Nama : Ripai Putra Winanda
Nim : 4243250031
Kelas - PSIK 24A
UTS PBO

Essay

1. encapsulation, inheritance, polymorphism, dan abstraction adalah pilar utama dalam pbo, yang saling mendukung untuk menciptakan perangkat lunak yang mudah di kembangkan juga lebih sederhana untuk dipelihara.
1. Abstraction, membantu mengidentifikasi entitas penting dan menyembunyikan detail yang tidak perlu.
2. Encapsulation, menggambil abstraction tsb dan melindungnya dari akses luar yg tidak sah, memastikan integritas data.
3. Inheritance, memungkinkan kita untuk membangun diatas abstraksi dan enkapsulasi yang sudah ada, menciptakan hierarki kelas yang mengurangi duplikasi code dan mempromosikan penggunaan kembali
4. Polymaphism, melengkapi inheritance dgn memungkinkan objek dan kelas yang berbeda namun terkait.

ctt Analogi : Pengelolaan Armada kendaraan perusahaan logistik
1. Abstraction, focus pada inti konsep "kendaraan", yang melakukan sebuah aksi seperti mulai perjalanan (tujuan) dan tidak perlu tahu jenis kendaraan apa yg digunakan
2. Encapsulation, melindungi detail internal kendaraan, seperti detail bagaimana pemeriksaan kendaraan dilakukan, hanya manajer pengelola kendaraan tersebut yang tahu seluk beluk kondisi kendaraan tsb.
3. Inheritance, menciptakan jenis kendaraan spesifik. dalam kendaraan tentu ada mesin, jika ingin perubahan pada mesin, kita tidak perlu membuat ulang mesin, kita bisa menambahkannya, seperti turbo, menambah kecepatan tanpa menggganti mesin.
4. polymorphism, berinteraksi dengan semua jenis kendaraan yg ada melalui antarmuka yg sama, semua kendaraan akan merespon dengan sifat-sifat kendaraannya.

2. Java 21 menawarkan peningkatan performa dan fitur dibanding versi lama.
a. Virtual threads (Project Loom), memungkinkan pembuatan ribuan bahkan jutaan threads dengan overhead minimal. Objek^ya dpt berjalan secara independen dan konkuren lebih mudah
b. Record Pattern & Pattern Matching for Switch, memungkinkan dekonstruksi objek record secara langsung dalam ekspresi switch. yang mengurangi Boilerplate dan manipulasi objek lebih aman dan ekspresif.

3. • Class, adalah blueprint / rancangan yang mendefinisikan karakteristik dan perilaku seperti atribut atau method.
• Object, adalah perwujudan dari class, ketika class dibuat ada Objelt yg menguji class tsb.
Cth :
Untuk mempresentasikan data mahasiswa, diperlukan sebuah tempat (class) untuk menampung data tsb. yg disebut mahasiswa. dalam class mahasiswa tsb mendefinisikan

bahwa mahasiswa memiliki nama, nim dll.
lalu Objelt adalah data dari mahasiswa tersebut yg berisikan nama, nim, prodinya.

4. proses enkapsulasi data balance pada class BankAccount.
* Deklarasi variabel balance sebagai Private, membuat variabel balance hanya bisa diakses langsung dari dalam kelas itu sendiri.
* Menyediakan metode public untuk mengakses (Getter & setter)
↳ Getter, mengembalikan nilai balance, artinya pengguna bisa melihat isi saldo tanpa mengubah isinya
↳ Setter, membuat logika validasi seperti deposit / withdraw, yang memungkinkan bisa mengubah balance secara tidak langsung melalui proses validasi
Encapsulation penting karena memainkan peran krusial pada keamanan sistem, artinya:
* perlindungan data, * Fleksibilitas.
* Kontrol Akses.
* mencegah perubahan yang tidak diinginkan

5. * Panggil eksplisit dengan superclass super(). dan harus menjadi pernyataan pertama dalam konstruktor subclass.
* panggilan implisit, jika tidak menggunakan eksplisit super(), Java akan otomatis menyisipkan panggilan ke constructor default dari superclass.
Eth:
Kelas Karyawan (superclass)
Kelas Manager (subclass dari karyawan).
ketika membuat objek Manager, constructor akan memanggil objek dahulu. pada panggilan pertama adalah constructor karyawan. setelah eksekusi constructor selesai, eksekusi akan kembali ke constructor Manager. jadi constructor chaining memastikan bahwa bagian "karyawan" dari objek "Manager" diinisialisasi terlebih dahulu oleh constructor karyawan, sebelum bagian spesifik "Manager" diinisialisasi oleh constructor manager.

6. sebuah interface adalah kontrak atau blueprint perilaku yg menduking polymorphism dengan:
* menyediakan Common Type.
* Decoupling (pemisahan).
* Flexibilitas
eth dlm pemesanan online.
* pada kelas pemesanan, tidak tergantung pada pembayaran secara spesifik. ia hanya bekerja pada kontrak metode pembayaran.
* jika logika pembayaran dengan eth kartu kredit, berubah, hanya perlu mengubah kelas kartu kredit, yg tidak memengaruhi kelas yang lain.

7. *Abstract class, class yg tidak dapat diinstansi secara langsung, dimaksudkan menjadi kelas dasar bagi kelas-kelas lain.
- cocok digunakan ketika ingin berbagi kode implementasi diantara class yg terkait
- ketika ingin class turunan memiliki banyak metode atau field yang sama

* Interface, tipe referensi yang sepenuhnya abstract. men definisikan sebuah "kontrak" atau sekumpulan perilaku yang harus diimplementasikan oleh class yang mengimplementasikan
- cocok digunakan ketika ingin mendefenisikan kontrak perilaku yang dapat diimplementasikan oleh class** yang tidak terkait
- ketika ingin memanfaatkan multiple ~~en~~ inheritance of type
- ketika ingin menciptakan komponen decoupled.

* Sealed class, konstruksi yang memungkinkan untuk membatasi secara eksplisit class atau interface mana saja yang diizinkan untuk menjadi subclass atau pengimplementasinya
- cocok digunakan ketika ingin secara ketat mengontrol hierarki pewarisan
- ketika merancang domain model tertutup.
- ketika ingin memanfaatkan pattern matching yang aman dan lengkap.