Slip 1

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

→

```
package P2D_convUpper2lower;


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


      String str1 = "Software Architecture & Design Pattern List of Assignments";
      StringBuffer newStr = new StringBuffer(str1);
      System.out.println("\nString before case conversion : ");
      System.out.println(newStr);
      for (int i = 0; i < str1.length(); i++) {
         // Checks for lower case character
         // if(Character.isLowerCase(str1.charAt(i)))
         // {
         // Convert it into upper case using toUpperCase() function
         // newStr.setCharAt(i, Character.toUpperCase(str1.charAt(i)));
         // }
         // Checks for upper case character
         // else
         if (Character.isUpperCase(str1.charAt(i))) {
            // Convert it into upper case using toLowerCase() function
            newStr.setCharAt(i, Character.toLowerCase(str1.charAt(i)));
         }
      }
      System.out.println("\n String after case conversion : ");
      System.out.println(newStr);
   }
}
```

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

→

```
import pandas as pd
```

```
import matplotlib.pyplot as plt
from sklearn import preprocessing
iris = pd.read_csv("iris.csv")
#Drop id column
iris = iris.drop('Id',axis=1)
#Convert Species columns in a numerical column of the iris dataframe
#creating labelEncoder
le = preprocessing.LabelEncoder()
# Converting string labels into numbers.
iris.Species = le.fit_transform(iris.Species)
x = iris.iloc[:, :-1].values
y = iris.iloc[:, 4].values
plt.scatter(x[:,0], x[:, 3], c=y, cmap ='flag')
plt.xlabel('Sepal Length cm')
plt.ylabel('Petal Width cm')
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>
<head>
   <script>
   function validate()
{
   var fname = document.RegForm.FName;
   var lname = document.RegForm.LName;
   var age =document.RegForm.Age;
   var phone = document.RegForm.Telephone;
   var what =  document.RegForm.Subject;
   var address = document.RegForm.Address;
  function allLetter()
{
var letters = /^[A-Za-z]+$/;
```

```
//if(letters.test(fname.value))
if(fname.value.match(letters))
{
   fname.focus();
   return true;
}
else{
fname.focus();
return false;
}
}
function allNumeric()
 {
   var num= /^[0-9]{10}$/;
   if(num.test(phone.value))
   {
      phone.focus();
      return true;
   }
      else
      {
      return false;
   }

}
if(!allLetter())
{
   alert('Username must have alphabet characters only');
 }
 if(!allNumeric())
 window.alert("Enter correct number");
   if(age.value < 18 || age.value > 50)
   {
```

```
      window.alert("Please enter correct age.");
      age.focus();
      return false;
    }

  if (address.value == "")
   {
      window.alert("Please enter your address.");
      address.focus();
      return false;
    }
}
  </script>
   <title>Document</title>
</head>
<body bgcolor="aqua">
   <h1 style="text-align: left"> REGISTRATION FORM </h1>
<form name="RegForm" action="#"  method="post" >

   <p>First Name: <input type="text" size=40 name="FName"> </p><br>
   <p>Last Name: <input type="text" size=40 name="LName"> </p><br>
   <p>Age :     <input type="text" size="40" name="Age"></p>  <br>
  <p> Address:  <input type="text" size=40 name="Address">  </p><br>
   <p>Telephone: <input type="text" size=40 name="Telephone"> </p><br>

   <p>SELECT YOUR COURSE
      <select type="text" value="" name="Subject">
         <option>BTECH</option>
         <option>BBA</option>
         <option>BCA</option>
         <option>B.COM</option>
         <option>BCS</option>
      </select></p><br><br>
   <p>Comments:  <textarea cols="55" name="Comment">  </textarea></p>
```

```
   <p><input type="submit" value="submit" name="Submit" onclick="validate()"/>
      <input type="reset" value="Reset" name="Reset"/>
   </p>
</form>
</body>
</html>
```

Slip 2

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

→

```
package P4D_SingletonPattern;
public class Test {
   public static void main(String ar[]) {
      Test1 t = new Test1();
      Test1 t2 = new Test1();
      Test1 t3 = new Test1();
      Thread tt = new Thread(t);
      Thread tt2 = new Thread(t2);
      Thread tt3 = new Thread(t3);
      Thread tt4 = new Thread(t);
      Thread tt5 = new Thread(t);
      tt.start();
      tt2.start();
      tt3.start();
      tt4.start();
      tt5.start();
   }
}
final class Test1 implements Runnable {
   @Override
   public void run() {
      for (int i = 0; i < 5; i++) {
         System.out.println(Thread.currentThread().getName() + " : " + Single.getInstance().hashCode());
      }
```

```
    }
}
class Single {
    private final static Single sing = new Single();
    private Single() {
    }
    public static Single getInstance() {
        return sing;
    }
}
```

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

→

```
#drop
import numpy as np
import matplotlib.pyplot as mtp
import pandas as pd
dict={'First Score':[100,90,np.nan,95],
      'Second Score':[30,45,56,np.nan],
      'Third Score':[np.nan,40,80,98],
      'Fourth Score':[80,90,np.nan,50]}
#creating a dataframe from list
df=pd.DataFrame(dict)
print(df)
df.isnull()
```

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

->

```
<!DOCTYPE html>
<html lang="en">
<head>

    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title> Employee Registration Form</title>
```

```
<script>
function validate()
{
   var fname=document.empform.fname.value;
   var lname=document.empform.lname.value;
   var dob=document.empform.dob.value;
   var jdate=document.empform.jdate.value;
   var salary=document.empform.salary.value;



   var NPattern = /^[a-zA-Z]+$/;
   var dobPattern= /^([0-9]{1,2})([-/.])+([0-9]{1,2})([-/.])+([0-9]{4})$/;
   var JdatePattern= /^([0-9]{2})-([0-9]{2})-([0-9]{4})$/;
   var salaryPattern=/^[0-9]+$/;

   if(!NPattern.test(fname)){
      alert('Please Enter valid FirstName');
      return false;
   }

   if(!NPattern.test(lname)){
    alert('Please Enter valid LastName');
    return false;
   }



   if (!dobPattern.test(dob)) {
      alert("Invalid date of birth");
      return false;
   }
   if (!JdatePattern.test(jdate)) {
      alert("Invalid date of Joining");
      return false;
   }
```

```
   if(salary=="")
   {
    alert("Salary Required");
    return false;
   }
   if(!salaryPattern.test(salary))
   {
    alert("Salary Must be in digits)");
    return false;
    }



}</script>
</head>
<body>
   <table border="1" align="center">
   <tr><td>
   <form action="#" name="empform" onsubmit="return(validate());">
      <font size="4"><b>

      <h1>
         Employee Registration Form
      </h1>
      <pre>
    <ul>
    <li>First Name:    <input type="text" name="fname" ></li>
    <li>Last Name:     <input type="text" name="lname"></li>
    <li>DOB:          <input type="text" name="dob">format: DD-MM-YYYY</li>
    <li>Joining Date:  <input type="text" name="jdate">format:DD-MM-YYYY</li>
    <li>Salary:       <input type="text" name="salary"> </li>
    <li>Job Location:      <select name="Location">
       <option value="-1" selected>section location</option>
      <option value="Chennai">Chennai</option>
      <option value="Bangalore">Bangalore</option>
```

```
      <option value="Chennai">Pune</option>
      <option value="Bangalore">Mysore</option>
      <option value="Chennai">Chandigarh</option>

      </select></li>
     <li> Designation:  <select name="designation">
        <option value="-1" selected>select Designation</option>
         <option value="Systems Engineer">Systems Engineer</option>
         <option value="Senior Systems Engineer">Senior Systems Engineer</option>
         <option value="Technology Analyst">Technology Analyst</option>
         <option value="Technology Lead">Technology Lead</option>
         <option value="Project Manager">Project Manager</option>
       </select> </li>
      <li><input type="submit" value ="Submit">  <input type="reset" value ="Reset"></li>
   </ul>
   </pre>
</b></font>
</td></tr>
   </table>
</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()

->

DisplayElement.java

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

Observer.java

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

Subject.java

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

WeatherData.java

```
import java.util.*;
public 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;
}
```

```
}
```

CurrentConditionsDisplay.java

```
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");
   }
}
```

StatisticsDisplay.java

```
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);
    }
}
```

ForecastDisplay.java

```
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.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, rainy weather");
   }
}
```

```
}
```

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

->

```
import pandas as pd
df = pd.read_csv('salary.csv')
dummies = pd.get_dummies(df.Education)
merged = pd.concat([df, dummies], axis='columns')
merged.drop(['Education', 'Under-Graduate'], axis='columns')
print(merged)
```

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

->

```
<html>
<head>
</head>
<body>
<h2>Input an email and Submit</h2>
<form name="form1">
<ul>
<li><input type='email' name='text1' id="email"/></li>
 
<li><input type="submit" name="submit" value="Submit" onclick="ValidateEmail()"/></li>
 
</ul>
</form>
<script>
function ValidateEmail()
{
var mailformat = document.getElementById('email').value;
var pattern=/^[a-zA-Z0-9._]+@[a-zA-Z0-9._]+\.[a-zA-Z]{2,3}$/;
// /^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/;
if(pattern.test(mailformat))
{
```

```
   alert("welcome");
return true;
}
else
{
alert("You have entered an invalid email address!");
return false;
}
}
</script>
</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.

