

TUGAS PBO MINGGU 4

Memenuhi Tugas Mata Kuliah Praktikum Pemrograman Berbasis Objek

Dosen : Odhitya Desta Triswidrananta, S.Pd, M.Pd



Nama : Raden Dimas Erlangga

Kelas : D-III Manajemen Informatika 2E

Nim : 2031710121

PROGRAM STUDI D-III MANAJEMEN INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI

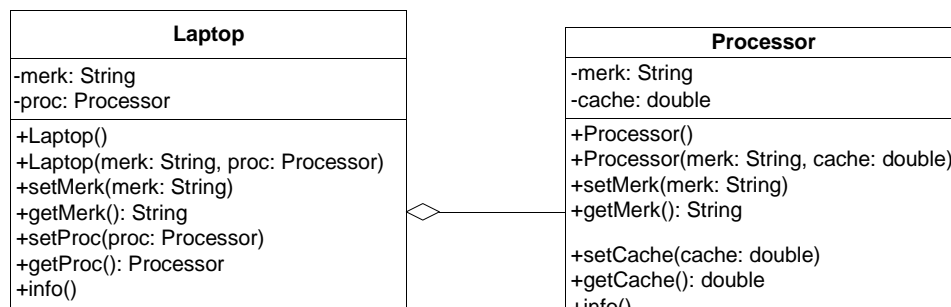
POLITEKNIK NEGERI MALANG

2021

JOB SHEET 4 - Relasi Kelas

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* Processor tersebut. Sehingga terlihat antara *class* Laptop dan *class* Processor memiliki relasi (*has-a*).

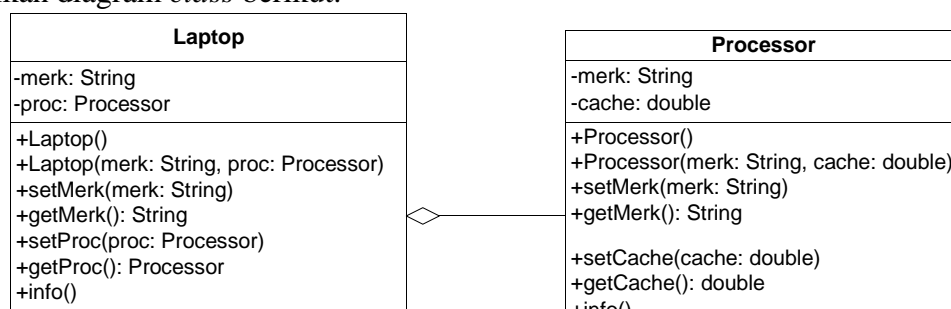


Jenis relasi *has-a* ini yang akan dicontohkan pada percobaan di jobsheet ini. Apabila dilihat lebih rinci lagi, relasi tersebut disebut juga dengan agregasi (*has-a*). Relasi antar kelas yang lain adalah dependensi (*uses-a*) dan *inheritance* (*is-a*). Diperlukan inisiatif mandiri dari tiap mahasiswa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

PRAKTIKUM

Percobaan 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`.
- e. Buatlah *constructor default* untuk class Processor.
- f. Buatlah *constructor* untuk class Processor dengan parameter merk dan *cache*.
- g. Implementasikan **setter** dan **getter** untuk class Processor.
- h. Implementasikan *method* `info()` seperti berikut:

```
public void info() {
    System.out.printf("Merk Processor = %s\n",
    merk); System.out.printf("Cache Memory =
    %.2f\n", cache);
}
```

Struktur Class Processor :

```
1
2 package raden.relasticlass.percobaan1;
3
4 /**
5  * @Raden Dimas Erlangga
6  */
7 public class Processor {
8     private String merk;
9     private double cache;
10
11     //constructor default
12     public Processor() {
13     }
14
15     //constructor berparameter
16     public Processor(String merk, double cache) {
17         this.merk = merk;
18         this.cache = cache;
19     }
20
21     //setter dan getter class processor
22     public void setMerk(String merk) {
23         this.merk = merk;
24     }
25     public String getMerk() {
26         return merk;
27     }
28
29     public void setCache(double cache) {
30         this.cache = cache;
31     }
32     public double getCache() {
33         return cache;
34     }
35
36     // implementasi method info
37     public void info() {
38         System.out.printf("Merek Processor = %s/n", merk);
39         System.out.printf("Cache Memory = %.2f\n", cache);
40     }
41 }
```

- 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
- k. Buatlah *constructor* default untuk *class* Laptop .
- l. Buatlah *constructor* untuk *class* Laptop dengan parameter merk dan proc .
- m. Selanjutnya implementasikan method `info()` pada *class* Laptop sebagai berikut

```
public void info() {
    System.out.println("Merk Laptop = " +
        merk); proc.info();
}
```

Struktur Class Laptop :

```
1
2 package raden.relasticlass.percobaan1;
3 /**
4  * @Raden Dimas Erlangga
5  */
6 public class Laptop {
7     private String merk;
8     private Processor proc;
9
10    //constructor default untuk Laptop
11    public Laptop() {
12    }
13
14    //constructor dengan parameter
15    public Laptop(String merk, Processor proc) {
16        this.merk = merk;
17        this.proc = proc;
18    }
19
20    //method info
21    public void info(){
22        System.out.println("Merek Laptop = "+merk);
23        proc.info();
24    }
25
26    //setter
27    public void setMerk(String merk) {
28        this.merk = merk;
29    }
30
31    public void setProc(Processor proc) {
32        this.proc = proc;
33    }
34 }
```

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*.

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

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

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

```
L.info();
```

r. Tambahkan baris kode berikut

```
Processor p1 = new  
Processor();  
p1.setMerk("Intel i5");  
p1.setCache(4);  
Laptop L1 = new Laptop();  
L1.setMerk("Think  
pad");  
L1.setProc(p1);  
L1.info();
```

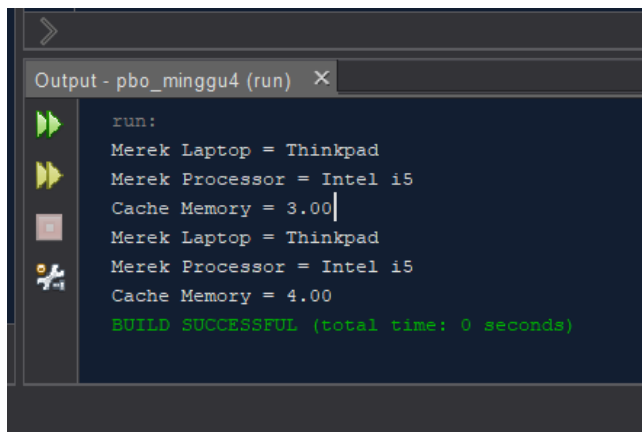
s. *Compile* kemudian *run* class `MainPercobaan1`, akan didapatkan hasil seperti berikut:

```
run:  
Merk Laptop = Thinkpad  
Merk Processor = Intel i5  
Cache Memory = 3.00  
Merk Laptop = Thinkpad  
Merk Processor = Intel i5  
Cache Memory = 4.00  
BUILD SUCCESSFUL (total time: 0 seconds)
```

Struktur class main :

```
1 package raden.relasiclass.percobaan1;  
2  
3 /**  
4  * @Raden Dimas Erlangga  
5  */  
6 public class MainPercobaan1 {  
7     public static void main(String[] args) {  
8         Processor p = new Processor("Intel i5", 3);  
9         Laptop L = new Laptop("Thinkpad", p);  
10        L.info();  
11  
12        Processor p1 = new Processor();  
13        p1.setMerk("Intel i5");  
14        p1.setCache(4);  
15        Laptop L1 = new Laptop();  
16        L1.setMerk("Thinkpad");  
17        L1.setProc(p1);  
18        L1.info();  
19    }  
20 }  
21
```

Hasil Running :



```
Output - pbo_minggu4 (run) X
run:
Merek Laptop = Thinkpad
Merek Processor = Intel i5
Cache Memory = 3.00
Merek Laptop = Thinkpad
Merek Processor = Intel i5
Cache Memory = 4.00
BUILD SUCCESSFUL (total time: 0 seconds)
```

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?

