JAVA Coding Challenge – Hospital Management System

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

   Table: Patient
   create table Patient (patientId int primary key, firstName varchar(50), lastName varchar(50), dateOfBirth date, gender varchar(10), contactNumber varchar(15), address text not null);

   Table: Doctor
   create table Doctor (doctorId int primary key, firstName varchar(50), lastName varchar(50), specialization varchar(100), contactNumber varchar(15));

   Table: Appointment
   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));

```
mysql> create database hospitalManagementSystem;
Query OK, 1 row affected (0.01 sec)

mysql> use hospitalManagementSystem;
Database changed
mysql> create table Patient (patientId int primary key, firstName varchar(50), lastName varchar(50), dateOfBirth date, g
ender varchar(10), contactNumber varchar(15), address text not null);
Query OK, 0 rows affected (0.81 sec)

mysql> create table Doctor (doctorId int primary key, firstName varchar(50), lastName varchar(50), specialization varcha
r(100), contactNumber varchar(15));
Query OK, 0 rows affected (0.48 sec)

mysql> create table Appointment (appointmentId int primary key, patientId int, doctorId int, appointmentdate date, descr
iption varchar(255), foreign key (patientid) references patient(patientId), foreign key (doctorId) references doctor(doc
torId));
Query OK, 0 rows affected (0.59 sec)
```

   NOTE: Since my case study didn't provide instructions to create methods for inserting patient and doctor data, I have manually inserted the data using insert command in MySQL commandline.

```
mysql> insert into Patient values (1, 'Madhu', 'Kalla', '2025-04-09', 'Male', '9876542310', 'Vijayawada, Andhra pradesh'
);
Query OK, 1 row affected (0.01 sec)

mysql> insert into Patient values (2, 'Sai', 'Vighnessh', '2025-04-08', 'Male', '9908287108', 'Tirupati, Andhra pradesh'
);
Query OK, 1 row affected (0.34 sec)

mysql> insert into Patient values (3, 'Shardul', 'Kulkarni', '2025-04-04', 'Male', '9908287108', 'Pune, Maharastra');
Query OK, 1 row affected (0.61 sec)

mysql> insert into Patient values (4, 'Sai', 'Saranya', '2025-03-02', 'Female', '9756874765', 'Tirupati, Andhra pradesh'
);
Query OK, 1 row affected (0.35 sec)

mysql> insert into Patient values (5, 'Kavin', 'Karthick', '2025-02-25', 'Male', '9945786578', 'Chennai, TamilNadu');
Query OK, 1 row affected (0.01 sec)
```

```
mysql> insert into doctor values (101, 'Dr. Prasad', 'Kumar', 'General', '8247821461');
Query OK, 1 row affected (0.36 sec)

mysql> insert into doctor values (102, 'Dr. Shruthi', 'Verma', 'Cardiology', '9784369850');
Query OK, 1 row affected (0.01 sec)

mysql> insert into doctor values (103, 'Dr. Arjun', 'K', 'Neurology', '9908362678');
Query OK, 1 row affected (0.00 sec)
```

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.

Patient.java

```
package entity;
import java.util.*;

public class Patient {
	private int patientId;
	private String firstName;
	private String lastName;
	private Date dateOfBirth;
	private String gender;
	private int contactNumber;
	private String address;
	public Patient() {
		super();
		// TODO Auto-generated constructor stub
	}
	public Patient(int patientId, String firstName, String lastName, Date dateOfBirth,
String gender,int contactNumber,String address) {
		super();
		this.patientId = patientId;
		this.firstName = firstName;
		this.lastName = lastName;
		this.dateOfBirth = dateOfBirth;
		this.gender = gender;
		this.contactNumber = contactNumber;
		this.address=address;
	}
	public int getPatientId() {
		return patientId;
	}
	public void setPatientId(int patientId) {
		this.patientId = patientId;
	}
	public String getFirstName() {
		return firstName;
	}
	public void setFirstName(String firstName) {
		this.firstName = firstName;
	}
	public String getLastName() {
		return lastName;
	}
	public void setLastName(String lastName) {
		this.lastName = lastName;
	}
	public Date getDateOfBirth() {
		return dateOfBirth;
	}
```

```
	public void setDateOfBirth(Date dateOfBirth) {
		this.dateOfBirth = dateOfBirth;
	}
	public String getGender() {
		return gender;
	}
	public void setGender(String gender) {
		this.gender = gender;
	}
	public int getContactNumber() {
		return contactNumber;
	}
	public void setContactNumber(int contactNumber) {
		this.contactNumber = contactNumber;
	}
	public int getAddress() {
		return contactNumber;
	}
	public void setAddress(String address) {
		this.address = address;
	}
	@Override
	public String toString() {
		return "Patient [patientId=" + patientId + ", firstName=" + firstName + ",
lastName=" + lastName
				+ ", dateOfBirth=" + dateOfBirth + ", gender=" + gender +
", contactNumber=" + contactNumber
				+ ", address=" + address + "]";
	}
}
```

