SAFRIZAL RAHMAN SIB 2G

https://github.com/safrizalrahman46/PBO_SAFRIZ_THEVIGILANTE

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.

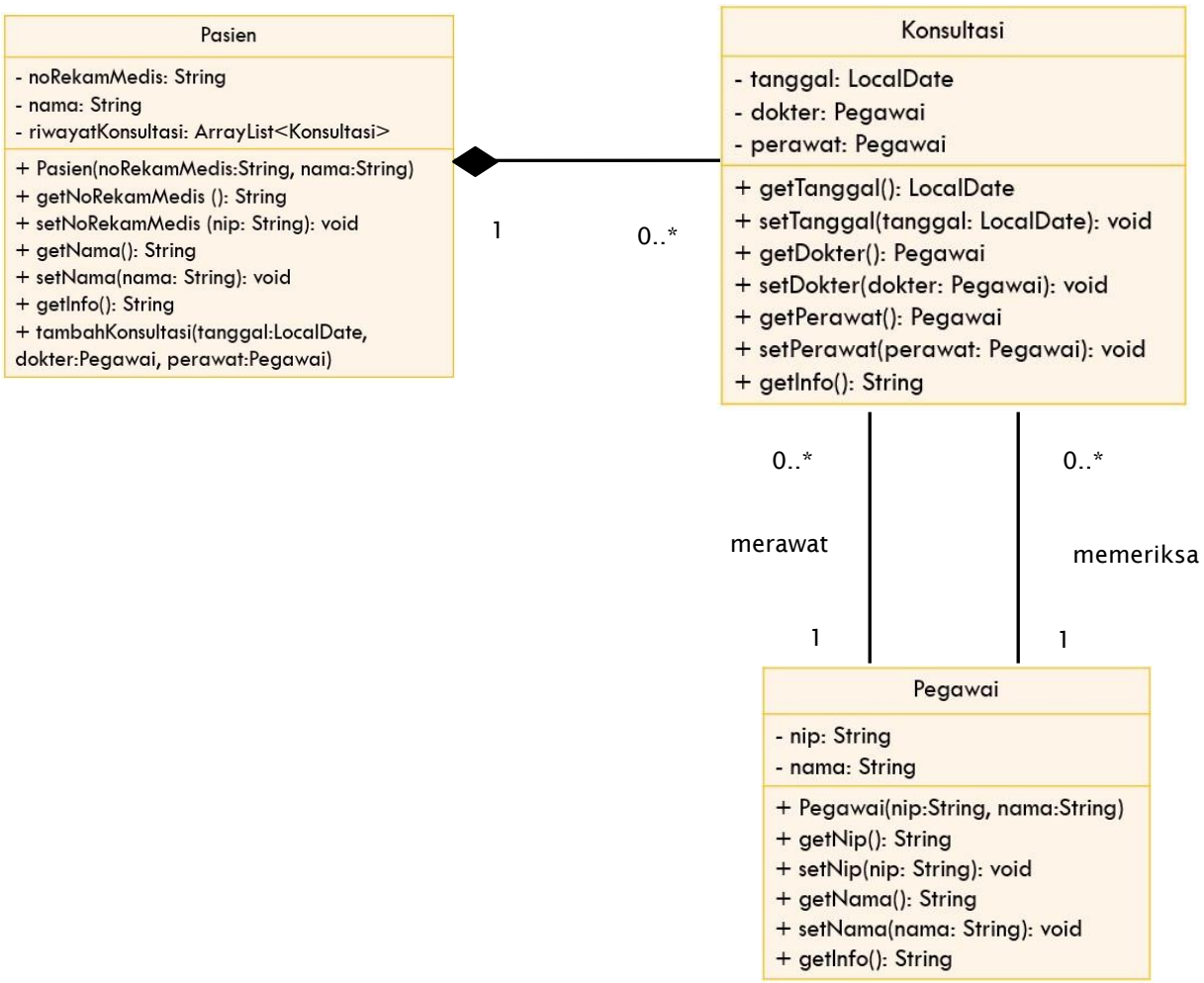
II. Introduction

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 :



