Lab – 11: DB_Project_Assignment_8-CRUD Operations using GUI

Lab – 11	DB_Project_Assignment_8-CRUD Operations using GUI
IT615 Database Management System, Autumn'2024; Instructor: minal_bhise@daiict,	

Objectives: I) Setup JDBC & Create Basic GUI

II) Implement CRUD Operations using GUI

Submission: Each student group needs to upload a **single.pdf** file, which will contain the following things for the specific case study assigned to your team.

1) Insert, Update, Delete and Fetch the data from tables of your project using GUI.

Step 1: Create a PostgreSQL Table

Let's assume you have a PostgreSQL table named students with the following structure:

CREATE TABLE students (id SERIAL PRIMARY KEY, name VARCHAR(50), age INTEGER, grade VARCHAR(10));

Step 2: Set Up Your Environment

1. **Ensure JDK and PostgreSQL are Installed**: You need Java Development Kit (JDK) and PostgreSQL installed.
2. **Download JDBC Driver**: Download the PostgreSQL JDBC driver (postgresql-42.7.4.jar) from the **https://jdbc.postgresql.org/download/.**

Step 3: Create the Java Project Structure

Create a New Project Directory:

- Create a directory for your project, e.g., PostgreSQLCRUDApp.

Create Subdirectory for Your Code:

- Inside your project folder, create a src folder for your Java code.

Place the JDBC Driver:

- Put the postgresql-42.7.4.jar file inside your project folder (src).

Step 4: Write the Java Code

Create a Class for Database Operations:

Create a new Java file named DatabaseManager.java in the src directory with the following code:

```
import java.sql.*;

public class DatabaseManager {
    private static final String URL = "jdbc:postgresql://localhost:5432/demo"; // Change to your database name
    private static final String USER = "postgres";  // Change to your username
```

```
private static final String PASSWORD = "prachi";  // Change to your password

// Connect to the database
public Connection connect() throws SQLException {
    return DriverManager.getConnection(URL, USER, PASSWORD);
}

// Insert student
public void insertStudent(String name, int age, String grade) {
    String insertSQL = "INSERT INTO students (name, age, grade) VALUES (?, ?, ?)";
    try (Connection connection = connect();
         PreparedStatement pstmt = connection.prepareStatement(insertSQL)) {
        pstmt.setString(1, name);
        pstmt.setInt(2, age);
        pstmt.setString(3, grade);
        pstmt.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

// Read students
public ResultSet readStudents() {
    String selectSQL = "SELECT * FROM students";
    try {
        Connection connection = connect();
        PreparedStatement pstmt = connection.prepareStatement(selectSQL);
        return pstmt.executeQuery();
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return null;
}

// Update student
public void updateStudent(int id, String name, int age, String grade) {
    String updateSQL = "UPDATE students SET name = ?, age = ?, grade = ? WHERE id = ?";
    try (Connection connection = connect();
         PreparedStatement pstmt = connection.prepareStatement(updateSQL)) {
        pstmt.setString(1, name);
        pstmt.setInt(2, age);
        pstmt.setString(3, grade);
        pstmt.setInt(4, id);
        pstmt.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

// Delete student
public void deleteStudent(int id) {
    String deleteSQL = "DELETE FROM students WHERE id = ?";
```

```
        try (Connection connection = connect();
             PreparedStatement pstmt = connection.prepareStatement(deleteSQL)) {
            pstmt.setInt(1, id);
            pstmt.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

Create the GUI Class

Create another file named StudentGUI.java in the src directory with the following code:

```
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.ResultSet;
import java.sql.SQLException;

public class StudentGUI extends JFrame {
    private JTextField nameField, ageField, gradeField, idField;
    private JButton addButton, updateButton, deleteButton, loadButton;
    private JTable studentTable;
    private DefaultTableModel tableModel;
    private DatabaseManager dbManager;

    public StudentGUI() {
        dbManager = new DatabaseManager();
        setTitle("Student Management");
        setLayout(new BorderLayout());

        // Input panel
        JPanel inputPanel = new JPanel(new GridLayout(5, 2));
        inputPanel.add(new JLabel("ID:"));
        idField = new JTextField();
        inputPanel.add(idField);
        inputPanel.add(new JLabel("Name:"));
        nameField = new JTextField();
        inputPanel.add(nameField);
        inputPanel.add(new JLabel("Age:"));
        ageField = new JTextField();
        inputPanel.add(ageField);
        inputPanel.add(new JLabel("Grade:"));
        gradeField = new JTextField();
        inputPanel.add(gradeField);

        // Button panel
        JPanel buttonPanel = new JPanel();
        addButton = new JButton("Add");
        updateButton = new JButton("Update");
```

```
deleteButton = new JButton("Delete");
loadButton = new JButton("Load");
buttonPanel.add(addButton);
buttonPanel.add(updateButton);
buttonPanel.add(deleteButton);
buttonPanel.add(loadButton);

// Table
tableModel = new DefaultTableModel(new String[]{"ID", "Name", "Age", "Grade"}, 0);
studentTable = new JTable(tableModel);
JScrollPane scrollPane = new JScrollPane(studentTable);

// Add components to the frame
add(inputPanel, BorderLayout.NORTH);
add(scrollPane, BorderLayout.CENTER);
add(buttonPanel, BorderLayout.SOUTH);

// Action listeners
addButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        String name = nameField.getText();
        int age = Integer.parseInt(ageField.getText());
        String grade = gradeField.getText();
        dbManager.insertStudent(name, age, grade);
        loadStudents();
    }
});

updateButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        int id = Integer.parseInt(idField.getText());
        String name = nameField.getText();
        int age = Integer.parseInt(ageField.getText());
        String grade = gradeField.getText();
        dbManager.updateStudent(id, name, age, grade);
        loadStudents();
    }
});

deleteButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        int id = Integer.parseInt(idField.getText());
        dbManager.deleteStudent(id);
        loadStudents();
    }
});

loadButton.addActionListener(new ActionListener() {
    @Override
```

```
            public void actionPerformed(ActionEvent e) {
                loadStudents();
            }
        });

        setSize(600, 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setVisible(true);
    }

    // Load students into the table
    private void loadStudents() {
        try {
            ResultSet resultSet = dbManager.readStudents();
            tableModel.setRowCount(0); // Clear existing data
            while (resultSet != null && resultSet.next()) {
                int id = resultSet.getInt("id");
                String name = resultSet.getString("name");
                int age = resultSet.getInt("age");
                String grade = resultSet.getString("grade");
                tableModel.addRow(new Object[]{id, name, age, grade});
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {
        new StudentGUI();
    }
}
```

Step 5: Compile the Program

1. **Open Command Prompt** or Terminal.
2. **Navigate to Your Project Directory**:

```
cd path_to_your_PostgreSQLCRUDApp/src
```

Compile the Java Files: `javac -cp .;postgresql-42.7.4.jar DatabaseManager.java StudentGUI.java`

Step 5: Run the Program

1. **Run the Program**:

```
java -cp .;postgresql-42.7.4.jar StudentGUI
```

Step 6: Perform CRUD operations using GUI

- **Add Students**: Enter the name, age, and grade, then click the **Add** button.
- **Update Students**: Enter the ID of the student you want to update along with the new details and click the **Update** button.

- **Delete Students**: Enter the ID of the student you want to delete and click the **Delete** button.
- **Load Students**: Click the **Load** button to fetch the list of students from the database.

Submit:

• Snapshots of all the above mentioned operations on at least two tables of your project.