->

Main.java

```
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("\n### Making a " + pizza.getName() + " ### \n");
		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");
        }
}
```

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

->

```
# Commented out IPython magic to ensure Python compatibility.
import pandas as pd
import numpy as np
```

```
import matplotlib.pyplot as plt
from sklearn import linear_model
from sklearn.cross_validation import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
# %matplotlib inline
dataset=pd.read_csv('Salary_Data.csv')
print(dataset.shape)
#dataset.head(5)
#dataset.describe()in scatter plot
#data can be represented
#dataset.plot(x='YearsExperience',y='Salary',style='*')
#plt.title('yearexp vs salary')
#plt.xlabel('Yearsofexp')
#plt.ylabel('Salary')
#Text(0,0.5,'Salary')
#dependancy linear in nature then go for Linear regression model
#convert i/p values in numpy array
x=dataset['YearsExperience'].values.reshape(-1,1)
y=dataset['Salary'].values.reshape(-1,1)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
#split data into 80% training and 20% testing
#algorithm train on traning dataset using LR
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
sc.fit(x_test)
x_train=sc.transform(x_train)
x_test=sc.transform(x_test)
y_train=sc.transform(x_train)
y_test=sc.transform(y_test)
LR=LinearRegression()
LR.fit(x_train,y_train)
print("Intercept",LR.intercept_)#for retrive slope
print("coefficient",LR.coef_)
```

```
y_pred=LR.predict(x_test)
#scatter plot where the line indicate line of regression
plt.scatter(x_train,y_train)
plt.plot(x_test,y_pred,color='red')
plt.title("simple regression")
plt.xlabel("YerasofExp")
plt.ylabel("salary")
plt.show()
accuracy=LR.score(x_test,y_test)
print('Accurecy='+str(accuracy))
print(LR.predict([[1.2]]))
```

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

->

```
<html>
<body>
<p>Click the button to convert the string to uppercase letters.</p>
<button onclick="myFunction()">Try it</button>
<p id="demo"></p>

<script>
function myFunction() {
  var str = "Hello World!";
  var res = str.toUpperCase();
  document.getElementById("demo").innerHTML = res;
}
</script>
</body>
</html>
```

Slip 5

Q 1. Write a Java Program to implement Adapter pattern for Enumeration iterator.

->

Duck.java

```
public interface Duck {
	public void quack();
	public void fly();
}
```

Turkey.java

```
public interface Turkey {  //Turkeys don't quack
	public void gobble();
	public void fly();
}
```

MallardDuck.java

```
public class MallardDuck implements Duck {
	public void quack() {
		System.out.println("Quack");
	}

	public void fly() {
		System.out.println("I'm flying");
	}
}
```

WildTurkey.java

```
public class WildTurkey implements Turkey {
	public void gobble() {
		System.out.println("Gobble gobble");
	}

	public void fly() {
		System.out.println("I'm flying a short distance");
```

```
        }
}
```

TurkeyAdapter.java

```
public class TurkeyAdapter implements Duck {
        Turkey turkey;

        public TurkeyAdapter(Turkey turkey) {
                this.turkey = turkey;
        }

        public void quack() {
                turkey.gobble();
        }

        public void fly() {
                for(int i=0; i < 5; i++) {
                        turkey.fly();
                }
        }
}
```

TurkeyTestDrive.java

```
public class TurkeyTestDrive {
        public static void main(String[] args) {
                MallardDuck duck = new MallardDuck();
                WildTurkey turkey = new WildTurkey();
                Duck turkeyAdapter = new TurkeyAdapter(turkey);

                System.out.println("\nThe Turkey says ...");
                turkey.gobble();
                turkey.fly();

                System.out.println("\nThe Duck says ...");
```

```
        testDuck(duck);

        System.out.println("\nThe TurkeyAdapter says ...");
        testDuck(turkeyAdapter);
    }

    static void testDuck(Duck duck) {
        duck.quack();
        duck.fly();

    }
}
```

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

->

```
# Commented out IPython magic to ensure Python compatibility.
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import linear_model
from sklearn.cross_validation import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
# %matplotlib inline
df=pd.read_csv("cars.csv")
df.head(5)
x=df[['debt','income']]
y=df['sales']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
from sklearn.linear_model import LinearRegression
MLR=LinearRegression()
MLR.fit(x_train,y_train)
print("intercept",MLR.intercept_)
print("coefficient",MLR.coef_)
```

```
predict=MLR.predict([[418,7017]])
print(predict)
```

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

->

```
<html>
<head><h2>Append contents of first file into second file</h2>
   <title>Document</title>
   <script type="text/javascript">
function validateFile(){
   var file1= document.getElementById("file_1").value;
   var file2 = document.getElementById("'file_2").value;
var fs = require('fs');
   fs.open('file1', 'r+', function (err, fd) {
      if (err) { return console.error(err);
      }

      var buffr = new Buffer.alloc(24);
      fs.read(fd, buffr, 0, buffr.length, 0, function (err, bytes) {
         if (err) throw err;
         console.log(buffr.toString());
      });

fs.appendFile('file2', buffr, function (err) {
  if (err) throw err;
  console.log('Saved!');
  fs.close(fd, function (err) { if (err) throw err; });
});

});
   //});
}
   </script>
```

```
</head>
<body>

   <table border="1">
   <form name="form1" actio=# onsubmit="return validateFile();">
    <tr><td> Enter First File: <input type="text" name="file1" id="file_1"><br></td></tr>
      
     <tr><td>Enter second File:<input type="text" name="file2" id="file_2"><br></td></tr>
      
     <tr><td><input type="button" name="Append" value="Append"></td></tr>
   </form>
   </table>

</body>
</html>
```

Slip 6

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

->

```
package P5D_commandPattern;
// An interface for command
interface Command {
   public void execute();
}
// Light class and its corresponding command
// classes
class Light {
   public void on() {
      System.out.println("Light is on");
   }
   public void off() {
      System.out.println("Light is off");
   }
}
```

```
class LightOnCommand implements Command {
   Light light;
   // The constructor is passed the light it
   // is going to control.
   public LightOnCommand(Light light) {
      this.light = light;
   }
   public void execute() {
      light.on();
   }
}
class LightOffCommand implements Command {
   Light light;

   public LightOffCommand(Light light) {
      this.light = light;
   }
   public void execute() {
      light.off();
   }
}
// Stereo and its command classes
class Stereo {
   public void on() {
      System.out.println("Stereo is on");
   }
   public void off() {
      System.out.println("Stereo is off");
   }
   public void setCD() {
      System.out.println("Stereo is set " +
            "for CD input");
   }
   public void setDVD() {
```

```
        System.out.println("Stereo is set" +
                " for DVD input");
    }
    public void setRadio() {
        System.out.println("Stereo is set" +
                " for Radio");
    }
    public void setVolume(int volume) {
        // code to set the volume
        System.out.println("Stereo volume set"
                + " to " + volume);
    }
}
class StereoOffCommand implements Command {
    Stereo stereo;

    public StereoOffCommand(Stereo stereo) {
        this.stereo = stereo;
    }
    public void execute() {
        stereo.off();
    }
}
class StereoOnWithCDCommand implements Command {
    Stereo stereo;

    public StereoOnWithCDCommand(Stereo stereo) {
        this.stereo = stereo;
    }
    public void execute() {
        stereo.on();
        stereo.setCD();
        stereo.setVolume(11);
    }
```

```
}
// A Simple remote control with one button
class SimpleRemoteControl {
   Command slot; // only one button
   public SimpleRemoteControl() {
   }
   public void setCommand(Command command) {
      // set the command the remote will
      // execute
      slot = command;
   }
   public void buttonWasPressed() {
      slot.execute();
   }
}
// Driver class
class RemoteControlTest {
   public static void main(String[] args) {
      SimpleRemoteControl remote = new SimpleRemoteControl();
      Light light = new Light();
      Stereo stereo = new Stereo();

      // we can change command dynamically
      remote.setCommand(new LightOnCommand(light));
      remote.buttonWasPressed();
      remote.setCommand(new StereoOnWithCDCommand(stereo));
      remote.buttonWasPressed();
      remote.setCommand(new StereoOffCommand(stereo));
      remote.buttonWasPressed();
   }
}
```

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

->

```
#polynomial regression
```

```
import numpy as np
import matplotlib.pyplot as mtp
import pandas as pd
dataset=pd.read_csv('Position_Salaries.csv')
print(dataset)
x=dataset.iloc[:,1:2].values
#print(x)
y=dataset.iloc[:,2].values
print(y)
from sklearn import linear_model
from sklearn.cross_validation import train_test_split
from sklearn.linear_model import LinearRegression
lin=LinearRegression()
lin.fit(x,y)
mtp.scatter(x,y,color='blue')
mtp.plot(x,lin.predict(x),color="red")
mtp.title("salary estimate")
mtp.xlabel("Position Levels")
mtp.ylabel("salary")
mtp.show()
from sklearn.preprocessing import PolynomialFeatures
poly=PolynomialFeatures(degree=2)
x_poly=poly.fit_transform(x)
lin_reg=LinearRegression()
lin_reg.fit(x_poly,y)
mtp.scatter(x,y,color='blue')
mtp.plot(x,linreg.predict(x_poly),color="red")
mtp.title("salary estimate")
mtp.xlabel("Position Levels")
mtp.ylabel("salary")
mtp.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.

->

.js file –

