LAPORAN RESMI

Praktikum 6 Pengantar Class Diagram

Mata Kuliah: Praktek Pemrograman Berbasis Objek

Disusun oleh:

M. Ainur Ramadhan (3122500047)

2 D3 Teknik Informatika B

Dosen Pengampu: Yanuar Risah Prayogi S.Kom., M.Kom.

PROGRAM STUDI D3 TEKNIK INFORMATIKA

POLITEKNIK ELEKTRONIKA NEGERI SURABAYA

2023/2024

A. TUGAS PENDAHULUAN

1. Sebutkan bagian-bagian dari class diagram!
    - Class (Kelas): Kelas merupakan elemen dasar dalam diagram kelas. Ia digunakan untuk merepresentasikan objek, entitas, atau konsep dalam sistem. Kelas biasanya memiliki atribut (variabel anggota) dan metode (fungsi anggota).

    - Atribut (Attribute): Atribut adalah variabel anggota dari sebuah kelas. Mereka menggambarkan karakteristik atau sifat dari objek yang didefinisikan oleh kelas tersebut. Atribut biasanya memiliki tipe data dan dapat bersifat publik, privat, atau dilindungi.

    - Metode (Method): Metode adalah fungsi anggota dari sebuah kelas. Mereka mendefinisikan perilaku atau tindakan yang dapat dilakukan oleh objek yang merupakan instance dari kelas tersebut.

    - Associations (Asosiasi): Asosiasi menggambarkan hubungan antara dua atau lebih kelas. Ini menunjukkan bagaimana objek dari kelas-kelas tersebut berinteraksi satu sama lain. Asosiasi dapat memiliki arah dan multiplicities (berapa banyak objek yang terlibat dalam hubungan tersebut).

    - Aggregations (Aggregasi): Aggregasi adalah jenis asosiasi yang menggambarkan hubungan bagian-keseluruhan antara sebuah kelas dengan kelas lainnya. Ini menunjukkan bahwa sebuah kelas adalah bagian dari atau memiliki komponen yang merupakan kelas lain.

    - Generalization (Generalisasi): Generalisasi menggambarkan hubungan hierarki antara kelas. Ini menunjukkan bahwa sebuah kelas merupakan turunan atau subclass dari kelas lain yang lebih umum atau superclass.

    - Dependency (Ketergantungan): Ketergantungan menggambarkan ketergantungan antara kelas-kelas di mana satu kelas membutuhkan kelas lainnya dalam konteks tertentu.

    - Interface: Interface adalah kontrak yang menentukan metode-metode yang harus diimplementasikan oleh kelas-kelas yang menggunakannya. Interface digunakan untuk menggambarkan perilaku yang harus diikuti oleh kelas-kelas terkait.

    - Package: Package adalah cara untuk mengelompokkan kelas-kelas terkait dalam unit yang lebih besar dalam diagram kelas. Ini membantu dalam mengorganisasi dan mengelola kompleksitas sistem yang lebih besar.

    - Dependency Line (Garis Ketergantungan): Garis ketergantungan menghubungkan kelas yang memiliki ketergantungan. Ini digunakan untuk menunjukkan bahwa perubahan dalam satu kelas dapat memengaruhi kelas lainnya.

- Multiplicity: Multiplicity menggambarkan berapa banyak objek yang terlibat dalam hubungan asosiasi atau agregasi antara kelas-kelas.

- Stereotype (Stereotipe): Stereotipe adalah tag atau anotasi yang digunakan untuk memberikan informasi tambahan tentang kelas atau elemen UML lainnya. Stereotipe digunakan untuk menambahkan makna khusus pada elemen-elemen tersebut.

2. Sebutkan relasi yang digunakan dalam class diagram!
   - Association (Asosiasi): Asosiasi menggambarkan hubungan yang umum antara dua kelas. Ini adalah relasi dasar yang mengindikasikan bahwa objek dari satu kelas dapat berhubungan dengan objek dari kelas lain. Asosiasi dapat memiliki arah dan multiplicities (berapa banyak objek yang terlibat dalam hubungan tersebut).

