HOSPITAL MANAGEMENT – PYTHON CODING CHALLENGE:

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

```
CREATE DATABASE Hospital;
USE Hospital;

CREATE TABLE Patient (
    patientId INT PRIMARY KEY IDENTITY,
    firstName VARCHAR(255),
    lastName VARCHAR(255),
    dateOfBirth DATE,
    gender VARCHAR(1),
    contactNumber VARCHAR(15),
    address VARCHAR(255)
);

CREATE TABLE Doctor (
    doctorId INT PRIMARY KEY IDENTITY,
    firstName VARCHAR(255),
    lastName VARCHAR(255),
    specialization VARCHAR(255),
    contactNumber VARCHAR(15)
);

CREATE TABLE Appointment (
    appointmentId INT PRIMARY KEY IDENTITY,
    patientId INT,
    doctorId INT,
    appointmentDate DATETIME,
    description TEXT,
    FOREIGN KEY (patientId) REFERENCES Patient(patientId),
    FOREIGN KEY (doctorId) REFERENCES Doctor(doctorId)
);
SET IDENTITY_INSERT patient ON;

INSERT INTO Patient (patientId, firstName, lastName, dateOfBirth, gender, contactNumber, address)
VALUES
    (1, 'John', 'Doe', '1980-01-01', 'M', '1234567890', '123 Main St'),
    (2, 'Jane', 'Smith', '1985-05-15', 'F', '9876543210', '456 Elm St'),
    (3, 'Michael', 'Johnson', '1976-09-22', 'M', '5551234567', '789 Oak Ave'),
    (4, 'Emily', 'Brown', '1990-03-10', 'F', '5559876543', '101 Maple St'),
    (5, 'William', 'Wilson', '1982-07-08', 'M', '5552223333', '222 Pine St'),
    (6, 'Amanda', 'Taylor', '1988-11-30', 'F', '5554445555', '333 Cedar St'),
    (7, 'James', 'Anderson', '1975-04-18', 'M', '5556667777', '444 Birch St'),
    (8, 'Sarah', 'Martinez', '1995-06-25', 'F', '5558889999', '555 Willow St'),
    (9, 'Matthew', 'Hernandez', '1983-02-14', 'M', '5551112222', '666 Oak St'),
    (10, 'Jennifer', 'Garcia', '1978-08-20', 'F', '5553334444', '777 Elm St');
    SET IDENTITY_INSERT patient OFF;

SET IDENTITY_INSERT Doctor ON;

INSERT INTO Doctor (doctorId, firstName, lastName, specialization, contactNumber)
VALUES
    (1, 'Dr. Smith', 'Smithson', 'Cardiologist', '5551234567'),
    (2, 'Dr. Johnson', 'Johnsonson', 'Neurologist', '5559876543'),
    (3, 'Dr. Williams', 'Williamson', 'Dermatologist', '5551112222'),
    (4, 'Dr. Brown', 'Browner', 'Endocrinologist', '5553334444'),
    (5, 'Dr. Jones', 'Joneson', 'Gastroenterologist', '5555556666'),
    (6, 'Dr. Davis', 'Davison', 'Hematologist', '5557778888'),
    (7, 'Dr. Miller', 'Millerson', 'Nephrologist', '5559990000'),
    (8, 'Dr. Wilson', 'Wilsonson', 'Oncologist', '5552223333'),
    (9, 'Dr. Moore', 'Moorer', 'Ophthalmologist', '5554445555'),
    (10, 'Dr. Taylor', 'Taylorson', 'Orthopedic Surgeon', '5556667777');
    SET IDENTITY_INSERT Doctor OFF;

SET IDENTITY_INSERT Appointment ON;
```

```sql
INSERT INTO Appointment (appointmentId, patientId, doctorId, appointmentDate, description)
VALUES
    (1, 1, 1, '2024-03-20 09:00:00', 'Routine checkup'),
    (2, 2, 2, '2024-03-21 10:00:00', 'Follow-up appointment'),
    (3, 3, 3, '2024-03-22 11:00:00', 'Annual physical exam'),
    (4, 4, 4, '2024-03-23 12:00:00', 'Diabetes consultation'),
    (5, 5, 5, '2024-03-24 13:00:00', 'Colonoscopy'),
    (6, 6, 6, '2024-03-25 14:00:00', 'Blood transfusion'),
    (7, 7, 7, '2024-03-26 15:00:00', 'Kidney biopsy'),
    (8, 8, 8, '2024-03-27 16:00:00', 'Chemotherapy session'),
    (9, 9, 9, '2024-03-28 17:00:00', 'Eye surgery'),
    (10, 10, 10, '2024-03-29 18:00:00', 'Knee replacement');
SET IDENTITY_INSERT Appointment OFF;
```

