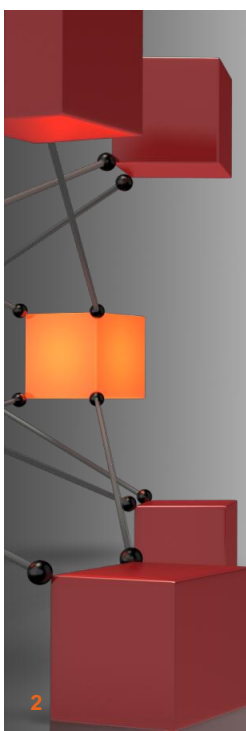

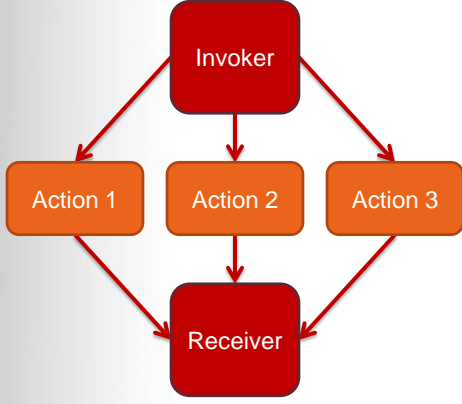


**Command Design Pattern**  
COS 121 – Christoph Stallmann




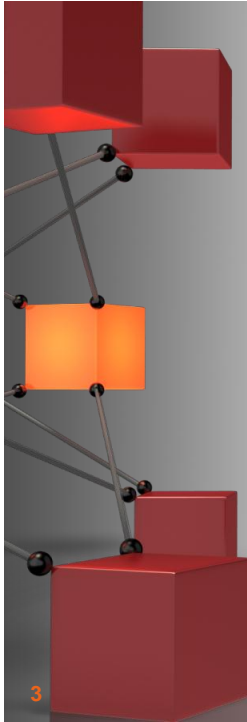
**Introduction**

- Encapsulate a request or action as an object.
- Classification: Behavioural



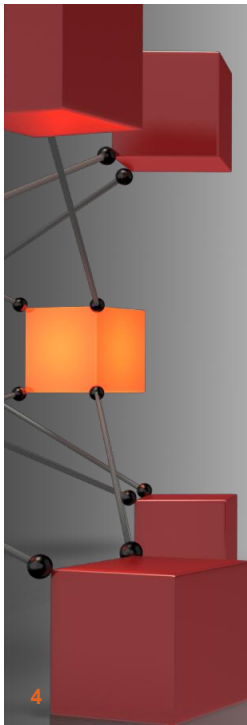
```
graph TD; Invoker[Invoker] --> Action1[Action 1]; Invoker --> Action2[Action 2]; Invoker --> Action3[Action 3]; Action1 --> Receiver[Receiver]; Action2 --> Receiver; Action3 --> Receiver;
```



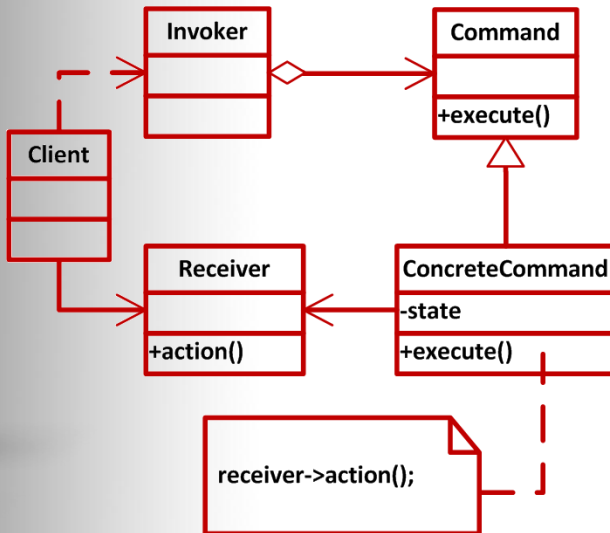


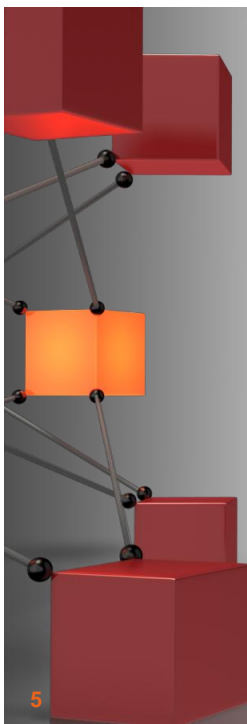
## Reason

- Easy to add new commands.
- Queue different commands and execute them in order.
- Log different actions.
- Support undoable operations.



