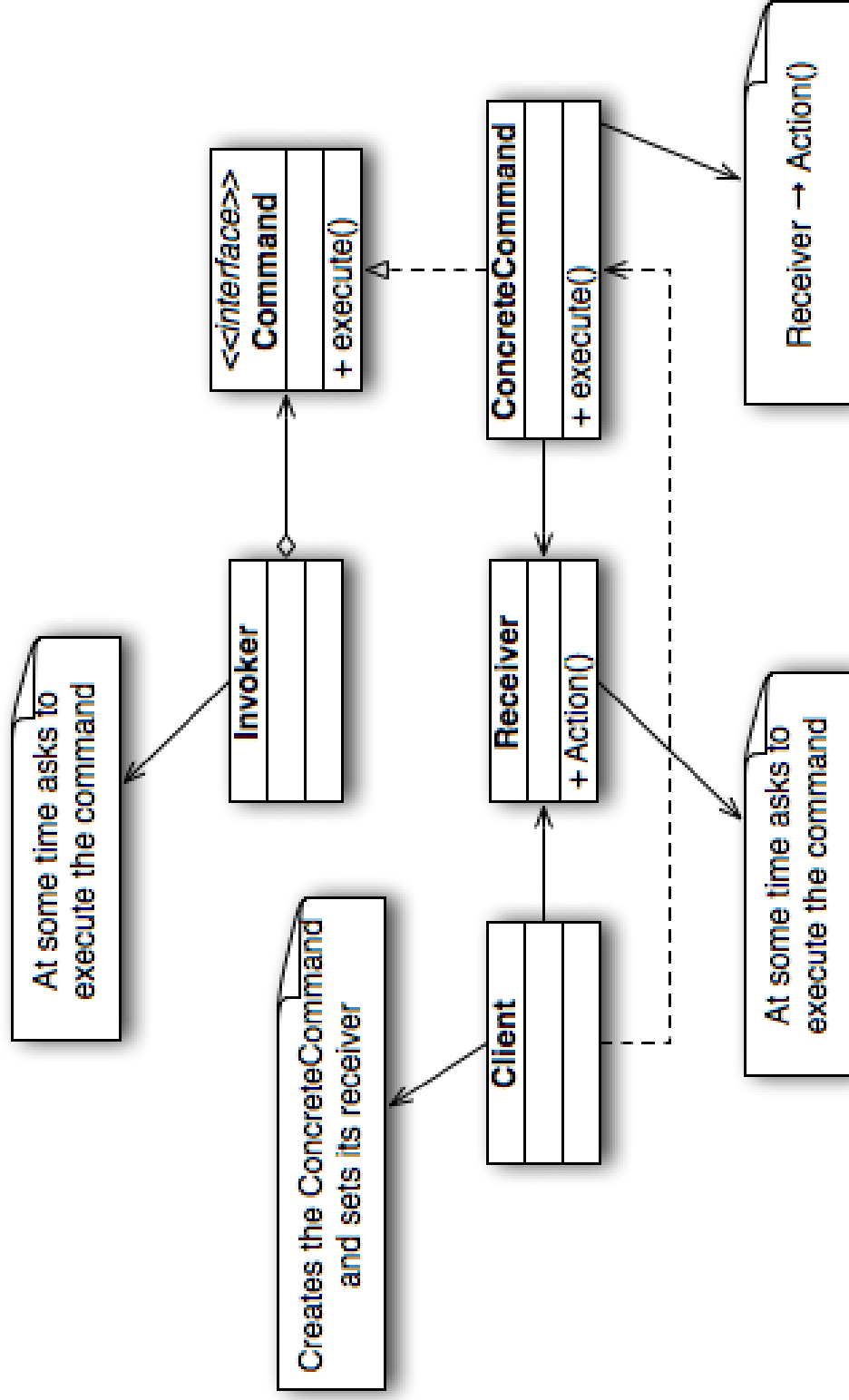


# Estrutura



# Participantes

- ▶ *Client* – instancia o objeto de comando e fornece as informações necessárias para chamar o método em um momento posterior. Cria um objeto *ConcreteCommand* especificando seu receptor.



# Participantes

- ▶ Invoker – decide quando o método deve ser chamado.
- ▶ Receptor – é uma instância da classe que contém o código do método. Sabe como realizar uma operação associada com o cumprimento de um pedido. Qualquer classe pode servir como receptor.



# Exemplo – Interruptor de Lâmpada

- ▶ Considere um `switch(interruptor)` "simples".
- ▶ Neste exemplo, iremos configurar o `switch` com 2 comandos: para acender a luz e desligar a luz.



# Exemplo – Interruptor de Lâmpada

- ▶ A vantagem desta implementação particular do padrão command é que o switch pode ser usado com qualquer aparelho, não apenas uma luz.
- ▶ Considere, então, as seguintes classes:



# Exemplo – Implementação

► */\*A interface Command \*/*

```
public interface Command {  
    void execute();  
}
```




# Exemplo – Implementação

```
/*A classe Invoker */  
  
public class Switch {  
    private List<Command> history = new ArrayList<Command>();  
  
    public Switch() {}  
  
    public void storeAndExecute(Command cmd) {  
  
        this.history.add(cmd); // optional  
  
        cmd.execute();  
    }  
}
```



# Exemplo – Implementação

```
/*A classe Receiver */  
  
public class Light {  
    public Light() { }  
  
    public void turnOn() {  
        System.out.println("The light is on");  
    }  
  
    public void turnOff() {  
        System.out.println("The light is off");  
    }  
}
```





# Exemplo – Implementação

*/\*the Command for turning on the light –  
ConcreteCommand #1\*/*

```
public class FlipUpCommand implements Command {  
    private Light theLight;
```

```
    public FlipUpCommand(Light light) {  
        this.theLight = light;  
    }
```

```
    public void execute(){  
        theLight.turnOn();  
    }  
}
```



# Exemplo – Implementação

*/\*the Command for turning off the light –  
ConcreteCommand #2\*/*

```
public class FlipDownCommand implements Command {  
    private Light theLight;
```


```
    public FlipDownCommand(Light light) {  
        this.theLight = light;  
    }
```

```
    public void execute() {  
        theLight.turnOff();  
    }  
}
```



# Exemplo – Implementação

```
/*The test class or client*/  
  
public class PressSwitch {  
  
    public static void main(String[] args){  
  
        Light lamp = new Light();  
  
        Command switchUp = new FlipUpCommand(lamp);  
        Command switchDown = new  
            FlipDownCommand(lamp);  
  
        Switch s = new Switch();  
        (...)
```



# Exemplo – Implementação

```
try {  
    if (args[0].equalsIgnoreCase("ON")) {  
        s.storeAndExecute(switchUp);  
        System.exit(0);  
    }  
  
    if (args[0].equalsIgnoreCase("OFF")) {  
        s.storeAndExecute(switchDown);  
        System.exit(0);  
    }  
    System.out.println("Argument \"ON\" or \"OFF\" is  
        required.");  
}
```

(...)



# Exemplo – Implementação

```
catch (Exception e) {  
    System.out.println("Arguments required.");  
}  
    } //fim do main  
} //fim da classe PressSwitch
```

