

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.

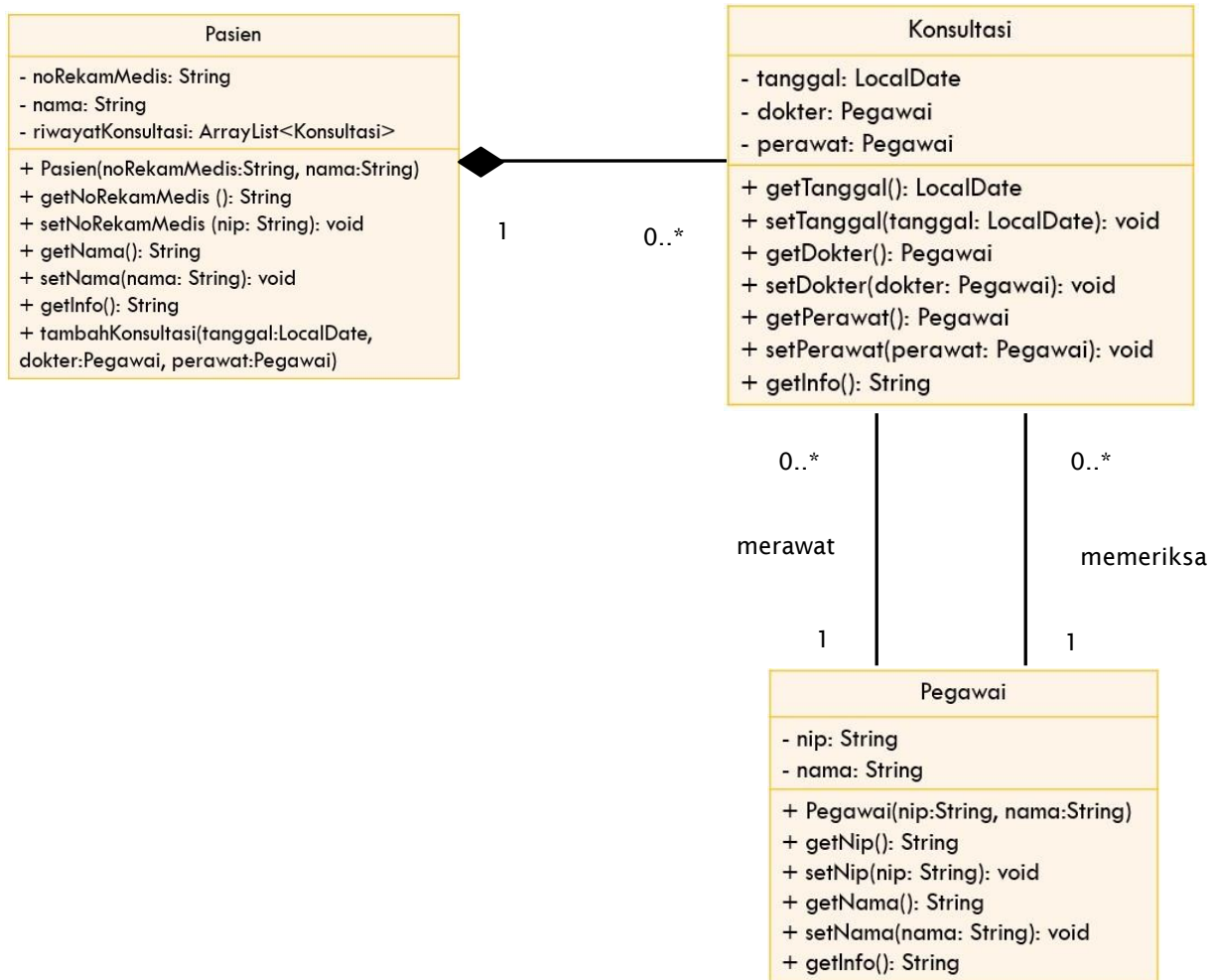
II. Pendahuluan

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:



- Buatlah folder baru dengan nama RumahSakit
- Buatlah class Pegawai. Tambahkan atribut nip dan nama pada class Pegawai dengan akses modifier private

```

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

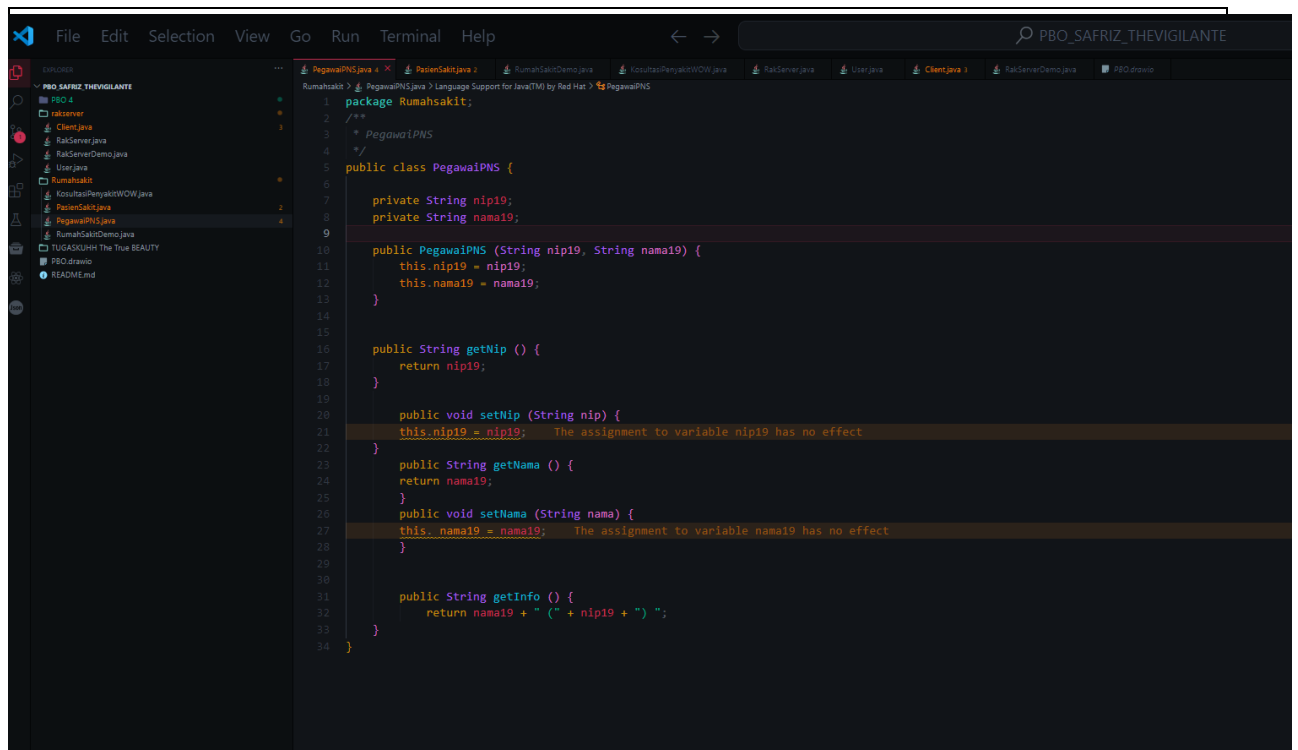
```

