Design Patterns & Software Architecture

Command

dr. Joost Schalken-Pinkster

Windesheim University of Applied Science

The Netherlands

The contents of these course slides is (in great part) based on:
Chris Loftus, *Course on Design Patterns & Software Architecture for NEU*. Aberystwyth University, 2013.
Jeroen Weber & Christian Köppe, *Course on Patterns and Frameworks*. Hogeschool Utrecht, 2013.
Leo Pruijt, *Course on Software Architecture*. Hogeschool Utrecht, 2010-2013.

Session overview

- Command

Command design pattern

Let’s find a design pattern

Will now present, on the board,
and using Eclipse,
a solution that utilises
the command design pattern…

Case: menu system
Requirements

You have been asked to develop a menu system for a word processor.

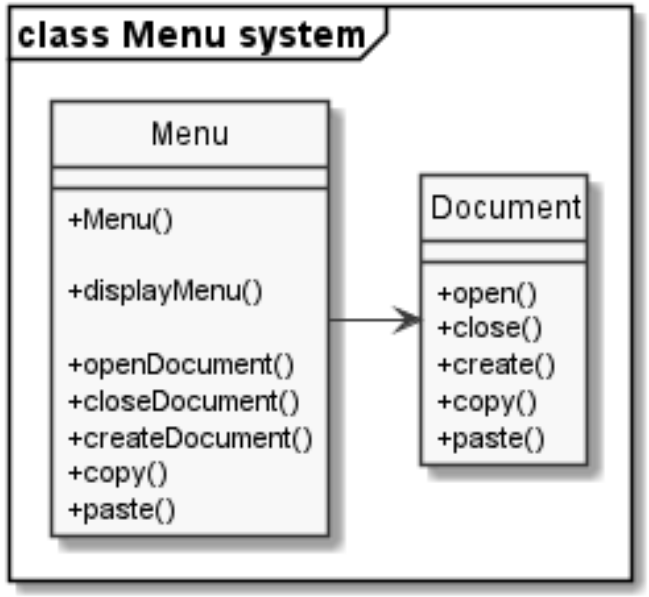
The initial requirements was:

1. The user should be able to open, close and create documents and to copy and paste text.

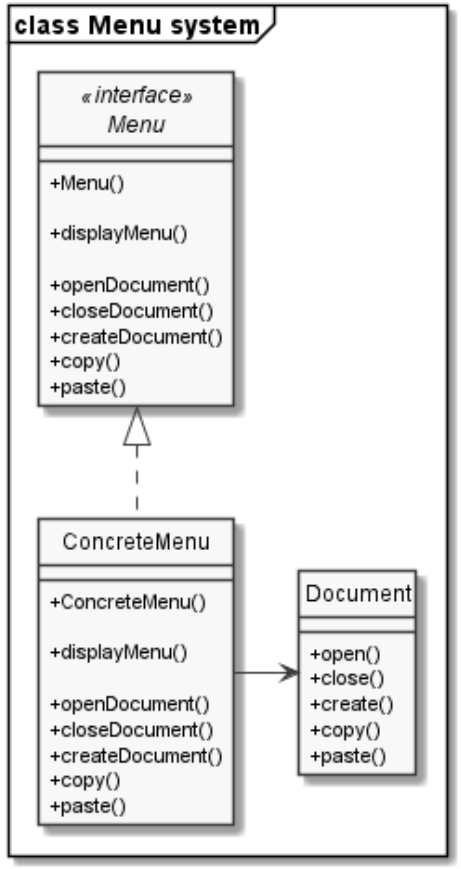
Then slightly later the architect asks you

