

K203317 Pemrograman Berorientasi Objek

Konsep Dasar Pemrograman Berorientasi Objek

Semester Genap 2021/2022
Ibnu Santoso



Politeknik Statistika STIS
Prodi D-IV Komputasi Statistik


Agenda




- Class vs Object
- Inheritance
- Polymorphism
- Aggregation dan Composition

Class vs Object

- Di pertemuan sebelumnya...



Objek dan Class



For Better Official Statistics


Objek adalah entitas apapun yang memiliki properties dan behaviours tertentu baik dalam bentuk fisik ataupun logical.

Class adalah sebuah template atau cetak biru untuk menciptakan satu atau lebih objek dengan properties dan behaviours yang serupa

Properties = Variabel ; Behaviour = Method

Properties

FirstName:
LastName:
Gender:
DateOfBirth:
Occupation:




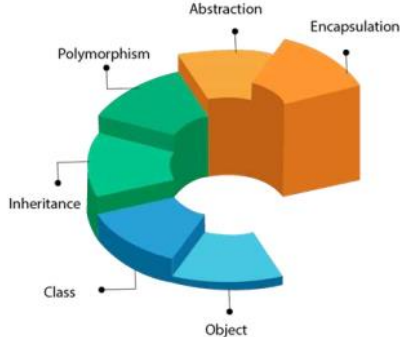
Behaviours

PayWages ()
BookHoliday ()
Appraisal ()
Promotion ()
SaveDetails ()

Class Person

```
P4.FirstName = "Bob"  
P4.LastName = "Jones"  
P4.Gender = "Male"  
P4.DateOfBirth = #01.01.1964#  
P4.Occupation = "Athlete"
```





OOPs (Object-Oriented Programming System)

<https://www.javatpoint.com>

The Circle Class



Pada Kelas **Circle** terdapat:

- Satu *instance private variable* bernama **radius** dengan tipe *double* dengan **default value 1**
- Dua *overloaded constructor*:
 - **Default constructor** tanpa argument
 - **Constructor** dengan argument **radius** bertipe **double**
- Satu method **getter** untuk mendapatkan nilai **radius**
- Satu method **setter** untuk mengeset nilai **radius**
- Dua **public method** untuk mengembalikan nilai **luas dan keliling**
- Satu method **toString** yang mengembalikan **informasi tentang instance**.

Circle	
▼ Attributes	+ -
-radius: double = 1.0	
▼ Operations	+ -
+CirIce() +Circle(radius: double) +getRadius(): double +setRadius(radius: double) : void +getArea(): double +getCircumference(): double +toString(): String	

The Code



Circle	
▼ Attributes	+ -
-radius: double = 1.0	
▼ Operations	+ -
+Cirlce() +Circle(radius: double) +getRadius(): double +setRadius(radius: double) : void +getArea(): double +getCircumference(): double +toString(): String	

- Save as “Circle.java” dan compile
- Dapatkah anda run? Mengapa?

```
public class Circle {  
    private double radius;  
  
    public Circle() {  
        radius = 1.0;  
    }  
  
    public Circle(double radius) {  
        this.radius = radius;  
    }  
  
    public double getRadius() {  
        return radius;  
    }  
  
    public void setRadius(double radius) {  
        this.radius = radius;  
    }  
  
    public double getArea() {  
        return Math.PI*radius*radius;  
    }  
  
    public double getCircumference() {  
        return Math.PI*radius*2;  
    }  
  
    public String toString(){  
        return "Circle[radius=" + radius +"]";  
    }  
}
```

How to Run

- Kelas **Circle** tidak memiliki **method main** sehingga tidak bisa dijalankan langsung
- Kelas **Circle** sebagai “**building block**” untuk digunakan di program lain
- Buat program untuk tes kelas Circle. Beri nama **TestCircle.java**

```
public class TestCircle {  
    public static void main(String[] args){  
        Circle c1 = new Circle();  
        System.out.println(c1);  
        System.out.println(c1.getArea());  
        System.out.println(c1.getCircumference());  
  
        Circle c2 = new Circle(2.0);  
        //c2.radius = 7.0; error  
        c2.setRadius(7.0);  
        System.out.println(c2);  
    }  
}
```

```
C:\Users\User\Documents\Old Doc\Kuliah\PBO>javac Circle.java  
  
C:\Users\User\Documents\Old Doc\Kuliah\PBO>javac TestCircle.java  
  
C:\Users\User\Documents\Old Doc\Kuliah\PBO>java TestCircle  
Circle[radius=1.0]  
3.141592653589793  
6.283185307179586  
Circle[radius=7.0]
```

Concept



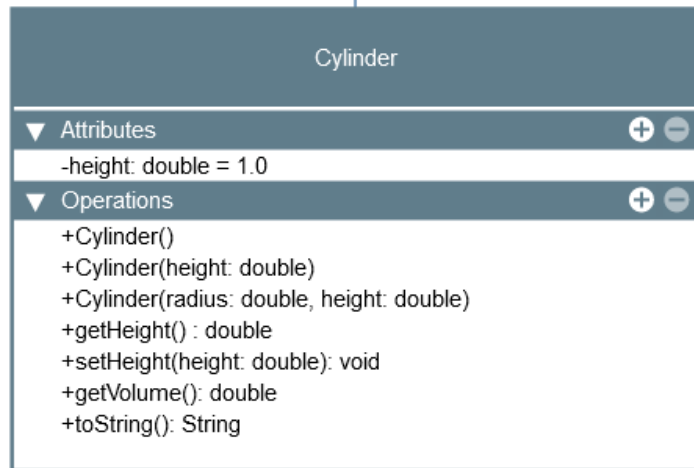
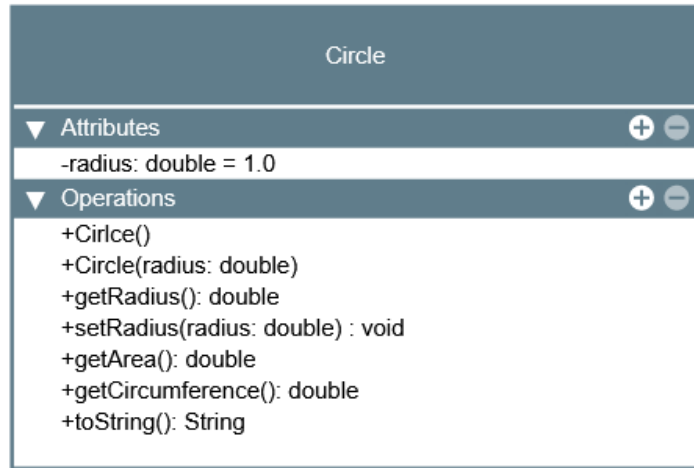
- **Constructor**: Untuk membuat objek dari kelas (instansiasi objek)
- **Getter**: Untuk mendapatkan atribut objek
- **Setter**: Untuk mengeset atribut objek
- **Public vs private**: Modifier untuk pengaturan akses
- **Keyword this**: this merujuk pada objek
- **Metode toString()**: Untuk mendapatkan deskripsi dari objek atau instance

Inheritance



- **Inheritance** mengacu pada **adopsi** semua properti non-privat dan metode dari satu kelas (**superclass**) oleh kelas lain (**subclass**).
- **Inheritance** adalah cara **membuat salinan kelas yang sudah ada sebagai titik awal untuk kelas yang lain**
- *Kembali ke **Kelas Circle***
- Misal programmer **kelas Circle** ingin **membuat kelas baru Bernama Kelas Silinder**.
- Maka programmer ini memiliki **2 pilihan**:
 - Membuat kelas Silinder dari awal
 - **Mengembangkan dari kelas Circle** karena pada dasarnya **silinder adalah circle dengan tambahan atribut tinggi**
- Misal programmer tersebut mengambil **pilihan yang kedua**

Cylinder is just a circle with height, isn't it?



Pada Kelas **Cylinder** terdapat:

- Satu *instance private variable* bernama **height** dengan tipe *double* dengan **default value 1**
- Tiga *overloaded constructor*:
 - **Default constructor** tanpa argument
 - **Constructor** dengan argument **height** bertipe **double**
 - **Constructor** dengan argument **radius** bertipe **double** dan **height** bertipe **double**
- Satu method **getter** untuk mendapatkan nilai **height**
- Satu method **setter** untuk mengeset nilai **height**
- Satu **public method** untuk mengembalikan nilai **volume**
- Satu method **toString** yang mengembalikan **informasi tentang instance**.

Let's code



```
public class Cylinder extends Circle {
    private double height;

    public Cylinder() {
        super();
        height = 1.0;
    }

    public Cylinder(double height) {
        super();
        this.height = height;
    }

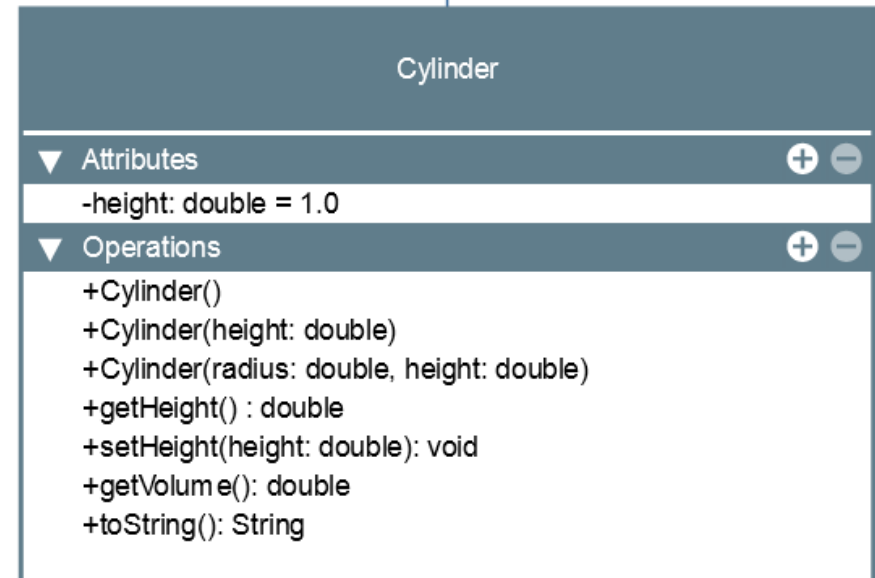
    public Cylinder(double radius, double height) {
        super(radius);
        this.height = height;
    }

    public double getHeight() {
        return height;
    }

    public void setHeight(double height) {
        this.height = height;
    }
}
```

```
public double getVolume() {
    return getArea()*height;
}

@Override
public String toString() {
    return "Cylinder: subclass of " +
        super.toString() + " height=" + height;
}
}
```



Buat Kelas TestCylinder.java

```
public class TestCylinder {  
    public static void main (String[] args) {  
  
        Cylinder c1 = new Cylinder();  
        System.out.println("Cylinder:"  
            + " radius=" + c1.getRadius()  
            + " height=" + c1.getHeight()  
            + " base area=" + c1.getArea()  
            + " volume=" + c1.getVolume());  
  
        Cylinder c2 = new Cylinder(10.0);  
        System.out.println("Cylinder:"  
            + " radius=" + c2.getRadius()  
            + " height=" + c2.getHeight()  
            + " base area=" + c2.getArea()  
            + " volume=" + c2.getVolume());  
  
        Cylinder c3 = new Cylinder(2.0, 10.0);  
        System.out.println("Cylinder:"  
            + " radius=" + c3.getRadius()  
            + " height=" + c3.getHeight()  
            + " base area=" + c3.getArea()  
            + " volume=" + c3.getVolume());  
  
        System.out.println(c3);  
    }  
}
```

```
C:\Users\User\Documents\Old Doc\Kuliah\PBO>javac Cylinder.java
```

```
C:\Users\User\Documents\Old Doc\Kuliah\PBO>javac TestCylinder.java
```

```
C:\Users\User\Documents\Old Doc\Kuliah\PBO>java TestCylinder
```

```
Cylinder: radius=1.0 height=1.0 base area=3.141592653589793 volume=3.141592653589793
```

```
Cylinder: radius=1.0 height=10.0 base area=3.141592653589793 volume=31.41592653589793
```

```
Cylinder: radius=2.0 height=10.0 base area=12.566370614359172 volume=125.66370614359172
```

```
Cylinder: subclass of Circle[radius=2.0] height=10.0
```

Polymorphisme



- **Poly**= banyak
- **Morphisme** = bentuk
- Dalam OOP bisa diartikan dengan sederhana sebagai **method** dengan nama yang sama namun mengerjakan hal yang berbeda
- *Static polymorphism* -> *method overloading*
- *Dynamic Polymorphism* -> *method overriding*

Method Overloading



- Sebuah kelas memiliki **dua atau lebih method** dengan **nama yang sama**
 - namun memiliki **parameter yang berbeda**
 - atau **tipe data kembalian yang berbeda**
- **Tujuan** dari *method overloading* yaitu memudahkan penggunaan atau pemanggilan *method* dengan **fungsionalitas yang mirip**.
- **Aturan Method Overloading**
 - Nama method **harus sama** dengan method lainnya.
 - Parameter haruslah **berbeda**.
 - Return **boleh sama, boleh berbeda**.

Contoh: Cetak.java



```
public class Cetak {
    static double maxNumber(double a, double b) {
        if (a < b) {
            return a;
        }else{
            return b;
        }
    }

    static int maxNumber(int a, int b) {
        if (a < b){
            return a;
        }else {
            return b;
        }
    }

    public static void main(String[] args) {
        System.out.println(maxNumber(5.5, 7.5));
        System.out.println(maxNumber(10, 20));
    }
}
```

```
C:\Users\User\Documents\Old Doc\Kuliah\PBO>javac Cetak.java
```

```
C:\Users\User\Documents\Old Doc\Kuliah\PBO>java Cetak
5.5
10
```

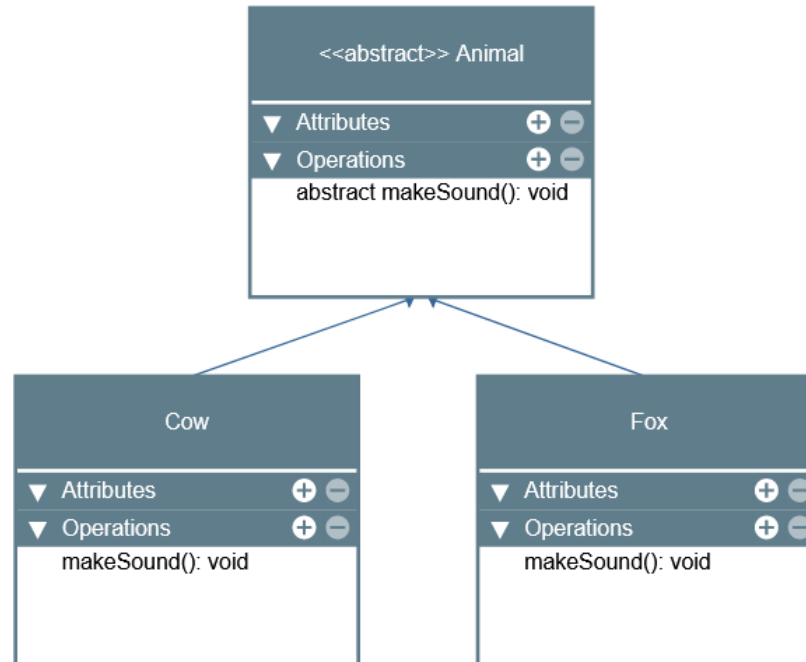
OOT

- Terdapat keyword **static**
- **Static variable** artinya **variable kelas**.
Nilainya sama untuk semua instance.
- **Static method** artinya **metode kelas**. **Dapat langsung dipanggil tanpa instansiasi objek dengan [namakelas].[namametodestatic]**

Method Overriding



- Sebuah kelas yang merupakan *turunan kelas lain*, atau menerapkan *interface tertentu*, mengimplementasikan sebuah metode yang sama dengan cara yang berbeda dengan kelas lainnya



