**Slip-1**

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

Program Name:- vi uplo.java

import java.io.\*;

import java.util.\*;

class LowerCaseInputStream extends FilterInputStream

{

public LowerCaseInputStream(InputStream in)

{

super(in);

}

public int read() throws IOException

{

int c=super.read();

return (c==-1?c:Character.toLowerCase((char)c));

}

public int read(byte[] b,int offset,int len) throws IOException

{

int result =super.read(b,offset,len);

for (int i=offset;i<offset+result;i++)

{

b[i]=(byte)Character.toLowerCase((char)b[i]);

}

return result;

}

}

class uplo

{

public static void main(String[] args) throws IOException

{

int c;

try

{

InputStream in =new LowerCaseInputStream(new BufferedInputStream(new FileInputStream("a.txt")));

while((c = in.read()) >= 0)

{

System.out.print((char)c);

}

in.close();

}

catch(IOException e)

{

e.printStackTrace();

}

}

}

OUTPUT:-

vi a.txt

Harshada

HARSHADA

TEJU

Tejaswi

Tejaswini

Run:-

student@localhost ~]$ javac uplo.java

[student@localhost ~]$ java uplo

harshada

harshada

teju

tejaswi

tejaswini

Q2 Write a Python program to prepare Scatter Plot for Iris Dataset.

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

iris = pd.read\_csv("Iris.csv") # Reading the dataset “Iris.csv”.

print (iris.head(10)) # head() will display the top rows of the dataset, the default value of this function is 5,

#that is it will show top 5 rows when no argument is given to it.

plt.plot(iris.Id, iris["SepalLengthCm"],"r--")

plt.show #plt.show () will display the current figure that you are working on

iris.plot(kind ="scatter", x ='SepalLengthCm', y ='PetalLengthCm')

plt.grid() # grid () function to add grid lines to the plot

Q3 Create an HTML form that contain the Student Registration details and write a JavaScript to validate Student first and last name as it should not contain other than alphabets and age should be between 18 to 50.

<html>

<head>

<script type="text/javascript">

function validate()

{

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

var fname=document.getElementById("txtfname").value;

var lname=document.getElementById("txtlname").value;

var age=document.getElementById("txtage").value;

var mobno=document.getElementById("txtmobno").value;

if(age<18||age>50)

alert("student age must be 18 to 50");

if(!regName.test(fname))

alert("invalid name is given");

else

alert("valid name is given");

}

</script>

</head>

<body>

<form>

enter student first name

<input type="text" name="txtfname" id="txtfname"><br>

enter student last name

<input type="text" name="txtlname" id="txtlname"><br>

enter student age

<input type="text" name="txtage" id="txtage"><br>

enter mobile no

<input type="text" name="txtmobno" id="txtmobno"><br>

<input type="button" value="validate" onclick="validate()">

</form>

</body>

</html>

Slip-2

Q1 import java.io.\*;

import java.util.\*;

class slip2Q1

{

   private static volatile slip2Q1 instance = new slip2Q1();

   private slip2Q1(){}

   public static slip2Q1 getInstance()

   {

    {

        if (instance == null)

        {

            // To make thread safe

            synchronized (slip2Q1.class)

            {

                // check again as multiple threads

                // can reach above step

                if (instance==null)

                    instance = new slip2Q1();

            }

        }

      return instance;}

   }

   public void showMessage()

   {

      System.out.println("Hello Everyone!!!!");

   }

}

class silp2Q1

{

   public static void main(String[] args)

   {

      slip2Q1 object = slip2Q1.getInstance();

      object.showMessage();

   }

}

Q2 Write a python program to find all null values in a given dataset and remove them.

import pandas as pd

import numpy as np

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)

df.head()

df.isnull()

df.notnull()

#df=pd.DataFrame(dict)

df.fillna(0)

df.fillna(method='pad')

df.fillna(method='bfill')

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

df.dropna()

df.dropna(axis=1)

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

new\_data

3) Create an HTML form that contain the Employee Registration details and write a JavaScript to validate DOB, Joining Date, and Salary.

<html>

<head>

<script type="text/javascript">

function validate()

{

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

var dateformatdob = /^(0?[1-9]|[12][0-9]|3[01])[\/\-](0?[1-9]|1[012])[\/\-]\d{4}$/;

var dateformatjdate = /^(0?[1-9]|[12][0-9]|3[01])[\/\-](0?[1-9]|1[012])[\/\-]\d{4}$/;

//Max six digits, a dot, max two digits after dot

var salaryformat=/^\d{1,6}(?:\.\d{0,2})?$/

var name=document.getElementById("txtname").value;

var dob=document.getElementById("txtdob").value;

var jdate=document.getElementById("txtjdate").value;

var salary=document.getElementById("txtsalary").value;

if(!regName.test(name))

alert("invalid name is given");

else

alert("valid name is given");

if(!dateformatjdate.test(jdate))

alert("invalid joining date is given");

else

alert("valid joining date is given");

if(!dateformatdob.test(dob))

alert("invalid date of birth is given");

else

alert("valid date of birth is given is given");

if(!salaryformat.test(salary))

alert("invalid salary");

else

alert("salary is valid");

}

</script>

</head>

<body>

<form>

<h1>Employee Rsgistration Details</h1>

enter employee first name

<input type="text" name="txtfname" id="txtname"><br>

enter date of birth

<input type="text" name="txtdob" id="txtdob"><br>

enter joining date

<input type="text" name="txtjdate" id="txtjdate"><br>

enter salary

<input type="text" name="txtsalary" id="txtsalary"><br>

<input type="button" value="validate" onclick="validate()">

</form>

</body>

</html>

**Slip-3**

Q1 Write a JAVA Program to implement built-in support (java.util.Observable) Weather station with members temperature, humidity, pressure and methods mesurmentsChanged(), setMesurment(), getTemperature(), getHumidity(),

getPressure()

import java.util.\*;

interface Observer {

public void update(float temp, float humidity, float pressure);

}

interface DisplayElement {

public void display();

}

interface Subject {

public void registerObserver(Observer o);

public void removeObserver(Observer o);

public void notifyObservers();

}

class WeatherData implements Subject {

private ArrayList<Observer> observers;

private float temperature;

private float humidity;

private float pressure;

public WeatherData() {

observers = new ArrayList<>();

}

public void registerObserver(Observer o) {

observers.add(o);

}

public void removeObserver(Observer o) {

int i = observers.indexOf(o);

if (i >= 0) {

observers.remove(i);