Doctor.java

```
package entity;
public class Doctor {
	private int doctorId;
	private String firstName;
	private String lastName;
	private String specialization;
	private int contactNumber;
	public Doctor() {
		super();
		// TODO Auto-generated constructor stub
	}
	public Doctor(int doctorId, String firstName, String lastName, String specialization, int
contactNumber) {
		super();
		this.doctorId = doctorId;
		this.firstName = firstName;
		this.lastName = lastName;
		this.specialization = specialization;
		this.contactNumber = contactNumber;
```

```
	}
	public int getDoctorId() {
		return doctorId;
	}
	public void setDoctorId(int doctorId) {
		this.doctorId = doctorId;
	}
	public String getFirstName() {
		return firstName;
	}
	public void setFirstName(String firstName) {
		this.firstName = firstName;
	}
	public String getLastName() {
		return lastName;
	}
	public void setLastName(String lastName) {
		this.lastName = lastName;
	}
	public String getSpecialization() {
		return specialization;
	}
	public void setSpecialization(String specialization) {
		this.specialization = specialization;
	}
	public int getContactNumber() {
		return contactNumber;
	}
	public void setContactNumber(int contactNumber) {
		this.contactNumber = contactNumber;
	}
	@Override
	public String toString() {
		return "Doctor [doctorId=" + doctorId + ", firstName=" + firstName + ",
lastName=" + lastName+ ", specialization=" + specialization + ", contactNumber=" +
contactNumber + "]";
	}
}
```

Appointment.java

```
package entity;
import java.util.*;

public class Appointment {
	private int appointmentId;
	private int patientId;
	private int doctorId;
	private Date appointmentDate;
	private String description;
	public Appointment() {
		super();
		// TODO Auto-generated constructor stub
```

```
	}
	public Appointment(int appointmentId, int patientId, int doctorId, Date
appointmentDate, String description) {
		super();
		this.appointmentId = appointmentId;
		this.patientId = patientId;
		this.doctorId = doctorId;
		this.appointmentDate = appointmentDate;
		this.description = description;
	}
	public int getAppointmentId() {
		return appointmentId;
	}
	public void setAppointmentId(int appointmentId) {
		this.appointmentId = appointmentId;
	}
	public int getPatientId() {
		return patientId;
	}
	public void setPatientId(int patientId) {
		this.patientId = patientId;
	}
	public int getDoctorId() {
		return doctorId;
	}
	public void setDoctorId(int doctorId) {
		this.doctorId = doctorId;
	}
	public Date getAppointmentDate() {
		return appointmentDate;
	}
	public void setAppointmentDate(Date appointmentDate) {
		this.appointmentDate = appointmentDate;
	}
	public String getDescription() {
		return description;
	}
	public void setDescription(String description) {
		this.description = description;
	}
	@Override
	public String toString() {
		return "Appointment [appointmentId=" + appointmentId + ", patientId=" +
patientId + ", doctorId=" + doctorId
				+ ", appointmentDate=" + appointmentDate + ",
description=" + description + "]";
	}
}
```

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

IHospitalService.java

```
package dao;
import java.util.*;
import entity.*;
import myexceptions.PatientNumberNotFoundException;
public interface IHospitalService {
        public Appointment getAppointmentById(int appointmentId);
        public HashMap<Integer, Appointment> getAppointmentsForPatient(int patientId)
throws PatientNumberNotFoundException;
        public HashMap<Integer, Appointment> getAppointmentsForDoctor(int doctorId);
        public boolean scheduleAppointment(Appointment appointment);
        public boolean updateAppointment(Appointment appointment);
        public boolean CancelAppointment(int appointmentId);
}
```

