PEMROGRAMAN BERBASIS OBJEK
JOBSHEET 4

MUHAMMAD RAYHAN ZAMZAMI
244107020027
TI-2G
PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNOLOGI INFORMASI
POLITEKNIK NEGERI MALANG
2025

PERCOBAAN 1

VERIFIKASI HASIL PERCOBAAN

Class Laptop

```
Windsurf: Refactor | Explain
public class Laptop {
    private String merk;
    private Processor proc;

    public Laptop() {
    }

    public Laptop(String merk, Processor proc) {
        this.merk = merk;
        this.proc = proc;
    }

    Windsurf: Refactor | Explain | Generate Javadoc | ×
    public void setMerk(String merk) {
        this.merk = merk;
    }

    Windsurf: Refactor | Explain | Generate Javadoc | ×
    public String getMerk() {
        return merk;
    }

    Windsurf: Refactor | Explain | Generate Javadoc | ×
    public void setProc(Processor proc) {
        this.proc = proc;
    }

    Windsurf: Refactor | Explain | Generate Javadoc | ×
    public Processor getProc() {
        return proc;
    }

    Windsurf: Refactor | Explain | Generate Javadoc | ×
    public void info() {
        System.out.println("Merk Laptop = " + merk);
        if (proc != null) {
            proc.info();
        } else {
            System.out.println(x:"Processor : (null)");
        }
    }
}
```

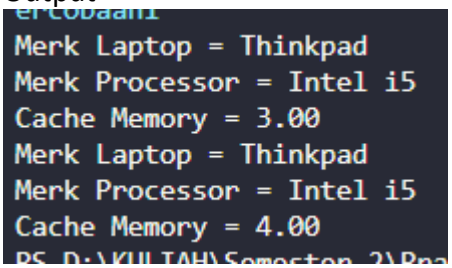
Class Processor

```java
Processor.java > Processor > Processor(String, double)
Windsurf: Refactor | Explain
public class Processor {
    private String merk;
    private double cache;

    public Processor() {
    }

    public Processor(String merk, double cache) {
        this.merk = merk;
        this.cache = cache;
    }

    Windsurf: Refactor | Explain | Generate Javadoc | ×
    public void setMerk(String merk) {
        this.merk = merk;
    }

    Windsurf: Refactor | Explain | Generate Javadoc | ×
    public String getMerk() {
        return merk;
    }

    Windsurf: Refactor | Explain | Generate Javadoc | ×
    public void setCache(double cache) {
        this.cache = cache;
    }

    Windsurf: Refactor | Explain | Generate Javadoc | ×
    public double getCache() {
        return cache;
    }

    Windsurf: Refactor | Explain | Generate Javadoc | ×
    public void info() {
        System.out.printf(format:"Merk Processor = %s\n", merk);
        System.out.printf(format:"Cache Memory = %.2f\n", cache);
    }
}
```

Class Main

```java
Windsurf: Refactor | Explain
public class MainPercobaan1 {
    Run | Debug | Windsurf: Refactor | Explain | Generate Javadoc | ×
    public static void main(String[] args) {
        Processor p = new Processor(merk:"Intel i5", cache:3);
        Laptop L = new Laptop(merk:"Thinkpad", p);
        L.info();

        Processor p1 = new Processor();
        p1.setMerk(merk:"Intel i5");
        p1.setCache(cache:4);

        Laptop L1 = new Laptop();
        L1.setMerk(merk:"Thinkpad");
        L1.setProc(p1);
        L1.info();
    }

}
```

Output



Pertanyaan

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

   Setter dan getter berfungsi sebagai penghubung aman antara atribut private dalam kelas dengan kode di luar kelas, sesuai prinsip enkapsulasi.

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

   Default constructor : objek dibuat kosong dulu, lalu atribut diisi belakangan lewat setter.
   Parameterized constructor : objek dibuat langsung beserta nilai atribut awalnya.

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

   Processor merupakan sebuah atribut yang bertipe object

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

   private Processor proc; di kelas Laptop

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

   untuk memamggil metode info() dari objek processor yang meruapakan atribut dari kelas Laptop

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

   sebuah objek dari kelas Processor yang sudah diinstasiasi sebelumnya

   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 ?

   tidak ada perubahan pada output program, perbedaanya hanya pada cara instasiasi objeknya

