

1. Set Up MySQL Database

```
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)
);
```

The screenshot shows the phpMyAdmin interface. On the left, the database 'employee_db' is selected, and the 'employees' table is highlighted. The main panel displays the table structure and data. The table has four columns: 'id', 'name', 'position', and 'salary'. The data is as follows:

id	name	position	salary
1	John Doe	Senior Software Engineer	90000.00
3	steve Brown	Team Lead	8500.00
4	Alice Cooper	Developer	70000.00
5	Bob Marley	Manager	80000.00

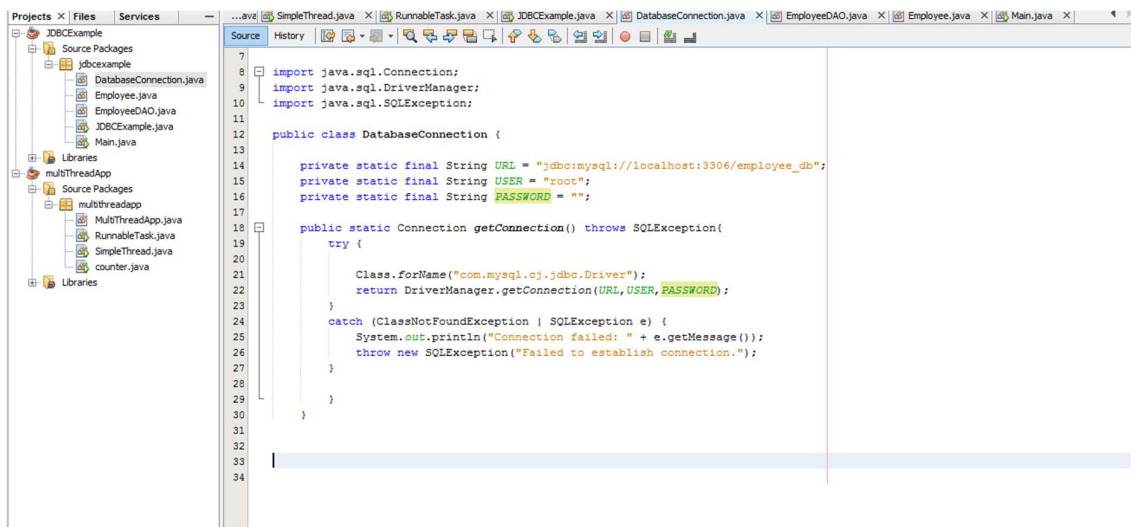
The interface also shows a SQL query bar with the query 'SELECT * FROM `employees`' and a status bar indicating 'Showing rows 0 - 3 (4 total. Query took 0.0003 sec)'.

2. Establish JDBC Connection

```
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"; // Database URL
    private static final String USER = "root"; // Your MySQL username
    private static final String PASSWORD = "password"; // Your MySQL
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.");
        }
    }
}
```



