GAM/IT/2022/F/0079 – J.S.Senarath

Java JDBC Lab Practical using NetBeans IDE 8.2

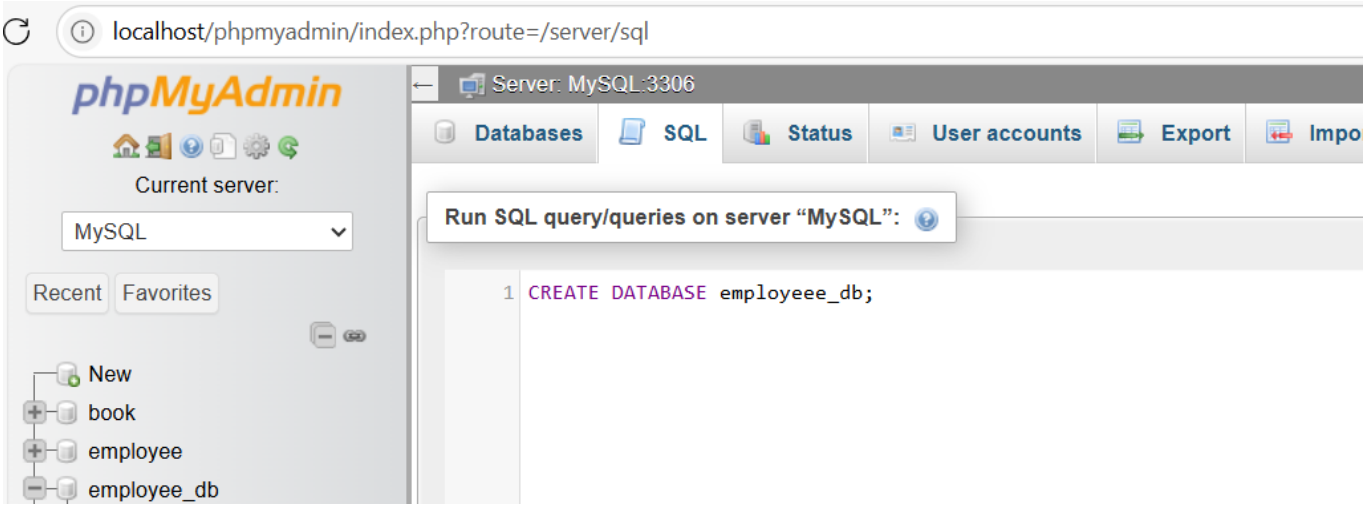
Java JDBC Lab Practical using NetBeans IDE 8.2

