Nama : Mariza Winarni Br. Sihtonga
NIM = 4243250032
Kelas = PSK 24A

Jawaban Soal Essay

1 Prinsip Dasar OOP

- Encapsulation (Enkapsulasi) : Membungkus data (attribut) dan kode (metode) dalam satu unit (objek). Menyembunyikan detail internal, hanya mengekspos yang perlu. → Analogi : Remote TV. Saya pakai tombolnya tanpa perlu tahu sirkuit dalamnya
- Inheritance (Pewarisan) : kelas anak (subclass) mewarisi sifat dan perilaku dari kelas induk (superclass). Untuk code reusability and membuat hirarki "adalah-sejenis". → Analogi : Anak mewarisi ciri-ciri orang tua.
- Polymorphism (Polimorfisme) : Objek berbeda bisa merespons perintah/metode yang sama dengan cara yang berbeda sesuai tipenya. "Banyak bentuk". Analogi : Tombol "Mainkan" (di MP3 putar lagu, di video player putar video).
- Abstraction (Abstraksi) : Menyembunyikan kompleksitas, hanya menampilkan yang relevan. Berfokus pada "apa" bukan "bagaimana". Analogi : Mengendarai Mobil, tahu cara mengemudi tanpa perlu tahu kerja mesin dalamnya.

Saling Dukung : Enkapsulasi & abstraksi membuat kode rapi dan mudah dimengerti. Bersama, membangun sistem yang modular, mudah dikembangkan, dan dipelihara.

2. Kelebihan Java 21 untuk OOP

Java 21 adalah versi LTS (Long-Term Support) yang membawa banyak perbaikan. Kelebihan utama: performa lebih baik & fitur bahasa modern.

- Dua fitur Modern java 21 :

1. Record Patterns : Mempermudah ekstraksi data dari objek record (tipe data immutable) dalam switch atau instanceof.

→ Mendukung OOP : Mengurangi kode boilerplate (kode berulang) saat memproses data objek, Membuat kode lebih ringkas dan mudah dibaca, terutama untuk DTO (Data Transfer Object).

2. Virtual Threads (Project Loom) : Thread ringan yang dikelola JVM. Memungkinkan penanganan banyak tugas konkuren tanpa overhead thread OS.

→ Mendukung OOP : Membuat penulisan kode konkuren (berjalan paralel) jauh lebih sederhana, karena kita bisa menulis kode blocking tapi berjalan sangat efisien. Aplikasi lebih skalabel dan responsif.

3. Perbedaan Class dan Object

- Class (kelas):
  .. Blueprint / Cetak Biru : Rancangan atau definisi tentang bagaimana sebuah objek akan dibuat.
  .. Logis: konsep, tidak mengalokasikan memori.
  .. Sekali didefinisikan dipakai berkali-kali.
  .. Mendefinisikan atribut (data) dan metode (perilaku).
- Object (objek):
  .. Instance/contoh nyata: Implementasi nyata dari sebuah kelas
  .. Ada di memori saat program berjalan.
  .. Bisa banyak
  .. Memiliki nilai spesifik untuk atributnya dan dapat melakukan perilaku yang didefinisikan kelas.

Contoh (Manajemen Mahasiswa) ..
Class Mahasiswa adalah cetak birunya. Mahasiswa mhs1 = new Mahasiswa ("Budi", "123"); mhs1 adalah objek nyata dari kelas Mahasiswa.

4. Implementasi: Encapsulation pada BankAccount

1. atribut balance private.
2. Sediakan metode public (getter/setter) seperti getBalance(), deposit(amount), withdraw(amount) untuk mengontrol akses dan perubahan.

```
public class BankAccount{
  private double balance; // PRIVATE : data tersembunyi

  public BankAccoount(double initialBalance) { /* Inisialisasi */ }
  public double getBalance() { return balance; } // GETTER : baca
  public void deposit(double amount) { /* Validasi & tambah */ } // SETTER:
  ubah terkontrol
  public void withdraw(double amount) { /* Validasi & kurang */ } // SETTER:
  ubah terkontrol
}
```

Penting untuk keamanan:

- Mencegah perubahan data sembarangan.
- Integritas Data: Memastikan data selalu valid (misal, saldo tidak negatif).
- Perubahan data terkontrol, mudah dilacak.

5. Mekanisme Constructor Chaining pada Pewarisan
- Mekanisme: Saat objek subclass dibuat, constructor superclass selalu dipanggil lebih dulu.
- Java akan otomatis menyisipkan super(); jika tidak ada panggilan super(...) eksplisit.
- Jika superclass tidak punya constructor default: Akan compile-time error jika super() tidak dipanggil eksplisit.

```
Class karyawan {
    String nama;
    public karyawan(String nama) {/* inisialisasi nama */}
}
class Manager extends karyawan {
    String departemen;
    public Manager(String nama, String departemen) {
        super(nama);
        this.departemen = departemen;
    }
}
```

6. Polymorphism dengan Interface (Sistem Pemesanan Makanan)
- Interface seperti kontrak prilaku.
- Polymorphism itu artinya "banyak bentuk". Bisa diperlakukan banyak objek sebagai type yang sama. Bisa panggil metode yang sama pada mereka (getHarga()), tapi hasilnya beda sesuai objeknya.

Contoh:
1. Kontrak Interface:
   Interface ItemDipesan {String getNama(); double getHarga();}
   (Setiap barang yang dipesan harus punya nama & harga.)
2. Barang-Barang (keras):
   - Class NasiGoreng implements ItemDipesan {/* punya getNama(), getHarga() */}
   - class EsTeh implements ItemDipesan {/* punya getNama(), getHarga() */}
   (Masing-masing implementasi getHarga() bisa beda, misal EsTeh ada pajak.)
3. Keranjang Belanja (Polymorphism):
   Kita bisa masukkan semua NasiGoreng dan EsTeh ke dalam satu List<ItemDipesan>. Lalu, saat menghitung total, kita tinggal panggil item.getHarga() untuk semua item di list. Java akan tahu getHarga() mana yang harus dipakai (punya NasiGoreng atau EsTeh).

9) Perbandingan Abstract class, Interface, dan Sealed class

1. Abstract class :
   - Kelas abstract (tidak bisa dibuat objek). Punya metode kosong dan abstract
   - Jika ada hubungan "is-a" kuat (misal : Hewan adalah kucing). Mengedepankan hierarki dasar dengan sebagian implementasi.
   - Contoh : abstract class Bentuk { abstract double luas(); }
2. Interface :
   - Kontrak perilaku (interface). Hanya deklarasi metode (kecuali default, static).
   - Jika objek "bisa melakukan" hal yang sama tapi tidak ada hubungan "is-a" kuat (misal : Terbang). Untuk multiple inheritance (satu kelas bisa implement banyak). - Contoh : interface BisaTerbang { void terbang(); }
3. Sealed class / Interface (Java 17+):
   - Kelas/ interface yang membatasi kelas mana saja yang boleh mewarisi/ mengimplementasikannya (sealed class x permits A, B {...}).
   - Ketika mengontrol dan mengetahui semua kemungkinan tipenya secara pasti. Berguna untuk pattern matching di switch.
   Contoh : Sealed interface Bentuk2D permits Lingkaran, Persegi,