**JDBC (Java Database Connectivity)**

JDBC (Java Database Connectivity) is an API for connecting and executing queries on a database.

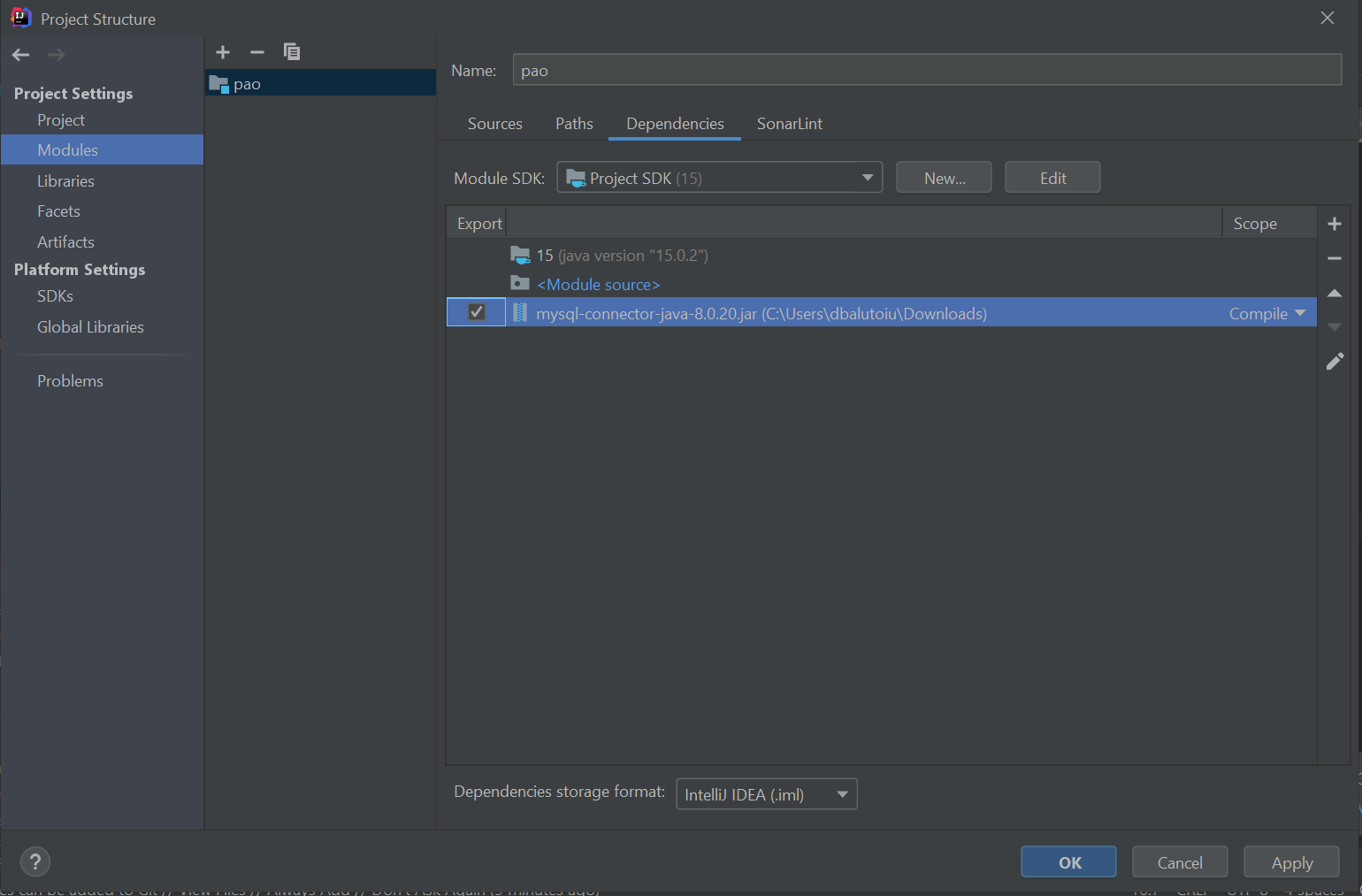
JDBC can work with any database as long as proper drivers are provided.

