Experiment No. 8

Title: Implementation of CRUD operation.

Batch:SY-IT(B3) **Roll No.:16010423076** **Experiment No.:8**

Aim: Create a form using Swing form to accept the user input and store it in database. Perform and demonstrate CRUD operation on the button click.

Resources needed: Java Development kit, Netbeans, Mysql Server(XAMPP)

Theory:

Swing API is set of extensible GUI Components to ease developer's life to create JAVA based Front End/ GUI Applications. It is built upon top of AWT API and acts as replacement of AWT API as it has almost every control corresponding to AWT controls. It has more powerful and flexible components than AWT. Swing component follows a Model-View-Controller architecture. Java Swing is Light Weight; Rich in controls and Highly Customizable. Swing controls inherit properties from following Component class hierarchy. In addition to familiar components such as buttons, check boxes and labels, Swing provides several advanced components such as tabbed panel, scroll panes, trees, tables, and lists.

No.	Class	Description
1	**Component**	A Container is the abstract base class for the non menu user-interface controls of SWING. Component represents an object with graphical representation
2	**Container**	A Container is a component that can contain other SWING components.
3	**JComponent**	A JComponent is a base class for all swing UI components. In order to use a swing component that inherits from JComponent, component must be in a containment hierarchy whose root is a top-level Swing container.

Following is the list of commonly used controls while designed GUI using SWING.

No.	Control	Description
1	**JLabel**	A JLabel object is a component for placing text in a container.
2	**JButton**	This class creates a labeled button.

3	**JColorChooser**	A JColorChooser provides a pane of controls designed to allow a user to manipulate and select a color.
4	**JCheck Box**	A JCheckBox is a graphical component that can be in either an **on**(true) or **off** (false) state.
5	**JRadioButton**	The JRadioButton class is a graphical component that can be in either an **on** (true) or **off** (false) state. in a group.
6	**JList**	A JList component presents the user with a scrolling list of text items.
7	**JComboBox**	A JComboBox component presents the user with a to show up menu of choices.
8	**JTextField**	A JTextField object is a text component that allows for the editing of a single line of text.
9	**JPasswordField**	A JPasswordField object is a text component specialized for password entry.
10	**JTextArea**	A JTextArea object is a text component that allows for the editing of a multiple lines of text.
11	**ImageIcon**	A ImageIcon control is an implementation of the Icon interface that paints Icons from Images
12	**JScrollbar**	A Scrollbar control represents a scroll bar component in order to enable user to select from range of values.
13	**JOptionPane**	JOptionPane provides set of standard dialog boxes that prompt users for a value or informs them of something.
14	**JFileChooser**	A JFileChooser control represents a dialog window from which the user can select a file.
15	**JProgressBar**	As the task progresses towards completion, the progress bar displays the task's percentage of completion.
16	**JSlider**	A JSlider lets the user graphically select a value by sliding a knob within a bounded interval.
17	**JSpinner**	A JSpinner is a single line input field that lets the user select a number or an object value from an ordered sequence.

java.awt.GridLayout Package: The GridLayout class is a **layout manager** that lays out a container's components in a rectangular grid. The container is divided into equal-sized rectangles, and one component is placed in each rectangle, like rows and columns.

Event Handling

Any program that uses GUI (graphical user interface) such as Java application written for windows, is event driven. Changing the state of an object is known as an event. For example, click on button, dragging mouse etc. Event Handling is the mechanism that controls the event and decides what should happen if an event occurs. Event handling has three main components.

- **Events :** An event is a change of state of an object.
- **Events Source :** Event source is an object that generates an event. Source is responsible for providing information of the occurred event to it's handler.
- **Listeners :** A listener is an object that listens to the event. A listener gets notified when an event occurs.**Listener** also known as event handler is responsible for generating response to an event. Listener waits until it receives an event. The java.awt.event package provides many event classes and Listener interfaces for event handling. In this package we use **ActionListener Interface**. The listener interface is for receiving action events. The class that is interested in processing an action event implements this interface, and the object created with that class is registered with a component, using the component's**addActionListener** method. When the action event occurs, that object's **actionPerformed** method is invoked.Following steps are required to perform event handling:

  1. Implement the Listener interface and overrides its methods
  2. Register the component with the Listener

For registering the component with the Listener, many classes provide the registration methods. For example:

Button: public void addActionListener(ActionListener a){}

We can put the event handling code in Same class or Other class or Annonymous class.

