

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: Yosia Agnesa Togatorop

NPM: 231712616

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
}

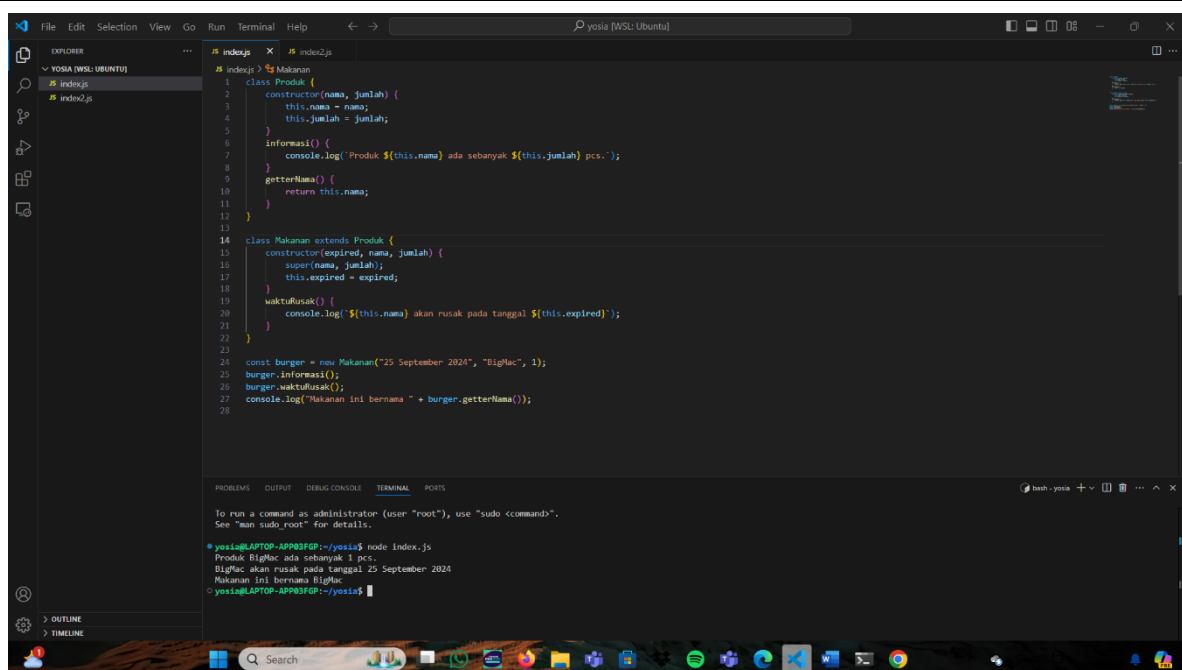
// Membuat objek dari kelas anak
const burger = new Makanan("25 September 2024", "BigMac", 1);

// 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 code editor with the following code in `index.js`:

```
1 class Produk {
2   constructor(nama, jumlah) {
3     this.nama = nama;
4     this.jumlah = jumlah;
5   }
6   informasi() {
7     console.log("Produk ${this.nama} ada sebanyak ${this.jumlah} pcs.");
8   }
9   getterNama() {
10    return this.nama;
11  }
12 }
13
14 class Makanan extends Produk {
15   constructor(expired, nama, jumlah) {
16     super(nama, jumlah);
17     this.expired = expired;
18   }
19   waktuRusak() {
20     console.log(`${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());
28
```

The terminal output shows the following results:

```

yosia@LAPTOP-APR03FSP:~/yosia$ node index.js
Produk BigMac ada sebanyak 1 pcs.
BigMac akan rusak pada tanggal 25 September 2024
Makanan ini bernama BigMac
yosia@LAPTOP-APR03FSP:~/yosia$

```

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)

