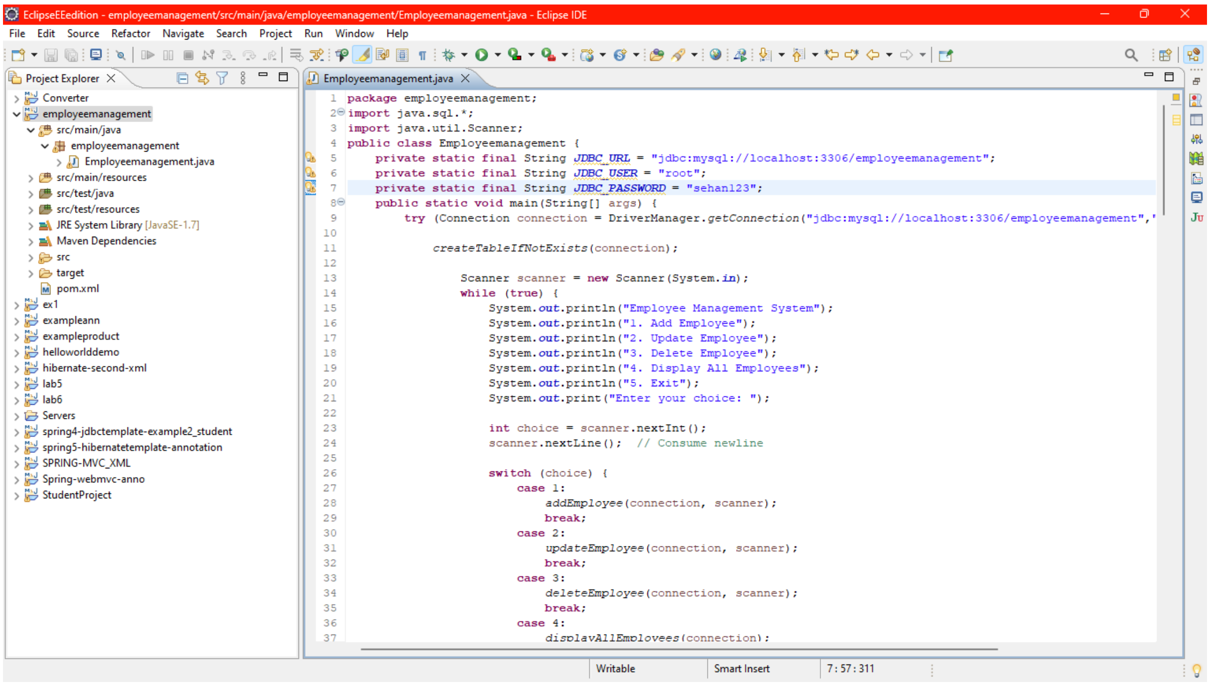
JDBC Assignment

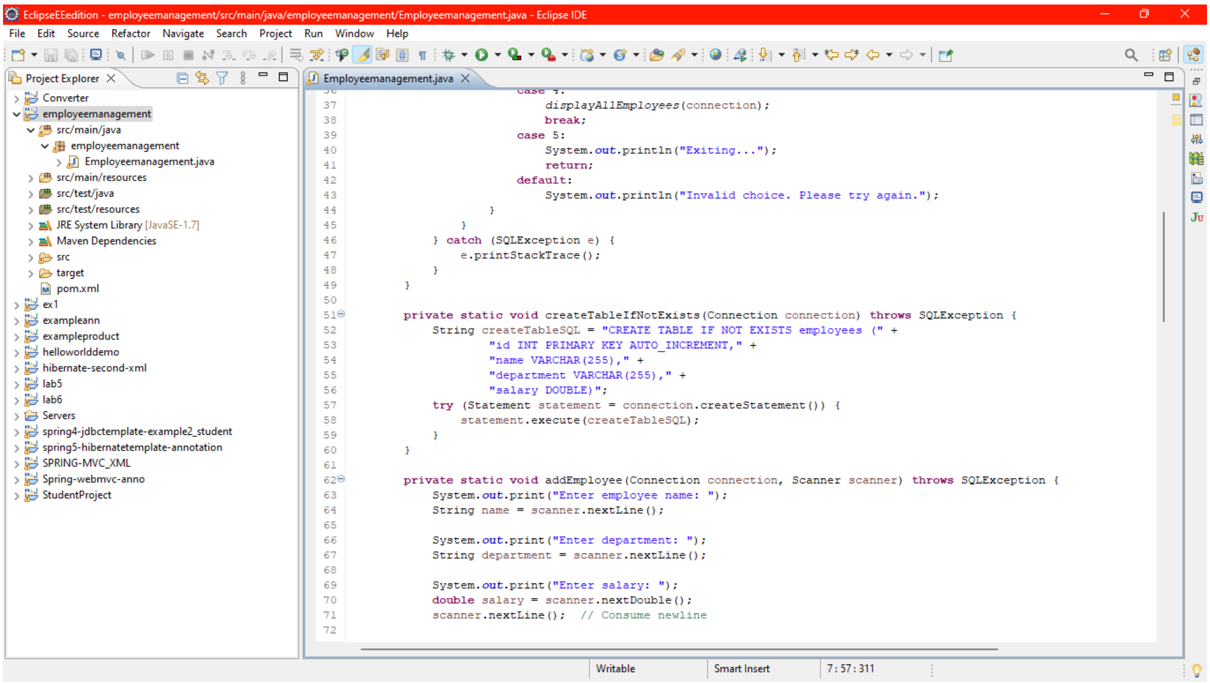
```
package employeemanagement;

import java.sql.*;

import java.util.Scanner;

public class Employeemanagement {

 private static final String JDBC_URL = "jdbc:mysql://localhost:3306/employeemanagement";

  private static final String JDBC_USER = "root";

  private static final String JDBC_PASSWORD = "root";

 public static void main(String[] args) {

  try (Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/employeemanagement","root",
"sehan")) {


    createTableIfNotExists(connection);


       Scanner scanner = new Scanner(System.in);

       while (true) {

         System.out.println("Employee Management System");

         System.out.println("1. Add Employee");

         System.out.println("2. Update Employee");

         System.out.println("3. Delete Employee");

         System.out.println("4. Display All Employees");

         System.out.println("5. Exit");

         System.out.print("Enter your choice: ");


         int choice = scanner.nextInt();

         scanner.nextLine();  // Consume newline


         switch (choice) {

           case 1:

             addEmployee(connection, scanner);
```

```
                        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);
            }
        }
    }

}
```

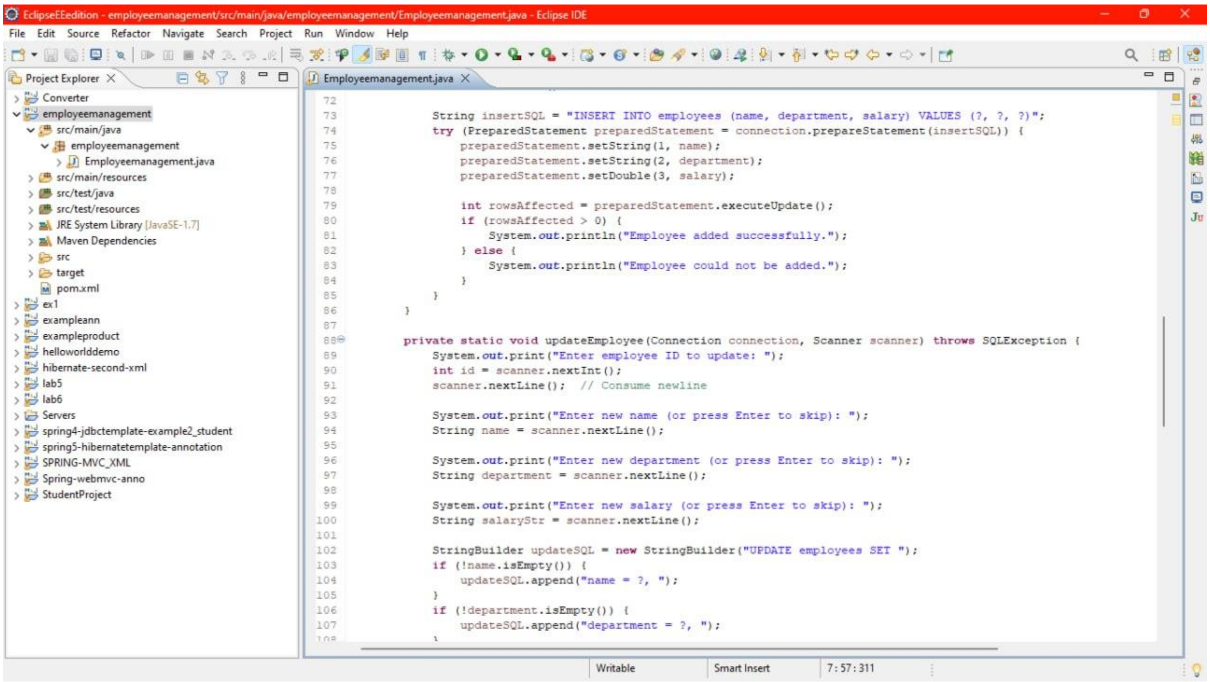
Code image



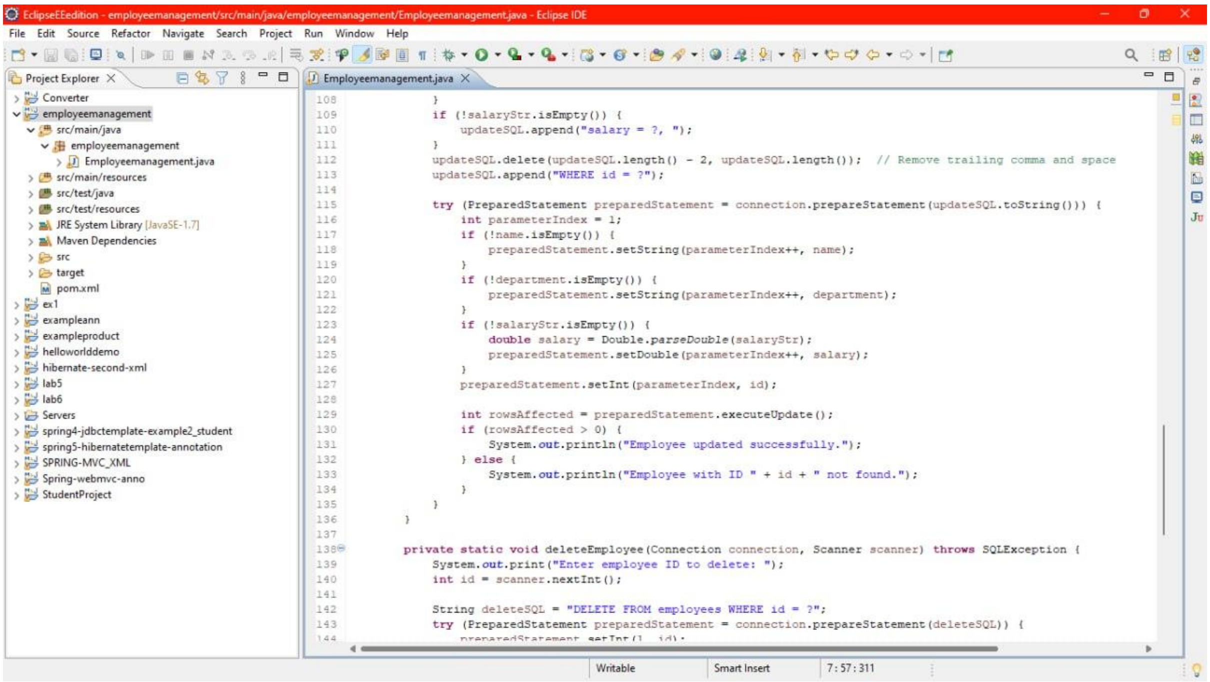
```java
package employeemanagement;
import java.sql.*;
import java.util.Scanner;
public class Employeemanagement {
    private static final String JDBC_URL = "jdbc:mysql://localhost:3306/employeemanagement";
    private static final String JDBC_USER = "root";
    private static final String JDBC_PASSWORD = "sehan123";
    public static void main(String[] args) {
        try (Connection connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/employeemanagement","

            createTableIfNotExists(connection);

                Scanner scanner = new Scanner(System.in);
                while (true) {
                    System.out.println("Employee Management System");
                    System.out.println("1. Add Employee");
                    System.out.println("2. Update Employee");
                    System.out.println("3. Delete Employee");
                    System.out.println("4. Display All Employees");
                    System.out.println("5. Exit");
                    System.out.print("Enter your choice: ");

                    int choice = scanner.nextInt();
                    scanner.nextLine();  // Consume newline

                    switch (choice) {
                        case 1:
                            addEmployee(connection, scanner);
                            break;
                        case 2:
                            updateEmployee(connection, scanner);
                            break;
                        case 3:
                            deleteEmployee(connection, scanner);
                            break;
                        case 4:
                            displayAllEmployees(connection);
```