2. The user should be able to create and execute macro commands and attach them to the menu.

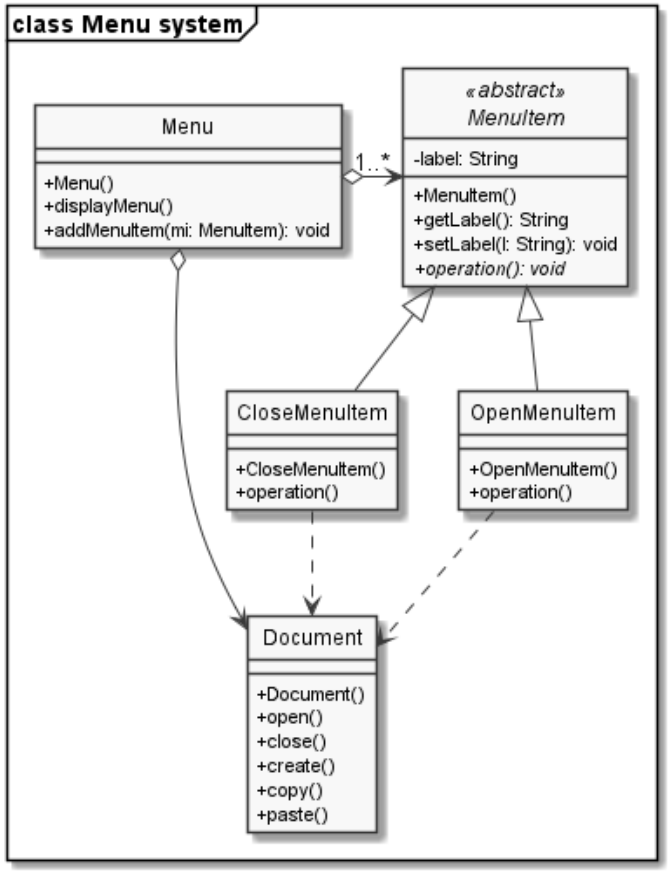
Case: menu system
Design 1: Not open closed for menu additions



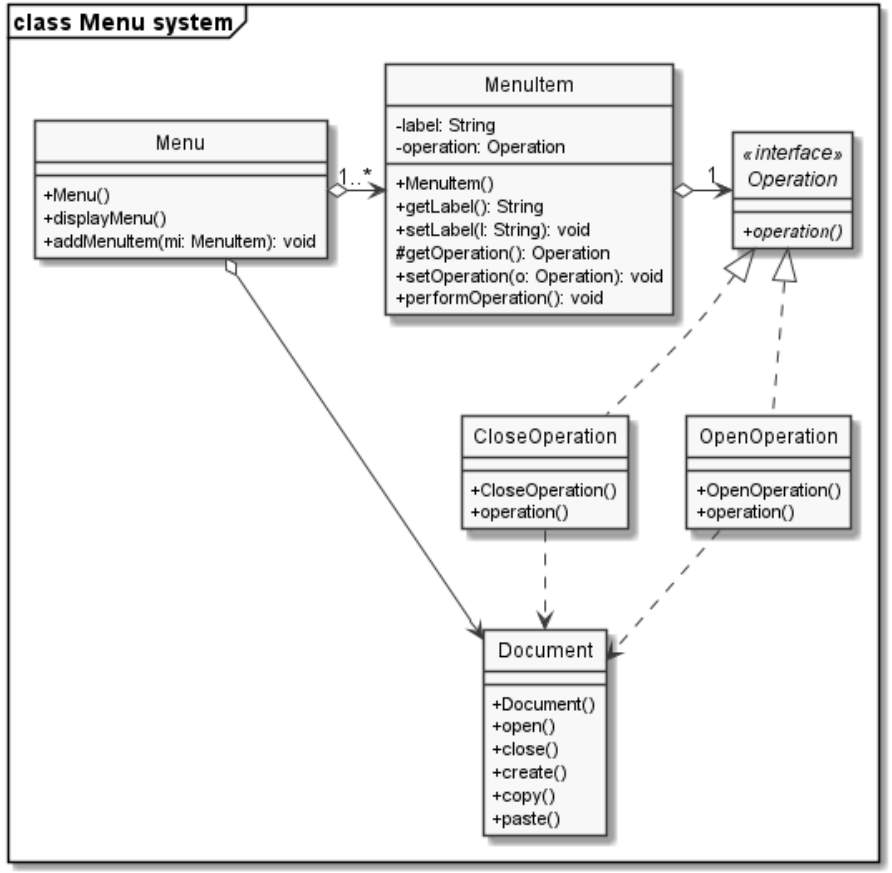
Case: menu system
Design 2: Not better



Case: menu system
Design 3: Is open closed, but no reuse of Open and Close functionality



