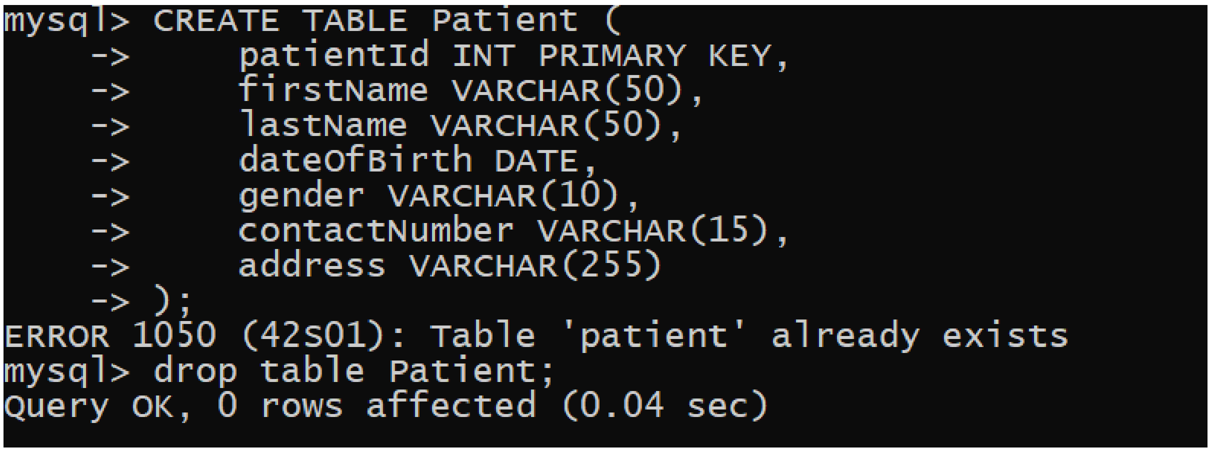
Y.Asritha

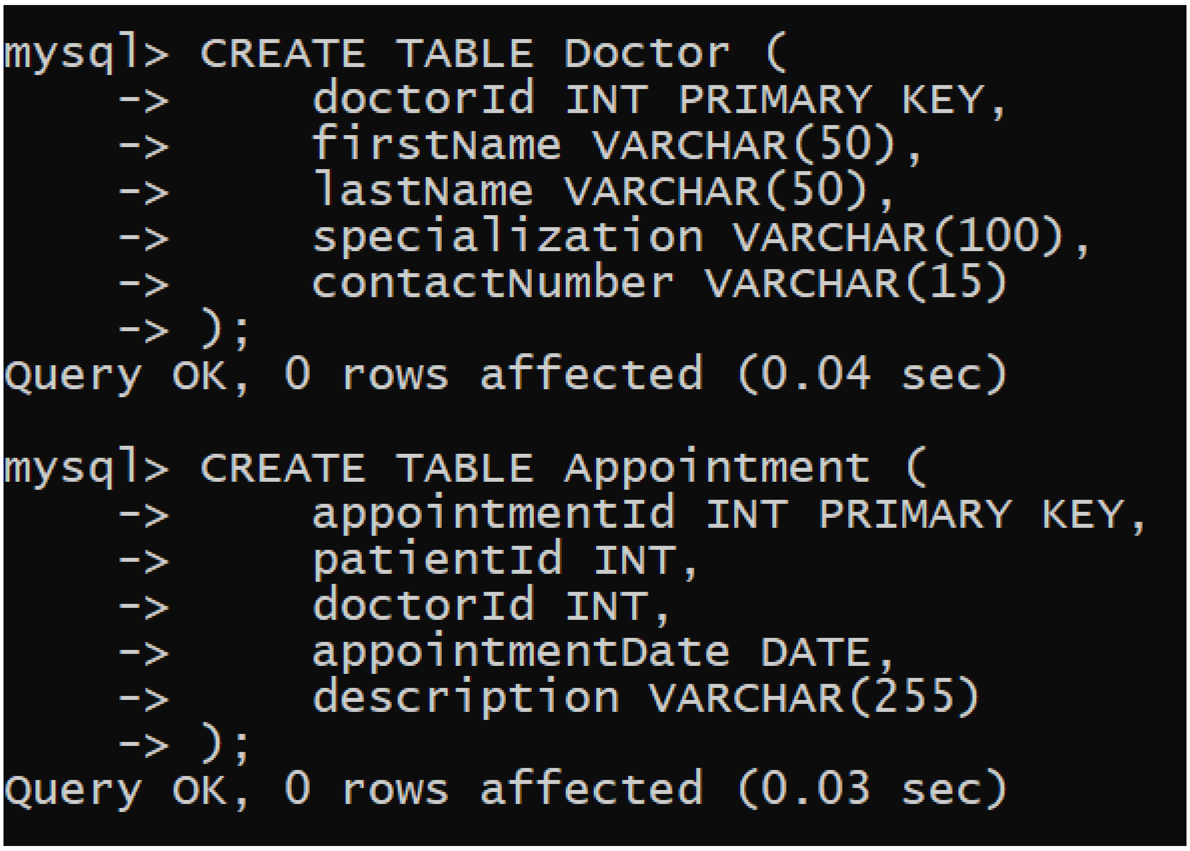
Coding challenge-3

# Hospital management system

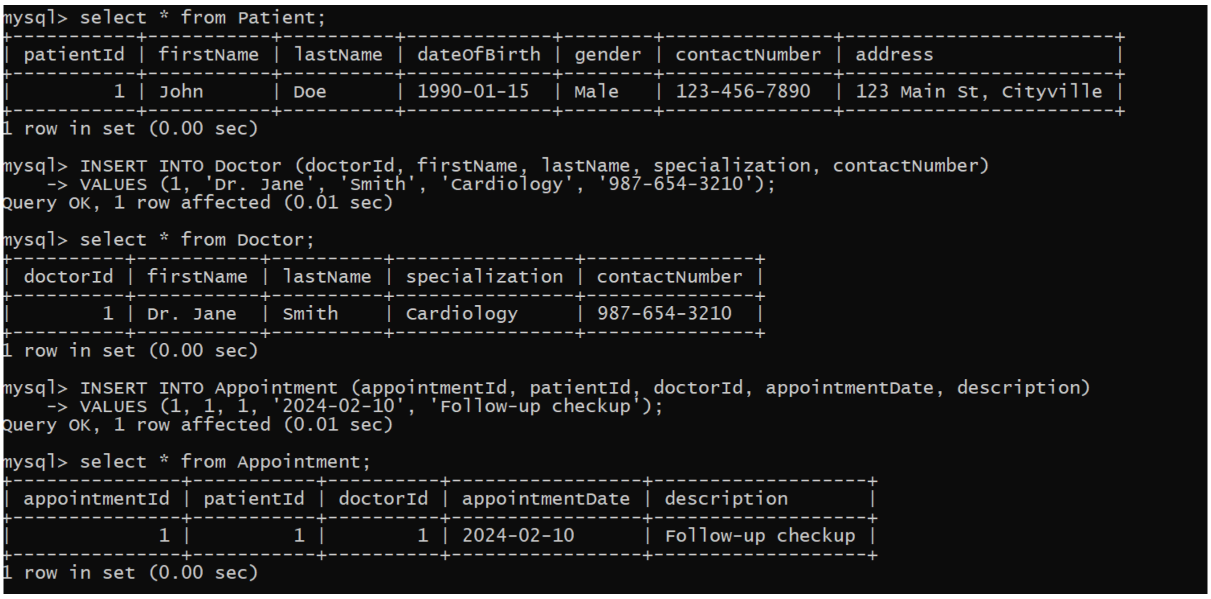
1**Create SQL Schema from the following classes class, use the class attributes for table column names.**



```
mysql> CREATE TABLE Patient (
    ->     patientId INT PRIMARY KEY,
    ->     firstName VARCHAR(50),
    ->     lastName VARCHAR(50),
    ->     dateOfBirth DATE,
    ->     gender VARCHAR(10),
    ->     contactNumber VARCHAR(15),
    ->     address VARCHAR(255)
