

LAPORAN PRAKTIKUM

PEMROGRAMAN BERORIENTASI OBJEK

PERTEMUAN KE-8



Disusun Oleh :

NAMA : Raden Isnawan Argi Aryasatya

NIM : 195410257

JURUSAN : Teknik Informatika

JENJANG : S1

KELAS : TI-5

Laboratorium Terpadu
Sekolah Tinggi Manajemen Informatika Komputer
AKAKOM
YOGYAKARTA

2021

PERTEMUAN KE-8 **(PEWARISAN)**

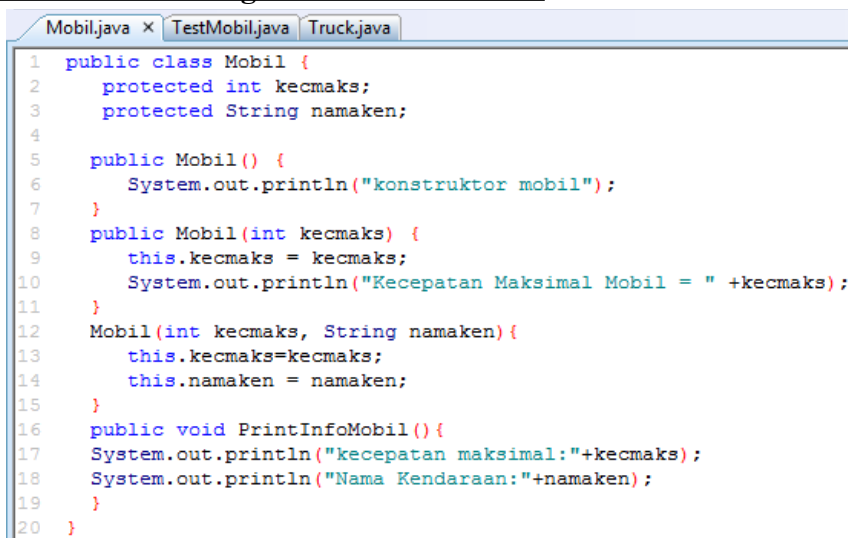
TUJUAN

Dapat menggunakan overriding metode, menggunakan klausa super dan dapat menggunakan klausa final pada metode dan kelas

DASAR TEORI

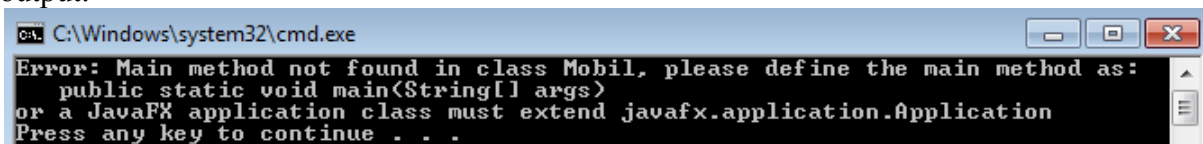
Dalam Java, juga memungkinkan untuk mendeklarasikan class-class yang tidak dapat menjadi subclass. Class ini dinamakan class final. Untuk mendeklarasikan class untuk menjadi final kita hanya menambahkan kata kunci final dalam deklarasi class. Lalu ada keyword super, yaitu pada saat penciptaan objek dari subclass (kelas anak), konstruktor dari superclass (kelas induk) akan dipanggil. Kemudian ada overriding method yaitu method instan dalam subclass dengan nama dan paramater yang sama, serta return type sama dengan yang ada dalam superclass.

PRAKTIK 1: Program Membuat Mobil



```
Mobil.java x TestMobil.java Truck.java
1 public class Mobil {
2     protected int kecmaks;
3     protected String namaken;
4
5     public Mobil() {
6         System.out.println("konstruktor mobil");
7     }
8     public Mobil(int kecmaks) {
9         this.kecmaks = kecmaks;
10        System.out.println("Kecepatan Maksimal Mobil = " +kecmaks);
11    }
12    Mobil(int kecmaks, String namaken){
13        this.kecmaks=kecmaks;
14        this.namaken = namaken;
15    }
16    public void PrintInfoMobil(){
17        System.out.println("kecepatan maksimal:"+kecmaks);
18        System.out.println("Nama Kendaraan:"+namaken);
19    }
20 }
```

output:



```
C:\Windows\system32\cmd.exe
Error: Main method not found in class Mobil, please define the main method as:
  public static void main(String[] args)
or a JavaFX application class must extend javafx.application.Application
Press any key to continue . . .
```

penjelasan:

Pada praktik pertama pada laporan ini, kita membuat sebuah program berisi class induk (parent class), super class, atau bisa juga disebut dengan base class yang nantinya variabel/atribut dan method-method nya akan di-inherit ke child class. Class parent tersebut disini kita sebut dengan Mobil. Kita mendeklarasikannya dengan baris kode **public class Mobil {**. Di dalam kelas tersebut ada dua variabel/atribut yang memiliki modifier protected yaitu kecmaks yang ditulis dengan **protected int kecmaks;** dan namaken yang ditulis dengan **protected String namaken;**. Lalu kita membuat konstruktor mobil yang isinya tidak bisa di-inherit. Lalu kita membuat konstruktor sebagai method yang digunakan untuk membuat dan menginisialisasi sebuah object baru dan sekaligus mencetak outputnya yaitu diawali dengan menuliskan **public Mobil(int kecmaks)** yang di dalamnya ada pengaksesan variabel yaitu **this.kecmaks = kecmaks;** lalu mencetak output dengan menuliskan **System.out.println("Kecepatan Maksimal Mobil = " +kecmaks);**.

Kemudian kita membuat satu lagi konstruktor yang memiliki parameter int kecmaks, String namaken yaitu dengan menuliskan **Mobil(int kecmaks, String namaken)** yang di dalamnya ada deklarasi keyword this yang digunakan untuk mengakses atribut/variabel pada class yang ditulis seperti ini:

```
this.kecmaks=kecmaks;  
    this.namaken = namaken;
```

Terakhir kita buat method tidak memiliki nilai kembali/return yaitu dengan menuliskan baris kode **public void PrintInfoMobil(){**. Method ini kita buat untuk mencetak kalimat yang nanti akan ditampilkan pada output. Di dalam method tersebut, kalimat yang akan dicetak pada output adalah:

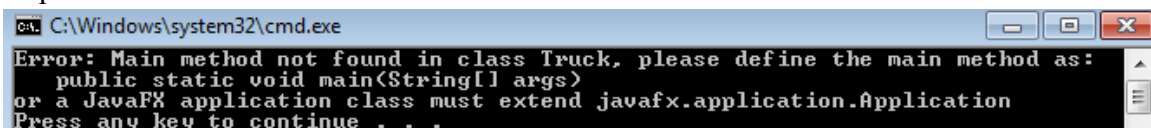
```
System.out.println("kecepatan maksimal:"+kecmaks);  
    System.out.println("Nama Kendaraan:"+namaken);
```

output error karena main method untuk pembuatan dan inisialisasi objek belum dibuat

PRAKTIK 2: Membuat Kelas Truck

```
public class Truck extends Mobil {  
    protected String speksifikasi;  
  
    public Truck(String speksifikasi, int kec) {  
        super(kec);  
        this.speksifikasi = speksifikasi;  
        System.out.println(speksifikasi);  
    }  
  
    public Truck(int kecmaks, String namaken, String speksifikasi){  
        super(kecmaks, namaken);  
        this.speksifikasi = speksifikasi;  
    }  
  
    @Override  
    public void PrintInfoMobil() {  
        super.PrintInfoMobil();  
        System.out.println("Gandengan:"+ speksifikasi);  
    }  
    public void PrintInfoTruck(){  
        System.out.println("kecepatan maksimal:"+kecmaks);  
        System.out.println("Nama Kendaraan:"+namaken);  
        System.out.println("Speksifikasi:"+ speksifikasi);  
    }  
}
```

output:



C:\Windows\system32\cmd.exe
Error: Main method not found in class Truck, please define the main method as:
 public static void main(String[] args)
or a JavaFX application class must extend javafx.application.Application
Press any key to continue . . .

penjelasan:

Kali ini kita membuat kelas Truck yang inherit dari kelas Mobil yaitu dengan menuliskan **public class Truck extends Mobil**. extends pada baris kode tersebut berfungsi untuk memanggil fungsi, method, dan atribut dari class parent (Mobil), sehingga kita tidak perlu lagi membuat script yang sama pada class yang akan kita buat dengan class yang kita buat sebelumnya. Di dalam class Truck, terdapat satu variabel/atribut yaitu speksifikasi yang dideklarasikan dengan menuliskan **protected String speksifikasi**. Modifier protected akan membuat member dan class hanya bisa diakses dari: class itu sendiri; sub class atau class anak; package (class yang berada satu package dengannya).

