Hospital Management System

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

create database hospital_management;

use hospital_management;

```
CREATE TABLE Patient (
    patientId INT PRIMARY KEY,
    firstName VARCHAR(255),
    lastName VARCHAR(255),
    dateOfBirth DATE,
    gender VARCHAR(10),
    contactNumber VARCHAR(20),
    address VARCHAR(255)
);
```

```
mysql> select * from patient
    -> ;
+-----------+-------------+-----------+-------------+--------+---------------+--------------------+
| patientId | firstName   | lastName  | dateOfBirth | gender | contactNumber | address            |
+-----------+-------------+-----------+-------------+--------+---------------+--------------------+
|         1 | John        | Doe       | 1990-05-15  | Male   | 1234567890    | 123 Main St        |
|         2 | Jane        | Smith     | 1985-08-20  | Female | 9876543210    | 456 Elm St         |
|         3 | Michael     | Johnson   | 1978-12-10  | Male   | 5555555555    | 789 Oak St         |
|         4 | Emily       | Brown     | 1995-03-25  | Female | 4443332221    | 321 Maple Ave      |
|         5 | David       | Taylor    | 1982-06-18  | Male   | 7778889990    | 567 Pine St        |
|         6 | Sarah       | Wilson    | 1998-09-30  | Female | 1112223334    | 890 Cedar St       |
|         7 | Christopher | Martinez  | 1970-02-05  | Male   | 9998887776    | 234 Birch St       |
|         8 | Jessica     | Lee       | 1989-11-12  | Female | 3332221118    | 678 Spruce St      |
|         9 | Brian       | Garcia    | 1992-07-22  | Male   | 6665554447    | 901 Oakwood Ave    |
|        10 | Amanda      | Rodriguez | 1984-04-08  | Female | 8889990001    | 543 Pinecrest Blvd |
+-----------+-------------+-----------+-------------+--------+---------------+--------------------+
10 rows in set (0.00 sec)

mysql>
```

```
CREATE TABLE Doctor (
    doctorId INT PRIMARY KEY,
    firstName VARCHAR(255),
    lastName VARCHAR(255),
    specialization VARCHAR(255),
    contactNumber VARCHAR(20)
);
```

```
mysql> select * from doctor;
+----------+-------------+----------+--------------------+---------------+
| doctorId | firstName   | lastName | specialization     | contactNumber |
+----------+-------------+----------+--------------------+---------------+
|        1 | Dr. James   | Smith    | Cardiologist       | 1111111111    |
|        2 | Dr. Lisa    | Johnson  | Pediatrician       | 2222222222    |
|        3 | Dr. Robert  | Williams | Dermatologist      | 3333333333    |
|        4 | Dr. Jennifer| Brown    | Orthopedic Surgeon | 4444444444    |
|        5 | Dr. David   | Jones    | Neurologist        | 5555555555    |
+----------+-------------+----------+--------------------+---------------+
5 rows in set (0.00 sec)

mysql>
```

CREATE TABLE Appointment (

appointmentId INT PRIMARY KEY,

patientId INT,

doctorId INT,

appointmentDate DATE,

description TEXT,

FOREIGN KEY (patientId) REFERENCES Patient(patientId),

FOREIGN KEY (doctorId) REFERENCES Doctor(doctorId)

);

```
mysql> select * from appointment;
+---------------+-----------+----------+-----------------+---------------------------+
| appointmentId | patientId | doctorId | appointmentDate | description               |
+---------------+-----------+----------+-----------------+---------------------------+
|             1 |         1 |        2 | 2024-06-10      | Regular checkup           |
|             2 |         2 |        4 | 2024-06-12      | Knee pain treatment       |
|             3 |         3 |        1 | 2024-06-15      | Heart examination         |
|             4 |         4 |        3 | 2024-06-18      | Skin allergy treatment    |
|             5 |         5 |        5 | 2024-06-20      | Headache consultation     |
|             6 |         6 |        2 | 2024-06-22      | Eye examination           |
|             7 |         7 |        1 | 2024-06-25      | Diabetes treatment        |
|             8 |         8 |        3 | 2024-06-28      | Stomach pain consultation |
|             9 |         9 |        4 | 2024-06-30      | Cancer screening          |
|            10 |        10 |        5 | 2024-07-02      | Skin cancer checkup       |
+---------------+-----------+----------+-----------------+---------------------------+
10 rows in set (0.00 sec)

mysql>
```

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.

