

# Rajalakshmi Engineering College

Name: Kishore Kumar T  
Email: 240701269@rajalakshmi.edu.in  
Roll no: 240701269  
Phone: 6381094420  
Branch: REC  
Department: CSE - Section 7  
Batch: 2028  
Degree: B.E - CSE

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

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();
}
}

// You are using Java

```

```

public static void addMenuItem(Connection conn, Scanner scanner) {
    try{
        int item_id=scanner.nextInt();
        scanner.nextLine();
        String name=scanner.nextLine();
        String category=scanner.nextLine();
        double price=scanner.nextDouble();
        String sql="Insert into menu(item_id,name,category,price) values (?,?,,?)";
        try(PreparedStatement pstmt=conn.prepareStatement(sql)){
            pstmt.setInt(1,item_id);
            pstmt.setString(2,name);
            pstmt.setString(3,category);
            pstmt.setDouble(4,price);
            int r=pstmt.executeUpdate();
            if(r>0)
                System.out.println("Menu item added successfully");
            else
                System.out.println("Item not found");
        }
    }
    catch (SQLException e) {

```

```
        System.out.println(e.getMessage());
    }
}
```

```
public static void updateItemPrice(Connection conn, Scanner scanner) {
    //Write your code here
    try{
        int item_id=scanner.nextInt();
        double newPrice=scanner.nextDouble();
        String sql="Update menu set price=? where item_id=?";
        try(PreparedStatement pstmt=conn.prepareStatement(sql)){
            pstmt.setDouble(1,newPrice);
            pstmt.setInt(2,item_id);
            int r=pstmt.executeUpdate();
            if(r>0)
                System.out.println("Item price updated successfully");
            else
                System.out.println("Item not found");
        }
    }
    catch (SQLException e) {
        System.out.println(e.getMessage());
    }
}
```

```
public static void viewItemDetails(Connection conn, Scanner scanner) {
    //Write your code here
    int item_id=scanner.nextInt();
    String sql="Select * from menu where item_id =?";
    try(PreparedStatement pstmt=conn.prepareStatement(sql)){
        pstmt.setInt(1,item_id);
        try(ResultSet rs=pstmt.executeQuery()){
            if(rs.next()){
                System.out.println("ID: "+rs.getInt("item_id")+" | Name:
"+rs.getString("name")+" | Category: "+rs.getString("category")+" | Price:
"+rs.getDouble("price"));
            }
            else
                System.out.println("Item not found");
        }
    }
    catch (SQLException e) {
```



```

        System.out.println(e.getMessage());
    }
}

public static void displayAllMenuItems(Connection conn) throws
SQLException{
    //Write your code here
    try{
        String sql="Select * from menu order by item_id";
        System.out.println("ID | Name | Category | Price");
        try(Statement stmt=conn.createStatement();
            ResultSet rs=stmt.executeQuery(sql)){
            while(rs.next()){
                System.out.printf("%d | %s | %s | %.2f\n",rs.getInt("item_id"),rs.getStrin
g("name"),rs.getString("category"),rs.getDouble("price"));
            }
        }
    }catch (SQLException e) {
        System.out.println(e.getMessage());
    }
}

```

```

static class MenuItem {
    private int item_id;
    private String name;
    private String category;
    private double price;

    // Constructor
    public MenuItem(){

    }

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

    //Include getters and setters
    public int getItem_id(){
        return item_id;
    }
}

```

```

    public void setItem_id(int item_id){
        this.item_id=item_id;
    }

    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**

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

```

// You are using Java

```

public static void addStudent(Connection conn, Scanner scanner) {
    // write your code here
    int student_id=scanner.nextInt();
    scanner.nextLine();
    String name=scanner.nextLine();
    String grade_level=scanner.nextLine();
    double gpa=scanner.nextDouble();
    String sql="Insert into students(student_id,name,grade_level,gpa) values
(?,?,?,?)";
    try(PreparedStatement pstmt=conn.prepareStatement(sql)){
        pstmt.setInt(1,student_id);
        pstmt.setString(2,name);
        pstmt.setString(3,grade_level);
        pstmt.setDouble(4,gpa);
        int r=pstmt.executeUpdate();
        if(r>0)
            System.out.println("Student added successfully");
        else
            System.out.println("Student Not found");
    }
    catch(SQLException e){
        System.out.println(e.getMessage());
    }
}
}

```

```

public static void updateGrades(Connection conn, Scanner scanner) {
    // write your code here
    int student_id=scanner.nextInt();
    double gpa=scanner.nextDouble();
    if(gpa < 0.0 || gpa > 4.0){
        System.out.println("GPA must be betwwen 0.0 and 4.0.");
        return;
    }
    String sql="Update students set gpa=? where student_id=?";
    try(PreparedStatement pstmt=conn.prepareStatement(sql)){
        pstmt.setDouble(1,gpa);
        pstmt.setInt(2,student_id);
        int r=pstmt.executeUpdate();
        if(r>0)
            System.out.println("GPA updated successfully");
        else
            System.out.println("Student Not found");
    }
    catch(SQLException e){
        System.out.println(e.getMessage());
    }
}

```

```

public static void viewStudentRecord(Connection conn, Scanner scanner) {
    // write your code here
    int student_id=scanner.nextInt();
    String sql="select * from students where student_id=?";
    try(PreparedStatement pstmt=conn.prepareStatement(sql)){
        pstmt.setInt(1,student_id);
        try(ResultSet rs=pstmt.executeQuery()){
            if(rs.next()){
                System.out.println("ID: "+rs.getInt("student_id")+" | Name:
"+rs.getString("name")+" | Grade Level: "+rs.getString("grade_level")+" | GPA:
"+String.format("%.2f",rs.getDouble("gpa")));
            }
            else
                System.out.println("Student Not found");
        }
    }
    catch(SQLException e){
        System.out.println(e.getMessage());
    }
}

```

```

    }
    public static void displayAllStudents(Connection conn) throws SQLException{
        // write your code here
        try{
            String sql="Select * from students order by student_id";
            System.out.println("ID | Name | Grade Level | GPA");
            try(Statement stmt=conn.createStatement();
                ResultSet rs=stmt.executeQuery(sql)){
                while(rs.next()){
                    System.out.printf("%d | %s | %s | %.2f\n",rs.getInt("student_id"),rs.get
String("name"),rs.getString("grade_level"),rs.getDouble("gpa"));
                }
            }
        } catch(SQLException e){
            System.out.println(e.getMessage());
        }
    }
    static class StudentDetails{
        private int student_id;
        private String name;
        private String grade_level;
        private double gpa;
        public StudentDetails(){
        }
        public StudentDetails(int student_id,String name,String grade_level,double
gpa){
            this.student_id=student_id;
            this.name=name;
            this.grade_level=grade_level;
            this.gpa=gpa;
        }
        public int getStudent_id(){
            return student_id;
        }
        public void setStudent_id(int student_id){
            this.student_id=student_id;
        }
        public String getName(){
            return name;
        }
    }

```



```
public void setName(String name){
    this.name=name;
}
public String getGrade_Level(){
    return grade_level;
}
public void setGrade_Level(String grade_level){
    this.grade_level=grade_level;
}
public double getGpa(){
    return gpa;
}
public void setGpa(double gpa){
    this.gpa=gpa;
}
}
```

**Status :** Correct

**Marks : 10/10**