Define `Patient` class with the following confidential attributes:

```
class Patient:

  def __init__(self, patientId=None, firstName=None, lastName=None, dateOfBirth=None, gender=None, contactNumber=None, address=None):

    self.patientId = patientId

    self.firstName = firstName

    self.lastName = lastName

    self.dateOfBirth = dateOfBirth

    self.gender = gender

    self.contactNumber = contactNumber

    self.address = address

  def getPatientId(self):

    return self.patientId

  def setPatientId(self, patientId):

    self.patientId = patientId


  def getFirstName(self):

    return self.firstName

  def setFirstName(self, firstName):

    self.firstName = firstName

  def getLastName(self):

    return self.lastName

  def setLastName(self, lastName):
```

```
        self.lastName = lastName
    def getDateOfBirth(self):
        return self.dateOfBirth
    def setDateOfBirth(self, dateOfBirth):
        self.dateOfBirth = dateOfBirth
    def getGender(self):
        return self.gender
    def setGender(self, gender):
        self.gender = gender
    def getContactNumber(self):
        return self.contactNumber
    def setContactNumber(self, contactNumber):
        self.contactNumber = contactNumber
    def getAddress(self):
        return self.address
    def setAddress(self, address):
        self.address = address
    def __str__(self):
return f"Patient ID: {self.patientId}, Name: {self.firstName} {self.lastName}, DOB:
{self.dateOfBirth}, Gender: {self.gender}, Contact: {self.contactNumber}, Address:
{self.address}"
```

Define 'Doctor` class with the following confidential attributes:

```
class Doctor:
    def __init__(self, doctorId=None, firstName=None, lastName=None,
specialization=None, contactNumber=None):
        self.doctorId = doctorId
        self.firstName = firstName
        self.lastName = lastName
        self.specialization = specialization
```

```
        self.contactNumber = contactNumber
    def getDoctorId(self):
        return self.doctorId
    def setDoctorId(self, doctorId):
        self.doctorId = doctorId
    def getFirstName(self):
        return self.firstName
    def setFirstName(self, firstName):
        self.firstName = firstName
    def getLastName(self):
        return self.lastName
    def setLastName(self, lastName):
        self.lastName = lastName
    def getSpecialization(self):
        return self.specialization
    def setSpecialization(self, specialization):
        self.specialization = specialization
    def getContactNumber(self):
        return self.__contactNumber
    def setContactNumber(self, contactNumber):
        self.contactNumber = contactNumber
    def __str__(self):
        return f"Doctor ID: {self.doctorId}, Name: {self.firstName} {self.lastName},
Specialization: {self.specialization}, Contact: {self.contactNumber}"
```

Appointment Class:

```
class Appointment:
```

```
    def __init__(self, appointmentId, patientId, doctorId, appointmentDate,
description):
        self.appointmentId = appointmentId
        self.patientId = patientId
        self.doctorId = doctorId
        self.appointmentDate = appointmentDate
        self.description = description
    def get_appointment_id(self):
        return self.appointmentId
    def set_appointment_id(self, appointmentId):
        self.__appointment_id = appointmentId


    def get_patient_id(self):
        return self.patientId
    def set_patient_id(self, patientId):
        self.patientId = patientId
    def get_doctor_id(self):
        return self.doctorId
    def set_doctor_id(self, doctorId):
        self.doctorId = doctorId
    def get_appointment_date(self):
        return self.appointmentDate
    def set_appointment_date(self, appointmentDate):
        self.appointmentDate = appointmentDate
    def get_description(self):
        return self.description
    def set_description(self, description):
```

```
        self.__description = description
    def __str__(self):
        return f"Appointment ID: {self.appointmentId}, Patient ID: {self.patientId}, Doctor ID: {self.doctorId}, Date: {self.appointmentDate}, Description: {self.description}"
```

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

HOSPITAL_SERVICEIMPL.

```
import pyodbc
import sys
import os
# Add parent directory of 'dao' to Python path
sys.path.append(os.path.abspath(os.path.join(os.path.dirname(__file__), '..')))
from dao.hospital_service import IHospitalService
class HospitalServiceImpl(IHospitalService):
    def __init__(self):
        self.connection_string = self.get_connection_string()
        self.connection = pyodbc.connect(self.connection_string)
        self.cursor = self.connection.cursor()
    def get_connection_string(self):
        server_name = "MUGEE"
        database_name = "Hospital_Management"
        trusted_connection = "yes"
        return f'Driver={{SQL Server}};Server={server_name};Database={database_name};Trusted_Connection={trusted_connection};'
    def getAppointmentById(self, appointmentId):
        self.cursor.execute("SELECT * FROM Appointment WHERE appointmentId = ?", (appointmentId,))
        appointment = self.cursor.fetchone()
```

```
        return appointment
    def getAppointmentsForPatient(self, patientId):
        self.cursor.execute("SELECT * FROM Appointment WHERE patientId = ?", (patientId,))
        appointments = self.cursor.fetchall()
        return appointments

    def getAppointmentsForDoctor(self, doctorId):
        self.cursor.execute("SELECT * FROM Appointment WHERE doctorId = ?", (doctorId,))
        appointments = self.cursor.fetchall()
        return appointments
    def scheduleAppointment(self, appointment):
        self.cursor.execute("INSERT INTO Appointment (patientId, doctorId, appointmentDate, description) VALUES (?, ?, ?, ?)",
                            (appointment.patientId, appointment.doctorId, appointment.appointmentDate, appointment.description))
        self.connection.commit()
        return True
    def updateAppointment(self, appointment):
        self.cursor.execute("UPDATE Appointment SET patientId = ?, doctorId = ?, appointmentDate = ?, description = ? WHERE appointmentId = ?",
                            (appointment.patientId, appointment.doctorId, appointment.appointmentDate, appointment.description, appointment.appointmentId))
        self.connection.commit()
        return True

    def cancelAppointment(self, appointmentId):
        self.cursor.execute("DELETE FROM Appointment WHERE appointmentId = ?", (appointmentId,))
        self.connection.commit()
        return True
```

```
    def __del__(self):
        # Close database connection when the object is destroyed
        if hasattr(self, 'cursor'):
            self.cursor.close()
        if hasattr(self, 'connection'):
            self.connection.close()
```

HOSPITAL _SERVICE:

```
import sys
import os
# Add parent directory of 'dao' to Python path
sys.path.append(os.path.abspath(os.path.join(os.path.dirname(__file__), '..')))
from abc import ABC, abstractmethod
from models.Appointment import Appointment
from typing import List
class IHospitalService(ABC):
    @abstractmethod
    def getAppointmentById(self, appointment_id):
        pass
    @abstractmethod
    def getAppointmentsForPatient(self, patient_id):
        pass
    @abstractmethod
    def getAppointmentsForDoctor(self, doctor_id):
        pass
    @abstractmethod
    def scheduleAppointment(self, appointment):
        pass
    @abstractmethod
    def updateAppointment(self, appointment):
```

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

PATIENTS_EXCEPTION:

```
class PatientNumberNotFoundException(Exception):
    pass
```

UTIL:

Db_connection:

```
import pyodbc
try:
    conn = pyodbc.connect('Driver={SQL Server};'
                          'Server=MUGEE;'
                          'Database=Hospital;'
                          'Trusted_Connection=yes;')
    print("Connected Successfully")
except pyodbc.Error as e:
    print("Connection failed:", e)
    exit()
c = conn.cursor()
# Adjust your SQL query to select data from a table
sql_query = "SELECT * FROM doctor"  # Replace YourTableName with the actual table name
c.execute(sql_query)
data = c.fetchall()
for row in data:
    print(row[0], " ", row[1], " ", row[3])


c.close()
conn.close()
```

propertyutil:

```
import sys
import os
# Add parent directory of 'dao' to Python path
sys.path.append(os.path.abspath(os.path.join(os.path.dirname(__file__), '..')))
import pyodbc
class PropertyUtil:
    @staticmethod
    def getPropertyString(self):
        server_name = "MUGEE"
        database_name = "Hospital"
        trusted_connection = "yes"
        self.cursor = self.connection.cursor()
        return f"Driver={{SQL
Server}};Server={server_name};Database={database_name};Trusted_Connection={trusted_c
onnection};"
```

