

## Form Guided 03

### Pewarisan Access Modifier dan Pewarisan Hierarkis

#### Tujuan:

- Mahasiswa memahami konsep pewarisan dalam pemrograman berorientasi objek (OOP).
- Memahami access modifiers (modifikasi akses) dalam konteks pewarisan
- Memahami pewarisan hierarkis dalam membentuk hirarki kelas

Nama:

NPM:

#### 1.1 Petunjuk Pewarisan dan Hubungan "Parent-Child" (+10).

Buatlah kelas Produk sebagai kelas induk. Produk memiliki atribut **nama** dan **jumlah**. Kelas produk memiliki 2 fungsi, yaitu **informasi()** dan **getterNama()**.

```
// Kelas Induk (Parent Class)
class Produk {
    // Konstruktor untuk membuat objek dari kelas Induk.
    constructor(nama, jumlah) {
        this.nama = nama;
        this.jumlah = jumlah;
    }
    informasi() {
        console.log(`Produk ${this.nama} ada sebanyak ${this.jumlah} pcs.`);
    } // akan menampilkan nama dan jumlah dari Produk ke console.

    getterNama(){
        return this.nama;
    } // akan mengembalikan nilai nama
}
```

Buatlah kelas Makanan (kelas anak) yang merupakan turunan kelas Produk. Kelas Makanan memiliki atribut tambahan yaitu, **expired** dan fungsi **waktuRusak()**.

**Note :** Makanan membentuk sebuah rantai hierarkis dari kelas Produk.

```
// Kelas Anak (Child Class)
// jangan lupa memakai extends untuk pewarisan kelas
class Makanan extends Produk {
  constructor(expired, nama, jumlah) {
    super(nama, jumlah); // Memanggil konstruktor kelas induk (Jangan sampai terbalik)
    this.expired = expired;
  }
  waktuRusak() {
    console.log(`${this.nama} akan rusak pada tanggal ${this.expired}`);
  } // akan menampilkan waktu rusak Makanan
}

// Membuat objek dari kelas anak
const burger = new Makanan("25 September 2024", "BigMac", 1);

// Fungsi dari kelas induk (Produk) yang diturunkan;
burger.informasi();

// Fungsi dari kelas anak (Makanan) yang dimiliki sendiri;
burger.waktuRusak();

// Fungsi getter nama dari induk yang akan mengeluarkan nama;
console.log("Makanan ini bernama " + burger.getterNama());
```

(JALANKAN KODENYA LALU SCREENSHOT KODE DAN OUTPUT KODENYA)

The screenshot shows a VS Code editor window with a file named 'unguided.js'. The code defines a 'Produk' class with methods 'informasi()', 'getterNama()', and 'waktuRusak()'. A 'Makanan' class extends 'Produk' and overrides the 'waktuRusak()' method. An object 'burger' is created from the 'Makanan' class. The terminal output shows the execution of 'node unguided.js', resulting in three lines of log messages: 'Produk BigMac ada sebanyak 1 pcs.', 'BigMac akan rusak pada tanggal 25 September 2024', and 'Makanan ini bernama BigMac'.

```
1 class Produk {
2   constructor(nama, jumlah) {
3     super(nama, jumlah);
4   }
5   informasi() {
6     console.log(`Produk ${this.nama} ada sebanyak ${this.jumlah} pcs.`);
7   }
8   getterNama(){
9     return this.nama;
10  }
11 }
12
13
14 class Makanan extends Produk {
15   constructor(expired, nama, jumlah) {
16     super(nama, jumlah);
17     this.expired = expired;
18   }
19   waktuRusak() {
20     console.log(`${this.nama} akan rusak pada tanggal ${this.expired}`);
21   }
22 }
23
24 const burger = new Makanan("25 September 2024", "BigMac", 1);
25 burger.informasi();
26 burger.waktuRusak();
27 console.log("Makanan ini bernama " + burger.getterNama());
```

```
alip@Mochalifbudisetyawan:~/tes$ node unguided.js
Produk BigMac ada sebanyak 1 pcs.
BigMac akan rusak pada tanggal 25 September 2024
Makanan ini bernama BigMac
alip@Mochalifbudisetyawan:~/tes$
```

## 1.2 Tugas Pewarisan (+10)

Tambahkan kelas anak dari Kelas Produk sesuai kreativitas kalian dengan **satu atribut tambahan** serta **satu fungsi tambahan** (fungsi harus memakai/melibatkan atribut tambahan). **Buatlah 1 Objek** dari kelas tersebut dan **panggil semua fungsi** yang dapat dipanggil oleh kelas buatan kalian!

