Nama : Naufal Aqilah Asra
NIM : 4241250020
Kelas : PSIK 24A
Matkul : Pemrograman Berorientasi Objek

Soal 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.

⇒ Prinsip dari encapsulation yaitu menyembunyikan detail internal dari objek agar tidak bisa diakses langsung dari luar sehingga memudahkan debugging dan mengurangi risiko bug, karena data hanya bisa diubah lewat method yang disediakan. Contoh: Pada smartphone, kita bisa menelpon dan mengirim pesan tanpa tahu bagaimana sistem operasinya bekerja. Semua itu disembunyikan dibalik layar. Pada inheritance yaitu suatu kelas baru bisa mewarisi kelas lama yang dapat mengurangi duplikasi kode, dan mempercepat pengembangan sehingga cukup tulis suatu logika dasar sekali saja. Contoh: seorang anak mewarisi sifat dari orang tuanya seperti warna mata atau rambut, namun tetap bisa memiliki sifat unik sendiri. Pada polymorphism yaitu objek dari kelas turunan bisa dibuat sebagai objek dari kelas induk sehingga interaksi lebih fleksibel antar objek dengan salah satu fungsi bisa dipanggil untuk semua objek, meskipun mereka punya implementasi masing-masing. Contoh : diberbagai perangkat atau aplikasi, memiliki tombol play melakukan fungsi yang berbeda tergantung konteks seperti memutar video atau lagu. Dan pada abstraction yaitu menyembunyikan kompleksitas dan hanya menampilkan fitur penting. Contoh : kita dapat melihat nama dan dekskripsi makanan dalam menu restoran tanpa perlu tau bagaimana makanan itu dibuat didapur.

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.

⇒ Java 21 membawa berbagai peningkatan yang mendukung prinsip-prinsip OOP seperti abstraksi, enkapsulasi, dan polimorfisme, serta menjadikan pengembangan perangkat lunak lebih efisien dan aman. Fitur modern yang ada pada Java 21, yaitu :

1. Record Patterns: Memungkinkan kita mendekonstruksi objek record secara langsung dalam ekspresi seperti "if" atau "switch", sehingga kita bisa mengekstrak data dengan cara yang ringkas dan jelas, seperti mengurangi kode boilerplate saat mengakses data objek. dan mendukung abstraksi data yang lebih bersih
2. Sealed Classes : Membatasi class mana saja yang boleh mewarisi suatu class. Hal ini meningkatkan kontrol terhadap hierarki pewarisan sehingga menghindari perluasan class yang tidak diinginkan dan menjadi alternatif lebih aman dibanding abstract class saat hierarki objek sudah ditentukan

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.

⇒ Class adalah cetakan atau template utk membuat objek, Sedangkan Object adalah hasil nyata dari class, yang berarti class tidak menyimpan data dan hanya menyediakan struktur dan perilaku, sedangkan object digunakan untuk menyimpan data dan menjalankan fungsi. Sehingga menggunakan atribut class tanpa membuat object akan error begitupun sebaliknya.

Contoh dalam konteks program manajemen data mahasiswa :

```
class Mahasiswa {  // contoh class
    String nama ;
    string nim;

    void belajar () {
        System. out . println ( nama + " sedang belajar.");
    }
}
public class main {
    public static void main(String [] args) {
        Mahasiswa mhs = new Mahasiswa (); // contoh object
        mhs. nama = " Naufal";
        mhs. nim  = "4241250020";
        mhs. belajar();
    }
}
```

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?

⇒ Untuk menerapkannya, saya akan membuat atribut "balance" sebagai private, sehingga tidak dapat diakses langsung dari luar class. Lalu, saya akan membuat method "getBalance()" sebagai getter untuk membaca nilai saldo dan method deposit(double amount) serta withdraw(double amount) sebagai setter untuk mengatur perubahan saldo secara terkendali. Encapsulation ini penting untuk mencegah perubahan langsung dari luar class atau melindungi data sensitif dan meningkatkan keandalan sistem sehingga bug atau manipulasi data bisa dicegah

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.

⇒ Constructor dari subclass secara otomatis atau eksplisit memanggil constructor dari superclass untuk mewarisi dengan "super()". Jika tidak ada pemanggilan secara eksplisit, Maka Java akan memanggil constructor tanpa parameter (default constructor) dari superclass. Jika tidak tersedia, akan error

ilustrasinya :

```
class Karyawan {
Karyawan (String nama) {
    this. nama = nama ;
}
  void tampilInfo () {
      System. out. println ("Karyawan : " + nama);
  }
}

class Manager extends Karyawan {
   Manager (String nama) {
     super (nama) ;
   }
   void tampilInfo() {
     super. tampilInfo();
     System. out. println ("Manager : " + nama) ;
   }
}
```

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.

⇒ Interface mendukung polymorphism karena mereka menyediakan kontrak yang dapat diikuti oleh berbagai class berbeda. Ketika banyak class mengimplementasikan interface yang sama, kita bisa memperlakukan objek-objek tersebut secara polimorfik yang artinya kita bisa menyimpan atau mengaksesnya lewat referensi tipe interface, tanpa peduli class aslinya sehingga sistem lebih fleksibel, mudah dikembangkan, dan pemeliharaan kode lebih bersih dan mudah dirawat.

Contoh dalam sistem pemesanan makanan online :

```
interface Pembayaran {  // Interface
   void prosesBayar(double total);
}
class Gopay implements Pembayaran {
   Public void prosesBayar(double total) {
       System.out. println ("Bayar " + total + " lewat Gopay.");
   }
}
class cash implements Pembayaran {
   public void prosesBayar(double total){
       system. out. println ("Bayar " + total + " secara tunai")
   }
}
class Pemesanan {
   void proses (Pembayaran Metode, double total) {
       Metode. prosesBayar (total) ;  // Polymorphism
   }
}
```

7) Abstraction membantu menyembunyikan kompleksitas internal. Bandingkan penggunaan abstract class, interface, dan sealed class di Java. Dalam kasus apa masing-masing lebih tepat digunakan?

⇒ Abstract class, interface, dan sealed class sama-sama digunakan untuk menyembunyikan detail teknis (abstraction), tapi punya perbedaan fungsi. Abstract class cocok digunakan saat ingin membuat kerangka dasar suatu class dengan method sudah jadi. Contoh kasusnya yaitu ketika ada class dasar dengan beberapa method yang memiliki implementasi umum, dan lainnya dideklarasikan secara abstrak. Interface digunakan untuk membuat aturan atau kontrak bagi class lain yang tidak saling berhubungan. Contoh kasusnya yaitu ketika beberapa class (yang tidak saling berhubungan) harus berbagi perilaku umum. Sementara itu, sealed class berguna kalau kita ingin membatasi class mana saja yang boleh mewarisi class tertentu, sehingga tidak semua class bisa mengubah atau memperluas fungsinya. Contoh kasusnya yaitu ketika kita ingin membatasi class mana saja yang boleh mewarisi class tertentu seperti "public sealed class Pembayaran permits Tunai, Gopay {}"