Kemudian kita membuat konstruktor yang memiliki dua parameter yaitu spesifikasi dan kec. Kec diakses dengan super yang berfungsi untuk digunakan untuk mengakses superclass. Lalu kita akses variabel speksifikasi dengan this.speksifikasi = speksifikasi;. Kemudian cetak output spesifikasi dengan **System.out.println(speksifikasi)**

Selanjutnya, kita buat konstruktor overloading yang digunakan untuk inisialisasi dan mengakses atribut yang ditulis dengan **public Truck(int kecmaks, String namaken, String spesifikasi){**. Di dalamnya ada **super(kecmaks, namaken)**. Kenapa untuk mengakses kecmaks dan namaken harus menggunakan super? Karena pada baris kode tersebut kita memanggil dan mengakses variabel milik parent class Mobil, sehingga dibutuhkan super untuk mengakses atribut pada parent class. Variabel spesifikasi diakses dengan this karena memang milik class Truck.

Kemudian kita membuat dan mendeklarasikan overriding method. Method Overriding dilakukan saat kita ingin membuat ulang sebuah method milik parent class pada sub-class atau child class. Method overriding dapat dibuat dengan menambahkan anotasi **@Override** di atas nama method atau sebelum pembuatan method. Pada kasus program ini, kita meng-override method **public void PrintInfoMobil()** dan di dalamnya kita mengakses method tersebut dengan super yaitu dengan menuliskan **super.PrintInfoMobil();**

Pada class Mobil, isi dari method PrintInfoMobil adalah:
System.out.println("kecepatan maksimal:"+kecmaks);
System.out.println("Nama Kendaraan:"+namaken);

Pada class Truck, method PrintInfoMobil sudah di-override dan isinya menjadi seperti berikut:
System.out.println("Gandengan:"+ spesifikasi);

Terakhir, kita buat method **public void PrintInfoTruck(){** yang di dalamnya berisi baris-baris kode untuk mencetak kalimat yaitu:

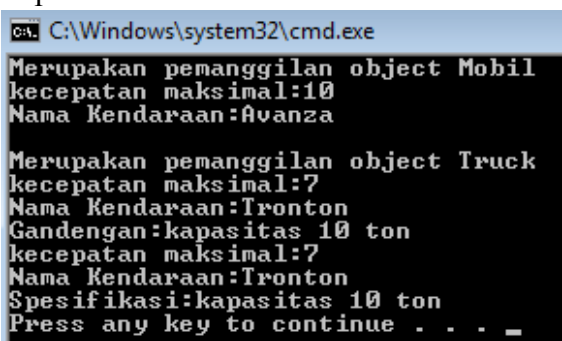
System.out.println("kecepatan maksimal:"+kecmaks);
System.out.println("Nama Kendaraan:"+namaken);
System.out.println("Spesifikasi:"+ spesifikasi);

output error karena main method untuk pembuatan dan inisialisasi objek belum dibuat

PRAKTIK 3: Membuat Kelas TestMobil

```
public class TestMobil{  
    public static void main(String args[]){  
        System.out.println("Merupakan pemanggilan object Mobil");  
        Mobil avanza = new Mobil(10, "Avanza");  
        avanza.PrintInfoMobil();  
        System.out.println("");  
  
        System.out.println("Merupakan pemanggilan object Truck");  
        Truck truk = new Truck(7,"Tronton","kapasitas 10 ton");  
        truk.PrintInfoMobil();  
        truk.PrintInfoTruck();  
    }  
}
```

output:



```
C:\Windows\system32\cmd.exe  
Merupakan pemanggilan object Mobil  
kecepatan maksimal:10  
Nama Kendaraan:Avanza  
  
Merupakan pemanggilan object Truck  
kecepatan maksimal:7  
Nama Kendaraan:Tronton  
Gandengan:kapasitas 10 ton  
kecepatan maksimal:7  
Nama Kendaraan:Tronton  
Spesifikasi:kapasitas 10 ton  
Press any key to continue . . . _
```

