|  |
| --- |
| **Department of Software Engineering**  **Mehran University of Engineering and Technology, Jamshoro** |

|  |  |  |  |
| --- | --- | --- | --- |
| **Course: SWE121 – Object Oriented Programming** | | | |
| **Instructor** | Mr. Asmatullah | **Practical/Lab No.** | 13 |
| **Date** | 26-09-2022 | **CLOs** | CLO-3 |
| **Signature** |  | **Assessment Score** | 1 Marks |

|  |  |
| --- | --- |
| **Topic** | **Java Database Connectivity** |
| **Objectives** | * To learn about Java Database Connectivity (JDBC) Java API and various JDBC driver types. * To learn how to use JDBC API to develop database applications * Performing Basic JDBC operations such as Submitting SQL Queries |

|  |
| --- |
| **Lab Discussion: Theoretical concepts and Procedural steps** |

**Tools:**  Java Development Kit, Text Pad, Netbeans, Eclipse

**Theory**

Outline

* Explain field, record, file and database.
* Describe JDBC API and JDBC drivers
* Explain practical steps to connect to and retrieving data from a relational database with JDBC.
* Writing java database applications.

JDBC stands for **J**ava **D**ata**b**ase **C**onnectivity, which is a standard Java API for database-independent connectivity between the Java programming language and a wide range of databases.

The JDBC library includes APIs for each of the tasks mentioned below that are commonly associated with database usage.

* Making a connection to a database.
* Creating SQL or MySQL statements.
* Executing SQL or MySQL queries in the database.
* Viewing & Modifying the resulting records.

## Applications of JDBC

Fundamentally, JDBC is a specification that provides a complete set of interfaces that allows for portable access to an underlying database. Java can be used to write different types of executables, such as −

* Java Applications
* Java Applets
* Java Servlets
* Java ServerPages (JSPs)
* Enterprise JavaBeans (EJBs).

All of these different executables are able to use a JDBC driver to access a database, and take advantage of the stored data.

JDBC provides the same capabilities as ODBC, allowing Java programs to contain database-independent code.

## Pre-Requisite

Before moving further, you need to have a good understanding of the following two subjects −

* [Core JAVA Programming](https://www.tutorialspoint.com/java/index.htm)
* [SQL or MySQL Database](https://www.tutorialspoint.com/mysql/index.htm)

## JDBC Architecture

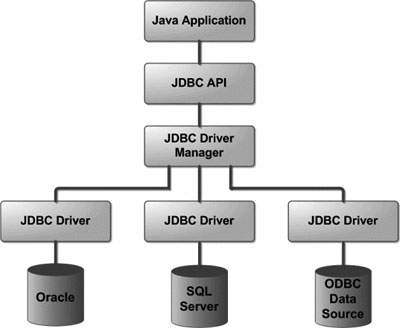
The JDBC API supports both two-tier and three-tier processing models for database access but in general, JDBC Architecture consists of two layers −

* **JDBC API:** This provides the application-to-JDBC Manager connection.
* **JDBC Driver API:** This supports the JDBC Manager-to-Driver Connection.

The JDBC API uses a driver manager and database-specific drivers to provide transparent connectivity to heterogeneous databases.

The JDBC driver manager ensures that the correct driver is used to access each data source. The driver manager is capable of supporting multiple concurrent drivers connected to multiple heterogeneous databases.

Following is the architectural diagram, which shows the location of the driver manager with respect to the JDBC drivers and the Java application −



## Common JDBC Components

The JDBC API provides the following interfaces and classes −

* **DriverManager:** This class manages a list of database drivers. Matches connection requests from the java application with the proper database driver using communication sub protocol. The first driver that recognizes a certain subprotocol under JDBC will be used to establish a database Connection.
* **Driver:** This interface handles the communications with the database server. You will interact directly with Driver objects very rarely. Instead, you use DriverManager objects, which manages objects of this type. It also abstracts the details associated with working with Driver objects.
* **Connection:** This interface with all methods for contacting a database. The connection object represents communication context, i.e., all communication with database is through connection object only.
* **Statement:** You use objects created from this interface to submit the SQL statements to the database. Some derived interfaces accept parameters in addition to executing stored procedures.
* **ResultSet:** These objects hold data retrieved from a database after you execute an SQL query using Statement objects. It acts as an iterator to allow you to move through its data.
* **SQLException:** This class handles any errors that occur in a database application.

ollowing the steps shown below. We assume that you are working on a Windows platform.