Case: menu system
Design 4: better design

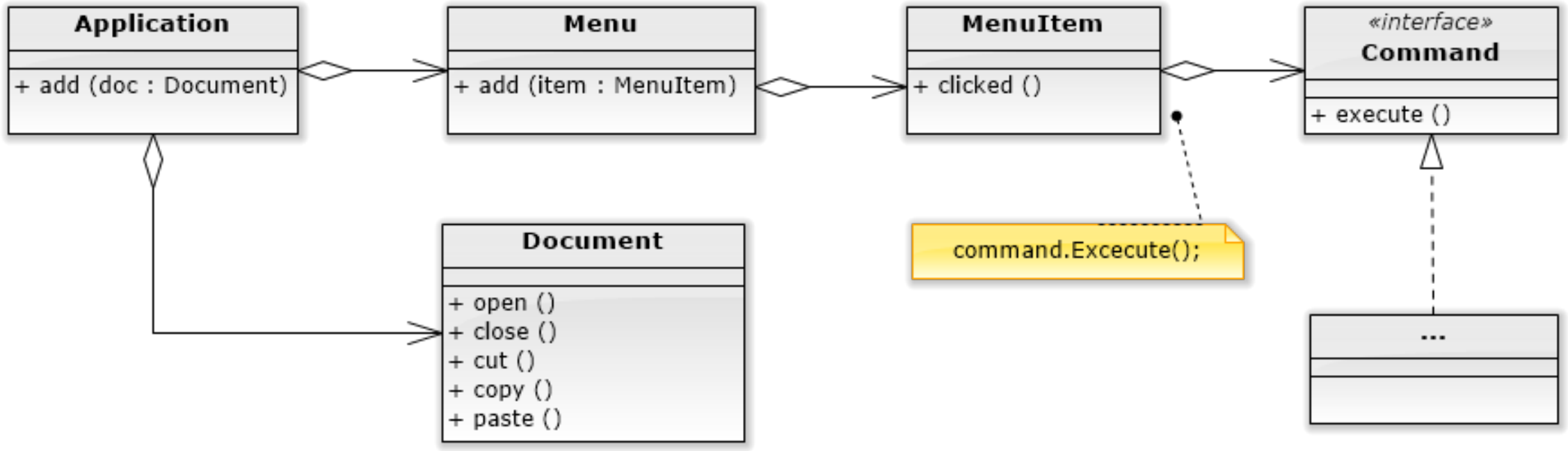


Command pattern definition

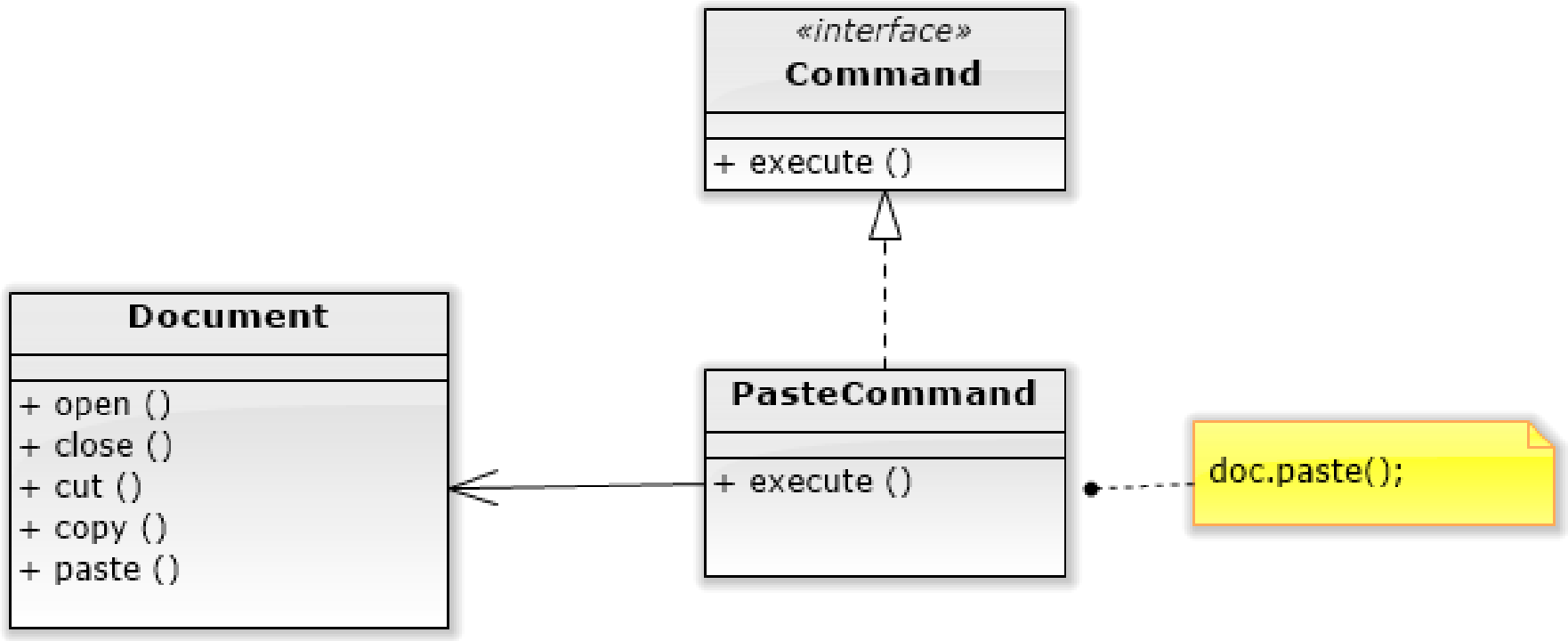
- **Intent**: Encapsulate a request as an object, thereby letting you parameterize clients with different requests, queue or log requests, and support undoable operations.

- **Motivation**: Sometimes it's necessary to issue requests to objects without knowing anything about the operation being requested or the receiver of the request. For example, user interface toolkits include objects like buttons and menus that carry out a request in response to user input.
  But the toolkit can't implement the request explicitly in the button or menu, because only applications that use the toolkit know what should be done on which object.