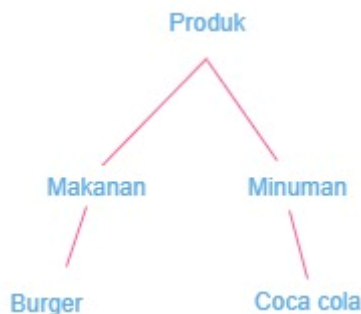
```
1 class Produk {
2   constructor(nama, jumlah) {
3     this.nama = nama;
4     this.jumlah = jumlah;
5   }
6   informasi() {
7     console.log('Produk ${this.nama} ada sebanyak ${this.jumlah} pcs.');
8   }
9   getNama() {
10    return this.nama;
11  }
12 }
13
14 class Makanan extends Produk {
15   constructor(expired, nama, jumlah) {
16     super(nama, jumlah);
17     this.expired = expired;
18   }
19   waktuKadaluarsa() {
20     console.log(`${this.nama} akan rusak pada tanggal ${this.expired}`);
21   }
22 }
23
24 class Minuman extends Produk {
25   constructor(volume, nama, jumlah) {
26     super(nama, jumlah);
27     this.volume = volume; // string/tampilan
28   }
29   tampilkanVolume() {
30     console.log(`${this.nama} memiliki volume sebanyak ${this.volume} ml.`);
31   }
32 }
33
34 // Membuat objek dari kelas Minuman
35 const cola = new Minuman(500, "Coca-Cola", 3);
36
37 // Memanggil semua fungsi
38 cola.informasi(); // Fungsi dari kelas Produk
39 cola.tampilkanVolume(); // Fungsi dari kelas Minuman
40 console.log('Minuman ini bernama ' + cola.getNama()); // Fungsi dari kelas Produk
```

Terminal Output:

```
Produk Coca-Cola ada sebanyak 3 pcs.
Coca-Cola memiliki volume sebanyak 500 ml.
Minuman ini bernama Coca-Cola
```

1.3 Tugas Pewarisan Hierarkis (+5)

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



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   constructor(nama, jumlah) {
4     this.nama = nama;
5     this.jumlah = jumlah;
6   }
7   informasi() {
8     console.log(`Produk ${this.nama} ada sebanyak ${this.jumlah} pcs.`);
9   }
10  getterNama() {
11    return this.nama;
12  }
13 }
14
15 class Makanan extends Produk {
16   #expired;
17   constructor(expired, nama, jumlah) {
18     super(nama, jumlah);
19     this.expired = expired;
20   }
21   waktuRusak() {
22     console.log(`${this.nama} akan rusak pada tanggal ${this.expired}`);
23   }
24 }
25
26 const burger = new Makanan("25 September 2024", "Burger", 1);
27 burger.informasi();
28 burger.waktuRusak();
29 console.log(`Makanan ini bernama * + burger.getterNama();
30
```

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   constructor(nama, jumlah) {
3     this.nama = nama;
4     this.jumlah = jumlah;
5   }
6   informasi() {
7     console.log(`Produk ${this.nama} ada sebanyak ${this.jumlah} pcs.`);
8   }
9   getterNama() {
10    return this.nama;
11  }
12 }
13
14 class Makanan extends Produk {
15   #expired;
16   constructor(expired, nama, jumlah) {
17     super(nama, jumlah);
18     this.expired = expired;
19   }
20   waktuRusak() {
21     console.log(`${this.nama} akan rusak pada tanggal ${this.expired}`);
22   }
23 }
24
25 class Minuman extends Produk {
26   #volume;
27   constructor(volume, nama, jumlah) {
28     super(nama, jumlah);
29     this.volume = volume;
30   }
31   tampilkanVolume() {
32     console.log(`${this.nama} memiliki volume sebanyak ${this.volume} ml.`);
33   }
34 }
35
36 const cola = new Minuman(500, "Coca-Cola", 3);
37 cola.informasi();
38 cola.tampilkanVolume();
39 console.log(`Minuman ini bernama * + cola.getterNama();
40
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

To run a command as administrator (user "root"), use "sudo" command.
See "man sudo_root" for details.

