Nama : Postar A.E Ginting
NIM : 4213250003
Kelas : PSIK 21 A

Soal Essay

Prinsip Encapsulation, Inheritance, polymorphism, dan abstraction saling mendukung dalam pengembangan sistem

* Encapsulation → menyembunyikan detail internal objek dan hanya memperlihatkan interface yang jelas.
  Analogi → remote tv yang hanya menyediakan tombol untuk mengoperasikan tv tanpa tahu bagaimana tv bekerja.
* Inheritance → membuat kelas baru berdasarkan kelas yang sudah ada untuk menghindari duplikasi kode
  Analogi → Seperti seorang anak mewarisi sifat orangtuanya, namun bisa menambah atau mengubah beberapa kebiasaan
* Polymorphism → kemampuan objek untuk merespons metode yang sama dengan cara berbeda, tergantung tipe objek.
  Analogi → seperti kata "buka" yang bisa berarti buka pintu, buka buku atau membuka aplikasi bergantung konteks
* Abstraction → Menyediakan gambaran umum tanpa harus mengetahui detail implementasi
  Analogi → Seperti saat menggunakan mobil, kita tahu cara mengemudi tanpa perlu tahu bagaimana cara mobil bekerja secara detail.

Keempat konsep ini bekerja sama untuk membuat sistem yang modular, mudah dikembangkan dan mudah dipelahara. Misalnya, abstraction dan encapsulation mengurangi kompleksitas, Inheritance mengurangi duplikasi, dan polymorphism memberikan fleksibilitas.

Date:

2. Kelebihan Java 21 dan fitur modern dalam konteks OOP.
↳ kelebihan Java 21 dibanding versi sebelumnya.
* lebih cepat, lebih efisien dengan optimasi runtime dan garbage collection
* Pukungan fitur modern yang mengembangkan kode bersih dan ekspresif
Dua fitur modern Java 21
1. Pattern Matching for Switch (switch expression)
↳ yaitu memungkinkan switch yang lebih powerfull dengan pola matching, menyederhanakan pengecekan tipe dan kondisi dalam OOP
2. Record Classes
↳ yaitu memudahkan membuat class immutable untuk data carrier tanpa harus menulis boilerplate code seperti getter, setter, equals, dan hashcode

Kedua fitur ini membuat kode lebih ringkas, mudah dibaca, dan mengurangi kesalahan, sehingga pengembangan aplikasi OOP jadi lebih efisien.

3. Perbedaan class dan object dengan contoh manajemen data mahasiswa
* class -> blueprint atau template yang mendefinisikan atribut (data) dan method (perilaku)
* object -> instance nyata dari class yang menyimpan data sebenarnya

Contoh

```
Class Mahasiswa { // class
  String nama;
  Int umur;

  Void tampilkan Info () {
    System.out.println ("nama" + nama + ", umur: " + umur);
  }
}
Mahasiswa mhs1 = New Mahasiswa (); // object dari class mahasiswa
```

```
mhs1.nama = "Budi";
mhs1.umur = 20;
mhs1.tampilkanInfo();
```

↳ keterangan : Mahasiswa → Class (template)
mhs1 → object konkret berisi data mahasiswa

4. Penerapan encapsulation pada class BankAccount untuk melindungi balance

• Cara menerapkan

* Buat attribute balance sebagai private agar tidak bisa diakses langsung dari luar class
* Sediakan method public seperti deposit(), withdraw() untuk mengubah balance dengan validasi yang sesuai
* tidak menyediakan setter langsung untuk balance agar tidak diubah sembarang

Contoh:

```
Public Class BankAccount {
    Private double balance;

    Public BankAccount(double initialBalance) {
        this.balance = initialBalance;
    }
    Public void deposit(double amount) {
        if (amount > 0) {
            balance += amount;
        }
    }
    Public void withdraw(double amount) {
        if (amount > 0 && amount <= balance) {
            balance -= amount;
```

```
    }
  }
  Public double getBalance () {
    return balance;
  }
}
```

Pentingnya encapsulation => melindungi data sensitif agar tidak diubah sembarang, menjaga konsistensi dan keamanan data dalam sistem.

Mekanisme constructor chaining pada inheritance di Java

* Saat sebuah subclass dibuat, constructor superclass selalu dipanggil terlebih dahulu untuk menginisialisasi bagian dari superclass.
* Constructor chaining terjadi ketika constructor subclass memanggil constructor superclass secara eksplisit dengan super()
* Jika tidak dipanggil secara eksplisit, Compiler otomatis memanggil Constructor default Superclass. Jika Superclass tidak mempunyai Constructor Default = Error

Ilustrasi

```
Class karyawan {
  String nama;

  Public Karyawan (String nama) {
    this.nama = nama;
  }
}

Class Manager extends karyawan {
  int level;

  Public Manager (String nama, int level) {
    Super (nama); //memanggil constructor karyawan
    this.level = level;
  }
}
```

Date:

6. Interface mendukung polymorphism dan contohnya di sistem pemesanan makanan online

* Interface mendefinisikan kontrak method tanpa implementasi
* dengan interface, objek dari berbagai kelas yang berbeda bisa diperlakukan sama jika mengimplementasikan interface tersebut. (polymorphism).

Contoh:

```
Interface Pembayaran {
  Void bayar (double jumlah);
}
Class TransferBank implements Pembayaran {
  Public Void bayar (double jumlah) {
    System.out.println ("Bayar melalui transfer bank sebesar" + jumlah);
  }
}
Class DompetDigital implements Pembayaran {
  Public Void baya (double jumlah) {
    System.out.println ("Bayar melalui dompet digital sebesar" + jumlah);
  }
}
```

Date:

7. Perbandingan abstract class, Interface, dan sealed class di Java

No	Konsep	Penjelasan	Kapan digunakan
1	Abstract Class	class yang tidak bisa di instansiasi, bisa punya method dengan implementasi dan abstrak.	Saat ingin membuat template class dengan beberapa metode yang sudah diimplementasikan dan beberapa yang abstrak. Contoh: base class Fitur dasar
	Interface	Hanya mendefinisikan method abstrak (sejak java 8 bisa juga default dan static method)	Saat ingin mendefinisikan kontrak yang bisa diimplementasi oleh banyak kelas tanpa keterikatan hirarki
	Sealed class	membatasi class mana saja yang boleh menjadi subclassnya	Saat ingin kontrol ketat hirearki inheritance supaya hanya class tertentu yang bisa mewarisi