Nama : Tia Risky Yasmin Saketang
Nim : 4243550014
Kelas : Psik 24-A
Mata Kuliah : Pemrograman Berorientasi Objek
Dosen Pengampu : Insan Taufik, M.Kom

Soal Essay

1. Jelaskan bagaimana prinsip encaptulation, 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 : Menyembunyikan detail implementasi dan hanya menampilkan interface yang diperlukan. Data dan metode dibungkus dalam satu unit (class), sehingga akses didata bisa dikontrol
- Inheritance : Memungkinkan class baru mewarisi sifat dan perilaku dari class yang sudah ada Memudahkan pengembangan dengan mengurangi duplikasi kode dan mendukung perluasan fitur
- Polymorphism : Kemampuan objek untuk diakses melalui interface yang sama walaupun implementasinya berbeda. Membuat kode lebih fleksibel dan mudah diperluas
- Abstraction : Menyederhanakan kompleksitas dengan hanya menampilkan fitur penting dan menyembunyikan detail yang tidak relevan

Keempat prinsip ini saling mendukung : Encaptulation menjaga keamanan data, Inheritance dan abstraction menyederhanakan pengembangan dan pelihataan, polymorphism memungkinkan penggunaan kode yang generik dan mudah diubah tanpa menggangu sistem yang ada.

Analogi dalam kehidupan Nyata :

- Encaptulation : Seperti remote TV, pengguna hanya menekan tombol tanpa tau rangkaian elektronik didalamnya
- Inheritance : Seperti anak yang mewarisi sifat dari orang tua, misal warna mata
- polymorphism : Seperti anak "jalan", bisa berarti berjalan kaki, naik mobil, atau naik sepeda, tergantung konteksnya
- Abstraction : Seperti mengendarai mobil, Pengemudi cukup tau cara mengemudi tanpa tau cara cara kerja mesin

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

Jawaban:

(RKY)

Java 21 membawa berbagai peningkatan yang memudahkan penimbangan oop dibanding versi sebelum

Dua fitur modern Java 21 :

- Record Patterns : Memudahkan ekstraksi data dari objek record secara langsung dalam pola, sehingga kode menjadi lebih ringkas dan mudah dibaca :
- Sealed classes : Membatasi siapa yang boleh mewarisi satu class, sehingga hirearki class lebih terkontrol dan aman. Membantu dalam desain domain yang jelas dan mencegah subclassing yang tidak diinginkan

Kedua fitur ini menyederhanakan pengembangan oop :

- Record patterns mengurangi boilerplate code saat memproses data
- Sealed classes mempermudah maintenance dan meningkatkan keamanan desain dengan membatasi ekstensi class

3. Mahasiswa seringkali 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 blueprint atau cetakan, mendefinisikan struktur dan perilaku objek tetapi belum merepresentasikan data nyata
- Object adalah instansiasi nyata dari class, memiliki nilai nilai data yang spesifik

Contoh :

```
class Mahasiswa {
    String nama ;
    String nim ;
    void tampilkanData ( ) {
        System.out.println (nama + "-" + nim);
    }
}
Mahasiswa mhs1 = new Mahasiswa ( ) ;
mhs1.nama = "Tia";
mhs1.nim = "4243550014";
mhs1.tampilkanData ( ) ;
```

Mahasiswa adalah class, sedangkan mhs1 adalah object yang mempresentasikan satu mahasiswa

KKY

4. Anda diminta membuat class BankAccount. Jelaskan bagaimana anda akan menerapkan encaptulation agar data balance tidak bisa diubah sembarangan. Mengapa Encaptulation penting untuk keamanan sistem

Jawaban :

Untuk menerapkan encaptulation agar data balance tidak bisa diubah sembarangan :

- Jadikan balance sebagai private
- Sediakan getter dan setter yang mengontrol akses dan validasi perubahan

Contoh :

```
Public class BankAccount {
    Privare double balance;

    public double getBalance() {
        return balance;
    }

    Public void deposit (double amount) {
        IF (amount > 0) balance += amount;
    }

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

}
```

Alasan encaptulation penting : Melindungi data sensitif dari perubahan sembarangan, menjaga integritas data, dan mencegah akses langsung bisa menyebabkan bug atau celah keamanan

5. Jelaskan bagaimana mekanisme constructor chaining bekerja pada pewarisan java. Apa yang terjadi jika constructor pada superclass tidak dipanggil secara eksplisit ? Sertakan ilustrasi class karyawan dan subclass manager

Jawaban :

* Mekanisme constructor chaining pada pewarisan di Java *

- Constructor chaining terjadi saat subclass memanggil constructor superclass menggunakan super()
- Jika constructor superclass tidak dipanggil secara eksplisit, Java otomatis memanggil construc tor default superclass
- Jika superclass tidak punya constructor default, harus dipanggil secara eksplisit

Ilustrasi :

```
class Karyawan {
    String nama;
    Karyawan (String nama) {
        this.nama = nama;
    }

}

class Manager extends Karyawan
    String departemen ;
    Manager ( String nama, String departemen) {
        super (nama);
        this.departemen = departemen ;
    }

}
```

Jika super (nama) tidak dipanggil, akan terjadi error jika superclass tidak punya constructor default

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

Jawaban :

Interface mendefinisikan kontrak yang harus diikuti, sehingga berbagai class dapat diakses secara polimorfik. Contoh :

```
interface Pembayaran {
    void prosesPembayaran (double jumlah);
}

class PembayaranGopay implements Pembayaran {
    public void prosesPembayaran (double jumlah) {/*...*/}
}

class PembayaranOVO implements Pembayaran {
    Public void prosesPembayaran (double jumlah) {/*...*/}
}
```

(RKY)

```
void proses (pembayaran p, double sumlah) {
    p. prosesPembayaran ( sumlah);
}
```

Dengan interface, sistem bisa menerima berbagai jenis pembayaran tanpa tergantung pada implementasi spesifik

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 :

* perbandingan Abstract class, Interface, dan Sealed class di Java

Fitur	Abstract class	Interface	Sealed class
Isi	Metode abstract dan konkrit	Hanya deklarasi metode	Bisa abstrak / konkrit, subclass
		(Java 8+ : default method)	terbatas
Pewarisan	Satu superclass	Multi - interface	Subclass terbatas
Instansiasi	Tidak bisa	Tidak bisa	Tidak bisa
Kapan diguna	Ada perilaku dasar	Hanya kontrak, multi-	Hirearki terbatas
kan		inherit	

Kapan digunakan :

- Abstract class → jika ada perilaku dasar yang ingin diwariskan ke semua subclass
- Interface → jika hanya butuh kontrak tanpa implementasi, atau butuh multiple inheritance
- Sealed class → jika ingin membatasi subclass yang boleh mengimplementasikan, cocok untuk domain model yang tertutup