Assignment1:

<html>

<head>

<title>

OOJS + DOM

</title>

</head>

<body>

<div id="root">

FirstName: <input type="text" id="firstName" /> <br>

LastName: <input type="text" id="lastName" /> <br>

Number Of Working days: <input type="text" class="nowd" /> <br>

Salary: <input type="text" id="salary" /> <br>

Net Salary: <input type="text" class="salary" /> <br>

</div>

<script>

// create a class => Person (firstName, lastName, age)

class Person

{

constructor(firstName,lastName)

{

this.firstName=firstName;

this.lastName=lastName;

}

}

// create a class Employee <= Person (numberOfWorkingDays, salary)

class Employee extends Person

{

constructor(firstName,lastName,numberOfWorkingDays,salary)

{

super(firstName,lastName);

this.numberOfWorkingDays=numberOfWorkingDays;

this.salary=salary;

}

getSalary()

{

return (this.salary/30)\*this.numberOfWorkingDays;

l

}

}

// Add a function to Employee class => getSalary() => (salary / 30) \* numberOfWorkingDays

// create obj <= Employee (Jane, Doe, 25, 25000);

const obj=new Employee("Shiva","rudra",25,25000);

// Assign the properties ir. firstName, lastName, numberOfWorkingDays, salary

// input => docuement,getElementById('firstName').value = '';

document.getElementById('firstName').value=obj.firstName;

document.getElementById('lirstName').value=obj.lastName;

document.getElementsByClassName('nowd')[0].value=obj.numberOfWorkingDays;

document.getElementById('salary').value=obj.numberOfWorkingDays;

document.getElementsByClassName('salary')[0].value=obj.getSalary();

</script>

</body>

</html>

Assignment2:

// Create class Sale with properties => name, amount

class Sale {

constructor(name, amount) {

this.name = name;

this.amount = amount;

}

}

// Create class Purchase with properties => name, amount

class Purchase {

constructor(name, amount) {

this.name = name;

this.amount = amount;

}

}

// Create a Class Company with properties => name, sales(array of Sale), purchases(array of Purchase)

class Company {

#sales;

#purchases;

constructor(name, sales = [], purchases = []) {

this.name = name;

this.#sales = sales;

this.#purchases = purchases;

}

get sales() {

return this.#sales;

}

set sales(sale) {

this.#sales.push(sale);

}

get purchases() {

return this.#purchases;

}

set purchases(purchase) {

this.#purchases.push(purchase);

}

// Add a function addSale => will create new objects of Sale and push it to the sales arrays

addSale(name, amount) {

const newSale = new Sale(name, amount);

this.#sales.push(newSale);

console.log(`${newSale.name} - ${newSale.amount}`);

}

// Add a function addPurchase => will create new objects of Purchase and push it to the purchaese arrays

addPurchase(name, amount) {

const newPurchase = new Purchase(name, amount);

this.#purchases.push(newPurchase);

console.log(`${newPurchase.name} - ${newPurchase.amount}`);

}

/\*\*

\* Add a function getProfitLossReport

\* 1. Calculate Profit using sales

\* 2. Calculate spends using purchases

\* 3. Calculate difference between profit and spends

\* 4. return an object in following format

\* {

\* income: sum of sales,

\* expense: sum of purchases,

\* profit: income - expense, (if income > expense)

\* loss: expense - income, (if income < expense) \* } \* \*/

getProfitLossReport() {

let income = this.#sales.reduce(

(totalIncome, next) => (

totalIncome += next.amount

)

, 0);

let expense = this.#purchases.reduce(

(totalExpense, next) => (

totalExpense += next.amount

)

, 0);

let profit = "N/A";

let loss = "N/A";

if (income > expense) {

profit = income - expense;

} else {

loss = expense - income;

}

return {

Income: income,

Expense: expense,

Profit: profit,

Loss: loss

}

}

}

// Create a Class MNC by inheriting Company with extra

//properties=> taxRate, country

class MNC extends Company {

constructor(name, sales = [], purchases = [], taxRate, country) {

super(name, (sales = []), (purchases = []));

this.taxRate = taxRate;

this.country = country;

}

// Add a function calculateTax

// If profit is more than 5,00,000 tax will be applied to amount (profit - 500000)

// ex. if profit = 600000 & taxRate = 10

// taxable amount will be 600000 - 500000 = 100000

// tax = 100000 \* 10/100 = 10000

// Create an object of MNC and calculate tax with differnet input combination

// add Sales & purchases to make it realtime

calculateTax() {

let profitReport = super.getProfitLossReport();

let { Profit: Profit } = profitReport;

if (Profit > 500000) {

let taxable\_amt = Profit - 500000;

let tax = taxable\_amt \* this.taxRate / 100;

return tax;

}

else {

return "Tax not Applicable";

}

}

}

const vivo = new MNC("VIVO ", [], [], 10, "chaina");

console.log(

`Company Name - ${ vivo.name} Country - ${ vivo.country}`

);

console.log("----- Purchases -----");

vivo.addPurchase("LED", 15000);

vivo.addPurchase("Lenses", 250000);

vivo.addPurchase("Camera", 350000);

vivo.purchases;

console.log("");

console.log("");

console.log("");

console.log("----- Sales -----");

vivo.addSale("vivo Watch", 100000);

vivo.addSale("vivo TV", 200000);

vivo.addSale("vivo Mobiles", 300000);

vivo.sales;

console.log("");

console.log("");

console.log("-----Profit & Loss Report -----");

console.log(vivo.getProfitLossReport());

console.log("");

console.log("-----Tax -----");

console.log(vivo.calculateTax());