Prac 4) Create Exam Registration Form using JDBC Connectivity

PreparedStatement in Java :

A PreparedStatement is a pre-compiled SQL statement. It is a subinterface of Statement. Prepared Statement objects have some useful additional features compared to Statement objects. PreparedStatement allows you to execute SQL queries with parameters, avoiding the need to hard-code values directly into the query.

- When a PreparedStatement is created, the SQL query is passed as a parameter. This Prepared Statement contains a pre-compiled SQL query, so when the PreparedStatement is executed, the DBMS can just run the query instead of first compiling it.
- The same PreparedStatement can be reused with different parameter values during execution.
- An important advantage of PreparedStatements is that they prevent SQL injection attacks.

Steps to use PreparedStatement :

Step 1: Create a Connection to DB :

```
Connection conn = DriverManager.getConnection(URL, USER, PASS);
```

Step 2: Prepare Statement :

Set parameter placeholders(use question mark for placeholders) like

```
String sql = "INSERT INTO exam_registration (student_name, email,
course, exam_date) VALUES (?, ?, ?, ?)"
```

```
PreparedStatement ps = conn.prepareStatement(sql);
```

Step 3. Set parameter values for type and position :

```
ps.setString(1, name);
 ps.setString(2, email);
 ps.setString(3, course);
 ps.setDate(4, Date.valueOf(date));
 ps.setInt(5, id);
```

Step 4. Execute the Query :

```
ps.executeUpdate();
```

Step 5: Close All Resources :

ps.close();

conn.close();

Methods of PreparedStatement:

- **setInt(int, int):** This method can be used to set integer value at the given parameter index.
- **setString(int, string):** This method can be used to set string value at the given parameter index.
- **setFloat(int, float):** This method can be used to set float value at the given parameter index.
- **setDouble(int, double):** This method can be used to set a double value at the given parameter index.
- **executeUpdate():** This method can be used to create, drop, insert, update, delete etc. It returns int type.
- **executeQuery():** It returns an instance of ResultSet when a select query is executed**.**

Code:

```java
package CRUD_oprations;

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.sql.*;

public class ExamRegistrationCRUD extends JFrame {

    // Database details
    static final String URL = "jdbc:mysql://localhost:3306/studentdb";
    static final String USER = "root";
    static final String PASS = "root";

    // GUI Components
    JTextField idField, nameField, emailField, courseField, dateField;
    JButton registerBtn, viewBtn, updateBtn, deleteBtn;
    JTextArea outputArea;

    public ExamRegistrationCRUD() {
        // Frame settings
        setTitle("Exam Registration CRUD");
        setSize(450, 500);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new FlowLayout());

        // ID field for Update/Delete
        add(new JLabel("Student ID (for Update/Delete):"));
        idField = new JTextField(20);
        add(idField);

        add(new JLabel("Name:"));
        nameField = new JTextField(20);
        add(nameField);

        add(new JLabel("Email:"));
        emailField = new JTextField(20);
        add(emailField);

        add(new JLabel("Course:"));
        courseField = new JTextField(20);
        add(courseField);

        add(new JLabel("Exam Date (YYYY-MM-DD):"));
        dateField = new JTextField(20);
        add(dateField);

        // Buttons
        registerBtn = new JButton("Register");
        updateBtn = new JButton("Update");
        deleteBtn = new JButton("Delete");
        viewBtn = new JButton("View Students");
        add(registerBtn);
        add(updateBtn);
        add(deleteBtn);
        add(viewBtn);

        // Output area
        outputArea = new JTextArea(12, 35);
        outputArea.setEditable(false);
        add(new JScrollPane(outputArea));

        // Button actions
        registerBtn.addActionListener(e -> registerStudent());
        updateBtn.addActionListener(e -> updateStudent());
        deleteBtn.addActionListener(e -> deleteStudent());
        viewBtn.addActionListener(e -> viewStudents());
    }

    // CREATE: Insert student record
    private void registerStudent() {
        String name = nameField.getText();
        String email = emailField.getText();
        String course = courseField.getText();
        String date = dateField.getText();

        String sql = "INSERT INTO exam_registration (student_name, email, course,
exam_date) VALUES (?, ?, ?, ?)";

        try (Connection conn = DriverManager.getConnection(URL, USER, PASS);
             PreparedStatement ps = conn.prepareStatement(sql)) {

            ps.setString(1, name);
            ps.setString(2, email);
            ps.setString(3, course);
            ps.setDate(4, Date.valueOf(date));

            int rows = ps.executeUpdate();
            outputArea.setText(rows > 0 ? "✅ Registration Successful!\n" : "⚠
Registration Failed.\n");
            clearFields();

        } catch (SQLException ex) {
            outputArea.setText("Error: " + ex.getMessage());
        }
    }

    // READ: View all students
    private void viewStudents() {
        String sql = "SELECT * FROM exam_registration";

        try (Connection conn = DriverManager.getConnection(URL, USER, PASS);
             Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery(sql)) {
```

```java
            outputArea.setText("--- Registered Students ---\n");
            while (rs.next()) {
                outputArea.append("ID: " + rs.getInt("reg_id") +
                        ", Name: " + rs.getString("student_name") +
                        ", Email: " + rs.getString("email") +
                        ", Course: " + rs.getString("course") +
                        ", Exam Date: " + rs.getDate("exam_date") + "\n");
            }

        } catch (SQLException ex) {
            outputArea.setText("Error: " + ex.getMessage());
        }
    }

    // UPDATE: Update student details
    private void updateStudent() {
        int id = Integer.parseInt(idField.getText());
        String name = nameField.getText();
        String email = emailField.getText();
        String course = courseField.getText();
        String date = dateField.getText();

        String sql = "UPDATE exam_registration SET student_name=?, email=?, course=?,
exam_date=? WHERE reg_id=?";

        try (Connection conn = DriverManager.getConnection(URL, USER, PASS);
             PreparedStatement ps = conn.prepareStatement(sql)) {

            ps.setString(1, name);
            ps.setString(2, email);
            ps.setString(3, course);
            ps.setDate(4, Date.valueOf(date));
            ps.setInt(5, id);

            int rows = ps.executeUpdate();
            outputArea.setText(rows > 0 ? "✅ Student Updated.\n" : "⚠️ Student Not
Found.\n");
            clearFields();

        } catch (SQLException ex) {
            outputArea.setText("Error: " + ex.getMessage());
        }
    }

    // DELETE: Delete student record
    private void deleteStudent() {
        int id = Integer.parseInt(idField.getText());

        String sql = "DELETE FROM exam_registration WHERE reg_id=?";

        try (Connection conn = DriverManager.getConnection(URL, USER, PASS);
             PreparedStatement ps = conn.prepareStatement(sql)) {

            ps.setInt(1, id);
            int rows = ps.executeUpdate();
            outputArea.setText(rows > 0 ? "✅ Student Deleted.\n" : "⚠ Student Not Found.\n");
            clearFields();

        } catch (SQLException ex) {
            outputArea.setText("Error: " + ex.getMessage());
        }
    }

    // Clear text fields
    private void clearFields() {
        idField.setText("");
        nameField.setText("");
        emailField.setText("");
        courseField.setText("");
        dateField.setText("");
    }

    // Main method
    public static void main(String[] args) {
        new ExamRegistrationCRUD().setVisible(true);
    }
}
```

SQL :

CREATE TABLE exam_registration (reg_id INT PRIMARY KEY AUTO_INCREMENT, student_name VARCHAR(100), email VARCHAR(100), course VARCHAR(50), exam_date DATE);