MODUL PERKULIAHAN

PEMROGRAMAN VI (Object II)

Relationship menggunakan Java & Sub Class dan Super Class

Fakultas	Program Studi	Tatap Muka	Kode MK	Disusun Oleh
Teknik	Informatika	12	06610005	Ardiles Sinaga, S.T., M.T. Kurnia Jaya Eliazar, S.T., M.T.

Abstract

Modul Pertemuan 12 Berisi Mengenai Relationship menggunakan Java & Sub Class dan Super Class.

Kompetensi

Mahasiswa memiliki kemampuan untuk menerapkan konsep Relationship menggunakan Java, Sub Class dan Super Class, dan Studi Kasus.

Pertemuan 13 – Relationship menggunakan Java & Sub Class dan Super Class

Memahami Asosiasi/Relationship

Membuat program menggunakan teknik object oriented pada intinya adalah membuat class yang akan digunakan oleh objek. Adakalanya pada saat membuat program tidak cukup menggunakan satu class saja atau dengan maksud lain adalah memisahkan satu class menjadi beberapa class yang lain. Pemisahan class diperlukan karena antar objek dari sebuah class tidaklah sama, baik dari segi atribut ataupun dari methodenya.

Beberapa class pada satu program tidak berdiri sendiri. Melainkan saling berhubungan antara satu class dengan class yang lain. Tentunya, penghubung antar class adalah menggunakan objek dari masing-masing class. Hubungan antar class ini lah yang disebut dengan relasi/relationship. Relasi ini akan menunjukkan tingkatan relasi antara satu class dengan class yang lain. Relasi pada pemrograman berorientasi objek relasi antar class di sebut dengan ASOSIASI.

Antara pemrograman terstruktur dan pemrograman berorientasi objek mempunyai istilah yang berbeda untuk menyatakan tingkatan relasi antar class/entitas .

Jika pada pemrograman terstruktur tingkatan relasi disebut dengan CARDINALITY. Pada pemrograman berorientasi objek tingkatan relasi disebut dengan MULTIPLICITY.

CARDINALITY pada pemrograman terstruktur digunakan untuk menyatakan tingkatan relasi antar entitas pada ENTITY RELATIONSHIP DIAGRAM (ERD) / DIAGRAM ER. Umumnya ERD mempunyai tiga tingkatan relasi yaitu :

a. 1 .. 1, Satu Entitas hanya memiliki satu relasi pada entitas lain.
b. 1 .. M, Satu Entitas memiliki satu atau banyak relasi pada entitas yang lain dengan entitas yang sama.
c. M .. N, Satu Entitas memiliki satu atau banyak relasi pada entitas yang lain dengan entitas yang sama dan begitu juga sebaliknya.

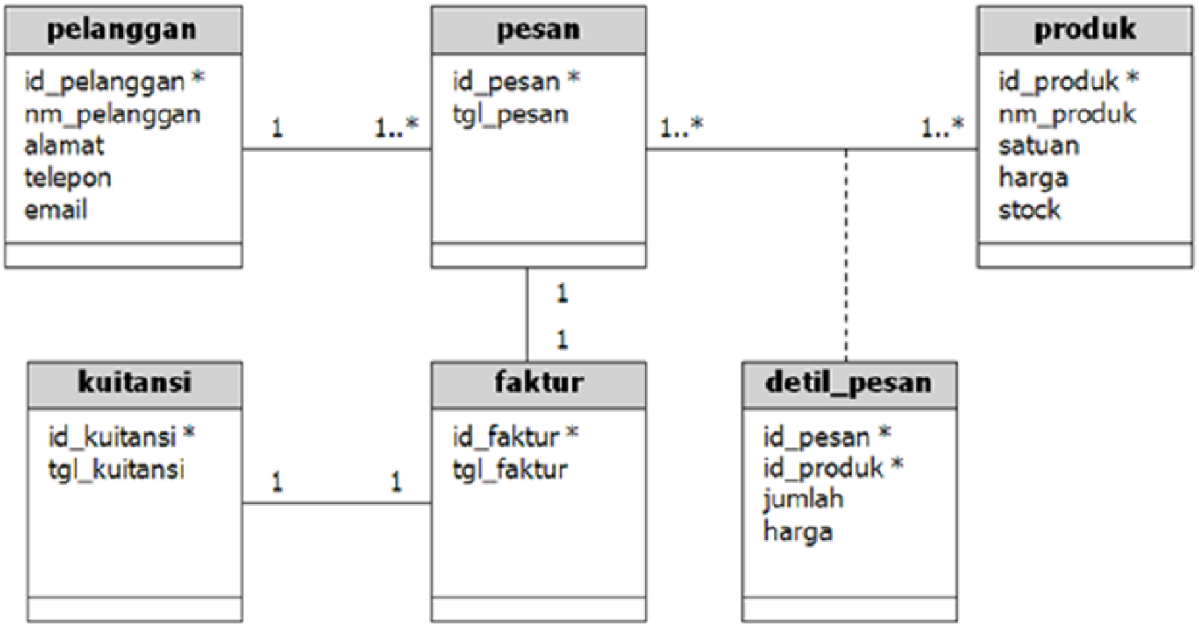
Gambaran Cardinality pada Entity Relationship Diagram diperlihatkan pada contoh sebgai berikut:

MULTIPLICITY pada pemrograman berorientasi objek digunakan untuk menyatakan tingkatan asosiasi antar class. Biasanya tingkatan asosiasi ditunjukkan pada CLASS

DIAGRAM. Tingkatan asosiasi mempunyai beberapa macam dan ditunjukkan pada tabel berikut:

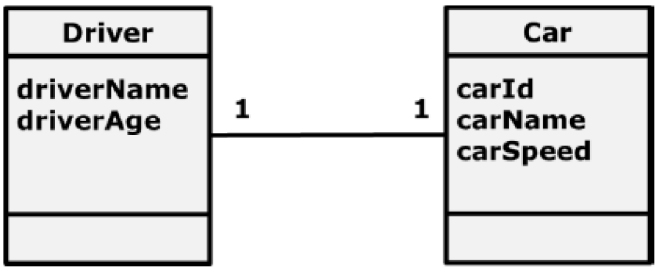
Indicator	Meaning	Example
0..1	Zero or one	
0..*	Zero or more	
0..n	Zero to n (where n>1)	0..3
1	One only	
1..*	One or more	
1..n	One to n (where n>1)	1..5
*	Many	
n	Only n (where n>1)	9
n..*	n or more, where n >1	7..*
n..m	Where n& m both >1	3..10

Adapun contoh MULTIPLICITY pada Class Diagram ditunjukkan sebagai berikut:



Penerapan Asosiasi/Relationship Pada Java

Berikut adalah contoh penerapan asosiasi pada java dengan tingkatan MULTIPLICITY adalah 1 .. 1 berikut dengan contoh class diagramnya.



Nama File : TransportCompany.java

Sintaks program:

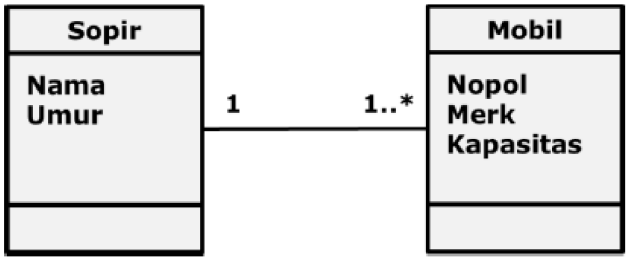
```
class CarClass{
   int carId;
   String carName;
   double carSpeed;

   CarClass(String name, double speed, int Id)
   {
    this.carName=name;
    this.carSpeed=speed;
    this.carId=Id;
   }
}
class Driver{
   String driverName;
   int driverAge;
   Driver(String name, int age){
      this.driverName=name;
      this.driverAge=age;
   }
}
class TransportCompany{
   public static void main(String args[])
   {
    CarClass obj= new CarClass("Ford", 180.15, 9988);
    Driver obj2 = new Driver("Andy", 45);
    System.out.println(obj2.driverName+" is a driver of "+obj.carName+" with car Id: "+obj.carId);
   }
}
```

Penjelasan Program :

- Pada class utama terdapat dua buah objek, masing-masing untuk class CarClass dan Driver. Statement menyatakan bahwa seorang driver/sopir mempunyai satu buah mobil berdasarkan car ID nya. Statement ini yang menyatakan asosiasi antar class adalah 1 .. 1 .

Berikut adalah contoh penerapan asosiasi pada java dengan tingkatan MULTIPLICITY adalah 1 .. * berikut dengan contoh class diagramnya.



