

Java Coding Challenge – Hospital Management System

Dinesh S

Key Features:

1. **getAppointmentById()** – Retrieves the available appointment details using the appointment ID.
2. **getAppointmentsForPatient()** – Fetches all appointments associated with a specific patient ID.
3. **getAppointmentsForDoctor()** – Retrieves all available appointments for a specific doctor using the doctor ID.
4. **scheduleAppointment()** – Adds a new appointment for the respective patient ID.
5. **updateAppointment()** – Updates the appointment details for the respective patient.
6. **cancelAppointment()** – Cancels or deletes the appointment for the respective patient.

Creating SQL Schema:

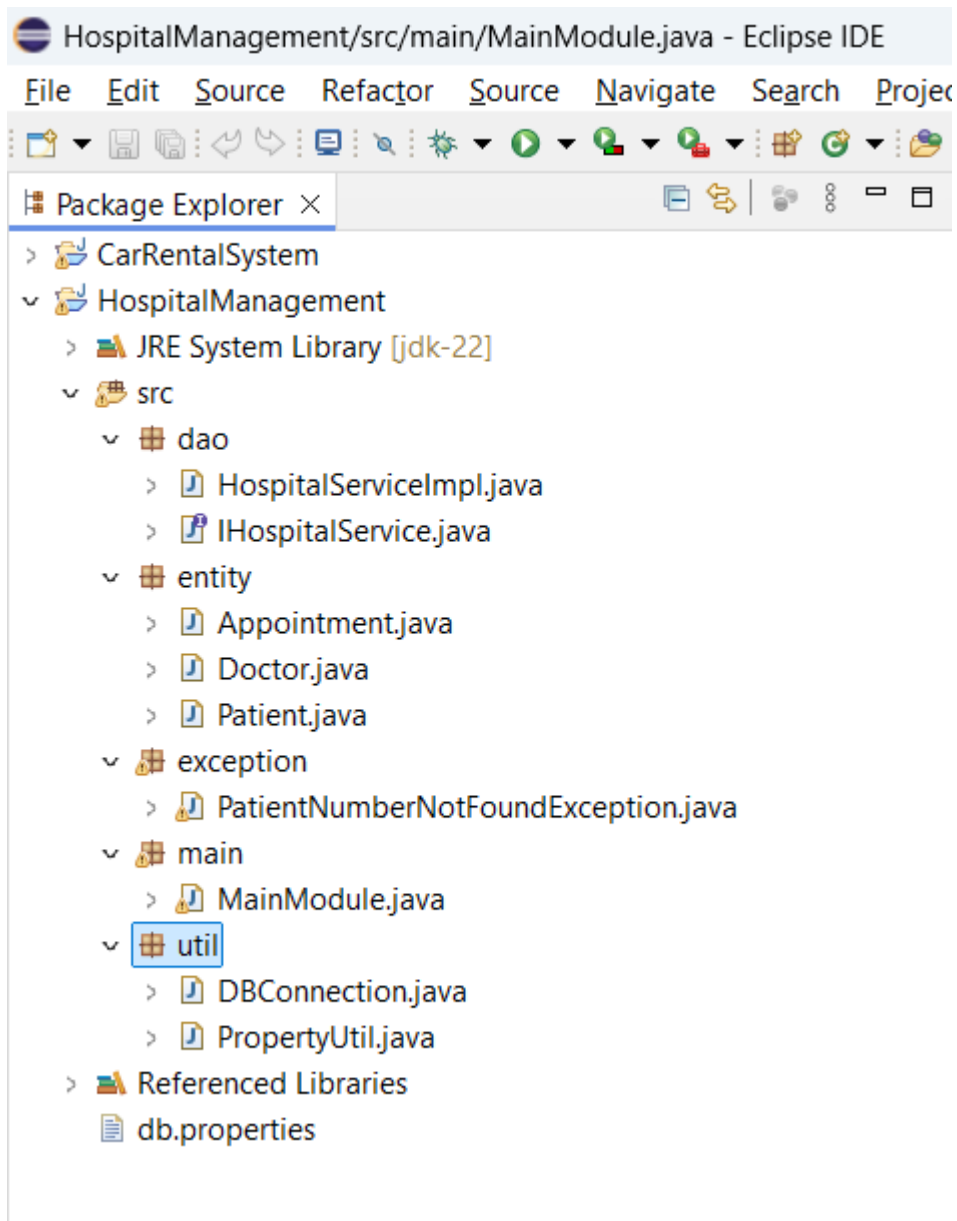
```
mysql> use hospitaldb;
Database changed
mysql> show tables;
+-----+
| Tables_in_hospitaldb |
+-----+
| appointment          |
| doctor               |
| patient              |
+-----+
3 rows in set (0.04 sec)
```

```
mysql> select*from appointment;
+-----+-----+-----+-----+-----+
| appointmentId | patientId | doctorId | appointmentDate | description |
+-----+-----+-----+-----+-----+
| 1 | 1 | 2 | 2024-11-25 | General check-up |
| 3 | 3 | 3 | 2024-11-27 | Knee surgery consultation |
| 4 | 4 | 4 | 2024-11-28 | Skin rash treatment |
| 5 | 5 | 5 | 2024-11-29 | Routine physical exam |
| 13 | 1 | 1 | 2024-12-13 | leg-operation |
| 24 | 5 | 1 | 2025-11-11 | cardiological |
+-----+-----+-----+-----+-----+
6 rows in set (0.02 sec)

mysql> select*from doctor;
+-----+-----+-----+-----+-----+
| doctorId | firstName | lastName | specialization | contactNumber |
+-----+-----+-----+-----+-----+
| 1 | Dr. Adam | Jones | Cardiologist | 1233214321 |
| 2 | Dr. Sarah | Williams | Pediatrician | 2344325432 |
| 3 | Dr. David | Taylor | Orthopedic | 3455436543 |
| 4 | Dr. Olivia | Moore | Dermatologist | 4566547654 |
| 5 | Dr. James | White | General Practitioner | 5677658765 |
+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql> select*from patient;
+-----+-----+-----+-----+-----+-----+-----+
| patientId | firstName | lastName | dateOfBirth | gender | contactNumber | address |
+-----+-----+-----+-----+-----+-----+-----+
| 1 | John | Doe | 1985-06-15 | Male | 1234567890 | 123 Elm Street, City |
| 2 | Jane | Smith | 1990-04-20 | Female | 9876543210 | 456 Oak Avenue, City |
| 3 | Robert | Brown | 1982-12-25 | Male | 5678901234 | 789 Pine Road, City |
| 4 | Emily | Davis | 1995-08-30 | Female | 5432109876 | 321 Maple Boulevard, City |
| 5 | Michael | Wilson | 1978-02-10 | Male | 6789012345 | 654 Birch Lane, City |
+-----+-----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

Hospital Management System Directory Structure



Package – entity

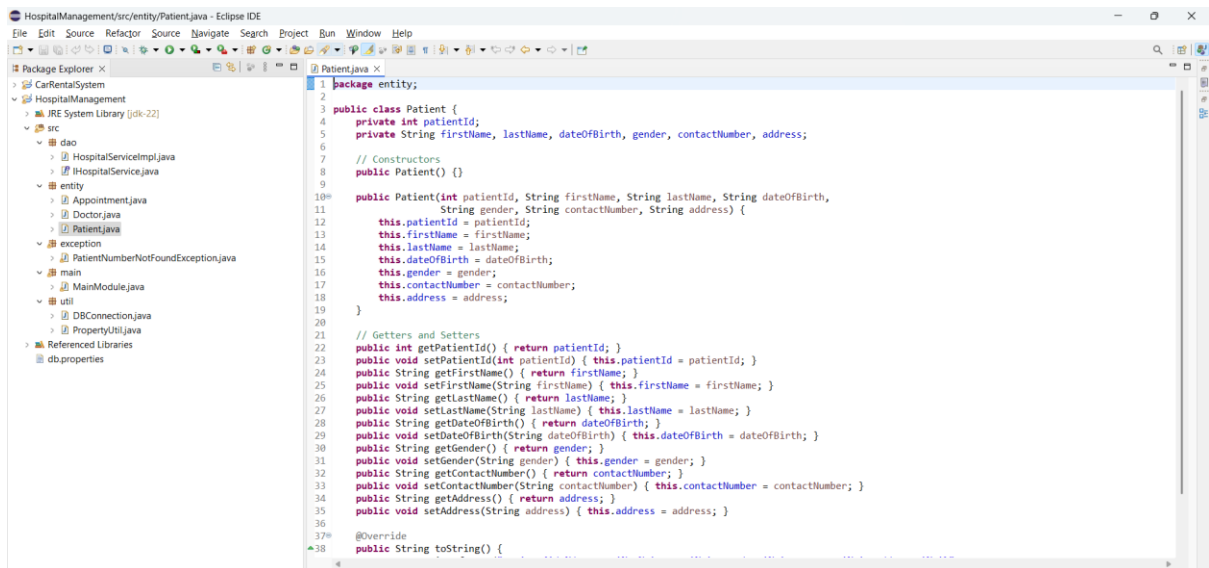
Appointment.java

