LAPORAN PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK

JOBSHEET 4: RELASI KELAS

oleh :
Halim Teguh Saputro
2E
2141762122

PROGRAM STUDI D-IV SISTEM INFORMASI BISNIS
JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG
Jl. Soekarno Hatta No .9, Jatimulyo, Kec. Lowokwaru, Kota Malang,
JawaTimur 65141

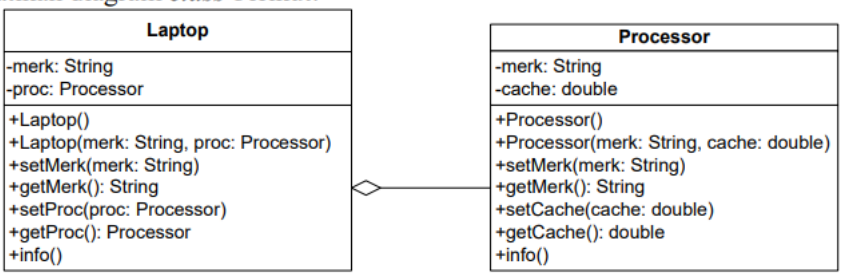
KOMPETENSI

Setelah melakukan percobaan pada modul ini, mahasiswa memahami konsep:

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

PRAKTIKUM 1

a. Perhatikan diagram *class* berikut:



b. 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.

c. Buatlah class `Processor` dalam *package* tersebut.

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

```
src > Processor.java > Processor > setCache(double)
1   public class Processor {
2       private String merk;
3       private double cache;
```

e. Buatlah *constructor default* untuk `class Processor`.

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

```
4
5       Processor() {
6       };
7
8       Processor(String merk, double cache) {
9           this.merk = merk;
10          this.cache = cache;
11      }
```

g. Implementasikan **setter** dan **getter** untuk `class Processor`.

```
13      public String getMerk() {
14          return merk;
15      }
16
17      public double getCache() {
18          return cache;
19      }
20
21      public void setMerk(String merk) {
22          this.merk = merk;
23      }
24
25      public void setCache(double cache) {
26          this.cache = cache;
27      }
```

h. Implementasikan *method* `info()` seperti berikut:

```
29      public void info() {
30          System.out.printf(format: "Merek Processor = %s\n", merk);
31          System.out.printf(format: "Cache Memory = %.2f\n", cache);
32      }
```

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

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

```
src > Laptop.java > Laptop
1   public class Laptop {
2       private String merk;
3       private Processor proc;
```

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

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

```
5       Laptop() {
6       };
7
8       Laptop(String merk, Processor proc) {
9           this.merk = merk;
10          this.proc = proc;
11      }
```

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

```
29      public void info() {
30          System.out.println("Merek Laptop = " + merk);
31          proc.info();
32      }
```

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

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

```
1  public class MainPercobaan1 {
       Run | Debug
2      public static void main(String[] args) {
3          Processor p = new Processor(merk: "Intel i5", cache: 3);
```

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

```
4          Laptop l = new Laptop(merk: "Thinkpad", p);
```

q. Panggil method `info()` dari Objek `L`.

```
5
6          l.info();
7
```

r. Tambahkan baris kode berikut

```
8          Processor p1 = new Processor();
9          p1.setMerk(merk: "Intel i5");
10         p1.setCache(cache: 4);
11
12         Laptop l1 = new Laptop();
13         l1.setMerk(merk: "ThinkPad");
14         l1.setProc(p1);
15
16         l1.info();
```

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

```
erbasis Objek\Praktikum\05. P
Merek Laptop = Thinkpad
Merek Processor = Intel i5
Cache Memory = 3.00
Merek Laptop = ThinkPad
Merek Processor = Intel i5
Cache Memory = 4.00
```

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 ?

JAWAB:

Setter dan getter digunakan untuk bisa mengakses atribut private yang ada dalam class-class tersebut. Kemudian bisa di ubah data atau nilai dari atribut tersebut. Setter digunakan untuk mengubah atau menambahkan data pada atribut. Sedangkan, getter digunakan untuk mengambil nilai dari atribut tersebut.

