

Jobsheet 04 - Relasi Kelas

I. Kompetensi

Setelah menempuh pokok bahasan ini, mahasiswa mampu:

1. Memahami konsep relasi kelas;
2. Mengimplementasikan relasi asosiasi ke dalam program.

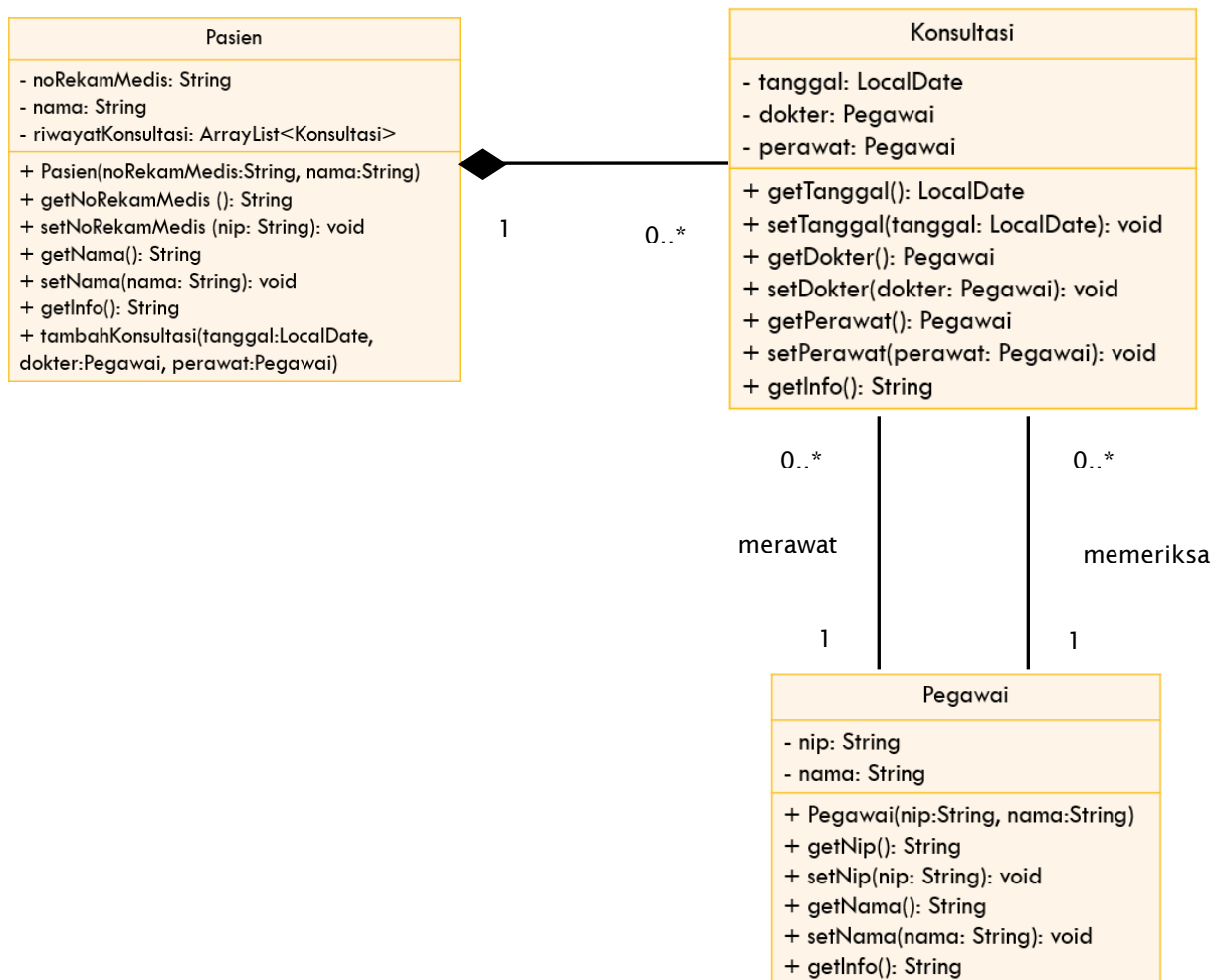
II. Pendahuluan

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 berkaitan.

III. Praktikum

Pada praktikum ini akan dikembangkan suatu sistem informasi rumah sakit yang menyimpan data riwayat konsultasi pasien.

