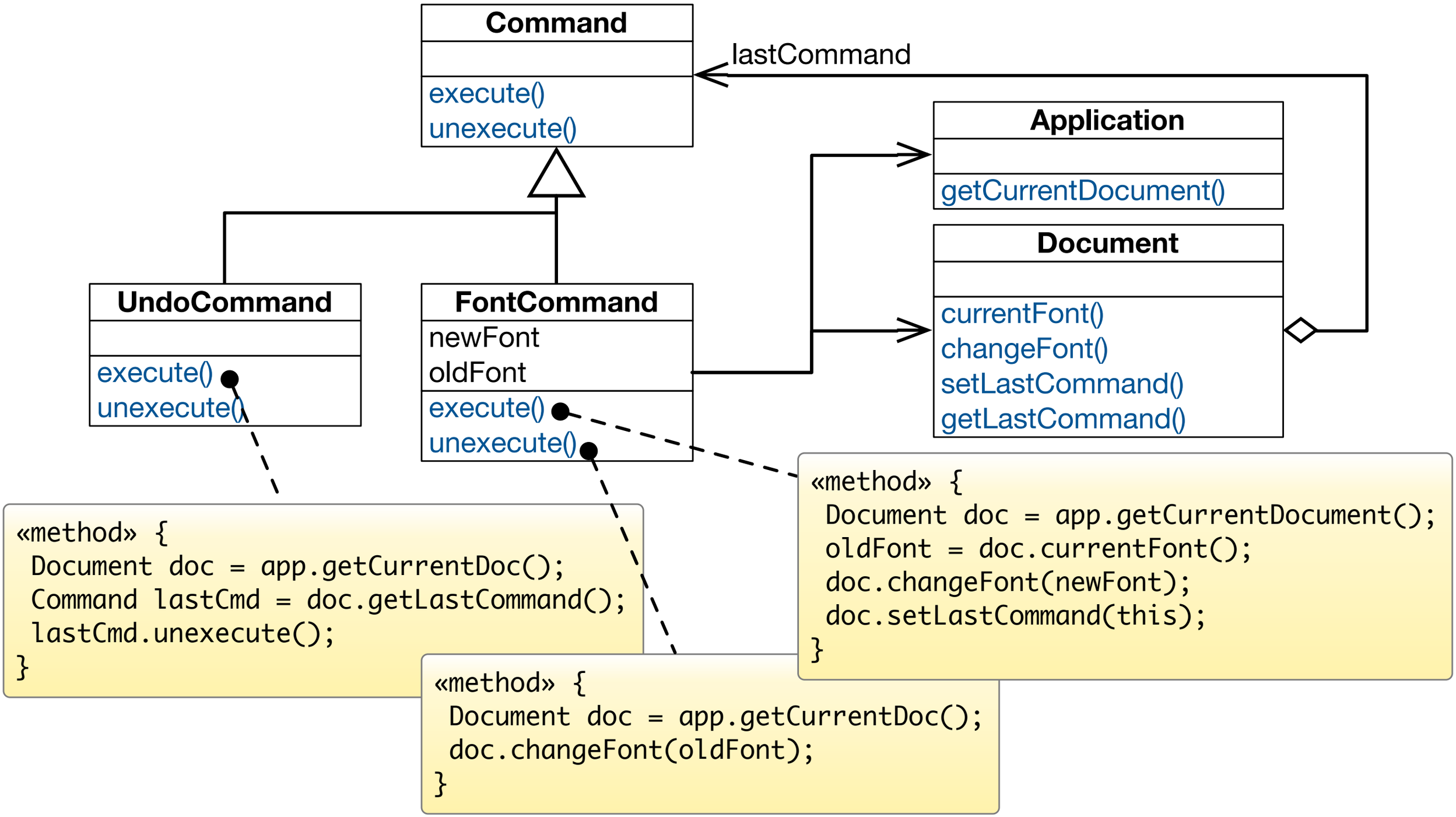
Command Pattern - Collaboration



Implementation Sharing



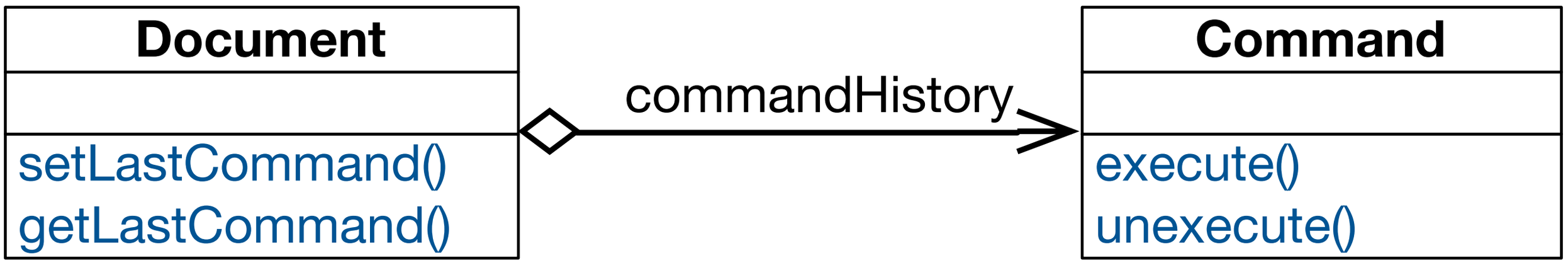
Supporting Undoable Operations



Supporting Multiple Levels Of Undo

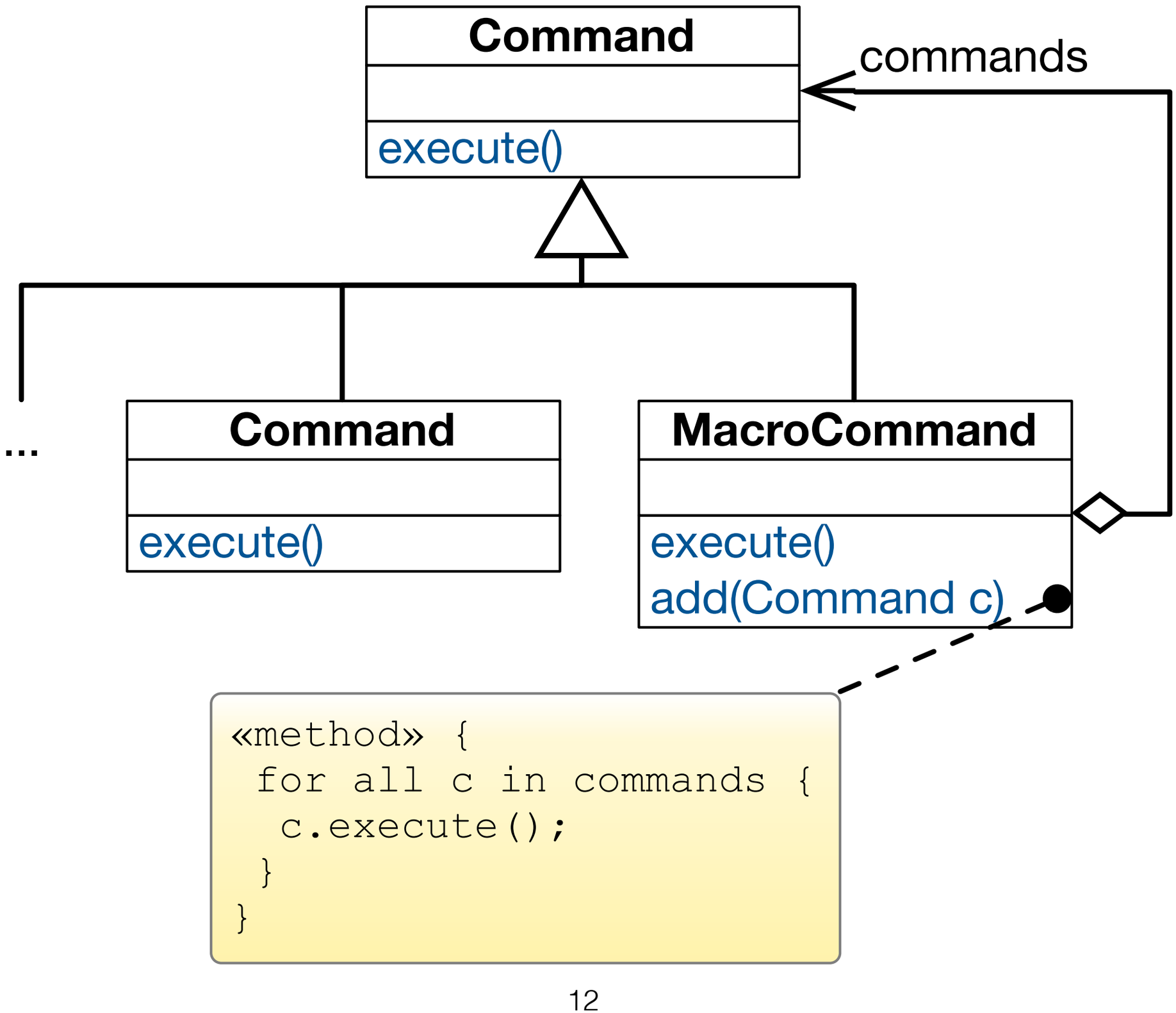
Single Level of Undo

Multiple Levels of Undo



Implementing a Command History

Macro Commands



Takeaway

- Command allows to decouple the invoker of an operation from the receiver of that operation.

- A `Command` object encapsulates the knowledge about a concrete operation and a concrete receiver of that operation.

- As a result:

  - the same invoker can be reused with different operation-receiver pairs.

  - the same operation-receiver pair can be plugged into different invokers.

  - commands can be queued, undone/redone, and composed into macro-commands.