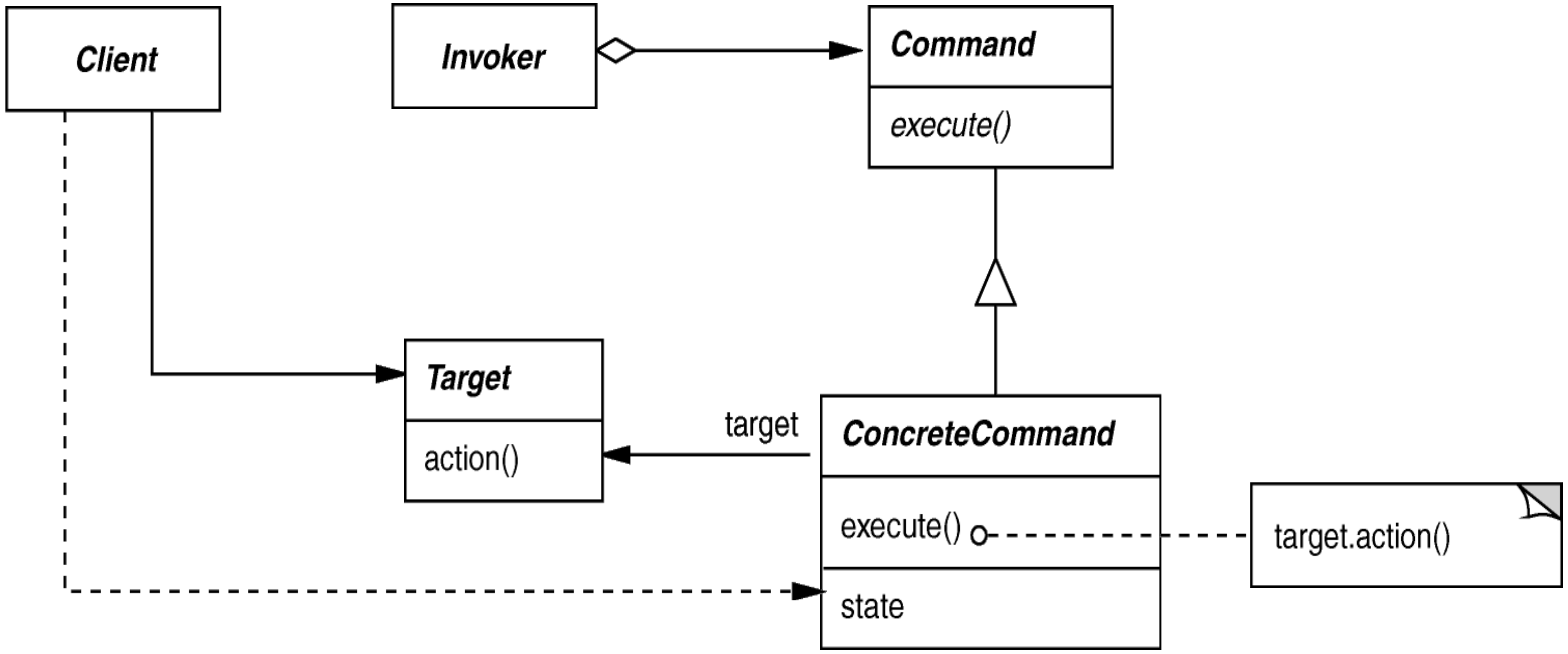
The Command Pattern

Structure & Functionality

Douglas C. Schmidt

Learning Objectives in This Lesson

- Recognize how the *Command* pattern can be applied to perform user-requested commands consistently & extensibly in the expression tree processing app.
- Understand the structure & functionality of the *Command* pattern.



