

Desain dan Pemrograman Berorientasi Objek

{

Polimorfisme
(*Polymorphism*)

Rosa Ariani Sukamto



- ✎ Blog: <http://hariiniadalahhadiah.wordpress.com>
- ✎ Facebook: <https://www.facebook.com/rosa.ariani.sukamto>
- ✎ Email: rosa.ariani@upi.edu
- ✎ Website: <https://rosa-as.id>
- ✎ Youtube: <https://www.youtube.com/c/RosaArianiSukamto>

- ↳ Polimorfisme berarti banyak bentuk
- ↳ Pada pemrograman berorientasi objek, polimorfisme merupakan konsep yang menyatakan sesuatu yang sama dapat memiliki berbagai bentuk dan perilaku yang berbeda
- ↳ Polimorfisme misalnya beberapa metode yang memiliki nama yang sama diijinkan dalam pemrograman berorientasi objek di dalam sebuah kelas atau berada pada kelas turunannya asalkan masih memiliki identitas yang tidak sama persis, misalnya berbeda parameter masukan metode atau berbeda nama kelas

Polimorfisme

EXAMPLES (1)

Constructor Summary

Constructors

Constructor and Description

Scanner(File source)

Constructs a new Scanner that produces values scanned from the specified file.

Scanner(File source, String charsetName)

Constructs a new Scanner that produces values scanned from the specified file.

Scanner(InputStream source)

Constructs a new Scanner that produces values scanned from the specified input stream.

Scanner(InputStream source, String charsetName)

Constructs a new Scanner that produces values scanned from the specified input stream.

Scanner(Path source)

Constructs a new Scanner that produces values scanned from the specified file.

Scanner(Path source, String charsetName)

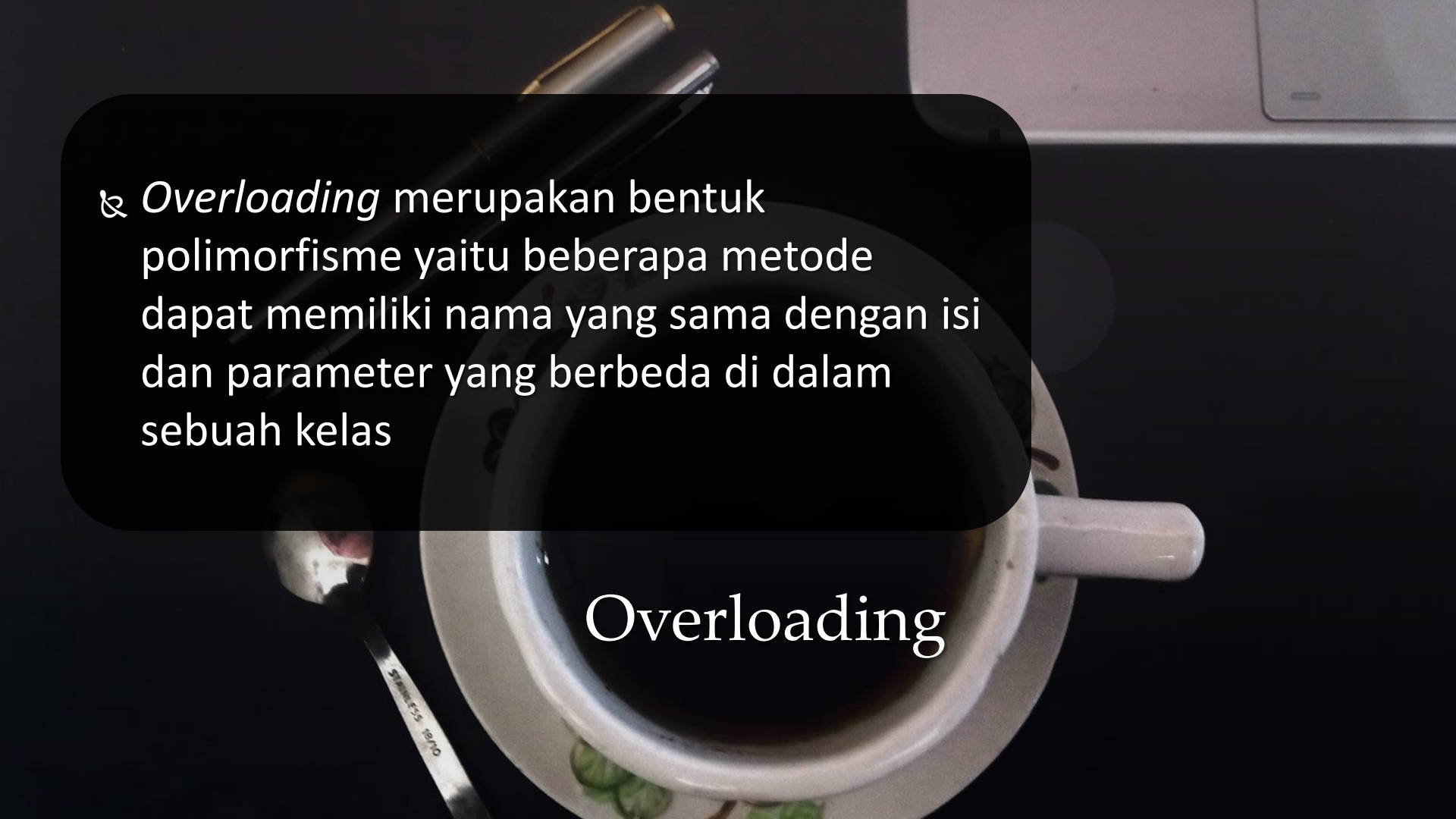
Constructs a new Scanner that produces values scanned from the specified file.

