Command

Design Pattern

Rohan Bedarkar

CSPP 51023 - Winter 2011

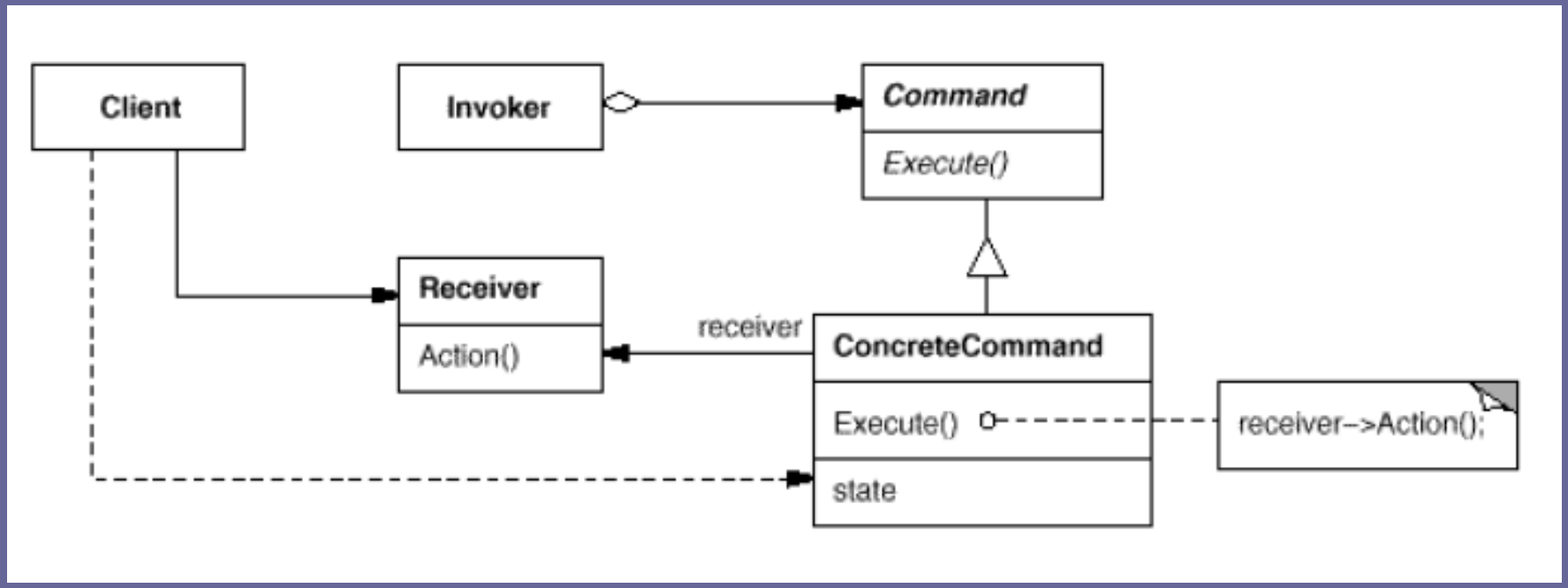
“I don't know, it just seems like a "glorified" function call to me.”

..and that’s in-fact true!

