**Coding Challenge: Hospital Management System**

**Name:** Kannan G

**Superset ID:** 5372153

**Create SQL Schema from the following classes class, use the class attributes for table column names**

A computer screen shot of a black screen

Description automatically generated

A computer screen shot of white text

Description automatically generated

**1**. Create the following **model/entity classes** within package **entity** with variables declared private, constructors(default and parametrized,getters,setters and toString())

1. Define **`Patient`** class with the following confidential attributes:

a. patientId

b. firstName

c. lastName;

d. dateOfBirth

e. gender

f. contactNumber

g. address;

2**.** Define **‘Doctor`** class with the following confidential attributes:

a. doctorId

b. firstName

c. lastName

d. specialization

e. contactNumber;

3. **Appointment** Class:

a. appointmentId

b. patientId

c. doctorId

d. appointmentDate

e. description

**2**. Implement the following for all model classes. Write default constructors and overload the

constructor with parameters, getters and setters, method to print all the member variables and

values.

Patient class:

**package** entity;

**import** java.time.LocalDate;

**public** **class** Patient {

**private** **int** patientId;

**private** String firstName;

**private** String lastName;

**private** LocalDate dateOfBirth;

**private** String gender;

**private** String contactNumber;

**private** String address;

**public** Patient() {

// **TODO** Auto-generated constructor stub

**super**();

}

**public** Patient(**int** patientId, String firstName, String lastName, LocalDate dateOfBirth, String gender,

String contactNumber, String address) {

**super**();

**this**.patientId = patientId;

**this**.firstName = firstName;

**this**.lastName = lastName;

**this**.dateOfBirth = dateOfBirth;

**this**.gender = gender;

**this**.contactNumber = contactNumber;

**this**.address = address;

}

**public** **int** getPatientId() {

**return** patientId;

}

**public** **void** setPatientId(**int** patientId) {

**this**.patientId = patientId;

}

**public** String getFirstName() {

**return** firstName;

}

**public** **void** setFirstName(String firstName) {

**this**.firstName = firstName;

}

**public** String getLastName() {

**return** lastName;

}

**public** **void** setLastName(String lastName) {

**this**.lastName = lastName;

}

**public** LocalDate getDateOfBirth() {

**return** dateOfBirth;

}

**public** **void** setDateOfBirth(LocalDate dateOfBirth) {

**this**.dateOfBirth = dateOfBirth;

}

**public** String getGender() {

**return** gender;

}

**public** **void** setGender(String gender) {

**this**.gender = gender;

}

**public** String getContactNumber() {

**return** contactNumber;

}

**public** **void** setContactNumber(String contactNumber) {

**this**.contactNumber = contactNumber;

}

**public** String getAddress() {

**return** address;

}

**public** **void** setAddress(String address) {

**this**.address = address;

}

@Override

**public** String toString() {

**return** "Patient{" +"patientId=" + patientId +", firstName='" + firstName + '\'' +", lastName='" + lastName + '\'' +", "

+ "dateOfBirth=" + dateOfBirth +", gender='" + gender + '\'' +", contactNumber='" + contactNumber + '\'' +

", address='" + address + '\'' +'}';

}

}

Doctor class:

**package** entity;

**public** **class** Doctor {

**private** **int** doctorId;

**private** String firstName;

**private** String lastName;

**private** String specialization;

**private** String contactNumber;

**public** Doctor() {

// **TODO** Auto-generated constructor stub

**super**();

}

**public** Doctor(**int** doctorId, String firstName, String lastName, String specialization, String contactNumber) {

**super**();

**this**.doctorId = doctorId;

**this**.firstName = firstName;

**this**.lastName = lastName;

**this**.specialization = specialization;

**this**.contactNumber = contactNumber;

}

**public** **int** getDoctorId() {

**return** doctorId;

}

**public** **void** setDoctorId(**int** doctorId) {

**this**.doctorId = doctorId;

}

**public** String getFirstName() {

**return** firstName;

}

**public** **void** setFirstName(String firstName) {

**this**.firstName = firstName;

}

**public** String getLastName() {

**return** lastName;

}

**public** **void** setLastName(String lastName) {

**this**.lastName = lastName;

}

**public** String getSpecialization() {

**return** specialization;

}

**public** **void** setSpecialization(String specialization) {

**this**.specialization = specialization;

}

**public** String getContactNumber() {

**return** contactNumber;

}

**public** **void** setContactNumber(String contactNumber) {

**this**.contactNumber = contactNumber;

}

@Override

**public** String toString() {

**return** "Doctor{" +"doctorId=" + doctorId +", firstName='" + firstName + '\'' +", lastName='" + lastName + '\'' +

", specialization='" + specialization + '\'' +", contactNumber='" + contactNumber + '\'' +'}';