    -> );
ERROR 1050 (42S01): Table 'patient' already exists
mysql> drop table Patient;
Query OK, 0 rows affected (0.04 sec)
```

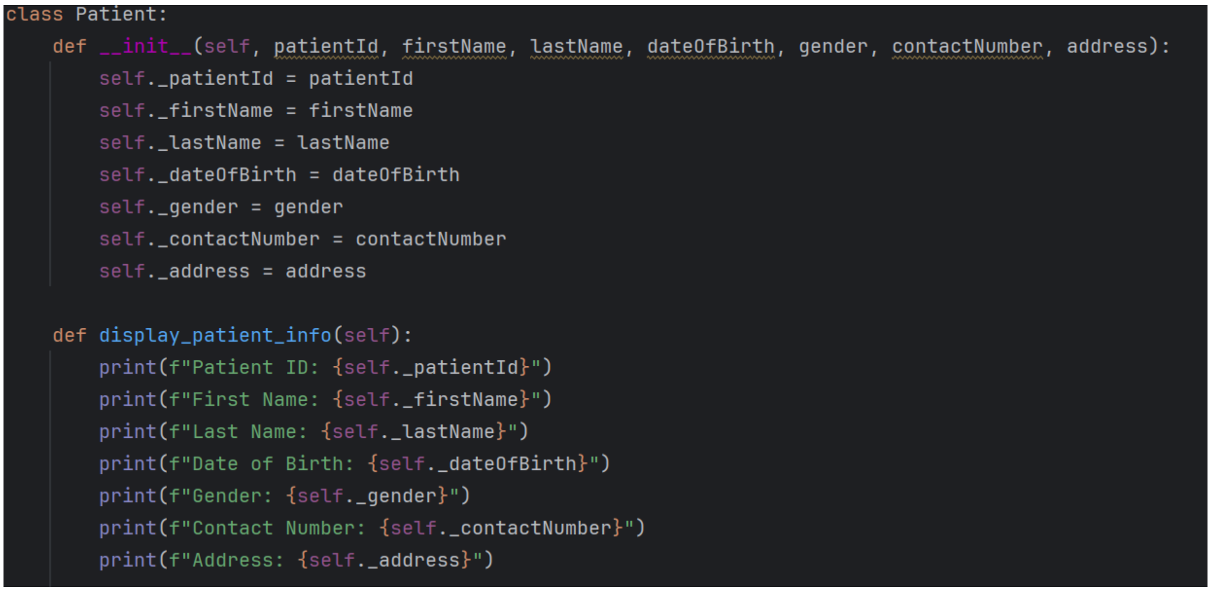


```
mysql> CREATE TABLE Doctor (
    ->     doctorId INT PRIMARY KEY,
    ->     firstName VARCHAR(50),
    ->     lastName VARCHAR(50),
    ->     specialization VARCHAR(100),
    ->     contactNumber VARCHAR(15)
    -> );
Query OK, 0 rows affected (0.04 sec)

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

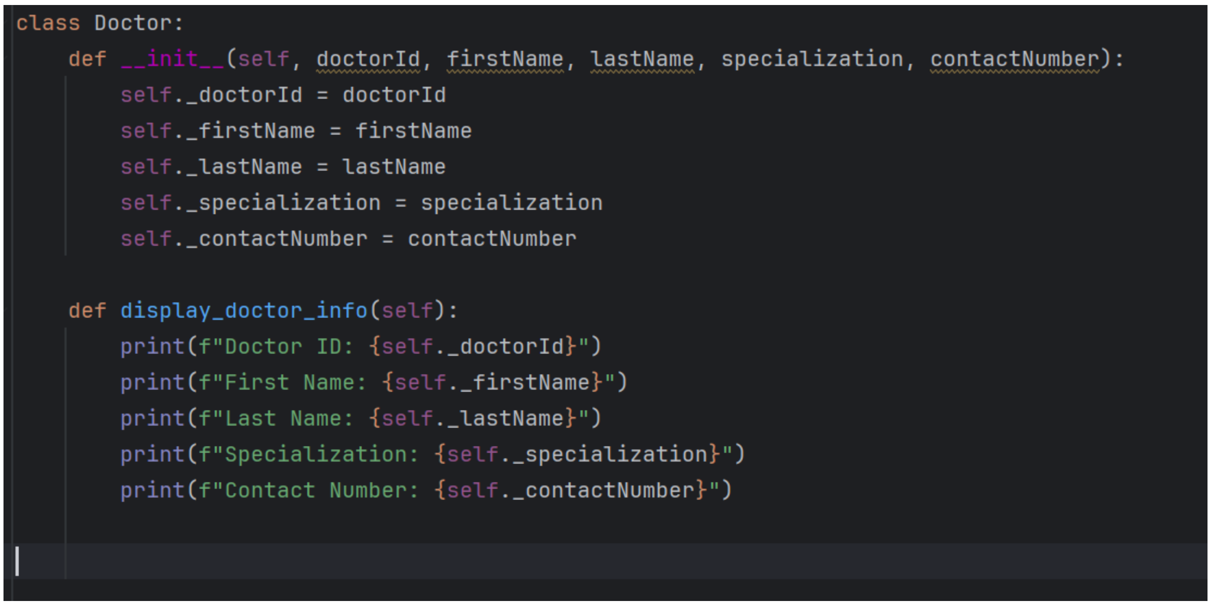


