



# MODUL PRAKTIKUM

## PEMROGRAMAN BERORIENTASI OBYEK

**Prodi Sistem Informasi  
Fakultas Sains dan Teknologi  
UIN Sunan Ampel Surabaya  
Tahun 2021**

Dwi Rolliawati, MT  
*dwi\_roll@uinsby.ac.id*

# 1

## BAB 1

### KONSEP PEMROGRAMAN BERORIENTASI OBYEK

#### TUJUAN

1. Mahasiswa mampu menjelaskan konsep dasar pemrograman OO
2. Mahasiswa mampu menjelaskan perbedaan pemrograman OO dan Terstruktur

#### INDIKATOR

Ketepatan dalam menjelaskan perbedaan mendasar OOP structured programming dan unstructured programming

#### DASAR TEORI

##### REFERENSI :

[HTTPS://WWW.GURU99.COM/JAVA-OOPS-CONCEPT.HTML](https://www.guru99.com/java-oops-concept.html)

[https://www.w3schools.com/java/java\\_oop.asp](https://www.w3schools.com/java/java_oop.asp)

Object-Oriented Programming System (OOPs) adalah konsep pemrograman yang bekerja berdasarkan prinsip-prinsip abstraksi, enkapsulasi, pewarisan, dan polimorfisme. Ini memungkinkan pengguna untuk membuat objek yang diinginkan dan membuat metode untuk menangani objek tersebut. Konsep dasar OOP adalah membuat objek, menggunakannya kembali di seluruh program, dan memanipulasi objek ini untuk mendapatkan hasil.

# 1

## BAB 1

### KONSEP PEMROGRAMAN BERORIENTASI OBYEK

#### LATIHAN

1. Perhatikan ketiga potongan source code studi kasus aplikasi bank sederhana dibawah ini dengan seksama dan jelaskan apa perbedaan mendasar dari ketiga paradigma pemrograman dibawah ini.

```
int account_number = 20;
int account_balance = 100;
account_balance = account_balance + 100;
printf("Account Number = %d", account_number);
printf("Account Balance = %d", account_balance);
account_balance = account_balance - 50;
printf("Account Number = %d", account_number);
printf("Account Balance = %d", account_balance);
account_balance = account_balance - 10;
printf("Account Number = %d", account_number);
printf("Account Balance = %d", account_balance);
```

unstructured programming  
same code is repeated

```
void showData() {
    printf("Account Number = %d", account_number);
    printf("Account Balance = %d", account_balance);
}

int account_number = 20;
int account_balance = 100;
account_balance = account_balance + 100;
printf("Account Number = %d", account_number);
printf("Account Balance = %d", account_balance);
account_balance = account_balance - 50;
showData();
```

structured programming

common code put into function

call being made to the function

```
int account_number = 20;
int account_balance = 100;
account_balance = account_balance + 100;
showData();
account_balance = account_balance - 50;
showData();
```

DATA

ACTIONS

2. Lengkapi potongan source code dari gambar ketiga (disamping) sehingga menghasilkan output. Tulis source code dan hasil capture output program tersebut.

3. Apa saja konsep dasar OOP/PBO silakan dijelaskan

# 2

## BAB 2

### CLASS DAN OBJECT

## TUJUAN

1. Memahami konsep class & object.
2. Dapat mengimplementasikan konsep class & object pada kasus sederhana.

## INDIKATOR

Ketepatan dalam menerapkan konsep class & object

## DASAR TEORI

### REFERENSI :

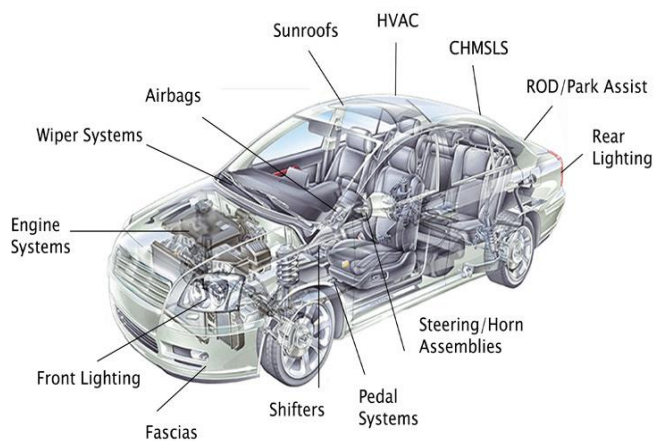
<https://www.geeksforgeeks.org/classes-objects-java/?ref=lbp>

[https://www.w3schools.com/java/java\\_classes.asp](https://www.w3schools.com/java/java_classes.asp)

<https://www.guru99.com/java-oops-class-objects.html>

<https://codebridgeplus.com/access-modifiers-in-java/>

Kelas dan objek adalah dua aspek utama dari pemrograman berorientasi objek. Kelas dan objek adalah gambaran dari entity, baik dunia nyata atau konsep dengan batasan-batasan dan pengertian yang tepat. Objek-objek ini kemudian juga dapat berupa gabungan dari beberapa objek yang lebih kecil. Sebagai contoh, lihatlah sebuah mobil. Mobil adalah sebuah objek dalam kehidupan nyata. Namun mobil sendiri merupakan gabungan beberapa objek yang lebih kecil seperti roda ban, mesin, jok, dan lainnya. Mobil sebagai objek yang merupakan gabungan dari objek yang lebih kecil dibentuk dengan membentuk hubungan antara objek-objek penyusunnya.



**Kelas** memiliki variabel yang disebut sebagai attribute dan subroutine (*a set of instructions designed to perform a frequently used operation*) yang biasa disebut *method*. Dalam sudut pandang pemrograman, kelas digunakan untuk menciptakan suatu obyek. Atau dengan kata lain, kelas merupakan pembuat objek. Pada *class* terdapat suatu **access modifier**. Hal ini berguna untuk menentukan tipe hak akses bagi sebuah *attribute* dan *method*.

Access Modifiers in java				
	public	private	protected	< unspecified >
class	allowed	not allowed	not allowed	allowed
constructor	allowed	allowed	allowed	allowed
variable	allowed	allowed	allowed	allowed
method	allowed	allowed	allowed	allowed

	class	subclass	package	outside
private	allowed	not allowed	not allowed	not allowed
protected	allowed	allowed	allowed	not allowed
public	allowed	allowed	allowed	allowed
< unspecified >	allowed	not allowed	allowed	not allowed

(Source: <https://codebridgeplus.com/access-modifiers-in-java/>)

**Method** dikenal juga sebagai suatu function dan procedure. Dalam OOP, method digunakan untuk memodularisasi program melalui pemisahan tugas dalam suatu class. Pemanggilan method menspesifikasikan nama method dan menyediakan informasi (parameter) yang diperlukan untuk melaksanakan tugasnya.

Di dalam Java terdapat suatu besaran referensi khusus yang disebut **keyword this**, yang digunakan di dalam method yang dirujuk untuk objek yang sedang berlaku. Nilai **this** merujuk pada objek di mana method yang sedang berjalan dipanggil.

Perbedaan Class dan Object:

Class	Object
A class is a template for creating objects in program.	The object is an instance of a class.
A class is a logical entity	Object is a physical entity
A class does not allocate memory space when it is created.	Object allocates memory space whenever they are created.
You can declare class only once.	You can create more than one object using a class.
Example: Car.	Example: Jaguar, BMW, Tesla, etc.
Class generates objects	Objects provide life to the class.
Classes can't be manipulated as they are not available in memory.	They can be manipulated.
It doesn't have any values which are associated with the fields.	Each and every object has its own values, which are associated with the fields.
You can create class using "class" keyword.	You can create object using "new" keyword in Java

**Constructor** atau **konstruktor** digunakan untuk melakukan inisialisasi variable-variabel instan class serta melakukan persiapan-persiapan yang diperlukan oleh suatu objek untuk dapat beroperasi dengan baik. Format umum pendeklarasian dan pendefinisian constructor adalah :

- Nama constructor sama dengan nama class.
- Sebelum itu dapat diberi access modifier untuk mengatur visibility constructor.

