JOBSHEET 4

OBJECT ORIENTED PROGRAMMING

Arranged By :

Lenka Melinda Florienka

2241720074

Class 2I

INFORMATION TECHNOLOGY

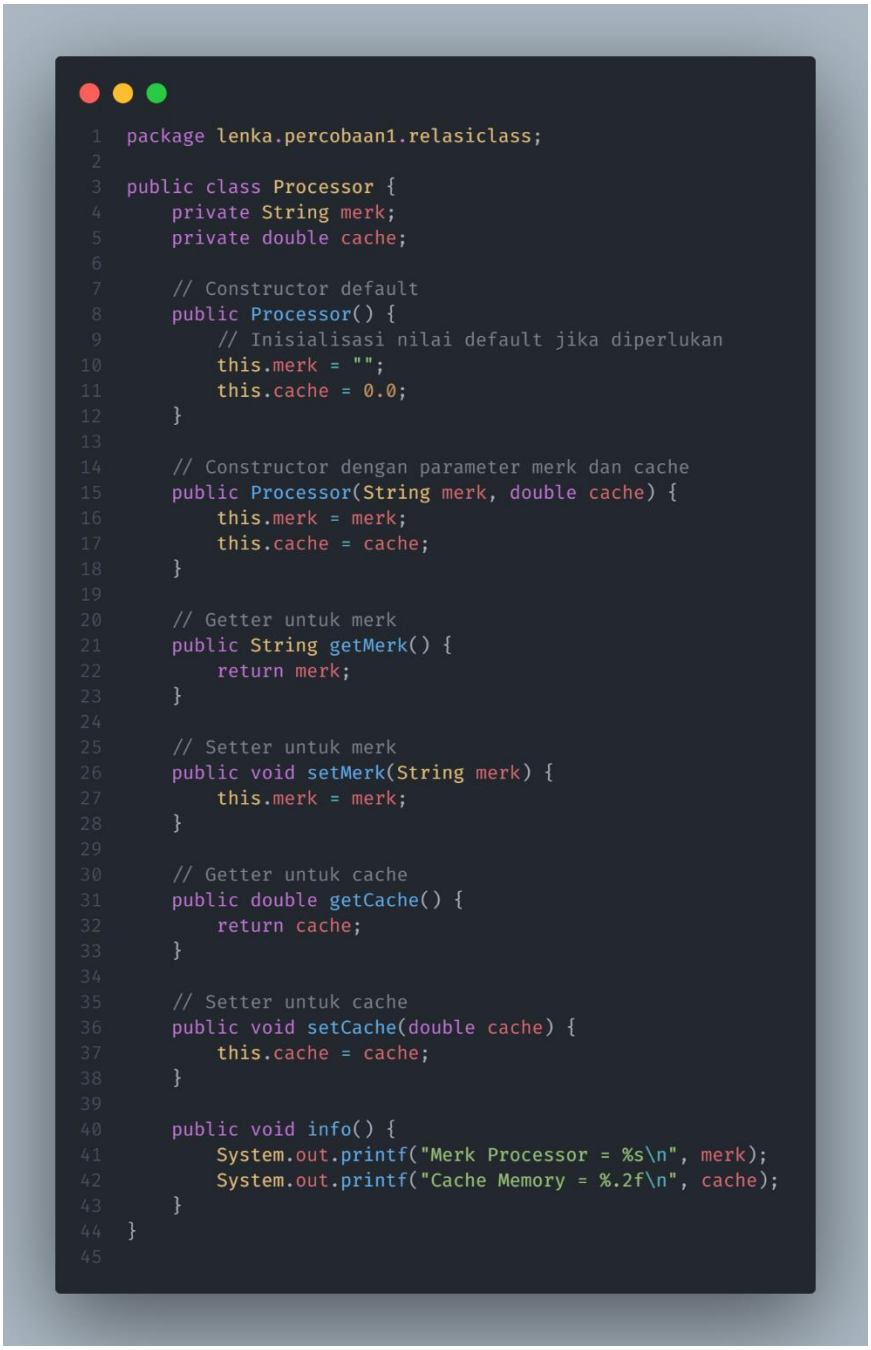
D-IV INFORMATICS ENGINEERING

MALANG STATE POLYTECHNIC

2023

Practicum 1

Code :



```java
package lenka.percobaan1.relasiclass;

public class Processor {
    private String merk;
    private double cache;

    // Constructor default
    public Processor() {
        // Inisialisasi nilai default jika diperlukan
        this.merk = "";
        this.cache = 0.0;
    }

    // Constructor dengan parameter merk dan cache
    public Processor(String merk, double cache) {
        this.merk = merk;
        this.cache = cache;
    }

    // Getter untuk merk
    public String getMerk() {
        return merk;
    }

    // Setter untuk merk
    public void setMerk(String merk) {
        this.merk = merk;
    }

    // Getter untuk cache
    public double getCache() {
        return cache;
    }

    // Setter untuk cache
    public void setCache(double cache) {
        this.cache = cache;
    }

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

```

```java
package lenka.percobaan1.relasiclass;

public class Laptop {
    private String merk;
    private Processor proc;

    // Constructor default
    public Laptop() {

        this.merk = "";
        this.proc = new Processor();
    }

    // Constructor dengan parameter merk dan proc
    public Laptop(String merk, Processor proc) {
        this.merk = merk;
        this.proc = proc;
    }

    // Getter untuk merk
    public String getMerk() {
        return merk;
    }

    // Setter untuk merk
    public void setMerk(String merk) {
        this.merk = merk;
    }

    // Getter untuk Processor
    public Processor getProc() {
        return proc;
    }

    // Setter untuk Processor
    public void setProc(Processor proc) {
        this.proc = proc;
    }

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

```

```java
package lenka.percobaan1.relasiclass;

public class MainPercobaan1 {
    public static void main(String[] args) {
        Processor p = new Processor("Intel i5", 3);

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

        L.info();

        Processor p1 = new Processor();
        p1.setMerk("Intel i5");
        p1.setCache(4);
        Laptop L1 = new Laptop();
        L1.setMerk("Thinkpad");
        L1.setProc(p1);

        L1.info();
    }
}
```

Output :

```
Merk Laptop = Thinkpad
Merk Processor = Intel i5
Cache Memory = 3.00
Merk Laptop = Thinkpad
Merk Processor = Intel i5
Cache Memory = 4.00
PS C:\Dev\OOP>
```

Questions

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

Method setter digunakan untuk mengatur (set) nilai atribut dari luar class, sedangkan method getter digunakan untuk mengambil (get) nilai atribut dari luar class.

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

Konstruktor default digunakan untuk membuat objek dengan nilai-nilai default atau kosong untuk semua atribut objek. Konstruktor berparameter digunakan untuk membuat objek dengan nilai-nilai yang telah ditentukan saat pembuatan objek.

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

Atribut yang bertipe objek dalam class Laptop adalah proc, yang merupakan objek dari class Processor.

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

private Processor proc;

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

Sintaks proc.info() digunakan untuk memanggil method info() dari objek proc yang merupakan objek dari class Processor. Ini memungkinkan kita untuk menampilkan informasi tentang prosesor yang terkait dengan laptop.

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: Page 4 of 10 Laptop l = new Laptop("Thinkpad", new Processor("Intel i5", 3)); Bagaimanakah hasil program saat dijalankan, apakah ada perubahan ?

Pada baris kode Laptop l = new Laptop("Thinkpad", p);, variabel p adalah objek Processor yang telah dibuat sebelumnya dengan nilai "Intel i5" untuk atribut merk dan 3 untuk atribut cache. Jika kode tersebut diubah menjadi Laptop l = new Laptop("Thinkpad", new Processor("Intel i5", 3));, hasil program akan tetap sama. Kedua baris kode tersebut akan membuat objek Laptop dengan merk

"Thinkpad" dan objek Processor dengan merk "Intel i5" dan cache 3, sehingga tidak ada perubahan dalam hasil program saat dijalankan.

