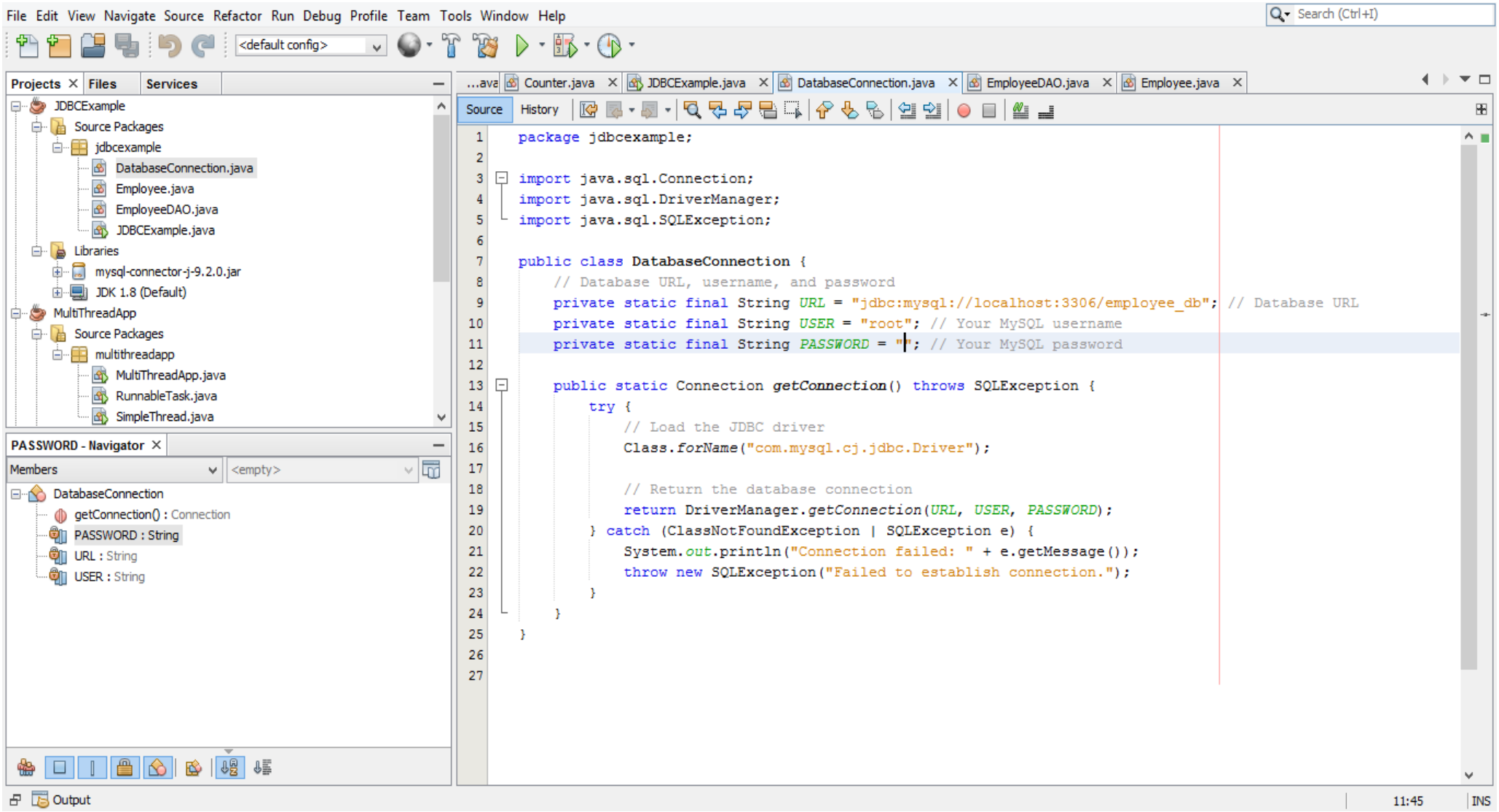
1. Create a DatabaseConnection.java class to establish a connection to your database

```
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, USER, PASSWORD);
        } catch (ClassNotFoundException | SQLException e) {
            System.out.println("Connection failed: " + e.getMessage());
            throw new SQLException("Failed to establish connection.");
        }
    }
}
```