(JALANKAN KODENYA LALU SCREENSHOT KODE DAN OUTPUT KODENYA)

```
24 class Elektronik extends Produk {
25   constructor(garansi, nama, jumlah) {
26     super(nama, jumlah);
27     this.garansi = garansi;
28   }
29   informasiGaransi() {
30     console.log(`${this.nama} Memiliki garansi selama ${this.garansi} tahun`);
31   }
32 }
33
34 const burger = new Makanan("25 September 2024", "BigMac", 1);
35 const laptop = new Elektronik(5, "Laptop ASUS", 3);
36 burger.informasi();
37 burger.waktuRusak();
38 console.log("Makanan ini bernama " + burger.getterNama());
39 laptop.informasi();
40 laptop.informasiGaransi();
41 console.log("Nama Produk ini adalah " + laptop.getterNama());
```

DEBUG CONSOLE TERMINAL PORTS

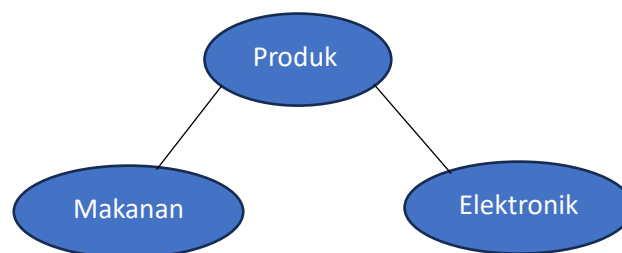
alip@Mochalifbudisetyawan:~/tes\$ node unguided.js

```
Produk BigMac ada sebanyak 1 pcs.
BigMac akan rusak pada tanggal 25 September 2024
Makanan ini bernama BigMac
Produk Laptop ASUS ada sebanyak 3 pcs.
Laptop ASUS Memiliki garansi selama 5 tahun
Nama Produk ini adalah Laptop ASUS
```

### 1.3 Tugas Pewarisan Hierarkis (+5)

Buatlah **Diagram Representasi Hierarki** Pewarisan untuk kode di atas. (Sesuai dengan **Diagram Representasi Hirarki Modul**)

c



## 2.1 Petunjuk Pewarisan dengan Access Modifiers (+5).

**Modifikasi kelas Produk sesuai dengan kode ini!**

**Ubahlah** atribut **nama** menjadi tipe “**Private**” serta modifikasi fungsi di dalamnya agar bisa memakai atribut “Private”.

```
// Kelas Induk (Parent Class)
class Produk {
    #nama; // Private properti, tidak dapat diakses dari luar kelas

    // Konstruktor untuk membuat objek dari kelas Induk.
    constructor(nama, jumlah) {
        this.#nama = nama; // mengisi variabel private
        this.jumlah = jumlah;
    }
    informasi() { // informasi bisa memakai variable private (#nama)
        console.log(`Produk ${this.#nama} ada sebanyak ${this.jumlah} pcs.`);
    } // akan menampilkan nama dan jumlah dari Produk ke console.
}
```

**Modifikasilah kelas Anak sesuai dengan kode ini!**

**Ubahlah** atribut **expired** menjadi “**private**” serta modifikasi fungsi-fungsi di dalamnya agar dapat memakai atribut “private”

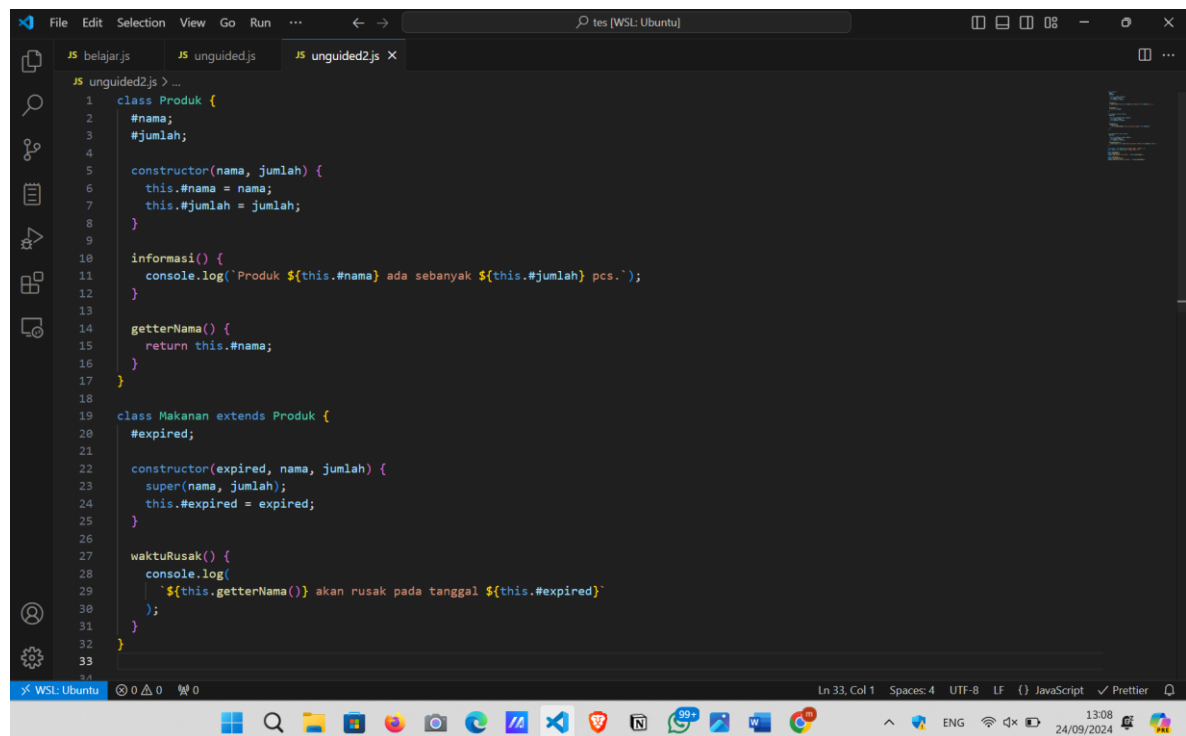
```
// Kelas Anak (Child Class)
// jangan lupa memakai extends untuk pewarisan kelas
class Makanan extends Produk {
    #expired;
    constructor(expired, nama, jumlah) {
        super(nama, jumlah); // Memanggil konstruktor kelas induk (Jangan sampai terbalik)
        this.#expired = expired;
    }
    waktuRusak() { // fungsi public tetap bisa dipanggil meskipun memakai variabel private #
        console.log(`Makanan ini akan rusak pada tanggal ${this.#expired}`);
        // jangan lupa memakai # untuk
        // memakai variabel private
    } // akan menampilkan waktu rusak Makanan
}

// Membuat objek dari kelas anak
const burger = new Makanan("25 September 2024", "BigMac", 1);

// Fungsi dari kelas induk (Produk) yang diturunkan;
burger.informasi();

// Fungsi dari kelas anak (Makanan) yang dimiliki sendiri;
burger.waktuRusak();
```