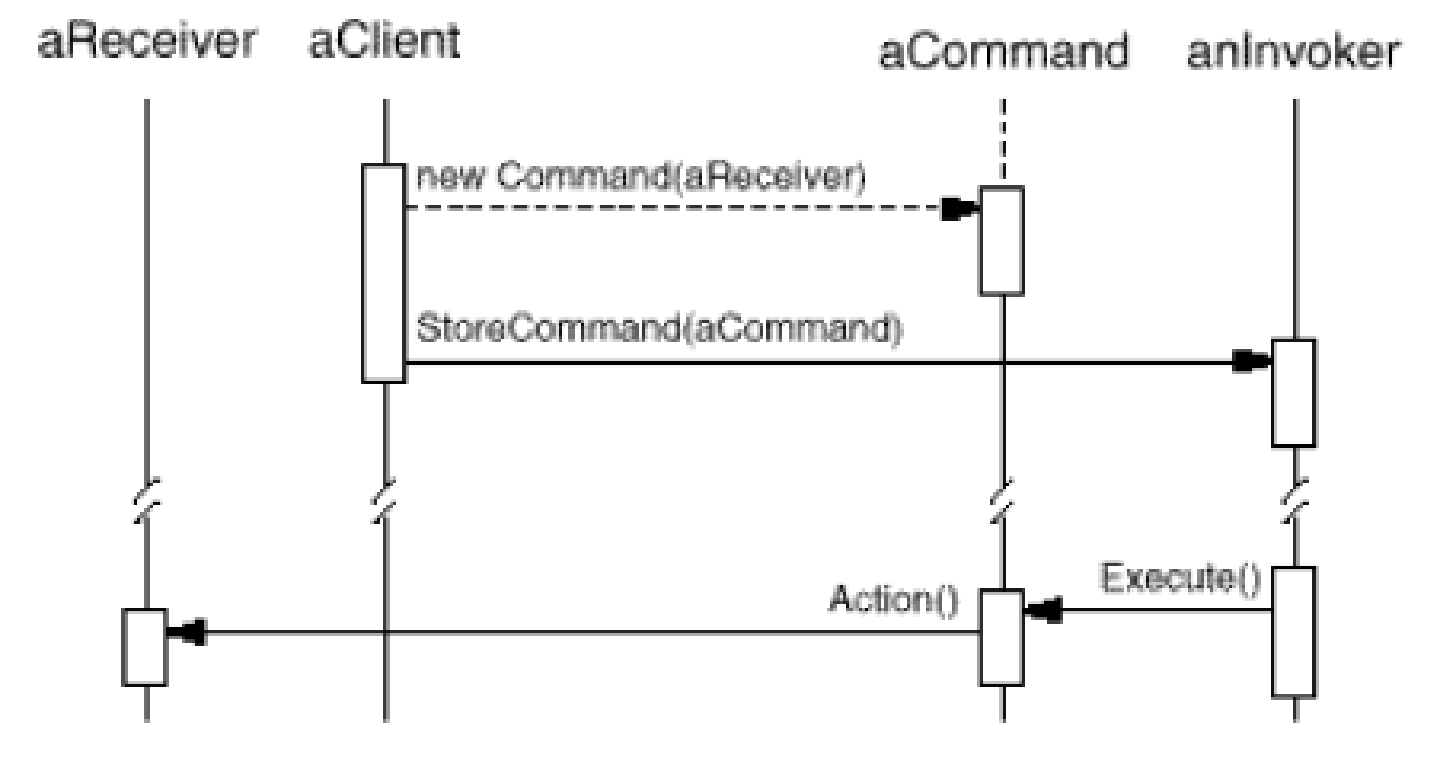
Overview

- Decouples object that invokes the operation from the one that actually performs it
- Commands are first-class objects so they can be used like any other objects
- You can sequence them into macros
- Easy to add new commands without changing existing classes

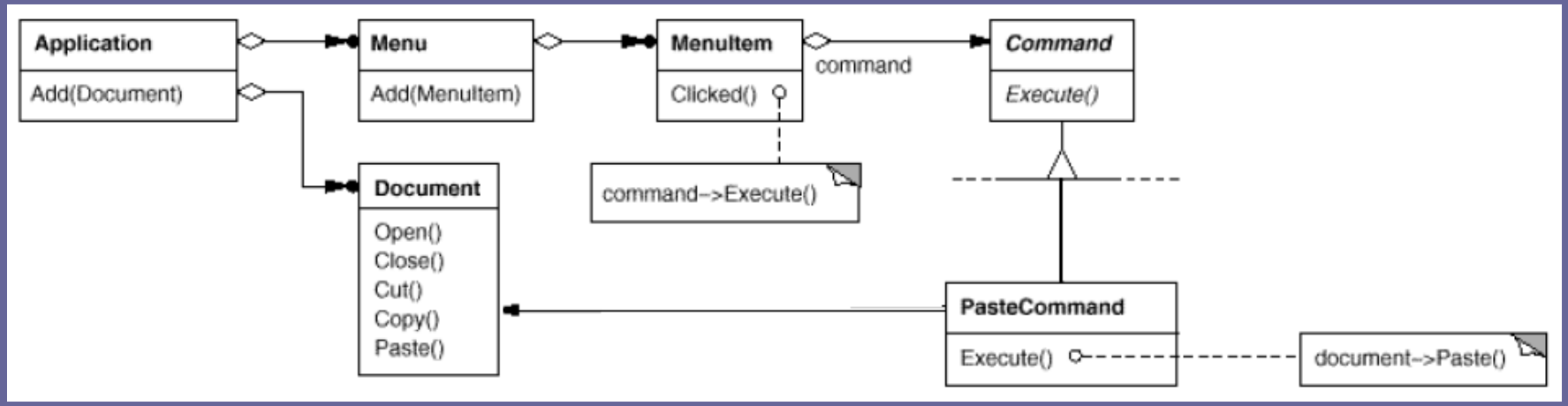
Participants



Sequence of Operation



Paste-Command Example



Applications

- Used to implement an object oriented version of callbacks
- Specify, queue and execute requests at different times
- Support “Undo” feature
- Support logging of changes so they can be reapplied after a system crash