```
1 package entity;
2
3 public class Appointment {
4     private int appointmentId, patientId, doctorId;
5     private String appointmentDate, description;
6
7     // Constructors
8     public Appointment() {}
9
10    public Appointment(int appointmentId, int patientId, int doctorId, String appointmentDate, String description) {
11        this.appointmentId = appointmentId;
12        this.patientId = patientId;
13        this.doctorId = doctorId;
14        this.appointmentDate = appointmentDate;
15        this.description = description;
16    }
17
18    // Getters and Setters
19    public int getAppointmentId() { return appointmentId; }
20    public void setAppointmentId(int appointmentId) { this.appointmentId = appointmentId; }
21    public int getPatientId() { return patientId; }
22    public void setPatientId(int patientId) { this.patientId = patientId; }
23    public int getDoctorId() { return doctorId; }
24    public void setDoctorId(int doctorId) { this.doctorId = doctorId; }
25    public String getAppointmentDate() { return appointmentDate; }
26    public void setAppointmentDate(String appointmentDate) { this.appointmentDate = appointmentDate; }
27    public String getDescription() { return description; }
28    public void setDescription(String description) { this.description = description; }
29
30    @Override
31    public String toString() {
32        return String.format("Appointment(id=%d, patientId=%d, doctorId=%d, date='%s', description='%s')",
33            appointmentId, patientId, doctorId, appointmentDate, description);
34    }
35 }
36
```

Doctor.java

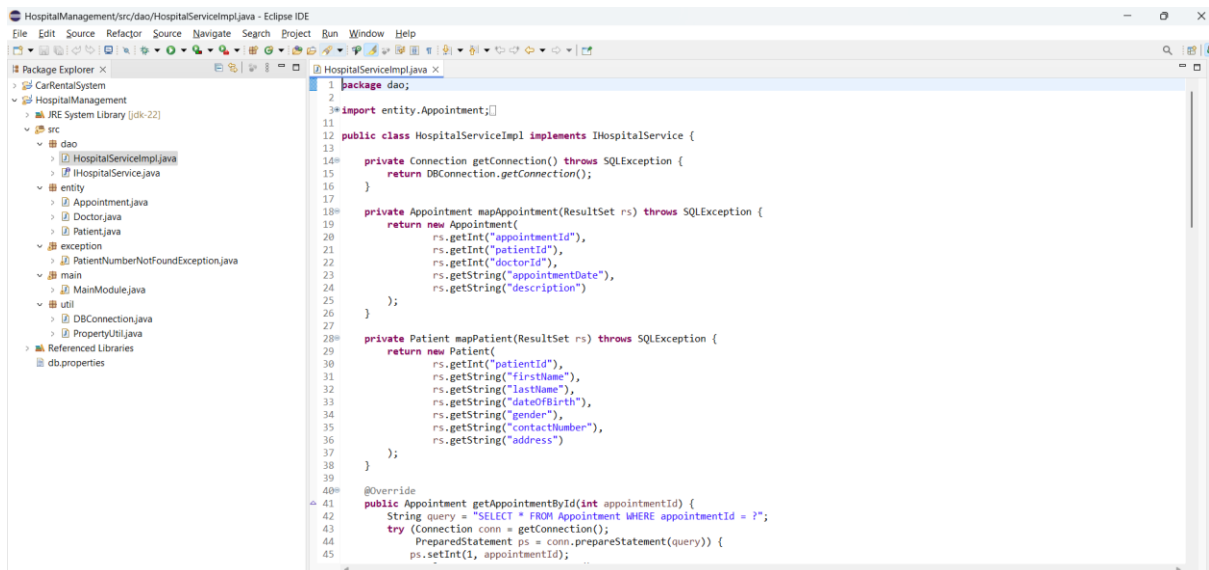
```
1 package entity;
2
3 public class Doctor {
4     private int doctorId;
5     private String firstName, lastName, specialization, contactNumber;
6
7     // Constructors
8     public Doctor() {}
9
10    public Doctor(int doctorId, String firstName, String lastName, String specialization, String contactNumber) {
11        this.doctorId = doctorId;
12        this.firstName = firstName;
13        this.lastName = lastName;
14        this.specialization = specialization;
15        this.contactNumber = contactNumber;
16    }
17
18    // Getters and Setters
19    public int getDoctorId() { return doctorId; }
20    public void setDoctorId(int doctorId) { this.doctorId = doctorId; }
21    public String getFirstName() { return firstName; }
22    public void setFirstName(String firstName) { this.firstName = firstName; }
23    public String getLastName() { return lastName; }
24    public void setLastName(String lastName) { this.lastName = lastName; }
25    public String getSpecialization() { return specialization; }
26    public void setSpecialization(String specialization) { this.specialization = specialization; }
27    public String getContactNumber() { return contactNumber; }
28    public void setContactNumber(String contactNumber) { this.contactNumber = contactNumber; }
29
30    @Override
31    public String toString() {
32        return String.format("Doctor(id=%d, name='%s %s', specialization='%s', contact='%s')",
33            doctorId, firstName, lastName, specialization, contactNumber);
34    }
35 }
36
```

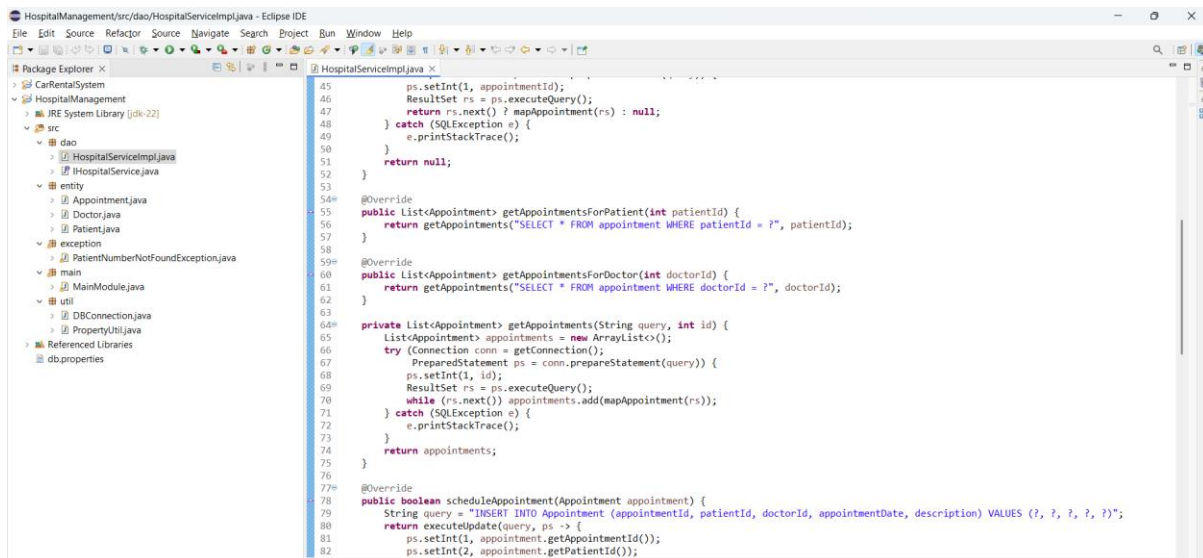
Patient.java



Package – dao

HospitalServiceImpl.java

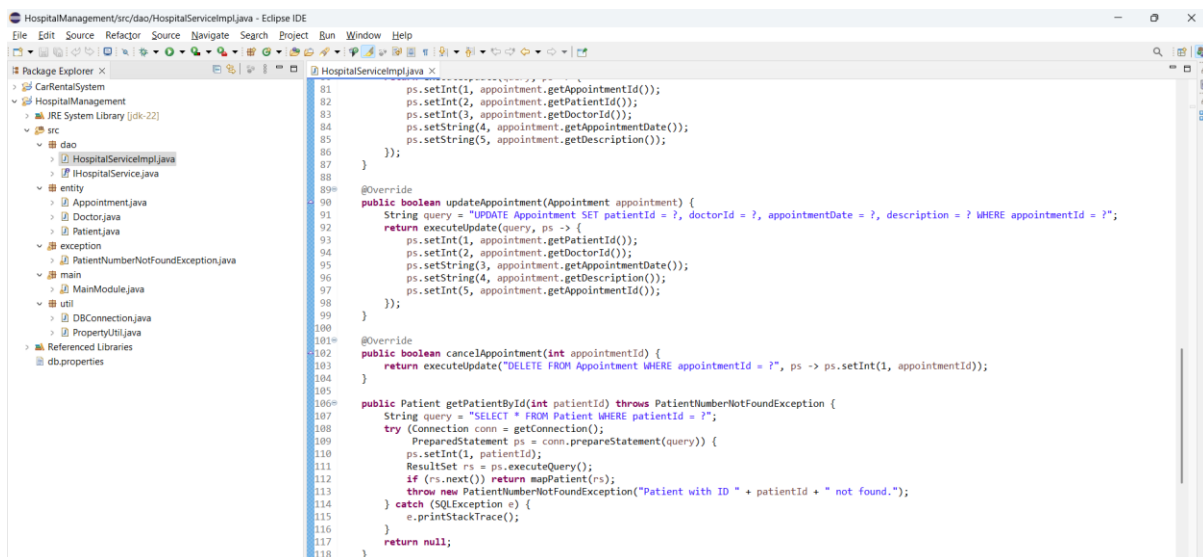




