* The Command Pattern encapsulates a request as an object, allowing for the separation of sender and receiver.
* Commands can be parameterized, meaning you can create different commands with different parameters without changing the invoker(responsible for initiating command execution).
* It decouples the sender (client or invoker) from the receiver (object performing the operation), providing flexibility and extensibility.
* The pattern supports undoable(action or a series of actions that can be reversed or undone in a system) operations by storing the state or reverse commands.





*class* Light {

 *private boolean* switchedOn = *false*;

 *public void* switchLights() {
 switchedOn = !switchedOn;
 System.*out*.println("Light is now " + (switchedOn ? "ON" : "OFF"));
 }

 *public boolean* isSwitchedOn() {
 *return* switchedOn;
 }
}

*interface Command* {
 *void* turnOn();
 *void* turnOff();
}

*class* LightCommand *implements Command* {
 *private* Light light;

 *public* LightCommand(Light light) {
 *this*.light = light;
 }

 *public void* turnOn() {
 *if* (!light.isSwitchedOn()) {
 light.switchLights();
 } *else* {
 System.*out*.println("Light is already ON");
 }
 }

 *public void* turnOff() {
 *if* (light.isSwitchedOn()) {
 light.switchLights();
 } *else* {
 System.*out*.println("Light is already OFF");
 }
 }
}

*class* RemoteControl {
 *private Command* command;

 *public* RemoteControl(*Command* command) {
 *this*.command = command;
 }

 *public void* pressOnButton() {
 command.turnOn();
 }

 *public void* pressOffButton() {
 command.turnOff();
 }
}

*public class* CommandPattern {
 *public static void* main(String[] args) {
 Light light = *new* Light();
 *Command* lightCommand = *new* LightCommand(light);
 RemoteControl remoteControl = *new* RemoteControl(lightCommand);

 remoteControl.pressOnButton(); *// Light is now ON* remoteControl.pressOffButton(); *// Light is now OFF* remoteControl.pressOffButton(); *// Light is already OFF* remoteControl.pressOnButton(); *// Light is now ON* }
}

[PlantUML Web Server](https://www.plantuml.com/plantuml/uml/SyfFKj2rKt3CoKnELR1Io4ZDoSa700002)

@startuml

skinparam handwritten true

skinparam shadowing false

interface Command {

 + turnOn(): void

 + turnOff(): void

}

class Light {

 - switchedOn: boolean

 + switchLights(): void

 + isSwitchedOn(): boolean

}

class LightCommand {

 - light: Light

 + LightCommand(light: Light)

 + turnOn(): void

 + turnOff(): void

}

class RemoteControl {

 - command: Command

 + RemoteControl(command: Command)

 + pressOnButton(): void

 + pressOffButton(): void

}

class CommandPatternDemo {

 + main(args: String[]): void {static}

}

' Relationships'

Command <|.. LightCommand : implements

LightCommand o-- Light : "1" has "1"

RemoteControl o-- Command : "1" has "1"

CommandPatternDemo ..> Light : uses

CommandPatternDemo ..> LightCommand : uses

CommandPatternDemo ..> RemoteControl : uses

@enduml