Important Event Classes and Interfaces

Event Classes	Description	Listener Interface
ActionEvent	generated when button is pressed, menu-item is selected, list-item is double clicked	ActionListener
MouseEvent	generated when mouse is dragged, moved,clicked,pressed or released also when the enters or exit a component	MouseListener
KeyEvent	generated when input is received from keyboard	KeyListener
ItemEvent	generated when check-box or list item is clicked	ItemListener

TextEvent	generated when value of textarea or textfield is changed	TextListener
MouseWheelEvent	generated when mouse wheel is moved	MouseWheelListener
WindowEvent	generated when window is activated, deactivated, deiconified, iconified, opened or closed	WindowListener
ComponentEvent	generated when component is hidden, moved, resized or set visible	ComponentEventListener
ContainerEvent	generated when component is added or removed from container	ContainerListener
AdjustmentEvent	generated when scroll bar is manipulated	AdjustmentListener
FocusEvent	generated when component gains or loses keyboard focus	FocusListener

Connecting to the Database

Input obtained from GUI can be stored in database using the database connectivity code provided. Validation process is included before storing data into database. Java JDBC is a java API to connect and execute query with the database. JDBC API uses jdbc drivers to connect with the database. There are 6 steps to connect any java application with the database in java using JDBC. They are as follows:

1. **Import the packages:** Requires that you include the packages containing the JDBC classes needed for database programming. These 3 import statements should be present.
    - **import java.sql.Connection;**
    - **import java.sql.DriverManager;**
    - **import java.sql.SQLException;**
2. **Register the driver class:** The forName() method of Class class is used to register the driver class. This method is used to dynamically load the driver class. Eg: Class.forName("com.mysql.jdbc.Driver");
3. **Creating connection:** To connect to a database you need a Connection object. The Connection object uses a **DriverManager**. The DriverManager passes in your database username, your password, and the location of the database. The getConnection() method of DriverManager class is used to establish connection with the database. For the mysql database the driver class is **com.mysql.jdbc.Driver**., connection is **jdbc:mysql://localhost:3306/sonoo** where jdbc is the API, mysql is the database, localhost is the server name on which mysql is running, we may also use IP address, 3306 is the port number and sonoo is the database name. The default username for the mysql database is **root**. Password is given by the user at the time of installing the mysql

database.
Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/pl2db","root","root");

4. **Creating statement:** The createStatement() method of Connection interface is used to create statement. The object of statement is responsible. The JDBC*Statement, CallableStatement,* and *PreparedStatement* interfaces define the methods and properties that enable you to send SQL or PL/SQL commands and receive data from your database.They also define methods that help bridge data type differences between Java and SQL data types used in a database.
5. **Executing queries:** The executeQuery() method of Statement interface is used to execute queries to the database. This method returns the object of ResultSet that can be used to get all the records of a table.

   Eg: ResultSet rs=stmt.executeQuery("select * from emp");

   while(rs.next()){

   System.out.println(rs.getInt(1)+" "+rs.getString(2));

   }

6. **Closing connection:** By closing connection object statement and ResultSet will be closed automatically. The close() method of Connection interface is used to close the connection. Eg: con.close();

Running the Application/Form

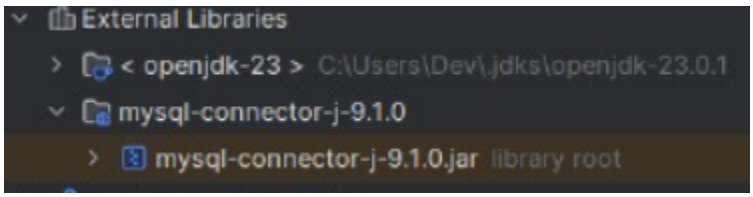
For running the application following files are required:

- open project then right click open new -> jFrame-> name your Frame and untick the main class.
- Right click on project and select properties add mysql-connector.bin jar file to the library.
- Design your application page by Drag and drop components from Palette
- open xampp control panel and Start apache and mysql
- Type **localhost/phpmyadmin** in browser.
- Click on Database tab ,Name your database and click on create
- Database will be listed then click on that to create table
- Name your table add no. of columns required for your table. (can be modified later)—click on GO
- Add attributes as per application requirement.Save this table
- Table will be created inside your database

Results: (Code with Output)

GUI Code (Java) :

Java MySQL JDBC in intelliJ:

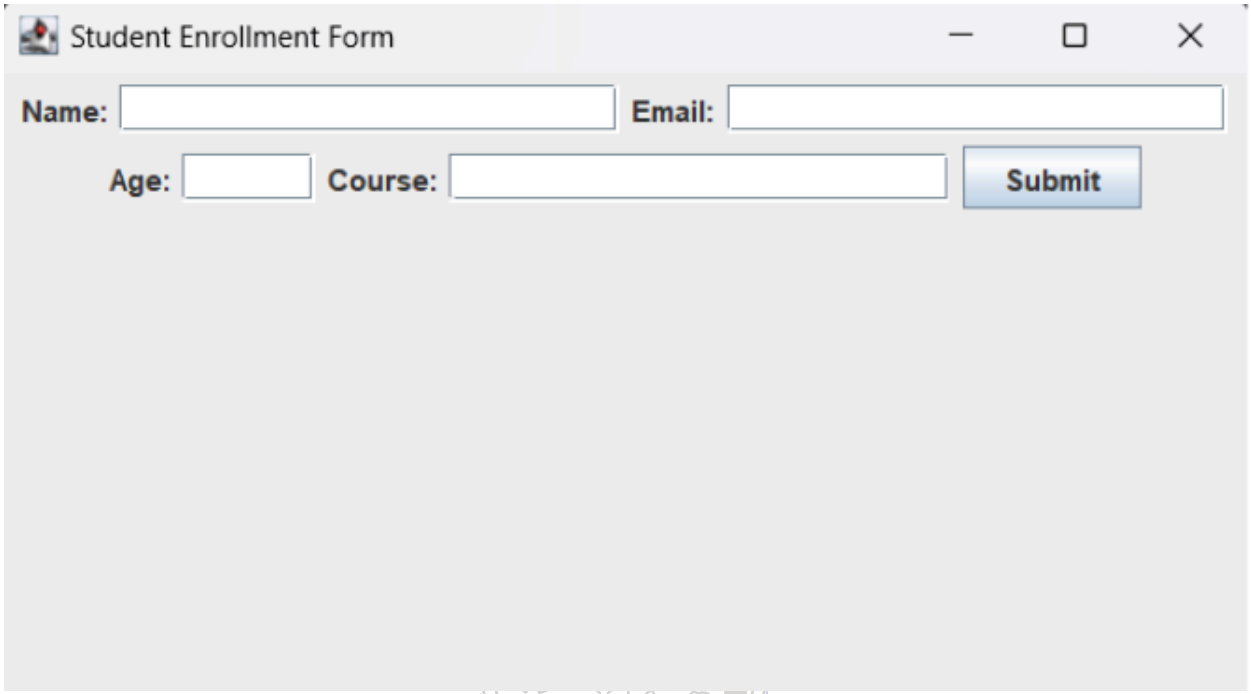


Form.java

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

public class Form extends JFrame {
    private JTextField nameField, emailField, ageField, courseField;
    private JButton submitButton;

    public Form(String role) {
        setTitle(role + " Enrollment Form");
        setSize(500, 300);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new FlowLayout());

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

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

        JLabel ageLabel = new JLabel("Age:");
        ageField = new JTextField(5);

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

        submitButton = new JButton("Submit");
```

```
        add(nameLabel);
        add(nameField);
        add(emailLabel);
        add(emailField);
        add(ageLabel);
        add(ageField);
        add(courseLabel);
        add(courseField);
        add(submitButton);

        submitButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                String name = nameField.getText();
                String email = emailField.getText();
                String age = ageField.getText();
                String course = courseField.getText();

                saveData(name, email, age, course);  // Save student data to database
            }
        });
    }

    private void saveData(String name, String email, String age, String course) {
        String url = "jdbc:mysql://localhost:3306/college_system";
        String user = "root";
        String password = "";

        try {
            Connection con = DriverManager.getConnection(url, user, password);