3. Perform CRUD Operations

```
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();
        }
    }

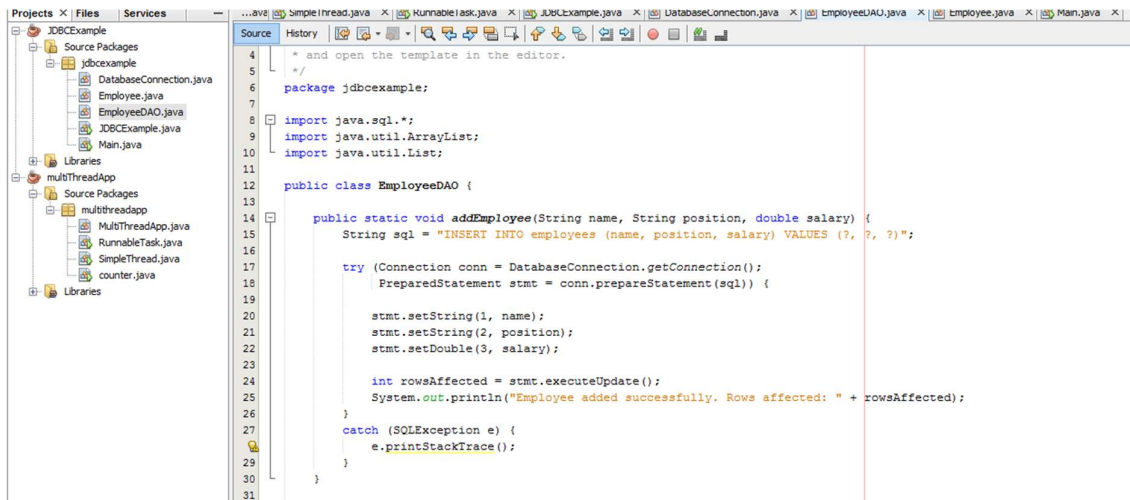
    // Delete an employee
    public static void deleteEmployee(int id) {
        String sql = "DELETE FROM employees WHERE id = ?";

        try (Connection conn = DatabaseConnection.getConnection();
            PreparedStatement stmt = conn.prepareStatement(sql)) {

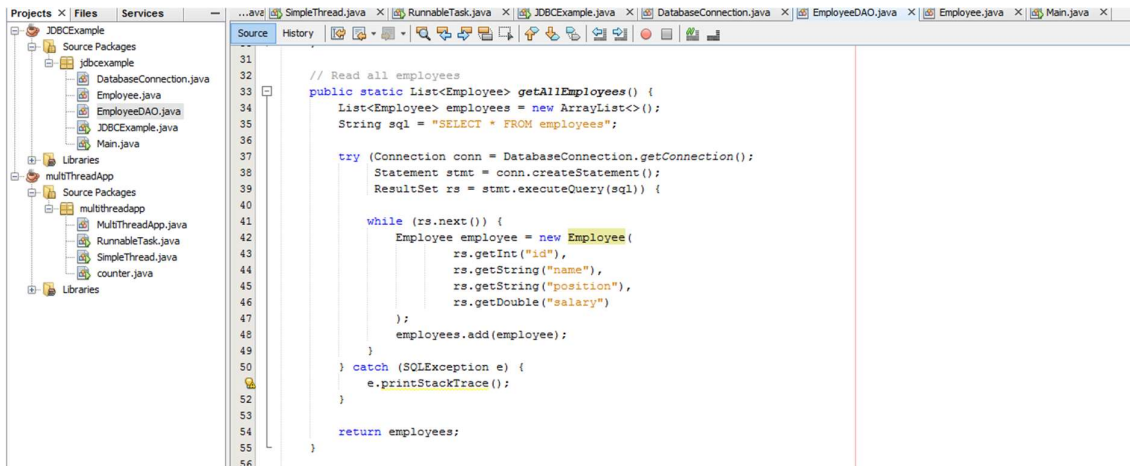
            stmt.setInt(1, id);
            int rowsAffected = stmt.executeUpdate();
            System.out.println("Employee deleted successfully. Rows
affected: " + rowsAffected);
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}