HospitalServiceImpl.java

```
package dao;
import entity.*;
import myexceptions.PatientNumberNotFoundException;
import util.DBConnection;
import java.sql.*;
import java.util.*;

public class HospitalServiceImpl implements IHospitalService {
   private Connection con;
   public HospitalServiceImpl() throws SQLException {
      con = DBConnection.getConnection();
   }
   @Override
   public Appointment getAppointmentById(int id) {
      Appointment a = null;
      try {
         PreparedStatement ps = con.prepareStatement("select * from appointment where
appointmentid = ?");
         ps.setInt(1, id);
         ResultSet rs = ps.executeQuery();
         if (rs.next()) {
            a = new Appointment(
               rs.getInt("appointmentid"),
               rs.getInt("patientid"),
               rs.getInt("doctorid"),
               rs.getDate("appointmentdate"),
               rs.getString("description")
            );
         }
      } catch (Exception e) {
         System.out.println("Problem getting appointment");
         e.printStackTrace();
      }
      return a;
```

```
    }
    @Override
    public HashMap<Integer, Appointment> getAppointmentsForPatient(int pid) throws
PatientNumberNotFoundException {
        HashMap<Integer, Appointment> hm = new HashMap<>();
        try {
            PreparedStatement ps = con.prepareStatement("select * from appointment where
patientid = ?");
            ps.setInt(1, pid);
            ResultSet rs = ps.executeQuery();
            while (rs.next()) {
                Appointment a = new Appointment(
                    rs.getInt("appointmentid"),
                    rs.getInt("patientid"),
                    rs.getInt("doctorid"),
                    rs.getDate("appointmentdate"),
                    rs.getString("description")
                );
                hm.put(a.getAppointmentId(), a);
            }
        } catch (Exception e) {
            System.out.println("Problem getting patient appointments");
            e.printStackTrace();
        }
        return hm;
    }
    @Override
    public HashMap<Integer, Appointment> getAppointmentsForDoctor(int did) {
        HashMap<Integer, Appointment> hm = new HashMap<>();
        try {
            PreparedStatement ps = con.prepareStatement("select * from appointment where
doctorid = ?");
            ps.setInt(1, did);
            ResultSet rs = ps.executeQuery();
            while (rs.next()) {
                Appointment a = new Appointment(
                    rs.getInt("appointmentid"),
                    rs.getInt("patientid"),
                    rs.getInt("doctorid"),
                    rs.getDate("appointmentdate"),
                    rs.getString("description")
                );
                hm.put(a.getAppointmentId(), a);
            }
        } catch (Exception e) {
            System.out.println("Problem getting doctor appointments");
            e.printStackTrace();
        }
        return hm;
    }
    @Override
    public boolean scheduleAppointment(Appointment a) {
        boolean done = false;
```

```
        try {
            PreparedStatement ps = con.prepareStatement("insert into appointment values (?, ?,
?, ?, ?)");
            ps.setInt(1, a.getAppointmentId());
            ps.setInt(2, a.getPatientId());
            ps.setInt(3, a.getDoctorId());
            ps.setDate(4, new java.sql.Date(a.getAppointmentDate().getTime()));
            ps.setString(5, a.getDescription());
            ps.executeUpdate();
            done = true;
        } catch (Exception e) {
            System.out.println("Problem scheduling appointment");
            e.printStackTrace();
        }
        return done;
    }
    @Override
    public boolean updateAppointment(Appointment a) {
        boolean updated = false;
        try {
            PreparedStatement ps = con.prepareStatement("update appointment set patientid =
?, doctorid = ?, appointmentdate = ?, description = ? where appointmentid = ?");
            ps.setInt(1, a.getPatientId());
            ps.setInt(2, a.getDoctorId());
            ps.setDate(3, new java.sql.Date(a.getAppointmentDate().getTime()));
            ps.setString(4, a.getDescription());
            ps.setInt(5, a.getAppointmentId());
            ps.executeUpdate();
            updated = true;
        } catch (Exception e) {
            System.out.println("Couldn't updating appointment");
            e.printStackTrace();
        }
        return updated;
    }
    @Override
    public boolean CancelAppointment(int id) {
        boolean deleted = false;
        try {
            PreparedStatement ps = con.prepareStatement("delete from appointment where
appointmentid = ?");
            ps.setInt(1, id);
            ps.executeUpdate();
            deleted = true;
        } catch (Exception e) {
            System.out.println("Couldn't cancel your appointment!");
            e.printStackTrace();
        }
        return deleted;
    }
    public void close() {
        try {
            if (con != null && !con.isClosed()) {
```

```
            con.close();
        }
      } catch (Exception e) {
        System.out.println("Problem closing connection");
        e.printStackTrace();
      }
    }
}
```

