LECTURE 10

Mediator and **Command**

Software Development Principle 115~135

Mediator(中介模式)

Intent

Define an object that encapsulates how a set of objects interact. Mediator promotes loose coupling by keeping objects from referring to each other explicitly, and it lets you vary their interaction independently.

Motivation

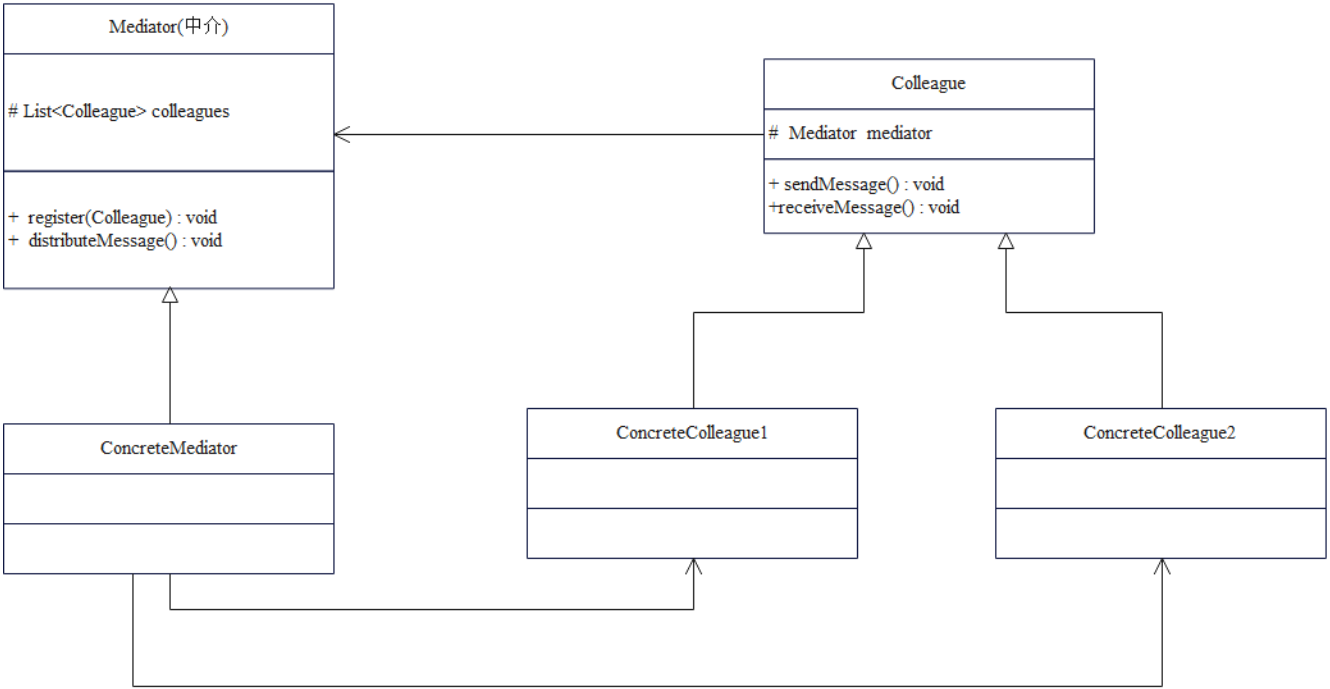
Object-oriented design encourages the distribution of behavior among objects. Such distribution can result in an object structure with many connections between objects; in the worst case, every object ends up knowing about every other.

Applicability

Use the Mediator pattern when

a. a set of objects communicate in well-defined but complex ways. The resulting interdependencies are unstructured and difficult to understand.
b. reusing an object is difficult because it refers to and communicates with many other objects.
c. a behavior that's distributed between several classes should be customizable without a lot of subclassing.

* Class Diagram



Participants

Mediator

- defines an interface for communicating with Colleague objects.

ConcreteMediator

- implements cooperative behavior by coordinating Colleague objects.
- knows and maintains its colleagues.

Colleague classes

- each Colleague class knows its Mediator object.
- each colleague communicates with its mediator whenever it would have otherwise communicated with another colleague.

Collaborations

- Colleagues send and receive requests from a Mediator object. The mediator implements the cooperative behavior by routing requests between the appropriate colleague(s).