            String query = "INSERT INTO students (name, email, age, course) VALUES (?, ?, ?, ?)";
            PreparedStatement pst = con.prepareStatement(query);
            pst.setString(1, name);
            pst.setString(2, email);
            pst.setInt(3, Integer.parseInt(age));
            pst.setString(4, course);

            pst.executeUpdate();

            JOptionPane.showMessageDialog(this, "Student enrolled successfully!");

            con.close();
        } catch (SQLException ex) {
            JOptionPane.showMessageDialog(this, "Error: " + ex.getMessage());
```

```
        }
    }

    public static void main(String[] args) {
        new Form("Student").setVisible(true);
    }
}
```

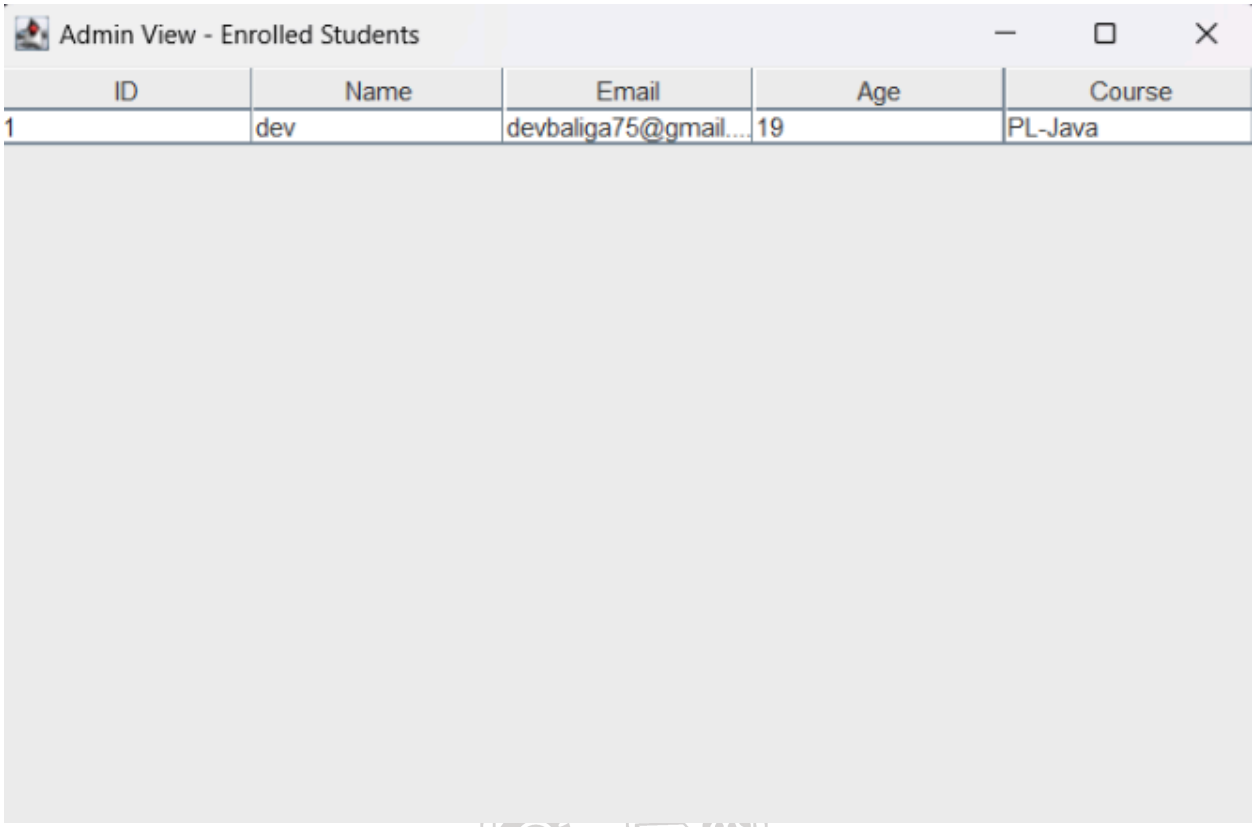


AdminView.java