Pakage:entity

→patient.py

→doctor.py

→appointment.py

Doctor Class:

```
class Doctor:
    def __init__(self, doctor_id=None, first_name=None, last_name=None, specialization=None,
contact_number=None):
        self.doctor_id = doctor_id
        self.first_name = first_name
        self.last_name = last_name
        self.specialization = specialization
        self.contact_number = contact_number

    # Getters
    def get_doctor_id(self):
        return self.doctor_id

    def get_first_name(self):
        return self.first_name

    def get_last_name(self):
        return self.last_name

    def get_specialization(self):
        return self.specialization

    def get_contact_number(self):
        return self.contact_number

    # Setters
    def set_doctor_id(self, doctor_id):
        self.doctor_id = doctor_id

    def set_first_name(self, first_name):
        self.first_name = first_name

    def set_last_name(self, last_name):
        self.last_name = last_name

    def set_specialization(self, specialization):
        self.specialization = specialization

    def set_contact_number(self, contact_number):
        self.contact_number = contact_number
```

Patient Class:

```
class Patient:
    def __init__(self, patient_id=None, first_name=None, last_name=None, date_of_birth=None, gender=None,
                 contact_number=None, address=None):
        self.patient_id = patient_id
        self.first_name = first_name
        self.last_name = last_name
        self.date_of_birth = date_of_birth
        self.gender = gender
        self.contact_number = contact_number
        self.address = address
```

```
    # Getters
    def get_patient_id(self):
        return self.patient_id

    def get_first_name(self):
        return self.first_name

    def get_last_name(self):
        return self.last_name

    def get_date_of_birth(self):
        return self.date_of_birth

    def get_gender(self):
        return self.gender

    def get_contact_number(self):
        return self.contact_number

    def get_address(self):
        return self.address

    # Setters
    def set_patient_id(self, patient_id):
        self.patient_id = patient_id

    def set_first_name(self, first_name):
        self.first_name = first_name

    def set_last_name(self, last_name):
        self.last_name = last_name

    def set_date_of_birth(self, date_of_birth):
        self.date_of_birth = date_of_birth

    def set_gender(self, gender):
        self.gender = gender

    def set_contact_number(self, contact_number):
        self.contact_number = contact_number

    def set_address(self, address):
        self.address = address
```

Appointment Class:

```
# Appointment.py
class Appointment:
    def __init__(self, appointment_id=None, patient_id=None, doctor_id=None, appointment_date=None,
description=None):
        self.appointment_id = appointment_id
        self.patient_id = patient_id
        self.doctor_id = doctor_id
        self.appointment_date = appointment_date
        self.description = description
```

```
    # Getters
    def get_appointment_id(self):
        return self.appointment_id

    def get_patient_id(self):
        return self.patient_id

    def get_doctor_id(self):
        return self.doctor_id

    def get_appointment_date(self):
        return self.appointment_date

    def get_description(self):
        return self.description

    # Setters
    def set_appointment_id(self, appointment_id):
        self.appointment_id = appointment_id

    def set_patient_id(self, patient_id):
        self.patient_id = patient_id

    def set_doctor_id(self, doctor_id):
        self.doctor_id = doctor_id

    def set_appointment_date(self, appointment_date):
        self.appointment_date = appointment_date

    def set_description(self, description):
        self.description = description
```

3. Define IHospitalService interface/abstract class with following methods to interact with database .Keep the interfaces and implementation classes in package dao

package:doa

→IHospitalService.py

→ IHospitalServiceImpl.py

Interface:

```
# IHospitalService.py
from abc import ABC, abstractmethod


class IHospitalService(ABC):

    @abstractmethod
    def getAppointmentById(self, appointmentId):
        pass

    @abstractmethod
    def getAppointmentsForPatient(self, patientId):
        pass

    @abstractmethod
```

```
    def getAppointmentsForDoctor(self, doctorId):
        pass

    @abstractmethod
    def scheduleAppointment(self, appointment):
        pass

    @abstractmethod
    def updateAppointment(self, appointment):
        pass

    @abstractmethod
    def cancelAppointment(self, appointmentId):
        pass
```

