

1. Create a Database Connection.java class to establish a connection to your database

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */

/**
 *
 * @author NIRWAN
 */
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 = "316830059"; // 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:URL, user:USER, password:PASSWORD);
        } catch (ClassNotFoundException | SQLException e) {
            System.out.println("Connection failed: " + e.getMessage());
            throw new SQLException("Failed to establish connection.");
        }
    }
}
```

2. Create EmployeeDAO.java for CRUD Operations

```
/*  
package jdbcexample;  
  
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( string:sql)) {

        while (rs.next()) {
            Employee employee = new Employee(
                id:rs.getInt( string:"id"),
                name:rs.getString( string:"name"),
                position:rs.getString( string:"position"),
                salary:rs.getDouble( string:"salary")
            );
            employees.add( e: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( string:sql)) {

        stmt.setString( i:1, string:name);
        stmt.setString( i:2, string:position);
        stmt.setDouble( i:3, d:salary);
        stmt.setInt( i:4, i: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( string:sql)) {

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

}
```

3. 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;
    }

    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 + "'}";
    }
}
```

```
] /**
  * @param args the command line arguments
  */
] public static void main(String[] args) {
    // Add employees
    EmployeeDAO.addEmployee( name: "Alice Cooper", position: "Developer", salary: 70000);
    EmployeeDAO.addEmployee( name: "Bob Marley", position: "Manager", salary: 80000);

    // Update employee
    EmployeeDAO.updateEmployee( id: 1, name: "John Doe", position: "Senior Software Engineer", salary: 90000);

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

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

Output

Output - JDBCExample (run)



run:



Employee added successfully. Rows affected: 1

Employee added successfully. Rows affected: 1



Employee updated successfully. Rows affected: 1



Employee{id=1, name='John Doe', position='Senior Software Engineer', salary=90000.0}

Employee{id=2, name='Jane Smith', position='HR Manager', salary=65000.0}

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 deleted successfully. Rows affected: 1

BUILD SUCCESSFUL (total time: 2 seconds)

Server: MySQL:3306 » Database: employee_db » Table: employees

Browse

Structure

SQL

Search

Insert

Export

Import

Privileges

Operations

Triggers

Showing rows 0 - 3 (4 total, Query took 0.0025 seconds.)

SELECT * FROM `employees`

☐ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

☐ Show all

Number of rows: 25

Filter rows: Search this table

Sort by key: None

Extra options

↩

↪

▼

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

⬆

☐ Check all

With selected:

Edit

Copy

Delete

Export

☐ Show all

Number of rows: 25

Filter rows: Search this table

Sort by key: None

