LECTURE-8

Basic Concepts of Patterns

Basic Concepts of Patterns

- **What is a pattern?**
- **Part of a pattern**
- **Some example patterns**
- **Modeling a pattern with UML**

What is a pattern?

What is a pattern?

- A common problem
  - and a proven solution
  - in a context
- A structured, packaged problem solution in literary form.
- A way of recording experience, “best practices”
  - In a standard format
  - A repository for knowledge
- “What’s new is that there’s nothing new here.

Patterns are about what works. Patterns give us a way to talk about what works.” – Brian Foote, 1997.

Parts of a Pattern

- ***Name*:**

  a good name is essential because pattern names help designers to communicate.

- ***Context*:**

  where the pattern can be applied

- ***Forces*:**

  to be balanced in the solution

- ***Problem*:**

  usually describes in terms of the forces.

- ***Solution*:**

  a proven way of balancing the forces

Some Example Patterns

- Alexander pattern: Window place
- Architectural pattern: MVC
- Design pattern: Observer
- Analysis pattern: Party

An Alexander Pattern - Window Place

- **Name: Window Place**
- **Context and forces:**

  a room has a window and a place to sit

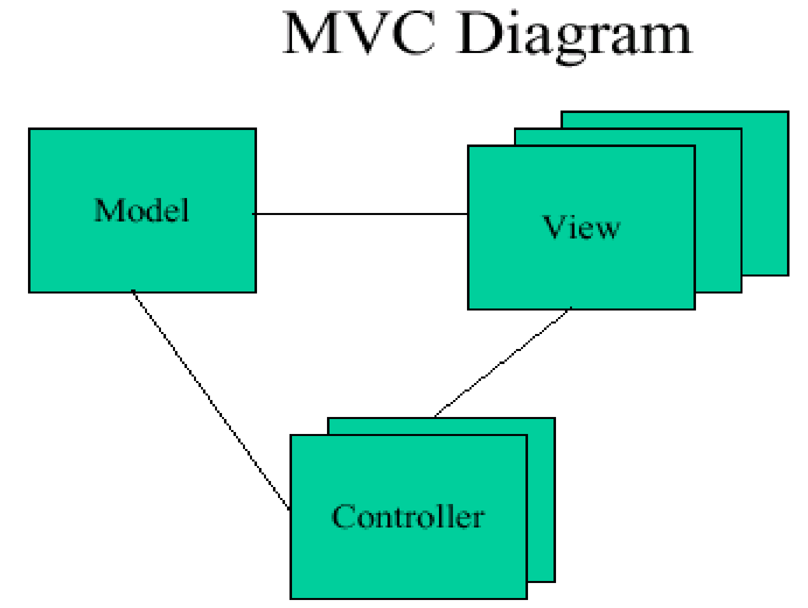
  - We are drawn towards the light
  - We want to sit comfortably
- **Problem:**

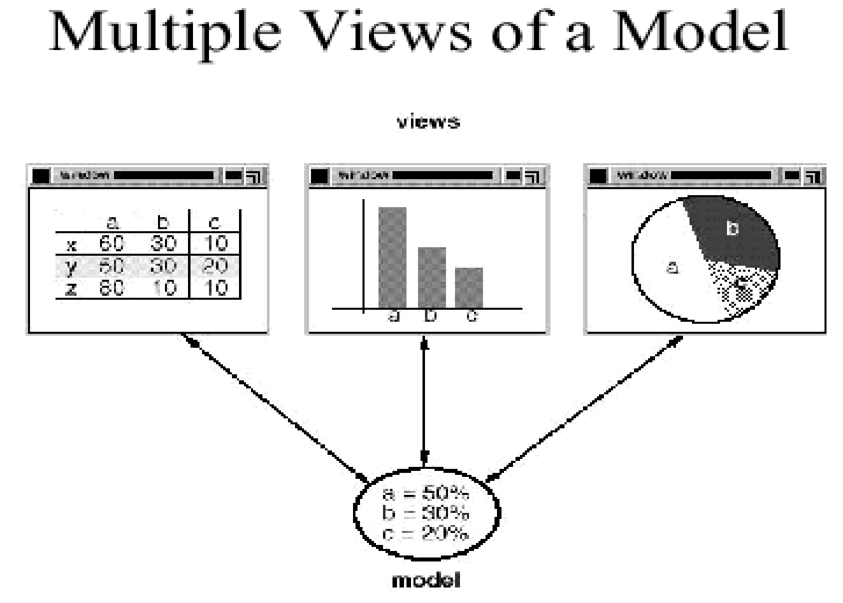
  how to be comfortable and still near the natural light
- **Solution:**

  place the comfortable sitting place near the window (e.g., a window seat)

