Slip 1

Q.1 Write a Java Program to implement I/O Decorator for converting uppercase letters to lower case letters.(SADP)

->

(LowercaseOutputStream)

```
package uppercase;

import java.io.FilterOutputStream;
import java.io.IOException;
import java.io.OutputStream;

public class LowercaseOutputStream extends FilterOutputStream
{
        public LowercaseOutputStream(OutputStream out)
        {
            super(out);
        }
          @Override
          public void write(int b) throws IOException
          {
             if (Character.isUpperCase(b))
             {
                b = Character.toLowerCase(b);
             }
             super.write(b);
          }
          @Override
          public void write(byte[] b, int off, int len) throws IOException
          {
             for (int i = off; i < off + len; i++)
```

```
            {
                if (Character.isUpperCase(b[i]))
                {
                    b[i] = (byte) Character.toLowerCase(b[i]);
                }
            }
            super.write(b, off, len);
        }
        public static void main(String[] args) throws IOException
        {
         LowercaseOutputStream lowercaseStream = new LowercaseOutputStream(System.out);
            String text = "HELLO WORLD!";
            byte[] bytes = text.getBytes();
            lowercaseStream.write(bytes);
            lowercaseStream.close();
        }
}
```

(FilterOutputStream)

```
package uppercase;

public interface FilterOutputStream {

}
```

Q.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
data = pd.read_csv("Iris.csv")
print (data.head(10))
x=data["sepal_length"]
```

```
y=data["petal_length"]

plt.scatter(x,y)

plt.show()
```

```
 import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.datasets import load_iris
iris = load_iris()
df= pd.DataFrame(data= np.c_[iris['data'], iris['target']],
columns= iris['feature_names'] + ['target'])
# select setosa and versicolor
y = df.iloc[0:100, 4].values
y = np.where(y == 'Iris-setosa', 0, 1)
# extract sepal length and petal length
X = df.iloc[0:100, [0, 2]].values
# plot data
plt.scatter(X[:50, 0], X[:50, 1],
color='blue', marker='o', label='Setosa')
plt.scatter(X[50:100, 0], X[50:100, 1],
color='green', marker='s', label='Versicolor')
plt.xlabel('Sepal length [cm]')
plt.ylabel('Petal length [cm]')
plt.legend(loc='upper left')
# plt.savefig('images/02_06.png', dpi=300)
plt.show()
```

Q.3 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>

<body>

<center>

<h1>Student Registration Form></h1>

<p>First Name:<input type="text"id="fn"></p>

<p>Last Name:<input type="text"id="ln"></p>

<p>Age:<input type="text"id="num"></p>

<p id="demo"></p>
```

```
<p>Address:<input type="text"id="add"></p>
<button type="button"onclick="validate()">VALIDATE</button>
</center>

<script>
	function validate()
	{
		var x,text;
		n1=document.getElementById("fn").value;
		n2=document.getElementById("ln").value;
		x=document.getElementById("num").value;
		var letters=/^[A-Za-z]+$/;
		if((!n1.match(letters))||(!n2.match(letters)))
		{
			alert("Invalid User Name");
		}
		else if(isNaN(x)||x<18||x>50)
		{
			document.getElementById("demo").innerHTML="Incorrect Age";
		}
		else
		{
			alert("Form Submitted");
		}
	}
</script>
</body>
</html>
```

Slip 2

Q.1 Write a Java Program to implement Singleton pattern for multithreading

->

```
package SingletonMultithreading;

public class SingletonMultithreading
{
	private static SingletonMultithreading
	INSTANCE=new SingletonMultithreading();
	private SingletonMultithreading()
	{

	}
	public static SingletonMultithreading getInstance()
	{
		return INSTANCE;
	}
}
```

```
package SingletonMultithreading;

public class SingletonMultithreading2
{
	private static SingletonMultithreading2 INSTANCE;
	private SingletonMultithreading2()
	{
	}
	public static SingletonMultithreading2 getInstance()
	{
		if(null==INSTANCE)
		{
			INSTANCE=new SingletonMultithreading2();
```

```
                }
                return INSTANCE;
        }
        public static void main(String[] args)
        {
                System.out.println("HELLO LAZY");
        }
}

package SingletonMultithreading;

public class SingletonMultithreading3
{
        private  static volatile SingletonMultithreading3 INSTANCE;
        private SingletonMultithreading3()
        {
        }
        public static SingletonMultithreading3 getInstance()
        {
                synchronized(SingletonMultithreading3.class)
                {
                        if(null==INSTANCE)
                        {
                                INSTANCE=new SingletonMultithreading3();
                        }
                        return INSTANCE;
                }
        }
        public static void main(String[] args)
        {
                System.out.println("HELLO WORLD");
```

```
	}
}
```

```
package SingletonMultithreading;

public class SingletonMultithreading4
{
	private static volatile SingletonMultithreading4 INSTANCE;
	private SingletonMultithreading4()
	{
	}
	public static SingletonMultithreading4 getInstance()
	{
		if(null==INSTANCE)
		{
			synchronized(SingletonMultithreading4.class)
			{
				if(null==INSTANCE)
				{
					INSTANCE= new SingletonMultithreading4();
				}
			}
		}
		return INSTANCE;
	}
}
```

Q.2 Write a python program to find all null values in a given dataset and remove them.

->

```
import pandas as pd
# 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)
```

```
 # Load dataset from CSV file
file_path = 'data.csv'  # Replace 'data.csv' with your file path
df = pd.read_csv(file_path)
# Display the dataset and null value counts before removal
print("Original dataset:")
print(df)
print("\nNull value counts:")
print(df.isnull().sum())
# Remove rows with any null values
df_cleaned = df.dropna()
# Display cleaned DataFrame
print("\nDataFrame after removing null values:")
print(df_cleaned)
```

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

```
<html>

<head>

<title>Employee Registration </title>

<script type="text/javascript">

        function valid()

{

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

        var birth_date=document.getElementById("BDate").value;

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

        var join_date=document.getElementById("JDate").value;
```

```
        if(/[a-zA-Z]/.test(name)==false)
        {
                alert("Enter the name.");
        }
        else if(birth_date==null)
        {
                alert("Enter the date of birth.");
        }
        else if(isNaN(salary)||salary<5000)
        {
                alert("Salary must be a number above 5000.");
        }
        else
        {
                alert("Employee Registration Submitted Successfully.");
        }
}
</script>
</head>
<body>
<center><b>
<h2>Employee Registration Form</h2>
<form action ="" onsubmit="return valid()" method="post">
Full Name: <input type="text" id="Name"><br><br>
Birthday: <input type="date" id="BDate"><br><br>
Salary: <input type="text" id="Salary"><br><br>
Joining Date: <input type="date" id="JDate"><br><br>
<input type="Submit" value="Submit"><br><br>
</form>
</b></center>
</body>
```

</html>

Slip 3

Q.1 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() [20 M]

