**Exercise 8: 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.

1. Create a new Java Project:

* Create a new Java Project named CommandPatternExample.
* A Java project named CommandPatternExample is created in eclipse IDE.

2. Define Command Interface:

* Create an interface Command with a method **execute()**.

public interface Command {

void execute();

}

* The Command interface defines a single method execute() to be implemented by all concrete command classes.

3. Implement Concrete Commands:

* Create classes **LightOnCommand**, **LightOffCommand** that implement Command.

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();

}

}

* The LightOnCommand and LightOffCommand classes implement the Command interface and provide specific implementations of the execute() method.

4. Implement Invoker Class:

* Create a class **RemoteControl** that holds a reference to a Command and a method to execute the command.

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.");

}

}

}

* The RemoteControl class holds a reference to a Command and has a method pressButton() to execute the command.

5. Implement Receiver Class:

* Create a class **Light** with methods to turn on and off.

public class Light {

private String location;

public Light(String location) {

this.location = location;

}

public void turnOn() {

System.out.println(location + " light is ON");

}

public void turnOff() {

System.out.println(location + " light is OFF");

}

}

* The Light class represents the receiver and has methods to turn on and off the light.

5. Test the Command Implementation:

* Create a test class to demonstrate issuing commands using the **RemoteControl**.

public class Test {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

RemoteControl remote = new RemoteControl();

Map<String, Light> lights = new HashMap<>();

while (true) {

System.out.println("Select command: 1. Light ON 2. Light OFF 3. Exit");

int choice = scanner.nextInt();

scanner.nextLine();

if (choice == 3) {

scanner.close();

System.exit(0);

}

System.out.print("Enter light location: ");

String location = scanner.nextLine();

Light light = lights.get(location);

if (light == null) {

light = new Light(location);

lights.put(location, light);

}

Command command;

switch (choice) {

case 1:

command = new LightOnCommand(light);

break;

case 2:

command = new LightOffCommand(light);

break;

default:

System.out.println("Invalid choice.");

continue;

}

remote.setCommand(command);

remote.pressButton();

}

}

}

* Compile and run the CommandPatternTest class.
* Select the command to turn the light on or off.
* Enter the location of the light.
* Observe that the appropriate command is executed for the specified light location.

Output:

