Object - Oriented Programming 2

Week 9. SQL Package, Steps to Connect to Database, Connecting to MySQL Database

By Elubu Joseph - MSc.IS

Lecturer

Department of Information Technology

Kumi University

Email: josebulinda@gmail.com

jose@kumiuniversity.ac.ug

Agenda

1. SQL Package,
2. Steps to Connect to Database,
3. Connecting to MySQL Database

JDBC Vs SQL Package

JDBC API

Before we talk about sql package, we first need to note that JDBC is mainly divided into two packages. And that whenever we want to connect to the database, we have to import these packages to use classes and interfaces in our application. Below are the two packages; the sql under java package and the sql under javax package.

1.java.sql

2.javax.sql

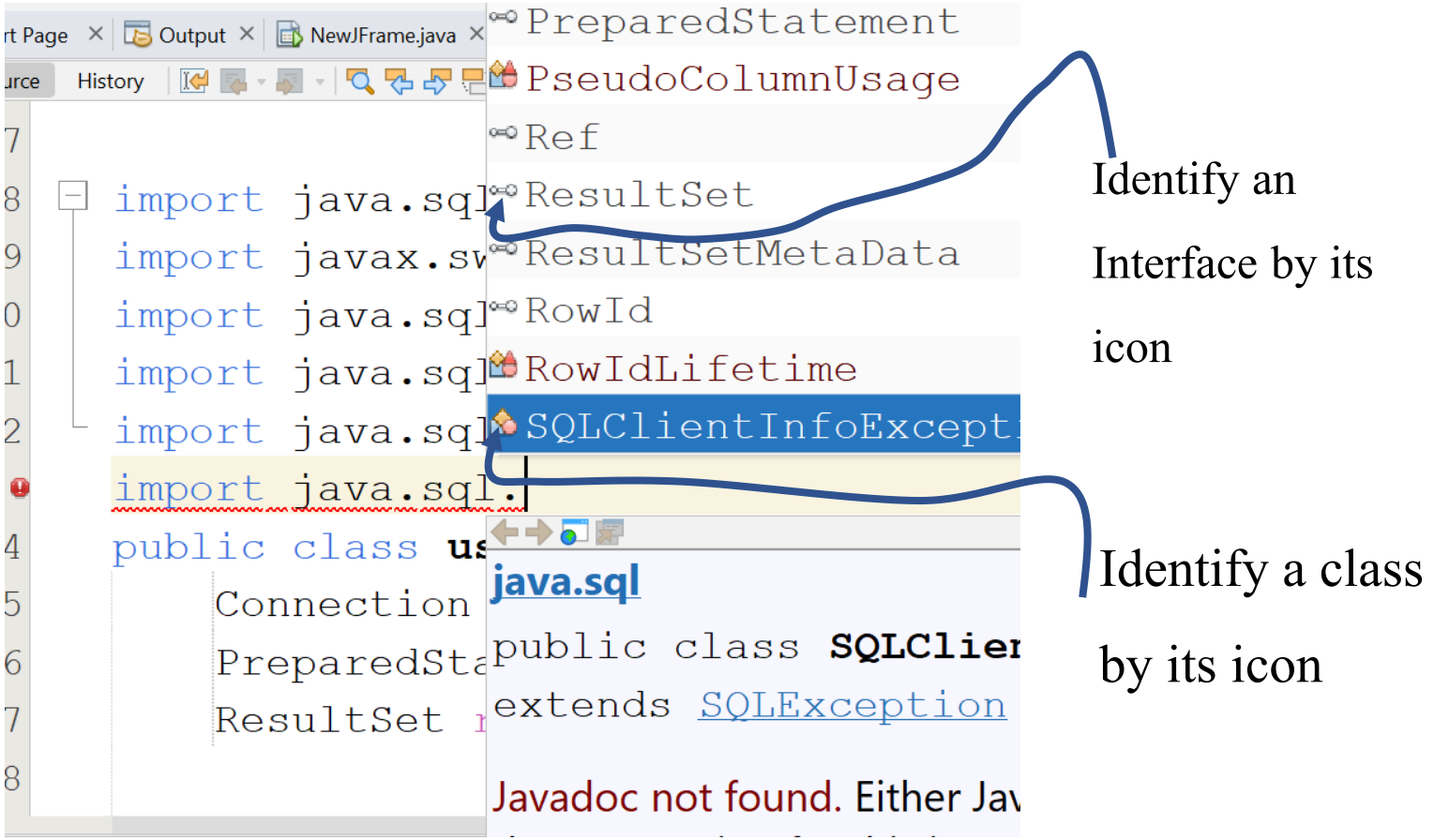
Note. javax.sql is an extension to java.sql. It contains additional classes and interfaces.

java.sql package

include classes and interfaces to perform almost all JDBC operation such as creating and executing SQL Queries.

Important classes and interfaces

of java.sql package



classes/interface	Description
java.sql.**BLOB**	Interface that provide support for BLOB(Binary Large Object) SQL type.
java.sql.**Connection**	Interface used to create a connection with specific database
java.sql.**CallableStatement**	Interface used to Execute stored procedures
java.sql.**CLOB**	Provide support for CLOB(Character Large Object) SQL type.
java.sql.**Date**	Class Provide support for Date SQL type.
java.sql.**Driver**	Class is used to create an instance of a driver with the DriverManager.
java.sql.**DriverManager**	This class manages database drivers.
java.sql.**PreparedStatement**	An interface used to create and execute parameterized query.
java.sql.**ResultSet**	An interface that provide methods to access the result row-by-row.
java.sql.**Savepoint**	Interface that Specify savepoint in transaction.
java.sql.**SQLException**	This class Encapsulate all JDBC related exception.
java.sql.**Statement**	This interface is used to execute SQL statements.

DatabaseMetaData	Comprehensive information about the database as a whole.
DriverAction	An interface that must be implemented when a Driver wants to be notified by DriverManager.
ResultSetMetaData	An object that can be used to get information about the types and properties of the columns in a ResultSet object.
RowId	An interface that is representation (mapping) in the Java programming language of an SQL ROWID value.
Savepoint	The representation of a savepoint, which is a point within the current transaction that can be referenced from the **Connection.rollback** method.
SQLData	The interface used for the custom mapping of an SQL user-defined type (UDT) to a class in the Java programming language.
SQLInput	An input stream that contains a stream of values representing an instance of an SQL structured type or an SQL distinct type.
SQLOutput	The output stream for writing the attributes of a user-defined type back to the database.
SQLType	An object that is used to identify a generic SQL type, called a JDBC type or a vendor specific data type.
SQLXML	Interface that represents mapping in the JavaTM programming language for the SQL XML type.
Statement	The object used for executing a static SQL statement and returning the results it produces.
Struct	The standard mapping in the Java programming language for an SQL structured type.
Wrapper	Interface for JDBC classes which provide the ability to retrieve the delegate instance when the instance in question is in fact a proxy class.

The javax.sql package

This package is also known as JDBC extension API. It provides classes and interface to access server-side data.

Classes and interfaces of javax.sql package

classes/interface	**Description**
javax.sql.ConnectionEvent	Provide information about occurence of event.
javax.sql.ConnectionEventListener	Used to register event generated by **PooledConnection** object.
javax.sql.DataSource	Represent the **DataSource** interface used in an application.
javax.sql.PooledConnection	provide object to manage connection pools.

