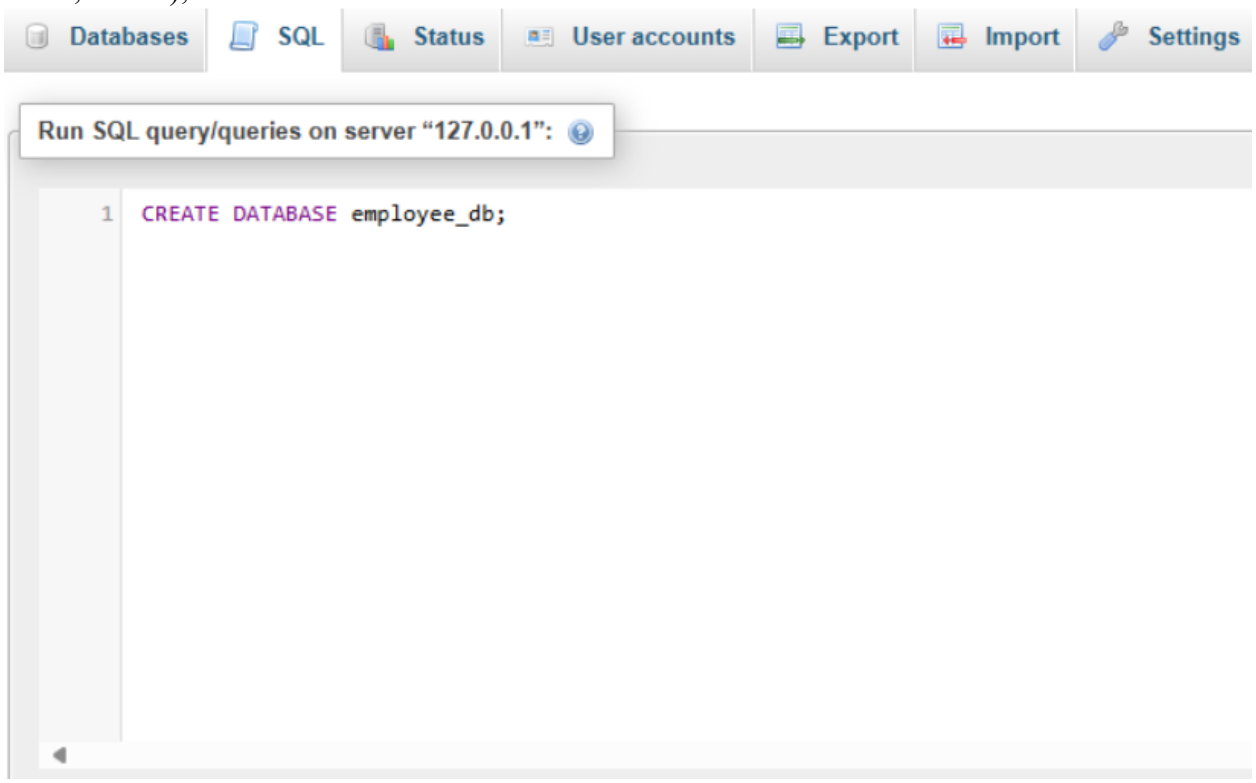
GAM/IT/2022/F/0084 – R.D.S. NISANALA

ASSIGNMENT – JAVA JDBC

1. **Set Up MySQL Database**

```
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);
```



Server: 127.0.0.1 » Database: employee_db

Structure | SQL | Search | Query | Export | Import | Operations | Pr

Run SQL query/queries on database employee_db:

```sql
USE employee_db;
CREATE TABLE employees (
    id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100),
    position VARCHAR(100),
    salary DECIMAL(10, 2)
);
```

Browse | Structure | SQL | Search | Insert | Export | Import | Privileges

Showing rows 0 - 2 (3 total, Query took 0.0003 seconds.)

```sql
SELECT * FROM `employees`
```

