**Jobsheet 04 ‑ Relasi Kelas**

# I. Kompetensi

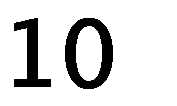
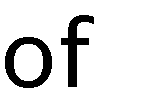
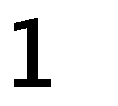
Setelah menempuh pokok bahasan ini, mahasiswa mampu:

1. Memahami konsep relasi kelas;
2. Mengimplementasikan relasi *has‑a* dalam program.

# II. Pendahuluan

Pada kasus yang lebih kompleks, dalam suatu sistem akan ditemukan lebih dari satu *class* yang saling memiliki keterkaitan antara *class* satu dengan yang lain. Pada percobaan‑percobaan sebelumnya, mayoritas kasus yang sudah dikerjakan hanya fokus pada satu *class* saja. Pada jobsheet ini akan dilakukan percobaan yang melibatkan beberapa *class* yang saling berelasi.

Misalnya terdapat *class* Laptop yang memiliki atribut berupa merk dan prosesor. Jika diperhatikan lebih rinci, maka atribut prosesor sendiri didalamnya memiliki data berupa merk, nilai *cache* memori, dan nilai *clock*‑nya. Artinya, ada *class* lain yang namanya *Processor* yang memiliki atribut merk, *cache* dan *clock*, dan atribut prosesor yang ada di dalam *class* Laptop itu merupakan objek dari *class* Proceessor tersebut. Sehingga terlihat antara *class* Laptop dan class Processor memiliki relasi (*has‑a*).



+

Laptop

()

+

Laptop

(

merk

:

String

,

proc

:

Processor

)

+

setMerk

(

merk

:

String

)

+

getMerk

():

String

+

setProc

(

proc

:

Processor

)

+

getProc

():

Processor

+

info

()

**Processor**

-

merk

:

String

-

cache

:

double

**Laptop**

-

merk

:

String

-

proc

:

Processor

+

Processor

()

+

Processor

(

merk

:

String

,

cache

:

double

)

+

setMerk

(

merk

:

String

)

+

getMerk

():

String

+

setCache

(

cache

:

double

)

+

getCache

():

double

+

info

()

Jenis relasi *has‑a* ini yang akan dicontohkan pada percobaan di jobsheet ini. Apabila dilihat lebih rinci lagi, relasi tersebut disebut juga dengan agregasi (*has-a*). Relasi antar kelas yang lain adalah dependensi (*uses-a*) dan *inheritance* (*is-a*). Diperlukan inisiatif mandiri dari tiap mahasiswa untuk memperdalam jenis relasi lain terutama yang tidak dibahas pada mata kuliah ini.

**III. Praktikum**

# Percobaan 1

1. Perhatikan diagram *class* berikut:

+

Laptop

()

+

Laptop

(

merk

:

String

,

proc

:

Processor

)

+

setMerk

(

merk

:

String

)

+

getMerk

():

String

+

setProc

(

proc

:

Processor

)

+

getProc

():

Processor

+

info

()

**Processor**

-

merk

:

String

-

cache

:

double

**Laptop**

-

merk

:

String

-

proc

:

Processor

+

Processor

()

+

Processor

(

merk

:

String

,

cache

:

double

)

+

setMerk

(

merk

:

String

)

+

getMerk

():

String

+

setCache

(

cache

:

double

)

+

getCache

():

double

+

info

()

1. Buka *project* baru di *Netbeans* dan buat *package* dengan format berikut:

<identifier>.relasiclass.percobaan1 (ganti <identifier> dengan identitas anda atau nama domain), Contoh: ac.id.polinema, jti.polinema, dan sebagainya).

**Catatan:** Penamaan *package* dengan tambahan identifier untuk menghindari adanya kemungkinan penamaan *class* yang bentrok.

1. Buatlah class Processor dalam *package* tersebut.

**public class Processor {**

**}**

1. Tambahkan atribut merk dan *cache* pada class Processor dengan akses modifier private .

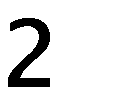
**private String** merk;

**private double** cache;

1. Buatlah *constructor default* untuk class Processor.

1. Buatlah *constructor* untuk class Processor dengan parameter merk dan *cache*.

1. Implementasikan **setter** dan **getter** untuk class Processor.



1. Implementasikan *method* info() seperti berikut:

**public void** **info**() {

System.out.printf("Merk Processor = %s\n", merk);

System.out.printf("Cache Memory = %.2f\n", cache);

}

public class Processor {

    private String **merk**;

    private double **cache**;

    public Processor(String merk, double cache) {

        this.**merk** = merk;

        this.**cache** = cache;

    }

    public Processor() {

    }

    public String getMerk() {

        return **merk**;

    }

    public void setMerk(String merk) {

        this.**merk** = merk;

    }

    public double getCache() {

        return **cache**;

    }

    public void setCache(double cache) {

        this.**cache** = cache;

    }

    public void info() {

        System.**out**.printf("Merk Processor = %s\n", **merk**);

        System.**out**.printf("Cache Memory = %.2f\n", **cache**);

    }

}

1. Kemudian buatlah class Laptop di dalam package yang telah anda buat.

1. Tambahkan atribut merk dengan tipe String dan proc dengan tipe Object Processor

**private** String merk;

**private** Processor proc;

1. Buatlah *constructor* default untuk *class* Laptop .

1. Buatlah *constructor* untuk *class* Laptop dengan parameter merk dan proc .

1. Selanjutnya implementasikan method info() pada *class* Laptop sebagai berikut

**public void** **info**() {

System.out.println("Merk Laptop = " + merk); proc.info();

}

public class Laptop {

    private String **merk**;

    private Processor **proc**;

    public Laptop (String merk, Processor proc) {

        this.**merk** = merk;

        this.**proc** = proc;

    }

    public Laptop() {

    }

    public String getMerk() {

        return **merk**;

    }

    public void setMerk(String merk) {

        this.**merk** = merk;

    }

    public Processor getProc() {

        return **proc**;

    }

    public void setProc(Processor proc) {

        this.**proc** = proc;

    }

    public void info() {

        System.**out**.println("Merk Laptop = " + **merk**);

**proc**.info();

    }

}

1. Pada *package* yang sama, buatlah class MainPercobaan1 yang berisi method main().

1. Deklarasikan Object Processor dengan nama p kemudian instansiasi dengan informasi atribut Intel i5 untuk nilai merk serta 3 untuk nilai *cache* .

Processor p = **new** Processor("Intel i5", 3);

1. Kemudian deklarasikan serta instansiasi Objek Laptop dengan nama L dengan informasi atribut Thinkpad dan Objek Processor yang telah dibuat.

1. Panggil method info() dari Objek L.

L.info();

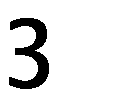
1. Tambahkan baris kode berikut

Processor p1 = **new** Processor(); p1.setMerk("Intel i5"); p1.setCache(4);

Laptop L1 = **new** Laptop();

L1.setMerk("Thinkpad");

L1.setProc(p1); L1.info();



1. *Compile* kemudian *run* *class* MainPercobaan1, akan didapatkan hasil seperti berikut:

public class Main {

    public static void main(String[] args) {

        Processor p = new Processor("Intel i5", 3);

        Laptop l = new Laptop("Thinkpad", p);

        l.info();

        Processor p1 = new Processor();

        Laptop l1 = new Laptop();

        p1.setMerk("Intel i5");

        p1.setCache(4);

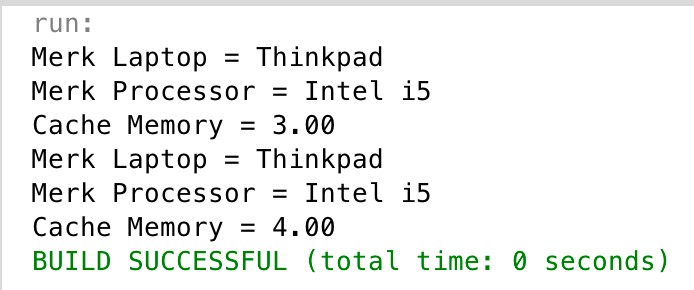
        l1.setMerk("Thinkpad");

        l1.setProc(p1);

        l1.info();

    }

}



# Pertanyaan

Berdasarkan percobaan 1, jawablah pertanyaan‑pertanyaan yang terkait:

1. Di dalam *class* Processor dan *class* Laptop , terdapat method *setter* dan *getter* untuk masing‑masing atributnya. Apakah gunanya *method* *setter* dan *getter* tersebut ?

* Untuk setter berguna untuk memberi nilai pada suatu variable yang telah dibuat dalam sebuah class. Sedangkan untuk getter dibuat untuk mengambil nilai dari suatu variable yang telah dibuat dalam sebuah class.

1. Di dalam *class* Processor dan *class* Laptop, masing‑masing terdapat konstruktor default dan konstruktor berparameter. Bagaimanakah beda penggunaan dari kedua jenis konstruktor tersebut ?

* Untuk konstruktor default, saat penginisiasian objek tidak perlu untuk mengisi nilai pada parameter. Sedangkan untuk konstruktor diperlukan untuk mengisi parameter pada saat pendeklarasian objek saat dibuat.

1. Perhatikan *class* Laptop, di antara 2 atribut yang dimiliki (*merk* dan *proc*), atribut manakah yang bertipe *object* ?

* Atribut yang bertipe objek adalah proc.

1. Perhatikan *class* Laptop, pada baris manakah yang menunjukan bahwa *class* Laptop memiliki relasi dengan *class* Processor ?

* Pada saat penginisiasian berikut pada class Laptop :

    private Processor **proc**;

1. Perhatikan pada *class* Laptop , Apakah guna dari sintaks proc.info() ?

* Untuk memanggil function yang ada dalam class Processor.

1. Pada *class* MainPercobaan1, terdapat baris kode:

Laptop l = new Laptop("Thinkpad", p);.

Apakah p tersebut ?

Dan apakah yang terjadi jika baris kode tersebut diubah menjadi:

Laptop l = new Laptop("Thinkpad", new Processor("Intel i5",

3));

Bagaimanakah hasil program saat dijalankan, apakah ada perubahan ?

* Baris program tersebut adalah pendeklarasian objek baru dari class laptop sekaligus pengisian nilai untuk objek yang dibuat.

# Percobaan 2

Perhatikan diagram *class* berikut yang menggambarkan sistem rental mobil. Pelanggan bisa menyewa mobil sekaligus sopir. Biaya sopir dan biaya sewa mobil dihitung per hari.

+

Sopir

()

+

setNama

(

nama

:

String

)

+

getNama

():

String

+

setBiaya

(

biaya

:

int

)

+

getBiaya

():

int

+

hitungBiayaSopir

(

hari

:

int

):

int

**Pelanggan**

-

nama

:

String

-

mobil

:

Mobil

-

sopir

:

Sopir

-

hari

:

int

**Sopir**

-

nama

:

String

-

biaya

:

int

+

Pelanggan

()

+

setNama

(

nama

:

String

)

+

getNama

():

String

+

setMobil

(

mobil

:

Mobil

)

+

getMobil

():

Mobil

+

setSopir

(

sopir

:

Sopir

)

+

getSopir

():

Sopir

+

setHari

(

hari

:

int

)

+

getHari

():

int

+

hitungBiayaTotal

():

int

+

Mobil

()

+

setNama

(

nama

:

String

)

+

getNama

():

String

+

setBiaya

(

biaya

:

int

)

+

getBiaya

():

int

+

hitungBiayaMobil

(

hari

:

int

):

int

**Mobil**

-

nama

:

String

-

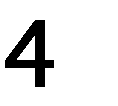
biaya

:

int

1. Tambahkan *package* <identifier>.relasiclass.percobaan2.

1. Buatlah *class* Mobil di dalam *package* tersebut.



1. Tambahkan atribut *merk* tipe String dan biaya tipe int dengan akses *modifier*

private.

1. Tambahkan *constructor* *default* serta setter dan getter.

1. Implementasikan method hitungBiayaMobil

**public int** hitungBiayaMobil(**int** hari) { **return** biaya \* hari;

}

public class Mobil {

    private String **merk**;

    private int **biaya**;

    public Mobil(String merk, int biaya) {

        this.**merk** = merk;

        this.**biaya** = biaya;

    }

    public Mobil() {

    }

    public String getMerk() {

        return **merk**;

    }

    public void setMerk(String merk) {

        this.**merk** = merk;

    }

    public int getBiaya() {

        return **biaya**;

    }

    public int setBiaya(int biaya) {

        return this.**biaya** = biaya;

    }

    public int hitungBiayaMobil(int hari) {

        return **biaya** \* hari;

    }

}

1. Tambahkan *class* Sopir dengan atribut nama tipe String dan biaya tipe *int* dengan akses *modifier* private berikut dengan constructor default.

1. Implementasikan method hitungBiayaSopir

**public int** hitungBiayaSopir(**int** hari) { **return** biaya \* hari;

}

public class Sopir {

    private String **nama**;

    private int **biaya**;

    public Sopir(String nama, int biaya) {

        this.**nama** = nama;

        this.**biaya** = biaya;

    }

    public Sopir() {

    }

    public int hitungBiayaSopir(int hari) {

        return **biaya** \* hari;

    }

    public String getNama() {

        return **nama**;

    }

    public void setNama(String nama) {

        this.**nama** = nama;

    }

    public int getBiaya() {

        return **biaya**;

    }

    public void setBiaya(int biaya) {

        this.**biaya** = biaya;

    }

}

1. Tambahkan *class* Pelanggan dengan *constructor* *default*.

1. Tambahkan atribut‑atribut dengan akses modifier *private* berikut:

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

1. Implementasikan *setter* dan *getter*.

1. Tambahkan method hitungBiayaTotal

**public int** **hitungBiayaTotal**() { **return** mobil.hitungBiayaMobil(hari) + sopir.hitungBiayaSopir(hari);

}

public class Pelanggan {

    private String **nama**;

    private Sopir **sopir**;

    private Mobil **mobil**;

    private int **hari**;

    public Pelanggan(String nama, Sopir sopir, Mobil mobil, int hari) {

        this.**nama** = nama;

        this.**sopir** = sopir;

        this.**mobil** = mobil;

        this.**hari** = hari;

    }

    public Pelanggan() {

    }

    public void setHari(int hari) {

        this.**hari** = hari;

    }

    public String getNama() {

        return **nama**;

    }

    public void setNama(String nama) {

        this.**nama** = nama;

    }

    public Sopir getSopir() {

        return **sopir**;

    }

    public void setSopir(Sopir sopir) {

        this.**sopir** = sopir;

    }

    public Mobil getMobil() {

        return **mobil**;

    }

    public void setMobil(Mobil mobil) {

        this.**mobil** = mobil;

    }

    public int getHari() {

        return **hari**;

    }

    public int hitungBiayaTotal() {

        return **mobil**.hitungBiayaMobil(**hari**) + **sopir**.hitungBiayaSopir(**hari**);

    }

}

1. Buatlah *class* MainPercobaan2 yang berisi method main(). Tambahkan baris kode berikut:

Mobil m = **new** Mobil();

m.setMerk("Avanza");

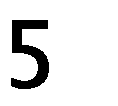
m.setBiaya(350000); Sopir s = **new** Sopir(); s.setNama("John Doe");

s.setBiaya(200000);

Pelanggan p = **new** Pelanggan();

p.setNama("Jane Doe");

p.setMobil(m);

p.setSopir(s);

p.setHari(2);

System.out.println("Biaya Total = " +

p.hitungBiayaTotal());

1. *Compile* dan jalankan class MainPercobaan2, dan perhatikan hasilnya!

* Tambahan class penyewaan :

public class penyewaan {

    private int **hari**;

    private int **biayaMobil**;

    private int **biayaSopir**;

    private Mobil **mobil**;

    private Sopir **sopir**;

    public penyewaan(int hari, int biayaMobil, int biayaSopir) {

        this.**hari** = hari;

        this.**biayaMobil** = biayaMobil;

        this.**biayaSopir** = biayaSopir;

    }

    public void setHari(int hari) {

        this.**hari** = hari;

    }

    public int hitungBiayaTotal() {

        return **mobil**.hitungBiayaMobil(**hari**) + **sopir**.hitungBiayaSopir(**hari**);

    }

}

public class Main {

    public static void main(String[] args) {

        Mobil m = new Mobil();

        Pelanggan p = new Pelanggan();

        Sopir s = new Sopir();

        m.setMerk("Avanza");

        m.setBiaya(350000);

        s.setNama("John Doe");

        s.setBiaya(200000);

        p.setNama("Jane Doe");

        p.setMobil(m);

        p.setSopir(s);

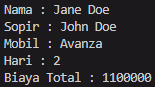
        p.setHari(2);

        penyewaan ps = new penyewaan(p.getHari());

        System.**out**.println(p.getMobil().getMerk());

    }

}



# Pertanyaan

1. Perhatikan *class* Pelanggan. Pada baris program manakah yang menunjukan bahwa *class* Pelanggan memiliki relasi dengan *class* Mobil dan *class* Sopir ?

* Pada baris program berikut yang ada pada class Pelanggan :

    private Sopir **sopir**;

    private Mobil **mobil**;

1. Perhatikan *method* hitungBiayaSopir pada class Sopir, serta method hitungBiayaMobil pada class Mobil. Mengapa menurut Anda *method* tersebut harus memiliki argument hari ?

* Karena biaya penyewaan dihitung perhari, maka diperlukan variable hari untuk menghitung total biaya yang nantinya akan dikalikan dengan hari.

1. Perhatikan kode dari *class* Pelanggan. Untuk apakah perintah mobil.hitungBiayaMobil(hari) dan sopir.hitungBiayaSopir(hari) ?

* Untuk menghitung total biaya penyewaan yang dilakukan oleh pelanggan dari penyewaan sopir beserta mobil yang disewanya.

1. Perhatikan *class* MainPercobaan2. Untuk apakah sintaks p.setMobil(m) dan p.setSopir(s) ?

* Untuk memberi nilai atribut objek yang dibuat pada class pelanggan dari mobil maupun sopir yang ada pada objek yang telah dibuat.

1. Perhatikan class MainPercobaan2. Untuk apakah proses p.hitungBiayaTotal() tersebut ?

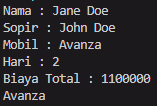
* Untuk menghitung total biaya yang telah dikeluarkan oleh pelanggan setelah melakukan penyewaan sopir maupun mobil.

1. Perhatikan class MainPercobaan2, coba tambahkan pada baris terakhir dari *method*

*main* dan amati perubahan saat di‑*run*!

System.out.println(p.getMobil().getMerk());

Jadi untuk apakah sintaks p.getMobil().getMerk() yang ada di dalam *method main* tersebut?

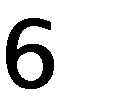


* Untuk mengambil nilai atribut merk mobil yang telah dibuat dalam class pelanggan.

# Percobaan 3

Pada percobaan‑percobaan sebelumnya, relasi dalam *class* dinyatakan dalam *one‑to‑one*. Tetapi ada kalanya relasi *class* melibatkan lebih dari satu. Hal ini disebut dengan *multiplicity*. Untuk relasi lebih rinci mengenai *multiplicity*, dapat dilihat pada tabel berikut.

|  |  |
| --- | --- |
| **Multiplicity** | **Keterangan** |
| 0..1 | 0 atau 1 instance |
| 1 | Tepat 1 instance |
| 0..\* | 0 atau lebih instance |
| 1..\* | setidaknya 1 instance |
| n | Tepat n instance (n diganti dengan sebuah angka) |
| m..n | Setidaknya m instance, tetapi tidak lebih dari n |

1. Sebuah Kereta Api dioperasikan oleh Masinis serta seorang Asisten Masinis. Baik Masinis maupun Asisten Masinis keduanya merupakan Pegawai PT. Kereta Api Indonesia. Dari ilustrasi cerita tersebut, dapat digambarkan dalam diagram kelas sebagai berikut:

+

KeretaApi

(

nama

:

String

,

kelas

:

String

,

masinis

:

Pegawai

)

+

KeretaApi

(

nama

:

String

,

kelas

:

String

,

masinis

:

Pegawai

,

asisten

:

Pegawai

)

+

setNama

(

nama

:

String

)

+

getNama

():

String

+

setKelas

(

kelas

:

String

)

+

getKelas

():

String

+

setMasinis

(

masinis

:

Pegawai

)

+

getMasinis

():

Pegawai

+

setAsisten

(

asisten

:

Pegawai

)

+

getAsisten

():

Pegawai

+

info

():

String

**Pegawai**

-

nip

:

String

-

nama

:

String

**KeretaApi**

-

nama

:

String

-

kelas

:

String

-

masinis

:

Pegawai

-

asisten

:

Pegawai

+

Pegawai

(

nip

;

String

,

nama

:

String

)

+

setNip

(

nip

:

String

)

+

getNip

():

String

+

setNama

(

nama

:

String

)

+

getNama

():

String

+

info

():

String

1. Perhatikan dan pahami diagram kelas tersebut, kemudian bukalah IDE anda!

1. Buatlah *package* <identifier>.relasiclass.percobaan3, kemudian tambahkan *class* Pegawai.

1. Tambahkan atribut‑atribut ke dalam class Pegawai

**private** String nip;

**private** String nama;

1. Buatlah *constructor* untuk *class* Pegawai dengan parameter nip dan nama.

1. Tambahkan *setter* dan *getter* untuk masing‑masing atribut.
2. Implementasikan *method* info() dengan menuliskan baris kode berikut:

**public** String **info**() { String info = ""; info += "Nip: " + **this**.nip + "\n"; info += "Nama: " + **this**.nama + "\n"; **return** info;

}

public class Pegawai {

    private String **nama**, **nip**;

    public Pegawai(String nip, String nama) {

        this.**nama** = nama;

        this.**nip** = nip;

    }

    public String getNama() {

        return **nama**;

    }

    public void setNama(String nama) {

        this.**nama** = nama;

    }

    public String getNip() {

        return **nip**;

    }

    public void setNip(String nip) {

        this.**nip** = nip;

    }

    public String info() {

        String info = "";

info += "Nip: " + this.**nip** + "\n";

        info += "Nama: " + this.**nama** + "\n";

        return info;

    }

}

1. Buatlah *class* KeretaApi berdasarkan diagram *class*.

1. Tambahkan atribut‑atribut pada *class* KeretaApi berupa nama, kelas, masinis, dan asisten.

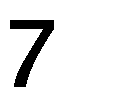
**private** String nama; **private** String kelas; **private** Pegawai masinis;

**private** Pegawai asisten;

1. Tambahkan *constructor* 3 parameter (nama, kelas, masinis) serta 4 parameter (nama, kelas, masinis, asisten).

1. Tambahkan *setter* dan *getter* untuk atribut‑atribut yang ada pada *class* KeretaApi .

1. Kemudian implementasikan *method* info()

**public** String **info**() { String info = "";

info += "Nama: " + **this**.nama + "\n"; info += "Kelas: " + **this**.kelas + "\n"; info += "Masinis: " + **this**.masinis.info() + "\n"; info += "Asisten: " + **this**.asisten.info() + "\n"; **return** info;