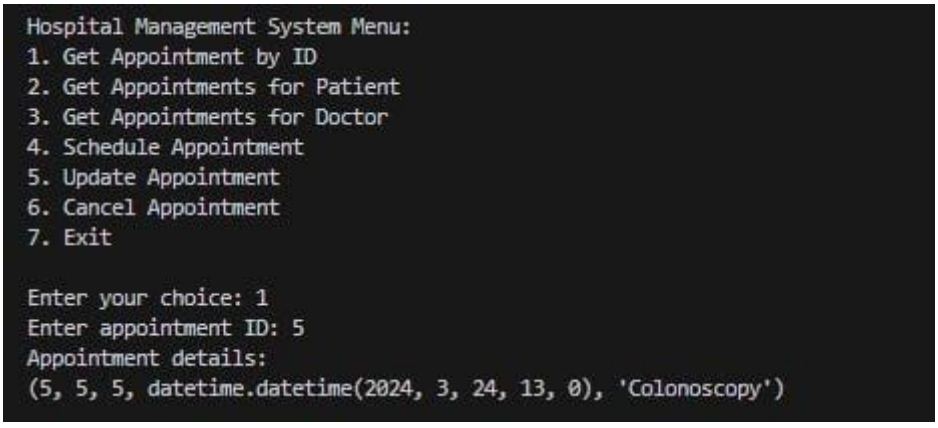
MAIN:

```
import sys
import os
# Add parent directory of 'dao' to Python path
sys.path.append(os.path.abspath(os.path.join(os.path.dirname(__file__), '..')))
# Now you should be able to import 'dao'
from dao.hospital_service_impl import HospitalServiceImpl
from exceptions.patient_exceptions import PatientNumberNotFoundException
from models.Appointment import Appointment
class MainModule:
    def __init__(self):
        self.hospital_service = HospitalServiceImpl()
    def display_menu(self):
        print("\nHospital Management System Menu:")
```

```
    print("1. Get Appointment by ID")
    print("2. Get Appointments for Patient")
    print("3. Get Appointments for Doctor")
    print("4. Schedule Appointment")
    print("5. Update Appointment")
    print("6. Cancel Appointment")
    print("7. Exit")
def get_input(self, prompt):
    return input(prompt)
def main(self):
    try:
        while True:
            self.display_menu()
            choice = int(self.get_input("\nEnter your choice: "))

            if choice == 1:
                appointment_id = int(self.get_input("Enter appointment ID: "))
                appointment = self.hospital_service.getAppointmentById(appointment_id)
                print("Appointment details:")
                print(appointment)

            elif choice == 2:
                patient_id = int(self.get_input("Enter patient ID: "))
                appointments = self.hospital_service.getAppointmentsForPatient(patient_id)
                print("Appointments for patient:")
                for appointment in appointments:
                    print(appointment)

            elif choice == 3:
```

```
            doctor_id = int(self.get_input("Enter doctor ID: "))
            appointments = self.hospital_service.getAppointmentsForDoctor(doctor_id)
            print("Appointments for doctor:")
            for appointment in appointments:
                print(appointment)

        elif choice == 4:
            appointment_id = int(self.get_input("Enter appointment ID: "))
            patient_id = int(self.get_input("Enter patient ID: "))
            doctor_id = int(self.get_input("Enter doctor ID: "))
            appointment_date = self.get_input("Enter appointment date (YYYY-MM-DD): ")
            description = self.get_input("Enter appointment description: ")
            new_appointment = Appointment(appointment_id, patient_id, doctor_id,
appointment_date, description)
            success = self.hospital_service.scheduleAppointment(new_appointment)
            if success:
                print("Appointment scheduled successfully.")
            else:
                print("Failed to schedule appointment.")
        elif choice == 5:
            appointment_id = int(self.get_input("Enter appointment ID: "))
            patient_id = int(self.get_input("Enter patient ID: "))
            doctor_id = int(self.get_input("Enter doctor ID: "))
            appointment_date = self.get_input("Enter updated appointment date (YYYY-
MM-DD): ")
            description = self.get_input("Enter updated appointment description: ")
            updated_appointment = Appointment(appointment_id, patient_id, doctor_id,
appointment_date, description)
            success = self.hospital_service.updateAppointment(updated_appointment)
            if success:
```

```
                print("Appointment updated successfully.")
            else:
                print("Failed to update appointment.")
        elif choice == 6:
            appointment_id = int(self.get_input("Enter appointment ID to cancel: "))
            success = self.hospital_service.cancelAppointment(appointment_id)
            if success:
                print("Appointment cancelled successfully.")
            else:
                print("Failed to cancel appointment.")
        elif choice == 7:
            print("Exiting...")
            break
        else:
            print("Invalid choice. Please enter a number between 1 and 7.")
    except PatientNumberNotFoundException as e:
        print("Patient number not found in the database:", e)
def new_method(self):
    return HospitalServiceImpl()
if __name__ == "__main__":
    main_module = MainModule()
    main_module.main()
```



```
Hospital Management System Menu:
1. Get Appointment by ID
2. Get Appointments for Patient
3. Get Appointments for Doctor
4. Schedule Appointment
5. Update Appointment
6. Cancel Appointment
7. Exit

Enter your choice: 4
Enter appointment ID: 16
Enter patient ID: 16
Enter doctor ID: 16
Enter appointment date (YYYY-MM-DD): 2024-9-27
Enter appointment description: Hemodialysis
Appointment scheduled successfully.
```