Slip 7

7.1

// Receiver class

class CeilingFan {

private String location;

private int speed;

public CeilingFan(String location) {

this.location = location;

this.speed = 0;

}

public void turnOn() {

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

}

public void turnOff() {

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

}

public void setSpeed(int speed) {

this.speed = speed;

System.out.println(location + " Ceiling Fan speed set to " + speed);

}

public int getSpeed() {

return speed;

}

}

// Command interface

interface CeilingFanCommand {

void execute();

void undo();

}

// Concrete Command class for changing fan speed

class CeilingFanSpeedCommand implements CeilingFanCommand {

private CeilingFan ceilingFan;

private int previousSpeed;

public CeilingFanSpeedCommand(CeilingFan ceilingFan) {

this.ceilingFan = ceilingFan;

}

@Override

public void execute() {

previousSpeed = ceilingFan.getSpeed();

ceilingFan.setSpeed(previousSpeed + 1);

}

@Override

public void undo() {

ceilingFan.setSpeed(previousSpeed);

}

}

// Invoker class

class CeilingFanRemote {

private CeilingFanCommand command;

public void setCommand(CeilingFanCommand command) {

this.command = command;

}

public void pressButton() {

command.execute();

}

public void pressUndoButton() {

command.undo();

}

}

// Client class to test Ceiling Fan with undo command

public class CeilingFanTest {

public static void main(String[] args) {

// Creating the Ceiling Fan and the corresponding Command objects

CeilingFan livingRoomCeilingFan = new CeilingFan("Living Room");

CeilingFanSpeedCommand increaseSpeedCommand = new CeilingFanSpeedCommand(livingRoomCeilingFan);

// Creating the Ceiling Fan Remote

CeilingFanRemote remote = new CeilingFanRemote();

// Setting the command for the remote control

remote.setCommand(increaseSpeedCommand);

// Pressing the button on the remote control

remote.pressButton();

// Pressing the undo button on the remote control

remote.pressUndoButton();

}

}

7.2

from sklearn import datasets

from sklearn import metrics

from sklearn.naive\_bayes import GaussianNB

dataset = datasets.load\_iris()

model=GaussianNB()

model.fit(dataset.data,dataset.target)

expected=dataset.target

predicted=model.predict(dataset.data)

print(metrics.classification\_report(expected,predicted))

print(metrics.confusion\_matrix(expected,predicted))

7.3 a

const express = require('express');

const multer = require('multer');

const path = require('path');

const app = express();

const port = 3000;

// Set up the storage engine for multer

const storage = multer.diskStorage({

destination: (req, file, cb) => {

cb(null, 'uploads/'); // Set the destination folder for uploads

},

filename: (req, file, cb) => {

const uniqueSuffix = Date.now() + '-' + Math.round(Math.random() \* 1E9);

const extension = path.extname(file.originalname);

cb(null, file.fieldname + '-' + uniqueSuffix + extension);

}

});

// Create the multer middleware

const upload = multer({ storage: storage });

// Serve the HTML form with an upload field

app.get('/', (req, res) => {

res.sendFile(\_\_dirname + '/index.html');

});

// Handle file uploads

app.post('/upload', upload.single('file'), (req, res) => {

const uploadedFile = req.file;

if (!uploadedFile) {

return res.status(400).send('No file uploaded.');

}

res.send(`File uploaded successfully: ${uploadedFile.filename}`);

});

app.listen(port, () => {

console.log(`Server is running on http://localhost:${port}`);

});

7.3 b

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>File Upload Form</title>

</head>

<body>

<h1>File Upload Form</h1>

<form action="/upload" method="post" enctype="multipart/form-data">

<label for="file">Choose a file:</label>

<input type="file" id="file" name="file" required>

<br>

<button type="submit">Upload File</button>

</form>

</body>

</html>