**Exercise 9: Implementing the Command Pattern**

**Code:**

interface Command{

    void execute();

}

class Light{

    public void turnOn(){

        System.out.println("Light is ON");

    }

    public void turnOff(){

        System.out.println("Light is OFF");

    }

}

class LightOnCommand implements Command{

    private Light light;

    public LightOnCommand(Light light){

        this.light = light;

    }

    @Override

    public void execute(){

        light.turnOn();

    }

}

class LightOffCommand implements Command {

    private Light light;

    public LightOffCommand(Light light){

        this.light = light;

    }

    @Override

    public void execute(){

        light.turnOff();

    }

}

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 commands assigned");

        }

    }

}

public class CommandPatternExample{

    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();

        remote.setCommand(null);

        remote.pressButton();

    }

}

**Output:**

