PYTHON CODING CHALLENGE

CHALLENGE 3 : HOSPITAL MANAGEMENT SYSTEM

NAME: VENKATA SAI SUKHESH THIRUVEEDULA

DESCRIPTION :

THE ABOVE CODING CHALLENGE – HOSPITAL MANAGEMENT SYSTEM INVOLVES CREATING SQL SCHEMAS FROM PATIENT, DOCTOR, AND APPOINTMENT CLASSES, WRITING DEFAULT CONSTRUCTORS WITH OVERLOADING THE CONSTRUCTOR WITH PARAMETERS, GETTER, AND SETTER METHODS, DEFINING IHospitalService, HospitalServiceImpl INTERFACE/ABSTRACT CLASS TO INTERACT WITH DATABASE

HERE ARE THE EXECUTION STEPS AS FOLLOWS:

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

-> Tables Patient, Doctor, Appointment are created and the samples are inserted as follows:

1. Patient:

```
mysql> CREATE TABLE Patient (
    -> patientId INT PRIMARY KEY,
    -> firstName VARCHAR(50),
    -> lastName VARCHAR(50),
    -> dateOfBirth DATE,
    -> gender CHAR(1),
    -> contactNumber VARCHAR(15),
    -> address VARCHAR(255)
    -> );
Query OK, 0 rows affected (0.02 sec)
```

2. Doctor:

```
mysql> CREATE TABLE Doctor (
    -> doctorId INT PRIMARY KEY,
    -> firstName VARCHAR(50),
    -> lastName VARCHAR(50),
    -> specialization VARCHAR(50),
    -> contactNumber VARCHAR(15)
    -> );
Query OK, 0 rows affected (0.02 sec)
```

3. Appointment:

```
mysql> CREATE TABLE Appointment (
    -> appointmentId INT PRIMARY KEY,
    -> patientId INT,
    -> doctorId INT,
    -> appointmentDate DATE,
    -> description VARCHAR(255),
    -> FOREIGN KEY (patientId) REFERENCES Patient(patientId),
    -> FOREIGN KEY (doctorId) REFERENCES Doctor(doctorId)
    -> );
Query OK, 0 rows affected (0.03 sec)
```

Inserting Sample values in Each Table:

Patient:

```
mysql> INSERT INTO Patient (patientId, firstName, lastName, dateOfBirth, gender, contactNumber, address)
    -> VALUES
    ->     (101, 'Aishwarya', 'Suresh', '1990-05-15', 'F', '8765432101', '123, MG Road, Bangalore'),
    ->     (102, 'Arjun', 'Nair', '1985-08-20', 'M', '7654321090', '456, Brigade Road, Chennai'),
    ->     (103, 'Deepa', 'Menon', '1998-12-10', 'F', '6543210987', '789, Jubilee Hills, Hyderabad'),
    ->     (104, 'Karthik', 'Rao', '1982-06-25', 'M', '6543210986', '234, MG Road, Bangalore'),
    ->     (105, 'Rekha', 'Menon', '1995-09-18', 'F', '5432109876', '567, Brigade Road, Chennai'),
    ->     (106, 'Vijay', 'Shetty', '1988-11-30', 'M', '4321098765', '890, Jubilee Hills, Hyderabad');
Query OK, 6 rows affected (0.01 sec)
Records: 6  Duplicates: 0  Warnings: 0
```

Doctor:

```
mysql> INSERT INTO Doctor (doctorId, firstName, lastName, specialization, contactNumber)
    -> VALUES
    ->      (1, 'Rajesh', 'Kumar', 'Cardiologist', '9876543210'),
    ->      (2, 'Saraswathi', 'Menon', 'Pediatrician', '8765432109'),
    ->      (3, 'Prakash', 'Nair', 'Dermatologist', '7654321098'),
    ->      (4, 'Priya', 'Sharma', 'Gynecologist', '9876543211'),
    ->      (5, 'Ganesh', 'Iyer', 'Orthopedic Surgeon', '8765432102'),
    ->      (6, 'Shalini', 'Raj', 'Ophthalmologist', '7654321091');
Query OK, 6 rows affected (0.01 sec)
Records: 6  Duplicates: 0  Warnings: 0
```

Appointments:

```
mysql> INSERT INTO Appointment (appointmentId, patientId, doctorId, appointmentDate, description)
    -> VALUES
    ->      (1001, 101, 1, '2023-01-10', 'Routine checkup'),
    ->      (1002, 102, 2, '2023-02-15', 'Vaccination'),
    ->      (1003, 103, 3, '2023-03-20', 'Skin allergy consultation'),
    ->      (1004, 104, 4, '2023-04-05', 'Prenatal checkup'),
    ->      (1005, 105, 5, '2023-05-10', 'Bone fracture consultation'),
    ->      (1006, 106, 6, '2023-06-15', 'Eye exam');
Query OK, 6 rows affected (0.00 sec)
Records: 6  Duplicates: 0  Warnings: 0
```

4: 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.

