**CORE JAVA MINI PROJECT**

**Name:** Aryan Gawde

**Roll No:** A008

**SAP ID:** 45207220012

**Synopsis:**

This Java code implements a simple Contact Management System using Swing for the GUI components and JDBC for database connectivity. Here's a synopsis of the code:

1. **Import Statements**: The code imports necessary Java libraries for AWT, Swing, SQL, and regular expressions.
2. **Class Declaration**: The main class is named **ContactManagementSystem**, which extends **Frame** indicating it's a GUI application.
3. **Static Variables**: The class contains various static variables for database connection details, Swing components, and regular expression patterns.
4. **Constructor**: The constructor initializes the GUI components and sets up the layout using **GridBagLayout**.
5. **Main Method**: The **main** method creates an instance of **ContactManagementSystem** and makes the GUI visible.
6. **Database Connection Methods**:
   * **connect()**: Establishes a connection to the MySQL database using JDBC.
   * **close()**: Closes the database connection, prepared statement, and result set.
7. **Helper Methods**:
   * **addLabelAndField()**: Adds labels and corresponding text fields to the GUI.
   * **isValidEmail()**: Validates email addresses using a regular expression.
   * **isValidPhoneNumber()**: Validates phone numbers using a regular expression.
   * **clearFields()**: Clears all text fields in the GUI.
8. **CRUD Operations**:
   * **addContact()**: Inserts a new contact into the database.
   * **viewContacts()**: Retrieves and displays all contacts from the database.
   * **searchContact()**: Searches for contacts based on user-selected criteria.
   * **editContact()**: Edits existing contact details based on user-selected criteria.
   * **deleteContact()**: Deletes contacts based on user-selected criteria.
9. **Event Listeners**:
   * Action listeners are added to buttons for performing CRUD operations.
   * Each button triggers a corresponding method to perform the desired operation.
10. **Error Handling**:
    * SQLExceptions are caught and printed for database-related errors.
    * JOptionPane is used to display error messages to the user.

Overall, this code provides a basic framework for managing contacts with functionalities such as adding, viewing, searching, editing, and deleting contacts stored in a MySQL database.

**Objective and Scope:**

The objective of the provided code is to implement a simple Contact Management System using Java Swing for the graphical user interface (GUI) and JDBC for database connectivity. The system aims to provide basic functionalities for managing contacts, including adding, viewing, searching, editing, and deleting contact records stored in a MySQL database.

Overall, the scope of the Contact Management System is to offer essential functionalities for managing contact information efficiently, providing users with a convenient tool for organizing and accessing their contacts.

Top of Form

**Merits of the Contact Management System:**

1. **Functionalities**: The code provides essential functionalities for managing contacts, including adding, viewing, searching, editing, and deleting contacts, fulfilling the basic requirements of a contact management system.
2. **User Interface**: The use of Java Swing components offers a simple and intuitive graphical user interface, making it easy for users to interact with the system.
3. **Database Connectivity**: JDBC is used to establish a connection to a MySQL database, enabling the storage and retrieval of contact information, ensuring data persistence.
4. **Error Handling**: The code includes error handling mechanisms to catch and handle SQL exceptions, providing informative error messages to users, which enhances the robustness and reliability of the system.
5. **Validation**: Basic validation is implemented for email addresses and phone numbers, ensuring data integrity and accuracy by preventing the insertion of invalid contact details.

**Demerits of the Contact Management System:**

1. **Single-Threaded GUI**: The GUI operates on a single thread, which may lead to unresponsiveness or freezing of the interface when performing database operations, especially for large datasets or slow database connections.
2. **Static Variables and Methods**: The heavy use of static variables and methods may lead to issues with maintainability and testability, making it difficult to manage the state of the application and perform unit testing effectively.
3. **Limited Scalability**: The code is designed for managing contacts in a single MySQL database. Scaling the system to handle multiple users or larger datasets may require significant refactoring and optimization.
4. **Limited Error Reporting**: While the code handles SQL exceptions, it may not provide detailed error messages or logging capabilities, making it challenging to diagnose and troubleshoot issues, especially in production environments.
5. **Limited User Feedback**: The GUI does not provide real-time feedback or progress indicators during long-running database operations, potentially leaving users unaware of the system's status or progress.
6. **UI Design Limitations**: The GUI layout is based on AWT's GridBagLayout, which may not offer the flexibility or aesthetics of more modern GUI frameworks or design patterns.