```
* yosia@IP: /workspace --yosia$ node index2.js
Produk Coca-Cola ada sebanyak 3 pcs.
Coca-Cola memiliki volume sebanyak 500 ml.
Minuman ini bernama Coca-Cola
*yosia@IP: /workspace --yosia$
```

Form Praktikum 03

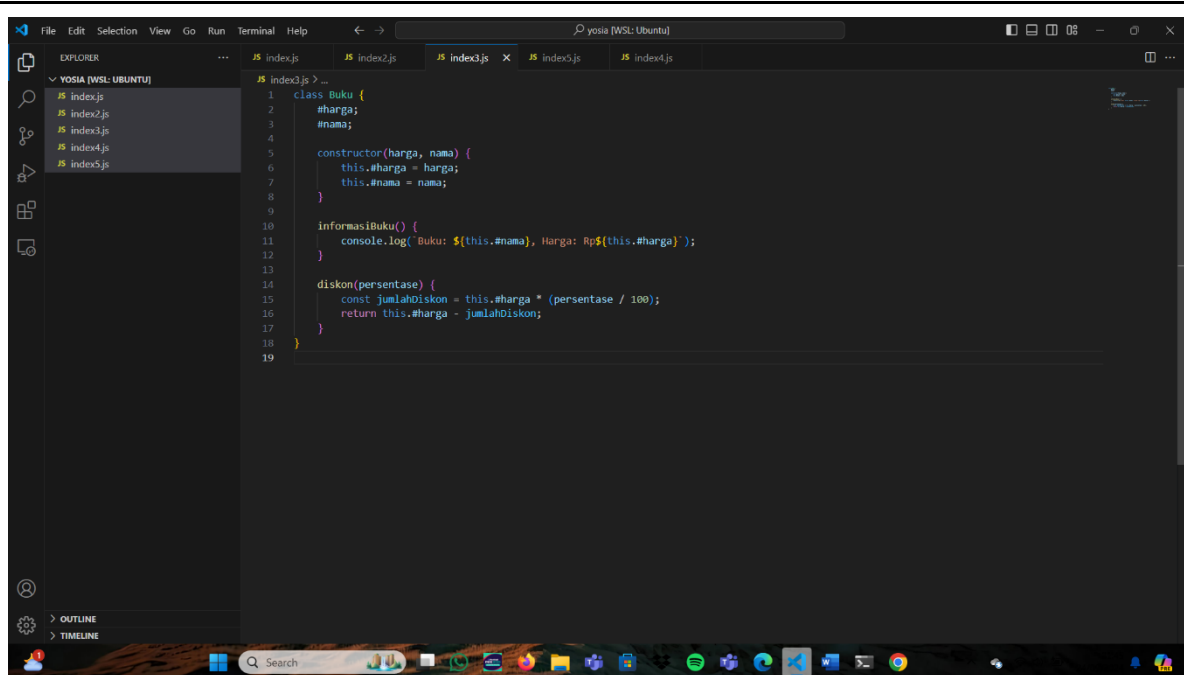
Pewarisan Access Modifier dan Pewarisan Hierarkis

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)

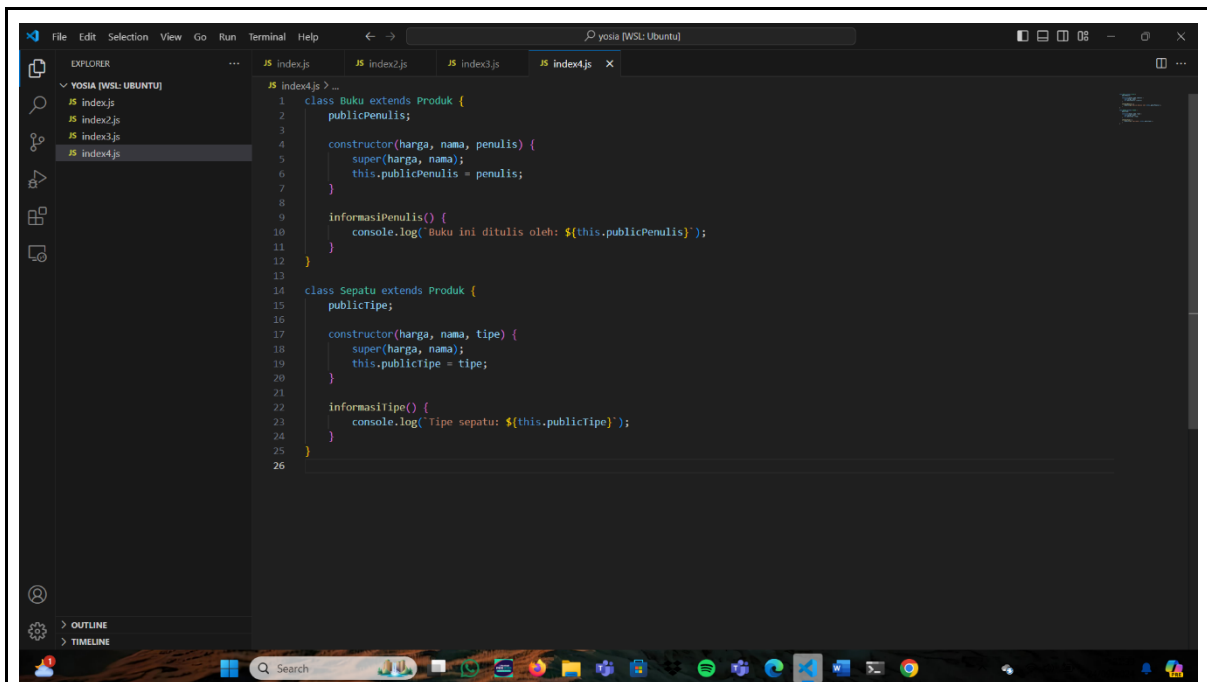