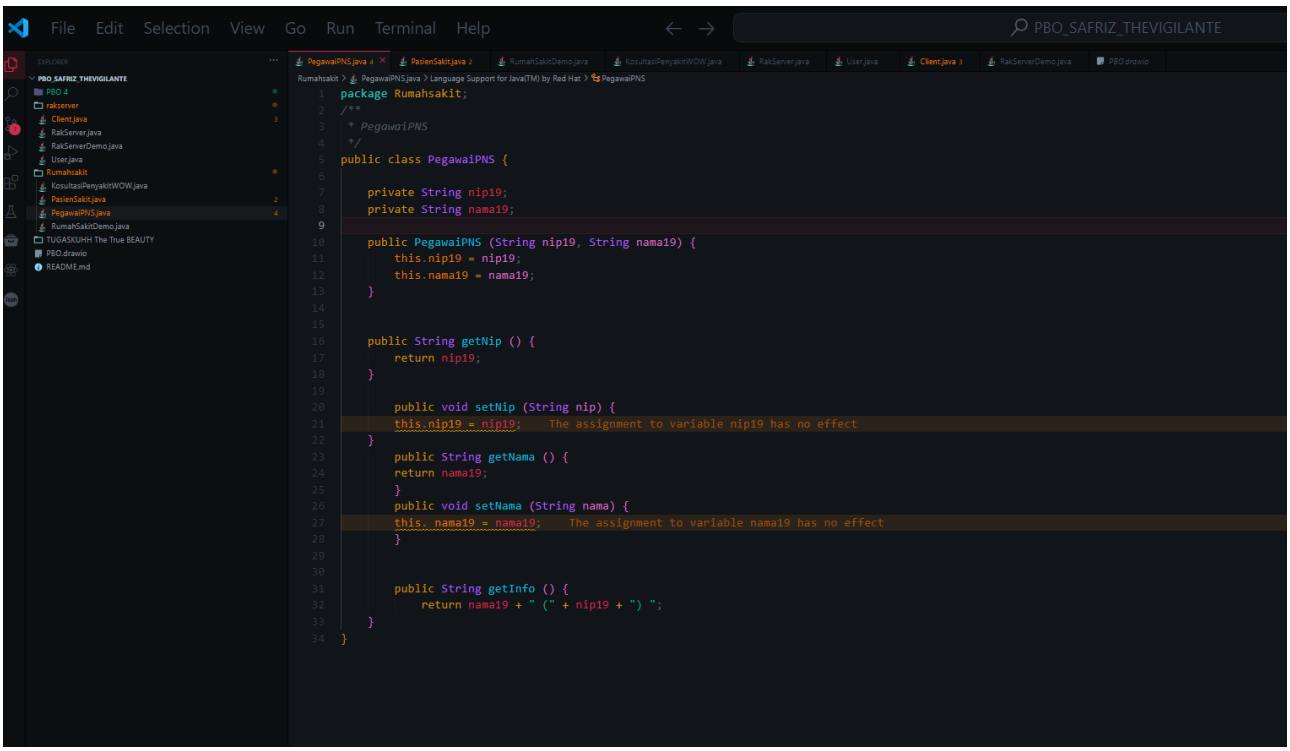
a. Create a new folder with the name Hospital.

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;
}
```



```java
package Rumahsakit;
/**
 * PegawaiPNS
 */
public class PegawaiPNS {

    private String nip19;
    private String nama19;

    public PegawaiPNS (String nip19, String nama19) {
        this.nip19 = nip19;
        this.nama19 = nama19;
    }


    public String getNip () {
        return nip19;
    }

        public void setNip (String nip) {
        this.nip19 = nip19;    The assignment to variable nip19 has no effect
    }
        public String getNama () {
        return nama19;
        }
        public void setNama (String nama) {
        this. nama19 = nama19;    The assignment to variable nama19 has no effect
        }


        public String getInfo () {
            return nama19 + " (" + nip19 + ") ";
    }
}
```

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

```java
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:

```java
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;
    }
}
```

```java
import java.util.ArrayList; // Add this import

public class PasienSakit {

    private String noRekamMedis;
    private String nama;
    private ArrayList<KosultasiPenyakitWOW> riwayatKonsultasi;    Field riwayatKonsultasi can be final

    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;
}
```

```java
import java.util.ArrayList; // Add this import

public class PasienSakit {

    private String noRekamMedis;
    private String nama;
    private ArrayList<KosultasiPenyakitWOW> riwayatKonsultasi;    Field riwayatKonsultasi can be final

    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;
    }
```

```java
package Rumahsakit;
/**
 * PegawaiPNS
 */
public class PegawaiPNS {

    private String nip19;
    private String nama19;

    public PegawaiPNS (String nip19, String nama19) {
        this.nip19 = nip19;
        this.nama19 = nama19;
    }


    public String getNip () {
        return nip19;
    }

        public void setNip (String nip) {
        this.nip19 = nip19;    The assignment to variable nip19 has no effect
    }
        public String getNama () {
        return nama19;
        }
        public void setNama (String nama) {
        this. nama19 = nama19;    The assignment to variable nama19 has no effect
        }


        public String getInfo () {
            return nama19 + " (" + nip19 + ") ";
    }
}
```

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;
}
```

aiPNS.java 4 | PasienSakit.java 2 | RumahSakitDemo.java | KosultasiPenyakitWOW.java | RakServer.java 3 | User.java 3 | Client.java 3 | RakServerDemo.java

kit > KosultasiPenyakitWOW.java > ...