```

45         ps.setInt(1, appointmentId);
46         ResultSet rs = ps.executeQuery();
47         return rs.next() ? mapAppointment(rs) : null;
48     } catch (SQLException e) {
49         e.printStackTrace();
50     }
51     return null;
52 }
53
54 @Override
55 public List<Appointment> getAppointmentsForPatient(int patientId) {
56     return getAppointments("SELECT * FROM appointment WHERE patientId = ?", patientId);
57 }
58
59 @Override
60 public List<Appointment> getAppointmentsForDoctor(int doctorId) {
61     return getAppointments("SELECT * FROM appointment WHERE doctorId = ?", doctorId);
62 }
63
64 private List<Appointment> getAppointments(String query, int id) {
65     List<Appointment> appointments = new ArrayList<>();
66     try (Connection conn = getConnection();
67         PreparedStatement ps = conn.prepareStatement(query)) {
68         ps.setInt(1, id);
69         ResultSet rs = ps.executeQuery();
70         while (rs.next()) appointments.add(mapAppointment(rs));
71     } catch (SQLException e) {
72         e.printStackTrace();
73     }
74     return appointments;
75 }
76
77 @Override
78 public boolean scheduleAppointment(Appointment appointment) {
79     String query = "INSERT INTO Appointment (appointmentId, patientId, doctorId, appointmentDate, description) VALUES (?, ?, ?, ?, ?)";
80     return executeUpdate(query, ps -> {
81         ps.setInt(1, appointment.getAppointmentId());
82         ps.setInt(2, appointment.getPatientId());

```



```

81         ps.setInt(1, appointment.getAppointmentId());
82         ps.setInt(2, appointment.getPatientId());
83         ps.setInt(3, appointment.getDoctorId());
84         ps.setString(4, appointment.getAppointmentDate());
85         ps.setString(5, appointment.getDescription());
86     });
87 }
88
89 @Override
90 public boolean updateAppointment(Appointment appointment) {
91     String query = "UPDATE Appointment SET patientId = ?, doctorId = ?, appointmentDate = ?, description = ? WHERE appointmentId = ?";
92     return executeUpdate(query, ps -> {
93         ps.setInt(1, appointment.getAppointmentId());
94         ps.setInt(2, appointment.getDoctorId());
95         ps.setString(3, appointment.getAppointmentDate());
96         ps.setString(4, appointment.getDescription());
97         ps.setInt(5, appointment.getAppointmentId());
98     });
99 }
100
101 @Override
102 public boolean cancelAppointment(int appointmentId) {
103     return executeUpdate("DELETE FROM Appointment WHERE appointmentId = ?", ps -> ps.setInt(1, appointmentId));
104 }
105
106 private Patient getPatientById(int patientId) throws PatientNumberNotFoundException {
107     String query = "SELECT * FROM Patient WHERE patientId = ?";
108     try (Connection conn = getConnection();
109         PreparedStatement ps = conn.prepareStatement(query)) {
110         ps.setInt(1, patientId);
111         ResultSet rs = ps.executeQuery();
112         if (rs.next()) return mapPatient(rs);
113         throw new PatientNumberNotFoundException("Patient with ID " + patientId + " not found.");
114     } catch (SQLException e) {
115         e.printStackTrace();
116     }
117     return null;
118 }