```
1 class Buku {
2   #harga;
3   #nama;
4
5   constructor(harga, nama) {
6     this.#harga = harga;
7     this.#nama = nama;
8   }
9
10  informasiBuku() {
11    console.log('Buku: ${this.#nama}, Harga: Rp${this.#harga}');
12  }
13
14  diskon(persentase) {
15    const jumlahDiskon = this.#harga * (persentase / 100);
16    return this.#harga - jumlahDiskon;
17  }
18 }
19
```

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)

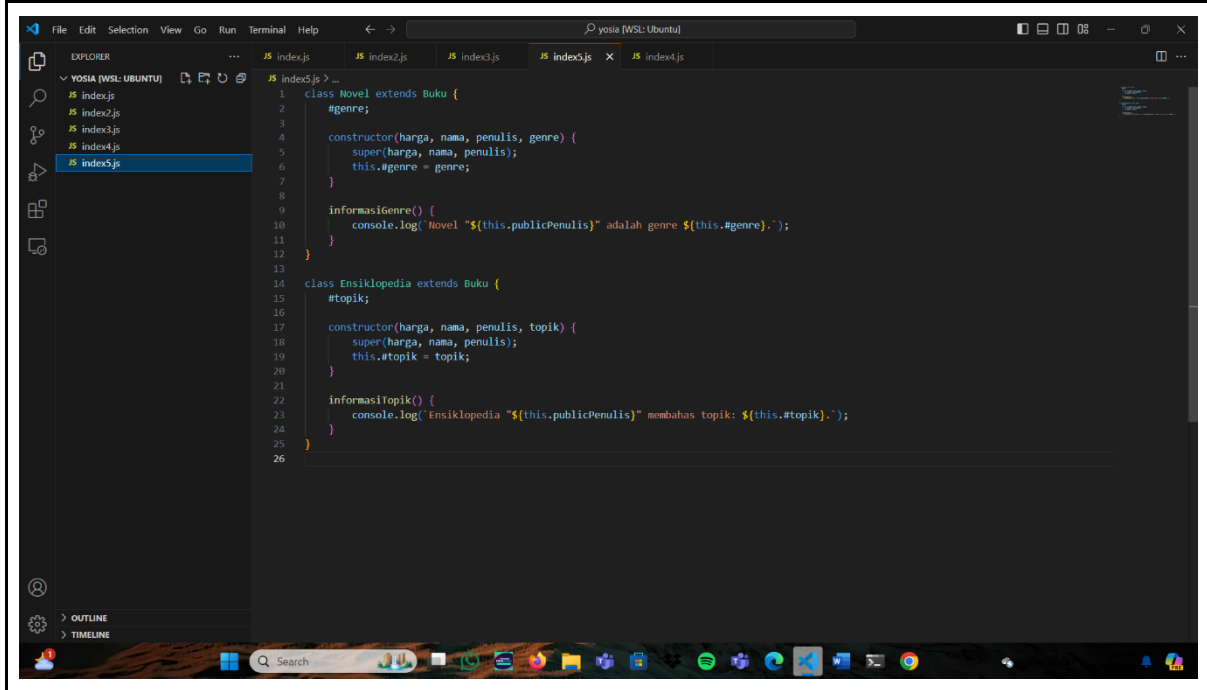