}

}

public void notifyObservers() {

for (int i = 0; i < observers.size(); i++) {

Observer observer = (Observer) observers.get(i);

observer.update(temperature, humidity, pressure);

}

}

public void measurementsChanged() {

notifyObservers();

}

public void setMeasurements(float temperature, float humidity, float pressure) {

this.temperature = temperature;

this.humidity = humidity;

this.pressure = pressure;

measurementsChanged();

}

public float getTemperature() {

return temperature;

}

public float getHumidity() {

return humidity;

}

public float getPressure() {

return pressure;

}

}

class ForecastDisplay implements Observer, DisplayElement {

private float currentPressure = 29.92f;

private float lastPressure;

private WeatherData weatherData;

public ForecastDisplay(WeatherData weatherData) {

this.weatherData = weatherData;

weatherData.registerObserver(this);

}

public void update(float temp, float humidity, float pressure) {

lastPressure = currentPressure;

currentPressure = pressure;

display();

}

public void display() {

System.out.print("Forecast: ");

if (currentPressure > lastPressure) {

System.out.println("Improving weather on the way!");

} else if (currentPressure == lastPressure) {

System.out.println("More of the same");

} else if (currentPressure < lastPressure) {

System.out.println("Watch out for cooler weather, rainy weather");

}

}

}

class HeatIndexDisplay implements Observer, DisplayElement {

float heatIndex = 0.0f;

private WeatherData weatherData;

public HeatIndexDisplay(WeatherData weatherData) {

this.weatherData = weatherData;

weatherData.registerObserver(this);

}

public void update(float t, float rh, float pressure) {

heatIndex = computeHeatIndex(t, rh);

display();

}

private float computeHeatIndex(float t, float rh) {

float index = (float) ((16.923 + (0.185212 \* t) + (5.37941 \* rh) - (0.100254 \* t \* rh)

+ (0.00941695 \* (t \* t)) + (0.00728898 \* (rh \* rh))

+ (0.000345372 \* (t \* t \* rh)) - (0.000814971 \* (t \* rh \* rh)) +

(0.0000102102 \* (t \* t \* rh \* rh)) - (0.000038646 \* (t \* t \* t)) + (0.0000291583 \*

(rh \* rh \* rh))

+ (0.00000142721 \* (t \* t \* t \* rh)) +

(0.000000197483 \* (t \* rh \* rh \* rh)) - (0.0000000218429 \* (t \* t \* t \* rh \* rh)) +

0.000000000843296 \* (t \* t \* rh \* rh \* rh)) -

(0.0000000000481975 \* (t \* t \* t \* rh \* rh \* rh)));

return index;

}

public void display() {

System.out.println("Heat index is " + heatIndex);

}

}

class StatisticsDisplay implements Observer, DisplayElement {

private float maxTemp = 0.0f;

private float minTemp = 200;

private float tempSum = 0.0f;

private int numReadings;

private WeatherData weatherData;

public StatisticsDisplay(WeatherData weatherData) {

this.weatherData = weatherData;

weatherData.registerObserver(this);

}

public void update(float temp, float humidity, float pressure) {

tempSum += temp;

numReadings++;

if (temp > maxTemp) {

maxTemp = temp;

}

if (temp < minTemp) {

minTemp = temp;

}

display();

}

public void display() {

System.out.println("Avgerage/Maximum/Minimum temperature = " + (tempSum / numReadings)

+ "/" + maxTemp + "/" + minTemp);

}

}

class CurrentConditionsDisplay implements Observer, DisplayElement {

private float temperature;

private float humidity;

private Subject weatherData;

public CurrentConditionsDisplay(Subject weatherData) {

this.weatherData = weatherData;

weatherData.registerObserver(this);

}

public void update(float temperature, float humidity, float pressure) {

this.temperature = temperature;

this.humidity = humidity;

display();

}

public void display() {

System.out.println("Current conditions: " + temperature

+ "F degrees and " + humidity + "% humidity");

}

}

class weather {

public static void main(String[] args) {

WeatherData weatherData = new WeatherData();

CurrentConditionsDisplay currentDisplay = new CurrentConditionsDisplay(weatherData);

StatisticsDisplay statisticsDisplay = new StatisticsDisplay(weatherData);

ForecastDisplay forecastDisplay = new ForecastDisplay(weatherData);

weatherData.setMeasurements(70, 55, 40.4f);

weatherData.setMeasurements(72, 60, 39.2f);

weatherData.setMeasurements(68, 80, 39.2f);

}

}

2) Write a python program to make Categorical values in numeric format for a given dataset.

import pandas as pd

#load Iris data set

iris = pd.read\_csv('Iris.csv')

iris.head()

iris['code']=pd.factorize(iris.Species)[0]

iris.tail()

iris.code.value\_counts()

3) Create an HTML form for Login and write a JavaScript to validate email ID using Regular Expression.

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1">

<style>

</style>

</head>

<body>

<script>

function validateform(){

var email = document.getElementById("email").value;

var password = document.getElementById("psw").value;

if (/^\w+([\.-]?\w+)\*@\w+([\.-]?\w+)\*(\.\w{2,3})+$/.test(email))

{

alert("Valid Email Id..")

return (true)

}

else{ alert("You have entered an invalid email address!")

return (false)

}

}

</script>

<form name="myform" onsubmit="return validateform()">

<div class="container">

<p>Please fill in this form to Login.</p>

<hr>

<label for="email"><b>Email</b></label>

<input type="text" autocomplete="off" placeholder="Enter Email" name="email" id="email" required>

<label for="psw"><b>Password</b></label>

<input type="password" autocomplete="off" placeholder="Enter Password" name="psw" id="psw" required>

<hr>

<button type="submit" class="registerbtn">Register</button>

</div>

</form>

</body>

</html>

**Slip-4**

Q1 Write a Java Program to implement Factory method for Pizza Store with createPizza(), orederPizza(), prepare(), Bake(), cut(), box(). Use this to create variety of pizza’s like NyStyleCheesePizza, ChicagoStyleCheesePizza etc.

import java.util.ArrayList;

class ChicagoPizzaStore extends PizzaStore

{Pizza createPizza(String item)

{if (item.equals("cheese"))

{return new ChicagoStyleCheesePizza();

}

else if (item.equals("veggie"))

{return new ChicagoStyleVeggiePizza();

}

else if (item.equals("clam"))

{return new ChicagoStyleClamPizza();

