


```

        break;
    case 2:
        updateEmployee(connection, scanner);
        break;
    case 3:
        deleteEmployee(connection, scanner);
        break;
    case 4:
        displayAllEmployees(connection);
        break;
    case 5:
        System.out.println("Exiting...");
        return;
    default:
        System.out.println("Invalid choice. Please try again.");
    }
}
} catch (SQLException e) {
    e.printStackTrace();
}
}

private static void createTableIfNotExists(Connection connection) throws SQLException {
    String createTableSQL = "CREATE TABLE IF NOT EXISTS employees (" +
        "id INT PRIMARY KEY AUTO_INCREMENT," +
        "name VARCHAR(255)," +
        "department VARCHAR(255)," +
        "salary DOUBLE)";
    try (Statement statement = connection.createStatement()) {
        statement.execute(createTableSQL);
    }
}

```

```
}
```

```
private static void addEmployee(Connection connection, Scanner scanner) throws  
SQLException {
```

```
    System.out.print("Enter employee name: ");
```

```
    String name = scanner.nextLine();
```

```
    System.out.print("Enter department: ");
```

```
    String department = scanner.nextLine();
```

```
    System.out.print("Enter salary: ");
```

```
    double salary = scanner.nextDouble();
```

```
    scanner.nextLine(); // Consume newline
```

```
    String insertSQL = "INSERT INTO employees (name, department, salary) VALUES (?, ?, ?)";
```

```
    try (PreparedStatement preparedStatement = connection.prepareStatement(insertSQL)) {
```

```
        preparedStatement.setString(1, name);
```

```
        preparedStatement.setString(2, department);
```

```
        preparedStatement.setDouble(3, salary);
```

```
        int rowsAffected = preparedStatement.executeUpdate();
```

```
        if (rowsAffected > 0) {
```

```
            System.out.println("Employee added successfully.");
```

```
        } else {
```

```
            System.out.println("Employee could not be added.");
```

```
        }
```

```
    }
```

```
}
```

```
private static void updateEmployee(Connection connection, Scanner scanner) throws  
SQLException {
```

```
    System.out.print("Enter employee ID to update: ");
```

```
int id = scanner.nextInt();

scanner.nextLine(); // Consume newline


System.out.print("Enter new name (or press Enter to skip): ");

String name = scanner.nextLine();

System.out.print("Enter new department (or press Enter to skip): ");

String department = scanner.nextLine();


System.out.print("Enter new salary (or press Enter to skip): ");

String salaryStr = scanner.nextLine();


StringBuilder updateSQL = new StringBuilder("UPDATE employees SET ");

if (!name.isEmpty()) {
    updateSQL.append("name = ?, ");
}

if (!department.isEmpty()) {
    updateSQL.append("department = ?, ");
}

if (!salaryStr.isEmpty()) {
    updateSQL.append("salary = ?, ");
}

updateSQL.delete(updateSQL.length() - 2, updateSQL.length()); // Remove trailing comma
and space

updateSQL.append("WHERE id = ?");


try (PreparedStatement preparedStatement =
connection.prepareStatement(updateSQL.toString())) {

    int parameterIndex = 1;

    if (!name.isEmpty()) {
        preparedStatement.setString(parameterIndex++, name);
    }

    if (!department.isEmpty()) {
```

```

        preparedStatement.setString(parameterIndex++, department);
    }
    if (!salaryStr.isEmpty()) {
        double salary = Double.parseDouble(salaryStr);
        preparedStatement.setDouble(parameterIndex++, salary);
    }
    preparedStatement.setInt(parameterIndex, id);

    int rowsAffected = preparedStatement.executeUpdate();
    if (rowsAffected > 0) {
        System.out.println("Employee updated successfully.");
    } else {
        System.out.println("Employee with ID " + id + " not found.");
    }
}
}

```

```

private static void deleteEmployee(Connection connection, Scanner scanner) throws
SQLException {
    System.out.print("Enter employee ID to delete: ");
    int id = scanner.nextInt();

    String deleteSQL = "DELETE FROM employees WHERE id = ?";
    try (PreparedStatement preparedStatement = connection.prepareStatement(deleteSQL)) {
        preparedStatement.setInt(1, id);

        int rowsAffected = preparedStatement.executeUpdate();
        if (rowsAffected > 0) {
            System.out.println("Employee deleted successfully.");
        } else {
            System.out.println("Employee with ID " + id + " not found.");
        }
    }
}

```

```

    }
}

private static void displayAllEmployees(Connection connection) throws SQLException {
    String selectSQL = "SELECT * FROM employees";
    try (Statement statement = connection.createStatement()) {
        ResultSet resultSet = statement.executeQuery(selectSQL) {
            System.out.println("Employee List:");
            System.out.println("ID\tName\tDepartment\tSalary");
            while (resultSet.next()) {
                int id = resultSet.getInt("id");
                String name = resultSet.getString("name");
                String department = resultSet.getString("department");
                double salary = resultSet.getDouble("salary");
                System.out.println(id + "\t" + name + "\t" + department + "\t" + salary);
            }
        }
    }
}