```
1 class Buku extends Produk {
2   publicPenulis;
3
4   constructor(harga, nama, penulis) {
5     super(harga, nama);
6     this.publicPenulis = penulis;
7   }
8
9   informasiPenulis() {
10    console.log('Buku ini ditulis oleh: ${this.publicPenulis}');
11  }
12 }
13
14 class Sepatu extends Produk {
15   publicTipe;
16
17   constructor(harga, nama, tipe) {
18     super(harga, nama);
19     this.publicTipe = tipe;
20   }
21
22   informasiTipe() {
23    console.log('Tipe sepatu: ${this.publicTipe}');
24  }
25 }
26
```

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)



```
1 class Novel extends Buku {
2   #genre;
3
4   constructor(harga, nama, penulis, genre) {
5     super(harga, nama, penulis);
6     this.#genre = genre;
7   }
8
9   informasiGenre() {
10    console.log('Novel "${this.publicPenulis}" adalah genre ${this.#genre}.');
11  }
12 }
13
14 class Ensiklopedia extends Buku {
15   #topik;
16
17   constructor(harga, nama, penulis, topik) {
18     super(harga, nama, penulis);
19     this.#topik = topik;
20   }
21
22   informasiTopik() {
23    console.log('Ensiklopedia "${this.publicPenulis}" membahas topik: ${this.#topik}.');
24  }
25 }
26
```

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!

The first screenshot shows the code for a class hierarchy. It defines a base class `Buku` with attributes `#harga` and `#nama`, a constructor, an `informasiBuku()` method, and a `diskon(persentase)` method. Two subclasses are defined: `Novel` (extending `Buku`) with an additional `#genre` attribute and `informasiGenre()` method, and `Ensiklopedia` (extending `Buku`) with an additional `#topik` attribute and `informasiTopik()` method.