   - Aggregation (Aggregasi): Aggregasi adalah jenis asosiasi yang menggambarkan hubungan bagian-keseluruhan antara kelas. Ini menunjukkan bahwa sebuah kelas adalah bagian dari atau memiliki komponen yang merupakan kelas lain. Aggregasi biasanya digambarkan dengan panah yang mengarah ke kelas yang "memiliki" bagian.

   - Composition (Komposisi): Komposisi adalah jenis asosiasi yang lebih kuat daripada aggregasi. Ini menggambarkan hubungan bagian-keseluruhan yang sangat erat, di mana bagian-bagian tersebut hanya ada dalam konteks keseluruhan. Jika objek keseluruhan dihapus, maka bagian-bagian yang terkait juga dihapus.

   - Generalization (Generalisasi atau Inheritance): Generalisasi menggambarkan hubungan hierarki antara kelas-kelas. Ini menunjukkan bahwa sebuah kelas (subclass) adalah turunan dari kelas yang lebih umum (superclass) atau bahwa kelas subclass mewarisi atribut dan metode dari kelas superclass.

   - Realization (Realisasi atau Implementasi): Realisasi menghubungkan sebuah kelas dengan sebuah interface atau abstrak kelas yang harus diimplementasikan oleh kelas tersebut. Ini menunjukkan bahwa kelas tersebut "melaksanakan" atau "mengimplementasikan" perilaku yang didefinisikan oleh interface atau abstrak kelas.

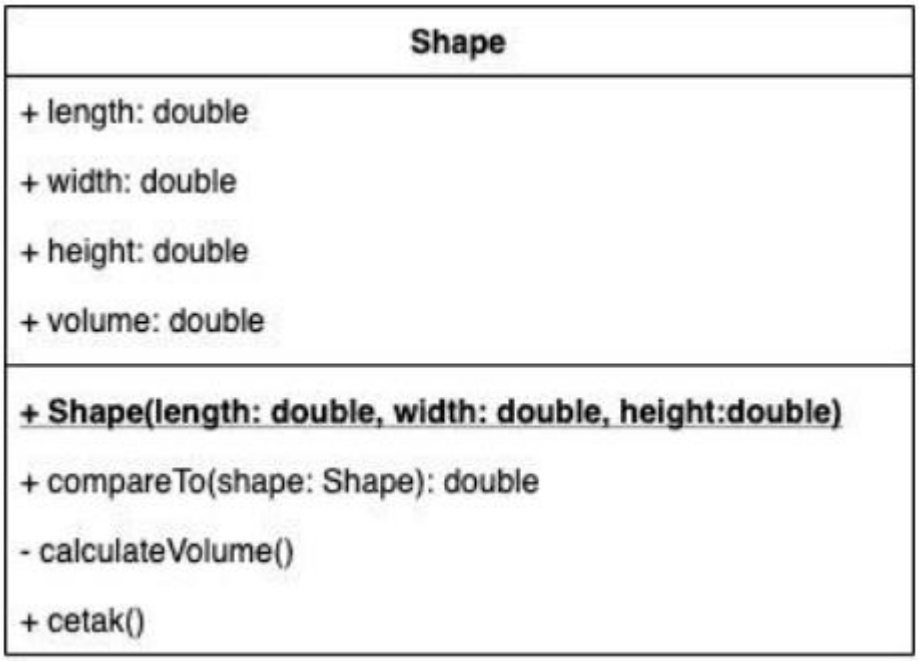
   - Dependency (Ketergantungan): Ketergantungan menggambarkan ketergantungan antara kelas-kelas di mana satu kelas membutuhkan kelas lainnya dalam konteks tertentu. Ketergantungan dapat digunakan untuk menunjukkan bahwa sebuah kelas menggunakan kelas lain atau bahwa perubahan dalam kelas satu dapat memengaruhi kelas lain.

