

# Rajalakshmi Engineering College

Name: sri sidharthan .T  
Email: 241501210@rajalakshmi.edu.in  
Roll no: 241501210  
Phone: 9344823470  
Branch: REC  
Department: AI & ML - Section 4  
Batch: 2028  
Degree: B.E - AI & ML

Scan to verify results



## 2024\_28\_III\_OOPS Using Java Lab

### REC\_2028\_OOPS using Java\_Week 11

Attempt : 1  
Total Mark : 20  
Marks Obtained : 20

#### Section 1 : Project

##### 1. Problem Statement

Create a JDBC-based Hospital Management System that handles runtime input to manage patient records. The system should allow users to:

Add a new patient (patient ID, name, age, status).

Update a patient's status.

View a specific patient's record by patient ID.

Display all patient records in the database.

Exit the application.

The system should connect to a MySQL database using the following default credentials:

DB URL: jdbc:mysql://localhost/ri\_db

USER: test

PWD: test123

The patients table has already been created with the following structure:

Table Name: patients

### ***Input Format***

The first line of input consists of an integer choice, representing the operation to be performed:

(1 for Add Patient, 2 for Update Patient Status, 3 for View Patient Record, 4 for Display All Patients, 5 for Exit)

For choice 1 (Add Patient):

- The second line consists of an integer patient\_id.
- The third line consists of a string name.
- The fourth line consists of an integer age.
- The fifth line consists of a string status.

For choice 2 (Update Patient Status):

- The second line consists of an integer patient\_id.
- The third line consists of a string new\_status.

For choice 3 (View Patient Record):

- The second line consists of an integer patient\_id.

For choice 4 (Display All Patients):

- No additional inputs are required.

For choice 5 (Exit):

- No additional inputs are required.

### **Output Format**

For choice 1 (Add Patient):

- Print "Patient added successfully" if the patient was added.
- Print "Failed to add patient." if the insertion failed.

For choice 2 (Update Patient Status):

- Print "Patient status updated successfully" if the update was successful.
- Print "Patient not found." if the specified patient ID does not exist.

For choice 3 (View Patient Record):

- Display the patient details in the format:
- ID: [patient\_id] | Name: [name] | Age: [age] | Status: [status]
- Print "Patient not found." if the specified patient ID does not exist.

For choice 4 (Display All Patients):

- Display each patient on a new line in the format:
- ID | Name | Age | Status
- If no records are available, print nothing (or handle it with an appropriate message if desired).

For choice 5 (Exit):

- Print "Exiting Hospital Management System."

For invalid input:

- Print "Invalid choice. Please try again."

### **Sample Test Case**

Input: 1

101

John Doe

45

Admitted

4

5

Output: Patient added successfully

ID | Name | Age | Status

101 | John Doe | 45 | Admitted

Exiting Hospital Management System.

**Answer**

```
import java.sql.*;
```

```
import java.util.Scanner;
```

```
class HospitalManagementSystem {
```

```
    public static void main(String[] args) {
```

```
        try (Connection conn = DriverManager.getConnection("jdbc:mysql://localhost/ri_db", "test", "test123"));
```

```
            Scanner scanner = new Scanner(System.in)) {
```

```
                boolean running = true;
```

```
                while (running) {
```

```
                    int choice = scanner.nextInt();
```

```
                    switch (choice) {
```

```
                        case 1:
```

```
                            addPatient(conn, scanner);
```

```
                            break;
```

```
                        case 2:
```

```
                            updatePatientStatus(conn, scanner);
```

```
                            break;
```

```
                        case 3:
```

```
                            viewPatientRecord(conn, scanner);
```

```
                            break;
```

```
                        case 4:
```

```
                            displayAllPatients(conn);
```

```
                            break;
```

```
                        case 5:
```

```
                            System.out.println("Exiting Hospital Management System.");
```

```
                            running = false;
```

```
                            break;
```

```
                        default:
```

```
                            System.out.println("Invalid choice. Please try again.");
```

```
                    }
```

```

    }
} catch (SQLException e) {
    e.printStackTrace();
}
}

```

```

public static void addPatient(Connection conn, Scanner scanner) {
    try {
        int id = scanner.nextInt();
        scanner.nextLine();
        String name = scanner.nextLine();
        int age = scanner.nextInt();
        scanner.nextLine();
        String status = scanner.nextLine();
        PreparedStatement ps = conn.prepareStatement("INSERT INTO patients VALUES
        (?, ?, ?, ?)");
        ps.setInt(1, id);
        ps.setString(2, name);
        ps.setInt(3, age);
        ps.setString(4, status);
        int rows = ps.executeUpdate();
        if (rows > 0) System.out.print("Patient added successfully");
        else System.out.print("Failed to add patient.");
    } catch (Exception e) {
        System.out.print("Failed to add patient.");
    }
}

```

```

public static void updatePatientStatus(Connection conn, Scanner scanner) {
    try {
        int id = scanner.nextInt();
        scanner.nextLine();
        String newStatus = scanner.nextLine();
        PreparedStatement ps = conn.prepareStatement("UPDATE patients SET status=?
        WHERE patient_id=?");
        ps.setString(1, newStatus);
        ps.setInt(2, id);
        int rows = ps.executeUpdate();
        if (rows > 0) System.out.print("Patient status updated successfully");
        else System.out.print("Patient not found.");
    } catch (Exception e) {
        System.out.print("Patient not found.");
    }
}

```

```
}
```

```
public static void viewPatientRecord(Connection conn, Scanner scanner) {  
    try {  
        int id = scanner.nextInt();  
        PreparedStatement ps = conn.prepareStatement("SELECT * FROM patients  
        WHERE patient_id=?");  
        ps.setInt(1, id);  
        ResultSet rs = ps.executeQuery();  
        if (rs.next()) {  
            System.out.print("ID: " + rs.getInt(1) + " | Name: " + rs.getString(2) + " | Age: " +  
            rs.getInt(3) + " | Status: " + rs.getString(4));  
        } else {  
            System.out.print("Patient not found.");  
        }  
    } catch (Exception e) {  
        System.out.print("Patient not found.");  
    }  
}
```

```
public static void displayAllPatients(Connection conn) {  
    try {  
        Statement st = conn.createStatement();  
        ResultSet rs = st.executeQuery("SELECT * FROM patients");  
        System.out.print("ID | Name | Age | Status");  
        while (rs.next()) {  
            System.out.print("\n" + rs.getInt(1) + " | " + rs.getString(2) + " | " + rs.getInt(3) + " |  
            " + rs.getString(4));  
        }  
    } catch (Exception e) {}  
}
```

**Status :** Correct

**Marks : 10/10**

## 2. Problem Statement

In Café Central, the menu is cataloged and stored in a database.

To efficiently manage the restaurant's menu using Java and JDBC, you

must build a Restaurant Management System that supports:

Adding new menu items

Updating menu item prices

Viewing details of a menu item

Displaying all menu items in sorted order

You are given two files:

File 1: MenuItem.java (POJO Class)

This class represents the MenuItem entity.

A MenuItem contains the following details:

Field	Description
itemId	Unique Menu Item ID (Integer)
name	Item Name (String)
category	Item Category (String)
price	Item Price (Double)

Students must write code in the marked area:

```
class MenuItem {  
    private int itemId;  
    private String name;  
    private String category;  
    private double price;  
  
    public MenuItem() {}  
  
    public MenuItem(int itemId, String name, String category, double price) {  
        // write your code here  
    }  
}
```

```
// Include getters and setters
```

Expected in this part:

Assign parameter values to instance variables inside the constructor.

Add getters and setters for all attributes.

File 2: MenuItemDAO.java (Data Access Layer)

This class handles all database operations using JDBC.

Students must complete the missing JDBC logic in the following methods:

```
class MenuItemDAO {  
  
    public void addItem(Connection conn, MenuItem menuItem)  
    throws SQLException {  
        // write your code here  
    }  
  
    public void updateItemPrice(Connection conn, int itemId, double  
    newPrice) throws SQLException {  
        // write your code here  
    }  
  
    public void deleteMenuItem(Connection conn, int itemId) throws  
    SQLException {  
        // write your code here  
    }  
  
    public MenuItem viewItemDetails(Connection conn, int itemId) throws  
    SQLException {  
        // write your code here  
    }  
}
```

```

    }

    public List<MenuItem> displayAllMenuItems(Connection conn) throws
SQLException {
        // write your code here
    }

    private MenuItem mapToMenuItem(ResultSet rs) throws SQLException {
        return new MenuItem(
            // write your code here
        );
    }
}

```

Expected in this part:

Write SQL queries for INSERT, UPDATE, DELETE, SELECT.

Execute queries using PreparedStatement or Statement.

Map ResultSet rows to MenuItem objects using mapToMenuItem().

Return a List<MenuItem> where required.

The system should connect to a MySQL database using the following default credentials:

DB URL: jdbc:mysql://localhost/ri\_db

USER: test

PWD: test123

The menu table has already been created with the following structure:

Table Name: menu

***Input Format***

The first line of input consists of an integer choice, representing the operation to be performed (1 for Add Item, 2 for Restock item, 3 for reduce item, 4 for Display, 5 for Exit).

For choice 1 (Add Menu Item):

- The second line consists of an integer item\_id.
- The third line consists of a string name.
- The fourth line consists of a string category.
- The fifth line consists of a double price.

For choice 2 (Update Item Price):

- The second line consists of an integer item\_id.
- The third line consists of a double new\_price.

For choice 3 (View Item Details):

- The second line consists of an integer item\_id.

For choice 4 (Display All Menu Items):

- No additional inputs are required.

For choice 5 (Exit):

- No additional inputs are required.

### **Output Format**

For choice 1 (Add Menu Item):

- Print "Menu item added successfully" if the item was added.
- Print "Failed to add item." if the insertion failed.

For choice 2 (Update Item Price):

- Print "Item price updated successfully" if the price update was successful.
- Print "Item not found." if the specified item ID does not exist.