}

}

Appointment class:

**package** entity;

**import** java.time.LocalDate;

**public** **class** Appointment {

**private** **int** appointmentId;

**private** **int** patientId;

**private** **int** doctorId;

**private** LocalDate appointmentDate;

**private** String description;

**public** Appointment() {

// **TODO** Auto-generated constructor stub

**super**();

}

**public** Appointment(**int** appointmentId, **int** patientId, **int** doctorId, LocalDate appointmentDate, String description) {

**super**();

**this**.appointmentId = appointmentId;

**this**.patientId = patientId;

**this**.doctorId = doctorId;

**this**.appointmentDate = appointmentDate;

**this**.description = description;

}

**public** **int** getAppointmentId() {

**return** appointmentId;

}

**public** **void** setAppointmentId(**int** appointmentId) {

**this**.appointmentId = appointmentId;

}

**public** **int** getPatientId() {

**return** patientId;

}

**public** **void** setPatientId(**int** patientId) {

**this**.patientId = patientId;

}

**public** **int** getDoctorId() {

**return** doctorId;

}

**public** **void** setDoctorId(**int** doctorId) {

**this**.doctorId = doctorId;

}

**public** LocalDate getAppointmentDate() {

**return** appointmentDate;

}

**public** **void** setAppointmentDate(LocalDate appointmentDate) {

**this**.appointmentDate = appointmentDate;

}

**public** String getDescription() {

**return** description;

}

**public** **void** setDescription(String description) {

**this**.description = description;

}

@Override

**public** String toString() {

**return** "Appointment{" +"appointmentId=" + appointmentId +", patientId=" + patientId +", doctorId=" + doctorId +

", appointmentDate='" + appointmentDate + '\'' +", description='" + description + '\'' +'}';

}

}

**3**. Define IHospitalService interface/abstract class with following methods to interact with

database

Keep the interfaces and implementation classes in package dao

a. getAppointmentById()

i. Parameters: appointmentId

ii. ReturnType: Appointment object

b. getAppointmentsForPatient()

i. Parameters: patientId

ii. ReturnType: List of Appointment objects

c. getAppointmentsForDoctor()

i. Parameters: doctorId

ii. ReturnType: List of Appointment objects

d. scheduleAppointment()

i. Parameters: Appointment Object

ii. ReturnType: Boolean

e. updateAppointment()

i. Parameters: Appointment Object

ii. ReturnType: Boolean

f. cancelAppointment()

i. Parameters: AppointmentId

ii. ReturnType: Boolean

IHospitalService.java:

**package** dao;

**import** java.util.List;

**import** entity.Appointment;

**public** **interface** IHospitalService {

Appointment getAppointmentById(**int** appointmentId);

List<Appointment> getAppointmentsForPatient(**int** patientId);

List<Appointment> getAppointmentsForDoctor(**int** doctorId);

**boolean** scheduleAppointment(Appointment appointment);

**boolean** updateAppointment(Appointment appointment);

**boolean** cancelAppointment(**int** appointmentId);

}

6.Define **HospitalServiceImpl** class and implement all the methods I**HospitalServiceImpl .**

HospitalServiceImpl.java:

**package** dao;

**import** java.sql.Connection;

**import** java.sql.PreparedStatement;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**import** java.util.ArrayList;

**import** java.sql.Date;

**import** java.util.List;

**import** entity.Appointment;

**import** myexceptions.PatientNumberNotFoundException;

**import** util.DBConnection;

**public** **class** HospitalServiceImpl **implements** IHospitalService{

**private** Connection con;

**public** HospitalServiceImpl() {

// **TODO** Auto-generated constructor stub

con = DBConnection.*getConnection*();

