**Exercise 9: Command Pattern – Home Automation System**

**Step 1: Command Interface – Command.java**

public interface Command {

void execute();

}

**Step 2: Receiver Class – Light.java**

public class Light {

public void turnOn() {

System.out.println("The light is ON");

}

public void turnOff() {

System.out.println("The light is OFF");

}

}

**Step 3: Concrete Command Classes**

**// File: LightOnCommand.java**

public class LightOnCommand implements Command {

private Light light;

public LightOnCommand(Light light) {

this.light = light;

}

public void execute() {

light.turnOn();

}

}

**// File: LightOffCommand.java**

public class LightOffCommand implements Command {

private Light light;

public LightOffCommand(Light light) {

this.light = light;

}

public void execute() {

light.turnOff();

}

}

**Step 4: Invoker Class – RemoteControl.java**

public class RemoteControl {

private Command command;

// set command dynamically

public void setCommand(Command command) {

this.command = command;

}

public void pressButton() {

if (command != null) {

command.execute();

} else {

System.out.println("No command assigned!");

}

}

}

**Step 5: Test Class – TestCommandPattern.java**

public class TestCommandPattern {

public static void main(String[] args) {

Light bedroomLight = new Light();

Command lightOn = new LightOnCommand(bedroomLight);

Command lightOff = new LightOffCommand(bedroomLight);

RemoteControl remote = new RemoteControl();

// Turn on the light

remote.setCommand(lightOn);

remote.pressButton();

// Turn off the light

remote.setCommand(lightOff);

remote.pressButton();

}

}

**OUTPUT:**