Script di atas adalah main method yang berfungsi untuk instansiasi objek (proses membuat objek) dan sekaligus mendeklarasikan dan mencetak output. Kita membuat class bernama TestMobil. Pertama kita tulis **public static void main(String args[]){** untuk mengawali eksekusi. Public berarti argumen atau metode main() merupakan sebuah metode yang bersifat publik, dengan kata lain metode main dapat dipanggil dimana saja. Void merupakan suatu tipe data yang menyatakan bahwa deklarasi code tidak memerlukan nilai balik atau return. Pada kasus ini metode main tidak memerlukan sebuah nilai balik. String merupakan kumpulan char atau sekumpulan character, dalam bahasa Indonesia string dapat diartikan sebagai kata atau kalimat sedangkan char merupakan character, character meliputi huruf, angka dan beberapa simbol. Dan args merupakan argumen bertipe data string yang mengandung array.

Pertama, kita melakukan instansiasi objek milik class Mobil yang diawali dengan mencetak sebuah kalimat dengan **System.out.println("Merupakan pemanggilan object Mobil");**. Di dalamnya kita buat dan deklarasikan objek bernama avanza sekaligus memberikan nilai pada objek avanza tersebut dengan menuliskan baris kode **Mobil avanza = new Mobil(10, "Avanza");**. Kemudian cetak output dengan cara memanggil output **avanza.PrintInfoMobil();**

Kedua, kita melakukan instansiasi objek milik class Truck yang diawali dengan mencetak sebuah kalimat dengan **System.out.println("Merupakan pemanggilan object Truck");**. Di dalamnya kita buat dan deklarasikan objek bernama truk sekaligus memberikan nilai pada objek truk tersebut dengan menuliskan baris kode **Truck truk = new Truck(7,"Tronton","kapasitas 10 ton");**. Lalu kita panggil method untuk mencetak output yaitu overriding method **truk.PrintInfoMobil();** dan method **truk.PrintInfoTruck();**. Bisa dilihat bahwa di output, bahwa method overriding PrintInfoMobil() milik class Truck mengeluarkan output baru yaitu Gandengan: kapasitas 10 ton.

PRAKTIK 4: Jelaskan tentang override dan adakah contoh implementasi dari praktik diatas

Method instan dalam subclass dengan nama dan parameter yang sama, serta return type sama dengan yang ada dalam superclass dikatakan meng-override (menimpa) method pada superclass. Kemampuan subclass meng-override method memungkinkan kelas mewarisi dari superclass yang mempunyai perilaku "cukup dekat" dan kemudian memodifikasi perilaku jika dibutuhkan. Method overriding dapat dibuat dengan menambahkan anotasi **@Override** di atas nama method atau sebelum pembuatan method. **@Override** yang menyuruh compiler bahwa anda bermaksud meng-override method dalam superclass.

Contoh implementasi dari praktik di atas dapat kita lihat pada class Truck yaitu saat kita meng-override method PrintInfoMobil untuk memodifikasi isinya. Hal itu kita lakukan dengan menulis:

```
-----
@Override
public void PrintInfoMobil() {
    super.PrintInfoMobil();
    System.out.println("Gandengan:"+ spesifikasi);
}
```

di dalamnya kita mengakses method tersebut dengan super karena kita "mengambil" method tersebut dari parent class Mobil. Lalu kita modif method tersebut dengan membuat baris kode untuk mencetak kalimat dengan **System.out.println("Gandengan:"+ spesifikasi);**

PRAKTIK 5: Jelaskan tentang penggunaan klausa super dan adakah contoh implementasi dari praktik diatas