```
mysql> select * from Patient;
+-----------+-----------+----------+-------------+--------+---------------+-------------------------+
| patientId | firstName | lastName | dateOfBirth | gender | contactNumber | address                 |
+-----------+-----------+----------+-------------+--------+---------------+-------------------------+
|         1 | John      | Doe      | 1990-01-15  | Male   | 123-456-7890  | 123 Main St, Cityville |
+-----------+-----------+----------+-------------+--------+---------------+-------------------------+
1 row in set (0.00 sec)

mysql> INSERT INTO Doctor (doctorId, firstName, lastName, specialization, contactNumber)
    -> VALUES (1, 'Dr. Jane', 'Smith', 'Cardiology', '987-654-3210');
Query OK, 1 row affected (0.01 sec)

mysql> select * from Doctor;
+----------+-----------+----------+----------------+---------------+
| doctorId | firstName | lastName | specialization | contactNumber |
+----------+-----------+----------+----------------+---------------+
|        1 | Dr. Jane  | Smith    | Cardiology     | 987-654-3210  |
+----------+-----------+----------+----------------+---------------+
1 row in set (0.00 sec)

mysql> INSERT INTO Appointment (appointmentId, patientId, doctorId, appointmentDate, description)
    -> VALUES (1, 1, 1, '2024-02-10', 'Follow-up checkup');
Query OK, 1 row affected (0.01 sec)