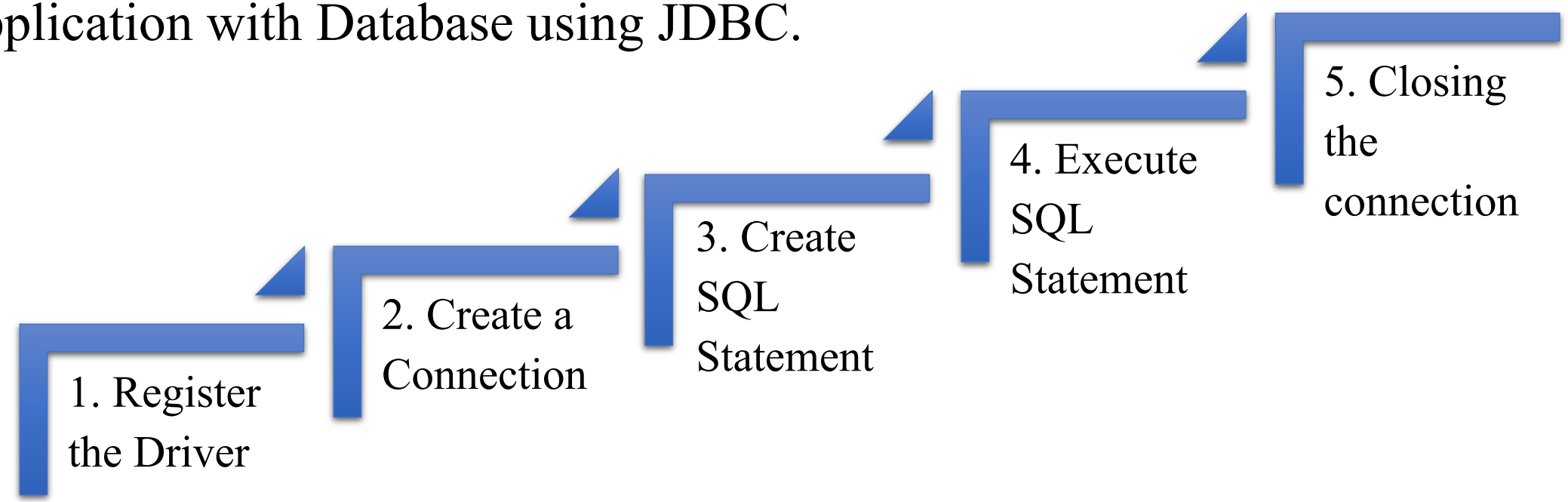
The javax.sql package

Classes and interfaces of javax.sql package

classes/interface	**Description**
CommonDataSource	Interface that defines the methods which are common between **DataSource, XADataSource and ConnectionPoolDataSource**.
RowSet	The interface that adds support to the JDBC API for the JavaBeans™ component model.
RowSetInternal	The interface that a RowSet object implements in order to present itself to a **RowSetReader or RowSetWriter** object.
RowSetListener	An interface that must be implemented by a component that wants to be notified when a significant event happens in the life of a **RowSet** object.
RowSetMetaData	An object that contains information about the columns in a **RowSet** object.
RowSetReader	The facility that a disconnected **RowSet** object calls on to populate itself with rows of data.
RowSetWriter	An object that implements the **RowSetWriter** interface, called a **writer**.
StatementEventListener	An object that registers to be notified of events that occur on PreparedStatements that are in the Statement pool.
XAConnection	An object that provides support for distributed transactions.
XADataSource	A factory for **XAConnection** objects that is used internally.

Steps to connect a Java Application to Database

The following 5 steps are the basic steps involve in connecting a Java application with Database using JDBC.



1 Register the Driver

It is first an essential part to create JDBC connection. JDBC API provides a method Class.forName() which is used to load the driver class explicitly. For example, if we want to load a jdbc-odbc driver then the we call it like following.

Example to register with JDBC-ODBC Driver

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
```

2 Create a Connection

After registering and loading the driver in step1, now we will create a connection using getConnection() method of DriverManager class. This method has several overloaded methods that can be used based on the requirement. Basically it require the hostname, database name, username and password to establish connection. Syntax of this method is given below.

```
getConnection(String url)
getConnection(String url, String username, String password)
getConnection(String url, Properties info)
```

2 Create a Connection+

Below are two examples to establish connection with Oracle Driver and mysql Driver

1. Oracle Driver

```
Connection con;
con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","username","password");
```

2. MySQL Driver

```
Connection con;
 con = DriverManager.getConnection("jdbc:mysql://localhost:3306/databaseName","username","password");
```

3 Create SQL Statement

In this step we will create statement object using **PreparedStatement()** which is used to create and execute parameterized queries. It is used to execute the sql queries and defined in Connection class. Syntax of the method is given below.

Syntax

Example to create a SQL statement

```
String sql = "SELECT * FROM users";
PreparedStatement pst = con.prepareStatement(sql);
```

4 Execute SQL Statement

After creating statement, now execute using executeQuery() method of ResultSet interface. This method is used to execute SQL statements. Syntax of the method is given below.

Syntax

```
public ResultSet executeQuery(String query) throws SQLException
```

Example to execute a SQL statement

In this example, we are executing a sql query to select all the records from the **users** table specifically from **“genter”** field and stored into Resultset that further is used to display on a Label called **genderV**.

```
ResultSet rs=pst.executeQuery();
while(rs.next()){
      String genderV = rs.getString("gender");
                        role.addItem(genderV);
}
```

5 Closing the connection

This is final step which includes closing all the connection that we opened in our previous steps. After executing SQL statement you need to close the connection and release the session. The close() method of Connection interface is used to close the connection.