For choice 3 (View Item Details):

- Display the item details in the format:

- ID: [item\_id] | Name: [name] | Category: [category] | Price: [price]
- Print "Item not found." if the specified item ID does not exist.

For choice 4 (Display All Menu Items):

- Display each item on a new line in the format:
- ID | Name | Category | Price
- If no items are available, print nothing (or handle with an appropriate message if desired).

For choice 5 (Exit):

- Print "Exiting Restaurant Management System."

For invalid input:

- Print "Invalid choice. Please try again."

### **Sample Test Case**

Input: 1

11

Margherita Pizza

Main Course

12.99

4

5

Output: Menu item added successfully

ID | Name | Category | Price

11 | Margherita Pizza | Main Course | 12.99

Exiting Restaurant Management System.

### **Answer**

```
import java.sql.*;
```

```
import java.util.Scanner;
```

```
class RestaurantManagementSystem {
```

```
    public static void main(String[] args) {
```

```
        try (Connection conn = DriverManager.getConnection("jdbc:mysql://localhost/ri_db", "test", "test123");
```

```
            Scanner scanner = new Scanner(System.in)) {
```

```
                boolean running = true;
```

```

while (running) {
    int choice = scanner.nextInt();

    switch (choice) {
        case 1:
            addMenuItem(conn, scanner);
            break;
        case 2:
            updateItemPrice(conn, scanner);
            break;
        case 3:
            viewItemDetails(conn, scanner);
            break;
        case 4:
            displayAllMenuItems(conn);
            break;
        case 5:
            System.out.println("Exiting Restaurant Management System.");
            running = false;
            break;
        default:
            System.out.println("Invalid choice. Please try again.");
    }
}

```

```

} catch (SQLException e) {
    e.printStackTrace();
}
}

```

```

public static void addMenuItem(Connection conn, Scanner scanner) {
    try {
        int id = scanner.nextInt();
        scanner.nextLine();
        String name = scanner.nextLine();
        String category = scanner.nextLine();
        double price = scanner.nextDouble();
        MenuItem item = new MenuItem(id, name, category, price);
        PreparedStatement ps = conn.prepareStatement("INSERT INTO menu
VALUES(?,?,?,?)");
        ps.setInt(1, item.getItemId());
        ps.setString(2, item.getName());
        ps.setString(3, item.getCategory());
    }
}

```

```
ps.setDouble(4, item.getPrice());
int rows = ps.executeUpdate();
if (rows > 0) System.out.print("Menu item added successfully");
else System.out.print("Failed to add item.");
} catch (Exception e) {
System.out.print("Failed to add item.");
}
}
```

```
public static void updateItemPrice(Connection conn, Scanner scanner) {
try {
int id = scanner.nextInt();
double newPrice = scanner.nextDouble();
PreparedStatement ps = conn.prepareStatement("UPDATE menu SET price=?
WHERE item_id=?");
ps.setDouble(1, newPrice);
ps.setInt(2, id);
int rows = ps.executeUpdate();
if (rows > 0) System.out.print("Item price updated successfully");
else System.out.print("Item not found.");
} catch (Exception e) {
System.out.print("Item not found.");
}
}
```

```
public static void viewItemDetails(Connection conn, Scanner scanner) {
try {
int id = scanner.nextInt();
PreparedStatement ps = conn.prepareStatement("SELECT * FROM menu WHERE
item_id=?");
ps.setInt(1, id);
ResultSet rs = ps.executeQuery();
if (rs.next()) {
System.out.print("ID: " + rs.getInt(1) + " | Name: " + rs.getString(2) + " | Category: "
+ rs.getString(3) + " | Price: " + rs.getDouble(4));
} else {
System.out.print("Item not found.");
}
} catch (Exception e) {
System.out.print("Item not found.");
}
}
```

```

public static void displayAllMenuItems(Connection conn) {
    try {
        Statement st = conn.createStatement();
        ResultSet rs = st.executeQuery("SELECT * FROM menu ORDER BY item_id ASC");
        System.out.print("ID | Name | Category | Price");
        boolean first = true;
        while (rs.next()) {
            System.out.print("\n" + rs.getInt(1) + " | " + rs.getString(2) + " | " + rs.getString(3)
                + " | " + String.format("%.2f", rs.getDouble(4)));
        }
    } catch (Exception e) {}
}

```

```

class MenuItem {
    private int itemId;
    private String name;
    private String category;
    private double price;

```

```

    public MenuItem(int itemId, String name, String category, double price) {
        this.itemId = itemId;
        this.name = name;
        this.category = category;
        this.price = price;
    }

```

```

    public int getItemId() { return itemId; }
    public String getName() { return name; }
    public String getCategory() { return category; }
    public double getPrice() { return price; }

```

```

    public void setItemId(int itemId) { this.itemId = itemId; }
    public void setName(String name) { this.name = name; }
    public void setCategory(String category) { this.category = category; }
    public void setPrice(double price) { this.price = price; }
}

```

```

//

```

**Status :** Correct

**Marks :** 10/10