Jawaban : pada class processor dan class laptop, terdapat method getter dan setter untuk mendapatkan nilai dan mengubah nilai dari atribut class processor dan laptop, karena atribut class processor dan laptop bersifat private, maka di-implementasikan getter dan setter.

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

Jawaban : pada class Processor dan Laptop terdapat dua konstruktor yaitu default dan berparameter, Konstruktor default adalah konstruktor yang tidak menggunakan argument pada konstruktor nya, konstruktor ini digunakan untuk membuat sebuah Object dari sebuah class, dimana variable akan terisi sesuai default value dari masing-masing variable.

Konstruktor berparameter adalah Konstruktor yang mengambil satu atau lebih argument, Biasanya kita akan membutuhkan suatu Konstruktor yang memiliki satu atau lebih parameter didalamnya. dengan tujuan untuk membuat suatu Object, dimana object tersebut sudah mempunyai default value dari variables didalamnya.

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

Jawaban : atribut dengan tipe object pada class Laptop adalah : *proc*

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

Jawaban :

Pada kode program yang saya buat, baris yang menunjukkan bahwa *class* Laptop memiliki relasi dengan *class* processor berada pada baris ke-8

```
1
2 package raden.relasiclass.percobaan1;
3 /**
4  * @Raden Dimas Erlangga
5  */
6 public class Laptop {
7     private String merk;
8     private Processor proc;
```

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

Jawaban : `proc.info` digunakan untuk mengambil method `info` dari *class* Processor.

6. Pada *class* MainPercobaan1, terdapat baris kode: `Laptop l = new Laptop("Thinkpad", p);`. Apakah `p` tersebut ?

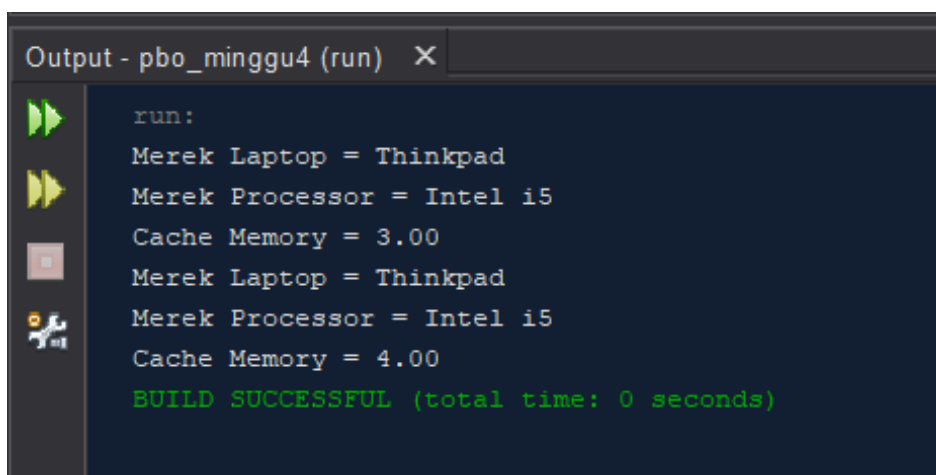
Jawaban : `p` adalah instansiasi dari *class* Processor

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 ?

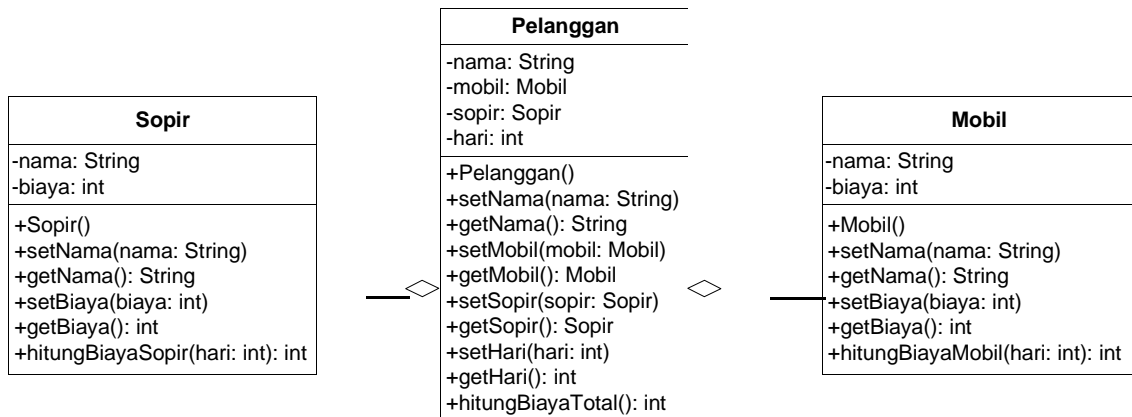
Jawaban : hasil nya sama, dikarenakan instansiasi Processor dilakukan didalam kurung parameter dari konstruktor laptop.



```
Output - pbo_minggu4 (run) X
run:
Merek Laptop = Thinkpad
Merek Processor = Intel i5
Cache Memory = 3.00
Merek Laptop = Thinkpad
Merek Processor = Intel i5
Cache Memory = 4.00
BUILD SUCCESSFUL (total time: 0 seconds)
```

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.



- Tambahkan *package* <identifier>.relasiclass.percobaan2.
- Buatlah *class* Mobil di dalam *package* tersebut.
- Tambahkan atribut *merk* tipe String dan biaya tipe int dengan akses *modifier* private.
- Tambahkan *constructor default* serta setter dan getter.
- Implementasikan method *hitungBiayaMobil*

```

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

```

Struktur class Mobil :

```

1  package raden.relasiclass.percobaan2;
2  /**
3   * @Raden Dimas Erlangga
4   */
5  public class Mobil {
6
7      private String merk;
8      private int biaya;
9
10     //constructor default
11     public Mobil() {
12     }
13
14     //setter getter
15     public void setMerk(String merk) {
16         this.merk = merk;
17     }
18     public String getMerk() {
19         return merk;
20     }
21
22
23     public void setBiaya(int biaya) {
24         this.biaya = biaya;
25     }
26     public int getBiaya() {
27         return biaya;
28     }
29
30     public int hitungBiayaMobil(int hari) {
31         return biaya * hari;
32     }
33 }

```


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

g. Implementasikan method *hitungBiayaSopir*

```
public int hitungBiayaSopir(int hari) {  
    return biaya * hari;  
}
```

Struktur class Sopir :

```
1 package raden.relasticlass.percobaan2;  
2 /**  
3  * @Raden Dimas Erlangga  
4  */  
5 public class Sopir {  
6  
7     private String nama;  
8     private int biaya;  
9  
10    public Sopir() {  
11    }  
12  
13    public void setNama(String nama) {  
14        this.nama = nama;  
15    }  
16    public String getNama() {  
17        return nama;  
18    }  
19  
20    public void setBiaya(int biaya) {  
21        this.biaya = biaya;  
22    }  
23    public int getBiaya() {  
24        return biaya;  
25    }  
26  
27    public int hitungBiayaSopir(int hari) {  
28        return biaya * hari;  
29    }  
30  
31 }
```

- h. Tambahkan *class* Pelanggan dengan *constructor default*.
- i. Tambahkan atribut-atribut dengan akses modifier *private* berikut:

Atribut	Tipe
nama	String
mobil	Mobil
sopir	Sopir
hari	int

j. Implementasikan *setter* dan *getter*.

k. Tambahkan method `hitungBiayaTotal`

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

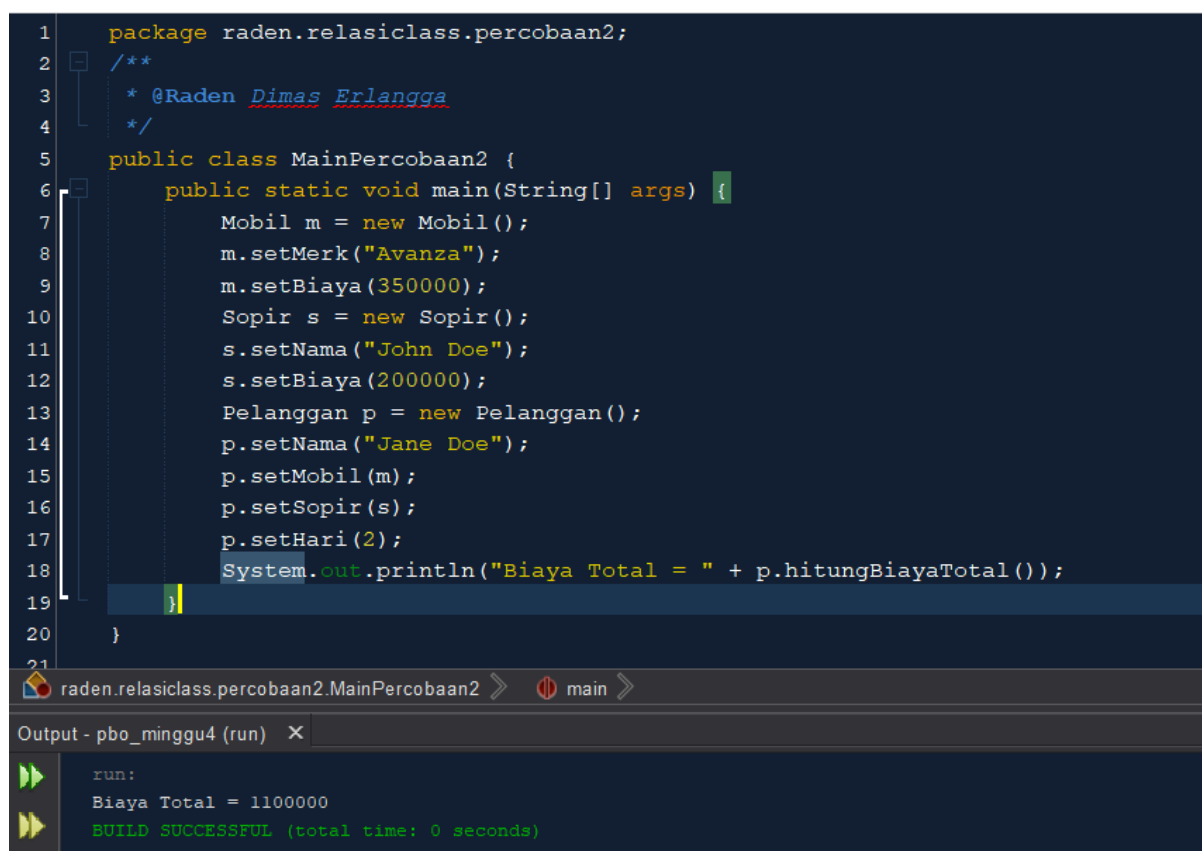
Struktur class Pelanggan :

```
1 package raden.relasticlass.percobaan2;
2
3 /**
4  * @Raden Dimas Erlangga
5  */
6 public class Pelanggan {
7     private String nama;
8     private Mobil mobil;
9     private Sopir sopir;
10    private int hari;
11
12    public Pelanggan() {
13    }
14
15    public void setNama(String nama) {
16        this.nama = nama;
17    }
18
19    public String getNama() {
20        return nama;
21    }
22
23    public void setMobil(Mobil mobil) {
24        this.mobil = mobil;
25    }
26
27    public Mobil getMobil() {
28        return mobil;
29    }
30
31    public void setSopir(Sopir sopir) {
32        this.sopir = sopir;
33    }
34
35    public Sopir getSopir() {
36        return sopir;
37    }
38
39    public void setHari(int hari) {
40        this.hari = hari;
41    }
42
43    public int getHari() {
44        return hari;
45    }
46
47    public int hitungBiayaTotal() {
48        return mobil.hitungBiayaMobil(hari) + sopir.hitungBiayaSopir(hari);
49    }
50 }
```

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

```
Mobil m = new
Mobil();
m.setMerk("Avanz
a");
m.setBiaya(35000
0); Sopir s =
new Sopir();
s.setNama("John
Doe");
s.setBiaya(20000
0);
Pelanggan p = new Pelanggan();
p.setNama("Jane
Doe");
p.setMobil(m);
p.setSopir(s);
p.setHari(2);
System.out.println("Biaya Total
= " + p.hitungBiayaTotal());
```

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



```
1 package raden.relasticlass.percobaan2;
2 /**
3  * @Raden Dimas Erlangga
4  */
5 public class MainPercobaan2 {
6     public static void main(String[] args) {
7         Mobil m = new Mobil();
8         m.setMerk("Avanza");
9         m.setBiaya(350000);
10        Sopir s = new Sopir();
11        s.setNama("John Doe");
12        s.setBiaya(200000);
13        Pelanggan p = new Pelanggan();
14        p.setNama("Jane Doe");
15        p.setMobil(m);
16        p.setSopir(s);
17        p.setHari(2);
18        System.out.println("Biaya Total = " + p.hitungBiayaTotal());
19    }
20 }
21
```

raden.relasticlass.percobaan2.MainPercobaan2 > main >

Output - pbo_minggu4 (run) X

```
run:
Biaya Total = 1100000
BUILD SUCCESSFUL (total time: 0 seconds)
```

Pertanyaan

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

Jawaban : pada kode program yang saya buat, relasi pelanggan dengan class Mboil dan class Sopir berada pada baris ke 7 & 8 :

```
5      public class Pelanggan {  
6          private String nama;  
7          private Mobil mobil;  
8          private Sopir sopir;  
9          private int hari;
```

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

Jawaban : dikarenakan return statement dari method tersebut adalah hasil dari perhitungan biaya dikali dengan hari maka parameter hari harus berada pada method hitungBiayaSopir dan hitungBiayaMobil

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

Jawaban : untuk memanggil method hitungBiayaMobil dan hitungBiayaSopir, agar dapat menampilkan hasil dari perhitungan biaya mobil perhari nya dan biaya sopir perhari nya.

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

Jawaban : sintaks p.setMobil(m) dan p.setSopir adalah sintak yang digunakan untuk menambahkan data berdasarkan hasil inputan objek m dan s.

```
public class MainPercobaan2 {  
    public static void main(String[] args) {  
        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());  
    }  
}
```

