## ****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.

### ****Step 1: Create a New Java Project****

Create a new Java project named CommandPatternExample.

### ****Step 2: Define Command Interface****

public interface Command {

void execute();

}

### ****Step 3: Implement Concrete Commands****

public class LightOnCommand implements Command {

private Light light;

public LightOnCommand(Light light) {

this.light = light;

}

@Override

public void execute() {

light.turnOn();

}

}

public class LightOffCommand implements Command {

private Light light;

public LightOffCommand(Light light) {

this.light = light;

}

@Override

public void execute() {

light.turnOff();

}

}

### ****Step 4: Implement Invoker Class****

public class RemoteControl {

private Command command;

public void setCommand(Command command) {

this.command = command;

}

public void pressButton() {

if (command != null) {

command.execute();

} else {

System.out.println("No command set.");

}

}

}

### ****Step 5: Implement Receiver Class****

public class Light {

public void turnOn() {

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

}

public void turnOff() {

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

}

}

### ****Step 6: Test the Command Implementation****

public class TestCommandPattern {

public static void main(String[] args) {

Light livingRoomLight = new Light();

Command lightOn = new LightOnCommand(livingRoomLight);

Command lightOff = new LightOffCommand(livingRoomLight);

RemoteControl remote = new RemoteControl();

remote.setCommand(lightOn);

remote.pressButton();

remote.setCommand(lightOff);

remote.pressButton();

}

}

OUTPUT:

