Coding Challenge – Hospital Management System – Python

Submitted By-

Subrat Shukla, Python Batch 1

• Project submissions should be done through the partcipants' Github repository and the link should be shared with trainers and Hexavarsity.

• Follow object-oriented principles throughout the project. Use classes and objects to model real world entities, encapsulate data and behavior, and ensure code reusability.

• Throw user defined exceptions from corresponding methods and handled.

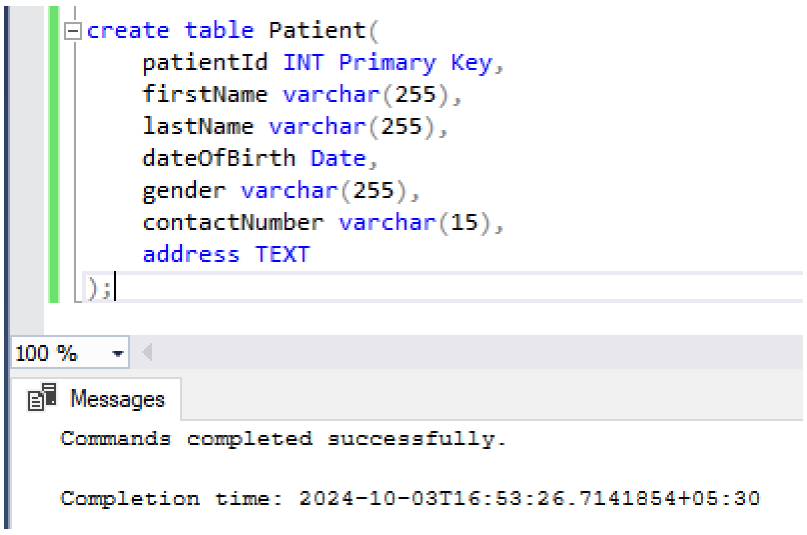
• The following Directory structure is to be followed in the application.

- entity
  - ▪ Create entity classes in this package. All entity class should not have any business logic.
- dao
  - ▪ Create Service Provider interface to showcase functionalities.
  - ▪ Create the implementation class for the above interface with db interaction.
- exception
  - ▪ Create user defined exceptions in this package and handle exceptions whenever needed.
- util
  - ▪ Create a DBPropertyUtil class with a static function which takes property file name as parameter and returns connection string.
  - ▪ Create a DBConnUtil class which holds static method which takes connection string as parameter file and returns connection object(Use method defined in DBPropertyUtil class to get the connection String).
- main
  - ▪ Create a class MainModule and demonstrate the functionalities in a menu driven application.

Problem Statement:

1. Create SQL Schema from the following classes class, use the class attributes for table column names.
2. Create the following model/entity classes within package entity with variables declared private, constructors(default and parametrized,getters,setters and toString())

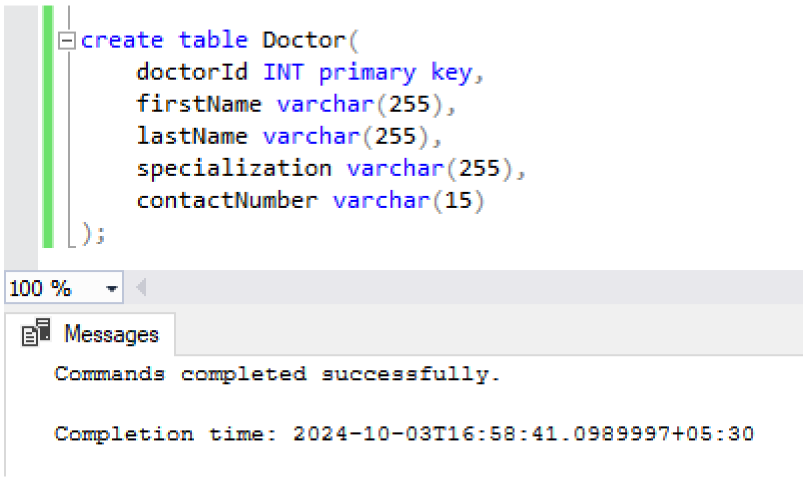
Creating Patient Table-



```
create table Patient(
    patientId INT Primary Key,
    firstName varchar(255),
    lastName varchar(255),
    dateOfBirth Date,
    gender varchar(255),
    contactNumber varchar(15),
    address TEXT
);
```

```
100 %
Messages
Commands completed successfully.

Completion time: 2024-10-03T16:53:26.7141854+05:30
```

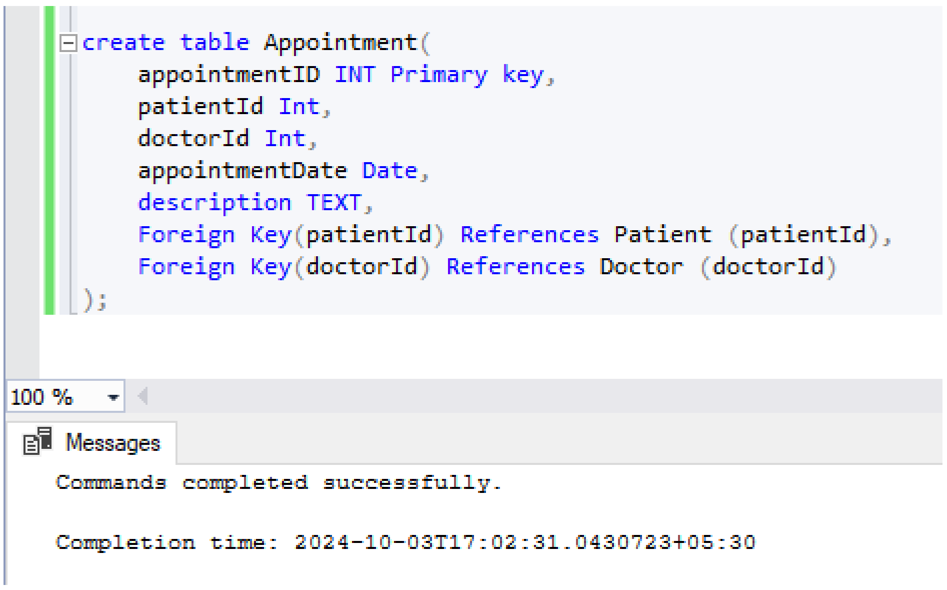
Creating Doctor Table-



```
create table Doctor(
    doctorId INT primary key,
    firstName varchar(255),
    lastName varchar(255),
    specialization varchar(255),
    contactNumber varchar(15)
);
```

```
100 %
Messages
Commands completed successfully.

Completion time: 2024-10-03T16:58:41.0989997+05:30
```

Creating Appointment Table-



```sql
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)
);
```

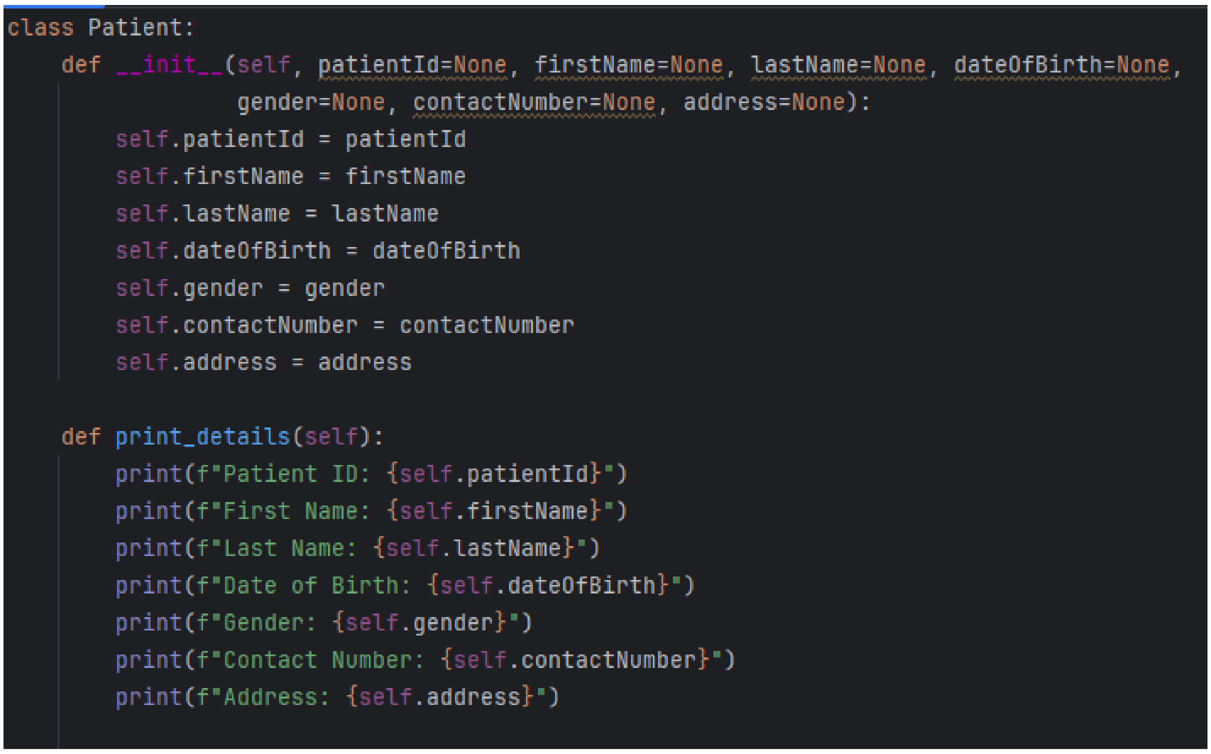
```
Commands completed successfully.

Completion time: 2024-10-03T17:02:31.0430723+05:30
```

3. 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.