```
var http=require('http');
var url= require('url');
var fs= require('fs');
http.createServer(function(req,res){

var p= url.parse(req.url,true);
var filename="."+p.pathname;
console.log('file received'+filename);
fs.readFile(filename,function(err,data){
   if(err)
   {
      res.writeHead(404,{'content-type':'text/html'});
      res.end('404 page Not found');
   }
   else
   {
      res.writeHead(200,{'content-type':'text/html'});
      res.write(data);
      res.end();

   }
});

}).listen(8000);
console.log('server is running on port 8000');
```

Slip 7

Q 1. Write a Java Program to implement undo command to test Ceiling fan.

->

```
package Slip7.Q1;
interface Command {
   public void execute();
```

```
}
class CeilingFan {
   public void on() {
      System.out.println("Ceiling Fan is on");
   }
   public void off() {
      System.out.println("Ceiling Fan is off");
   }
}
class CeilingFanOnCommand implements Command {
   CeilingFan c;
   public CeilingFanOnCommand(CeilingFan l) {
      this.c = l;
   }
   public void execute() {
      c.on();
   }
}
class CeilingFanOffCommand implements Command {
   CeilingFan c;
   public CeilingFanOffCommand(CeilingFan l) {
      this.c = l;
   }
   public void execute() {
      c.off();
   }
}
class SimpleRemoteControl {
   Command slot;
   public SimpleRemoteControl() {
   }
   public void setCommand(Command command) {
      slot = command;
   }
```

```
    public void buttonWasPressed() {
        slot.execute();
    }
}
public class Main {
    public static void main(String[] args) {
        SimpleRemoteControl remote = new SimpleRemoteControl();
        CeilingFan ceilingFan = new CeilingFan();
        CeilingFanOnCommand ceilingFanOn = new CeilingFanOnCommand(ceilingFan);
        remote.setCommand(ceilingFanOn);
        remote.buttonWasPressed();
        CeilingFanOffCommand ceilingFanOff = new CeilingFanOffCommand(ceilingFan);
        remote.setCommand(ceilingFanOff);
        remote.buttonWasPressed();
    }
}
```

Q 2. Write a python program to implement Naive Bayes.

->

```
#naive bayes
import numpy as np
import matplotlib.pyplot as mtp
import pandas as pd
dataset=pd.read_csv('suv_data.csv')
print(dataset)
x=dataset.iloc[:,[2,3]].values
print(x)
y=dataset.iloc[:,4].values
print(y)
#splitting dataset into training and testing dataset
from sklearn import linear_model
from sklearn.cross_validation import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
#feature scaling
```

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_train=sc.fit_transform(x_train)
x_test=sc.transform(x_test)
from sklearn.naive_bayes import GaussianNB
classifier=GaussianNB()
classifier.fit(x_train,y_train)
GaussianNB(priors=None,var_smoothing=1e-09)
#predecting the test set result
y_pred=classifier.predict(x_test)
#making confusion matrix
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred)
print(cm)
from sklearn.metrics import accuracy_score
print("Accuracy",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 formidable = require('formidable');
var fs = require('fs');

http.createServer(function (req, res) {
  if (req.url == '/fileupload') {
    var form = new formidable.IncomingForm();
    form.parse(req, function (err, fields, files) {
      var oldpath = files.filetoupload.filepath;
      var newpath = 'C:/Users/mahes/Downloads/' + files.filetoupload.originalFilename;
      fs.rename(oldpath, newpath, function (err) {
        if (err) throw err;
        res.write('File uploaded and moved!');
        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);

// npm install formidable
// node upload.js
```

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, behaviours and state transition that can happen.

->

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(5);
                gumballMachine.insertQuarter();
                gumballMachine.turnCrank();
                System.out.println(gumballMachine);
```

```
	}
}
```

State.java

```
public interface State {
	public void insertQuarter();
	public void ejectQuarter();
	public void turnCrank();
	public void dispense();
	public void refill();
}
```

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");
                result.append("\nInventory: " + count + " gumball");
                if (count != 1) {
                        result.append("s");
                }
                result.append("\n");
                result.append("Machine is " + state + "\n");
                return result.toString();
        }
}
```

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";
	}
}
```

SoldState.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";
        }
}
```

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

->

```
# -*- coding: utf-8 -*-
"""Untitled
Automatically generated by Colaboratory.
Original file is located at
   https://colab.research.google.com/drive/1xJTHdf4gm5WGU-RyycRQh8csATZIkJ7B
"""
import numpy as np
import matplotlib.pyplot as mtp
import pandas as pd
#loding a play tennis data
PlayTennis=pd.read_csv('playtennis.csv')
PlayTennis
from sklearn.preprocessing import LabelEncoder
```

```
Le=LabelEncoder()
PlayTennis['outlook']=Le.fit_transform(PlayTennis['outlook'])
PlayTennis['temp']=Le.fit_transform(PlayTennis['temp'])
PlayTennis['humidity']=Le.fit_transform(PlayTennis['humidity'])
PlayTennis['windy']=Le.fit_transform(PlayTennis['windy'])
PlayTennis['play']=Le.fit_transform(PlayTennis['play'])
PlayTennis
y=PlayTennis['play']
x=PlayTennis.drop(['play'],axis=1)
from sklearn import tree
clf=tree.DecisionTreeClassifier(criterion='entropy')
clf=clf.fit(x,y)
#we can visualize
tree.plot_tree(clf)
```

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: "yourusername",
  password: "yourpassword",
  database: "mydb"
});
con.connect(function(err) {
  if (err) throw err;
  console.log("Connected!");
  var sql = "CREATE TABLE customers (name VARCHAR(255), address VARCHAR(255))";
  con.query(sql, function (err, result) {
    if (err) throw err;
    console.log("Table created");
  });
});
// node demo_create_table.js
```

Slip 9

Q 1. Design simple HR Application using Spring Framework. (code not available)

->

Q 2. Write a python program to implement Linear SVM.

->

```
# -*- coding: utf-8 -*-
"""SVM.ipynb
Automatically generated by Colaboratory.
Original file is located at
    https://colab.research.google.com/drive/1AyDIq1AufzUoaIuwUkloBfYjO-BIjEna
"""
import numpy as np
import matplotlib.pyplot as mtp
import pandas as pd
dataset=pd.read_csv('suv_data.csv')
x=dataset.iloc[:,[2,3]].values
y=dataset.iloc[:,4].values
#splitting dataset into traning and testing dataset
from sklearn import linear_model
from sklearn.cross_validation import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25,random_state=0)
#feature scaling
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_train=sc.fit_transform(x_train)
x_test=sc.transform(x_test)
from sklearn.svm import SVC
classifier=SVC(kernel='linear',random_state=0)
classifier.fit(x_train,y_train)
y_pred=classifier.predict(x_test)
#making confusion matrix
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred)
```

```
print(cm)
from sklearn.metrics import accuracy_score
print("Accuracy",accuracy_score(y_test,y_pred))
#from this 8+2 incorrect prediction and 66+24 correct prediction
```

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: "yourusername",
  password: "yourpassword",
  database: "mydb"
});
con.connect(function(err) {
  if (err) throw err;
  con.query("SELECT * FROM customers", function (err, result, fields) {
    if (err) throw err;
    console.log(result);
  });
});
// node demo_db_select.js
```

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.

->

```
package Slip10.Q1;
/*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 q;
    }
}
class MallardDuck extends Duck {
    public MallardDuck() {
        quackBehaviour = new Quack();
        flyBehaviour = new FlyWithWings();
    }
    public void display() {
        System.out.println("I'm a real Mallard duck");
    }
}
interface FlyBehaviour {
    public void fly();
}
interface QuackBehaviour {
    public void quack() {
    System.out.println("Quack");
    }
}
class Quack implements QuackBehaviour {
    public void quack() {
```

```
        System.out.println("Quack");
    }
}
class FlyWithWings implements FlyBehaviour {
    public void fly() {
        System.out.println("I'm flying!!");
    }
}
public class Main {
    public static void main(String[] args) {
        Duck mallard = new MallardDuck();
        mallard.performQuack();
        mallard.performFly();
    }
}
*/
/* Simple Programme */
interface DuckB {
    public void oper();
}
class Fly implements DuckB {
    public void oper() {
        System.out.println("Duck Flies");
    }
}
class Quack implements DuckB {
    public void oper() {
        System.out.println("Duck Sounds Quack Quack");
    }
}
class Context {
    private DuckB s1;

    public Context(DuckB p) {
```

```
        this.s1 = p;
    }
    public void est() {
        s1.oper();
    }
}
public class Main {
    public static void main(String[] args) {
        Context c1 = new Context(new Fly());
        System.out.println("Duck Behaviour");
        c1.est();
        c1 = new Context(new Quack());
        System.out.println("Duck Behaviour ");
        c1.est();
    }
}
```

