# Jobsheet 4- Relasi Class

| | |
|---|---|
| @ Name | Davis Maulana Hermanto |
| Class | TI 2i |
| # NIM | 2241720255 |
| Subject | Object Oriented Programming |
| Type | Assignment |
| Semester | Semester 3 |
| Time | @October 4, 2023 |

## Pertanyaan 1

1. Di dalam class Processor dan class Laptop , terdapat method setter dan getter untuk
   masing-masing atributnya. Apakah gunanya method setter dan getter tersebut ?
   a. Setter : digunakan untuk mengatur nilai suatu atribut seperti di class Processor, digunakan untuk mengatur nilai atribut “merk” dan “cache” dan di class Laptop mengatur nilai atribut “merk” dan “proc”.
   b. Getter : Digunakan untuk mengambil nilai dari suatu atribut.
2. Di dalam class Processor dan class Laptop, masing-masing terdapat konstruktor default dan konstruktor berparameter. Bagaimanakah beda penggunaan dari kedua jenis
   konstruktor tersebut ?
   a. Konstruktor Default : jika kita memanggil class yang berkonstruktor default, kita tidak perlu melakukan instansiasi nilai atributnya pada saat pendeklarasian object. Jadi nilai atribut sudah diinstansiasi didalam konstruktor.

b. Konstruktor Berparameter : jika kita memanggil class yang berkonstruktor berparameter, kita perlu melakukan instansiasi nilai atributnya pada saat pendeklarasian object.

3. Perhatikan class Laptop, di antara 2 atribut yang dimiliki (merk dan proc), atribut manakah yang bertipe object ?
   a. proc
4. Perhatikan class Laptop, pada baris manakah yang menunjukan bahwa class Laptop
   memiliki relasi dengan class Processor ?

```
private Processor proc;
```

```
proc.info();
```

5. Perhatikan pada class Laptop , Apakah guna dari sintaks proc.info() ?
   a. Digunakan untuk memanggil method info milik class Processor.
6. Pada class MainPercobaan1, terdapat baris kode:
   Laptop l = new Laptop("Thinkpad", p);.
   a. Apakah p tersebut ?
      i. itu adalah object yang dideklarasikan dari class Processor
   b. 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 ?

      - Hasilnya tidak ada perubahan, akan tetapi bisa disimpulkan bahwa itu adalah cara untuk menggabungkan 2 object menjadi 1 object saja.

```
Merk Laptop = Thinkpad
Merk Processor = Intel i5
Cache Memory = 3.00
```

# Pertanyaan 2

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

```
private Mobil mobil;
2 usages
private Sopir sopir;
```

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

2. Perhatikan method hitungBiayaSopir pada class Sopir, serta method hitungBiayaMobil pada class Mobil. Mengapa menurut Anda method tersebut harus memiliki argument hari ?
   a. Karena ya itu adalah salah satu cara untuk menghitung total biaya yang harus dibayar oleh pelanggan sesuai dengan jumlah hari yang ditentukan.
3. Perhatikan kode dari class Pelanggan. Untuk apakah perintah mobil.hitungBiayaMobil(hari) dan sopir.hitungBiayaSopir(hari) ?
   a. Untuk memanggil method hitungBiayaMobil dari class Mobil dan hitungBiayaSopir dari class Sopir
4. Perhatikan class MainPercobaan2. Untuk apakah sintaks p.setMobil(m) dan p.setSopir(s) ?
   a. Untuk mengisi atribut “mobil” dan “sopir” pada class Pelanggan sesuai dengan nilai atribut yang telah di tentukan pada classnya masing masing.
5. Perhatikan class MainPercobaan2. Untuk apakah proses p.hitungBiayaTotal() tersebut ?
   a. Untuk memanggil method hitungBiayaTotal pada class Pelanggan dan mendapatkan nilai yang dihasilkan pada method tersebut.

6. Perhatikan class MainPercobaan2, coba tambahkan pada baris terakhir dari method
   main dan amati perubahan saat di-run!

```
Biaya Total = 1100000
Avanza
```

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

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

   - Untuk mendapatkan nilai merek dari atribut “merk” pada class Mobil

# Pertanyaan 3

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

   a. Untuk memanggil method info() didalam 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());

```
public static void main(String[] args) {
    Pegawai masinis = new Pegawai( nip: "1234",  nama: "Spongebob Squarepants");
    KeretaApi keretaApi = new KeretaApi( nama: "Gaya Baru",  kelas: "Bisnis", masinis)
    System.out.println(keretaApi.info());
}
```

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

```
Exception in thread "main" java.lang.NullPointerException Create breakpoint
    at ac.relasiclass.percobaan3.KeretaApi.info(KeretaApi.java:58)
    at ac.relasiclass.percobaan3.MainPertanyaan.main(MainPertanyaan.java:7)
```

4. Perbaiki class KeretaApi sehingga program dapat berjalan !

```
public class MainPertanyaan {
    no usages
    public static void main(String[] args) {
        Pegawai masinis = new Pegawai( nip: "1234", nama: "Spongebob Squarepants");
        Pegawai asisten = new Pegawai( nip: "9999", nama: "John");
        KeretaApi keretaApi = new KeretaApi( nama: "Gaya Baru", kelas: "Bisnis", masinis, asisten);
        System.out.println(keretaApi.info());
    }
}
```

```
Nama: Gaya Baru
Kelas: Bisnis
Asisten: Nip: 9999
Nama: John

Masinis: Nip: 1234
Nama: Spongebob Squarepants
```

# Pertanyaan 4

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

```
Kode: A01
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
```

- Ada 10 kursi

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

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

- Jika nilai pada atribut penumpang tidak null, maka akan ditampilkan nilai dari atribut penumpang. Jadi potongan kode tersebut bermaksud untuk mengecek terlebih dahulu apakah nilai dari semua atribut yang ada di class Penumpang telah terisi atau belum. Jika sudah maka akan ditampilkan semua nilai atribut yang ada di dalam class Penumpang.