File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
<default config>
Projects
Files
Services
JDBCExample
Source Packages
jdbcexample
DatabaseConnection.java
Employee.java
EmployeeDAO.java
JDBCExample.java
Libraries
mysql-connector-j-9.2.0.jar
JDK 1.8 (Default)
MultiThreadApp
Source Packages
multithreadapp
MultiThreadApp.java
RunnableTask.java
SimpleThread.java
PASSWORD - Navigator
Members
<empty>
DatabaseConnection
getConnection() : Connection
PASSWORD : String
URL : String
USER : String
Counter.java
JDBCExample.java
DatabaseConnection.java
EmployeeDAO.java
Employee.java
Source
History
1 package jdbcexample;
2
3 import java.sql.Connection;
4 import java.sql.DriverManager;
5 import java.sql.SQLException;
6
7 public class DatabaseConnection {
8 // Database URL, username, and password
9 private static final String URL = "jdbc:mysql://localhost:3306/employee_db"; // Database URL
10 private static final String USER = "root"; // Your MySQL username
11 private static final String PASSWORD = ""; // Your MySQL password
12
13 public static Connection getConnection() throws SQLException {
14 try {
15 // Load the JDBC driver
16 Class.forName("com.mysql.cj.jdbc.Driver");
17
18 // Return the database connection
19 return DriverManager.getConnection(URL, USER, PASSWORD);
20 } catch (ClassNotFoundException | SQLException e) {
21 System.out.println("Connection failed: " + e.getMessage());
22 throw new SQLException("Failed to establish connection.");
23 }
24 }
25 }
26
27
Output
11:45
INS

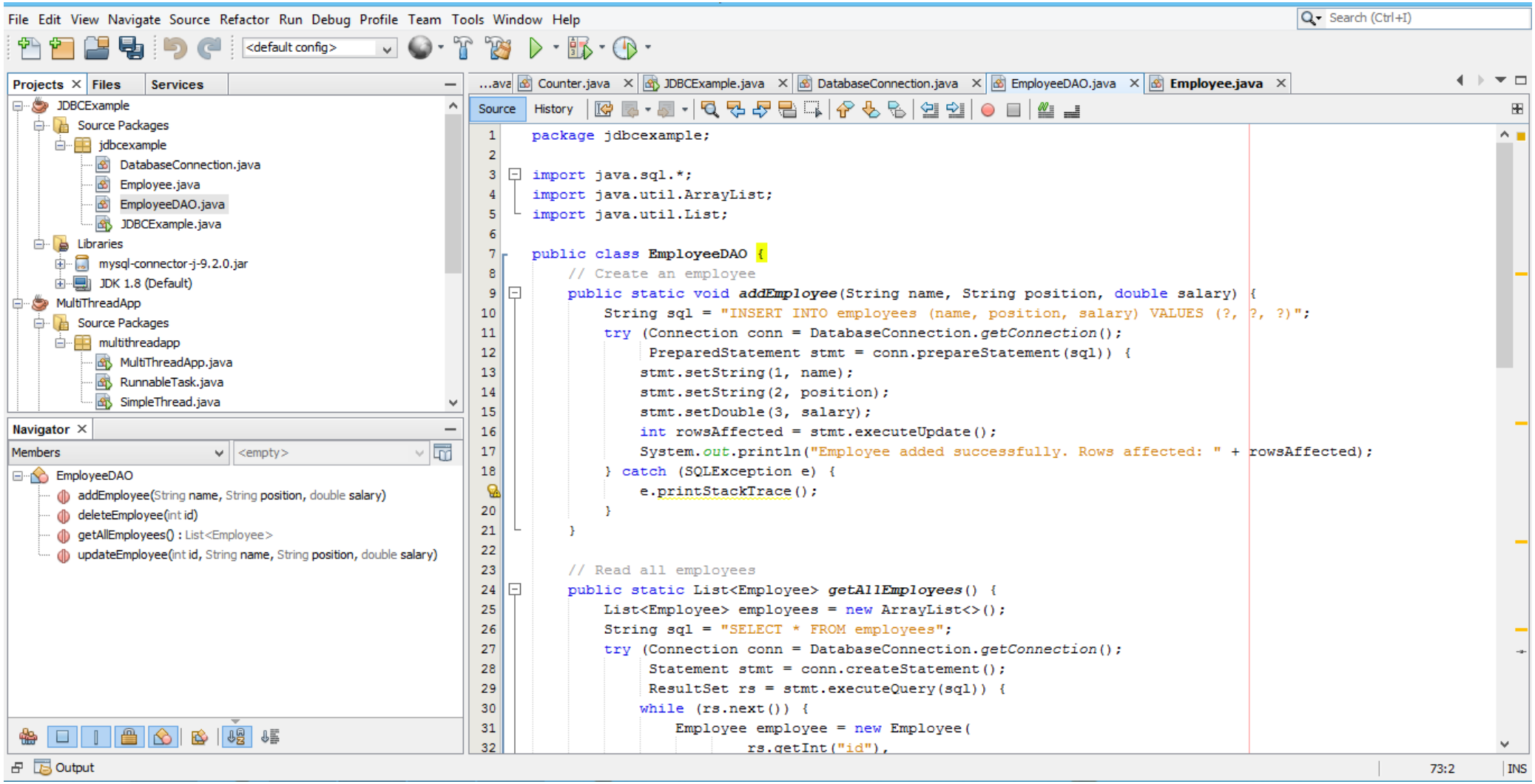
2. Create EmployeeDAO.java for CRUD Operations