AnimalTest.java



```
abstract class Animal{
    abstract void makeSound();
}

class Cow extends Animal{
    @Override
    void makeSound(){
        System.out.println("Cow goes moo..moo..");
    }
}

class Fox extends Animal{
    @Override
    void makeSound(){
        System.out.println("What does the fox say?");
        System.out.println("ring ding ding dinging
ding...");
    }
}

public class AnimalTest{
    public static void main (String args[]){
        Animal sapi = new Cow();
        sapi.makeSound();
        Animal fox = new Fox();
        fox.makeSound();
    }
}
```

```
C:\Users\User\Documents\Old Doc\Kuliah\PBO>javac AnimalTest.java
```

```
C:\Users\User\Documents\Old Doc\Kuliah\PBO>java AnimalTest
Cow goes moo..moo..
What does the fox say?
ring ding ding dinging ding...
```

Ref: https://youtu.be/jofNR_WkoCE

OOT

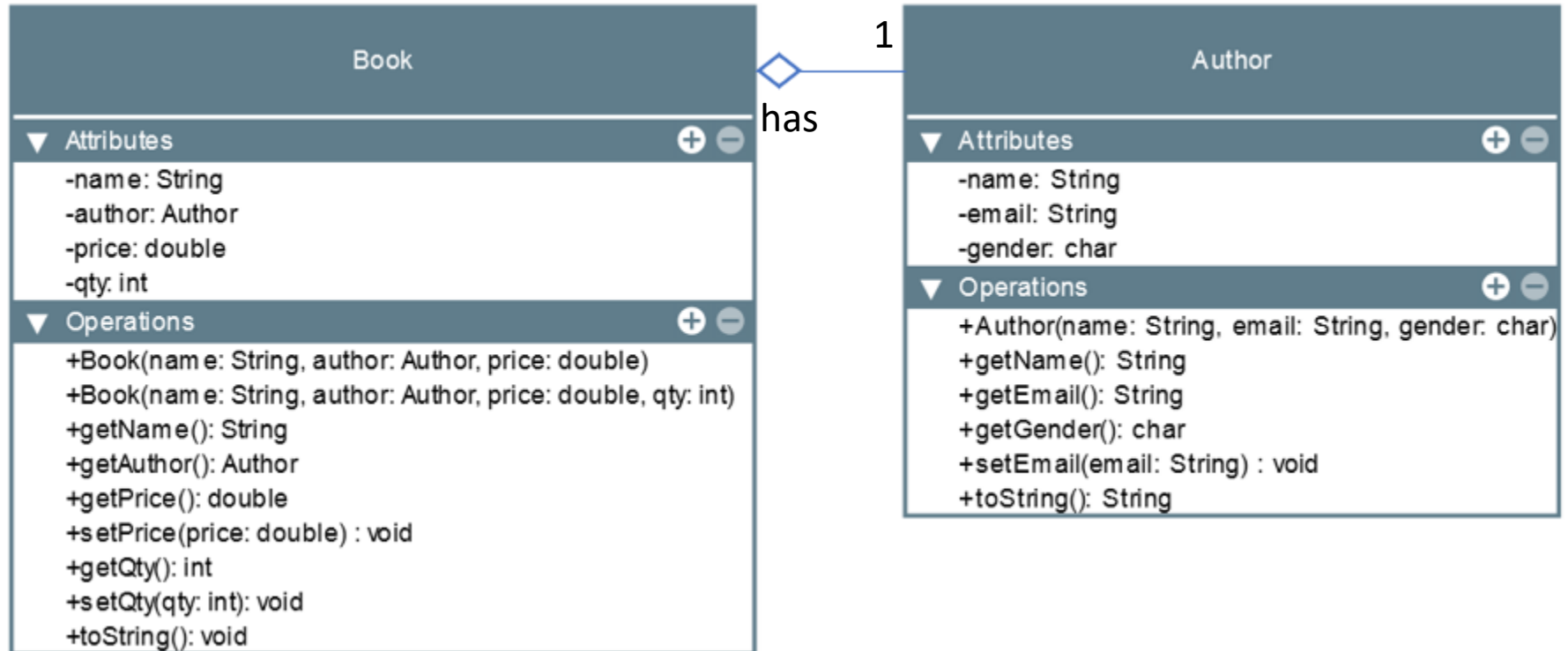
- Terdapat keyword **abstract**
- **Abstract method** adalah metode tanpa implementasi, yang **dimaksudkan untuk di-override** oleh kelas turunannya.
- Pembahasan lebih lanjut/mendalam di pekan depan, stay tuned...