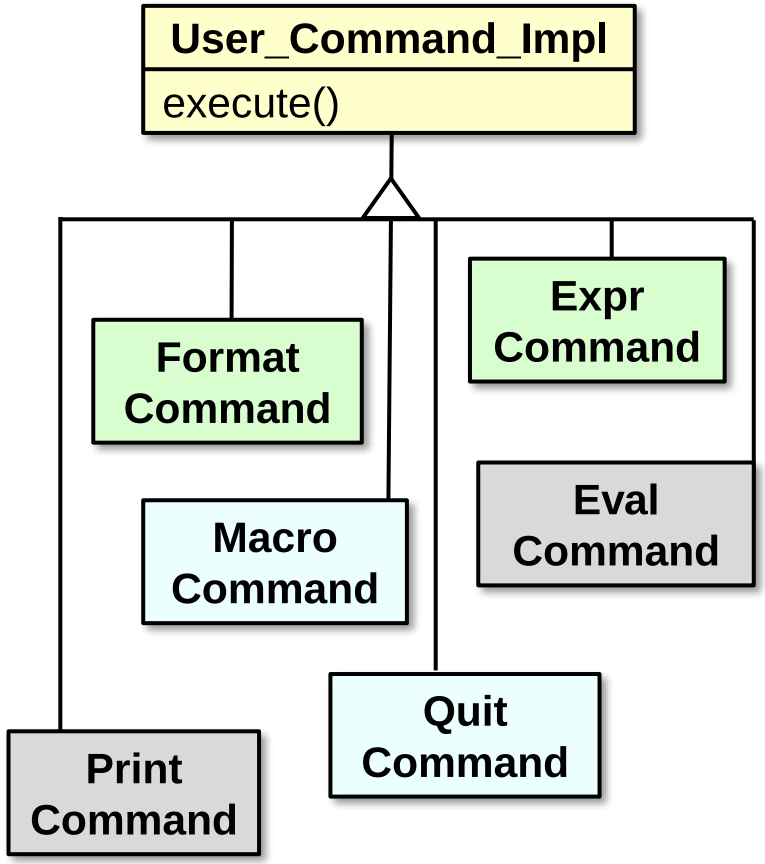
Douglas C. Schmidt

Structure & Functionality of the Command Pattern

Command GoF Object Behavioral

Intent

- Encapsulate the request for a service as an object

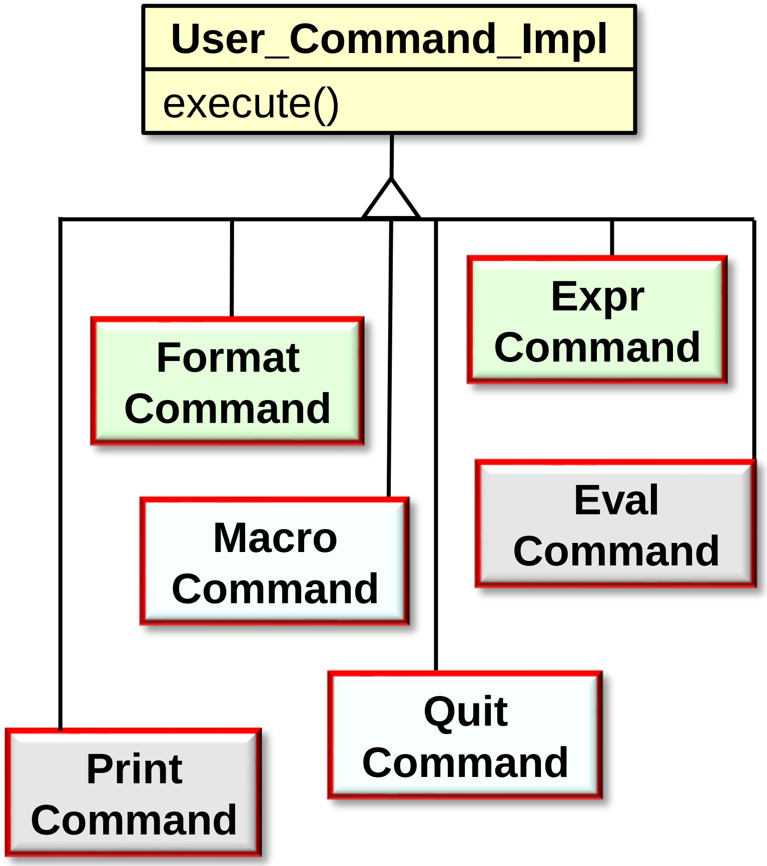


See en.wikipedia.org/wiki/Command_pattern

Command GoF Object Behavioral

Applicability

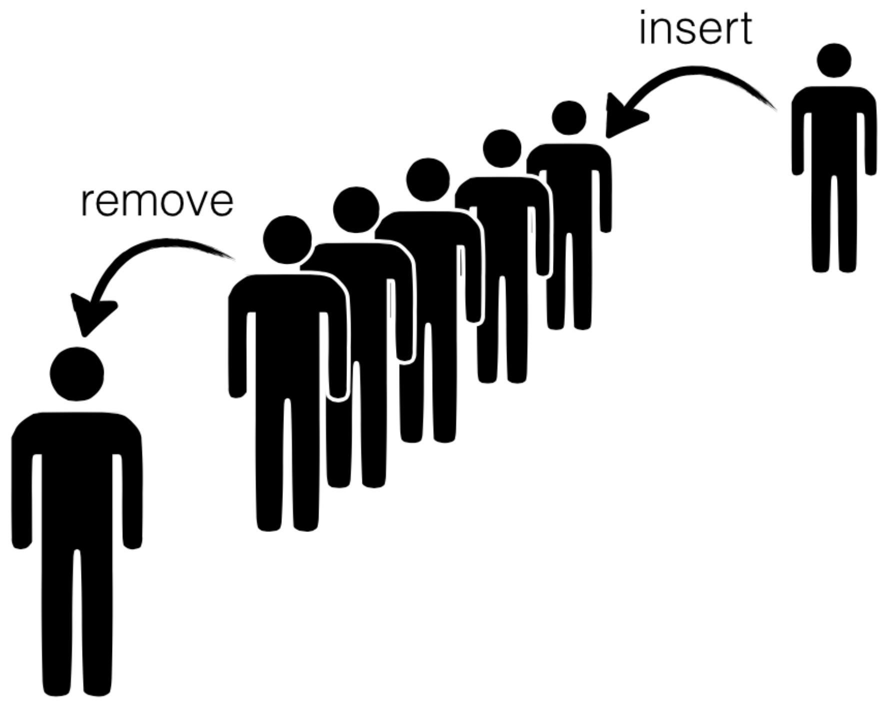
- Want to parameterize objects with an action to perform