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

->

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import preprocessing
iris = pd.read_csv("iris.csv")
#Drop id column
iris = iris.drop('Id',axis=1)
#Convert Species columns in a numerical column of the iris dataframe
#creating labelEncoder
le = preprocessing.LabelEncoder()
# Converting string labels into numbers.
iris.Species = le.fit_transform(iris.Species)
x = iris.iloc[:, :-1].values
y = iris.iloc[:, 4].values
plt.scatter(x[:,0], x[:, 3], c=y, cmap ='flag')
plt.xlabel('Sepal Length cm')
plt.ylabel('Petal Width cm')
```

```
plt.show()
```

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

->

```
// include mysql module
var mysql = require('mysql');
// create a connection variable with the required details
var con = mysql.createConnection({
  host: "localhost",    // ip address of server running mysql
  user: "root",    // user name to your mysql database
  password: "", // corresponding password
  database: "students" // use the specified database
});

// make to connection to the database.
con.connect(function(err) {
  if (err) throw err;
  // if connection is successful
  var records = [
    ['Ramesh', 13, 85],
    ['Suresh', 14, 87],
    ['Jitesh', 15, 74]
  ];
  con.query("INSERT INTO students (name,rollno,marks) VALUES ?", [records], function (err, result, fields) {
    // if any error while executing above query, throw error
    if (err) throw err;
    // if there is no error, you have the result
    console.log(result);
  });
});
```

Slip 11

Q 1. Write a java program to implement Adapter pattern to design Heart Model to Beat Model. (code not available)

->

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

->

```
#drop
import numpy as np
import matplotlib.pyplot as mtp
import pandas as pd
dict={'First Score':[100,90,np.nan,95],
     'Second Score':[30,45,56,np.nan],
     'Third Score':[np.nan,40,80,98],
     'Fourth Score':[80,90,np.nan,50]}
#creating a dataframe from list
df=pd.DataFrame(dict)
print(df)
df.isnull()
```

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: "yourusername",
  password: "yourpassword",
  database: "mydb"
});
con.connect(function(err) {
  if (err) throw err;
  var sql = "DELETE FROM customers WHERE address = 'Mountain 21'";
  con.query(sql, function (err, result) {
    if (err) throw err;
    console.log("Number of records deleted: " + result.affectedRows);
  });
});
// node demo_db_delete.js
```

Q 1. Write a Java Program to implement Decorator Pattern for interface Car to define the assemble() method and then decorate it to Sports car and Luxury Car.

->

```
package Slip12.Q1;
interface Car {
   public void assemble();
}
class BasicCar implements Car {
   @Override
   public void assemble() {
      System.out.print("Basic Car.");
   }

class CarDecorator implements Car {
   protected Car car;
   public CarDecorator(Car c) {
      this.car = c;
   }

   @Override
   public void assemble() {
      this.car.assemble();
   }
}
class SportsCar extends CarDecorator {
   public SportsCar(Car c) {
      super(c);
   }
   @Override
   public void assemble() {
      car.assemble();
      System.out.print(" Adding features of Sports Car.");
   }
}
```

```
class LuxuryCar extends CarDecorator {
   public LuxuryCar(Car c) {
      super(c);
   }
   public void assemble() {
      car.assemble();
      System.out.print(" Adding features of Luxury Car.");
   }
}
public class Main {
   public static void main(String[] args) {
      Car s1 = new SportsCar(new BasicCar());
      s1.assemble();
      Car s2 = new LuxuryCar(new BasicCar());
      s2.assemble();
   }
}
```

Q 2. Write a python program to make Categorical values in numeric format for a given dataset. (code not available)

->

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

->

```
var http =require('http');
var url =require('url');
http.createServer(function(req,res){
   res.writeHead(200,{'content-type':'text/html'});
res.write(req.url);
res.end();
}).listen(5000);
// node server.js
```

Slip 13

Q 1. Write a Java Program to implement an Adapter design pattern in mobile charger. Define two classes – Volt (to measure volts) and Socket (producing constant volts of 120V). Build an adapter that can produce 3 volts, 12 volts and default 120 volts. Implements Adapter pattern using Class Adapter.

->

```
package Slip13.Q1;
class Volt {
   private int volts;
   public Volt(int v) {
      this.volts = v;
   }
   public int getVolts() {
      return volts;
   }
   public void setVolts(int volts) {
      this.volts = volts;
   }
}
class Socket {
   public Volt getVolt() {
      return new Volt(120);
   }
}
interface SocketAdapter {
   public Volt get120Volt()
   public Volt get12Volt();
   public Volt get3Volt();
}
class SocketClassAdapterImpl extends Socket implements SocketAdapter {
   @Override
   public Volt get120Volt() {
      return getVolt();
   }
   @Override
   public Volt get12Volt() {
      Volt v = getVolt();
      return convertVolt(v, 10);
   }
   @Override
```

```
    public Volt get3Volt() {
        Volt v = getVolt();
        return convertVolt(v, 40);
    }
    private Volt convertVolt(Volt v, int i) {
        return new Volt(v.getVolts() / i);
    }
}
class SocketObjectAdapterImpl implements SocketAdapter {
    // using composition for adapter pattern
    private Socket sock = new Socket();

    @Override
    public Volt get120Volt() {
        return sock.getVolt();
    }
    @Override
    public Volt get12Volt() {
        Volt v = sock.getVolt();
        return convertVolt(v, 10);
    }
    @Override
    public Volt get3Volt() {
        Volt v = sock.getVolt();
        return convertVolt(v, 40);
    }
    private Volt convertVolt(Volt v, int i) {
        return new Volt(v.getVolts() / i);
    }
}
public class Main {
    public static void main(String[] args) {
        testClassAdapter();
        testObjectAdapter();
```

```
    private static void testObjectAdapter() {
        SocketAdapter sockAdapter = new SocketObjectAdapterImpl();
        Volt v3 = getVolt(sockAdapter, 3);
        Volt v12 = getVolt(sockAdapter, 12);
        Volt v120 = getVolt(sockAdapter, 120);
        System.out.println("v3 volts using Object Adapter=" + v3.getVolts());
        System.out.println("v12 volts using Object Adapter=" + v12.getVolts());
        System.out.println("v120 volts using Object Adapter=" + v120.getVolts());
    }
    private static void testClassAdapter() {
        SocketAdapter sockAdapter = new SocketClassAdapterImpl();
        Volt v3 = getVolt(sockAdapter, 3);
        Volt v12 = getVolt(sockAdapter, 12);
        Volt v120 = getVolt(sockAdapter, 120);
        System.out.println("v3 volts using Class Adapter=" + v3.getVolts());
        System.out.println("v12 volts using Class Adapter=" + v12.getVolts());
        System.out.println("v120 volts using Class Adapter=" + v120.getVolts());
    }
    private static Volt getVolt(SocketAdapter sockAdapter, int i) {
        switch (i) {
            case 3:
                return sockAdapter.get3Volt();
            case 12:
                return sockAdapter.get12Volt();
            case 120:
                return sockAdapter.get120Volt();
            default:
                return sockAdapter.get120Volt();
        }
    }
}
```

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

->

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import preprocessing
iris = pd.read_csv("iris.csv")
#Drop id column
iris = iris.drop('Id',axis=1)
#Convert Species columns in a numerical column of the iris dataframe
#creating labelEncoder
le = preprocessing.LabelEncoder()
# Converting string labels into numbers.
iris.Species = le.fit_transform(iris.Species)
x = iris.iloc[:, :-1].values
y = iris.iloc[:, 4].values
plt.scatter(x[:,0], x[:, 3], c=y, cmap ='flag')
plt.xlabel('Sepal Length cm')
plt.ylabel('Petal Width cm')
plt.show()
```

Q 3. Using node js create a User Login System. (code not available)

->

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.

->

```
package slip14.Q1;
interface Command {
   public void execute();
}
class Stereo {
   public void On() {
      System.out.println("Stereo is on");
   }
```

```
}
class GarageDoor {
   public void Up() {
      System.out.println("Garage Door is Up");
   }
}
class GarageDoorUpCommand implements Command {
   GarageDoor c;
   public GarageDoorUpCommand(GarageDoor l) {
      this.c = l;
   }
   public void execute() {
      c.Up();
   }
}

class Light {
   public void on() {
      System.out.println("Light is on");
   }
   public void off() {
      System.out.println("Light is off");
   }
}
class LightOnCommand implements Command {
   Light light;
   public LightOnCommand(Light light) {
      this.light = light;
   }
   public void execute() {
      light.on();
   }
}
class LightOffCommand implements Command {
```

```
    Light light;
    public LightOffCommand(Light light) {
        this.light = light;
    }
    public void execute() {
        light.off();
    }
}
class StereoOn implements Command {
    Stereo s;
    public StereoOn(Stereo l) {
        this.s = l;
    }
    public void execute() {
        s.On();
    }
}
class SimpleRemoteControl {
    Command slot;
    public SimpleRemoteControl() {
    }
    public void setCommand(Command command) {
        slot = command;
    }
    public void buttonWasPressed() {
        slot.execute();
    }
}
public class Main {
    public static void main(String[] args) {
        SimpleRemoteControl remote = new SimpleRemoteControl();
        Light light = new Light();
        LightOnCommand lightOn = new LightOnCommand(light);
        remote.setCommand(lightOn);
```

