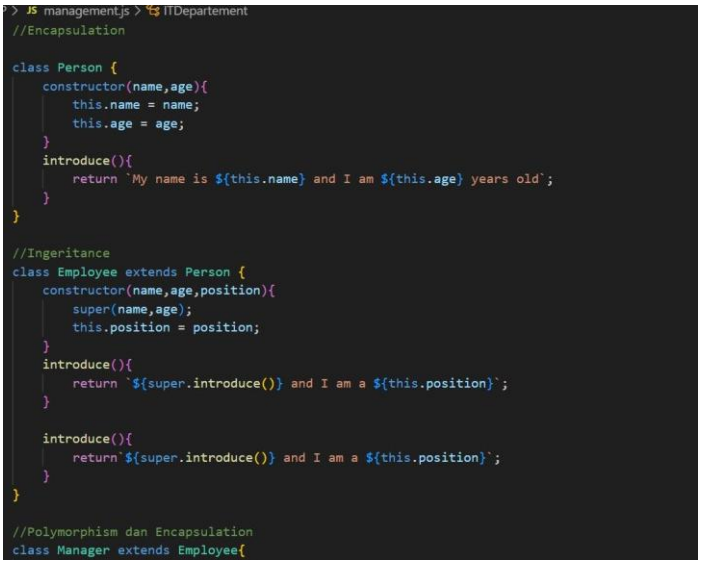
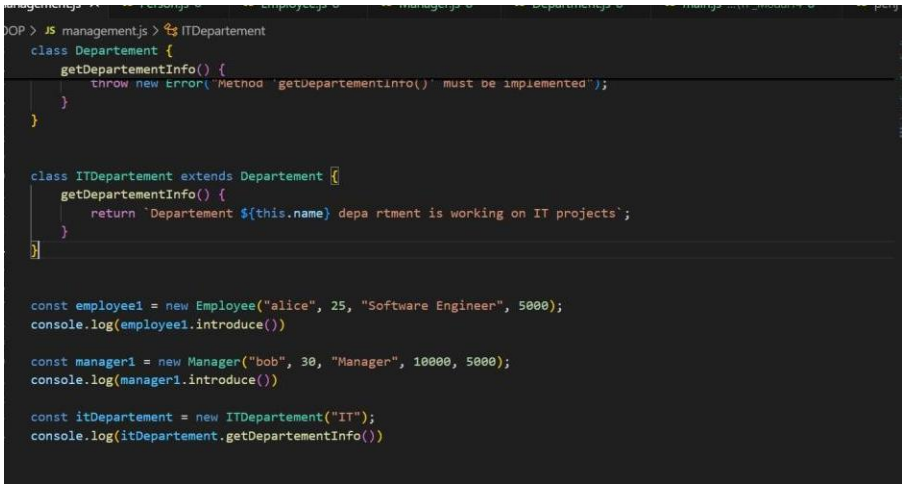
TP MODUL 14

Modul 2 – Code yang belum rapih



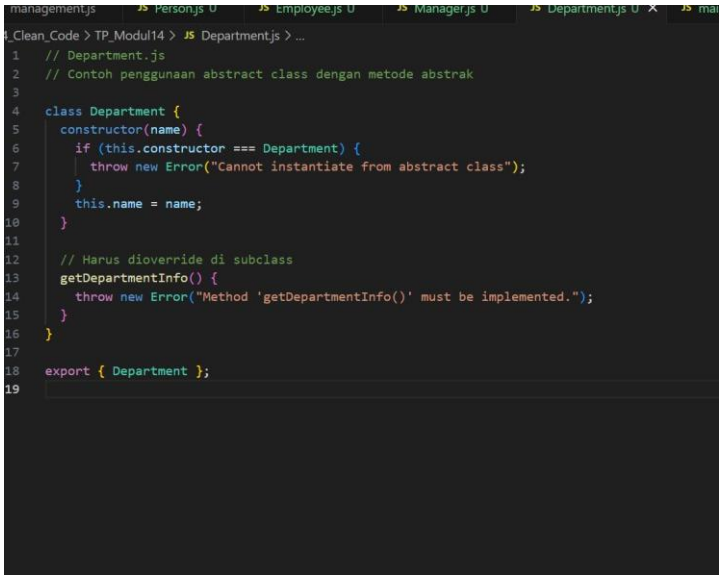
```
> JS management.js > ITDepartement
//Encapsulation

class Person {
    constructor(name,age){
        this.name = name;
        this.age = age;
    }
    introduce(){
        return `My name is ${this.name} and I am ${this.age} years old`;
    }
}

//Ingeritance
class Employee extends Person {
    constructor(name,age,position){
        super(name,age);
        this.position = position;
    }
    introduce(){
        return `${super.introduce()} and I am a ${this.position}`;
    }

    introduce(){
        return`${super.introduce()} and I am a ${this.position}`;
    }
}

//Polymorphism dan Encapsulation
class Manager extends Employee{
```

```
//Polymorphism dan Encapsulation
class Manager extends Employee{
    constructor(name, age, jobTitle, salary, bonus){
        super(name, age, jobTitle, salary);
        this.bonus = bonus;
    }

    getTotalSalary(){
        return this.salary + this.bonus;
    }

    //Polymorphism
    introduce() {
        return `${super.introduce()} I am a ${this.jobTitle}. My total salary is ${this.getTotalSalary()}.`;
    }
}

class Departement {
    constructor(name) {
        if(this.constructor === Departement) {
            throw new Error("Cannot instantiate from abstract class");
        }

        this.name = name;
    }

    getDepartementInfo() {
```