Consequences

The Mediator pattern has the following benefits and drawbacks:

a. It limits subclassing.

b. It decouples colleagues.

c. It simplifies object protocols.

d. It abstracts how objects cooperate.

e. It centralizes control.

Implementation

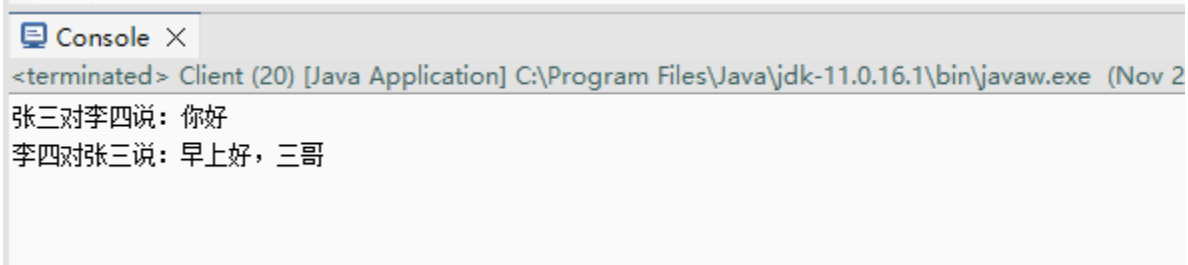
The following implementation issues are relevant to the Mediator pattern:

1. Omitting the abstract Mediator class. There's no need to define an abstract Mediator class when colleagues work with only one mediator.
2. Colleagues have to communicate with their mediator when an event of interest occurs.

People.java | Client.java | User.java | ChatRoom.java | Client.java | ChatRoom.java | User.java | PublicChatRoom.java

```java
package human;

public class People {
    private String name;
    private People other;

    public People(String name) {
        this.name = name;
    }

    public String getName() {
        return this.name;
    }

    public void connect(People other) {
        this.other = other;
    }

    public void talk(String msg) {
        other.listen(msg);
    }

    public void listen(String msg) {
        System.out.println(
                other.getName()+"对"+this.name+"说: "+msg
                );
    }

}
```

```java
package human;

public class Client {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        People p3 = new People("张三");
        People p4 = new People("李四");
        p3.connect(p4);
        p4.connect(p3);
        p3.talk("你好");
        p4.talk("早上好，三哥");
    }

}
```



```java
package interactivePlatform;

public class User {
    private String name;
    private ChatRoom chatRoom;

    public User(String name) {
        this.name = name;
    }
    public String getName() {
        return this.name;
    }
    public void login(ChatRoom chatRoom) {
        this.chatRoom = chatRoom;
        this.chatRoom.register(this);
    }

    public void talk(String msg) {
        chatRoom.sendMsg(this,msg);
    }
    public void listen(User formWhom, String msg) {
        System.out.print("【"+this.name+"的对话框】");
        System.out.println(formWhom.getName()+"说: "+msg);
    }
}
```

```java
package interactivePlatform;

import java.util.ArrayList;
import java.util.List;

public class ChatRoom {
    private String name;

    public ChatRoom(String name) {
        this.name = name;
    }
    List<User> users = new ArrayList<>();
    public void register(User user) {
        this.users.add(user);
        System.out.print("系统消息：欢迎【");
        System.out.print(user.getName());
        System.out.println("】加入聊天室【"+this.name+"】");
    }
    public void sendMsg(User fromWhom, String msg){
        users.stream().forEach(toWhom->toWhom.listen(fromWhom, msg));
    }
}
```

```java
package interactivePlatform;

public class Client {
    public static void main(String args[]) {
        ChatRoom chatRoom = new ChatRoom("设计模式：聊天室");
        User user3 = new User("张三");
        User user4 = new User("李四");
        User user5 = new User("王五");
        user3.login(chatRoom);
        user4.login(chatRoom);
        user4.talk("大家好");
        user3.talk("你好");
        user5.login(chatRoom);
        user5.talk("大家好，我来了");
    }
}
```

Console

<terminated> Client (21) [Java Application] C:\Program Files\Java\jdk-11.0.16.1\bin\javaw.exe (Nov 2, 2022, 5:

```
系统消息：欢迎【张三】加入聊天室【设计模式：聊天室】
系统消息：欢迎【李四】加入聊天室【设计模式：聊天室】
【张三的对话框】李四说：大家好
【李四的对话框】李四说：大家好
【张三的对话框】张三说：你好
【李四的对话框】张三说：你好
系统消息：欢迎【王五】加入聊天室【设计模式：聊天室】
【张三的对话框】王五说：大家好，我来了
【李四的对话框】王五说：大家好，我来了
【王五的对话框】王五说：大家好，我来了
```

Command (命令模式)

Intent

Encapsulate a request as an object, thereby letting you parameterize clients with different requests, queue or log requests, and support undoable operations.