## Structure



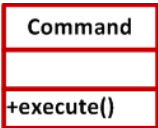



## Participants – Command


- Often abstract.
- Defines an interface for executing a command or action.

```

classDiagram
    class Command {
        +execute()
    }
  
```

5



## Participants – Concrete Command

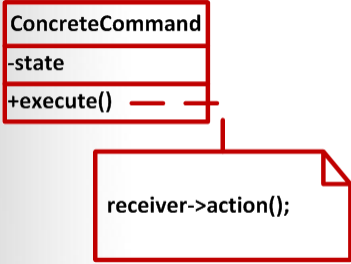

- Defines a binding between the receiver and an action.
- Implements the execute function.
  - The execute() function calls the Receiver's action() function.

```

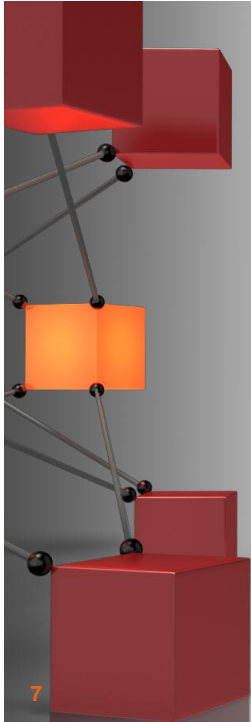
classDiagram
    class ConcreteCommand {
        -state
        +execute()
    }
  
```

```

classDiagram
    class ConcreteCommand {
        -state
        +execute()
    }
    class Receiver {
        +action()
    }
    ConcreteCommand --> Receiver : receiver->action();
  
```

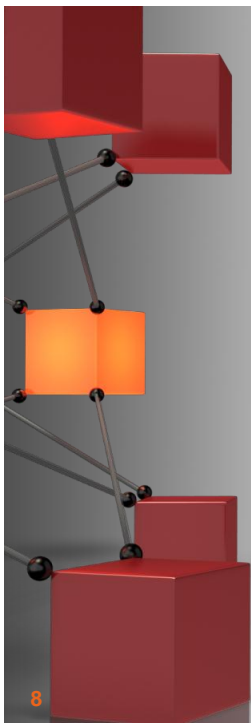
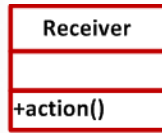



6



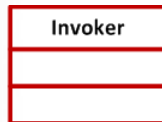
## Participants – Receiver

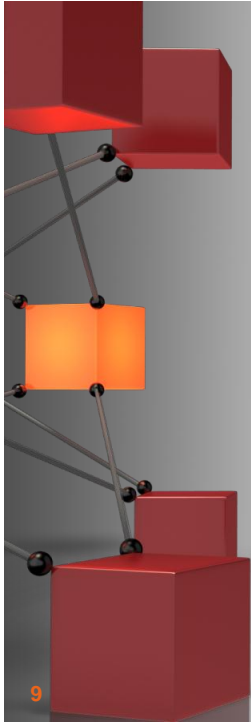
- Knows how to execute a specific action.
- Any class can serve as a Receiver.



## Participants – Invoker

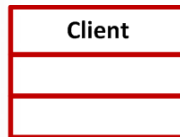
- Asks the Command to carry out a request.



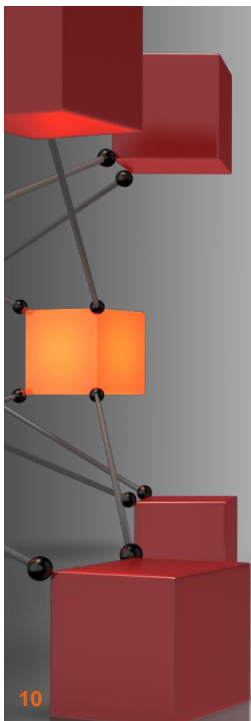


## Participants – Client

- The application.
- Creates the Concrete Command and sets its receiver.
- Often this responsibility is delegated to the Invoker.