Scanner(Readable source)

Constructs a new Scanner that produces values scanned from the specified source.

EXAMPLES (2)

String	findInLine (Pattern pattern) Attempts to find the next occurrence of the specified pattern ignoring delimiters.
String	findInLine (String pattern) Attempts to find the next occurrence of a pattern constructed from the specified string, ignoring delimiters.
String	findWithinHorizon (Pattern pattern, int horizon) Attempts to find the next occurrence of the specified pattern.
String	findWithinHorizon (String pattern, int horizon) Attempts to find the next occurrence of a pattern constructed from the specified string, ignoring delimiters.
boolean	hasNext () Returns true if this scanner has another token in its input.
boolean	hasNext (Pattern pattern) Returns true if the next complete token matches the specified pattern.
boolean	hasNext (String pattern) Returns true if the next token matches the pattern constructed from the specified string.
boolean	hasNextBigDecimal () Returns true if the next token in this scanner's input can be interpreted as a BigDecimal using the nextBigDecimal () method.

A top-down photograph of a white ceramic coffee cup filled with dark liquid, sitting on a matching saucer. A silver spoon with a black handle lies to the left of the cup. A black pen with a silver clip is positioned diagonally across the top left. The entire scene is set against a dark, reflective surface.

& *Overloading* merupakan bentuk polimorfisme yaitu beberapa metode dapat memiliki nama yang sama dengan isi dan parameter yang berbeda di dalam sebuah kelas