Keyword super berfungsi sebagai variable referensi class, yang digunakan untuk rujukan dari super class atau parent class. Kita dapat menempatkan keyword super pada variable, method dan juga konstruktor. Dengan keyword super ini kita dapat membedakan fitur class dasar (super class) dengan Class Turunan (Sub Class). Contoh implementasi pada praktik di atas dapat kita lihat

di dalam class Truck pada saat kita membuat konstruktor untuk pengaksesan atribut/variabel yaitu:

```
public Truck(int kecmaks, String namaken, String speksifikasi){
    super(kecmaks, namaken);
    this.speksifikasi = speksifikasi;
}
```

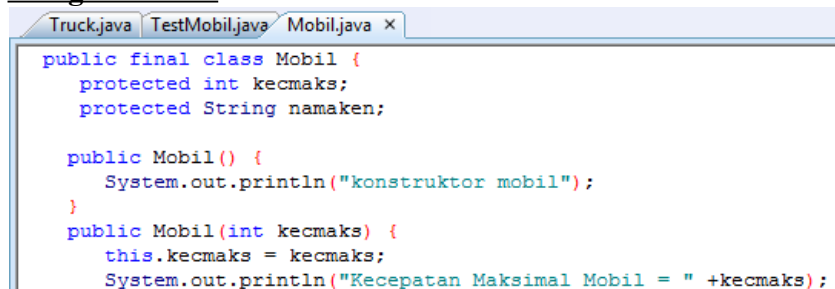
kita menggunakan super pada saat mengakses variabel kecmaks dan namaken karena variabel - variabel tersebut berasal dari parent class Mobil.

Contoh lainnya adalah baris kode berikut:

```
@Override
public void PrintInfoMobil() {
    super.PrintInfoMobil();
    System.out.println("Gandengan:" + speksifikasi);
}
```

overriding method tersebut berada di dalam class Truck. Pada saat kita memanggil method PrintInfoMobil, kita harus mengaksesnya dengan super karena method tersebut di-inherit dari parent class Mobil.

PRAKTIK 6: Modifikasi kelas Mobil dengan menambahkan klausa final pada deklarasi kelas sebagai berikut

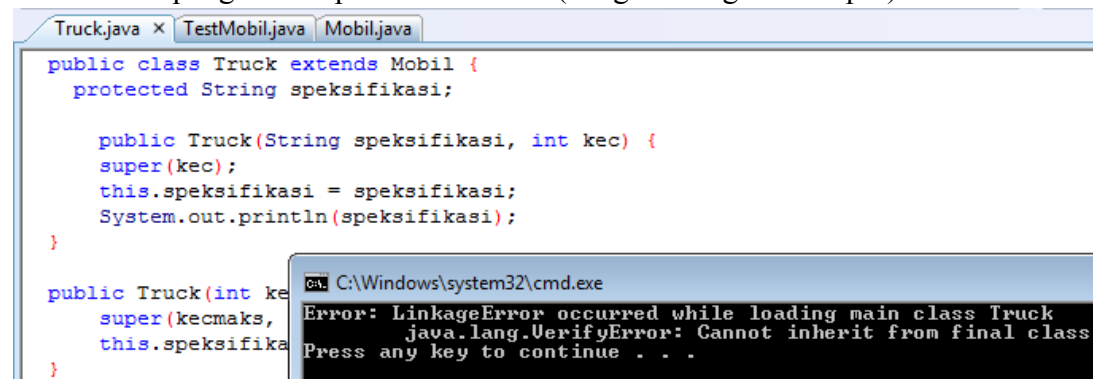


```
public final class Mobil {
    protected int kecmaks;
    protected String namaken;

    public Mobil() {
        System.out.println("konstruktor mobil");
    }

    public Mobil(int kecmaks) {
        this.kecmaks = kecmaks;
        System.out.println("Kecepatan Maksimal Mobil = " + kecmaks);
    }
}
```

lalu lakukan pengamatan pada class Truck (dengan mengecek output):



```
public class Truck extends Mobil {
    protected String speksifikasi;

    public Truck(String speksifikasi, int kec) {
        super(kec);
        this.speksifikasi = speksifikasi;
        System.out.println(speksifikasi);
    }

    public Truck(int kecmaks, String speksifikasi) {
        super(kecmaks, speksifikasi);
    }
}
```

```
C:\Windows\system32\cmd.exe
Error: LinkageError occurred while loading main class Truck
java.lang.VerifyError: Cannot inherit from final class
Press any key to continue . . .
```

penjelasan:

Error tersebut menyatakan bahwa class Truck tidak bisa inherit dari class yang memiliki klausa final. Mengapa hal itu bisa terjadi? Karena kata kunci final digunakan untuk pendeklarasian yg absolute (mutlak), jadi ketika suatu class dideklarasikan sebagai final maka class tersebut tidak bisa diwariskan. Sehingga class Truck tidak bisa mendapat warisan dari parent class Mobil.

