Fakultas Rekayasa Industri

ESD Laboratory

ENTERPRISE APPLICATION DEVELOPMENT

{ OBJECT ORIENTED PROGRAMMING }

MODUL 1

KONSEP OOP & PROCEDURAL LANGUAGE DI PYTHON DAN JAVA BASIC

Prepared by :
EAD LABORATORY

ENTERPRISE APPLICATION DEVELOPMENT

Asisten Praktikum

ADNR
Abid Naufal Rafif

DEKA
Adrian Daniel K .

RAIN
Amirul Rizqi N .

PIPI
Avicenna Nabighani

GNOY
Farid Ghani

HAQI
Dhifulloh Dhiya U .

GYRO
Katon Bagaskoro

GUTS
Endra Lazuardi A R .

KRSY
Kresna Mukti W .

MAUL
Maulida Afifi Utami

MAJA
Muadzam Haqqani

KYRA
Kirana Adira Syifa

PDLY
Muhammad Fadli M .

MUZU
Muhammad Zulfikri

ICEA
Naisya Najmi

PALS
Naufal Akmal R .

HALF
Sahal Fajri

WYEN
Vanessa Wiyen C .

PERATURAN

1. Setiap peserta praktikum harus datang tepat waktu sesuai dengan jadwal. Toleransi keterlambatan hadir 15 menit. Jika melebihi batas waktu tersebut dan tidak dapat memberikan bukti alasan keterlambatan maka praktikan tidak diperkenankan mengikuti praktikum.

2. Perizinan Praktikum :

a. Izin berkaitan dengan Sakit atau Kemalangan, maka praktikan dapat memberikan surat perizinan kepada pihak Komisi Disiplin maksimal 3 hari setelah jadwal (shift) praktikum.

b. Izin lomba atau penugasan institusi tidak berlaku apabila tidak terdapat bukti dispensasi dari Igracias. NB : screenshot dispensasi dari igracias wajib dilampirkan.

3. Seragam Praktikum :

a. Mahasiswa wajib menggunakan celana bahan hitam (bukan chino atau jeans) pada saat praktikum.

b. Mahasiswi wajib menggunakan rok hitam/biru gelap panjang tidak ketat pada saat praktikum.

c. Dresscode praktikum (Mengikuti Peraturan Telkom)

I. Senin : Menggunakan kemeja merah telkom atau kemeja putih polos

II. Selasa s/d Rabu : Menggunakan kemeja putih polos

III. Kamis s/d Sabtu : Menggunakan kemeja formal berkerah (bukan kerah sanghai dan

IV. Jumat : Menggunakan kemeja/ baju batik bukan outer.

d. Jika terdapat kendala dalam baju seragam maka diperbolehkan mengganti menggunakan kemeja putih.

e. Membuka sepatu saat memasuki ruangan lab.

PERATURAN

4. Peraturan Pengerjaan

a. Studi Kasus dikerjakan secara individu.

b. Jawaban tidak boleh sama dengan setiap individu.

c. Hasil pengerjaan di push ke repository github tiap individu.

d. Submit hasil pengerjaan berupa file format PDF dan link github ke LMS tiap individu.

e. Format Pengumpulan Hasil Pengerjaan

I. Format nama repository :

● OOP-KODEASISTEN-NAMAPENDEK-NIM

Contoh : OOP-PIPI-AVICENNA-1202223243

II. Format nama file Jurnal PDF:

● OOP_KODEASISTEN_NAMAPENDEK_NIM_SSMODULX

Contoh : OOP_PIPI_AVICENNA_1202223243_SSMODUL1

III. Format nama file Tugas Pendahuluan PDF:

● OOP_KODEASISTEN_NAMAPENDEK_NIM_SSTPX

Contoh : OOP_PIPI_AVICENNA_1202223243_SSTP1

IV. Format nama folder per modul github

● Jurnal: MODULX_NAMAPENDEK

Contoh: MODUL1_AVICENNA

● Tugas Pendahuluan: TPMODULX_NAMAPENDEK

Contoh: TPMODUL1_AVICENNA

f. Salah penamaan pada file pengerjaan nilai modul akan dipotong sebesar 10%

g. Terlambat mengumpulkan file pengerjaan nilai modul akan dipotong sebesar 20%

h. Segala alat komunikasi dikumpulkan di loker yang tersedia dalam ruangan.

i. Jika ada perangkat praktikum yang bermasalah dapat menghubungi asprak yang bertugas.

j. Segala bentuk kecurangan dan plagiarisme akan diproses ke komisi disiplin dan nilai akhir modul menjadi 0.