```
        remote.buttonWasPressed();
        LightOffCommand lightOff = new LightOffCommand(light);
        remote.setCommand(lightOff);
        remote.buttonWasPressed();
        GarageDoor garageDoor = new GarageDoor();
        GarageDoorUpCommand garageDoorUp = new GarageDoorUpCommand(garageDoor);
        remote.setCommand(garageDoorUp);
        remote.buttonWasPressed();
        Stereo s1 = new Stereo();
        StereoOn s2 = new StereoOn(s1);
        remote.setCommand(s2);
        remote.buttonWasPressed();
    }
}
```

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

->

```
#drop
import numpy as np
import matplotlib.pyplot as mtp
import pandas as pd
dict={'First Score':[100,90,np.nan,95],
     'Second Score':[30,45,56,np.nan],
     'Third Score':[np.nan,40,80,98],
     'Fourth Score':[80,90,np.nan,50]}
#creating a dataframe from list
df=pd.DataFrame(dict)
print(df)
df.isnull()
```

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

->

```
<!DOCTYPE html>
<html lang="en">
<head>
```

```
<script>
var http = require('http');
var fs = require('fs');
http.createServer(function (req, res) {
   fs.readFile('index.html', function(err, data) {
      res.writeHead(200, {'Content-Type': 'text/html'});
      res.write(data);
      return res.end();
  });
}).listen(8080);

// node file.js
// http://localhost:8080/
</script>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Document</title>
</head>
<body>
   <h1>Ramesh Here</h1>
</body>
</html>
```

Slip 16

Q 1. Write a Java Program to implement Observer Design Pattern for number conversion. Accept a number in Decimal form and represent it in Hexadecimal, Octal and Binary. Change the Number and it reflects in other forms also

->

```
package Slip16.Q1;
import java.util.ArrayList;
import java.util.List;
class Subject {
   private List<Observer> observers = new ArrayList<Observer>();
   private int state;
```

```
    public int getState() {
        return state;
    }
    public void setState(int s) {
        this.state = s;
        notifyAllObservers();
    }
    public void attach(Observer o1) {
        observers.add(o1);
    }
    public void notifyAllObservers() {
        for (Observer o1 : observers) {
            o1.update();
        }
    }
abstract class Observer {
    protected Subject s1;
    public abstract void update();
}
class BinaryObserver extends Observer {
    public BinaryObserver(Subject s) {
        this.s1 = s;
        this.s1.attach(this);
    }
    public void update() {
        System.out.println("Binary String:" + Integer.toBinaryString(s1.getState()));
    }
}
class OctalObserver extends Observer {
    public OctalObserver(Subject s) {
        this.s1 = s;
        this.s1.attach(this);
    }
    public void update() {
```

```
        System.out.println("Octal String:" + Integer.toOctalString(s1.getState()));
    }
}
class HexaObserver extends Observer {
    public HexaObserver(Subject s) {
        this.s1 = s;
        this.s1.attach(this);
    }
    public void update() {
        System.out.println("Heaxdeciamal String:" + Integer.toHexString(s1.getState()));
    }
}
public class Main {
    public static void main(String[] args) {
        Subject s1 = new Subject();
        new BinaryObserver(s1);
        new OctalObserver(s1);
        new HexaObserver(s1);
        System.out.println("First state Change:15");
        s1.setState(15);
        System.out.println("Second state Change:10");
        s1.setState(10);
    }
}
```

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

->

```
# Commented out IPython magic to ensure Python compatibility.
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import linear_model
from sklearn.cross_validation import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
```

```
# %matplotlib inline
dataset=pd.read_csv('Salary_Data.csv')
print(dataset.shape)
#dataset.head(5)
#dataset.describe()in scatter plot
#data can be represented
#dataset.plot(x='YearsExperience',y='Salary',style='*')
#plt.title('yearexp vs salary')
#plt.xlabel('Yearsofexp')
#plt.ylabel('Salary')
#Text(0,0.5,'Salary')
#dependancy linear in nature then go for Linear regression model
#convert i/p values in numpy array
x=dataset['YearsExperience'].values.reshape(-1,1)
y=dataset['Salary'].values.reshape(-1,1)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
#split data into 80% training and 20% testing
#algorithm train on traning dataset using LR
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
sc.fit(x_test)
x_train=sc.transform(x_train)
x_test=sc.transform(x_test)
y_train=sc.transform(x_train)
y_test=sc.transform(y_test)
LR=LinearRegression()
LR.fit(x_train,y_train)
print("Intercept",LR.intercept_)#for retrive slope
print("coefficient",LR.coef_)
y_pred=LR.predict(x_test)
#scatter plot where the line indicate line of regression
plt.scatter(x_train,y_train)
plt.plot(x_test,y_pred,color='red')
plt.title("simple regression")
```

```
plt.xlabel("YerasofExp")
plt.ylabel("salary")
plt.show()
accuracy=LR.score(x_test,y_test)
print('Accurecy='+str(accuracy))
print(LR.predict([[1.2]]))
```

Q 3. Create a js file named main.js for event-driven application. There should be a main loop that listens for events, and then triggers a callback function when one of those events is detected.

->

```
 var events = require('events');
var myeventEmitter = new events.EventEmitter();
myeventEmitter.on('myevent', function Listener1(){
    console.log('listener1 executed!');
});
myeventEmitter.on('myevent' ,function Listener2(a,b){
    console.log('listener2 executed! parameter');
});
console.log(myeventEmitter.listeners('myevent'));
myeventEmitter.emit('myevent' ,10,20);
```

Slip 17

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

->

```
interface Shape {
   void draw();
}
class RoundedRectangle implements Shape {
   public void draw() {
      System.out.println(" Inside RR");
   }
}
class RoundedSquare implements Shape {
   public void draw() {
```

```
      System.out.println(" Inside RS");
   }
}
class Rectangle implements Shape {
   public void draw() {
      System.out.println(" Inside Simple R");
   }
}
class Square implements Shape {
   public void draw() {
      System.out.println(" Inside Simple Sq");
   }
}
abstract class AbstractFactory {
   abstract Shape getShape(String st);
}
class ShapeFactory extends AbstractFactory {
   public Shape getShape(String st) {
      if (st.equalsIgnoreCase("Rectangle")) {
         return new Rectangle();
      } else if (st.equalsIgnoreCase("Square")) {
         return new Square();
      }
      return null;
   }
}
class RoundedShapeFactory extends AbstractFactory {
   public Shape getShape(String st) {
      if (st.equalsIgnoreCase("Rectangle")) {
         return new RoundedRectangle();
      } else if (st.equalsIgnoreCase("Square")) {
         return new RoundedSquare();
      }
      return null;
```

```
   }
}
class FactoryProducer {
   public static AbstractFactory getFactory(boolean rounded) {
      if (rounded) {
         return new RoundedShapeFactory();
      } else {
         return new ShapeFactory();
      }
   }
}
public class Main {
   public static void main(String[] args) {
      AbstractFactory shapeFactory = FactoryProducer.getFactory(false);
      Shape shape1 = shapeFactory.getShape("Rectangle");
      shape1.draw();
      Shape shape2 = shapeFactory.getShape("SQuare");
      shape2.draw();
      AbstractFactory shapeFactory1 = FactoryProducer.getFactory(true);
      Shape shape3 = shapeFactory1.getShape("REctangle");
      shape3.draw();
      Shape shape4 = shapeFactory1.getShape("square");
      shape4.draw();
   }
}
```

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

->

```
# Commented out IPython magic to ensure Python compatibility.
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import linear_model
from sklearn.cross_validation import train_test_split
from sklearn.linear_model import LinearRegression
```

```
from sklearn import metrics

# %matplotlib inline

df=pd.read_csv("cars.csv")

df.head(5)

x=df[['debt','income']]

y=df['sales']

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)

from sklearn.linear_model import LinearRegression

MLR=LinearRegression()

MLR.fit(x_train,y_train)

print("intercept",MLR.intercept_)

print("coefficient",MLR.coef_)

predict=MLR.predict([[418,7017]])

print(predict)
```

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. (code not available)

->

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()

->

DisplayElement.java

```
public interface DisplayElement

{

   public void display();

}
```

Observer.java

```
public interface Observer

{

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

}
```

Subject.java

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

WeatherData.java

```
import java.util.*;
public 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;
    }
}
```

CurrentConditionsDisplay.java

```
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");
    }
}
```

StatisticsDisplay.java

```
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);
  }
}
```

ForecastDisplay.java

```
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.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, rainy weather");
        }
    }
}
```

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

->

```
# Commented out IPython magic to ensure Python compatibility.
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import linear_model
from sklearn.cross_validation import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
# %matplotlib inline
df=pd.read_csv("cars.csv")
df.head(5)
x=df[['debt','income']]
y=df['sales']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
from sklearn.linear_model import LinearRegression
MLR=LinearRegression()
MLR.fit(x_train,y_train)
print("intercept",MLR.intercept_)
print("coefficient",MLR.coef_)
predict=MLR.predict([[418,7017]])
print(predict)
```

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. (code not available)

->

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.

->

Main.java

