**Exercise 9: Implementing the Command Pattern**

**Scenario:** You are developing a home automation system where commands can be issued to turn devices on or off. Use the Command Pattern to achieve this.

**COMMAND PATTERN:**

The Command Design Pattern is behavioral design pattern that turns a request into a stand-alone object called a command. With the help of this pattern, you can capture each component of a request, including the object that owns the method, the parameters for the method, and the method itself. By doing this, you can easily pass, queue, or log requests and support operations like undo/redo.

**Steps:**

1. **Create a New Java Project:**
   * Create a new Java project named **CommandPatternExample**.
2. **Define Command Interface:**
   * Create an interface Command with a method **execute()**.
3. **Implement Concrete Commands:**
   * Create classes **LightOnCommand**, **LightOffCommand** that implement Command.
4. **Implement Invoker Class:**
   * Create a class **RemoteControl** that holds a reference to a Command and a method to execute the command.
5. **Implement Receiver Class:**
   * Create a class **Light** with methods to turn on and off.
6. **Test the Command Implementation:**
   * Create a test class to demonstrate issuing commands using the **RemoteControl**

**1. Command Interface**

public interface Command {

void execute();

}

**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");

}

}

**3. Concrete Command Classes**

// LightOnCommand.java

public class LightOnCommand implements Command {

private Light light;

public LightOnCommand(Light light) {

this.light = light;

}

@Override

public void execute() {

light.turnOn();

}

}

// LightOffCommand.java

public class LightOffCommand implements Command {

private Light light;

public LightOffCommand(Light light) {

this.light = light;

}

@Override

public void execute() {

light.turnOff();

}

}

**4. Invoker Class**

// RemoteControl.java

public class RemoteControl {

private Command command;

public void setCommand(Command command) {

this.command = command;

}

public void pressButton() {

command.execute();

}

}

**5. Test Class**

// CommandPatternTest.java

public class CommandPatternTest {

public static void main(String[] args) {

// Receiver

Light light = new Light();

// Commands

Command lightOn = new LightOnCommand(light);

Command lightOff = new LightOffCommand(light);

// Invoker

RemoteControl remote = new RemoteControl();

//To turn light on

remote.setCommand(lightOn);

remote.pressButton();

// To turn light off

remote.setCommand(lightOff);

remote.pressButton();

}

}

**To run this :**

javac \*.java # Compile all files

java CommandPatternTest # Run the file

**Output:**

The light is ON

The light is OFF