9

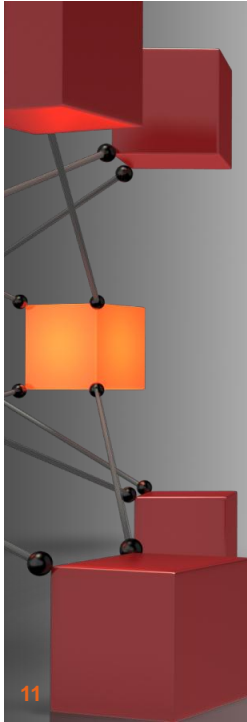


## The Process

- Initial state.

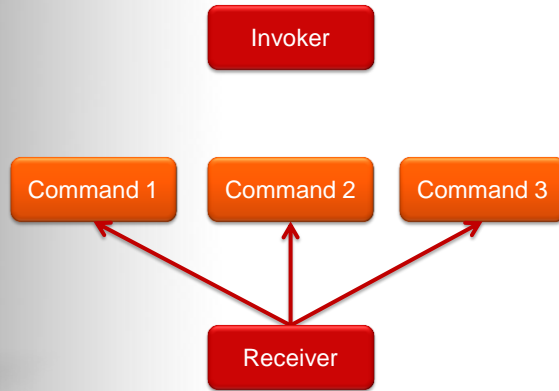


10

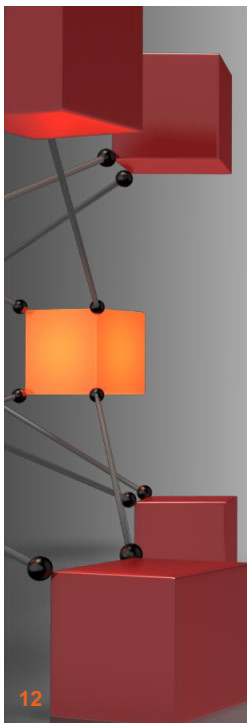


## The Process

- Connect the Receiver to the Commands.

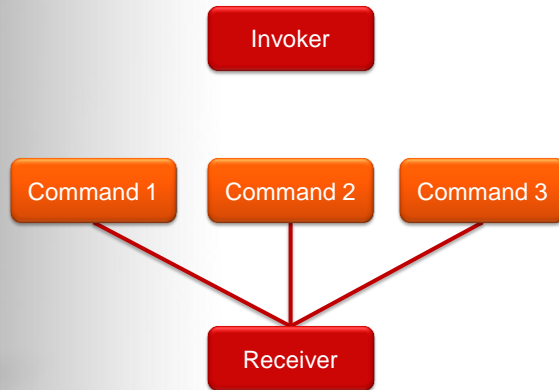


11

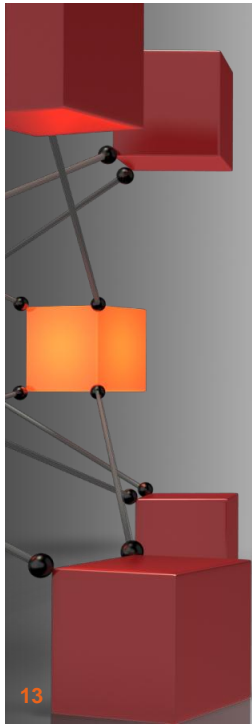


## The Process

- Commands are now connected to the Receiver.

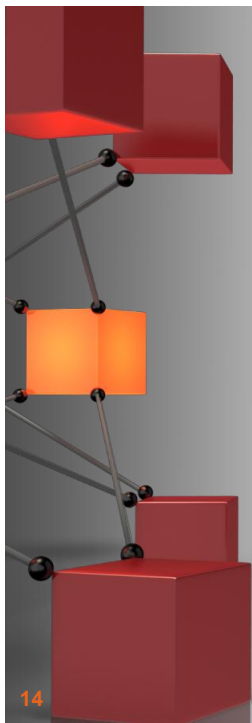
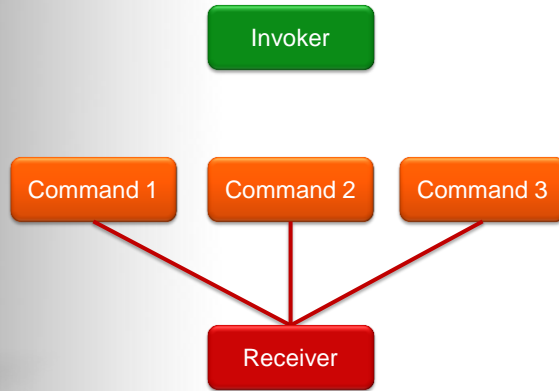