(SCREENSHOT MODIFIKASI KELAS INDUK DAN ANAK, TIDAK PERLU DI RUN)



```
1 class Produk {
2   #nama;
3   #jumlah;
4
5   constructor(nama, jumlah) {
6     this.#nama = nama;
7     this.#jumlah = jumlah;
8   }
9
10  informasi() {
11    console.log(`Produk ${this.#nama} ada sebanyak ${this.#jumlah} pcs.`);
12  }
13
14  getterNama() {
15    return this.#nama;
16  }
17 }
18
19 class Makanan extends Produk {
20   #expired;
21
22   constructor(expired, nama, jumlah) {
23     super(nama, jumlah);
24     this.#expired = expired;
25   }
26
27   waktuRusak() {
28     console.log(
29       `${this.getterNama()} akan rusak pada tanggal ${this.#expired}`
30     );
31   }
32 }
33
```

## 2.2 Tugas Pewarisan dengan Access Modifiers (+10).

Modifikasi kelas anak yang sebelumnya kalian buat.

Ubahlah atribut tambahan yang kalian buat menjadi tipe “Private” dan modifikasi fungsi yang kalian buat agar tetap bisa berjalan dengan normal.

(JALANKAN KODENYA LALU SCREENSHOT KODE DAN OUTPUT KODENYA)

```
1 class Produk {
2   #nama;
3   #jumlah;
4
5   constructor(nama, jumlah) {
6     this.#nama = nama;
7     this.#jumlah = jumlah;
8   }
9
10  informasi() {
11    console.log("Produk ${this.#nama} ada sebanyak ${this.#jumlah} pcs.");
12  }
13
14  getterNama() {
15    return this.#nama;
16  }
17 }
18
19 class Makanan extends Produk {
20   #expired;
21
22   constructor(expired, nama, jumlah) {
23     super(nama, jumlah);
24     this.#expired = expired;
25   }
26
27   waktuRusak() {
28     console.log("Makanan ini akan rusak pada tanggal " + this.#expired);
29   }
30 }
31
32 class Elektronik extends Produk {
33   #garansi;
34
35   constructor(garansi, nama, jumlah) {
36     super(nama, jumlah);
37     this.#garansi = garansi;
38   }
39
40   informasiGaransi() {
41     console.log(`${this.#nama} memiliki garansi selama ${this.#garansi} tahun.`);
42   }
43 }
44
45 const burger = new Makanan("25 September 2024", "BigMac", 1);
46 const laptop = new Elektronik(5, "Laptop ASUS", 3);
47
48 burger.informasi();
49 burger.waktuRusak();
50 console.log("Makanan ini bernama " + burger.getterNama());
51
52 laptop.informasi();
53 laptop.informasiGaransi();
```

DEBUG CONSOLE TERMINAL PORTS

alip@Mochalifbudisetyawan:~/tes\$ node unguided2.js

```
Produk BigMac ada sebanyak 1 pcs.
BigMac akan rusak pada tanggal 25 September 2024
Makanan ini bernama BigMac
Produk Laptop ASUS ada sebanyak 3 pcs.
Laptop ASUS memiliki garansi selama 5 tahun
Nama produk ini adalah Laptop ASUS
```