->

```
package ObserverPattern;


public class WeatherStation
{
	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(80, 65, 30.4f);
		weatherData.setMeasurements(82, 70, 29.2f);
		weatherData.setMeasurements(78, 90, 29.2f);
	}
}


package ObserverPattern;


import java.util.ArrayList;


public class WeatherData implements Subject
{
```

```
private ArrayList<Observer> observers;
private float temperature;
private float humidity;
private float pressure;
public WeatherData()
{
	observers =new ArrayList<Observer>();
}
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=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;
	}
}
```

```
package ObserverPattern;

public interface Subject
{
	public void registerObserver(Observer o);
	public void removeObserver(Observer o);
	public void notifyObservers();
}
```

```
package ObserverPattern;

public 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("Avg/Max/Min temperature =" + (tempSum / numReadings)
                                + "/"+ maxTemp +"/"+ minTemp);
        }
}
```

```
package ObserverPattern;

public interface Observer
{
        public void update(float temp,float humidity,float pressure);
}
```

```
package ObserverPattern;

public 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);
        }
}

package ObserverPattern;

public 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.println("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, rainy weather");
		}
	}
}

package ObserverPattern;
```

```
public interface DisplayElement
{
	public void display();
}

package ObserverPattern;

public 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");
```

```
        }
}
```

Q. 2. Write a python program to make Categorical values in numeric format for a given dataset [20 M]

->

```
cars = pd.read_csv('data.csv')
print(cars.to_string())
ohe_cars = pd.get_dummies(cars[['Car']])
print(ohe_cars.to_string())
```

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

->

```
<html>
<head>
<title> Login Form Validation </title>
<script type = "text/javascript">
function valid()
{

        var name = document.ContactForm.name.value;
        var email = document.ContactForm.email.value;
        var pno = document.ContactForm.pno.value;
        var inputError = nameErr, mailErr, PhnoErr;
        if(/[a-zA-Z]/.test(name)==false)
        {
                alert("Please enter your name correctly.(Characters should be between a-z,A-Z)");
        }
        else if(/^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/.test(email)==false)
        {
                alert("Please enter valid email id.");
```

```
        }
        else if(/^\d{10}$/.test(pno)==false)
        {
                alert("Please enter valid phone number.(Phone number should be of 10 digits)");
        }
        else if((name||email||pno)==true)
         return false;
        else
        {
                var dataPreview = "You've entered the following details: \n"+
                  "Full Name: " + name + "\n"+
                  "Email ID: " + email + "\n"+
                  "Phone Number: " + pno + "\n";
        }
        alert(dataPreview);
}
</script>
</head>
<body>
<center>
<form name = "ContactForm" action = "" onsubmit = "return valid()" method = "post">
<h2>Login Form</h2>
<div class = "row">
<label>Full Name</label>
<input type = "text" name = "name">
<div class = "error" id = "nameErr"></div>
</div>
<div class = "row">
<label>Email ID</label>
<input type = "text" name = "email">
<div class = "error" id = "mailErr"></div>
```

```
</div>
<div class = "row">
<label>Phone Number</label>
<input type = "number" name = "pno">
<div class = "error" id = "PhnoErr"></div>
</div>
<div class = "row">
<input type = "Submit" name = "Submit">
</div>
</form>
</center>
</body>
</html>
```

Slip 4

Q.1 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. [20 M]

->

```
package FactoryPizza;

public class CheesePizza extends Pizza
{
	public CheesePizza()
	{
		name="Cheese Pizza";
		dough="Regular Crust";
		sauce="Marinara Pizza Sauce";
		toppings.add("Fresh Mozzarella");
		toppings.add("Parmesan");
	}
}
```

```
package FactoryPizza;

public class ClamPizza extends Pizza
{
	public ClamPizza()
	{
		name="Clam Pizza";
		dough="Thin Crust";
		sauce="White Garlic Sauce";
		toppings.add("Clams");
		toppings.add("Grated Parmesan Cheese");
	}
}

package FactoryPizza;

public class PepperoniPizza extends Pizza
{
	public PepperoniPizza()
	{
		name="Pepperoni Pizza";
		dough=" Crust";
		sauce="Marinara Sauce";
		toppings.add("Sliced Pepperoni");
		toppings.add("Sliced Onion");
		toppings.add("Grated Parmesan Cheese");
	}
}

package FactoryPizza;
```

```
import java.util.ArrayList;
public class Pizza
{
	String name;
	String dough;
	String sauce;
	ArrayList toppings=new ArrayList();
	public String getName()
	{
		return name;
	}
	public void prepare()
	{
		System.out.println("Preparing"+name);
	}
	public void bake()
	{
		System.out.println("Baking"+name);
	}
	public void cut()
	{
		System.out.println("Cutting"+name);
	}
	public void box()
	{
		System.out.println("Boxing"+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();
}
}

package FactoryPizza;

public class PizzaStore
{
        SimplePizzaFactory factory;
        public PizzaStore(SimplePizzaFactory factory)
        {
                this.factory=factory;
        }
        public Pizza orderPizza(String type)
        {
                Pizza pizza;
                pizza=factory.createPizza(type);
                pizza.prepare();
                pizza.bake();
                pizza.cut();
                pizza.box();
                return pizza;
        }
}
```

```
package FactoryPizza;

public class PizzaTestDrive
{
	public static void main(String[] args)
	{
		SimplePizzaFactory factory=new SimplePizzaFactory();
		PizzaStore store=new PizzaStore(factory);

		Pizza pizza=store.orderPizza("cheese");
		System.out.println("We ordered a"+pizza.getName()+"\n");

	   pizza=store.orderPizza("veggie");
		System.out.println("We ordered a "+pizza.getName()+"\n");
	}
}

package FactoryPizza;

public class SimplePizzaFactory
{
	public Pizza createPizza(String type)
	{
		Pizza pizza=null;
		if(type.equals("cheese"))
		{
			pizza=new CheesePizza();
		}
		else if(type.equals("pepperoni"))
		{
			pizza=new PepperoniPizza();
```

```
                }
                else if(type.equals("clam"))
                {
                        pizza=new ClamPizza();
                }
                else if(type.equals("veggie"))
                {
                        pizza=new VeggiePizza();
                }
                return pizza;
        }
}

package FactoryPizza;

public class VeggiePizza extends Pizza
{
        public VeggiePizza()
        {
                name="Veggie Pizza";
                dough=" Crust";
                sauce="Marinara Sauce";
                toppings.add("Shredded Mozzarella");
                toppings.add("Sliced Red Pepper");
                toppings.add("Sliced Black Olives");
                toppings.add("Sliced Mushrooms");
                toppings.add("Diced Onion");
                toppings.add("Grated Parmesan");
        }
}
```

Q. 2 Write a python program to Implement Simple Linear Regression for predicting house price. [20 M]

->

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

dataset=pd.read_csv('Salary.csv')
X= dataset.iloc[:,:-1].values
y=dataset.iloc[:,-1].values

