2023

Quiz Paper

OBJECT ORIENTED PROGRAMMING (OOP) PRACTICUM

MOCHAMAD WILDANI AZIZI

2241720171 | 16 | 2I

1. Class dan Object:
   - Apa yang dimaksud dengan "class" dalam pemrograman berorientasi objek?
     Answer: Dalam pemrograman berorientasi objek (OOP), "class" adalah suatu blueprint atau cetak biru untuk menciptakan objek. Class berisi definisi atribut (variabel) dan metode (fungsi) yang mendefinisikan karakteristik dan perilaku dari objek yang akan dihasilkan. Dengan kata lain, class adalah entitas yang membungkus data dan fungsi-fungsi terkait ke dalam satu unit.
   - Bagaimana Anda mendefinisikan objek dari suatu class dalam bahasa pemrograman Java?
     Answer: Dalam bahasa pemrograman Java, kita mendefinisikan objek dari suatu kelas dengan langkah-langkah berikut:
     a. Membuat Class: Pertama-tama, kita perlu membuat kelas yang akan digunakan untuk membuat objek.
     b. Membuat Objek: Setelah kita membuat class, kita dapat membuat objek dari kelas tersebut. Kita perlu menggunakan kata kunci new diikuti oleh nama kelas dan tanda kurung ().
   - Misalkan Anda memiliki class "Barang" dalam sistem informasi inventaris. Bagaimana Anda akan membuat objek "laptop" dari class tersebut?
     Answer: Untuk membuat objek "laptop" dari class "Barang" dalam bahasa pemrograman Java, Anda perlu mengikuti beberapa langkah dasar. Berikut adalah contoh sederhana bagaimana Anda bisa melakukannya:

```java
// Definisi class Barang
public class Barang {
    // Attribut atau properti class
    String nama;
    int jumlah;
    double harga;

    // Konstruktor class
    public Barang(String nama, int jumlah, double harga) {
        this.nama = nama;
        this.jumlah = jumlah;
        this.harga = harga;
    }

    // Metode atau perilaku class
    public void tampilkanInfo() {
        System.out.println("Nama: " + nama);
        System.out.println("Jumlah: " + jumlah);
        System.out.println("Harga: " + harga);
    }
}

// Kelas lain untuk membuat objek laptop
public class Main {
    public static void main(String[] args) {
        // Membuat objek laptop dari class Barang
        Barang laptop = new Barang("Laptop", 1, 1200.00);

        // Memanggil metode untuk menampilkan informasi
```

```
        laptop.tampilkanInfo();
    }
}
```

Pada contoh di atas, kita memiliki class Barang dengan atribut nama, jumlah, dan harga. Terdapat juga konstruktor untuk menginisialisasi objek dan metode tampilkanInfo untuk menampilkan informasi objek.

Dalam kelas Main, kita membuat objek laptop dengan menggunakan konstruktor class Barang dan kemudian memanggil metode tampilkanInfo untuk menampilkan informasi objek laptop tersebut.

2. Encapsulation:
   - Jelaskan konsep encapsulation dalam pemrograman berorientasi objek dan mengapa hal ini penting dalam pengembangan sistem informasi inventaris barang.

     Answer:

     Encapsulation adalah salah satu prinsip dasar dalam pemrograman berorientasi objek (OOP) yang menggabungkan data dan metode yang bekerja pada data ke dalam satu unit tunggal, yang disebut sebagai class. Dengan menggunakan encapsulation, properti dan metode yang terkait dengan suatu objek dapat dienkapsulasi dan diatur hak aksesnya. Artinya, kita dapat menyembunyikan rincian implementasi internal suatu objek dari luar, dan hanya memberikan akses terkontrol ke bagian-bagian tertentu.

     Mengapa Encapsulation Penting dalam Pengembangan Sistem Informasi Inventaris Barang:

     a. Keamanan Data: Encapsulation membantu melindungi data dari modifikasi yang tidak sah. Dengan mengatur hak akses menggunakan modifier akses (seperti private, protected, dan public), Anda dapat memastikan bahwa hanya metode yang diizinkan dapat mengakses atau memodifikasi data.
     b. Abstraksi: Encapsulation memungkinkan pembuatan abstraksi yang mempermudah kompleksitas. Pengguna objek hanya perlu mengetahui metode yang tersedia dan tidak perlu memahami detail implementasi internal.
     c. Fleksibilitas: Dengan enkapsulasi, implementasi internal suatu objek dapat diubah tanpa memengaruhi kode yang menggunakan objek tersebut. Ini memberikan fleksibilitas dalam pengembangan dan pemeliharaan sistem.
     d. Keseluruhan Konsep OOP: Encapsulation adalah salah satu pilar OOP. Dengan memisahkan antara state (data) dan behavior (metode), Anda meningkatkan kemampuan untuk merancang sistem dengan lebih modular dan mudah dimengerti.

   - Dalam konteks sistem informasi inventaris, sebutkan contoh atribut (variabel) yang harus di-encapsulate dan mengapa.