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

```
from dao.IHospitalService import IHospitalService
from util.DBConnection import DBConnection
from entity.appointment import Appointment

# IHospitalServiceImpl
class IHospitalServiceImpl(IHospitalService):
    def getAppointmentById(self, appointmentId):
        con=DBConnection.getConnection()
        cursor=con.cursor()

        query="select * from appointment where appointmentId=%s"
        cursor.execute(query,(appointmentId,))
        rows=cursor.fetchall()
        for data in rows:
            print(data)

        con.close()


# getAppointmentsForPatient
    def getAppointmentsForPatient(self, patientId):
        con = DBConnection.getConnection()
        cursor = con.cursor()

        query="select * from appointment where patientId=%s"
        cursor.execute(query,(patientId,))
        rows = cursor.fetchall()
        for data in rows:
            print(data)

        con.close()

# getAppointmentsForDoctor
    def getAppointmentsForDoctor(self, doctorId):
        con = DBConnection.getConnection()
        cursor = con.cursor()
        query = "select * from appointment where doctorId=%s"
        cursor.execute(query, (doctorId,))
        rows = cursor.fetchall()
```

```
        for data in rows:
            print(data)

        con.close()
```

 # scheduleAppointment

```
    def scheduleAppointment(self, appointment):
        con = DBConnection.getConnection()
        cursor = con.cursor()

        query="INSERT INTO appointment (appointmentId, patientId, doctorId, appointmentDate, description)
VALUES (%s, %s, %s, %s, %s)"
        cursor.execute(query,
                    (appointment.appointment_id,appointment.patient_id,appointment.doctor_id,
                     appointment.appointment_date,appointment.description))
        cursor.execute("select * from appointment")
        rows = cursor.fetchall()
        for data in rows:
            print(data)

        con.close()
```

 # updateAppointment

```
    def updateAppointment(self, appointment):
        con = DBConnection.getConnection()
        cursor = con.cursor()

        query = "update appointment set patientId = %s, doctorId = %s, appointmentDate = %s, description = %s
where appointmentId = %s"
        cursor.execute(query,
                    (appointment.patientId, appointment.doctorId, appointment.appointmentDate,
appointment.description,
                    appointment.appointmentId))
        print("updated successfully")

        con.commit()
        con.close()
```

#cancelAppointment

```
    def cancelAppointment(self, appointmentId):
        con = DBConnection.getConnection()
        cursor = con.cursor()

        query="delete * from appointment where appointmentid=%s"
        cursor.execute(query,(appointmentId,))
        print("deleted successfully")

        con.close()
```

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.

Pakage:util

→DBConnection.py

```
import mysql.connector

class DBConnection:
    @staticmethod
    def getConnection():
        # property=PropertyUtil.getPropertyString()
        conn = mysql.connector.connect(
            host="localhost",
            user="root",
            password="Vaishu@28",
            database="hospital_management"
        )
        return conn
```

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

Pakage:exceptions

→exception.py

Exception.py

```
class PatientNotFound(Exception):
    pass
```

In mail function:

```
patient_id = int(input("Enter patient ID: "))
try:
    service.getAppointmentsForPatient(patient_id)
except PatientNotFound as e:
    print(e)
```