-> constructors are created with parameters, getters, and setters to the following Model classes with input and output are depicted as follows:

Patient:

```python
class Patient:
    def __init__(self, patientId=None, firstName=None, lastName=None, dateOfBirth=None, gender=None, contactNumb
        self.patientId = patientId
        self.firstName = firstName
        self.lastName = lastName
        self.dateOfBirth = dateOfBirth
        self.gender = gender
        self.contactNumber = contactNumber
        self.address = address


#Sample data
    def print_details(self):
        print("Patient ID:", self.patientId)
        print("First Name:", self.firstName)
        print("Last Name:", self.lastName)
        print("Date of Birth:", self.dateOfBirth)
        print("Gender:", self.gender)
        print("Contact Number:", self.contactNumber)
        print("Address:", self.address)

patient = Patient(patientId=101, firstName='Aishwarya', lastName='Suresh', dateOfBirth='1990-05-15', gender='F',
patient.print_details()
```

Output:

```python
Coding_Challenge > 🐍 Patient.py > ...
  1  v class Patient:
  2  v      def __init__(self, patientId=None, firstName=None, lastName=None, dateOfBirth=None, gender=None, contactNumbe
  3             self.patientId = patientId
  4             self.firstName = firstName
  5             self.lastName = lastName
  6             self.dateOfBirth = dateOfBirth
  7             self.gender = gender
  8             self.contactNumber = contactNumber
  9             self.address = address
 10
 11
 12  v #Sample data
 13  v      def print_details(self):
 14             print("Patient ID:", self.patientId)
 15             print("First Name:", self.firstName)
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

Gender: F
Contact Number: 8765432101
Address: 123, MG Road, Bangalore
● (.venv) PS C:\Users\Venkat Sri Prasad\Downloads\Python> & "c:/Users/Venkat Sri Prasad/Downloads/Python/.venv/Scripts/python.exe"
  "c:/Users/Venkat Sri Prasad/Downloads/Python/Coding_Challenge/Patient.py"
○ Patient ID: 101
  First Name: Aishwarya
  Last Name: Suresh
  Date of Birth: 1990-05-15
  Gender: F
  Contact Number: 8765432101
  Address: 123, MG Road, Bangalore
  (.venv) PS C:\Users\Venkat Sri Prasad\Downloads\Python>
```

Doctor:

```python
🐍 Doctor.py ●    🐍 Patient.py    🗋 IHospitalService.py    🐍 main.py Coding_Challenge    🐍 myexceptions.py    🐍 DatabaseConnecti

Coding_Challenge > 🐍 Doctor.py > ...
  1  class Doctor:
  2      def __init__(self, doctorId=None, firstName=None, lastName=None, specialization=None, contactNumber=None):
  3          self.doctorId = doctorId
  4          self.firstName = firstName
  5          self.lastName = lastName
  6          self.specialization = specialization
  7          self.contactNumber = contactNumber
  8
  9      #Sample Data
 10      def print_details(self):
 11          print("Doctor ID:", self.doctorId)
 12          print("First Name:", self.firstName)
 13          print("Last Name:", self.lastName)
 14          print("Specialization:", self.specialization)
 15          print("Contact Number:", self.contactNumber)
 16
 17  doctor = Doctor(doctorId=1, firstName='Rajesh', lastName='Kumar', specialization='Cardiologist', contactNumber='
 18  doctor.print_details()
 19
```

Output:

```python
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

    #Sample Data
    def print_details(self):
        print("Doctor ID:", self.doctorId)
        print("First Name:", self.firstName)
        print("Last Name:", self.lastName)
        print("Specialization:", self.specialization)
        print("Contact Number:", self.contactNumber)
```

```
"c:/Users/Venkat Sri Prasad/Downloads/Python/Coding_Challenge/Doctor.py"
Doctor ID: 1
First Name: Rajesh
Last Name: Kumar
Specialization: Cardiologist
Contact Number: 9876543210
(.venv) PS C:\Users\Venkat Sri Prasad\Downloads\Python>
```

Appointment:

```python
class Appointment:
    def __init__(self, appointmentId=None, patientId=None, doctorId=None, appointmentDate=None, description=None
        self.appointmentId = appointmentId
        self.patientId = patientId
        self.doctorId = doctorId
        self.appointmentDate = appointmentDate
        self.description = description

    # Sample data
    def print_details(self):
        print("Appointment ID:", self.appointmentId)
        print("Patient ID:", self.patientId)
        print("Doctor ID:", self.doctorId)
        print("Appointment Date:", self.appointmentDate)
        print("Description:", self.description)

appointment = Appointment(appointmentId=1001, patientId=101, doctorId=1, appointmentDate='2023-01-10', descripti
appointment.print_details()
```