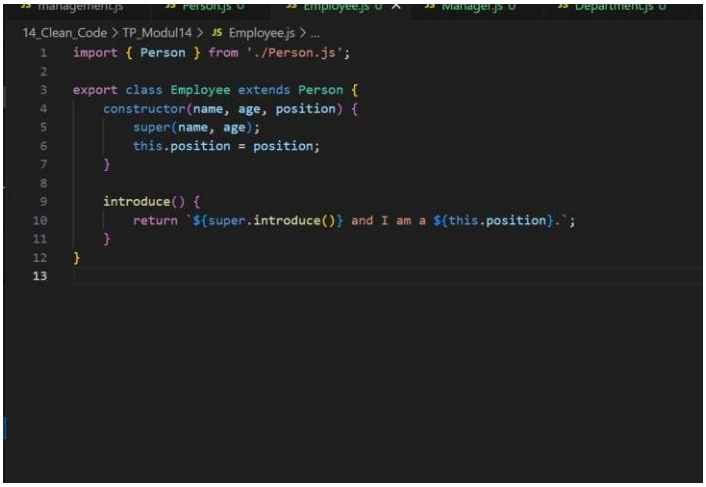


```
OOP > JS management.js > ITDepartement
class Departement {
    getDepartementInfo() {
        throw new Error("Method 'getDepartementInfo()' must be implemented");
    }
}


class ITDepartement extends Departement {
    getDepartementInfo() {
        return `Departement ${this.name} depa rtment is working on IT projects`;
    }
}


const employee1 = new Employee("alice", 25, "Software Engineer", 5000);
console.log(employee1.introduce())

const manager1 = new Manager("bob", 30, "Manager", 10000, 5000);
console.log(manager1.introduce())

const itDepartement = new ITDepartement("IT");
console.log(itDepartement.getDepartementInfo())
```