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

a. patientId b. firstName c. lastName d. dateOfBirth e. gender

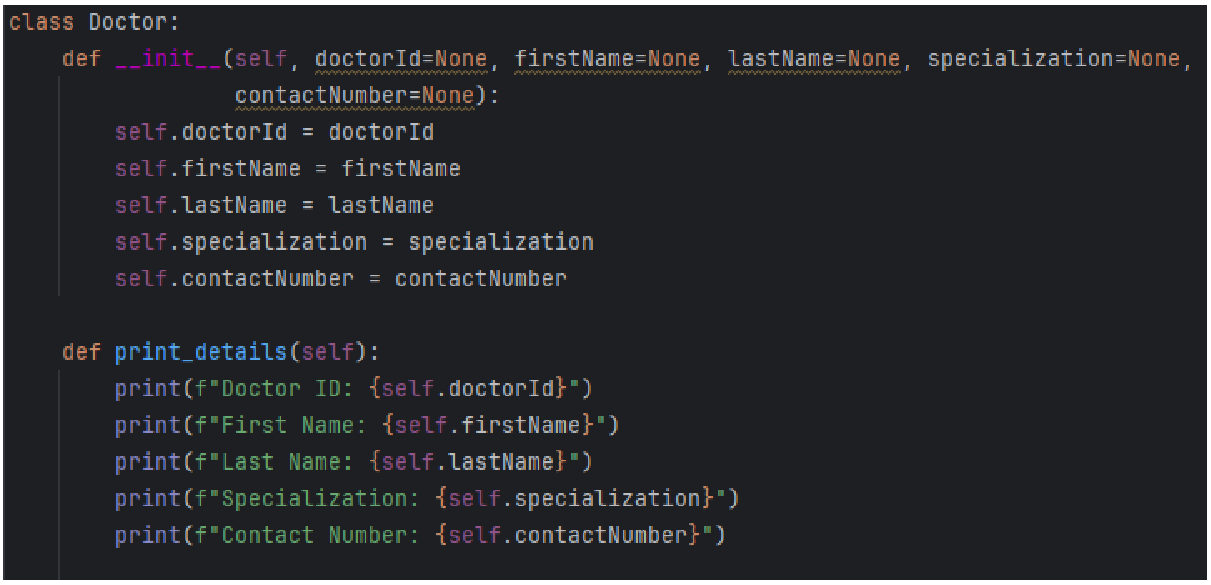
f. contactNumber g. address;



```python
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 print_details(self):
        print(f"Patient ID: {self.patientId}")
        print(f"First Name: {self.firstName}")
        print(f"Last Name: {self.lastName}")
        print(f"Date of Birth: {self.dateOfBirth}")
        print(f"Gender: {self.gender}")
        print(f"Contact Number: {self.contactNumber}")
        print(f"Address: {self.address}")
```

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

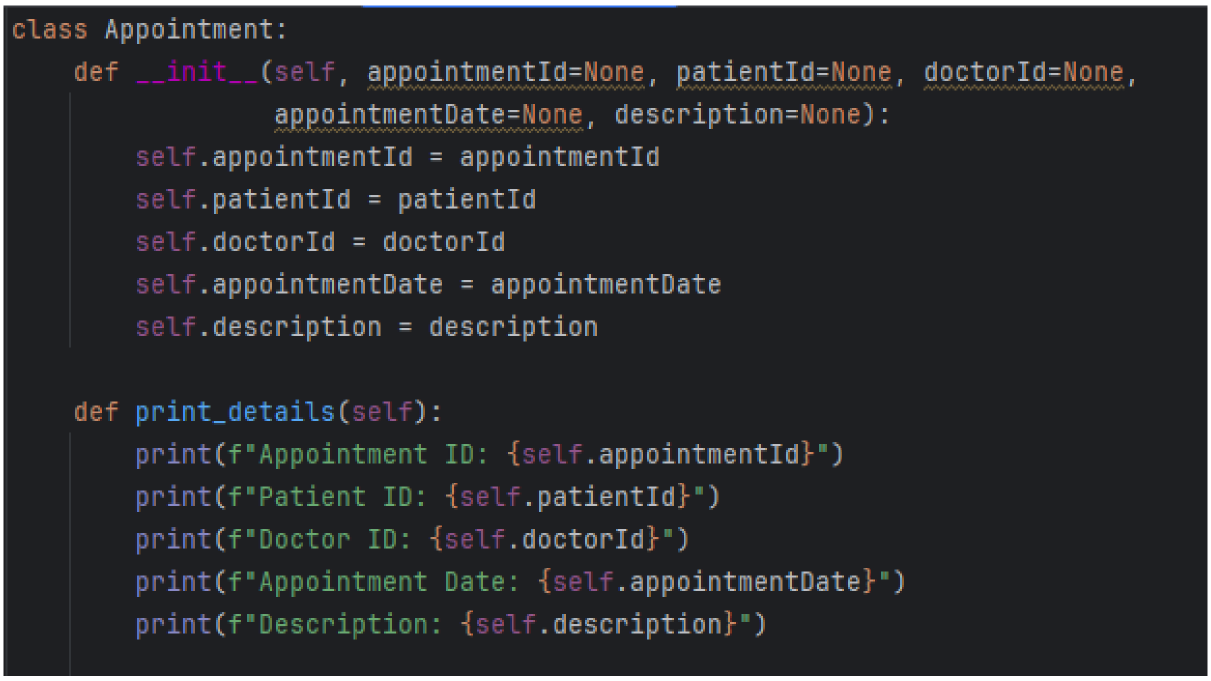
a. doctorId   b. firstName   c. lastName   d. specialization   e. contactNumber;



```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

    def print_details(self):
        print(f"Doctor ID: {self.doctorId}")
        print(f"First Name: {self.firstName}")
        print(f"Last Name: {self.lastName}")
        print(f"Specialization: {self.specialization}")
        print(f"Contact Number: {self.contactNumber}")
```

3. Appointment Class:

a. appointmentId   b. patientId   c. doctorId   d. appointmentDate   e. description



```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

    def print_details(self):
        print(f"Appointment ID: {self.appointmentId}")
        print(f"Patient ID: {self.patientId}")
        print(f"Doctor ID: {self.doctorId}")
        print(f"Appointment Date: {self.appointmentDate}")
        print(f"Description: {self.description}")
```

4. 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()

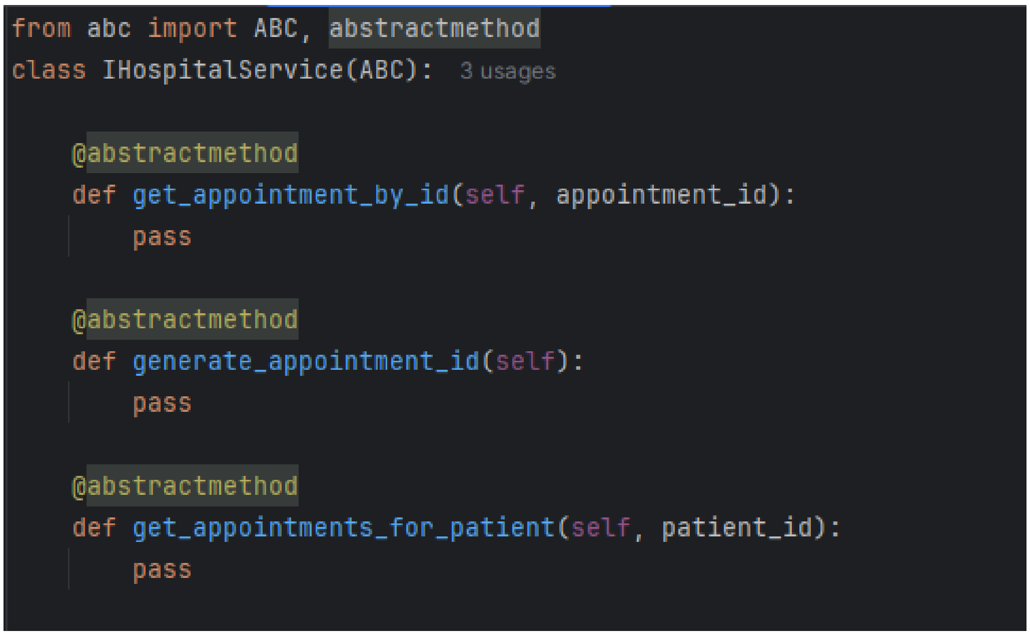
      i. Parameters: Appointment Object

      ii. ReturnType: Boolean

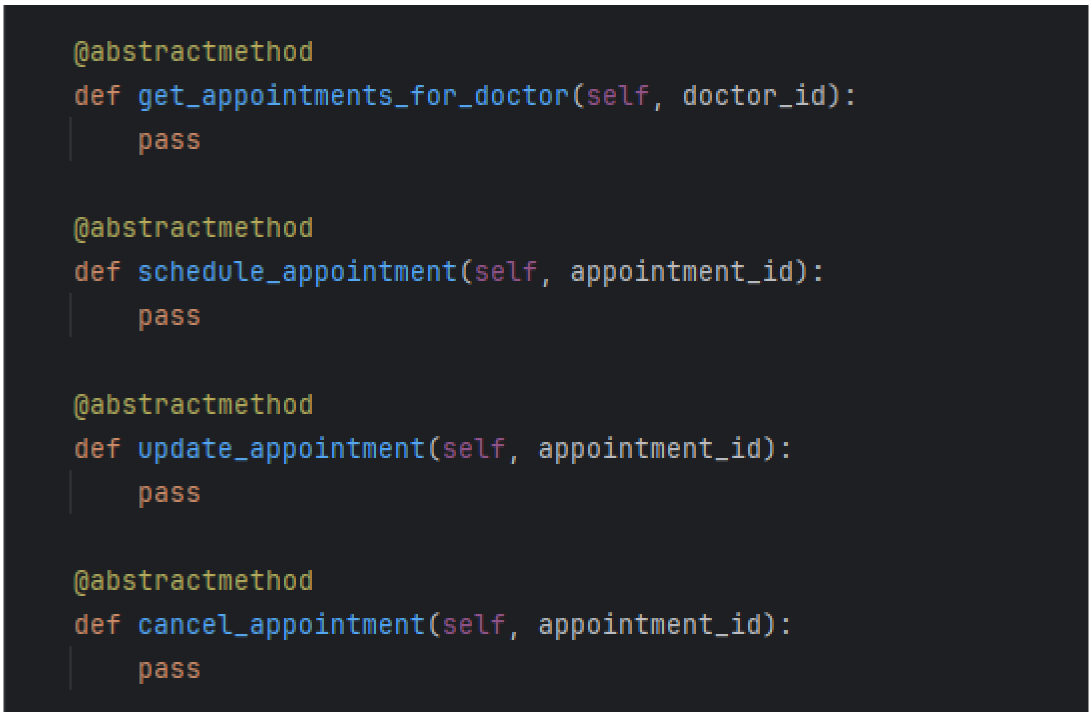
   f. cancelAppointment()