**Conclusion:**

In conclusion, the provided code offers a functional foundation for a Contact Management System implemented in Java. It provides essential features such as adding, viewing, searching, editing, and deleting contacts stored in a MySQL database, all within a simple graphical user interface built using Java Swing components. Despite its merits in providing basic functionalities and database connectivity, the code also exhibits several limitations and areas for improvement.

The Contact Management System can become more robust, secure, scalable, and user-friendly, meeting the needs of users for managing their contact information efficiently and effectively. Additionally, continued maintenance and iterative improvements will be necessary to keep the system up-to-date and aligned with evolving requirements and best practices in software development.

**References:** Google, W3schools

**Source Code:**

import java.awt.\*;

import java.awt.event.\*;

import java.sql.\*;

import java.util.regex.\*;

import javax.swing.JOptionPane;

public class ContactManagementSystem extends Frame {

    private static final String JDBC\_URL = "jdbc:mysql://localhost:3307/contacts";

    private static final String USERNAME = "root";

    private static final String PASSWORD = "Aryan@1311";

    private static Connection connection;

    private static PreparedStatement preparedStatement;

    private static ResultSet resultSet;

    private static TextField nameField, emailField, phoneField, addressField, simCompanyField, occupationField, interestedField;

    private static Button addButton, viewButton, searchButton, editButton, deleteButton;

    private static TextArea resultArea;

    private static final Pattern EMAIL\_PATTERN = Pattern.compile("^[a-zA-Z0-9.\_%+-]+@gmail\\.com$");

    private static final Pattern PHONE\_PATTERN = Pattern.compile("^\\d{10}$");

    public ContactManagementSystem() {

        setTitle("Contact Management System");

        setSize(800, 400); // Adjusted default window size

        setLayout(new GridBagLayout());

        GridBagConstraints gbc = new GridBagConstraints();

        gbc.anchor = GridBagConstraints.WEST;

        gbc.insets = new Insets(5, 5, 5, 5);

        // Labels and Text Fields

        addLabelAndField("Name:", gbc, 0, 0);

        addLabelAndField("Email:", gbc, 0, 1);

        addLabelAndField("Phone:", gbc, 0, 2);

        addLabelAndField("Address:", gbc, 0, 3);

        addLabelAndField("Sim Company:", gbc, 0, 4);

        addLabelAndField("Occupation Type:", gbc, 0, 5);

        addLabelAndField("Interested (yes/no):", gbc, 0, 6);

        // Text Area for Result

        gbc.gridx = 0;

        gbc.gridy = 7;

        gbc.gridwidth = 2;

        gbc.fill = GridBagConstraints.HORIZONTAL;

        resultArea = new TextArea();

        resultArea.setEditable(false);

        add(resultArea, gbc);

        // Buttons

        gbc.gridx = 2;

        gbc.gridy = 0;

        gbc.gridwidth = 1;

        gbc.gridheight = 1;

        gbc.fill = GridBagConstraints.NONE;

        addButton = new Button("Add Contact");

        addButton.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {

                try {

                    addContact();

                    JOptionPane.showMessageDialog(null, "Contact added successfully.");

                } catch (SQLException ex) {

                    ex.printStackTrace();

                }

            }

        });

        add(addButton, gbc);

        viewButton = new Button("View Contacts");

        viewButton.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {

                try {

                    viewContacts();

                } catch (SQLException ex) {

                    ex.printStackTrace();

                }

            }

        });

        gbc.gridy = 1;

        add(viewButton, gbc);

        searchButton = new Button("Search Contact");

        searchButton.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {

                try {

                    searchContact();