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

JAWAB:

Untuk constructor default tidak memiliki parameter sehingga penggunaan objeknya tidak perlu diberikan parameter saat instansiasi objek. untuk mengisikan nilainya bisa menggunakan fungsi setter & getter.

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

JAWAB:

Attribute yang bertipe object yaitu atribut proc, karena tipe datanya yaitu Processor yang mengarah ke class Processor.

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

JAWAB:

```
28
29      public void info() {
30          System.out.println("Merek Laptop = " + merk);
31          proc.info();
32      }
33
```

Pada baris 31 di dalam method info() class Laptop karena baris tersebut akan memanggil method info() yang ada di class Processor.

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

JAWAB:

Sintaks proc.info() digunakan untuk menjalankan method info() pada class Processor yaitu method untuk menampilkan data merek processor dan cache processor.

6. 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 ?

JAWAB:

Parameter p tersebut mengarah ke variable objek Processor yang telah di inisialiasasi sebelumnya

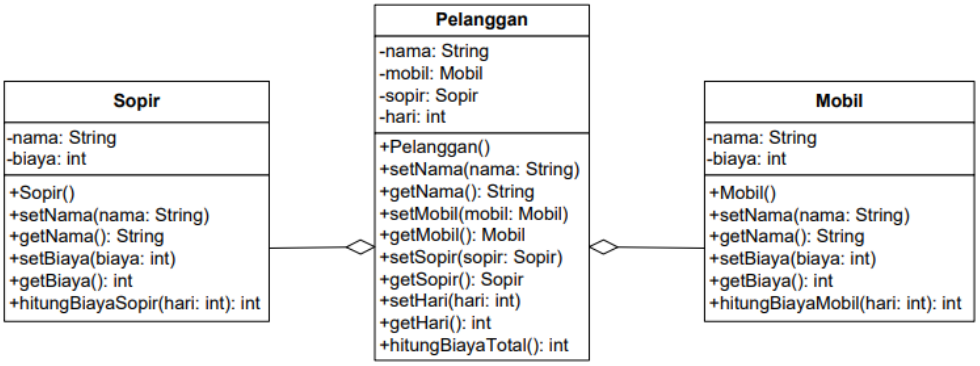
```
src > MainPercobaan1.java > MainPercobaan1 > main(String[])
1   public class MainPercobaan1 {
        Run | Debug
2       public static void main(String[] args) {
3           Processor p = new Processor(merk: "Intel i5", cache: 3);
4           Laptop l = new Laptop(merk: "Thinkpad", new Processor(merk: "Intel i5", cache: 3));
5
```

```
PROBLEMS 1   OUTPUT   DEBUG CONSOLE   TERMINAL        Run: MainPercobaan1

heet_5\bin' 'MainPercobaan1'
Merek Laptop = Thinkpad
Merek Processor = Intel i5
Cache Memory = 3.00
Merek Laptop = ThinkPad
Merek Processor = Intel i5
Cache Memory = 4.00
```

Hasil yang didapatkan tetap sama, berarti inisialisasi objek langsung pada parameter itu bisa dilakukan.

PRAKTIKUM 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.



a. Tambahkan *package* `<identifier>.relasiclass.percobaan2`.

b. Buatlah *class* `Mobil` di dalam *package* tersebut.

c. Tambahkan atribut *merk* tipe `String` dan biaya tipe `int` dengan akses *modifier* `private`.

```
src > Mobil.java > Mobil > setMerk(String)
1   public class Mobil {
2       private String merk;
3       private int biaya;
4
5       Mobil() {
6
7       }
8
9       Mobil(String merk, int biaya) {
10          this.merk = merk;
11          this.biaya = biaya;
12      }
```

d. Tambahkan *constructor default* serta `setter` dan `getter`.

```
14      public String getMerk() {
15          return merk;
16      }
17
18      public int getBiaya() {
19          return biaya;
20      }
21
22      public void setMerk(String merk) {
23          this.merk = merk;
24      }
25
26      public void setBiaya(int biaya) {
27          this.biaya = biaya;
28      }
```

e. Implementasikan method `hitungBiayaMobil`

```
30      public int hitungBiayaMobil(int hari) {
31          return biaya * hari;
32      }
```