9. Create class named MainModule with main method in package mainmod.
Trigger all the methods in service implementation class

```
from dao.IHospitalServiceImpl import IHospitalServiceImpl
from exceptions.exception import PatientNotFound
from entity.appointment import Appointment

class MainModule:
    @staticmethod
    def main():
        service = IHospitalServiceImpl()

        while True:
            print("\nMenu:")
            print("1. Find appointment by ID")
            print("2. Find appointments for patient")
            print("3. Find appointments for doctor")
            print("4. Schedule appointment")
```

```
        print("5. Update appointment")
        print("6. Cancel appointment")
        print("7. Exit")

        choice = input("Enter your choice: ")

        if choice == '1':
            appointment_id = int(input("Enter appointment ID: "))
            service.getAppointmentById(appointment_id)

        elif choice == '2':
            patient_id = int(input("Enter patient ID: "))
            try:
                service.getAppointmentsForPatient(patient_id)
            except PatientNotFound as e:
                print(e)

        elif choice == '3':
            doctor_id = int(input("Enter doctor ID: "))
            service.getAppointmentsForDoctor(doctor_id)

        elif choice == '4':
            appointment_id=int(input("Enter appointmentId:"))
            patient_id=int(input("Enter patientId:"))
            doctor_id=int(input("Enter doctorId:"))
            date=input("Enter date(yyyy-mm-dd):")
            desc=input("Enter description:")

            appointment = Appointment(appointment_id,patient_id,doctor_id,date,desc)
            service.scheduleAppointment(appointment)

        elif choice == '5':
            appointment_id=int(input("Enter appointmentId to be updated:"))
            patient_id = int(input("Enter patientId:"))
            doctor_id = int(input("Enter doctorId:"))
            date = input("Enter date(yyyy-mm-dd):")
            desc = input("Enter description:")

            updated_appointment = Appointment(appointment_id,patient_id,doctor_id,date,desc)
            service.updateAppointment(updated_appointment)
        elif choice == '6':
            appointment_id_to_cancel = int(input("Enter appointment ID to cancel: "))
            service.cancelAppointment(appointment_id_to_cancel)
        elif choice == '7':
            print("Exiting...")
            break
        else:
            print("Invalid choice. Please enter a number between 1 and 7.")

if __name__ == "__main__":
    MainModule.main()
```

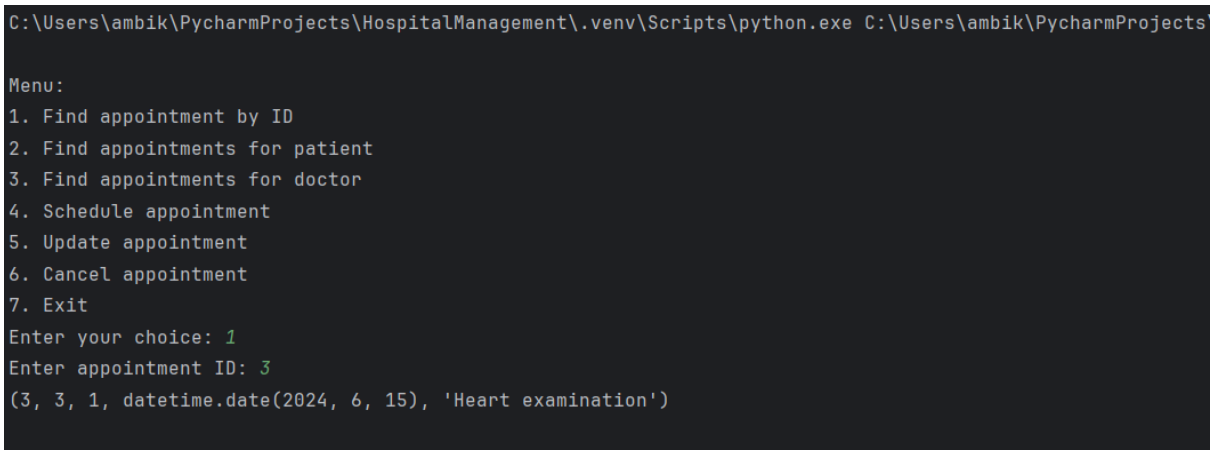
Output for ServiceFunction:

1. getAppointmentById()

```
def getAppointmentById(self, appointmentId):
    con=DBConnection.getConnection()
    cursor=con.cursor()

    query="select * from appointment where appointmentId=%s"
    cursor.execute(query,(appointmentId,))
    rows=cursor.fetchall()
    for data in rows:
        print(data)

    con.close()
```