   - Association Class (Kelas Asosiasi): Kelas asosiasi adalah kelas yang digunakan untuk menggambarkan atribut dan perilaku yang terkait dengan asosiasi tertentu

dalam class diagram. Kelas ini biasanya digunakan ketika asosiasi memerlukan informasi tambahan atau atribut.

- N-ary Association (Asosiasi N-ary): Asosiasi N-ary menggambarkan hubungan antara lebih dari dua kelas. Ini digunakan ketika ada keterhubungan yang kompleks antara beberapa kelas.

B. LATIHAN



Kemudian buatlah class berikut!

```
public class ShapeTester{
      public static void main(String args[]){
            Shape kubus = new Shape(5, 5, 5);
            Shape balok = new Shape(10, 5, 5);
            kubus.cetak();
            balok.cetak();
            Double selisihVolume = kubus.compareTo(balok);
            System.out.println("Selisih Volume =
"+selisihVolume);
      }
}
```

Output dari program itu adalah sebagai berikut:

Panjangxlebarxtinggi = 5x5x5

Volume = 125

Panjangxlebarxtinggi = 10x5x5

Volume = 250

Selisih Volume = 125

Kode Program ➔ *Shape.java*

```
public class Shape{
    public double length;
    public double width;
    public double height;
    public double volume;

    public Shape (double length, double width, double height){
        this.length = length;
        this.width = width;
        this.height = height;
        calculateVolume();
    }

    public double compareTo(Shape shape){
        return Math.abs(this.volume - shape.volume);
    }

    public void calculateVolume(){
        this.volume = length * width * height;
    }

    public void cetak() {
        System.out.println("PanjangxLebarxTinggi: " + (int) length + "x" +
(int) width + "x" + (int) height );
        System.out.println("Volume = " + (int) volume);
    }
}
```

Analisa Kode Program :

1. **Atribut**: Class **Shape** memiliki empat atribut **length**, **width**, **height**, dan **volume** yang digunakan untuk menyimpan informasi tentang dimensi objek dan hasil perhitungan volume.
2. **Constructor**: Terdapat constructor **Shape** yang menerima tiga parameter (**length**, **width**, dan **height**) untuk menginisialisasi atribut-atribut tersebut saat objek **Shape** dibuat. Selain menginisialisasi atribut, constructor juga memanggil metode **calculateVolume()** untuk menghitung volume objek.

3. **Metode compareTo**: Metode ini digunakan untuk membandingkan dua objek **Shape** berdasarkan selisih volume antara keduanya. Hasilnya adalah selisih volume yang diambil nilai absolutnya dengan menggunakan **Math.abs**. Ini adalah metode untuk membandingkan dua objek **Shape** berdasarkan volume mereka.
4. **Metode calculateVolume**: Metode ini digunakan untuk menghitung volume objek **Shape** berdasarkan panjang, lebar, dan tinggi yang telah diinisialisasi. Hasil perhitungan disimpan dalam atribut **volume**.
5. **Metode cetak**: Metode ini digunakan untuk mencetak informasi tentang objek **Shape** ke layar. Ini mencakup mencetak nilai panjang, lebar, tinggi, dan volume dalam format yang telah ditentukan.
6. Penggunaan **Math.abs** dalam metode **compareTo** dan penggunaan **(int)** dalam metode **cetak** digunakan untuk memastikan bahwa nilai-nilai yang dicetak adalah bilangan bulat positif.

Untuk kode program ShapeTester terdapat sedikit yang saya rubah, agar saya mendapatkan output yang sesuai dengan perintah soal.

Berikut kode programnya ➔ *ShapeTester.java*

```
public class ShapeTester{
    public static void main(String args[]){
        Shape kubus = new Shape(5, 5, 5);
        Shape balok = new Shape(10, 5, 5);
        kubus.cetak();
        balok.cetak();
        double selisihVolume = kubus.compareTo(balok);
        System.out.println("Selisih Volume = " + (int) selisihVolume);
    }
}
```

Analisa Kode Program :
Kode program ini melakukan pengujian terhadap class **Shape** yang telah Anda buat, khususnya untuk menghitung dan mencetak selisih volume antara objek **kubus** dan **balok**. Program ini membantu memastikan bahwa class **Shape** berfungsi dengan benar dalam menghitung dan mencetak informasi objek-objeknya.

