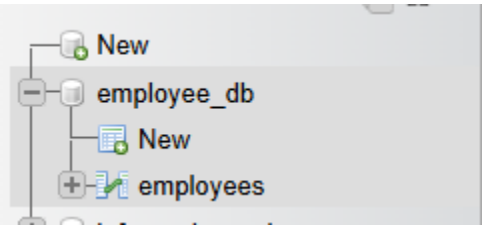
Enterprise Architecture – 4232

Lab sheet 03

Task 01

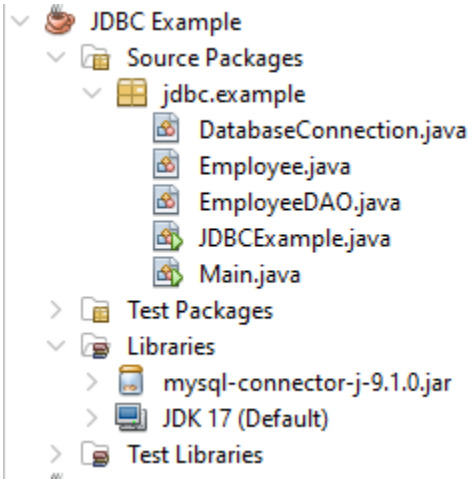
```
CREATE DATABASE employee_db;

USE employee_db;

CREATE TABLE employees (
    id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100),
    position VARCHAR(100),
    salary DECIMAL(10, 2)
);

-- Insert some sample data

INSERT INTO employees (name, position, salary) VALUES ('John Doe', 'Software Engineer', 75000);

INSERT INTO employees (name, position, salary) VALUES ('Jane Smith', 'HR Manager', 65000);

INSERT INTO employees (name, position, salary) VALUES ('Steve Brown', 'Team Lead', 85000);
```



Task 02

2. Set Up NetBeans Project

1. Open NetBeans IDE 8.2.
2. Create a new Java application:
    - Go to File > New Project.
    - Select Java as the project type, and choose Java Application.
    - Name your project JDBCExample.
3. Add MySQL JDBC Driver to your project:
    - Right-click on the project in the Projects pane.
    - Select Properties.
    - In the Libraries tab, click Add JAR/Folder.
    - Navigate to the location of your mysql-connector-java-x.x.xx.jar file and add it.



Task 03

3. Establish JDBC Connection

- Create a **DatabaseConnection.java** class to establish a connection to your database.

```
package jdbc.example;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
public class DatabaseConnection {
private static final String URL ="jdbc:mysql://localhost:3306/employee_db";
private static final String USER = "root";
private static final String PASSWORD = "";
public static Connection getConnection() throws SQLException {
try {
// Load the JDBC driver
Class.forName("com.mysql.cj.jdbc.Driver");
// Return the database connection
return DriverManager.getConnection(URL, USER, PASSWORD);
}
catch (ClassNotFoundException | SQLException e) {
System.out.println("Connection failed" + e.getMessage());
throw new SQLException("Failed to establish connection.");
}}
}
```

Task 04

4. Perform CRUD Operations

```
package jdbc.example;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;
public class EmployeeDAO {
    // Create an employee
    public static void addEmployee(String name, String position, double salary) {
        String sql = "INSERT INTO employees (name, position, salary) VALUES (?, ?, ?)";

        try (Connection conn = DatabaseConnection.getConnection();
             PreparedStatement stmt = conn.prepareStatement(sql)) {
            stmt.setString(1, name);
            stmt.setString(2, position);
            stmt.setDouble(3, salary);
            int rowsAffected = stmt.executeUpdate();
            System.out.println("Employee added successfully. Rows affected: " + rowsAffected);
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
 // Read all employees
    public static List<Employee> getAllEmployees() {
        List<Employee> employees = new ArrayList<>();
        String sql = "SELECT * FROM employees";
```

```
        try (Connection conn = DatabaseConnection.getConnection();
             Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery(sql)) {
            while (rs.next()) {
                Employee employee = new Employee(
                    rs.getInt("id"),
                    rs.getString("name"),
                    rs.getString("position"),
                    rs.getDouble("salary"));
                employees.add(employee);
            }
        } catch (SQLException e) {
            e.printStackTrace();  }
return employees;
    }
    // Update an employee's information
    public static void updateEmployee(int id, String name, String position, double salary) {
        String sql = "UPDATE employees SET name = ?, position = ?, salary = ? WHERE id = ?";
try (Connection conn = DatabaseConnection.getConnection();
             PreparedStatement stmt = conn.prepareStatement(sql)) {
            stmt.setString(1, name);
            stmt.setString(2, position);
            stmt.setDouble(3, salary);
            stmt.setInt(4, id);
            int rowsAffected = stmt.executeUpdate();
            System.out.println("Employee updated successfully. Rows affected: " + rowsAffected);
        } catch (SQLException e) {
```

```
            e.printStackTrace();
        }
    }
// Delete an employee
    public static void deleteEmployee(int id) {
        String sql = "DELETE FROM employees WHERE id = ?";

        try (Connection conn = DatabaseConnection.getConnection();
             PreparedStatement stmt = conn.prepareStatement(sql)) {
            stmt.setInt(1, id);
            int rowsAffected = stmt.executeUpdate();
            System.out.println("Employee deleted successfully. Rows affected: " + rowsAffected);
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }}
```