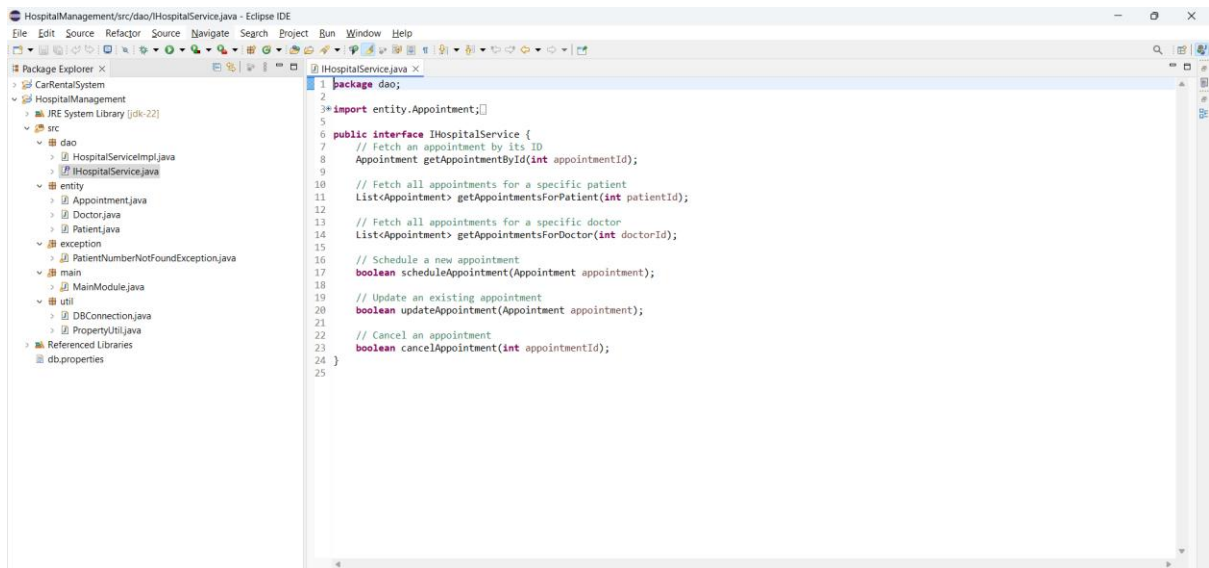
```

```

118     }
119
120 private boolean executeUpdate(String query, ThrowingConsumer<PreparedStatement> consumer) {
121     try (Connection conn = getConnection();
122         PreparedStatement ps = conn.prepareStatement(query)) {
123         consumer.accept(ps);
124         return ps.executeUpdate() > 0;
125     } catch (SQLException e) {
126         e.printStackTrace();
127     }
128     return false;
129 }
130
131 @FunctionalInterface
132 interface ThrowingConsumer<T> {
133     void accept(T t) throws SQLException;
134 }
135 }