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

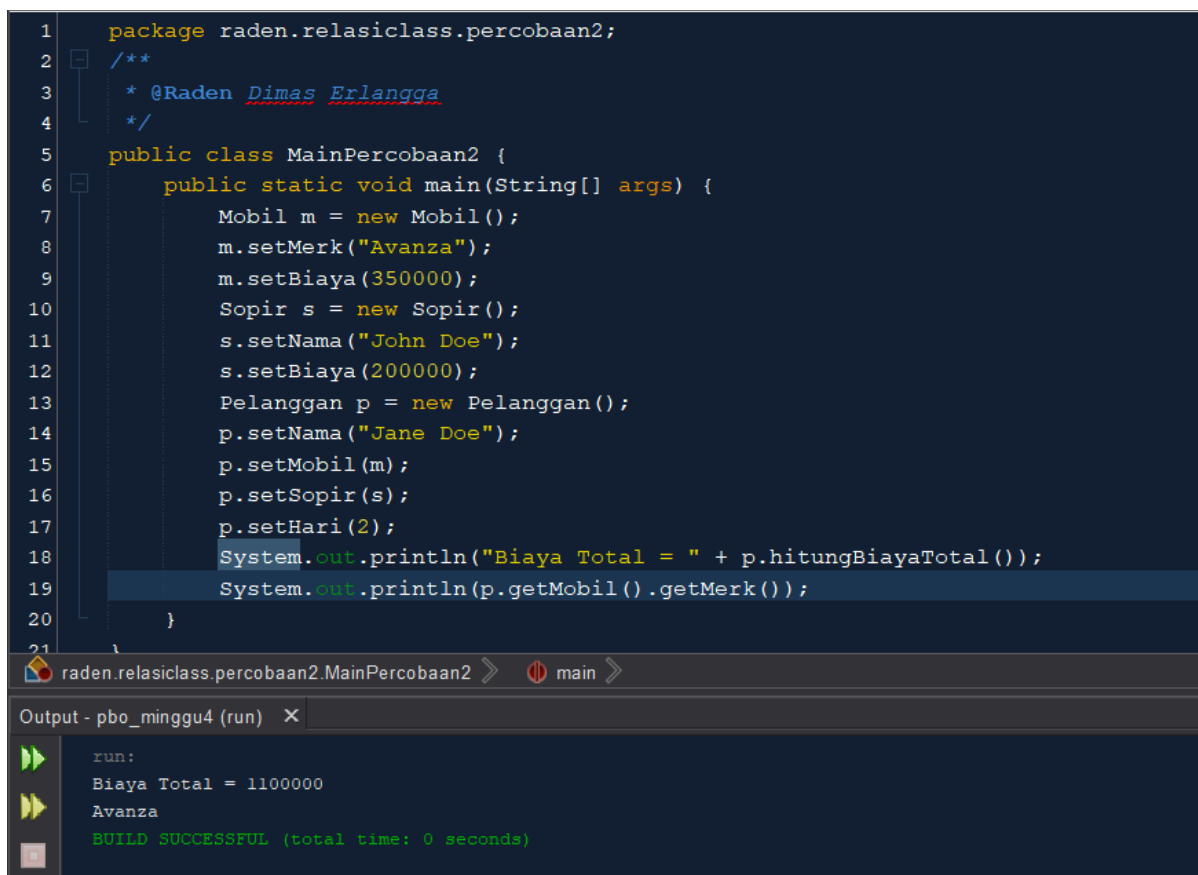
Jawaban : `p.hitungBiayaTotal` adalah perintah untuk menghitung biaya dari hasil method `hitungBiayaSopir` dan `hitungBiayaMobil`. Proses ini nantinya akan mengembalikan value dari hasil pertambahan tersebut.

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?

Jawaban : `p.getMobil().getMerk()` adalah method yang memanggil merk mobil sesuai dari inputan sebelum nya yang sudah di deklarasikan pada objek `m` di perintah `p.setMobil(m)`.



```
1 package raden.relasticlass.percobaan2;
2 /**
3  * @Raden Dimas Erlangga
4  */
5 public class MainPercobaan2 {
6     public static void main(String[] args) {
7         Mobil m = new Mobil();
8         m.setMerk("Avanza");
9         m.setBiaya(350000);
10        Sopir s = new Sopir();
11        s.setNama("John Doe");
12        s.setBiaya(200000);
13        Pelanggan p = new Pelanggan();
14        p.setNama("Jane Doe");
15        p.setMobil(m);
16        p.setSopir(s);
17        p.setHari(2);
18        System.out.println("Biaya Total = " + p.hitungBiayaTotal());
19        System.out.println(p.getMobil().getMerk());
20    }
21 }
```

raden.relasticlass.percobaan2.MainPercobaan2 > main >

Output - pbo_minggu4 (run) X

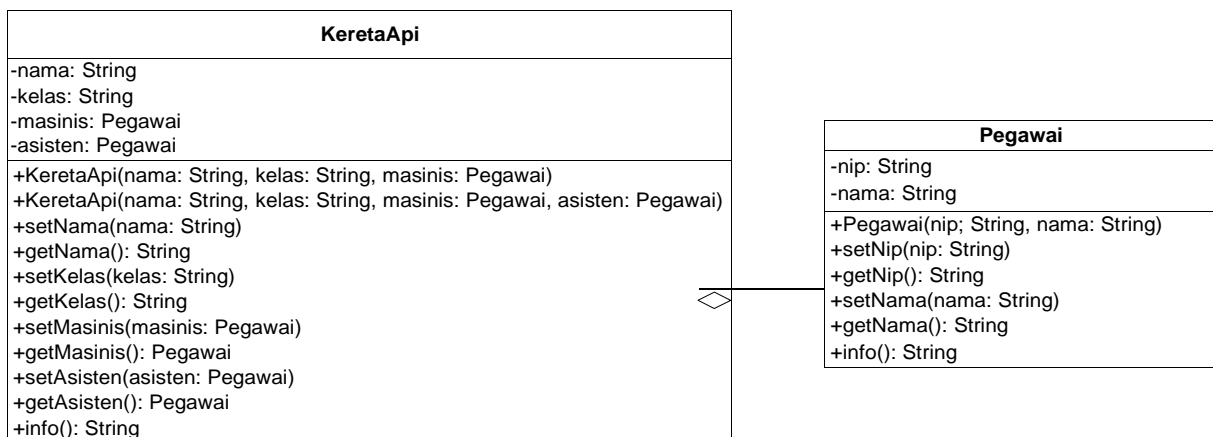
```
run:
Biaya Total = 1100000
Avanza
BUILD SUCCESSFUL (total time: 0 seconds)
```

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

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

```
private String nip;
```

```
private String nama;
```

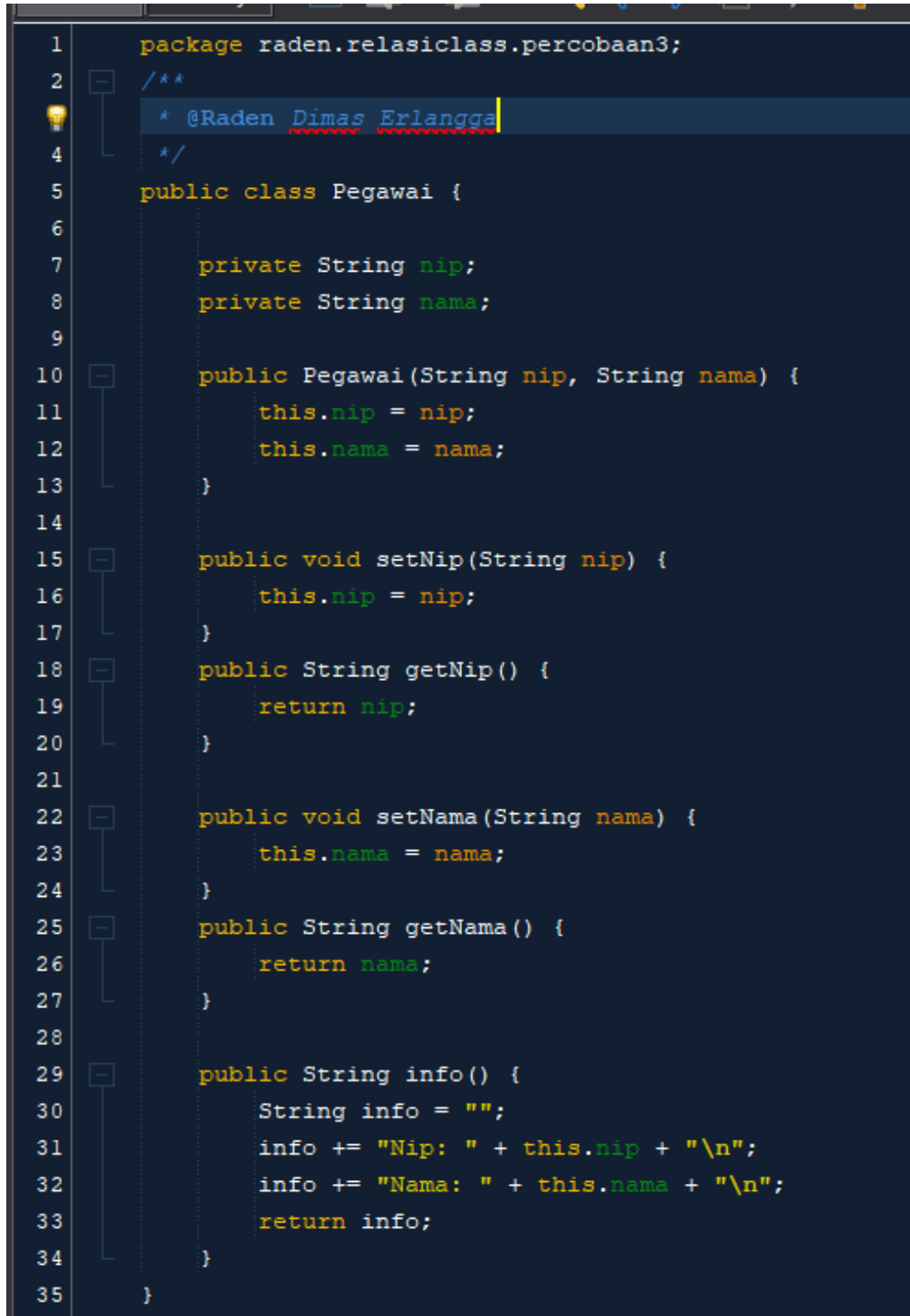
e. Buatlah *constructor* untuk *class* Pegawai dengan parameter nip dan nama.