## Install Java

Install J2SE Development Kit 5.0 (JDK 5.0) from [Java Official Site](http://java.sun.com/j2se/1.5.0/download.jsp).

Make sure following environment variables are set as described below −

* **JAVA\_HOME:** This environment variable should point to the directory where you installed the JDK, e.g. C:\Program Files\Java\jdk1.5.0.
* **CLASSPATH:** This environment variable should have appropriate paths set, e.g. C:\Program Files\Java\jdk1.5.0\_20\jre\lib.
* **PATH:** This environment variable should point to appropriate JRE bin, e.g. C:\Program Files\Java\jre1.5.0\_20\bin.

It is possible you have these variable set already, but just to make sure here's how to check.

* Go to the control panel and double-click on System. If you are a Windows XP user, it is possible you have to open Performance and Maintenance before you will see the System icon.
* Go to the Advanced tab and click on the Environment Variables.
* Now check if all the above mentioned variables are set properly.

You automatically get both JDBC packages **java.sql** and **javax.sql,** when you install J2SE Development Kit 5.0 (JDK 5.0).

## Install Database

The most important thing you will need, of course is an actual running database with a table that you can query and modify.

Install a database that is most suitable for you. You can have plenty of choices and most common are −

* **MySQL DB:** MySQL is an open source database. You can download it from [MySQL Official Site](http://dev.mysql.com/downloads/mysql). We recommend downloading the full Windows installation.

In addition, download and install [MySQL Administrator](http://dev.mysql.com/downloads/gui-tools/) as well as [MySQL Query Browser.](http://dev.mysql.com/downloads/gui-tools/) These are GUI based tools that will make your development much easier.

Finally, download and unzip [MySQL Connector/J](http://dev.mysql.com/downloads/connector/j/3.1.html) (the MySQL JDBC driver) in a convenient directory. For the purpose of this tutorial we will assume that you have installed the driver at C:\Program Files\MySQL\mysql-connector-java-5.1.8.

Accordingly, set CLASSPATH variable to C:\Program Files\MySQL\mysql-connector-java-5.1.8\mysql-connector-java-5.1.8-bin.jar. Your driver version may vary based on your installation.

To connect Java application with the MySQL database, we need to follow 5 following steps.

In this example we are using MySql as the database. So we need to know following informations for the mysql database:

1. **Driver class:**The driver class for the mysql database is **com.mysql.jdbc.Driver**.
2. **Connection URL:**The connection URL for the mysql database 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. We may use any database, in such case, we need to replace the sonoo with our database name.
3. **Username:**The default username for the mysql database is **root**.
4. **Password:**It is the password given by the user at the time of installing the mysql database. In this example, we are going to use root as the password.

Let's first create a table in the mysql database, but before creating table, we need to create database first.

1. create database sonoo;
2. use sonoo;
3. create table emp(id **int**(10),name varchar(40),age **int**(3));

### **Example to Connect Java Application with mysql database**

In this example, sonoo is the database name, root is the username and password both.

1. **import** java.sql.\*;
2. **class** MysqlCon{
3. **public** **static** **void** main(String args[]){
4. **try**{
5. Class.forName("com.mysql.jdbc.Driver");
6. Connection con=DriverManager.getConnection(
7. "jdbc:mysql://localhost:3306/sonoo","root","root");
8. //here sonoo is database name, root is username and password
9. Statement stmt=con.createStatement();
10. ResultSet rs=stmt.executeQuery("select \* from emp");
11. **while**(rs.next())
12. System.out.println(rs.getInt(1)+"  "+rs.getString(2)+"  "+rs.getString(3));
13. con.close();
14. }**catch**(Exception e){ System.out.println(e);}
15. }
16. }

The above example will fetch all the records of emp table.

|  |
| --- |
| **Lab Tasks** |

1. Write a JDBC application with the previous GUI by adding another button named database. The Fields of GUI application includes.