Practicum 2

Code :

```java
package lenka.relasiclass.percobaan2;

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());
    }
}
```

```java
package lenka.relasiclass.percobaan2;

public class Pelanggan {
    private String nama;
    private Mobil mobil;
    private Sopir sopir;
    private int hari;

    public Pelanggan() {
    }

    public String getNama() {
        return nama;
    }

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

    public Mobil getMobil() {
        return mobil;
    }

    public void setMobil(Mobil mobil) {
        this.mobil = mobil;
    }

    public Sopir getSopir() {
        return sopir;
    }

    public void setSopir(Sopir sopir) {
        this.sopir = sopir;
    }

    public int getHari() {
        return hari;
    }

    public void setHari(int hari) {
        this.hari = hari;
    }

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

```java
package lenka.relasiclass.percobaan2;

public class Sopir {
    private String nama;
    private int biaya;

    public Sopir() {
    }

    public String getNama() {
        return nama;
    }

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

    public int getBiaya() {
        return biaya;
    }

    public void setBiaya(int biaya) {
        this.biaya = biaya;
    }

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

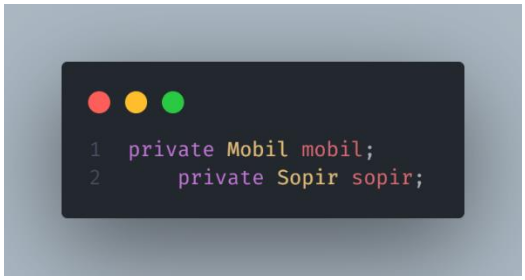
```java
package lenka.relasiclass.percobaan2;

public class Mobil {
    private String merk;
    private int biaya;

    public Mobil() {
    }

    public String getMerk() {
        return merk;
    }

    public void setMerk(String merk) {
        this.merk = merk;
    }

    public int getBiaya() {
        return biaya;
    }

    public void setBiaya(int biaya) {
        this.biaya = biaya;
    }

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

Output :

```
User\workspaceStorage\2abd1bc
Biaya Total = 1100000
PS C:\Dev\OOP>
```

Questions

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



Di sini, kita mendefinisikan dua atribut, mobil dan sopir, yang merupakan objek dari class Mobil dan class Sopir. Ini adalah contoh hubungan asosiasi antara class Pelanggan dengan class Mobil dan class Sopir.

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

Method hitungBiayaSopir pada class Sopir dan method hitungBiayaMobil pada class Mobil memerlukan argumen hari karena biaya sopir dan biaya mobil seringkali bergantung pada berapa hari pelanggan menggunakan layanan tersebut. Dengan menggunakan hari sebagai argumen, Anda dapat menghitung biaya berdasarkan jumlah hari yang dipilih oleh pelanggan. Hal ini memungkinkan fleksibilitas dalam perhitungan biaya.

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

Perintah mobil.hitungBiayaMobil(hari) dan sopir.hitungBiayaSopir(hari) pada class Pelanggan digunakan untuk menghitung biaya mobil dan biaya sopir berdasarkan jumlah hari yang diberikan (parameter hari). Ini adalah bagian dari perhitungan total biaya dalam method hitungBiayaTotal di class Pelanggan.

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

Sintaks p.setMobil(m) dan p.setSopir(s) digunakan untuk mengatur objek Mobil m dan objek Sopir s ke dalam objek Pelanggan p. Ini berarti bahwa pelanggan p memiliki mobil m dan sopir s.

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

Proses p.hitungBiayaTotal() menghitung total biaya yang harus dibayar oleh pelanggan. Ini melibatkan penghitungan biaya mobil (mobil.hitungBiayaMobil(hari)) dan biaya sopir (sopir.hitungBiayaSopir(hari)) berdasarkan jumlah hari yang diberikan (hari). Total biaya adalah hasil dari penjumlahan biaya mobil dan biaya sopir.

6. Perhatikan class MainPercobaan2, coba tambahkan pada baris terakhir dari method main dan amati perubahan saat di-run! Page 6 of 10 System.out.println(p.getMobil().getMerk());

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

Sintaks System.out.println(p.getMobil().getMerk()) dalam method main digunakan untuk mencetak merk mobil yang dimiliki oleh pelanggan p. Ini memperlihatkan cara mengakses atribut objek mobil yang dimiliki oleh pelanggan dan mendapatkan nilai dari atribut merk pada objek mobil tersebut.

Practicum 3

Code :

```java
package lenka.relasiclass.percobaan3;

public class MainPercobaan3 {
    public static void main(String[] args) {
        Pegawai masinis = new Pegawai("1234", "Spongebob Squarepants");
        Pegawai asisten = new Pegawai("4567", "Patrick Star");
        KeretaApi keretaApi = new KeretaApi("Gaya Baru", "Bisnis", masinis, asisten);
        System.out.println(keretaApi.info());
    }
}
```

```java
package lenka.relasiclass.percobaan3;