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

g. 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;
}
```

Struktur class Pegawai :



```
1 package raden.relasticlass.percobaan3;
2 /**
3  * @Raden Dimas Erlangga
4  */
5 public class Pegawai {
6
7     private String nip;
8     private String nama;
9
10    public Pegawai(String nip, String nama) {
11        this.nip = nip;
12        this.nama = nama;
13    }
14
15    public void setNip(String nip) {
16        this.nip = nip;
17    }
18    public String getNip() {
19        return nip;
20    }
21
22    public void setNama(String nama) {
23        this.nama = nama;
24    }
25    public String getNama() {
26        return nama;
27    }
28
29    public String info() {
30        String info = "";
31        info += "Nip: " + this.nip + "\n";
32        info += "Nama: " + this.nama + "\n";
33        return info;
34    }
35 }
```

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

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

```
private String nama;
private String kelas;
private Pegawai masinis;
private Pegawai asisten;
```

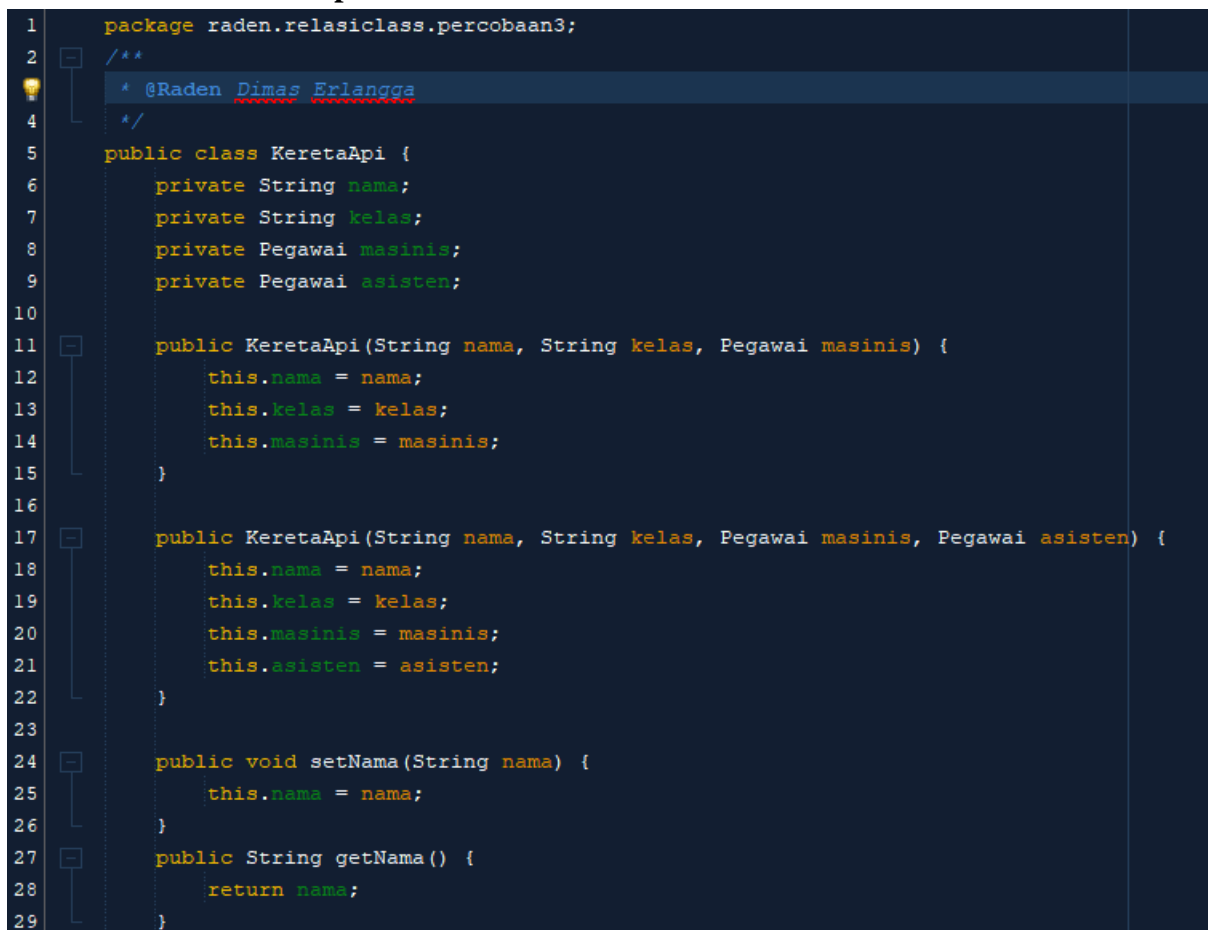
j. Tambahkan *constructor* 3 parameter (nama, kelas, masinis) serta 4 parameter (nama, kelas, masinis, asisten).

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

l. 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;
}
```

Struktur Class KeretaApi :



```
1 package raden.relasticlass.percobaan3;
2 /**
3  * @Raden Dimas Erlangga
4  */
5 public class KeretaApi {
6     private String nama;
7     private String kelas;
8     private Pegawai masinis;
9     private Pegawai asisten;
10
11     public KeretaApi(String nama, String kelas, Pegawai masinis) {
12         this.nama = nama;
13         this.kelas = kelas;
14         this.masinis = masinis;
15     }
16
17     public KeretaApi(String nama, String kelas, Pegawai masinis, Pegawai asisten) {
18         this.nama = nama;
19         this.kelas = kelas;
20         this.masinis = masinis;
21         this.asisten = asisten;
22     }
23
24     public void setNama(String nama) {
25         this.nama = nama;
26     }
27
28     public String getNama() {
29         return nama;
30     }
31 }
```



```

30
31 public void setKelas(String kelas) {
32     this.kelas = kelas;
33 }
34 public String getKelas() {
35     return kelas;
36 }
37
38 public void setMasinis(Pegawai masinis) {
39     this.masinis = masinis;
40 }
41 public Pegawai getMasinis() {
42     return masinis;
43 }
44
45 public void setAsisten(Pegawai asisten) {
46     this.asisten = asisten;
47 }
48 public Pegawai getAsisten() {
49     return asisten;
50 }
51
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     info += "Asisten: " + this.asisten.info() + "\n";
58     return info;
59 }
60 }

```

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

n. 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);

