Slip 1

1.1

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

      String str1="Great Power";
      StringBuffer newStr=new StringBuffer(str1);

      for(int i = 0; i < str1.length(); i++) {

         if(Character.isLowerCase(str1.charAt(i))) {
            newStr.setCharAt(i, Character.toUpperCase(str1.charAt(i)));
         }
       else if(Character.isUpperCase(str1.charAt(i))) {
            newStr.setCharAt(i, Character.toLowerCase(str1.charAt(i)));
         }
       }
       System.out.println("String after case conversion : " + newStr);
   }
}
```

1.2

```
# Write a python Program to prepare scatter plot for iris dataset
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
iris = pd.read_csv("iris.csv")
print(iris.head(20))
plt.plot(iris.Id,iris["sepal.length"],"r--")
plt.show
iris.plot(kind = "scatter", x='sepal.length', y ='petal.length')
plt.show()
```

1.3

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

```
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Student Registration</title>
<style>
  body {
    font-family: Arial, sans-serif;
    background-color: #f4f4f4;
    margin: 20px;
  }

  form {
    max-width: 400px;
    margin: 20px auto;
    background: #fff;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
  }

  label {
    display: block;
    margin-bottom: 8px;
  }

  input {
    width: 100%;
    padding: 8px;
    margin-bottom: 10px;
    box-sizing: border-box;
  }
```

```
        button {
            background-color: #4caf50;
            color: #fff;
            padding: 10px 15px;
            border: none;
            border-radius: 4px;
            cursor: pointer;
        }
    </style>
</head>
<body>

    <form id="registrationForm">
        <label for="firstName">First Name:</label>
        <input type="text" id="firstName" name="firstName" required>

        <label for="lastName">Last Name:</label>
        <input type="text" id="lastName" name="lastName" required>

        <label for="age">Age:</label>
        <input type="number" id="age" name="age" required>

        <button type="button" onclick="validateForm()">Submit</button>
    </form>

    <script>
        function validateForm() {
            var firstName = document.getElementById('firstName').value;
            var lastName = document.getElementById('lastName').value;
            var age = document.getElementById('age').value;
```

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

        if (!nameRegex.test(firstName)) {
            alert('First name should contain only alphabets');
            return;
        }

        if (!nameRegex.test(lastName)) {
            alert('Last name should contain only alphabets');
            return;
        }

        if (isNaN(age) || age < 18 || age > 50) {
            alert('Age should be a number between 18 and 50');
            return;
        }

        alert('Form submitted successfully!');
    }
  </script>

</body>
</html>
```

Slip 2

2.1

```
public class Singleton {

  private Singleton() {
  }

  private static class SingletonHolder {
```

```
        private static final Singleton INSTANCE = new Singleton();
    }

    public static Singleton getInstance() {
        return SingletonHolder.INSTANCE;
    }

}
```

2.2

```
#Write a python Program to find all null values in given dataset and remove them
 import numpy as np
 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]}
df=pd.DataFrame(dict)
 print(df)
 x=df.isnull()
print(x)
 y=df.notnull()
print(y)
 z=df.fillna(0)
 print(z)
 s=df.fillna(method='pad')
 print(s)
a=df.fillna(method='bfill')
print(a)
 b=df.replace(to_replace=np.nan,value=-99)
print(b)
c=df.dropna()
print(c)
d=df.dropna(axis=1)
```

```
print(d)
 new_data=df.dropna(axis=0)
print(new_data)
```

2.3

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Employee Registration</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f4f4f4;
            margin: 20px;
        }

        form {
            max-width: 400px;
            margin: 20px auto;
            background: #fff;
            padding: 20px;
            border-radius: 8px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
        }

        label {
            display: block;
            margin-bottom: 8px;
        }
```

```
        input {
            width: 100%;
            padding: 8px;
            margin-bottom: 10px;
            box-sizing: border-box;
        }

        button {
            background-color: #4caf50;
            color: #fff;
            padding: 10px 15px;
            border: none;
            border-radius: 4px;
            cursor: pointer;
        }
    </style>
</head>
<body>

    <form id="employeeRegistrationForm">
        <label for="dob">Date of Birth:</label>
        <input type="date" id="dob" name="dob" required>

        <label for="joiningDate">Joining Date:</label>
        <input type="date" id="joiningDate" name="joiningDate" required>

        <label for="salary">Salary:</label>
        <input type="number" id="salary" name="salary" required>

        <button type="button" onclick="validateForm()">Submit</button>
    </form>
```

```
<script>
    function validateForm() {
        var dob = document.getElementById('dob').value;
        var joiningDate = document.getElementById('joiningDate').value;
        var salary = document.getElementById('salary').value;

        // Validate Date of Birth (DOB)
        var dobDate = new Date(dob);
        var currentDate = new Date();
        if (dobDate >= currentDate) {
            alert('Date of Birth should be in the past');
            return;
        }

        // Validate Joining Date
        var joiningDateDate = new Date(joiningDate);
        if (joiningDateDate > currentDate) {
            alert('Joining Date should not be in the future');
            return;
        }

        // Validate Salary
        if (isNaN(salary) || salary <= 0) {
            alert('Salary should be a positive number');
            return;
        }

        // If all validations pass, you can submit the form or perform other actions.
        alert('Form submitted successfully!');
        // Uncomment the next line to submit the form
```

```
            document.getElementById('employeeRegistrationForm').submit();
        }
    </script>

</body>
</html>
```

Slip 3

3.1

```
import java.util.Observable;
import java.util.Observer;

class WeatherData extends Observable {
    private float temperature;
    private float humidity;
    private float pressure;

    public void measurementsChanged() {
        setChanged();
        notifyObservers();
    }

    public void setMeasurements(float temperature, float humidity, float pressure) {
        this.temperature = temperature;
        this.humidity = humidity;
        this.pressure = pressure;
        measurementsChanged();
    }

    public float getTemperature() {
        return temperature;
    }
```

```
    public float getHumidity() {
        return humidity;
    }

    public float getPressure() {
        return pressure;
    }
}

class WeatherDisplay implements Observer {
    private Observable observable;

    public WeatherDisplay(Observable observable) {
        this.observable = observable;
        observable.addObserver(this);
    }

    @Override
    public void update(Observable obs, Object arg) {
        if (obs instanceof WeatherData) {
            WeatherData weatherData = (WeatherData) obs;
            display(weatherData.getTemperature(), weatherData.getHumidity(),
weatherData.getPressure());
        }
    }

    public void display(float temperature, float humidity, float pressure) {
        System.out.println("Temperature: " + temperature + " °C");
        System.out.println("Humidity: " + humidity + "%");
        System.out.println("Pressure: " + pressure + " hPa");
```

```
        System.out.println();
    }
}

public class WeatherStation {
    public static void main(String[] args) {
        WeatherData weatherData = new WeatherData();

        WeatherDisplay display1 = new WeatherDisplay(weatherData);
        WeatherDisplay display2 = new WeatherDisplay(weatherData);

        // Simulate weather changes
        weatherData.setMeasurements(25.5f, 65.2f, 1012.3f);
        weatherData.setMeasurements(28.0f, 70.5f, 1010.0f);
    }
}
```

3.2

```
#Write a python program to make categorial values in numeric format
import pandas as pd
 df=pd.read_
csv('PlayTennis.csv')
print(df)
from sklearn.preprocessing import LabelEncoder
 le=LabelEncoder()
 label=le.fit_transform(df['Play Tennis'])
print(label)
 df.drop("Play Tennis",axis=1, inplace=True)
 df["Play Tennis"]=label
print(df
```

3.3