}

public class KeretaApi {

    private String **nama**, **kelas**;

    private Pegawai **masinis**, **asisten**;

    public KeretaApi(String nama, String kelas, Pegawai masinis) {

        this.**nama** = nama;

        this.**kelas** = kelas;

        this.**masinis** = masinis;

    }

    public KeretaApi(String nama, String kelas, Pegawai masinis, Pegawai asisten) {

        this.**nama** = nama;

        this.**kelas** = kelas;

        this.**masinis** = masinis;

        this.**asisten** = asisten;

    }

    public String getNama() {

        return **nama**;

    }

    public void setNama(String nama) {

        this.**nama** = nama;

    }

    public String getKelas() {

        return **kelas**;

    }

    public void setKelas(String kelas) {

        this.**kelas** = kelas;

    }

    public Pegawai getMasinis() {

        return **masinis**;

    }

    public void setMasinis(Pegawai masinis) {

        this.**masinis** = masinis;

    }

    public Pegawai getAsisten() {

        return **asisten**;

    }

    public void setAsisten(Pegawai asisten) {

        this.**asisten** = asisten;

    }

    public String info() {

        String info = "";

        info += "Nama: " + this.**nama** + "\n";

        info += "Kelas: " + this.**kelas** + "\n\n";

        info += "Masinis: \n" + this.**masinis**.info() + "\n"; info += "Asisten: \n" + this.**asisten**.info();

        return info;

    }

}

1. Buatlah sebuah *class* MainPercobaan3 dalam *package* yang sama.

1. Tambahkan *method* main() kemudian tuliskan baris kode berikut.

Pegawai masinis = **new** Pegawai("1234", "Spongebob

Squarepants");

Pegawai asisten = **new** Pegawai("4567", "Patrick Star"); KeretaApi keretaApi = **new** KeretaApi("Gaya Baru", "Bisnis", masinis, asisten);

System.out.println(keretaApi.info());

public class Main {

    public static void main(String[] args) {

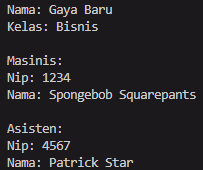
        Pegawai masinis = new Pegawai("1234", "Spongebob Squarepants");

        Pegawai asisten = new Pegawai("4567", "Patrick Star"); KeretaApi keretaApi = new KeretaApi("Gaya Baru", "Bisnis", masinis, asisten);

        System.**out**.println(keretaApi.info());

    }

}



# Pertanyaan

1. Di dalam *method* info() pada *class* KeretaApi, baris this.masinis.info() dan this.asisten.info() digunakan untuk apa ?

* Untuk memanggil fungsi yang ada di class pegawai yang mana fungsi tersebut digunakan untuk memunculkan info dari pegawai.

1. Buatlah *main* program baru dengan nama *class* MainPertanyaan pada *package* yang sama. Tambahkan kode berikut pada *method* main() !

Pegawai masinis = **new** Pegawai("1234", "Spongebob

Squarepants");

KeretaApi keretaApi = **new** KeretaApi("Gaya Baru", "Bisnis", masinis);

System.out.println(keretaApi.info());

1. Apa hasil output dari *main* program tersebut ? Mengapa hal tersebut dapat terjadi ?

* Saat program dijalankan akan terjadi error karena terdapat atribut yang tidak diisi.

1. Perbaiki *class* KeretaApi sehingga program dapat berjalan !

public String info() {

        String info = "";

        info += "Nama: " + this.**nama** + "\n";

        info += "Kelas: " + this.**kelas** + "\n\n";

        info += "Masinis: \n" + this.**masinis**.info() + "\n";

        if(this.**asisten** != null) {

            info += "Asisten: \n" + this.**asisten**.info();

        }

        return info;

    }

# Percobaan 4

**Penumpang Kursi Gerbong**

-ktp: String -nomor: String -kode: String

-nama: String -penumpang: Penumpang -arrayKursi: Kursi[]

+Penumpang(ktp: String, nama: String) +Kursi(nomor: String) +Gerbong(kode: String, jumlah: int)

+setKtp(ktp: String) +setNomor(nomor: String) -initKursi()

+getKtp(): String +getNomor(): String +setKode(kode: String)

+setNama(nama: String) +setPenumpang(penumpang: Penumpang) +getKode(): String

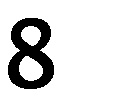
+getNama(): String +getPenumpang(): Penumpang +setPenumpang(penumpang: String, nomor: int)

+info(): String +info(): String +getArrayKursi(): Kursi[]

+info(): String

1. Perhatikan dan pahami diagram *class* tersebut.

1. Buatlah masing‑masing *class* Penumpang, Kursi dan Gerbong sesuai rancangan tersebut pada *package* <identifier>.relasiclass.percobaan4.