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

```
src > Sopir.java > Sopir > Sopir()
1  public class Sopir {
2      private String nama;
3      private int biaya;
4
5      Sopir() {
6
7      }
```

g. Implementasikan method `hitungBiayaSopir`

```
9      public String getNama() {
10         return nama;
11     }
12
13     public int getBiaya() {
14         return biaya;
15     }
16
17     public void setNama(String nama) {
18         this.nama = nama;
19     }
20
21     public void setBiaya(int biaya) {
22         this.biaya = biaya;
23     }
24
25     public int hitungBiayaSopir(int hari) {
26         return biaya * hari;
27     }
```

h. Tambahkan *class* `Pelanggan` dengan *constructor default*.

i. Tambahkan atribut-atribut dengan akses modifier *private* berikut:

```
src > Pelanggan.java > Pelanggan >
1  public class Pelanggan {
2      private String nama;
3      private Mobil mobil;
4      private Sopir sopir;
5      private int hari;
6
7      Pelanggan() {
8
9      }
```

j. Implementasikan *setter* dan *getter*.

```
11     public Mobil getMobil() {
12         return mobil;
13     }
14
15     public void setMobil(Mobil mobil) {
16         this.mobil = mobil;
17     }
```

```
19      public Sopir getSopir() {
20          return sopir;
21      }
22
23      public void setSopir(Sopir sopir) {
24          this.sopir = sopir;
25      }
26
27      public String getNama() {
28          return nama;
29      }
30
31      public void setNama(String nama) {
32          this.nama = nama;
33      }
34
35      public int getHari() {
36          return hari;
37      }
38
39      public void setHari(int hari) {
40          this.hari = hari;
41      }
```

k. Tambahkan method `hitungBiayaTotal`

```
43      public int hitungBiayaTotal() {
44          return mobil.hitungBiayaMobil(hari) +
45                  sopir.hitungBiayaSopir(hari);
46      }
```

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

```
src > MainPercobaan2.java > MainPercobaan2 > main(String[])
1   public class MainPercobaan2 {
        Run | Debug
2       public static void main(String[] args) {
3           Mobil m = new Mobil();
4           m.setMerk(merk: "Avanza");
5           m.setBiaya(biaya: 350000);
6
7           Sopir s = new Sopir();
8           s.setNama(nama: "Halim");
9           s.setBiaya(biaya: 200000);
10
```

```
11          Pelanggan p = new Pelanggan();
12          p.setNama(nama: "Teguh");
13          p.setMobil(m);
14          p.setSopir(s);
15          p.setHari(hari: 2);
16
17          System.out.println("Biaya Total = " +
18              p.hitungBiayaTotal());
19      }
20  }
```

m. *Compile* dan jalankan class `MainPercobaan2`, dan perhatikan hasilnya!

```
05. Relasi Kelas\jobshee
Biaya Total = 1100000
PS C:\Users\Halim\Downlo
```

Pertanyaan

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

JAWAB:

```
1  public class Pelanggan {
2      private String nama;
3      private Mobil mobil;
4      private Sopir sopir;
5      private int hari;
```

Baris program yang memiliki hubungan Dengan class Mobil dan Sopir terdapat saat deklarasi atribut. Ada atribut yang menggunakan tipe data Mobil yang mengarah ke class Mobil dan atribut yang menggunakan tipe data Sopir yang mengarah ke class Sopir.

```
11      public Mobil getMobil() {
12          return mobil;
13      }
14
15      public void setMobil(Mobil mobil) {
16          this.mobil = mobil;
17      }
18
19      public Sopir getSopir() {
20          return sopir;
21      }
22
23      public void setSopir(Sopir sopir) {
24          this.sopir = sopir;
25      }
```

Selanjutnya terdapat di setter dan getter agar bisa mengisikan atribut yang mengarah ke class Mobil dan Sopir sebelumnya.

```
43      public int hitungBiayaTotal() {
44          return mobil.hitungBiayaMobil(hari) +
45                  sopir.hitungBiayaSopir(hari);
46      }
```

Kemudian yang terakhir, terdapat di method hitungBiayaTotal() yaitu untuk mereturn method hitung biaya mobil dan hitung biaya sopir.