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 1/3, random_state = 0)
from sklearn.linear_model import LinearRegression
regressor=LinearRegression()
regressor.fit(X_train,y_train)
y_pred = regressor.predict(X_test)

plt.scatter(X_train, y_train, color = 'red')
plt.plot(X_train, regressor.predict(X_train), color = 'blue')
plt.title('Salary vs Experience (Training set)')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.show()

plt.scatter(X_test, y_test, color = 'red')
plt.plot(X_train, regressor.predict(X_train), color = 'blue')
plt.title('Salary vs Experience (Test set)')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.show()
```

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

->

```
var http=require('http');
var uc=require('upper-case');
http.createServer(function (req,res) {
   res.writeHead(200,{'Content-Type':'text/html'});
   res.write(uc.upperCase("hello world!"));
   res.end();
}).listen(1234);
```

Slip5

Q.1 Write a Java Program to implement Adapter pattern for Enumeration iterator [20 M]

->

```
package EnumerationIterator;
import java.util.*;
public 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();
        }
}
```

```
package EnumerationIterator;

import java.util.Arrays;
import java.util.Iterator;
import java.util.*;

public class EnumerationIteratorTestDrive
{
        public static void main(String args[])
        {
                Vector v=new Vector();
                v.add("a");
                v.add("b");
                v.add("c");
                Iterator iterator=new EnumerationIterator(v.elements());
                while(iterator.hasNext())
                {
                        System.out.println(iterator.next());
                }
        }
}
```

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

[20 M]

->

```
# Multiple Linear Regression

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('50_Startups.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
print(X)

# Encoding categorical data
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)

# Splitting the dataset into the Training set and Test set
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, random_state = 0)

# Training the Multiple Linear Regression model on the Training set
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)

# Predicting the Test set results
```

```
y_pred = regressor.predict(X_test)
np.set_printoptions(precision=2)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))
```

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

->

```
var http=require('http');
var fs=require('fs');
var formidable=require('formidable');
http.createServer(function(req,res)
{
  if(req.url=='/')
  {
    res.writeHead(200,{'content-type':'text/html'});
    res.write('<form action="fapp" method="post" enctype="multipart/form-data">');
    res.write('<h1> SELECT TWO FILES</h1>');
    res.write('<input type="file" name="rf"><br>');
    res.write('<input type="file" name="wf"><br>');
    res.write('<input type="submit">');
    res.end();
  }
  else if(req.url=='/fapp')
  {
    var form=new formidable.IncomingForm();
    form.parse(req,function(err,fields,files)
    {
      if(!err)
      {
        var w=fs.createWriteStream(files.wf.name,{flags:'a'});
        var r=fs.createWriteStream(files.rf.name);
        w.on('close',function()
```

```
            {
                console.log("Writing Done");
            });
            r.pipe(w);
            res.write(files.rf.name);
            res.end("Appended Successfully");
          }
          else
          {
            res.write("Error in writing");
          }
       });
   }
   else
   {
      res.end("Page not found");
   }
}).listen(2830);
```

Slip 6

Q.1 Write a Java Program to implement command pattern to test Remote Control

[20 M]

->

```
package RemoteControl;


public interface Command
{
        void execute();


}
class LightOnCommand implements Command
{
        private Light light;
```

```
        public LightOnCommand(Light light)
        {
           this.light = light;
        }
        public void execute()
        {
           light.turnOn();
        }

}
class LightOffCommand implements Command
{
        private Light light;
        public LightOffCommand(Light light)
        {
           this.light = light;
        }
        public void execute()
        {
           light.turnOff();
        }

}
class Light
{
        public void turnOn()
        {
           System.out.println("Light is ON");
        }
        public void turnOff()
        {
```

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

}
class RemoteControl
{
    private Command command;
    public void setCommand(Command command)
    {
        this.command = command;
    }
    public void pressButton()
    {
        command.execute();
    }
}

package RemoteControl;

public class RemoteControlTest
{
    public static void main(String[] args)
    {
        Light light = new Light();
        LightOnCommand lightOn = new LightOnCommand(light);
        LightOffCommand lightOff = new LightOffCommand(light);
        RemoteControl remoteControl = new RemoteControl();
        remoteControl.setCommand(lightOn);
        remoteControl.pressButton();
        remoteControl.setCommand(lightOff);
        remoteControl.pressButton();
```

```
        }
}
```

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

[20 M]

->

```
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

from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(X, y)

from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree = 4)
X_poly = poly_reg.fit_transform(X)
lin_reg_2 = LinearRegression()
lin_reg_2.fit(X_poly, y)

plt.scatter(X, y, color = 'red')
plt.plot(X, lin_reg.predict(X), color = 'blue')
plt.title('Truth or Bluff (Linear Regression)')
plt.xlabel('Position Level')
plt.ylabel('Salary')
plt.show()
```

```
plt.scatter(X, y, color = 'red')
plt.plot(X, lin_reg_2.predict(poly_reg.fit_transform(X)), color = 'blue')
plt.title('Truth or Bluff (Polynomial Regression)')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()

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.plot(X_grid, lin_reg_2.predict(poly_reg.fit_transform(X_grid)), color = 'blue')
plt.title('Truth or Bluff (Polynomial Regression)')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()
```

Q.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 http=require('http');
var url=require('url');
var fs=require('fs');
http.createServer(function(req,res)
{
   var q=url.parse(req.url,true);
   var filename='.'+q.pathname;
   fs.readFile(filename,function(err,data)
   {
      if(err)
      {
         res.writeHead(404,{'Content-Type':'text/html'});
```

```
        return res.end('404 not found');
    }
    res.writeHead(200,{'Content-Type':'text/html'});
    res.write(data);
    return res.end();
  });
}).listen(2001);
```

Slip 7

Q.1 Write a Java Program to implement undo command to test Ceiling fan. [20 M]

->

```
package CommandPattern;

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 CeilingFan(CeilingFan fan) {}
        public void high() {
                speed = HIGH;
        }
        public void medium() {
```

```
		speed = MEDIUM;
	}
	public void low() {
		speed = OFF;
	}
	public void off() {
		speed = OFF;
	}
	public int getSpeed() {
		return speed;
	}
}

package CommandPattern;

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();
                }
        }
}

package CommandPattern;

public class CeilingFanMediumCommand implements Command
{
        public CeilingFanMediumCommand(CeilingFan ceilingFan) {}
        public void CeilingFanMediumCommand(CeilingFan ceilingFan) {}
        @Override
        public void execute() {}
        @Override
        public void undo()
        {

        }
}

package CommandPattern;

public class CeilingFanOffCommand {
```

```
}
package CommandPattern;

public interface Command
{
	public void execute();
	public void undo();
}

package CommandPattern;

public class NoCommand implements Command
{
	public void execute(){}
	public void undo() {}
}

package CommandPattern;

public class RemoteControl
{
	Command[] onCommand;
	Command[] offCommand;
	Command undoCommand;

	public RemoteControl() {

		onCommand = new Command[7];
		offCommand = new Command[7];
		Command noCommand = new NoCommand();
```

