

Laporan Praktikum

MODUL 7

Relasi Antar Kelas



Institut Teknologi Telkom Purwokerto

Nama :

Nathaya Elang Mariantaka (2211103128)

Nama Dosen :

Nicolaus Euclides Wahyu Nugroho, S.Kom., M.Cs.

**PROGRAM STUDI S1 SISTEM INFORMASI
FAKULTAS INFORMATIKA**

**INSTITUT TEKNOLOGI TELKOM PURWOKERTO
2023**

BAB I. TUJUAN

I. Tujuan

1. Mengerti dan memahami tentang konsep relasi kelas
2. Mampu mewujudkan berbagai jenis relasi kelas dalam Java

II. Tool

- IntelliJ 2023

III. Dasar Teori

Pemrograman OO (Object-Oriented) mengambil realita masalah dalam kehidupan sehari-hari. Dalam OO, sering terjadi relasi antara satu objek dengan objek yang lain.

Ada beberapa relasi yang mungkin terjadi antara suatu kelas dengan kelas yang lain, yaitu :

a. Inheritance / pewarisan

Terdiri dari pewarisan tunggal (single inheritance) dan pewarisan jamak (multiple inheritance) yang telah dijelaskan pada modul sebelumnya.

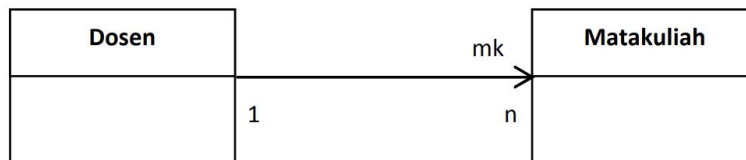
b. Agregasi

Agregasi adalah relasi antara dua buah objek dengan mengatakan bahwa satu objek memiliki atau mengandung atau berisi objek yang lain. Misalnya, sebuah mobil memiliki mesin, roda, body; sebuah rumah memiliki dapur, kamar mandi, kamar makan. Apabila suatu objek tertentu tersusun atas objek-objek lain, maka objek tersebut dikatakan sebagai objek agregat atau objek komposit (aggregate, composite object). Relasi ini sering disebut juga dengan relasi 'has-a' atau relasi 'whole-part'. Dalam diagram UML, relasi agregat ini digambarkan dengan simbol diamond. Simbol ini menunjukkan adanya inklusi struktural sebuah objek terhadap objek yang lain yang diwujudkan dengan cara **membuat kelas agregat yang memiliki atribut yang bertipe kelas penyusun.**



c. Asosiasi

Relasi asosiasi menyatakan suatu hubungan struktural antara beberapa objek yang menggambarkan objek dari suatu kelas dihubungkan dengan objek lain. Relasi ini **menunjukkan variabel dalam suatu kelas yang menyimpan rujukan bertipe kelas lain**. Diagram berikut menunjukkan relasi asosiasi antara kelas Dosen dan kelas Matakuliah, dengan arah panah asosiasi dari kelas Dosen menuju kelas Matakuliah yang menunjukkan bahwa Dosen menyimpan rujukan ke Matakuliah.



Nama yang berada di anak panah (mk) merupakan role name yang kelak akan menjadi atribut dari kelas Dosen.

Pada saat kita akan merancang relasi kelas, ada beberapa hal yang perlu diperhatikan, yaitu :

- Ada berapa objek dari masing-masing kelas yang terlibat dalam relasi.
- Apakah relasi tersebut bersifat wajib (mandatory) atau optional.

