Jobsheet 04 - Class Relations

I. Competence

After studying this subject, students are able to:

1. Understand the concept of class relations;
2. Implement association relations into the program.

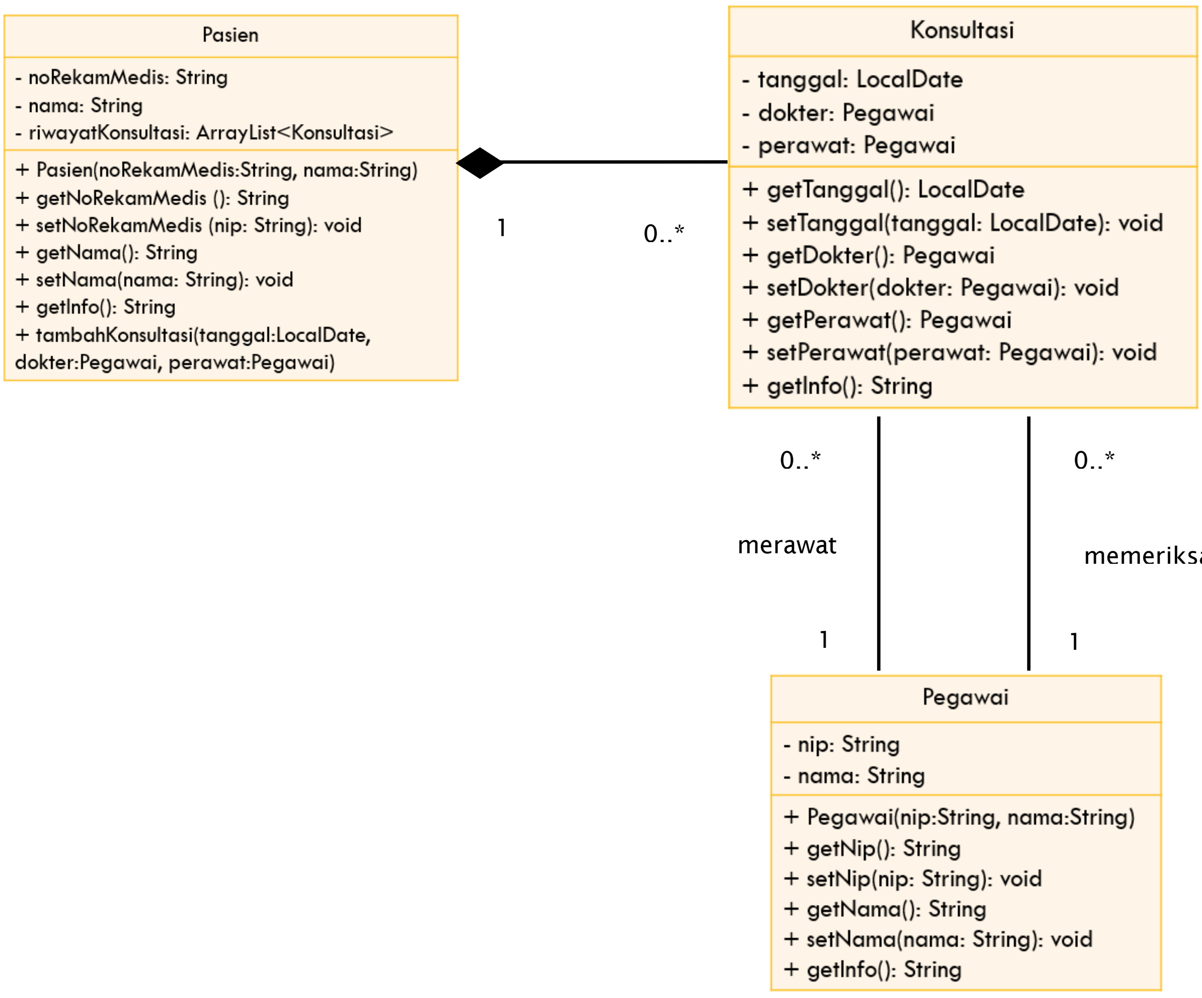
II. Introduction

In more complex cases, in a system there will be more than one *class* that is related to each other. In previous experiments, the majority of cases that have been worked on have only focused on one *class*. In this jobsheet, an experiment will be carried out involving several classes that are related to each other.

III. Practicum

In this practicum, a hospital information system will be developed that stores patient consultation history data.