```
con.close();
```

Example Application Connecting to Database using MySQL Driver

Now lets combine all these steps into a single example and create a complete example of JDBC connectivity.

Before we connect to the database, we will need two things, the database and the connection application.

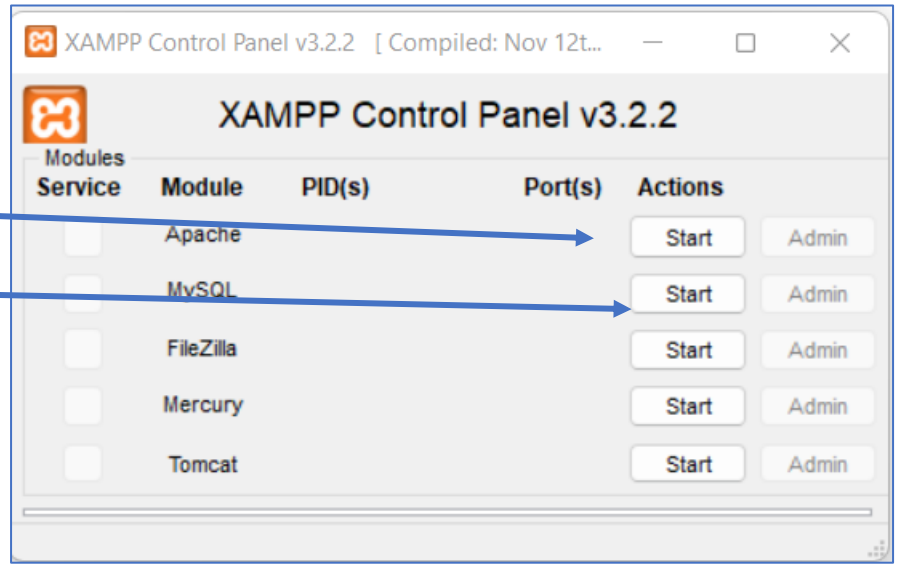
Now lets start by creating a database with the following details:-

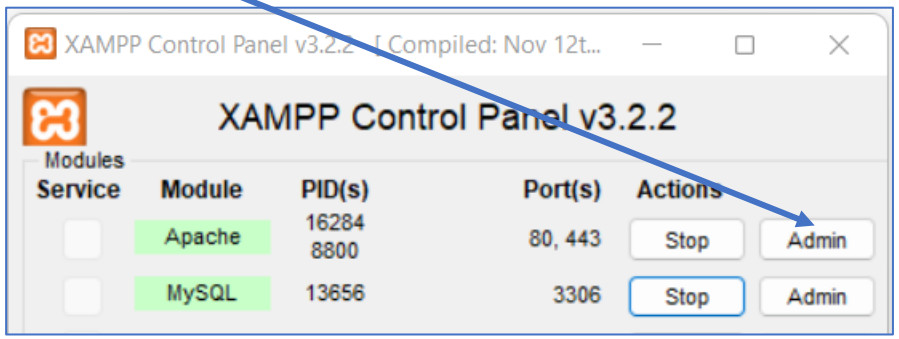
1. Database name: **jdb**
2. Database username: **jdb**
3. Database unser password: **jdb**
4. Host Name: lococalhost.
5. Table Name: **users**

NOTE! We will create these items based on phpMyAdmin of xampp server system, if you do not have xampp installed, please download and install one.

Accessing and Exploring phpMyAdmin Window

1. Open xampp
2. Start Appache
3. Start SQL
4. Click Admin button go to localhost dashboard

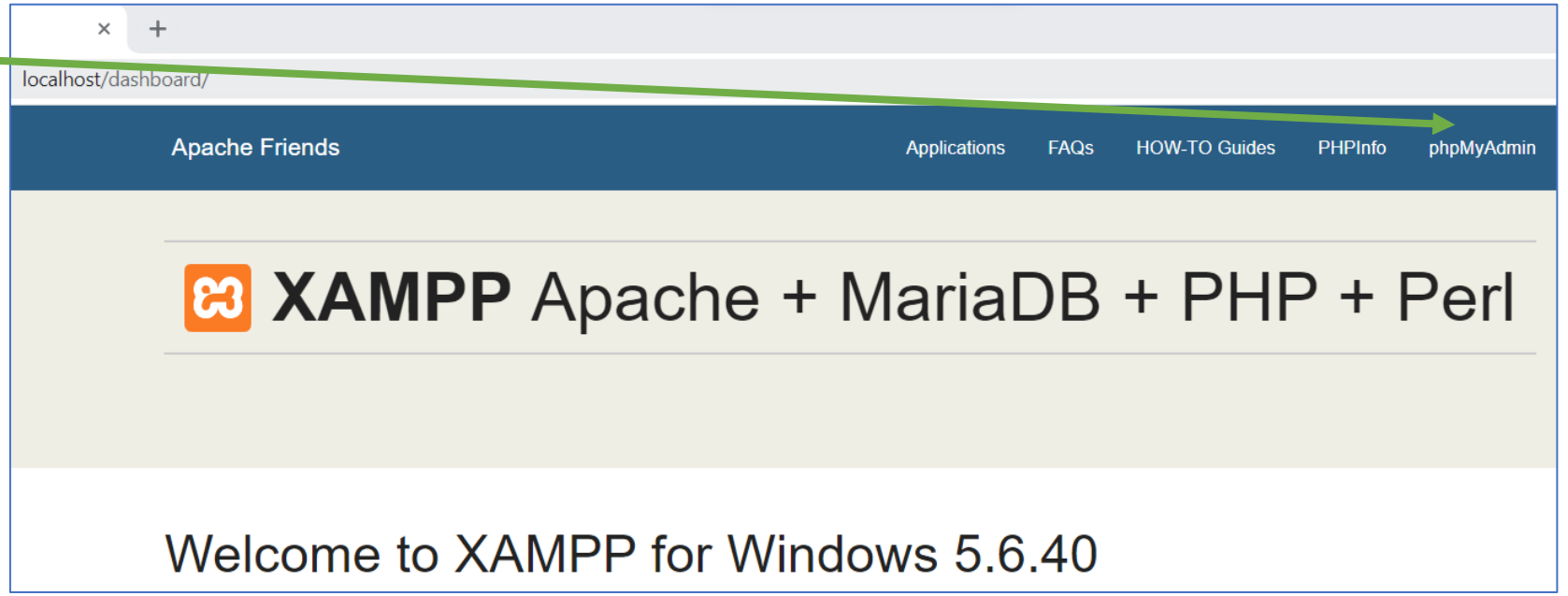




Accessing and Exploring phpMyAdmin Window+

After clicking Admin button on xampp page,

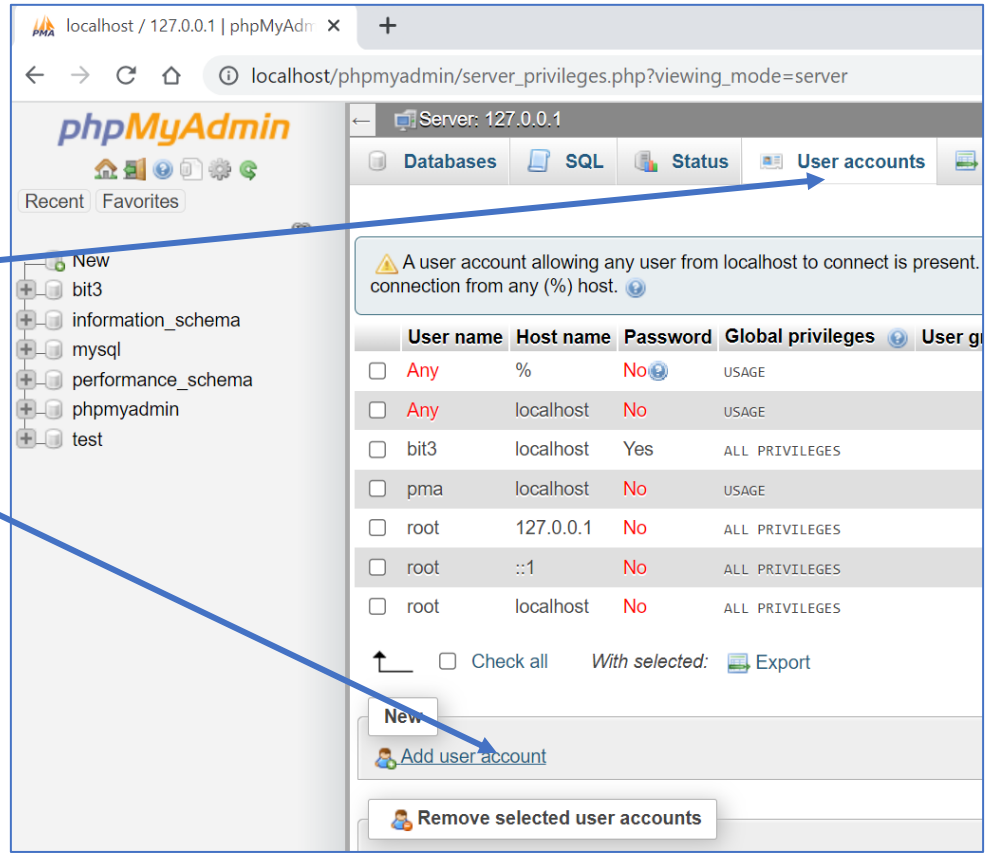
Click phpMyAdmin



Creating MySQL Database a longside Database user.

How?

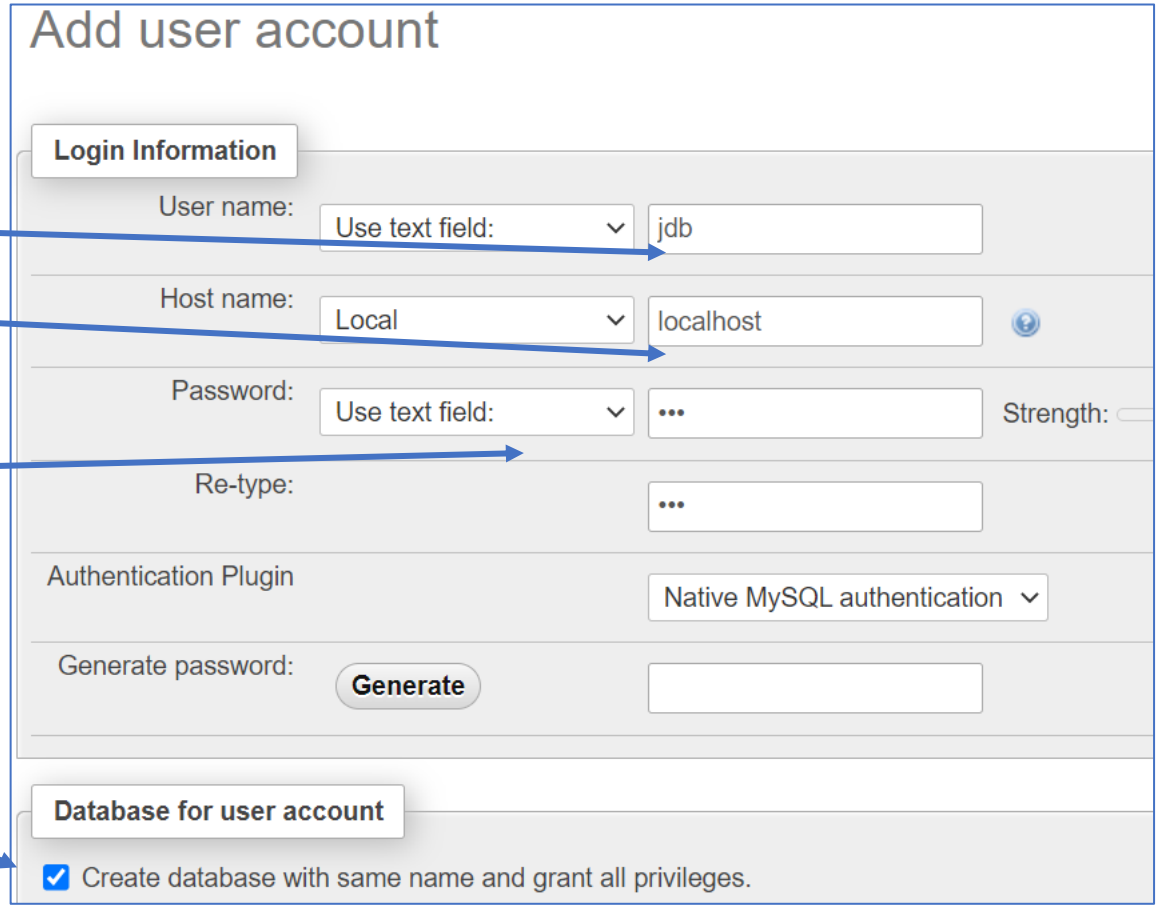
1. After Clicking phpMyAdmin on the window above, Click User Accounts
2. Click Add user account
3. Enter Database user name as jdb
4. Click Create



