

# Pemrograman Berorientasi Obyek

## Overriding dan Overloading


Oleh Politeknik Elektronika Negeri Surabaya  
2020



Politeknik Elektronika Negeri Surabaya  
Departemen Teknik Informatika dan Komputer

1

# Overriding



Politeknik Elektronika Negeri Surabaya  
Departemen Teknik Informatika & Komputer

2

## Overriding

- Suatu keadaan dimana subclass yang berusaha memodifikasi tingkah laku (method) yang diwarisi dari superclass.
- Tujuan: subclass memiliki tingkah laku yang lebih spesifik.
- Dilakukan dengan cara mendeklarasikan kembali method milik parent class di subclass.



## Overriding

- Deklarasi method pada subclass harus sama dengan yang terdapat di super class. Kesamaan pada:
  - Nama
  - Return type
  - Daftar parameter (jumlah, tipe, dan urutan)
- Method pada parent class disebut overridden method
- Method pada subclass disebut overriding method.



## Contoh Overriding

```
public class Employee{
    public String name;
    public double salary;

    public String getDetail(){
        return "Name : " + name +
            "\n" +
            "Salary : " + salary;
    }
}
```

```
public class Manager extends Employee{
    public String departemen;

    public String getDetail(){
        return "Name : " + name +
            "\n" +
            "Salary : " + salary +
            "\n" +
            "Departemen : " + departemen;
    }
}
```



## Aturan Overriding

- Mode akses (modifier) overriding method harus sama atau lebih luas dari pada overridden method.
- Subclass hanya boleh meng-override method superclass satu kali saja, tidak boleh ada lebih dari satu method pada kelas yang sama yang sama persis.
- Overriding method tidak boleh throw checked exceptions yang tidak dideklarasikan oleh overridden method.



# Overloading



Politeknik Elektronika Negeri Surabaya  
Departemen Teknik Informatika & Komputer

7

Politeknik Elektronika Negeri Surabaya

## Overloading

- Overloading adalah suatu keadaan dimana beberapa method sekaligus dapat mempunyai nama yang sama, akan tetapi mempunyai fungsionalitas yang berbeda tapi mirip
- Cara: Menuliskan kembali method dengan nama yang sama pada suatu class atau antar parent dan subclass.
- Tujuan : memudahkan penggunaan/pemanggilan method dengan fungsionalitas yang **mirip**.



Departemen Teknik Informatika & Komputer

8

## Contoh Overloading

- 1 titik → menggambar titik
- 2 titik → menggambar garis
- 3 titik → menggambar segitiga
- 4 titik → menggambar persegi empat
- dst...



## Pemrograman Terstruktur

- Jika pemrograman terstruktur, kita harus menyelesaikan kasus diatas dengan cara membuat fungsi untuk masing-masing gambar dengan nama yang harus berbeda dan melakukan tugas penggambaran masing-masing fungsi.
  - titik(int t1)  
1 parameter titik, untuk menggambar titik
  - garis(int t1, int t2)  
2 parameter titik, untuk menggambar garis
  - segitiga(int t1, int t2, int t3)  
3 parameter titik, untuk menggambar segitiga
  - persegiEmpat(int t1, int t2, int t3, int t4)  
4 parameter titik, untuk menggambar persegi empat
  - dst...



## Contoh method overloading

- `void gambar(int t1)`  
1 parameter titik, untuk menggambar titik
- `void gambar(int t1, int t2)`  
2 parameter titik, untuk menggambar garis
- `void gambar(int t1, int t2, int t3)`  
3 parameter titik, untuk menggambar segitiga
- `void gambar(int t1, int t2, int t3, int t4)`  
4 parameter titik, untuk menggambar persegi empat
- dst...



## Aturan Overloading

- Nama method harus sama
- Daftar parameter harus berbeda
- Return type boleh sama, juga boleh berbeda



## Daftar Parameter pada Overloading

- Perbedaan daftar parameter bukan hanya terjadi pada perbedaan banyaknya parameter, tetapi juga urutan dari parameter tersebut.
- Misalnya saja dua buah parameter berikut ini :
  - `function_member(int x, String n)`
  - `function_member(String n, int x)`
- Dua parameter tersebut juga dianggap berbeda daftar parameternya.



## Signature Method

- Signature method pada OO yaitu jumlah parameter yang ada pada method dan tipe data parameternya.
- Return type tidak termasuk signature
- Jadi kita tidak dapat melakukan overloading hanya dengan perbedaan return type.



## Contoh Overloading

- `public void println(int nilai)`
- `public void println(double luas)`
- `public void println(String nama)`
- `public void println(String nama, int nilai)`



## Contoh

```
public class Bentuk {
    public void gambar(int t1) {
        // 1 parameter titik, untuk menggambar titik
    }
    public void gambar(int t1, int t2) {
        // 2 parameter titik, untuk menggambar garis
    }
    public void gambar(int t1, int t2, int t3) {
        // 3 parameter titik, untuk menggambar segitiga
    }
    public void gambar(int t1, int t2, int t3, int t4) {
        // 4 parameter titik, untuk menggambar persegi empat
    }
}
```





## Aturan Overloading

<u>return type</u>	<u>nama method</u>	<u>daftar parameter</u>
void	Gambar	(int t1)
void	Gambar	(int t1, int t2)
void	Gambar	(int t1, int t2, int t3)
void	Gambar	(int t1, int t2, int t3, int t4)
↓	↓	↓
Boleh sama atau beda	sama	berbeda