      i. Parameters: AppointmentId

      ii. ReturnType: Boolean

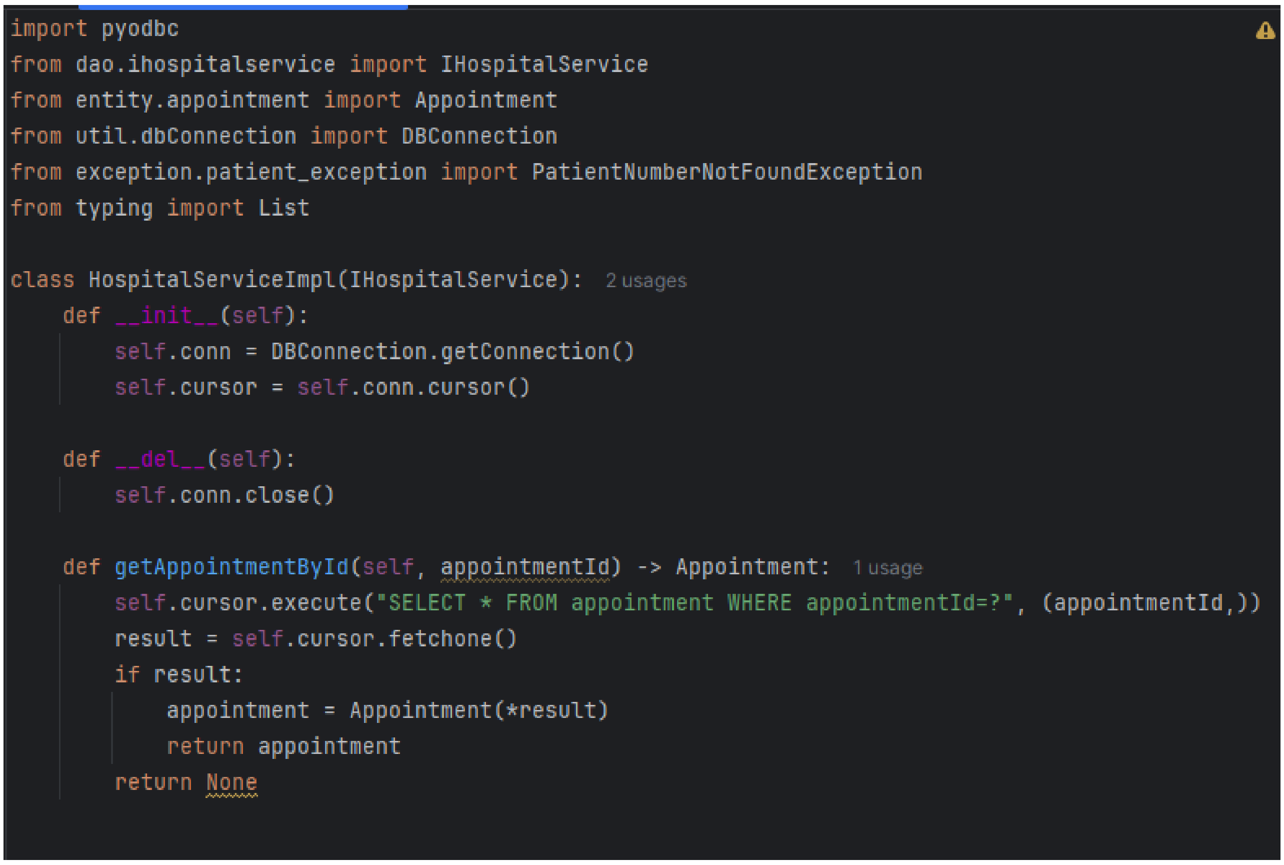


```python
from abc import ABC, abstractmethod
class IHospitalService(ABC):  3 usages

    @abstractmethod
    def get_appointment_by_id(self, appointment_id):
        pass

    @abstractmethod
    def generate_appointment_id(self):
        pass

    @abstractmethod
    def get_appointments_for_patient(self, patient_id):
        pass
```

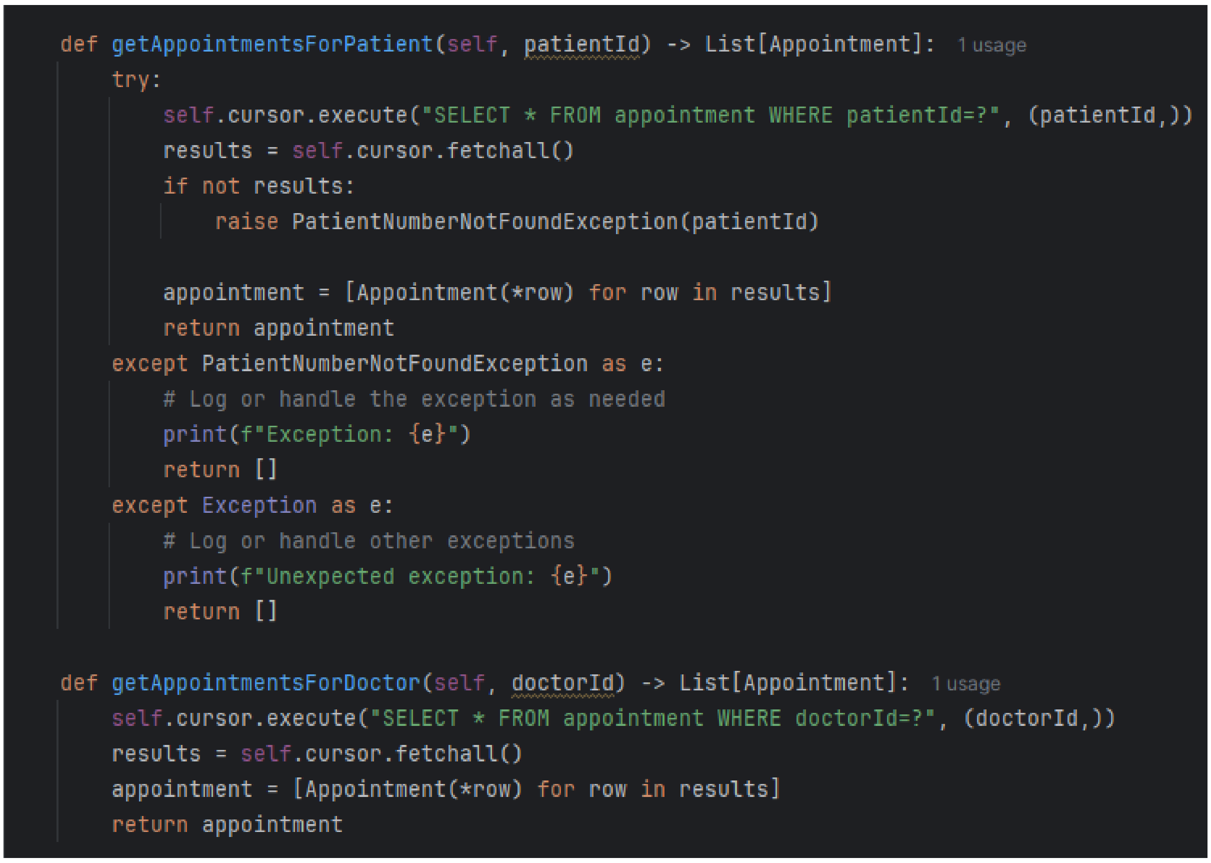


```python
    @abstractmethod
    def get_appointments_for_doctor(self, doctor_id):
        pass

    @abstractmethod
    def schedule_appointment(self, appointment_id):
        pass

    @abstractmethod
    def update_appointment(self, appointment_id):
        pass

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

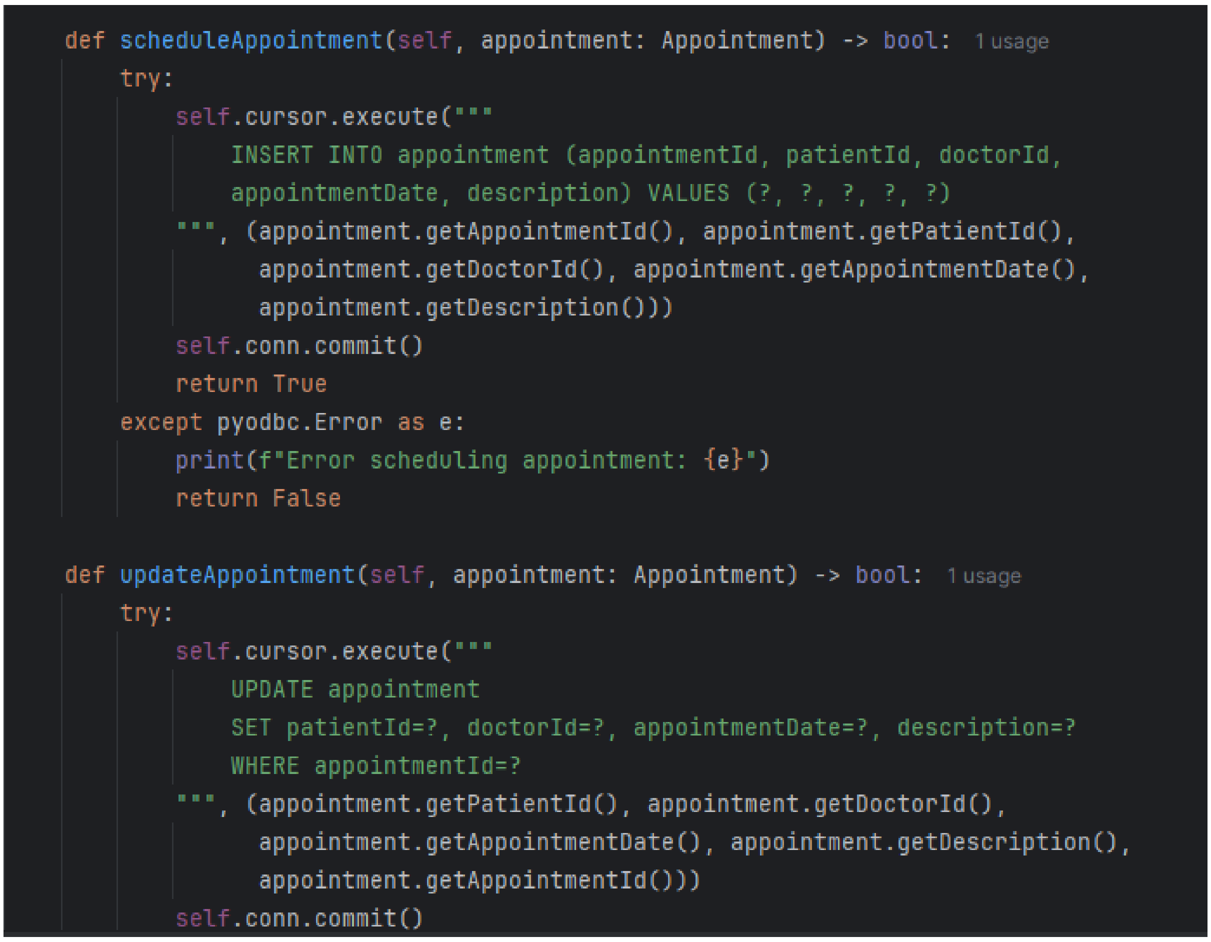
5. Define **HospitalServiceImpl** class and implement all the methods **IHospitalServiceImpl** .



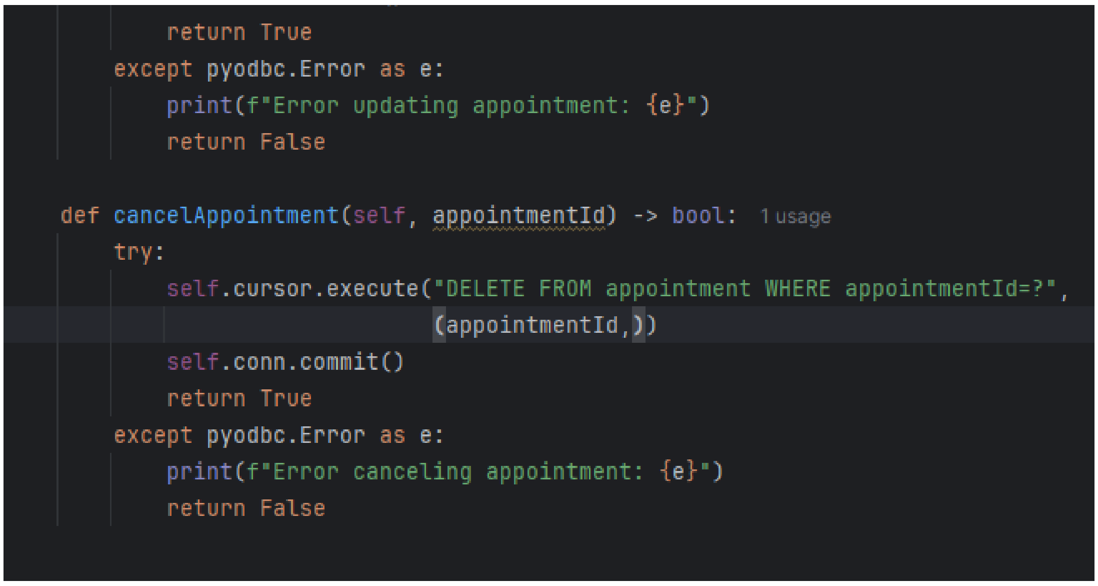
```python
import pyodbc
from dao.ihospitalservice import IHospitalService
from entity.appointment import Appointment
from util.dbConnection import DBConnection
from exception.patient_exception import PatientNumberNotFoundException
from typing import List

