EXERCISE-2

Q1

```
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;
}
```

```
}
// recursively traverse a (sub)tree
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);}
```

Q2

```
public interface TaxCalculator {
public abstract void hra();
}
public class Humanity implements TaxCalculator {
private int basic_salary;
public Order(int basic_salary) {
this.basic_salary = basic_salary;
}
@Override
public void hra() {
HRA=(10/100)*basic_salary;
}
}
public class Logistic implements TaxCalculator {
private int basic_salary;
public Order(int basic_salary) {
this.basic_salary = basic_salary;
}
@Override
public void hra() {
HRA=(10/100)*basic_salary;
}
}
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.hra();
humanity = new humanity(basic_salary);
logistic = new Logistic(basic_salary);
Logistic.hra();}}
```

Q3

```
const arr = [1,2,3,4,5,6,7,8,9,10];
const sum = arr.reduce((acc, val) => acc + val);
var mean=sum/arr.length;
console.log("mean :",mean);
const { length: num } = arr;
let variance = 0;
arr.forEach(arr => {
variance += ((arr - mean) * (arr - mean));
});
console.log("variance ",variance/arr.length);
```

Q4

```
class productId
{
constructor( productId, ProductName,Productprice)
{
this.productId=productId;
this.ProductName=ProductName;
this.Productprice=Productprice;
}
```

```
}
let ob1=new productId(1,abc,10);
let ob2=new productId(22,def,100);
```