4. **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.**

db.properties

```
user=root
password=b14163709
port=3306
system=localhost
protocol=jdbc:mysql:
database=hospitalManagementSystem
```

PropertyUtil.java

```
import java.io.FileInputStream;
import java.io.IOException;
import java.util.Properties;

public class PropertyUtil {
        public static String getPropertyString(String filename) throws IOException{
                String constr=null;
                Properties prop=new Properties();
                FileInputStream fis=new FileInputStream(filename);
                prop.load(fis);
                String user=prop.getProperty("user");
                String password=prop.getProperty("password");
                String port=prop.getProperty("port");
                String protocol=prop.getProperty("protocol");
                String system=prop.getProperty("system");
                String database=prop.getProperty("database");

        constr=protocol+"//"+system+":"+port+"/"+database+"?"+"user="+user+"&passw
ord="+password;
                return constr;
  }
}
```

DBConnection.java

```
package util;
import java.io.IOException;
import java.sql.Connection;
```

```
import java.sql.DriverManager;
import java.sql.SQLException;

public class DBConnection {
        private static String filename="db.properties";
        public static Connection getConnection() {
                Connection con=null;
                String constr=null;
                try {
                        constr=PropertyUtil.getPropertyString(filename);
                }
                catch(IOException e){
                        System.out.println("Failed to generate the connection string");
                }
                if(constr!=null) {
                    try {
                     con=DriverManager.getConnection(constr);
                    }catch (SQLException e) {
                     System.out.println("Database connection failed!");
                     e.printStackTrace();
                    }
            }
               return con;
        }
}
```

5. Create the exceptions in package myexceptions

PatientNumberNotFoundException.java

```
package myexceptions;

public class PatientNumberNotFoundException extends Exception {

        public PatientNumberNotFoundException(String m) {

                super(m);

                // TODO Auto-generated constructor stub

        }

}
```

6. Create class named MainModule with main method in package mainmod.

MainModule.java

```
package mainmod;
import dao.*;
import entity.*;
import java.sql.Date;
import java.util.*;
```