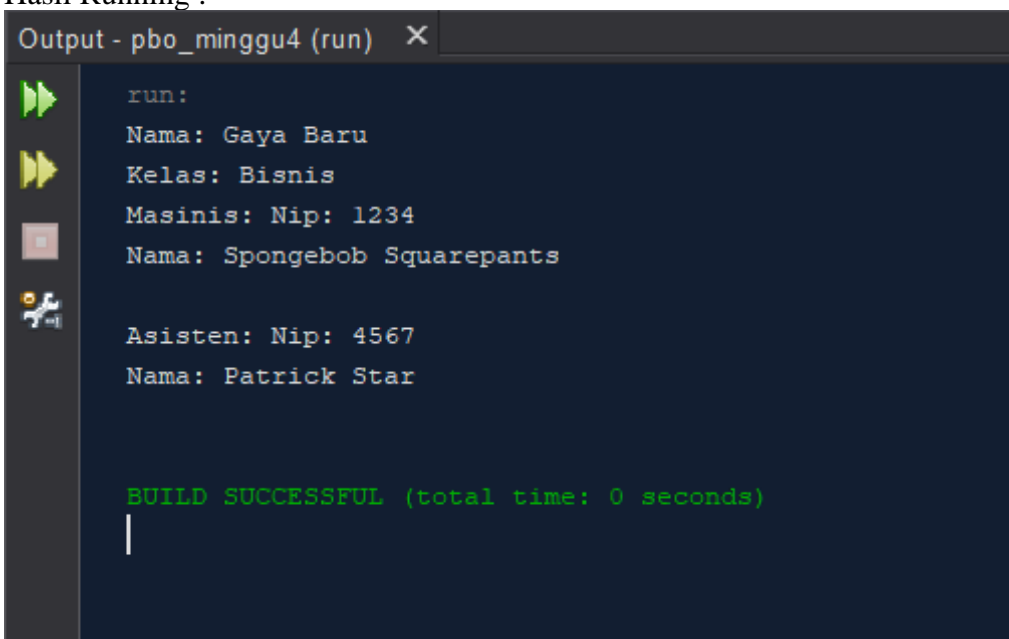
```

```

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

```

Hasil Running :



```

Output - pbo_minggu4 (run) X
run:
Nama: Gaya Baru
Kelas: Bisnis
Masinis: Nip: 1234
Nama: Spongebob Squarepants
Asisten: Nip: 4567
Nama: Patrick Star

BUILD SUCCESSFUL (total time: 0 seconds)

```

Pertanyaan

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

Jawaban : digunakan untuk memanggil *method* `info` untuk menampilkan nilai yang berasal dari atribut `masinis` dan `asisten`.

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());
```

Jawaban :

```
1 package raden.relasticlass.percobaan3;  
2 /**  
3  * @Raden Dimas Erlangga  
4  */  
5 public class MainPertanyaan {  
6     public static void main(String[] args) {  
7         Pegawai masinis = new Pegawai("1234", "Spongebob Squarepants");  
8         KeretaApi keretaApi = new KeretaApi("Gaya Baru", "Bisnis", masinis);  
9         System.out.println(keretaApi.info());  
10    }  
11 }
```

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

Jawaban :

```
Output - pbo_minggu4 (run) X  
run:  
Exception in thread "main" java.lang.NullPointerException: Cannot invoke "raden.relasticlass.percobaan3.Pegawai.info()" because "this.asisten" is null  
at raden.relasticlass.percobaan3.KeretaApi.info(KeretaApi.java:57)  
at raden.relasticlass.percobaan3.MainPertanyaan.main(MainPertanyaan.java:9)  
C:\Users\HP\AppData\Local\NetBeans\Cache\12.5\executor-snippets\run.xml:111: The following error occurred while executing this line:  
C:\Users\HP\AppData\Local\NetBeans\Cache\12.5\executor-snippets\run.xml:60: Java returned: 1  
BUILD FAILED (total time: 0 seconds)
```

Hal tersebut terjadi dikarenakan nilai `asisten` belum di masukkan

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

Jawaban :

```
1 package raden.relasticlass.percobaan3;  
2 /**  
3  * @Raden Dimas Erlangga  
4  */  
5 public class MainPertanyaan {  
6     public static void main(String[] args) {  
7         Pegawai masinis = new Pegawai("1234", "Spongebob Squarepants");  
8         Pegawai asisten = new Pegawai("4567", "Patrick Star");  
9         KeretaApi keretaApi = new KeretaApi("Gaya Baru", "Bisnis", masinis, asisten);  
10        System.out.println(keretaApi.info());  
11    }  
12 }  
13 }
```

```
Output - pbo_minggu4 (run) X  
run:  
Nama: Gaya Baru  
Kelas: Bisnis  
Masinis: Nip: 1234  
Nama: Spongebob Squarepants  
  
Asisten: Nip: 4567  
Nama: Patrick Star
```

Percobaan 4

Struktur Class Penumpang :

```
1 package raden.relasiclass.percobaan4;
2 /**
3  * @Raden Dimas Erlangga
4  */
5 public class Penumpang {
6
7     private String ktp;
8     private String nama;
9
10    public Penumpang(String ktp, String nama) {
11        this.ktp = ktp;
12        this.nama = nama;
13    }
14
15    public void setKtp(String ktp) {
16        this.ktp = ktp;
17    }
18    public String getKtp() {
19        return ktp;
20    }
21
22    public void setName(String nama) {
23        this.nama = nama;
24    }
25    public String getName() {
26        return nama;
27    }
28
29    public String info() {
30        String info = "";
31        info += "Ktp: " + ktp + "\n";
32        info += "Nama: " + nama + "\n";
33        return info;
34    }
35 }
```

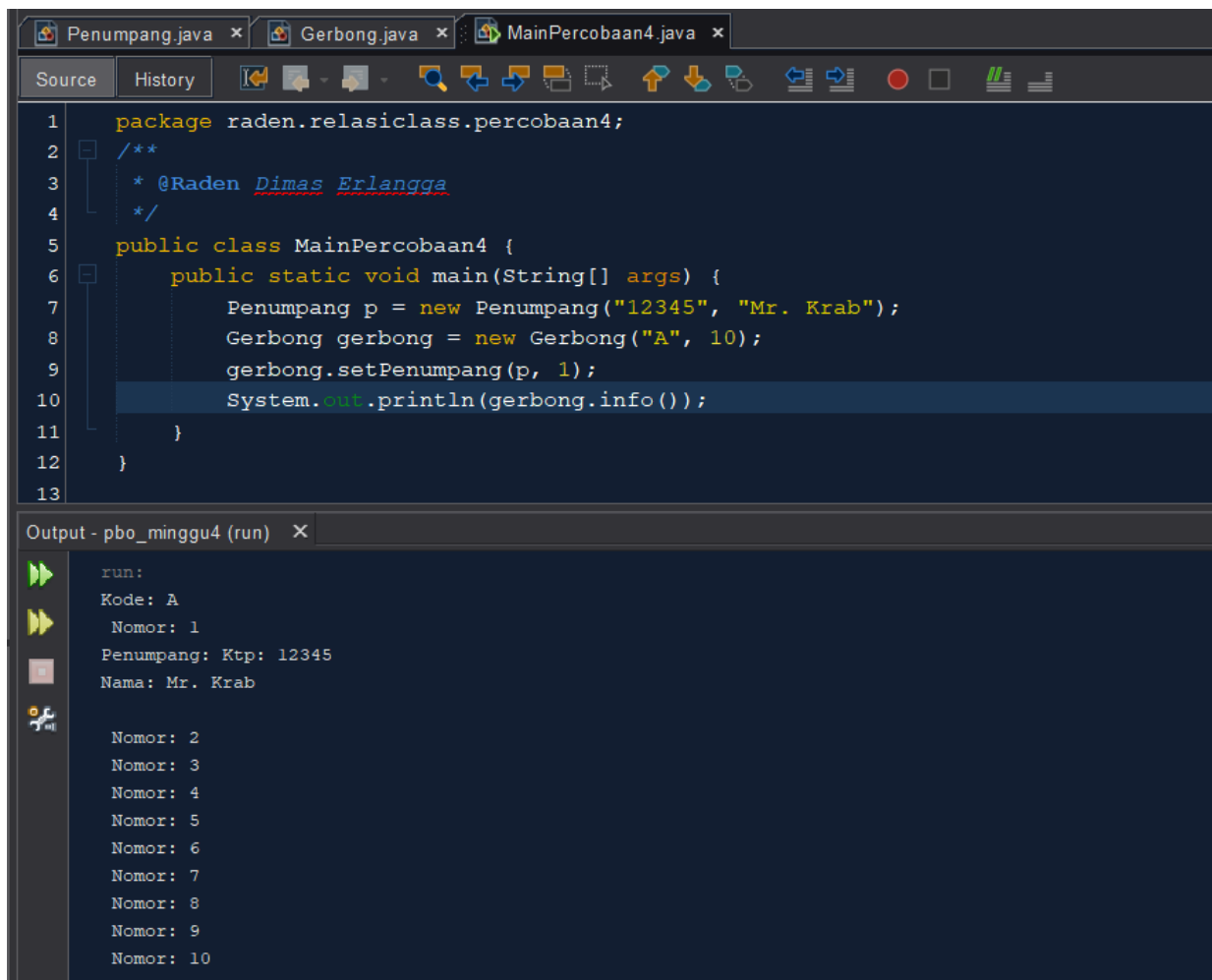
Struktur Class Kursi :

```
1 package raden.relasticlass.percobaan4;
2 /**
3  * @Raden Dimas Erlangga
4  */
5 public class Kursi {
6     private String nomor;
7     private Penumpang Penumpang;
8     public Kursi(String nomor) {
9         this.nomor = nomor;
10    }
11
12    public void setNomor(String nomor) {
13        this.nomor = nomor;
14    }
15
16    public String getNomor() {
17        return nomor;
18    }
19
20    public void setPenumpang(Penumpang Penumpang) {
21        this.Penumpang = Penumpang;
22    }
23
24    public Penumpang getPenumpang() {
25        return Penumpang;
26    }
27
28    public String info() {
29        String info = " ";
30        info += "Nomor: " + nomor + "\n";
31        if (this.Penumpang != null) {
32            info += "Penumpang: " + Penumpang.info() + "\n";
33        }
34        return info;
35    }
36 }
```

