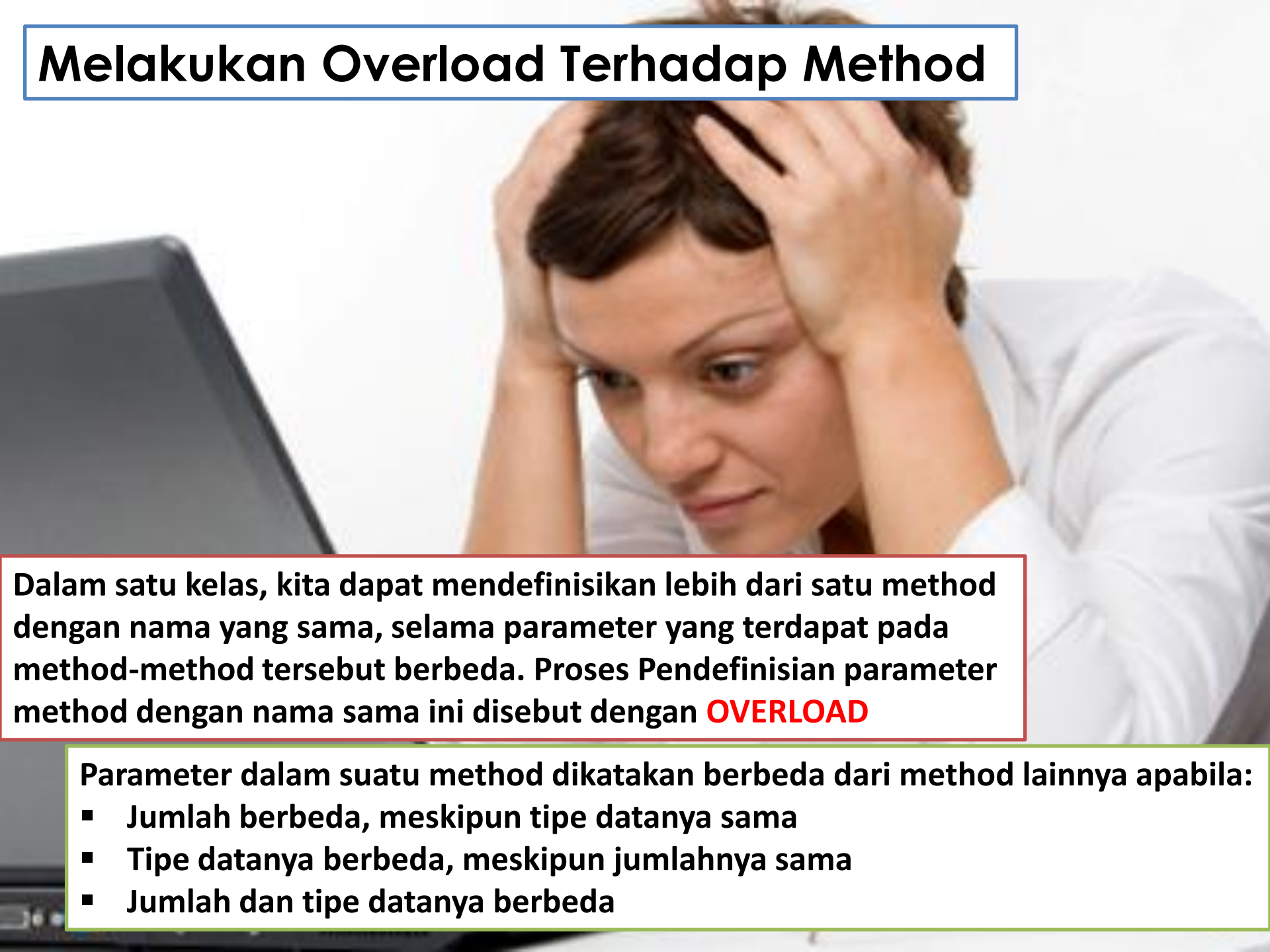
Class dan Objek (2)