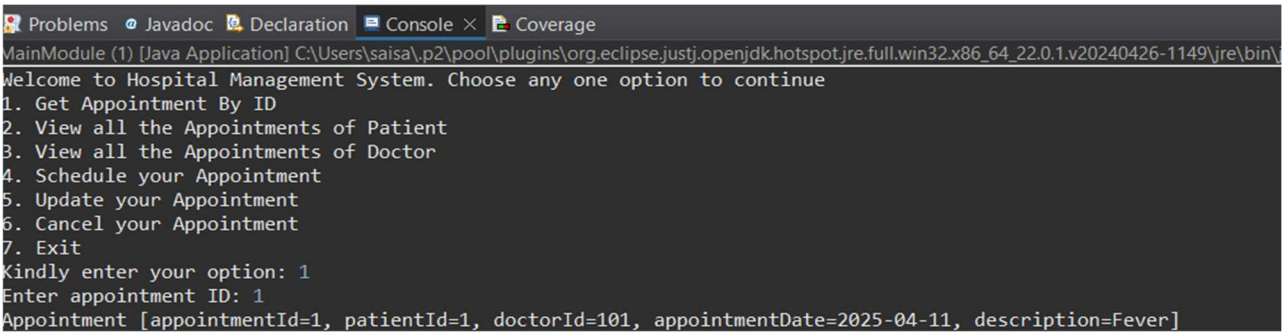
```
public class MainModule {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        IHospitalService service = null;
        try {
            service = new HospitalServiceImpl();
            boolean running = true;
            while (running) {
                System.out.println("Welcome to Hospital Management System. Choose any one
option to continue");
                System.out.println("1. Get Appointment By ID");
                System.out.println("2. View all the Appointments of Patient");
                System.out.println("3. View all the Appointments of Doctor");
                System.out.println("4. Schedule your Appointment");
                System.out.println("5. Update your Appointment");
                System.out.println("6. Cancel your Appointment");
                System.out.println("7. Exit");
                System.out.print("Kindly enter your option: ");
                int key = scan.nextInt();
                switch (key) {
                    case 1 -> {
                        try {
                            System.out.print("Enter appointment ID: ");
                            int id = scan.nextInt();
                            Appointment app = service.getAppointmentById(id);
                            if (app != null) {
                                System.out.println(app);
                            } else {
                                System.out.println("Appointment not found.");
                            }
                        } catch (InputMismatchException e) {
                            System.out.println("Invalid appointment ID format.");
                            scan.nextLine();
                        }
                    }
                    case 2 -> {
                        try {
                            System.out.print("Enter patient ID: ");
                            int pid = scan.nextInt();
                            HashMap<Integer, Appointment>
hm=service.getAppointmentsForPatient(pid);
                            Collection<Appointment> apps = hm.values();
                            System.out.println("Appointments of patient:");
                            for (Appointment a : apps) {
                                System.out.println(a);
                            }
                        } catch (Exception e) {
                            System.out.println("Couldn't ftech the patient appointments.");
                            e.printStackTrace();
                        }
                    }
                    case 3 -> {
                        try {
```

```
                        System.out.print("Enter doctor ID: ");
                        int did = scan.nextInt();
                        HashMap<Integer, Appointment> hm =
service.getAppointmentsForDoctor(did);
                        Collection<Appointment> apps = hm.values();
                        System.out.println("Appointments for doctor:");
                        for (Appointment a : apps) {
                            System.out.println(a);
                        }
                    } catch (Exception e) {
                        System.out.println("Couldn't fetch doctor appointments.");
                        e.printStackTrace();
                    }
                }
                case 4 -> {
                    try {
                        System.out.print("Enter appointment ID: ");
                        int id = scan.nextInt();
                        System.out.print("Enter patient ID: ");
                        int pid = scan.nextInt();
                        System.out.print("Enter doctor ID: ");
                        int did = scan.nextInt();
                        scan.nextLine();
                        System.out.print("Enter appointment date (yyyy-mm-dd): ");
                        Date date = Date.valueOf(scan.nextLine());
                        System.out.print("Enter description: ");
                        String desc = scan.nextLine();
                        Appointment a = new Appointment(id, pid, did, date, desc);
                        if (service.scheduleAppointment(a)) {
                            System.out.println("Appointment has scheduled successfully.");
                        } else {
                            System.out.println("Failed to schedule an appointment.");
                        }
                    } catch (Exception e) {
                        System.out.println("Something went wrong while scheduling.");
                        e.printStackTrace();
                    }
                }
                case 5 -> {
                    try {
                        System.out.print("Enter appointment ID to update: ");
                        int id = scan.nextInt();
                        System.out.print("Enter patient ID: ");
                        int pid = scan.nextInt();
                        System.out.print("Enter doctor ID: ");
                        int did = scan.nextInt();
                        scan.nextLine();
                        System.out.print("Enter new appointment date (yyyy-mm-dd): ");
                        Date date = Date.valueOf(scan.nextLine());
                        System.out.print("Enter new description: ");
                        String desc = scan.nextLine();
                        Appointment a = new Appointment(id, pid, did, date, desc);
                        if (service.updateAppointment(a)) {
```