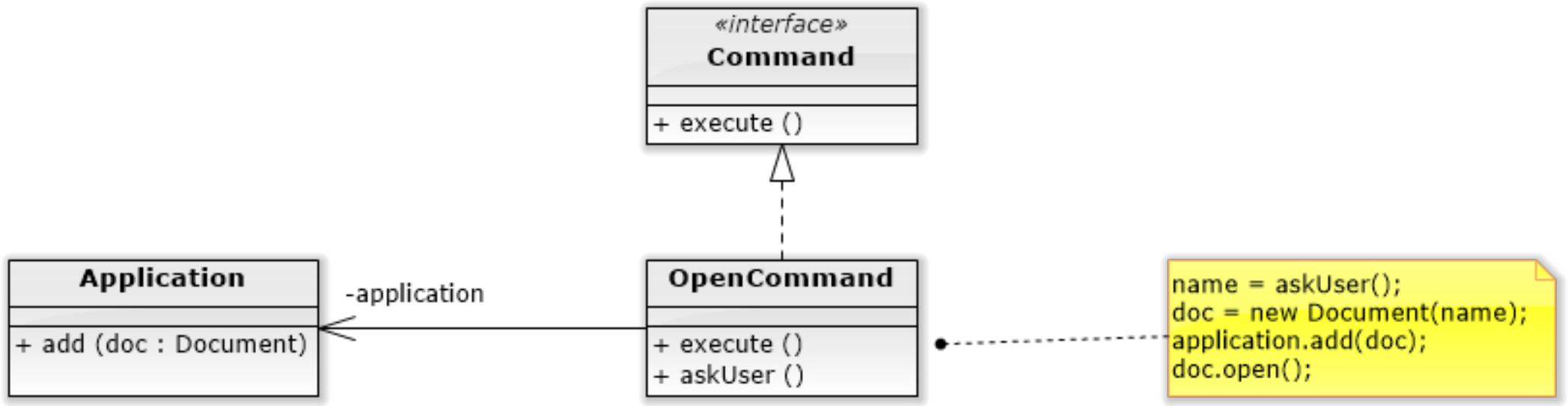
Motivation



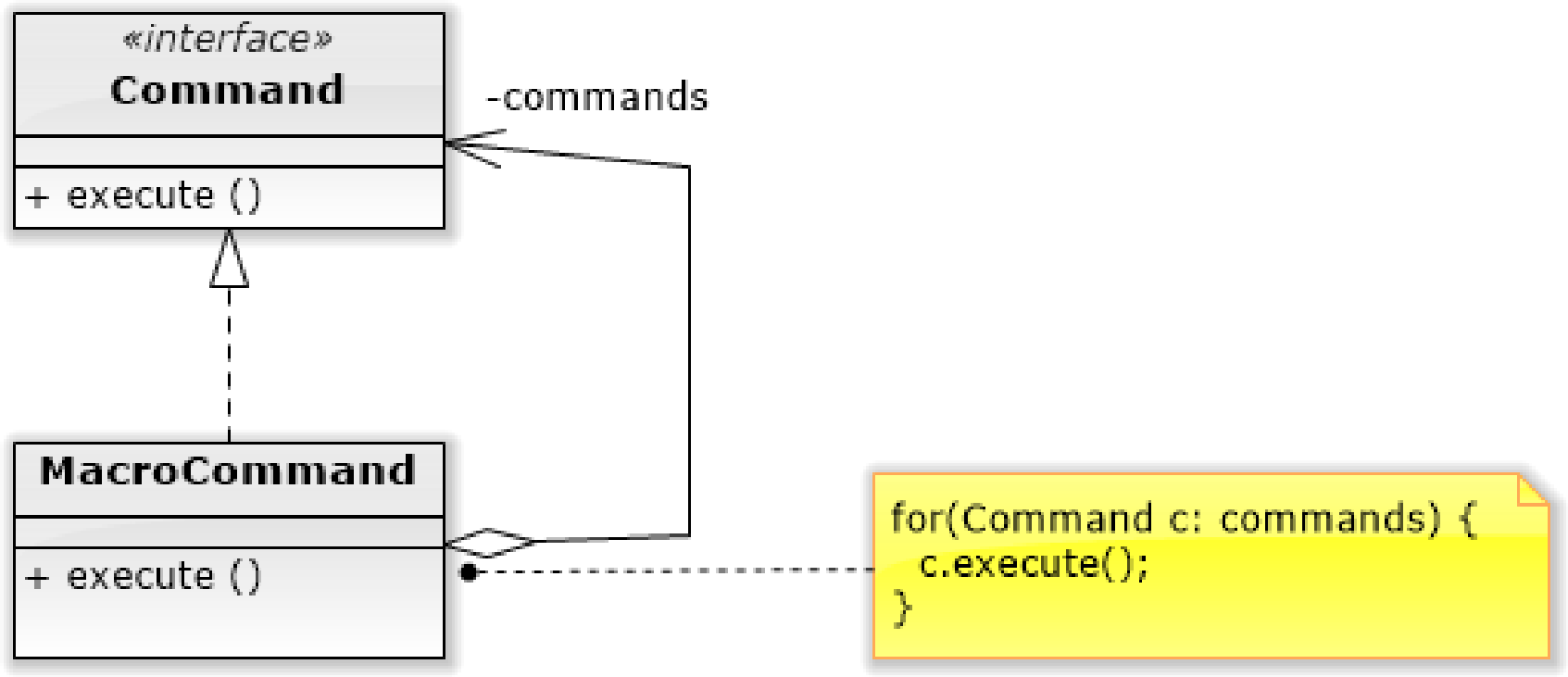
Gamma (1995), Design Patterns: Elements of Reusable Object-Oriented Software, p. 263