```
<!DOCTYPE html>
```

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Login Form</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f4f4f4;
            margin: 20px;
        }

        form {
            max-width: 400px;
            margin: 20px auto;
            background: #fff;
            padding: 20px;
            border-radius: 8px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
        }

        label {
            display: block;
            margin-bottom: 8px;
        }

        input {
            width: 100%;
            padding: 8px;
            margin-bottom: 10px;
            box-sizing: border-box;
```

```
        }

        button {
            background-color: #4caf50;
            color: #fff;
            padding: 10px 15px;
            border: none;
            border-radius: 4px;
            cursor: pointer;
        }
    </style>
</head>
<body>

    <form id="loginForm">
        <label for="email">Email:</label>
        <input type="text" id="email" name="email" placeholder="Enter your email" required>

        <button type="button" onclick="validateEmail()">Login</button>
    </form>

    <script>
        function validateEmail() {
            var email = document.getElementById('email').value;
            var emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

            if (!emailRegex.test(email)) {
                alert('Please enter a valid email address');
                return;
            }
```

```
            // If the email is valid, you can perform other actions, such as submitting the form.
            alert('Email is valid!');

            // Uncomment the next line to submit the form
            // document.getElementById('loginForm').submit();
        }
    </script>

</body>
</html>
```

Slip 4

4.1

```
// Pizza interface
interface Pizza {
    void prepare();
    void bake();
    void cut();
    void box();
}

// Concrete Pizza class for NY style cheese pizza
class NyStyleCheesePizza implements Pizza {
    @Override
    public void prepare() {
        System.out.println("Preparing NY Style Cheese Pizza");
    }

    @Override
    public void bake() {
        System.out.println("Baking NY Style Cheese Pizza");
    }
```

```
    @Override
    public void cut() {
        System.out.println("Cutting NY Style Cheese Pizza");
    }

    @Override
    public void box() {
        System.out.println("Boxing NY Style Cheese Pizza");
    }
}

// Concrete Pizza class for Chicago style cheese pizza
class ChicagoStyleCheesePizza implements Pizza {
    @Override
    public void prepare() {
        System.out.println("Preparing Chicago Style Cheese Pizza");
    }

    @Override
    public void bake() {
        System.out.println("Baking Chicago Style Cheese Pizza");
    }

    @Override
    public void cut() {
        System.out.println("Cutting Chicago Style Cheese Pizza");
    }

    @Override
    public void box() {
```

```
        System.out.println("Boxing Chicago Style Cheese Pizza");
    }
}

// Pizza Store interface with the factory method createPizza()
interface PizzaStore {
    Pizza createPizza(String type);

    // Other methods like orderPizza() can be added here
}

// Concrete Pizza Store class for NY
class NyPizzaStore implements PizzaStore {
    @Override
    public Pizza createPizza(String type) {
        if ("cheese".equalsIgnoreCase(type)) {
            return new NyStyleCheesePizza();
        }
        // Add more pizza types as needed
        return null;
    }
}

// Concrete Pizza Store class for Chicago
class ChicagoPizzaStore implements PizzaStore {
    @Override
    public Pizza createPizza(String type) {
        if ("cheese".equalsIgnoreCase(type)) {
            return new ChicagoStyleCheesePizza();
        }
        // Add more pizza types as needed
```

```
        return null;
    }
}

public class PizzaStoreDemo {
    public static void main(String[] args) {
        PizzaStore nyStore = new NyPizzaStore();
        PizzaStore chicagoStore = new ChicagoPizzaStore();

        Pizza nyCheesePizza = nyStore.createPizza("cheese");
        Pizza chicagoCheesePizza = chicagoStore.createPizza("cheese");

        // Example of preparing and ordering pizzas
        nyCheesePizza.prepare();
        nyCheesePizza.bake();
        nyCheesePizza.cut();
        nyCheesePizza.box();

        System.out.println();

        chicagoCheesePizza.prepare();
        chicagoCheesePizza.bake();
        chicagoCheesePizza.cut();
        chicagoCheesePizza.box();
    }
}
```

4.2

```
# write python program to implement Simple Linear Regression for predicting house price
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
from sklearn.linear_model import LinearRegression

from sklearn.model_selection import train_test_split

from sklearn.model_selection import cross_val_predict

data = pd.read_csv(r'kc_house_data.csv')

 data.head(5);

 print(data.shape)

f = ['price','bedrooms','bathrooms','sqft_living','floors','condition','sqft_abov
e','sqft_basement','yr_built','yr_renovated']

 data = data[f]

print(data.shape)

 data = data.dropna()

print(data.shape)

 data.describe()

 X=data[f[1:]]

 y=data['price']

X_train,X_test,y_train,y_test = train_test_split(X,y,test_size =0.2,random_state=42)

print(X_train.shape)

 print(X_test.shape)

print(y_train.shape

print(y_test.shape)

lr=LinearRegression()

lr.fit(X_train,y_train)

print(lr.coef_)

 y_test_predict = lr.predict(X_test)

print(y_test_predict.shape)

g = plt.plot((y_test-y_test_predict),marker='o',linestyle="")

plt.show()
```

4.3

```
// Importing the 'readline' module to take user input

const readline = require('readline');
```

```
// Creating an interface to read input
const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout
});

// Prompt the user for input
rl.question('Enter a string: ', (inputString) => {
  // Convert the input string to uppercase
  const uppercaseString = inputString.toUpperCase();

  // Print the result
  console.log('Uppercase Output:', uppercaseString);

  // Close the interface
  rl.close();
});
```

Slip 5

5.1

```
import java.util.Enumeration;
import java.util.Iterator;

// Enumeration to Iterator Adapter
class EnumerationAdapter<T> implements Iterator<T> {
    private Enumeration<T> enumeration;

    public EnumerationAdapter(Enumeration<T> enumeration) {
        this.enumeration = enumeration;
    }

    @Override
```

```
    public boolean hasNext() {
        return enumeration.hasMoreElements();
    }

    @Override
    public T next() {
        return enumeration.nextElement();
    }

    // The remove operation is not supported in Enumeration, so it throws UnsupportedOperationException.
    @Override
    public void remove() {
        throw new UnsupportedOperationException("Remove operation not supported");
    }
}

// Example usage
public class AdapterPatternExample {
    public static void main(String[] args) {
        // Creating an Enumeration (e.g., Vector's elements())
        java.util.Vector<String> vector = new java.util.Vector<>();
        vector.add("One");
        vector.add("Two");
        vector.add("Three");
        Enumeration<String> enumeration = vector.elements();

        // Using the Enumeration to Iterator Adapter
        Iterator<String> iterator = new EnumerationAdapter<>(enumeration);

        // Iterating through elements using Iterator
```

```
        while (iterator.hasNext()) {
            System.out.println(iterator.next());
        }
    }
}
```

5.2

```
# Python program to implement Multiple Linear Regression
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
dataset=pd.read_csv('50_Startups.csv')
x=dataset.iloc[:,:-1].values
y=dataset.iloc[:,-1].values
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct=ColumnTransformer(transformers=[('encoder',OneHotEncoder(),[3]) ],remainder='passthrough')
x=np.array(ct.fit_transform(x)) print(x) from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
 from sklearn.linear_model import LinearRegression
regressor=LinearRegression()
regressor.fit(x_train,y_train)
 LinearRegression()
y_pred=regressor.predict(x_test)
 df=pd.DataFrame({'Real Values':y_test,'Predicted Values':y_pred})
print(df)
```

5.3

```
const express = require('express');
const fs = require('fs');
const bodyParser = require('body-parser');

const app = express();
```

```
const port = 3000;

// Middleware to parse form data
app.use(bodyParser.urlencoded({ extended: true }));

// Serve the HTML form
app.get('/', (req, res) => {
  res.sendFile(__dirname + '/index.html');
});

// Handle form submission
app.post('/appendFiles', (req, res) => {
  const { firstFileName, secondFileName } = req.body;

  // Read the contents of the first file
  fs.readFile(firstFileName, 'utf8', (err, data) => {
    if (err) {
      return res.status(500).send('Error reading the first file');
    }

    // Append the contents to the second file
    fs.appendFile(secondFileName, data, 'utf8', (err) => {
      if (err) {
        return res.status(500).send('Error appending contents to the second file');
      }

      res.send('Contents appended successfully!');
    });
  });
});
```

```
app.listen(port, () => {
   console.log(`Server is running on http://localhost:${port}`);
});
```

