Slip 1:

Q1) Write a Java Program to implement I/O Decorator for converting uppercase letters to lower case letters.

import java.io.\*;

class LowercaseDecorator extends FilterReader {

public LowercaseDecorator(Reader in) {

super(in);

}

@Override

public int read() throws IOException {

int c = super.read();

if (c != -1) {

return Character.toLowerCase((char) c);

}

return -1;

}

@Override

public int read(char[] cbuf, int off, int len) throws IOException {

int bytesRead = super.read(cbuf, off, len);

if (bytesRead != -1) {

for (int i = off; i < off + bytesRead; i++) {

cbuf[i] = Character.toLowerCase(cbuf[i]);

}

}

return bytesRead;

}

}

public class IODecoratorExample {

public static void main(String[] args) {

try {

// Create a FileReader for the input file

FileReader fileReader = new FileReader("input.txt");

// Wrap it with LowercaseDecorator

LowercaseDecorator lowercaseDecorator = new LowercaseDecorator(fileReader);

// Create a BufferedReader for reading lines

BufferedReader bufferedReader = new BufferedReader(lowercaseDecorator);

// Read and print lines

String line;

while ((line = bufferedReader.readLine()) != null) {

System.out.println(line);

}

// Close readers

bufferedReader.close();

fileReader.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}

Q2) iris

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

data = pd.read\_csv("Iris.csv")

print (data.head(10))

x=data["sepal\_length"]

y=data["petal\_length"]

plt.scatter(x,y)

plt.show()

Q3) HTML FORM

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Student Registration Form</title>

<style>

body {

font-family: Arial, sans-serif;

}

.error {

color: red;

}

</style>

</head>

<body>

<h2>Student Registration Form</h2>

<form id="registrationForm" onsubmit="return validateForm()">

<label for="firstName">First Name:</label>

<input type="text" id="firstName" name="firstName" required>

<span id="firstNameError" class="error"></span>

<br>

<label for="lastName">Last Name:</label>

<input type="text" id="lastName" name="lastName" required>

<span id="lastNameError" class="error"></span>

<br>

<label for="age">Age:</label>

<input type="number" id="age" name="age" required>

<span id="ageError" class="error"></span>

<br>

<input type="submit" value="Register">

</form>

<script>

function validateForm() {

var firstName = document.getElementById('firstName').value;

var lastName = document.getElementById('lastName').value;

var age = document.getElementById('age').value;

// Regular expression to check if the name contains only alphabets

var nameRegex = /^[a-zA-Z]+$/;

// Validate First Name

if (!nameRegex.test(firstName)) {

document.getElementById('firstNameError').innerHTML = 'First name should contain only alphabets.';

return false;

} else {

document.getElementById('firstNameError').innerHTML = '';

}

// Validate Last Name

if (!nameRegex.test(lastName)) {

document.getElementById('lastNameError').innerHTML = 'Last name should contain only alphabets.';

return false;

} else {

document.getElementById('lastNameError').innerHTML = '';

}

// Validate Age

if (age < 18 || age > 50 || isNaN(age)) {

document.getElementById('ageError').innerHTML = 'Age should be between 18 and 50.';

return false;

} else {

document.getElementById('ageError').innerHTML = '';

}

// If all validations pass, the form is submitted

return true;

}

</script>

</body>

</html>

Slip 11:

Q1 Heart beat

// Existing BeatModel

interface BeatModel {

void beat();

}

// HeartModel (Adapter) implementing BeatModel

class HeartModelAdapter implements BeatModel {

private HeartModel heartModel;

public HeartModelAdapter(HeartModel heartModel) {

this.heartModel = heartModel;

}

@Override

public void beat() {

heartModel.heartbeat();

}

}

// Existing HeartModel

class HeartModel {

void heartbeat() {

System.out.println("Heart is beating!");

}

}

// Client code using BeatModel

class Client {

public static void main(String[] args) {

// Use the existing HeartModel with the help of the adapter

HeartModel heartModel = new HeartModel();

BeatModel adapter = new HeartModelAdapter(heartModel);

// Use the adapted interface

adapter.beat();

}

}

Q2) dataset null remove

import pandas

# reading the CSV file

csvFile = pandas.read\_csv('employees.csv')

# displaying the contents of the CSV file

print(csvFile)

count=csvFile.isnull()

#displaying NULL content

print(count)

newdf = csvFile.dropna()

print(newdf)

Q3)

npm install mysql

const mysql = require('mysql');

// Create a connection to the database

const connection = mysql.createConnection({

host: 'your\_host',

user: 'your\_user',

password: 'your\_password',

database: 'your\_database',

});

// Connect to the database

connection.connect();

// Select all records from the "customers" table

const selectQuery = 'SELECT \* FROM customers';

connection.query(selectQuery, (error, results) => {

if (error) throw error;

console.log('All records from "customers" table:', results);

// Specify the record to delete (replace 'your\_condition' with your specific condition)

const deleteQuery = 'DELETE FROM customers WHERE your\_condition';

// Delete the specified record

connection.query(deleteQuery, (deleteError, deleteResults) => {

if (deleteError) throw deleteError;

console.log('Record deleted successfully');

// Close the connection

connection.end();

});

});