```
		for(int i=0; i<7; i++) {
			onCommand[i] = noCommand;
			offCommand[i] = noCommand;
		}
		undoCommand = noCommand;
	}
	public void setCommand(int slot, Command ceilingFanMedium, Command ceilingFanOff) {
		onCommand[slot] = ceilingFanMedium;
		offCommand[slot] = ceilingFanOff;
	}
	public void onButtonWasPushed(int slot) {
		onCommand[slot].execute();
		undoCommand = onCommand[slot];
	}
	public void offButtonWasPushed(int slot) {
		offCommand[slot].execute();
		undoCommand = offCommand[slot];
	}
	public void undoButtonWasPushed() {
		undoCommand.undo();
	}
	public void setCommand(int slot, CeilingFanMediumCommand ceilingFanMedium, CeilingFan
ceilingFanOff)
	{


	}
	public void setCommand(int slot, CeilingFanHighCommand ceilingFanHigh, CeilingFan
ceilingFanOff)
	{


	}
}
```

```
package CommandPattern;


public class RemoteLoader
{
public static void main(String[] args) {

		RemoteControl remoteControl = new RemoteControl();


		System.out.println("Living Room");
		CeilingFan ceilingFan = new CeilingFan("Living Room");


		CeilingFanMediumCommand ceilingFanMedium = new
CeilingFanMediumCommand(ceilingFan);
		System.out.println("Fan speed is medium");


		CeilingFanHighCommand ceilingFanHigh = new CeilingFanHighCommand(ceilingFan);
		System.out.println("Fan speed is high");


		CeilingFan ceilingFanOFF = new CeilingFan(ceilingFan);
		System.out.println("Fan is turned off");
	}
}
```

Q.2. Write a python program to implement Naive Bayes. [20 M]

->

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd


dataset = pd.read_csv('Social_Network_Ads.csv')
```

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

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
print(X_train)
print(y_train)
print(X_test)
print(y_test)

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
print(X_train)
print(X_test)

from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)

print(classifier.predict(sc.transform([[30,87000]])))

y_pred = classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))

from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

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

->

```
var http=require('http');
var fs=require('fs');
var formidable=require('formidable');
http.createServer(function(req,res)
{
   if(req.url=='/')
   {
      res.writeHead(200,{'content-type':'text/html'});
      res.write('<form action="fapp" method="post" enctype="multipart/form-data">');
      res.write('<h1> Registration Form</h1>');
      res.write('Application Name:<input type="text" name="t1"><br>');
      res.write('Phone:<input type="text" name="t2"><br>');
      res.write('Address:<input type="text" name="t3"><br>');
      res.write('Resume upload:<input type="file" name="filetoupload"><br>');
      res.write('<input type="submit" name="upload"><br>');
      res.end();
   }
   else if(req.url=='/fapp')
   {
      var form=new formidable.IncomingForm();
      form.parse(req,function(err,fields,files)
      {
         res.write('<h1>Name:'+fields.t1+'</h1>');
         res.write('<h1>Phone:'+fields.t2+'</h1>');
         res.write('<h1>Address:'+fields.t3+'</h1>');
         var oldpath=files.filetoupload.path;
         var newpath='C:/MSC/'+files.filetoupload.name;
         fs.rename(oldpath,newpath,function(err)
```

```
        {
            if(err) throw err;
            else{
                res.write('Resume uploaded and moved successfully');
                res.end();
            }
        });
    });
  }
  else{
     res.end("Page not found");
  }
}).listen(3400);
```

Slip 8

Q. 1 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 [20 M]

->

```
package StatePattern;


public class GumballDemo
{
        public static void main(String args[])throws IllegalMonitorStateException
        {
                try
                {
                        GumballMachine gum=new GumballMachine(4);
                        gum.insertQuarter();
                        gum.ejectQuarter();
                        gum.equals(gum);
                        gum.notify();
```

```
                    gum.notifyAll();
                    gum.refill(0);
                    gum.toString();
                }
                catch(IllegalMonitorStateException e)
                {
                    System.out.println(false);
                }
        }
}

package StatePattern;

public class GumballMachine
{
        final static int SOLD_OUT=0;
        final static int NO_QUARTER=1;
        final static int HAS_QUARTER=2;
        final static int SOLD=3;
        int state=SOLD_OUT;
        int count=0;
        public GumballMachine(int count)
        {
                this.count=count;
                if(count>0)
                {
                        state=NO_QUARTER;
                }
        }
        public void insertQuarter()
        {
```

```
        if(state==HAS_QUARTER)
        {
                System.out.println("You cant insert another quarter");
        }
        else if(state==NO_QUARTER)
        {
                state=HAS_QUARTER;
                System.out.println("You inserted a quarter");
        }
        else if(state==SOLD_OUT)
        {
                System.out.println("You cant insert a quarter,the machine is sold out");
        }
        else if(state==SOLD)
        {
                System.out.println("Please wait, we're already giving you a gumball");
        }
}
public void ejectQuarter()
{
        if(state==HAS_QUARTER)
        {
                System.out.println("Quarter returned");
                state=NO_QUARTER;
        }
        else if(state==NO_QUARTER)
        {
                System.out.println("You haven't inserted a quarter");
        }
        else if(state==SOLD)
        {
```

```
                System.out.println("Sorry,you already turned the crank");
        }
        else if(state==SOLD_OUT)
        {
                System.out.println("You cant eject,you haven't inserted a quarter yet");
        }
}
public void turnCrank()
{
        if(state==SOLD)
        {
                System.out.println("Turning twice doesn't get you another gumball");
        }
        else if(state==NO_QUARTER)
        {
                System.out.println("You turned but there's no quarter");
        }
        else if(state==SOLD_OUT)
        {
                System.out.println("You turned but there are no gumballs");
        }
        else if(state==HAS_QUARTER)
        {
                System.out.println("No gumball dispensed");
        }
}
public void refill(int numGumBalls)
{
        this.count=numGumBalls;
        state=NO_QUARTER;
}
```

```
	public String toString()
	{
		StringBuffer result=new StringBuffer();
		result.append("\n Mighty Gumball Inc.");
		result.append("\n Java-enabled Standing Gumball Model #2004\n");
		result.append("Inventory:"+count+"gumball");
		if(count!=1)
		{
			result.append("s");
		}
		result.append("\n Machine is");
		if(state==SOLD_OUT)
		{
			result.append("sold out");
		}
		else if(state==NO_QUARTER)
		{
			result.append("waiting for quarter");		}
		else if(state==HAS_QUARTER)
		{
			result.append("waiting for turn of crank");
		}
		else if(state==SOLD)
		{
			result.append("delivering a gumball");
		}
		result.append("\n");
		return result.toString();
	}
}
```

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

[20 M]

```
-import numpy as np
import matplotlib.pyplot as plt
import pandas as pd>

dataset = pd.read_csv('Social_Network_Ads.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
print(X_train)
print(y_train)
print(X_test)
print(y_test)

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
print(X_train)
print(X_test)
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
classifier.fit(X_train, y_train)

print(classifier.predict(sc.transform([[30,87000]])))

y_pred = classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))
```

```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

Q. 3 Create a Node.js file that demonstrates create database and table in MySQL. [20 M]

->