Gamma (1995), Design Patterns: Elements of Reusable Object-Oriented Software, p. 264



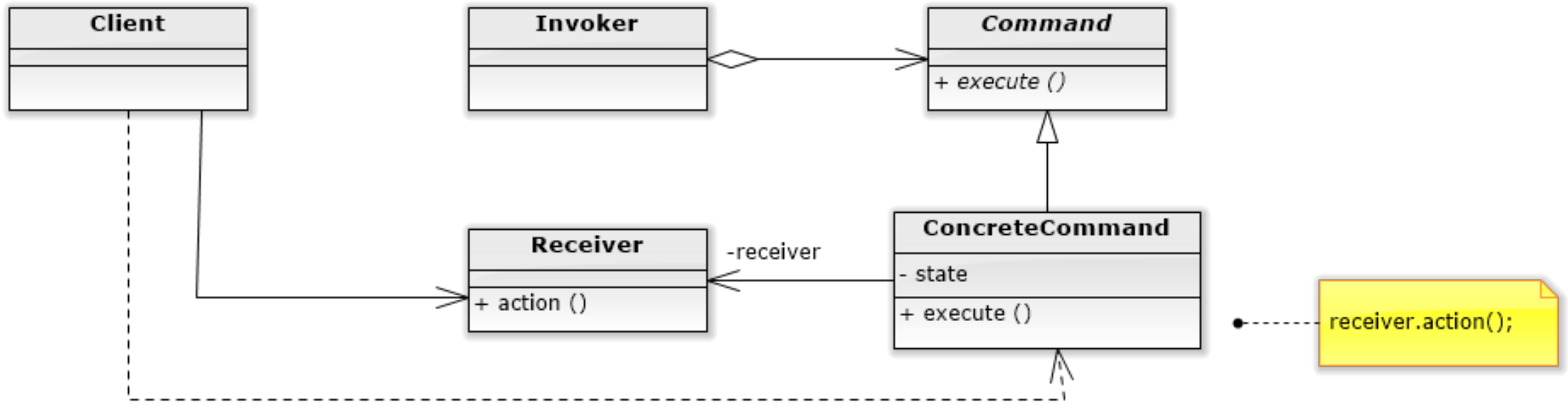
Gamma (1995), Design Patterns: Elements of Reusable Object-Oriented Software, p. 264



Gamma (1995), Design Patterns: Elements of Reusable Object-Oriented Software, p. 265

- **Applicability**: Use the Command pattern when:
  - an action to be performed needs to be parameterized (by an objects)
  - requests are specified, queued, and executed at different times.
  - actions need to be undoable
  - the application needs to support logging changes so that they can be reapplied in case of a system crash
  - a system needs to be structured around high-level operations built on primitives operations