Consider the following class diagram :



Pasien
- noRekamMedis: String
- nama: String
- riwayatKonsultasi: ArrayList<Konsultasi>
+ Pasien(noRekamMedis:String, nama:String)
+ getNoRekamMedis (): String
+ setNoRekamMedis (nip: String): void
+ getNama(): String
+ setNama(nama: String): void
+ getInfo(): String
+ tambahKonsultasi(tanggal:LocalDate, dokter:Pegawai, perawat:Pegawai)
1
0..*
Konsultasi
- tanggal: LocalDate
- dokter: Pegawai
- perawat: Pegawai
+ getTanggal(): LocalDate
+ setTanggal(tanggal: LocalDate): void
+ getDokter(): Pegawai
+ setDokter(dokter: Pegawai): void
+ getPerawat(): Pegawai
+ setPerawat(perawat: Pegawai): void
+ getInfo(): String
0..*
merawat
1
0..*
memeriksa
1
Pegawai
- nip: String
- nama: String
+ Pegawai(nip:String, nama:String)
+ getNip(): String
+ setNip(nip: String): void
+ getNama(): String
+ setNama(nama: String): void
+ getInfo(): String

a. Create a new folder with the name Hospital.

b. Create an Employee class. Add nip and name attributes to Employee class with private modifier access

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

c. Create a *constructor* for the Officer class with the nip and name parameters.

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

d. Implement **setters** and **getters** for the Employee class.

```
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. Implement *the getInfo()* method as follows:

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

f. Next, create a Patient class then add the noReReRecordMedical attribute and name to the Patient class with a private access level modifier. Also provide setters and getters for these two attributes.

```java
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. Create a constructor for the Patient class with the parameter noReReMedical , and the name

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

h. Implement *the getInfo()* method as follows:

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

    return info;
}
```

i. This system will store data on every consultation that the patient conducts. Patients can have a consultation more than once. Therefore, the consultation data will be stored in the form of an ArrayList of objects of type Consultation.

j. Create a class called Consultation with date attributes of type LocalDate, doctor type employee, and nurse type employee. Set private access level modifiers for all attributes. Import `java.time.LocalDate` to declare a date attribute of type LocalDate.

```java
import java.time.LocalDate;

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

k. Provide setters and getters for each attribute in the Consult class

```java
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. Implement the getInfo() method as follows:

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

    return info;
}
```

m. To store patient consultation history data, add the Consultation history attribute to the Patient class with the arrayList<Consultation> type. This attribute will store a series of objects of type Consultation. Import java.util.ArrayList in order to declare an attribute of type ArrayList of object.

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

n. Create a parameterized constructor for the Patient class. Initiation of the value of the noReRecordMedical attribute and the name based on the name attribute. Instantiate/create a new ArrayList for the Consultation history attribute;

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

o. Import java.time.LocalDate to declare a date attribute of type LocalDate in the Patient class. Next, implement the method addConsultation() as follows:

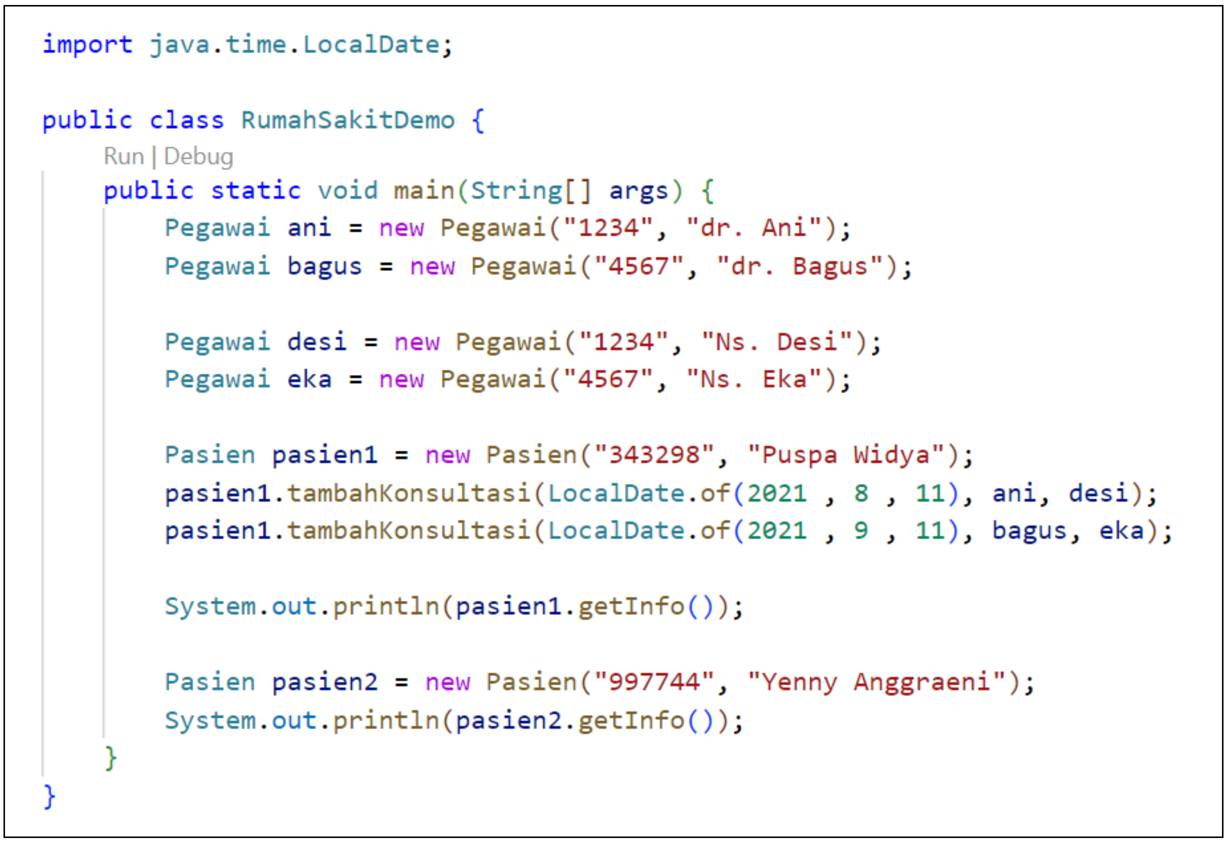
```java
public void tambahKonsultasi(LocalDate tanggal, Pegawai dokter, Pegawai perawat){
    Konsultasi konsultasi = new Konsultasi();
    konsultasi.setTanggal(tanggal);
    konsultasi.setDokter(dokter);
    konsultasi.setPerawat(perawat);
    riwayatKonsultasi.add(konsultasi);
}
```