```
1 // Kelas Induk
2 class Buku {
3     #harga;
4     #nama;
5
6     constructor(harga, nama) {
7         this.#harga = harga;
8         this.#nama = nama;
9     }
10
11     informasiBuku() {
12         console.log("Buku: ${this.#nama}, Harga: Rp${this.#harga}");
13     }
14
15     diskon(persentase) {
16         const jumlahDiskon = this.#harga * (persentase / 100);
17         return this.#harga - jumlahDiskon;
18     }
19 }
20
21 // Kelas Cucu 1
22 class Novel extends Buku {
23     #genre;
24
25     constructor(harga, nama, genre) {
26         super(harga, nama);
27         this.#genre = genre;
28     }
29
30     informasiGenre() {
31         console.log("Novel "${this.#nama}" adalah genre ${this.#genre}.");
32     }
33 }
34
35 // Kelas Cucu 2
36 class Ensiklopedia extends Buku {
37     #topik;
38
39     constructor(harga, nama, topik) {
40         super(harga, nama);
41         this.#topik = topik;
42     }
43
44     informasiTopik() {
45         console.log("Ensiklopedia "${this.#nama}" membahas topik: ${this.#topik}.");
46     }
47 }
48
49 // Membuat objek dari kelas Cucu
50 const novel1 = new Novel(100000, "Laskar Pelangi", "Fiksi");
51 const ensiklopedia1 = new Ensiklopedia(200000, "Ensiklopedia Umum", "Ilmu Pengetahuan");
52
53 // Memanggil semua fungsi secara berurutan
54 console.log("Informasi Novel:");
55 novel1.informasiBuku(); // Fungsi dari kelas Induk
56 novel1.informasiGenre(); // Fungsi dari kelas Cucu
57
58 console.log("\nInformasi Ensiklopedia:");
59 ensiklopedia1.informasiBuku(); // Fungsi dari kelas Induk
60 ensiklopedia1.informasiTopik(); // Fungsi dari kelas cucu
```

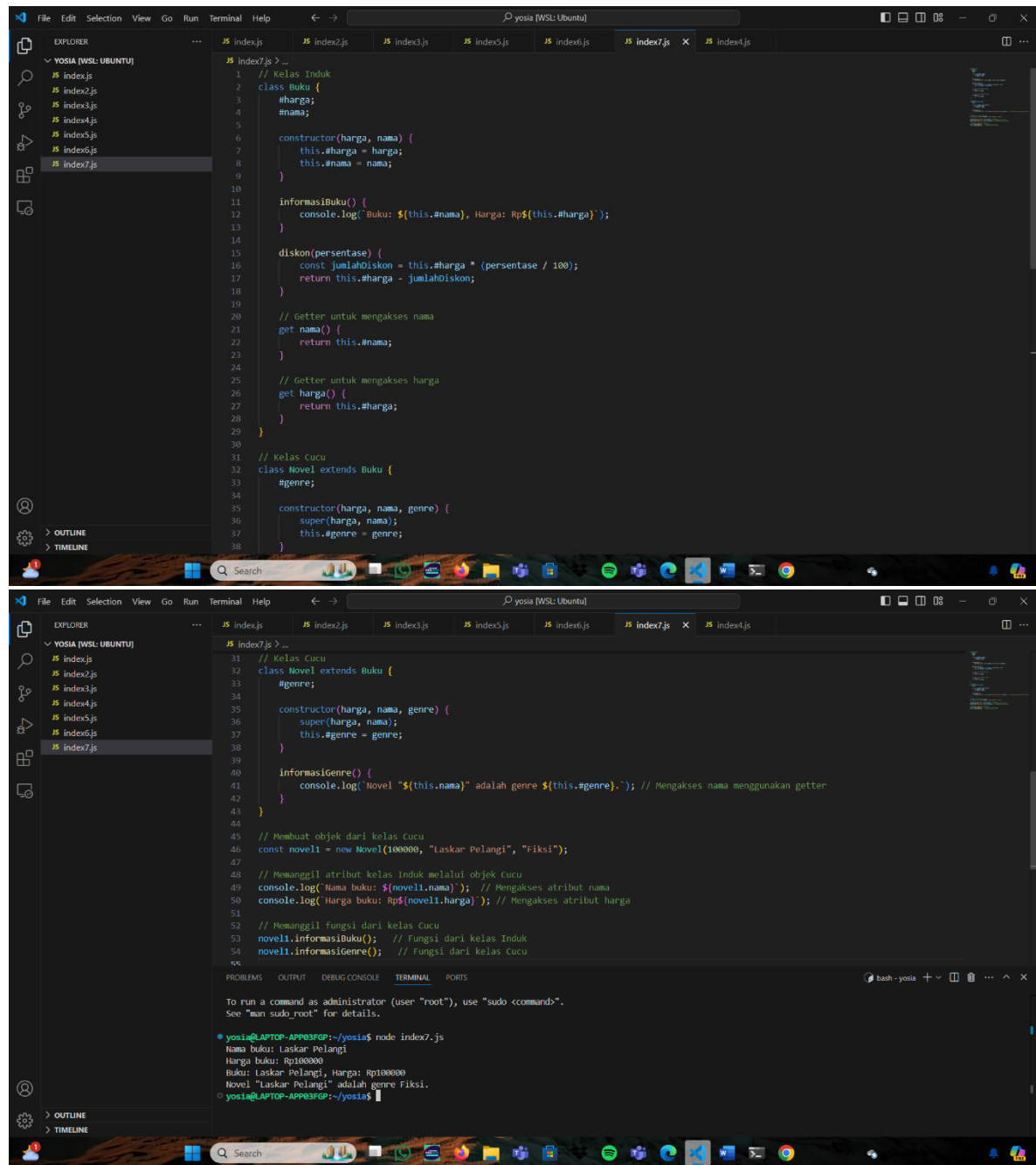
The second screenshot shows the terminal output of the code execution. It displays the output of the `console.log` statements, showing the details of the `Novel` and `Ensiklopedia` objects and the results of their methods.