PRAKTIK 7: Modifikasi kelas Mobil dengan menambahkan klausa final pada deklarasi metode printInfoMobil sebagai berikut

jawaban di halaman selanjutnya

```
Truck.java TestMobil.java Mobil.java x
public class Mobil {
    protected int kecmaks;
    protected String namaken;

    public Mobil() {
        System.out.println("konstruktor mobil");
    }
    public Mobil(int kecmaks) {
        this.kecmaks = kecmaks;
        System.out.println("Kecepatan Maksimal Mobil = " +kecmaks);
    }
    Mobil(int kecmaks, String namaken){
        this.kecmaks=kecmaks;
        this.namaken = namaken;
    }
    public final void PrintInfoMobil(){
        System.out.println("kecepatan maksimal:"+kecmaks);
        System.out.println("Nama Kendaraan:"+namaken);
    }
}
```

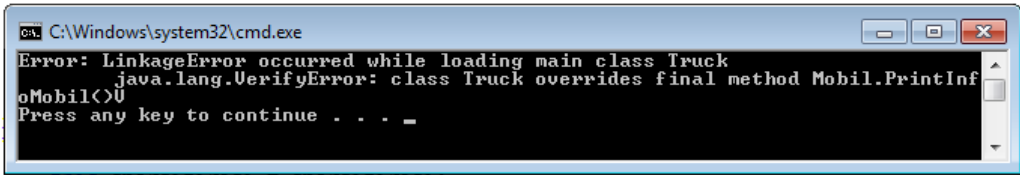
lalu lakukan pengamatan pada class Truck (dengan mengecek output):

```
Truck.java x TestMobil.java Mobil.java
public class Truck extends Mobil {
    protected String speksifikasi;

    public Truck(String speksifikasi, int kec) {

        this.speksifikasi = speksifikasi;
    }

    @Override
    public void PrintInfoMobil() {
        super.PrintInfoMobil();
        System.out.println("Gandengan:"+ speksifikasi);
    }
}
```



penjelasan:

Error tersebut menyatakan overriding method PrintInfoMobil() di dalam class Truck tidak bisa dilakukan. Mengapa hal itu terjadi? Seperti yang sudah saya katakan tadi, karena kata kunci final digunakan untuk pendeklarasian yg absolute (mutlak). Sehingga method PrintInfoMobil pada class Mobil yang sudah diubah menjadi final method tidak bisa diakses dan di-override oleh child class.

LATIHAN

1. Buatlah kelas baru yang merupakan turunan dari kelas Mobil praktik 1!

```
TestBus.java Bus.java x Mobil.java
1 public class Bus extends Mobil {
2     protected String speksifikasi;
3     protected String warna;
4     protected String merk;
5
6     public Bus(int kecmaks, String namaken, String speksifikasi,
7               String warna, String merk ){
8
9         super(kecmaks, namaken);
10        this.speksifikasi = speksifikasi;
11        this.warna = warna;
12        this.merk = merk;
13    }
14 }
```



```

13 }
14
15 @Override
16 public void PrintInfoMobil() {
17     super.PrintInfoMobil();
18     System.out.println("Spesifikasi:" + spesifikasi);
19     System.out.println("Warna:" + warna);
20     System.out.println("Merk:" + merk);
21 }
22 }

```

penjelasan:

Pada program di atas, kita membuat sebuah class bernama Bus yang mendapat warisan dari class Mobil. Hal itu diawali dengan menuliskan `public class Bus extends Mobil`. `extends` pada baris kode tersebut berfungsi untuk memanggil fungsi, method, dan atribut dari class parent (Mobil), sehingga kita tidak perlu lagi membuat script yang sama pada class yang akan kita buat dengan class yang kita buat sebelumnya. Di dalam class Bus, terdapat beberapa variabel/atribut baru yaitu spesifikasi, warna, merk. Semuanya memiliki tipe data string dan memiliki modifier `protected`. Modifier `protected` akan membuat member dan class hanya bisa diakses dari: class itu sendiri; sub class atau class anak; package (class yang berada satu package dengannya).

