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.

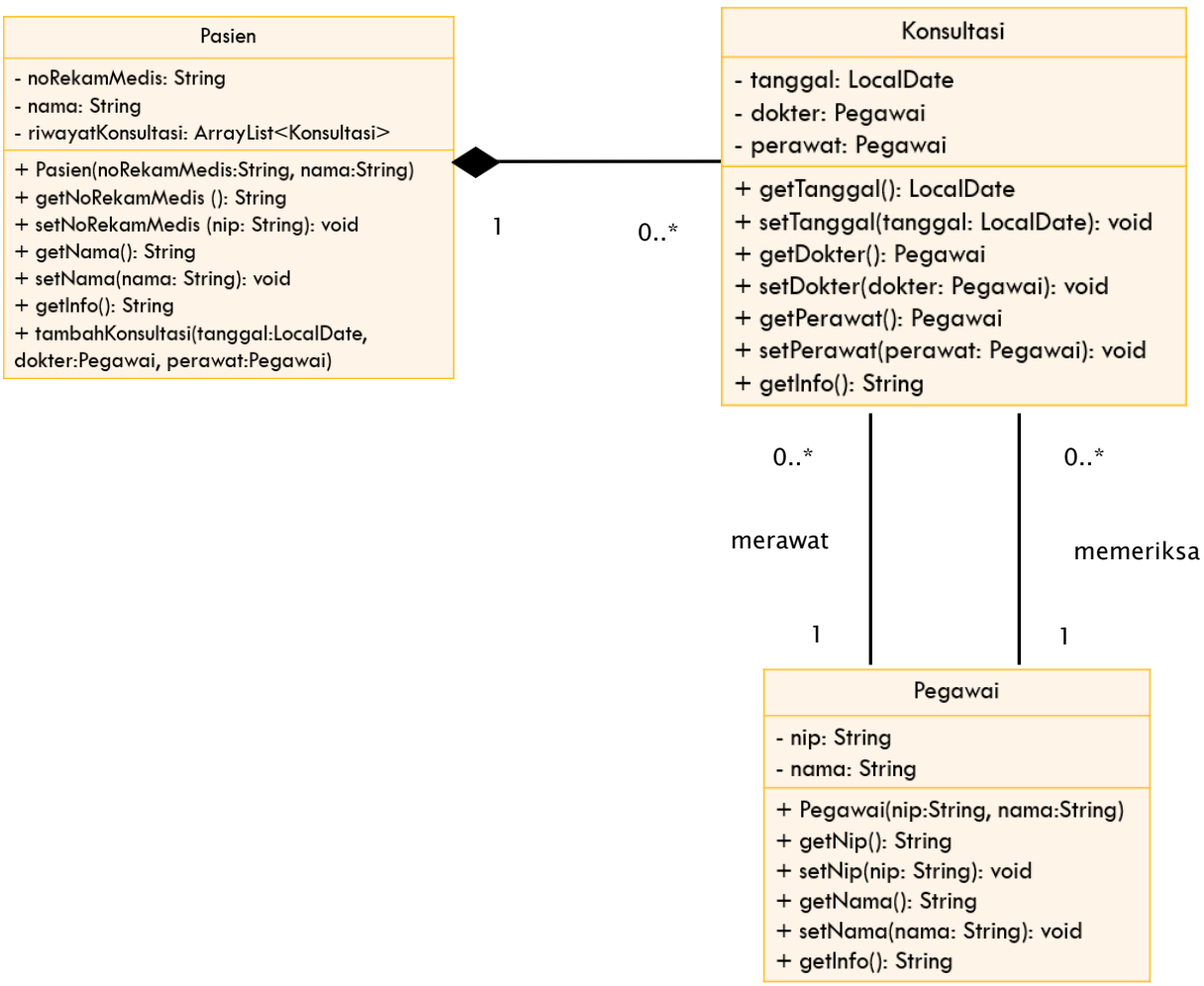
II. Introduction

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 :



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 + ")";
}
```

Jobsheet_4

Welcome | Pegawai.java

Pegawai.java

```java
package Jobsheet4;

/**
 * Employe
 */
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 getInfo(){
        return nama + " (" + nip + ")";
    }

}
```

⊗ 0 ⚠ 0 | 0 | Java: Lightweight Mode | Ln 1, Col 1 | Spaces: 4 | UTF-8 | CRLF | Java

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.

