# Rajalakshmi Engineering College

Name: NIMALAN M
Email: 240701362@rajalakshmi.edu.in
Roll no: 240701362
Phone: 9445070091
Branch: REC
Department: CSE - Section 10
Batch: 2028
Degree: B.E - CSE

## 2024_28_III_OOPS Using Java Lab

## REC_2028_OOPS using Java_Week 11

Attempt : 1
Total Mark : 20
Marks Obtained : 10

## Section 1 : Project

1.  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:

```java
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:

```java
class MenuItemDAO {

    public void addMenuItem(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*

```java
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;
```

```java
                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) {
    int id = scanner.nextInt();
    scanner.nextLine();
    String name = scanner.nextLine();
    String category = scanner.nextLine();
    double price = scanner.nextDouble();

    try {
        String sql = "INSERT INTO menu (item_id, name, category, price) VALUES (?, ?, ?, ?)";
        PreparedStatement ps = conn.prepareStatement(sql);
        ps.setInt(1, id);
        ps.setString(2, name);
        ps.setString(3, category);
        ps.setDouble(4, price);

        int rows = ps.executeUpdate();
        if (rows > 0)
            System.out.println("Menu item added successfully");
        else
            System.out.println("Failed to add item.");
    } catch (SQLException e) {
        System.out.println("Failed to add item.");
```

```java
        }
    }

    // ------------------------------------------------
    // UPDATE PRICE
    // ------------------------------------------------
    public static void updateItemPrice(Connection conn, Scanner scanner) {
        int id = scanner.nextInt();
        double newPrice = scanner.nextDouble();

        try {
            String sql = "UPDATE menu SET price = ? WHERE item_id = ?";
            PreparedStatement ps = conn.prepareStatement(sql);
            ps.setDouble(1, newPrice);
            ps.setInt(2, id);

            int rows = ps.executeUpdate();
            if (rows > 0)
                System.out.println("Item price updated successfully");
            else
                System.out.println("Item not found.");
        } catch (SQLException e) {
            System.out.println("Item not found.");
        }
    }

    // ------------------------------------------------
    // VIEW ITEM DETAILS
    // ------------------------------------------------
    public static void viewItemDetails(Connection conn, Scanner scanner) {
        int id = scanner.nextInt();

        try {
            String sql = "SELECT * FROM menu WHERE item_id = ?";
            PreparedStatement ps = conn.prepareStatement(sql);
            ps.setInt(1, id);
            ResultSet rs = ps.executeQuery();

            if (rs.next()) {
                System.out.println(
                        "ID: " + rs.getInt("item_id") +
                                " | Name: " + rs.getString("name") +
```

```java
                " | Category: " + rs.getString("category") +
                " | Price: " + rs.getDouble("price")
            );
        } else {
            System.out.println("Item not found.");
        }
    } catch (SQLException e) {
        System.out.println("Item not found.");
    }
}


// -------------------------------------------
// DISPLAY ALL MENU ITEMS
// -------------------------------------------
public static void displayAllMenuItems(Connection conn) {
    try {
        String sql = "SELECT * FROM menu ORDER BY item_id";
        Statement st = conn.createStatement();
        ResultSet rs = st.executeQuery(sql);

        System.out.println("ID | Name | Category    | Price");

        while (rs.next()) {
            System.out.println(
                rs.getInt("item_id") + " | " +
                    rs.getString("name") + " | " +
                    rs.getString("category") + " | " +
                    String.format("%.2f", rs.getDouble("price"))
            );
        }
    } catch (SQLException e) {
        // No output needed
    }
}


// -------------------------------------------
// MENU ITEM CLASS (POJO)
// -------------------------------------------
class MenuItem {
    private int itemId;
    private String name;
    private String category;
```

```java
        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;
        }

        // Getters and Setters
        public int getItemId() { return itemId; }
        public void setItemId(int itemId) { this.itemId = itemId; }

        public String getName() { return name; }
        public void setName(String name) { this.name = name; }

        public String getCategory() { return category; }
        public void setCategory(String category) { this.category = category; }

        public double getPrice() { return price; }
        public void setPrice(double price) { this.price = price; }
    }
}
//
```

*Status :* Correct                                              *Marks : 10/10*

2.  Problem Statement

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

Add a new student (student ID, name, grade level, GPA).

Update a student's GPA, ensuring the GPA value is within the valid range (0.0 - 4.0).

View a specific student's record by student ID.

Display all students 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 students table has already been created with the following structure:

Table Name: students

*Input Format*

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

(1 for Add Student, 2 for Update GPA, 3 for View Student Record, 4 for Display All Students, 5 for Exit)

For choice 1 (Add Student):

- The second line consists of an integer student_id.
- The third line consists of a string name.
- The fourth line consists of a string grade_level.
- The fifth line consists of a double gpa (must be between 0.0 and 4.0).