Architectural Pattern - MVC

- **Name:** MVC (Model-View-Controller)
- **Context and forces:** we have a data model and several representations of the data
  - We want to modularize the system
  - Data representation must be kept up to date
- **Problem:** how to modularize the system
- **Solution:** the model holds the data (and does data modification), the view represents the data, the controller handles user input

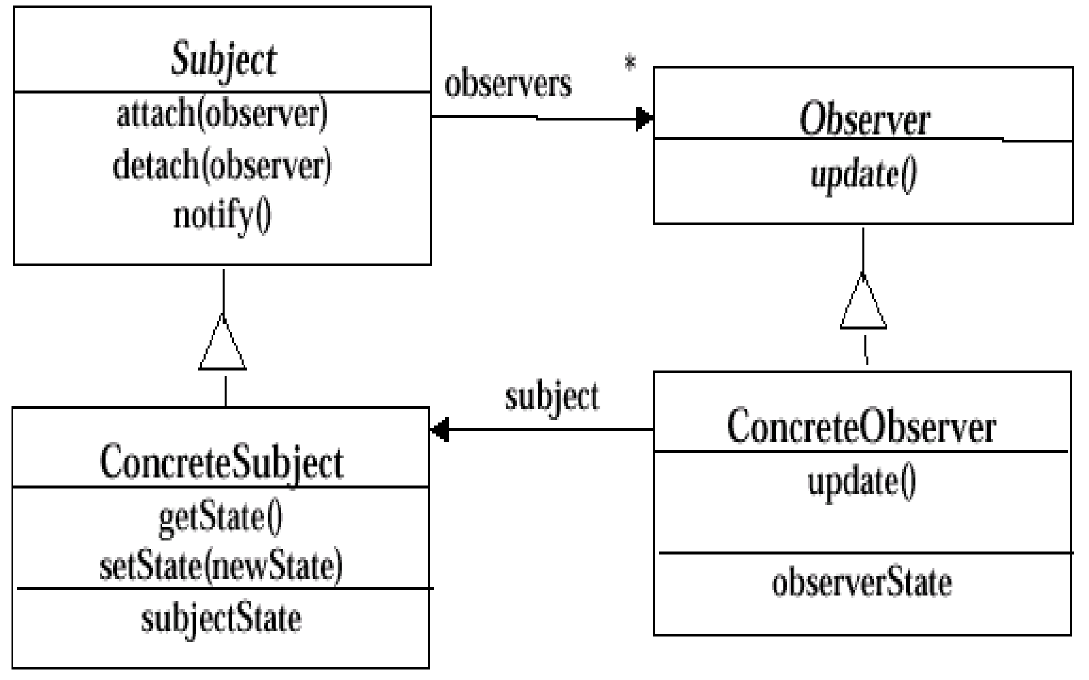


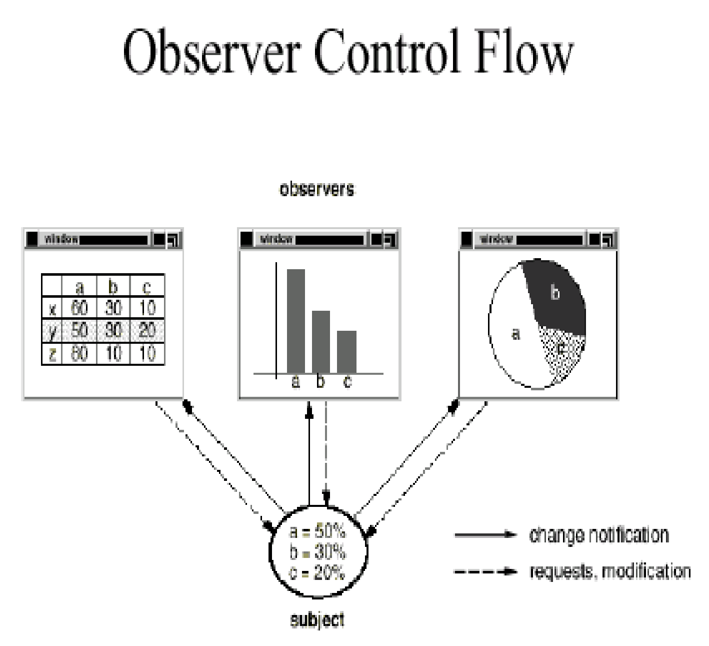


	a	b	c
x	60	30	10
y	50	30	20