class HospitalServiceImpl(IHospitalService):  2 usages
    def __init__(self):
        self.conn = DBConnection.getConnection()
        self.cursor = self.conn.cursor()

    def __del__(self):
        self.conn.close()

    def getAppointmentById(self, appointmentId) -> Appointment:  1 usage
        self.cursor.execute("SELECT * FROM appointment WHERE appointmentId=?", (appointmentId,))
        result = self.cursor.fetchone()
        if result:
            appointment = Appointment(*result)
            return appointment
        return None
```



```
def getAppointmentsForPatient(self, patientId) -> List[Appointment]:  1 usage
    try:
        self.cursor.execute("SELECT * FROM appointment WHERE patientId=?", (patientId,))
        results = self.cursor.fetchall()
        if not results:
            raise PatientNumberNotFoundException(patientId)

        appointment = [Appointment(*row) for row in results]
        return appointment
    except PatientNumberNotFoundException as e:
        # Log or handle the exception as needed
        print(f"Exception: {e}")
        return []
    except Exception as e:
        # Log or handle other exceptions
        print(f"Unexpected exception: {e}")
        return []

def getAppointmentsForDoctor(self, doctorId) -> List[Appointment]:  1 usage
    self.cursor.execute("SELECT * FROM appointment WHERE doctorId=?", (doctorId,))
    results = self.cursor.fetchall()
    appointment = [Appointment(*row) for row in results]
    return appointment
```

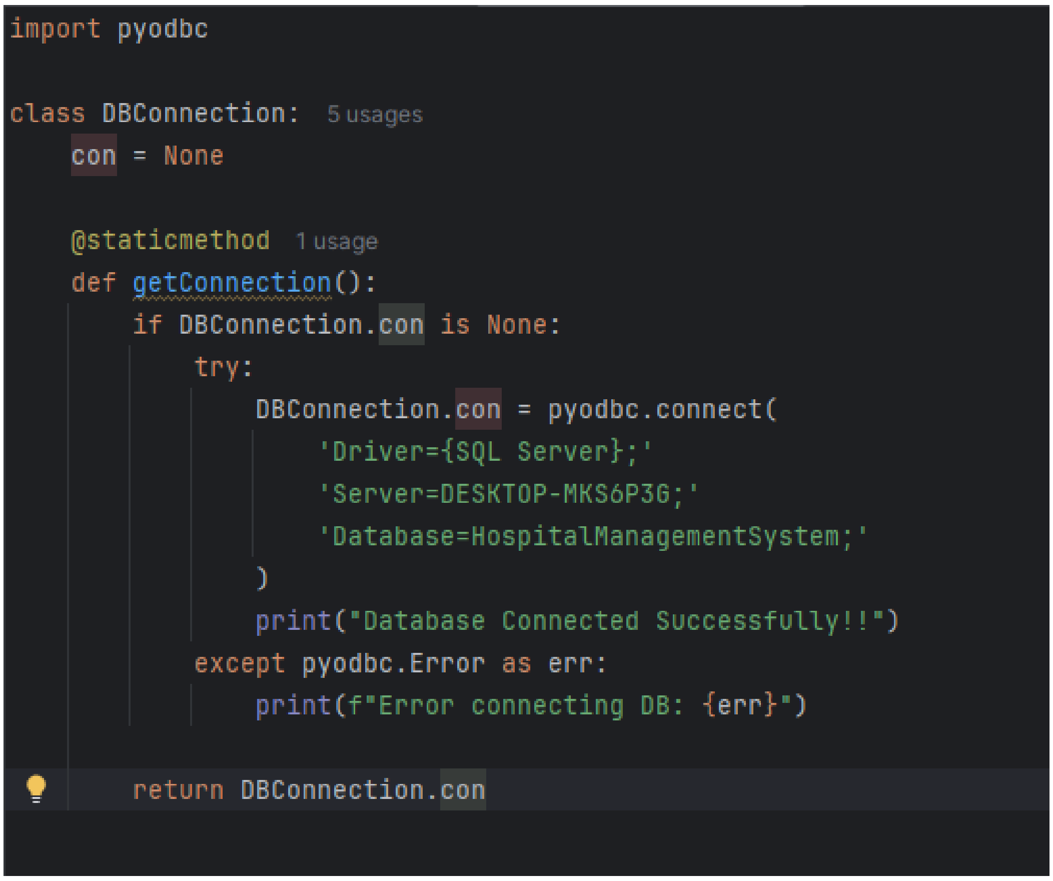


```
def scheduleAppointment(self, appointment: Appointment) -> bool:  1 usage
    try:
        self.cursor.execute("""
            INSERT INTO appointment (appointmentId, patientId, doctorId,
            appointmentDate, description) VALUES (?, ?, ?, ?, ?)
        """, (appointment.getAppointmentId(), appointment.getPatientId(),
              appointment.getDoctorId(), appointment.getAppointmentDate(),
              appointment.getDescription()))
        self.conn.commit()
        return True
    except pyodbc.Error as e:
        print(f"Error scheduling appointment: {e}")
        return False

def updateAppointment(self, appointment: Appointment) -> bool:  1 usage
    try:
        self.cursor.execute("""
            UPDATE appointment
            SET patientId=?, doctorId=?, appointmentDate=?, description=?
            WHERE appointmentId=?
        """, (appointment.getPatientId(), appointment.getDoctorId(),
              appointment.getAppointmentDate(), appointment.getDescription(),
              appointment.getAppointmentId()))
        self.conn.commit()
```



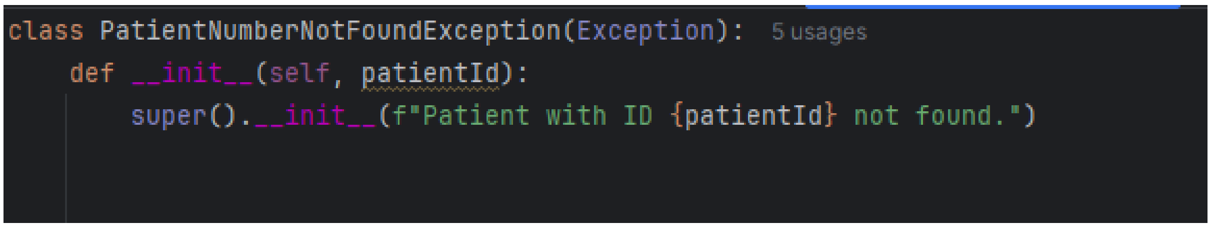
```
            return True
        except pyodbc.Error as e:
            print(f"Error updating appointment: {e}")
            return False

    def cancelAppointment(self, appointmentId) -> bool:  1 usage
        try:
            self.cursor.execute("DELETE FROM appointment WHERE appointmentId=?",
                                (appointmentId,))
            self.conn.commit()
            return True
        except pyodbc.Error as e:
            print(f"Error canceling appointment: {e}")
            return False
```

6. 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.



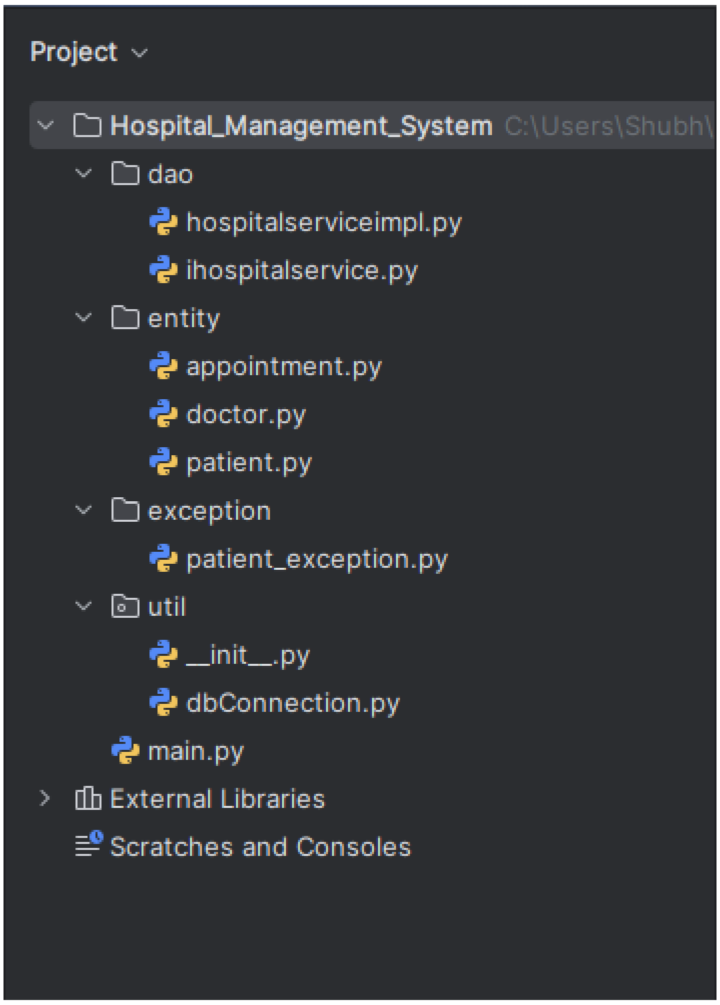
```
import pyodbc