```
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("\n### Making a " + pizza.getName() + " ### \n");
                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");
    }
}
```

Q 2. Write a python program to implement Naive Bayes.

->

```
#naive bayes
import numpy as np
import matplotlib.pyplot as mtp
import pandas as pd
dataset=pd.read_csv('suv_data.csv')
print(dataset)
x=dataset.iloc[:,[2,3]].values
print(x)
y=dataset.iloc[:,4].values
print(y)
#splitting dataset into training and testing dataset
from sklearn import linear_model
from sklearn.cross_validation import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
#feature scaling
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_train=sc.fit_transform(x_train)
x_test=sc.transform(x_test)
from sklearn.naive_bayes import GaussianNB
classifier=GaussianNB()
classifier.fit(x_train,y_train)
GaussianNB(priors=None,var_smoothing=1e-09)
#predecting the test set result
y_pred=classifier.predict(x_test)
```

```
#making confusion matrix
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred)
print(cm)
from sklearn.metrics import accuracy_score
print("Accuracy",accuracy_score(y_test,y_pred))
```

Q 3. Design a Django application that adds web pages with views and templates. (code not available)

->

Slip 20

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

->

```
package P2D_convUpper2lower;
public class changeCase {
   public static void main(String[] args) {

      String str1 = "Software Architecture & Design Pattern List of Assignments";
      StringBuffer newStr = new StringBuffer(str1);
      System.out.println("\nString before case conversion : ");
      System.out.println(newStr);

      for (int i = 0; i < str1.length(); i++) {
         // Checks for lower case character
         // if(Character.isLowerCase(str1.charAt(i)))
         // {
         // Convert it into upper case using toUpperCase() function
         // newStr.setCharAt(i, Character.toUpperCase(str1.charAt(i)));
         // }
         // Checks for upper case character
         // else
         if (Character.isUpperCase(str1.charAt(i))) {
            // Convert it into upper case using toLowerCase() function
            newStr.setCharAt(i, Character.toLowerCase(str1.charAt(i)));
```

```
        }
      }
      System.out.println("\n String after case conversion : ");
      System.out.println(newStr);
   }
}
```

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

->

```
# -*- coding: utf-8 -*-
"""Untitled
Automatically generated by Colaboratory.
Original file is located at
   https://colab.research.google.com/drive/1xJTHdf4gm5WGU-RyycRQh8csATZIkJ7B
"""
import numpy as np
import matplotlib.pyplot as mtp
import pandas as pd
#loding a play tennis data
PlayTennis=pd.read_csv('playtennis.csv')
PlayTennis
from sklearn.preprocessing import LabelEncoder
Le=LabelEncoder()
PlayTennis['outlook']=Le.fit_transform(PlayTennis['outlook'])
PlayTennis['temp']=Le.fit_transform(PlayTennis['temp'])
PlayTennis['humidity']=Le.fit_transform(PlayTennis['humidity'])
PlayTennis['windy']=Le.fit_transform(PlayTennis['windy'])
PlayTennis['play']=Le.fit_transform(PlayTennis['play'])
PlayTennis
y=PlayTennis['play']
x=PlayTennis.drop(['play'],axis=1)
from sklearn import tree
clf=tree.DecisionTreeClassifier(criterion='entropy')
clf=clf.fit(x,y)
#we can visualize
```

```
tree.plot_tree(clf)
```

Q 3. Develop a basic poll application (app).It should consist of two parts:

a) A public site in which user can pick their favourite programming language and vote.

b) An admin site that lets you add, change and delete programming languages.

->

(code not available)

Slip 21

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

->

```
package P5D_commandPattern;
// An interface for command
interface Command {
   public void execute();
}
// Light class and its corresponding command
// classes
class Light {
   public void on() {
      System.out.println("Light is on");
   }
   public void off() {
      System.out.println("Light is off");
   }
}
class LightOnCommand implements Command {
   Light light;
   // The constructor is passed the light it
   // is going to control.
   public LightOnCommand(Light light) {
      this.light = light;
   }
   public void execute() {
```

```
        light.on();
    }
}
class LightOffCommand implements Command {
    Light light;
    public LightOffCommand(Light light) {
        this.light = light;
    }
    public void execute() {
        light.off();
    }
}
// Stereo and its command classes
class Stereo {
    public void on() {
        System.out.println("Stereo is on");
    }
    public void off() {
        System.out.println("Stereo is off");
    }
    public void setCD() {
        System.out.println("Stereo is set " +
                "for CD input");
    }
    public void setDVD() {
        System.out.println("Stereo is set" +
                " for DVD input");
    }
    public void setRadio() {
        System.out.println("Stereo is set" +
                " for Radio");
    }
    public void setVolume(int volume) {
        // code to set the volume
```

```
        System.out.println("Stereo volume set"
                + " to " + volume);
    }
}
class StereoOffCommand implements Command {
    Stereo stereo;
    public StereoOffCommand(Stereo stereo) {
        this.stereo = stereo;
    }
    public void execute() {
        stereo.off();
    }
}
class StereoOnWithCDCommand implements Command {
    Stereo stereo;
    public StereoOnWithCDCommand(Stereo stereo) {
        this.stereo = stereo;
    }
    public void execute() {
        stereo.on();
        stereo.setCD();
        stereo.setVolume(11);
    }
}
// A Simple remote control with one button
class SimpleRemoteControl {
    Command slot; // only one button
    public SimpleRemoteControl() {
    }
    public void setCommand(Command command) {
        // set the command the remote will
        // execute
        slot = command;
    }
```

```
    public void buttonWasPressed() {
        slot.execute();
    }
}
// Driver class
class RemoteControlTest {
    public static void main(String[] args) {
        SimpleRemoteControl remote = new SimpleRemoteControl();
        Light light = new Light();
        Stereo stereo = new Stereo();
        // we can change command dynamically
        remote.setCommand(new LightOnCommand(light));
        remote.buttonWasPressed();
        remote.setCommand(new StereoOnWithCDCommand(stereo));
        remote.buttonWasPressed();
        remote.setCommand(new StereoOffCommand(stereo));
        remote.buttonWasPressed();
    }
}
```

Q 2. Write a python program to implement Linear SVM.

->

```
import numpy as np
import matplotlib.pyplot as mtp
import pandas as pd
dataset=pd.read_csv('suv_data.csv')
x=dataset.iloc[:,[2,3]].values
y=dataset.iloc[:,4].values
#splitting dataset into traning and testing dataset
from sklearn import linear_model
from sklearn.cross_validation import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25,random_state=0)
#feature scaling
from sklearn.preprocessing import StandardScaler
```

```
sc=StandardScaler()
x_train=sc.fit_transform(x_train)
x_test=sc.transform(x_test)
from sklearn.svm import SVC
classifier=SVC(kernel='linear',random_state=0)
classifier.fit(x_train,y_train)
y_pred=classifier.predict(x_test)
#making confusion matrix
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred)
print(cm)
from sklearn.metrics import accuracy_score
print("Accuracy",accuracy_score(y_test,y_pred))
#from this 8+2 incorrect prediction and 66+24 correct prediction
```

Q 3. Design a Django application: A public site in which user can pick their favourite programming language and vote.

> (code not available)

Slip 22

Q 1. Design simple HR Application using Spring Framework.

->

> (code not available)

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

->

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import preprocessing
iris = pd.read_csv("iris.csv")
#Drop id column
iris = iris.drop('Id',axis=1)
#Convert Species columns in a numerical column of the iris dataframe
#creating labelEncoder
le = preprocessing.LabelEncoder()
# Converting string labels into numbers.
iris.Species = le.fit_transform(iris.Species)
```

```
x = iris.iloc[:, :-1].values

y = iris.iloc[:, 4].values

plt.scatter(x[:,0], x[:, 3], c=y, cmap ='flag')

plt.xlabel('Sepal Length cm')

plt.ylabel('Petal Width cm')

plt.show()
```

Q 3. Design a Django application: An admin site that lets you add, change and delete programming languages.

->

(code not available)

Slip 23

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.

->

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(5);
		gumballMachine.insertQuarter();
		gumballMachine.turnCrank();
		System.out.println(gumballMachine);
	}
}
```

State.java

```
public interface State {
```

```
	public void insertQuarter();
	public void ejectQuarter();
	public void turnCrank();
	public void dispense();
	public void refill();
}
```

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");
                result.append("\nInventory: " + count + " gumball");
                if (count != 1) {
                        result.append("s");
                }
                result.append("\n");
                result.append("Machine is " + state + "\n");
                return result.toString();
        }
}
```

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";
        }
}
```

SoldState.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";
        }
}
```

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

->

```
#drop
import numpy as np
import matplotlib.pyplot as mtp
import pandas as pd
dict={'First Score':[100,90,np.nan,95],
    'Second Score':[30,45,56,np.nan],
    'Third Score':[np.nan,40,80,98],
    'Fourth Score':[80,90,np.nan,50]}
#creating a dataframe from list
df=pd.DataFrame(dict)
print(df)
df.isnull()
```

Q 3. Create your own blog using Django. (code not available)

Q 1. Write a Java Program to implement Iterator Pattern for Designing Menu like Breakfast, Lunch or Dinner Menu.