Creating MySQL Database a longside Database user.+

How?

3. Enter user name as jdb
4. Enter Host name as localhost
5. Enter password as jdb
6. Re-type password as jdb
7. Check Create database with same name and grant all privileges.

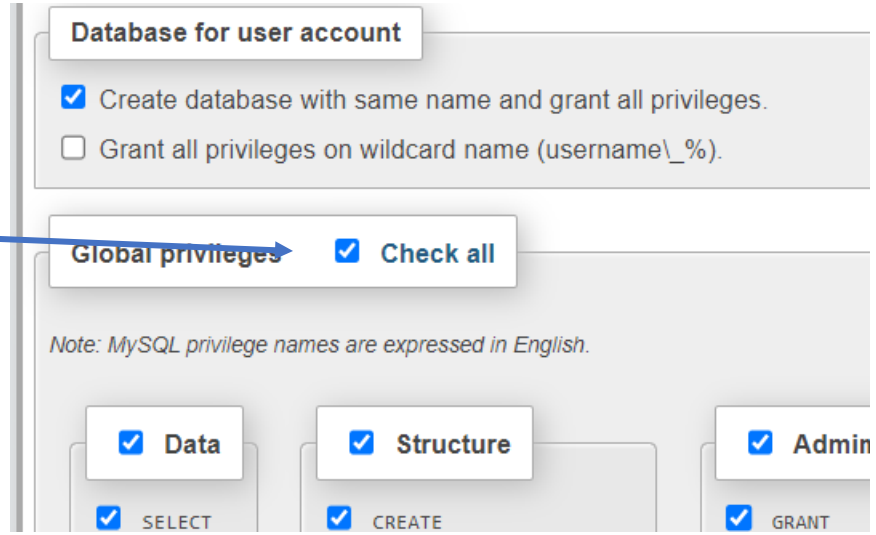


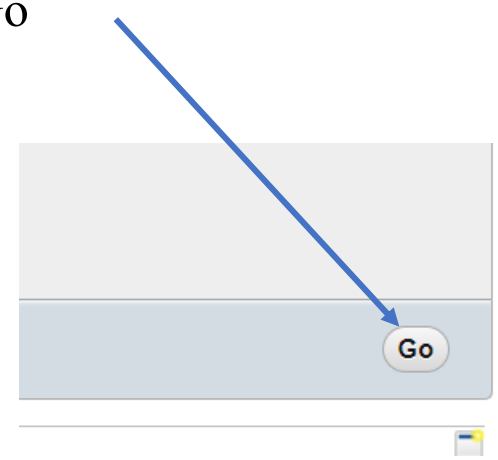
Creating MySQL Database a longside Database user.+

How?

8. Scroll down and click Check all

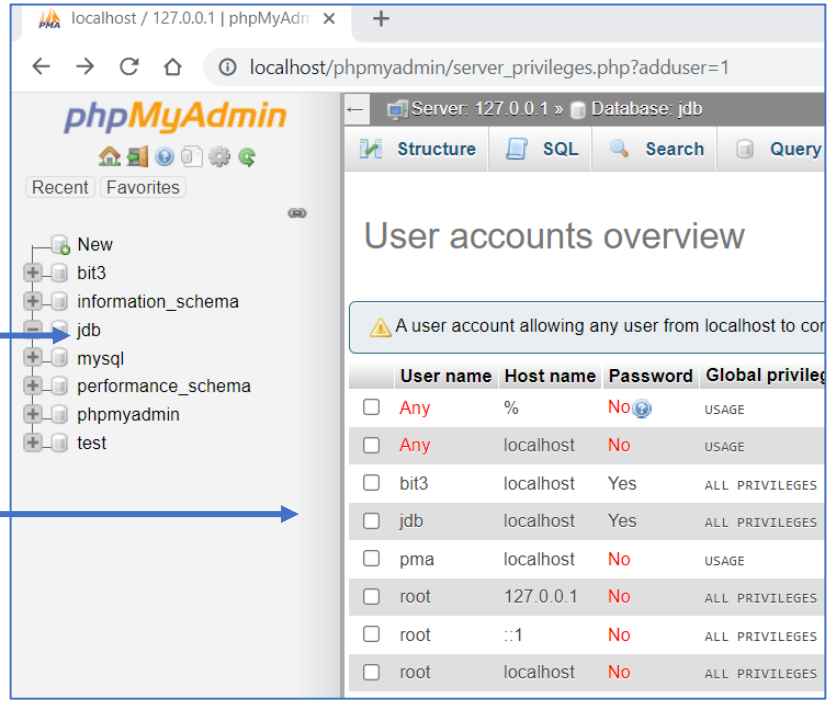
9. Scroll down and click Go





Creating MySQL Database a long side Database user.+

1. Note that you have Database called **jdb**
2. And user name called **jdb**.

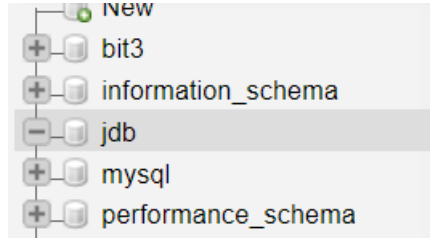


Creating a Table into the Database

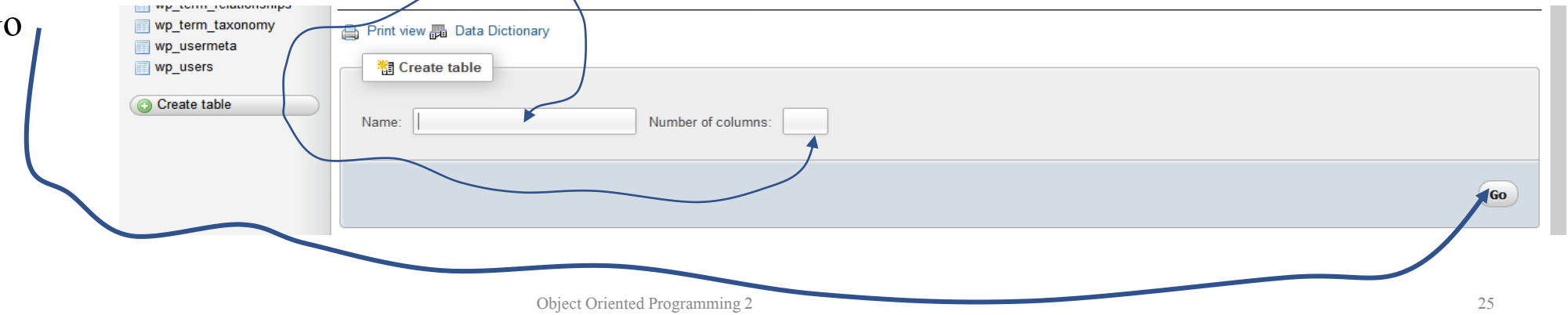
After creating a database, you can now create tables for it to use

How?

1. Click the database you want to create the table for



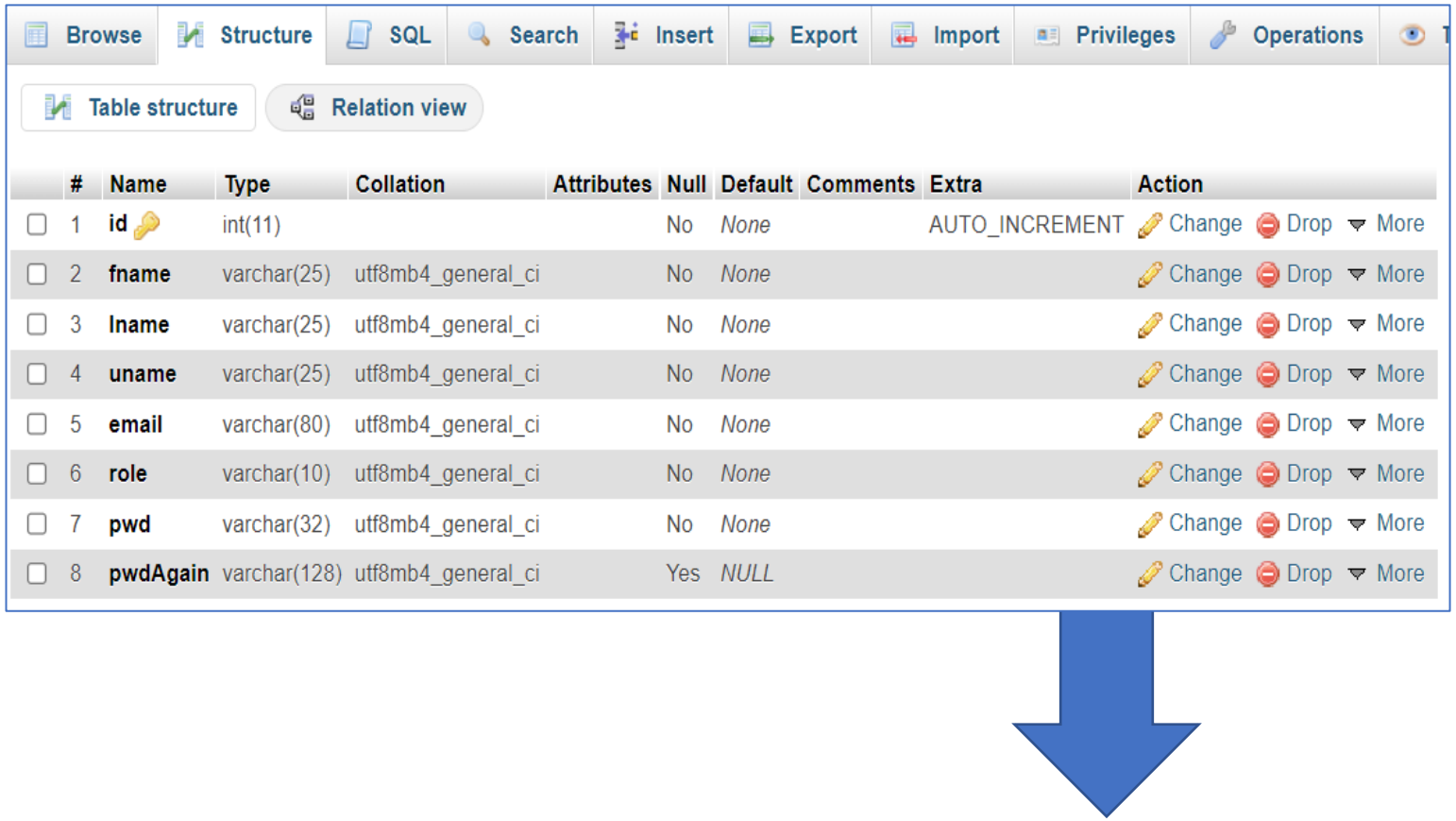
2. Type table name ‘users’ in the name box found under the Create table field
3. Enter the number of columns you want
4. Click Go



Modify Table structure with names and data typs as below.

How?

1. **id**=int(11)PK,
2. **fname**=varchar(25),
3. **lname** = varchar(25),
4. **uname** = varchar(25),
5. **email** = varchar(80),
6. **role** = varchar(10),
7. **pwd** = varchar(32),
8. and **pwdAagin** = varchar(128).

