

PERTEMUAN 10

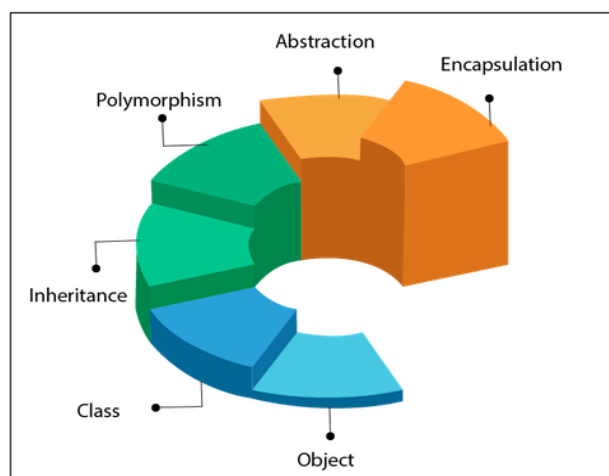
KONSEP PEMROGRAMAN BERORIENTASI OBJEK *CLASS, OBJECT, ATTRIBUTE, METHOD*

A. TUJUAN PEMBELAJARAN

1. Mahasiswa memahami Konsep Pemrograman berorientasi Objek
2. Mahasiswa memahami apa itu object , *class* pada Pemrograman berorientasi Objek
3. Mahasiswa memahami apa itu *attribute, method* pada Pemrograman berorientasi Objek
4. Mahasiswa dapat mempraktekan pembuatan class,object , attribute dan method pada Pemrograman berorientasi Objek

B. URAIAN MATERI

Pemrograman berorientasi Objek atau Object-oriented programming System (OOPs) adalah paradigma pemrograman berdasarkan konsep "objek" yang berisi atribut atau data dan metode. Tujuan utama dari pemrograman berorientasi objek adalah untuk meningkatkan fleksibilitas dan pemeliharaan program. Pemrograman berorientasi objek menyatukan data dan perilakunya (metode) dalam satu lokasi (objek) sehingga pembuatannya lebih mudah untuk dipahami bagaimana sebuah program bekerja.



Gambar 10. 1 Paradigma pemrograman berorientasi objek

1. **Object**

Setiap entitas yang memiliki state/atribut dan behavior dikenal sebagai objek. Misalnya, kursi, pena, meja, keyboard, sepeda, dll. Objek bisa berupa fisik atau logikal.

Objek dapat didefinisikan sebagai turunan (instance of) dari kelas. Sebuah objek berisi alamat dan mengambil beberapa ruang di memori. Objek dapat berkomunikasi tanpa mengetahui detail data atau kode satu sama lain.

Contoh: motor adalah *object* karena memiliki state atau atribut seperti warna, tahun pembuatan, jenis, dll. serta perilaku/methode seperti cara hidupkan, cara menambahkan kecepatan, menyalakan lampu depan dll.



Gambar 10. 2 ilustrasi sebuah object Konsep OOP

Object Motor

Attribute :

Jenis, Warna, tahun produksi, no seri, no mesin

Behaviour :

Cara menghidupkan, cara menjalankan, cara berbelok, cara akselerasi.

Attribute → Variable

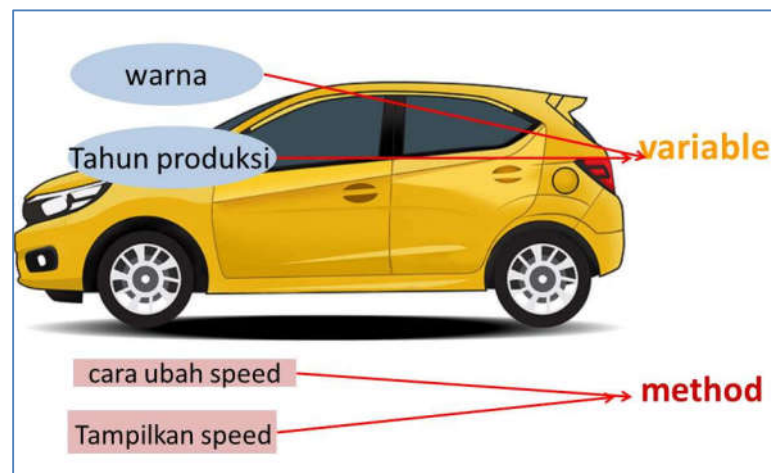
Behaviour → method/ fungsi

2. Class

Class adalah Kumpulan objek, *class* jika di instansiasi menjadi objek *Class* juga dapat didefinisikan sebagai cetak biru dari objek individual, dimana hasil cetakan itu menjadi *object*. *Class* merupakan kesatuan *Method* (Behavior) dan *variable* (*Attribute*)