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

JAWAB:

Argument hari tersebut diperlukan untuk menghitung biaya total.

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

JAWAB:

Kode tersebut digunakan untuk memanggil method hitung biaya mobil dan hitung biaya sopir yang menghitung biaya * hari. Sehingga bisa digunakan untuk menhitung biaya total.

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

JAWAB:

Sintaks tersebut berguna untuk mengambil objek yang telah di inisialisasi sebelumnya

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

JAWAB:

Sintaks tersebut berguna untuk memanggil method hitungBiayaTotal() yang akan menghitung seluruh biaya dari mobil dan biaya dari sopir. Kemudian akan ditampilkan di output.

6. 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?

JAWAB:

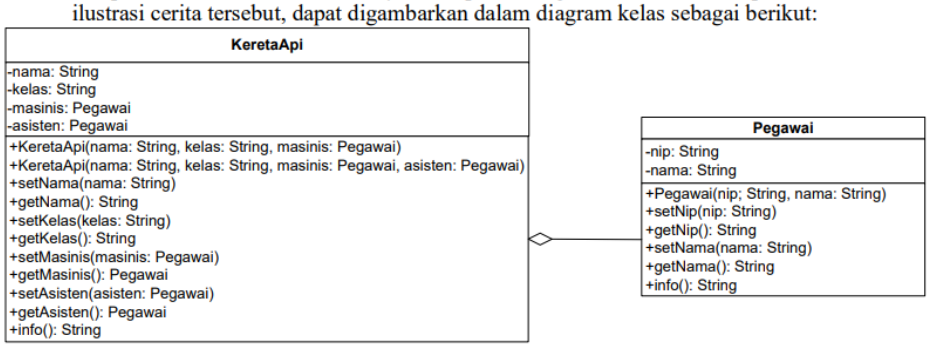
Sintaks tersebut berfungsi untuk mengarah ke objek mobil dan mengambil nama Merek dari mobil tersebut.

PRAKTIKUM 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

a. 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:



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

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

d. Tambahkan atribut-atribut ke dalam class `Pegawai`

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

```java
public class Pegawai {
    private String nip;
    private String nama;

    Pegawai(String nip, String nama) {
        this.nip = nip;
        this.nama = nama;
    }
```

f. Tambahkan *setter* dan *getter* untuk masing-masing atribut.

```java
    public void setNip(String nip) {
        this.nip = nip;
    }

    public String getNip() {
        return nip;
    }

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

    public String getNama() {
        return nama;
    }
```

g. Implementasikan *method* `info()` dengan menuliskan baris kode berikut:

```java
    public String info() {
        String info = "";
        info += "NIP: " + this.nip + "\n";
        info += "Nama: " + this.nama + "\n";
        return info;
    }
```

h. Buatlah *class* `KeretaApi` berdasarkan diagram *class*.

i. Tambahkan atribut-atribut pada *class* `KeretaApi` berupa `nama`, `kelas`, `masinis`, dan `asisten`.

src > KeretaApi.java > KeretaApi > info()

```java
public class KeretaApi {
    private String nama;
    private String kelas;
    private Pegawai masinis;
    private Pegawai asisten;
```

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

```
7       KeretaApi(String nama, String kelas, Pegawai masinis) {
8           this.nama = nama;
9           this.kelas = kelas;
10          this.masinis = masinis;
11      }
12
13      KeretaApi(String nama, String kelas, Pegawai masinis, Pegawai asisten) {
14          this.nama = nama;
15          this.kelas = kelas;
16          this.masinis = masinis;
17          this.asisten = asisten;
18      }
```