                } catch (SQLException ex) {

                    ex.printStackTrace();

                }

            }

        });

        gbc.gridy = 2;

        add(searchButton, gbc);

        editButton = new Button("Edit Contact");

        editButton.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {

                try {

                    editContact();

                } catch (SQLException ex) {

                    ex.printStackTrace();

                }

            }

        });

        gbc.gridy = 3;

        add(editButton, gbc);

        deleteButton = new Button("Delete Contact");

        deleteButton.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {

                try {

                    deleteContact();

                } catch (SQLException ex) {

                    ex.printStackTrace();

                }

            }

        });

        gbc.gridy = 4;

        add(deleteButton, gbc);

    }

    public static void main(String[] args) {

        ContactManagementSystem gui = new ContactManagementSystem();

        gui.addWindowListener(new WindowAdapter() {

            public void windowClosing(WindowEvent e) {

                System.exit(0);

            }

        });

        gui.setVisible(true);

    }

    public static void connect() throws SQLException {

        connection = DriverManager.getConnection(JDBC\_URL, USERNAME, PASSWORD);

    }

    public void addLabelAndField(String labelText, GridBagConstraints gbc, int x, int y) {

        Label label = new Label(labelText);

        gbc.gridx = x;

        gbc.gridy = y;

        add(label, gbc);

        TextField field = new TextField(20);

        gbc.gridx = x + 1;

        gbc.gridy = y;

        add(field, gbc);

        switch (labelText) {

            case "Name:":

                nameField = field;

                break;

            case "Email:":

                emailField = field;

                break;

            case "Phone:":

                phoneField = field;

                break;

            case "Address:":

                addressField = field;

                break;

            case "Sim Company:":

                simCompanyField = field;

                break;

            case "Occupation Type:":

                occupationField = field;

                break;

            case "Interested (yes/no):":

                interestedField = field;

                break;

        }

    }

    public static void addContact() throws SQLException {

        String name = nameField.getText();

        String email = emailField.getText();

        String phone = phoneField.getText();

        String address = addressField.getText();

        String simCompany = simCompanyField.getText();

        String occupationType = occupationField.getText();

        String interested = interestedField.getText();

        if (!isValidEmail(email)) {

            JOptionPane.showMessageDialog(null, "Invalid email format. Please enter a valid Gmail address.");

            return;

        }

        if (!isValidPhoneNumber(phone)) {

            JOptionPane.showMessageDialog(null, "Invalid phone number format. Please enter a 10-digit number without spaces or special characters.");

            return;

        }

        connect();

        preparedStatement = connection.prepareStatement("INSERT INTO contact\_details (name, email, phone, address, sim\_company, occupation\_type, interested) VALUES (?, ?, ?, ?, ?, ?, ?)");

        preparedStatement.setString(1, name);

        preparedStatement.setString(2, email);

        preparedStatement.setString(3, phone);

        preparedStatement.setString(4, address);

        preparedStatement.setString(5, simCompany);

        preparedStatement.setString(6, occupationType);

        preparedStatement.setString(7, interested);

        preparedStatement.executeUpdate();

        close();

        clearFields();

    }

    public static boolean isValidEmail(String email) {

        return EMAIL\_PATTERN.matcher(email).matches();

    }

    public static boolean isValidPhoneNumber(String phone) {

        return PHONE\_PATTERN.matcher(phone).matches();

    }

    public static void clearFields() {

        nameField.setText("");

        emailField.setText("");

        phoneField.setText("");

        addressField.setText("");

        simCompanyField.setText("");

        occupationField.setText("");

        interestedField.setText("");

    }

    public static void viewContacts() throws SQLException {

        connect();

        preparedStatement = connection.prepareStatement("SELECT \* FROM contact\_details");

        resultSet = preparedStatement.executeQuery();

        StringBuilder result = new StringBuilder();

        while (resultSet.next()) {

            int id = resultSet.getInt("id");