Also known as

Action, Transaction

Motivation

Sometimes it's necessary to issue requests to objects without knowing anything about the operation being requested or the receiver of the request.

Applicability

Use the Command pattern when you want to

parameterize objects by an action to perform, as MenuItem objects did above.

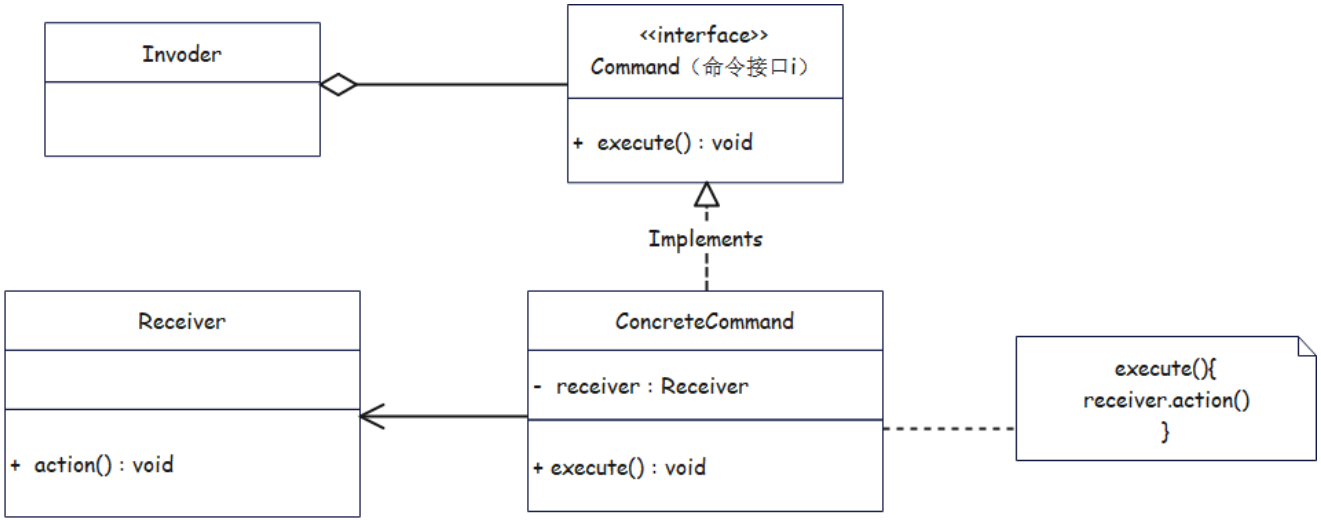
specify, queue, and execute requests at different times.

support undo. The Command's Execute operation can store state for reversing its effects in the command itself.

support logging changes so that they can be reapplied in case of a system crash.

structure a system around high-level operations built on primitives operations.

* Class Diagram



Participants

Command

- declares an interface for executing an operation.

ConcreteCommand

- defines a binding between a Receiver object and an action.
- implements Execute by invoking the corresponding operation(s) on Receiver.

Client

- creates a ConcreteCommand object and sets its receiver.

Participants

Invoker

- asks the command to carry out the request.

CareTaker

- knows how to perform the operations associated with carrying out a
- request. Any class may serve as a Receiver.

Collaborations

- The client creates a ConcreteCommand object and specifies its receiver.
- An Invoker object stores the ConcreteCommand object.
- The invoker issues a request by calling Execute on the command. When commands are undoable, ConcreteCommand stores state for undoing the command prior to invoking Execute.
- The ConcreteCommand object invokes operations on its receiver to carry out the request.

