Jobsheet 04 ‑ Relasi Kelas

Name : Evan Diantha Fafian

Class : SIB 2G

Absent : 09

NIM : 2341760163

# Kompetensi

Setelah menempuh pokok bahasan ini, mahasiswa mampu:

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

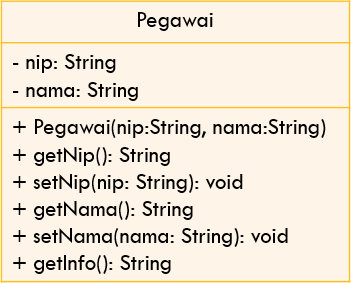
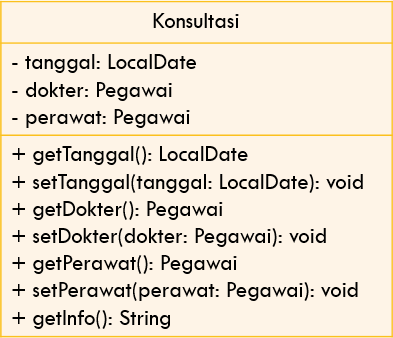
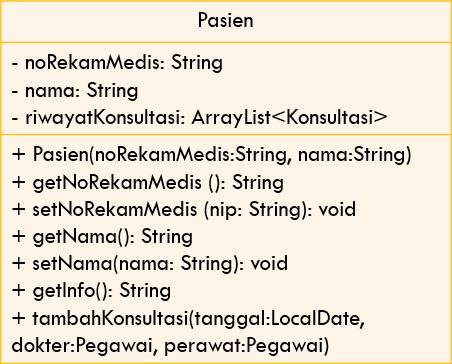
# Pendahuluan

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.

# Praktikum

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

Perhatikan diagram *class* berikut:



1

0..\*

0..\*

0..\*

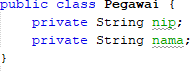
merawat

memeriksa

1

1

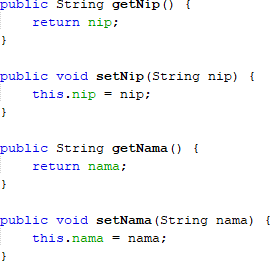
1. Buatlah folder baru dengan nama RumahSakit
2. Buatlah class Pegawai. Tambahkan atribut nip dan nama pada class Pegawai dengan akses modifier private



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



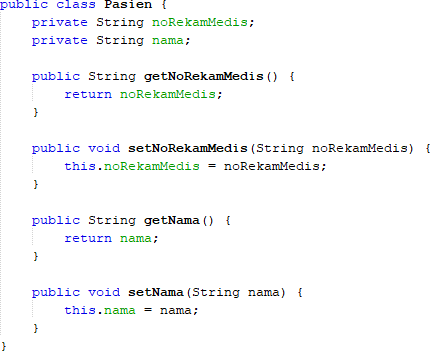
1. Implementasikan **setter** dan **getter** untuk class Pegawai.



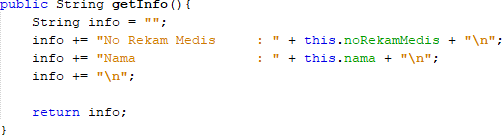
1. Implementasikan *method* getInfo()sebagai berikut:



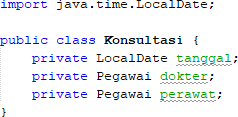
1. Selanjutnya buatlah class Pasien kemudian tambahkan atribut noRekamMedis dan nama pada class Pasien dengan access level modifier private. Sediakan pula setter dan getter untuk kedua atribut tersebut.
2. Buatlah constructor untuk class Pasien dengan parameter noRekamMedis dan nama