## Form Praktikum 03

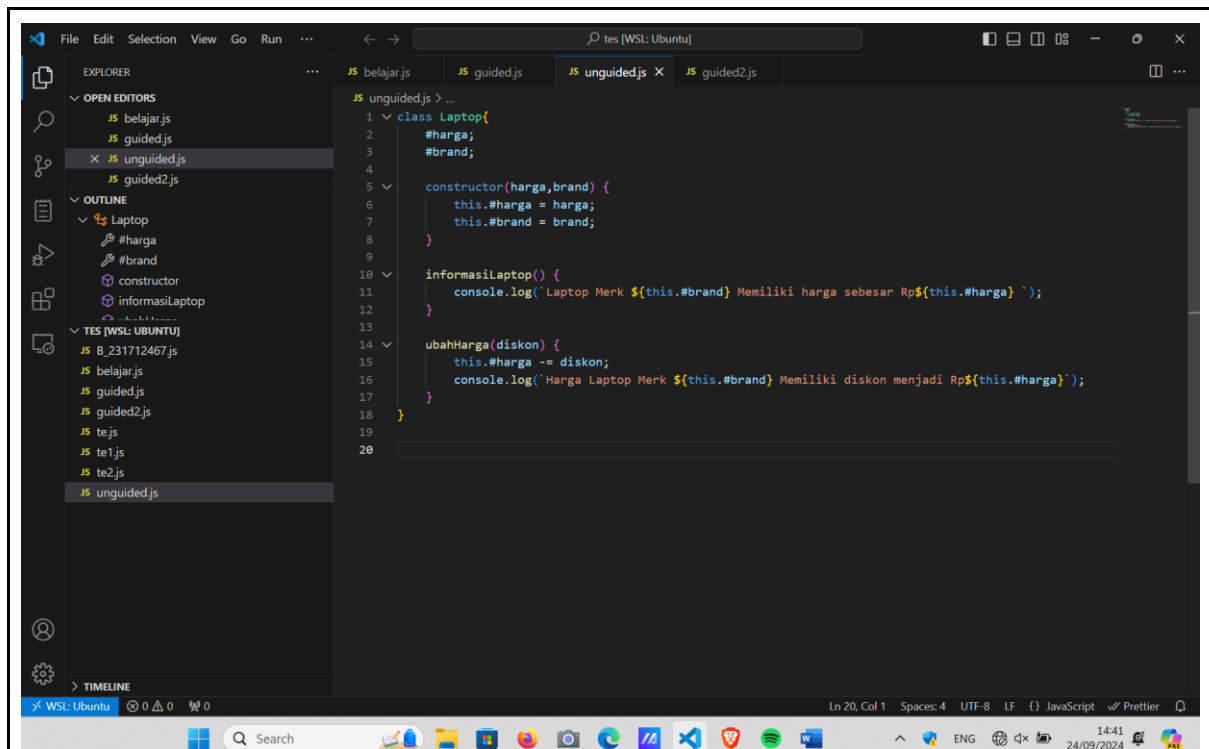
### Pewarisan Access Modifier dan Pewarisan Hierarkis

#### 1.0 PEMBUATAN KELAS (+20)

Buatlah Satu kelas **Induk (Parent Class)** sesuai dengan kreativitas Kalian masing-masing dengan ketentuan :

1. Memiliki 2 atribut, atribut pertama HARUS bertipe angka, atribut kedua HARUS bertipe tulisan (string).
2. Atribut harus memiliki tipe **modifikasi akses PRIVATE**
3. Harus memiliki 2 fungsi, dengan isi / kegunaan bebas, tetapi harus memakai minimal salah satu dari atribut kelas Induk dalam fungsinya.

