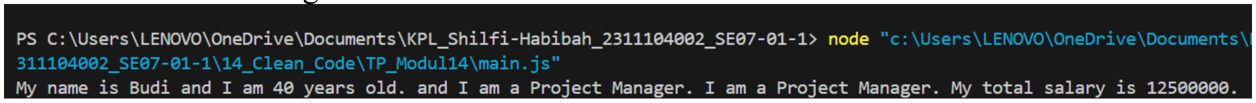
TP14_2311104002_Shilfi Habibah_SE0701

I. Link github
https://github.com/shilfihabibah/KPL_Shilfi-Habibah_2311104002_SE07-01/tree/master/14_Clean_Code

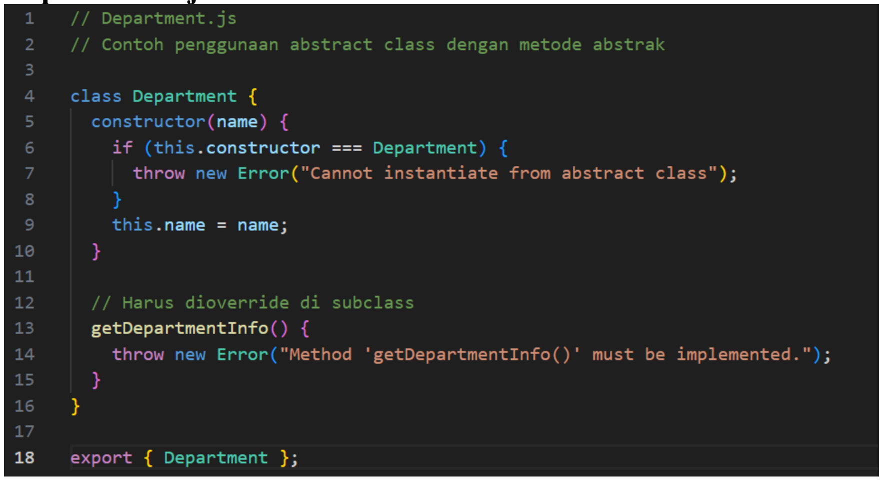
II. Screenshot hasil running



```
PS C:\Users\LENOVO\OneDrive\Documents\KPL_Shilfi-Habibah_2311104002_SE07-01-1> node "c:\Users\LENOVO\OneDrive\Documents\
311104002_SE07-01-1\14_Clean_Code\TP_Modul14\main.js"
My name is Budi and I am 40 years old. and I am a Project Manager. I am a Project Manager. My total salary is 12500000.
```

III. Codingan

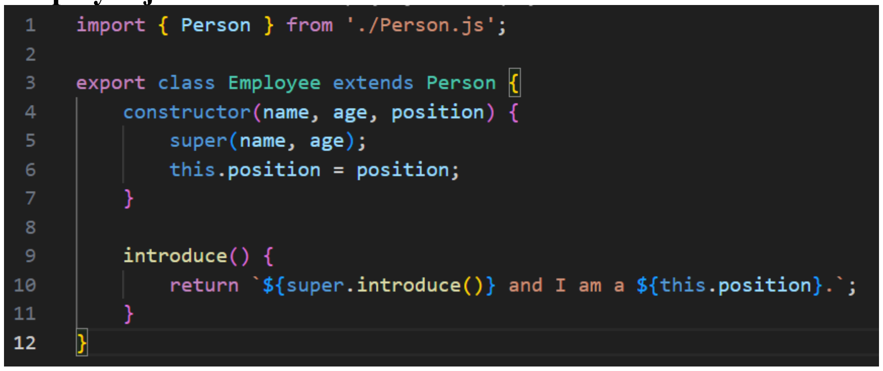
Departement.js



```javascript
// Department.js
// Contoh penggunaan abstract class dengan metode abstrak

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

  // Harus dioverride di subclass
  getDepartmentInfo() {
    throw new Error("Method 'getDepartmentInfo()' must be implemented.");
  }
}

export { Department };
```

Class abstrak yang tidak bisa langsung diinstansiasi, berisi method abstrak getDepartmentInfo().