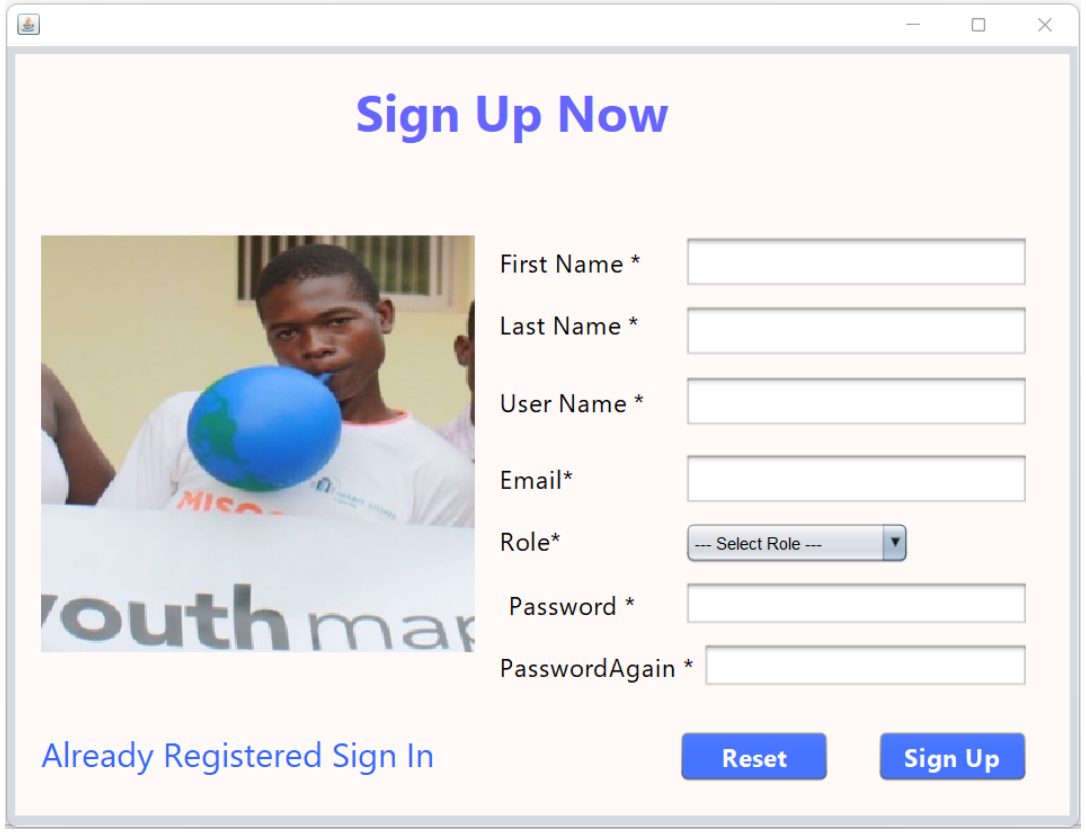


#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)			No	*None*		AUTO_INCREMENT	Change Drop More
2	fname	varchar(25)	utf8mb4_general_ci		No	*None*			Change Drop More
3	lname	varchar(25)	utf8mb4_general_ci		No	*None*			Change Drop More
4	uname	varchar(25)	utf8mb4_general_ci		No	*None*			Change Drop More
5	email	varchar(80)	utf8mb4_general_ci		No	*None*			Change Drop More
6	role	varchar(10)	utf8mb4_general_ci		No	*None*			Change Drop More
7	pwd	varchar(32)	utf8mb4_general_ci		No	*None*			Change Drop More
8	pwdAgain	varchar(128)	utf8mb4_general_ci		Yes	*NULL*			Change Drop More

Creating Java Application

Having finished with the creation of the database and the table, we now need to create a java application that will connect to the database and insert some data from the form.

Basing on the ideas shared on how to create JFrame Form with a number of controls such as label, we will now create a Java Application as below.

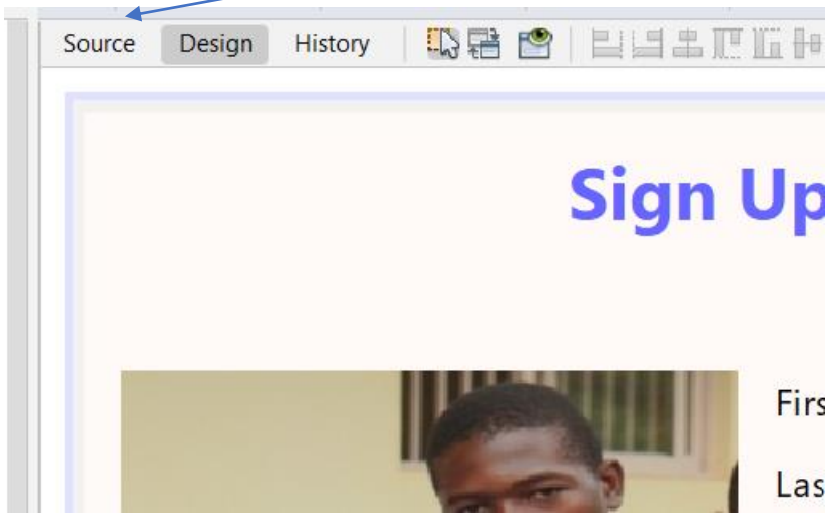


Creating Java Application+

Before we connect to the database and insert data into the table, lets first understand the variable name for each of the following controls.

To be able to see the variable names of the controls on the above Form, we will click on the source table then scroll down.