3. Mengapa pada method setPenumpang() dalam class Gerbong, nilai nomor dikurangi

dengan angka 1 ?

- Agar bisa meletakkan nilai dari atribut “nomor” dengan posisi yang tepat sesuai dengan nilai atributnya di dalam array karena pada dasarnya array mulainya dari angka 0 bukan 1.

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

```
package ac.relasiclass.percobaan4;

no usages
public class MainPercobaan4 {
    no usages
    public static void main(String[] args) {
        Penumpang p = new Penumpang( ktp: "12345",  nama: "Mr. Krab");
        Gerbong gerbong = new Gerbong( kode: "A01",  jumlah: 10);
        gerbong.setPenumpang(p,  nomor: 1);

        Penumpang budi = new Penumpang( ktp: "99999",  nama: "Budi");
        gerbong.setPenumpang(budi,  nomor: 1);

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

```
Kode: A01
Nomor: 1
Penumpang: KTP: 99999
Nama: Budi

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

- Tempat duduk Mr. Krab diduduki oleh Budi

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

```
Codeium: Refactor | Explain | Generate Javadoc
public void setPenumpang (Penumpang penumpang, int nomor){
    if (arrayKursi[nomor-1].getPenumpang() == null){
        arrayKursi[nomor-1].setPenumpang(penumpang);
    }else{
        System.out.println("Kursi " + nomor + " sudah terisi");
    }
}
```

# Tugas

- Berdasarkan latihan di pertemuan teori, 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)

- Class Penumpang

```java
package ac.relasiclass.percobaan4;

public class Penumpang {
    private String ktp;
    private String nama;

    public Penumpang(String ktp, String nama){
        this.ktp = ktp;
        this.nama = nama;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setKtp(String ktp) {
        this.ktp = ktp;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public String getKtp() {
        return ktp;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setNama(String nama) {
        this.nama = nama;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public String getNama() {
        return nama;
    }
```

```
    Codeium: Refactor | Explain | Generate Javadoc
    public String info(){
        String info = "";
        info += "KTP: " + ktp + "\n";
        info += "Nama: " + nama + "\n";
        return info;
    }
}
```

- Class Kursi

```java
package ac.relasiclass.percobaan4;

public class Kursi {
    private String nomor;
    private Penumpang penumpang;

    public Kursi(String nomor){
        this.nomor = nomor;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setNomor(String nomor) {
        this.nomor = nomor;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public String getNomor() {
        return nomor;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setPenumpang(Penumpang penumpang) {
        this.penumpang = penumpang;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public Penumpang getPenumpang() {
        return penumpang;
    }
```

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

- Class Stasiun

```java
package ac.relasiclass.percobaan4;

public class Stasiun {
    String nama;
    String kota;

    public Stasiun(String nama, String kota){
        this.nama = nama;
        this.kota = kota;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setNama(String nama) {
        this.nama = nama;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public String getNama() {
        return nama;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setKota(String kota) {
        this.kota = kota;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public String getKota() {
        return kota;
    }
```

```
    Codeium: Refactor | Explain | Generate Javadoc
    public String info(){
        String info = "";
        info += "Nama Stasiun: " + nama + "\n";
        info += "Kota: " + kota + "\n";
        return info;
    }

}
```

- Class Gerbong

```
package ac.relasiclass.percobaan4;

public class Gerbong {
    private String kode;
    private Kursi[] arrayKursi;
    private Stasiun stasiun;

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

    Codeium: Refactor | Explain | Generate Javadoc
    private void initKursi(){
        for (int i = 0; i < arrayKursi.length; i++){
            this.arrayKursi[i] = new Kursi(String.valueOf(i + 1));
        }
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setKode(String kode) {
        this.kode = kode;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public String getKode() {
        return kode;
    }
```

```
    Codeium: Refactor | Explain | Generate Javadoc
    public String info(){
        String info = "";
        info += stasiun.info() + "\n";
        info += "Kode: " + kode + "\n";
        for (Kursi kursi : arrayKursi){
            info += kursi.info();
        }
        return info;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setPenumpang (Penumpang penumpang, int nomor){
        if (arrayKursi[nomor-1].getPenumpang() == null){
            arrayKursi[nomor-1].setPenumpang(penumpang);
        }else{
            System.out.println("Kursi " + nomor + " sudah terisi");
        }
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public Kursi[] getArrayKursi (){
        return this.arrayKursi;
    }

}
```

- Class Main

```java
package ac.relasiclass.percobaan4;

public class MainPercobaan4 {
    Run | Debug | Codeium: Refactor | Explain | Generate Javadoc
    public static void main(String[] args) {
        Stasiun stasiun = new Stasiun(nama:"Kota Lama", kota:"Malang");
        Gerbong gerbong = new Gerbong(kode:"A01", jumlah:10,stasiun);

        Penumpang p1 = new Penumpang(ktp:"12345", nama:"Mr. Krab");
        gerbong.setPenumpang(p1, nomor:1);

        Penumpang p2 = new Penumpang(ktp:"67890", nama:"Rusdi");
        gerbong.setPenumpang(p2, nomor:2);

        Penumpang p3 = new Penumpang(ktp:"11111", nama:"Joko");
        gerbong.setPenumpang(p3, nomor:3);

        Penumpang p4 = new Penumpang(ktp:"22222", nama:"Budi");
        gerbong.setPenumpang(p4, nomor:4);

        Penumpang p5 = new Penumpang(ktp:"33333", nama:"Siti");
        gerbong.setPenumpang(p5, nomor:5);


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