CONTENT

MODUL 1 OOP

Konsep OOP

Pemrograman Berbasis Objek (OOP) adalah paradigma pemrograman yang berfokus pada konsep "objek" yang merepresentasikan entitas dunia nyata. Dalam OOP, kita membuat **kelas**, yang seperti blueprint atau template. Dari blueprint ini, kita bisa membuat **objek** yang memiliki sifat (data) dan kemampuan (fungsi). Misalnya, jika kita membuat kelas "Mobil," maka objek dari kelas ini bisa punya sifat seperti warna dan merek, serta kemampuan seperti berjalan atau berhenti. Prinsip utama OOP adalah memudahkan pengelolaan program dengan membuat kode lebih teratur, mudah dipakai ulang, dan lebih mudah dipahami karena mengikuti pola yang mirip dengan kehidupan sehari-hari.

Keunggulan Java dalam Object Oriented Programming

Java dipilih sebagai bahasa pemrograman yang digunakan untuk mempelajari konsep OOP karena memiliki dukungan yang kuat dan implementasi yang terstruktur untuk OOP serta prinsip-prinsip seperti enkapsulasi. Java menyediakan fitur-fitur utama seperti class, objek, inheritance, polimorfisme, dan access modifiers seperti private, protected, dan public, yang memberikan kontrol ketat terhadap aksesibilitas atribut dan metode dalam suatu kelas. Hal ini membuat Java sangat cocok untuk memahami dan menerapkan konsep OOP secara mendalam, karena setiap elemen OOP dapat diimplementasikan dengan cara yang konsisten dan terstruktur. Di sisi lain, meskipun Python juga mendukung OOP, pendekatan Python lebih fleksibel dan longgar, terutama dalam hal enkapsulasi. Python tidak memiliki modifier private secara ketat, sehingga kontrol akses terhadap atribut dan metode lebih longgar dan dapat diakses dari luar kelas jika diinginkan. Fleksibilitas ini, meskipun memudahkan untuk menulis kode dengan lebih cepat, dapat mengurangi pemahaman mendalam terhadap konsep seperti enkapsulasi yang lebih ketat di Java. Oleh karena itu, Java lebih sering dipilih untuk mempelajari

OOP, karena memberikan landasan yang lebih kuat dalam penerapan prinsip-prinsip OOP secara disiplin.

Perbedaan OOP dan Procedural Language

Perbedaan antara Object-Oriented Programming (OOP) dan bahasa pemrograman prosedural terletak pada cara keduanya mengorganisir kode. OOP berfokus pada objek, yaitu entitas yang mewakili sesuatu dari dunia nyata yang memiliki sifat (atribut) dan kemampuan (metode), sedangkan bahasa prosedural berfokus pada serangkaian instruksi atau langkah-langkah yang dieksekusi secara berurutan.

Contoh Procedural Language:

```java
// Program untuk menghitung luas lingkaran dengan pendekatan prosedural
public class Main {
    // Fungsi untuk menghitung luas lingkaran
    public static double hitungLuasLingkaran(double jariJari) {
        double luas = 3.14 * (jariJari * jariJari);
        return luas;
    }

    public static void main(String[] args) {
        // Contoh penggunaan fungsi hitungLuasLingkaran
        double jariJari = 5;
        double luas = hitungLuasLingkaran(jariJari);
        System.out.println("Luas lingkaran dengan jari-jari " + jariJari + " adalah " + luas);
    }
}
```