Struktur Class Gerbong :

```
1  package raden.relaiclass.percobaan4;
2  /**
3   * @Raden Dimas Erlangga
4   */
5  public class Gerbong {
6
7      private String kode;
8      private Kursi[] arrayKursi;
9
10     public Gerbong(String kode, Kursi[] arrayKursi) {
11         this.kode = kode;
12         this.arrayKursi = arrayKursi;
13     }
14
15     public void setKode(String kode) {
16         this.kode = kode;
17     }
18     public String getKode() {
19         return kode;
20     }
21
22
23     public Kursi[] getArrayKursi() {
24         return arrayKursi;
25     }
26     private void initKursi() {
27         for (int i = 0; i < arrayKursi.length; i++) {
28             this.arrayKursi[i] = new Kursi(String.valueOf(i + 1));
29         }
30     }
31
32     public Gerbong(String kode, int jumlah) {
33         this.kode = kode;
34         this.arrayKursi = new Kursi[jumlah];
35         this.initKursi();
36     }
37
38     public String info() {
39         String info = "";
40         info += "Kode: " + kode + "\n";
41         for (Kursi kursi : arrayKursi) {
42             info += kursi.info();
43         }
44         return info;
45     }
46
47     public void setPenumpang(Penumpang penumpang, int nomor) {
48         this.arrayKursi[nomor - 1].setPenumpang(penumpang);
49     }
50 }
```

Struktur Class Main :



The screenshot shows an IDE with three tabs: Penumpang.java, Gerbong.java, and MainPercobaan4.java. The MainPercobaan4.java tab is active, displaying the following code:

```
1 package raden.relasticlass.percobaan4;
2 /**
3  * @Raden Dimas Erlangga
4  */
5 public class MainPercobaan4 {
6     public static void main(String[] args) {
7         Penumpang p = new Penumpang("12345", "Mr. Krab");
8         Gerbong gerbong = new Gerbong("A", 10);
9         gerbong.setPenumpang(p, 1);
10        System.out.println(gerbong.info());
11    }
12 }
13
```

Below the code editor is an output window titled "Output - pbo_minggu4 (run)". It shows the following output:

```
run:
Kode: A
Nomor: 1
Penumpang: Ktp: 12345
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 ?

Jawaban : ada 10 jumlah kursi dalam Gerbong A

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

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

Jawaban : maksud dari kode tersebut adalah if statement apabila isi variable penumpang tidak kosong, maka, maka akan menampilkan informasi penumpang.

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

Jawaban : karena nilai array itu dimulai dari 0 bukan 1, jadi penumpang pertama terdapat pada array ke-0.

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

Jawaban :

```
1 package raden.relasticlass.percobaan4;
2 /**
3  * @Raden Dimas Erlangga
4  */
5 public class MainPercobaan4 {
6     public static void main(String[] args) {
7         Penumpang p = new Penumpang("12345", "Mr. Krab");
8         Gerbong gerbong = new Gerbong("A", 10);
9         gerbong.setPenumpang(p, 1);
10        System.out.println(gerbong.info());
11        gerbong.setPenumpang("budi", 1);
12    }
13 }
```

Tidak bisa, karena parameter set penumpang hanya mengambil dari tipe data class penumpang, jadi jika menginputkan tipe data selain objek Penumpang, akan terjadi error.

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

Jawaban :

```
46
47 public void setPenumpang(Penumpang penumpang, int nomor) {
48     if (this.arrayKursi[nomor - 1].getPenumpang() != null) {
49         System.out.println("maaf kursi sudah ter-isi, silahkan pilih kursi lain");
50     } else {
51         this.arrayKursi[nomor - 1].setPenumpang(penumpang);
52     }
53 }
54 }
```

```
1 package raden.relasticlass.percobaan4;
2 /**
3  * @Raden Dimas Erlangga
4  */
5 public class MainPercobaan4 {
6     public static void main(String[] args) {
7         Penumpang p = new Penumpang("12345", "Mr. Krab");
8         Penumpang b = new Penumpang("11111", "Budi");
9         Gerbong gerbong = new Gerbong("A", 10);
10        gerbong.setPenumpang(p, 1);
11        gerbong.setPenumpang(b, 1);
12        System.out.println(gerbong.info());
13    }
14 }
```

Output - pbo_minggu4 (run)

```
run:
maaf kursi sudah ter-isi, silahkan pilih kursi lain
Kode: A
Nomor: 1
Penumpang: Ktp: 12345
Nama: Mr. Krab

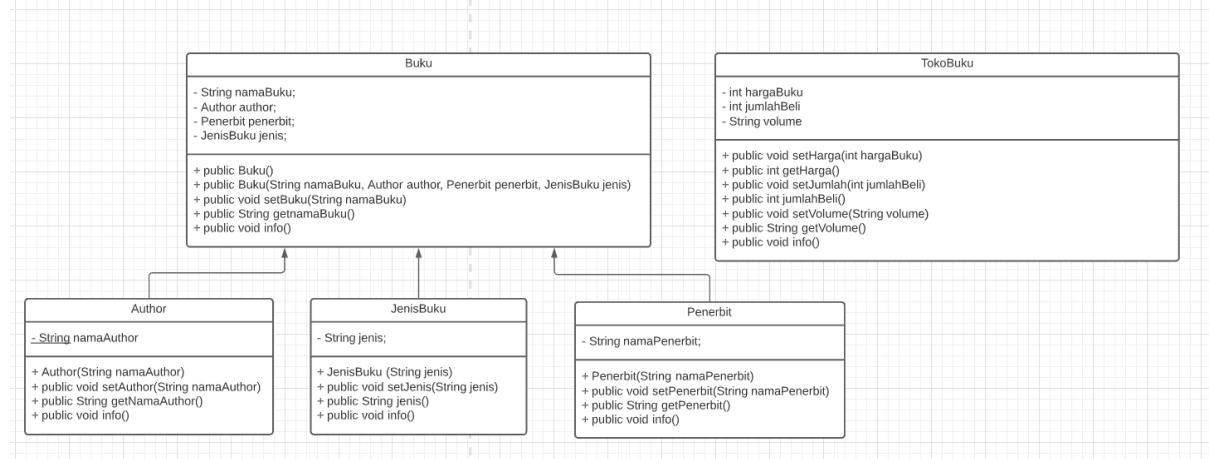
Nomor: 2
Nomor: 3
Nomor: 4
Nomor: 5
Nomor: 6
Nomor: 7
Nomor: 8
Nomor: 9
Nomor: 10