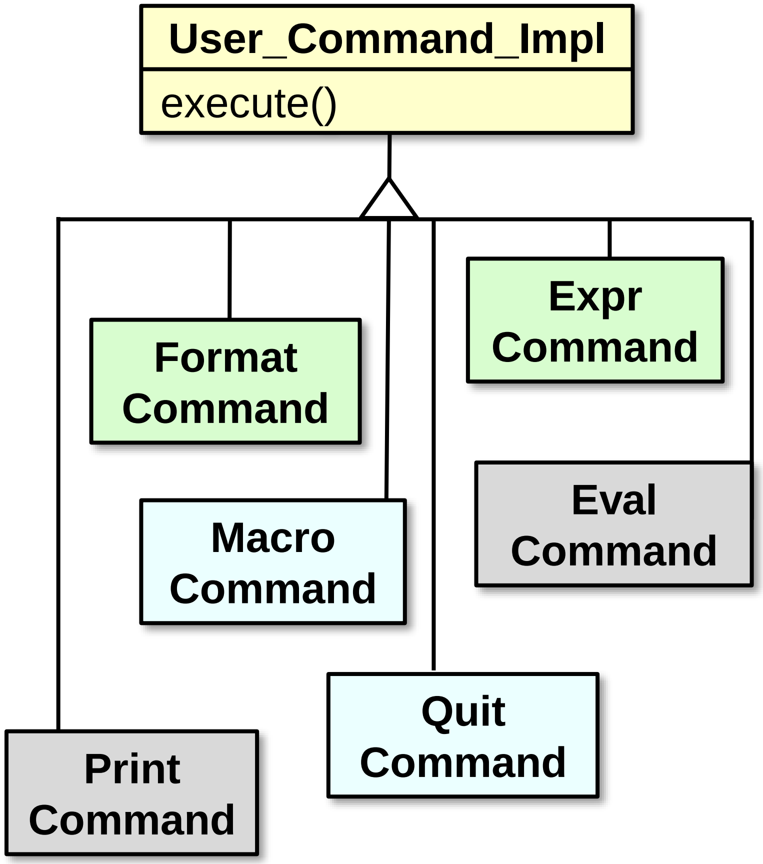


Command GoF Object Behavioral

Applicability

- Want to parameterize objects with an action to perform
- Want to specify, queue, & execute requests at different times