```java
public class KosultasiPenyakitWOW {
    public void setTanggal(LocalDate tanggal) {
        this.tanggal = tanggal;
    }

    public PegawaiPNS getDokter() {
        return dokter;
    }

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

    public PegawaiPNS getPerawat() {
        return perawat;
    }

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

    // public String getInfo() {
    //     String info = "";
    //     info += "Tanggal: " + tanggal.toString();
    //     info += "\nDokter: " + dokter.getInfo();
    //     info += "\nPerawat: " + perawat.getInfo();
    //     return info;
    // }
    public String getInfo() {
        String info = "";
        info += "\tTanggal: " + tanggal;
        info += ", Dokter: " + dokter.getInfo();
        info += ", Perawat: " + perawat.getInfo();
        info += "\n";
        return info;
    }

}
```

```java
//     }
// }
package Rumahsakit;
import java.time.LocalDate;
import java.util.ArrayList; // Add this import

public class PasienSakit {

    private String noRekamMedis;
    private String nama;
    private ArrayList<KosultasiPenyakitWOW> riwayatKonsultasi;    Field riwayatKonsultasi can be final

    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;
    }

    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 (KosultasiPenyakitWOW konsultasi : riwayatKonsultasi) {
                info += konsultasi.getInfo();
            }
        } else {
            info += "Belum ada riwayat konsultasi";
        }
        info += "\n";
        return info;
    }

    public PasienSakit(String noRekamMedis, String nama) {
        this.noRekamMedis = noRekamMedis;
        this.nama = nama;
        this.riwayatKonsultasi = new ArrayList<KosultasiPenyakitWOW>();    Redundant type arguments in new expression (use diamond operator instead).
    }

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

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>();
}
```

```java
import java.util.ArrayList; // Add this import

public class PasienSakit {

    private String noRekamMedis;
    private String nama;
    private ArrayList<KosultasiPenyakitWOW> riwayatKonsultasi;    Field riwayatKonsultasi can be final

    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;
    }

    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 (KosultasiPenyakitWOW konsultasi : riwayatKonsultasi) {
                info += konsultasi.getInfo();
            }
        } else {
            info += "Belum ada riwayat konsultasi";
        }
        info += "\n";
        return info;
    }

    public PasienSakit(String noRekamMedis, String nama) {
        this.noRekamMedis = noRekamMedis;
        this.nama = nama;
        this.riwayatKonsultasi = new ArrayList<KosultasiPenyakitWOW>();    Redundant type arguments in new expression (use diamond operator instead).
    }

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

```
PS D:\TUGAS\SEMESTER 3\PBO\PBO_SAFRIZ_THEVIGILANTE>
```

```java
//     }
// }
package Rumahsakit;
import java.time.LocalDate;
import java.util.ArrayList; // Add this import

public class PasienSakit {

    private String noRekamMedis;
    private String nama;
    private ArrayList<KosultasiPenyakitWOW> riwayatKonsultasi;    Field riwayatKonsultasi can be final

    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;
    }

    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 (KosultasiPenyakitWOW konsultasi : riwayatKonsultasi) {
                info += konsultasi.getInfo();
            }
        } else {
            info += "Belum ada riwayat konsultasi";
        }
        info += "\n";
        return info;
    }

    public PasienSakit(String noRekamMedis, String nama) {
        this.noRekamMedis = noRekamMedis;
        this.nama = nama;
        this.riwayatKonsultasi = new ArrayList<KosultasiPenyakitWOW>();    Redundant type arguments in new expression (use diamond operator instead).
    }

    public void tambahKonsultasi(LocalDate tanggal, PegawaiPNS dokter, PegawaiPNS perawat) {
        KosultasiPenyakitWOW konsultasi = new KosultasiPenyakitWOW();
        konsultasi.setTanggal(tanggal);
        konsultasi.setDokter(dokter);
        konsultasi.setPerawat(perawat);
        riwayatKonsultasi.add(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:

```
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);
}
```

```
        this.noRekamMedis = noRekamMedis;
    }

    public String getNama() {
        return nama;
    }

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

    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 (KosultasiPenyakitWOW konsultasi : riwayatKonsultasi) {
                info += konsultasi.getInfo();
            }
        } else {
            info += "Belum ada riwayat konsultasi";
        }
        info += "\n";
        return info;
    }

    public PasienSakit(String noRekamMedis, String nama) {
        this.noRekamMedis = noRekamMedis;
        this.nama = nama;
        this.riwayatKonsultasi = new ArrayList<KosultasiPenyakitWOW>();    Redundant type arguments in new expression (use diamond op
    }

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

OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   DEVDB
:\TUGAS\SEMESTER 3\PBO\PBO_SAFRIZ_THEVIGILANTE>
```

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;
}
```

```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 (KosultasiPenyakitWOW konsultasi : riwayatKonsultasi) {
                info += konsultasi.getInfo();
            }
        } else {
            info += "Belum ada riwayat konsultasi";
        }
        info += "\n";
        return info;
    }

    public PasienSakit(String noRekamMedis, String nama) {
        this.noRekamMedis = noRekamMedis;
        this.nama = nama;
        this.riwayatKonsultasi = new ArrayList<KosultasiPenyakitWOW>();    Redundant type argum
    }
```

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 {
    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());
    }
}
```

```java
package Rumahsakit;
import java.time.LocalDate;

public class RumahSakitDemo {
    Run main | Debug main | Run | Debug
    public static void main(String[] args) {
        PegawaiPNS ani = new PegawaiPNS(nip19:"1234", nama19:"dr. Ani");
        PegawaiPNS bagus = new PegawaiPNS(nip19:"4567", nama19:"dr. Bagus");

        PegawaiPNS desi = new PegawaiPNS(nip19:"1234", nama19:"Ns. Desi");
        PegawaiPNS eka = new PegawaiPNS(nip19:"4567", nama19:"Ns. Eka");

        PasienSakit pasien1 = new PasienSakit(noRekamMedis:"343298", nama:"Puspa Widya");
        pasien1.tambahKonsultasi(LocalDate.of(year:2021 , month:8,dayOfMonth:11), ani, desi);
        pasien1.tambahKonsultasi(LocalDate.of(year:2021, month:9,dayOfMonth:11), bagus, eka);

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

        PasienSakit pasien2 = new PasienSakit(noRekamMedis:"997744", nama:"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

ANSWER

1. Purpose of Setter and Getter Methods:

- Setter: Used to set or change the value of an object attribute. This is useful for validating or manipulating data before it is stored.

- Getter: Used to retrieve the value of an object attribute. It helps in retrieving data from an object without directly accessing its attributes, supporting encapsulation.

2. Constructors in Consultation Class:

- If the `ConsultingWOWDisease` class does not explicitly define a constructor with parameters, Java automatically provides a default constructor without parameters. Therefore, even if no constructor is defined with parameters, the class still has a default constructor. Object Type Attributes of the Consultation Class:

- In class `KonsultationPainWOW`, the attributes of the object type are:

- doctor (of type CivilServant)

- nurse (of type CivilServantEmployee) Employee Class Relationship:

- The relationship between the ConsultationPainWOW class and the Civil ServantEmployee class can be seen in the following lines in the ConsultationPainWOW class:

- doctor Private civil servant;

- private civil servant nurse;

- This shows that ConsultationDiseaseWOW has two attributes that are objects of the Civil Servant class, indicating that the ConsultationDiseaseWOW class is related to the Civil Servant class. The function of the consultation.getInfo() code in the Patient class:

- The consultation.getInfo() code in the getInfo method of the PatientSick class calls the getInfo() method of the ConsultationDiseaseWOW object, which returns information related to the consultation, including the date, doctor, and nurse. This is used to display the patient's consultation details.

6. The function of the if(!historyConsultation.isEmpty()) line:

- This line checks whether the historyConsultation list is empty or not. If it is not empty, there is a consultation history stored, and the information will be displayed. If it is empty, display a message indicating that there is no consultation history.

7. This line.historyConsultation = new ArrayList(); serves to initialize the historyConsultation as a new ArrayList when creating the PatientSick object. This ensures that historyConsultation is ready to store consultations. If this line is deleted,

historyConsultation will become null and will generate a NullPointerException when trying to add a consultation.