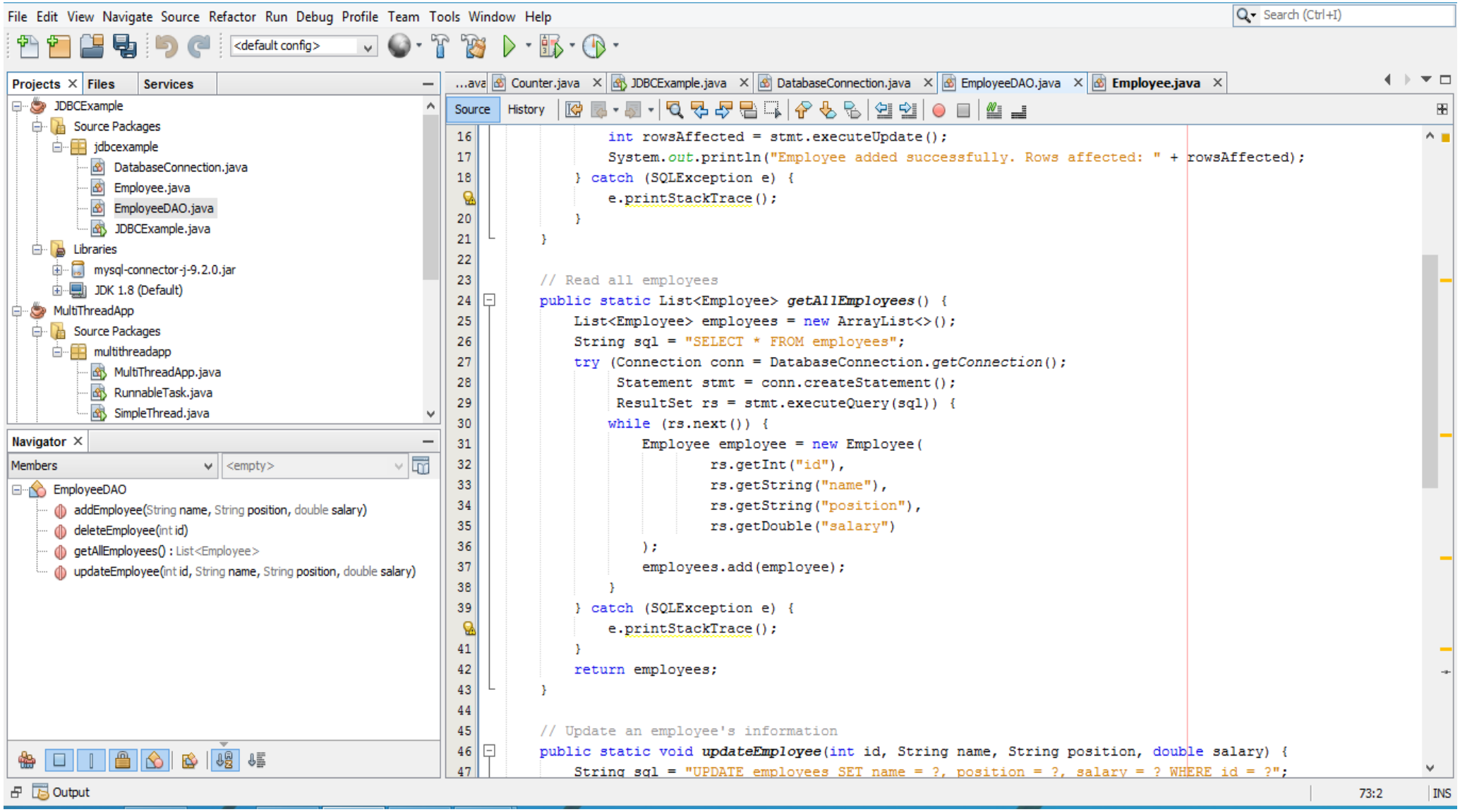
```
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

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();
```