z	60	10	10

Design Patterns - Observer

- **Name:** Observer
- **Context and forces:** data is kept in one object and displayed in other objects
  - We want to distribute the functions
  - We want the system to stay consistent
- **Problem:** keep the information consistent
- **Solution:** the display objects observe the object holding the data and are notified of changes in the data





	a	b	c
x	60	30	10
y	50	30	20
z	80	10	10

Analysis Pattern - Party

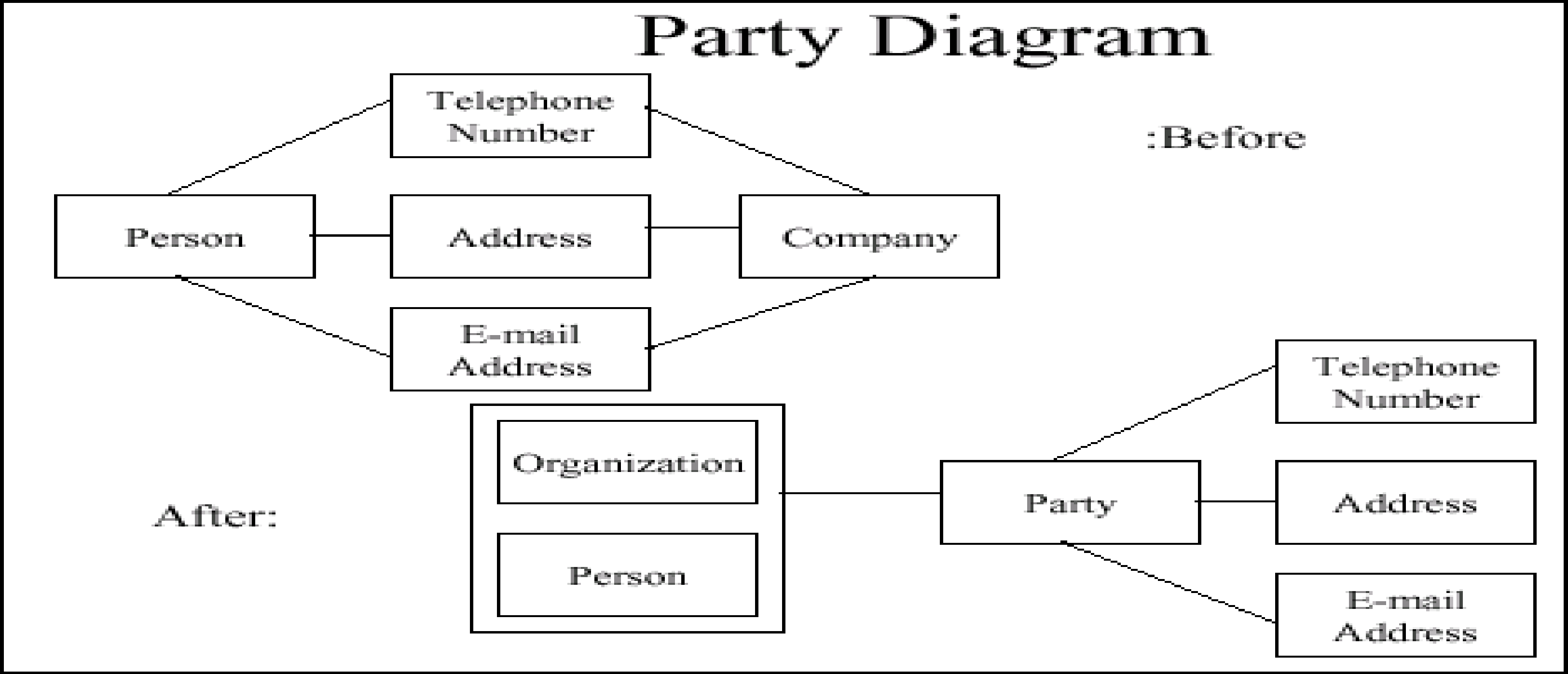
- **Name:** Party
- **Context and forces:**