Dalam implementasi, asosiasi secara sintaks tidak memiliki perbedaan dengan implementasi agregasi, kecuali asosiasi bersifat dua arah.

d. Dependency

Relasi kebergantungan (dependency) menyatakan bahwa suatu kelas bergantung pada kelas yang lain, maka perubahan pada kelas yang menjadi ketergantungan dari kelas lain menyebabkan perubahan terhadap kelas yang tergantung tersebut. Misalnya, kelas tumbuhan membutuhkan kelas air, jika kelas air mengalami perubahan, maka menyebabkan perubahan pada kelas tumbuhan. Relasi dependency dapat digambarkan dengan diagram UML sebagai berikut :



Perwujudan relasi ini dapat dilakukan dalam 3 bentuk :

1. Penggunaan kelas B sebagai parameter pada fungsi di kelas A.
2. Penggunaan kelas B sebagai nilai balikan pada fungsi di kelas A.
3. Penggunaan kelas B sebagai variabel lokal pada fungsi di kelas A.

Keterangan : Kelas A adalah kelas yang bergantung pada kelas B.

IV. GUIDED

1. Relasi Asosiasi

IbuAnak.java

```
package GUIDED;

public class IbuAnak
{
    public static void main(String []args)
    {
        Manusia ibu1 = new Manusia("Budi", 30);
        Manusia anak1 = new Manusia("Ani", 4);
        Manusia ibu2 = new Manusia("Diana", 40 );
        Manusia anak2 = new Manusia("Andi", 5, ibu2);
        //Relasi antara Manusia dengan Manusia
        System.out.println("=====\\n");
        ibu1.cetak();
        anak1.cetak();
        System.out.println("=====\\n");
        ibu1.adopsi(anak1);
        ibu1.cetak();
        anak1.cetak();
        System.out.println("=====\\n");
        ibu2.cetak();
        anak2.cetak();
    }
}
```

Manusia.java

```
package GUIDED;

public class Manusia
{
    private String nama;
    private int umur;
    private Manusia ibu;
    private Manusia anak;
    //Letak asosiasi, kelas anak jadi variabel
    public Manusia() { }
    public Manusia(String nm, int umr)
    {
        nama = nm;
        umur = umr;
        ibu = new Manusia();
        anak = new Manusia();
        ibu=null;
        anak=null;
    }
    public Manusia(String nm, int umr, Manusia ibu_angkat)
    {
        nama = nm;
        umur = umr;
        ibu = new Manusia();
        anak = new Manusia();
        ibu=ibu_angkat;
        ibu_angkat.anak=this;
    }
    //Relasi yang menunjukkan asosiasi
    public void adopsi(Manusia anak_angkat)
    {
        anak = anak_angkat;
        anak_angkat.ibu = this;
    }
    public void cetak()
    {
        System.out.println("\n- Data Pribadi -");
        System.out.println("Nama : "+nama);
        System.out.println("Umur : "+umur);
        if(ibu!=null)
            System.out.println("Nama ibu : "+ibu.nama);
        else if(anak!=null)
            System.out.println("Nama anak : "+anak.nama);
    }
}
```

```
- Data Pribadi -  
Nama : Budi  
Umur : 30  
  
- Data Pribadi -  
Nama : Ani  
Umur : 4  
=====
```

```
- Data Pribadi -  
Nama : Budi  
Umur : 30  
Nama anak : Ani  
  
- Data Pribadi -  
Nama : Ani  
Umur : 4  
Nama ibu : Budi  
=====
```

```
- Data Pribadi -  
Nama : Diana  
Umur : 40  
Nama anak : Andi  
  
- Data Pribadi -  
Nama : Andi  
Umur : 5  
Nama ibu : Diana
```

PENJELASAN : Program ini memberikan gambaran sederhana tentang interaksi antar objek dalam konteks hubungan keluarga, termasuk perubahan hubungan melalui adopsi. Meskipun implementasi memiliki beberapa potensi kebingungan dan masalah logika, hal tersebut memberikan pemahaman tentang bagaimana konsep adopsi dapat diimplementasikan dalam pemrograman berorientasi objek.