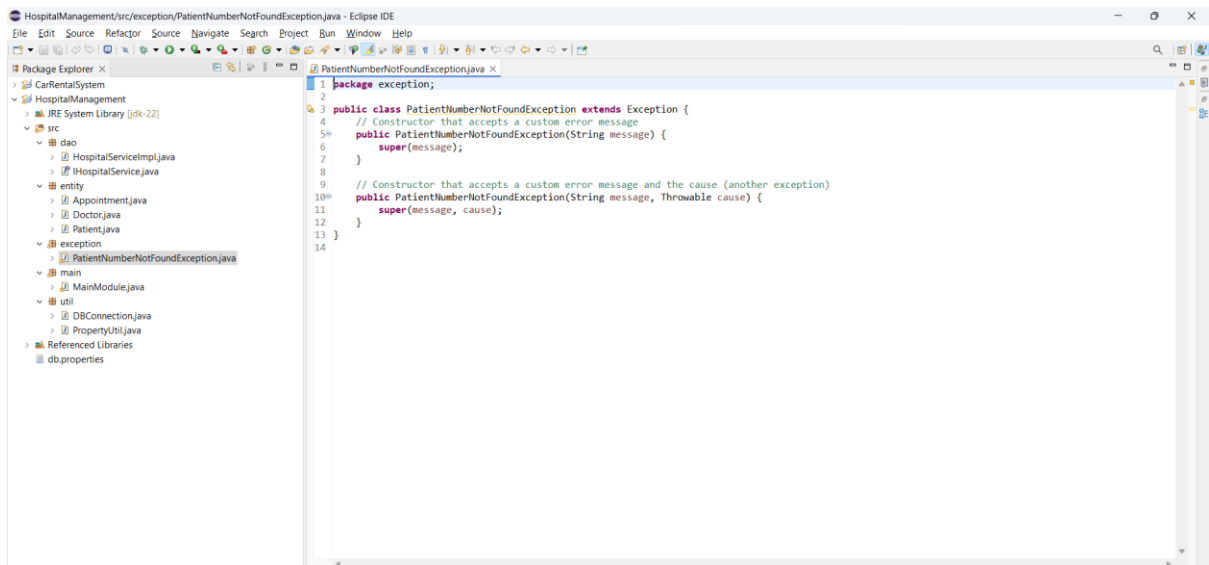
```

IHospitalService.java



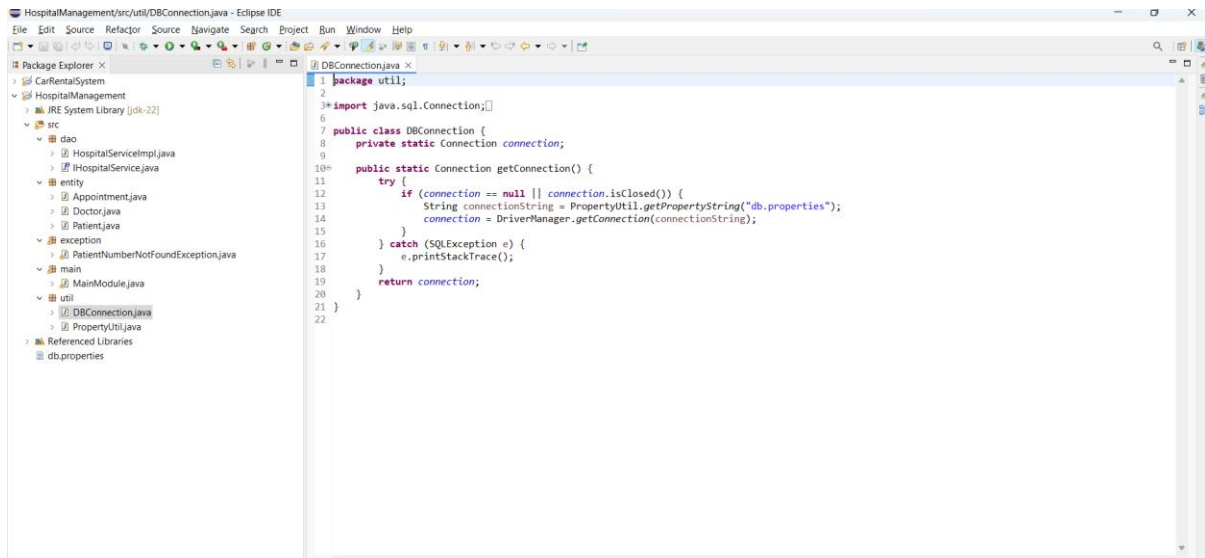
Package – exception

PatientNumberNotFoundException.java

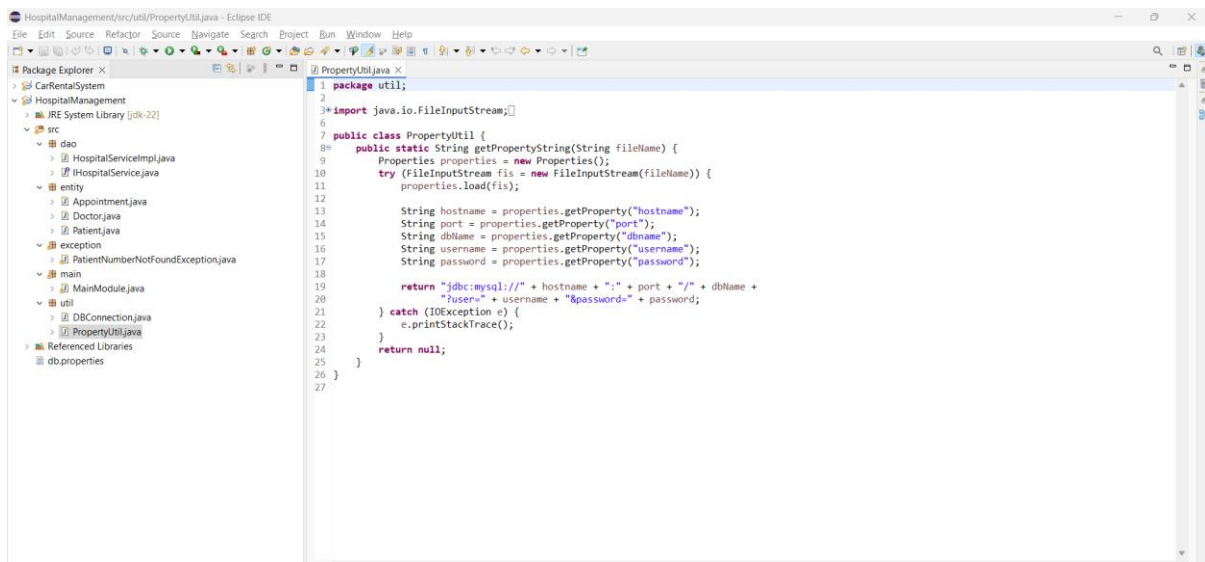