Output:

```python
class Appointment:
    def __init__(self, appointmentId=None, patientId=None, doctorId=None, appointmentDate=None, description=None
        self.appointmentId = appointmentId
        self.patientId = patientId
        self.doctorId = doctorId
        self.appointmentDate = appointmentDate
        self.description = description

    # Sample data
    def print_details(self):
        print("Appointment ID:", self.appointmentId)
        print("Patient ID:", self.patientId)
        print("Doctor ID:", self.doctorId)
        print("Appointment Date:", self.appointmentDate)
        print("Description:", self.description)
```

```
"c:/Users/Venkat Sri Prasad/Downloads/Python/Coding_Challenge/Appointment.py"
Appointment ID: 1001
Patient ID: 101
Doctor ID: 1
Appointment Date: 2023-01-10
Description: Routine checkup
(.venv) PS C:\Users\Venkat Sri Prasad\Downloads\Python>
```

5. Define IHospitalService interface/abstract class with following methods to interact with database

getAppointmentById():

```
32        def get_appointment_by_id(self, appointment_id: int):
33            query = "SELECT * FROM appointment WHERE appointmentId = %s"
34            self.cursor.execute(query, (appointment_id,))
35            result = self.cursor.fetchone()
36
37            if result:
38                print(result[0], result[1], result[2], result[3], result[4])
```

getAppointmentsForPatient():

-> The Abstract class with the above methods is implemented as follows:

```
def get_appointments_for_patient(self, patient_id: int):
    query = "SELECT * FROM appointment WHERE patientId = %s"
    self.cursor.execute(query, (patient_id,))
    results = self.cursor.fetchall()

    if not results:
        raise PatientNumberNotFoundException(f"No appointments found for Patient with ID {patient_id}")
```

getAppointmentsForDoctor():

```
def get_appointments_for_doctor(self, doctor_id: int):
    query = "SELECT * FROM appointment WHERE doctorId = %s"
    self.cursor.execute(query, (doctor_id,))
    results = self.cursor.fetchall()

    appointments = []
    for result in results:
        print(result[0], result[1], result[2], result[3], result[4])
        appointments.append(result)

    return appointments
```

scheduleAppointment():

```
    def schedule_appointment(self, appointment: Appointment):
        query = "INSERT INTO appointment (appointmentId,patientId, doctorId, appointmentDate, description) VALUE
        print("Appointment is  scheduled" )
        values = (appointment[0], appointment[1], appointment[2], appointment[3],appointment[4])
        self.cursor.execute(query, values)
        self.connection.commit()
        return True
```

updateAppointment():

```
    def update_appointment(self, appointment: Appointment):
        query = "UPDATE appointment SET appointmentDate = %s, description = %s WHERE appointmentId = %s"
        values = (appointment.appointmentDate, appointment.description, appointment.appointmentId)
        self.cursor.execute(query, values)
        self.connection.commit()
        print("Appointment is Updated")
        return True
```

CancelAppointment():

```python
def cancel_appointment(self, appointment_id: int):
    query = "DELETE FROM appointment WHERE appointmentId = %s"
    self.cursor.execute(query, (appointment_id,))
    self.connection.commit()
    print("Delection Successfull")
    return True
```

6. Define HospitalServiceImpl class and implement all the methods IHospitalServiceImpl .

getAppointmentById():

->class is defined and the methods are implemented as follows:

```
(.venv) PS C:\Users\Venkat Sri Prasad\Downloads\Python> & "c:/Users/Venkat Sri
Prasad/Downloads/Python/.venv/Scripts/python.exe" "c:/Users/Venkat Sri Prasad/D
ownloads/Python/Coding_Challenge/main.py"
Database Connection Succesfull
1003 103 3 2023-03-20 Skin allergy consultation
None
```

getAppointmentsForPatient():

```
(.venv) PS C:\Users\Venkat Sri Prasad\Downloads\Python> & "c:/Users/Venkat Sri Prasad/Downloads/Python/.venv/Scripts/python.exe"
"c:/Users/Venkat Sri Prasad/Downloads/Python/Coding_Challenge/main.py"
Database Connection Succesfull
1002 102 2 2023-02-15 Vaccination
[(1002, 102, 2, datetime.date(2023, 2, 15), 'Vaccination')]
```

getAppointmentsForDoctor():

```
                                    & "c:/Users/Venkat Sri Prasad/Downloads/Python/.venv/Scripts/python.exe"
 "c:/Users/Venkat Sri Prasad/Downloads/Python/Coding_Challenge/main.py"
Database Connection Succesfull
1004 104 4 2023-04-05 Prenatal checkup
[(1004, 104, 4, datetime.date(2023, 4, 5), 'Prenatal checkup')]
```

scheduleAppointment():

```
 "c:/Users/Venkat Sri Prasad/Downloads/Python/Coding_Challenge/main.py"
Database Connection Succesfull
Appointment Scheduled
```

updateAppointment():

```
Database Connection Succesfull
Appointment is Updated
True
(.venv) PS C:\Users\Venkat Sri Prasad\Downloads\Python>
```

