Nama : Kayla Ameka Putri
Jum'at, 23 Mei 2025
Nim : 4242590006
Kelas : PSIK-24A
Matkul : Pemograman Berorientasi Objek

"ESSAY UTS"

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.

Jawaban :

- Encaptulation bertindak sebagai pelindung data dengan membatasi akses langsung, yang hanya bisa diakses dengan metode tertentu yang telah ditentukan.

contohnya seperti Mobil, dimana didalam kap mesin tersembunyi sebuah mesin, dan yang pengendara hanya berinteraksi dengan setir dan pedal.

- Inheritance memungkinkan suatu class yang akan mewarisi sifat dan metode dari class lain.

contohnya : Seperti dalam keluarga yang dimana sang anak mewarisi sifat orang tuanya namun memiliki karakteristiknya sendiri yang unik.

- Polymorphism yaitu dimana objek berbeda merespons metode yang sama namun dengan cara yang berbeda.

contohnya : Seseorang yang berperan sebagai dosen dikampus dan sebagai orang tua dirumah. dimana menyesuaikan perilaku sesuai konteks.

- Abstraction tidak menampilkan detail kompleks dan hanya menampilkan fungsionalitas penting

Contohnya : Pengemudi motor tidak tau bagaimana cara mesin bekerja, cukup tahu cara mengemudi seperti menancap gas dan mengerem.

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

Jawaban :

- pattern matching for switch : berguna untuk mempermudah penanganan objek dalam blok switch dengan deklarasi variabel langsung.

example :
```
switch (obj) {
    case String s -> System.out.println(s.length());
    case Integer i -> System.out.println(i * 2);
}
```

dimana fitur ini berguna untuk menyederhanakan kode dan mengurangi boilerplate.

Forte

- Record patterns : Memungkinkan dekomposisi record secara langsung

example : if (obj instanceof point (int x, int y)) {
System . out. println (x + y) ;
}

Membuat kode lebih ringkas dan mudah dibaca

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.

Jawaban :

- class adalah cetakan yang mendefinisikan struktur (misalnya, Class Mahasiswa { String nama ; String nim ; })
- object adalah realisasi dari class tersebut (misalnya, Mahasiswa mhs 1 = new Mahasiswa () ; mhs 1. nama = "Nina" ;)

Contoh implementasi :

class berisi definisi atribut seperti Nim dan metode daftarKuliah(), sedangkan object menciptakan mahasiswa spesifik seperti Budi (Nim: 4242550006) yang menjalankan metode tersebut.

4. Untuk memastikan data balance dalam class BankAccount tidak bisa diubah sembarangan, encapsulation diterapkan dengan: • membuat variabel private
• menyediakan metode akses terkontrol

Penerapan Encapsulation di BankAccount

Public class BankAccount {
private double balance ;

Public void withdraw (double amount) {
if (amount > 0 && amount <= balance) balance -= amount ;
}
}

alasan pentingnya karena untuk mencegah perubahan yang tidak sah (seperti, saldo negatif) dan memusatkan logika validasi dalam satu tempat.

5. Constructor subclass harus memanggil constructor superclass (eksplisit dengan super() atau otomatis oleh Java). Jika superclass tidak memiliki constructor default, programmer wajib memanggilnya secara manual.

example: class karyawan {
karyawan (String nama) { System. out . println ("karyawan :" + nama);
}
}
Class Manager extends Karyawan {

```
    Manager () {
        Super ("kayla");
    }
}
```

6. Polymorphism melalui interface memungkinkan banyak class berbeda mengimplementasikan metode dengan nama yang sama namun menggunakan logika yang berbeda

example : pemesanan makanan

```
{   interface Pemesanan {
        void ProsesPemesanan ();
    }
    class PesanOnline implements Pemesan {
        Public void ProsesPesanan () { /* logika khusus */ }
    }
```

Jadi sistem memiliki keunggulan dimana sistem bisa menerima berbagai jenis pemesanan tanpa mengubah kode utama.

7. • Abstract Class vs Interface vs Sealed Class

* Abstract class : digunakan ketika ada logika inti yang ingin dibagikan ke subclass
  Mis: abstract class kendaraan ${ abstract void start (); })

* Interface : cocok untuk mendefinisikan kontrak tanpa implementasi
  Mis : Interface Listrik { void charge () ; })

* Sealed Class : Membatasi inheritance ke class tertentu
  Mis : Sealed class Role permits Admin , User {})

Penulisan :

1. pilih Abstrac Class untuk mendefinisi class dengan logika umum dan state
2. Interface untuk kontrak murni tanpa state
3. Sealed Class : Membatasi pewarisan ke class tertentu.