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;
    }

    public String getNama() {
        return nama;
    }

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

    public String getKelas() {
        return kelas;
    }

    public void setKelas(String kelas) {
        this.kelas = kelas;
    }

    public Pegawai getMasinis() {
        return masinis;
    }

    public void setMasinis(Pegawai masinis) {
        this.masinis = masinis;
    }

    public Pegawai getAsisten() {
        return asisten;
    }

    public void setAsisten(Pegawai asisten) {
        this.asisten = asisten;
    }

    public String info() {
        String info = "";
        info += "Nama: " + this.nama + "\n";
        info += "Kelas: " + this.kelas + "\n";
        info += "Masinis: " + this.masinis.info() + "\n";
        if (asisten != null) {
            info += "Asisten: " + this.asisten.info() + "\n";
        }
        return info;
    }
}
```

```java
package lenka.relasiclass.percobaan3;

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

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

    public String getNip() {
        return nip;
    }

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

    public String getNama() {
        return nama;
    }

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

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

Output :

```
User\workspaceStorage\2abd1bcf84
Nama: Gaya Baru
Kelas: Bisnis
Masinis: Nip: 1234
Nama: Spongebob Squarepants

Asisten: Nip: 4567
Nama: Patrick Star


PS C:\Dev\OOP>
```

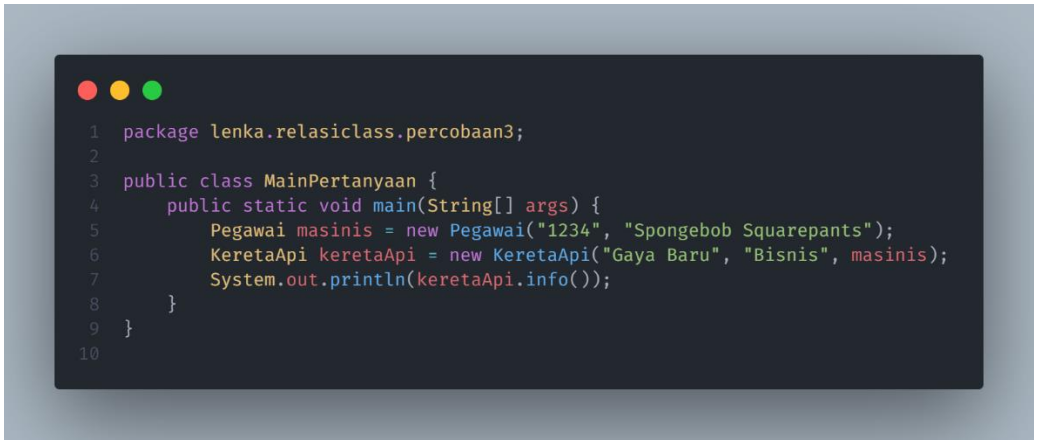
Questions

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

Di dalam method info() pada class KeretaApi, baris this.masinis.info() dan this.asisten.info() digunakan untuk mengakses method info() dari objek Pegawai yang merupakan atribut dari class

KeretaApi. Ini memungkinkan Anda untuk mendapatkan informasi tentang masinis dan asisten dari kereta api.

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
package lenka.relasiclass.percobaan3;

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

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

Output dari main program tersebut akan menghasilkan informasi tentang kereta api yang hanya mencakup nama, kelas, dan masinis. Hal ini terjadi karena Anda hanya melewatkan objek Pegawai masinis sebagai parameter ketika membuat objek KeretaApi. Anda tidak menyertakan asisten dalam objek KeretaApi, sehingga hanya informasi masinis yang ditampilkan.

4. Perbaiki class KeretaApi sehingga program dapat berjalan !

```java
package lenka.relasiclass.percobaan3;

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;
    }

    public String getNama() {
        return nama;
    }

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

    public String getKelas() {
        return kelas;
    }

    public void setKelas(String kelas) {
        this.kelas = kelas;
    }

    public Pegawai getMasinis() {
        return masinis;
    }

    public void setMasinis(Pegawai masinis) {
        this.masinis = masinis;
    }

    public Pegawai getAsisten() {
        return asisten;
    }

    public void setAsisten(Pegawai asisten) {
        this.asisten = asisten;
    }

    public String info() {
        String info = "";
        info += "Nama: " + this.nama + "\n";
        info += "Kelas: " + this.kelas + "\n";
        info += "Masinis: " + this.masinis.info() + "\n";
        if (asisten != null) {
            info += "Asisten: " + this.asisten.info() + "\n";
        }
        return info;
    }
}
```