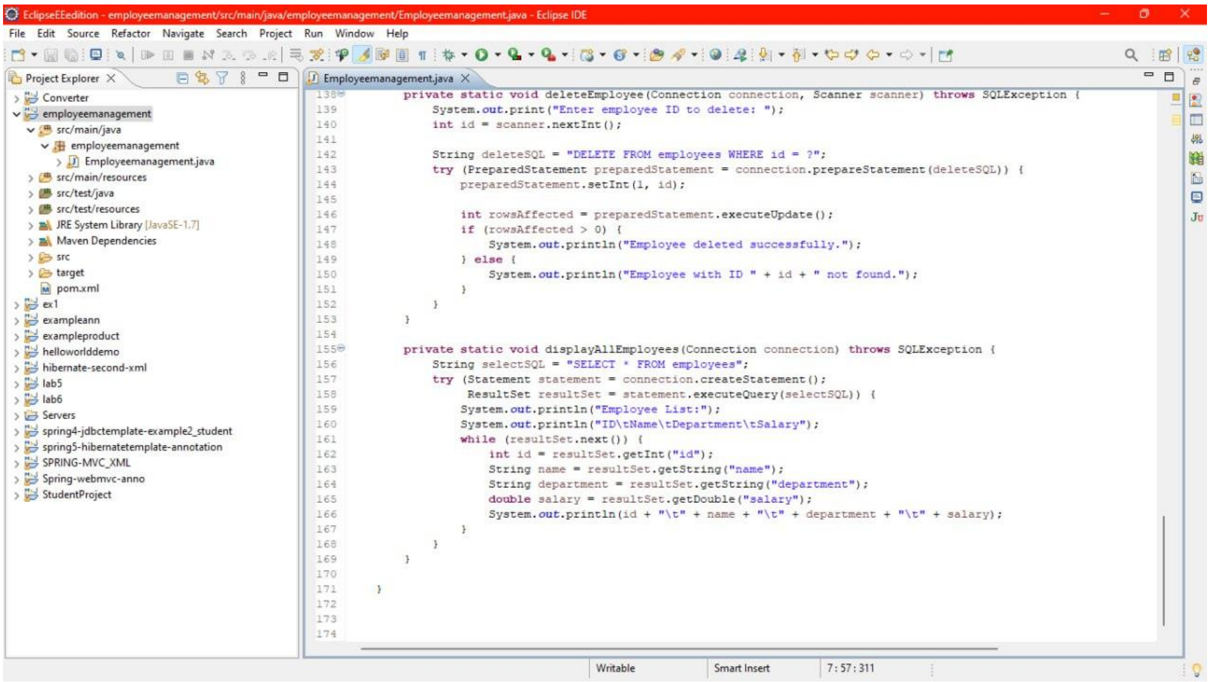
```java
                            displayAllEmployees(connection);
                            break;
                        case 5:
                            System.out.println("Exiting...");
                            return;
                        default:
                            System.out.println("Invalid 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