- 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;
}

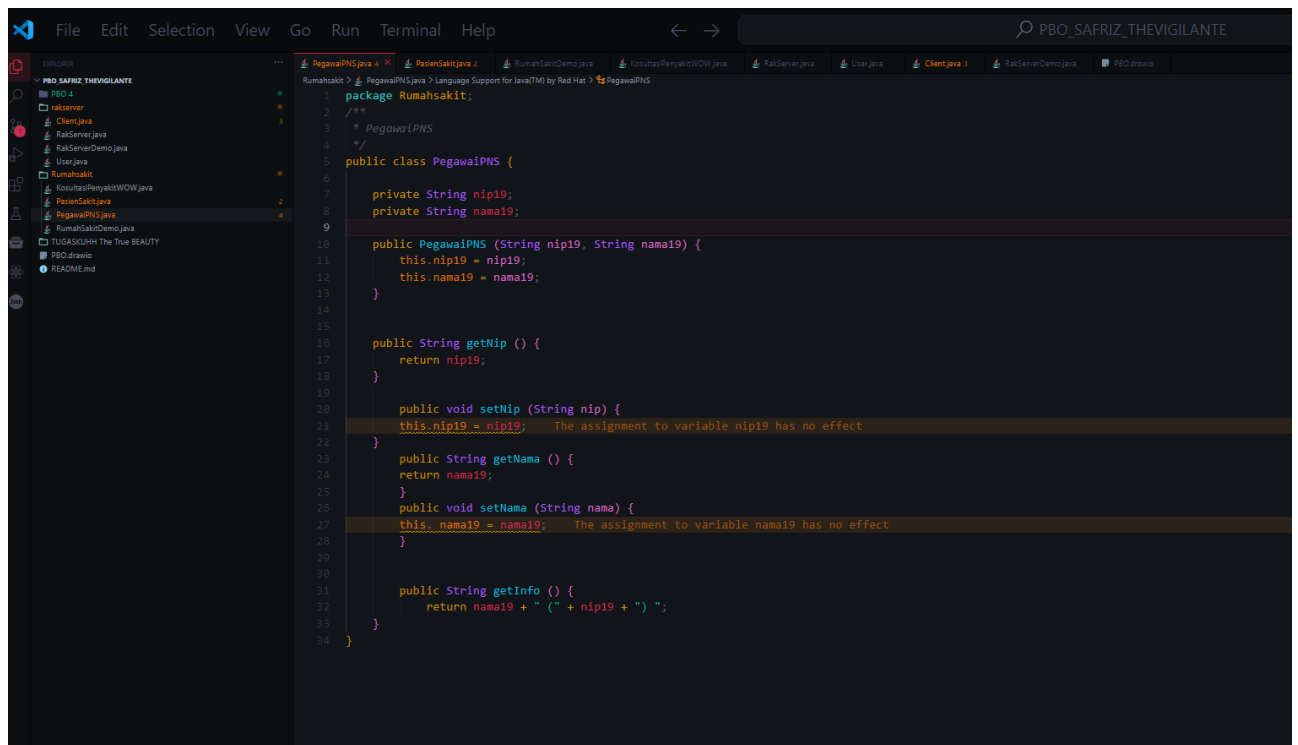
public void setNip(String nip) {
    this.nip = nip;
}

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 getName() {
        return nama;
    }

    public void setName(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;
}

```

```

71 //
72 //
73 package Rumahsakit;
74 import java.time.LocalDate;
75 import java.util.ArrayList; // Add this import
76
77 public class PasienSakit {
78
79     private String noRekamMedis;
80     private String nama;
81     private ArrayList<KonsultasiPenyakitMOM> riwayatKonsultasi; // Field riwayatKonsultasi can be final
82
83     public String getNoRekamMedis() {
84         return noRekamMedis;
85     }
86
87     public void setNoRekamMedis(String noRekamMedis) {
88         this.noRekamMedis = noRekamMedis;
89     }
90
91     public String getNama() {
92         return nama;
93     }
94
95     public void setNama(String nama) {
96         this.nama = nama;
97     }
98
99     public String getInfo() {
100         String info = "";
101         info += "No Rekam Medis: " + this.noRekamMedis + "\n";
102         info += "Nama: " + this.nama + "\n";
103         if (!riwayatKonsultasi.isEmpty()) {
104             info += "Riwayat Konsultasi:\n";
105             for (KonsultasiPenyakitMOM konsultasi : riwayatKonsultasi) {
106                 info += konsultasi.getInfo();
107             }
108         } else {
109             info += "Belum ada riwayat konsultasi";
110         }
111         info += "\n";
112         return info;
113     }
114
115     public PasienSakit(String noRekamMedis, String nama) {
116         this.noRekamMedis = noRekamMedis;
117         this.nama = nama;
118         this.riwayatKonsultasi = new ArrayList<KonsultasiPenyakitMOM>(); // Redundant type arguments in new expression (use diamond operator instead).
119     }
120
121     public void tambahKonsultasi(LocalDate tanggal, PegawaiPNS dokter, PegawaiPNS perawat) {
122         KonsultasiPenyakitMOM konsultasi = new KonsultasiPenyakitMOM();
123         konsultasi.setTanggal(tanggal);
124         konsultasi.setDokter(dokter);
125         konsultasi.setPerawat(perawat);
126         riwayatKonsultasi.add(konsultasi);
127     }
128 }
129

```

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;
}