# 2

## BAB 2 CLASS DAN OBJECT

```
public class MyClass{  
    // Constructor  
    MyClass(){  
        System.out.println("BeginnersBook.com");  
    }  
    public static void main(String args[]){  
        MyClass obj = new MyClass();  
        ...  
    }  
}
```

New keyword creates the object of MyClass & invokes the constructor to initialize the created object.

```
public class Hello {  
    String name;  
    //Constructor  
    Hello(){  
        this.name = "BeginnersBook.com";  
    }  
    public static void main(String[] args) {  
        Hello obj = new Hello();  
        System.out.println(obj.name);  
    }  
}
```

(Source: <https://beginnersbook.com/2013/03/constructors-in-java/>)

Dalam suatu Class dapat lebih dari satu constructor, masing-masing harus mempunyai parameter yang berbeda sebagai penandanya. Hal seperti ini disebut **Overloading Constructor**.

```
public class Demo {  
    Demo(){  
        ...  
    }  
    Demo(String s){  
        ...  
    }  
    Demo(int i){  
        ...  
    }  
    ....  
}
```

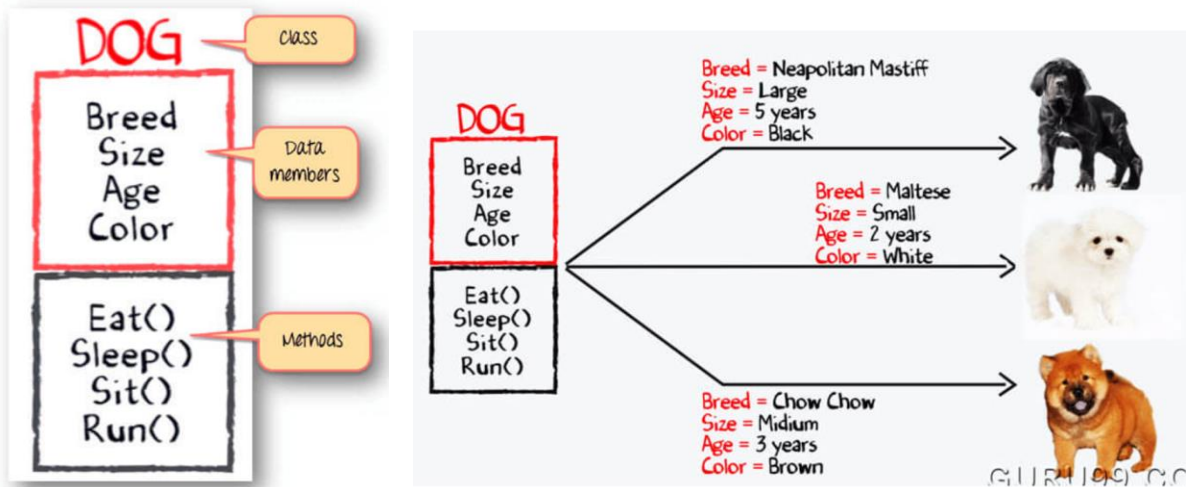
Three overloaded constructors - They must have different Parameters list

# 2

## BAB 2 CLASS DAN OBJECT

### LATIHAN

1. Buatlah baris program dari ilustrasi gambar dibawah ini lengkap dengan konstruktornya.



2. Uji cobakan 3 model source code dibawah ini. Deskripsikan hasil analisa anda setelah anda melakukan uji coba hasil programnya

```
class Person {
    String fname = "John";
    String lname = "Doe";
    String email = "john@doe.com";
    int age = 24;

    public static void main(String[] args) {
        Person myObj = new Person();
        System.out.println("Name: " + myObj.fname + " " + myObj.lname);
        System.out.println("Email: " + myObj.email);
        System.out.println("Age: " + myObj.age);
    }
}
```

```
public class Main {
    public String fname = "John";
    public String lname = "Doe";
    public String email = "john@doe.com";
    public int age = 24;
}
```

```
class Person {
    protected String fname = "John";
    protected String lname = "Doe";
    protected String email = "john@doe.com";
    protected int age = 24;
}

class Student extends Person {
    private int graduationYear = 2018;
    public static void main(String[] args) {
        Student myObj = new Student();
        System.out.println("Name: " + myObj.fname + " " + myObj.lname);
        System.out.println("Email: " + myObj.email);
        System.out.println("Age: " + myObj.age);
        System.out.println("Graduation Year: " + myObj.graduationYear);
    }
}
```