12



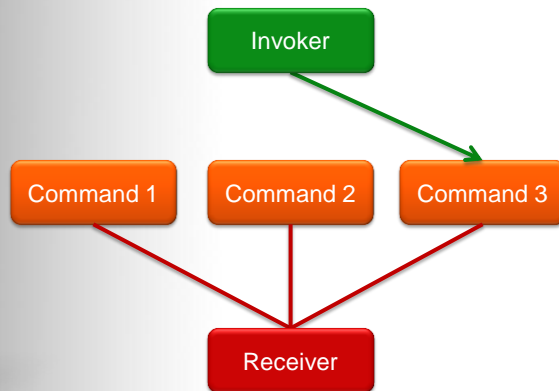
## The Process

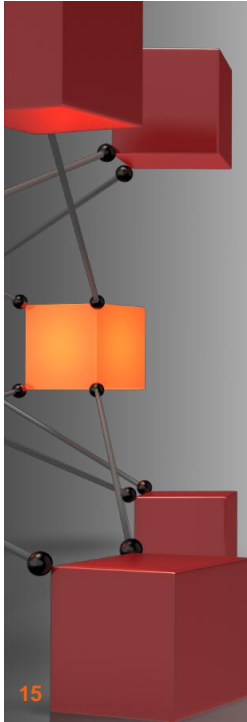
- The Invoker changes or receives external request.



## The Process

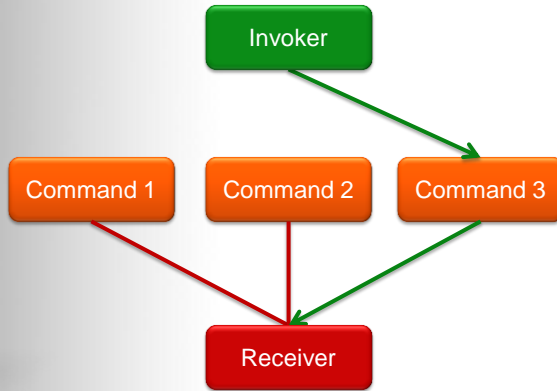
- The Invoker executes a specific Command.



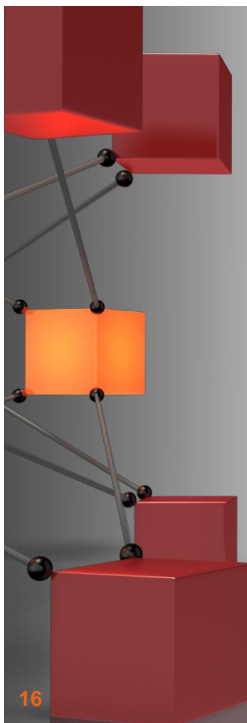


## The Process

- The Command calls a specific action on the receiver.

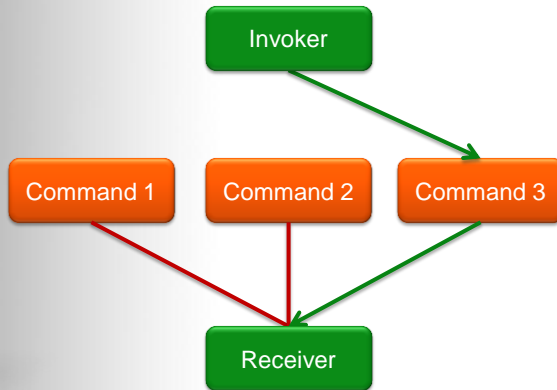


15



## The Process

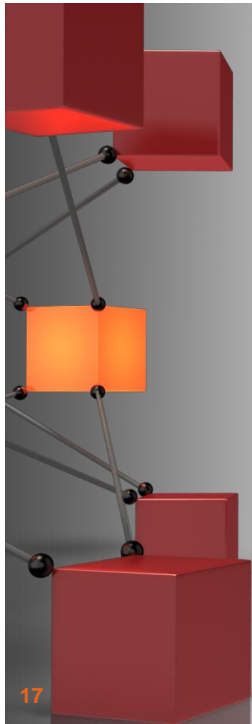
- The Receiver executes a specific action.