```

```
71 //  
72 // }  
73 package RumahSakit;  
74 import java.time.LocalDate;  
75 import java.util.ArrayList; // Add this import  
76  
77 public class PasienSakit {  
78  
79     private String noRekamMedis;  
80     private String nama;  
81     private ArrayList<KonsultasiPenyakitMOM> riwayatKonsultasi; // Field riwayatKonsultasi can be final  
82  
83     public String getNoRekamMedis() {  
84         return noRekamMedis;  
85     }  
86  
87     public void setNoRekamMedis(String noRekamMedis) {  
88         this.noRekamMedis = noRekamMedis;  
89     }  
90  
91     public String getNama() {  
92         return nama;  
93     }  
94  
95     public void setNama(String nama) {  
96         this.nama = nama;  
97     }  
98  
99     public String getInfo() {  
100         String info = "";  
101         info += "No Rekam Medis: " + this.noRekamMedis + "\n";  
102         info += "Nama: " + this.nama + "\n";  
103         if (riwayatKonsultasi.isEmpty()) {  
104             info += "Riwayat Konsultasi:\n";  
105             for (KonsultasiPenyakitMOM konsultasi : riwayatKonsultasi) {  
106                 info += konsultasi.getInfo();  
107             }  
108         } else {  
109             info += "Belum ada riwayat konsultasi";  
110         }  
111         info += "\n";  
112         return info;  
113     }  
114  
115     public PasienSakit(String noRekamMedis, String nama) {  
116         this.noRekamMedis = noRekamMedis;  
117         this.nama = nama;  
118         this.riwayatKonsultasi = new ArrayList<KonsultasiPenyakitMOM>(); // Redundant type arguments in new expression (use diamond operator instead).  
119     }  
120  
121     public void tambahKonsultasi(LocalDate tanggal, PegawaiPNS dokter, PegawaiPNS perawat) {  
122         KonsultasiPenyakitMOM konsultasi = new KonsultasiPenyakitMOM();  
123         konsultasi.setTanggal(tanggal);  
124         konsultasi.setDokter(dokter);  
125         konsultasi.setPerawat(perawat);  
126         riwayatKonsultasi.add(konsultasi);  
127     }  
128 }  
129
```

- 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;
}
```

The screenshot shows an IDE with the following code in the `KonsultasiPenyakitMOW.java` file:

```
1 package Rumahakit;
2 import java.time.LocalDate;
3
4
5 public class KonsultasiPenyakitMOW {
6     private LocalDate tanggal;
7     private PegawaiPNS dokter;
8     private PegawaiPNS perawat;
9
10
11     public LocalDate getTanggal() {
12         return tanggal;
13     }
14
15     public void setTanggal(LocalDate tanggal) {
16         this.tanggal = tanggal;
17     }
18
19     public PegawaiPNS getDokter() {
20         return dokter;
21     }
22
23     public void setDokter(PegawaiPNS dokter) {
24         this.dokter = dokter;
25     }
26
27     public PegawaiPNS getPerawat() {
28         return perawat;
29     }
30
31     public void setPerawat(PegawaiPNS perawat) {
32         this.perawat = perawat;
33     }
34
35
36     // public String getInfo() {
37     //     String info = "";
38     //     info += "Tanggal: " + tanggal.toString();
39     //     info += "\ndokter: " + dokter.getInfo();
40     //     info += "\nPerawat: " + perawat.getInfo();
41     //     return info;
42     // }
43
44     public String getInfo() {
45         String info = "";
46         info += "\nTanggal: " + tanggal;
47         info += ", Dokter: " + dokter.getInfo();
48         info += ", Perawat: " + perawat.getInfo();
49         info += "\n";
50         return info;
51     }
52 }
```

- 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;  
}
```

