命令模式实验

**【实验目的及要求】**

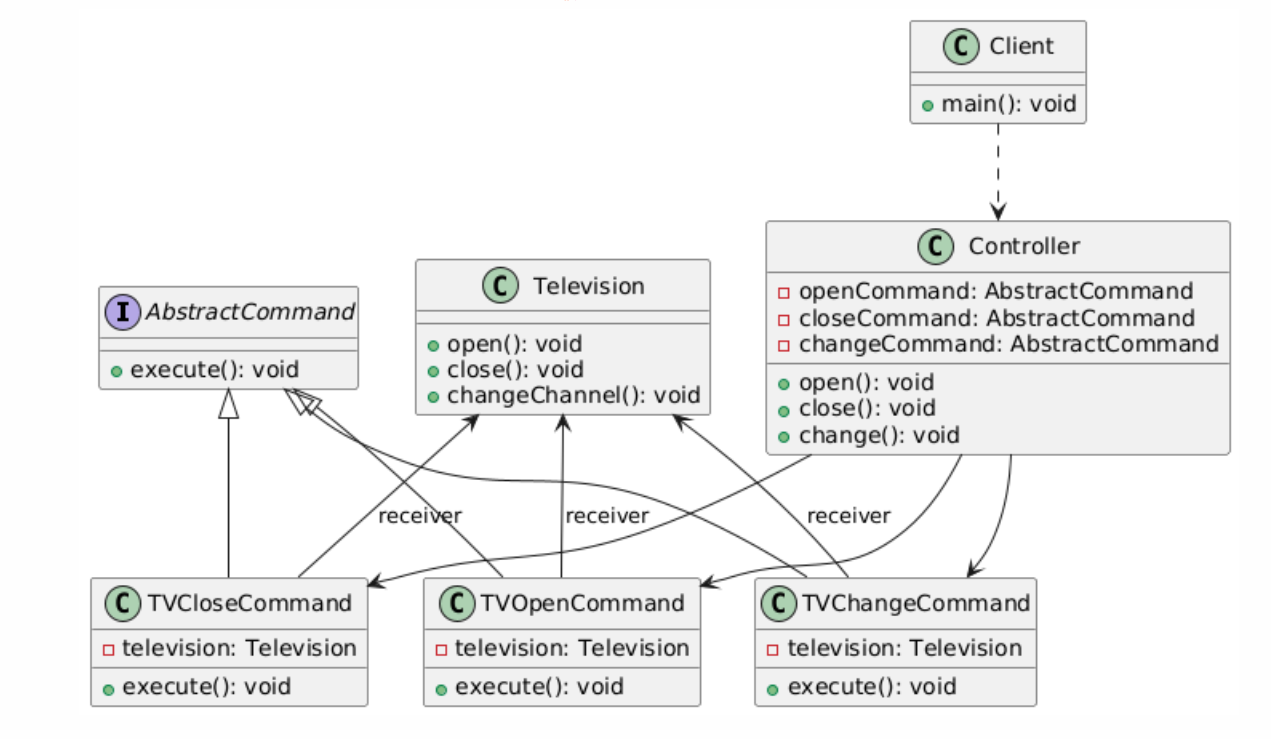
1. 熟悉命令模式的定义；
2. 掌握命令模式角色及作用，熟悉建造者模式结构图；
3. 掌握命令者模式编程。

**【实验内容】**

**假设电视遥控器有三个按钮分别代表三种操作：打开电视机、关闭电视机和切换频道，通过遥控器对电视机进行控制操作。电视机是请求的接收者，遥控器是请求的发送者，一个具体按钮对应一个具体命令类。请使用命令模式实现上述描述。**

**【实验步骤】**

1. **类图**

****

1. **实现代码**
2. **接受者类（Television）**

**package** com.command;

**public** **class** Television {

**public** **void** open() {

System.***out***.println("打开电视机！");

}

**public** **void** close() {

System.***out***.println("关闭电视机！");

}

**public** **void** changeChannel() {

System.***out***.println("切换电视频道！");