Contoh OOP:

```java
// Membuat kelas Circle untuk menghitung luas lingkaran
class Circle {
    private double radius;

    // Konstruktor untuk inisialisasi radius
    public Circle(double radius) {
        this.radius = radius;
    }

    // Method untuk menghitung luas lingkaran
    public double hitungLuas() {
        return 3.14 * (radius * radius);
    }
}

public class Main {
    public static void main(String[] args) {
        // Membuat objek dari kelas Circle
        Circle lingkaran = new Circle(5);

        // Menggunakan method untuk menghitung luas
        double luas = lingkaran.hitungLuas();
        System.out.println("Luas lingkaran dengan jari-jari " + lingkaran.radius + " adalah " + luas);
    }
}
```

Object dan Class

Dalam pemrograman berorientasi objek (OOP), object dan class adalah konsep utama. Class merupakan cetakan atau template yang mendefinisikan atribut dan perilaku yang akan dimiliki oleh suatu objek. Bayangkan class seperti blueprint dari rumah; ia menetapkan bagaimana bentuk rumah tersebut seharusnya dibangun, termasuk jumlah pintu, jendela, dan material yang digunakan. Sedangkan object adalah instansiasi dari class. Ketika class diimplementasikan dalam program, maka terbentuklah object, yang merupakan rumah sebenarnya berdasarkan blueprint tadi. Object memiliki data atau atribut spesifik dan perilaku atau metode yang telah didefinisikan oleh class. Dengan kata lain, class mendefinisikan struktur dan perilaku, sementara object adalah representasi nyata yang dapat digunakan dan dimanipulasi dalam program.

- Deklarasi Class

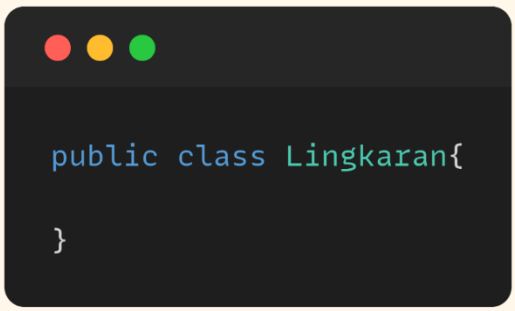
Pada pemrograman Java, pendeklarasian kelas penamaan kelas harus

sama dengan nama file java dan diawali dengan huruf kapital, dapat

dilakukan dengan menggunakan syntax sebagai berikut:

[modifier] class Class_identifier

Contoh Pendeklarasian Class:



- Deklarasi Variable

Dalam Java, variabel biasanya dideklarasikan di dalam kelas dan dikenal sebagai atribut atau *fields* dari kelas tersebut. Atribut ini menggambarkan karakteristik yang dimiliki oleh objek yang akan dibuat dari kelas tersebut. Proses deklarasi variabel melibatkan penentuan tipe data, nama variabel, dan pilihan untuk memberikan nilai awal (inisialisasi). Sintaks umum deklarasi dan inisialisasi variabel adalah sebagai berikut:

[modifier] data_type identifier [=value];

Penjelasan:

- **Modifier**: Merupakan kata kunci untuk mendefinisikan makna dari suatu variable, method, atau class. Seperti private, protected, public, dll.
- **Private**: Hanya dapat diakses di dalam kelas yang sama. Anggota tersebut tidak dapat dilihat oleh kelas lain, termasuk subclass (kelas turunan).
- **Protected**: Dapat diakses di dalam kelas yang sama, serta di dalam subclass (kelas turunan) dari kelas tersebut. Namun, anggota tersebut tidak dapat diakses di luar hierarki kelas tersebut.
- **Public**: Dapat diakses dan dimodifikasi dari mana saja, baik di dalam kelas maupun di luar kelas.
- **Data_type**: Merupakan kata kunci tipe data dari variabel, misalnya: int, float, double, string, dll.
- **Identifier**: Merupakan nama variabel (Nama variable tidak boleh ada spasi).
- **Value**: Merupakan nilai awal dari variabel, bersifat opsional.