16

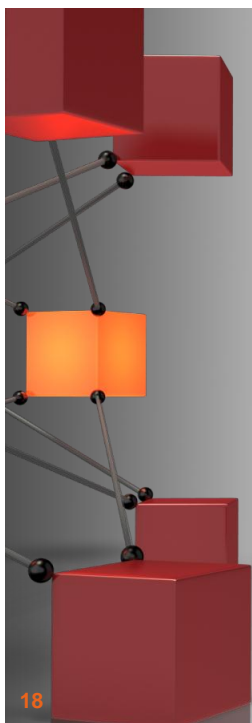
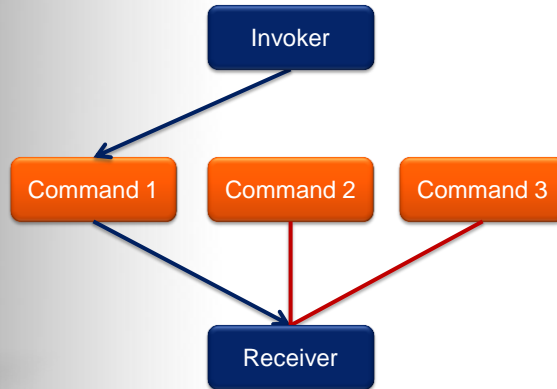






## The Process

- Depending on the request, the Invoker might also execute a different command.



## Example - Video

- <http://youtu.be/K1XeTF87Rgk>




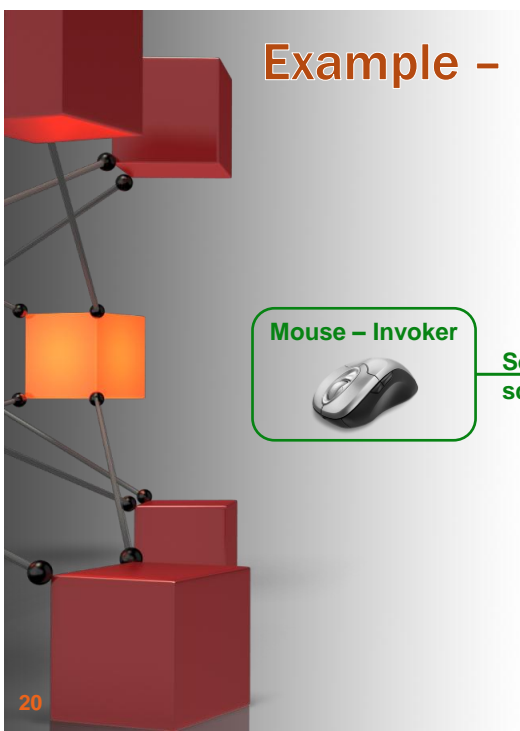


### Example - Layout

Mouse - Invoker



Squad - Receiver

### Example - Layout

Mouse - Invoker



Select squad



Squad - Receiver






# Example - Layout

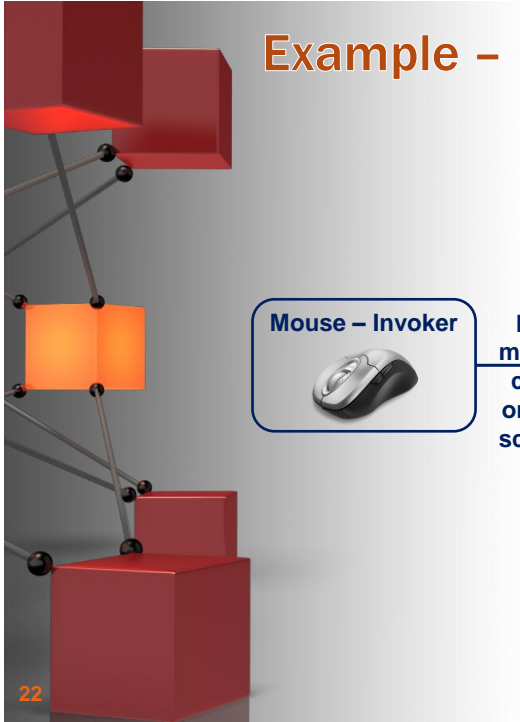
Mouse - Invoker

 A rounded rectangular box with a green border containing the text "Mouse - Invoker" and a realistic computer mouse icon.