**Connecting to a Database:**

To connect to a database, we simply have to initialize the driver and open a database connection.

**Registering the Driver:** Class.forName("com.mysql.cj.jdbc.Driver"); We're using a MySQL database, we need the mysql-connector-java dependency: [https://search.maven.org/classic/remotecontent?filepath=mysql/mysql-connector-java/8.0.20/my sql-connector-java-8.0.20.jar](https://search.maven.org/classic/remotecontent?filepath=mysql/mysql-connector-java/8.0.20/my%20sql-connector-java-8.0.20.jar)

(Ctrl + Alt + Shift + S -> “+”)



**Creating the Connection:**

We'll assume that we already have a MySQL server installed and running on localhost (default port 3306)



We will also need to create the database and user:

***CREATE DATABASE lab11;***

***CREATE USER 'user' IDENTIFIED BY 'pass';***

***GRANT ALL on lab11.\* TO 'user';***

To send SQL instructions to the database, we can use instances of type **Statement**, **PreparedStatement** or **CallableStatement**. These are obtained using the **Connection** object.

After executing a query, the result is represented by a **ResultSet** object, with has a structure similar to a table, with lines and columns. The ResultSet uses the **next()** method to move to the next line.

* Statement - The Statement interface contains the essential functions for executing SQL commands.
* PreparedStatement - PreparedStatement objects contain precompiled SQL sequences. They can have one or more parameters denoted by a question mark.
* CallableStatement – used to call stored procedures
  + Setting input parameter values for the stored procedure is done like in the PreparedStatement interface, using setX() methods
  + If the stored procedure has output parameters, we need to add them using the registerOutParameter() method

For the CallableStatement example to work, we need to create the stored procedure into the database:

delimiter //

CREATE PROCEDURE insertPerson(OUT id int, IN name varchar(30), IN age double)

BEGIN

INSERT INTO persons(name,age)

VALUES (name,age);

SET id = LAST\_INSERT\_ID();

END //

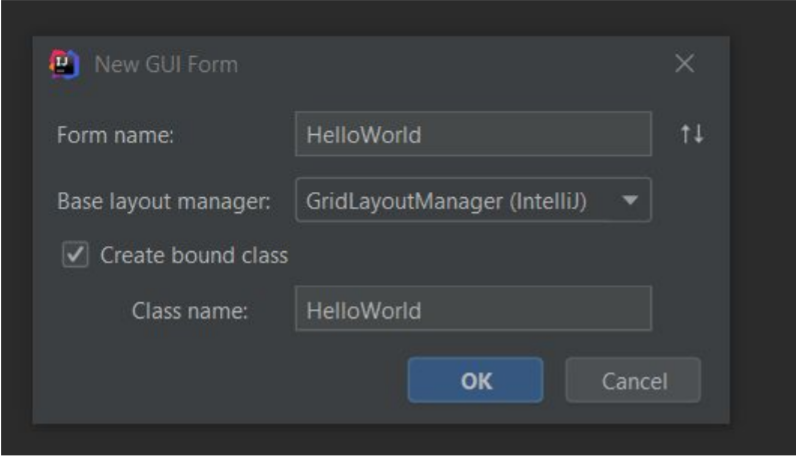
delimiter ;

**Closing the Connection** - When we're no longer using, it's necessary to close the connection to release database resources.