Kemudian kita membuat konstruktor yang memiliki parameter atribut-atribut dari parent class Mobil dan dari class Bus itu sendiri. Berikut pendeklarasiannya:

```
public Bus(int kecmaks, String namaken, String spesifikasi, String warna, String merk)
```

di dalamnya kita lakukan pengaksesan atribut/variabel. Jika dari parent class, gunakan `super`. Jika dari sub class, gunakan `this`. Berikut penulisannya:

```
super(kecmaks, namaken);
this.spesifikasi = spesifikasi;
this.warna = warna;
this.merk = merk;
```

Kemudian kita membuat dan mendeklarasikan overriding method. Method Overriding dilakukan saat kita ingin membuat ulang sebuah method milik parent class pada sub-class atau child class. Method overriding dapat dibuat dengan menambahkan anotasi `@Override` di atas nama method atau sebelum pembuatan method. Pada kasus program ini, kita meng-override method **`public void PrintInfoMobil()`** dan di dalamnya kita mengakses method tersebut dengan `super` yaitu dengan menuliskan **`super.PrintInfoMobil();`** untuk mendapat warisan dari method tersebut, kemudian tinggal kita modif/tambahkan seperti baris kode berikut:

```
@Override
public void PrintInfoMobil() {
    super.PrintInfoMobil();
    System.out.println("Spesifikasi:" + spesifikasi);
    System.out.println("Warna:" + warna);
    System.out.println("Merk:" + merk);
}
```

kita tidak perlu mencetak kalimat untuk atribut kecmaks dan namaken kembali karena kita sudah memanggilnya dengan `super.PrintInfoMobil()`. Jadi, tinggal kita tambahkan saja dengan pencetakan kalimat spesifikasi, warna, dan merk.

2. Buatlah contoh pembuatan objek dan penggunaan membernya!


```
TestBus.java × Bus.java Mobil.java
public class TestBus{
    public static void main(String args[]){

        System.out.println("Merupakan pemanggilan object Bus");
        Bus bis = new Bus(7,"Bus Kota","40 penumpang","Putih","Mercedes-Benz");
        bis.PrintInfoMobil();
    }
}
```

Output:

```
C:\Windows\system32\cmd.exe
Merupakan pemanggilan object Bus
kecepatan maksimal:7
Nama Kendaraan:Bus Kota
Spesifikasi:40 penumpang
Warna:Putih
Merk:Mercedes-Benz
Press any key to continue . . .
```

penjelasan:

script di atas adalah sebuah main method untuk instansiasi objek (pembuatan objek). Pertama, kita buat nama class nya dengan menulis **public class TestBus**. Lalu eksekusinya kita awali dengan **public static void main(String args[]){**. Sebelum membuat objek, kita awali dengan mencetak kalimat **System.out.println("Merupakan pemanggilan object Bus");**. Kemudian kita buat objek dengan nama bis sekaligus berikan argumen/nilai pada setiap variabel atau atributnya yang ditulis pada baris kode berikut:

Bus bis = new Bus(7,"Bus Kota","40 penumpang","Putih","Mercedes-Benz");

Terakhir, kita cetak outputnya dengan **bis.PrintInfoMobil();**

KESIMPULAN

Pada pertemuan kali ini, kita mempelajari tentang inheritance (pewarisan), sama seperti dengan pertemuan ke-7, hanya saja pada pertemuan kali ini lebih difokuskan pada 2 keyword penting yaitu final dan super, ditambah dengan method overriding. Final digunakan untuk pendeklarasian yg absolute (mutlak), jadi ketika suatu class dideklarasikan sebagai final maka class tersebut tidak bisa diwariskan. Super berfungsi sebagai variable referensi class , yang digunakan untuk rujukan dari super class atau parent class. Method overriding adalah ketika super class memiliki method dan kemudian method ini dideklarasikan di sub class untuk kemudian ditambahkan atau dimodif.

Terima Kasih.