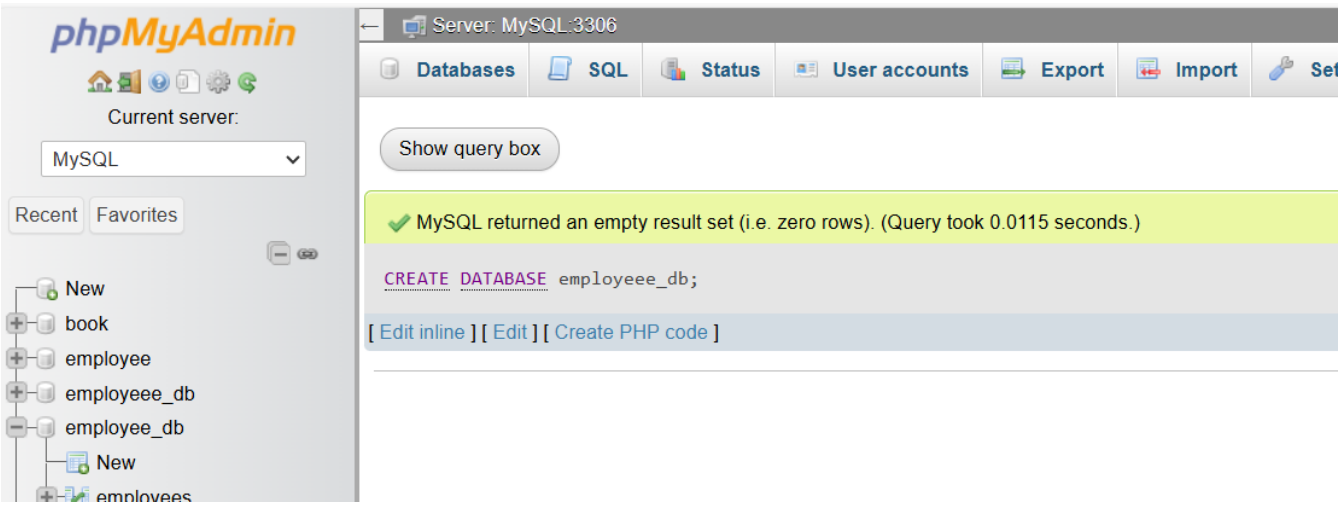
1. Set Up MySQL Database

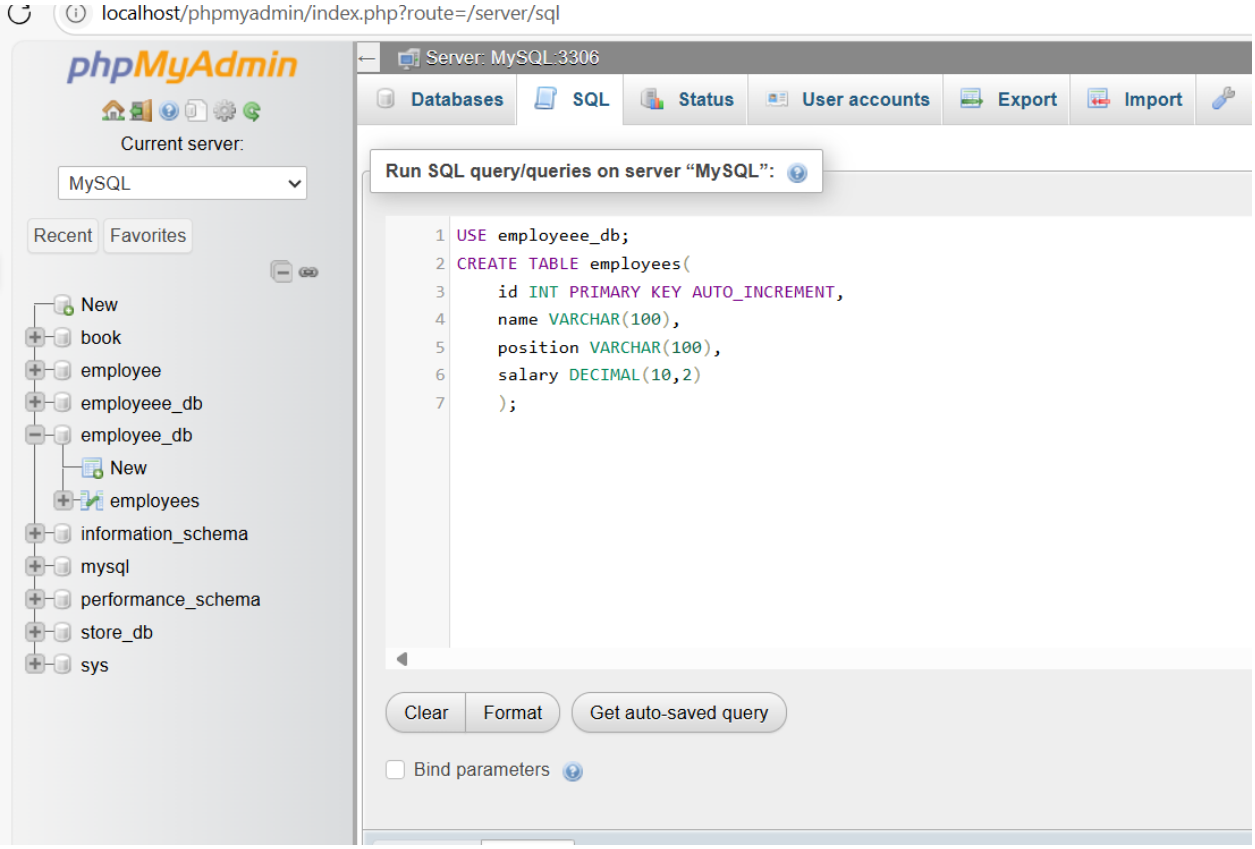
● Install MySQL (if you haven't already) and create a database for the lab.