```
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.sql.*;
import java.util.Vector;

public class AdminView extends JFrame {

    private JTable studentTable;
    private DefaultTableModel tableModel;

    public AdminView() {
        // Setup the frame
        setTitle("Admin View - Enrolled Students");
        setSize(600, 400);
```

```
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new BorderLayout());

        tableModel = new DefaultTableModel(new String[]{"ID", "Name", "Email", "Age", "Course"},
0);
        studentTable = new JTable(tableModel);
        JScrollPane tableScrollPane = new JScrollPane(studentTable);

        add(tableScrollPane, BorderLayout.CENTER);
        viewStudents();
    }

    private void viewStudents() {
        String url = "jdbc:mysql://localhost:3306/college_system";
        String user = "root";
        String password = "";

        try {
            Connection con = DriverManager.getConnection(url, user, password);

            String query = "SELECT * FROM students";
            Statement stmt = con.createStatement();
            ResultSet rs = stmt.executeQuery(query);

            tableModel.setRowCount(0);

            while (rs.next()) {
                Vector<String> row = new Vector<>();
                row.add(String.valueOf(rs.getInt("id")));
                row.add(rs.getString("name"));
                row.add(rs.getString("email"));
                row.add(String.valueOf(rs.getInt("age")));
                row.add(rs.getString("course"));
                tableModel.addRow(row);
            }

            con.close();
        } catch (SQLException ex) {
            JOptionPane.showMessageDialog(this, "Error: " + ex.getMessage());
        }
    }
    public static void main(String[] args) {
        new AdminView().setVisible(true);
    }
```

}



Backend Code(MySQL) :

```
CREATE DATABASE college_system;

USE college_system;

-- Table for storing student data
CREATE TABLE students (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    email VARCHAR(255) NOT NULL,
    age INT NOT NULL,
    course VARCHAR(255) NOT NULL
);

-- Table for storing user roles
CREATE TABLE roles (
    id INT AUTO_INCREMENT PRIMARY KEY,
    role_type VARCHAR(255) NOT NULL,
```

```
   user_id INT NOT NULL
);
```

Questions:

1. Explain event handling in java.

Event handling in Java is a mechanism that responds to user actions like clicks, key presses, or mouse movements within a graphical user interface (GUI). It follows the **event-driven programming model**, where an event is generated by an action and handled by a listener. Java uses interfaces like ActionListener, MouseListener, and KeyListener to define event handlers, while classes like ActionEvent or MouseEvent represent the events. The process involves associating a component (e.g., button) with an event listener, where the listener's method is called automatically when the event occurs. This allows Java applications to react dynamically to user interactions, improving interactivity and usability.

Outcomes: CO4: Illustrate the use of collection classes ,functional programming and GUI programming with java

Conclusion: (Conclusion to be based on the objectives and outcomes achieved)

PAGE No.
DATE / /

Ritesh S. Jha
16010423076
7/10/24

I learned how to connect a Java application in intelliJ to a XAMPP MySQL database using JDBC. I created a database, a table, inserting data into it and retrieving it through code. I also setup the MySQL connector as a library and integrated it into the Java project. Finally retrieving the data from SQL tables onto the Java Terminal.

Ritesh Jha
16010423076
21/10/24

Conclusion :
Today, I understood the rubriks for Java miniproject. Also, learnt more advanced concepts of GUI such as JTable. Understood the creation & usage of CRUD Operations in Java. CRUD - Create, Read, update & Delete operations using GUI, JDBC & MySQL.

21/10/24

Grade: AA / AB / BB / BC / CC / CD/DD

Signature of faculty in-charge with date

References:

1. Herbert Schildt ; ” Java The Complete Reference”; 10th Edition 2017; McGraw-Hill Publication
2. D.T. Editorial Services; “Java 8 Programming Black Book” ; Edition 2015 ; Dreamtech Press2004