```
Informasi Novel:
Buku: Laskar Pelangi, Harga: Rp100000
Novel "undefined" adalah genre Fiksi.

Informasi Ensiklopedia:
Buku: Ensiklopedia Umum, Harga: Rp200000
Ensiklopedia "undefined" membahas topik: Ilmu Pengetahuan.
```

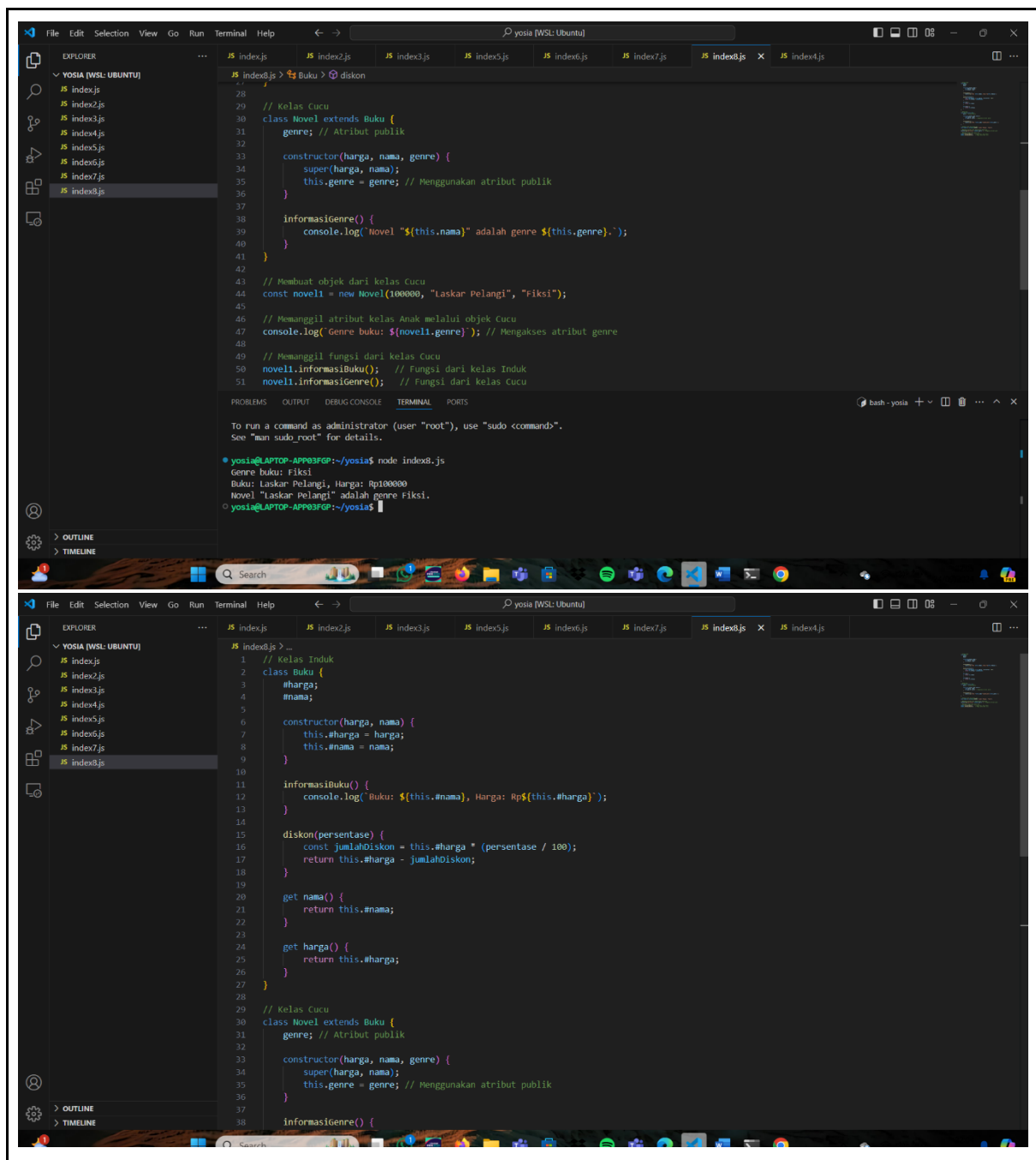
Silahkan panggil atribut kelas Induk melalui objek Cucu secara langsung memakai `console.log()`. (Cukup 1 Objek Cucu saja)

SCREENSHOT PEMANGGILAN ATRIBUT BESERTA OUTPUTNYA!



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

SCREENSHOT PEMANGGILAN ATRIBUT BESERTA OUTPUTNYA!



3.0 ESSAY (+20)

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

Dalam pemrograman berorientasi objek, kita dapat memanggil semua fungsi pada objek cucu karena adanya konsep pewarisan (inheritance). Kelas Novel sebagai turunan dari kelas Buku mewarisi semua atribut dan metode yang ada di dalam kelas induk tersebut. Dengan demikian, objek novel1 yang merupakan instansi dari kelas Novel dapat mengakses metode seperti informasiBuku() dan diskon() yang didefinisikan dalam kelas Buku. Selain itu, kelas Novel juga memiliki metode spesifiknya sendiri, seperti informasiGenre(), yang dapat dipanggil langsung melalui objek tersebut. Penggunaan

getter untuk atribut private, seperti nama, memungkinkan kita untuk mengakses nilai atribut tanpa mengaksesnya secara langsung, sehingga menjaga prinsip enkapsulasi. Semua fitur ini berkontribusi pada konsistensi dan fleksibilitas objek dalam pemrograman, memungkinkan kita untuk menggunakan metode dari kelas induk atau kelas anak dengan cara yang terstruktur dan modular. Dengan demikian, semua fungsi dapat dipanggil dengan mudah melalui objek cucu, memperkuat keunggulan dari pendekatan berorientasi objek.