```
var mysql=require('mysql');
var con=mysql.createConnection({
   host:"localhost",
   user:"root",
   password:"Msc@student2"
});
con.connect(function(err){
   if(err)throw err;
   console.log("connected");
   con.query("CREATE DATABASE inventory",function(err,result)
   {
      if(err) throw err;
      else{
         console.log("Database created");
         con.query("use inventory");
         var sql="CREATE TABLE customers(cid INT,cname VARCHAR(25),phone INT(10),City
VARCHAR(30))";
         con.query(sql,function(err,result)
         {
            if(err) throw err;
            console.log("Table Created");
         })
      }
   })
```

})

Slip 9

Q.1 Design simple HR Application using Spring Framework [20 M]

->

Q 2. Write a python program to implement Linear SVM. [20 M]

->

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

dataset = pd.read_csv('Social_Network_Ads.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
print(X_train)
print(y_train)
print(X_test)
print(y_test)

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
print(X_train)
print(X_test)

from sklearn.svm import SVC
classifier = SVC(kernel = 'linear', random_state = 0)
```

```
classifier.fit(X_train, y_train)


print(classifier.predict(sc.transform([[30,87000]])))


y_pred = classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))


from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

Q. 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:"satara@123", database: "mydb"
});
con.connect(function(err){ if (err) throw err;
con.query("select name , address from Customer",function(err,result,fields){ if (err) throw err;
console.log(result);
});
});
```

Slip 10

Q.1 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 [20 M]

->

Q. 2 Write a Python program to prepare Scatter Plot for Iris Dataset. [20 M]

->

```
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()
```

Q. 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:"satara@123", database:"mydb"
}); con.connect(function(err){ if (err) throw err; console.log("Connected!!!"); var sql="INSERT INTO Student (name,address) Values ?"; var values=[
['Pritam','Highway 71'],
['Sneha','Lowstreet 4'],
['Sid','Apple st 652'],
['Sam','Valley 345'],
['Michael','Green Greass 1'],
['Griss','One way 98'],
['Richard','Sky st 331']
]; con.query(sql, [values],function(err,result){ if (err) throw err;
console.log("Number of records inserted:" + result.affectedRows);
});
});
```

Slip 11

Q.1 Write a java program to implement Adapter pattern to design Heart Model to Beat Model [20 M]

->

Q2. Write a python program to find all null values in a given dataset and remove them. [20 M]

->

```
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)
```

Q.3 Create a node.js file that Select all records from the "customers" table, and delete the specified record.

-> var mysql= require('mysql'); var con= mysql.createConnection({ host:"localhost", user:"root", password:"satara@123", database: "mydb"

});

con.connect(function(err){ if (err) throw err;

con.query("select name , address from Customer",function(err,result,fields){ if (err) throw err;
console.log(result);

}); }); var sql = "DELETE FROM Customer WHERE address='Valley 345'";
con.query(sql,function(err,result){ if (err) throw err;

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

});

Slip 12

Q.1 Write a Java Program to implement Decorator Pattern for interface Car to define the

package decorator;

```
public class BasicCar implements Car
{
	public void assemble()
	{
		System.out.println("Basic Car");
	}
}

package decorator;

public interface Car
{
	public void assemble();
}

package decorator;

public class CarDecorator implements Car
{
	protected Car car;
	public CarDecorator(Car c)
	{
		this.car=c;
	}
	public void assemble()
	{
		this.car.assemble();
	}
}
package decorator;
```

```
public class DecoratorDemo
{
	public static void main(String[] args)
	{
		Car sportsCar=new SportsCar(new BasicCar());
		sportsCar.assemble();
		System.out.println("\n*************");
		Car sportsLuxuryCar=new SportsCar(new LuxuryCar(new BasicCar()));
		sportsLuxuryCar.assemble();
	}
}
```

```
package decorator;

public class LuxuryCar extends CarDecorator
{
	public LuxuryCar(Car c)
	{
		super(c);
	}
	public void assemble()
	{
		super.assemble();
		System.out.println("Adding Features of Luxury Car");
	}
}
```

```
package decorator;

public class SportsCar extends CarDecorator
```

```
{
	public SportsCar(Car c)
	{
		super(c);
	}
	public void assemble()
	{
		super.assemble();
		System.out.println("Adding features of Sports Car");
	}
}
```

Q. 2 Write a python program to make Categorical values in numeric format for a given dataset [20 M]

->

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
# Load dataset from CSV file
file_path = 'your_dataset.csv' # Replace 'your_dataset.csv' with your file path
df = pd.read_csv(file_path)
# Display the dataset before
encoding
print("Original dataset:")
print(df)
# Initialize LabelEncoder
label_encoder = LabelEncoder()
# Iterate through each column and check if it's categorical for encoding
for col in df.columns:
if df[col].dtype == 'object': # If the column contains categorical data
df[col + '_Encoded'] = label_encoder.fit_transform(df[col])
# Display the dataset after encoding categorical columns
```

```
print("

nEncoded dataset:")

print(df)
```

Q.3 Create a Simple Web Server using node js

```
-> var mysql= require('mysql'); var con= mysql.createConnection({ host:"localhost",

user:"root", password:"satara@123", database: "mydb"

});

con.connect(function(err){ if (err) throw err;

con.query("select name , address from Customer",function(err,result,fields){ if (err) throw err;
console.log(result);

}); }); var sql = "DELETE FROM Customer WHERE address='Valley 345'";
con.query(sql,function(err,result){ if (err) throw err; console.log("Number of records Deleted:" +

result.affectedRows); });

var http=require('http');

var server=http.createServer(function(req,res){

   res.write("HELLO WELCOME TO ATSS COLLEGE");

   res.end("end");

});

server.listen(5000);

console.log('Node.js web server at port 5000 is running..')
```

Slip 14

Q.1 Write a Java Program to implement Command Design Pattern for Command Interface

with execute() . Use this to create variety of commands for LightOnCommand,

LightOffCommand, GarageDoorUpCommand, StereoOnWithCDComman. [20 M]

->

```
package RemoteControl;


public interface Command

{

        void execute();
```

```
}
class LightOnCommand implements Command
{
        private Light light;
        public LightOnCommand(Light light)
        {
           this.light = light;
        }
        public void execute()
        {
           light.turnOn();
        }

}
class LightOffCommand implements Command
{
        private Light light;
        public LightOffCommand(Light light)
        {
           this.light = light;
        }
        public void execute()
        {
           light.turnOff();
        }

}
class Light
{
        public void turnOn()
```

```
        {
            System.out.println("Light is ON");
        }
        public void turnOff()
        {
            System.out.println("Light is OFF");
        }

}
class RemoteControl
{
        private Command command;
        public void setCommand(Command command)
        {
            this.command = command;
        }
        public void pressButton()
        {
            command.execute();
        }
}

package RemoteControl;

public class RemoteControlTest
{
        public static void main(String[] args)
        {
             Light light = new Light();
             LightOnCommand lightOn = new LightOnCommand(light);
             LightOffCommand lightOff = new LightOffCommand(light);
```

```
        RemoteControl remoteControl = new RemoteControl();
        remoteControl.setCommand(lightOn);
        remoteControl.pressButton();
        remoteControl.setCommand(lightOff);
        remoteControl.pressButton();
    }
}
```

Q.2. Write a python program to find all null values in a given dataset and remove them. [20 M]

->

```
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)
```

Q.3 Write node js script to interact with the filesystem, and serve a web page from a file .