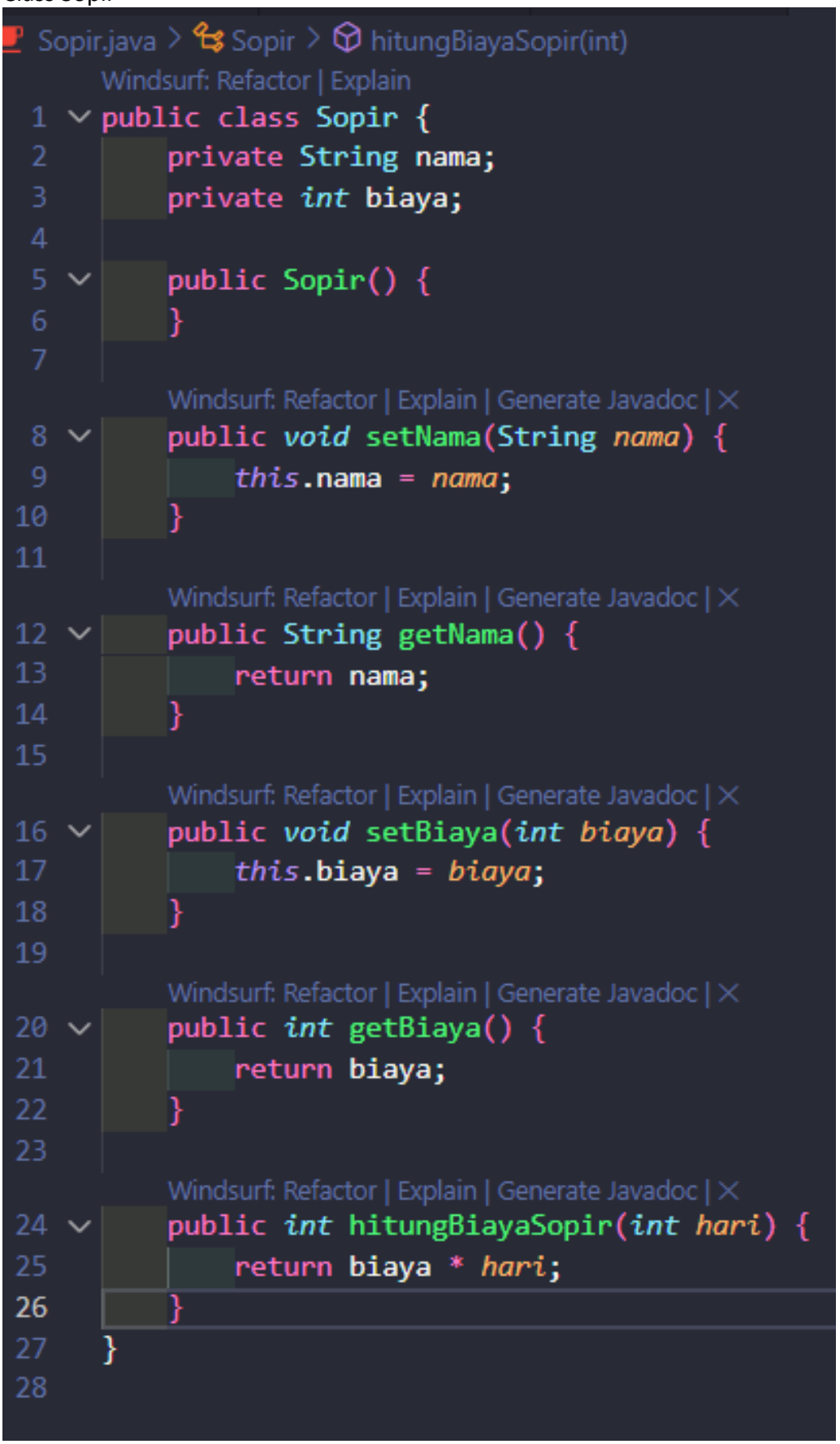
PERCOBAAN 2

VERIFIKASI HASIL PERCOBAAN

Class Mobil

```
Windsurf: Refactor | Explain
public class Mobil {
    private String merk;
    private int biaya;

    public Mobil() {
    }

    Windsurf: Refactor | Explain | Generate Javadoc | ×
    public void setMerk(String merk) {
        this.merk = merk;
    }

    Windsurf: Refactor | Explain | Generate Javadoc | ×
    public String getMerk() {
        return merk;
    }

    Windsurf: Refactor | Explain | Generate Javadoc | ×
    public void setBiaya(int biaya) {
        this.biaya = biaya;
    }

    Windsurf: Refactor | Explain | Generate Javadoc | ×
    public int getBiaya() {
        return biaya;
    }

    Windsurf: Refactor | Explain | Generate Javadoc | ×
    public int hitungBiayaMobil(int hari) {
        return biaya * hari;
    }
}
```