we have people and organizations that both take on roles in the system

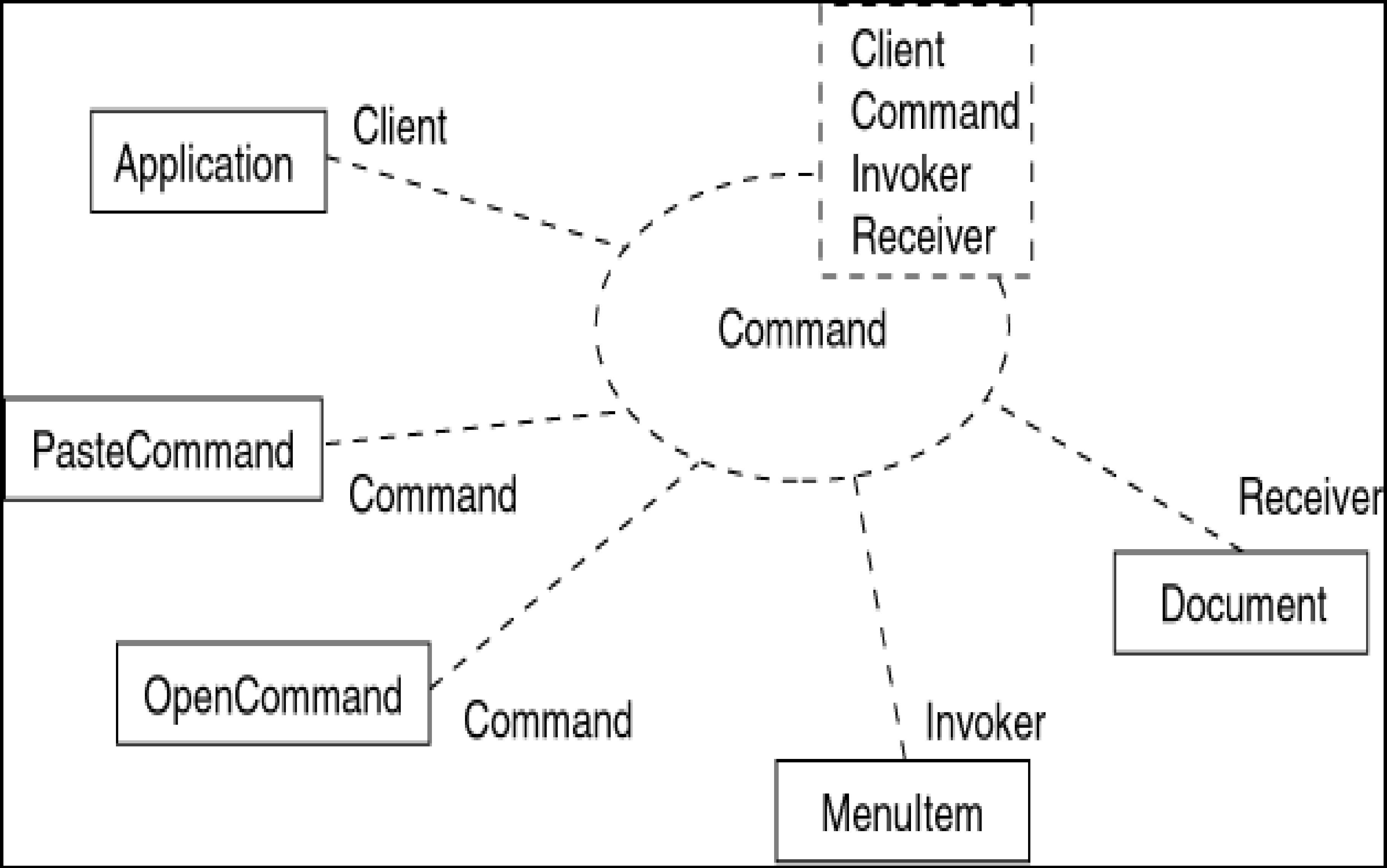
– We want to allow several types of entities