Package – util

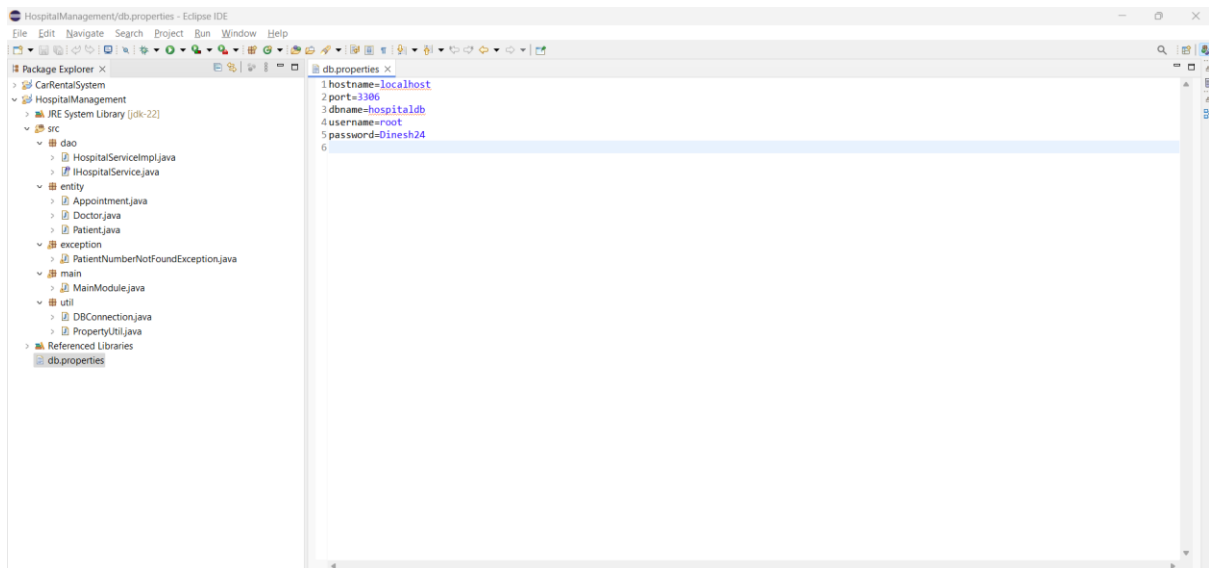
DBConnection.java



PropertyUtil.java

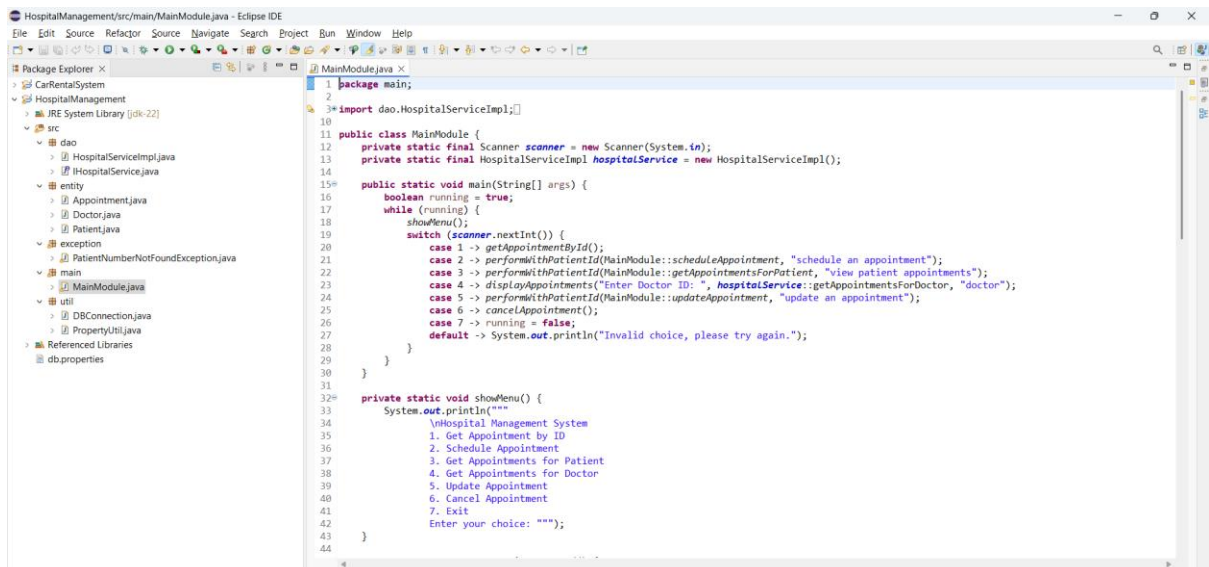


db.properties (outside src folder)



Package – main

MainModule.java