2. Relasi Agregasi

Perusahaan.java

```
package GUIDED;
```

```
// Class Perusahaan
```

```
public class Perusahaan
```

```
{
```

```
    private String namaPer;
```

```
    private Pegawai peg[]; //Agregasi antara pegawai dan perusahaan
```

```
    private int counter;
```

```
    public Perusahaan(String namaPer)
```

```
    {
```

```
        this.namaPer=namaPer;
```

```
        counter=0;
```

```
        peg = new Pegawai[3];
```

```
        System.out.println("Konstruktor perusahaan dijalankan...");
```

```
    }
```

```
    //Relasi agregasi antara pegawai dan perusahaan
```

```
    public void insertPegawai(Pegawai p)
```

```
    {
```

```
        peg[counter]=p;
```

```
        counter++;
```

```
    }
```

```
    public void tampilPer()
```

```
    {
```

```
        System.out.println("Perusahaan "+namaPer+" memiliki pegawai : ");
```

```
        for(int i=0; i<counter; i++)
```

```
        {
```

```
            peg[i].tampilPeg();
```

```
        }
```

```
    }
```

```
}
```

Pegawai.java

```

package GUIDED;

// Class Pegawai
public class Pegawai
{
    private String nama;
    private String NIP;
    public Pegawai()
    {
        System.out.println("Konstruktor pegawai dijalankan...");
    }
    public Pegawai(String NIP, String nama)
    {
        this.NIP=NIP;
        this.nama=nama;
        System.out.println("Konstruktor pegawai dijalankan...");
    }
    public void tampilPeg()
    {
        System.out.println("NIP : "+NIP+" ,Nama : "+nama);
    }
}

```

Relasi.java

```

package GUIDED;

public class Relasi
{
    public static void main(String[] args)
    {
        Perusahaan Per = new Perusahaan("Nusantara Jaya");
        Pegawai Peg1, Peg2, Peg3;
        Peg1 = new Pegawai("P001", "Rudi");
        Peg2 = new Pegawai("P002", "Toni");
        Peg3 = new Pegawai("P003", "Ani");
        //Relasi antara Class Pegawai dengan Class Perusahaan
        Per.insertPegawai(Peg1);
        Per.insertPegawai(Peg2);
        Per.insertPegawai(Peg3);
        System.out.println();
        Per.tampilPer();
    }
}

```



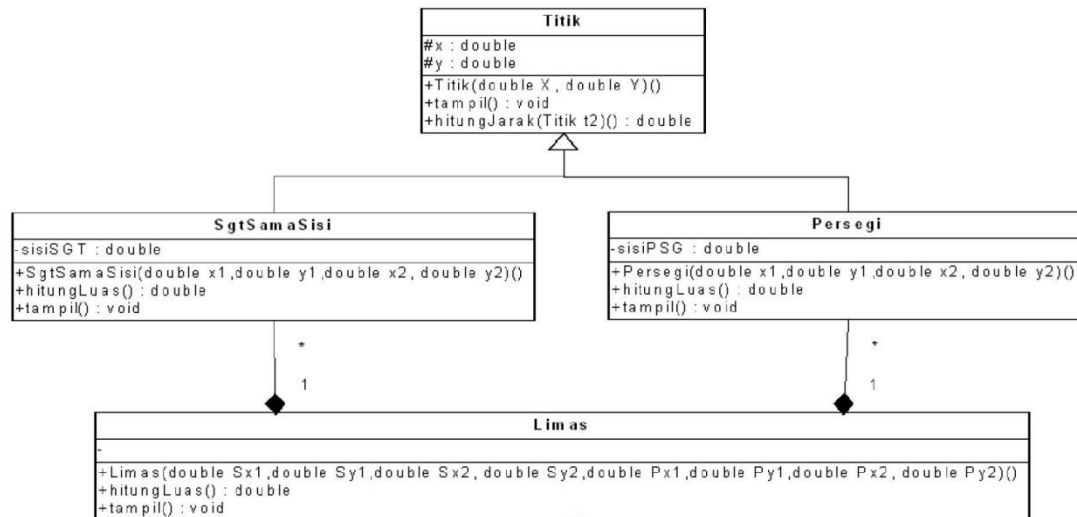
```
Konstruktor perusahaan dijalankan...
Konstruktor pegawai dijalankan...
Konstruktor pegawai dijalankan...
Konstruktor pegawai dijalankan...

Perusahaan Nusantara Jaya memiliki pegawai :
NIP : P001 ,Nama : Rudi
NIP : P002 ,Nama : Toni
NIP : P003 ,Nama : Ani
```