Contoh deklarasi variable:

```java
public class ContohVariabel {

    public int angkaInt = 100000;
    public float angkaFloat = 10.5f;
    public double angkaDouble = 10.5555;
    public char karakter = 'A';
    public boolean isAvailable = true;
    public String nama = "Java Programming";
    public int jumlah; //tanpa nilai awal
}
```

Method

Dalam Java, method adalah sebuah wadah dalam kelas yang berisi baris-baris kode untuk melakukan suatu fungsi atau tugas tertentu. Method ini membantu mengenkapsulasi proses; misalnya, jika ada objek Mobil yang dapat berjalan, maka berjalan adalah method dari objek tersebut. Method dibagi menjadi dua jenis yaitu method void, yang tidak mengembalikan nilai, dan method return, yang mengembalikan suatu nilai setelah eksekusi.

Method Void

Method void adalah metode yang digunakan untuk memberikan nilai pada suatu attribute, object, dan lain - lain. Method void memiliki ciri ciri memiliki kata “void” dan “this”.
Contoh Method Void:

```java
public class Siswa {

    private String nama;
    private int umur;

    // Method void untuk mengatur nama
    public void setNama(String namaBaru) {
        nama = namaBaru;
    }

    // Method void untuk mengatur umur
    public void setUmur(int umurBaru) {
        umur = umurBaru;
    }
}
```

Method Return

Method Return adalah konsep dalam pemrograman di mana sebuah metode mengembalikan nilai setelah melakukan suatu proses. Nilai yang dikembalikan ini bisa berupa data yang dihasilkan oleh metode tersebut, seperti angka, teks, atau objek lainnya. Tidak seperti metode yang menggunakan "void" yang tidak mengembalikan apa-apa, metode dengan return selalu mengembalikan suatu nilai dengan menggunakan kata kunci "return".

Contoh Method Return

```java
public class Calculator {
    public int add(int a, int b) {
            int result = a + b;
            return result;
        }

    public static void main(String[] args) throws Exception {
        Calculator calc = new Calculator();

        int sum = calc.add(5, 3);
        System.out.println("Hasil penjumlahan: " + sum);
    }
}
```

Output

```
Hasil penjumlahan: 8
```

Method Static

Method Static adalah method yang dideklarasikan dengan keyword **static** di dalam suatu class yang dimana method ini bisa dipanggil tanpa perlu membuat objek dari class tersebut.

Contoh Method Static

[modifiers] static return_type method_identifier (parameter) {method_body; }

```
public class Calcu {
    public static int multiply(int a, int b) {
        return a * b;
    }

    public static void main(String[] args) {
        int result = Calcu.multiply(4, 5); // Memanggil method static tanpa membuat objek
        System.out.println("Hasil perkalian: " + result);
    }
}
    }
}
```

Output

```
Hasil perkalian: 20
```

Method Setter Getter

Method Setter Getter adalah metode di dalam pemrograman berorientasi objek yang digunakan untuk menetapkan (set) dan mengambil (get) nilai dari variabel private dalam sebuah class. Metode setter digunakan untuk mengubah nilai variabel, sedangkan metode getter digunakan untuk mengambil nilai variabel. Biasanya, variabel dideklarasikan dengan akses modifier private, sehingga tidak bisa diakses langsung dari luar class. Sebagai gantinya, digunakan setter dan getter untuk mengontrol akses ke variabel tersebut.

Contoh Method Setter Getter

```java
public class bio {
    private String name;
    private int age;

    // Getter untuk variabel 'name'
    public String getName(){
        return name; // Mengembalikan nilai name
    }

    // Setter untuk variabel 'name'
    public void setName(String name){
        this.name = name; // Menetapkan nilai name
    }

    // Getter untuk variabel 'age'
    public int getAge(){
        return age; // Mengembalikan nilai age
    }

    // Setter untuk variabel 'age'
    public void setAge(String age){
        this.age = age; // Menetapkan nilai age
    }
}
```

Scanner pada Java

Scanner adalah class di Java yang digunakan untuk membaca input dari berbagai sumber, seperti **keyboard** (input dari pengguna), **file**, dan **stream input lainnya**. Scanner mempermudah program untuk menerima input yang bervariasi, seperti angka, string, atau baris teks. Untuk menggunakan Scanner, Anda perlu mengimpor paket java.util.Scanner. Metode yang biasanya digunakan :

a. nextInt(): Membaca input integer.
b. nextDouble(): Membaca input tipe double.
c. nextLine(): Membaca seluruh baris teks.
d. next(): Membaca satu kata (hingga spasi atau enter).