Practicum 4

Code :

```java
package lenka.relasiclass.percobaan4;

public class MainPercobaan4 {
    public static void main(String[] args) {
        Penumpang p = new Penumpang("12345", "Mr. Krab");
        Gerbong gerbong = new Gerbong("A", 10);
        gerbong.setPenumpang(p, 1);
        System.out.println(gerbong.info());
    }
}
```

```java
package lenka.relasiclass.percobaan4;

class Penumpang {
    private String ktp;
    private String nama;

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

    public String info() {
        return "KTP: " + ktp + ", Nama: " + nama;
    }
}
```

```java
package lenka.relasiclass.percobaan4;

class Gerbong {
    private String kode;
    private Kursi[] arrayKursi;

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

    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 = "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);
    }
}
```

```java
package lenka.relasiclass.percobaan4;

class Kursi {
    private String noKursi;
    private Penumpang penumpang;

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

    public void setPenumpang(Penumpang penumpang) {
        this.penumpang = penumpang;
    }

    public Penumpang getPenumpang() {
        return penumpang;
    }

    public String info() {
        if (penumpang != null) {
            return "Nomor Kursi: " + noKursi + ", Penumpang: " + penumpang.info() + "\n";
        } else {
            return "Nomor Kursi: " + noKursi + ", Tidak ada penumpang\n";
        }
    }
}

```

Output :

```
Kode: A
Nomor Kursi: 1, Penumpang: KTP: 12345, Nama: Mr. Krab
Nomor Kursi: 2, Tidak ada penumpang
Nomor Kursi: 3, Tidak ada penumpang
Nomor Kursi: 4, Tidak ada penumpang
Nomor Kursi: 5, Tidak ada penumpang
Nomor Kursi: 6, Tidak ada penumpang
Nomor Kursi: 7, Tidak ada penumpang
Nomor Kursi: 8, Tidak ada penumpang
Nomor Kursi: 9, Tidak ada penumpang
Nomor Kursi: 10, Tidak ada penumpang
```

Questions

1. Pada main program dalam class MainPercobaan4, berapakah jumlah kursi dalam Gerbong A ?
   Jumlah kursi dalam Gerbong A adalah 10. Ini ditentukan ketika objek Gerbong dibuat dengan gerbong = new Gerbong("A", 10); dalam method main.

2. Perhatikan potongan kode pada method info() dalam class Kursi. Apa maksud kode tersebut ? ... if (this.penumpang != null) { info += "Penumpang: " + penumpang.info() + "\n"; }

Pada potongan kode dalam method info() di class Kursi, kode tersebut digunakan untuk mengecek apakah kursi sudah diduduki oleh seorang penumpang. Jika this.penumpang tidak null (artinya ada penumpang di kursi tersebut), maka informasi penumpang ditambahkan ke string info.

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

Pada method setPenumpang() dalam class Gerbong, nilai nomor dikurangi 1 karena dalam array, indeks dimulai dari 0, sedangkan dalam pemberian nomor kursi dimulai dari 1. Oleh karena itu, kita perlu mengonversi nomor kursi dari yang dimasukkan oleh pengguna (dimulai dari 1) ke indeks array (dimulai dari 0) untuk mengakses kursi yang sesuai dalam array arrayKursi.

4. Instansiasi objek baru budi dengan tipe Penumpang, kemudian masukkan objek baru tersebut pada gerbong dengan gerbong.setPenumpang(budi, 1). Apakah yang Page 10 of 10 terjadi ?

Jika Anda mencoba memasukkan objek budi ke kursi pertama (nomor 1) dalam gerbong yang telah diisi oleh objek p sebelumnya, maka budi akan menggantikan p di kursi tersebut. Artinya, objek p sebelumnya akan dihapus dari kursi nomor 1 dan digantikan oleh budi. Jadi, gerbong.setPenumpang(budi, 1) akan menghasilkan perubahan di kursi nomor 1.