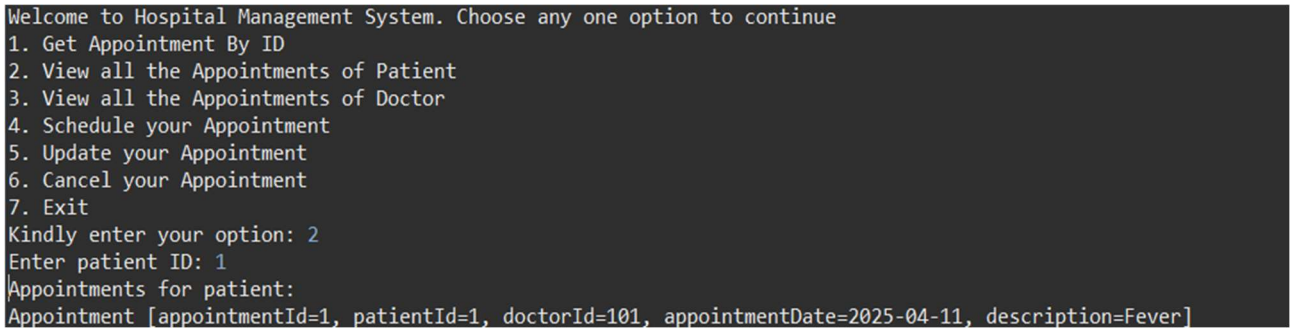
```
                        System.out.println("Appointment has been updated successfully.");
                    } else {
                        System.out.println("Update failed.");
                    }
                } catch (Exception e) {
                    System.out.println("Could not update your appointment.");
                    e.printStackTrace();
                }
            }
            case 6 -> {
                try {
                    System.out.print("Enter appointment ID to cancel: ");
                    int id = scan.nextInt();
                    if (service.CancelAppointment(id)) {
                        System.out.println("Appointment cancelled successfully.");
                    } else {
                        System.out.println("Could not cancel appointment.");
                    }
                } catch (Exception e) {
                    System.out.println("Couldn't cancelling appointment.");
                    e.printStackTrace();
                }
            }
            case 7 -> {
                System.out.println("Thank you!");
                running = false;
            }
            default -> {
                System.out.println("Your option is invalid. Please try again with another
option.");
                scan.nextLine();
            }
        }
    }
} catch (Exception e) {
    System.out.println("Something went wrong. Try again");
    e.printStackTrace();
} finally {
    scan.close();
}
}
}
```