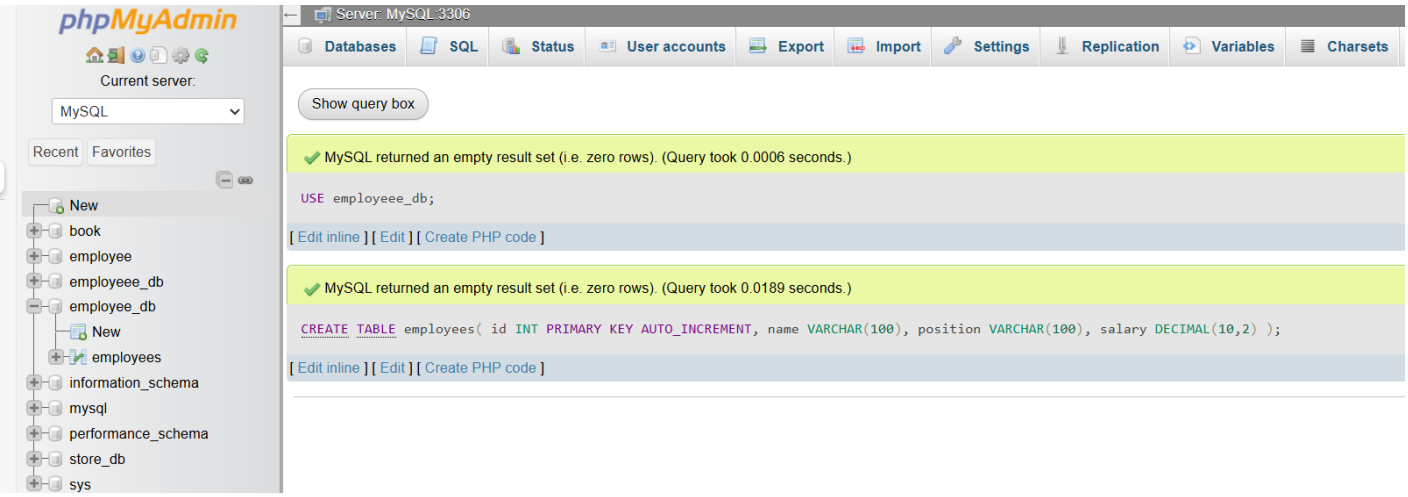
● Open MySQL Workbench or command line and execute the following SQL commands:

```
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)

);

-- Insert some sample data

INSERT INTO employees (name, position, salary) VALUES ('John Doe', 'Software Engineer', 75000);

INSERT INTO employees (name, position, salary) VALUES ('Jane Smith', 'HR Manager', 65000);

INSERT INTO employees (name, position, salary) VALUES ('Steve Brown', 'Team Lead', 85000);
```