Slip 6

6.1

```
// Command interface
interface Command {
   void execute();
}

// Concrete Command classes
class LightOnCommand implements Command {
   private Light light;

   public LightOnCommand(Light light) {
      this.light = light;
   }

   @Override
   public void execute() {
      light.turnOn();
   }
}

class LightOffCommand implements Command {
   private Light light;

   public LightOffCommand(Light light) {
      this.light = light;
   }
```

```
    @Override
    public void execute() {
        light.turnOff();
    }
}

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

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

// Invoker class
class RemoteControl {
    private Command command;

    public void setCommand(Command command) {
        this.command = command;
    }

    public void pressButton() {
        command.execute();
    }
}

// Client class to test the Remote Control with Command Pattern
```

```
public class RemoteControlTest {
    public static void main(String[] args) {
        // Creating the Light and the corresponding Command objects
        Light livingRoomLight = new Light();
        LightOnCommand livingRoomLightOn = new LightOnCommand(livingRoomLight);
        LightOffCommand livingRoomLightOff = new LightOffCommand(livingRoomLight);

        // Creating the Remote Control
        RemoteControl remoteControl = new RemoteControl();

        // Setting the command for the remote control
        remoteControl.setCommand(livingRoomLightOn);

        // Pressing the button on the remote control
        remoteControl.pressButton();

        // Changing the command for the remote control
        remoteControl.setCommand(livingRoomLightOff);

        // Pressing the button again on the remote control
        remoteControl.pressButton();
    }
}
```

6.2

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
dataset=pd.read_csv('Position_Salaries.csv')
x=dataset.iloc[:,1:-1].values
y=dataset.iloc[:,-1].values
print(dataset.head(5))
```

```
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
 p_r=PolynomialFeatures(degree=4)
x_poly=p_r.fit_transform(x)
lin_reg=LinearRegression()
lin_reg.fit(x_poly,y)
LinearRegression()
y_pred=lin_reg.predict(x_poly)
df=pd.DataFrame({'Real Values':y,'Predicted Values':y_pred})
print(df)
x_grid=np.arange(min(x),max(x),0.1)
x_grid=x_grid.reshape((len(x_grid),1))
plt.scatter(x,y,color='yellow')
plt.scatter(x,y_pred,color='red')
plt.plot(x_grid,lin_reg.predict(p_r.fit_transform(x_grid)),color='black')
plt.title('Polynomial Regression')
plt.xlabel('position level')
plt.ylabel('Salary')
plt.show()
```

6.3

```
const http = require('http');
const fs = require('fs');
const path = require('path');

const server = http.createServer((req, res) => {
   // Extract the requested file path from the URL
   const filePath = path.join(__dirname, req.url);

   // Read the file and send its content to the client
   fs.readFile(filePath, 'utf8', (err, data) => {
      if (err) {
```

```
        // Handle 404 error if the file is not found
        if (err.code === 'ENOENT') {
            res.writeHead(404, { 'Content-Type': 'text/plain' });
            res.end('404 Not Found');
        } else {
            // Handle other errors
            res.writeHead(500, { 'Content-Type': 'text/plain' });
            res.end('500 Internal Server Error');
        }
    } else {
        // Send the file content to the client
        res.writeHead(200, { 'Content-Type': 'text/plain' });
        res.end(data);
    }
  });
});

const PORT = 3000;

server.listen(PORT, () => {
  console.log(`Server is running on http://localhost:${PORT}`);
});
```

Slip 7

7.1

```
// Receiver class
class CeilingFan {
  private String location;
  private int speed;

  public CeilingFan(String location) {
    this.location = location;
```

```
        this.speed = 0;
    }

    public void turnOn() {
        System.out.println(location + " Ceiling Fan is ON");
    }

    public void turnOff() {
        System.out.println(location + " Ceiling Fan is OFF");
    }

    public void setSpeed(int speed) {
        this.speed = speed;
        System.out.println(location + " Ceiling Fan speed set to " + speed);
    }

    public int getSpeed() {
        return speed;
    }
}

// Command interface
interface CeilingFanCommand {
    void execute();
    void undo();
}

// Concrete Command class for changing fan speed
class CeilingFanSpeedCommand implements CeilingFanCommand {
    private CeilingFan ceilingFan;
    private int previousSpeed;
```

```
    public CeilingFanSpeedCommand(CeilingFan ceilingFan) {
        this.ceilingFan = ceilingFan;
    }

    @Override
    public void execute() {
        previousSpeed = ceilingFan.getSpeed();
        ceilingFan.setSpeed(previousSpeed + 1);
    }

    @Override
    public void undo() {
        ceilingFan.setSpeed(previousSpeed);
    }
}

// Invoker class
class CeilingFanRemote {
    private CeilingFanCommand command;

    public void setCommand(CeilingFanCommand command) {
        this.command = command;
    }

    public void pressButton() {
        command.execute();
    }

    public void pressUndoButton() {
        command.undo();
```

```
    }
}

// Client class to test Ceiling Fan with undo command
public class CeilingFanTest {
    public static void main(String[] args) {
        // Creating the Ceiling Fan and the corresponding Command objects
        CeilingFan livingRoomCeilingFan = new CeilingFan("Living Room");
        CeilingFanSpeedCommand increaseSpeedCommand = new
CeilingFanSpeedCommand(livingRoomCeilingFan);

        // Creating the Ceiling Fan Remote
        CeilingFanRemote remote = new CeilingFanRemote();

        // Setting the command for the remote control
        remote.setCommand(increaseSpeedCommand);

        // Pressing the button on the remote control
        remote.pressButton();

        // Pressing the undo button on the remote control
        remote.pressUndoButton();
    }
}
```

7.2

```
from sklearn import datasets
from sklearn import metrics
from sklearn.naive_bayes import GaussianNB
dataset = datasets.load_iris()
model=GaussianNB()
model.fit(dataset.data,dataset.target)
```

```
expected=dataset.target
predicted=model.predict(dataset.data)
print(metrics.classification_report(expected,predicted))
print(metrics.confusion_matrix(expected,predicted))
```

7.3 a

```
const express = require('express');
const multer = require('multer');
const path = require('path');

const app = express();
const port = 3000;

// Set up the storage engine for multer
const storage = multer.diskStorage({
  destination: (req, file, cb) => {
    cb(null, 'uploads/'); // Set the destination folder for uploads
  },
  filename: (req, file, cb) => {
    const uniqueSuffix = Date.now() + '-' + Math.round(Math.random() * 1E9);
    const extension = path.extname(file.originalname);
    cb(null, file.fieldname + '-' + uniqueSuffix + extension);
  }
});

// Create the multer middleware
const upload = multer({ storage: storage });

// Serve the HTML form with an upload field
app.get('/', (req, res) => {
  res.sendFile(__dirname + '/index.html');
});
```

```
// Handle file uploads
app.post('/upload', upload.single('file'), (req, res) => {
    const uploadedFile = req.file;
    if (!uploadedFile) {
        return res.status(400).send('No file uploaded.');
    }

    res.send(`File uploaded successfully: ${uploadedFile.filename}`);
});

app.listen(port, () => {
    console.log(`Server is running on http://localhost:${port}`);
});
```

7.3 b

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>File Upload Form</title>
</head>
<body>
    <h1>File Upload Form</h1>
    <form action="/upload" method="post" enctype="multipart/form-data">
        <label for="file">Choose a file:</label>
        <input type="file" id="file" name="file" required>
        <br>
        <button type="submit">Upload File</button>
    </form>
</body>
```

```
</html>
```