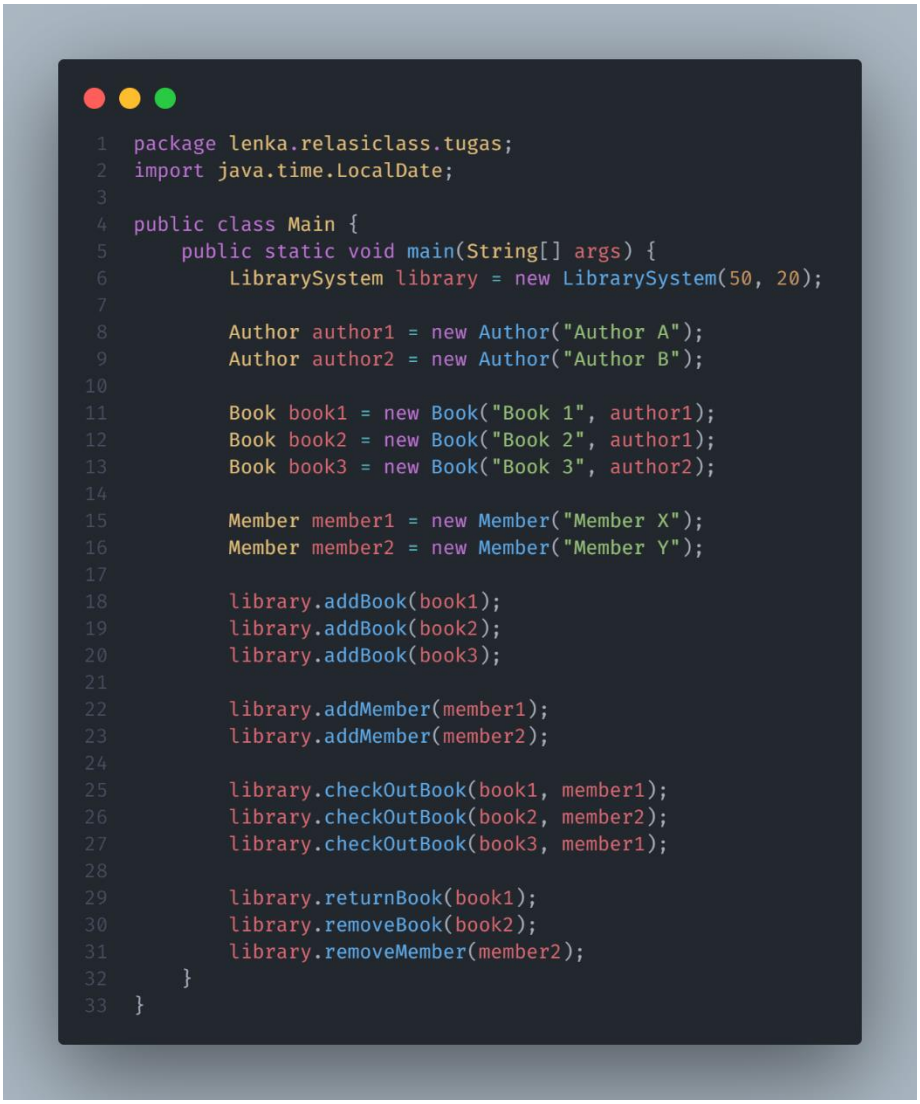
5. Modifikasi program sehingga tidak diperkenankan untuk menduduki kursi yang sudah ada penumpang lain !

Untuk memastikan bahwa kursi yang sudah diduduki oleh penumpang lain tidak dapat diduduki kembali, Anda perlu memodifikasi method setPenumpang() di class Gerbong untuk melakukan pemeriksaan sebelum mengatur penumpang pada kursi. Kita dapat memodifikasinya sebagai berikut:

```
public void setPenumpang(Penumpang penumpang, int nomor) {
    if (this.arrayKursi[nomor - 1].getPenumpang() == null) {
        this.arrayKursi[nomor - 1].setPenumpang(penumpang);
    } else {
        System.out.println("Kursi nomor " + nomor + " sudah diduduki.");
    }
}
```

Tasks

Code :