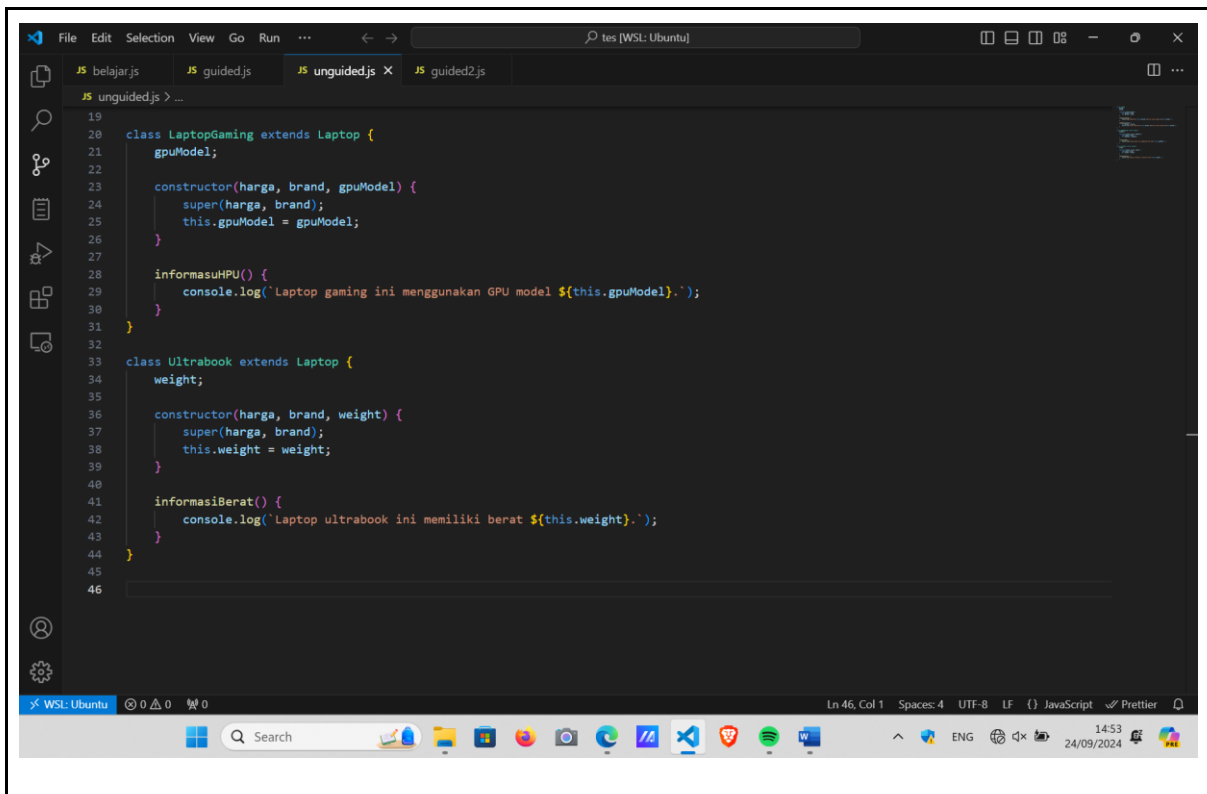
**SCREENSHOT KELAS INDUK TERSEBUT ( TIDAK PERLU MEMBUAT OBJEK)**



Buatlah Dua kelas **Anak (Child Class)** sesuai dengan kreativitas Kalian masing-masing dengan ketentuan :

1. Memiliki 1 atribut unik, atribut HARUS bertipe tulisan (string).
2. Atribut harus memiliki tipe **modifikasi akses PUBLIC**
3. Harus memiliki 1 fungsi, dengan isi / kegunaan bebas, tetapi harus memakai atribut kelas Anak dalam fungsinya.

**SCREENSHOT KELAS ANAK TERSEBUT ( TIDAK PERLU MEMBUAT OBJEK)**



The screenshot shows a VS Code editor window with the file explorer on the left and the editor area on the right. The editor area contains two JavaScript files: `belajar.js` and `guided.js`. The `guided.js` file is open, showing the following code:

```
19
20 class LaptopGaming extends Laptop {
21   gpuModel;
22
23   constructor(harga, brand, gpuModel) {
24     super(harga, brand);
25     this.gpuModel = gpuModel;
26   }
27
28   informasugpu() {
29     console.log('Laptop gaming ini menggunakan GPU model ${this.gpuModel}.');
30   }
31 }
32
33 class Ultrabook extends Laptop {
34   weight;
35
36   constructor(harga, brand, weight) {
37     super(harga, brand);
38     this.weight = weight;
39   }
40
41   informasiBerat() {
42     console.log('Laptop ultrabook ini memiliki berat ${this.weight}.');
43   }
44 }
45
46
```

The status bar at the bottom indicates the file is in the `WSL: Ubuntu` environment, with the cursor at line 46, column 1. The editor is using the `JavaScript` language and the `Prettier` formatter.

Buatlah Dua kelas **Cucu (GrandChild Class)** sesuai dengan kreativitas Kalian masing-masing dengan ketentuan :

1. Masing-masing kelas Cucu diturunkan dari satu kelas Anak.
2. Memiliki 1 atribut unik.
3. Atribut harus memiliki tipe **modifikasi akses PRIVATE**
4. Harus memiliki 1 fungsi, dengan isi / kegunaan bebas, tetapi harus memakai atribut kelas Cucu dalam fungsinya.

**SCREENSHOT KELAS CUCU TERSEBUT ( TIDAK PERLU MEMBUAT OBJEK)**



```
46 class HighLaptop extends LaptopGaming {
47   #overLimit;
48
49   constructor(harga, brand, gpu, overLimit) {
50     super(harga, brand, gpu);
51     this.#overLimit = overLimit;
52   }
53
54   informasiOverLimit() {
55     console.log(`Laptop gaming merk ${this.getterbrand()} memiliki batas overlock GPU sebesar ${this.#overLimit} MHz.`);
56   }
57 }
58
59 class LightUltrabook extends Ultrabook {
60   #baterai;
61
62   constructor(harga, brand, berat, baterai) {
63     super(harga, brand, berat);
64     this.#baterai = baterai;
65   }
66
67   informasiBaterai() {
68     console.log(`Ultrabook merk ${this.getterbrand()} Memiliki daya tahan baterai hingga ${this.#baterai} jam.`);
69   }
70 }
71
```

## 2.0 PEMBUATAN OBJEK DAN PEMANGGILAN FUNGSI (+20)

Buatlah OBJEK dari masing-masing kelas Cucu sesuai dengan kelas yang kalian buat! Panggil semua fungsi yang dapat dipanggil oleh kelas cucu tersebut secara berurutan dari paling atas (kelas induk) ke paling bawah (kelas cucu).