mysql> select * from Appointment;
+---------------+-----------+----------+-----------------+-------------------+
| appointmentId | patientId | doctorId | appointmentDate | description       |
+---------------+-----------+----------+-----------------+-------------------+
|             1 |         1 |        1 | 2024-02-10      | Follow-up checkup |
+---------------+-----------+----------+-----------------+-------------------+
1 row in set (0.00 sec)
```

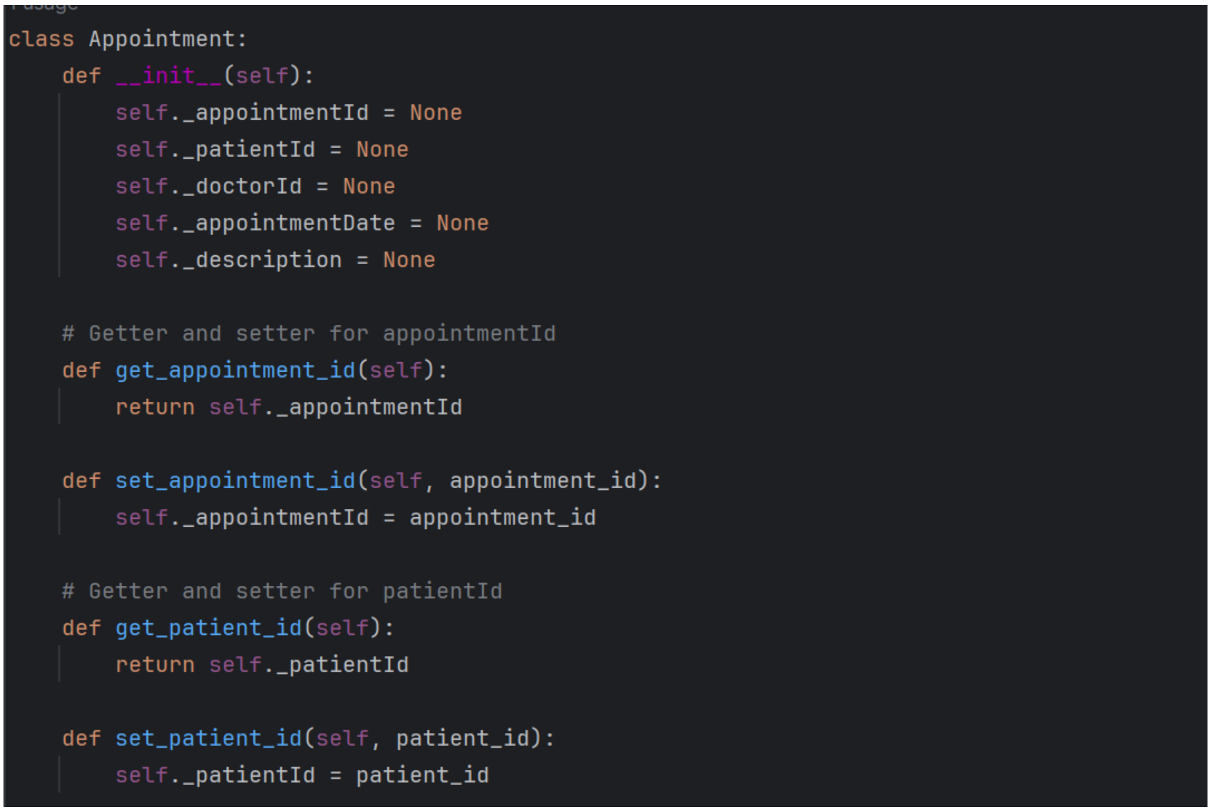
1. Implement the following for all model classes. Write default constructors and overload the constructor with parameters, getters and setters, method to print all the member variables and values.



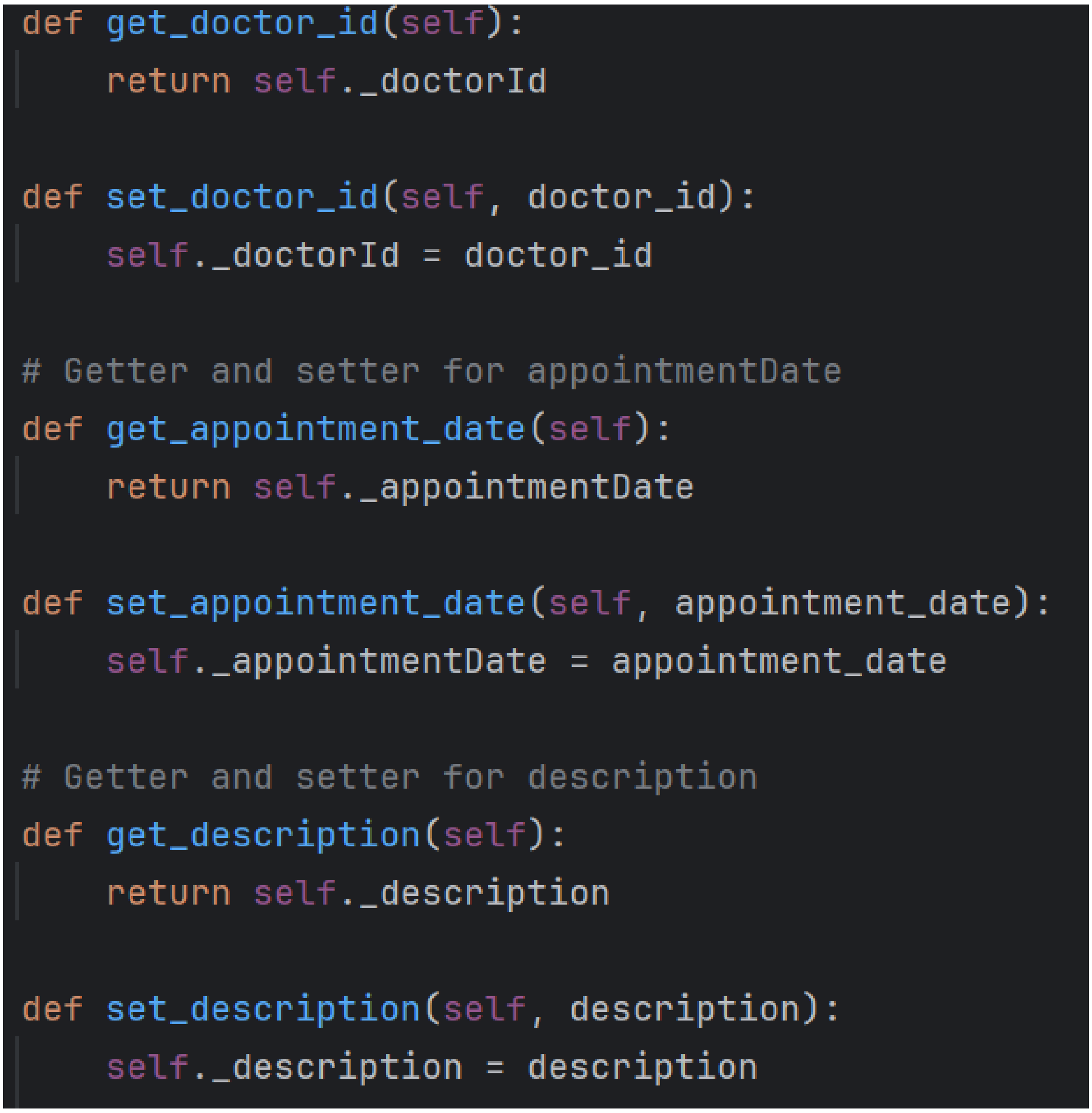
```python
class Patient:
    def __init__(self, patientId, firstName, lastName, dateOfBirth, gender, contactNumber, address):
        self._patientId = patientId
        self._firstName = firstName
        self._lastName = lastName
        self._dateOfBirth = dateOfBirth
        self._gender = gender
        self._contactNumber = contactNumber
        self._address = address

    def display_patient_info(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}")
```



```python
class Doctor:
    def __init__(self, doctorId, firstName, lastName, specialization, contactNumber):
        self._doctorId = doctorId
        self._firstName = firstName
        self._lastName = lastName
        self._specialization = specialization
        self._contactNumber = contactNumber

    def display_doctor_info(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}")
```

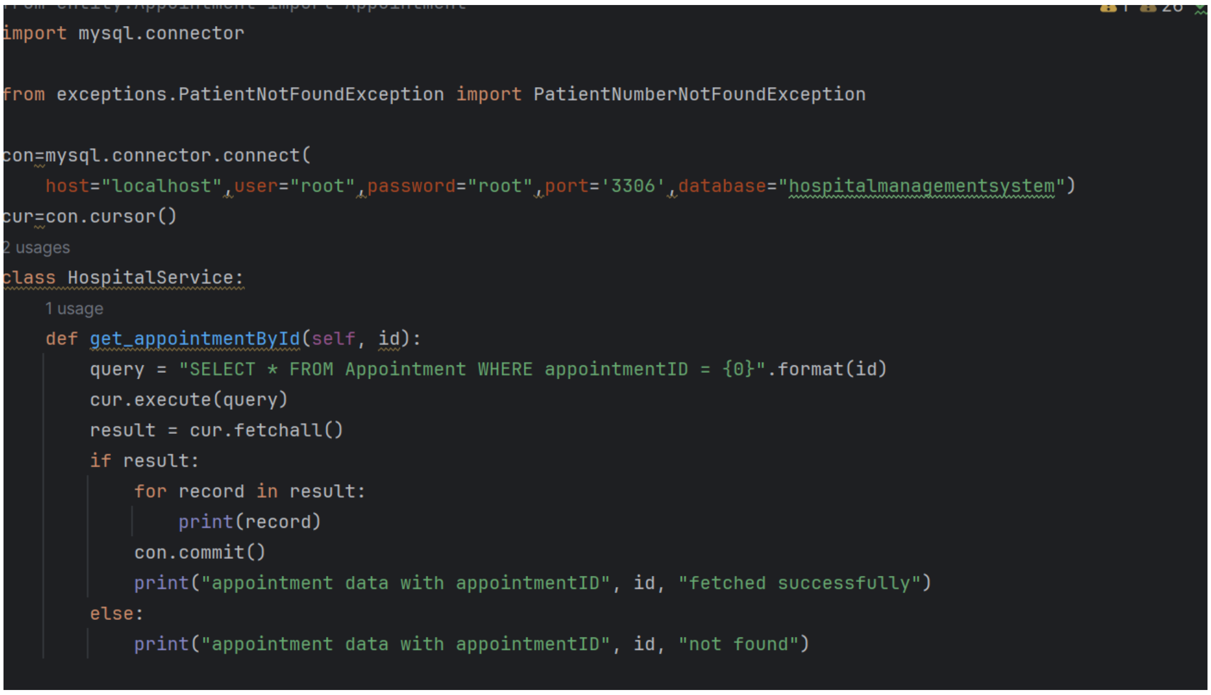