Class Sopir


Sopir.java > Sopir > hitungBiayaSopir(int)
Windsurf: Refactor | Explain
public class Sopir {
private String nama;
private int biaya;
public Sopir() {
}
Windsurf: Refactor | Explain | Generate Javadoc |
public void setNama(String nama) {
this.nama = nama;
}
Windsurf: Refactor | Explain | Generate Javadoc |
public String getNama() {
return nama;
}
Windsurf: Refactor | Explain | Generate Javadoc |
public void setBiaya(int biaya) {
this.biaya = biaya;
}
Windsurf: Refactor | Explain | Generate Javadoc |
public int getBiaya() {
return biaya;
}
Windsurf: Refactor | Explain | Generate Javadoc |
public int hitungBiayaSopir(int hari) {
return biaya * hari;
}
}

Class Pelanggan

```
Pelanggan.java > Pelanggan > hitungBiayaTotal()
Windsurf: Refactor | Explain
public class Pelanggan {
    private String nama;
    private Mobil mobil;
    private Sopir sopir;
    private int hari;

    public Pelanggan() {
    }

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

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

    Windsurf: Refactor | Explain | Generate Javadoc | X
    public void setMobil(Mobil mobil) {
        this.mobil = mobil;
    }

    Windsurf: Refactor | Explain | Generate Javadoc | X
    public Mobil getMobil() {
        return mobil;
    }

    Windsurf: Refactor | Explain | Generate Javadoc | X
    public void setSopir(Sopir sopir) {
        this.sopir = sopir;
    }

    Windsurf: Refactor | Explain | Generate Javadoc | X
    public Sopir getSopir() {
        return sopir;
    }

    Windsurf: Refactor | Explain | Generate Javadoc | X
    public void setHari(int hari) {
        this.hari = hari;
    }

    Windsurf: Refactor | Explain | Generate Javadoc | X
    public int getHari() {
        return hari;
    }

    Windsurf: Refactor | Explain | Generate Javadoc | X
    public int hitungBiayaTotal() {
        int biayaMobil = (mobil != null) ? mobil.hitungBiayaMobil(hari) : 0;
        int biayaSopir = (sopir != null) ? sopir.hitungBiayaSopir(hari) : 0;
        return biayaMobil + biayaSopir;
    }
}
```

Class Main

```
Windsurf: Refactor | Explain
public class MainPercobaan2 {
    Run | Debug | Windsurf: Refactor | Explain | Generate Javadoc | ×
    public static void main(String[] args) {
        Mobil m = new Mobil();
        m.setMerk(merk:"Avanza");
        m.setBiaya(biaya:350000);
        Sopir s = new Sopir();
        s.setNama(nama:"John Doe");
        s.setBiaya(biaya:200000);
        Pelanggan p = new Pelanggan();
        p.setNama(nama:"John Doe");
        p.setMobil(m);
        p.setSopir(s);
        p.setHari(hari:2);
        System.out.println("Biaya Total = " + p.hitungBiayaTotal());
    }
}
```

Output

```
ertemuan4_38a2208b\bin' '
Biaya Total = 1100000
PS D:\KULIAH\Semester 2\P
```

Pertanyaan

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

   baris private Mobil mobil; dan private Sopir sopir; di dalam class Pelanggan adalah bukti utama adanya relasi “has-a” antara Pelanggan dengan Mobil dan Sopir.

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

   hari diperlukan agar method bisa menghitung total biaya sewa berdasarkan lamanya pemakaian, bukan hanya tarif per hari.

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

   Kedua perintah tersebut dipakai untuk mengambil total biaya masing-masing berdasarkan lamanya hari sewa, lalu hasilnya dijumlahkan agar didapat biaya total.

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

   Sintaks p.setMobil(m) dan p.setSopir(s) digunakan untuk mengaitkan objek pelanggan dengan mobil dan sopir yang disewanya sehingga perhitungan biaya total bisa dilakukan.

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

   untuk menghitung dan menampilkan total biaya sewa mobil beserta sopir selama jumlah hari yang ditentukan.

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?

   Sintaks p.getMobil().getMerk() digunakan untuk menampilkan merk mobil yang disewa oleh objek Pelanggan.

PERCOBAAN 3

VERIFIKASI HASIL PERCOBAAN

Class Pegawai

```java
Windsurf: Refactor | Explain
public class Pegawai {
    private String nip;
    private String nama;

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

    Windsurf: Refactor | Explain | Generate Javadoc | ×
    public void setNip(String nip) {
        this.nip = nip;
    }

    Windsurf: Refactor | Explain | Generate Javadoc | ×
    public String getNip() {
        return nip;
    }

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

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

    Windsurf: Refactor | Explain | Generate Javadoc | ×
    public String info() {
        String info = "";
        info += "Nip: " + this.nip + "\n";
        info += "Nama: " + this.nama + "\n";
        return info;
    }
}
```

Class KeretaApi

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

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

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

    Windsurf: Refactor | Explain | Generate Javadoc | ×
    public void setKelas(String kelas) {
        this.kelas = kelas;
    }