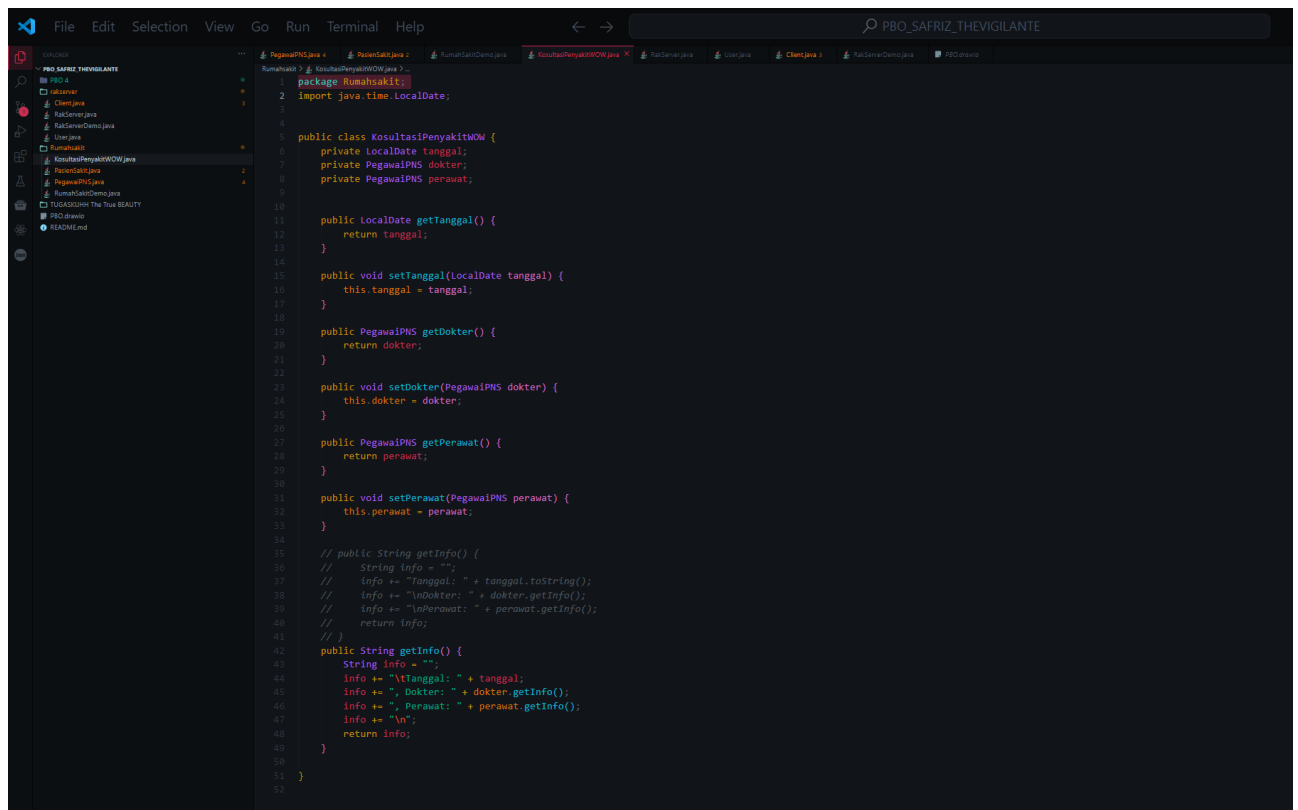
1. Implementasikan method getInfo() sebagai berikut:


```

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>();
}

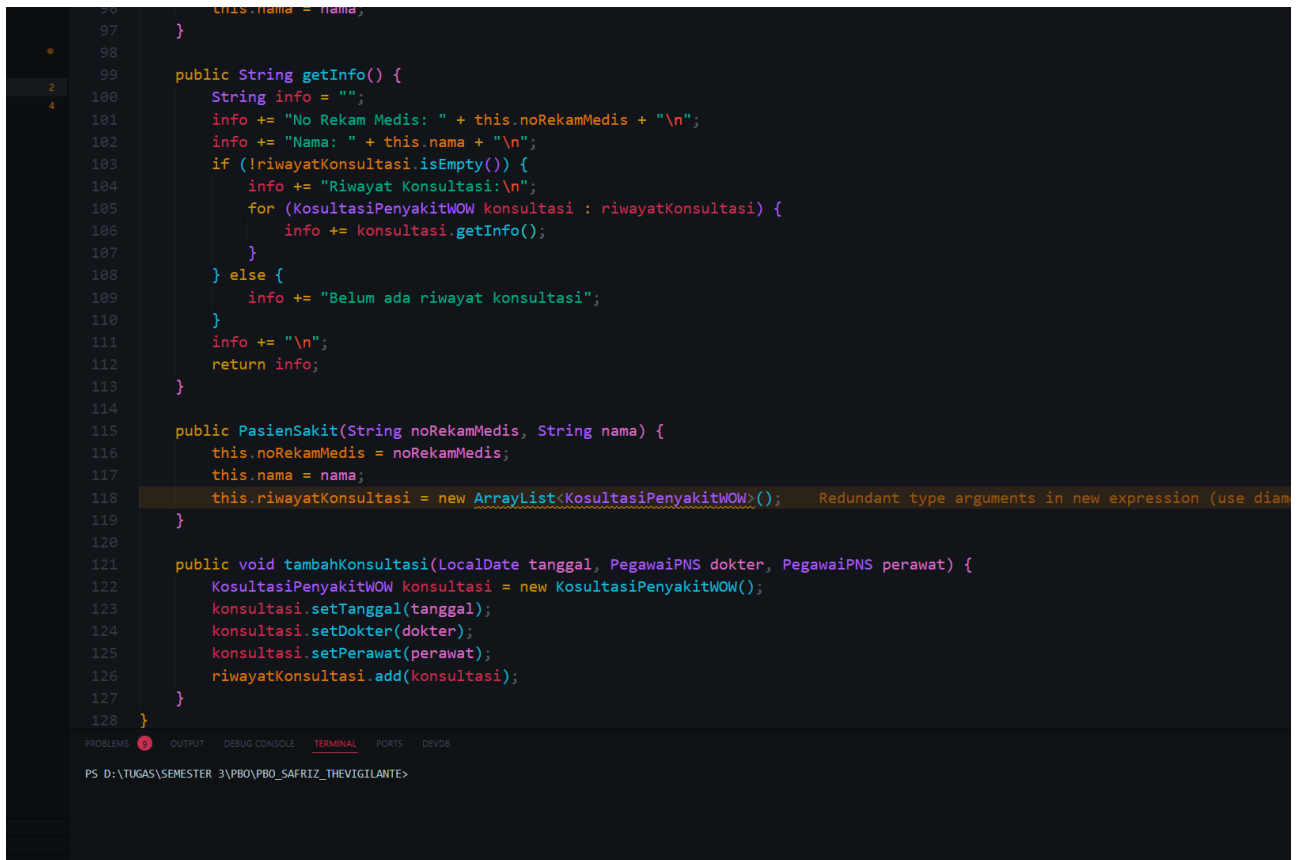
```

