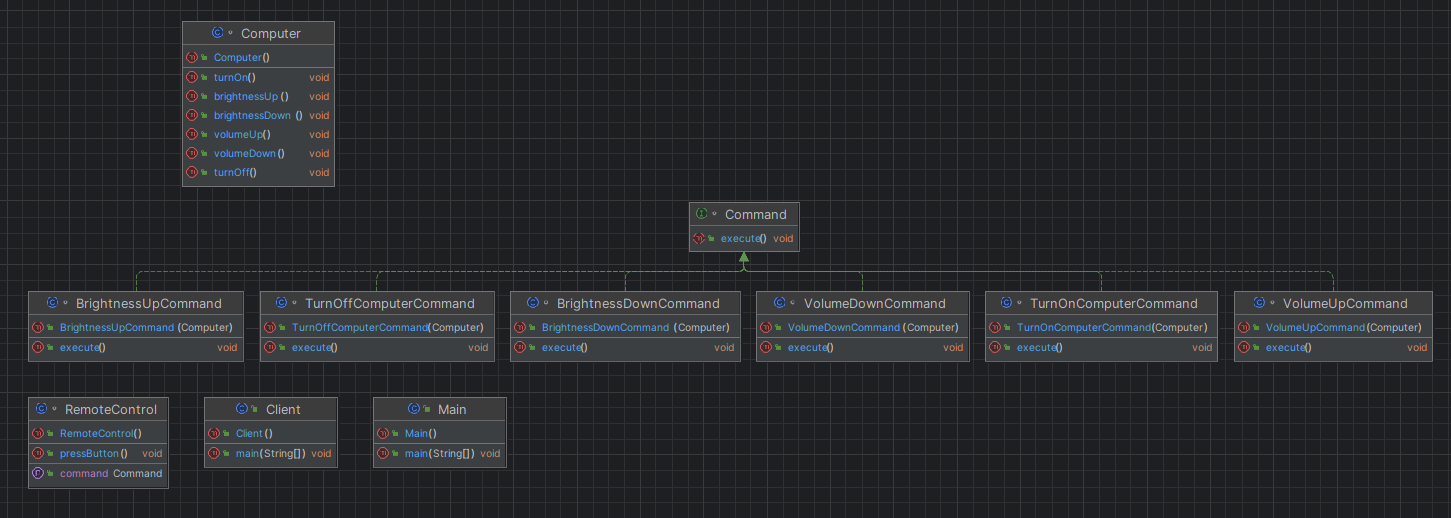
**Mẫu Command:**

**Giới thiệu:**

Đóng gói một yêu cầu như một đối tượng, giúp thực hiện, hoàn tác hoặc lưu trữ các hành động. Mẫu này thường được sử dụng trong các ứng dụng hỗ trợ undo/redo.

**Sau đây là ví dụ về mẫu Command:**



Giả sử bạn đang phát triển một ứng dụng điều khiển từ xa cho TV. Ứng dụng này có các nút để bật/tắt TV, tăng/giảm âm lượng và chuyển kênh. Mỗi nút này đại diện cho một yêu cầu.

**Cấu trúc:**

1. Interface Command:

Giao diện này định nghĩa phương thức execute(), là phương thức bắt buộc phải triển khai trong các lớp Command cụ thể. Mỗi hành động (command) được đại diện bởi một lớp implements Command.

interface Command {

void execute();

}

Mỗi lớp Command này đều có một hàm constructor nhận đối tượng Computer làm tham số và implements phương thức execute(). Bên trong phương thức execute(), các hành động cụ thể liên quan đến máy tính (ví dụ: computer.brightnessDown()) sẽ được thực thi.

BrightnessDownCommand: Giảm độ sáng màn hình.

BrightnessUpCommand: Tăng độ sáng màn hình.

TurnOffComputerCommand: Tắt máy tính.

TurnOnComputerCommand: Bật máy tính.

VolumeDownCommand: Giảm âm lượng.

VolumeUpCommand: Tăng âm lượng.