Berikut Output nya :

```
C:\Users\USER\OOP\Praktikum\Praktikum 6\Latihan>javac *.java

C:\Users\USER\OOP\Praktikum\Praktikum 6\Latihan>java ShapeTester
PanjangxLebarxTinggi: 5x5x5
Volume = 125
PanjangxLebarxTinggi: 10x5x5
Volume = 250
Selisih Volume = 125
```

C. TUGAS

1. Class Diagram 1 ➔ Person Class

Person
name: String age: int
setName(name: String) setAge(age: int) getName(): String getAge(): int

Kode Program: *Person.java*

```java
import java.util.Scanner;

public class Person {
    String name;
    int age;

    public void setName(String name) {
        this.name = name;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public String getName() {
        return name;
    }

    public int getAge() {
        return age;
    }

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        Person person = new Person();

        System.out.print("Masukkan Nama: ");
        String nameInput = input.nextLine();
        person.setName(nameInput);

        System.out.print("Masukkan Umur: ");
        int ageInput = input.nextInt();
        person.setAge(ageInput);

        System.out.println("\nInformasi Person:");
```

```
        System.out.println("Nama: " + person.getName());
        System.out.println("Umur: " + person.getAge());

        input.close();
    }
}
```

Analisa kode Program:
Program tersebut adalah program Java yang mendefinisikan kelas "Person" dengan atribut "name" dan "age," serta metode setter dan getter untuk mengatur dan mengambil nilai atribut tersebut. Program ini juga memiliki metode "main" yang memungkinkan pengguna untuk memasukkan nama dan umur, kemudian menampilkan informasi tersebut ke layar.

Output:

```
C:\Users\USER\OOP\Praktikum\Praktikum 6\Tugas>javac *.java

C:\Users\USER\OOP\Praktikum\Praktikum 6\Tugas>java Person
Masukkan Nama: Rama
Masukkan Umur: 18

Informasi Person:
Nama: Rama
Umur: 18
```

2. Class Diagram 2 ➔ Alamat Class

MataKuliah
Jalan: String Kota: String
Alamat(jalan: String, kota: String) getJalan(): String getKota(): String setJalan(jalan: String) setKota(kota: String)

Kode Program: *Alamat.java*

```
import java.util.Scanner;

public class Alamat {
    String jalan;
    String kota;

    public Alamat(String jalan, String kota) {
        this.jalan = jalan;
        this.kota = kota;
    }

    public String getJalan() {
        return jalan;
```

```java
    }

    public String getKota() {
        return kota;
    }

    public void setJalan(String jalan) {
        this.jalan = jalan;
    }

    public void setKota(String kota) {
        this.kota = kota;
    }

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        System.out.print("Masukkan alamat: ");
        String jalanInput = input.nextLine();

        System.out.print("Masukkan kota: ");
        String kotaInput = input.nextLine();

        Alamat alamat = new Alamat(jalanInput, kotaInput);

        System.out.println("Alamat: " + alamat.getJalan());
        System.out.println("Kota: " + alamat.getKota());

        input.close();
    }
}
```

Analisa Kode Program:
Kode program Java tersebut mendefinisikan kelas "Alamat" dengan atribut "jalan" dan "kota" serta memiliki metode setter dan getter untuk mengelola nilai atribut tersebut. Program ini juga memungkinkan pengguna untuk memasukkan alamat dan kota melalui keyboard, kemudian menampilkan informasi tersebut.