```python
class Appointment:
    def __init__(self):
        self._appointmentId = None
        self._patientId = None
        self._doctorId = None
        self._appointmentDate = None
        self._description = None

    # Getter and setter for appointmentId
    def get_appointment_id(self):
        return self._appointmentId

    def set_appointment_id(self, appointment_id):
        self._appointmentId = appointment_id

    # Getter and setter for patientId
    def get_patient_id(self):
        return self._patientId

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

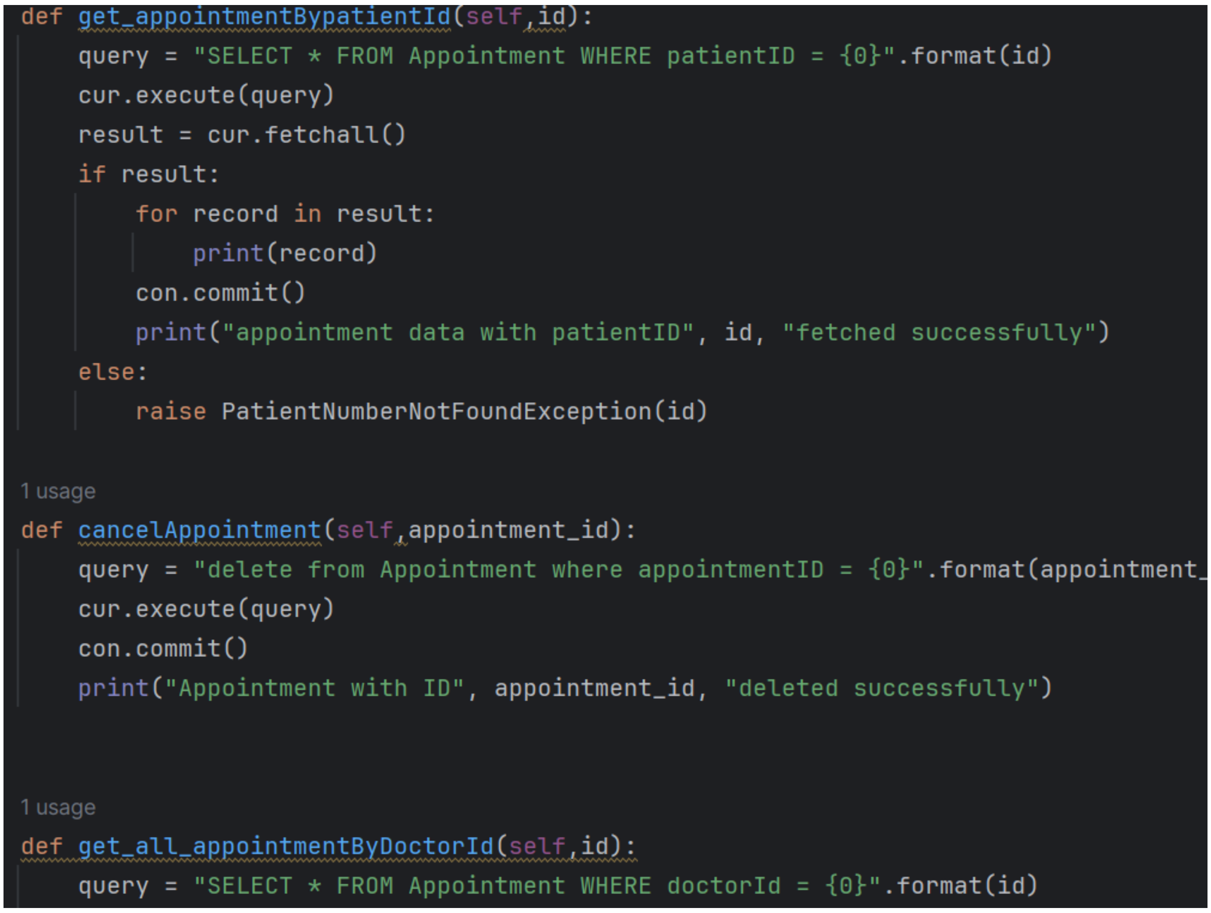


```
def get_doctor_id(self):
    return self._doctorId

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

# Getter and setter for appointmentDate
def get_appointment_date(self):
    return self._appointmentDate

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

# Getter and setter for description
def get_description(self):
    return self._description

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

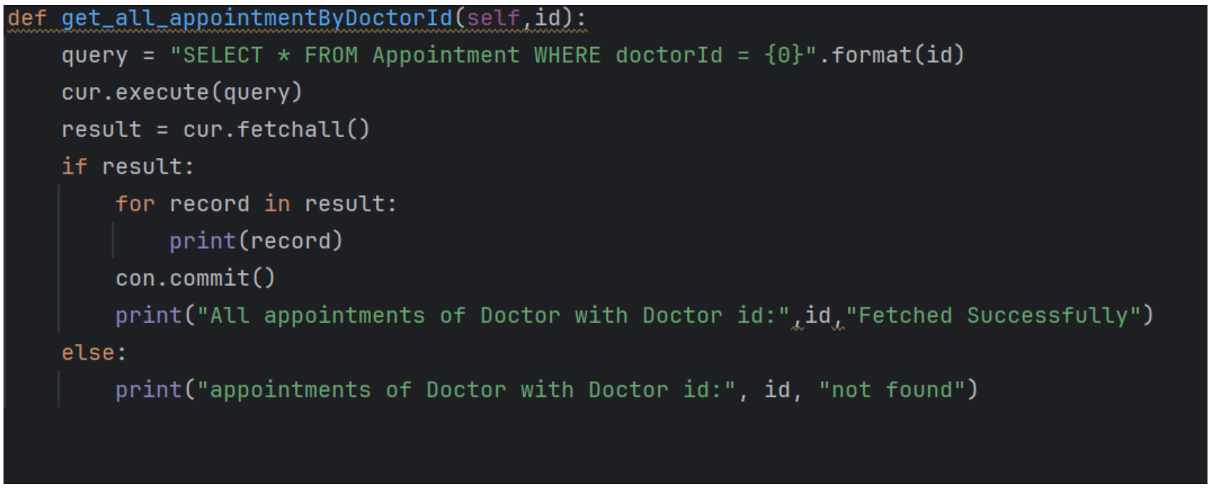
Define **HospitalServiceImpl** class and implement all  the methods  I**HospitalServiceImpl**