    Windsurf: Refactor | Explain | Generate Javadoc | ×
    public String getKelas() {
        return kelas;
    }

    Windsurf: Refactor | Explain | Generate Javadoc | ×
    public void setMasinis(Pegawai masinis) {
        this.masinis = masinis;
    }

    Windsurf: Refactor | Explain | Generate Javadoc | ×
    public Pegawai getMasinis() {
        return masinis;
    }

    Windsurf: Refactor | Explain | Generate Javadoc | ×
    public void setAsisten(Pegawai asisten) {
        this.asisten = asisten;
    }

    Windsurf: Refactor | Explain | Generate Javadoc | ×
    public Pegawai getAsisten() {
        return asisten;
    }

    Windsurf: Refactor | Explain | Generate Javadoc | ×
    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;
    }
}
```

Class Main

```java
Windsurf: Refactor | Explain
public class MainPercobaan3 {
    Run | Debug | Windsurf: Refactor | Explain | Generate Javadoc | ×
    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

```
b\bin' 'MainPercobaan3'
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 ?

   this.masinis.info() dan this.asisten.info() dipakai untuk mengambil informasi lengkap (NIP dan nama) dari pegawai masinis dan asisten melalui method info() milik class Pegawai, lalu ditampilkan sebagai bagian dari informasi kereta.

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());

```java
public class MainPertanyaan {
    Run | Debug | Windsurf: Refactor | Explain | Generate Javadoc | ×
    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 ?

   Program menampilkan error NullPointerException karena konstruktor 3 parameter untuk KeretaApi tidak menginisialisasi atribut asisten

4. Perbaiki class KeretaApi sehingga program dapat berjalan !

```
if (this.asisten != null) {
    info += "Asisten: \n" + this.asisten.info();
} else {
    info += "Asisten: (null)\n";
}
return info;
```

Program akan tetap bisa menampilkan data tetapi bila asisten tidak di isi maka akan memunculkan (null)

PERCOBAAN 4

VERIFIKASI HASIL PERCOBAAN

Class Kursi

```
Windsurf: Refactor | Explain
public class Kursi {
    private String nomor;
    private Penumpang penumpang;

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

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

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

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

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

    Windsurf: Refactor | Explain | Generate Javadoc | ×
    public String info() {
        String info = "";
        info += "Nomor: " + nomor + "\n";
        if (this.penumpang != null) {
            info += "Penumpang: " + penumpang.info() + "\n";
        }
        return info;
    }
}
```

Class Gerbong

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

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

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

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

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

    Windsurf: Refactor | Explain | Generate Javadoc | ×
    public void setPenumpang(Penumpang penumpang, int nomor) {
        this.arrayKursi[nomor - 1].setPenumpang(penumpang);
    }


    Windsurf: Refactor | Explain | Generate Javadoc | ×
    public Kursi[] getArrayKursi() {
        return arrayKursi;
    }

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

Class Penumpang

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

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

    public void setKtp(String ktp) {
        this.ktp = ktp;
    }

    public String getKtp() {
        return ktp;
    }

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

    public String getNama() {
        return nama;
    }

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

Class Main

```java
public class MainPercobaan4 {
    Run | Debug | Windsurf: Refactor | Explain | Generate Javadoc | ×
    public static void main(String[] args) {
        Penumpang p = new Penumpang(ktp:"12345", nama:"Mr. Krab");
        Gerbong gerbong = new Gerbong(kode:"A", jumlah:10);
        gerbong.setPenumpang(p, nomor:1);
        System.out.println(gerbong.info());
    }
}
```

Output

```
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 ?

   Jumlah kursi dalam Gerbong A adalah 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";
   }
   ...
   ```

   Kode tersebut berfungsi untuk menampilkan informasi penumpang hanya jika kursi sudah ditempati; jika kursi masih kosong, tidak akan ditampilkan apa pun.

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

   Nilai nomor dikurangi 1 agar nomor kursi yang dimasukkan user (1-based) bisa sesuai dengan indeks array kursi (0-based) di Java.

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

   Budi dan Mr. Krab sama sama menduduki kursi 1

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

```
Windsurf: Refactor | Explain | Generate Javadoc | ×
public void setPenumpang(Penumpang penumpang, int nomor) {
    if (nomor < 1 || nomor > arrayKursi.length) {
        System.out.println("Nomor kursi tidak valid: " + nomor);
        return;
    }

    Kursi k = this.arrayKursi[nomor - 1];
    if (k.getPenumpang() == null) {
        k.setPenumpang(penumpang);
    } else {
        System.out.println("Kursi nomor " + nomor + " sudah ditempati oleh:");
        System.out.println(k.getPenumpang().info());
    }
}
```