Slip 8

8.1

```
// GumballMachine class representing the Context
public class GumballMachine {
    private State soldOutState;
    private State noQuarterState;
    private State hasQuarterState;
    private State soldState;

    private State currentState;
    private int count = 0;

    public GumballMachine(int numberOfGumballs) {
        soldOutState = new SoldOutState(this);
        noQuarterState = new NoQuarterState(this);
        hasQuarterState = new HasQuarterState(this);
        soldState = new SoldState(this);

        this.count = numberOfGumballs;
        if (numberOfGumballs > 0) {
            currentState = noQuarterState;
        } else {
            currentState = soldOutState;
        }
    }

    // Actions and behaviors
    public void insertQuarter() {
        currentState.insertQuarter();
```

```
}

public void ejectQuarter() {
    currentState.ejectQuarter();
}

public void turnCrank() {
    currentState.turnCrank();
    currentState.dispense();
}

public void releaseBall() {
    System.out.println("A gumball comes rolling out the slot...");
    if (count != 0) {
        count--;
    }
}

// Other methods
public void refill(int numberOfGumballs) {
    this.count = numberOfGumballs;
    currentState = noQuarterState;
}

// Getters and setters
public State getSoldOutState() {
    return soldOutState;
}

public State getNoQuarterState() {
    return noQuarterState;
```

```
}

public State getHasQuarterState() {
   return hasQuarterState;
}

public State getSoldState() {
   return soldState;
}

public int getCount() {
   return count;
}

public void setState(State state) {
   this.currentState = state;
}

public State getState() {
   return currentState;
}

public static void main(String[] args) {
   GumballMachine gumballMachine = new GumballMachine(5);

   // Test the Gumball Machine
   gumballMachine.insertQuarter();
   gumballMachine.turnCrank();

   gumballMachine.insertQuarter();
   gumballMachine.ejectQuarter();
```

```
        gumballMachine.turnCrank();

        gumballMachine.insertQuarter();
        gumballMachine.turnCrank();
        gumballMachine.insertQuarter();
        gumballMachine.turnCrank();
        gumballMachine.refill(10);
        gumballMachine.insertQuarter();
        gumballMachine.turnCrank();
    }
}
```

8.2

```
#python program to implement Decision Tree whether or not to play Tennis
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
playTennis = pd.read_csv('/home/pc10/ML_Slips_Solution/PlayTennis.csv')
from sklearn.preprocessing import LabelEncoder
Le=LabelEncoder()
playTennis['Outlook']=Le.fit_transform(playTennis['Outlook'])
playTennis['Temperature']=Le.fit_transform(playTennis['Temperature'] )
playTennis['Humidity']=Le.fit_transform(playTennis['Humidity'])
playTennis['Wind']=Le.fit_transform(playTennis['Wind'])
playTennis['Play Tennis']=Le.fit_transform(playTennis['Play Tennis'])
print(playTennis)
y=playTennis['Play Tennis']
x=playTennis.drop(['Play Tennis'],axis=1)
from sklearn import tree
clf = tree.DecisionTreeClassifier(criterion='entropy')
clf=clf.fit(x,y)
fig = plt.figure(figsize=(25,20)) _ = tree.plot_tree(clf, filled=True)
```

```
plt.show()
```

8.3

```
const mysql = require('mysql');

// MySQL connection configuration
const connection = mysql.createConnection({
  host: 'localhost',
  user: 'your_username',
  password: 'your_password',
  database: 'your_database_name',
});

// Connect to MySQL server
connection.connect((err) => {
  if (err) {
    console.error('Error connecting to MySQL server:', err.message);
    return;
  }

  console.log('Connected to MySQL server');

  // Create a new database
  const createDatabaseQuery = 'CREATE DATABASE IF NOT EXISTS mydatabase';
  connection.query(createDatabaseQuery, (err) => {
    if (err) {
      console.error('Error creating database:', err.message);
      return;
    }

    console.log('Database created or already exists');
```

```
// Use the newly created database
connection.changeUser({ database: 'mydatabase' }, (err) => {
 if (err) {
  console.error('Error selecting database:', err.message);
  return;
 }

 console.log('Using database: mydatabase');

 // Create a new table
 const createTableQuery = `
  CREATE TABLE IF NOT EXISTS users (
   id INT PRIMARY KEY AUTO_INCREMENT,
   username VARCHAR(255) NOT NULL,
   email VARCHAR(255) NOT NULL
  )
 `;
 connection.query(createTableQuery, (err) => {
  if (err) {
   console.error('Error creating table:', err.message);
   return;
  }

  console.log('Table created or already exists');

  // Close the MySQL connection
  connection.end((err) => {
   if (err) {
    console.error('Error closing connection:', err.message);
    return;
   }
```

```
        console.log('Connection closed');
      });
    });
  });
});
```

Slip 9

9.1

```
// Strategy interface for flying behavior
interface FlyBehavior {
    void fly();
}

// Concrete implementations of flying behavior
class FlyWithWings implements FlyBehavior {
    @Override
    public void fly() {
        System.out.println("Flying with wings");
    }
}

class FlyNoWay implements FlyBehavior {
    @Override
    public void fly() {
        System.out.println("Unable to fly");
    }
}

// Strategy interface for quacking behavior
interface QuackBehavior {
```

```
    void quack();
}

// Concrete implementations of quacking behavior
class Quack implements QuackBehavior {
    @Override
    public void quack() {
        System.out.println("Quack");
    }
}

class MuteQuack implements QuackBehavior {
    @Override
    public void quack() {
        System.out.println("<< Silence >>");
    }
}

// Context class (Duck)
class Duck {
    private FlyBehavior flyBehavior;
    private QuackBehavior quackBehavior;

    public Duck(FlyBehavior flyBehavior, QuackBehavior quackBehavior) {
        this.flyBehavior = flyBehavior;
        this.quackBehavior = quackBehavior;
    }

    public void performFly() {
        flyBehavior.fly();
    }
```

```
    public void performQuack() {
        quackBehavior.quack();
    }

    public void swim() {
        System.out.println("All ducks float, even decoys!");
    }

    // Setter methods to change behaviors dynamically
    public void setFlyBehavior(FlyBehavior flyBehavior) {
        this.flyBehavior = flyBehavior;
    }

    public void setQuackBehavior(QuackBehavior quackBehavior) {
        this.quackBehavior = quackBehavior;
    }
}

// Test program
public class DuckBehaviorTest {
    public static void main(String[] args) {
        // Create a Mallard Duck with flying and quacking behaviors
        Duck mallardDuck = new Duck(new FlyWithWings(), new Quack());

        System.out.println("Mallard Duck:");
        mallardDuck.performFly();
        mallardDuck.performQuack();
        mallardDuck.swim();

        // Change Mallard Duck's flying behavior dynamically
```

```
        mallardDuck.setFlyBehavior(new FlyNoWay());
        System.out.println("Mallard Duck (after changing flying behavior):");
        mallardDuck.performFly();
        mallardDuck.performQuack();
        mallardDuck.swim();

        // Create a Rubber Duck with different flying and quacking behaviors
        Duck rubberDuck = new Duck(new FlyNoWay(), new MuteQuack());

        System.out.println("\nRubber Duck:");
        rubberDuck.performFly();
        rubberDuck.performQuack();
        rubberDuck.swim();
    }
}
```

9.2

```
# Write a python Program to prepare scatter plot for iris dataset
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
iris = pd.read_csv("iris.csv")
print(iris.head(20))
plt.plot(iris.Id,iris["sepal.length"],"r--")
plt.show
iris.plot(kind = "scatter", x='sepal.length', y ='petal.length')
plt.show()
```

9.3

```
const mysql = require('mysql');
// MySQL connection configuration
const connection = mysql.createConnection({
  host: 'localhost',
  user: 'your_username',
```

```
  password: 'your_password',
  database: 'your_database_name',
});

// Connect to MySQL server
connection.connect((err) => {
  if (err) {
    console.error('Error connecting to MySQL server:', err.message);
    return;
  }

  console.log('Connected to MySQL server');

  // Insert multiple records into the "student" table
  const students = [
    { name: 'John Doe', age: 20, grade: 'A' },
    { name: 'Jane Smith', age: 22, grade: 'B' },
    { name: 'Bob Johnson', age: 21, grade: 'C' },
  ];

  const insertQuery = 'INSERT INTO student (name, age, grade) VALUES ?';
  connection.query(insertQuery, [students.map(student => [student.name, student.age,
student.grade])], (err, result) => {
    if (err) {
      console.error('Error inserting records:', err.message);
      connection.end();
      return;
    }

    console.log(`${result.affectedRows} records inserted into the "student" table`);
    console.log('Result object:', result);
```

```
  // Close the MySQL connection
  connection.end((err) => {
    if (err) {
      console.error('Error closing connection:', err.message);
    } else {
      console.log('Connection closed');
    }
  });
 });
});
```