Squad - Receiver

 A rounded rectangular box with a green border containing the text "Squad - Receiver" and four identical icons of a soldier in a black tactical suit holding a rifle.

21



# Example - Layout

Mouse - Invoker

 A rounded rectangular box with a blue border containing the text "Mouse - Invoker" and a realistic computer mouse icon.

Left mouse click on the screen

 An arrow pointing from the mouse icon box towards the squad box.

Squad - Receiver

 A rounded rectangular box with a green border containing the text "Squad - Receiver" and four identical icons of a soldier in a black tactical suit holding a rifle.

22




### Example - Layout

**Mouse - Invoker**

**Squad - Receiver**

23




### Example - Layout

**Mouse - Invoker**

**Squad - Receiver**

Mouse click on the grenade button

24




### Example - Layout

**Mouse - Invoker**

**Squad - Receiver**

25




### Example - Layout

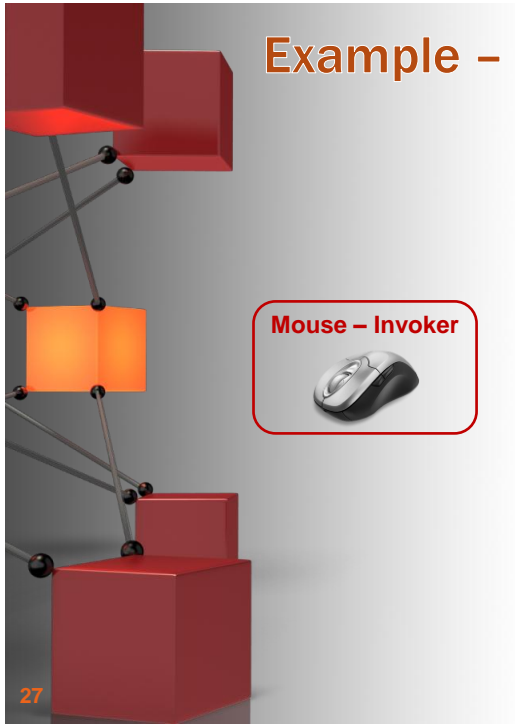
**Mouse - Invoker**

**Squad - Receiver**

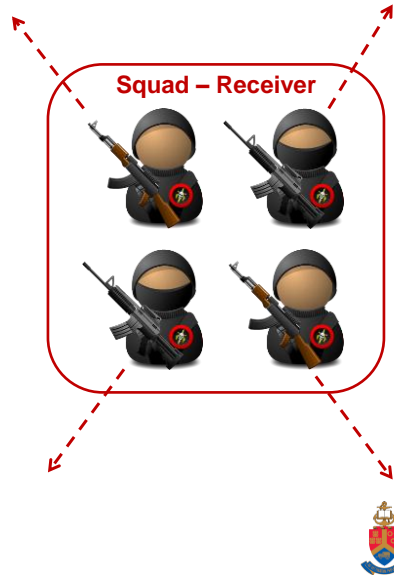