```
    // Variables declaration - do not modify
    private javax.swing.JLabel Error;
    private javax.swing.JTextField email;
    private javax.swing.JTextField fname;
    private javax.swing.JTextField lname;
    private javax.swing.JPasswordField pwd;
    private javax.swing.JPasswordField pwdAgain;
    private javax.swing.JButton reset;
    private javax.swing.JComboBox<String> role;
    private javax.swing.JButton signup;
    private javax.swing.JTextField uname;
```

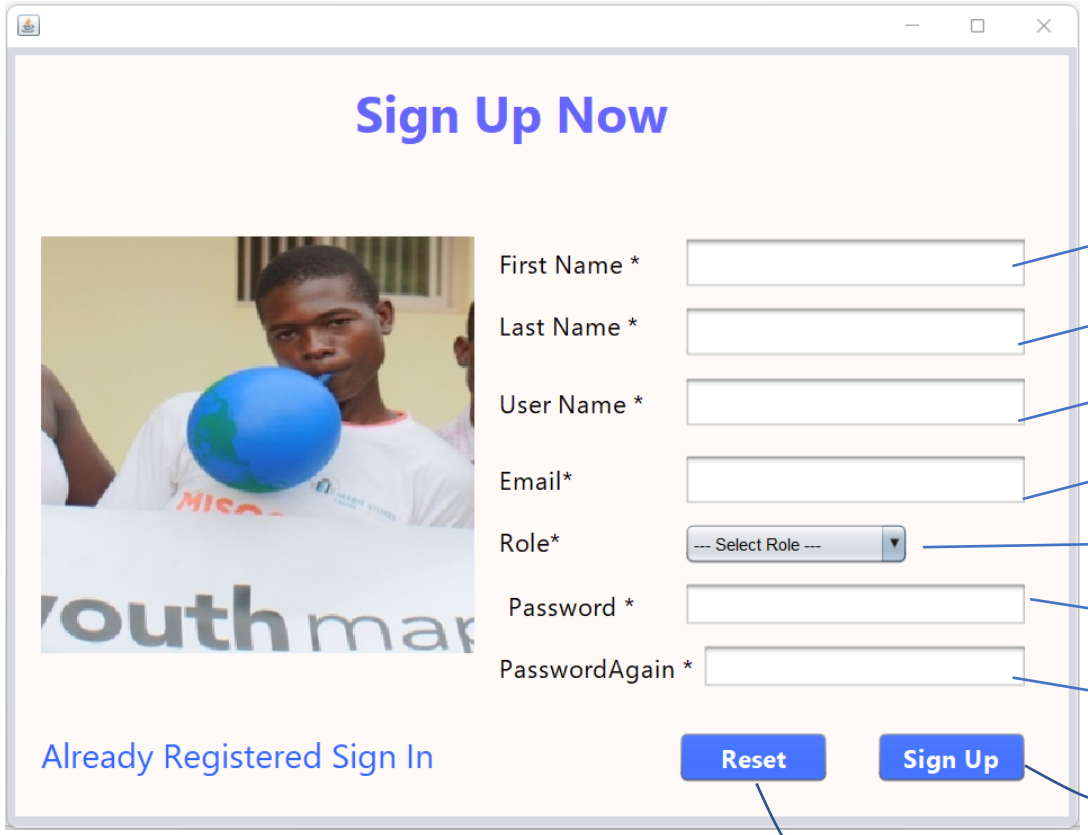


The following are the variable names here.

fname, lname, email, pwd, pwdAgain, uname, role, signup, resetv and Error.

Creating Java Application+Adding variable names

As seen in the above two slides, the following Controls have their variable names set as below.



The following are the variable names for the controls in this applicagtion.

fname,

lname,

uname,

email,

role,

pwd,

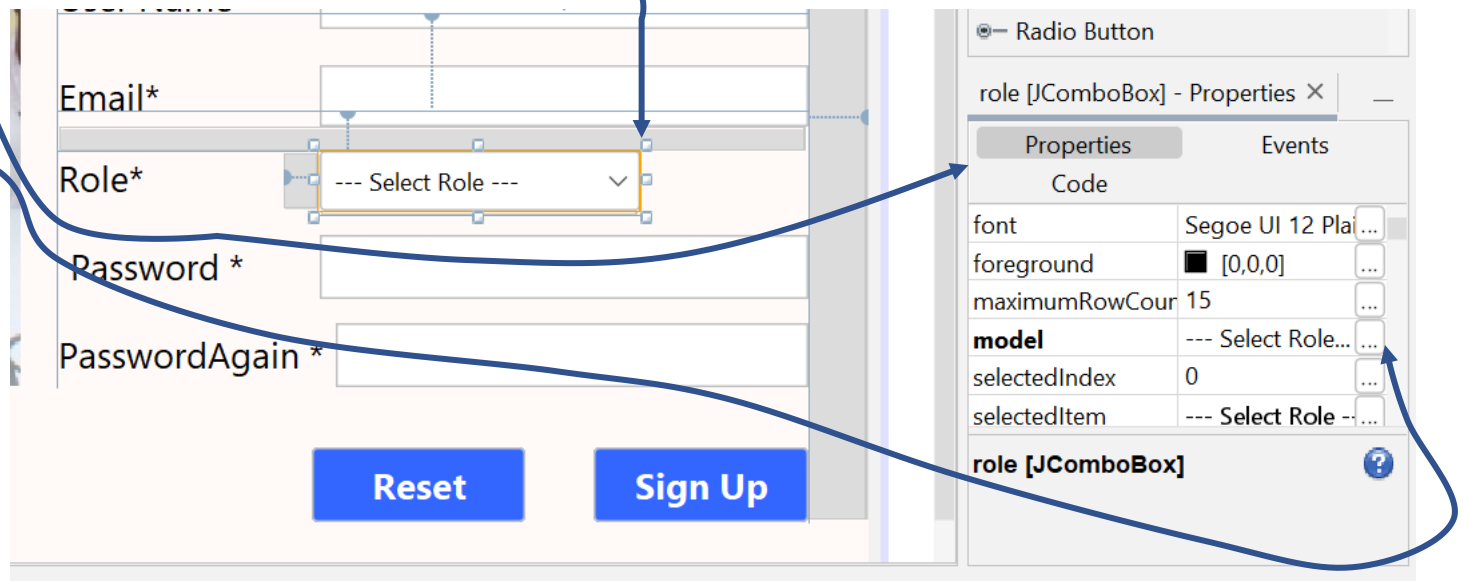
pwdAgain,

signup, reset and Error which is a JLabel.

Adding Items into ComboBox

Steps

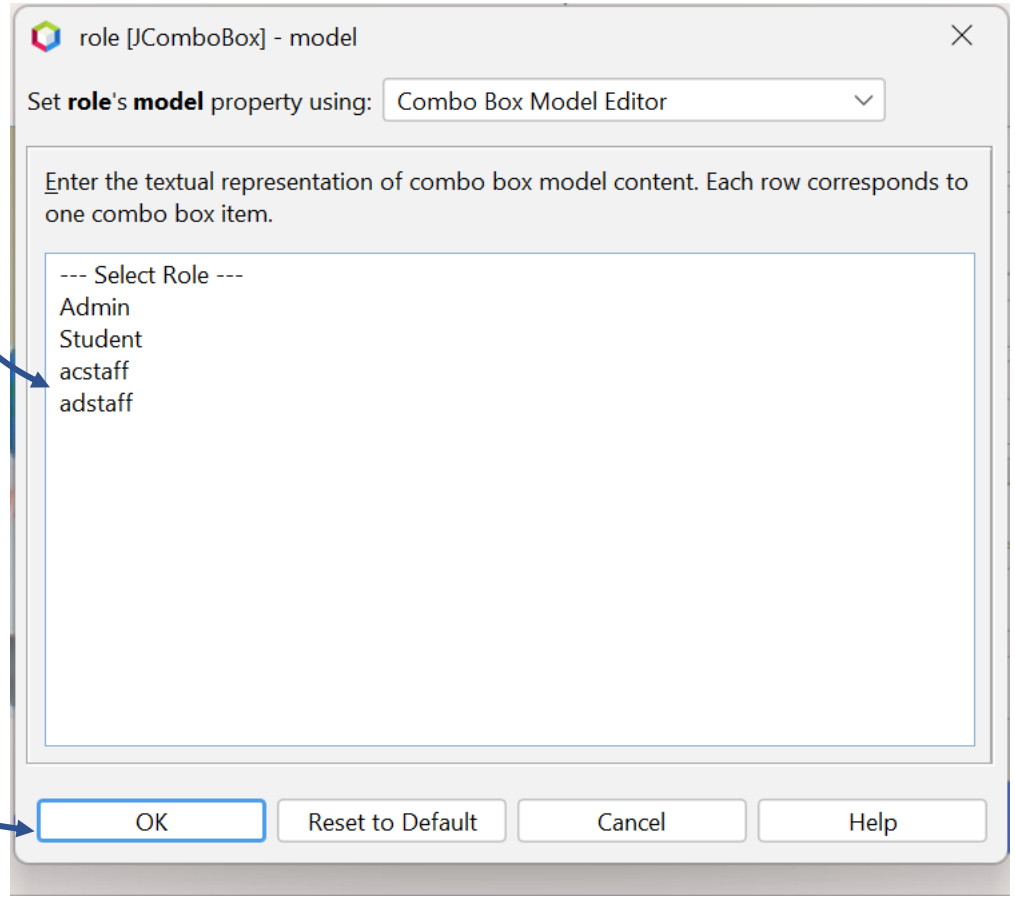
1. Click on the ComboBox
2. Move to Properties section
3. Click on **model** menu button



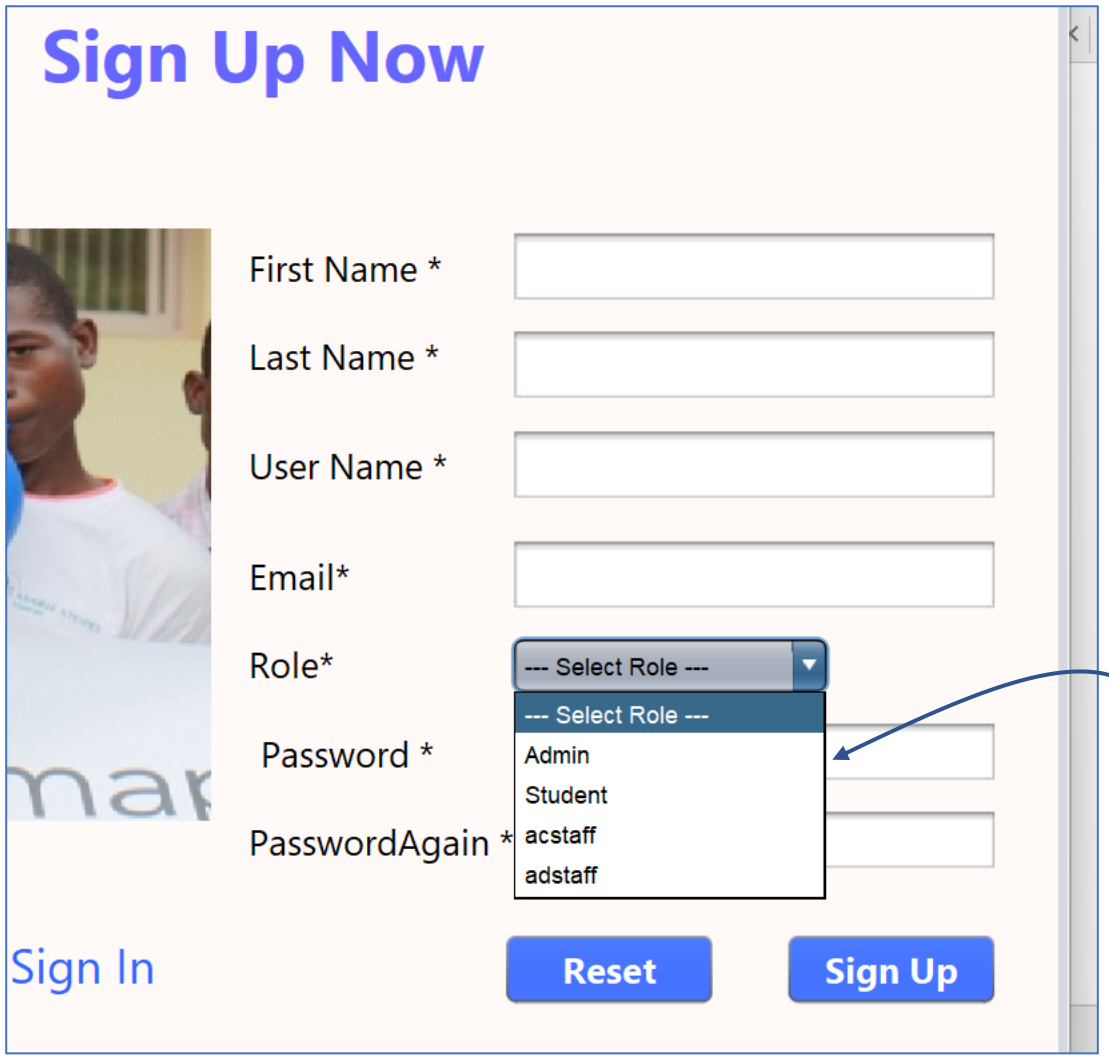
Adding Items into ComboBox+

Steps

4. Enter the List of items you want to populate e.g. Admin, Student, acstaff etc.
5. Click OK



Run the App to view your List



You should be seeing the list you entered

Looking at the code.