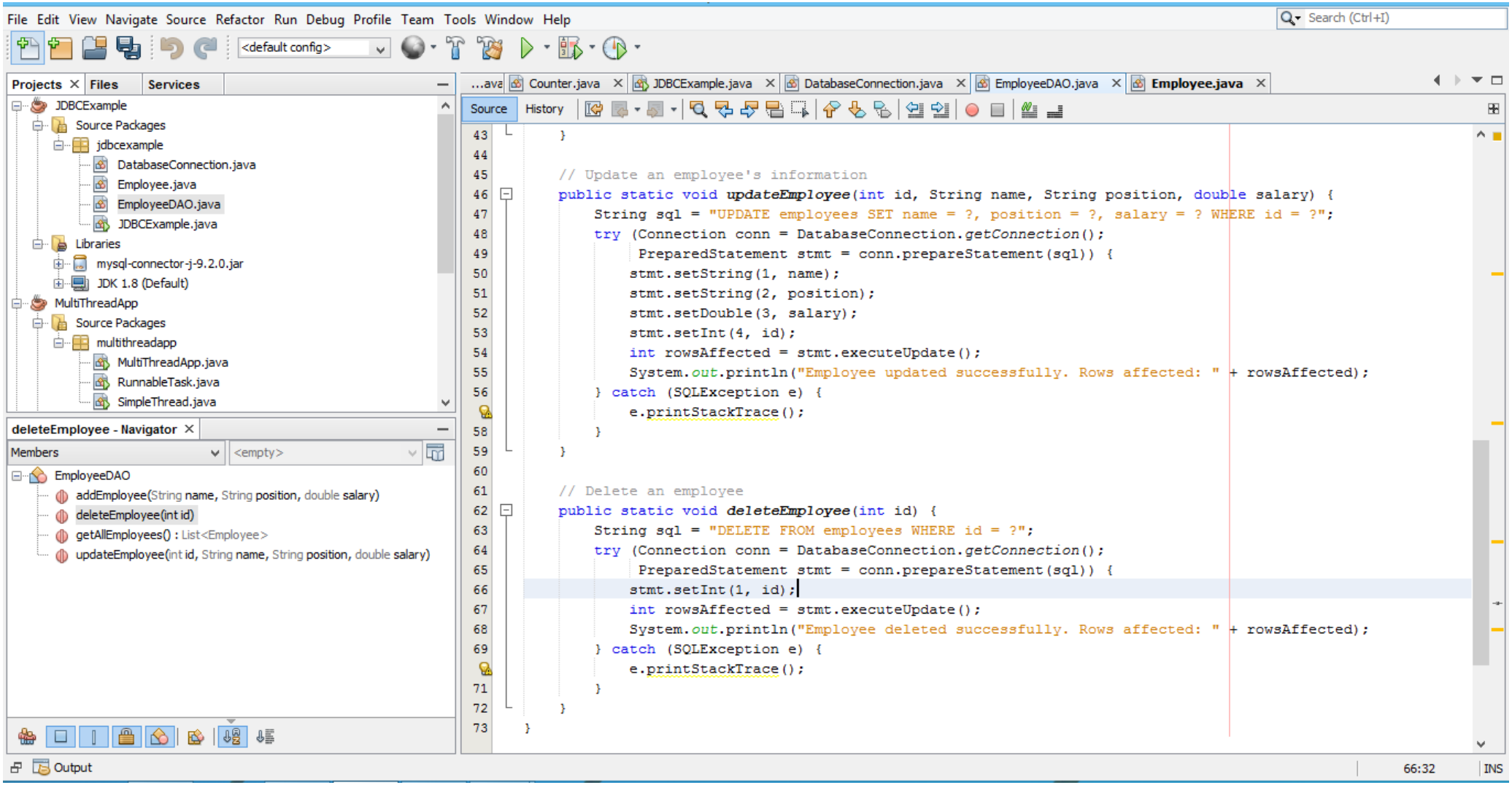
```
    }
}

// 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();
        }
    }
}
```


File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
<default config>
Projects
Files
Services
JDBCExample
Source Packages
jdbcexample
DatabaseConnection.java
Employee.java
EmployeeDAO.java
JDBCExample.java
Libraries
mysql-connector-j-9.2.0.jar
JDK 1.8 (Default)
MultiThreadApp
Source Packages
multithreadapp
MultiThreadApp.java
RunnableTask.java
SimpleThread.java
Navigator
Members
<empty>
EmployeeDAO
addEmployee(String name, String position, double salary)
deleteEmployee(int id)
getAllEmployees() : List<Employee>
updateEmployee(int id, String name, String position, double salary)
Counter.java
JDBCExample.java
DatabaseConnection.java
EmployeeDAO.java
Employee.java
Source
History
1 package jdbcexample;
2
3 import java.sql.*;
4 import java.util.ArrayList;
5 import java.util.List;
6
7 public class EmployeeDAO {
8 // Create an employee
9 public static void addEmployee(String name, String position, double salary) {
10 String sql = "INSERT INTO employees (name, position, salary) VALUES (?, ?, ?)";
11 try (Connection conn = DatabaseConnection.getConnection();
12 PreparedStatement stmt = conn.prepareStatement(sql)) {
13 stmt.setString(1, name);
14 stmt.setString(2, position);
15 stmt.setDouble(3, salary);
16 int rowsAffected = stmt.executeUpdate();
17 System.out.println("Employee added successfully. Rows affected: " + rowsAffected);
18 } catch (SQLException e) {
e.printStackTrace();
20 }
21 }
22
23 // Read all employees
24 public static List<Employee> getAllEmployees() {
25 List<Employee> employees = new ArrayList<>();
26 String sql = "SELECT * FROM employees";
27 try (Connection conn = DatabaseConnection.getConnection();
28 Statement stmt = conn.createStatement();
29 ResultSet rs = stmt.executeQuery(sql)) {
30 while (rs.next()) {
31 Employee employee = new Employee(
32 rs.getInt("id"),
Output
73:2
INS

