The Command Pattern defined: the class diagram

The Client is responsible for creating a ConcreteCommand and setting its Receiver.

The Invoker holds a command and at some point asks the command to carry out a request by calling its execute() method.

Command declares an interface for all commands. As you already know, a command is invoked through its execute() method, which asks a receiver to perform an action. You'll also notice this interface has an undo() method, which we'll cover a bit later in the chapter.

Client

Invoker

<<interface>> Command

execute()
undo()

Receiver

action()

ConcreteCommand

execute()
undo()

The Receiver knows how to perform the work needed to carry out the request. Any class can act as a Receiver.

The execute method invokes the action(s) on the receiver needed to fulfill the request.

The ConcreteCommand defines a binding between an action and a Receiver. The Invoker makes a request by calling execute() and the ConcreteCommand carries it out by calling one or more actions on the Receiver.

How does the design of the Command Pattern support the decoupling of the invoker of a request and the receiver of the request?