```
from entity.Appointment import Appointment
import mysql.connector

from exceptions.PatientNotFoundException import PatientNumberNotFoundException

con=mysql.connector.connect(
    host="localhost",user="root",password="root",port='3306',database="hospitalmanagementsystem")
cur=con.cursor()
2 usages
class HospitalService:
    1 usage
    def get_appointmentById(self, id):
        query = "SELECT * FROM Appointment WHERE appointmentID = {0}".format(id)
        cur.execute(query)
        result = cur.fetchall()
        if result:
            for record in result:
                print(record)
            con.commit()
            print("appointment data with appointmentID", id, "fetched successfully")
        else:
            print("appointment data with appointmentID", id, "not found")
```

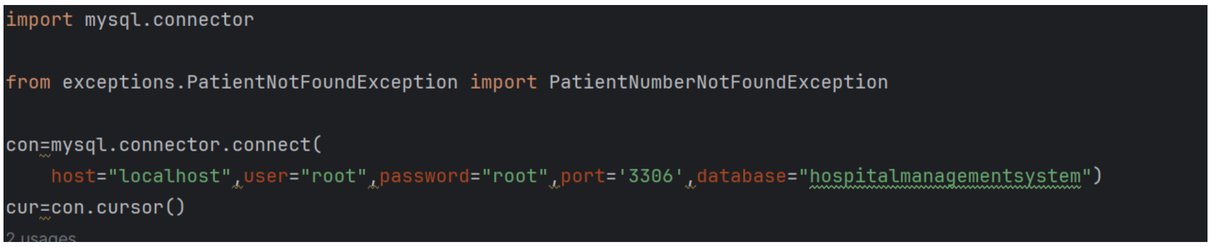


```
def get_appointmentBypatientId(self,id):
    query = "SELECT * FROM Appointment WHERE patientID = {0}".format(id)
    cur.execute(query)
    result = cur.fetchall()
    if result:
        for record in result:
            print(record)
        con.commit()
        print("appointment data with patientID", id, "fetched successfully")
    else:
        raise PatientNumberNotFoundException(id)

1 usage
def cancelAppointment(self,appointment_id):
    query = "delete from Appointment where appointmentID = {0}".format(appointment_
    cur.execute(query)
    con.commit()
    print("Appointment with ID", appointment_id, "deleted successfully")



1 usage
def get_all_appointmentByDoctorId(self,id):
    query = "SELECT * FROM Appointment WHERE doctorId = {0}".format(id)
```



```python
def get_all_appointmentByDoctorId(self,id):
    query = "SELECT * FROM Appointment WHERE doctorId = {0}".format(id)
    cur.execute(query)
    result = cur.fetchall()
    if result:
        for record in result:
            print(record)
        con.commit()
        print("All appointments of Doctor with Doctor id:",id,"Fetched Successfully")
    else:
        print("appointments of Doctor with Doctor id:", id, "not found")
```

Create a utility class **DBConnection** in a package **util** with a static variable **connectin** 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.

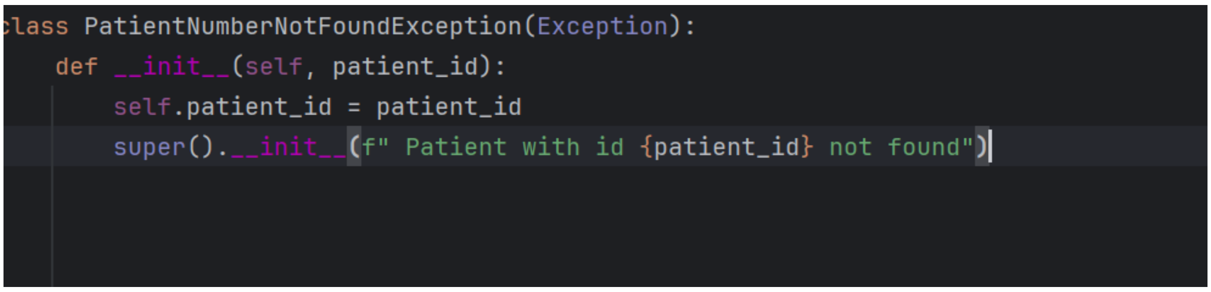


