Паттерн Command

Назначение

Инкапсулирует запрос как объект, позволяя тем самым задавать параметры клиентов для обработки соответствующих запросов, ставить запросы в очередь или протоколировать их, а также поддерживать отмену операций.

Применимость

Используйте паттерн команда, когда хотите:

- Параметризовать объекты выполняемым действием, как в случае с пунктами меню.
- Определять, ставить в очередь и выполнять запросы в разное время.
- Поддержать отмену операций. Операция Execute объекта Command может сохранить состояние,
- необходимое для отката действий, выполненных командой.
- Поддержать протоколирование изменений, чтобы их можно было выполнить повторно после
- аварийной остановки системы.
- Структурировать систему на основе высокоуровневых операций, построенных из примитивных.

Участники

- Command - команда: объявляет интерфейс для выполнения операции;
- ConcreteCommand - конкретная команда: определяет связь между объектом-получателем Receiver и действием; реализует операцию Execute путем вызова соответствующих операций объекта Receiver;
- Client - клиент: создает объект класса ConcreteCommand и устанавливает его получателя;
- Invoker - инициатор: обращается к команде для выполнения запроса;
- Receiver (Document, Application) - получатель: располагает информацией о способах выполнения операций, необходимых для удовлетворения запроса. В роли получателя может выступать любой класс.

Результаты

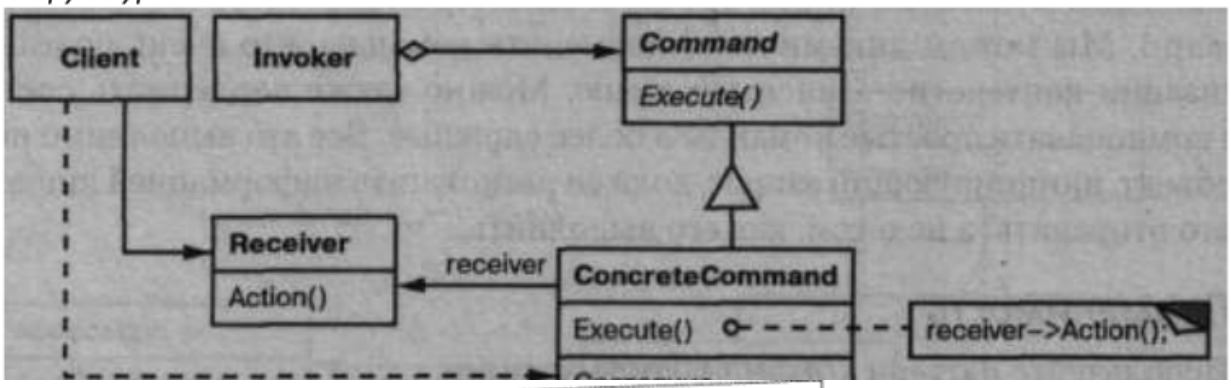
Результаты применения паттерна команда таковы:

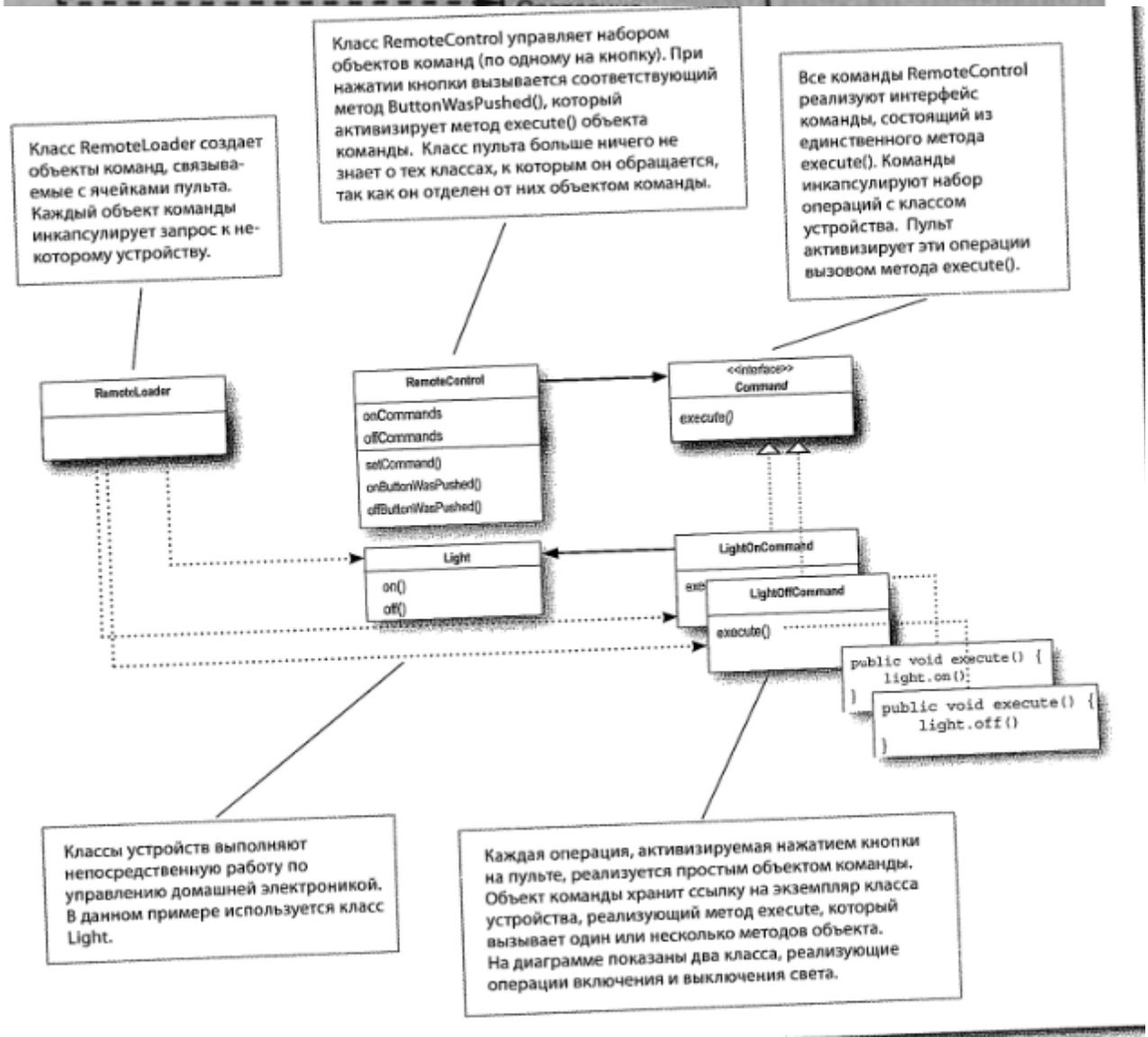
- Команда разрывает связь между объектом, инициирующим операцию, и объектом, имеющим
- информацию о том, как ее выполнить;
- Команды - это самые настоящие объекты. Допускается манипулировать ими и расширять их точно так
- же, как в случае с любыми другими объектами;
- Из простых команд можно собирать составные. В общем случае составные команды описываются
- паттерном компоновщик;
- Добавлять новые команды легко, поскольку никакие существующие классы изменять не нужно.

- Паттерн Команда отделяет объект, выдающий запросы, от объекта, который умеет эти запросы выполнять.
- Объект команды инкапсулирует получателя с операцией (или набором операций).
- Инициатор вызывает метод execute() объекта команды, что приводит к выполнению соответствующих операций с получателем.

- Возможна параметризация инициаторов командами (даже динамическая во время выполнения).
- Команды могут поддерживать механизм отмены, восстанавливающий объект в состоянии
- до последнего вызова метода execute().
- Макрокоманды — простое расширение паттерна Команда, позволяющее выполнять
- цепочки из нескольких команд.
- В них также легко реализуетсямеханизм отмены.
- На практике нередко встречаются «умные» объекты команд, которые реализуют запрос само-
- стоятельно вместо его делеги рования получателю.
- Команды также могут использоваться для реализации систем регистрации команд и поддержки транзакций.

Структура





```cpp
#include <iostream>
using namespace std;
class Command
{
public:
    virtual void execute()=0;
    virtual void undo()=0;
};
class Light{
public:
    void on()
    {
        cout<<"ON"<<endl;
    }
    void off()
    {
        cout<<"OFF"<<endl;
    }
};
class LigntOnComman:public Command
{
    Light* light;
public:
    LigntOnComman(Light& lignt):light(&lignt){}
    void execute()
    {
        light->on();
    }
    void undo()
    {
        light->off();
    }
```

```
};
class LigntOffComman:public Command
{
    Light* light;
public:
    LigntOffComman(Light& lignt):light(&lignt){}
    void execute()
    {
        light->off();
    }
    void undo()
    {
        light->on();
    }
};
class SimpleRemoteControl
{
    Command* slot;
    Command* undocommand;
public:
    SimpleRemoteControl(){};
    void setcommand(Command& com)
    {
        slot=&com;
    }
    void ButtonWasPressed()
    {
        slot->execute();
        undocommand=slot;
    }
    void ButtonUndoPressed()
    {
        undocommand->execute();
    }
};
int main() {
    SimpleRemoteControl* remoteControl=new SimpleRemoteControl();
    Light* lg=new Light();
    LigntOnComman turnon(*lg);
    LigntOffComman turnoff(*lg);
    remoteControl->setcommand(turnon);
    remoteControl->ButtonWasPressed();
    remoteControl->setcommand(turnoff);
    remoteControl->ButtonWasPressed();
    remoteControl->ButtonUndoPressed();
    std::cout << "Hello, World!" << std::endl;
    return 0;
}
```