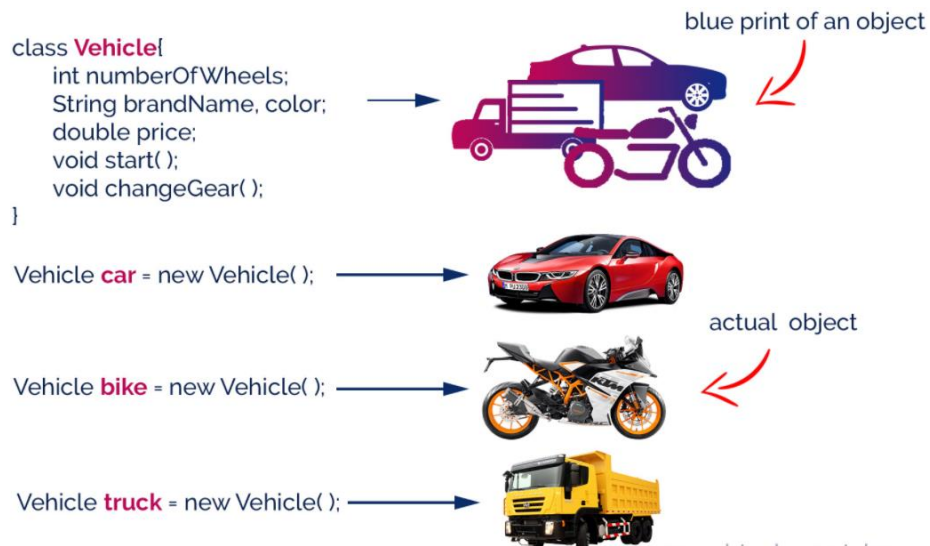
```
class Second {
    public static void main(String[] args) {
        Main myObj = new Main();
        System.out.println("Name: " + myObj.fname + " " + myObj.lname);
        System.out.println("Email: " + myObj.email);
        System.out.println("Age: " + myObj.age);
    }
}
```



# 2

## BAB 2 CLASS DAN OBJECT

3. Selesaikan ilustrasi dibawah ini dalam bentuk source code. Tulis source code anda dan capture hasil keluarannya.





## TUJUAN

1. Mahasiswa mampu mengidentifikasi pola enkapsulasi dan menerapkannya
2. Mahasiswa mampu membuat program dengan menggunakan konsep enkapsulasi

## INDIKATOR

1. Ketepatan identifikasi dan penerapan enkapsulasi dan abstraksi
2. Ketepatan program atau code

## DASAR TEORI

### REFERENSI :

[https://www.w3schools.com/java/java\\_encapsulation.asp](https://www.w3schools.com/java/java_encapsulation.asp)

**Enkapsulasi** merupakan proses pemaketan objek beserta methodnya untuk menyembunyikan rincian implementasi dari pemakai/objek lainnya. Inti dari enkapsulasi atau pengkapsulan adalah ketidaktahuan apa yang ada dalam suatu objek dan bagaimana pengimplementasiannya. Yang dibutuhkan hanyalah apa kegunaan, bagaimana cara memakainya dan apa yang akan terjadi.

**Arti Enkapsulasi**, adalah untuk memastikan bahwa data "sensitif" disembunyikan dari pengguna. Untuk mencapai ini, Kita harus: mendeklarasikan variabel/atribut kelas sebagai *private* dan menyediakan metode *get* dan *set* *public* untuk mengakses dan memperbarui nilai variabel *private*

### Mengapa Enkapsulasi?

1. Kontrol yang lebih baik dari atribut dan metode kelas
2. Atribut class dapat dibuat read-only (jika hanya menggunakan metode *get*), atau write-only (jika hanya menggunakan metode *set*)
3. Fleksibel: programmer dapat mengubah satu bagian dari kode tanpa mempengaruhi bagian lain
4. Peningkatan keamanan data

## LATIHAN

1. Uji cobakan source code berikut, terangkan bagaimana analisa anda dan bagaimana hasil programnya supaya memiliki keluaran tanpa error?