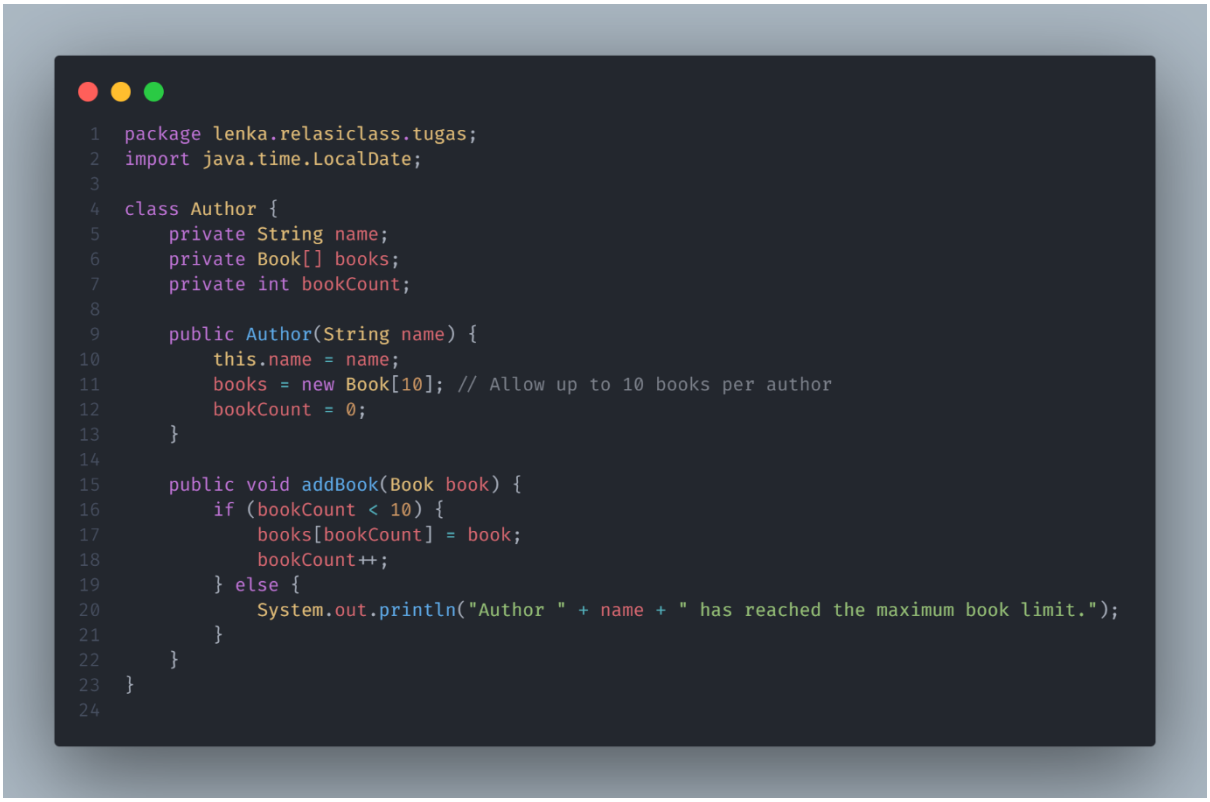


```java
package lenka.relasiclass.tugas;
import java.time.LocalDate;

public class Main {
    public static void main(String[] args) {
        LibrarySystem library = new LibrarySystem(50, 20);

        Author author1 = new Author("Author A");
        Author author2 = new Author("Author B");

        Book book1 = new Book("Book 1", author1);
        Book book2 = new Book("Book 2", author1);
        Book book3 = new Book("Book 3", author2);

        Member member1 = new Member("Member X");
        Member member2 = new Member("Member Y");

        library.addBook(book1);
        library.addBook(book2);
        library.addBook(book3);

        library.addMember(member1);
        library.addMember(member2);

        library.checkOutBook(book1, member1);
        library.checkOutBook(book2, member2);
        library.checkOutBook(book3, member1);

        library.returnBook(book1);
        library.removeBook(book2);
        library.removeMember(member2);
    }
}
```

```java
package lenka.relasiclass.tugas;
import java.time.LocalDate;

class LibrarySystem {
    private Book[] books;
    private Member[] members;
    private Loan[] loans;
    private int bookCount;
    private int memberCount;
    private int loanCount;

    public LibrarySystem(int maxBooks, int maxMembers) {
        books = new Book[maxBooks];
        members = new Member[maxMembers];
        loans = new Loan[100]; // Allow up to 100 loans
        bookCount = 0;
        memberCount = 0;
        loanCount = 0;
    }

    public void addBook(Book book) {
        if (bookCount < books.length) {
            books[bookCount] = book;
            bookCount++;
        } else {
            System.out.println("The library has reached the maximum book limit.");
        }
    }

    public void removeBook(Book book) {
        for (int i = 0; i < bookCount; i++) {
            if (books[i] == book) {
                for (int j = i; j < bookCount - 1; j++) {
                    books[j] = books[j + 1];
                }
                books[bookCount - 1] = null;
                bookCount--;
                book.returnBook();
                System.out.println("Book '" + book.getTitle() + "' removed from the library.");
                return;
            }
        }
        System.out.println("Book '" + book.getTitle() + "' not found in the library.");
    }

    public void addMember(Member member) {
        if (memberCount < members.length) {
            members[memberCount] = member;
            memberCount++;
            System.out.println("Member " + member.getName() + " added to the library.");
        } else {
            System.out.println("The library has reached the maximum member limit.");
        }
    }

    public void removeMember(Member member) {
        for (int i = 0; i < memberCount; i++) {
            if (members[i] == member) {
                for (int j = i; j < memberCount - 1; j++) {
                    members[j] = members[j + 1];
                }
                members[memberCount - 1] = null;
                memberCount--;
                System.out.println("Member " + member.getName() + " removed from the library.");
                return;
            }
        }
        System.out.println("Member " + member.getName() + " not found in the library.");
    }

    public void checkOutBook(Book book, Member member) {
        if (book.isAvailable()) {
            Loan loan = new Loan(book, member);
            loans[loanCount] = loan;
            loanCount++;
            book.borrow(member);
        } else {
            System.out.println("Book '" + book.getTitle() + "' is not available for borrowing.");
        }
    }

    public void returnBook(Book book) {
        if (book == null) {
            System.out.println("Invalid book.");
            return;
        }

        for (int i = 0; i < loanCount; i++) {
            if (loans[i] != null && loans[i].getBook() == book) {
                loans[i] = null;
                loanCount--;
                book.returnBook();
                System.out.println("Book '" + book.getTitle() + "' returned to the library.");
                return;
            }
        }
        System.out.println("Book '" + book.getTitle() + "' was not borrowed from this library.");
    }
}