Perhatikan diagram *class* berikut:



a. Buatlah folder baru dengan nama RumahSakit

b. Buatlah class Pegawai. Tambahkan atribut nip dan nama pada class Pegawai dengan akses modifier private

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

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

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

d. Implementasikan **setter** dan **getter** untuk class Pegawai.

```
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;  
}
```

e. Implementasikan *method* getInfo() sebagai berikut:

```
public String getInfo() {  
    return nama + " (" + nip + ")";  
}
```

f. Selanjutnya buatlah class Pasien kemudian tambahkan atribut noRekamMedis dan nama pada class Pasien dengan access level modifier private. Sediakan pula setter dan getter untuk kedua atribut tersebut.

```

public class Pasien {
    private String noRekamMedis;
    private String nama;

    public String getNoRekamMedis() {
        return noRekamMedis;
    }

    public void setNoRekamMedis(String noRekamMedis) {
        this.noRekamMedis = noRekamMedis;
    }

    public String getNama() {
        return nama;
    }

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

```

- g. Buatlah constructor untuk class Pasien dengan parameter noRekamMedis dan nama

```

public Pasien(String noRekamMedis, String nama) {
    this.noRekamMedis = noRekamMedis;
    this.nama = nama;
}

```

- h. Implementasikan *method* getInfo() sebagai berikut:

```

public String getInfo() {
    String info = "";
    info += "No Rekam Medis      : " + this.noRekamMedis + "\n";
    info += "Nama                  : " + this.nama + "\n";
    info += "\n";

    return info;
}

```

- i. Sistem ini akan menyimpan data setiap konsultasi yang dilakukan pasien. Pasien bisa melakukan konsultasi lebih dari sekali. Oleh karena itu, data konsultasi akan disimpan dalam bentuk ArrayList dari objek-objek yang bertipe Konsultasi.
- j. Buatlah class dengan nama Konsultasi dengan atribut tanggal bertipe LocalDate, dokter bertipe Pegawai, dan perawat bertipe Pegawai. Set access level modifier private untuk seluruh atribut. Lakukan import `java.time.LocalDate` agar dapat mendeklarasikan atribut tanggal bertipe LocalDate.

```
import java.time.LocalDate;

public class Konsultasi {
    private LocalDate tanggal;
    private Pegawai dokter;
    private Pegawai perawat;
}
```

- k. Sediakan setter dan getter untuk masing-masing atribut pada class Konsultasi

```
public LocalDate getTanggal() {
    return tanggal;
}

public void setTanggal(LocalDate tanggal) {
    this.tanggal = tanggal;
}

public Pegawai getDokter() {
    return dokter;
}

public void setDokter(Pegawai dokter) {
    this.dokter = dokter;
}

public Pegawai getPerawat() {
    return perawat;
}

public void setPerawat(Pegawai perawat) {
    this.perawat = perawat;
}
```

- l. Implementasikan method getInfo() sebagai berikut:

```
public String getInfo() {
    String info = "";
    info += "\tTanggal: " + tanggal;
    info += ", Dokter: " + dokter.getInfo();
    info += ", Perawat: " + perawat.getInfo();
    info += "\n";

    return info;
}
```

- m. Untuk menyimpan data riwayat konsultasi pasien, maka tambahkan atribut riwayatKonsultasi pada class Pasien dengan tipe arrayList<Konsultasi>. Atribut ini akan menyimpan serangkaian objek bertipe Konsultasi. Import java.util.ArrayList agar dapat mendeklarasikan atribut bertipe ArrayList of object.

```
private String noRekamMedis;
private String nama;
private ArrayList<Konsultasi> riwayatKonsultasi;
```

- n. Buatlah constructor berparameter untuk class Pasien. Inisiasi nilai atribut noRekamMedis dan nama berdasarkan atribut nama. Instansiasi/buat ArrayList baru untuk atribut riwayatKonsultasi;

```
public Pasien(String noRekamMedis, String nama) {
    this.noRekamMedis = noRekamMedis;
    this.nama = nama;
    this.riwayatKonsultasi = new ArrayList<Konsultasi>();
}
```

- o. Lakukan import java.time.LocalDate agar dapat mendeklarasikan atribut tanggal bertipe LocalDate pada class Pasien. Selanjutnya, implementasikan method tambahKonsultasi() sebagai berikut:

```
public void tambahKonsultasi(LocalDate tanggal, Pegawai dokter, Pegawai perawat){
    Konsultasi konsultasi = new Konsultasi();
    konsultasi.setTanggal(tanggal);
    konsultasi.setDokter(dokter);
    konsultasi.setPerawat(perawat);
    riwayatKonsultasi.add(konsultasi);
}
```