Contoh penggunaan Scanner

```java
import java.util.Scanner;

    public class Biodata {
    public static void main(String[] args) {
        // Membuat objek Scanner untuk membaca input dari keyboard
        Scanner scanner = new Scanner(System.in);

        // Membaca input berupa string
        System.out.print("Masukkan nama Anda: ");
        String name = scanner.nextLine();

        // Membaca input berupa integer
        System.out.print("Masukkan usia Anda: ");
        int age = scanner.nextInt();

        // Menampilkan hasil input pengguna
        System.out.println("Halo, " + name + "! Anda berusia " + age + " tahun.");

        // Menutup scanner
        scanner.close();
    }
}
```

Output

```
Masukkan nama Anda: EAD Laboratory
Masukkan usia Anda: 10
Halo, EAD Laboratory! Anda berusia 10 tahun.
```

Create Object (Constructor)

Constructor adalah method khusus dalam suatu class di Java yang digunakan untuk menginisialisasi objek ketika objek tersebut dibuat. Constructor memiliki nama yang sama dengan nama class dan tidak memiliki tipe kembalian (return type), bahkan void. Constructor dipanggil secara otomatis saat kita membuat objek dari class.

```java
public class Person {
    String name;
    int age;

    public Person() {
        name = "Andi";
        age = 10;
    }

    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public void display() {
        System.out.println("Name: " + name + ", Age: " + age);
    }

    public static void main(String[] args) {
        Person person1 = new Person();
        Person person2 = new Person("Budi", 12);
        person1.display();
        person2.display();
    }
}
```

Output

```
Name: Andi, Age: 10
Name: Budi, Age: 12
```

1. Array

Array adalah struktur data yang digunakan untuk menyimpan sekumpulan nilai yang memiliki tipe data yang sama. Setiap elemen dalam array diakses menggunakan indeks (posisi elemen dalam array), dan indeks tersebut dimulai dari angka nol (0). Array memungkinkan penyimpanan banyak nilai dalam satu variabel, sehingga lebih efisien dalam pengelolaan data yang sejenis dibandingkan dengan mendeklarasikan banyak variabel secara terpisah.

2. ArrayList

ArrayList adalah salah satu implementasi dari interface List di Java yang termasuk dalam paket java.util. Kelas ini digunakan untuk menyimpan sekumpulan data yang bersifat dinamis. Berbeda dengan array biasa yang ukurannya tetap setelah dideklarasikan, ukuran ArrayList dapat berubah-ubah sesuai dengan penambahan atau penghapusan elemen.

Ciri-ciri ArrayList:

- Dinamis: Ukurannya bisa bertambah atau berkurang.
- Generik: Bisa digunakan untuk menyimpan tipe data tertentu dengan menggunakan parameter tipe data (ArrayList<Type>).
- Mudah digunakan: Operasi seperti menambah, menghapus, dan mencari elemen sangat sederhana.

Contoh Penggunaan ArrayList:

```java
import java.util.ArrayList;

public class ContohArrayList {
    public static void main(String[] args) {
        ArrayList<String> daftarNama = new ArrayList<>();
        daftarNama.add("Andi");
        daftarNama.add("Budi");
        daftarNama.add("Citra");

        System.out.println("Daftar Nama: " + daftarNama);
        daftarNama.remove("Budi");
        System.out.println("Daftar Nama Setelah Penghapusan: " + daftarNama);
    }
}
```

Output:

```
Daftar Nama: [Andi, Budi, Citra]
Daftar Nama Setelah Penghapusan: [Andi, Citra]
```

3. Looping (Perulangan)

Looping adalah mekanisme dalam pemrograman yang memungkinkan kita untuk menjalankan suatu blok kode berulang kali berdasarkan kondisi tertentu. Java menyediakan beberapa jenis loop seperti For Loop dan While Loop.

a. For Loop

For Loop digunakan ketika kita mengetahui berapa kali perulangan harus dijalankan. Biasanya terdiri dari tiga bagian: inisialisasi, kondisi, dan perubahan.