k. Tambahkan *setter* dan *getter* untuk atribut-atribut yang ada pada *class* `KeretaApi` .

```
20      public String getNama() {
21          return nama;
22      }
23
24      public void setNama(String nama) {
25          this.nama = nama;
26      }
27
28      public String getKelas() {
29          return kelas;
30      }
31
32      public void setKelas(String kelas) {
33          this.kelas = kelas;
34      }
35
36      public Pegawai getMasinis() {
37          return masinis;
38      }
39
40      public void setMasinis(Pegawai masinis) {
41          this.masinis = masinis;
42      }
```

```
44      public Pegawai getAsisten() {
45          return asisten;
46      }
47
48      public void setAsisten(Pegawai asisten) {
49          this.asisten = asisten;
50      }
```

l. Kemudian implementasikan *method* `info()`

```java
    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;
    }
```

m. Buatlah sebuah *class* `MainPercobaan3` dalam *package* yang sama.

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

```java
src > MainPercobaan3.java > MainPercobaan3 > main(String[])
public class MainPercobaan3 {
    Run | Debug
    public static void main(String[] args) {
        Pegawai masinis = new Pegawai(
                nip: "1234", nama: "Spongebob Squarepants");
        Pegawai asisten = new Pegawai(
                nip: "4567", nama: "Patrick Star");
        KeretaApi keretaApi = new KeretaApi(
                nama: "Gaya Baru", kelas: "Bisnis", masinis, asisten);

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

Output

```
jobsheet_5\bin' 'MainPercobaan
Nama: Gaya Baru
Kelas: Bisnis
Masinis: NIP: 1234
Nama: Spongebob Squarepants

Asisten: NIP: 4567
Nama: Patrick Star
```

Pertanyaan

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

JAWAB:

Sintaks tersebut berfungsi untuk mengambil nilai atribut asisten melalui method info yang mengarah ke class pegawai.

2. 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());
```

JAWAB:

```
src > MainPertanyaan.java > ...
1   public class MainPertanyaan {
        Run | Debug
2       public static void main(String[] args) {
3           Pegawai masinis = new Pegawai(nip: "1234", nama: "Spongebob Squarpants");
4           KeretaApi keretaApi = new KeretaApi(nama: "Gaya Baru", kelas: "Bisnis", masinis);
5
6           System.out.println(keretaApi.info());
7       }
8   }
```

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

JAWAB:

Ketika program mainPertanyaan dijalankan akan terjadi Error karena untuk menjalankan keretaApi.info() memerlukan nilai dari asisten juga, namun karena nilai dari asisten tidak di isi. Maka terjadi Error NullPointerException.

4. Perbaiki *class* `KeretaApi` sehingga program dapat berjalan !

JAWAB:

Menurut saya ada 3 penyelesaian. yang pertama, pada class KeretaApi bisa menghilangkan sintaks

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

yang kedua, pada class MainPertanyaan menambahkan sintaks inisialisasi untuk asisten dan menambah argument asisten pada inisialisasi keretaApi

```
src > MainPertanyaan.java > MainPertanyaan
1   public class MainPertanyaan {
        Run | Debug
2       public static void main(String[] args) {
3           Pegawai masinis = new Pegawai(nip: "1234", nama: "Spongebob Squarpants");
4           Pegawai asisten = new Pegawai(nip: "1234", nama: "Halim");
5           KeretaApi keretaApi = new KeretaApi(nama: "Gaya Baru", kelas: "Bisnis", masinis, asisten);
6
7           System.out.println(keretaApi.info());
8       }
```

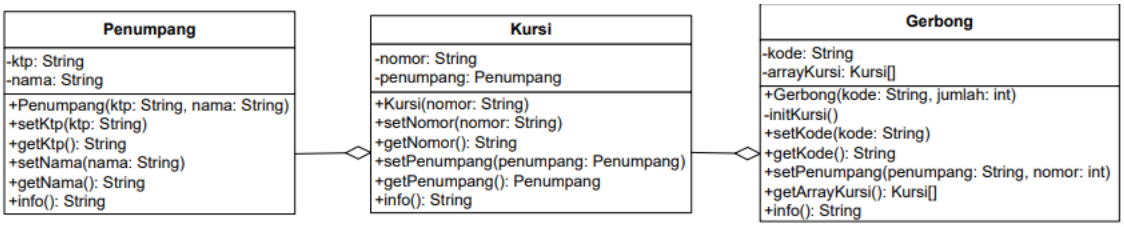
```
Nama: Gaya Baru
Kelas: Bisnis
Masinis: NIP: 1234
Nama: Spongebob Squarpants

Asisten: NIP: 1234
Nama: Halim
```

