Slip 14

14.1

// Command interface with execute method

interface Command {

void execute();

}

// Receiver class - Light

class Light {

public void turnOn() {

System.out.println("Light is ON");

}

public void turnOff() {

System.out.println("Light is OFF");

}

}

// Concrete Command - LightOnCommand

class LightOnCommand implements Command {

private Light light;

public LightOnCommand(Light light) {

this.light = light;

}

@Override

public void execute() {

light.turnOn();

}

}

// Concrete Command - LightOffCommand

class LightOffCommand implements Command {

private Light light;

public LightOffCommand(Light light) {

this.light = light;

}

@Override

public void execute() {

light.turnOff();

}

}

// Receiver class - GarageDoor

class GarageDoor {

public void up() {

System.out.println("Garage door is UP");

}

}

// Concrete Command - GarageDoorUpCommand

class GarageDoorUpCommand implements Command {

private GarageDoor garageDoor;

public GarageDoorUpCommand(GarageDoor garageDoor) {

this.garageDoor = garageDoor;

}

@Override

public void execute() {

garageDoor.up();

}

}

// Receiver class - Stereo

class Stereo {

public void onWithCD() {

System.out.println("Stereo is ON with CD");

}

}

// Concrete Command - StereoOnWithCDCommand

class StereoOnWithCDCommand implements Command {

private Stereo stereo;

public StereoOnWithCDCommand(Stereo stereo) {

this.stereo = stereo;

}

@Override

public void execute() {

stereo.onWithCD();

}

}

// Invoker class

class RemoteControl {

private Command command;

public void setCommand(Command command) {

this.command = command;

}

public void pressButton() {

command.execute();

}

}

// Client code to test the Command Design Pattern

public class CommandPatternExample {

public static void main(String[] args) {

// Create instances of receivers

Light light = new Light();

GarageDoor garageDoor = new GarageDoor();

Stereo stereo = new Stereo();

// Create instances of concrete commands

Command lightOnCommand = new LightOnCommand(light);

Command lightOffCommand = new LightOffCommand(light);

Command garageDoorUpCommand = new GarageDoorUpCommand(garageDoor);

Command stereoOnWithCDCommand = new StereoOnWithCDCommand(stereo);

// Create invokers and set commands

RemoteControl remoteControl1 = new RemoteControl();

remoteControl1.setCommand(lightOnCommand);

RemoteControl remoteControl2 = new RemoteControl();

remoteControl2.setCommand(lightOffCommand);

RemoteControl remoteControl3 = new RemoteControl();

remoteControl3.setCommand(garageDoorUpCommand);

RemoteControl remoteControl4 = new RemoteControl();

remoteControl4.setCommand(stereoOnWithCDCommand);

// Press buttons to execute commands

remoteControl1.pressButton(); // Turns on the light

remoteControl2.pressButton(); // Turns off the light

remoteControl3.pressButton(); // Opens the garage door

remoteControl4.pressButton(); // Turns on the stereo with CD

}

}

14.2

#Write a python Program to find all null values in given dataset and remove them

import numpy as np

import pandas as pd

dict = {'first score':[100,90,np.nan,95], 'second score':[30,45,56,np.nan], 'third score':[np.nan,40,80,98]}

df=pd.DataFrame(dict)

print(df)

x=df.isnull()

print(x)

y=df.notnull()

print(y)

z=df.fillna(0)

print(z)

s=df.fillna(method='pad')

print(s)

a=df.fillna(method='bfill')

print(a)

b=df.replace(to\_replace=np.nan,value=-99)

print(b)

c=df.dropna()

print(c)

d=df.dropna(axis=1)

print(d)

new\_data=df.dropna(axis=0)

print(new\_data)

14.3

Html code

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Simple Web Page</title>

</head>

<body>

<h1>Hello, World!</h1>

<p>This is a simple web page served by a Node.js script.</p>

</body>

</html>

Js code

const http = require('http');

const fs = require('fs');

const path = require('path');

const server = http.createServer((req, res) => {

// Set the content type to HTML

res.writeHead(200, { 'Content-Type': 'text/html' });

// Read the HTML file and stream it to the response

const filePath = path.join(\_\_dirname, 'index.html');

const readStream = fs.createReadStream(filePath);

// Pipe the read stream to the response stream

readStream.pipe(res);

// Handle errors

readStream.on('error', (error) => {

console.error('Error reading file:', error.message);

res.writeHead(500, { 'Content-Type': 'text/plain' });

res.end('Internal Server Error');

});

});

const PORT = 3000;

server.listen(PORT, () => {

console.log(`Server running at http://localhost:${PORT}/`);

});