     Answer:

     Contoh Atribut yang Harus Di-Encapsulate dalam Sistem Informasi Inventaris Barang:

Dalam konteks sistem informasi inventaris barang, beberapa contoh atribut yang sebaiknya di-encapsulate meliputi:

a. Jumlah Barang (quantity): Atribut yang menunjukkan jumlah barang dalam inventaris sebaiknya di-encapsulate untuk mengontrol akses dan memastikan bahwa perubahan hanya dapat dilakukan melalui metode tertentu, misalnya, metode penambahan atau pengurangan stok.
b. Harga Barang (price): Informasi harga barang per unit dapat di-encapsulate untuk mengontrol akses dan menghindari perubahan langsung yang tidak sah.
c. Detail Barang (details): Atribut seperti nama, deskripsi, dan atribut terkait lainnya sebaiknya di-encapsulate untuk memberikan kontrol lebih terhadap cara data ini diakses dan dimodifikasi.
d. Dengan cara ini, kita dapat memastikan bahwa operasi-operasi pada atribut-inventaris hanya dapat dilakukan melalui metode yang telah ditentukan, meminimalkan risiko kesalahan dan meningkatkan keamanan data.

3. Relasi Kelas:
   - Apa yang dimaksud dengan relasi antara kelas dalam pemrograman berorientasi objek?
     Answer: Dalam pemrograman berorientasi objek (OOP), relasi antara kelas mengacu pada cara dua atau lebih kelas berinteraksi atau saling berhubungan satu sama lain. Terdapat beberapa jenis relasi antara kelas, di antaranya:

     a. Agregasi: Suatu kelas "mengandung" atau "berisi" objek dari kelas lain. Misalnya, kelas "Mobil" dapat memiliki objek dari kelas "Mesin" sebagai atributnya.
     b. Asosiasi: Hubungan umum antara dua kelas. Ini bisa berupa hubungan satu arah atau dua arah. Contohnya, kelas "Pelanggan" dapat dihubungkan dengan kelas "Pembelian" melalui asosiasi, karena pelanggan dapat melakukan pembelian.
     c. Komposisi: Bentuk khusus dari agregasi di mana objek dari satu kelas adalah bagian dari objek kelas lain dan tidak dapat ada tanpa objek tersebut. Sebagai contoh, sebuah kelas "Ruang" dapat memiliki objek "Meja" sebagai bagian dari komposisi.
     d. Warisan (Inheritance): Suatu kelas dapat mewarisi sifat-sifat dan metode dari kelas lain. Kelas yang mewarisi disebut kelas anak, sementara kelas yang memberikan warisan disebut kelas induk.
   - Dalam sistem informasi inventaris barang, bagaimana Anda akan menggambarkan relasi antara kelas "Barang" dan kelas "Kategori"?
     Answer: Dalam konteks sistem informasi inventaris barang, kelas "Barang" dan kelas "Kategori" dapat memiliki relasi asosiasi. Artinya, barang-barang tertentu dapat terkait dengan kategori tertentu, dan sebaliknya. Sebagai contoh, sebuah barang dapat termasuk dalam kategori "Elektronik" atau "Peralatan Kantor".

4. PBL:
   - Berdasarkan kasus sistem informasi inventaris barang, coba buat sebuah class sederhana beserta atribut dan metodenya yang menggambarkan suatu entitas dalam sistem tersebut (misalnya, class "Barang").
     Answer:
     Berikut adalah contoh sederhana class "Barang" beserta atribut dan metodenya:

```java
public class Barang {
    // Atribut-atribut class
    private String kodeBarang;
    private String nama;
    private int jumlah;
    private double harga;

    // Konstruktor class
    public Barang(String kodeBarang, String nama, int jumlah, double harga) {
        this.kodeBarang = kodeBarang;
        this.nama = nama;
        this.jumlah = jumlah;
        this.harga = harga;
    }

    // Metode untuk menampilkan informasi barang
    public void tampilkanInfo() {
        System.out.println("Kode Barang: " + kodeBarang);
        System.out.println("Nama: " + nama);
        System.out.println("Jumlah: " + jumlah);
        System.out.println("Harga: " + harga);
    }

    // Getter dan setter untuk atribut tertentu (contoh: harga)
    public double getHarga() {
        return harga;
    }

    public void setHarga(double harga) {
        this.harga = harga;
    }
}
```

   - Bagaimana Anda akan menggunakan encapsulation untuk melindungi atribut-atribut dalam class tersebut?
     Answer: Dalam class di atas, atribut-atribut seperti kodeBarang, nama, jumlah, dan harga dijadikan private untuk menerapkan encapsulation. Metode tampilkanInfo digunakan untuk mengekspose informasi barang, sementara getter dan setter digunakan untuk memberikan akses terkontrol ke atribut harga.
   - Gambarkan hierarki class atau hubungan antar class yang mungkin ada dalam sistem informasi inventaris barang di jurusan Teknologi Informasi. Berikan contoh relasi antar class (misalnya, inheritance atau association) dalam konteks tersebut.
     Answer: Hierarki Class atau Hubungan Antar Class:

Dalam sistem informasi inventaris barang di jurusan Teknologi Informasi, mungkin terdapat hierarki class atau hubungan antar class seperti berikut:

a. Class "Barang": Mewakili entitas barang dengan atribut seperti kode barang, nama, jumlah, dan harga.
b. Class "KategoriBarang": Mewakili entitas kategori barang dengan atribut seperti kode kategori dan nama kategori.
c. Class "Inventaris": Mewakili entitas inventaris secara umum dengan atribut seperti tanggal masuk, tanggal keluar, dan metode untuk pencatatan barang.

Contoh relasi antar class:

a. Association (Asosiasi): Class "Inventaris" dapat berhubungan dengan class "Barang" melalui asosiasi, karena inventaris terdiri dari barang-barang tertentu.
b. Inheritance (Warisan): Jika ada hierarki lebih lanjut, misalnya class "Elektronik" yang merupakan turunan dari class "Barang," kita bisa menggunakan inheritance.