Mouse click on the retreat button

26

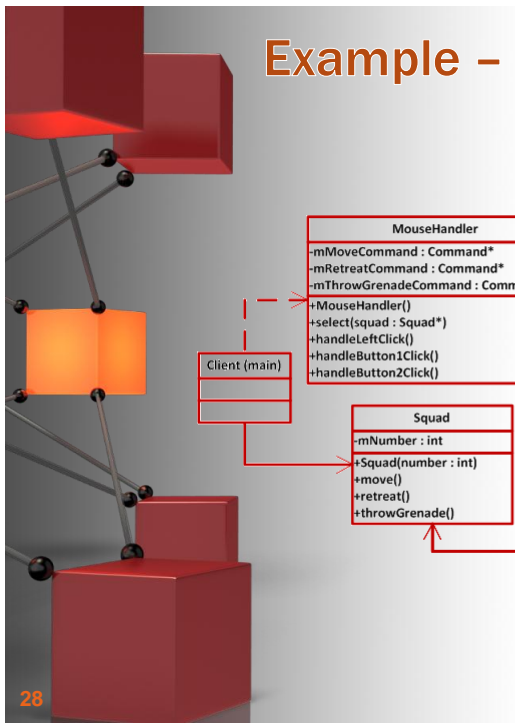




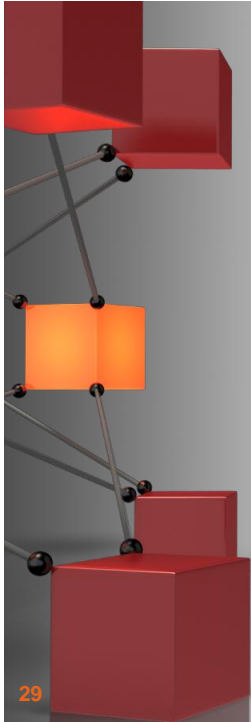
## Example - Layout



27

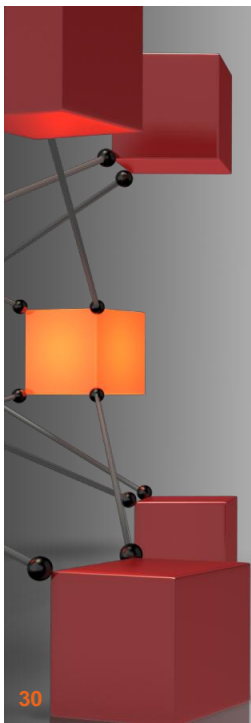


28



## Example - Code

29



## Example - Output

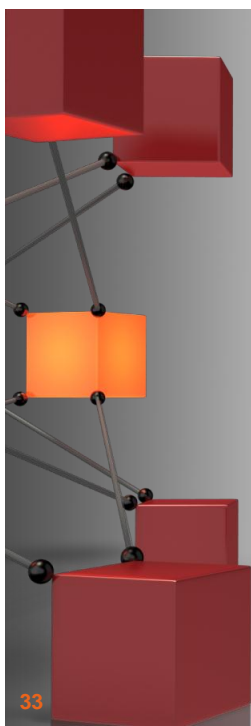
30

```
visore@ubuntu: ~/Desktop/Com
File Edit View Search Terminal Help
*****
**      Game Patterns      **
**      Command           **
*****
**      Christoph Stallmann  **
**      University of Pretoria **
**      COS121 - 2012       **
*****
Squad 1 is moving.
Squad 1 is throwing a grenade.
Squad 2 is moving.
Squad 2 is retreating.
Squad 1 is retreating.
```



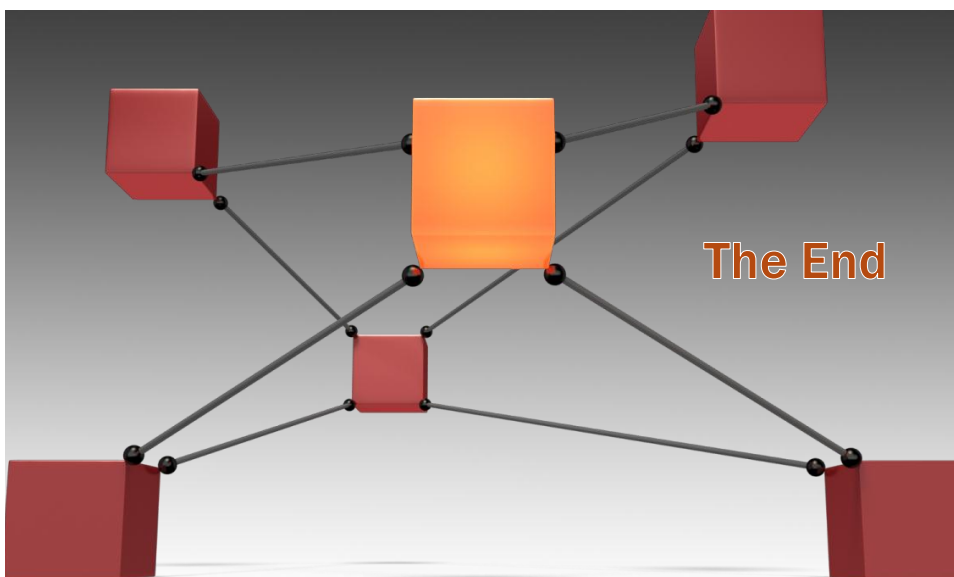






## Related Patterns

- Chain of Responsibility:
  - Commands can serve as request objects.
- Composite:
  - To implement macro Commands.
- Memento:
  - Keep state for undo operations.
- Prototype:
  - Commands can be copied to be placed in a history list.



## Command Design Pattern

COS 121 – Christoph Stallmann