Yang ketiga, bisa menambahkan sintaks if untuk menyeleksi bila program memiliki argument asisten atau tidak.

```
52      public String info() {
53          String info = "";
54          info += "Nama: " + this.nama + "\n";
55              info += "Kelas: " + this.kelas + "\n";
56              info += "Masinis: " + this.masinis.info() + "\n";
57          if(this.asisten != null){
58              info += "Asisten: " + this.asisten.info() + "\n";
59          }
60          return info;
61      }
```

```
Nama: Gaya Baru
Kelas: Bisnis
Masinis: NIP: 1234
Nama: Spongebob Squarpants
```

PRAKTIKUM 4



a. Perhatikan dan pahami diagram *class* tersebut.

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

1. Penumpang

```
src > Penumpang.java > Penumpang > info()
1   public class Penumpang {
2       private String ktp;
3       private String nama;
```

```
5       Penumpang() {
6       }
7
8       Penumpang(String ktp, String nama) {
9           this.ktp = ktp;
10          this.nama = nama;
11      }
12
13      public void setKtp(String ktp) {
14          this.ktp = ktp;
15      }
16
17      public String getKtp() {
18          return ktp;
19      }
20
21      public void setNama(String nama) {
22          this.nama = nama;
23      }
```

s

```
25      public String getNama() {
26          return nama;
27      }
```

2. Kursi

```
1   public class Kursi {
2       private String nomor;
3       private Penumpang penumpang;
4
5       Kursi() {
6
7       }
8
9       Kursi(String nomor) {
10          this.nomor = nomor;
11      }
12
13      public void setPenumpang(Penumpang penumpang) {
14          this.penumpang = penumpang;
15      }
```

```
21      public void setKursi(String nomor) {
22          this.nomor = nomor;
23      }
```

```
25      public String getNomor() {
26          return nomor;
27      }
```

3. Gerbong

src > Gerbong.java > Gerbong > Gerbong(String, int)

```java
public class Gerbong {
    private String kode;
    private Kursi[] arrayKursi;

    Gerbong() {
    }

    Gerbong(String kode, int jumlah) {
        this.kode = kode;
        this.arrayKursi = new Kursi[jumlah];
        this.initKursi();
    }

    public String getKode() {
        return kode;
    }

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

    public Kursi[] getArrKursi() {
        return arrayKursi;
    }
```

```java
    public void setArrKursi(Kursi[] arrKursi) {
        this.arrayKursi = arrKursi;
    }
```

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

    public String info() {
        String info = "";
        info += "Kode: " + kode + "\n";
        for (Kursi kursi : arrayKursi) {
            info += kursi.info();
        }
        return info;
    }

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

c. Tambahkan *method* `info()` pada *class* `Penumpang`

```
29      public String info() {
30          String info = "";
31          info += "KTP: " + ktp + "\n";
32          info += "Nama: " + nama + "\n";
33          return info;
34      }
35  }
```

d. Tambahkan *method* `info()` pada *class* `Kursi`

```
25      public String getNomor() {
26          return nomor;
27      }
28
29      public String info() {
30          String info = "";
31          info += "Nomor: " + nomor + "\n";
32          if (this.penumpang != null) {
33              info += "Penumpang: " + penumpang.info() + "\n";
34          }
35          return info;
36      }
```

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

```
30      private void initKursi() {
31          for (int i = 0; i < arrayKursi.length; i++) {
32              this.arrayKursi[i] = new Kursi(String.valueOf(i + 1));
33          }
34      }
```

f. Panggil *method* `initKursi()` dalam *constructor* `Gerbong` sehingga baris kode menjad berikut:

```
8       Gerbong(String kode, int jumlah) {
9           this.kode = kode;
10          this.arrayKursi = new Kursi[jumlah];
11          this.initKursi();
12      }
```

g. Tambahkan *method* `info()` pada *class* `Gerbong`

```
36      public String info() {
37          String info = "";
38          info += "Kode: " + kode + "\n";
39          for (Kursi kursi : arrayKursi) {
40              info += kursi.info();
41          }
42          return info;
43      }
```

