

Pemrograman Berorientasi Objek

Pengantar



Silabus:

- Mata kuliah ini memberikan pemahaman atas konsep dan keahlian dalam membangun program dengan paradigma berorientasi objek dengan bahasa pemrograman Java.

Silabus: Kompetensi

- Mahasiswa memahami prinsip OOP
- Mahasiswa dapat membedakan Class vs Objek
- Mahasiswa memahami dan dapat mengimplementasikan komponen class: atribut dan method
- Mahasiswa memahami konsep dan dapat mengimplementasikan inheritance dan polymorphism
- Mahasiswa memahami dan dapat mengimplementasikan interface dan abstract class
- Mahasiswa memahami ,merancang dan mengimplementasikan konsep OOP

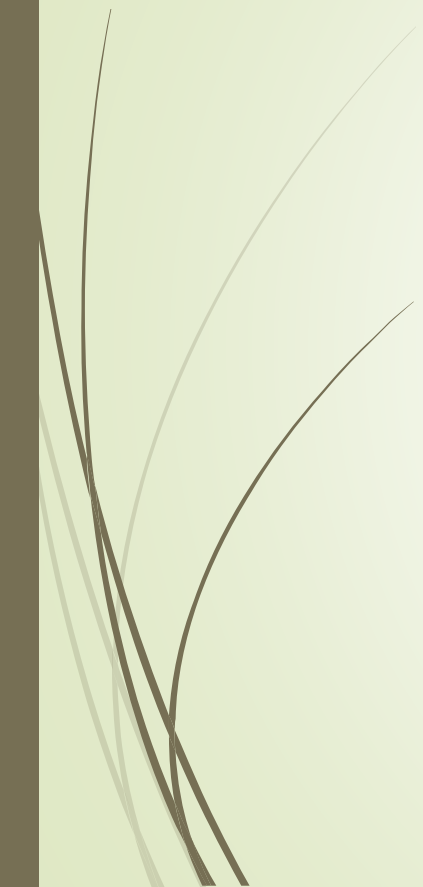


Aturan Umum

- Susulan UTS atau UAS
- Nilai nol di salah satu komponen: indeks E
- Semua bentuk kecurangan: sanksi indeks E
 - Termasuk menolong terjadinya kecurangan
 - Termasuk kecurangan dalam tubes atau tugas lainnya: E untuk semua anggota



Referensi Kuliah

- ➡ **Yudi Wibisono, Object Oriented Programming di Java, Diktat**
 - ➡ **Yudi Wibisono, Tutorial Bahasa Java. Bagian 1 (Dasar-Dasar Java), Diktat**
 - ➡ **Core Java Volume I--Fundamentals (9th Edition) (Core Series)**
- 

Apa itu Object Oriented?

Memandang sistem sebagai kumpulan kelas (class) dan objek.

Contoh: Budi dan Ani adalah objek dari class Mahasiswa.

Class adalah abstraksi dari objek. Objek adalah perwujudan (instance) dari class.

Class dapat berbentuk benda: mahasiswa, dosen

Class dapat sesuatu yang abstrak: jadwal, kontrak

Class

- Gabungan dari DATA (state) dan AKSI (behaviour)
- **Data** pada Class Mahasiswa:
 - NIM, NAMA, ALAMAT dsb.
- **Aksi** pada Class Mahasiswa:
 - Kontrak, Cuti, ApakahMhsAktif?

Apa data dan aksi untuk class Ruangankuliah?

Class vs Objek

Class Mahasiswa:

NIM

NAMA

ALAMAT

KONTRAK

CUTI

SUD AHLULUS?

Objek Mahasiswa:

NIM: 0001

NAMA: BUDI MARTAMI

ALAMAT: GERLONG

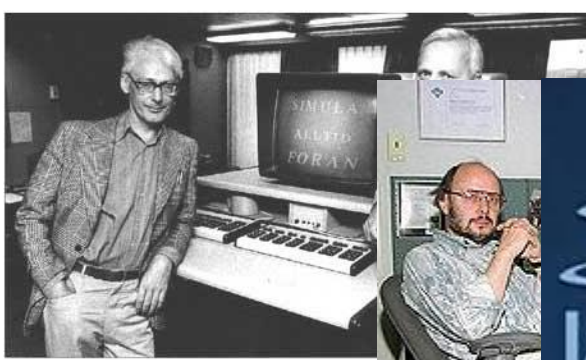
KONTRAK

CUTI: TRUE

SUD AHLULUS?