Brigida Arie Minartiningtyas, M.Kom

Melakukan Overload Terhadap Method



Dalam satu kelas, kita dapat mendefinisikan lebih dari satu method dengan nama yang sama, selama parameter yang terdapat pada method-method tersebut berbeda. Proses Pendefinisian parameter method dengan nama sama ini disebut dengan OVERLOAD

Parameter dalam suatu method dikatakan berbeda dari method lainnya apabila:

- **Jumlah berbeda, meskipun tipe datanya sama**
- **Tipe datanya berbeda, meskipun jumlahnya sama**
- **Jumlah dan tipe datanya berbeda**

```java
public class Pembagian {

    int bagi (int a, int b){
        return a/b;
    }

    double bagi (double a, double b){
        return a/b;
    }
}
```

*Method yang dipanggil adalah method yang daftar parameternya **paling sesuai** dengan tipe argumen yang dilewatkan*

```java
public class DemoOverload1 {
    public static void main (String [] args){

        Pembagian b = new Pembagian();

        int x = b.bagi (10,4);
        double y = b.bagi(10.0, 4.0);

        System.out.println("Hasil bagi tipe int = " +x);
        System.out.println("Hasil bagi tipe double = " +y);
    }
}
```

DemoOverload1 > main

put - Pertemuan6 (run)

```
run:
Hasil bagi tipe int = 2
Hasil bagi tipe double = 2.5
BUILD SUCCESSFUL (total time: 1 second)
```

```
public class myclass {

    void tulis (){
        System.out.println("Tanpa Parameter");
    }

    void tulis (double d){
        System.out.println("Satu parameter dengan nilai argumen : " +d);
    }
}
```

```
public class DemoOverload2 {
    public static void main (String [] args){

        myclass c = new myclass ();

        c.tulis();
        c.tulis(4);
    }
}
```

Dikonversi menjadi tipe double

DemoOverload2 > main

tput - Pertemuan6 (run)

```
run:
Tanpa Parameter
Satu parameter dengan nilai argumen : 4.0
BUILD SUCCESSFUL (total time: 1 second)
```

```java
public class myclass2 {

    void tulis (int a, String s){
        System.out.println("int : " +a+ ", String : \""+s+"\"");
    }

    void tulis (String s, int a){
        System.out.println("String : \"" +s+ "\", int : "+a+"");
    }
}
```

Meskipun jumlah dan tipe dari parameter yang terdapat pada beberapa method sama, tapi jika urutannya berbeda, maka kedua method tersebut juga akan dianggap berbeda.

```java
public class DemoOverload3 {
    public static void main (String [] args){

        myclass2 c = new myclass2();

        c.tulis(4, "Contoh method pertama");
        c.tulis("Contoh method kedua", 8);
    }
}
```

DemoOverload3 > main

put - Pertemuan6 (run)

```
run:
int : 4, String : "Contoh method pertama"
String : "Contoh method kedua", int : 8
BUILD SUCCESSFUL (total time: 0 seconds)
```

Overload Pada Constructor

Selain pada method normal pada umumnya, proses overload juga dapat diaplikasikan ke dalam constructor suatu kelas.

Hal ini disebabkan karena sebenarnya constructor juga adalah sebuah method yang mengembalikan tipe kelas (dirinya sendiri)

```java
public class Kotak {
    double panjang;
    double lebar;
    double tinggi;

    Kotak(){
        panjang = 0;
        lebar = 0;
        tinggi = 0;
    }

    Kotak (double p, double l, double t){
        panjang = p;
        lebar = l;
        tinggi = t;
    }
    Kotak (double sisi){
        panjang = lebar = tinggi = sisi;
    }

    double hitungVolume(){
        return (panjang*lebar*tinggi);
    }
}
```