**SCREENSHOT PEMBUATAN OBJEK DAN PEMANGGILAN FUNGSI BESERTA OUTPUTNYA!**

```
File Edit Selection View Go Run ... tes [WSL: Ubuntu]
JS belajar.js JS guided.js JS unguided.js JS guided2.js
JS unguided.js > HighLaptop > informasiOverLimit
56 class LightUltrabook extends Ultrabook {
64   informasiBaterai() {
65     console.log("Ultrabook ini memiliki daya tahan baterai hingga 12 jam.");
66   }
67 }
68
69 const HighLaptop1 = new HighLaptop(22000000, "ADVAN", "NVIDIA RTX 3090", 2100);
70 HighLaptop1.informasiGPU();
71 HighLaptop1.informasiOverLimit();
72 const ultrabookRingan = new LightUltrabook(22000000, "HP", "1.0 kg", 12);
73 ultrabookRingan.informasiLaptop();
74 ultrabookRingan.informasiBerat();
75 ultrabookRingan.informasiBaterai();

DEBUG CONSOLE TERMINAL PORTS
TERMINAL
Laptop ultrabook ini memiliki berat 1.0 kg.
Ultrabook merk undefined Memiliki daya tahan baterai hingga 12 jam.
alip@Mochalifbudisetyawan:~/tes$ node unguided.js
Laptop gaming merk ini menggunakan GPU model NVIDIA RTX 3090.
Laptop gaming merk ini memiliki batas overlock GPU sebesar 2100 MHz.
Laptop Merk HP Memiliki harga sebesar Rp22000000
Laptop ultrabook ini memiliki berat 1.0 kg.
Ultrabook merk ini Memiliki daya tahan baterai hingga 12 jam.
alip@Mochalifbudisetyawan:~/tes$
```

Silahkan panggil atribut kelas Induk melalui objek Cucu secara langsung memakai console.log().  
(Cukup 1 Objek Cucu saja)

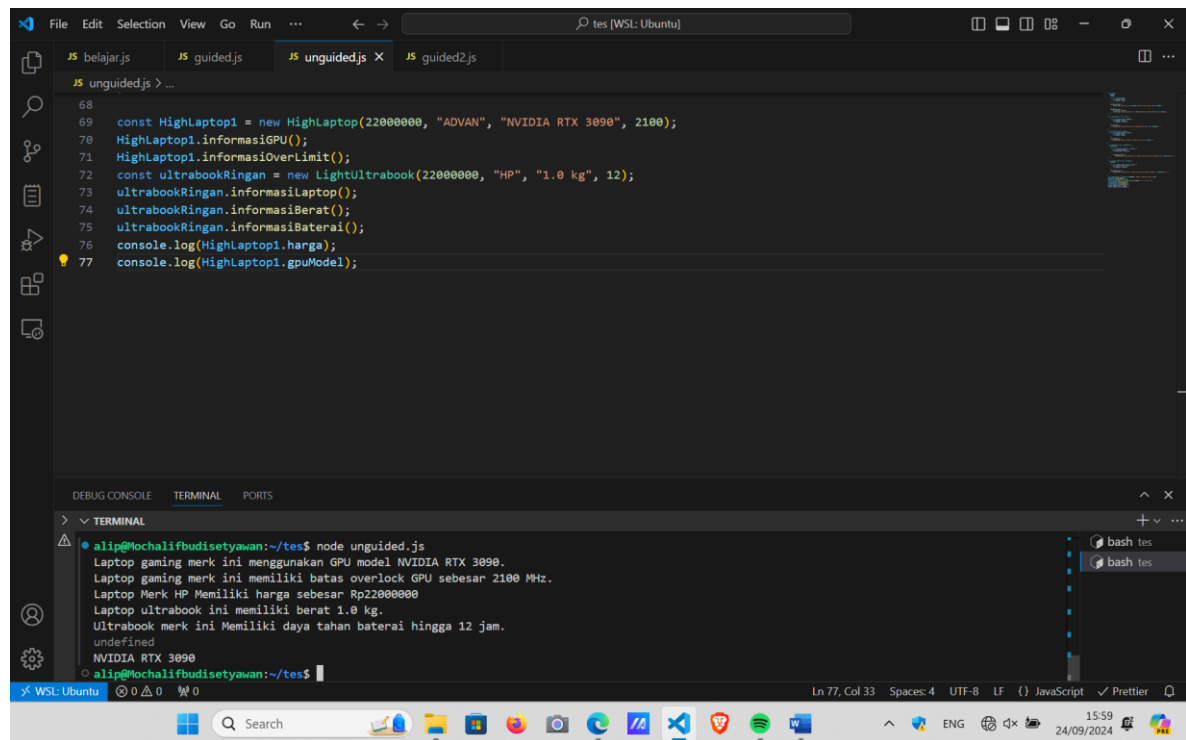
## SCREENSHOT PEMANGGILAN ATRIBUT BESERTA OUTPUTNYA!

```
File Edit Selection View Go Run ... tes [WSL: Ubuntu]
JS belajar.js JS guided.js JS unguided.js JS guided2.js
JS unguided.js > ...
67 }
68
69 const HighLaptop1 = new HighLaptop(22000000, "ADVAN", "NVIDIA RTX 3090", 2100);
70 HighLaptop1.informasiGPU();
71 HighLaptop1.informasiOverLimit();
72 const ultrabookRingan = new LightUltrabook(22000000, "HP", "1.0 kg", 12);
73 ultrabookRingan.informasiLaptop();
74 ultrabookRingan.informasiBerat();
75 ultrabookRingan.informasiBaterai();
76 console.log(HighLaptop1.harga);

DEBUG CONSOLE TERMINAL PORTS
TERMINAL
Harga Laptop Merk ADVAN Memiliki diskon menjadi Rp20000000
alip@Mochalifbudisetyawan:~/tes$ node unguided.js
Laptop gaming merk ini menggunakan GPU model NVIDIA RTX 3090.
Laptop gaming merk ini memiliki batas overlock GPU sebesar 2100 MHz.
Laptop Merk HP Memiliki harga sebesar Rp22000000
Laptop ultrabook ini memiliki berat 1.0 kg.
Ultrabook merk ini Memiliki daya tahan baterai hingga 12 jam.
undefined
alip@Mochalifbudisetyawan:~/tes$
```