Consequences

The Command pattern has the following consequences:

- Command decouples the object that invokes the operation from the one that knows how to perform it.
- Commands are first-class objects. They can be manipulated and extended like any other object.
- You can assemble commands into a composite command. An example is the MacroCommand class described earlier. In general, composite commands are an instance of the Composite (183) pattern.
- It's easy to add new Commands, because you don't have to change existing classes

Implementation

Consider the following issues when implementing the Command pattern:

- How intelligent should a command be?
- Supporting undo and redo.
- Avoiding error accumulation in the undo process.

```java
package bulb;

public class Bulb {
    public void on() {
        System.out.println("开灯");
    }
    public void off() {
        System.out.println("关灯");
    }
}
```

```java
package bulb;

public class Switcher {
    private Bulb bulb;
    public Switcher(Bulb bulb) {
        this.bulb = bulb;
    }
    public void buttonPush() {
        System.out.println("按下按钮……");
        bulb.on();
    }
    public void buttonPop() {
        System.out.println("弹起按钮……");
        bulb.off();
    }
}
```

```java
package bulb;

public class Client {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Switcher switcher = new Switcher(new Bulb());
        switcher.buttonPush();
        switcher.buttonPop();
    }

}
```

Console

<terminated> Client (22) [Java Application] C:\Program Files\Java\jdk-11.0.16.

```
按下按钮.....
开灯
弹起按钮.....
关灯
```

```java
package commandBulb;

public interface Command {
    void exe();
    void unexe();
}
```

```java
public class Switcher {
    private Command command;
    public void setCommand(Command command) {
        this.command = command;
    }
    public void buttonPush() {
        System.out.println("按下按钮.....");
        command.exe();
    }
    public void buttonPop() {
        System.out.println("弹起按钮.....");
        command.unexe();
    }
}
```

```java
package commandBulb;

import bulb.Bulb;

public class SwitchCommand implements Command {
    private Bulb bulb;

    public SwitchCommand(Bulb bulb) {
        this.bulb = bulb;
    }
    @Override
    public void exe() {
        // TODO Auto-generated method stub
        bulb.on();
    }

    @Override
    public void unexe() {
        // TODO Auto-generated method stub
        bulb.off();
    }

}
```

```java
package commandBulb;

import bulb.Bulb;

public class Client {
    public static void main(String[] args) {
        Switcher switcher = new Switcher();
        Bulb bulb = new Bulb();
        Command switchCommand = new SwitchCommand(bulb);

        switcher.setCommand(switchCommand);
        switcher.buttonPush();
        switcher.buttonPop();
    }
}
```

Console

<terminated> Client (23) [Java Application] C:\Program Files\Java\jdk-11.0.16.1\bin\javaw.exe (No

```
按下按钮.....
开灯
弹起按钮.....
关灯
```

```java
import bulb.Bulb;

public class FlashCommand implements Command{
    private Bulb bulb;
    private volatile boolean neonRun = false;//volatile 用来保持可见性

    public FlashCommand(Bulb bulb) {
        this.bulb = bulb;
    }

    @Override
    public void exe() {
        // TODO Auto-generated method stub
        if(!neonRun) {
            neonRun = true;
            System.out.println("霓虹灯闪烁任务启动");
            new Thread(()->{//判断开启线程
                try {
                    while(neonRun) {
                        bulb.on();
                        Thread.sleep(500);
                        bulb.off();
                        Thread.sleep(500);
                    }
                }catch(InterruptedException e) {
                    e.printStackTrace();
                }
            }).start();
        }
    }

    @Override
    public void unexe() {
        // TODO Auto-generated method stub
        neonRun = false;
        System.out.println("霓虹灯停止闪烁");
    }


}
```

```java
package neonLights;

import bulb.Bulb;

public class Client {

    public static void main(String[] args) throws InterruptedException {
        // TODO Auto-generated method stub
        Switcher switcher = new Switcher();
        Bulb bulb = new Bulb();
        Command flashCommand = new FlashCommand(bulb);
        switcher.setCommand(flashCommand);
        switcher.buttonPush();
        Thread.sleep(3000);
        switcher.buttonPop();
    }

}
```

Console

<terminated> Client (24) [Java Application] C:\Program Files\Java\jdk-11.0.16.1\bin\javaw.exe (Nov 3, 2022, 12:15:51 AM – 12:15

```
按下按钮.....
霓虹灯闪烁任务启动
开灯
关灯
开灯
关灯
开灯
关灯
弹起按钮.....
霓虹灯停止闪烁
```

```java
public class TV {
    public void on() {
        System.out.println("电视机开启");
    }
    public void off() {
        System.out.println("电视机关机");
    }
    public void channelUp() {
        System.out.println("电视频道+");
    }
    public void channelDown() {
        System.out.println("电视频道-");
    }
    public void volumUp() {
        System.out.println("电视音量+");
    }
    public void volumDown() {
        System.out.println("电视音量-");
    }
}
```