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.

Dalam pemrograman berorientasi objek, perbedaan aksesibilitas antara atribut kelas induk dan kelas anak berkaitan dengan modifikasi akses yang diterapkan pada atribut tersebut. Atribut yang dideklarasikan sebagai private di kelas induk, seperti atribut yang ada pada kelas Buku, hanya dapat diakses di dalam kelas itu sendiri. Meskipun kelas cucu, seperti Novel, mewarisi atribut dan metode dari kelas induk, ia tidak dapat mengakses atau memodifikasi atribut private secara langsung, melainkan hanya dapat menggunakan metode publik yang disediakan oleh kelas induk. Hal ini dirancang untuk melindungi data dan menjaga integritas objek, mendukung prinsip enkapsulasi. Sebaliknya, atribut yang dideklarasikan sebagai publik di kelas anak, seperti atribut genre di kelas Novel, dapat diakses langsung melalui objek cucu, memberikan fleksibilitas dalam penggunaan atribut tersebut. Dengan demikian, perbedaan ini menunjukkan pentingnya prinsip enkapsulasi dalam desain kelas, di mana atribut private melindungi data dari akses luar, sementara atribut publik dirancang untuk diakses dan digunakan dengan bebas, meningkatkan keamanan dan struktur kode dalam pemrograman berorientasi objek.

Gambarkan Diagram Representasi Hirarki Pewarisan Kelas Anda secara lengkap! (Boleh memakai menggambar di kertas dan foto kesini jika mau, yang terpenting rapi!)