            String name = resultSet.getString("name");

            String email = resultSet.getString("email");

            String phone = resultSet.getString("phone");

            String address = resultSet.getString("address");

            String simCompany = resultSet.getString("sim\_company");

            String occupationType = resultSet.getString("occupation\_type");

            String interested = resultSet.getString("interested");

            result.append("ID: ").append(id).append(", Name: ").append(name).append(", Email: ").append(email).append(", Phone: ").append(phone).append(", Address: ").append(address).append(", Sim Company: ").append(simCompany).append(", Occupation Type: ").append(occupationType).append(", Interested: ").append(interested).append("\n");

        }

        resultArea.setText(result.toString());

        close();

    }

    public static void searchContact() throws SQLException {

        String[] options = { "ID", "Phone", "Sim Company", "Occupation Type", "Interested (yes/no)" };

        String selectedOption = (String) JOptionPane.showInputDialog(null, "Choose an option:", "Search Contact",

                JOptionPane.DEFAULT\_OPTION, null, options, options[0]);

        if (selectedOption != null) {

            String searchText = JOptionPane.showInputDialog("Enter value to search:");

            if (searchText != null) {

                connect();

                String query = "";

                switch (selectedOption) {

                    case "ID":

                        query = "SELECT \* FROM contact\_details WHERE id = ?";

                        break;

                    case "Phone":

                        query = "SELECT \* FROM contact\_details WHERE phone LIKE ?";

                        break;

                    case "Sim Company":

                        query = "SELECT \* FROM contact\_details WHERE sim\_company LIKE ?";

                        break;

                    case "Occupation Type":

                        query = "SELECT \* FROM contact\_details WHERE occupation\_type LIKE ?";

                        break;

                    case "Interested (yes/no)":

                        query = "SELECT \* FROM contact\_details WHERE interested LIKE ?";

                        break;

                    default:

                        break;

                }

                preparedStatement = connection.prepareStatement(query);

                preparedStatement.setString(1, "%" + searchText + "%");

                resultSet = preparedStatement.executeQuery();

                StringBuilder result = new StringBuilder();

                while (resultSet.next()) {

                    int id = resultSet.getInt("id");

                    String name = resultSet.getString("name");

                    String email = resultSet.getString("email");

                    String phone = resultSet.getString("phone");

                    String address = resultSet.getString("address");

                    String simCompany = resultSet.getString("sim\_company");

                    String occupationType = resultSet.getString("occupation\_type");

                    String interested = resultSet.getString("interested");

                    result.append("ID: ").append(id).append(", Name: ").append(name).append(", Email: ")

                            .append(email).append(", Phone: ").append(phone).append(", Address: ").append(address)

                            .append(", Sim Company: ").append(simCompany).append(", Occupation Type: ")

                            .append(occupationType).append(", Interested: ").append(interested).append("\n");

                }

                resultArea.setText(result.toString());

                close();

            }

        }

    }

    public static void editContact() throws SQLException {

        String[] options = { "ID", "Phone", "Sim Company", "Occupation Type", "Interested (yes/no)" };

        String selectedOption = (String) JOptionPane.showInputDialog(null, "Choose an option:", "Edit Contact",

                JOptionPane.DEFAULT\_OPTION, null, options, options[0]);

        if (selectedOption != null) {

            String searchText = JOptionPane.showInputDialog("Enter value to edit:");

            if (searchText != null) {

                connect();

                String query = "";

                switch (selectedOption) {

                    case "ID":

                        query = "SELECT \* FROM contact\_details WHERE id = ?";

                        break;

                    case "Phone":

                        query = "SELECT \* FROM contact\_details WHERE phone LIKE ?";

                        break;

                    case "Sim Company":

                        query = "SELECT \* FROM contact\_details WHERE sim\_company LIKE ?";

                        break;

                    case "Occupation Type":

                        query = "SELECT \* FROM contact\_details WHERE occupation\_type LIKE ?";

                        break;

                    case "Interested (yes/no)":