Task 05

5**. Create Employee.java Class**

```
package jdbc.example;

public class Employee {
    private int id;
    private String name;
    private String position;
    private double salary;
    public Employee(int id, String name, String position, double salary) {
        this.id = id;
        this.name = name;
        this.position = position;
        this.salary = salary;}
    // Getters and setters
    public int getId() { return id; }
    public void setId(int id) { this.id = id; }
    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
    public String getPosition() { return position; }
    public void setPosition(String position) { this.position = position; }
    public double getSalary() { return salary; }
    public void setSalary(double salary) { this.salary = salary; }
    @Override
    public String toString() {
        return "Employee{id=" + id + ", name='" + name + "', position='" + position + "', salary=" +
salary + '}';
}}
```

Task 06

6.Test the Application

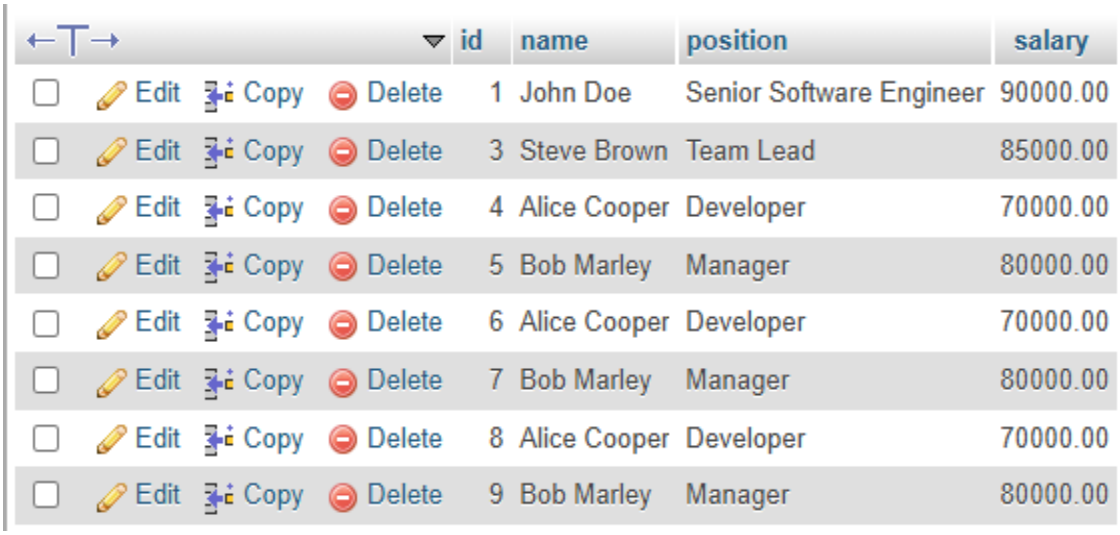
```
package jdbc.example;

import java.util.List;

public class Main {
    public static void main(String[] args) {
        // Add employees
        EmployeeDAO.addEmployee("Alice Cooper", "Developer", 70000);
        EmployeeDAO.addEmployee("Bob Marley", "Manager", 80000);

        // Update employee
        EmployeeDAO.updateEmployee(1, "John Doe", "Senior Software Engineer", 90000);

        // Get all employees
        List<Employee> employees = EmployeeDAO.getAllEmployees();
        employees.forEach(System.out::println);

        // Delete employee
        EmployeeDAO.deleteEmployee(2);
    }
}
```

Task 07

7. Run the Application

Run the program and observe how the database is updated with the CRUD operations.

- First, the employees will be added to the database.
- Then, one employee's details will be updated.
- All employees will be fetched and displayed in the console.
- Finally, one employee will be deleted.



	id	name	position	salary
Edit Copy Delete	1	John Doe	Senior Software Engineer	90000.00
Edit Copy Delete	3	Steve Brown	Team Lead	85000.00
Edit Copy Delete	4	Alice Cooper	Developer	70000.00
Edit Copy Delete	5	Bob Marley	Manager	80000.00
Edit Copy Delete	6	Alice Cooper	Developer	70000.00
Edit Copy Delete	7	Bob Marley	Manager	80000.00
Edit Copy Delete	8	Alice Cooper	Developer	70000.00
Edit Copy Delete	9	Bob Marley	Manager	80000.00