Java GUI - Form Validation, Data Saving, and Database Connection

1. Introduction

In Java Swing applications, adding form validation, saving user input, and connecting to a database are essential skills for building real-world applications. This guide will walk you through:

- Validating user input using Java Swing
- Saving data to a file or database
- Connecting your GUI with a MySQL database using JDBC

2. Form Validation

Form validation ensures the user provides correct and complete data. In Swing, you can validate fields before processing the input.

Example: Ensuring the user enters a name and email.

```
1.  Code Sample:
JTextField nameField = new JTextField();
JTextField emailField = new JTextField();
JButton submitButton = new JButton("Submit");

submitButton.addActionListener(e -> {
    String name = nameField.getText().trim();
    String email = emailField.getText().trim();

    if (name.isEmpty() || email.isEmpty()) {
        JOptionPane.showMessageDialog(null, "Please fill all fields");
    } else if (!email.contains("@")) {
        JOptionPane.showMessageDialog(null, "Invalid email address");
    } else {
        // Proceed with saving
    }
});
2.
```

3. Saving Data to a File

You can save user inputs to a local text file using FileWriter or BufferedWriter.

Example:

```
1. try (FileWriter writer = new FileWriter("output.txt", true)) {
    writer.write("Name: " + name + ", Email: " + email + "\n");
    JOptionPane.showMessageDialog(null, "Data saved successfully!");
} catch (IOException ex) {
    ex.printStackTrace();
}
2.
```

4. Connecting GUI to MySQL Database

To connect a Java application with MySQL, you need the MySQL JDBC driver. Use the following steps:

1. Add the MySQL connector JAR to your classpath.
2. Use JDBC code to insert data from GUI into a MySQL table.

Example:

```
1. String url = "jdbc:mysql://localhost:3306/yourDatabase";
String user = "root";
String password = "yourPassword";

try {
    Connection con = DriverManager.getConnection(url, user, password);
    String query = "INSERT INTO users (name, email) VALUES (?, ?)";
    PreparedStatement pst = con.prepareStatement(query);
    pst.setString(1, name);
    pst.setString(2, email);
    pst.executeUpdate();
    JOptionPane.showMessageDialog(null, "Data inserted successfully!");
    con.close();
} catch (SQLException ex) {
    ex.printStackTrace();
}
2.
```

5. Conclusion

Form validation, data saving, and database connectivity are core features for real-world Java GUI applications. Proper validation improves reliability, while storing and retrieving data enhances functionality and user experience.