Hospital Management

Tim 3

- Aizar Rahima Suprayitno (2702363885): Programmer
- Andre Dharmawan Ericson Sirait (2702382562): Laporan
- Ayman Abdullah Jabal (2702367473): Ide
- Evan Daniel (2702350573): PPT
- Musthofa Sahid Kamal (2702353026): Tester

PERMASALAHAN

Kesulitan dalam Mengelola data

Dalam pengelolaan data biasanya masih sering terdapat kesalahan input maupun sulit dalam mengakses data kembali

Kesulitan Pasien dalam Membuat Janji dengan Dokter

Pasien harus datang kerumah sakit untuk membuat janji dengan dokter, namun juga belum tentu bertemu dengan dokter yang diinginkan.

MyHospital

UNTUK MEMPERMUDAHKAN MENGELOLA DATA RUMAH SAKIT
DAN MEMUDAHKAN PASIEN UNTUK MEMBUAT JANJI DENGAN DOKTER

MANFAAT

Data Tertata dan Mudah Diakses

Data-data akan mudah digunakan dan aman

Pasien mudah Membuat Janji Dokter

Tanpa datang langsung kerumah sakit, pasien dapat melakukan janji dengan dokter.

```java
class Patient {
    String name;
    int age;
    String gender;

    public Patient(String name, int age, String gender) {
        this.name = name;
        this.age = age;
        this.gender = gender;
    }

    @Override
    public String toString() {
        return "Patient{" +
                "name='" + name + '\'' +
                ", age=" + age +
                ", gender='" + gender + '\'' +
                '}';
    }
}
```

Try Pitch

Patient.java

Pemrograman Java kelas ini untuk data pasien, dengan menggunakan tiga properti yaitu: nama, umur, dan jenis kelamin.

Method konstruktor yang ada dalam kelas ini digunakan untuk menginisialisasi objek Patient baru. Lalu, kelas ini juga memiliki method toString() yang telah di-override. Method ini digunakan untuk mengubah objek Patient menjadi representasi string.

Kode ini menciptakan kelas Patient yang sederhana dan memudahkan penanganan data pasien dalam aplikasi Java.

@Override adalah sebuah anotasi yang digunakan untuk memberi tahu kompiler bahwa metode toString() yang ditulis di dalam kelas Patient akan menggantikan metode toString() yang ada dalam kelas induk Object.

@Override juga dapat membantu meningkatkan efisiensi kompilasi kode. Karena dengan menggunakan simbol @Override, kompiler hanya perlu melakukan pengecekan satu kali saja apakah metode yang ditulis di kelas itu adalah override dari metode di kelas induknya, bukan mengulang pengecekan ini setiap kali kelas tersebut di-compile.

Doctor.java

```java
public class Doctor {
    String name;
    int age;
    String specialist;

    public Doctor(String name, int age, String specialist) {
        this.name = name;
        this.age = age;
        this.specialist = specialist;
    }

    @Override
    public String toString() {
        return "Doctor{" +
                "name='" + name + '\'' +
                ", age=" + age +
                ", specialist='" + specialist + '\'' +
                '}';
    }
}
```

Kelas ini memiliki tiga variabel instance, yaitu "name", "age", dan "specialist". Konstruktor yang ditulis pada baris ke 5 sampai dengan 9 digunakan untuk menginisialisasi nilai-nilai variabel tersebut.