}

@Override

**public** Appointment getAppointmentById(**int** appointmentId) {

// **TODO** Auto-generated method stub

Appointment appointment = **null**;

**try** {

PreparedStatement pstmt = con.prepareStatement("select \* from appointment where appointmentId=?");

pstmt.setInt(1, appointmentId);

ResultSet rs = pstmt.executeQuery();

**if**(rs.next()) {

appointment = **new** Appointment(rs.getInt("appointmentId"),rs.getInt("patientId"),rs.getInt("doctorId"),rs.getDate("appointmentDate").toLocalDate(),rs.getString("description"));

}

}

**catch**(SQLException e) {

System.***err***.println("Error getting appointment by id: " + e.getMessage());

}

**return** appointment;

}

@Override

**public** List<Appointment> getAppointmentsForPatient(**int** patientId) **throws** PatientNumberNotFoundException {

// **TODO** Auto-generated method stub

List<Appointment> appointments = **new** ArrayList<>();

**try** {

PreparedStatement pstmt = con.prepareStatement("select \* from appointment where patientId = ?");

pstmt.setInt(1, patientId);

ResultSet rs = pstmt.executeQuery();

**while**(rs.next()) {

appointments.add(**new** Appointment(rs.getInt("appointmentId"),rs.getInt("patientId"),rs.getInt("doctorId"),rs.getDate("appointmentDate").toLocalDate(),rs.getString("description")));

}

**if** (appointments.isEmpty()) {

**throw** **new** PatientNumberNotFoundException("No appointments found for patient ID: " + patientId);

}

}

**catch**(SQLException e) {

System.***err***.println("Error getting appointment for patient: " + e.getMessage());

}

**return** appointments;

}

@Override

**public** List<Appointment> getAppointmentsForDoctor(**int** doctorId) {

// **TODO** Auto-generated method stub

List<Appointment> appointments = **new** ArrayList<>();

**try** {

PreparedStatement pstmt = con.prepareStatement("select \* from appointment where doctorId = ?");

pstmt.setInt(1, doctorId);

ResultSet rs = pstmt.executeQuery();

**while**(rs.next()) {

appointments.add(**new** Appointment(rs.getInt("appointmentId"),rs.getInt("patientId"),rs.getInt("doctorId"),rs.getDate("appointmentDate").toLocalDate(),rs.getString("description")));

}

}

**catch**(SQLException e) {

System.***err***.println("Error getting appointment for doctor: " + e.getMessage());

}

**return** appointments;

}

@Override

**public** **boolean** scheduleAppointment(Appointment appointment) {

// **TODO** Auto-generated method stub

**try** {

PreparedStatement pstmt = con.prepareStatement("insert into appointment (patientId, doctorId, appointmentDate, description) VALUES (?, ?, ?, ?)");

pstmt.setInt(1, appointment.getPatientId());

pstmt.setInt(2, appointment.getDoctorId());

pstmt.setDate(3, Date.*valueOf*(appointment.getAppointmentDate()));

pstmt.setString(4, appointment.getDescription());

pstmt.executeUpdate();

System.***out***.println("Appointment scheduled successfully.");

**return** **true**;

}

**catch**(SQLException e) {

System.***err***.println("Error scheduling appointment: " + e.getMessage());

}

**return** **false**;

}

@Override

**public** **boolean** updateAppointment(Appointment appointment) {

// **TODO** Auto-generated method stub

**try** {

PreparedStatement pstmt = con.prepareStatement("update appointment set patientId = ?, doctorId = ?, appointmentDate = ?, description = ? where appointmentId = ?");

pstmt.setInt(1, appointment.getPatientId());

pstmt.setInt(2, appointment.getDoctorId());

pstmt.setDate(3, Date.*valueOf*(appointment.getAppointmentDate()));

pstmt.setString(4, appointment.getDescription());

pstmt.setInt(5, appointment.getAppointmentId());

**int** rows = pstmt.executeUpdate();

**if** (rows > 0) {

//System.out.println("Appointment updated successfully.");

**return** **true**;

}

}

**catch** (SQLException e) {

System.***err***.println("Error updating appointment: " + e.getMessage());