Slip 11

11.1

```
// Heart Model (Adaptee)
class HeartModel {
  private boolean beating;

  public void startBeating() {
    System.out.println("Heart is beating");
    beating = true;
  }

  public void stopBeating() {
    System.out.println("Heart has stopped beating");
    beating = false;
  }

  public boolean isBeating() {
    return beating;
  }
}
```

```
// Beat Model (Target)
interface BeatModel {
    void start();
    void stop();
    boolean isRunning();
}

// Adapter class to adapt Heart Model to Beat Model
class HeartAdapter implements BeatModel {
    private HeartModel heartModel;

    public HeartAdapter(HeartModel heartModel) {
        this.heartModel = heartModel;
    }

    @Override
    public void start() {
        heartModel.startBeating();
    }

    @Override
    public void stop() {
        heartModel.stopBeating();
    }

    @Override
    public boolean isRunning() {
        return heartModel.isBeating();
    }
}
```

```
// Test program
public class AdapterPatternTest {
   public static void main(String[] args) {
      // Create an instance of Heart Model
      HeartModel heartModel = new HeartModel();

      // Create an adapter for the Heart Model to conform to Beat Model interface
      BeatModel beatModel = new HeartAdapter(heartModel);

      // Test the Beat Model
      System.out.println("Start the Beat Model:");
      beatModel.start();
      System.out.println("Is the Beat Model running? " + beatModel.isRunning());

      System.out.println("\nStop the Beat Model:");
      beatModel.stop();
      System.out.println("Is the Beat Model running? " + beatModel.isRunning());
   }
}
```

11.2

```
#Write a python Program to find all null values in given dataset and remove them
 import numpy as np
 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]}
df=pd.DataFrame(dict)
 print(df)
 x=df.isnull()
print(x)
 y=df.notnull()
```

```
print(y)
 z=df.fillna(0)
 print(z)
 s=df.fillna(method='pad')
 print(s)
a=df.fillna(method='bfill')
print(a)
 b=df.replace(to_replace=np.nan,value=-99)
print(b)
c=df.dropna()
print(c)
d=df.dropna(axis=1)
print(d)
 new_data=df.dropna(axis=0)
print(new_data)
```

11.3

```
const mysql = require('mysql');

// MySQL connection configuration
const connection = mysql.createConnection({
  host: 'localhost',
  user: 'your_username',
  password: 'your_password',
  database: 'your_database_name',
});

// Connect to MySQL server
connection.connect((err) => {
  if (err) {
    console.error('Error connecting to MySQL server:', err.message);
    return;
```

```
}

console.log('Connected to MySQL server');

// Select all records from the "customers" table
const selectQuery = 'SELECT * FROM customers';
connection.query(selectQuery, (err, rows) => {
  if (err) {
    console.error('Error selecting records:', err.message);
    connection.end();
    return;
  }

  console.log('All records from the "customers" table:');
  console.table(rows);

  // Delete a specific record from the "customers" table
  const customerIdToDelete = 3; // Replace with the ID of the record you want to delete
  const deleteQuery = 'DELETE FROM customers WHERE id = ?';
  connection.query(deleteQuery, [customerIdToDelete], (err, result) => {
    if (err) {
      console.error('Error deleting record:', err.message);
      connection.end();
      return;
    }

    console.log(`${result.affectedRows} record(s) deleted from the "customers" table`);

    // Select all records again to verify the deletion
    connection.query(selectQuery, (err, updatedRows) => {
      if (err) {
```

```
        console.error('Error selecting records after deletion:', err.message);
        connection.end();
        return;
      }

      console.log('All records after deletion:');
      console.table(updatedRows);

      // Close the MySQL connection
      connection.end((err) => {
        if (err) {
          console.error('Error closing connection:', err.message);
        } else {
          console.log('Connection closed');
        }
      });
    });
  });
});
```

Slip 12

12.1

```
// Car interface
interface Car {
    void assemble();
}

// Concrete implementation of Car
class BasicCar implements Car {
    @Override
    public void assemble() {
```

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

// Decorator pattern for SportsCar
class SportsCar implements Car {
    private Car car;

    public SportsCar(Car car) {
        this.car = car;
    }

    @Override
    public void assemble() {
        car.assemble();
        System.out.println("Adding features of Sports Car");
    }
}

// Decorator pattern for LuxuryCar
class LuxuryCar implements Car {
    private Car car;

    public LuxuryCar(Car car) {
        this.car = car;
    }

    @Override
    public void assemble() {
        car.assemble();
        System.out.println("Adding features of Luxury Car");
```

```
    }
}

// Main class to test the Decorator Pattern
public class DecoratorPatternExample {
    public static void main(String[] args) {
        // Create a basic car
        Car basicCar = new BasicCar();

        // Decorate the basic car with SportsCar features
        Car sportsCar = new SportsCar(basicCar);

        // Decorate the sports car with LuxuryCar features
        Car luxurySportsCar = new LuxuryCar(sportsCar);

        // Assemble the final decorated car
        luxurySportsCar.assemble();
    }
}
```

12.2

```
#Write a python program to make categorial values in numeric format
import pandas as pd
 df=pd.read_
csv('PlayTennis.csv')
print(df)
from sklearn.preprocessing import LabelEncoder
 le=LabelEncoder()
 label=le.fit_transform(df['Play Tennis'])
print(label)
 df.drop("Play Tennis",axis=1, inplace=True)
 df["Play Tennis"]=label
```

```
print(df
```

12.3

```
// Import the 'http' module to create an HTTP server
const http = require('http');

// Configure the HTTP server to respond with "Hello, World!" to all requests
const server = http.createServer((req, res) => {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello, World!\n');
});

// Listen on port 3000, and IP address defaults to 127.0.0.1
const PORT = 3000;
const IP = '127.0.0.1';

server.listen(PORT, IP, () => {
  console.log(`Server running at http://${IP}:${PORT}/`);
});
```

Slip 13

13.1

```
// Volt class to measure volts
class Volt {
    private int volts;

    public Volt(int volts) {
        this.volts = volts;
    }

    public int getVolts() {
        return volts;
```

```
    }
}

// Socket class producing constant volts of 120V
class Socket {
    public Volt getVolt() {
        return new Volt(120);
    }
}

// Adapter interface
interface SocketAdapter {
    Volt get3Volts();
    Volt get12Volts();
    Volt getDefaultVolts();
}

// Class Adapter implementing the SocketAdapter interface
class SocketClassAdapter extends Socket implements SocketAdapter {
    @Override
    public Volt get3Volts() {
        Volt volt = getVolt();
        return new Volt(volt.getVolts() / 40); // Dividing by 40 to get 3 volts
    }

    @Override
    public Volt get12Volts() {
        Volt volt = getVolt();
        return new Volt(volt.getVolts() / 10); // Dividing by 10 to get 12 volts
    }
```

```
    @Override
    public Volt getDefaultVolts() {
        return getVolt();
    }
}

// Client code to test the Adapter pattern
public class AdapterPatternExample {
    public static void main(String[] args) {
        SocketAdapter socketAdapter = new SocketClassAdapter();

        Volt volt3 = socketAdapter.get3Volts();
        System.out.println("3 Volts: " + volt3.getVolts());

        Volt volt12 = socketAdapter.get12Volts();
        System.out.println("12 Volts: " + volt12.getVolts());

        Volt defaultVolt = socketAdapter.getDefaultVolts();
        System.out.println("Default Volts: " + defaultVolt.getVolts());
    }
}
```

13.2

```
# Write a python Program to prepare scatter plot for iris dataset
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
iris = pd.read_csv("iris.csv")
print(iris.head(20))
plt.plot(iris.Id,iris["sepal.length"],"r--")
plt.show
iris.plot(kind = "scatter", x='sepal.length', y ='petal.length')
plt.show()
```

13.3

