NAME: Akshay Kumar Singh

REGD. NO.: 19410121102

Exercise-2

1. We will implement different file handlers for different types of files such as text, image and xml files. Which design pattern will be preferred for this problem. Provide suitable code snippets for this.

Ans.

var Node = function (name) {

this.children = [];

this.name = name;

}

Node.prototype = {

add: function (child) {

this.children.push(child);

},

remove: function (child) {

var length = this.children.length;

for (var i = 0; i < length; i++) {

if (this.children[i] === child) {

this.children.splice(i, 1);

return;

}

}

},

getChild: function (i) {

return this.children[i];

},

hasChildren: function () {

return this.children.length > 0;

}

}

function traverse(indent, node) {

console.log(Array(indent++).join("--") + node.name);

for (var i = 0, len = node.children.length; i < len; i++) {

traverse(indent, node.getChild(i));

}

}

function run() {

var tree = new Node("root");

var left = new Node("left")

var right = new Node("right");

var leftleft = new Node("leftleft");

var leftright = new Node("leftright");

var rightleft = new Node("rightleft");

var rightright = new Node("rightright");

tree.add(left);

tree.add(right);

tree.remove(right); // note: remove

tree.add(right);

left.add(leftleft);

left.add(leftright);

right.add(rightleft);

right.add(rightright);

traverse(1, tree);

}

2. One organisation has one department as HR department and two child departments as Humanity Department and Logistic Department under Hr department. We have to calculate tax as HRA is different for different departments but it should implement the main TaxCalulator interface. Which design pattern will be preferred for this problem. Provide suitable code snippets for this.

Ans.

Behavioral Pattern will be preferred for this problem.

public interface TaxCalculator {

public abstract void execute();

}

public class Humanity implements TaxCalculator {

private int basic\_salary;

public Order(int basic\_salary) {

this.basic\_salary = basic\_salary;

}

@Override

public void execute() {

HRA=(10/100)\*basicsalary;

}

}

public class Logistic implements TaxCalculator {

private int basic\_salary;

public Order(int basic\_salary) {

this.basic\_salary = basic\_salary;

}

@Override

public void execute() {

HRA=(10/100)\*basicsalary;

}

}

public class Department {

public static void main(String[] args) {

basic\_salary basic\_salary = new basic\_salary();

Humanity humanity = new Humanity(basic\_salary);

Logistic logistic = new Logistic(basic\_salary);

Humanity.execute();

humanity = new humanity(basic\_salary);

logistic = new Logistic(basic\_salary);

Logistic.execute();

}

}

3. Write a javascript function to find average of all numbers and variance of those numbers ? Write Async/await function for both of these calculations.

Ans.

const arr = [1,2,3,4,5,6,7,8,9,10];

const findVariance = (arr = []) => {

if(!arr.length){

return 0;

};

const sum = arr.reduce((acc, val) => acc + val);

const { length: num } = arr;

const median = sum / num;

let variance = 0;

arr.forEach(num => {

variance += ((num - median) \* (num - median));

});

variance /= num;

return variance;

};

console.log(findVariance(arr))

4. Create a class as Product in Javascript which will have productId, ProductName and Productprice fields in that class. Create a few instances and store them in JSON format. Now access those data and print to console  using Promise object.

Ans.

class productId

{

constructor( productId, ProductName,Productprice)

{

this.productId=productId;

this.ProductName=ProductName;

this.Productprice=Productprice;

}

}

let ob1=new productId(1234,abcd,4324);

let ob2=new productId(11,abc,654);