– We want to treat all entities consistently

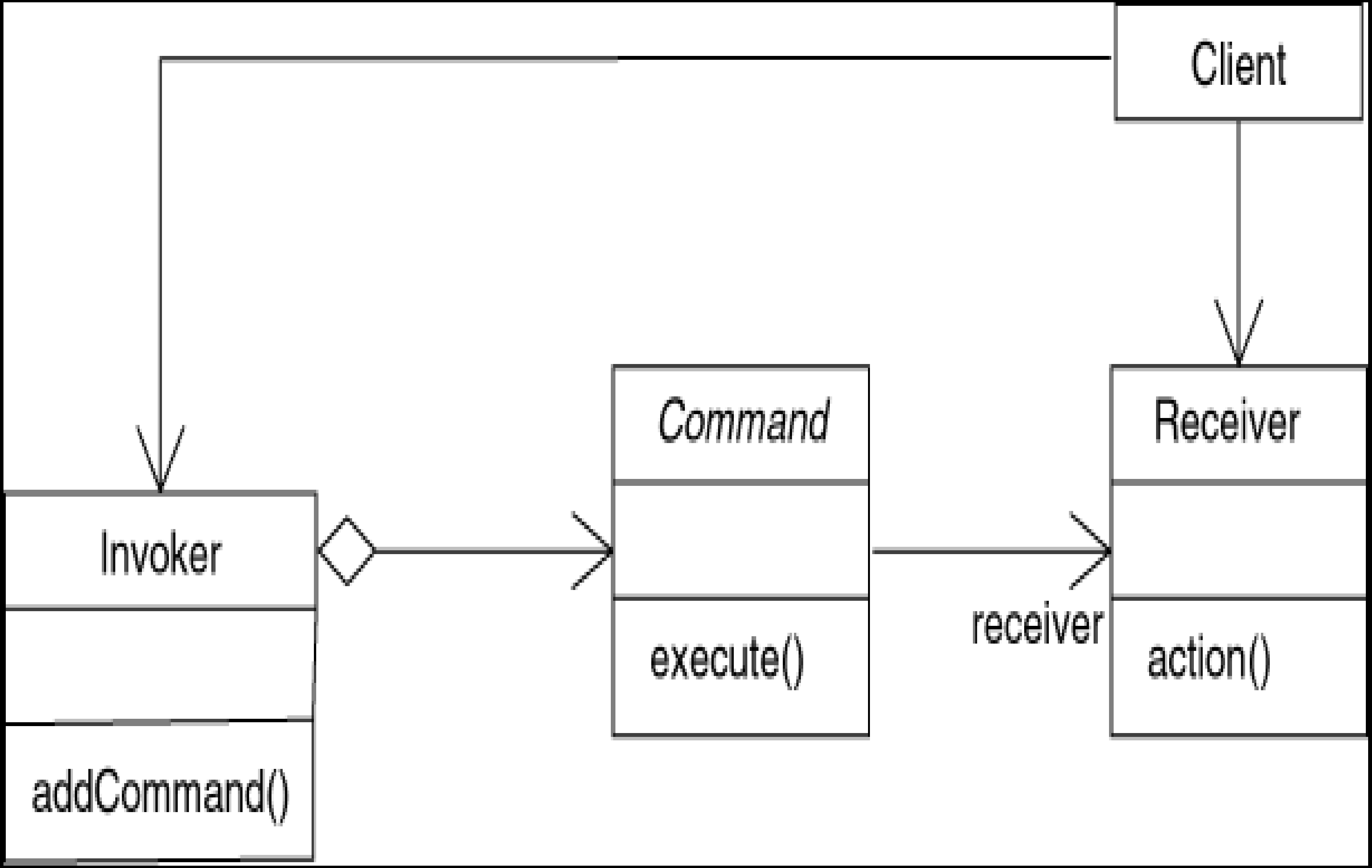
- **Problem:** how can we treat them uniformly without complicating the diagrams?
- **Solution:** Add a party entity which unifies the two entities