Contoh ilustrai *Class* :

class mobil

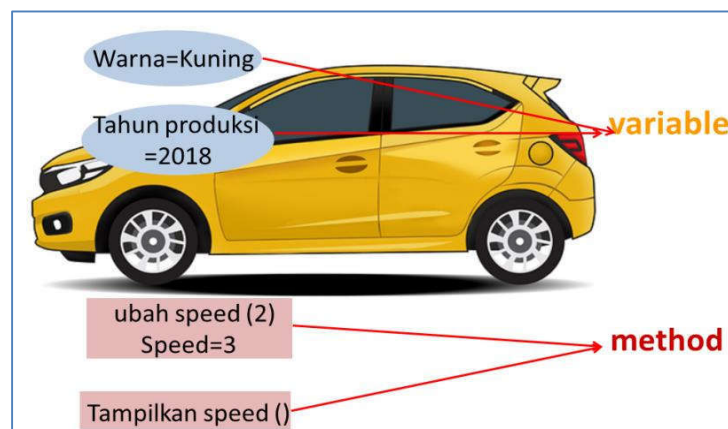


Gambar 10. 3 ilustrasi sebuah class Konsep OOP

Sementara *object* adalah *Method* (Behavior) dan *variable* (*Attribute*) yang diberikan nilai.

Contoh ilustrasi *object* :

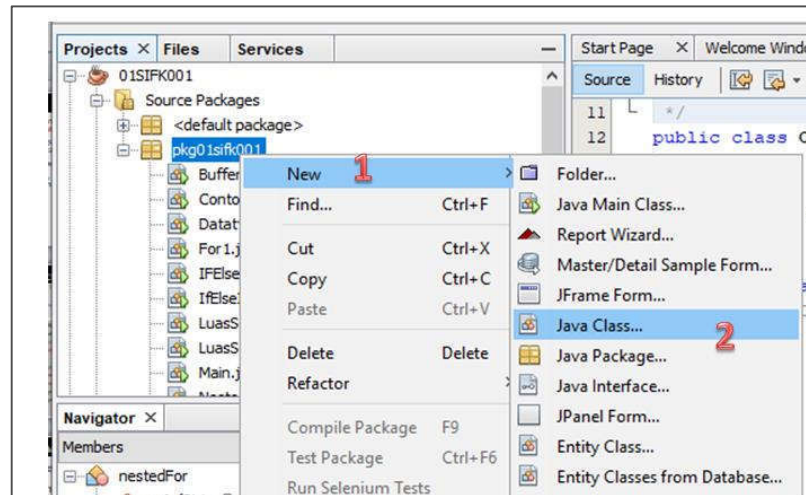
class mobilku :



Gambar 10. 4 ilustrasi sebuah object dengan Value

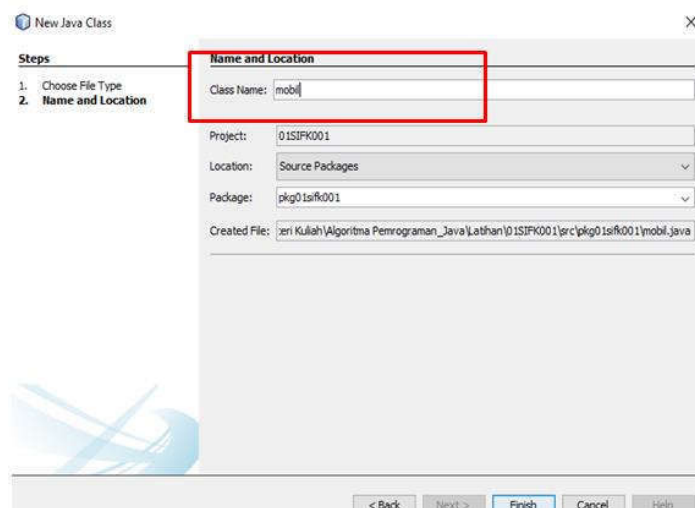
3. Membuat class , object dan attribute pada program java

