Nama : Alfina Dwi Rachmawati

NPM : 5230411288

Matkul : Pemrograman Berbasis Objek (VIII)

1. **Jelaskan perbedaan use case diagram dengan class diagram?**
    - **Use Case Diagram :** Use case diagram menggambarkan interaksi antara pengguna (aktor) dan sistem. Diagram ini berfokus pada apa yang dilakukan oleh sistem, yaitu fungsionalitas yang ditawarkan kepada pengguna. Komponen utama dalam use case diagram adalah aktor, use case, dan hubungan antara keduanya. Biasanya, use case digambarkan dalam bentuk oval, sementara aktor digambarkan sebagai stick figure.
    - **Class Diagram :** Class diagram, di sisi lain, menunjukkan struktur statis dari sistem dengan mendetailkan kelas, atribut, metode, dan hubungan antar kelas. Fokusnya adalah pada bagaimana sistem diorganisir dan struktur data yang digunakan. Dalam class diagram, kelas digambarkan dalam bentuk persegi panjang yang terbagi menjadi bagian untuk nama kelas, atribut, dan metode.

2. **Jelaskan jenis-jenis dependensi?**

    Dependensi Proyek

    - **Definisi**: Terkait dengan hubungan antara berbagai proyek dalam sebuah organisasi.
    - **Contoh**: Proyek A harus selesai sebelum proyek B dapat dimulai. Dalam hal ini, proyek A adalah dependensi untuk proyek B.

    Dependensi Tugas (Task Dependency)

    - **Definisi**: Terkait dengan hubungan antara tugas-tugas dalam satu proyek.
    - **Tipe**:

    - **Finish-to-Start (FS)**: Tugas B tidak dapat dimulai sebelum tugas A selesai.
    - **Start-to-Start (SS)**: Tugas B tidak dapat dimulai sebelum tugas A dimulai.
    - **Finish-to-Finish (FF)**: Tugas B tidak dapat selesai sebelum tugas A selesai.
    - **Start-to-Finish (SF)**: Tugas B tidak dapat selesai sebelum tugas A dimulai (kurang umum).

Dependensi Kode (Code Dependency)

- **Definisi**: Terkait dengan hubungan antara berbagai komponen atau modul dalam kode sumber.
- **Contoh**: Modul A memanggil fungsi dari modul B. Dalam hal ini, modul B adalah dependensi untuk modul A.

Dependensi Perpustakaan (Library Dependency)

- **Definisi**: Ketergantungan pada pustaka eksternal yang digunakan dalam proyek.
- **Contoh**: Jika sebuah proyek menggunakan pustaka pihak ketiga, seperti jQuery atau React, maka pustaka tersebut menjadi dependensi.

Dependensi Lingkungan (Environment Dependency)

- **Definisi**: Ketergantungan pada konfigurasi lingkungan tempat perangkat lunak dijalankan.
- **Contoh**: Sebuah aplikasi web mungkin bergantung pada versi tertentu dari sistem operasi, database, atau server aplikasi.

Dependensi Data (Data Dependency)

- **Definisi**: Ketergantungan pada data yang digunakan oleh berbagai bagian dari perangkat lunak.
- **Contoh**: Sebuah modul yang memerlukan data dari database lain sebelum dapat melakukan operasinya.

Dependensi Waktu (Time Dependency)

- **Definisi**: Terkait dengan waktu pelaksanaan tugas atau proyek.
- **Contoh**: Tugas A harus diselesaikan sebelum batas waktu tertentu agar tugas B dapat dimulai pada waktu yang tepat.

Dependensi Versi (Version Dependency)

- **Definisi**: Terkait dengan ketergantungan pada versi tertentu dari perangkat lunak atau komponen.
- **Contoh**: Aplikasi mungkin memerlukan versi spesifik dari pustaka agar berfungsi dengan baik, seperti Python 3.8 daripada versi lain.

3. Apa perbedaan pemrograman terstruktur dengan berorientasi objek? Jelaskan?

Pemrograman Terstruktur berfokus pada pemecahan masalah menjadi prosedur atau fungsi, dengan data dan logika terpisah. Struktur kontrol seperti loop dan kondisi digunakan untuk mengontrol aliran program.

Pemrograman Berorientasi Objek berfokus pada pemodelan objek dunia nyata yang memiliki atribut (data) dan metode (fungsi). Prinsip utama seperti enkapsulasi, pewarisan, dan polimorfisme digunakan untuk mengorganisir dan menyusun kode secara modular dan mudah dipelihara.

4. Jelaskan konsep objek dan beri contohnya?

Konsep objek dalam pemrograman berorientasi objek (OOP) merujuk pada entitas yang memiliki data (atribut) dan perilaku (metode) tertentu. Objek adalah instansi dari kelas, yang merupakan cetak biru atau template untuk objek-objek yang akan dibuat. Dalam OOP, objek dapat berinteraksi satu sama lain melalui metode, dan mereka dapat memiliki status internal yang menyimpan data.

Contoh: Dalam program penjualan, kita bisa memiliki kelas "Produk" dengan atribut seperti nama, harga, dan stok. Objek produk1 adalah instans dari kelas "Produk" yang mungkin memiliki nama = "Laptop", harga = 10000, dan stok = 50.

5. **Jelaskan jenis-jenis access modifier beri contohnya dalam baris pemrograman?**

   a. Public

   - Anggota yang dideklarasikan dengan `public` bisa diakses dari mana saja, baik dari dalam kelas, subclass, maupun dari kelas lain di luar package.
   - Contoh :

```
public class Mobil {

    public String merk;

    public void nyalakanMesin() {

        System.out.println("Mesin dinyalakan");

    }

}
```

   b. Private

   - Anggota yang dideklarasikan dengan private hanya bisa diakses dari dalam kelas itu sendiri. Kelas lain, bahkan subclass, tidak bisa mengaksesnya.
   - Contoh :

```
public class Mobil {

    private String nomorMesin;

    private void cekNomorMesin() {

        System.out.println("Nomor mesin: " +
nomorMesin);
```

```
    }

}
```

c. Protected

- Anggota yang dideklarasikan dengan `protected` bisa diakses dari kelas yang berada dalam package yang sama dan oleh subclass, bahkan jika subclass berada di package yang berbeda.
- Contoh:

```
public class Mobil {

    protected String warna;

    protected void gantiWarna(String warnaBaru) {

        warna = warnaBaru;

    }

}
```

d. Default

- Jika tidak ada modifier yang diberikan, maka aksesnya bersifat **default** (atau **package-private**), yang berarti hanya bisa diakses oleh kelas dalam package yang sama. Ini tidak bisa diakses dari luar package, bahkan oleh subclass.
- Contoh:

```
class Mobil {

    String model;

    void setModel(String modelBaru) {

        model = modelBaru;
```

```
        }



}
```

6. **Gambarkan contoh pewarisan dalam diagram class?**