Date

Nama : Mika M.F. simanullang
NIM : 4243250052
Kelas : PSIK A
Mata kuliah : Program Berorientasi Objek
Dosen pengampu : Insan Taufik, M. kom

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 konsep
Jawaban :
Prinsip-prinsip encapsulation, inheritance, polymorphism, dan abstraction saling melengkapi dalam pengembangan perangkat lunak yang terstruktur, mudah di kembangkan, efesin dalam pemeliharaan. Encapsulation berperan melindungi data dan memisahkan antar bagian sistem agar tidak saling tergantung. Inheritance yang memungkinkan pewarisan fitur dari kelas induk sehingga proses pembuatan kode menjadi lebih cepat dan hemat usaha. Polymorphism memberikan kemampuan bagi objek yang berbeda untuk merespon yang sama dengan cara masing-masing sehingga sistem menjadi fleksibel. Abstraction menyajikan elemen penting saja, menyembunyikan detail yang kompleks agar lebih mudah dipahami dan digunakan

Apa kelebihan menggunakan java versi terbaru (java 21) dibandingkan 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 :
Versi java 21 memberikan banyak kelebihan yang menghadirkan fitur modern yang menyederhanakan penulisan, minim boilerplate, namun tetap mempertahankan prinsip OOP dan juga beberapa fitur baru membuat Class dan hierarchy bisa dimodelkan lebih baik.
dua contoh fitur java 21 :
* Pattern Matching for instanceof & Record Patterns
memudahkan pengecekan tipe dan ekstraksi dalam satu langkah. Ini berguna dalam OOP saat menangani banyak subclass atau record.
* Sealed class
Membatasi class mana saja yang boleh mewarisi superclass tertentu. Sangat bermanfaat dalam pewarisan karena membuat struktur hierarki lebih terkendali dan aman.

Date

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 template atau cetak biru yang mendefinisikan struktur (atribut) dan perilaku (method) dari suatu entitas. Object adalah intance nyata dari class yang menyimpan data aktual dan dapat menjalankan fungsi-fungsi yang di definisikan dalam class.

Contoh penggunaan :

*Class Mahasiswa

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

    Void tampilan () {
      System.out.println ("Nama : " + nama );
      System.Out.println ("NIM : " + nim ) ;
    }
}
```

*Main program → menggunakan Object

```
public class Main
    Pubuc static void main (String[] args ){
        Mahasiswa m = new Mahasiswa ( ); // object dibuat dari class
        m.nama = "Mika" ;
        m.nim = "4243250052";

        m.tampilan ( ); // panggil method
    }
}
```

Mahasiswa = class (template)

m = object (data nyata; Mika, 4243250052)

4 Anda diminta membuat class BankAccount. Jelaskan bagaimana Anda akan menerapkan encapsulation agar data balance tidak diubah sembarangan. Mengapa encapsulation penting untuk keamanan sistem ?

Jawaban :

*Cara menerapkan encapsulation di class BankAccount

- Menjadikan variabel balance private
- Menggunakan method getBalance () dan deposit () atau withdraw ().

Date

Dengan encapsulation, data sensitif seperti saldo bank tetap aman dan juga terkendali karena hanya bisa diakses melalui fungsi resmi yang diberi aturan dan validasi. Ini adalah praktik terbaik dalam membangun sistem yang kuat dan tahan terhadap kesalahan atau penyalahgunaan.

5 Jelaskan bagaimana sistem 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.

Jawaban :

Constructor chaining adalah mekanisme dimana contructor satu class memanggil constructor lain, baik dalam class yang sama maupun superclass. Dalam pewarisan, digunakan untuk memastikan bahwa constructor superclass dijalankan lebih dulu sebelum constructor subclass. Jika constructor class tidak dipanggil maka akan terjadi error pada saat kompilasi.

*Class karyawan (Superclass)

```
public class karyawan {
    public karyawan() {
        System.out.println(" Constructor karyawan dipanggil");
    }
}
```

* class Manager (subclass)

```
public class Manager extends karyawan {
    public Manager {
        super(); //Memanggil constructor karyawan
        System.out.println("Konstructor Manager dipanggil");
    }
}
```

*Main Class

```
public class Main {
    public static void main(String[] args) {
        Manager m = new Manager(); //membuat objek Manager
    }
}
```

6 Polymorphism memungkinkan kita menulis kode yang fleksibel dan mudah dimaintain. Jelaskan bagaimana penggunaan interface mendukung konsep ini, dan berikan contoh penggunaannya dalam pemesanan makanan online.

Jawaban :

Interface mendefinisikan kontak perilaku (method) tanpa implementasi. Class-

Date

class berbeda bisa mengimplementasikan interface tersebut dengan caranya masing-masing. kita bisa menyimpan objek dari berbagai class.
contoh :

```
interface Pesanan { void proses(); }
class Makanan implements Pesanan {
    public void proses() { System.out.println("Proses makanan."); }
}

public class Main {
  public static void main(String[] args) {
      Pesanan p = new Makanan(); // polymorphism
      p.proses();
  }
}
```

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

Jawaban :

Abstraction dijava menyembunyikan detail kompleks dari pengguna dan hanya menampilkan fungsionalitas penting. Abstrad class cocok digunakan ketika beberapa kelas memiliki hubungan logis yang kuat dan berbagi sebagian implementasi, sehingga memudahkan pengelompokan logika umum. Interface lebih fleksibel karena memungkinkan class yang tidak terhubung untuk berbagi perilaku tanpa pewarisan langsung. Sealed class untuk membatasi pewarisan hanya pada class-class tertentu yang telah ditentukan, memberikan kontrol penuh class dan menjaga struktur kode tetap aman dan terprediksi.