

Praktikum Pemrograman Berbasis Objek

Pertemuan 2

Review Time

1. Struktur Kode Java
2. Membuat Objek
3. Input dan Output
4. Statements dalam Java
5. Compile dan Run File Java
6. Built-in Methods dalam Class Java
7. Seputar assignment atau lainnya

Materi Pertemuan 2

Gambaran Materi Pembelajaran Hari Ini

Materi Pertemuan 2

01

Program Modular Java

Membuat program modular dengan file java lebih dari satu

03

Enkapsulasi

Mengenal konsep dan penggunaan enkapsulasi

02

Array of Object

Penggunaan array of object dalam java

04

Overloading

Mengenal konsep dan penggunaan overloading

01

Program Modular Java

Modularitas Java

Dalam bahasa pemrograman berorientasi objek apapun termasuk Java dan C++, konsep modularitas menjadi sebuah standar untuk membuat setiap modul bersifat individual. Hal ini memungkinkan komunikasi antar modul yang lebih mudah.

Library yang sering kita sertakan ketika membuat suatu program juga dapat disebut sebagai module. Maka dari itu, modularitas ini sangat berguna terutama untuk bahasa pemrograman berorientasi objek.

Di Java sendiri, terdapat sebuah keyword bernama `package` yang berfungsi untuk mengorganisir sekumpulan `class` dan `interface` dalam suatu direktori tertentu.

Package

```
package child1;

public class File1 {
    String data;

    public File1(String data) {
        this.data = data;
    }

    public String getData() {
        return this.data;
    }

    public void setData(String data) {
        this.data = data;
    }
}
```

← Deklarasi package

← Isi dari class

Package (lanjutan)

```
package child1;

public class File1 {
    String data;

    public File1(String data) {
        this.data = data;
    }

    public String getData() {
        return this.data;
    }

    public void setData(String data) {
        this.data = data;
    }
}
```

Contoh File1.java

```
package child2;

import child1.File1;

public class File2 {
    public static void main(String[] args) {
        File1 test = new File1("Tes package!");

        System.out.println(test.getData());
    }
}
```

Contoh File2.java

Package (lanjutan)

```
▼ parent
  ▼ child1
    J File1.class
    J File1.java
  ▼ child2
    J File2.class
    J File2.java
    J Test.java
```

Struktur folder dan file nya dapat dilihat di gambar disamping. Workspace yang sedang digunakan bernama `parent`, memiliki folder `child1` dan `child2` dengan isi file yang ada di slide sebelumnya.

Current directory harus berada di folder `parent` dan cara mengcompile nya dapat dilihat dibawah.

```
s_v\asprak\pbo\sandbox\parent
•> javac .\child2\File2.java; java child2/File2
Tes package!
```

02

Array of Object

Membuat Array dari Object

```
import child1.File1;

public class Test {
    public static void main(String[] args) {
        File1[] tes = new File1[5];

        for (int i = 0; i < 5; i++) {
            tes[i] = new File1("Ini tes ke-"+i);
        }

        for (File1 f : tes) {
            System.out.println(f.getData());
        }
    }
}
```

← Instansiasi array of object

← Pengisian array of object melalui constructornya

← Print isi dari array of object

Output dari kode →

```
s_v\asprak\pbo\sandbox\parent
• > javac Test.java; java Test
Ini tes ke-0
Ini tes ke-1
Ini tes ke-2
Ini tes ke-3
Ini tes ke-4
```

03

Enkapsulasi

Access Modifier

Berdasarkan apakah class dapat diakses oleh class lain

	class yang satu package	class beda package	subclass yang satu package	subclass beda package
public	✓	✓	✓	✓
protected	✓	✗	✓	✓
package (default)	✓	✗	✓	✗
private	✗	✗	✗	✗

Enkapsulasi

Enkapsulasi merupakan salah satu konsep fundamental PBO yaitu membungkus suatu kode dan data menjadi satu kesatuan.

Enkapsulasi ini dapat disebut juga sebagai **data hiding**. Salah satu keuntungan dari enkapsulasi adalah membuat suatu class hanya dapat diakses melalui class itu sendiri.

Syarat enkapsulasi meliputi :

1. Mendeklarasi variabel sebagai ``private`` .
2. Membuat method ``public`` ``setter`` dan ``getter`` untuk memodifikasi dan mengakses variabel tadi.