}

else if (item.equals("pepperoni"))

{return new ChicagoStylePepperoniPizza();

}

else return null;

}}

class ChicagoStyleCheesePizza extends Pizza

{public ChicagoStyleCheesePizza()

{

name = "Chicago Style Deep Dish Cheese Pizza";

dough = "Extra Thick Crust Dough";

sauce = "Plum Tomato Sauce";

toppings.add("Shredded Mozzarella Cheese");

}

void cut()

{System.out.println("Cutting the pizza into square slices");

}}

class ChicagoStyleClamPizza extends Pizza

{public ChicagoStyleClamPizza()

{

name = "Chicago Style Clam Pizza";

dough = "Extra Thick Crust Dough";

sauce = "Plum Tomato Sauce";

toppings.add("Shredded Mozzarella Cheese");

toppings.add("Frozen Clams from Chesapeake Bay");

}

void cut()

{System.out.println("Cutting the pizza into square slices");

}}

class ChicagoStylePepperoniPizza extends Pizza

{public ChicagoStylePepperoniPizza()

{

name = "Chicago Style Pepperoni Pizza";

dough = "Extra Thick Crust Dough";

sauce = "Plum Tomato Sauce";

toppings.add("Shredded Mozzarella Cheese");

toppings.add("Black Olives");

toppings.add("Spinach");

toppings.add("Eggplant");

toppings.add("Sliced Pepperoni");

}

void cut()

{System.out.println("Cutting the pizza into square slices");

}}

class ChicagoStyleVeggiePizza extends Pizza

{public ChicagoStyleVeggiePizza()

{name = "Chicago Deep Dish Veggie Pizza";

dough = "Extra Thick Crust Dough";

sauce = "Plum Tomato Sauce";

toppings.add("Shredded Mozzarella Cheese");

toppings.add("Black Olives");

toppings.add("Spinach");

toppings.add("Eggplant");

}

void cut()

{System.out.println("Cutting the pizza into square slices");

}}

class DependentPizzaStore

{public Pizza createPizza(String style, String type)

{ Pizza pizza = null;

if (style.equals("NY"))

{if (type.equals("cheese"))

{pizza = new NYStyleCheesePizza();

}

else if (type.equals("veggie"))

{pizza = new NYStyleVeggiePizza();

}

else if (type.equals("clam"))

{pizza = new NYStyleClamPizza();

}

else if (type.equals("pepperoni"))

{pizza = new NYStylePepperoniPizza();

}}

else if (style.equals("Chicago"))

{if (type.equals("cheese"))

{pizza = new ChicagoStyleCheesePizza();

}

else if (type.equals("veggie"))

{pizza = new ChicagoStyleVeggiePizza();

}

else if (type.equals("clam"))

{pizza = new ChicagoStyleClamPizza();

}

else if (type.equals("pepperoni"))

{pizza = new ChicagoStylePepperoniPizza();

}}

else

{System.out.println("Error: invalid type of pizza");

return null;

}

pizza.prepare();

pizza.bake();

pizza.cut();

pizza.box();

return pizza;

}}

class NYPizzaStore extends PizzaStore

{Pizza createPizza(String item)

{if (item.equals("cheese"))

{return new NYStyleCheesePizza();

}

else if (item.equals("veggie"))

{return new NYStyleVeggiePizza();

}

else if (item.equals("clam"))

{return new NYStyleClamPizza();

}

else if (item.equals("pepperoni"))

{return new NYStylePepperoniPizza();

}

else return null;

}}

class NYStyleCheesePizza extends Pizza

{public NYStyleCheesePizza()

{

name = "NY Style Sauce and Cheese Pizza";

dough = "Thin Crust Dough";

sauce = "Marinara Sauce";

toppings.add("Grated Reggiano Cheese");

}}

class NYStyleClamPizza extends Pizza

{public NYStyleClamPizza()

{

name = "NY Style Clam Pizza";

dough = "Thin Crust Dough";

sauce = "Marinara Sauce";

toppings.add("Grated Reggiano Cheese");

toppings.add("Fresh Clams from Long Island Sound");

}}

class NYStylePepperoniPizza extends Pizza

{public NYStylePepperoniPizza()

{

name = "NY Style Pepperoni Pizza";

dough = "Thin Crust Dough";

sauce = "Marinara Sauce";

toppings.add("Grated Reggiano Cheese");

toppings.add("Sliced Pepperoni");

toppings.add("Garlic");

toppings.add("Onion");

toppings.add("Mushrooms");

toppings.add("Red Pepper");

}}

class NYStyleVeggiePizza extends Pizza

{public NYStyleVeggiePizza()

{

name = "NY Style Veggie Pizza";

dough = "Thin Crust Dough";

sauce = "Marinara Sauce";

toppings.add("Grated Reggiano Cheese");

toppings.add("Garlic");

toppings.add("Onion");

toppings.add("Mushrooms");

toppings.add("Red Pepper");

}}

abstract class Pizza

{

String name;

String dough;

String sauce;

ArrayList toppings = new ArrayList();

void prepare()

{

System.out.println("Preparing " + name);

System.out.println("Tossing dough...");

System.out.println("Adding sauce...");

System.out.println("Adding toppings: ");

for (int i = 0; i < toppings.size(); i++)

{System.out.println(" " + toppings.get(i));

}}

void bake()

{System.out.println("Bake for 25 minutes at 350");

}

void cut()

{System.out.println("Cutting the pizza into diagonal slices");

}

void box()

{System.out.println("Place pizza in official PizzaStore box");

}

public String getName()

{return name;

}

public String toString()

{StringBuffer display = new StringBuffer();

display.append("---- " + name + " ----\n");

display.append(dough + "\n");

display.append(sauce + "\n");

for (int i = 0; i < toppings.size(); i++)

{display.append((String )toppings.get(i) + "\n");

}

return display.toString();

}}

abstract class PizzaStore

{abstract Pizza createPizza(String item);

public Pizza orderPizza(String type)

{Pizza pizza = createPizza(type);

System.out.println("--- Making a " + pizza.getName() + " ---");

pizza.prepare();

pizza.bake();

pizza.cut();

pizza.box();

return pizza;

}}

public class Main

{public static void main(String[] args)

{

PizzaStore nyStore = new NYPizzaStore();

PizzaStore chicagoStore = new ChicagoPizzaStore();

Pizza pizza = nyStore.orderPizza("cheese");