```
const express = require('express');
const mongoose = require('mongoose');
const bcrypt = require('bcrypt');
const session = require('express-session');
const MongoStore = require('connect-mongo')(session);

const app = express();
const PORT = 3000;

// Connect to MongoDB
mongoose.connect('mongodb://localhost/user_login_system', {
  useNewUrlParser: true,
  useUnifiedTopology: true,
});

// Define User schema
const userSchema = new mongoose.Schema({
  username: String,
  password: String,
});

const User = mongoose.model('User', userSchema);

// Middleware
app.use(express.urlencoded({ extended: true }));
app.use(
  session({
    secret: 'your-secret-key',
    resave: false,
    saveUninitialized: true,
```

```
    store: new MongoStore({ mongooseConnection: mongoose.connection }),
  })
);

// Routes
app.get('/', (req, res) => {
  res.send('Welcome to the User Login System');
});

app.get('/login', (req, res) => {
  res.send('Login Page');
});

app.post('/login', async (req, res) => {
  const { username, password } = req.body;

  try {
    const user = await User.findOne({ username });

    if (user && bcrypt.compareSync(password, user.password)) {
      req.session.user = user;
      res.redirect('/dashboard');
    } else {
      res.send('Invalid username or password');
    }
  } catch (error) {
    res.status(500).send('Internal Server Error');
  }
});

app.get('/dashboard', (req, res) => {
```

```
  if (req.session.user) {
    res.send(`Welcome ${req.session.user.username} to the Dashboard`);
  } else {
    res.redirect('/login');
  }
});

// Server
app.listen(PORT, () => {
  console.log(`Server is running on http://localhost:${PORT}`);
});
```

Slip 14

14.1

```
// Command interface with execute method
interface Command {
    void execute();
}

// Receiver class - Light
class Light {
    public void turnOn() {
        System.out.println("Light is ON");
    }

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

// Concrete Command - LightOnCommand
class LightOnCommand implements Command {
```

```
    private Light light;

    public LightOnCommand(Light light) {
        this.light = light;
    }

    @Override
    public void execute() {
        light.turnOn();
    }
}

// Concrete Command - LightOffCommand
class LightOffCommand implements Command {
    private Light light;

    public LightOffCommand(Light light) {
        this.light = light;
    }

    @Override
    public void execute() {
        light.turnOff();
    }
}

// Receiver class - GarageDoor
class GarageDoor {
    public void up() {
        System.out.println("Garage door is UP");
    }
```

```
}

// Concrete Command - GarageDoorUpCommand
class GarageDoorUpCommand implements Command {
    private GarageDoor garageDoor;

    public GarageDoorUpCommand(GarageDoor garageDoor) {
        this.garageDoor = garageDoor;
    }

    @Override
    public void execute() {
        garageDoor.up();
    }
}

// Receiver class - Stereo
class Stereo {
    public void onWithCD() {
        System.out.println("Stereo is ON with CD");
    }
}

// Concrete Command - StereoOnWithCDCommand
class StereoOnWithCDCommand implements Command {
    private Stereo stereo;

    public StereoOnWithCDCommand(Stereo stereo) {
        this.stereo = stereo;
    }
```

```
    @Override
    public void execute() {
        stereo.onWithCD();
    }
}

// Invoker class
class RemoteControl {
    private Command command;

    public void setCommand(Command command) {
        this.command = command;
    }

    public void pressButton() {
        command.execute();
    }
}

// Client code to test the Command Design Pattern
public class CommandPatternExample {
    public static void main(String[] args) {
        // Create instances of receivers
        Light light = new Light();
        GarageDoor garageDoor = new GarageDoor();
        Stereo stereo = new Stereo();

        // Create instances of concrete commands
        Command lightOnCommand = new LightOnCommand(light);
        Command lightOffCommand = new LightOffCommand(light);
        Command garageDoorUpCommand = new GarageDoorUpCommand(garageDoor);
```

```
        Command stereoOnWithCDCommand = new StereoOnWithCDCommand(stereo);

        // Create invokers and set commands
        RemoteControl remoteControl1 = new RemoteControl();
        remoteControl1.setCommand(lightOnCommand);

        RemoteControl remoteControl2 = new RemoteControl();
        remoteControl2.setCommand(lightOffCommand);

        RemoteControl remoteControl3 = new RemoteControl();
        remoteControl3.setCommand(garageDoorUpCommand);

        RemoteControl remoteControl4 = new RemoteControl();
        remoteControl4.setCommand(stereoOnWithCDCommand);

        // Press buttons to execute commands
        remoteControl1.pressButton(); // Turns on the light
        remoteControl2.pressButton(); // Turns off the light
        remoteControl3.pressButton(); // Opens the garage door
        remoteControl4.pressButton(); // Turns on the stereo with CD
    }
}
```

14.2

```
#Write a python Program to find all null values in given dataset and remove them
 import numpy as np
 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]}
df=pd.DataFrame(dict)
 print(df)
 x=df.isnull()
```

```
print(x)
 y=df.notnull()
print(y)
 z=df.fillna(0)
 print(z)
 s=df.fillna(method='pad')
 print(s)
a=df.fillna(method='bfill')
print(a)
 b=df.replace(to_replace=np.nan,value=-99)
print(b)
c=df.dropna()
print(c)
d=df.dropna(axis=1)
print(d)
 new_data=df.dropna(axis=0)
print(new_data)
```

14.3

Html code

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Simple Web Page</title>
</head>
<body>
    <h1>Hello, World!</h1>
    <p>This is a simple web page served by a Node.js script.</p>
</body>
</html>
```

Js code

```
const http = require('http');
const fs = require('fs');
const path = require('path');

const server = http.createServer((req, res) => {
   // Set the content type to HTML
   res.writeHead(200, { 'Content-Type': 'text/html' });

   // Read the HTML file and stream it to the response
   const filePath = path.join(__dirname, 'index.html');
   const readStream = fs.createReadStream(filePath);

   // Pipe the read stream to the response stream
   readStream.pipe(res);

   // Handle errors
   readStream.on('error', (error) => {
      console.error('Error reading file:', error.message);
      res.writeHead(500, { 'Content-Type': 'text/plain' });
      res.end('Internal Server Error');
   });
});

const PORT = 3000;

server.listen(PORT, () => {
   console.log(`Server running at http://localhost:${PORT}/`);
});
```

Slip 15

15.1

```
// Subsystem 1: DVD Player
class DVDPlayer {
    public void on() {
        System.out.println("DVD Player is ON");
    }

    public void play(String movie) {
        System.out.println("Playing movie: " + movie);
    }

    public void off() {
        System.out.println("DVD Player is OFF");
    }
}

// Subsystem 2: Projector
class Projector {
    public void on() {
        System.out.println("Projector is ON");
    }

    public void setInput(DVDPlayer dvdPlayer) {
        System.out.println("Setting input to DVD Player");
    }

    public void off() {
        System.out.println("Projector is OFF");
    }
}
```

```
// Subsystem 3: Lights
class Lights {
    public void dim() {
        System.out.println("Dimming the lights");
    }

    public void brighten() {
        System.out.println("Brightening the lights");
    }
}

// Facade: HomeTheaterFacade
class HomeTheaterFacade {
    private DVDPlayer dvdPlayer;
    private Projector projector;
    private Lights lights;

    public HomeTheaterFacade(DVDPlayer dvdPlayer, Projector projector, Lights lights) {
        this.dvdPlayer = dvdPlayer;
        this.projector = projector;
        this.lights = lights;
    }

    public void watchMovie(String movie) {
        System.out.println("Get ready to watch a movie!");
        lights.dim();
        projector.on();
        projector.setInput(dvdPlayer);
        dvdPlayer.on();
        dvdPlayer.play(movie);
    }
```

```
    public void endMovie() {
        System.out.println("Shutting down the home theater");
        dvdPlayer.off();
        projector.off();
        lights.brighten();
    }
}

// Client code to test the Facade Design Pattern
public class FacadePatternExample {
    public static void main(String[] args) {
        DVDPlayer dvdPlayer = new DVDPlayer();
        Projector projector = new Projector();
        Lights lights = new Lights();

        HomeTheaterFacade homeTheater = new HomeTheaterFacade(dvdPlayer, projector, lights);

        // Watch a movie
        homeTheater.watchMovie("Inception");

        // End the movie
        homeTheater.endMovie();
    }
}
```

