Slip 12

12.1

// Car interface

interface Car {

void assemble();

}

// Concrete implementation of Car

class BasicCar implements Car {

@Override

public void assemble() {

System.out.println("Basic Car");

}

}

// Decorator pattern for SportsCar

class SportsCar implements Car {

private Car car;

public SportsCar(Car car) {

this.car = car;

}

@Override

public void assemble() {

car.assemble();

System.out.println("Adding features of Sports Car");

}

}

// Decorator pattern for LuxuryCar

class LuxuryCar implements Car {

private Car car;

public LuxuryCar(Car car) {

this.car = car;

}

@Override

public void assemble() {

car.assemble();

System.out.println("Adding features of Luxury Car");

}

}

// Main class to test the Decorator Pattern

public class DecoratorPatternExample {

public static void main(String[] args) {

// Create a basic car

Car basicCar = new BasicCar();

// Decorate the basic car with SportsCar features

Car sportsCar = new SportsCar(basicCar);

// Decorate the sports car with LuxuryCar features

Car luxurySportsCar = new LuxuryCar(sportsCar);

// Assemble the final decorated car

luxurySportsCar.assemble();

}

}

12.2

#Write a python program to make categorial values in numeric format

import pandas as pd

df=pd.read\_

csv('PlayTennis.csv')

print(df)

from sklearn.preprocessing import LabelEncoder

le=LabelEncoder()

label=le.fit\_transform(df['Play Tennis'])

print(label)

df.drop("Play Tennis",axis=1, inplace=True)

df["Play Tennis"]=label

print(df

12.3

// Import the 'http' module to create an HTTP server

const http = require('http');

// Configure the HTTP server to respond with "Hello, World!" to all requests

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

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

res.end('Hello, World!\n');

});

// Listen on port 3000, and IP address defaults to 127.0.0.1

const PORT = 3000;

const IP = '127.0.0.1';

server.listen(PORT, IP, () => {

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

});