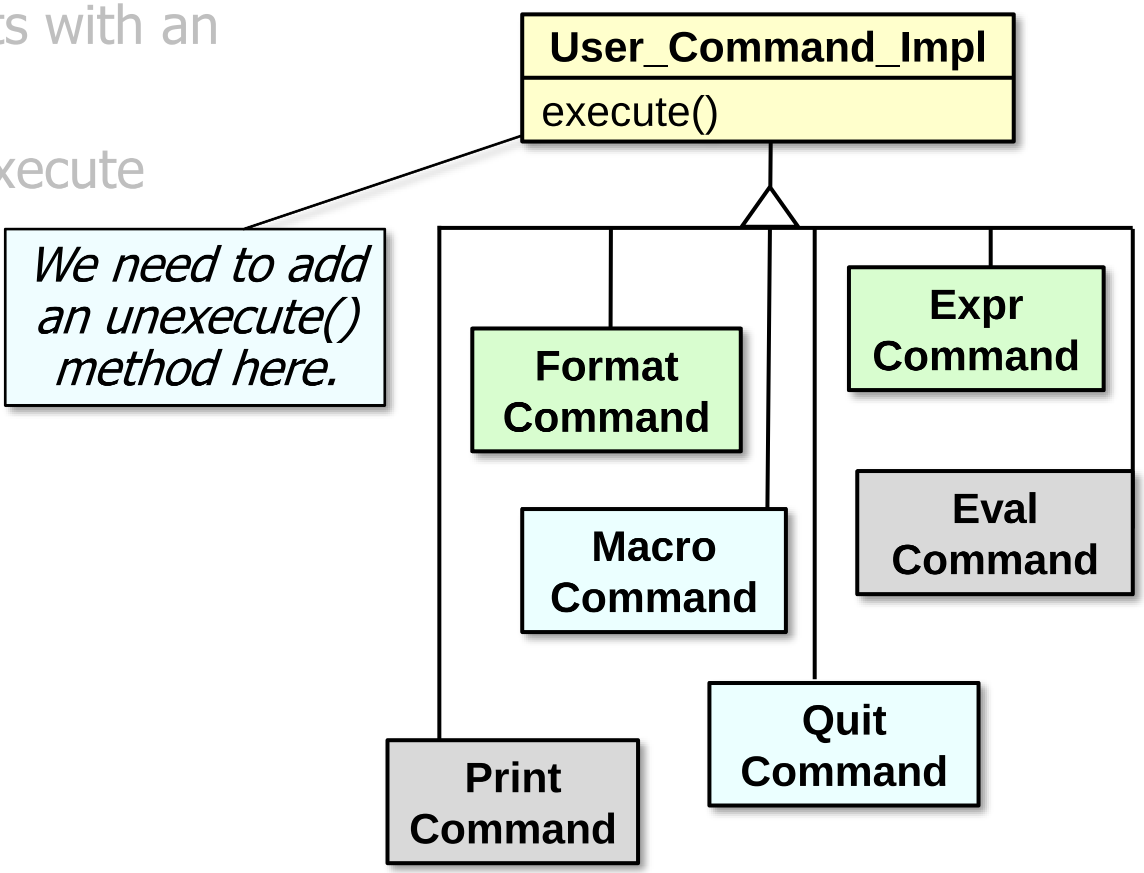




Command GoF Object Behavioral

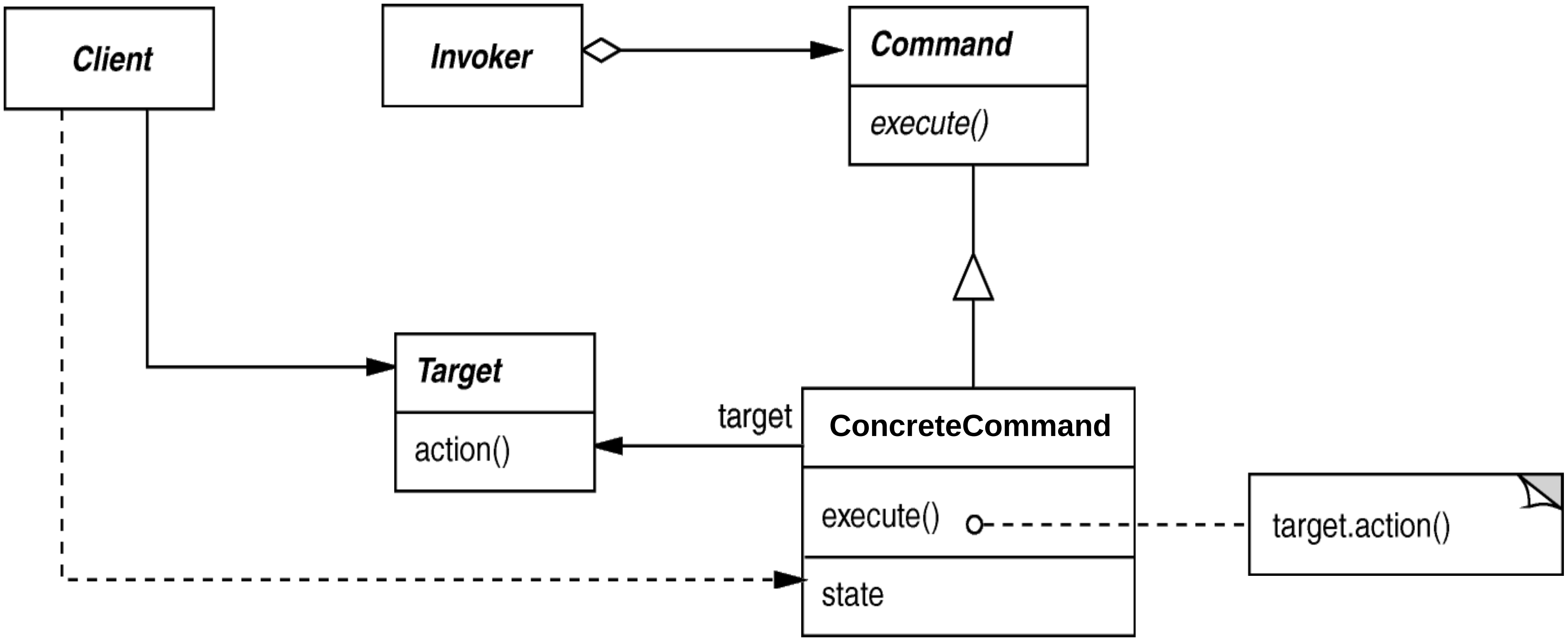
Applicability

- Want to parameterize objects with an action to perform
- Want to specify, queue, & execute requests at different times
- Want to support multilevel undo/redo



