Nama : Maissy Angelica Palipahan
NIM : 4243250041
Kelas : PSIK 24 A
Mata kuliah : Pemograman Berorientasi Objek

UJI 1, ESSAY

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 untuk masing-masing konsep.
2. Apa kelebihan menggunakan Java versi terbaru (Java 21) dibanding versi-versi sebelumnya dlm konteks pengembangan berbasis OOP? Berikan minimal dua fitur modern Java 21 dan jelaskan bgmn fitur tsb menyederhanakan pengembangan aplikasi OOP.
3. Mahasiswa sering kali salah memahami perbedaan antara class dan object. Jelaskan secara detail perbedaan keduanya dan berikan contoh penggunaan class dan object dalam konteks program manajemen data mahasiswa.
4. Anda diminta membuat class BankAccount. Jelaskan bagaimana anda akan menerapkan encapsulation agar data balance tidak bisa diubah sembarangan. Mengapa encapsulation penting untuk keamanan sistem?
5. Jelaskan bagaimana mekanisme constructor chaining bekerja pada pewarisan di Java. Apa yang terjadi jika constructor pada superclass tidak dipanggil secara eksplisit? Sertakan ilustrasi class Karyawan dan subclass Manager.
6. Polymorphism memungkinkan kita menulis kode yang fleksibel dan mudah di-maintain. Jelaskan bagaimana penggunaan interface mendukung konsep ini, dan berikan contoh penggunaannya dalam sistem pemesanan makanan online.
7. Abstraction membantu menyembunyikan kompleksitas internal. Bandingkan penggunaan abstract class, interface, dan sealed class di Java. Dalam kasus apa masing-masing lebih tepat digunakan?

JAWABAN

1. 1) Encapsulation, merupakan prinsip menyembunyikan data dan detail internal objek, serta hanya memberikan akses melalui antarmuka tertentu seperti metode (getter & setter). Hal ini menjaga keamanan data dan mengurangi kompleksitas sistem. Contoh nyatanya adalah mesin mobil dimana pengguna hanya menekan tombol start tanpa perlu memahami proses internal mesin.
2) Inheritance, memungkinkan kelas anak mewarisi atribut dan metode dari kelas induk. Ini mendorong penggunaan kembali kode dan memudahkan pengembangan

fitur baru. Contoh nyatanya adlh identitas warga negara dimana masyarakat mewarisi atribut dasar warga seperti nama dan alamat, namun juga memiliki atribut tambahan spt: NIP.

3) Polymorphism, memunglinkan objek yg berbeda merespon dengan cara berbeda thdp metode yang sama. Contohnyatanya adalah tombol "cetak" di berbagai aplikasi.

4) Abstraction, merupakan proses menyederhanakan sistem dgn hanya menampilkan bagian penting dan menyembunyikan detail yg tdk relevan. Contohnya adalah remote TV dimana pengguna hanya perlu memahami tombol & dasar tanpa tahu cara kerja elektronik di dalamnya.

2. Kelebihan menggunakan Java versi terbaru (Java 21) dlm hubungan pengembangan berbasis OOP adalah kode lebih ringkas dan bersih, kontrol pewarisan lebih luas, meningkatkan keamanan dan kestabilan arsitektur, dan dukungan yang lebih baik thdp prinsip OOP. Dua fitur modern Java 21:

1) Record Classes, Saat menggunakan record, Java secara otomatis membuat constructor, getter, equals(), hashCode(), dan toString(). Ini mendukung prinsip encapsulation dan abstraction karena data tetap tersimpan secara aman dan hanya dapat diakses melalui metode yg sdh disediakan.

2) Pattern Matching for Switch, menggunakan switch yg lebih ekspresif dan aman dengan pattern matching thdp type objek. Mendukung polymorphism & memperjelas alur logika saat bekerja dgn berbagai subclass / type data berbeda tanpa perlu byk if-else atau casting manual.

3. - Class merupakan blueprint (cetakan) atau template untuk membuat objek. Class mendefinisikan struktur object tetapi tidak menyimpan data nyatanya. Contoh:

```
public class Mahasiswa {
    String nama;
    String nim;
    String prodi;

    void tampilkanData() {
        System.out.println("Nama: " + nama);
        System.out.println("nim: " + nim);
        System.out.println("prodi: " + prodi);
    }
}
```

- Object adalah instance (perwujudan) dari class. Objek adalah data nyata yang dibuat berdasarkan class dan dapat menyimpan nilai2 sesungguhnya. Contoh:

```
public class Main {
    public static void main(String[] args) {
        Mahasiswa mhs1 = new Mahasiswa()
        mhs1.nama : "Maussy";
        mhs1.nim  : "4243250041";
        mhs1.prodi : "Ilmu Komputer";

        mhs1.tampilkanData();
    }
}
```

4. Utk membuat class BankAccount dgn prinsip encapsulation, harus menyembunyikan data sensitif spti balance agar tdk bisa diakses atau diubah secara langsung dari luar kelas. Variabel balance dibuat private artinya hanya bisa diakses dalam class BankAccount. Akses ke balance hanya bisa dilakukan melalui method public yang terkontrol, seperti getBalance(), deposit(), dan withdraw(). Dgn metode ini, sistem bisa memvalidasi setiap perubahan saldo agar tdk terjadi kesalahan / manipulasi data secara langsung. Encapsulation penting utk keamanan sistem krn dapat melindungi data sensitif, mencegah penyalahgunaan & meningkatkan pemeliharaan.

5. Ketika sebuah objek dari subclass dibuat, Java secara otomatis memanggil constructor superclass terlebih dahulu. memastikan bahwa semua atribut dan logika dalam superclass telah dijalankan sebelum subclass melanjutkan inisialisasinya. Jika constructor superclass tidak dipanggil secara eksplisit, maka program akan menghasilkan error kompilasi karena Java tdk tahu bagaimana menginisialisasi bagian superclass.

Ilustrasi class karyawan & subclass Manager.

```
public class Karyawan {
    String nama;
    public karyawan (String nama) {
        this.nama = nama;
        System.out.println("Constructor karyawan dijalankan");
    }
}

Public class Manager extends karyawan {
    String departemen;
    public Manager (String nama, String departemen ) {
        super (nama);
        this.departemen = departemen;
    System.out.println("Constructor manager dijalankan");
}
```

6. Interface mendukung polymorphism dgn memungkinkan berbagai kelas berbeda memiliki perilaku yang sama melalui method yg diimplementasi. Dlm sistem pemesanan makanan online, misalnya interface pembayaran digunakan oleh berbagai metode pembayaran spti ovo, Gopay, dan kartu kredit. Semua metode ini bisa dipanggil dgn cara yg sama (prosesPembayaran()), shg kode lebih fleksibel, mudah diperluas, dan tidak bergantung pada implementasi spesifik.

7. - Abstract class digunakan ketika membuat kelas dasar yang memiliki atribut & logika bersama utk diturunkan ke subclass, misalnya class kendaraan yang diwarisi oleh mobil dan motor.
- Interface digunakan ketika mendefinisikan perilaku umum yg dapat diimplementasikan oleh berbagai kelas yg tdk saling terkait, spti interface pembayaran yg dapat digunakan oleh kelas ovo, Gopay, atau kartukredit.
- Sealed class digunakan utk membatasi subclass mana yg boleh mewarisi sebuah class. Contohnya pd class ErrorJenis yg hanya boleh diwarisi oleh DatabaseError, ValidationError, dan SystemError.