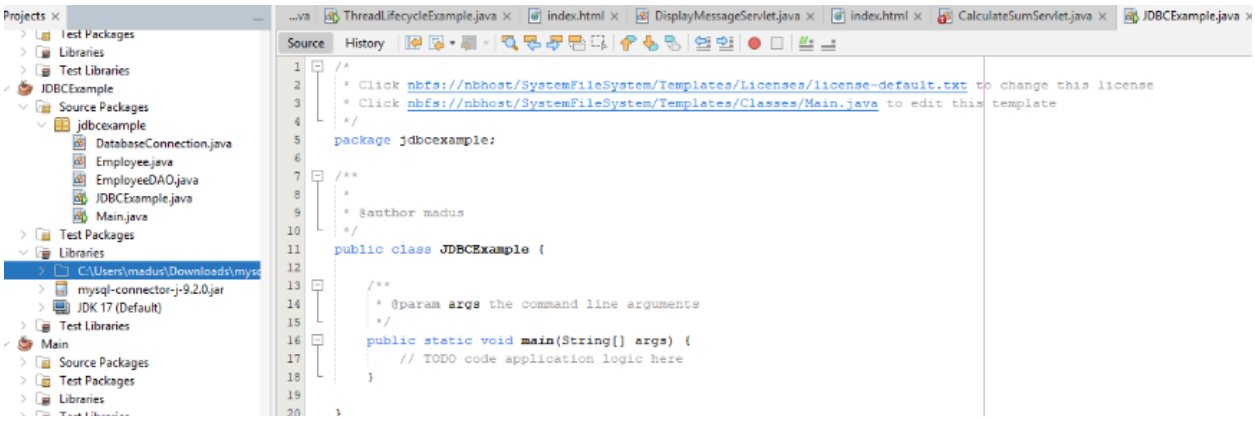
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

←T→				id	name	position	salary
Edit	Copy	Delete		1	John Doe	Software Engineer	75000.00
Edit	Copy	Delete		2	Jane Smith	HR Manager	65000.00
Edit	Copy	Delete		3	Steve Brown	Team Lead	85000.00

2. Set Up NetBeans Project



3. Establish JDBC Connection

```
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 = "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.");
}
}
}
```

```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/employee_db"; // Database URL
    private static final String USER = "root"; // Your MySQL username
    private static final String 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

```java
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();
}
}
}
```

...va | ThreadLifecycleExample.java | index.html | DisplayMessageServlet.java | index.html | CalculateSumServlet.java | JDBCExample.java | DatabaseConnection.java | EmployeeDAO.java

Source | History

```
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(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<>();
        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;
    }
```

```java
    }
    // 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();
        }
    }
}
```

5. **Create Employee.java Class**

```java
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 + '}';
}
}
```

```java
package jdbcexample;


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

```java
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);
}
}
```

```java
package jdbcexample;
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(52, "John Doe", "Developer", 90000);
        // Get all employees
        List<Employee> employees = EmployeeDAO.getAllEmployees();
        employees.forEach(System.out::println);
        // Delete employee
        EmployeeDAO.deleteEmployee(15);
    }
}
```

```
run:
Employee added successfully. Rows affected: 1
Employee added successfully. Rows affected: 1
Employee updated successfully. Rows affected: 1
Employee{id=9, name='John Doe', position='Software Engineer', salary=75000.0}
Employee{id=10, name='Jane Smith', position='HR Manager', salary=65000.0}
Employee{id=11, name='Steve Brown', position='Team Lead', salary=85000.0}
Employee{id=29, name='Bob Marley', position='Manager', salary=80000.0}
Employee{id=32, name='Alice Cooper', position='Developer', salary=70000.0}
Employee{id=42, name='Alice Cooper', position='Developer', salary=70000.0}
Employee{id=48, name='Alice Cooper', position='Developer', salary=70000.0}
Employee{id=49, name='Bob Marley', position='Manager', salary=80000.0}
Employee{id=50, name='Alice Cooper', position='Developer', salary=70000.0}
Employee{id=51, name='Bob Marley', position='Manager', salary=80000.0}
Employee{id=52, name='John Doe', position='Developer', salary=90000.0}
Employee{id=53, name='Bob Marley', position='Manager', salary=80000.0}
Employee{id=54, name='Alice Cooper', position='Developer', salary=70000.0}
Employee{id=55, name='Bob Marley', position='Manager', salary=80000.0}
Employee deleted successfully. Rows affected: 0
BUILD SUCCESSFUL (total time: 0 seconds)
```