



Command Pattern

[Read](#) [Discuss](#)

Like [previous](#) articles, let us take up a design problem to understand command pattern. Suppose you are building a home automation system. There is a programmable remote which can be used to turn on and off various items in your home like lights, stereo, AC etc. It looks something like this.

You can do it with simple if-else statements like

```
if (buttonPressed == button1)
    lights.on()
```

But we need to keep in mind that turning on some devices like stereo comprises of many steps like setting cd, volume etc. Also we can reassign a button to do something else. By using simple if-else we are coding to implementation rather than interface. Also there is tight coupling.

So what we want to achieve is a design that provides loose coupling and remote control should not have much information about a particular device. The command pattern helps us do that.

Definition: The **command pattern** encapsulates a request as an object, thereby letting us parameterize other objects with different requests, queue or log requests, and support undoable operations.

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

Got It !

request by binding together a set of actions on a specific receiver. It does so by exposing just one method `execute()` that causes some actions to be invoked on the receiver.

Parameterizing other objects with different requests in our analogy means that the button used to turn on the lights can later be used to turn on stereo or maybe open the garage door.

queue or log requests, and support undoable operations means that Command's `Execute` operation can store state for reversing its effects in the Command itself. The Command may have an added `unExecute` operation that reverses the effects of a previous call to `execute`. It may also support logging changes so that they can be reapplied in case of a system crash.

Below is the Java implementation of above mentioned remote control example:

Java

```
// A simple Java program to demonstrate
// implementation of Command Pattern using
// a remote control example.

// An interface for command
interface Command
{
    public void execute();
}

// Light class and its corresponding command
// classes
class Light
{
    public void on()
    {
        System.out.println("Light is on");
    }
}
```

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

```

}
class LightOnCommand implements Command
{
    Light light;

    // The constructor is passed the light it
    // is going to control.
    public LightOnCommand(Light light)
    {
        this.light = light;
    }
    public void execute()
    {
        light.on();
    }
}
class LightOffCommand implements Command
{
    Light light;
    public LightOffCommand(Light light)
    {
        this.light = light;
    }
    public void execute()
    {
        light.off();
    }
}

// Stereo and its command classes
class Stereo
{
    public void on()
    {
        System.out.println("Stereo is on");
    }
    public void off()
    {
        System.out.println("Stereo is off");
    }
    public void setCD()
    {
        System.out.println("Stereo is set " + " ");
    }
}

```

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

```

        System.out.println("Stereo is set"+
                           " for DVD input");
    }
    public void setRadio()
    {
        System.out.println("Stereo is set" +
                           " for Radio");
    }
    public void setVolume(int volume)
    {
        // code to set the volume
        System.out.println("Stereo volume set"
                           + " to " + volume);
    }
}
class StereoOffCommand implements Command
{
    Stereo stereo;
    public StereoOffCommand(Stereo stereo)
    {
        this.stereo = stereo;
    }
    public void execute()
    {
        stereo.off();
    }
}
class StereoOnWithCDCommand implements Command
{
    Stereo stereo;
    public StereoOnWithCDCommand(Stereo stereo)
    {
        this.stereo = stereo;
    }
    public void execute()
    {
        stereo.on();
        stereo.setCD();
        stereo.setVolume(11);
    }
}

// A simple remote control with one button

```

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

```

public SimpleRemoteControl()
{
}

public void setCommand(Command command)
{
    // set the command the remote will
    // execute
    slot = command;
}

public void buttonWasPressed()
{
    slot.execute();
}
}

// Driver class
class RemoteControlTest
{
    public static void main(String[] args)
    {
        SimpleRemoteControl remote =
            new SimpleRemoteControl();

```

Trending Now DSA Data Structures Algorithms Interview Preparation Data Science T

```

        // we can change command dynamically
        remote.setCommand(new
            LightOnCommand(light));
        remote.buttonWasPressed();
        remote.setCommand(new
            StereoOnWithCDCommand(stereo));
        remote.buttonWasPressed();
        remote.setCommand(new
            StereoOffCommand(stereo));
        remote.buttonWasPressed();
    }
}

```

Output:

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

```
Stereo is set for CD input  
Stereo volume set to 11  
Stereo is off
```

Notice that the remote control doesn't know anything about turning on the stereo. That information is contained in a separate command object. This reduces the coupling between them.

Advantages:

- Makes our code extensible as we can add new commands without changing existing code.
- Reduces coupling between the invoker and receiver of a command.