1. Class BrightnessDownCommand:

class BrightnessDownCommand implements Command {

private Computer computer;

public BrightnessDownCommand(Computer computer) {

this.computer = computer;

}

*@Override*

public void execute() {

computer.brightnessDown();

}

}

1. Class BrightnessUpCommand:

class BrightnessUpCommand implements Command {

private Computer computer;

public BrightnessUpCommand(Computer computer) {

this.computer = computer;

}

*@Override*

public void execute() {

computer.brightnessUp();

}

}

1. Class Client:

public class Client {

public static void main(String[] args) {

Computer computer = new Computer();

RemoteControl remoteControl = new RemoteControl();

// Bật máy tính

remoteControl.setCommand(new TurnOnComputerCommand(computer));

remoteControl.pressButton();

// Tăng âm lượng

remoteControl.setCommand(new VolumeUpCommand(computer));

remoteControl.pressButton();

// Tăng độ sáng màn hình

remoteControl.setCommand(new BrightnessUpCommand(computer));

remoteControl.pressButton();

// Giảm âm lượng

remoteControl.setCommand(new VolumeDownCommand(computer));

remoteControl.pressButton();

// Tắt máy tính

remoteControl.setCommand(new TurnOffComputerCommand(computer));

remoteControl.pressButton();

}

}

1. Class Computer:

Lớp này chứa các phương thức điều khiển chức năng của máy tính, chẳng hạn như: turnOn(), turnOff(), volumeUp(), volumeDown(), brightnessUp(), brightnessDown().

class Computer {

public void turnOn() {

System.***out***.println("Bật máy tính");

}

public void turnOff() {

System.***out***.println("Tắt máy tính");

}

public void volumeUp() {

System.***out***.println("Tăng âm lượng");

}

public void volumeDown() {

System.***out***.println("Giảm âm lượng");

}

public void brightnessUp() {

System.***out***.println("Tăng độ sáng màn hình");

}

public void brightnessDown() {

System.***out***.println("Giảm độ sáng màn hình");

}

}

1. Class RemoteControl:

Lớp này có thuộc tính commands lưu trữ danh sách các đối tượng Command.

Phương thức setCommand(Command command) cho phép thiết lập Command hiện tại.

Phương thức pressButton() thực hiện duyệt qua danh sách các Command và gọi phương thức execute() của từng Command để thực thi hành động. Sau đó, danh sách Command sẽ được xóa để sẵn sàng cho lần bấm nút tiếp theo.

class RemoteControl {

private List<Command> commands;

public RemoteControl() {

this.commands = new ArrayList<>();

}

public void setCommand(Command command) {

commands.add(command);

}

public void pressButton() {

for (Command command : commands) {

command.execute();

}

commands.clear();

}

}

1. Class TurnOffComputerCommand:

class TurnOffComputerCommand implements Command {

private Computer computer;

public TurnOffComputerCommand(Computer computer) {

this.computer = computer;

}

*@Override*

public void execute() {

computer.turnOff();

}

}

1. Class TurnOnComputerCommand:

class TurnOnComputerCommand implements Command {

private Computer computer;

public TurnOnComputerCommand(Computer computer) {

this.computer = computer;

}

*@Override*

public void execute() {

computer.turnOn();

}

}

1. Class VolumeDownCommand:

class VolumeDownCommand implements Command {

private Computer computer;

public VolumeDownCommand(Computer computer) {

this.computer = computer;

}

*@Override*

public void execute() {

computer.volumeDown();

}

1. }
2. Class VolumeUpCommand:

class VolumeUpCommand implements Command {

private Computer computer;

public VolumeUpCommand(Computer computer) {

this.computer = computer;

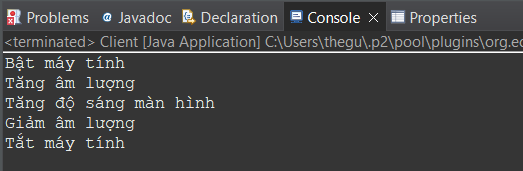
}

*@Override*

public void execute() {

computer.volumeUp();

**Kết quả và thử nghiệm:**

****

Ví dụ này chỉ là một cách đơn giản để sử dụng mẫu Command. Mẫu này có thể được sử dụng trong nhiều trường hợp khác nhau, tùy thuộc vào nhu cầu cụ thể của ứng dụng của bạn.

**Lợi ích:**

Tính linh hoạt: Bạn có thể dễ dàng thêm, xóa hoặc thay đổi các yêu cầu mà không cần sửa đổi mã ứng dụng chính.

Khả năng tái sử dụng: Các Command có thể được sử dụng lại trong các phần khác nhau của ứng dụng.

Khả năng theo dõi: Bạn có thể dễ dàng theo dõi và quản lý các yêu cầu đã được thực thi.

Khả năng mở rộng: Bạn có thể dễ dàng mở rộng ứng dụng bằng cách thêm các Command mới.