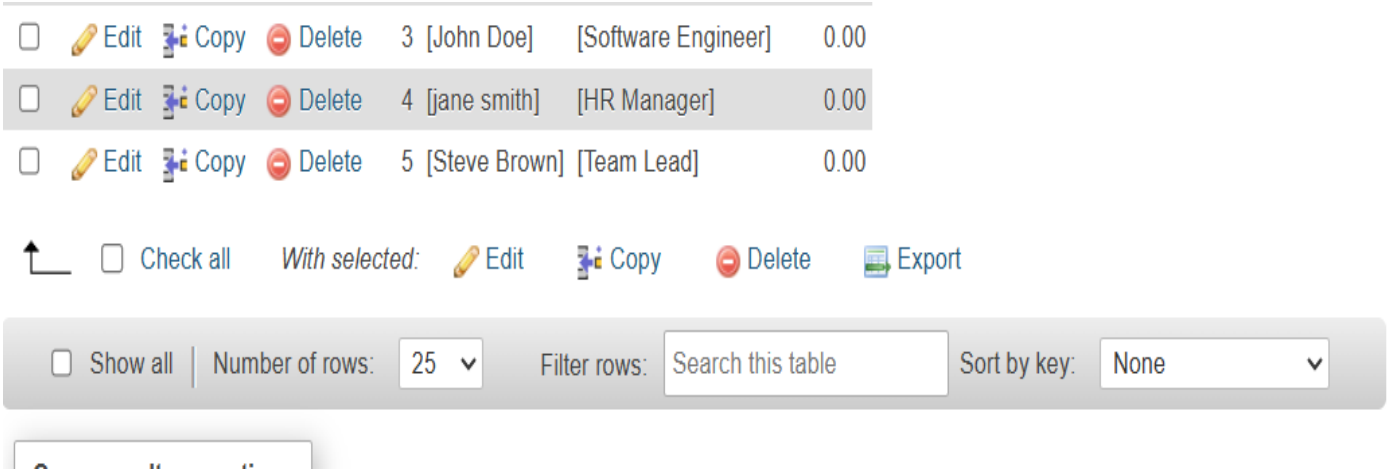

localhost/phpmyadmin/index.php?route=/server/sql
phpMyAdmin
Current server:
MySQL
Recent
Favorites
New
book
employee
employee_db
Server: MySQL:3306
Databases
SQL
Status
User accounts
Export
Impo
Run SQL query/queries on server "MySQL":
1 CREATE DATABASE employeee_db;


phpMyAdmin
Current server:
MySQL
Recent
Favorites
New
book
employee
employeee_db
employee_db
New
employees
Server: MySQL:3306
Databases
SQL
Status
User accounts
Export
Import
Show query box
MySQL returned an empty result set (i.e. zero rows). (Query took 0.0115 seconds.)
CREATE DATABASE employeee_db;
[Edit inline] [Edit] [Create PHP code]


localhost/phpmyadmin/index.php?route=/server/sql
phpMyAdmin
Current server:
MySQL
Recent
Favorites
New
book
employee
employeee_db
employee_db
New
employees
information_schema
mysql
performance_schema
store_db
sys
Server: MySQL:3306
Databases
SQL
Status
User accounts
Export
Import
Run SQL query/queries on server "MySQL":
USE employeee_db;
CREATE TABLE employees(
id INT PRIMARY KEY AUTO_INCREMENT,
name VARCHAR(100),
position VARCHAR(100),
salary DECIMAL(10,2)
);
Clear
Format
Get auto-saved query
Bind parameters


phpMyAdmin
Current server:
MySQL
Recent
Favorites
New
book
employee
employeee_db
employee_db
New
employees
information_schema
mysql
performance_schema
store_db
sys
Server: MySQL:3306
Databases
SQL
Status
User accounts
Export
Import
Settings
Replication
Variables
Charsets
Show query box
MySQL returned an empty result set (i.e. zero rows). (Query took 0.0006 seconds.)
USE employeee_db;
[Edit inline] [Edit] [Create PHP code]
MySQL returned an empty result set (i.e. zero rows). (Query took 0.0189 seconds.)
CREATE TABLE employees(id INT PRIMARY KEY AUTO_INCREMENT, name VARCHAR(100), position VARCHAR(100), salary DECIMAL(10,2));
[Edit inline] [Edit] [Create PHP code]

```
INSERT INTO `employees`( `name`, `position`, `salary`) VALUES ('[John Doe]','[Software Engineer]','[75000]');
INSERT INTO `employees`( `name`, `position`, `salary`) VALUES ('[jane smith]','[HR Manager]','[65000]');
INSERT INTO `employees`( `name`, `position`, `salary`) VALUES ('[Steve Brown]','[Team Lead]','[85000]');
```



2. Set Up NetBeans Project

1. Open NetBeans IDE 8.2.

2. Create a new Java application:

o Go to File > New Project.

o Select Java as the project type, and choose Java Application.

o Name your project JDBCExample.

3. Add MySQL JDBC Driver to your project:

o Right-click on the project in the Projects pane.

o Select Properties.

o In the Libraries tab, click Add JAR/Folder.

o Navigate to the location of your mysql-connector-java-x.x.xx.jar file and add it.

3. Establish JDBC Connection

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

Code for DatabaseConnection.java:

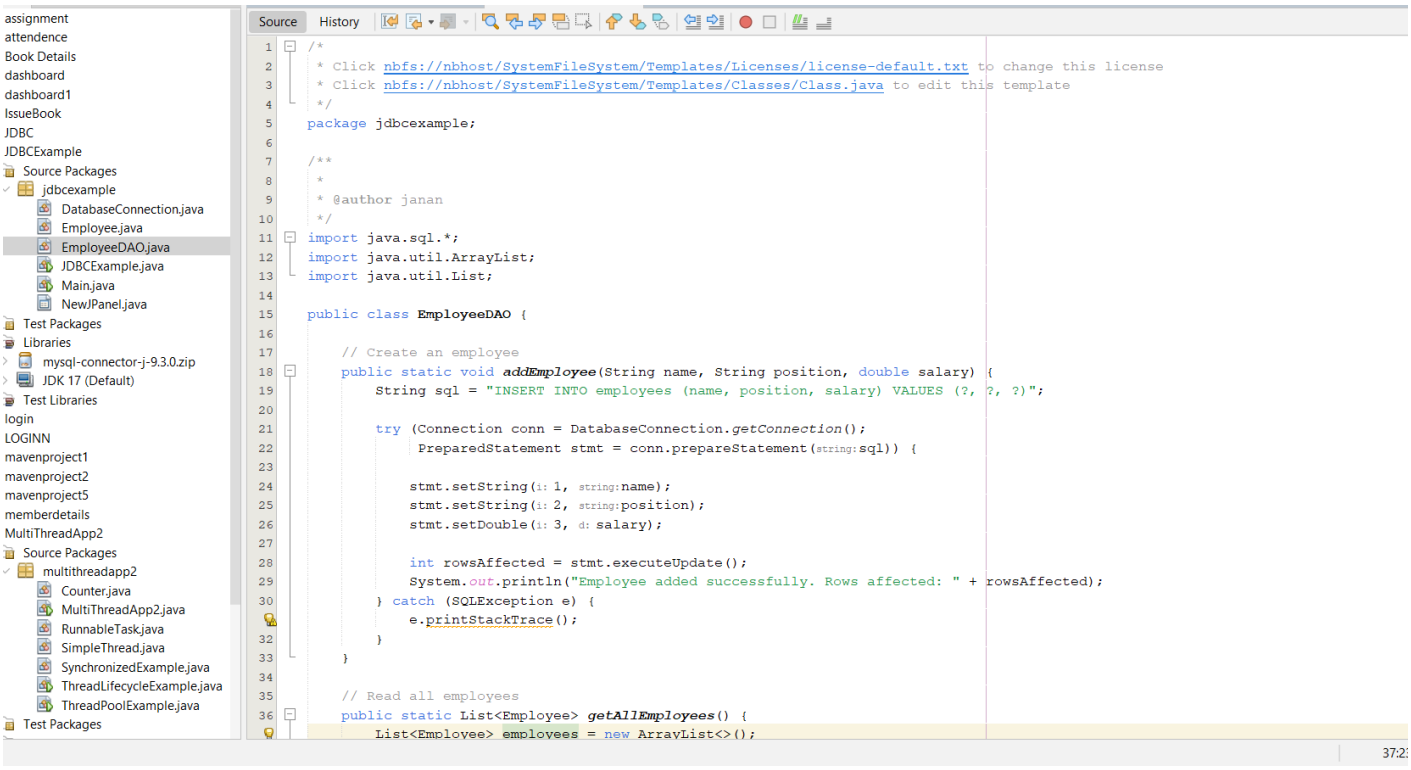
```
package jdbcexample;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DatabaseConnection {
    private static final String URL = "jdbc:mysql://localhost:3306/employeee_db"; // Database URL
```