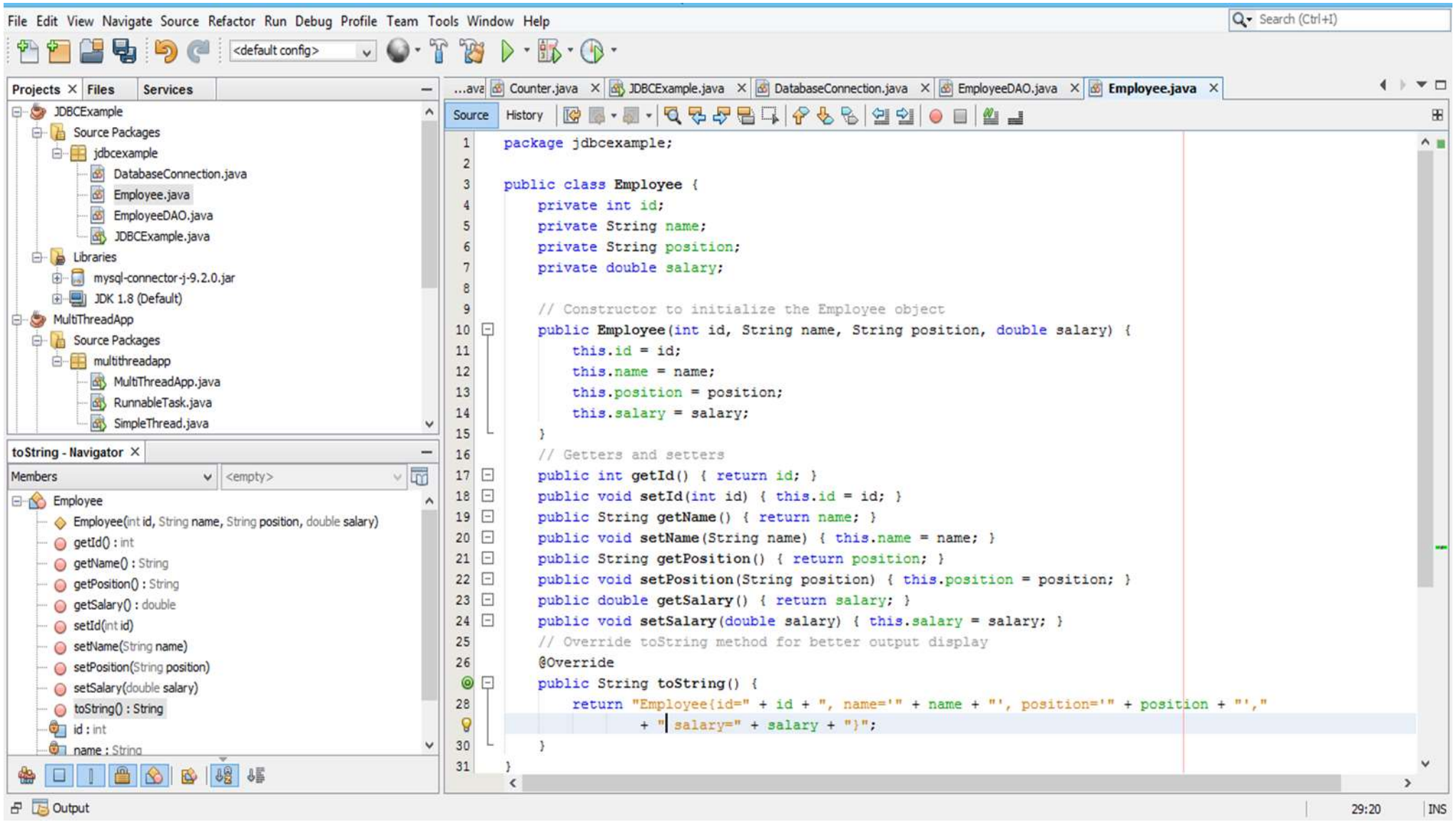

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
<default config>
Projects
Files
Services
JDBCExample
Source Packages
jdbcexample
DatabaseConnection.java
Employee.java
EmployeeDAO.java
JDBCExample.java
Libraries
mysql-connector-j-9.2.0.jar
JDK 1.8 (Default)
MultiThreadApp
Source Packages
multithreadapp
MultiThreadApp.java
RunnableTask.java
SimpleThread.java
Navigator
Members
<empty>
EmployeeDAO
addEmployee(String name, String position, double salary)
deleteEmployee(int id)
getAllEmployees() : List<Employee>
updateEmployee(int id, String name, String position, double salary)
Counter.java
JDBCExample.java
DatabaseConnection.java
EmployeeDAO.java
Employee.java
Source
History
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 = ?";
Output
73:2
INS