class DBConnection:  5 usages
    con = None

    @staticmethod  1 usage
    def getConnection():
        if DBConnection.con is None:
            try:
                DBConnection.con = pyodbc.connect(
                    'Driver={SQL Server};'
                    'Server=DESKTOP-MKS6P3G;'
                    'Database=HospitalManagementSystem;'
                )
                print("Database Connected Successfully!!")
            except pyodbc.Error as err:
                print(f"Error connecting DB: {err}")

        return DBConnection.con
```

7. 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.



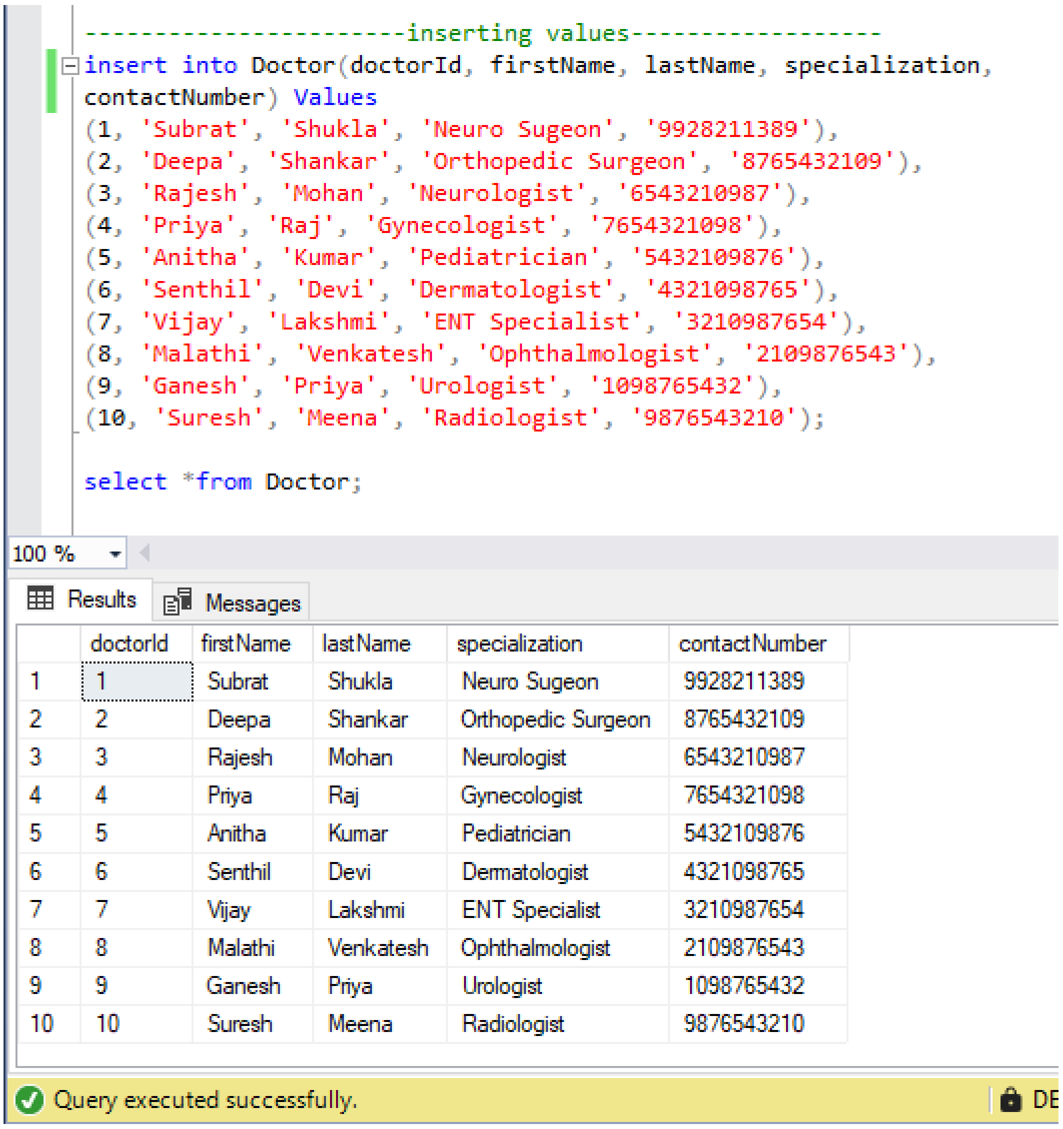
```
class PatientNumberNotFoundException(Exception):  5 usages
    def __init__(self, patientId):
        super().__init__(f"Patient with ID {patientId} not found.")
```

My Project Structure:



```
Project
Hospital_Management_System C:\Users\Shubh\
    dao
        hospitalserviceimpl.py
        ihospitalservice.py
    entity
        appointment.py
        doctor.py
        patient.py
    exception
        patient_exception.py
    util
        __init__.py
        dbConnection.py
    main.py
External Libraries
Scratches and Consoles
```

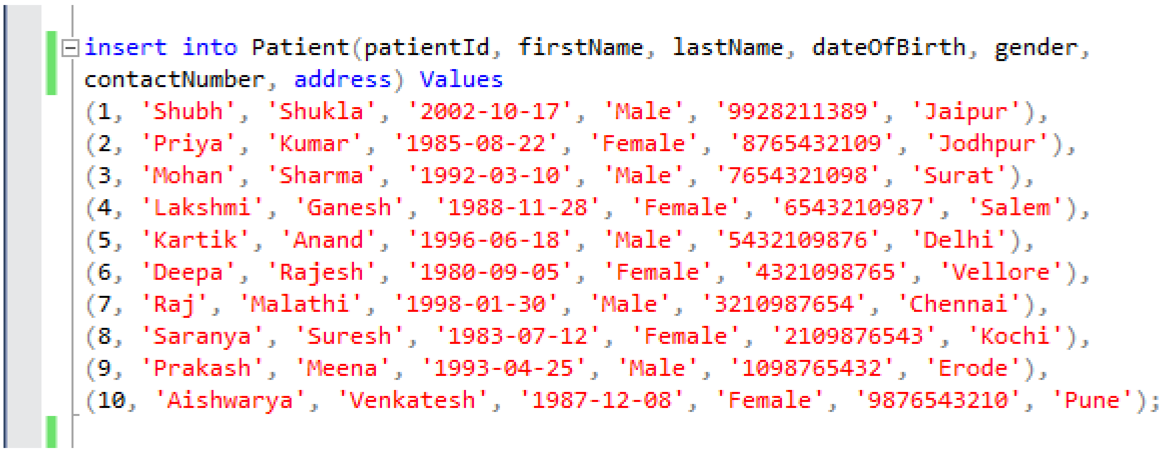
Inserting 10 sample datas into Doctor and Patient tables so as to perform the implementation:



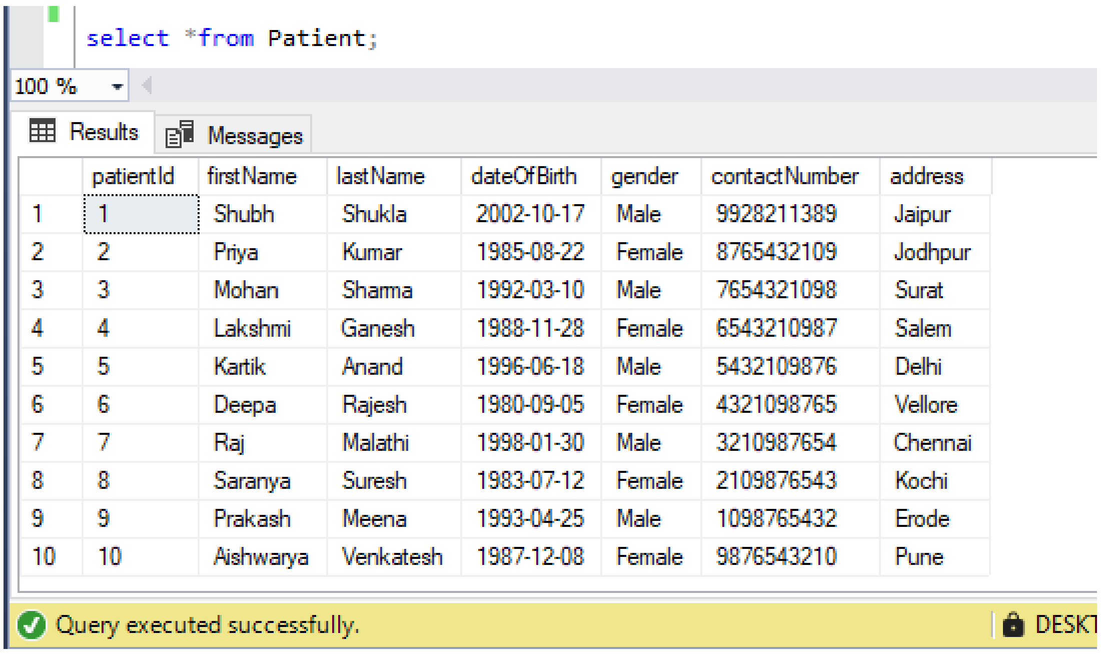
```
-----------------------inserting values-----------------
insert into Doctor(doctorId, firstName, lastName, specialization,
contactNumber) Values
(1, 'Subrat', 'Shukla', 'Neuro Sugeon', '9928211389'),
(2, 'Deepa', 'Shankar', 'Orthopedic Surgeon', '8765432109'),
(3, 'Rajesh', 'Mohan', 'Neurologist', '6543210987'),
(4, 'Priya', 'Raj', 'Gynecologist', '7654321098'),
(5, 'Anitha', 'Kumar', 'Pediatrician', '5432109876'),
(6, 'Senthil', 'Devi', 'Dermatologist', '4321098765'),
(7, 'Vijay', 'Lakshmi', 'ENT Specialist', '3210987654'),
(8, 'Malathi', 'Venkatesh', 'Ophthalmologist', '2109876543'),
(9, 'Ganesh', 'Priya', 'Urologist', '1098765432'),
(10, 'Suresh', 'Meena', 'Radiologist', '9876543210');

select *from Doctor;
```

	doctorId	firstName	lastName	specialization	contactNumber
1	1	Subrat	Shukla	Neuro Sugeon	9928211389
2	2	Deepa	Shankar	Orthopedic Surgeon	8765432109
3	3	Rajesh	Mohan	Neurologist	6543210987
4	4	Priya	Raj	Gynecologist	7654321098
5	5	Anitha	Kumar	Pediatrician	5432109876
6	6	Senthil	Devi	Dermatologist	4321098765
7	7	Vijay	Lakshmi	ENT Specialist	3210987654
8	8	Malathi	Venkatesh	Ophthalmologist	2109876543
9	9	Ganesh	Priya	Urologist	1098765432
10	10	Suresh	Meena	Radiologist	9876543210

Query executed successfully.