15.2

```
#Write a python program to make categorial values in numeric format
import pandas as pd
 df=pd.read_
csv('PlayTennis.csv')
print(df)
```

```
from sklearn.preprocessing import LabelEncoder
 le=LabelEncoder()
 label=le.fit_transform(df['Play Tennis'])
print(label)
 df.drop("Play Tennis",axis=1, inplace=True)
 df["Play Tennis"]=label
print(df
```

15.3

Module.js

```
// modules.js

// Function to return today's date and time
function getCurrentDateTime() {
   const currentDate = new Date();
   return currentDate.toLocaleString();
}

// Export the function to make it available externally
module.exports = {
   getCurrentDateTime: getCurrentDateTime
};
```

Server.js

```
// server.js

// Import the modules.js module
const myModule = require('./modules');

// Import the built-in http module
const http = require('http');

// Create a local server
```

```
const server = http.createServer((req, res) => {
   res.writeHead(200, {'Content-Type': 'text/plain'});

   // Use the function from modules.js to get the current date and time
   const dateTime = myModule.getCurrentDateTime();

   res.end(`Current Date and Time: ${dateTime}`);
});

// Set the server to listen on port 3000
const PORT = 3000;
server.listen(PORT, () => {
   console.log(`Server running at http://localhost:${PORT}/`);
});
```

Slip 17

17.1

```
// Abstract Product: Shape interface
interface Shape {
   void draw();
}

// Concrete Products: Circle, Square, Rectangle
class Circle implements Shape {
   @Override
   public void draw() {
      System.out.println("Drawing Circle");
   }
}

class Square implements Shape {
   @Override
```

```
    public void draw() {
        System.out.println("Drawing Square");
    }
}

class Rectangle implements Shape {
    @Override
    public void draw() {
        System.out.println("Drawing Rectangle");
    }
}

// Abstract Factory: ShapeFactory interface
interface ShapeFactory {
    Shape createShape();
}

// Concrete Factories: CircleFactory, SquareFactory, RectangleFactory
class CircleFactory implements ShapeFactory {
    @Override
    public Shape createShape() {
        return new Circle();
    }
}

class SquareFactory implements ShapeFactory {
    @Override
    public Shape createShape() {
        return new Square();
    }
}
```

```
class RectangleFactory implements ShapeFactory {
    @Override
    public Shape createShape() {
        return new Rectangle();
    }
}

// Client Code
public class AbstractFactoryPatternExample {
    public static void main(String[] args) {
        // Create factories
        ShapeFactory circleFactory = new CircleFactory();
        ShapeFactory squareFactory = new SquareFactory();
        ShapeFactory rectangleFactory = new RectangleFactory();

        // Create shapes using the factories
        Shape circle = circleFactory.createShape();
        Shape square = squareFactory.createShape();
        Shape rectangle = rectangleFactory.createShape();

        // Draw shapes
        circle.draw();     // Output: Drawing Circle
        square.draw();     // Output: Drawing Square
        rectangle.draw();  // Output: Drawing Rectangle
    }
}
```

17.2

```
# Python program to implement Multiple Linear Regression
import numpy as np
import matplotlib.pyplot as plt
```

```
import pandas as pd
dataset=pd.read_csv('50_Startups.csv')
x=dataset.iloc[:,:-1].values
y=dataset.iloc[:,-1].values
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct=ColumnTransformer(transformers=[('encoder',OneHotEncoder(),[3]) ],remainder='passthrough')
x=np.array(ct.fit_transform(x)) print(x) from sklearn.model_selection import train_test_split
```

17.3

```
const express = require('express');
const path = require('path');

const app = express();
const PORT = 3000;

// Serve static files from the 'public' directory
app.use(express.static(path.join(__dirname, 'public')));

// Endpoint to trigger file download
app.get('/download', (req, res) => {
  const filePath = path.join(__dirname, 'public', 'example.txt');

  // Set headers to make the browser prompt for download
  res.setHeader('Content-Disposition', 'attachment; filename=example.txt');
  res.sendFile(filePath);
});

// Start the server
app.listen(PORT, () => {
  console.log(`Server is running on http://localhost:${PORT}`);
});
```

Slip 18

18.1

```
import java.util.Observable;
import java.util.Observer;

// WeatherData is the concrete subject that extends Observable
class WeatherData extends Observable {
    private float temperature;
    private float humidity;
    private float pressure;

    public void measurementsChanged() {
        setChanged();
        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;
    }
}

// DisplayElement is an interface implemented by concrete observers
interface DisplayElement {
    void display();
}

// CurrentConditionsDisplay is a concrete observer
class CurrentConditionsDisplay implements Observer, DisplayElement {
    private float temperature;
    private float humidity;
    private Observable observable;

    public CurrentConditionsDisplay(Observable observable) {
        this.observable = observable;
        observable.addObserver(this);
    }

    @Override
    public void update(Observable obs, Object arg) {
        if (obs instanceof WeatherData) {
            WeatherData weatherData = (WeatherData) obs;
            this.temperature = weatherData.getTemperature();
            this.humidity = weatherData.getHumidity();
            display();
        }
    }
```

```
    @Override
    public void display() {
        System.out.println("Current conditions: " + temperature + "F degrees and " + humidity + "% humidity");
    }
}

// Client code to test the Observable pattern
public class WeatherStation {
    public static void main(String[] args) {
        // Create an observable (subject)
        WeatherData weatherData = new WeatherData();

        // Create observers (displays)
        CurrentConditionsDisplay currentConditionsDisplay = new CurrentConditionsDisplay(weatherData);

        // Simulate measurements change
        weatherData.setMeasurements(80, 65, 30.4f);
    }
}
```

18.2

```
 import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
dataset=pd.read_csv('Position_Salaries.csv')
x=dataset.iloc[:,1:-1].values
y=dataset.iloc[:,-1].values
print(dataset.head(5))
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
```

```
 p_r=PolynomialFeatures(degree=4)
x_poly=p_r.fit_transform(x)
lin_reg=LinearRegression()
lin_reg.fit(x_poly,y)
LinearRegression()
y_pred=lin_reg.predict(x_poly)
df=pd.DataFrame({'Real Values':y,'Predicted Values':y_pred})
print(df)
x_grid=np.arange(min(x),max(x),0.1)
x_grid=x_grid.reshape((len(x_grid),1))
plt.scatter(x,y,color='yellow')
plt.scatter(x,y_pred,color='red')
plt.plot(x_grid,lin_reg.predict(p_r.fit_transform(x_grid)),color='black')
plt.title('Polynomial Regression')
plt.xlabel('position level')
plt.ylabel('Salary')
plt.show()
```

Slip 25

25.1

```
public class Singleton {
   // Declare a volatile instance variable to ensure visibility across threads
   private static volatile Singleton instance;

   // Private constructor to prevent instantiation from outside the class
   private Singleton() {
      // Initialization code, if needed
   }

   // Double-checked locking for thread safety
   public static Singleton getInstance() {
      if (instance == null) {
         synchronized (Singleton.class) {
            // Check again inside synchronized block to avoid race condition
            if (instance == null) {
               instance = new Singleton();
            }
         }
```

```
        }
        return instance;
    }

    // Other methods, if any
}
```

25.2

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_predict
data = pd.read_csv(r'kc_house_data.csv')
data.head(5);
print(data.shape)
f =
['price','bedrooms','bathrooms','sqft_living','floors','condition','sqft_abov
e','sqft_basement','yr_built','yr_renovated']
data = data[f]
print(data.shape)
data = data.dropna()
print(data.shape)
data.describe()
X=data[f[1:]]
y=data['price']
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size
=0.2,random_state=42)
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)

print(y_test.shape)
lr=LinearRegression()
lr.fit(X_train,y_train)
print(lr.coef_)
y_test_predict = lr.predict(X_test)
print(y_test_predict.shape)
g = plt.plot((y_test-y_test_predict),marker='o',linestyle="")
plt.show()
```

25.3