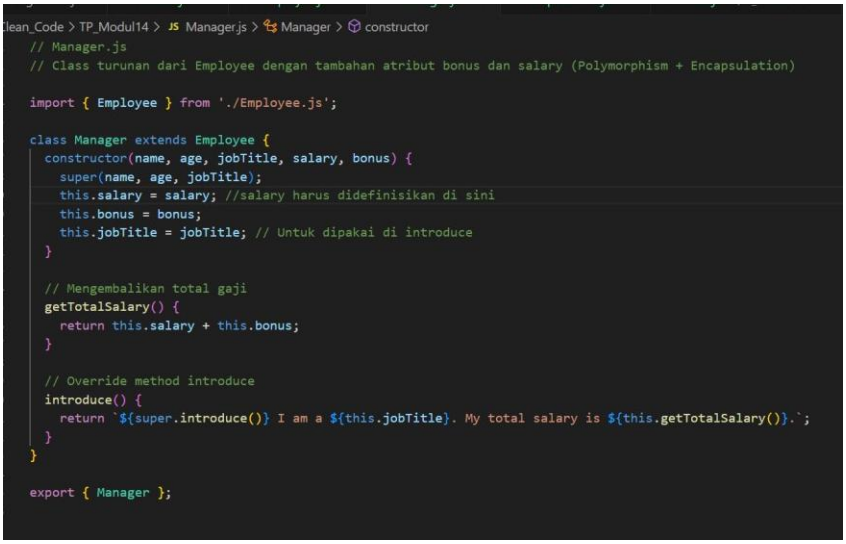
Code yang sudah di rapihkan



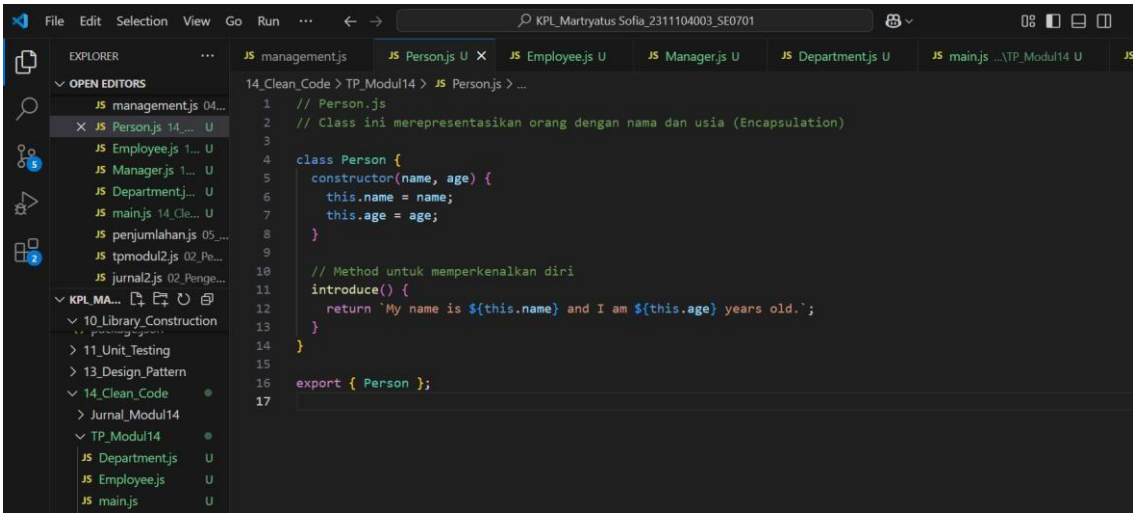
```
management.js  Person.js U  Employee.js U  Manager.js U  Department.js U  ma
4_Clean_Code > TP_Modul14 > JS Department.js > ...
1   // Department.js
2   // Contoh penggunaan abstract class dengan metode abstrak
3
4   class Department {
5     constructor(name) {
6       if (this.constructor === Department) {
7         throw new Error("Cannot instantiate from abstract class");
8       }
9       this.name = name;
10    }
11
12    // Harus dioverride di subclass
13    getDepartmentInfo() {
14      throw new Error("Method 'getDepartmentInfo()' must be implemented.");
15    }
16  }
17
18  export { Department };
19
```



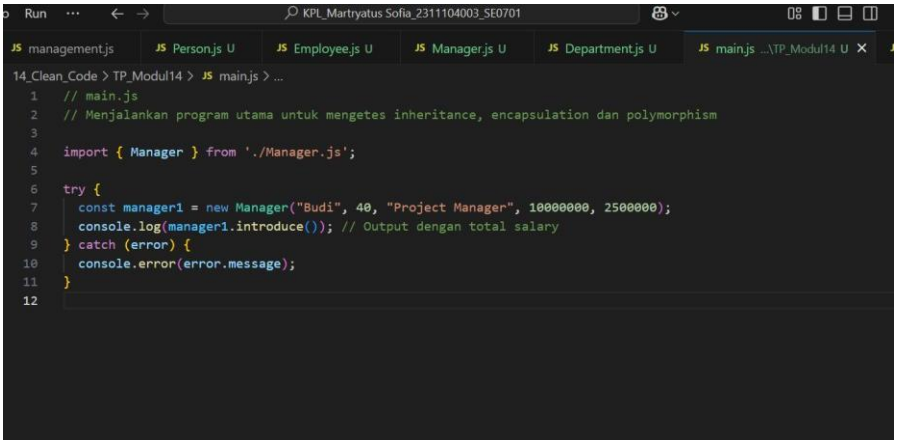
```
14_Clean_Code > TP_Modul14 > JS Employee.js > ...
import { Person } from './Person.js';

export class Employee extends Person {
    constructor(name, age, position) {
        super(name, age);
        this.position = position;
    }

    introduce() {
        return `${super.introduce()} and I am a ${this.position}.`;
    }
}
```



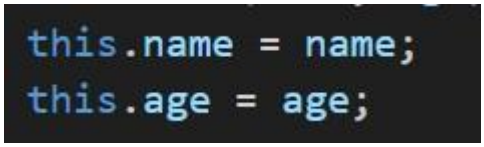
```
lean_Code > TP_Modul14 > JS Manager.js > Manager > constructor
// Manager.js
// Class turunan dari Employee dengan tambahan atribut bonus dan salary (Polymorphism + Encapsulation)

import { Employee } from './Employee.js';

class Manager extends Employee {
  constructor(name, age, jobTitle, salary, bonus) {
    super(name, age, jobTitle);
    this.salary = salary; //salary harus didefinisikan di sini
    this.bonus = bonus;
    this.jobTitle = jobTitle; // Untuk dipakai di introduce
  }

  // Mengembalikan total gaji
  getTotalSalary() {
    return this.salary + this.bonus;
  }

  // Override method introduce
  introduce() {
    return `${super.introduce()} I am a ${this.jobTitle}. My total salary is ${this.getTotalSalary()}.`;
  }
}

export { Manager };
```