                        query = "SELECT \* FROM contact\_details WHERE interested LIKE ?";

                        break;

                    default:

                        break;

                }

                preparedStatement = connection.prepareStatement(query);

                preparedStatement.setString(1, "%" + searchText + "%");

                resultSet = preparedStatement.executeQuery();

                if (resultSet.next()) {

                    int id = resultSet.getInt("id");

                    String name = resultSet.getString("name");

                    String email = resultSet.getString("email");

                    String phone = resultSet.getString("phone");

                    String address = resultSet.getString("address");

                    String simCompany = resultSet.getString("sim\_company");

                    String occupationType = resultSet.getString("occupation\_type");

                    String interested = resultSet.getString("interested");

                    // Now you can populate the fields with the retrieved values

                    nameField.setText(name);

                    emailField.setText(email);

                    phoneField.setText(phone);

                    addressField.setText(address);

                    simCompanyField.setText(simCompany);

                    occupationField.setText(occupationType);

                    interestedField.setText(interested);

                    // Perform any necessary UI updates or validation checks

                } else {

                    JOptionPane.showMessageDialog(null, "Contact not found.");

                }

                close();

            }

        }

    }

    public static void deleteContact() throws SQLException {

        String[] options = { "ID", "Phone", "Sim Company", "Occupation Type", "Interested (yes/no)" };

        String selectedOption = (String) JOptionPane.showInputDialog(null, "Choose an option:", "Delete Contact",

                JOptionPane.DEFAULT\_OPTION, null, options, options[0]);

        if (selectedOption != null) {

            String searchText = JOptionPane.showInputDialog("Enter value to delete:");

            if (searchText != null) {

                connect();

                String query = "";

                switch (selectedOption) {

                    case "ID":

                        query = "DELETE FROM contact\_details WHERE id = ?";

                        break;

                    case "Phone":

                        query = "DELETE FROM contact\_details WHERE phone LIKE ?";

                        break;

                    case "Sim Company":

                        query = "DELETE FROM contact\_details WHERE sim\_company LIKE ?";

                        break;

                    case "Occupation Type":

                        query = "DELETE FROM contact\_details WHERE occupation\_type LIKE ?";

                        break;

                    case "Interested (yes/no)":

                        query = "DELETE FROM contact\_details WHERE interested LIKE ?";

                        break;

                    default:

                        break;

                }

                preparedStatement = connection.prepareStatement(query);

                preparedStatement.setString(1, "%" + searchText + "%");

                int rowsAffected = preparedStatement.executeUpdate();

                if (rowsAffected > 0) {

                    JOptionPane.showMessageDialog(null, "Contact deleted successfully.");

                } else {

                    JOptionPane.showMessageDialog(null, "Contact not found.");

                }

                close();

            }

        }

    }

    public static void close() throws SQLException {

        if (resultSet != null) {

            resultSet.close();

        }

        if (preparedStatement != null) {

            preparedStatement.close();

        }

        if (connection != null) {

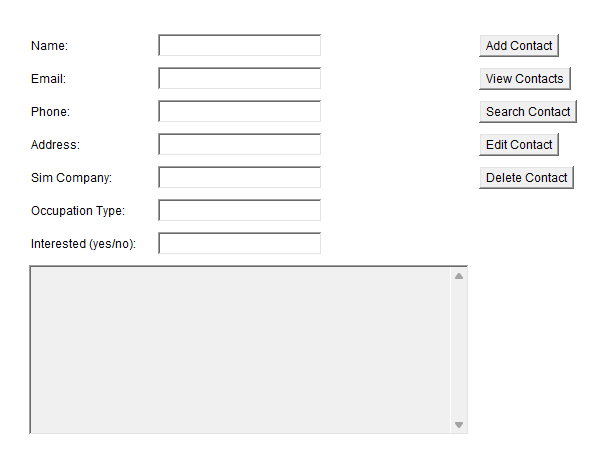
            connection.close();

        }

    }

}

**Screenshots of Output:**

****