```



```java

        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 = ?, ");
```

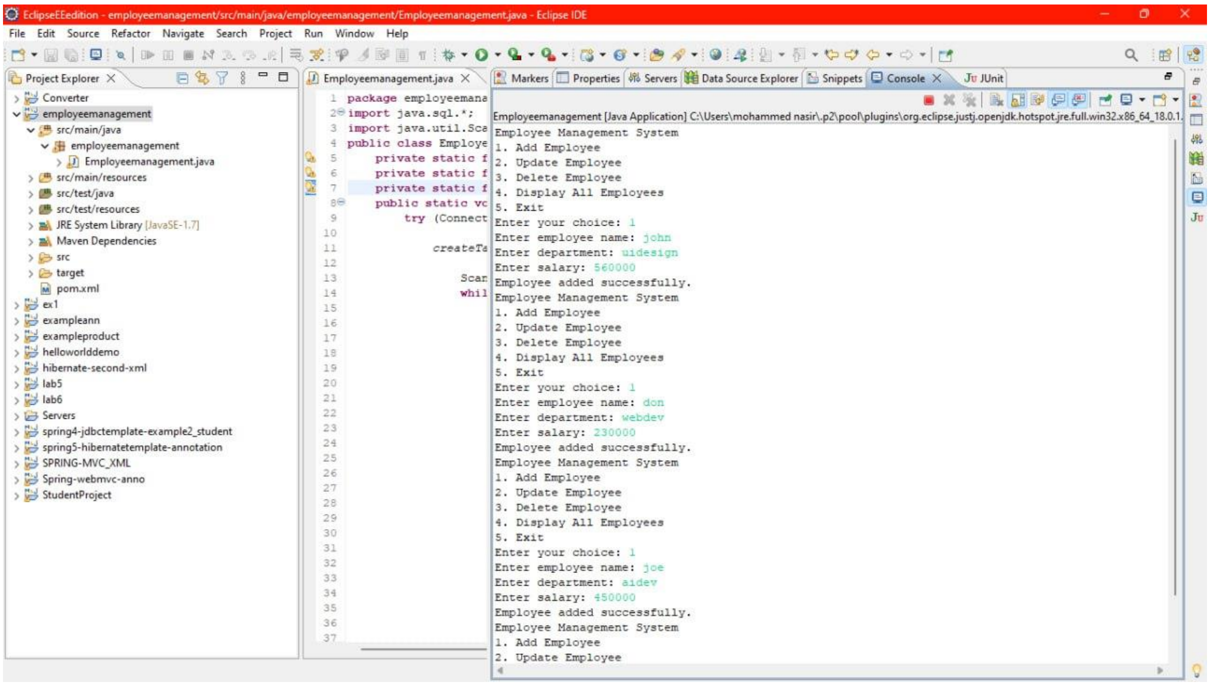


```java
        }
        if (!salaryStr.isEmpty()) {
            updateSQL.append("salary = ?, ");
        }
        updateSQL.delete(updateSQL.length() - 2, updateSQL.length());  // Remove trailing comma and space
        updateSQL.append("WHERE id = ?");

        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);
```



```java
    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);
            }
        }
    }

}
```