```

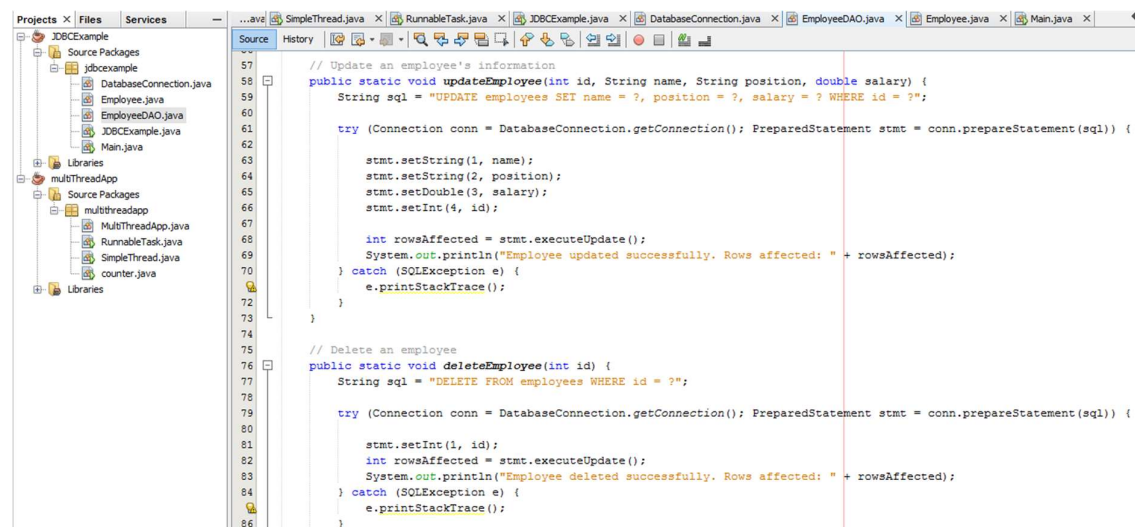
Code:



```
4  * and open the template in the editor.
5  */
6  package jdbcexample;
7
8  import java.sql.*;
9  import java.util.ArrayList;
10 import java.util.List;
11
12 public class EmployeeDAO {
13
14     public static void addEmployee(String name, String position, double salary) {
15         String sql = "INSERT INTO employees (name, position, salary) VALUES (?, ?, ?)";
16
17         try (Connection conn = DatabaseConnection.getConnection();
18             PreparedStatement stmt = conn.prepareStatement(sql)) {
19
20             stmt.setString(1, name);
21             stmt.setString(2, position);
22             stmt.setDouble(3, salary);
23
24             int rowsAffected = stmt.executeUpdate();
25             System.out.println("Employee added successfully. Rows affected: " + rowsAffected);
26
27         } catch (SQLException e) {
28             e.printStackTrace();
29         }
30     }
31 }
```



```
31
32 // Read all employees
33 public static List<Employee> getAllEmployees() {
34     List<Employee> employees = new ArrayList<>();
35     String sql = "SELECT * FROM employees";
36
37     try (Connection conn = DatabaseConnection.getConnection();
38         Statement stmt = conn.createStatement();
39         ResultSet rs = stmt.executeQuery(sql)) {
40
41         while (rs.next()) {
42             Employee employee = new Employee(
43                 rs.getInt("id"),
44                 rs.getString("name"),
45                 rs.getString("position"),
46                 rs.getDouble("salary")
47             );
48             employees.add(employee);
49         }
50     } catch (SQLException e) {
51         e.printStackTrace();
52     }
53
54     return employees;
55 }
56
```



```
57 // Update an employee's information
58 public static void updateEmployee(int id, String name, String position, double salary) {
59     String sql = "UPDATE employees SET name = ?, position = ?, salary = ? WHERE id = ?";
60
61     try (Connection conn = DatabaseConnection.getConnection(); PreparedStatement stmt = conn.prepareStatement(sql)) {
62
63         stmt.setString(1, name);
64         stmt.setString(2, position);
65         stmt.setDouble(3, salary);
66         stmt.setInt(4, id);
67
68         int rowsAffected = stmt.executeUpdate();
69         System.out.println("Employee updated successfully. Rows affected: " + rowsAffected);
70     } catch (SQLException e) {
71         e.printStackTrace();
72     }
73 }
74
75 // Delete an employee
76 public static void deleteEmployee(int id) {
77     String sql = "DELETE FROM employees WHERE id = ?";
78
79     try (Connection conn = DatabaseConnection.getConnection(); PreparedStatement stmt = conn.prepareStatement(sql)) {
80
81         stmt.setInt(1, id);
82         int rowsAffected = stmt.executeUpdate();
83         System.out.println("Employee deleted successfully. Rows affected: " + rowsAffected);
84     } catch (SQLException e) {
85         e.printStackTrace();
86     }
87 }
```