```

43 }
44
45 private static void getAppointmentById() {
46     int appointmentId = getIntInput("Enter Appointment ID: ");
47     Appointment appointment = hospitalService.getAppointmentById(appointmentId);
48     System.out.println(appointment != null ? "Appointment Details: " + appointment : "No appointment found for ID " + appointmentId);
49 }
50
51 private static void scheduleAppointment(int patientId) {
52     Appointment appointment = new Appointment(
53         getIntInput("Enter Appointment ID: "), patientId,
54         getIntInput("Enter Doctor ID: "),
55         getStringInput("Enter Appointment Date (YYYY-MM-DD): "),
56         getStringInput("Enter Appointment Description: ")
57     );
58     System.out.println(hospitalService.scheduleAppointment(appointment)
59         ? "Appointment scheduled successfully!" : "Failed to schedule appointment.");
60 }
61
62 private static void getAppointmentsForPatient(int patientId) {
63     List<Appointment> appointments = hospitalService.getAppointmentsForPatient(patientId);
64     if (appointments.isEmpty()) {
65         System.out.printf("No appointments found for patient ID %d.\n", patientId);
66     } else {
67         System.out.printf("Appointments for patient ID %d:\n", patientId);
68         appointments.forEach(System.out::println);
69     }
70 }
71
72 private static void updateAppointment(int patientId) {
73     Appointment updatedAppointment = new Appointment(
74         getIntInput("Enter Appointment ID to Update: "), patientId,
75         getIntInput("Enter new Doctor ID: "),
76         getStringInput("Enter new Appointment Date (YYYY-MM-DD): "),
77         getStringInput("Enter new Appointment Description: ")
78     );
79     System.out.println(hospitalService.updateAppointment(updatedAppointment)
80         ? "Appointment updated successfully!" : "Failed to update appointment.");
81 }

```