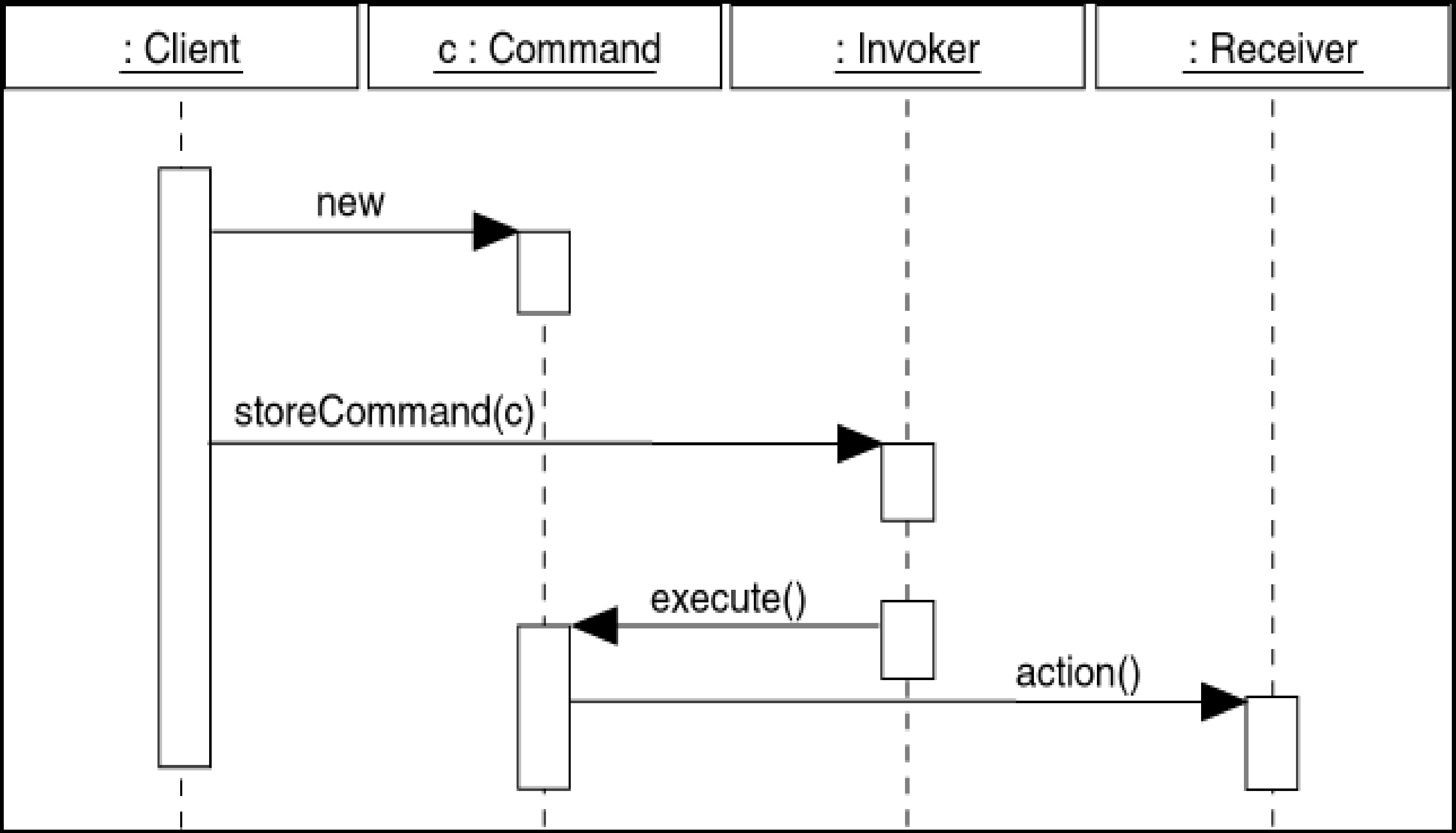
Modeling a pattern with UML



Modeling a pattern with UML



Modeling a pattern with UML


: Client
c : Command
: Invoker
: Receiver
new
storeCommand(c)
execute()
action()