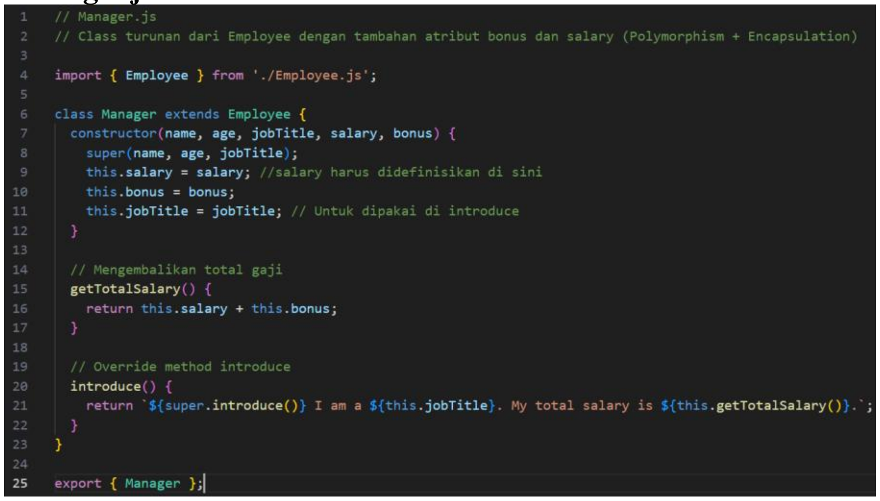
Employee.js



```javascript
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}.`;
    }
}
```

Mewarisi Person, menambah atribut position, dan override introduce().

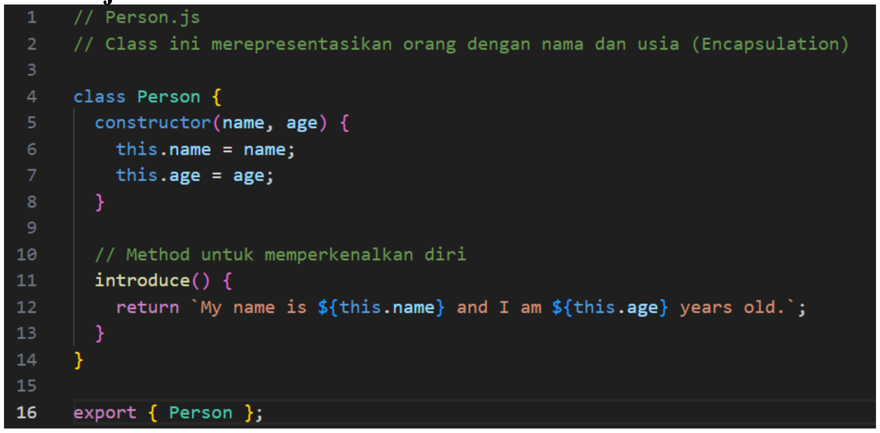
Manager.js



```javascript
// 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 };
```

Mewarisi Employee, menambah salary dan bonus, serta override introduce() untuk menampilkan total gaji.

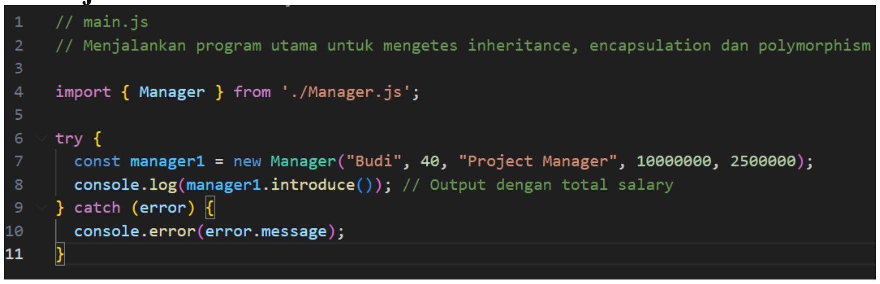
Person.js



```javascript
// Person.js
// Class ini merepresentasikan orang dengan nama dan usia (Encapsulation)

class Person {
  constructor(name, age) {
    this.name = name;
    this.age = age;
  }

  // Method untuk memperkenalkan diri
  introduce() {
    return `My name is ${this.name} and I am ${this.age} years old.`;
  }
}

export { Person };
```

Class dasar dengan atribut name dan age, serta method introduce().

Main.js



```javascript
// main.js
// Menjalankan program utama untuk mengetes inheritance, encapsulation dan polymorphism

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

try {
  const manager1 = new Manager("Budi", 40, "Project Manager", 10000000, 2500000);
  console.log(manager1.introduce()); // Output dengan total salary
} catch (error) {
  console.error(error.message);
}
```

Membuat objek Manager, lalu mencetak perkenalan lengkap beserta total gajinya.

Konsep OOP:
Encapsulation: salary, bonus, getTotalSalary()
Inheritance: Manager → Employee → Person
Polymorphism: Method introduce() dioverride
Abstraction: Class Department sebagai abstrak