```python
import mysql.connector

from exceptions.PatientNotFoundException import PatientNumberNotFoundException

con=mysql.connector.connect(
    host="localhost",user="root",password="root",port='3306',database="hospitalmanagementsystem")
cur=con.cursor()
2 usages
```

Create the exceptions in package myexceptions

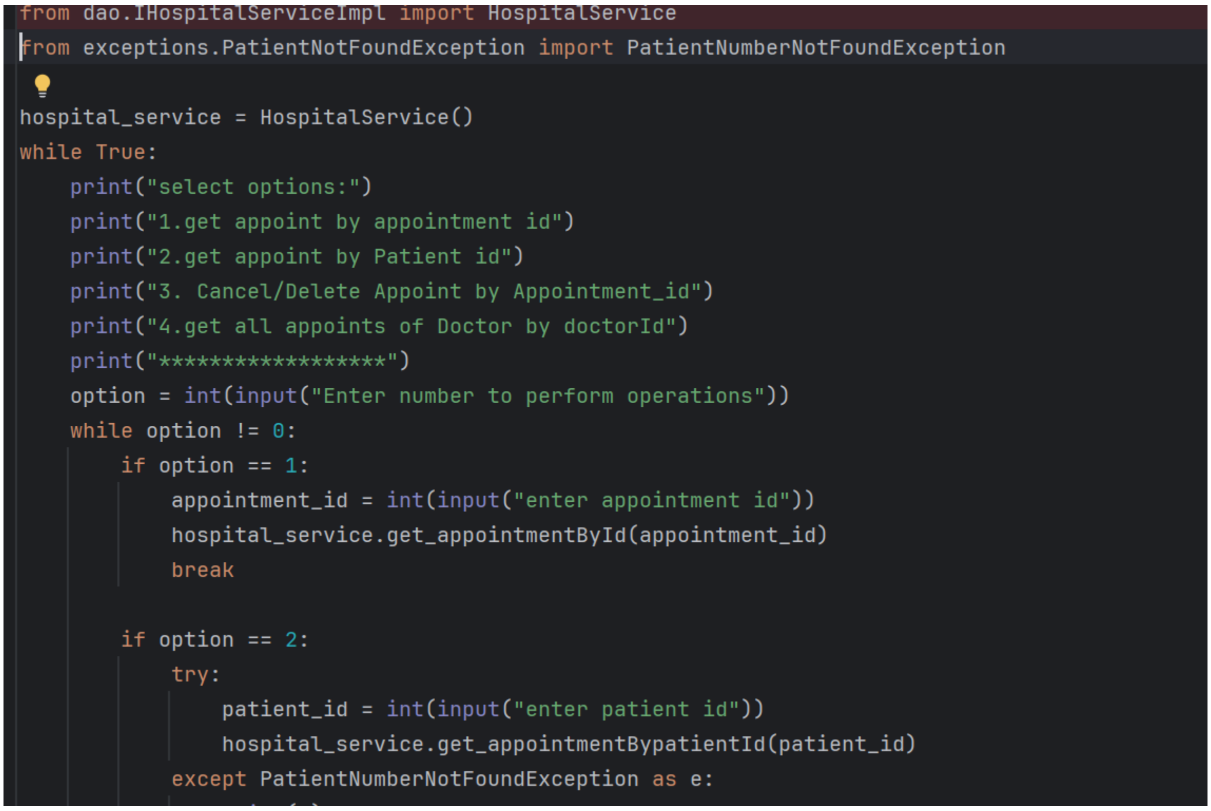
Define the following custom exceptions and throw them in methods whenever needed. Handle all the exceptions in main method,

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

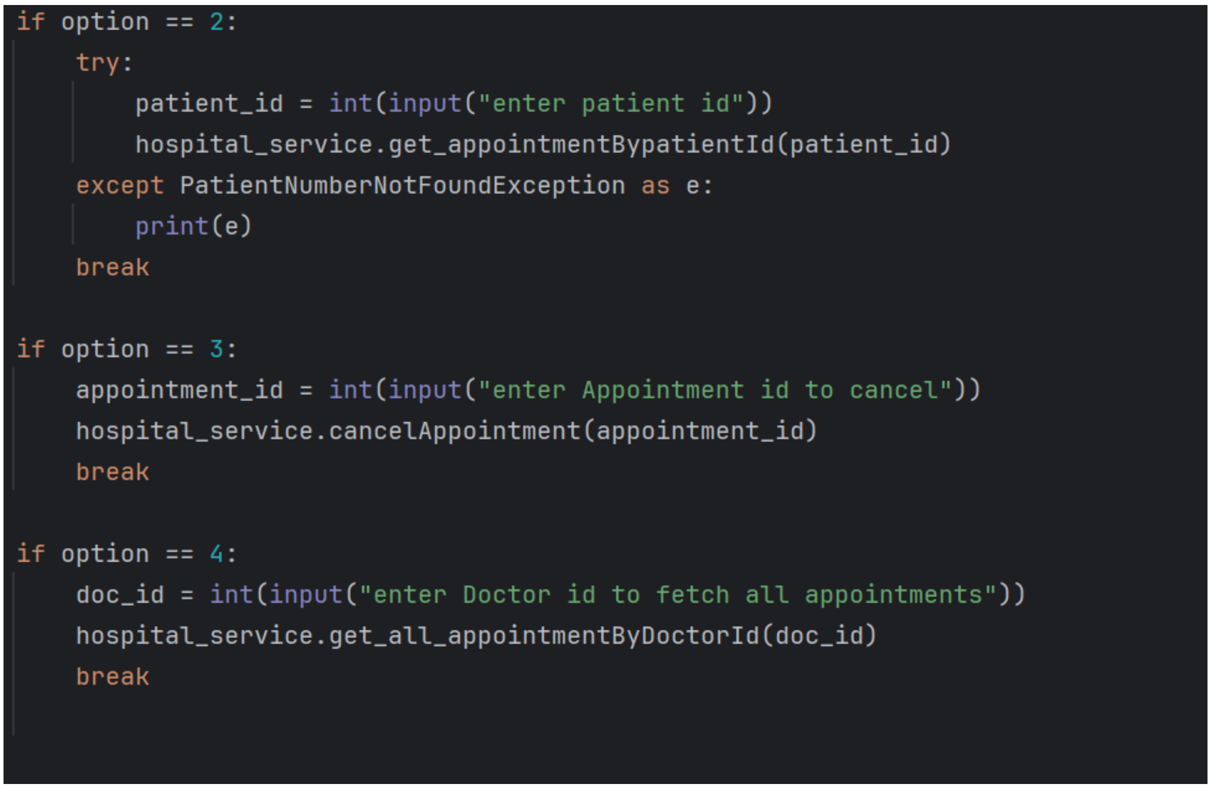