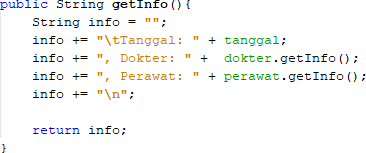
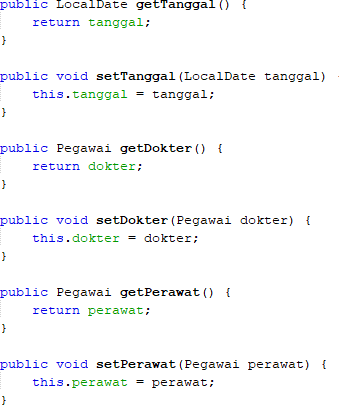
1. Implementasikan *method* getInfo()sebagai berikut:



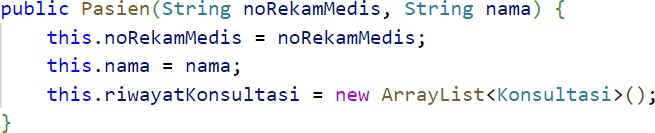
1. Sistem ini akan menyimpan data setiap konsultasi yang dilakukan pasien. Pasien bisa melakukan konsultasi lebih dari sekali. Oleh karena itu, data konsultasi akan disimpan dalam bentuk ArrayList dari objek-objek yang bertipe Konsultasi.
2. Buatlah class dengan nama Konsultasi dengan atribut tanggal 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.
3. Sediakan setter dan getter untuk masing-masing atribut pada class Konsultasi



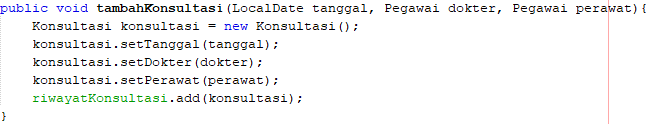
1. Implementasikan method getInfo() sebagai berikut:



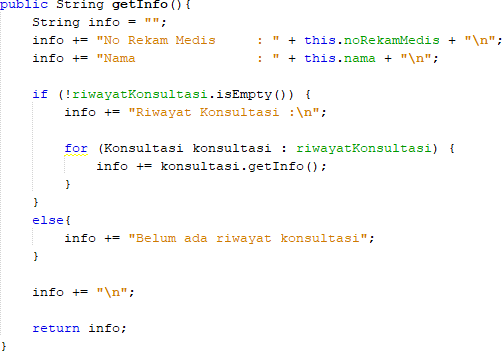
1. Untuk menyimpan data riwayat konsultasi pasien, maka tambahkan atribut riwayatKonsultasi pada class Pasien dengan tipe arrayList<Konsultasi>. Atribut ini akan menyimpan serangkaian objek bertipe Konsultasi. Import java.util.ArrayList agar dapat mendeklarasikan atribut bertipe ArrayList of object.
2. Buatlah constructor berparameter untuk class Pasien. Inisiasi nilai atribut noRekamMedis dan nama berdasarkan atribut nama. Instansiasi/buat ArrayList baru untuk atribut riwayatKonsultasi;



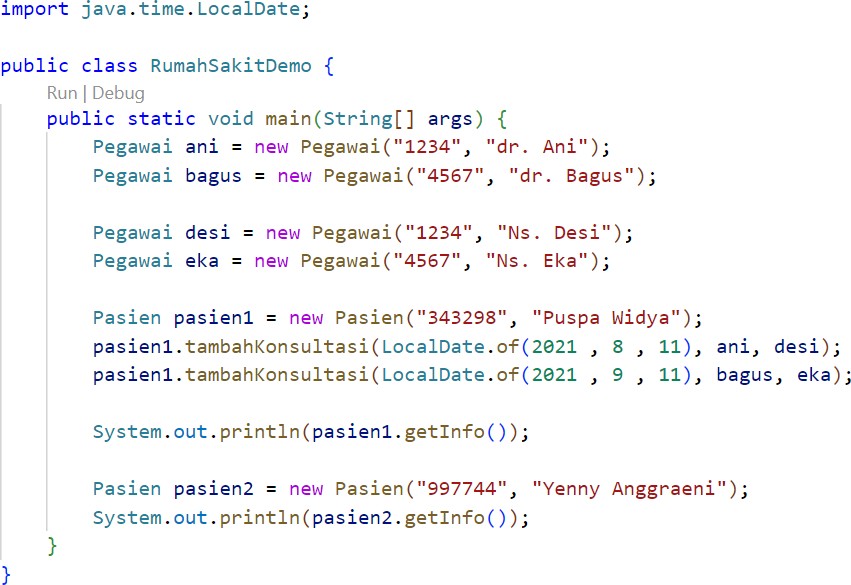
1. Lakukan import java.time.LocalDate agar dapat mendeklarasikan atribut tanggal bertipe LocalDate pada class Pasien. Selanjutnya, implementasikan method tambahKonsultasi() sebagai berikut:



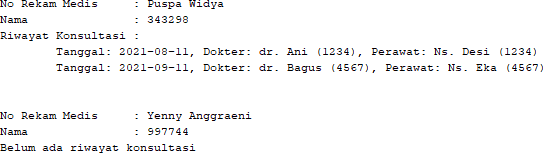
1. Modifikasi method getInfo() untuk mengembalikan info pasien dan daftar konsultasi yang pernah dilakukan



1. 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:



1. *Compile* kemudian *run* RumahSakitDemo dan didapatkan hasil seperti berikut:



# 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 ?

* Setters are used to change the value of an attribute. For example, if there is a name attribute, the setter method setName() is used to set (or change) the value of that attribute.
* Getters are used to retrieve (access) the value of an attribute. If we want to know the value of the name attribute, then we use the getName() method.

1. Di dalam *class* Konsultasi tidak secara eksplisit terdapat constructor dengan parameter. Apakah ini berarti class Konsultasi tidak memiliki constructor?

* Java automatically provides a default constructor without parameters, even though the "Consult" class does not have a special constructor. This constructor allows the creation of a “Consult” object, but attributes such as “date”, “doctor”, and “nurse” start with default values, namely “null” for objects and “0” for primitive data types. We can write a constructor with clear parameters to initialize the attributes when the object is created if desired. Therefore, classes still have constructors even though they cannot be seen directly in the code.

1. Perhatikan *class* Konsultasi, atribut mana saja yang bertipe *object*?

* dokter
* perawat

1. Perhatikan *class* Konsultasi, pada baris manakah yang menunjukan bahwa *class*

Konsultasi memiliki relasi dengan *class* Pegawai?

* 

1. Perhatikan pada *class* Pasien, apa yang dilakukan oleh kode konsultasi.getInfo()?

* In the Patient class, the code consult.getInfo() is used to retrieve information from each Consultation object in the Consultationhistory list and combine it into a string containing the patient's consultation history.

1. Pada method getInfo() dalam class Pasien, terdapat baris kode:

if (!riwayatKonsultasi.isEmpty())

Apakah yang dilakukan oleh baris tersebut?

* The if (!Consultation history.isEmpty()) code line in the getInfo() method in the Patient class functions to check whether the Consultation history list (which is an ArrayList of Consultation objects) is empty or not.