System.out.println("Ethan ordered a " + pizza.getName() + "\n");

pizza = chicagoStore.orderPizza("cheese");

System.out.println("Joel ordered a " + pizza.getName() + "\n");

pizza = nyStore.orderPizza("clam");

System.out.println("Ethan ordered a " + pizza.getName() + "\n");

pizza = chicagoStore.orderPizza("clam");

System.out.println("Joel ordered a " + pizza.getName() + "\n");

pizza = nyStore.orderPizza("pepperoni");

System.out.println("Ethan ordered a " + pizza.getName() + "\n");

pizza = chicagoStore.orderPizza("pepperoni");

System.out.println("Joel ordered a " + pizza.getName() + "\n");

pizza = nyStore.orderPizza("veggie");

System.out.println("Ethan ordered a " + pizza.getName() + "\n");

pizza = chicagoStore.orderPizza("veggie");

System.out.println("Joel ordered a " + pizza.getName() + "\n");

}}

Q2 Write a python program to Implement Simple Linear Regression for predicting house price.

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

from sklearn.model\_selection import train\_test\_split

from sklearn.model\_selection import cross\_val\_predict

data = pd.read\_csv(r'kc\_house\_data.csv')

data.head(5)

print(data.shape)

# Make a list of importatnt feature which is needed to be incuding in training data

f = ['price', 'bedrooms', 'bathrooms', 'sqft\_living', 'floors', 'condition', 'sqft\_above', 'sqft\_basement', 'yr\_built',

'yr\_renovated']

data = data[f]

print(data.shape)

# Drop the missing values

data = data.dropna()

print(data.shape)

# Get the statictial information of the dataset

data.describe()

# Now,Divide the dataset into two parts:independent variable and dependent variable

X = data[f[1:]]

y = data['price']

# Split the dataset into training data and testing data

X\_train, X\_test, y\_train, y\_test = train\_test\_split( X, y, test\_size=0.2, random\_state=42)

print(X\_train.shape)

print(X\_test.shape)

print(y\_train.shape)

print(y\_test.shape)

# Fit the regression model

lr = LinearRegression() # Create object of linear regression class

lr.fit(X\_train,y\_train) #fit training data

print(lr.coef\_)

# Create the Prediction

y\_test\_predict = lr.predict(X\_test)

print(y\_test\_predict.shape)

# Plot the error

g=plt.plot((y\_test - y\_test\_predict),marker='o',linestyle='')

# # Fit the regression model without b(w0)

lr = LinearRegression(fit\_intercept=False)

lr.fit(X\_train,y\_train)

y\_test\_predict = lr.predict(X\_test)

g=plt.plot((y\_test - y\_test\_predict),marker='o',linestyle='')

3) Create a Node.js file that will convert the output "Hello World!" into upper-case letters.

var http = require('http'); // includes the http module

var uc = require('upper-case'); // include the upper-case module

http.createServer(function (req, res) {

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

res.write(("hello world!").toUpperCase()); // assign the upper-case module

res.end();

}).listen(8080);

**Slip-5**

Q1 Write a Java Program to implement Adapter pattern for Enumeration iterator

import java.util.\*;

class EnumerationIterator implements Iterator {

Enumeration enumeration;

public EnumerationIterator(Enumeration enumeration) {

this.enumeration = enumeration;

}

public boolean hasNext() {

return enumeration.hasMoreElements();

}

public Object next() {

return enumeration.nextElement();

}

public void remove() {

throw new UnsupportedOperationException();

}

}

class EnumerationIteratorTestDrive1 {

public static void main (String args[]) {

Vector v = new Vector(Arrays.asList("apple","mango","grapes"));

Iterator iterator = new EnumerationIterator(v.elements());

while (iterator.hasNext()) {

System.out.println(iterator.next());

}

}

}

2) Write a python program to implement Multiple Linear Regression for given dataset.

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

dataset = pd.read\_csv('50\_Startups.csv')

X = dataset.iloc[:, :-1].values

y = dataset.iloc[:, -1].values

from sklearn.compose import ColumnTransformer

from sklearn.preprocessing import OneHotEncoder

ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [3])],

remainder='passthrough')

X = np.array(ct.fit\_transform(X))

print(X)

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2)

from sklearn.linear\_model import LinearRegression

regressor = LinearRegression()

regressor.fit(X\_train, y\_train)

y\_pred = regressor.predict(X\_test)

df = pd.DataFrame({'Real Values':y\_test, 'Predicted Values':y\_pred})

df

3) Using nodejs create a web page to read two file names from user and append contents of first file into second file.

const fs = require('fs');

console.log("\nFile Contents of file before append:",

a=fs.readFileSync("file1.txt", "utf8"));

fs.appendFile("file2.txt", a, (err) => {

if (err) {

console.log(err);

}

else {

console.log("\nFile Contents of file after append:",

fs.readFileSync("file2.txt", "utf8"));

}

});

**Slip-6**

Q1 Write a Java Program to implement command pattern to test Remote Control.

public interface Command {

public void execute();

public void undo();

}

public class CeilingFanOffCommand implements Command {

CeilingFan ceilingFan;

int prevSpeed;

public CeilingFanOffCommand(CeilingFan ceilingFan) {

this.ceilingFan = ceilingFan;

}

public void execute() {

prevSpeed = ceilingFan.getSpeed();

ceilingFan.off();

}

public void undo() {

if (prevSpeed == CeilingFan.HIGH) {

ceilingFan.high();

} else if (prevSpeed == CeilingFan.MEDIUM) {

ceilingFan.medium();

} else if (prevSpeed == CeilingFan.LOW) {

ceilingFan.low();

} else if (prevSpeed == CeilingFan.OFF) {

ceilingFan.off();

}

}

}

public class CeilingFanMediumCommand implements Command {

CeilingFan ceilingFan;

int prevSpeed;

public CeilingFanMediumCommand(CeilingFan ceilingFan) {

this.ceilingFan = ceilingFan;

}

public void execute() {

prevSpeed = ceilingFan.getSpeed();

ceilingFan.medium();

}

public void undo() {

if (prevSpeed == CeilingFan.HIGH) {

ceilingFan.high();

} else if (prevSpeed == CeilingFan.MEDIUM) {

ceilingFan.medium();

} else if (prevSpeed == CeilingFan.LOW) {

ceilingFan.low();

} else if (prevSpeed == CeilingFan.OFF) {

ceilingFan.off();