CancelAppointment():

```
Database Connection Succesfull
Delection Successfull
True
(.venv) PS C:\Users\Venkat Sri Prasad\Downloads\Python>
```

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

-> The Utility Class is created within a util package as follows:

```python
import mysql.connector
from datetime import date

class HospitalService:
    def __init__(self, user, password, database):
        try:
            # Connect to the MySQL server
            self.connection = mysql.connector.connect(
                user='root',
                password='venkat',
                database='Hospital_Management_System',
                port=3305
            )
            print("Database Connection Succesfull")
            # Create a cursor to execute SQL queries
            self.cursor = self.connection.cursor()

        except mysql.connector.Error as err:
            if err.errno == mysql.connector.errorcode.ER_ACCESS_DENIED_ERROR:
                print("Error: Access denied. Check your username and password.")
            elif err.errno == mysql.connector.errorcode.ER_BAD_DB_ERROR:
                print("Error: Database does not exist.")
            else:
                print("Error:", err)

hospital_service = HospitalService( user='root', password='venkat', database='Hospital_Management_System')
```

Output:

```
1   import mysql.connector
2   from datetime import date
3
4   class HospitalService:
5       def __init__(self, user, password, database):
6           try:
7               # Connect to the MySQL server
8               self.connection = mysql.connector.connect(
9                   user='root',
10                  password='venkat',
11                  database='Hospital_Management_System',
12                  port=3305
13              )
14              print("Database Connection Succesfull")
15              # Create a cursor to execute SQL queries
16              self.cursor = self.connection.cursor()
17
18          except mysql.connector.Error as err:
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   Python + ∨ ☐ 🗑 …

```
● "c:/Users/Venkat Sri Prasad/Downloads/Python/Coding_Challenge/DatabaseConnection.py"
  Database Connection Succesfull
○ (.venv) PS C:\Users\Venkat Sri Prasad\Downloads\Python> []
```

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 does not exist in db
->Exceptions are created for the respective Tasks as follows:

Task 1:

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

Task 2:

```
def get_appointments_for_patient(self, patient_id: int):
    query = "SELECT * FROM appointment WHERE patientId = %s"
    self.cursor.execute(query, (patient_id,))
    results = self.cursor.fetchall()

    if not results:
        raise PatientNumberNotFoundException(f"No appointments found for Patient with ID {patient_id}")
```

Output:

```
    raise PatientNumberNotFoundException(f"No appointments found for Patient with ID {patient_id}")
myexceptions.PatientNumberNotFoundException: No appointments found for Patient with ID 1002
(.venv) PS C:\Users\Venkat Sri Prasad\Downloads\Python> []
```

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

-> Main class is created with all the methods in service implementation class as follows:

```
Coding_Challenge >  main.py > ...
 1    from IHospitalService import HospitalService
 2    from Appointment import Appointment
 3    from datetime import date
 4    from myexceptions import PatientNumberNotFoundException
 5
 6    class main:
 7
 8        def get_appointment_by_id(self):
 9            print(hospital_service.get_appointment_by_id(1003), "\n")
 10
 11       def get_appointments_for_patient(self):
 12           print(hospital_service.get_appointments_for_patient(102), "\n")
 13
 14       def get_appointments_for_doctor(self):
 15           print(hospital_service.get_appointments_for_doctor(4), "\n")
 16
 17       def schedule_appointment(self):
 18           new_appointment = Appointment(patientId=1, doctorId=1, appointmentDate=date(2023, 12, 31), description='
 19           print("Appointment Scheduled")
 20
 21       def update_appointment(self):
 22           updated_appointment = Appointment(appointmentId=1, patientId=1, doctorId=1, appointmentDate=date(2023, 1
 23           print(hospital_service.update_appointment(updated_appointment))
 24
 25       def cancel_appointment(self):
 26           print(hospital_service.cancel_appointment(1001))
 27
 28
```

```
if __name__ == "__main__":
    hospital_service = HospitalService( user='root', password='venkat', database='Hospital_Management_System')


    Execute = main()

    Execute.get_appointment_by_id()

    Execute.get_appointments_for_patient()

    Execute.get_appointments_for_doctor()

    Execute.schedule_appointment()

    Execute.update_appointment()

    Execute.cancel_appointment()
```

The Hospital Management System Application is created by following the Directory structure entity, Dao, exception, util and creation of Main Module class according to the problem statements