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}`);

});