React-Js Assignment:

1. OOJS+DOM

```
<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{
            Person(firstName,lastName,age){
                This.firstName=firstName;
                This.lastName=lastName;
                This.age=age;
            }

        }

        // create a class Employee <= Person (numberOfWorkingDays, salary)
        Class Employee extends Person{
            Constructor(numberOfWorkingDays,salary){
                Super(firstName);
                This.numberOfWorkingDays=numberOfWorkingDays;
                This.salary=salary
            }

        }

        // Add a function to Employee class => getSalary() => (salary / 30) *
numberOfWorkingDays
        Let x=getSalary(salary,numberOfWorkingDays);
        Function getSalary(salary,numberOfWorkingDays){
            Return (salary / 30) * numberOfWorkingDays;

        }
        // create obj <= Employee (Jane, Doe, 25, 25000);
        Const Employee = new Employee(“Jane”,”Doe”,25,25000);{
            Constructor(firstName,lastName,age,salary)
        {
            This.firstName=”Jane”;
            This.lastName=”doe”;
            This.age=”25”;
            This.salary=”25000”

        }
    }
        // Assign the properties ir. firstName, lastName, numberOfWorkingDays,
salary & output of getSalary(salary class)
        Const firstName=document.getElementById(“firstName”);
        Const lastName=document.getElementById(“lastName”);
        Const
numberOfWorkingDays=document.getElementsByClassName(“numberOfWorkingDays”);
        Const salary=document.getElementById(“salary”);
        Const netSalary=document.getElementsByClassName(“salary”)[0];
        // input => documents,getElementById(‘firstName’).value = ‘’;
        firstName.value=”Jane”;
        lastName.value=”Doe”;
        numberOfWorkingDays.value=”25”;
        salary.value=”25000”;
        console.log(obj.getSalary());
        netSalary.value=obj.getSalary().toString();

        </script>
    </body>
</html>
```

2. Company with sales and purchases

```
!DOCTYPE html>
<html lang="en">
  <head>
    <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>Assignment 2</title>
  </head>
  <body>
    <div id="root"></div>
    <script>
      // Create class Sale with properties => name, amount
      class Sale {
        constructor(name, amount) {
```

```javascript
          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 will be private members
        #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.amount);
      console.log(`${newSale.name} - ${newSale.amount}`);
      //   sales = 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.amount);
      console.log(`${newPurchase.name} - ${newPurchase.amount}`);
      //   purchases = 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)
      );
      console.log(`Income - ${income}`);
      let expense = this.#purchases.reduce(
        (totalExpense, next) => (totalExpense += next)
      );
            console.log(`expense - ${expense}`);
            let profit, loss;
            let isProfit = false;
            if (income > expense) {
              profit = income - expense;
              isProfit = true;
              console.log(`Profit - ${isProfit}    Profit - ${profit}`);
              return [profit, isProfit];
            } else {
```

```
      loss = expense - income;
      isProfit = false;
      console.log(`Profit - ${isProfit}    loss - ${loss}`);
      return [loss, isProfit];
    }
  }
}

// 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;
  }

  calculateTax() {
    let taxAmount, profit;
    [profit, isProfit] = getProfitLossReport();

    if (profit > 4_00_000 && isProfit) {
      taxAmount = profit - 4_00_000;
    }

    return taxAmount * (this.taxRate / 100);
  }
}

// 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


const Himalaya= new MNC("Himalaya", [], [], 10, "India");
console.log(
  `Company Name - ${Himalaya.name}      Country - ${Himalaya.country}`
```

```
          );
          console.log("Purchases");
          Himalaya.addPurchase("Himalaya Soap", 95);
          Himalaya.addPurchase("Alovera gel", 120);
          Himalaya.addPurchase("ToothPaste", 75);
          Himalaya.purchases;
          console.log("");
          console.log("");
        console.log("Sales");
          Himalaya.addSale("Soap", 95);
          Himalaya.addSale("Alovera gel", 250);
          Himalaya.addSale("Toothpaste", 150);
          Himalaya.sales;
          console.log("");
          console.log("");
          console.log("");

          console.log("Profit/Loss Report");

          dabur.getProfitLossReport();

          console.log("");
          console.log("");
          console.log("");
          console.log("Tax");
          console.log(`Tax = ${Himalaya.calculateTax()}`);
        </script>
      </body>
    </html>
```