```
package gui1;

import javax.swing.JOptionPane;import java.sql.PreparedStatement;
import java.sql.Connection; import java.sql.SQLException;
import java.sql.DriverManager;import java.sql.ResultSet;
public class Signup extends javax.swing.JFrame {
    Connection con = null;
    PreparedStatement pst;
    ResultSet rs = null;


       public Signup() {
         initComponents();


    }
```

ComboBox Data Handling

```
public void comboBox(){
        try{
          con=DriverManager.getConnection("jdbc:mysql://localhost:3306/jdb",
"jdb","jdb");
         String sql = "SELECT * FROM users";
          pst = con.prepareStatement(sql);
           rs = pst.executeQuery();
            while(rs.next()){
                    String genderV = rs.getString("gender");
                    role.addItem(genderV);
                }
            con.close();
        }
        catch(Exception ex){
            JOptionPane.showMessageDialog(null, ex);
        }
    }
```

Checking if some Field(s) on the form are empty When sign Up button is clicked

```
private void signupActionPerformed(java.awt.event.ActionEvent evt) {
        // check if the fields are not empty
        if(fname.getText().trim().isEmpty() || lname.getText().trim().isEmpty()||
            uname.getText().trim().isEmpty() || email.getText().trim().isEmpty()||
            pwd.getText().trim().isEmpty() || pwdAgain.getText().trim().isEmpty()
              ){

            Error.setText("Some field(s) are empty");
        }
        else{

            try{
```

Checking if pwd value is the same as pwdAgain before inserting data into the database-table.

```
if(pwd.getText().equals(pwdAgain.getText())){
        con = DriverManager.getConnection("jdbc:mysql://localhost:3306/jdb", "jdb","jdb");
String sql = "INSERT INTO users (fname, lname, uname, email, role, pwd) VALUES (
?,?,?,?,?,?)";
                pst = con.prepareStatement(sql);
                pst.setString(1, fname.getText());
                pst.setString(2, lname.getText());
                pst.setString(3, uname.getText());
                pst.setString(4, email.getText());
                pst.setString(5, role.getSelectedItem().toString());//combo box
                pst.setString(6, pwd.getText());

                 pst.executeUpdate(); //Data is finally inserted
       JOptionPane.showMessageDialog(null, "Success");
```

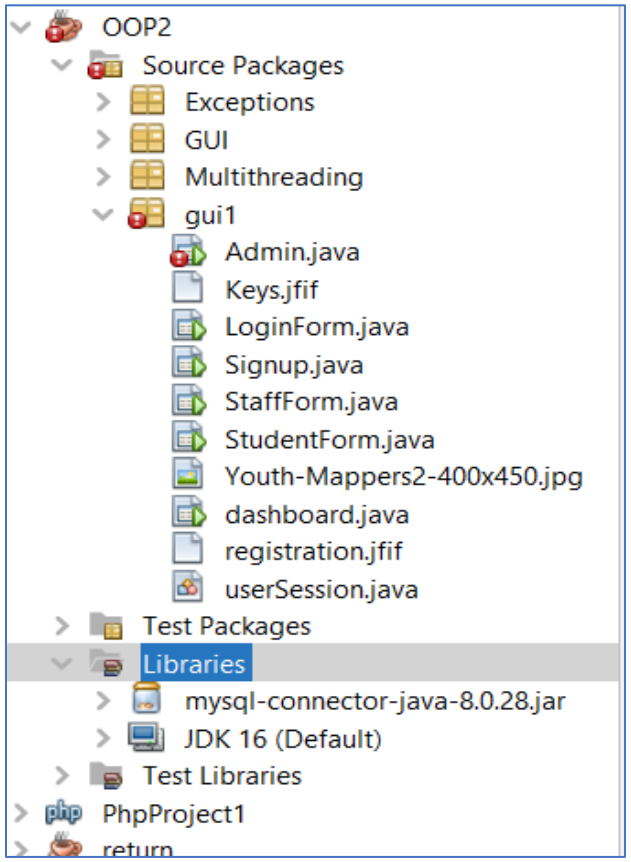
Checking if pwd value is the same as pwdAgain before inserting data into the database.

```
                }
                else{
                    JOptionPane.showMessageDialog(null, "Password Fields donot match");
                }

                }
            catch(SQLException e){
                JOptionPane.showMessageDialog(null, e);
            }
        }