1. Pada constructor class Pasien, terdapat baris kode: this.riwayatKonsultasi = new ArrayList<>();

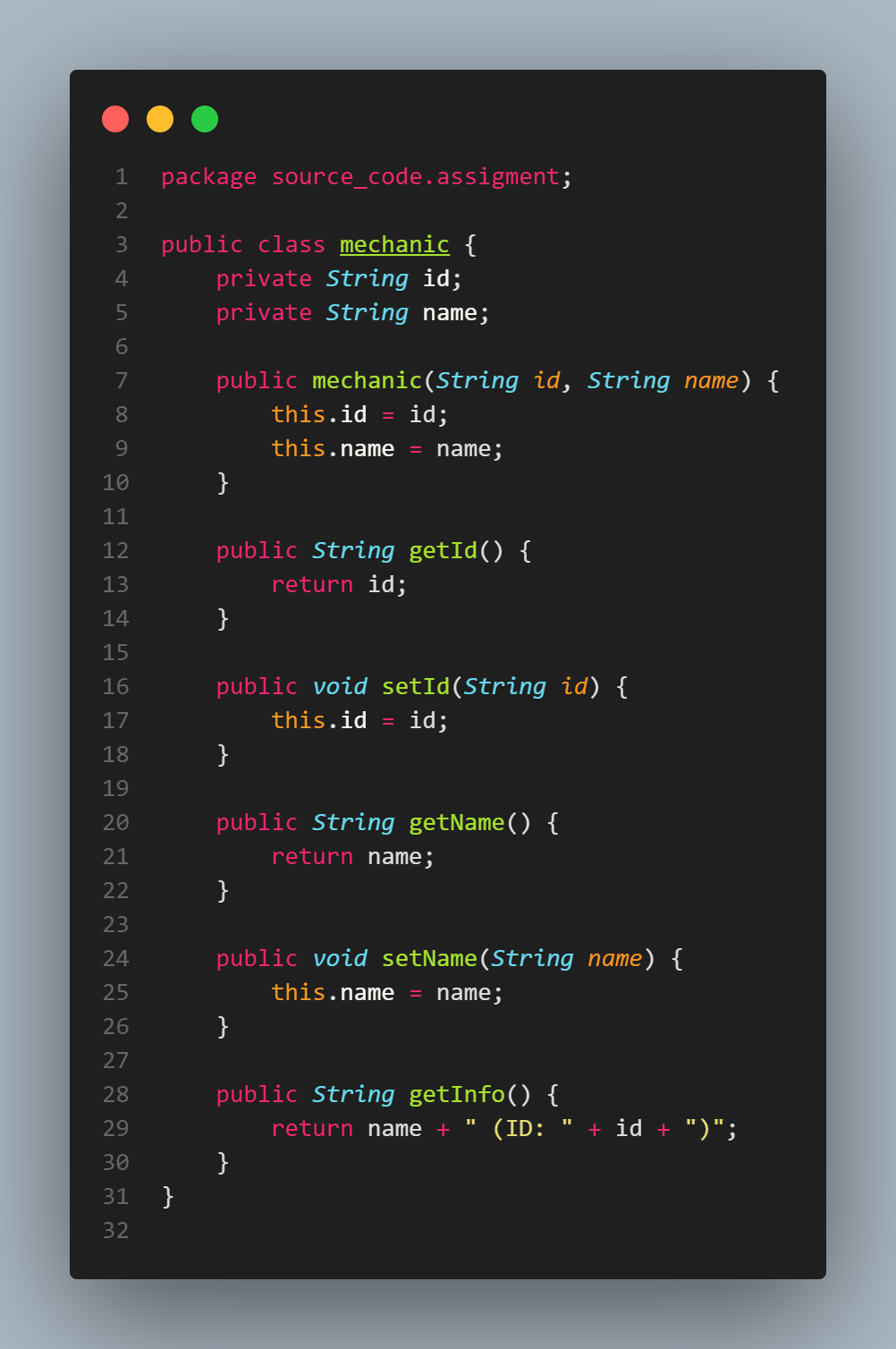
Apakah yang dilakukan oleh baris tersebut? Apakah yang terjadi jika baris tersebut dihilangkan?

* Code line this.Consultationhistory = new ArrayList<>(); in the constructor of the Patient class aims to initialize the Consultation history attribute as an empty ArrayList object. This line ensures that when the Patient object is created, theConsultationhistory attribute is ready to be used to store the patient's consultation history.

# Tugas

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

* mechanic class



Stores information about mechanics, such as their ID and name.

* service class



Session to service the car. It stores the date, the mechanic involved, and a description of the service (such as an oil change or tire rotation and others).

* car class



To represent a car. It stores the license plate number and owner's name. It also has a list (serviceHistory) of all the service sessions the car has had.

* ‘carServiceDemo’ main class



To run how to use the class by creating a car object, adding a service record, and printing the information.