PENJELASAN : Program ini memberikan gambaran sederhana tentang hubungan antara perusahaan dan pegawai. Implementasi menunjukkan cara objek pegawai dapat dihubungkan dengan objek perusahaan melalui metode insertPegawai. Pencetakan informasi perusahaan dan daftar pegawai memberikan gambaran tentang bagaimana relasi ini dapat direpresentasikan dalam pemrograman berorientasi objek.

V. UNGUIDED

Buatlah kelas Limas yang terdiri dari kelas SgtSamaSisi dan kelas Persegi. Kelas SgtSamaSisi dan kelas Persegi adalah turunan dari kelas Titik.



Kelas Titik memiliki spesifikasi sebagai berikut :

- Memiliki atribut garis x dan y bertipe double (5%)
- Memiliki method `tampil()` untuk menampilkan nilai atribut kelas Titik (10%)
- Memiliki method `hitungJarak(Titik t2)` untuk menampilkan nilai atribut kelas Titik (10%)

Kelas SgtSamaSisi memiliki spesifikasi sebagai berikut :

- Memiliki atribut `sisiSGT` bertipe double (5%)
- `sisiSGT` didapatkan dari perhitungan jarak 2 titik yang diinputkan melalui konstruktornya.
- Memiliki method `hitungLuas()` untuk menghitung luas segitiga (10%)
 - Memiliki method `tampil()` yang meredefinisikan method kelas induk untuk menampilkan nilai atribut kelas induk dan nilai atribut kelas `SgtSamaSisi` (10%)

Kelas Persegi memiliki spesifikasi sebagai berikut :

- Memiliki atribut `sisiPSG` bertipe double (5%)
- `sisiPSG` didapatkan dari perhitungan jarak 2 titik yang diinputkan melalui konstruktornya.
- Memiliki method `hitungLuas()` untuk menghitung luas persegi (10%)
 - Memiliki method `tampil()` yang meredefinisikan method kelas induk untuk menampilkan nilai atribut kelas induk dan nilai atribut kelas `Persegi` (10%)

Kelas Limas memiliki spesifikasi sebagai berikut :

- Limas terdiri dari 4 buah SgtSamaSisi dengan ukuran yang sama dan 1 buah persegi (5%)
- Memiliki method hitungLuas() untuk menghitung luas Limas. (10%)
- Memiliki method tampil() untuk menampilkan nilai atribut SgtSamaSisi, nilai atribut Persegi dan Luas Permukaan Limas (10%)

Ketentuan Program :