Silahkan panggil atribut kelas Anak melalui objek Cucu secara langsung memakai console.log().  
(Cukup 1 Objek Cucu saja)

### SCREENSHOT PEMANGGILAN ATRIBUT BESERTA OUTPUTNYA!



The screenshot shows a VS Code editor with a file named 'unguided.js'. The code defines two classes: 'HighLaptop' and 'LightUltrabook'. 'HighLaptop' has attributes 'price', 'brand', 'gpuModel', and 'overclockLimit'. 'LightUltrabook' has attributes 'price', 'brand', 'weight', and 'batteryLife'. The code creates instances of these classes and calls methods like 'informasiGPU()', 'informasiOverLimit()', 'informasiLaptop()', 'informasiBerat()', and 'informasiBaterai()'. It also uses 'console.log()' to print the price and GPU model of the 'HighLaptop' instance.

```
68
69 const HighLaptop1 = new HighLaptop(22000000, "ADVAN", "NVIDIA RTX 3090", 2100);
70 HighLaptop1.informasiGPU();
71 HighLaptop1.informasiOverLimit();
72 const ultrabookRingan = new LightUltrabook(22000000, "HP", "1.0 kg", 12);
73 ultrabookRingan.informasiLaptop();
74 ultrabookRingan.informasiBerat();
75 ultrabookRingan.informasiBaterai();
76 console.log(HighLaptop1.harga);
77 console.log(HighLaptop1.gpuModel);
```

The terminal output shows the execution of the code. It prints the price of the laptop (Rp22000000), the GPU model (NVIDIA RTX 3090), and the battery life of the ultrabook (12 jam). There is also an 'undefined' output for the 'informasiOverLimit()' method.

```
alip@tehalifbudisetyawan:~/tes$ node unguided.js
Laptop gaming merk ini menggunakan GPU model NVIDIA RTX 3090.
Laptop gaming merk ini memiliki batas overlock GPU sebesar 2100 MHz.
Laptop Merk HP Memiliki harga sebesar Rp22000000
Laptop ultrabook ini memiliki berat 1.0 kg.
Ultrabook merk ini Memiliki daya tahan baterai hingga 12 jam.
undefined
NVIDIA RTX 3090
alip@tehalifbudisetyawan:~/tes$
```

### 3.0 ESSAY (+20)

Jelaskan mengapa Anda bisa memanggil semua fungsi pada objek Cucu dengan bahasa Anda.

Karena di JavaScript memiliki konsep pewarisan. Dimana di dalam pewarisan, kelas cucu bisa mewarisi semua properti dan metode dari kelas induk dan kelas anak. Ketika membuat objek dari kelas cucu, objek tersebut memiliki akses ke semua metode yang didefinisikan dalam kelas induk dan kelas anak. Jadi, meskipun memanggil metode yang didefinisikan di kelas induk, objek cucu akan dapat mengakses dan menjalankannya.

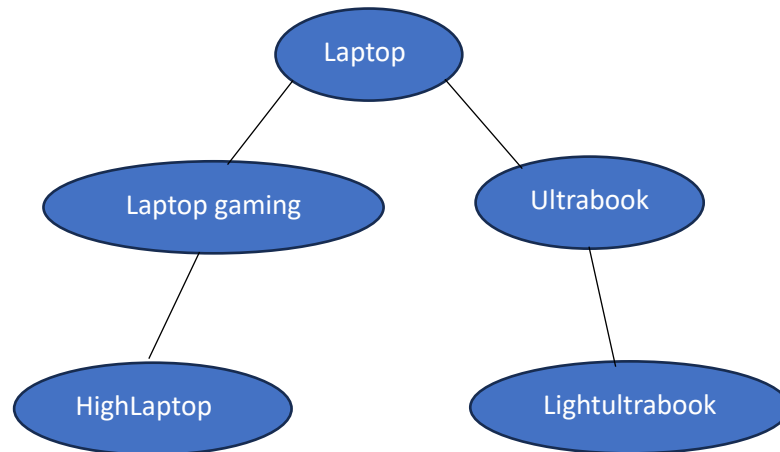
Jelaskan mengapa Anda tidak bisa mengakses atribut kelas Induk di objek Cucu secara langsung, sedangkan Anda bisa mengakses atribut kelas anak pada objek Cucu secara langsung.

Karena kelas induk di deklarasi dengan akses private secara langsung di objek cucu karena sifat private di JavaScript membatasi akses hanya dalam kelas tempat atribut itu di deklarasi. Dan di

class anak bisa diakses langsung oleh cucu karena sifatnya Public. Dimana penggunaan akses public bisa dilakukan dari mana saja, termasuk di luar kelas tempat atribut itu di deklarasikan.

