SUBHRA JYOTI NANDI

19410121105

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

```
class file_handler{
    constructor(name,ext)
    {
        this.name = name;
this.ext = ext;
    }
    check_ext()
    {
        if(this.ext === 'txt')
return 'This is text file';
        else if(this.ext === 'img' || this.ext === 'png')
return 'This is an image file';        else if(this.ext
=== 'xml')            return 'This is a xml file';
else
            return 'Invalid file format';
    }

}

let file_name = 'temp.img';
let file = file_name.split('.');
if(file.length !== 2)
    console.log("Invalid file name"); else
{
    let obj1 = new file_handler(file[0],file[1]);
    console.log(obj1.check_ext());
}
```


1. One organization have one depratment as HR department and two child department as Humanity Department and Logistic Department under Hr department. We have to calulate tax as HRA is different for different departments but it should

implement main TaxCalulator interface. Which design pattern will be preferred for this problem. Provide suitable code snippet for this.

Ans: Behavioral Pattern will be preferred for this problem.

```java
public interface TaxCalculator {
public abstract void execute();
}
public class Humanity implements TaxCalculator {
 private int basic_salary;
 private int hra_all;

 public Order(int basic_salary, int hra_all)
{     this.basic_salary = basic_salary; this.hra_all
     = hra_all;
}

 @Override
 public void execute() {
HRA=(hra_all/100)*basicsalary;

 }
}
public class Logistic implements TaxCalculator {
private int basic_salary;
private int hra_all;

 public Order(int basic_salary, int hra_all) {
  this.basic_salary = basic_salary;
      this.hra_all = hra_all;
 }

 @Override
 public void execute() {
HRA=(hra_all/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 calculations

```javascript
function findAverage(arr)

{

    let sum=0;
    for(let i=0;i<arr.length;i++)

    {

        sum+=arr[i];

    }

    return sum/arr.length;

}
async function findVariance(arr)

{

    let mean = await findAverage(arr);

    // console.log(mean);

let sum=0;

    for(let i=0;i<arr.length;i++)

    {

        sum+=Math.pow((arr[i]-mean),2)
    }

    return sum/arr.length;
```

```
}

let arr = [1,2,3,4,5];

console.log(findAverage(arr)); //

console.log(findVariance(arr));

findVariance(arr).then(Response => console.log(Response)).catch(error =>
console.error())
```

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

```
class product {

    constructor(productId,
productName, productPrice)
    {
        this.productId = productId;
this.productName = productName;
        this.productPrice =
productPrice;
    }

}

let product_1 = new product(9999,
pro_1, 2200);
console.log(product_1); let
product_2 = new product(7777,
pro_2, 4400);
console.log(product_2);
```

5. Create ReactJs/Angular web project on local system for below mentioned usability.
(For ReactJS Group)

Design a login page with username and password as textfields. There will be a submit button and cancel button in that page. Now create a dummy data for valid username and password in the corresponding Javascript/Typescript file. Use onclick event in (ReactJs) to validate userbname and password and direct to another page(home.html)

```
import { useState } from 'react';

const validator = [{name: 'Helox',password: '09'}]
export default function Form() {

const [name, setName] = useState(''); const
[email, setEmail] = useState('');
const [password, setPassword] = useState('');

const [submitted, setSubmitted] = useState(false);
const [error, setError] = useState(false);

const handleName = (e) => {
setName(e.target.value);
    setSubmitted(false);
};

const handleEmail = (e) => {
setEmail(e.target.value);
setSubmitted(false);
};

const handlePassword = (e) => {
setPassword(e.target.value);
setSubmitted(false);
};

const handleSubmit = (e) => {
e.preventDefault();    let check =
false;    for(let
i=0;i<validator.length;i++)
if(validator[i].name === name &&
validator[i].password === password)
    {
        setSubmitted(true);
check = true;
setError(false);
break;
    }
    // console.log(check, name, password);
if(!check)
```

```
      {
setError(true);
setSubmitted(false);
      }
};

const successMessage = () => {
   return (    <div
className="success"
      style={{
      display: submitted ? '' : 'none',
      }}>
      <h1>User {name} successfully logged
in!!</h1>    </div>
   );
};

// Showing error message if error is true
const errorMessage = () => {
   return (    <div
className="error"
style={{
      display: error ? '' : 'none',
      }}>
      <h1>Incorrect Username or password</h1>
   </div>
   );
};

return (
   <div className="form">
   <div>
      <h1>User Login</h1>
   </div>

   {/* Calling to the methods */}
   <div className="messages">
     {errorMessage()}
     {successMessage()}
   </div>

   <form>
     {/* Labels and inputs for form data */}
     <label className="label">UserName</label>
<input onChange={handleName}
className="input"
     value={name} type="text" />

     <label className="label">Password</label>
<input onChange={handlePassword}
className="input"
```

```jsx
          value={password} type="password" />

          <button onClick={handleSubmit}
className="btn" type="submit">
          Submit
          </button>
      </form>
      </div>
);
}
```