->

MenuTest.java

```
public class MenuTest {
   public static void main(String[] args) {
      PancakeHouseMenu pancakeHouseMenu = new PancakeHouseMenu();
      DinnerMenu dinnerMenu = new DinnerMenu();
      CoffeeMenu coffeeMenu = new CoffeeMenu();
      Waitress waitress = new Waitress(pancakeHouseMenu,dinnerMenu,coffeeMenu);
      waitress.printMenu();
   }
}
```

Menu.java

```
import java.util.*;
public interface Menu {
   public Iterator createIterator();
}
```

PancakeHouseMenu.java

```
import java.util.*;
public class PancakeHouseMenu implements Menu {
   ArrayList menuItems;
   public PancakeHouseMenu() {
      menuItems = new ArrayList();
      addItem("kobe's pancake breakfast","pancakes with eggs",false,2.99);
      addItem("lilei's pancake breakfast", "pancakes with toast", false, 3.59);
   }
   public void addItem(String s, String s1, boolean b, double v) {
      MenuItem menuItem = new MenuItem(s,s1,b,v);
      menuItems.add(menuItem);
   }
   public Iterator createIterator(){
      return menuItems.iterator();
   }
}
```

MenuItem.java

```
public class MenuItem {
   private String name;
   private String desc;
   private boolean vegetarian;
   private double price;
   public MenuItem(String name, String desc, boolean vegetarian, double price) {
      this.name = name;
      this.desc = desc;
      this.vegetarian = vegetarian;
      this.price = price;
   }
   public String getName() {
      return name;
   }
   public void setName(String name) {
      this.name = name;
   }
   public String getDesc() {
      return desc;
   }
   public void setDesc(String desc) {
      this.desc = desc;
   }
   public boolean isVegetarian() {
      return vegetarian;
   }
   public void setVegetarian(boolean vegetarian) {
      this.vegetarian = vegetarian;
   }
   public double getPrice() {
      return price;
   }
   public void setPrice(double price) {
```

```
        this.price = price;
    }
}
```

DinnerMenu.java

```
import java.util.*;
public class DinnerMenu implements Menu {
    private static final int MAX_SIZE = 6;
    int numOfItems = 0;
    MenuItem[] menuItems;
    public DinnerMenu() {
        menuItems = new MenuItem[MAX_SIZE];
        addItem("Vegetarian BLT","Bacon with tomato",true,2.99);
        addItem("Hot dog","With onions and cheese",false,3.05);
    }
    private void addItem(String s, String s1, boolean b, double v) {
        MenuItem menuItem = new MenuItem(s,s1,b,v);
        if(numOfItems >= MAX_SIZE){
            System.err.println("sorry,menu is full!");
        }else{
            menuItems[numOfItems]=menuItem;
            numOfItems = numOfItems + 1;
        }
    }
    @Override
    public Iterator createIterator() {
        return new DinerMenuIterator(menuItems);
    }
}
```

DinerMenuIterator.java

```
import java.util.*;
public class DinerMenuIterator implements Iterator {
    MenuItem[] list;
    int position = 0;
    public DinerMenuIterator(MenuItem[] list) {
```

```
        this.list = list;
    }
    @Override
    public boolean hasNext() {
        if(position>=list.length || list[position] == null){
            return false;
        }else{
            return true;
        }
    }
    @Override
    public Object next() {
        MenuItem menuItem = list[position];
        position = position + 1;
        return menuItem;
    }
    @Override
    public void remove() {
        if(position <=0){
            throw new IllegalStateException("now  you can not remove an item");
        }
        if(list[position] != null){
            for(int i=position-1;i<(list.length-1);i++){
                list[i] = list[i+1];
            }
            list[list.length-1]=null;
        }
    }
}
```

CoffeeMenu.java

```
import java.util.*;
public class CoffeeMenu implements Menu {
    Hashtable menuItems = new Hashtable();
    public CoffeeMenu() {
```

```
        addItem("Mocha","Han Meimei order on couple of Mocha",false,3.01);
    }
    private void addItem(String s, String s1, boolean b, double v) {
        MenuItem menuItem = new MenuItem(s,s1,b,v);
        menuItems.put(menuItem.getName(),menuItem);
    }
    @Override
    public Iterator createIterator() {
        return menuItems.values().iterator();
    }
}
```

Waitress.java

```
import java.util.*;
public class Waitress {
    Menu pancakeHouseMenu;
    Menu dinnerMenu;
    Menu coffeeMenu;
    public Waitress(Menu pancakeHouseMenu, Menu dinnerMenu, Menu coffeeMenu) {
        this.pancakeHouseMenu = pancakeHouseMenu;
        this.dinnerMenu = dinnerMenu;
        this.coffeeMenu = coffeeMenu;
    }
    public void printMenu(){
        Iterator pancakeHouseIterator = pancakeHouseMenu.createIterator();
        Iterator dinnerIterator = dinnerMenu.createIterator();
        Iterator coffeeIterator = coffeeMenu.createIterator();
        System.out.println("Menu\n=====Breakfast==start======");
        printMenu(pancakeHouseIterator);
        System.out.println("Menu\n=====Breakfast===end=====");
        System.out.println("Menu\n=====Lunch==start======");
        printMenu(dinnerIterator);
        System.out.println("Menu\n=====Lunch===end=====");
        System.out.println("Menu\n=====Coffee==start======");
        printMenu(coffeeIterator);
```

```
        System.out.println("Menu\n=====Coffee===end=====");
    }
    private void printMenu(Iterator iterator){
        while (iterator.hasNext()){
            MenuItem menuItem = (MenuItem)iterator.next();
            System.out.println(menuItem.getName()+", ");
            System.out.println(menuItem.getPrice()+", ");
            System.out.println(menuItem.getDesc());
        }
    }
}
```

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

->

```
import pandas as pd
df = pd.read_csv('salary.csv')
dummies = pd.get_dummies(df.Education)
merged = pd.concat([df, dummies], axis='columns')
merged.drop(['Education', 'Under-Graduate'], axis='columns')
print(merged)
```

Q 3. Implement Login System using Django.

> (code not available)

Slip 25

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

->

```
package P4D_SingletonPattern;
public class Test {
    public static void main(String ar[]) {
        Test1 t = new Test1();
        Test1 t2 = new Test1();
        Test1 t3 = new Test1();
```

```
        Thread tt = new Thread(t);
        Thread tt2 = new Thread(t2);
        Thread tt3 = new Thread(t3);
        Thread tt4 = new Thread(t);
        Thread tt5 = new Thread(t);
        tt.start();
        tt2.start();
        tt3.start();
        tt4.start();
        tt5.start();
    }
}
final class Test1 implements Runnable {
    @Override
    public void run() {
        for (int i = 0; i < 5; i++) {
            System.out.println(Thread.currentThread().getName() + " : " + Single.getInstance().hashCode());
        }
    }
}
class Single {
    private final static Single sing = new Single();
    private Single() {
    }
    public static Single getInstance() {
        return sing;
    }
}
```

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

->

```
    # Commented out IPython magic to ensure Python compatibility.
    import pandas as pd
```

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import linear_model
from sklearn.cross_validation import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
# %matplotlib inline
dataset=pd.read_csv('Salary_Data.csv')
print(dataset.shape)
#dataset.head(5)
#dataset.describe()in scatter plot
#data can be represented
#dataset.plot(x='YearsExperience',y='Salary',style='*')
#plt.title('yearexp vs salary')
#plt.xlabel('Yearsofexp')
#plt.ylabel('Salary')
#Text(0,0.5,'Salary')
#dependancy linear in nature then go for Linear regression model
#convert i/p values in numpy array
x=dataset['YearsExperience'].values.reshape(-1,1)
y=dataset['Salary'].values.reshape(-1,1)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
#split data into 80% training and 20% testing
#algorithm train on traning dataset using LR
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
sc.fit(x_test)
x_train=sc.transform(x_train)
x_test=sc.transform(x_test)
y_train=sc.transform(x_train)
y_test=sc.transform(y_test)
LR=LinearRegression()
LR.fit(x_train,y_train)
print("Intercept",LR.intercept_)#for retrive slope
```

```
print("coefficient",LR.coef_)



y_pred=LR.predict(x_test)
#scatter plot where the line indicate line of regression
plt.scatter(x_train,y_train)
plt.plot(x_test,y_pred,color='red')
plt.title("simple regression")
plt.xlabel("YerasofExp")
plt.ylabel("salary")
plt.show()
accuracy=LR.score(x_test,y_test)
print('Accurecy='+str(accuracy))
print(LR.predict([[1.2]]))
```

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

->

```
var http =require('http');
var url =require('url');
http.createServer(function(req,res){
   res.writeHead(200,{'content-type':'text/html'});
res.write(req.url);
res.end();
}).listen(5000);

// node server.js
```

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.

->