}

}

}

public class CeilingFanHighCommand implements Command {

CeilingFan ceilingFan;

int prevSpeed;

public CeilingFanHighCommand(CeilingFan ceilingFan) {

this.ceilingFan = ceilingFan;

}

public void execute() {

prevSpeed = ceilingFan.getSpeed();

ceilingFan.high();

}

public void undo() {

if (prevSpeed == CeilingFan.HIGH) {

ceilingFan.high();

} else if (prevSpeed == CeilingFan.MEDIUM) {

ceilingFan.medium();

} else if (prevSpeed == CeilingFan.LOW) {

ceilingFan.low();

} else if (prevSpeed == CeilingFan.OFF) {

ceilingFan.off();

}

}

}

public class CeilingFanLowCommand implements Command {

CeilingFan ceilingFan;

int prevSpeed;

public CeilingFanLowCommand(CeilingFan ceilingFan) {

this.ceilingFan = ceilingFan;

}

public void execute() {

prevSpeed = ceilingFan.getSpeed();

ceilingFan.low();

}

public void undo() {

if (prevSpeed == CeilingFan.HIGH) {

ceilingFan.high();

} else if (prevSpeed == CeilingFan.MEDIUM) {

ceilingFan.medium();

} else if (prevSpeed == CeilingFan.LOW) {

ceilingFan.low();

} else if (prevSpeed == CeilingFan.OFF) {

ceilingFan.off();

}

}

}

public class CeilingFan {

public static final int HIGH = 3;

public static final int MEDIUM = 2;

public static final int LOW = 1;

public static final int OFF = 0;

String location;

int speed;

public CeilingFan(String location) {

this.location = location;

speed = OFF;

}

public void high() {

speed = HIGH;

System.out.println(location + " ceiling fan is on high");

}

public void medium() {

speed = MEDIUM;

System.out.println(location + " ceiling fan is on medium");

}

public void low() {

speed = LOW;

System.out.println(location + " ceiling fan is on low");

}

public void off() {

speed = OFF;

System.out.println(location + " ceiling fan is off");

}

public int getSpeed() {

return speed;

}

}

2) Write a python program to implement Polynomial Linear Regression for given dataset

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

dataset = pd.read\_csv('Position\_Salaries.csv')

X = dataset.iloc[:, 1:-1].values

y = dataset.iloc[:, -1].values

dataset.head(5)

from sklearn.preprocessing import PolynomialFeatures

from sklearn.linear\_model import LinearRegression

poly\_reg = PolynomialFeatures(degree = 4)

X\_poly = poly\_reg.fit\_transform(X)

lin\_reg = LinearRegression()

lin\_reg.fit(X\_poly, y)

y\_pred = lin\_reg.predict(X\_poly)

df = pd.DataFrame({'Real Values':y, 'Predicted Values':y\_pred})

df

X\_grid = np.arange(min(X), max(X), 0.1)

X\_grid = X\_grid.reshape((len(X\_grid), 1))

plt.scatter(X, y, color = 'red')

plt.scatter(X, y\_pred, color = 'green')

plt.plot(X\_grid,

lin\_reg.predict(poly\_reg.fit\_transform(X\_grid)), color =

'black')

plt.title('Polynomial Regression')

plt.xlabel('Position level')

plt.ylabel('Salary')

plt.show()

3) Create a Node.js file that opens the requested file and returns the content to the client. If anything goes wrong, throw a 404 error.

var fs = require('fs');

//var http = require('http');

//http.createServer(function (req, res) {

// Use fs.readFile() method to read the file

fs.readFile('file1.txt', 'utf8', function(err, data){

/\*if (err){

res.writeHead(404, {'Content-Type': 'text/html'});

return res.end("404 Not Found");

}\*/

// Display the file content

if(err) return console.error(err);

console.log(data);

});

console.log('readFile called');

**Slip-7**

Q1 Write a Java Program to implement undo command to test Ceiling fan.

public interface Command {

public void execute();

public void undo();

}

public class CeilingFanOffCommand implements Command {

CeilingFan ceilingFan;

int prevSpeed;

public CeilingFanOffCommand(CeilingFan ceilingFan) {

this.ceilingFan = ceilingFan;

}

public void execute() {

prevSpeed = ceilingFan.getSpeed();

ceilingFan.off();

}

public void undo() {

if (prevSpeed == CeilingFan.HIGH) {

ceilingFan.high();

} else if (prevSpeed == CeilingFan.MEDIUM) {

ceilingFan.medium();

} else if (prevSpeed == CeilingFan.LOW) {

ceilingFan.low();

} else if (prevSpeed == CeilingFan.OFF) {

ceilingFan.off();

}

}

}

public class CeilingFanMediumCommand implements Command {

CeilingFan ceilingFan;

int prevSpeed;

public CeilingFanMediumCommand(CeilingFan ceilingFan) {

this.ceilingFan = ceilingFan;

}

public void execute() {

prevSpeed = ceilingFan.getSpeed();

ceilingFan.medium();

}

public void undo() {

if (prevSpeed == CeilingFan.HIGH) {

ceilingFan.high();

} else if (prevSpeed == CeilingFan.MEDIUM) {

ceilingFan.medium();

} else if (prevSpeed == CeilingFan.LOW) {

ceilingFan.low();

} else if (prevSpeed == CeilingFan.OFF) {

ceilingFan.off();

}

}

}

public class CeilingFanHighCommand implements Command {

CeilingFan ceilingFan;

int prevSpeed;

public CeilingFanHighCommand(CeilingFan ceilingFan) {

this.ceilingFan = ceilingFan;

}

public void execute() {

prevSpeed = ceilingFan.getSpeed();

ceilingFan.high();

}

public void undo() {

if (prevSpeed == CeilingFan.HIGH) {

ceilingFan.high();

} else if (prevSpeed == CeilingFan.MEDIUM) {

ceilingFan.medium();

} else if (prevSpeed == CeilingFan.LOW) {

ceilingFan.low();

} else if (prevSpeed == CeilingFan.OFF) {

ceilingFan.off();

}

}

}

public class CeilingFanLowCommand implements Command {

CeilingFan ceilingFan;

int prevSpeed;

public CeilingFanLowCommand(CeilingFan ceilingFan) {

this.ceilingFan = ceilingFan;

}

public void execute() {

prevSpeed = ceilingFan.getSpeed();

ceilingFan.low();

}

public void undo() {

if (prevSpeed == CeilingFan.HIGH) {

ceilingFan.high();

} else if (prevSpeed == CeilingFan.MEDIUM) {

ceilingFan.medium();

} else if (prevSpeed == CeilingFan.LOW) {

ceilingFan.low();

} else if (prevSpeed == CeilingFan.OFF) {

ceilingFan.off();

}

}

}

public class CeilingFan {

public static final int HIGH = 3;

public static final int MEDIUM = 2;

public static final int LOW = 1;

public static final int OFF = 0;

String location;

int speed;

public CeilingFan(String location) {

this.location = location;

speed = OFF;

}

public void high() {

speed = HIGH;

System.out.println(location + " ceiling fan is on high");

}

public void medium() {

speed = MEDIUM;

System.out.println(location + " ceiling fan is on medium");

}

public void low() {

speed = LOW;

System.out.println(location + " ceiling fan is on low");

}

public void off() {

speed = OFF;

System.out.println(location + " ceiling fan is off");

}

public int getSpeed() {

return speed;

}

}

2) Write a python program to implement Naive Bayes.