File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
<default config>
Projects
Files
Services
JDBCExample
Source Packages
jdbcexample
DatabaseConnection.java
Employee.java
EmployeeDAO.java
JDBCExample.java
Libraries
mysql-connector-j-9.2.0.jar
JDK 1.8 (Default)
MultiThreadApp
Source Packages
multithreadapp
MultiThreadApp.java
RunnableTask.java
SimpleThread.java
deleteEmployee - Navigator
Members
<empty>
EmployeeDAO
addEmployee(String name, String position, double salary)
deleteEmployee(int id)
getAllEmployees() : List<Employee>
updateEmployee(int id, String name, String position, double salary)
Counter.java
JDBCExample.java
DatabaseConnection.java
EmployeeDAO.java
Employee.java
Source
History
43 }
44
45 // Update an employee's information
46 public static void updateEmployee(int id, String name, String position, double salary) {
47 String sql = "UPDATE employees SET name = ?, position = ?, salary = ? WHERE id = ?";
48 try (Connection conn = DatabaseConnection.getConnection();
49 PreparedStatement stmt = conn.prepareStatement(sql)) {
50 stmt.setString(1, name);
51 stmt.setString(2, position);
52 stmt.setDouble(3, salary);
53 stmt.setInt(4, id);
54 int rowsAffected = stmt.executeUpdate();
55 System.out.println("Employee updated successfully. Rows affected: " + rowsAffected);
56 } catch (SQLException e) {
e.printStackTrace();
58 }
59 }
60
61 // Delete an employee
62 public static void deleteEmployee(int id) {
63 String sql = "DELETE FROM employees WHERE id = ?";
64 try (Connection conn = DatabaseConnection.getConnection();
65 PreparedStatement stmt = conn.prepareStatement(sql)) {
66 stmt.setInt(1, id);
67 int rowsAffected = stmt.executeUpdate();
68 System.out.println("Employee deleted successfully. Rows affected: " + rowsAffected);
69 } catch (SQLException e) {
e.printStackTrace();
71 }
72 }
73 }
Output
66:32
INS

3. Create Employee.java Class

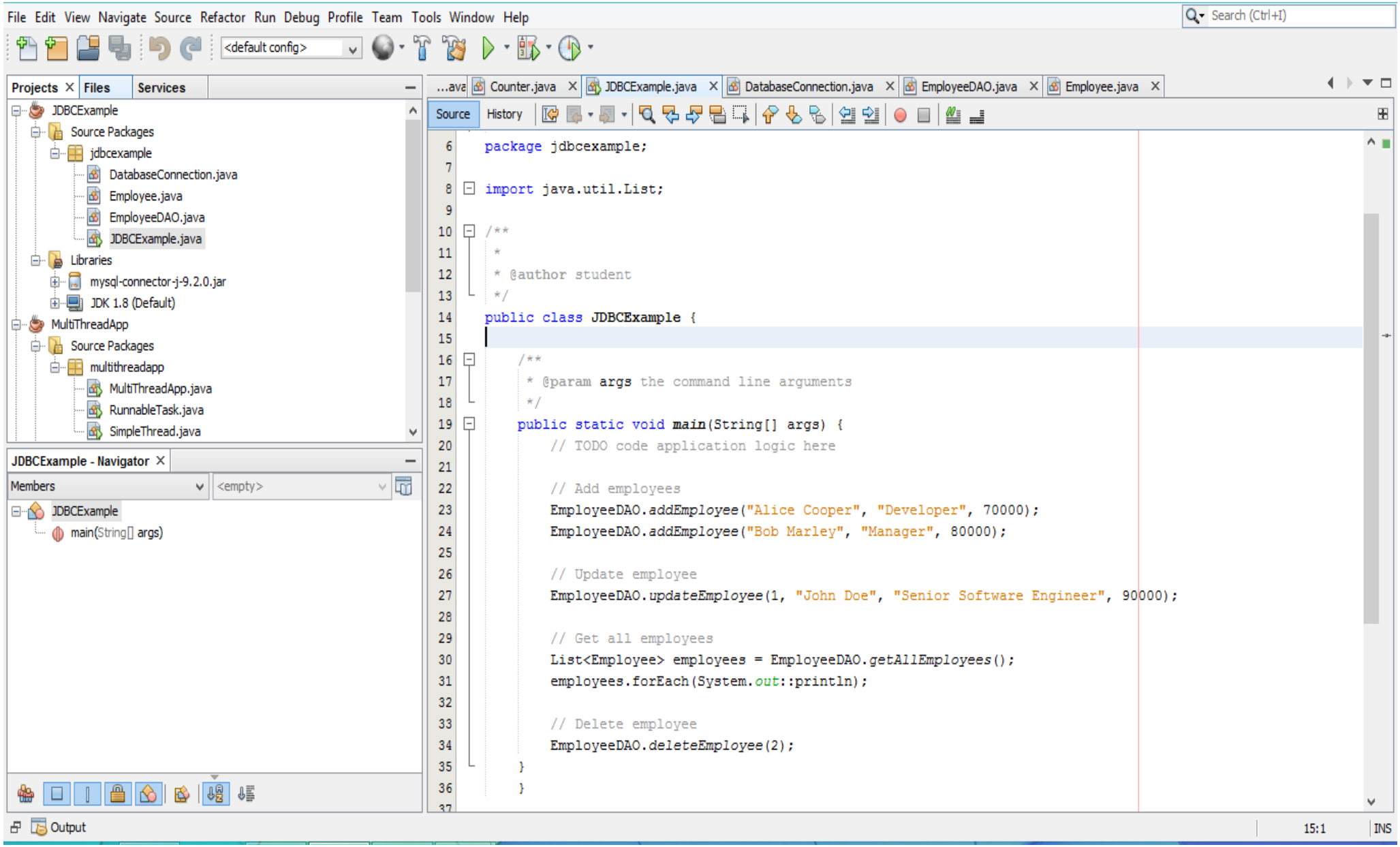
```
public class Employee {
    private int id;
    private String name;
    private String position;
    private double salary;

    public Employee(int id, String name, String position, double salary) {
        this.id = id;
        this.name = name;
        this.position = position;
        this.salary = salary;
    }

    // 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; }
```