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

```
45      public void setPenumpang(Penumpang penumpang, int nomor) {
46          this.arrayKursi[nomor - 1].setPenumpang(penumpang);
47      }
```

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

```
src > MainPercobaan4.java > MainPercobaan4
1   public class MainPercobaan4 {
        Run | Debug
2       public static void main(String[] args) {
3           Penumpang p = new Penumpang(ktp: "1234", nama: "Mr. Krab");
4           Gerbong gerbong = new Gerbong(kode: "A", jumlah: 10);
5
6           gerbong.setPenumpang(p, nomor: 1);
7
8           System.out.println(gerbong.info());
9       }
10  }
```

Output

```
Kode: A
Nomor: 1
Penumpang: KTP: 1234
Nama: Mr. Krab

Nomor: 2
Nomor: 3
Nomor: 4
Nomor: 5
Nomor: 6
Nomor: 7
Nomor: 8
Nomor: 9
Nomor: 10
```

Pertanyaan

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

JAWAB:

Ada 10 kursi dan yang terisi oleh penumpang adalah 1 kursi.

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

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

JAWAB:

Sintaks tersebut digunakan untuk mengecek apakah pada kursi terdapat penumpang, jika ada maka akan ditampilkan info dari penumpang tersebut.

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

JAWAB:

Nilai nomor dikurangi angka 1 karena saat memasukkan nomor pasti dimulai dengan angka 1, sedangkan indeks pada array akan dimulai dari 0. Agar kursi terisi di kursi pertama maka nomor akan di kurangi satu agar bisa masuk ke indek ke-0 (indek pertama). Bila tidak di kurangi dengan 1 maka penumpang yang mengisi nomor 1 akan masuk ke nomor kursi 2.

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

JAWAB:

```
1   public class MainPercobaan4 {
        Run | Debug
2       public static void main(String[] args) {
3           Penumpang p1 = new Penumpang(ktp: "1234", nama: "Mr. Krab");
4           Penumpang budi = new Penumpang(ktp: "4567", nama: "budi");
5
6           Gerbong gerbong = new Gerbong(kode: "A", jumlah: 10);
7
8           gerbong.setPenumpang(p1, nomor: 1);
9           gerbong.setPenumpang(budi, nomor: 1);
10
11          System.out.println(gerbong.info());
12
13      }
14  }
```

Saya membuat objek penumpang baru dengan nama budi, kemudian memasukkan nya kedalam gerbong.

```
Kode: A
Nomor: 1
Penumpang: KTP: 4567
Nama: budi
Nomor: 2
Nomor: 3
Nomor: 4
Nomor: 5
Nomor: 6
Nomor: 7
Nomor: 8
Nomor: 9
Nomor: 10
```

Hasil dari program tersebut adalah penumpang kedua yaitu budi akan menindih penumpang pertama, sehingga data penumpang pertama akan hilang tergantikan dengan penumpang kedua

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

JAWAB:

```
45    public void setPenumpang(Penumpang penumpang, int nomor) {
46        for (int i = nomor - 1; i < arrayKursi.length; i++) {
47            if (this.arrayKursi[i].getPenumpang() == null) {
48                this.arrayKursi[i].setPenumpang(penumpang);
49                break;
50            }
51            nomor++;
52        }
53    }
```

Saya mengubah method setPenumpang menjadi seperti pada gambar, menggunakan perulangan, sehingga bila beberapa kali menginputkan dengan nomor yang sama maka akan langsung mengisi kursi yang kosong.

OUTPUT:

```
Kode: A
Nomor: 1
Penumpang: KTP: 1234
Nama: Mr. Krab

Nomor: 2
Penumpang: KTP: 4567
Nama: budi

Nomor: 3
Nomor: 4
Nomor: 5
Nomor: 6
Nomor: 7
Nomor: 8
Nomor: 9
Nomor: 10
```