

Nama : Nazwa Salsyabilla Ramadhani

Nim : 9241250016

Kelas : PSIK 21A

Mata kubah : Pemrograman berorientasi objek

1. Jelaskan bagaimana prinsip encapsulation, inheritance, polymorphism, dan abstraction saling mendukung dalam membangun sistem perangkat lunak yang mudah dikembangkan dan dipelihara. Sertakan contoh analogi dalam kehidupan nyata dalam masing-masing konsep.

- ↳ • Encapsulation adalah proses menyembunyikan detail implementasi dari suatu objek dan hanya memperhatikan antarmuka yang diperlukan. Dalam Java, ini biasanya dilakukan dengan menjadikan atribut bersifat private dan menyediakan getter serta setter method untuk mengaksesnya. Contohnya: remote televisi.
- Inheritance ( pewarisan ) memungkinkan sebuah class mewarisi properti dan method dari class lain. Ini sangat berguna untuk menghindari duplikasi kode dan meningkatkan keteraturan sistem. Contoh: Seorang anak mewarisi sifat dan orang tuanya seperti warna rambut dan bentuk wajah.
- Polymorphism (polimorfisme) memungkinkan satu interface untuk memiliki banyak implementasi yang berbeda, ini terjadi pada method overriding dan interface. Contoh: tombol "print" yang bisa mencetak file dokumen, gambar, maupun PDF.
- Abstraction (abstraksi) adalah proses menyembunyikan kompleksitas dan hanya menampilkan informasi penting dari sebuah objek. Hal ini memudahkan developer fokus pada fungsi utama tanpa harus melihat detail teknis. Contoh: Setir mobil hanya digunakan untuk mengemudi tanpa mengetahui mekanisme kega mesin di dalamnya.

2. Apa kelebihan menggunakan Java versi terbaru (Java 21) dibanding versi sebelumnya dalam konteks pengembangan berbasis OOP? Berikan minimal dua fitur modern Java 21 dan jelaskan bagaimana fitur tersebut menyederhanakan pengembangan aplikasi OOP.

↳ Java 21 terbaru dari Java membawa berbagai peningkatan modern yang memberikan pengembangan aplikasi berbasis OOP. Dibanding versi sebelumnya, Java 21 menawarkan sintaks yang lebih ringkas, efisien dan aman.

Contoh fitur:

- Record patterns (Fitur final - JEP 290). Memungkinkan kita untuk langsung mendekomposisi objek record dalam satu langkah matching, khususnya dalam if atau switch.
- Sealed Classes (Fitur final - JEP 409) membatasi subclass apa saja yang dapat meng-extend sebuah class atau mengimplementasi interface tertentu.



### 3. Jawaban soal no 3

#### ↳ Class

- Blueprint atau cetak biru untuk membuat objek
- Menentukan struktur dan perintah
- Tidak nyata, hanya definisi
- Seperti rancangan bangunan

Contoh program

#### 1 Class Mahasiswa

```
public class Mahasiswa {
```

```
    String nama;
```

```
    String nim;
```

```
    String jurusan;
```

```
    public Mahasiswa (String nama, String nim, String jurusan) {
```

```
        this.nama = nama;
```

```
        this.nim = nim;
```

```
        this.jurusan = jurusan;
```

```
    public void tampilkanData () {
```

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

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

```
        System.out.println ("Jurusan : " + jurusan);
```

```
}
```

```
}
```

#### 2. Object dan class Mahasiswa

```
public class Main {
```

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

```
        Mahasiswa mahasiswa1 = new Mahasiswa ("Nazwa", "9241250016", "Iku Komputer");
```

```
        mahasiswa1.tampilkanData ();
```

```
}
```

```
}
```

#### 4. Jawaban soal no 4

- ↳ • Mengadakan atribut balance bersifat privat, kemudian menyediakan metode khusus (seperti deposit dan withdraw) yang berfungsi untuk menambah atau mengurangi saldo, dengan aturan setoran harus lebih dari nol atau penarikan tidak boleh melebihi saldo



Encapsulation sangat penting untuk menjaga keamanan dan integritas sistem, terutama dalam aplikasi keuangan. Tanpa encapsulation siapapun bisa mengubah saldo menjadi negatif atau angka tidak wajar, yang dapat merusak sistem.

### 5. Jawaban soal no 5

#### Mekanisme constructor

- Subclass secara otomatis memanggil constructor superclass tersebut dahulu sebelum melanjutkan ke constructor miliknya sendiri dan dilakukan dengan keyword super (...)
- Jika tidak dipanggil secara eksplisit, maka Java akan memanggil constructor default superclass secara implisit.

Jika constructor superclass tidak dipanggil

↳ Java akan secara otomatis memanggil constructor tanpa parameter dari superclass dan akan terjadi errors (komplikasi)

#### Ilustrasi

Superclass : karyawan

Public class karyawan {

String nama;

Superclass Manager

Public class Manager extends karyawan {

String departement;

Public karyawan (String nama){

this.nama = nama;

System.out.println ("constructor karyawan  
dipanggil");

}

Public Manager (String nama, String departement) {

super(nama);

this.departement = departement;

System.out.println ("Constructor Manager dipanggil");

}

### 6. Jawab soal no 6

↳ dengan memungkinkan berbagai class memberi pihaka berbeda namun dapat diakses melalui tipe yang sama

Contoh : SistemPembayaran, Java

Interface MetodePembayaran {

void bayar (double jumlah);

}

Class Ewallet implements MetodePembayaran {

public void bayar (double jumlah) {

System.out.println ("pembayaran sebesar Rp " + jumlah + " melalui E-wallet berhasil.");

}

}

Class Pemesanan {

private String namaMakanan;



Private double harga;

```
Public pesanan (String namaMakanan, double harga) {  
    this.namaMakanan = namaMakanan;  
    this.harga = harga;  
}
```

```
Public void prosesPembayaran (MetodePembayaran metode) {  
    System.out.println ("Menpesan;" + namaMakanan + "-Rp" + harga);  
    metode.bayar (harga);  
}
```

```
}  
  
Public class SistemPemesanan {  
    public static void main (String [] args) {  
        Pesanan pesanan1 = new Pesanan ("ayam geprek", 23000);
```

```
        MetodePembayaran ewallet = new Ewallet ();
```

```
        pesanan1.prosesPembayaran (ewallet);
```

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

### 7. Jawaban soal no 7

Fitur	Abstract Class	Interface	Sealed class
Mewarisi implementasi	ya	Tidak (default method)	ya
Menyimpan state	ya	Tidak	ya
Multiple Inheritance	Tidak	ya	Tidak
Pembatas Subclass	Tidak	Tidak	Ya (permits)
Tujuan utama	Pewarisan dan generalisasi	Kontrak/fungsi umum	Kontrol ketat pewarisan

- Gunakan Abstract Class saat butuh kerangka dasar dengan logika yang bisa diwariskan
- Gunakan Interface saat ingin mendefinisikan perlaku umum yang bisa dibagi ke banyak class tanpa tercair
- Gunakan sealed class ingin mengontrol secara ketat pewarisan, misalnya untuk keamanan dan validasi tipe dalam API