```
insert into Patient(patientId, firstName, lastName, dateOfBirth, gender,
contactNumber, address) Values
(1, 'Shubh', 'Shukla', '2002-10-17', 'Male', '9928211389', 'Jaipur'),
(2, 'Priya', 'Kumar', '1985-08-22', 'Female', '8765432109', 'Jodhpur'),
(3, 'Mohan', 'Sharma', '1992-03-10', 'Male', '7654321098', 'Surat'),
(4, 'Lakshmi', 'Ganesh', '1988-11-28', 'Female', '6543210987', 'Salem'),
(5, 'Kartik', 'Anand', '1996-06-18', 'Male', '5432109876', 'Delhi'),
(6, 'Deepa', 'Rajesh', '1980-09-05', 'Female', '4321098765', 'Vellore'),
(7, 'Raj', 'Malathi', '1998-01-30', 'Male', '3210987654', 'Chennai'),
(8, 'Saranya', 'Suresh', '1983-07-12', 'Female', '2109876543', 'Kochi'),
(9, 'Prakash', 'Meena', '1993-04-25', 'Male', '1098765432', 'Erode'),
(10, 'Aishwarya', 'Venkatesh', '1987-12-08', 'Female', '9876543210', 'Pune');
```



```
select *from Patient;
```

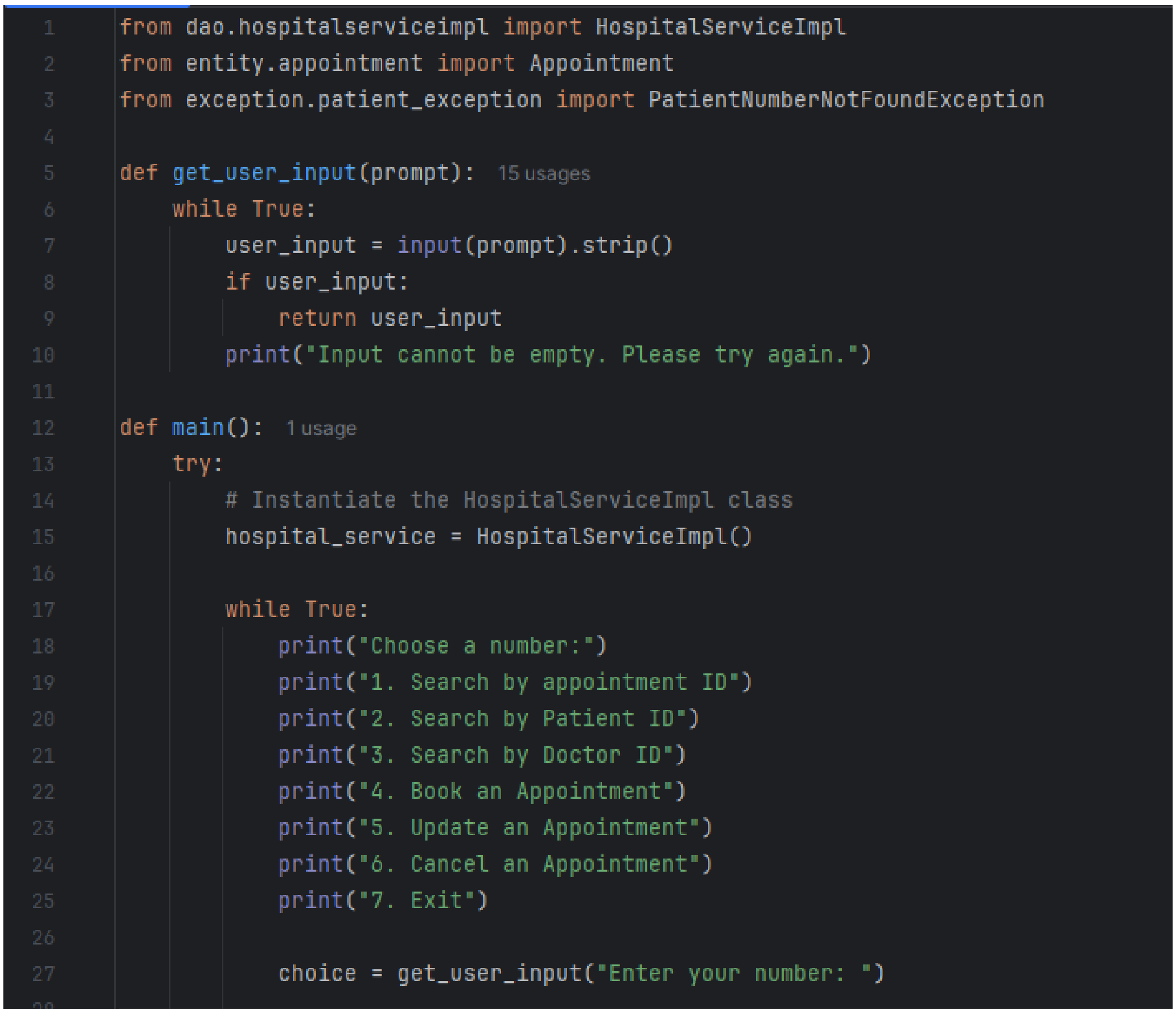
100 %

Results | Messages

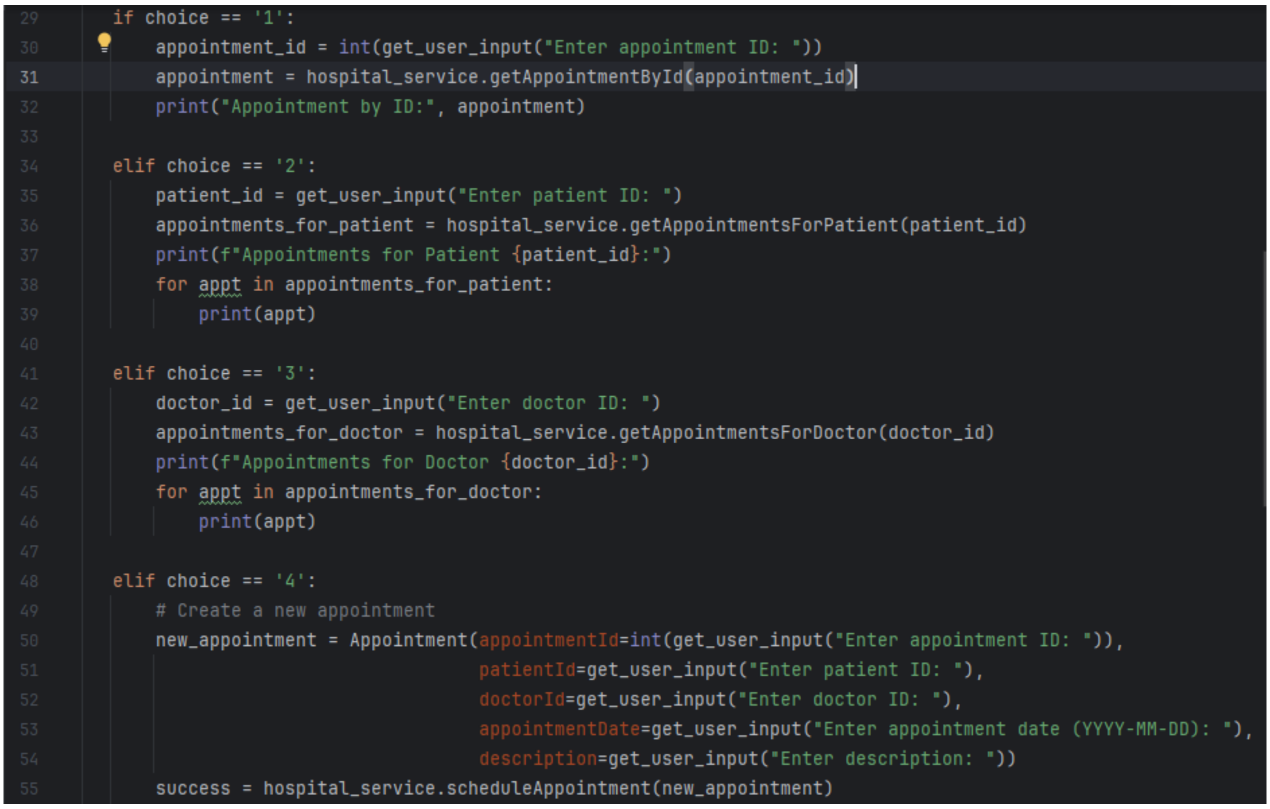
	patientId	firstName	lastName	dateOfBirth	gender	contactNumber	address
1	1	Shubh	Shukla	2002-10-17	Male	9928211389	Jaipur
2	2	Priya	Kumar	1985-08-22	Female	8765432109	Jodhpur
3	3	Mohan	Sharma	1992-03-10	Male	7654321098	Surat
4	4	Lakshmi	Ganesh	1988-11-28	Female	6543210987	Salem
5	5	Kartik	Anand	1996-06-18	Male	5432109876	Delhi
6	6	Deepa	Rajesh	1980-09-05	Female	4321098765	Vellore
7	7	Raj	Malathi	1998-01-30	Male	3210987654	Chennai
8	8	Saranya	Suresh	1983-07-12	Female	2109876543	Kochi
9	9	Prakash	Meena	1993-04-25	Male	1098765432	Erode
10	10	Aishwarya	Venkatesh	1987-12-08	Female	9876543210	Pune

Query executed successfully. DESKT

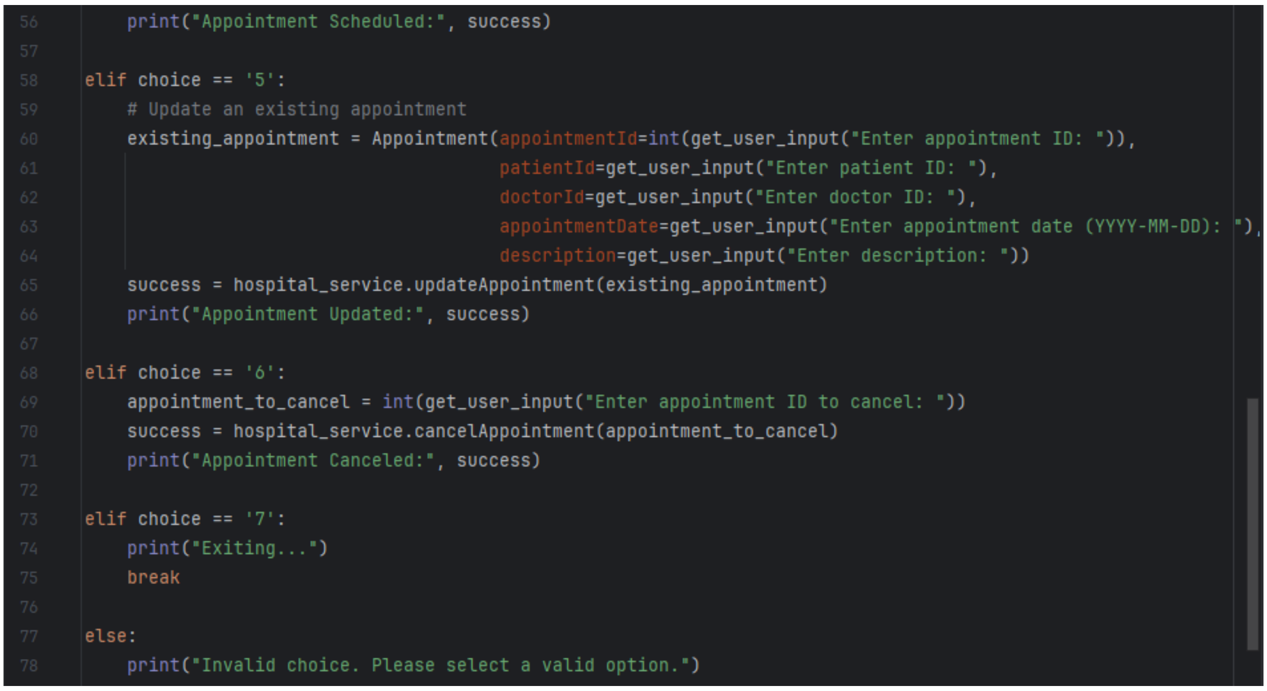
8. Create class named MainModule with main method in package mainmod. Trigger all the methods in service implementation class.



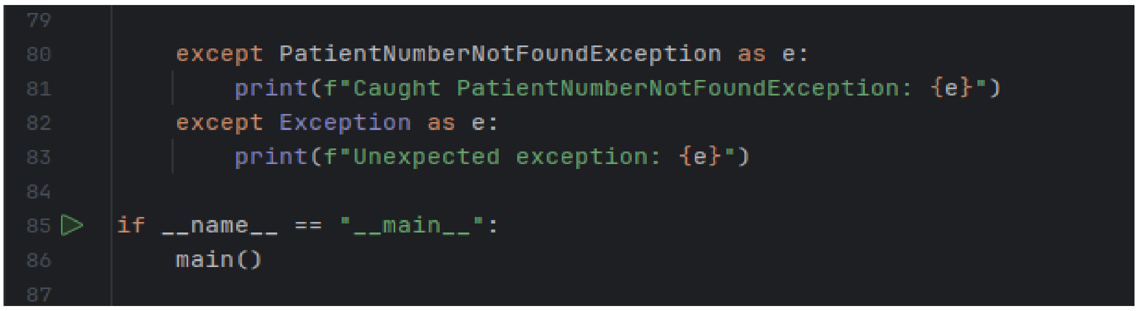
```
from dao.hospitalserviceimpl import HospitalServiceImpl
from entity.appointment import Appointment
from exception.patient_exception import PatientNumberNotFoundException