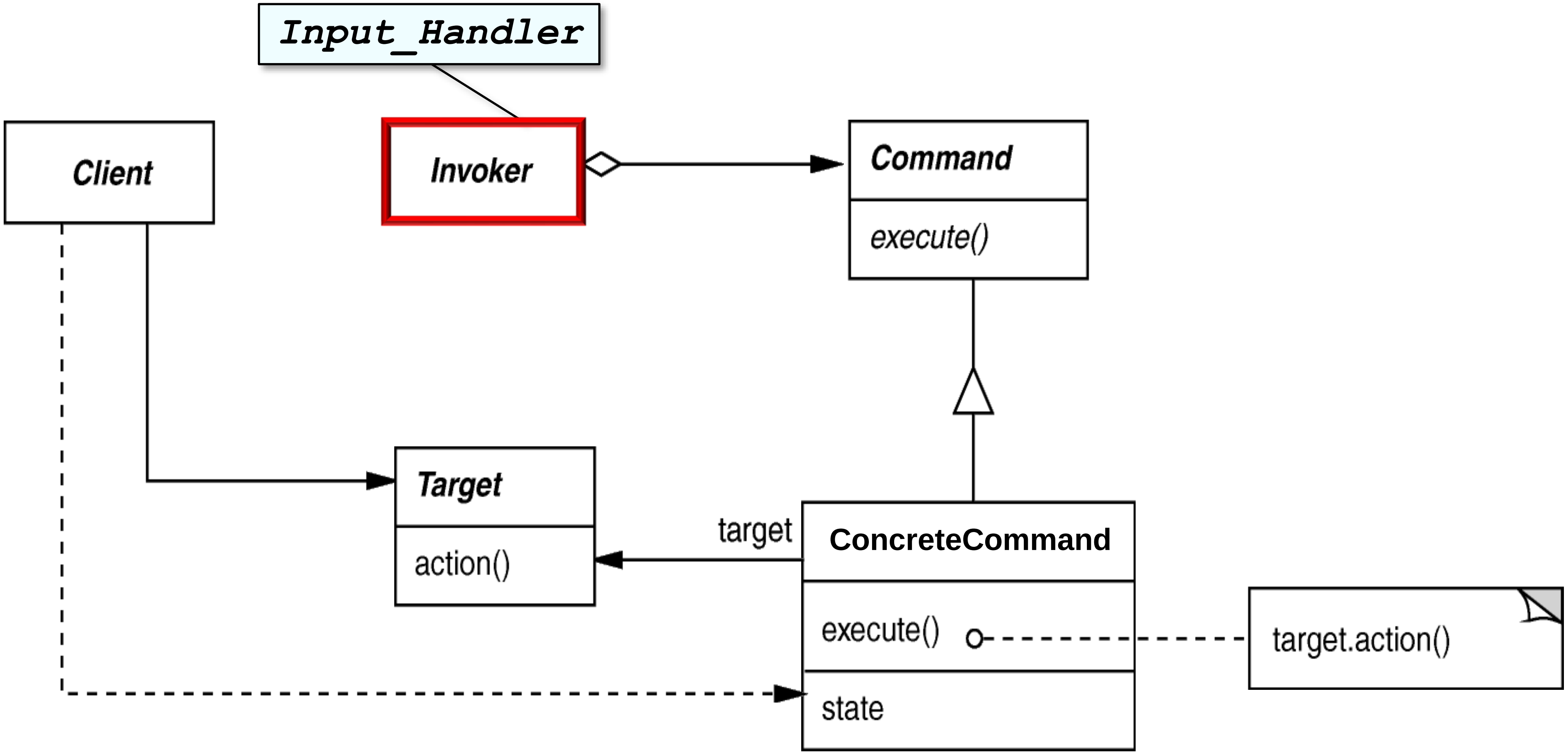
Command GoF Object Behavioral

Structure & participants



Command GoF Object Behavioral

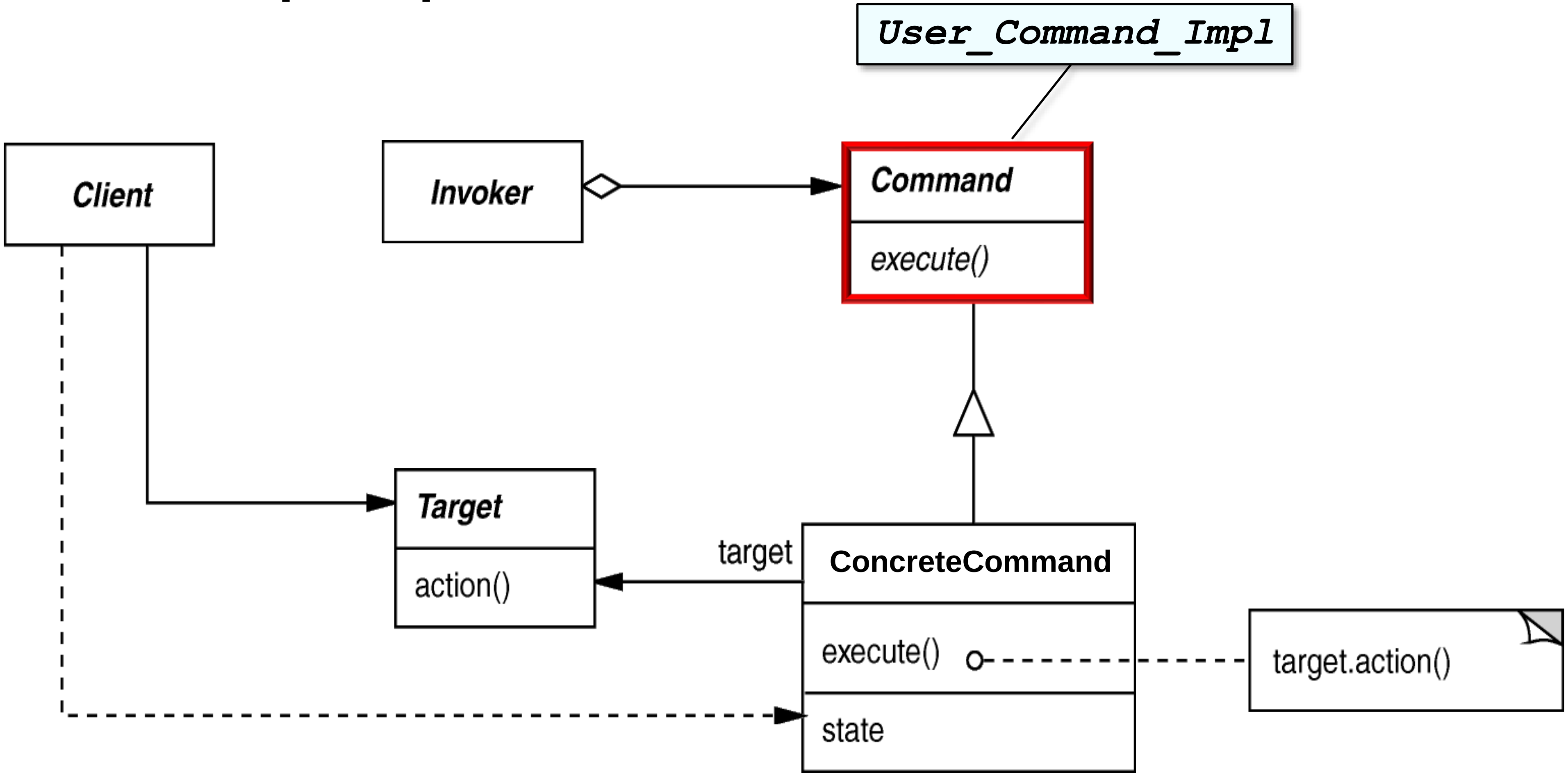