Enkapsulasi (lanjutan)

```
public class Rekening {  
    private float money;  
  
    public float retrieveMoney() {  
        // for retrieving money  
    }  
  
    public void setMoney(float amount) {  
        // for setting money value  
    }  
}
```

Apa yang terjadi jika money tidak diberi access modifier private / tidak menggunakan konsep enkapsulasi?

Enkapsulasi (contoh)

```
package child1;

public class File1 {
    private String data;

    public File1(String data) {
        this.data = data;
    }

    public String getData() {
        return this.data;
    }

    public void setData(String data) {
        this.data = data;
    }
}
```

```
package child2;

import child1.File1;

public class File2 {
    public static void main(String[] args) {
        File1 test = new File1("Tes package!");

        System.out.println(test.data);

        System.out.println(test.getData());
    }
}
```

Pemanggilan variabel `data` pada class lain akan menghasilkan **error**

Ketika variabel `data` dijadikan `private` ...

Manfaat Enkapsulasi

1. Data hiding

Program lebih *secure*.

Tidak perlu tahu bagaimana cara variable dibuat atau disimpan. Begitupun dengan method.

2. Flexibility

Bisa diatur apakah variable Read-only, atau Read-Write. Lebih maintainable.

3. Reusability

Method bisa dipanggil dimanapun berkali-kali secara efektif (dalam arti mudah dibaca dan digunakan)

04

Overloading

Overloading (constructor)

Overloading sebuah constructor berarti membuat banyak konstruktor yang memiliki parameter yang berbeda-beda

```
public Manusia {  
    private String nama;  
    private int umur;  
  
    public Manusia() {  
        nama = null;  
        umur = 0;  
    }  
  
    public Manusia(String nama, int umur) {  
        this.nama = nama;  
        this.umur = umur;  
    }  
  
    public Manusia(int umur, String nama) {  
        this.umur = umur;  
        this.nama = nama;  
    }  
}
```

← Constructor default (tanpa parameter)

← Constructor dengan paramater `String`, `int`

← Constructor dengan paramater `int`, `String`

Overloading (method)

Overloading method sama halnya dengan overloading constructor karena constructor merupakan (special) method.

```
public void setUmur(int umur) {  
    this.umur = umur;  
}  
  
public void setUmur(String umur) {  
    this.umur = Integer.parseInt(umur);  
}
```

← Method setter dengan parameter `int`

← Method setter dengan parameter `String`

Exercise!

Mari kita berlatih langsung agar cepat bisa

Exercise

Main.java

Buat sebuah program untuk mencetak selisih terdekat dari tiga angka. Sebuah angka diinstansiasi pada sebuah objek.

Angka pertama dalam argumen method akan selalu menunjuk angka terkecil. Angka kedua dalam argumen method akan selalu menunjuk angka terbesar, namun bisa tidak dituliskan. Jika tidak dituliskan, angka terbesar akan selalu bernilai `100`.

Kembalikan selisih terkecil dari angka tengah dengan angka terkecil dan/ atau angka terbesar.

INPUT

- a : int (angka tengah)
- lo : int (angka terkecil)
- hi : int (optional - angka terbesar, default `100`)

OUTPUT

int (selisih terkecil antara angka tengah dengan angka terkecil atau angka terbesar)

Sampel Input & Output 1

```
26
15
```

```
11
```

CONTOH

Sampel Input & Output 2

```
62
31 88
26
```

Snippets

Berhubung di mata kuliah PBO ini diminta untuk menguasai implementasi PBO di bahasa Java, C++ dan Python, maka kita sediakan repository snippets untuk ketiga bahasa tersebut dari setiap pertemuan praktikum.

Repository tersebut bisa diakses melalui tautan berikut :

[snippets-PBO-02](#)

Silahkan untuk kalian mencoba menjalankan dan mempelajari snippets yang disediakan pada repository tersebut. Selamat belajar!

Assignment!

Seperti biasa akan setiap selesai praktikum pasti ada tugas

Assignment 2 Soal 1

soal1/Test.java

Buat sebuah program **kalkulator** sederhana yang menggunakan 2 package. Package pertama bernama “io”, package kedua bernama “util”.

Dalam package “io” buat sebuah class yang memiliki method-method untuk **input dan output**.

Dalam package “util” buat sebuah class yang memiliki method-method untuk **operasi aritmatika** (tambah, kurang, kali, bagi, dan modular) bilangan **bulat**.