Gambarkan Diagram Representasi Hirarki Pewarisan Kelas Anda secara lengkap!

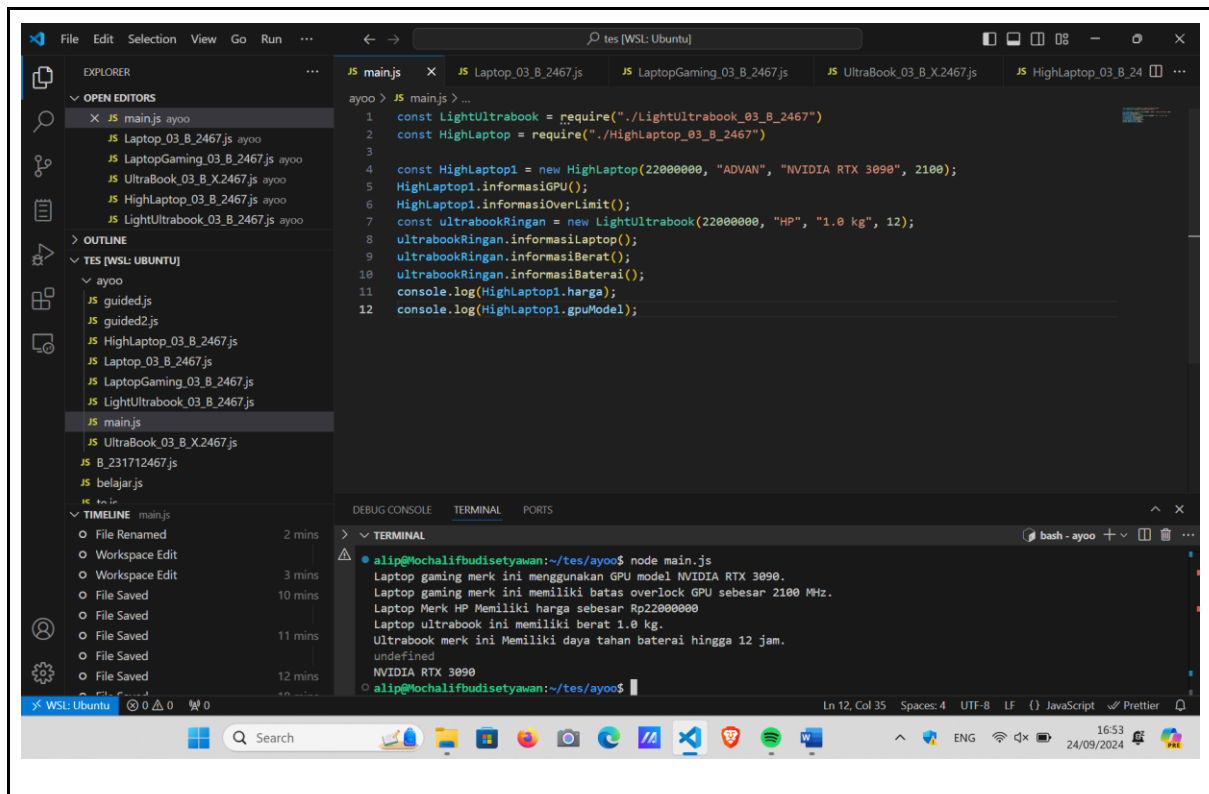
(Boleh memakai menggambar di kertas dan foto kesini jika mau, yang terpenting rapi!)



#### 4.0 BONUS (+10)

Gunakan modularisasi file di Javascript untuk setiap kelas. Untuk file yang digunakan untuk menjalankan program beri nama 'main.js'.

**SCREENSHOT SELURUH KELAS DAN MAIN.JS DALAM MODULARISASI FILE.**



## Ketentuan Pengerjaan

1. Pengumpulan maksimal **23.59, 24 September 2024**.
2. Screenshot tidak sesuai ketentuan soal **(-5 / kesalahan)**
3. Kumpulkan file javascript Kalian dengan ketentuan :
  - a. Bagi yang tidak mengerjakan bonus, penamaan file javascript akan menjadi :
    - i. **GD\_03\_Y\_XXXX.js** : untuk Guided
    - ii. **UGD\_03\_Y\_XXXX.js** : untuk Unguided / Praktikum
  - b. Bagi yang mengerjakan Bonus penamaan file menjadi :
    - i. **NamaKelas\_03\_Y\_XXXX.js** : untuk setiap kelas.
4. Kumpulkan dalam format .pdf dengan format penamaan file : **03\_Y\_XXXX**
  - a. **Y** = kelas Kalian
  - b. **X** = 4 Digit NPM Kalian
5. Gabungkan file javascript dan form praktikum menjadi 1 folder, lalu jadikan format .zip, dengan penamaan :
  - a. **03\_Y\_XXXX.zip**

6. Untuk yang mengambil bonus dikumpulkan di uploader Bonus. Dengan format penamaan **Bonus\_03\_Y\_XXXX.zip**
7. Ketentuan nama tidak sesuai **(-5)**