Contoh For Loop:

```java
public class ContohForLoop {
    public static void main(String[] args) {
        for (int i = 0; i < 5; i++) {
            System.out.println("Angka: " + i);
        }
    }
}
```

Output:

```
Angka: 0
Angka: 1
Angka: 2
Angka: 3
Angka: 4
```

b. While Loop

While Loop digunakan ketika kita tidak mengetahui jumlah pasti perulangan dan ingin loop berjalan sampai kondisi tertentu terpenuhi.

Contoh While Loop:

```java
public class ContohWhileLoop {
    public static void main(String[] args) {
        int i = 0;
        while (i < 5) {
            System.out.println("Angka: " + i);
            i++;
        }
    }
}
```

Output:

```
Angka: 0
Angka: 1
Angka: 2
Angka: 3
Angka: 4
```

c. ArrayList Loop

ArrayList Loop adalah teknik untuk mengiterasi atau melakukan perulangan pada elemen-elemen yang terdapat dalam sebuah ArrayList di Java. Java menyediakan beberapa cara untuk melakukan loop atau perulangan pada ArrayList, yaitu:

a. For-Each Loop

For-Each Loop adalah cara sederhana dan lebih bersih untuk mengakses semua elemen dalam sebuah ArrayList. Dalam perulangan ini, kita tidak perlu memikirkan indeks atau ukuran ArrayList. For-Each Loop digunakan ketika kita ingin membaca atau mengakses setiap elemen tanpa mengubah atau membutuhkan akses terhadap indeks elemen tersebut.

Sintaks:

```
for (Type element : ArrayList) {
    // kode yang akan dijalankan untuk setiap elemen
}
```

Contoh:

```
import java.util.ArrayList;

public class ContohForEachLoop {
    public static void main(String[] args) {
        ArrayList<String> daftarNama = new ArrayList<>();
        daftarNama.add("Andi");
        daftarNama.add("Budi");
        daftarNama.add("Citra");

        for (String nama : daftarNama) {
            System.out.println(nama);
        }
    }
}
```

Output:

```
Andi
Budi
Citra
```

b. For index Loop

For Index Loop memungkinkan kita mengakses elemen-elemen ArrayList dengan menggunakan indeks. Loop ini berguna ketika kita perlu mengakses elemen berdasarkan posisi indeksnya, atau ketika kita perlu melakukan perubahan pada elemen tersebut.

Sintaks:

```
for (int i = 0; i < ArrayList.size(); i++) {
    // kode yang akan dijalankan untuk setiap elemen
}
```

Contoh:

```
import java.util.ArrayList;

public class ContohForIndexLoop {
    public static void main(String[] args) {
        ArrayList<Integer> angka = new ArrayList<>();
        angka.add(10);
        angka.add(20);
        angka.add(30);

        for (int i = 0; i < angka.size(); i++) {
            System.out.println("Elemen ke-" + i + ": " + angka.get(i));
        }

        for (int i = 0; i < angka.size(); i++) {
            angka.set(i, angka.get(i) * 2);
        }

        System.out.println("ArrayList setelah modifikasi: " + angka);
    }
}
```

Output:

```
Elemen ke-0: 10
Elemen ke-1: 20
Elemen ke-2: 30
ArrayList setelah modifikasi: [20, 40, 60]
```

Fakultas Rekayasa Industri

ESD Laboratory

ENTERPRISE APPLICATION DEVELOPMENT

“Every challenge is an opportunity to grow, don't be afraid to step up.”

find us on :

@eadlaboratory