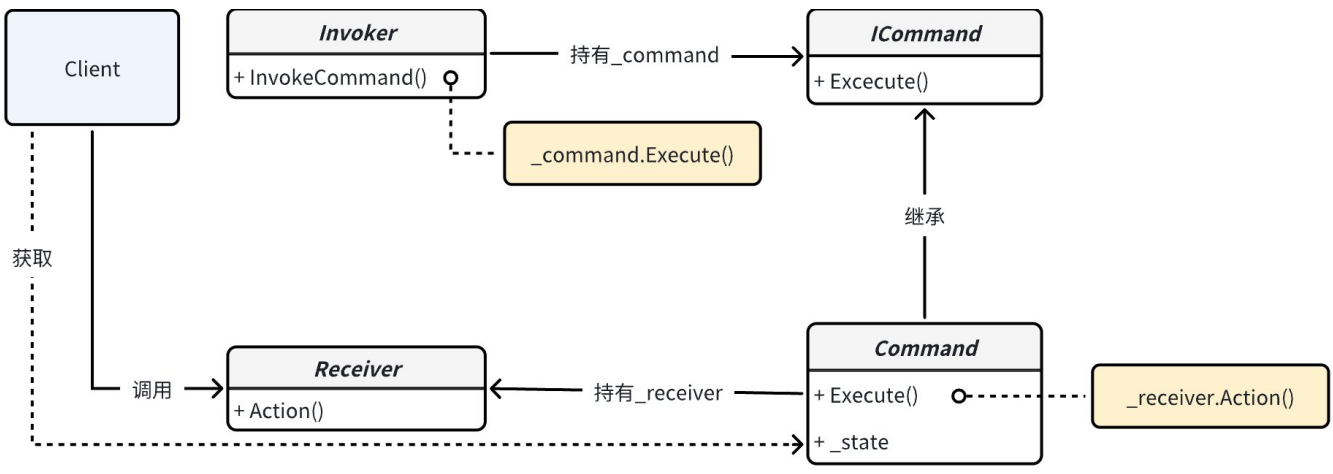
命令模式

概念

- 命令模式（Command Pattern）是一种行为设计模式，它将请求封装成对象，从而使用户可以用不同的请求、队列或者日志来参数化其他对象。命令模式也支持可撤销的操作。
- 应用场景：
  - 需要参数化对象的操作：例如，菜单项的点击操作。
  - 需要将操作放入队列中执行：例如，任务调度系统。
  - 需要支持撤销操作：例如，文本编辑器中的撤销和重做功能。
- 构成：
  - 命令接口（Command）：声明执行操作的接口。
  - 接收者（Receiver）：执行命令相关的操作。
  - 具体命令类（ConcreteCommand）：持有接收者，实现命令接口，定义具体的操作。
  - 调用者（Invoker）：持有命令对象，并在某个时间点调用命令对象的执行方法。



实例

- 命令接口（Command）

```
public interface ICommand
{
    void Execute();
    void Undo();
```

```
}
```

- 接收者（Receiver）

```
public class Light
{
    public void TurnOn()
    {
        Console.WriteLine("The light is on");
    }

    public void TurnOff()
    {
        Console.WriteLine("The light is off");
    }
}
```

- 具体命令类（ConcreteCommand）

```
public class LightOnCommand : ICommand
{
    private Light _light;

    public LightOnCommand(Light light)
    {
        _light = light;
    }

    public void Execute()
    {
        _light.TurnOn();
    }

    public void Undo()
    {
        _light.TurnOff();
    }
}
```

- 调用者（Invoker）

```
public class RemoteControl
{
    private ICommand _command;

    public void SetCommand(ICommand command)
    {
        _command = command;
    }

    public void PressButton()
    {
        _command.Execute();
    }

    public void PressUndo()
    {
        _command.Undo();
    }
}
```

- 客户端

```
static void Main(string[] args)
{
    Light livingRoomLight = new Light();
    ICommand lightOn = new LightOnCommand(livingRoomLight);

    RemoteControl remote = new RemoteControl();
    remote.SetCommand(lightOn);
    remote.PressButton(); // 输出: The light is on
    remote.PressUndo();   // 输出: The light is off
}
```