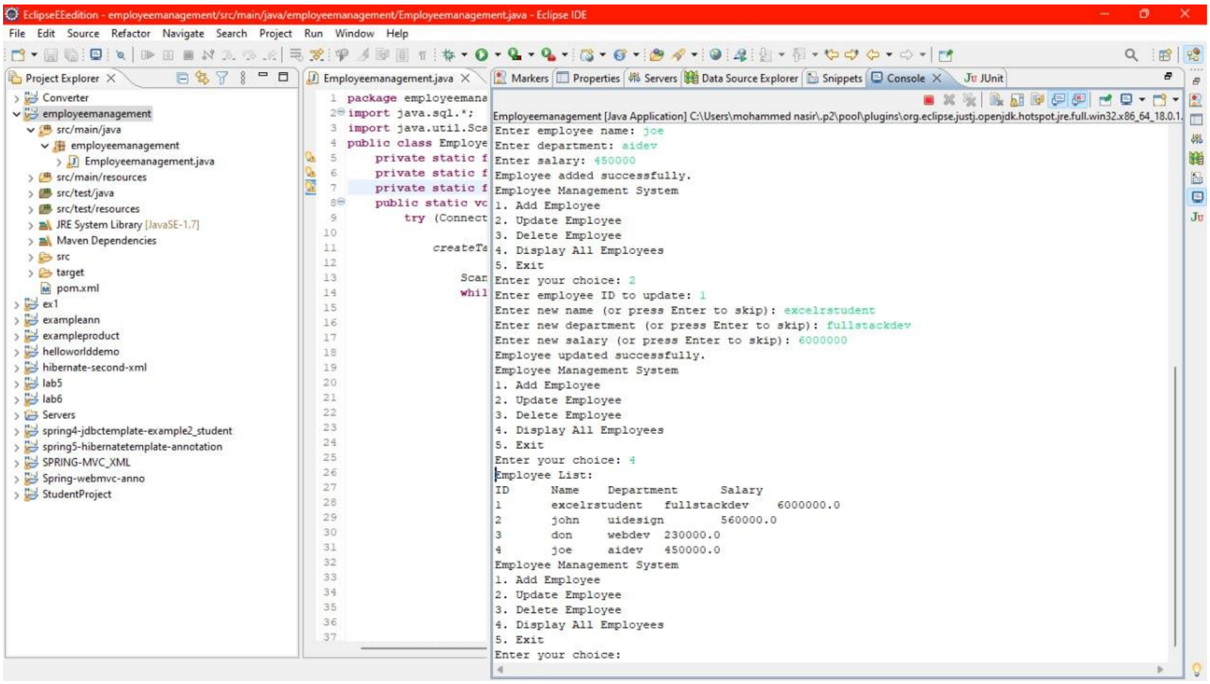
Output:



```java
package employeemana
import java.sql.*;
import java.util.Sca
public class Employe
    private static f
    private static f
    private static f
    public static vc
        try (Connect

            createTa

                Scar
                whil
```

Employeemanagement [Java Application] C:\Users\mohammed nasir\.p2\pool\plugins\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: aidev
Enter salary: 450000
Employee added successfully.
Employee Management System
1. Add Employee
2. Update Employee
```

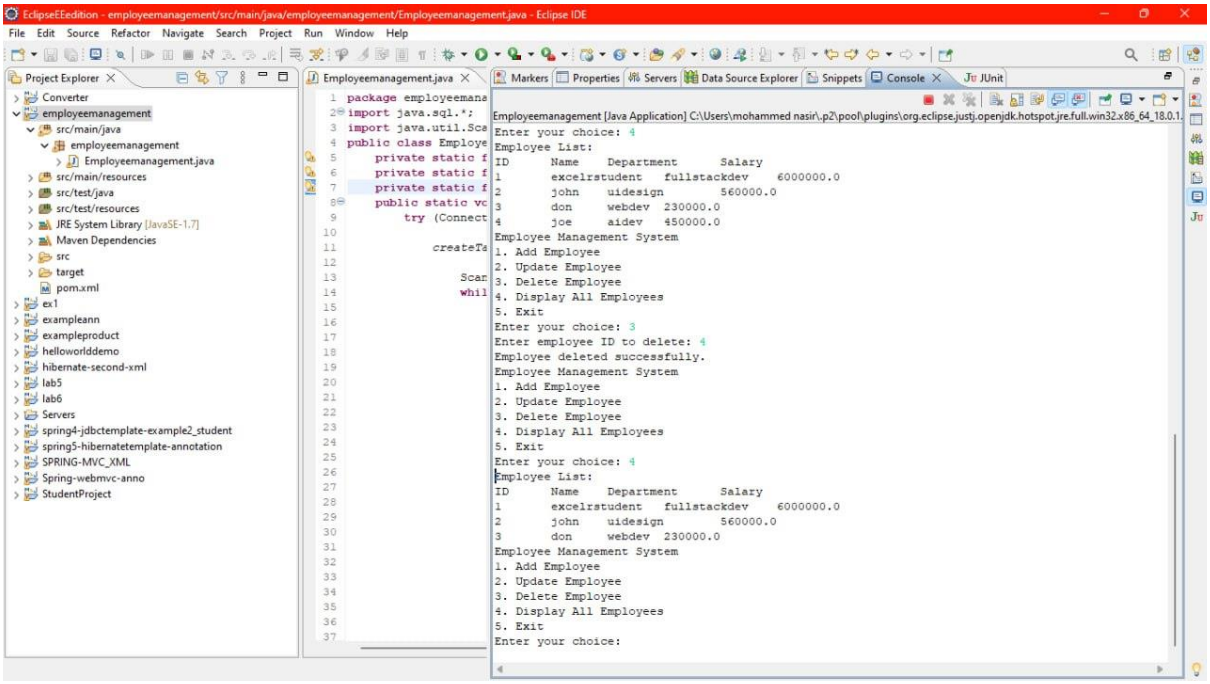


EclipseEEedition - employeemanagement/src/main/java/employeemanagement/Employeemanagement.java - Eclipse IDE