```

```java
package lenka.relasiclass.tugas;

class Member {
    private String name;

    public Member(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }
}
```

```java
package lenka.relasiclass.tugas;
import java.time.LocalDate;

class Loan {
    private Book book;
    private Member member;
    private LocalDate dueDate;

    public Loan(Book book, Member member) {
        this.book = book;
        this.member = member;
        this.dueDate = LocalDate.now().plusDays(14); // Assume 14 days loan period
    }

    public Book getBook() {
        return book;
    }
}
```



```java
package lenka.relasiclass.tugas;
import java.time.LocalDate;

class Author {
    private String name;
    private Book[] books;
    private int bookCount;

    public Author(String name) {
        this.name = name;
        books = new Book[10]; // Allow up to 10 books per author
        bookCount = 0;
    }

    public void addBook(Book book) {
        if (bookCount < 10) {
            books[bookCount] = book;
            bookCount++;
        } else {
            System.out.println("Author " + name + " has reached the maximum book limit.");
        }
    }
}

```

```java
package lenka.relasiclass.tugas;

class Book {
    private String title;
    private Author author;
    private boolean isAvailable;

    public Book(String title, Author author) {
        this.title = title;
        this.author = author;
        this.isAvailable = true;
        author.addBook(this);
    }

    public void borrow(Member member) {
        if (isAvailable) {
            isAvailable = false;
            System.out.println("Book '" + title + "' borrowed by " + member.getName());
        } else {
            System.out.println("Book '" + title + "' is not available for borrowing.");
        }
    }

    public void returnBook() {
        isAvailable = true;
        System.out.println("Book '" + title + "' returned to the library.");
    }

    public String getTitle() {
        return title;
    }

    public boolean isAvailable() {
        return isAvailable;
    }
}
```

Output :

```
Member Member X added to the library.
Member Member Y added to the library.
Book 'Book 1' borrowed by Member X
Book 'Book 2' borrowed by Member Y
Book 'Book 3' borrowed by Member X
Book 'Book 1' returned to the library.
Book 'Book 1' returned to the library.
Book 'Book 2' returned to the library.
Book 'Book 2' removed from the library.
Member Member Y removed from the library.
PS C:\Dev\OOP>
```