Overloading

Implementasi overloading - C++ (1)

```
class Buku{
    private:
        string judul;
        int tahun;
        string pengarang;

    public:
        Buku() {
        }

        Buku(string judul, int tahun,
            string pengarang){
            this->judul = judul;
            this->tahun = tahun;
            this->pengarang = pengarang;
        }

        void setBuku(string judul){
            this->judul = judul;
        }
}
```

```
void setBuku(string judul,
    int tahun){
    this->judul = judul;
    this->tahun = tahun;
}

void setBuku(string judul,
    int tahun, string pengarang){
    this->judul = judul;
    this->tahun = tahun;
    this->pengarang = pengarang;
}

//get dan set
.....
~Buku() {
}

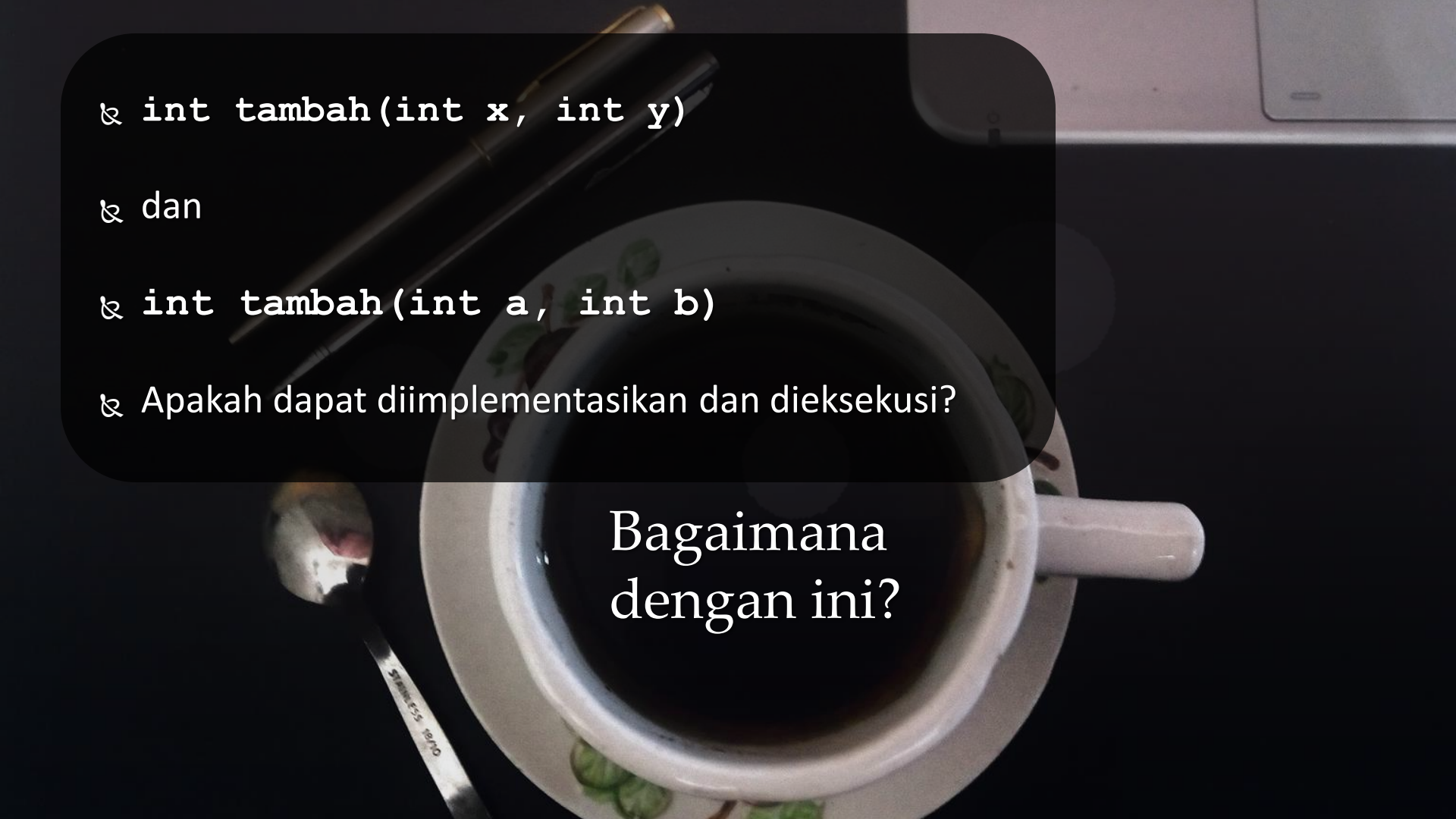
};
```