Structure & participants



See upcoming lesson on the *Template Method* pattern

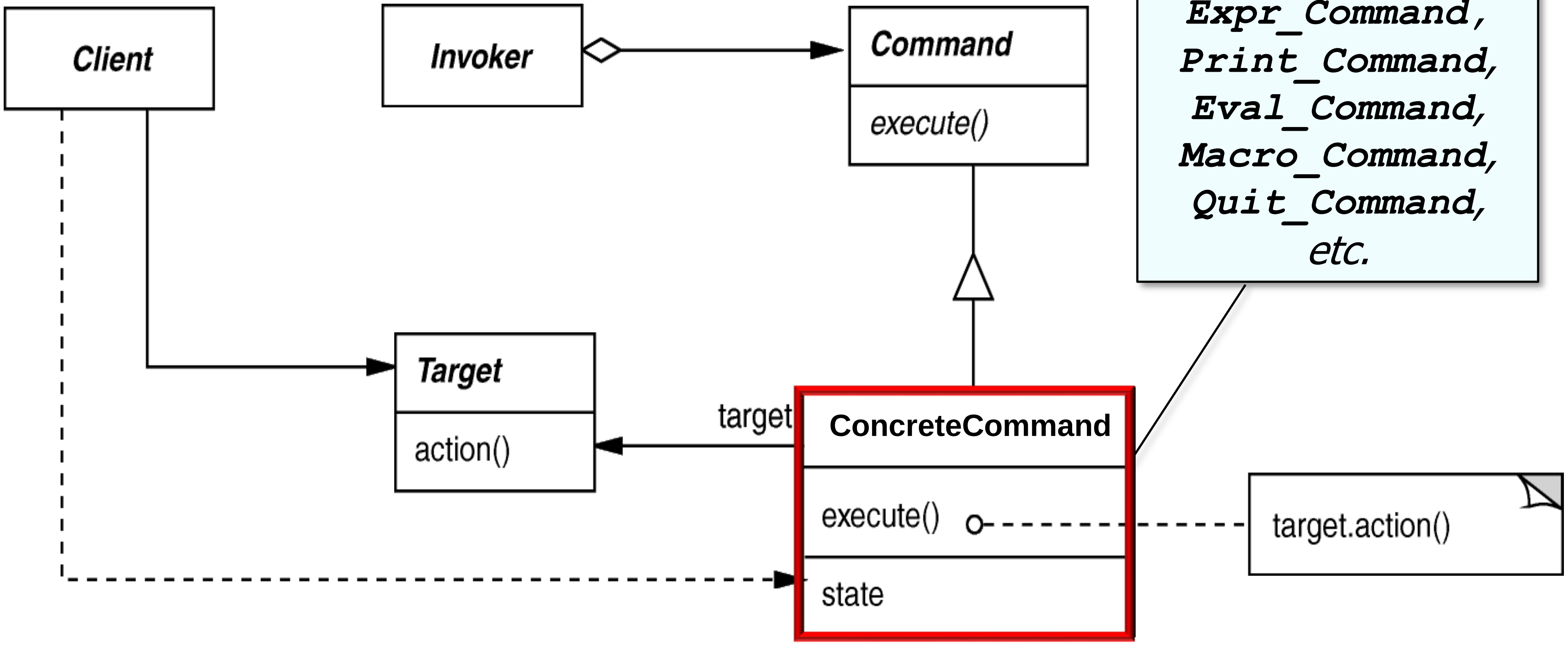
Command GoF Object Behavioral

Structure & participants



