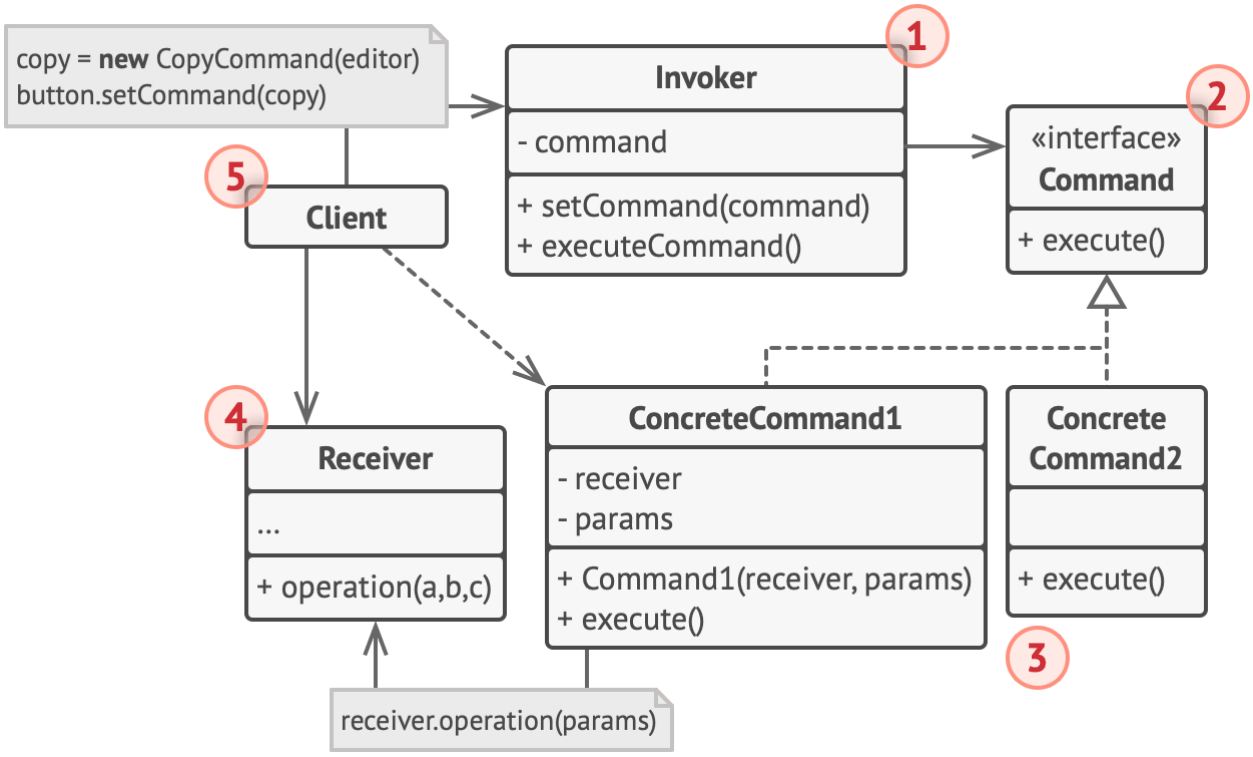

1. The **Sender** class (aka *invoker*) is responsible for initiating requests. This class must have a field for storing a reference to a command object. The sender triggers that command instead of sending the request directly to the receiver. Note that the sender isn't responsible for creating the command object. Usually, it gets a pre-created command from the client via the constructor.

2. The **Command** interface usually declares just a single method for executing the command.

3. **Concrete Commands** implement various kinds of requests. A concrete command isn't supposed to perform the work on its own, but rather to pass the call to one of the business logic objects. However, for the sake of simplifying the code, these classes can be merged.

   Parameters required to execute a method on a receiving object can be declared as fields in the concrete command. You can make command objects immutable by only allowing the initialization of these fields via the constructor.

4. The **Receiver** class contains some business logic. Almost any object may act as a receiver. Most commands only handle the details of how a request is passed to the receiver, while the receiver itself does the actual work.

5. The **Client** creates and configures concrete command objects. The client must pass all of the request parameters, including a receiver instance, into the command's constructor. After that, the resulting command may be associated with one or multiple senders.