- 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);
}

```



```

96     this.nama = nama;
97 }
98
99 public String getInfo() {
100     String info = "";
101     info += "No Rekam Medis: " + this.noRekamMedis + "\n";
102     info += "Nama: " + this.nama + "\n";
103     if (!riwayatKonsultasi.isEmpty()) {
104         info += "Riwayat Konsultasi:\n";
105         for (KonsultasiPenyakitWOW konsultasi : riwayatKonsultasi) {
106             info += konsultasi.getInfo();
107         }
108     } else {
109         info += "Belum ada riwayat konsultasi";
110     }
111     info += "\n";
112     return info;
113 }
114
115 public PasienSakit(String noRekamMedis, String nama) {
116     this.noRekamMedis = noRekamMedis;
117     this.nama = nama;
118     this.riwayatKonsultasi = new ArrayList<KonsultasiPenyakitWOW>(); Redundant type arguments in new expression (use diam
119 }
120
121 public void tambahKonsultasi(LocalDate tanggal, PegawaiPNS dokter, PegawaiPNS perawat) {
122     KonsultasiPenyakitWOW konsultasi = new KonsultasiPenyakitWOW();
123     konsultasi.setTanggal(tanggal);
124     konsultasi.setDokter(dokter);
125     konsultasi.setPerawat(perawat);
126     riwayatKonsultasi.add(konsultasi);
127 }
128 }

```

PROBLEMS 0 OUTPUT DEBUG CONSOLE TERMINAL PORTS DEVOPS

PS D:\TUGAS\SEMESTER 3\PRO\PRO_SAFRIZ_THEVIGILANTE>

- 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;
}

```

```

96         this.nama = nama;
97     }
98
99     public String getInfo() {
100         String info = "";
101         info += "No Rekam Medis: " + this.noRekamMedis + "\n";
102         info += "Nama: " + this.nama + "\n";
103         if (!riwayatKonsultasi.isEmpty()) {
104             info += "Riwayat Konsultasi:\n";
105             for (KonsultasiPenyakitWOW konsultasi : riwayatKonsultasi) {
106                 info += konsultasi.getInfo();
107             }
108         } else {
109             info += "Belum ada riwayat konsultasi";
110         }
111         info += "\n";
112         return info;
113     }
114
115     public PasienSakit(String noRekamMedis, String nama) {
116         this.noRekamMedis = noRekamMedis;
117         this.nama = nama;
118         this.riwayatKonsultasi = new ArrayList<KonsultasiPenyakitWOW>(); Redundant type arguments in new expression (use diam
119     }
120
121     public void tambahKonsultasi(LocalDate tanggal, PegawaiPNS dokter, PegawaiPNS perawat) {
122         KonsultasiPenyakitWOW konsultasi = new KonsultasiPenyakitWOW();
123         konsultasi.setTanggal(tanggal);
124         konsultasi.setDokter(dokter);
125         konsultasi.setPerawat(perawat);
126         riwayatKonsultasi.add(konsultasi);
127     }
128 }

```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS DEVDB

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

- 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;

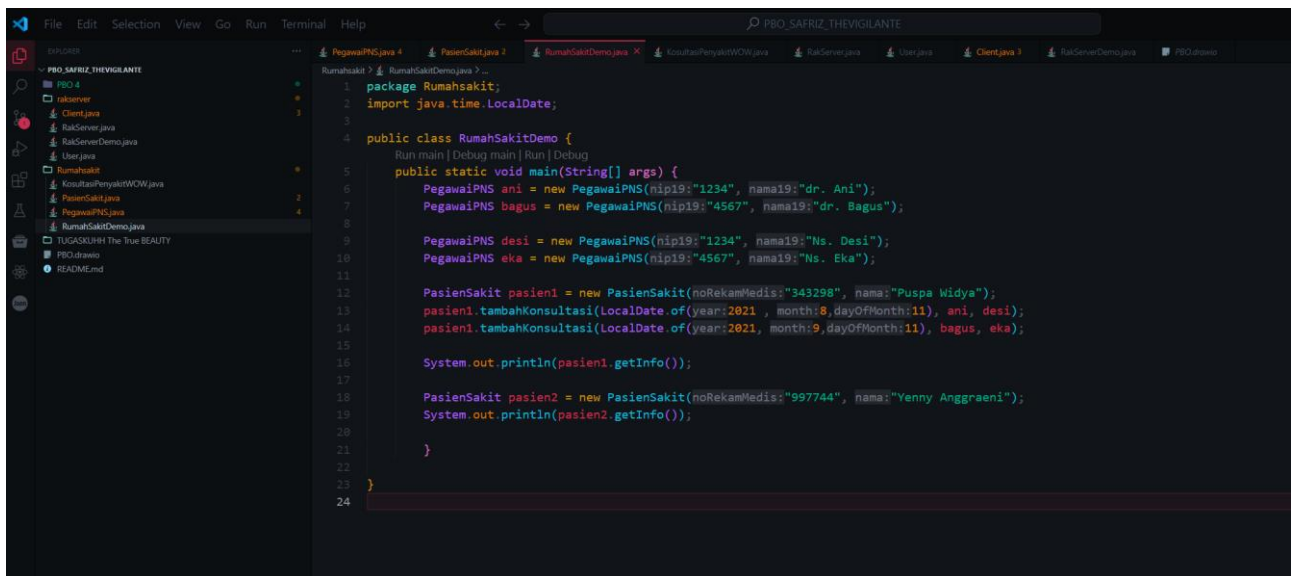
public class RumahSakitDemo {
    Run | Debug
    public static void main(String[] args) {
        Pegawai ani = new Pegawai("1234", "dr. Ani");
        Pegawai bagus = new Pegawai("4567", "dr. Bagus");

        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

```

```

PROBLEMS 6 OUTPUT DEBUG CONSOLE TERMINAL PORTS DEVD8
● PS D:\TUGAS\SEMESTER 3\PBO\PBO_SAFRIZ_THEVIGILANTE> & 'C:\Program Files\Java\jdk-17\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' '65db461\redhat.java\jdt_ws\PBO_SAFRIZ_THEVIGILANTE_9fb04fb0\bin' 'RumahSakit.RumahSakitDemo'
○ No Rekam Medis: 343298
  Nama: Puspa Widya
  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: 997744
  Nama: Yenny Anggraeni
  Belum ada riwayat konsultasi