}

}

1. **抽象命令类（AbstractCommand）**

**package** com.command;

**public** **interface** AbstractCommand {

**public** **void** execute();

}

1. **具体命令类**

**TVOpenCommand类，负责电视机打开命令**

**public class TVOpenCommand implements AbstractCommand{  
 private Television television; //维持对接收者的引用  
  
 public TVOpenCommand() {  
 this.television = new Television();  
 }  
  
 @Override  
 public void execute() {  
 television.open();  
 }  
}**

**TVCloseCommand类，负责电视机关闭命令**

**public class TVCloseCommand implements AbstractCommand{  
 private Television television; //维持对接收者的引用  
  
 public TVCloseCommand() {  
 this.television = new Television();  
 }  
  
 @Override  
 public void execute() {  
 television.close();  
 }  
}**

**TVChangeCommand类，负责电视机频道切换命令**

**//具体命令类  
public class TVChangeCommand implements AbstractCommand{  
 private Television television; //维持对接收者的引用  
  
 public TVChangeCommand() {  
 this.television = new Television();  
 }  
  
 @Override  
 public void execute() {  
 television.changeChannel();  
 }  
}**

1. **调用者类（Controller遥控器类）**

**package** com.command;

**public** **class** Controller {

**private** AbstractCommand openCommand,closeCommand,changeCommand;

**public** Controller(AbstractCommand openCommand, AbstractCommand closeCommand, AbstractCommand changeCommand) {

**this**.openCommand = openCommand;

**this**.closeCommand = closeCommand;

**this**.changeCommand = changeCommand;

}

//打开电视

public void open() {

openCommand.execute();

}

//切换电视频道

public void close() {

closeCommand.execute();

}

//关闭电视

public void change() {

changeCommand.execute();

}}

1. **客户端测试类Client**

**package** com.command;

**public** **class** Client {

**public** **static** **void** main(String[] args) {

AbstractCommand openCommand,closeCommand,changeCommand;

openCommand = **new** TVOpenCommand();

closeCommand = **new** TVCloseCommand();

changeCommand = **new** TVChangeCommand();

Controller control = **new** Controller(openCommand,closeCommand,changeCommand);

//打开电视

openCommand.execute();

//切换电视频道

closeCommand.execute();

//关闭电视

changeCommand.execute(); }

}

**【思考】**

如果需要使用该遥控器来控制空调（AirConditioner），分别对应三个命令是打开空调（ACOpenCommand），关闭空调（ACCloseCommand）和改变空调温度（ACChangeCommand）。在上述实验基础上如何进一步完善？

public class AirConditioner {  
 public void open() {  
 System.*out*.println("打开空调！");  
 }  
 public void close() {  
 System.*out*.println("关闭空调！");  
 }  
 public void changeTemperature() {  
 System.*out*.println("改变空调温度！");  
 }  
}

public class ACChangeCommand implements AbstractCommand{  
 private AirConditioner airConditioner; //维持唯一接收者  
  
 public ACChangeCommand() {  
 this.airConditioner = new AirConditioner();  
 }  
  
 @Override  
 public void execute() {  
 airConditioner.changeTemperature();  
 }  
}

public class ACCloseCommand implements AbstractCommand{  
 private AirConditioner airConditioner; //维持唯一接收者  
  
 public ACCloseCommand() {  
 this.airConditioner = new AirConditioner();  
 }  
  
 @Override  
 public void execute() {  
 airConditioner.close();  
 }  
}

public class ACOpenCommand implements AbstractCommand{  
 private AirConditioner airConditioner; //维持唯一接收者  
  
 public ACOpenCommand() {  
 this.airConditioner = new AirConditioner();  
 }  
  
 @Override  
 public void execute() {  
 airConditioner.open();  
 }  
}

public class Client {  
 public static void main(String[] args) {  
 AbstractCommand openCommand,closeCommand,changeCommand;  
 AbstractCommand aCChangeCommand,aCCloseCommand,aCOpenCommand;  
 openCommand = new TVOpenCommand();  
 closeCommand = new TVCloseCommand();  
 changeCommand = new TVChangeCommand();  
 Controller control = new Controller(openCommand,closeCommand,changeCommand);  
 //打开电视  
 openCommand.execute();  
  
 //切换电视频道  
 closeCommand.execute();  
  
 //关闭电视  
 changeCommand.execute();  
  
 System.*out*.println("-----------------------------------------");  
  
 aCOpenCommand = new TVOpenCommand();  
 aCCloseCommand = new TVCloseCommand();  
 aCChangeCommand = new ACChangeCommand();  
  
// 打开空调  
 aCOpenCommand.execute();  
// 关闭空调  
 aCCloseCommand.execute();  
// 改变温度  
 aCChangeCommand.execute();  
  
  
 }  
}

**【总结】**

|  |  |
| --- | --- |
| 实验总结  （不少于100字） | 在本次实验中，我深入学习了命令模式的定义、角色及其作用，并熟练掌握了命令模式的编程实现。命令模式将请求的发送者和接收者解耦，使得请求的调用者不需要知道具体的处理逻辑，从而简化了代码的维护和扩展。此外，我还学习了命令模式的相关角色，包括命令（Command）、接收者（Receiver）、调用者（Invoker）和客户端（Client）。在实验过程中，我通过具体编程实现了命令模式的应用，进一步理解了如何利用命令模式来处理不同操作的封装和执行。通过该实验，我对命令模式的实际应用场景有了更深刻的理解，并能够灵活运用在实际开发中。 |
| 学号 | 22154A206 |
| 姓名 | 冯艳芳 |
| 成绩 |  |