4. Create `Employee.java` Class

```
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; }

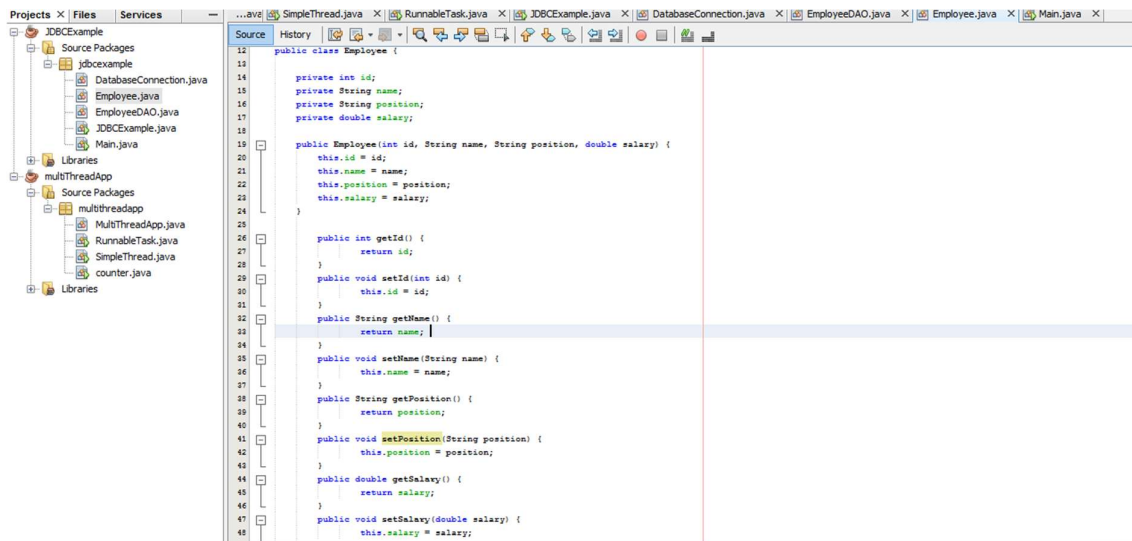
    public String getName() { return name; }
    public void setName(String name) { this.name = name; }

    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 + "'}";
    }
}
```

Code:



5. Test the Application

```
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);
    }
}
```

Code:

```
import java.util.List;

public class Main {

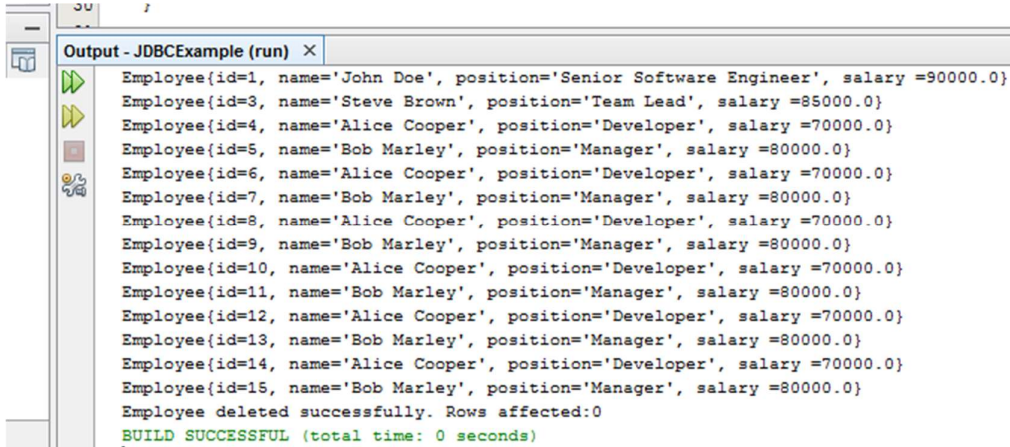
    public static void main(String[] args) {
        // Add employees
        EmployeeDAO.addEmployee("Alice Cooper", "Developer", 70000);
        EmployeeDAO.addEmployee("Bob Marley", "Manager", 80000);

        // 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);
    }
}
```

Output:



```
Output - JDBCEXample (run) X
Employee{id=1, name='John Doe', position='Senior Software Engineer', salary =90000.0}
Employee{id=3, name='Steve Brown', position='Team Lead', salary =85000.0}
Employee{id=4, name='Alice Cooper', position='Developer', salary =70000.0}
Employee{id=5, name='Bob Marley', position='Manager', salary =80000.0}
Employee{id=6, name='Alice Cooper', position='Developer', salary =70000.0}
Employee{id=7, name='Bob Marley', position='Manager', salary =80000.0}
Employee{id=8, name='Alice Cooper', position='Developer', salary =70000.0}
Employee{id=9, name='Bob Marley', position='Manager', salary =80000.0}
Employee{id=10, name='Alice Cooper', position='Developer', salary =70000.0}
Employee{id=11, name='Bob Marley', position='Manager', salary =80000.0}
Employee{id=12, name='Alice Cooper', position='Developer', salary =70000.0}
Employee{id=13, name='Bob Marley', position='Manager', salary =80000.0}
Employee{id=14, name='Alice Cooper', position='Developer', salary =70000.0}
Employee{id=15, name='Bob Marley', position='Manager', salary =80000.0}
Employee deleted successfully. Rows affected:0
BUILD SUCCESSFUL (total time: 0 seconds)
```