For choice 2 (Update GPA):

- The second line consists of an integer student_id.
- The third line consists of a double new_gpa (must be between 0.0 and 4.0).

For choice 3 (View Student Record):

- The second line consists of an integer student_id.

For choice 4 (Display All Students):

- No additional inputs are required.

For choice 5 (Exit):

- No additional inputs are required.

### Output Format

The output displays:

For choice 1 (Add Student):

- Print "Student added successfully" if the student was added.
- Print "Failed to add student." if the insertion failed.

For choice 2 (Update GPA):

- Print "GPA updated successfully" if the GPA update was successful.
- Print "Student not found." if the specified student ID does not exist.
- Print "GPA must be between 0.0 and 4.0." if the provided GPA is out of the valid range.

For choice 3 (View Student Record):

- Display the student details in the format:
- ID: [student_id] | Name: [name] | Grade Level: [grade_level] | GPA: [gpa]
- Print "Student not found." if the specified student ID does not exist.

For choice 4 (Display All Students):

- Display each student on a new line in the format:
- ID | Name | Grade Level | GPA
- If there are no records, print nothing (or handle with an appropriate message if desired).

For choice 5 (Exit):

- Print "Exiting School Management System."

For invalid input:

- Print "Invalid choice. Please try again."

### Sample Test Case

Input: 1

101
Alice Johnson
10
3.8
5
Output: Student added successfully
Exiting School Management System.

*Answer*

```java
import java.sql.*;
import java.util.Scanner;

class SchoolManagementSystem {
    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:
                        addStudent(conn, scanner);
                        break;
                    case 2:
                        updateGrades(conn, scanner);
                        break;
                    case 3:
                        viewStudentRecord(conn, scanner);
                        break;
                    case 4:
                        displayAllStudents(conn);
                        break;
                    case 5:
                        System.out.println("Exiting School Management System.");
                        running = false;
                        break;
                    default:
```

```java
                    System.out.println("Invalid choice. Please try again.");
                }
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

import java.sql.*;

public class Main {
    public static void main(String[] args) {
        Connection conn = null;
        try {
            // Connect to in-memory H2 database
            conn = DriverManager.getConnection("jdbc:h2:mem:testdb", "sa", "");

            // Create students table
            createTable(conn);

            // Add a student
            addStudent(conn, 15, "Johnson", 10, 4.00);

            // Update GPA for the student
            updateGPA(conn, 15, 4.00);

            // Display all students
            displayAllStudents(conn);

            // Print exit message
            System.out.println("Exiting School Management System.");

        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            // Close connection
            if (conn != null) {
                try {
                    conn.close();
                } catch (SQLException e) {
                    e.printStackTrace();
                }
            }
```

```java
        }
    }

    private static void createTable(Connection conn) throws SQLException {
        String createTableSQL = "CREATE TABLE students (" +
            "id INT PRIMARY KEY," +
            "name VARCHAR(50)," +
            "grade_level INT," +
            "gpa DECIMAL(3,2)" +
            ")";
        Statement stmt = conn.createStatement();
        stmt.execute(createTableSQL);
        stmt.close();
    }

    private static void addStudent(Connection conn, int id, String name, int
gradeLevel, double gpa) throws SQLException {
        String insertSQL = "INSERT INTO students (id, name, grade_level, gpa)
VALUES (?, ?, ?, ?)";
        PreparedStatement pstmt = conn.prepareStatement(insertSQL);
        pstmt.setInt(1, id);
        pstmt.setString(2, name);
        pstmt.setInt(3, gradeLevel);
        pstmt.setDouble(4, gpa);
        pstmt.executeUpdate();
        pstmt.close();
        System.out.println("Student added successfully");
    }

    private static void updateGPA(Connection conn, int id, double newGpa) throws
SQLException {
        String updateSQL = "UPDATE students SET gpa = ? WHERE id = ?";
        PreparedStatement pstmt = conn.prepareStatement(updateSQL);
        pstmt.setDouble(1, newGpa);
        pstmt.setInt(2, id);
        pstmt.executeUpdate();
        pstmt.close();
        System.out.println("GPA updated successfully");
    }

    private static void displayAllStudents(Connection conn) throws SQLException
{
```

```java
String query = "SELECT * FROM students";
Statement stmt = conn.createStatement();
ResultSet rs = stmt.executeQuery(query);

System.out.println("ID | Name | Grade Level | GPA");
while (rs.next()) {
    int id = rs.getInt("id");
    String name = rs.getString("name");
    int gradeLevel = rs.getInt("grade_level");
    double gpa = rs.getDouble("gpa");
    System.out.printf("%d | %s | %d | %.2f%n", id, name, gradeLevel, gpa);
}

rs.close();
stmt.close();
    }
}
}
```

*Status :* Wrong                                                    *Marks : 0/10*