Selain itu, pada kelas ini juga ada method toString yang digunakan untuk mengubah objek dari kelas ini menjadi string. Method ini akan mengembalikan representasi string dari objek Doctor dengan format "Doctor{name='<nama>', age=<umur>, specialist='<spesialis>'}".

```java
import java.util.ArrayList;

class Hospital {
    ArrayList<Doctor> doctors = new ArrayList<>();
    ArrayList<Patient> patients = new ArrayList<>();
    ArrayList<Appointment> appointments = new ArrayList<>();

    public void registerDoctor(String name, int age, String specialist) {
        Doctor newDoctor = new Doctor(name, age, specialist);
        doctors.add(newDoctor);
        System.out.println("Doctor registered successfully: " + newDoctor.toString());
    }

    public void registerPatient(String name, int age, String gender) {
        Patient newPatient = new Patient(name, age, gender);
        patients.add(newPatient);
        System.out.println("Patient registered successfully: " + newPatient.toString());
    }

    public void displayDoctors() {
        System.out.println(x:"List of Doctors:");
        for (Doctor doctor : doctors) {
            System.out.println(doctor.toString());
        }
    }

    public void displayPatients() {
        System.out.println(x:"List of Patients:");
        for (Patient patient : patients) {
            System.out.println(patient.toString());
        }
    }

    public Doctor searchDoctor(String name) {
        for (Doctor doctor : doctors) {
            if (doctor.name.equalsIgnoreCase(name)) {
                return doctor;
            }
        }
        return null;
    }

    public Patient searchPatient(String name) {
        for (Patient patient : patients) {
            if (patient.name.equalsIgnoreCase(name)) {
                return patient;
            }
        }
        return null;
    }

    public void scheduleAppointment(String patientName, String doctorName, String date) {
        Patient patient = searchPatient(patientName);
        Doctor doctor = searchDoctor(doctorName);

        if (patient != null && doctor != null) {
            Appointment newAppointment = new Appointment(patient, doctor, date);
            appointments.add(newAppointment);
            System.out.println("Appointment scheduled successfully: " + newAppointment.toString());
        } else {
            System.out.println(x:"Patient or Doctor not found. Please check the names.");
        }
    }

    public void displayAppointments() {
        System.out.println(x:"List of Appointments:");
        for (Appointment appointment : appointments) {
            System.out.println(appointment.toString());
            System.out.println(appointment.toString());
        }
    }
}
```

Try Pitch

Hospital.java

Hospital adalah class yang merepresentasikan rumah sakit. Di dalam class ini, terdapat method-method untuk mendaftarkan dokter (registerDoctor), mendaftarkan pasien (registerPatient), menampilkan daftar dokter (displayDoctors), menampilkan daftar pasien (displayPatients), mencari dokter berdasarkan nama (searchDoctor), mencari pasien berdasarkan nama (searchPatient), menjadwalkan janji medis (scheduleAppointment), dan menampilkan daftar janji medis (displayAppointments).

Doctor, Patient, dan Appointment adalah class tambahan yang merepresentasikan dokter, pasien, dan janji medis masing-masing. Di dalam masing-masing class ini, terdapat atribut yang merepresentasikan detail tentang dokter, pasien, atau janji medis tersebut.

Appointment.java

```java
class Appointment {
    Patient patient;
    Doctor doctor;
    String date;

    public Appointment(Patient patient, Doctor doctor, String date) {
        this.patient = patient;
        this.doctor = doctor;
        this.date = date;
    }

    @Override
    public String toString() {
        return "Appointment{" +
                "patient=" + patient +
                ", doctor=" + doctor +
                ", date='" + date + '\'' +
                '}';
    }
}
```

Pemrograman Java dapat digunakan untuk mengembangkan aplikasi manajemen rumah sakit yang mencakup kelas "Appointment". Contoh penerapan konsep ini dapat ditemukan dalam penelitian tentang aplikasi pendaftaran pasien berobat di praktek dokter menggunakan Java. Selain itu, terdapat juga aplikasi manajemen rumah sakit "MyHospital" berbasis Java yang mencakup fitur pendaftaran pasien dan antarmuka untuk memasukkan data pasien ke dalam sistem. Hal ini menunjukkan bahwa pemrograman Java dapat diterapkan dalam pengembangan aplikasi manajemen rumah sakit yang melibatkan kelas "Appointment".