1. Reg. No. 2. Name 3. Batch 4. Section

5. Gender (must be a choice object) eg Radio Button

6. Qualification (must be a choice object) eg Check Box

7. Address (must have multiple lines) eg TextArea

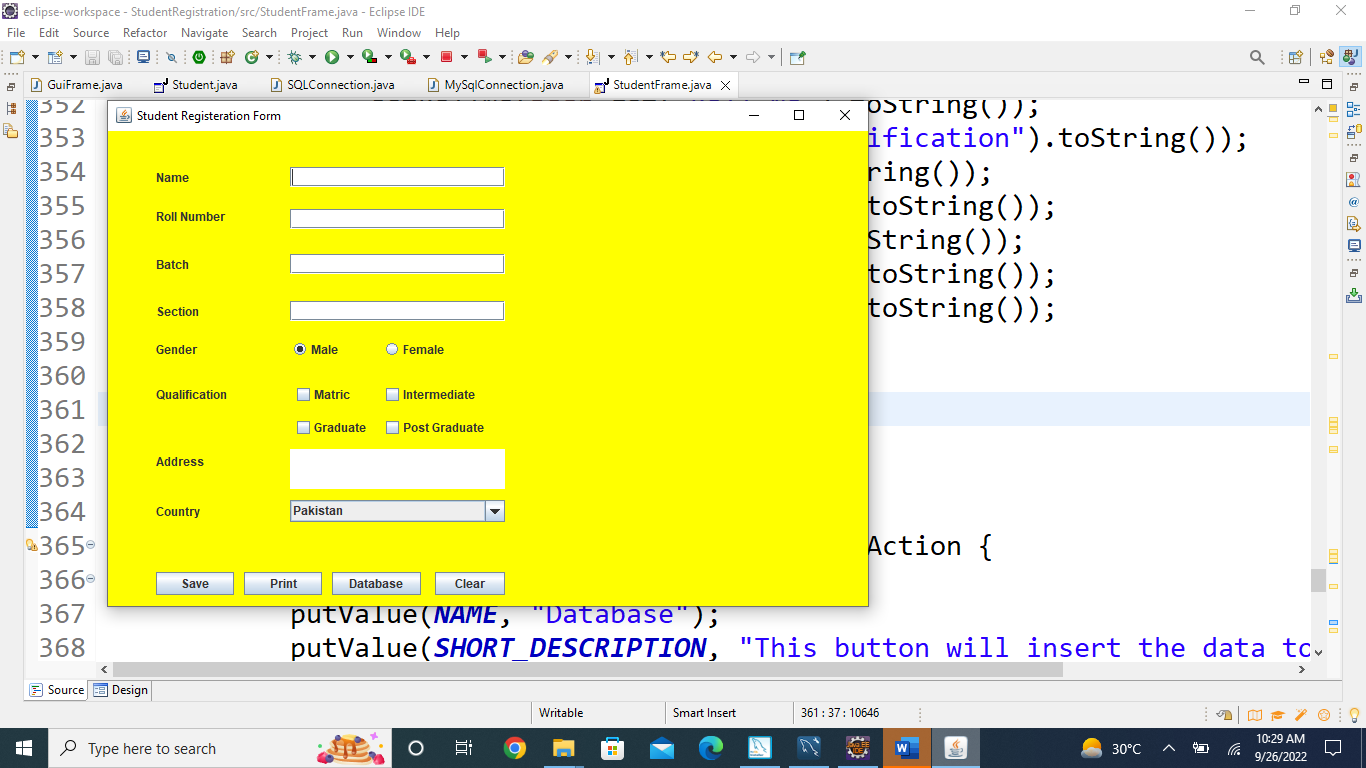
8. Country (must be list item) eg ComboBox

This application must have 4 buttons on the screen 1st one is for Save the record to JSON file and 2nd for Display, and 3rd one for inserting the record into database. And fourth to fetch from database and display.

Create a colorful GUI by adding background color or image to buttons and frame.

This task contain 4 Marks. Two to complete task with JSON file and colorfull GUI and 1 to upload it on github. And 1 for database implementation.

The GUI mus look like following screenshot.



Graphical user interface, chart

Description automatically generated