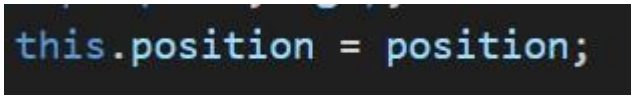


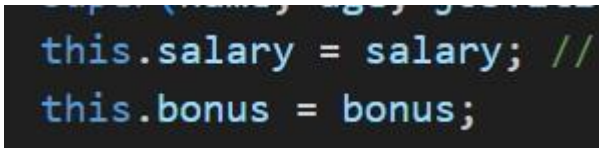
```
File  Edit  Selection  View  Go  Run  ···   KPL_Martryatus Sofia_2311104003_SE0701

EXPLORER
OPEN EDITORS
  management.js 04...
  Person.js 14_... U
  Employee.js 1... U
  Manager.js 1... U
  Department.j... U
  main.js 14_Cle... U
  penjumlahan.js 05_...
  tpmodul2.js 02_Pe...
  jurnal2.js 02_Penge...
KPL_MA...
  10_Library_Construction
  11_Unit_Testing
  13_Design_Pattern
  14_Clean_Code
    Jurnal_Modul14
    TP_Modul14
      Department.js U
      Employee.js U
      main.js U

management.js | Person.js U | Employee.js U | Manager.js U | Department.js U | main.js ...\TP_Modul14 U
14_Clean_Code > TP_Modul14 > JS Person.js > ...
1  // Person.js
2  // Class ini merepresentasikan orang dengan nama dan usia (Encapsulation)
3
4  class Person {
5    constructor(name, age) {
6      this.name = name;
7      this.age = age;
8    }
9
10   // Method untuk memperkenalkan diri
11   introduce() {
12     return `My name is ${this.name} and I am ${this.age} years old.`;
13   }
14 }
15
16 export { Person };
17
```



1. Naming convention 1) Variable / Property / Attribute



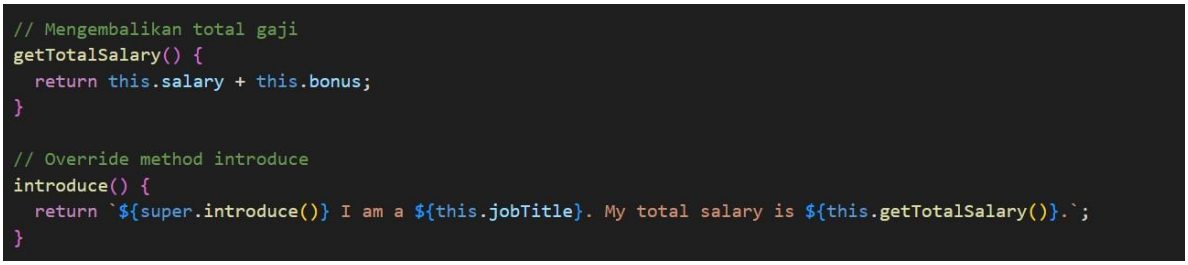




Sudah menggunakan **camelCase**.

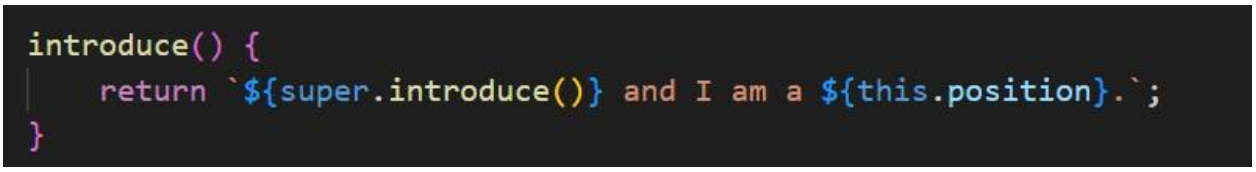
Nama-nama variabel sudah deskriptif dan sesuai dengan fungsinya.

2) Method / Function / Procedure



Nama method sudah sesuai konvensi camelCase dan deskriptif.

2. White space dan indentation



secara umum indentasi (tab/spasi) sudah rapi

3. Variable / attribute declarations

```
class Manager extends Employee {
  constructor(name, age, jobTitle, salary, bonus) {
    super(name, age, jobTitle);
    this.salary = salary; //salary harus didefinisikan di sini
    this.bonus = bonus;
    this.jobTitle = jobTitle; // Untuk dipakai di introduce
  }
```

4. Comments

```
// Manager.js
// Class turunan dari Employee dengan tambahan atribut bonus dan salary (Polymorphism + Encapsulation)
```

Hasil Running

```
Hallo, nama saya Ahmad Uffi, saya berusia 40 tahun, bekerja
sebagai Project Manager dengan total gaji Rp12500000.
```