Output

```
C:\Users\USER\OOP\Praktikum\Praktikum 6\Tugas>javac *.java

C:\Users\USER\OOP\Praktikum\Praktikum 6\Tugas>java Alamat
Masukkan alamat: Jl. Platuk Donomulyo No.1b, Sidotopo Wetan, Kec. Kenjeran
Masukkan kota: Surabaya
Alamat: Jl. Platuk Donomulyo No.1b, Sidotopo Wetan, Kec. Kenjeran
Kota: Surabaya
```

3. Class Diagram 3 ➔ MataKuliah Class

MataKuliah
kode: String nama: String sks: int dosen: Sring
MataKuliah(kode: String, nama: String, sks: int, dosen: String) getKode(): String getNama(): String getSKS(): int getDosen: String setKode(kode: String) setNama(nama: String) setSKS(sks: int) setDosen(dosen: String)

Kode Program: *MataKuliah.java*

```
import java.util.Scanner;

public class MataKuliah {
    private String kode;
    private String nama;
    private int sks;
    private String dosen;

    public MataKuliah(String kode, String nama, int sks, String
dosen) {
        this.kode = kode;
        this.nama = nama;
        this.sks = sks;
        this.dosen = dosen;
    }

    public String getKode() {
        return kode;
    }

    public String getNama() {
        return nama;
    }

    public int getSKS() {
        return sks;
    }

    public String getDosen() {
        return dosen;
    }
```

```
    public void setKode(String kode) {
        this.kode = kode;
    }

    public void setNama(String nama) {
        this.nama = nama;
    }

    public void setSKS(int sks) {
        this.sks = sks;
    }

    public void setDosen(String dosen) {
        this.dosen = dosen;
    }

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        System.out.print("Masukkan Kode Mata Kuliah: ");
        String kode = input.nextLine();

        System.out.print("Masukkan Nama Mata Kuliah: ");
        String nama = input.nextLine();

        System.out.print("Masukkan Jumlah SKS: ");
        int sks = input.nextInt();
        input.nextLine(); // Mengonsumsi newline

        System.out.print("Masukkan Nama Dosen Pengajar: ");
        String dosen = input.nextLine();

        MataKuliah matkul = new MataKuliah(kode, nama, sks, dosen);

        System.out.println("\nInformasi Mata Kuliah:");
        System.out.println("Kode Mata Kuliah: " + matkul.getKode());
        System.out.println("Nama Mata Kuliah: " + matkul.getNama());
        System.out.println("SKS: " + matkul.getSKS());
        System.out.println("Dosen Pengajar: " + matkul.getDosen());

        input.close();

    }
}
```

Analisa Kode Program:
Kode program java tersebut mendefinisikan kelas "MataKuliah" untuk merepresentasikan informasi mata kuliah dengan atribut seperti kode mata kuliah,

nama mata kuliah, jumlah SKS, dan nama dosen pengajar. Program ini memungkinkan pengguna untuk memasukkan informasi mata kuliah melalui keyboard dan kemudian menampilkannya kembali.

Output:

```
C:\Users\USER\OOP\Praktikum\Praktikum 6\Tugas>javac *java

C:\Users\USER\OOP\Praktikum\Praktikum 6\Tugas>java MataKuliah
Masukkan Kode Mata Kuliah: G200JI
Masukkan Nama Mata Kuliah: Praktek Pemrograman Berorientasi Objek
Masukkan Jumlah SKS: 2
Masukkan Nama Dosen Pengajar: Yanuar Risah Prayogi

Informasi Mata Kuliah:
Kode Mata Kuliah: G200JI
Nama Mata Kuliah: Praktek Pemrograman Berorientasi Objek
SKS: 2
Dosen Pengajar: Yanuar Risah Prayogi
```

D. KESIMPULAN

Dalam rangka pengembangan perangkat lunak berorientasi objek, class diagram adalah alat yang sangat penting untuk perencanaan, pemodelan, dan komunikasi yang efektif di antara tim pengembangan. Ini membantu dalam membangun sistem yang lebih terstruktur, mudah dipahami, dan lebih mudah dipelihara.