Implementasi overloading - C++ (2)

```
#include <string>
#include <iostream>
using namespace std;
#include "Buku.cpp"

int main() {
    Buku obuku;
    obuku.setBuku("PBO");
    cout << "Judul: " << obuku.getJudul() << endl;
    cout << "Tahun: " << obuku.getTahun() << endl;
    cout << "Pengarang: " << obuku.getPengarang() << endl;

    obuku.setBuku("PBO2", 2011);
    cout << "Judul: " << obuku.getJudul() << endl;
    cout << "Tahun: " << obuku.getTahun() << endl;
    cout << "Pengarang: " << obuku.getPengarang() << endl;
    return 0;
}
```

& `int tambah(int x, int y)`

& dan

& `int tambah(int a, int b)`

& Apakah dapat diimplementasikan dan dieksekusi?

Bagaimana
dengan ini?



& Tidak

& Karena yang dilihat adalah tipe, bukan nama, sehingga kedua metode itu sama-sama

& `int tambah(int, int)`

Jawabnya.....

Implementasi overloading - PHP (1)

```
<?php
```

```
class Buku{
```

```
    private $judul;
```

```
    private $tahun;
```

```
    private $pengarang;
```

```
    function __construct(){
```

```
    }
```

```
    function setBuku($judul){
```

```
        $this->judul = $judul;
```

```
    }
```

```
    function setBuku($judul, $tahun){
```

```
        $this->judul = $judul;
```

```
        $this->tahun = $tahun;
```

```
    }
```

```
function setBuku($judul,
```

```
    $tahun, $pengarang){
```

```
    $this->judul = $judul;
```

```
    $this->tahun = $tahun;
```

```
    $this->pengarang = $pengarang;
```

```
    }
```

```
}
```

```
?>
```

Implementasi overloading - PHP (2)

```
<?php
    include "Buku.php";

    $obuku = new Buku();
    $obuku->setBuku("PBO");

    echo "Judul: ".$obuku->getJudul()."<br/>";
    echo "Tahun: ".$obuku->getTahun()."<br/>";
    echo "Pengarang: ".$obuku->getPengarang()."<br/>";

    $obuku->setBuku("PBO2", 2011);
    echo "Judul: ".$obuku->getJudul()."<br/>";
    echo "Tahun: ".$obuku->getTahun()."<br/>";
    echo "Pengarang: ".$obuku->getPengarang()."<br/>";

?>
```

Implementasi overloading - PHP (3)

```
function setBuku($judul="", $tahun = 0){  
    $this->judul = $judul;  
    $this->tahun = $tahun;  
}
```


Implementasi overloading - Java (1)

```
class Buku{

    private String judul;
    private int tahun;
    private String pengarang;

    Buku() {
    }

    Buku(String judul, int tahun,
    String pengarang) {

        this.judul = judul;
        this.tahun = tahun;
        this.pengarang = pengarang;
    }

    public void setBuku(String judul) {
        this.judul = judul;
    }
}
```

```
    public void setBuku(String judul,
        int tahun) {
        this.judul = judul;
        this.tahun = tahun;
    }

    public void setBuku(String judul,
        int tahun, String pengarang) {

        this.judul = judul;
        this.tahun = tahun;
        this.pengarang = pengarang;
    }

    //get dan set
    .....
}
```

Implementasi overloading - java (2)

```
class Main{

    public static void main(String[] args){

        Buku obuku;

        obuku = new Buku();

        obuku.setBuku("PBO");

        System.out.println("Judul: " + obuku.getJudul());
        System.out.println("Tahun: " + obuku.getTahun());
        System.out.println("Pengarang: " + obuku.getPengarang());

        obuku.setBuku("PBO2", 2011);

        System.out.println("Judul: " + obuku.getJudul());
        System.out.println("Tahun: " + obuku.getTahun());
        System.out.println("Pengarang: " + obuku.getPengarang());

    }

}
```