```
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

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

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

```
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:

```
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* 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
```

```
No Rekam Medis          : 343298
Nama                    : Puspa Widya
Riwayat Konsultasi :
         Tanggal: 2021-08-11, Dokter: dr. Ani (1234), Perawat: Ns. Desi (5678)
         Tanggal: 2021-09-11, Dokter: dr. Bagus (4567), Perawat: Ns. Eka (91011)


No Rekam Medis          : 997744
Nama                    : Yenny Anggraeni
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?
   - In the **Employee**, **Patient**, and **Consultation** classes, **setter** and **getter** methods are used to access and modify the private attributes of each class.
   - The benefit of use setter and getter is a have encapsulation the attributes cannot be accessed directly.
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?
   - while the Consult class may not have an explicitly defined parameterized constructor, it still has a default constructor provided by Java unless another constructor is defined.
3. Notice the *Consult* class, which attributes are of type *object*?
   - In summary, the attributes of the Konsultasi class that are of type object are:
   - dokter (of type Pegawai)
   - perawat (of type Pegawai)

4. Pay attention to *the Consultation* class, on which line does it show that the Consultation class has a relationship with the Employee class?
   - Line 4:
     private Pegawai dokter;
   - This line declares an attribute dokter of type Pegawai, indicating that the Konsultasi class has a relationship with the Pegawai class.

5. Notice in the Patient class, what does the consultation code.getInfo() do?
   - The method getInfo() in the Konsultasi class provides a structured way to display relevant details about a consultation by aggregating information from its attributes and associated objects (the doctor and nurse).
6. In the getInfo() method in the Patient class, there is a line of code: if (!historyConsultation.isEmpty()) What does the line do?
   - The line if (!historyConsultation.isEmpty()) serves as a conditional check to ensure that there is relevant consultation history available before executing further code related to that history.
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?

- the line this.historyConsultation = new ArrayList<>(); is crucial for initializing the consultation history for a patient. Omitting this line results in a null reference for historyConsultation, leading to runtime errors when attempting to use it. Proper initialization ensures that the object can safely store and manage consultation records.

IV. Assignment

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

Tugasjs4 > KursusMengemudi.java > {} Jobsheet4.Tugasjs4

```java
package Jobsheet4.Tugasjs4;
import java.util.ArrayList;
import java.time.LocalDate;

/**
 * KursusMengemudi
 */
public class KursusMengemudi {

    private String courseName;
    private LocalDate tanggalCourse;
    private String instructorName;
    private double priceModul;
    private ArrayList<Student>StudentList;

    public KursusMengemudi(String courseName, String instuctorName, double priceMo
        this.courseName = courseName;
        this.priceModul = priceModul;
        this.instructorName = instuctorName;
        this.StudentList = new ArrayList<>();
    }

    public void setPricePraktek(double priceModul){
        this.priceModul = priceModul;
    }

    public void setLisenseValid(String praktekModul){
```

Tugasjs4 > ModulPraktek.java > ModulPraktek > setTanggal(LocalDate)

```java
package Jobsheet4.Tugasjs4;
import java.time.LocalDate;
/**
 * ModulPraktek
 */
public class ModulPraktek {

    private LocalDate tanggalInstructorDate;
    private Pegawai InstructorMobil;
    private int nilaiStudent;

    public ModulPraktek(LocalDate tanggalInstructorDate, Pegawai InstructorMobil,
        this.tanggalInstructorDate = tanggalInstructorDate;
        this.InstructorMobil = InstructorMobil;
        this.nilaiStudent = nilaiStudent;
    }
    public LocalDate getTanggal(){
        return tanggalInstructorDate;
    }

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

    public Pegawai getInstructorMobil(){
        return InstructorMobil;
    }
```

Tugasjs4 > DemoCourseMobil.java > DemoCourseMobil

```java
package Jobsheet4.Tugasjs4;

import java.time.LocalDate;

/**
 * DemoCourse
 */
public class DemoCourseMobil {

    Run | Debug
    public static void main(String[] args) {
        Pegawai ani = new Pegawai(instructorName:"Ani Siswanto", instrukturID:"221
        Pegawai bagus = new Pegawai(instructorName:"Bagus Laras", instrukturID:"11

        Student student1 = new Student(studentName:"Bani Ismail", studentCourse:"M
        Student student2 = new Student(studentName:"Bahar Subahar", studentCourse:

        ModulPraktek modulPraktek1 = new ModulPraktek(LocalDate.of(year:2024,month
        ModulPraktek modulPraktek2 = new ModulPraktek(LocalDate.of(year:2024,month

        student1.addModulPraktek(modulPraktek1);
        System.out.println(student1.getValidasiLisensi());
        System.out.println();

        student2.addModulPraktek(modulPraktek2);
        System.out.println(student2.getValidasiLisensi());
        System.out.println();
```

-

```
Student Course: Mengemudi Mobil
Student Name: Bani Ismail
Riwayat Student:

         Tanggal Praktek : 2024-08-11    InstructorMobil: Nama InstructorBagus LarasID : 112     Nilai Student: 75
Student TIDAK memenuhi syarat untuk mendapatkan lisensi


Student Course: Mengemudi Mobil
Student Name: Bahar Subahar
Riwayat Student:
         Tanggal Praktek : 2024-07-15    InstructorMobil: Nama InstructorAni SiswantoID : 221    Nilai Student: 85
Student memenuhi syarat untuk mendapatkan lisensi
```