1. Tambahkan *method* info() pada *class* Penumpang

**public** String **info**() { String info = ""; info += "Ktp: " + ktp + "\n"; info += "Nama: " + nama + "\n"; **return** info;

}

1. Tambahkan *method* info() pada *class* Kursi

**public** String **info**() { String info = "";

info += "Nomor: " + nomor + "\n"; **if** (**this**.penumpang != **null**) {

info += "Penumpang: " + penumpang.info() + "\n";

} **return** info;

}

1. Pada *class* Gerbong buatlah *method* initKursi() dengan akses private.

**private** **void** **initKursi**() { **for** (**int** i = 0; i < arrayKursi.length; i++) { **this**.arrayKursi[i] = **new** Kursi(String.valueOf(i + 1)); }

}

1. Panggil *method* initKursi() dalam *constructor* Gerbong sehingga baris kode menjadi berikut:

**public** **Gerbong**(String kode, **int** jumlah) { **this**.kode = kode;

**this**.arrayKursi = **new** Kursi[jumlah]; **this**.initKursi();

}

1. Tambahkan *method* info() pada *class* Gerbong

**public** String **info**() { String info = "";

info += "Kode: " + kode + "\n"; **for** (Kursi kursi : arrayKursi) { info += kursi.info();

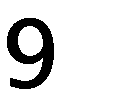
} **return** info;

}

1. Implementasikan *method* untuk memasukkan penumpang sesuai dengan nomor kursi.

**public void** **setPenumpang**(Penumpang penumpang, **int** nomor) { **this**.arrayKursi[nomor - 1].setPenumpang(penumpang);

}

1. Buatlah *class* MainPercobaan4 yang berisi *method* main(). Kemudian tambahkan baris berikut!

Penumpang p = **new** Penumpang("12345", "Mr. Krab");

Gerbong gerbong = **new** Gerbong("A", 10); gerbong.setPenumpang(p, 1);

System.out.println(gerbong.info());

# Pertanyaan

1. Pada *main* program dalam *class* MainPercobaan4, berapakah jumlah kursi dalam Gerbong A ?

* Terdapat total 10 kursi yang ada.

1. Perhatikan potongan kode pada *method* info() dalam *class* Kursi. Apa maksud kode tersebut ?

... **if** (**this**.penumpang != **null**) { info += "Penumpang: " + penumpang.info() + "\n";

} ...

* Untuk menampilkan data dari para penumpang. Dan jika terdapat kursi kosong tanpa penumpang, maka akan langsung menuju ke data kursi selanjutnya.

1. Mengapa pada *method* setPenumpang() dalam *class* Gerbong, nilai nomor dikurangi dengan angka 1 ?

* Karena hal ini merujuk ke index array yang mana indexnya dimulai dari 0.

1. Instansiasi objek baru budi dengan tipe Penumpang, kemudian masukkan objek baru tersebut pada gerbong dengan gerbong.setPenumpang(budi, 1). Apakah yang terjadi ?

* Yang terjadi adalah data sebelumnya(mr. krab) akan tertimpa dengan data yang baru.

1. Modifikasi program sehingga tidak diperkenankan untuk menduduki kursi yang sudah ada penumpang lain !

public void setPenumpang(Penumpang penumpang, int nomor) {

        if(this.**kursi**[nomor - 1].getPenumpang() == null) {

            this.**kursi**[nomor - 1].setPenumpang(penumpang);

        } else {

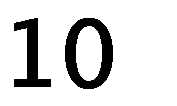
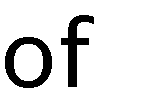
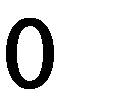
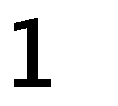
            System.**out**.println("Kursi ini telah diisi. Anda akan dipindahkan ke kursi kosong");

            this.**kursi**[nomor].setPenumpang(penumpang);