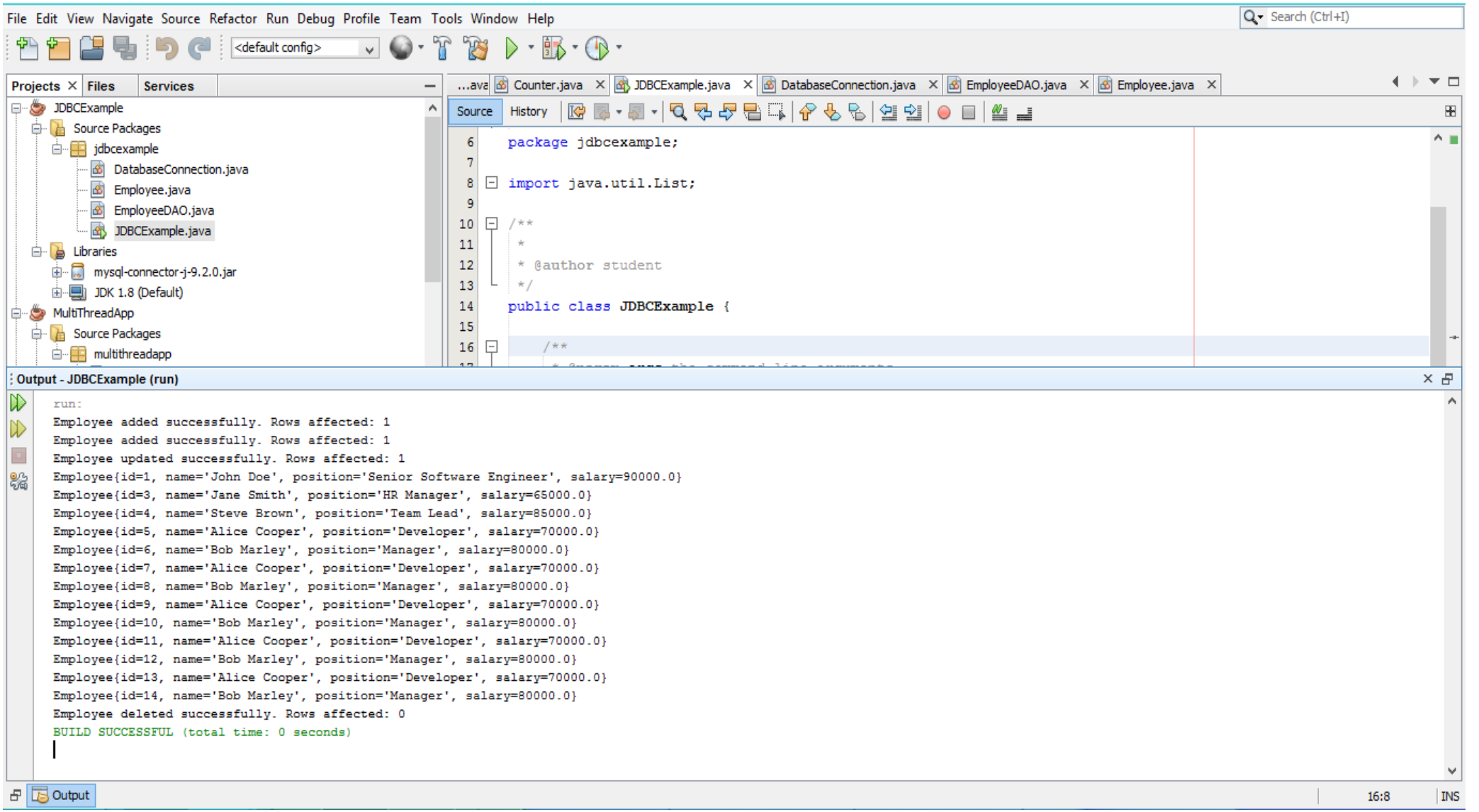
```
    public String getPosition() { return position; }
    public void setPosition(String position) { this.position = position; }

    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 + '}';
    }
}
```