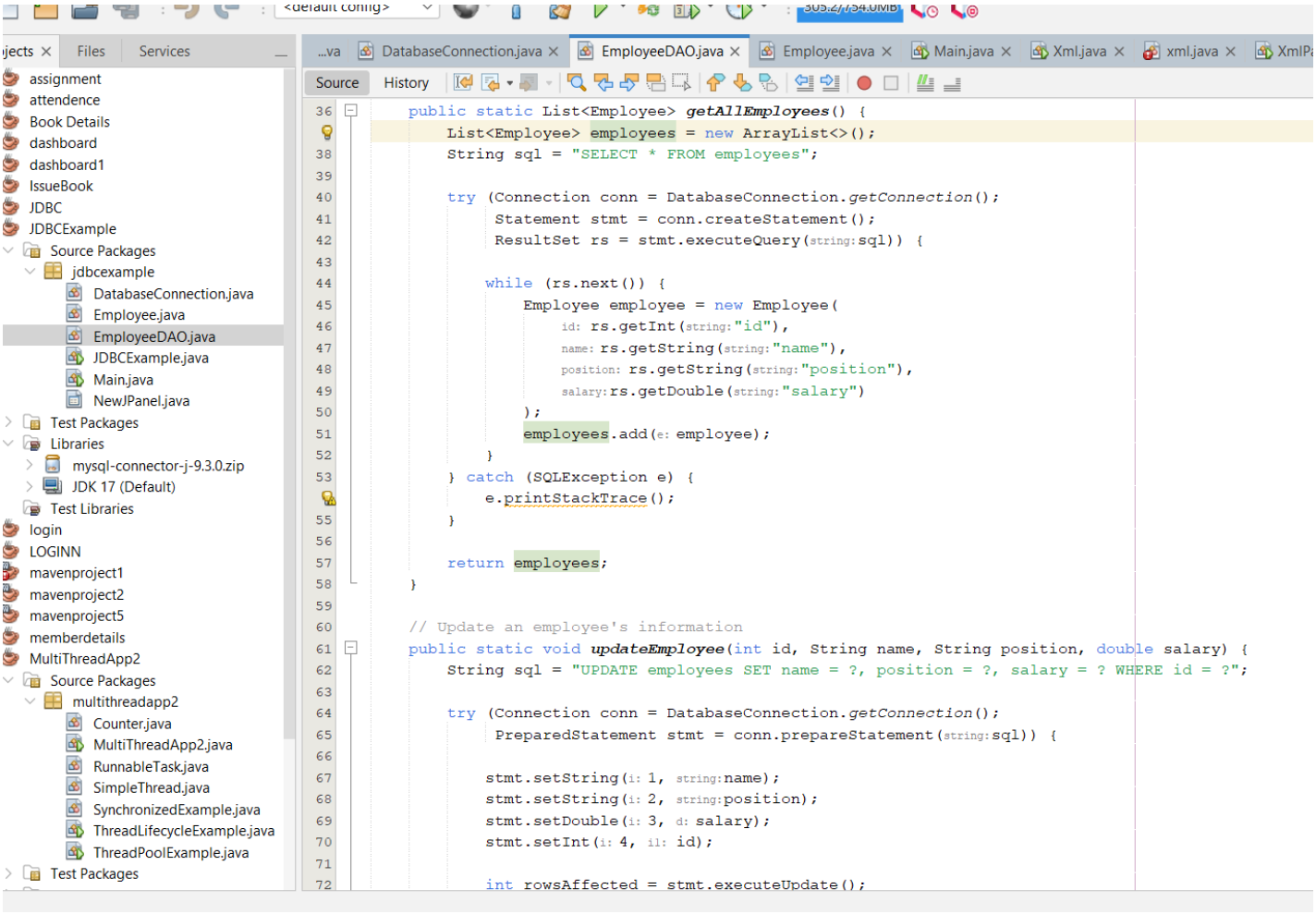
```
    private static final String USER = "root"; // Your MySQL username
    private static final String PASSWORD = ""; // Your MySQL password

    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.");
        }
    }

}
```