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

```

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?

JAWABAN

1. Purpose of Method Setter and Getter:

- Setter: Used to set or change the value of an object attribute. This is useful for validating or manipulating data before it is saved.
- Getter: Used to retrieve the value of an object attribute. This aids in retrieving data from an object without directly accessing its attributes, supporting encapsulation.

2. Constructor in Consultation Class:

- If the class 'ConsultationDiseaseWOW' does not explicitly define a constructor with parameters, Java automatically provides a default constructor without parameters. Therefore, even though there is no defined constructor with parameters, the class still has a default constructor.

Attribut of Object Type in Consultation Class:

- In the class 'ConsultationDiseaseWOW', the attributes of object type are:

- doctor (of type CivilServantEmployee)

- nurse (of type CivilServantEmployee)

Employee Class Relationship:
 - The relationship between the ConsultationDiseaseWOW class and the Civil Servant Employee class can be seen in the following lines within the ConsultationDiseaseWOW class:

- private Civil Servant Employee doctor;

- private Civil Servant Employee nurse;

- This indicates that ConsultationDiseaseWOW has two attributes that are objects of the Civil Servant Employee class, showing that the ConsultationDiseaseWOW class is related to the Civil Servant Employee class.

Fungsi dari kode `konsultasi.getInfo()` dalam *class* Pasien:

- Kode `konsultasi.getInfo()` dalam method `getInfo` di *class* PasienSakit memanggil method `getInfo()` dari objek *KonsultasiPenyakitWOW*, yang mengembalikan informasi terkait konsultasi, termasuk tanggal, dokter, dan perawat. This is used to display patient consultation details.

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

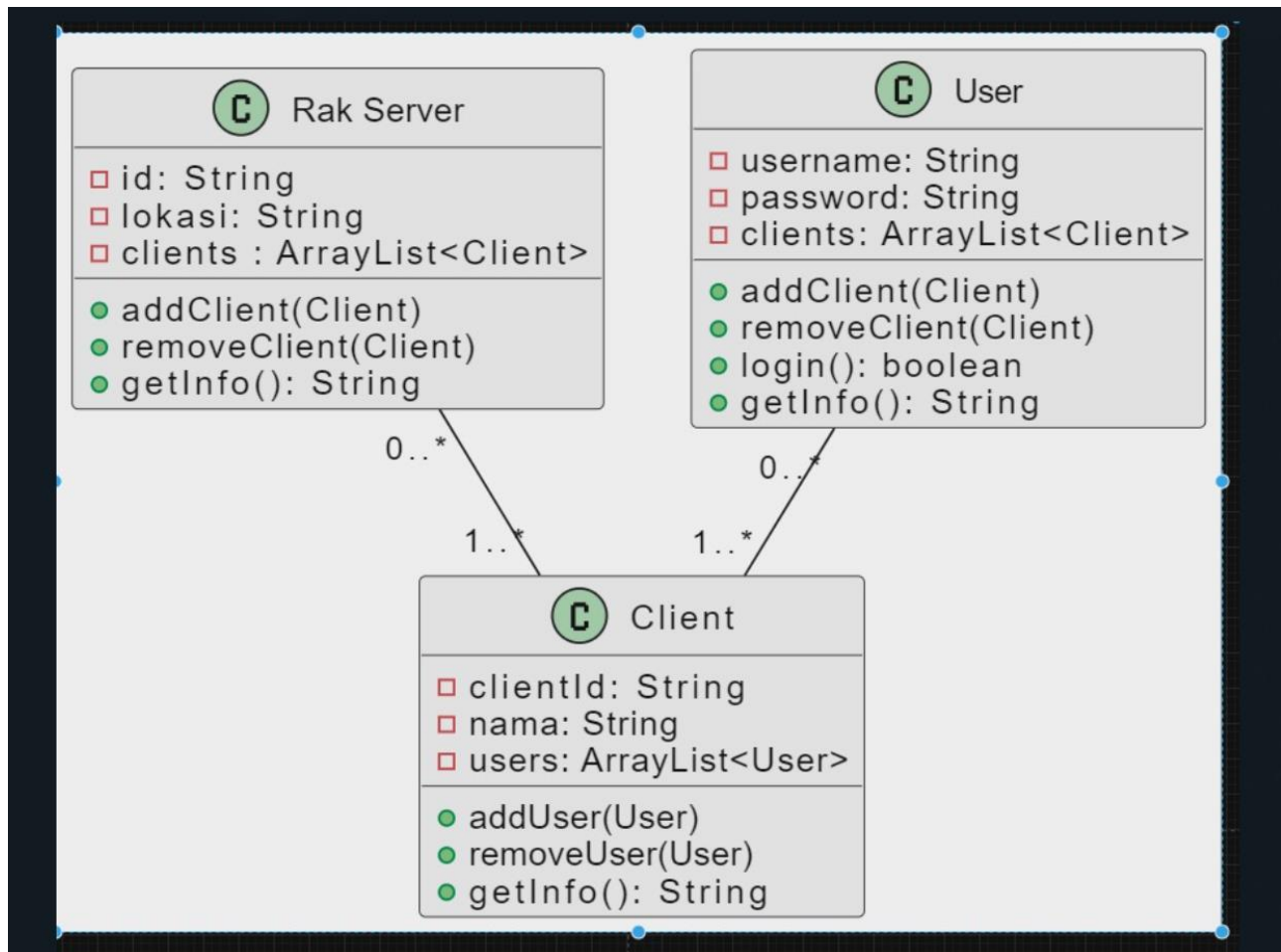
- This line checks whether the `riwayatKonsultasi` list is empty or not. If not empty, it means there is a stored consultation history, and that information will be displayed. If empty, display a message indicating that there is no consultation history.

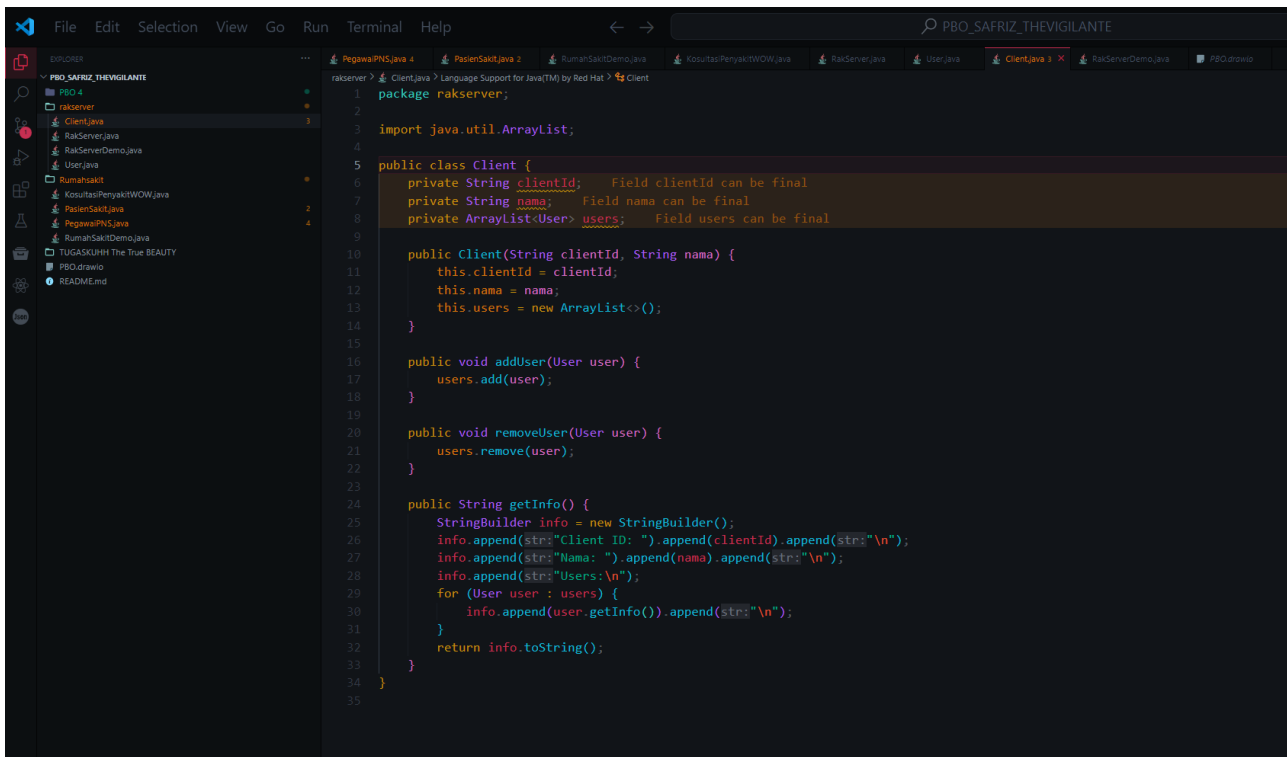
7. The line `this.riwayatKonsultasi = new ArrayList();` serves to initialize `riwayatKonsultasi` as a new `ArrayList` when creating a *PasienSakit* object. This ensures that the `consultationHistory` is

ready for storing consultations. If this line is removed, riwayatKonsultasi will become null and will result in a NullPointerException when trying to add a consultation.

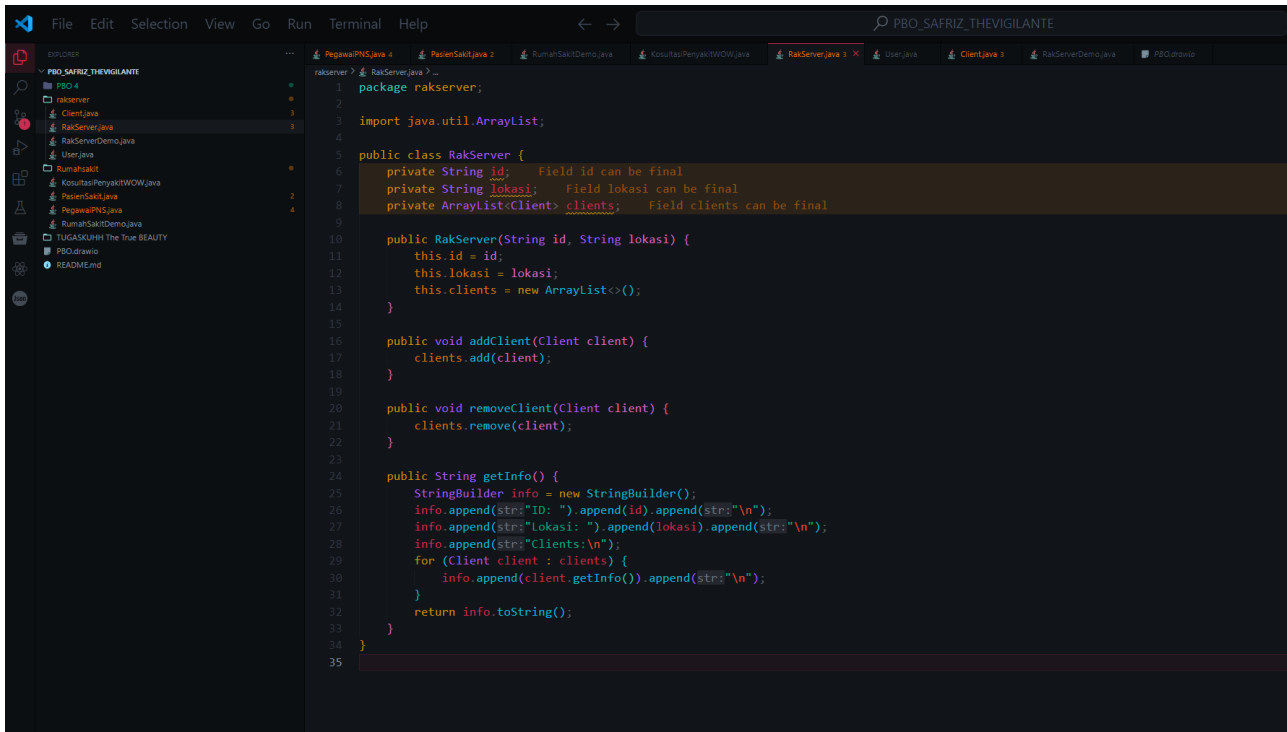
IV. Tugas

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

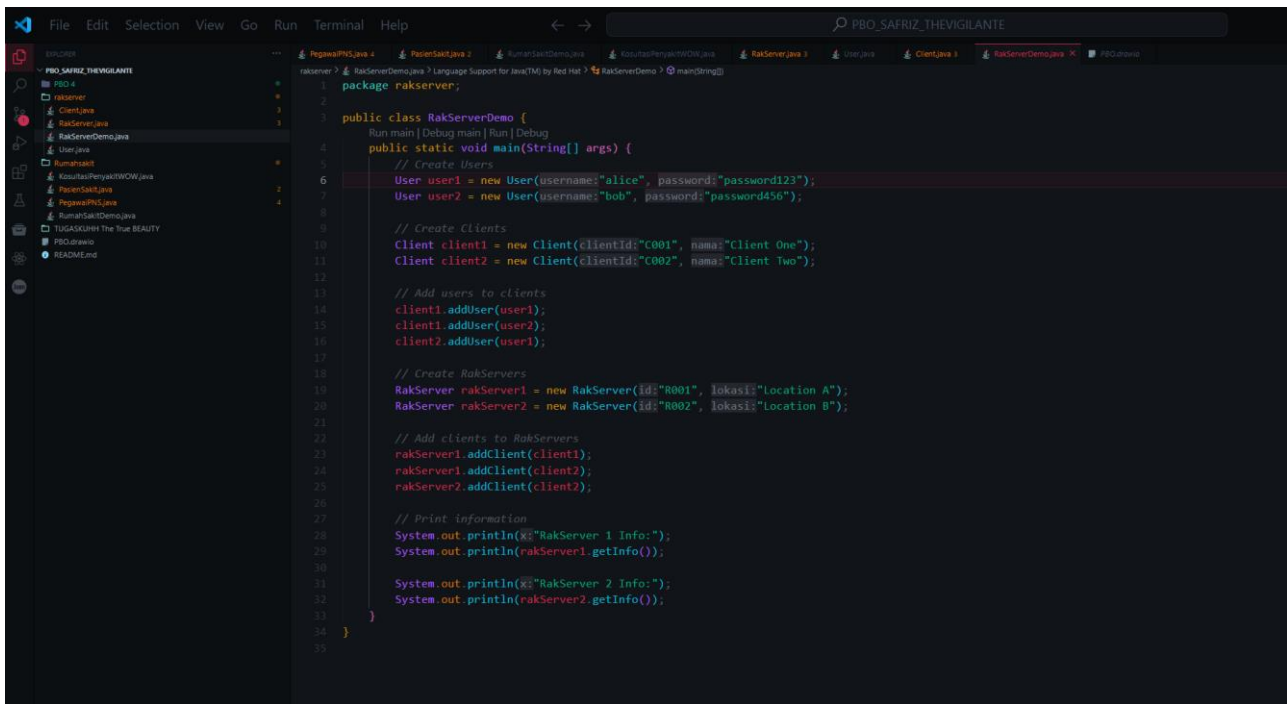