- a. Buat class *mobil* pada program java langkahnya klik menu file new [1], pilih java class[2]



Gambar 10. 5 membuat class mobil

- b. Beri nama class pada kotak isian penamaan dan lokasi penyimpanan contoh nama class : mobil



Gambar 10. 6 memberi penamaan class mobil

- c. Buat attributnya (Variable) Jenis, warna, tahun produksi pada class mobil

```
-  ~/  
    public class mobil {  
        String jenis;  
        String warna;  
        int tahunproduksi;  
    }
```

- d. Buat program utama namakan OopMobil untuk membuat objek dan instansiasi dari *class*

Contoh Nama objek =mymobil()

Contoh pemberian nilai attribute jenis="SUV" ", warna="Kuning", tahun produksi=2010

```
public class OopMobil {  
    /**  
     * @param args the command line arguments  
     */  
    public static void main(String[] args) {  
        // TODO code application logic here  
        mobil mymobil=new mobil(); // membuat objek mymobil dari class mobil  
        mymobil.jenis="SUV";  
        mymobil.warna="Kuning";  
        mymobil.tahunproduksi=2010;  
        System.out.println("Jenis Mobil "+mymobil.jenis);  
        System.out.println("Warna Mobil "+mymobil.warna);  
        System.out.println("Tahun Produksi Mobil "+mymobil.tahunproduksi);  
    }  
}
```

Output :

```
Jenis Mobil SUV  
Warna Mobil Kuning  
Tahun Produksi Mobil 2010
```

4. Membuat class, object dan method nya pada program java

Method adalah suatu fungsi atau urutan instruksi yang mengakses data dari objek. Pada penulisan method biasanya di akhiri () , method melakukan manipulasi data, perhitungan matematika dan memonitor kejadian dari suatu event.

Contoh membuat method pada class mobil sebelumnya, kita akan membuat method tampilkan nilai nilai dari atribut.

- a. Modifikasi class mobil sebelumnya dengan menambah method printdatamobil()



```

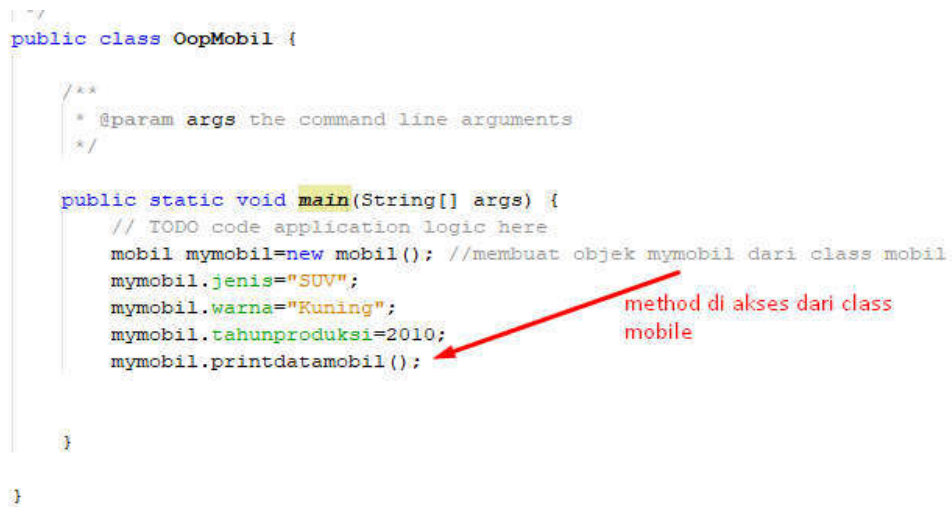
/**
 *
 */
public class mobil {
    String jenis;
    String warna;
    int tahunproduksi;

    void printdatamobil(){
        System.out.println("Jenis: " + jenis);
        System.out.println("Warna: " + warna);
        System.out.println("Tahun: " + tahunproduksi);
    }
}

```

A red arrow points from the text "method printdatamobil()" to the method definition in the code.

- b. Selanjutnya akses method tersebut di program utama, seperti gambar dibawah ini



```

/**
 *
 */
public class OopMobil {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        mobil mymobil=new mobil(); //membuat objek mymobil dari class mobil
        mymobil.jenis="SUV";
        mymobil.warna="Kuning";
        mymobil.tahunproduksi=2010;
        mymobil.printdatamobil();
    }
}

```

A red arrow points from the text "method di akses dari class mobile" to the call to `mymobil.printdatamobil();` in the code.