p. Modify the getInfo() method to return patient info and a list of consultations that have been done

```java
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;
}
```

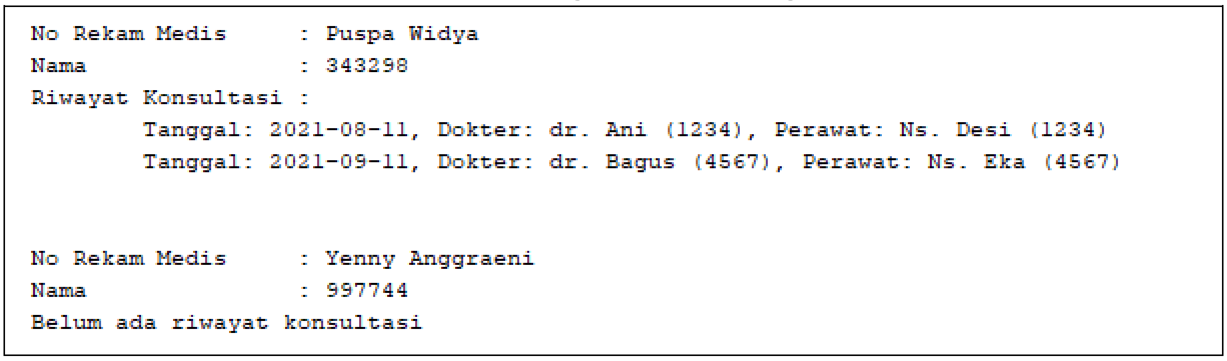
q. Import java.time.LocalDate in order to declare a date attribute of type LocalDate in the

HospitalDemo class. Test the program that has been created by creating objects in the RumahSakit Demo class. The new object instance of type Employee with the name ani uses the Employee constructor (String nip, String name) with the value of the argument nip "1234" and the name "dr. Ani". Continue the object instantiation as follows:



```java
import java.time.LocalDate;

public class RumahSakitDemo {
    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* then *run* RumahSakitDemo and get the following results:



```
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
```

Question

Based on experiment 1, answer the related questions:

1. In the *Employee, Patient,* and Consultation classes, there are method *setters* and *getters* for each of their attributes. What is the use of *the setter and getter* methods ?
2. In the *Consult* class there is not explicitly a constructor with parameters. Does this mean that the Consult class doesn't have a constructor?
3. Notice the *Consult* class, which attributes are of type *object*?
4. Pay attention to *the Consultation* class, on which line does it show that the

Consultation class has a relationship with the *Employee* class?
5. Notice in the *Patient* class, what does the `consultation code.getInfo()` do?
6. In the `getInfo() method` in the Patient class, there is a line of code:
   `if (!historyConsultation.isEmpty())`
   What does the line do?
7. In the Patient constructor class, there is a line of code:
   `this.historyConsultation = new ArrayList<>();`
   What does the line do? What happens if the line is omitted?

IV. Assignment

Implement the case studies that have been made on the Theory PBO assignment into the program.