Implementasi overloading - PYTHON (1)

```
class Buku:
```

```
    def __init__(self, judul, tahun = 0,  
    pengarang = ''):
```

```
        #konstruktor
```

```
            self.judul = judul
```

```
            self.tahun = tahun
```

```
            self.pengarang = pengarang
```

```
    def setBuku(self, judul, tahun = 0,  
    pengarang = ''):
```

```
        self.judul = judul
```

```
        self.tahun = tahun
```

```
        self.pengarang = pengarang
```

```
    def getJudul(self):
```

```
        return self.judul
```

```
    def getTahun(self):
```

```
        return self.tahun
```

```
    def getPengarang(self):
```

```
        return self.pengarang
```

Implementasi overloading - PYTHON (2)

```
from Buku import Buku

b1 = Buku('Java')
b2 = Buku('PBO', 2011, 'RAS')

print("b1 : Judul : " + str(b1.getJudul()))
print("b2 : Judul : " + str(b2.getJudul()))
print("b2 : Tahun : " + str(b2.getTahun()))
print("b2 : Pengarang : " + str(b2.
getPengarang()))
```

& *Overriding* merupakan bentuk polimorfisme yaitu beberapa metode pada kelas orang tua dapat ditulis ulang pada kode kelas anak dalam pewarisan (*inheritance*) dengan memiliki nama yang sama dan memiliki isi ataupun parameter yang sama atau berbeda

Overriding

Implementasi overriding- C++ (1)

```
class BangunDatar{  
  
    public:  
  
        BangunDatar() {  
        }  
  
        int luas() {  
            return 0;  
        }  
  
        ~BangunDatar() {  
        }  
  
};
```

```
class Persegi : public BangunDatar{  
  
    public:  
  
        Persegi() {  
        }  
  
        int luas(int p, int l){  
  
            return (p * l);  
        }  
  
        ~Persegi() {  
        }  
  
};
```

Implementasi overriding - C++ (2)

```
#include <iostream>
using namespace std;
#include "BangunDatar.cpp"
#include "Persegi.cpp"

int main() {

    Persegi opersegi;

    opersegi.luas(5, 6);

    return 0;
}
```

Implementasi overriding - PHP (1)

```
<?php

class BangunDatar{

    function BangunDatar(){
    }

    function luas(){
        return 0;
    }

}

?>
```

```
<?php

class Persegi extends BangunDatar{

    function Persegi(){
    }

    function luas($p, $l){

        return ($p * $l);
    }

}

?>
```

Implementasi overriding - PHP (2)

```
<?php

include "BangunDatar.php";
include "Persegi.php";

$persegi = new Persegi();

$persegi->luas(5, 6);

?>
```

Implementasi overriding - Java (1)

```
class BangunDatar{  
  
    BangunDatar () {  
    }  
  
    int luas () {  
        return 0;  
    }  
  
}
```

```
class Persegi extends BangunDatar{  
  
    Persegi () {  
    }  
  
    int luas (int p, int l) {  
        return (p * l);  
    }  
  
}
```


Implementasi overriding - Java (2)

```
class Main{  
  
    public static void main(String[] args){  
  
        Persegi opersegi;  
        opersegi = new Persegi();  
  
        opersegi.luas(5, 6);  
  
    }  
  
}
```

Implementasi overriding - PYTHON

```
class BangunDatar:

    def __init__(self):
        #konstruktor
        print("Bangun Datar")

    def luas(self):
        return 0

class Persegi(BangunDatar):

    def __init__(self):
        #konstruktor
        print("Persegi")

    def luas(self, p=0, l=0):
        print(p)
        print(l)
        return (p * l)

opersegi = Persegi

print(opersegi.luas(opersegi, 5,6))
```



```
Ortu a = new Ortu();  
Anak b = a;
```

Atau

```
Anak b = new Anak();  
Ortu a = b;
```

& Yang bisa diimplementasikan dan dieksekusi

Coba di coding