Tanpa parameter

Tiga parameter

Satu parameter

```java
public class DemoOverloadConstructor {
    public static void main (String [] args){

        Kotak k1, k2, k3;

        k1 = new Kotak();
        k2 = new Kotak(10);
        k3 = new Kotak(4,3,2);

        System.out.println("Volume k1 = " +k1.hitungVolume());
        System.out.println("Volume k1 = " +k2.hitungVolume());
        System.out.println("Volume k1 = " +k3.hitungVolume());
    }
}
```

DemoOverloadConstructor > main

put - Pertemuan6 (run)

```
run:
Volume k1 = 0.0
Volume k1 = 1000.0
Volume k1 = 24.0
BUILD SUCCESSFUL (total time: 1 second)
```

Objek Sebagai Parameter

Dalam Java, objek juga dapat berperan sebagai parameter dari sebuah method

```java
public class Kotak2 {
    double panjang;
    double lebar;
    double tinggi;

    Kotak2 (double p, double l, double t){
        panjang = p;
        lebar = l;
        tinggi = t;
    }

    double hitungVolume(){
        return (panjang*lebar*tinggi);
    }

    boolean sama (Kotak2 k){
        if ((k.panjang == this.panjang)&&(k.lebar == this.lebar)&&(k.tinggi == this.tinggi)){
            return true;
        }
        else {
            return false;
        }
    }
}
```

```java
public class DemoParamObjek1 {
    public static void main (String [] args){

        Kotak2 k1,k2,k3,k4;

        k1 = new Kotak2(4,3,2);
        k2 = new Kotak2(6,5,4);
        k3 = new Kotak2(4,3,2);
        k4 = new Kotak2(6,5,4);

        System.out.println("k1 == k2 : " +k1.sama(k2));
        System.out.println("k1 == k2 : " +k1.sama(k3));
        System.out.println("k1 == k2 : " +k2.sama(k4));
    }
}
```

put - Pertemuan6 (run)

```
run:
k1 == k2 : false
k1 == k2 : true
k1 == k2 : true
BUILD SUCCESSFUL (total time: 0 seconds)
```

Method dengan parameter objek k

Objek yang dilewatkan pada umumnya digunakan untuk proses inisialisasi nilai dari objek baru yang akan dibentuk

```java
public class Kotak3 {
    double panjang;
    double lebar;
    double tinggi;

    Kotak3(double p, double l, double t){
        panjang = p;
        lebar = l;
        tinggi = t;
    }

    Kotak3 (Kotak3 k){
        this.panjang = k.panjang;
        this.lebar = k.lebar;
        this.tinggi = k.tinggi;
    }

    double hitungVolume (){
        return (panjang*lebar*tinggi);
    }

    boolean sama (Kotak3 k){
        if ((k.panjang == this.panjang)&&(k.lebar == this.lebar)&&(k.tinggi == this.tinggi)){
            return true;
        }
        else {
            return false;
        }
    }
}
```

```java
public class DemoParamObjek2 {
    public static void main (String [] args){

        Kotak3 k1,k2;

        k1 = new Kotak3 (4,3,2);
        k2 = new Kotak3 (k1);

        System.out.println("k1 == k2 : " +k1.sama(k2));
        System.out.println("Volume k1 = " +k1.hitungVolume());
        System.out.println("Volume k2 = " +k1.hitungVolume());
    }
}
```

DemoParamObjek2 > main

put - Pertemuan6 (run)

```
run:
k1 == k2 : true
Volume k1 = 24.0
Volume k2 = 24.0
BUILD SUCCESSFUL (total time: 0 seconds)
```

Melewatkan Argumen di dalam Java

Di dalam java kita tidak dapat secara eksplisit melewatkan parameter berdasarkan nilai atau referensinya seperti yang dapat kita lakukan pada Pascal ataupun C++

Pass by Value : parameter berupa tipe data sederhana (int, cha, boolean, dll)
Pass by References : parameter berupa objek

```java
public class Contoh {
    void tambahSatu (int x){
        x+=1;
        System.out.println("Nilai x di dalam method : " +x);
    }
}
```

```java
public class DemoPassByValue {
    public static void main (String [] args){

        Contoh obj;

        obj = new Contoh();

        int a = 5;
        System.out.println("Nilai a sebelum pemanggilan method : " +a);

        obj.tambahSatu(a);

        System.out.println("Nilai a setelah pemanggilan method : " +a);
    }
}
```