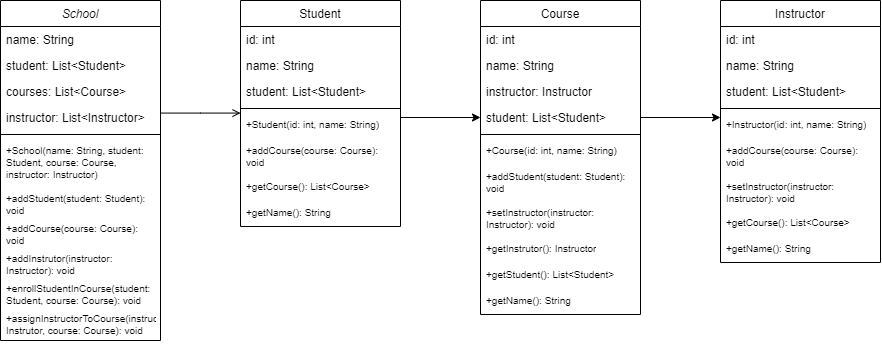
        }

    }

# IV. Tugas



Buatlah sebuah studi kasus, rancang dengan *class* diagram, kemudian implementasikan ke dalam program! Studi kasus harus mewakili relasi *class* dari percobaan‑percobaan yang telah dilakukan pada materi ini, setidaknya melibatkan minimal 4 *class* (*class* yang berisi *main* tidak dihitung).



package tugas;

import java.util.ArrayList;

import java.util.*List*;

public class School {

    private String **name**;

    private *List*<Student> **students**;

    private *List*<Course> **courses**;

    private *List*<Instructor> **instructors**;

    public School(String name) {

        this.**name** = name;

        this.**students** = new ArrayList<>();

        this.**courses** = new ArrayList<>();

        this.**instructors** = new ArrayList<>();

    }

    public void addStudent(Student student) {

**students**.add(student);

    }

    public void addCourse(Course course) {

**courses**.add(course);

    }

    public void addInstructor(Instructor instructor) {

**instructors**.add(instructor);

    }

    public void enrollStudentInCourse(Student student, Course course) {

        student.addCourse(course);

        course.addStudent(student);

    }

    public void assignInstructorToCourse(Instructor instructor, Course course) {

        instructor.addCourse(course);

        course.setInstructor(instructor);

    }

}

package tugas;

import java.util.ArrayList;

import java.util.*List*;

public class Student {

    private int **id**;

    private String **name**;

    private *List*<Course> **courses**;

    public Student(int id, String name) {

        this.**id** = id;

        this.**name** = name;

        this.**courses** = new ArrayList<>();

    }

    public void addCourse(Course course) {

**courses**.add(course);

    }

    public *List*<Course> getCourses() {

        return **courses**;

    }

    public String getName () {

        return this.**name**;

    }

}

package tugas;

import java.util.ArrayList;

import java.util.*List*;

public class Course {

    private int **id**;

    private String **name**;

    private Instructor **instructor**;

    private *List*<Student> **students**;

    public Course(int id, String name) {

        this.**id** = id;

        this.**name** = name;

        this.**students** = new ArrayList<>();

    }

    public void addStudent(Student student) {

**students**.add(student);

    }

    public void setInstructor(Instructor instructor) {

        this.**instructor** = instructor;

    }

    public Instructor getInstructor() {

        return **instructor**;

    }

    public *List*<Student> getStudents() {

        return **students**;

    }

}

package tugas;

import java.util.ArrayList;

import java.util.*List*;

public class Instructor {

    private int **id**;

    private String **name**;

    private *List*<Course> **courses**;

    public Instructor(int id, String name) {

        this.**id** = id;

        this.**name** = name;

        this.**courses** = new ArrayList<>();

    }

    public void addCourse(Course course) {

**courses**.add(course);

    }

    public *List*<Course> getCourses() {

        return **courses**;

    }

    public String getName() {

        return this.**name**;

    }

}

package tugas;

public class Main {

    public static void main(String[] args) {

        School university = new School("School of Java");

        Student student1 = new Student(1, "John Doe");

        Student student2 = new Student(2, "Jane Doe");

        Course course1 = new Course(1, "Java Programming");

        Course course2 = new Course(2, "Data Structures");

        Instructor instructor1 = new Instructor(1, "Dr. Java");

        Instructor instructor2 = new Instructor(2, "Dr. Jawa");

        university.addStudent(student1);

        university.addStudent(student2);

        university.addCourse(course1);

        university.addCourse(course2);

        university.addInstructor(instructor1);

        university.addInstructor(instructor2);

        university.enrollStudentInCourse(student1, course1);

        university.enrollStudentInCourse(student2, course2);

        university.assignInstructorToCourse(instructor1, course1);

        university.assignInstructorToCourse(instructor2, course2);

        System.**out**.println("Students enrolled in course 1:");

        for (Student student : course1.getStudents()) {

            System.**out**.println(student.getName());

        }

        System.**out**.println("Instructor of course 1:");

        System.**out**.println(course1.getInstructor().getName());

        System.**out**.println();

        System.**out**.println("Students enrolled in course 2:");

        for (Student student : course2.getStudents()) {

            System.**out**.println(student.getName());

        }

        System.**out**.println("Instructor of course 2:");

        System.**out**.println(course2.getInstructor().getName());

    }

}

