Nama : Dinda Syafitri 23 Mei 2025 / Jum'at
Nim : 4243250013
Matkul : Pemograman Berorientasi Objek

UJIAN TENGAH SEMESTER

1. Prinsip encapsulation, inheritance, polymorphism, dan abstraction serta analoginya
: •) encapsulation : Membungkus data (variabel) dan method yang bekerja pada data tersebut Dalam satu unit (class) dengan akses terkontrol (public, private, protected)
Analogi → Kita hanya perlu tahu cara memasukkan kartu dan pin, tidak perlu Tahu cara mesin bekerja didalamnya.
•) Inheritance : Class anak (subclass) mewarisi atribut dan method dari class induk (superclass).
Analogi → Dokumen template yang bisa diadaptasi untuk berbagai keperluan
•) polymorphism : Kemampuan object untuk memiliki banyak bentuk melalui overriding dan over loading
Analogi → Tombol power pada hp jika ditekan sebentar, layar akan mati dan jika ditekan lama akan merestart hp
•) Abstraction : Menyembunyikan detail kompleks dengan menampilkan fungsionalitas penting. Dicapai melalui abstact class dan interface.
Analogi → Kita hanya perlu tau cara gas dan rem tanpa perlu paham mesin bekerja

2. Kelebihan java versi terbaru dalam konteks pengembangan berbasis oop dan fitur modernnya serta bagaimana fitur menyederhanakan pengembangan aplikasi oop
: •) record patterns : Memudahkan ekstraksi data dari objek dengan sintaks sederhana. membantu dalam menulis kode yang rapi dan jelas
•) virtual threads : Membuat pembuatan thread jauh lebih ringan dan efisien, sehingga cocok untuk aplikasi yang butuh banyak proses persamaan

3. Perbedaan class dan object dan contoh penggunaan dalam konteks program manajemen data Mahasiswa
: •) class adalah blueprint / template yang mendefinisikan variabel atau method dan hanya dibuat sekali namun bisa dipakai berkali-kali. contoh: Formulir pendaftaran mahasiswa
•) object adalah instansi nyata yang dibuat dari class, setiap object punya nilai untuk atributnya. contoh: Formulir yang sudah diisi dengan data mahasiswa tertentu.

class:	object:
class Mahasiswa {	Mahasiswa mhs1 = new Mahasiswa ();
string nama;	Mhs1.nama = "Dinda" ;
string nim;	Mhs1.nim = "4243250013" ;
}	

4. Membuat class BankAccount. Bagaimana menerapkan encapsulation agar data balance tidak bisa diubah sembarangan. mengapa encapsulation penting untuk keamanan sistem?

: untuk melindungi data balance, menggunakan:

•) private : agar data tidak bisa diakses langsung dari luar

•) getter / setter : untuk mengatur akses yang aman

contoh:

```
Public class BankAccount {
  private double balance;          → Private

  public double getBalance() {     → Getter
    return balance;
  }

  public void setBalance(double amount) {  → setter
    if (amount > 0) {
      balance = amount;
    }
  }
}
```

penting digunakan encapsulation agar data keuangan tidak bisa diubah sembarangan.

7. Bandingkan penggunaan abstarct class, interface, dan sealed class di java. Dalam kasus apa masing-masing lebih tepat digunakan?

: •) abstract class → bisa punya implementasi dan field. buat class yang punya logika dasar.
contoh: class kendaraan dengan method umum. Gunakan abstract class jika ada kode dasar yang ingin diwarisi

•) Interface → untuk mendefinisikan kontrak perilaku. Gunakan interface jika hanya butuh kontrak umum

•) sealed class → untuk membatasi pewarisan. berguna saat ingin kontrol lebih pada pewarisan. Gunakan sealed class jika ingin pembatasan pewaris yang ketat dan aman.

5. Bagaimana mekanisme constructor chaining bekerja pada pewarisan java. Apa yang terjadi jika constructor pada superclass tidak dipanggil secara eksplusit?

: constructor chaining adalah memanggil constructor manager memanggil constructor karyawan. Jika constructor manager tidak memanggil super() akan terjadi error.

6. Jelaskan bagaimana penggunaan interface mendukung konsep polymorphism, dan berikan contoh penggunaannya dalam sistem pemesanan makanan online

: Dengan interface, class-class yang berbeda bisa mengimplementasikan interface dengan cara masing-masing. interface mendefinisikan kontrak atau perilaku umum

contoh:

```
interface pemesanan {
    void pesan();
}

class makanan implements pemesanan {
    public void pesan()
    system.out.println("pesan makanan");
  }
}
```