```

Code image

```
1 package employeeManagement;
2 import java.sql.*;
3 import java.util.Scanner;
4 public class EmployeeManagement {
5     private static final String JDBC_URL = "jdbc:mysql://localhost:3306/employeeManagement";
6     private static final String JDBC_USER = "root";
7     private static final String JDBC_PASSWORD = "sehani123";
8     public static void main(String[] args) {
9         try (Connection connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/employeeManagement",
10             createTableIfNotExists(connection);
11
12         Scanner scanner = new Scanner(System.in);
13         while (true) {
14             System.out.println("Employee Management System");
15             System.out.println("1. Add Employee");
16             System.out.println("2. Update Employee");
17             System.out.println("3. Delete Employee");
18             System.out.println("4. Display All Employees");
19             System.out.println("5. Exit");
20             System.out.print("Enter your choice: ");
21
22             int choice = scanner.nextInt();
23             scanner.nextLine(); // Consume newline
24
25             switch (choice) {
26                 case 1:
27                     addEmployee(connection, scanner);
28                     break;
29                 case 2:
30                     updateEmployee(connection, scanner);
31                     break;
32                 case 3:
33                     deleteEmployee(connection, scanner);
34                     break;
35                 case 4:
36                     displayAllEmployees(connection);
37             }
38         }
39     }
40 }
```

```
37     displayAllEmployees(connection);
38     break;
39     case 5:
40         System.out.println("Exiting...");
41         return;
42     default:
43         System.out.println("Invalid choice. Please try again.");
44 }
45 }
46 } catch (SQLException e) {
47     e.printStackTrace();
48 }
49 }
50
51 private static void createTableIfNotExists(Connection connection) throws SQLException {
52     String createTableSQL = "CREATE TABLE IF NOT EXISTS employees (" +
53         "id INT PRIMARY KEY AUTO_INCREMENT, " +
54         "name VARCHAR(255), " +
55         "department VARCHAR(255), " +
56         "salary DOUBLE)";
57     try (Statement statement = connection.createStatement()) {
58         statement.execute(createTableSQL);
59     }
60 }
61
62 private static void addEmployee(Connection connection, Scanner scanner) throws SQLException {
63     System.out.print("Enter employee name: ");
64     String name = scanner.nextLine();
65
66     System.out.print("Enter department: ");
67     String department = scanner.nextLine();
68
69     System.out.print("Enter salary: ");
70     double salary = scanner.nextDouble();
71     scanner.nextLine(); // Consume newline
72 }
```

Eclipse IDE - employeemanagement/src/main/java/employeemanagement/EmployeeManagement.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Project Explorer

- Converter
- employeemanagement
 - src/main/java
 - employeemanagement
 - EmployeeManagement.java
 - src/main/resources
 - src/test/java
 - src/test/resources
 - JRE System Library [JavaSE-1.7]
 - Maven Dependencies
 - src
 - target
 - pom.xml
- ex1
- exampleann
- exampleproduct
- helloworlddemo
- hibernate-second.xml
- lab5
- lab6
- Servers
- spring4-jdbctemplate-example2_student
- spring5-hibernetatemplate-annotation
- SPRING-MVC_XML
- Spring-webmvc-anno
- StudentProject

EmployeeManagement.java

```
72
73 String insertSQL = "INSERT INTO employees (name, department, salary) VALUES (?, ?, ?)";
74 try (PreparedStatement preparedStatement = connection.prepareStatement(insertSQL)) {
75     preparedStatement.setString(1, name);
76     preparedStatement.setString(2, department);
77     preparedStatement.setDouble(3, salary);
78
79     int rowsAffected = preparedStatement.executeUpdate();
80     if (rowsAffected > 0) {
81         System.out.println("Employee added successfully.");
82     } else {
83         System.out.println("Employee could not be added.");
84     }
85 }
86
87
88
89 private static void updateEmployee(Connection connection, Scanner scanner) throws SQLException {
90     System.out.print("Enter employee ID to update: ");
91     int id = scanner.nextInt();
92     scanner.nextLine(); // Consume newline
93
94     System.out.print("Enter new name (or press Enter to skip): ");
95     String name = scanner.nextLine();
96
97     System.out.print("Enter new department (or press Enter to skip): ");
98     String department = scanner.nextLine();
99
100     System.out.print("Enter new salary (or press Enter to skip): ");
101     String salaryStr = scanner.nextLine();
102
103     StringBuilder updateSQL = new StringBuilder("UPDATE employees SET ");
104     if (!name.isEmpty()) {
105         updateSQL.append("name = ?, ");
106     }
107     if (!department.isEmpty()) {
108         updateSQL.append("department = ?, ");
109     }
```