Disadvantages:

- Increase in the number of classes for each individual command

Further Read – [Command Method in Python](#)**References:**

- Head First Design Patterns (book)
- <https://github.com/bethrobson/Head-First-Design-Patterns/tree/master/src/headfirst/designpatterns/command>

If This article is contributed by **Sulabh Kumar**. you like GeeksforGeeks and would like to contribute, you can also write an article and mail your article to review-team@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#)

1. Observer Pattern | Set 1 (Introduction)

2. Observer Pattern | Set 2 (Implementation)

3. Singleton Design Pattern | Implementation

4. Decorator Pattern | Set 1 (Background)

5. The Decorator Pattern | Set 2 (Introduction and Design)

6. Decorator Pattern | Set 3 (Coding the Design)

7. Strategy Pattern | Set 2 (Implementation)

8. Iterator Pattern

9. Flyweight Design Pattern

10. Singleton Design Pattern | Introduction

[Previous](#)[Next](#)

Article Contributed By :



GeeksforGeeks

Vote for difficulty

Current difficulty : [Medium](#)

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

Improved By : [fasiabbu](#)

Article Tags : [Design Pattern](#)

[Improve Article](#)[Report Issue](#)

A-143, 9th Floor, Sovereign Corporate
Tower, Sector-136, Noida, Uttar Pradesh -
201305

feedback@geeksforgeeks.org

Company

[About Us](#)
[Careers](#)
[In Media](#)
[Contact Us](#)
[Terms and Conditions](#)
[Privacy Policy](#)
[Copyright Policy](#)
[Third-Party Copyright Notices](#)
[Advertise with us](#)

Languages

[Python](#)

Explore

[Job Fair For Students](#)
[POTD: Revamped](#)
[Python Backend LIVE](#)
[Android App Development](#)
[DevOps LIVE](#)
[DSA in JavaScript](#)

Data Structures

[Array](#)

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

[SQL](#)[R Language](#)[Android Tutorial](#)[Queue](#)[Tree](#)[Graph](#)

Algorithms

[Sorting](#)[Searching](#)[Greedy](#)[Dynamic Programming](#)[Pattern Searching](#)[Recursion](#)[Backtracking](#)

Web Development

[HTML](#)[CSS](#)[JavaScript](#)[Bootstrap](#)[ReactJS](#)[AngularJS](#)[NodeJS](#)

Computer Science

[GATE CS Notes](#)[Operating Systems](#)[Computer Network](#)[Database Management System](#)[Software Engineering](#)[Digital Logic Design](#)[Engineering Maths](#)

Python

[Python Programming Examples](#)[Django Tutorial](#)[Python Projects](#)[Python Tkinter](#)[OpenCV Python Tutorial](#)[Python Interview Question](#)

Data Science & ML

[Data Science With Python](#)[Data Science For Beginner](#)[Machine Learning Tutorial](#)[Maths For Machine Learning](#)[Pandas Tutorial](#)

DevOps

[Git](#)[AWS](#)[Docker](#)[Kubernetes](#)[Azure](#)

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

Competitive Programming

Top DSA for CP
Top 50 Tree Problems
Top 50 Graph Problems
Top 50 Array Problems
Top 50 String Problems
Top 50 DP Problems
Top 15 Websites for CP

Interview Corner

Company Preparation
Preparation for SDE
Company Interview Corner
Experienced Interview
Internship Interview
Competitive Programming
Aptitude

Commerce

Accountancy
Business Studies
Microeconomics
Macroeconomics
Statistics for Economics
Indian Economic Development

SSC/ BANKING

System Design

What is System Design
Monolithic and Distributed SD
Scalability in SD
Databases in SD
High Level Design or HLD
Low Level Design or LLD
Top SD Interview Questions

GfG School

CBSE Notes for Class 8
CBSE Notes for Class 9
CBSE Notes for Class 10
CBSE Notes for Class 11
CBSE Notes for Class 12
English Grammar

UPSC

Polity Notes
Geography Notes
History Notes
Science and Technology Notes
Economics Notes
Important Topics in Ethics
UPSC Previous Year Papers

Write & Earn

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

[IBPS PO Syllabus](#)

[Write Interview Experience](#)

[IBPS Clerk Syllabus](#)

[Internships](#)

[Aptitude Questions](#)

[Video Internship](#)

[SSC CGL Practice Papers](#)

@geeksforgeeks , Some rights reserved

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).