}

**return** **false**;

}

@Override

**public** **boolean** cancelAppointment(**int** appointmentId) {

// **TODO** Auto-generated method stub

**try** {

PreparedStatement pstmt = con.prepareStatement("delete from appointment where appointmentId = ?");

pstmt.setInt(1, appointmentId);

**int** rows = pstmt.executeUpdate();

**if** (rows > 0) {

//System.out.println("Appointment cancelled successfully.");

**return** **true**;

}

}

**catch** (SQLException e) {

System.***err***.println("Error cancelling appointment: " + e.getMessage());

}

**return** **false**;

}

}

**7**. Create a utility class **DBConnection** in a package **util** with a static variable **connection** of Type **Connection** and a static method **getConnection()** which returns connection.

Connection properties supplied in the connection string should be read from a property file.

Create a utility class **PropertyUtil** which contains a static method named **getPropertyString()** which reads a property fie containing connection details like hostname, dbname, username, password, port number and returns a connection string.

PropertyUtil.java:

**package** util;

**import** java.io.FileInputStream;

**import** java.io.FileNotFoundException;

**import** java.io.IOException;

**import** java.util.Properties;

**public** **class** PropertyUtil {

**public** **static** String getPropertyString(String fileName) **throws** IOException {

String connStr = **null**;

Properties props = **new** Properties();

FileInputStream fis = **new** FileInputStream(fileName);

props.load(fis);

String protocol = props.getProperty("protocol");

String hostname = props.getProperty("hostname");

String port = props.getProperty("port");

String dbname = props.getProperty("dbname");

String user = props.getProperty("user");

String password = props.getProperty("password");

connStr = protocol+"//"+hostname+":"+port+"/"+dbname+"?user="+user+"&password="+password;

**return** connStr;

}

}

DBConnection.java:

**package** util;

**import** java.io.IOException;

**import** java.sql.Connection;

**import** java.sql.DriverManager;

**import** java.sql.SQLException;

**public** **class** DBConnection {

**private** **static** **final** String ***fileName*** = "db.properties";

**private** **static** Connection *connection* = **null**;

**public** **static** Connection getConnection() {

String connString = **null**;

**try** {

connString = PropertyUtil.*getPropertyString*(***fileName***);

}

**catch**(IOException e) {

System.***out***.println("Connection string creation failed");

e.printStackTrace();

}

**if**(connString != **null**) {

**try** {

*connection* = DriverManager.*getConnection*(connString);

}

**catch**(SQLException e) {

System.***out***.println("Error while establishisng DBConnection...");

e.printStackTrace();

}

}

**return** *connection*;

}

}

**8**. Create the exceptions in package myexceptions

Define the following custom exceptions and throw them in methods whenever needed. Handle all the exceptions in main method,

1. **PatientNumberNotFoundException** :throw this exception when user enters an invalid patient number which doesn’t exist in db.

PatientNumberNotFoundException**.**java:

**package** myexceptions;

**public** **class** PatientNumberNotFoundException **extends** Exception{

**public** PatientNumberNotFoundException(String message) {

// **TODO** Auto-generated constructor stub

**super**(message);

}

}

HospitalServiceImpl.java:

@Override

**public** List<Appointment> getAppointmentsForPatient(**int** patientId) **throws** PatientNumberNotFoundException {

// **TODO** Auto-generated method stub

List<Appointment> appointments = **new** ArrayList<>();

**try** {

PreparedStatement pstmt = con.prepareStatement("select \* from appointment where patientId = ?");

pstmt.setInt(1, patientId);

ResultSet rs = pstmt.executeQuery();

**while**(rs.next()) {

appointments.add(**new** Appointment(rs.getInt("appointmentId"),rs.getInt("patientId"),rs.getInt("doctorId"),rs.getDate("appointmentDate").toLocalDate(),rs.getString("description")));

}

**if** (appointments.isEmpty()) {

**throw** **new** PatientNumberNotFoundException("No appointments found for patient ID: " + patientId);

}

}

**catch**(SQLException e) {

System.***err***.println("Error getting appointment for patient: " + e.getMessage());

}

**return** appointments;

}

**9.** Create class named MainModule with main method in package mainmod.

Trigger all the methods in service implementation class.