```
1 package rakserver;
2
3 import java.util.ArrayList;
4
5 public class Client {
6     private String clientId; Field clientId can be final
7     private String nama; Field nama can be final
8     private ArrayList<User> users; Field users can be final
9
10    public Client(String clientId, String nama) {
11        this.clientId = clientId;
12        this.nama = nama;
13        this.users = new ArrayList<>();
14    }
15
16    public void addUser(User user) {
17        users.add(user);
18    }
19
20    public void removeUser(User user) {
21        users.remove(user);
22    }
23
24    public String getInfo() {
25        StringBuilder info = new StringBuilder();
26        info.append(str:"Client ID: ").append(clientId).append(str:"\n");
27        info.append(str:"Nama: ").append(nama).append(str:"\n");
28        info.append(str:"Users:\n");
29        for (User user : users) {
30            info.append(user.getInfo()).append(str:"\n");
31        }
32        return info.toString();
33    }
34 }
35
```



```
1 package rakserver;
2
3 import java.util.ArrayList;
4
5 public class RakServer {
6     private String id; Field id can be final
7     private String lokasi; Field lokasi can be final
8     private ArrayList<Client> clients; Field clients can be final
9
10    public RakServer(String id, String lokasi) {
11        this.id = id;
12        this.lokasi = lokasi;
13        this.clients = new ArrayList<>();
14    }
15
16    public void addClient(Client client) {
17        clients.add(client);
18    }
19
20    public void removeClient(Client client) {
21        clients.remove(client);
22    }
23
24    public String getInfo() {
25        StringBuilder info = new StringBuilder();
26        info.append(str:"ID: ").append(id).append(str:"\n");
27        info.append(str:"Lokasi: ").append(lokasi).append(str:"\n");
28        info.append(str:"Clients:\n");
29        for (Client client : clients) {
30            info.append(client.getInfo()).append(str:"\n");
31        }
32        return info.toString();
33    }
34 }
35
```

```

1 package rakserver;
2
3 import java.util.ArrayList;
4
5 public class User {
6     private String username;    Field username can be final
7     private String password;    Field password can be final
8     private ArrayList<Client> clients;    Field clients can be final
9
10    public User(String username, String password) {
11        this.username = username;
12        this.password = password;
13        this.clients = new ArrayList<>();
14    }
15
16    public void addClient(Client client) {
17        clients.add(client);
18    }
19
20    public void removeClient(Client client) {
21        clients.remove(client);
22    }
23
24    public boolean login(String password) {
25        return this.password.equals(password);
26    }
27
28    public String getInfo() {
29        StringBuilder info = new StringBuilder();
30        info.append("\nUsername: ").append(username).append("\n");
31        info.append("\nClients:\n");
32        for (Client client : clients) {
33            info.append(client.getInfo()).append("\n");
34        }
35        return info.toString();
36    }
37 }
38

```

```

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\
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:

```