Aggregation dan Composition



- Terdapat **dua macam** tipe hubungan antara objek pada konsep OOP:
 - **IS-A**
 - **HAS-A**
- **Hubungan IS-A** adalah **inheritance**.
 - *A Parrot is A Bird*: Bird adalah *base class*, Parrot diturunkan dari Bird. Hubungan antar dua kelas ini adalah hubungan **IS-A**
- **Hubungan HAS-A** adalah association
 - *A Car has an Engine*: di dalam kelas Car terdapat kelas Engine. Hubungan antar dua kelas ini adalah hubungan **HAS-A**
 - Hubungan **HAS-A** bisa berupa **one-to-one, one-to-many, many-to-one, many-to-many**
- Terdapat dua macam hubungan association:
 - **Aggregation**: loosely coupled relation, weak association
 - **Composition**: more tightly coupled relation, strong association

UML Diagram



Author.java



```
public class Author{
    private String name;
    private String email;
    private char gender;

    public Author(String name, String email, char gender){
        this.name = name;
        this.email = email;
        this.gender = gender;
    }

    public String getName(){
        return name;
    }

    public String getEmail(){
        return email;
    }

    public char getGender(){
        return gender;
    }

    public void setEmail(String email){
        this.email = email;
    }

    public String toString(){
        return "Author[name="+ name +
            ",email=" + email +
            ",gender="+ gender + " ]";
    }
}
```

Author	
▼ Attributes	+ -
-name: String -email: String -gender: char	
▼ Operations	+ -
+Author(name: String, email: String, gender: char) +getName(): String +getEmail(): String +getGender(): char +setEmail(email: String) : void +toString(): String	

Book.java



```
public class Book{
    private String name;
    private Author author;
    private double price;
    private int qty;

    public Book(String name, Author author, double price){
        this.name = name;
        this.author = author;
        this.price = price;
        this.qty = 0;
    }

    public Book(String name, Author author, double price, int qty){
        this.name = name;
        this.author = author;
        this.price = price;
        this.qty = qty;
    }

    public String getName(){
        return name;
    }

    public Author getAuthor(){
        return author;
    }

    public double getPrice(){
        return price;
    }
}
```

```
    public void setPrice(double price){
        this.price=price;
    }

    public int getQty(){
        return qty;
    }

    public void setQty(int qty){
        this.qty = qty;
    }

    public String toString(){
        return "Book[name=" +name+
            ",author="+author+
            ",price="+price+
            ",qty="+qty+"]";
    }
}
```

BookTest.java



```
public class BookTest{
    public static void main(String args[]){
        Author nano = new Author("Nano Yulian P.", "nano@bps.go.id", 'm');
        System.out.println(nano);

        Book oopBook = new Book("OOP for dummies", nano, 50000, 100);
        System.out.println(oopBook);

        oopBook.setPrice(35000);
        oopBook.setQty(50);

        Book javaBook = new Book("Java for dummies",
        new Author("Wa Ode Zuhayeni M.", "yeni@bps.go.id", 'f'), 10000);

        System.out.println(javaBook);
        Author yeni = javaBook.getAuthor();
        System.out.println(yeni);
    }
}
```

```
C:\Users\User\Documents\Old Doc\Kuliah\PBO>javac Author.java
```

```
C:\Users\User\Documents\Old Doc\Kuliah\PBO>javac Book.java
```

```
C:\Users\User\Documents\Old Doc\Kuliah\PBO>javac BookTest.java
```

```
C:\Users\User\Documents\Old Doc\Kuliah\PBO>java BookTest
```

```
Author[name=Nano Yulian P.,email=nano@bps.go.id,gender=m]
```

```
Book[name=OOP for dummies,author=Author[name=Nano Yulian P.,email=nano@bps.go.id,gender=m],price=50000.0,qty=100]
```

```
Book[name=Java for dummies,author=Author[name=Wa Ode Zuhayeni M.,email=yeni@bps.go.id,gender=f],price=10000.0,qty=0]
```

```
Author[name=Wa Ode Zuhayeni M.,email=yeni@bps.go.id,gender=f]
```

Terima Kasih