Translated with DeepL.com (free version)

Based on experiment 1, answer the related questions:

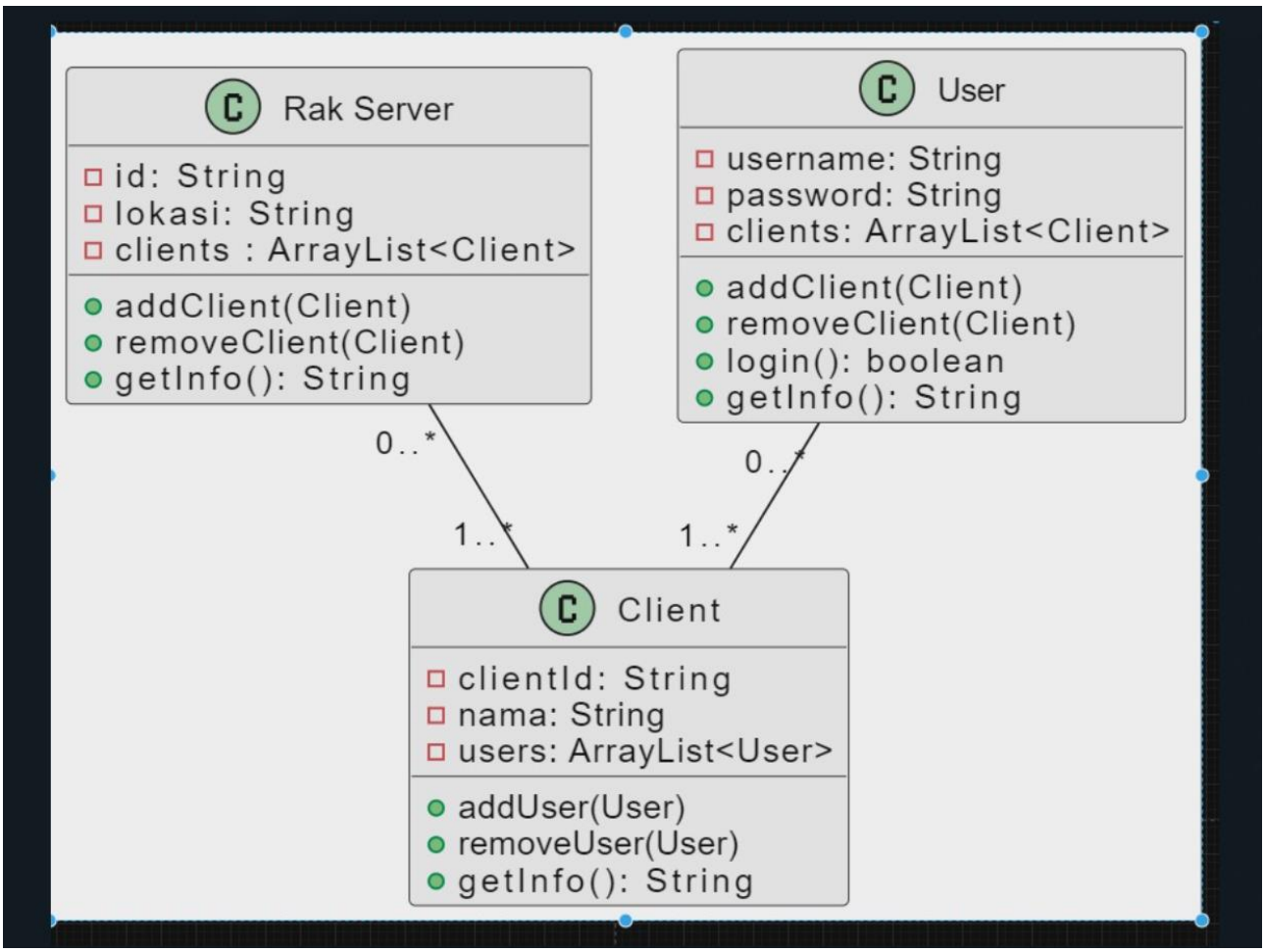
1. In the *Employee, Patient,* and Consultation classes, there are method *setters* and *getters* for each of their attributes. What is the use of *the setter and getter* methods ?
2. In the *Consult* class there is not explicitly a constructor with parameters. Does this mean that the Consult class doesn't 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.



```java
package rakserver;

import java.util.ArrayList;

public class Client {
    private String clientId;    Field clientId can be final
    private String nama;    Field nama can be final
    private ArrayList<User> users;    Field users can be final

    public Client(String clientId, String nama) {
        this.clientId = clientId;
        this.nama = nama;
        this.users = new ArrayList<>();
    }

    public void addUser(User user) {
        users.add(user);
    }

    public void removeUser(User user) {
        users.remove(user);
    }

    public String getInfo() {
        StringBuilder info = new StringBuilder();
        info.append(str:"Client ID: ").append(clientId).append(str:"\n");
        info.append(str:"Nama: ").append(nama).append(str:"\n");
        info.append(str:"Users:\n");
        for (User user : users) {
            info.append(user.getInfo()).append(str:"\n");
        }
        return info.toString();
    }
}
```

```java
package rakserver;

import java.util.ArrayList;

public class RakServer {
    private String id;    Field id can be final
    private String lokasi;    Field lokasi can be final
    private ArrayList<Client> clients;    Field clients can be final

    public RakServer(String id, String lokasi) {
        this.id = id;
        this.lokasi = lokasi;
        this.clients = new ArrayList<>();
    }

    public void addClient(Client client) {
        clients.add(client);
    }

    public void removeClient(Client client) {
        clients.remove(client);
    }

    public String getInfo() {
        StringBuilder info = new StringBuilder();
        info.append(str:"ID: ").append(id).append(str:"\n");
        info.append(str:"Lokasi: ").append(lokasi).append(str:"\n");
        info.append(str:"Clients:\n");
        for (Client client : clients) {
            info.append(client.getInfo()).append(str:"\n");
        }
        return info.toString();
    }
}
```

```java
package rakserver;

public class RakServerDemo {
    Run main | Debug main | Run | Debug
    public static void main(String[] args) {
        // Create Users
        User user1 = new User(username:"alice", password:"password123");