Kelas ini merepresentasikan jadwal pemeriksaan medis antara pasien (Patient) dan dokter (Doctor) pada tanggal tertentu.

```java
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Hospital hospital = new Hospital();

        while (true) {
            System.out.println("\nHospital Management System");
            System.out.println("1. Register Doctor");
            System.out.println("2. Display Doctors");
            System.out.println("3. Search Doctor");
            System.out.println("4. Register Patient");
            System.out.println("5. Display Patients");
            System.out.println("6. Search Patient");
            System.out.println("7. Schedule Appointment");
            System.out.println("8. Display Appointments");
            System.out.println("9. Exit");
            System.out.print("Enter your choice: ");

            int choice = scanner.nextInt();
            scanner.nextLine();

            switch (choice) {
                case 1:
                    System.out.print("Enter doctor name: ");
                    String doctorName = scanner.nextLine();
                    System.out.print("Enter doctor age: ");
                    int doctorAge = scanner.nextInt();
                    scanner.nextLine();
                    System.out.print("Enter specialist of doctor: ");
                    String specialist = scanner.nextLine();

                    hospital.registerDoctor(doctorName, doctorAge, specialist);
                    break;

                case 2:
                    hospital.displayDoctors();
                    break;

                case 3:
                    System.out.print("Enter doctor name to search: ");
                    String searchDoctorName = scanner.nextLine();
                    Doctor foundDoctor = hospital.searchDoctor(searchDoctorName);
                    if (foundDoctor != null) {
                        System.out.println("Doctor found: " + foundDoctor.toString());
                    } else {
                        System.out.println("Doctor not found.");
                    }
                    break;
```

Try Pitch

OUTPUT

```
Hospital Management System
1. Register Doctor
2. Display Doctors
3. Search Doctor
4. Register Patient
5. Display Patients
6. Search Patient
7. Schedule Appointment
8. Display Appointments
9. Exit
Enter your choice: 1
Enter doctor name: James
Enter doctor age: 40
Enter specialist of doctor: epidermis
Doctor registered successfully: Doctor{name='James', age=40, specialist='
epidermis'}

Hospital Management System
1. Register Doctor
2. Display Doctors
3. Search Doctor
4. Register Patient
5. Display Patients
6. Search Patient
7. Schedule Appointment
8. Display Appointments
9. Exit
Enter your choice: 4
Enter patient name: John
Enter patient age: 20
Enter patient gender: male
Patient registered successfully: Patient{name='John', age=20, gender='mal
e'}

Hospital Management System
1. Register Doctor
2. Display Doctors
3. Search Doctor
4. Register Patient
5. Display Patients
6. Search Patient
7. Schedule Appointment
8. Display Appointments
9. Exit
Enter your choice: 7
Enter patient name: john
Enter doctor name: james
Enter appointment date: 20 jan 2024
Appointment scheduled successfully: Appointment{patient=Patient{name='Joh
n', age=20, gender='male'}, doctor=Doctor{name='James', age=40, specialis
t='epidermis'}, date='20 jan 2024'}
```

Main.java

Kode ini adalah sebuah program yang merupakan sistem manajemen rumah sakit. Program ini memiliki beberapa fitur, yaitu:

Register Doctor: Untuk mendaftarkan dokter baru dengan memasukkan nama, umur, dan spesialisasi dokter.

Display Doctors: Untuk menampilkan daftar seluruh dokter yang telah terdaftar.

Search Doctor: Untuk mencari dokter berdasarkan nama.

Register Patient: Untuk mendaftarkan pasien baru dengan memasukkan nama, umur, dan jenis kelamin pasien.

Display Patients: Untuk menampilkan daftar seluruh pasien yang telah terdaftar.

Search Patient: Untuk mencari pasien berdasarkan nama.

Schedule Appointment: Untuk membuat janji temu antara pasien dan dokter tertentu pada tanggal tertentu.

Display Appointments: Untuk menampilkan daftar seluruh janji temu yang telah terbuat.

Exit: Untuk keluar dari program.

Terimakasih

TEAM 3

Pitch

Want to make a presentation like this one?

Start with a fully customizable template, create a beautiful deck in minutes, then easily share it with anyone.

Create a presentation (It's free)