-> var http = require('http'); var url = require('url'); var fs =

require('fs'); http.createServer(function (req, res) { var pathname = url.parse(req.url, true).pathname; console.log("Request for" + pathname + "received."); fs.readFile(pathname.substr(1), function (err, data) {

if (err) { console.log(err); res.writeHead(404, { 'content-type': 'text/html' });

res.end('<html><body><h1>404 Not found</h1></body></html>');

}

else {

```
'content-type': 'text/html' });
res.writeHead(200, {
res.write(data);
res.end();
}
});
}).listen(9030); console.log('server is
running on port
```

Index.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<title>Sample Page</title>
</head>
<body>
Hello World! Welcome to web module.
</body>
</html>
```

Slip 17

Q.1 Write a Java Program to implement Abstract Factory Pattern for Shape interface.
[20 M].

->

```
package AbstractFactory;

public abstract class AbstractFactory
{
	abstract Color getColor(String color);
	abstract Shape getShape(String shape);
}
```

```
package AbstractFactory;

public class AbstractFactoryDemo
{
public static void main(String[] args) {

		AbstractFactory shapeFactory = FactoryProducer.getFactory("SHAPE");
		Shape shape1 = shapeFactory.getShape("CIRCLE");
		shape1.draw();
		Shape shape2 = shapeFactory.getShape("RECTANGLE");
		shape2.draw();
		Shape shape3 = shapeFactory.getShape("SQUARE");
		shape3.draw();

		AbstractFactory colorFactory = FactoryProducer.getFactory("COLOR");
		Color color1 = colorFactory.getColor("RED");
		color1.fill();
		Color color2 = colorFactory.getColor("GREEN");
		color2.fill();
		Color color3 = colorFactory.getColor("BLUE");
		color3.fill();

	}
}

package AbstractFactory;

public class Blue implements Color{

	@Override
```

```
	public void fill()
	{
		System.out.println("Inside Blue :: fill() method.");
	}


}
```

```
package AbstractFactory;

public class Circle implements Shape{

	@Override
	public void draw()
	{
		System.out.println("Inside Circle :: draw() method.");
	}
}
```

```
package AbstractFactory;

public interface Color
{
	void fill();
}
```

```
package AbstractFactory;

public class ColorFactory extends AbstractFactory{

	@Override
```

```
	Shape getShape(String shapeType) {
		return null;
	}
	@Override
	Color getColor(String color) {
		if(color == null) return null;
		else if(color.equalsIgnoreCase("RED")) return new Red();
		else if(color.equalsIgnoreCase("GREEN")) return new Green();
		else if(color.equalsIgnoreCase("BLUE")) return new Blue();
		return null;
	}

}

package AbstractFactory;

public class FactoryProducer
{
	public static AbstractFactory getFactory(String choice) {
		if(choice.equalsIgnoreCase("SHAPE")) return new ShapeFactory();
		else if(choice.equalsIgnoreCase("COLOR")) return new ColorFactory();
		return null;
	}
}

package AbstractFactory;

public class Green implements Color{

		@Override
		public void fill()
```

```
		{
			System.out.println("Inside Green :: fill() method.");
		}

}

package AbstractFactory;

public class Rectangle implements Shape{

	@Override
	public void draw()
	{
		System.out.println("Inside Rectangle :: draw() method.");
	}

}

package AbstractFactory;

public class Red implements Color
{
	@Override
	public void fill()
	{
		System.out.println("Inside Red :: fill() method.");
	}
}

package AbstractFactory;
```

```
public interface Shape
{
	void draw();
}

package AbstractFactory;

public class ShapeFactory extends AbstractFactory{

	@Override
	Color getColor(String color) {
		return null;
	}
	@Override
	public Shape getShape(String shapeType) {
		if(shapeType == null) return null;
		else if(shapeType.equalsIgnoreCase("CIRCLE")) return new Circle();
		else if(shapeType.equalsIgnoreCase("RECTANGLE")) return new Rectangle();
		else if(shapeType.equalsIgnoreCase("SQUARE")) return new Square();
		return null;
	}

}

package AbstractFactory;

public class Square implements Shape {

	@Override
	public void draw()
	{
```

```
                System.out.println("Inside Square :: draw() method.");
        }

}
```

Q.2. Write a python program to implement Multiple Linear Regression for a given dataset. [20 M]

-> # Multiple Linear Regression

```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('50_Startups.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
print(X)

# Encoding categorical data
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)

# Splitting the dataset into the Training set and Test set
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, random_state = 0)

# Training the Multiple Linear Regression model on the Training set
from sklearn.linear_model import LinearRegression
```

```
regressor = LinearRegression()

regressor.fit(X_train, y_train)


# Predicting the Test set results

y_pred = regressor.predict(X_test)

np.set_printoptions(precision=2)

print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))
```

Q.3 Write node js application that transfer a file as an attachment on web and enables browser to prompt the user to download file using express js.

-> var express =

require('express'); var app = express(); var PORT = 3000; window = {}; app.get('/', function(req, res){

res.download('Hello.txt');

});

app.listen(PORT, function(err){

if

(err) console.log(err); console.log("Server

listening on PORT", PORT);

});

Slip 18

Q.1 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() [20 M]

->->

```
package ObserverPattern;


public class WeatherStation

{

	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(80, 65, 30.4f);
                weatherData.setMeasurements(82, 70, 29.2f);
                weatherData.setMeasurements(78, 90, 29.2f);
        }
}

package ObserverPattern;

import java.util.ArrayList;

public class WeatherData implements Subject
{
        private ArrayList<Observer> observers;
        private float temperature;
        private float humidity;
        private float pressure;
        public WeatherData()
        {
                observers =new ArrayList<Observer>();
        }
        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=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;
	}
}
```

```
package ObserverPattern;

public interface Subject
{
	public void registerObserver(Observer o);
	public void removeObserver(Observer o);
	public void notifyObservers();
}
```

```
package ObserverPattern;

public 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("Avg/Max/Min temperature =" + (tempSum / numReadings)
                                + "/"+ maxTemp +"/"+ minTemp);
        }
}

package ObserverPattern;

public interface Observer
{
```

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



package ObserverPattern;


public 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);
        }
}

package ObserverPattern;

public 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.println("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, rainy weather");
                }
        }
}

package ObserverPattern;

public interface DisplayElement
{
        public void display();
}

package ObserverPattern;

public 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");
	}
}
```