- Overloading juga bisa terjadi antara parent class dengan subclass-nya jika memenuhi ketiga syarat overload.
- Misalnya saja dari class Bentuk pada contoh sebelumnya kita turunkan sebuah class baru yang bernama WarnaiBentuk.



```
public class WarnaiBentuk extends Bentuk {  
    public void gambar(String warna, int t1, int t2, int3) {  
        ...  
    }  
  
    public void gambar(String warna, int t1, int t2, int3, int t4) {  
        ...  
    }  
    ...  
}
```



## Overloading Constructor



## Overloading Constructor

- Overloading yang terjadi pada constructor sebuah class
- Aturan overloading constructor sama dengan overloading method
- Hanya saja tidak ada return type pada constructor



## Contoh Overloading Constructor

```
public class Employee{
    public Employee(String name){
        this.name = name;
    }
    public Employee(String name, String address){
        this.name = name;
        this.address = address;
    }
    public Employee(String name, String address, double salary){
        this.name = name;
        this.address = address;
        this.salary = salary;
    }
}
```



## Ringkasan

Overriding	Overloading	Overloading Constructor
Terjadi antara super-class dan sub-class	Dapat terjadi pada satu class atau Antara super class dan subclass	Terjadi pada satu class
Nama method harus sama	Nama method harus sama	Nama constructor harus sama
Return type harus sama	Return type boleh sama boleh beda	Tidak ada return type
Daftar parameter harus sama	Daftar parameter harus beda	Daftar parameter harus beda

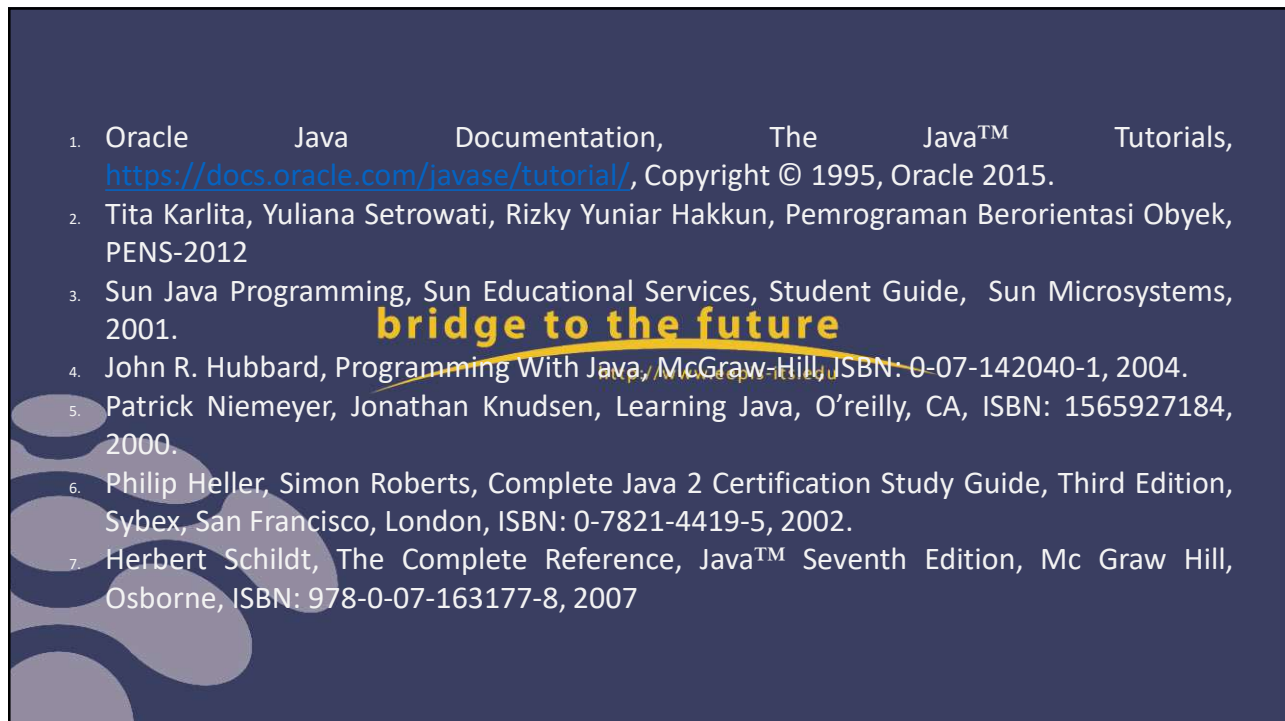


Overriding	Overloading	Overloading Constructor
<pre>public class Parent{     public void cetak(){} }  public class Child extends Parent{     public void cetak(){} }</pre>	<pre>public class testOverloading{     public void gambar(int t1){}     public void gambar(int t1, int t2){}     public void gambar(int t1, int t3){} }</pre>	<pre>public class Canvas{     public Canvas(){}     public Canvas(int t1){}     public Canvas(int t1, int t2){} }</pre>
	<pre>public class Parent{     public void gambar(int t1){}     public void gambar(int t1, int t2){} }  public class Child extends Parent{     public void gambar(String warna, int t1){}     public void gambar(String warna, int t1, int t2){} }</pre>	





25



26