BUILD SUCCESSFUL (total time: 0 seconds)
```

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).

Disini saya membuat desain sistem kasir pada sebuah toko buku, dengan konsep class relasi dimana class Author, JenisBuku dan Penerbit ber-relasi dengan class Buku.



Terdapat tambahan class TokoBuku, dimana class tersebut tidak ber-relasi dengan buku secara langsung, namun ber-relasi dari hal konteks, dimana class TokoBuku mengatasi perhitungan jumlah buku yang di beli, yang dimana nanti nya class tersebut akan di-panggil pada class main.

A. Class Author :

```
1 package raden.relasiclass.tugas;
2 /**
3  * Raden Dimas Erlangga
4  */
5 public class Author {
6     private String namaAuthor;
7
8     Author(String namaAuthor) {
9         this.namaAuthor = namaAuthor;
10    }
11    public void setAuthor(String namaAuthor) {
12        this.namaAuthor = namaAuthor;
13    }
14    public String getNamaAuthor() {
15        return namaAuthor;
16    }
17    public void info() {
18        System.out.print("Author\t : " + namaAuthor + "\n");
19    }
20 }
```


B. Class Penerbit

```
1 package raden.relasticlass.tugas;
2 /**
3  * Raden Dimas Erlangga
4  */
5 public class Penerbit {
6     private String namaPenerbit;
7
8     Penerbit(String namaPenerbit){
9         this.namaPenerbit = namaPenerbit;
10    }
11    public void setPenerbit(String namaPenerbit){
12        this.namaPenerbit = namaPenerbit;
13    }
14    public String getPenerbit(){
15        return namaPenerbit;
16    }
17    public void info(){
18        System.out.print("Penerbit : " + namaPenerbit + "\n");
19    }
20 }
```

C. Class Jenis Buku

```
1 package raden.relasticlass.tugas;
2 /**
3  * Raden Dimas Erlangga
4  */
5 public class JenisBuku {
6     private String jenis;
7
8     JenisBuku(String jenis){
9         this.jenis= jenis;
10    }
11    public void setJenis(String jenis){
12        this.jenis = jenis;
13    }
14    public String jenis(){
15        return jenis;
16    }
17    public void info(){
18        System.out.print("Jenis Buku : " + jenis + "\n");
19    }
20 }
```

D. Class Buku

```
1 package raden.relasiclass.tugas;
2 /**
3  * Raden Dimas Erlangga
4  */
5 public class Buku {
6     private String namaBuku;
7     private Author author;
8     private Penerbit penerbit;
9     private JenisBuku jenis;
10
11     public Buku() {
12     }
13
14     public Buku(String namaBuku, Author author, Penerbit penerbit, JenisBuku jenis) {
15         this.namaBuku = namaBuku;
16         this.author = author;
17         this.penerbit = penerbit;
18         this.jenis = jenis;
19     }
20
21     public void setBuku(String namaBuku) {
22         this.namaBuku = namaBuku;
23     }
24     public String getnamaBuku() {
25         return namaBuku;
26     }
27     public void info() {
28         System.out.print("Buku\t : " + namaBuku + "\n");
29         author.info();
30         penerbit.info();
31         jenis.info();
32     }
33 }
```

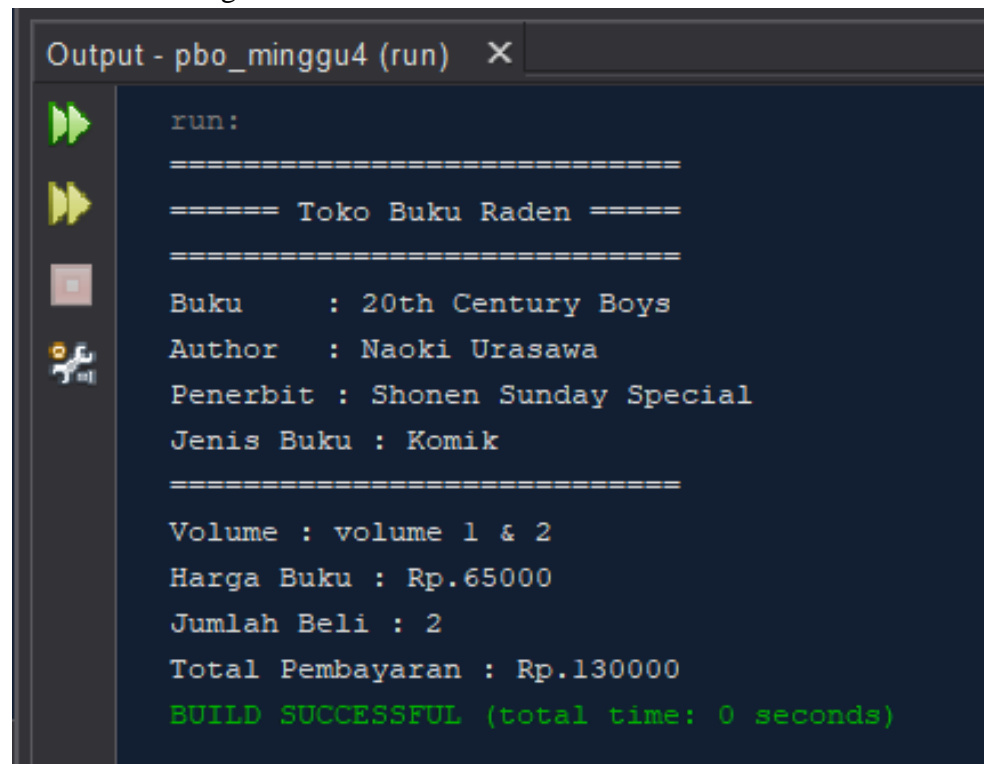
E. Class TokoBuku

```
1 package raden.relasiclass.tugas;
2 /**
3  * Raden Dimas Erlangga
4  */
5 public class TokoBuku {
6     private int hargaBuku;
7     private int jumlahBeli;
8     private String volume;
9
10     TokoBuku(int hargaBuku, int jumlahBeli, String volume) {
11         this.hargaBuku = hargaBuku;
12         this.jumlahBeli = jumlahBeli;
13         this.volume = volume;
14     }
15
16     public void setHarga(int hargaBuku) {
17         this.hargaBuku = hargaBuku;
18     }
19     public int getHarga() {
20         return hargaBuku;
21     }
22
23     public void setJumlah(int jumlahBeli) {
24         this.jumlahBeli = jumlahBeli;
25     }
26     public int jumlahBeli() {
27         return jumlahBeli;
28     }
29
30     public void setVolume(String volume) {
31         this.volume = volume;
32     }
33     public String getVolume() {
34         return volume;
35     }
36
37     public void info() {
38         System.out.print("Volume : " + volume + "\n");
39         System.out.print("Harga Buku : Rp." + hargaBuku + "\n");
40         System.out.print("Jumlah Beli : " + jumlahBeli + "\n");
41         System.out.println("Total Pembayaran : Rp." + jumlahBeli * hargaBuku);
42     }
43 }
```

F. Class Main

```
1  package raden.relasticlass.tugas;
2  /**
3   * Raden Dimas Erlangga
4   */
5  public class MainTugas {
6      public static void main(String[] args) {
7          //sistem informasi kasir toko buku
8          System.out.println("=====");
9          System.out.println("===== Toko Buku Raden =====");
10         System.out.println("=====");
11         Author a = new Author("Naoki Urasawa");
12         Penerbit p = new Penerbit("Shonen Sunday Special");
13         JenisBuku j = new JenisBuku("Komik");
14         Buku b = new Buku("20th Century Boys", a, p, j);
15         b.info();
16         System.out.println("=====");
17         TokoBuku t = new TokoBuku(65000, 2, "volume 1 & 2");
18         t.info();
19     }
20 }
```

G. Hasil Running :



```
Output - pbo_minggu4 (run) X
run:
=====
===== Toko Buku Raden =====
=====
Buku      : 20th Century Boys
Author    : Naoki Urasawa
Penerbit  : Shonen Sunday Special
Jenis Buku : Komik
=====
Volume : volume 1 & 2
Harga Buku : Rp.65000
Jumlah Beli : 2
Total Pembayaran : Rp.130000
BUILD SUCCESSFUL (total time: 0 seconds)
```