Q.2. Write a python program to implement Polynomial Linear Regression for given dataset [20 M]

->

```
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

from sklearn.linear_model import LinearRegression
```

```
lin_reg = LinearRegression()
lin_reg.fit(X, y)

from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree = 4)
X_poly = poly_reg.fit_transform(X)
lin_reg_2 = LinearRegression()
lin_reg_2.fit(X_poly, y)

plt.scatter(X, y, color = 'red')
plt.plot(X, lin_reg.predict(X), color = 'blue')
plt.title('Truth or Bluff (Linear Regression)')
plt.xlabel('Position Level')
plt.ylabel('Salary')
plt.show()

plt.scatter(X, y, color = 'red')
plt.plot(X, lin_reg_2.predict(poly_reg.fit_transform(X)), color = 'blue')
plt.title('Truth or Bluff (Polynomial Regression)')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()

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.plot(X_grid, lin_reg_2.predict(poly_reg.fit_transform(X_grid)), color = 'blue')
plt.title('Truth or Bluff (Polynomial Regression)')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()
```

Q.3 Create your Django app in which after running the server, you should see on the browser, the text “Hello! I am learning Django”, which you defined in the index view.

->

Slip 19

Q.1 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. [20 M]

-> package FactoryPizza;

```
public class CheesePizza extends Pizza
{
	public CheesePizza()
	{
		name="Cheese Pizza";
		dough="Regular Crust";
		sauce="Marinara Pizza Sauce";
		toppings.add("Fresh Mozzarella");
		toppings.add("Parmesan");
	}
}
```

```
package FactoryPizza;

public class ClamPizza extends Pizza
{
	public ClamPizza()
	{
		name="Clam Pizza";
		dough="Thin Crust";
```

```
                sauce="White Garlic Sauce";
                toppings.add("Clams");
                toppings.add("Grated Parmesan Cheese");
        }
}

package FactoryPizza;

public class PepperoniPizza extends Pizza
{
        public PepperoniPizza()
        {
                name="Pepperoni Pizza";
                dough=" Crust";
                sauce="Marinara Sauce";
                toppings.add("Sliced Pepperoni");
                toppings.add("Sliced Onion");
                toppings.add("Grated Parmesan Cheese");
        }
}

package FactoryPizza;
import java.util.ArrayList;
public class Pizza
{
        String name;
        String dough;
        String sauce;
        ArrayList toppings=new ArrayList();
        public String getName()
        {
```

```
        return name;
    }
    public void prepare()
    {
        System.out.println("Preparing"+name);
    }
    public void bake()
    {
        System.out.println("Baking"+name);
    }
    public void cut()
    {
        System.out.println("Cutting"+name);
    }
    public void box()
    {
        System.out.println("Boxing"+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();
}
}
```

```
package FactoryPizza;

public class PizzaStore
{
	SimplePizzaFactory factory;
	public PizzaStore(SimplePizzaFactory factory)
	{
		this.factory=factory;
	}
	public Pizza orderPizza(String type)
	{
		Pizza pizza;
		pizza=factory.createPizza(type);
		pizza.prepare();
		pizza.bake();
		pizza.cut();
		pizza.box();
		return pizza;
	}
}

package FactoryPizza;

public class PizzaTestDrive
{
	public static void main(String[] args)
	{
		SimplePizzaFactory factory=new SimplePizzaFactory();
		PizzaStore store=new PizzaStore(factory);
```

```
                Pizza pizza=store.orderPizza("cheese");
                System.out.println("We ordered a"+pizza.getName()+"\n");

            pizza=store.orderPizza("veggie");
                System.out.println("We ordered a "+pizza.getName()+"\n");
        }
}

package FactoryPizza;

public class SimplePizzaFactory
{
        public Pizza createPizza(String type)
        {
                Pizza pizza=null;
                if(type.equals("cheese"))
                {
                        pizza=new CheesePizza();
                }
                else if(type.equals("pepperoni"))
                {
                        pizza=new PepperoniPizza();
                }
                else if(type.equals("clam"))
                {
                        pizza=new ClamPizza();
                }
                else if(type.equals("veggie"))
                {
                        pizza=new VeggiePizza();
                }
```

```
                return pizza;
        }
}

package FactoryPizza;

public class VeggiePizza extends Pizza
{
        public VeggiePizza()
        {
                name="Veggie Pizza";
                dough=" Crust";
                sauce="Marinara Sauce";
                toppings.add("Shredded Mozzarella");
                toppings.add("Sliced Red Pepper");
                toppings.add("Sliced Black Olives");
                toppings.add("Sliced Mushrooms");
                toppings.add("Diced Onion");
                toppings.add("Grated Parmesan");
        }
}
```

Q.2. Write a python program to implement Naive Bayes. [20 M]

->

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

dataset = pd.read_csv('Social_Network_Ads.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
print(X_train)
print(y_train)
print(X_test)
print(y_test)

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
print(X_train)
print(X_test)

from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)

print(classifier.predict(sc.transform([[30,87000]])))

y_pred = classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))

from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

Q.3 Design a Django application that adds web pages with views and templates.

Slip 25

Q.1 Write a Java Program to implement Singleton pattern for multithreading [20 M]

->

```
package SingletonMultithreading;

public class SingletonMultithreading
{
	private static SingletonMultithreading
	INSTANCE=new SingletonMultithreading();
	private SingletonMultithreading()
	{

	}
	public static SingletonMultithreading getInstance()
	{
		return INSTANCE;
	}
}
```

```
package SingletonMultithreading;

public class SingletonMultithreading2
{
	private static SingletonMultithreading2 INSTANCE;
	private SingletonMultithreading2()
	{
	}
	public static SingletonMultithreading2 getInstance()
	{
		if(null==INSTANCE)
		{
			INSTANCE=new SingletonMultithreading2();
```

```
                }
                return INSTANCE;
        }
        public static void main(String[] args)
        {
                System.out.println("HELLO LAZY");
        }
}

package SingletonMultithreading;

public class SingletonMultithreading3
{
        private  static volatile SingletonMultithreading3 INSTANCE;
        private SingletonMultithreading3()
        {
        }
        public static SingletonMultithreading3 getInstance()
        {
                synchronized(SingletonMultithreading3.class)
                {
                        if(null==INSTANCE)
                        {
                                INSTANCE=new SingletonMultithreading3();
                        }
                        return INSTANCE;
                }
        }
        public static void main(String[] args)
        {
                System.out.println("HELLO WORLD");
```

```
	}
}
```

```
package SingletonMultithreading;

public class SingletonMultithreading4
{
	private static volatile SingletonMultithreading4 INSTANCE;
	private SingletonMultithreading4()
	{
	}
	public static SingletonMultithreading4 getInstance()
	{
		if(null==INSTANCE)
		{
			synchronized(SingletonMultithreading4.class)
			{
				if(null==INSTANCE)
				{
					INSTANCE= new SingletonMultithreading4();
				}
			}
		}
		return INSTANCE;
	}
}
```