Sejarah OOP

- 1960an: SIMULA
- 1980an: C++
- 1990an: JAVA



Dahl and Nygaard at the time of Simul



Sekarang: C#, PHP, Ruby, Python, JScript, Objective C, Swift, R, Scala, Go, Delphi

Seperti pada kuliah pemrograman lain, bahasa berbeda tapi konsep akan tetap sama

Mengapa perlu belajar OOP?



Mengapa perlu Belajar OOP

- Mempermudah pengembangan software
 - Class dapat di-reuse → duplikasi ditekan
 - Data dan aksi terintegrasi → modular
 - Information hiding (class sebagai black box)
- Saat ini hampir semua bahasa mendukung OOP

Mengapa JAVA?

- Bahasa yang sudah matang. Kinerja tinggi, kaya fitur, kaya library.
- Banyak digunakan (Bahasa no 1 pada Tiobe Index 2015)
- Digunakan untuk mengembangkan Android App.
- Open Source (tapi masih dipegang Oracle)
- Cross platform (Linux, Windows, Mac)

Kelemahan: cukup kompleks

Integrated Development Environment (IDE)

- Membantu developer membuat aplikasi. Berisi editor, compiler, sampai debugger.
- Bisa saja menggunakan Notepad lalu command line, tapi..?
- IDE yang paling umum untuk Java:
 - Eclipse
 - Netbeans (Oracle)
 - IntelliJ

Class dan Objek di Java

- ➡ Contoh kasus, buat sistem informasi perkuliahan
- ➡ Apa saja kelasnya?

Kelas Mahasiswa

Java:

- Satu file satu class
- Nama kelas diawali huruf kapital

File: Mahasiswa.java

Isi minimal:

```
public class Mahasiswa {  
}
```

Mahasiswa.java → Data / Atribut

```
public class Mahasiswa {  
    int angkatan;  
    double ipk;  
    String tempatLahir;  
    String nim;  
    String nama;  
    boolean isSudahLulus;  
}
```

cat: nama variabel diawali huruf kecil (kesepakatan)

Mahasiswa.java → Data + Aksi/fungsi/method

```
public class Mahasiswa {
```

```
    String nim;
```

```
    String nama;
```

```
    void printData() {
```

```
        System.out.println("Nama: "+nama);
```

```
        System.out.println("NIM: "+nim);
```

```
    }
```

```
}
```

Data

Fungsi

Encapsulation: data dan fungsi digabung

Encapsulation

- Data + Method digabung dalam satu class
- Data encapsulation: menyembunyikan data (internal state) dan membutuhkan interaksi melalui method objek
- Kenapa ?

Encapsulation

- Data + Method digabung dalam satu class
- Data encapsulation: menyembunyikan data (internal state) dan membutuhkan interaksi melalui method objek
- Kenapa ?
 - Mempermudah reuse (library)
 - Mempermudah debugging

OBJEK Mahasiswa

```
public class Mahasiswa {  
    String nim;  
    String nama;  
  
    void printData() {  
        System.out.println("Nama:" + nama);  
        System.out.println("NIM:" + nim);  
    }  
  
    public static void main(String[] args) {  
        Mahasiswa mhs; //objek mhs dengan class mahasiswa  
        mhs = new Mahasiswa(); //buat instance mahasiswa  
        mhs.nim = "123";  
        mhs.nama = "Budi Martami";  
        mhs.printData();  
    }  
}
```

Mahasiswa: IP per semester dan hitung IPK

```
public class Mahasiswa {  
    private double[] ipSem = new double[9]; //array  
        double ip per semester  
    int jumSem;  
  
    //isi ip per semester  
    void tambahIpSem(int sem, double ip) {  
        ipSem[sem] = ip;  
    }  
    //hitung IPK berdasarkan rata2 ip per sem  
    double hitungIPK() {  
        double ipk = 0;  
        double total = 0;  
        for (int i=1; i<=8; i++) {  
            total = total + ipSem[i];  
        }  
        ipk = total / jumSem;  
        return ipk;  
    }  
}
```

Mahasiswa: IP per semester dan hitung IPK (lanj)

```
public static void main(String[] args) {  
    Mahasiswa mhs;  
    mhs = new Mahasiswa();  
    mhs.tambahIpSem(1, 3.0);  
    mhs.tambahIpSem(2, 4.0);  
    mhs.jumSem = 2;  
    double ipk = mhs.hitungIPK();  
    System.out.println("IPK="+ipk);  
}
```

Information Hiding

Perhatikan private:

```
private double [] ipSem = new double [9]; /
```

Diset private, ipSem tidak dapat diakses oleh objek lain. Jika diisi harus melalui method tambahIpSem: mhs.tambahIpSem(1, 3.0);
Kenapa tidak langsung? misalnya mhs.ipSem[1] = 3.0

?