    }
```

Adding MySQL connection jar file



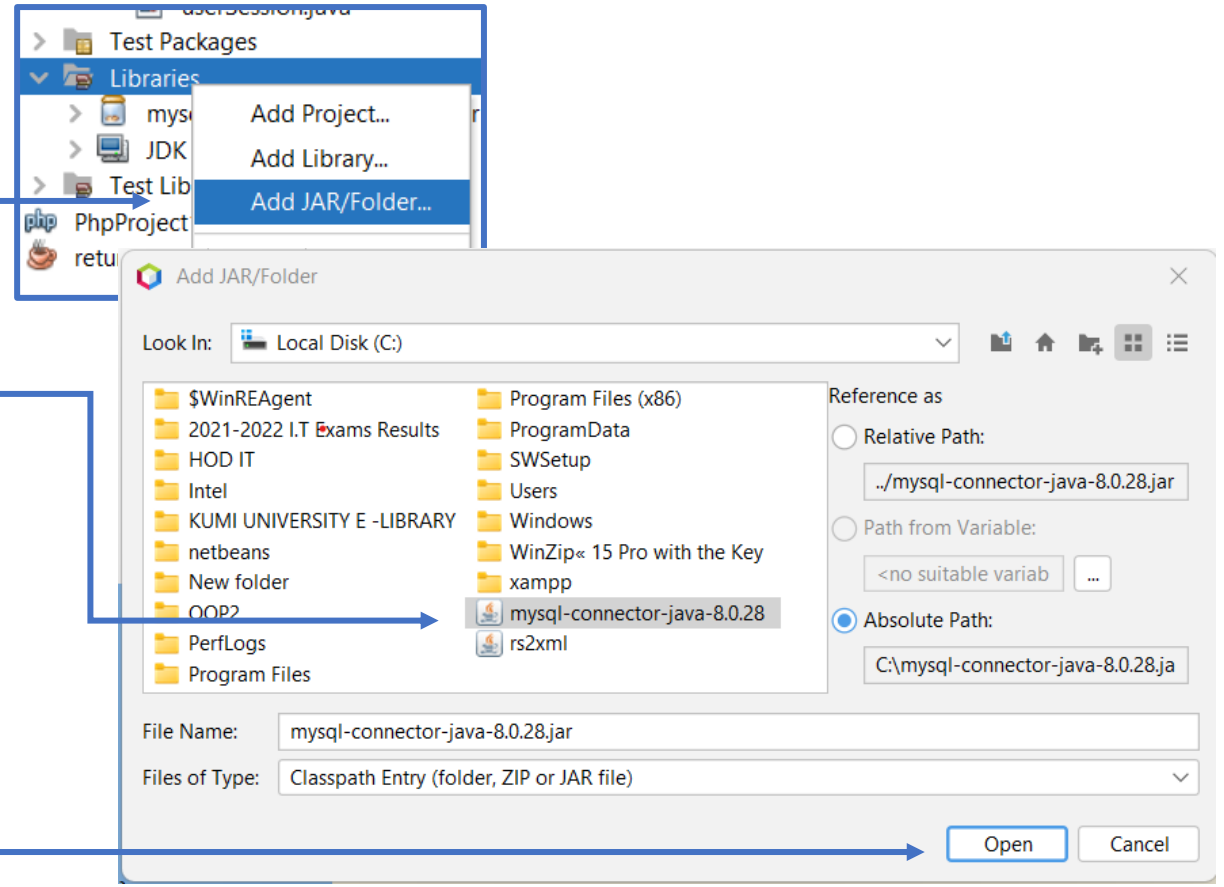
After coding your application, do not forget to add one very important file(**mysql-connector-java-8.0.28.jar**) to your project Libraries. Without it, you will not be able to connect to the database.

The name of the file may vary depending on the version and the date you accessed this material but you can just ask Google for mysql connector, download it and add it into your Projects **Libraries by;**

Adding MySQL connection jar file

Steps for adding mysql connection jar file into your libraries

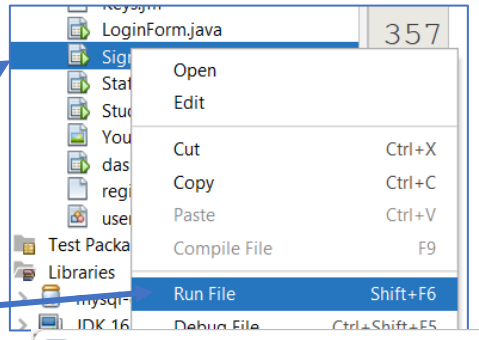
1. Right Click on Libraries
2. Click Add JAR/Folder….
3. Locate the file from wherever you stored it
4. Click on it (mysql-connector-java.8.28.jar)
5. Then click Open

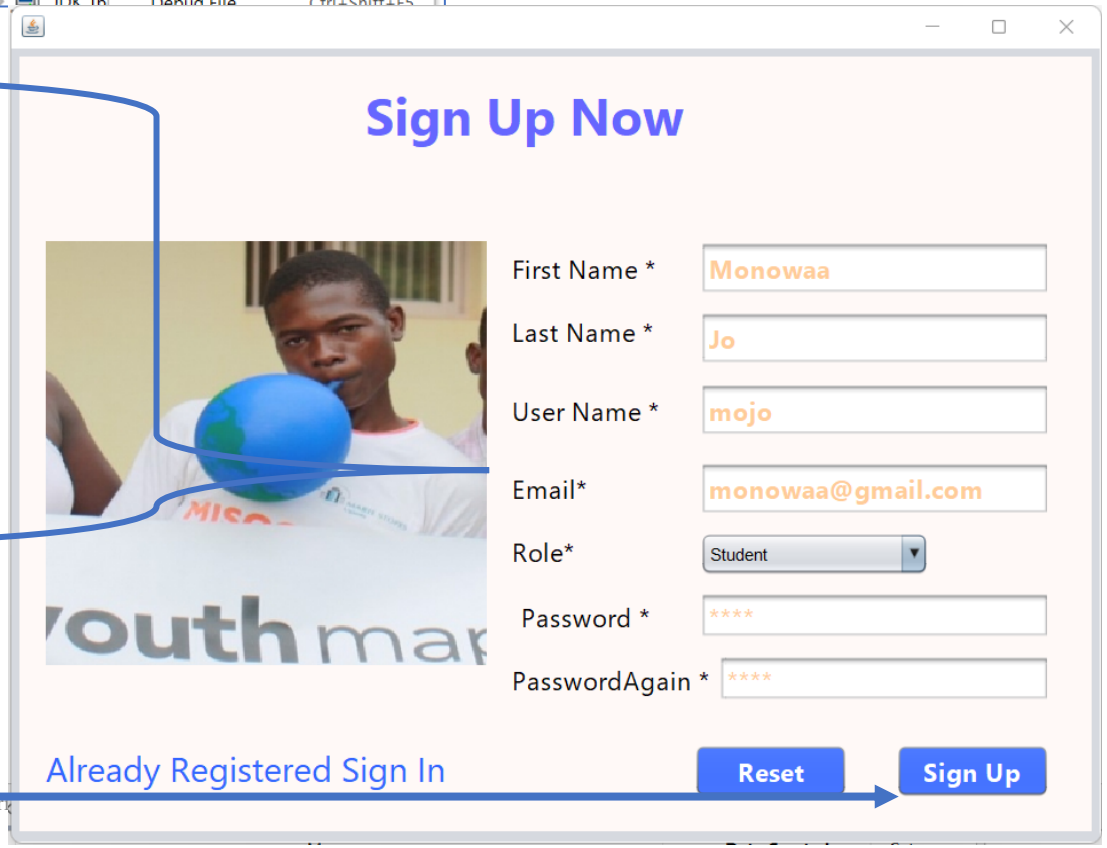


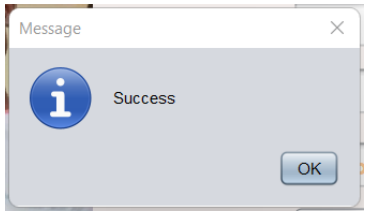
Testing Application

As usual,

1. Right click on you application e.g signup.java
2. Click Run File
3. Enter Data e.g. First Name: Monowaa, Last Name: Jo, User Name mojo, Email: monowaa@gmail.com, Role: Student, Password: 1234 and, PasswordAgain: 1234
4. Click Sign Up







Verify Data insertion

Before you check whether data has been inserted or not, first check the table be you insert the data then check again fir the data intended has been inserted by refreshing the browser.

Before Sign Up

id	fname	lname	uname	email	role	pwd	pwdAgain
1	Elubu	joseph	josebulinda	josebulinda@gmail.com	Admin	123	NULL
24	olele	olele	olele	olele@gmail.com	Student	123	123
25	odele	odele	odele	odele@gmail.com	Student	1	NULL
26	Odoch	Isaac	isaac	isaac@gmail.com	acstaff	123	NULL
27	laki nelson	micheal	crispus	crispusmicheal@gmail.com	Student	003344	NULL
28	ade		ibra	stevennyalimo6@gmail.com	Admin		NULL
29	Okilima	james	okilima	okilimajames@gmail.com	acstaff	123	NULL
30	Ekwelu	Simon	simpet	simpet@gmail.com	Admin	123	NULL
31	Okello	Emmanuel	okelloemma	okelloemma@gmail.com	Admin	123	NULL
32	ok	emma	emma	emma@gmail.com	Student	123	NULL
33	3	3	3	3	Student	3	NULL

lected: Edit Copy Delete Export

After Sign Up

id	fname	lname	uname	email	role	pwd	pwdAgain
1	Elubu	joseph	josebulinda	josebulinda@gmail.com	Admin	123	NULL
24	olele	olele	olele	olele@gmail.com	Student	123	123
25	odele	odele	odele	odele@gmail.com	Student	1	NULL
26	Odoch	Isaac	isaac	isaac@gmail.com	acstaff	123	NULL
27	laki nelson	micheal	crispus	crispusmicheal@gmail.com	Student	003344	NULL
28	ade		ibra	stevennyalimo6@gmail.com	Admin		NULL
29	Okilima	james	okilima	okilimajames@gmail.com	acstaff	123	NULL
30	Ekwelu	Simon	simpet	simpet@gmail.com	Admin	123	NULL
31	Okello	Emmanuel	okelloemma	okelloemma@gmail.com	Admin	123	NULL
32	ok	emma	emma	emma@gmail.com	Student	123	NULL
33	3	3	3	3	Student	3	NULL
34	Monowaa	Jo	mojo	monowaa@gmail.com	Student	1234	NULL

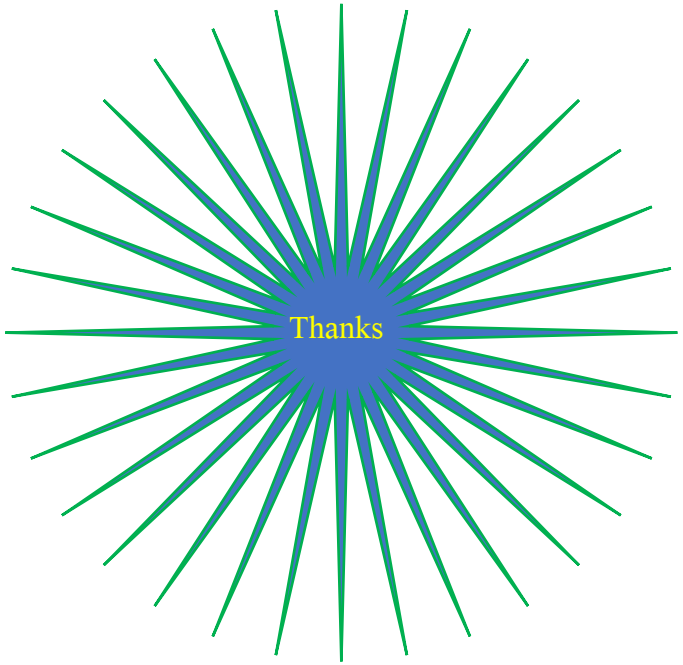
lected: Edit Copy Delete Export

Note that a Student called **Monowaa** has bee registered successfully

Summary

1. SQL Package(Looked at classes and interfaces of both Java.sql package and javax.sql.package) ,
2. Steps to Connect to Database went through 5 steps including registration of the driver, creating connection, creating SQL statemen, Executing sql statement then finally closure of the connection,
3. Connecting to MySQL Database-We built user register Application with ability to insert data into the database. Remember, creation of the database, table, Creating a JFrame form with the controls and coding the controls to allow input.

Thank you for Listening



References

What is javax? how is it useful? - quora. (n.d.). Retrieved October 24, 2022, from https://www.quora.com/What-is-Javax-How-is-it-useful

David-Engel. (n.d.). *Microsoft Open Database Connectivity (ODBC) - open database connectivity (ODBC)*. Open Database Connectivity (ODBC) | Microsoft Learn. Retrieved October 24, 2022, from https://learn.microsoft.com/en-us/sql/odbc/microsoft-open-database-connectivity-odbc?view=sql-server-ver16

Wikimedia Foundation. (2022, September 7). *Java database connectivity*. Wikipedia. Retrieved October 23, 2022, from https://en.wikipedia.org/wiki/Java_Database_Connectivity