Nilai dari argumen a akan disalin ke parameter x sehingga variabel x juga akan bernilai 5

DemoPassByValue > main

put - Pertemuan6 (run)

```
run:
Nilai a sebelum pemanggilan method : 5
Nilai x di dalam method : 6
Nilai a setelah pemanggilan method : 5
BUILD SUCCESSFUL (total time: 0 seconds)
```

Setiap perubahan nilai b yang terjadi pada referensi o tentu akan berpengaruh terhadap nilai b yang terdapat pada objek obj

```
public class Contoh2 {
    int b;

    Contoh2 (int b){
        this.b = b;
    }

    void tambahSatu(Contoh2 o){
        o.b +=1;
    }
}
```

```
public class DemoPassByReference {
    public static void main (String [] args){

        Contoh2 obj;

        obj = new Contoh2(5);

        System.out.println("Nilai obj.b sebelum pemanggilan method : " +obj.b);
        obj.tambahSatu(obj);
        System.out.println("Nilai obj.b sebelum pemanggilan method :" +obj.b);
    }
}
```

DemoPassByReference > main >

put - Pertemuan6 (run)

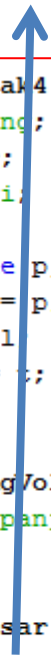
```
run:
Nilai obj.b sebelum pemanggilan method : 5
Nilai obj.b sebelum pemanggilan method :6
BUILD SUCCESSFUL (total time: 0 seconds)
```

Objek Sebagai Nilai Kembalian

Objek juga dapat diperankan sebagai nilai kembalian pada suatu method.

Pada situasi seperti ini, di dalam method kita perlu membentuk sebuah objek temporari yang akan dikembalikan

Objek temporari akan dibuang secara otomatis oleh garbage collector sehingga kita tidak perlu pusing untuk memikirkan cara pendealokasian objek tersebut.



```
public class Kotak4 {
    double panjang;
    double lebar;
    double tinggi;

    Kotak4(double p, double l, double t){
        panjang = p;
        lebar = l;
        tinggi = t;
    }

    double hitungVolume(){
        return (panjang*lebar*tinggi);
    }

    Kotak4 perbesar (int m){
        Kotak4 temp = new Kotak4 (panjang*m, lebar*m, tinggi*m);
        return temp;
    }
}
```

```
public class DemoKembalianObjek {
    public static void main (String [] args){
        Kotak4 kOriginal, kBaru;

        kOriginal = new Kotak4(4,3,2);
        kBaru = kOriginal.perbesar(2);

        System.out.println("Nilai pada objek kOriginal");
        System.out.println("panjang\t:" +kOriginal.panjang);
        System.out.println("lebar\t:" +kOriginal.lebar);
        System.out.println("tinggi\t:" +kOriginal.tinggi);
        System.out.println("volume\t:" +kOriginal.hitungVolume());

        System.out.println("Nilai pada objek kBaru");
        System.out.println("panjang\t:" +kBaru.panjang);
        System.out.println("lebar\t:" +kBaru.lebar);
        System.out.println("tinggi\t:" +kBaru.tinggi);
        System.out.println("volume\t:" +kBaru.hitungVolume());
    }
}
```

DemoKembalianObjek > main

put - Pertemuan6 (run)

```
run:
Nilai pada objek kOriginal
panjang :4.0
lebar   :3.0
tinggi  :2.0
volume  :24.0
Nilai pada objek kBaru
panjang :8.0
lebar   :6.0
tinggi  :4.0
volume  :192.0
BUILD SUCCESSFUL (total time: 1 second)
```

Rekursi

Rekursi adalah proses pemanggilan method oleh dirinya sendiri secara berulang.

TUGAS!

Buat dua contoh rekursi di dalam java