```
package Slip10.Q1;
/*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 q;
    }
}
class MallardDuck extends Duck {
    public MallardDuck() {
        quackBehaviour = new Quack();
        flyBehaviour = new FlyWithWings();
    }
    public void display() {
        System.out.println("I'm a real Mallard duck");
    }
}
interface FlyBehaviour {
    public void fly();
}
interface QuackBehaviour {
    public void quack() {
```

```
    System.out.println("Quack");
    }
}
class Quack implements QuackBehaviour {
    public void quack() {
        System.out.println("Quack");
    }
}
class FlyWithWings implements FlyBehaviour {
    public void fly() {
        System.out.println("I'm flying!!");
    }
}
public class Main {
    public static void main(String[] args) {
        Duck mallard = new MallardDuck();
        mallard.performQuack();
        mallard.performFly();
    }
}
*/
/* Simple Programme */
interface DuckB {
    public void oper();
}
class Fly implements DuckB {
    public void oper() {
        System.out.println("Duck Flies");
    }
}
class Quack implements DuckB {
    public void oper() {
        System.out.println("Duck Sounds Quack Quack");
    }
```

```
}
class Context {
   private DuckB s1;

   public Context(DuckB p) {
      this.s1 = p;
   }
   public void est() {
      s1.oper();
   }
}
public class Main {
   public static void main(String[] args) {
      Context c1 = new Context(new Fly());
      System.out.println("Duck Behaviour");
      c1.est();
      c1 = new Context(new Quack());
      System.out.println("Duck Behaviour ");
      c1.est();
   }
}
```

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

->

```
# Commented out IPython magic to ensure Python compatibility.
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import linear_model
from sklearn.cross_validation import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
# %matplotlib inline
df=pd.read_csv("cars.csv")
```

```
df.head(5)
x=df[['debt','income']]
y=df['sales']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
from sklearn.linear_model import LinearRegression
MLR=LinearRegression()
MLR.fit(x_train,y_train)
print("intercept",MLR.intercept_)
print("coefficient",MLR.coef_)
predict=MLR.predict([[418,7017]])
print(predict)
```

Q 3. Create a Node.js file that demonstrates create database and table in MySQL. (code not available)

Slip 27

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

->

```
interface Shape {
   void draw();
}
class RoundedRectangle implements Shape {
   public void draw() {
      System.out.println(" Inside RR");
   }
}
class RoundedSquare implements Shape {
   public void draw() {
      System.out.println(" Inside RS");
   }
}
class Rectangle implements Shape {
   public void draw() {
      System.out.println(" Inside Simple R");
   }
```

```
}
class Square implements Shape {
   public void draw() {
      System.out.println(" Inside Simple Sq");
   }
}
abstract class AbstractFactory {
   abstract Shape getShape(String st);
}
class ShapeFactory extends AbstractFactory {
   public Shape getShape(String st) {
      if (st.equalsIgnoreCase("Rectangle")) {
         return new Rectangle();
      } else if (st.equalsIgnoreCase("Square")) {
         return new Square();
      }
      return null;
   }
}
class RoundedShapeFactory extends AbstractFactory {
   public Shape getShape(String st) {
      if (st.equalsIgnoreCase("Rectangle")) {
         return new RoundedRectangle();
      } else if (st.equalsIgnoreCase("Square")) {
         return new RoundedSquare();
      }
      return null;
   }
}
class FactoryProducer {
   public static AbstractFactory getFactory(boolean rounded) {
      if (rounded) {
         return new RoundedShapeFactory();
      } else {
```

```
         return new ShapeFactory();
      }
   }
}
public class Main {
   public static void main(String[] args) {
      AbstractFactory shapeFactory = FactoryProducer.getFactory(false);
      Shape shape1 = shapeFactory.getShape("Rectangle");
      shape1.draw();
      Shape shape2 = shapeFactory.getShape("SQuare");
      shape2.draw();
      AbstractFactory shapeFactory1 = FactoryProducer.getFactory(true);
      Shape shape3 = shapeFactory1.getShape("REctangle");
      shape3.draw();
      Shape shape4 = shapeFactory1.getShape("square");
      shape4.draw();
   }
}
```

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

->

```
#polynomial regression
import numpy as np
import matplotlib.pyplot as mtp
import pandas as pd
dataset=pd.read_csv('Position_Salaries.csv')
print(dataset)
x=dataset.iloc[:,1:2].values
#print(x)
y=dataset.iloc[:,2].values
print(y)
from sklearn import linear_model
from sklearn.cross_validation import train_test_split
from sklearn.linear_model import LinearRegression
```

```
lin=LinearRegression()
lin.fit(x,y)
mtp.scatter(x,y,color='blue')
mtp.plot(x,lin.predict(x),color="red")
mtp.title("salary estimate")
mtp.xlabel("Position Levels")
mtp.ylabel("salary")
mtp.show()
from sklearn.preprocessing import PolynomialFeatures
poly=PolynomialFeatures(degree=2)
x_poly=poly.fit_transform(x)
lin_reg=LinearRegression()
lin_reg.fit(x_poly,y)
mtp.scatter(x,y,color='blue')
mtp.plot(x,linreg.predict(x_poly),color="red")
mtp.title("salary estimate")
mtp.xlabel("Position Levels")
mtp.ylabel("salary")
mtp.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. (code not available)

->

Slip 28

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()

->

DisplayElement.java

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

Observer.java

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

Subject.java

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

WeatherData.java

```
import java.util.*;
public 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;
}
```

```
}
```

CurrentConditionsDisplay.java

```
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");
   }
}
```

StatisticsDisplay.java

```
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);
    }
}
```

ForecastDisplay.java

```
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.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, rainy weather");
   }
}
```

}

Q.2. Write a python program to implement Naive Bayes.

->

```
#naive bayes
import numpy as np
import matplotlib.pyplot as mtp
import pandas as pd
dataset=pd.read_csv('suv_data.csv')
print(dataset)
x=dataset.iloc[:,[2,3]].values
print(x)
y=dataset.iloc[:,4].values
print(y)
#splitting dataset into training and testing dataset
from sklearn import linear_model
from sklearn.cross_validation import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
#feature scaling
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_train=sc.fit_transform(x_train)
x_test=sc.transform(x_test)
from sklearn.naive_bayes import GaussianNB
classifier=GaussianNB()
classifier.fit(x_train,y_train)
GaussianNB(priors=None,var_smoothing=1e-09)
#predecting the test set result
y_pred=classifier.predict(x_test)
#making confusion matrix
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred)
print(cm)
from sklearn.metrics import accuracy_score
```

```
print("Accuracy",accuracy_score(y_test,y_pred))
```

Q 3. Create your own blog using Django (code not available)

->

Slip 29

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

->

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(5);
		gumballMachine.insertQuarter();
		gumballMachine.turnCrank();
		System.out.println(gumballMachine);
	}
}
```

State.java

```
public interface State {
	public void insertQuarter();
	public void ejectQuarter();
	public void turnCrank();
	public void dispense();
	public void refill();
}
```

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

```
            if (count != 1) {
                result.append("s");
            }
            result.append("\n");
            result.append("Machine is " + state + "\n");
            return result.toString();
        }
}
```

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";
        }
}
```

SoldState.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";
        }
}
```

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

->

```
import numpy as np
import matplotlib.pyplot as mtp
import pandas as pd
#loding a play tennis data
PlayTennis=pd.read_csv('playtennis.csv')
PlayTennis
from sklearn.preprocessing import LabelEncoder
Le=LabelEncoder()
PlayTennis['outlook']=Le.fit_transform(PlayTennis['outlook'])
PlayTennis['temp']=Le.fit_transform(PlayTennis['temp'])
PlayTennis['humidity']=Le.fit_transform(PlayTennis['humidity'])
PlayTennis['windy']=Le.fit_transform(PlayTennis['windy'])
PlayTennis['play']=Le.fit_transform(PlayTennis['play'])
PlayTennis
y=PlayTennis['play']
x=PlayTennis.drop(['play'],axis=1)
from sklearn import tree
clf=tree.DecisionTreeClassifier(criterion='entropy')
clf=clf.fit(x,y)
#we can visualize
tree.plot_tree(clf)
```

Q 3. Create a clone of the “Hacker News” website. (code not available)

->

Slip 30

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.

->

Main.java

```
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("\n### Making a " + pizza.getName() + " ### \n");
		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");
    }
}
```

Q 2. Write a python program to implement Linear SVM

->

```
import numpy as np
import matplotlib.pyplot as mtp
import pandas as pd
dataset=pd.read_csv('suv_data.csv')
x=dataset.iloc[:,[2,3]].values
y=dataset.iloc[:,4].values
#splitting dataset into traning and testing dataset
from sklearn import linear_model
from sklearn.cross_validation import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25,random_state=0)
#feature scaling
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
```

```
x_train=sc.fit_transform(x_train)
x_test=sc.transform(x_test)
from sklearn.svm import SVC
classifier=SVC(kernel='linear',random_state=0)
classifier.fit(x_train,y_train)
y_pred=classifier.predict(x_test)
#making confusion matrix
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred)
print(cm)
from sklearn.metrics import accuracy_score
print("Accuracy",accuracy_score(y_test,y_pred))
#from
```

Q 3. Implement Login System using Django (code not available)

->