```

private static void cancelAppointment() {
    System.out.println(hospitalService.cancelAppointment(getIntInput("Enter Appointment ID to Cancel: "))
        ? "Appointment cancelled successfully!" : "Failed to cancel appointment.");
}

```

```

85
86 private static void performWithPatientId(java.util.function.Consumer<Integer> action, String operation) {
87     try {
88         int patientId = getIntInput("Enter Patient ID: ");
89         validatePatientId(patientId);
90         action.accept(patientId);
91     } catch (PatientNumberNotFoundException e) {
92         System.out.println("Error: " + e.getMessage());
93     }
94 }
95
96 private static int getIntInput(String prompt) {
97     System.out.print(prompt);
98     return scanner.nextInt();
99 }
100
101 private static String getStringInput(String prompt) {
102     System.out.print(prompt);
103     return scanner.next();
104 }
105
106 private static void validatePatientId(int patientId) throws PatientNumberNotFoundException {
107     if (hospitalService.getPatientById(patientId) == null) {
108         throw new PatientNumberNotFoundException("Patient with ID " + patientId + " not found.");
109     }
110 }
111
112 private static void displayAppointments(String prompt, java.util.function.Function<Integer, List<Appointment>> fetchAppointments, String type) {
113     int id = getIntInput(prompt);
114     List<Appointment> appointments = fetchAppointments.apply(id);
115     if (appointments.isEmpty()) {
116         System.out.printf("No appointments found for this %s.\n", type);
117     } else {
118         System.out.printf("Appointments for %s ID %d:\n", type, id);
119         appointments.forEach(System.out::println);
120     }
121 }
122

```

Output – Run as java application in MainModule:

1)

```

Hospital Management System
1. Get Appointment by ID
2. Schedule Appointment
3. Get Appointments for Patient
4. Get Appointments for Doctor
5. Update Appointment
6. Cancel Appointment
7. Exit
Enter your choice:
1
Enter Appointment ID: 1
Appointment Details: Appointment{id=1, patientId=1, doctorId=2, date='2024-11-25', description='General check-up'}

```

2)

```
Hospital Management System
1. Get Appointment by ID
2. Schedule Appointment
3. Get Appointments for Patient
4. Get Appointments for Doctor
5. Update Appointment
6. Cancel Appointment
7. Exit
Enter your choice:
2
Enter Patient ID: 5
Enter Appointment ID: 20
Enter Doctor ID: 1
Enter Appointment Date (YYYY-MM-DD): 2025-10-01
Enter Appointment Description: General
Appointment scheduled successfully!
```

3)

```
Hospital Management System
1. Get Appointment by ID
2. Schedule Appointment
3. Get Appointments for Patient
4. Get Appointments for Doctor
5. Update Appointment
6. Cancel Appointment
7. Exit
Enter your choice:
3
Enter Patient ID: 5
Appointments for patient ID 5:
Appointment{id=5, patientId=5, doctorId=5, date='2024-11-29', description='Routine physical exam'}
Appointment{id=20, patientId=5, doctorId=1, date='2025-10-01', description='General'}
Appointment{id=24, patientId=5, doctorId=1, date='2025-11-11', description='cardiological'}
```

4)

```
Hospital Management System
1. Get Appointment by ID
2. Schedule Appointment
3. Get Appointments for Patient
4. Get Appointments for Doctor
5. Update Appointment
6. Cancel Appointment
7. Exit
Enter your choice:
4
Enter Doctor ID: 1
Appointments for doctor ID 1:
Appointment{id=13, patientId=1, doctorId=1, date='2024-12-13', description='leg-operation'}
Appointment{id=20, patientId=5, doctorId=1, date='2025-10-01', description='General'}
Appointment{id=24, patientId=5, doctorId=1, date='2025-11-11', description='cardiological'}
```

5)

```
Hospital Management System
1. Get Appointment by ID
2. Schedule Appointment
3. Get Appointments for Patient
4. Get Appointments for Doctor
5. Update Appointment
6. Cancel Appointment
7. Exit
Enter your choice:
5
Enter Patient ID: 5
Enter Appointment ID to Update: 20
Enter new Doctor ID: 1
Enter new Appointment Date (YYYY-MM-DD): 2025-10-22
Enter new Appointment Description: OutPatient
Appointment updated successfully!
```

6)

```
Hospital Management System
1. Get Appointment by ID
2. Schedule Appointment
3. Get Appointments for Patient
4. Get Appointments for Doctor
5. Update Appointment
6. Cancel Appointment
7. Exit
Enter your choice:
6
Enter Appointment ID to Cancel: 20
Appointment cancelled successfully!
```