Q.2. Write a python program to Implement Simple Linear Regression for predicting house price. [20 M]

```
-> import numpy as np
import matplotlib.pyplot as plt
```

```
import pandas as pd

dataset=pd.read_csv('Salary.csv')
X= dataset.iloc[:,:-1].values
y=dataset.iloc[:,-1].values

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 1/3, random_state = 0)
from sklearn.linear_model import LinearRegression
regressor=LinearRegression()
regressor.fit(X_train,y_train)
y_pred = regressor.predict(X_test)

plt.scatter(X_train, y_train, color = 'red')
plt.plot(X_train, regressor.predict(X_train), color = 'blue')
plt.title('Salary vs Experience (Training set)')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.show()

plt.scatter(X_test, y_test, color = 'red')
plt.plot(X_train, regressor.predict(X_train), color = 'blue')
plt.title('Salary vs Experience (Test set)')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.show()
```

Q.3 Create a Simple Web Server using node js.

```
-> var mysql= require('mysql'); var con= mysql.createConnection({ host:"localhost",
user:"root", password:"satara@123", database: "mydb"
```

```
});

con.connect(function(err){ if (err) throw err;

con.query("select name , address from Customer",function(err,result,fields){ if (err) throw err;
console.log(result);

}); }); var sql = "DELETE FROM Customer WHERE address='Valley 345'";
con.query(sql,function(err,result){ if (err) throw err; console.log("Number of records Deleted:" +

result.affectedRows); });
```

Slip 26

Q.1 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. [20 M]

->

Q.2. Write a python program to implement Multiple Linear Regression for given dataset. [20 M]

->

```
# Multiple Linear Regression


# Importing the libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd


# Importing the dataset

dataset = pd.read_csv('50_Startups.csv')

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

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

print(X)


# Encoding categorical data

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)

# Splitting the dataset into the Training set and Test set
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, random_state = 0)

# Training the Multiple Linear Regression model on the Training set
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)

# Predicting the Test set results
y_pred = regressor.predict(X_test)
np.set_printoptions(precision=2)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))
```

Q.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: "satara@123"
}); con.connect(function(err){ if (err) throw err; console.log("Connected!!"); con.query("create
DATABASE mydb",function(err,result){ if (err) throw err; console.log("Database created");
});
});
```

Slip 27

Q.1 Write a Java Program to implement Abstract Factory Pattern for Shape interface.

[20 m]

->

```
package AbstractFactory;

public abstract class AbstractFactory
{
	abstract Color getColor(String color);
	abstract Shape getShape(String shape);
}

package AbstractFactory;

public class AbstractFactoryDemo
{
public static void main(String[] args) {

		AbstractFactory shapeFactory = FactoryProducer.getFactory("SHAPE");
		Shape shape1 = shapeFactory.getShape("CIRCLE");
		shape1.draw();
		Shape shape2 = shapeFactory.getShape("RECTANGLE");
		shape2.draw();
		Shape shape3 = shapeFactory.getShape("SQUARE");
		shape3.draw();

		AbstractFactory colorFactory = FactoryProducer.getFactory("COLOR");
		Color color1 = colorFactory.getColor("RED");
		color1.fill();
		Color color2 = colorFactory.getColor("GREEN");
		color2.fill();
		Color color3 = colorFactory.getColor("BLUE");
		color3.fill();

	}
```

```
}
```

```
package AbstractFactory;

public class Blue implements Color{

	@Override
	public void fill()
	{
		System.out.println("Inside Blue :: fill() method.");
	}


}
```

```
package AbstractFactory;

public class Circle implements Shape{

	@Override
	public void draw()
	{
		System.out.println("Inside Circle :: draw() method.");
	}
}
```

```
package AbstractFactory;

public interface Color
{
	void fill();
```

```
}

package AbstractFactory;

public class ColorFactory extends AbstractFactory{

	@Override
	Shape getShape(String shapeType) {
		return null;
	}
	@Override
	Color getColor(String color) {
		if(color == null) return null;
		else if(color.equalsIgnoreCase("RED")) return new Red();
		else if(color.equalsIgnoreCase("GREEN")) return new Green();
		else if(color.equalsIgnoreCase("BLUE")) return new Blue();
		return null;
	}

}

package AbstractFactory;

public class FactoryProducer
{
	public static AbstractFactory getFactory(String choice) {
		if(choice.equalsIgnoreCase("SHAPE")) return new ShapeFactory();
		else if(choice.equalsIgnoreCase("COLOR")) return new ColorFactory();
		return null;
	}
}
```

```
package AbstractFactory;

public class Green implements Color{

		@Override
		public void fill()
		{
			System.out.println("Inside Green :: fill() method.");
		}

}
```

```
package AbstractFactory;

public class Rectangle implements Shape{

	@Override
	public void draw()
	{
		System.out.println("Inside Rectangle :: draw() method.");
	}

}
```

```
package AbstractFactory;

public class Red implements Color
{
	@Override
	public void fill()
```

```
	{
		System.out.println("Inside Red :: fill() method.");
	}
}
```

```
package AbstractFactory;

public interface Shape
{
	void draw();
}
```

```
package AbstractFactory;

public class ShapeFactory extends AbstractFactory{

	@Override
	Color getColor(String color) {
		return null;
	}
	@Override
	public Shape getShape(String shapeType) {
		if(shapeType == null) return null;
		else if(shapeType.equalsIgnoreCase("CIRCLE")) return new Circle();
		else if(shapeType.equalsIgnoreCase("RECTANGLE")) return new Rectangle();
		else if(shapeType.equalsIgnoreCase("SQUARE")) return new Square();
		return null;
	}

}
```

```
package AbstractFactory;

public class Square implements Shape {

	@Override
	public void draw()
	{
		System.out.println("Inside Square :: draw() method.");
	}

}
```

Q.2. Write a python program to implement Polynomial Linear Regression for given dataset [20 M]

->

```
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

from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(X, y)

from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree = 4)
X_poly = poly_reg.fit_transform(X)
lin_reg_2 = LinearRegression()
lin_reg_2.fit(X_poly, y)
```

```
plt.scatter(X, y, color = 'red')
plt.plot(X, lin_reg.predict(X), color = 'blue')
plt.title('Truth or Bluff (Linear Regression)')
plt.xlabel('Position Level')
plt.ylabel('Salary')
plt.show()

plt.scatter(X, y, color = 'red')
plt.plot(X, lin_reg_2.predict(poly_reg.fit_transform(X)), color = 'blue')
plt.title('Truth or Bluff (Polynomial Regression)')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()

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.plot(X_grid, lin_reg_2.predict(poly_reg.fit_transform(X_grid)), color = 'blue')
plt.title('Truth or Bluff (Polynomial Regression)')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()
```

Q.3 Create your Django app in which after running the server, you should see on the browser, the text “Hello! I am learning Django”, which you defined in the index view.