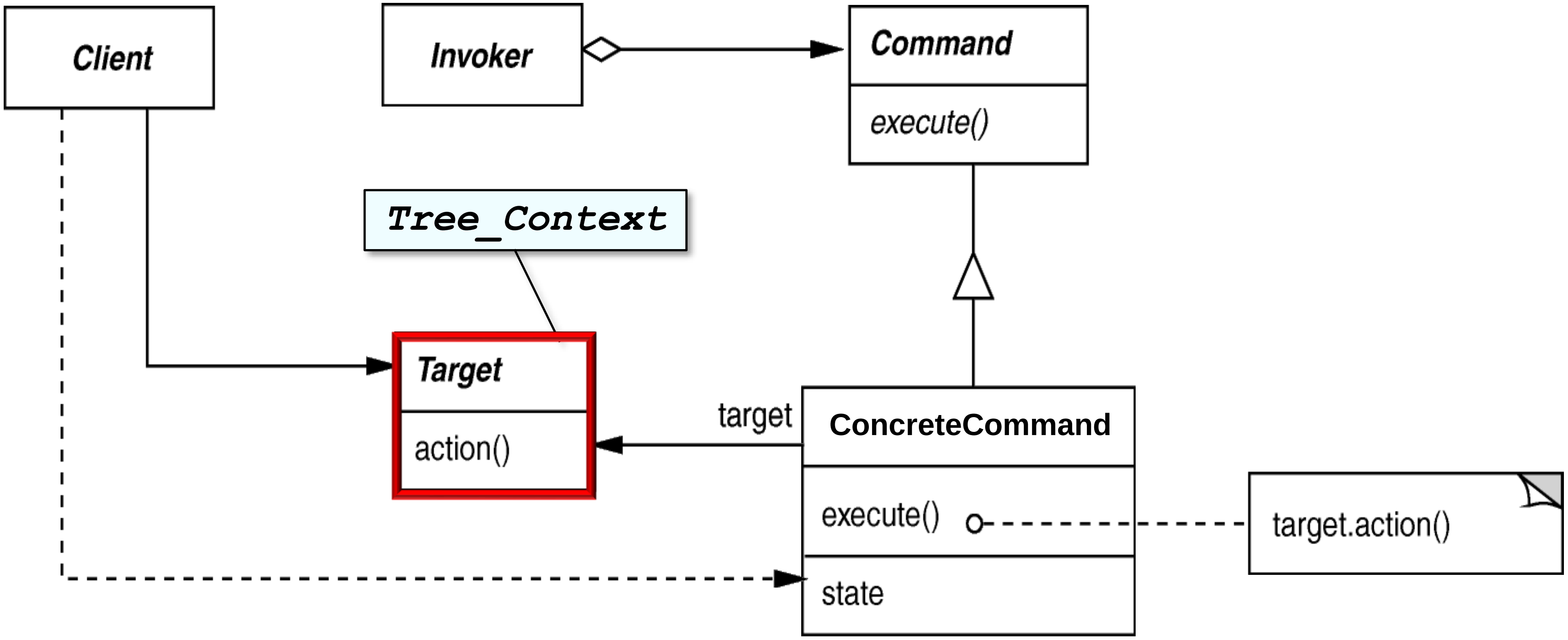
Command GoF Object Behavioral

Structure & participants



Format_Command, Expr_Command, Print_Command, Eval_Command, Macro_Command, Quit_Command, *etc.*