2. getAppointmentsForPatient()

```
def getAppointmentsForPatient(self, patientId):
    con = DBConnection.getConnection()
    cursor = con.cursor()

    query="select * from appointment where patientId=%s"
    cursor.execute(query,(patientId,))
    rows = cursor.fetchall()
    for data in rows:
        print(data)

    con.close()
```

```
C:\Users\ambik\PycharmProjects\HospitalManagement\.venv\Scripts\python.exe C:\Users\ambik\PycharmProjects\

Menu:
1. Find appointment by ID
2. Find appointments for patient
3. Find appointments for doctor
4. Schedule appointment
5. Update appointment
6. Cancel appointment
7. Exit
Enter your choice: 2
Enter patient ID: 4
(4, 4, 3, datetime.date(2024, 6, 18), 'Skin allergy treatment')
```

3. getAppointmentsForDoctor()

```
def getAppointmentsForDoctor(self, doctorId):
    con = DBConnection.getConnection()
    cursor = con.cursor()
    query = "select * from appointment where doctorId=%s"
    cursor.execute(query, (doctorId,))
    rows = cursor.fetchall()
    for data in rows:
        print(data)

    con.close()
```

```
C:\Users\ambik\PycharmProjects\HospitalManagement\.venv\Scripts\python.exe C:\Users\ambik\PycharmProjects\

Menu:
1. Find appointment by ID
2. Find appointments for patient
3. Find appointments for doctor
4. Schedule appointment
5. Update appointment
6. Cancel appointment
7. Exit
Enter your choice: 3
Enter doctor ID: 4
(2, 2, 4, datetime.date(2024, 6, 12), 'Knee pain treatment')
(9, 9, 4, datetime.date(2024, 6, 30), 'Cancer screening')
```

4. scheduleAppointment()

```
def scheduleAppointment(self, appointment):
    con = DBConnection.getConnection()
    cursor = con.cursor()

    query="INSERT INTO appointment (appointmentId, patientId, doctorId, appointmentDate, description)
VALUES (%s, %s, %s, %s, %s)"
    cursor.execute(query,
                   (appointment.appointment_id,appointment.patient_id,appointment.doctor_id,
                    appointment.appointment_date,appointment.description))
    print("inserted successfully")

    con.close()
```

```
C:\Users\ambik\PycharmProjects\HospitalManagement\.venv\Scripts\python.exe C:\Users\ambik\PycharmProjects\

Menu:
1. Find appointment by ID
2. Find appointments for patient
3. Find appointments for doctor
4. Schedule appointment
5. Update appointment
6. Cancel appointment
7. Exit
Enter your choice: 4
Enter appointmentId:14
Enter patientId:6
Enter doctorId:3
Enter date(yyyy-mm-dd):2025-01-27
Enter description:health checkup
inserted successfully
```

5. updateAppointment()

```
def updateAppointment(self, appointment):
    con = DBConnection.getConnection()
    cursor = con.cursor()

    query = "update appointment set patientId = %s, doctorId = %s, appointmentDate = %s, description = %s
where appointmentId = %s"
    cursor.execute(query,
                   (appointment.patient_id, appointment.doctor_id, appointment.appointment_date,
appointment.description,
                   appointment.appointment_id))
    print("updated successfully")

    con.commit()
    con.close()
```

```
C:\Users\ambik\PycharmProjects\HospitalManagement\.venv\Scripts\python.exe C:\Users\ambik\PycharmProjects\

Menu:
1. Find appointment by ID
2. Find appointments for patient
3. Find appointments for doctor
4. Schedule appointment
5. Update appointment
6. Cancel appointment
7. Exit
Enter your choice: 5
Enter appointmentId to be updated:2
Enter patientId:4
Enter doctorId:5
Enter date(yyyy-mm-dd):2024-12-01
Enter description:knee checkup
updated successfully
```

6. CancelAppointment()

```
def cancelAppointment(self, appointmentId):
    con = DBConnection.getConnection()
    cursor = con.cursor()

    query="delete from appointment where appointmentid=%s"
    cursor.execute(query,(appointmentId,))
    print("deleted successfully")

    con.close()
```

```
C:\Users\ambik\PycharmProjects\HospitalManagement\.venv\Scripts\python.exe C:\Users\ambik\PycharmProjects\

Menu:
1. Find appointment by ID
2. Find appointments for patient
3. Find appointments for doctor
4. Schedule appointment
5. Update appointment
6. Cancel appointment
7. Exit
Enter your choice: 6
Enter appointment ID to cancel: 2
deleted successfully
```