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

%matplotlib inline

import seaborn as sns

data=pd.read\_csv('pima-indians-diabetes.csv')

data.shape

data.isnull().sum()

data.isnull().values.any()

data.dtypes

#visualisation

data.hist()

columns=list(data)[0:-1]

data[columns].hist()

#identifty the correlation

data.corr()

sns.heatmap(data.corr(),annot=True)

sns.pairplot(data)

#calculate diabetes ratio of true or false target varible

n\_true=len(data.loc[data['class']==True])

n\_false=len(data.loc[data['class']==False])

print("No.of true cases:{0} {1}%".format(n\_true,(n\_true/(n\_true+n\_false))\*100))

print("No.of false cases:{0} {1}%".format(n\_false,(n\_false/(n\_true+n\_false))\*100))

#split the data

from sklearn.model\_selection import train\_test\_split

x=data.drop('class',axis=1)

y=data['class']

X\_train,X\_test,Y\_train,Y\_test=train\_test\_split(x,y,test\_size=0.30,random\_state=1)

from sklearn.impute import SimpleImputer

rep\_0=SimpleImputer(missing\_values=0,strategy='mean')

cols=X\_train.columns

X\_train=pd.DataFrame(rep\_0.fit\_transform(X\_train))

X\_test=pd.DataFrame(rep\_0.fit\_transform(X\_test))

X\_train.columns=cols

X\_test.columns=cols

X\_train.head()

from sklearn.naive\_bayes import GaussianNB

diab\_mode=GaussianNB()

diab\_mode.fit(X\_train,Y\_train)

diab\_train\_predict=diab\_mode.predict(X\_train)

from sklearn import metrics

print("Model Accuracy:{0}".format(metrics.accuracy\_score(Y\_train,diab\_train\_predict)))

diab\_train\_predict=diab\_mode.predict(X\_test)

from sklearn import metrics

print("Model Accuracy:{0}".format(metrics.accuracy\_score(Y\_test,diab\_train\_predict)))

cm1=metrics.confusion\_matrix(Y\_test,diab\_train\_predict,labels=[1,0])

df\_cm1=pd.DataFrame(cm1,index=[i for i in['1','0']],columns=[i for i in['predict 1','predict o']] )

df\_cm1

3) Create a Node.js file that writes an HTML form, with an upload field.

var http = require('http');

var formidable = require('formidable');

http.createServer(function (req, res) {

if (req.url == '/fileupload') {

var form = new formidable.IncomingForm();

form.parse(req, function (err, fields, files) {

res.write('File uploaded');

res.end();

});

} else {

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

res.write('<form action="fileupload" method="post" enctype="multipart/form-data">');

res.write('<input type="file" name="filetoupload"><br>');

res.write('<input type="submit">');

res.write('</form>');

return res.end();

}

}).listen(8080);

**Slip-8**

Q1 Write a Java Program to implement State Pattern for Gumball Machine. Create instance variable that holds current state from there, we just need to handle all actions, behaviors and state transition that can happen.

State.java

public interface State {

public void insertQuarter();

public void ejectQuarter();

public void turnCrank();

public void dispense();

public void refill();

}

SoldStat.java

public class SoldState implements State {

GumballMachine gumballMachine;

public SoldState(GumballMachine gumballMachine) {

this.gumballMachine = gumballMachine;

}

public void insertQuarter() {

System.out.println("Please wait, we're already giving you a gumball");

}

public void ejectQuarter() {

System.out.println("Sorry, you already turned the crank");

}

public void turnCrank() {

System.out.println("Turning twice doesn't get you another gumball!");

}

public void dispense() {

gumballMachine.releaseBall();

if (gumballMachine.getCount() > 0) {

gumballMachine.setState(gumballMachine.getNoQuarterState());

} else {

System.out.println("Oops, out of gumballs!");

gumballMachine.setState(gumballMachine.getSoldOutState());

}

}

public void refill() { }

public String toString() {

return "dispensing a gumball";

}

}

SoldOutState.java

public class SoldOutState implements State {

GumballMachine gumballMachine;

public SoldOutState(GumballMachine gumballMachine) {

this.gumballMachine = gumballMachine;

}

public void insertQuarter() {

System.out.println("You can't insert a quarter, the machine is sold out");

}

public void ejectQuarter() {

System.out.println("You can't eject, you haven't inserted a quarter yet");

}

public void turnCrank() {

System.out.println("You turned, but there are no gumballs");

}

public void dispense() {

System.out.println("No gumball dispensed");

}

public void refill() {

gumballMachine.setState(gumballMachine.getNoQuarterState());

}

public String toString() {

return "sold out";

}

}

NoQuarterState.java

public class NoQuarterState implements State {

GumballMachine gumballMachine;

public NoQuarterState(GumballMachine gumballMachine) {

this.gumballMachine = gumballMachine;

}

public void insertQuarter() {

System.out.println("You inserted a quarter");

gumballMachine.setState(gumballMachine.getHasQuarterState());

}

public void ejectQuarter() {

System.out.println("You haven't inserted a quarter");

}

public void turnCrank() {

System.out.println("You turned, but there's no quarter");

}

public void dispense() {

System.out.println("You need to pay first");

}

public void refill() { }

public String toString() {

return "waiting for quarter";

}

}

HasQuarterState.java

public class HasQuarterState implements State {

GumballMachine gumballMachine;

public HasQuarterState(GumballMachine gumballMachine) {

this.gumballMachine = gumballMachine;

}

public void insertQuarter() {

System.out.println("You can't insert another quarter");

