**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.

**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**.

using System;

public interface ICommand

{

void Execute();

}

public class Light

{

public void TurnOn()

{

Console.WriteLine("Light is ON");

}

public void TurnOff()

{

Console.WriteLine("Light is OFF");

}

}

public class LightOnCommand : ICommand

{

private Light light;

public LightOnCommand(Light light)

{

this.light = light;

}

public void Execute()

{

light.TurnOn();

}

}

public class LightOffCommand : ICommand

{

private Light light;

public LightOffCommand(Light light)

{

this.light = light;

}

public void Execute()

{

light.TurnOff();

}

}

public class RemoteControl

{

private ICommand command;

public void SetCommand(ICommand command)

{

this.command = command;

}

public void PressButton()

{

command.Execute();

}

}

public class Program

{

public static void Main()

{

Light livingRoomLight = new Light();

ICommand lightOn = new LightOnCommand(livingRoomLight);

ICommand lightOff = new LightOffCommand(livingRoomLight);

RemoteControl remote = new RemoteControl();

remote.SetCommand(lightOn);

remote.PressButton();

remote.SetCommand(lightOff);

remote.PressButton();

}

}