Nama File : JadwalSupir.java

Sintaks program:

```
class Sopir{
   String Nama;
   int Umur;
   Sopir(String name, int age){
      this.Nama=name;
      this.Umur=age;
   }
}

class Mobil{
   String Nopol;
   String Jenis;
   int Kapasitas;

   Mobil(String Id, String name, int kps)
   {
    this.Jenis=name;
    this.Kapasitas=kps;
    this.Nopol=Id;
   }
}

class JadwalSopir{
   public static void main(String args[])
   {
    Mobil mobil1 = new Mobil("BN 9978 CH", "Sedan", 4);
    Mobil mobil2 = new Mobil("BN 1234 HH", "MiniBus", 7);
    Sopir andi = new Sopir("Andy Haja", 45);

    System.out.print(andi.Nama+" adalah seorang sopir ");
    System.out.print("yang bertugas membawa mobil :\n");
    System.out.print("1. No. Polisi : "+mobil1.Nopol);
    System.out.print(", Jenis : "+mobil1.Jenis);
    System.out.print(", Kapasitas : "+mobil1.Kapasitas+"\n");
    System.out.print("2. No. Polisi : "+mobil2.Nopol);
    System.out.print(", Jenis : "+mobil2.Jenis);
    System.out.print(", Kapasitas : "+mobil2.Kapasitas+"\n");

   }
}
```

Penjelasan Program :

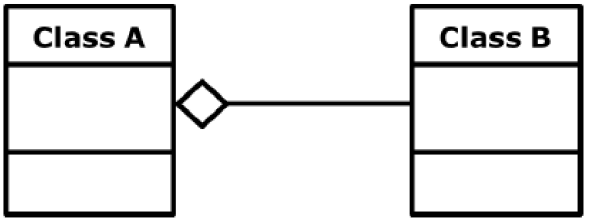
- Pada class utama terdapat tiga buah objek, masing-masing satu objek untuk class Sopir dan dua objek untuk class Mobil. Statement menyatakan bahwa seorang driver/sopir bertanggung jawab untuk dua buah kendaraan. Statement ini yang menyatakan asosiasi antar class adalah 1 .. * .

Pengenalan Agregasi

Agregasi adalah salah satu bentuk asosiasi yang lain pada pemrograman berorientasi objek. Agregasi menyatakan bahwa salah satu class merupakan bagian dari class yang lain, tetapi walaupun class tersebut merupakan bagian dari class yang lain tetapi class tersebut dapat berdiri sendiri.

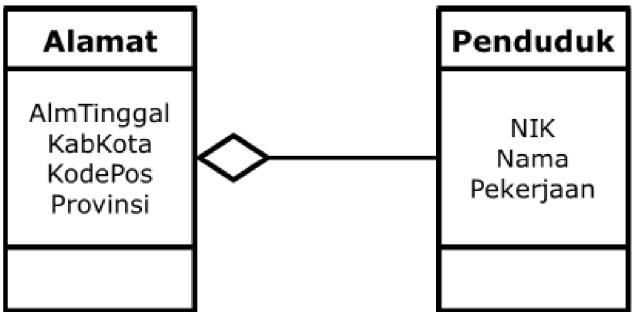
Agregasi sering disebut juga dengan relasi "part of" atau relasi "whole-part".

Pada class diagram, bentuk agregasi digambarkan sebagai berikut:



Penerapan Agregasi Pada Java

Berikut contoh penggunaan agregasi pada java. Contoh menampilkan class Alamat dan class Penduduk. Agregasi yang dimaksudkan pada contoh berikut adalah Seorang penduduk pasti memiliki alamat, sehingga penduduk merupakan bagian (part of) alamat. Walaupun penduduk bagian dari alamat, tetapi data penduduk dapat dipergunakan tanpa melibatkan data alamat. Contoh proses ini yang dimaksud dengan sebuah class merupakan bagian dari class lain tetapi dapat berdiri sendiri.



Nama File: Agregasi.java

Sintaks program:

```
class Alamat
{
   String AlmTinggal;
   String KabKota;
   String KodePos;
   String Provinsi;
   Alamat(String alm, String kab, String kode, String prov)
   {
        this.AlmTinggal=alm;
        this.KabKota =kab;
        this.KodePos = kode;
        this.Provinsi = prov;
   }
}
class Penduduk
{
   String NIK;
   String Nama;
   String Pekerjaan;
   Alamat almt;
   Penduduk(String NK, String Nm, String pkj,  Alamat al){
        this.NIK=NK;
        this.Nama=Nm;
        this.almt = al;
   }
}

public class Agregasi
{
   public static void main(String args[])
    {
       Alamat pkp = new Alamat("Jln. Jendral Sudirman","Pangkalpinang","33117","Kepulauan Bangka Belitung");
       Penduduk ani = new Penduduk("12345","Ani Rhoma","Karyawan Swasta",pkp );

       System.out.println(" DATA PENDUDUK ");
       System.out.println("---------------");
       System.out.println(" NIK      : "+ani.NIK);
       System.out.println(" Nama     : "+ani.Nama);
       System.out.println(" Alamat   : "+ani.almt.AlmTinggal);
       System.out.println(" Kota     : "+ani.almt.KabKota);
       System.out.println(" Provinsi : "+ani.almt.Provinsi);

   }
}
```