MainModule.java:

package main;

import java.sql.Connection;

import java.time.LocalDate;

import java.util.InputMismatchException;

import java.util.List;

import java.util.Scanner;

import dao.HospitalServiceImpl;

import entity.Appointment;

import myexceptions.PatientNumberNotFoundException;

import util.DBConnection;

public class MainModule {

public static void main(String[] args) {

// TODO Auto-generated method stub

Scanner sc = new Scanner(System.in);

HospitalServiceImpl service = new HospitalServiceImpl();

while(true) {

System.out.println("---Hospital Management System---");

System.out.println("1. Schedule Appointment");

System.out.println("2. Get Appointment By ID");

System.out.println("3. Get Appointments For Patient");

System.out.println("4. Get Appointments For Doctor");

System.out.println("5. Update Appointment");

System.out.println("6. Cancel Appointment");

System.out.println("7. Exit");

System.out.print("Provide Option :\t");

int choice = sc.nextInt();

switch(choice) {

case 1 ->{

Appointment appointment = new Appointment();

System.out.print("Enter Patient ID: ");

appointment.setPatientId(sc.nextInt());

System.out.print("Enter Doctor ID: ");

appointment.setDoctorId(sc.nextInt());

sc.nextLine();

System.out.print("Enter Appointment Date (yyyy-mm-dd): ");

appointment.setAppointmentDate(LocalDate.parse(sc.nextLine()));

System.out.print("Enter Description: ");

appointment.setDescription(sc.nextLine());

if (service.scheduleAppointment(appointment))

System.out.println("Appointment scheduled successfully.");

else

System.out.println("Failed to schedule appointment.");

}

case 2 -> {

try {

System.out.print("Enter Appointment ID to find: ");

int id = sc.nextInt();

Appointment app = service.getAppointmentById(id);

if (app != null)

System.out.println(app);

else

System.out.println("Appointment not found.");

}

catch (InputMismatchException ime) {

System.out.println("Invalid input for appointment ID.");

}

}

case 3 -> {

try {

System.out.print("Enter Patient ID: ");

int patId = sc.nextInt();

List<Appointment> appointments = service.getAppointmentsForPatient(patId);

for (Appointment a : appointments) {

System.out.println(a);

}

}

catch (PatientNumberNotFoundException e) {

System.err.println("Patient number not found: " + e.getMessage());

}

catch (InputMismatchException ime) {

System.err.println("Invalid input for patient ID.");

}

}

case 4 -> {

System.out.print("Enter Doctor ID: ");

int docId = sc.nextInt();

List<Appointment> appointments = service.getAppointmentsForDoctor(docId);

for (Appointment a : appointments)

{

System.out.println(a);

}

}

case 5 -> {

Appointment updatedAppointment = new Appointment();

System.out.print("Enter Appointment ID to update: ");

updatedAppointment.setAppointmentId(sc.nextInt());

System.out.print("Enter new Patient ID: ");

updatedAppointment.setPatientId(sc.nextInt());

System.out.print("Enter new Doctor ID: ");

updatedAppointment.setDoctorId(sc.nextInt());

sc.nextLine();

System.out.print("Enter new Appointment Date (yyyy-mm-dd): ");

updatedAppointment.setAppointmentDate(LocalDate.parse(sc.nextLine()));

System.out.print("Enter new Description: ");

updatedAppointment.setDescription(sc.nextLine());

if (service.updateAppointment(updatedAppointment)) {

System.out.println("Appointment updated successfully.");

}

else

System.out.println("Failed to update appointment.");

}

case 6 -> {

System.out.print("Enter Appointment ID to cancel: ");

int cancelId = sc.nextInt();

if (service.cancelAppointment(cancelId)) {

System.out.println("Appointment cancelled successfully.");

}

else

System.out.println("Failed to cancel appointment.");

}

case 7 -> {

System.out.println("Exiting from Hospital Management System...");

sc.close();

System.exit(0);

}

default -> throw new IllegalArgumentException("Invalid Option: " + choice);

}

}

}

}

**Sample Output:**

**A screen shot of a computer program

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A white background with black text

Description automatically generated**

**A close-up of a computer screen

Description automatically generated**

**A close-up of a computer screen

Description automatically generated**

**A screenshot of a computer program

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screen shot of a computer program

Description automatically generated**