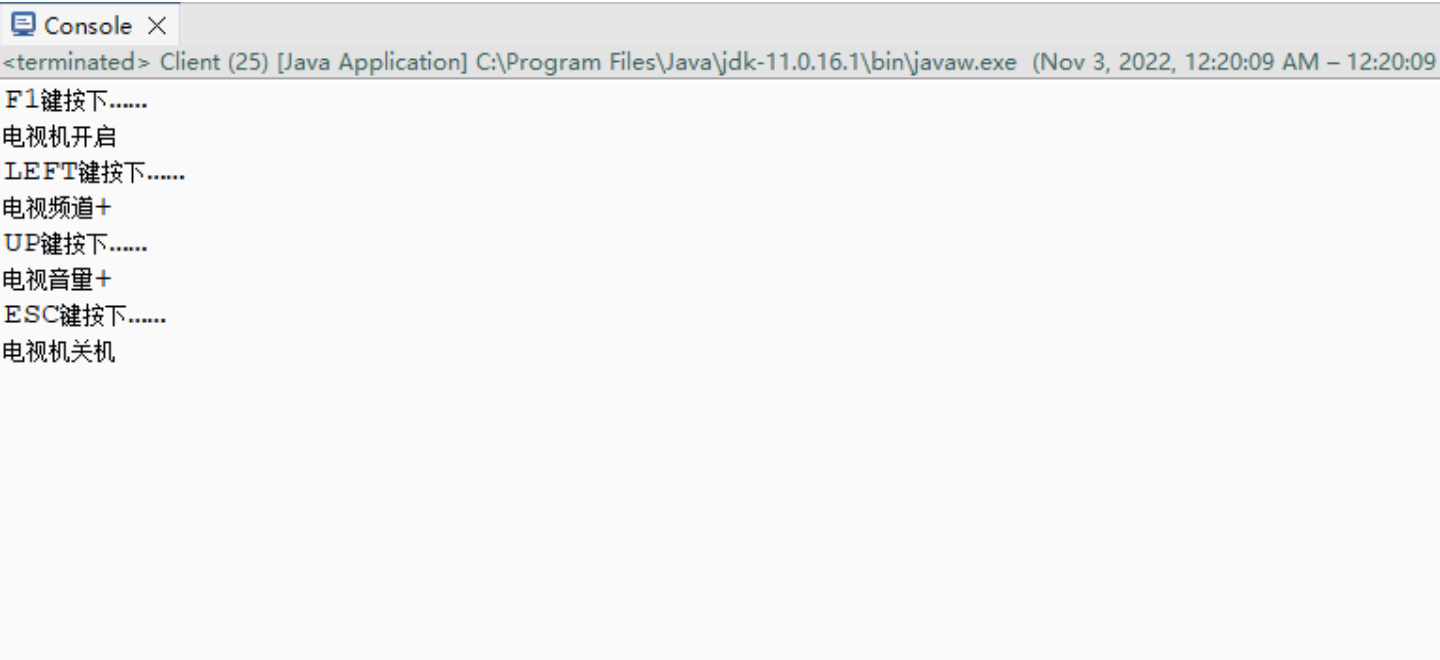
```java
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import commandBulb.Command;

public class Keyboard {
    public enum KeyCode{
        F1, F2, ESC, UP, DOWN, LEFT, RIGHT;
    }
    private Map<KeyCode,List<Command>> keyCommands = new HashMap<>();
    public void bindKeyCommand(KeyCode keyCode, List<Command> commands) {
        this.keyCommands.put(keyCode, commands);
    }
    public void onKeyPressed(KeyCode keyCode) {
        System.out.println(keyCode + "键按下……");
        List<Command> commands = this.keyCommands.get(keyCode);
        if(commands == null) {
            System.out.println("警告：无效的命令。");
            return;
        }
        commands.stream().forEach(command -> command.exe());
    }
}
```

```java
import commandBulb.Command;

public class TVChannelDownCommand implements Command {
    private TV tv;

    public TVChannelDownCommand(TV tv) {
        this.tv = tv;
    }

    @Override
    public void exe() {
        // TODO Auto-generated method stub
        tv.channelDown();
    }

    @Override
    public void unexe() {
        // TODO Auto-generated method stub
        tv.channelUp();
    }
}
```

```java
import commandBulb.Command;

public class TVChannelUpCommand implements Command{
    private TV tv;
    public TVChannelUpCommand(TV tv) {
        this.tv = tv;
    }

    @Override
    public void exe() {
        // TODO Auto-generated method stub
        tv.channelUp();
    }

    @Override
    public void unexe() {
        // TODO Auto-generated method stub
        tv.channelDown();
    }

}
```

```java
import commandBulb.Command;

public class TVoffCommand implements Command {
    private TV tv;
    public TVoffCommand(TV tv) {
        this.tv = tv;
    }
    @Override
    public void exe() {
        // TODO Auto-generated method stub
        tv.off();
    }
    @Override
    public void unexe() {
        // TODO Auto-generated method stub
        tv.on();
    }
}
```

```java
import java.util.Arrays;

import TVCommand.Keyboard.KeyCode;
import commandBulb.Command;

public class Client {
    public static void main(String[] args) {
        Keyboard keyboard = new Keyboard();
        TV tv = new TV();
        Command tvOnCommand = new TVonCommand(tv);
        Command tvOffCommand = new TVoffCommand(tv);
        Command tvChannelUpCommand = new TVChannelUpCommand(tv);
        Command tvChannelDownCommand = new TVChannelDownCommand(tv);
        Command tvVolumUp = new TVVolumUp(tv);
        Command tvVolumDown = new TVVolumDown(tv);

        keyboard.bindKeyCommand(Keyboard.KeyCode.F1, Arrays.asList(tvOnCommand));
        keyboard.bindKeyCommand(Keyboard.KeyCode.ESC, Arrays.asList(tvOffCommand));
        keyboard.bindKeyCommand(Keyboard.KeyCode.LEFT,Arrays.asList(tvChannelUpCommand));
        keyboard.bindKeyCommand(Keyboard.KeyCode.RIGHT, Arrays.asList(tvChannelDownCommand));
        keyboard.bindKeyCommand(Keyboard.KeyCode.UP, Arrays.asList(tvVolumUp));
        keyboard.bindKeyCommand(Keyboard.KeyCode.DOWN, Arrays.asList(tvVolumDown));

        keyboard.onKeyPressed(Keyboard.KeyCode.F1);
        keyboard.onKeyPressed(Keyboard.KeyCode.LEFT);
        keyboard.onKeyPressed(Keyboard.KeyCode.UP);
        keyboard.onKeyPressed(Keyboard.KeyCode.ESC);
    }
}
```