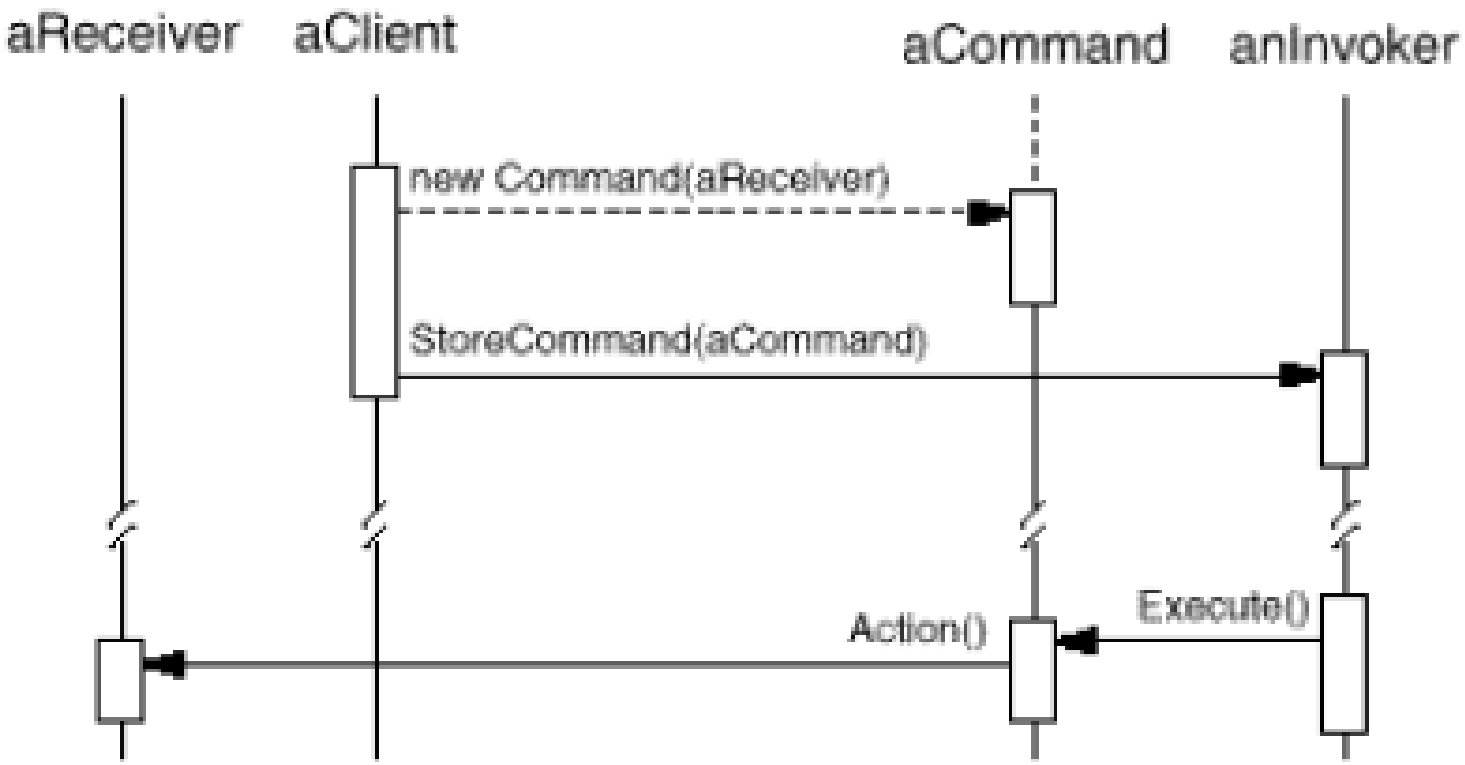
- **Structure:**



Gamma (1995), Design Patterns: Elements of Reusable Object-Oriented Software, p. 266

- **Participants:**
  - Command
    - declares an interface for executing an operation.
  - ConcreteCommand (PasteCommand, OpenCommand)
    - defines a binding between a Receiver object and an action.
    - implements Execute by invoking the corresponding operation(s) on Receiver.
  - Client (Application)
    - creates a ConcreteCommand object and sets its receiver.
  - Invoker (MenuItem)
    - asks the command to carry out the request.
  - Receiver (Document, Application)
    - knows how to perform the operations associated with carrying out a request. Any class may serve as a Receiver.

Gamma (1995), Design Patterns: Elements of Reusable Object-Oriented Software, p. 266



Gamma (1995), Design Patterns: Elements of Reusable Object-Oriented Software, p. 266

- **Consequences**: The command pattern leads to:
  - Command decouples the object that invokes the operation from the one that knows how to perform it.
  - Commands are first-class objects. They can be manipulated and extended like any other object.
  - You can assemble commands into a composite command.
  - It's easy to add new Commands, because you don't have to change existing classes.

Gamma (1995), Design Patterns: Elements of Reusable Object-Oriented Software, p. 267

- **Implementation**: Some issues
  - How intelligent should a command be?
  - Supporting undo and redo
    - Avoiding error accumulation in the undo process

Reading

For this lesson please read:

- Chapter 6 (Encapsulating Invocation) of Head First Design Patterns.