- p. Modifikasi method getInfo() untuk mengembalikan info pasien dan daftar konsultasi yang pernah dilakukan

```
public String getInfo(){
    String info = "";
    info += "No Rekam Medis      : " + this.noRekamMedis + "\n";
    info += "Nama                  : " + this.nama + "\n";

    if (!riwayatKonsultasi.isEmpty()) {
        info += "Riwayat Konsultasi :\n";

        for (Konsultasi konsultasi : riwayatKonsultasi) {
            info += konsultasi.getInfo();
        }
    }
    else{
        info += "Belum ada riwayat konsultasi";
    }

    info += "\n";

    return info;
}
```

- q. Lakukan import `java.time.LocalDate` agar dapat mendeklarasikan atribut tanggal bertipe `LocalDate` pada class `RumahSakitDemo`. Test program yang sudah dibuat dengan membuat objek-objek pada class `RumahSakitDemo`. Instansiasi objek baru bertipe `Pegawai` dengan nama ani menggunakan constructor `Pegawai(String nip, String nama)` dengan nilai argumen nip "1234" dan nama "dr. Ani". Lanjutkan instansiasi objek sebagai berikut:

```
import java.time.LocalDate;

public class RumahSakitDemo {
    Run | Debug
    public static void main(String[] args) {
        Pegawai ani = new Pegawai("1234", "dr. Ani");
        Pegawai bagus = new Pegawai("4567", "dr. Bagus");

        Pegawai desi = new Pegawai("1234", "Ns. Desi");
        Pegawai eka = new Pegawai("4567", "Ns. Eka");

        Pasien pasien1 = new Pasien("343298", "Puspa Widya");
        pasien1.tambahKonsultasi(LocalDate.of(2021, 8, 11), ani, desi);
        pasien1.tambahKonsultasi(LocalDate.of(2021, 9, 11), bagus, eka);

        System.out.println(pasien1.getInfo());

        Pasien pasien2 = new Pasien("997744", "Yenny Anggraeni");
        System.out.println(pasien2.getInfo());
    }
}
```

- r. *Compile* kemudian *run* `RumahSakitDemo` dan didapatkan hasil seperti berikut:

```
No Rekam Medis      : Puspa Widya
Nama                : 343298
Riwayat Konsultasi :
    Tanggal: 2021-08-11, Dokter: dr. Ani (1234), Perawat: Ns. Desi (1234)
    Tanggal: 2021-09-11, Dokter: dr. Bagus (4567), Perawat: Ns. Eka (4567)

No Rekam Medis      : Yenny Anggraeni
Nama                : 997744
Belum ada riwayat konsultasi
```

Pertanyaan

Berdasarkan percobaan 1, jawablah pertanyaan-pertanyaan yang terkait:

1. Di dalam *class* `Pegawai`, `Pasien`, dan `Konsultasi`, terdapat method *setter* dan *getter* untuk masing-masing atributnya. Apakah gunanya *method setter* dan *getter* tersebut ?
2. Di dalam *class* `Konsultasi` tidak secara eksplisit terdapat constructor dengan parameter. Apakah ini berarti *class* `Konsultasi` tidak memiliki constructor?
3. Perhatikan *class* `Konsultasi`, atribut mana saja yang bertipe *object*?

4. Perhatikan *class* Konsultasi, pada baris manakah yang menunjukkan bahwa *class* Konsultasi memiliki relasi dengan *class* Pegawai?
5. Perhatikan pada *class* Pasien, apa yang dilakukan oleh kode `konsultasi.getInfo()`?
6. Pada method `getInfo()` dalam *class* Pasien, terdapat baris kode:
`if (!riwayatKonsultasi.isEmpty())`
Apakah yang dilakukan oleh baris tersebut?
7. Pada constructor *class* Pasien, terdapat baris kode:
`this.riwayatKonsultasi = new ArrayList<>();`
Apakah yang dilakukan oleh baris tersebut? Apakah yang terjadi jika baris tersebut dihilangkan?

IV. Tugas

Implementasikan studi kasus yang telah dibuat pada tugas PBO Teori ke dalam program