File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
<default config>
Projects
Files
Services
JDBCExample
Source Packages
jdbcexample
DatabaseConnection.java
Employee.java
EmployeeDAO.java
JDBCExample.java
Libraries
mysql-connector-j-9.2.0.jar
JDK 1.8 (Default)
MultiThreadApp
Source Packages
multithreadapp
MultiThreadApp.java
RunnableTask.java
SimpleThread.java
toString - Navigator
Members
<empty>
Employee
Employee(int id, String name, String position, double salary)
getId() : int
getName() : String
getPosition() : String
getSalary() : double
setId(int id)
setName(String name)
setPosition(String position)
setSalary(double salary)
toString() : String
id : int
name : String
...ava
Counter.java
JDBCExample.java
DatabaseConnection.java
EmployeeDAO.java
Employee.java
Source
History
1 package jdbcexample;
2
3 public class Employee {
4 private int id;
5 private String name;
6 private String position;
7 private double salary;
8
9 // Constructor to initialize the Employee object
10 public Employee(int id, String name, String position, double salary) {
11 this.id = id;
12 this.name = name;
13 this.position = position;
14 this.salary = salary;
15 }
16 // Getters and setters
17 public int getId() { return id; }
18 public void setId(int id) { this.id = id; }
19 public String getName() { return name; }
20 public void setName(String name) { this.name = name; }
21 public String getPosition() { return position; }
22 public void setPosition(String position) { this.position = position; }
23 public double getSalary() { return salary; }
24 public void setSalary(double salary) { this.salary = salary; }
25 // Override toString method for better output display
26 @Override
27 public String toString() {
28 return "Employee{id=" + id + ", name='" + name + "', position='" + position + "',"
+ " salary=" + salary + "}";
30 }
31 }
Output
29:20
INS

4. Create a Main.java class to test the CRUD operations

```
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);
    }
}
```


File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
<default config>
Projects
Files
Services
JDBCExample
Source Packages
jdbcexample
DatabaseConnection.java
Employee.java
EmployeeDAO.java
JDBCExample.java
Libraries
mysql-connector-j-9.2.0.jar
JDK 1.8 (Default)
MultiThreadApp
Source Packages
multithreadapp
MultiThreadApp.java
RunnableTask.java
SimpleThread.java
JDBCExample - Navigator
Members
<empty>
JDBCExample
main(String[] args)
Counter.java
JDBCExample.java
DatabaseConnection.java
EmployeeDAO.java
Employee.java
Source
History
6 package jdbcexample;
8 import java.util.List;
10 /**
11 *
12 * @author student
13 */
14 public class JDBCExample {
16 /**
17 * @param args the command line arguments
18 */
19 public static void main(String[] args) {
20 // TODO code application logic here
22 // Add employees
23 EmployeeDAO.addEmployee("Alice Cooper", "Developer", 70000);
24 EmployeeDAO.addEmployee("Bob Marley", "Manager", 80000);
26 // Update employee
27 EmployeeDAO.updateEmployee(1, "John Doe", "Senior Software Engineer", 90000);
29 // Get all employees
30 List<Employee> employees = EmployeeDAO.getAllEmployees();
31 employees.forEach(System.out::println);
33 // Delete employee
34 EmployeeDAO.deleteEmployee(2);
35 }
36 }
Output
15:1
INS


File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
<default config>
Projects
Files
Services
JDBCExample
Source Packages
jdbcexample
DatabaseConnection.java
Employee.java
EmployeeDAO.java
JDBCExample.java
Libraries
mysql-connector-j-9.2.0.jar
JDK 1.8 (Default)
MultiThreadApp
Source Packages
multithreadapp
Counter.java
JDBCExample.java
DatabaseConnection.java
EmployeeDAO.java
Employee.java
Source
History
6 package jdbcexample;
7
8 import java.util.List;
9
10 /**
11 *
12 * @author student
13 */
14 public class JDBCExample {
15
16 /**
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=3, name='Jane Smith', position='HR Manager', salary=65000.0}
Employee{id=4, name='Steve Brown', position='Team Lead', salary=85000.0}
Employee{id=5, name='Alice Cooper', position='Developer', salary=70000.0}
Employee{id=6, name='Bob Marley', position='Manager', salary=80000.0}
Employee{id=7, name='Alice Cooper', position='Developer', salary=70000.0}
Employee{id=8, name='Bob Marley', position='Manager', salary=80000.0}
Employee{id=9, name='Alice Cooper', position='Developer', salary=70000.0}
Employee{id=10, name='Bob Marley', position='Manager', salary=80000.0}
Employee{id=11, name='Alice Cooper', position='Developer', salary=70000.0}
Employee{id=12, name='Bob Marley', position='Manager', salary=80000.0}
Employee{id=13, name='Alice Cooper', position='Developer', salary=70000.0}
Employee{id=14, name='Bob Marley', position='Manager', salary=80000.0}
Employee deleted successfully. Rows affected: 0
BUILD SUCCESSFUL (total time: 0 seconds)
Output
16:8
INS