```python
lass PatientNumberNotFoundException(Exception):
    def __init__(self, patient_id):
        self.patient_id = patient_id
        super().__init__(f" Patient with id {patient_id} not found")
```

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

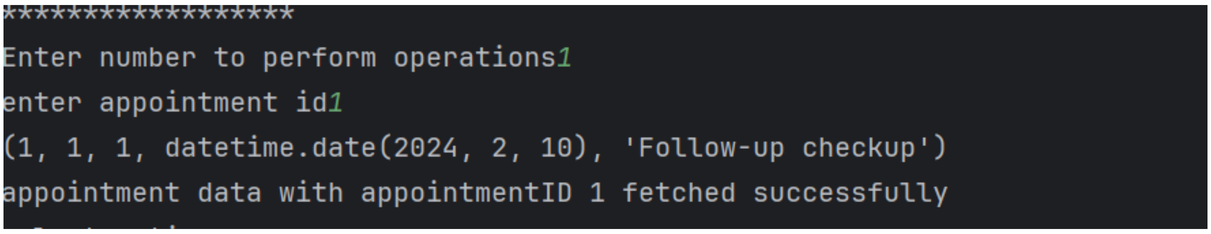


```
from dao.IHospitalServiceImpl import HospitalService
from exceptions.PatientNotFoundException import PatientNumberNotFoundException

hospital_service = HospitalService()
while True:
    print("select options:")
    print("1.get appoint by appointment id")
    print("2.get appoint by Patient id")
    print("3. Cancel/Delete Appoint by Appointment_id")
    print("4.get all appoints of Doctor by doctorId")
    print("*****************")
    option = int(input("Enter number to perform operations"))
    while option != 0:
        if option == 1:
            appointment_id = int(input("enter appointment id"))
            hospital_service.get_appointmentById(appointment_id)
            break

        if option == 2:
            try:
                patient_id = int(input("enter patient id"))
                hospital_service.get_appointmentBypatientId(patient_id)
            except PatientNumberNotFoundException as e:
```



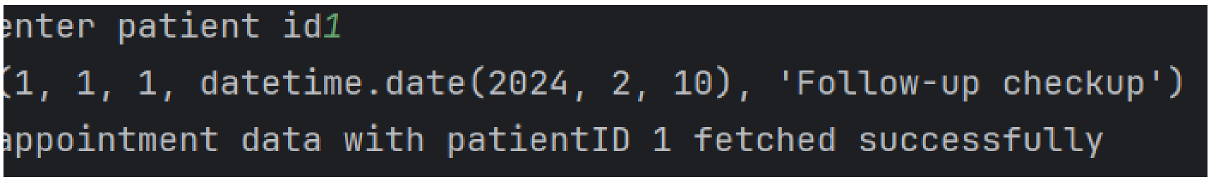
```
if option == 2:
    try:
        patient_id = int(input("enter patient id"))
        hospital_service.get_appointmentBypatientId(patient_id)
    except PatientNumberNotFoundException as e:
        print(e)
    break

if option == 3:
    appointment_id = int(input("enter Appointment id to cancel"))
    hospital_service.cancelAppointment(appointment_id)
    break

if option == 4:
    doc_id = int(input("enter Doctor id to fetch all appointments"))
    hospital_service.get_all_appointmentByDoctorId(doc_id)
    break
```

a. getAppointmentById()
   i. Parameters: appointmentId
   ii. ReturnType: Appointment object



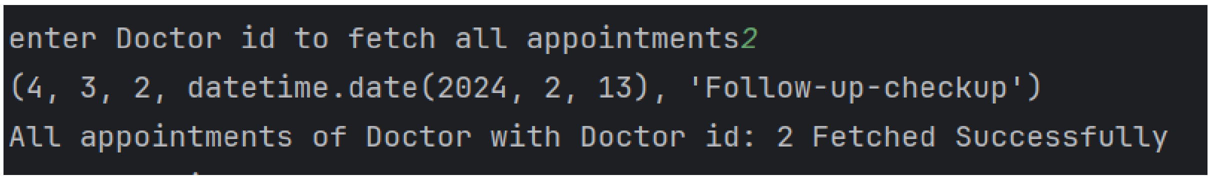
```
******************
Enter number to perform operations1
enter appointment id1
(1, 1, 1, datetime.date(2024, 2, 10), 'Follow-up checkup')
appointment data with appointmentID 1 fetched successfully
```

b. getAppointmentsForPatient()
   i. Parameters: patientId
      ii. ReturnType: List of Appointment objects



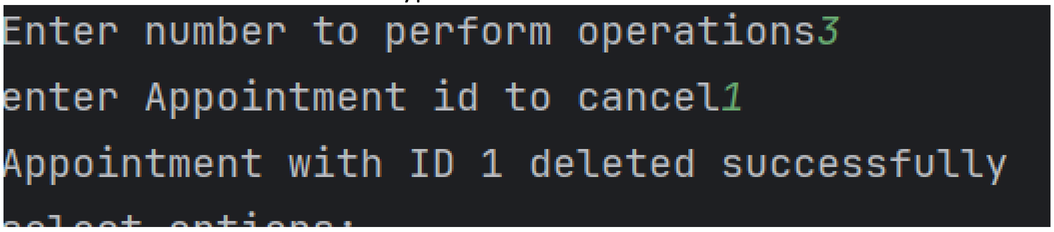
```
enter patient id1
(1, 1, 1, datetime.date(2024, 2, 10), 'Follow-up checkup')
appointment data with patientID 1 fetched successfully
```

c. getAppointmentsForDoctor()
   i. Parameters: doctorId
      ii. ReturnType: List of Appointment objects



```
enter Doctor id to fetch all appointments2
(4, 3, 2, datetime.date(2024, 2, 13), 'Follow-up-checkup')
All appointments of Doctor with Doctor id: 2 Fetched Successfully
```

a. ancelAppointment()
      iii. Parameters: AppointmentId
      iv. ReturnType: Boolean



```
Enter number to perform operations3
enter Appointment id to cancel1
Appointment with ID 1 deleted successfully
```