}

public void ejectQuarter() {

System.out.println("Quarter returned");

gumballMachine.setState(gumballMachine.getNoQuarterState());

}

public void turnCrank() {

System.out.println("You turned...");

gumballMachine.setState(gumballMachine.getSoldState());

}

public void dispense() {

System.out.println("No gumball dispensed");

}

public void refill() { }

public String toString() {

return "waiting for turn of crank";

}

}

GumballMachine.java

public class GumballMachine {

State soldOutState;

State noQuarterState;

State hasQuarterState;

State soldState;

State state;

int count = 0;

public GumballMachine(int numberGumballs) {

soldOutState = new SoldOutState(this);

noQuarterState = new NoQuarterState(this);

hasQuarterState = new HasQuarterState(this);

soldState = new SoldState(this);

this.count = numberGumballs;

if (numberGumballs > 0) {

state = noQuarterState;

} else {

state = soldOutState;

}

}

public void insertQuarter() {

state.insertQuarter();

}

public void ejectQuarter() {

state.ejectQuarter();

}

public void turnCrank() {

state.turnCrank();

state.dispense();

}

void releaseBall() {

System.out.println("A gumball comes rolling out the slot...");

if (count != 0) {

count = count - 1;

}

}

int getCount() {

return count;

}

void refill(int count) {

this.count += count;

System.out.println("The gumball machine was just refilled; it's new count is: " + this.count);

state.refill();

}

void setState(State state) {

this.state = state;

}

public State getState() {

return state;

}

public State getSoldOutState() {

return soldOutState;

}

public State getNoQuarterState() {

return noQuarterState;

}

public State getHasQuarterState() {

return hasQuarterState;

}

public State getSoldState() {

return soldState;

}

public String toString() {

StringBuffer result = new StringBuffer();

result.append("\nMighty Gumball, Inc.");

result.append("\nJava-enabled Standing Gumball Model #2004");

result.append("\nInventory: " + count + " gumball");

if (count != 1) {

result.append("s");

}

result.append("\n");

result.append("Machine is " + state + "\n");

return result.toString();

}

}

GumballMachineTestDrive.java

public class GumballMachineTestDrive{

public static void main(String[] args) {

GumballMachine gumballMachine = new GumballMachine(2);

System.out.println(gumballMachine);

gumballMachine.insertQuarter();

gumballMachine.turnCrank();

System.out.println(gumballMachine);

gumballMachine.insertQuarter();

gumballMachine.turnCrank();

gumballMachine.insertQuarter();

gumballMachine.turnCrank();

gumballMachine.refill(3);

gumballMachine.insertQuarter();

gumballMachine.turnCrank();

System.out.println(gumballMachine);

}

}

2) Write a python program to implement Decision Tree whether or not to play Tennis.

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

%matplotlib inline

from sklearn.tree import DecisionTreeClassifier

from sklearn.model\_selection import train\_test\_split

from scipy.stats import zscore

import seaborn as sns

data=pd.read\_csv('play\_tennis.csv')

data.isnull().sum()

data.dtypes

data.head()

data.outlook.value\_counts()

from sklearn.preprocessing import LabelEncoder

l=LabelEncoder()

for i in data.columns:

if data[i].dtypes=='object' or data[i].dtypes=='bool':

data[i]=pd.Categorical(data[i])

for i in data.columns:

data[i]=l.fit\_transform(data[i])

data.dtypes

data.head()

x=data.drop(['play'],axis=1)

y=data['play']

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.30,random\_state=1)

dtree=DecisionTreeClassifier(criterion='gini',random\_state=1)

dtree.fit(x\_train,y\_train)

print(dtree.score(x\_train,y\_train)) #data is over fitted so we use max\_depth =5 means prunning technique

print(dtree.score(x\_test,y\_test))

dtree1=DecisionTreeClassifier(criterion='gini',max\_depth=5,random\_state=1)

dtree1.fit(x\_train,y\_train)

print(dtree1.score(x\_train,y\_train)) #data is over fitted so we use max\_depth =5 means prunning technique

print(dtree1.score(x\_test,y\_test))

y\_predict=dtree.predict(x\_test)

from sklearn import metrics

cm=metrics.confusion\_matrix(y\_test,y\_predict,labels=[1,0])

df\_cm=pd.DataFrame(cm,index=[i for i in['1','0']],columns=[i for i in['predicted 1','predicted 0']] )

df\_cm

sns.heatmap(df\_cm,annot=True)

from sklearn.metrics import classification\_report

m=classification\_report(y\_test,y\_predict)

print(m)

3) Create a Node.js file that demonstrates create database and table in MySQL.

var mysql = require('mysql');

var con = mysql.createConnection({

host: "localhost",

user: "root",

password: "Root@123",

database: "node"

});

con.connect(function(err) {

if (err) throw err;

console.log("Connected!");

con.query("CREATE DATABASE Noded", function (err, result) {

if (err) throw err;

console.log("Database created");

});

});

var sql = "CREATE TABLE customers (name VARCHAR(25), address VARCHAR(25))";

con.query(sql, function (err, result) {

if (err) throw err;

console.log("Table created");

});

**Slip-9**

Q1 Design simple HR Application using Spring Framework

2) Write a python program to implement Linear SVM.

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

%matplotlib inline

from sklearn.model\_selection import train\_test\_split

from sklearn import svm

data=pd.read\_csv('letterdata.csv')

data.head()

def getaccuracy(testset,prediction):

correct=0

for x in range(len(testset)):

if(testset[x]==prediction[x]):

correct=correct+1

return (correct/float(len(testset)))\*100.0

data.isnull().values.any()

X,Y=np.array(data)[:,1:16],np.array(data.letter)[:]

X\_train=X[:16000,:]

X\_test=X[16001:,:]

Y\_train=Y[:16000]

Y\_test=Y[16001:]

clf=svm.SVC(gamma=0.025,C=3)

clf.fit(X\_train,Y\_train)

Y\_predict=clf.predict(X\_test)

getaccuracy(Y\_test,Y\_predict)

y\_g=(np.column\_stack([Y\_test,Y\_predict]))

#column stack used for matching supose x=1,2,3,4,5 and y= 10 20 30 40 match x axis on 1 check on y axis on 10 ...

print(y\_g)

import numpy as np

import matplotlib.pyplot as plt

from sklearn import svm, datasets

# import some data to play with

iris = datasets.load\_iris()