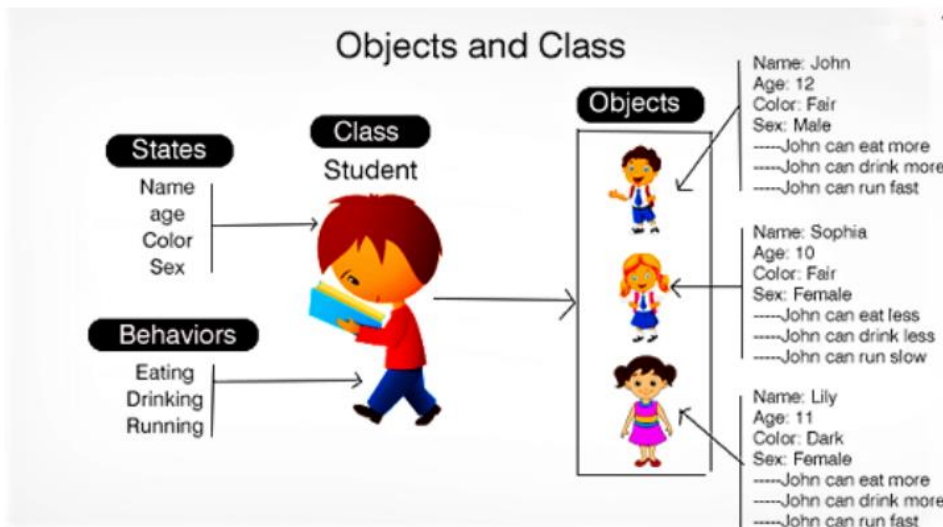
```
public class Person {
    private String name; // private = restricted access

    // Getter
    public String getName() {
        return name;
    }

    // Setter
    public void setName(String newName) {
        this.name = newName;
    }
}
```

```
public class Main {
    public static void main(String[] args) {
        Person myObj = new Person();
        myObj.name = "John"; // error
        System.out.println(myObj.name); // error
    }
}
```

2. Tuislah source code dan hasil keluaran dari ilustrasi berikut dengan mengimplementasikan konsep enkapsulasi



3. Annisa adalah salah satu mahasiswa penggemar aplikasi Go MakYam. Selama pandemi Covid 19 dia seringkali dapat promo atas pembeliannya. Go MakYam memiliki program promo : Diskon 50% sd 35k untuk pembelian diatas 200k. Diskon ongkir sd 7k untuk pembelian diatas 200k. Buatlah program dari ilustrasi kasus diatas dengan menerapkan enkapsulasi

# 4

## BAB 4 INHERITANCE

### TUJUAN

1. Mahasiswa mampu mengidentifikasi pola inheritance dan menerapkannya
2. Mahasiswa mampu menggunakan constructor dan destructor dalam program

### INDIKATOR

1. Ketepatan identifikasi dan penerapan inheritance
2. Ketepatan program atau code

### DASAR TEORI

#### REFERENSI :

[https://www.w3schools.com/java/java\\_encapsulation.asp](https://www.w3schools.com/java/java_encapsulation.asp)

Di Java, dimungkinkan untuk mewarisi atribut dan metode dari satu kelas ke kelas lainnya. Kami mengelompokkan "konsep pewarisan" ke dalam dua kategori:

subclass (anak) - kelas yang mewarisi dari kelas lain

superclass (induk) - kelas yang diwarisi dari

Untuk mewarisi dari kelas, gunakan kata kunci **extends**. Pewarisan adalah konsep pemrograman berorientasi objek yang sangat berguna dan kuat. Di java, dengan menggunakan konsep pewarisan, kita bisa menggunakan fitur-fitur yang ada dari satu kelas di kelas lain. Warisan memberikan keuntungan besar yang disebut penggunaan kembali kode. Dengan bantuan penggunaan kembali kode, kode yang umum digunakan dalam suatu aplikasi tidak perlu ditulis berulang kali.