Sub Class dan Super Class

- Suatu class yang mempunyai class turunan dinamakan parent class atau base class atau super class.
- Sedangkan class turunan itu sendiri seringkali disebut subclass atau child class.
- Suatu subclass dapat mewarisi apa-apa yang dipunyai oleh parent class.

Karena suatu subclass dapat mewarisi apa Karena suatu subclass dapat mewarisi apa-apa yang dipunyai oleh parent class-nya, maka member dari suatu subclass adalah terdiri dari apa-apa yang ia punyai dan juga apa-apa yang ia warisi dari class parent-nya.
Kesimpulannya, boleh dikatakan bahwa suatu subclass adalah tidak lain hanya memperluas (extend) parent class (extend) parent class-nya.
Dengan menambahkan kata kunci Dengan menambahkan kata kunci extends setelah deklarasi nama class, kemudian diikuti dengan nama parent kemudian diikuti dengan nama parent class-nya.
Kata kunci extends tersebut memberitahu Kata kunci extends tersebut memberitahu kompiler Java bahwa kita ingin melakukan perluasan class perluasan class.

```
public class B extends A {
…
}
```

- Semua class di dalam Java adalah merupakan sub class dari class super induk yang bernama Object.
- Misalnya saja terdapat sebuah class sederhana:

```
public class A {
…
}
```

- Pada saat dikompilasi Kompiler Java Pada saat dikompilasi, Kompiler Java akan membacanya sebagai subclass dari class Object class Object.

```
public class A extends Object {
…
}
```

Kapan kita menerapkan inheritance?

- Kita baru perlu menerapkan inheritance Kita baru perlu menerapkan inheritance pada saat kita jumpai ada suatu class yang dapat diperluas dari class lain yang dapat diperluas dari class lain.
- Keuntungan dari Inheritance adalah Reusability
  - Sekali perilaku(method) didefinisikan pada super class, maka perilaku perilaku tersebut tersebut secara otomatis otomatis diwariskan ke subclass. Sehingga hanya perlu menulis method sekali dan bisa digunakan untuk semua subclass.
  - Sekali properti/field di definisikan di superclass, maka semua properti di wariskan ke semua subclass.
  - Superclass dan subclass berbagi properti
  - Subclass hanya perlu mengimplementasikan jika ada perbedaan dengan parentnya.

Misal terdapat class Pegawai

```
public class Pegawai {
      public String nama;
      public double gaji;
}
```

Misal terdapat class Manager

```
public class Manajer {{
      public String nama;
      public double gaji;
      public String departemen;
}
```

- Dari 2 buah class diatas, kita lihat class Manajer mempunyai data member yang identik sama dengan class Pegawai, hanya saja ada tambahan data member departemen.
- Sebenarnya yang terjadi disana adalah class Manajer merupakan perluasan dari class Pegawai dengan tambahan data member departemen tambahan data member departemen.
- Disini perlu memakai konsep inheritance, sehingga class Manajer dapat kita tuliskan seperti berikut

```
public class Manajer extends Pegawai {
      public String departemen;
}
```

Daftar Pustaka

[1] Deitel P.J., Deitel H.M., “Java: How to Program”, Prentice Hall, 2004

[2] Holmes B. J., Joyce D.T., “Object-Oriented Programming With Java, Second Edition”, JONES AND BARTLETT PUBLISHERS, 2001

[3] Keogh J., “JAVA DEMYSTIFIED”, McGraw-Hill/Osborne, 2004

[4] Wu C.T., “AN INTRODUCTION TO OBJECT-ORIENTED PROGRAMMING WITH JAVA™, FIFTH EDITION”, McGraw-Hill, 2010

[5] Wu C.T., “A COMPREHENSIVE INTRODUCTION TO OBJECT-ORIENTED PROGRAMMING WITH JAVA”, McGraw-Hill, 2008

[6] Poo D., Kiong D., Ashok S., “Object-Oriented Programming and Java Second edition”, Springer, 2008

[7] Sintes T., “Sams Teach Yourself Object Oriented Programming in 21 Days”, Sam Publishing, 2002