        User user2 = new User(username:"bob", password:"password456");

        // Create Clients
        Client client1 = new Client(clientId:"C001", nama:"Client One");
        Client client2 = new Client(clientId:"C002", nama:"Client Two");

        // Add users to clients
        client1.addUser(user1);
        client1.addUser(user2);
        client2.addUser(user1);

        // Create RakServers
        RakServer rakServer1 = new RakServer(id:"R001", lokasi:"Location A");
        RakServer rakServer2 = new RakServer(id:"R002", lokasi:"Location B");

        // Add clients to RakServers
        rakServer1.addClient(client1);
        rakServer1.addClient(client2);
        rakServer2.addClient(client2);

        // Print information
        System.out.println(x:"RakServer 1 Info:");
        System.out.println(rakServer1.getInfo());

        System.out.println(x:"RakServer 2 Info:");
        System.out.println(rakServer2.getInfo());
    }
}
```

```java
package rakserver;

import java.util.ArrayList;

public class User {
    private String username;    Field username can be final
    private String password;    Field password can be final
    private ArrayList<Client> clients;    Field clients can be final

    public User(String username, String password) {
        this.username = username;
        this.password = password;
        this.clients = new ArrayList<>();
    }

    public void addClient(Client client) {
        clients.add(client);
    }

    public void removeClient(Client client) {
        clients.remove(client);
    }

    public boolean login(String password) {
        return this.password.equals(password);
    }

    public String getInfo() {
        StringBuilder info = new StringBuilder();
        info.append(str:"Username: ").append(username).append(str:"\n");
        info.append(str:"Clients:\n");
        for (Client client : clients) {
            info.append(client.getInfo()).append(str:"\n");
        }
        return info.toString();
    }
}
```

```
PS D:\TUGAS\SEMESTER 3\PBO\PBO_SAFRIZ_THEVIGILANTE>  d:; cd 'd:\TUGAS\SEMESTER 3\PBO\PBO_SAFRIZ_THEVIGILANTE'; & 'C:\Program Files\Java\jdk-17\bin\j
RAHMAN\AppData\Roaming\Code\User\workspaceStorage\dd9db320bee5261e3e407b28365db461\redhat.java\jdt_ws\PBO_SAFRIZ_THEVIGILANTE_9fb04fb0\bin' 'rakserv
RakServer 1 Info:
ID: R001
Lokasi: Location A
Clients:
Client ID: C001
Nama: Client One
Users:
Username: alice
Clients:

Username: bob
Clients:


Client ID: C002
Nama: Client Two
Users:
Username: alice
Clients:



RakServer 2 Info:
ID: R002
Lokasi: Location B
Clients:
Client ID: C002
Nama: Client Two
Users:
Username: alice
Clients:
```