```
package employeemana
import java.sql.*;
import java.util.Sca
public class Employe
    private static f
    private static f
    private static f
    public static vc
        try (Connect

                createTa

                Scar
                whil
```

Employeemanagement [Java Application] C:\Users\mohammed nasir\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_18.0.1.

```
Enter employee name: joe
Enter department: aidev
Enter salary: 450000
Employee added successfully.
Employee Management System
1. Add Employee
2. Update Employee
3. Delete Employee
4. Display All Employees
5. Exit
Enter your choice: 2
Enter employee ID to update: 1
Enter new name (or press Enter to skip): excelrstudent
Enter new department (or press Enter to skip): fullstackdev
Enter new salary (or press Enter to skip): 6000000
Employee updated successfully.
Employee Management System
1. Add Employee
2. Update Employee
3. Delete Employee
4. Display All Employees
5. Exit
Enter your choice: 4
Employee List:
ID      Name    Department      Salary
1       excelrstudent   fullstackdev    6000000.0
2       john    uidesign        560000.0
3       don     webdev  230000.0
4       joe     aidev   450000.0
Employee Management System
1. Add Employee
2. Update Employee
3. Delete Employee
4. Display All Employees
5. Exit
Enter your choice:
```



EclipseEEedition - employeemanagement/src/main/java/employeemanagement/Employeemanagement.java - Eclipse IDE

```
package employeemana
import java.sql.*;
import java.util.Sca
public class Employe
    private static f
    private static f
    private static f
    public static vc
        try (Connect

                createTa

                Scar
                whil
```

Employeemanagement [Java Application] C:\Users\mohammed nasir\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_18.0.1.

```
Enter your choice: 4
Employee List:
ID      Name    Department      Salary
1       excelrstudent   fullstackdev    6000000.0
2       john    uidesign        560000.0
3       don     webdev  230000.0
4       joe     aidev   450000.0
Employee Management System
1. Add Employee
2. Update Employee
3. Delete Employee
4. Display All Employees
5. Exit
Enter your choice: 3
Enter employee ID to delete: 4
Employee deleted successfully.
Employee Management System
1. Add Employee
2. Update Employee
3. Delete Employee
4. Display All Employees
5. Exit
Enter your choice: 4
Employee List:
ID      Name    Department      Salary
1       excelrstudent   fullstackdev    6000000.0
2       john    uidesign        560000.0
3       don     webdev  230000.0
Employee Management System
1. Add Employee
2. Update Employee
3. Delete Employee
4. Display All Employees
5. Exit
Enter your choice:
```