Writable Smart Insert 7:57:311

Eclipse IDE - employeemanagement/src/main/java/employeemanagement/EmployeeManagement.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Project Explorer

- Converter
- employeemanagement
 - src/main/java
 - employeemanagement
 - EmployeeManagement.java
 - src/main/resources
 - src/test/java
 - src/test/resources
 - JRE System Library [JavaSE-1.7]
 - Maven Dependencies
 - src
 - target
 - pom.xml
- ex1
- exampleann
- exampleproduct
- helloworlddemo
- hibernate-second.xml
- lab5
- lab6
- Servers
- spring4-jdbctemplate-example2_student
- spring5-hibernetatemplate-annotation
- SPRING-MVC_XML
- Spring-webmvc-anno
- StudentProject

EmployeeManagement.java

```
108 }
109 if (!salaryStr.isEmpty()) {
110     updateSQL.append("salary = ?, ");
111 }
112 updateSQL.delete(updateSQL.length() - 2, updateSQL.length()); // Remove trailing comma and space
113 updateSQL.append("WHERE id = ?");
114
115 try (PreparedStatement preparedStatement = connection.prepareStatement(updateSQL.toString())) {
116     int parameterIndex = 1;
117     if (!name.isEmpty()) {
118         preparedStatement.setString(parameterIndex++, name);
119     }
120     if (!department.isEmpty()) {
121         preparedStatement.setString(parameterIndex++, department);
122     }
123     if (!salaryStr.isEmpty()) {
124         double salary = Double.parseDouble(salaryStr);
125         preparedStatement.setDouble(parameterIndex++, salary);
126     }
127     preparedStatement.setInt(parameterIndex, id);
128
129     int rowsAffected = preparedStatement.executeUpdate();
130     if (rowsAffected > 0) {
131         System.out.println("Employee updated successfully.");
132     } else {
133         System.out.println("Employee with ID " + id + " not found.");
134     }
135 }
136
137
138 private static void deleteEmployee(Connection connection, Scanner scanner) throws SQLException {
139     System.out.print("Enter employee ID to delete: ");
140     int id = scanner.nextInt();
141
142     String deleteSQL = "DELETE FROM employees WHERE id = ?";
143     try (PreparedStatement preparedStatement = connection.prepareStatement(deleteSQL)) {
144         preparedStatement.setInt(1, id);
```

Writable Smart Insert 7:57:311


```
139 private static void deleteEmployee(Connection connection, Scanner scanner) throws SQLException {
140     System.out.print("Enter employee ID to delete: ");
141     int id = scanner.nextInt();
142
143     String deleteSQL = "DELETE FROM employees WHERE id = ?";
144     try (PreparedStatement preparedStatement = connection.prepareStatement(deleteSQL)) {
145         preparedStatement.setInt(1, id);
146
147         int rowsAffected = preparedStatement.executeUpdate();
148         if (rowsAffected > 0) {
149             System.out.println("Employee deleted successfully.");
150         } else {
151             System.out.println("Employee with ID " + id + " not found.");
152         }
153     }
154 }
155
156 private static void displayAllEmployees(Connection connection) throws SQLException {
157     String selectSQL = "SELECT * FROM employees";
158     try (Statement statement = connection.createStatement()) {
159         ResultSet resultSet = statement.executeQuery(selectSQL);
160         System.out.println("Employee List:");
161         System.out.println("ID\tName\tDepartment\tSalary");
162         while (resultSet.next()) {
163             int id = resultSet.getInt("id");
164             String name = resultSet.getString("name");
165             String department = resultSet.getString("department");
166             double salary = resultSet.getDouble("salary");
167             System.out.println(id + "\t" + name + "\t" + department + "\t" + salary);
168         }
169     }
170 }
171 }
172
173
174
```

Output:

```
EmployeeManagement (Java Application) C:\Users\mohammed.nasir\AppData\Local\Temp\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64.18.0.1
Employee Management System
1. Add Employee
2. Update Employee
3. Delete Employee
4. Display All Employees
5. Exit
Enter your choice: 1
Enter employee name: john
Enter department: uidesign
Enter salary: 560000
Employee added successfully.
Employee Management System
1. Add Employee
2. Update Employee
3. Delete Employee
4. Display All Employees
5. Exit
Enter your choice: 1
Enter employee name: don
Enter department: webdev
Enter salary: 230000
Employee added successfully.
Employee Management System
1. Add Employee
2. Update Employee
3. Delete Employee
4. Display All Employees
5. Exit
Enter your choice: 1
Enter employee name: joe
Enter department: audev
Enter salary: 450000
Employee added successfully.
Employee Management System
1. Add Employee
2. Update Employee
```