OUTPUT OF HOSPITAL MANAGEMENT SYSTEM

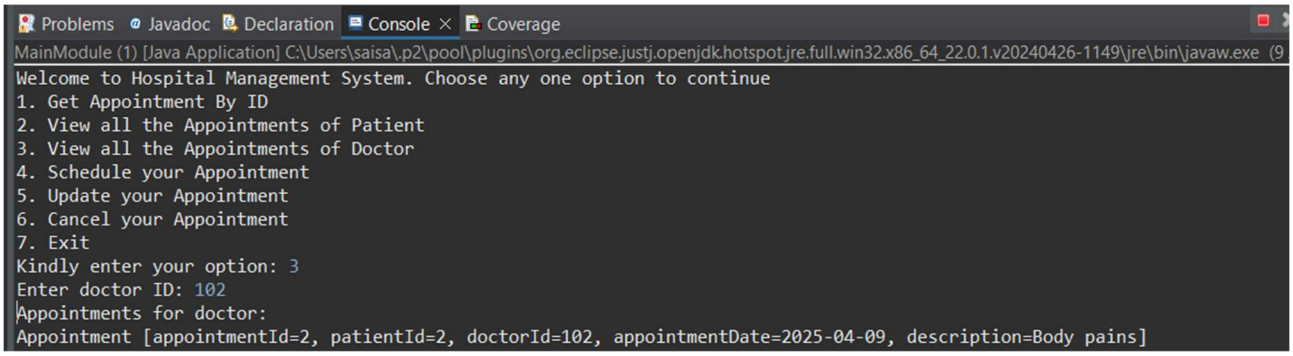
1. Get appointment by ID



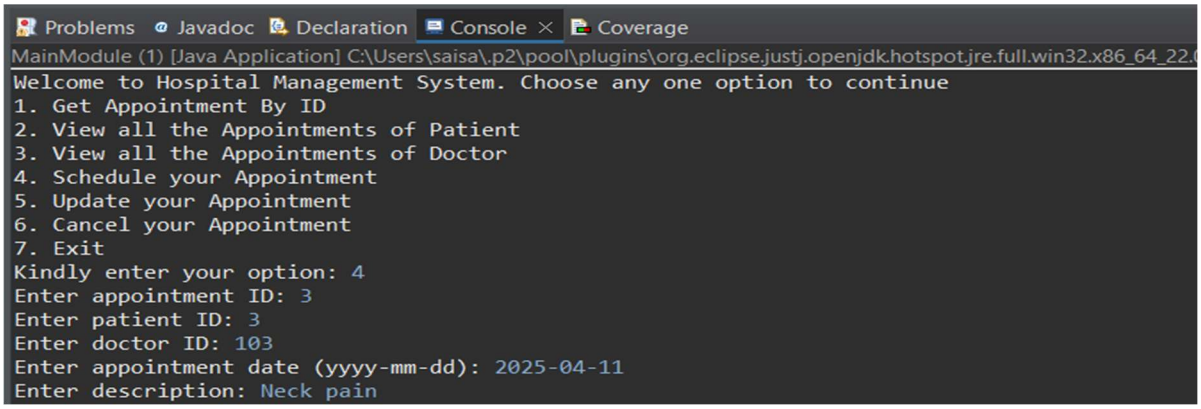
2. View all appointments of patients



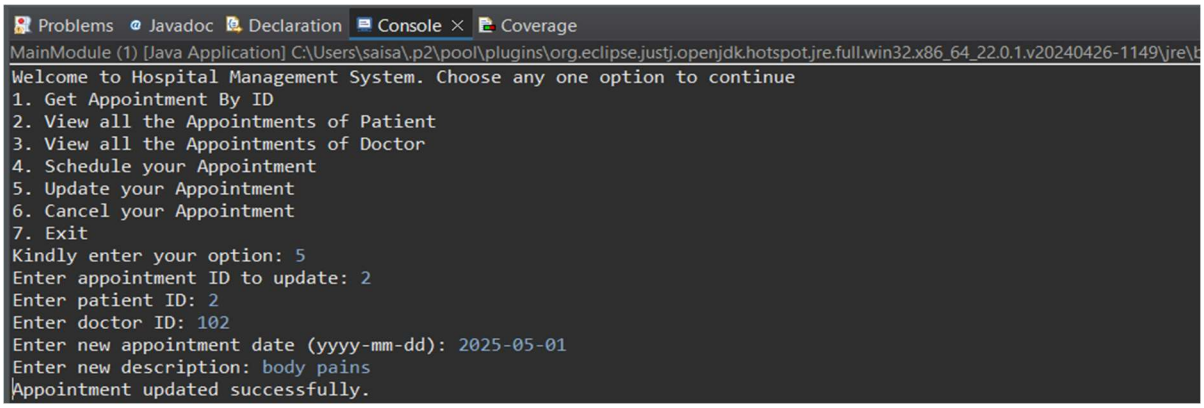
3. View all appointments of Dcotors



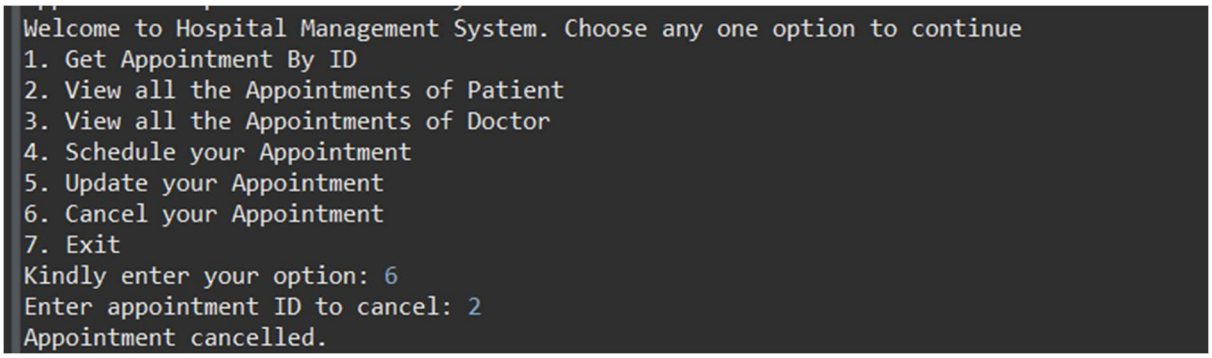
4. Scheduling appointment



5. Updating appointment



6. Cancelling appointment



7. Exit