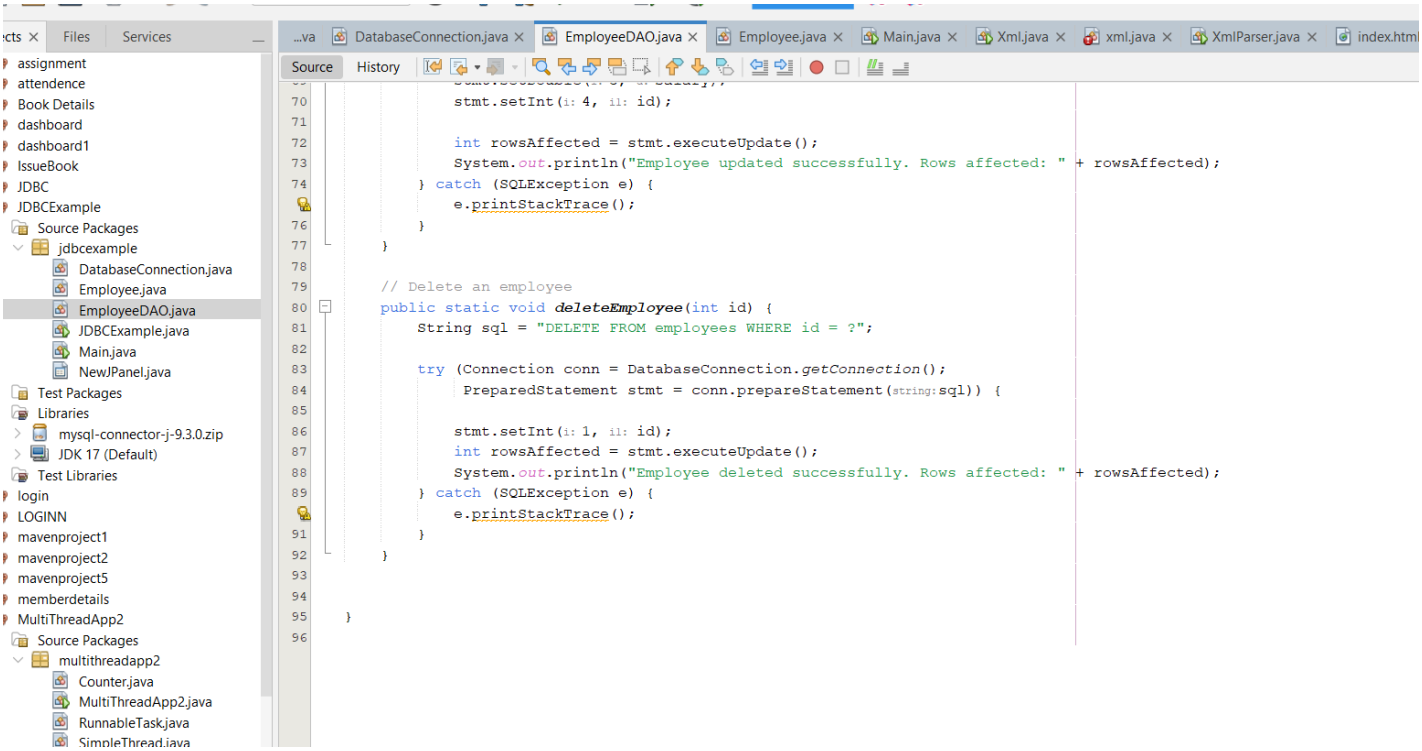
4. Perform CRUD Operations

Next, we will create a class called EmployeeDAO.java that contains methods for performing CRUD operations.

1. Create EmployeeDAO.java for CRUD Operations:


assignment
attendence
Book Details
dashboard
dashboard1
IssueBook
JDBC
JDBCExample
Source Packages
jdbcexample
DatabaseConnection.java
Employee.java
EmployeeDAO.java
JDBCExample.java
Main.java
NewJPanel.java
Test Packages
Libraries
mysql-connector-j-9.3.0.zip
JDK 17 (Default)
Test Libraries
login
LOGINN
mavenproject1
mavenproject2
mavenproject5
memberdetails
MultiThreadApp2
Source Packages
multithreadapp2
Counter.java
MultiThreadApp2.java
RunnableTask.java
SimpleThread.java
SynchronizedExample.java
ThreadLifecycleExample.java
ThreadPoolExample.java
Test Packages
Source
History
/*
* 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
*/
package jdbcexample;
/**
*
* @author janan
*/
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(string:sql)) {
stmt.setString(i: 1, string:name);
stmt.setString(i: 2, string:position);
stmt.setDouble(i: 3, d: 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<>();
37:2


Files
Services
assignment
attendence
Book Details
dashboard
dashboard1
IssueBook
JDBC
JDBCExample
Source Packages
jdbcexample
DatabaseConnection.java
Employee.java
EmployeeDAO.java
JDBCExample.java
Main.java
NewJPanel.java
Test Packages
Libraries
mysql-connector-j-9.3.0.zip
JDK 17 (Default)
Test Libraries
login
LOGINN
mavenproject1
mavenproject2
mavenproject5
memberdetails
MultiThreadApp2
Source Packages
multithreadapp2
Counter.java
MultiThreadApp2.java
RunnableTask.java
SimpleThread.java
SynchronizedExample.java
ThreadLifecycleExample.java
ThreadPoolExample.java
Test Packages
DatabaseConnection.java
EmployeeDAO.java
Employee.java
Main.java
Xml.java
xml.java
Source
History
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, i1: id);
int rowsAffected = stmt.executeUpdate();



5. Create Employee.java Class

Create a simple Employee.java POJO (Plain Old Java Object) to represent employee data.
Code for Employee.java:

```
 *
 * @author janan
 */
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 + '}';
    }
}
```

6.Test the Application

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

Code for Main.java:

```java
/*
 * 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
 */
package jdbcexample;
import java.util.List;

public class Main {

    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);
    }
}
```