def get_user_input(prompt):  15 usages
    while True:
        user_input = input(prompt).strip()
        if user_input:
            return user_input
        print("Input cannot be empty. Please try again.")

def main():  1 usage
    try:
        # Instantiate the HospitalServiceImpl class
        hospital_service = HospitalServiceImpl()

        while True:
            print("Choose a number:")
            print("1. Search by appointment ID")
            print("2. Search by Patient ID")
            print("3. Search by Doctor ID")
            print("4. Book an Appointment")
            print("5. Update an Appointment")
            print("6. Cancel an Appointment")
            print("7. Exit")

            choice = get_user_input("Enter your number: ")
```



```python
            if choice == '1':
                appointment_id = int(get_user_input("Enter appointment ID: "))
                appointment = hospital_service.getAppointmentById(appointment_id)
                print("Appointment by ID:", appointment)

            elif choice == '2':
                patient_id = get_user_input("Enter patient ID: ")
                appointments_for_patient = hospital_service.getAppointmentsForPatient(patient_id)
                print(f"Appointments for Patient {patient_id}:")
                for appt in appointments_for_patient:
                    print(appt)

            elif choice == '3':
                doctor_id = get_user_input("Enter doctor ID: ")
                appointments_for_doctor = hospital_service.getAppointmentsForDoctor(doctor_id)
                print(f"Appointments for Doctor {doctor_id}:")
                for appt in appointments_for_doctor:
                    print(appt)

            elif choice == '4':
                # Create a new appointment
                new_appointment = Appointment(appointmentId=int(get_user_input("Enter appointment ID: ")),
                                              patientId=get_user_input("Enter patient ID: "),
                                              doctorId=get_user_input("Enter doctor ID: "),
                                              appointmentDate=get_user_input("Enter appointment date (YYYY-MM-DD): "),
                                              description=get_user_input("Enter description: "))
                success = hospital_service.scheduleAppointment(new_appointment)
```



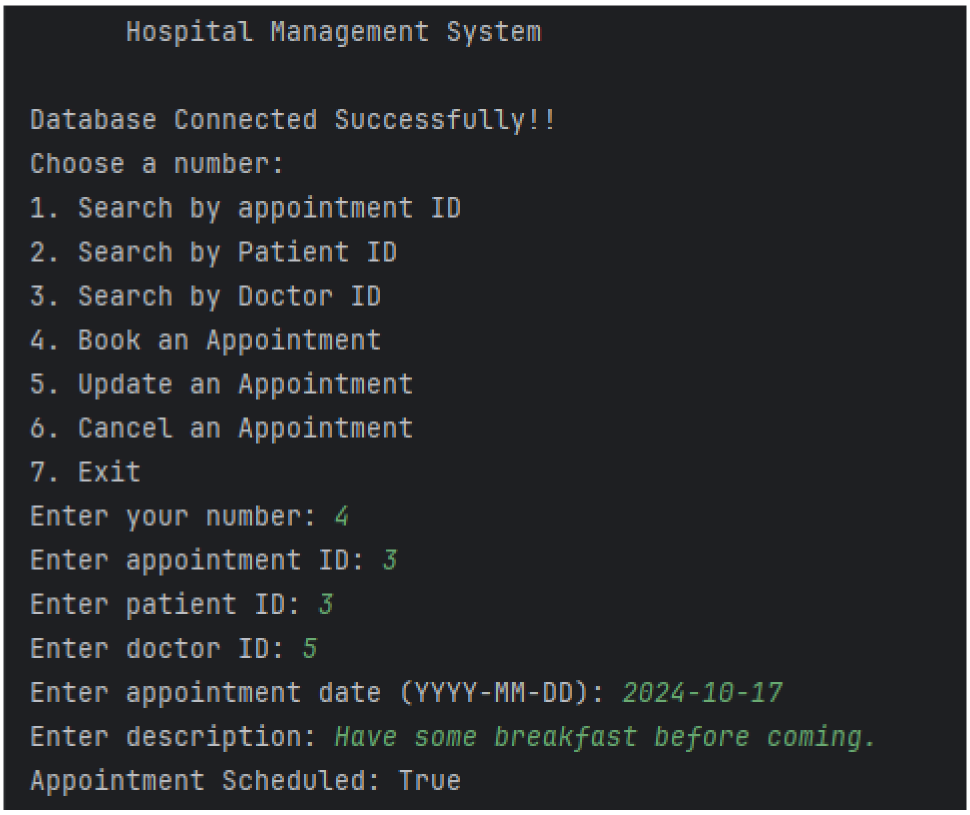
```python
                print("Appointment Scheduled:", success)

            elif choice == '5':
                # Update an existing appointment
                existing_appointment = Appointment(appointmentId=int(get_user_input("Enter appointment ID: ")),
                                                   patientId=get_user_input("Enter patient ID: "),
                                                   doctorId=get_user_input("Enter doctor ID: "),
                                                   appointmentDate=get_user_input("Enter appointment date (YYYY-MM-DD): "),
                                                   description=get_user_input("Enter description: "))
                success = hospital_service.updateAppointment(existing_appointment)
                print("Appointment Updated:", success)

            elif choice == '6':
                appointment_to_cancel = int(get_user_input("Enter appointment ID to cancel: "))
                success = hospital_service.cancelAppointment(appointment_to_cancel)
                print("Appointment Canceled:", success)

            elif choice == '7':
                print("Exiting...")
                break

            else:
                print("Invalid choice. Please select a valid option.")
```



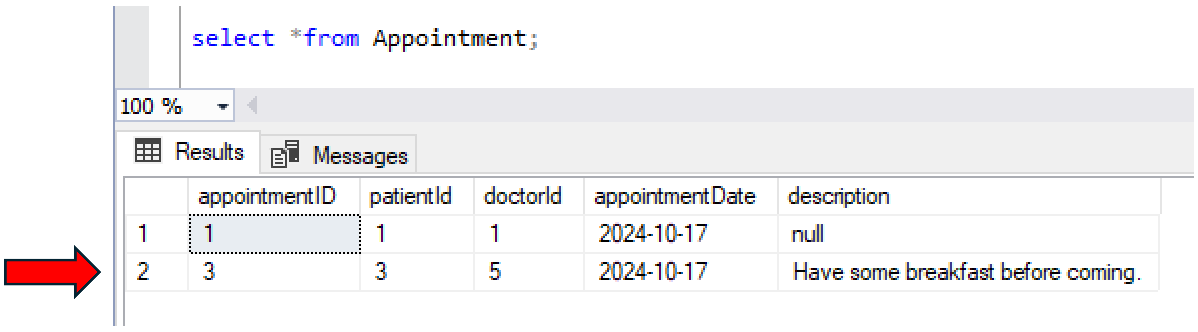
```python

    except PatientNumberNotFoundException as e:
        print(f"Caught PatientNumberNotFoundException: {e}")
    except Exception as e:
        print(f"Unexpected exception: {e}")

if __name__ == "__main__":
    main()

```

Implementation:

Scheduling appointment for the patient with patient ID 3 to the doctor with doctor ID 5 on Current Date.



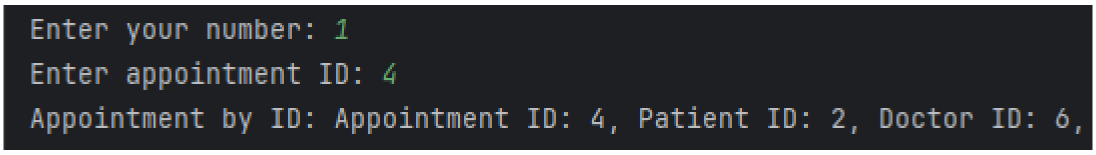
```
       Hospital Management System

