**Exercise 9: Implementing the Command Pattern**

**Code:**

**Command Interface: -**

public interface Command {  
 void execute();  
}

**Light Class: -**

public class Light {  
 public void turnOn(){  
 System.*out*.println("The light is ON.");  
 }  
 public void turnOff(){  
 System.*out*.println("The light is OFF.");  
 }  
}

**LightOnCommand Class: -**

public class LightOnCommand implements Command{  
 private Light light;  
 public LightOnCommand(Light light){  
 this.light=light;  
 }  
 @Override  
 public void execute() {  
 light.turnOn();  
 }  
}

**LightOffCommand Class: -**

public class LightOffCommand implements Command{  
 private Light light;  
 public LightOffCommand(Light light){  
 this.light=light;  
 }  
 @Override  
 public void execute(){  
 light.turnOff();  
 }  
}

**RemoteControl 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.");  
 }  
 }  
}

**TestClass (Main Class): -**

public class TestClass {  
 public static void main(String[] args){  
 Light light=new Light();  
 Command lightOn=new LightOnCommand(light);  
 Command lightOff=new LightOffCommand(light);  
 RemoteControl remote=new RemoteControl();  
  
 remote.setCommand(lightOn);  
 remote.pressButton();  
  
 remote.setCommand(lightOff);  
 remote.pressButton();  
  
 remote.setCommand(lightOn);  
 remote.pressButton();  
  
 }  
}

**Output:**

**A screenshot of a computer program

AI-generated content may be incorrect.**