Console
<terminated> Client (25) [Java Application] C:\Program Files\Java\jdk-11.0.16.1\bin\javaw.exe (Nov 3, 2022, 12:20:09 AM – 12:20:09
F1键按下.....
电视机开启
LEFT键按下.....
电视频道+
UP键按下.....
电视音量+
ESC键按下.....
电视机关机

```java
import java.util.Arrays;

import bulb.Bulb;
import commandBulb.Command;

public class CommandClient {
    public static void main(String[] args) {
        Keyboard keyborad = new Keyboard();
        TV tv = new TV();
        Bulb bulb = new Bulb();
        Command tvOnCommand = new TVonCommand(tv);
        Command tvOffCommand = new TVoffCommand(tv);
        Command tvChannelUpCommand = new TVChannelUpCommand(tv);
        Command tvChannelDownCommand = new TVChannelDownCommand(tv);
        Command tvVolumUp = new TVVolumUp(tv);
        Command tvVolumDown = new TVVolumDown(tv);
        keyborad.bindKeyCommand(
                Keyboard.KeyCode.F2,
                Arrays.asList(
                        //bulbOnCommand,//将开灯命令也加入按键命令映射
                        tvOnCommand,
                        tvChannelUpCommand,
                        tvChannelUpCommand,
                        tvChannelUpCommand
                        )
                );
        keyborad.onKeyPressed(Keyboard.KeyCode.F2);
    }
}
```

Console

```
<terminated> CommandClient [Java Application] C:\Program Files\Java\jdk-11.0.16.1\bin\javaw.exe  (Nov 3, 2022, 12:21:08 AM – 12:21:09 AM) [pid: 23456]
F2键按下.....
电视机开启
电视频道+
电视频道+
电视频道+
```