Step 1: Define interface

- Define “Command” interface with method Execute()
- Define “UndoableCommand” with method Undo()

```
public abstract class Command
{
    public abstract void Execute();
}

public abstract class UndoableCommand : Command
{
    public abstract void Undo();
}
```

Step 2: Implement concrete class

- Create derived class: BoldCommand
- Encapsulate: a receiver, a method, arguments (if any)

```
class BoldCommand : UndoableCommand
{
    private Document document;
    private string previousText;
    public BoldCommand(Document doc)
    {
        this.document = doc;
        previousText = this.document.Text;
    }

    public override void Execute()
    {
        document.BoldSelection();
    }

    public override void Undo()
    {
        document.Text = previousText;
    }
}
```

Receiver

"Do"

"Undo"

Method to invoke

Step 3: Manage commands

- Create a command manager for multiple commands

Stack of undoable commands

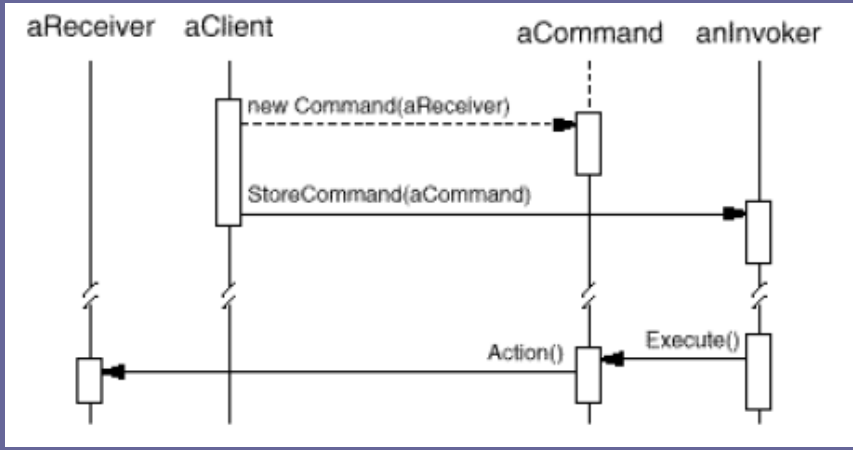
```
class CommandManager
{
    private Stack commandStack = new Stack();

    public void ExecuteCommand(Command cmd)
    {
        cmd.Execute();
        if (cmd is UndoableCommand)
        {
            commandStack.Push(cmd);
        }
    }

    public void Undo()
    {
        if (commandStack.Count > 0)
        {
            UndoableCommand cmd = (UndoableCommand)commandStack.Pop();
            cmd.Undo();
        }
    }
}
```

"undo" command

"do" command

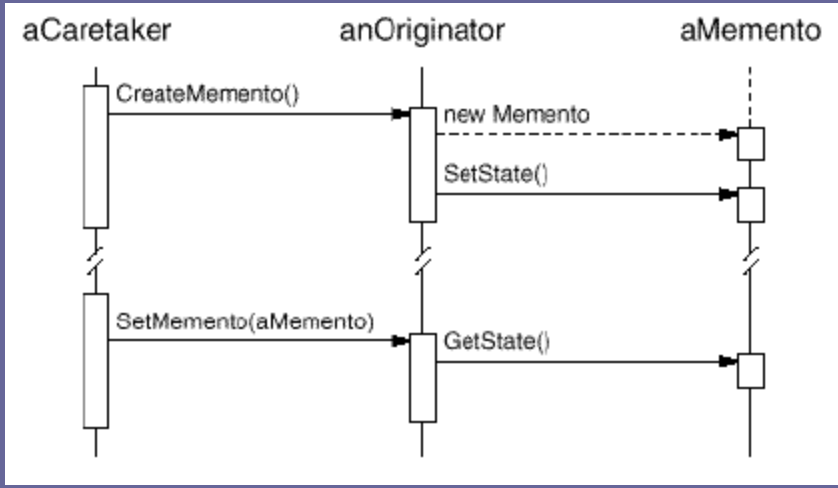
COMMAND

Encapsulate a function call into an object



MEMENTO

Store the internal state of Originator object



Bibliography

- http://www.codeproject.com/KB/architecture/sharped.aspx
- http://johnlindquist.com/2010/09/09/patterncraft-command-pattern/
- http://www.javaworld.com/javatips/jw-javatip68.html?page=2
- http://home.earthlink.net/~huston2/dp/command.html