```
// Import the HTTP module
const http = require('http');

// Configure the HTTP server to respond with "Hello, World!" to all requests
const server = http.createServer((req, res) => {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello, World!\n');
});

// Listen on port 3000 and IP address 127.0.0.1
const PORT = 3000;
const IP = '127.0.0.1';
```

```
server.listen(PORT, IP, () => {
  console.log(`Server running at http://${IP}:${PORT}/`);
});
```

Slip 26

26.1

```
// Define the Duck interface
interface Duck {
    void display();

    void performFly();

    void performQuack();

    void setFlyBehavior(FlyBehavior flyBehavior);

    void setQuackBehavior(QuackBehavior quackBehavior);
}

// Define the FlyBehavior interface
interface FlyBehavior {
    void fly();
}

// Define the QuackBehavior interface
interface QuackBehavior {
    void quack();
}

// Concrete implementation of FlyBehavior for flying
class FlyWithWings implements FlyBehavior {
    @Override
    public void fly() {
        System.out.println("Flying with wings");
    }
}

// Concrete implementation of FlyBehavior for not flying
class FlyNoWay implements FlyBehavior {
    @Override
    public void fly() {
        System.out.println("I can't fly");
    }
}

// Concrete implementation of QuackBehavior for quacking
class Quack implements QuackBehavior {
    @Override
    public void quack() {
        System.out.println("Quack");
    }
}

// Concrete implementation of QuackBehavior for not quacking
```

```
class MuteQuack implements QuackBehavior {
    @Override
    public void quack() {
        System.out.println("<< Silence >>");
    }
}

// Concrete implementation of Duck
class MallardDuck implements Duck {
    private FlyBehavior flyBehavior;
    private QuackBehavior quackBehavior;

    public MallardDuck() {
        // Default behaviors
        this.flyBehavior = new FlyWithWings();
        this.quackBehavior = new Quack();
    }

    @Override
    public void display() {
        System.out.println("Displaying Mallard Duck");
    }

    @Override
    public void performFly() {
        flyBehavior.fly();
    }

    @Override
    public void performQuack() {
        quackBehavior.quack();
    }

    @Override
    public void setFlyBehavior(FlyBehavior flyBehavior) {
        this.flyBehavior = flyBehavior;
    }

    @Override
    public void setQuackBehavior(QuackBehavior quackBehavior) {
        this.quackBehavior = quackBehavior;
    }
}

public class DuckSimulator {
    public static void main(String[] args) {
        Duck mallardDuck = new MallardDuck();

        mallardDuck.display();
        mallardDuck.performFly();
        mallardDuck.performQuack();

        // Change fly behavior dynamically
        mallardDuck.setFlyBehavior(new FlyNoWay());
        mallardDuck.performFly();
    }
}
```

26.2

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
dataset=pd.read_csv('50_Startups.csv')
x=dataset.iloc[:,:-1].values
y=dataset.iloc[:,-1].values
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct=ColumnTransformer(transformers=[('encoder',OneHotEncoder(),[3])
],remainder='passthrough')
x=np.array(ct.fit_transform(x))
print(x)
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
from sklearn.linear_model import LinearRegression
regressor=LinearRegression()
regressor.fit(x_train,y_train)
LinearRegression()
y_pred=regressor.predict(x_test)
df=pd.DataFrame({'Real Values':y_test,'Predicted Values':y_pred})
print(df)
```

26.3

```
const mysql = require('mysql2');

// Create a connection to the MySQL server
const connection = mysql.createConnection({
  host: 'localhost', // Change this to your MySQL server host
  user: 'root',      // Change this to your MySQL username
  password: 'password',  // Change this to your MySQL password
});

// Connect to MySQL server
connection.connect((err) => {
  if (err) {
    console.error('Error connecting to MySQL server: ', err);
    return;
  }
  console.log('Connected to MySQL server');

  // Create a new database
  connection.query('CREATE DATABASE IF NOT EXISTS mydatabase', (createDbErr) => {
    if (createDbErr) {
      console.error('Error creating database: ', createDbErr);
      connection.end(); // Close the connection
      return;
    }
    console.log('Database created or already exists');

    // Use the newly created database
    connection.query('USE mydatabase', (useDbErr) => {
      if (useDbErr) {
        console.error('Error selecting database: ', useDbErr);
        connection.end(); // Close the connection
```

```
      return;
    }
    console.log('Using database: mydatabase');

    // Create a new table
    const createTableQuery = `
      CREATE TABLE IF NOT EXISTS users (
        id INT AUTO_INCREMENT PRIMARY KEY,
        name VARCHAR(255) NOT NULL,
        email VARCHAR(255) NOT NULL
      )
    `;

    connection.query(createTableQuery, (createTableErr) => {
      if (createTableErr) {
        console.error('Error creating table: ', createTableErr);
      } else {
        console.log('Table "users" created or already exists');
      }

      // Close the connection
      connection.end((endErr) => {
        if (endErr) {
          console.error('Error closing connection: ', endErr);
        } else {
          console.log('Connection closed');
        }
      });
    });
  });
});
```

Slip 27

27.1

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


// Concrete implementations of Shape
class Circle implements Shape {
    @Override
    public void draw() {
        System.out.println("Drawing Circle");
    }
```

```
}

class Rectangle implements Shape {
    @Override
    public void draw() {
        System.out.println("Drawing Rectangle");
    }
}

class Square implements Shape {
    @Override
    public void draw() {
        System.out.println("Drawing Square");
    }
}

// Abstract Factory interface
interface ShapeFactory {
    Shape createShape();
}

// Concrete implementations of ShapeFactory
class CircleFactory implements ShapeFactory {
    @Override
    public Shape createShape() {
        return new Circle();
    }
}

class RectangleFactory implements ShapeFactory {
    @Override
```

```
    public Shape createShape() {
        return new Rectangle();
    }
}

class SquareFactory implements ShapeFactory {
    @Override
    public Shape createShape() {
        return new Square();
    }
}

// Client code using Abstract Factory
public class AbstractFactoryPatternExample {
    public static void main(String[] args) {
        // Creating a Circle using CircleFactory
        ShapeFactory circleFactory = new CircleFactory();
        Shape circle = circleFactory.createShape();
        circle.draw();

        // Creating a Rectangle using RectangleFactory
        ShapeFactory rectangleFactory = new RectangleFactory();
        Shape rectangle = rectangleFactory.createShape();
        rectangle.draw();

        // Creating a Square using SquareFactory
        ShapeFactory squareFactory = new SquareFactory();
        Shape square = squareFactory.createShape();
        square.draw();
    }
}
```

27.2

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
dataset=pd.read_csv('Position_Salaries.csv')
x=dataset.iloc[:,1:-1].values
y=dataset.iloc[:,-1].values
print(dataset.head(5))
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
 p_r=PolynomialFeatures(degree=4)
x_poly=p_r.fit_transform(x)
lin_reg=LinearRegression()
lin_reg.fit(x_poly,y)
LinearRegression()
y_pred=lin_reg.predict(x_poly)
df=pd.DataFrame({'Real Values':y,'Predicted Values':y_pred})
print(df)
x_grid=np.arange(min(x),max(x),0.1)
x_grid=x_grid.reshape((len(x_grid),1))
plt.scatter(x,y,color='yellow')
plt.scatter(x,y_pred,color='red')
plt.plot(x_grid,lin_reg.predict(p_r.fit_transform(x_grid)),color='black')
plt.title('Polynomial Regression')
plt.xlabel('position level')
plt.ylabel('Salary')
plt.show()
```

27.3

```
cd myproject
python manage.py startapp myapp
```

python code

```
# myapp/views.py
from django.shortcuts import render
from django.http import HttpResponse

def index(request):
    return HttpResponse("Hello! I am learning Django")
# myapp/urls.py
from django.urls import path
from .views import index

urlpatterns = [
    path('', index, name='index'),
]
# myproject/urls.py
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('myapp.urls')),
]
```

python manage.py runserver