1. Program diimplementasikan dalam Java dengan menggunakan prinsip Relasi Kelas.

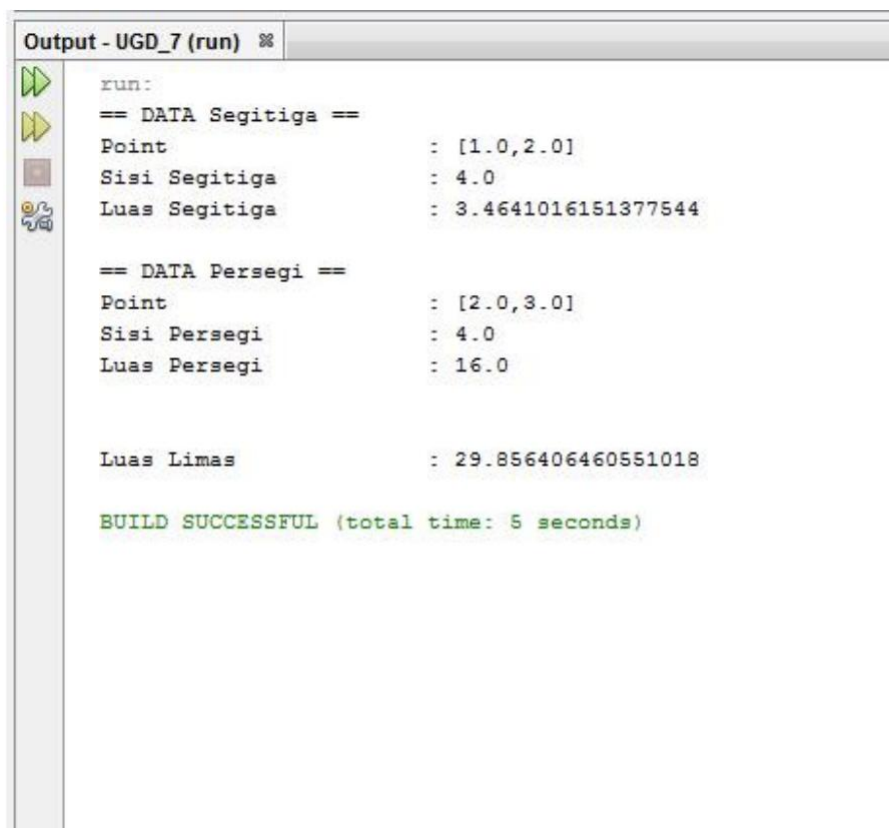
2. Cara Input Data -> Melalui konsturktornya

Limas P = new Limas(1.0,2.0,5.0,2.0,2.0,3.0,2.0,7.0);

3. Rumus : Diasumsikan Titik1(x1,y1) dan Titik2(x2,y2)

Jarak 2 Titik : $\sqrt{(x2 - x1)^2 + (y2 - y1)^2}$
 Luas Persegi : sisiPSG * sisi PSG
 Luas Segitiga Sama Sisi : $0.5 * sisiSGT * 3$
 Luas Limas : $(4 * Luas Segitiga) + Luas Persegi$

4. Perhatikan Pengujian Unguidednya !! Unguided yang berhasil harus dapat dieksekusi sehingga tertampil seperti tampilan berikut ini.



```

Output - UGD_7 (run) ✖
run:
== DATA Segitiga ==
Point           : [1.0,2.0]
Sisi Segitiga   : 4.0
Luas Segitiga   : 3.4641016151377544

== DATA Persegi ==
Point           : [2.0,3.0]
Sisi Persegi    : 4.0
Luas Persegi    : 16.0

Luas Limas      : 29.856406460551018

BUILD SUCCESSFUL (total time: 5 seconds)
  
```

4. Sertakan comment yang terdiri dari NAMA dan NIM di bagian atas program Anda

Titik.java

```
package UNGUIDED;

public class Titik {
    private double x1, x2;
    private double y1, y2;

    // konstruktor
    public Titik(double x1, double y1, double x2, double y2) {
        this.x1 = x1;
        this.y1 = y1;
        this.x2 = x2;
        this.y2 = y2;
    }

    public String koordinat() {
        return "[" + x1 + ", " + y1 + "]";
    }

    public static double hitungJarak(Titik t2) {
        double xx = t2.x2 - t2.x1;
        double yy = t2.y2 - t2.y1;
        return Math.sqrt((xx * xx) + (yy * yy));
    }

    public void tampilTitik() {
        System.out.println("Point : " + koordinat());
    }
}
```