Output :

```
Jenis: SUV
Warna: Kuning
Tahun: 2010
```

5. jenis method accesor dan mutator

a. method *accesor*

Metode ***accesor*** digunakan untuk mengembalikan nilai dari *private field*. skema penamaan diawali dengan kata "get" ke awal nama metode.

b. method *mutator*

Metode ***mutator*** digunakan untuk menetapkan nilai *private field*. skema penamaan awalan diawali dengan kata "set"

Contoh penulisan method accesor dan mutator :

```
public class aritmatika{
    int angka=10;

    // ini method (mutator) dengan parameter
    void setAngka(int pertambahan) {
        angka= angka+ pertambahan;
    }

    // ini method (accessor)
    int getAngka() {
        return angka;
    }
}
```

Contoh membuat program pertambahan nya dengan nama program `artimatikaMain` :

```
public class artimatikaMain {
```

```
/**
 * @param args the command line arguments
 */
public static void main(String[] args) {
    // TODO code application logic here
    aritmatika aritmatikaku=new aritmatika();
    aritmatikaku.setTambah(20);

    System.out.println("Hasil Pertambahan
"+aritmatikaku.getTambah());
}
}
```

Output

```
run:
Hasil Pertambahan 20
BUILD SUCCESSFUL (total time: 0 seconds)
```

Contoh membuat method dengan 2 paramater

a. Buat class aritmatika2 dengan *attribute* dan *method* nya

```
public class aritmatika {

    int angka;
    // ini method (mutator) dengan 2 parameter bil1 dan
    bil2
    void setTambah(int bil1, int bil2) {
        angka= bil1+bil2;
    }
    // ini method (accessor)
    int getTambah() {
        return angka;
    }
}
```



```
}
```

- b. Buat program class aritmatika_Utama2 untuk membuat objek dan mengakses class aritmatika

```
public class aritmatikaMain2 {  
  
    /**  
     * @param args the command line arguments  
     */  
    public static void main(String[] args) {  
        // TODO code application logic here  
  
        aritmatika2 aritmatikaku=new aritmatika2();  
        int a=10,b=20;  
        aritmatikaku.setTambah(a,b);  
        System.out.println("Bilangan 1= " +a);  
        System.out.println("Bilangan 2= " +b);  
        System.out.println("Hasil Pertambahan  
"+aritmatikaku.getTambah());  
  
    }  
}
```

Output :

```
run:  
Bilangan 1= 10  
Bilangan 2= 20  
Hasil Pertambahan 30  
BUILD SUCCESSFUL (total time: 0 seconds)
```

C. LATIHAN / TUGAS

1. Buat Class aritmatika3 modifikasi dari class aritmatika2, yang berisi method dengan dua parameter tambahkan method :
 - a. pengurangan(int a, int b)
 - b. perkalian(int a, int b)
 - c. pembagian(int a, int b)
 - d. pangkat(int a, int b)
2. Buat Class aritmatika3Main untuk membuat object serta meng akses method pada class aritmatika 1 nilai nilai objeknya sebagai berikut ;
 - a. Pengurangan: bilangan1 – bilangan2
 - b. Perkalian: bilangan1*bilangan2
 - c. Pembagian: bilangan1/bilangan2
 - d. Pangkat: bilangan1^bilangan2

D. REFERENSI

- Horstmann Cay S., (2011). *Big Java 4th Edition* ,san jose university , united state Of America. RRD jefferson city publishing.
- Deitel Paul , Deitel Harvey, (2012) Java how to program eighth edition, pearson education, Boston Massachusetts , USA, *publishing as prentice hall*.
- Rose Cristhoper, (2017), Java Succinctly Part 2, Morrisville, NC 27560, USA, Syncfusion, Inc.
- Downey Allen B. , Mayfield Chris, (2017), Think Java, Needham, Massachusetts, USA, Green Tea Press
- Hayes Helen, (2021), BeginnersBook.com, <https://beginnersbook.com/java-tutorial-for-beginners-with-examples/>, di akses pada tanggal 21 November 2021.
- Sonoo Jaiswal , (2021) JavaTpoint offers college campus training , <https://www.javatpoint.com/java-tutorial>, diakses pada tanggal 1 Desember 2021