Database Connected Successfully!!
Choose a number:
1. Search by appointment ID
2. Search by Patient ID
3. Search by Doctor ID
4. Book an Appointment
5. Update an Appointment
6. Cancel an Appointment
7. Exit
Enter your number: 4
Enter appointment ID: 3
Enter patient ID: 3
Enter doctor ID: 5
Enter appointment date (YYYY-MM-DD): 2024-10-17
Enter description: Have some breakfast before coming.
Appointment Scheduled: True
```



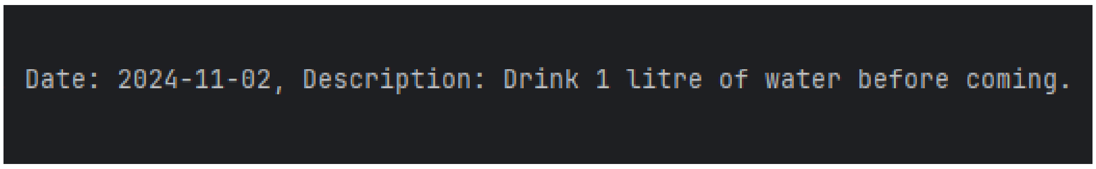
```
select *from Appointment;
```

	appointmentID	patientId	doctorId	appointmentDate	description
1	1	1	1	2024-10-17	null
2	3	3	5	2024-10-17	Have some breakfast before coming.

Getting the appointment details by giving **appointment ID** as input:

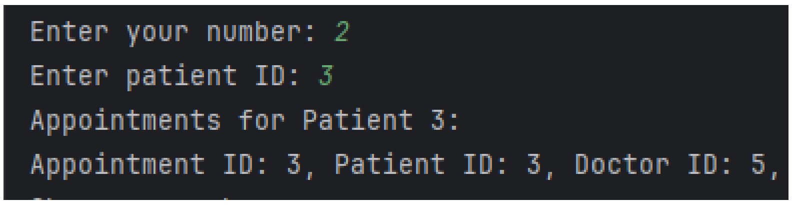


```
Enter your number: 1
Enter appointment ID: 4
Appointment by ID: Appointment ID: 4, Patient ID: 2, Doctor ID: 6,
```

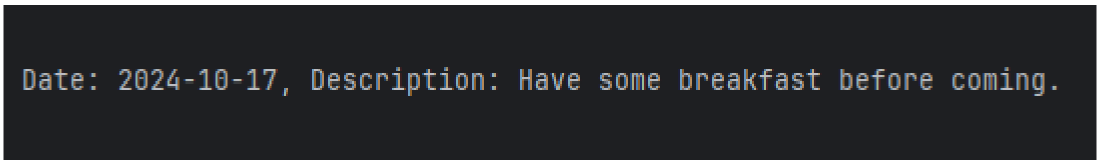


```
Date: 2024-11-02, Description: Drink 1 litre of water before coming.
```

Getting appointment details for patient by giving **patient ID** as input:

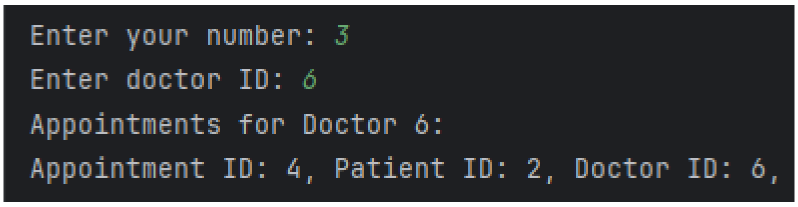


```
Enter your number: 2
Enter patient ID: 3
Appointments for Patient 3:
Appointment ID: 3, Patient ID: 3, Doctor ID: 5,
```

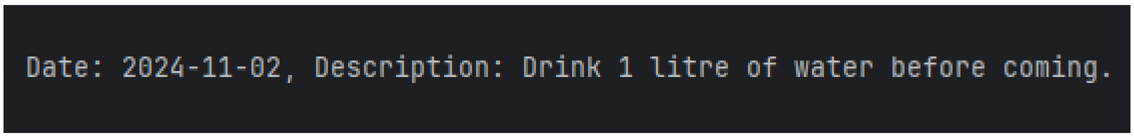


```
Date: 2024-10-17, Description: Have some breakfast before coming.
```

Getting appointment details for doctor by giving **Doctor ID** as input:

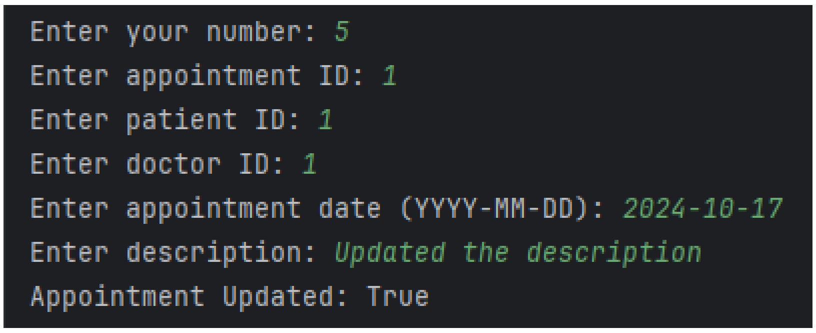


```
Enter your number: 3
Enter doctor ID: 6
Appointments for Doctor 6:
Appointment ID: 4, Patient ID: 2, Doctor ID: 6,
```



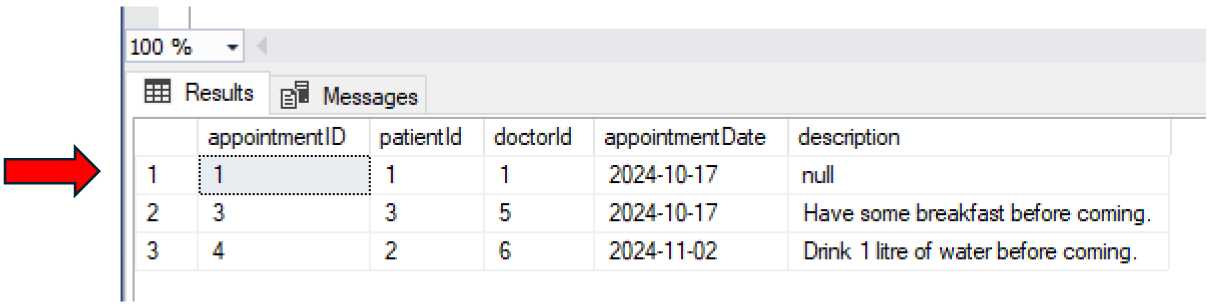
```
Date: 2024-11-02, Description: Drink 1 litre of water before coming.
```

Updating Appointment with appointment ID: 1



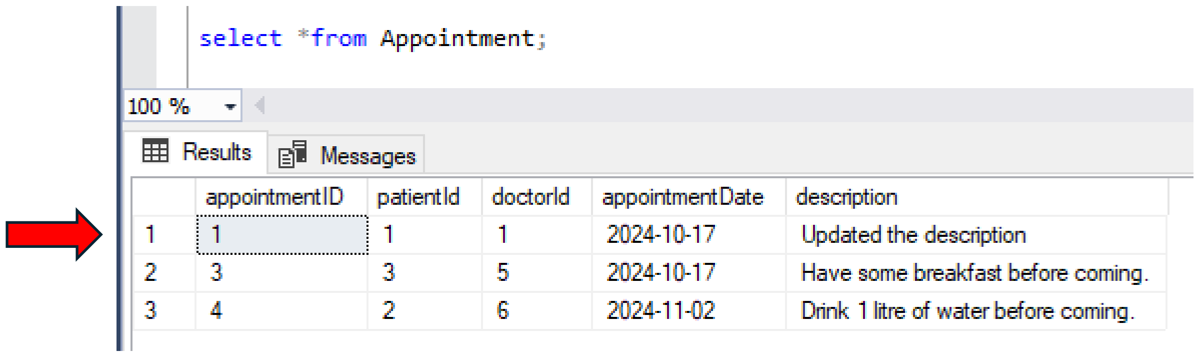
```
Enter your number: 5
Enter appointment ID: 1
Enter patient ID: 1
Enter doctor ID: 1
Enter appointment date (YYYY-MM-DD): 2024-10-17
Enter description: Updated the description
Appointment Updated: True
```

Before Update:



	appointmentID	patientId	doctorId	appointmentDate	description
1	1	1	1	2024-10-17	null
2	3	3	5	2024-10-17	Have some breakfast before coming.
3	4	2	6	2024-11-02	Drink 1 litre of water before coming.

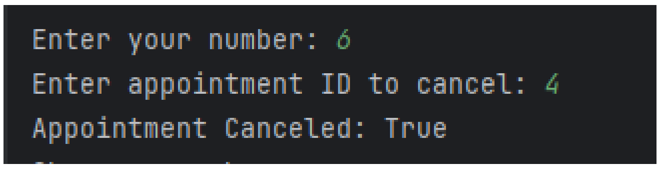
After Updating:



```
select *from Appointment;
```

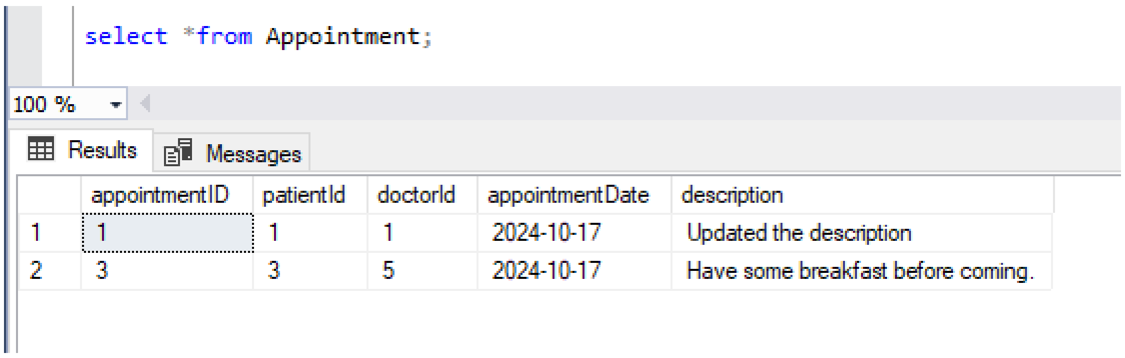
	appointmentID	patientId	doctorId	appointmentDate	description
1	1	1	1	2024-10-17	Updated the description
2	3	3	5	2024-10-17	Have some breakfast before coming.
3	4	2	6	2024-11-02	Drink 1 litre of water before coming.

Cancelling a particular appointment:



```
Enter your number: 6
Enter appointment ID to cancel: 4
Appointment Canceled: True
```

After cancelling it removes the appointment record from the database:



```
select *from Appointment;
```

	appointmentID	patientId	doctorId	appointmentDate	description
1	1	1	1	2024-10-17	Updated the description
2	3	3	5	2024-10-17	Have some breakfast before coming.