INPUT

num1: int (operan pertama)
op: char (operator “+”, “-”, “*”, atau “/”)
num2: int (operan kedua)

OUTPUT

Hasil perhitungan antara num1 dan num2 menggunakan op

Sampel Input

8 / 4

CONTOH

Sampel Output

= 2

Assignment 2 Soal 2

soal2/Test.java

Menggunakan class *Makanan* dari praktikum sebelumnya, buatlah array objek *Makanan* berukuran sesuai dengan input user, kemudian isi nama, harga, dan rating yang diberikan oleh user input. Kemudian outputkan keterangan setiap objek tersebut ke console.

INPUT

Untuk setiap makanan, ada input sebagai berikut secara konsekutif

- *nama* : string
- *harga* : int
- *rating* : string

CONTOH INPUT

```
Total makanan : 2
Nama makanan ke-1 : Mie
Harga : 10000
Rating : mantap
Nama makanan ke-2 : Nasi Goreng
Harga : 15000
Rating : kureng
```

OUTPUT

Untuk setiap makanan, ada output sebagai berikut secara konsekutif

- "*Makanan ke-i*", i adalah index
- "*Nama: nama*", nama adalah nama Makanan ke i
- "*Harga: harga*", harga adalah harga Makanan ke i
- "*Rating: rating*", rating adalah Rating Makanan ke i

CONTOH OUTPUT

```
Makanan ke-1
Nama : Mie
Harga : 10000
Rating : mantap

Makanan ke-2
Nama : Nasi Goreng
Harga : 15000
Rating : kureng
```

Assignment 2 Soal 3

soal3/Test.java

Buat class “Rekening” yang mempunyai default balance = 0, kemudian buatlah 3 (tiga) public method (selain constructor)

- `getBalance() : int`
- `deposit() : void`
- `withdraw() : void`

Class lain tidak boleh mengakses member yang lain kecuali constructor dan 3 method diatas.

Program berjalan dan menerima input terus menerus (*looping*) sampai user keluar dari program.

INPUT

`op` : int (1 = get, 2 = deposit, 3 = withdraw, 0 = exit)

`balance` : int (jika `op` = 1 atau 2)

OUTPUT

Jika `op` = 1 → Jumlah balance saat ini.

Jika `op` = 2 ATAU `op` = 3 → Tambah/Kurang balance .

Jika `op` = 0 → Keluar dari program.

Sampel Input

```
Operasi    : 2
Balance    : 2500
```

CONTOH

Sampel Output

```
Saldo      : 2500
```

Assignment 2 Soal 4

soal4/Test.java

Buat 2 buah method untuk menghitung luas 2 buah persegi. Gunakan overloading untuk tipe data Integer dan Float.

INPUT

Sisi 1 Persegi (A): int
Sisi 2 Persegi (A): int
Sisi 1 Persegi (B): float
Sisi 2 Persegi (B): float

CONTOH

OUTPUT

Luas Persegi (A): int
Luas Persegi (B): float

Sampel Input

```
Persegi A  
Sisi 1 : 2  
Sisi 2 : 10
```

```
Persegi B  
Sisi 1 : 6.5  
Sisi 2 : 4.5
```

Sampel Output

```
Luas Persegi A : 20  
Luas Persegi B : 29.25
```

Teknis Pengumpulan

Pengerjaan dan pengumpulan tugas akan dilakukan di **Github Classroom**

Kelas A:

Link Tugas Kelas A

Kelas B:

Link Tugas Kelas B

Accept assignment terlebih dahulu lalu link akun Github dengan slot nama yang sesuai di Github Classroom

Teknis Pengumpulan

Format setiap file `` . java `` didahulukan dengan Nama, NPM, Kelas, Tanggal, dan Deskripsi

Cara menambah comment di java

```
// untuk single line  
/* untuk multiple line */
```

Contoh Format

```
/*  
  Nama   : Jane Doe  
  NPM    : 99  
  Kelas  : A  
  Tanggal : 1 September 2021  
  Deskripsi : Class jawaban exercise-01 soal-01  
*/
```

Deadline Pengumpulan

Kelas A:

7 September 2025, 23:59 WIB

Kelas B:

10 September 2025, 23:59 WIB

Waktu yang dilihat adalah waktu last commit.

Jika ada yang commit melewati deadline walaupun sudah commit sebelumnya akan dianggap telat

Terima Kasih!

Do you have any questions? Please use respective class discussion channel on Discord.

Semangat terus menjalani kuilahnya!! 🔥 🔥 🔥