Kata kunci super mirip dengan kata kunci this. Berikut ini adalah skenario di mana kata kunci super digunakan.

1. Digunakan untuk membedakan anggota superclass dari anggota subclass, jika mereka memiliki nama yang sama.
2. Ini digunakan untuk memanggil konstruktor superclass dari subclass

```
super.variable  
super.method();
```

#### Memanggil Konstruktor Superclass

Jika sebuah kelas mewarisi properti dari kelas lain, subkelas secara otomatis memperoleh konstruktor default dari superclass. Tetapi jika Anda ingin memanggil konstruktor berparameter dari superclass, Anda perlu menggunakan kata kunci super seperti yang ditunjukkan di bawah ini.

# 4

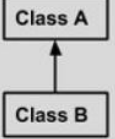
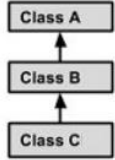
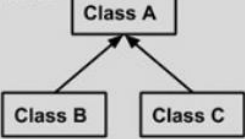
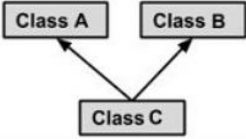
## BAB 4 INHERITANCE

```
super(values);
```

Contoh implementasi :

```
1 - class Superclass {
2     int age;
3
4     Superclass(int age) {
5         this.age = age;
6     }
7
8     public void getAge() {
9         System.out.println("The value of the variable named age in super class is: " +age);
10    }
11 }
12
13 - public class Subclass extends Superclass {
14     Subclass(int age) {
15         super(age);
16     }
17
18     public static void main(String argd[]) {
19         Subclass s = new Subclass(24);
20         s.getAge();
21     }
22 }
```

Beberapa tipe inheritance :

<b>Single Inheritance</b>  <pre> graph BT     B[Class B] --&gt; A[Class A]         </pre>	<pre> public class A {     ..... } public class B extends A {     ..... }         </pre>
<b>Multi Level Inheritance</b>  <pre> graph BT     C[Class C] --&gt; B[Class B]     B --&gt; A[Class A]         </pre>	<pre> public class A { .....} public class B extends A {.....} public class C extends B {.....}         </pre>
<b>Hierarchical Inheritance</b>  <pre> graph BT     B[Class B] --&gt; A[Class A]     C[Class C] --&gt; A         </pre>	<pre> public class A { .....} public class B extends A {.....} public class C extends A {.....}         </pre>
<b>Multiple Inheritance</b>  <pre> graph BT     A[Class A] --&gt; C[Class C]     B[Class B] --&gt; C         </pre>	<pre> public class A { .....} public class B {.....} public class C extends A,B {     ..... } // Java does not support mutiple Inheritance         </pre>

## LATIHAN

1. Ujicobakan source berikut dan tulis hasil serta analisa anda

```
class Super_class {
    int num = 20;

    // display method of superclass
    public void display() {
        System.out.println("This is the display method of superclass");
    }
}

public class Sub_class extends Super_class {
    int num = 10;

    // display method of sub class
    public void display() {
        System.out.println("This is the display method of subclass");
    }

    public void my_method() {
        // Instantiating subclass
        Sub_class sub = new Sub_class();

        // Invoking the display() method of sub class
        sub.display();

        // Invoking the display() method of superclass
        super.display();

        // printing the value of variable num of subclass
        System.out.println("value of the variable named num in sub class:"+ sub.num);

        // printing the value of variable num of superclass
        System.out.println("value of the variable named num in super class:"+ super.num)
    }

    public static void main(String args[]) {
        Sub_class obj = new Sub_class();
        obj.my_method();
    }
}
```

# 4

## BAB 4 INHERITANCE

2. Eksplorasi gambar berikut sehingga menghasilkan program luaran yang bermaksa dan menerapkan konsep OOP