SgtSamaSisi.java

```
package UNGUIDED;
```

```
public class SgtSamaSisi extends Titik {  
    private double sisiSegitiga;
```

```
    // konstruktor
```

```
    public SgtSamaSisi(double x1, double y1, double x2, double y2) {  
        super(x1, y1, x2, y2);  
        this.sisiSegitiga = Titik.hitungJarak(this);  
    }
```

```
    // menghitung jarak antar titik
```

```
    public void jtSegitigaSamaSisi() {  
        Titik.hitungJarak(this);  
    }
```

```
    // menghitung luas segitiga sama sisi
```

```
    public double luasSegitigaSamaSisi() {  
        return 0.5 * sisiSegitiga * Math.sqrt(3);  
    }
```

```
    public void tampil() {
```

```
        System.out.println(" ");
```

```
        System.out.println("====DATA Segitiga====");
```

```
        tampilTitik();
```

```
        System.out.println("Sisi Segitiga : " + sisiSegitiga);
```

```
        System.out.println("Luas Segitiga : " + luasSegitigaSamaSisi());
```

```
        System.out.println(" ");
```

```
    }
```

```
}
```

Persegi.java

```
package UNGUIDED;

public class Persegi extends Titik{
    private double sisiPersegi = super.hitungJarak(this);
    //konstruktor
    public Persegi(double x1, double y1, double x2, double y2) {
        super(x1, y1, x2, y2);
    }
    //menghitung jarak antar titik
    public void jaraktitikPersegi() {
        super.hitungJarak(this);
    }
    public double luasPersegi() {
        return sisiPersegi * sisiPersegi;
    }
    public void tampil() {
        System.out.println("====DATA Persegi====");
        tampilTitik();
        System.out.println("sisi Persegi : " + sisiPersegi);
        System.out.println("luas Persegi : " + luasPersegi());
        System.out.println(" ");
    }
}
```

Limas.java

```

package UNGUIDED;

public class Limas {
    SgtSamaSisi segitiga;
    Persegi persegi;
    double luasSegitiga, luasPersegi;

    // konstruktor
    public Limas(double Sx1, double Sy1, double Sx2, double Sy2, double Px1, double
Py1, double Px2, double Py2) {
        segitiga = new SgtSamaSisi(Sx1, Sy1, Sx2, Sy2);
        luasSegitiga = segitiga.luasSegitigaSamaSisi();
        persegi = new Persegi(Px1, Py1, Px2, Py2);
        luasPersegi = persegi.luasPersegi();
    }

    public double luasLimas() {
        return (4 * luasSegitiga) + luasPersegi;
    }

    public void tampil() {
        segitiga.tampil();
        persegi.tampil();
        System.out.println("Luas Limas : " + luasLimas());
    }
}

```

Relasi.java

```

package UNGUIDED;

```

```

public class Relasi {
    public static void main(String[] args) {
        System.out.println();
        System.out.println("2211103128 NATHAYA ELANG MARIANTAKA");
        Limas limas = new Limas(3.0, 6.0, 1.0, 9.0, 8.0, 1.0, 2.0, 9.0);
        limas.tampil(); // memanggil method tampil dari objek limas
    }
}

```

```
"C:\Program Files\Java\jdk-21\bin\java.exe"

2211103128 NATHAYA ELANG MARIANTAKA

====DATA Segitiga====
Point : [ 3.0, 6.0]
Sisi Segitiga : 3.605551275463989
Luas Segitiga : 3.1224989991991987

====DATA Persegi====
Point : [ 8.0, 1.0]
sisi Persegi : 10.0
luas Persegi : 100.0

Luas Limas : 112.4899959967968
```

PENJELASAN : Program ini menggunakan konsep pemrograman berorientasi objek dengan pewarisan untuk merepresentasikan objek-objek geometris seperti titik, segitiga sama sisi, persegi, dan limas. Program ini melakukan perhitungan geometris seperti menghitung jarak antar titik, luas segitiga sama sisi, luas persegi, dan luas limas, serta menampilkan informasi tersebut ke konsol. Keseluruhan, program ini memberikan contoh implementasi sederhana dari konsep-konsep dasar dalam pemrograman berorientasi objek.

LINK REPOSITORY GIT HUB : <https://github.com/NathayaElang/Laprak-PBO-Modul7>