X = iris.data[:, :2] # we only take the first two features. We could

# avoid this ugly slicing by using a two-dim dataset

y = iris.target

# we create an instance of SVM and fit out data. We do not scale our

# data since we want to plot the support vectors

C = 1.0 # SVM regularization parameter

svc = svm.SVC(kernel='linear', C=1,gamma=10).fit(X, y)

# create a mesh to plot in

x\_min, x\_max = X[:, 0].min() - 1, X[:, 0].max() + 1

y\_min, y\_max = X[:, 1].min() - 1, X[:, 1].max() + 1

h = (x\_max / x\_min)/100

xx, yy = np.meshgrid(np.arange(x\_min, x\_max, h),

np.arange(y\_min, y\_max, h))

plt.subplot(1, 1, 1)

Z = svc.predict(np.c\_[xx.ravel(), yy.ravel()])

Z = Z.reshape(xx.shape)

plt.contourf(xx, yy, Z, cmap=plt.cm.Paired, alpha=0.8)

plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Paired)

plt.xlabel('Sepal length')

plt.ylabel('Sepal width')

plt.xlim(xx.min(), xx.max())

plt.title('SVC with linear kernel')

plt.show()

svc = svm.SVC(kernel='rbf', C=1,gamma=10).fit(X, y)

plt.subplot(1, 1, 1)

Z = svc.predict(np.c\_[xx.ravel(), yy.ravel()])

Z = Z.reshape(xx.shape)

plt.contourf(xx, yy, Z, cmap=plt.cm.Paired, alpha=0.8)

svc = svm.SVC(kernel='poly', C=1,gamma=100).fit(X, y)

plt.subplot(1, 1, 1)

Z = svc.predict(np.c\_[xx.ravel(), yy.ravel()])

Z = Z.reshape(xx.shape)

plt.contourf(xx, yy, Z, cmap=plt.cm.Paired, alpha=0.8)

3) Create a node.js file that Select all records from the "customers" table, and display the result object on console.

var mysql = require('mysql');

var con = mysql.createConnection({

host: 'localhost',

user: "root",

password: "Root@123",

database:'node'

});

con.connect(function(err) {

if (err) throw err;

console.log("Connected!");

});

con.query('SELECT \* FROM customers', (err,rows) => {

if(err) throw err;

console.log('Data received from Db:');

console.log(rows);

});

**Slip-10**

Q1 Write a Java Program to implement Strategy Pattern for Duck Behavior. Create instance variable that holds current state of Duck from there, we just need to handle all Flying Behaviors and Quack Behavior

Duck.Java

public abstract class Duck {

FlyBehaviour flyBehaviour;

QuackBehaviour quackBehaviour;

public Duck() {

}

public abstract void display();

public void performFly() {

flyBehaviour.fly();

}

public void performQuack() {

quackBehaviour.quack();

}

public void swim() {

System.out.println("All ducks float, even decoys!);

}

public void setFlyBehaviour(FlyBehaviour fb) {

flyBehaviour = fb;

}

public void setQuackBehaviour(QuackBehaviour qb) {

quackBehaviour qb;

}

}

MallardDuck.Java

public class MallardDuck extends Duck {

public MallardDuck() {

quackBehaviour = new Quack();

flyBehaviour = new FlyWithWings();

}

public void display() {

System.out.println("I'm a real Mallard duck");

}

}

ModelDuck.Java

public class ModolDuck extends Duck {

public ModelDuck() {

flyBehaviour = new FlyNoWay();

quackBehaviour = new Quack();

}

public void display() {

System.out.println("I'm a model duck");

}

}

FlyBehaviour.Java

public interface FlyBehaviour {

public void fly():

}

FlyWithWings.Java

public class FlyWithWings implements FlyBehaviour {

public void fly() {

System.out.println("I'm flying!!");

}

}

FlyNoWay.Java

public class FlyNoWay implements FlyBehaviour {

public void fly() {

System.out.printlin("I can't fly");

}

}

FlyRocketPowered.Java

public class FlyRocketPowered implements FlyBehaviour {

public void fly() {

System.out.println("I'm flying with a rocket!");

}

}

QuackBehaviour.Java

public interface QuackBehaviour {

public void quack() {

System.out.println("Quack");

}

}

Quack.Java

public class Quack implements QuackBehaviour {

public void quack() {

System.out.println("Quack");

}

}

MuteQuack.Java

public class MuteQuack implements QuackBehaviour {

public void quack() {

System.out.println("<< Silence >>");

}

}

Squeak.Java

public class Squeak implements QuackBehaviour {

public void quack() {

System.out.println("Squeak");

}

}

MiniDuckSimulator.Java

public class MiniDuckSimulator {

public static void main(String[] args) {

Duck mallard = new MallardDuck();

mallard.performQuack();

mallard.performFly();

}

}

MiniDuckSimulator.Java - Model Duck Edition

public class MiniDuckSimulator {

public static void main(String[] args) {

Duck mallard = new MallardDuck();

mallard.performQuack();

mallard.performFly();

Duck model = new ModelDuck();

model.performFly();

model.setFlyBehaviour(new FlyRocketPowered());

model.performFly();

}

}

2) Write a Python program to prepare Scatter Plot for Iris Dataset.

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

iris = pd.read\_csv("Iris.csv") # Reading the dataset “Iris.csv”.

print (iris.head(10)) # head() will display the top rows of the dataset, the default value of this function is 5,

#that is it will show top 5 rows when no argument is given to it.

plt.plot(iris.Id, iris["SepalLengthCm"],"r--")

plt.show #plt.show () will display the current figure that you are working on

iris.plot(kind ="scatter", x ='SepalLengthCm', y ='PetalLengthCm')

plt.grid() # grid () function to add grid lines to the plot

3) Create a node.js file that Insert Multiple Records in "student" table, and display the result object on console.

var mysql = require('mysql');

var con = mysql.createConnection({

host: "localhost",

user: "root",

password: "Root@123",

database: "node"

});

con.connect(function(err) {

if (err) throw err;

console.log("Connected!");

var sql = "INSERT INTO student (rollno,namee, percentage) VALUES ?";

var values = [

[1,'abc', 77.6],

[2,'def', 89.6],

[3,'ghi', 91.6]

];

con.query(sql, [values], function (err, result)

{

if (err) throw err;

console.log("Number of records inserted: " + result.affectedRows);

});

con.query("SELECT \* FROM student", function (err, result, fields) {

if (err) throw err;

console.log(result);

});

});