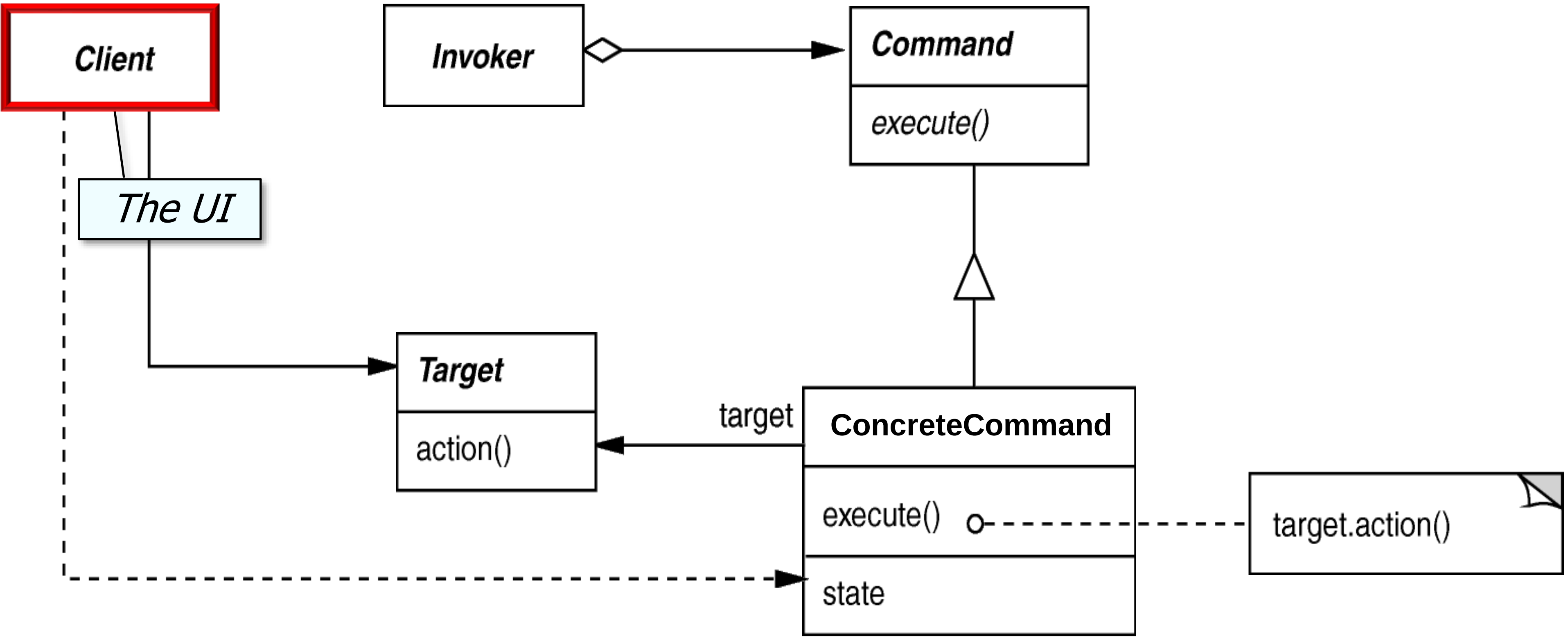
Structure & participants



See upcoming lesson on the *State* pattern

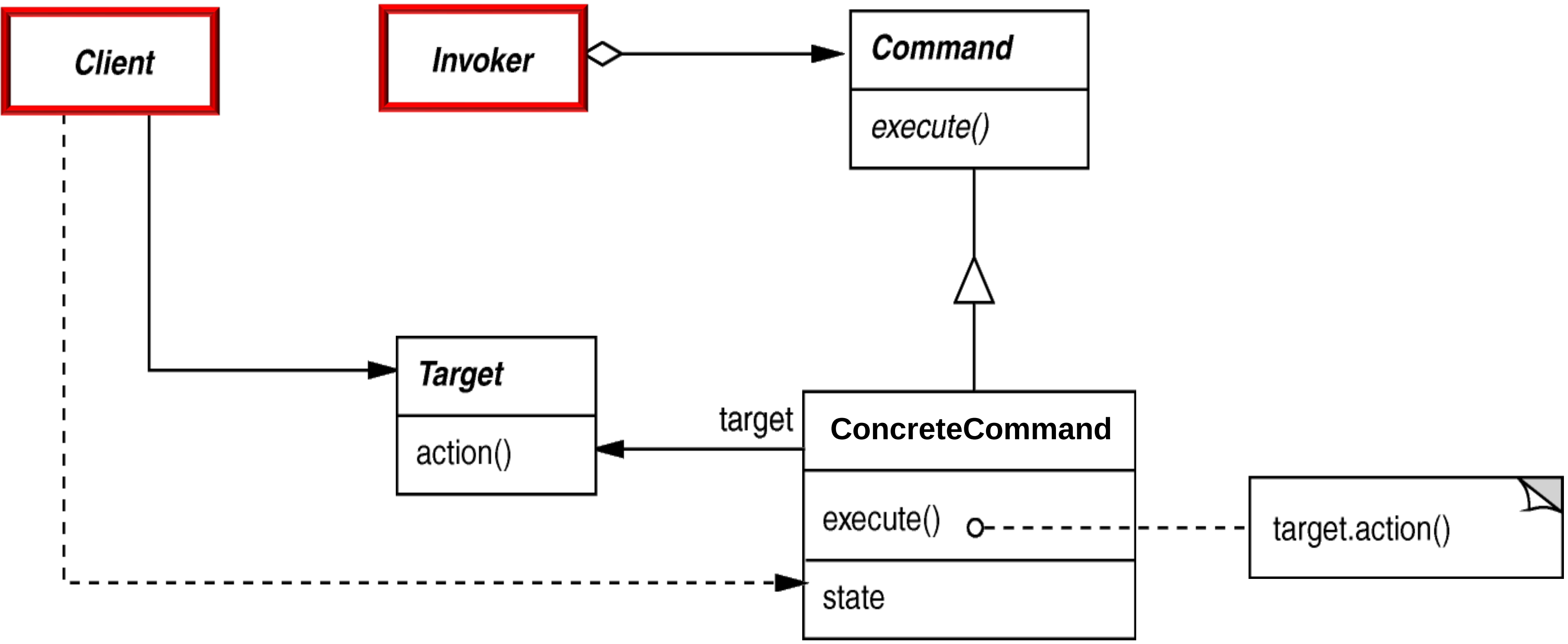
Command GoF Object Behavioral

Structure & participants



Command GoF Object Behavioral

Structure & participants



The **`Client`** & **`Invoker`** objects may be the same or different.