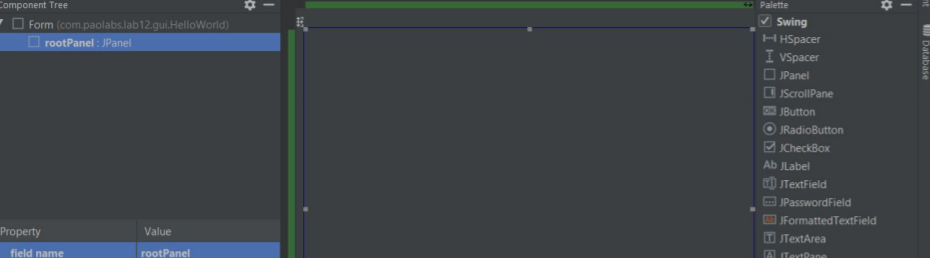
**Swing Graphical User Interface (GUI)**

Java provides a set of components 'Swing components' which allow the creation of a User Interface that works across different platforms (e.g. Mac / Windows).

Right click on the package and choose: *New -> Swing UI Designer -> GUI Form*. Name the form 'HelloWorld', and configure it:



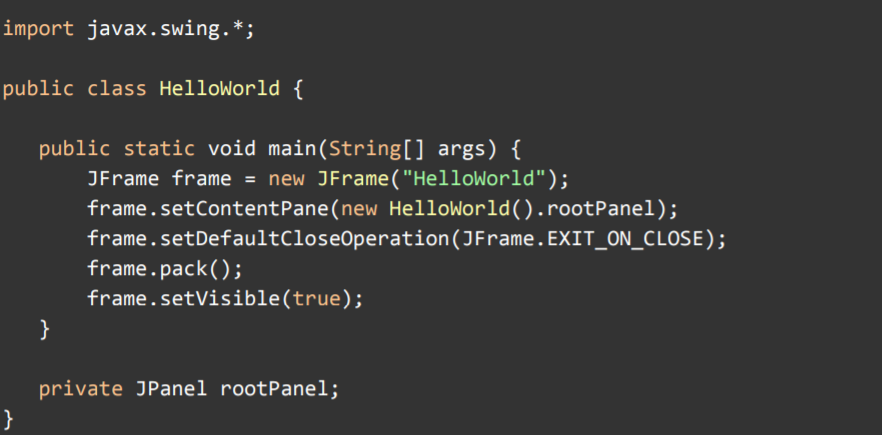
The form will be displayed and will include a panel. In the component tree (select the JPanel and enter the value 'rootPanel' into the field name property box:



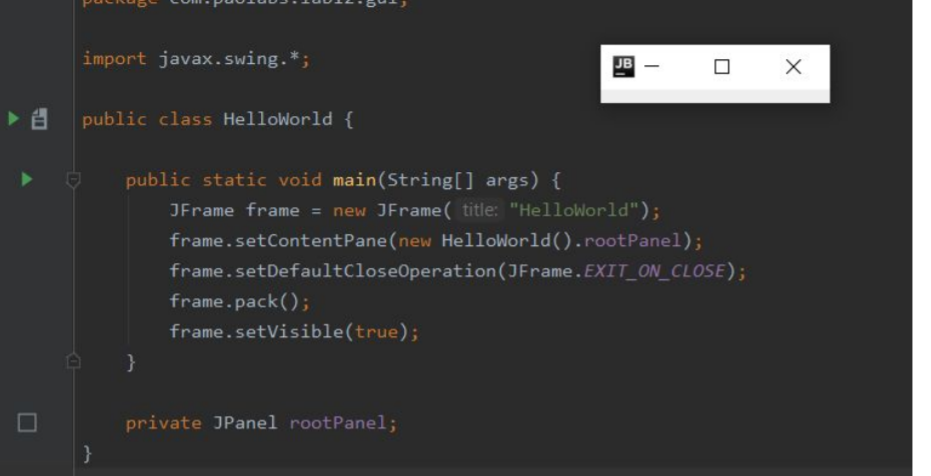
Navigate to the HelloWorld.java file and ensure that you can now see the jPanel as a variable in the HelloWorld class as follows:



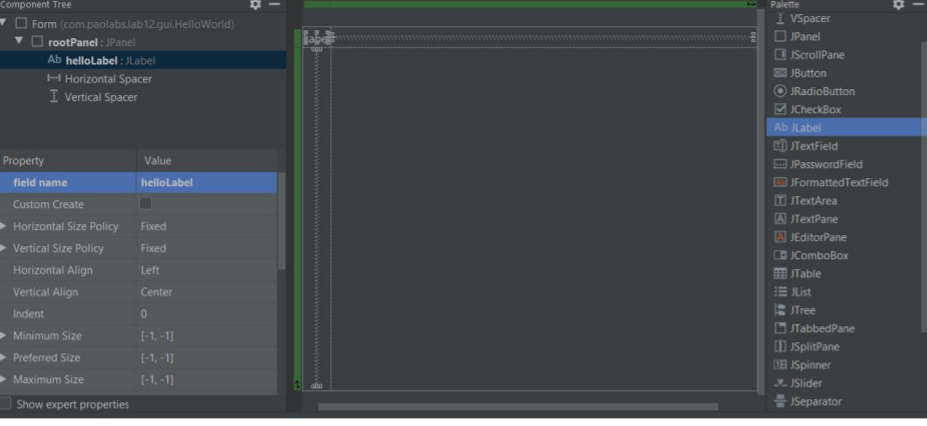
Try to run HelloWorld.main():



When running the code, as the JPanel has no content, you will only see the title bar of an application, with no contents:

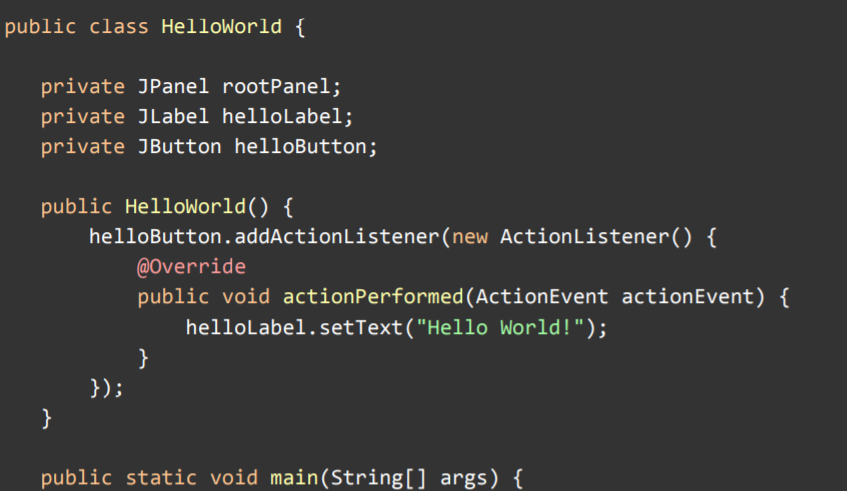


Next, drag a JLabel onto it at the top of the Window:

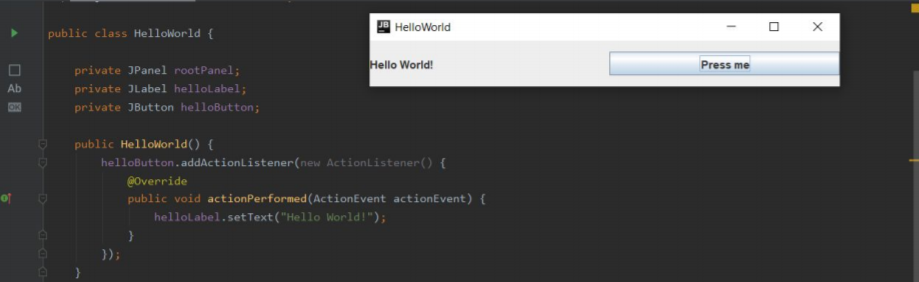


Select the label in the Component tree and set it to have a field name of helloLabel. Add a JButton below the label and set its field name to helloButton.

Right-click on the button and choose 'Create Listener' from the menu, then from the dialogue box that appears, choose 'ActionListener'. This creates a method in the code which will be called when a user clicks on the button



Result:



Resources:

* <https://www.javatpoint.com/java-swing>